Universidad
Rey Juan Carlos

TESIS DOCTORAL

Variational and Deep Learning Methods in Computer Vision

Autor:
Iván Ramírez Díaz

Directores:
Dr. Juan José Pantrigo Fernández
Dr. Emanuele Schiavi

Programa de Doctorado en
TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
Escuela Internacional de Doctorado

2019

A mis abuelos. A mis padres. A mi familia.

Cocorín en la quimera
perdió el sombrero,
y lo tienen los niños
de los herreros;
Bocanegra le dice
Cocorín toma el sombrero
y Cocorín le contesta
tómalo, no lo quiero
porque tengo cinco duros
para comprarme otro nuevo.

This is my fear
as if I am nothing
who pretends all the time
to be somebody,
and has to be hyperactive all the time,
just to fascinate people enough
so that they don't notice
that there is nothing.
Slavoj Žižek

Abstract

Computer Vision is a field that aims to simulate the human visual system. In the last decade, with the continuous emergence of multimedia data and applications, there has been an increasing interest to exploit all this available information, which mainly consists in images and videos. Classic approaches to Computer Vision problems constitute a *Bag of Tricks* that have been useful for many years. With the irruption of Deep Learning, most of these techniques, all of a sudden, became old. The reasons are the impressive outperforming results of Deep Learning techniques that, taking advantage of the available data, provide an end-to-end solution that is nowadays easy to use, even for non experts users. Surprisingly, some Variational Methods, which can be considered as classical methods in Computer Vision, survived and maintained the state of the art leading in some specific tasks: Medical Imaging Registration for instance.

The impact of Deep Learning applications in society is undeniable. Moreover, the profit of automatizing many processes and tedious tasks that are still nowadays realized by humans, should be taken as good news, since it would provide more free time for people... and thus, more time to live. The dark side of such automatization will rely on how this new techniques, framed in the Artificial Intelligence field, are democratized across society. This is, how useful in practice are those new emerging tools and who has access to them.

Autonomous driving, Medical Imaging, earthquakes and pollution predictions are few examples of critical application fields where being inaccurate implies disastrous consequences. In such scenarios, classic approaches in Computer Vision, provides less uncertainty in outcomes. In this sense, classical methods are more robust, in particular Variational Methods which have a deep and strong mathematical foundations. Moreover, recently, adversarial attacks on Neural Networks have shown how easy is to fool Deep Learning systems, increasing skepticism for potential Deep Learning users as Medical experts.

In this thesis we address Computer Vision problems in real scenarios from two perspectives with the usage of: (1) Variational Methods and (2) Deep Learning techniques. The former is a powerful tool that gives an extraordinary control over the expected outcomes with very accurate results if some hyper-parameterization is carried out properly. However, this required (usually manual) hyper-parameterization constitutes a huge shortcoming in practice, and a limitation for a wide use by non experts. The later relies mainly on data and solves, until a certain point, an high-dimensional interpolation problem with astonishing results that, however, are sometimes unpredictable (and thus dangerous) when unseen data from different distribution is tested (extrapolation).

To this end:

1. We start using Variational Methods to solve a Saliency detection problem that leads to a nearly binary image (segmentation) through a novel non local non convex variational model. Such method is applied to Magnetic Resonance images, where the goal is to detect and segment tumoral tissues. Then, the hyper-parameterization of such a model is addressed using Deep Learning. To this end, the numerical resolution is re-interpreted as extra layers embedded in a global Neural Network architecture. This constitute a first attempt to combine Variational Methods with Deep Learning. Consequently, drawbacks of a technique can be circumvent by the goods of the other.

2. Then, a Deep Learning method is considered to face an image classification task for automatically recognize public dumpsters. The difficulty of such a task, out from a theoretical viewpoint and/or laboratory, is the lack of data. Deep Learning is data hungry. Acquiring all these data is often an expensive investment out of reach for many companies or public entities, even more when data must be structured and labeled for supervised learning. We propose a semi-automatic method for selecting appropriate images candidates to leverage the manual labelling procedure and reach top performance results. We also show that the predictions uncertainty may be address in order to improve the robustness of such Deep Neural Networks.

3. With the acquired experience of Variational and Deep Learning methods, we derive both techniques from a more general framework known as Bayesian Inference, showing that the majority of novel techniques that succeed in one domain can be explained trough this perspective. In fact, Variational Methods and, Deep Learning or Machine Learning, are two sides of the same coin. To test the power of such generalist methodology, we consider a very ill-posed problem: 3D Human Pose Estimation from 2D Images. Using recent Deep Learning architectures as Capsule Networks and novel approaches as Bayesian Deep Learning, we propose a simple end-to-end Bayesian Capsule Network. This proposal makes use of Deep Learning techniques and Variational Inference to reach state of the art results while keeping a general purpose approach.

Variational and Deep Learning methods are shown to be very powerful and performing tools. However, both have several drawbacks that limit their usage. In the case of Variational methods, despite the very accurate results they provide, the need of an optimal hyper-parameterization to achieve those performances makes them impracticable. On the other hand, Deep Learning methods manage to avoid this inconvenient relying on data but sacrifices robustness. The general results show that, by combining both methods, it is possible to keep accurate predictions and robustness. Finally, as a consequence of our research and results, we conclude that in the future, Variational and Deep Learning Methods in Computer Vision are condemned to get along. The same applies for experts in both fields.

Resumen

Antecedentes La Visión Artificial tiene como objetivo simular el sistema de visión humano. En la última década, el contínuo crecimiento del mundo multimedia ha despertado mucho interés por hacer uso de tan ingente cantidad de datos, en gran parte, imagenes y vídeos. Las herramientas y técnicas clásicas usadas en Visión Artificial constituyen un *Bag of Tricks* con el que muchos de los problemas de visión han sido resueltos con éxito hasta cierto punto. Con la irrupción del Aprendizaje Profundo (Deep Learning), la mayoría de las técnicas hasta entonces punteras, de repente envejecieron. La razón estriba en los impresionantes resultados obtenidos por estas nuevas técnicas que hacen uso de muchos de esos datos ahora disponibles. Además, gracias al desarrollo de herramientas software con soporte computacional, se hace fácil y accesible su uso incluso para usuarios no expertos. Sorprendentemente, algunas técnicas clásicas, como los Métodos Variacionales, sobrevivieron manteniendo el liderazgo en el estado del arte de algunos campos específicos, por ejemplo, en el Registro de Imagen Médica.

Los Métodos Variacionales están presentes en muchos ámbitos de la ciencia e ingeniería como la mecánica de fluidos, electromagnetismo, o la física cuántica. Además, consituye la base teórica de muchos de los métodos de resolución numérica usados hoy en día. En el ámbito de la Visión Artificial, estos métodos permiten resolver problemas tales como la eliminación de ruido (*denoising*), segmentación de imágenes, restauración (*inpainting*) y muchos más.

El Aprendizaje Profundo en cambio, es una técnica particular dentro del campo del Aprendizaje Automático (Machine Learning) que consiste en la extracción y jerarquización de características de imágenes o datos en general. Esta técnica ha permitido recientemente dar solución a problemas de Visión Artificial que hasta entonces eran inabordables o muy difíciles. A pesar de que dichas técnicas fueron descubiertas décadas atrás, fue gracias al desarrollo de procesadores gráficos y el aumento de la potencia computacional que empezaron a usarse ampliamente.

El impacto del Aprendizaje Profundo en la sociedad es innegable. La posibilidad de automatización de tediosos procesos que hasta ahora han sido realizados por personas debería constituir un avance social, pues se traduciría en más tiempo libre para, simplemente, vivir. Estos avances están sujetos a la democratización de dichas técnicas, y por lo tanto al acceso a ellas.

Hipótesis y Objetivos La conducción automática, el análisis de imagen médica, la predicción de terremotos o de contaminación son algunos ejemplos de aplicaciones críticas donde resultados imprecisos pueden acarrear consecuencias desastrosas. En muchos escenarios, los enfoques clásicos de Visión Artificial proporcionan menos incertidumbre en estos resultados. En este sentido, los métodos clásicos son más robustos, en particular los Métodos Variacionales que tienen profundas y sólidas bases matemáticas. Además, recientemente se ha mostrado cómo crear simples ataques adversarios a Redes Neuronales y sistemas de Deep Learning, lo que incrementa el escepticismo en usuarios potenciales como profesionales de la medicina.

En resumen, el Aprendizaje Profundo ha conseguido desbancar a una gran cantidad de métodos clásicos aplicados a la Visión Artificial. Sin embargo, en algunas tareas particulares, los Métodos Variacionales, siguen dando resultados mejores y más fiables. La hipótesis central de esta tesis se traduce en dar respuesta a las siguientes preguntas:

1. ¿Es posible combinar ambas técnicas y mejorar así los resultados del estado del arte en Visión Artificial?

2. ¿Podemos formular una metodología general que explique y modele ambas técnicas?

En esta tesis nos enfrentamos a problemas de Visión Artificial en escenarios reales desde dos perspectivas: el uso de (1) Métodos Variacionales y (2) técnicas de Aprendizaje Profundo. Los primeros, permiten un control extraordinario sobre las soluciones que provee con resultados muy precisos siempre y cuando se lleve a cabo una correcta hiper-parametrización. De hecho, este requisito de hiper-parametrización (habitualmente manual) es la principal limitación para externder su uso a usuarios no expertos. Los segundos, dependen de la disponibilidad de una gran cantidad de datos para el aprendizaje de los parámetros de la red y resuelven, hasta cierto punto, un problema de interpolación en alta dimensión con resultados sin precedentes que, sin embargo, son impredecibles (por tanto peligrosos) cuando se evalúan nuevos datos que no pertenecen a la distribución de datos de entrenamiento, es decir, en la resolución de problemas que requieran extrapolación.

Los principales objetivos fijados en este trabajo son:

1. Analizar los Métodos Variacionales y de Aprendizaje Profundo para identificar los elementos comunes y diferenciales.

2. Combinar estas dos técnicas de manera sinérgica dando lugar a métodos híbridos que mejores el rendimiento en problemas de Visión Artificial.

3. Proponer una metodología general que integre ambos métodos.

Metodología En primer lugar, hemos aplicado Métodos Variacionales para resolver un problema de detección de Saliencia que resulta en imágenes casi binarias (segmentación). En concreto, mediante un modelo variacional no local y no convexo que se ha aplicado la segmentación de tejidos tumorales en imágenes de Resonancia Magnética. A continuación, el problema de la hiper-parametrización del modelo se aborda con el uso de Aprendizaje Profundo. Para ello, la resolución numérica del modelo se re-interpreta como 'capas' extra embebidas en una Red Neuronal. Con ello, tenemos un primer intento de combinar ambas técnicas de forma que las debilidades de una se soslayen con los virtudes de la otra.

En segundo lugar, desde un enfoque clásico de Aprendizaje Profundo, nos enfrentamos a un problema de clasificación de imágenes, en particular, de contenedores de basura. La dificultad de esta tarea fuera de un entorno teórico reside en la falta de datos. Las técnincas de Aprendizaje Profundo requieren muchos datos que no siempre son fáciles de obtener. Además, en la mayoría de los casos necesitamos que esos datos esten estructurados y etiquetados para un aprendizaje supervisado. Proponemos un método semi-automático para seleccionar imágenes adecuadas, reduciendo así el coste del etiquetado manual y alcanzando altos niveles de precisión. También mostramos que paliar la incertidumbre en las predicciones que caracterizan a las Redes Neuronales se traduce en una mayor robustez del sistema.

Por último, con la experiencia adquirida en Métodos Variacionales y de Aprendizaje Profundo, derivamos ambas técnicas desde un marco más general conocido como Inferencia Bayesiana, mostrando que la mayoría de técnicas novedosas que tienen éxito en Aprendizaje Profundo se pueden explicar de forma fundamentada desde esta perspectiva. De hecho, los Métodos Variacionales y, el Aprendizaje Profundo y Aprendizaje Automático, son dos caras de una misma moneda. A tal fin, consideramos resolver un problema muy mal planteado: Estimación de Pose Humana 3D a partir de Imágenes 2D. Haciendo uso de arquitecturas recientes en Aprendizaje Profundo como son las Redes de Cápsulas, y nuevas propuestas de Aprendizaje Profundo Bayesiano, proponemos una Red de Cápsulas Bayesiana (BCN) extremo a extremo. Dicha arquitectura hace uso de técnicas de Aprendizaje Profundo e Inferencia Variacional alcanzando resultados del estado del arte mientras mantiene un enfoque de propósito general.

Resultados Los Métodos Variacionales y de Aprendizaje Profundo han desmostrado ser unas poderosas herramientas. Sin embargo, ambas presentan limitaciones en su uso. Por un lado, los Métodos Variacionales, a pesar de los precisos resultados que proveen, requieren de una hiper-parametrización que los hace a veces impracticables. Por otro lado, los Métodos de Aprendizaje Profundo son capaces de soslayar este problema haciendo uso de multitud de datos y ejemplos disponibles. En contrapartida, estos métodos aumentan la incertidumbre de los resultados reduciendo la robustez del sistema. Los resultados generales muestran que, con la combinación de ambos métodos, es posible mantener tanto la precisión en las predicciones como la robustez del sistema. En concreto:

En el problema de deteción de saliencia, los resultados mejoran un 30,8% en términos de Dice cuando se combinan ambos métodos. Es decir, de un valor Dice de 0,655 en el modelo variacional (TVS), a un 0,857 en el modelo combinado propuesto (TVS+CNN).

Los dos métodos semi-automáticos para el entrenamiento de una Red Neuronal Convolucional (CNN) de classificacion de imágenes permiten reducir el número de ejemplos necesarios y seleccionar los mejores para incrementar el rendimiento y reducir a la vez la incertidumbre en las predicciones. En concreto, el número de imágenes de entrenamiento se reduce a un 2,28% del total. Los métodos propuestos mejoran un 25% y 37,5% la precisión con respecto al margen de mejora dado por el baseline y una CNN entrenada con el 80% de los datos.

La arquitectura propuesta para la detección de pose 3D, Bayesian Capsule Network (BCN), alcanza resultados del estado del arte, 87,22 mm de error promedio por articulación, con una desviación típica, sin embargo, de 17,15 mm que constituyen una reducción de un 52,83% con respecto al segundo mejor candidato.

Conclusiones Finalmente, como consecuencia de esta investigación y de sus resultados concluímos que:

1. Ambos métodos considerados en esta tesis (Variacional y Aprendizaje Profundo) pueden ser combinados obteniendo mejoras en los resultados de los problemas resueltos.
2. Ambas técnicas forman parte de una metodología más general (Inferencia Bayesiana) de la cual pueden derivarse individualmente.
3. El uso de dicha metodología permite mejorar los resultados aprovechando las propiedades de ambos métodos.

Por último, con vistas al futuro, observamos que los Métodos Variacionales y de Aprendizaje Profundo están condenados a entenderse en el campo de la Visión Artificial. Lo mismo se aplica a expertos de dicho campo.

Agradecimientos

Quiero agradecer a mis directores de tesis la confianza y el trato que me han dado. Esta nos es más que otra etapa en la que me ha tocado aprender. En concreto, aprender a investigar, a buscar información, a ordenar ideas y plasmarlas en papel. Para que un niño aprenda a ser adulto, hay que tratarlo como tal. Supongo que esto también es generalizable, en la línea de la tesis, a otras edades y aprendizajes. Gracias por tratarme como a uno más del grupo.

Gracias Juanjo por crear un ambiente en el que es fácil centrarse en lo más divertido de la ciencia y por la capacidad para desenredar problemas cuando me atasco. Gracias Emanuele por mirar por mí, por el interés contagioso por investigar y la paciencia para enseñar matemáticas. Gracias por eso, y por algo que destaca aún mas, y es que ambos sois muy buenas personas. De hecho, la mayor incertidumbre que me causa el post-doctorado es precisamente no saber si voy a estar rodeado de gente a vuestra altura.

Agradezco también a Antonio por animarme a hacer el doctorado. A Ana y a Gonzalo por la calidez y las colaboraciones. A Alfredo por las ideas que tiene y las que genera. A todos los del departamento de Ciencias de la Computación, a Fran, a David. A Roberto, Alessandra, Pilar, Andrew y todos los de Matemáticas. También al departamento de Médica, que me han adoptado como uno más a pesar de ser un intruso. A Helena, a Mario, a Norberto. Gracias a Esteban por los chistes y canciones de los 80. A Javi por los debates en los que, por supuesto, todos sabemos quién tiene razón.

También a Edu por las colaboraciones y compartir server. A Felipe, que sabe lo que es la ESI, ETSI ahora. A Victoria por compartir las clases de Máster. Y por último y no menos importante, quiero agradecer a Adri por el apoyo, las colaboraciones y, otra vez, por allanar el camino.

Contents

Abstract **VIII**

Resumen **XII**

Agradecimientos **XIII**

List of Figures **XX**

List of Tables **XXIII**

1 Introduction **1**

1.1 What is *Information*? 2

1.2 Variational Methods 3

1.3 Machine Learning 4

1.4 Motivation 4

1.5 Hypothesis 5

1.6 Objectives 6

1.7 Manuscript 6

2 Preliminaries - State of The Art **9**

2.1 Variational Methods 9

2.1.1 Ill-posed inverse problems 10

2.1.2 Regularization 11

2.1.3 Bayesian modeling 12

2.1.4 Tikhonov Regularization in Image Processing 13

2.1.5 Well-posed regularized inverse problems 13

2.1.6 The Euler-Lagrange equations 15

2.1.7 Gradient Descent 17

2.1.8 Advanced numerical method 18

2.2 Machine Learning 18

2.2.1 Problem Statement - Bayesian Approach 19

2.2.2 Regression Problems 19

2.2.3 Classification Problems 21
2.2.4 Deep Learning 23
2.2.5 Regularization 28

3 Modeling - Bayesian Inference 39
3.1 Bayesian Inference 40
3.2 Variational Inference 42
3.3 Variational and Machine Learning Problems as instances of Bayesian Inference 44
3.3.1 Dropout 45

4 P1. Proposed Variational Models for Saliency detection 47
4.1 Introduction: Saliency Detection 47
4.1.1 Saliency modeling 53
4.2 Proposed Methods: NLTVS and TVS+CNN 54
4.2.1 Non Local Total Variation Saliency model(NLTVS) 54
4.2.2 Local Total Variation Saliency model (LSTV) embedded in a CNN 61
4.3 Experimental results and discussions 68
4.3.1 Non local TVS model 70
4.3.2 Local TVS - Deep Variational Framework 74

5 P2. Deep Learning for Dumpster recognition and classification 77
5.1 Introduction: Recognition and classification of Dumpsters 77
5.1.1 CNN based approach for dumpsters classification 79
5.1.2 Transfer Learning 81
5.2 Proposed Methods: Semi-Supervised Learning 81
5.2.1 Baseline method 82
5.2.2 Half-Worst method 82
5.2.3 GMM method 82
5.3 Experimental results and discussions 84

6 P3. BCN for 3D Human Pose Estimation 87
6.1 Introduction: BCNs for 3D Pose Estimation and Advanced Deep Learning (DL) 87
6.1.1 Capsnet architecture for 3D Human Pose 88
6.1.2 Preliminaries 89
6.2 Proposed Methods: Bayesian Capsule Networks 92
6.2.1 CNN with Residual blocks 94
6.2.2 Bayesian Capsnet 95
6.2.3 Estimation and Reconstruction 96
6.3 Experimental results and discussions 96
6.3.1 Human 3.6 Data set and related works 96
6.3.2 Comparison with the State Of The Art 98

7 Conclusions 103
7.1 Practical Conclusions 103

7.2 General Conclusions 104
7.3 Future work 107

Bibliography **111**

8 Appendixes **125**
8.1 Convolution as Matrix Operation 125
8.2 Multi-class Study 126

Glossary

AD Automatic Differentiation. 4, 81

AGI Artificial General Intelligence. 107

AI Definition of Artificial Intelligence. 4

AI Artificial Intelligence. 4, 87, 106, 107

AWGN Additive White Gaussian Noise. 10

BCN Bayesian Capsule Network. 7, 87–89, 95, 98, 107

BFGS Broyden–Fletcher–Goldfarb–Shanno. 38

BNN Bayesian Neural Network. 87–89, 91

CNN Convolutional Neural Networks. XXI, 4, 6, 11, 26–28, 39, 46, 47, 54, 65, 68, 70, 75, 78, 79, 81, 82, 84, 88, 89, 91, 92, 94, 95, 99, 104–106

CV Computer Vision. 4–6, 39, 46, 48, 78, 87, 88, 92, 103, 105, 107

CVPR Conference on Computer Vision and Pattern Recognition. XXIII, 98

DAG Directed Acyclic Graph Model. XX, 41, 42

DBN Deep Bayesian Network. 88

DC Difference of Convex. 61, 62, 65

DL Deep Learning. XV, 4, 9, 36, 38–40, 42, 44, 47–50, 70, 77, 78, 81, 87–89, 91, 92, 105, 106

DNN Deep Neural Network. 25, 27, 45, 87, 88, 91

ELBO Evidence Lower Bound. 43

FCNN Fully Connected Neural Network. 45, 78, 81, 96, 98, 99

FLAIR Fluid Attenuated Inversion Recovery. XXI, 47–49, 64, 75, 103

fMRI Functional Magnetic Resonance Imaging. 104

GDI Generalized Definition of Information. 2

GMDH Group Method of Data Handling. 23

GMM Gaussian Mixtures Model. XXI, 78, 82, 104

GPU Graphical Processor Unit. 27

i.i.d. Independent and Indentically Distribuited. 12, 19

IJCV International Journal of Computer Vision. XXIII, 100

IP Image Processing. 1, 4, 9, 92

KL Kullback-Leibler Divergence. 43, 90

LSTM Long Short Term Memory. 4, 25

MAP Maximun a Posteriori. 12, 19, 21, 89, 91

ML Maximum Likelihood. 40

ML Machine Learning. 1, 9, 19, 28, 42, 77, 78, 81, 87

MLP Multilayer Perceptron. XXI, 67, 68

MR Magnetic Resonance. 6, 10, 47, 48

MRI Magnetic Resonance Imaging. 48, 49, 64, 71

MSE Mean Squared Error. 21

NAG Nesterov Accelerated Gradient. 38

NN Neural Network. 6, 7, 9, 23, 25–28, 40, 42, 45, 46, 48, 49, 65, 67, 77, 88–92, 94, 98, 99, 105–107

PDE Partial Differential Equation. 9, 17

PET Positron Emission Tomography. 48

PSNR Peak Signal Noise Ratio. 15

RNN Recurrent Neural Networks. 4, 25

SAR Synthetic Aperture Radar. 10

SGD Stochastic Gradient Descent. 36, 37, 45, 46

SNARC Stochastic Neural Analog Reinforcement Calculator. 4

SP Signal Processing. 1, 13

SSIM Structural Similarity. 15

SVM Support Vector Machines. 4

TV Total Variation. 14–16, 18, 27, 39, 49, 50, 61, 63, 76, 104–106

List of Figures

1.1 From left to right: a subject process information. In the case of traditional Information Processing, is a human that program through prior methods how the object (information) shall be processed. On the other hand, in Machine Learning, the specific processing that is carried out by a Machine is to learn a particular task. 2

2.1 Direct and Inverse problems schemes. 10
2.2 There exists infinity of combinations of u and η that are mapped to the same fixed f, i.e., there is not injectivity. The dashed line in $\mathcal{N}$ depicts that only a sample $\eta \sim \mathcal{N}(0, \sigma_\eta)$ is drawn, but it remains unknown. 11
2.3 u versus ∇u based *priors* 14
2.4 Hyper-Laplacian potential functions 15
2.5 Neural Network with 3 hidden layers that are composed as: $\mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))$. 23
2.6 A single neuron 24
2.7 First layer in a Neural Network 25
2.8 Activation functions 26
2.9 Toeplitz matrix generated from laplacian kernel 29
2.10 Shallow Feedforward Neural Network 30
2.11 Unwrapped Feedforward Neural Network 31
2.12 Path from the output to a last layer weight. 32
2.13 Paths from the output to a first layer weight 34
2.14 Comparison: Batch, Mini-batch and Stochastic Gradient Descent 36

3.1 DAG Model 41
3.2 Directed Acyclic Graph Model (DAG) ($\mathbf{f}$-parametric) 42
3.3 Dropout example 45

4.1 Saliency example 49
4.2 Saliency terms 53
4.3 Left column: original image. Right column: Output of the algorithm. Parameters: $\lambda = 1$, $\alpha = 1$, $\delta \approx 2$. 65

4.4 Example for natural images. Left column: original image. Right column: Output of the algorithm. Parameters: $\lambda = 1$, $\alpha = 2$, $\delta = 2$. 66

4.5 Left: original image. Middle: background subtraction (input image). Right: Output of the algorithm. Parameters: $\lambda = 20$, $\alpha = 4$, $\delta = 10$. The background subtraction is performed with RGB images. The given input (middle) is converted to a gray-scale image. 66

4.6 Scheme of the proposed Deep Variational Framework. The Convolutional Neural Networks (CNN) provides an adaptive parametrization of the variational model through the $\delta(x)$ function. A Multilayer Perceptron (MLP) allows the estimation of the remaining parameters which are fed to the variational model in a top layer of the global Deep Variational Framework. 68

4.7 Outputs and adaptive $\delta(\mathbf{x})$ computed by the CNN. 69

4.8 Scalar estimated hyper-parameters: λ, α during training. 69

4.9 A comparison between the Yosida's approximating solutions and the hard truncated solutions. In first row: input images, second row are the respective outputs for the r-convergence approximation method and the last row, the hard truncation approximation. Differences in the final solution are colored in red. 71

4.10 Relative differences: $r-$convergence vs hard truncation. Stabilization of the $r \rightarrow 0$ limit. The outer time n-loop, $n = 1$ ... N is considered together with the inner j-loop, $j = 1$ ... J. The jumps from u_5 to u_6, u_{10} to u_{11} etc are caused by the truncation with J fixed to 5 for each n. We see in Figure 4.9 that the (binary) output solution is practically indistinguishable from the almost binary approximations in r. The final outputs only differ a 4,2%, which in practice turns out to be few single pixels. 72

4.11 Time comparison between patch based and kernel based resolution. The surface which remains constant with the quantization levels corresponds to the patch based resolution while the other one corresponds to the kernel based resolution. Using a kernel based approach allows a nearly invariant dependency w.r.t. the size of ρ (radius of the kernel), which in turn promotes the non-locality effect. 72

4.12 From left to right: 2D reconstruction (our proposed model applied slice by slice), 3D reconstruction (our proposed model applied to the whole volume), Ground-truth. A significant number of false-positives is reduced when the fully 3D finite difference schemes are applied. 73

4.13 Distribution of the mean of the tumor intensities obtained from the ground-truth provided in the BRATS2015 dataset. The linear regression mapping the brain intensity mean μ_{brain} to the tumor intensity mean μ_{tumor}. 74

4.14 From left to right columns: input Fluid Attenuated Inversion Recovery (FLAIR)-glsmri images, TVS, CNN, CNN+TVS and Ground-truth segmentation. Notice the output of the tree settings are nearly binary, thus a simple thresholding (0.5 in the range [0,1]) is applied in order to get binary segmentations. 75

5.1 (a)-(g) A sample of each dumpster class, and (h) the distribution of the whole data set provided. 80

5.2 Sample of the diversity of dumpsters in Class C_4. 80

5.3 The three panels show the distribution of the confidence on the labeling after the first round, i.e. training with a manual selection 30 images × 7 folders. The left and central panels also show the search region S_{HW} (left) and S_{GM} (central). The right panel depicts the two gaussians of the Gaussian Mixtures Model (GMM) that models the distribution. 82

5.4 Experiment results. (Left) Accuracy at the end of the three rounds of every method (baseline, Half-Worst and 2-GMM) repeated 10 times. (Right) Mean and standard deviation of the accuracy as the number of training images increases at each round. 85

5.5 Other performance measures, split in the 7 different classes for panels (a) to (c). Specifically, (a) True Positive Ratio per Class (b) False Positive Ratio per Class (c) ROC per Class. In addition panel (d) shows the averaged ROC curve of every method. 86

6.1 CapsNet module. Capsules are created reshaping the feature maps incoming from a previous NN. Then each one is mapped K times into a higher dimensional space. The results are combined and squashed into the outgoing capsules. The block with the red line is the squash function. In this Figure $M = N = D = 16$, $J = 512$, $S = 8$, $S' = 16$ and $K = 17$. Dropout is proposed here as part of the Bayesian formulation. 93

6.2 Left: 2D - 3D analogy. Right: retro-projection and inverted projection pipeline. 93

6.3 Error bounds on projection and retro-projection steps. 94

6.4 Global structure of the proposed CapsNet-based Bayesian Neural Network for the Human3.6 challenge. The input image is first analyzed through a CNN with Residual blocks. The feature maps are sent into a Bayesian CapsNet as defined in this paper. The CapsNet output provides encoded vectorial features that are decoded in three different versions of the same concept: 2D and 3D coordinates together with Joint heat maps. The loss is a self-balanced combination of the loss from each task plus the one due to the prior on the Capsnet. 95

6.5 (a) An input RGB image from the validation set and its 2D predicted coordinates. (b) 3D predicted joints together with the inner epistemic uncertainty (the circle magnitude). (c) Ground truth for the joint heat maps. (d) Joint heat maps estimated. Each joint is depicted with a different color. 97

6.6 Some results of our proposed Bayesian Capsule Network in a test subject. (Left) The 2D RGB image input, (middle) the 3D predicted Human Pose, and (right)the given ground-truth. 3D skeletons have different viewpoints for the sake of visualization, but all of them have the same orientation. 101

List of Tables

4.1 Results of a 3D and 2D resolution with the same parametrization and $p = 0.5$ 71

4.2 Results obtained for different p values using the whole BRATS2015 data-set. Metrics are defined in the first row where: tp-true positives, tn-true negatives, fp-false positives, fn-false negatives. The Dice measure which accounts for a compromise betweeen Precision and Recall is improving a 5.28% and a 2.62% when p takes values $p = 1$ and $p = 0.5$, respectively. . . 74

4.3 Summary of results of the considered methods for brain tumor segmentation. 75

6.1 State of the art works that have used the Human3.6 dataset. 97

6.2 Comparison of three versions of the Bayesian Capsule Networks with respect to the top-3 in Conference on Computer Vision and Pattern Recognition (CVPR)'17 Human 3.6 challenge, due to Zhou [Zhou et al., 2016], Tekin [Tekin et al., 2017] and Tome [Tome et al., 2017]. The background bars are sized with respect to the max. and min. of every row 98

6.3 Comparison of three versions of the Bayesian Capsule Networks with respect to the top-3 in International Journal of Computer Vision (IJCV)'18, due to Sanzari [Sanzari et al., 2016], Bogo [Bogo et al., 2016] and Katircioglu [Katircioglu et al., 2018]. The background bars are sized with respect to the max. and min. of every row. 100

List of Algorithms

2.1 AdaGrad 37
2.2 RMSProp 37
2.3 Adam 38

4.1 DC Primal-Dual algorithm 64

5.1 HW-Method 83
5.2 GMM-Method 83

6.1 Capsule Network 92

CHAPTER 1

Introduction

Mare, llévame al colegio
a educarme la memoria,
mira que no quiero soñar
con el burro de la noria [...]
Cante flamenco

Image Processing (IP) is a sub-field of a wider one referred to as Signal Processing (SP), which in turn belongs to the Mathematics, Information and Electrical Engineering domain. In SP, a signal is understood as a function that carries information. Besides, Information processing is the task of changing or modify (process) information with a particular objective. The common factor throughout these fields and sub-fields is the concept of information, which plays the role of the main object at each discipline. The definition of information has evolved and may differ a lot from a context to another, and yet there is not a consensus that satisfies the wide range of uses for such a word.

It is also the case for a relatively new field embedded in the so-called domain of Artificial Intelligence, known as Machine Learning (ML). Learning is defined as the process of acquiring knowledge or information. Thus, Machine Learning is the task of acquiring (process) knowledge (information) performed by a machine. Comparing to SP or Information Processing, the main difference with respect to Machine Learning in a first approach is concerned to who or what is carrying out the task. To be more precise, since in each case the solutions come out from an algorithm, we can add that the difference is if the rules that govern the algorithm are pre-defined by human or, on the contrary, are inferred/learned automatically.

In any case, again, the key-point is what is meant by *information*. In words of Pearl and Mackenzie:

> *My emphasis on language also comes from a deep conviction that language shapes our thoughts. You cannot answer a question that you cannot ask, and you cannot ask a question that you have no words for.* [Pearl and Mackenzie, 2018]

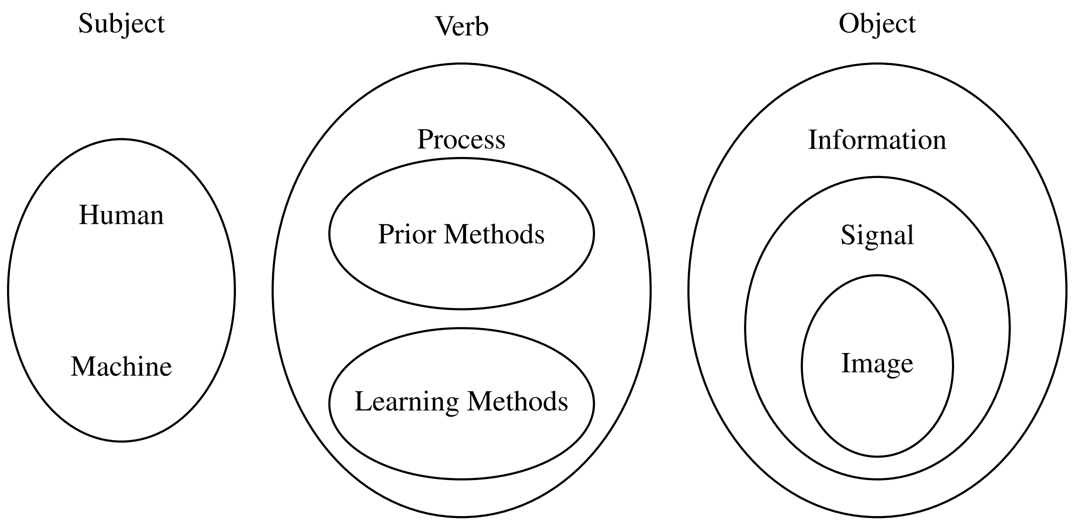


Figure 1.1: From left to right: a subject process information. In the case of traditional Information Processing, is a human that program through prior methods how the object (information) shall be processed. On the other hand, in Machine Learning, the specific processing that is carried out by a Machine is to learn a particular task.

1.1 What is *Information*?

In [Capurro and Hjørland, 2003], the authors provide a review of the use of the term, and how it has been interpreted from the ancient Greeks to the present days and show the importance of going back to the roots to have a better understanding of what ideas hide behind the concept of *information*:

> *[A] word never–well, hardly ever–shakes off its etymology and its formation. In spite of all changes in and extensions of and additions to its meanings, and indeed rather pervading and governing these, there will still persist the old idea. [...] Going back into the history of a word, very often into Latin, we come back pretty commonly to pictures or models of how things happen or are done. [Austin, 1961]*

Following this recommendation, and looking in the etymology of the work, *information* may be composed by the prefix *in* from Latin, meaning to *generate something inward (from outside)*; and *formatio* (Latin noun, f, genitive *formationis*), in the sense of *give form to, instruct, educate*. If we take a look in the Asian culture however, we will notice that Chinese is a language that has maintained a strong dialectic component, which lead us to a sighly different interpretation. In fact, as many other words in traditional Chinese, the word *information* is written with two characters: 信息. The first separately means "letter" while the second means "interest". This contrast of ideas advances us the distinction between **content** and **container** that we will have to make later.

From a mathematical point of view, however, there is a lack of consensus on the meaning and definition of *information* which also vary substantially depending on the considered grade of abstraction [Floridi, 2017]. Some efforts have been done from different fields related with Information processing to converge to a Generalized Definition of Information (GDI) [1]. GDI nevertheless relies in turn on the

[1] according to [Floridi, 2017], σ is an instance of information, understood as semantic content, if and only if:
(GDI.1) σ consists of one or more data;
(GDI.2) the data in are well-formed;
(GDI.3) the well-formed data in are meaningful

definition of data, that can also be an ambiguous and polyhedral concept, leading to scientist to use an ad-hoc designation of what is understood by information in each particular task that they carry out.

Consequently, to avoid this tangle of definitions, in Image Processing, we shall assume that an image is itself the information (content) or, alternatively, the container of such information. In that sense, the Image Processing consists of acquiring images or extracting features of interest from those images, which are signals with specific structure and properties that come from a concrete source of information, usually, a camera. Of course, this distinction can also be applied to Machine Learning generalizing for any sort of signal. Machine Learning with respect to Image Processing is more often used to extract information and thus considering the signal as a container.

In other words, sometimes we will focus on separating mixed signals that come from different sources and selecting those of interest. An example of that is *denoising*, which is an Image Processing (or Machine Learning) task that consists of removing the noise added unwittingly to a signal intended to be acquired. Other times, the considered Machine Learning (or Image Processing) task shall aim to obtain high-level semantic features, as in semantic segmentation, e.g., cropping a person in a given picture. The former, an instance of a so-called Restoration Problems family, aim at recovering an original image (or signal) that has been corrupted, while the latter relates to transform a signal (maybe an image) into another space of features where the relevant information is present.

1.2 Variational Methods

Calculus of Variations can trace its history back to the XVII century, precisely with *Newton's minimal resistance problem* in *Philosophiæ naturalis principia mathematica* in 1687 [Newton, 1987], and is concerned with the extrema of a functional (maxima or minima), understood as a function that associates a scalar to another function (function of functions). The field has attracted the attention of brilliant mathematicians to the point the history of Mathematics has been intertwined with that of the Calculus of Variations for over 2 centuries. From Leibniz and Bernoulli bothers to Dirichlet, Hamilton, Hilbert and Jacobi, passing through Euler, Lagrange and Laplace, this field has acquired such prominence with ground-breaking theoretical results that have impacted a wide range of applications in other fields such as Physics and Engineering. Sometimes it is maybe not easy to find, hidden behind emerging fields that seem to be purely genuine, the remains and links to the foundations of the Calculus of Variations, but they are present and drive solutions for quite different problems like in fluid mechanics, electromagnetism, gravitation, quantum mechanics and many others.

As is often the case, we are continuously reinventing the wheel. Nonetheless each time a new problem is re-written in a different way being able to be cast as a particular or a more general case of a previous well-known problem, we are indeed increasing cumulatively the knowledge and the understanding of the matter. On the one hand, from the perspective of what was at the beginning though as something new, it is a step back, in the sense we realize that it was not new, but on the other hand, it is a step forward in an abstract sense, which is an overriding objective in the field of Mathematics.

Image Processing by variational methods is an established field in applied mathematics and computer vision [Morel and Solimini, 1995], [Aubert and Kornprobst, 2006], [Vese and Guyader, 2015] which aims to model typical low and mid level image reconstruction tasks and restoration processes such as denoising, deconvolution, inpainting, segmentation, registration and super-resolution of digital degraded images. Since the mathematical approach of Tikhonov and Arsenin on ill-posed inverse problems [Bell, 1978] and the applied work of Rudin, Osher and Fatemi [Rudin et al., 1992] through their celebrated ROF model, a standing effort to deal with new Image Processing tasks through variational methods has arose.

1.3 Machine Learning

First termed by Arthur Lee Samuel in 1959 at IBM, Machine Learning arises in the context of Computing Gaming and Artificial Intelligence (AI) with the idea of, as we stated before, equipping machines with methods to acquire information (or the required knowledge) to perform specific tasks. This learning procedure is supposed to be achieved using data. Nowadays, the increasing interest of Machine Learning is in large measure due to the huge amount of available data that is stored and generated every day. Despite the undeniable importance of Machine Learning from two decades to the present, it has not always been the case. The first neural network machine Stochastic Neural Analog Reinforcement Calculator (SNARC), with learning capabilities, was created by Marvin Minsky and Dean Edmonds in 1951. In the 1960s Bayesian methods were used in Machine Learning to perform probabilistic inference [Solomonoff, 1964]. A decade after, in the 70s, in the wake of a book published by Minsky and Seymour Papert, *Perceptrons* (1969) [Minsky and Papert, 2017], in where the authors shown the limitations of neural networks, the AI community started to perceive those neural networks as a blind alley for AI research, leading to a so-called AI Winter. Curiously, it was a that time that the precursor of the nowadays well-established *back-propagation* algorithm, the general method for Automatic Differentiation (AD) appeared, proposed by Seppo Linnainmaa [Linnainmaa, 1976]. *Connectionism*, how the field of modern neural networks was called at that time, was about to nearly disappear although important achievements in applying back-propagation to neural networks was reached in 1974 by Paul Werbos [Werbos, 1994]. It is in the middle of the 80s that the back-propagation algorithm was proved to be an efficient an practical way to successfully train neural networks, with the works of David Rumelhart, Geoffrey Hinton and Ronald J. Williams [Rumelhart et al., 1988]. After that, some discovers came up, among others: Random Forest algorithm and Support Vector Machines (SVM) in 1995, Long Short Term Memory (LSTM) in 1997 from Recurrent Neural Networks (RNN) in 1982, followed by the creation, in 2009, of one of the largest image dataset, the ImageNet, at Stanford University. The rise of the field can be considered to have started in 2012, with novel CNN and DL methods, when the state of the art in image classification was greatly improved. From that achievement on, there has been a hype around Machine Learning an Artificial Intelligence, where an important part of the applications focused on the field of Computer Vision (CV) and IP.

1.4 Motivation

Computer Vision is a field that aims to simulate the human visual system. In the last decade, with the continuous emergence of multimedia data and applications, there has been an increasing interest to exploit all this available information, which mainly consists in images and videos. Classic approaches to Computer Vision problems constitute a *Bag of Tricks* that have been useful for many years. With the irruption of Deep Learning, most of these techniques, all of a sudden, became old. The reasons are the impressive outperforming results of Deep Learning techniques that, taking advantage of the available data, provide an end-to-end solution that is nowadays easy to use, even for non experts users. Surprisingly, some

Variational Methods, which can be considered as classical methods in Computer Vision, survived and maintained the state-of-the-art leading in some specific tasks: Medical Imaging Registration for instance.

The impact of Deep Learning applications in society is undeniable. Moreover, the profit of automatizing many processes and tedious tasks that are still nowadays realized by humans, should be taken as good news, since it would provide more free time for people... and thus, more time to live. The dark side of such automatization will rely on how this new techniques, framed in the Artificial Intelligence field, are democratized across society. This is, how useful in practice are those new emerging tools and who has access to them.

Frequently, wide used Deep Learning applications are in some way disappointing. For example, many Deep Learning architectures are built to create facial filters that puts a hat on you or change the color of your eyes. Others, learn to recommend adds, which does not help to get rid of adds, at all.

Leaving aside some personal opinions, there is at least one technical reason (among non technical others) that can explain the success of Deep Learning on non transcendental applications that are not likely to improve our lives. Deep Learning and Neural Networks are still non-reliable tools. In fact, the main drawback of Deep Learning and its supported architectures (Neural Networks) is they act as uninterpretable Black Boxes from which very little knowledge and comprehension can be retrieved. From an engineering point of view, this is not a huge problem as long as everything continues to work. But then, of course, Murphy appears.

Autonomous driving, Medical Imaging, earthquakes and pollution predictions are few examples of critical application fields where being inaccurate implies disastrous consequences. In such scenarios, classic approaches in Computer Vision, provides less uncertainty in outcomes. In this sense, classical methods are more robust, in particular Variational Methods which have a deep and strong mathematical foundations. Moreover, recently, adversarial attacks on Neural Networks have shown how easy is to fool Deep Learning systems, increasing skepticism for potential Deep Learning users as Medical experts.

1.5 Hypothesis

When a ground-breaking method or model is discovered and appears to explain many things that were unanswerable until then, it is a matter of science to find where this model or method fails. Keeping this in mind, the hypothesis of this thesis can be formulated through the following questions:

If Variational and Deep Learning Methods provide the state of the art results in different applications that in turn belong to the same field, i.e., CV, then:

1. Is it possible to combine both methods to outperform the current state of the art?

2. Can we find a general methodology that explains or contents both approaches?

Along this dissertation we will answer these questions and find examples of applications in order to shed light on them, as well as their limitations.

1.6 Objectives

In this thesis we address CV problems in real scenarios from two perspectives with the usage of: (1) Variational Methods and (2) Deep Learning techniques. The former is a powerful tool that gives an extraordinary control over the expected outcomes with very accurate results if some hyper-parameterization is carried out properly. However, this required (usually manual) hyper-parameterization constitutes a huge shortcoming in practice, and a limitation for a wide use by non experts. The later relies mainly on data and solves, until a certain point, an high-dimensional interpolation problem with astonishing results that, however, are sometimes unpredictable (and thus dangerous) when unseen data from different distribution is tested (extrapolation).

With the aim of overcoming these issues, the following objectives are stated:

1. Analyze both methods to determine their common elements and differences.

2. Combine these two methods.

3. Propose or find a general methodology such that each methods are integrated in a principled way.

To this end, different problems in CV shall be considered. First, separately, with the aim of identifying particular issues of both approaches. And second, based on a more general methodology, to face a difficult CV problem, and thus, prove and validate such methodology.

1.7 Manuscript

This manuscript is organized as follows:

In Chapter 2, we briefly revise the basis and some technical background from the state of the art of Variational and Deep Learning methods. Both approaches are derived from the same bayesian framework to provide a common perspective. Variational methods are presented in the context of inverse problems while Deep Learning techniques, in the field of Machine Learning, are formulated, without loss of generality, as direct problems.

In Chapter 3, we present a more general interpretation of the aforementioned approaches, this is, a Bayesian Inference framework. We then show how to derive the Variatonal and Deep Learning methods from it, and relate some of the new and successful techniques in Neural Networks (NNs), such as dropout. Both approaches are also presented as instances of an Approximation Inference, in particular: a Variational Inference method.

In Chapter 4, we propose two variational models for Saliency detection applied to Magnetic Resonance (MR) images. The first, a non local model, is solved trough a gradient descent based algorithm. The second, a local model, is solved using duality arguments. This local version is then embedded in a CNN to create a Deep Variational Framework that allows to an automatic hyper-parameterization of the model. Both methods are tested on BRATS215 dataset [Menze et al., 2014]. This constitutes a first attempt to combine the two main techniques of this thesis.

In Chapter 5, we aim to solve a classification problem based on CNNs. State of the art techniques and novel architectures are used. Along with this, the problem is faced in a real context where data are limited. We propose two methods for a semi-automatic training procedure that deals with the central drawback of NNs, the uncertainty that is not generally taken into account. The positives results motivates the use of Bayesian Inference in Deep Learning from a principled way.

In Chapter 6, we address an ill-posed problem consisting on the recognition of 3D human pose from 2D images. To this end, we make use of the Variational Inference scheme presented in Chapter 3. We propose a Bayesian Capsule Network (BCN) that, despite its simplicity, achieves state of the art results in the 3D pose estimation. This is done using techniques from Deep Learning and Variational methods, which constitutes the second attempt to combined both methods, this time in a principled way.

In Chapter 7, conclusions are drawn from the particular problems addressed in this dissertation, relating concepts from Variational and Deep Learning techniques. Then, general conclusions and discussions are given to summarize the whole presented work and point out new research directions and future work.

CHAPTER 2

Preliminaries - State of The Art

En los mismos ríos entramos y no entramos, somos y no somos.
Heráclito

This chapter is devoted to present the main needed concepts and the state of the art in the research field. Firstly, in Section 2.1 Variational Methods for IP based on degradation models are introduced. Bayesian modeling and other interpretations of a Restoration problem are given and motivated. In Section 2.2, basis of ML are revised as well as we introduce NNs and the DL technique.

2.1 Variational Methods

In the last 30 years there has been a great effort in IP based on Partial Differential Equation (PDE) and Variational Methods. Filtering techniques can be used to palliate the effect of noise in images. In such techniques, physical process than are found in nature are applied to images, e.g., the heat equation. In Variational Methods, the problem is prompted in such a way it allows to control the variations of the solutions, which are the considered images.

We first start showing a general framework based on bayesian modeling. Such approach can be particularized to lead to the Tikhonov regularization method, introduced to give a meaning to ill-posed Inverse problems. When non linear operators are considered, Generalized Tikonov is invoked. Bayesian modeling is an alternative, statistical approach which can converge to the same model but allowing a different interpretation which highlights the rationale behind the Tikhonov method. It was in the seminal Rudin-Osher-Fatemi paper [Rudin et al., 1992] that non linear non smooth operators were introduced. This paved to way to the development and establishment of variational methods and PDE in image processing and Computer Vision. We give a brief review of it in order to give the basis of the development of our models in Chapter 4. Our introduction is by no means exhaustive and should be complemented with relevant literature in the topics. Fundamental references shall be presented all along this manuscript.

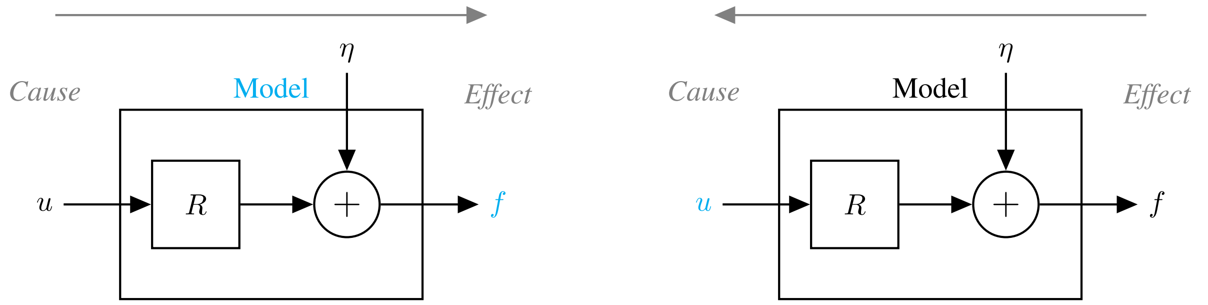
We consider a degradation model characterizing the transformations that the original image has suffered, in order to invert them and recover a restored version of it. It is well established as basic degradation model, the following:

$$f = Ru + \eta, \tag{2.1}$$

where $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ is the ideal image or signal that we want to obtain, R is usually a linear operator that acts on u, f is the degraded image that is acquired and η is an added noise that must be characterized. In Image Processing, this noise is often assumed as Additive White Gaussian Noise (AWGN). However, other kind of noise as Rician noise in MR images, impulse noise (audio signals), speckle noise Synthetic Aperture Radar (SAR) among others. Since u is the target and we are given f, the main goal in a Restoration problem is to invert the transformation, or at least, to find an approximation of such process. In fact, Image Processing problems can be reformulated as:

1. Direct problems: to identify effects from causes. This implies to analyze the considered system and to model it in order to determine an output f, given an input u 2.1a.

2. Inverse problems: to identify causes from effects. Given a model (or making assumptions about it) and an output f, the objective is to determine the input u 2.1b.



(a) Direct Problem. Given u, the targets are: Model and f.

(b) Inverse Problem. Given Model and f, the target is u.

Figure 2.1: Direct and Inverse problems schemes.

A Restoration problem is therefore an Inverse problem since the *information* that we seek is the image itself (cause), and we are given a degraded version by means of a transformation (effect).

2.1.1 Ill-posed inverse problems

A vast majority of Restoration Problems are ill-posed in the sense of Jacques Hadamard [Hadamard, 1902], according to whom a problem is well-posed if the following conditions over the solution are met, namely: (1) existence, (2) uniqueness and (3) stability. Roughly speaking, we are asking if the inverse transformation (solution) exists and is unique, and if the output of such transformation depends continuously on the input.

To illustrate and give a light intuition of why those problems are ill-posed, let us set $R = \mathbb{I}$, where $\mathbb{I}$ is the identity function. Then, the model is simplified to $f = u + \eta$, where $\eta \sim \mathcal{N}(0, \sigma_\eta)$, i.e., a noise addition model (*denoising*). Moreover, $f \sim \mathcal{N}(u, \sigma_\eta)$ is an random variable that follows a normal distribution centered in u. In fact, if for a fixed u we could draw as many samples as we wish from $\mathcal{N}(u, \sigma_\eta)$, then recovering u would be as simple as computing the mean of u.

However, in *denoising* there is only one sample f given, which means in turn there is only one sample of noise η, and therefore for an unknown η there exists infinity of image candidates u that provides the same f. This is illustrated in Figure 2.2.

Adding noise is not the only way of falling in a non invertible problem. The operator can also be non injective. This may be the case for some Restoration Problems such as *deblurring*, *deconvolution*, *dehazing*, *compressed sensing*, *inpainting* among others. This lack of injectivity also makes ill-posed the restoration problem. In addition, even though the problem is well-posed, it may occur that is ill-conditioned in that a small change in the input produces big changes in the output.

Formally, the ill-posed Inverse problem for *denoising* reads as follows:

$$u^* = \arg\min_u \int_\Omega (u-f)^2 d\mathbf{x} \tag{2.2}$$

where σ_η is the standard deviation of η. Notice that it is necessary to include a constraint related to the noise power to avoid a trivial solution $u^* = f$, which is to assume that the noisy image f, is indeed, free of noise. Thus, the expected value for the energy of $(u - f)$ is the noise power of η, namely σ_η^2.

2.1.2 Regularization

Introduced firstly by Andrey Tikhonov in [Tikhonov and Arsenin, 1977], regularization is a technique that tackles ill-posed and ill-conditioned problems, introducing constraints that lead to a sort of injectivity and stability. In Figure 2.2, we see that infinity of different paths connect the set of solutions $\mathcal{U}$ to the output (given datum) f. The idea of the regularization is to select a unique solution from this infinite set $\mathcal{U}$, allowing to recover the ideal image u, or at least, find some u^* that is close to u. Therefore, regularization is to impose a constrain over the space of solutions $\mathcal{U}$ such that these space is reduced to only 1 candidate following a chosen criteria. In such a way, the 1st Hadamard condition is met, and the problem becomes well-posed.

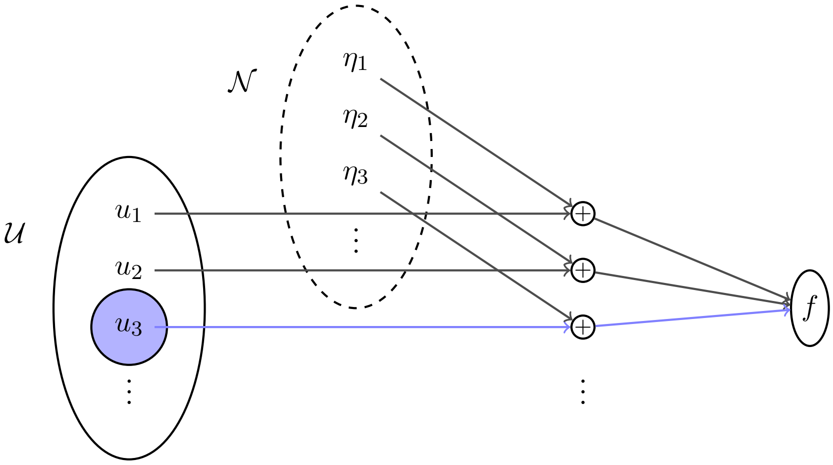


Figure 2.2: There exists infinity of combinations of u and η that are mapped to the same fixed f, i.e., there is not injectivity. The dashed line in $\mathcal{N}$ depicts that only a sample $\eta \sim \mathcal{N}(0, \sigma_\eta)$ is drawn, but it remains unknown.

Regularization also arises in different fields as in Machine Learning where there are plenty of data available and the goal is to learn a model that generalizes well on new data. To avoid problems like over-fitting, regularization is one of the most common used techniques. This will be extended in particular regarding CNNs in Section 2.2.5.

2.1.3 Bayesian modeling

Image Processing problems can be faced relying on a Bayesian framework since stochastic process arise and are, in many cases, the main source of signal degradation [Kornprobst, 2006]. It is also possible to model deterministic elements that are unknown in practice as random variables with an associated probabilistic distribution, like patterns, blurring kernels, etc. Bayesian modelling allows to do inference not only based on the available data but also adding prior knowledge to improve predictions. Using **Bayes' Rule** we have:

$$\mathrm{p}(u|f) = \frac{\mathrm{p}(f|u)\mathrm{p}(u)}{\mathrm{p}(f)} \tag{2.3}$$

where $\mathrm{p}(u|f)$ is the *posterior* probability density of the hypothesis u given the data f, $\mathrm{p}(f|u)$ is the so-called *likelihood* that represent what is expected in data f for a given hypothesis u, $\mathrm{p}(u)$ is the *prior* probability density that tells us prior knowledge about the hypothesis u and $\mathrm{p}(f)$ is the probability density of the data, referred to as *evidence*. Thus, we aim at maximize the *posterior* probability density, i.e., using a Maximun a Posteriori (MAP) scheme:

$$\max\left\{\mathrm{p}(u|f) = \frac{\mathrm{p}(f|u)\mathrm{p}(u)}{\mathrm{p}(f)}\right\} \tag{2.4}$$

or equivalently,

$$\max\left\{\log(\mathrm{p}(u|f)) = \log(\mathrm{p}(f|u)) + \log(\mathrm{p}(u)) - \log(\mathrm{p}(f))\right\}. \tag{2.5}$$

Since $\mathrm{p}(f)$ is the probability density of a given f, it remains constant and can be dropped in the maximization problem that consists of finding a optimal u^* that maximize equation (2.5),

$$u^* = \arg\max_u \left\{\log(\mathrm{p}(f|u)) + \log(\mathrm{p}(u))\right\}. \tag{2.6}$$

Based on the degradation model presented in Section (2.1), we now define each term before taking logs making the assumption that both follow a Gaussian distribution,

$$\mathrm{p}(f|u) = \frac{1}{\sigma_1\sqrt{2\pi}} \exp\left(-\frac{||f - Ru||^2}{2\sigma_1^2}\right), \tag{2.7}$$

$$\mathrm{p}(u) = \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(-\frac{||u||^2}{2\sigma_2^2}\right). \tag{2.8}$$

Notice that u and f are functions in Ω so that for the sake of notation we use $u = u(\mathbf{x})$ and $f = f(\mathbf{x})$ where $\mathbf{x} \in \Omega$. Analogously, $\sigma_1 = \sigma_1(\mathbf{x})$ and $\sigma_2 = \sigma_2(\mathbf{x})$. However, since a digital image is a discretization of a continuous function and we shall assume each sample $u_{i,j} = u(\mathbf{x} = (i,j))$ to be Independent and Indentically Distribuited (i.i.d.), the standard deviations $\sigma_1(\mathbf{x}) = \sigma_1$ and $\sigma_2(\mathbf{x}) = \sigma_2$ will be therefore considered as constants. Replacing (2.7) and (2.8) in (2.6), and changing the sign of the equation so that the optimization problem turns into a minimization problem, we have:

$$u^* = \arg\min_u \left\{\frac{1}{2(\sigma_1)^2}||f - Ru||^2 + \log\left(\sigma_1\sqrt{2\pi}\right) + \frac{1}{2(\sigma_2)^2}||u||^2 + \log\left(\sigma_2\sqrt{2\pi}\right)\right\}.$$

Here, $||\cdot||$ denotes the ℓ_2-norm. Dropping constant terms that does not depend on u, multiplying the whole functional by $\sigma_2\sqrt{2\pi}$ and defining the change variable $\lambda = (\sigma_2/\sigma_1)^2$, we are lead to:

$$u^* = \arg\min_u \left\{\lambda \underbrace{\int_\Omega (f - Ru)^2 d\mathbf{x}}_{\text{Fidelity: } F(u)} + \underbrace{\int_\Omega u^2 d\mathbf{x}}_{\text{Prior: } P(u)}\right\}. \tag{2.9}$$

The first term is usually known as the Fidelity $F(u)$ that gives us a measure of how far is the solution u from f through the operator R, while the second is the Prior $P(u)$, also refereed to as the Tikhonov regularizing term. In (2.9), the prior imposes a condition over the energy of the solution, promoting the unique minimal energy solution.

2.1.4 Tikhonov Regularization in Image Processing

As in (2.9), the regularizing term that comes naturally when the prior distribution $\mathrm{p}(u)$ is modeled as Gaussian, penalizes the energy of the solution. This is widely used in SP for the reason that this energy is often related with a power consumption that always implies a cost desirable to be reduced. In Image Processing however, this regularizing term does not perform very well. It is more suitable to use an L^2. A good introduction to functional spaces can be found in [Demengel and Demengel, 2012]. penalization over the gradient of the solution instead of over the solution itself. The prior $P(u) = ||u||^2$ is then replaced by $P(u) = ||\nabla u||^2$. Consequently, the prior distribution considering 2D images becomes

$$\mathrm{p}(u) = \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left(-\frac{||\nabla u||^2}{2\sigma_2^2}\right). \tag{2.10}$$

The improved results obtained by this prior term have their foundations in the way it introduces relevant prior knowledge to the problem, by penalizing the oscillations of the solution. In fact (2.8) is a Gaussian distribution centered in 0, which means that the solution that has more probability density is a constant image $u = 0$. It is easy to see that such assumption is totally wrong since an image is often whatever but a constant. On the contrary, (2.10) describes a probability distribution that can be read as: the most likely image u is the one which has very low gradient energy. The gradient of an image is indeed a measure of its variation of the images on the domain Ω. In Figure 2.3 we see how considering the prior in (2.10) is more suitable for images since most of the pixels in image $|\nabla u|$ are 0, which fits the zeros-mean distribution. The gradient, that is the precursor of image's edges, $\nabla u = (\partial u/\partial x, \partial u/\partial y)$ [1], is a vector of changes along vertical and horizontal axes x, y.

The problem (2.9) with gradient-based priors reads as follows:

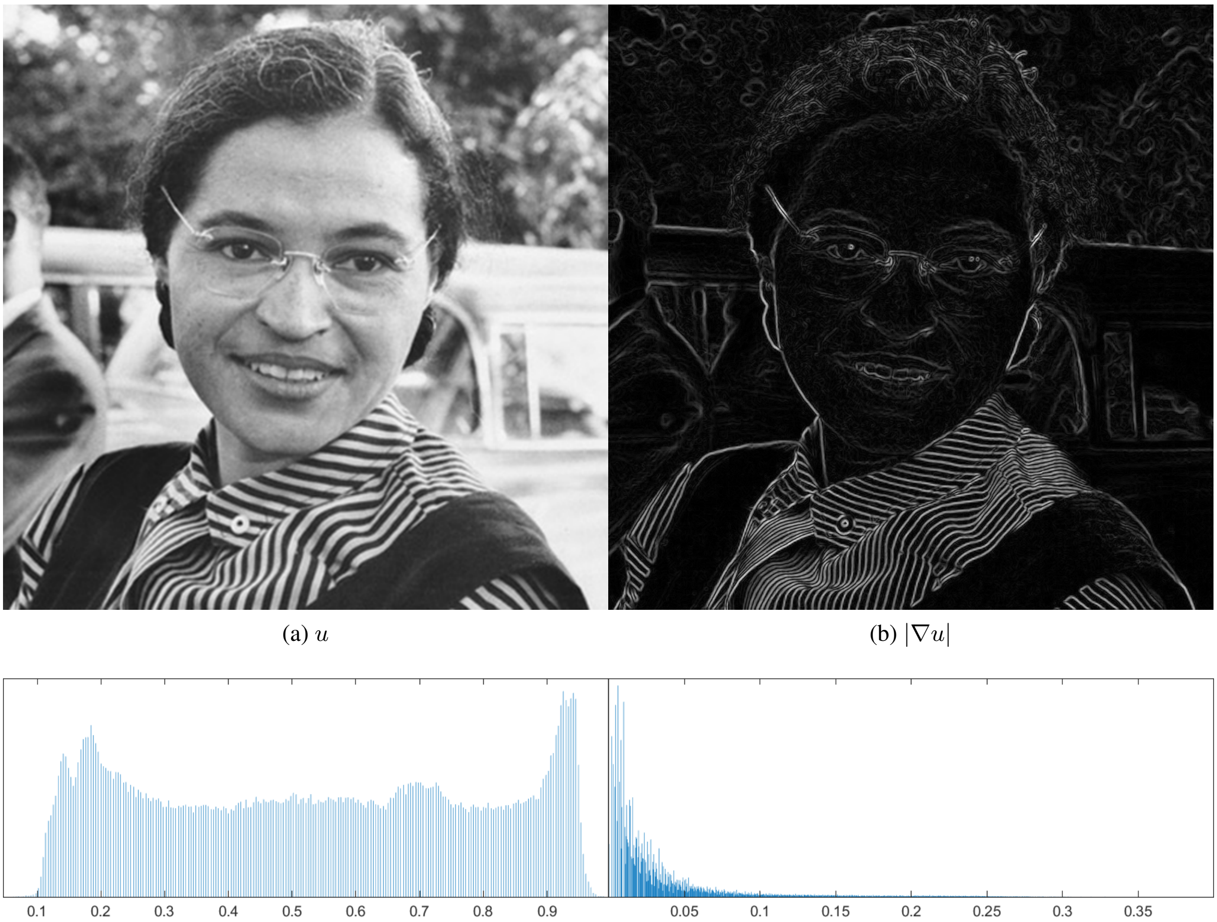
$$u^* = \arg\min_u \left\{ \lambda \int_\Omega (f - Ru)^2 d\mathbf{x} + \int_\Omega |\nabla u|^2 d\mathbf{x} \right\} \tag{2.11}$$

where the functional of energy $E(u) = \lambda F(u) + P(u)$ has control over u and its variations ∇u. That is how Variational Methods arise in Image Processing and why they constitute an important and powerful tool to tackle such problems. Furthermore, to solve the minimization problem above, it is necessary to define the first order optimality conditions of $E(u)$. To this end, we shall consider the concept Fréchet-derivative and its weak variants. Formal definitions can be found in many book of Functional Analysis such as [Brézis et al., 2003].

2.1.5 Well-posed regularized inverse problems

We have seen in the previous Section the importance of well-posedness in order to solve Inverse problems and how regularization give us a way to overcome ill-posed and ill-conditioned problems. We have also shown how imposing constraints that leads to a regularized optimization problem can also be derived

[1] Note that $\mathbf{x} = (x, y) \in \Omega$. In literature it is often used $x \in \Omega$ where $x = (x_1, x_2)$ indistinctly.



(a) u

(b) $|\nabla u|$

(c) Histogram of u

(d) Histogram of $|\nabla u|$

Figure 2.3: u versus ∇u based *priors*

from a Bayesian Framework, where the regularization term is identified with the prior knowledge, and it is combined with the likelihood term to obtain a good estimation of the posterior distribution. Tikhonov regularization with gradient-based priors exhibit better performances, but they can be widely improved if we change the way the variations of the solution are measured. This is known as the Generalized Tikhonov regularization.

This is the case for the celebrated Total Variation (TV) operator, introduced to solve a denoising problem through the ROF model in [Rudin et al., 1992], which has been extensively used with success in the Image Processing field. Here, the prior term $P(u)$ is defined (formally) by:

$$P(u) = \text{TV}(u) = \int_{\Omega} |\nabla u| d\mathbf{x} \tag{2.12}$$

that gives a L^1-norm measure of the gradient ∇u. For a detailed introduction to the TV operator and its properties see [Chambolle et al., 2010]. The benefits of this operator mainly consists of the edge-preserving property, which is a key-tool in general in Image Processing since an image and its content are highly determined by the shapes and structures of the objects. This tell us about the nature of the solution u that, in the Tikhonov denoising model (2.11), is confined to the Sobolev space $W^{1,2}(\Omega)$ (see [Dautray and Lions, 2012]) since its derivative is square-integrable. This implies that the solution of the model is continuous generating blurring effect in the image. On the contrary, when the Total

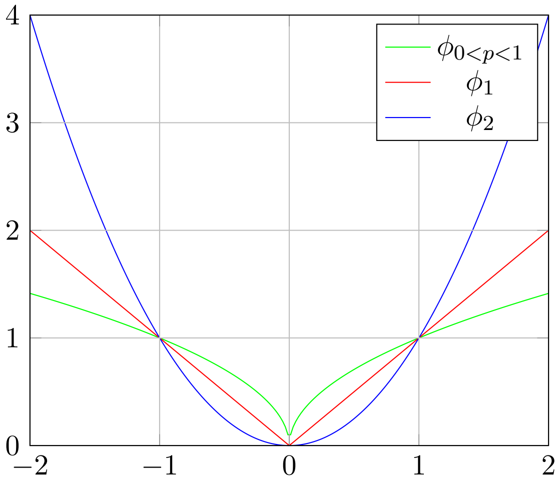


Figure 2.4: Hyper-Laplacian potential functions

Variation operator is considered it can be shown that the solution belongs to the space of Bounded Variations, i.e. $u \in BV(\Omega)$. Relevant results and details can be found in [Ambrosio et al., 2000]. This framework allows discontinuities in the solution and the visual results are sharp and clear images. As mentioned in [Rudin et al., 1992], such effect may be partially psychological, but the improvement is also reflected in terms of evaluation metrics as the Peak Signal Noise Ratio (PSNR) or the Structural Similarity (SSIM). The solution of the related minimization problem (the Gaussian Denoising model [Chambolle and Lions, 1997] or even others one that take into account different noise models, such as the Rician Denoising model [Martín et al., 2017]), belongs to the space of functions of Bounded Variation, $BV(\Omega)$, among which the piece-wise constant functions play an important role in Image Processing tasks as we will further detail in chapter 5.

Further works have been carried out in order to find appropriated prior regularizing terms. It turns out that, based on approximating results in practice, to get those prior improved we have to consider non-convex regularization terms that, of course, complicate the analysis of the problem and therefore its resolution. This is the case, among others, for the Hyper-Laplacian potential functions family:

$$\phi_p(s) = |s|^p \tag{2.13}$$

that defines in turn Hyper-Laplacian prior distributions $p(u) \propto e^{-k|s|^p}$. In Figure 2.4 some instances of the Hyper-Laplacian family are plotted for different values of p. The curves show how the selected potential function penalizes the functional of energy depending on the magnitude of the gradient. The behaviour of the model when the different values of p are considered, is that greater values of p penalize strong gradients which cause blurring effect while small values of p penalize lower preserving sparse gradients images. When $0 < p \leq 1$ the functional is not differentiable. This can be avoid by introducing a regularization family of potential functions. In the case of TV operator we make use of duality arguments which do not hold when $0 < p < 1$.

2.1.6 The Euler-Lagrange equations

Returning to the minimization problem (2.11), replacing the prior term by the Hyper-Laplacian family $\phi_p(s)$ and setting $R = \mathbb{I}$, $p = 1$ to recover the above mentioned denoising ROF model [Rudin et al., 1992], we have:

$$E(u) = \frac{\lambda}{2}\int_\Omega (f-u)^2 d\mathbf{x} + \text{TV}(u). \tag{2.14}$$

Notice that, for convenience, the Fidelity term is multiplied by $1/2$ for the sake of notation in further equations (Euler-Lagrange). The notation $\int_\Omega |\nabla u| d\mathbf{x}$ for the TV operator is widely used in the Image Processing community but constitutes an abuse of notation since it requires the solution u to be differentiable and thus continuous: $u \in W^{1,1}$, which therefore contravenes the preserving edge property desired in images. Instead of this and following [Chambolle and Lions, 1997], we replace the previous term by the differential distribution $|Du|$ that is a finite Radon measure in Ω of u.

$$E(u) = \frac{\lambda}{2}\int_\Omega (f-u)^2 d\mathbf{x} + \int_\Omega |Du| d\mathbf{x}. \tag{2.15}$$

Then, we proceed to compute the first order optimality conditions. In the case of equation (2.11), since the functional is Fréchet differentiable, we have

$$E^{'}(u) = 0,$$

$$E^{'}(u) = -\Delta u + \lambda(u - f)$$

that is the so-called **Euler-Lagrange Equation**. The problem is to solve the elliptic problem for which it is necessary to include the boundary condition. It is common in Image Processing to set homogeneous Neumann boundary conditions as a mean to preserve the mass of the given data f, i.e., $\int_\Omega u d\mathbf{x} = \int_\Omega f d\mathbf{x}$. We have

$$P_{\text{linear}} \begin{cases} -\Delta u + \lambda(u-f) = 0 \\ \nabla u \cdot \vec{n} = 0 \end{cases} \tag{2.16}$$

where $\vec{n}$ is the unitary normal vector to the boundaries and

$$\text{div}\,(\nabla u) = \Delta u.$$

If we now consider the Euler-Lagrange equation of (2.15) which is not Fréchet differentiable due to the TV term, the concept of subdifferential must be introduced.

Definition 2.1.1. (Subgradient) Let $E : X \to \mathbb{R}$ a convex proper functional. The subgradient of E at u is defined as:

$$\partial E(u) := \{u^* \in X^* | E(v) \geq E(u) + < u^*, v - u >, \forall u^* \in X^*\}$$

A functional E is said to be subdifferentiable in u if $E(u)$ is finite, and the set $\partial E(u)$ is not empty.

Thus, the Euler-Lagrange of equation (2.15) also accompanied by the boundary conditions, reads

$$P_{\text{TV}} \begin{cases} -\Delta_1 u + \lambda(u-f) \in \partial E \\ Du \cdot \vec{n} = 0 \end{cases} \tag{2.17}$$

where

$$\text{div}\left(\frac{Du}{|Du|}\right) = \Delta_1 u \in \partial P.$$

Finally, the problems above can be generalized introducing the aforementioned hyper-laplacian potential functions (2.1.5):

$$P_p \begin{cases} -\Delta_p u + \lambda(u-f) \in \partial E \\ Du \cdot \vec{n} = 0 \end{cases} \tag{2.18}$$

and the divergence term in this equation, termed as p-*Laplacian* diffusion operator, is

$$\text{div}\left(Du|Du|^{p-2}\right) = \Delta_p u \in \partial P.$$

2.1.7 Gradient Descent

In order to solve the proposed problems P_{linear} (2.16) P_{TV} (2.17) and P_p (2.18), the first classical approach commonly used is the gradient descent method, which is easy to implement but perform poorly in terms time computing and efficiency. Gradient descent consists of introducing an auxiliar time t variable, so that when $t \to \infty$, an iterative evolution of the solution converges to the minimizer of the PDE. In fact, the gradient descent scheme is to solve the parabolic problem

$$\frac{\partial u}{\partial t} = -\partial E(u) \tag{2.19}$$

which tends to the elliptic problem $\partial E(u) = 0$. Based on these approach, we introduce the discretization problem that leds to the numerical resolution.

Discrete Framework

To obtain a numerical resolution it is necessary to provide a discrete version of the input images $u_{i,j} = u(i\Delta x, j\Delta y) : \mathbb{Z}^2 \to \mathbb{R}$, the required differential operators and the gradient descent method as well. The divergence operator relies on the gradient which in turn is defined as $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) = (\nabla x, \nabla y)$ we define $\nabla x^+, \nabla x^-, \nabla y^+, \nabla y^-$ as the forward (+) and backward (-) finite difference operators in each directions, x and y. Derivatives can be computed by discretizing in several ways, among others :

- One-sided difference $\nabla_x u = \nabla_x^+ u = \dfrac{u_{i+1,j} - u_{i,j}}{h}$
- Central difference $\nabla_x u = \left(\nabla_x^+ u + \nabla_x^- u\right)/2$

Central differences are more accurate (error of the order $O(h^2)$) than One-sided ones ($O(h)$), however, they miss the dependency on the central sample $u_{i,j}$, which is not suitable for detecting thin structures. There exist different ways to avoid this problem using One-sided differences, and palliate the lack of symmetry. For instance, to compute the divergence, it is possible to use a forward one-side difference for the gradient computation, and a backward version for the second derivatives: $\text{div}(\cdot) = \nabla_x^-(\nabla_x^+(\cdot)) + \nabla_y^-(\nabla_y^+(\cdot))$. The discrete version of the gradient descent step (2.19) is obtained discretizing the fictional time introduced in the parabolic problem as follows:

$$\frac{u_{k+1} - u_k}{\Delta t} = -\partial E(u_{i,j})$$

so the step iteration

$$u_{k+1} = u_k - \Delta t \cdot \partial E(u_{i,j}).$$

2.1.8 Advanced numerical method

In this Section we will focus on an advanced method that allows for powerful numerical resolutions of the aforementioned problems. As we stated, the TV operator is central in Image Processing Restoration Problems based on Variational methods due to its properties, among which the preservation of edges stands out. They main difficulty lies on the lack of differentiability of such operator, thus a different definition or formulation of the deemed TV must be found. In the already mentioned paper [Rudin et al., 1992], the authors provides a new and ground-breaking definition of the TV operator through a seminorm. The formal definition reads as follows:

Definition 2.1.2. (Total Variation). Let $\Omega \subset \mathbb{R}^d$ be a domain for $u \in L^1_{loc}(\Omega)$ we define the *Total Variation* as the value of the functional

$$\mathrm{TV}(u) = \sup \left\{ \int_\Omega u \mathrm{div} v \, dx \mid v \in \mathcal{C}^\infty_c(\Omega, \mathbb{R}^d), \; ||v||_\infty \leq 1 \right\},$$

which is a seminorm. Thus, the space $\mathrm{BV}(\Omega) = \{u \in L^1_{loc}(\Omega) \mid \mathrm{TV}(u) < \infty\}$, endowed with the norm $||u||_{\mathrm{BV}} = ||u||_1 + \mathrm{TV}(u)$ is called the space of functions of bounded variations.

Based on such a definition, [Chambolle and Pock, 2011] introduced the Primal-Dual algorithm for convex problem where the main objective is to speed-up the computation time (with a rate of O(1/N)) avoiding any relaxation or regularization of non-differentiable operators, in this context, the TV. The general problem presented by the authors is in the form of a saddle point problem.

Let X, Y be finite-dimensional real vector spaces, and $K : X \to Y$ a continuous linear mapping operator with induced norm:

$$||K|| = \max \{||Kx|| : x \in X \text{ with } ||x|| \leq 1\}$$

The general problem is then:

$$\min_x \max_y \langle Kx, y\rangle - J^*(y) + G(x), \tag{2.20}$$

which is solved iteratively through

- $y^{n+1} = (\mathbb{I} + \tau_d \partial J^*)^{-1} (y^n + \tau_d K \bar{x}^n)$
- $x^{n+1} = (\mathbb{I} + \tau_p \partial G)^{-1} \left(x^n - \tau_p K^* y^{n+1}\right)$
- $\bar{x}^{n+1} = x^{n+1} \theta (x^{n+1} - x^n)$,

where τ_d, τ_p are the step sizes corresponding to the dual and the primal step respectively, G and F^* are proper, convex, lower-semicontinuous functionns and F^* denotes the convex-conjugate of F. When setting $\theta = 0$, we recover the Arrow-Hurwicz method [Hartley, 1960] which proposed to solve the ROF models. This approach as well as the gradient-based approach shall be considered to solve the numerical problems in this dissertation.

2.2 Machine Learning

In the Introduction Chapter 1 we have shown that most of the problems addressed in Machine Learning aim at extracting features from given signals which in turn are used to be a direct problem in the sense of causality as defined in Section 2.1. The Machine Learning problem is then to find a function or model that correctly transform from a given cause to its effect. Without entering, yet, in the details of how these functions are defined, we will consider a family of functions $\mathbf{f}(\boldsymbol{\omega}, \mathbf{x})$ that depend on some parameters $\boldsymbol{\omega}$ to define the ML problem from a Bayesian point of view.

2.2.1 Problem Statement - Bayesian Approach

To find the function $\mathbf{f}$ that transforms the data $\mathbf{X}$ into labels $\mathbf{Y}$, we must first define a cost function that evaluates how correct the predictions are made by $\mathbf{f}$. That is, a measure of the error or cost that is committed by $\mathbf{f}$. Therefore, the objective will be to minimize this cost. We can intuitively decide to minimize a distance $d(\mathbf{y}, \hat{\mathbf{y}})$ among the predictions $\hat{\mathbf{y}} = \mathbf{f}(\boldsymbol{\omega}, \mathbf{x})$ and the true labels $\mathbf{y}$, or on the contrary, to deduce in a well-founded way, from Bayes, which are the distances to be minimized. In the following, we will show how these cost functions arise naturally if a MAP scheme is used.

Let $\mathbf{X} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ...\mathbf{x}^{(N)}\right\}$ a set of elements (input) corresponding each of them to a single element (label) content in the set $\mathbf{Y} = \left\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ...\mathbf{y}^{(N)}\right\}$, where $i = 1, .., N$. Each element of $\mathbf{X}$ or $\mathbf{Y}$ is, in general, a vector: $\mathbf{x}^{(i)} \in \mathbb{R}^m$, $\mathbf{y}^{(i)} \in \mathbb{R}^n$. Applying Bayes' Rule, the conditional *posterior* distribution is defined as follows:

$$p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y} \mid \boldsymbol{\omega}, \mathbf{X}) p(\boldsymbol{\omega})}{p(\mathbf{Y} \mid \mathbf{X})} \tag{2.21}$$

where $p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})$ is the *likelihood* and $p(\boldsymbol{\omega})$ the *prior* distribution placed over the parameters' model $\boldsymbol{\omega}$. Depending on how we define each term of the previous equation, we will be considering a specific problem. In Machine Learning, there exists mainly two different type of problems depending on the nature of the target data.

- Regression: the objective is to predict a **continuous** variable.
- Classification: the objective is to predict a **discrete** variable.

2.2.2 Regression Problems

We start with equation (2.21) where both, the **likelihood** term and the *prior* will be considered as gaussians, the same way it is done in Section 2.1.3.

$$p(\mathbf{Y} \mid \boldsymbol{\omega}, \mathbf{X}) = \frac{1}{(2\pi\sigma_1^2)^{\frac{N}{2}}} \exp\left(-\frac{||\mathbf{Y} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{X})||_2^2}{2\sigma_1^2}\right) \tag{2.22}$$

$$p(\boldsymbol{\omega}) = \frac{1}{(2\pi\sigma_2^2)^{\frac{K}{2}}} \exp\left(-\frac{||\boldsymbol{\omega}||_2^2}{2\sigma_2^2}\right) \tag{2.23}$$

where $\boldsymbol{\omega} = \{\omega_1, \omega_2, ..., \omega_k\}$ are the K parameters by which the $\mathbf{f}$ function is composed. $\boldsymbol{\omega}$ is in general a set of parametric objects such as scalars, vectors, tensors or even mixture of them. It is common to think of them as a vector constructed as a concatenation of all the objects in $\boldsymbol{\omega}$, i.e., a vectorization of $\boldsymbol{\omega}$, and it will also be refereed to as $\boldsymbol{\omega}$ indistinctly as long as there is no confusion. Moreover, it is also assumed that all the pair of elements $(\mathbf{x}, \mathbf{y}) \in (\mathbf{X}, \mathbf{Y})$ are i.i.d. as well as the parametric objects $\omega_k \in \boldsymbol{\omega}$. This allows us to re-write equations (2.22) and (2.23) as

$$p(\mathbf{Y} \mid \boldsymbol{\omega}, \mathbf{X}) = \prod_{i}^{N} \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2}{2\sigma_1^2}\right) \tag{2.24}$$

$$= \frac{1}{(2\pi\sigma_1^2)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma_1^2} \sum_{i}^{N} ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2\right) \tag{2.25}$$

$$p(\boldsymbol{\omega}) = \prod_{i}^{K} \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left(\frac{||\boldsymbol{\omega}||_2^2}{2\sigma_2^2}\right) \tag{2.26}$$

$$= \frac{1}{(2\pi\sigma_1^2)^{\frac{K}{2}}} \exp\left(-\frac{1}{2\sigma_1^2} \sum_{i}^{K} ||\boldsymbol{\omega}_i||_2^2\right) \tag{2.27}$$

Since we want to maximize the *posterior* distribution and the probability distribution of the *evidence* (data) $p(\mathbf{Y} \mid \mathbf{X})$ remains constant w.r.t. the parameters $\boldsymbol{\omega}$,

$$p(\mathbf{Y} \mid \boldsymbol{\omega}, \mathbf{X}) \propto p(\mathbf{Y} \mid \boldsymbol{\omega}, \mathbf{X}) p(\boldsymbol{\omega}), \tag{2.28}$$

we have a maximization problem

$$\boldsymbol{\omega}^* = \arg\max_{\boldsymbol{\omega}} p(\mathbf{Y}, \boldsymbol{\omega}, \mathbf{X}). \tag{2.29}$$

that, in turn, is re-written by taking $-\log(\cdot)$ in both sides of the equation (2.28) as a minimization problem:

$$\boldsymbol{\omega}^*_{\boldsymbol{\omega} \in \mathbb{R}} = \arg\min_{\boldsymbol{\omega}} \mathcal{C}(\boldsymbol{\omega}) \tag{2.30}$$

where $\mathcal{C}(\boldsymbol{\omega})$ is the so-called *cost* function:

$$\mathcal{C}(\boldsymbol{\omega}) = \frac{1}{2\pi\sigma_1^2} ||\mathbf{Y} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{X})||_2^2 + \log((2\pi\sigma_1^2)^{\frac{N}{2}})$$
$$+ \frac{1}{2\pi\sigma_2^2} ||\boldsymbol{\omega}||_2^2 + \log((2\pi\sigma_2^2)^{\frac{K}{2}}).$$

Simplifying,

$$\mathcal{C}(\boldsymbol{\omega}) = \frac{1}{2\pi\sigma_1^2} ||\mathbf{Y} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{X})||_2^2 + N\log(\sigma_1) + \frac{1}{2\pi\sigma_2^2} ||\boldsymbol{\omega}||_2^2 + K\log(\sigma_2) \tag{2.31}$$

and getting rid of the terms that do not depends on $\boldsymbol{\omega}$, we end up with a variational minimization problem:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \left\{ \frac{1}{2\pi\sigma_1^2} ||\mathbf{Y} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{X})||_2^2 + \frac{1}{2\pi\sigma_2^2} ||\boldsymbol{\omega}||_2^2 \right\} \tag{2.32}$$

Unlike in the previous subsection 2.1.3, here we multiply the whole functional by $2\pi\sigma_1^2$, and set $\lambda' = 2\pi\sigma_1^2 / 2\pi\sigma_2^2$

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \left\{ ||\mathbf{Y} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{X})||_2^2 + \lambda_1 ||\boldsymbol{\omega}||_2^2 \right\} \tag{2.33}$$

so the hyper-parameter controls the regularizer prior term instead of the fidelity. It is easy to see how the Tikhonov regularization arises again and the minimization problem can be read as selecting, from the set of predictions $\mathbf{f}(\boldsymbol{\omega}, \cdot)$ generated by the solutions $\boldsymbol{\omega}^*$ (if there exists more than one), the one with lowest energy in $\boldsymbol{\omega}$. This regularization term promotes each weight (parameter) ω_k to be near to 0 and penalizes quadratically high values for them (see Figure 2.4). Finally, multiplying the whole functional by $1/N$ and setting $\lambda = \lambda_1/N$

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \left\{ \frac{1}{N} \sum_{i}^{N} ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2 + \frac{\lambda_1}{N} \sum_{i}^{K} ||\omega_i||_2^2 \right\},$$

i.e.

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \left\{ MSE(\mathbf{Y}, \mathbf{f}(\mathbf{X})) + \lambda||\boldsymbol{\omega}||_2^2 \right\}. \tag{2.34}$$

The first term is the Mean Squared Error (MSE) for the training set and the second is the already mentioned $L2$ or Tikohnov regularization term with the hyper-parameter λ commonly referred to as *weight decay* due to the effect it promotes over the weights $\boldsymbol{\omega}$ in the optimization step based on gradient descent algorithms.

Equation (2.34) has been derived considering to solve a regression problem, this is, assuming a gaussian error between the real values $\mathbf{Y}$ and the predictions $\mathbf{f}(\mathbf{X}, \boldsymbol{\omega})$. This placed gaussian distribution, however, must be changed to address classification problems.

2.2.3 Classification Problems

Classification is the task of assigning, to an input, a probability of belonging to a class. The output $\mathbf{f}$ is therefore a vector of probabilities that sum up to 1, indeed, a probability distribution function. The most used distribution in classification problems in Machine Learning is the *softmax*:

$$\text{Softmax}(\mathbf{f}) = \frac{e^{\mathbf{f}}}{\sum_j e^{f_j}},$$

a function with inputs called *logits*, Softmax : $\mathbb{R}^n \to [0, 1]^n$. The *posterior* distribution is proportional to the *softmax likelihood* and the gaussian *prior*

$$p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y}) \propto \text{Softmax}(\mathbf{f}(\boldsymbol{\omega}, \mathbf{X}))p(\boldsymbol{\omega}).$$

One requirement or assumption over this *likelihood* election is the fact that classes are mutually exclusive. This often translates to a *One-hot* encoding of the given labels $\mathbf{Y}$. Applying the MAP scheme to maximize the *posterior*, it is easy to realize that, as a consequence of selecting the *softmax* distribution function, maximizing the probability of a given example $\mathbf{x}^{(i)}$ implies minimizing the remaining probabilities. For that reason, the maximization step is performed on the correct output component $\hat{y}_c$, i.e.

Taking $-\log(\cdot)$ we have:

$$\begin{aligned} -\log(p(y_c \mid \boldsymbol{\omega}, \mathbf{x})) &= -\sum_{j=1}^{n} \mathbf{1}\{y = c\} \log(\hat{y}_j) \\ &= -\sum_{j=1}^{n} \mathbf{1}\{y = c\} \log\left(\frac{e^{f_j(\boldsymbol{\omega}, \mathbf{x})}}{\sum_{k=1}^{n} e^{f_k(\boldsymbol{\omega}, \mathbf{x})}}\right) \end{aligned}$$

where $\mathbf{1}\{\cdot\}$ is the characteristic function:

$$\mathbf{1}\{x\} = \begin{cases} 1, & \text{if } x \text{ is true} \\ 0, & \text{if } x \text{ is false.} \end{cases} \tag{2.35}$$

Finally, the resulting functional of energy that we aim at minimize is:

$$\mathcal{L}(\boldsymbol{\omega}) = -\frac{1}{N}\sum_{i}^{N}\left(\sum_{j=1}^{n}\mathbf{1}\left\{y=c\right\}\log\left(\frac{e^{f_j(\boldsymbol{\omega},\mathbf{x})}}{\sum_{k=1}^{n}e^{f_k(\boldsymbol{\omega},\mathbf{x})}}\right)\right) + \frac{1}{2\sigma_2^2}||\boldsymbol{\omega}||_2^2 + \log((2\pi\sigma_2^2)^{\frac{K}{2}}). \tag{2.36}$$

Logistic function as a particular case

The *logistic* function is wide used in Machine Learning when solving binary classification problems. Nonetheless, it is a particular instance of the *softmax* distribution function where there are only 2 classes ($\mathbf{f} \in \mathbb{R}^2$) and thus, their probabilities are complementary:

$$\hat{\mathbf{y}} = Softmax(\mathbf{f}) = \left(\frac{e^{f_1(\boldsymbol{\omega},\mathbf{x})}}{e^{f_1(\boldsymbol{\omega},\mathbf{x})}+e^{f_2(\boldsymbol{\omega},\mathbf{x})}}, \frac{e^{f_2(\boldsymbol{\omega},\mathbf{x})}}{e^{f_1(\boldsymbol{\omega},\mathbf{x})}+e^{f_2(\boldsymbol{\omega},\mathbf{x})}}\right)^T.$$

In effect, we have an output for each class $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2)$. However, since both outputs are complementary probabilities: $\hat{y}_2 = 1 - \hat{y}_1$, we can do without the second output and its parameters in **f** (it is said **f** is over-parametrized). The probability vector is:

$$\hat{\mathbf{y}} = (\hat{y}_1, 1 - \hat{y}_2).$$

Taking $-\log(\cdot)$ as usual obtaining

$$\begin{aligned} -\log(p(y_c \mid \boldsymbol{\omega}, \mathbf{x})) &= -\sum_{i}^{N} y_1\log(\hat{y}_1) + y_2\log(y_2) \\ &= -\sum_{i}^{N} y_1\log(\hat{y}_1) + (1-y_1)\log(1-\hat{y}_1) \end{aligned} \tag{2.37}$$

2.2.4 Deep Learning

Deep Learning, as a particular applied tool that belongs to the Machine Learning paradigm, has revolved many fields of applied Mathematics and Engineering in the last decade. The term was first used by Rina Dechter [Dechter,] in 1986 in the context of efficiency in searching algorithms. However, the first working architecture that can be considered the precursor of the DL was presented in [Ivakhnenko,] by Alexey Ivakhnenko and Lapa. Ivakhenko is considered to be the "Father of Deep Learning" for its contribution developing Group Method of Data Handling (GMDH), an inductive statistical learning method.

The distinction between Machine Learning and Deep Learning lies in the structure of the parametric functions used to map input data $\mathbf{X}$ into labels $\mathbf{Y}$. Those properties have enabled to achieve new and performing algorithms that have improved the state of the art in many related fields. In Machine Learning, any parametric and flexible function is susceptible to be used. When we move to the Deep Learning field, these functions are NN. It is said that they are inspired by the neurological network of the brain, and it worth to remark "inspired by" in the sense brain connections are significantly more complex. A good analogy given by Yann Lecun, one of the founding fathers of Convolutional Neural Networks, is that, the same way we have found the principles of aeronautics, helped or inspired by birds (biology) and not exactly mimicking them, we should aim at finding the principles of Learning rather than replicating human or animal brains. The basic NN employed in Deep Learning are the so-called Feedforward Networks, where there is an implicit notion of direction. In fact, in such networks the information flows forward through the whole neural structure. Besides, this structure is a composition of functions such that a Neural Network is mathematically expressed as $\mathbf{f} = \mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))$ (see Figure 2.5). All the intermediate functions have, in addition, the same structure, and are commonly identified as *layers*.

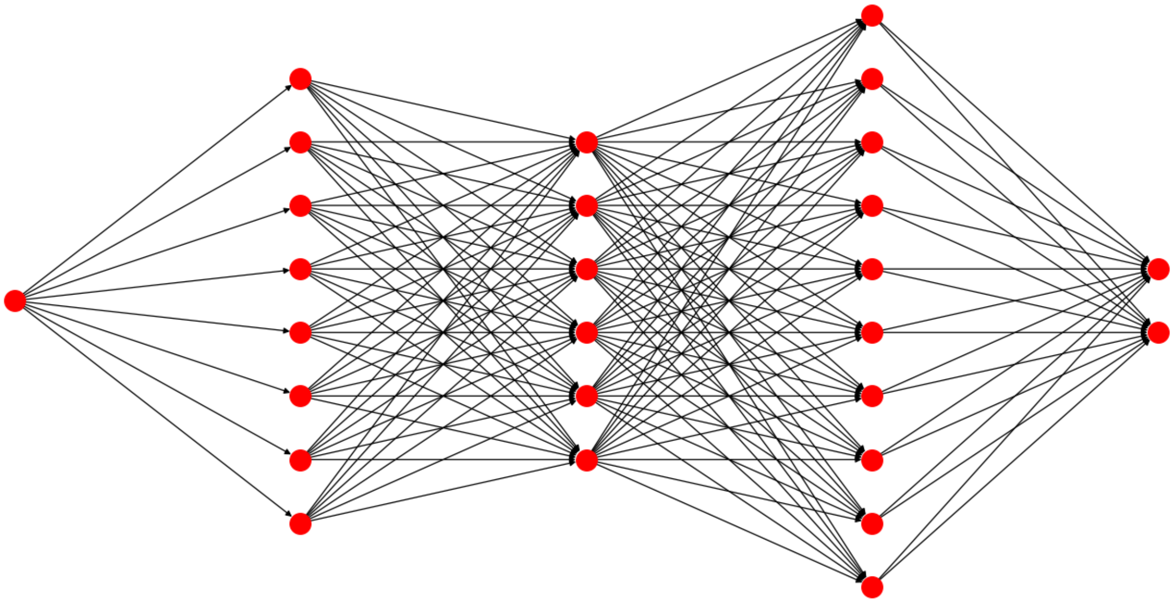


Figure 2.5: Neural Network with 3 hidden layers that are composed as: $\mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))$.

Neural Networks

In the following, we will give a formal definition of a Neural Network and its elements, namely: *nodes*, *layers*, *activation functions*, etc. A node acts as a *neuron* in the brain limited by the fact it has several inputs but an unique output. Each node aims at *firing* (activate the output) depending on the given combination of entries. The criteria used by a node to decide whether or not to activate the output is shaped by a so-called *activation function*. This activation is then sent forward as an input for the nodes or neurons of the next *layers*. This is, firstly a linear combination of the inputs:

$$f_1 = \mathbf{w}^T\mathbf{x} + b = \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + b. \tag{2.38}$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$ are the input vector and a node parameters, respectively. And secondly, the *activation function* $\sigma(\cdot)$. A typical *activation function* is the hyperbolic tangent $\tanh(\cdot) : \mathbb{R} \to [-1, 1]$ among many others, so the linear activation (2.38) ranging in $\mathbb{R}$, is now bounded to $[-1, 1]$

$$f_1 = \sigma\left(\mathbf{w}^T\mathbf{x} + b\right) \tag{2.39}$$

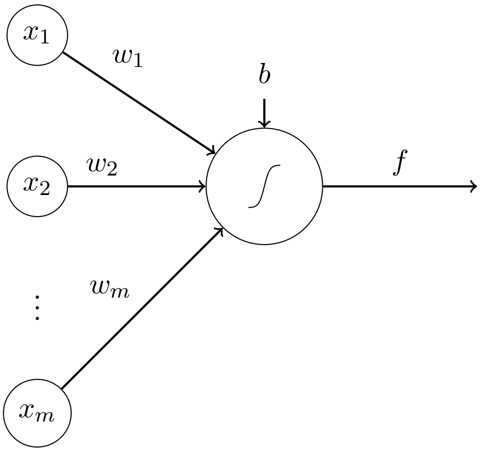


Figure 2.6: A single neuron

In Figure 2.6 there are m inputs and 1 output. Generalizing to n outputs we obtain a *layer*: $\mathbf{x} \in \mathbb{R}^m$, $W \in \mathbb{R}^{n \times m}$, $\mathbf{b} \in \mathbb{R}^n$ y $\mathbf{f} \in \mathbb{R}^n$, we have

$$\mathbf{f} = \sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{1,m} \\ \vdots & & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}\right)$$

whose graphical representation is depicted in Figure 2.7. The composition of several layers lead to the construction of the neural network in which each layer can have a different number of neurons and therefore different number of parameters (weights). Those layers that take as inputs the outputs of a previous layers and sent forward their outputs to a following layer, are called *hidden layers*.

$$\mathbf{f} = \sigma\left(\begin{bmatrix} w^{(2)}_{1,1} & \cdots & w^{(2)}_{1,m} \\ \vdots & \ddots & \vdots \\ w^{(2)}_{n,1} & \cdots & w^{(2)}_{n,m} \end{bmatrix} \sigma\left(\begin{bmatrix} w^{(1)}_{1,1} & \cdots & w^{(1)}_{1,m} \\ \vdots & \ddots & \vdots \\ w^{(1)}_{n,1} & \cdots & w^{(1)}_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b^{(1)}_1 \\ \vdots \\ b^{(1)}_n \end{bmatrix}\right) + \begin{bmatrix} b^{(2)}_1 \\ \vdots \\ b^{(2)}_n \end{bmatrix}\right)$$

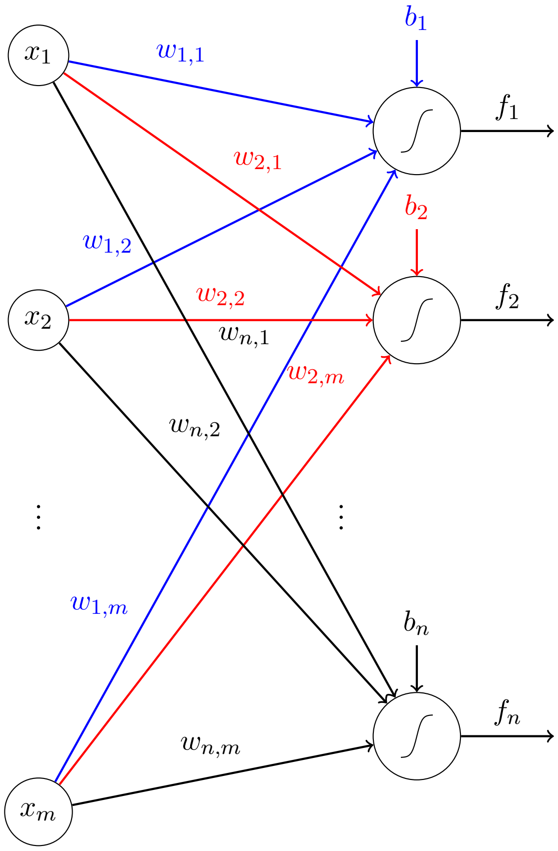


Figure 2.7: First layer in a Neural Network

i.e.,

$$\mathbf{f} = \sigma(W^{(2)}(\sigma(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)})) + \mathbf{b}^{(2)}) = \mathbf{f}_2(\mathbf{f}_1(\mathbf{x})) = \mathbf{f}_2 \circ \mathbf{f}_1(\mathbf{x}).$$

As has already been said, the relevant feature of Feedforward Network is related to its factorization: each layers has the same function so that $\mathbf{f}_n = \sigma(W^{(n)}\mathbf{f}_{n-1} + \mathbf{b}^{(n)})$, giving us an recursive rule to built increasingly deeper networks Deep Neural Network (DNN). After these Feedforward Networks, others have appeared enabling feedback connections and thus allowing the information to flow backward RNN. These feedback loops introduce memory blocks in Neural Networks which in turn can be used to define states. An example of this are the LSTM.

But, probably the most important property of a Feedforward Neural Network is its capacity and flexibility to map (approximate) input data $\mathbf{X}$ into labels $\mathbf{Y}$. It is said NN are Universal Approximation functions. Proof of this is the following theorem [Cybenko, 1989]:

Theorem 1. *Let σ be a non constant, bounded, and continuous function. Let I_n denote the n-dimensional unit hyper-cube, $[0,1]^n$. The space of real-valued continuous functions on I_n is $C(I_n)$. Then, given any $\epsilon > 0$ and any function $f \in C(I_n)$, there exist an integer N such that we can define:*

$$F(x) = \sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b})$$

as an approximation of f; i.e.

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_n$. This is, functions $F(x)$ are dense in $C(I_n)$.

Roughly speaking, George Cybenko demonstrated that those functions can approximate any continuous functions in a compact set. It is however interesting to see that, in such theorem, the condition over the *activation function* is that it must be *sigmoidal* (non constant, continuous and bounded),

Definition 2.2.1. A function σ is said to be *sigmoidal* if:

$$\sigma(x) = \begin{cases} 1, & \text{as } x \to +\infty \\ 0, & \text{as } x \to -\infty, \end{cases} \tag{2.40}$$

condition that is theoretically violated in most of the applied NN nowadays. For example, one of the most common and successful activation function is the Rectified Linear Unit $ReLU(x) = \max(0, x)$ among other variants (Figure 2.8). Such activation function is clearly not bounded, however, in practice, since all the experiments are numerically solved, this bound exist and it is reached by the bit depth of the machine.

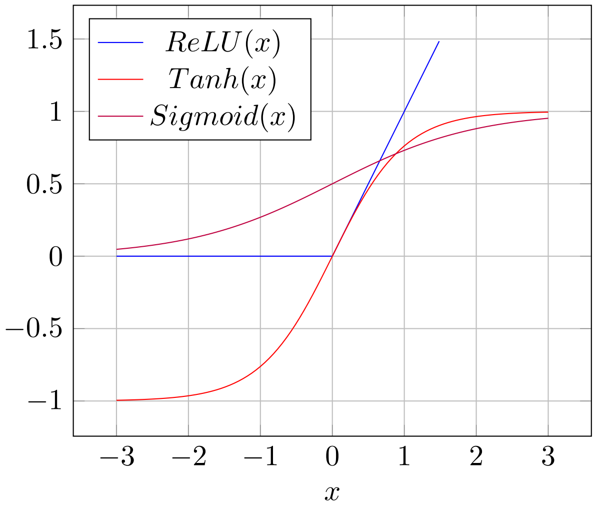


Figure 2.8: Activation functions

Convolutional Neural Networks

Convolutional Neural Networks constitute an essential element in Machine Learning applied to the Computer Vision and Image Processing fields. One of the first and successful CNN was proposed by Yann LeCun et al. [LeCun et al., 1998] applied to characters and documents recognition (LeNet). However, the first properly named Convolutional Network is known as *Neocognitron* proposed by [Fukushima and Miyake, 1982] in 1982 that was able to detect patterns in images. Such Neural Networks and even the Deep Learning field are deeply inspired and related to some experiments carried out in 1959 by [Hubel and Wiesel, 1959] analyzing the visual receptive field of cats: *"Receptive fields of single neurones in the cat's striate cortex"*. In these works, Hubel and Wiesel showed a hierarchy in the visual system that was composed of a group of neurons specialized in the recognition (firing) of basic patterns. The success of CNNs lies on their structure and the replacement of the linear classifiers $W\mathbf{x} + \mathbf{b}$ by a convolution operation. I.e., we replace equation (2.39) by:

$$f = \sigma(K * X + B), \tag{2.41}$$

where $K \in \mathbb{R}^{k \times k}$ is the considered kernel, $X \in \mathbb{R}^{n \times n}$ represents the 2-dimensional input and $B \in \mathbb{R}^{(n+k-1)\times(n+k-1)}$ matrix of biases.

CNNs are one of the best NN architecture used in Image Processing or Machine Learning when the *information* is an image or contains it. The ability of CNNs to learn meaningful image hierarchies by finding convolutional kernels that extract at each layer more and more high level features, lead us to ask why those linear operations (convolutions) are so powerful. In fact, both, a Neural Network layer and a convolution node, are linear operations, but performs differently when a network is trained to classify images, for example. To figure out the reason of such difference, we have to analyze the convolution as a matrix operation.

Convolution as Matrix Operation In general, a convolution can be cast to a matrix multiplication by simply transforming one of the inputs into a Toeplitz matrix [Gray et al., 2006]. To this end, both entries must be properly vectorized, and then reshaped to fit the data structure (images). We have,

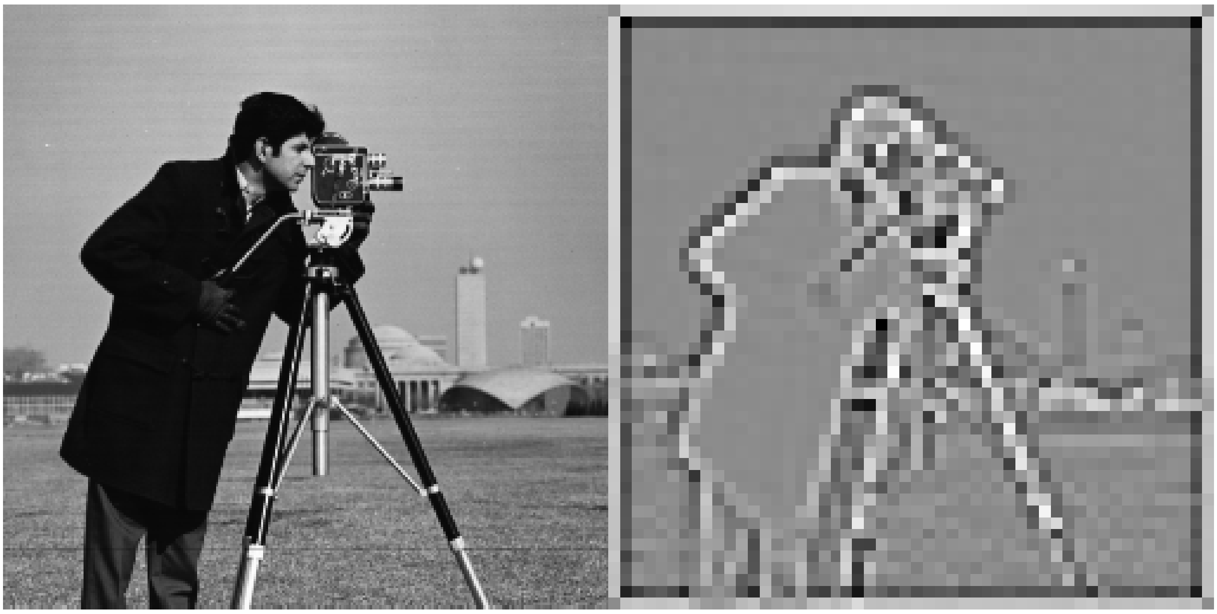
$$(K * X)(:) = W\mathbf{x} \tag{2.42}$$

where **x** is the vectorized input matrix X (image), W is the associated kernel matrix K and $(:)$ denotes a row-vectorization of a matrix. See a detailed example in Appendix Section 8.1. It turns out that, the conversion of the kernel K into a matrix W, generates a Toeplitz matrix which turns to be very sparse. In fact, for a laplacian kernel:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.43}$$

and an input image $X \in \mathbb{R}^{50\times 50}$ (down-sampled from image in Figure 2.9b), the associated matrix is $W \in \mathbb{R}^{2704\times 2704}$ in where only a 0.1813% of the coefficients are non-zero 2.9. It is easy to see that a convolution product is indeed a Neural Network with most of its weights set to 0. This is similar to impose a prior over those weights or a constrain in the energy functional to minimize, introducing a Lagrange multiplier, for example: $\lambda_{i,j} w_{i,j}^2$ and then, make $\lambda_{i,j} \to \infty$. For such reason it is said that a convolution operation imposes an infinitely strong prior. Moreover, not only a majority of weights are forced to be 0, those that are non-zero have also strong priors in the sense that all of them are the same along the diagonals (see Figure 2.9) which translates the fact that the filter (or the image) remains the same throughout the image. In other words, the generated Toeplitz matrix is built to exploit local properties of the input. This makes sense, reason why it works so well. As mentioned in Section 2.1.4, a property of the images is that nearby pixels are likely to be similar and those that are significantly different define the structure of an object or part of it, and they must be preserved. Note that, the properties of the input data drive the election of the correct prior: TV and hyper-laplacian priors in Image Processing and CNNs in Machine Learning (when dealing with images). From the computational side, convolutions are prone to be parallelised in hardware like Graphical Processor Units (GPUs) and the memory requirement can also be reduced while keeping similar or better performance and capacities; number of weights with matrix application vs kernel convolution: $(m \times n \times m \times n) >> (k \times k)$.

A CNN uses to be a DNN, where each layer is composed by several convolution nodes. The features extracted along these layers are shown hierarchical. The first layers detect basic features in images as corners or edges, middle layers extract basic structures and shapes composed by the previous basic features and the deepest layers detect high level features. The ability to detect such features is related to the so-called receptive field, which is a neighborhood region around each pixel that allows to relate all the



(a) Original image

(b) Laplacian down-sampled filtered image

considered pixel and thus, find different shapes or structures. Contrarily to non-local methods that we will further analyze in chapter 4, in a CNN the non-locality property is built throughout the layers, modifying the resolution of the input features and maintaining the kernels' size, increasing the receptive field. This technique is known in literature as *pooling*. However, *pooling* refers to a selection rule to decimate an input image. This is widely known as sub-sampling a signal, and it can be formally expressed through different ℓ_p norms, among others:

$$Pooling(x) = ||\mathcal{N}(x)||_p,\ p = \{1, 2, \infty\} \tag{2.44}$$

where ℓ_1 norm acts as an average pooling (for non-negative inputs), ℓ_∞ as max-pooling and $\mathcal{N}(\cdot)$ is the considered neighborhood around a pixel. It is also possible to use a stochastic pooling rule [Zeiler and Fergus, 2013], which is closely related to Dropout that shall be discussed in Chapter 3.

2.2.5 Regularization

Regularization in NN can be introduced by placing a prior distribution function over the solution ($\boldsymbol{\omega}^*$) of the optimization problem, also known as *weak* prior, or, impose by construction an *infinitely strong* prior, as we have shown in previous Section with CNNs. We have also seen that NN are Universal Approximation functions, which means that, with enough nodes and/or layers, any *possible* mapping problem can be solve with an arbitrary error. Then, why do we care about placing priors over the weights? Theorem 1 tells us nothing about how to find parameters $\{\alpha_i, \mathbf{w}, \mathbf{b}\}$. Moreover, it is assumed that, if a data based algorithm is used to find such optimal parameters, the required data is available. Obviously, this is not always the case. Regularization in NN in particular and in ML in general, is used to reduce the uncertainty of the optimal solution derived from the problem itself (ill-posedness) and/or because of the lack of data. Such lack of data usually implies a poor sample of data distribution, which in turn, provokes a bad generalization of the solution. When the available data does not represent the whole space of data, training a Neural Network can over-fit the training data and thus perform badly in new samples. Performing well in new samples that may differ a lot from the given data (training set), is the same as

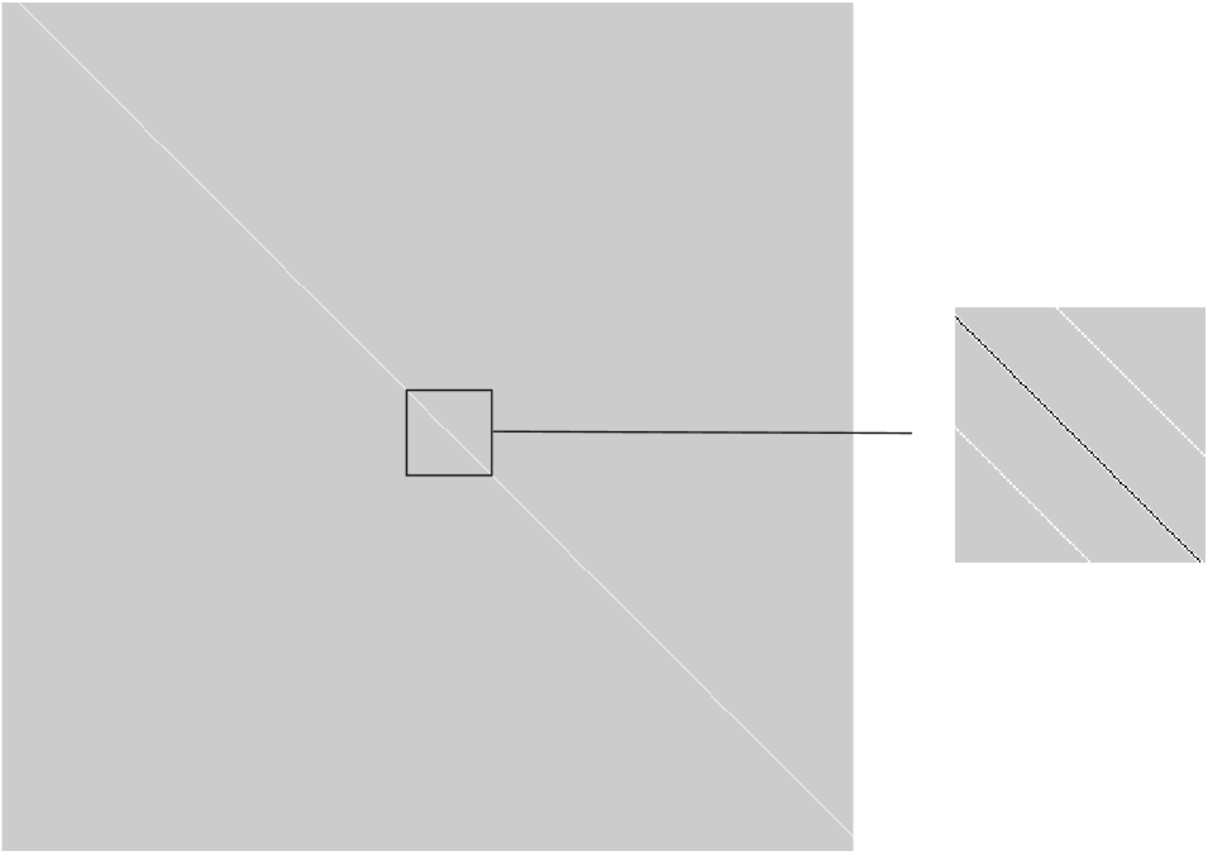


Figure 2.9: Toeplitz matrix generated from laplacian kernel

extrapolating over a partial distribution. Regularization is the prior knowledge or the hypothesis we make over the undiscovered distribution.

Gradient Based Numerical Resolution - Backpropagation

Let us consider problem (2.34), and more precisely and without loss of generality, the negative $\log$-likelihood term:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \mathcal{C}(\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} \frac{1}{N}\sum_{i}^{N} ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2. \tag{2.45}$$

As for the variational methods described in Section 2.1, a gradient-based approach is commonly used to solve the Euler-Lagrange equation. In other words, the first order optimally conditions are:

$$\frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = 0.$$

Recall that, the weights or parameters that we are looking for are, in principle, unbounded, this is $\boldsymbol{\omega} \in \mathbb{R}^k$. Consequently, the only constrain deemed will be introduced by a regularization term that comes from the prior distribution as it is done in subsection 2.2.1. This term must also be included in the Euler-Lagrange equation, but we avoid it for the sake of notation.

To illustrate the back-propagation algorithm is an efficient implementation of the old and well-known *chain rule*, we set a shallow and tiny neural network and deduce the *gradient* necessary to update the weights. Moreover, for simplicity, the output vector $\mathbf{y} \in \mathbb{R}^2$ will only have two components.

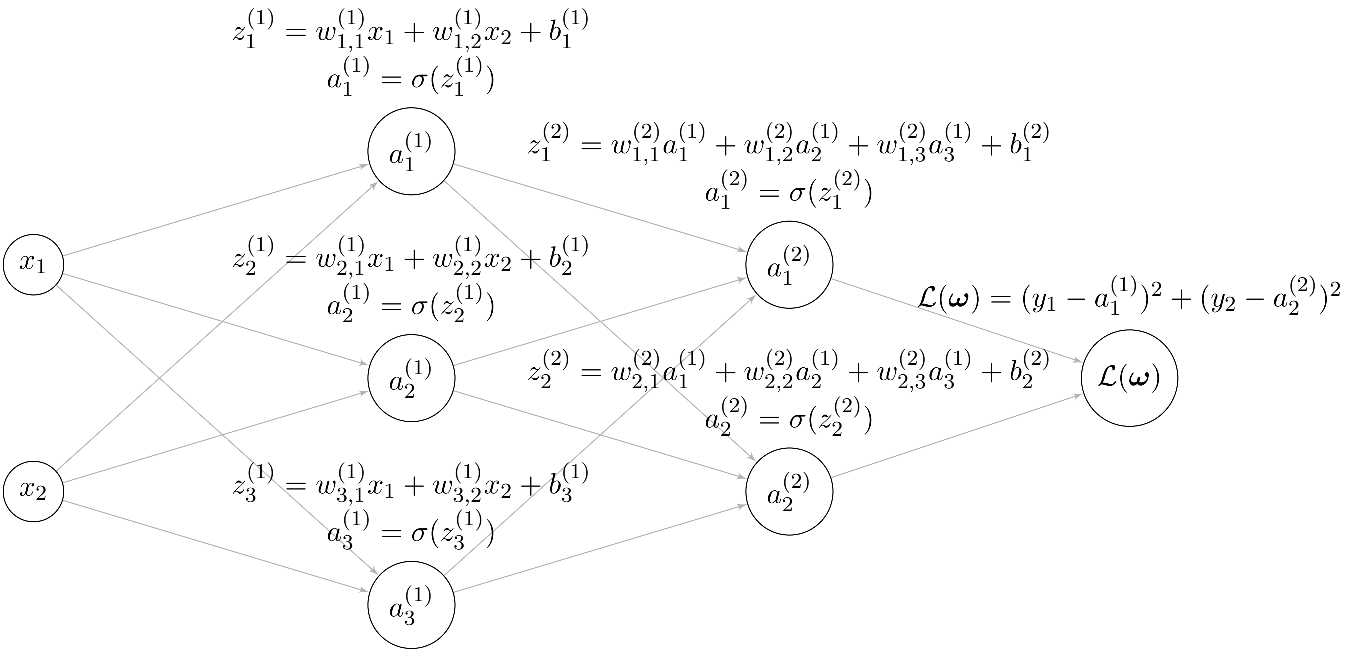

$z_1^{(1)} = w_{1,1}^{(1)}x_1 + w_{1,2}^{(1)}x_2 + b_1^{(1)}$
$a_1^{(1)} = \sigma(z_1^{(1)})$
$a_1^{(1)}$
$z_1^{(2)} = w_{1,1}^{(2)}a_1^{(1)} + w_{1,2}^{(2)}a_2^{(1)} + w_{1,3}^{(2)}a_3^{(1)} + b_1^{(2)}$
$a_1^{(2)} = \sigma(z_1^{(2)})$
x_1
$z_2^{(1)} = w_{2,1}^{(1)}x_1 + w_{2,2}^{(1)}x_2 + b_2^{(1)}$
$a_2^{(1)} = \sigma(z_2^{(1)})$
$a_1^{(2)}$
$\mathcal{L}(\boldsymbol{\omega}) = (y_1 - a_1^{(1)})^2 + (y_2 - a_2^{(2)})^2$
$a_2^{(1)}$
$z_2^{(2)} = w_{2,1}^{(2)}a_1^{(1)} + w_{2,2}^{(2)}a_2^{(1)} + w_{2,3}^{(2)}a_3^{(1)} + b_2^{(2)}$
$a_2^{(2)} = \sigma(z_2^{(2)})$
$\mathcal{L}(\boldsymbol{\omega})$
x_2
$z_3^{(1)} = w_{3,1}^{(1)}x_1 + w_{3,2}^{(1)}x_2 + b_3^{(1)}$
$a_3^{(1)} = \sigma(z_3^{(1)})$
$a_2^{(2)}$
$a_3^{(1)}$

Figure 2.10: Shallow Feedforward Neural Network

Figure 2.10 represents a Feedforward Neural Network in the most usual way, where each output $a_i^{(L)}$ in layer L depicts the non-linear activation (2.39). However, to easy and better understand the back-propagation algorithm it worth to separate theses outputs in two different nodes, i.e., the linear outputs $z_i^{(L)}$ and the activations $a_i^{(L)} = \sigma(z_i^{(L)})$ like in Figure 2.11. For this setting, the vector of parameters for the neural network 2.11, is

$$\begin{aligned}\boldsymbol{\omega} = & \quad (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}, \omega_{11}, \omega_{12}, \omega_{13}, \omega_{14}, \omega_{15}, \omega_{16}, \omega_{17})^T \\ = & \quad (w_{1,1}^{(1)}, w_{1,2}^{(1)}, w_{2,1}^{(1)}, w_{2,2}^{(1)}, w_{3,1}^{(1)}, w_{3,2}^{(1)}, b_1^{(1)}, b_2^{(1)}, b_3^{(1)}, w_{1,1}^{(2)}, w_{1,2}^{(2)}, w_{1,3}^{(2)}, w_{2,1}^{(2)}, w_{2,2}^{(2)}, w_{2,3}^{(2)}, b_1^{(2)}, b_2^{(2)})^T,\end{aligned}$$

that can also be seen as nodes and included in the graph. We then call the derivatives of the cost function w.r.t. the weights: *gradient* and denote it by

$$\begin{aligned}\nabla_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega}) = & \left(\frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial\omega_1}, \frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial\omega_2}, \cdots, \frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial\omega_{17}}\right)^T \\ = & \left(\frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial w_{1,1}^{(1)}}, \frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial w_{1,2}^{(1)}}, \cdots, \frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial b_2^{(2)}}\right)^T\end{aligned}$$

where the derivatives w.r.t. each parameter are:

$$\begin{aligned}\frac{\partial\mathcal{C}(\boldsymbol{\omega})}{\partial\omega_k} &= \frac{\partial}{\partial\omega_k}\frac{1}{N}\sum_i^N ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2 \\ &= \frac{\partial}{\partial\omega_k}\frac{1}{N}\sum_i^N\sum_j^n \left(y_j^{(i)} - f_j(\boldsymbol{\omega}, \mathbf{x}^{(i)})\right)^2.\end{aligned}$$

Besides, we will consider the *gradient* for a single sample as if $N = 1$ since the gradient with respect to the whole dataset is just the sum of each of them. For this reason we use the loss function $\mathcal{L}(\cdot)$ rather than the cost function $\mathcal{C}(\cdot)$,

$$\frac{\partial\mathcal{L}(\boldsymbol{\omega})}{\partial\boldsymbol{\omega}} = \nabla_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega})$$

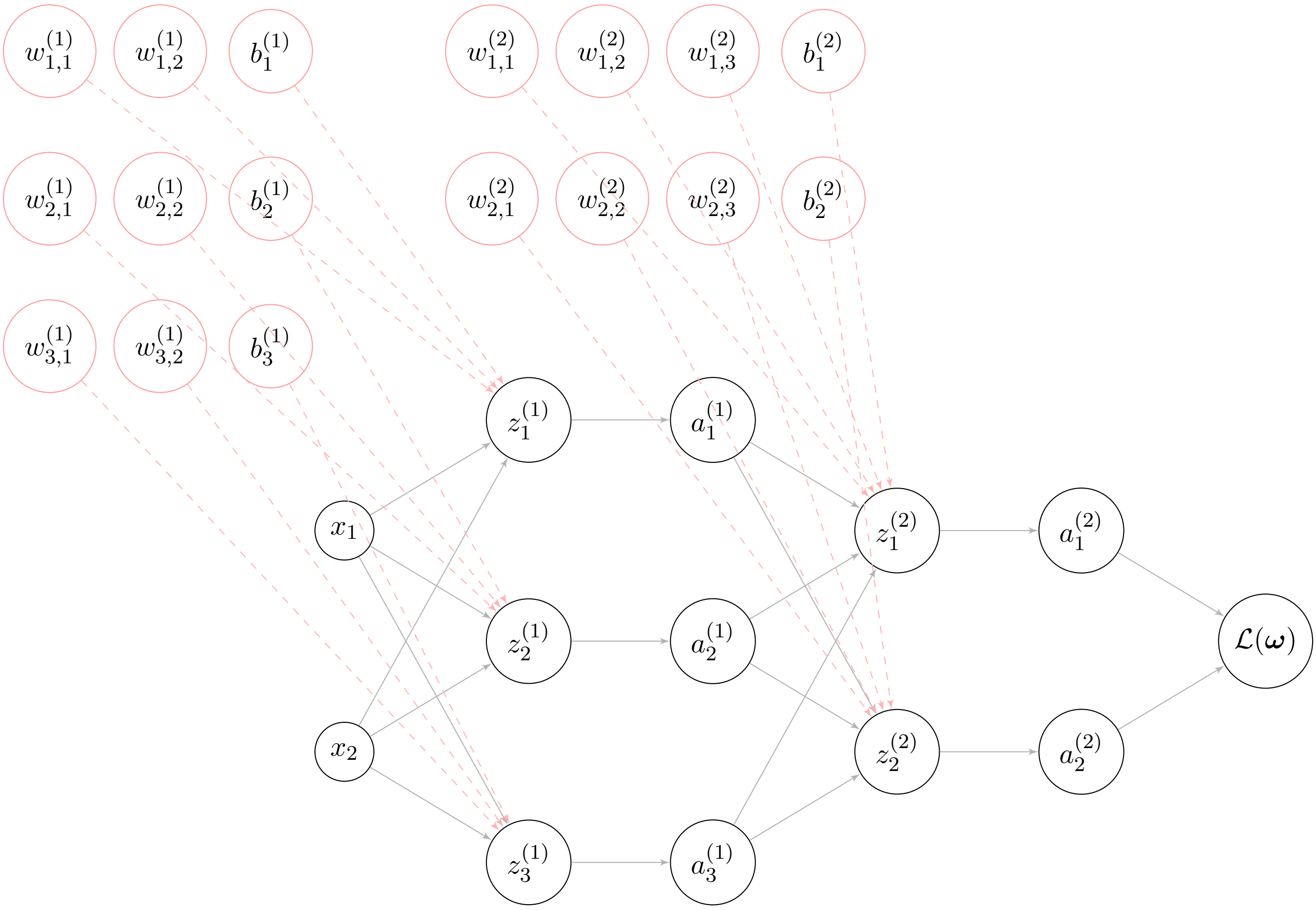


Figure 2.11: Unwrapped Feedforward Neural Network

this is, for each parameter $\boldsymbol{\omega}_k$ we have

$$\nabla_{\boldsymbol{\omega}} \mathcal{L}(\boldsymbol{\omega})_k = \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial \omega_k}.$$

As we set, the output has two components, so for each sample the loss function is

$$\mathcal{L}(\boldsymbol{\omega}) = (y_1 - f_1(\boldsymbol{\omega}, \mathbf{x}))^2 + (y_2 - f_2(\boldsymbol{\omega}, \mathbf{x}))^2$$

or equivalently,

$$\mathcal{L}(\boldsymbol{\omega}) = \left(y_1 - a_1^{(2)}\right)^2 + \left(y_2 - a_2^{(2)}\right)^2.$$

We now use the *chain rule* to compute the derivative of the loss function w.r.t. a parameter, for instance, $\omega_{10} = w_{1,1}^{(2)}$; this is

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial w_{1,1}^{(2)}} &= \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{1,1}^{(2)}} + \cancelto{0}{\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} \frac{\partial a_2^{(2)}}{\partial w_{1,1}^{(2)}}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial w_{1,1}^{(2)}} \\
&= -2\left(y_1 - a_1^{(2)}\right) \sigma'(z_1^{(2)}) a_1^{(1)}
\end{aligned}$$

where

$$
\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} &= \frac{\partial}{\partial a_1^{(2)}} \left(y_1 - a_1^{(2)}\right)^2 = -2\left(y_1 - a_1^{(2)}\right) \\
\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} &= \frac{\partial}{\partial z_1^{(2)}} \sigma(z_1^{(2)}) = \sigma^{'}(z_1^{(2)}) \\
\frac{\partial z_1^{(2)}}{\partial w_{1,1}^{(2)}} &= \frac{\partial}{\partial w_{1,1}^{(2)}} \left(w_{1,1}^{(2)} a_1^{(1)} + w_{1,2}^{(2)} a_2^{(1)} + w_{1,3}^{(2)} a_3^{(1)} + b_1^{(2)}\right) = a_1^{(1)}
\end{aligned}
$$

Each requited partial derivative, in fact, describes the path from the output to the weight in the graph 2.12. Because of the quadratic loss function term, the derivative corresponds to the *error* between the real label **y** and the predicted output $\mathbf{a}^{(2)} = \mathbf{f}(\boldsymbol{\omega}, \mathbf{x})$. Thus, each partial derivative is commonly seen in Machine Learning as the error between nodes or layers. For such a reason it is said that back-propagation algorithm consists of propagating the error from the output to the inputs (backward).

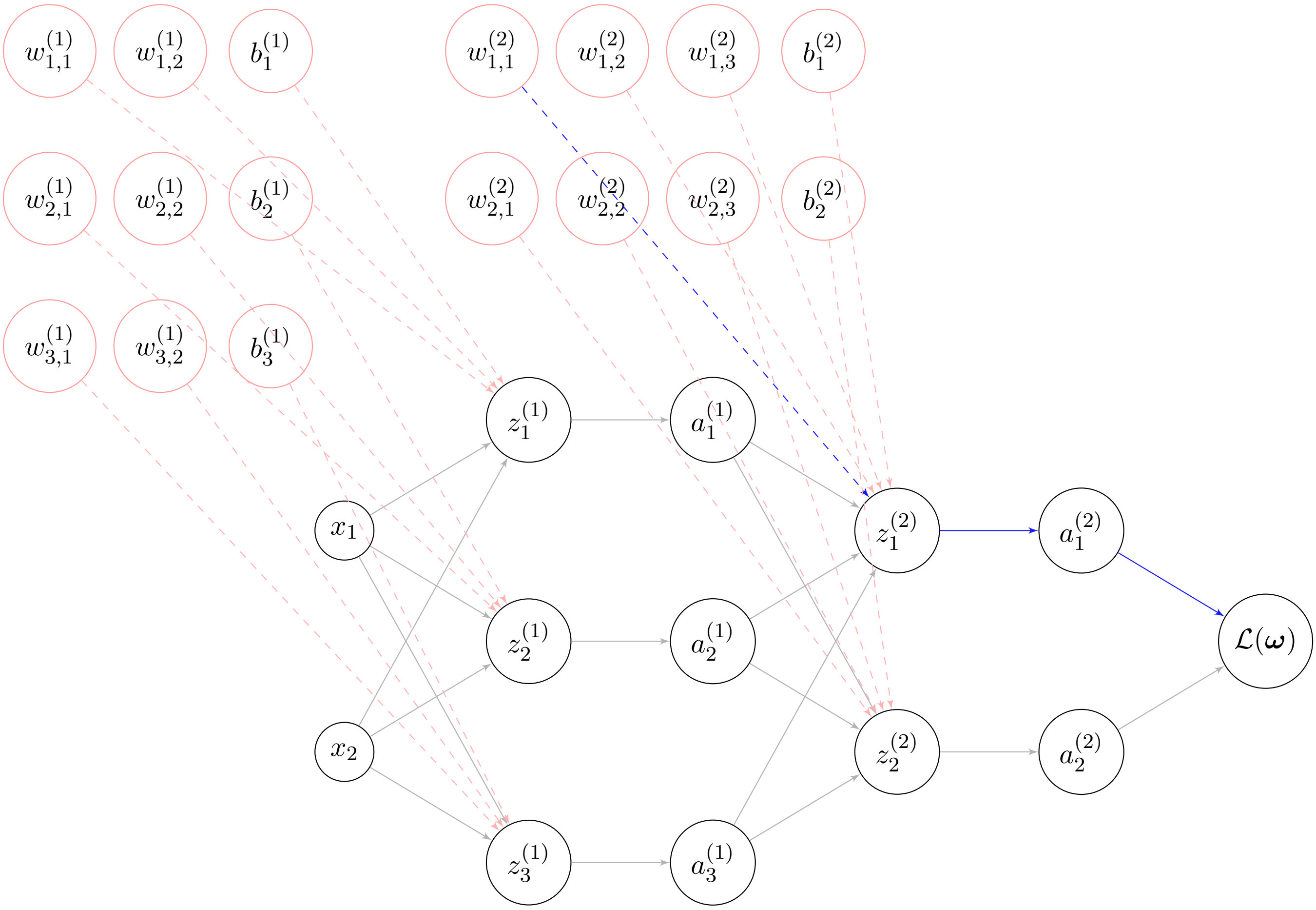


Figure 2.12: Path from the output to a last layer weight.

In the same way, we derive with respect to the weight $\omega_1 = w_{1,1}^{(1)}$ from the first layer, and notice that there exists 2 different paths 2.13. The partial derivatives that are shared in both paths can be factorized to avoid redundancy in their computations. But, even more important is the fact that several partial derivatives have already been computed in the previous layers.

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial w_{1,1}^{(1)}} &= \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{1,1}^{(1)}} + \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} \frac{\partial a_2^{(2)}}{\partial w_{1,1}^{(1)}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial w_{1,1}^{(1)}} + \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial w_{1,1}^{(1)}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} \frac{\partial a_1^{(1)}}{\partial w_{1,1}^{(1)}} + \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_1^{(1)}} \frac{\partial a_1^{(1)}}{\partial w_{1,1}^{(1)}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}} + \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_1^{(1)}} \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}} \\
&= \left(\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} + \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial a_1^{(1)}} \right) \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}} \\
&= -2 \left[\left(y_1 - a_1^{(2)} \right) \sigma^{'}(z_1^{(2)}) w_{1,1}^{(1)} + \left(y_2 - a_2^{(2)} \right) \sigma^{'}(z_2^{(2)}) w_{2,1}^{(1)} \right] \sigma^{'}(z_1^{(1)}) x_1
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_1^{(2)}} &= \frac{\partial}{\partial a_1^{(2)}} \left(y_1 - a_1^{(2)} \right)^2 = -2 \left(y_1 - a_1^{(2)} \right) \\
\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} &= \frac{\partial}{\partial z_1^{(2)}} \sigma(z_1^{(2)}) = \sigma^{'}(z_1^{(2)}) \\
\frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} &= \frac{\partial}{\partial a_1^{(1)}} \left(w_{1,1}^{(2)} a_1^{(1)} + w_{1,2}^{(2)} a_2^{(1)} + w_{1,3}^{(2)} a_3^{(1)} + b_1^{(2)} \right) = w_{1,1}^{(2)} \\
\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial a_2^{(2)}} &= \frac{\partial}{\partial a_2^{(2)}} \left(y_2 - a_2^{(2)} \right)^2 = -2 \left(y_2 - a_2^{(2)} \right) \\
\frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} &= \frac{\partial}{\partial z_2^{(2)}} \sigma(z_2^{(2)}) = \sigma^{'}(z_2^{(2)}) \\
\frac{\partial z_2^{(2)}}{\partial a_1^{(1)}} &= \frac{\partial}{\partial a_1^{(1)}} \left(w_{2,1}^{(2)} a_1^{(1)} + w_{2,2}^{(2)} a_2^{(1)} + w_{2,3}^{(2)} a_3^{(1)} + b_2^{(2)} \right) = w_{2,1}^{(2)} \\
\frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} &= \frac{\partial}{\partial z_1^{(1)}} \sigma(z_1^{(1)}) = \sigma^{'}(z_1^{(1)}) \\
\frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}} &= \frac{\partial}{\partial w_{1,1}^{(1)}} w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 + b_1^{(1)} = x_1
\end{aligned}$$

It is easy to see that the systematic way of computing each derivative in the *gradient* vector is throughout the paths from the output to the input. If we compute each component of the *gradient* $\nabla_{\boldsymbol{\omega}\mathcal{L}(\boldsymbol{\omega})}$ independently, many terms will be computed several times increasing the redundancy. For such a reason, back-propagation computes all the derivatives once, in two different steps: *forward* step and *backward* step. Furthermore, each step can be rearranged and computed using matrix products.

Jacobian Matrix

The key-point for an efficient computation of the back-propagation algorithm is to compute the Jacobian matrix for each node, only once. Indeed, the Jacobian matrix is defined by the first order partial derivatives

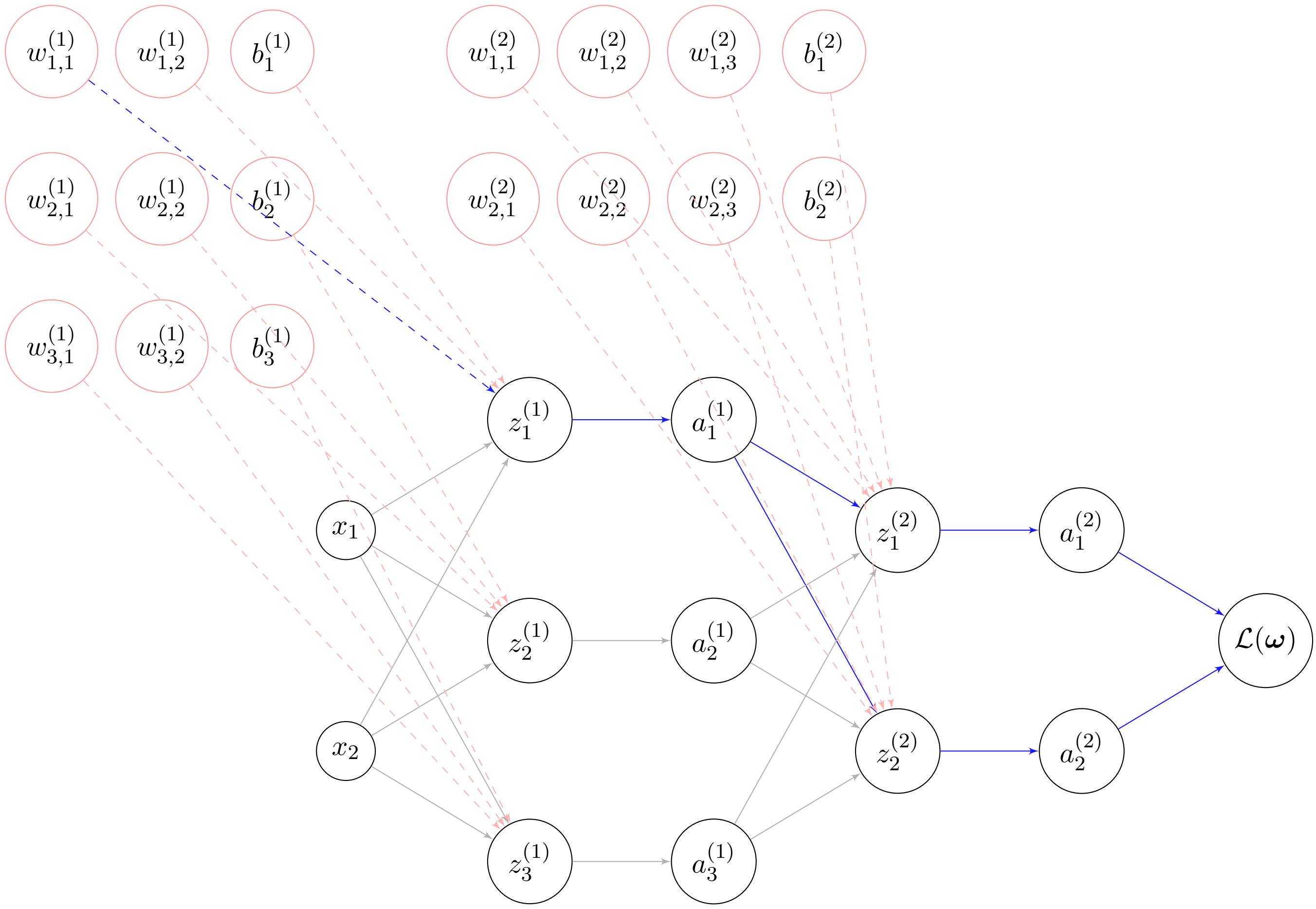


Figure 2.13: Paths from the output to a first layer weight

of a function such that $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$. This is, the Jacobian matrix of a function $\mathbf{f}(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^m$, is:

$$J_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \dots & \dfrac{\partial f_1}{\partial x_1} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \dots & \dfrac{\partial f_2}{\partial x_m} \\ \dfrac{\partial f_3}{\partial x_1} & \dfrac{\partial f_3}{\partial x_2} & \dots & \dfrac{\partial f_3}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \dfrac{\partial f_n}{\partial x_2} & \dots & \dfrac{\partial f_n}{\partial x_m} \end{bmatrix},$$

which in turn corresponds to a generic layer. Computing the Jacobian matrix for each layer until the weights are reached led us to the *gradient* of the functional. Taking the previous example of a feedforward neural network, we have:

$$J_{\mathcal{L}}(\mathbf{a}^{(2)}) = \begin{bmatrix} \frac{\partial J}{\partial a^{(2)}_1} & \frac{\partial J}{\partial a^{(2)}_2} \end{bmatrix}, J_{\mathbf{a}^{(2)}}(\mathbf{z}^{(2)}) = \begin{bmatrix} \frac{\partial a^{(2)}_1}{\partial z^{(2)}_1} & \frac{\partial a^{(2)}_1}{\partial z^{(2)}_2} \\ \frac{\partial a^{(2)}_2}{\partial z^{(2)}_1} & \frac{\partial a^{(2)}_2}{\partial z^{(2)}_2} \end{bmatrix},$$

$$J_{\mathbf{z}^{(2)}}(\mathbf{a}^{(1)}) = \begin{bmatrix} \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} & \frac{\partial z_1^{(2)}}{\partial a_2^{(1)}} & \frac{\partial z_1^{(2)}}{\partial a_3^{(1)}} \\ \frac{\partial z_2^{(2)}}{\partial a_1^{(1)}} & \frac{\partial z_2^{(2)}}{\partial a_2^{(1)}} & \frac{\partial z_2^{(2)}}{\partial a_3^{(1)}} \end{bmatrix}, J_{\mathbf{a}^{(1)}}(\mathbf{z}^{(1)}) = \begin{bmatrix} \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} & \frac{\partial a_1^{(1)}}{\partial z_2^{(1)}} & \frac{\partial a_1^{(1)}}{\partial z_3^{(1)}} \\ \frac{\partial a_2^{(1)}}{\partial z_1^{(1)}} & \frac{\partial a_2^{(1)}}{\partial z_2^{(1)}} & \frac{\partial a_2^{(1)}}{\partial z_3^{(1)}} \\ \frac{\partial a_3^{(1)}}{\partial z_1^{(1)}} & \frac{\partial a_3^{(1)}}{\partial z_2^{(1)}} & \frac{\partial a_3^{(1)}}{\partial z_3^{(1)}} \end{bmatrix}$$

and also the Jacobian matrices w.r.t. the parameters:

$$J_{\mathbf{z}^{(2)}}(\boldsymbol{\omega}^{(2)}) = \begin{bmatrix} \frac{\partial z_1^{(2)}}{\partial w_{1,1}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{1,2}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{1,3}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial b_1^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{2,1}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{2,3}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial b_2^{(2)}} \\ \frac{\partial z_2^{(2)}}{\partial w_{1,1}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{1,2}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{1,3}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial b_1^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{2,1}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{2,3}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial b_2^{(2)}} \end{bmatrix},$$

$$J_{\mathbf{z}^{(1)}}(\boldsymbol{\omega}^{(1)}) = \begin{bmatrix} \frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial w_{1,2}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial b_1^{(1)}} & \frac{\partial z_1^{(1)}}{\partial w_{2,1}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_1^{(1)}}{\partial b_2^{(1)}} & \frac{\partial z_1^{(1)}}{\partial w_{3,1}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial w_{3,2}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial b_3^{(1)}} \\ \frac{\partial z_2^{(1)}}{\partial w_{1,1}^{(1)}} & \frac{\partial z_2^{(1)}}{\partial w_{1,2}^{(1)}} & \frac{\partial z_2^{(1)}}{\partial b_1^{(1)}} & \frac{\partial z_2^{(1)}}{\partial w_{2,1}^{(1)}} & \frac{\partial z_2^{(1)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_2^{(1)}}{\partial b_2^{(1)}} & \frac{\partial z_2^{(1)}}{\partial w_{3,1}^{(1)}} & \frac{\partial z_2^{(1)}}{\partial w_{3,2}^{(1)}} & \frac{\partial z_2^{(1)}}{\partial b_3^{(1)}} \\ \frac{\partial z_3^{(1)}}{\partial w_{1,1}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial w_{1,2}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial b_1^{(1)}} & \frac{\partial z_3^{(1)}}{\partial w_{2,1}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_3^{(1)}}{\partial b_2^{(1)}} & \frac{\partial z_3^{(1)}}{\partial w_{3,1}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial w_{3,2}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial b_3^{(1)}} \end{bmatrix}.$$

Notice that, in the activation steps, there exist no dependencies between cross outputs, this is, we can a priori set zeros in those matrix coefficient (derivatives):

$$J_{\mathbf{a}^{(2)}}(\mathbf{z}^{(2)}) = \begin{bmatrix} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} & 0 \\ 0 & \frac{\partial a_2^{(2)}}{\partial z_2^{(2)}} \end{bmatrix}$$

$$J_{\mathbf{a}^{(1)}}(\mathbf{z}^{(1)}) = \begin{bmatrix} \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} & 0 & 0 \\ 0 & \frac{\partial a_2^{(1)}}{\partial z_2^{(1)}} & 0 \\ 0 & 0 & \frac{\partial a_3^{(1)}}{\partial z_3^{(1)}} \end{bmatrix}$$

$$J_{\mathbf{z}^{(2)}}(\boldsymbol{\omega}^{(2)}) = \begin{bmatrix} \frac{\partial z_1^{(2)}}{\partial w_{1,1}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{1,2}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial w_{1,3}^{(2)}} & \frac{\partial z_1^{(2)}}{\partial b_1^{(2)}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial z_2^{(2)}}{\partial w_{2,1}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial w_{2,3}^{(2)}} & \frac{\partial z_2^{(2)}}{\partial b_2^{(2)}} \end{bmatrix},$$

$$J_{\mathbf{z}^{(1)}}(\boldsymbol{\omega}^{(1)}) = \begin{bmatrix} \frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial w_{1,2}^{(1)}} & \frac{\partial z_1^{(1)}}{\partial b_1^{(1)}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial z_2^{(1)}}{\partial w_{2,1}^{(1)}} & \frac{\partial z_2^{(1)}}{\partial w_{2,2}^{(2)}} & \frac{\partial z_2^{(1)}}{\partial b_2^{(1)}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial z_3^{(1)}}{\partial w_{3,1}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial w_{3,2}^{(1)}} & \frac{\partial z_3^{(1)}}{\partial b_3^{(1)}} \end{bmatrix}.$$

Finally, the *gradient* of the whole neural network w.r.t. the weights can be obtained as follows:

$$\nabla_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega}) = \begin{bmatrix} J_{\mathcal{L}}(\boldsymbol{\omega}^{(1)})^T \\ J_{\mathcal{L}}(\boldsymbol{\omega}^{(2)})^T \end{bmatrix} = \begin{bmatrix} \left(J_{\mathcal{L}}(\mathbf{a}^{(2)})J_{\mathbf{a}^{(2)}}(\mathbf{z}^{(2)})J_{\mathbf{z}^{(2)}}(\boldsymbol{\omega}^{(2)})\right)^T \\ \left(J_{\mathcal{L}}(\mathbf{a}^{(2)})J_{\mathbf{a}^{(2)}}(\mathbf{z}^{(2)})J_{\mathbf{z}^{(2)}}(\boldsymbol{\omega}^{(2)})J_{\mathbf{z}^{(2)}}(\mathbf{a}^{(1)})J_{\mathbf{a}^{(1)}}(\mathbf{z}^{(1)})J_{\mathbf{z}^{(1)}}(\boldsymbol{\omega}^{(1)})\right)^T \end{bmatrix}$$

The gradient descent step is then defined as usual, updating the weights iteratively:

$$\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t - lr \cdot \nabla_{\boldsymbol{\omega}^t}\mathcal{L}(\boldsymbol{\omega}^t). \tag{2.46}$$

Optimizers

We have seen how to obtain through the back-propagation algorithm the gradient of the cost functional. The traditional scheme (2.46) has, however, different modifications that are wide used by the Deep Learning community. Those *optimizers*, an abstracted concept that allows us to find critical points (solutions) for an optimization problem, can be classified first based on the number of samples considered at each update iteration. In that sense we have:

1. Batch (Gradient Descent): the gradient computed considers the whole dataset (all samples).

$$\frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{\partial}{\partial \boldsymbol{\omega}} \frac{1}{N} \sum_{i}^{N} ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2$$

2. Mini-Batch (Gradient Descent): the train set is randomly split in several subsets.

$$\frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{\partial}{\partial \boldsymbol{\omega}} \frac{1}{B} \sum_{i}^{B} ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2$$

   where $B \subset N$.

3. Stochastic (Gradient Descent) - SGD: the computed gradient is obtained for each sample individually, i.e., $B = 1$

$$\frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{\partial}{\partial \boldsymbol{\omega}} ||\mathbf{y}^{(i)} - \mathbf{f}(\boldsymbol{\omega}, \mathbf{x}^{(i)})||_2^2$$

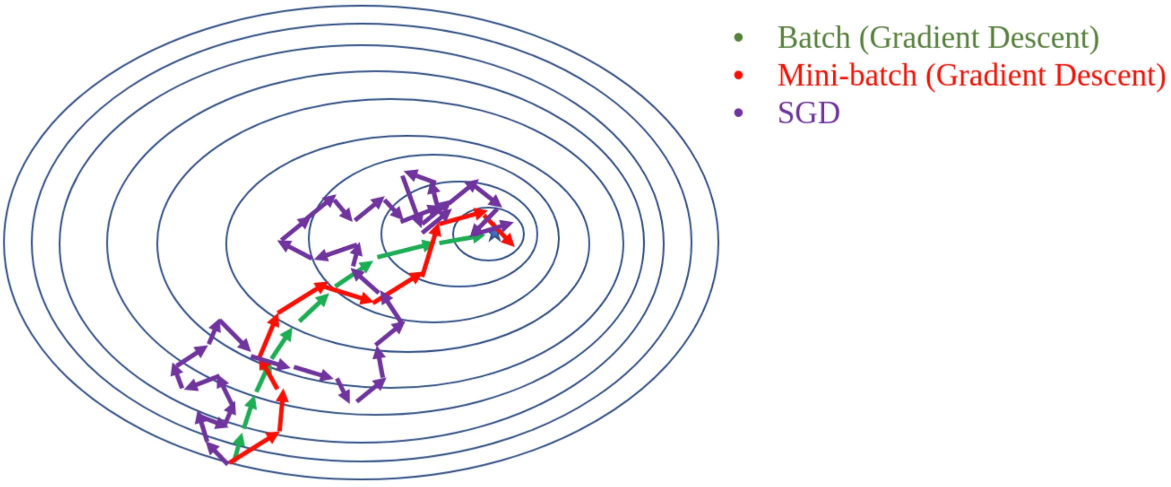


Figure 2.14: Comparison: Batch, Mini-batch and Stochastic Gradient Descent

The main difference among these classification options is the number of samples used to compute the gradient needed to update the parameters. The effect, as depicted in Figure 2.14 is a trade-off (Mini-batch) between a smooth and stable trajectory (Gradient Descent) to the critical point, and a more exploratory behaviour (Stochastic Gradient Descent (SGD)). However, has difficulties when the critical point is a saddle point due to the fact it is surrounded by a flat region, an thus, each direction seem to have 0 slope. Such saddle points are a common issue in DL optimization problems [Dauphin et al., 2014]. The main good of the Gradient Descent is its convergence property although in practice it does not seem to be very significant (all the local critical points are quite similar in terms of energy). Moreover, it is an expensive approach in terms of memory since all the marginal gradient w.r.t. the samples must be stored. On the contrary, with Mini-batch and SGD, the required memory is significantly reduced (by the number of samples in the mini-batch) allowing a faster update weight iteration, improving the results if combined with some momentum terms (by escaping from *plateau* regions) and reducing over-fitting effects. Contrary to the Gradient Descent algorithm, SGD allows for online optimization training since they take one sample at a time.

A second classification focus on modifying the gradient. Some of these changes consist of tweaking the *learning rate* or time step lr from equation (2.46). Once the gradient has been computed, different terms can be added to modify its direction according to previous values by creating inertial components that may improved the results (accuracy) or accelerate the convergence of the algorithm. Among others, we can highlight:

1. AdaGrad [Duchi et al., 2011]: this algorithm adjusts the *learning rate* inversely proportional to the past gradient energy. A pseudocode is depicted in Algorithm 2.1.

   Algorithm 2.1: AdaGrad

   ```
   Set: $lr, \delta = 1e-7$
   Initialize: $\mathbf{r} = 0$
   while not converged do
       $\mathbf{g} \leftarrow \frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}$
       $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}$
       $\boldsymbol{\omega} = \boldsymbol{\omega} - lr \frac{1}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$
   ```

   It has good convergence properties when the considered error surface is convex, however, in practice it suffers from an excessive vanishing *learning rate*.

2. RMSProp [Hinton et al.,]: this algorithm modifies the previous AdaGrad to improve the behaviour when dealing with non-convex functionals. To this end, it introduces a moving average factor for the accumulated gradient. A pseudocode is depicted in Algorithm 2.2.

   Algorithm 2.2: RMSProp

   ```
   Set: $lr, \delta = 1e-6, \rho$
   Initialize: $\mathbf{r} = 0$
   while not converged do
       $\mathbf{g} \leftarrow \frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}$
       $\mathbf{r} \leftarrow \rho \mathbf{r} + (1-\rho)\mathbf{g} \odot \mathbf{g}$
       $\boldsymbol{\omega} = \boldsymbol{\omega} - lr \frac{1}{\sqrt{\delta + \mathbf{r}}} \odot \mathbf{g}$
   ```

3. Adam [Kingma and Ba, 2014]: this algorithm in turn extend the RMSProp by considering two momentum terms. While RMSProp only takes into account the second order moment of the gradient with an exponential decay, Adam adds a first order term over the gradient. A pseudocode is depicted in Algorithm 2.3.

Algorithm 2.3: Adam

```
Set: $lr, \delta = 1e-8, \beta_1 = 0.9, \beta_2 = 0.999$
Initialize: $\mathbf{r} = 0, \mathbf{s} = 0$
while not converged do
    $\mathbf{g} \leftarrow \frac{\partial \mathcal{C}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}$
    $\mathrm{t} \leftarrow t + 1$
    $\mathbf{s} \leftarrow \beta_1 \mathbf{s} + (1 - \beta_1)\mathbf{g}$
    $\mathbf{r} \leftarrow \beta_2 \mathbf{r} + (1 - \beta_2)\mathbf{g} \odot \mathbf{g}$
    $\hat{\boldsymbol{s}} \leftarrow \frac{\mathbf{s}}{1-\beta_1^t}$
    $\hat{\boldsymbol{r}} = \frac{\mathbf{r}}{1-\beta_2^t}$
    $\boldsymbol{\omega} = \boldsymbol{\omega} - lr \frac{\hat{\boldsymbol{s}}}{\delta + \sqrt{\hat{\boldsymbol{r}}}} \odot \mathbf{g}$
```

Another interesting modification based on momentum is the so-called Nesterov Accelerated Gradient (NAG) proposed by Yurii Nesterov in [Nesterov, 1983]. The basic idea comes to solve the problem of accumulating too much inertia (a ball rolling down a hill) that when the solution gets to the minimum, such inertia forces it to move away. NAG helps to overcome this problem by anticipating the next step using the momentum term and thus adapt the next gradient step according to it.

All these modifications aim to improve the finding of optimal solutions and the time of convergence to them, however, they are all based on the computation of first order optimal conditions (gradient). The natural next step is then to consider second order derivatives using methods like Newton, Broyden–Fletcher–Goldfarb–Shanno (BFGS) or the limited memory L-BFGS that require computing Hessian matrices. In DL, all these powerful methods could theoretically be used but the computational burden make them nowadays unfeasible since a neural network can easily have hundreds of thousands of parameters. An alternative to a more advanced optimizer that has not yet been exploited to a large extent is primal-dual schemes or primal-dual neural networks. Some speculative reasons point to the fact that the improvement in neural networks relies in the architecture rather than obtaining better critical points.

CHAPTER **3**

Modeling - Bayesian Inference

. . . I am a good Hegelian. If you have a good theory, forget about the reality.
Slavoj Žižek

In this chapter we show how both Variational Methods and Deep Learning problems for Image Processing can be addressed from a more general and thus abstract perspective, based on Bayesian Inference. In the following we will describe what is understood by Bayesian Inference and how each problem can be derived as a particular instance.

The methodology described in this chapter can be used to address the proposed problems in this dissertation. In chapter 4 we propose two pure variational models for saliency detection and their numerical resolutions. One of them is embedded in a CNN and end-to-end trained, overcoming the manual hyper-parameterization tunning. In chapter 5 we aim to solve a classification problem based on DL techniques and propose two methods for a semi-supervised training procedure allowing to train with very little data (real case scenarios). We also show the typical problems that arises in such architectures and link them to the lack of Bayesian modeling. Finally, in chapter 6 a very ill-posed problem is faced from a Bayesian perspective that justifies and provides hints to correctly get advantage of all new techniques and tricks that can be found in recent literature.

Variational methods and Deep Learning techniques are commonly seen as different approaches to possibly similar problems, mainly if the field of interest is, as deemed in this thesis, Computer Vision. To motivate why a Bayesian Inference approach is useful to address any problem in CV and other fields, we first focus on the main drawbacks of both Variational and Deep Learning tools.

On the one hand, Variational models require priors and a hypothesis model to describe and analyze the theoretical properties and provide numerical resolutions. The performance of the resulting models relies on two decisions: (1) a correct selection of adequate priors and (2) a correct balance between them. As we have seen in Section 2.1, the success of prior operators, which are ubiquitous in Restoration Problems and Image Processing tasks, is explained through the nature of data (images). In that sense, the correct selection of priors (as the celebrated TV operator in [Rudin et al., 1992]) is driven by the inner properties of the target images (smooth regions and sharp edges). This is the main contribution of many Variational models for Image Processing, however, since they are parametric models, it is often the case that finding a correct hyper-parameterization is unfeasible for a non-expert user and thus the model becomes impracticable. Here is where data take importance. In fact, if data are available, the hyper-parameters can be found via optimization or meta-algorithms that provide a static setting of such parameters. Nonetheless, this is clearly sub-optimal when this setting of parameters depends on the concrete example. We propose in chapter 4 a Deep Variational Framework to boost our proposed variational model and get rid of the manual tuning and finding the optimal hyper-parameters based on data. Interestingly, data does not only allow for automatically tuning parameters but also open the possibility to *learn* the required priors [Chen et al., 2014]. These methods are often solved through bi-level algorithms [Colson et al., 2007], which have a great resemblance with the DL formulation when some regularizing priors are included.

On the other hand, DL and NNs suffer from the so-called *black box* effect, when they are treated as black boxes. In fact, uninterpretability of neural networks is a shortcoming that reduces the control of these systems and increase the uncertainty of their outcomes. Somehow, NNs circumvents this fact by relying on a huge amount of data, that is supposed to cover and be representative of the whole expected distribution of examples. We are thus lead to focus on data. We have already seen in chapter 2 that NNs are Universal Approximation Functions, and therefore they can "mimic" almost every *reasonable* function, see Theorem 1. However, this theorem does not provides a way to obtain such function and, even more important, assumes infinite examples are available. In this framework, over-fitting and extrapolation problems arise. The lack of information, either for unavailable data or a non-representative sub-sampled set of examples, must be addressed in terms of regularization, i.e., imposing priors. That reveals a pure Maximum Likelihood (ML) approach is not enough even if a big amount of data has been collected.

3.1 Bayesian Inference

Bayesian Inference is a Statistical Inference method that rests on Bayes' Rule, used to update a prior model probability with evidence [Casella and Berger, 2002]. The hypothesis or model about the world is constantly evolving as new experiments are performed and thus data are coming. This is, in a more general framework as those presented in chapter 2:

$$\mathrm{p}(M|D) = \frac{\mathrm{p}(M,D)}{\mathrm{p}(D)} \tag{3.1}$$

where M represents the model and D the data evidence, usually $D = (\mathbf{X}, \mathbf{Y})$, being $\mathbf{X}$ and $\mathbf{Y}$ random variables of inputs and outputs. In this sense we are interpreting M and D as random variables, and therefore, the outcomes represent a set of models and data points respectively. Narrowing the models as functions that transforms some input domain into a co-domain $M = \mathbf{F} : \mathbf{x} \sim \mathrm{p}(\mathbf{X}) \rightarrow \mathbf{y} \sim \mathrm{p}(\mathbf{Y})$, where depending on whether $\mathbf{x}$ and $\mathbf{y}$ are causes or effects, we will be facing a direct or inverse problems, as explained in Section 2.1. More interesting in that scheme is that the model or function used to obtain the *prediction* $\mathbf{y}$ given the data point $\mathbf{x}$ is randomly selected, i.e., we draw samples from $\mathbf{f} \sim \mathrm{p}(\mathbf{F})$. Re-writing (3.1) we have:

$$\mathrm{p}(\mathbf{F} \mid \mathbf{X}, \mathbf{Y}) = \frac{\mathrm{p}(\mathbf{F}, \mathbf{X}, \mathbf{Y})}{\mathrm{p}(\mathbf{X}, \mathbf{Y})} = \frac{\mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{X})\mathrm{p}(\mathbf{F}, \mathbf{X})}{\mathrm{p}(\mathbf{X}, \mathbf{Y})} = \frac{\mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{X})\cancel{\mathrm{p}(\mathbf{X} \mid \mathbf{F})}\mathrm{p}(\mathbf{F})}{\mathrm{p}(\mathbf{Y} \mid \mathbf{X})\cancel{\mathrm{p}(\mathbf{X})}} \tag{3.2}$$

where $\mathrm{p}(\mathbf{X} \mid \mathbf{F}) = \mathrm{p}(\mathbf{X})$ due to the fact $\mathbf{X}$ does not depend on $\mathbf{F}$, resulting in:

$$\mathrm{p}(\mathbf{F} \mid \mathbf{X}, \mathbf{Y}) = \frac{\mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{X})\mathrm{p}(\mathbf{F})}{\mathrm{p}(\mathbf{Y} \mid \mathbf{X})}. \tag{3.3}$$

In general, computing the joint probability $\mathrm{p}(\mathbf{F}, \mathbf{X}, \mathbf{Y})$ is expensive. Alternatively, to facilitate such joint probability term to be computed, the whole system can be seen as a DAG. Those models, also known as or , harness of their representations exploiting dependencies between nodes to reduce greatly the number of term that require to be computed. For a comprehensive introduction and further details see [Jensen et al., 1996], [Jordan, 1998, Jordan et al., 1999] and [Nielsen and Jensen, 2009]. For variational methods and graphical models for modern machine learning approaches, see [Bishop, 1998].

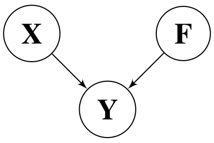


Figure 3.1: DAG Model

In this case, $\mathrm{p}(\mathbf{F}, \mathbf{X}, \mathbf{Y})$ is obtained as the product of the probability of each node conditioned to its *parents*:

- Node $\mathbf{X}$: has not *parent* nodes → $\mathrm{p}(\mathbf{X})$
- Node $\mathbf{F}$: has not *parent* nodes → $\mathrm{p}(\mathbf{F})$
- Node $\mathbf{Y}$: has two *parent* nodes → $\mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{X})$

which results in:

$$\mathrm{p}(\mathbf{F}, \mathbf{X}, \mathbf{Y}) = \mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{X})\mathrm{p}(\mathbf{X})\mathrm{p}(\mathbf{F}).$$

Moreover, the denominator in equation (3.2) $\mathrm{p}(\mathbf{X}, \mathbf{Y})$ associated to the probability of the *evidence*, using Bayes' rule, can be write as:

$$\mathrm{p}(\mathbf{X}, \mathbf{Y}) = \begin{cases} \mathrm{p}(\mathbf{Y} \mid \mathbf{X})\mathrm{p}(\mathbf{X}) \\ \text{or} \\ \mathrm{p}(\mathbf{X} \mid \mathbf{Y})\mathrm{p}(\mathbf{Y}) \end{cases}$$

where it is easy to see that the correct election is the first by simply looking at the graph of figure (3.1). Equation (3.3) can be seen as finding a distribution of functions $\mathbf{F}$ that are likely to generate the output distribution of labels $\mathbf{Y}$ given a input distribution of points $\mathbf{X}$. Once those distributions are obtained we can use Bayesian Inference to make estimates in a possible new test dataset $D_{test} = (\mathbf{X}_{test}, \mathbf{Y}_{test})$:

$$\mathbf{x}_{test} \sim \mathrm{p}(\mathbf{X}_{test}),$$

$$\mathbf{y}_{test} \sim \mathrm{p}(\mathbf{Y}_{test}).$$

Then, doing *Inference* is equivalent to use posterior probability (3.1) and the *prediction equation* defined as follows:

$$\mathrm{p}(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{X}, \mathbf{Y}) = \int \mathrm{p}(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{F})\mathrm{p}(\mathbf{F} \mid \mathbf{X}, \mathbf{Y})\mathrm{d}\mathbf{F}. \tag{3.4}$$

Notice that for such purpose, it is necessary to sample from the posterior distribution of the model (3.3):

$$\mathbf{f} \sim \mathrm{p}(\mathbf{F} \mid \mathbf{X}, \mathbf{Y}).$$

For all these models $\mathbf{F}$, we will integrate each prediction to infer a density distribution function of observing a single data point $\mathbf{y}^*$ given $\mathbf{x}^*$. The prior prediction is however computed as a *marginalisation*, i.e., computing the integral over the prior models:

$$\mathrm{p}(\mathbf{Y} \mid \mathbf{X}) = \int \mathrm{p}(\mathbf{Y} \mid \mathbf{X}, \mathbf{F})\mathrm{p}(\mathbf{F})\mathrm{d}\mathbf{F} \tag{3.5}$$

constitutes the normalizing term in equation (3.3). Further, all models shall be characterized as parametric functions, which means that all the set of functions in $\mathbf{F}$ relies on some parameters $\mathbf{W}$ which in turn are also drawn from a distribution function:

$$\begin{aligned} \mathbf{f}(\mathbf{W}) &\sim \mathrm{p}(\mathbf{F}) \\ \mathbf{w} &\sim \mathrm{p}(\mathbf{W}). \end{aligned}$$

tranforming equation ((3.4)) in:

$$\mathrm{p}(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{X}, \mathbf{Y}) = \int \mathrm{p}(\mathbf{y}_{test} \mid \mathbf{F})\mathrm{p}(\mathbf{F} \mid \mathbf{x}_{test}, \mathbf{W})\mathrm{p}(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})\mathrm{d}\mathbf{F}\mathrm{d}\mathbf{W}. \tag{3.6}$$

To give a simple intuition of what the set of functions $\mathbf{F}$ and the set of parametrizations $\mathbf{W}$ depicts, one can think of it as NN with different architectures and each of them with plenty of different weights configurations. In ML and DL, the deemed functions are NN that, as we have seen in Section 2.2, are *Universal Approximation Functions* and therefore, roughly speaking, each of this instance of NN (assuming enough capacity, i.e., nodes) is able to approximate "any" function. For such a reason we can integrate (3.6) over $\mathbf{F}$ or, alternatively, consider that the set of functions $\mathbf{F}$ is composed by only 1 element s.t. $\mathbf{F} = \mathbf{f}$:

$$\mathrm{p}(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{X}, \mathbf{Y}) = \int \mathrm{p}(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{W})\mathrm{p}(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})\mathrm{d}\mathbf{W} \tag{3.7}$$

where the required posterior distribution is now $\mathrm{p}(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$ and its associated DAG is depicted in Figure 3.2.

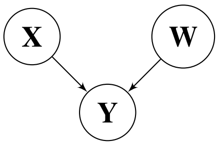


Figure 3.2: DAG ($\mathbf{f}$-parametric)

Despite of this assumption over $\mathbf{F}$, the posterior distribution $\mathrm{p}(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ is still intractable analytically. An alternate way or approximation to evaluate this term is then necessary. One of these Approximate Inference methods among other is known as Variational Inference that connect us with Variational methods.

3.2 Variational Inference

We have briefly reviewed the Bayesian Inference idea from which we can derive Deep Learning and Machine Learning problems. To make it feasible in practice, we must resort to an approximate inference that translates in a efficient way to compute the integral equation (3.7), and concretely, and easy way to draw samples from the posterior distribution $\mathrm{p}(\mathbf{W}|\mathbf{X}, \mathbf{Y})$. Variational Inference is then to assume and replace such distribution by an easy to sample approximate *parametric* one $q_{\boldsymbol{\theta}}(\mathbf{W}) \approx p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$. Note that the approximating distribution $q_\theta(\mathbf{W})$ does not longer depend on data evidence! The problem is then to select a suitable parametric function that fits the real posterior distribution by finding its optimal parameters refereed to as *variational* parameters. At this point it is important to understand that the parameters $\mathbf{W}$ on which $\mathbf{f}$ depends are the parameters of the model (distributions) while $\boldsymbol{\theta}$ are the variational parameters with which we want to approximate $p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$. That is, they are different parameters that we will relate later. Once $q_{\boldsymbol{\theta}}(\mathbf{W})$, from which we can easily sample $\mathbf{w}_t \sim q_{\boldsymbol{\theta}}(\mathbf{W})$ (where t denotes time), has been obtained, the equation (3.6) approximates as

$$p(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{X}, \mathbf{Y}) \approx \int p(\mathbf{y}_{test} \mid \mathbf{x}_{test}, \mathbf{W}) q_{\boldsymbol{\theta}}(\mathbf{W}) d\mathbf{W}. \tag{3.8}$$

We now define a measure between both, the real and the approximate distribution in order state an optimization problem that lead us to find a good approximating posterior. A common election in Approximate Inference is the well-known Kullback-Leibler Divergence (KL) [Kullback and Leibler, 1951], which is a particular case of a greater family called Rènyi's α-divergences [Li and Turner, 2016]. A broader approach has been recently presented in [Hernández-Lobato et al., 2016]. The minimization problems reads as follows:

$$\theta^* = \arg\min_{\theta} \mathrm{KL}(q_{\boldsymbol{\theta}}(\mathbf{W}) || p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})). \tag{3.9}$$

It is well stated that the problem above is equivalent to the maximization of the Evidence Lower Bound (ELBO). Using Jensen's inequality, we argue as follows:

$$\begin{aligned}
\log(p(\mathbf{X}, \mathbf{Y})) &= \log\left(\int_{\mathbf{W}} p(\mathbf{X}, \mathbf{Y}, \mathbf{W}) d\mathbf{W}\right) \\
&= \log\left(\int_{\mathbf{W}} p(\mathbf{X{,}Y{,}W}) \frac{q(\mathbf{W})}{q(\mathbf{W})} d\mathbf{W}\right) \\
&= \log\left(\mathbb{E}_q\left[\frac{p(\mathbf{X}, \mathbf{Y}, \mathbf{W})}{q(\mathbf{W})}\right]\right) \geq \mathbb{E}_q\left[\log\left(\frac{p(\mathbf{X}, \mathbf{Y}, \mathbf{W})}{q(\mathbf{W})}\right)\right]
\end{aligned}$$

in such a way, the lower bound for $-\log(p(\mathbf{X}, \mathbf{Y}))$ is

$$\mathrm{ELBO} = \int_{\mathbf{W}} q(\mathbf{W}) \log\left(\frac{p(\mathbf{X{,}Y{,}W})}{q(\mathbf{W})}\right) d\mathbf{W}. \tag{3.10}$$

On the other hand, by the definition of the KL we have:

$$\begin{aligned}
\mathrm{KL}(q_\theta(\mathbf{W}) || p(\mathbf{W}|\mathbf{Y}, \mathbf{X})) &= \int_{\mathbf{W}} q_\theta(\mathbf{W}) \log\left(\frac{q_\theta(\mathbf{W})}{p(\mathbf{W}|\mathbf{X}, \mathbf{Y})}\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} q_\theta(\mathbf{W}) \log\left(\frac{p(\mathbf{W}|\mathbf{X}, \mathbf{Y})}{q_\theta(\mathbf{W})}\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} q_\theta(\mathbf{W}) \log\left(\frac{p(\mathbf{X}, \mathbf{Y}, \mathbf{W})}{q_\theta(\mathbf{W}) \int_{\mathbf{W}} p(\mathbf{X}, \mathbf{Y}, \mathbf{W}) d\mathbf{W}}\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} q_\theta(\mathbf{W}) \log\left(\frac{p(\mathbf{X}, \mathbf{Y}, \mathbf{W})}{q_\theta(\mathbf{W})}\right) d\mathbf{W} + \int_{\mathbf{W}} q_\theta(\mathbf{W}) \log\left(p(\mathbf{X}, \mathbf{Y})\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} q_\theta(\mathbf{W}) \log\left(\frac{p(\mathbf{X}, \mathbf{Y}, \mathbf{W})}{q_\theta(\mathbf{W})}\right) d\mathbf{W} + \log\left(p(\mathbf{X}, \mathbf{Y})\right)
\end{aligned}$$

where we identify the (3.10) term and therefore obtain:

$$\mathrm{KL}(q_\theta(\mathbf{W}) || p(\mathbf{W}|\mathbf{Y}, \mathbf{X})) = -\mathrm{ELBO} + \log\left(p(\mathbf{X}, \mathbf{Y})\right). \tag{3.11}$$

Dropping the log-evidence term $\log(\mathrm{p}(\mathbf{X},\mathbf{Y}))$ from the minimization problem (3.9) since it does not depend on θ, the optimization problem turns into:

$$\arg\min_{\theta} J(\theta,\mathbf{W})$$

where

$$\begin{aligned}
J(\theta,\mathbf{W}) &= -\int_{\mathbf{W}} \mathrm{q}_\theta(\mathbf{W}) \log\left(\frac{\mathrm{p}(\mathbf{X},\mathbf{Y},\mathbf{W})}{\mathrm{q}_\theta(\mathbf{W})}\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} \mathrm{q}_\theta(\mathbf{W}) \log\left(\frac{\mathrm{p}(\mathbf{Y}|\mathbf{X},\mathbf{W})\mathrm{p}(\mathbf{X}|\mathbf{W})\mathrm{p}(\mathbf{W})}{\mathrm{q}_\theta(\mathbf{W})}\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} \mathrm{q}_\theta(\mathbf{W}) \log\big(\mathrm{p}(\mathbf{Y}|\mathbf{X},\mathbf{W})\big) d\mathbf{W} + \int_{\mathbf{W}} \mathrm{q}_\theta(\mathbf{W}) \log\left(\frac{\mathrm{q}_\theta(\mathbf{W})}{\mathrm{p}(\mathbf{W})}\right) d\mathbf{W} \\
&= -\int_{\mathbf{W}} \mathrm{q}_\theta(\mathbf{W}) \log\big(\mathrm{p}(\mathbf{Y}|\mathbf{X},\mathbf{W})\big) d\mathbf{W} + \mathrm{KL}\big(\mathrm{q}_\theta(\mathbf{W})||\mathrm{p}(\mathbf{W})\big)
\end{aligned}$$

i.e.,

$$\arg\min_{\theta} -\int_{\mathbf{W}} \mathrm{q}_\theta(\mathbf{W}) \log\big(\mathrm{p}(\mathbf{Y}|\mathbf{X},\mathbf{W})\big) d\mathbf{W} + \mathrm{KL}\big(\mathrm{q}_\theta(\mathbf{W})||\mathrm{p}(\mathbf{W})\big). \tag{3.12}$$

First term in equation (3.12) is known as the *expected negative* log-*likelihood* while the second one is refereed to as *prior* KL. The *prior* KL, modeled by selecting prior functions $\mathrm{p}(\mathbf{W})$, is often deemed as a regulariser term in Variational Inference literature. See [Hron et al., 2018] for further details on selecting improper prior functions and its implications.

3.3 Variational and Machine Learning Problems as instances of Bayesian Inference

Here the connection to Deep Learning or Machine Learning is established when the approximating distribution $\mathrm{q}_\theta(\mathbf{W})$ provides only 1 configuration for the model parameters $\mathbf{W}$, so the *variational* parameters are no longer the hyper-parameters of the distribution but those of the model itself. In other words,

$$\mathrm{q}_\theta(\mathbf{W}) = \delta(\mathbf{W}-\theta).$$

A common gaussian prior assumption over the parameters lead us to:

$$\mathrm{KL}(\mathrm{q}_\theta(\mathbf{W})||\mathrm{p}(\mathbf{W})) = \lambda||\theta||^2$$

obtaining similar Tikhonov regularizing term as it has been done in sections 2.1 and 2.2. For further details we refer the reader to [Gal and Ghahramani, 2016a], Appendix A. In this particular case, following the equation (3.12), we have

$$\arg\min_{\theta} -\log\big(\mathrm{p}(\mathbf{Y}|\mathbf{X},\mathbf{W})\big) + \lambda||\theta||^2 \tag{3.13}$$

and recover exactly the same DL problem stated in Section 2.2.

Finally, we can re-think as well the variational denoising presented in Section 2.1, without loss of generality, identifying the *variational* parameters as the parameters of our model which is exactly the image that we aim at recovering, that is,

$$\mathrm{q}_u(\mathbf{U}) = \delta(\mathbf{U}-u).$$

Moreover, the given datum f is a single sample of the whole dataset $\mathbf{F}$, so that the optimization problem derived from the Variational Inference scheme is

$$\begin{aligned} &\arg\min_u -\log\big(\mathrm{p}(\mathbf{F}|u)\big) + \lambda||\phi(u)||^2 \\ &\arg\min_u \sum_{i=1}^{N} ||u - f_i||^2 + \lambda||\phi(u)||^2 \end{aligned} \tag{3.14}$$

where N is the number of samples from the set of noisy images $\mathbf{F}$. Nonetheless, we must recall that, in such problems, we only have access to 1 sample of noisy image ($N = 1$), so that

$$\arg\min_u ||u - f||^2 + \lambda||\phi(u)||^2 \tag{3.15}$$

recovers the original denoising problem and thus, the regulariser term becomes mandatory, otherwise, a simple average would suffice.

3.3.1 Dropout

The arrival of DNN has introduced the technique of dropout [Hinton et al., 2012, Srivastava et al., 2014] as an element to avoid over-fitting, which nowadays is being incorporated in all kinds of NNs. Different interpretations arise from this technique that drops connections between nodes of a Fully Connected Neural Network (FCNN) at random. A common one, is to think that at each training step of a SGD, the considered NN is a sub-model of the principal NN (see Figure 3.3). By construction, all these sub-models have less parameters to train since their connections are set to 0, and thus, the sub-model is less prone to over-fit.

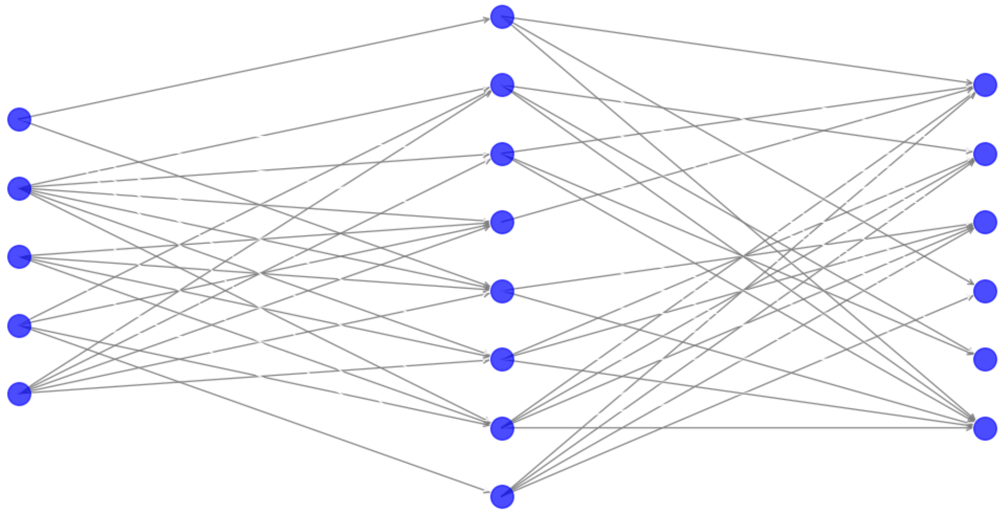


Figure 3.3: Dropout example

Dropout, according to the authors, breaks up complex co-adaptations between neurons that, rather than learning a good firing criteria, may learn to compensate errors made by other neurons. They motivate dropout through the theory of the role of sex in evolution [Livnat et al., 2010], where sexual reproduction, as opposed to asexual reproduction, combines half genes of both parents to produce an offspring (a reason for setting the keep-probability of a connection to $p = 0.5$). Analogously, one can think that, in NN, removing such complex co-adaptations is intuitively bad, precisely due to the same reason: it removes complex structures of neurons that have learned to "work" together. On the other hand, asexual reproduction is prone to replicate those complex neuron's relationships and thus, translating to NN, the neurons can over-fit the train set of examples that leads to badly generalize. Those contradictions are solved showing that in evolution, sexual reproduction is the way most of the advanced organism have evolved. Authors also add other interpretation based on conspiracies.

Interestingly, straying from those sexual reproduction and conspiracy interpretations, dropout has recently been revisited as a bayesian approximation for representing and estimating the model uncertainty [Gal and Ghahramani, 2016b]; whereas SGD has been investigated as an approximation of bayesian inference in [Mandt et al., 2017]. In fact, seeing a NN as a non-static structure allows us to find a principled explanation of dropout. This is, each time we measure its parameters or weights we obtain different values that are drawn from a distribution. Dropout can be seen as using a discrete distribution with two modes where one of them is located in 0. In other words, a $Bernoulli(p)$ distribution is placed over each weight. Of course, again, this constitute an a priori that models the weights in a way that it directly affects the structure of the NN. Recall that, as aforementioned in 2.2.4, the success of CNN in CV problems is due to the structure of the convolution operation and the infinitely strong prior it impose.

CHAPTER **4**

P1. Proposed Variational Models for Saliency detection

Ça ne rapproche pas, le téléphone, ça confirme les distances.
Simone de Beauvoir

This chapter is organized as follows. We start introducing the Saliency concept and the adressed problem. In section 4.1, we introduce the variational mathematical framework of our models. Starting with the local equations as guide for the modelling exercise, we focus on the non-local diffusive terms, explicited in the form of p-Laplacian operators, for $p > 1$ and extend it to the range $0 < p \leq 1$ through a differentiable family of fluxes to cover the resulting non-local non-convex hyper-Laplacian operators. Then, we introduce a multi-valued concave saliency detection term which defines an obstacle problem for the non-local diffusion models. In Section 4.2 we describe and solve two versions of the proposed model: local and non local. We deduce the corresponding Euler-Lagrange equations. A gradient descent approximation is used to solve the elliptic non-local problems until stabilization of the associated evolution problems, and a Primal-Dual based algorithm for the associated local. Moreover, the local version is embedded in a DL architecture. Such setting allows us to take advance of the goods of both CNN and the proposed variational model while avoiding their major drawbacks: uncertainty and manual hyper-parameterization, respectively. Section 4.3 contains the numerical experiments on the proposed models and present the simulations performed on FLAIR sequences of MR images obtained from the BRATS2015 dataset [Menze et al., 2015] which consists of 220 subjects for High Grade Glioblastomas (HGG) detection.

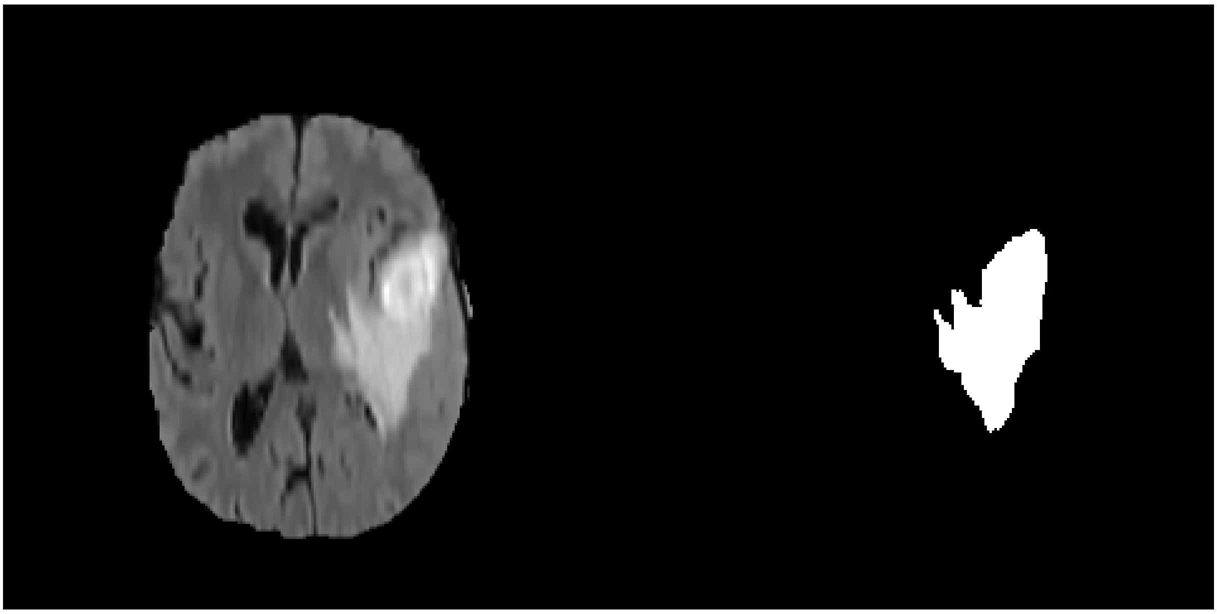
4.1 Introduction: Saliency Detection

Currently there is a growing interest in Image Processing and computer vision applications for visual saliency driven models, able to focus on perceptually relevant information within digital images. Despite of the lack of a general consensus on a proper mathematical definition of saliency, it has a clear biologically perceptive meaning: it models the mechanism of human attention, that is, of finding relevant objects within an image. Saliency-based models are then grouped into different families depending on if they predict either human gaze [Cerf et al., 2008], [Wang and Shen, 2018] or salient objects [Riche and Mancas, 2016], [Wang et al., 2015a], [Zhu et al., 2018]. In the last case, termed as computational saliency, the final step of the saliency detection algorithm is a segmentation of the salient object. Recently, there has been a burst of research on saliency due to its wide application. Semantic segmentation [Bergbauer et al., 2013], object detection [Li et al., 2018], image clustering [TANG Li-Ming, 2014], retrieval and cognitive saliency applications [Wu et al., 2018], are just few examples of saliency driven based models which in turn favour image captioning [Bernardi et al., 2016] and high-level image understanding [Singh et al., 2017]. Saliency is also of interest for improving computational efficiency and for increasing robustness in clustering and thresholding in the sense it allows to discard regions that are unlike to be relevant. For instance, if a Saliency method select the regions in an image that must be processed by a different and expensive method avoiding irrelevant parts of it, the computational burden is greatly reduced.

At this point, one can ask what is the difference between Saliency detection and Segmentation. The question is pertinent and in fact collides again with the concept of *information* discussed in the Introduction Chapter 1. Recall that, due to the lack of formal definition of what is understood by information, we shall precise the objective of our method which, as aforementioned, can be cast as: (1) recovering a given image (Restoration) or (2) extracting features from it. This classification can be found in almost every book of CV, which displays how terminology sometimes makes difficult rather than easier the ultimate understanding of the problem. Nevertheless, the reason of the use of Saliency is is two-fold. First, the deemed application in the following sections was inspired in similar variational models casted as saliency detection models. Second, those methods do not provides a direct segmentation since the outcome is a continuous images ranging in $[0, 1]$ and thus can be interpreted as a heat or probability point-wise map.

In this chapter we shall focus on variational saliency detection and accurate object segmentation [Donoser et al., 2009], [Li et al., 2013], [Li et al., 2017] as a first step for successful image understanding. The role of saliency in models is application dependent, and thus several different techniques and approaches have been introduced to construct saliency maps. They vary from low dimensional manifold features minimization [Zhan, 2011] to non-local sparse minimization [Wang et al., 2014], graphs techniques [Harel et al., 2007], partial differential equations (PDE) [Li et al., 2013], superpixels [Liu et al., 2013], learning methods [Liu et al., 2014], or neural networks based approaches [Bylinskii et al.,]. State of the art methods are currently based on DL techniques using NNs [Zhang et al., 2018] which provide high level features and semantic information which are not considered in a pure variational approach. Therefore we aim at exploring the applications and algorithms of non-smooth, non-local, non-convex optimization of saliency models and also the possibility of combining both variational and state of the art DL methods, [Pereira et al., 2016, Havaei et al., 2017, Zhao et al., 2018].

Leading medical disciplines such as neuroscience [Yang et al., 2017] and cardiology have also incorporated this concept. Considering medical images modalities such as Magnetic Resonance Imaging (MRI) or Positron Emission Tomography (PET), the automatic obtention of saliency maps is used for pathology detection, disease classification [Rueda et al., 2013], location and segmentation of brain strokes, gliomas, myocardium detection for PET images, tumors quantification in FLAIR MRI [Thota et al., 2016] and so forth. Inspired by the variety of models ranging from non-local properties to hyper-laplacian priors, we present two variational models for saliency segmentation, their numerical resolutions and the results obtained by its application to MR images for accurate location of tumor and edema. The underlying hypothesis, fulfilled by such images, is that the salient region is brighter than the rest. A example of FLAIR input image and the expected output is depicted below in Figure 4.1.



(a) FLAIR input image

(b) Segmentation of the expected Saliency output

Figure 4.1: Saliency example

Indeed, brain tumor segmentation is one of the most important and difficult tasks in medical imaging. A proper segmentation provides quantitative and qualitative information of the cancer that helps clinicians to find the most effective treatments for each patient or in case of need to better plan a chirurgical intervention. Nevertheless, this work is usually done manually by experts resulting in a slow, difficult and tedious task that is subject to errors and differences between expert's criteria. In order to overcome these problems many methods have been proposed to automatically perform this task specially when using MRI of the brain [Gordillo et al., 2013].

The variational models for saliency detection we propose are based on a TV restoration functional (local and analogous non-local version) plus a concave saliency term which provokes a sort of binarization of the solution. The resulting functional is non-smooth because of the non-differentiability of the TV and non-convex because of the new saliency term. While the non-local version is further solved with a gradient descent based algorithm, the local version takes advantaged of the fact that the global energy functional has a special structure called Difference of Convex (DC) functionals allowing the use of a proximal point algorithm to find a critical point of the minimization problem [Sun et al., 2003]. Furthermore, the Chambolle and Pock primal-dual algorithm [Chambolle and Pock, 2011] is used to deal with the TV's non-smoothness in a resulting subproblem.

Finally, this local but fast resolution of the model allows us to go a step further by embedding the proposed model into a DL framework, following recent ideas as [Kobler et al., 2017]. The main motivation for such a purpose can be argued as follows: a fine tuning of the parameters balancing the variational model is unavoidable in order to optimize the results and it is one of the most difficult tasks for real applications. Using a NN and unwrapping the numerical resolution of our model as extra layers allows us to train and find the optimal parameters of our saliency detection model using knowledge from experts in brain tumor segmentation. Moreover, the neural network can learn spatially adaptive parameter maps based on the input image optimizing the performance of our model, which is completely out of range for a manual tuning. On the other hand, despite the fact DL techniques hold the state of the art results in this field, reliability is sometimes put in doubt due to the variability of the outcomes. This effect is often known as *black box effects*. Recently, some works have been carried out in order to show how neural networks are sometimes easy to fool. In [Athalye et al., 2017], authors create adversarial examples introducing tiny perturbations on the input image. In [Su et al., 2019], they achieve to fool a deep learning trained neural network by just re-placing 1 pixel. DL is indeed, an interpolation problem (see [Mallat, 2016]) where the given points are images, audios, i.e, any high-dimensional data. This translates into the need of tones of examples that cover and capture the entire distribution of the data.

Using an embedded variational model in a deep learning scheme prevent the whole system from these detrimental effects since the final outcome is provided by the variational model over which we keep a full control.

Models for saliency detection try to transform a given image, f, defined in the pixel domain, Ω, into a constant-wise image, u, whose level sets correspond to salient regions of the original image. They are usually formulated through the inter-relation among three energies: fidelity, regularization, and saliency, being the latter the mechanism promoting the classification of pixels into two or more classes. There is a general agreement in considering the fidelity term as determined by the L^2 norm, that can also be explained from Bayesian modeling placing a gaussian distribution over the likelihood term (see 2.1.3), that is

$$F(u) = \frac{1}{2}\int_\Omega |u-f|^2,$$

so that departure from the original state is penalized in the minimization procedure. For regularization, an edge preserving energy should be preferred. The use of the TV energy as described in 2.1.2, or commonly expressed with abuse of notation:

$$TV(u) = \int_\Omega |\nabla u|,$$

is defined on the space of Bounded Variation. Recall that the TV energy allows discontinuous functions to be solutions of the corresponding minimization problem, with discontinuities representing edges, in contrast to Sobolev norms, which enforce continuity across level lines and thus introduce image blurring, as seen in chapter 2.

Only recently, the non-local version of the TV energy and, in general, of the energy associated to the p-Laplacian, for $p > 1$, has been considered in restoration modeling. In saliency modeling, just the range $p \geq 2$ seems to have been treated [Li et al., 2013]. One of the main advantages of introducing these non-local energies is the lack of the hard regularizing effect influencing their local counterparts, [Pérez-LLanos and Rossi, 2011]. Such effect erases the boundary conditions needed in the local versions, such as (2.16), (2.17) and (2.18). A general discrete framework for non-local p-Laplacian regularization on graphs covering the case $p > 0$ can be found in [Elmoataz et al., 2008]. For $p = 0$ a saliency discrete model based on superpixels is given in [Wang et al., 2015b].

For the remaining saliency term, a phase-transition model can be considered. Developed by Ginzburg-Landau [Ginzburg, 1955], CahnHilliard [Cahn and Hilliard, 1958] and Van der Walls [Van der Waals, 1979] in the field of mechanics and materials, this model consist of a double-well absorption-reaction term.

Those phase-transition models are often applied in Image Processing for segmentation or binarization problems (see [Kornprobst, 2006, Aubert et al., 2005] for further details). The resulting energy is a functional of the type

$$W(u) = \int_\Omega w(u) = \int_\Omega \left(1 - |u|^2\right)^2,$$

whose minimization drives the solution towards the discrete set of values $\{-1, 1\}$, facilitating in this way the labeling process. However, due to the vanishing slope of $g(u) = (1 - |u|^2)^2$ at $u = \pm 1$, the resulting algorithm has a slow convergence to the minimizer (Figure 4.2a). Notice that setting the minimizer in $\{-1, 1\}$ requires to re-scale the input image to such range. Similar non-convex variants are proposed in [Li et al., 2013] and [Li et al., 2017].

Local p-Laplacian To introduce the modelling assumptions we briefly consider the minimization of the energy functional

$$E_p(u) = \lambda F(u) + P_p(u), \tag{4.1}$$

where $\lambda > 0$ is a constant, $P_p(u)$ and $F(u)$ are the *regularization* and the *fidelity* terms, respectively, given by

$$P_p(u) = \frac{1}{p}\int_\Omega \phi_p(Du)d\mathbf{x}, \quad F(u) = \frac{1}{2}\int_\Omega |u - f|^2 d\mathbf{x},$$

where $\Omega \subset \mathbb{R}^2$ is a bounded domain (the set of pixels in the discrete case), $f : \Omega \to [0, 1]$ is the image to be processed, $u : \Omega \to \mathbb{R}$ belongs to a space of functions for which the minimization problem admits a solution and $\phi_p(\cdot)$ is the potential function covering the range of different values of p (Figure 2.4). The idea behind this minimization problem is, as a denoising problem: given a non-smooth (e.g. noisy) image, f, to obtain another image which is close to the original (fidelity term) but regular (bounded gradient in $L^p(\Omega)$). The parameter λ is a ratio between the associated standard deviations ($\lambda = (\sigma_1/\sigma_2)^2$, see 2.1), indeed a weight balancing the respective importance of the two terms in the functional. When first order necessary optimality conditions are imposed on the energy functional, the PDE problem presented in (2.18) arises. The Eulear-Lagrange is then:

$$-\left(|\nabla u|^{p-2}\nabla u\right) + \lambda(u - f) = 0. \tag{4.2}$$

Properties of the so-called *p-Laplacian* term have been extensively studied in the last decades for the range of exponents $p \geq 1$. For $p > 1$, the energy $P_p(u)$ is convex and differentiable, and the solution to the minimization problem belongs to the Sobolev space $W^{1,p}(\Omega)$, implying that u can not have discontinuities across level lines. Therefore, the solution, u, is smooth even if the original image, f, has steep discontinuities (edges). This effect is known as *blurring* as already explained in chapter 2 which translates into diffused edges of the resulting image.

In the case $p = 1$, the energy term $P_1(u)$ is convex but not differentiable. Thus, in this case, the edges of f are preserved in the solution, u, because a function of bounded variation may have discontinuities across surface levels. We are specially interested in the range $0 < p < 1$, for which the energy $P_p(u)$ is neither convex nor differentiable, and it only generates a quasi-norm on the corresponding $L^p(\Omega)$ space. In this parameter range the problem lacks of a sound mathematical theory, although some progress is being carried on [Hintermüller and Wu, 2014]. Despite the difficulties for the mathematical analysis, there is numerical evidence on interesting properties arising from this model, among others, by adapting the prior to the real estimated negative log-likelihood direct or indirectly based on data, huge improvements can be achieved. In particular, the non-convexity forces the gradient to be *sparse* in so far it minimizes the number of jumps in the image domain, which is consistent with the idea of seeing the gradient of an image as the precursor of its edges. Actually, if only sharp jumps are preserved, the resulting image tends to look like a cartoon piecewise constant image. For this reason a hyper-laplacian operator is suitable to be used in a Saliency variational model.

Nonlocal p-Laplacian While the use of the local p-Laplacian energy is not specially relevant in Image Processing for $p > 1$ due to its regularizing effect on solutions which produces over-smoothing of the spatial structures, for its non-local version the initial data and the final solution belong to the same functional space, i.e., no global regularization takes place. See [Andreu-Vaillo et al., 2010], where a thorough study on non-local diffusion evolution problems, including existence and uniqueness theory, may be found. Non-local operators were introduced in Image Processing in [Kindermann et al., 2005] and developed in [Gilboa and Osher, 2008]. An easy interpretation of these Non-local hyper-laplacian operators for the sake of understanding and motivation of their use in Saliency detection models is the following: as the local analogous promotes sparse gradients and thus sharp edges, the non local hyper-laplacian operator promotes sparse non-local gradients which can be seen as a measure of the number of range levels. Minimizing such a term in thus equivalent to reduce the number of different classes within the image. This property fits well with the aim at obtaining a nearly binary outcome.

The non-local analogous of the energy $P_p(u)$, for $p > 1$, is

$$P_p^{nl}(u) = \frac{1}{2p}\int_{\Omega\times\Omega} w(\mathbf{x}-\mathbf{y})|u(\mathbf{y})-u(\mathbf{x})|^p d\mathbf{y}d\mathbf{x}, \tag{4.3}$$

where w is a continuous non-negative radial function with $w(0) > 0$ and $\int_{\mathbb{R}^2} w = 1$. The Fréchet differential of $P_p^{nl}(u)$ is

$$DP_p^{nl}(u) = \int_\Omega w(\mathbf{x}-\mathbf{y})|u(\mathbf{y})-u(\mathbf{x})|^{p-2}(u(\mathbf{y})-u(\mathbf{x}))d\mathbf{y}.$$

Thus, the Euler-Lagrange equation for the minimization problem (4.1) when $P_p(u)$ is replaced by $P_p^{nl}(u)$ is

$$\int_\Omega w(\mathbf{x}-\mathbf{y})|u(\mathbf{y})-u(\ \mathbf{x})|^{p-2}(u(\mathbf{y})-u(\mathbf{x}))d\mathbf{y}+\lambda(f-u)=0. \tag{4.4}$$

For $p \leq 1$, the Euler-Lagrange equation (4.4) does not have a precise meaning due to the singularities that may arise when the denominator vanishes. To overcome this situation, we approximate the non-differentiable energy functional $P_p^{nl}(u)$ by

$$P_{\epsilon,p}^{nl}(u) = \frac{1}{4}\int_{\Omega\times\Omega} w(\mathbf{x}-\mathbf{y})\phi_{\epsilon,p}(u(\mathbf{y})-u(\mathbf{x}))d\mathbf{x}d\mathbf{y},$$

for $\epsilon > 0$, where

$$\phi_{\epsilon,p}(s) = \frac{2}{p}\left(s^2+\epsilon^2\right)^{p/2} - \frac{2}{p}\epsilon^p$$

is regularized edge preserving functions family (presented in chapter 2) when $0 < p \leq 1$.

Observe that the corresponding minimization problem is now well-posed due to the differentiability of $P_{\epsilon,p}^{nl}(u)$. Therefore, a solution may be calculated solving the associated Euler-Lagrange equations. However, for $p < 1$, the solution is in general just a local minimum, due to the lack of convexity. Of course, the same reasoning may be followed for the local diffusion equation (4.2).

4.1.1 Saliency modeling

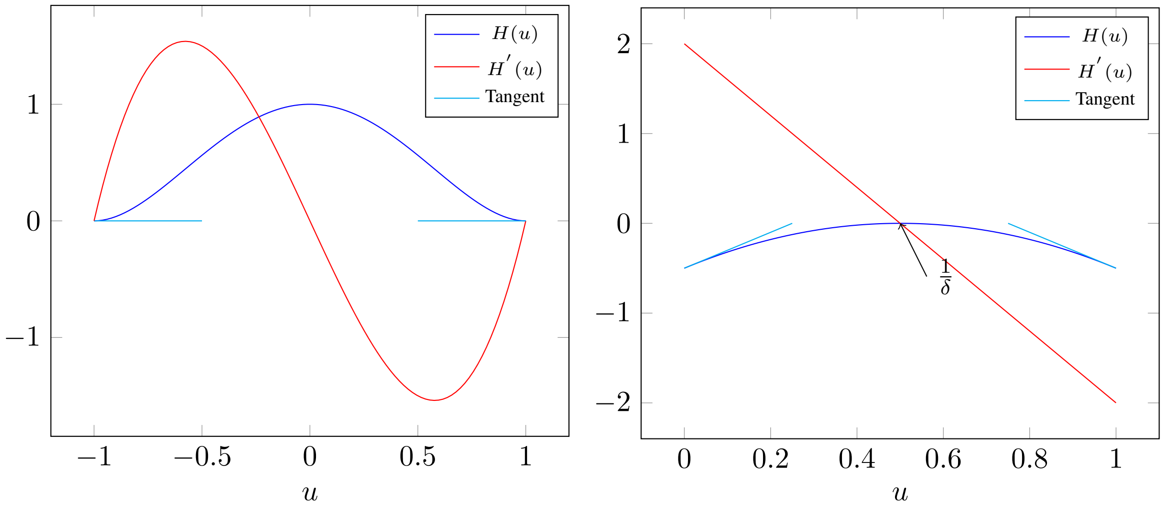
Finally we introduce a new saliency term that enhances the convergence of the classification algorithm (salient vs background) while keeping a good quality compromise. The general idea is pushing the values of u towards the discrete set of extremal image values $\{0, 1\}$, determining the labels we impose for saliency detection: $u = 1$ for foreground, and $u = 0$ for background. To model this behavior we propose a two-terms based energy, where the first causes a reaction extremizing the values of the solution and the second accounts for the problem constraints ($0 \leq u \leq 1$). The former is captured by the concave energy (depicted in Figure 4.2b):

$$H(u) = \int_\Omega h(u) = -\frac{1}{2}\int_\Omega (1 - \delta u)^2, \tag{4.5}$$

with $\delta > 0$ constant, which fastly drives the minimization procedure so that $H(u) \to -\infty$. This is a *concave* quadratic energy term that differs with the more common use of *convex* 4-th order polynomial terms to model *double-well* potentials for image classification. Our term (4.5) is always negative except when $u = 1/\delta$, therefore its minimization pushes the solution away from this value. The latter, to counteract this tendency and remain in the meaningful interval $u \in I = [0, 1]$, is introduced as an obstacle which penalizes the minimization when the solution lies outside I and a (global) minimum is obtained in agreement with basic calculus. The modeling of such obstacle is given in terms of the indicator function

$$\mathcal{I}_I(u) = \begin{cases} 0 & \text{if } u \in I, \\ \infty & \text{if } u \notin I, \end{cases}$$

and the resulting saliency term is then defined as a weighted sum of the operators $H(u)$ and $\mathcal{I}_I(u)$. Observe that since I is convex and closed, the functional $\mathcal{I}_I(u)$ is convex and lower semi-continuous, and



(a) Double wheel Potential Function

(b) Proposed Concave Saliency Term

Figure 4.2: Saliency terms

that its sub-differential is the maximal monotone graph of $\mathbb{R} \times \mathbb{R}$, given by

$$\partial \mathcal{I}_I(u) = \begin{cases} (-\infty, 0] & \text{if } u = 0, \\ 0 & \text{if } 0 < u < 1, \\ [0, +\infty) & \text{if } u = 1, \\ \emptyset & \text{otherwise.} \end{cases}$$

The saliency term we propose is the sum of the fast saliency promotion, $H(u)$, and of the range limiting mechanism, $\mathcal{I}_I(u)$, i.e.

$$S(u) = H(u) + \mathcal{I}_I(u).$$

4.2 Proposed Methods: NLTVS and TVS+CNN

In the following we propose two different models and their resolutions for Saliency detection. The first, a non local model solved through a gradient descent algorithm with different simplifications that boost its numerical resolution. The second consists of a local version however solved through a Primal-Dual algorithm and avoiding any type of operator's regularization. Moreover, this numerical resolution is combined, indeed embedded, into a CNN to create a Deep Variational Framework.

4.2.1 Non Local Total Variation Saliency model(NLTVS)

Gathering the fidelity, the regularizing and the saliency energies, we define a bilateral constrained obstacle problem associated to the following energy

$$E^{nl}_{\epsilon,p}(u) = \alpha P^{nl}_{\epsilon,p}(u) + \lambda F(u) + \frac{1}{\alpha} S(u), \tag{4.6}$$

where $\alpha > 0$ is a parameter modulating the relationship between regularization and saliency promotion. Observe that there is no use in multiplying $\mathcal{I}_I(u)$ by the constant $1/\alpha$, so we omit it for clarity. The Euler-Lagrange equation corresponding to (4.6) together with the use of a gradient descent method leads to the consideration of non-local multi-valued evolution problems. In particular we have:

$$\arg\min_u \alpha P^{nl}_{\epsilon,p}(u) + \lambda F(u) + \frac{1}{\alpha} S(u). \tag{4.7}$$

Multi-valued Problems $P_\epsilon(u)$

Set $Q_T = (0, T) \times \Omega$ and let α, δ, λ and ϵ be real fixed positive parameters. Let moreover $f \in L^\infty(\Omega)$. For some given $T > 0$, find $u : [0, T] \times \Omega \to \mathbb{R}$ solving the approximating smooth (in fact differentiable) multivalued problem

$$P_\epsilon(u) \begin{cases} \partial_t u - \alpha K_{\varepsilon,p}(u) + \partial \mathcal{I}_I(u) \ni au - b \quad \text{in } Q_T, \\ u(0, \cdot) = f \quad \text{on } \Omega, \end{cases} \tag{4.8}$$

which model non-linear non-local non-convex reactive flows that we shall consider in the range $0 < p \leq 1$.

For the sake of presentation, we have introduced the following notation in (4.8): we rewrote the Fréchet differential of $H(u)/\alpha + \lambda F(u)$ as $au(x) - b(x)$, with

$$a = \frac{\delta^2}{\alpha} - \lambda, \quad b(x) = \frac{\delta}{\alpha} - \lambda f(x), \quad \text{for } x \in \Omega, \tag{4.9}$$

and defined the non-local hyper-Laplacian ($0 < p < 1$) and 1-Laplacian ($p = 1$) diffusion operators

$$K_{\epsilon,p}(u)(t,x) = \int_\Omega w(x-y) k_{\epsilon,p}(u(t,y) - u(t,x)) dy,$$

with differentiable kernels

$$k_{\epsilon,p}(s) = \frac{1}{2}\phi'_{\epsilon,p}(s) = s\left(s^2 + \epsilon^2\right)^{\frac{p-2}{2}}. \tag{4.10}$$

Notice that, while for the local diffusion problem we must explicitly impose the homogeneous Neumann boundary conditions, which are the most common boundary conditions for Image Processing tasks, for the non-local diffusion problem this is no longer necessary since these conditions are implicitly imposed by the non-local diffusion operator [Andreu-Vaillo et al., 2010].

Yosida's approximants

We now show that the solutions of the multivalued problem (4.8) may be approximated by the introduction of Yosida's approximants, leading to the single-valued problems $P_{\epsilon,r}(u)$ in (4.12) that depend on the Yosida's approximation parameter r. We also prove that in the limit $r \to 0$ the corresponding solutions lie in the relevant range of values for Image Processing tasks, this is, in the interval $[0,1]$. Introducing the maximal monotone graphs $\beta, \gamma \subset \mathbb{R} \times \mathbb{R}$ given by

$$\beta(u) = \begin{cases} \emptyset & \text{if } u < 0, \\ (-\infty, 0] & \text{if } u = 0, \\ 0 & \text{if } u > 0, \end{cases} \qquad \gamma(u) = \begin{cases} 0 & \text{if } u < 0, \\ [0, \infty) & \text{if } u = 0, \\ \emptyset & \text{if } u > 0, \end{cases}$$

we may express the subdifferential of $\mathcal{I}_I(u)$ as $\partial\mathcal{I}_I(u) = \beta(u) + \gamma(u-1)$. The Yosida's approximants of β and γ are then

$$\beta_r(u) = \begin{cases} u/r & \text{if } u \leq 0, \\ 0 & \text{if } u > 0, \end{cases} \qquad \gamma_r(u) = \begin{cases} 0 & \text{if } u < 0, \\ u/r & \text{if } u \geq 0, \end{cases}$$

for $r > 0$, allowing us to approximate the multi-valued formulation (4.8) by single-valued equations in which β and γ are replaced by β_r and γ_r. This is, by the evolution integro-differential equation

$$\partial_t u - \alpha K_{\varepsilon,p}(u) + \beta_r(u) + \gamma_r(u-1) = au - b. \tag{4.11}$$

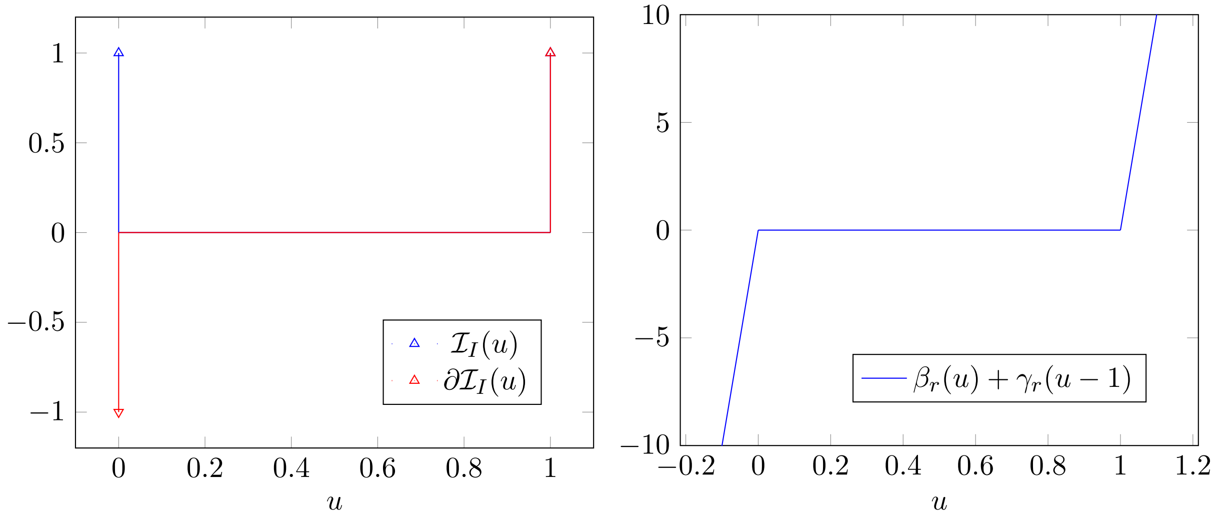
Assume that a solution, u, of (4.11) with initial data $u(0,\cdot) = f$ does exist, and consider the characteristic function of a set C, defined as $\chi_C(x) = 1$ if $x \in C$ and $\chi_C(x) = 0$ otherwise. Introducing the sets

$$\Omega_0(t) = \{x \in \Omega \,|\, u(t,x) > 0\}, \qquad \Omega_1(t) = \{x \in \Omega \,|\, u(t,x) < 1\},$$

for $t \in [0,T)$, we may express the Yosida's approximants appearing in (4.11) in terms of characteristics functions in form

$$\beta_r(u(t,x)) = \frac{1}{r}u(t,x)\chi_0(t,x), \quad \gamma_r(u(t,x)) = \frac{1}{r}(u(t,x) - 1)\chi_1(t,x),$$

with, for $i = 0, 1$, $\chi_i(t,x) = \chi_{\Omega\setminus\Omega_i(t)}(x)$.



(a) Indicator function $\mathcal{I}_I(u)$ and subdifferential

(b) Approximating $\partial\mathcal{I}_I, r(u)$ through Yosida's approximants β_r, γ_r

Approximating Single-Valued Problems $P_{\epsilon,r}(u)$

We rewrite (4.11) as a family of approximating problems $P_{\epsilon,r}(u)$ using the characteristic functions introduced before. Set $Q_T = (0,T)\times\Omega$. Given $a > 0$, $f \in L^\infty(\Omega)$, $f(x) \geq 0$ a.e. in Ω, define $b(x) \geq 0$ using (4.9) and solve

$$P_{\epsilon,r}(u) \begin{cases} \partial_t u - \alpha K_{\varepsilon,p}(u) + \frac{1}{r}\big(u\chi_0 + (u-1)\chi_1\big) = au - b & \text{in } Q_T, \\ u(0,\cdot) = f \quad \text{on } \Omega, \end{cases} \tag{4.12}$$

Notice that $u\chi_0 = -u^-$ and $(u-1)\chi_1 = (u-1)^+$, where we used the notation $u^+ = \max(u,0)$, $u^- = -\min(u,0)$, so that $u = u^+ - u^-$. The following result generalizes to the non-local framework the results of [Murea and Tiba, 2013], establishing that the solution of (4.12) is such that the subset of $(0,T)\times\Omega$ where $u(t,x) \notin [0,1]$ may be done arbitrarily small by decreasing r. Thus, in the limit $r \to 0$ the solution does not overpass the obstacles $u = 0$ and $u = 1$ and fulfills the bilateral constraints.

Theorem 2. *Let $b \in L^2(\Omega)$ and assume that the parameters $\alpha, \epsilon, p, r, a$ are positive. If $u \in H^1(0,T;L^2(\Omega))$ is the corresponding solution of (4.12), then*

$$\int_0^T \int_\Omega \big(|u^-|^2 + |(u-1)^+|^2\big) \leq C(T)r,$$

for some constant $C(T)$ independent of r.

Proof. Multiplying (4.12) by $-u^-$ and integrating in Ω, we obtain

$$\frac{d}{dt}\int_\Omega |u^-|^2 + \alpha\int_\Omega K_{\varepsilon,p}(u)u^- + \frac{1}{r}\int_\Omega |u^-|^2 = a\int_\Omega |u^-|^2 + \int_\Omega bu^-, \tag{4.13}$$

where we used $\chi_1 u^- = 0$. Since $k_{\epsilon,p}$ is an odd function, the following *integration by parts* formula holds

$$\int_\Omega K_{\varepsilon,p}(u)(t,x)u^-(t,x)dx =$$

$$= \int_\Omega \Big(\int_\Omega w(x-y) k_{\epsilon,p}(u(t,y)-u(t,x))dy \Big) u^-(t,x)dx =$$

$$= -\frac{1}{2} \int_\Omega \int_\Omega w(x-y) k_{\epsilon,p}(u(t,y)-u(t,x))(u^-(t,y)-u^-(t,x))dydx.$$

Thus, noting that u^- is non-increasing as a function of u, we deduce

$$(u(t,y)-u(t,x))(u^-(t,y)-u^-(t,x)) \leq 0,$$

and therefore, see (4.10),

$$\int_\Omega K_{\varepsilon,p}(u)(t,x)u^-(t,x)dx \geq 0. \tag{4.14}$$

Using (4.14) and the Schwarz's inequality in (4.13) we get,

$$\frac{d}{dt}\int_\Omega |u^-|^2 + \frac{1}{r}\int_\Omega |u^-|^2 \leq (a+\frac{1}{2})\int_\Omega |u^-|^2 + \frac{1}{2}\int_\Omega b^2. \tag{4.15}$$

Getting rid of the term $r^{-1}\int_\Omega |u^-|^2 \geq 0$, we apply Gronwall's inequality to the resulting inequality to obtain

$$\int_\Omega |u^-(t,\cdot)|^2 \leq C_1(t)\int_\Omega b^2,$$

with $C_1(t) = t\exp((a+1/2)t)$. Using this estimate in (4.15) yields

$$\frac{d}{dt}\int_\Omega |u^-|^2 + \frac{1}{r}\int_\Omega |u^-|^2 \leq C_2(t)\int_\Omega b^2,$$

with $C_2(t) = (a+1/2)C_1(t) + 1/2$. Finally, integrating in $(0,T)$ and using that $u(0,\cdot) = f \geq 0$, we obtain

$$\int_0^T \int_\Omega |u^-|^2 \leq C(T)r\int_\Omega b^2, \tag{4.16}$$

for some constant $C(T)$ independent of r. To finish the proof we must show that also

$$\int_0^T \int_\Omega |(u-1)^+|^2 \leq C(T)r.$$

Since, once we multiply (4.12) by $(u-1)^+$ and integrate in Ω, the arguments are similar to those employed to get estimate (4.16), we omit the proof.

Discretization of the problem $P_{\epsilon,r}(u)$

In this Section we provide a fully discrete algorithm to numerically approximate the limit solution of (4.12) when $r \to 0$. First, we introduce a time semi-implicit Euler discretization of the evolution equation in (4.12), that we show to retain the stability property of its continuous counterpart, stated in Theorem 2.

The resulting space dependent non-local PDE is discretized by finite differences. Since the problem is nonlinear and, in addition, we want to pass to the limit $r \to 0$, we introduce an iterative algorithm which renders the problem to a linear form and, at the same time, replaces the fixed parameter r by a decreasing sequence $r_j \to 0$.

Time discretization For the time discretization, let $N \in \mathbb{N}$, $\tau = T/N$, and consider the decomposition $(0,T] = \cup_{n=0}^{N-1}(t_n, t_{n+1}]$, with $t_n = n\tau$. We denote by $u^n(x)$ to $u(t_n, x)$ and by $\chi_i^n(x)$ to $\chi_i(t_n, x)$, for $i = 0, 1$. Then, we consider a time discretization of (4.12) in which all the terms are implicit but the diffusion term, which is semi-implicit. The resulting time discretization iterative scheme is:

Iterative Problems $P_{\epsilon,r}(u^n)$ Given positive parameters ϵ, p, a time discretization step τ, a constant $a > 0$ and a function $b \in L^\infty(\Omega)$, $b(x) \geq 0$ a.e. $x \in \Omega$ set $u^0 = f$ and for $n = 0, \ldots, N-1$ find $u^{n+1} : \Omega \to \mathbb{R}$ such that

$$P_{\epsilon,r}(u^n) \begin{cases} u^n(x) - \tau b(x) = (1-\tau a)u^{n+1}(x) - \tau\alpha\tilde{K}_{\epsilon,p}(u^n, u^{n+1})(x) \\ +\dfrac{\tau}{r}\big(u^{n+1}(x)\chi_0^{n+1}(x) + (u^{n+1}(x)-1)\chi_1^{n+1}(x)\big), \end{cases} \tag{4.17}$$

where, for $\tilde{k}_{\epsilon,p}(s,\sigma) = \sigma\left(s^2+\epsilon^2\right)^{(p-2)/2}$, we define

$$\tilde{K}_{\epsilon,p}(u^n, u^{n+1})(x) = \int_\Omega w(x-y)\tilde{k}_{\epsilon,p}(u^n(y)-u^n(x), u^{n+1}(y)-u^{n+1}(x))dy. \tag{4.18}$$

That is, only the modulus part of the diffusion term is evaluated in the previous time step. Equation (4.17) is still nonlinear (in fact piece-wise linear) due to the Yosida's approximants of the penalty term. In addition, its solution depends on the fixed parameter r that, in view of Theorem 2, we wish to make arbitrarily small, so that the corresponding solution values are effectively constrained to the set $[0,1]$. To do this, we consider the following iterative algorithm to approximate the r-dependent solution, u^{n+1}, of (4.17) when $r \to 0$.

Remark 1. The stability result for the time continuous problem (4.12) stated in Theorem 2 may be adapted with minor changes to the semi-implicit time discrete problem $P_{\epsilon,r}(u^n)$ in (4.17).

Iterative Approximating Problems $P_j(u^n)$ Let ϵ, p, τ, a, b and f be as assumed in problem (4.17). Let $u^0 = f$ and $r_0 > 0$ be given. For $n = 0, \ldots, N-1$, set $u_0^{n+1} = u^n$. Then, for $j = 0, 1 \ldots$, define $r_j = 2^{-j}r_0$ and, until convergence, solve the following problem: find $u_{j+1}^{n+1} : \Omega \to \mathbb{R}$ such that

$$P_j(u^n) \begin{cases} u^n(x) - \tau b(x) = (1-\tau a)u_{j+1}^{n+1}(x) - \tau\alpha\tilde{K}_{\epsilon,p}(u^n, u_{j+1}^{n+1})(x) \\ +\dfrac{\tau}{r_j}\big(u_{j+1}^{n+1}(x)\chi_{0,j}^{n+1}(x) + (u_{j+1}^{n+1}(x)-1)\chi_{1,j}^{n+1}(x)\big), \end{cases} \tag{4.19}$$

where $\chi_{i,j}^{n+1}(x) = \chi_{\Omega\setminus\Omega_{i,j}^{n+1}}(x)$ for $i = 0, 1$, being

$$\Omega_{0,j}^{n+1} = \{x \in \Omega \,|\, u_j^{n+1}(x) > 0\}, \qquad \Omega_{1,j}^{n+1} = \{x \in \Omega \,|\, u_j^{n+1}(x) < 1\}.$$

We use the stopping criteria

$$\|u_{j+1}^{n+1} - u_j^{n+1}\|_{L^\infty(\Omega)} < tol, \tag{4.20}$$

for values of tol chosen empirically and, when satisfied, we set $u^{n+1} = u_{j+1}^{n+1}$.

Space discretization For the space discretization, we consider the usual uniform mesh associated to image pixels contained in a rectangular domain, $\Omega = [0, L-1] \times [0, M-1]$, with mesh step size normalized to one. We denote by x_k a generic node of the mesh, with $k = 0, \dots, LM-1$, and by $u[k]$ a generic function u evaluated at x_k. To discretize the non-local diffusion term in space, we assume that u^n is a constant-wise interpolator, and to fix ideas, we use the common choice of spatial kernel used in bilateral theory filtering [Tomasi and Manduchi, 1998], that is, the Gaussian kernel

$$w(x) = \frac{1}{C} \exp\Big(-\frac{|x|^2}{\rho^2} \Big),$$

being C a normalizing constant such that $\int_{\mathbb{R}^2} w = 1$. Assuming that the discretized version of w is compactly supported in Ω, with the support contained in the box $B = B_{2\rho}(x)$, we use the zero order approximation (4.18)

$$\tilde{K}_{\epsilon,p}(u^n, u^{n+1})(x_k) \approx \sum_{m \in I_B^k} w[k,m] \tilde{k}_{\epsilon,p}(u^n[m] - u^n[k], u^{n+1}[m] - u^{n+1}[k]),$$

where $w[k,m] = w(x_k - x_m)$ and $I_B^k = \{m = 0, \dots, LM-1 : |x_k - x_m| < 2\rho\}$.
The values of the characteristic functions $\chi_{i,j}^{n+1}(x_k)$ of the set $\Omega \setminus \Omega_{i,j}^{n+1}$ are the last terms of (4.19) that we must spatially discretize. This is done by simply examining whether $u_j^{n+1}[k] > 0$ or not, for $\chi_{0,j}^{n+1}[k]$, and similarly for $\chi_{1,j}^{n+1}[k]$.
The full discretization of (4.19) takes the form of the following linear algebraic problem: For $k = 0, \dots, LM-1$, let $u^0[k] = f(x_k)$. For $n = 0, \dots, N-1$, set $u_0^{n+1}[k] = u^n[k]$. Then, for $j = 0, 1 \dots$ until convergence, solve the following problem: find $u_{j+1}^{n+1}[k] \in \mathbb{R}$ such that

$$\begin{aligned}(1-\tau a) u_{j+1}^{n+1}[k] - \tau\alpha \sum_{m \in I_B^k} w[k,m] \tilde{k}_{\epsilon,p}(u^n[m] - u^n[k], u_{j+1}^{n+1}[m] - u_{j+1}^{n+1}[k]) \\ + \frac{\tau}{r_j} \big(u_{j+1}^{n+1}[k] \chi_{0,j}^{n+1}[k] + (u_{j+1}^{n+1}[k] - 1) \chi_{1,j}^{n+1}[k] \big) = u^n[k] - \tau b[k]. \end{aligned} \tag{4.21}$$

The convergence of the algorithm is checked at each j-step according to the spatial discretization of the stopping criterium (4.20), that is

$$\max_{0 \le k \le LM-1} \| u_{j+1}^{n+1}[k] - u_j^{n+1}[k] \| < tol.$$

When the stopping criterium is satisfied, we set $u^{n+1}[k] = u_{j+1}^{n+1}[k]$ and advance a new time step, until $n = N-1$ is reached.

A simplified computational approach

In previous sections we have deduced, through a series of approximations, a discrete algorithm to compute approximated solutions of the obstacle problem $P_\epsilon(u)$ in (4.8). We have shown that our scheme is stable with respect to the approximating parameter r, producing solutions of problems $P_{\epsilon,r}(u)$ that, in the limit $r \to 0$, lie effectively in the image value range $[0, 1]$, apart from producing the required edge preserving saliency detection on images. In this section, by introducing some hard nonlinearities (truncations) to replace one of the iterative loops of (4.21), we provide a simplified algorithm for solving a problem closely related to (4.8). In addition, we use an approximation technique, based on the discretization of the image range, to compute the non-local diffusion term. These modifications allow for a fast computation of what we demonstrate to be fair approximations to the solutions of the original problem, (4.8). Considering the time discrete problem (4.17), we introduce two changes which greatly alleviate the computational burden:

1. Compute the non-local diffusion term fully explicitly, and
2. Replace the obstacle term by a hard truncation.

Thus, we replace problem $P_{\epsilon,r}(u^n)$ in (4.17) by the following which can be deduced from problem $P_\epsilon(u)$ in (4.8) using the two above strategies.

Truncated Problems $P_{\epsilon,0}(u^n)$ Given $u^0 = f$, and for $n = 0, \dots, N-1$, find $u^{n+1} : \Omega \to \mathbb{R}$ such that

$$P_{\epsilon,0}(u^n) \left\{ (1-\tau a)u^{n+1}(x) = \tau\alpha K_{\epsilon,p}(u^n)(x) + u^n(x) - \tau b(x) \right. \tag{4.22}$$

followed by a truncation of u^{n+1} within the range $[0, 1]$. Observe that the explicit Euler scheme, as remarked in [Pérez-LLanos and Rossi, 2011], is well suited for non-local diffusion since it does not need a restrictive stability constraint for the time step, as it occurs when considering the corresponding local diffusion operator. This is related to the lack of regularizing effect in non-local problems. Spatial discretization of (4.22) leads to the following algorithm which we shall refer as the *patch based scheme*:

Set $u^0 = f$. For $n = 0, \dots, N-1$, and for $k = 0, \dots, LM-1$, compute

$$u^{n+1}[k] = \left(\frac{\tau\alpha}{1-\tau a}\right) \sum_{m\in I_B^k} w[k,m] k_{\epsilon,p}(u^n[m] - u^n[k]) + u^n[k] - \tau b[k] \tag{4.23}$$

and truncate $u^{n+1}[k]$

$$\tilde{u}^{n+1}[k] = \min(1, \max(0, \tilde{u}^{n+1}[k])),$$

We shall show that there are very small differences between the solutions of the explicit truncated problem $P_{\epsilon,0}(u^n)$ and the solutions of the $P_{\epsilon,r}(u^n)$ problems for r sufficiently small. Nevertheless, the numerical scheme is greatly improved and much more efficient because costly iteration in r-loop is avoided, see Section **??**. We finally describe the efficient approach of [Yang et al., 2009] (see also [Galiano and Velasco, 2015] and [Galiano et al., 2016] for a related approach) that we use for computing the sum in (4.23), corresponding to the non-local diffusion term, by discretizing also the range of image values. Let $q = \{q_1, \dots, q_Q\}$ be a quantization partition, with $0 = q_1 < q_2 < \dots < q_{Q-1} < q_Q = 1$, where Q is the number of quantization levels. Let $v : \Omega \to [0, 1]$ be a quantized function, that is, taking values on q. For each $i = 1, \dots, Q$, we introduce the discrete convolution operator

$$K^i_{\epsilon,p}(v)[k] = \sum_{m\in I_B^k} w[k,m] k_{\epsilon,p}(v[m] - q_i), \tag{4.24}$$

where we recall that $w[k, m] = w(x_k - x_m)$. We then have

$$K_{\epsilon,p}(v)[k] = K^i_{\epsilon,p}(v)[k] \quad \text{if } v[k] = q_i, \quad \text{for some } i = 1, \dots, Q.$$

Notice that, for any v taking values in q, the computation of each $K^i_{\epsilon,p}(v)$ may be carried out in parallel by means of fast convolution algorithms, e.g. the fast Fourier transform. It is possible that, after a time iteration, a quantized iterand u^n leads to values of u^{n+1} not contained in the quantized partition q, implying that the new operators $K^i_{\epsilon,p}(u^{n+1})$ should be computed in a new quantization partition, say q^{n+1}. Since, for small time step, we expect q^n and q^{n+1} to be close to each other, we overcome this inconvenient by rounding u^{n+1} to the closest value of the initial quantization vector, q, so that this vector remains fixed.

The final simplified algorithm, which we call the *kernel based scheme*, is then:
Set $u^0 = f$. For $n = 0, \dots, N-1$, and for each $k = 0, \dots, LM-1$, perform the following steps:

- Step 1. If $u^n[k] = q_i$ then using (4.24)

$$\tilde{u}^{n+1}[k] = \frac{1}{1-\tau a}\Big(\tau\alpha K^i_{\epsilon,p}(u^n)[k] + q_i - \tau b[k]\Big). \tag{4.25}$$

- Step 2. $u^{n+1}[k] = q_j$, where $j = \underset{1\le i\le Q}{\operatorname{argmin}}|q_i - \tilde{u}^{n+1}[k]|$.

4.2.2 Local Total Variation Saliency model (LSTV) embedded in a CNN

We first detail the general local variational model for salency detection, which is to substitute the non local diffusion term by the TV as defined in 2.1.2. Let us consider the problem:

$$\arg \min_{u\in \mathrm{BV}(\Omega)\cap[0,1]} \quad P(u) + \frac{\lambda}{\alpha}F(u) + \frac{1}{\alpha^2}H(u), \tag{4.26}$$

where $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ is the image domain that we extend to also consider 3D images, $P(u)$, $H(u)$ and $F(u)$ are as usual, a regularizing functional, the saliency term and the restoration data fidelity term. The positive real parameters λ and α indicate, respectively, the relative importance of $F(u)$ and $H(u)$ with respect to the regularization term $P(u)$. Conversely to the non local model (4.7), in equation (4.26) we multiply the whole energy functional by α. The reason relies on the solving strategy we will use in the next section. In particular, the problem reads as follows:

$$\arg \min_{u\in \mathrm{BV}(\Omega)\cap[0,1]} \quad \mathrm{TV}(u) + \frac{\lambda}{2\alpha}\int_\Omega (u-f)^2 - \frac{1}{2\alpha^2}\int_\Omega (1-\delta u)^2 \tag{4.27}$$

where we recall TV(u) denotes the BV-semi norm $|Du|(\Omega)$. Following the results presented in the previous Section (Truncated Problems 4.22), we replace the obstacle term introduced through the indicator function $\mathcal{I}_I(u)$ by the truncation step such that the admissible set for the problem solution is $(\Omega) \cap [0,1]$. It is crucial to observe that depending on the parametric values this functional can be not convex but it is always a difference of convex functions. If we denote

$$F_1(u) = P(u) + \frac{\lambda}{2\alpha}F(u) = |Du|(\Omega) + \frac{\lambda}{2\alpha}\int_\Omega (u-f)^2\,d\mathbf{x}$$

and

$$F_2(u) = -\frac{1}{\alpha^2}H(u) = \frac{1}{2\alpha^2}\int_\Omega (1-\delta u)^2\,d\mathbf{x}$$

we can write (4.27) as:

$$\arg\min_{u\in K} E(u) = \arg\min_{u\in K} F_1(u) - F_2(u),$$

with F_1 and F_2 convex functions. This special structure, known as Difference of Convex (DC) functionals shall be exploited in the numerical resolution of the model. The structure and performance of the model can be again highlighted considering the Euler-Lagrange equation associated to the minimization problem (4.27):

$$-\Delta_1 u + \frac{\delta}{\alpha^2}(1-\delta u) + \frac{\lambda}{\alpha}(u-f) = 0 \tag{4.28}$$

$$a = \frac{\delta^2}{\alpha^2} - \frac{\lambda}{\alpha}, \qquad b(x) = \frac{\delta}{\alpha^2} - \frac{\lambda}{\alpha} f(x)$$

then (4.28) can be re-written as

$$-\Delta_1 u + b(x) = au$$

Recall that the constant a and the function $b(x)$ are computed from the parameters values and the data of the problem. We shall consider data $f \in Y$ and positive parametric values $(\alpha, \delta, \lambda)$ such that $a > 0$ and $b(x) >$ a.e in Ω. For feasible values of a and b we have that when u is small, say $u \approx 0$, absorption takes place but when u is large, say $u \approx 1$, reaction dominates. This mechanism acts as a contrast-enhancing filter during the iterations and promotes nearly binary solutions. When $\Omega \subset \mathbb{R}^2$, the quasilinear elliptic equation in (4.28) can be seen as the critical Sobolev exponent case given by the continuous inyection $BV(\Omega) \to L^2(\Omega)$. When $\Omega \subset \mathbb{R}^3$ we are in the supercritical case. It is also an eigenvalue type problem for the $1-$laplacian operator (see [Demengel, 1999] for a related problem). Nothing is known about the existence of solutions when (4.28) is complemented with homogeneous Neumann boundary conditions. Mathematically we can have a preliminary insight of the behavior of the model solutions observing that, when the non-linear non-smooth diffusion caused by the prior $P(u)$ takes place, the (linear) differential of the (quadratic) concave term $H(u)$ acts as a reaction term in the Euler-Lagrange equations of the minimization problem (4.26). As a consequence, nodal (i.e changing sign) solutions of the model can appear overflowing the range of $[0,1]$. This phenomenon has been observed also numerically, depending on the parameter values choice.

Numerical Implementation - Difference of Convex Functionals

The numerical resolution exploit the DC functionals structure of (4.27), which can be written in form

$$\arg \min_{u \in \mathrm{BV}(\Omega)} F_1(u) - F_2(u),$$

for $F_1(u) = \mathrm{TV}(u) + \frac{\lambda}{2\alpha} F(u) + \mathcal{I}_I(u)$ and $F_2(u) = -\frac{1}{\alpha^2} H(u)$, in where we have removed the box constraint $0 \le u \le 1$ by introducing the indicator function $\mathcal{I}_I(u)$, where $u \in I = [0,1]$, that enforces the solution u to be in the feasible set that it turn will later translates into a truncation step. Now, following Sampaio et al. [Sun et al., 2003] we find a critical point by iterating until convergence an explicit gradient ascent step on $F_2(u)$ with an implicit descent step on $F_1(u)$. This can be summarized in the general iteration

$$\begin{aligned} u_{k+1/2} &= u_k + \sigma F_2'(u_k), \\ u_{k+1} &= (\mathbb{I} + \sigma \partial F_1)^{-1}\left(u_{k+1/2}\right). \end{aligned}$$

Here σ is a positive step parameter which is set to 1 in our case for the sake of simplicity. Since $F_2(u)$ is differentiable, the ascent step is the following explicit update:

$$u_{k+1/2} = u_k + F_2'(u_k) = u_k - \frac{\delta}{\alpha^2}(1 - \delta u_k).$$

However, the case of the implicit descent step on F_1 requires calculating a proximal map that results in the following ROF-type [Rudin et al., 1992] minimization problem:

$$\arg \min_{u \in \mathrm{BV}(\Omega)} \mathrm{TV}(u) + \frac{\lambda}{2\alpha}\|u - f\|_2^2 + \frac{1}{2\sigma}\|u - u_{k+1/2}\|_2^2 + \mathcal{I}_I(u). \tag{4.29}$$

This non-smooth problem is strictly convex and therefore the proximal map is unique, but there is no closed form solution. We overcome this non-smoothness using the Chambolle and Pock primal-dual algorithm [Chambolle and Pock, 2011] to solve the its equivalent discrete saddle point problem. To that end, we make use of the Fenchel-Legendre transform of the TV term $P(u)$, in fact $P(\nabla u)$, and write:

$$\min_{u \in \mathbb{R}} \max_{p \in \mathbb{R}^2} \langle \nabla u, p \rangle - \mathcal{I}_P(p) + \frac{\lambda}{2\alpha}\|u - f\|_2^2 + \frac{1}{2\sigma}\|x - u_{k+1/2}\|_2^2 + \mathcal{I}_I(u), \tag{4.30}$$

where $\mathcal{I}_P(y)$ is the indicator function of the convex set $P = \left\{p \in \mathbb{R}^2 : ||p||_\infty \leq 1\right\}$.

Primal-Dual Algorithm Following [Chambolle and Pock, 2011], a general saddle point problem in the form

$$\min_x \max_y \langle Kx, y \rangle - J^*(y) + G(x),$$

can be solved by iterating

- $y^{n+1} = (\mathbb{I} + \tau_d \partial J^*)^{-1} \left(y^n + \tau_d K \bar{x}^n\right)$
- $x^{n+1} = (\mathbb{I} + \tau_p \partial G)^{-1} \left(x^n - \tau_p K^* y^{n+1}\right)$
- $\bar{x}^{n+1} = 2x^{n+1} - x^n,$

with τ_d, τ_p the step sizes corresponding to the dual and the primal step respectively. The final step is $\bar{x}^{n+1}$ has a similar effect as the Nesterov momentum commented in previous Section 2.2.5 in the sense it is an extrapolation step.

We can identify in our problem (4.30) K operator as ∇ and $J^*(y)$ as $\mathcal{I}_P(p)$ such that

$$G(u) = \frac{\lambda}{2\alpha}\|u - f\|_2^2 + \frac{1}{2\sigma}\|u - u_{k+1/2}\|_2^2 + \mathcal{I}_I(u)$$

and thus

$$\min_{u \in \mathbb{R}} \max_{p \in \mathbb{R}^2} \langle \nabla u, p \rangle - \mathcal{I}_P(p) + G(u) \tag{4.31}$$

that leads to the iterating problem

- $p^{n+1} = (\mathbb{I} + \tau_d \partial \mathcal{I}_P)^{-1} \left(p^n + \tau_d \nabla \bar{u}^n\right)$
- $u^{n+1} = (\mathbb{I} + \tau_p \partial G)^{-1} \left(u^n + \tau_p \mathrm{div} p^{n+1}\right)$
- $\bar{u}^{n+1} = 2u^{n+1} - u^n.$

Here, the calculation of the proximal map from $\mathcal{I}_P$ is straightforward. Since $\mathcal{I}_P(\cdot)$ is the indicator function of a convex set, its proximal map is simply a projection into this set (see for instance

[Parikh and Boyd, 2014] for more details). In the case of G, the calculation of $(\mathbb{I}+\tau_p \partial G)^{-1}(\tilde{u})$ results in solving the problem

$$\min_u \frac{\lambda}{2\alpha}\|u-f\|_2^2+\frac{1}{2\sigma}\|u-u_{k+1/2}\|_2^2+\frac{1}{2\tau_p}\|u-\tilde{u}\|_2^2+\mathcal{I}_I(u), \tag{4.32}$$

whose optimality condition reads as

$$-\left(\frac{\lambda}{\alpha}(x-f)+\frac{1}{\sigma}(x-u_{k+1/2})+\frac{1}{\tau_p}(x-\tilde{x})\right)\in \partial\mathcal{I}_I(x).$$

The solution to the proximal map is hence obtained by projecting into the interval $[0,1]$ the minimizer of the differentiable part of problem (4.32).

$$(\mathbb{I}+\tau_p\partial G)^{-1}(\tilde{x})=\textbf{proj}_{[0,1]}\left(\frac{\frac{\lambda}{\alpha}f+\frac{1}{\sigma}u_{k+1/2}+\frac{1}{\tau_p}\tilde{x}}{\lambda/\alpha+1/\sigma+1/\tau_p}\right)$$

which is to apply a truncation as we already mentioned. Finally, we are in the position of summarizing the complete algorithm in 4.1.

Algorithm 4.1: DC Primal-Dual algorithm

```
Given $f$ (input data) and $\lambda$, $\delta$, $\alpha$ (hyper-parameters) and fixed $\tau_d$, $\tau_p$ and $\epsilon_{tol}$ .
Set $u_0 = f$
while $\|u_{k+1} - u_k\|_2 \leq \epsilon_{tol}$ do
    $u_{k+1/2} = u_k - \frac{\delta}{\alpha^2}(1 - \delta u_k)$
    $x^0 = u_{k+1/2}$, $\bar{x}^0 = x^0$
    while $\|x^{n+1} - x^n\|_2 \leq \epsilon_{tol}$ do
        $y^{n+1} = \textbf{proj}_P(y^n + \tau_d \nabla \bar{x}^n)$
        $x^{n+1} = \textbf{proj}_I\left(\frac{\frac{\lambda}{\alpha}f + \frac{1}{\sigma}u_{k+1/2} + \frac{1}{\tau_p}(x^n + \tau_p \text{div}\, y^{n+1})}{\lambda/\alpha + 1/\sigma + 1/\tau_p}\right)$
        $\bar{x}^{n+1} = 2x^{n+1} - x^n$
    end
    $u_{k+1} = x^n$
end
```

To illustrate the qualitative performance of this model solved with algorithm 4.1 we show some results on different applications. In medical image we can also find situations in where such a method is useful. Considering our main application, i.e. MRI FLAIR images where a glioblastoma detection and segmentation is required. Figure 4.3 depicts two instances of Flair images in where tumor and edema are detected by our algorithm. As a second instance, we test it with natural (gray-scale) images. Depending on the contrast of the relevant region from the rest (which must be brighter according to the saliency term), these natural images will be labeled correctly, namely 0 or 1 for background and foreground regions respectively. A third application we consider here is background-subtraction 4.5. Background-subtraction (or foreground detection) aims to detect objects in a scene having a given model of the background. Such technique is widely used for tracking detection and other tasks in the field of dynamic vision. The basic subtraction between a frame and the background results in a suitable image to be processed with our method.

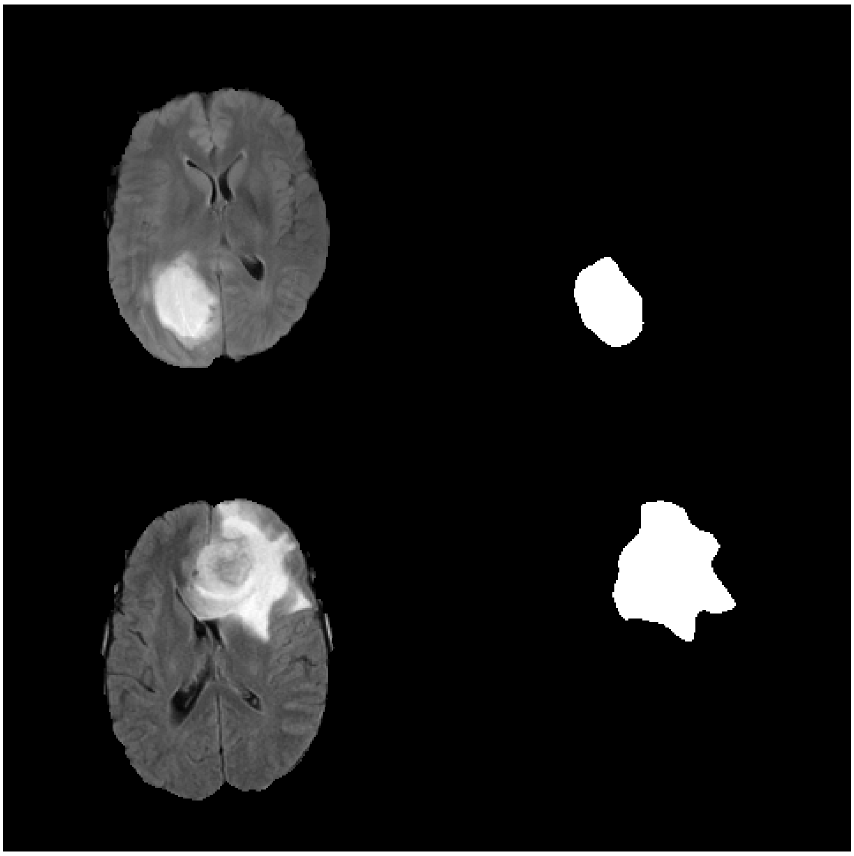


Figure 4.3: Left column: original image. Right column: Output of the algorithm. Parameters: $\lambda = 1$, $\alpha = 1$, $\delta \approx 2$.

LTVS embedded in a CNN

With the local version of our saliency model and its DC-based primal-dual numerical resolution we now aim at overcoming the standard parameter tuning problem by training a CNN with the presented model end-to-end. Recall that, finding optimal parameters of a variational model to perform saliency detection providing accurate nearly binary images (segmentation) of the region of interest (glioblastoma) is an expensive and difficult task that is often carried out by the ultimate user of the method, in this case, possibly a medic. Such a task requires however a deep understanding of the nature of those hyper-parameters and how they interact to each others. In other words, we are asking to the medicine expert to learn the functioning of the model to adjust for each input image (and according to it) the correct and optimal parameters. The motivation of the following proposed framework is to replace the expert by a NN that only will learn, based on data, this optimal hyper-parameterization.

NNs are conceived to learn, so there is not a great innovation training such a structure. However, in our case, we do not have or want to get the ground-truth of hyper-parameters. In fact, the goal is to train the CNN to provide the optimal parameters for our model indirectly, this is, without an explicit known target. The only ground-truth are the final segmentation that are compared with the output of our variational model.

Deep Learning Framework model formulation We detail the formulation of the Deep Learning model constrained to our Variational Saliency model. This formulation can be seen as a bi-level problem.

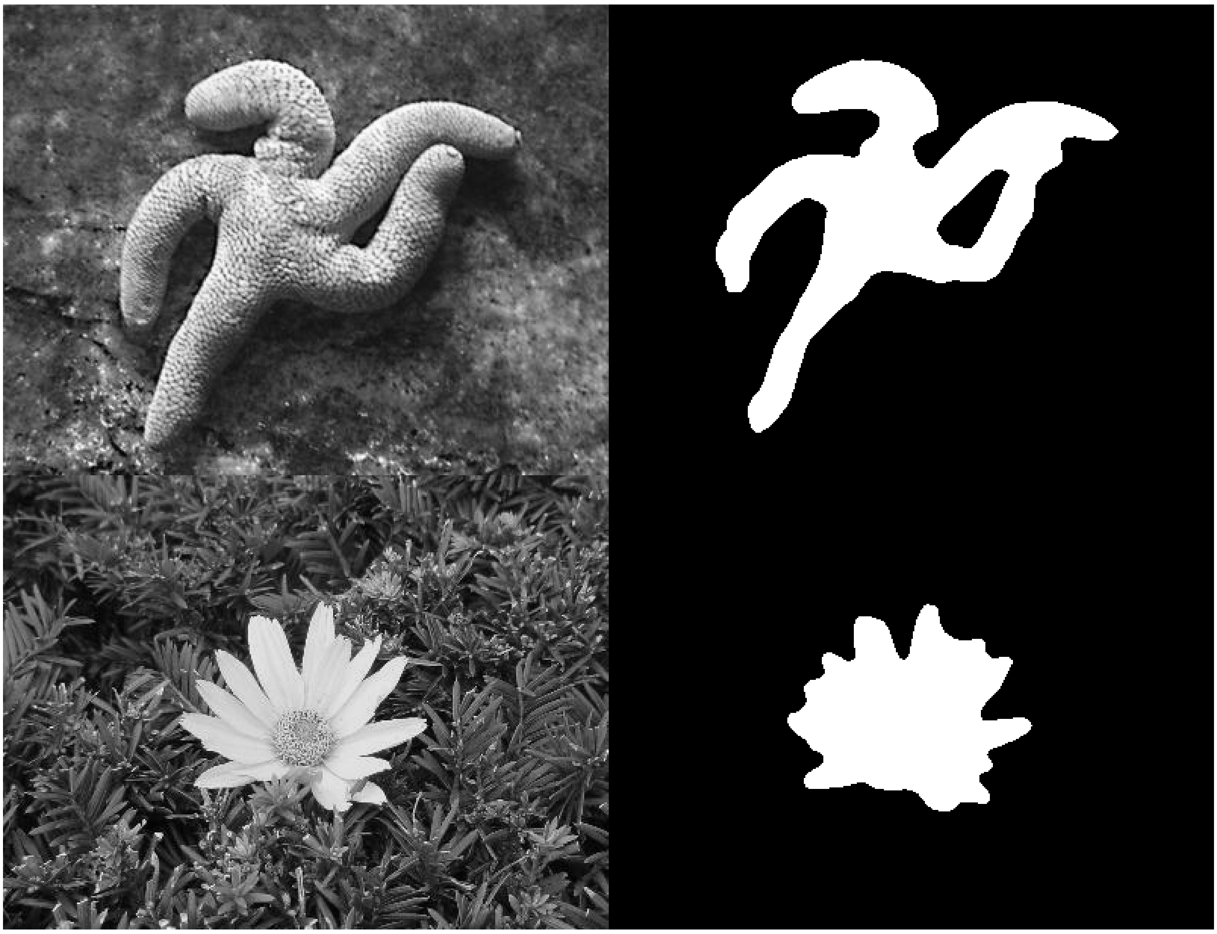


Figure 4.4: Example for natural images. Left column: original image. Right column: Output of the algorithm. Parameters: $\lambda = 1$, $\alpha = 2$, $\delta = 2$.

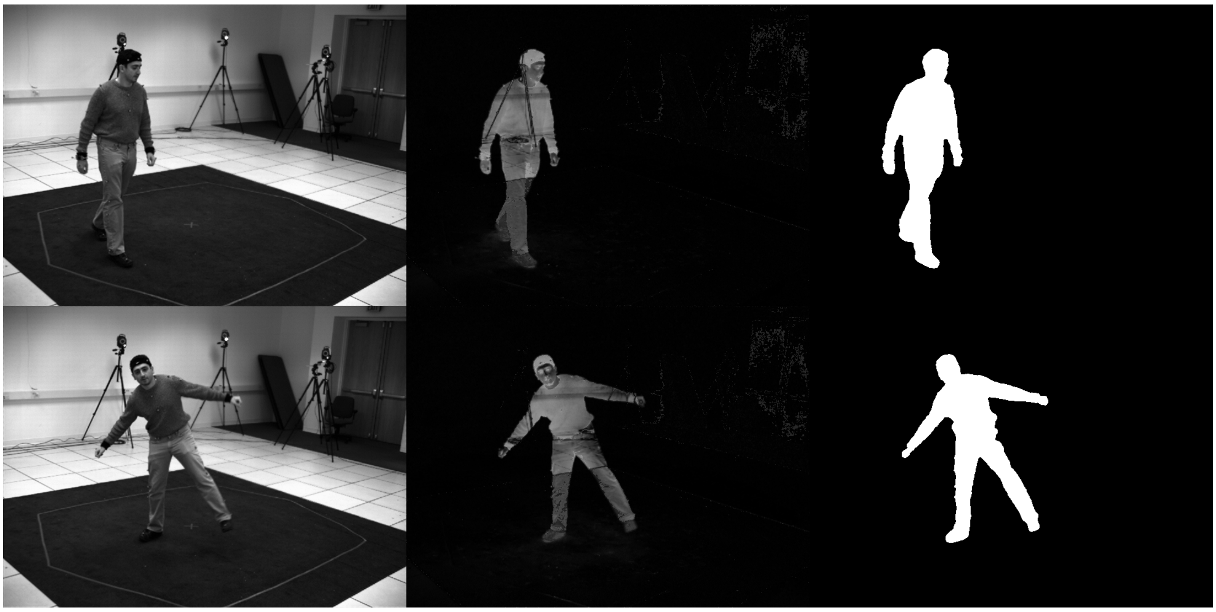


Figure 4.5: Left: original image. Middle: background subtraction (input image). Right: Output of the algorithm. Parameters: $\lambda = 20$, $\alpha = 4$, $\delta = 10$. The background subtraction is performed with RGB images. The given input (middle) is converted to a gray-scale image.

The upper problem is derived from a Maximum Likelihood or Maximum a Posteriori scheme. The lower problem acts as a constrain and consists of solving an optimization problem, in our case, our proposed variational problem. The former requires a whole data set since it aims at finding optimal parameters of a NN while the latter transform the input into a nearly binary segmentation output. Given a dataset $\mathbf{F} = \{f^{(1)}, f^{(2)}, ..., f^{(N)}\}$, $\mathbf{Y} = \{y^{(1)}, y^{(2)}, ..., y^{(N)}\}$, we define the following target cost function to minimize:

$$\arg\min_{\theta} \frac{1}{N} \sum_{i} \int_{\Omega} u^{*(i)}(\theta, f^{(i)}, \mathbf{x}) - y^{(i)}(\mathbf{x})^2 d\mathbf{x} \tag{4.33}$$

$$\text{s.t.} \quad u^{*(i)}(\theta, f^{(i)}, \mathbf{x}) = \arg\min_{u^{(i)}} \frac{\lambda(\theta, f^{(i)}, \mathbf{x})}{\alpha(\theta, f^{(i)}, \mathbf{x})} \int_{\Omega} (u^{(i)}(\mathbf{x}) - f^{(i)}(\mathbf{x}))^2 d\mathbf{x} + \\ + \mathrm{TV}(u^{(i)}(\mathbf{x})) - \frac{1}{\alpha^2(\theta, f^{(i)}, \mathbf{x})} \int_{\Omega} (1 - \delta^{(i)}(\theta, f^{(i)}, \mathbf{x}) u^{(i)}(\mathbf{x}))^2 d\mathbf{x}$$

where f are the given data s.t. $u_0 = f$ (in the numerical resolution scheme), θ are the Neural Network parameters to be optimized. Note λ, α and $\delta(\mathbf{x})$ depends on Neural Network parameters and the input f. Providing adaptive parameters gives also quite control over them. For instance, λ and α that are scalar parameters can be bounded to a 'save' range, and parameters that are suitable to be adaptive as $\delta(\mathbf{x})$ can be easily regularized with some priors.

We have already shown that the δ parameter must be set in an adequate range ($\delta \geq 1$) such that the concave Saliency term is able separate in two class the input. Moreover, the effect of δ is to drive the variational model increasing or decreasing the reference threshold along the image. One can think that such parameter may be smooth and, consequently, use a smooth prior potential function, for example $\psi(s) = |s|^2$, recovering the well-known Tikhonov regularization. The upper problem then is

$$\arg\min_{\theta} \frac{1}{N} \sum_{i} \int_{\Omega} u^{*(i)}(\theta, f^{(i)}, \mathbf{x}) - y^{(i)}(\mathbf{x})^2 d\mathbf{x} + \psi(\delta^{(i)}(\theta, f^{(i)}, \mathbf{x})). \tag{4.34}$$

In order to be able to train the model we need to fix the number of iterations of our model (resulting in only 2 for the outer loop and 10 for the inner). Then, each step is unrolled as an added layer to the whole network as it is done in [Riegler et al., 2016]. The resulting scheme including the whole model is depicted in Figure 4.6.

The proposed CNN is inspired in a U-Net architecture [Ronneberger et al., 2015]. The number of features in each layer is fixed to 64 with filters kernel size 3 × 3. Moreover, we use a MLP to estimate the remaining parameters λ and α. Since we randomly initialize all weights of filters, the estimated parameters at the beginning are random too, consequently, the training procedure takes long time. We aim at alleviate this problem using a so-called Transfer-Learning technique [Pan and Yang, 2010] which consists in re-training a pre-trained network, as it is done in next chapter 5. We thus drop the last convolutional layer of the CNN, adding 2 more convolutional layers and re-train the full architecture.

We show in Figure 4.7 some qualitative results of the Deep Variational Framework. It is of particular interest the predicted adaptive $\delta(\mathbf{x})$ map. Each of them seems to compensate those regions of the input that, although they are tumoral according to the ground-truth, are also darker than the rest. This is done by decreasing the threshold reference (increasing δ) in those regions. A rough way to interpret those δ maps is as they are a guidance for the variational model. Likewise, the scalar hyper-parameters that we have also considered, λ and α are depicted in Figure 4.8 during training, where each of them converge to

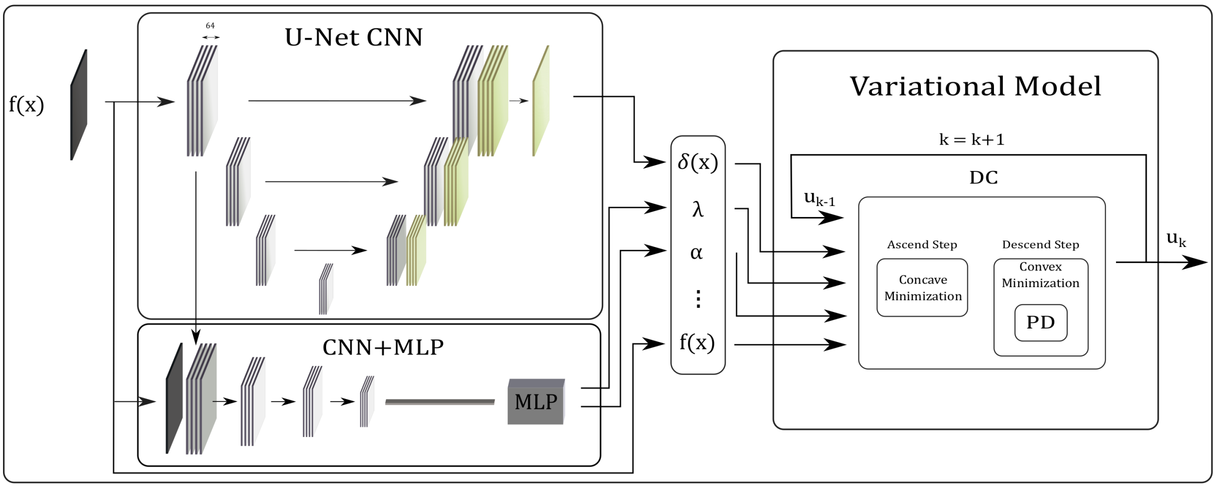


Figure 4.6: Scheme of the proposed Deep Variational Framework. The CNN provides an adaptive parametrization of the variational model through the $\delta(x)$ function. A MLP allows the estimation of the remaining parameters which are fed to the variational model in a top layer of the global Deep Variational Framework.

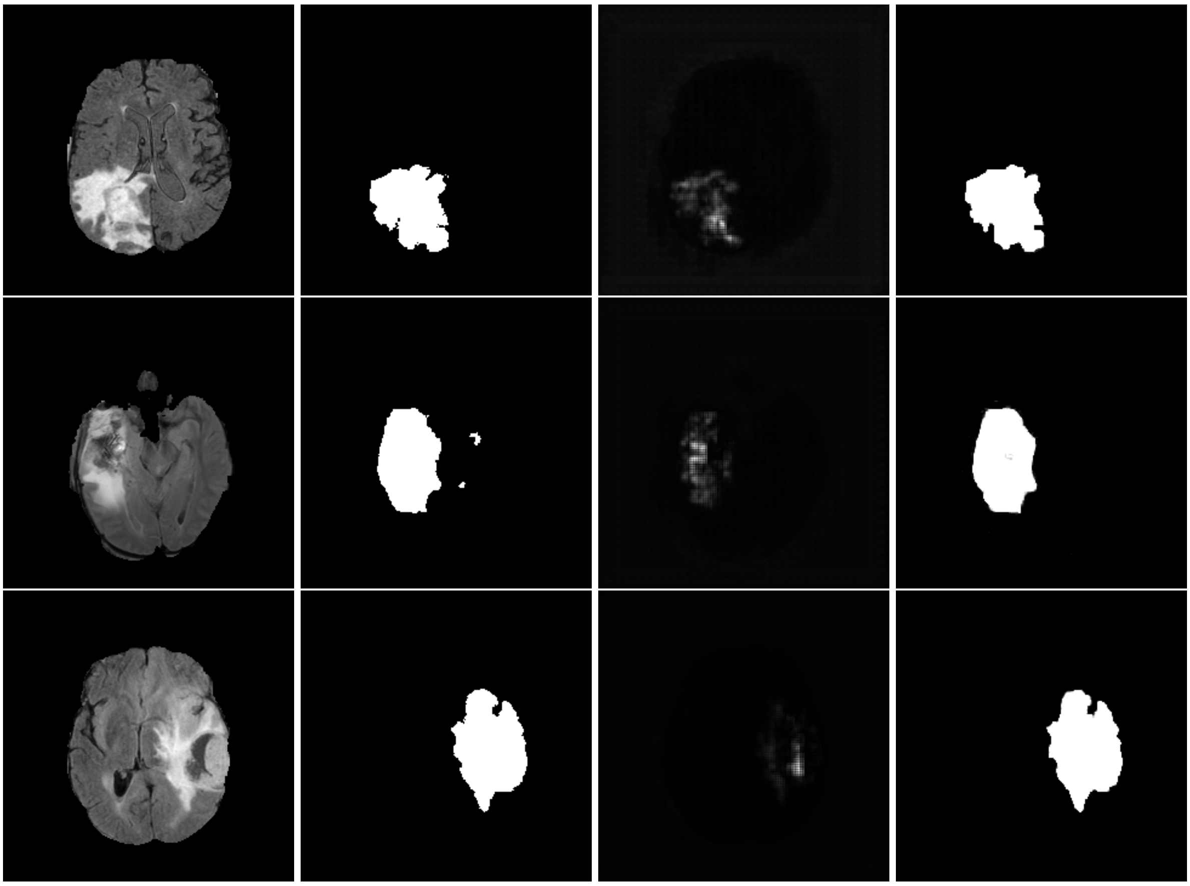
some DC component while they vary depending on the inputs. As expected, λ takes a little value since the fidelity term is measuring a distance from the given datum f, and the outcome u is a nearly binary image, and thus, quite different.

4.3 Experimental results and discussions

In this Section we describe the numerical experiments we have carried out. Images from the BRATS2015 dataset [Menze et al., 2015] are considered and processed.

In the non local setting, the first experiment is to solve the discrete problem (4.19) using the hard truncation technique proposed in (4.22) and compare the results with those obtained with the iterative scheme (4.17) where the Yosida's Approximants are used. The numerical experiment shows a great improvement in terms of time consumption and minor difrefences in the final segmentation when the hard truncation scheme is considered. We then choose (4.22) as a base for subsequent analysis. In a second test the proposed quantized kernel based approach in 4.25 is compared on some sample images with the patch based scheme in 4.23. It turns out that the kernel approach, based on the Fast Fourier Transform, allows an ulterior speed-up of the computation when large size kernels are implemented, contrary to the patch based approach which demands a prohibitive increasing amount of time computation. As a result of the above analysis we choose the kernel based approach 4.25 for solving the truncated problems in (4.22). As an application we test our numerical methods over the whole dataset. Different values of p are considered to show that the non-local non-convexities, associated to small values of p, attain the best scores in classical metrics for image segmentation. This highlight also the edge-preserving property of the non-local reactive flows we propose. Third, we show that our model can be generalized to a fully 3D model presenting some preliminary results which extend our variational non-local saliency approach.

In the local setting, finally, we train and test using the BRATS2015 dataset or Deep Variational Framework and compare with: 1) baseline (a U-net CNN), 2) our local variational model (TVS) and the proposed architecture, TVS+CNN. The results show a huge improvement when combining both,



Input Ground-truth $\delta(\mathbf{x})$ Output

Figure 4.7: Outputs and adaptive $\delta(\mathbf{x})$ computed by the CNN.

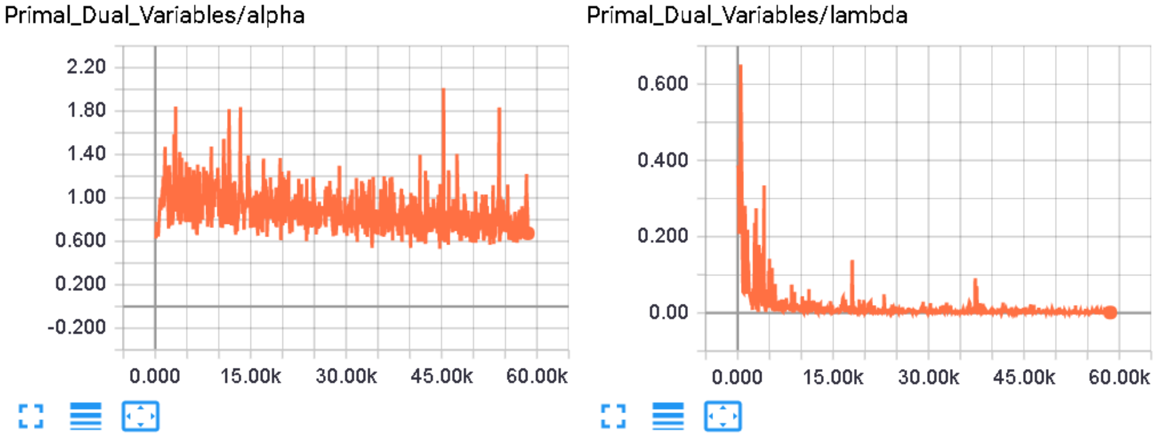


Figure 4.8: Scalar estimated hyper-parameters: λ, α during training.

Variational and DL tools. While our model, with fixed and manual hyper-parameterization performs poorly compared with the CNN, the automatic adaptive parameters provided by the Deep Variational architecture boosts our variational model outperforming the single CNN and approaching the state of the art results.

4.3.1 Non local TVS model

Experiment 1: Comparison between limit approximation and truncation In this experiment we show the differences between the limit approximation $r \to 0$ described in Section 4.2.1 and the proposed truncation alternative given in (4.22). We recall that the purpose of such hard truncation is to get rid of the r-loop in the numerical resolution, boosting the computation efficiency. In practice, instead of using the stopping criteria 4.20, it is sufficient (and more efficient) to fix a small number of iterations which results into 5 in the r-loop starting with $r^1 = 0.5$ and setting $r^{j+1} = 2^{-j} r^j, \;\; j = 1, \ldots, J, J = 5$. We found in our experiments that this is enough in order to ensure that the final output of the approximating scheme (4.19) is very close to the solution of 4.23.

Each j-step consists of solving the equation (4.19) which is carried out through a conjugate gradient descent algorithm. This is an inner loop for each j-step which increases substantially the global time execution of the algorithm. In order to show that the hard truncation is a good strategy to get rid of the r-loop we compute the relative differences $||u_J - u_T||_2/||u_T||_2$ between each j-step image (u_J) and the truncated version (u_T) of the n-step solution. At each step of the r-loop, the relative difference from the final truncated version is reduced (see Figure 4.10). Figure 4.9 depicts the qualitative difference of using truncation. For all the subjects we tested the results clearly show that the same saliency (tumor) region is detected in both images. The differences, barely visibles, are colored in red. Indeed only few pixels differ from the assumed correct solution calculated through the r-convergence scheme. This justifies the use of the hard truncation.

Experiment 2: Kernel based approach In this experiment we compare the time execution between patch based numerical resolution and the proposed kernel approach based on [Yang et al., 2009]. Taking advantage of the fact that convolutions can be fast computed in Fourier domain, we use a GPU implementation to carry out these experiments. In both cases we fix the same hyper-parameters and perform a sweep where $\rho = 2, 3, \ldots, 30$ is the kernel radio and $q = 2^3, 2^4, \ldots, 2^{11}$ are the quantization levels. The tests are performed over 4 brains (2 slices per brain) and results are averaged.

Notice that in a classical patch based approach no quantization is required and the time execution will grow up with the size of the considered region $(B_{2\rho}(x))$. On the contrary, a kernel based resolution remains robust to different kernel sizes while the time execution depends mainly on the number of quantization levels as it can be seen in Figure 4.11. This justifies the use of the kernel based method whereas it allows to use bigger kernels properly modelling the non-local diffusion term.

Experiment 3: 3D versus 2D model Focusing in the particular application of brain tumor segmentation, it is reasonable to argue that processing each image (slice) independently will result in a sub-optimal saliency segmentation since no axial information is taken into account. Our model can be easily extended to process 3D brains volumes so that the non-local regularization will prevent from false positive classification using 3D spatial information. The results are greatly improved as it is reported in Table 4.1 and shown in

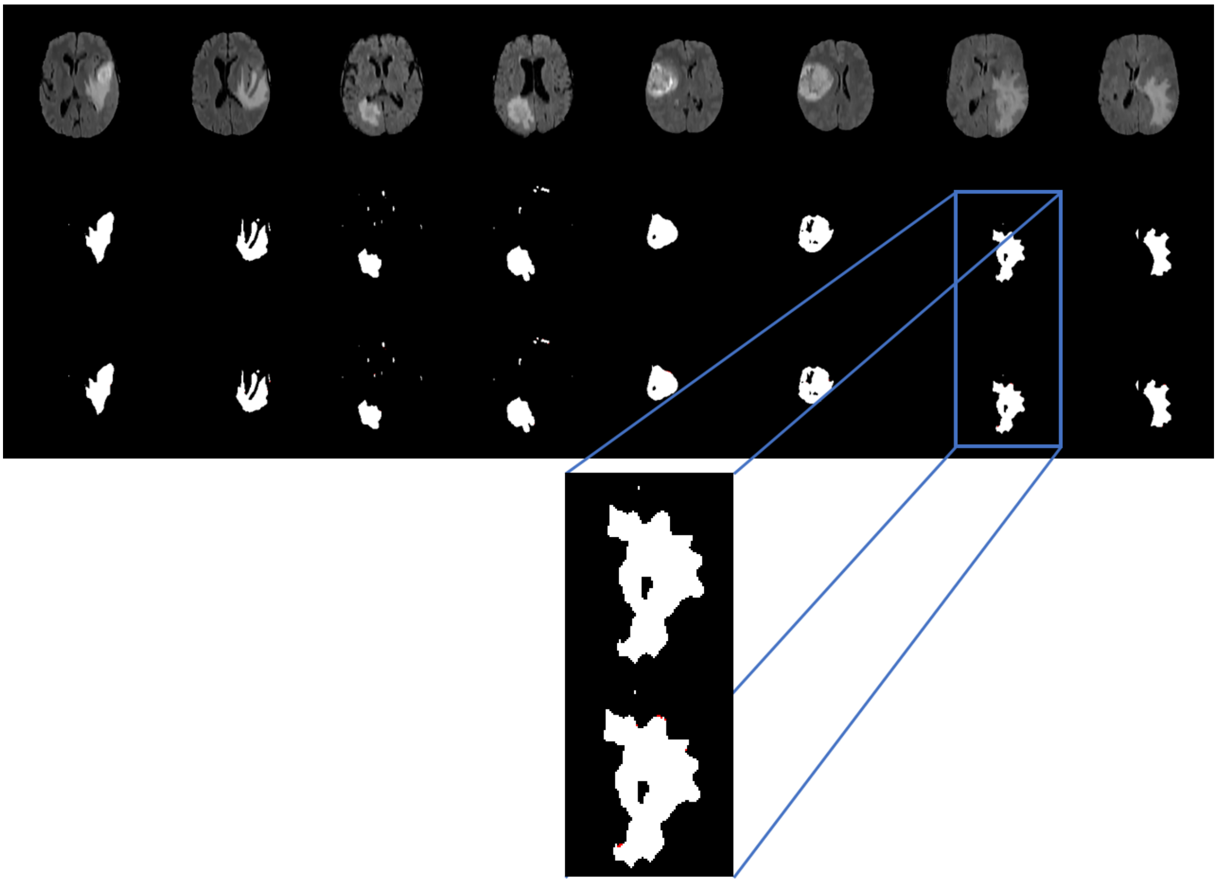


Figure 4.9: A comparison between the Yosida's approximating solutions and the hard truncated solutions. In first row: input images, second row are the respective outputs for the r-convergence approximation method and the last row, the hard truncation approximation. Differences in the final solution are colored in red.

Figure 4.12. It is easy to see that some artifacts arise when processing the volume slice by slice (2D, first image), which disappear if a fully 3D scheme is considered (second image). False positives can also be avoided in this 3D approach obtaining a cleaner image that results in a very high accuracy in common metrics (see Table 4.1).

Even though the results obtained with the fully 3D scheme show promising performance with an improved final segmentation w.r.t. 2D slice by slice processing a question remains about the existence of a proper and robust δ-parameter feasible for the whole 3D volume. This is due to the non-homogeneous contrast and illumination (bias) in different regions of the MRI image which depends on the acquisition step. We aim at overcoming this issue in the local experiment Section 4.3.2.

	Accuracy	Specificity	Precision	Recall	Dice
2D	0.96285	0.98403	0.86578	0.79856	0.83082
3D	0.98108	0.99595	0.96488	0.86557	**0.91253**

Table 4.1: Results of a 3D and 2D resolution with the same parametrization and $p = 0.5$

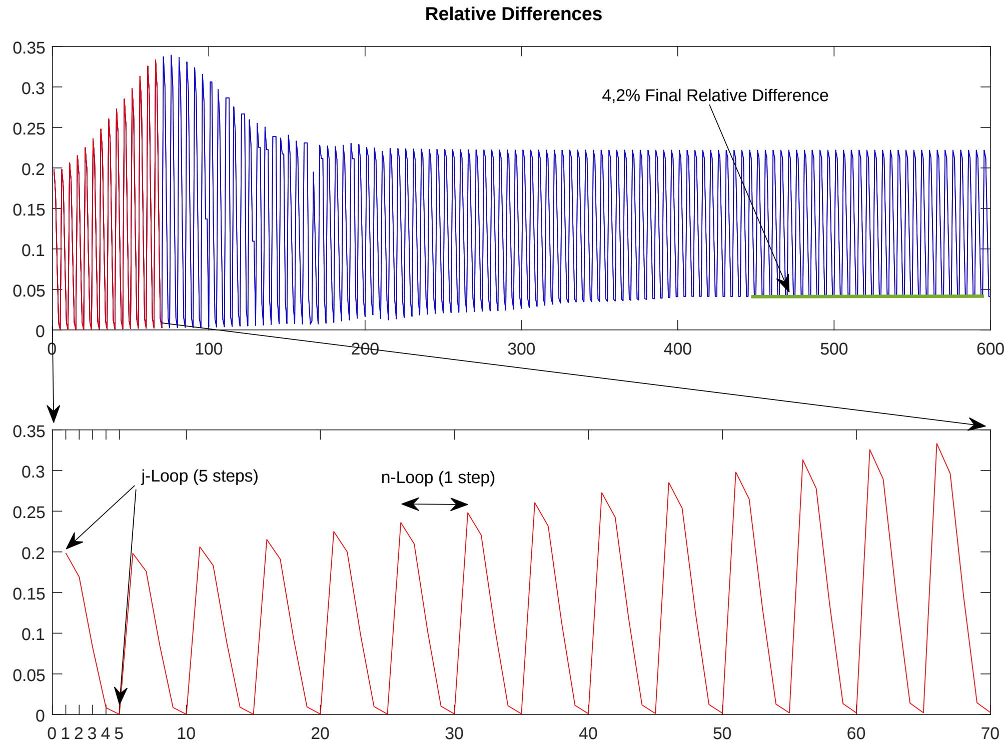


Figure 4.10: Relative differences: $r-$convergence vs hard truncation. Stabilization of the $r \to 0$ limit. The outer time n-loop, $n = 1$ ... N is considered together with the inner j-loop, $j = 1$ ... J. The jumps from u_5 to u_6, u_{10} to u_{11} etc are caused by the truncation with J fixed to 5 for each n. We see in Figure 4.9 that the (binary) output solution is practically indistinguishable from the almost binary approximations in r. The final outputs only differ a 4,2%, which in practice turns out to be few single pixels.

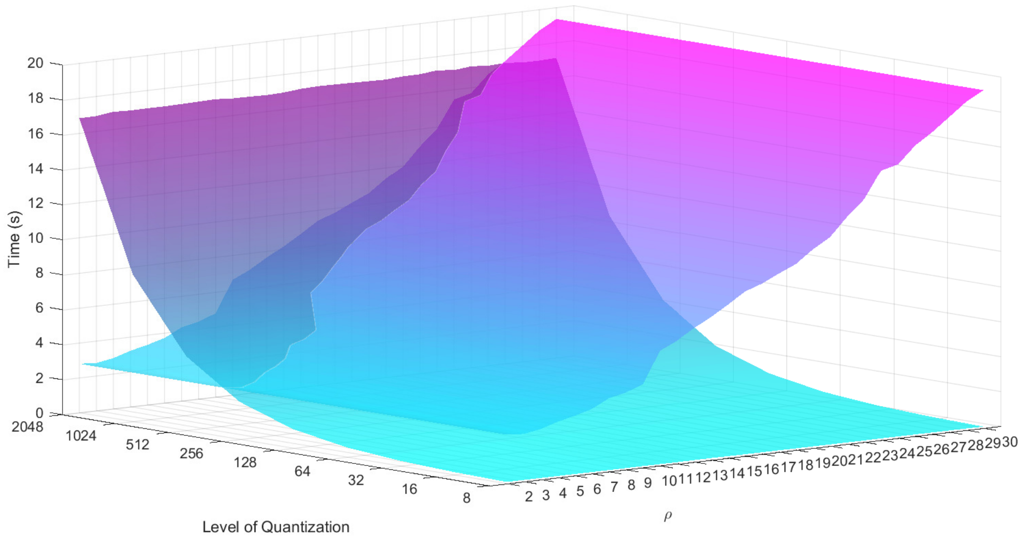


Figure 4.11: Time comparison between patch based and kernel based resolution. The surface which remains constant with the quantization levels corresponds to the patch based resolution while the other one corresponds to the kernel based resolution. Using a kernel based approach allows a nearly invariant dependency w.r.t. the size of ρ (radius of the kernel), which in turn promotes the non-locality effect.

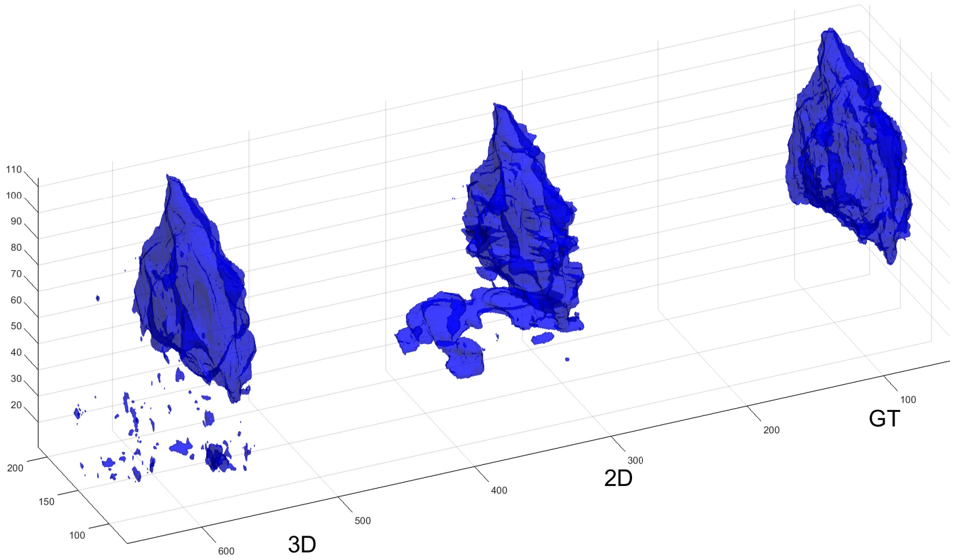


Figure 4.12: From left to right: 2D reconstruction (our proposed model applied slice by slice), 3D reconstruction (our proposed model applied to the whole volume), Ground-truth. A significant number of false-positives is reduced when the fully 3D finite difference schemes are applied.

Results: MRI Dataset

As a result of the previous experiments we reduce the computational time by using the hard truncation scheme and model the non locality with the kernel based approach. We then apply our above findings testing the whole BRATS2015 dataset, a set of 16114 images after removing those slices where there is no brain. Each image is re-scaled to the range $[0, 1]$. In order to study the effect of the proposed non-local non-convex hyper-Laplacian operators we shall consider the behavior of the model fixing all the parameters $p, \epsilon, \alpha, \lambda, \rho, \tau$ of the model and letting the p-parameter to vary. We introduce a simple automated rule for the δ-parameter estimation which avoids a manual tuning of the model for each image of the dataset. This results in a sub-optimal performance of the model in terms of accuracy but it will provide us with an estimation of the reach of our proposed model. Observing that $1/\delta$ acts as a threshold between classes (background and foreground), we seek a rule to determine such a threshold for each image leading to an approximately correct estimation of δ. We observe that the reaction flows generate nearly binary solutions and this simplifies our task. By averaging the whole given brain (values of pixels where there is brain), and comparing with the average of the tumor intensities, it turns out (see Figure 4.13) that the relationship is nearly linear, and a simple linear regression gives a prediction of the mean of the tumor in the considered image:

$$\mu_{tumor} \approx \alpha\mu_{brain} + \beta = 1.176\mu_{brain} + 0.101$$

We then select a reference threshold. A simple choice is to compute the average between μ_{brain} and μ_{tumor} in form $(1/\delta) = (\mu_{brain} + \mu_{tumor})/2$. Finally, since it is always possible to compute the average of the whole brain (μ_{brain}), we end up with the following rule for the δ-parameter estimation which depends on the given image:

$$\delta(\mu_{brain}) = \frac{2}{(1+\alpha)\mu_{brain} + \beta}$$

Our proposed model includes hyper-Laplacian non-local diffusion terms by setting $0 < p < 1$. We also

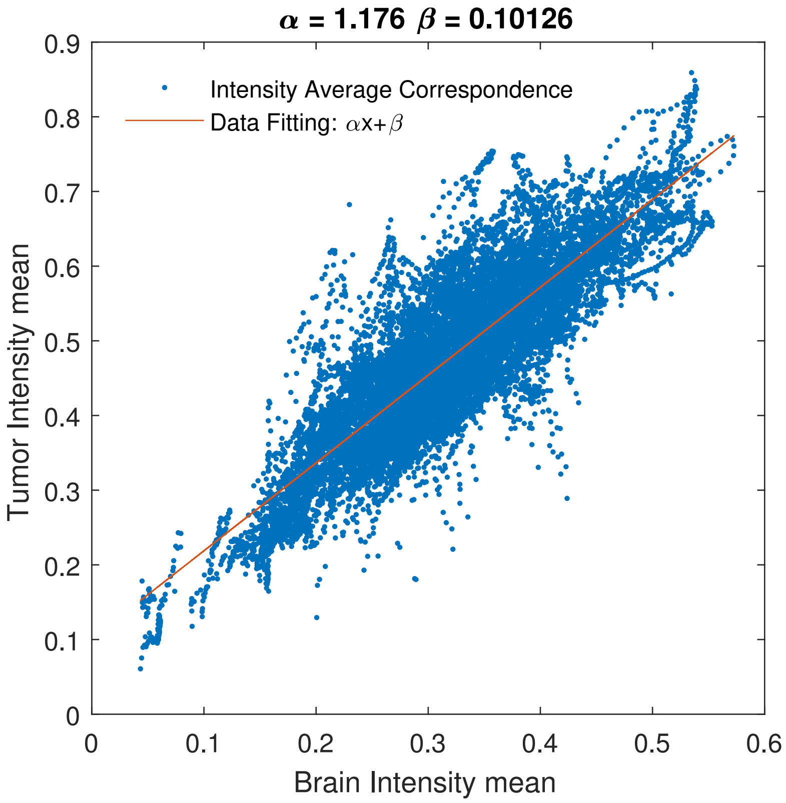


Figure 4.13: Distribution of the mean of the tumor intensities obtained from the ground-truth provided in the BRATS2015 dataset. The linear regression mapping the brain intensity mean μ_{brain} to the tumor intensity mean μ_{tumor}.

compare different values of p ($p = 2, 1, 0.5$) with the same parametrization. The results, in terms of typical reference metrics, [Powers, 2011], are shown in Table 4.2 and they indicate that the Dice measure is monotonically increased as p is decreased.

p	Accuracy	Specificity	Precision	Recall	Dice
	$\frac{tp+tn}{tp+fp+tn+fn}$	$\frac{tn}{tn+fp}$	$\frac{tp}{tp+fp}$	$\frac{tp}{tp+fn}$	$\frac{2tp}{2tp+fp+fn}$
2	0.99089	0.99337	0.59594	0.79039	0.64844
1	0.99303	0.99562	0.67992	0.77977	0.70128
0.5	0.99425	0.99735	0.76575	0.73214	**0.72755**

Table 4.2: Results obtained for different p values using the whole BRATS2015 data-set. Metrics are defined in the first row where: tp-true positives, tn-true negatives, fp-false positives, fn-false negatives. The Dice measure which accounts for a compromise betweeen Precision and Recall is improving a 5.28% and a 2.62% when p takes values $p = 1$ and $p = 0.5$, respectively.

4.3.2 Local TVS - Deep Variational Framework

Results: MRI Dataset

In order to evaluate the performance of our proposed local model we compare three different settings. The first, referred in the following as 'TVS'(Total Variation Saliency), is our baseline: the variational model (4.27) with fixed manually optimized parameters ($\lambda = 0.1, \alpha = 0.75, \tau_p = \tau_d = 0.3, \sigma = 1$), and a single δ for the whole image which is heuristically estimated from the available data as we have done with the Non local Saliency model. The second, referred as 'CNN', is the stand-alone U-Net type CNN trained to directly segment high grade gioblastomas. The third, referred as 'CNN+TVS', is our proposed Deep Variational Framework combining the CNN and the variational model. The numerical resolution is carried out in a Tensorflow based framework [Abadi et al., 2015] using a GeForce GTX 1080Ti GPU.

We test these three methods again on the BRATS2015 dataset using only slices images with glioblastoma strokes. Moreover, the considered dataset is split in training and test sets (80% for training, 20% for test). The results are briefly summarized in table 4.3 were the commonly used Precision, Recall and Dice metrics of each method are reported over the whole dataset. In all of them the 'CNN+TVS' proposed framework achieves the best results showing that the optimization of the parameters of the variational model given by the CNN framework in an improved and more robust performance.

	Precision	**Recall**	**Dice**
TVS	0,739	0,632	0,655
CNN	0,740	0,875	0,791
CNN+TVS	**0,845**	**0,882**	**0,857**

Table 4.3: Summary of results of the considered methods for brain tumor segmentation.

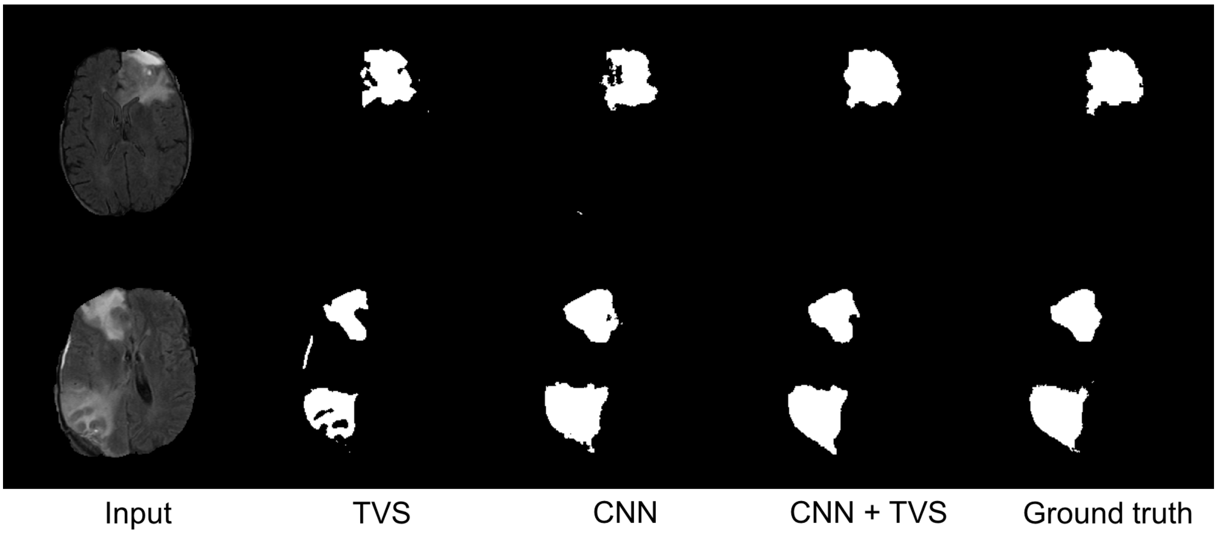


Figure 4.14: From left to right columns: input FLAIR-glsmri images, TVS, CNN, CNN+TVS and Ground-truth segmentation. Notice the output of the tree settings are nearly binary, thus a simple thresholding (0.5 in the range [0,1]) is applied in order to get binary segmentations.

In Figure 4.14 we can see the segmentation of the tumors achieved for the methods considered in two different cases. From left to right in the image we can see the input images, the segmentation performed by TVS, CNN, CNN + TVS and the provided ground truth. On the one hand, it is clear that the results from the TVS model depend too strongly from the global intensity differences in the image and miss to detect as parts of the tumor areas that have a lower local intensity. On the other hand, in the case of the stand-alone CNN, we observe some false positives and a stranger delineation of the tumors when comparing to the ground truth. Finally, the proposed CNN + TVS model is able to achieve a better localization and segmentation of the whole tumors by using the power of the CNN to correct for the problems observed in the TVS related with the global intensity changes while also outperforming the stand-alone CNN by adding the spatial information given by the TV regularizer.

CHAPTER 5

P2. Deep Learning for Dumpster recognition and classification

If a machine is expected to be infallible, it cannot also be intelligent.
Alan Turing

This chapter is organised as follows. In Section 5.1 we introduce the problem of dumpster recognition using DL and its motivation. Then, in Section 5.1.1 the selected NN architecture and the image dataset used are depicted. In Sections 5.1.2 and 5.2, we detail the modeling and methodology proposed to achieve high performance using very little number of samples in training step. In the last Section 5.3 we present our results.

5.1 Introduction: Recognition and classification of Dumpsters

In the last years computer vision is witnessing an outstanding revolution, mainly due to three factors. Firstly, there are inexpensive parallel computing platforms such as GPUs and other hardware accelerators (locally), and clusters of computers and cloud computing (on-line). Secondly, there are fast and inexpensive devices that can store huge volumes of labeled data. Thirdly, there are several open source libraries, developed by prestigious corporations, that implement those methods under the umbrella of DL. Recall that, as described in Section 2.2, Deep Learning is nothing but a tool from a wider ML field, where "deep" reefers to the number of layers in a NN. Supported on these triad, classification tasks have improved their performance in many competitive challenges such as CIFAR, [Krizhevsky et al., 2012]. This improvement has a positive impact on applications such as object detection and tracking [Sermanet et al., 2014, Li et al., 2014], segmentation [Kolesnikov and Lampert, 2016], human pose estimation [Cao et al., 2017], or visual attention and saliency [Ba et al., 2014]; but it has also made possible and successful others that were unreachable a decade ago, namely image and video captioning and description [Fang et al., 2015, Lebret et al., 2015], or question answering [Fukui et al., 2016].

Despite of these excellent results, a computer vision practitioner attempting to solve a particular problem with any of the available DL libraries usually finds a number of difficulties. First of all, it is necessary a large training data set; otherwise the model will tend to over-fit, even if dropout is used, [Zheng et al., 2014]. Data augmentation is a solution when images keep their meaning under rotations and deformations. For instance a pedestrian upside down makes no sense when trying to detect it on a street, but numbers in an envelope can take any angle if it is in a conveyor belt with a camera on the top. In addition, since most of the applications involve a classification task at some point, the data set must be labeled. So even if the acquisition of images for each class is fast, the labeling process may require a considerable time and human effort. Another shortcoming is that DL presents much more degrees of freedom when it comes to decisions on the architecture of the net. One has to decide not only the number of hidden layers in a FCNN, but also (at least) the number of convolutional layers, the size of convolution kernels in them, and the number of pooling layers. In other words, the hyper-parameterization of a DL solution is often a barrier to a successful usage of such techniques (we will address this issue in chapter 6). Moreover, currently there are many different architectures different from a CNN followed by a FCNN, usually known as AlexNet, [Krizhevsky et al., 2012], or LeNet, [LeCun and Bengio, 1995] depending on the number of layers. Others are, for example VGG-Net, [Simonyan and Zisserman, 2014], GoogleNet, [Szegedy et al., 2015], ResNet, [He et al., 2015] and Single-Shot Detectors, [Liu et al., 2016]. Trying many combinations is also a problem because training time usually takes hours, and even days, depending on both the size of the data set, the architecture of the net and the computing resources. Nowadays a common DL neural network architecture can easily have hundred of thousands of parameters and sometimes even hundred of millions! LeNet-5: 60.000 parameters, AlexNet: 60 millions, Inception v3: 23 millions, VGG: 138 millions and so forth.

Due to the reasons above, there is a growing interest both in transfer learning and semi-supervised learning, not only in the scope of DL but also, more broadly, in the ML community. In transfer learning the *target* class, i.e. the one we actually want to distinguish, is learned not only with data but also with knowledge extracted from *source* tasks; which are also classification tasks but in other domains, with other data sets, or with different classes. Thus, in Bayesian Transfer the source tasks provide a prior to the data and the target task is the posterior, see for instance [Dai et al., 2007]; whereas in hierarchical transfer the source tasks learn simpler problems, and solutions are combined upwards the target task, as in [Taylor et al., 2007]. On the other hand, semi-supervised learning has the same goal than supervised learning, i.e. to accurately classify instances, but only with a small subset of labeled examples, while the rest remains unlabeled. As the name suggests, these kind of tasks require, to some extent, unsupervised techniques, either for clustering or for dimension reduction. [Zhang and Rudnicky, 2006] proposed a method for retraing based on performance-drive selection of labeled examples. Generative models such as GMMs are also popular and efficient approaches, [Gao et al., 2017, Paul and Pal, 2016]; but discriminative models are also used, for instance in [Li and Zhou, 2015].

In this chapter we face a classification problem from a purely ML perspective, and combine both transfer and semi-supervised learning with CV and DL for classifying a large data set containing images of dumpsters with the purpose of having a correct census of their number and type. Such a task has the following difficulties. First of all, although dumpsters usually exhibit uniform colors and simple geometric shapes, there are many sources of variability in their appearance. Shapes range from igloos to large bins or a small pipe in the sidewalk if the dumpster is under-grounded. Since they are a disposal facility, in permanent contact with garbage, together with the lack of care by cleaning service operators, vandalism, open-air exposure, sometimes 24×7, they all provoke a fast degradation. Secondly, to be efficient in the acquisition of images, the camera may be mounted on a car driven through the area of interest taking georeferenced pictures or even recording in video. Therefore occlusions by pedestrians and vehicles, back-light, reflections and shades over the dumpsters are sources of unpredictable visual noise.

Waste collection, separation and management is generally carried out in two distinct but non-exclusive ways. In the first one every citizen keeps the garbage at home for a period that depends on the municipal regulations. In the other, a large number of dumpsters is deployed throughout the city, and people throw the garbage in them at some restricted hours. The latter is usually more broadly implemented in communities with policies that favor social spending and public investment. Thus, dumpsters become a public asset that requires inventory control and maintenance. For this reason, this topic is a relevant issue to the city hall and impacts on different areas and budgets, including health or city care, and may create a perception of insecurity and mis-attention to the neighborhood by local rulers and, in the long run, dissatisfaction of taxpayers. Despite of its economic interest, whereas there are a number of papers about waste classification by means of computer vision, such as [Brinez et al., 2015, Sudha et al., 2016, Wang et al., 2016], there are only few a focused on detecting and assessing their conservation state. For instance, [Mujumdar et al., 2013] does so with computer vision techniques prior to deep learning; but usually some extra aid with wireless sensors networks is proposed, as in [Hong et al., 2014] or [Idwan et al., 2016].

Our proposal uses the Google Inception-v3, a CNN pretrained with 1,500,000 images and 1000 different classes, [Szegedy et al., 2016]. Thus, a rich set of many different visual features, along with their spatial relationship, is encoded in the parameters of the net. Then, we do transfer learning by removing the output layer and substituting it with an extra hidden layer plus a last layer with as many output neurons as types of dumpsters we want to classify. For training the resulting convolutional network we use only 2.5% labeled images of a fully labeled data set of images provided by Ecoembes[1], a non-profit organization whose self-proclaimed mission consist on promoting the sustainable development trough recycling and the eco-design of packaging in Spain. We then propose a semi-supervised method for classifying the full data set based on retraining and intelligent selection of samples.

5.1.1 CNN based approach for dumpsters classification

The goal of this work is to obtain a high-level accuracy in a CNN trained for dumpster classification with a very small set of labeled samples. To this end, Ecoembes provided the EcoDID-2017 data set with a total 27,624 images of dumpsters, showing different conservation states, shapes and colors along with different lighting conditions and point of view of the camera. The resolution of the images is 299 × 299 pixels, and are given in seven folders according to the following features:

(a) C_1: For light recyclable packs, 2400 liters; with 1,730 images.

(b) C_2: For light recyclable packs, 3200 liters and circular hole of $\oslash$ 30 cm.; with 2,986 images.

(c) C_3: For light recyclable packs, 3200 liters and rectangular hole of 40×25 cm.; with 1,393 images.

(d) C_4: For the rest of light recyclable packs; with 3,576 images.

(e) C_5: For non-recyclable waste, on the street; with 10,965 images.

[1] www.ecoembes.com

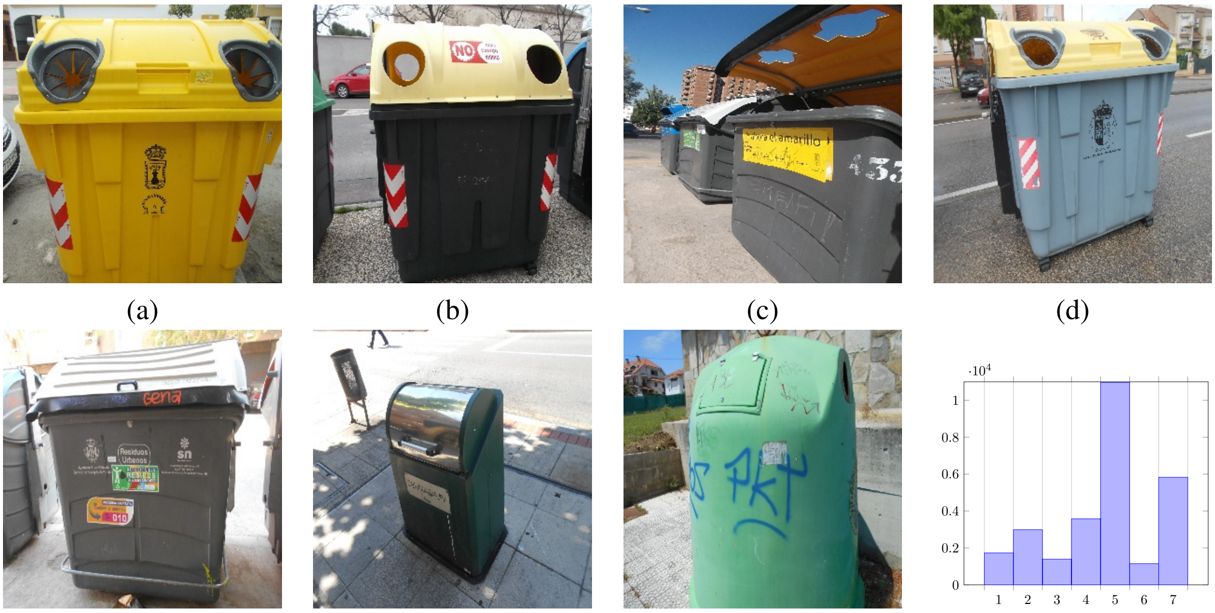


Figure 5.1: (a)-(g) A sample of each dumpster class, and (h) the distribution of the whole data set provided.

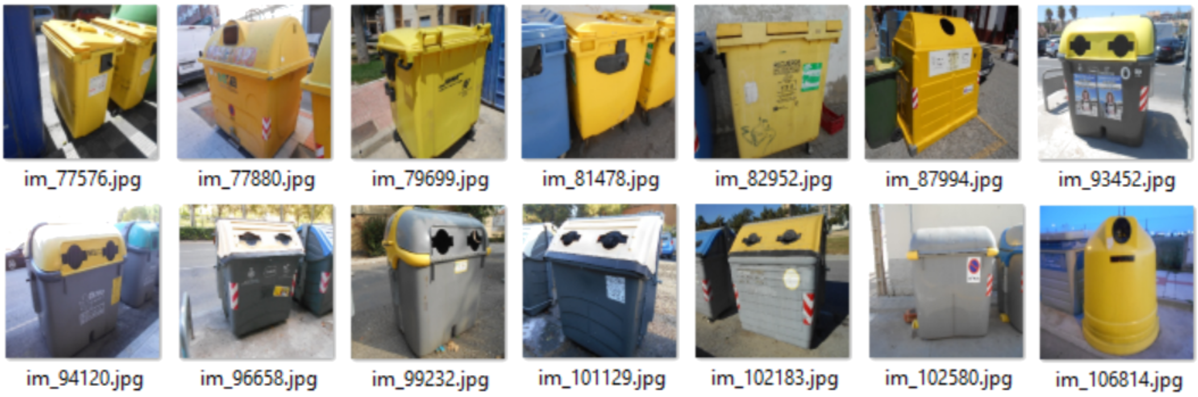


Figure 5.2: Sample of the diversity of dumpsters in Class C_4.

(f) C_6: For non-recyclable waste, undergrounded; with 1,147 images.

(g) C_7: For glass; with 5,827 images.

Figure 5.1(a)-(g) shows a sample from each folder together with the distribution of the whole data set according to each one of them, in panel (h). As a result, each folder can be considered as the label for all the images that are contained in it, so the given data set serves as ground-truth to validate the proposed methods.

Thus, we begin assuming that the whole data set is unlabeled, and we aim at correctly classifing it with a minimal manual labeling effort. We remark that this is a challenging problem because the difference between classes is hard to asses in an image. Specifically, between classes C_1 to C_3 the difference in volume is a matter of a few of centimeters in every dimension, (134cm)^3 vs. (147cm)^3, which is hardly noticeable in a picture; whereas class C_4 is assumed to contain dumpsters with features different from classes C_1 to C_3, which accounts for a large diversity of images. A little sample of 14 thumbnails is given in Figure 5.2.

Since there is a ground-truth available, we first use transfer learning for estimating the best performance that can be attained with a state of the art CNN. Then, only a minimal subset of labeled images is used, so the problem requires a semi-supervised solution. We present three approaches of it, one of them serves as baseline to be outranked by the other two.

Convolutional Networks

We have explained in a previous chapter 2.2.4 that CNNs are a special type of neural networks specially well suited for 2 dimensional inputs, i.e., images. We recall that CNNs can be seen as FCNNs with *infinitely strong priors* that fit very well the nature and patterns found in images, which are, smooth regions with sharp edges that are "locally" (non-local but in a close neighborhood) dependent. While the basic pattern that are found in natural images are likely to be similar among of the types of images one can think of, the high level features are obtained as a combination of the previous ones. This motivates the use of the so-called Transfer Learning technique that we briefly described in next section.

5.1.2 Transfer Learning

Transfer learning consist of re-training a network that was previously trained for a similar task to the one we are interested in. Such a network is then slightly modified, substituting the output layer with a new hidden layer and a new output layer. By doing so, the resulting network is adapted to the new classes, i.e., the output is a vector whose length fits the number of classes that our problem imposes. The final FCNN placed on top of the CNN, is trained then with a reduced amount of images from the given data set.

Google Inception-v3 is a pretrained CNN obtained by learning from scratch the whole ImageNet data set, a more than one million images database [Deng et al., 2012], and 1000 different categories. Thus, the extracted features (mainly the basics) are discriminant enough to perform a good generalization. Taking advantage of this fact, we leave all the weights of the CNN untouched and retrain only the last layer which now has 7 neurons, one for each of the new 7 classes, and train with our own data set of images. The framework used for such work is based on TensorFlow, [Abadi et al., 2015], an open source library developed by Google for DL and ML purposes, that provides AD (back-propagation algorithm).

We use this approach with 80% of the Ecomembes data set for training and validation, and the remaining 20% for testing. The resulting classifier is taken as the top-performance method under the assumption that when using much less images for training the accuracy will be, at most, as high as in this case.

5.2 Proposed Methods: Semi-Supervised Learning

We present two methods for classifying large image data sets only with a tiny subset of labeled images. Hence, it is a semi-supervised problem that we tackle with a three-round retraining. Specifically, in the first round 30 images from each folder are chosen. Thus we end up with a first data set of 210 labeled images, the label being the folder. We then train the modified Inception-v3 CNN only with those 210 images and, with the resulting classifier, compute the probability of each label for every single image in the whole data set.

At this point, second and third rounds have three variations: (i) Baseline method, (ii) Half-Worst method and (iii) Gaussian Mixture Model (GMM) method.

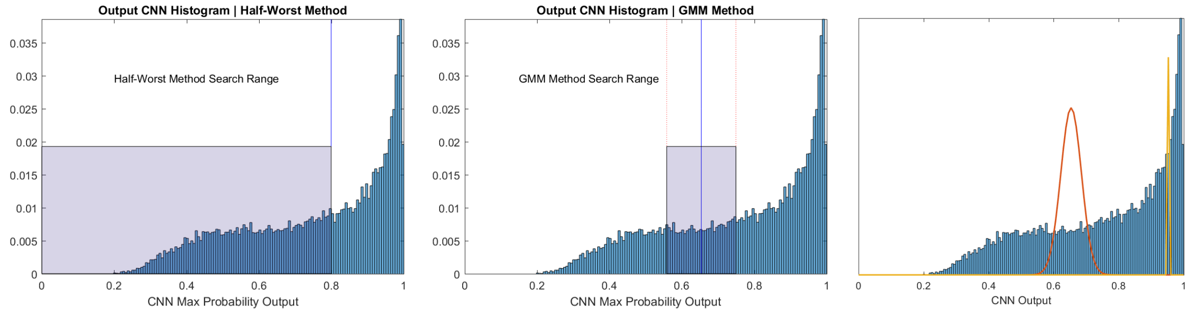


Figure 5.3: The three panels show the distribution of the confidence on the labeling after the first round, i.e. training with a manual selection 30 images × 7 folders. The left and central panels also show the search region S_{HW} (left) and S_{GM} (central). The right panel depicts the two gaussians of the GMM that models the distribution.

5.2.1 Baseline method

It simply consists of increasing the number of labeled images taken from each folder, 30 more each round, and retrain. Thus we end up with a CNN classifier trained only with 630 images, a 2.28% of the given data set. Thus, the proposed methods below should get an accuracy that outranks the one attained by this one.

5.2.2 Half-Worst method

The CNN returns a probability distribution function across the 7 classes for each image of the data set. Hence, the label assigned to an image is the one with higher probability, as a measure of the confidence in the labeling, but it is not granted that all the images will be labeled with a high confidence. For instance, it could happen that one label got probability 0.16, and the remaining six got probability 0.14. Thus, once the first round is over, we can make the distribution of the confidence in the labels, that is the CNN output taken for labeling each image.

In our first approach we propose to split the dataset using as a criterion the median of the confidence distribution. The images with the worst CNN outcome constitute the 50% of the dataset from where we extract 210 new images. In other words, the confidence given for the current CNN over an image will determines if that image is a good candidate to be manually labeled and added to the training data set.

Let $p(x)$ be the confidence distribution, where $0 < x < 1$ is the confidence of the label assigned by the trained CNN. We define the search region $S_R = S_{HW}$ as the subset of images with $x < \texttt{median}(p(x))$. In other words, the search region contains the 'half-worst' confident images. The left panel on Figure 5.3 shows an example. Out of them, we randomly select 30 images from each folder and append them to the training set. Thus, the new training set contains 50% of highly confident labeled images, and 50% with some features that are not captured yet by the CNN. At this point we run the second retrain using 420 images. We repeat the whole process one round more, or until the maximum of allowed labeled images has been covered. The procedure is summarized in Algorithm (5.1).

5.2.3 GMM method

The second method that we propose consists of modelling the confidence distribution as a two Gaussian Mixture model (2-GMM). The assumption is that one of the gaussians models the images with low confidence, whereas the other models those with high confidence. For instance, Figure 5.3-right shows

Algorithm 5.1: HW-Method

```
* 7 unbalanced categories of images
* 30 randomly sampled images from each category ;
* $x$ is the confidence in the label assigned to an image
* $p(x)$ is the distribution of the confidence
Initialization: Create initial training and validation dataset
    1: Retrain CNN with current training set
    2: Classify the whole dataset with the CNN
    3: Create a subset $S_{HW}$ of the whole dataset with those images meeting the following condition:
           $x \leq \texttt{median}(p(x))$
    4: Randomly select 30 images for each category from $S_{HW}$ subset
    5: Increase training set merging the new labeled images with the previous ones
    6: Dropout the current CNN
    7: Repeat until the maximum number of manually labeled image is reached
```

the 2-GMM over the distribution of confidence once the first round is finished. Let $\mathcal{N}_L(\mu_L, \sigma_L)$ be the and $\mathcal{N}_H(\mu_H, \sigma_H)$ be both Gaussian distributions. Then the search region $S_R = S_{GM}$ for this method is defined as the set of all the images such that their confidence is in the interval $[\mu_L - 3\sigma_L, \mu_L + 3\sigma_L]$. As shown in Figure 5.3, compared with S_{HW}, which is 50% of the whole data set, S_{GM} is much smaller, usually no more than 10%. The reason is that appending images with high confidence to the training set may produce over-fitting, whereas appending images with low confidence may entail the risk of learning out-layers or even misclassified images.

Finally, as in the Half-Worst method, we randomly select 30 new images from S_{GM} that are appended to the training set and we repeat the learning process, which completes the second round. The third round is a whole new iteration of the process. The procedure is summarized in Algorithm (5.2).

Algorithm 5.2: GMM-Method

```
* 7 unbalanced categories of images
* 30 randomly sampled images from each category ;
* $x$ is the confidence in the label assigned to an image
* $p(x)$ is the distribution of the confidence
Initialization: Create initial training and validation dataset
    1: Retrain CNN with current training set
    2: Classify the whole dataset with the CNN
    3: Estimate the 2-GMM model of the confidence for every image in the data set
    4: Select the component with lowest mean: $\mathcal{N}_L(\mu_L, \sigma_L)$
    5: Create search region $S_{GM}$ with those images meeting the following condition:
           $x \in [\mu_L - 3\sigma_L, \mu_L + 3\sigma_L]$
    6: Randomly select 30 images for each category from $S_{GM}$
    7: Increase training set merging new labeled images with the previous ones
    8: Dropout the current CNN
    9: Repeat until the maximum number of manually labeled image is reached
```

5.3 Experimental results and discussions

As we have explained above, there are four experiments to be carried out, namely: to estimate both a lower and an upper bound of the accuracy that we should expect from the methods proposed, along with these two methods. Besides, in order to measure the dispersion of the outcomes, we have repeated every experiment 10 times.

We begin estimating the lower bound, by running three rounds of the baseline method, and keeping the outcome of each one. As presented above, we take 30 images from each folder in the first round, summing up to a training set of 210 labeled images for the first round. In each coming round we increase the training set in 30 images per folder and retrain the CNN again. Notice that the baseline method would only require one round with 630 images if we are going to execute three rounds of the Half-Worst and GMM methods. The only purpose of the first two rounds here, i.e. training with 210 and 420 labeled images, it is to compare the evolution of accuracy as the training set increases.

Thus, after the third round, baseline shows up a 85.28% of accuracy with a standard deviation of 1.25%. Hence, despite there is already 15% accuracy to improve, it is unrealistic to expect a 100% of accuracy in a dataset so challenging.

Testing all the possible combinations requires to train and test $\binom{26{,}994}{630}$ combinations, which is clearly unfeasible. To avoid that problem, we will use as upper bound the accuracy of a CNN trained with 80% of the whole dataset, keeping 10% for validation, and 10% for test. It turns out this maximum performance is about 93%. In other words, the feasible range for improving is $[85\%, 93\%]$, leaving a width of 8%.

Having both the lower and the upper bounds of the accuracy we can achieve, we are ready to test our both proposed methods. Results for Half-Worst and GMM method are shown in Figure 5.4. Compared to the baseline, at the end of the 10 executions the averaged accuracy of the Half-Worst method is 2% greater whereas the improvement with the 2-GMM method is a 3%. However, in relative terms, i.e. compared with the 8% that we have estimated would be possible if we had 80% of the data set labeled, the Half-Worst method attains a 25% of it, whereas the 2-GMM rises up to 37.5%.

As a secondary outcome, it is also remarkable that the dispersion on the accuracy of the CNN decreases when using both methods compared with the baseline, which accounts for a more stable performance independently from the initial dataset set. Figure 5.4 shows the individual outcome of each one of the 10 executions in the left panel. It is clear that the band around the Half-Worst method is thinner than the one around the base line and a little wider than the one around the 2-GMM method. On the other hand, the right panel on Figure 5.4 shows how the dispersion of the accuracy evolves in the second and third rounds.

For the sake of completeness, we also show other performance measures in the four panels on Figure 5.5. In this case we consider our multi-classification problem as a binary classification of a each given class against the six others. We firstly represent the variation of the True Positive Ratio (TPR) and the False Positive Ratio (FPR) in terms of the decision threshold; i.e. the minimum value required to label the image positively. Panels (a) and (b) on Figure 5.5 depict the curves for every class and every method. Since, for TPR the better the higher, and for FPR the better the lower, results show that 2-GMM is, in most of the cases, the best of them. We also show the ROC curve for every class in Figure 5.5c. Notice that the FPR axis is not scaled to $[0, 1]$ but to $[0, 0.05]$ in order to make the ROC curves visible. Hence all the binary classifiers have a very good performance, but the binary classifier for class 6 excels. The reason is that it corresponds to the undergrounded dumpsters, which are arguably the most different with respect to all the rest. On the other hand, the classifier for class 4, that collects all images of light recyclable pack that don't belong to class 1, 2 or 3, shows the worst performance due to its large variability. Figure 5.5d shows the averaged ROC curves for every method. All those common metrics definitions are summarized in Appendix 8.2.

Finally, we discuss the convenience of the proposed methods. Notice that, in every round, we append 30 images from each folder to the training set. Since the folders are the labels, these 630 images are necessarily considered to be part of a supervised process and therefore the whole problem is a semi-supervised task. However only the first selection is due to a human. There is neither a query function nor human-in-the-loop, so none Active Learning approach has been considered. If the data set was completely unlabeled, for instance images downloaded after a workday, the methods should either query a human at the end of each round to select 30 images of each class, or to randomly sample 210 new images, both obviously from the region search. When dealing with images, such as in this problem, the first one requires little effort, just a visual inspection of many thumbnails at the same time. Indeed, for this particular problem, the separation given in folders can be seen as a bias towards the human-in-the-loop solution. On the other hand, the second solution is more appropriate when the labeling effort can not be *parallelized*. For instance if, instead of images, there were audio files, it is not possible to hear more than one at the same time.

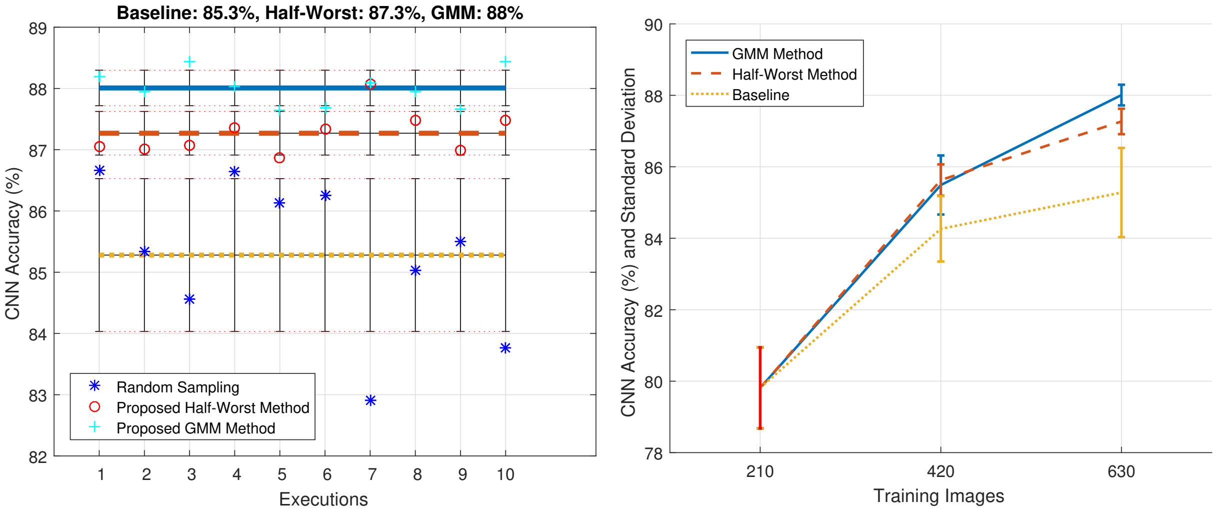
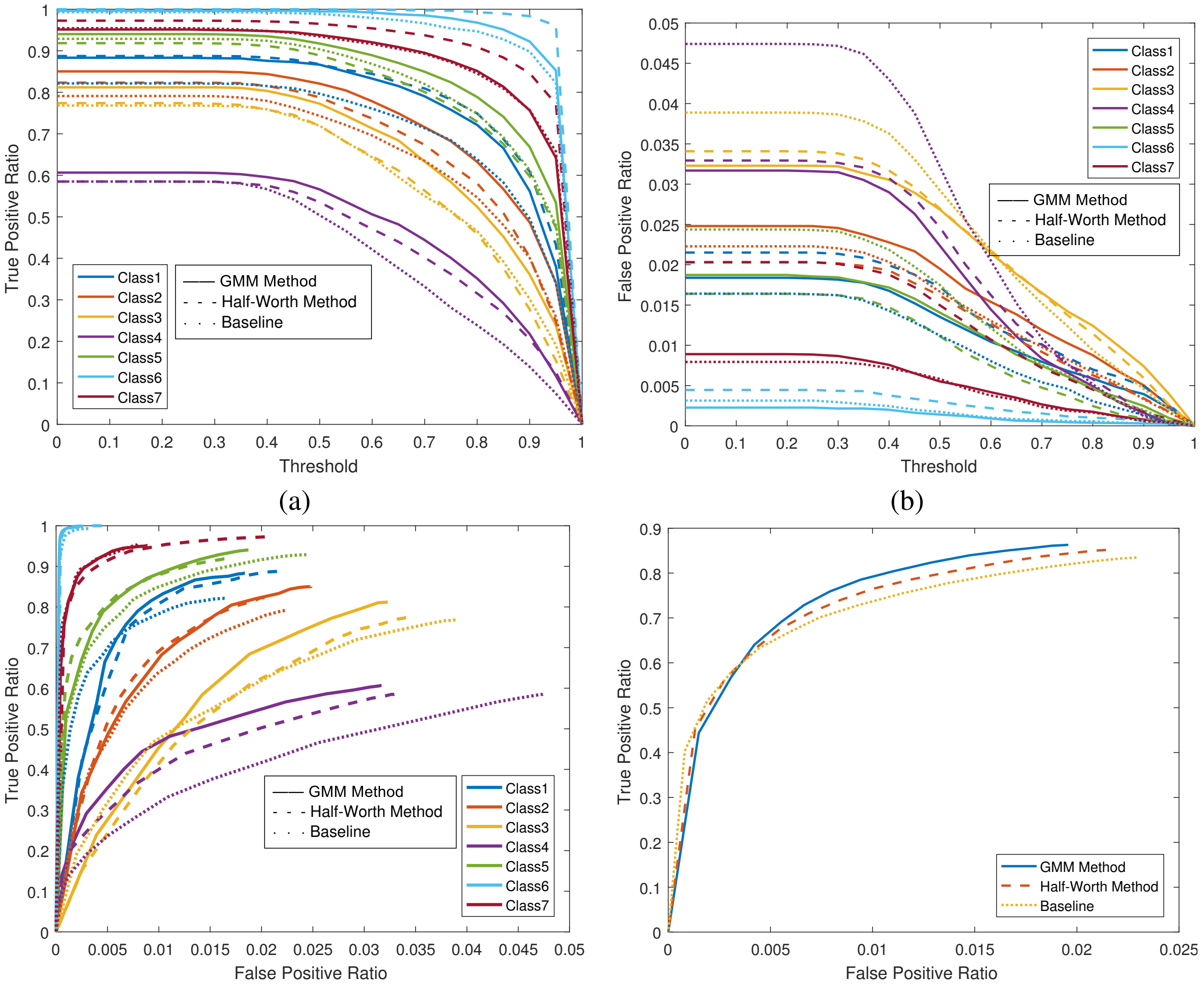


Figure 5.4: Experiment results. (Left) Accuracy at the end of the three rounds of every method (baseline, Half-Worst and 2-GMM) repeated 10 times. (Right) Mean and standard deviation of the accuracy as the number of training images increases at each round.



(a)

(b)

(c)

(d)

Figure 5.5: Other performance measures, split in the 7 different classes for panels (a) to (c). Specifically, (a) True Positive Ratio per Class (b) False Positive Ratio per Class (c) ROC per Class. In addition panel (d) shows the averaged ROC curve of every method.

Chapter 6

P3. BCN for 3D Human Pose Estimation

No.
José Saramago

This chapter is organized as follows. First, in Section 6.1, we revise common problems in Modern DL tools and issues that a deep learning practitioner may face, as well as the basic theoretical background on Bayesian Neural Network (BNN) and Capsnets. We also present the network architecture for the case study of human 3D pose estimation. In Section 6.2 we introduce the bayesian formulation of Capsule Networks: BCN which results in using dropout and a regularization term imposed in the capsule matrices. Finally, experiments, results and comparisons with the state of the art methods are shown in Section 6.3.

6.1 Introduction: BCNs for 3D Pose Estimation and Advanced DL

DNN have been the subject of a fast growing research effort mainly related to AI and CV applications. In less than a decade there are readily available numerous large datasets, computational power, open source libraries and a multitude of tutorials for and from the ML community of researchers, practitioners and academia. Many of the state of the art DNN architectures are off-the-shelf, pretrained with huge datasets, during many hours and with powerful computers, so that with minimal changes and much less computational effort can be adapted to a specific task and go on production quickly [Hao, 2019].

DNN as presented above are an archetype of discriminative, black-box model: lacking in transparency and poor at representing uncertainty, with the additional shortcoming of being impossible to interpret. For instance, consider the simple task of single-label classification with a DNN. Given a new instance, the DNN will return a probability over all the possible classes (if using Softmax as likelihood distribution, see 2.2), but we do not have a measure of how much uncertainty there is in every probability mass. In order to assess the quality of the estimators one can proceed in a frequentist way as in the *Bag of little Bootstraps* (BLB) [Kleiner et al., 2014]. But again, the quality of the estimator does not provide a reliable way to tell the confidence in their estimations.

On the other hand, in the bayesian formulation of NNs (chapter 3), weights, biases, number of neurons and number of hidden layers are all considered sources of uncertainty, and therefore modeled by probability distributions that can be assumed a priori and updated with data a posteriori. Classification and regression tasks are casted as the problem of estimating the distribution over the network outputs given the dataset and a model. Additionally, it also makes possible to estimate the distribution over the network outputs given the dataset but not necessarily conditioned on a particular model, and even over a set of models, given the data. BNNs were early introduced in [Denker and LeCun, 1990] within a proposal to compute the posterior of the parameters algorithmically via the Laplace approximation, and more broadly in [MacKay, 1992], with a quantitative and practical approach that tackles with architecture selection, choice of weight decay terms and uncertainty in network parameters and outputs. After some years of decreasing interest in NN, the arrival of DL has brought new results in BNN [Blundell et al., 2015, Hernández-Lobato and Adams, 2015], and has sparked the Deep Bayesian Network (DBN), [Wang and Yeung, 2016, Chien and Ku, 2016, Zhu and Zabaras, 2018].

Dealing with the interpretability of DNN models is much harder. CNNs attain excellent results in CV because images have strong local correlations which are exploited by the convolutional neurons. Within this scope it is possible to explain the contribution of each convolutional neuron as a kernel that enhances some visual features. But this good properties are not transferable to the general problem in which instances are described by an array of attributes arbitrarily ordered. A novel neural network architecture known as Capsule Networks (Capsnets), introduced in [Sabour et al., 2017, Hinton et al., 2018], transforms the feature space into a higher dimensional space in which each instance is represented by a set of vectors bounded to norm one. Each vector encodes the input, but having many of them makes also possible to encode the variability of the abstract *concept* or general *entity*. This variability may include pose, illumination, texture, etc. depending on the data set. Besides, the norm of every vector is interpreted as the probability of such a vector representing a given *concept*. The intuition is that having one of these vectors close to norm one may be enough to classify the instance because it is very distinctive, so the representation obtained with Capsnets hierarchizes the features in a some way.

6.1.1 Capsnet architecture for 3D Human Pose

We attempt to infer the 3D human pose from a single 2D RGB image; where the pose is given by the 3D coordinates of 17 joints in a human skeleton. It is easy to get 3D to 2D projections of a skeleton. Moreover, much development have been carried out to equip computer with real time methods for such a purpose, which constitutes the base of Video Games. However, the 3D pose reconstruction using 2D data is an ill-posed optimization problem that needs to be regularized. This can be do in a variational framework as in [Kolev et al., 2009]. Another approach is to cast it as a regression problem to be solved with a DNN that estimates every joint. Dealing in this way with such an ill-posed problem requires prior information. For instance in [Tome et al., 2017] a biological probabilistic model is proposed. The other option is to do Deep Regression Bayesian Networks, for instance[Nie et al., 2018].

In this chapter we solve the regression problem in a model-free NN, end-to-end, architecture that incorporates our proposed BCN formulation. The whole system attempts to minimize a multi-task objective to improve the accuracy of the main target (3D coordinates) avoiding the use of other techniques such as transfer-learning, using other data sets or taking into account the previous images if the image belongs to a video-sequence. The data set used is known as *Human 3.6*, described in detail in Section 6.3.

The architecture proposed consists of 3 modules, each being a NN of a different kind. The input image is firstly processed by a CNN with residual blocks. Then a BCN transforms the feature maps into capsules and finally three different Bayesian Fully Connected NNs produce the estimation of the 3D pose together with a reconstruction of the 2D pose and heat maps of each joint.

6.1.2 Preliminaries

We first introduce the notation used throughout this chapter as well as recap basic background on BNN and Capsule Networks.

Notations We mark with a *hat* those variables that are estimations obtained by means of a NN, e.g. $\hat{y}$, with a *star* those which are solution of an optimization problem, e.g. w^*, and with a *tilde* the samples from a probability distribution, e.g. $\tilde{x} \sim p(x)$. $\|X\|_F$ represents the Frobenius norm of a matrix X, and $\mathbb{I}_n$ is the identity matrix of size $n \times n$.

As usual, let $\mathbf{X}$ be a data set of N instances $\{x^{(i)}\}$, and let $\mathbf{Y}$ be a set of the corresponding labels, or a ground truth, $\{y^{(i)}\}$, for $i = 1 \ldots N$. We will also assume that the pairs $(x^{(i)}, y^{(i)})$ are i.i.d. Let $f : \mathbf{X} \to \mathbf{Y}$ be the function implemented by a given NN, and let $\hat{y} = f(x; \mathbf{W})$ be the output (prediction) of such a NN for a given input x and a given configuration of the weights and biases of the NN, jointly represented by $\mathbf{W}$ and referred to simply as the *weights*.

Bayesian Neural Networks

The bayesian approach is two-fold. On one hand it is the maximum a posteriori estimation of the NN weights. On the other hand, it is the bayesian inference of the target output according to a prediction distribution. We next present the basic background of both together with the issues risen and the state of the art solutions.

MAP estimation of the weights Following the presented Bayesian Inference in chapter 3, we set $p(\mathbf{W})$ as *prior* probability distribution over the weights of the NN, and then look for $\mathbf{W}^*$, i.e., the MAP weights given the data $\mathbf{X}$ and $\mathbf{Y}$. According to the Bayes theorem and taking into account that the input is independent of the weights, the problem is formalized as

$$\mathbf{W}^* = \arg\max_{\mathbf{W}}\{p(\mathbf{W}|\mathbf{X}, \mathbf{Y})\} = \arg\max_{\mathbf{W}}\{p(\mathbf{W})p(\mathbf{Y}|\mathbf{X}, \mathbf{W})\}, \tag{6.1}$$

where $p(\mathbf{Y}|\mathbf{X}, \mathbf{W})$ is the *likelihood* and $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ is the *posterior*. The proposal is to assume that both the likelihood and the prior are Normal distributions. However, and following [Kendall and Gal, 2017], we will also consider as objective parameters, those of both normal distribution rather that fixing their values (which is in fact to consider a bayesian inference approach vs machine learning task). Specifically, the prior over the weights is a multivariate normal centred in 0 with covariance matrix $\Sigma = \sigma_w \mathbb{I}$. For the sake of clarity, we use σ_w.

$$p(\mathbf{W}) = \frac{1}{\sqrt{2\pi}\sigma_w} \exp\left(-\frac{\|\mathbf{W}\|_2^2}{2\sigma_w^2}\right). \tag{6.2}$$

The likelihood of a pair $(x^{(i)}, y^{(i)})$ is centered in the output of the NN $f(x^{(i)}; \mathbf{W})$ with standard deviation σ,

$$p(y^{(i)}|x^{(i)}, \mathbf{W}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|y^{(i)} - f(x^{(i)}; \mathbf{W})\|_2^2}{2\sigma^2}\right). \tag{6.3}$$

The standard deviations introduced in the last two expressions are associated to different types of uncertainty in bayesian modeling. According to [Kendall and Gal, 2017], σ_w is the Epistemic uncertainty of the model, and σ is the Aleatoric uncertainty associated to the difficulty on the particular task. If the aleatoric uncertainty is the same for all the inputs, it is said to be homoscedastic. On the contrary, if it depends on the input data it is known as heteroscedastic. Here we shall assume homoscedasticity.

Taking into account that $p(\mathbf{Y}|\mathbf{X}, \mathbf{W}) = \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}, \mathbf{W})$, introducing (6.2) and (6.3) into (6.1), and taking logarithms we have

$$\{\mathbf{W}^*, \sigma^*, \sigma_w^*\} = \underset{\mathbf{W},\sigma,\sigma_w}{\arg\min}\{\mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{H}} + \mathcal{L}_{\text{T}} + \mathcal{L}_{\text{E}}\}, \tag{6.4}$$

$$\text{where } \mathcal{L}_{\text{MSE}} = \frac{1}{2\sigma^2}\frac{1}{N}\sum_{i=1}^{N}\left(\|y^{(i)} - f(x^{(i)};\ \mathbf{W})\|_2^2\right), \tag{6.5}$$

$$\mathcal{L}_{\text{H}} = \log\sigma, \tag{6.6}$$

$$\mathcal{L}_{\text{T}} = \frac{1}{2\sigma_w^2}\|\mathbf{W}\|_2^2, \tag{6.7}$$

$$\mathcal{L}_{\text{E}} = \log\sigma_w. \tag{6.8}$$

In the functional above, (6.5) is the Mean Squared Error (MSE) or L_2 loss function and (6.7) is the Tikhonov regularization over the weights which in NN is usually introduced in the minimization step referred to as *weight decay*. Additionally, the normality assumptions have brought the homoscedastic term (6.6) and the epistemic term (6.8). Otherwise these two terms can be renamed into a constant λ, recovering the loss function $\mathcal{L} = \text{MSE} + \lambda\|\mathbf{W}\|_2^2$ for non bayesian NN.

Notice that $\lim_{\sigma\to 0}\log\sigma = \lim_{\sigma_w\to 0}\log\sigma_w = -\infty$; so both (6.6) and (6.8) may dominate the minimization (6.4). Pushing σ towards 0 means attaining 100% accuracy, but σ_w towards 0 carries $\|\mathbf{W}\|_2^2$ to 0 and viceversa. We deal with this issue in the Practical BNN section below.

Bayesian inference Another bayesian approach to is through inference, predicting the target y for a new given input x according to the aforementioned prediction distribution

$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int p(y|x, \mathbf{W})p(\mathbf{W}|\mathbf{X}, \mathbf{Y})d\mathbf{W}. \tag{6.9}$$

We have shown that, since integrating over all possible weight configurations is computationally intractable, the posterior is approximated by a family of probability distributions $q_\theta(\mathbf{W})$. The parameter θ is found looking for the one that minimizes the KL divergence with the actual posterior distribution, which is known as the variational approach, that leads to maximizing the *Evidence Lower Bound* (ELBO) and an optimization problem expressed as follows (see chapter 3 for a detailed derivation):

$$\begin{aligned}\theta^* &= \underset{\theta}{\arg\min}\left\{\text{KL}\big(q_\theta(\mathbf{W})\big)\|p(\mathbf{W}|\mathbf{X}, \mathbf{Y})\right\} = \underset{\theta}{\arg\max}\{\text{ELBO}\} \\ &= \underset{\theta}{\arg\min}\left\{-\mathbb{E}_{q_\theta}\big(\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W})\big) + \text{KL}\big(q_\theta(\mathbf{W})\|p(\mathbf{W})\big)\right\}.\end{aligned} \tag{6.10}$$

In fact, both terms in equation (6.10) are the negative log-likelihood and the imposed or assumed prior over the weights $\mathbf{W}$, recovering both expressions from the MAP approach. This is tractable via Monte Carlo integration. Let $\{\widetilde{\mathbf{W}}_t\}_{t=1\dots T} \sim q_\theta(\mathbf{W})$ be a set of T samples drawn from the distribution q_θ. Then,

$$-\mathbb{E}_{q_\theta}\big(\log p(\mathbf{Y}|\mathbf{X},\mathbf{W})\big) \approx -\frac{1}{T}\sum_{t=1}^{T}\log p\big(\mathbf{Y}|\mathbf{X},\widetilde{\mathbf{W}}_t\big), \tag{6.11}$$

$$\mathrm{KL}\big(q_\theta(\mathbf{W})\|p(\mathbf{W})\big) \approx \frac{1}{T}\sum_{t=1}^{T}\log \widetilde{\mathbf{W}}_t - \frac{1}{T}\sum_{t=1}^{T}\log p(\widetilde{\mathbf{W}}_t). \tag{6.12}$$

Once θ^* is estimated, $\mathbb{E}_{q_\theta^*}\big(p(y|x,\mathbf{W})\big)$ is also approximated via Monte Carlo.

Practical BNN The choice of the prior is a key of the Bayesian approach. We have shown that the assumption of normality leads to terms in the loss function that may difficult the training, whereas choosing another prior should also take into account to be easy to sample from. The solution to the first issue, proposed in [Gal and Ghahramani, 2016b], is to remove (6.7) and (6.8) from the functional in (6.4) and use dropout instead. Dropout was firstly introduced in [Srivastava et al., 2014] as a way of preventing overfit in DNN. Later in [Kendall and Gal, 2017], it was explained as if every weight in a NN was multiplied by a binary random variable z with probability π, i.e. $z \sim \mathrm{Ber}(\pi)$. Hence, doing dropout in every layer is a practical way to get samples $\widetilde{\mathbf{W}}_t \sim p(\mathbf{W})$, not necessarily normally distributed. By doing so, and defining $s = \log \sigma^2$, (6.4) is rewritten as.

$$\{\mathbf{W}^*, s^*\} = \underset{\mathbf{W},s}{\arg\min}\left\{e^{-s}\mathcal{L}_{\text{MSE}} + s\right\}. \tag{6.13}$$

Moreover, if we consider $\mathcal{L}_{\text{MSE}}$ as the loss of a given task, we can extend (6.13) for multiple tasks carried out by the same NN as follows:

$$\{\mathbf{W}^*, s_1^* \dots s_\tau^*\} = \underset{\mathbf{W},s_1\dots s_\tau}{\arg\min}\left\{e^{-s_1}\mathcal{L}_1 + s_1 + \cdots e^{-s_\tau}\mathcal{L}_\tau + s_\tau\right\}. \tag{6.14}$$

In summary, dropout in every layer is an easy way towards BNN in practice and provides two extra adventages: it allows to minimize the uncertainty in the model by using (6.13); and it also allows to self-balance multiple losses in the same NN because $s_1, \dots, s_\tau$ are optimized at the same time in (6.14). Notice that (6.14) requires the loss functions to be MSE. Kendall et al. show in [Kendall et al., 2017] that it is also possible to use softmax as the activation of the last layer and then cross-entropy for the loss. Thus, BNN can be applied both in regression and classification tasks.

Capsule Networks

The so-called Capsule Network architecture has been released recently [Sabour et al., 2017], initially with the purpose of image recognition, introducing some variations with respect to the CNN. The Capsule Network architecture can be divided in 5 steps: Input, Encapsulation, Inverse graphics, Routing and Output, depicted in Figure 6.1. The feed-forward step during training is detailed in Algorithm 6.1. Notice that the coefficients $c_{k,j}$ are updated in this step, whereas the matrices $A_{k,j}$ are updated in the back-propagation.

Algorithm 6.1: Capsule Network

```
Hyperparameters: $J$, $K$ and $S'$ are positive integers arbitrarily chosen.
Preconditions: The reminder of $M \cdot N \cdot D$ divided by $J$ must be 0.
Postconditions: $0 < \|\phi_k\|_2 \leq 1$ for $k = 1 \dots K$.
Def: $\texttt{Reshape}_{J\times S} : \mathbb{R}^{M\times N\times D} \to \mathbb{R}^{J\times S}$, with $J = MND/S$.
Def: $\texttt{squash}(\mathbf{u}) = \left(\|\mathbf{u}\|_2^2 \cdot \mathbf{u}\right) / \left(1 + \|\mathbf{u}\|_2^2 \cdot \|\mathbf{u}\|_2\right)$
Steps:
    1. Input:               $\psi \in \mathbb{R}^{M\times N\times D}$.
    2. Encapsulation:       $\mathbf{U} = \{\mathbf{u}_j\}_{j=1\dots J} = \texttt{Squash}\left(\texttt{Reshape}_{J\times S}(\psi)\right)$,
                              with $\mathbf{u}_j \in \mathbb{R}^S$ and $\mathbf{U} \in \mathbb{R}^{J\times S}$.
    3. Inverse graphics:    $\mathbf{v}_{k,j} = A_{k,j} \cdot \mathbf{u}_j$,   with $\mathbf{v}_{k,j} \in \mathbb{R}^{S'}$,
                              $A_{k,j} \in \mathbb{R}^{S'\times S}$, and $\mathbf{V}_k = \{\mathbf{v}_{k,j}\}_{k=1\dots K} \in \mathbb{R}^{K\times S'}$.
    4. Routing:             Compute coefficients $c_{k,j}$ with the
                              Routing by agreement algorithm, as in [Sabour et al., 2017],
                              s.t. $\sum_{k=1}^{K} c_{k,j} = 1$; $c_{k,j} \in \mathbb{R}$.
    5. Output:              $\phi_k = \texttt{Squash}\left(\sum_{j=1}^{J} c_{k,j}\mathbf{v}_{k,j}\right)$
Return: $\Phi = \{\phi_k\}_{k=1\dots K}$, where $\phi_k \in \mathbb{R}^{S'}$ and $\Phi \in \mathbb{R}^{K\times S'}$
```

6.2 Proposed Methods: Bayesian Capsule Networks

Experience tells us that, in many cases, when some ground-breaking technique shows up in the field of DL, it is likely to relate it to a sort of regularization or the introduction of a prior. This may be not the case in order fields as Variational Methods, where those regularizing terms are, in some way, the bread and the butter of such techniques when applied to CV and IP. Keeping this in mind, the following proposal makes use of: (1) a novel NN that imposes a prior structure regularization (as a CNN does, see 2.2.5), (2) a regularization term over the matrices of the mentioned network and (3) a Bayesian Inference Approximation approach through Variational Inference using dropout.

We start introducing a Bayesian formulation of Capsnets. To this end, we propose to introduce prior knowledge as restrictions over the matrices $A_{k,j}$ related to their geometric interpretation. In [Sabour et al., 2017] authors motivate these matrices as doing the inverse of computer graphics, in which a 3D scene is shown in a 2D screen, thus reducing the dimensionality. In Figure 6.2, a given 2D projection (image) of an object (cube) is retro-projected to infer the original object. Thus, this retro-projected inferred object (prediction) is projected again to check if the original datum is recovered.

Here, on the contrary, every capsule $\mathbf{u}_j \in \mathbb{R}^S$ is mapped K times into a higher dimensional space by means of linear transformations $A_{k,j} \in \mathbb{R}^{S'\times S}$, to obtain $\mathbf{v}_{k,j} = A_{k,j} \cdot \mathbf{u}_j$, with $\mathbf{v}_{k,j} \in \mathbb{R}^{S'}$ and $S' > S$. Following with the intuition behind Inverse graphics, every $\mathbf{v}_{k,j}$ is an estimation in a higher dimensional space of its projection $\mathbf{u}_j$. Hence, it is reasonable to expect recovering the projection by means of an inverse transformation $B_{k,j} \in \mathbb{R}^{S\times S'}$, such that

$$\hat{\mathbf{u}}_j = B_{k,j} \cdot \mathbf{v}_{k,j}, \quad \text{subject to} \quad B_{k,j}A_{k,j} = \mathbb{I}_S.$$

Clearly, the first candidate for $B_{k,j}$ is $A_{k,j}^{(-1)}$, the Moore-Penrose pseudo-inverse of $A_{k,j}$. However, this pseudo-inverse has a closed form when $A_{k,j}$ has full rank, which is a very weak constraint. Instead we

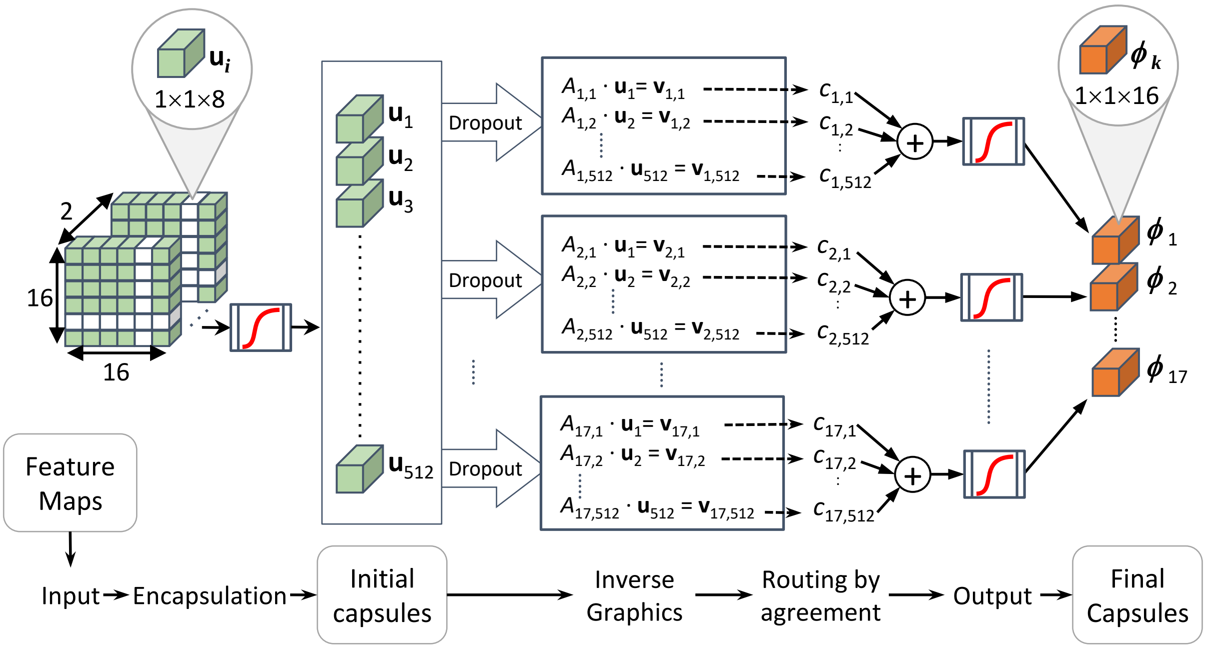


Figure 6.1: CapsNet module. Capsules are created reshaping the feature maps incoming from a previous NN. Then each one is mapped K times into a higher dimensional space. The results are combined and squashed into the outgoing capsules. The block with the red line is the squash function. In this Figure $M = N = D = 16$, $J = 512$, $S = 8$, $S' = 16$ and $K = 17$. Dropout is proposed here as part of the Bayesian formulation.

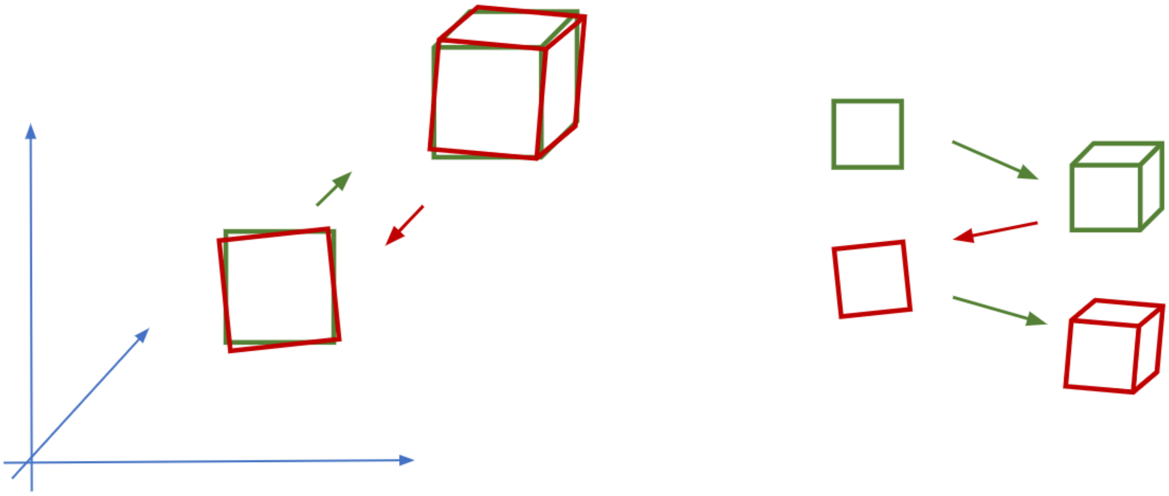


Figure 6.2: Left: 2D - 3D analogy. Right: retro-projection and inverted projection pipeline.

propose to learn the matrix $B_{k,j}$ at the same time than $A_{k,j}$. This is equivalent to relax the constraint on the invertivility of the retro-projection so $\hat{\mathbf{u}}_j = B_{k,j} A_{k,j} \mathbf{u}_j \approx \mathbf{u}_j$.

Lemma *Let A and B two matrices such that B is left-compatible with A. If $\|BA - \mathbb{I}\|_F^2 \approx 0$ then B is a consistent left pseudo-inverse matrix of A in the sense of the Frobenius norm.*

Proof Consider the vector $\mathbf{u}$, its projection $\mathbf{v} = A\mathbf{u}$ due to the linear transformation A and its estimated retro-projection $\hat{\mathbf{u}} = B\mathbf{v}$ due to the linear transformation B. Assuming that $\|BA - \mathbb{I}\|_F^2 \approx 0$, we have that

$$0 \leq \|\hat{\mathbf{u}} - \mathbf{u}\|_F^2 = \|BA\mathbf{u} - \mathbf{u}\|_F^2 \leq \|(BA - \mathbb{I})\|_F^2 \cdot \|\mathbf{u}\|_F^2 \approx 0.$$

Let $\hat{\mathbf{v}} = A\hat{\mathbf{u}}$ be the projection of the estimated retro-projection $\hat{\mathbf{u}}$. Hence, $0 \leq \|\hat{\mathbf{v}} - \mathbf{v}\|_F^2 = \|A\hat{\mathbf{u}} - A\mathbf{u}\|_F^2 \leq \|A\|_F^2 \cdot \|\hat{\mathbf{u}} - \mathbf{u}\|_F^2$. Therefore, if $\|\hat{\mathbf{u}} - \mathbf{u}\|_F^2 \approx 0$, then $\|\hat{\mathbf{v}} - \mathbf{v}\|_F^2 \approx 0$ too. □

Thus we are allowing a certain amount of uncertainty in the only step of the Capsnet that is learned during back-propagation. Moreover, we can control it by imposing the assuming a prior jointly over $A_{k,j}$ and $B_{k,j}$,

$$p(A_{k,j}, B_{k,j}) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp\left(-\frac{\|B_{k,j}A_{k,j} - \mathbb{I}_S\|_F^2}{2\sigma_b^2}\right). \tag{6.15}$$

We then can incorporate this prior to (6.1), assume the same likelihood than in (6.3), and assume that matrices $A_{k,j}$, $B_{k,j}$ are independent from parameters $\mathbf{W}$, to obtain the following optimization problem, similar to (6.13),

$$\{\mathbf{W}^*, s^*, s_b^*\} = \underset{\mathbf{W},s,s_b}{\arg\min} \left\{e^{-s}\mathcal{L}_{\text{MSE}} + s + e^{-s_b}\mathcal{L}_{\text{b}} + s_b\right\}, \tag{6.16}$$

$$\text{where } \mathcal{L}_{\text{b}} = \frac{1}{JK}\sum_{j=1}^{J}\sum_{k=1}^{K} \|B_{k,j}A_{k,j} - \mathbb{I}_S\|_F^2 \tag{6.17}$$

and $s_b = \log \sigma_b$. Recall that, in order to be bayesian, the above expression imposes doing dropout in all the layers of the full NN. With regards to a Capsnet module within such a NN, this is only possible after the Encapsulation and before the Inverse Graphics. Thus we propose a dropout that randomly sets to zero some capsule components before going into the Inverse graphics step. With this dropout technique and regularization term we achieve approximate invertibility along capsule layers. In other words, following the analogy depicted in Figure 6.2, the result of such a bayesian regularized capsule is the minimization of the projection and retro-projection errors, as depicted in Figure 6.3 below.

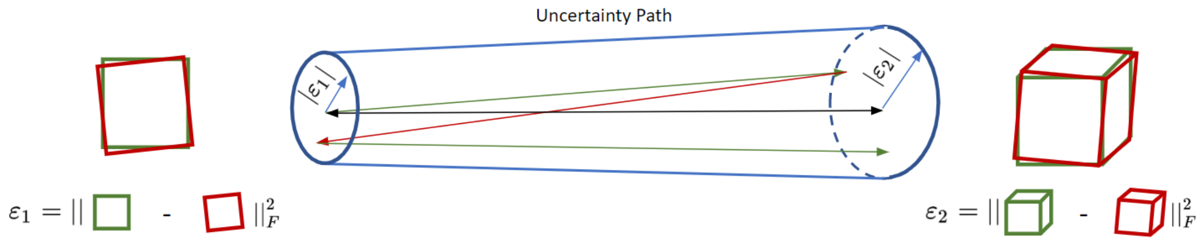


Figure 6.3: Error bounds on projection and retro-projection steps.

6.2.1 CNN with Residual blocks

The input image is first processed by a CNN consisting of 4 convolutional layers with residual units [He et al., 2016], kernels of size 9 × 9, *stride*= 2 and *padding* "valid", followed by a ReLU activation function. The combination of stride and padding produces a reduction of the input size. Besides, the layers produce 32, 64, 128 and 16 feature maps respectively. Thus, with this configuration the output tensor of this module has size 16 × 16 × 16. The residual units were proposed in [He et al., 2016] for allowing features to skip convolutional layers, thus acting as an identity function, which is known to be hard to "mimic" by a convolutional network. In this module Dropout is not used. The reason is, again, that

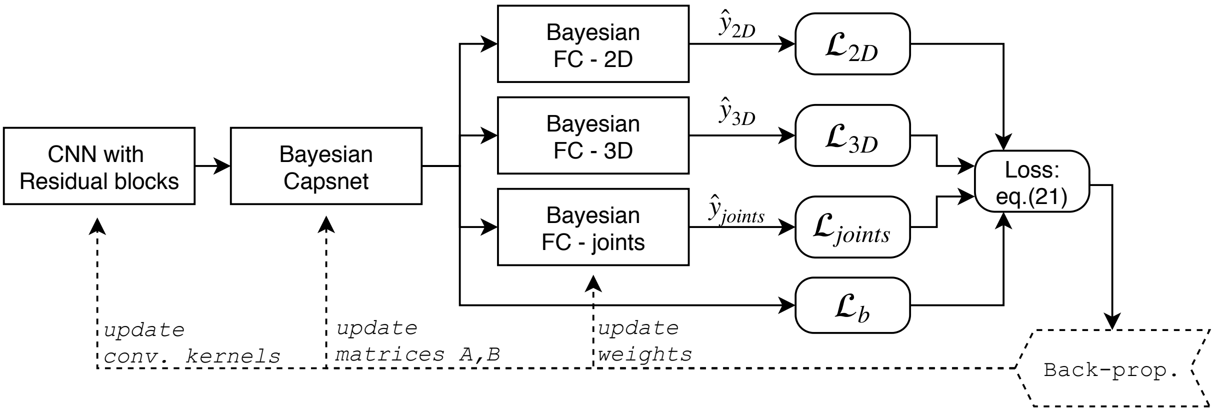


Figure 6.4: Global structure of the proposed CapsNet-based Bayesian Neural Network for the Human3.6 challenge. The input image is first analyzed through a CNN with Residual blocks. The feature maps are sent into a Bayesian CapsNet as defined in this paper. The CapsNet output provides encoded vectorial features that are decoded in three different versions of the same concept: 2D and 3D coordinates together with Joint heat maps. The loss is a self-balanced combination of the loss from each task plus the one due to the prior on the Capsnet.

convolutional neurons are equivalent to fully connected neurons with many of their inputs disconnected, i.e., thus having dropout implicitly implemented. Finally, we stress that the activation of the last layer in [Sabour et al., 2017] is the squash function, not ReLU, as in ours. Thus, we force the CNN to produce representations in the positive $(16\times16\times16)$-dimension *cuadrant*.

6.2.2 Bayesian Capsnet

The proposed BCN is summarize through 5 steps as follows.

Input It is a tensor of size $M \times N \times D = 16 \times 16 \times 16$. As in [Sabour et al., 2017], it is modified with the squash function so its norm is upper bounded to 1.

Encapsulation According to Algorithm 6.1, we have to choose J and S, the number of capsules that we initialize and their size such that the precondition is satisfied. Given the input tensor, our choice is $J = 512$ capsules of size $S = 8$, satisfying that $512 \times 8 = 16 \times 16 \times 16$. Each capsule is cloned $K = 17$ times. This choice is arbitrary so we set 17 because there are 17 joints to predict. Thus we will have one final capsule for every joint. In this step, we introduce dropout by setting to zero a 30% of the $512 \times 8 \times 7$ components that we have considering all the capsules and all the clones. The result is grouped in blocks as depicted in Figure 6.1.

Inverse graphics The procedure is the one described in Algorithm 6.1. Specifically, there are 17 blocks and 512×17 matrices $A_{k,j}$, each one producing a respective capsule $\mathbf{v}_{k,j}$ of size $S' = 16$.

Routing by agreement Capsules outgoing from each block are averaged according to coefficients $c_{k,j}$. These coefficients are computed using the *Routing by agreement* algorithm, described in detail in [Sabour et al., 2017], where authors defined it as a version of the Expectation-Maximization procedure. This algorithm runs twice in the feed-forward step of training and none in the back-propagation step.

Output The capsules resulting by the Routing are finally squashed. Therefore this Capsnet module produces 17 capsules of 16 components, each capsule with norm upper bounded to one.

6.2.3 Estimation and Reconstruction

In this proposed architecture, capsules encode joints of the skeleton sketch. The goal is to estimate the coordinates of each joint in 3D, their projections in 2D and a heat map of each joint at the same time. The hypothesis is that these three tasks are complementary, so they help each other. Such approach, in fact, forces the architecture to create a meaningful space of latent variables from which the three aforementioned targets must be reconstructed. Conversely to other approaches, here we directly infer 2D and 3D coordinates, this is, no post-processing or joint locations is deemed. This greatly increments the problem difficulty. To this end we make use of three FCNNs, one for each output, $\hat{y}_{3D}$, $\hat{y}_{2D}$ and $\hat{y}_{joints}$.

Let us define the following layers:

$\texttt{Dense}_{[x]}$ a Dense layer of x neurons with ReLU activation,

$\texttt{Sigm}_{[x]}$ a Dense layer of x neurons with Sigmoid activation,

$\texttt{Drop}_{[x]}$ a $x\%$ dropout layer,

then the following expressions describe each FCNN:

$$z_1 = z_2 = z_3 = \texttt{Drop}_{[15]}\left(\texttt{Dense}_{[2048]}\left(\texttt{Dense}_{[1024]}\left(\Phi\right)\right)\right), \tag{6.18}$$

$$\hat{y}_{2D} = \texttt{Sigm}_{[34=17\times 2]}\left(z_1\right), \tag{6.19}$$

$$\hat{y}_{3D} = \texttt{Sigm}_{[51=17\times 3]}\left(z_2\right), \tag{6.20}$$

$$\hat{y}_{joint} = \texttt{Reshape}_{256\times 256\times 16}\left(\texttt{Dense}_{[699632]}\left(z_3\right)\right). \tag{6.21}$$

Notice that they don't share layers but the structure is identical and the header is different. The loss function for this problem is the expansion of (6.16) to our set of tasks $\mathcal{T} = \{2D, 3D, joints\}$ so

$$\mathcal{L} = e^{-s_b}\mathcal{L}_{\text{b}} + s_b + \sum_{\tau\in\mathcal{T}}\left(e^{-s_\tau}\mathcal{L}_\tau + s_\tau\right). \tag{6.22}$$

6.3 Experimental results and discussions

In this Section we first describe the *Human 3.6* data set [Ionescu et al., 2014] together with works related to the 3D human pose estimation from a single 2D image. Then we compare our proposal with the works that attained the best performances in the CVPR'17 and also in the IJCV, Jan.'18.

6.3.1 Human 3.6 Data set and related works

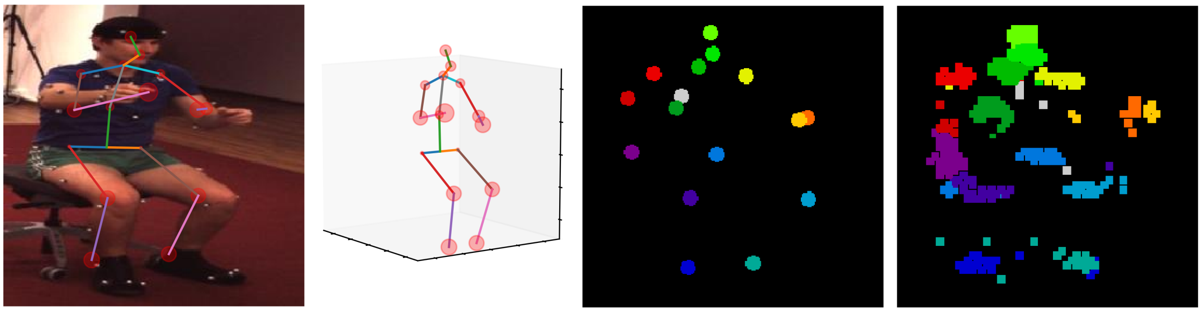
The 3D human pose estimation problem has attracted a lot of interest during the last years both in the computer vision and machine learning communities; but not all the works are comparable to our proposal. For example, the variational approach in [Kolev et al., 2009] uses multiple views for reconstructing the volume rather than the position of the joints; while the Deep Regression Bayesian Network proposed in [Nie et al., 2018] in for reconstructing 2D images (inpainting, block occlusion and face restoration).

The Table 6.1 collects the state of the art works that aim at estimating the 3D coordinates of every joint in the skeleton sketch from 2D images taken from a single view, summarizing the main features of their

Works	E2E	R.I.	P.	T.	T.L.	M.	O.D.
Ours	Y	Y	Y*				
Tekin [Tekin et al., 2017]					Y	Y	Y
Tome [Tome et al., 2017]					Y	Y	Y
Zhou [Zhou et al., 2016]				Y		Y	Y
Katircioglu [Katircioglu et al., 2018]			Y*		Y	Y	Y
Bogo [Bogo et al., 2016]			Y*			Y	Y
Sanzari [Sanzari et al., 2016]			Y*			Y	Y

Y: yes, **Y***: yes in table 6.3; **E2E.** End to end approach, **R.I.** Rotation invariant,
P. Procrustes transformation, **T.** Temporal information, **T.L.** Transfer Learning,
M. Biometric Model, **O.D.** Other datasets.

Table 6.1: State of the art works that have used the Human3.6 dataset.



(a) (b) (c) (d)

Figure 6.5: (a) An input RGB image from the validation set and its 2D predicted coordinates. (b) 3D predicted joints together with the inner epistemic uncertainty (the circle magnitude). (c) Ground truth for the joint heat maps. (d) Joint heat maps estimated. Each joint is depicted with a different color.

solutions. The two columns on the left are qualities that our work exhibit, namely being an end-to-end approach (column E2E) and that the pose is invariant to rotations with respect to the vertical axis (column R.I.) By end-to-end we mean that the 3D estimation does not have to depend on anything other than the 2D image. In that sense other works use a *pipeline* that firstly focus on performing very well in an estimation of the the 2D joints, an then estimate the third dimension. Additionally, these works use other data sets, not only Human3.6. (column O.D.) Another not end-to-end approach, that also uses more data sets, consists of learning a richer set of features that, afterwards, by Transfer Learning (column TL), are used for training their solutions. The rest of columns refer to aids, such as Procrustes transformations (column P), using previous frames to aid the prediction in the current time (column T) or having a biometric model (column M).

The Human 3.6 data set consists of 15 videos that add up to a total of 3.6 million RGB images. Each video shows a person doing a different activity. The videos are sub-sampled at 10 fps, resulting in 311,724 images for training and 110,040 for testing. The person is bounding-boxed in every frame and, together with the images, the data set provides a ground truth given by the 3D coordinates of each joint in the skeleton, scaled to the interval $[0, 1]$, as an array of size 17×3 elements. Additionally, from this information we generate *Joint heat maps*, i.e. images with 17 channels, each of them representing a binary mask of the joint locations as depicted in Figure 6.5. We also crop every frame by the bounding box and re-scale to 300×300 to obtain the input image.

We trained our NN following the *Protocol* $\#1$ as described in [Ionescu et al., 2014]: subjects S1, S5,

Activity	Zhou	Tekin	Tome, I	**Ours, I**	Tome, II	**Ours, II**	**Ours, III**
Directions	87.36	85.03	68.55	79.42	**64.98**	73.15	73.33
Discussion	109.31	108.79	78.27	83.73	**73.47**	84.95	83.45
Eating	87.05	84.38	77.22	84.01	**76.82**	85.87	85.33
Greeting	103.16	98.94	89.05	83.15	86.43	80.12	**79.08**
Phoning	116.18	119.39	91.63	86.42	**86.28**	91.44	89.99
Photo	143.32	**95.65**	110.05	112.38	110.67	109.42	109.95
Posing	106.88	98.49	74.92	81.34	**68.93**	76.4	76.08
Purchases	99.78	93.77	83.71	77.65	74.79	76.72	**73.61**
Sitting	124.52	**73.76**	115.94	105.10	110.19	105.54	104.12
SittingDown	199.23	170.4	185.72	135.55	173.91	**130.15**	136.27
Smoking	107.42	85.08	88.25	88.25	**84.95**	88.07	87.59
Waiting	118.09	116.91	88.73	79.24	85.78	80.25	**79.19**
WalkDog	114.23	113.72	92.37	87.45	**86.26**	88.75	87.13
Walking	79.39	**62.08**	76.48	67.56	71.36	66.1	66.31
WalkTogether	97.7	94.83	77.95	80.45	**73.14**	76.84	76.88
Avg. by activity	112.91	100.08	93.26	88.78	88.53	87.58	**87.22**
Std. Dev.	27.78	24.21	27.63	16.28	26.21	**15.86**	17.15

Table 6.2: Comparison of three versions of the Bayesian Capsule Networks with respect to the top-3 in CVPR'17 Human 3.6 challenge, due to Zhou [Zhou et al., 2016], Tekin [Tekin et al., 2017] and Tome [Tome et al., 2017]. The background bars are sized with respect to the max. and min. of every row

S6, S7, S8 are used for training, while S9 and S11 are kept for test. The reported metrics are the averaged errors of the Euclidean Distances of the 17 joints (we do not use others joints to help the training step). We used `AdamOptimizer` starting with a learning rate of 10^{-5} and batch-size of 1, and increasing it by 10 until 20 when the loss reaches a plateau.

6.3.2 Comparison with the State Of The Art

We first compare our NN with the top-3 less averaged error on the same dataset and with the same protocol as reported in Tome et al.[Tome et al., 2017] at CVPR'17, namely: Zhou et al.[Zhou et al., 2016], Tekin et al.[Tekin et al., 2017] and Tome et al.[Tome et al., 2017]. We have removed Sanzari et al [Sanzari et al., 2016] from this top-3 because they use the Procrustes transformation. For such particular comparison, we include it them in table 6.3. A Procrustes transformation consists of a geometric transformation where only a combination of translation, rotation and uniform scaling is allowed.

The best result is due to the second proposal of Tome et al; in which the authors use a 6-stage deep architecture to improve belief-maps in each stage, projecting in 2D a proposed 3D pose from a Probabilistic pre-trained 3D Model. It should be noted that all those state of the art approaches belongs to a category of methods that follows a pipeline, i.e., does not directly infer the 3D coordinates. In general, direct methods performs quite poorly in comparison to pipeline approaches. However, our BCN arquitecture achieves state of the art results in an end-to-end fashion while uses only the considered Human3.6M dataset to train.

To assess the contribution of the bayesian approach both in the Capsnet and in the FCNN headers, we try three different versions of our solution. The first version (Ours-I) has dropout only in the Capsnet, but there is no regularization over matrices $A_{k,j}$ nor dropout in the FCNN headers. The second version (Ours-II) incorporates the regularization over matrices $A_{k,j}$. Finally, the third version (Ours-III) is the full solution as described in Section 6.1.1.

The results of the comparison are detailed per activity in the rows of Table 6.2. Each cell in the table presents the error accumulated by the 17 joints averaged over the whole activity video-sequence. The last two rows are the average by activity and its standard deviation respectively. Results show that:

1. The best method in [Tome et al., 2017] is almost matched by Ours-I, and overtaken by Ours-II and Ours-III. We show 8 frames of one test video-sequence together with the 3D estimation and the 3D ground truth in Figure 6.6, from different views.

2. The homoscedastic uncertainty σ is much lower with any version of our proposed Bayesian Capsnet than with the methods reported in [Tome et al., 2017]. This is an expected result, since the loss function includes the homoscedastic uncertainty as an objective. The minimum is attained by Ours-II, i.e. when FCNN headers have no dropout. Moreover, the improvement due to Ours-III in the average is not as much as the improvement due to Ours-II, suggesting that it could account for an excess of regularization.

3. The epistemic uncertainty σ_w is indirectly estimated as the variance of predictions due to different samples of the NN. In the bayesian formulation, sampling a NN is approximated as dropout during the evaluation of an instance in all those layers where it is possible. To this end we take the image shown in Figure 6.5(a) and produce 50 predictions, each one with a sample from Ours-III. The standard deviation of the 2D and 3D predictions is shown as the magnitude of balls surrounding each joint in Figures 6.5(a) and 6.5(b) respectively.

4. Compared with the rest of solutions in Table 6.2, our proposal is much more straightforward since it directly aims at inferring the 3D, 2D and heat maps. Specifically, the proposal in [Tome et al., 2017] uses other data sets in the stages that precede the estimation of the 3D coordinates, which accounts for a complex system structure.

The best performance in 3D human pose estimation from a single image so far has been recently reported in [Katircioglu et al., 2018]. When using Procrustes for comparing with respect to the ground-truth For the sake of completeness, we incorporate this transformation to the fully bayesian architecture, referred to as Ours-IV in Table 6.3, and compare with Sanzari et al.[Sanzari et al., 2016], Bogo et al.[Bogo et al., 2016] and Katircioglu et al. [Katircioglu et al., 2018].

Our proposal ranks second with a significant improvement both in the accuracy and in the homoscedastic uncertainty. However, we remark that [Katircioglu et al., 2018] is also aided by Transfer learning, Biometric model and the use of other data sets. Specifically, their approach consists in the use of a pre-trained overcomplete 3D pose autoencoder, using the latent space created by each 3D sample as target for a CNN. The final 3D output is given by the pre-trained decoder, which receives an estimation of the latent space from the CNN feeded with a single image. They propose a *ShallowNet-Autoencoder* CNN, which is comparable to our CNN with residual connections, and a *ResNet50-Autoencoder* pre-trained for 2D joint heat-map predictions. With the ShallowNet, they achieve 127.07mm error while the ResNet50 boosts the results to their best (without Procrustes), 67.27 mm; which shows that the improvement is probably due to the deep ResNet50 architecture.

Activity	Sanzari	Bogo	**Ours, IV**	Katircioglu
Directions	48.82	62	57.55	**43.89**
Discussion	56.31	60.2	61.32	**48.54**
Eating	95.98	67.8	66.48	**46.57**
Greeting	84.78	76.5	64.49	**49.95**
Phoning	96.47	92.1	68	**53.94**
Photo	105.58	77	83.16	**59.29**
Posing	66.3	73	56.05	**43.77**
Purchases	107.41	75.3	54.85	**43.94**
Sitting	116.89	100.3	77.65	**60.2**
SittingDown	129.63	137.3	97.32	**73.64**
Smoking	97.84	83.4	67.31	**51.15**
Waiting	65.94	77.3	59.63	**46.3**
WalkDog	130.46	79.7	64.76	**52.25**
Walking	92.58	86.8	49.96	**39.81**
WalkTogether	102.21	81.7	60.47	**47.18**
Avg. by Activity	93.15	82.03	65.93	**50.69**
Std. Dev.	23.97	17.90	11.74	**8.23**

Table 6.3: Comparison of three versions of the Bayesian Capsule Networks with respect to the top-3 in IJCV'18, due to Sanzari [Sanzari et al., 2016], Bogo [Bogo et al., 2016] and Katircioglu [Katircioglu et al., 2018]. The background bars are sized with respect to the max. and min. of every row.

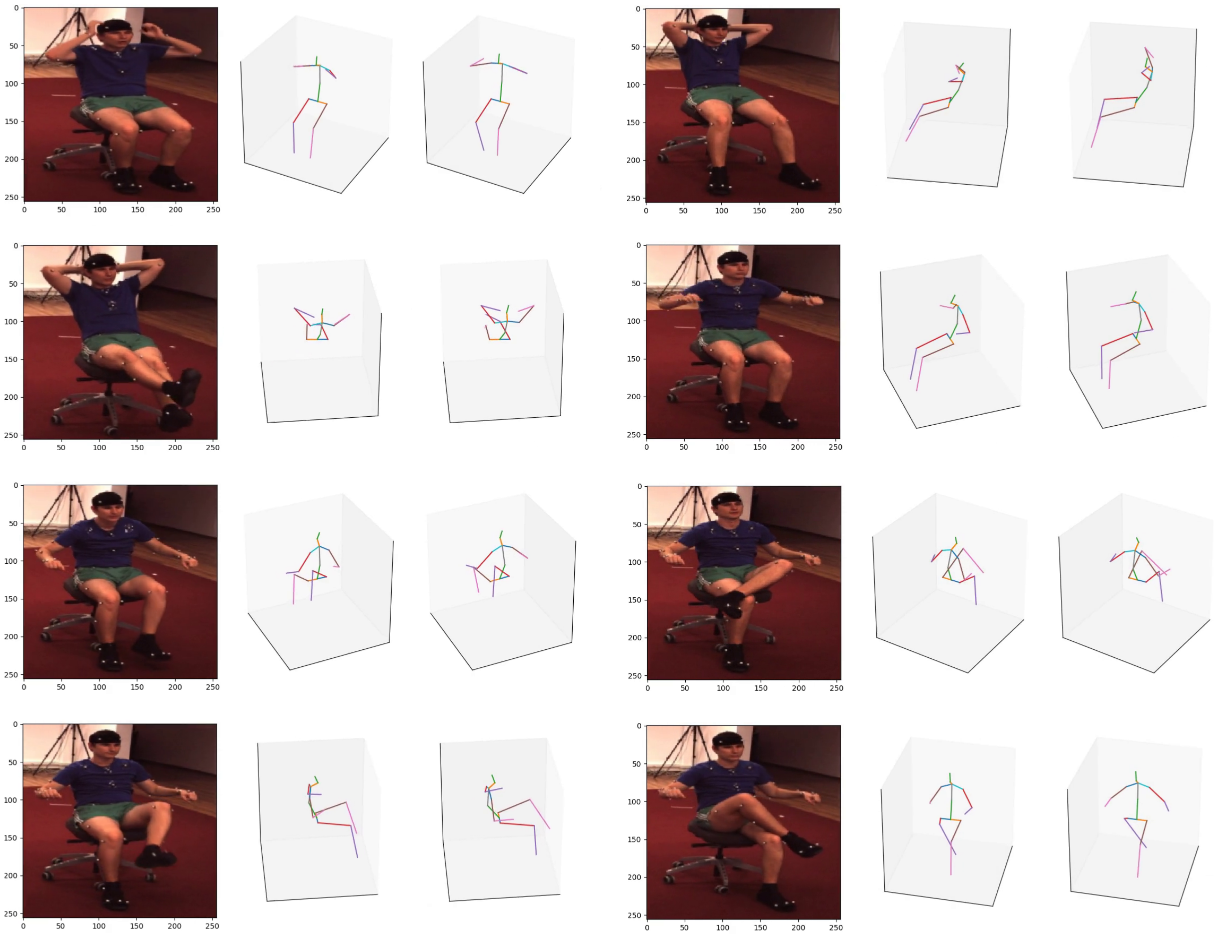


Figure 6.6: Some results of our proposed Bayesian Capsule Network in a test subject. (Left) The 2D RGB image input, (middle) the 3D predicted Human Pose, and (right)the given ground-truth. 3D skeletons have different viewpoints for the sake of visualization, but all of them have the same orientation.

CHAPTER 7

Conclusions

La realidad imita a la tele ...
Eduardo Galeano

This thesis aims to propose new models and their resolutions for problems concerning the CV field. Applications in CV deal with a particular signal of information, namely, images. We have seen that many problems can be addressed with Variational Methods and Machine Learning techniques, but, although those approaches seem, and eventually are, quite different in practice, there are some key elements that are very similar. This Chapter is devoted to resume the main conclusions and contributions of this work, as well as outline research lines that I plan to address in the near future.

7.1 Practical Conclusions

In Chapters 2 and 3 , we aim to lay bridges between Variational Methods and Deep Learning. For this purpose, both approaches are explained from a bayesian modeling framework. Besides, we highlight common elements that generalize the understanding of those techniques. Finally, we show that, in fact, they may be derived from a general methodology framed in the field of Bayesian Inference.

In Chapter 4 we have presented a new non-local non-convex diffusion model for saliency detection and classification which promotes a fast foreground detection when it is applied to a FLAIR given image. This new approach is based on reactive flows which facilitates the saliency detection task promoting binary solutions which encapsulate the underlying classification problem. The computational cost of the resulting algorithm is greatly alleviated by using recent ideas on quantized convolutional operators filters, making suth an approach practical and efficient.

The results reveal that this method can achieve very high accurate statistics metrics over the ground-truth BRATS2015 dataset [Menze et al., 2015]. Also, as a by-product of the reactive model, the solution has, after few iterations, a reduced number of quantized values making simpler the final thresholding step. Such a technique could be improved computationally by observing that the diffusion process combined with the saliency term evolves producing more cartoon like piece-wise constant solutions which can be coded with less number of quantization values while converging to a binary mask. This is related to the absorption-reaction balance in the PDE where absorption is active where the solution is small, $u \approx 0$ and the reaction is active where $u \approx 1$.

The non-local diffusion properties of the model also allow to detect salient objects which are not spatially close as well as connected regions (disjoint areas). This can be useful in many other medical images modalities, specially in Functional Magnetic Resonance Imaging (fMRI). Non-convex properties, meanwhile, promote *sparse* non-local gradient, pushing the solution to a cartoon piece-wise constant image.

We have also proposed the analogous local version of the model that particularizes the regularization term to the TV operator. The elliptic equation that results from the model is then solved through the Chambolle and Pock Primal Dual algorithm [Chambolle and Pock, 2011]. As a counterpart of the non local model, this local version is faster in its numerical resolution. We take advantage of this fact to embed it in a CNN and built a Deep Variational Framework that allows to optimize the parameters of our model using previous knowledge from the application. The test results confirm the potential of the proposed approach. Moreover, this Variational Framework can easily consider different variational models without more than adapt their numerical resolution to a trainable graph.

In Chapter 5 we have presented a computer vision-based method for dumpsters classification using CNNs. The first approach we have carried out is based on Transfer Learning, which allows to take advantage of the already trained Google Inception-v3 CNN. Such strategy reduces the training time as well as the amount of labeled images we need for feeding the CNN. This has led us to a semi-supervised approach where we have presented two iterative methods for getting an accurate CNN. The first one, based on a worst cases selection, and the second one, based on a GMM technique. We have shown that is it possible to use such methods for selecting a very small set of images to be manually labeled, obtaining a higher performance than selecting them randomly. Furthermore, the effort to label manually the training set of images in both cases is the same. The results we have presented show that our proposal increased the performance of the intended CNN, 25% and 34% for the Half-Worst and GMM based methods respectively, in terms of accuracy. In comparison with a random selection of the tiny training set, the obtained CNN using the proposed methods reveals a sightly better prevention of the over-fitting.

In Chapter 6 we present, for the first time, a Bayesian Capsule Network. As far as we know, it is also the first attempt to address a regression problem (rather than a classification on) with a Capsule Network. Bayesian Neural Networks are easy to implement and offer two big advantages. By doing dropout and small modifications in the usual loss functions, both in classification and regression, it is possible to: 1) minimize the homoscedastic uncertainty, and 2) use multiple losses for different complementary tasks, self-balancing their contribution to the total loss. Due to the structure of Capsnets, its bayesian formulation consists of doing dropout on the initial capsules and a regularization term. We have tested the proposal on an ill-posed problem and have shown that the results are comparable to those of the state of the art, but using a straightforward and much simpler approach over the Human3.6 dataset.

7.2 General Conclusions

There is a significant concept throughout this dissertation that, from my perspective, deserves to be treated in a more general way. Regularization, prior knowledge and generalization are the reflect of the need of abstraction.

For example, from the Variational field, Tikhonov regularization term such as the L^2 energy of the gradient of the solution, in the Image Restoration case, makes sense because it exploits the fact that images have very often smooth regions (nearby pixels are similar). An improvement of this kind of regularization is the TV operator that preserves sharp edges while smoothes the rest of the image. Regarding the statistics of images (Figure 2.3b), we can realize that other regularizing terms could be more appropriated (hyper-laplacian among others). This is similar to learn from data in the DL context, the difference is who processes that information. On the other hand, from DL and NNs, huge improvements have been achieved when using CNNs in CV problems. We have also shown how to understand convolutions as neural network layers with *infinitely strong priors*, i.e., the regularization here is introduced implicitly in the structure of the convolution operation. This was obtained after understanding the hierarchical features extraction that the animal's visual system performs to detect objects. In short, improvements come when someone finds out a way to include extra knowledge.

The target field of this thesis is CV. However, the willingness to solve problems of this field using different approaches and, possibly, a combination of them, has opened the way to link up concepts that lead us to think that they have more in common than we often realize, to the extend that they can be merged in one. Similarly, not only the techniques to solve problems of CV, but problems themselves, can also be taken from a more general viewpoint.

Abstraction is the key concept.

Of course, this is not new. Nonetheless, knowledge seems to be increasingly specialized. More and more universities degrees appear and *the key of success* relies on choosing a particular topic and try to deepen on it (notice the irony). As a result of this work, I claim, this is a mistake. Analysis without synthesis is incomplete.

Returning to Variational and Deep Learning Methods, we have seen that both have advantages and drawbacks that are complementary. Variational problems provide a great control as well as very accurate results if adequate priors are found. DL aims at finding those priors from data, avoiding any direct human supervision and thus increasing uncertainty. Beyond how to resolve their inconveniences, this lead us to a more important question. What is the correct way to acquire such prior knowledge? Is there a principled way to do it? In brief:

What is learning?

We return back to the Introductory Chapter 1, where we addressed this concept as a kind of process. Learning is the process of acquiring knowledge. Thus, what is knowledge? One can quickly fall on philosophical questions. It is well-known that philosophy does not provide answers, rather, she aims to formulate the correct questions. DL, despite its lack of foundations mainly due to its rapid growth, has pointed out this very interesting and still nowadays unsolved question of learning. It often occurs in science that before discovering the principles and paradigms of a field, some practical results are found:

the first airplanes began to fly before the principles of aeronautics were discovered. I personally believe, despite the hype in media of DL and AI, which are sometimes unjustified and not realistic, this must be seen as an invitation to other fields and researchers.

> *"The isolated man does not develop any intellectual power. It is necessary for him to be immersed in an environment of other men, whose techniques he absorbs during the first twenty years of his life. He may then perhaps do a little research of his own and make a very few discoveries which are passed on to other men. From this point of view the search for new techniques must be regarded as carried out by the human community as a whole, rather than by individuals." Alan Turing*

To this end, it is necessary to share a common language to avoid misunderstandings. For instance, the *pooling* step that is done between layers of CNN is nothing but a sub-sampling operation. NNs are usually explained by DL practitioners through their structures and objectives rather than by their definition, in the sense that they are Universal Approximation Functions. Even if this may seem trivial, it is indeed a limitation depending on the background of each individual that starts in this field. From the Mathematical viewpoint, it is well-known how solid are the foundations of such a discipline, however, here the language is very often a limitation if we do not have previous background. In my opinion the reason is, while in DL there is a lack of foundations but many applied examples and available tools, in Variational Methods there are solid basis but too little applied examples to allow a wide spread use of them. For example, the gap between the development and properties of the TV and the solution of the ROF denosing model with duality arguments to its practical code implementation (just few lines) is too big. This thesis aims also to reduce this gap to motivate collaboration across Variational and Deep Learning practitioners.

The study of Variational Methods and Deep Learning in parallel has led me to better understand them. Also, it has allowed me to find a framework that extends both: Bayesian Inference. NNs explained through Bayesian Inference in Chapter 3, find correlations in data. Based on this, a large number of tasks can be faced. AI community is starting to go a step further, and consider to replace correlations between features or events by causes and effects. Causal Inference may be a good opportunity to get involved in the AI field, which, certainly, will required to a huge effort in both of theoretical and practical development.

This causality perspective give us in fact a bright view in hindsight of the problems we were solving in this thesis.

First, in Chapter 4 we proposed a variational model for Saliency detection. From a causal perspective, it is a direct problem (2.1) as long as the output of the method is an effect (segmentation). However, since we also consider a fidelity term, in fact we start from a denoising model (restoration), an inverse problem is also involved in the process. A reaction-absorption term (saliency) is opposed to a diffusion term. This must make us think if some changes could improve the model. For instance, create a fidelity term between in the space of the gradients rather that with respect to the given image that is confined to evolve to a nearly binary one.

Second, in Chapter 5, the proposed semi-automatic methods to train a CNN for image classification of dumpsters, display that a highly accurate DL system can be obtain using very little data reducing, in addition, the uncertainty. A causal analysis of such results tell us that, besides the importance of selecting good images rather than increasing the quantity of them, the order of samples in the training step can be very relevant. This is quite natural if we translate to how humans learn, however, very little works have been carried out.

Finally, in Chapter 6 we aim at recovering a 3D human pose from single 2D images. Such a process is an inverse problem as we have defined in Chapter 2. Making use of novel NN architectures and the Bayesian Inference tools, we have formulated the BCNs that include a regularization over the considered parameters. This regularization comes from the fact that if a NN is trained to inverse a process (which is our case), it is mandatory to guarantee a sort of invertibility of this NN (the direct problem).

7.3 Future work

Future works include all the above mentioned ideas. Moreover, the resulting conclusions of this work lead us to expand the scope of research from CV to AI. Several ideas and hints have been drawn to define new lines of research. Among others:

1. Uncertainty must be addressed in Deep Learning
2. Variational Methods are boosted if data is considered
3. Causality matters
4. Abstraction is the way to Artificial General Intelligence (AGI)

The last refers to achieve Machines that can perform any task a human is able to do. In this thesis we have shown that generalizing the methodology for CV problems allows us to built simple and reduced architectures that in turn achieve state of the art problems and can be used for other different purposes as well. Learning with less data, Predictive models, Self-supervised Learning (formerly called Unsupervised Learning) or Causal Inference are few topics that can shed light in the next years. In this sense, the next step starts formulating the following question:

What is the general mechanism for abstraction?

List of Publications

Published Articles

- **Ramírez, I.**, Cuesta-Infante, A., Pantrigo, J. J., Montemayor, A. S., Moreno, J. L., Alonso, V., Anguita, G. Palombarani, L. (2018). Convolutional neural networks for computer vision-based detection and recognition of dumpsters. Neural Computing and Applications, 1-9.

In Peer-reviewed Articles

- Galiano, G., **Ramírez, I.**, Schiavi, E. (2018).Non-convex non-local flows for saliency detection. *ArXiv e-prints,arXiv:1805.09408v1. Under Review. Journal of Computation and Applied Mathematics*

- **Ramírez, I.** Cuesta-Infante, A., Schiavi, E., Pantrigo, J. Ph.D. Bayesian Capsule Networks for 3D human pose estimation from single 2D images. *Under Review. Neurocomputing*

International Conference Contributions

- **Ramírez, I.**, Martín, A., Schiavi, E. (2018). Optimization of a variational model using deep learning: an application to brain tumor segmentation. In Biomedical imaging (isbi 2018), 2018 ieee 15th international symposium on (pp. 631–634). IEEE.

- **Ramírez, I.**, Galiano, G., Malpica, N., Schiavi, E. (2017). A non-local diffusion saliency model for magnetic resonance imaging. In Proceedings of the 10th international joint conference on biomedical engineering systems and technologies - volume 2: bioimaging, (biostec 2017) (pp. 100–107). INSTICC. SciTePress. doi:10.5220/0006172101000107

National Conference Contributions

- **Ramírez, I.**, Martín, A., Schiavi, E. (2017). Numerical resolution of a quasi-linear elliptic equation for saliency detection. In Cedya + cma 2017 xxv congreso de ecuaciones diferenciales y aplicaciones / xv congreso de matemática aplicada - communications book.

Book Chapters

- Alcaín, E., Muñoz, A. I., **Ramírez, I.**, Schiavi, E. (2019). Modelling sparse saliency maps on manifolds: numerical results and applications. Cham: Springer International Publishing. doi:10.1007/978-3-030-00341-8_10

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Ambrosio et al., 2000] Ambrosio, L., Fusco, N., and Pallara, D. (2000). *Functions of bounded variation and free discontinuity problems*. Oxford university press.

[Andreu-Vaillo et al., 2010] Andreu-Vaillo, F., Mazón, J. M., Rossi, J. D., and Toledo-Melero, J. J. (2010). *Nonlocal diffusion problems*, volume 165. American Mathematical Society.

[Athalye et al., 2017] Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2017). Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*.

[Aubert et al., 2005] Aubert, G., Aujol, J.-F., and Blanc-Féraud, L. (2005). Detecting codimension—two objects in an image with ginzburg-landau models. *International Journal of Computer Vision*, 65(1-2):29–42.

[Aubert and Kornprobst, 2006] Aubert, G. and Kornprobst, P. (2006). *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer Science & Business Media.

[Austin, 1961] Austin, J. L. (1961). Philosophical papers. In Urmson, J. O. and Warnock, G. J., editors, *Philosophical papers*.

[Ba et al., 2014] Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. In *Proceedings of Int. Conf. on Learning Representations*.

[Bell, 1978] Bell, J. B. (1978). Solutions of ill-posed problems.

[Bergbauer et al., 2013] Bergbauer, J., Nieuwenhuis, C., Souiai, M., and Cremers, D. (2013). Proximity priors for variational semantic segmentation and recognition. *2013 IEEE International Conference on Computer Vision Workshops*, pages 15–21.

[Bernardi et al., 2016] Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2016). Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

[Bishop, 1998] Bishop, C. M. (1998). Variational learning in graphical models and neural networks. In *ICANN 98*, pages 13–22. Springer London.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proc. of the 32Nd Int. Conf. on Machine Learning*, ICML'15, pages 1613–1622.

[Bogo et al., 2016] Bogo, F., Kanazawa, A., Lassner, C., Gehler, P. V., Romero, J., and Black, M. J. (2016). Keep it SMPL: automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, pages 561–578.

[Brézis et al., 2003] Brézis, H., Chang, K., Li, S., and Rabinowitz, P. (2003). *Topological Methods, Variational Methods and Their Applications: Taiyuan, Shan Xi, PR China, August 14-18, 2002*. World Scientific.

[Brinez et al., 2015] Brinez, L. J. C., Rengifo, A., and Escobar, M. (2015). Automatic waste classification using computer vision as an application in colombian high schools. In *6th Latin-American Conference on Networked and Electronic Media (LACNEM 2015)*, pages 1–5.

[Bylinskii et al.,] Bylinskii, Z., Judd, T., Borji, A., Itti, L., Durand, F., Oliva, A., and Torralba, A. Mit saliency benchmark.

[Cahn and Hilliard, 1958] Cahn, J. W. and Hilliard, J. E. (1958). Free energy of a nonuniform system. i. interfacial free energy. *The Journal of chemical physics*, 28(2):258–267.

[Cao et al., 2017] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*.

[Capurro and Hjørland, 2003] Capurro, R. and Hjørland, B. (2003). The concept of information. *Annual review of information science and technology*, 37(1):343–411.

[Casella and Berger, 2002] Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

[Cerf et al., 2008] Cerf, M., Harel, J., Einhäuser, W., and Koch, C. (2008). Predicting human gaze using low-level saliency combined with face detection. In *Advances in neural information processing systems*, pages 241–248.

[Chambolle et al., 2010] Chambolle, A., Caselles, V., Cremers, D., Novaga, M., and Pock, T. (2010). An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227.

[Chambolle and Lions, 1997] Chambolle, A. and Lions, P.-L. (1997). Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188.

[Chambolle and Pock, 2011] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145.

[Chen et al., 2014] Chen, Y., Ranftl, R., and Pock, T. (2014). Insights into analysis operator learning: From patch-based sparse models to higher order mrfs. *IEEE Transactions on Image Processing*, 23(3):1060–1072.

[Chien and Ku, 2016] Chien, J. and Ku, Y. (2016). Bayesian recurrent neural network for language modeling. *IEEE Trans. Neural Netw. Learning Syst.*, 27(2):361–374.

[Colson et al., 2007] Colson, B., Marcotte, P., and Savard, G. (2007). An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[Dai et al., 2007] Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007). Transferring naive bayes classifiers for text classification. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 540–545. AAAI Press.

[Dauphin et al., 2014] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941.

[Dautray and Lions, 2012] Dautray, R. and Lions, J.-L. (2012). *Mathematical Analysis and Numerical Methods for Science and Technology: Volume 6 Evolution Problems II*. Springer Science & Business Media.

[Dechter,] Dechter, R. *Learning while searching in constraint-satisfaction problems.*

[Demengel, 1999] Demengel, F. (1999). On some nonlinear partial differential equations involving the "1"-laplacian and critical sobolev exponent. *ESAIM: Control, Optimisation and Calculus of Variations*, 4:667–686.

[Demengel and Demengel, 2012] Demengel, G. and Demengel, F. (2012). *Espaces fonctionnels-Utilisation dans la résolution des équations aux dérivées partielles: Utilisation dans la résolution des équations aux dérivées partielles.* EDP Sciences.

[Deng et al., 2012] Deng, J., Krause, J., Berg, A. C., and Fei-Fei, L. (2012). Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3450–3457. IEEE.

[Denker and LeCun, 1990] Denker, J. S. and LeCun, Y. (1990). Transforming neural-net output levels to probability distributions. In *Advances in Neural Information Processing Systems 3*, pages 853–859.

[Donoser et al., 2009] Donoser, M., Urschler, M., Hirzer, M., and Bischof, H. (2009). Saliency driven total variation segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 817–824. IEEE.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

[Elmoataz et al., 2008] Elmoataz, A., Lezoray, O., and Bougleux, S. (2008). Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE Trans. Image Processing*, 17(7):1047–1060.

[Fang et al., 2015] Fang, H., Gupta, S., Iandola, F., Srivastava, R., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J. C., Zitnick, C. L., and Zweig, G. (2015). From captions to visual concepts and back. In *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*.

[Floridi, 2017] Floridi, L. (2017). Semantic conceptions of information. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition.

[Fukui et al., 2016] Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M. (2016). Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA.

[Fukushima and Miyake, 1982] Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

[Gal and Ghahramani, 2016a] Gal, Y. and Ghahramani, Z. (2016a). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

[Gal and Ghahramani, 2016b] Gal, Y. and Ghahramani, Z. (2016b). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. of The 33rd Int. Conf. on Machine Learning*, volume 48, pages 1050–1059.

[Galiano et al., 2016] Galiano, G., Schiavi, E., and Velasco, J. (2016). Well-posedness of a nonlinear integro-differential problem and its rearranged formulation. *Nonlinear Analysis: Real World Applications*, 32:74–90.

[Galiano and Velasco, 2015] Galiano, G. and Velasco, J. (2015). On a fast bilateral filtering formulation using functional rearrangements. *Journal of Mathematical Imaging and Vision*, 53(3):346–363.

[Gao et al., 2017] Gao, Y., Ma, J., Zhao, M., and Yuille, A. L. (2017). Semi-supervised sparse representation based classification for face recognition with insufficient labeled samples. *IEEE Transactions on Image Processing*, PP(99):1–1.

[Gilboa and Osher, 2008] Gilboa, G. and Osher, S. (2008). Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028.

[Ginzburg, 1955] Ginzburg, V. L. (1955). On the theory of superconductivity. *Il Nuovo Cimento (1955-1965)*, 2(6):1234–1250.

[Gordillo et al., 2013] Gordillo, N., Montseny, E., and Sobrevilla, P. (2013). State of the art survey on mri brain tumor segmentation. *Magnetic Resonance Imaging*, 31(8):1426 – 1438.

[Gray et al., 2006] Gray, R. M. et al. (2006). Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239.

[Hadamard, 1902] Hadamard, J. (1902). Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, 13(49-52):28.

[Hao, 2019] Hao, K. (2019). "We analyzed 16,625 papers to figure out where AI is headed next". *MIT Technology Review.*

[Harel et al., 2007] Harel, J., Koch, C., and Perona, P. (2007). Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552.

[Hartley, 1960] Hartley, H. O. (1960). *Journal of the American Statistical Association*, 55(292):758–760.

[Havaei et al., 2017] Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35(Supplement C):18 – 31.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hernández-Lobato and Adams, 2015] Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proc. of the 32Nd Int. Conf. on Machine Learning*, ICML'15, pages 1861–1869.

[Hernández-Lobato et al., 2016] Hernández-Lobato, J. M., Li, Y., Rowland, M., Bui, T. D., Hernández-Lobato, D., and Turner, R. E. (2016). Black-box alpha divergence minimization. In *Proc. of the 33nd Int. Conf. on Machine Learning, ICML 2016*, pages 1511–1520.

[Hintermüller and Wu, 2014] Hintermüller, M. and Wu, T. (2014). A smoothing descent method for nonconvex tv^ q-models. In *Efficient Algorithms for Global Optimization Methods in Computer Vision*, pages 119–133. Springer.

[Hinton et al.,] Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

[Hinton et al., 2018] Hinton, G. E., Sabour, S., and Frosst, N. (2018). Matrix capsules with EM routing. In *Int. Conf. on Learning Representations*.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

[Hong et al., 2014] Hong, I., Park, S., Lee, B., Lee, J., Jeong, D., and Park, S. (2014). Iot-based smart garbage system for efficient food waste management. *The Scientific World Journal*, pages 493–497.

[Hron et al., 2018] Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. (2018). Variational bayesian dropout: pitfalls and fixes. *arXiv preprint arXiv:1807.01969*.

[Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591.

[Idwan et al., 2016] Idwan, S., Zubairi, J. A., and Mahmood, I. (2016). Smart solutions for smart cities: Using wireless sensor network for smart dumpster management. In *2016 International Conference on Collaboration Technologies and Systems (CTS)*, pages 493–497.

[Ionescu et al., 2014] Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014). Human3.6m: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339.

[Ivakhnenko,] Ivakhnenko, A. Cybernetic predicting devices. Technical report.

[Jensen et al., 1996] Jensen, F. V. et al. (1996). *An introduction to Bayesian networks*, volume 210. UCL press London.

[Jordan, 1998] Jordan, M. I. (1998). *Learning in graphical models*, volume 89. Springer Science & Business Media.

[Jordan et al., 1999] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

[Katircioglu et al., 2018] Katircioglu, I., Tekin, B., Salzmann, M., Lepetit, V., and Fua, P. (2018). Learning latent representations of 3D human pose with deep neural networks. *International Journal of Computer Vision*, pages 1–16.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5580–5590.

[Kendall et al., 2017] Kendall, A., Gal, Y., and Cipolla, R. (2017). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*.

[Kindermann et al., 2005] Kindermann, S., Osher, S., and Jones, P. W. (2005). Deblurring and denoising of images by nonlocal functionals. *Multiscale Modeling & Simulation*, 4(4):1091–1115.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kleiner et al., 2014] Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. (2014). A scalable bootstrap for massive data. *J. R. Stat. Soc. B*, 76:795–816.

[Kobler et al., 2017] Kobler, E., Klatzer, T., Hammernik, K., and Pock, T. (2017). Variational networks: Connecting variational methods and deep learning. In Roth, V. and Vetter, T., editors, *GCPR 2017, Basel, Switzerland, Sep 12–15, 2017, Proceedings*, pages 281–293. Springer International Publishing.

[Kolesnikov and Lampert, 2016] Kolesnikov, A. and Lampert, C. H. (2016). *Seed, Expand and Constrain: Three Principles for Weakly-Supervised Image Segmentation*, pages 695–711. Springer International Publishing.

[Kolev et al., 2009] Kolev, K., Klodt, M., Brox, T., and Cremers, D. (2009). Continuous global optimization in multiview 3D reconstruction. *Int. J. of Computer Vision*, 84(1):80–96.

[Kornprobst, 2006] Kornprobst, G. A. P. (2006). Mathematical problems in image processing. *Applied mathematical sciences*, 147.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

[Lebret et al., 2015] Lebret, R., Pinheiro, P., and Collobert, R. (2015). Phrase-based image captioning. In Blei, D. and Bach, F., editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2085–2094. JMLR Workshop and Conference Proceedings.

[LeCun and Bengio, 1995] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li et al., 2014] Li, H., Li, Y., and Porikli, F. (2014). Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

[Li et al., 2018] Li, H., Su, X., Wang, J., Kan, H., Han, T., Zeng, Y., and Chai, X. (2018). Image processing strategies based on saliency segmentation for object recognition under simulated prosthetic vision. *Artificial intelligence in medicine*, 84:64–78.

[Li et al., 2017] Li, M., Liu, X., and Tang, L. (2017). A phase field variational model with arctangent regularization for saliency detection. In *Applications of Computer Vision Workshops (WACVW), 2017 IEEE Winter*, pages 29–35. IEEE.

[Li et al., 2013] Li, M., Zhan, Y., and Zhang, L. (2013). Nonlocal variational model for saliency detection. *Mathematical Problems in Engineering*, 2013.

[Li and Turner, 2016] Li, Y. and Turner, R. E. (2016). Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081.

[Li and Zhou, 2015] Li, Y. F. and Zhou, Z. H. (2015). Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):175–188.

[Linnainmaa, 1976] Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160.

[Liu et al., 2014] Liu, R., Cao, J., Lin, Z., and Shan, S. (2014). Adaptive partial differential equation learning for visual saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3866–3873.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). *SSD: Single Shot MultiBox Detector*, pages 21–37. Springer International Publishing.

[Liu et al., 2013] Liu, Z., Meur, L., and Luo, S. (2013). Superpixel-based saliency detection. In *2013 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4. IEEE.

[Livnat et al., 2010] Livnat, A., Papadimitriou, C., Pippenger, N., and Feldman, M. W. (2010). Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences*, 107(4):1452–1457.

[MacKay, 1992] MacKay, D. J. C. (1992). A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

[Mallat, 2016] Mallat, S. (2016). Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203.

[Mandt et al., 2017] Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *J. Mach. Learn. Res.*, 18(1):4873–4907.

[Martín et al., 2017] Martín, A., Schiavi, E., and de León, S. S. (2017). On 1-laplacian elliptic equations modeling magnetic resonance image rician denoising. *Journal of Mathematical Imaging and Vision*, 57(2):202–224.

[Menze et al., 2014] Menze, B., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., Lanczi, L., Gerstner, E., Weber, M.-A., Arbel, T., Avants, B., Ayache, N., Buendia, P., Collins, L., Cordier, N., Corso, J., Criminisi, A., Das, T., Delingette, H., Demiralp, C., Durst, C., Dojat, M., Doyle, S., Festa, J., Forbes, F., Geremia, E., Glocker, B., Golland, P., Guo, X., Hamamci, A., Iftekharuddin, K., Jena, R., John, N., Konukoglu, E., Lashkari, D., Antonio Mariz, J., Meier, R., Pereira, S., Precup, D., Price, S. J., Riklin-Raviv, T., Reza, S., Ryan, M., Schwartz, L., Shin, H.-C., Shotton, J., Silva, C., Sousa, N., Subbanna, N., Szekely, G., Taylor, T., Thomas, O., Tustison, N., Unal, G., Vasseur, F., Wintermark, M., Hye Ye, D., Zhao, L., Zhao, B., Zikic, D., Prastawa, M., Reyes, M., and Van Leemput, K. (2014). The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, page 33.

[Menze et al., 2015] Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., et al. (2015). The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024.

[Minsky and Papert, 2017] Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

[Morel and Solimini, 1995] Morel, J. M. and Solimini, S. (1995). *Variational Methods in Image Segmentation*. Birkhauser Boston Inc., Cambridge, MA, USA.

[Mujumdar et al., 2013] Mujumdar, S., Rajamani, N., Subramaniam, L. V., and Porat, D. (2013). Efficient multi-stage image classification for mobile sensing in urban environments. In *2013 IEEE International Symposium on Multimedia*, pages 237–240.

[Murea and Tiba, 2013] Murea, C. M. and Tiba, D. (2013). A penalization method for the elliptic bilateral obstacle problem. In *IFIP Conference on System Modeling and Optimization*, pages 189–198. Springer.

[Nesterov, 1983] Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547.

[Newton, 1987] Newton, I. (1987). Philosophiæ naturalis principia mathematica (mathematical principles of natural philosophy). *London (1687)*, 1687.

[Nie et al., 2018] Nie, S., Zheng, M., and Ji, Q. (2018). The deep regression bayesian network and its applications: Probabilistic deep learning for computer vision. *IEEE Signal Process. Mag.*, 35(1):101–111.

[Nielsen and Jensen, 2009] Nielsen, T. D. and Jensen, F. V. (2009). *Bayesian networks and decision graphs*. Springer Science & Business Media.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Parikh and Boyd, 2014] Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239.

[Paul and Pal, 2016] Paul, M. K. and Pal, B. (2016). Gaussian mixture based semi supervised boosting for imbalanced data classification. In *2016 2nd International Conference on Electrical, Computer Telecommunication Engineering (ICECTE)*, pages 1–4.

[Pearl and Mackenzie, 2018] Pearl, J. and Mackenzie, D. (2018). *The book of why: the new science of cause and effect*. Basic Books.

[Pereira et al., 2016] Pereira, S., Pinto, A., Alves, V., and Silva, C. A. (2016). Brain tumor segmentation using convolutional neural networks in mri images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251.

[Pérez-LLanos and Rossi, 2011] Pérez-LLanos, M. and Rossi, J. D. (2011). Numerical approximations for a nonlocal evolution equation. *SIAM Journal on Numerical Analysis*, 49(5):2103–2123.

[Powers, 2011] Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[Riche and Mancas, 2016] Riche, N. and Mancas, M. (2016). *Bottom-Up Saliency Models for Still Images: A Practical Review*, pages 141–175. Springer New York, New York, NY.

[Riegler et al., 2016] Riegler, G., Ferstl, D., Rüther, M., and Bischof, H. (2016). A deep primal-dual network for guided depth super-resolution. *arXiv preprint arXiv:1607.08569*.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer.

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

[Rueda et al., 2013] Rueda, A., González, F., and Romero, E. (2013). Saliency-based characterization of group differences for magnetic resonance disease classification. *Dyna*, 80(178):21–28.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Sabour et al., 2017] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

[Sanzari et al., 2016] Sanzari, M., Ntouskos, V., and Pirri, F. (2016). Bayesian image based 3D pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 566–582. Springer.

[Sermanet et al., 2014] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and Lecun, Y. (2014). *Overfeat: Integrated recognition, localization and detection using convolutional networks*.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Singh et al., 2017] Singh, V., Girish, D., and Ralescu, A. L. (2017). Image understanding - a brief review of scene classification and recognition. In *MAICS Conference*.

[Solomonoff, 1964] Solomonoff, R. J. (1964). A formal theory of inductive inference. part i. *Information and control*, 7(1):1–22.

[Srivastava et al., 2014] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Su et al., 2019] Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*.

[Sudha et al., 2016] Sudha, S., Vidhyalakshmi, M., Pavithra, K., Sangeetha, K., and Swaathi, V. (2016). An automatic classification method for environment: Friendly waste segregation using deep learning. In *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, pages 65–70.

[Sun et al., 2003] Sun, W.-y., Sampaio, R. J., and Candido, M. (2003). Proximal point algorithm for minimization of dc function. *Journal of computational Mathematics*, pages 451–462.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

[Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

[TANG Li-Ming, 2014] TANG Li-Ming, TIAN Xue-Quan, H. D.-R. W. X.-F. (2014). Image segmentation model combined with fcms and variational level set. *Acta Automatica Sinica*, 40(6):1233.

[Taylor et al., 2007] Taylor, M. E., Kuhlmann, G., and Stone, P. (2007). Accelerating search with transferred heuristics. In *ICAPS-07 workshop on AI Planning and Learning*.

[Tekin et al., 2017] Tekin, B., Marquez Neila, P., Salzmann, M., and Fua, P. (2017). Learning to fuse 2D and 3D image cues for monocular body pose estimation. In *Int. Conf. on Computer Vision (ICCV)*.

[Thota et al., 2016] Thota, R., Vaswani, S., Kale, A., and Vydyanathan, N. (2016). Fast 3d salient region detection in medical images using gpus. In *Machine Intelligence and Signal Processing*, pages 11–26. Springer.

[Tikhonov and Arsenin, 1977] Tikhonov, A. N. and Arsenin, V. I. (1977). *Solutions of ill-posed problems*. Vh Winston.

[Tomasi and Manduchi, 1998] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE.

[Tome et al., 2017] Tome, D., Russell, C., and Agapito, L. (2017). Lifting from the deep: Convolutional 3D pose estimation from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Van der Waals, 1979] Van der Waals, J. D. (1979). The thermodynamic theory of capillarity under the hypothesis of a continuous variation of density. *Journal of Statistical Physics*, 20(2):200–244.

[Vese and Guyader, 2015] Vese, L. A. and Guyader, C. L. (2015). *Variational Methods in Image Processing*. Chapman & Hall/CRC.

[Wang and Yeung, 2016] Wang, H. and Yeung, D. (2016). Towards bayesian deep learning: A framework and some existing methods. *IEEE Trans. Knowl. Data Eng.*, 28(12):3395–3408.

[Wang et al., 2016] Wang, J., Deng, B., and Ren, H. (2016). Municipal solid waste classification using microwave nondestructive testing technique. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 599–603.

[Wang et al., 2015a] Wang, J., Li, S., and Jiang, H. (2015a). Salient object segmentation. US Patent 9,042,648.

[Wang and Shen, 2018] Wang, W. and Shen, J. (2018). Deep visual attention prediction. *IEEE Trans. Image Process*, 27(5):2368–2378.

[Wang et al., 2014] Wang, Y., Liu, R., Song, X., and Su, Z. (2014). Saliency detection via nonlocal l_ {0} minimization. In *Asian Conference on Computer Vision*, pages 521–535. Springer.

[Wang et al., 2015b] Wang, Y., Liu, R., Song, X., and Su, Z. (2015b). Saliency detection via nonlocal $$l_{0}$$minimization. In Cremers, D., Reid, I., Saito, H., and Yang, M.-H., editors, *Computer Vision – ACCV 2014*, pages 521–535, Cham. Springer International Publishing.

[Werbos, 1994] Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons.

[Wu et al., 2018] Wu, Y., Liu, H., Yuan, J., and Zhang, Q. (2018). Is visual saliency useful for content-based image retrieval? *Multimedia Tools and Applications*, 77(11):13983–14006.

[Yang et al., 2017] Yang, B., Zhang, X., Chen, L., Yang, H., and Gao, Z. (2017). Edge guided salient object detection. *Neurocomputing*, 221:60–71.

[Yang et al., 2009] Yang, Q., Tan, K.-H., and Ahuja, N. (2009). Real-time o (1) bilateral filtering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 557–564. IEEE.

[Zeiler and Fergus, 2013] Zeiler, M. D. and Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

[Zhan, 2011] Zhan, Y. (2011). The nonlocal-laplacian evolution for image interpolation. *Mathematical Problems in Engineering*, 2011.

[Zhang and Rudnicky, 2006] Zhang, R. and Rudnicky, A. I. (2006). A new data selection principle for semi-supervised incremental learning. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 780–783.

[Zhang et al., 2018] Zhang, X., Gao, T., and Gao, D. (2018). A new deep spatial transformer convolutional neural network for image saliency detection. *Design Automation for Embedded Systems*, pages 1–14.

[Zhao et al., 2018] Zhao, X., Wu, Y., Song, G., Li, Z., Zhang, Y., and Fan, Y. (2018). A deep learning model integrating fcnns and crfs for brain tumor segmentation. *Medical image analysis*, 43:98–111.

[Zheng et al., 2014] Zheng, H., Chen, M., Liu, W., Yang, Z., and Liang, S. (2014). Improving deep neural networks by using sparse dropout strategy. In *2014 IEEE China Summit International Conference on Signal and Information Processing (ChinaSIP)*, pages 21–26.

[Zhou et al., 2016] Zhou, X., Zhu, M., Leonardos, S., Derpanis, K. G., and Daniilidis, K. (2016). Sparseness meets deepness: 3d human pose estimation from monocular video. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 4966–4975.

[Zhu et al., 2018] Zhu, D., Luo, Y., Dai, L., Shao, X., Zhou, Q., Itti, L., and Lu, J. (2018). Salient object detection via a local and global method based on deep residual network. *Journal of Visual Communication and Image Representation*, 54:1–9.

[Zhu and Zabaras, 2018] Zhu, Y. and Zabaras, N. (2018). Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415 – 447.

CHAPTER **8**

Appendixes

8.1 Convolution as Matrix Operation

Here we give an example of how to built a matrix W from a given kernel K. Let $K \in \mathbb{R}^{3\times 3}$, $X \in \mathbb{R}^{4\times 4}$

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}, \quad X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{21,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix}.$$

The output dimension of the convolution is: $(3+4-1, 3+4-1) = (6,6)$. The vectorization of X in **x** if done by zero-padding such that the input vector dimensions fit the number of row of the ouput: $6 \times 6 = 36$

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ 0 & 0 & x_{2,1} & x_{21,2} & x_{2,3} & x_{2,4} \\ 0 & 0 & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ 0 & 0 & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix}$$

then $\mathbf{x} = X(:)$

$$\mathbf{x} = [0, \cdots, 0, x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4}, 0, \cdots, 0, x_{4,1}, x_{4,2}, x_{4,3}, x_{4,4}]^T \in \mathbb{R}^{36}$$

In the same way, the kernel K is also vectorized with zero-padding:

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & 0 & 0 & 0 \\ k_{2,1} & k_{2,2} & k_{2,3} & 0 & 0 & 0 \\ k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

this is:

$$\mathbf{k} = [k_{1,1}, k_{1,2}, k_{1,3}, 0, 0, 0, k_{2,1}, k_{2,2}, k_{2,3}, 0, 0, 0, k_{3,1}, k_{3,2}, k_{3,3}, 0, \cdots, 0]^T \in \mathbb{R}^{36}$$

Finally, the matrix W is built as follows:

$$W = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & \cdots & k_{36} \\ 0 & k_1 & k_2 & k_3 & \cdots & k_{35} \\ 0 & 0 & k_1 & k_2 & \cdots & k_{34} \\ 0 & 0 & 0 & k_1 & \cdots & k_{33} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & k_1 \end{bmatrix}$$

recovering the equivalence of equation (2.42).

8.2 Multi-class Study

As we face a multi-class problem, we want to study the by-class performance computing some of well-known metrics as True-False Positives Ratio, ROCs. We first introduce some common definitions which allow us to evaluate and compare between different methods. In our particular case, we have a partition of 7 different classes, i.e., 7 subsets

$$C = \{C_1, \ldots, C_7\}, \text{ where } C_n \subset C$$

each of them have several samples

$$C_n = \{s_1, \ldots, s_k\}$$

We define the number of samples for each class as a column vector

$$S_n = \begin{bmatrix} \partial C_1 \\ \vdots \\ \partial C_7 \end{bmatrix}$$

where ∂C_n denotes the cardinality of the subset Cn. The total number of samples is then

$$TS = \sum_n S_n$$

A multiclass confusion matrix is defined as:

$$\mathrm{M}_{i,j} = \begin{array}{c} \\ Cat_1 \\ Cat_2 \\ \vdots \\ Cat_{n-1} \\ Cat_n \end{array} \begin{array}{c} \begin{array}{ccccc} Pred_1 & Pred_2 & ... & Pred_{n-1} & Pred_n \end{array} \\ \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n-1} & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n-1} & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{n-1,1} & m_{n-1,2} & \cdots & m_{n-1,n-1} & m_{n-1,n} \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n-1} & m_{n,n} \end{bmatrix} \end{array}$$

where i, j depict *gold-label* and *prediction* respectively. Based on this multi-class confusion matrix we define usual metrics for each class.

Per-class metrics:

- True Positives (TP)

$$\text{TP}_n = diag(\text{M}) = \begin{bmatrix} tp_1 \\ \vdots \\ tp_n \end{bmatrix}$$

- True Negatives (TN)

$$\text{TN}_n = TS - S_n - \text{FP}_n = \begin{bmatrix} tn_1 \\ \vdots \\ tn_n \end{bmatrix}$$

- False Positives (FP)

$$\text{FP}_n = \left[\sum_i \text{M}_{i \neq n, n} \right]^T = \begin{bmatrix} fp_1 \\ \vdots \\ fp_n \end{bmatrix}$$

- False Negatives (FN)

$$\text{FN}_n = \left[\sum_i \text{M}_{n, i \neq n} \right]^T = S_n - \text{TP}_n = \begin{bmatrix} fn_1 \\ \vdots \\ fn_n \end{bmatrix}$$

In fact,

$$\sum_n \text{FP}_n = \sum_n \text{FN}_n = \sum_{i,j} \text{M}_{i,j} - \sum_n \text{TP}_n$$

Summarizing, the results are averaged obtaining the performance estimation for the whole system:

$$\text{TP} = \frac{1}{n} \sum_n \text{TP}_n, \quad \text{FN} = \frac{1}{n} \sum_n \text{FN}_n, \quad \text{FP} = \frac{1}{n} \sum_n \text{FP}_n, \quad \text{TN} = \frac{1}{n} \sum_n \text{TN}_n.$$