

Learning Visual Recognition Models with Limited Data

by

Saypraseuth MOUNSAVENG

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN
PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, OCTOBER 13, 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC



Saypraseuth Mounsaveng, 2023



This Creative Commons license allows readers to download this work and share it with others as long as the author is credited. The content of this work cannot be modified in any way or used commercially.

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Marco Pedersoli, Thesis supervisor
Department of System Engineering, École de Technologie Supérieure

Mr. Ismail Ben Ayed, Thesis Co-Supervisor
Department of System Engineering, École de Technologie Supérieure

Mr. Hervé Lombaert, Chair, Board of Examiners
Department of Software and Information Technology Engineering, École de Technologie Supérieure

Mr. Eric Granger, Member of the Jury
Department of System Engineering, École de Technologie Supérieure

Ms. Adriana Romero-Soriano, External Independent Examiner
Meta AI
McGill University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON SEPTEMBER 28, 2023

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

As this life chapter closes, I would like to express my deepest gratitude to my thesis director and co-director Marco and Ismail, who believed in me and gave me the opportunity to pursue this PhD endeavor despite my atypical profile. I would also like to warmly thank David, who coached me to become a better researcher during my internship at ElementAI, and my friends Simone and Jonathan, who convinced me to step out of my comfort zone and explore the academic world. Special thanks to my collaborators Issam, Pau, Florent, and Malik for helping me to get things done and progress towards the finish line.

I would also like to thank my Livia colleagues and friends for the interesting discussions and the nice moments we shared. Among other things, I remember the relaxing chalet weekend, the decompressing sport sessions, or the thrilling scavenger hunts at conferences. Special thanks to Djibril, Le Thanh, Jérôme, Hoel, Mathilde, Zko, Karthik, Rafael, Luiz Gustavo, Jonathan, Teruo, Mirmohammad, Heitor, Julien, Mohammad Hasan, and many others I forget. I also got a little help for those friends: Sukesh, Raghav, and Bala. Thank you for having tried to develop my tolerance to spicy food. Many thanks also to my ElementAI colleagues and friends Arantxa, Bahare, Chen, Dima, Konrad, and Thomas for their support, the crazy board and video games sessions, and the ping pong training.

I also deeply appreciated the exceptional support of my friends outside academia, especially during Covid: Sophie B.&Arnaud, Kristy&Vincent, Dorota&Romain, Raphaële&Manuel, Stéphanie&Régis, Aline&Guillaume, Stéphanie&Romain and Sophie D. (and of course their respective families). Thanks for your input during my transition to the academic world, the thrilling game and sport sessions, and for having literally fed my motivation. To my friends from the table tennis world: Bing, Cédric, Claudiu, Sergio, Martin, José David, Arnaud, Pierre-Yves, Guy, Laurent, and Elliot, thanks a lot to all of you for your encouragement and for having helped me to maintain my fitness to finish this adventure!

Last but not least, I am beyond grateful to my family, who despite the distance, constantly and unconditionally supported me during this adventure. Without them, daring to start and then finish this PhD would not have been possible. May this achievement also inspire the rising generation.

Apprentissage de modèles de reconnaissance visuelle dans un contexte de données et de ressources de calcul limitées.

Saypraseuth MOUNSAVENG

RÉSUMÉ

L'apprentissage profond, en particulier à travers l'usage des réseaux neuronaux profonds, a connu un franc succès dans le domaine de la vision par ordinateur. Les modèles à grande échelle, constitués de millions de paramètres, ont révolutionné le domaine en capturant des schémas complexes pour offrir des performances compétitives dans des tâches telles que la classification d'images, la détection d'objets ou la segmentation sémantique. L'entraînement avec des bases de données de grande taille est essentiel pour améliorer la généralisation des modèles, et permet des prédictions précises sur de nouvelles données. Malgré les performances exceptionnelles de ces modèles, des défis subsistent en raison des difficultés pouvant survenir lors de l'acquisition des données et du potentiel décalage entre les distributions des données d'entraînement et de test. Cette thèse vise à relever ces défis en explorant différentes façons d'optimiser l'apprentissage et l'adaptation des réseaux neuronaux profonds. Dans un premier chapitre, nous explorons l'utilisation de modèles génératifs pour créer de nouvelles images utiles pour une tâche sous-jacente. Plus particulièrement, nous exploitons la capacité des réseaux antagonistes génératifs (GAN) à générer de nouveaux échantillons augmentés permettant d'améliorer la robustesse et les performances d'un classificateur d'images. Contrairement aux transformations traditionnelles choisies de façon heuristique, l'approche présentée apprend l'augmentation de données optimale directement à partir des données d'entraînement en utilisant une architecture encodeur-décodeur et un réseau transformateur spatial, produisant des échantillons plus complexes au sein de la même classe. Dans un second chapitre, nous proposons une approche visant à réduire les calculs nécessaires pour déterminer la meilleure augmentation de données possible. Nous optimisons les paramètres d'augmentation à l'aide d'un ensemble de validation par une optimisation bi-niveaux, améliorant ainsi la généralisation du modèle sans avoir besoin d'une boucle de validation externe coûteuse. La méthode a été validée aussi bien sur des images naturelles que sur des images histologiques. Enfin, dans un troisième chapitre, nous explorons l'adaptation pendant l'inférence (TTA) et présentons une sélection et catégorisation de techniques TTA intéressantes pour adapter les modèles aux dérives de données. Ces techniques sont la normalisation par petits lots, le rebalancement des classes du flux, la sélection d'échantillons fiables et l'étalonnage de la confiance du réseau. Nous donnons un aperçu de leur impact dans différents scénarios et mettons en évidence des compromis nécessaires entre précision, puissance de calcul et complexité du modèle. Nous présentons également les synergies qui résultent de la combinaison de ces techniques. Les travaux présentés ouvrent de nouvelles voies pour des recherches futures et offrent des aperçus et des solutions pratiques pour l'entraînement et l'adaptation des réseaux neuronaux profonds dans un contexte de données limitées.

Mots-clés: classification d'images, augmentation de données, réseaux antagonistes génératifs, optimisation bi-niveaux, adaptation au moment du test

Learning Visual Recognition Models with Limited Data

Saypraseuth MOUNSAVENG

ABSTRACT

Deep learning, particularly through deep neural networks, has achieved remarkable success in computer vision. Large-scale models with millions of parameters have revolutionized the field, capturing complex patterns and improving performance across tasks like image classification, object detection, or semantic segmentation. Training with extensive datasets is key to enhancing model generalization, enabling accurate predictions on new data and adaptability to real-world complexities. However, despite the exceptional benefits, challenges arise due to the cost associated with data acquisition and the potential distribution shift between train and test data. This thesis aims at tackling those challenges and explores different ways to optimize the learning and adaptation of deep neural networks while maintaining or enhancing performance. In a first work, we explore the usage of generative models to generate images useful for a downstream task. More particularly, we leverage the power of generative adversarial networks (GAN) to generate new augmented samples useful to improve the training of an image classifier and increase its robustness and performance. Unlike traditional heuristic transformations, the approach presented learns data augmentation directly from training data using an encoder-decoder architecture and a spatial transformer network, producing more complex samples within the same class. In a second work, we further explore data augmentation and propose an efficient approach to reduce the computational power needed to define the best data augmentation parameters, improving generalization without requiring domain knowledge or an exhaustive search. We optimize augmentation parameters using a validation set through bi-level optimization, removing the need for an expensive external validation loop. We validated the method on natural images but also on histological images. Finally, in a third work, we explore test-time adaptation (TTA) and present a categorization of selected orthogonal TTA techniques interesting for adapting models to data drifts, such as small batch normalization, stream rebalancing, reliable sample selection, and network confidence calibration. We give insights into their impact on different scenarios, highlighting trade-offs in accuracy, computational power, and model complexity, while also revealing the synergies that arise from combining techniques. The presented works open up new avenues for further research, offering insights and practical solutions for training and adapting deep neural networks under challenging conditions.

Keywords: image classification, data augmentation, generative adversarial networks, bi-level optimization, test-time adaptation

TABLE OF CONTENTS

	Page
INTRODUCTION	1
0.1 Challenges	1
0.1.1 Data Acquisition Difficulty	1
0.1.2 Distribution Shift	2
0.2 Research objectives	3
0.2.1 Training a model with limited labeled data	4
0.2.2 Adapting a pretrained model using unlabeled data from target domain	4
0.3 Contributions	5
CHAPTER 1 LEARNING WITH LIMITED DATA: BACKGROUND	9
1.1 Introduction to Computer Vision and Deep Learning	9
1.1.1 Computer Vision	9
1.1.2 A brief history of Deep Learning for Computer Vision	10
1.2 Building blocks	11
1.2.1 Neural Networks	11
1.2.2 Convolutional Neural Networks	12
1.2.3 Transformers	18
1.2.4 Spatial Transformer Networks	19
1.2.5 Generative Adversarial Networks	22
1.3 Useful concepts and methods	24
1.3.1 Data Augmentation	24
1.3.2 Bilevel Optimization	27
1.3.3 Domain adaptation	28
CHAPTER 2 ADVERSARIAL LEARNING OF GENERAL TRANSFORMA- TIONS FOR DATA AUGMENTATION	33
2.1 Introduction	33
2.1.1 Contributions	36
2.2 Related Work	36
2.2.1 Standard Data Augmentation	36
2.2.2 Model-based Transformations	37
2.2.3 Adversarial Training	38
2.2.4 Generic Transformations with GANs	38
2.3 Proposed Model	39
2.3.1 Generation of augmented samples	40
2.3.2 Constraints on transformations	42
2.3.3 Classification	43
2.4 Experimental setup	44
2.4.1 Datasets	44

2.4.2	Implementation Details	45
2.4.3	Model architecture	45
2.5	Results	45
2.5.1	Comparison with Standard Data Augmentation	46
2.5.2	Comparison with State of the Art	47
2.5.3	Joint Training	48
2.5.4	Ablation Study	50
2.5.5	Generated Transformations	52
2.6	Conclusion and Discussion	52
CHAPTER 3 LEARNING DATA AUGMENTATION WITH ONLINE BILEVEL OPTIMIZATION FOR IMAGE CLASSIFICATION		55
3.1	Introduction	55
3.1.1	Contributions	58
3.2	Related Work	58
3.2.1	Generative images-based	59
3.2.2	AutoAugment	60
3.2.3	Hyperparameter Learning	61
3.3	Proposed Method	63
3.3.1	Approximate Online Bilevel Optimization	65
3.3.2	Augmenter Networks	67
3.4	Experimental Setup	69
3.4.1	Datasets	69
3.4.2	Evaluation	71
3.4.3	Implementation Details	71
3.5	Results	73
3.5.1	Natural images	73
3.5.2	Histological images	79
3.6	Conclusion and Discussion	86
CHAPTER 4 BAG OF TRICKS FOR FULL TEST-TIME ADAPTATION		89
4.1	Introduction	89
4.1.1	Contributions	91
4.2	Related Work	91
4.2.1	Test-time adaptation (TTA)	91
4.2.2	TTA in the broader literature	92
4.2.3	Fully TTA	93
4.3	Selected Tricks and Techniques	93
4.3.1	Architecture and Normalization	94
4.3.2	Class Rebalancing	94
4.3.3	Sample Selection	95
4.3.4	Calibration	97
4.4	Experimental Setup	97
4.4.1	Datasets	97

4.4.2	Implementation Details	98
4.5	Results	99
4.5.1	Architecture and Normalization	100
4.5.2	Class Rebalancing	101
4.5.3	Sample Selection	102
4.5.4	Calibration	103
4.5.5	Tricks Combination	103
4.5.6	Comparison to other methods and on other datasets	106
4.6	Conclusion and Discussion	107
CONCLUSION AND RECOMMENDATIONS		113
5.1	Summary of contributions	113
5.2	Limitations of our work	114
5.2.1	Automatic Data Augmentation Learning	115
5.2.2	Test-Time Adaptation	115
5.3	Future work	116
5.3.1	Data Augmentation Transformations Sampling	116
5.3.2	Further exploration of Foundation Models	116
APPENDIX I	SUPPLEMENTARY MATERIAL FOR CHAPTER 2 ADVERSARIAL LEARNING OF GENERAL TRANSFORMA- TIONS FOR DATA AUGMENTATION	118
APPENDIX II	SUPPLEMENTARY MATERIAL FOR CHAPTER 3 LEARNING DATA AUGMENTATION WITH ONLINE BILEVEL OPTIMIZATION FOR IMAGE CLASSIFICATION	123
APPENDIX III	SUPPLEMENTARY MATERIAL FOR CHAPTER 4 BAG OF TRICKS FOR FULL TEST-TIME ADAPTATION	125
BIBLIOGRAPHY		126

LIST OF TABLES

	Page
Table 2.1	Comparison of classification accuracy (%) with DA on different datasets 47
Table 2.2	Comparison of classification accuracy (%) to other automatic DA Methods 48
Table 2.3	Ablation Study on CIFAR-10 with 4000 training samples 50
Table 3.1	Impact and training cost of different geometric data augmentation strategies on classification accuracy (%) on CIFAR10 74
Table 3.2	Impact of architecture on classification accuracy (%) 75
Table 3.3	Impact of color and affine transformations on classification accuracy (%) on CIFAR10 76
Table 3.4	Classification Accuracy (%) of our model on different datasets 76
Table 3.5	Comparison of classification accuracy (%) with other models 78
Table 3.6	Impact of color and affine transformations on classification accuracy (%) 81
Table 3.7	Impact of the pretraining on the classification accuracy (%) on BACH dataset 85
Table 3.8	Comparison to a RandAugment based model 86
Table 3.9	Qualitative results 87
Table 4.1	Overview of TTA problem settings 92
Table 4.2	Impact of Additional Buffer on Tent accuracy (%) on different architecture on ImageNet-C in the single point learning scenario 102
Table 4.3	Impact of Temperature on classification accuracy (%) of Tent method performance on different architecture on ImageNet-C 104
Table 4.4	Effect of Tricks Combination on model accuracy (%) 105
Table 4.5	Accuracy (%) on ImageNet-C 106

Table 4.6	Accuracy (%) on ImageNet-Rendition	108
Table 4.7	Accuracy (%) on ImageNet-Sketch	109
Table 4.8	Accuracy (%) on VisDA2017	109

LIST OF FIGURES

	Page
Figure 1.1	Representation of images as a pixel matrix 13
Figure 1.2	Illustration of a convolution operation 14
Figure 1.3	Illustration of stride and padding 15
Figure 1.4	Illustration of Max pooling with a 2×2 filter and stride=2 16
Figure 1.5	Overview of different activation functions 16
Figure 1.6	Overview of a Residual Block 17
Figure 1.7	Two variants of the Residual Blocks. 17
Figure 1.8	Transformer model architecture 19
Figure 1.9	Spatial Transformer Networks architecture 20
Figure 1.10	STN sampler 21
Figure 1.11	GAN framework 22
Figure 2.1	Automatic data augmentation approaches 34
Figure 2.2	Proposed GAN model 40
Figure 2.3	Classification Accuracy (%) vs number of training samples on CIFAR10 47
Figure 2.4	Classification Accuracy (%) over epochs on 4000 samples of CIFAR10 for a baseline classifier and our joint training 49
Figure 2.5	Real and transformed images from MNIST, Fashion-MNIST, SVHN and CIFAR10 53
Figure 3.1	Model training 56
Figure 3.2	Computational graph of our model at iteration t=J 63
Figure 3.3	Qualitative results 79
Figure 3.4	Classification Accuracy (%) on BACH dataset in fonction of the amount of training data 84

Figure 4.1	Classification Accuracy (%) in function of Batch Size for different methods and architectures on ImageNet-C	90
Figure 4.2	Impact of Normalization, Architecture, and Batch Size on classification accuracy (%) of Tent method on ImageNet-C	100
Figure 4.3	Impact of Imbalance Factor, Architecture, and Batch Size on classification accuracy (%) of different methods on ImageNet-C	110
Figure 4.4	Impact of Architecture and Batch Size on the classification accuracy (%) of different methods on ImageNet-C	111
Figure 4.5	Impact of Sample Selection and Architecture on classification accuracy (%) of different methods on ImageNet-C	112

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
ViT	Vision Transformer
RN	ResNet
WRN	Wide ResNet
VGG	Visual Geometry Group
BN	Batch Normalization
GN	Group Normalization
LN	Layer Normalization
STN	Spatial Transformer Networks
GAN	Generative Adversarial Networks
WGAN	Wasserstein Generative Adversarial Networks
GD	Gradient Descent
i.i.d.	Independent and Identically Distributed
Adam	Adaptive Moment Estimation
RMSProp	Root Mean Square Propagation

XX

Acc	Accuracy
SFDA	Source-Free Domain Adaptation
TTA	Test-Time Adaptation
FTTA	Fully Test-Time Adaptation
TTT	Test-time Training
MNIST	Mixed National Institute of Standards and Technology
SVHN	Street View House Numbers
CIFAR	Canadian Institute for Advanced Research
BACH	BreAst Cancer Histology images
HICL	Histology Image Collection Library
GPU	Graphical Processing Unit
TPU	Tensor Processing Unit
RGB	Red Green Blue
HSV	Hue Saturation Value

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

\mathcal{X}	Space of images
\mathcal{Y}	Space of labels
\mathcal{T}	Space of transformations
x	Input image
y	One-hot label
z	Random latent variable
C	Classifier
D	Discriminator
G	Generator
A	Augmentor
θ	Network parameters
ω	Network weights

INTRODUCTION

Deep learning, has achieved remarkable success in visual recognition tasks, especially since the introduction of deep convolutional neural networks (CNN) (Lecun, Bottou, Bengio & Haffner, 1998) and more recently vision transformers (ViT) (Dosovitskiy *et al.*, 2021). Best performances are obtained by large-scale models with billions of parameters (Dehghani *et al.*, 2023), which exhibit a remarkable ability to capture complex patterns and relationships within the data. Training those models with extensive datasets (Russakovsky *et al.*, 2015) plays a crucial role in enhancing model generalization (Sun, Shrivastava, Singh & Gupta, 2017). By exposing a model to a diverse and extensive set of examples, it learns to generalize better, making accurate predictions on new, unseen data. The vast amount of data ensures that the model encounters a wide variety of scenarios, making it robust and adaptable to real-world complexities.

0.1 Challenges

If combining large-scale models and training with extensive datasets has proven to be the key to achieving exceptional performance (Sun *et al.*, 2017), it is not without challenges.

0.1.1 Data Acquisition Difficulty

To understand the scale of such large datasets, following examples can serve as informative points of reference. In August 2017, IBM set a new record in the Imagenet top-1 22K classes image classification challenge (33.8% accuracy) with a model trained with 7.5M images (Cho *et al.*, 2017). Another dataset of interest, Tencent-ML (Wu *et al.*), is composed of 18M images. In July 2017, Google created the JFT-300M dataset (Sun *et al.*, 2017), which contains 300M images. In May 2018, Facebook achieved a new record in the top-1 1k classes Imagenet image classification challenge (85.4% accuracy) by training their model with 1B images (Mahajan *et al.*, 2018). In a subsequent work, Facebook introduced the IG-1B-Targeted dataset (Yalniz,

Jégou, Chen, Paluri & Mahajan, 2019), which contains nearly 1B images. On a comparable scale, Mahajan *et al.* (2018) trained their model using 3.5B Instagram images.

In most real-life scenarios, accessing a large amount of annotated data is a significant challenge. First, gathering a large number of images can be challenging. For example, special equipment with limited availability can be required for the image acquisition, like for satellite images. Second, the nature of the problem itself can hinder data collection. For instance, in the medical imaging field, obtaining images of a rare pathology can prove challenging. Third, annotating a dataset, which is still often done manually by humans, can also require a significant effort: as the data amount and complexity (for example due to the level of expertise needed) increases, it becomes not only too expensive but also more prone to mistakes. Finally, cost constraints can further complicate the acquisition of a large amount of annotated images.

Employing a limited training dataset can lead to a reduction of model performance for two reasons. On the one hand, the information contained in the limited dataset might not be sufficient to capture the full complexity of the real dataset, resulting in a model unable to learn enough information to handle unseen data. On the other hand, the model may become too specific and overfit the limited dataset, leading to generalization issues.

0.1.2 Distribution Shift

Deep learning models are usually trained with the assumption that training data and test data seen at inference time come from the same distribution. However, when the training distribution (source) and the test distribution (target) are different, the issue of distribution shift arises, leading to a significant challenge in model generalization. As neural networks are trained on a specific dataset, they learn to capture patterns and features characteristic of that dataset's distribution. However, when a model encounters new data from a different distribution during inference, its performance tends to degrade due to the distributional mismatch. This shift in data distribution poses a formidable obstacle in real-world applications where the availability of

labeled data from the target domain is often limited. It also represents an interesting extension of the data scarcity problem mentioned in the previous paragraph.

Domain adaptation techniques aim at bridging the gap between the source and target domains to ensure robust and reliable neural network performance in varying and unforeseen environments. The idea of domain adaptation is to enable models to leverage knowledge gained from pre-training on a different yet related task or dataset. In this approach, a model is initially trained on a large, well-labeled dataset, and then the knowledge acquired is transferred to a target task with limited labeled data or only unlabeled data. The pre-trained model serves as feature extractor, allowing for improved performance with minimal data.

0.2 Research objectives

The motivations and potential rewards for addressing the problems presented in the previous paragraph are multiple. First, reducing the amount of labeled data required can yield time and cost savings, including necessary equipment and expertise. Second, enhancing the model's efficiency can provide interesting benefits in terms of energy consumption.

The main research objective of this thesis is to develop and analyze methods and techniques to make the best of a limited amount of data available to ensure a decent performance level of a visual recognition model. We tackle this main objective by elaborating on two specific objectives corresponding to two data scarcity scenarios. In the first one, what is available is a limited amount of labeled data, that we can use to train a model from scratch, and the objective is to develop efficient data augmentation techniques to improve the model performance. In the second one, what we have at disposal is a model pretrained on a large scale dataset and unlabeled data from a possibly shifted target domain and the objective is to assess empirically recently proposed methods leveraging only the unlabeled data to adapt the pretrained model. We will now give more details on each scenario.

0.2.1 Training a model with limited labeled data

Obtaining a sufficient amount of annotated data to train a visual recognition model can be a significant challenge in many real-world scenarios. This data scarcity problem poses a hurdle to achieving optimal model performance. In response, researchers have explored various techniques to address this issue and make the most of the available data. For example, in situations where acquiring fully labeled datasets is difficult, using different level of supervision can relax the need for fully annotated data, by leveraging unlabeled data or partial annotations. Another solution is data augmentation, which is a widely used technique to increase the effective size of the dataset. There exists two ways of doing data augmentation: first, by generating new samples from a learned representation of the dataset, second, by applying various transformations to existing samples. These transformations do not alter the class labels of the existing data points but provide valuable diversity to the training process. Common augmentations include rotations, flips, translations, and changes in lighting conditions. By expanding the dataset with augmented samples, the model becomes more robust and better generalizes to unseen variations in the data. More details are available in Section 1.3.1 of Chapter 1.

In a first contribution, we tried to leverage the power of generative adversarial networks to create useful samples to improve the training of a classifier. Then, building on the limitations of the approach we proposed, we chose in a second contribution to investigate a way to learn automatically the best data augmentation transformations using a bilevel optimization framework.

0.2.2 Adapting a pretrained model using unlabeled data from target domain

The second scenario considered is a case of distribution shift. In such cases, domain adaptation techniques are needed to ensure a reliable performance of the model on the target data. Domain adaptation involves utilizing knowledge acquired from one task or domain to improve performance on another related task. Pretrained model, especially recently emerging foundation models

(Bommasani *et al.*, 2021), are often built using large-scale datasets and extensive training and capture intricate patterns and representations that are transferable across various domains. By using these pretrained models as a starting point, good performances can be obtained by reducing the amount of data and computational resources required for training on new tasks. More details are available in Section 1.3.3 of Chapter 1.

In this thesis, we investigate Fully Test-Time Adaptation, which is a particular case of domain adaptation focusing on leveraging pretrained models during inference to adapt to the specific characteristics of the target domain. Rather than fine-tuning the entire model, test-time adaptation involves fine-tuning only certain model components, enabling rapid adaptation to new domains and reducing the computational overhead during the adaptation process. We analyze and categorize different source free test-time adaptation techniques recently proposed, highlighting weaknesses but also opportunities to improve performance, for example by using specific combinations of techniques.

0.3 Contributions

The core contributions of this thesis are:

- In Chapter 2, we leverage the power of generative adversarial networks to generate new samples augmented with general transformations, useful to make the trained model more robust and increase its performance. Unlike traditional heuristic transformations, the approach presented learns data augmentation directly from training data using an encoder-decoder architecture and a spatial transformer network, producing more complex samples within the same class.

Related publication:

- Adversarial Learning of General Transformations for Data Augmentation. Published at *Learning with Limited Data Workshop. International Conference on Learning Representa-*

tions (ICLR), 2019.

- In Chapter 3, we propose an efficient approach to reduce the computational power needed to define the best data augmentation when training an image classifier, improving generalization without requiring domain knowledge or an exhaustive search. We optimize augmentation parameters using a validation set through bi-level optimization, removing the need for an expensive external validation loop.

Related publications:

- Data Augmentation with Online Bilevel Optimization for Image Classification. Published at IEEE Winter Conference on Applications of Computer Vision (WACV), 2021.
 - Method of and system for joint data augmentation and classification learning. *US Patent App. 16/778,480*, 2021.
 - Automatic Data Augmentation Learning using Bilevel Optimization for Histopathological Images. Submitted to the *Journal of Machine Learning for Biomedical Imaging (MELBA)*, 2023.
- Finally, in Chapter 4, we explore test-time adaptation (TTA) and present a categorization of selected orthogonal TTA techniques interesting for adapting models to data drifts, such as small batch normalization, stream rebalancing, reliable sample selection, and network confidence calibration. We give insights into their impact on different scenarios, highlighting trade-offs in accuracy, computational power, and model complexity, while also revealing the synergies that arise from combining techniques.

Related publication:

- Bag of Tricks for Fully Test-Time Adaptation. Submitted to *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2024.

To facilitate further research and improve the reproducibility of results, the code of all the papers is public and available online.

CHAPTER 1

LEARNING WITH LIMITED DATA: BACKGROUND

This chapter introduces the background of our research. We begin with a short presentation of computer vision and deep learning, followed by the introduction of the essential technical building blocks required to understand the presented work. Subsequently, we cover key concepts like data augmentation, bi-level optimization and conclude with domain adaptation.

1.1 Introduction to Computer Vision and Deep Learning

In this section, we introduce briefly the different computer vision tasks and provide a short history of Deep Learning for Computer Vision.

1.1.1 Computer Vision

Computer vision is a multidisciplinary field aimed at teaching computers a high-level understanding of digital images. It encompasses a wide range of tasks that mirror human visual understanding and perception. For example, Pose Estimation determines the position and orientation of specific objects, such as the human body, within an image. Optical Character Recognition (OCR) focuses on translating visual text into machine-encoded text. In videos, Action Recognition seeks to identify specific activities or behaviors. Depth Estimation and 3D Reconstruction are geared toward understanding the three-dimensional structure of a scene.

Object Recognition is usually structured around four different tasks of increasing complexity. Image Classification categorizes images into predefined classes by recognizing the single object they contain. Object Localization goes a bit further by not only classifying an image according to the object it contains, but also locating this object within the image. Object Detection extends Object Localization to images containing several objects. Finally, Object Segmentation goes one step further by classifying each individual pixel of an image. These tasks, collectively, represent the broad spectrum of challenges in computer vision and aim to bridge the gap between visual data and meaningful interpretation, with applications ranging from autonomous driving

to medical imaging.

In this thesis, we focus our attention on Image Classification and, more particularly, the learning methodologies, which could potentially be extended to other tasks.

1.1.2 A brief history of Deep Learning for Computer Vision

The foundation of deep learning lies in the McCulloch-Pitts neuron model (McCulloch & Pitts, 1943), which draws inspiration from the functioning of the biological neurons in the brain. The first learning network, known as the perceptron (Rosenblatt, 1958), served as a basic binary classifier before evolving into the multi-layer perceptron (MLP) with hidden layers (Rosenblatt, 1963). A significant breakthrough in deep learning occurred in 1986 with the application of the backpropagation algorithm to artificial neural networks (Rumelhart, Hinton & Williams, 1986), enabling the training of MLPs and allowing the learning of non-linear representations. Subsequently, Convolutional Neural Networks (CNNs) further revolutionized the field. CNNs, like the LeNet-5 model (Lecun *et al.*, 1998), harnessed local feature hierarchies through concepts such as local receptive fields, shared weights, and spatial or temporal subsampling, leading to remarkable performance improvements in computer vision tasks.

Later on, two major factors contributed to the success of deep learning. First, the development of specialized hardware such as Graphical Processing Units (GPU) or more recently Tensor Processing Units (TPU), allowed the design of significantly larger architectures. After LeNet-5, the following significant architectures in deep learning for computer vision have centered around CNNs, with architectures like AlexNet (Krizhevsky, Hinton *et al.*, 2009), VGG (Simonyan & Zisserman, 2015), Inception (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2016), ResNet (He, Zhang, Ren & Sun, 2015), DenseNet (Huang, Liu, van der Maaten & Weinberger, 2017) and EfficientNet (Tan & Le, 2019), progressively advancing the state-of-the-art in vision tasks. However, more recently, attention-based models, inspired by the Transformer architecture (Vaswani *et al.*, 2017) initially designed in the Natural Language Processing domain, have displayed promising results in computer vision tasks. Vision Transformers (ViT) (Dosovitskiy *et al.*, 2021) have emerged as powerful contenders, outperforming CNNs in image classification by treating

the image as a sequence of patches and utilizing self-attention mechanisms to capture global interactions. Vision transformers are an active area of research and many improvements of the Vanilla ViT have already been proposed (Touvron *et al.*, 2021; Wang *et al.*, 2021b; Han *et al.*, 2021; Liu *et al.*, 2021b; Dong *et al.*, 2022). Second, the creation of large training datasets, pioneered by ImageNet (Russakovsky *et al.*, 2015), has largely contributed to facilitating the training of deep neural networks.

Deep learning remains an active area of research. Indeed, despite its remarkable success in various domains, it exhibits certain limitations that warrant further exploration and study. First, deep learning models typically require substantial amounts of data for training, which may not be available in all contexts. Second, the interpretability of these models remains a challenge, as they often function as “black boxes”, making it difficult to understand the underlying mechanisms driving their decisions. Additionally, deep learning is computationally intensive, requiring significant hardware resources, and thus may not be accessible to all researchers or industries. Lastly, issues related to model robustness and generalization need to be addressed, as small changes in input data can lead to vastly different outputs, as illustrated by adversarial samples. The future of research in this area likely lies in developing methodologies that mitigate those challenges.

1.2 Building blocks

In this section, we introduce the different components of neural networks and provide insights on the architectural elements used in the different works presented in this thesis.

1.2.1 Neural Networks

Neural networks are computational models inspired by the human brain’s interconnected neurons, designed to recognize patterns and make decisions by processing input data through layers of artificial nodes called “neurons”. They began with the concept of perceptron, a simple binary classification algorithm developed by Frank Rosenblatt in the late 1950s (Rosenblatt, 1958). The

perceptron served as a foundational model with a single neuron, modeling a threshold function with weighted input connections (weights are noted w), a bias b and an activation function, mapping its input x (a real-valued vector) to an output value $f(x)$ (a single binary value):

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the context of neural networks, a perceptron is an artificial neuron using the Heaviside step function as the activation function. Artificial neurons are simple linear transformers, multiplying an input vector x by a matrix of learnable weights W and adding a bias vector b , outputting a transformed vector y called logits:

$$y = Wx + b$$

In following sections, we will see the role of activation functions and introduce some of them. The limitation of the perceptron, which is the simplest feedforward neural network, was its inability to solve problems that were not linearly separable. This led to the evolution of Multi-Layer Perceptrons (MLP) Rosenblatt (1963), which consist of multiple layers of interconnected neurons, thus enabling the modeling of more complex, non-linear relationships. MLPs marked a significant advancement in neural network capabilities, but still had limitations in handling spatial hierarchies in data, such as images. The transition to Convolutional Neural Networks (CNNs) addressed this issue.

1.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) find their roots in the pioneering work of neurobiologists Hubel&Wiesel in the early 1960s (Hubel & Wiesel, 1959, 1962), which focused on the visual cortex of cats and monkeys. In their experiments, they recorded neuron activations in response to a bright line presented to their test subjects and identified two distinct types of neurons crucial to visual perception. The first type, simple cells (S-cells), responded to lines at specific positions and orientations on the retina, having similar receptive fields that overlapped, while the second type, complex cells (C-cells), exhibited larger receptive fields and were sensitive to

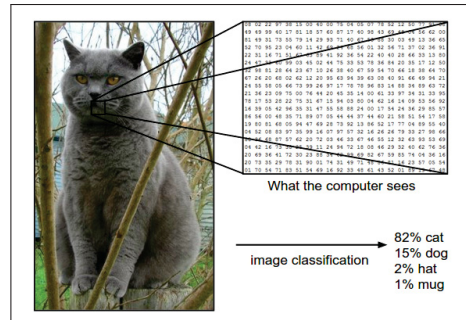


Figure 1.1 Representation of images as a pixel matrix
Taken from Li *et al.* (2023)

line orientation regardless of position. Inspired by this biological model, Fukushima (1980) introduced the neocognitron, an artificial neural network mimicking the characteristics of S-cells and C-cells and their hierarchical arrangement. CNNs emerged as a natural extension of the neocognitron, offering enhanced capabilities in various tasks, including image recognition. One notable milestone in the evolution of CNNs was the introduction of LeNet-5 in Lecun *et al.* (1998). This model, designed for handwritten character recognition, laid the foundations for successful object recognition and contributed significantly to advancing CNNs.

Convolutional neural networks are composed of different elements:

Input The input of a CNN, i.e. what the model sees, is digital images. Similarly to the light exciting the retina cells, the pixels of an image or more precisely their value, as illustrated in Fig.1.1, are stimulating the so-called filters (or kernels). In Hubert&Wiesel's model, those filters are equivalent to the receptive fields. The input layer is followed by a block or a series of block composed of a combination of a convolutional layer, an activation layer and a pooling layer.

Convolution The stimulation of the network is achieved by applying an operation called convolution to the image pixels and the filters, resulting in an output called a feature map.

Fig. 1.2 illustrates the application of the convolution operation between the input (the blue square) and the filter (the green square). The sum of the convolution outputs the feature map

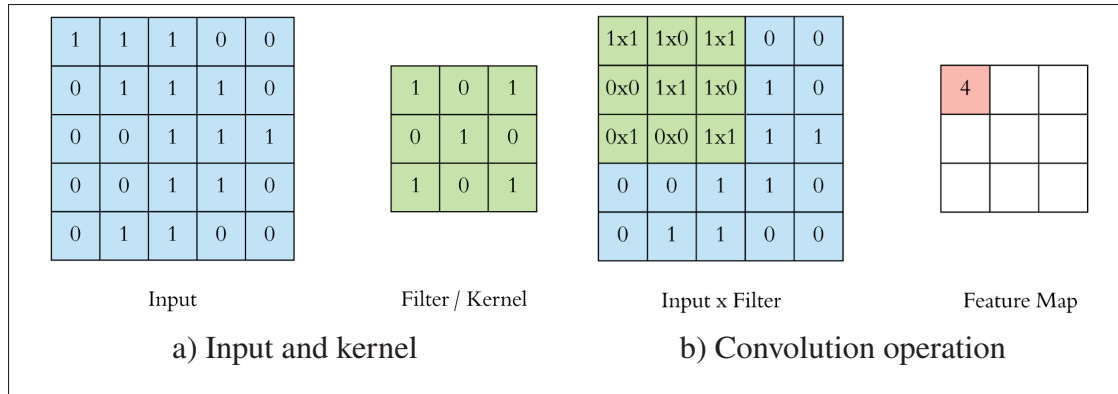


Figure 1.2 Illustration of a convolution operation

(a) In image processing, input and filter are represented as matrices (b) The convolution operation is a dot product operation between the input and the filter
Adapted from Dertat (2017)

(the red square). In the case of a multi-layer model, the input of the first layer is the input image, and the subsequent layers take as input the feature maps produced by the previous layer. In mathematical terms, a convolution is the combination of two functions to produce a third function. Intuitively, this third function represents a similarity measure between the two input functions. In a discrete case, the convolution operation is expressed as:

$$f \circ g(x) = \sum_{n=0}^{N-1} f(n)g(x-n) \quad (1.1)$$

or in the 2D case:

$$f \circ g(x, y) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n)g(x-m, y-n) \quad (1.2)$$

where f and g are the two input functions and N the size of the convolution filter.

A convolution can be interpreted as a sliding dot product between the matrix containing the image pixels and the filters of the first layer of the network, or between the output feature maps of a layer and the filters of the next layer. Interesting to note is that strictly speaking, a convolution in deep learning is actually a cross-correlation in signal/image processing.

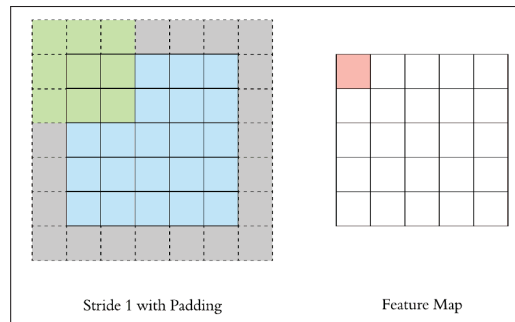


Figure 1.3 Illustration
of stride and padding
Adapted from Dertat (2017)

The distance the filter is slid horizontally or vertically on the input matrix during a convolution operation is called stride. To control the size of the features maps, for example to avoid having them too small, it is sometimes necessary to add elements at the border of the input matrix. The number of rows or columns added on each side of the input matrix is called padding. Stride, padding as well as the dimension of the filters are hyperparameters of CNNs defined in the design phase of the model.

Pooling As illustrated in Fig. 1.4, the pooling layer downsamples the layer's input by applying a function to non-overlapping groups of pixels, typically using the max function to select the highest pixel value, or other functions like average, ultimately reducing computational power and memory required for model training by reducing parameters.

Activation In a neural network, the stimulation of the neurons are done with linear operations. The raw values delivered by those layers are called logits. However, high-dimensionality data usually exhibits a nonlinear relationship between the inputs and the outputs. To reflect that, activation layers were introduced. Intuitively, the role of an activation layer is to decide, based on the logits, if a neuron should fire or not and the intensity of the firing.

There exists several activation functions. Sigmoid (Fig. 1.5a) was initially proposed to simulate

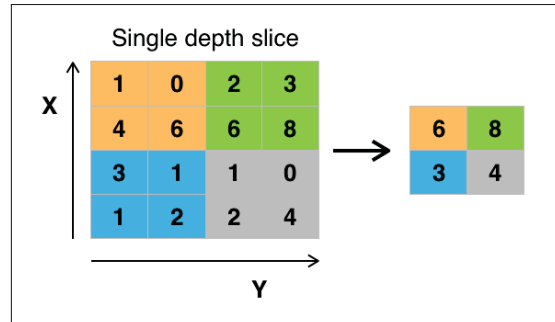


Figure 1.4 Illustration of Max pooling with a 2×2 filter and stride=2
Taken from Wikipedia contributors (2023)

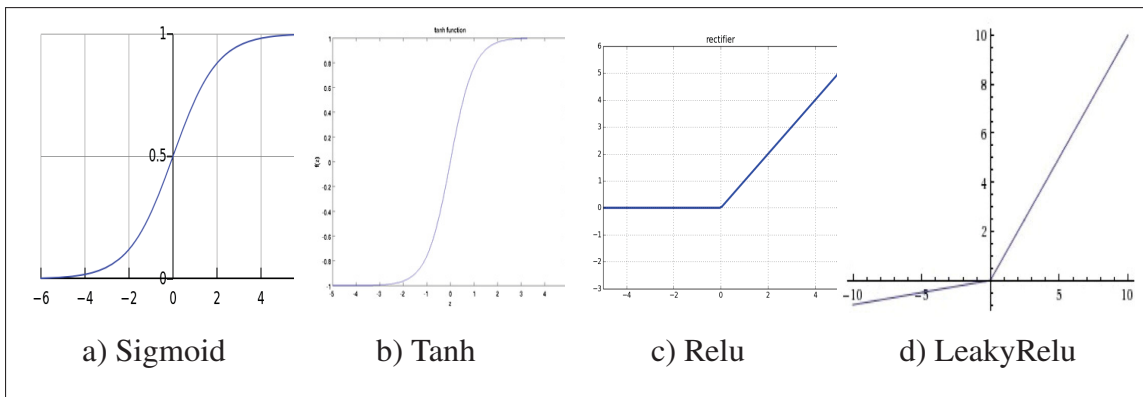


Figure 1.5 Overview of different activation functions
Adapted from Dertat (2017)

biological neurons. This function, bounded between 0 and 1, is defined as:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (1.3)$$

It used to be one of the most used activation function but tends to be replaced as it can suffer from vanishing gradients problems during the backpropagation phase.

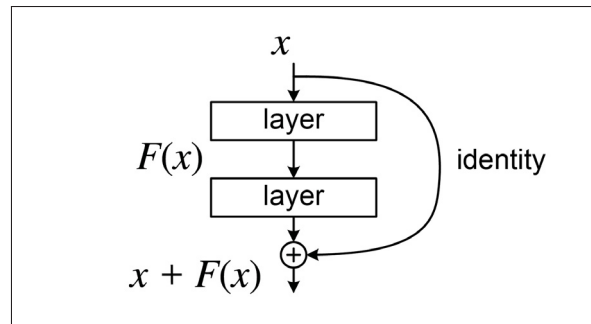


Figure 1.6 Overview of a Residual Block
Taken from He *et al.* (2015)

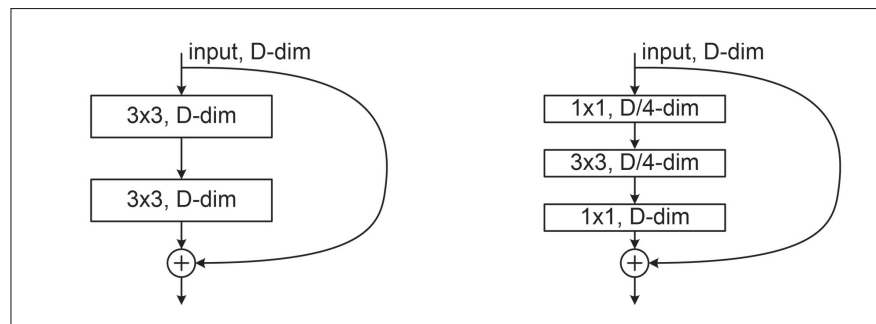


Figure 1.7 Two variants of the Residual Blocks
Left: a Basic Block with two 3×3 convolutional layers. Right:
a Bottleneck Block with a 1×1 convolutional layer for dimension
reduction followed by a 3×3 convolutional layer, and
a final 1×1 convolutional layer for dimension restoration.
Taken from He *et al.* (2015)

Residual connections Residual networks (He *et al.*, 2015), or ResNets, are a prominent architecture in deep learning that addresses the vanishing gradient problem, particularly in very deep networks. They incorporate residual connections, also known as skip connections, which are shortcuts allowing the gradient to be directly back-propagated to earlier layers. As shown in Fig. 1.6, these connections bypass one or more layers and add the output from an earlier layer to a later layer. The inclusion of skip connections promotes the training of deep networks by mitigating the loss of information through the network's depth, thereby improving the ability to capture complex patterns and relationships in data.

Later on, different variants of the residual blocks were introduced, like the basic block or the bottleneck block, as illustrated in Fig. 1.7. This architectural innovation has paved the way for more efficient training and the design of deeper neural networks, as described in He *et al.* (2015). For example, a ResNet50 model is composed of a stack of 50 residual layers, whereas a ResNet152 has a total of 152 residual layers. Interesting to note is that residual connections are not limited to CNNs, and are also used in more recent architecture like Transformers presented in the following section.

1.2.3 Transformers

Transformers, originating from seminal work (Vaswani *et al.*, 2017) in the Natural Language Processing (NLP) field, have recently been adapted to the domain of computer vision. The key idea behind transformer networks is to leverage the power of self-attention mechanisms to connect all input elements directly to each other and establish long range relationships between them.

The core components of transformer networks are multi-head self-attention layers and feed-forward neural networks. The self-attention mechanism enables the model to assign different weights to various elements in the input sequence, depending on their relevance to each other. The multi-head attention employs multiple sets of attention weights to capture different types of dependencies, allowing the model to attend to multiple aspects of the input. Additionally, transformer networks use positional encoding to provide the model with information about the order of the input elements, since the self-attention mechanism alone does not inherently capture sequential information.

Vision Transformers (ViTs) were introduced in Dosovitskiy *et al.* (2021) and then improved in Deit (Touvron *et al.*, 2021), PVT (Wang *et al.*, 2021b), TNT (Han *et al.*, 2021), SWIN (Liu *et al.*, 2021b) or more recently CSWIN (Dong *et al.*, 2022). These models aim at leveraging the transformer's capacity to model global dependencies within data, rather than relying on the locality-focused, convolution-based strategies that have dominated the field of vision for

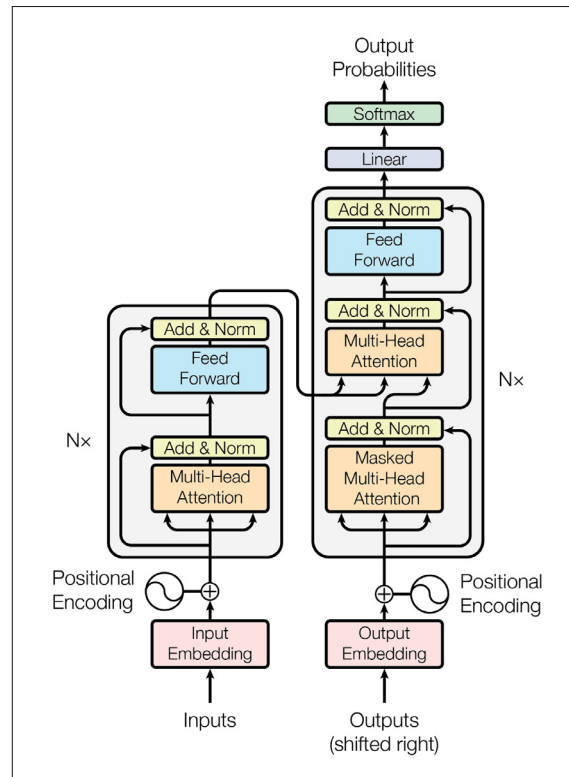


Figure 1.8 Transformer model architecture
Taken from Vaswani *et al.* (2017)

years. Unlike traditional Convolutional Neural Networks (CNNs), ViTs partition an image into a sequence of patches and process these as inputs in a manner similar to a sequence of words in a sentence. This approach demonstrates the potential to better understand intricate and global patterns in image data, marking a paradigm shift in visual data processing.

1.2.4 Spatial Transformer Networks

In previous paragraphs, we gained an insight into the CNNs and Transformers architecture. One significant weakness of CNN layers is that they are equivariant under translation and not invariant to any other transformations. Incorporating equivariance or invariance knowledge in the representation learning in vision tasks is important to improve the performance of the trained model. If this knowledge is available, data augmentation can be used by applying selected

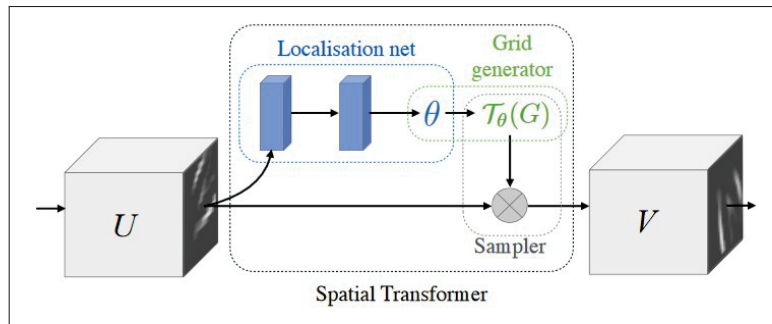


Figure 1.9 Spatial Transformer Networks architecture
Taken from Jaderberg *et al.* (2015)

transformations to the training dataset to learn a representation invariant to those transformations. If this knowledge is not available, useful transformations can be learned directly from the data.

Several works have proposed extensions to the standard CNN architecture to make networks invariant to certain types of spatial transformations, for example Group Equivariant CNNs (Cohen & Welling, 2016; Cohen, Weiler, Kicirko & Welling, 2019). Spatial Transformer Networks (STN) (Jaderberg, Simonyan, Zisserman *et al.*, 2015) is a module that can be inserted one or several times into a CNN architecture to make the model invariant to affine transformations by learning an input dependent transformation. An input can be an image if the STN module is added at the beginning of a CNN model or feature maps if the STN module is inserted after an intermediate layer. Despite similar names, it is important to note that Spatial Transformer Networks are not to be confused with Transformers presented in the previous section. It is also essential to understand the difference between STNs and data augmentation. While data augmentation contributes to a model's invariance to certain transformations by introducing new samples exhibiting those transformations during training, STNs aim to align data by eliminating such variations.

Fig. 1.9 illustrates the three parts of the STN: i) The localization network takes an image or feature map as input and outputs the transformation parameters θ , representing the transformation \mathcal{T}_θ to be applied (for example a 2×3 matrix for affine transformations). ii) The grid generator learns a mapping between input and output images or feature maps using the parameters θ from

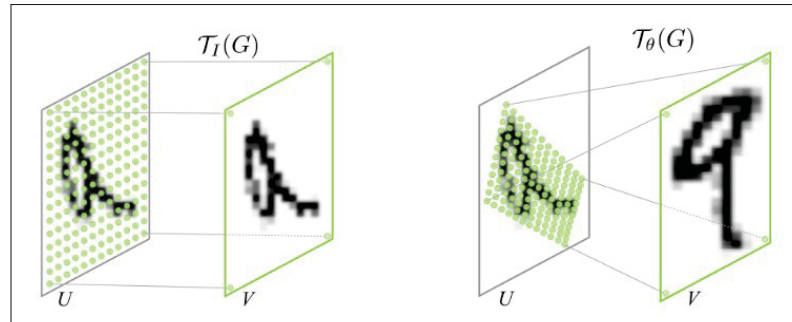


Figure 1.10 STN sampler
Taken from Jaderberg *et al.* (2015)

the localization network. For example, the pointwise transformation in an affine case can be formulated as:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

where (x_i^t, y_i^t) are the target coordinates of the grid in the output, G_i the input grid, \mathcal{T}_θ the transformation learned by the localization net, (x_i^s, y_i^s) are the source coordinates in the input image or feature map and A_θ is the affine transformation matrix. iii) The sampler uses the mapping defined by the grid generator, taking input U (image or feature map) to generate the output V (image or feature map), depending on the STN module's position in the network.

On Fig. 1.10, we can see two examples of transformations applied to an input grid. \mathcal{T}_I is the identity transformation and \mathcal{T}_θ is an affine transformation (rotation in this case). As the module is fully differentiable, it can be trained during the training of the whole model without any changes in the optimization procedure.

Following seminal STN work (Jaderberg *et al.*, 2015), improvements were proposed to extend the space of transformations learned, for example Deep diffeomorphic transformer networks (Skafte Detlefsen, Freifeld & Hauberg, 2018), Polar transformer networks (Esteves, Allen-Blanchette, Zhou & Daniilidis, 2018) or Equivariant transformer networks (Tai, Bailis & Valiant, 2019).

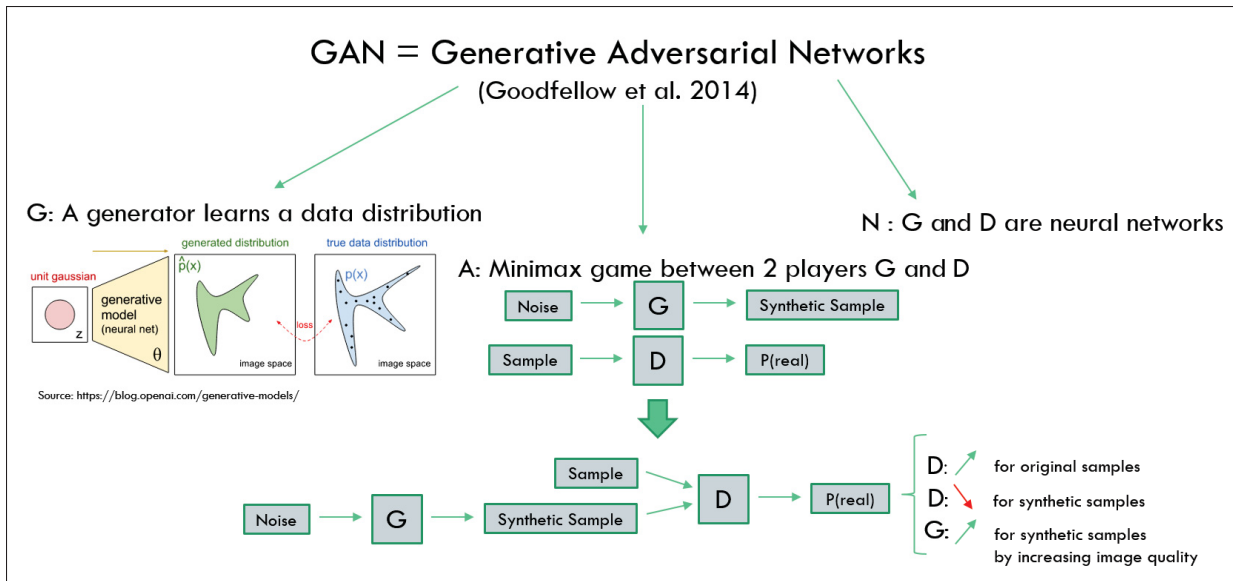


Figure 1.11 GAN framework

1.2.5 Generative Adversarial Networks

In the previous section, we introduced Spatial Transformer Networks, an architecture where two networks collaborate. However, there also exist architectures, such as generative adversarial networks, in which the networks function in an adversarial manner. Generative adversarial networks (GANs) were introduced in seminal work by Goodfellow et al. (Goodfellow *et al.*, 2014). They are based on a minimax-type game where two networks with antagonist goals learn simultaneously. The first player, known as the generator, tries to generate samples that are as close as possible to a given training dataset and improves iteratively by incorporating the feedback provided by the second player, the discriminator. The discriminator's role is to distinguish "genuine" samples coming from the original training dataset from "fake" samples generated by the generator. Given an input sample, the discriminator yields a probability that this sample belongs to the original dataset. An illustration of the GAN framework is given in Fig.1.11.

In the GAN literature, the usual concept used to illustrate GANs is the one of a money counterfeiter trying to fool the police. The GAN generator is the money counterfeiter and strives

to produce banknotes that are as realistic as possible, whereas the GAN discriminator plays the role of the police, trained to discern counterfeit from genuine notes. By getting feedback from the discriminator on the quality of the generated samples, the generator improves continuously to the point of successfully deceiving the discriminator. The discriminator is then not able to differentiate the fake samples from the genuine ones anymore.

The first GAN architecture described in Goodfellow *et al.* (2014) is based on deep neural networks. Both the generator and the discriminator are multi-layer perceptrons (MLP). They are trained using gradient based optimization methods like stochastic gradient descent (SGD), Adaptive Moment Estimation (Adam) or RMSProp. The cost functions are different for the generator and the discriminator as they have different goals, but they are trained jointly in a minimax game.

The global minimax objective function is defined as:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x \sim p_{data}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p_z} \log (1 - D_{\theta_D}(G_{\theta_G}(z)))$$

where D and G represent the discriminator and generator networks, θ_D and θ_G their respective parameters, x a real example and z a random noise vector. $D_{\theta_D}(x)$ is the discriminator output for a real example x . $D_{\theta_D}(G_{\theta_G}(z))$ is the discriminator output for a fake example generated by the generator.

The GAN learning process consists in training alternately the discriminator and the generator. The discriminator D_{θ_D} tries to maximize the objective so that $D_{\theta_D}(x)$ gets close to 1 (real) and $D_{\theta_D}(G_{\theta_G}(z))$ gets close to 0 (fake). After that, the generator G_{θ_G} tries to minimize the objective so that $D_{\theta_D}(G_{\theta_G}(z))$ gets close to 1. In other words, it tries to fool the discriminator into thinking the generated sample $G_{\theta_G}(z)$ is real. Another way for G_{θ_G} to do that is to maximize $\log D_{\theta_D}(G_{\theta_G}(z))$, which yields better empirical results in terms of gradient conservation.

GANs have found extensive application in the imaging field, serving not only for image generation but also in diverse areas such as image reconstruction, image coloring, and image

super-resolution. Following seminal work Goodfellow *et al.* (2014), different variants and improvements have been proposed to improve the image resolution or stabilize the training. For example, DCGAN (Radford, Metz & Chintala, 2016) uses CNNs instead of simple MLPs, Wasserstein GAN (WGAN) (Arjovsky, Chintala & Bottou, 2017) stabilizes the training by using the earthmover algorithm to optimize the loss function, BigGAN (Brock, Donahue & Simonyan, 2019) compiles several architecture improvements to scale GAN architectures to generate high resolution images.

However, due to the complexity of their training and the limitations in their scalability, preventing the generation of high resolution images, GANs have been replaced recently by Diffusion models Dhariwal & Nichol (2021) introduced in Song & Ermon (2019) and Ho, Jain & Abbeel (2020).

1.3 Useful concepts and methods

In the previous section, an overview of the essential technical components of this thesis was introduced. In this section, we will cover key concepts and methods, starting with data augmentation, which represents the main topic of this thesis. Then we will explore bilevel optimization, an optimization technique we used to enhance model training efficiency. Finally, we will focus on domain adaptation, an important topic in data scarcity scenarios.

1.3.1 Data Augmentation

Data augmentation is a form of regularization commonly used to train deep neural networks and consists in creating new data points from an existing dataset to get a larger one. It was found essential for achieving state-of-the-art image classification results (Hernández-García & König, 2018b).

We can first distinguish between online and offline data augmentation. Online Data Augmentation involves dynamically generating augmented data during the training process, thereby introducing randomness and potentially enhancing the model's ability to generalize. This method allows for endless variations in the data, but it can increase the computational cost, as the augmentations

are applied on-the-fly. Conversely, Offline Data Augmentation involves creating and storing a fixed set of augmented data prior to the initiation of training. This approach can make the training process more computationally efficient, as the augmented data are readily available. However, it may require significant storage space and lacks the dynamic variability provided by online augmentation, limiting the model's exposure to diverse transformations.

Data augmentation can usually take two forms. The first one, which is also the most common one, is the creation of new samples by applying transformations to existing ones. Those transformations can be geometric transformations, for instance affine transformations like flipping, mirroring, cropping, scaling, etc.. but also color transformations like contrast or brightness alterations or style transfer. The second form of data augmentation is based on image generation, for example with generative adversarial networks. GANs learn the data distribution of the dataset and generate new samples by sampling from this learnt distribution. More recently, Diffusion Models (Dhariwal & Nichol, 2021) have also been explored for data augmentation (Hataya, Bao & Arai, 2023).

Transformation based data augmentation Transformation based data augmentation consists in extending the dataset at disposal by applying transformations to the existing samples to create new ones, without changing the class labels. The selection of the data augmentation transformations is usually done using heuristics or expert domain knowledge. Usual transformations for natural images are image flip, rotation and color changes as described in Perez & Wang (2017); Ciresan, Meier, Gambardella & Schmidhuber (2012); Krizhevsky, Sutskever & Hinton (2012). However, the usage of more complex transformations such as occluding parts of an image (DeVries & Taylor, 2017b) or blending images (Zhang, Cisse, Dauphin & Lopez-Paz, 2018; Lemley, Bazrafkan & Corcoran, 2017) has also been proposed. Recently, some works have investigated adversarial samples as a possibility of data augmentation (Gong, Ren, Ye & Liu, 2021; Suzuki, 2022). However, transformations in data augmentation are not limited to geometric transformations. Some work have explored the impact of color transformation on the performance of image classification models. Karargyris (2015) learns the transformation of the color space that will improve the most the classification accuracy. Krizhevsky *et al.* (2012) proposes a

decomposition of the color spectrum following the Principal Component Analysis method in order to keep and boost the colors that contribute the most to the variance of the dataset. Perez & Wang (2017) compares data augmentation based on style transfer to geometric and GAN based approaches in the context of image classification. Besides geometric and color centric transformations, Lemley *et al.* (2017); Zhang *et al.* (2018); Takahashi, Matsubara & Uehara (2018) create new samples by merging or patching two or more samples drawn randomly from the same class and DeVries & Taylor (2017b) by occluding part of an image.

Generated images based data augmentation Instead of transforming samples, another way to do data augmentation is to create new samples by leveraging the image generation capacities of generative models like generative adversarial networks or diffusion models. GANs learn the data distribution of a particular dataset, and new samples are then created by sampling from this learned distribution. There are several ways to use GANs in the context of data augmentation for visual recognition tasks. The first one is to use GANs as simple image generators. Odena, Olah & Shlens (2017) and Mirza & Osindero (2014) propose to generate new samples conditioned on a class label. Instead of learning a mapping from random noise to the image space, Antoniou, Storkey & Edwards (2018) learn an image-to-image mapping and create new samples by transforming input images. One weakness of those approaches is that a preprocessing step to learn data augmentation is needed additionally to the training of the end task itself. In the context of image classification, some GAN models like Chongxuan, Xu, Zhu & Zhang (2017); Tran, Pham, Carneiro, Palmer & Reid address this issue by training the classifier jointly with the other elements of the GAN architecture in an end-to-end fashion. Recently, Diffusion Models (Ho *et al.*, 2020) have been proposed as alternative generative models and are slowly replacing GANs (Dhariwal & Nichol, 2021). Some works have already explored data augmentation using those model (Trabucco, Doherty, Gurinas & Salakhutdinov, 2023; Hataya *et al.*, 2023).

1.3.2 Bilevel Optimization

Bilevel optimization (Colson, Marcotte & Savard, 2007) is an optimization framework involving the resolution of two nested optimization problems. In this framework, one optimization problem (the upper-level problem) includes another optimization problem (the lower-level problem) either as a constraint or as part of its objective function. A general bilevel optimization problem can be formulated as:

$$\begin{aligned} \min_{x,y} \quad & f(x, y^*(x)) \\ \text{s.t.} \quad & y^*(x) = \arg \min_y g(x, y) \end{aligned} \tag{1.4}$$

where f is the upper-level and g the inner-level function. Solving this optimization problem means minimizing f with respect to x , where y^* is obtained by solving the lower-level minimization problem. The interplay between the upper and lower-level problems makes bilevel optimization challenging. Typically, the solutions of the lower-level problem are not explicit functions of the variables in the upper-level problem, making traditional optimization techniques not applicable. As a result, specialized algorithms and techniques are often required to effectively solve bilevel optimization problems.

Bilevel optimization in the context of deep learning is an active area of research (Chen, Chen, Ma, Liu & Liu, 2022; Liu, Gao, Zhang, Meng & Lin, 2021a) and has many applications:

Hyperparameter Tuning In deep learning, the performance of models is heavily reliant on the choice of hyperparameters. Bilevel optimization provides an effective framework for automating this process. Here, the upper-level problem is the performance of the model on a validation set, and the lower-level problem is the training of the model on a training set. A seminal work is Bengio (2000), which uses the implicit function theorem to perform the bilevel optimization. Subsequently, Domke (2012) is the first work to propose a gradient-based approach.

Meta-learning Bilevel optimization is a natural fit for meta-learning, or "learning to learn.", which presents some similarities with hyperparameter optimization as shown in Franceschi, Frasconi, Salzo, Grazzi & Pontil (2018). The outer level can be responsible for learning a

learning strategy, while the inner level performs learning under the strategy defined by the outer level. One seminal work in this category is MAML (Finn, Abbeel & Levine, 2017).

Model Selection Bilevel optimization can be used for model selection by framing the selection of an architecture as an upper-level problem, with the lower-level problem being the training of a model given a particular architecture. A seminal work is DARTS (Liu, Simonyan & Yang, 2019).

Adversarial Training Bilevel optimization can provide a theoretical underpinning for adversarial training, where the inner problem involves minimizing the training loss, and the outer problem involves maximizing the adversarial loss. This concept is well illustrated in GAN architectures (Goodfellow *et al.*, 2014).

One interesting thing to note is the difference between joint learning and bilevel optimization. Joint learning refers to training multiple tasks simultaneously using a single model, where the parameters are optimized to minimize a combined loss function that includes contributions from all tasks. In contrast, bilevel optimization involves solving two nested optimization problems. Bilevel optimization is typically used in scenarios where one optimization problem is embedded within another, leading to a hierarchical structure. Joint learning, on the other hand, focuses on optimizing multiple tasks concurrently, sharing information and parameters among them within a single unified framework.

1.3.3 Domain adaptation

Despite their remarkable capabilities, Deep learning models often face the issue of domain shift or distributional shift. Such models, trained on a specific domain (or source domain), often lack generalization capacity when applied to a different but related domain (or target domain) and tend to perform poorly. In this thesis, we consider domain adaptation as a way to cope with this issue, which can be seen as a consequence of data scarcity. Indeed, gathering a large amount of labeled data in every possible domain to train a model is resource-intensive and often impracticable due to time, cost, and privacy concerns. Domain adaptation techniques provide a

remedy by enabling the models to leverage the knowledge learned in one domain and apply it to another. Domain adaptation is crucial in scenarios where it is not feasible or possible to capture all potential variations during training, for example in biomedical imaging (Guan & Liu, 2021). It allows models to handle such variations, enhancing their performance and versatility. Domain adaptation also extends the lifespan and utility of models by allowing them to adapt to evolving circumstances. For example, models can adapt to changes in data over time (Yao, Choi, Lee, Koh & Finn, 2022).

Domain adaptation and associated concepts have been an active area of research and has generated a large corpus of work (Csurka, 2017; Patel, Gopalan, Li & Chellappa, 2015; Wang, Lan, Liu, Ouyang & Qin, 2023; Wilson & Cook, 2020).

Domain adaptation methods initially required access to the target domain during training (Csurka, 2017). Unsupervised domain adaptation (UDA) (Pan, Tsang, Kwok & Yang, 2011; Patel *et al.*, 2015) relaxed this by not needing target domain labels (Wilson & Cook, 2020). In UDA, the model is trained on a labeled source domain and adapted to an unlabeled target domain, aiming to minimize the distribution discrepancy between the two domains. Common strategies include learning domain-invariant features (Kang, Jiang, Yang & Hauptmann, 2019; Long, Cao, Wang & Jordan, 2015; Sun & Saenko, 2016), using a “domain discriminator” (Ganin & Lempitsky, 2015; Purushotham, Carvalho, Nilanon & Liu, 2017), adversarial training (Ganin & Lempitsky, 2015) or domain alignment directly in the input space (Hoffman *et al.*, 2018). More recently, contrastive learning has also been explored in the context of UDA in Shen *et al.* (2022). However, Despite their interesting performance, these methods still require access to both source and target domains during training.

Test Time Adaptation (TTA) (Liang, Hu & Feng, 2020; Iwasawa & Matsuo, 2021), which is also referred to as Source-free domain adaptation (SFDA) in some works (Chidlovskii, Clinchant & Csurka, 2016; Liang *et al.*, 2020; Yang, Wang, van de Weijer, Herranz & Jui, 2021a; Kundu, Venkat, M V & Babu, 2020; Yang, Wang, van de Weijer, Herranz & Jui, 2021b; Boudiaf, Denton, van Merriënboer, Dumoulin & Triantafillou, 2023), allows adaptation without using

training data during the process. TTA scenarios assume access to a pretrained model and aims at leveraging unlabeled test instances from a (shifted) target distribution to make better predictions.

There are many possible taxonomies of TTA methods (Liang, He & Tan, 2023). In this thesis chapter, we limit our scope to the distinction between Offline and Online TTA methods.

Offline TTA The particularity of Offline TTA methods is that the model adaptation is done using test data already available, and the evaluation on a held-out subset of the test data. In this category belong conventional UDA methods (Saito, Watanabe, Ushiku & Harada, 2018; Zhang, Levine & Finn, 2022), UDA methods using generative modeling (Li, Jiao, Cao, Wong & Wu, 2020; Kundu *et al.*, 2020; Yeh, Yang, Yuen & Harada, 2021; Kurmi, Subramanian & Namboodiri, 2021; Qiu *et al.*, 2021) to align features without access to source data, or methods using information maximisation (Liang *et al.*, 2020). Niu *et al.* (2022) also consider conventional Continual Learning methods, for example Kirkpatrick *et al.* (2016); Li & Hoiem (2016), as Offline TTA methods.

Online TTA Online TTA is of particular interest for online applications, in which the model receives samples as a stream, which can be seen as a special case of data scarcity. Online TTA shares important motivations and similarities with concurrent settings mentioned earlier, like SFDA. In SFDA, methods also leverage samples from the target distribution of interest and have no access to source data, but the evaluation is still done on held-out test data. In other words, Online TTA is the transductive counterpart of SFDA. Operational requirements for online applications break crucial properties of the vanilla TTA setting, e.g. large batch size or class balance. Under such operational requirements, standard TTA methods degrade, underperforming the non-adapted baseline and even degenerating to random performance in some cases (Boudiaf, Mueller, Ayed & Bertinetto, 2022; Niu *et al.*, 2023). Proposed Online TTA methods usually employ one or a combination of the following techniques: self-training to reinforce the model’s own predictions through entropy minimization (Wang, Shelhamer, Liu, Olshausen & Darrell, 2021a) or Pseudo-Labeling schemes (Lee, 2013), manifold regularization to enforce smoother decision boundaries through data augmentation (Zhang *et al.*, 2022) or clustering (Boudiaf *et al.*,

2022), feature alignment to mitigate covariate shift by batch norm statistic adaptation (Li, Wang, Shi, Liu & Hou, 2017; Schneider *et al.*, 2020; Nado *et al.*, 2020; Lim, Kim, Choo & Choi, 2023; Zhao, Chen & Xia, 2023), sample re-weighting (Zhao *et al.*, 2023; Niu *et al.*, 2022) and meta-learning methods (Goyal, Sun, Raghunathan & Kolter, 2022) that try to meta-learn the best adaptation loss. Some methods also adopt continual learning constraints and consider the accuracy on previously seen domain during the adaptation process (Niu *et al.*, 2022; Wang, Fink, Van Gool & Dai, 2022). Test-time Training(TTT) methods (Sun *et al.*, 2020; Osowiechi *et al.*, 2022) are a particular case of Online TTA involving test-time adaptation via self-supervision. TTT works by constructing an auxiliary task that can be solved both at training and adaptation time and therefore, requires an adhoc training procedure.

CHAPTER 2

ADVERSARIAL LEARNING OF GENERAL TRANSFORMATIONS FOR DATA AUGMENTATION

In this chapter, we begin our exploration of data augmentation as a strategy for improving the performance of visual recognition models. The work presented leverages the capabilities of generative models, specifically generative adversarial models (GANs), to produce new images that improve the performance of an image classifier. The choice of this architecture type is based on the intuitive understanding that generative models have the power to learn more complex variations within an existing dataset, thus alleviating the limitations of traditional data augmentation methods relying on simple transformations chosen heuristically.

2.1 Introduction

Convolutional neural networks have shown impressive results in visual recognition tasks (Hu, Shen & Sun, 2018). However, for a proper training and good performance, they require large labeled datasets (Mahajan *et al.*, 2018; Sun *et al.*, 2017). If the amount of training data is small, regularization techniques (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014; Krogh & Hertz, 1992) can help the model avoid overfitting. Among these techniques, data augmentation is one of the most effective in improving the final performance of the network (Hernández-García & König, 2018a; Perez & Wang, 2017).

In image processing, data augmentation consists in applying predefined transformations such as geometric transformations like flip or rotations and color changes (Krizhevsky *et al.*, 2012; Ciresan *et al.*, 2012) to samples at disposal. This approach works quite well and provides a consistent improvement of the accuracy when training a classifier. However, selecting the right transformations require prior knowledge and chosen transformations are dataset dependent. For instance, while horizontal flipping is appropriate for natural images, it produces ambiguities (e.g. 2 and 5) on number datasets.

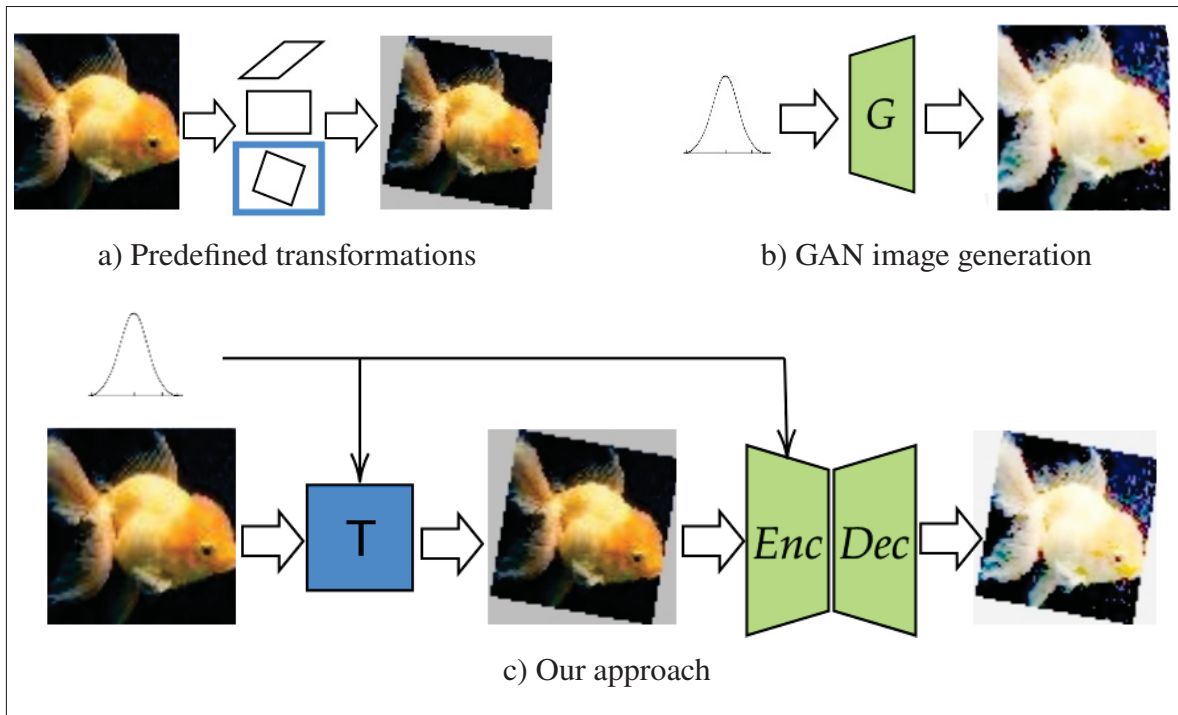


Figure 2.1 Automatic data augmentation approaches

(a) Sequence of predefined transformations are automatically selected. (b) GAN model generates new images from the same distribution. (c) Our approach combines the advantages of these two methods by combining affine transformations generated by a spatial transformer network and a transformation generated by a convolutional encoder decoder model.

Several studies have investigated automatic data augmentation learning, to avoid the manual selection of transformations. Ratner, Ehrenberg, Hussain, Dunnmon & Ré (2017) define a large set of transformations and learn how to combine them (Fig.2.1(a)). This approach yields good results, but is based on predefined transformations, which prevents the learning of other transformations that could be useful for the classifier. Alternatively, Chongxuan *et al.* (2017) and Tran *et al.* generate new samples via a generative adversarial networks (GAN) based model from the probability distribution of the data $p(X)$ (Fig.2.1(b)). Those methods show their limit when the number of training samples is low, as training a good generative model with a reduced training dataset is challenging. Hauberg, Freifeld, Larsen, Fisher & Hansen (2016) learn the natural transformations existing in a dataset by aligning pairs of samples from the same

class. This approach is efficient on easy datasets like MNIST but seems not applicable on more complex datasets.

Our work combines the advantages of generative models and transformations learning approaches in a single end-to-end network architecture. Firstly, instead of learning to generate samples, our model learns to generate transformations of a given sample. In other words, instead of generating samples from $p(X)$, we learn to generate samples from $p(\hat{X}|X)$, with X a training data point, which is easier, especially when the training data is reduced. As shown in Fig. 2.1(c), we propose an approach that combines a first transformation defined by an affine matrix with a transformation defined by a convolutional encoder-decoder architecture. In practice, we find that the affine transformation learns global image transformations, while the encoder-decoder architecture learns more localized transformations such as local image distortions and color changes. Thus, the combination of the two leads to a general distribution of transformation that can be applied to any image-based training data and is not specific to a given domain or application.

Secondly, affine transformations are learned by an adaptation of spatial transformer networks (STN) (Jaderberg *et al.*, 2015), so that the entire architecture is differentiable and can be learned with standard backpropagation. The original purpose of STN is to learn to transform the input data so that it becomes invariant to certain transformations. In contrast, our approach uses STN to generate a distribution of augmented samples in an adversarial way. We experimentally show that our approach is more effective in improving the classifier accuracy. Finally, we show that for optimal performance, it is important to jointly train the generator of the augmented samples with the classifier in an end-to-end fashion. By doing that, we can also add an adversarial loss between the generator and classifier such that the generated samples are difficult, or adversarial, for the classifier. This further increases the final classifier accuracy.

We tested our approach on MNIST, fashion MNIST, SVHN and CIFAR-10 datasets, both in full dataset and low-data regime. Our empirical results show that (i) each component of the network

is important for optimal performance; and that (ii) for a given classifier architecture, our method outperforms the hand-defined data augmentation and most of the previous methods.

2.1.1 Contributions

To summarize, the contributions of this work are:

- We propose a data augmentation network that is fully differentiable, trainable end-to-end, and can significantly improve the performance of any image-based classifier;
- We devise STN in an adversarial way that together with an encoder-decoder architecture is able to learn a distribution of general transformations for augmenting the training data;
- We experimentally show that, for data augmentation, learning image transformations is better than generating images from scratch and that learning data augmentation and classification jointly is more effective than in two separate tasks.

2.2 Related Work

2.2.1 Standard Data Augmentation

Data augmentation is an efficient regularizer for improving the performance of visual recognition methods (Hernández-García & König, 2018b), especially when dealing with small training sets prone to overfitting (Wagner, Thom, Schweiger, Palm & Rothermel, 2013). Data augmentation is based on specific domain knowledge about data transformations useful for an end task, while keeping the semantic meaning of the data. For natural image classification, the standard form of data augmentation is affine transformations like flip, rotation or color changes (Perez & Wang, 2017). However, more complex transformations such as occluding parts of an image (DeVries & Taylor, 2017b) or blending two or more images (Zhang *et al.*, 2018; Lemley *et al.*, 2017) were also proposed. Finally, DeVries & Taylor (2017a) uses a form of data augmentation by adding noise, interpolating, or extrapolating between samples in the feature space, instead of in the input space. Adding the transformed samples in the training data can highly improve

the end task performance, but, especially with new tasks and datasets, it is not clear which transformations are helpful or, on the opposite, harmful.

2.2.2 Model-based Transformations

Recently, some approaches tried to make the data augmentation automatic, to avoid the manual selection of transformations. Ratner *et al.* (2017) propose to learn a sequence of predefined transformations to generate new samples. By using a generative adversarial network (GAN), the generated samples are enforced to be close to the training data distribution. This approach works well, but is limited to a set of predefined transformations, which prevents other possibly helpful transformations to be learned. Additionally, as transformations might not be differentiable, optimization is performed with a reinforcement learning approach. This can make the training more difficult and slow. Sixt, Wild & Landgraf (2018) and Shrivastava *et al.* (2017) generate augmented samples from initial 3D models of the data and refine them with GANs. These approaches work quite well, but need a strong prior knowledge of how to generate the initial 3D view. We follow a similar approach of first generating a global transformation and then refining it, but without using any domain specific 3D model. Instead, we learn a transformation from the given samples with a spatial transformer network Jaderberg *et al.* (2015). This is more difficult because the transformations are learned, but more generic and applicable to any dataset. Hauberg *et al.* (2016) learn class specific transformations by considering pairs of samples within a class and learning the distribution of the transformations, morphing one element of the pair to the other. This approach seems applicable only to simple datasets like MNIST. In the context of medical imaging, Zhao, Balakrishnan, Durand, Guttag & Dalca (2019) learn distributions of spatial and appearance transformations by aligning labeled and unlabeled samples to create new synthesized labeled samples. Those new labeled samples are used to improve the performance of an image segmentation models for brain MRIs. Finally, Peng, Tang, Yang, Feris & Metaxas (2018) propose to train jointly a data augmenter (again based on pre-defined transformations) and an end-task network for human pose estimation.

2.2.3 Adversarial Training

Goodfellow, Shlens & Szegedy (2015) showed how to induce a trained neural network to perform a wrong classification by minimally changing the image. They compute the gradient of the loss of the network with respect to the image pixels and use it to modify only the most influential pixels. This results in new images (adversarial) that are almost indistinguishable for humans, but wrongly classified by the net. Miyato, Maeda, Koyama & Ishii (2018) extended the approach to unlabeled samples. The adversarial images can be added to the training data to improve the robustness to adversarial examples. This also improves the classifier accuracy and can be seen as a form of data augmentation. However, this method does not fully exploit the data augmentation power as the learned transformations are constrained to small changes in order to maintain unaltered the appearance of the image.

2.2.4 Generic Transformations with GANs

Multiple approaches use generative models to generate the augmented samples at pixel level with a convolutional encoder-decoder architecture. In theory, this is more powerful and flexible than defining a set of predefined transformations, provided that the dataset is large enough to learn the generative model. For instance, Mirza & Osindero (2014) and Odena *et al.* (2017) proposed to generate images conditioned on their class, which could be directly used to augment a dataset. CatGAN (Springenberg, 2016) on the other hand, performs unsupervised and semi-supervised learning as a regularized information maximization problem Krause, Perona & Gomes (2010) with a regularization based on the generated samples. Also based on GAN, but directly used for data augmentation, is the model proposed by Antoniou *et al.* (2018). In this case, the authors condition directly on a given image. From our experiments, this method seems to produce suboptimal results because the generation is performed independently of the classification. Salimans *et al.* (2016) train a discriminator coupled to a classifier by adding an additional class to the classifier for generated images. In this case, the unlabeled data generated by the generator can be seen as a form of data augmentation. Zhang, Wang, Liu & Ling (2019) extend this idea to low data regime by using a finer grain for the classifier. Instead of using $K+1$ classes as in

Salimans *et al.* (2016), the classifier in this model uses in 2K classes, K classes for real data and K classes for generated data. Triple GAN Chongxuan *et al.* (2017) and Bayesian Data Augmentation Tran *et al.* train a classifier jointly with the generator. We follow the same strategy. However, these models are based on the direct generation of samples from noise, and, as we show in our experiments, this seems more difficult than transforming a given image (as we do), especially when the training data is reduced. Notice that most of the presented GAN models are designed for semi-supervised learning. Instead, our aim is to train with a reduced dataset, without additional (non annotated) images. This is a more challenging task, and therefore a direct comparison is difficult and out of the scope of this work.

2.3 Proposed Model

In this work, we aim to improve the performance of an image classifier by augmenting the training dataset with samples synthesized by transforming the initial dataset with learned transformations.

Our goal is to learn a distribution of image transformations \mathcal{T} , so that given an input image x_i , $\mathcal{T}(x_i)$ represents all image transformations such that the semantic meaning of the image, i.e. its class y_i , is preserved. We expect this distribution to be the optimal set of transformations to augment the training data of a given classifier C .

To learn this distribution, we propose the GAN based architecture shown in Fig. 2.2(a). This architecture involves four modules. A generator G transforms the input images and is supported by two discriminators, imposing constraints on the generated samples. The first one, D^C , is the class discriminator and ensures that the generated sample stays in the same class as the input sample. The second one, D^D , is the dissimilarity discriminator, and ensures that the transformed sample is different from the input sample. This is necessary to prevent the generator from learning the identity transformation, which would not help the classifier. Finally, a classifier C performs the final classification task. Training our model consists in finding the equilibrium in a multiple two-player game. Indeed, we solve jointly the adversarial game between G and D^C , G and D^D and finally between G and C . In contrast to previous GAN architectures also

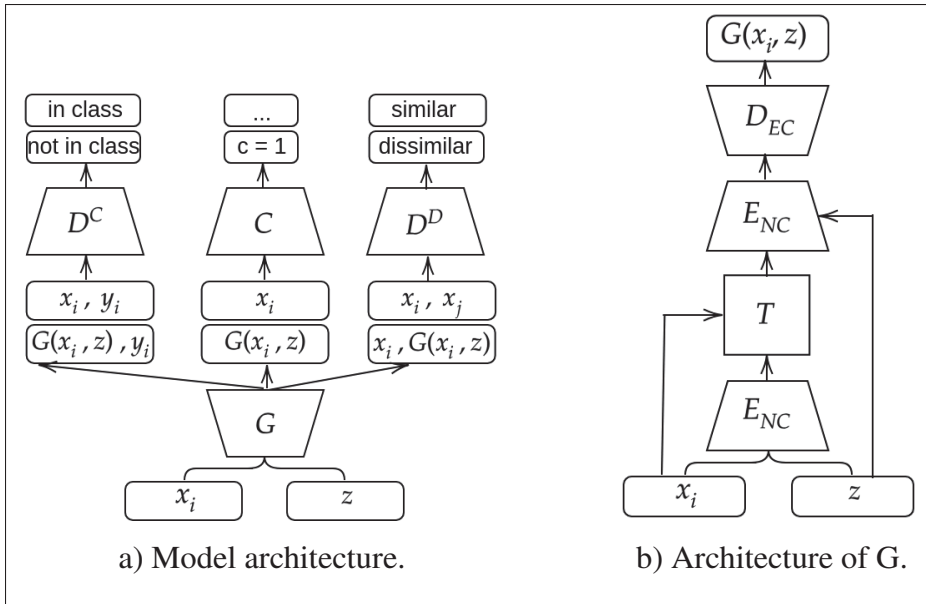


Figure 2.2 Proposed model

(a) A classifier C receives augmented images from a generator G challenged by two discriminators D^C and D^D . The class discriminator D^C ensures that the generated image $G(x_i, z)$ belongs to the same class as the input image x_i with label y_i . The dissimilarity discriminator D^D ensures that the transformed sample $G(x_i, z)$ is dissimilar from the input sample x_i but similar to a sample x_j from the same class. (b) Given an input image x_i and a noise vector z , our generator first performs a global transformation using a spatial transformer network, followed by more localized transformations using a convolutional encoder-decoder network.

including a classifier (e.g. Triple GAN Chongxuan *et al.* (2017) and Bayesian DA Tran *et al.*), we introduce an additional loss pushing the generator to produce images that are difficult to classify, and this also helps to improve the classifier.

In the following paragraphs, we formally describe in detail each part of the model.

2.3.1 Generation of augmented samples

The role of the generator G , is to learn the distribution \mathcal{T} of the transformations of input images that are the most useful to train the classifier C . In our intuition, learning an image transformation

instead of learning a mapping from noise to image to generate new samples (as in previous work), is an easier task in low data regime.

As shown in Fig. 2.2(b) the generator is composed of two elements: a Spatial Transformer Network (STN) that learns global affine transformations and a U-Net (Ronneberger, Fischer & Brox (2015)) variant that can learn, in principle, any other transformation. While in the original paper the spatial transformer module was used for removing invariances from the input data, the proposed model generates transformed samples (controlled by z) in an adversarial way.

The entire transformations of an input image x_i and a random noise vector z can be formulated as:

$$G(x_i, z) = D_{EC}(E_{NC}(T(x_i, E_{NC}(x_i, z)), z)). \quad (2.1)$$

The input image x_i and noise z_i are encoded into a vectorial representation through E_{NC} . This representation is then passed to the spatial transformer network T , which generates an affine transformation and applies it to transform x_i . Finally, the transformed image is passed to a U-Net, composed of a convolutional encoder E_{NC} and decoder D_{EC} .

The loss function of the generator can be formulated as the weighted sum of three terms:

$$\begin{aligned} \mathcal{L}_G = & -\alpha \mathbb{E}_{x_i, y_i \sim p_{data}, z \sim p_z} [\log (D^C(G(x_i, z), y_i))] \\ & -\beta \mathbb{E}_{x_i \sim p_{data}, z \sim p_z} [\log (D^D(x_i, G(x_i, z)))] \\ & -\gamma \mathbb{E}_{x_i, y_i \sim p_{data}} [\log (1 - C_{y_i}(G(x_i, z)))] , \end{aligned} \quad (2.2)$$

in which D^C , D^D are respectively the class and dissimilarity discriminator. Their role is to enforce constraints on the transformed image. More details will be given in the following paragraphs. C_{y_i} is the softmax output of the classification network for the class y_i , i.e. the probability that the given image belongs to the class y_i . Finally, α , β and γ are hyperparameters introduced to balance the three loss terms and stabilize the training of the model.

The first term of the loss function increases the probability $D^C(G(x_i, z), y_i)$ that a transformed sample $G(x_i, z)$ belongs to the same class y_i as the original sample. The second term increases

the probability $D^D(x_i, G(x_i, z))$ that the original sample x_i and the transformed sample $G(x_i, z)$ are different. Finally, the third term reduces the probability of a correct classification C_{y_i} of the transformed sample $G(x_i, z)$. This loss pushes the generator to produce images that are difficult to classify and, if properly balanced, helps to improve the classifier (see Sec. 2.5.4 Adversarial loss).

2.3.2 Constraints on transformations

To enforce transformations that do not alter the class of an image, two discriminators support the generator. The first one, the class discriminator D^C , ensures that the generated image belongs to the same class as the original image, whereas the second, the dissimilarity discriminator D^D , ensures that the generated sample is different from the original sample. The motivation behind this design is that we want to create new samples that are as different as possible from the original samples but still in the same class distribution.

The first discriminator, D^C , receives as input an image (either a real image x_i or a transformed image $G(x_i, z)$) and a class label y_i and outputs the probability of the image to belong to that class. Its loss function can be formulated as:

$$\begin{aligned} \mathcal{L}_{D^C} = & -\mathbb{E}_{x_i, y_i \sim p_{data}} [\log(D^C(x_i, y_i))] \\ & -\mathbb{E}_{x_i, y_i \sim p_{data}, z \sim p_z} [\log(1 - D^C(G(x_i, z), y_i))]. \end{aligned} \quad (2.3)$$

The first term increases the probability $D^C(x_i, y_i)$ that a real sample x_i belongs to class y_i , whereas the second term reduces the probability $D^C(G(x_i, z), y_i)$ that a generated sample $G(x_i, z)$ belongs to the same class y_i . In this way, the discriminator learns to distinguish between real and generated samples of a certain class.

The second discriminator, D^D , takes a pair of samples as input (either two different samples of the same class x_i, x_j or a sample x_i and its transformation $G(x_i, z)$), and outputs a dissimilarity

probability between the two samples. Its loss function can be formulated as:

$$\begin{aligned} \mathcal{L}_{D^D} = & -\mathbb{E}_{x_i, x_j \sim p_{data}} [\log (D^D(x_i, x_j))] \\ & -\mathbb{E}_{x_i \sim p_{data}, z \sim p_z} [\log (1 - D^D(x_i, G(x_i, z)))] \end{aligned} \quad (2.4)$$

The first term of the loss function increases the probability $D^D(x_i, x_j)$ of a sample x_i and another sample from the same class x_j belonging to the same class, whereas the second term reduces the probability $D^D(x_i, G(x_i, z))$ of the same sample x_i and the corresponding transformed sample $G(x_i, z)$ to be part of the same class. In this way this discriminator enforces that the transformed sample $G(x_i, z)$ belong to the same class as x_i while being different (as we never use $x_j = x_i$). Thus, it enforces dissimilarity between the transformed sample and the original one.

2.3.3 Classification.

The image classifier C is trained jointly with the generator and the two discriminators. C receives as input real samples x_i as well as augmented samples, i.e. samples transformed by G . Its loss function can be formulated as:

$$\begin{aligned} \mathcal{L}_C = & -\mathbb{E}_{x_i, y_i \sim p_{data}} [\log (C_{y_i}(x_i))] \\ & -\mathbb{E}_{x_i, y_i \sim p_{data}, z \sim p_z} [\log (C_{y_i}(G(x_i, z)))] \end{aligned} \quad (2.5)$$

This loss is a classical cross-entropy loss that enforces the classifier to give higher probability to the correct class in case of real x_i or augmented samples $G(x_i, z)$.

Global Loss. Training our model consists in finding the equilibrium in a multiple two-player game. Indeed, we solve jointly the adversarial game between G and D^C , G and D^D and finally between G and C . The global loss can be formulated as:

$$\mathcal{L} = \mathcal{L}_G + \mathcal{L}_{D^C} + \mathcal{L}_{D^D} + \mathcal{L}_C \quad (2.6)$$

During optimization, we sequentially minimize a mini-batch of each loss. Notice that \mathcal{L}_G tries to maximize D^C of the transformed samples $G(x_i, z)$, while \mathcal{L}_{D^C} tries to maximize $1 - D^C$, which

corresponds to minimize D^C . The same also for D^D and C . This is not a problem, in fact it shows that the defined loss is adversarial, in the sense that generator and discriminator/classifier “fight” to push the losses in different directions. This mechanism generates augmented samples that help the training of the classifier, i.e. samples that belongs to the right class but are close to the decision boundaries.

2.4 Experimental setup

2.4.1 Datasets

In our experiments, we validated our model on the four following datasets:

MNIST (Lecun *et al.*, 1998) is a dataset of handwritten grayscale digits. The full dataset contains 70,000 samples, 60,000 training samples and 10,000 test samples. In our experiments with reduced dataset, we use a subset of 550 samples as in Ratner *et al.* (2017). Fashion-MNIST (Xiao, Rasul & Vollgraf, 2017) is a dataset of grayscale fashion articles. The full dataset is composed of 70,000 samples, 60,000 training samples and 10,000 test samples. Similarly to MNIST, we use a subset of 550 samples in our experiments with reduced dataset. Street View House Numbers (SVHN) (Netzer *et al.*, 2011) is a dataset of 32×32 pixels images of real world color photos of house numbers. It is composed of 73257 training samples and 26032 test samples. In our experiments with reduced dataset, we use a subset of 1000 training samples. CIFAR10 (Krizhevsky *et al.*, 2009) is a dataset of 10 classes color natural images of size 32×32 . The full dataset is composed of 60,000 images, 50,000 training images and 10,000 training images. In our experiments with reduced dataset, we use a subset of 4000 training samples as in Ratner *et al.* (2017).

We did not experiment on datasets with higher resolution images due to the known difficulties of standard GAN models to generate good quality high resolution images. We leave experiments with more advanced GAN optimization and models to a future work.

2.4.2 Implementation Details

In all our experiments, we standardize images as basic pre-processing, which consists in subtracting the mean pixel value, and then dividing by the pixel standard deviation.

2.4.3 Model architecture

The generator network architecture is described in Appx. I-3. It takes as input an image and a Gaussian noise vector (100 dimensions), which are concatenated in the first layer of the network. The three parameters α , β and γ of the generator loss are estimated on a validation set. For the class discriminator D^C , we use the same architecture as Dai, Yang, Yang, Cohen & Salakhutdinov (2017) and described in Appx. I-1. The network is adapted to take as input an image and a label (as a one hot vector). These are concatenated and given as input to the first layer of the architecture. For the dissimilarity discriminator D^D , we use a similar architecture described in Appx. I-2. The network is adapted to take as input a pair of images, which are concatenated in the first layer of the architecture. For the classifier, we use the architecture used in Dai *et al.* (2017) and described in Appx. I-4. We use Adam as optimizer.

2.5 Results

We present several experiments to better understand our model and compare it with the State-of-the-art in automatic data augmentation. In a first set of experiments, we compare the performance of learning data transformations to standard data augmentation for different sizes of the training datasets. In a second one, we compare the performance of our model to other State-of-the-art models. In a third one, we show the importance of learning jointly the data augmentation and the classifier, and finally, we analyze the contribution of each model component to the classifier accuracy.

2.5.1 Comparison with Standard Data Augmentation

In this series of experiments, we compare the data augmentation learned by our model to standard pre-defined data augmentation. In order to do this, we define two different levels of data augmentation. *Light DA* refers to random padding of 4 pixels on each side of the image, followed by a crop back to the original image dimensions. *Strong DA* includes the same augmentation as *light DA* but also rotation in range $[-10, 10]$ degrees, scaling, with factor in range $[0.5, 2]$. For CIFAR10, *strong DA* also includes a horizontal image flip.

In a first experiment, we compare the accuracy of the baseline classifier without any data augmentation (*Baseline*), the baseline with two levels of pre-defined data augmentation (*Baseline + light DA* or *Baseline + Strong DA*), and our data augmentation model (*Our Model*) while increasing the number of training samples. For very few samples (1000) the predefined data augmentation is still better than our approach. When the generation of the samples is learned with a dataset too small, the generator produces poor samples that are not helpful for the classifier. When the number of samples increases, our approach obtains a much better accuracy than the approach with standard data augmentation. For instance, at 4000 training samples, the baseline obtains an accuracy of 66%, the predefined data augmentation obtains an accuracy of 76% and our model reaches an accuracy of 80.5%, thus a net gain of 14 points compared with the baseline and 4 points compared to the data augmented model. If we add more examples, the gap between our learned data augmentation and the standard data augmentation tends to reduce. With the full dataset we reach about a half a point more than the standard data augmentation.

In a second experiment, we compare different types of data augmentation on four datasets with a reduced number of samples. As shown in Tab. 2.1, *our best model* is always performing better than *light DA* and *strong DA*. This means that our data augmentation model learns transformations that are more useful for the final classifier. Notice that on FMNIST, *light DA* decreases performance of the final classifier. This suggests that data augmentation is dataset dependent, and transformations producing useful new samples in some domains might not be usable in others.

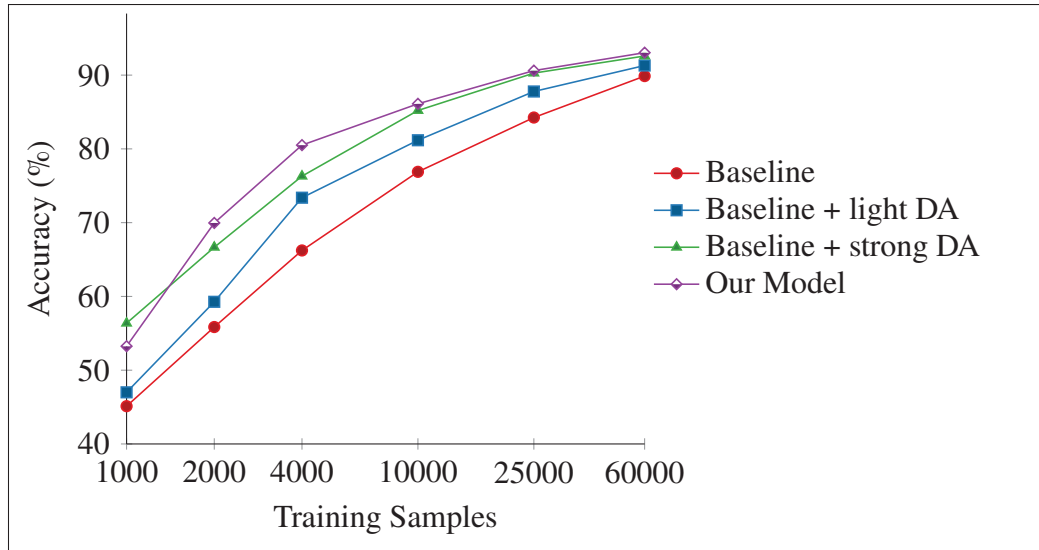


Figure 2.3 Classification Accuracy (%) vs number of training samples on CIFAR10

Our method is effective when the number of samples is reduced. However, for too few samples, normal data augmentation it is still slightly better.

Table 2.1 Comparison of classification accuracy (%) with DA on different datasets

In low data regime, our model performs better than *light DA* and *strong DA* on the four considered datasets. With training data, we expect a saturation of the improvement similar to Fig. 2.3 for CIFAR10.

Method	MNIST 550	FMNIST 550	SVHN 1000	CIFAR10 4000
Baseline	90.81	79.02	79.55	66.73
Baseline + light DA	97.55	78.96	84.48	74.76
Baseline + strong DA	98.50	80.37	84.33	77.74
Our model	98.61	82.43	86.07	80.5

2.5.2 Comparison with State of the Art

In Tab. 2.2 we compare our method with other approaches for automatic data augmentation. Compared with TANDA (Ratner *et al.*, 2017), our method yields slightly lower accuracies. However, TANDA is based on the selection of predefined transformations. This means that its

Table 2.2 Comparison of classification accuracy (%) to other automatic DA Methods

We compare the accuracy of our model with other methods performing automatic data augmentation on MNIST and CIFAR10. Notice that we use only labelled samples for the training, therefore the task is harder than in semi-supervised learning.

Method	Model	MNIST 550	CIFAR10 4000	CIFAR10 Full
Baseline	ConvNet	90.81	66.23	89.88
Bayesian DA Tran <i>et al.</i>	ResNet18	-	-	91.0
DADA Zhang <i>et al.</i> (2019)	ResNet56	-	79.3	-
TANDA Ratner <i>et al.</i> (2017)(MF)	ResNet56	96.5	79.5	94.4
TANDA Ratner <i>et al.</i> (2017)(LSTM)	ResNet56	96.7	81.5	94.0
Our model	ConvNet	96.0	80.50	93.0

learning is reduced to a set of manually selected transformation, which facilitates the task. Also, TANDA uses an additional standard data augmentation based on image crop, while our method does not need any additional data augmentation. On the other hand, our method compares favorably to Bayesian DA Tran *et al.* and DADA Zhang *et al.* (2019), both based on GAN models with a larger neural network as classifier. This shows that our combination of global and local transformations helps to improve the final performance of the model. Notice that our approach considers only fully supervised data, thus a direct comparison with semi-supervised methods, such as Chongxuan *et al.* (2017); Springenberg (2016); Miyato *et al.* (2018), that make use of unlabeled data, would not be fair.

2.5.3 Joint Training

In this experiment, we compare the performance of our method, in case of joint and separate training. In joint training, the augmented images generator and the classifier are trained simultaneously end-to-end, as explained in our method. In separate training, the generator is first trained to generate augmented images, and these images are then used as data augmentation to improve the classifier. In case of separate training, we collect samples from different phases of the training: at epochs 200, 500, 700, 1000.

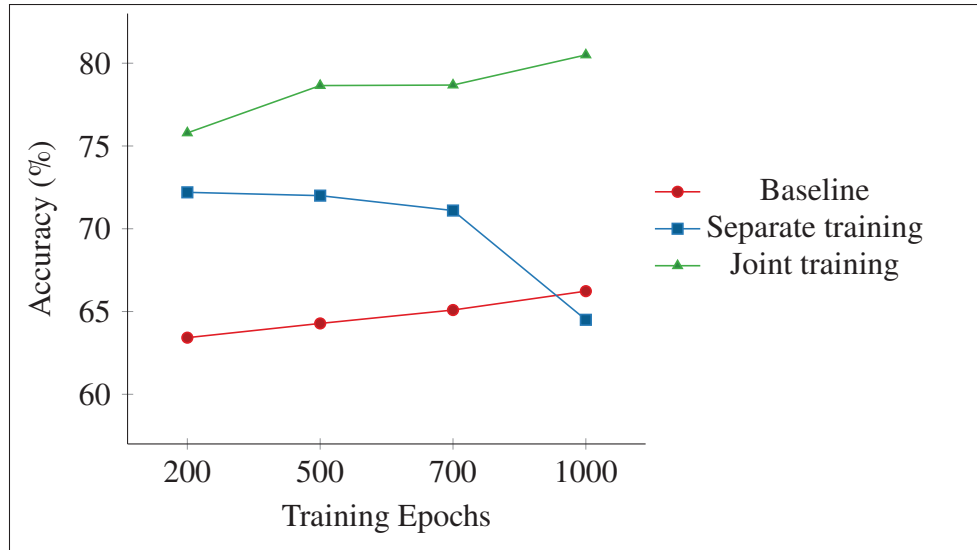


Figure 2.4 Classification Accuracy (%) over epochs on 4000 samples of CIFAR10 for a baseline classifier and our joint training. We compare them with a separate data augmentation training at 200, 500, 700 and 1000 training epochs.

On Fig. 2.4, it is interesting to notice the different behaviour of the two methods. In the early phase of training, at epochs 200, both the *Separate training* (blue line) and the *Joint training* (green line) perform above 70%, whereas the *Baseline* (red line) is much lower. However, with additional training epochs, the performance of *Separate training* decreases, while *baseline* and *joint training* accuracies increase. From this experiment, it seems clear that for sample generation based on generic transformations (in contrast to predefined transformations as in Ratner *et al.* (2017)), the joint training of the generator and the classifier is important for optimal performance. This may be why Data Augmentation GAN (Antoniou *et al.*, 2018) seems to work only with a very reduced set of examples.

We believe that for good performance in data augmentation it is not just about generating plausible augmented samples, but also about generating the right samples at the right moment, as in curriculum learning Bengio, Louradour, Collobert & Weston (2009). Our understanding is that in the beginning of training, even poor generated samples can help to improve the optimization. However, towards the end of the training, only realistic samples are needed for

improved accuracy. Notice also that if we use samples generated towards the end of training, the performance of the classifier drops even below the baseline model. This is probably due to the fact that the generated samples are too complex for the classifier, and it cannot learn how to generalize on them. Also, consider that in case of separate training, it would not be possible to use the adversarial loss on the classifier G_{ADV} , which also helps to improve the final performance.

2.5.4 Ablation Study

Table 2.3 Ablation Study on CIFAR-10 with 4000 training samples.

C = CNN classifier; T = standard spatial transformer network applied to the image; D_{EC} = CNN decoder, from noise or small code to image; $E_{NC} + D_{EC}$ = combination of CNN encoder and decoder to obtain a new image; $T + E_{NC} + D_{EC}$ = adversarial use of the STN; D^C = Class Discriminator; D^D = Dissimilarity Discriminator; G_{ADV} = adversarial loss on the classifier.

Conf.	Components	Acc. (%)
(a)	C	66.73
(b)	$C + T$	64.75
(c)	$C + T + D^C + D^D$	70.62
(d)	$C + D_{EC} + D^C$	68.03
(e)	$C + E_{NC} + D_{EC} + D^C$	67.24
(f)	$C + E_{NC} + D_{EC} + D^D$	67.11
(g)	$C + E_{NC} + D_{EC} + D^C + D^D$	73.82
(h)	$C + E_{NC} + D_{EC} + T + D^C + D^D$	80.14
(l)	$C + E_{NC} + D_{EC} + T + D^C + D^D + G_{ADV}$	80.51

In Tab. 2.3 we evaluate the importance of the different components of our network on CIFAR10 with 4,000 training samples. First we evaluate the impact of each basic module used for data augmentation and then consider their combination. Notice how the classification accuracy goes from 66% accuracy of a normal classifier (a) to 80.5% accuracy for our best model (l), without using any predefined data augmentation.

STN and U-Net First, we compare the standard use of spatial transformer network $C + T$ (b) as a transformation invariant approach with our adversarial approach $C + T + D^C + D^D$ (c) to generate augmented samples. In our approach the input image, together with a noise vector, are encoded into a small representation that is passed to the spatial transformer network T , producing an affine matrix. The input image transformed with this affine matrix is then used as data augmentation for the classifier. While the standard spatial transformer network (STN) does not really help to improve results, our adversarial STN already improves the baseline accuracy by 4%. The transformations generated by our adversarial STN are simply affine transformations that can help to improve the classifier. For more general transformations, able to independently change every pixel of an image we use a convolutional encoder-decoder model $E_{NC} + D_{EC}$ (g) based on U-Net, thus improving model accuracy to 73.82%. Finally, combining both transformations (h) further improves the classification accuracy to 80.14%.

Discriminators We assess the contribution of the two discriminators, D^C and D^D used in our final model. As shown in Tab. 2.3, D^C alone (e) slightly improves performance. This is probably because using only a class discriminator does not prevent the generator to generate the identity transformation, thus hindering the classifier. When using only the dissimilarity discriminator D^D (f), the accuracy is also only slightly improved. The best performance is reached when the two discriminators work together (g), boosting the accuracy to 73.82%.

Generation vs. Transformation We compare our approach based on transforming an image with the direct generation of a new sample $D_{EC} + D^C$ (d) as in Bayesian Data Augmentation (Tran *et al.*). We see that this model is better than the basic classifier (a) but still far from the performance levels obtained by our approach (g). This result makes us believe that transforming an image is simpler than generating an image from scratch.

Adversarial loss Finally, we evaluate the effect of adding a loss that enforces the generator to be adversarial to the classifier, i.e. generate samples that are difficult for the classifier. This corresponds to the last term of \mathcal{L}_G in equ. 2.2. As shown in Tab. 2.3 (l) the contribution of this additional loss helps to further improve the final accuracy.

2.5.5 Generated Transformations

In Fig. 2.5 we show some samples (*left*) with the associated learned transformations (*right*) for the four datasets considered. On MNIST, notice how the transformed numbers sometimes seem adversarial, in the sense that the applied transformation makes them look almost like another number. Our intuition is that hard samples are close to the boundaries between two or more classes, and they are the most informative for improving the classification performance. For Fashion-MNIST it is interesting to notice that the transformation seems to reduce the variability of the dataset, transforming the objects into a set of simpler templates. Nevertheless, these transformations still help to improve the performance. On SVHN, only the digit in the center is used for classification. Interestingly, our model seems to learn that the generated samples are all zoomed to the central digit. Eventually, on CIFAR10, it is interesting to notice how the image colors and contrast are changed in a meaningful way.

2.6 Conclusion and Discussion

In this work, we have presented a new approach for improving the learning of a classifier through the automatic generation of augmented samples. The presented method learns general transformations end-to-end and is fully differentiable. In our experiments, we have shown that several elements are important to obtain the best performance. First, the generator and the classifier should be trained jointly. Second, it is important to transform the given images instead of generating samples from scratch. Finally, the combined use of global transformations with STN and local transformation with U-Net is also essential to obtain the best data augmentation. However, we have also seen that performing data augmentation using a generative adversarial network is not without challenges. Intuitively, generating general transformation without constraints might be too difficult. In practice, learning optimal transformations from a predefined set seems to work better, as presented in the next chapter.

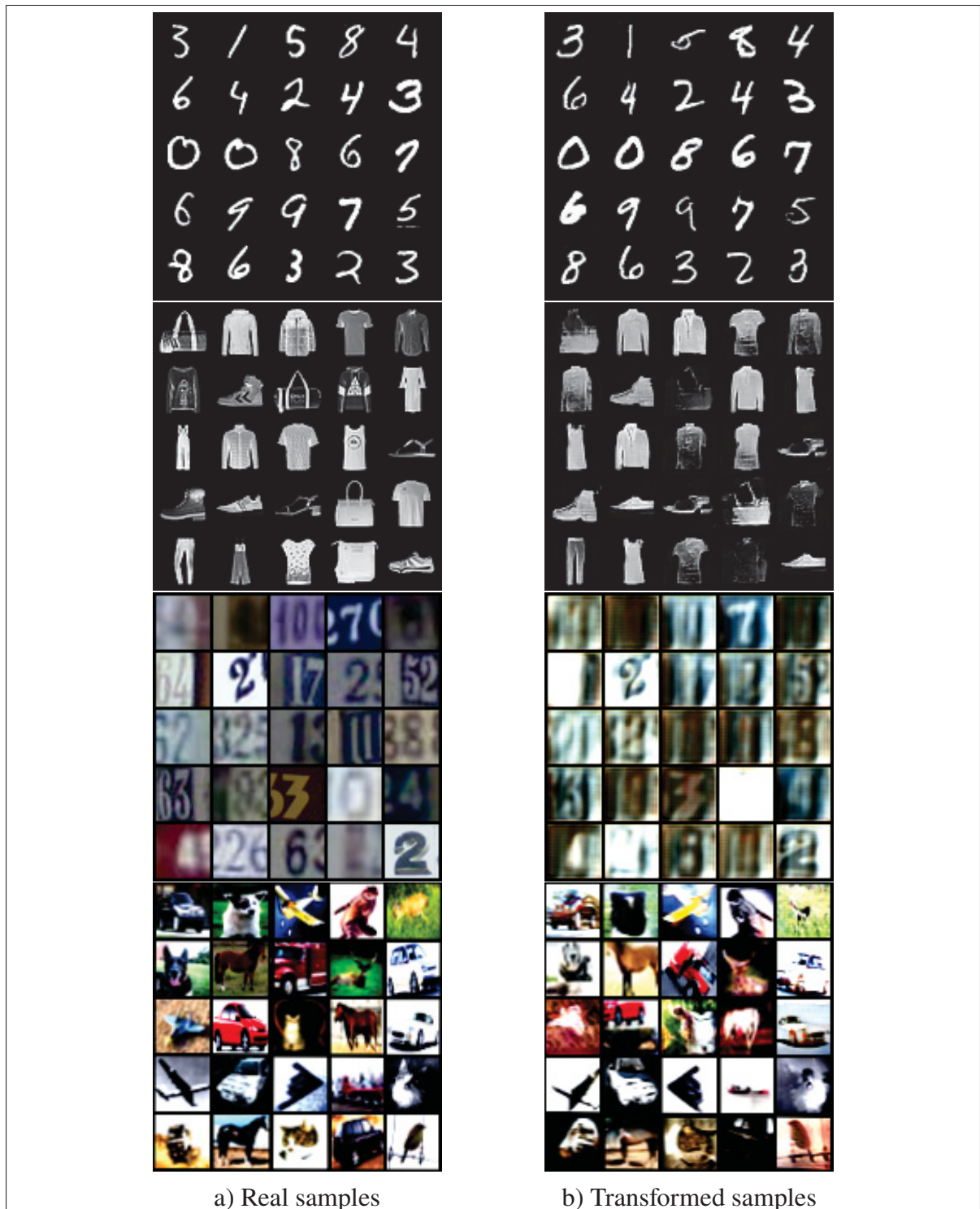


Figure 2.5 Real and transformed images from MNIST, Fashion-MNIST, SVHN and CIFAR10.

Our approach learns to apply the right transformations for each dataset. For instance on MNIST and Fashion-MNIST there is no flip, nor zoom, because not useful, while on SVHN zoom is often used and on CIFAR10, both zoom, flip and color changes are applied.

CHAPTER 3

LEARNING DATA AUGMENTATION WITH ONLINE BILEVEL OPTIMIZATION FOR IMAGE CLASSIFICATION

Having explored generative models as a way to do data augmentation for improving the performance of visual recognition models, we now examine, in this chapter, the selection of the data augmentation transformations and parameters and aim at improving this process. This choice is motivated by the understanding that, although GANs have the ability to learn more complex variations within an existing dataset, their training remains challenging, with issues like mode collapse, non-convergence or vanishing gradients as potential obstacles. Furthermore, as demonstrated in the previous chapter, learning general transformations is not a trivial task. In response, we explore the possibility of constraining the space of possible transformations, which might facilitate the learning of more useful transformations.

3.1 Introduction

Best results in deep learning methods are obtained by large models in which the number of parameters is much higher than the dimensionality of the input data, as well as the number of available samples (Simonyan & Zisserman, 2015; He *et al.*, 2015). In this setting, overfitting is a major problem (Srivastava *et al.*, 2014). Standard regularization techniques applied directly to the model parameters only add very general knowledge about the parameter values, which leads to modest improvement in the final model accuracy (Nowlan & Hinton, 1992). Adding training samples artificially generated by applying predefined transformations to the initial samples, which is referred to as data augmentation, has shown to be a promising regularization technique to increase a model performance (Hernández-García & König, 2018a). However, the selection of the best data augmentation is challenging and requires specific domain knowledge. Indeed, the applicability of data transformations is often domain-specific, and the heuristic choice of such transformations can lead to counterproductive results, e.g. when misapplied across different domains. For instance, a data augmentation transformation like a horizontal flip is quite fitting for natural images, as the horizontal mirror image of an object in nature remains

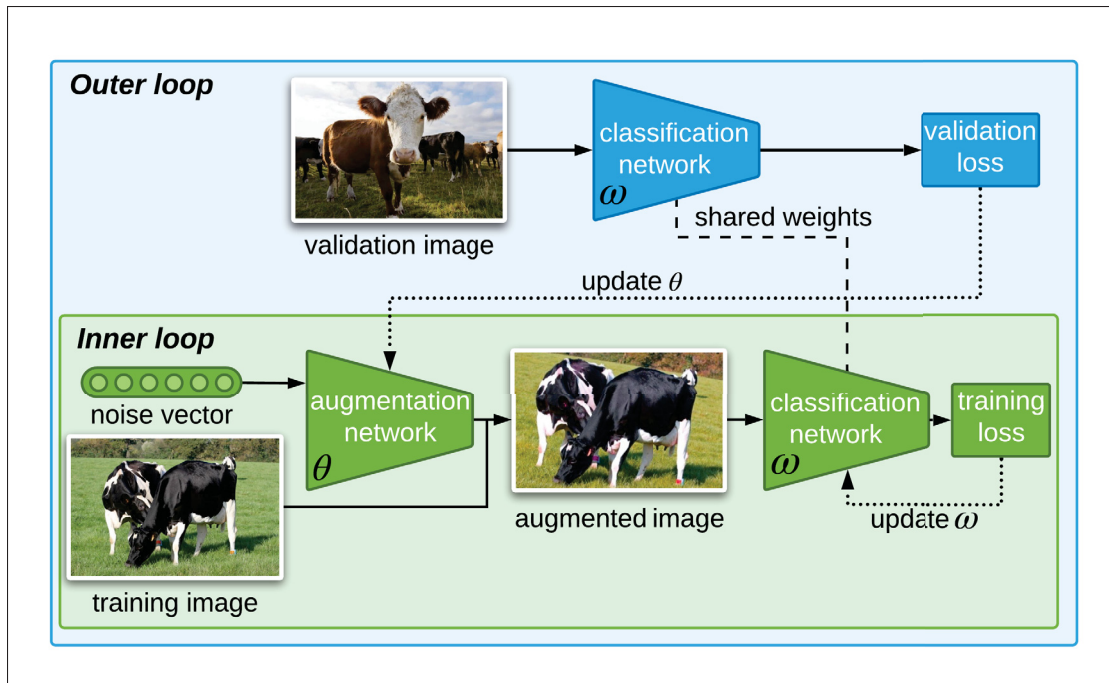


Figure 3.1 Model training

In an epoch, the classifier parameters ω are trained in the standard supervised way in the inner loop in the standard supervised way. Then, in the outer loop, the parameters of the augmentation network generating the data augmentation parameters are trained on the validation set using an online differentiable method.

visually plausible. However, when applied to a dataset comprising numbers or letters, this same transformation can create non-existent symbols or even different symbols of the alphabet, which would introduce confusion during the model training.

A simple way to define the best data augmentation is to use expert knowledge to define the best transformations and their parameters for a given dataset. However, this approach is not practical, as an expert should be consulted for each single dataset to obtain a possibly useful set of transformations and their parameters, which is not always possible due to cost constraints or limited expert knowledge availability.

To mitigate this challenge, an alternative approach is to select those transformations heuristically, and subsequently find their optimal parameters using validation data. Conventionally, a hyperparameter search is performed across different sets of transformations, with the set leading

to the best validation accuracy being chosen as the optimal set of transformations. This approach is appealing as it allows the model to learn the best transformations directly from the data, yet it is not scalable. Given a large range of transformations to test, retraining the algorithm every time with a different transformation set is computationally very demanding. Moreover, selecting the right transformations is not trivial, and selecting the wrong transformations can lead to a degradation of the model performance as shown in Chen, Dobriban & Lee (2020). To tackle this problem, automatic data augmentation methods based on bilevel optimization like Cubuk, Zoph, Mane, Vasudevan & Le (2019a) have been proposed for natural images. Those methods can be computationally expensive as for each new set of parameters, the model in the inner loop needs to be fully trained till convergence. Efforts to reduce the computational cost have led to methods improved using a gradient-based approach like in Hataya, Zdenek, Yoshizoe & Nakayama (2019); Lin *et al.* (2019), or more recently Hataya, Zdenek, Yoshizoe & Nakayama (2022).

In our work, as we can see in Fig.3.1, we do not need to train the classifier in the inner loop until convergence for each different set of data augmentation parameters to test. We propose a method consisting in training an image classifier while learning the best data augmentation transformations in a bi-level optimization framework. The classifier is trained in the inner loop, while the best data augmentation parameters are learned in the outer loop. To make the method computationally efficient, the gradient of the validation loss used to update the data augmentation parameters is estimated using truncated backpropagation and with only one iteration of the inner loop. The augmenter network generating the right data augmentation is trained at the same time as the classifier by alternating between the outer and the inner loop at each iteration.

In our experiments, we validate our method not only on natural images, but also on histological images, for which data augmentation is particularly relevant. Indeed, in histological images, one hurdle to a good model generalization is the color and shape variability of the cells and tissues in the images. Those variations can be inherent to the image acquisition process. More precisely, color variations originate from the cell staining, which is done to make the cells visible to the human eyes, whereas shape variations are due to tissue deformation. Data augmentation is particularly interesting in this case, as it can address the challenges of color and shape variations

simultaneously. By generating images that cover those variations, and employing them during training, the classification model can be taught to be invariant to such transformations.

3.1.1 Contributions

In this work, we tackle the problem of efficiently learning the data augmentation that maximizes the validation accuracy by proposing a method based on bilevel optimization. With this framework, we aim at identifying image transformations that minimize the validation loss while training the end task model at the same time. However, as outlined in Sec. 3.3, instead of solving the complete bilevel optimization problem, we approximate it with an online version where in every iteration a new set of transformations is learned and adapted to the learning phase. This adaption makes the framework computationally more efficient.

To summarize, the contributions of this work are:

- we propose an online, differentiable approach for learning the optimal data augmentation regime using a validation set. As this method is differentiable, we can efficiently optimize a large transformation network learning useful color and affine transformations to perform data augmentation automatically using backpropagation.
- we show that our proposed model, using different sets of transformations, achieves comparable or better results than conventional methods on five different natural images datasets. Further improvements were observed in the medical imaging field, where effective transformations are difficult to define. Our model achieved comparable or better results than hand-defined transformations or RandAugment based methods on six different histological images datasets.

3.2 Related Work

Data augmentation consists in creating new data points from existing ones in order to get a larger training set. It was found to be essential for achieving state-of-the-art image classification results (Hernández-García & König, 2018b).

The selection of the data augmentation transformations used to train a model is usually done using heuristics or expert domain knowledge. Image transformations for natural images have been extensively researched, and usual transformations are image flip, rotation and color changes as described in Perez & Wang (2017). The usage of more complex transformations such as occluding parts of an image (DeVries & Taylor, 2017b) or blending images (Zhang *et al.*, 2018; Lemley *et al.*, 2017) has also been proposed. Recently, some works have investigated adversarial samples as a possibility of data augmentation, as in Gong *et al.* (2021) or Suzuki (2022).

Although not as extensively as for natural images, data augmentation transformations based on color alterations or affine transformations have been also researched in the context of histopathological images (Ciompi *et al.*, 2017; Tellez *et al.*, 2019; Ataky, De Matos, Britto, Oliveira & Koerich, 2020; Faryna, van der Laak & Litjens, 2021; de Matos, Ataky, de Souza Britto, Soares de Oliveira & Lameiras Koerich, 2021; Wagner *et al.*, 2021; Garcea, Serra, Lamberti & Morra, 2022). Other data augmentation technics have also been explored, like mix-up (Chang *et al.*, 2021), pyramid blending (Ataky *et al.*, 2020), or style transfer (Wagner *et al.*, 2021). Specific color data augmentation technics for histopathological images have been proposed in Tellez *et al.* (2019), or Faryna *et al.* (2021).

Although the data augmentation transformations usually chosen can improve the task performance, there is no guarantee that they are optimal nor that they are even useful at all. Badly chosen transformations can lead to a degradation of the model performance, as shown in Chen *et al.* (2020). To avoid the manual selection of transformations, automatic data augmentation learning methods have been proposed for natural images.

We distinguish those methods in 3 categories: Generative images-based, AutoAugment-based approaches and methods coming from the hyperparameter optimization field.

3.2.1 Generative images-based

Generative Adversarial Networks (GANs) (Goodfellow *et al.*, 2014) can generate realistic new samples of a certain dataset or class, thus they can be adapted for data augmentation.

(Mirza & Osindero, 2014) and (Odena *et al.*, 2017) proposed to generate images conditioned on their class that could be directly used to augment a dataset. CatGAN (Springenberg, 2016) on the other hand, performs unsupervised and semi-supervised learning as a regularized information maximization problem (Krause *et al.*, 2010) with a regularization based on the generated samples. Also based on GAN, but directly used for data augmentation, DAGAN (Antoniou *et al.*, 2018) conditions the augmented image on the input image. TripleGAN (Chongxuan *et al.*, 2017) and Bayesian data augmentation (Tran *et al.*) train a classifier jointly with the generator. These approaches generate general image transformations, but in practice, it is not as performant as using predefined transformations. TANDA (Ratner *et al.*, 2017) is the only GAN-based approach that uses predefined transformations. It defines a large set of transformations and learns how to combine them to generate new samples that follow the same distribution as the original data. This approach is better, but it is still based on the assumption that the augmented data should follow the same distribution as the original data. Instead, we argue that data augmentation should improve the performance of the classifier, independently of the visual similarity of the generated data. In medical images, Frid-Adar, Klang, Amitai, Goldberger & Greenspan (2018) use a GAN to generate new CT-Scan images to improve the training of a liver lesion classifier. Our model is more efficient than GAN based models, as it does not require learning a separate model before training the classifier. It learns the best transformation parameters and classifier at the same time. Note that, due to the complexity of their training and the limitations in their scalability, preventing the generation of high resolution images, GANs tend to be replaced by Diffusion models (Dhariwal & Nichol, 2021) introduced in Song & Ermon (2019) and Ho *et al.* (2020). In the medical imaging field, the usage of Diffusion models has been studied in Kazerouni *et al.* (2023).

3.2.2 AutoAugment

AutoAugment (Cubuk *et al.*, 2019a) is a data augmentation method that learns sequences of transformations that maximize the classifier accuracy on a validation set. This objective is better than simply reproducing the same data distribution as in GAN-based models, as it favors

transformations that generalize well on unseen data. However, it is computationally expensive, as it performs the complete bilevel optimization by training the classifier in the inner loop until convergence for each set of evaluated transformations. Some solutions to reduce the computational cost are proposed in follow-up works. Fast AutoAugment (Lim, Kim, Kim, Kim & Kim, 2019) optimizes the search space by matching the density between the training set and the augmented data. Alternatively, Population Based Augmentation (PBA) (Ho, Liang, Stoica, Abbeel & Chen, 2019) focuses on learning the optimal augmentation schedule rather than only the transformations. However, even if these approaches reduce the computational cost of AutoAugment, they do not leverage gradient information. Faster AutoAugment (Hataya *et al.*, 2019) does this by combining AutoAugment with a GAN discriminator and considering transformations as differentiable functions. OHL-Auto-Aug (Lin *et al.*, 2019) uses an online bilevel optimization approach and the REINFORCE algorithm on an ensemble of classifiers to estimate the gradient of the validation loss and learn an augmentation probability distribution. RandAugment (Cubuk, Zoph, Shlens & Le, 2019b) goes further by showing that a same performance level as AutoAugment can be obtained by randomly selecting transformations from the predefined pool and just tune the number of transformations to use and a global (same for all transformations) magnitude factor. However, this approach also requires prior knowledge of useful transformations. In histopathological images, Faryna *et al.* (2021) use the RandAugment method with a set of transformations extended with some specific color transformations. This leads to an improved performance of the classifier.

Our model is more efficient than search-based methods, as the data augmentation parameters are updated at each training iteration using the gradient of the validation loss obtained in the inner loop. This gradient is estimated using truncated backpropagation, which removes the need to train the model until convergence for each set of evaluated transformations.

3.2.3 Hyperparameter Learning

Our work has some roots in the hyperparameter optimization field, as data augmentation parameters can be considered as hyperparameters to tune. Hyperparameters tuning is important

to obtain the best performances when training neural networks on a given dataset. Classic approaches assume that the learning model is a black-box and use methods like grid search, random search (Bergstra, Bardenet, Bengio & Kégl, 2011; Bergstra, Yamins & Cox, 2013), Bayesian optimisation (Snoek, Larochelle & Adams, 2012), or a tree-search approach (Hutter, Hoos & Leyton-Brown, 2011). These approaches are simple but expensive because they repeat the optimization from scratch for each sampled value of the hyperparameters and so are only applicable to low dimensional hyperparameter spaces. A different line of research is to leverage the gradient of these (continuous) hyperparameters (or hyper-gradients) to perform the hyper-optimization. The first work proposing this idea (Bengio, 2000), shows that the implicit function theorem can be used to this aim. This idea was developed more recently in Bertrand *et al.* (2020). Domke (2012) was the first work to propose a gradient-based method using a bilevel optimization approach (Colson *et al.*, 2007) to learn hyperparameters. Using a bilevel optimization approach to train a neural network is challenging, as usually there is no closed-form expression of the function learned in the inner loop (Section 3.3). To address this, Maclaurin, Duvenaud & Adams (2015) and later Franceschi, Donini, Frasconi & Pontil (2017) proposed methods to reverse the forward pass to compute the gradient of the validation loss. However, these methods are applicable only when the number of hyperparameters and the complexity of the models are limited due to the memory needed to save the intermediate steps. Another approach to address the computational hurdle in the inner loop is to calculate an approximation of the gradient, like in Pedregosa (2016) Luketina, Berglund, Greff & Raiko (2016) or MacKay, Vicol, Lorraine, Duvenaud & Grosse (2019). Our method differentiates from those by using truncated back propagation to estimate the gradient of the validation loss. Finally, note that hyperparameter optimization presents some similarities to meta learning, as shown in Franceschi *et al.* (2018). For instance, in MAML (Finn *et al.*, 2017), a shared model initialization is learned to minimize the validation loss and therefore improve the generalization capabilities of the model. More recently, Hataya *et al.* (2022) can be positioned at the intersection of AutoAugment and meta-learning based approaches.

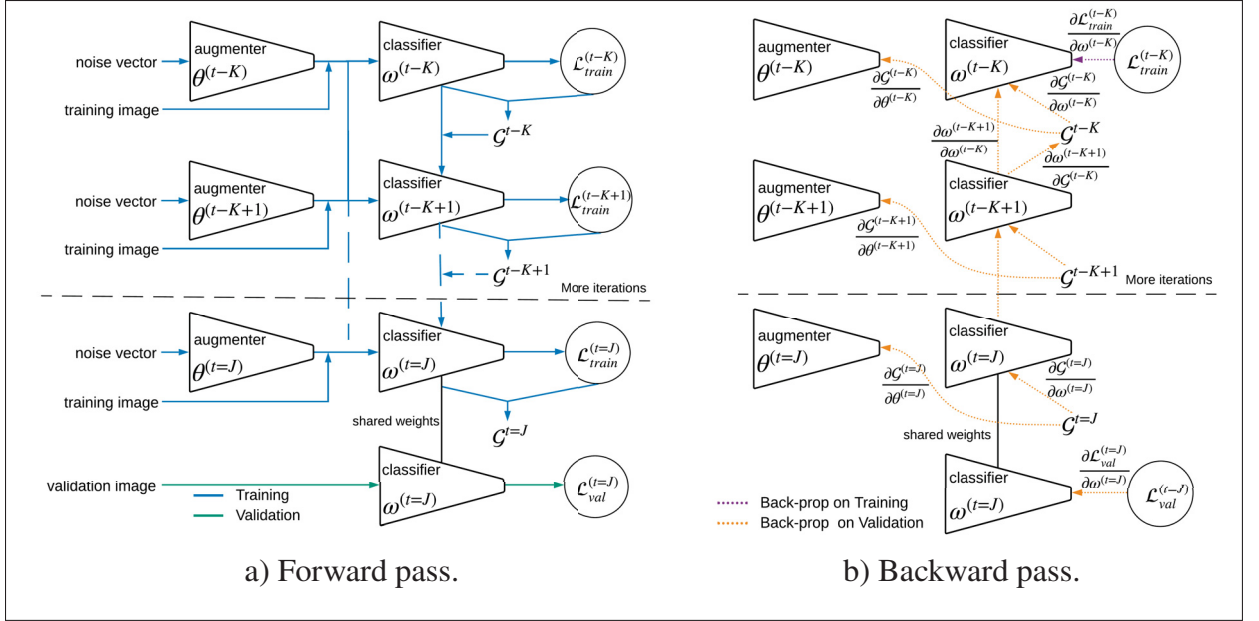


Figure 3.2 Computational graph of our model at iteration $t=J$

K is the number of gradient unfolding steps, and J is the number of inner loop iterations after which θ gets updated. The case where $K=J=T$ (T being the iteration of the classifier convergence) is the complete bilevel optimization as in Eq.3.1 whereas $K=J=1$ corresponds to updating θ at each mini-batch ($K = 1$), using only one step of gradient unfolding ($J = 1$).

3.3 Proposed Method

Consider a labeled set $\mathcal{X} := \{x_i, y_i\}_{i=1}^N$, where x_i is an input image, y_i the associated class label, N the number of samples and $\hat{\mathcal{X}}$ the set of transformed images. We formulate the problem of identifying effective data augmentation transformations as a bilevel optimization problem. In this setup, the augmenter $\mathcal{A}_\theta : \mathcal{X} \rightarrow \hat{\mathcal{X}}$ is parametrized by θ and is used to minimize the loss \mathcal{L} on the validation data \mathcal{X}_{val} in the outer loop. In the inner loop, the classifier parameters ω are optimized on the training data \mathcal{X}_{tr} in the standard supervised way. This formulation can be written as:

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\mathcal{X}_{val}, \omega^*) \quad (3.1)$$

$$s.t. \quad \omega^* = \arg \min_{\omega} \mathcal{L}(\mathcal{A}_\theta(\mathcal{X}_{tr}), \omega). \quad (3.2)$$

While optimizing a few hyperparameters on the validation data is feasible with black-box approaches such as grid and random search (Bergstra & Bengio, 2012) or Bayesian optimization (Snoek *et al.*, 2012), it is not efficient. With bilevel optimization, our aim is to efficiently learn an entire neural network \mathcal{A}_θ (possibly with thousands of parameters θ) which defines a distribution of transformations that should be applied on the training data to improve generalization.

Gradient descent was shown to be an efficient method for optimizing parameters of large networks. In problems such as architecture search (Liu *et al.*, 2019), the parameters can be directly optimized with gradient descent (or second order methods) against the training and validation data. However, this is not the case for data augmentation. The reason is that the transformation network \mathcal{A}_θ is optimized to maximize the validation score, but applies transformations only on the training set. Therefore, first order methods would not work. The aim of data augmentation is to introduce transformations during the training phase that can make the model invariant or partially invariant to any transformations that can occur at test time. If we optimize the transformation network directly on the validation data, the model will simply select trivial solutions, such as the identity transformation. This approach has been used for object localization (Jaderberg *et al.*, 2015) and it did not improve the model generalization performance as much as data augmentation. To solve this issue, new methods relied on reinforcement learning instead of gradient descent to learn effective data augmentation (Cubuk *et al.*, 2019a; Lim *et al.*, 2019; Ho *et al.*, 2019).

In this work, we show that in the case of a differentiable augmentser \mathcal{A}_θ , there is a simple, efficient way to find optimal data transformations based on gradient descent that generalize well to validation data. We formulate our problem as an approximation to bilevel optimization by using truncated back-propagation as it allows our method to:

- efficiently estimate a large number of parameters to generate the optimal data augmentation transformations by gradient descent;
- obtain an online estimation of the optimal data augmentation during the different phases of the training, which can also be beneficial (Golatkar, Achille & Soatto, 2019);

- change the training data to adapt to different validation conditions as in supervised domain adaptation.

Although approximate bilevel optimization has already been proposed for hyperparameter optimization (Shaban, Cheng, Hatch & Boots., 2019; Franceschi *et al.*, 2018, 2017), in this paper we show that it can be used for training a large, complex model (the augmenter \mathcal{A}_θ network) in order to learn an effective distribution of transformations.

3.3.1 Approximate Online Bilevel Optimization

As shown in Eq. 3.1 and 3.2, the problem of finding the optimal data augmentation transformations \mathcal{A}_θ can be cast as a bilevel optimization problem. This problem can be solved by iteratively solving Eq. 3.2 to find the optimal network weight ω^* , given the parameters of the transformation θ and then updating θ :

$$\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}(\mathcal{X}_{val}, \omega^*) \quad (3.3)$$

where η_θ is the learning rate used to train the augmenter network.

However, as the augmentations are to be applied only on the training dataset and not on the validation set, calculating $\frac{\partial \mathcal{L}(\mathcal{X}_{val}, \omega^*)}{\partial \theta}$ is not trivial. To enable this calculation, we use the fact that the weights ω of the network are shared between training and validation data and use the chain rule to differentiate the validation loss $\mathcal{L}(\mathcal{X}_{val}, \omega^*)$ with respect to the hyperparameters θ . In other words, instead of using a very slow black-box optimization for θ , we can exploit gradient information because the model parameters ω^* are shared between the validation and the training loss.

We define the gradient of the validation loss with respect to θ as follows:

$$\begin{aligned} \nabla_\theta \mathcal{L}(\mathcal{X}_{val}, \omega^*) &= \frac{\partial \mathcal{L}(\mathcal{X}_{val}, \omega^*)}{\partial \theta} \\ &= \frac{\partial \mathcal{L}(\mathcal{X}_{val}, \omega^*)}{\partial \omega^*} \frac{\partial \omega^*}{\partial \theta} \end{aligned} \quad (3.4)$$

By defining $\mathcal{G}^{(t)}$ as the gradient of the training loss at iteration t :

$$\mathcal{G}^{(t)} = \nabla_{\omega} \mathcal{L}(\mathcal{A}_{\theta}(\mathcal{X}_{tr}), \omega^t) \quad (3.5)$$

we can write $\frac{\partial \omega^*}{\partial \theta}$ in Eq. 3.4 as:

$$\frac{\partial \omega^*}{\partial \theta} = \sum_{i=1}^{T-1} \frac{\partial \omega^{(T)}}{\partial \omega^{(i)}} \frac{\partial \omega^{(i)}}{\partial \mathcal{G}^{(i-1)}} \frac{\partial \mathcal{G}^{(i-1)}}{\partial \theta} \quad (3.6)$$

where T is the iteration when the classifier converges.

As ω^* represents the model weights at training convergence, they depend on θ for each iteration of gradient descent. Thus, to compute $\frac{\partial \omega^*}{\partial \theta}$, one has to back-propagate throughout the entire T iterations of the training cycle. An example of this approach is Maclaurin *et al.* (2015). This approach is feasible only for small problems due to the large requirements in terms of computation and memory. However, as optimizing ω^* is an iterative process, instead of computing $\frac{\partial \omega}{\partial \theta}$ only at the end of the training loop, we can estimate it at every iteration t :

$$\frac{\partial \omega^*}{\partial \theta} \approx \frac{\partial \omega^{(t)}}{\partial \theta^{(t)}} = \sum_{i=1}^t \frac{\partial \omega^{(t)}}{\partial \omega^{(i)}} \frac{\partial \omega^{(i)}}{\partial \mathcal{G}^{(i-1)}} \frac{\partial \mathcal{G}^{(i-1)}}{\partial \theta^{(i)}}, \quad (3.7)$$

This procedure corresponds to dynamically changing θ during the training iterations (thus it becomes $\theta^{(t)}$) to minimize the current validation loss based on the training history. Although this formulation is different from the original objective function, adapting the data augmentation transformations dynamically with the evolution of the training process can improve generalization performance (Golatkar *et al.*, 2019). This relaxation is often used in constrained optimization for deep models, in which constraints are reformulated as penalties and their gradients are updated online, without waiting for convergence, to save computation (Pathak, Krähenbühl & Darrell, 2015). However, in our case, we cannot write the bilevel optimization as a single unconstrained formulation in which the constraint in ω^* is summed with a multiplicative factor that is maximized (i.e., Lagrange multipliers), because the upper level optimization should be performed only on

θ , while the lower level optimization should be performed only on ω . Nonetheless, even with this relaxation, estimating $\frac{\partial \omega^*}{\partial \theta}$ still remains a challenge as it does not scale well. Indeed, the computational cost of computing $\frac{\partial \omega^{(t)}}{\partial \theta^{(t)}}$ grows with the number of iterations t as shown in Eq. 3.7. To make the gradient computation constant at each iteration we use truncated back-propagation similarly to what is commonly used in recurrent neural networks (Williams & Peng, 1990):

$$\frac{\partial \omega^{(t)}}{\partial \hat{\theta}} \approx \sum_{i=t-K}^t \frac{\partial \omega^{(t)}}{\partial \omega^{(i)}} \frac{\partial \omega^{(i)}}{\partial \mathcal{G}^{(i-1)}} \frac{\partial \mathcal{G}^{(i-1)}}{\partial \theta^{(i)}}, \quad (3.8)$$

where K represents the number of gradient unfolding that we use.

Fig. 3.2. shows the computational graph used for this computation.

Additionally, as in Williams & Peng (1990), we consider a second parameter J which defines the number of inner loop training iterations after which θ is updated, in other words how often the computation of the gradients of θ is performed. The situation where $K = J = T$ is the exact bilevel optimization as shown in Eq. 3.1 while $K = J = 1$ corresponds to updating θ at each iteration, in our case mini-batch ($K = 1$), using only one step of gradient unfolding ($J = 1$). A theoretical analysis of the convergence of this approach is presented in (Shaban *et al.*, 2019).

3.3.2 Augmenter Networks

In this work, we use an augmenter network that can learn two types of transformations: geometric and color. We use the transformation model of spatial transformer networks (Jaderberg *et al.*, 2015) presented in Chap. 1 Sec.1.2.4, but for data augmentation instead of data alignment. Thus, as illustrated in Fig. 3.1, the augmenter is composed of a module that generates a set of transformation parameters followed by a module that applies the generated transformations to the original image. Note that the learned transformations are not conditioned on the input image but defined only based on random noise.

Geometric transformation are particularly relevant for data augmentation. In natural images, they simulate the fact that in the real world, the same object can be located at multiple positions and seen from different viewpoints. For histological images, they are motivated by the fact that tissues

are deformed during the image acquisition process. By considering affine transformations in our learned data augmentation, we aim to train the model to become invariant to those geometric deformations. For affine transformations, the augmenter network receives as input a random noise vector and generates a 2×3 matrix of values representing a variation from the identity transformation. In some experiments, the augmenter network learns affine transformations, but in some particular scenarios, it learns only translation. In this case, only two values are learned (translation values respectively on x and y-axis).

Color transformations are important as they can help the trained model to become invariant to color perturbations, appearing for examples during the cell staining process for histological images. Color transformations considered are: contrast, brightness, and in the HSV space, hue and saturation. For each color transformations, the augmenter receives as input a random noise vector and generates a single value representing a variation value. In our implementation, we use the kornia library (Riba, Mishkin, Ponsa, Rublee & Bradski, 2019), which follows the specifications of Szeliski (2010).

For contrast, the value learned is a non-negative factor applied to the actual color values. 1 represents the initial image, whereas values tending to 0 mean a black-and-white image. If we consider the variables r , g , and b representing the values of the red, green, and blue colors of the images and cf the contrast factor learned by our network, the new RGB values are obtained using the update rule:

$$(r, g, b) \leftarrow clamp((r, g, b) \cdot cf, 0, 1) \quad (3.9)$$

For brightness, the value learned represents a shift applied to the actual color values. 0 represents the initial image. If we consider the variables r , g , and b representing the values of the red, green, and blue colors of the images and bs the brightness shift learned by our network, the new RGB values are obtained using the update rule:

$$(r, g, b) \leftarrow clamp((r, g, b) + bs, 0, 1) \quad (3.10)$$

In the case of saturation, the value learned by the augmenter is a non-negative factor applied to the actual saturation value. A value of 1 represents the original image, whereas 0 means a

black-and-white image. If we consider the variables h , s , and v representing the values of the hue, saturation, and value of the images and sf the saturation factor learned by our network, the new HSV values are obtained using the update rule:

$$(h, s, v) \leftarrow \text{clamp}((h, s \cdot sf, v), 0, 1) \quad (3.11)$$

Finally, the value learned by our augmenter for hue is a shift of the hue channel. 0 represents no shift to the hue channel, and any other value negative or non-negative is added to the actual value. If we consider the variables h , s , and v representing the values of the hue, saturation, and value of the images and hs the hue shift learned by our network, the new HSV values are obtained using the update rule:

$$(h, s, v) \leftarrow (\text{mod}(h + hs, 2\pi), s, v) \quad (3.12)$$

3.4 Experimental Setup

3.4.1 Datasets

In experiments with natural images, we consider the four following datasets:

CIFAR10 (Krizhevsky *et al.*, 2009) is a dataset composed of 60,000 32×32 natural color images distributed in 10 different classes (6,000 images per class). This dataset is split into a training set of 50,000 images and a test set of 10,000 images. CIFAR100 (Krizhevsky *et al.*, 2009) is an extension of CIFAR10 dataset. It contains the same number of images at the same resolution, but they are distributed in 100 classes instead of 10. ImageNet (Russakovsky *et al.*, 2015) is a dataset of 1.28 million natural color images in the training set and 50,000 images in the test set. As the image size is variable, we resize them in our experiments to a resolution of 224×224 . Tiny ImageNet is a subset of ImageNet (Russakovsky *et al.*, 2015) containing 200 classes and images resized to 64×64 . Each class has 500 training images, 50 validation images, and 50 test images. Since the test labels are not available, the validation set is used as test set and 20% of

the training set is used for validation.

In experiments with histological images, we consider the four following datasets:

BACH (Aresta *et al.*, 2019) is a dataset of 400 H&E (hematoxylin and eosin) stained breast cancer histology images of resolution 2048×1536, distributed in 4 balanced classes of 100 images. As there is no test set publicly available, we use in our experiments 40% of the dataset for training, 10% for validation and 50% for testing as in Rony *et al.* (2023). The values used for the predefined color transformations are brightness=0.5, contrast=0.5, saturation=0.5 and hue=0.05. Glas (Sirinukunwattana *et al.*, 2016) is a dataset of 165 H&E stained colon cancer histology images of variable resolution (in our experiments, we use an image size of 430×430) distributed in 2 classes (benign and malignant). The dataset is divided in a train set of 85 images (37 benign and 48 malignant) and a test set of 80 images (37 benign and 43 malignant). In our experiments, we use 80% of the training set for training and 20% for validation. The values used for the predefined color transformations are brightness=0.25, contrast=0.25, saturation=0.25 and hue=0.4. HICL Larynx (Ninos *et al.*, 2015) is a dataset of 450 H&E and P63 stained larynx cancer histology images with 2 magnifying factors (20x and 40x). It has 3 classes corresponding to cancer grades: Grade I, II and III. For the 20x magnification factor, the image resolution is 1728×1296 and the number of images per class is I:87, II:73 and III:64. For the 40x magnification factor, the image resolution is 1300×1030 and the number of images per class is I:88, II:74 and III:64. As there is no test set publicly available, we use in our experiments 70% of the dataset for training, 20% for validation set, and 10% for test. The values used for the predefined color transformations are brightness=0.25, contrast=0.25, saturation=0.25 and hue=0.4. HICL Brain (Glotsos *et al.*, 2008) is a dataset of 2548 H&E and P63 stained brain cancer histology images with 2 magnifying factors (20x and 40x). It has 7 classes corresponding to cancer grades: Grade I, I-II, II, II-III, III, III-IV and IV. For the 20x magnification factor, the image resolution is 1728×1296 and the number of images per class is I:123, I-II: 94, II:208, II-III:47, III:367, III-IV:45 and IV:373. For the 40x magnification factor, the image resolution is also 1728×1296 and the number of images per class is I:132, I-II: 73, II:210, II-III:53, III:434, III-IV:32 and IV:357. As there is no test set publicly available, we use in our experiments 70% of the dataset for training, 20% for validation set, and 10% for test. The values used for

the predefined color transformations are brightness=0.25, contrast=0.25, saturation=0.25 and hue=0.4.

3.4.2 Evaluation

To evaluate the performance of our models, we use the classification accuracy metric, which is defined as the number of samples correctly classified divided by the total number of samples. In scenarios with medical images, we do a 5-fold cross-validation and the result reported is the average of the results obtained by the 5 folds. The hyperparameters search is done separately for each dataset. The hyperparameters selected are the ones yielding the best validation results averaged over the 5 folds. We also follow this protocol for Randaugment hyperparameters in experiments with histological images, where we compare our approach to this method.

3.4.3 Implementation Details

Model architecture Our model is composed of a classifier and an augmenter network. To facilitate fair comparison of the results, we use in our experiments the same classifiers as in previous works, for natural images BadGAN (Dai *et al.*, 2017), ResNet18 (He *et al.*, 2015), ResNet50 (He *et al.*, 2015) and Wide-ResNet-28-10 (Zagoruyko & Komodakis, 2016) and for histological images a ResNet18 pretrained on ImageNet. BadGAN is a simple CNN based architecture composed of 9 convolutional layers with Leaky ReLUs and a MLP classifier. Its architecture is detailed in Appx. II-1. ResNet18 and ResNet50 are respectively 18 and 50 layers deep neural networks with residual connections, and WideResNet 28-10 is a ResNet network with 28 layers and a width factor of 10.

One particularity in our experiments with histological images, is that to align with the image size used during pretraining, we use in our training phase patches of size 224×224 and evaluate the model on whole images during the testing phase.

The augmenter learning the geometric and color transformations is a MLP receiving a noise vector as input and generating the transformation parameters. For affine transformations, the augmenter network learns a 2×3 matrix containing the parameters of the affine transformations, whereas for color transformations, it outputs 1 value per learned transformation i) Hue in range $[-0.5:0.5]$; ii) Saturation in range $[0:1]$; iii) Contrast in range $[-1:1]$ iv) Brightness in range $[0:1]$.

For natural images, we experimented with three augmenter sizes. The *small* network has an input and output size of n , n being the number of hyperparameters to optimize (6 for affine transformations, 2 for translation and 1 for each color transformations), and it has two layers with respectively n and $10n$ neurons. The *medium* one has an input size of 100 and two layers of 64 and 32 neurons. The *large* one has an input size of 100 and four layers of 512, 1024, 124 and 512 neurons. Details of the architecture can be found in Appx. II-2 For histological images, we used only the medium-sized model. In order to have differentiable affine and color transformations, we use the Kornia (Riba *et al.*, 2019) library and the `affine_grid` and `grid_samples` functions of the torchvision package of Pytorch framework Paszke *et al.* (2019).

In experiments with histological images where we compare to RandAugment, to be fair in the comparison of the results, we limited the pool of transformations used by RandAugment to the transformations learned by our proposed model. This pool of transformations is detailed in Appx. II-3.

Training parameters In all experiments with natural image, we use 20% of the training set to form the validation set. Although in principle we usually use a separate validation set for training the augmenter, in practice, we noticed that reusing the training data in a variant of this holdout approach (the training set is randomly split into train and validation at each epoch) yields better results. However, it is important that the batch of samples used to learn the augmenter is different from the one used to train the classifier to ensure that the model learns data augmentation parameters that generalize well. The data splits used in experiments with histological images are described in Sec. 3.4.1. In preliminary experiments, we tried different values for the frequency of updating θ J and the number of steps of backpropagation K , but they

did not show relevant improvements. Therefore, for all our experiments, we use $K = J = 1$. In practice, the classifier is updated after each training mini-batch and the augmenter after each validation mini-batch.

3.5 Results

The goal of our method is to automatically learn data augmentation. Our experiments compare the performance of different classifiers without data augmentation (baselines), the same classifiers with the best-known hyperparameters for data augmentation (predefined), state-of-the-art methods, and our method. We experiment with two groups of transformations: geometric and color transformations.

In a first part, we run experiments on natural images, then in a second one we focus on histological images.

3.5.1 Natural images

Geometric Transformations In this section, we evaluate our model by investigating learned geometric transformations.

In a first experiment, we assess the computational efficiency of our method and the utility of the learned geometric transformations for the classification task. In Tab. 3.1 we compare the performance of our method on CIFAR10 (with ResNet18) against several methods in terms of accuracy and training cost for translation and affine transformations. We define our *baseline* as a training without any data augmentation, and we consider its training time cost as 1. *Predefined* represents a classifier trained with the usual standard geometric data augmentation: horizontal flip and random translation between -4 and 4 pixels along x and y-axis. To estimate the training cost of this scenario, we consider the general case where the best data augmentation setting is not known and many different values have to be tested using a grid or random search. For instance, for the 6 parameters of an affine transformation, and 2 different values to try for each parameter, the number of models to validate is $2^6 = 64$. In the third case, the augmenter is trained to be

Table 3.1 Impact and training cost of different geometric data augmentation strategies on classification accuracy (%) on CIFAR10
 Considering only translation and affine transformations, our approach is faster than methods requiring a validation loop and is more efficient than predefined data augmentation, STN and validated magnitude of predefined data augmentation.

ResNet18 / CIFAR10	Trans.	Affine	Cost
Baseline	88.55	88.55	1
Predefined	95.28	94.59	> 60
Transf. invariant (STN)	92.14	90.31	1.1
Validated magnitude	94.58	93.43	11.5
Our model	95.35	95.16	5.3

transformation invariant similarly to the spatial transformer networks (Jaderberg *et al.*, 2015). The transformations generated by the augmenter are applied on training as well as on test, and the update of the augmenter parameters θ is done on the same data as the update of the classifier parameters ω . This approach has a very low computational cost (1.1, just the overhead of applying the augmenter) and its accuracy is better than using no data augmentation, but far from a model trained with a good data augmentation. Finally, we consider a *validated magnitude* approach that selects only a single parameter defining the magnitude of the transformation parameters from which an actual transformation is sampled from. This is similar to the strategy used in RandAugment (Cubuk *et al.*, 2019b). This performs surprisingly well, but is still inferior to our model and with a higher cost for validating the magnitude of the transformations. The last row presents the results of our approach. For both translation only and affine transformations, we obtain better results than the other approaches. In terms of computational cost, our approach is around 5 times slower than a basic training without data augmentation. However, as our approach learns the data augmentation parameters directly and does not need to loop over possible values, it is already 14x faster than the simple case of predefined data augmentation described above where we consider only 2 possible values for each parameter.

Table 3.2 Impact of architecture on Classification Accuracy (%)
 Increasing the classifier size improves the model performance. Increasing the augmenter size has no significant impact on the final classification accuracy.

CIFAR-10 Aug.-Class.	BadGAN		ResNet18	
	Tr.	Aff.	Tr.	Aff.
Small	93.65	93.62	95.35	95.16
Medium	93.75	93.63	95.25	95.06
Large	93.65	93.39	95.00	94.83

In a second experiment, we investigate the influence of the augmenter network and the classifier size on the performance of a model trained on CIFAR10. In Tab. 3.2, results show that a larger classifier improves the performance. However, the size of the augmenter network does not have a significant impact on the accuracy of the classifier. Thus, in the following experiments, we use the *small* augmenter, which is faster to train.

Color Transformations In this section, we investigate the impact of color transformations alone and in combination with affine transformations on different datasets.

In this experiment, we study color transformations alone and in combination with affine transformations on CIFAR10. For the predefined color jitter, we use the same settings as in Cubuk *et al.* (2019a). We consider 2 versions of our model, the first one learning only color transformations and the second one learning color and affine transformations. In Tab. 3.3, we can see that in both cases, the learned transformations are yielding better results than predefined ones, which illustrates the efficiency of our approach for color transformations. The best results are obtained when combining color and affine transformations.

Evaluation on Different Datasets We now evaluate our approach on different datasets. In addition to the CIFAR10 results, we report in Tab. 3.4 additional results on CIFAR100, Tiny ImageNet and ImageNet. Predefined transformations used for CIFAR100 and Tiny Imagenet are the same as defined for CIFAR10 in Sec. 3.3.2. Predefined transformations for Imagenet

Table 3.3 Impact of color and affine transformations on classification accuracy (%) on CIFAR10
 Transformations in parentheses are learned, others are predefined. For this dataset, both color and affine transformations improve the classification accuracy. Best performances are obtained with a combination of transformations of both types.

ResNet18 / CIFAR10	Acc. (%)
Baseline	88.55
Baseline + Color Jitter	88.63
Baseline + Affine + HFlip	94.59
Baseline + Affine + Color Jitter + HFlip	94.96
Our model (color)	94.18
Our model (color) + Color Jitter	94.63
Our model (affine + color) + HFlip	95.16
Our model (affine + color) + HFlip + Color Jitter	95.18

Table 3.4 Classification Accuracy (%) of our model on different datasets
 ImageNet results reported are Top1. For all datasets, our model performs better than a classifier trained only with standard predefined data augmentation.

	CIFAR10 ResNet18	CIFAR100 ResNet18	Tiny ImageNet ResNet18	ImageNet ResNet50
Baseline	88.55	68.99	59.69	69.39
Predefined	94.69	73.61	61.10	76.02
Ours (affine)	95.16	74.31	62.92	76.10
Ours (full)	95.42	76.10	63.61	76.20

are a resize to 256×256 followed by a random crop of size 224×224, horizontal flip and color transformations as in He *et al.* (2015). Results show that our model performs better than a classifier trained only with predefined transformations on the four datasets considered already with learned *Affine* transformations, but performances are even better when adding color transformations (*Full*). This shows that our approach can be applied to datasets with different characteristics.

Comparison with SotA Methods In Tab. 3.5, we compare our model to state-of-the-art methods on CIFAR10, CIFAR100 and ImageNet. Predefined transformations are the same as described in Sec. 3.5.1. Results show that on CIFAR10 and using ResNet18 as classifier, our method obtains a better accuracy than GAN-based automatic data augmentation learning methods. AutoAugment has a slightly better accuracy, but note that our model obtains very close results with a smaller network. On bigger networks like Wide ResNet 28-10 and ResNet 50, our approach performs very close to search-based methods. The performance gap is explained by the fact that the search-based methods are using more transformations, in particular non-differentiable transformations, to train the end classifier. On the other end, our model requires less prior knowledge as it does not require defining a list of possible transformation and to perform an additional loop to learn the best augmentation policy from this predefined list. Considering this, it represents an interesting trade-off between training speed and accuracy, especially for datasets where potentially useful augmentations are not trivial to define.

Table 3.5 Comparison of classification accuracy (%) with other models
 ImageNet results reported are Top1/Top5. Our model based on affine and color transformations outperforms previous GAN-based models and performs at a level very close to search-based approaches. Those approaches perform better by considering also non-differentiable transformations, but our approach requires less prior knowledge and no policy search loop, which makes it easier to train and more suitable for datasets where predefining data augmentation is not trivial.

	Classifier	CIFAR10	CIFAR100	ImageNet
Baseline	ResNet18	88.55	68.99	-
Predefined	ResNet18	91.18	73.61	-
Bayesian DA (Tran <i>et al.</i>)	ResNet18	91.00	72.10	-
DAN (Mounsaveng <i>et al.</i> , 2019)	BadGAN	93.00	-	-
TANDA (Ratner <i>et al.</i> , 2017)	ResNet56	94.40	-	-
AutoAugment (Cubuk <i>et al.</i> , 2019a)	ResNet32	95.50	-	-
Ours	ResNet18	95.42	74.31	-
Baseline	WRN 28-10	94.83	69.90	-
Predefined	WRN 28-10	95.76	81.10	-
AutoAugment	WRN 28-10	97.40	82.90	-
Fast AA	WRN 28-10	97.30	82.70	-
PBA	WRN 28-10	97.40	83.30	-
RandAugment	WRN 28-10	97.30	83.30	-
Our model	WRN 28-10	96.44	81.90	-
Baseline	ResNet50	-	-	69.39/89.41
Predefined	ResNet50	-	-	76.02/92.84
Faster AA	ResNet50	-	-	76.50/93.20
AutoAugment	ResNet50	-	-	77.60/93.80
Fast AA	ResNet50	-	-	77.60/93.70
RandAugment	ResNet50	-	-	77.60/93.80
Our model	ResNet50	-	-	76.20/92.90

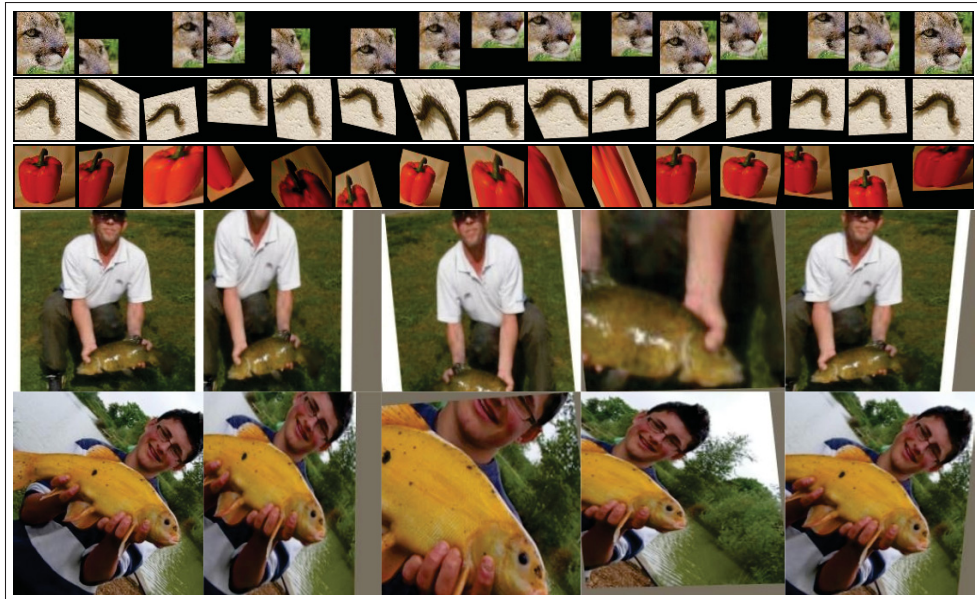


Figure 3.3 Qualitative results

The images of the first column are original images, the following ones are images transformed by our augmenter at different epochs. The first three rows contain images from Tiny ImageNet, and the last two rows from ImageNet.

On Fig. 3.3, we show some examples of transformations learned during the training process. The first 3 rows show examples on Tiny ImageNet. What is interesting to note is that at the beginning of the training (left) transformations tend to be strong, while towards the end of the training (right) they are smaller and tend to approach identity. This behavior can also be seen during training on Imagenet (row 4 and 5).

3.5.2 Histological images

In this section, we validate our method on the four different datasets presented in Sec. 3.4.1: BACH, Glas, Medisp HICL Larynx with magnification factor 20x and 40x, and Medisp HICL Brain with magnification factor 20x and 40x.

In a first series of experiments, we follow the same experimental protocol as for natural images and compare for each dataset the best performance of an image classifier trained with 3 different kinds of transformations: affine transformations, color transformations, and finally a combination

of both transformation types. For each type of transformation, we compare the classification performance of 3 models: a classifier trained without data augmentation (baseline), a classifier trained with hyperparameters for data augmentation found by grid search on a validation set (predefined), and finally a classifier trained with the data augmentation learned by our method.

Then, in a second series of experiments, we investigate our model more in-depth, focusing on the BACH dataset. We chose this dataset as it is the most challenging of the four considered in terms of image size and difficulty to learn the classification task. More precisely, we investigate two aspects of our model: i) the impact of the amount of training data available on the quality of the learned augmentations ii) the impact of starting the training from a classifier pretrained on Imagenet.

Finally, in a third series of experiments, we compare our approach to a model trained in a data augmentation framework similar to RandAugment Cubuk *et al.* (2019b). Instead of searching for the best sequence of transformations and the best magnitude for each transformation at the same time as in other models of the AutoAugment family, RandAugment relaxes the search problem to the tuning of 2 hyperparameters M and N , M being a global magnitude for all considered transformations and N the number of transformations selected in each sequence of transformations. In our experiments, we define M and N by doing a grid search with values between 1 and 5 for both M and N . The obtained values are available in Appx II-4. This approach is simple yet very efficient and has proven to be state of the art in Faryna *et al.* (2021) on Camelyon 17 dataset. However, we argue that even if RandAugment is very simple and efficient to use, it still requires prior knowledge to define the initial pool of transformations. For our method, we also need to fine-tune a limited number of hyperparameters (the hyperparameters of the augmenter network), but we can also define a more generic set of differentiable transformations. Moreover, learning the optimal data augmentation at each epoch can be beneficial for the model, as the time when the transformations are presented to the model is important as reported in Golatkar *et al.* (2019).

Table 3.6 Impact of color and affine transformations on classification accuracy (%) Transformations in parentheses are learned, others are predefined. Our model performs better than hand-defined transformations on the six different datasets. Best performances are obtained with a combination of learned color and affine transformations.

Scenario / Dataset	BACH	Glas	Larynx 20x	Larynx 40x	Brain 20x	Brain 40x
Baseline	83.30 \pm 1.18	89.50 \pm 1.22	87.51 \pm 1.27	86.67 \pm 1.16	99.53 \pm 0.43	96.97 \pm 1.13
Baseline + color DA	85.10 \pm 1.19	96.00 \pm 1.25	90.24 \pm 1.38	86.67 \pm 1.23	99.53 \pm 0.43	97.42 \pm 1.20
Baseline + affine DA	83.70 \pm 1.20	97.75 \pm 1.27	95.92 \pm 1.11	94.29 \pm 1.38	99.54 \pm 0.68	98.18 \pm 1.05
Baseline + color&affine DA	84.60 \pm 1.23	98.25 \pm 1.12	95.24 \pm 1.25	95.24 \pm 1.37	99.53 \pm 0.43	98.94 \pm 1.26
Our model (color DA)	85.60 \pm 1.18	97.25 \pm 1.23	91.11 \pm 1.28	86.67 \pm 1.22	99.53 \pm 0.43	97.57 \pm 1.24
Our model (affine DA)	85.40 \pm 1.25	98.25 \pm 1.20	96.19 \pm 1.22	95.24 \pm 1.26	99.69 \pm 0.43	98.63 \pm 1.36
Our model (color&affine DA)	88.90 \pm 1.25	99.25 \pm 0.56	96.61 \pm 1.23	97.14 \pm 1.26	99.84 \pm 0.35	99.24 \pm 0.54

Geometric Transformations In this paragraph, we evaluate our model by investigating the impact of geometric transformations on the classification accuracy. Results are presented in Tab.3.6. For BACH dataset, our model performs better than predefined affine transformations (+2.1% accuracy over baseline and +1.7% over predefined transformations). However, it does not perform as well as when learning only color transformations, which indicates that this kind of transformation is less efficient for this dataset. For Glas dataset, our model performs also better than predefined affine transformations (+8.75% accuracy over baseline and +0.5% over predefined transformations). Interesting to note is also that for this dataset, affine transformations are helping more to train the model than using only color transformations. For Larynx 20x dataset, we can see that our model performs slightly better than predefined transformations (+8.68% accuracy over baseline and +0.27% over predefined transformations). Also, for this dataset, using geometric transformations seems to help train the model more than using color transformations only. For Larynx 40x dataset, similarly to Larynx 20x dataset, our model performs slightly better (+8.57% accuracy over baseline and +0.95% over predefined augmentations) and affine transformations are more helpful than only color transformations. For Brain 20x dataset, our model performs slightly better than predefined affine transformations (+0.16% accuracy over baseline and +0.15% over predefined affine transformations. As opposed to color transformations, affine transformations have a positive impact on the performance of

the classification model. For Brain 40x dataset, our model performs also slightly better than predefined affine transformations (+1.66% accuracy over baseline and +0.45% over predefined augmentations). Also for this dataset, affine transformations have a bigger positive impact on the model accuracy than color transformations.

Color Transformations In this paragraph, we investigate the impact of color transformations alone on the training of an image classifier. Results are presented in Tab.3.6. For BACH dataset, using predefined color augmentations to train the model yields an increased classification performance compared to the baseline, but the model has the best performance when trained with the augmentations learned by our augmenter (+2.3% accuracy VS baseline and +0.5% accuracy VS predefined color augmentations). For Glas dataset, the classifier performs better with learned transformations than with predefined ones (+7.75% over baseline and +1.25% over baseline). For Larynx 20x dataset, our model also performs better than predefined augmentations (+3.6% accuracy VS baseline and +0.87% accuracy VS predefined augmentations). For Larynx 40x dataset, our model performs similarly to predefined transformations. However, in both cases, we do not see any improvement over the baseline, which indicates that either color transformations might not be the best ones to use for this dataset or that the performance of the classifier is already saturated to see the improvement brought by those transformations. For Brain 20x dataset, similarly to the Larynx 20x dataset, our model performs on-par with predefined transformations and brings no improvement over the baseline. Also in this case, it seems that color augmentations used are not useful to train the classifier or that the performance is too saturated to see the improvement. For Brain 40x dataset, our model performs slightly better than predefined augmentations (+0.6% accuracy VS baseline and +0.13% accuracy over predefined data augmentations).

Combination of color and affine transformations In this paragraph, we evaluate our model by investigating the impact of geometric transformations on the classification accuracy. Results are presented in Tab.3.6. For BACH dataset, the combination of both kind of transformations is significantly improving the classification score (+5.6% accuracy over baseline and +4.3% over predefined augmentations). For Glas dataset also, the combination of both color and

geometric transformations is improving the model performance (+9.75% accuracy over baseline and +1% over predefined augmentations). For Larynx 20x dataset, the combination of color and affine transformations learned by our model has a bigger positive impact on the model final accuracy (+9.1% over baseline and +1.37% over predefined transformations). For Larynx 40x dataset, learning both color and affine transformations is also yielding the model with the best classification accuracy (+10.47% over baseline and +1.9% over predefined transformations). For Brain 20x dataset, our model learning color and affine transformations is performing slightly better than predefined transformations (+0.31% over baseline and predefined transformations). For Brain 40x dataset, when learning color and affine transformations at the same time, our model is performing better than the predefined transformations (+2.27% over baseline and +0.3% over predefined transformations). To summarize the results of this series of experiments, we can see that the combination of both color and affine transformations yields the best results, which shows that our model became more invariant to color and shape perturbations thanks to the data augmentation transformation learned along the training.

Additional experiments on BACH dataset In this section, we run a series of experiments on BACH dataset to have a better understanding of our model. BACH was chosen as it is the most challenging dataset of the six considered in terms of image size and difficulty of the classification task, as shown in Tab. 3.6.

In a first experiment, we investigate the evolution of the model accuracy with respect to the amount of training data. In Fig.3.4, we can see that our model performs better than using only predefined transformations when using the full training set. When we reduce the amount of data gradually, we can see that the amplitude of the improvement decreases for color only and geometric only transformations. Below a threshold of 50% of the training set, our model performs on par with predefined augmentations when learning color or geometric transformations only, but yields an inferior performance when learning both types of transformations at the same time. In this case, our model does not have enough data to learn useful transformations. This shows that having a minimum amount of training data is a limitation and a prerequisite of our data-based learning method.

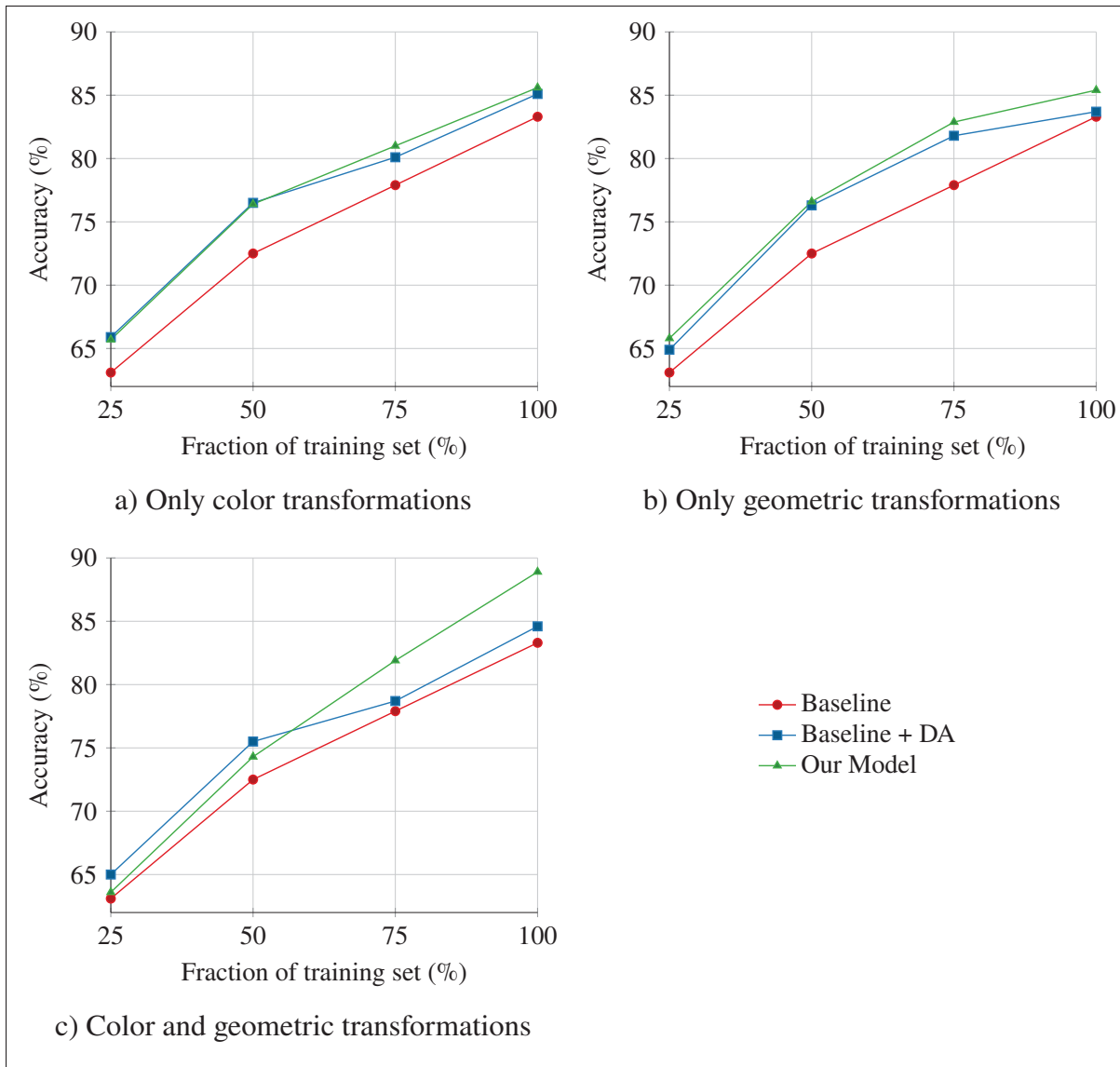


Figure 3.4 Classification Accuracy (%) on BACH dataset in function of the amount of training data

Our model performs better than using only predefined transformations when using the full training set. When we reduce the amount of data gradually, we can see that the amplitude of the improvement decreases for color and geometric only transformations. Below a threshold of 50% of the training set, our model performs on par with predefined augmentations when learning color or geometric transformations, but yields an inferior performance when learning both types of transformations at the same time. In this case, our model does not have enough data to learn useful transformations.

Table 3.7 Impact of the pretraining on the classification accuracy (%) on BACH dataset

Transformations in parentheses are learned, others are predefined.

The best classification accuracy is obtained when training a model pretrained on ImageNet. However, we can see that when training a model from scratch, the baseline accuracy is lower and using data augmentation has a bigger impact. (+14% when training from scratch for our learned augmentations VS + 6.6% when starting from a pretrained model.)

BACH	From Scratch	Pretrained model
Baseline	71.60 \pm 1.29	83.30 \pm 1.18
Baseline + color	75.20 \pm 1.04	85.10 \pm 1.19
Baseline + affine	74.60 \pm 1.26	83.70 \pm 1.20
Baseline + color&affine	83.20 \pm 1.29	84.60 \pm 1.23
Our model (color DA)	83.90 \pm 1.21	85.60 \pm 1.18
Our model (affine DA)	82.70 \pm 1.99	85.40 \pm 1.25
Our model (color&affine DA)	85.60\pm1.29	88.90\pm1.25

In a second experiment, we investigate the impact of starting from a pretrained model when training a classifier with our proposed method. In Tab. 3.7, we can see that the best classification accuracy is obtained when starting from a model pretrained on ImageNet. However, we can see that when training a model from scratch, the baseline accuracy is lower and using data augmentation has a bigger impact. (+14% when training from scratch for our learned augmentations VS + 6.6% when starting from a pretrained model). This experiment shows that using a pretrained model to boost the performances as usually done in the literature is helping, but using an appropriate data augmentation on top during training can further increase the final model performance.

Comparison with random sequences of data augmentation transformations In Tab. 3.8, we compare our model to a model trained with a RandAugment based framework on the six same datasets. To be fair in the comparison of the results, we limited the transformations in the RandAugment set of available transformations to the only ones that our model is learning, as shown in Tab. II-3. On 5 datasets, our model yields a better classification accuracy than the RandAugment based method. On Glas, both models yield similar results. Similarly to RandAug-

Table 3.8 Comparison to a RandAugment based model
 Our model yields better results than a model trained in a RandAugment based framework on 5 datasets. On Glas it is performing on-par. Our model represents a good solution to learn the optimal data augmentation automatically for color and affine transformations.

	BACH	Glas	Larynx 20x	Larynx 40x	Brain 20x	Brain 40x
Baseline	83.30 \pm 1.18	89.50 \pm 1.22	87.51 \pm 1.27	86.67 \pm 1.16	99.53 \pm 0.43	96.97 \pm 1.13
Predefined color&affine DA	84.60 \pm 1.23	98.25 \pm 1.12	95.24 \pm 1.25	95.24 \pm 1.37	99.53 \pm 0.43	98.94 \pm 1.26
RandAugment based model	87.25 \pm 1.48	99.25 \pm 0.68	95.83 \pm 0.23	96.14 \pm 1.31	99.53 \pm 0.35	99.18 \pm 1.03
Our approach	88.90 \pm 1.25	99.25 \pm 0.56	96.61 \pm 1.23	97.14 \pm 1.26	99.84 \pm 0.35	99.24 \pm 0.54

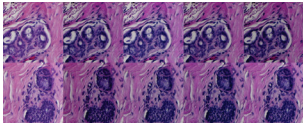
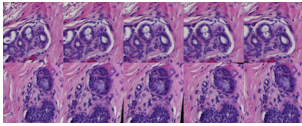
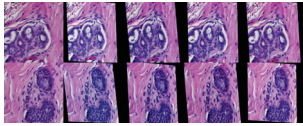
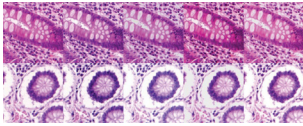
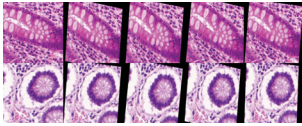
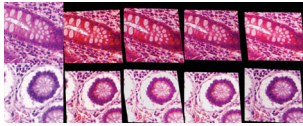
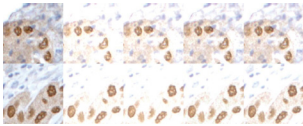
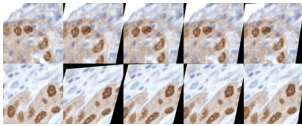
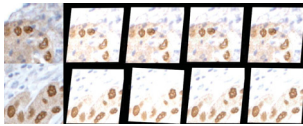
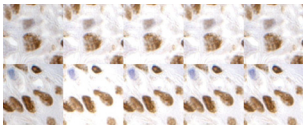
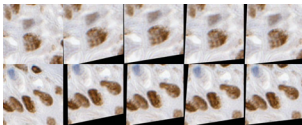
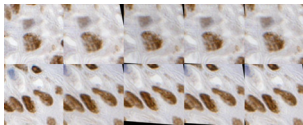
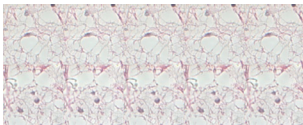
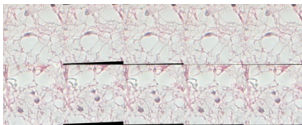
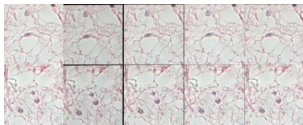

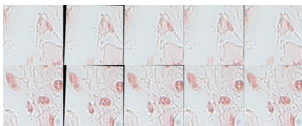
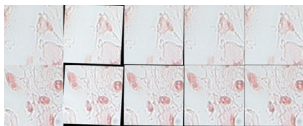
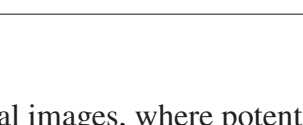
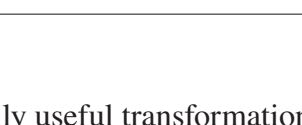
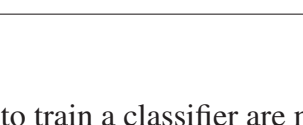
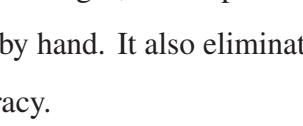
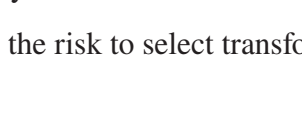
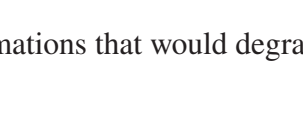
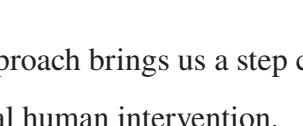
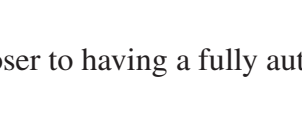
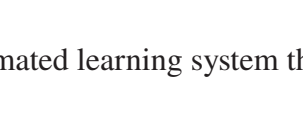









ment, our model has only a few model-specific hyperparameters to tune (the augementer network parameters). However, our model requires less prior knowledge as it does not require defining a precise list of possible transformations but works with a more generic set of differentiable transformations. Our intuition to explain the improved classification performance is that learning the optimal data augmentation for each epoch is beneficial for the model, as the time when the transformations are presented to the model is important, as reported in Golatkar *et al.* (2019).

3.6 Conclusion and Discussion

We have presented a novel approach to automatically learn the transformations needed for effective data augmentation. The method is based on an online approximation of the bilevel optimization problem, defined by alternating between optimizing the model parameters and the data augmentation hyperparameters. By doing so, we train an augementer network to generate the right transformations at the same time as we train the classifier network. We evaluated the proposed approach with different models against a variety of datasets (4 in natural and 4 in histological images) and transformations (geometric and color). The obtained results were comparable or better than the results obtained from defining hand-engineered transformations. In experiments with histological images, it also yielded better results than a model trained with a RandAugment based framework. This shows that our method is very suitable in the context of

Table 3.9 Qualitative results

For each dataset and each scenario, we see the evolution of the learned transformations along the training. Transformations at the beginning of the training are stronger and tend later to finer transformations useful enough to improve the classification accuracy of the trained model. In each row, the first image is the original patch and the last one is the same patch at the end of the training. The images in-between were extracted at respectively at 25%, 50% and 75% of the total number of training epochs.

	Color	Affine	Color and Affine
BACH			
			
			
Glas			
			
			
Larynx20x			
			
			
Larynx40x			
			
			
Brain20x			
Brain40x			

histopathological images, where potentially useful transformations to train a classifier are not trivial to define by hand. It also eliminates the risk to select transformations that would degrade the model accuracy.

Overall, this approach brings us a step closer to having a fully automated learning system that requires minimal human intervention.

CHAPTER 4

BAG OF TRICKS FOR FULL TEST-TIME ADAPTATION

After having explored data augmentation through the lens of generative models and the automatic learning of transformations via bilevel optimization, we delve in this chapter into the world of domain adaptation. Principally a response to the distributional shift issue between train and test data, domain adaptation can also be seen as a possible solution to the challenge of data scarcity. Indeed, given the impracticality of collecting data from every possible domain that might appear at test time, techniques must be devised to adapt the trained model to data distributions unseen during training. In this chapter, we focus more specifically on Fully Test-Time Adaptation (TTA), classifying and benchmarking selected recently proposed State-of-the-Art techniques.

4.1 Introduction

Deep neural networks perform well at inference time when test data comes from the same distribution as training data. However, they become inaccurate when there is a distribution shift (Quionero-Candela, Sugiyama, Schwaighofer & Lawrence, 2009). This distribution shift can be caused by natural variations (Koh *et al.*, 2021) or corruptions (Hendrycks & Dietterich, 2019; Hendrycks *et al.*, 2021). Test-Time adaptation (TTA) aims at addressing this problem by adapting a model pre-trained on source data to make better predictions on shifted target data (Sun *et al.*, 2020; Iwasawa & Matsuo, 2021; Bartler, Bühler, Wiewel, Döbler & Yang, 2022). In this work, we focus on the particular case of Fully Test-Time Adaptation (Fully TTA) (Wang *et al.*, 2021a; Niu *et al.*, 2023; Zhao *et al.*, 2023). In this setting, the adaptation is done source free and relies only on: i) a model pre-trained on data from a source domain and ii) unlabeled test data from a shifted target domain. Separating the training phase from the adaptation phase is particularly relevant for privacy-oriented applications where the training data is not available or can not be disclosed. Fully TTA is also online. Test data is received as a continuous stream, and the model adaptation is done on-the-fly as data is received. This makes the setup more realistic and closer to real-world “in-the-wild” scenarios, where information about potential distribution shifts or about the quantity of data to be received is not necessarily available.

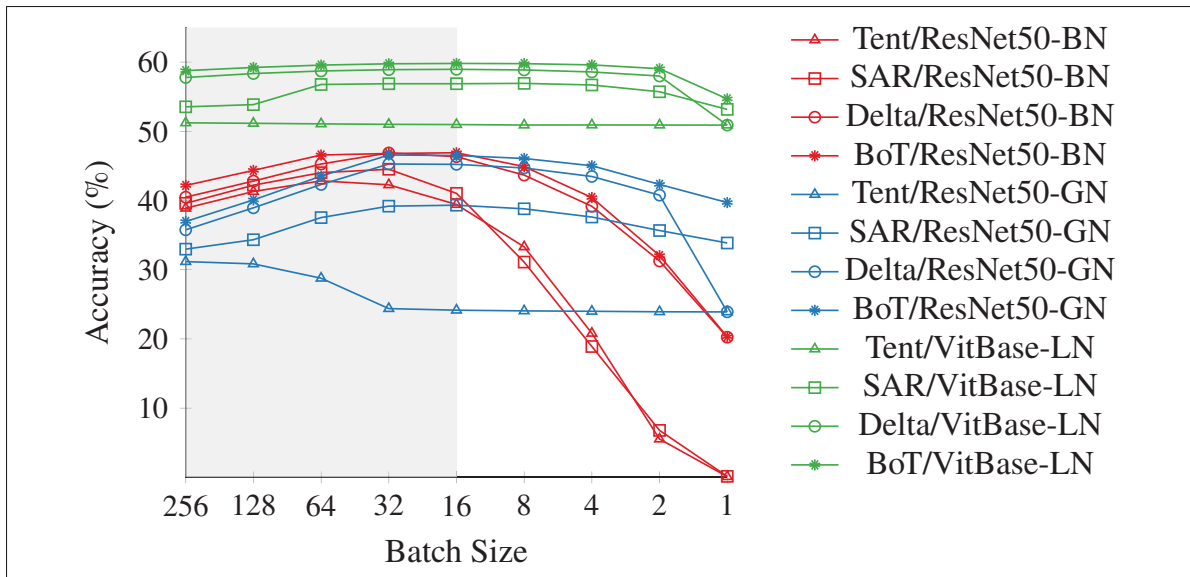


Figure 4.1 Classification Accuracy (%) in function of Batch Size for different methods and architectures on ImageNet-C

In this work, we choose to focus on small batches (16 and below, white zone). As the batch size decreases, the model performances remain stable until a batch size of 32 and then drops significantly for methods running on ResNet50-BN. Results reported are averaged over 15 corruptions and 3 runs. Confidence intervals are too small to be displayed.

Most of the recent solutions proposed to address Fully TTA are follow-ups of seminal work Tent (Wang *et al.*, 2021a) and aim at solving problems inherent to the online and unsupervised aspect of Fully TTA. For example, Zhao *et al.* (2023); Wang, Minku & Yao (2016) deal with the problem of class imbalance in the data stream, Niu *et al.* (2023); Zhang *et al.* (2022) improve the quality of the predictions used to adapt a model by selecting samples with a low entropy or leveraging the predictions of augmented samples and Zhang *et al.* (2022); Zhao *et al.* (2023); Lim *et al.* (2023); Zhang *et al.* (2022) investigate different normalization to stabilize the adaptation process. However, most of the tricks and techniques are presented in combination with others, which makes it difficult to identify their impact on the final model performance. Some techniques might already help when applied alone, whereas others might only work or work better in combination with other tricks.

As this area of research is very active and developing fast, we aim in this study at disentangling the impact of some techniques recently proposed and evaluate objectively their contribution

to the performance of Fully TTA models. We also propose possible improvements in specific cases.

Most TTA papers report results using a batch size of 64. However, in this work, we choose to give particular attention to the online aspect of fully TTA and focus on smaller batch sizes (16 and below), which are closer to the potentially uncontrollable batch sizes of real-world scenarios. Moreover, as we can see in Fig. 4.1, most methods are stable with bigger batch sizes but start showing lower performances when the batch size drops below 16.

4.1.1 Contributions

To address the Fully Test-Time Adaptation problem, we analyzed the following techniques: i) Usage of batch renormalization or batch-agnostic normalization ii) Class re-balancing iii) Entropy-based sample selection iv) Temperature scaling. These analyses were made considering small batch sizes (16 and below), which are closer to the potentially uncontrollable batch sizes in real-world scenarios. Our experimental results show that these techniques, when used alone, are already boosting the performance at test time. However, combining all of them leads to the best classification accuracy compared to a vanilla Tent method and 2 recent state-of-the-art methods across 4 different datasets. In addition to improving accuracy, the chosen techniques confer additional interesting benefits, such as a greater stability of performance with small batch sizes and a reduced computational burden through model adaptation with a reduced set of selected data.

4.2 Related Work

4.2.1 Test-time adaptation (TTA)

As illustrated in Tab. 4.1 proposed initially by Niu *et al.* (2022) and extended based on our understanding of recent work in the field, Test-time adaptation, which is a particular case of

Domain Adaptation, assumes access to a pre-trained model and aims at leveraging unlabeled test instances from a (shifted) target distribution to make better predictions.

Table 4.1 Overview of TTA problem settings
In our work, we consider the Fully Test-Time Adaptation (FTTA) scenario, which is source-free and online.

Setting	Source Data	Target Data	Training Loss	Testing Loss	Online	Source Acc. Maintained
Fine-tuning	✗	x^t, y^t	$\mathcal{L}(x^t, y^t)$	-	✗	✗
Continual learning	✗	x^t, y^t	$\mathcal{L}(x^t, y^t)$	-	✗	✓
Unsupervised domain adaptation	x^s, y^s	x^t	$\mathcal{L}(x^s, y^s) + \mathcal{L}(x^s, x^t)$	-	✗	✓
Test-time training	x^s, y^s	x^t	$\mathcal{L}(x^s, y^s) + \mathcal{L}(x^s)$	$\mathcal{L}(x^t)$	✓	✗
Continual test-time adaptation	✗	x^t	✗	$\mathcal{L}(x^t)$	✓	✓
Fully test-time adaptation (FTTA)	✗	x^t	✗	$\mathcal{L}(x^t)$	✓	✗

Proposed methods usually employ one or a combination of the following techniques: *self-training* to reinforce the model’s own predictions through entropy minimization (Wang *et al.*, 2021a) or Pseudo-Labeling schemes (Lee, 2013), *manifold regularization* to enforce smoother decision boundaries through data augmentation (Zhang *et al.*, 2022) or clustering (Boudiaf *et al.*, 2022), *feature alignment* to mitigate covariate shift by batch norm statistic adaptation (Li *et al.*, 2017; Schneider *et al.*, 2020), and *meta-learning* methods (Goyal *et al.*, 2022) that try to meta-learn the best adaptation loss.

4.2.2 TTA in the broader literature

Although recently introduced (Wang *et al.*, 2021a), TTA shares important motivations and similarities with earlier or concurrent settings that are source-free domain adaptation (SFDA) (Liang *et al.*, 2020; Yang *et al.*, 2021a; Boudiaf *et al.*, 2023) and test-time training (TTT) (Sun *et al.*, 2020; Osowiechi *et al.*, 2022). In SFDA, methods also leverage samples from the target distribution of interest but have no access to source data, and the evaluation is still done on held-out test data. In other words, TTA is the transductive counterpart of SFDA. On the other hand, TTT works by constructing an auxiliary task that can be solved both at training and

adaptation time and therefore, unlike TTA, is not agnostic to the training procedure or to the model architecture.

4.2.3 Fully TTA

TTA is of particular interest for online applications, in which the model receives samples as a stream. Operational requirements for online applications break crucial properties of the vanilla TTA setting, e.g. large batch size or class balance. Under such operational requirements, standard TTA methods degrade, underperforming the non-adapted baseline and even degenerating to random performance in some cases (Boudiaf *et al.*, 2022; Niu *et al.*, 2023). Multiple regularization procedures have been proposed to address such shortcomings. Among them, (i) Improved feature alignment procedures that interpolate, between source and target statistics (Nado *et al.*, 2020; Lim *et al.*, 2023; Zhao *et al.*, 2023), thereby improving overall estimation and decreasing reliance upon specific test batches, (ii) Sample re-weighting (Zhao *et al.*, 2023; Niu *et al.*, 2022) to alleviate the influence of class biases, and (iii) Improving loss' intrinsic robustness to noisy samples, either encouraging convergence towards local minima (Niu *et al.*, 2023) or preventing large deviations from the base model's predictions (Boudiaf *et al.*, 2022; Niu *et al.*, 2022).

4.3 Selected Tricks and Techniques

In this section, we present a classified selection of recently proposed Fully TTA tricks and techniques. As this line of work grows, our work aims at providing an objective evaluation of how they translate into actual robustness for Fully TTA, quantifying the progress made so far, as well as pinpointing possible areas of improvement. We focus our attention on the following topics: architecture and normalization, class rebalancing, sample selection and calibration.

4.3.1 Architecture and Normalization

We start our study by investigating the influence of different architectures and normalization on model performance. Normalization, in particular, has been an active area of research in the TTA literature. Zhao *et al.* (2023) show that in the case of a distribution shift, normalization statistics are inaccurate within test mini-batches, and the gradient of the loss can exhibit strong fluctuations potentially destructive for the model. To address this issue, Lim *et al.* (2023) proposed combining linearly the statistics learned during training with those computed at test time to reduce the gap between the source domain and the target domain. However, this method is not applicable in Fully TTA, as it requires access to labeled source data to learn the linear combination in a post-training phase before using it at test time. Zhang *et al.* (2022) also use a linear combination of the train and test statistics to handle the distribution shift. Zhao *et al.* (2023) adapt batch renormalization (Ioffe, 2017) to test-time adaptation. Batch normalization parameters are updated using a combination of the mini-batch statistics and moving averages of these statistics like in the original paper, but in the TTA context, statistics and moving averages are computed using test batches. Another way to address the issues inherent to batch normalization is to use group or layer normalization instead, as investigated in Niu *et al.* (2023). As the normalization varies greatly between works, this study aims to disentangle its effect from other techniques used.

4.3.2 Class Rebalancing

Subsequently, we explore the problem of online class imbalance in the context of Fully TTA. This problem is strongly relevant in this setting, as data is received as a continuous stream. In this case, there is no guarantee that classes will appear in a balanced way or that different classes will appear in a given batch, especially when the batch size becomes much smaller than the total number of classes in the dataset. Imbalanced data can be particularly detrimental to the model performance as shown in Wang *et al.* (2016); Niu *et al.* (2023); Zhao *et al.* (2023) and can lead in extreme cases to a model collapse to trivial solutions like assigning all samples to the dominant class. To evaluate methods in regard to this problem, we consider two approaches.

In the first one, we follow the setup proposed in Niu *et al.* (2023), where the online imbalanced label distribution shift is simulated by controlling the order of the input samples using a dataset generated with a controlled imbalance ratio. More details about the generation of this dataset are available in 4.4.1. Then, in a second approach, we investigate the evolution of the classification accuracy of different models simply in function of the batch size. We consider small batch sizes already as a factor of online class imbalance, as not all classes can be present in the same batch. In our experiments, we compare three methods: i) Tent without any class rebalancing method is used as a baseline. ii) SAR (Niu *et al.*, 2023) is not a class rebalancing method per se, but the sample selection method introduced in this work is presented as a way to address the class imbalance problem by the authors. iii) DOT is an adaptation of the class-wise reweighting method proposed in Cui, Jia, Lin, Song & Belongie (2019) adapted to the context of test-time adaptation in Zhao *et al.* (2023). The idea of DOT is to estimate the class frequencies in the test set by maintaining a momentum-based class-frequency vector $z \in \mathbb{R}^K$ where K is the total number of classes, based on the prediction of the model of each sample seen previously. Then at inference time, each new sample receives a weight in function of its pseudo label and the current z vector. A sample belonging to a rare class will receive a higher weight than a sample from a class seen more often. The DOT algorithm is detailed in Algo. 4.1.

4.3.3 Sample Selection

In the previous sections, we explored standard mechanisms to address covariate shift (through normalization) and label shift (through class rebalancing). In this section, we go one step further and explore mechanisms that cast TTA as a noisy learning problem. In particular, we explore the sample selection method first proposed in Niu *et al.* (2022) and analyzed more thoroughly after in Niu *et al.* (2023). The main idea of this method is to select only reliable samples for the model adaptation. Indeed, in Niu *et al.* (2023), authors show that samples with high entropy are more likely to have a strong and noisy gradient potentially harmful to the model performance. Furthermore, low-entropy samples contribute more to the model adaptation than high-entropy ones. However, there is no easy way to directly filter out samples with a strong gradient from

Algorithm 4.1 Dynamic Online reweighTing (DOT)
Taken from Zhao *et al.* (2023)

```

Input: inference step  $t := 0$ ; test stream samples  $\{x_j\}$ ; pre-trained model  $f_{\{\theta_0, a_0\}}$ ;
class-frequency vector  $z_0$ ; loss function  $\mathcal{L}$ ; smooth coefficient  $\lambda$ .
1 while the test mini-batch  $\{x_{m_t+b}\}_{b=1}^B$  arrives do
2    $t = t + 1$ 
3    $\{p_{m_t+b}\}_{b=1}^B, f_{\{\theta_{t-1}, a_t\}} \leftarrow \text{Forward}(\{x_{m_t+b}\}_{b=1}^B, f_{\{\theta_{t-1}, a_{t-1}\}})$  // output predictions
4   for  $b = 1$  to  $B$  do
5      $k_{m_t+b}^* = \arg \max_{k \in [1, K]} p_{m_t+b}[k]$  // predicted label
6      $w_{m_t+b} = 1 / (z_{t-1}[k_{m_t+b}^*] + \epsilon)$  // assign sample weight
7   end for
8    $\bar{w}_{m_t+b} = B \cdot w_{m_t+b} / \sum_{b'=1}^B w_{m_t+b'}$ ,  $b = 1, 2, \dots, B$  // normalize sample weight
9    $l = \frac{1}{B} \sum_{b=1}^B \bar{w}_{m_t+b} \cdot \mathcal{L}(p_{m_t} + b)$  // combine sample weight with loss
10   $f_{\{\theta_t, a_t\}} \leftarrow \text{Backward\&Update}(l, f_{\{\theta_{t-1}, a_t\}})$  // update  $\theta$ 
11   $z_t \leftarrow \lambda z_{t-1} + \frac{(1-\lambda)}{B} \sum_{b=1}^B p_{m_t+b}$  // update  $z$ 
12 end while

```

the optimization process. So, instead, an entropy-based filtering method was proposed. More precisely, a threshold entropy E_0 is defined as the maximum entropy $\log K$ multiplied by a factor F , which is a scalar with a value between 0 and 1, 1 meaning no selection at all. All samples with an entropy below this threshold $F \log K$ are kept, whereas the others are discarded when computing the loss value to update the model. Formally, this filtering method can be expressed as a sample selection function S :

$$S(x) = \mathbb{I}_{\{E(x; \Theta) < E_0\}}(x) \quad (4.1)$$

where $\mathbb{I}_{\{\cdot\}}(\cdot)$ is an indicator function, $E(x; \Theta)$ is the entropy of sample x , and E_0 is a threshold predefined as:

$$E_0 = F \log K \quad (4.2)$$

where K is the total number of classes in the dataset and F is a real number in $[0; 1]$.

4.3.4 Calibration

Our last topic of interest is the problem of network calibration in the context of Fully TTA. The calibration of classification networks is a measure of the confidence of the predictions. It is of utmost importance in the context of Fully TTA as it impacts directly the predictions' entropy. Temperature scaling is one technique introduced in Guo, Pleiss, Sun & Weinberger (2017) to improve the calibration of under- or overconfident neural networks by correcting the logits in the softmax function. Formally, it is expressed as:

$$\text{softmax}_\tau(z)_i = \frac{e^{z_i/\tau}}{\sum_{j=1}^K e^{z_j/\tau}} \quad (4.3)$$

where τ is the temperature scaling factor, z is the logits vector of an input sample, i is a class index and K is the total number of classes. A τ value above 1 will lead to a higher entropy with a flattened distribution of the model predictions, whereas a τ value smaller than 1 will lead to a low entropy with a more peaky predictions' distribution. In the context of test-time adaptation, Goyal *et al.* (2022) show that using temperature scaling improves the model accuracy after adaptation when using an entropy minimization-based method. Lee (2013) also shows that when meta-learning the optimal loss for test-time adaptation, the result is an entropy minimization loss with a temperature scaling factor.

4.4 Experimental Setup

In this section, we present the details of our experimental setup. Firstly, we introduce the datasets used, then the different methods we want to compare and the different models, and finally, we explain the evaluation metric and protocol.

4.4.1 Datasets

We evaluate the different methods on several datasets used by prior SFDA or TTA studies: ImageNet-C (Hendrycks & Dietterich, 2019) is a variant of ImageNet (Russakovsky *et al.*, 2015)

where 19 corruption types and 5 levels of severity were applied. For our experiments, we report results using 15 corruption types at the most severe level of corruption (level 5) and keep the 4 remaining extra (speckle noise, gaussian blur, spatter, and saturate) as “validation” corruptions to select hyperparameters, following Zhao *et al.* (2023) and Niu *et al.* (2023). ImageNet-Rendition (Hendrycks *et al.*, 2021) consists of 30,000 images distributed in 200 Imagenet classes obtained by the rendition of ImageNet images like art, cartoons, tattoos, or video games. ImageNet-Sketch (Wang, Ge, Lipton & Xing, 2019) is a dataset of 50,000 images distributed in all ImageNet classes and obtained by querying Google Images with "sketch of ___" where ___ is the name of original ImageNet classes. Images are black and white. Finally, VisDA2017 (Peng *et al.*, 2017) is a dataset of over 72K images distributed in 12 ImageNet classes and containing a mix of synthetic and real domain images.

In the sections where we analyze tricks (Class rebalancing Sec. 4.3.2, Sample Selection Sec. 4.3.3, Calibration Sec. 4.3.4, and Tricks combination Sec. 4.5.5), experiments are done using ImageNet-C. In experiments investigating Class Rebalancing (Sec. 4.5.2 and more particularly Fig.4.3), the ImageNet-C variant used to simulate the online imbalanced label distribution shift is generated using the following sampling strategy: a probability vector $Q_t(y) = [q_1, q_2, \dots, q_K]$ is defined, where t is a time step and T is the total number of steps and is equal to K the total number of classes, and $q_k = q_{max}$ if $k = t$ and $q_k = q_{min} \triangleq (1 - q_{max}) / (K - 1)$ if $k \neq t$. The ratio q_{max} / q_{min} represents the imbalance ratio. At each time step $t \in 1, 2, \dots, T = K$, 100 images are sampled using $Q_t(y)$. So, in total, the newly created dataset is composed of 100×1000 images. An imbalance factor of 500,000 is represented in Fig. 4.3 as ∞ and represents a setup very close to the adaptation of the model one class after the other.

4.4.2 Implementation Details

Methods In this work, we chose to analyze the following tricks and methods: (i) Tent (Wang *et al.*, 2021a) is a seminal work in Fully Test-Time Adaptation and is the first work to use an entropy-based loss in the adaptation process. (ii) SAR (Niu *et al.*, 2023) is a state-of-the-art method in Fully TTA and proposes a method to select the most useful samples based on their

entropy. (iii) Delta (Zhao *et al.*, 2023) is also a state-of-the-art method in Fully TTA and focuses on addressing the problem of online class rebalancing. (iv) in our experimental setup, we call BoT the model combining the best tricks selected in the different experiments. For reproducibility purposes, the links to the code of the methods mentioned are provided in Appx. III-2.

Model architectures In our experiments, we use different architectures depending on the datasets tested. In experiments with ImageNet-C, we follow Niu *et al.* (2023) and use two variants of the ResNet50 architecture and a ViT-Base/16 transformer architecture. ResNet50 is a ResNet model (He *et al.*, 2015) with 50 layers, as explained in the background chapter. The first ResNet50 variant (ResNet50-BN) uses batch normalization layers (Ioffe & Szegedy, 2015) whereas the second one (ResNet50-GN) uses group normalization layers (Wu & He, 2018). In experiments investigating the impact of normalization on model performance, we also include a variant of ResNet50-BN where batch normalization is replaced by batch renormalization (ResNet50-BReN). The ViTBase/16 transformer uses layer normalization (Ba, Kiros & Hinton, 2016) and will be referred to as VitBase-LN. For experiments with VisDA2017, we follow Yang *et al.* (2021a) and Boudiaf *et al.* (2023) and use a ResNet101 architecture. The number of parameters of each architecture is available in Appx. III-1. For reproducibility purposes, we also provide the links to the weights used in our experiments in Appx. III-3.

Evaluation metrics To evaluate the different approaches, we use the classification accuracy metric. To compute this metric, we follow Niu *et al.* (2023) and Zhao *et al.* (2023) and consider the accumulated predictions of the test samples after each model update. In other words, we do not compute the classification accuracy on the whole test set after the model has seen all test samples but online after each batch. Results reported are averaged over 3 runs.

4.5 Results

In this section, we present the results of the experiments investigating the topics presented in Sec.4.3.

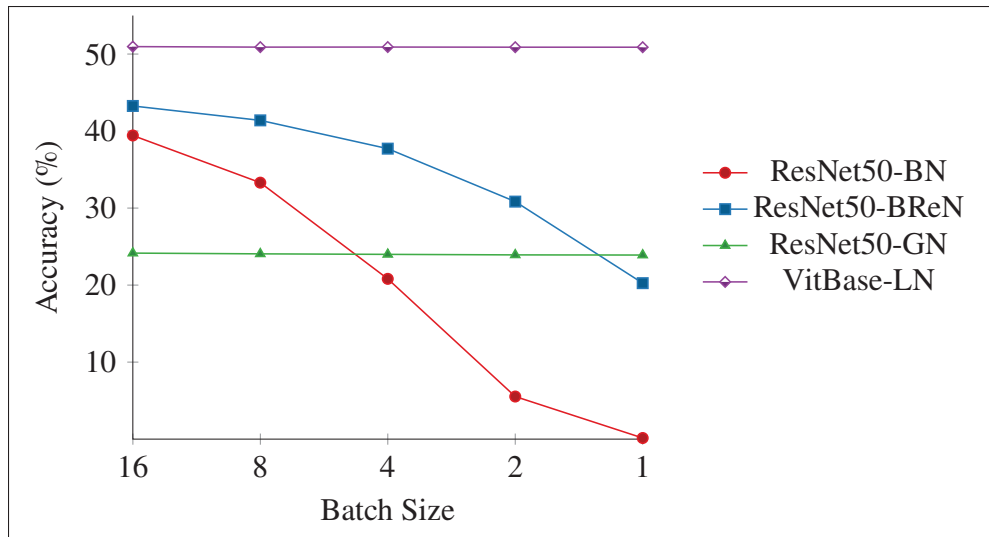


Figure 4.2 Impact of Normalization, Architecture, and Batch Size on classification accuracy of Tent method on ImageNet-C. Using a batch renormalization layer leads to better performance than using a vanilla batch normalization. Tent performance is more stable on architectures with batch-agnostic normalization like group or layer normalization.

4.5.1 Architecture and Normalization

We start by reporting results about the impact of architecture and normalization.

In Fig. 4.2, we observe that the performance of Tent method on a ResNet50-BN architecture is dropping when the batch size is becoming small, with a particularly low performance when the batch size is 2 (5.53% accuracy) or 1 (0.14% accuracy). Intuitively, those results can be explained by the fact that batch normalization layers are normalizing the weights based on the statistics of the current batch. When the batch is becoming too small, the statistics computed have a high variance and are not representative anymore of the test distribution and are not informative enough about the domain shift. However, we can see that using batch renormalization instead of standard batch normalization improves the performance of a ResNet50 model and avoids a complete collapse of the model when the batch size is 1. Also in Fig. 4.2, we observe that Tent performance on architectures with batch-agnostic normalization layers such as GroupNorm or LayerNorm is more stable and less impacted by a reduction of the batch size.

4.5.2 Class Rebalancing

We now continue this section by presenting results related to the class rebalancing problem.

In Fig. 4.3, we can observe the following: i) On ResNet50-BN, the performance of all methods and for all batch sizes is dropping when the imbalance factor is increasing. Batch normalization does not seem to be a suitable normalization method when the test set is unbalanced ii) The performances of Tent and SAR are more stable when the imbalance factor varies on ResNet50-GN. On this architecture, DOT is the most performing method when the batch size is still high and the imbalance factor is still low. However, DOT performance is dropping drastically when the batch size becomes very small or the imbalance factor is very high. iii) Best performances are obtained by the VitBase-LN architecture. Performances are stable for all methods when the imbalance factor increases for a batch size of 16 or 8 but decrease when the imbalance factor increases for lower batch sizes. Our main takeaways from Fig. 4.3 are that group normalization and layer normalization are less sensitive than batch normalization to imbalance classes and that even if DOT and SAR are both performing better than Tent, the sample selection of SAR yields more stable performance in the case of small batch sizes and stronger class imbalance factor.

In Fig. 4.4, we observe that the performance of all methods on ResNet50-BN is dropping when the batch size decreases. On ResNet50-GN and VitBase-LN, the classification accuracy remains stable when the batch size decreases for all models, DOT yielding the best results except when the batch size is equal to 1. This particular case is explained in the next paragraph. Our main takeaways from Fig. 4.4 are that architectures with group or layer normalization are more suitable to handle small batch sizes, and that the class rebalancing method DOT is performing better than the sample selection method SAR for small batch sizes greater than 1.

Single point learning for DOT method In Fig. 4.4, we observe that in the specific case of batch size 1, the performance of DOT drops to the level of Tent. This is because in DOT, the weight of each sample in a batch is normalized by the sum of all weights of this batch. So, when the batch size is 1, the sum of the weights of the batch is equal to the weight of the single sample of the batch. Thus, the normalization of the weight of this single sample by the sum

Table 4.2 Impact of Additional Buffer on Tent accuracy (%) on different architecture on ImageNet-C in the single point learning scenario
An additional buffer of size 2 yields a significant performance improvement. Higher buffer sizes can lead to noisy sample weights and yield no additional improvement on ResNet50-BN or a performance decrease on ResNet50-GN and VitBase-LN.

BatchSize=1	DOT	DOT+buff=2	DOT+buff=4	DOT+buff=8	DOT+buff=16
ResNet50-BN	0.14 \pm 0.00	20.31 \pm 0.02	20.31 \pm 0.02	20.31 \pm 0.02	20.31 \pm 0.02
ResNet50-GN	23.91 \pm 0.60	38.94 \pm 0.03	38.32 \pm 0.06	36.23 \pm 0.03	34.13 \pm 0.02
VitBase-LN	50.89 \pm 0.00	54.15 \pm 0.03	50.56 \pm 0.04	46.39 \pm 0.01	42.13 \pm 0.06

of all weights of the batch gives a weight of 1 and brings back to the same loss formulation as Tent. To address this issue, we propose to approximate the weight of a single sample in this particular case as if it was part of a bigger batch of size N . This approach does not require any additional processing time as we can still infer the class of an input test sample immediately, and it is very cheap in terms of memory as we do not need to save any sample in a queue but just the weights of the N previous samples, which are only scalars. In Tab. 4.2, we analyze the impact of a buffer of different sizes on Tent performance on different architecture when the batch size is 1. We can see that an additional buffer of size 2 yields a significant performance improvement. Higher buffers yield no additional improvement on ResNet50-BN and a performance decrease on ResNet50-Gn and VitBase-LN. We assume that they lead to sample weights that are too noisy.

4.5.3 Sample Selection

In this section, we take a look at results obtained in experiments investigation the sample selection topic.

In Fig. 4.5, we can see that fine-tuning the selection threshold via factor F can lead to a significant increase in the performances in all cases. We also observe that in the case of smaller batch sizes, the optimal value for F is smaller than the value of 0.5 recommended in Niu *et al.* (2023) for a batch size of 64. Moreover, as mentioned in Niu *et al.* (2023), another advantage of this method

is that it requires less computational power to perform the adaptation as fewer samples are used in the optimization. e.g. for the Gaussian noise corruption, severity level 5, on ResNet50-GN and an entropy factor F of 0.4, the model forward passes 50K samples but keep less than 13K after selection for the backward pass, which is only 26% of the whole dataset for this corruption.

4.5.4 Calibration

We now focus our attention on experiments exploring the topic of network calibration in the context of Fully TTA.

To determine the temperature scaling factor in our experiments, we follow *Zhao et al. (2023)* in the way to select hyperparameters using the 4 Imagenet-C validation corruptions. For each network architecture, we select the temperature scaling factor τ for each validation corruption using a grid search on values between 0.5 and 1.5 with a step of 0.1 and keep the average of the 4 values. For the 3 network architectures considered, we obtain a temperature scaling factor of 1.2, which means that without correction, the models are too confident in their predictions.

In Tab. 4.3, we observe that applying temperature scaling during adaptation leads to an increase in Tent performance on ResNet50-BN and ViTBase-LN. On ResNet50-GN, the mean is slightly lower, but the standard deviation is significantly reduced, which means overall a better performance in terms of statistical significance. The performance increase is not very high when using temperature alone. However, we will see in Sec. 4.5.5 that it leads to higher performance when combined with other tricks.

4.5.5 Tricks Combination

We investigate now the performance of Tent when using different combinations of the tricks presented in the previous sections.

For ResNet50-BN, we consider the usage of batch renormalization as an essential trick when dealing with very small batch sizes as presented in Sec.4.3.1 and always integrate it in the

Table 4.3 Impact of Temperature on classification accuracy (%) of Tent method performance on different architecture on ImageNet-C
Using a temperature scaling factor increases the mean accuracy on ResNet50-BN and VitBase-LN. On ResNet50-GN, using temperature decreases slightly the mean classification accuracy but decreases also the standard deviation, which means that the model is better with respect to statistical significance.

	16	8	4	2	1
ResNet50-BN	39.43 \pm 0.13	33.30 \pm 0.04	20.81 \pm 0.08	5.53 \pm 0.01	0.14 \pm 0.00
ResNet50-BN+ temp	39.45 \pm 0.06	33.86 \pm 0.04	20.84 \pm 0.07	6.11 \pm 0.01	0.15 \pm 0.00
ResNet50-GN	24,15 \pm 0.55	24,00 \pm 0.54	23,99 \pm 0.56	23,92 \pm 0.57	23,90 \pm 0.58
ResNet50-GN+ temp	24.01 \pm 0.17	23.87 \pm 0.17	23.82 \pm 0.15	23.76 \pm 0.19	23.74 \pm 0.19
VitBase-LN	50.97 \pm 0.07	50.90 \pm 0.04	50.91 \pm 0.07	50.89 \pm 0.06	50.89 \pm 0.04
VitBase-LN + temp	52.84 \pm 0.27	52.81 \pm 0.26	52.76 \pm 0.26	52.76 \pm 0.20	52.77 \pm 0.22

different tricks combinations tested. In the ResNet50-BN section of Tab. 4.4, we report first the results already presented in Fig.4.2 to see the performance improvement when using batch renormalization. Then we consider all the possible combinations of 2 of the tricks and finally, we consider the combination of all the tricks. For ResNet50-GN and VitBase-LN, we also present results considering all the possible combinations of 2 of the tricks presented previously and then combining all the tricks.

In Tab. 4.4, we observe that when using a ResNet50-BN network, the best pair of tricks is the class rebalancing method DOT combined with the entropy-based sample selection. The best results overall are obtained when using this pair with a temperature scaling factor, in other words when using all tricks together. In this case, compared to Tent, we obtain an average improvement of +17.08% accuracy over all batch sizes. In the case of a ResNet50-GN architecture, the best pair of tricks is class rebalancing combined with the temperature scaling factor. Surprisingly, combining temperature scaling with sample selection is performing better than vanilla Tent but much lower than other pairs of tricks. We assume that as the temperature scaling is changing the entropy of the test samples, a finer tuning of the sample selection margin should be done to ensure that samples useful for the model adaptation are not discarded. The best performances are obtained using all tricks. In this case, we obtain an average improvement of +19.92% accuracy

Table 4.4 Effect of Tricks Combination on model accuracy (%)
 Best results are obtained when combining all tricks and this for the 3 architectures and the different batch sizes considered. Among the different architectures, VitBase-LN has the best classification accuracy in all the different setups.

	Tent +				Batch Size				
	BR	CR	SS	T	16	8	4	2	1
ResNet50-BN					39.40 \pm 0.13	33.30 \pm 0.04	20.81 \pm 0.08	5.53 \pm 0.01	0.14 \pm 0.00
	✓				43.26 \pm 0.01	41.39 \pm 0.06	37.72 \pm 0.05	30.84 \pm 0.04	20.25 \pm 0.01
	✓	✓		✓	45.89 \pm 0.06	43.70 \pm 0.05	39.17 \pm 0.05	31.44 \pm 0.04	20.31 \pm 0.02
	✓		✓	✓	45.17 \pm 0.26	43.03 \pm 0.11	39.02 \pm 0.07	31.60 \pm 0.05	20.26 \pm 0.01
	✓	✓	✓		46.57 \pm 0.07	44.46 \pm 0.01	39.95 \pm 0.01	31.65 \pm 0.01	20.30 \pm 0.02
	✓	✓	✓	✓	46.90 \pm 0.12	44.90 \pm 0.09	40.42 \pm 0.14	32.03 \pm 0.05	20.31 \pm 0.02
ResNet50-GN					24.15 \pm 0.55	24.06 \pm 0.54	23.99 \pm 0.57	23.92 \pm 0.57	23.90 \pm 0.58
		✓		✓	46.35 \pm 0.07	45.89 \pm 0.09	44.77 \pm 0.01	42.07 \pm 0.03	39.31 \pm 0.64
			✓	✓	26.85 \pm 0.17	27.34 \pm 0.55	29.03 \pm 0.59	30.19 \pm 0.20	27.20 \pm 0.48
		✓	✓		45.78 \pm 0.09	45.31 \pm 0.11	44.21 \pm 0.01	41.33 \pm 0.01	38.94 \pm 0.03
		✓	✓	✓	46.50 \pm 0.05	46.07 \pm 0.08	45.02 \pm 0.01	42.32 \pm 0.01	39.70 \pm 0.04
VitBase-LN					50.97 \pm 0.07	50.90 \pm 0.04	50.91 \pm 0.07	50.89 \pm 0.06	50.89 \pm 0.04
		✓		✓	59.26 \pm 0.03	59.20 \pm 0.02	58.97 \pm 0.04	58.52 \pm 0.05	54.68 \pm 0.03
			✓	✓	57.59 \pm 0.44	58.11 \pm 0.14	57.88 \pm 0.09	57.02 \pm 0.10	55.10 \pm 0.07
		✓	✓		59.31 \pm 0.06	59.22 \pm 0.04	58.96 \pm 0.00	57.51 \pm 0.78	54.15 \pm 0.03
		✓	✓	✓	59.80 \pm 0.07	59.77 \pm 0.04	59.59 \pm 0.03	59.04 \pm 0.06	55.15 \pm 0.03

BR=BatchRenorm, T=Temperature, CR=Class Rebalancing, SS=Sample Selection

over all batch sizes compared to Tent. When considering the VitBase-LN architecture, we can see that the two pairs of tricks class rebalancing and temperature and class rebalancing and sample selection are close over all the batch sizes and yield the best results of the pairs of tricks. The overall best results are obtained when combining all tricks. Doing this leads to an average improvement compared to Tent of +7.66% over all batch sizes. Our main takeaway for this series of experiments is that the best results are obtained when combining all tricks (class rebalancing, sample selection, and temperature scaling), and this for the 3 architectures and the different batch sizes considered. Among the different architectures, VitBase-LN has the best classification accuracy when combining all the tricks and on all the batch sizes tested.

Table 4.5 Accuracy (%) on ImageNet-C

BoT obtains better results than Tent and the 2 state-of-the-art methods in all cases. If the performance increase of BoT is not significant on ResNet50-BN (+0.78% accuracy in average versus Delta), it is much more noticeable on ResNet50-GN (+4.31% accuracy in average versus Delta) and ViTBase-LN (+1.53% accuracy in average versus Delta).

	Method	Batch Size				
		16	8	4	2	1
ResNet50-BN	Tent	39.43 \pm 0.13	33.30 \pm 0.04	20.81 \pm 0.08	5.53 \pm 0.01	0.14 \pm 0.00
	SAR	41.02 \pm 0.29	31.10 \pm 0.08	18.90 \pm 0.04	6.78 \pm 0.00	0.14 \pm 0.00
	Delta	46.33 \pm 0.78	43.67 \pm 0.05	39.16 \pm 0.04	31.26 \pm 0.05	20.25 \pm 0.01
	BoT	46.90 \pm 0.1	44.90 \pm 0.09	40.42 \pm 0.14	32.03 \pm 0.05	20.31 \pm 0.02
ResNet50-GN	Tent	24.15 \pm 0.55	24.05 \pm 0.54	23.99 \pm 0.57	23.92 \pm 0.57	23.90 \pm 0.58
	SAR	39.32 \pm 0.17	38.80 \pm 0.14	37.61 \pm 0.39	35.66 \pm 0.28	33.86 \pm 0.06
	Delta	45.22 \pm 0.06	44.70 \pm 0.09	43.47 \pm 0.02	40.77 \pm 0.01	23.91 \pm 0.60
	BoT	46.50 \pm 0.05	46.07 \pm 0.08	45.02 \pm 0.01	42.32 \pm 0.01	39.70 \pm 0.04
ViTBase-LN	Tent	50.97 \pm 0.07	50.90 \pm 0.04	50.91 \pm 0.07	50.90 \pm 0.06	50.89 \pm 0.04
	SAR	56.87 \pm 0.15	56.92 \pm 0.10	56.69 \pm 0.13	55.71 \pm 0.16	53.16 \pm 0.16
	Delta	58.95 \pm 0.05	58.86 \pm 0.04	58.57 \pm 0.03	57.98 \pm 0.04	50.89 \pm 0.04
	BoT	59.80 \pm 0.07	59.77 \pm 0.04	59.59 \pm 0.03	59.04 \pm 0.06	54.68 \pm 0.03

4.5.6 Comparison to other methods and on other datasets

In this final experimental section, we compare the performance of BoT (i.e. Tent with all the tricks presented in this article) to a vanilla Tent and 2 state-of-the-art methods, SAR (Niu *et al.*, 2023) and Delta (Zhao *et al.*, 2023). This comparison is performed on different network architectures and different datasets: ResNet50-BN, ResNet50-GN, ViTBase-LN for ImageNet-C, ImageNet-Rendition and ImageNet-Sketch, and ResNet101 for VisDA2017.

Experimental results In Tab. 4.5, we can see that on the ImageNet-C dataset, BoT obtains better results than a vanilla Tent, and the two state-of-the-art methods for all the batch sizes considered. Interesting to see is the collapse of SAR performance for very small batch sizes (2 and 1) on ResNet50-BN that we do not observe with Delta due to the usage of batch renormalization. If the

performance increase by using all the tricks is not significant on ResNet50-BN (+0.78% accuracy on average versus Delta), it is much more noticeable on ResNet50-GN (+4.31% accuracy on average versus Delta) and ViTBase-LN (+1.53% accuracy in average versus Delta). In Tab. 4.6, we also observe that BoT performs the best in all cases. Interesting to note is that results are more stable over the different batch sizes with ResNet50-GN compared to ResNet50-BN, which is in line with observations from previous experiments. Delta performs better than SAR, but worse than BoT. The performance increase of BoT compared to Delta is similar on ResNet50-BN and ResNet50-GN (respectively +0.85% and +0.87% accuracy) but reaches +1.23% accuracy on ViTBase-LN. In Tab. 4.7, we make the same observations on ImageNet-Sketch as on the other ImageNet variants. ResNet50-BN performance drops when the batch size becomes small. In all cases, Delta performs better than SAR, but not as good as BoT. BoT performs best in all cases. The performance increase of BoT versus Delta is +0.72% accuracy on ResNet50-BN, +1.32% accuracy on ResNet50-GN, and +1.03% accuracy on ViTBase-LN. In Tab. 4.8, we observe that also for the VisDA2017 dataset, results are in line with previous experiments. Delta performs better than Tent and SAR, but not as well as BoT. The performance improvement of BoT versus Delta is +0.36% accuracy on ResNet101.

4.6 Conclusion and Discussion

In this work, we addressed the Fully Test-Time Adaptation problem when dealing with small batch sizes by analyzing the following tricks and methods: i) Usage of Batch renormalization or batch-agnostic normalization ii) Class re-balancing iii) Entropy-based sample selection iv) Temperature scaling. Our experimental results show that if those tricks used alone already bring an improvement in the classification accuracy compared to a vanilla Tent, using them in pairs is even better, and the best results are obtained by combining them all. By doing that, we significantly improve the current state-of-the-art across 4 different image datasets in terms of prediction performances. Furthermore, the selected tricks bring additional benefits concerning the computational load: i) Using group normalization instead of batch normalization in ResNet50 yields more stable results for the same number of total parameters ii) using the

Table 4.6 Accuracy (%) on ImageNet-Rendition

The performance increase of BoT compared to Delta is similar on ResNet50-BN and ResNet50-GN (respectively +0.85% and +0.87% accuracy) but reaches +1.23% accuracy on VitBase-LN.

	Method	Batch Size				
		16	8	4	2	1
ResNet50-BN	Tent	40.80 \pm 0.11	37.75 \pm 0.12	29.70 \pm 0.21	14.24 \pm 0.05	0.56 \pm 0.00
	SAR	42.11 \pm 0.10	38.95 \pm 0.21	30.07 \pm 0.05	16.13 \pm 0.12	0.57 \pm 0.00
	Delta	43.11 \pm 0.15	41.80 \pm 0.23	39.64 \pm 0.16	35.17 \pm 0.06	26.75 \pm 0.01
	BoT	44.68 \pm 0.24	43.12 \pm 0.11	40.61 \pm 0.22	35.55 \pm 0.04	26.75 \pm 0.00
ResNet50-GN	Tent	39.35 \pm 0.16	39.29 \pm 0.18	39.28 \pm 0.19	39.27 \pm 0.18	39.26 \pm 0.18
	SAR	42.94 \pm 0.108	42.75 \pm 0.05	42.28 \pm 0.09	41.75 \pm 0.06	41.84 \pm 0.05
	Delta	43.10 \pm 0.05	43.11 \pm 0.05	42.74 \pm 0.12	41.89 \pm 0.10	42.18 \pm 0.03
	BoT	44.21 \pm 0.06	44.18 \pm 0.10	43.84 \pm 0.20	42.96 \pm 0.16	42.49 \pm 0.08
VitBase-LN	Tent	43.28 \pm 1.04	42.81 \pm 1.04	42.48 \pm 0.87	42.28 \pm 1.05	42.49 \pm 1.32
	SAR	52.72 \pm 0.19	52.59 \pm 0.25	52.20 \pm 0.16	50.92 \pm 0.11	49.95 \pm 0.18
	Delta	53.32 \pm 0.23	53.31 \pm 0.28	53.03 \pm 0.24	52.25 \pm 0.34	49.76 \pm 0.20
	BoT	54.63 \pm 0.18	57.74 \pm 0.19	54.62 \pm 0.25	53.86 \pm 0.28	51.91 \pm 0.15

entropy-based sample selection reduces the computational burden by using fewer samples to adapt the model.

We hope that this study will be useful for the community and that the presented tricks and techniques will be integrated into future baselines and benchmarks.

Table 4.7 Accuracy (%) on ImageNet-Sketch
 BoT performs best in all case. The performance increase of BoT versus Delta is +0.72% accuracy on ResNet50-BN, +1.32% accuracy on ResNet50-GN and +1.03% accuracy on VitBase-LN.

	Method	Batch Size				
		16	8	4	2	1
ResNet50-BN	Tent	27.82 \pm 0.30	22.47 \pm 0.40	10.71 \pm 0.42	2.94 \pm 0.08	0.13 \pm 0.00
	SAR	31.05 \pm 0.29	26.73 \pm 0.20	16.80 \pm 0.07	6.72 \pm 0.05	0.13 \pm 0.00
	Delta	31.92 \pm 0.11	30.36 \pm 0.16	27.32 \pm 0.16	22.56 \pm 0.16	15.58 \pm 0.04
	BoT	33.24 \pm 0.13	31.50 \pm 0.21	28.16 \pm 0.12	22.86 \pm 0.16	15.58 \pm 0.04
ResNet50-GN	Tent	23.04 \pm 0.40	22.95 \pm 0.38	22.93 \pm 0.38	22.92 \pm 0.38	22.92 \pm 0.35
	SAR	32.11 \pm 0.50	32.26 \pm 0.07	31.89 \pm 0.16	31.16 \pm 0.20	31.64 \pm 0.25
	Delta	34.50 \pm 0.20	34.26 \pm 0.09	33.57 \pm 0.18	31.56 \pm 0.08	30.93 \pm 0.07
	BoT	35.77 \pm 0.03	35.49 \pm 0.19	34.91 \pm 0.15	33.19 \pm 0.10	32.07 \pm 0.09
VitBase-LN	Tent	5.83 \pm 0.32	5.69 \pm 0.43	5.59 \pm 0.44	5.38 \pm 0.28	5.51 \pm 0.49
	SAR	25.40 \pm 0.65	25.88 \pm 0.64	27.87 \pm 0.08	32.89 \pm 0.57	30.68 \pm 0.99
	Delta	38.67 \pm 0.08	38.50 \pm 0.08	38.18 \pm 0.11	37.18 \pm 0.14	33.90 \pm 0.08
	BoT	39.69 \pm 0.06	39.68 \pm 0.06	39.50 \pm 0.09	38.64 \pm 0.03	34.09 \pm 0.10

Table 4.8 Accuracy (%) on VisDA2017
 Delta performs better than Tent and SAR but not as good as BoT. The performance improvement of BoT versus Delta is +0.36% accuracy and +4.75% versus Tent on ResNet101.

	Method	Batch Size				
		16	8	4	2	1
ResNet101	Tent	65.30 \pm 0.08	64.65 \pm 0.18	63.47 \pm 0.12	58.89 \pm 0.33	49.10 \pm 0.04
	SAR	63.08 \pm 0.03	57.47 \pm 0.05	46.20 \pm 0.09	24.81 \pm 0.16	18.63 \pm 0.01
	Delta	73.20 \pm 0.08	71.52 \pm 0.12	68.16 \pm 0.11	61.41 \pm 0.20	49.08 \pm 0.03
	BoT	73.54 \pm 0.09	71.70 \pm 0.07	68.17 \pm 0.19	61.49 \pm 0.10	50.28 \pm 0.09

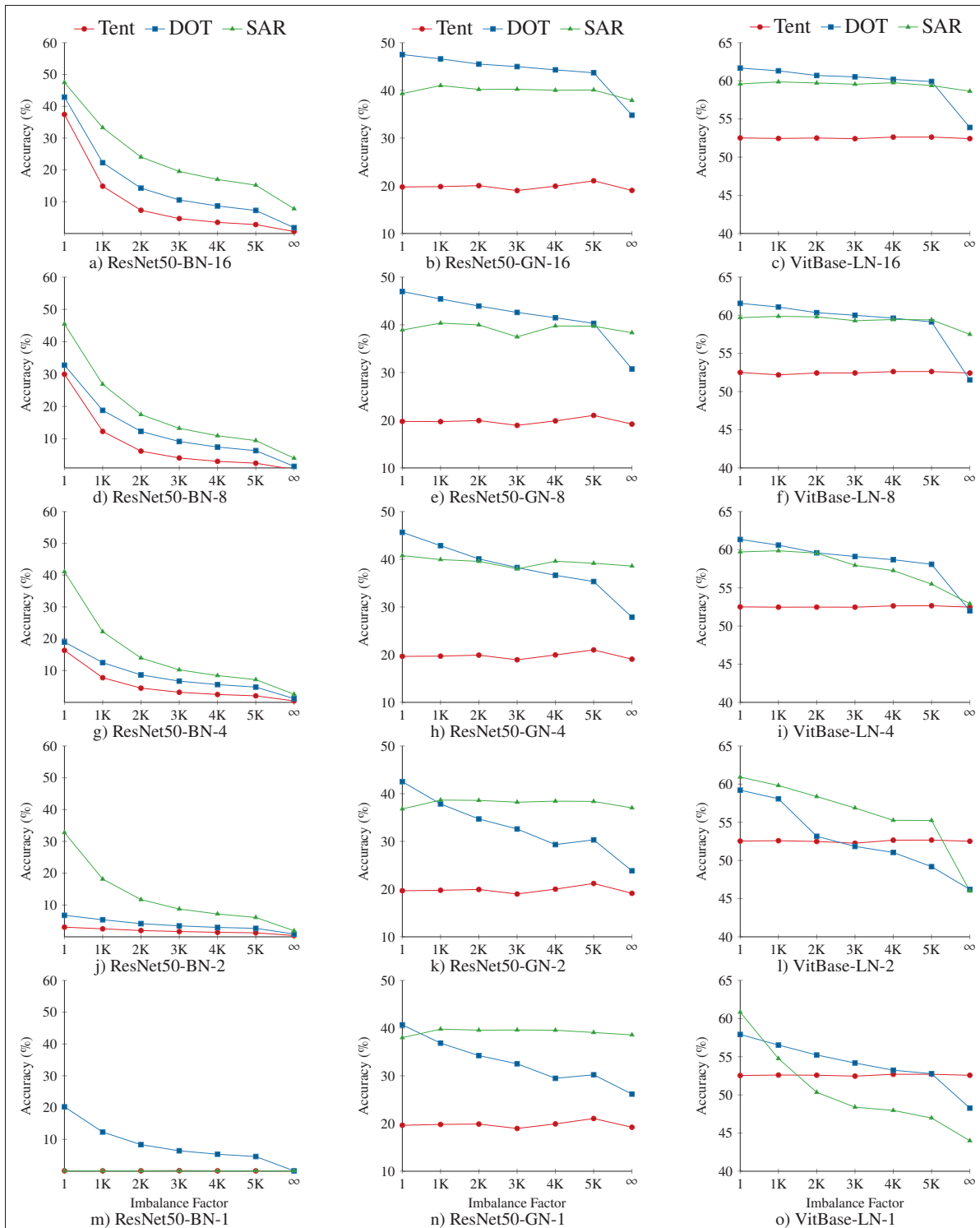


Figure 4.3 Impact of Imbalance Factor, Architecture, and Batch Size

on classification accuracy (%) of different methods on ImageNet-C

On ResNet50-BN, the performance of all models decreases when the imbalance factor increases. On ResNet50-GN, DOT, and SAR are more efficient than Tent, but SAR is more stable with very small batch sizes and stronger imbalance factors. On VitBase-LN, Tent performs lower than DOT and SAR with a batch size 4 and a moderate imbalance factor.

However, DOT and SAR performance is dropping significantly for small batch sizes and strong imbalance factors. The number after the architecture in the legend is the batch size.

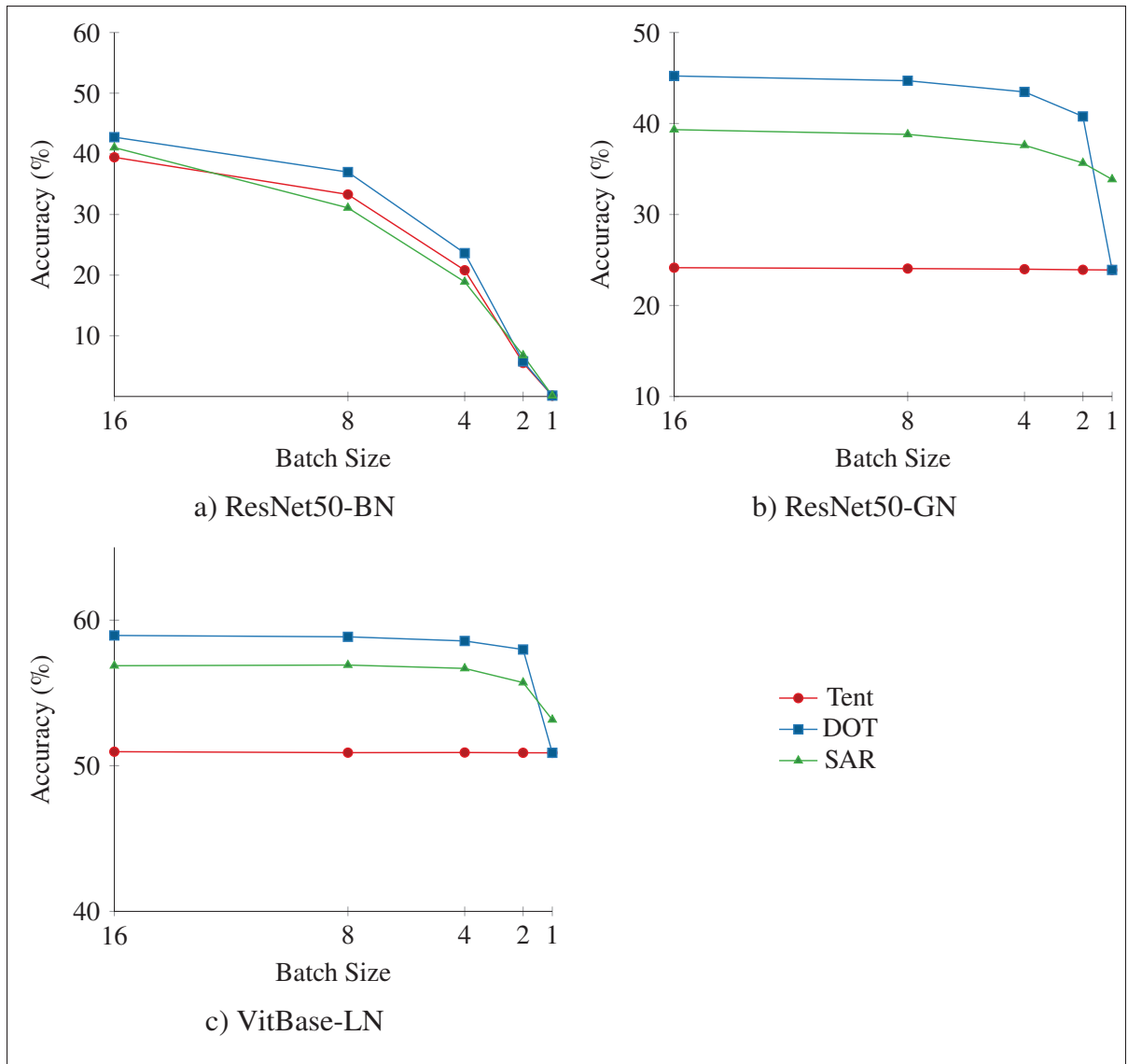


Figure 4.4 Impact of Architecture and Batch Size on the classification accuracy (%) of different methods on ImageNet-C. Batch-agnostic normalizations like group or layer normalization are more suitable to handle small batch sizes. Moreover, in this scenario, the class rebalancing method DOT is performing better than the sample selection method of SAR.

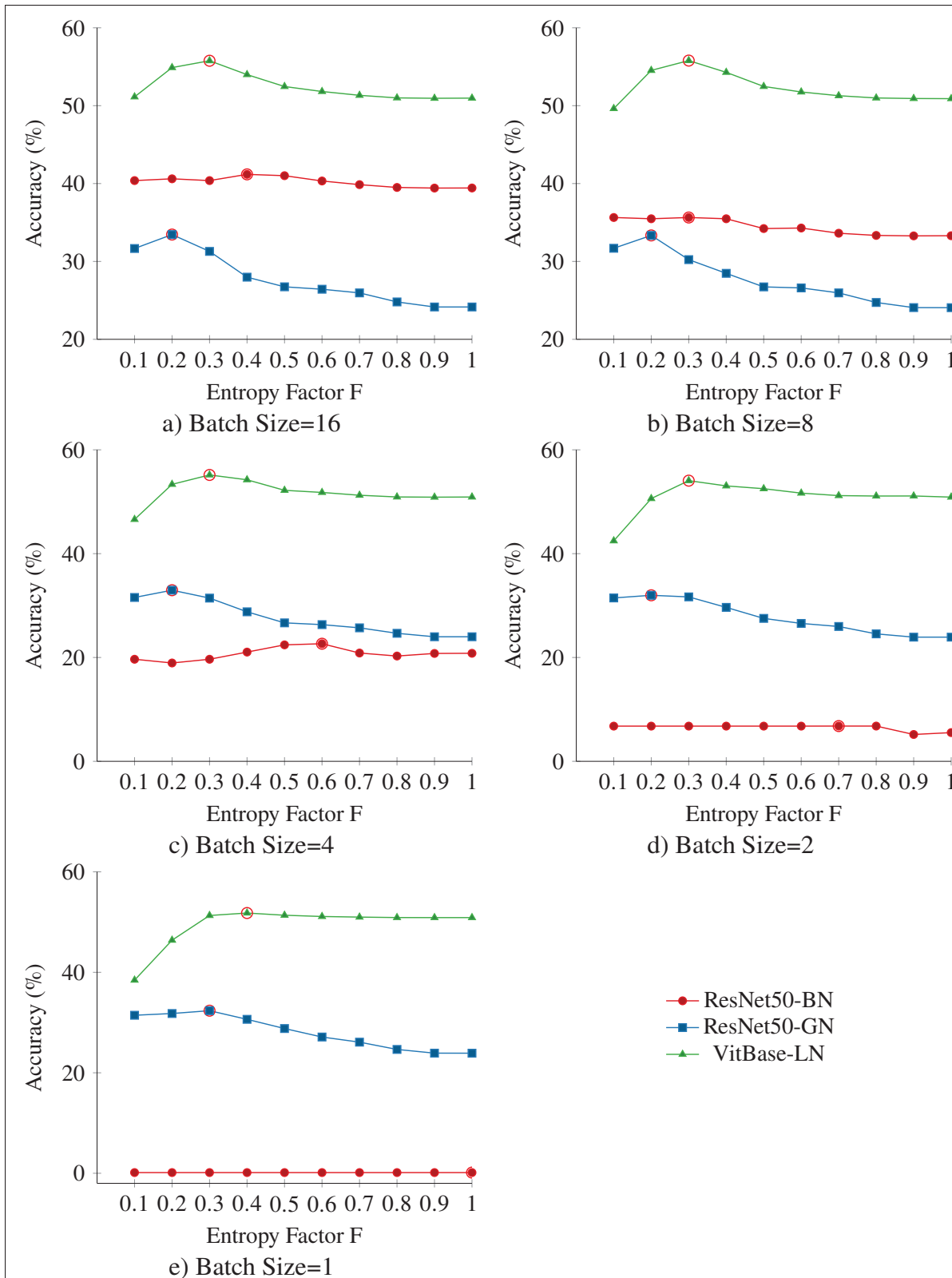


Figure 4.5 Impact of Sample Selection and Architecture on classification accuracy of different methods on ImageNet-C

The best results are circled in red. The optimal threshold varies in function of the architecture and the batch size and is lower for the smaller batch sizes than the values 0.5 or 0.4 for a batch size of 64 recommended in (Niu *et al.*, 2022).

CONCLUSION AND RECOMMENDATIONS

This thesis explored important challenges occurring when learning deep learning-based visual recognition models with limited data. In the previous sections, we started by introducing the research problem and the research directions, then presented the solutions we proposed to tackle the research challenges. In this section, we summarize the research contributions and the limitations of our work, and finally, we propose two possible directions for future work.

5.1 Summary of contributions

In Chapter 1, we presented the research problem, first by giving an overview of the technical blocks necessary to understand the following chapters, and then by introducing the challenges induced by data scarcity and distribution shift when training a visual recognition model. In the following chapters, we presented three works addressing those challenges.

In Chapter 2, we explored first the usage of generative models as a mean to extend a dataset by generating additional images useful to train an image classifier and increase its robustness and performance. More particularly, we leveraged the power of generative adversarial networks (GAN) to generate new augmented samples. Unlike traditional heuristic transformations, the approach presented learns data augmentation directly from training data using an encoder-decoder architecture and a spatial transformer network, producing more complex samples within the same class.

Taking a step back and looking at the broader picture, GANs were a promising direction in the generative models field, but they were recently replaced by diffusion models, mainly due to the difficulty of their training and their inability to scale to high-resolution images. Moreover, recent studies tend to indicate that including generated images in large-scale datasets to train image classifiers might not be the way to go, as it can lead to a decrease in the model performance.

In Chapter 3, we proposed an efficient approach to reduce the computational resources needed to determine the best data augmentation when training an image classifier. We optimized augmentation parameters using a validation set through bi-level optimization in a model trained end-to-end. Our method improved generalization while removing the need for domain knowledge or an expensive external validation loop. Our method is particularly relevant in cases where defining a set of possible transformations is not trivial. Indeed, we validated our model on natural images, but also obtained good performances on histological images, where defining a set of potentially useful transformations requires expert knowledge in the medical field.

Finally, in Chapter 4, we explored Fully test-time adaptation (TTA) and presented a categorization of selected orthogonal TTA techniques interesting for adapting models to data drifts, such as small batch normalization, stream rebalancing, reliable sample selection, and network confidence calibration. We gave insights into their impact on different scenarios, highlighting trade-offs in accuracy, computational power, and model complexity, while also revealing the synergies that arise from combining techniques. The outcome of this study and the related work associated are very interesting as it shows that even if test-time adaptation methods are progressing rapidly, they still rely on strong experimental assumptions that can not be guaranteed in real-life scenarios, like a minimum batch size or a balanced input data stream, thus proving the necessity for further research to close the gap between research and real-life applications.

5.2 Limitations of our work

In the previous section, we summarized the contributions of this thesis. In this section, we discuss the limitations of the proposed methods.

5.2.1 Automatic Data Augmentation Learning

In Chapters 2 and 3, we proposed methods to learn automatically the best data augmentation transformations to train an image classifier. Our experiments showed that those approaches led to an improved classification accuracy for natural images and for histological images. However, they are not free from weaknesses.

In Chapter 2, our proposed method optimizes the data augmentation parameters on the training set, which can be suboptimal, as we want to learn the data augmentation parameters that will yield the best performance on a validation set. This problem was addressed in Chapter 3, where we proposed a method based on a bilevel optimization framework, optimizing the data augmentation parameters using the validation loss.

Moreover, the learned transformations are only differentiable transformations. This limits the range of transformations applied to the dataset available. Further research should be done to integrate non-differentiable transformations by using for example gradient estimators.

5.2.2 Test-Time Adaptation

In Chapter 4, we ran an empirical evaluation of recently proposed Fully Test-Time Adaptation techniques. Our experiments showed that fine-tuning them correctly and using them in combination led to an interesting increase of the classification accuracy after adaptation. However, we identified two main limitations.

First, the techniques investigated are applied to handle a distribution shift. Yet, mechanisms to detect the distribution shift must be first put in place, to know when to start and when to stop using those methods, which might not be trivial.

Moreover, some techniques yield good results but require a few labels to fine-tune their hyperparameters. This can be problematic in a fully online mode, where collecting labels might

not be possible or where a model in production mode might not be put offline to update the adaptation hyperparameters.

5.3 Future work

In this section, we propose two possible directions for future work.

5.3.1 Data Augmentation Transformations Sampling

By taking a broader look at the recent data augmentation literature, we can see that if automatic data augmentation learning has been an active area of research, different stochastic augmentation methods have recently emerged. Those methods aim at defining data augmentation parameters more efficiently by simply sampling some transformations and their magnitude from a big pool of predefined transformations. Seminal work RandAugment (Cubuk *et al.*, 2019b) reached state-of-the-art results without any expensive sequence search like in AutoAugment (Cubuk *et al.*, 2019a) or data augmentation hyperparameter tuning. Follow-up works of RandAugment like TrivialAugment (Müller & Hutter, 2021) went even further by showing that simple strategies like sampling only one transformation from a pool instead of a sequence and sampling its magnitude are enough to obtain good results. Sampling data augmentation transformations instead of learning them seems more efficient. However, the reasons behind this improvement are not clear. Further research is needed to understand the impact of the sampled transformations on the trained model and to check if sampling-based approaches would still perform well in the case of a very limited amount of training data.

5.3.2 Further exploration of Foundation Models

In Chapter 3, we saw that adapting a pretrained model at test time can help to perform a task on a shifted data distribution. Recent trends have shown the emergence of Foundations

models (Bommasani *et al.*, 2021), which are models pre-trained on vast, heterogeneous datasets and acting as a general-purpose starting point for a wide array of downstream tasks. They provide strong discriminative features that allow them to be adapted ad-hoc from a very limited amount of supervision, thus reducing the need to collect a large amount of annotated data. If using those models is appealing, what they already learned is not clear. Further research is needed to open the Foundation models "black boxes" to collect insights on the knowledge already acquired, and determine what could still be needed to perform a task more efficiently. In the context of data augmentation, some recent work (Basu *et al.*, 2022) tried for instance to analyze to which transformations Foundation models were already invariant or equivariant to and proposed methods to extend this to a broader range of transformations.

In summary, this thesis provides a substantive contribution to the domain of deep learning by addressing some of its key data-related challenges. By proposing solutions for efficient data augmentation, and leveraging test-time adaptation, we aimed to offer interesting and useful starting points for the research community for further exploration. Indeed, further research is encouraged to improve the robustness and applicability of deep learning models in real-world scenarios.

APPENDIX I

SUPPLEMENTARY MATERIAL FOR CHAPTER 2 ADVERSARIAL LEARNING OF GENERAL TRANSFORMATIONS FOR DATA AUGMENTATION

1. Implementation details

Model architecture

Table-A I-1 Details of D^C network

Discriminator D^C	
Input 32x32 Image	Input One-hot class representation
3x3 conv. 48 LReLU(0.2)	32x32 deconv. 48 LReLU(0.2)
3x3 conv. 96 LReLU(0.2)	
3x3 conv. 96 LReLU(0.2), 0.5 dropout	
3x3 conv. 192 LReLU(0.2)	
3x3 conv. 192 LReLU(0.2)	
3x3 conv. 192 LReLU(0.2), 0.5 dropout	
3x3 conv. 192 LReLU(0.2)	
1x1 conv. 192 LReLU(0.2)	
1x1 conv. 192 LReLU(0.2), 0.5 dropout	
MLP 1 unit, sigmoid	

Table-A I-2 Details of D^D network

Discriminator D^D	
Input 32x32 Image	Input 32x32 Image
3x3 conv. 48 LReLU(0.2)	3x3 conv. 48 LReLU(0.2)
3x3 conv. 96 LReLU(0.2)	
3x3 conv. 96 LReLU(0.2), 0.5 dropout	
3x3 conv. 192 LReLU(0.2)	
3x3 conv. 192 LReLU(0.2)	
3x3 conv. 192 LReLU(0.2), 0.5 dropout	
3x3 conv. 192 LReLU(0.2)	
1x1 conv. 192 LReLU(0.2)	
1x1 conv. 192 LReLU(0.2), 0.5 dropout	
MLP 1 unit, sigmoid	

Table-A I-3 Details of G network

Generator G	
Input 32x32 Image	100-dim noise vector
3x3 conv. 32, batchNorm, LReLU(0.2)	-
3x3 conv. 32, batchNorm, LReLU(0.2)	32x32 deconv. 32, LReLU(0.2)
Down STN	
2x2 max-pooling	
3x3 conv. 64, batchNorm, LReLU(0.2)	
3x3 conv. 128, batchNorm, LReLU(0.2)	
2x2 max-pooling	
3x3 conv. 256, batchNorm, LReLU(0.2)	
3x3 conv. 256, batchNorm, LReLU(0.2)	
2x2 max-pooling	
3x3 conv. 512, batchNorm, LReLU(0.2)	
3x3 conv. 512, batchNorm, LReLU(0.2)	
2x2 max-pooling	
3x3 conv. 1024, batchNorm, LReLU(0.2)	
3x3 conv. 1024, batchNorm, LReLU(0.2)	
MLP 32 unit, ReLU	
MLP 6 unit	
Down U-Net	
2x2 max-pooling	
3x3 conv. 64, batchNorm, LReLU(0.2)	
3x3 conv. 128, batchNorm, LReLU(0.2)	
2x2 max-pooling	
3x3 conv. 256, batchNorm, LReLU(0.2)	
3x3 conv. 256, batchNorm, LReLU(0.2)	
2x2 max-pooling	
3x3 conv. 512, batchNorm, LReLU(0.2)	
3x3 conv. 512, batchNorm, LReLU(0.2)	
2x2 max-pooling	
3x3 conv. 1024, batchNorm, LReLU(0.2)	
3x3 conv. 1024, batchNorm, LReLU(0.2)	
Up U-Net	
3x3 conv. 512, batchNorm, LReLU(0.2)	
3x3 conv. 512, batchNorm, LReLU(0.2)	
3x3 conv. 256, batchNorm, LReLU(0.2)	
3x3 conv. 256, batchNorm, LReLU(0.2)	
3x3 conv. 128, batchNorm, LReLU(0.2)	
3x3 conv. 128, batchNorm, LReLU(0.2)	
3x3 conv. 64, batchNorm, LReLU(0.2)	
3x3 conv. 64, batchNorm, LReLU(0.2)	
1x1 conv. (3 for color, 1 for grayscale), batchNorm, LReLU(0.2)	

Table-A I-4 Details of C network

Classifier C
Input 32x32 Image
3x3 conv. 96 LReLU(0.2)
3x3 conv. 96 LReLU(0.2)
3x3 conv. 96 LReLU(0.2), 0.5 dropout
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2), 0.5 dropout
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2)
MLP 10 unit, sigmoid
10-class Softmax

Training parameters

To train our model, we use following values for the optimization parameters.

Generator: We use Adam as optimizer with a initial learning rate of 0.0005, a β_1 value of 0.5 and a β_2 value of 0.999.

Class Discriminator: We use Adam as optimizer with a initial learning rate of 0.0005, a β_1 value of 0.5 and a β_2 value of 0.999. As balance factor (see Sec. 2.3), we use as value for α 0.1 for MNIST, 1 for SVHN and 0.1 for CIFAR10.

Similarity Discriminator: We use Adam as optimizer with a initial learning rate of 0.0005, a β_1 value of 0.5 and a β_2 value of 0.999. As balance factor (see Sec. 2.3), we use as value for β 0.05 for MNIST, 1 for SVHN and 0.05 for CIFAR10.

Classifier: We use Adam as optimizer with a initial learning rate of 0.006, a β_1 value of 0.5 and a β_2 value of 0.999. As balance factor (see Sec. 2.3), we use as value for γ 0.005 for MNIST, 0.0005 for SVHN and 0.001 for CIFAR10.

Training algorithm

We train our model according to algorithm I-1.

Algorithm-A I-1 Minibatch stochastic gradient descent training of our model - Fully supervised

1 **for** number of training iterations **do**

2 Sample a first minibatch $batch_i$ of size m of labeled data $(x_i, y_i) \sim p_{data}$

3 Sample a minibatch $batch_{noise}$ of size m of noise $z \sim p_g(z)$

4 Compute \mathcal{L}_{D^C} and update D^C by descending along its stochastic gradient:

$$\begin{aligned} \mathcal{L}_{D^C} = & -\mathbb{E}_{x_i, y_i \sim p_{data}} [\log (D^C(x_i, y_i))] \\ & -\mathbb{E}_{x_i, y_i \sim p_{data}, z \sim p_z} [\log (1 - D^C(G(x_i, z), y_i))] \end{aligned}$$

5 Sample a second minibatch $batch_k$ of size m of labeled data $(x_k, y_k) \sim p_{data}$ with the same labels as $batch_i$

6 Compute \mathcal{L}_{D^D} and update D^D by descending along its stochastic gradient:

$$\begin{aligned} \mathcal{L}_{D^D} = & -\mathbb{E}_{x_i, x_j \sim p_{data}} [\log (D^D(x_i, x_j))] \\ & -\mathbb{E}_{x_i \sim p_{data}, z \sim p_z} [\log (1 - D^D(x_i, G(x_i, z)))] \end{aligned}$$

7 Compute \mathcal{L}_G and update G by descending along its stochastic gradient:

$$\begin{aligned} \mathcal{L}_G = & -\alpha \mathbb{E}_{x_i, y_i \sim p_{data}, z \sim p_z} [\log (D^C(G(x_i, z), y_i))] \\ & -\beta \mathbb{E}_{x_i \sim p_{data}, z \sim p_z} [\log (D^D(x_i, G(x_i, z)))] \\ & -\gamma \mathbb{E}_{x_i, y_i \sim p_{data}} [\log (1 - C_{y_i}(G(x_i, z)))] , \end{aligned}$$

8 Compute \mathcal{L}_C and update C by descending along its stochastic gradient:

$$\begin{aligned} \mathcal{L}_C = & -\mathbb{E}_{x_i, y_i \sim p_{data}} [\log (C_{y_i}(x_i))] \\ & -\mathbb{E}_{x_i, y_i \sim p_{data}, z \sim p_z} [\log (C_{y_i}(G(x_i, z)))] \end{aligned}$$

9 **end for**

2. Reproducibility

Table-A I-5 Links to the source code

Method	Code Link
Mounsaveng et al. (2019)	https://github.com/smounsav/DATGAN

APPENDIX II

SUPPLEMENTARY MATERIAL FOR CHAPTER 3 LEARNING DATA AUGMENTATION WITH ONLINE BILEVEL OPTIMIZATION FOR IMAGE CLASSIFICATION

1. Implementation details

Model architecture

Table-A II-1 BadGAN classifier network

Classifier C
Input 32x32 Image
3x3 conv. 96 LReLU(0.2)
3x3 conv. 96 LReLU(0.2)
3x3 conv. 96 LReLU(0.2), 0.5 dropout
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2), 0.5 dropout
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2)
3x3 conv. 192 LReLU(0.2)
MLP 10 unit, sigmoid
10-class Softmax

Table-A II-2 Augmenter network for affine and color transformations For *Small*, n is the number of parameters to learn (6 for affine, 4 for color and 10 when combining both).

Augmenter A		
<i>Small</i>	<i>Medium</i>	<i>Large</i>
Input n^* dim.	Input 100 dim.	Input 100 dim.
MLP n units	MLP 64 unit	MLP 512 unit
relu, 0.2 dropout	relu, 0.2 dropout	relu, 0.2 dropout
MLP 10 x n units	MLP 32 unit	MLP 1024 unit
relu, 0.2 dropout	relu, 0.2 dropout	relu, 0.2 dropout
-	-	MLP 1024 unit
-	-	relu, 0.2 dropout
-	-	MLP 512 unit
-	-	relu, 0.2 dropout
MLP n units, tanh		

Table-A II-3 Transformations considered by our adapted RandAugment framework To be fair in the comparison with our proposed model, we limited the set of transformations to the transformations learned by our model.

Transformation type	Magnitude Range
identity	-
rotation	[-30.0, 30.0]
translation x	[-0.45, 0.45]
translation y	[-0.45, 0.45]
shear x	[-0.3, 0.3]
shear y	[-0.3, 0.3]
contrast	[0, 2]
brightness	[-1, 1]
hue	[-0.5, 0.5]
Saturation	[0, 1]

Table-A II-4 M and N hyperparameters used for the RandAugment based model in Tab. 3.8.

	BACH	Glas	Larynx 20x	Larynx 40x	Brain 20x	Brain 40x
(M,N) hyperparameters	3,2	3,2	4,2	4,2	3,3	3,3

2. Reproducibility

Table-A II-5 Links to the source code

Method	Code Link
Mounsaveng et al. (2021)	https://github.com/ElementAI/bilevel_augment
MELBA Submission	https://github.com/smounsav/bilevel_augment_histo

APPENDIX III

SUPPLEMENTARY MATERIAL FOR CHAPTER 4 BAG OF TRICKS FOR FULL TEST-TIME ADAPTATION

1. Implementation details

Table-A III-1 Number of parameters of each architecture used in our experimental setup

Architecture	Number of parameter
ResNet50-BN	25M
ResNet50-GN	25M
ResNet-101	43M
VitBase-LN	86M

2. Reproducibility

Table-A III-2 Links to the source code
of the methods mentioned in our work

Method	Code Link
Tent (Wang <i>et al.</i> , 2021a)	https://github.com/DequanWang/tent
SAR (Niu <i>et al.</i> , 2023)	https://github.com/mr-eggplant/SAR
Delta (Zhao <i>et al.</i> , 2023)	https://github.com/bwbwzhao/DELTA

Table-A III-3 Links to the weights of the pretrained models mentioned in the paper

Architecture	Code Link
ResNet50-BN	https://download.pytorch.org/models/resnet50-9c8e357.pth (Paszke <i>et al.</i> , 2019)
ResNet50-GN	timm (Wightman, 2019)
ResNet-101	https://github.com/Albert0147/NRC_SFDA (Yang <i>et al.</i> , 2021a)
VitBase-LN	timm (Wightman, 2019)

BIBLIOGRAPHY

- Antoniou, A., Storkey, A. & Edwards, H. (2018). Augmenting Image Classifiers Using Data Augmentation Generative Adversarial Networks. *International Conference on Artificial Neural Networks (ICANN)*.
- Aresta, G., Araújo, T., Kwok, S., Chennamsetty, S. S., MohammedSafwanK., P., Varghese, A., Marami, B., Prastawa, M., Chan, M., Donovan, M. J., Fernandez, G., Zeineh, J., Kohl, M., Walz, C., Ludwig, F., Braunewell, S., Baust, M., Vu, Q. D., To, M. N. N., Kim, E., Kwak, J. T., Galal, S., Sanchez-Freire, V., Brancati, N., Frucci, M., Riccio, D., Wang, Y., Sun, L., Ma, K., Fang, J., Koné, I., Boulmane, L., Campilho, A., Eloy, C., Polónia, A. & Aguiar, P. (2019). BACH: Grand Challenge on Breast Cancer Histology Images. *Medical Image Analysis (MedIA)*, 56, 122–139.
- Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein Generative Adversarial Networks. *International Conference on Machine Learning (ICML)*.
- Ataky, S. T. M., De Matos, J., Britto, A. d. S., Oliveira, L. E. & Koerich, A. L. (2020). Data augmentation for histopathological images based on gaussian-laplacian pyramid blending. *International joint conference on neural networks (IJCNN)*.
- Ba, J., Kiros, J. R. & Hinton, G. E. (2016). Layer Normalization. *arXiv preprint arXiv: 1607.06450*.
- Bartler, A., Bühler, A., Wiewel, F., Döbler, M. & Yang, B. (2022). MT3: Meta Test-Time Training for Self-Supervised Test-Time Adaption. *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Basu, S., Sattigeri, P., Ramamurthy, K. N., Chenthamarakshan, V., Varshney, K. R., Varshney, L. R. & Das, P. (2022). Equi-Tuning: Group Equivariant Fine-Tuning of Pretrained Models. *Conference on Artificial Intelligence (AAAI)*.
- Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural computation*, 12(8), 1889-1900.
- Bengio, Y., Louradour, J., Collobert, R. & Weston, J. (2009). Curriculum learning. *International Conference on Machine Learning (ICML)*.
- Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *Journal of Machine Learning Research (JMLR)*, 13(10), 281–305.

- Bergstra, J., Yamins, D. & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Journal of Machine Learning Research (JMLR)*, 28(1), 115–123.
- Bergstra, J. S., Bardenet, R., Bengio, Y. & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Bertrand, Q., Klopfenstein, Q., Blondel, M., Vaiter, S., Gramfort, A. & Salmon, J. (2020). Implicit differentiation of Lasso-type models for hyperparameter optimization. *International Conference on Machine Learning (ICML)*.
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E. et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv: 2108.07258*.
- Boudiaf, M., Mueller, R., Ayed, I. B. & Bertinetto, L. (2022). Parameter-free Online Test-time Adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Boudiaf, M., Denton, T., van Merriënboer, B., Dumoulin, V. & Triantafillou, E. (2023). In Search for a Generalizable Method for Source Free Domain Adaptation. *International Conference on Machine Learning (ICML)*.
- Brock, A., Donahue, J. & Simonyan, K. (2019). Large Scale GAN Training for High Fidelity Natural Image Synthesis. *International Conference on Learning Representations (ICLR)*.
- Chang, J.-R., Wu, M.-S., Yu, W.-H., Chen, C.-C., Yang, C.-K., Lin, Y.-Y. & Yeh, C.-Y. (2021). Stain Mix-Up: Unsupervised Domain Generalization for Histopathology Images. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*.
- Chen, C. S., Chen, X., Ma, C., Liu, Z. & Liu, X. (2022). Gradient-based Bi-level Optimization for Deep Learning: A Survey. *arXiv preprint arXiv: 2207.11719*.
- Chen, S., Dobriban, E. & Lee, J. H. (2020). A group-theoretic framework for data augmentation. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Chidlovskii, B., Clinchant, S. & Csurka, G. (2016). Domain Adaptation in the Absence of Source Domain Data. *International Conference on Knowledge Discovery and Data Mining*.
- Cho, M., Finkler, U., Kumar, S., Kung, D. S., Saxena, V. & Sreedhar, D. (2017). PowerAI DDL. *arXiv preprint arXiv: 1708.02188*.

- Chongxuan, L., Xu, T., Zhu, J. & Zhang, B. (2017). Triple generative adversarial nets. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Ciampi, F., Geessink, O. G. F., Bejnordi, B. E., de Souza, G. S., Baidoshvili, A., Litjens, G. J. S., van Ginneken, B., Nagtegaal, I. D. & van der Laak, J. (2017). The importance of stain normalization in colorectal tissue classification with convolutional networks. *International Symposium on Biomedical Imaging (ISBI)*.
- Ciresan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. (2012). Deep Big Multilayer Perceptrons For Digit Recognition. In *Neural Networks Tricks of the Trade: Second Edition* (pp. 581–598). Springer Berlin Heidelberg.
- Cohen, T., Weiler, M., Kicanaoglu, B. & Welling, M. (2019). Gauge Equivariant Convolutional Networks and the Icosahedral CNN. *International Conference on Machine Learning (ICML)*.
- Cohen, T. & Welling, M. (2016). Group Equivariant Convolutional Networks. *International Conference on Machine Learning (ICML)*.
- Colson, B., Marcotte, P. & Savard, G. (2007). An overview of bilevel optimization. *Annals of Operations Research*, 153, 235-256.
- Csurka, G. (2017). A Comprehensive Survey on Domain Adaptation for Visual Applications. In *Domain Adaptation in Computer Vision Applications* (pp. 1–35). Springer International Publishing.
- Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. & Le, Q. V. (2019a). AutoAugment: Learning Augmentation Strategies From Data. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Cubuk, E. D., Zoph, B., Shlens, J. & Le, Q. V. (2019b). Randaugment: Practical automated data augmentation with a reduced search space. *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.
- Cui, Y., Jia, M., Lin, T.-Y., Song, Y. & Belongie, S. J. (2019). Class-Balanced Loss Based on Effective Number of Samples. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Dai, Z., Yang, Z., Yang, F., Cohen, W. W. & Salakhutdinov, R. R. (2017). Good semi-supervised learning that requires a bad gan. *Conference on Neural Information Processing Systems (NeurIPS)*.

- de Matos, J., Ataky, S. T. M., de Souza Britto, A., Soares de Oliveira, L. E. & Lameiras Koerich, A. (2021). Machine Learning Methods for Histopathological Image Analysis: A Review. *Electronics*, 10(5).
- Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme Ruiz, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., Steenkiste, S. V., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M., Gritsenko, A. A., Birodkar, V., Vasconcelos, C. N., Tay, Y., Mensink, T., Kolesnikov, A., Pavetic, F., Tran, D., Kipf, T., Lucic, M., Zhai, X., Keysers, D., Harmsen, J. J. & Houlsby, N. (2023). Scaling Vision Transformers to 22 Billion Parameters. *International Conference on Machine Learning (ICML)*.
- Dertat, A. (2017). Applied Deep Learning - Part 4: Convolutional Neural Networks. Retrieved from: <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>.
- DeVries, T. & Taylor, G. (2017a). Dataset Augmentation in Feature Space. *International Conference on Learning Representations (ICLR) Workshop Track*.
- DeVries, T. & Taylor, G. W. (2017b). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.
- Dhariwal, P. & Nichol, A. Q. (2021). Diffusion Models Beat GANs on Image Synthesis. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Domke, J. (2012). Generic methods for optimization-based modeling. *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D. & Guo, B. (2022). CSWin Transformer: A General Vision Transformer Backbone With Cross-Shaped Windows. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations (ICLR)*.
- Esteves, C., Allen-Blanchette, C., Zhou, X. & Daniilidis, K. (2018). Polar Transformer Networks. *International Conference on Learning Representations (ICLR)*.
- Faryna, K., van der Laak, J. & Litjens, G. (2021). Tailoring automated data augmentation to H&E-stained histopathology. *Medical Imaging with Deep Learning (MIDL)*.

- Finn, C., Abbeel, P. & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*.
- Franceschi, L., Donini, M., Frasconi, P. & Pontil, M. (2017). Forward and Reverse Gradient-Based Hyperparameter Optimization. *International Conference on Machine Learning (ICML)*.
- Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R. & Pontil, M. (2018). Bilevel Programming for Hyperparameter Optimization and Meta-Learning. *International Conference on Machine Learning (ICML)*.
- Frid-Adar, M., Klang, E., Amitai, M. M., Goldberger, J. & Greenspan, H. (2018). Synthetic data augmentation using GAN for improved liver lesion classification. *International Symposium on Biomedical Imaging (ISBI)*.
- Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36, 193-202.
- Ganin, Y. & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. *International Conference on Machine Learning (ICML)*.
- Garcea, F., Serra, A., Lamberti, F. & Morra, L. (2022). Data augmentation for medical imaging: A systematic literature review. *Computers in Biology and Medicine*, 152, 106391.
- Glotsos, D. T., Kalatzis, I., Spyridonos, P., Kostopoulos, S., Daskalakis, A., Athanasiadis, E. I., Ravazoula, P., Nikiforidis, G. & Cavouras, D. A. (2008). Improving accuracy in astrocytomas grading by integrating a robust least squares mapping driven support vector machine classifier into a two level grade classification scheme. *Computer methods and programs in biomedicine*, 90 3, 251-261.
- Golatkar, A., Achille, A. & Soatto, S. (2019). Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Gong, C., Ren, T., Ye, M. & Liu, Q. (2021). MaxUp: Lightweight Adversarial Training with Data Augmentation Improves Neural Network Training. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. *Conference on Neural Information Processing Systems (NeurIPS)*.

- Goodfellow, I., Shlens, J. & Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. *International Conference on Learning Representations (ICLR)*.
- Goyal, S., Sun, M., Raghunathan, A. & Kolter, Z. (2022). Test-Time Adaptation via Conjugate Pseudo-labels. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Guan, H. & Liu, M. (2021). Domain Adaptation for Medical Image Analysis: A Survey. *IEEE Transactions on Biomedical Engineering*, 69(3), 1173–1185.
- Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. *International Conference on Machine Learning (ICML)*.
- Han, K., Xiao, A., Wu, E., Guo, J., Xu, C. & Wang, Y. (2021). Transformer in Transformer. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Hataya, R., Zdenek, J., Yoshizoe, K. & Nakayama, H. (2019). Faster AutoAugment: Learning Augmentation Strategies using Backpropagation. *European Conference on Computer Vision (ECCV)*.
- Hataya, R., Zdenek, J., Yoshizoe, K. & Nakayama, H. (2022). Meta Approach to Data Augmentation Optimization. *Winter Conference on Applications of Computer Vision (WACV)*.
- Hataya, R., Bao, H. & Arai, H. (2023). Will Large-scale Generative Models Corrupt Future Datasets? *International Conference on Computer Vision (ICCV)*.
- Hauberg, S., Freifeld, O., Larsen, A. B. L., Fisher, J. & Hansen, L. (2016). Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Hendrycks, D. & Dietterich, T. (2019). Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *International Conference on Learning Representations (ICLR)*.
- Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J. & Gilmer, J. (2021). The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. *International Conference on Computer Vision (ICCV)*.

- Hernández-García, A. & König, P. (2018a). Data augmentation instead of explicit regularization. *arXiv preprint arXiv: 1806.03852*.
- Hernández-García, A. & König, P. (2018b). Do deep nets really need weight decay and dropout? *arXiv preprint arXiv: 1802.07042*.
- Ho, D., Liang, E., Stoica, I., Abbeel, P. & Chen, X. (2019). Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. *International Conference on Machine Learning (ICML)*.
- Ho, J., Jain, A. & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A. & Darrell, T. (2018). CyCADA: Cycle-Consistent Adversarial Domain Adaptation. *International Conference on Machine Learning (ICML)*.
- Hu, J., Shen, L. & Sun, G. (2018). Squeeze-and-excitation networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Huang, G., Liu, Z., van der Maaten, L. & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Hubel, D. & Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160.
- Hubel, D. H. & Wiesel, T. N. (1959). Receptive Fields of Single Neurons in the Cat's Striate Cortex. *Journal of Physiology*, 148.
- Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization (LION)*.
- Ioffe, S. (2017). Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning (ICML)*.

- Iwasawa, Y. & Matsuo, Y. (2021). Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Jaderberg, M., Simonyan, K., Zisserman, A. et al. (2015). Spatial transformer networks. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Kang, G., Jiang, L., Yang, Y. & Hauptmann, A. G. (2019). Contrastive Adaptation Network for Unsupervised Domain Adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Karargyris, A. (2015). Color Space Transformation Network. *arXiv preprint arXiv: 1511.01064*.
- Kazerouni, A., Aghdam, E. K., Heidari, M., Azad, R., Fayyaz, M., Hacihaliloglu, I. & Merhof, D. (2023). Diffusion models in medical imaging: A comprehensive survey. *Medical Image Analysis (MedIA)*, 88, 102846.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. & Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114, 3521–3526.
- Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C. & Liang, P. (2021). WILDS: A Benchmark of in-the-Wild Distribution Shifts. *International Conference on Machine Learning (ICML)*.
- Krause, A., Perona, P. & Gomes, R. G. (2010). Discriminative clustering by regularized information maximization. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Krizhevsky, A., Hinton, G. et al. (2009). Learning multiple layers of features from tiny images. Retrieved from: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Krogh, A. & Hertz, J. A. (1992). A simple weight decay can improve generalization. *Conference on Neural Information Processing Systems (NeurIPS)*.

- Kundu, J. N., Venkat, N., M V, R. & Babu, R. V. (2020). Universal Source-Free Domain Adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Kurmi, V. K., Subramanian, V. K. & Namboodiri, V. P. (2021). Domain Impression: A Source Data Free Domain Adaptation Method. *Winter Conference on Applications of Computer Vision (WACV)*.
- Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *International Conference on Machine Learning (ICML) Workshop on Challenges in Representation Learning*.
- Lemley, J., Bazrafkan, S. & Corcoran, P. (2017). Smart Augmentation Learning an Optimal Data Augmentation Strategy. *IEEE Access*, 5, 5858-5869.
- Li, F.-F., Li, Y., Gao, R. & Byun, A. (2023). CS231n: Deep Learning for Computer Vision. Stanford University. Retrieved from: <https://cs231n.github.io/>.
- Li, R., Jiao, Q., Cao, W., Wong, H.-S. & Wu, S. (2020). Model Adaptation: Unsupervised Domain Adaptation Without Source Data. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Li, Y., Wang, N., Shi, J., Liu, J. & Hou, X. (2017). Revisiting Batch Normalization For Practical Domain Adaptation. *International Conference on Learning Representations (ICLR)*.
- Li, Z. & Hoiem, D. (2016). Learning without Forgetting. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40, 2935-2947.
- Liang, J., Hu, D. & Feng, J. (2020). Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation. *International Conference on Machine Learning (ICML)*.
- Liang, J., He, R. & Tan, T. (2023). A Comprehensive Survey on Test-Time Adaptation under Distribution Shifts. *arXiv preprint arXiv: 2303.15361*.
- Lim, H., Kim, B., Choo, J. & Choi, S. (2023). TTN: A Domain-Shift Aware Batch Normalization in Test-Time Adaptation. *International Conference on Learning Representations (ICLR)*.
- Lim, S., Kim, I., Kim, T., Kim, C. & Kim, S. (2019). Fast AutoAugment. *Conference on Neural Information Processing Systems (NeurIPS)*.

- Lin, C., Guo, M., Li, C., Wu, W., Lin, D., Ouyang, W. & Yan, J. (2019). Online Hyper-Parameter Learning for Auto-Augmentation Strategy. *International Conference on Computer Vision (ICCV)*.
- Liu, H., Simonyan, K. & Yang, Y. (2019). DARTS: Differentiable Architecture Search. *International Conference on Learning Representations (ICLR)*.
- Liu, R., Gao, J., Zhang, J., Meng, D. & Lin, Z. (2021a). Investigating Bi-Level Optimization for Learning and Vision From a Unified Perspective: A Survey and Beyond. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44, 10045–10067.
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. & Guo, B. (2021b). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *International Conference on Computer Vision (ICCV)*.
- Long, M., Cao, Y., Wang, J. & Jordan, M. I. (2015). Learning Transferable Features with Deep Adaptation Networks. *International Conference on Machine Learning (ICML)*.
- Luketina, J., Berglund, M., Greff, K. & Raiko, T. (2016). Scalable gradient-based tuning of continuous regularization hyperparameters. *International Conference on Machine Learning (ICML)*.
- MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D. & Grosse, R. (2019). Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions. *International Conference on Learning Representations (ICLR)*.
- Maclaurin, D., Duvenaud, D. & Adams, R. (2015). Gradient-based Hyperparameter Optimization through Reversible Learning. *International Conference on Machine Learning (ICML)*.
- Mahajan, D., Girshick, R. B., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A. & van der Maaten, L. (2018). Exploring the Limits of Weakly Supervised Pretraining. *European Conference on Computer Vision (ECCV)*.
- McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5, 115-133.
- Mirza, M. & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.
- Miyato, T., Maeda, S.-i., Koyama, M. & Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(8), 1979-1993.

- Müller, S. G. & Hutter, F. (2021). TrivialAugment: Tuning-Free Yet State-of-the-Art Data Augmentation. *International Conference on Computer Vision (ICCV)*.
- Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B. & Snoek, J. (2020). Evaluating prediction-time batch normalization for robustness under covariate shift. *International Conference on Machine Learning (ICML) Uncertainty and Robustness in Deep Learning Workshop*.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning and Unsupervised Feature Learning*.
- Ninos, K., Kostopoulos, S., Kalatzis, I., Sidiropoulos, K., Ravazoula, P., Sakellaropoulos, G., Panayiotakis, G. S., Economou, G. & Cavouras, D. A. (2015). Microscopy image analysis of p63 immunohistochemically stained laryngeal cancer lesions for predicting patient 5-year survival. *European Archives of Oto-Rhino-Laryngology*, 273, 159-168.
- Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S. D., Zhao, P. & Tan, M. (2022). Efficient Test-Time Model Adaptation without Forgetting. *International Conference on Machine Learning (ICML)*.
- Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P. & Tan, M. (2023). Towards Stable Test-Time Adaptation in Dynamic Wild World. *International Conference on Learning Representations (ICLR)*.
- Nowlan, S. J. & Hinton, G. E. (1992). Simplifying Neural Networks by Soft Weight-Sharing. *Neural Computation*, 4(4), 473-493.
- Odena, A., Olah, C. & Shlens, J. (2017). Conditional Image Synthesis with Auxiliary Classifier GANs. *International Conference on Machine Learning (ICML)*.
- Osowiechi, D., Hakim, G. A. V., Noori, M., Cheraghalikhani, M., Ayed, I. B. & Desrosiers, C. (2022). TTTFlow: Unsupervised Test-Time Training with Normalizing Flow. *Winter Conference on Applications of Computer Vision (WACV)*.
- Pan, S. J., Tsang, I. W., Kwok, J. T. & Yang, Q. (2011). Domain Adaptation via Transfer Component Analysis. *IEEE Transactions on Neural Networks*, 22(2), 199-210.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Conference on Neural Information Processing Systems (NeurIPS)*.

- Patel, V. M., Gopalan, R., Li, R. & Chellappa, R. (2015). Visual Domain Adaptation: A survey of recent advances. *IEEE Signal Processing Magazine*, 32(3), 53-69.
- Pathak, D., Krähenbühl, P. & Darrell, T. (2015). Constrained Convolutional Neural Networks for Weakly Supervised Segmentation. *International Conference on Computer Vision (ICCV)*.
- Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. *International Conference on Machine Learning (ICML)*.
- Peng, X., Tang, Z., Yang, F., Feris, R. S. & Metaxas, D. (2018). Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D. & Saenko, K. (2017). VisDA: The Visual Domain Adaptation Challenge. *arXiv preprint arXiv: 1710.06924*.
- Perez, L. & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv: 1712.04621*.
- Purushotham, S., Carvalho, W., Nilanon, T. & Liu, Y. (2017). Variational Recurrent Adversarial Deep Domain Adaptation. *International Conference on Learning Representations (ICLR)*.
- Qiu, Z., Zhang, Y., Lin, H., Niu, S., Liu, Y., Du, Q. & Tan, M. (2021). Source-free Domain Adaptation via Avatar Prototype Generation and Adaptation. *International Joint Conference on Artificial Intelligence (IJCAI)*.
- Quionero-Candela, J., Sugiyama, M., Schwaighofer, A. & Lawrence, N. D. (2009). *Dataset Shift in Machine Learning*. Yale University Press in association with the Museum of London.
- Radford, A., Metz, L. & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations (ICLR)*.
- Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J. & Ré, C. (2017). Learning to Compose Domain-Specific Transformations for Data Augmentation. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Riba, E., Mishkin, D., Ponsa, D., Rublee, E. & Bradski, G. R. (2019). Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. *Winter Conference on Applications of Computer Vision (WACV)*.

- Ronneberger, O., Fischer, P. & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*.
- Rony, J., Belharbi, S., Dolz, J., Ayed, I. B., McCaffrey, L. & Granger, E. (2023). Deep Weakly-Supervised Learning Methods for Classification and Localization in Histology Images: A Comparative Study. *Medical Imaging with Deep Learning (MIDL)*.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386-408.
- Rosenblatt, F. (1963). Principles of Neurodynamics. Perceptrons and the Theory of Brain mechanisms. *American Journal of Psychology*, 76(4), 705.
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations* (pp. 318-362).
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*.
- Saito, K., Watanabe, K., Ushiku, Y. & Harada, T. (2018). Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. & Chen, X. (2016). Improved techniques for training gans. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W. & Bethge, M. (2020). Improving robustness against common corruptions by covariate shift adaptation. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Shaban, A., Cheng, C.-A., Hatch, N. & Boots., B. (2019). Truncated Backpropogation for Bi-Level Optimization. *Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Shen, K., Jones, R. M., Kumar, A., Xie, S. M., Haochen, J. Z., Ma, T. & Liang, P. (2022). Connect, Not Collapse: Explaining Contrastive Learning for Unsupervised Domain Adaptation. *International Conference on Machine Learning (ICML)*.

- Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W. & Webb, R. (2017). Learning from Simulated and Unsupervised Images through Adversarial Training. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.
- Sirinukunwattana, K., Pluim, J. P. W., Chen, H., Qi, X., Heng, P.-A., Guo, Y. B., Wang, L. Y., Matuszewski, B. J., Bruni, E., Sanchez, U., Böhm, A., Ronneberger, O., Cheikh, B. B., Racoceanu, D., Kainz, P., Pfeiffer, M., Urschler, M., Snead, D. R. J. & Rajpoot, N. M. (2016). Gland segmentation in colon histology images: The glas challenge contest. *Medical Image Analysis (MedIA)*, 35, 489-502.
- Sixt, L., Wild, B. & Landgraf, T. (2018). RenderGAN: Generating Realistic Labeled Data. *Frontiers in Robotics and AI*, 5.
- Skafted Detlefsen, N., Freifeld, O. & Hauberg, S. (2018). Deep Diffeomorphic Transformer Networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Snoek, J., Larochelle, H. & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Song, Y. & Ermon, S. (2019). Generative Modeling by Estimating Gradients of the Data Distribution. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Springenberg, J. T. (2016). Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(56), 1929–1958.
- Sun, B. & Saenko, K. (2016). Deep CORAL: Correlation Alignment for Deep Domain Adaptation. *European Conference on Computer Vision (ECCV) Workshops*.
- Sun, C., Shrivastava, A., Singh, S. & Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *International Conference on Computer Vision (ICCV)*.
- Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. & Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts. *International Conference on Machine Learning (ICML)*.

- Suzuki, T. (2022). TeachAugment: Data Augmentation Optimization Using Teacher Knowledge. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer-Verlag.
- Tai, K. S., Bailis, P. & Valiant, G. (2019). Equivariant Transformer Networks. *International Conference on Machine Learning (ICML)*.
- Takahashi, R., Matsubara, T. & Uehara, K. (2018). RICAP: Random Image Cropping and Patching Data Augmentation for Deep CNNs. *Asian Conference on Machine Learning (ACML)*.
- Tan, M. & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *International Conference on Machine Learning (ICML)*.
- Tellez, D., Litjens, G. J. S., Bándi, P., Bulten, W., Bokhorst, J. M., Ciompi, F. & van der Laak, J. (2019). Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical Image Analysis (MedIA)*, 58, 101544.
- Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A. & Jegou, H. (2021). Training data-efficient image transformers & distillation through attention. *International Conference on Machine Learning (ICML)*.
- Trabucco, B., Doherty, K., Gurinas, M. & Salakhutdinov, R. (2023). Effective Data Augmentation With Diffusion Models. *International Conference on Learning Representations (ICLR) Workshop on Understanding Foundation Models*.
- Tran, T., Pham, T., Carneiro, G., Palmer, L. & Reid, I. A Bayesian Data Augmentation Approach for Learning Deep Models. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Wagner, R., Thom, M., Schweiger, R., Palm, G. & Rothermel, A. (2013). Learning convolutional neural networks from few samples. *International joint conference on neural networks (IJCNN)*.

- Wagner, S. J., Khalili, N., Sharma, R., Boxberg, M., Marr, C., Back, W. d. & Peng, T. (2021). Structure-preserving multi-domain stain color augmentation using style-transfer with disentangled representations. *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*.
- Wang, D., Shelhamer, E., Liu, S., Olshausen, B. A. & Darrell, T. (2021a). Tent: Fully Test-Time Adaptation by Entropy Minimization. *International Conference on Learning Representations (ICLR)*.
- Wang, H., Ge, S., Lipton, Z. & Xing, E. P. (2019). Learning Robust Global Representations by Penalizing Local Predictive Power. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Wang, J., Lan, C., Liu, C., Ouyang, Y. & Qin, T. (2023). Generalizing to Unseen Domains: A Survey on Domain Generalization. *IEEE Transactions on Knowledge & Data Engineering*, 35(08), 8052-8072.
- Wang, Q., Fink, O., Van Gool, L. & Dai, D. (2022). Continual Test-Time Domain Adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Wang, S., Minku, L. L. & Yao, X. (2016). Dealing with Multiple Classes in Online Class Imbalance Learning. *International Joint Conference on Artificial Intelligence (IJCAI)*.
- Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P. & Shao, L. (2021b). Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction Without Convolutions. *International Conference on Computer Vision (ICCV)*.
- Wightman, R. (2019). PyTorch Image Models. GitHub. Retrieved from: <https://github.com/rwightman/pytorch-image-models>.
- Wikipedia contributors. (2023). Convolutional neural network — Wikipedia, The Free Encyclopedia. Retrieved from: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1178788445.
- Williams, R. J. & Peng, J. (1990). An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 2(4), 490-501.
- Wilson, G. & Cook, D. J. (2020). A Survey of Unsupervised Deep Domain Adaptation. *ACM Transactions on Intelligent Systems and Technology*, 11(5).
- Wu, B., Chen, W., Fan, Y., Zhang, Y., Hou, J., Liu, J. & Zhang, T. Tencent ML-Images: A Large-Scale Multi-Label Image Database for Visual Representation Learning. *IEEE Access*, 7, 172683-172693.

- Wu, Y. & He, K. (2018). Group Normalization. *International Journal of Computer Vision (IJCV)*, 128, 742–755.
- Xiao, H., Rasul, K. & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv: 1708.07747*.
- Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M. & Mahajan, D. K. (2019). Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv: 1905.00546*.
- Yang, S., Wang, Y., van de Weijer, J., Herranz, L. & Jui, S. (2021a). Exploiting the Intrinsic Neighborhood Structure for Source-free Domain Adaptation. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Yang, S., Wang, Y., van de Weijer, J., Herranz, L. & Jui, S. (2021b). Generalized Source-free Domain Adaptation. *International Conference on Computer Vision (ICCV)*.
- Yao, H., Choi, C., Lee, Y., Koh, P. W. & Finn, C. (2022). Wild-Time: A Benchmark of in-the-Wild Distribution Shift over Time. *International Conference on Machine Learning (ICML) Shift Happens Workshop*.
- Yeh, H.-W., Yang, B., Yuen, P. C. & Harada, T. (2021). SoFA: Source-data-free Feature Alignment for Unsupervised Domain Adaptation. *Winter Conference on Applications of Computer Vision (WACV)*.
- Zagoruyko, S. & Komodakis, N. (2016). Wide Residual Networks. *British Machine Vision Conference (BMVC)*.
- Zhang, H., Cisse, M., Dauphin, Y. N. & Lopez-Paz, D. (2018). mixup: Beyond Empirical Risk Minimization. *International Conference on Learning Representations (ICLR)*.
- Zhang, M., Levine, S. & Finn, C. (2022). MEMO: Test Time Robustness via Adaptation and Augmentation. *Conference on Neural Information Processing Systems (NeurIPS)*.
- Zhang, X., Wang, Z., Liu, D. & Ling, Q. (2019). DADA: Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.
- Zhao, A., Balakrishnan, G., Durand, F., Gutttag, J. V. & Dalca, A. V. (2019). Data augmentation using learned transforms for one-shot medical image segmentation. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Zhao, B., Chen, C. & Xia, S. (2023). DELTA: degradation-free fully test-time adaptation. *International Conference on Learning Representations (ICLR)*.