NEW METHODS FOR DOMAIN ADAPTATION AND LOW DATA DEEP LEARNING

Muawiz Chaudhary

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

November 2023

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: Muawiz Chaudhary

Entitled: **New Methods For Domain Adaptation And Low Data Deep Learning**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Charalambos Poullis

_____ Examiner
Dr. Mahdi Hosseini

_____ Examiner
Dr. Charalambos Poullis

_____ Supervisor
Dr. Eugene Belilovsky

Approved by _____
Dr. Leila Kosseim, Graduate Program Director
Department of Computer Science and Software Engineering

_____ 20 _____ _____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

New Methods For Domain Adaptation And Low Data Deep Learning

Muawiz Chaudhary

Real-world data coming from settings like hospital collections for detecting disease experience multiple sources of distributional shifts. These issues affect the performance of diagnostic methods, reducing the quality of service provided and leading to health or economic harm. Deep learning has emerged as a promising method for classification tasks, including diagnostics, and recent progress has led to methods that allow a neural network to adapt network statistics to shifts in specific settings at test time. However, problems arise in these methods adapting to general shifts and domains. In addition, they underperform when data is limited. In our first contribution, we tackle general domain shifts by investigating the key issues leading Test Time Adaptive algorithms to fail under label shift, proposing a means for mitigating these failures. In the second contribution, we tackle few-shot cross-domain adaptation by modifying the affine parameters of the batch norm during few-shot train time, generally enhancing performance. The third contribution parameterizes Scattering Networks, where we enhance a method for low data regimes by providing problem-specific adaptation.

Acknowledgments

After two long years, this academic journey is finally coming to an end. I thank my advisor Eugene Belilovsky for all of the time and effort he has spent on me. It was not always easy, but you looked out for my best interests and put me on projects where I could do well and learn much, and no other advisor would have been able to do that for me.

It takes a village to raise a child, and I thank the academic network behind Kymatio and Scattering Networks for the mentorship they've given me; Matthew, Edouard, Joakim, Vincent, Michael, and Guy. Special thanks go to Edouard for his involvement during my undergrad in encouraging me to contribute to Kymatio and involving me in research. I know it took much time and resources on your part, but I'm here now with a master's and some ideas on how to do research. Special thanks go to Michael and Guy for their mentorship during my master's, where they collaborated on projects I participated in. I want to also thank Vincent for conversations about what it means to be an academic and to do research. A project you suggested for a class of mine, and our discussions around this project, were key to advancing my scientific thinking.

I thank those from my undergrad and my past life; Daniel Tarnu, Jeff Katen, Sean Haight, Ryan Haight, Chris Carega, Robin Cosbey, Ana Usenko, Mia Kasperek, Nick Knowles, Eric Slyman, Brian Hutchinson, Parissa Rad, Citlalli Aquino.

I am glad to have been a part of a well-resourced research institute. There are far too many people to thank here. But I will try nonetheless.

I give thanks to the collaborators in the research projects I participated in. Shanel Gauthier, Benjamin Therien, and Laurent Alsene-Racicot were an absolute joy to work with on Parametric Scattering Networks. I enjoyed our time figuring out how Scattering Networks worked and being a collaborator on the work. Shanel, your leadership is the best I've had, Laurent, you know your mathematics, and Ben, you are always a joy to be around, which I enjoyed every minute of, especially at CVPR 2022. Thanks are also given to Ben for his excellent counsel. I additionally thank Moslem Yazdanpanah, Aamer Abdul Rahman, Amir Sarfi, and Pedro Vianna for our

insightful and fun conversations on the other projects we worked on together. I give special thanks to Pedro Vianna for his insights and work on the medical data we worked with. Thanks also to the professors involved in some of this work, such as Drs. Irina Rish and Samira E. Kahou.

I thank the institution of the LabReps, specifically the 2021-2022 and 2022-2023 cohorts, the latter of which I had the privilege and honor of being a part. Thanks go to Alex Hernandez-Garcia, Dongyan Lin, Mandana Samiei, Mashbayar Tugsbayar, Mélisande Teng, Nikolaus Howe, Sékou-Oumar Kaba, Victor Schmidt, Kshitij Gupta, Alekhya Dronavalli, Ezekiel Williams, Mattie Tesfaldet, Mats L. Richter, Anna Richter, Alexandra (Sasha) Volokhova, Zhaocheng Zhu, Subhrajyoti Dasgupta, Beheshteh T. Rakhshan, Sophie Xhonneux, Roy Eyono, and Gabriela Moisescu-Pareja. Special thanks go to Victor, Melisande, Alex, and Oumar for their insights and a listening ear, and special thanks to much of my LabReps cohort for the support I needed during a difficult time. I also give special thanks to Mattie Tesfaldet, Mats L. Richter, and Anna Richter for their friendship and helpful advice during my master's.

I thank the members of the academic labs I was a part of. Thank you to Wolf's lab, in particular Ladislav Rampasek, Yanlei (Kaly) Zhang, Albert Orozco Camacho, Semih Canturk, Guillaume Huguet, Stefan Horoi, Lydia Mezrag, Sasha Morin, Shuang Ni, Frederik Wenkel, Laurent Alsene-Racicot, Myriam Lizotte, and Hasti Nafisi. I enjoyed the fun times we had; thanks to Shuang for the Chinese candies and the ping pong games, to Hasti and Guillaume (and others, of course) for their care, and to Semih, Stefan, and Myriam for the fun we had during our outings. Thank you to the Belilovsky lab, specifically Reza Davari, Abhinav Moudgil, Albert Orozco Camacho, Gwen Legate, Alexander Fulleringer, Geraldin Nanfack, Nader Asadi, Nasir Khalid, Amir Sarfi, Adeetya Patel, Medric Sonwa, Irene Tenison, and Benjamin Therien. I found conversations with Abhinav, Gwen, and Geraldin to be very insightful in the realm of research, and I also thank them for their entertaining discussions on more social matters.

There was a point where I wasn't sure if I should finish my master's. I am thankful that some people told me otherwise. More specifically, I am grateful to Cyrus Vahidi, Han Han, Michelle Lin, Michelle Liu, Maude Lizaire, Mattie Tesfaldet, Mats L. Richter, Anna Richter, Khaoula Chehbouni, Raesetje Sefala, Timnit Gebru, Michael Przystupa, Emmanuel Bengio, Aristides Milios, Jazlyn Hellman, Christina Isaicu, Michael Runningwolf, David Yu-Tung Hui, among others, who encouraged me to stay on track, focus on my research, and get my master's.

I am grateful for the opportunity to have interacted with senior researchers in

Deep Learning, enriching my academic experience. Thank you to Gauthier Gidel for allowing me to present my work on Scattering Networks to your research group; that presentation gave me confidence for future research presentations. Thank you to Irina Rish for your scaling laws class; even if I disagree vehemently with a reliance on scaling, you gave me insights on how to do scientific research and the value of a negative result. Thank you to Samira Kahou for your research insights and helpful advice on navigating Mila; I may not have followed it perfectly, but I did follow it for a time! Thank you to Aaron Courville for the opportunity to TA for your Deep Representation Learning class; the challenge and freedom to develop my own assignment for a fundamental class impacted how I see and do research. Thank you to Timnit Gebru for your critical work and advice; focusing on Mua-self and my research is the best advice I could apply for my scientific career. I am also grateful to other seniors, such as Golnoosh Farnadi, Kosta Derpanis, Glen Berseth, Nicolas Le Roux, and Simon Lacoste-Julien, for their helpful insights.

I am immensely grateful to Olexa Bilaniuk for his help on matters related to the Mila Cluster and Linux command line, his counsel on my academic career, his support, and the numerous entertaining conversations we've shared.

There are far too many researchers at Mila for me to remember to give thanks to in this thesis of mine. Thanks to Reza Bayat, Amin Me'marian, Diganta Misra, Rozhin Nobahari, Matthew Fortier, Jules Gagnon-Marchand, Adam Ibrahim, Dinghuai Zhang, Johan Obando, Alexey Ostapenko, Yuchen Lu, Abdelrahman Zayed, Mohammad Reza Samsami, Aarash Feizi, Samuel Lavoie, Martin Weiss, Andrew Zeng, Vitoria Barin Pacela, David Kanaa, Tristian Deleu, Amin Mansouri, Bonaventure Dossou, Andjela Mladenovic, Saba Ahmadi, Joseph Viviano, Sonia Dastani, Rohan Sukumaran, Rohan Banerjee, Cristian Dargos Manta, Joey Bose, Vincent Mai, Motahareh Sohrabi, Nithya Shikarpur, Aneri Muni, Charlie Gauthier, Behnoush Khavari, Jacob Buckman, Xing Han Lu, Benno Krojer, Darshan Patil, Mizu Nishikawa-Toomey, Florian Golemo, Fabrice Normandin, Sonia Joseph, Juan Ramirez, Sahand Razaei-Shoshtari, Maryam Molamohammad, Milad Aghajohari, Amirhossein Kazemnejad, Simon Demeule, Nicola Neophytou, Rebecca Salganik, Alex Tong, Sumana Basu, Padideh Noruri, Arnav Jain, Reyhane Askari, Pedram Khorsandi, David Dobre, Chen Sun, Simon Guiroy, Rupali Bhati, Arian Khorasni, Samuel Garcin, Christos Tsirigotis, Ivaxi Sheth, Ethan Caballero, Tom Marty, Etienne Boucher, MJ Bayazi, Sangie Bhardwaj, Hattie Zhou, Mehrnaz Mofakhami, Benhamin Akera, Mohammad Yaghoubi, Breandan Considine, Divyat Mahajan, Dishank Bansal, Emy Yue Hu, Kolya Malkin, Karam Ghanem, Evelin Fonseca Cruz, Mahta Ramezanian, Jie Bao, and Prishruit Punia. I apologize

dearly to anyone that I have missed.

Thank you to my family for your support.

Thank you to each of the countless flowers I held during my master's.

Contents

List of Figures

List of Tables

List of Abbreviations

BN Batch Normalization

CDFSL Cross-Domain Few-Shot Learning

CL Continual Leanring

CNN Convolutional Neural Network

FN Feature Normalization

FSL Few-Shot Learning

LP Linear Probe

MLP Multi-Layer Perceptron

NN Neural Network

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

SSL Self-supervised Learning

Chapter 1

Introduction

1.1 Overview

Hospitals are critical sites for diagnostics around illness. Classification of disease has many sources of influence; patient populations, calibration of tools used to diagnose disease, and prevalence of infection [3, 4]. Sources, characterized as distributions, can shift over geography, time, and other variables [3, 4]. The development of diagnostic tools may only consider some possible sources, as data might be unobtainable in some environments due to capacity or privacy constraints, and thus require the health practitioner applying a diagnostic to consider several distributional shift issues [5] which introduce a possible vector of diagnostic errors, or impossibility of detecting disease. Failure of these tools to adapt to shifts means that patients can suffer, having downstream impacts on economies. Automating the adaptation of diagnostic algorithms as sources shift can allow for a solution [4].

Deep learning methods enable a parametrized model to learn from data through an optimization process with a specified loss function [6]. A non-trivial number of scientific efforts study the use of deep learning algorithms to automate standard processes where data is available [7–13]. Also known as neural networks, these methods have found successful applications in computer vision [14], protein folding [15], and even playing video games [16]. Historically these methods required massive amounts of labeled data to train networks. Developments in methodology and model architecture, such as BatchNorm [17], led to networks that could better transfer across tasks, such as classification to recognition, after finetuning on a target set. Initially, these transfer methods were limited to tasks with the same underlying distribution, where natural images are one such example of a distribution, to eventually adapting between distributions, such as natural image to medical imaging transfer [18, 19]. Further

developments led to small sample and few-shot methods that rapidly adjust neural networks in specific low-data scenarios across domains [20]. Recent investigations have looked into on-the-fly unsupervised optimization during deployment settings, the latter of which commonly are of shifting distributions and can be composed of limited samples [21, 22]. These test-time-adaptation (TTA) methods are prone to failure when facing multiple distribution shifts, limiting their use in clinical settings which require robust algorithms. The aim of this thesis is to study this distributional shift's impact on neural networks, alongside other issues such as low data availability, and propose methods for adapting neural networks to these distribution shifts.

Specifically, this thesis studies the test time adaptation, few-shot, and small-sample contexts. Firstly, we explore the failure cases of a well-known baseline for TTA, Test Time Normalization (TTN), finding that TTN fails under various distribution shifts. We investigate this failure, seeing that shifting class means due to particular distribution shifts is likely responsible, and this failure in performance is mainly constrained to later layers. Secondly, we look into adapting the Batchnorm affine parameters in cross-domain few-shot learning settings, finding that this method works well. Finally, we analyze the performance and deformation stability properties of the parametric scattering transforms, an extension of the scattering transforms developed initially to generate discriminative features for small sample natural image and texture classification. We find that parametric scattering networks mostly keep deformation properties and increase performance on the classification of medical images, giving evidence that a method developed explicitly for natural images can be slightly modified to generalize to different distributions.

1.2 Main Contributions and Outline

In chapter 3, we show a common failure case of Test Time Normalization (TTN) under label shift. We then hypothesize that this is related to the shift in class means deeper into the network and proceed to give evidence for this hypothesis. A method called Hybrid-TTN is introduced, keeping covariate shift robustness of the TTN while mitigating the impact of label shift. In the paper "Investigating Prediction-Time Batch Normalization Under Label Shift" [23], unpublished at time of writing, I led the project, discovering a critical failure in a popular test time adaptation method, conducting experiment design (developing the experiments for figures 2, 3, and 4), conducting a literature review, organizing the writing of the paper, and running the experiments for tables 2 and 3, alongside with writing initial drafts of the document.

- Muawiz Sajjad Chaudhary, Pedro Vianna, Michael Eickenberg, An Tang, Guy Cloutier, Guy Wolf, and Eugene Belilovsky. "Investigating Prediction-Time Batch Normalization Under Label Shift." Unpublished as of yet. 2023.

In chapter 4, we show that freezing the affine parameters of the BatchNorm during pre-training and then adapting these same affine parameters during few-shot evaluation time improves performance on cross domain datasets. We investigate the sparsity induced before and after the ReLU activation in networks trained with BatchNorm and networks trained with affines disabled in the BatchNorm, and find that the later has denser features, which we hypothesize are important to our methods improvement. In the paper "Revisiting learnable affines for batch norm in few-shot transfer learning" [24], accepted to CPVR 2022 as a poster presentation, I developed the FineAffine method, which has seen application in a few few-shot papers, ran all of the experiments for the first table, much of the experiments for the second (main) table, half of the experiments for table 3 (ablation study on fine tuning specific affine parameters), and developed table 6 (sparsity study). I developed the poster presentation. The method was based off the experimental design from Johnathan Frankles Training Batchnorm and only Batchnorm paper, adapted for use in the Few Shot Setting.

- Moslem Yazdanpanah*, Aamer Abdul Rahman*, Muawiz Chaudhary, Christian Desrosiers, Mohammad Havaei, Eugene Belilovsky, and Samira Ebrahimi Kahou. "Revisiting Learnable Affines for Batch Norm in Few-Shot Transfer Learning." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9109-9118. 2022.
  * denotes equal contribution

In "Simulated Annealing in Early Layers Leads to Better Generalization" [25], accepted to CPVR 2023 as a poster presentation, I did all of the experiments for the few shot learning section, which additionally used my FineAffine method proving that the method works on additional pre-training regimes. I provided some minor editing of the text.

- Amir Sarfi*, Zahra Karimpour*, Muawiz Sajjad Chaudhary, Nasir Mohammad Khalid, Mirco Ravanelli, Sudhir Mudur, and Eugene Belilovsky. "Simulated Annealing in Early Layers Leads to Better Generalization." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20205-20214. 2023.
  * denotes equal contribution

In chapter 5 we paramterize the wavelets of the Scattering Network, and show various properties of these new Parametric Scattering Networks. We then apply these new Parametric Scattering Networks in an unsupervised learning pipeline, showing competitive performance over Scattering Networks. "Parametric Scattering Networks" [26], accepted to CPVR 2022 as an oral and poster presentation, I ran few of the main table experiments as needed, the "no auto-augment" experiments, equivariant ablation experiments, number of filters ablation experiments, most of the deformation stability experiments and conducted the initial analysis that discovered that learned Parametric Scattering Networks keeps similar deformation stability properties as Scattering Networks, computational and memory complexity section, and the unsupervised scattering section, along with developing the little wood paley diagram used for visualizing dataset specific parameterizations. I helped develop the oral and poster presentation for CVPR. Additionally, I did massive refactorizations of the code base, and developed further code/google colab notebooks for tutorials on the Parametric Scattering Transforms. Code that I developed during my undergrad, under the banner of the Kymatio framework, was used heavily in this project.

- Shanel Gauthier, Benjamin Thérien, Laurent Alsène-Racicot, Muawiz Chaudhary, Irina Rish, Eugene Belilovsky, Michael Eickenberg, and Guy Wolf. "Parametric Scattering Networks." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5749-5758. 2022.
  * denotes equal contribution

Chapter 2

Background

In this chapter, we explain briefly the background materials related to the works presented in this manuscript.

2.1 Domain Adaptation

A standard evaluation of machine learning models is to evaluate a dataset sampled from the same or a similar distribution as the training set. This setting is known as in-distribution or the source domain and often has the property of having easy-to-annotate data and is a simpler problem [18]. When the fine-tuning task and/or evaluation task comes from a significantly different distribution, this is known as out-of-distribution or a target domain [27, 28]. Often these out-of-distribution settings have limited labeled or unlabeled data, and models pre-trained on a different distribution often fail to generalize [29, 30]. Examples include a model trained on natural images now applied to medical imaging tasks, a model trained to summarize newspapers of economic forecasts used to simplify scientific articles, or detecting anomalous objects/actions in a setting. Out-of-distribution settings can happen due to dataset shift, such as covariate shift, where the input distribution experiences a change; label shift, where the output distribution shifts but labels stay the same; and concept shift, where there is a change in the relationships between inputs and outputs [31]. Domain adaptation, a subset of transfer learning, aims to mitigate the impact of these distributional shifts. Domain adaptation provides methods for adapting to new domains with the same labels as the pre-training task. As a result, using domain adaptation can avoid labeling distribution-shifted data or spending significant resources to re-train models on the out-of-distribution data. Adapting or aligning feature distributions is essential here. Several methods used here include reweighting examples in the source domain

(instance-based adaptation) [32], iteratively refining a model on a target domain (iterative refinement) [33], learning invariant feature representations across domains (feature-based adaptation) [34], or using a Bayesian model [35].

2.2 Test Time Adaptation

Deployment settings commonly have dataset shifts [31]. As mentioned, these shifts, such as covariate, label, or concept shift, can severely impact the performance of a deep learning model during deployment. While some settings, such as recommender systems, might be tolerant to failure cases, other settings, such as autonomous driving or medical imaging, can be expected to be deployed in settings where dataset shift is typical and have massive impacts on individuals [21, 22]. Making our models adapt in real-time robustly is then essential and includes developing methods for when there is specifically a covariate, label, or concept shift. Evaluation settings can consist of evaluating your test time adapted learner in a source-free, batch/instance, or online setting [31]. Given that we cannot access ground truth labels and are adapting to distribution shifts, the domain of Test Time Adaptation is actually a special case of Domain adaptation. Test Time Adaptation uses many standard Unsupervised Domain Adaptation methods [31]. However, a notable difference with Domain Adaptation is that the model trained on the source data, termed a source model, has no access to the source dataset during the evaluation of the target dataset, possibly due to a privacy constraint or a lack of resources. This difference in the setting is referred to as Test Time Adaptation and intersects with Federated and Continual Learning. Test Time Adaptation may also use self-supervised [36], semi-supervised, pseudo-labeling [37], contrastive [38], or clustering [37]methods. Our work focuses explicitly on the batch setting. Much work exists which takes advantage of the batch setting, such as a simple adaptation of batch norm statistics [21, 22], alongside an entropy minimization on the batch norm parameters [39]. These methods work well in covariate-shifted settings when there is corruption on the input images but fail upon encountering label-shifted scenarios.

Formally, we assume we have f_θ, where f is our model and θ the parameters of this model trained on the source domain $\{X_{train}, Y_{train}\} \sim D_{Source}$, where D_{Source} is the source distribution, and we want to adapt θ to $X_{test} \sim D_{Target}$ without access to D_{Source}, where D_{Target} is the target distribution shifted from the source distribution.

2.3 Few-Shot Learning

Traditionally, machine learning models require large labeled domain-specific data sets to classify images correctly [6]. Adding new classes requires significant resources to generate clean labeled inputs and training from scratch models. Typical resource constraints include the limited availability of data, time and monetary cost to clean and annotate data, a need to deploy models rapidly, or privacy concerns [40, 41]. In contrast, young human children can tell the difference between a horse and a zebra, given limited prior knowledge of a horse and a single image of a zebra. Of great interest then is developing a set of methods for making a model rapidly generalize to novel classes across various domains [42, 43]. This paradigm is formatted and evaluated using the notion of episodic N-Way K-Shot learning. An episode refers to an instance of examples sampled from a dataset composed of a (support) set we may train or fine-tune consisting of NK images and a (query) set of images we wish to evaluate N refers to the number of novel classes we see during the few-shot tuning, and K refers to the number of examples for each novel class we see in the support set. Of interest in recent years is a cross-domain setting with which to evaluate these Few Shot Learning methodologies [1].

Meta-learning, or "learning-to-learn," is a related, more general field [43]. During meta-training, we construct multiple episodes with samples from a dataset. Each episode trains a meta-learner to adapt a model to a support set and evaluation done on a query set, producing a signal to update the meta-learner. After several episodes, an assessment of the meta-learner takes place during a meta-eval phase, with the meta-learner's performance averaged over multiple episodes to measure the ability to generalize to new data rapidly. Few-shot learning is an instance of this meta-learning paradigm, where there is no notion of a meta-train and only a meta-evaluation. Usually, models in this latter setting are assumed to pre-train on a large dataset.

One set of methods to attack the meta-learning problem is through learned similarity metrics, such as SiameseNets [44], TripletNets [20], or networks such as MatchingNets [45], Proto-typicalNets [46], and RelationNets [47]. The earlier set of mentioned networks takes embeddings and computes similarity for a binary answer to "Is this query embedding in the same class as this support embedding" while the last set of networks computes similarity across multiple novel classes. Another set of methods uses prior knowledge, whether fine-tuning specific components of a pre-trained network (such as MAML which learns an initial model that is updated [48]) or learning update rules (LSTM-based meta-learners [49]).

2.4 Scattering Transforms

Classification problems demand an ability to learn invariances to certain affine transformations of the input signal [50–52]. These affine transformations can include but are not limited to, translations, rotations, and dilations. Additionally, it is usually helpful to have resistance to small deformations of the input signal. Scattering transforms [51, 52] build an affine-invariant, stable representation of an input signal that keeps high-frequency information. To be more specific, the scattering transforms are a cascade of convolutions with wavelet filters followed by nonlinear modulus and averaging operations. Mathematical principles guide the design of wavelet filters, which play a similar role to convolutional filters [52]. Morelet wavelets make up the filters of the Scattering Transform, constructed according to a tight-frame initialization. Initialization of the filter bank features a mother wavelet and a family of wavelets generated from dilations and rotations from this mother wavelet. The initialized filter bank then covers half of a frequency plane. This construction of the wavelet family, alongside the properties of the modulus non-linearity, gives Scattering Transforms energy preservation and stability to small deformations. The Scattering Transform is traditionally defined to be translation invariant but can encode further structures suited for a particular task, such as rotation or time-frequency invariance [53–55]. As a result, the Scattering Transform produces discriminative representations that often perform well in the low data regime [56].

Due to the cascaded structure, Scattering Transforms are similar to a CNN and, as a result, are used as a simplified, unlearned mathematical model for showing how one-layer CNNs work [52]. Previous work has shown the filters of the early layers of CNNs to learn Gabor wavelets as filters [57], not unlike the filters specified in the Scattering Transform, and that initialization of these layers as Gabor wavelets gives similar transfer learning results as pre-trained networks [19]. Further work has demonstrated that Hybrid Scattering Networks, CNNs where Scattering Transform substitutes the earlier layers, keep impressive performance, showing that Scattering Transforms can replace the earlier layers and even scale to datasets such as Imagenet [58, 59].

Chapter 3

Investigating Prediction-Time Batch Normalization Under Label Shift

3.1 Introduction

Deep neural networks are able to achieve strong performance on a wide variety of tasks from computer vision, natural language processing, speech, and others [60,61]. However, a commonly cited limitation of existing deep learning models is limited generalization under natural distribution shifts [62, 63]. Indeed, in real-world deployment scenarios models may encounter various corruptions and unexpected changes in environment. In the context of perceptual data such as images, many of these shifts are easily handled by humans (e.g. a change in the illumination through sunlight for pedestrian detection or sensor noise in a camera) but can cause catastrophic failures for many machine-learning and deep-learning models.

A recently emerging paradigm to deal with distribution shift is test-time adaptation (TTA) [64], a subset of unsupervised domain adaptation (UDA) [65] where it is assumed that access to the source data is not possible, which is realistic in deployment scenarios. It is also often assumed that data arrives in batches and thus an unlabeled subset of data from the target domain is available. Many algorithms have been proposed in this setting that take advantage of batch-level information to adapt to the distribution shift [21, 66, 67]. Test time normalization (TTN), also called prediction-time batch-normalization (BatchNorm) [21], is a simple and popular method for adaptation [21,22].

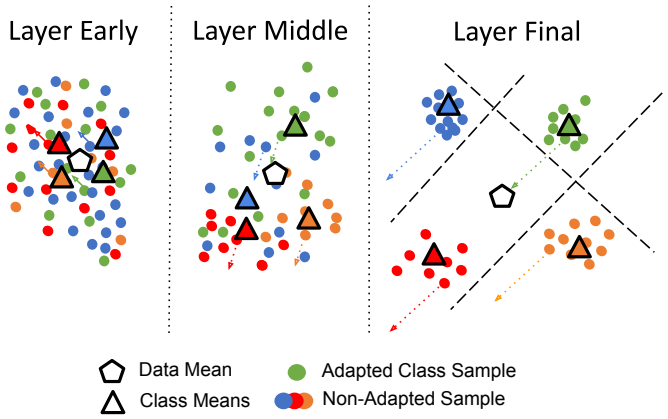

Figure 1: We illustrate a mechanism for explaining the observed behavior under label distribution shift. We consider one class mean (green) which is shifted towards the data mean, as would be the case in a highly imbalanced setting. Classes are not well separated in early layers and thus shifts in any mean are relatively small and unintrusive. In later layers classes are well separated and a large shift of points from one mean towards the data mean is likely to cross a decision boundary. Data points in other classes moving away from the data mean are less likely to cross a decision boundary.

In TTN, BatchNorm statistics [68] are updated on batches from the target data. Despite being a simple approach, it has been shown to yield strong performance for adaptation in prior work, handling particularly well various cases of image corruption. Furthermore, other TTA methods utilize TTN as a critical component [66].

In many realistic scenarios the label distribution of data can shift from training to testing scenarios. For example, object distributions encountered by autonomous vehicles can shift with the location; or medical data can vary in the prevalence of an underlying disease from one location to another. In order to be practically useful, TTA methods relying on batched data must then be able to deal gracefully with label distribution shifts alongside co-variate shifts. On the other hand, in the existing TTA literature methods such as TTN have been only evaluated in the optimistic scenario of class-balanced train and evaluation. In this work we investigate the effect of label distribution shift on TTN, and observe that it can lead to catastrophic failures.

In particular, our contributions are:

1. We investigate the effect of label distribution shift on TTN, observing that under severe imbalance, the performance of the model can be highly degraded compared to using no adaptation at all.

2. We argue and demonstrate that this degradation can be attributed to increased class separation at later layers in a deep network

3. We propose a simple method, named *Hybrid-TTN*, which more gracefully deals with label distribution shifts and, in particular, highly imbalanced target data, while maintaining the benefits of TTN under covariate shift.

The paper is organized as follows, in Sec. 3.2 we review the related work. In Sec. 3.3 we formalize test-time normalization and our Hybrid-TTN. In Sec. 3.5 we show the failure cases of TTN under label distribution shift, motivate our method to correct for them, and directly apply our method to various datasets. Finally, in Sec. 3.6 we discuss the conclusions of our findings and propose further directions. Code for our experiments is provided in the supplementary materials.

3.2 Related Work

BatchNorm [68] significantly aids in the optimization of deep neural networks, regularizing overfitting, and normalizing and scaling features and gradients thus relieving the exploding or vanishing gradient problem. A batch of features is normalized by a mean and variance calculated within a batch. Methods extending this approach without looking at the batch dimension include a normalization based on groups of channels, instance-based normalization, and a layer based normalization [69–71]. Weight normalization further extends this line of work, not modifying the features but the weights of the model themselves [72].

Various studies have been conducted to further understand how batch-normalization works. [73] notes that internal covariate shift is not a good explanation, and that the loss landscapes of BatchNorm models appear to be a lot smoother than those of equivalent models without. [74] takes randomly initialized networks and optimizes only the BatchNorm parameters of the network, finding that a third of the features are shut down and the resulting networks obtain surprisingly good performance on CIFAR-10 and Imagenet.

Batch-normalization has been extensively studied in the few-shot setting. Given small amounts of data in the cross-domain setting, [75] find that shutting off the affine transformations computed by BatchNorm during pre-training and then re-enabling the affines when finetuning the linear classifier head on a downstream cross domain few shot task improves performance. FiLM layers, a conditional normalization method, improve few shot performance [76, 77].

TTN is a method for adapting BatchNorm parameters to distributions with covariate shift [78, 79]. It is closely related to the UDA method AdaBN [80, 81]. A

more recent TTA method, TENT utilizes an entropy minimization procedure on the BatchNorm affine parameters [39].

Recent TTA methods have also utilized ideas from self-supervised learning [67, 82]. However they can lead to a costly and sensitive approach at adaptation time. Furthermore, [82] has been improved over by TTN based methods.

Another class of methods does not utilize batches for test time adaptation have been proposed. In [83] a framework is developed for adapting to single samples at a time, motivated by inference time speed constraints. However, these classes of approaches cannot fully leverage the distribution level statistics. Furthermore, they are often heavily specialized to image data. [84] recently adapts the setting to Automatic Speech Recognition.

The long tailed setting assumes training on an imbalanced train set and evaluation on a balanced test set [85]. This is in contrast to our setting where training can be on a balanced test set and evaluation on imbalanced data. [86], during inference, standardizes the predictions of models trained on long-tailed distributions. It is noted in [87] that there is often a distribution shift associated with various methods for long-tailed recognition and a solution is proposed. [88] utilizes a Gaussian mixture to reduce the dominance of the majority classes in estimating statistics and affine parameters. However, it is not noted in these works that the performance of dominant classes is significantly degraded, and effects are only explored during training.

3.3 Methods

3.3.1 Hybrid-TTN

The main idea behind TTA in general, and TTN in particular, is that while label information is not available at test time, some information is available to estimate impact of domain shifts on neural network operations and internal representations. In particular, the setup typically considered here is based on data being processed in batches, enabling assessment of distribution shifts between source and target domains. In order to implement TTA in these settings, TTN views a neural network f as split into blocks separated by batch normalization (BatchNorm) layers:

$$f = f_K \circ B_{K-1}^{s_{K-1}} \circ f_{K-1} \cdots \circ B_1^{s_1} \circ f_1 \circ B_0^{s_0} \circ f_0, \tag{1}$$

where $f_0, \ldots, f_K$ are blocks (i.e., sub-networks) of hidden layers and $B_0^{s_0}, \ldots, B_{K-1}^{s_{K-1}}$ are batch normalization operators, with $s_0, \ldots, s_{K-1}$ specifying which of these operations

is "active" or "frozen". Whether frozen or active, each BatchNorm layer modifies each neural activation by

$$B^s(h(x)) = \beta \frac{h(x) - \mu^s}{\sigma^s} + \gamma, \tag{2}$$

where β and γ are parameters learned during the training process, and μ^s and σ^s represent estimates of the mean and standard deviation of neuron activation over data. The main difference between active and frozen operations is whether this estimate is actively computed on each input batch at test time, or frozen at test time based on the statistics computed over the training data. We note that at training time all BatchNorm operations are in active state, while the main difference between TTN and traditional supervised learning is whether BatchNorm operations are active or frozen, respectively, during test time.

The main premise of the TTN approach is that changes in the distributions of activations of each neuron between source and target batches would predominantly be caused by unwanted covariate shifts, and therefore should be eliminated. However, this does not take into account other distribution shifts that *should* affect, at the very least, the output distribution of the network. In particular, in many real world applications, one cannot guarantee that classes will be well balanced in general and certainly at test time. Therefore it is often the case that the distribution of available labels during the training process i.e. on the source domain will differ from one of unknown labels encountered at test time. Most successful applications of TTN did not contain such label distribution shift, and recent work has indicated possible sensitivity of TTN to such shifts [81]. Indeed our work also establishes, as shown in Sec. 5.4, that TTN may cause a significant adverse effect in the presence of label distribution shifts.

In order to mitigate the risk of adverse effects by TTN, we study its impact on early versus later layers of a neural network. To this end we first recall that later layers in neural networks are understood to be more specialized than earlier layers, which perform generally useful feature extraction [89–91]. Furthermore, several studies have shown that linear separability between classes typically increases, gradually, within the depth of the network [92, 93]. With this in mind, we can expect that the global statistics (e.g., mean and standard deviation of neuron activation) over a data batch would be more similar to the statistics of each individual class in early layers than in later layers. Therefore the dependence of BatchNorm operations and the shift (or scaling) they apply to earlier layers would be less affected by class and label distributions than their effect in later layers. As a result, partial application of TTN limited to sufficiently early parts of the network should be relatively resilient to label

distribution shifts while at the same time obtaining some representation robustness to covariate shifts, which will propagate through the rest of the network and partially retain the advantages of full TTN.

Based on this understanding, we propose a simple approach we call **Hybrid-TTN**: we set the first ℓ BatchNorm operations to active, while fixing the rest of the $K - \ell$ BatchNorms to frozen. Here, we choose the exact cutting point ℓ via a heuristic based on single-class accuracy degradation estimated on the train set, which represents a worst-case label distribution shift[1]. We simulate several sets of extreme label distribution shift, giving us only one single class in the target data batch, and compute the resulting accuracy of Hybrid-TTN for each possible cutoff layer. Then, for each possible cutoff, we take the average accuracy and analyse the drop in accuracy relative to the zero cutoff (i.e., no test-time active BatchNorm setting). The Hybrid-TTN network then uses the largest ℓ with accuracy drop below a given threshold, which is a hyperparameter to be tuned based on the number of classes (e.g., in our experiments we use 5% for datasets with fewer labels, such as CIFAR-10, and 10% for bigger ones, such as Imagenet).

3.4 Experimental Setup

In this section we discuss our experimental analysis highlighting the issues that arise under label distribution shift.

We utilize three datasets in our evaluations, two popular benchmarks, CIFAR-10-C and Imagenet-1K-C, and a LiverUltrasound Dataset captured from multiple devices.

CIFAR-10 and CIFAR-10-C. We use the CIFAR-10 [94] dataset along with CIFAR-10-C [62]. CIFAR-10 is a small natural image dataset with 50k training images and 10k validation images. CIFAR-10-C contains corrupted versions of the CIFAR-10 Validation set at varying severities. We trained our models on the uncorrupted data set.

Imagenet-1K and Imagenet-1K-C. We utilized the Imagenet-1K [95] dataset along with Imagenet-1K-C [62]. Imagenet-1K is a large natural image dataset with 1.2 million training images and 50k validation images. Imagenet-1K-C, similarly to CIFAR-10-C, contains corrupted versions of the Imagenet-1K Validation set at

[1]We note that this worst-case approach is chosen for simplicity. We expect that in practice, and in future work, our heuristic can be relaxed to use other assumed label distribution shift priors.

varying severities. Both CIFAR-10-C and Imagenet-1K-C are popular as a measure of robustness to covariate shift.

Liver Ultrasound We utilized a private dataset of ultrasound abdominal images collected at the University of Montreal Hospital from patients under suspicion of fatty liver disease [96]. Images were collected between September 2011 to October 2020 with several devices, which we use to provide an additional and realistic covariate shift for our analysis. Specifically, we train our models on one Philips iU22 device (denoted Philips) and evaluate on another target device, Aplio 500 (Toshiba), alongside the application of corruptions by adding two distinct levels of Gaussian noise to the images. The dataset contains 2076 images (23% without fatty liver) from 89 patients collected with Philips and 1014 images (11% without fatty liver) from a different set of 39 patients collected with Toshiba, and it was used for binary classification, detecting fatty liver versus non-fatty liver.

3.4.1 Training and Architecture Details

On CIFAR-10 we trained a ResNet-26 model as defined in [97]. We used SGD with a batch size of 128. An initial learning rate set to 0.1 is used in combination with a cosine annealing schedule [98] trained over 200 epochs. Weight decay set to 5e-4 was used along with momentum set to 0.9 [99]. Standard augmentation used random crop of size 32 with 4 padding and random horizontal flips.

For Imagenet-1K we utilized a pre-trained Resnet18 model.

For the Liver Ultrasound data, we compared VGG-16 with BatchNorm, and Resnet-18. Both architectures were trained with similar configurations, using a SGD optimizer with learning rate set to 0.001 and momentum 0.9, with a batch size of 32.

Adaptation Details For CIFAR-10 and Imagenet-1K we use a batch size of 500 for the experiments (sampled over multiple seeds). For Liver Ultrasound, the batch size was the entire target set containing 222 images.

Target Label Distributions In order to analyze label distribution shifts we consider two approaches for constructing these imbalances. N-class refers to randomly selecting N classes from the set of all classes. We also utilized a more natural imbalanced dataset simulation construction technique popular in e.g. federated learning literature [100] which samples from a dirichlet distribution with parameter α. Smaller α typically will yield more sample batches with higher imbalances. This allowed us to mimic a

	VGG-16		Resnet18	
Labels proportion	Source	TTN	Base	TTN
Original (23%/77%)	79.2%	80.4%	82.3%	83.1%
0%/100%	93.0%	91.4%	92.2%	84.4%
50%/50%	62.5%	61.3%	71.9%	67.2%
100%/0%	32.0%	26.6%	51.6%	21.9%

Table 1: Illustration of failure case on binary classification medical ultrasound data (detecting presence of fatty liver). If TTN is deployed on target data from the same domain (same equipment) with the original training proportions (23%/77%) performance is maintained or slightly improved. On the other hand, if the target batch presented has a label distribution shift we see substantial degradation, with some models being more severely effected than others.

more natural scenario where the label distribution is not only varying in classes but proportions.

3.5 Experiments

3.5.1 Label Distribution Shift Impact on TTN

We first illustrate the potential pitfalls of TTN on several examples of Label Distribution Shift. Using the CIFAR-10 dataset we show the effect on TTN of an extreme version of label distribution shift in Figure 2. Specifically, we adapt on batches containing one class or all classes (as can be seen in the label distributions shown in Figure 2). We first observe on adaptation to one class batches that the accuracy of TTN degrades severely compared to the source model in cases where there is a covariate shift (application of Gaussian noise to the input) but also *in the absence of covariate shift*. The latter observation is also confirmed in Table 1 on ultrasound data where we modify the target label proportion of the test data coming from the same distribution (but without changing the device or adding noise). This catastrophic failure, even without covariate shift, is a clearly undesired behavior when operating in a realistic deployment environment where target label distribution is unknown and can vary for each incoming batch. Furthermore, we can observe from the per-class performance evaluations of the adapted models that the majority class is disproportionately affected by TTN, and that classes underrepresented (or

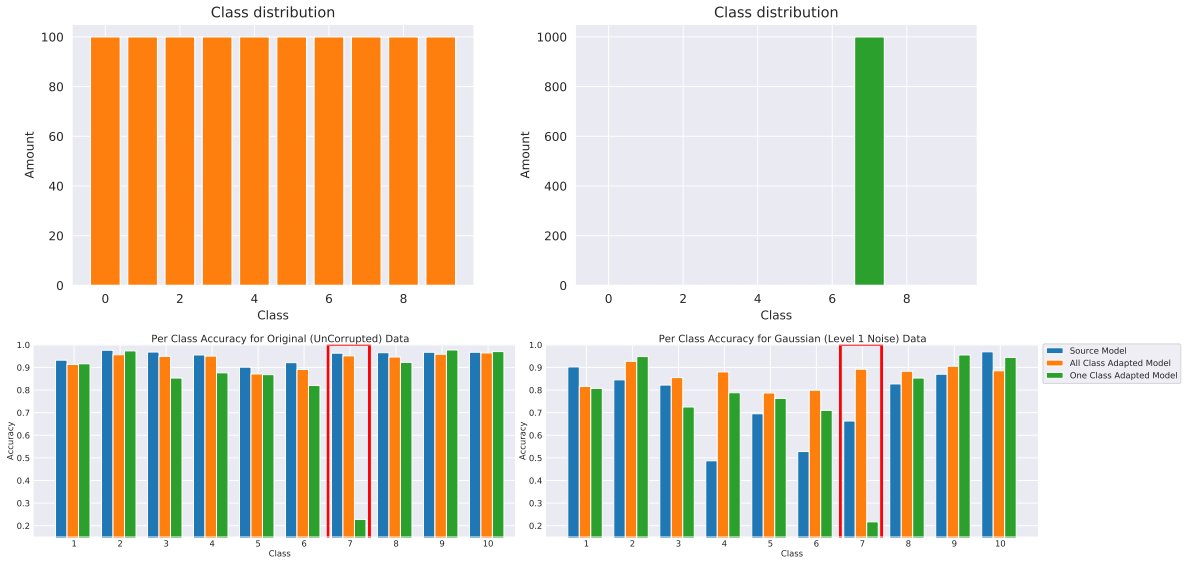

Figure 2: Effect of class imbalance and covariate shift through Gaussian noise on test-time-normalized (TTN) model performance: TTN mitigates distributional shift but greatly suffers from class imbalance. In the top row, two class distributions, uniform and one-class are shown. In the bottom rows we show per-class accuracy plots for a Source model (blue), a class-balanced TTN model (orange), and a class-imbalanced TTN model (green). Observe in the bottom left plot that the class-imbalanced TTN model performs very poorly on the most prevalent class label of the imbalanced adaptation set (label 7, red rectangle). Observe in the bottom right plot that the class-imbalanced TTN model greatly improves performance for Gaussian-corrupted inputs while also greatly suffering from class imbalance. The goal of this paper is to bring the advantage of TTN to the class-imbalanced setting.

not present) will, un-intuitively, experience less or slightly improved over source performance degradation.

We can obtain intuition about this behavior by considering a strong classification model where representations from each class will be well separated and clustered in the final representation layer (see e.g. Fig 1). A biased BatchNorm computation will compute the mean of a specific label cluster and re-center the data by this mean. Indeed, consider a neural network layer whose activations are tightly clustered, such that each cluster corresponds to one label. Further, assume that the different cluster centers occupy approximately orthogonal directions in this space. Such quasi-orthogonality is not an uncommon property in high dimensions. In this setting, linear decision directions would align with cluster means. Subtracting a cluster mean instead of a class-balanced batch mean would then significantly move the label cluster along its decision direction and will likely lead to some examples passing the classification

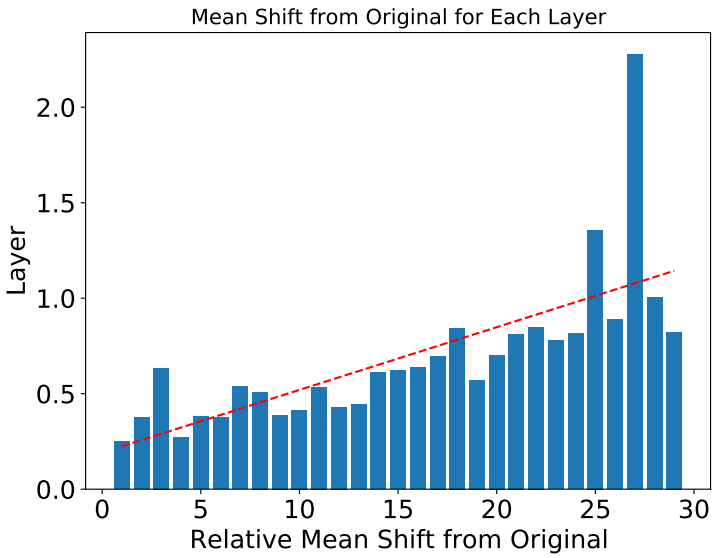

Figure 3: We evaluate the relative shift in the BatchNorm means (measured with a normalized L2 distance) for each layer when adapting to only one class. We observe that as depth increases (and classes expectedly become more well separated) the shift in the mean becomes larger.

boundary. Other classes will be barely affected due to the orthogonality of the cluster centers and hence of the decision boundaries.

3.5.2 Analyzing BatchNorm Stat. Variation in Depth

As mentioned previously, many works in deep learning have shown that supervised trained neural networks become more specialized as depth of a network increases [89–91] and that classes progressively become more separated with depth. Thus, in particular, as the classes separate, their individual means and data points will be farther from the overall data mean. As discussed in Sec. 4.2, this would lead the BatchNorm centering operation to more drastically move points in upper layers, potentially forcing them to cross decision boundaries. For example, Figure 1 illustrates this intuition by showing how class separation changes in depth, and the effect of the mean shift operation in the extreme case where it is computed with respect to just one class in the target batch (i.e., the same extreme shift as will be used in our Hybrid-TTN heuristic).

To empirically confirm this intuition, we attempt to measure the relative shift in the majority class mean at different layers. Results of this experiment are shown in Figure 3. Here we compare $\|u_{orig} - u_{1-class}\|$ for each layer, where u_{orig} are the original training data means and $u_{1-class}$ are means computed from adapting the model on

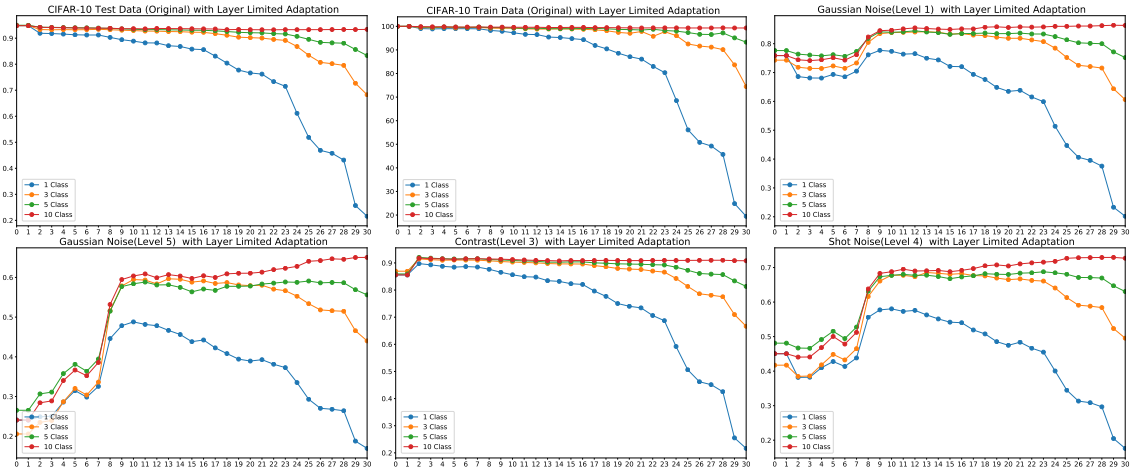

Figure 4: On CIFAR-10 we adapt models only up to the layer shown on the x-axis, the y-axis showing the performance on the target data. We consider target data with different corruptions (and no corruptions) and for each we test with different label distributions. We consider label distributions with all (10) classes as well as 5,3, and 1 randomly selected and balanced classes. Note the x-axis starting value is the source model performance and the ending value the TTN model performance. We observe that adapting only earlier layers can avoid some of the catastrophic collapse due to TTN observed on original data while maintaining the benefits of TTN over the source model in covariate shift.

only one class. We use the Resnet model and CIFAR-10 data from Sec. 4.1. To allow for cross-layer comparisons, we also normalize this ℓ_2 distance by the average distances between class-means in the layer. We observe that, as depth increases, we have larger relative shifts in the mean, confirming our intuition (see Fig. 3).

3.5.3 Adaptation of Early Layers

Having observed that depth can have an effect on the behavior of TTN and taking into account that earlier layers tend to be less specialized we now consider only adapting earlier parts of the network and study this effect on the classification performance under label distribution shift. We perform experiments on layer-limited adaptation for a Resnet-18 model on CIFAR-10 both with and without noise. Our results are shown in in Figure 4. Specifically, *for each index on the x-axis we perform test-time normalization of the model up to this layer*. Consequently, the extremes of the x-axis (0 and, here, 30) correspond to Source and TTN models respectively. First, we observe that on the non-corrupted test and train data the performance of class-imbalanced data degrades gradually at first and increasingly faster towards the later layers. This is in line with our observations of the mean shifts in Sec 4.2. This suggests that

Cifar-10		Covariate Shift			
Label Dist. Shift	Adaptation Method	Original	Corrupt-1	Corrupt-3	Corrupt-5
Original (balanced)	Source	94.8%	86.1%	73.3%	51.7%
	TTN (Δ)	-0.9%	+3.5%	+11.9%	+25.8%
	Hybrid-TTN (Δ)	-0.8%	+0.6%	+10.4 %	+22.7 %
1-class	Source	95.1%	86.7%	73.6%	52.2%
	TTN (Δ)	-73.4%	-66%	-53.7%	-33.7%
	Hybrid-TTN (Δ)	-7.9%	-6.4%	-0.6%	+10.8%
5-class	Source	95.1%	86.9%	74.1%	52.3%
	TTN (Δ)	-12.1%	-8.4%	-.1%	+14.5%
	Hybrid-TTN (Δ)	-1.7%	+1.0%	+8%	+20.3%
Dirichlet alpha = 0.1	Source	95.5%	86.9%	73.7%	51.3%
	TTN (Δ)	-36.7%	-31.4%	-21.4%	-3.6%
	Hybrid-TTN (Δ)	-3.6%	-0.1%	+7.7%	+21.3%
Dirichlet alpha = 0.5	Source	94.0%	85.5%	72.7%	52.9%
	TTN (Δ)	-19.1%	-15.1%	-6.4%	+6.7%
	Hybrid-TTN (Δ)	-2 %	+0.5%	+7.1%	+17.4%

Table 2: CIFAR-10 evaluations on multiple label shifted distributions and covariate shifts (corruptions) with different degrees of label imbalance. Corruptions values are aggregated over 15 corruptions, all values are aggregated over 10 seeds. We show the source model performance and the improvement (or degradation) as a delta with TTN and Hybrid-TTN. Standard error is below 0.05%. We observe that Hybrid-TTN provides benefits over source model when there is no covariate shift, while avoiding catastrophic failures and allowing benefits over source when there are label distribution shifts.

adapting only early layers up to a cutoff can provide a degree of invariance towards label distribution shift. Secondly, for corrupted data we observe that adapting up to earlier layers can allow enough label distribution invariance to provide benefits under covariate shift. For example models that are adapted up to layers 8 through 15 can perform much better than source models under covariate shift but without label distribution shift, while also being able to perform much better than fully adapted and often source models in various label distribution shift scenarios.

Finally we note that the rate of degradation in performance on training data can be predictive of the behavior on test data and corrupted data. For example, we observe some of the largest drops in accuracy on training data at layers 23-25 and 28-29, which are similarly mimicked amongst all the imbalance cases and as well under corruptions. This suggests we can use training data to select the layer that is predictive of the effect of label distribution shift. Furthermore, one label distribution shift scenario is predictive of others: severe relative degradation in 1-class also occur at the same layers in 3-class. This motivates our use of the Hybrid-TTN for which we

Imagenet		Covariate Shift			
Label Dist. Shift	Adaptation Method	Original	Corrupt-1	Corrupt-3	Corrupt-5
Original (balanced)	Source	69.5 ± 0.2 %	51.5 ± 0.3 %	31.1 ± 0.3 %	13.9 ± 0.2 %
	TTN (Δ)	-0.6 ± 0.2 %	+6.4± 0.4 %	+12.3 ± 0.4 %	+11.0± 0.3 %
	Hybrid-TTN (Δ)	+0.8 ± 0.2 %	+2.6± 0.3 %	+3.6± 0.3 %	+1.6 ± 0.2 %
1-class	Source	72.5 ± 2.1 %	52.1 ± 3.6 %	31.5 ± 3.8 %	13.9 ± 2.4 %
	TTN (Δ)	-71.1 ± 0.2 %	-50.8 ± 0.2 %	-30.3 ± 0.2 %	-13.2 ± 0.1 %
	Hybrid-TTN (Δ)	-6.3 ± 2.2 %	-0.9 ± 4.0 %	-1.9 ± 3.7 %	-1.5 ± 2.3 %
5-class	Source	70.7 ± 1.1 %	51.2 ± 2.5 %	29.6 ± 1.8 %	12.0 ± 0.8 %
	TTN (Δ)	-28.9 ± 1.0 %	-17.8 ± 2.0 %	-6.3 ± 1.5 %	+0.9 ±1.1 %
	Hybrid-TTN (Δ)	-1.6 ± 1.0 %	+0.8 ± 2.4 %	+2.2 ± 1.9 %	+0.9 ± 1 %
Dirichlet alpha = 0.01	Source	69.6 ± 1.3 %	50.7 ± 1.4 %	31.3 ± 1.0 %	13.7 ± 0.5 %
	TTN (Δ)	-10.3 ± 1.6 %	-2.8 ± 1.5 %	+3.7 ± 1.2 %	+6.4 ± 0.5 %
	Hybrid-TTN (Δ)	-0.6 ± 1.4%	+1.5 ± 1.4 %	+2.1 ± 1.1 %	+0.8 ± 0.6 %

Table 3: Imagenet-C evaluations on multiple label shifted distributions and covariate shifts (corruptions) with different degrees of label imbalance. Corruptions values are aggregated over 16 corruptions, all values are aggregated over 10 seeds.We observe that Hybrid-TTN provides benefits over source model when there is covariate shift, while avoiding catastrophic failures when there are label distribution shifts. Specifically, we obtain benefits in Original, 5-class, and Dirichlet.

select an adaptation layer based on the training data. We evaluate Hybrid-TTN in the subsequent section.

Threshold Selection To select the layer up to which to apply adaptation in Hybrid-TTN, we utilize the training data from the respective dataset and cut off at the layer at which we find an accuracy drop of $T\%$ or more with respect to the training accuracy of the source model. Specifically, we perform 1-class adaptation starting from layer 1 until we reach a layer that degrades the training accuracy below the threshold. We emphasize that the goal of Hybrid-TTN is to mitigate the effects of label distribution shift while being able to maintain benefits of TTN for a variety of possible label distributions that might be encountered at deployment. In this work we thus set the threshold based on an extreme shift (1-class) and show that this can control for label distribution shift in a variety of less severe label distribution shift scenarios. Since the effect of label distribution shift can be gauged directly from the training data, thresholds can be selected using the most severe label distribution shift expected from the application at inference time by direct simulation on the training data, or on validation data if available. For our experiments we utilize a T of 5% for CIFAR-10 and LiverUltrasound, while for Imagenet we utilize a higher threshold (10%) due to a much larger number of classes.

For CIFAR-10 and Imagenet we observe that Hybrid-TTN is able to control the degradation in performance under a variety of label distribution shifts in the absence

of covariate shift, while TTN leads to serious failures. When corruptions are applied, Hybrid-TTN is able to bring benefits (substantially improve on source model) for original distribution. Unlike TTN, it is able to handle the label distribution shift, in many cases avoiding catastrophic failure, and in a variety of combinations of severe label and covariate shift improving over the source model.

For the ultrasound dataset we similarly observe that Hybrid-TTN can be applied on multiple architectures. Under a shift to a new device Hybrid-TTN can minimize degradation in performance, particularly in Resnet-18, where switching from label 1 to 0 causes a large failure with TTN, but is controlled by Hybrid-TTN. Similarly we observe with corruptions added to the data in the Resnet18 case, Hybrid-TTN can allow controlling for both covariate and label shift. We also observe for Resnet-18 that we can control label shift problems in TTN when going from imbalanced training (77% label 1) to balanced (50% label 1).

Ultrasound		VGG-16			Resnet18		
Device/Corruption Shift	Proportion Label 1	Source	TTN (Δ)	Hybrid-TTN (Δ)	Source	TTN (Δ)	Hybrid-TTN (Δ)
Unseen Device	Orig. (77%)	78.8%	-0.2%	-1.1%	77.7%	+1.1%	+0.6%
	100%	93.4%	-0.3%	-0.6%	91.3%	+2.7%	+0.6%
	50%	64.3%	-2.4%	-2.0%	62.8%	-6.9%	-2.4%
	0%	32.9%	-9.2%	-7.3%	31.4%	-26.1%	-12.6%
	Avg. Shifted	63.5%	-3.9%	-3.3%	61.8%	-10.1%	-4.8%
Corruption Level 1	Orig. (77%)	79.6%	-1.1%	-1.6%	75.5%	+1.6%	+2.8%
	100%	95.8%	-1.5%	-1.8%	82.3%	+12.0%	+9.3%
	50%	62.2%	-0.8%	-0.8%	68.3%	-15.1%	-8.2%
	0%	26.1%	-4.8%	-3.4%	46.9%	-42.1%	-26.6%
	Avg. Shifted	61.4%	-2.4%	-2.0%	65.8%	-15.0%	-8.5%
Corruption Level 2	Orig. (77%)	79.0%	-0.1%	-0.2%	31.9%	+44.7%	+46.6%
	100%	93.1%	+1.5%	+2.4%	13.8%	+82.9%	+78.7%
	50%	61.0%	-1.1%	-0.2%	55.7%	-3.0%	+4.7%
	0%	29.5%	-11.6%	-7.3%	93.7%	-87.9%	-75.3%
	Avg. Shifted	61.2%	-3.7%	-1.7%	54.4%	-2.7%	+2.7%

Table 4: Evaluation on ultrasound data. Binary classification, label 0 for patients without fatty liver, and label 1 for patients with fatty liver disease. Models were trained on one device (Philips iU22) and are evaluated on another (Toshiba Aplio 500). We present evaluations on targets that include just the shift in device as well as the shift in device along with corruptions. We observe that Hybrid-TTN is able to control the degradation in performance under label distribution shift, while allowing improvements over source model when target data has a large covariate shift.

3.6 Conclusions

We have studied a popular batch level Test-time Adaptation method in the context of label distribution shift. We observed that in realistic scenarios where batches at deployment time have label distribution shifts, this method can fail catastrophically and proposed a direction for solving this problem to allow for obtaining the benefits of adaptation without risking catastrophic failure due to label shift.

Chapter 4

Revisiting learnable affines for batch norm in few-shot transfer learning

4.1 Introduction

Large-scale data powers many advancements in automating tasks with machine learning. The usual aim is to collect massive sets of annotated data and have deep learning algorithms learn from this source data to produce models that generalize to some target task. Collecting large amounts of data in domains of significant impacts, such as medical imaging, often requires much expertise, care, or resources to annotate. Privacy constraints may additionally make this approach harder. All this motivates methods for rapidly generalizable learners which can quickly adapt to novel classes. Plenty of work has been conducted on this front, producing few-shot methods that use metric or matching methods to adapt to new classes quickly. Unfortunately, most of these approaches focus on domain-to-domain approaches, and not until recently were benchmarks available for evaluating few-shot methods across domains, including various medical domains.

Developing methods for generalizable few-shot learners across substantial domain shifts will be vital to diagnose diseases better. A corpus of previous work exists in the meta-learning literature, where, running like a vein throughout, a focus exists on batch norms to help adapt to shifts. Several few-shot methods adapt batch norm statistics during few-shot tuning time, likely helping the learner cope with domain

shift. Others condition the features produced by the normalization layers. Another set of researchers studies domain-specific batch normalization.

Our work utilizes two methods in tandem; during pretraining of the backbone, we do not learn the parameters of the affines in normalization layers, which are instead set to identity and fixed; During adaptation on the support set, we enable and directly adapt the affine parameters of the normalization layers of the backbone. We investigate the potential reasons for improvement of the first method by looking at the sparsifying properties of Batch-Norm. The second method has recently seen application in further works.

4.1.1 Related Work

4.1.1.1 Few-Shot learning

In recent years, significant efforts have been directed towards the development of FSL [101–107]. FSL aims to adapt learners to novel classes using only a limited number of labelled samples. Research in FSL has typically been predicated to settings with limited domain shifts between the source and novel classes. Meta-learning techniques have garnered significant attention in FSL based on their coherent and simplistic qualities. Current meta-learning methods can be broadly classified into metric and optimization-based approaches. Metric-based approaches [44, 107–110] utilize the distance between embeddings of the support and query samples to classify the novel query images. Optimization approaches [101, 102] incorporate the fine-tuning component within the representation learning phase. Furthermore, several works propose a transfer learning [106, 111–113] approach following the hypothesis that the base and novel classes share discriminative features. Other methods instead employ model initialization techniques to speed up convergence and improve the classifier, based on the assumption that the initialization which works well on the source domain will be effective on the novel target domain [103, 104].

Recently, research in FSL has focused on settings where there is a significant domain gap between the source and target data [1]. Despite the popularity of meta-learning, Guo et al. [1] demonstrated that the standard transfer learning and fine-tuning approach outperforms current state-of-the-art meta-learning methods when facing a large distribution shift. Furthermore, several methods utilize unlabelled data from the target domain in the evaluation stages in order to reduce the distributional shift [110, 114–116]. Progress in self-training and self-supervised learning methods have

led to promising solutions for CDFSL problems. STARTUP [117] is a notable state-of-the-art approach in distant tasks which employs a combination of self-supervised and self-training components for CDFSL.

4.1.1.2 Batch Normalization

The introduction of BN layers [17] has sped up model convergence and enabled the training of deeper networks. The initial hypothesis stated that BN alleviates the issue of internal covariate shift following the notion that the standardization of features reduces dramatic shifts to the inputs of convolutional layers [17]. Since then, this explanation has been cast into doubt in [118], where internal covariate shift was induced in BN layers to find a negligible effect on BN effectiveness. Another study suggests that the optimization of weight magnitude and direction is decoupled by BatchNorm [119]. Empirical experiments demonstrate that BatchNorm layers smoothen the optimization landscape [118]; while providing a slight regularization effect [120] and aiding in deterring the exploding activations problem [121].

Our work investigates the role of BatchNorm and its affine parameters when facing a shifted domain, particularly in few-shot settings. On domain adaptation, Li et al. [122] use BN layers towards domain adaptation in their AdaBN method. This method assumes that data from different domains will be transformed into representations with similar distributions. The authors of AdaBN present its benefits through empirical experiments carried out on CNNs for image classification tasks. MetaNorm [123] is another BN-based domain adaptation technique that utilizes a meta-learning approach to predict domain-specific BatchNorm statistics for domain-independent batch normalization. Frankle et al. [124] highlight the expressive powers of the BN affine parameters. They conduct experiments that show that BN affine parameters play a positive role in improving model performance. However, their work does not take into consideration settings where there is a distributional gap between the training and target data.

In this paper, we explore the role of affine parameters towards the generalizability of few-shot learners in the presence of a distributional shift between the source training and target data. We perform experiments on state-of-the-art methods such as STARTUP. Furthermore, we adapt AdaBN to an FSL environment to study the effect of the affine parameters on BN-based domain adaptation techniques on cross-domain few-shot transfer.

4.2 Methods

4.2.1 Definitions

Notations here are adopted from the survey paper [27]. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$.

4.2.2 Feature Normalization

Let S be a batch of labelled examples $\{(x_i^s, y_i^s)\}_{i=1}^N$ of size N from a source domain $\mathcal{D}^s$ where $x_i^s \in \mathcal{X}^s$ and $y_i^s \in \mathcal{Y}^s$, and Θ be a deep convolutional neural network consisting of L layers with weight matrices θ^l where l represents the layer index. If h represents the intermediate features of Θ for layer l, the Feature Normalization layer at layer l is computed for each channel and can be defined as[1]:

$$\mathrm{FN}(h_c) = \frac{h_c - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}. \tag{3}$$

Here, subscript c represents the channel index, μ_c and σ_c are the first and second moments of h_c respectively defined as:

$$\mu_c = \frac{1}{NHW} \sum\nolimits_{n,h,w} h_{nchw} \tag{4}$$

$$\sigma_c = \sqrt{\frac{1}{NHW} \sum\nolimits_{n,h,w} (h_{nchw} - \mu_c)^2}, \tag{5}$$

where H and W are the spatial dimensions of h_c.

4.2.3 Fine-tuning affines (Fine-Affine)

In much of the few-shot learning literature [106, 117], only the linear classifier is adapted in the fine-tuning stage, leaving the backbone frozen. Typically this is done to allow for rapid adaptation, but also because fine-tuning the backbone does not improve performance as the model becomes over-parameterized. In another work [125], the affine parameters are utilized to provide task specific conditioning. The affines represent a small number of parameters and may allow the model to adapt without overfitting to the few samples presented in the few-shot fine-tuning stage. It is thus natural to consider adapting both the linear layer and the affine parameters. In this

[1]For the sake of simplicity, we implement the Feature Normalize layer using standard Batch Norm modules with disabled affine parameters.

paper, we refer to the joint fine-tuning of the linear classifier and the affine parameters as Fine-Affine.

4.3 Experiments

We experimented with Feature Normalization on state-of-the-art few-shot learning frameworks, such as STARTUP [117], and present empirical results to evaluate FN in few-shot transfer settings. We adapt AdaBN [122], a BN-based domain adaptation technique, to an FSL setup and investigate the effect of replacing BN with FN. Ablation studies are carried out on the BatchNorm affine parameters γ and β to evaluate their isolated influence towards performance in cross-domain few-shot transfer. The overhead relative to baseline is calculated for all methods to emphasize the computational cost of more complex methods while achieving similar performance gains to FN. We compare the sparsity of feature representations at the end of a trained BN network across datasets and alongside their FN counterparts. Finally, we investigate the effect of reactivating the affine parameters while fine-tuning to the target domain.

4.3.1 Benchmarks

The challenging CDFSL benchmark introduced in [1] is used as the basis for our experiments. MiniImageNet [107], which consists of images based on object recognition tasks, is utilized as the base representation learning dataset. Experiments are conducted on the more extensive ImageNet [126] dataset as well. The benchmark's target data is composed from four datasets, each from very different domains relative to the source images of miniImageNet and ImageNet. The novel datasets consist of EuroSAT (satellite imagery to determine land usage), CropDiseases (plant images to identify botanical diseases), ChestX (chest X-rays to detect pathology), and ISIC2018 (images of skin abrasions to detect melanoma).

For methods with an unsupervised component such as STARTUP and AdaBN, we randomly sample 20% of unlabelled images from novel classes in the target dataset to use in the representation learning phase, following a similar setup to [117] . The remaining samples are used during inference. Similar to [1], we perform experiments in an FSL classification setting where the support set is composed of 5 classes with k samples per class (5-way k-shot), where k $\in \{1, 5, 20, 50\}$ and the overall scores are an average of accuracies over all target datasets for k $\in \{5, 20, 50\}$. Evaluation of models pre-trained on source miniImageNet are carried out over 600 episodes, 95%

confidence intervals with reported mean accuracy. Models that were pre-trained on source ImageNet are evalutated in similar settings, except over 2000 episodes.

4.3.2 Implementation details and variance of runs

The few-shot transfer experiments in Table 5 are carried out on the publicly available CDFSL benchmark [1]. The Baseline is standard transfer learning trained for 400 epochs on miniImageNet with a batch size of 128. STARTUP's teacher model is trained for 400 epochs on miniImageNet and its student model is trained for 1000 epochs on unlabelled samples from 20% of each target dataset, both using a batch size of 256. The remaining 80% of target datasets are utilized for fine-tuning, as described in Sec. 4.3.1. All methods make use of the ResNet-10 architecture [97]. The experiments in this paper were carried out using the Tesla V100 SXM2 16 GB GPU.

For miniImageNet source cases with very-low shot cases such as 1-shot, we observe a high variance in results across different seeds. For instance, on 5 different seeds, the fine-tuned baseline trained in [117] produced the following mean accuracies for 5-way 1-shot classification on EuroSAT: {63.11%; 63.01 %; 61.50%; 62.68 %; 61.91 %}, each with 95% confidence interval of about 0.9 across episodes. We note as well some reported improvements are often in the range of 2-3% in the mean [117], thus we can see the variance due to the training procedure can be higher than typically assumed. In order to take into consideration this high variance that has been unaccounted for in other studies, we average the results obtained from experiments carried out over 5 seeds.

Overhead calculation It is worth noting that different methods have varying computational demands and complexity. To make a fair comparison of the computational costs relative to performance gains, we have calculated the base training and adaptation times as a means of elaborating the cost differences between the evaluated methods. The base training time is evaluated as the amount of time required to train during the representation learning on labelled samples drawn from the source domain $\mathcal{D}^s$ relative to the training time taken by the Baseline method. Having an overhead of 1 is equivalent to the time taken by baseline, while 0.75 indicates that the method only requires 75% of the time taken by the Baseline. Using the same approach, the adaptation time ratio is calculated as the time needed to adapt a single sample of the target domain $\mathcal{D}^t$, either supervised or unsupervised, for each episode relative to the amount of time taken by the Baseline.

Evaluation setting Any inference technique that is dependent on a feature representation and is built with BN in its backbone can be used agnostically with FN. For fairness and simplicity, in this work, we follow the same evaluation setting used for experiments on the CDFSL benchmark [1] and STARTUP [117]. Here, the weights of the feature extractor are frozen after representation training on the base dataset. A linear classifier is then trained on the support set and the model is evaluated on the query set.

4.3.3 BatchNorm related methods

Our work focuses on Batch Normalization and thus we consider two additional approaches not as commonly used in the few-shot literature that are based on adjusting BatchNorm statistics or BatchNorm affine parameters to facilitate more rigid comparisons.

AdaBN few-shot setup AdaBN, introduced in [122], is a lightweight BatchNorm-based domain adaptation technique that has been shown to improve performance on transfer learning methods towards image classification tasks. The method is an unsupervised technique that utilizes unlabelled data from the target domain and adapts the BN statistics to bridge the domain gap between the source and target distributions. Despite the efficacy of this approach in transfer learning, it has been neglected in the few-shot literature. In this study, we evaluate AdaBN in few-shot settings both in near-domain and when facing a significant domain shift, with both BN and FN configurations. AdaBN utilizes the standard Baseline model pre-trained on the source dataset, and adapts for an additional few epochs of forward passes on unlabelled samples $\mathcal{D}^t$. Here, the statistics of the model's normalization layer are updated based on the target feature distribution $p(x)^t$ while the learnable parameters of the model remain frozen.

Adaptation of Affine Parameters We adapt affine parameters. Inspired by [124]. Other methods have learned affine parameters per domain, for a shared set of weights.

4.3.4 Results

Cross-domain few-shot transfer Table 5 reports the results of our experiments on the CDFSL benchmark. Across all datasets and 1, 5, 20, and 50 shot settings (consistent with the CDFSL benchmark), the average performance of the models

		EuroSAT	CropDisease	ISIC	ChestX	Adapt Time	Base-Train Time
				5-way, 1-shot			
Baseline	BN	61.54±0.89	68.87±0.84	31.96±0.60	**22.43±0.40**	1.00	1.00
	FN	**62.61±0.87**	**70.91±0.85**	**32.80±0.61**	22.20±0.40	1.00	1.00
Baseline ✗	BN	61.49±0.91	68.94±0.85	31.77±0.58	**22.54±0.40**	1.00	1.00
	FN	**61.81±0.87**	**71.11±0.86**	**32.58±0.60**	22.33±0.40	1.00	1.00
AdaBN	BN	59.44±0.84	68.07±0.85	**33.82±0.62**	**22.41±0.40**	7.25	1.00
	FN	**63.27±0.86**	**71.50±0.85**	33.67±0.63	22.11±0.39	7.25	1.00
AdaBN ✗	BN	60.40±0.87	68.04±0.85	33.31±0.61	**22.32±0.40**	7.25	1.00
	FN	**63.29±0.88**	**71.32±0.86**	**33.43±0.63**	22.14±0.40	7.25	1.00
STARTUP	BN	63.88±0.84	**75.93±0.80**	32.70±0.60	**23.09±0.43**	1251	1.00
	FN	**64.00±0.88**	74.56±0.85	**35.12±0.64**	22.93±0.43	1251	1.00
				5-way, 5-shot			
MAML*	BN	71.70±0.72	78.05±0.68	40.13±0.58	23.48±0.96	0.70	4.83
ProtoNet*	BN	73.29±0.71	79.72±0.67	39.57±0.57	24.05±1.01	0.35	4.18
Baseline	BN	79.90±0.69	89.93±0.52	43.47±0.60	**26.17±0.43**	1.00	1.00
	FN	**80.51±0.67**	**91.14±0.49**	**45.03±0.62**	25.90±0.43	1.00	1.00
Baseline ✗	BN	79.81±0.71	90.15±0.51	43.11±0.58	**26.39±0.43**	1.00	1.00
	FN	**80.03±0.70**	**91.11±0.49**	**45.34±0.60**	25.78±0.42	1.00	1.00
AdaBN	BN	80.47±0.63	90.11±0.52	**47.97±0.64**	**26.00±0.42**	7.25	1.00
	FN	**82.34±0.62**	**91.29±0.49**	47.92±0.64	25.87±0.43	7.25	1.00
AdaBN ✗	BN	80.39±0.65	89.95±0.51	**46.74±0.61**	**25.93±0.43**	7.25	1.00
	FN	**82.00±0.64**	**90.99±0.50**	47.20±0.62	25.86±0.43	7.25	1.00
STARTUP	BN	82.29±0.60	**93.02±0.45**	47.20±0.61	26.94±0.44	1251	1.00
	FN	**82.51±0.62**	92.86±0.43	**48.54±0.63**	**27.17±0.44**	1251	1.00
				5-way, 20-shot			
MAML*	BN	81.95±0.55	89.75±0.42	52.36±0.57	27.53±0.43	0.70	4.83
ProtoNet*	BN	82.27±0.57	88.15±0.51	49.50±0.55	28.21±1.15	0.35	4.18
Baseline	BN	87.59±0.45	95.83±0.29	54.67±0.58	**32.24±0.46**	1.00	1.00
	FN	**88.31±0.46**	**96.50±0.27**	**56.71±0.59**	32.11±0.46	1.00	1.00
Baseline ✗	BN	88.31±0.48	96.06±0.28	56.62±0.57	**32.58±0.46**	1.00	1.00
	FN	**88.94±0.46**	**96.62±0.26**	**58.92±0.57**	31.88±0.46	1.00	1.00
AdaBN	BN	88.90±0.45	96.03±0.28	59.04±0.60	31.33±0.46	7.25	1.00
	FN	**89.95±0.42**	**96.68±0.27**	**59.65±0.60**	**31.57±0.45**	7.25	1.00
AdaBN ✗	BN	88.87±0.46	95.99±0.28	58.23±0.58	31.58±0.46	7.25	1.00
	FN	**89.91±0.43**	**96.55±0.27**	**59.24±0.59**	**31.68±0.47**	7.25	1.00
STARTUP	BN	89.26±0.43	**97.51±0.21**	58.60±0.58	33.19±0.46	1251	1.00
	FN	**89.63±0.43**	97.43±0.23	**59.98±0.59**	**33.54±0.46**	1251	1.00
				5-way, 50-shot			
ProtoNet*	BN	80.48±0.57	90.81±0.43	51.99±0.52	29.32±1.12	0.35	4.18
Baseline	BN	90.43±0.41	97.58±0.21	60.84±0.56	35.71±0.47	1.00	1.00
	FN	**91.10±0.39**	**98.03±0.19**	**63.17±0.56**	**35.80±0.47**	1.00	1.00
Baseline ✗	BN	91.64±0.39	97.85±0.19	64.29±0.57	**36.25±0.48**	1.00	1.00
	FN	**92.34±0.36**	**98.27±0.17**	**65.90±0.58**	34.81±0.49	1.00	1.00
AdaBN	BN	91.75±0.37	97.77±0.20	63.69±0.58	34.36±0.47	7.25	1.00
	FN	**92.73±0.34**	**98.13±0.19**	**64.56±0.58**	**35.09±0.47**	7.25	1.00
AdaBN ✗	BN	92.04±0.37	97.73±0.20	64.15±0.56	35.08±0.47	7.25	1.00
	FN	**92.86±0.34**	**98.11±0.18**	**65.28±0.56**	**35.18±0.48**	7.25	1.00
STARTUP	BN	91.99±0.36	98.45±0.17	64.20±0.58	36.91±0.50	1251	1.00
	FN	**92.59±0.33**	**98.53±0.16**	**65.90±0.56**	**37.67±0.47**	1251	1.00

Table 5: Few-shot transfer results under extreme distribution shift. All methods make use of a ResNet-10 backbone evaluated over 600 episodes. (BN): BN configuration, linear classifier fine-tuned; (FN): FN configuration, linear classifier fine-tuned; methods marked with ✗: stands for Fine-Affine, linear classifier + affines fine-tuned.; The affines of (FN Fine-Affine) are restored prior to the fine-tuning stage. * Results from [1].

configured with FN exceed that of the BN models. Notably, there is an average improvement of 2.04% for 1 shot classification on the CropDisease dataset when the the Baseline is equipped with FN across 5 seeds. Simply configuring the Baseline model with FN obtains results that rival (within error bars) the more complex and computationally expensive STARTUP, which employs a large amount of unlabelled data to bridge the domain gap. Relative performance gains can be observed across all three methods (Baseline, AdaBN and STARTUP) when equipped with FN. The best overall results were produced by STARTUP with FN. The superior results produced by FN models indicate that the BN affine parameters, γ and β, have a generally negative impact on downstream few-shot transfer tasks when facing a significant domain shift.

	BN	FN	BN Fine-Affine	FN Fine-Affine
	5-way, 1-shot			
EuroSAT	65.17 ± 0.46	**67.04 ± 0.44**	66.32 ± 0.46	**68.69 ± 0.45**
CropDisease	72.98 ± 0.47	**76.97 ± 0.44**	74.01 ± 0.46	**77.52 ± 0.43**
ISIC	29.33 ± 0.29	**30.89 ± 0.31**	31.08 ± 0.32	**31.40 ± 0.31**
ChestX	22.37 ± 0.22	**22.67 ± 0.23**	22.28 ± 0.22	**22.71 ± 0.22**
	5-way, 5-shot			
EuroSAT	84.32 ± 0.31	**86.43 ± 0.28**	84.07 ± 0.34	**86.75 ± 0.29**
CropDisease	91.86 ± 0.25	**93.59 ± 0.23**	91.92 ± 0.25	**94.02 ± 0.22**
ISIC	42.11 ± 0.32	**45.12 ± 0.33**	**47.50 ± 0.36**	46.39 ± 0.33
ChestX	25.38 ± 0.23	**26.22 ± 0.24**	25.21 ± 0.23	**26.39 ± 0.24**
	5-way, 20-shot			
EuroSAT	91.32 ± 0.20	**92.49 ± 0.19**	92.43 ± 0.19	**93.02 ± 0.19**
CropDisease	96.80 ± 0.15	**97.65 ± 0.13**	97.48 ± 0.15	**98.01 ± 0.12**
ISIC	54.53 ± 0.33	**56.92 ± 0.33**	**62.00 ± 0.35**	60.04 ± 0.33
ChestX	29.55 ± 0.24	**30.73 ± 0.24**	30.20 ± 0.26	**31.77 ± 0.26**
	5-way, 50-shot			
EuroSAT	93.55 ± 0.17	**94.34 ± 0.15**	**95.18 ± 0.15**	95.15 ± 0.14
CropDisease	98.09 ± 0.10	**98.62 ± 0.09**	98.86 ± 0.07	**98.88 ± 0.07**
ISIC	60.78 ± 0.31	**63.16 ± 0.31**	**69.05 ± 0.32**	68.25 ± 0.32
ChestX	32.33 ± 0.25	**33.64 ± 0.25**	34.36 ± 0.28	**35.85 ± 0.27**

Table 6: Fine-tuning the linear classifier versus affines + linear classifier. All methods make use of a ResNet18 pre-trained on ImageNet and evaluated over 2000 episodes. BN: BN configuration, linear classifier finetuned; BN Fine-Affine: BN configuration, linear classifier + affines fine-tuned; FN: FN configuration, linear classifier fine-tuned; FN Fine-Affine: FN configuration, linear classifier + affines fine-tuned.

Fine-Affine (Fine-tuning the γ and β) The results of the affine fine-tuning experiment are presented in Table 6. Baseline models equipped with both BN and FN

were evaluated with the Fine-Affine configuration. After the affine parameters of FN Fine-Affine were disabled during the representation learning phase, they are restored and initialized to 1 and 0 for γ and β. The ImageNet dataset was chosen as the source domain on which both the BN and FN models are pre-trained on. It can be observed from the results that there are strong performance gains as a result of the Fine-Affine setup on both BN and FN models, but that FN models still outperform BN models. Improvements are noted for 1, 5, 20 and 50 shot classification across all four datasets, with noteworthy gains of 7.57% on 20-shot classification of ISIC and 2.21% on 50-shot classification of ChestX by the BN Fine-Affine and FN Fine-Affine models respectively. These results suggest that affine parameters are useful towards task specific adaption in few-shot transfer settings, without causing the models to overfit to the small number of samples presented in few-shot environments. The Fine-Affine adaptation was not as effective when using miniImageNet as the source dataset, as observed from Table 5. However, on both ImageNet and miniImageNet base datasets, FN provides a marked improvement over BN on the Fine-Affine method.

Computational overhead From a practical perspective, even though STARTUP produced the overall best results, its adaptation time ratio is 1251 times than taken by the Baseline approach. This is due to an expensive unsupervised learning step. This makes such computationally complex methods inapplicable in tight situations. On the other hand, despite the slow paced base training for MAML and ProtoNet (time ratios to Baseline are 4.83 and 4.18 respectively), they are relatively faster in adaptation time with a lower ratio for MAML (0.70) and a considerably small portion relative to the Baseline time for ProtoNet (0.35). AdaBN is an expensive method compared to MAML and ProtoNet as the adaptation time is larger than MAML's and ProtoNet's sum of adaptation and base training time. In practice, the adaptation time is not of the same scale as the base training and they can not be compared directly. In a real-world scenario, the adaptation happens on much smaller annotated samples from the target domain. Thus it can be considered as a smaller overhead compared to the base training which benefits from a large supervised sample set. And finally, the proposed modification, the FN, resulted in improving other methods, without any exposed extra overhead (even with a slight decrease from a smaller number of parameters).

AdaBN AdaBN is a domain adaptation technique based on BatchNorm that has been adapted to a few-shot environment in this paper. The evaluation of AdaBN

on cross-domain few-shot transfer can viewed in Table 5. The results indicate that AdaBN, with both BN and FN configurations, produce considerable improvements on the ISIC few-shot performance, with a notable 4.86% gain over the baseline on 5-shot classification. While on the other target datasets, AdaBN produced more marginal gains relative to the Baseline. In terms of AdaBN with the BN and FN configuration, FN consistently outperforms BN on most of the experiments. This shows that replacing BN with FN can produce substantial gains for BatchNorm-based domain adaptation techniques when facing a large domain shift.

Post-activation distributions under cross-domain Shifts We hypothesize that the issue with BatchNorm affine parameters under domain shift is related to the sparsifying properties of typical non-linearities like ReLU. A potentially small shift in a neuron's pre-activation output distribution, for example the distribution becoming more peaked, can result in substantial shifts in the post-activation distribution after thresholding. Moreover excessive thresholding can lead to information loss. To obtain further insights, we investigate the average number of non-zero entries (the sparsity) in the feature representations of the penultimate layer of imagenet trained ResNet18 and miniImageNet ResNet10 models under distribution shift. For each model we compute its sparsity on the source data (ImageNet or miniImageNet) and subsequently compare this to the sparsity of other datasets from the CDFSL benchmark. Furthermore, as seen from Table 7, distribution shift (going from imagenet to CDFSL data) tends to induce substantially sparser representations relative to in-distribution data. We hypothesize this excessive sparsity leads to degraded performance and less general features. On the other hand the centered distributions produced by the FN trained models do not have as high a sparsity both for source data and for target datasets motivating their potential for alleviating this issue with affine parameters and distribution shifts.

	ImageNet	Eurosat	ISIC	ChestXRay	CropDisease
ImageNet Pretrained ResNet18 (BN)	53.5	37.2	47.3	58.4	54.2
ImageNet Pretrained ResNet18 (FN)	60.9	53.7	58.9	64.5	62.1
miniImageNet Pretrained ResNet10 (BN)	30.0	16.9	16.2	20.7	30.4
miniImageNet Pretrained ResNet10 (FN)	50.7	26.3	27.6	37.3	47.7

Table 7: The percentage of non-zero entries in the feature maps is computed after the final ReLU activation in each pretrained model. Small changes in the continuous distribution lead to large changes in the discrete distribution. From in-domain to cross-domain transfer, we find that sparsity increases as we move cross-domain.

4.3.5 Ablation studies

As described, the BatchNorm layer consists of two learnable affine parameters whereas the Feature Normalization layer performs normalization in the absence of these affines. In this section, we carry out ablation experiments on these parameters to determine their isolated influence on few-shot transfer performance. The results of the ablation experiments on the CD-FSL benchmark are presented in Table 8. It can be observed that both BN(γ) and BN(β) produce more accurate classification than BN across 1, 5, 20 and 50 shots on all four datasets. The margin of improvement is higher on BN(β) relative to BN(γ). Feature Normalization, where both γ and β are removed, is the best performing configuration for distant domain few-shot transfer.

	BN (γ, β)	BN (γ)	BN (β)	FN (ours)
	5-way, 1-shot			
EuroSAT	65.17 ± 0.46	66.67 ± 0.80	66.69 ± 0.80	**67.04 ± 0.44**
CropDisease	72.98 ± 0.47	75.32 ± 0.88	75.68 ± 0.84	**76.97 ± 0.44**
ISIC	29.33 ± 0.29	30.11 ± 0.54	29.41 ± 0.55	**30.89 ± 0.31**
ChestX	22.37 ± 0.22	22.62 ± 0.39	22.47 ± 0.41	**22.67 ± 0.23**
	5-way, 5-shot			
EuroSAT	84.32 ± 0.31	85.56 ± 0.52	86.18 ± 0.52	**86.43 ± 0.28**
CropDisease	91.86 ± 0.25	92.91 ± 0.47	93.09 ± 0.43	**93.59 ± 0.23**
ISIC	42.11 ± 0.32	44.48 ± 0.58	43.26 ± 0.59	**45.12 ± 0.33**
ChestX	25.38 ± 0.23	26.09 ± 0.43	26.01 ± 0.44	**26.22 ± 0.24**
	5-way, 20-shot			
EuroSAT	91.32 ± 0.20	91.73 ± 0.35	92.11 ± 0.34	**92.49 ± 0.19**
CropDisease	96.80 ± 0.15	97.26 ± 0.26	97.51 ± 0.23	**97.65 ± 0.13**
ISIC	54.53 ± 0.33	56.41 ± 0.59	56.25 ± 0.60	**56.92 ± 0.33**
ChestX	29.55 ± 0.24	30.26 ± 0.43	30.15 ± 0.44	**30.73 ± 0.24**
	5-way, 50-shot			
EuroSAT	93.55 ± 0.17	93.59 ± 0.29	94.11 ± 0.27	**94.34 ± 0.15**
CropDisease	98.09 ± 0.10	98.31 ± 0.19	98.57 ± 0.16	**98.62 ± 0.09**
ISIC	60.78 ± 0.31	62.46 ± 0.58	**63.25 ± 0.57**	63.16 ± 0.31
ChestX	32.33 ± 0.25	33.03 ± 0.45	32.60 ± 0.46	**33.64 ± 0.25**

Table 8: Ablation studies on the affine parameters of the Batch Norm layer. All methods utilize a ResNet18 backbone pre-trained on the ImageNet dataset and evaluated over 2000 episodes. BN (γ, β): Standard BN configuration; FN: FN configuration; BN (γ): BN with disabled β; BN (β): BN with disabled γ.

4.4 Summary

Feature Normalization layers improve few-shot generalization performance on shifted domains by leveraging a smaller number of model parameters. By stabilizing the output distribution of convolutional layers, Feature Normalization improves robustness against distributional shifts. It captures and normalizes the statistical distribution of data features while preventing the affine from overfitting to the training source labels. Feature Normalization is consistent with widely used BatchNorm implementations and can be easily integrated into existing CNN architectures. It is observed that the proposed normalization technique only helps in few-shot transfer and the effect is more pronounced as the data distribution shifts.

Chapter 5

Parametric scattering networks

5.1 Introduction

Small sample settings are standard in hospital systems, where data is limited and expensive to collect for specific tasks. For example, internal fetal heart rate monitoring requires meeting specific conditions and can pose risks of injury to or disease transmission between infants and birthing parents. Additionally, privacy constraints disable data sharing across hospitals. Strong laws and regulations, such as HIPPA and PIPEDA, prevent data sharing across hospitals, and procedures for releasing medical data take time. Pertinent critiques underlie approaches to learning from massive data. Given the lack of scope in vast piles of data, often from a natural image domain and not medical imaging, some question whether deep learning researchers provide evidence for any valid hypothesis embedded within the data. Furthermore, the lack of labeled data in most medical imaging datasets does not allow for applying most deep-learned methods, and the domain shift of medical images complicates the design of methods robust to small sample settings. Building methods that lead to effective small-sample learners across domains would allow healthcare practitioners to navigate the lack of data and privacy constraints.

One of the most interesting of these low-sample methods is Scattering Transforms. Scattering Transforms provide a deformation-stable translation-invariant representation of input signals that work well in the low data regime and provide an excellent theoretical model of single and two-layered trained neural networks. The layers of a scattering transform mimic the layers of a convolutional neural network, with the non-linearities being modulus non-linearities and the convolutional filters being Morlet wavelets. These wavelets are spread in a fixed tight-frame layout enabling the

Scattering representations to be discriminative. The Scattering Transform originates as a feature extractor suitable for natural image classification and texture recognition, being transferred and evaluated successfully for specific small sample medical imaging settings.

Given that the intention of the design choices of the Scattering Transform is to induce similarities with trained neural networks, one natural question after all the literature is, "How vital is the tight-frame configuration of Scattering Transforms in small sample medical imaging settings?". We investigate this question, making our wavelets adaptable by allowing the parameters that generate each wavelet to be learnable. We call the Scattering Network with learnable wavelets a Parametric Scattering Network. We study the deformation properties of the learned wavelets across domains, finding that the problem-specific wavelets keep similar properties to the tight-frame wavelets that parameterize a scattering network. We then evaluate our parametric scattering networks on detecting Covid from ChestXrays and investigate unsupervised methods for training parametric scattering networks.

5.2 Related work

Learning useful representations from little training data [127] is arduous and a reality in a variety of domains such as in biomedicine and healthcare. Recent works have tried to tackle this problem. Lezama et al. [128] replace the categorical cross-entropy loss with a geometric loss called Orthogonal Low-rank Embedding (OLÉ) to reduce the intra-class variance and enforce inter-class margins. Barz and Denzler [129] also propose to replace the categorical cross-entropy loss, but this time with the cosine loss function in order to decrease overfitting in the small-sample classification settings. The cosine loss function, as opposed to the softmax function used with cross-entropy, does not push the logits of the true class to infinity. Other methods show promise by incorporating prior knowledge into the model [130–134]. Oyallon et al. [135] introduce hybrid networks where the scattering transform with fixed wavelets was shown to be an effective replacement for early layers of learned convolutional networks on a wide residual network architecture. Cotter and Kingsbury [136] also propose a hybrid network called a learnable ScatterNet, where learning layers are intermixed between the scattering orders, unlike our work where only a few parameters governing the wavelet construction are modified. Ulicny et al. [137] propose Harmonic Networks (HN), a hybrid network consisting of fixed Discrete Cosine Transform filters combined with learnable weights in CNNs.

Related to our work, adding learnable components to existing wavelet-based representations has been considered in a number of recent works in the context of time-series [138–141]. Balestriero et al. [138] and Seydoux et al. [139] learn a spline-parametrized mother wavelet for 1D problems. Similarly, Cosentino and Aazhang [140] parametrized the group transform in the context of time-series data. Our work, alternatively, focuses on 2D problems and maintains the canonical Morlet wavelet parameterization, but allows deviation from a tight-frame filter bank.

5.3 Methods

We first revisit the formulation of traditional scattering convolution networks in Sec. 5.3.1 and introduce our parametric scattering transform in Sec. 5.3.2. Finally, Sec. 5.3.3 discusses scattering parameter initialization.

5.3.1 Scattering Networks

For simplicity, we focus here on 2D scattering networks up to their 2nd order. Subsequent orders can be computed by following the same iterative scheme, but have been shown to yield negligible energy [51]. Given a signal $x(u)$, where u is the spatial position index, we compute the scattering coefficients S^0x, S^1x, S^2x, of order 0, 1, and 2 respectively. For an integer J, corresponding to the spatial scale of the scattering transform, and assuming an $N \times N$ signal input with one channel, the resulting feature maps are of size $\frac{N}{2^J} \times \frac{N}{2^J}$, with channel sizes varying with the scattering coefficient order (i.e., 1 channel at order 0, JL channels at order 1 and $L^2J(J-1)/2$ channels at order 2).

To calculate 0th-order coefficients, we consider a low-pass filter ϕ_J with a spatial window of scale 2^J, such as a Gaussian smoothing function. We then convolve this filter with the signal and downsample by a factor of 2^J to obtain $S^0x(u) = x * \phi_J(2^J u)$. Due to the low-pass filtering, high-frequency information is discarded here and is recovered in higher-order coefficients via wavelets introduced as in a filter bank.

Morlet wavelets are a typical example of filters used in conjunction with the scattering transform, and are defined as

$$\psi_{\sigma,\theta,\xi,\gamma}(u) = e^{-\|D_\gamma R_\theta(u)\|^2/(2\sigma^2)}(e^{i\xi u'} - \beta), \tag{6}$$

where β is a normalization constant to ensure wavelets integrate to 0 over the spatial domain, $u' = u_1 \cos\theta + u_2 \sin\theta$, R_θ is the rotation matrix of angle θ and $D_\gamma = \begin{pmatrix} 1 & 0 \\ 0 & \gamma \end{pmatrix}$.

Table 9: Canonical Parameters of Morlet wavelet

Param	Role	Param	Role
σ	Gaussian window scale	θ	Global orientation
ξ	Frequency scale	γ	Aspect Ratio

The four parameters can be adjusted and are presented in Table 9. From one wavelet $\psi_{\sigma',\theta',\xi',\gamma'}(u)$, the traditional wavelet filterbank is obtained by dilating it by factors 2^j, $0 \leq j < J$, and rotating by L angles θ equally spaced over the circle, to get $\{2^{-2j}\psi_{\sigma',\theta',\xi',\gamma'}(2^{-j}R_\theta(u))\}$, which is then completed with the lowpass ϕ_J. This can be written in terms of the parameters in Table 9 as $\psi_{2^j\sigma',\theta'-\theta,2^{-j}\xi',\gamma'}(u) = \psi(2^{-j}R_\theta(u))$. By slight abuse of notations, we use ψ_λ here, $\lambda = (\sigma_j, \theta, \xi_j, \gamma_j)$, to denote such wavelets indexed by θ and j. The resulting set of filters is visualized in the frequency domain in Figure 6.

First-order scattering coefficients are calculated by first convolving the input signal with one of the generated complex wavelets (i.e., indexed by the parameters in Table 9) and downsampling the resulting filtered signal by the scale factor 2^{j_1} of the wavelet chosen. Then, a pointwise complex modulus is used to add nonlinearity, and the resulting real signal is smoothed via a low-pass filter. Finally, another downsampling step is applied, this time by a factor of 2^{J-j_1}, to obtain an optimally compressed output size. Mathematically, we have

$$S^1x(\lambda_1, u) = |x * \psi_{\lambda_1}| * \phi_J(2^J u). \tag{7}$$

The resulting feature map has $J \cdot L$ channels, based on the number of wavelets in the generated family. Second-order coefficients are generated similarly, with the addition of another cascade of wavelet transform and modulus operator before the low-pass smoothing, i.e.,

$$S^2x(\lambda_1, \lambda_2, u) = ||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| * \phi_J(2^J u). \tag{8}$$

Due to the interaction between the bandwidths and frequency supports of first and second order, only coefficients with $j_1 < j_2$ have significant energy. Hence, the second-order output yields a feature map with $\frac{1}{2}J(J-1)L^2$ channels.

5.3.2 Morlet Canonical Parameterization

While the wavelet filters are traditionally fixed, we let the network learn the optimal parameters of each wavelet. In other words, we constrain our filters to always be

Morlet wavelets by only optimizing the parameters in Table 9. We call this approach the Morlet canonical parameterization of the wavelet. We adapted the Kymatio software package [142] to create the learnable scattering network.

5.3.3 Initialization

To evaluate the importance of the standard wavelet construction, we consider two initializations and study their impact on resulting performance in both learned and nonlearned settings. First, the standard wavelet construction follows common implementations of the scattering transform by setting $\sigma_{j,\ell} = 0.8 \times 2^j$, $\xi_{j,\ell} = \frac{3\pi}{4} 2^{-j}$, and $\gamma_{j,\ell} = \frac{4}{L}$ for $j = 1, \ldots, J$, $\ell = 1, \ldots, L$, while for each j, we set $\theta_{j,\ell}$ to be equally spaced on $[0, \pi)$. The construction ensures the resulting filter bank forms an efficient tight frame. Thus, we call this construction the tight-frame initialization. Second, as an alternative, we consider a random initialization where these parameters are sampled as $\sigma_{j,\ell} \sim \log(U[\exp 1, \exp 5])$, $\xi_{j,\ell} \sim U[0.5, 1]$, $\gamma_{j,\ell} \sim U[0.5, 1.5]$, and $\theta_{j,\ell} \sim U[0, 2\pi]$. That is, orientations are selected uniformly at random on the circle, the filter width σ is selected using an exponential distribution across available scales and the spatial frequency ξ is chosen to be in the interval $[0.5, 1]$, which lies in the center of the feasible range between aliasing ($> \pi$) and the fundamental frequency of the signal size ($2\pi/N$ where N is the number of pixels). Finally, we select the *aspect ratio* variable to vary around the spherical setting of 1.0, with a bias towards stronger orientation selectivity (0.5) compared to lesser orientation selectivity (1.5).

However, the global orientation of the L filters for each scale are set to be $[\Theta, \Theta + \frac{\pi}{L}, \Theta + \frac{2\pi}{L}, \ldots, \Theta + \frac{(L-1)\pi}{L}]$.

5.4 Experiments

Our empirical evaluations are based on three image datasets: CIFAR-10, COVIDx CRX-2, and KTH-TIPS2. CIFAR-10 and KTH-TIPS2 are natural image and texture recognition datasets, correspondingly. They are often used as general-purpose benchmarks in similar image analysis settings [143, 144]. COVIDx CRX-2 is a dataset of X-ray scans for COVID-19 diagnosis; its use here demonstrates the viability of our parametric scattering approach in practice, e.g., in medical imaging applications.

We evaluate the use of the parametrized scattering with two common models. In the first case, we consider the scattering as feeding into a simple linear model (denoted LL). The LL configurations are used to evaluate the linear separability of the

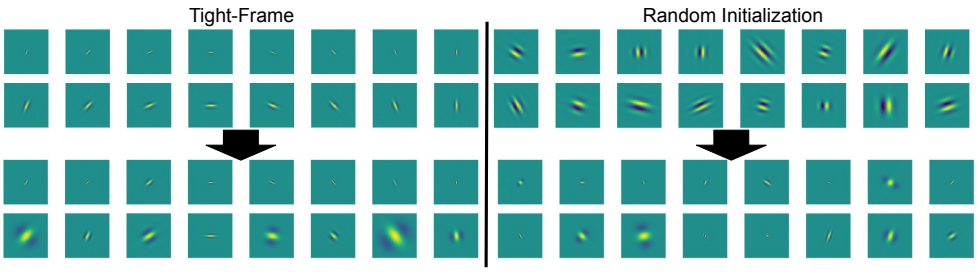

Figure 5: **Initialized wavelet filters pre and post-training.** Real part of Morlet wavelet filters initialized with *tight-frame* (left) and *random* (right) schemes before (top) and after (bottom) training. The filters were optimized on the entire CIFAR-10 training set with linear model. We use the Morlet canonical wavelet parameterization. For the tight-frame filters, we observe substantial changes in both scale and aspect ratio. On the other hand, all random filters undergo major changes in orientation and scale.

obtained scattering representations and have the added benefit of providing a more interpretable model. In the second case, we take the approach of [135] and consider the scattering as the first stage of a deeper CNN, specifically a Wide Residual Network (WRN) [145].

For both models (LL and WRN), we compare learned parametric scattering networks (LS) to fixed ones (S). For learned scattering (LS), we consider the *Morlet canonical* parameterization as described in Sec. 5.3.2. To show the importance of the parametric approach, we also ablate the naive parameterization where all pixels of the wavelets are adapted, which we refer to as a pixel-wise parameterization. For each scattering architecture, we consider both random and tight-frame (TF) initialization. The fixed scattering models determined by the TF construction are equivalent to traditional scattering transforms. Finally, we also compare our approach to a fully learned WRN (with no scattering priors) and ResNet-50 [97] applied directly to input data. We note that the latter is unmodified form its ImageNet architecture and that we do not initialize it with pre-trained weights.

Across all scattering configurations, a batch-normalization layer with learnable affine parameters is added after all scattering layers. Classification is performed via a softmax layer yielding the final output. All models are trained using cross-entropy loss, minimized by stochastic gradient descent with momentum of 0.9. Weight decay is applied to the linear model and to the WRN. The learning rate is scheduled according to the one cycle policy [146]. We replicate some of the experiments with learnable scattering networks followed by WRN on CIFAR-10, COVIDx-CRX2, and KTH-TIPS2 using the cosine loss function [129].

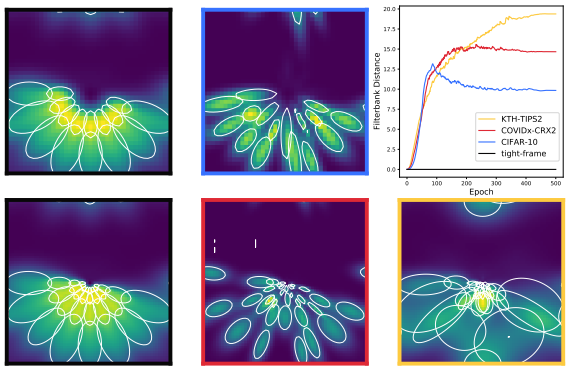

Figure 6: **Parametric scattering network learns dataset specific filters.** The graph (top right) shows the *filterbank distance* over epochs as the filters are trained on different datasets. We visualize dataset specific parameterizations of scattering filterbanks (border colors from the legend) in Fourier space. The x and y axis are the frequency axis. Scattering filters optimized for natural (CIFAR-10) and medical image (COVIDx CRX2) become more orientation-selective, i.e., thinner in the Fourier domain. On the other hand, filters optimized for texture discrimination (KTH-TIPS2) become less orientation-selective and deviate most from a tight-frame setup.

5.4.1 Exploring Dataset-specific Parameterizations

We first compare dataset-specific Morlet wavelet parameterizations and evaluate their similarities to a tight frame. Specifically, we train our parametric scattering networks using the canonical Morlet wavelet formulation with a linear classification layer and qualitatively compare the similarities of the learned filter bank to the tight-frame initialization. To facilitate quantitative comparison, we use a distance metric for comparing the sets of Morlet wavelet filters and Morlet wavelet filterbanks (i.e., scattering network instantiations), allowing us to measure deviations from the tight-frame initialization.

We evaluate distances between two individual Morlet wavelets as $\Upsilon(M_1, M_2) = \left\|(\sigma_1, \xi_1, \gamma_1)^T - (\sigma_2, \xi_2, \gamma_2)^T\right\|_2 + \text{arcdist}(\theta_1, \theta_2)$ where $M_i = (\sigma_i, \xi_i, \gamma_i, \theta_i)^T$ denotes the parameterization of the Morlet wavelet. We use the arc distance on the unit circle to compare values of theta. Since the set of learned scattering filters does not have a canonical order, to compare a learned scattering network to the tight frame scattering network, we use a matching algorithm to match one set of filters to another. Specifically, we first compute Υ between all combinations of filter pairs from both networks, then use a minimum cost bipartite matching algorithm [147] to find the minimal distance match between the two sets of filters. The final distance we use as a notion of similarity between two scattering networks is the sum of Υ for all matched pairs in the bipartite graph. Henceforth, we will refer to this distance as the *filterbank distance*.

The graph in Figure 6 leverages the *filterbank distance* to show the evolution of scattering networks initialized from a tight frame and trained on different datasets. Each network is trained on 1188 samples of its respective dataset (the standard size for KTH-TIPS2). All filters deviate quickly from a tight frame, but KTH-TIPS2's keep changing the longest and ultimately deviate the most. We also observe that filters initialized with the random initialization of Sec. 5.3 become more similar to our tight-frame initialization during the course of training.

On the left-hand side of Figure 6, we visualize the dataset-specific scattering network parameterizations in Fourier space. White contours are drawn around each Morlet wavelet for clarity. The top black border corresponds to tight-frame initialization at J=2, shown for comparison to CIFAR-10 in blue (also J=2). The bottom black border corresponds to tight-frame initialization at J=4, shown for comparison to COVIDX-CRX2 red and KTH-TIPS2 yellow (both J=4).

The filters optimized on the KTH-TIPS2 texture dataset (yellow) become less orientation-selective (wider in Fourier space) than the tight-frame initialization, with filters at J=0 becoming the least orientation-selective of the whole filter bank. In contrast, the filters optimized on COVIDx-CRX2 become more orientation-selective in general i.e., thinner in Fourier space. The filters optimized on CIFAR-10 mirror those optimized on COVIDx-CRX2, also becoming more orientation-selective than their tight-frame counterparts. We suspect that this is due to a reliance on edges for object classification datasets, which seem to require more orientation-selective filters. On the other hand, the morlet wavelets optimized for texture classification seem to discard some edge information in favor of less orientation-specific filters. Each dataset-specific parameterization seems to discard unneeded information from the tight-frame initialization in favor of accentuating problem-specific attributes. Nonetheless, a tight-frame does constitute a good starting point for learning. Indeed, the dataset-specific parameterizations for COVIDX-CRX2 and KTH-TIPS2 are, visually, very different, yet they move similar filterbank distances from the tight-frame initialization (see fig.6), which are small relative to the distances observed for randomly initialized and trained models.

5.4.2 Robustness to Deformation

In [52], it is shown that the scattering transform is stable to small deformations of the form $x(u - \tau(u))$ where $x(u)$ is a signal and τ a diffeomorphism. Given the substantial changes to the filter composition in the learning process, we ask now

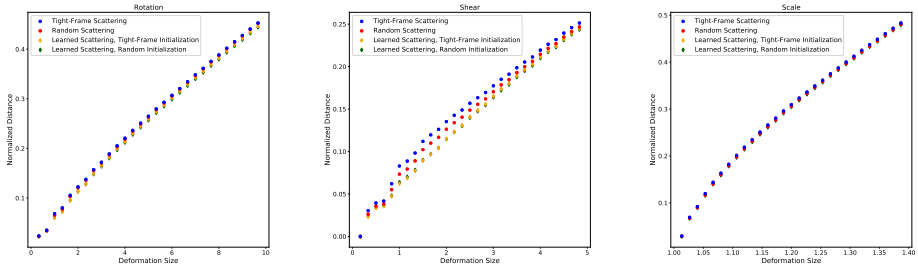

Figure 7: **Normalized distances between scattering representations of an image and its deformation.** Our parametric scattering transform shares similar stability to deformations as the scattering transform.

whether these seem to significantly deviate from the stability result obtained from the carefully handcrafted construction proposed in [52] and extensively used in previous work e.g., [51, 148]. To evaluate the robustness of our parametric scattering networks to different geometric distortions, we apply several tractable deformations to a chest X-ray image x with varying deformation strength and encode all images with different (learned and fixed) scattering networks. The learned ones are trained using the Morlet canonical wavelet formulation with a linear classification layer. The transformed image is denoted by $\tilde{x}$. For each of the different deformation strengths, we plot the Euclidean distance between the scattering feature constructed from the original image $S(x)$ and the scattering feature constructed from the transformed image $S(\tilde{x})$. We then normalize the obtained distance by $S(x)$ to measure the relative deviation in scattering coefficients (handcrafted or learned). Figure 7 demonstrates representative results for a small rotation, shear and scale on images from the COVIDx datasets. We observe that the substantial change in the filter construction retains the scattering robustness properties for these simple deformations, thus indicating that the use of learned filters (instead of designed ones) does not necessarily detract from the stability of the resulting transform.

5.4.3 Unsupervised Learning of Parameters

We have studied the adaptation of the wavelet parameters towards a supervised task. We now perform a preliminary investigation to determine if the scattering representation can be improved in a purely unsupervised manner. We consider the recently popularized SimCLR framework [149], which encourages representations from two data augmentations of the same input to lie close together. We learn scattering network parameters with the canonical Morlet parameterization on CIFAR-10 using

Method	100 samples	500 samples	1000 samples	All
Scattering (Fixed)	36.01 ± 0.55	48.12 ± 0.25	53.25 ± 0.24	65.58 ± 0.04
Unsupervised Learnt Scattering	$\mathbf{38.05} \pm 0.45$	$\mathbf{52.92} \pm 0.28$	$\mathbf{57.76} \pm 0.25$	$\mathbf{68.47} \pm 0.04$

Table 10: Scattering and learned unsupervised scattering features evaluated by training a linear classifier on CIFAR-10. We observe the unsupervised learned scattering improves the representation.

this unsupervised objective function and subsequently evaluate the discriminativeness of the features under a standard linear evaluation experiment on the full CIFAR-10 dataset and in the small data regimes comparing them to the standard scattering transform. The results are shown in Table 10. We observe the unsupervised learning of filter parameters can improve the scattering representation under standard unsupervised learning evaluation protocols.

5.4.4 Computational and memory complexity

The computational complexity of scattering networks and parametric scattering networks is directly related to the FFT (Fast Fourier transform), $O(N \cdot \log(N))$ for an image of size (N $\times$ N). In practice, the computational and memory complexity of our parametric scattering networks varies due to a number of factors.To summarize these factors, we compare runtime (higher is faster), memory, and parameter count per architecture and image size in Table 11. The models were trained using an NVIDIA Tesla T4 GPU. We observe that fixed scattering is two to three times faster than learned scattering for all image sizes and hybrid models. In contrast, WRN-16-8 is faster than LS+WRN at image size 32^2, but slower for larger images. This is due to the scattering transform's substantial spatial dimension reduction, which leads to speed and memory benefit versus regular CNNs [52]. While gradient computation of Morlet parameters adds compute overhead, learned scattering is still efficient with much fewer parameters than CNNs.

5.5 Summary

Limitations There are two limitations to this study that could be addressed in future research. First, the current implementation is limited to two-dimensional data. In future work, the implementation could naturally be extended to one-dimensional and three-dimensional data. Second, for popular datasets, such as CIFAR-10, there are pre-trained models available. In the study, to compare performance with our approach, we considered a fully learned WRN-16-8 and ResNet-50, but we did not

Architecture	Img. size	Train ($\frac{\text{imgs}}{\text{sec}}$)	Infer. ($\frac{\text{imgs}}{\text{sec}}$)	GPU Mem. (GB)	#Params (Million)	
LS+L	32^2	264	542	0.3	0.2	CIFAR
S+L	32^2	650	656	0.1	0.2	
LS+WRN	32^2	232	430	1.6	22.6	
S+WRN	32^2	498	588	1.4	22.6	
WRN-16-8*	32^2	510	1695	4.2	11.0	
LS+L	128^2	42	102	5.4	0.8	KTH
S+L	128^2	123	123	0.5	0.8	
LS+WRN	128^2	42	101	8.5	23.8	
S+WRN	128^2	115	120	3.3	23.8	
WRN-16-8*	128^2	31	90	61.1	11.0	
LS+L	224^2	24	72	13.7	0.5	COVID
S+L	224^2	78	79	1.3	0.5	
LS+WRN	224^2	22	67	16.1	23.7	
S+WRN	224^2	58	71	2.9	23.7	
WRN-16-8*	224^2	10	36	49.6	11.0	

Table 11: Comparison of training runtime, inference runtime, GPU memory, and parameter count per architecture and image size. $*$ from [2]

consider pre-trained models.

Conclusion This work showcases the competitive results of adapting a small number of Morlet wavelet filter parameters in scattering convolutional networks [51]. We demonstrate that filters learned by parametric scattering can be interpreted in the context of specific tasks (e.g., becoming thinner in object classification tasks that require sensitivity to edges). We also empirically demonstrate that our parametric scattering transform shares similar deformation stability to the traditional scattering transform. Overall, we find that our hybrid parametric scattering architectures (with LL and WRN) achieve state-of-the-art classification results in the low-data regime. These results verge upon bridging the gap between the handcrafted filter design in traditional scattering transforms, which provides tractable properties and supports low-parameter models, and fully parameterized convolutional neural networks, which lack interpretability but are more flexible.

Chapter 6

Conclusion

The problems we explore are related to domain shifts and limited data settings. Existing approaches involve Test Time Normalization, Few Shot methods, and Scattering Networks. These approaches are unable to adapt to unexpected domain shifts. This work explores several ways to adapt to these distribution shifts.

In the first contribution, we show the effect of adapting Test Time Normalization models to imbalanced label distribution and investigate this failure. TTN models fail on the particular class but generalize well on other classes with or without the covariate shift applied. Studying the shift over layers shows a trend toward increasing separation between the class means. We propose to avoid this earlier problem by adapting only the earlier, closer-to-input layers of the model, which preserve much covariate shift robustness and mitigate the impact of the label shift. Future work will look into applying this work to other normalization methods, methods that use TTN as a base, or federated learning settings with BatchNorm.

In the second contribution, we adapt the affine parameters of BatchNorm to cross-domain few-shot settings and conduct experiments on the sparsifying properties of BatchNorm affine parameters. This method is successful in Cross Domain settings, but improvements level off in near-domain settings or with additional data. Future work will look into applying this method and better studying what leads to improved performance.

In the third contribution, we parameterize scattering networks, learning a family of wavelets specified to a problem. Future work will look into what other contexts we can apply our Parametric Scattering Networks, such as Bayesian settings where having minimal parameters is vital. Ablating certain parts of the Parametric Scattering Network, or further studying the invariance and stability properties of the network, or parametrizing the scattering to different modalities will glean additional insights.

References

[1] Y. Guo, N. C. Codella, L. Karlinsky, J. V. Codella, J. R. Smith, K. Saenko, T. Rosing, and R. Feris, "A broader study of cross-domain few-shot learning," ECCV, 2020. xiii, 8, 26, 29, 30, 31, 32

[2] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016. xiv, 48

[3] J. Dockès, G. Varoquaux, and J. Poline, "Preventing dataset shift from breaking machine-learning biomarkers," *CoRR*, vol. abs/2107.09947, 2021. 2

[4] N. Karani, E. Erdil, K. Chaitanya, and E. Konukoglu, "Test-time adaptable neural networks for robust medical image segmentation," *Medical Image Analysis*, vol. 68, p. 101907, feb 2021. 2

[5] J. Schrouff, N. Harris, O. Koyejo, I. Alabdulmohsin, E. Schnider, K. Opsahl-Ong, A. Brown, S. Roy, D. Mincu, C. Chen, A. Dieng, Y. Liu, V. Natarajan, A. Karthikesalingam, K. A. Heller, S. Chiappa, and A. D'Amour, "Maintaining fairness across distribution shift: do we have viable solutions for real-world applications?," *CoRR*, vol. abs/2202.01034, 2022. 2

[6] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. `http://www.deeplearningbook.org`. 2, 8

[7] A. R. Tuor, S. P. Kaplan, B. J. Hutchinson, N. M. Nichols, and S. M. Robinson, "Deep learning for unsupervised insider threat detection in structured cyber security data streams," 2

[8] G. Roberts, S. Haile, R. Sainju, D. Edwards, B. Hutchinson, and Y. Zhu, "Deep learning for semantic segmentation of defects in advanced stem images of steels," *Scientific Reports*, vol. 9, 09 2019. 2

[9] B. Hutchinson, L. Deng, and D. Yu, "A deep architecture with bilinear modeling of hidden representations: Applications to phonetic recognition," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4805–4808, 2012. 2

[10] R. Cosbey, A. Wusterbarth, and B. Hutchinson, "Deep learning for classroom activity detection from audio," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3727–3731, 2019. 2

[11] E. Skomski, J.-Y. Lee, W. Kim, V. Chandan, S. Katipamula, and B. Hutchinson, "Sequence-to-sequence neural networks for short-term electrical load forecasting in commercial office buildings," *Energy and Buildings*, vol. 226, p. 110350, 2020. 2

[12] R. Olney, M. Kounkel, C. Schillinger, M. T. Scoggins, Y. Yin, E. Howard, K. R. Covey, B. Hutchinson, and K. G. Stassun, "Apogee net: Improving the derived spectral parameters for young stars through deep learning," *The Astronomical Journal*, vol. 159, p. 182, apr 2020. 2

[13] C. Careaga, B. Hutchinson, N. O. Hodas, and L. Phillips, "Metric-based few-shot learning for video action recognition," *CoRR*, vol. abs/1909.09602, 2019. 2

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. 2

[15] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021. 2

[16] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter,

J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019. 2

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015. 2, 27

[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014. 2, 6

[19] M. Raghu, C. Zhang, J. M. Kleinberg, and S. Bengio, "Transfusion: Understanding transfer learning with applications to medical imaging," *CoRR*, vol. abs/1902.07208, 2019. 2, 9

[20] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Similarity-Based Pattern Recognition* (A. Feragen, M. Pelillo, and M. Loog, eds.), (Cham), pp. 84–92, Springer International Publishing, 2015. 3, 8

[21] Z. Nado, S. Padhy, D. Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek, "Evaluating prediction-time batch normalization for robustness under covariate shift," *arXiv preprint arXiv:2006.10963*, 2020. 3, 7, 10

[22] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," *arXiv preprint arXiv:2006.16971*, 2020. 3, 7, 10

[23] M. Chaudhary, P. Vianna, M. Eickenberg, A. Tang, G. Cloutier, G. Wolf, and E. Belilovsky, "Investigating prediction-time batch normalization under label shift," 3

[24] M. Yazdanpanah, A. A. Rahman, M. Chaudhary, C. Desrosiers, M. Havaei, E. Belilovsky, and S. E. Kahou, "Revisiting learnable affines for batch norm in few-shot transfer learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9109–9118, June 2022. 4

[25] A. M. Sarfi, Z. Karimpour, M. Chaudhary, N. M. Khalid, M. Ravanelli, S. Mudur, and E. Belilovsky, "Simulated annealing in early layers leads to better generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20205–20214, June 2023. 4

[26] S. Gauthier, B. Thérien, L. Alsène-Racicot, M. Chaudhary, I. Rish, E. Belilovsky, M. Eickenberg, and G. Wolf, "Parametric scattering networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5749–5758, June 2022. 5

[27] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *CoRR*, vol. abs/1802.03601, 2018. 6, 28

[28] X. Liu, C. Yoo, F. Xing, H. Oh, G. E. Fakhri, J.-W. Kang, and J. Woo, "Deep unsupervised domain adaptation: A review of recent advances and perspectives," 2022. 6

[29] Y. Wald, A. Feder, D. Greenfeld, and U. Shalit, "On calibration and out-of-domain generalization," *CoRR*, vol. abs/2102.10395, 2021. 6

[30] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1180–1189, PMLR, 07–09 Jul 2015. 6

[31] J. Liang, R. He, and T. Tan, "A comprehensive survey on test-time adaptation under distribution shifts," 2023. 6, 7

[32] R. Xia, J. Yu, F. Xu, and S. Wang, "Instance-based domain adaptation in nlp via in-target-domain logistic approximation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, Jun. 2014. 7

[33] H. Wu, Y. Yan, G. Lin, M. Yang, M. K. Ng, and Q. Wu, "Iterative refinement for multi-source visual domain adaptation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2810–2823, 2022. 7

[34] W. M. Kouw, L. J. van der Maaten, J. H. Krijthe, and M. Loog, "Feature-level domain adaptation," *Journal of Machine Learning Research*, vol. 17, no. 171, pp. 1–32, 2016. 7

[35] J. Wen, N. Zheng, J. Yuan, Z. Gong, and C. Chen, "Bayesian uncertainty matching for unsupervised domain adaptation," *CoRR*, vol. abs/1906.09693, 2019. 7

[36] J. Liang, D. Hu, Y. Wang, R. He, and J. Feng, "Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer," *CoRR*, vol. abs/2012.07297, 2020. 7

[37] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," *CoRR*, vol. abs/2002.08546, 2020. 7

[38] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 295–305, June 2022. 7

[39] D. Wang, E. Shelhamer, S. Liu, B. A. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. 7, 13

[40] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. M. Wallach, H. D. III, and K. Crawford, "Datasheets for datasets," *CoRR*, vol. abs/1803.09010, 2018. 8

[41] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, (New York), Association for Computer Machinery – ACM, Mar. 2021. 8

[42] T. Deleu, T. Würfl, M. Samiei, J. P. Cohen, and Y. Bengio, "Torchmeta: A meta-learning library for pytorch," *CoRR*, vol. abs/1909.06576, 2019. 8

[43] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-learning in neural networks: A survey," *CoRR*, vol. abs/2004.05439, 2020. 8

[44] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," 2015. 8, 26

[45] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016. 8

[46] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *CoRR*, vol. abs/1703.05175, 2017. 8

[47] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 8

[48] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, PMLR, 06–11 Aug 2017. 8

[49] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *In International Conference on Learning Representations (ICLR)*, 2017. 8

[50] A. Ruderman, N. C. Rabinowitz, A. S. Morcos, and D. Zoran, "Learned deformation stability in convolutional neural networks," *CoRR*, vol. abs/1804.04438, 2018. 9

[51] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013. 9, 40, 46, 48

[52] S. Mallat, "Group invariant scattering," 2012. 9, 45, 46, 47

[53] E. Oyallon and S. Mallat, "Deep roto-translation scattering for object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 9

[54] J. Andén, V. Lostanlen, and S. Mallat, "Classification with joint time-frequency scattering," *CoRR*, vol. abs/1807.08869, 2018. 9

[55] C. Vahidi, H. Han, C. Wang, M. Lagrange, G. Fazekas, and V. Lostanlen, "Mesostructures: Beyond spectrogram loss in differentiable time-frequency analysis," 2023. 9

[56] V. Chudáček, J. Andén, S. Mallat, P. Abry, and M. Doret, "Scattering transform for intrapartum fetal heart rate variability fractal analysis: A case-control study," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 4, pp. 1100–1108, 2014. 9

[57] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. in NeurIPS 25*, 2012. 9

[58] E. Oyallon, E. Belilovsky, and S. Zagoruyko, "Scaling the scattering transform: Deep hybrid networks," *CoRR*, vol. abs/1703.08961, 2017. 9

[59] E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. B. Blaschko, and E. Belilovsky, "Scattering networks for hybrid representation learning," *CoRR*, vol. abs/1809.06367, 2018.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. 10

[61] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013. 10

[62] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019. 10, 15

[63] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, *et al.*, "Ad click prediction: a view from the trenches," in *KDD 2013: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013. 10

[64] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248, PMLR, 13–18 Jul 2020. 10

[65] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 6028–6039, PMLR, 13–18 Jul 2020. 10

[66] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020. 10, 11

[67] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022. 10, 13

[68] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456, JMLR.org, 2015. 11, 12

[69] Y. Wu and K. He, "Group normalization," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), vol. 11217 of *Lecture Notes in Computer Science*, pp. 3–19, Springer, 2018. 12

[70] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016. 12

[71] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016. 12

[72] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), p. 901, 2016. 12

[73] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 2488–2498, 2018. 12

[74] J. Frankle, D. J. Schwab, and A. S. Morcos, "Training batchnorm and only batchnorm: On the expressive power of random features in cnns," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. 12

[75] M. Yazdanpanah, A. A. Rahman, M. Chaudhary, C. Desrosiers, M. Havaei, E. Belilovsky, and S. E. Kahou, "Revisiting learnable affines for batch norm in

few-shot transfer learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 9099–9108, IEEE, 2022. 12

[76] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 3942–3951, AAAI Press, 2018. 12

[77] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, "Improved few-shot visual classification," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 14481–14490, Computer Vision Foundation / IEEE, 2020. 12

[78] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020. 12

[79] Z. Nado, S. Padhy, D. Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek, "Evaluating prediction-time batch normalization for robustness under covariate shift," *CoRR*, vol. abs/2006.10963, 2020. 12

[80] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," *arXiv preprint arXiv:1603.04779*, 2016. 12

[81] C. Burns and J. Steinhardt, "Limitations of post-hoc feature alignment for robustness," *CoRR*, vol. abs/2103.05898, 2021. 12, 14

[82] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International conference on machine learning*, pp. 9229–9248, PMLR, 2020. 13

[83] A. Khurana, S. Paul, P. Rai, S. Biswas, and G. Aggarwal, "SITA: single image test-time adaptation," *CoRR*, vol. abs/2112.02355, 2021. 13

[84] G. Lin, S. Li, and H. Lee, "Listen, adapt, better WER: source-free single-utterance test-time adaptation for automatic speech recognition," in *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022* (H. Ko and J. H. L. Hansen, eds.), pp. 2198–2202, ISCA, 2022. 13

[85] L. Yang, H. Jiang, Q. Song, and J. Guo, "A survey on long-tailed visual recognition," *International Journal of Computer Vision*, vol. 130, pp. 1837–1872, may 2022. 13

[86] L. Zhao, Y. Teng, and L. Wang, "Logit normalization for long-tail object detection," *CoRR*, vol. abs/2203.17020, 2022. 13

[87] Z. Zhong, J. Cui, S. Liu, and J. Jia, "Improving calibration for long-tailed recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 16489–16498, Computer Vision Foundation / IEEE, 2021. 13

[88] L. Cheng, C. Fang, D. Zhang, G. Li, and G. Huang, "Compound batch normalization for long-tailed image classification," *CoRR*, vol. abs/2212.01007, 2022. 13

[89] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014. 14, 19

[90] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to imagenet," in *International conference on machine learning*, pp. 583–593, PMLR, 2019. 14, 19

[91] E. Oyallon, "Building a regular decision boundary with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5106–5114, 2017. 14, 19

[92] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, OpenReview.net, 2017. 14

[93] M. Davari, S. Horoi, A. Natik, G. Lajoie, G. Wolf, and E. Belilovsky, "Reliability of CKA as a similarity measure in deep learning," *CoRR*, vol. abs/2210.16156, 2022. 14

[94] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009. 15

[95] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Computer Vision and Pattern Recognition*, 2009. 15

[96] P. Vianna, S. Gauthier, H. Li, S.-I. Calce, C. Larocque-Rigney, L. Patry-Beaudoin, P. Boustros, E. Aslan, T. Alamri, K.-N. Vu, J. Murphy-Lavaleé, J.-S. Billiard, E. Montagnon, E. Belilovsky, G. Wolf, A. Tang, and G. Cloutier, "Deep learning to assess hepatic steatosis severity using ultrasound b-mode images," 2022. Poster presented at: IEEE International Ultrasound Symposium; October 10-13, 2022; Venice, Italy. 16

[97] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, June 2016. 16, 30, 43

[98] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *CoRR*, vol. abs/1608.03983, 2016. 16

[99] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, 2016. 16

[100] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," *arXiv preprint arXiv:2102.02079*, 2021. 16

[101] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, pp. 1126–1135, PMLR, 2017. 26

[102] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016. 26

[103] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019. 26

[104] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019. 26

[105] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 2018. 26

[106] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *arXiv preprint arXiv:1904.04232*, 2019. 26, 28

[107] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, pp. 3630–3638, 2016. 26, 29

[108] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proceedings of the 31st International conference on neural information processing systems*, pp. 4080–4090, 2017. 26

[109] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018. 26

[110] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," *arXiv preprint arXiv:1803.00676*, 2018. 26

[111] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten, "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning," *arXiv preprint arXiv:1911.04623*, 2019. 26

[112] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pp. 491–507, Springer, 2020. 26

[113] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: a good embedding is all you need?," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 266–282, Springer, 2020. 26

[114] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," *Advances in Neural Information Processing Systems*, vol. 32, pp. 10276–10286, 2019. 26

[115] P. Rodríguez, I. Laradji, A. Drouin, and A. Lacoste, "Embedding propagation: Smoother manifold for few-shot classification," in *European Conference on Computer Vision*, pp. 121–138, Springer, 2020. 26

[116] Y. Wang, C. Xu, C. Liu, L. Zhang, and Y. Fu, "Instance credibility inference for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12836–12845, 2020. 26

[117] C. P. Phoo and B. Hariharan, "Self-training for few-shot transfer across extreme task differences," *arXiv preprint arXiv:2010.07734*, 2020. 27, 28, 29, 30, 31

[118] S. Santurkar, D. Tsipras, A. Ilyas, and A. Mądry, "How does batch normalization help optimization?," in *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488–2498, 2018. 27

[119] J. M. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, and K. Neymeyr, "Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization," in *Proceedings of machine learning research*, 2019. 27

[120] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," *ArXiv*, vol. abs/1809.00846, 2019. 27

[121] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, p. 7705–7716, 2018. 27

[122] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, 2018. 27, 29, 31

[123] Y. Du, X. Zhen, L. Shao, and C. G. M. Snoek, "Metanorm: Learning to normalize few-shot batches across domains," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. 27

[124] J. Frankle, D. J. Schwab, and A. S. Morcos, "Training batchnorm and only batchnorm: On the expressive power of random features in cnns," *arXiv preprint arXiv:2003.00152*, 2020. 27, 31

[125] B. N. Oreshkin, P. Rodriguez, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," *arXiv preprint arXiv:1805.10123*, 2018. 28

[126] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009. 29

[127] N. Bendre, H. T. Marín, and P. Najafirad, "Learning from few samples: A survey." arXiv:2007.15484, 2020. 39

[128] J. Lezama, Q. Qiu, P. Musé, and G. Sapiro, "Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8109–8118, 2018. 39

[129] B. Barz and J. Denzler, "Deep learning on small datasets without pre-training using cosine loss," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1371–1380, 2020. 39, 43

[130] R. Gens and P. M. Domingos, "Deep symmetry networks," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014. 39

[131] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. M. Smeulders, "Structured receptive fields in cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 39

[132] R.-J. Bruintjes, A. Lengyel, M. B. Rios, O. S. Kayhan, and J. van Gemert, "Vipriors 1: Visual inductive priors for data-efficient deep learning challenges." arXiv:2103.03768, 2021. 39

[133] O. S. Kayhan and J. C. v. Gemert, "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 39

[134] L. Brigato, B. Barz, L. Iocchi, and J. Denzler, "Tune it or don't use it: Benchmarking data-efficient image classification," in *2nd Visual Inductive Priors for Data-Efficient Deep Learning Workshop*, 2021. 39

[135] E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko, and E. Belilovsky, "Scattering networks for hybrid representation learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2208–2221, 2018. 39, 43

[136] F. Cotter and N. Kingsbury, "A learnable scatternet: Locally invariant convolutional layers," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 350–354, IEEE, 2019. 39

[137] M. Ulicny, V. A. Krylov, and R. Dahyot, "Harmonic networks with limited training samples," in *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5, 2019. 39

[138] R. Balestriero, R. Cosentino, H. Glotin, and R. Baraniuk, "Spline filters for end-to-end deep learning," in *International conference on machine learning*, pp. 364–373, PMLR, 2018. 40

[139] L. Seydoux, R. Balestriero, P. Poli, M. De Hoop, M. Campillo, and R. Baraniuk, "Clustering earthquake signals and background noises in continuous seismic data with unsupervised deep learning," *Nature communications*, vol. 11, no. 1, pp. 1–12, 2020. 40

[140] R. Cosentino and B. Aazhang, "Learnable group transform for time-series," in *International Conference on Machine Learning*, pp. 2164–2173, PMLR, 2020. 40

[141] R. Balestriero, H. Glotin, and R. G. Baraniuk, "Interpretable super-resolution via a learned time-series representation." arXiv:2006.07713, 2020. 40

[142] M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, J. Andén, E. Belilovsky, *et al.*, "Kymatio: Scattering transforms in python.," *J. Mach. Learn. Res.*, vol. 21, no. 60, pp. 1–6, 2020. 42

[143] I. Azuri and D. Weinshall, "Generative latent implicit conditional optimization when learning from small sample," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 8584–8591, 2021. 42

[144] L. Sifre and S. Mallat, "Rotation, scaling and deformation invariant scattering for texture discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1233–1240, 2013. 42

[145] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 43

[146] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, p. 1100612, 2019. 43

[147] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. 44

[148] M. Eickenberg, G. Exarchakis, M. Hirn, S. Mallat, and L. Thiry, "Solid harmonic wavelet scattering for predictions of molecule properties," *The Journal of chemical physics*, vol. 148, no. 24, p. 241732, 2018. 46

[149] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020. 46