

# RETHINKING A DEEP LEARNING PIPELINE FOR IMAGES

**RICARDO PEREIRA DE MAGALHÃES CRUZ**

TESE DE DOUTORAMENTO APRESENTADA  
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM  
INFORMÁTICA



# Rethinking a Deep Learning Pipeline for Images

**Ricardo Cruz**

Supervisors: Prof. Doctor Jaime S. Cardoso  
Prof. Doctor Joaquim F. Pinto Costa

MAPi: Doctoral Programme in Computer Science  
University of Minho • University of Aveiro • University of Porto

July 22th, 2021

## **Jury President**

- Doctor Gabriel de Sousa Torcato David (University of Porto)

## **Jury Members**

- Doctor Sara C. Madeira (University of Lisbon)
- Doctor Pedro Manuel Cunha Abreu (University of Coimbra)
- Doctor Maria Beatriz Alves de Sousa Santos (University of Aveiro)
- Doctor Jaime dos Santos Cardoso (Supervisor)

Ricardo Cruz: *Re-thinking a Deep Learning Pipeline for Images*, Doctoral Thesis, Doctoral Program in Computer Science of the Universities of Minho, Aveiro and Porto, July 2021.

WEBSITE: <https://rpmcruz.github.io/>

E-MAIL: [ricardo.pdm.cruz@gmail.com](mailto:ricardo.pdm.cruz@gmail.com)



Pattern recognition methods involve several stages starting from the raw data to the prediction output. This is known as a **pipeline**. While deep learning has been hailed as greatly simplifying these stages, it still requires data pre-processing or augmentation, defining evaluation metrics and class weights, choosing the loss and optimization dynamics, deciding the architecture, and tweaking the process several times over. This thesis proposes novel ways of looking at several aspects of this pipeline with a particular emphasis on classifying and segmenting images.

Firstly, a common practice is to synthetically augment the training set by applying small transformations to the original data while keeping the original distribution intact. For example, mirroring images or small rotations are often realistic transformations that preserve the original distribution. An **automatic data augmentation** heuristic is proposed so that these transformations are learned during the optimization process. Two inferences are performed at each optimization step so that the optimal transformation is found by perturbation differences.

Several techniques may also be required to address “class imbalance”. The problem is when data are not uniformly distributed among the different classes, which biases the pattern recognition method in the direction of the majority classes. This is due to the fact that the typical learning process optimizes each datum in isolation. It is an age-old problem and learning **pairwise models for class imbalance** is proposed as a new alternative to the literature, inspired by the “learning to rank” literature.

Furthermore, serious thought was given at how errors are evaluated. For example, in medicine, a false negative is naturally considered to be worse than a false positive. The orthodox approach of specifying costs consists in using a cost matrix and imposing absolute costs in each type of errors. This cost is difficult to assign – it might be easier to think in terms of the size of the population. The proposed **risk aversion** paradigm redefines error as a tolerance on the percentage of patients that will result in a false positive, and several approaches are proposed at how learning processes could be adapted to the new paradigm.

Pattern recognition can also be fooled by variations in the background of images. If the model is evaluated with images that have a background different than the ones it has seen before, then the model will output erroneous predictions. **Adversarial background learning** is shown to produce robust models against changes of background. It consists in training a secondary model to dynamically find changes in the background of the images to teach the primary model to be more robust. This method was evaluated using indoors and outdoors photographs of insulators, motivated by an autonomous drone problem.

After the learning process, certain types of images may fail to be modeled properly because they were not well represented in the training set. These failures can then be compensated by collecting more images from the real-world and including them in the learning process, an expensive process known as “active learning”. The proposed twist, called **active supervision**, uses the model itself to change the existing images in the direction where the boundary is less-defined, requesting feedback from the user on how the new image should be classified.

Finally, traditionally, neural networks learn iteratively, but inference happens in a single step. A different architecture and training procedure is proposed so that the segmentation inference itself is also performed iteratively. The network is trained to produce a score for segmentations of different quality so that new segmentations can then be improved by gradient ascent using the score as an oracle. The advantage of this **iterative inference** is that it becomes possible to improve pre-existing segmentations; also, it makes the inference process more observable.

**Keywords:** deep learning; neural networks; computer vision; machine learning pipeline; data augmentation; class imbalance; adversarial training; background invariance; active learning; risk aversion.

Métodos de reconhecimento de padrões envolvem vários passos, começando nos dados brutos, terminando na previsão final. Este processo é conhecido como uma **pipeline**. Enquanto que o “deep learning” tem sido elogiado por simplificar muito os vários passos, requer, mesmo assim, pré-processamento dos dados ou o seu aumento artificial, definir métricas de avaliação e pesos nas classes, escolher a função “loss” e as dinâmicas de otimização, decidir a arquitetura, e afinar o processo várias vezes. Esta tese propõe formas novas de olhar para os vários aspetos da pipeline com particular destaque para a classificação e segmentação de imagens.

Em primeiro lugar, é prática comum aumentar de forma sintética o conjunto de treino, aplicando pequenas transformações nos dados originais enquanto se mantém intacta a distribuição original. Por exemplo, espelhar imagens ou pequenas rotações são normalmente transformações realistas que preservam a distribuição original. Uma heurística de **aumento de dados automático** é proposta para que estas transformações sejam aprendidas durante o processo de otimização. Duas inferências são feitas a cada passo de otimização para que a transformação ótima seja encontrada pela diferença de perturbações.

Várias técnicas poderão também ser necessárias para mitigar desequilíbrio de classes. Este problema ocorre quando os dados não são uniformemente distribuídos entre as diferentes classes, o que enviesará o método de reconhecimento de padrões na direção da classe maioritária. Este fenómeno deve-se ao facto de que o processo típico de aprendizagem otimiza cada dado de forma isolada. É um velho problema e a aprendizagem de **modelos em pares para desequilíbrio de classes** é proposta como um método alternativo à literatura, inspirado pela literatura de “learning to rank”.

Além disso, um grande esforço foi feito na forma como os erros são avaliados. Por exemplo, em medicina, um falso negativo é considerado mais severo que um falso positivo. A abordagem ortodoxa consiste em especificar uma matriz de custos e impor custos absolutos em cada tipo de erro. Este custo é difícil de atribuir – pode ser mais fácil pensar em termos do tamanho da população. O paradigma proposto de **aversão ao risco** redefine o erro como uma tolerância na percentagem de pacientes que irão resultar em falsos positivos, e várias abordagens são propostas à forma como o processo de aprendizagem pode ser adaptado ao novo paradigma.

O reconhecimento de padrões pode também ser enganado por variações no fundo das imagens. Se o modelo avaliar imagens cujo fundo seja diferente daquelas que viu anteriormente, então o modelo irá cometer erros de previsão. A **aprendizagem adversa de fundos** permite produzir modelos robustos contra mudanças de fundo. Esta consiste em treinar um modelo secundário para dinamicamente encontrar modificações do fundo das imagens que ensinem ao modelo primário a ser mais robusto. Este método foi avaliado usando fotografias de insuladores tiradas dentro e fora de portas, motivado pelo uso de drones autónomos.

Depois do processo de aprendizagem, alguns tipos de imagens poderão não ser modeladas corretamente porque não estavam bem representadas no conjunto de treino. Estas falhas podem ser compensadas recolhendo mais imagens do mundo-real e incluindo-as no processo de aprendizagem, um processo dispendioso conhecido por “active learning”. Uma

reviravolta é proposta, chamada de **active supervision**, que faz uso do próprio modelo para modificar imagens existentes na direção onde a fronteira está menos definida, pedindo feedback ao utilizador sobre como as novas imagens devem ser classificadas.

Finalmente, redes neurais tradicionais aprendem de forma iterativa, mas a inferência é feita num único passo. Uma arquitetura e treino diferentes são propostas para a inferência duma segmentação seja também ela iterativa. Uma rede é treinada para produzir uma pontuação para segmentações de diferentes qualidades para que novas segmentações possam ser melhoradas através de “gradient ascent” usando a pontuação como oráculo. A vantagem desta **inferência iterativa** é que torna possível melhorar segmentações pré-existentes; além disso, também torna o processo de inferência mais observável.

**Palavras-chave:** deep learning; redes neurais; visão computacional; pipeline de machine learning; data augmentation; class imbalance; treino adversarial; invariância de fundo; active learning; aversão ao risco.

# Acknowledgments

---

I would like to start to thank those most directly involved with this work. My main supervisor, **Professor Jaime dos Santos Cardoso**, went above and beyond his duty as a supervisor and accompanied every work here described, from the conception to the finer-details in the experiments. My co-supervisor, **Professor Joaquim Fernando Pinto da Costa**, who recommended me for Ph.D., has always found time for me and was a valuable source of constructive criticisms and discussions along these years.

This work would also not have been the same without the people with whom I collaborated. When I was starting at INESC TEC at the end of 2015, even before I started my Ph.D., **Kelwin Fernandes**, who at the time was in the middle of his Ph.D. (since then finished), made me feel welcome and taught me the tools (Python and NumPy) that I have used throughout the Ph.D. He also participated in our discussions and we collaborated together, especially on class imbalance. Professors **María Pérez Ortiz** and **Margarida Silveira** also shown interest in our class imbalance work and we had a nice collaboration. **Ricardo Prates** did a Ph.D. internship at INESC TEC during which time we had interesting conversations that led to collaborations on background invariance. **Professor AMS Shihavuddin** was my DTU contact person during my stay there and the impetus for the active supervision work.

During my time at INESC TEC, I was part of the CTM unit (Centre for Telecommunications and Multimedia), and more particularly, the VCMi group (Visual Computing and Machine Intelligence) that was a very active group in promoting workshops, both inside and outside the group. In particular, the fortnightly meetings of Ph.D. and researchers supervised by Professor Jaime Cardoso were attended by helpful and interesting colleagues, too many to name, that were sources of inspiration and it was interesting to hear about their work during our group meetings. During a couple of years of my Ph.D., we had a club at FEUP called VILab, where we participated in machine learning competitions – I would like to thank in particular to Ivo Silva and Borge Gurié who were major sources of energy and enthusiasm during those times. A special thanks to these friends for putting up with me: Renato Fernandes, Marisa Reis, Cláudio Gomes, Filipe Rocha, Filipe Oliveira, Hélder Alves, and José Devezas – all of whom researchers or Ph.D. students that touched my life one way or another.

Lastly, a special thanks to my loving girlfriend **Carla Gonçalves** who accompanied me in the ups and downs of “our” journey – *our* journey since we both had a Ph.D. to do. And thanks to my family: mother, sister, father – and our cat, Pipa.

Furthermore, I would like to thank the institutional support without which this work would not have been possible:

**FCT** During the Ph.D., I was financially supported by the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia within the Ph.D. grant “SFRH/BD-/122248/2016” supported by POCH and the EU. Furthermore, some of the work contributed to the CLARE project “POCI-01-0145-FEDER-028857”, also funded by FCT, that contributed with financial support for conference attendance.

**INESC TEC** provided me with the physical space where I performed the majority of my work – this included a workstation and meeting rooms to discuss ideas with my supervisor and have joyful meetings with fellow Ph.D. and MSc students working on similar topics. INESC TEC also provided me with health insurance and valuable IT services.

**Universities of Porto, Aveiro, and Minho** where I had classes during my first year and also hosted our yearly Ph.D. symposiums which allowed me to follow the interesting work of my Ph.D. colleagues.

**Mathematics Department of FCUP** for providing me with an office and a workstation that I used when visiting my co-supervisor.

# Contents

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>Prologue I      Introduction</b>	<b>1</b>
I.1 Thesis Organization . . . . .	2
<b>Prologue II      Background Knowledge</b>	<b>5</b>
II.1 Artificial Neural Networks . . . . .	5
II.2 Computer Vision Tasks . . . . .	6
II.3 Convolutional Neural Networks . . . . .	7
II.4 Glossary . . . . .	8
<b>Prologue III      Datasets</b>	<b>11</b>
III.1 Tabular Data . . . . .	11
III.2 Computer Vision . . . . .	13
<b>Chapter 1      Data Augmentation through Hill Climbing</b>	<b>15</b>
1.1 Related Work . . . . .	15
1.2 Proposal . . . . .	16
1.3 Experiments . . . . .	17
1.4 Summary . . . . .	20
<b>Chapter 2      Class Imbalance using Pairwise Learning</b>	<b>21</b>
2.1 Related Work . . . . .	22
2.2 Proposal . . . . .	23
2.3 Binary Case . . . . .	23
2.4 Ordinal Case . . . . .	24
2.5 Survival Analysis . . . . .	28
2.6 Experiments . . . . .	29
2.7 Summary . . . . .	31
<b>Chapter 3      Risk Aversion</b>	<b>35</b>
3.1 Related Work . . . . .	35
3.2 Proposal . . . . .	36
3.3 Experiments . . . . .	39
3.4 Summary . . . . .	41
<b>Chapter 4      Background Invariance</b>	<b>43</b>
4.1 Motivation . . . . .	44
4.2 Related Work . . . . .	45
4.3 Proposal . . . . .	46

4.4	Experiments . . . . .	48
4.5	Summary . . . . .	50
<b>Chapter 5</b>	<b>Active Supervision</b>	<b>53</b>
5.1	Related Work . . . . .	53
5.2	Proposal . . . . .	54
5.3	Experiments . . . . .	57
5.4	Summary . . . . .	58
<b>Chapter 6</b>	<b>Iterative Inference</b>	<b>59</b>
6.1	Related Work . . . . .	59
6.2	Proposal . . . . .	61
6.3	Experiments . . . . .	63
6.4	Summary . . . . .	64
<b>Epilogue IV</b>	<b>Conclusion</b>	<b>67</b>
IV.1	Summary . . . . .	67
IV.2	Production . . . . .	68
IV.3	Limitations . . . . .	68
IV.4	Future Work . . . . .	69
<b>Appendix A</b>	<b>Appendices</b>	<b>71</b>
A.1	Data Augmentation through Hill Climbing . . . . .	71
A.2	Class Imbalance using Pairwise Learning . . . . .	72
A.3	Risk Aversion . . . . .	77
A.6	Iterative Inference . . . . .	79
<b>Bibliography</b>		<b>81</b>



# List of Figures

I.1	A possible deep learning pipeline in a nutshell. . . . .	1
II.1	Example of a typical CNN used for classification. . . . .	7
II.2	Example of an U-Net network used for segmentation. . . . .	8
1.1	The proposal uses the loss as a yardstick for data augmentation. . . . .	15
1.2	Hyperparameter evolution . . . . .	17
1.3	Evolution of the rotation hyperparameter for the proposed method. . . . .	19
1.4	Hierarchical clustering of the final augmentation policies of the proposed method. . . . .	20
2.1	Illustration of the impact of class imbalance . . . . .	21
2.2	Adaptation of pairwise ranking to class imbalance data. . . . .	23
2.3	Balanced ranking training. . . . .	25
2.4	Diagram showing the lexicographical violation when decision hyperplanes intersect within the feature domain. . . . .	27
2.5	Diagram showing the F&H adaptation for balanced ordinal classification. . . . .	28
2.6	Synthetically generated non-parallel classes. . . . .	28
2.7	Ranking models can be trained to take advantage of the finer labels. . . . .	29
3.1	Comparing the current methodology to the proposed one. . . . .	36
3.2	Example of the one-class cascade approach. . . . .	37
3.3	One-class approach using a synthetic example. . . . .	38
3.4	Two-class approach using synthetic example. . . . .	38
3.5	Averse loss terms . . . . .	38
3.6	Examples of varying the risk threshold. . . . .	41
3.7	Training history, as evaluated by the validation set, for the ISBI2017 dataset. The dotted horizontal line is the desired $\rho$ . . . . .	41
4.1	Background change can produce wild disparate accuracies . . . . .	43
4.2	The insulators dataset has four materials . . . . .	44
4.3	Image generator pipeline to make insulators background-insensitive . . . . .	45
4.4	Attention mechanism diagram . . . . .	46
4.5	Proposed adversarial background augmentation during training. . . . .	47
4.6	Backgrounds introduced for MNIST and Fashion-MNIST. . . . .	48
4.7	Examples of the dynamic background along the epochs. . . . .	50
4.8	Examples of misclassifications made by the traditionally-trained classifier, but not by the adversarially-trained one. . . . .	50
4.9	Sensitivity analysis for the naive method . . . . .	50
4.10	Sensitivity analysis for the mask-inference method . . . . .	51
5.1	Active supervision short-circuits the traditional active learning pipeline. . . . .	53
5.2	The classifier is extended so that a decoder reproduced the image using the same semantic vector. . . . .	54

## List of Figures

5.3	PCA of latent space of a MNIST classifier showing the transition between each class clusters and class “2”. . . . .	56
5.4	Examples of several extrapolations for MNIST. . . . .	56
5.5	Results of active supervision using MNIST to augment digit “2”. . . . .	57
6.1	Diagram contrasting the proposal to the traditional approach. . . . .	59
6.2	Illustration of two iterative segmentation methods. . . . .	60
6.3	Dilated kernels avoid the checkerboard effect. . . . .	61
6.4	The quality measure produced by the model (oracle) guides the iterative process of segmentation inference. . . . .	61
6.5	Examples of synthetically created segmentations. . . . .	62
6.6	Oracle network architecture in detail. . . . .	63
6.7	Iterative refinement of images . . . . .	64
IV.1	Pipeline diagram. . . . .	67

# List of Tables

---

I.1	Pattern recognition waves in computer vision. . . . .	2
III.1	Tabular datasets for binary classification. . . . .	12
III.2	Tabular datasets for ordinal classification. . . . .	12
III.3	Images datasets for classification. . . . .	13
III.4	Images datasets for semantic segmentation. . . . .	14
1.1	The six transformations used in the experiments. . . . .	18
1.2	Evaluation scores . . . . .	19
2.1	Results for the binary case . . . . .	30
2.2	Results for the ordinal case . . . . .	30
2.3	Results for the survival lung cancer case study . . . . .	31
3.1	Accuracy test results using kernel density estimation for desired risk . . . .	40
3.2	Aggregated test results using the loss adaptations for desired risk . . . . .	40
4.1	General results . . . . .	49
4.2	Effect of varying the random noise rate in terms of accuracy . . . . .	49
4.3	Results for drone case study. . . . .	50
6.1	Results in terms of Dice coefficient . . . . .	63
IV.1	Paper production during the thesis, split by chapter. . . . .	68
IV.2	Summary of Applicability Limitations . . . . .	69
A.1	Classification scores in percentage using the testing set . . . . .	71
A.2	Segmentation scores in percentage using the testing set . . . . .	71
A.3	Pairwise ranking for binary imbalance. . . . .	72
A.4	Combining pairwise ranking with traditional methods for ordinal imbalance (linear SVM). . . . .	73
A.5	Combining pairwise ranking with traditional methods for ordinal imbalance (RBF-kernel SVM). . . . .	74
A.6	Combining pairwise ranking with Frank & Hall for ordinal imbalance. . . .	75
A.7	Survival analysis results across several metrics. . . . .	76
A.8	Risk aversion results for Kernel Density Estimation . . . . .	77
A.9	Risk aversion results adapting the loss . . . . .	78
A.10	Contrasting models with and without iterative forecasting . . . . .	79



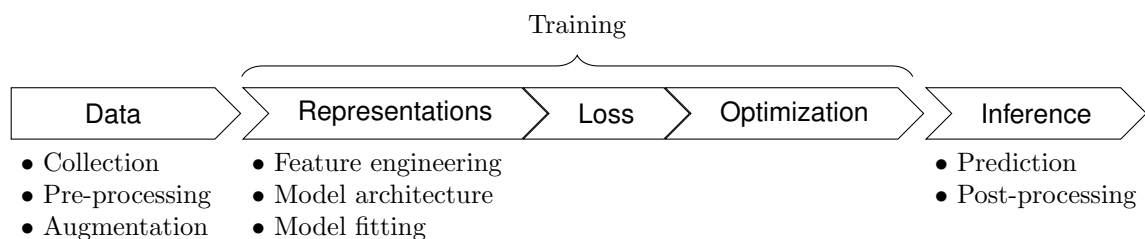
# Introduction

Pattern recognition and machine learning aim to find transformations  $f$  that convert raw data into useful knowledge, e.g.  $f(\text{cat image}) = \text{"cat"}$ . Constructing such a function  $f$  involves multiple steps, also known as a *pipeline*.

A possible representation of such a pipeline is illustrated in Figure I.1. Firstly, the *data* is ingested and possibly pre-processed (e.g. color-space conversion). The learning phase consists in converting the data into a higher-level *semantic representation*, and this can be done automatically or by feature engineering, with parts (or all) of the process *optimized* by minimizing an error function known as a *loss*. The decision process is known as *inference*.

The pattern recognition landscape has been flipped upside-down over the last few decades – especially when it comes to computer vision. Three waves are here briefly identified: rule-based systems (1<sup>st</sup> wave), classic machine learning (2<sup>nd</sup> wave), and, finally, deep learning (3<sup>rd</sup> wave). The 1<sup>st</sup> wave consisted of manually building both representations and inference using hand-coded rules. The 2<sup>nd</sup> wave also required manually extracted representations (which tried to quantify shapes and texture) but introduced models to learn the inference part of the pipeline, and a dataset with examples was required to learn. These first models did not have many learned parameters: for one output tasks, if you had  $n$  input features, then an SVM or a logistic regression would have  $n + 1$  parameters. The introduction of single-hidden layer neural networks with  $h$  neurons allowed having an arbitrary number of  $h(n + 2) + 1$  parameters – still, in practice, it was computationally infeasible to use very big values for  $h$ . The 3<sup>rd</sup> wave, known as deep learning, is based on these neural networks, but with multiple layers of neurons, and tries to learn everything end-to-end, completely driven by data.

Has this revolution reduced the need for complicated pipelines? In previous days, data practitioners would fiddle with the image color-space and manually design filters for each different task. These are all now automatically learned. On the one hand, the pre-processing step has been greatly simplified. On the other hand, the new data-driven/automatic approach is very data-hungry and so the need for pre-processing has not evaporated – data practitioners spend time tinkling with things like data augmentation



**Figure I.1:** A possible deep learning pipeline in a nutshell.

**Table I.1:** Pattern recognition waves in computer vision.

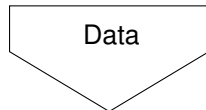
	1 <sup>st</sup> Wave Rule-based	2 <sup>nd</sup> Wave Classic	3 <sup>rd</sup> Wave Deep
<b>Data</b>	Filters to smooth the image, colorspace conversion		Data augmentation
<b>Representations</b>	Hand-crafted features e.g. HOG, LBP		Learned
<b>Loss</b>	n/a	Gini (decision tree), Huber (SVM), etc	Any $\mathcal{C}^1$ function, preferably convex
<b>Optimization</b>	n/a	Various	Gradient descent
<b>Inference</b>	Rule-based	Various	Matrix products and non- linearities

transformations, which are transformations applied to the existing dataset to produce new images that could plausibly be samples from the same distribution. Furthermore, the high malleability of these new networks means that more strenuous thought must be given to the architecture and objective functions, as well as introducing domain knowledge through such techniques as regularization. Finally, some age-old challenges such as class imbalance have been inherited from the previous family of models.

Table I.1 summarizes the aforementioned pattern recognition waves. Deep learning is described in more detail below in Section II.1.

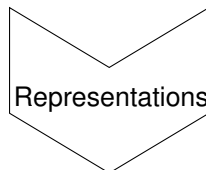
## I.1 Thesis Organization

The thesis covers several aspects and problems of the deep learning pipeline, as illustrated by Figure I.1, together with novel and creative approaches to tackle them:



Little pre-processing is applied in deep learning with one exception: datasets are commonly synthetically augmented to create new data by applying transformations on the existing data. **Chapter 1: Data Augmentation through Hill Climbing** proposes a heuristic to learn the parameters of these transformations. Publication:

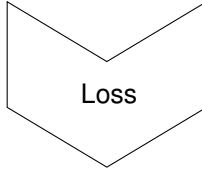
1. R. Cruz, J. F. P. Costa, and J. S. Cardoso, “Automatic Augmentation by Hill Climbing,” in *28th International Conference on Artificial Neural Networks (ICANN)*, Springer, 2019. [doi: 10.1007/978-3-030-30484-3\_10]



The way data is modeled will bias the training process. **Chapter 2: Class Imbalance using Pairwise Learning** focuses on addressing problems originating from a biased distribution of the data. Publications:

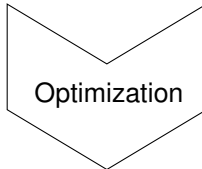
1. R. Cruz, K. Fernandes, J. S. Cardoso, and J. F. P. Costa, “Tackling Class Imbalance with Ranking,” in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016. [doi: 10.1109/IJCNN.2016.7727469]
2. R. Cruz, K. Fernandes, J. F. P. Costa, M. P. Ortiz, and J. S. Cardoso, “Ordinal Class Imbalance with Ranking,” in *Iberian Conference on Pattern Recognition, and Image Analysis (Ibpria)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-58838-4\_1]
3. R. Cruz, K. Fernandes, J. F. P. Costa, M. P. Ortiz, and J. S. Cardoso, “Combining Ranking with Traditional Methods for Ordinal Class Imbalance,” in *14th International Work-Conference on Artificial Neural Networks (IWANN)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-59147-6\_46]
4. R. Cruz, K. Fernandes, J. F. P. Costa, M. P. Ortiz, and J. S. Cardoso, “Binary ranking for ordinal class imbalance,” in *Pattern Analysis and Applications*, Springer, 2018. [doi: 10.1007/s10044-018-0705-4]

5. R. Cruz, M. Silveira, and J. S. Cardoso, “A Class Imbalance Ordinal Method for Alzheimer’s Disease Classification,” in *2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, IEEE, 2018. [doi: 10.1109/PRNI.2018.8423960]
6. M. P. Ortiz, K. Fernandes, R. Cruz, J. S. Cardoso, J. Briceño, and C. Hervás-Martínez, “Fine-to-Coarse Ranking in Ordinal and Imbalanced Domains: An Application to Liver Transplantation,” in *14th International Work-Conference on Artificial Neural Networks (IWANN)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-59147-6\_45]



The loss is the function that the learning process tries to minimize. It should be a close approximation to the evaluation metrics. **Chapter 3: Risk Aversion** proposes a new way of looking at the trade-off between false positives and false negatives and, consequently, possible losses are suggested to achieve that goal. Publications:

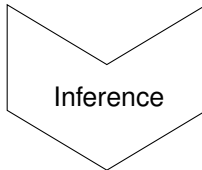
1. R. Cruz, K. Fernandes, J. F. P. Costa, and J. S. Cardoso, “Constraining Type II Error: Building Intentionally Biased Classifiers,” in *14th International Work-Conference on Artificial Neural Networks (IWANN)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-59147-6\_47]
2. R. Cruz, J. F. P. Costa, and J. S. Cardoso, “Averse Deep Semantic Segmentation,” in *41st Engineering in Medicine and Biology Conference (EMBC)*, IEEE, 2019. [doi: 10.1109/EMBC.2019.8857385]



Gradient descent is the bedrock optimization algorithm of deep learning. A couple of strategies are proposed on top of gradient descent to make learning more robust. **Chapter 4: Background Invariance** proposes a way for learning to focus on the object of interest and avoid background distractions. Publications:

1. R. M. Prates, R. Cruz, A. P. Marotta, R. P. Ramos, E. F. S. Filho, and J. S. Cardoso, “Insulator Visual Non-conformity Detection in Overhead Power Distribution Lines using Deep Learning,” in *Computer and Electrical Engineering*, Elsevier, 2019. [doi: 10.1016/j.compeleceng.2019.08.001]
2. R. Cruz, R. M. Prates, E. F. S. Filho, J. F. P. Costa, and J. S. Cardoso, “Background Invariance by Adversarial Learning,” in *25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021. [accepted]

Also, **Chapter 5: Active Supervision** introduces the human into the training loop to guide learning to be more robust. (Not published.)



Typically predictions are performed as a single forward-pass. **Chapter 6: Iterative Inference** presents a way to make predictions iteratively, by gradient ascent, with some advantages. Publication:

1. K. Fernandes, R. Cruz, and J. S. Cardoso, “Deep Image Segmentation by Quality Inference,” in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018. [doi: 10.1109/IJCNN.2018.8489696]

This division helps explain the motivation behind the work presented in the thesis. However, it should come as no surprise that the solutions presented do not fit a single box. For example, the new metrics presented as risk aversion require changes in the learning process itself.

The next chapter of the Prologue (Chapter II) provides a brief overview of neural networks with a glossary in section II.4. Then, Chapter III introduces the datasets used across the work. The meat of the work goes through Chapters 1–6. The work concludes in Chapter IV with a summary and thoughts for future work.

I made an effort at making the chapters succinct, especially regarding the experiments whenever they have already been published and are easily accessible. Expanded tables with results may also be found at Appendix A.





## Background Knowledge

---

This chapter provides a brief context for the rest of the thesis. As previously mentioned, classical computer vision involved a lot of hard-coded and specific rules for each given problem. The advent of neural networks made methods more generalized so they work well across a wider range of tasks. This renaissance in computer vision began with AlexNet, which was a deep neural network that won the ILSVRC-2012 computer vision competition, cutting by almost half the error of previous techniques [1]. All subsequent winners of the competition have used the same architecture: a convolutional neural network of several layers with rectifier-based activation functions. The term “deep learning” has since been coined to mean neural networks with many layers that learn directly from the raw data; i.e. pixels in the case of images.

### II.1 Artificial Neural Networks

A quick digression into (artificial) neural networks: it all started in 1957 when Frank Rosenblatt, a psychologist at Cornell Aeronautical Laboratory, USA, published the idea for his “brain analogue” [2], where, using modern notation, **the perceptron** was defined as the analogue of the neuron as  $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$ , where both  $\mathbf{w}$  and  $\mathbf{x}$  are vectors with  $\mathbf{x}$  being the input observation and  $\mathbf{w}$  the “weights” (parameters) which are to be determined so that the “activation function”  $\sigma$  produces a decision if the action potential  $\mathbf{w}^\top \mathbf{x}$  is greater than a given “bias” threshold  $b$  (also a parameter) or the opposite if lower. It has  $n + 1$  parameters which severely restricts what it can learn.

A **multilayer perceptron** was proposed to allow the recognition of more complex patterns in 1961. In this new formulation, the input of the  $h$ -th layer of neurons is the output of the previous  $(h-1)$ -th layer,  $f^{(h)}(\mathbf{x}) = \sigma^{(h)}\left(\sum_{i=1}^{n^{(h)}} (\mathbf{w}_i^{(h)})^\top f^{(h-1)}(\mathbf{x}) + b_i^{(h)}\right)$ , with  $f^{(0)}(\mathbf{x}) = \mathbf{x}$  and the final output is known as  $\hat{y}$ . In each layer, there are  $n^{(h)}$  neurons (weights and biases), each weight with a length of  $n^{(h-1)}$  since it is connected to each input. Several parameters must be previously defined by the user, such as the number of layers, the number of coefficients per layer ( $n^{(h)}$ ), as well as each activation function ( $\sigma^{(h)}$ ). These are known as *hyperparameters*. Typically, a single-hidden layer was used, and only the number of the neurons of that layer ( $n^{(1)}$ ) was adjusted. Not only because this simplified the hyperparameter search, but also because it has been shown that a single-hidden layer is enough for a universal approximator [3] (under certain caveats).

Hinton (1986) shows that the **backpropagation algorithm** was able to find values for the neurons that resulted in meaningful representations of the data [4]. This algorithm estimates parameters  $\mathbf{w}^{(h)}$  and  $b^{(h)}$  in two stages: (i) a forward pass is performed so the network produces its output  $\hat{y}$  (usually these parameters are initialized at random for the first iteration), and then (ii) the error is given by a loss function, such as  $\mathcal{L} = (y - \hat{y})^2$ , and is propagated back so that each layer gets updated by  $\mathbf{w}_i^{(h)} \leftarrow \mathbf{w}_i^{(h)} - \eta \partial \mathcal{L} / \partial \mathbf{w}_i^{(h)}$ , where  $\eta$  is a hyperparameter provided by the user and is known as the “learning rate”.

Several improvements exist on top of this basic algorithm, usually involving adding a “momentum” where the learning rate is an acceleration rather than a velocity [5].

Typically, the logistic function was used as the activation function, i.e.  $\sigma(a) = 1/(1 + e^{-a})$ . A new jump takes place in 2000 with the **ReLU** as an activation function of the form  $\sigma(a) = \max(0, a)$ , which helped subdue the vanishing gradient problem. The vanishing gradient problem happens when the update derivate,  $\partial\mathcal{L}/\partial\mathbf{w}_i^{(h)}$ , in the context of the backpropagation algorithm, becomes too small as it traverses back the network, because of the non-linearities introduced by the logistic and tanh activation functions since their derivatives approach zero as  $a$  becomes very large in magnitude while ReLU’s derivative is a constant [6]. Furthermore, the ReLU activation function produces sparser and faster models.

Finally, LeCun’s **Convolutional Neural Network** (CNN), originally known as LeNet, was first published in 1998 [7] and was shown in 2010 to surpass the performance of traditional computer vision models. This architecture greatly reduced the number of parameters of a neural network, which, in turn, made it possible to build larger and more powerful neural networks. These neural networks reduce the number of parameters by operating over patches so that each node is connected only to a neighborhood of pixels, rather than the entire an image. By analogy, what these neurons are performing is a *convolution* across the image. The ensuing activation maps outputted by each layer can themselves be seen as intermediate images, which are passed along to the next layer. In between these operations, the maps’ resolution can be reduced by such operations as *max-pooling*, i.e. selecting values with the highest intensity, or by applying convolutions with jumping strides. By reducing the image, each following convolution is now using information from farther away, i.e. the receptive field is enlarged [5].

These same networks used for computer vision have been exported and widely deployed in other fields, such as DNA pattern recognition, speech recognition, and even to process human language, among other uses [8], and have also been used on the recent AlphaGo victory to compute the pruning heuristic [9].

Nowadays, several frameworks are available to automatize the process so that the user needs only to define the forward pass and the backward pass (differentiation) is automatically symbolically determined. Some of this work was done during a time these frameworks were unripe and therefore some networks were implemented directly in C++, but most of the work uses TensorFlow which was first publicly released by the end of 2015.

## II.2 Computer Vision Tasks

Computer vision consists of interpreting images into higher-level semantics in order to perform certain tasks. An image is represented by a tensor  $\mathbb{R}^{h \times w \times c}$ , where  $w$  and  $h$  are the width and height dimensions, and  $c$  is the number of colors (channels). Two computer visions tasks are performed during the thesis – *classification* and *segmentation*. Classification concerns with assigning a probability to an entire image (e.g. classifying an image as a dog or a cat), while segmentation concerns with assigning probabilities to each pixel in the image of belonging to the object of interest.

- **Classification** consists in producing a function  $f$  that produces a discrete category of  $K$  classes from an image,  $f: \mathbb{R}^{h \times w \times c} \rightarrow \{1, \dots, K\}$  (e.g.  $f(\img{cat}) = \text{“cat”}$ ). The most common metric for classification is accuracy:

$$\text{Acc}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(y_i = \hat{y}_i), \quad (\text{II.1})$$

where  $\mathbb{1}$  is the indicator function and  $N$  the number of observations.

- **Segmentation** consists in producing a function  $f$  that generates a probability maps  $f: \mathbb{R}^{h \times w \times c} \rightarrow \{0, \dots, K-1\}^{h \times w}$  to highlight the relevant parts of the image (e.g.  $f(\text{image}) = \text{mask}$  for binary segmentation). Two popular evaluation metrics for binary segmentation ( $K = 2$ ) are:

$$\text{Jaccard index}(\mathbf{y}, \hat{\mathbf{y}}) = J(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{Nwh} \sum_{i=1}^N \sum_{j=1}^h \sum_{k=1}^w \frac{y_{ijk} \hat{y}_{ijk}}{y_{ijk} + \hat{y}_{ijk} - y_{ijk} \hat{y}_{ijk}} \quad (\text{II.2})$$

$$\text{Dice coefficient}(\mathbf{y}, \hat{\mathbf{y}}) = D(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{Nwhc} \sum_{i=1}^N \sum_{j=1}^h \sum_{k=1}^w \frac{2y_{ijk} \hat{y}_{ijk}}{y_{ijk} + \hat{y}_{ijk}}. \quad (\text{II.3})$$

Both use the product between real and predicted segmentations as the numerator to represent intersection with slight variations of the denominator. In all cases, the score has a range of  $[0, 1]$  with 1 being a perfect segmentation.

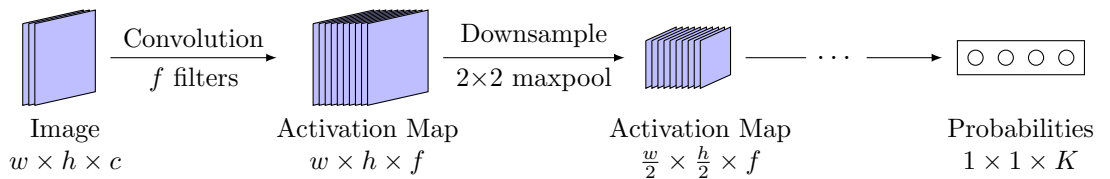
When it comes to segmentation, experiments throughout the thesis focus on (i) *binary* segmentation and, in particular, (ii) *semantic* segmentation. Binary segmentation regards distinguishing between only two classes ( $K = 2$ ), typically to distinguish between foreground and background. There is no loss of generality since using multiple classes is possible by simply predicting multiple binary segmentations. Also, we focus on semantic segmentation where the goal is to discriminate the different categories of objects, while instance segmentation would try to distinguish between individual objects.

## II.3 Convolutional Neural Networks

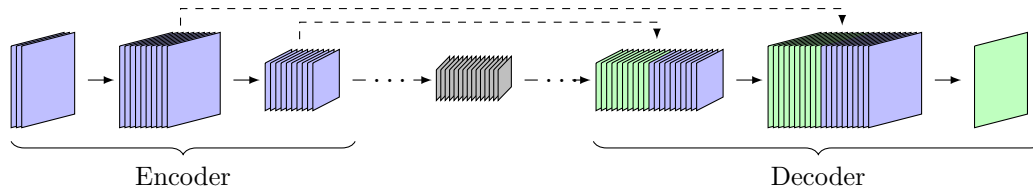
Several architectures of neural networks exist. When it comes to images, the most popular is the convolutional neural network (CNN), originally developed by LeCun in 1998 [7]. It has become popular even in other fields whenever the data can be arranged as a tensor (vector, matrices, etc) and local neighborhoods are important.

In a CNN, each neuron is connected locally to the neighbor pixels rather than to all pixels of the entire image like in multilayer perceptrons networks which is computationally infeasible for big images. Typically, each neuron would be connected to the  $3 \times 3$  neighbor pixels, therefore requiring nine weights plus the bias (ten parameters) – each one of these parameter arrangements is called a *filter*. A convolution consists of sliding each one of these filters across the image producing a *feature map*, i.e. one feature per pixel.

When it comes to classification, the input is the image – which has one feature (if monochrome) or three features (if color) – and each convolution layer applies many filters to increase the number of features. After each convolution, the image size is reduced by either explicitly downsampling it (e.g. using max-pooling which consists of choosing the



**Figure II.1:** Example of a typical CNN used for classification.



**Figure II.2:** Example of an U-Net network used for segmentation.

maximum of each  $2 \times 2$  neighborhood) or by implicitly downsampling it (e.g. by sliding the convolution in strides of two pixels). This reduction in size is important because it means that later filters are indirectly connected to farther away pixels, as the depth of focus increases. These layers are repeated several times over, successively reducing the breadth of the image, but increasing its depth, as illustrated in Figure II.1. Often, the final layers are fully-connected, and at the end, the initial image, which is a 2D or 3D tensor, is converted into a 1D vector of length  $K$ . Since the model is fully differentiable, if the loss function  $\mathcal{L}$  is also differentiable, then the weights  $\mathbf{w}$  are optimized iteratively by gradient descent to minimize the loss, i.e.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\partial \mathcal{L}}{\partial \mathbf{w}}. \quad (\text{II.4})$$

One of the most popular architecture for segmentation is U-Net [10], as illustrated in Figure II.2. This network is composed of two almost-symmetrical halves. In the *first half*, the image, with all its three (R,G,B) features (also known as channels), is initially expanded in terms of features (by the convolutions) and reduced in terms of breadth by either pooling layers or convolutional strides, the same process as discussed for classification. The *second half* does the reverse process by using as mechanisms upsampling operators either by linear interpolation of the outputs of the convolutions or by performing an inverse convolution (also known as transpose convolution), while the number of nodes in the convolutions is gradually reduced. Furthermore, each layer in the second half receives as input both the result of the previous layer, as well as the corresponding layer from the first half. They are known as “skip-layers” and these techniques help avoid losing minutia and help produce more robust gradients during the learning process. The rationale for this is later elaborated on Chapter 6. This causes a slight asymmetry where the neurons in the second phase receive twice the inputs of their counterparts.

## II.4 Glossary

**Activation function** A non-linearity introduced in neural networks at the output of each *neuron*. The most popular nowadays is ReLU. The function for the last *neurons* depends on the task: sigmoid and softmax are typically used when predicting probabilities.

**Convolutional Neural Network (CNN)** Model for grid-based data consisting of convolving layers composed of *filters*.

**Filter** A *neuron* that is connected to a local neighborhood of pixels, typically  $3 \times 3$ .

**Kernel** Another name for *filter*.

**Loss** The error function that the optimization process tries to minimize by adjusting the *neurons*’ weights. For classification, a popular loss function is cross-entropy.

**Max-pooling** Downsampling technique that takes the maximum value within a neighborhood of pixels, typically  $2 \times 2$ .

**Multilayer perceptron** Classical neural network where each *neuron* within a layer is connected to all outputs of the previous layer.

**Neuron** Processes a set of inputs by multiplying each one by a weight (linear transformation) and then applying an *activation function* to the output (non-linear transformation).



Several datasets are used along the thesis – they are here listed for easy reference. Most work uses images, but some work also uses tabular data. Datasets here presented are divided by the different tasks for which they are used.

## III.1 Tabular Data

### III.1.1 Binary Classification

Table III.1 lists the sixteen datasets considered for binary classification, with most coming from the “UCI machine learning repository” (**Ref.** column). Some of these were multiclass and were converted to binary using the classes mentioned on the **Minority** column.

The datasets were selected to represent varying degrees of class imbalance, as measured by a metric we defined as “Imbalance Ratio” (**IR**):  $IR = 2N_1/N$ , with  $IR \in [0, 1]$ , ranging from very imbalance ( $\ll 1$ ) to perfectly balanced (1). The number of features is represented in the **#vars** column and the number of samples by  $N$  with the number of minority samples being  $N_1$ . As conventionally done, and without loss of generality, class 1 (positive) is the minority class and class 0 (negative) is the majority class.  $N$  is typically in the order of thousands to ensure that what is being tested is “relative rarity” and avoid “absolute rarity” issues [15]. In this and all proceeding tables, datasets are ordered by IR to help emphasize the role of imbalance metrics.

Furthermore, an Overlap Ratio (**OR**) column is presented as a measurement of how intertwined the observations from the two classes are. There is a big amount of literature on the role of overlapping in class imbalance [16], with some authors arguing these problems are often conflated [17]. As an overlap metric (OR), we propose defining OR as the ratio of minority observations whose closest neighbor is an observation of the majority class.

### III.1.2 Ordinal Classification

Datasets from Table III.2 were mostly re-used from [19], including the same 30 folds for easy comparison. The Imbalance Ratio (**IR**) metric is now represented by  $IR = K \frac{\min_k N_k}{N}$ , i.e. the proportion of the minority class in the data scaled to  $[0, 1]$ . Some authors have used the arithmetic average of the per-class imbalance ratios, but we thought that was harder to interpret. Again, the table is ordered by IR. The variable  $K$  represents the number of classes in the dataset, while  $N_k$  is the number of samples for class  $k$ .

The metric Overlap Ratio (**OR**) is here formalized and extended to the ordinal case using the ratio of observations having an observation of another class as its nearest neighbor, using the normalized Euclidean distance,

$$OR = \frac{1}{N} \sum_{i=1}^N \mathbb{1} \left( y_i \neq y_j \text{ with } j = \arg \min_{k, k \neq i} (\|\mathbf{z}_i - \mathbf{z}_k\|_2) \right), \quad (\text{III.1})$$

**Table III.1:** Tabular datasets for binary classification.

Dataset	Ref.	Minority	N	#vars	IR% ↓	OR%
sonar	[11]	—	208	60	93.2	22
heart	[11]	—	270	13	88.8	38
breast-cancer	[12]	—	699	9	69.0	8
german	[11]	—	1000	24	60.0	56
haberman	[11]	—	306	3	53.0	61
transfusion	[11]	—	748	4	47.6	63
vehicle	[13]	van	846	18	47.0	9
CTG	[11]	—	2126	22	44.4	17
hepatitis	[11]	—	143	14	40.6	62
segment	[11]	1	2310	19	28.6	1
winequality-red	[14]	7,8	1599	11	27.2	51
vowel	[11]	1	990	13	18.2	1
abalone	[11]	9vs18	731	7	11.4	60
glass	[11]	6	214	9	8.4	56
car	[11]	good	1728	6	8.0	67
yeast	[11]	ME1	1484	8	6.0	34

**Table III.2:** Tabular datasets for ordinal classification.

Dataset	Ref.	N	#vars	K	IR% ↓	OR%
pyrim5	[18]	74	27	5	100	62
pyrim10	[18]	74	27	10	100	84
stock10	[18]	950	9	10	100	24
abalone5	[11]	4177	10	5	100	60
abalone10	[11]	4177	10	10	100	78
wisconsin5	[11]	194	32	5	100	74
wisconsin10	[11]	194	32	10	100	85
machine10	[11]	209	6	10	100	65
machine5	[11]	209	6	5	100	51
auto5	[11]	392	7	5	73.0	45
newthyroid	[11]	215	5	3	68.5	7
cooling	[18]	768	8	8	52.8	52
toy	[11]	300	2	5	51.0	12
contact-lenses	[11]	24	6	3	50.1	44
squash-stored	[11]	52	51	3	46.2	56
diabetes5	[18]	43	2	5	45.5	52
triazines5	[18]	186	60	5	40.5	52
triazines10	[18]	186	60	10	40.0	52
auto10	[11]	392	7	10	37.7	62
balance-scale	[11]	625	4	3	23.1	25
squash-unstored	[11]	52	52	3	23.1	23
diabetes10	[18]	43	2	10	16.7	52
car	[11]	1728	21	4	15.2	30
ESL12vs3vs456vs7vs89	[11]	488	4	5	15.0	25
LEV	[11]	1000	4	5	14.0	70
SWD	[11]	1000	10	4	12.8	61
ERA1vs23456vs7vs8vs9	[11]	1000	4	5	9.5	58
ERA	[11]	1000	4	9	7.6	88
ESL	[11]	488	4	9	4.5	48



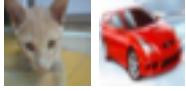
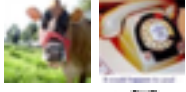

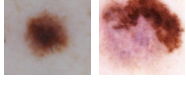




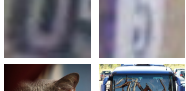
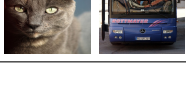
where  $\mathbf{z}$  is the normalized (using z-score)  $\mathbf{x}$ . Again, this metric was motivated to help testing the hypothesis advanced by some authors that class imbalance performance issues are due to overlapping problems [17].

## III.2 Computer Vision

These are datasets of images. An image is represented by a tensor  $\mathbb{R}^{w \times h \times c}$ , where the value  $c = 1$  represents a grayscale image, whereas an RGB color image uses  $c = 3$ . The images used throughout this work are square, i.e.  $w = h$ . In many datasets, the size of each image was different, therefore these datasets were normalized to sizes of  $128 \times 128$ .

Tables III.3 and III.4 represent the two tasks performed during the thesis: classification and semantic segmentation. A primer on semantic segmentation is provided in Section II.2, and so are variables defined.

**Table III.3:** Images datasets for classification.



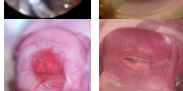
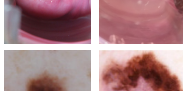
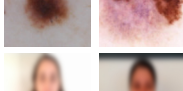
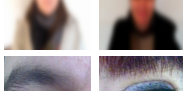
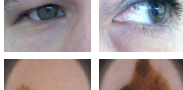
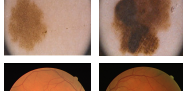
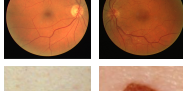












Dataset	Ref.	w	c	N	K	Seg	Examples
CIFAR-10	[20]	32	3	60k	10	—	
CIFAR-100	[20]	32	3	60k	100	—	
Fashion-MNIST	[21]	28	1	70k	10	—	
ISIC 2017	[22]	128	3	2750	3	*	
MNIST	[23]	28	1	70k	10	—	
PH2	[24]	128	3	200	3	*	
SMARTSKINS	[25]	128	3	292	3	*	
STL10	[26]	96	3	13k	10	—	
SVHN	[27]	32	3	≈99k	10	—	
PASCAL VOC 2012	[28]	128	3	≈12k	20	*	

\* These datasets can also be used for segmentation tasks.

The datasets were partitioned in 60-20-20 train-val-test partitions, or the original partitioning scheme was used when provided. The column %Fg shows the average percentage of foreground (positive) values relative to the entire image – notice that class imbalance

can also be here a problem.

**Table III.4:** Images datasets for semantic segmentation.

Dataset	Ref.	w	c	N	%Fg	Examples	
Breast-Aesthetics	[29]	128	3	120	19.1		
Cervix-HUC	[30]	128	3	261	5.8		
Cervix-MobileODT	[31]	128	3	1503	17.1		
ISIC 2017	[22]	128	3	2750	9.3		
MobBIO faces	[32]	128	3	2164	23.1		
MobBIO iris	[32]	128	3	2164	20.8		
PH2	[24]	128	3	200	49.1		
Retina vessels *	[33, 34, 35]	128	3	88	2.6		
SMARTSKINS	[25]	128	3	292	37.5		
Teeth-UCV	[36]	128	3	98	23.7		
PASCAL VOC 2012	[28]	128	3	≈12k	5.1		

\* Composition of multiple datasets.

# Data Augmentation through Hill Climbing

Data augmentation is the process of syntactically creating plausible new observations that could come from the source. Data augmentation has become a staple of deep learning, in particular when it comes to computer vision [1]. Transformations such as rotation or shear are applied to create new images from existing images.

Unfortunately, it is not always intuitive for the user how much shear or translation to apply. For this reason, training multiple models through a hyperparameter search is required to find the best augmentation policies. But these methods are computationally expensive. Furthermore, since they generate static policies, they do not take advantage of smoothly introducing more aggressive augmentation transformations.

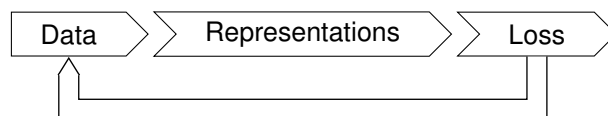
Our proposal: repeating each epoch twice with a small difference in data augmentation intensity, using the validation loss difference to guide data augmentation. This optimization process can be seen as a type of *hill climbing* on the loss, as illustrated in Figure 1.1. This slow increase in the amount of augmentation during training has also the added benefit of gradually increasing the difficulty of the observations, which typically adds a process known as curriculum learning [37]. This process doubles the number of epochs but avoids having to train multiple models. The method is compared against random and Bayesian search.

## 1.1 Related Work

A hyperparameter is any parameter that cannot be estimated by the normal estimation process of the model; this includes such disparate things as the learning rate, the size of the neural network, or, in our case, how much rotation or shear to apply during data augmentation.

Several search heuristics exist to navigate the hyperparameter search space that can be used as baselines for automatic augmentation. These techniques involve training many models to find the best hyperparameter(s)  $\theta^* = \arg \max_{\theta} s(f_{\theta}(X^{\text{val}}))$  such that a metric function  $s$  is maximized when a surrogate model  $f_{\theta}$ , trained with  $\theta$  augmentations, is evaluated using validation data  $X^{\text{val}}$ . Since the effect of the hyperparameters is not independent, the problem becomes combinatorial.

Given a budget  $B$  of how many surrogate models to train, the problem becomes how best to sample a user-defined range  $\theta \in [\underline{\theta}, \bar{\theta}]$ . All the existing hyperparameter search



**Figure 1.1:** The proposal uses the loss as a yardstick for data augmentation.

methods consist in suggesting different sampling functions  $\theta_i \sim \mathcal{F}$  for each surrogate model  $i$ ,  $1 \leq i \leq B$ .

- In **grid search**, the search space is sampled uniformly from  $\underline{\theta}$  to  $\bar{\theta}$  in increments of  $\underline{\theta} + \frac{i-1}{B}(\bar{\theta} - \underline{\theta})$ .
- Another common approach is **random search**, which samples of an uniform distribution,  $\mathcal{F} = \mathcal{U}(\underline{\theta}, \bar{\theta})$ . It has been found to produce better results for a smaller  $B$  [38].

Other techniques exist that focus on the most promising parts of the search space.

- **Bayesian optimization** samples from the posterior distribution to best solve the exploration-exploitation trade-off problem involved. This distribution is generally modeled using a Gaussian Process, and an acquisition function chooses the next point to sample based on an expectation/variance combination (exploitation/exploration) [39].
- **Successive halving** uses (i) random search to train each model for a few epochs and then (ii) discards the worst-half performing models, repeating (i)–(ii) ad nauseam until only one model is left [40].
- **AutoAugment** uses an RNN as its  $\mathcal{F}$  distribution function [41], so that  $\mathcal{F}$  evolves in time.
- **Evolutionary algorithms** have also been used for hyperparameter search [42].

Less research exists in optimizing hyperparameters during the training process itself. Gradient-based algorithms do exist that allow minimizing a validation loss for particular problems, such as L2 regularization [43].

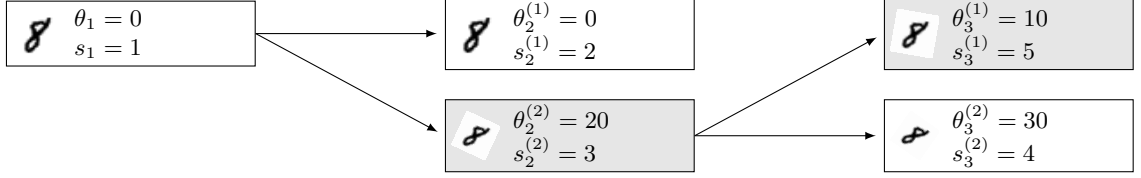
After a vector  $\theta$  is found, it is known as a *policy*. Most work find a single  $\theta$  which specifies a limit on the augmentation; for example, if  $\theta = 30$  for rotation, then, for each image, rotation is applied randomly using  $\mathcal{U}(-30, 30)$ . This is how our experimental section will work. Other work find two hyperparameters for each transformation: the probability of the transformation being applied and its absolute magnitude [41].

## 1.2 Proposal

Our proposal consists in starting with no augmentation ( $\theta_1 = 0$ ) and gradually make it more aggressive ( $\theta_{t+1} > \theta_t$ ). The index refers to the iteration (or epoch). Notice that only a single model is being used. At each epoch  $t$ , the proposal is to perform the epoch twice, for  $\theta_{t-1} - \varepsilon$  and  $\theta_{t-1} + \varepsilon$ , so that the impact of a small perturbation  $\varepsilon$  can be inferred using finite differences on the validation set. The procedure consists in the following four steps:

- Step 1.** Model  $f$  is trained for one (or more) epoch(s) without augmentation, obtaining weights  $\mathbf{w}_1$ .
- Step 2.** The weights are then forked in two,  $\mathbf{w}_{t+1}^{(1)}$  and  $\mathbf{w}_{t+1}^{(2)}$ , which are obtained by minimizing the loss  $\mathcal{L}$  for one epoch using the training set  $X^{\text{tr}}(\theta_t)$ , augmented by vector  $\theta_t$ , and labels  $y^{\text{tr}}$ ,

$$\mathbf{w}_{t+1}^{(1)} = \arg \min_{\mathbf{w}} \mathcal{L}(X^{\text{tr}}(\theta_t - \Delta), y^{\text{tr}} | \mathbf{w}_t)$$



**Figure 1.2:** Hyperparameter evolution where  $\theta$  controls the rotation and  $s$  is the validation score.

$$\mathbf{w}_{t+1}^{(2)} = \arg \min_{\mathbf{w}} \mathcal{L}(X^{\text{tr}}(\boldsymbol{\theta}_t + \boldsymbol{\Delta}), y^{\text{tr}} | \mathbf{w}_t).$$

$\boldsymbol{\Delta}$  is a vector which is zero for all values except for a single one  $j$ , for which  $\Delta_j = \varepsilon$ . This  $j$  is chosen randomly in this work. This hyperparameter  $j$  is the one that is being tested.

**Step 3.** The models are then evaluated and compared

$$\delta = s(f(X^{\text{val}} | \mathbf{w}_{t+1}^{(2)}), y^{\text{val}}) - s(f(X^{\text{val}} | \mathbf{w}_{t+1}^{(1)}), y^{\text{val}}),$$

so that

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \boldsymbol{\Delta} \text{ and } \mathbf{w}_{t+1} = \mathbf{w}_{t+1}^{(1)} & \text{if } \delta < 0, \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \boldsymbol{\Delta} \text{ and } \mathbf{w}_{t+1} = \mathbf{w}_{t+1}^{(2)} & \text{if } \delta > 0. \end{aligned}$$

Possible ties ( $\delta = 0$ ) are solved by using the validation loss.

**Step 4.** Go back to **Step 2**.

The procedure is illustrated in Figure 1.2. The hyperparameter  $\theta$  controls the range with how aggressive augmentation is applied, it controls the probability distribution of how aggressive the augmentation will be. For example, a rotation of  $\theta = 30$  means that the rotation of each image will be chosen randomly from  $\mathcal{U}(-30, 30)$ .

The augmentation techniques that have been used are the six transformations provided by the Keras Pre-processing toolkit<sup>1</sup> – rotation, x/y translation, shear, zoom in/out, channel shift (add a constant to each layer), and brightness (multiply each layer by a constant). The possible range of values for each hyperparameter  $\theta$  is defined differently per transformations (for example, rotations are naturally bounded within the range  $[0, 180]$ ), as detailed in Table 1.1.

## 1.3 Experiments

Two different types of tasks were experimented with: classification and semantic segmentation, using several datasets from Tables III.3 and III.4.





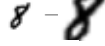


Segmentation is an interesting problem to consider since the image and the binary segmentation must synchronously suffer from the same augmentation. Naturally, no brightness or channel shift is applied to the segmentation.

**Methods:** The methods used in the experiments were:

- **Baseline:** no data augmentation;
- **Random search;**

<sup>1</sup><https://github.com/keras-team/keras-preprocessing>

**Table 1.1:** The six transformations used in the experiments.

Hyperparameter	Units	$\underline{\theta}$	$\bar{\theta}$	Example
none	—	—	—	
Rotation	degrees	0	180	
Translation x/y	pixels	0	width	
Shear	degrees	0	60	
Zoom in/out	factor	1/3	3	
Channel shift	value	0	50	
Brightness	value	0.5	2	

- **Bayesian search** using a Gaussian Process with an RBF kernel, with a seed of 10 random models;
- **Proposal** as detailed in the previous section;
- **Proposal\***: a static version of the proposal. This method takes the last policy found by the proposed method and applies that policy to a new model. The purpose is to evaluate how much of the gains from the proposed method are accrued from the incremental nature of the method.

**Metrics and loss:** The metrics used were accuracy for classification (II.1) and the Jaccard index for segmentation (II.2). In both cases, the loss used was cross-entropy weighted by the inverse frequency of each class because of class imbalance.

**Architecture:** The architecture used for classification was a CNN intertwined with convolutions and max-pooling blocks, while a U-Net [10] was used for segmentation.

The classification neural network is made of convolution and maxpooling layers, as explained in section II.3, reducing the activation maps by halves until it reaches about  $6 \times 6$ , then dense layers are applied until it outputs a probability of each class  $K$ , i.e.

$$\underbrace{w \times w \times c \rightarrow \dots \rightarrow \approx 6 \times 6 \times 32}_{\text{Encoder}} \rightarrow \approx 1152 \rightarrow 32 \rightarrow K. \quad (1.1)$$

For semantic segmentation, an U-Net architecture was used (as mentioned in section II.3), which is composed of an encoding and a decoding phase. The encoding phase used is the same as the one highlighted in (1.1), and the decoding phase is also the same but in reverse, with linear up-sampling being used instead of max-pooling to double the activation map. Skip-layers are used to connect the first convolutional layer with the last, the second with the penultimate, and so forth, like in U-Net.

**Optimization:** We tested our **proposal** together with no augmentation (**none**) trained for 250 epochs, each epoch augmenting a total of 1,024 images in batches of 128 images.

**Results:** Table 1.2 considers results for 10 datasets for each classification and segmentation tasks, showing the average metric for these 10 datasets, as well as, how many times (percentage-wise) each method ranked on the top-1 and top-2 of the best method for each dataset. For an extended version of the table, consult Appendix A.1. Please notice that top-1 may exceed 100% due to ties.

**Table 1.2:** Evaluation scores (higher is better).

Classification (%) (accuracy)					
	Baseline	Random	Bayesian	Proposal	Proposal*
<b>Average</b>	52.6	49.6	58.7	<b>58.9</b>	55.3
<b>% Top-1</b>	0	0	40	<b>50</b>	10
<b>% Top-2</b>	0	20	60	<b>90</b>	40

---

Semantic Segmentation (%) (Jaccard index)					
	Baseline	Random	Bayesian	Proposal	Proposal*
<b>Average</b>	88.2	88.4	89.8	<b>90.0</b>	89.5
<b>% Top-1</b>	10	10	30	<b>40</b>	20
<b>% Top-2</b>	30	30	50	<b>70</b>	40

Firstly, it is interesting to note that gains from data augmentation were much greater for classification than from segmentation. For each dataset, if we contrast the best performing method against applying no augmentation (baseline), then the average relative gains are of 32% and 3% for classification and segmentation, respectively. Interestingly, the baseline ranked three times in top-2 for segmentation.

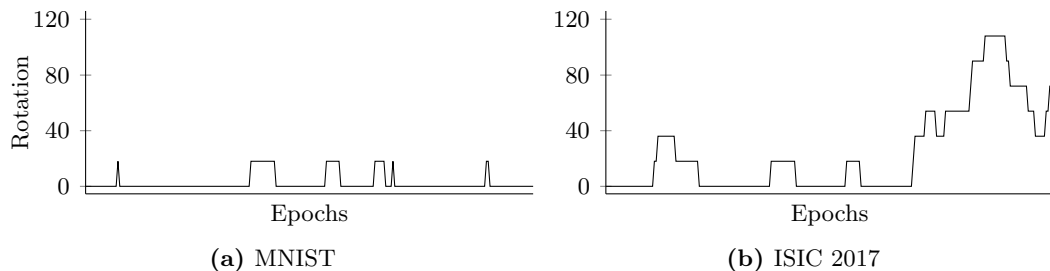
Secondly, the proposal performed better in the vast majority of cases – top-2 was 90% and 70% of the cases for classification and segmentation, respectively. The poor performance of the other methods may be explained by the vast search space from having 7 transformations, and the fact these methods treat the search space equally, but there is a bias towards conservative augmentation (centered in zero) performing better. Random search models performed worse than the baseline in many cases.

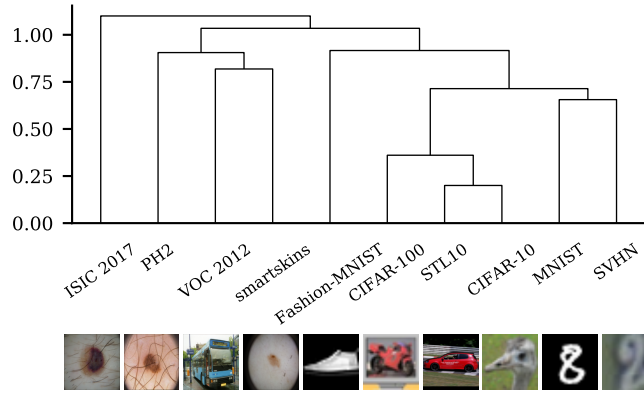
Furthermore, proposal\* can be contrasted with the proposal column to understand whether the dynamic nature of the method contributed to its performance, as suggested by the curriculum learning literature [37]. In most cases, the results are not significantly different. However, this static version was 60% worse than the dynamic version.

In terms of training time, while the proposal would be expected to double the training time relative to the baseline, it increased training time by around 5.1 times since weights must be copied around; this context switch requires extra disk-gpu interaction, which is something that could be improved.

The hyperparameters evolved more or less stably as can be seen in the rotation examples from Figure 1.3. It makes sense that rotation is not particularly useful when it comes to digit recognition. On the other hand, skin lesions do benefit from rotation.

A dendrogram (Figure 1.4) was built to find out how similar the policies produced by the proposed method, at the end of  $t = 250$  epochs, were between the datasets used.

**Figure 1.3:** Evolution of the *rotation* hyperparameter for the proposed method.



**Figure 1.4:** Hierarchical clustering of the final augmentation policies of the proposed method.

Euclidean distance is used between the hyperparameters  $\theta$ . It would be expected that the policies would be similar for similar datasets. Indeed, this happens: all melanoma-related datasets (on the left side of the dendrogram) had similar augmentations and formed a cluster. Fashion-MNIST and VOC 2012 relate miscellaneous classes and are distanced similarly, as does MNIST and SVHN which relate numbers.

## 1.4 Summary

This chapter was motivated out of frustration from finding sensible values for such transformations as elastic deformations. In the beginning, several multi-armed bandits algorithms were considered. However, these algorithms are only well developed for discrete decisions within stochastic environments, and therefore this simple algorithm was published as:

1. R. Cruz, J. F. P. Costa, and J. S. Cardoso, “Automatic Augmentation by Hill Climbing,” in *28th International Conference on Artificial Neural Networks (ICANN)*, Springer, 2019. [doi: 10.1007/978-3-030-30484-3\_10]

As future work, several details could be improved. The index  $j$ , which was here chosen randomly, could be chosen using a multi-armed bandit heuristic. The timing of when to make augmentation more aggressive could be based on the loss plateauing, rather than the end of each epoch. Furthermore, more augmentation could have been evaluated, especially elastic deformations which are particularly tricky for the user to define since they can involve multiple parameters.



## Class Imbalance using Pairwise Learning

In certain classification domains, *class imbalance* is pervasive. That is, the class distribution is not uniform, in some cases in an extreme fashion. In medicine, in particular, there is a broad dispersion of patients of differing disease severity; it is also inherent in fraud and fault detection where the anomaly is rare. The minority class contributes too little to the decision boundary because the learning process learns from each observation in isolation.

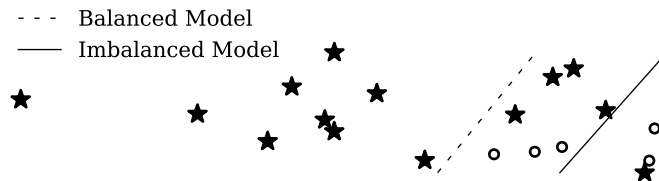
Over-populated classes (denoted as majority classes) can exert undue influence in the decision boundary, as illustrated in Figure 2.1. A naive classifier would have high accuracy by focusing on the majority classes but have no discriminatory power. Special metrics have been designed to evaluate the classifier and try to ensure it is unbiased.

As mentioned in the Introduction (Section II.2), the most popular metric for classification is accuracy,

$$\text{Acc}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^N \mathbb{1}(y_i = \hat{y}_i). \quad (2.1)$$

However, accuracy is biased towards the majority class since it is an unweighted average. Different metrics are used in cases of imbalance. For binary classification, where  $K = 2$  with labels defined as  $y \in \{0, 1\}$  (also called negative and positive, respectively), popular metrics are  $F_1$  and G-mean which correspond to the harmonic and geometric averages of precision and recall, respectively, and are insensitive to class frequency. Precision and recall are two popular metrics related to the two types of errors: false positives (FP) and false negatives (FN), respectively; that is, Precision =  $\frac{\text{TP}}{\text{TP} + \text{FP}}$  and Recall =  $\frac{\text{TP}}{\text{TP} + \text{FN}}$  with TP being the true positives. In this chapter, the focus is on  $F_1$  which can be written as

$$F_1(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^N \frac{2y_i\hat{y}_i}{y_i\hat{y}_i + y_i(1 - \hat{y}_i) + (1 - y_i)\hat{y}_i}. \quad (2.2)$$



**Figure 2.1:** Illustration of the impact of class imbalance on two linear SVMs; the balanced version uses class weights inversely proportional to the class frequency.

For ordinal classification, Mean Absolute Error (MAE) is typically used,

$$\text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|. \quad (2.3)$$

This metric approximates the original problem as a cardinal problem, more strongly punishing errors that are farther apart – yet, the metric is sensible to the class frequencies, and is therefore not suitable for class imbalance. Two suggestions from the literature are the Average Mean Absolute Error (AMAE) and the Maximum Mean Absolute Error (MMAE) which are the mean and the maximum of MAE classification errors across classes, respectively [44]. They are defined as

$$\text{AMAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K} \sum_{k=1}^K \text{MAE}_k(\mathbf{y}, \hat{\mathbf{y}}), \quad (2.4)$$

$$\text{MMAE}(\mathbf{y}, \hat{\mathbf{y}}) = \max \{ \text{MAE}_k(\mathbf{y}, \hat{\mathbf{y}}) \mid k = 1, \dots, K \}, \quad (2.5)$$

where  $K$  is the number of classes involved and  $\text{MAE}_k$  means MAE is computed only for class  $k$ , i.e.,

$$\text{MAE}_k(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_k} \sum_{i=1}^N \mathbb{1}(y_i = k) |y_i - \hat{y}_i| \quad (2.6)$$

with  $N_k$  being the number of observations of class  $k$ . Naturally, MAE is bounded by  $[0, K - 1]$  and so are AMAE and MMAE. Sometimes rank correlations are also used as ordinal metrics, such as Spearman  $\rho$  and Kendall  $\tau$ .

## 2.1 Related Work

Traditionally, class imbalance has been addressed by a wide range of approaches:

- A. Pre-processing** step changing the class priors by undersampling the majority class and/or creating new synthetic examples of the minority class using SMOTE [45].
- B. Training** can be adjusted, such as using a cost matrix (weights) so that the training algorithm maximizes a weighted accuracy, where the cost of misclassifying a class is inversely proportional to its frequency.
- C. Post-processing** by tweaking the decision boundary by such measures as changing a threshold after which one class is selected, sometimes with the aid of a ROC curve [46].

This list is by no means exhaustive. One-class models, which are models that model only one of the classes and ignore the others, are also sometimes deployed to focus on the minority class, though they do not usually produce very interesting results [47, see Table 4]. On the other hand, some rule induction models can be made to prioritize one of the classes, and have been found to produce interesting results [48]. Ensembles can also be used so that each model within the ensemble is trained with balanced subsets of the data – this balancing needs to use one of the previous pre-processing techniques with the advantage of not completely discarding undersampled data [49].

We propose a fourth major alternative family of solutions:

- D. Pairwise scoring ranking** which is a family of models borrowed from the learning to rank literature where the problem to be optimized uses pairs of observations.

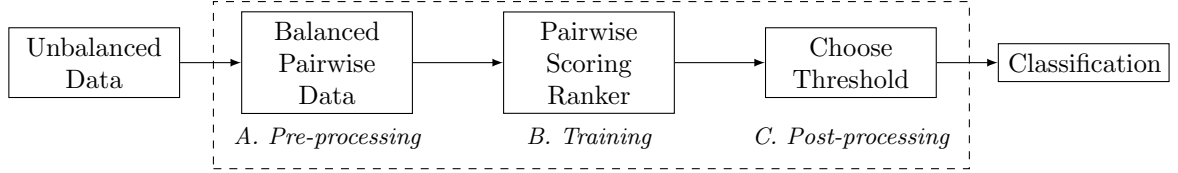


Figure 2.2: Adaptation of pairwise ranking to class imbalance data.

## 2.2 Proposal

In the learning to rank literature, a document  $\mathbf{x}_i$  is compared with another document  $\mathbf{x}_j$ , and we are interested in predicting whether  $\mathbf{x}_i \succ \mathbf{x}_j$ , which means document  $\mathbf{x}_i$  is “preferred” to document  $\mathbf{x}_j$  by some criterion. The three big umbrellas of rankers are [50]:

- **Pointwise**, in which each document  $\mathbf{x}_i$  is trained individually and a score function,  $s(\mathbf{x}_i)$ , is given based on its relevance;
- **Pairwise**: each document  $\mathbf{x}_i$  is compared against all others  $\mathbf{x}_j$ , and if  $\mathbf{x}_i \succ \mathbf{x}_j$ , then we train a function  $s$  so that:
  - **pairwise scoring ranker**: if  $\mathbf{x}_i \succ \mathbf{x}_j$  then  $s(\mathbf{x}_i) > s(\mathbf{x}_j)$ , with  $s: X \rightarrow \mathbb{R}$ ;
  - **pairwise non-scoring ranker**: the decision function is such that it decides which of two documents is preferred,  $s: X \times X \rightarrow X$ ;
- **Listwise**, where the training loss function is based on all documents and their scores.

We will be using the *pairwise scoring ranking* family for two reasons: *Pairwise scoring* models have the advantage that they are trained in pairs, therefore voiding any class imbalance if used within a binary context (since they are *pairwise*) and predictions are still produced individually in the form of a score (since they are *scoring*), making them suitable for classification.

To adapt the pairwise scoring rankers for classification, the general framework will involve first converting the classes to the ranking space (pre-processing) and then converting back from the ranking space to the space of classes by applying a threshold (post-processing). This pipeline is illustrated in Figure 2.2.

## 2.3 Binary Case

The adaptation of the previous framework for the binary classification case is relatively straight-forward.

- A. Pre-processing:** For two classes  $\mathcal{C}_-$  with  $N_-$  observations and  $\mathcal{C}_+$  with  $N_+$  observations, pairwise rankers are trained with observation pairs,  $\mathbf{x}'_{ij} = (\mathbf{x}_i, \mathbf{x}_j)$  with a label  $y'_{ij}$ . Without loss of generality, we can take  $\mathcal{C}_+$  as being “preferred” to  $\mathcal{C}_-$  and assign  $y'_{ij} = 1$  whenever  $y_i = \mathcal{C}_+ \wedge y_j = \mathcal{C}_-$ , and  $y_{ji} = -1$  otherwise.
- B. Training:** Training will involve  $2N_-N_+$  comparisons between all pairs of each class. Examples of pairwise scoring ranking models include RankNet (neural networks) [51], RankSVM (SVM) [52] and RankBoost (boosting) [53], which are detailed below.
- C. Post-processing:** The threshold  $T: \mathbb{R} \rightarrow \mathcal{C}$  can be chosen in order to maximize a balanced metric  $m(y, \hat{y})$ . The metric used could be  $F_1$  or G-mean, as described

in the opening of the chapter. Using the training data, we have  $s_i = s(\mathbf{x}_i)$  which is sorted, and then each midpoint  $s'_i = \frac{s_i + s_{i+1}}{2}$  is tested as a possible candidate for threshold  $T$ , so that

$$T = \arg \max_{s'_i} m(y, \hat{y}), \quad (2.7)$$

where  $\hat{y} = \mathcal{C}_-$  if  $s_i < s'_i$  and  $\hat{y} = \mathcal{C}_+$  otherwise.

Some specific examples and adaptations of ranking models are now described. Notice how all these methods follow the two required properties: *pairwise* and *scoring*.

**RankSVM [52]** is pairwise because it operates in the space of differences of the original dataset, so that  $\mathbf{X}$  becomes  $\mathbf{X}'$ , where  $\mathbf{x}'_{ij} = \mathbf{x}_i - \mathbf{x}_j$  and  $\mathbf{x}'_{ji} = \mathbf{x}_j - \mathbf{x}_i$ , for all pairs  $(i, j)$ . RankSVM is also scoring because the base estimator is linear SVM whose decision rule is  $\mathbf{w} \cdot (\mathbf{x}_i - \mathbf{x}_j) > 0$  and can be transformed into a scoring function since  $\mathbf{w} \cdot (\mathbf{x}_i - \mathbf{x}_j) > 0 \iff \mathbf{w} \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_j \iff s(\mathbf{x}_i) > s(\mathbf{x}_j)$ .

**RankNet [51]** consists in a neural network  $s$  that outputs a score given an observation,  $s: X \rightarrow [-1, 1]$ . The way this neural network is trained is by doing two forward-passes for each observation of all pairs  $(i, j)$  to output scores  $s_i$  and  $s_j$ ; cross-entropy ( $\mathcal{L}(y, \hat{y}) = y\hat{y}$ ) is then used on the score difference using the aforementioned labels:  $\mathcal{L}(y_{ij}, s_j - s_i) = y_{ij}(s_j - s_i)$ .

**RankBoost [53]** is based on AdaBoost, which consists in training a weak model sequentially,  $h_t$ , to improve on the predictions of the previous weak model,  $h_{t-1}$ . Each model, therefore, is trained on the *residuals* of all previous models. Furthermore, two sets of weights are used: each observation  $i$  and model  $t$  is given a weight  $w_{ti}$  and the model itself is given a weight  $\alpha_t$ ; both of these are computed based on an error function  $E$ , typically  $E(\mathbf{x}, y) = \exp(yh(\mathbf{x}))$ . The final prediction is given by a weighted average  $\hat{y} = \sum_t \alpha_t h_t(\mathbf{x})$ . In RankBoost, the only thing that changes is that residuals are given by the difference between the two predictions, therefore the error function becomes  $E(\mathbf{x}_i, \mathbf{x}_j, s_{ij}) = \exp(-s_{ij}(h(\mathbf{x}_i) - h(\mathbf{x}_j)))$ .

All in all, the sample size of the pairs ( $N' = N_- N_+$ ) is the quadratic of that of the regular sample size ( $N = N_+ + N_-$ ). If the regular training complexity was  $\mathcal{O}(N)$ , this might mean the new complexity is  $\mathcal{O}(N^2)$ . However, due to class imbalance,  $N_- \ll N_+$ . Table III.1 shows  $N_-$  is a small fraction of  $N_+$  that can be as low as 3%. Therefore, the true complexity might be closer to  $\mathcal{O}(N)$ .

## 2.4 Ordinal Case

The binary case is here extended for the ordinal case. Let  $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \dots \prec \mathcal{C}_K$  be the  $K$  classes involved. Let  $\mathcal{S}_k = \{\mathbf{x}_n^{(k)}\}$  be the set of  $N_k$  samples from  $\mathcal{C}_k$ , with  $N = \sum_{k=1}^K N_k$ . Construct the pairs  $\mathbf{x}_{mn}^{(k\ell)}$  with  $\mathcal{C}_k \prec \mathcal{C}_\ell$  and ranking labels -1 and +1 such that  $\{(\mathbf{x}_{mn}^{(k\ell)}, +1), (-\mathbf{x}_{mn}^{(k\ell)}, -1)\}$ .

An issue with this approach arises when one of the classes is strongly misrepresented when compared with the others. The data from each class  $\mathcal{C}_k$  is represented as  $2N_k(N - N_k)$  observations in the ranking space. If  $2N_k \ll 2N_\ell$  then  $N_k(N - N_k) \ll N_\ell(N - N_\ell)$ . For example, if  $N_1=10$  and  $N_2=N_3=100$ , then the data from  $\mathcal{C}_1$  is contributing with 4,000 elements in the new space, while the data from  $\mathcal{C}_2$  or  $\mathcal{C}_3$  is contributing with 22,000. So, the new learning problem will be dominated by the samples from  $\mathcal{C}_2$  and  $\mathcal{C}_3$  and it is likely

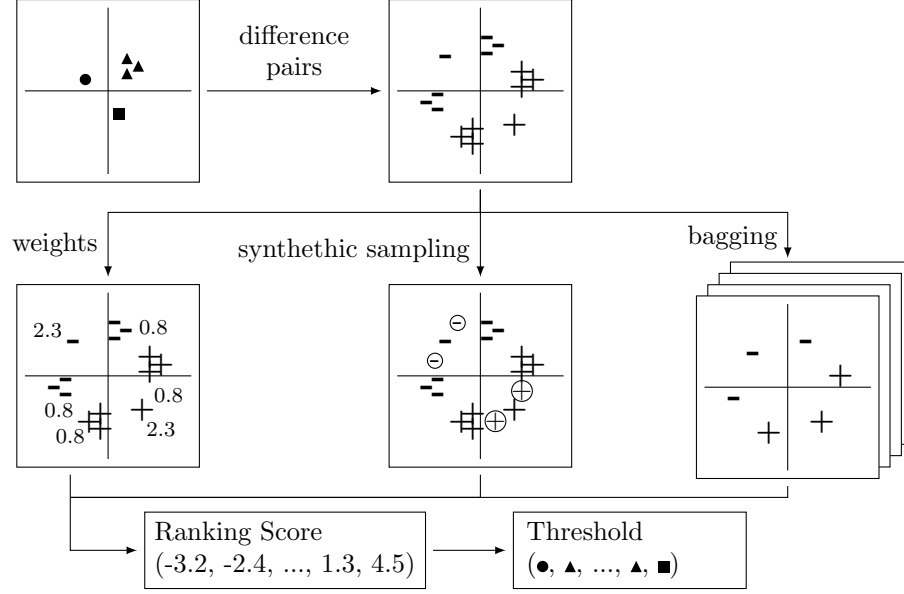


Figure 2.3: Balanced ranking training.

that  $\mathcal{C}_1$  will be poorly estimated. This makes the pairwise model imbalance in the  $K$ -wise case.

Two approaches at extending the previous framework to ordinal classes are here presented. The first approach combines ranking with the other balancing techniques (Section 2.4.1). The second approach involves adapting a multi-class ensemble (Section 2.4.2).

### 2.4.1 Combining Traditional Balancing Techniques

The aforementioned problem is here solved by combining pairwise ranking with conventional class imbalance approaches that were previously described in Section 2.1: (a) pre-processing, (b) training with costs, and (c) ensembles, as depicted in Figure 2.3.

The major modification is in how the ranking continuous score is transformed back to discrete ordinal classes. Based on the training data, we obtain a score  $\mathbf{s}$ , where each  $s_i$  is the score for each observation  $\mathbf{x}_i$  ordered by class  $y_i$ . To convert this score to classes,  $K - 1$  thresholds need to be optimized.

The best thresholds  $\mathbf{T}$  are chosen in order to minimize an error function  $f$ ,

$$\mathbf{T} = \arg \min_{\mathbf{T}} f(\mathbf{s}). \quad (2.8)$$

Testing all possible threshold combinations would be infeasible; it would run in factorial time since there are  $C_{K-1}^N$  combinations. With  $N=500$  and  $K=3$ , testing all 124,750 possible thresholds was empirically estimated to be more than 35 times slower than the following proposal, with the same results.

Fortunately, it is possible to take advantage of two things: (a) the score produced by the ranker grows with the class order, given that classes are ordinal, and (b) if the error metric is a linear function, and therefore each misclassification contributes to the metric additively, then the threshold selection can be divided into subproblems.

We propose a threshold strategy by defining recursively the threshold path of minimum error. Let  $s_i$  be the ordered score of the  $i$ -th observation and  $k_i$  be the true class, we search

the threshold increasingly by invoking the function  $f$  with initial parameters  $(s_0, k_0, 1)$ .

$$f(s_i, k_i, \hat{k}) = \begin{cases} 0, & \text{when } i = N \\ \varepsilon_{k_i \hat{k}} + f(s_{i+1}, k_{i+1}, \hat{k}), & \text{when } \hat{k} = K \\ \min \left\{ \varepsilon_{k_i \hat{k}} + f(s_{i+1}, k_{i+1}, \hat{k}), f(s_i, k_i, \hat{k} + 1) \right\}, & \text{otherwise,} \end{cases} \quad (2.9)$$

where  $\varepsilon = [\varepsilon_{k\hat{k}}]$  is a cost matrix. Informally,  $f$  tests whether, at any given time, it is less costly to continue assuming observation  $i$  to be of class  $\hat{k}$  or if it less costly to make a threshold and start assuming observations are now  $\hat{k}+1$ . Notice that observations have been ordered by the score and that classes are ordinal.

The threshold itself can then be inferred by transversing the  $\hat{k}+1$  breaking points (second term of min) which results in the minimum error.

By using dynamic programming's memoization, the function can be computed in  $\mathcal{O}(KN)$  in worst-case scenarios; for instance, when all classes belong to the last class,  $k_i = K, \forall i$ . This efficiency is based on the fact that classes are ordinal and comes at the expense of flexibility of the error function  $f$ . Error needs to be defined using a constant cost matrix because it is computed additively. Therefore, maximizing metrics from binary problems such as  $F_1$  or G-mean would not be possible.

The threshold optimization process itself is subjected to imbalance, since each observation contributes equally to the error function. This is why  $\varepsilon$  was defined as a cost matrix – several strategies can be chosen for this matrix:

- *Homogeneous*:  $\varepsilon_{k\hat{k}} = \begin{cases} 1 & \text{if } k \neq \hat{k} \text{ and } 0 \text{ otherwise} \end{cases} \mid \forall k, \hat{k}$
- *Absolute costs*:  $\varepsilon_{k\hat{k}} = \begin{cases} |k - \hat{k}| & \mid \forall k, \hat{k} \end{cases}$
- *Inverse class frequency*:  $\varepsilon_{k\hat{k}} = \begin{cases} \frac{N}{KN_k + 1} & \text{if } k \neq \hat{k} \text{ and } 0 \text{ otherwise} \end{cases} \mid \forall k, \hat{k}$ .

## 2.4.2 Frank & Hall Ensemble

A vast literature exists for ordinal classification, also referred to as ordinal regression. Indeed, one family of solutions is to simply treat it as a regression problem where the continuous prediction is discretized as a post-processing step. Considering only classification methods, two groups of ordinal models are identified:

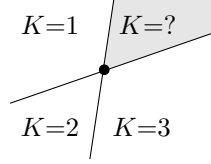
- (A) solving the ordinal problem by explicitly manipulating the loss function,
- (B) turning the ordinal problem into several binary classification problems (decomposition methods).

Neither of these groups specifically addresses ordinal class imbalance.

Within group (A), a popular SVM model is SVOR [54], with small versions, SVORIM and SVOREM, which differ only on how the constraints are defined. The idea is to find  $K-1$  parallel discriminant hyperplanes to properly separate the data into ordered classes by modeling ranks as intervals [54].

SVM, as originally formulated, solves binary classification by optimizing weights  $\mathbf{w}$  and intercept (or bias)  $b$  such that they minimize the hinge loss function,

$$\mathcal{L}(\mathbf{x}, \mathbf{y} \mid \mathbf{w}, b) = \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) + \lambda \|\mathbf{w}\|_2^2. \quad (2.10)$$



**Figure 2.4:** Diagram showing the lexicographical violation when decision hyperplanes intersect within the feature domain.

SVOR [55] adds  $K-1$  biases for the decision boundary, one between every two consecutive ordinal classes,

$$\mathcal{L}(\mathbf{x}, \mathbf{y} \mid \mathbf{w}, \mathbf{b}) = \sum_{k=1}^{K-1} \sum_{i=1}^N \max(0, 1 - y'_i(\mathbf{w} \cdot \mathbf{x}_i + b_k)) + \lambda \|\mathbf{w}\|_2^2, \quad (2.11)$$

where  $y'_i = 1$  if  $y_i = k$  and  $-1$  otherwise. This formulation is based on the adaptation which can be found in [56].

Notice that weights  $\mathbf{w}$  are shared between decision boundaries, making them parallel to each other. The intersection of decision boundaries is seen as problematic since it makes the ordinal prediction volatile within the intersection region [57]. Consider Figure 2.4, it is not obvious what class should be placed in the “?” decision space. Most classifiers solve this problem by ensuring parallelized decision boundaries. An adaptation that allows some flexibility of the decision boundaries without allowing intersections can be found in [58].

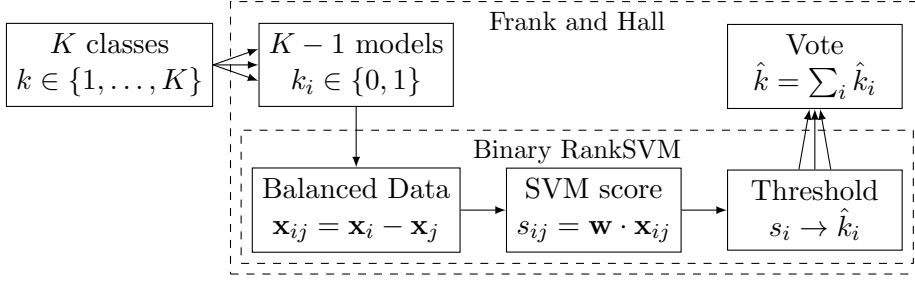
Group (B) are the decomposition methods, which solve the ordinal problem by reducing it into several binary problems. One such example is oSVM [59] which takes advantage of the fact that the decision boundaries are parallel in a well-formed ordinal problem. It starts by transforming the original ordinal problem into a binary problem by expanding the feature space as a pre-processing step and then translates the SVM coefficients of each new feature into separate biases as a post-processing step.

The most popular decomposition methods are ensembles: One-vs-rest or one-vs-all are two such popular ensemble strategies. These, however, exacerbated the imbalance problem and do not take advantage of the ordinality of the problem [60]. An ensemble designed for the ordinal case is the one proposed by Frank and Hall (F&H) [61]. This ensemble reduces the ordinal problem into  $K-1$  traditional binary classification problems. The  $k$ -th model is trained using classes  $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$  against  $\{\mathcal{C}_{k+1}, \dots, \mathcal{C}_K\}$ . If we have four classes, then three models are produced by using class (i)  $\mathcal{C}_1$  against  $\{\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ , (ii)  $\{\mathcal{C}_1, \mathcal{C}_2\}$  against  $\{\mathcal{C}_3, \mathcal{C}_4\}$  and (iii)  $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$  against  $\mathcal{C}_4$ . Each model  $f_k(\mathbf{x})$  is trained to produce 0 if  $y \leq k$  or 1 if  $y > k$ . This makes each classifier use highly imbalance data, even if the data was not originally so. The final prediction is then a simple cumulative voting,

$$\hat{y} = 1 + \sum_{k=1}^{K-1} f_k(\mathbf{x}). \quad (2.12)$$

There are two problems with this ensemble: (i) each binary classifier within the ensemble is highly imbalance, and (ii) the ensuing final model suffers from the aforementioned intersection problem. More specifically, assume  $y < i$ , an inconsistency may arise if model  $i$  votes for  $\hat{y} \leq i$  while model  $j > i$  votes for  $\hat{y} > j$ .

The proposal is made of two contributions: (1) use F&H and train each binary classifier in a balanced fashion. Furthermore, to avoid the aforementioned intersection problem, (2)



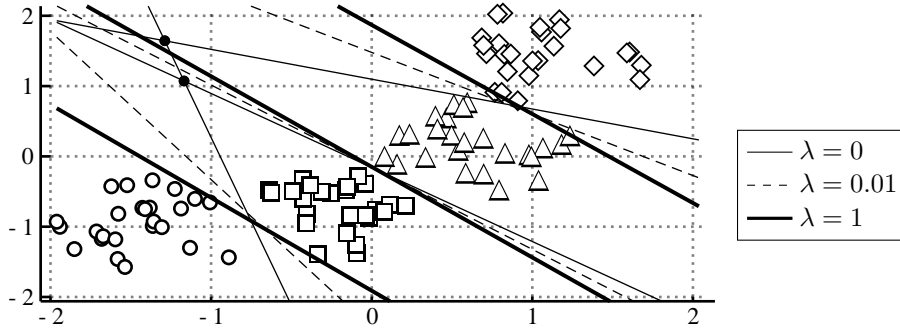
**Figure 2.5:** Diagram showing the F&H adaptation for balanced ordinal classification.

a regularization term is added so that each model within the ensemble does not diverge much. This combines the best of the group of methods (A) and (B).

The combination of F&H is illustrated in Figure 2.5. Contribution (2) alleviates the F&H violation of the lexicographical order. In order to tackle this problem, we propose: (a) training a base ordinal estimator  $g$ , and then (b) regularize each  $f_k$  to not diverge much from  $g$ . Using the previous SVM formulation, each individual model  $k$  must be tweaked so that regularization is related to the “base model”  $g$  with weights  $\mathbf{w}'$ ,

$$\mathcal{L}(\mathbf{x}, \mathbf{y} | \mathbf{w}_k, b_k) = \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w}_k \cdot \mathbf{x}_i + b_k)) + \lambda \|\mathbf{w}_k - \mathbf{w}'\|^2. \quad (2.13)$$

If regularization is set as  $\lambda = 0$ , then the pure F&H solution is used. If regularization  $\lambda \rightarrow \infty$ , then the decision boundary orientation will be parallel as in the base ordinal estimator while only the bias is provided by  $f_k$ . An illustration of the effect of changing  $\lambda$  is shown in Figure 2.6 – optimization was done using gradient descent with the learning rate being  $\eta = \frac{1}{\lambda it}$  where  $it$  is the number of iterations so far, as in Pegasos [62].

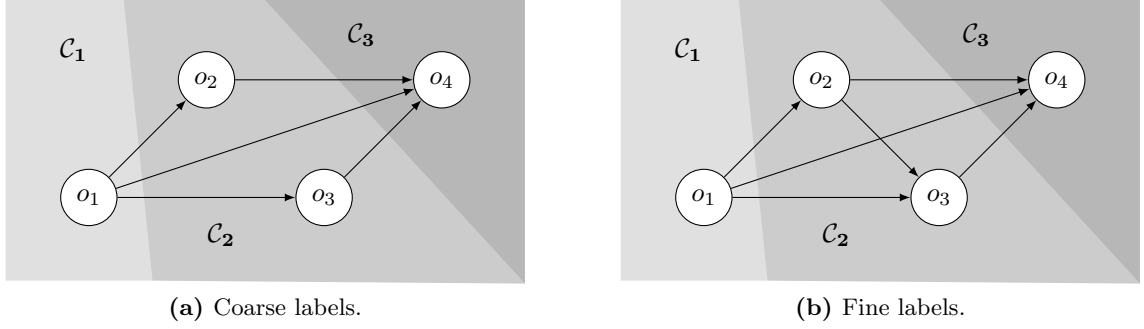


**Figure 2.6:** Synthetically generated non-parallel classes. Black points represent hyperplane intersections.

## 2.5 Survival Analysis

Some problems have a mix of ordinality and cardinality; the most prominent being survival analysis. Consider the problem of the number of days until a liver transplant failure, which is a problem considered in the following experimental section. Typically, failures tend to aggregate in clusters of unequal range and it is easier to think of them that way: “less





**Figure 2.7:** Ranking models can be trained to take advantage of the finer labels.

than 15 days”, “between 15 and 90 days”, “between 90 and 365 days”. Furthermore, there is one extra class, e.g. “more than one year”, when no failure happened during the study period. This makes a simple regression analysis infeasible because of the high number of cases that survive the time threshold of the study.

Our learning problem is framed as an ordinal regression problem with two levels of granularity on the target variable  $\mathcal{Y}_{\text{fine}} = \{\mathcal{C}_1^{\text{fine}}, \mathcal{C}_2^{\text{fine}}, \dots, \mathcal{C}_{Q_f}^{\text{fine}}\}$  and  $\mathcal{Y}_{\text{coarse}} = \{\mathcal{C}_1^{\text{coarse}}, \mathcal{C}_2^{\text{coarse}}, \dots, \mathcal{C}_{Q_c}^{\text{coarse}}\}$ , such that  $\mathcal{Y}_{\text{fine}}$  can be monotonously partitioned in  $Q_c$  disjoint groups.

This is a very common setting and can be seen in plenty of scenarios. For instance, students receive a continuous fine-grained mark (e.g. 0–100%) which is usually under-segmented into a smaller number of intervals (e.g. “fail”, “pass”, “pass with honors”). Another example can be observed in ranking companies being broadly categorized as small-medium-large enterprises based on a fine-grained scale given by the number of employees, annual turnover, etc.

In this sense, coarse labels are a semantic abstraction of the original phenomenon being quantified. While this abstraction may simplify the analysis for a human, it may impose a significant loss of information when building the model. For example, in the aforementioned liver example, the number of comparisons used for training in a coarse scheme is restricted to patients on different intervals, while on a fine scheme we can distinguish patients that survived  $n$  days from patients that survived  $n + 1$  days; see Figure 2.7.

The ranking models we have been using have the advantage they can be trained using the entire original information (fine labels) to obtain a sound and stable ranker, while, at the same time, the transformation from ranking to ordinal classification can be done using the ordinal classes (coarse labels).

## 2.6 Experiments

Some empirical experiments are now performed on the three ranking adaptations previously proposed. The experiments are here summarized – extended versions of the tables may be found at Appendix A.2.

**Binary case:** The datasets from Table III.1 are evaluated and averaged using  $F_1$ -score. Three families of models are here presented: SVM with linear kernel, multilayer perceptron (i.e. single-hidden layer neural network), and AdaBoost. For each family, the models tested are the unmodified baseline, training with weights inversely proportional to class frequency, application of SMOTE to synthetically oversample (with  $k=5$ ) from the minority class [45], and finally, the proposal rank adaptation, as detailed in Section 2.3. For each dataset,

**Table 2.1:** Results for the binary case, measured by  $F_1$  (higher is better).

<b>Linear SVM</b>				
	Baseline	Weights	SMOTE	Proposal
<b>Average</b>	50.3	62.7	62.8	<b>68.1</b>
<b>% Top-1</b>	20	20	0	<b>60</b>
<b>% Top-2</b>	27	33	53	<b>87</b>
<b>Multilayer Perceptron</b>				
	Baseline	Weights	SMOTE	Proposal
<b>Average</b>	65.1	<b>65.3</b>	58.3	64.0
<b>% Top-1</b>	27	20	20	<b>33</b>
<b>% Top-2</b>	47	<b>67</b>	27	60
<b>Adaboost</b>				
	Baseline	Weights	SMOTE	Proposal
<b>Average</b>	62.7	62.7	68.3	<b>68.9</b>
<b>% Top-1</b>	13	13	47	<b>53</b>
<b>% Top-2</b>	33	40	73	<b>100</b>

grid-search was used to optimize the number of neurons for the multilayer perceptron, and L2 regularization for the SVM.

The ranking models have performed statistically significantly better than their counterparts from the literature concerning the  $F_1$ -score; see Table 2.1. ROC curves were also analyzed, which is a common measure to evaluate how correctly classified observations would be if the decision threshold  $T$  was changed, and there is a slightly lower performance for the ranking model, which suggests the threshold optimization itself is partly responsible for the gain.

Furthermore, data overlap (OR) is contrasted with the imbalance ratio (IR), as described in Section III.1.1, to better understand whether the ranking model is helping more with the OR or IR. Performance across datasets was correlated using Spearman’s  $\rho$ . For linear SVM, IR  $\rho$ -correlation was of 48% for the baseline and 31% for ranking which suggests that ranking is more resilient to IR. Also, OR  $\rho$ -correlation was of -30% for the baseline and -19% for ranking, again, showing resiliency.

Correlations were also performed between models of the same family (e.g. SVM vs RankSVM) and of the same method (e.g. RankSVM vs RankNet) which show rankers more closely follow the decision function of their family of models than that of the rest of the rankers. Ranking techniques can therefore be concluded to be an extra technique of tackling class imbalance to try to improve a currently employed solution.

While we have used a single-hidden layer neural networks in this work and these were not the best performing family of models, later work by a colleague showed RankNet also having good results when using a convolutional neural network with images [63].

**Ordinal case:** Here, datasets from Table III.2 are evaluated and averaged using the MMAE metric, detailed in the opening of the chapter (2.5). A linear SVM was tested

**Table 2.2:** Results for the ordinal case, measured by MMAE (lower is better).

<b>Baselines</b>					<b>Ranking</b>		
	OvR/w	SVORIM	oSVM	F&H	Weights	SMOTE	F&H ( $\lambda=0.1$ )
<b>Average</b>	1.88	2.13	1.95	2.17	1.77	1.63	<b>1.60</b>
<b>% Top-1</b>	13	7	0	0	13	<b>33</b>	<b>33</b>
<b>% Top-2</b>	33	7	20	0	40	<b>53</b>	47

with the baselines being One-vs-Rest with weights inversely proportional to class frequency (OvR/w), SVORIM, oSVM, and the Frank & Hall ensemble using SVMs, as detailed in Section 2.4. Furthermore, the two previous ranking strategies to extend the binary case to the ordinal case are evaluated: combining ranking with traditional balance techniques is shown in column Weights and SMOTE (Section 2.4.1), while F&H implements our Frank & Hall adaptation (Section 2.4.2). Absolute costs seem like the  $\varepsilon$  matrix cost threshold that is most stable and has been used here.

Ranking performs competitively for ordinal classification, as shown in Table 2.2. Interestingly, One-vs-Rest, which is not an ordinal ensemble, is also competitive, possibly due to the extra flexibility offered by decision boundaries not being constrained to be parallel.

Kendall’s  $\tau$  correlation between  $N$  and the error was of -0.07 for  $\lambda = 0.01$  and of -0.01 for  $\lambda = 1$ . In either case, it makes sense that error reduces as dataset size increases since the models have more data from which the underlying distribution can be inferred. However, it is interesting to note that this reduction seems to be inversely proportional to the  $\lambda$  regularization being used, i.e. forcing parallel decision hyperplanes seems to work best for data with more observations. This puzzling fact may be answered by another correlation between IR and the error: -0.22 for  $\lambda = 0.01$  and -0.16 for  $\lambda = 1$ , confirming that parallel hyperplanes do work better for higher imbalance data (IR), possibly due to the difficulty of estimating parameters associated with the minority classes when data is scarce. The correlation between OR and error are similar, but a little more pronounced (0.26 and 0.20, respectively).

**Survival Analysis:** A case study was also performed to validate the adaptation suggested for survival analysis, as detailed in Section 2.5. The study is on liver transplantation in co-operation with seven Spanish transplantation units and the London King’s College hospital [64], and multiple metrics are used: MAE, MMAE, and AMAE (2.3)–(2.5). The previous baselines (SVM and SVORIM) were reused and tested against the normal RankSVM trained with the coarse labels (Coarse Rank) with the same model trained with the finer labels (Fine Rank). In either case, the threshold strategy is then trained using the coarse labels, naturally.

The rank-based learners never beat SVM or SVORIM, but seem to offer more balanced results across the metrics, see Table 2.3. In fact, the average of all metrics is lower for the ranking methods than the others.

**Table 2.3:** Results for the survival lung cancer case study (lower is better).

	SVM	SVORIM	Coarse Rank	Fine Rank
<b>MAE</b>	<b>0.30</b>	1.43	0.80	0.67
<b>MMAE</b>	3.00	<b>2.19</b>	2.42	2.45
<b>AMAE</b>	1.50	<b>1.10</b>	1.42	1.41

## 2.7 Summary

About two hundred papers have been published about class imbalance since 2012 if searching titles by “class imbalance” in Google Scholar. It is not clear that ranking is a superior solution, but it is a very competitive and promising alternative that we felt was sorely lacking in the literature.

Pairwise learning has been a neglected form of producing balanced models, and our suggestion is to adapt models from the learning-to-rank literature which uses such models. While pairwise learning is like fish in the water for binary classes, extending them to the

ordinal case is a possibility as described, and some cardinal problems, such as survival analysis, might also benefit from ranking because it can potentially take advantage of the more fine-grained information provided by the problem.

In most cases, the proposed ranking method is found to be highly competitive against traditional approaches, especially when considering the imbalance-sensitive metrics. While training times were not the focus of the experiments, a big inconvenience is that training times are usually higher, possibly insuperably higher for very big datasets.

There is a somehow related line of research on “AUC optimization” whose goal is to maximize the area under the curve of such metrics as ROC instead of a numeric value such as accuracy. This research has found rankers to be optimal ways to maximize AUC in the binary context [65].

Rankers have another latent benefit when considering rankers as possible classifiers; this latent benefit has not yet been discussed and is outside the scope of this chapter. But it should be noted that rankers can use extra information about the order of classes. That is, the data collection process itself could avoid constraining labels to broad categories such as “healthy” and “sick”, or “credit-worthy” and “not credit-worthy”. In many real-world applications, it might make more sense to label the data in terms of pairwise comparisons because it is often more intuitive for the human classifier to use relative labels rather than absolute labels. For example, it might be easier for a doctor to tell which of two images of disease is in a later stage, rather than specifying that a patient is in stage 3 or stage 5 of the disease.

**Publications:** This chapter is a summary of work that was published across several papers with special participation from my colleague Kelwin Fernandes which has expertise in ranking and contributed with several ideas. The original idea for binary classification was published as:

1. R. Cruz, K. Fernandes, J. S. Cardoso, and J. F. P. Costa, “Tackling Class Imbalance with Ranking,” in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016. [doi: 10.1109/IJCNN.2016.7727469]

Subsequent attempts at extending to the ordinal case were then published in special collaboration with María Pérez-Ortiz who is an expert on ordinal imbalance that we met at the IJCNN conference:

2. R. Cruz, K. Fernandes, J. F. P. Costa, M. P. Ortiz, and J. S. Cardoso, “Ordinal Class Imbalance with Ranking,” in *Iberian Conference on Pattern Recognition and Image Analysis (Ibpria)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-58838-4\_1]
3. R. Cruz, K. Fernandes, J. F. P. Costa, M. P. Ortiz, and J. S. Cardoso, “Combining Ranking with Traditional Methods for Ordinal Class Imbalance,” in *14th International Work-Conference on Artificial Neural Networks (IWANN)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-59147-6\_46]
4. R. Cruz, K. Fernandes, J. F. P. Costa, M. P. Ortiz, and J. S. Cardoso, “Binary ranking for ordinal class imbalance,” in *Pattern Analysis and Applications*, Springer, 2018. [doi: 10.1007/s10044-018-0705-4]

An experiment using Alzheimer data was performed in collaboration with Margarida Silveira of IST:

5. R. Cruz, M. Silveira, and J. S. Cardoso, “A Class Imbalance Ordinal Method for Alzheimer’s Disease Classification,” in *2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, IEEE, 2018. [doi: 10.1109/PRNI.2018.8423960]

The survival analysis experiment was performed by María Pérez-Ortiz using confidential data from London King’s College:

6. M. P. Ortiz, K. Fernandes, R. Cruz, J. S. Cardoso, J. Briceño, and C. Hervás-Martínez, “Fine-to-Coarse Ranking in Ordinal and Imbalanced Domains: An Application to Liver Transplantation,” in *14th International Work-Conference on Artificial Neural Networks (IWANN)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-59147-6\_45]

The first and original paper on ranking for binary classification [66] was cited 9 times by other authors than ourselves. Of these, 7 included the method in a review of methods for class imbalance, 1 cited the paper when discussing ranking, and 1 other extended on top of our work. In “AP-Loss for Accurate One-Stage Object Detection” from 2020 [67], the authors use ranking methods in the context of object detection. In that context, class imbalance rears its ugly head since there are many pixels of background for each pixel of relevant foreground (object), and using a loss based on pairwise ranking was found to be fruitful.



## Risk Aversion

In many applications, false positives (Type I error) and false negatives (Type II error) have a different impact. In medicine, it is considered worse to diagnosticate someone sick as healthy (false negative) than to falsely diagnosticate someone healthy as sick (false positive). Yet, we are also willing to accept some rate of false negatives errors to make the classification task possible at all. Where the line is drawn is subjective and prone to controversy – usually, this compromise is given by a cost matrix where an exchange rate between errors is defined.

However, for many reasons, it might not be natural to think of this trade-off in terms of relative costs. We explore novel ways of specifying this trade-off as an absolute cost on the amount of false negatives (FN) we are willing to tolerate. The classifier then tries to minimize false positives (FP) while keeping false negatives within that tolerance bound. In broad strokes, the problem is formulated as the following optimization problem:

$$\begin{aligned} &\text{Minimize FP} \\ &\text{subject to FN} \leq \rho. \end{aligned} \tag{3.1}$$

The user-defined parameter  $\rho$  will be used to represent the tolerance bound FN, and  $\hat{\rho}$  for the empirical FN produced by the model, as estimated from the data sample.

In this chapter, the focus is going to be on neural networks as trained by gradient descent, but we start by mentioning preliminary models using density estimation. The proposed techniques can be used for classification, but also for segmentation. Semantic segmentation consists of classifying each pixel as belonging to the region of interest or not, as was explained in Prologue II.2 – erring on the side of having an overdeflated or overinflated segmentation is also a problem.

### 3.1 Related Work

Imputing costs via a cost matrix is the *de facto* approach for tackling false classification trade-offs, where  $c_p$  and  $c_n$  are weights assigned by the user to the positive and negative cases. In the most common case, when the correct decision has a null cost, then the cost matrix has only one degree of freedom,

$$\begin{pmatrix} 0 & c_p/c_n \\ 1 & 0 \end{pmatrix} \tag{3.2}$$

These costs are then taken into account by the model through (A) pre-processing, (B) weights on the loss function, or (C) through post-processing, usually in the context of a ROC curve. These solutions are based on a relative trade-off between FP and FN. None of the approaches offers a means to define an absolute trade-off.

When it comes to classification using neural networks, a common loss function is binary cross-entropy with costs introduced as

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{p}}) = -\frac{1}{N} \sum_{i=1}^N [c_p y_i \log \hat{p}_i + c_n (1 - y_i) \log(1 - \hat{p}_i)], \quad (3.3)$$

where  $N$  is the number of observations,  $y_i \in \{0, 1\}$  are the labels, and  $\hat{p}_i \in [0, 1]$  are the probabilities estimated by the model, which then predicts  $\hat{y}_i = \mathbb{1}(\hat{p}_i \geq 0.5)$ .

One way to consider the proposed methodology is to consider that the current approach, when using loss (3.3), expands the decision boundary until errors are balanced by the given costs,

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}} = 0 \quad \Leftrightarrow \quad c_p \sum_{i=1}^N y_i (1 - \hat{p}_i) = c_n \sum_{i=1}^N (1 - y_i) \hat{p}_i \quad \Rightarrow \quad c_p \text{FN} = c_n \text{FP}. \quad (3.4)$$

Or, expressed in statistical learning theory using random variables,  $c_p P(\hat{Y} = 1 | Y = 0) = c_n P(\hat{Y} = 0 | Y = 1)$ , as in Figure 3.1 (a). On the other hand, our approach considers expanding the decision boundary of one class until the total error rate in the other class is controlled,  $P(\hat{Y} = 0 | Y = 1) \leq \rho$  and  $P(\hat{Y} = 1 | Y = 0) \rightarrow 0$ , see Figure 3.1 (b).

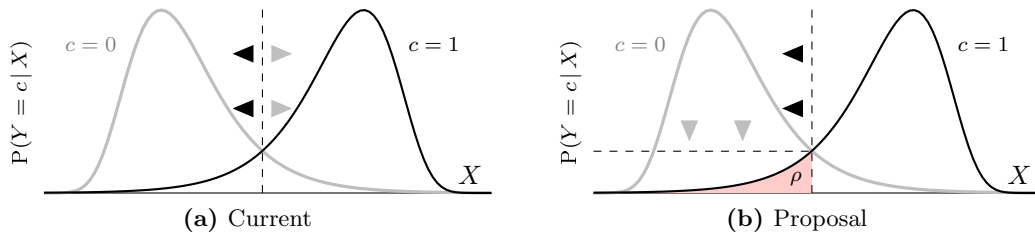


Figure 3.1: Comparing the current methodology to the proposed one.

## 3.2 Proposal

Machine learning models may be divided in *generative* and *discriminative* models. Generative models represent data  $X$  and labels  $Y$  using a **joint** probability distribution,  $P(X, Y)$  while discriminative models only learn **conditional** probability distributions,  $P(Y | X)$ .

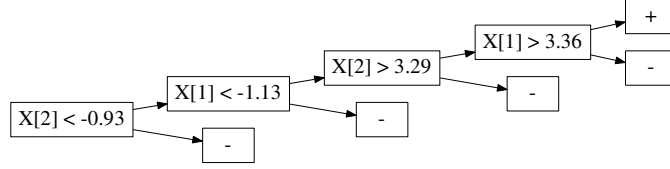
By applying Bayes' rule, any generative model may be converted into a discriminative model. However, in practice, discriminative models tend to be more accurate for classification tasks. This section will start by adapting the generative model KDE for risk aversion (Section 3.2.1) and then moves on to neural networks which are discriminative models (Section 3.2.2).

### 3.2.1 Kernel Density Estimation

By modeling data as  $P \sim P(X)$ , we can then use a threshold  $T$  as the decision. The threshold  $T$  can be estimated as the  $\rho$ -quantile using the training data of the positive cases,

$$P(P \leq T | Y = 1) = \rho. \quad (3.5)$$





**Figure 3.2:** Example of the one-class cascade approach.

However, modeling data,  $P(X)$ , requires making strong assumptions about the data, therefore, we will be modeling it using Kernel Density Estimation (KDE), also known as the Parzen-Rosenblatt window method.

KDE is a non-parametric way to estimate the probability density function of a random variable. Each observation is modeled by a kernel, all sharing the same parameters. Usually a Gaussian kernel is used (as we did), so that, in essence, the ensuing model is a Gaussian mixture. In the univariate case, for a given point  $x$ , the unknown function is thus being modeled as

$$\hat{p}(x) = \frac{1}{Nh} \sum_{i=1}^N K\left(\frac{x - X_i}{h}\right), \quad (3.6)$$

where  $K$  is the Gaussian kernel (the Gaussian density function for  $\mathcal{N}(0, 1)$ ), and  $h > 0$  is a smoothing hyperparameter. We have used the so-called Scott’s rule [68],  $h = N^{-\frac{1}{d+4}}$ .  $N$  and  $d$  are the number of observations and variables, respectively.

Two approaches that make use of KDE are now proposed:

**Multivariate approach:** Take the multivariate KDE and model the observations of only one of the classes – either positives,  $P(X | Y = 1)$  (Multivariate<sup>+</sup>), or negatives,  $P(X | Y = 0)$  (Multivariate<sup>-</sup>). The threshold  $T$  can then be estimated using the positives observations as in (3.5).

**Cascade approach:** To include information from both classes, instead of a multivariate KDE, we propose each variable of the negative class to be modeled independently in multiple univariate KDE models, i.e.,  $P(X^k | Y = 0)$  for each feature  $k$ .

As illustrated by Figure 3.2, the decision from these multiple models is composed in the form of a cascade. At each step  $j$ , there is a decision in the form of a filter: either observation  $i$  is predicted to be negative,  $\hat{y}_i = 0$ , or the decision is remitted to the next step  $j + 1$ . When steps are exhausted, the observation is considered positive,  $\hat{y}_i = 1$ .

The step at iteration  $j$  is chosen as

$$s_j = (\text{variable } k, \text{binop, threshold } T), \quad (3.7)$$

with  $\text{binop} = \{<, >\}$ . That is, we must choose what model  $k$  to use and what/where the threshold is. The threshold  $T$  at each iteration is fixed and chosen using half-interval search,  $T_j = \rho/2^j$ , so that  $\sum_{j=1}^{\infty} T_j = \rho$ . There are yet two parameters to estimate in  $s_j$ : the negative class guides the selection of  $k_j$  by a greedy criterion based on whichever maximizes the highest TN rate at the iteration,  $\arg \min_k \mathbb{E}[P^k | Y = 0]$ , where  $P^k$  is the model’s probability distribution for variable  $X^k$ . The positive class guides the selection of the threshold “binop” left/right tail by modeling the positives, choosing one of the tails:  $P(X^k | Y = 1) < T$  or  $P(X^k | Y = 1) > 1 - T$ , whatever minimizes the FN rate ( $\rho$ ).

These two approaches are exhibited in Figures 3.3 and 3.4, respectively. There is a trade-off here. On the one hand, the *multivariate approach* models only one class using information from both in the multivariate<sup>-</sup> case, while the *cascade approach* uses models of

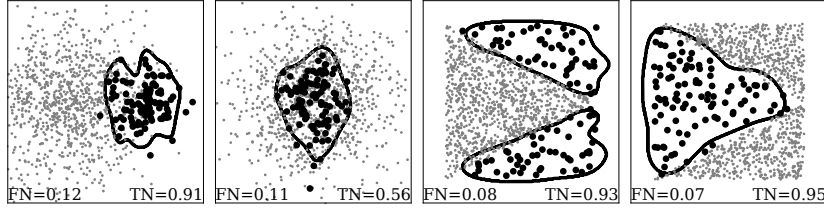


Figure 3.3: One-class approach using a synthetic example.

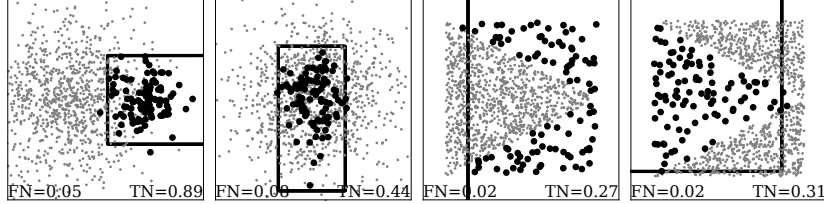


Figure 3.4: Two-class approach using synthetic example.

both classes. On the other hand, the cascade approach models each variable independently as a univariate model.

### 3.2.2 Neural Networks

The bulk of the chapter will focus on neural networks, in particular on the loss function: adding an averse term to the loss or alternating the loss to focus on one class of errors as necessary.

**Threshold:** Considering a model  $P \sim P(Y | X)$ , then a threshold  $T$  could be found such that  $P(P \leq T | Y = 1) = \rho$ . Taking the model's probability output as  $\hat{p}_i$  for observation  $i$ , then threshold  $T$  corresponds to the  $\rho$ -quantile of  $\hat{\mathbf{p}}$  when evaluated using an external validation set. This is the simplest of the methods since traditional training can be performed and the threshold can be adjusted at any time after training.

**Averse term:** Another line of approaches involves modifying the training itself. Gradient-based algorithms use a loss function  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{p}})$ , which must be  $\mathcal{C}^1$ . An averse term  $A$  may introduce the concept of absolute costs to the loss,

$$\mathcal{L}'(\mathbf{y}, \hat{\mathbf{p}}) = \mathcal{L}(\mathbf{y}, \hat{\mathbf{p}}) + \alpha A(\rho, \hat{\rho}). \quad (3.8)$$

where  $\hat{\rho}$  is the estimated FN rate based on the validation set. Possible  $A(\rho, \hat{\rho})$  terms could be: (i)  $\exp(\rho - \hat{\rho}) - 1$ , (ii)  $\max(0, \hat{\rho} - \rho)^2$  which is a truncated squared error, (iii)  $\max(0, -\log(1 - \hat{\rho} + \rho))$  which is less intuitive but has a behavior more similar to cross-entropy. The latter two are plotted in Figure 3.5. Terms (ii) and (iii) use max so

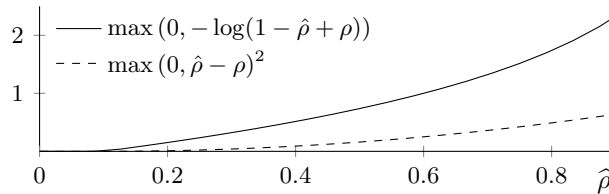


Figure 3.5: Averse loss terms, with  $\rho = 0.1$ .

that values for which  $\hat{\rho} \leq \rho$  are not penalized, since we had previously formulated the optimization constraint as such.

Since the loss must be differentiable of class  $C^1$ , so must be  $\hat{\rho}$ . We propose the following empirical FN rate,

$$\hat{\rho}(\mathbf{y}, \hat{\mathbf{p}}) = \frac{1}{\sum_{i=1}^N y_i} \sum_{i=1}^N y_i (1 - \hat{p}_i), \quad (3.9)$$

as an approximation to the true FN rate,  $\rho = \mathbb{E}[1 - \hat{Y} | Y = 1]$ .

**Alternating Loss:** Instead of the extra term  $A$ , binary cross-entropy  $\mathcal{L}$  could be split up into its positive and negative components and  $\beta = c_n$  could be adjusted dynamically,

$$\mathcal{L}' = \mathcal{L}^{(+)} + \beta \mathcal{L}^{(-)}. \quad (3.10)$$

At small update intervals (e.g. the end of a mini-batch), the empirical FN ( $\hat{\rho}$ ) is estimated from the validation set. If  $\hat{\rho}$  is above the reference value, i.e.  $\hat{\rho} > \rho$ , then  $\beta$  is set to zero to focus the learning process on the positive cases; that is,

$$\beta(\rho, \hat{\rho}) = \begin{cases} 1, & \text{if } \hat{\rho} \leq \rho, \\ 0, & \text{if } \hat{\rho} > \rho. \end{cases} \quad (3.11)$$

Also,  $\hat{\rho}$  no longer must be differentiable and so can now be estimated directly,

$$\hat{\rho}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{\sum_{i=1}^N y_i} \sum_{i=1}^N y_i (1 - \hat{y}_i), \quad (3.12)$$

without using the aforementioned approximation (3.9). We suggest allowing a “warmup” period of a few epochs where  $\beta = 1$  in order to help stabilize training.

**Fine-tune after training:** Here, normal cross-entropy training is used until convergence at epoch  $t'$ . The model can then be re-trained (fine-tuned) using  $\mathcal{L}' = \mathcal{L}^{(+)}$  until false negatives are restrained and training stops; that is,

$$\beta(t) = \begin{cases} 1, & \text{if } t \leq t', \\ 0, & \text{if } t > t', \text{ until } \hat{\rho} \leq \rho. \end{cases} \quad (3.13)$$

This can be seen as a type of transfer-learning whereby initially the model learns from positive and negative examples, and the final model is then tweaked to learn only from positive examples until  $\hat{\rho} \leq \rho$ . A smaller learning rate  $\lambda$  is recommended for this later fine-tuning period.

### 3.3 Experiments

The datasets used were the ones previously described in Prologue III for tabular data (Kernel Density Estimation) or image and segmentation cases (loss adaptation).

Results for Kernel Density Estimation methods are presented in Table 3.1 for the testing set. While multivariate<sup>+</sup> is able to properly contain  $\hat{\rho} < 5\%$  for the training set, it fails miserably when extrapolating to the testing set, perhaps because ignoring the negative class, which is the majority class, provides for a poor model. Multivariate<sup>-</sup> is maybe better at containing risk relative to the cascade approach, but at a considerably higher FN rate. An extended version of the tables is available in Appendix A.3.

**Table 3.1:** Aggregated test results using KDE for desired  $\rho = 5\%$ , broken down for the two types of errors.

	Multivariate <sup>+</sup>		Multivariate <sup>-</sup>		Cascade	
	FN	FP	FN	FP	FN	FP
Average	90.2	7.4	5.9	75.2	6.1	60.0
% FN $\leq \rho$		0		40		<b>50</b>
% Lowest FP		<b>90</b>		0		10

**Table 3.2:** Aggregated test results using the loss adaptations for desired  $\rho = 5\%$ , broken down for the two types of errors.

Classification (%)							
	Threshold		Averse term		Alternating		
	FN	FP	FN	FP	FN	FP	
Average	<b>5.9</b>	49.3	4.2	45.5	<b>17.5</b>	37.1	
% FN $\leq \rho$		20		<b>60</b>		30	
% Lowest FP		20		10		<b>80</b>	
Semantic Segmentation (%)							
	Threshold		Averse term		Alternating		Fine-tune
	FN	FP	FN	FP	FN	FP	FN FP
Average	<b>5.9</b>	25.6	2.3	52.9	3.9	44.4	2.6 33.6
% FN $\leq \rho$		56		<b>89</b>		78	<b>89</b>
% Lowest FP		<b>78</b>		0		11	22

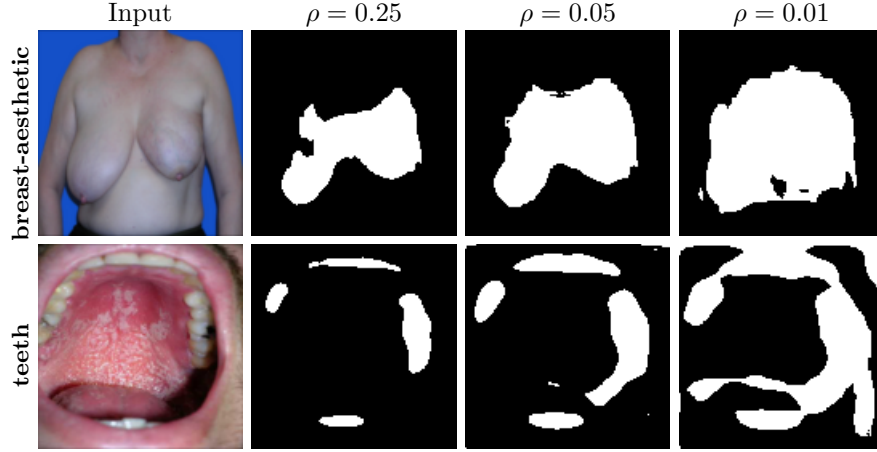
Results for convolutional neural networks using the loss adaptations are now presented in Table 3.2 using the four previous: threshold, averse term, alternative loss, and fine-tune after training.

For encoding and decoding, both in the CNN for classification and U-Net for segmentation, three blocks were used, each consisting of a convolution layer followed by max-pooling, plus the bottleneck block between the phases. Leaky ReLU activations were used, with a slope of 0.1, to avoid running into the dying ReLU problem. This was a prescient problem given the dynamic aspect of the training. The number of filters along the network was 3–32–64–128–256–128–64–32–1, with the last being the sigmoid output. The size of the pixel-map was 224–112–56–28–56–112–224, with downsampling and upsampling through max-pooling and bilinear interpolation, respectively. Adam was used as the optimizer for 75 epochs.

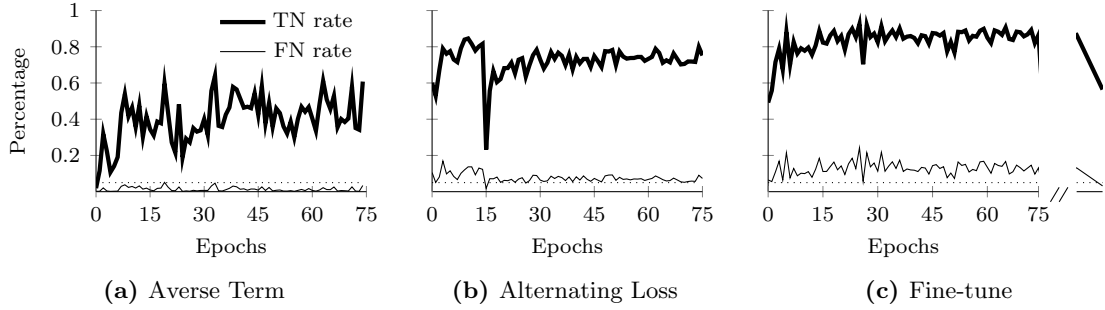
Firstly, not surprisingly the expressiveness of these models manage to greatly reduce FN error rate. More surprisingly, risk (FP rate) is also better contained.

The post-processing threshold method tends to be worst at ensuring FN stays below the user-defined  $\rho$  tolerance-bound. The averse term was the best in this criterion with “fine-tune” (only evaluated for semantic segmentation) beating all other methods.

Examples of segmentations are provided in Figure 3.6 for different values of  $\rho$  using the threshold method. Figure 3.7 illustrates the percentage of true and false negative rates (TN and FN, respectively) while the model is being trained. The alternating loss was found to be quite stable during training while the others were more stochastic. It may provide a good compromise and is the most flexible of methods for improvements because the alternating criterion needs not be differentiable.



**Figure 3.6:** Examples of varying the risk threshold.



**Figure 3.7:** Training history, as evaluated by the validation set, for the ISBI2017 dataset. The dotted horizontal line is the desired  $\rho$ .

### 3.4 Summary

A new way to define the class errors trade-off is proposed: instead of defining a relative trade-off using cost matrices, we suggest it might be useful in some cases for learning algorithms to allow defining an absolute trade-off in the form of a false negative threshold (risk). Some preliminary results using density estimation allow containing risk but at a generally low performance.

The methods involving neural networks, which are more expressive models, were more promising. These methods consist in: a threshold on the output probabilities, adding a term to the loss function, dynamically alternating losses whenever false negatives must be restrained, and change loss after training, akin to transfer-learning.

All these methods performed well except for the added term. The alternating loss and fine-tuning training methods provided more robust results at generalizing the false negative rate to out-of-train data than merely tweaking with the threshold probability. alternating and fine-tuning methods generalized better.

One major difficulty was to keep the training in tandem with the final test results regarding FPs. Solutions could encompass (a) aggressive regularization strategies, (b) using a smaller desired FP value  $\rho' = \eta\rho$ , with  $0 < \eta < 1$ , and obtained by cross-validation to ensure desired FP is controlled or (c) controlling for FN for a few more iterations even after the target  $\rho$  has been met.

**Publications:** This chapter was based on work first published on:

1. R. Cruz, K. Fernandes, J. F. P. Costa, and J. S. Cardoso, “Constraining Type II Error: Building Intentionally Biased Classifiers,” in *14th International Work-Conference on Artificial Neural Networks (IWANN)*, LNCS Springer, 2017. [doi: 10.1007/978-3-319-59147-6\_47]
2. R. Cruz, J. F. P. Costa and J. S. Cardoso, “Averse Deep Semantic Segmentation,” in *41st Engineering in Medicine and Biology Conference (EMBC)*, IEEE, 2019. [doi: 10.1109/EMBC.2019.8857385]

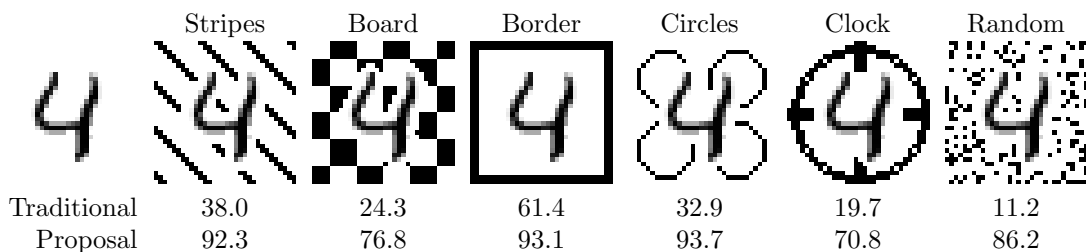
## Background Invariance

The third wave of machine learning (as mentioned by Prologue I) made it harder to introduce prior knowledge into the learning pipeline due to learning being largely hands-off. When this is done, it is typically done by introducing knowledge into the input data, the loss, or by constraining the output. For example, for ordinal classification, work exists that encodes classes in an ordinal fashion [69] and other work exists to constrain the output so that the output probabilities are unimodal [70].

We propose teaching the neural network what the foreground (object) is and to avoid being fooled by background changes. This will be done by introducing knowledge into the optimization process itself. Possibly due to the fact that neural networks learn from static images, and so do not have to deal with depth as us humans, they are vulnerable to changes in the background – for example, when there is a mismatch in the background between the training and test sets, performance degrades terribly, as exemplified by Figure 4.1. The classifier is trained with digits in a clean, white background (a trivial task) and then evaluated with digits inserted in diverse backgrounds. These after-training changes in the background have not been studied in detail. There is one work that uses an attention mechanism but only avoids some artifacts, such as irregular borders [71].

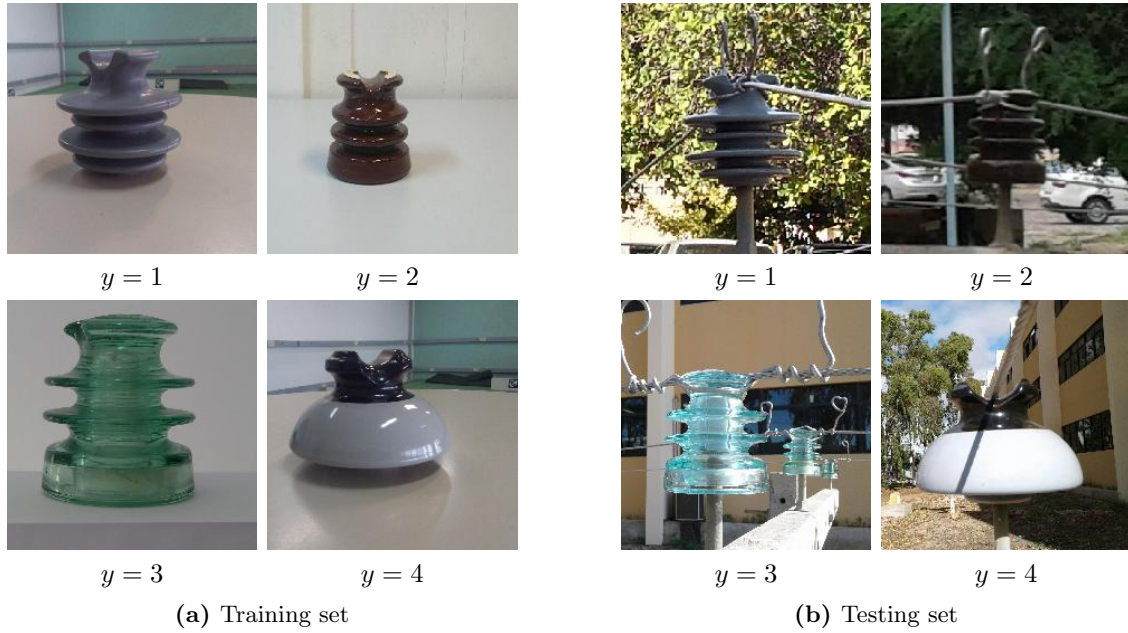
Data augmentation has been typically deployed to make neural networks more robust. This consists of expanding the training set by stochastically applying transformations [72]. For example, data augmentation through style transfer has been used to ensure the CNN is robust against changes in texture [73]. The method proposed here makes use of data augmentation on the background of the images in the training set.

In this chapter, a generator is proposed to augment the training set by producing backgrounds that purposefully harm the performance of the target neural network, making that target network more robust as a result. To avoid manual segmentation, backgrounds are introduced by a third neural network that unsupervisedly segments the object.



The model is a CNN with VGG blocks as detailed in Section 4.4.1, trained for MNIST. Accuracy values for the entire testing set when different backgrounds are used.

**Figure 4.1:** Background change can produce wild disparate accuracies (%).



**Figure 4.2:** The insulators dataset has four materials [75] and was constructed from images taken inside the laboratory (a) and outside the laboratory (b).

## 4.1 Motivation

In Brazil, overhead power distribution lines comprise more than 95% of the total medium voltage power circuits [74] and there is interest in using drones to help maintenance. In previous work [75], we collaborated with Ricardo Prates, which was a Ph.D. student from Brazil (Univasf) visiting INESC TEC. The goal was to produce a classifier that was trained with images of insulators acquired in-doors and that would successfully extrapolate to later out-doors images when used by a drone. The striking difference in background between the two datasets is illustrated in Figure 4.2.

The in-doors dataset (a) has been constructed by taking a series of photos of varying degrees from insulators made of four different materials – 480 in total (120 of each material) and an out-doors dataset (b) made of 520 images of each insulator taken from actual overhead power lines is used as a testing set. The dataset is available at [http://www.dee.eng.ufba.br/dslab/index.php/opdl\\_dataset/](http://www.dee.eng.ufba.br/dslab/index.php/opdl_dataset/). The task we propose solving is distinguishing between the four families of materials.

The original solution we published [75] was highly specific to this problem: (i) the training set was manually segmented, (ii) a data augmentation procedure was designed to randomly add patches taken from outdoor environments as the background, and (iii) some other foreground elements such as top tie and free electric conductors were also added to the data augmentation procedure. This generator is illustrated in Figure 4.3.

Not satisfied with this ad-hoc solution, we experimented with potential universal solutions and published a promising solution [76], which is detailed in this chapter. While we focus on classifiers, the method can be used on regression problems, reinforcement learning, or other problems using a CNN.



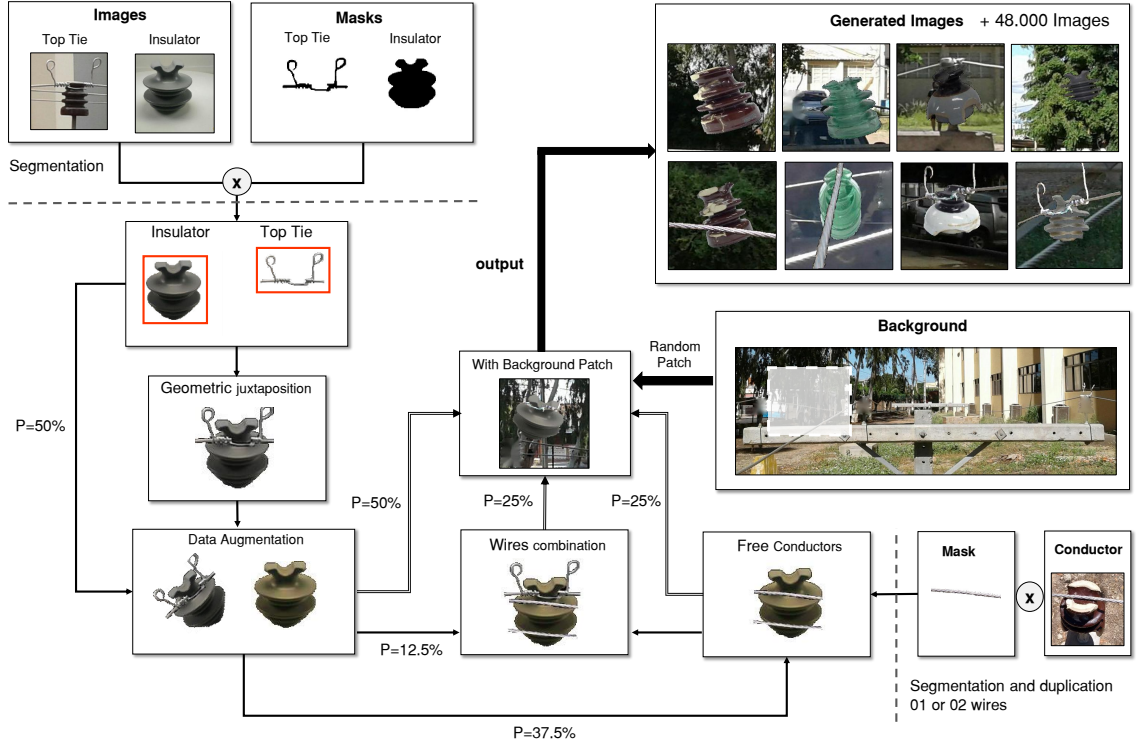


Figure 4.3: Image generator pipeline to make insulators background-insensitive [75].

## 4.2 Related Work

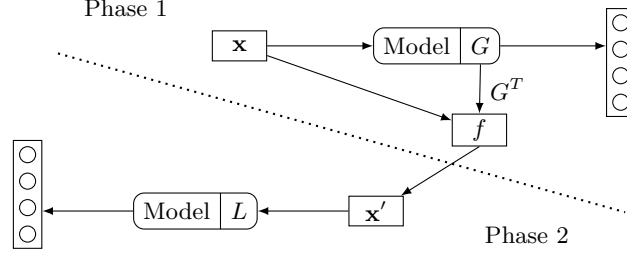
Literature exists in predicting classifier confidence for dataset shifts. It has been found that deep CNNs can produce wrong predictions with high confidence when the object is subject to translations, rotations, or changes in the background, sometimes even when using a completely unrelated dataset. The statistics of the image can be compared against the original training distribution through a probabilistic neural network and therefore detect a dataset shift, producing a low confidence score in such cases, so that those images can be rejected [77]. Using these techniques, we could detect if the background in the testing set is different from that used in the training set.

However, making the classifier itself robust to changes in the background seems to have been the subject of little study. One work proposes an attention mechanism to avoid artifacts, particularly irregular borders, from influencing the classifier [71]. Two classifiers are used: a *global* CNN,  $G$ , and a *local* CNN,  $L$ . The proposal works by having  $G$  find the bounding box of the relevant object to create a cropped version of the image and then use  $L$  to classify the cropped version.

As illustrated in Figure 4.4, this is done in two phases. Firstly,  $G$  is trained to classify the entire image  $\mathbf{x}$ . After  $G$  is trained, a truncated version of  $G$  is extracted, called  $G^T$ , which does not contain the last few layers, therefore producing an activation map,  $A$ . Then, a function  $f$  produces (i) a heatmap by merging the activation channels using the element-wise maximum absolute value,  $H_{ij} = \max_c |A_{ijc}|$ , and (ii) uses the connected components algorithm to produce a bounding box and extract a cropped version  $\mathbf{x}'$  of the image  $\mathbf{x}$ . Finally,  $L$  is then trained using  $\mathbf{x}'$  [71].

To then predict the class  $y$  of the image  $\mathbf{x}$ , this chained process outputs

$$\hat{y} = L(f(G^T(\mathbf{x}), \mathbf{x})). \quad (4.1)$$



**Figure 4.4:** Attention mechanism diagram proposed by [71].

Two disadvantages are immediate: (i)  $L$  operates on a rectangular cropped version of the image and therefore is still influenced by artifacts that remain within the rectangle, and (ii) the effect of the artifact on  $G$  may be strong enough to divert attention from the activation maps and capture attention away from the object.

Notice that model  $G$  is still influenced by artifacts because it did not have the benefit of being trained against the artifacts. While such artifacts are not presented in the training set, they could be generated in a controlled fashion.

Generative Adversarial Networks (GANs) generate images through a min-max optimization problem whereby two models try to optimize a given function in the opposite direction [78]. In an unsupervised fashion, the discriminator  $D$  tries to detect if an image  $\mathbf{x}$  comes from the training set  $X$  or has been produced by a generator  $G$ . Some random noise  $\mathbf{z}$  is given as input to the generator. The min-max optimization problem is thus

$$\min_G \max_D \sum_{i=1}^N [\log D(\mathbf{x}_i) + \log (1 - D(G(\mathbf{z}_i)))] , \quad (4.2)$$

where  $N$  is the number of training images. The proposed method makes use of the min-max idea, but no discriminator is used – instead, the generator and the target model are the competitors.

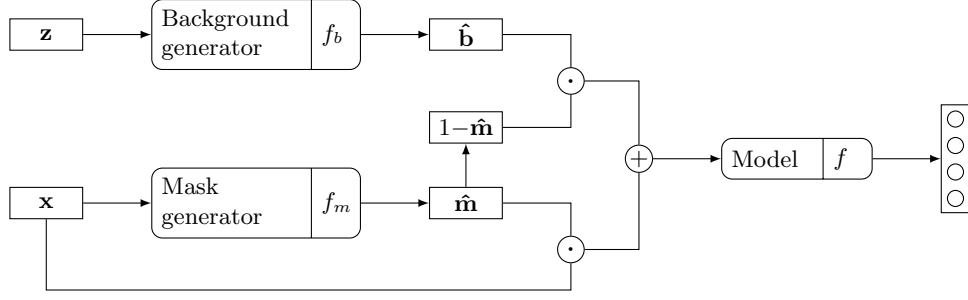
### 4.3 Proposal

The goal is to, during training, be able to place the object in a multitude of contexts (backgrounds), facilitating the learning of robust representations, focusing on “what” the object is rather than “where” the object is. We propose to adopt adversarially generated backgrounds to promote the learning of strong representations. However, the insertion of adversarial backgrounds in the image cannot be allowed to destroy the concept (class) one is trying to learn. Since the spatial delineation of the object is unknown, we propose to learn, simultaneously with the recognition, the segmentation mask. This mask is used to inject the adversarial background only in the non-object pixels.

**Model:** A model  $f$  is optimized to minimize a loss  $\mathcal{L}(f(\mathbf{x}), y)$  using an image  $\mathbf{x}$  as input with label  $y$  as the ground-truth. This image is subject to data augmentation through the process illustrated in Figure 4.5.

In the experiments, we will use cross-entropy for  $\mathcal{L}$  since we focus on classification tasks, but the framework is agnostic of the task and other losses could be used for different tasks: regression, semantic segmentation, reinforcement learning, etc.

**Mask generator:** Firstly, a model  $f_m$  is trained to produce a mask  $\hat{\mathbf{m}}$  using a sigmoid activation function to ensure  $\hat{m}_{ij} \in [0, 1]$ ,  $\forall i, j$ , so that it can be used to segment the image



**Figure 4.5:** Proposed adversarial background augmentation during training.

through a element-wise product,

$$\mathbf{x}' = \mathbf{x} \odot \hat{\mathbf{m}}. \quad (4.3)$$

The model  $f_m$  can be optimized in an unsupervised fashion by finding the mask that minimizes the previous loss,  $\mathcal{L}(f(\mathbf{x} \odot f_m(\mathbf{x})), y)$ . To help prevent the mask from including background, a term  $\mathcal{L}_A$  is used to constraint its size

$$\mathcal{L}_A(\hat{\mathbf{m}}) = \max(0, A(\hat{\mathbf{m}}) - a), \quad (4.4)$$

where  $A$  approximates the percentage of the area of the mask by computing

$$A(\hat{\mathbf{m}}) = \frac{1}{wh} \sum_{i,j} \hat{m}_{ij}, \quad (4.5)$$

and  $a$  is the average area for the object given as an hyperparameter ( $a = 0.2$  is used in all experiments). For better performance, after model  $f_m$  has been trained, a non-differentiable transformation  $t$  can henceforth be applied to further improve the segmentation. For example, a threshold  $T$ ,

$$t(\hat{\mathbf{m}}) = \mathbb{1}_{i,j}(\hat{m}_{ij} \geq T), \quad (4.6)$$

can be chosen using Otsu's method or the  $a$ -quantile such that  $A(t(\hat{\mathbf{m}})) = a$ .

Notice that the mask generator being background invariant is unimportant since it is only used during training and on training images. A typical architecture for the mask generator would be a U-Net [10]. For better results, the mask could be provided by the user through manual segmentation.

**Background generator:** Secondly, the background is generated by a neural network  $f_g$  that transforms noise  $\mathbf{z}$  into a background  $\hat{\mathbf{b}}$  image. Unlike the others, this model is trained to *maximize* the loss  $\mathcal{L}$ . The generator focuses on producing backgrounds or artifacts that could potentially adversely affect the output of the model. However, the scope of realistic backgrounds must be limited. Most importantly, copies of the object may not be generated in the background – this is ensured through the following constraint: the generator produces only a small  $n_b \times n_b$  patch of the background – so  $n_b^2$  independent forward-passes are performed on the generator and the patches are then concatenated,  $\parallel$ , together to produce the background

$$\hat{\mathbf{b}} = \parallel_{j=1}^{n_b^2} f_b(\mathbf{z}_j). \quad (4.7)$$

This avoids unrealistic backgrounds while keeping the pipeline differentiable.

Furthermore, in the case of MNIST and Fashion-MNIST where objects have the same color, the background generator could “cheat” by producing a background with the same color of the object, thus obfuscating the object. In such cases, an additional regularization  $\mathcal{L}_{BA}(\hat{\mathbf{b}})$  term is added to disallow the background from filling over half the pixels,

$$\mathcal{L}_{BA}(\hat{\mathbf{b}}) = \max \left( \frac{1}{N} \sum_{i=1}^N \hat{\mathbf{b}}_i - 0.5, 0 \right). \quad (4.8)$$

All in all, the min-max optimization problem can be summarized as

$$\min_{f, f_m} \max_{f_b} \sum_{i=1}^N \mathcal{L} \left( f \left( \hat{\mathbf{m}}_i \odot \mathbf{x}_i + \left\|_{j=1}^{n_b^2} \hat{\mathbf{b}}_j \odot (1 - \hat{\mathbf{m}}_i) \right\|, y_i \right) + \mathcal{L}_A(\hat{\mathbf{m}}_i). \quad (4.9)$$

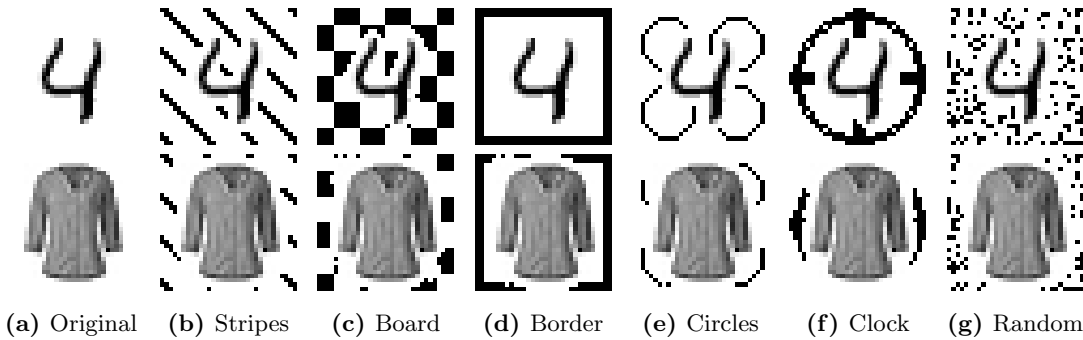
Notice that, while the optimization problem was inspired by GANs, this is not a GAN framework, there is no discriminator used. Also, while this problem could be optimized end-to-end, we have performed this optimization in three stages: (i) train model  $f$ , (ii) train mask generator  $f_m$ , (iii) train both model  $f$  and its adversarial background generator  $f_b$ . Training in stages is useful for debugging and fine-tuning and it also allows applying non-differentiable transformations on top of  $f_m$  such as thresholds to help produce more realistic masks.

## 4.4 Experiments

Two types of experiments are now briefly described: first experiments start by synthetically introducing backgrounds on MNIST and Fashion-MNIST from Table III.3, then the focus becomes the case study discussed in the previous Section 4.1.

### 4.4.1 Synthetic Experiments

MNIST [23] and Fashion-MNIST [21] are artificially enhanced by introducing backgrounds as illustrated in Figure 4.6. There is a one-pixel mask dilation to slightly highlight the object and ensure the performance difference comes from the different context (background) and not because the object itself has become harder to differentiate. This enhanced version will be used only for *testing* purposes, while the original unmodified dataset is used for *training*. The idea is to see how well the model performs when background textures are introduced.



**Figure 4.6:** Backgrounds introduced for MNIST and Fashion-MNIST.

**Table 4.1:** General results (testing accuracy in %).

Method	Original	Stripes	Board	Border	Circles	Clock	Random	Avg
<b>MNIST</b>								
Traditional	97.3	38.0	24.3	61.4	32.9	19.7	11.2	31.2
Attention [71]	93.4	28.1	26.8	57.3	40.1	29.3	25.1	34.5
Proposal	94.9	92.3	76.8	93.1	93.7	70.8	86.2	<b>85.5</b>
<b>Fashion MNIST</b>								
Traditional	90.1	21.3	24.6	36.9	28.5	29.6	16.8	26.7
Attention [71]	81.2	18.2	20.1	51.8	26.0	31.8	36.2	30.7
Proposal	70.7	62.9	61.5	66.5	60.9	60.8	45.9	<b>59.8</b>

**Table 4.2:** Effect of varying the random noise rate in terms of accuracy (%).









								
	0.0	0.01	0.05	0.1	0.2	0.3	0.4	0.5
Baseline	90.1	33.3	13.2	11.2	10.5	10.2	10.2	10.6
Proposal	70.7	70.1	61.6	56.7	45.5	43.2	39.6	35.0
Proposal with manual segmentation	85.4	85.4	84.4	83.4	82.1	80.6	78.1	78.4

Table 4.1 summarizes the results showing the proposal is unbeatable. Interestingly the attention mechanism results only negligibly improve on the baseline classifier. This mechanism works by cropping the image and, not surprisingly, it was found to perform best in the border case (with over 50% accuracy); still, this result was worse than the proposal.

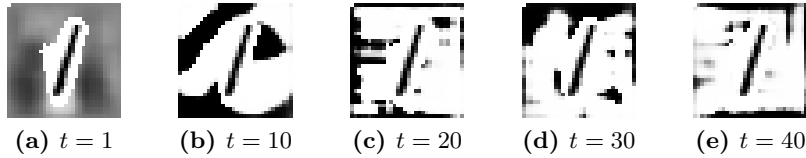
To better understand the impact of changes in the background, let us vary the rate of the random parameter from the previous Figure 4.6 (g). In Table 4.2, a Bernoulli distribution is used for the background with varying parameter values, as illustrated in the images. While the baseline naturally produces better results for the unchanged image, as the rate is increased, the drop in baseline’s performance is precipitous while the proposal drops more smoothly.

Furthermore, since we suspected that the mask generator is the major obstacle in the framework, we also performed experiments using manual segmentation instead of the one produced by the mask generator neural network. The last line of Table 4.2 (“proposal with manual segmentations”) shows much higher accuracy for manual segmentations, confirming that indeed the mask generator is a problem. Two conclusions are evident: (a) the fact that we train the mask generator in an unsupervised fashion means that the masks are imperfect which greatly influence performance, (b) using noise as the background is not sufficient to avoid the network being fooled by more intricate patterns as those in the testing set (Figure 4.6).

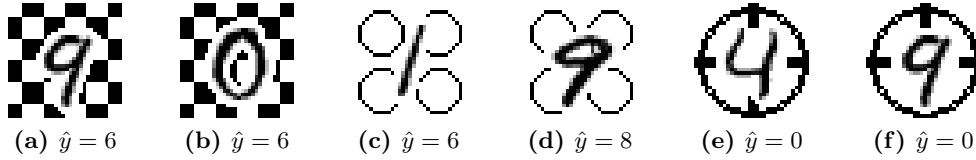
Finally, an example of the dynamic dataset created by the adversarial optimization is shown in Figure 4.7, and some examples of images fooled by the baseline but not by our classifier are shown in Figure 4.8.

#### 4.4.2 Case Study

Experiments were also performed on the aforementioned insulator dataset which motivated the work. A VGG-19, pre-trained using ImageNet, was used as the classifier, which had



**Figure 4.7:** Examples of the dynamic background along the epochs.



**Figure 4.8:** Examples of misclassifications made by the traditionally-trained classifier, but not by the adversarially-trained one.

previously been found to have good performance for this dataset [75].

The baseline arrives at an accuracy of 72% which the proposal can raise to 89%, as shown in Table 4.3. This is almost as good as using real background images, as described in Section 4.1, which arrived at a 94% accuracy (last column). Using random noise, instead of our generator, degrades performance to 60%, worse than just using the training set, showing that the background generator is indeed helping.

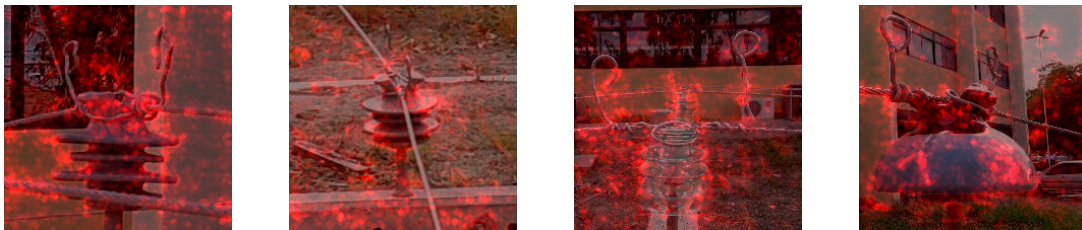
Finally, sensitivity analysis contrasts the baseline with the adversarially-trained classifier – the impact is illustrated in Figures 4.9 and 4.10. The heatmaps are produced by normalizing the gradients  $\frac{\partial \hat{y}}{\partial \mathbf{x}}$ , and taking the pixel-wise maximum of each channel. They clearly illustrate the fact that the proposed method makes the classifier significantly less sensitive to the target background.

**Table 4.3:** Results for drone case study.

Method	Traditional	Attention [71]	Proposal	Noise background	Real backgrounds
<b>Accuracy (%)</b>	71.9	45.8	88.7	59.6	93.8

## 4.5 Summary

Motivated by the problem of training drones with the in-doors images while generalizing to the out-doors, an adversarially trained model is proposed that is invariant to the back-



**Figure 4.9:** Sensitivity analysis for the naive method, with gradients in red.



**Figure 4.10:** Sensitivity analysis for the mask-inference method, with gradients in red.

ground. During training, while the target model tries to minimize its loss, a background generator augments the data by trying to search for backgrounds that maximize the target model’s loss, thus making the target model robust to background changes. The method was evaluated using synthetic datasets and a real dataset.

The proposal was evaluated using classifiers, but could potentially be used for other tasks involving a CNN, such as regression problems, segmentation, or reinforcement learning tasks.

**Publications:** This work was prompted by a collaboration with Ph.D. colleague Ricardo Prates of Univasf, Brazil, during his stay at INESC TEC. The initial physical collaboration involved mostly engineering work, building a process by which out-door visual elements were introduced in the in-doors through automatic heuristics:

1. R. M. Prates, R. Cruz, A. P. Marotta, R. P. Ramos, E. F. S. Filho, and J. S. Cardoso, “Insulator Visual Non-conformity Detection in Overhead Power Distribution Lines using Deep Learning,” in *Computer and Electrical Engineering*, Elsevier, 2019. [doi: 10.1016/j.compeleceng.2019.08.001]

The general-purpose way to train a classifier using adversarial training was later proposed in:

2. R. Cruz, R. M. Prates, E. F. S. Filho, J. F. P. Costa, and J. S. Cardoso, “Background Invariance by Adversarial Learning,” in *25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021. [accepted]

A preliminary version of this work earned the best oral presentation award at RECPAD 2020, a national conference – that paper is not included here since the proceedings of this conference are not indexed. Current plans are to publish a co-joint paper with Ricardo Prates combining the adversarially generated backgrounds with the insulators (foreground) themselves generated by a GAN. This might make training more resilient, and be especially useful in cases of condition imbalance, i.e. some of the insulators have been recently introduced to the network and, therefore, no or little examples exist of them with defects.

Classic adversarial training focuses solely on making classifiers robust to small image perturbations, often called adversarial examples. This framework opens a new line of research and it is inspired by another work from within Professor Jaime S. Cardoso’s group: classifiers trained to recognize sign language, the language used by people who are deaf, often focus on the person doing the sign (undesirable) rather than his/her hand (desirable) causing overfitting. Adversarial training had been used to train a person-invariant classifier [79]. The details are, however, very different from ours since they used a Variational Autoencoder. Since our work, promising experiments have been performed to use adversarial training to make various biometric models robust to intrusion – for example,

anti-spoofing measures may be improved if the spoofing detector is made invariant to the type of material used by the adversary [80].



## Active Supervision

In supervised learning, the role of the data practitioner is to collect and label data. After training, the misclassified classes may be identified and the practitioner can improve performance by (a) coarsely tuning the model by tweaking hyperparameters or by (b) collecting more data from the real world to improve classification of the disadvantaged classes, a costly process known as *active learning* [81].

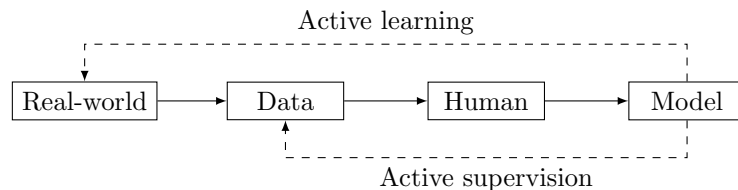
The proposal is to short-circuit active learning by acquiring data from the model itself – the model can interpolate or extrapolate from the existing data and have the practitioner label the new data. The model itself could be used to identify and manipulate images near the decision boundary in a direction orthogonal to the boundary. We have called this process *active supervision*, as illustrated by Figure 5.1.

The proposal is found to be especially beneficial in the context of few-shot learning, when there are very few cases of one of the classes. In the experiments, only 10 images of the digit “2” of MNIST are used (and 6,000 of each other digit), then active supervision is used to significantly improve recognition of this digit.

### 5.1 Related Work

The distance of the data relative to the decision boundary has been found to be a major predictor of overfitting [82].

This problem can be ameliorated in an unsupervised manner: by using adversarial training [83, 84] or by manifold interpolations [85]. The goal of both techniques is to make the boundary smoother and avoid small perturbations from percolating into big decision changes. A typical approach involves minimizing the loss  $\mathcal{L}$  both for data  $\mathbf{x}$  and also for  $\mathbf{x} + \delta$ , where  $\delta$  is Gaussian noise. However, these techniques make the model insensitive to small changes in the input, which sometimes is undesirable – think of breast cancer detection or melanoma classification which involve very small changes in morphology. While such techniques produce more resilient models that are harder to fool, most often they come at the cost of performance [83]. It would be highly desirable for the user to



**Figure 5.1:** Active supervision short-circuits the traditional active learning pipeline.

actively observe and classify the perturbations as they are performed during training; albeit that would be of course infeasible.

Another line of approaches is to use the distance to the decision boundary to ascertain uncertainty and request the data practitioner to focus labeling on the data where uncertainty is highest, a process known as active learning [81]. However, unless there is an existing pool of unlabeled data, this process requires collecting more data from the real-world which can be highly expensive. In the next section, we propose combining active learning with data perturbations during training.

## 5.2 Proposal

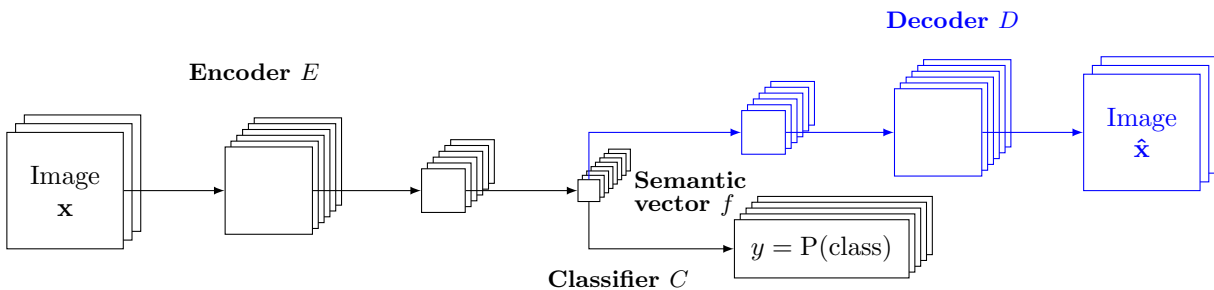
The proposal combines active learning with perturbations on the images, as previously described. The perturbations are introduced in the semantic layers and then reversed to the original image until the model produces fuzzy predictions, and the user is then asked for a label.

**Architecture:** A traditional classifier can be seen as having two components: an **encoder**  $E$  and a **classifier**  $C$  component. In the encoding phase, convolutions are applied until a (semantic) vector captures the essential semantics of the image. Then, the classifier consists of dense layers that transform this semantic vector into  $K$  outputs neurons, at which point a softmax is applied to ensure a probability distribution. Since AlexNet [1], most deep classifiers have been using this formula, including popular architectures such as VGG and ResNet.

Our proposal extends this model by adding a **decoder**  $D$ , as depicted in Figure 5.2 – this decoder is responsible to create new images from the semantic vector. The output of the decoder is an image of the same size as the original image; the goal is that it learns to make the neural network invertible. This extra component is an extension that can be trained independently of the rest of the classifier. An alternative to introducing a decoder would have been to back-propagate the feature vector back to the original images; however, in our experience, that typically produces poorer images.

**Training:** The previous architecture can be trained as follows: The encoder  $E$  and classifier  $C$  part are trained for the supervised task. Typically, the classifier produces a probability  $\hat{y}_{i,k} = P(k|\mathbf{x}_i) = C(E(\mathbf{x}_i))$  of each image  $\mathbf{x}_i$  being of class  $k$ , and the loss to minimize is cross-entropy where each  $y_{i,k}$  is an one-hot representation of the real class,

$$\mathcal{L}_{\text{CE}} = - \sum_{k=1}^K \sum_{i=1}^N y_{i,k} \log \hat{y}_{i,k}. \quad (5.1)$$



**Figure 5.2:** The classifier (in black) is extended so that a decoder (in blue) reproduced the image using the same semantic vector.

Afterwards, the decoder  $D$  is trained to produce an invertible representation of the neural network, i.e. the goal is for the decoder to be the inverse function of the encoder,  $D = E^{-1}$ . Both are optimized so that the output image  $\hat{\mathbf{x}}_i$  is the same as the input image  $\mathbf{x}_i$ , i.e.  $\hat{\mathbf{x}}_i = D(E(\mathbf{x}_i)) \approx \mathbf{x}_i$ . This correspondence is approximated using L2 as traditionally done when training autoencoders,

$$\mathcal{L}_D = \sum_{i=1}^N \|\phi(\mathbf{x}_i) - \phi(\hat{\mathbf{x}}_i)\|_2^2. \quad (5.2)$$

A function  $\phi$  may be applied to both the original and the produced image in order to correspond higher-level features rather than force pixel-wise equivalence. This is known as a perceptual loss [86]. In the experiments, the identity function is used as  $\phi$ .

**User Feedback:** After the previously described training phase, user feedback can be used to improve the classifier. In this phase, the goal is to have the user indirectly tune the decision boundary of the classifier by finding an observation of one class and manipulating it until the model thinks it has cross the boundary into a different class. The user is then asked about the class and the feedback is used to gradually reshape the decision boundary. The detailed procedure is shown in Algorithm 1.

---

**Algorithm 1** Training algorithm using user feedback

---

```

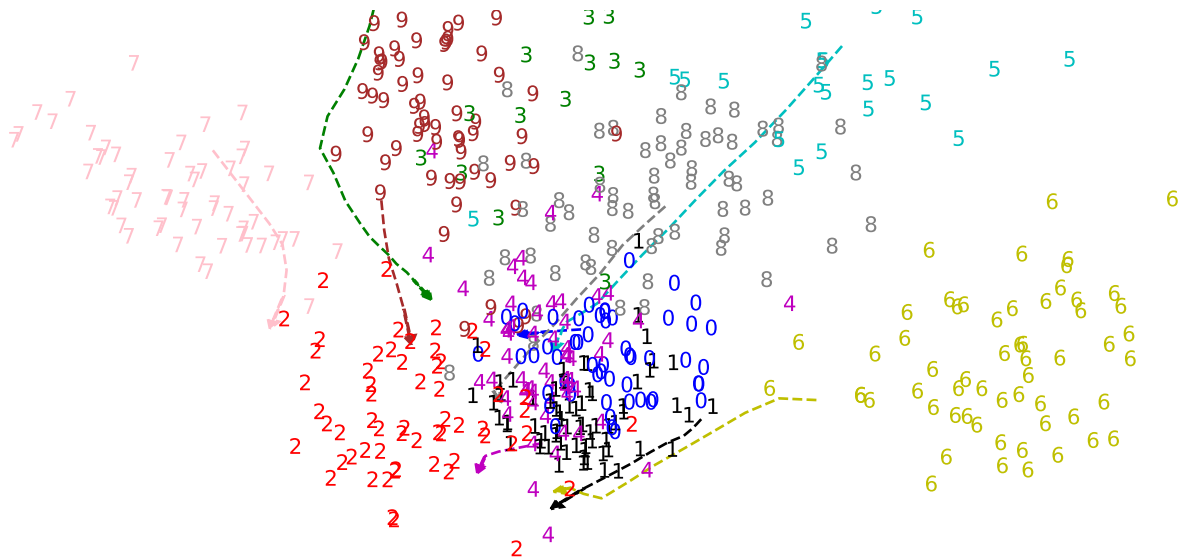
1: Input:  $\mathbf{x}_i, y_i, \bar{y}_i$ 
2: Output:  $\bar{\mathbf{x}}_i$ 
   // Produce interpolations of  $\mathbf{x}_i$  from  $y_i$  to desired class  $\bar{y}_i$ 
3:  $j \leftarrow 0$ 
4:  $\mathbf{v}^{(j)} \leftarrow E(\mathbf{x}_i)$ 
5: while  $(\hat{y}_i \leftarrow C(\mathbf{v}^{(j)})) \neq \bar{y}_i$  do
6:    $\mathbf{v}^{(j+1)} \leftarrow \mathbf{v}^{(j)} + \eta \frac{\partial \hat{y}_i}{\partial \mathbf{v}^{(j)}}$ 
7:    $j \leftarrow j + 1$ 
8:    $\hat{\mathbf{x}}_i^{(j)} \leftarrow D(\mathbf{v}^{(j)})$ 
9: end while
   // Ask user which images still belong to the original class  $y_i$ 
10:  $j \leftarrow \text{user\_feedback}(\hat{\mathbf{x}}_i^{(1)}, \dots, \hat{\mathbf{x}}_i^{(j)})$ 
11:  $\bar{\mathbf{x}}_i \leftarrow (\hat{\mathbf{x}}_i^{(1)}, \dots, \hat{\mathbf{x}}_i^{(j)})$ 
   // Update weights based on user feedback
12: for  $t \leftarrow 1$  to  $j$  do
13:   while  $(\hat{y}_i^{(t)} \leftarrow C(E(\bar{\mathbf{x}}_i^{(t)}))) \neq t$  do
14:      $\mathbf{w}_{E,C} \leftarrow \mathbf{w}_{E,C} - \eta \frac{\partial \mathcal{L}_{CE}(\hat{y}_i^{(t)}, \bar{y}_i)}{\partial \mathbf{w}_{E,C}}$ 
15:   end while
16: end for

```

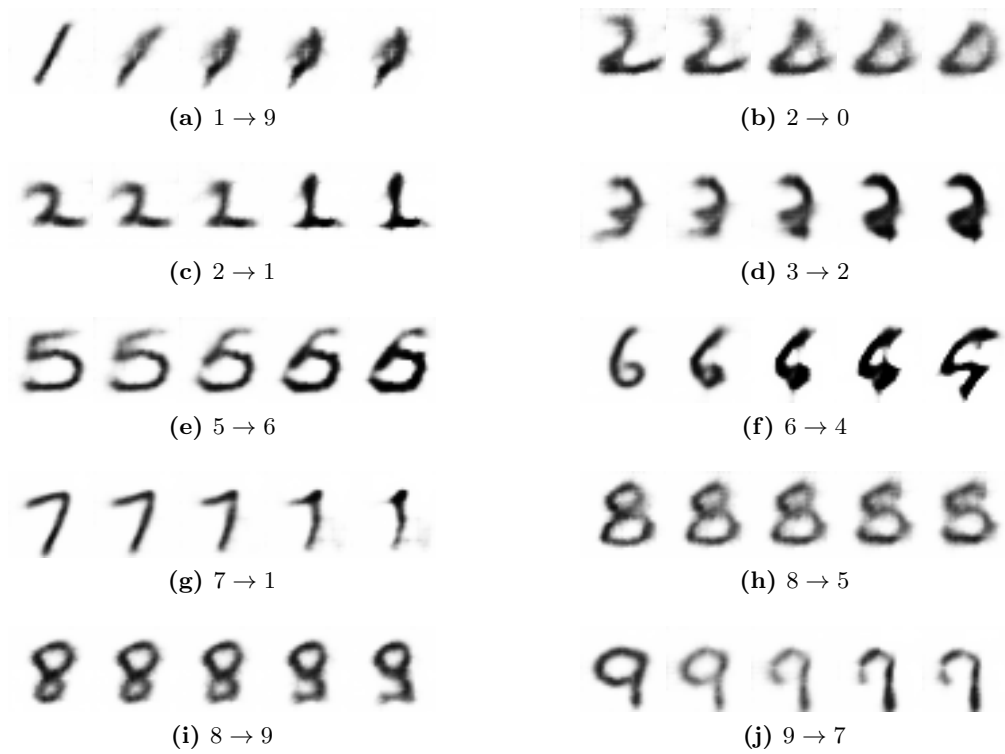
---

Various mechanisms of user feedback (line 10) could have been used – in the experiments, we just request the user to select the images which have been correctly transformed to the target class, i.e. crosses over the decision boundary. The model is then updated for several iterations using a bigger weight for these new images.

A visual example of how the feature space of random observations travel in the direction of class “2” of MNIST (lines 3–9) is depicted in Figure 5.3. For illustration purposes, PCA was performed to reduce the feature space to two-dimensions, and the dataset used is MNIST. Furthermore, Figure 5.4 shows results of the decoder (line 8) for several class extrapolations.



**Figure 5.3:** PCA of latent space of a MNIST classifier showing the transition between each class clusters and class “2”.



**Figure 5.4:** Examples of several extrapolations for MNIST.

### 5.3 Experiments

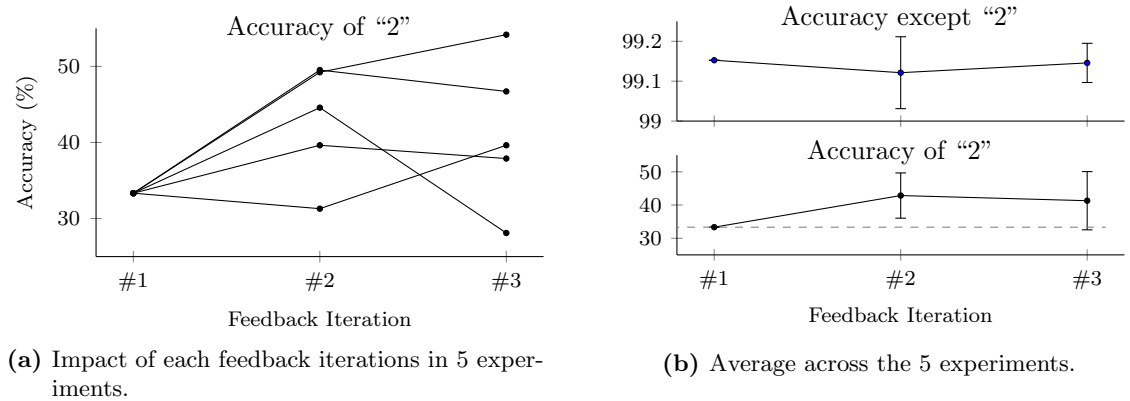
The MNIST digits dataset was used for the experiments. This dataset comprises 6,000 training examples of each one of the ten Indo-Arabic digits (0–9) hand-written; with also another 1,000 examples of each digit for testing purposes. Since LeNet (see Section II.1), it has been trivial to obtain an accuracy of over 99% for this dataset. Therefore, we down-sample the number of digits “2” from 6,000 images to only 10 images. Subsequently, we train a classifier for 1,000 epochs to ensure convergence and obtain a base accuracy of 33% for the digit “2” and of over 99% for all other digits. High rates of accuracy are common for this dataset.

The implementation of the architecture shown in Figure 5.2 uses an encoder that applies three convolutions with a stride of two, with each  $i$ -th layer having  $2^{4+i}$  filters. Thus, the original  $32 \times 32 \times 1$  images are reduced to a tensor of size  $4 \times 4 \times 128$ . The classifier flattens this tensor and applies a dense layer with 256 neurons followed by another with the 10 output classes. The decoder is the opposite of the encoder with transpose convolutions. All filters are  $3 \times 3$  and Adam was used as the optimizer.

After the base model is trained, Algorithm 1 is performed: ten random images of each one of the ten digits are extrapolated to convert them into something that the model considers a “2”, and ten intermediate extrapolations are used. All in all, there are 1,000 images from which to choose from, but the user typically chooses only a small fraction of these. The new images selected by the user are then incorporated and the model is optimized for 100 epochs. The new images are given a weight of 10 times the weight of the normal images.

Results are presented for 5 different experiments as Figure 5.5 using the base model with an accuracy of 33% for digit “2” at iteration #1. Then, user feedback is integrated for two iterations. The first thing to notice is how disparate the results are (Figure 5.5 (a)). In all cases, active supervision improves results dramatically, but somehow erratically since the process is time-consuming (albeit less than active learning would be) and the number of new labeled images has to be finite. A less gleeful result is that after the first iteration, performance plateaus and seems to, if anything, to decrease.

Average results are shown in Figure 5.5 (b). The average accuracy gain was of 9.5% for the target digit “2”. This could have conceivably come at the cost of misclassifying the other digits, but actually, no. The average accuracy for the other digits almost does not budge, and it increases in some cases.



**Figure 5.5:** Results of active supervision using MNIST to augment digit “2”.

## 5.4 Summary

The proposal is to introduce the user into the learning loop. The proposal consists in using the model itself to modify images across the decision boundary so that the user is able to more directly tune the boundary once the model has been trained. The process intersects with previous work on class imbalance, data augmentation, and optimization – the work was shown to be beneficial in an experiment where the dataset was augmented with the help of the user in the context of absolute class imbalance (few-shot learning).

The work has gone through several major revisions for a long time. This is because there are a few non-trivial difficulties associated with this process that are not immediately apparent and should be disclosed:

- (i) One inherent flaw in the concept is that, on the one hand, it requires enough data for the extrapolation to be trained and work seamlessly, but on the other hand, the classification component is easier to train than the extrapolation one, therefore, if we have enough data to train the extrapolation component, then the impact of the method in the classifier is almost minuscule to even be measured. For the purpose of this chapter, the problem was overcome by limiting this solution to the less ambitious context of few-shot learning. This context is perfect for this process since there is enough data to train a robust extrapolation component, but there is one class on which to focus efforts. How to generalize the process to other contexts is an open problem. Possibilities could involve using transfer-learning or by changing the way feedback is given and the way that feedback is integrated, which takes us to:
- (ii) There are many different ways to conceive each one of the components, some of which we alluded to, such as using back-propagation to extrapolate rather than a decoder, but there are also many conceivable ways how the way feedback may be requested and how it may be integrated into the optimization process. Small changes to the feedback process could include ranking or even do interpolations between two images rather than an extrapolation from one image. More interesting would be looking for different ways of how this feedback could be integrated; one possibility is to use only the direction that the user prefers, which would avoid having to create perfect images, another would be to ask for feedback on the decision gradients for each pixel. Pixel-level feedback is an interesting idea – the user could specify that some regions should not be considered for the decision – but it is trickier than it sounds: experiments using CIFAR-10 show that the user tends to underestimate how important things like background features are for classifying objects (for example, clouds are associated with airplanes and water with boats). Still, it is a promising line of approach for several problems.
- (iii) While this technique of active supervision is naturally less time-consuming than active learning, as we argued at the opening of the chapter, the technique itself is very time-consuming to research. For active learning, researchers already have a pre-arranged dataset for which they simulate the acquisition. On this line of research, any change, small as it may be, requires redoing experiments, each of which is very time consuming since it requires continuous user feedback, and must be repeated more than once to ensure reproducibility.

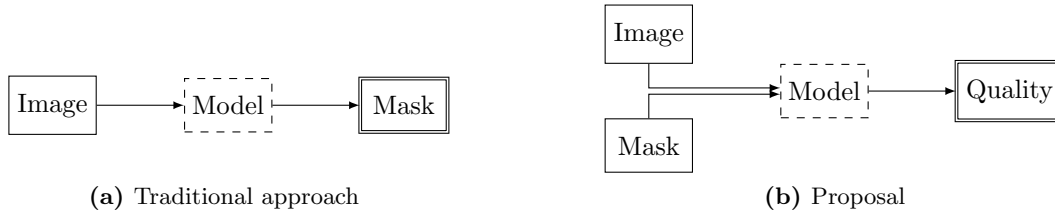
This process of active supervision was developed together with ASM Shihavuddin, originally at DTU in Denmark, and now at the Green University of Bangladesh. Despite the aforementioned problems, we feel there is potential for the sprout of a new line of research and for real-world applications.

## Iterative Inference

Segmentation of images into its constituent parts is a decades-old problem. Traditional methods range from the usage of color threshold to clustering, and iterative methods such as region growing and active contours. However, all these methods require strong human supervision and tuning to find the right parameters.

The advent of machine learning, in particular convolutional neural networks like SegNet [87], has allowed semantic segmentation – where the parameters of the model are optimized automatically in a supervised manner on the object of interest. These new methods lack the iterative nature of previous techniques. The downside of such methods is the great amount of data required for training.

In this chapter, a novel segmentation paradigm is presented: the convolutional neural network is trained to learn the quality of an image-segmentation pair. After training, the segmentation process makes use of the network as an oracle of the current segmentation quality to refine the mask in an iterative fashion using gradient ascent. The proposal is illustrated in Figure 6.1.



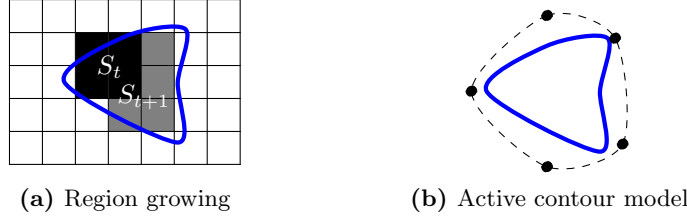
**Figure 6.1:** Diagram contrasting the proposal to the traditional approach.

### 6.1 Related Work

Many traditional computer vision techniques have involved iterative processes. This is the case, for example, of region growing and active contours (also known as snakes).

- **Region growing** (Figure 6.2 (a)) methods [88] are based on the principle of clustering pixels with similar properties to form a homogenous region. An image  $\mathbf{x}$  is divided by regions,  $\bigcup_{i=1}^n R_i = \mathbf{x}$  and  $R_i \cap R_j = \emptyset$ ,  $i \neq j$ , and each region  $R_i$  is connected to adjacent regions given by  $N(R_i)$ . Typically, each region is a pixel of the image and the adjacent regions are the 4-neighbors or 8-neighbors. An initial segmentation  $S_1$  (seed) grows to include its neighbors according to a predicate rule  $P(R)$  that evaluates whether region  $R$  is homogeneous or not,

$$S_{t+1} = S_t \cup \{R_i \mid R_i \in N(S_t), P(R_i \cup S_t)\}. \quad (6.1)$$



**Figure 6.2:** Illustration of two iterative segmentation methods.

The method converges when  $S_t = S_{t+1}$ . A common criteria for the predicate is whether the average gray-scale level is below a certain threshold  $T$ ,  $P(R) = \bar{R} < T$ . An alternative method is called *region splitting*, which works the other way around.

- **Active contour** [89] (Figure 6.2 (b)), also known as snake, is a curve represented by  $\mathbf{v}(s)$ , composed of discrete points, indexed by  $s \in [0, 1]$ . This curve starts off with a given shape (seed) and its shape is modified by minimizing an energy function

$$E_{\text{snake}} = \int_0^1 [E_{\text{internal}}(\mathbf{v}(s)) + E_{\text{external}}(\mathbf{v}(s))] ds, \quad (6.2)$$

where  $E_{\text{internal}}$  acts as a regularizer punishing oscillations in the curve, and  $E_{\text{external}}$  takes in consideration the image and corresponds to the intensity or gradients of the image. For example,  $E_{\text{external}} = \mathbf{v}(s)$  would grow the curve iteratively along the brighter parts of the image. Several proposals exist on how the curve should be initialized.

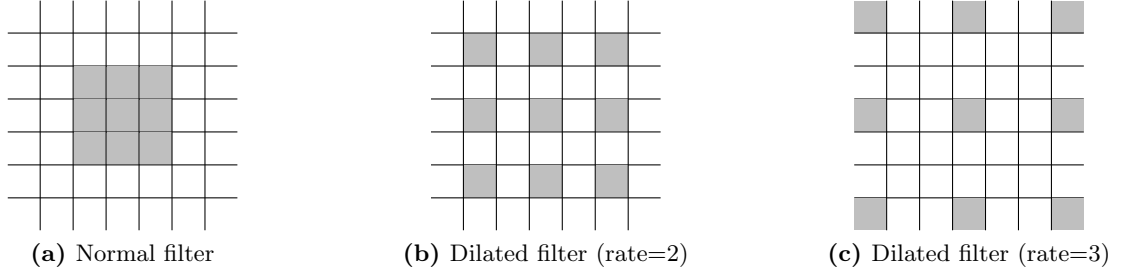
These traditional techniques have recently been surpassed by convolutional neural networks, as described in Section II.3. For the practitioner, the difference is that a training set is required to guide the learning process and inference happens in a single step as in Figure 6.1 (a), i.e. they are no longer iterative. The most widely used architectures are based on an **encoder-decoder** two-phased neural network; the image is first compressed into a smaller semantic representation, usually using convolutions and pooling (the encoding phase), and then decompressed into the final segmentation, usually using transposed convolutions (the decoding phase). The first example of this was SegNet [87].

A big problem in this architecture is the so-called checkerboard problem: some detail is lost during the encoding step, which prevents the decoding step from doing as good a job as it could in refining the segmentation. That helps explain the popularity of U-Net [10] which takes advantage of the symmetry nature of the encoder-decoder with “skip-layers”, as previously discussed and illustrated in Figure II.2. U-Net is one of our baselines.

Another important landmark in avoiding checkerboard effects are **dilated kernels**, originally known as atrous convolutions. DeepLab [90], which makes use of such kernels, ranks first place in many benchmarks, including PASCAL VOC (Table III.4). Previous architectures, integrated spacial information by making activation maps smaller and smaller during the encoding phase, which were then restored during the decoding phase. In this architecture, there are no distinct encoding and decoding phases, and all activation maps keep the size of the original image. Instead, spatial information is integrated by dilating the filters themselves, as illustrated in Figure 6.3. DeepLab is also used as a baseline.

Also, worth mentioning is that iterative segmentation already exists in the form of recurrent neural networks adapted for segmentation [91]. The current work is innovative in that it is far simpler than any such previous approach since it most resembles classical



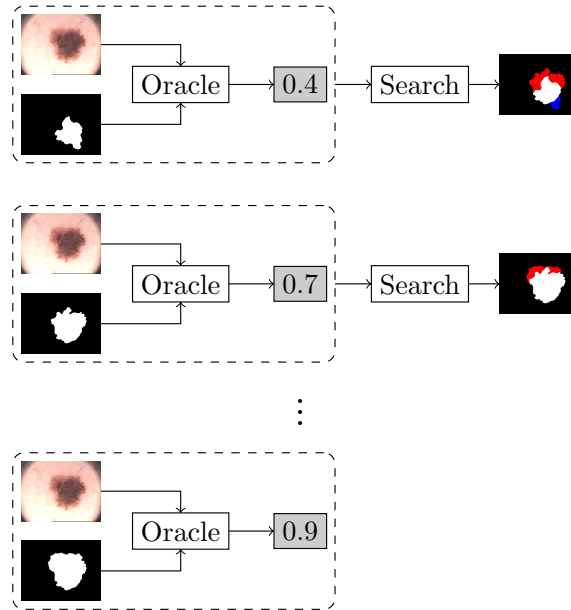


**Figure 6.3:** Dilated kernels avoid the checkerboard effect.

methods used for segmentation. The proposed model is illustrated in Figure 6.1 (b) and is presented in the next section.

## 6.2 Proposal

The proposed model takes an image-segmentation pair and outputs a quality score. The model may be seen as an *oracle*. This oracle is then used to guide the iterative search process to change the segmentation in the direction that increases the quality score. Any number of methods could be used for the search; since a CNN is used as the oracle, the process is differentiable, and we use gradient ascent. Figure 6.4 illustrates the process.



**Figure 6.4:** The quality measure produced by the model (oracle) guides the iterative process of segmentation inference.

In order to simplify the learning process (i.e. decision models and optimization), the neural network models the quality of a segmentation mask as a scoring function. This is a quality  $q_i$  representing the correspondence between an image  $\mathbf{x}_i$  and a segmentation  $\mathbf{y}_i$ .

This can be learnt by optimizing

$$\min_{\mathbf{w}} \sum_{i=1}^N \mathcal{L}_{\mathbf{w}}(q_i, \hat{q}_i), \quad (6.3)$$

where  $\hat{q}_i$  is the quality predicted by the neural network,  $\hat{q}_i = f(\mathbf{x}_i, \mathbf{y}_i)$ , and  $\mathcal{L}_{\mathbf{w}}$  can be instantiated to be the squared error, cross-entropy, etc.

The hope is that the complexity of the optimization problem is reduced since the output, instead of being of size  $2^{wh}$  which is the area of the entire segmentation probability, is a single number corresponding to the quality of the given segmentation. Only an encoder architecture is now necessary, instead of an encoder-decoder.

**Inference:** An initial segmentation  $\hat{\mathbf{y}}_i$  must be given, which could be empty (zeros), and is then updated by gradient ascent using backpropagation in order to maximize the predicted quality  $\hat{q}_i$ ,

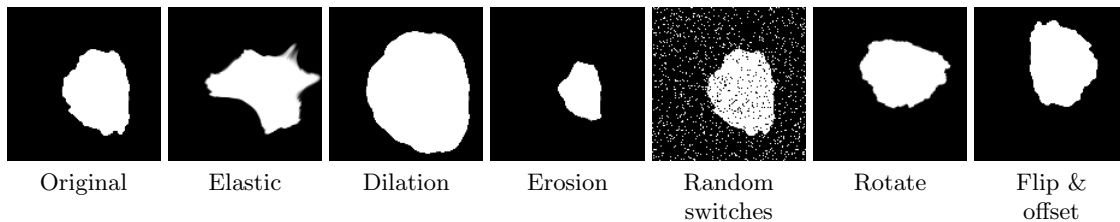
$$\hat{\mathbf{y}}_i \leftarrow \hat{\mathbf{y}}_i + \alpha \frac{\partial \hat{q}_i}{\partial \hat{\mathbf{y}}_i}. \quad (6.4)$$

An alternative to backpropagation would have been to use an exhaustive or heuristic exploration of the search space of segmentations using the network as an oracle. While these techniques would be able to discover non-local improvements, backpropagation stands as an efficient exploration strategy when the decision function is known and  $C^1$ .

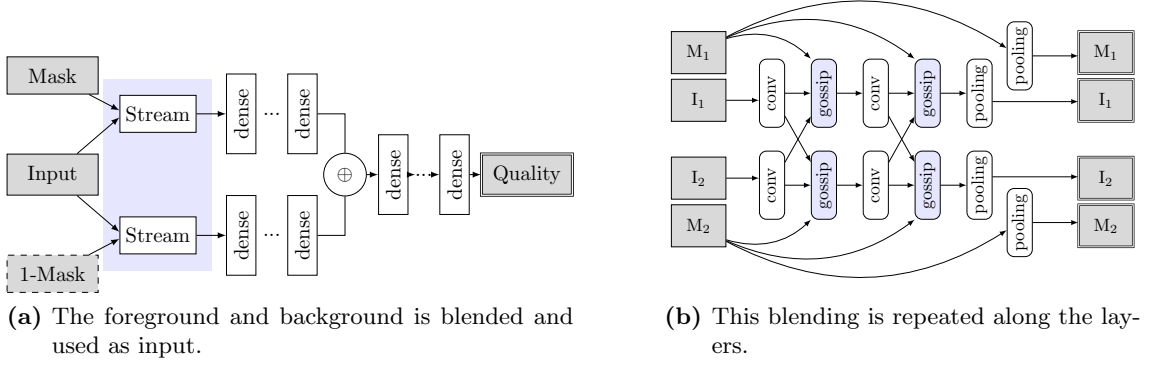
**Training:** This is the tricky part, since we must teach the network what a good and a bad segmentation is. For that, different transformations (see Figure 6.5) are applied to the known segmentation ( $\mathbf{y}_i$ ) and a new segmentation is created ( $\tilde{\mathbf{y}}_i$ ). The quality measure  $q_i$  is then computed between the real and the transformed segmentation. Dice coefficient was used to compute  $q_i$ , which corresponds to the intersection over the union of the segmentations,

$$q(\mathbf{y}_i, \tilde{\mathbf{y}}_i) = \frac{2|\mathbf{y}_i \cap \tilde{\mathbf{y}}_i|}{|\mathbf{y}_i| + |\tilde{\mathbf{y}}_i|}. \quad (6.5)$$

It is desirable to teach the network using a balanced range of qualities of Dice, therefore the impact of the transformations parameters and Dice was empirically estimated. For each transformation, the parameters are drawn by grid-search, and a similarity index  $D(\mathbf{y}, \tilde{\mathbf{y}})$  is computed between the ground-truth segmentation  $\mathbf{y}$  and the synthetically created  $\tilde{\mathbf{y}}$ . Dice was discretized into  $B$  bins (in our case,  $B = 8$ ). A frequency distribution  $p_{ib}$  was then found that represents how many times the parameter combination  $i$  resulted in Dice  $b$ . A second distribution  $q_{ib}$  can then be computed to find the proportion of each transformation  $i$  that produces Dice  $b$ , by minimizing a system composed of  $B$  equations,  $\sum_i p_{ib} q_{ib} = \frac{1}{B}$  for each  $b$ . This was solved as a non-negative least squares problem.



**Figure 6.5:** Examples of synthetically created segmentations.



**Figure 6.6:** Oracle network architecture in detail.

The optimization problem from (6.3) can then be solved by gradient descent,

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{L}_{\mathbf{w}}}{\partial \mathbf{w}}. \quad (6.6)$$

Notice how training and inference use the same optimization procedure, see (6.4) and (6.6).

**Architecture:** To ease the optimization process, instead of concatenating  $\mathbf{x}_i$  and  $\mathbf{y}_i$ , we first perform element-wise multiplication to extract the foreground ( $\mathbf{f}_i = \mathbf{x}_i \otimes \mathbf{y}_i$ ) and the same on the segmentation inverse to extract the background ( $\mathbf{b}_i = \mathbf{x}_i \otimes (1 - \mathbf{y}_i)$ ), as illustrated by Figure 6.6 (a). Both of these are then given to an encoder to produce the quality score. This blending of the image and mask is also performed along the layers to allow streams to communicate (“gossip”) in order to combine information from both streams, as in Figure 6.6 (b).

Furthermore, since the propagation of gradients through max-pooling blocks is sparse, we have found better results using average pooling which ensures gradients are propagated through all the pixels in the block.

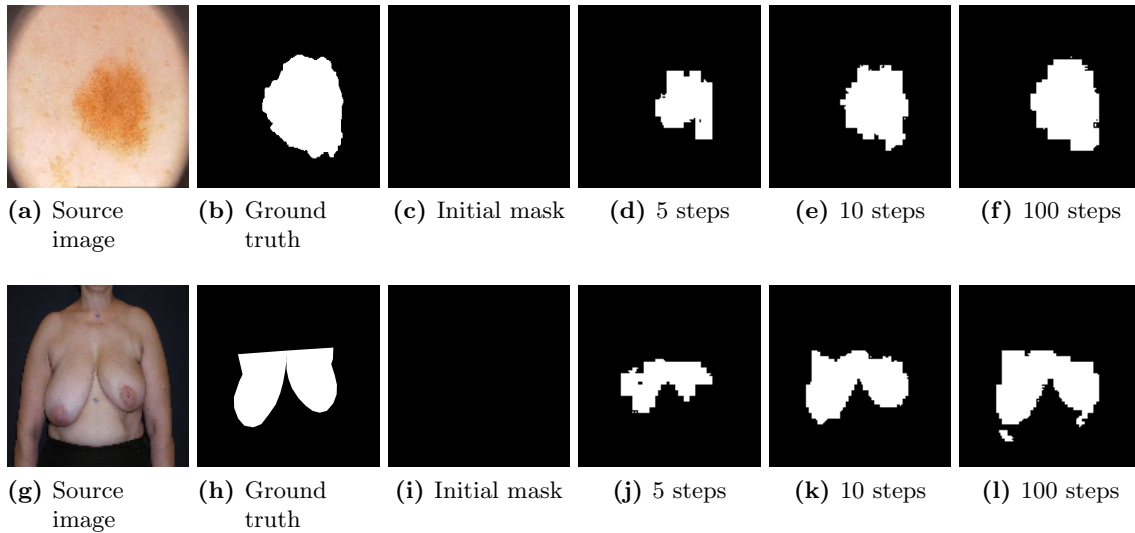
## 6.3 Experiments

The datasets used for the experiments can be found in Table 6.1, with an extended version at Appendix A.6. The proposal is contrasted with the state-of-the-art U-Net [10] and U-Net with dilated convolutions (DilatedNet) [90].

The architecture consisted of several conv-conv-pool blocks, the number of repetitions was chosen by the validation set. We use 32 filters on the first convolutional layer and double the value on each level as suggested by U-Net [10]. ReLU activations were used on the intermediate dense layers and a sigmoid activation on the final layer to predict a quality value between 0 and 1. The loss function for the oracle network was the mean

**Table 6.1:** Results in terms of Dice coefficient (%).

	U-Net	Dilated	Proposal
<b>Average</b>	80.5	80.7	<b>82.5</b>
<b>% Top-1</b>	12	0	<b>88</b>
<b>% Top-2</b>	38	75	<b>100</b>



**Figure 6.7:** Iterative refinement of images from PH2 and Breast Aesthetics datasets, respectively. Initial masks are completely void.

squared error, optimized using Adam for 500 epochs with early-stopping after 50 iterations without improvement, and the best validation model was used.

The proposal is shown to be slightly superior to the baselines. Furthermore, models were also trained in one dataset and evaluated in another: on average, Dice coefficients were 61% (U-Net), 59% (Dilated-Net), and 63% (proposal).

Then, we explored the performance of the model in the most extreme scenario, where the initial segmentation is completely empty (i.e. black). Figure 6.7 illustrates results after a few steps of refinement. The network converges to a good solution on about 20 iterations. The remaining steps of the optimization focus on minor details with little impact on the overall performance. The proposed approach emulates the traditional region-growing strategy [88], where the segmentation is progressively extended.

## 6.4 Summary

This chapter addresses the problem of semantic image segmentation with deep neural networks. We propose a new paradigm, based on similarity learning techniques, where the model tries to learn a quality function that maps an image-mask pair to the corresponding segmentation quality. Using the proposed architecture and, in combination with backpropagation, the proposed strategy is able to improve segmentation masks by maximizing the expected quality. By framing the problem as a regression task, the output complexity is reduced.

**Publications:** This chapter was a collaboration with Ph.D. colleague Kelwin Fernandes and was published at IJCNN:

1. K. Fernandes, R. Cruz, and J. S. Cardoso, “Deep Image Segmentation by Quality Inference,” in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018. [doi: 10.1109/IJCNN.2018.8489696]

A master’s thesis exploring further experiments was written by José Soares Rebelo, “CNN-Based Refinement for Image Segmentation” (2018). So far, four papers from other authors

have cited and mentioned this work in their state-of-the-art.

While the method was proposed and evaluated in the context of semantic segmentation, it could potentially be extended to other tasks such as signal processing or NLP. For example, generating text guided by a quality metric.



## Conclusion

Several titles were considered for the thesis: rethink, redesign, out-of-box, divergent. Throughout the Ph.D., several encrusted ideas from machine learning were challenged and new solutions proposed. The scope of contributions involves the entire learning process, also known as the *pipeline* (Figure IV.1). Most of these actually seem to outperform the state-of-the-art, or at least look like promising trailblazers. They have included ideas on automatic data augmentation (Chapter 1), class imbalance using pairwise learning (Chapter 2), a new way to look at risk optimization (Chapter 3), training for background invariance (Chapter 4), active supervision (Chapter 5), and iterative inference (Chapter 6). A general conclusion of these works is now provided, as is when the work was performed.

### IV.1 Summary

*Data augmentation through Hill Climbing* (2019) trains models in parallel to perform hyperparameter search to find the best augmentation policies. The method only doubles training time, and is shown to be beneficial for two reasons: (i) the hyperparameter search itself (static policy), and (ii) the dynamic process itself of gradually increasing augmentation intensity (dynamic policy). Results were similar to traditional Bayesian search for 4% of the training epochs.

*Class Imbalance using Pairwise Learning* (2016–2018) proposed a new family of solutions for an age-old problem for binary and ordinal classification tasks, and also for survival analysis. Pairwise models were borrowed from the learning to rank literature which learns two observations in unison. There was an overall 15% improvement relative to the state-of-the-art, as evaluated by a balanced metric.

*Risk Aversion* (2017, 2019) is a new way to look at how costs are considered in modeling. In the context of binary classification, two types of errors are possible: false positives and false negatives. Typically, the trade-off between the two types of errors is defined as a relative trade-off using a cost matrix. Several approaches are proposed to define this trade-off in terms of an absolute constraint on one type of error while trying to minimize the other. These methods are applied for binary classification and segmentation.

*Background Invariance* (2019–2020) is, as far as we know, the first work that proposes a method at making neural networks robust to changes in the background. The proposed method is an end-to-end method that augments the training set by introducing new

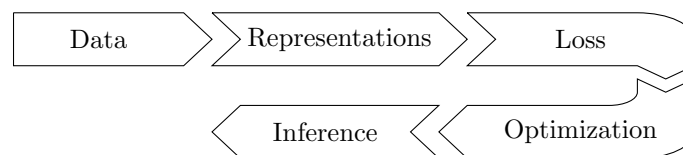


Figure IV.1: Pipeline diagram.

backgrounds during the training process. These backgrounds are created by a generative network that is trained as an adversary to the model. The proposed method improves performance by over 20% for a case study using overhead power line insulators detection.

*Active Supervision* (2019) is an attempt at avoiding the expensive task of performing active learning, where new observations are collected from the real-world to improve the model’s performance. The method proposed uses the model to extrapolate the existing dataset and uses the user to provide labels during training, helping improve the model’s performance without the expense of performing extra data acquisition.

*Iterative Inference* (2018) provides the first known way to make neural network inference be iterative rather than as a single-pass. This can have several applications, for example, in terms of improving existing predictions or making inference more intuitive. Furthermore, the performance of the proposed methodology seems to achieve competitive results relative to state-of-the-art segmentation methods, and even improve on their segmentations.

## IV.2 Production

The first work on class imbalance was published before joining the Ph.D. after becoming a researcher at INESC TEC in November 2015. The rest was published while doing the Ph.D. (2016–2020). All in all, 11 papers were published during the Ph.D., two of these in journals and the rest in conferences, three of these in top conferences (IEEE IJCNN and ICPR) and the impact factor of the journals is 1.5 and 2.6. These publications are divided by chapter and rank in Table IV.1. As a proxy to the quality of the publication venues, CORE2020 was used<sup>1</sup>, except for Ibpria that was not listed, for which ERA2010 listed as Rank C. Chapter 4 earned the best oral presentation award in RECPAD 2020; we did not include here papers submitted to RECPAD since it a national Portuguese conference whose proceedings are not indexed.

Along the way, we left a large graveyard of unfinished or impractical work. For reproducibility purposes, most work performed during the Ph.D. may be found at my personal github repository: <https://github.com/rpmcruz>.

**Table IV.1:** Paper production during the thesis, split by chapter.

	Chapter	1	2	3	4	5	6	$\Sigma$
CORE2020 <sup>1</sup>	Rank A	0	1	0	0	0	1	2
	Rank B	1	0	0	2	0	0	3
	Rank C	0	2	1	0	0	0	3
	National	0	2	1	0	0	0	3
	Unranked	0	1	0	0	0	0	1
	Awards	0	0	0	1	0	0	1

## IV.3 Limitations

A well-known limitation of deep learning experiments is that there is a tendency to be overly optimistic of one’s own proposals since more time is spent refining them. Yet, we have tried to validate them using several varied datasets, and in some cases, such as in class imbalance, perform statistical tests. For work such as background invariance (Chapter 4), while fewer datasets were used, many synthetic backgrounds are created and the difference

<sup>1</sup><https://www.core.edu.au/conference-portal>



**Table IV.2:** Summary of Applicability Limitations

Chapter	Only images	Only deep learning	Task limitations
1	Yes	Yes	None
2	No	No	Classification and segmentation
3	No	Yes	Binary classification
4	Yes	Yes	None
5	Yes	Yes	Classification
6	No	Yes	Multi-output tasks, such as segmentation

is so striking as to make it unlikely to be due to chance. Furthermore, experiments are repeated several times.

The thesis has focused on deep learning as the vehicle and images as the passengers. The proposals are in some cases limited to these use-cases, but not always. Table IV.2 summarizes these limitations. In some cases, such as active supervision (Chapter 5), while it might be possible to extend it beyond that realm, it would require significant changes.

Furthermore, the proposals have been evaluated in either classification or segmentation tasks, and are sometimes limited to one or the other task. For example, it would not be trivial to apply ranking for class imbalance (Chapter 2) to segmentation or iterative inference (Chapter 6) to classification.

## IV.4 Future Work

We feel several chapters in the thesis could potentially sprout new lines of research. Augmentation during training (Chapter 1) could potentially be improved by resorting to the multi-armed bandit literature, which are heuristics for dealing with the exploration-exploitation trade-off that do not require much data. Usually, multi-armed bandits are formulated for categorical decision-making, but variants exist, such as  $\mathcal{X}$ -Armed Bandits [92] which can be used for the numerical search space. In our work, fixed step increments were used, and we did also experimented with  $\mathcal{X}$ -Armed Bandits which could potentially be used to decide the step increment. One problem is that our problem is non-stationary (the loss naturally becomes smaller and smaller, on average) which would require deeper exploration of the literature.

The optimization/training proposals in background invariance (Chapter 4) and active supervision (Chapter 5) made use of the latest generative techniques from the literature and have potential to be improved, especially as the generative literature itself develops.

Iterative inference (Chapter 6) has been so far used only for image segmentation, but could potentially be used in other contexts such as NLP for text generation. In fact, it could apply to any problem where several decisions must be made but where it is harder to measure performance using more than one metric. For example, some interesting experiments were performed where a recurrent neural network evaluated a cardiac signal and iterative inference was used to classify its subsequent parts.

It is our hope that the work presented here endures and serves as inspiration for future research and new methods.



## Appendices

### A.1 Data Augmentation through Hill Climbing

**Table A.1:** Classification scores in percentage using the testing set (higher is better)

Classification (balanced accuracy)					
Dataset	None	Random	Bayesian	Proposal	Proposal*
CIFAR-10	64.1	70.0	73.6	<b>78.2</b>	<i>73.7</i>
CIFAR-100	27.4	27.3	<i>30.4</i>	<b>38.5</b>	18.5
Fashion-MNIST	91.4	83.7	<i>92.0</i>	<i>92.0</i>	<b>92.1</b>
ISIC 2017	48.7	46.7	<b>53.7</b>	<i>49.8</i>	43.0
MNIST	99.1	99.0	<b>99.5</b>	<i>99.4</i>	99.2
PH2	46.7	<i>62.8</i>	<b>64.7</b>	48.6	52.1
smartskins	32.7	34.3	35.9	<b>37.9</b>	<i>37.1</i>
STL10	57.2	<i>62.4</i>	58.5	<b>64.8</b>	56.8
SVHN	79.4	10.0	81.8	<b>87.1</b>	<i>85.6</i>
VOC 2012	12.7	28.9	<b>34.9</b>	<i>33.5</i>	30.8
# Top 1	<b>0</b>	<b>0</b>	<b>4</b>	<b>5</b>	<b>1</b>
# Top 2	<i>0</i>	<i>2</i>	<i>6</i>	<i>9</i>	<i>4</i>
Avg Rel Gain	0.0	8.5	26.6	27.2	14.6

**bold:** best method; *italic:* second-best.

**Table A.2:** Segmentation scores in percentage using the testing set (higher is better)

Semantic Segmentation (Jaccard index)					
Dataset	None	Random	Bayesian	Proposal	Proposal*
breast-aesthetic	95.6	90.6	<i>96.8</i>	95.9	<b>96.9</b>
cervix-huc	82.4	<i>84.5</i>	84.2	83.7	<b>84.9</b>
cervix-kaggle	92.6	91.1	<i>93.2</i>	<b>93.4</b>	<i>93.2</i>
iris	<i>99.1</i>	98.4	98.2	<b>99.2</b>	98.9
ISIC 2017	90.7	<i>91.5</i>	<b>91.7</b>	<b>91.7</b>	91.1
PH2	87.5	<b>91.3</b>	89.9	<i>90.7</i>	89.7
smartskins	<b>97.9</b>	94.6	96.7	96.2	<i>97.1</i>
teeth	<i>94.0</i>	91.5	91.0	<b>94.1</b>	93.0
vessels	67.7	75.4	<b>77.8</b>	<i>76.8</i>	73.6
VOC 2012	74.0	75.1	<b>78.7</b>	<i>77.8</i>	76.2
# Top 1	<b>1</b>	<b>1</b>	<b>3</b>	<b>4</b>	<b>2</b>
# Top 2	<i>3</i>	<i>3</i>	<i>5</i>	<i>7</i>	<i>4</i>
Avg Rel Gain	0.0	0.7	2.4	2.5	1.8

**bold:** best method; *italic:* second-best.

## A.2 Class Imbalance using Pairwise Learning

### A.2.1 Binary Case

Table A.3: Pairwise ranking for binary imbalance.

Dataset	Linear SVM family											
	F <sub>1</sub> score (higher is better)						ROC AUC (higher is better)					
	Ranking	Baseline	Weights	SMOTE	MSMOTE	MetaCost	Ranking	Baseline	Weights	SMOTE	MSMOTE	MetaCost
sonar	<b>0.741</b>	0.723	<b>0.734</b>	0.723	0.724	0.725	<b>0.832</b>	0.823	<b>0.826</b>	0.823	0.823	0.816
breast-cancer-wisconsin	<b>0.957</b>	0.953	0.954	<b>0.955</b>	0.954	0.953	0.994	<b>0.994</b>	0.994	0.994	<b>0.994</b>	<b>0.994</b>
german	<b>0.618</b>	0.568	<b>0.622</b>	0.616	<b>0.617</b>	0.598	<b>0.808</b>	<b>0.809</b>	<b>0.809</b>	0.806	0.804	<b>0.808</b>
haberman	<b>0.485</b>	0.188	<b>0.476</b>	<b>0.484</b>	<b>0.469</b>	0.253	<b>0.685</b>	<b>0.689</b>	<b>0.690</b>	0.677	0.668	<b>0.688</b>
transfusion	0.516	0.154	<b>0.528</b>	0.518	0.522	0.174	<b>0.758</b>	<b>0.758</b>	<b>0.757</b>	0.753	0.752	<b>0.758</b>
vehicle-van	<b>0.937</b>	<b>0.940</b>	0.930	0.932	0.934	0.927	<b>0.995</b>	0.995	0.995	0.995	0.994	0.994
CTG	0.925	0.931	0.915	0.917	0.919	<b>0.934</b>	<b>0.993</b>	<b>0.993</b>	<b>0.993</b>	0.993	0.992	0.993
hepatitis	<b>0.634</b>	<b>0.606</b>	<b>0.651</b>	<b>0.630</b>	<b>0.632</b>	<b>0.648</b>	<b>0.882</b>	<b>0.877</b>	<b>0.884</b>	<b>0.882</b>	<b>0.871</b>	<b>0.881</b>
segment-1	0.986	<b>0.991</b>	0.988	<b>0.990</b>	<b>0.990</b>	0.990	0.996	<b>0.998</b>	0.997	<b>0.998</b>	<b>0.998</b>	<b>0.999</b>
winequality-red-7,8	<b>0.517</b>	0.228	0.479	0.482	0.491	0.346	<b>0.858</b>	<b>0.857</b>	<b>0.858</b>	0.856	0.850	0.855
vowel-1	<b>0.457</b>	0.180	<b>0.445</b>	0.442	0.421	0.273	<b>0.892</b>	0.884	<b>0.893</b>	0.887	0.850	0.870
abalone-9vs18	<b>0.652</b>	0.502	0.473	0.493	0.500	<b>0.632</b>	0.948	<b>0.952</b>	<b>0.950</b>	0.948	0.926	<b>0.950</b>
glass-6	<b>0.695</b>	0.000	0.234	0.236	0.226	0.010	<b>0.984</b>	0.507	0.763	0.752	0.761	0.436
car-good	<b>0.476</b>	0.064	0.422	0.438	0.391	0.274	<b>0.959</b>	0.958	<b>0.959</b>	0.958	0.941	0.955
yeast-ME1	<b>0.612</b>	0.523	0.556	0.562	0.564	0.571	<b>0.986</b>	<b>0.986</b>	<b>0.986</b>	0.985	0.984	0.986
Average	0.681	0.503	0.627	0.628	0.624	0.554	0.905	0.872	0.890	0.887	0.881	0.865
Winner	80%	20%	40%	26%	26%	20%	80%	66%	73%	13%	20%	46%

Dataset	Multilayer perceptron family											
	F <sub>1</sub> score (higher is better)						ROC AUC (higher is better)					
	Ranking	Baseline	Weights	SMOTE	MSMOTE	MetaCost	Ranking	Baseline	Weights	SMOTE	MSMOTE	MetaCost
sonar	<b>0.805</b>	<b>0.804</b>	<b>0.801</b>	0.731	0.732	0.725	<b>0.881</b>	<b>0.896</b>	<b>0.895</b>	0.830	0.829	0.817
breast-cancer	0.946	0.942	0.947	<b>0.955</b>	<b>0.954</b>	<b>0.955</b>	0.975	0.989	0.991	0.994	0.993	<b>0.994</b>
german	0.513	0.540	0.543	<b>0.618</b>	0.609	<b>0.623</b>	0.665	0.744	0.721	<b>0.808</b>	0.807	<b>0.809</b>
haberman	0.444	0.361	0.459	0.435	0.431	<b>0.497</b>	0.604	<b>0.675</b>	<b>0.678</b>	0.676	0.663	<b>0.688</b>
transfusion	0.495	0.391	0.506	0.492	0.502	<b>0.525</b>	0.549	<b>0.764</b>	0.753	0.757	0.755	0.758
vehicle-van	<b>0.944</b>	<b>0.940</b>	<b>0.941</b>	0.593	0.596	0.596	<b>0.982</b>	<b>0.996</b>	<b>0.996</b>	0.985	0.985	0.923
CTG	0.961	<b>0.965</b>	<b>0.963</b>	0.919	0.919	0.925	0.993	0.997	<b>0.998</b>	0.993	0.993	0.993
hepatitis	<b>0.600</b>	0.502	0.528	0.520	0.515	<b>0.611</b>	<b>0.828</b>	0.803	0.801	0.795	0.791	<b>0.846</b>
segment-1	<b>0.989</b>	<b>0.990</b>	<b>0.975</b>	0.984	0.984	0.986	<b>0.999</b>	<b>0.999</b>	<b>0.999</b>	0.996	0.996	0.996
winequality-red-7,8	0.477	0.482	<b>0.508</b>	0.316	0.317	0.269	0.555	0.823	<b>0.843</b>	0.790	0.786	0.655
vowel-1	0.397	<b>0.946</b>	0.855	0.480	0.482	0.379	0.516	<b>0.989</b>	0.973	0.963	0.954	0.899
abalone-9vs18	<b>0.511</b>	0.485	0.362	0.301	0.347	0.358	0.801	<b>0.917</b>	0.907	<b>0.927</b>	<b>0.922</b>	0.889
glass-6	0.024	0.000	0.136	<b>0.350</b>	<b>0.347</b>	0.290	0.442	0.338	0.556	<b>0.683</b>	<b>0.698</b>	0.610
car-good	<b>0.839</b>	<b>0.849</b>	0.737	0.447	0.402	0.392	0.959	<b>0.996</b>	0.982	0.966	0.951	0.953
yeast-ME1	<b>0.653</b>	0.564	0.528	0.603	0.583	0.597	0.950	<b>0.986</b>	0.983	<b>0.986</b>	0.984	<b>0.986</b>
Average	0.640	0.651	0.653	0.583	0.581	0.582	0.780	0.861	0.872	0.876	0.874	0.854
Winner	46%	40%	33%	20%	13%	33%	26%	60%	40%	26%	13%	33%

Dataset	Adaboost family											
	F <sub>1</sub> score						ROC AUC					
	Ranking	Baseline	Weights	SMOTE	MSMOTE	MetaCost	Ranking	Baseline	Weights	SMOTE	MSMOTE	MetaCost
sonar	0.787	<b>0.824</b>	<b>0.824</b>	<b>0.824</b>	<b>0.824</b>	0.818	0.891	<b>0.917</b>	<b>0.917</b>	<b>0.917</b>	<b>0.917</b>	<b>0.916</b>
breast-cancer	0.937	0.932	0.932	0.937	0.934	<b>0.948</b>	<b>0.990</b>	0.989	0.989	0.990	0.989	<b>0.991</b>
german	<b>0.597</b>	0.541	0.541	<b>0.588</b>	0.583	<b>0.587</b>	0.783	<b>0.794</b>	<b>0.794</b>	<b>0.792</b>	<b>0.793</b>	<b>0.796</b>
haberman	0.419	0.375	0.375	<b>0.464</b>	<b>0.458</b>	0.441	0.638	0.663	0.663	0.671	0.673	<b>0.692</b>
transfusion	<b>0.502</b>	0.418	0.418	<b>0.511</b>	<b>0.509</b>	0.493	0.718	<b>0.745</b>	<b>0.745</b>	0.737	0.737	<b>0.740</b>
vehicle-van	<b>0.928</b>	0.901	0.901	0.905	0.909	0.906	<b>0.993</b>	0.991	0.991	0.991	0.991	0.989
CTG	0.973	0.972	0.972	0.970	0.971	<b>0.979</b>	0.996	<b>0.997</b>	<b>0.997</b>	0.996	<b>0.997</b>	0.996
hepatitis	<b>0.578</b>	0.525	0.525	<b>0.581</b>	<b>0.596</b>	<b>0.596</b>	0.822	0.808	0.808	<b>0.833</b>	<b>0.842</b>	<b>0.846</b>
segment-1	<b>0.993</b>	0.990	0.990	0.987	0.988	0.988	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	1.000
winequality-red-7,8	<b>0.520</b>	0.431	0.431	0.509	0.511	<b>0.528</b>	<b>0.867</b>	<b>0.868</b>	<b>0.868</b>	0.864	0.863	<b>0.869</b>
vowel-1	<b>0.692</b>	0.443	0.443	0.610	0.549	0.633	<b>0.953</b>	0.946	0.946	<b>0.948</b>	0.930	0.939
abalone-9vs18	<b>0.369</b>	0.318	0.318	0.287	0.298	<b>0.377</b>	0.803	<b>0.820</b>	<b>0.820</b>	0.793	0.791	<b>0.822</b>
glass-6	<b>0.801</b>	0.670	0.670	<b>0.825</b>	<b>0.777</b>	0.713	<b>0.996</b>	<b>0.998</b>	<b>0.998</b>	<b>0.993</b>	0.989	0.982
car-good	<b>0.573</b>	0.388	0.388	<b>0.596</b>	0.515	0.401	0.974	<b>0.977</b>	<b>0.977</b>	0.976	0.965	0.919
yeast-ME1	<b>0.667</b>	<b>0.671</b>	<b>0.671</b>	0.654	0.635	<b>0.698</b>	0.982	<b>0.986</b>	<b>0.986</b>	<b>0.985</b>	0.982	<b>0.986</b>
Average	0.689	0.627	0.627	0.683	0.671	0.674	0.894	0.900	0.900	0.899	0.897	0.899
Winner	73%	13%	13%	46%	33%	46%	40%	66%	66%	46%	33%	60%

## A.2.2 Ordinal Case

**Table A.4:** Combining pairwise ranking with traditional methods for ordinal imbalance (linear SVM).

	Linear SVM									
	MAE score (lower is better)									
Dataset	WRank	BRank	SRank	MSRank	OvR	OvR/w	SVOREX	SVORIM	oSVM	
balance-scale	0.12	0.52	0.11	1.00	0.20	0.19	<b>0.11</b>	<b>0.11</b>	0.12	
car	0.09	0.11	0.09	1.07	0.12	0.09	0.14	0.12	<b>0.08</b>	
contact-lenses	0.42	0.48	<b>0.38</b>	<b>0.39</b>	<b>0.42</b>	<b>0.44</b>	0.51	0.54	<b>0.42</b>	
cooling	<b>0.41</b>	1.05	1.13	1.28	0.44	0.55	0.48	0.50	0.49	
diabetes5	<b>0.64</b>	<b>0.67</b>	0.74	0.77	0.72	0.95	0.84	<b>0.67</b>	0.85	
diabetes10	<b>1.68</b>	<b>1.72</b>	1.77	1.77	2.06	2.15	1.81	<b>1.69</b>	2.41	
newthyroid	0.04	0.18	0.05	1.00	0.04	<b>0.03</b>	0.04	0.04	<b>0.03</b>	
pyrim5	<b>0.58</b>	0.99	1.16	1.16	<b>0.58</b>	0.70	1.08	0.99	0.65	
pyrim10	1.34	<b>1.26</b>	<b>1.29</b>	1.33	1.52	1.49	2.89	<b>1.32</b>	1.50	
squash-stored	0.46	0.86	1.11	1.13	0.47	0.47	<b>0.41</b>	0.44	<b>0.38</b>	
squash-unstored	<b>0.27</b>	<b>0.27</b>	<b>0.27</b>	<b>0.26</b>	0.30	<b>0.30</b>	<b>0.26</b>	0.26	0.33	
stock10	0.64	0.66	0.67	1.02	<b>0.42</b>	0.42	0.68	0.63	0.70	
toy	<b>0.84</b>	0.93	<b>0.87</b>	1.21	1.02	1.01	1.13	0.95	0.96	
triazines5	0.70	1.31	0.98	1.08	0.69	<b>0.67</b>	<b>0.67</b>	<b>0.67</b>	<b>0.70</b>	
triazines10	1.40	1.95	2.01	1.97	<b>1.33</b>	1.51	<b>1.37</b>	1.39	1.45	
Average	0.64	0.86	0.84	1.10	0.69	0.73	0.83	0.69	0.74	
Deviation	0.48	0.52	0.58	0.42	0.55	0.58	0.73	0.48	0.62	
Winner	40%	26%	26%	13%	26%	26%	33%	33%	33%	
	MMAE score (lower is better)									
Dataset	WRank	BRank	SRank	MSRank	OvR	OvR/w	SVOREX	SVORIM	oSVM	
balance-scale	0.21	1.01	0.15	1.10	1.00	0.96	<b>0.17</b>	<b>0.14</b>	0.21	
car	0.47	1.06	<b>0.28</b>	1.15	0.77	<b>0.29</b>	1.36	1.01	0.49	
contact-lenses	<b>0.81</b>	1.20	<b>0.78</b>	<b>0.76</b>	<b>0.88</b>	<b>0.73</b>	0.97	1.04	<b>0.82</b>	
cooling	2.32	<b>1.72</b>	<b>1.73</b>	1.81	2.24	<b>1.80</b>	2.98	2.88	2.07	
diabetes5	<b>1.15</b>	<b>1.17</b>	<b>1.20</b>	<b>1.23</b>	1.48	1.52	1.51	1.30	1.43	
diabetes10	<b>3.09</b>	<b>3.16</b>	<b>3.12</b>	3.26	3.82	3.98	3.47	<b>2.94</b>	4.33	
newthyroid	0.14	1.00	<b>0.09</b>	1.04	0.16	0.13	0.14	0.14	<b>0.13</b>	
pyrim5	<b>1.40</b>	1.83	2.26	2.31	<b>1.30</b>	1.62	3.00	2.00	1.87	
pyrim10	<b>3.80</b>	<b>3.91</b>	<b>3.84</b>	<b>3.75</b>	<b>3.86</b>	<b>3.84</b>	6.37	4.33	4.18	
squash-stored	0.83	1.23	1.42	1.46	1.06	0.94	0.76	0.83	<b>0.66</b>	
squash-unstored	0.57	1.00	0.55	0.52	0.76	0.80	<b>0.46</b>	0.46	<b>0.57</b>	
stock10	1.02	1.04	0.97	1.64	1.03	<b>0.85</b>	1.30	1.05	1.29	
toy	<b>1.79</b>	2.17	<b>1.67</b>	2.49	1.92	<b>1.57</b>	3.00	2.00	1.82	
triazines5	2.77	2.11	<b>1.73</b>	2.52	2.99	2.94	3.00	3.00	2.79	
triazines10	6.14	<b>4.46</b>	<b>4.58</b>	<b>4.55</b>	6.58	6.35	7.00	6.83	6.80	
Average	1.77	1.87	1.63	1.97	1.99	1.89	2.37	2.00	1.96	
Deviation	1.58	1.08	1.30	1.12	1.63	1.64	2.02	1.74	1.81	
Winner	40%	33%	66%	26%	20%	40%	13%	13%	26%	

**Table A.5:** Combining pairwise ranking with traditional methods for ordinal imbalance (RBF-kernel SVM).

	RBF-kernel SVM							
	MAE score (lower is better)							
Dataset	WRank	BRank	SRank	MSRank	OvR	OvR/w	SVOREX	SVORIM
balance-scale	0.11	0.92	0.14	0.15	0.14	0.18	0.11	<b>0.05</b>
car	0.13	0.19	0.25	0.24	<b>0.12</b>	0.21	0.41	0.41
contact-lenses	0.52	0.51	0.45	0.46	0.47	<b>0.33</b>	1.31	0.98
cooling	0.62	0.61	0.63	0.63	<b>0.54</b>	0.62	0.58	<b>0.55</b>
diabetes5	<b>0.65</b>	<b>0.68</b>	<b>0.64</b>	<b>0.65</b>	<b>0.65</b>	<b>0.67</b>	0.72	<b>0.69</b>
diabetes10	<b>1.35</b>	1.53	1.38	<b>1.37</b>	1.75	1.78	1.62	1.56
newthyroid	0.29	0.31	0.29	0.29	0.25	0.23	<b>0.16</b>	0.16
pyrim5	<b>0.47</b>	0.64	0.56	0.60	1.08	1.10	1.08	0.99
pyrim10	<b>1.02</b>	1.18	<b>1.03</b>	<b>1.06</b>	2.73	2.18	2.88	2.00
squash-stored	<b>0.57</b>	0.57	0.57	0.57	0.73	0.57	0.73	0.57
squash-unstored	0.54	0.54	0.54	0.54	<b>0.44</b>	0.44	0.49	0.50
stock10	1.35	1.38	1.33	1.30	0.18	<b>0.17</b>	0.27	0.26
toy	<b>0.03</b>	0.12	<b>0.03</b>	<b>0.04</b>	0.91	0.66	1.08	0.95
triazines5	<b>0.68</b>	1.10	0.88	0.87	<b>0.67</b>	1.18	0.67	0.67
triazines10	<b>1.28</b>	1.76	1.67	1.64	1.37	2.45	1.37	1.37
Average	0.64	0.80	0.69	0.69	0.80	0.85	0.90	0.78
Deviation	0.42	0.48	0.46	0.45	0.68	0.72	0.69	0.52
Winner	53%	6%	20%	26%	33%	20%	6%	20%

	MMAE score (lower is better)							
Dataset	WRank	BRank	SRank	MSRank	OvR	OvR/w	SVOREX	SVORIM
balance-scale	0.20	1.79	0.19	0.19	1.00	0.24	1.00	<b>0.13</b>
car	1.98	2.00	1.00	1.01	1.13	<b>0.27</b>	3.00	3.00
contact-lenses	1.27	1.97	1.23	1.22	0.88	<b>0.53</b>	1.82	1.32
cooling	1.26	1.22	<b>0.99</b>	<b>0.98</b>	1.77	<b>1.01</b>	2.00	2.00
diabetes5	<b>1.90</b>	<b>1.97</b>	<b>1.97</b>	<b>1.88</b>	2.00	2.00	2.00	2.00
diabetes10	3.57	3.77	3.63	3.57	4.27	4.27	<b>3.06</b>	<b>2.84</b>
newthyroid	1.00	1.03	1.00	1.00	1.00	0.97	<b>0.64</b>	0.64
pyrim5	<b>1.15</b>	1.78	<b>1.14</b>	<b>1.22</b>	3.00	2.90	3.00	2.00
pyrim10	<b>2.65</b>	3.65	2.82	2.82	6.63	5.67	6.30	4.93
squash-stored	<b>1.00</b>	1.00	1.00	1.00	2.00	1.00	2.00	1.00
squash-unstored	<b>1.00</b>	1.00	1.00	1.00	1.00	1.00	1.00	1.00
stock10	3.71	4.84	3.27	3.16	0.66	<b>0.45</b>	1.14	1.11
toy	<b>0.10</b>	1.00	<b>0.10</b>	<b>0.10</b>	1.83	1.15	2.80	2.00
triazines5	2.66	<b>1.93</b>	<b>1.97</b>	<b>1.91</b>	3.00	2.82	3.00	3.00
triazines10	5.83	<b>4.73</b>	<b>4.99</b>	<b>4.69</b>	7.00	5.66	7.00	7.00
Average	1.95	2.25	1.75	1.72	2.48	1.99	2.65	2.26
Deviation	1.47	1.29	1.32	1.25	1.95	1.81	1.76	1.71
Winner	40%	20%	40%	40%	0%	26%	13%	13%

**Table A.6:** Combining pairwise ranking with Frank & Hall for ordinal imbalance.

	← Regularization →					← Regularization →				
	AMAE score (lower is better)									
	SVOR	F&H w/ SVM				RankSVM	F&H w/ RankSVM			
		$\lambda=1$	$\lambda=0.1$	$\lambda=0.01$	$\lambda=0$		$\lambda=1$	$\lambda=0.1$	$\lambda=0.01$	$\lambda=0$
toy	1.20	1.20	1.20	1.18	1.18	1.35	1.10	0.98	0.93	<b>0.91</b>
wisconsin5	<b>1.16</b>	1.21	1.21	1.24	1.24	1.23	1.22	1.25	1.30	1.26
wisconsin10	<b>2.44</b>	2.51	2.51	2.56	2.51	2.59	2.67	2.71	2.75	2.70
diabetes10	1.46	1.48	1.48	1.57	1.66	1.56	<b>1.29</b>	<b>1.29</b>	1.38	1.39
contact-lenses	<b>0.43</b>	<b>0.44</b>	<b>0.44</b>	<b>0.44</b>	<b>0.40</b>	<b>0.40</b>	0.49	0.48	0.48	<b>0.45</b>
ERA	1.49	1.41	1.41	1.51	1.66	1.49	<b>1.27</b>	<b>1.26</b>	<b>1.26</b>	<b>1.26</b>
diabetes5	0.85	0.84	0.84	0.87	0.91	0.69	<b>0.63</b>	<b>0.60</b>	<b>0.60</b>	<b>0.60</b>
auto5	0.43	0.41	0.41	0.42	0.42	0.42	0.39	0.36	<b>0.34</b>	<b>0.34</b>
LEV	0.73	0.65	0.65	0.66	0.66	0.65	0.59	0.59	<b>0.58</b>	<b>0.58</b>
auto10	1.01	0.78	0.78	0.89	0.95	0.80	0.74	0.72	<b>0.67</b>	<b>0.68</b>
SWD	0.61	0.61	0.61	0.61	0.62	0.61	<b>0.53</b>	<b>0.53</b>	0.54	0.54
machine10	1.08	1.01	1.01	1.04	1.43	1.05	<b>0.96</b>	<b>0.94</b>	<b>0.93</b>	<b>0.93</b>
machine5	0.52	<b>0.45</b>	<b>0.45</b>	0.46	0.60	0.49	<b>0.43</b>	<b>0.41</b>	<b>0.41</b>	<b>0.42</b>
ESL12vs3vs456...	0.61	<b>0.29</b>	<b>0.29</b>	0.49	0.62	0.34	0.32	0.31	0.32	0.33
ERA1vs23456vs...	1.06	0.88	0.88	1.14	1.34	0.66	<b>0.62</b>	<b>0.61</b>	<b>0.62</b>	0.63
Average	1.00	0.94	0.94	1.00	1.08	0.96	0.88	0.87	0.87	0.87
Winner	20%	20%	20%	7%	7%	7%	47%	47%	53%	60%

	MMAE score (lower is better)									
	SVOR	F&H w/ SVM				RankSVM	F&H w/ RankSVM			
		$\lambda=1$	$\lambda=0.1$	$\lambda=0.01$	$\lambda=0$		$\lambda=1$	$\lambda=0.1$	$\lambda=0.01$	$\lambda=0$
toy	2.00	2.00	2.00	2.00	2.00	2.02	1.70	1.45	1.40	<b>1.38</b>
wisconsin5	1.88	1.90	1.90	1.91	1.98	<b>1.81</b>	<b>1.74</b>	<b>1.75</b>	1.84	<b>1.80</b>
wisconsin10	4.66	4.71	4.71	4.83	4.87	<b>4.31</b>	<b>4.47</b>	<b>4.54</b>	4.61	<b>4.49</b>
diabetes10	3.12	3.18	3.18	3.40	3.50	3.06	<b>2.58</b>	<b>2.66</b>	2.86	2.80
contact-lenses	<b>0.88</b>	<b>1.01</b>	<b>1.01</b>	<b>0.95</b>	<b>0.89</b>	<b>0.88</b>	<b>1.00</b>	<b>1.00</b>	1.06	<b>1.01</b>
ERA	2.47	2.26	2.26	2.52	2.99	2.25	<b>1.84</b>	<b>1.85</b>	<b>1.83</b>	<b>1.84</b>
diabetes5	1.48	1.50	1.50	1.48	1.52	1.24	<b>1.04</b>	<b>1.03</b>	<b>1.04</b>	<b>1.03</b>
auto5	1.00	1.00	1.00	1.00	1.00	0.83	0.83	<b>0.75</b>	<b>0.73</b>	<b>0.71</b>
LEV	1.37	1.30	1.30	1.29	1.28	<b>1.01</b>	<b>1.03</b>	<b>1.06</b>	<b>1.04</b>	<b>1.05</b>
auto10	2.97	2.05	2.05	2.48	2.67	<b>1.71</b>	<b>1.74</b>	<b>1.74</b>	<b>1.69</b>	<b>1.70</b>
SWD	1.11	1.12	1.12	1.10	1.10	1.00	<b>0.71</b>	0.73	0.77	0.78
machine10	2.97	3.02	3.02	<b>2.69</b>	3.62	3.17	2.98	<b>2.91</b>	<b>2.77</b>	<b>2.72</b>
machine5	1.12	<b>1.05</b>	<b>1.05</b>	1.06	1.35	1.12	<b>0.97</b>	<b>0.96</b>	<b>0.97</b>	<b>1.00</b>
ESL12vs3vs456...	1.07	0.68	0.68	0.98	1.09	0.72	<b>0.58</b>	<b>0.57</b>	<b>0.57</b>	<b>0.59</b>
ERA1vs23456vs...	2.10	1.51	1.51	2.16	2.77	<b>1.10</b>	<b>1.07</b>	<b>1.07</b>	<b>1.11</b>	1.11
Average	2.01	1.89	1.89	1.99	2.17	1.75	1.62	1.60	1.62	1.60
Winner	7%	13%	13%	13%	7%	40%	80%	87%	60%	80%

## A.2.3 Survival Analysis

Table A.7: Survival analysis results across several metrics.

Method	Accuracy	MAE	AMAE	MMAE
Nominal and ordinal classifiers				
LSVM	<b>85.11 ± 0.34</b>	<i>0.304 ± 0.004</i>	1.500 ± 0.000	3.000 ± 0.000
CS-LSVM	85.04 ± 0.47	0.306 ± 0.008	1.501 ± 0.002	3.000 ± 0.000
SVM	<b>85.11 ± 0.34</b>	<i>0.304 ± 0.004</i>	1.500 ± 0.000	3.000 ± 0.000
CS-SVM	<b>85.11 ± 0.34</b>	<b>0.303 ± 0.005</b>	1.500 ± 0.000	3.000 ± 0.000
LSVM+OGO	<b>85.11 ± 0.79</b>	<b>0.303 ± 0.017</b>	1.500 ± 0.000	3.000 ± 0.000
LSVORIM	20.20 ± 31.69	1.435 ± 0.196	<b>1.102 ± 0.196</b>	<b>2.185 ± 0.393</b>
LSVORIM+OGO	54.82 ± 31.97	0.994 ± 0.822	1.420 ± 0.179	2.566 ± 0.402
POM	<i>84.97 ± 0.34</i>	0.304 ± 0.006	1.500 ± 0.001	3.000 ± 0.000
POM+OGO	63.33 ± 3.12	0.540 ± 0.049	1.410 ± 0.086	2.562 ± 0.179
Rank-based learners				
Rank	64.37 ± 19.29	0.804 ± 0.439	1.422 ± 0.081	<i>2.416 ± 0.541</i>
F2C-Rank	67.75 ± 12.85	0.669 ± 0.281	<i>1.407 ± 0.096</i>	2.452 ± 0.511
Rank+OGO	69.74 ± 20.14	0.600 ± 0.399	1.443 ± 0.109	2.624 ± 0.497
F2C-Rank+OR	84.69 ± 0.45	0.310 ± 0.012	1.493 ± 0.021	2.975 ± 0.079
Comparison between threshold optimisation strategies for ranking				
Rank-Inv	64.37 ± 19.29	0.804 ± 0.439	1.422 ± 0.081	2.416 ± 0.541
Rank-Unif	84.55 ± 1.05	0.314 ± 0.018	1.495 ± 0.022	3.000 ± 0.000
Rank-Abs	85.04 ± 0.36	0.306 ± 0.010	1.489 ± 0.026	3.000 ± 0.000
F2C-Rank-Inv	67.75 ± 12.85	0.669 ± 0.281	1.407 ± 0.096	2.452 ± 0.511
F2C-Rank-Unif	<i>84.97 ± 0.32</i>	0.306 ± 0.009	1.495 ± 0.020	3.000 ± 0.000
F2C-Rank-Abs	84.83 ± 0.26	0.310 ± 0.010	1.499 ± 0.011	3.000 ± 0.000

**bold:** best method; *italic:* second-best.



### A.3 Risk Aversion

**Table A.8:** Risk aversion results for Kernel Density Estimation for  $\rho = 0.05$  (testing set).

	Multivariate <sup>+</sup>		Multivariate <sup>-</sup>		Cascade	
	FN	TN	FN	TN	FN	TN
breast-cancer-wisconsin	<b>0.98</b>	1.00	<b>0.06</b>	0.97	0.03	0.88
car-good	<b>1.00</b>	1.00	<b>0.19</b>	0.27	0.00	0.93
german	<b>1.00</b>	1.00	<b>0.06</b>	0.07	<b>0.06</b>	0.07
haberman	<b>0.36</b>	0.37	0.00	0.00	0.01	0.00
heart	<b>1.00</b>	1.00	<b>0.06</b>	0.36	<b>0.07</b>	0.08
sonar	<b>1.00</b>	1.00	0.05	0.04	<b>0.15</b>	0.32
transfusion	—	—	<b>0.07</b>	0.03	0.07	0.15
vehicle-van	<b>1.00</b>	1.00	<b>0.06</b>	0.66	0.02	0.67
vowel-1	<b>1.00</b>	1.00	0.00	0.00	<b>0.10</b>	0.60
winequality-red-7,8	<b>0.78</b>	0.96	0.04	0.08	<b>0.10</b>	0.30
#FNR $\leq \rho$		0		4	<b>5</b>	
#Highest TN		<b>9</b>		0	1	

**Table A.9:** Risk aversion results adapting the loss across several values of  $\rho$  (testing set).

	$\rho = 0.05$							
	Threshold		Term		Dynamic		Fine-tune	
Dataset	FN	TN	FN	TN	FN	TN	FN	TN
breast-aesthetic	0.04	<b>0.71</b>	0.01	0.63	0.01	0.61	0.03	<b>0.71</b>
cervixhuc	<b>0.06</b>	0.52	0.00	0.04	0.03	0.28	0.04	<b>0.47</b>
cervixkaggle	<b>0.06</b>	0.83	0.02	0.52	0.04	<b>0.63</b>	0.01	0.58
iris	0.05	<b>1.00</b>	0.05	0.98	0.01	0.99	0.00	0.98
ISBI2017	0.04	<b>0.64</b>	0.05	0.58	<b>0.07</b>	0.75	0.03	0.56
PH2	<b>0.08</b>	0.72	0.01	<b>0.41</b>	<b>0.14</b>	0.71	<b>0.07</b>	0.78
smartskins	0.05	<b>0.98</b>	<b>0.06</b>	0.61	0.02	0.51	0.00	0.71
teeth	0.05	<b>0.85</b>	0.00	0.13	0.02	0.18	0.03	0.79
vessels	<b>0.10</b>	0.45	0.01	0.34	0.01	0.34	0.02	<b>0.40</b>
#FN $\leq \rho$	5		8		7		8	
#Highest TN	5		1		1		3	
	$\rho = 0.01$							
	Threshold		Term		Dynamic		Fine-tune	
Dataset	FN	TN	FN	TN	FN	TN	FN	TN
breast-aesthetic	0.01	0.63	0.00	0.61	0.01	<b>0.64</b>	0.01	<b>0.64</b>
cervixhuc	<b>0.02</b>	0.27	0.00	0.12	0.01	0.16	0.01	<b>0.21</b>
cervixkaggle	0.01	<b>0.66</b>	<b>0.02</b>	0.57	0.01	0.54	0.01	0.56
iris	0.01	<b>0.99</b>	0.00	0.95	0.00	0.00	0.00	<b>0.99</b>
ISBI2017	0.01	<b>0.34</b>	<b>0.02</b>	0.42	<b>0.02</b>	0.49	0.00	0.26
PH2	0.01	0.30	0.01	<b>0.36</b>	<b>0.02</b>	0.49	<b>0.03</b>	0.64
smartskins	0.01	<b>0.93</b>	<b>0.02</b>	0.75	0.00	0.66	0.00	0.87
teeth	0.01	<b>0.65</b>	<b>0.04</b>	0.37	0.01	0.10	0.01	0.47
vessels	0.01	0.34	0.00	0.33	0.01	0.34	0.01	<b>0.37</b>
#FN $\leq \rho$	8		5		7		8	
#Highest TN	5		1		1		4	
	$\rho = 0.005$							
	Threshold		Term		Dynamic		Fine-tune	
Dataset	FN	TN	FN	TN	FN	TN	FN	TN
breast-aesthetic	0.005	<b>0.60</b>	<b>0.007</b>	0.68	<b>0.006</b>	0.67	<b>0.007</b>	0.67
cervixhuc	<b>0.010</b>	0.21	0.000	0.03	0.002	0.10	0.004	<b>0.15</b>
cervixkaggle	0.005	<b>0.58</b>	<b>0.020</b>	0.58	0.002	0.27	0.005	0.50
iris	<b>0.006</b>	0.99	<b>0.006</b>	0.97	0.003	0.96	0.004	<b>0.99</b>
ISBI2017	<b>0.006</b>	0.27	<b>0.024</b>	0.44	<b>0.010</b>	0.38	0.003	<b>0.21</b>
PH2	0.004	0.17	0.004	<b>0.33</b>	<b>0.015</b>	0.28	<b>0.015</b>	0.46
smartskins	0.004	<b>0.91</b>	0.002	0.90	0.000	0.05	0.000	0.30
teeth	0.005	<b>0.58</b>	<b>0.085</b>	0.57	<b>0.006</b>	0.07	0.000	0.12
vessels	0.003	0.33	0.001	0.33	0.003	<b>0.34</b>	<b>0.007</b>	0.34
#FN $\leq \rho$	6		4		5		6	
#Highest TN	4		1		1		3	

## A.6 Iterative Inference

**Table A.10:** Contrasting models with and without iterative forecasting, in terms of Dice coefficient (higher is better).

Dataset	U-Net		Dilated-Net	
	Original	Iterative	Original	Iterative
SmartSkins	76.62	79.45	76.35	<b>83.36</b>
PH2	83.70	84.09	85.52	<b>86.41</b>
ISBI 2017	71.35	<b>76.52</b>	72.06	76.11
Teeth-UCV	85.85	85.91	86.03	<b>86.14</b>
Breast Aesthetics	93.08	93.31	94.03	<b>94.15</b>
Cervix-HUC	77.25	<b>77.26</b>	75.37	75.37
Cervix-MobileODT	88.24	<b>88.25</b>	86.38	<b>88.25</b>
Mobbio	67.91	68.23	69.90	<b>70.11</b>

**bold:** best method.



# Bibliography

---

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.
- [2] F. Rosenblatt, “The perceptron: a perceiving and recognizing automaton,” Tech. Rep. 85-460-1, Cornell Aeronautical Laboratory, 1957.
- [3] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [4] D. Williams and G. Hinton, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–538, 1986.
- [5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [6] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.
- [7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [8] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [10] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.
- [11] D. Dua and C. Graff, “UCI Machine Learning Repository,” 2017.
- [12] O. L. Mangasarian, W. N. Street, and W. H. Wolberg, “Breast cancer diagnosis and prognosis via linear programming,” *Operations Research*, vol. 43, no. 4, pp. 570–577, 1995.
- [13] Turing Institute, Glasgow, Scotland, “Statlog (vehicle silhouettes) data set.”
- [14] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties,” *Decision Support Systems*, vol. 47, pp. 547–553, nov 2009.

- [15] H. He and E. A. Garcia, “Learning from imbalanced data sets,” *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2010.
- [16] R. Prati, G. Batista, and M. Monard, “Class imbalances versus class overlapping: an analysis of a learning system behavior,” *MICAI 2004: Advances in Artificial Intelligence*, pp. 312–321, 2004.
- [17] M. Denil and T. Trappenberg, “Overlap versus imbalance,” in *Canadian Conference on Artificial Intelligence*, pp. 220–231, Springer, 2010.
- [18] W. Chu and Z. Ghahramani, “Gaussian processes for ordinal regression,” *Journal of Machine Learning Research*, vol. 6, no. Jul, pp. 1019–1041, 2005.
- [19] M. Pérez-Ortiz, P. A. Gutiérrez, C. Hervás-Martínez, and X. Yao, “Graph-based approaches for over-sampling in the context of ordinal regression,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1233–1245, 2015.
- [20] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” tech. rep., Citeseer, 2009.
- [21] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” *CoRR*, vol. abs/1708.07747, 2017.
- [22] N. C. Codella, D. Gutman, M. E. Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kalloo, K. Liopyris, N. Mishra, H. Kittler, *et al.*, “Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI),” in *IEEE 15th International Symposium on Biomedical Imaging (ISBI)*, pp. 168–172, IEEE, 2018.
- [23] Y. LeCun, C. Cortes, and C. Burges, “MNIST handwritten digit database,” *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, vol. 2, 2010.
- [24] T. Mendonça, P. M. Ferreira, J. S. Marques, A. R. Marcal, and J. Rozeira, “PH<sup>2</sup> – a dermoscopic image database for research and benchmarking,” in *35th International Conference on Engineering in Medicine and Biology Society (EMBC)*, pp. 5437–5440, IEEE, 2013.
- [25] M. J. M. Vasconcelos, L. Rosado, and M. Ferreira, “Principal axes-based asymmetry assessment methodology for skin lesion image analysis,” in *International Symposium on Visual Computing*, pp. 21–31, Springer, 2014.
- [26] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.
- [27] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- [28] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes challenge: A retrospective,” *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

- [29] J. S. Cardoso and M. J. Cardoso, “Towards an intelligent medical system for the aesthetic evaluation of breast cancer conservative treatment,” *Artificial Intelligence in Medicine*, vol. 40, no. 2, pp. 115–126, 2007.
- [30] K. Fernandes, J. S. Cardoso, and J. Fernandes, “Transfer learning with partial observability applied to cervical cancer screening,” in *Iberian Conference on Pattern Recognition and Image Analysis (Ibpria)*, pp. 243–250, Springer, 2017.
- [31] Kaggle, “Intel & MobileODT cervical cancer screening competition.” <https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening>, 2017.
- [32] A. F. Sequeira, J. C. Monteiro, A. Rebelo, and H. P. Oliveira, “MobBIO: a multimodal database captured with a portable handheld device,” in *Computer Vision Theory and Applications (VISAPP)*, vol. 3, pp. 133–139, IEEE, 2014.
- [33] K. U. London, “Retinal chase db1 database.” <https://blogs.kingston.ac.uk/retinal/chasedb1/>, 2017. [Accessed 2018/12/01].
- [34] J. Staal, M. D. Abràmoff, M. Niemeijer, M. A. Viergever, and B. Van Ginneken, “Ridge-based vessel segmentation in color images of the retina,” *IEEE Transactions on Medical Imaging*, vol. 23, no. 4, pp. 501–509, 2004.
- [35] A. Hoover, V. Kouznetsova, and M. Goldbaum, “Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response,” *IEEE Transactions on Medical imaging*, vol. 19, no. 3, pp. 203–210, 2000.
- [36] K. Fernandez and C. Chang, “Teeth/palate and interdental segmentation using artificial neural networks,” in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 175–185, Springer, 2012.
- [37] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, ACM, 2009.
- [38] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
- [39] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2546–2554, 2011.
- [40] K. Jamieson and A. Talwalkar, “Non-stochastic best arm identification and hyperparameter optimization,” in *Artificial Intelligence and Statistics*, pp. 240–248, 2016.
- [41] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 113–123, 2019.
- [42] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, J. H. Moore, *et al.*, “Automating biomedical data science through tree-based pipeline optimization,” in *European Conference on the Applications of Evolutionary Computation*, pp. 123–137, Springer, 2016.

- [43] J. Larsen, L. K. Hansen, C. Svarer, and M. Ohlsson, “Design and regularization of neural networks: the optimal use of a validation set,” in *Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pp. 62–71, IEEE, 1996.
- [44] M. Cruz-Ramírez, C. Hervás-Martínez, J. Sánchez-Monedero, and P. A. Gutiérrez, “Metrics to guide a multi-objective evolutionary algorithm for ordinal classification,” *Neurocomputing*, vol. 135, pp. 21–31, 2014.
- [45] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [46] H. Núñez, L. Gonzalez-Abril, and C. Angulo, “A post-processing strategy for SVM learning from unbalanced data,” *ESANN 2011 proceedings, 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 195–200, 2010.
- [47] L. Nanni, C. Fantozzi, and N. Lazzarini, “Coupling different methods for overcoming the class imbalance problem,” *Neurocomputing*, vol. 158, pp. 48–61, 2015.
- [48] M. Kubat, R. Holte, and S. Matwin, “Learning when negative examples abound,” in *European Conference on Machine Learning*, pp. 146–153, Springer, 1997.
- [49] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory undersampling for class imbalance learning,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 39, no. 2, pp. 539–550, 2009.
- [50] H. Li, *Learning to Rank for Information Retrieval and Natural Language Processing*, vol. 4. Morgan & Claypool, 2011.
- [51] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, “Learning to rank using gradient descent,” in *Proceedings of the 22nd international conference on Machine learning - ICML '05*, (New York, New York, USA), pp. 89–96, ACM Press, 2005.
- [52] R. Herbrich, T. Graepel, and K. Obermayer, “Support vector learning for ordinal regression a risk formulation for ordinal regression,” *Proceedings of the Ninth International Conference on Artificial Neural Networks*, pp. 97–102, 1999.
- [53] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, “An efficient boosting algorithm for combining preferences,” *The Journal of Machine Learning Research*, vol. 4, pp. 933–969, 2003.
- [54] W. Chu and S. S. Keerthi, “New approaches to support vector ordinal regression,” in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 145–152, ACM, 2005.
- [55] W. Chu and S. S. Keerthi, “Support vector ordinal regression,” *Neural computation*, vol. 19, no. 3, pp. 792–815, 2007.
- [56] L. Li and H.-T. Lin, “Ordinal regression by extended binary classification,” in *Advances in neural information processing systems*, pp. 865–872, 2007.



- [57] J. F. P. Costa, R. Sousa, and J. S. Cardoso, “An all-at-once unimodal SVM approach for ordinal classification,” in *9th International Conference on Machine Learning and Applications (ICMLA)*, pp. 59–64, IEEE, 2010.
- [58] H. Wang, Y. Shi, L. Niu, and Y. Tian, “Nonparallel support vector ordinal regression,” *IEEE Transactions on Cybernetics*, 2017.
- [59] J. S. Cardoso and J. F. Costa, “Learning to classify ordinal data: The data replication method,” *Journal of Machine Learning Research*, vol. 8, no. Jul, pp. 1393–1429, 2007.
- [60] M. Sahare and H. Gupta, “A review of multi-class classification for imbalanced data,” *International Journal of Advanced Computer Research*, vol. 2, no. 5, pp. 160–164, 2012.
- [61] E. Frank and M. Hall, “A simple approach to ordinal classification,” *Machine Learning: ECML 2001*, pp. 145–156, 2001.
- [62] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: Primal estimated sub-gradient solver for svm,” *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 2011.
- [63] K. Fernandes, J. S. Cardoso, and B. S. Astrup, “A deep learning approach for the forensic evaluation of sexual assault,” *Pattern Analysis and Applications*, vol. 21, no. 3, pp. 629–640, 2018.
- [64] M. P. Ortiz, K. Fernandes, R. Cruz, J. S. Cardoso, J. Briceno, and C. Hervás-Martínez, “Fine-to-coarse ranking in ordinal and imbalanced domains: An application to liver transplantation,” in *LNCS, 14th International Work-Conference on Artificial Neural Networks*, Springer, 2017.
- [65] C. Cortes and M. Mohri, “AUC optimization vs. error rate minimization,” in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 313–320, 2004.
- [66] R. Cruz, K. Fernandes, J. S. Cardoso, and J. F. P. Costa, “Tackling class imbalance with ranking,” in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016.
- [67] K. Chen, W. Lin, J. See, J. Wang, J. Zou, *et al.*, “AP-Loss for accurate one-stage object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [68] D. W. Scott, *Multivariate Density Estimation*. Wiley Series in Probability and Statistics, Hoboken, NJ, USA: John Wiley & Sons, aug 1992.
- [69] J. Cheng, Z. Wang, and G. Pollastri, “A neural network approach to ordinal regression,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1279–1284, IEEE, 2008.
- [70] J. Costa and J. Cardoso, “Classification of ordinal data using neural networks,” in *European Conference on Machine Learning*, pp. 690–697, 2005.
- [71] Q. Guan, Y. Huang, Z. Zhong, Z. Zheng, L. Zheng, and Y. Yang, “Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification,” *arXiv preprint arXiv:1801.09927*, 2018.

- [72] R. Cruz, J. F. P. Costa, and J. S. Cardoso, “Automatic augmentation by hill climbing,” in *International Conference on Artificial Neural Networks*, pp. 115–124, Springer, 2019.
- [73] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness,” in *International Conference on Learning Representations*, 2019.
- [74] National Agency of Electric Energy, “Underground energy distribution networks: current situation and assessment of the need to improve the associated regulation,” tech. rep., ANEEL, Brazil, 2014.
- [75] R. M. Prates, R. Cruz, A. P. Marotta, R. P. Ramos, E. F. S. Filho, and J. S. Cardoso, “Insulator visual non-conformity detection in overhead power distribution lines using deep learning,” *Computer and Electrical Engineering*, 2019.
- [76] R. Cruz, R. M. Prates, E. F. S. Filho, J. F. P. Costa, and J. S. Cardoso, “Background invariance by adversarial learning,” in *25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021.
- [77] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift,” 2019.
- [78] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680, 2014.
- [79] P. M. Ferreira, D. Pernes, A. Rebelo, and J. S. Cardoso, “Learning signer-invariant representations with adversarial training,” in *Twelfth International Conference on Machine Vision (ICMV 2019)*, vol. 11433, p. 114333D, International Society for Optics and Photonics, 2020.
- [80] J. A. Pereira, A. F. Sequeira, D. Pernes, and J. S. Cardoso, “A robust fingerprint presentation attack detection method against unseen attacks through adversarial learning,” in *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pp. 1–5, IEEE, 2020.
- [81] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *Journal of machine learning research*, vol. 2, pp. 45–66, 2001.
- [82] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, “Predicting the generalization gap in deep networks with margin distributions,” in *International Conference on Learning Representations*, 2019.
- [83] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1–10, 2015.
- [84] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, “Adversarial training for free!,” in *Advances in Neural Information Processing Systems*, pp. 3353–3364, 2019.

- [85] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, “Manifold mixup: Better representations by interpolating hidden states,” in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, (Long Beach, California, USA), pp. 6438–6447, PMLR, 09–15 Jun 2019.
- [86] Q. Chen and V. Koltun, “Photographic image synthesis with cascaded refinement networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1511–1520, 2017.
- [87] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [88] J. Fan, D. K. Yau, A. K. Elmagarmid, and W. G. Aref, “Automatic image segmentation by integrating color-edge extraction and seeded region growing,” *IEEE transactions on image processing*, vol. 10, no. 10, pp. 1454–1466, 2001.
- [89] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [90] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and Fully Connected CRFs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [91] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, “Conditional random fields as recurrent neural networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.
- [92] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari, “X-armed bandits,” *arXiv preprint arXiv:1001.4475*, 2010.