KNOWLEDGE TRANSFER IN VISION TASKS WITH INCOMPLETE DATA

BY

ZHIZHONG LI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Associate Professor Derek Hoiem, Chair
Associate Professor Svetlana Lazebnik
Assistant Professor Alexander G. Schwing
Dr. Linjie Luo

ABSTRACT

In many machine learning applications, some assumptions are so prevalent as to be left unwritten: all necessary data are available throughout the training process, the training and test data are independent and identically distributed (i.i.d.), and the dataset sampling sufficiently represent the test data of the model's usage scenario. Transfer learning methods can help when some of these assumptions are broken in real life, but still often assume all-time availability of data that the old and new knowledge can be learned from. In practice, necessary data or aspects of them can become inaccessible due to incomplete knowledge of test scenarios, privacy or legal concerns, protection of business leverage, evolving goals, etc.. In this thesis, we address three transfer learning scenarios in neural networks that regularly occur in practice but differ from both standard i.i.d. assumptions and common transfer learning data availability assumptions.

First, when transferring knowledge from previous tasks but the data used for training them is no longer available, we propose a method to extend and fine-tune the neural network to incorporate new classifiers while retaining the performance of existing classifiers. Second, with unsupervised domain adaptation where the target domain annotations are unavailable, we propose a method to more effectively transfer models to the unsupervised target domain, but guiding it using a common auxiliary task whose ground truth can be obtained for free or is already annotated. Finally, we show that, when test data is not i.i.d. with training data, classifiers are prone to confident but wrong predictions. In practical scenarios where the test data distribution is unknown before deploying the model, we explore ideas in several research fields to reduce confident errors. We observe that calibrated ensembles are the most effective, followed by single models calibrated using temperature scaling.

To Earth, for its tolerance.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Professor Derek Hoiem, without whom I could not have accomplished what I have achieved. Derek is kind and supportive throughout my five years of Ph.D. study. He is a great critic of what valuable research is, and has always guided me towards right and better ways of doing research. And of course, Derek is great at advising and has contributed to most of the suggestions that turned out to be the key in my works. I doubt if I could ever find a Ph.D. advisor that makes my experience better if I were to re-live my life. Thank you!

Appreciation also goes to all faculties in the vision community at University of Illinois at Urbana-Champaign (UIUC). Thank you, Professor David Forsyth, for your pointed yet sharp observations and criticisms during paper discussions; thank you Professor Svetlana Lazebnik and Professor Alexander Schwing for repeated discussions of my research and various other topics; thank you Professor Sanmi Koyejo and David again for your help with managing the vision cluster; thank you, Professor Saurabh Gupta and Alex again, for mingling with us students. Special thanks go to my research collaborators: Linjie Luo, Sergey Tulyakov, Qieyun Dai, Arun Mallya, Hongxu Yin, and Pavlo Molchanov. I had a great time in all my internships and it was a great pleasure working with you!

My heartfelt thanks also go to all my friends here in the vision community. Saurabh Gupta, Kevin Shih, and Arun Mallya showed me the ropes of doing deep learning research and managing the vision cluster and the kind guidance in research life. Qieyun Dai talked with me about grad life and gaming. Daphne Tsatsoulis provided kindness and afternoon tea. Joseph Degol, Bryan Plummer, Liwei Wang, and Jiajun Lu shared their life hacks. Aditya Deshpande shared his enthusiasm for weird jokes. Jason Rock shared his David-ness which is always helpful in research. Tanmay Gupta and Raj Kataria made my grad life more lively, and reduced my snack stack (just kidding, it was Chuhang). Daniel McKee shared his joyfulness and helped a lot with the transition and management of the vision cluster. Unnat Jain shared many research suggestions and checked on my mental status. Anand Bhattad constantly poked me with new research ideas. Also thank you Theerasit Issaranon, Maryia Vasileva, Dominic Roberts, Min Jin Chong, Aiyu Cui, Jae Yong Lee, Jeffrey Zhang, Victor Gonzalez, and Mantas Mazeika for the time we spent together in the office and at leisure. Special thanks goes to Aiyu Cui for her kindness and listening to all my ramblings, and Shuai Tang for being an amazing roommate and all the discussions on research and life. I would also like to thank my friends and families back home; thank you for your support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 THESIS STATEMENT

Knowledge in humans seems to come naturally to us. Since we were babies, we act upon this world, and we get feedback. We learn knowledge piece after piece, year after year. For most humans, we can identify the category given an object, predict its attributes such as 3D shape, color, material, typical appearing environments, and how it acts and reacts to actions temporally, etc.. We can infer between correlated attributes, and apply these correlations on new types of objects or environments. We also know when our prediction is uncertain. These types of knowledge are not only inherently useful to us but also help us generalize by enabling logical inference.

The same things cannot be asserted for machine learning algorithms. For deep networks, for example, it is easier to predict categories and enumerated attributes, but the remainders require specially designed architectures or losses. It does not help that training frameworks often rely on assumptions that may not necessarily hold true in practice. For example, training and test data are assumed to be independent and identically distributed (i.i.d.), and the task for which the model predicts should stay the same, and training and test data are both easily accessible.

These assumptions can easily break down in real applications, due to incomplete knowledge of test scenarios, the difficulty of obtaining training data, privacy or legal concerns, protection of business leverage, changing or expanding tasks, and other challenges.

Knowledge transfer methods enable machine learning models to obtain some of the aforementioned, more complex types of human knowledge, and have made great progress in mitigating some aspects of the broken assumptions.

In this thesis, we highlight three special transfer learning scenarios with common, practical data constraints that are often overlooked in the literature, and investigate how to bridge these additional gaps between existing methods and practice.

1. Learning new knowledge without previous data may lead to catastrophic forgetting of existing knowledge. When we perform transfer or continual learning, what if the existing model is provided by an external entity that does not take continual learning into account, and is unwilling to share any data?

2. Can we use an extra task with cheaply obtainable ground truth to help the unsupervised domain adaptation of a main task?

	Available resources	Desired model	Proposed solution
Chapter 3	Existing old task model w/o training data	Extended model w/ good performance on original and new tasks	Use new or synthetic images & their old task predictions as substitute old dataset
Chapter 4	Unsupervised domain adaptation	Model with improved main task performance on target domain	Train with auxiliary task with cheap labels on both domains
Chapter 5	Training data w/ less diversity and scenarios than test data in the wild	Proper confidence predictions for all inputs	Calibrated ensemble

Table 1.1: Description of issues we address and proposed methods.

3. In real-life applications where it is impossible to anticipate all unknown test scenarios, how can we proceed in improving generalization for such differently distributed domains completely unknown during training?

A summary of issues we identify and ideas proposed are described in Table 1.1.

1.2 CHALLENGES AND CONTRIBUTIONS

1.2.1 Learning without Forgetting and DeepInverion

Suppose a company or a customer purchases a vision system, and they want to expand the capabilities of the system. They typically will not have access to the system's training data, nor will they be able to dictate how the vision system has been trained. One solution is continual learning, which gradually learns new knowledge from a stream of tasks while trying to reduce catastrophic forgetting of knowledge already learned. From an academic standpoint, it may be feasible to retain a subset of past task data to help retaining knowledge, but in practice, such as the company scenario above, it is not always true.

Our proposed Learning without Forgetting method uses only new task data to train the network while preserving the original capabilities, by substituting the original dataset. For each sample in the training set of the new task, we feed it into the original model to get its soft predictions. We then train using the input-prediction pair as substitute data for the old task while also learning the new task.

When tested on transferring from large datasets such as ImageNet to smaller ones such as CUB, our method performs favorably compared to commonly used feature extraction and fine-tuning adaption techniques and performs similarly to multitask learning that uses

original task data we assume unavailable. A more surprising observation is that Learning without Forgetting may be able to replace fine-tuning with similar old and new task datasets for improved new task performance.

One limitation of Learning without Forgetting is the poor performance when the old task dataset is drastically different from that of the new task, the new images and their soft predictions do not serve as good proxies for the old data. We follow up with DeepInversion, which synthesizes images that distribute similarly to the original training data. We regularize the synthesis by forcing the batch statistics of synthetic images to be close to the batch normalization statistics of the provided model.

The resulting synthesized images from networks trained on the CIFAR-10 and ImageNet datasets demonstrate high fidelity and degree of realism, and help enable a breed of data-free applications that do not require any real data for existing knowledge. In particular, we demonstrate our method on data-free class-incremental learning, approaching the performance of oracle methods.

We discuss these methods in Chapter 3, based on our papers *Learning without Forgetting* [1] and *Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion* [2].

1.2.2 Domain Adaptation with an Auxiliary Task

Usually, with unsupervised domain adaptation, we cannot access ground truth for the target domain. For example, some tasks, such as surface normals or single-view depth estimation, require per-pixel ground truth that is difficult to obtain on real images but easy to obtain on synthetic. Trying to estimate surface normals from depth can either produce extremely noisy ground truth, or if prior knowledge is applied, produce labels unfaithful to the original images. Synthetic surface normals come at no cost, but models learned on synthetic images often do not generalize well to real images due to the domain shift.

Supervised domain adaptation can use semantic information in ground truth to assert constraints in the adaptation, e.g. object of the same class should have similar features in both domains. Can we replicate that for unsupervised adaptation?

Our key idea to improve domain adaptation is to introduce a separate anchor task (such as landmarks) whose annotations can be obtained at no cost or are already available on both synthetic and real datasets. To further leverage the implicit relationship between the anchor and main tasks, we apply our HeadFreeze technique that learns the cross-task guidance on the source domain with the final network layers, and then use it on the target domain.

We evaluate our methods on surface normal estimation on two pairs of datasets (indoor

scenes and faces) with two kinds of anchor tasks (semantic segmentation and facial landmarks). We show that blindly applying domain adaptation or training the auxiliary task on only one domain may hurt performance, while using anchor tasks on both domains is better behaved. Our HeadFreeze technique outperforms competing approaches, reaching performance in facial images on par with a recently popular surface normal estimation method using shape from shading domain knowledge.

We discuss this method in Chapter 4, based on our paper *Task-Assisted Domain Adaptation with Anchor Tasks* [3].

1.2.3 Improving Generalization on Data outside the Known Distribution

In real-life applications, test images can differ from training data in both expected and unexpected ways. Domain adaptation often assumes that the test distribution is obtainable, by collecting a target domain dataset. However, gathering a set of "unexpected" target domain samples is by definition unviable.

Intuitively, unfamiliarity due to the input distribution difference should lead to a lack of confidence. In reality, current algorithms often make highly confident yet wrong predictions when faced with relevant but unfamiliar examples. A classifier we trained to recognize gender is 12 times more likely to be wrong with a 99% confident prediction if presented with a subject from a different age group than those seen during training. To draw attention to this issue, we provide a study to compare and evaluate several methods to improve confidence estimates for unfamiliar and familiar samples.

For our study, we propose a testing methodology of splitting unfamiliar and familiar samples by attribute (age, breed, subcategory) or sampling (similar datasets collected by different people at different times). We evaluate methods including confidence calibration, ensembles, distillation, and a Bayesian model and use several metrics to analyze label, likelihood, and calibration error. While all methods reduce over-confident errors, the ensemble of calibrated models performs best overall, and T-scaling performs best among the approaches with faster inference.

We present our study in Chapter 5, based on our paper *Improving Confidence Estimates for Unfamiliar Examples* [4].

CHAPTER 2: RELATED WORK

Scholars have drawn attention to scenarios where the i.i.d. train and test set assumptions are broken due to practical reasons. Sometimes the dataset is collected in a laboratory setting, which makes them distributed differently from data in the wild and limits a model's performance when deployed [5]. Inherent lack of knowledge about test cases during data collection may also lead to an incomplete dataset, inferior performance [6, 7], and even security risks [8]. The price of collecting and labeling the data can be prohibitive for medium to large datasets, so ground truth may be limited in availability [9]. Further, part of the dataset may become unavailable, perhaps because past data cannot be retained for legal, contractual, or cost-saving reasons [10].

Some works in the transfer learning literature try to address these mismatches between the main training data environment and the application data environment, a.k.a. source and target domains. Either the input data, the task, or the joint input-label distribution can be different between the two domains, just like scenarios in the wild. Different data constraint situations divide transfer learning methods into different sub-fields [9, 11]. In this chapter, we discuss two major groups in the literature: methods where the task is different (inductive transfer learning), and where the task is the same but data distribution is different (domain adaptation). We also compare related topics such as knowledge distillation and improving generalization.

2.1 TASK CHANGES

When tasks from domains are different, we can use *inductive transfer learning* to learn multiple tasks sequentially or at once. The knowledge from each task helps to learn one another and improves generalization. This branch of knowledge transfer is closely related to multi-task learning and incremental learning (a.k.a. continual learning or lifelong learning).

One common strategy for a data-scarce task is to pre-train a neural model on a data-rich task such as ImageNet classification, and then fine-tune the model on the data-scarce task [12, 13, 14]. Using appropriate hyper-parameters for training, the resulting model often outperforms feature extraction [12, 15] or learning from a randomly initialized network [13, 14]. A small learning rate is often used, and sometimes part of the network is frozen to prevent overfitting. However, the model tends to lose the power to perform the pre-trained task due to catastrophic forgetting [16], a phenomenon that neural networks forget past knowledge when learning new ones.

Feature extraction [17, 18, 19] uses a pre-trained deep CNN to compute features for an image. The extracted features are the activations of one layer (usually the last hidden layer) or multiple layers given the image. Classifiers trained on these features can achieve competitive results and can outperform human-engineered features [17]. Further studies [15] show how hyper-parameters, e.g., original network structure, should be selected for better performance. Taskonomy [19] studies the affinity between tasks by pre-training a backbone network on one task, and training a predictor on its extracted features, and finally gauging the relative performance. Feature extraction does not modify the original network and allows new tasks to benefit from complex features learned from previous tasks. However, these features are not specialized for the new task and can often be improved by fine-tuning.

Adding new nodes to the network, e.g. to each layer, is a way to preserve the original network parameters while learning new discriminative features. Parameters for the original network are untouched, and newly added nodes are fully connected to the layer beneath them. For example, Terekhov et al. [20] propose deep block-modular neural networks for expanding fully-connected networks, and Rusu et al. [21] propose progressive neural networks for reinforcement learning. Growing a Brain [22] increases layer sizes or network depth to improve fine-tuning performance on the new task, using a normalization in the activation scale of the new nodes. These methods have the downside of substantially expanding the number of parameters in the network, and can underperform [20] both fine-tuning and feature extraction if insufficient training data is available to learn the new parameters, since they add a substantial number of parameters to be trained from scratch.

Meta-learning (e.g. MAML [23]) aims to learn an algorithm that helps the learning of new tasks, usually done by splitting a dataset into multiple tasks and learning a model that helps transfer between a random subset of tasks. Meta-learning can also be applied to inductive transfer learning if enough number of source tasks are available.

Multi-task learning (e.g., [24]) is very closely related. It is only different in its aim at improving all tasks simultaneously, rather than just the target task, but most methods require the data for all tasks to be available. It is very popular for learning related tasks with neural networks [25, 26, 27, 28, 29], either to improve performance or to save computation. Each task provides extra training data for the parameters that are shared or constrained, serving as a form of regularization for each other [30]. Notably, UberNet [26] accommodates a large number of tasks by producing outputs at each convolution level and each image spatial pyramid level and fusing them across multiple resolutions. PADNet [27] first makes predictions for multiple tasks and then uses them to predict a refined prediction. As with inductive transfer, these methods assume that all task data is available throughout training. Cross-stitch Network [31] introduces a module that takes two same-structured inputs, applies

two same-structured network blocks, and averages their activations with two pairs of scalar weights to obtain two same-structured outputs.

Weakly-supervised learning [32] and *self-supervised learning* [33] are special instances of inductive transfer learning. The former uses a related, but easier-to-annotate "pretext" task as extra training data, such as using object detection to help instance segmentation [34] and using coarser categories to help finer categories [35]. The latter constructs a pretext task using just the input data alone, such as colorizing an image that has been converted to grayscale [36, 37], predicting the spatial positions of nearby patches [38], in-painting [39], rotation and clustering [40], and simply predicting the image index [41, 42]. The boundary between self-supervised learning and unsupervised learning is blurry, but self-supervised learning usually creates annotations from the input and uses supervised learning frameworks to learn a feature representation, rather than only using e.g. clustering or PCA.

Continual learning (a.k.a. incremental learning or lifelong learning, see [43], Chapter 4) is closely related to transfer learning. These methods assume training tasks or data can only be obtained incrementally, and the algorithm cannot see future data and has limited access to previous tasks' data. They have to sequentially learn new knowledge and mitigate catastrophic forgetting. Van de Ven et al. [44] identify three major categories in continual learning. Task-incremental learning learns tasks one by one and the identity of the task to be performed on each input is considered known. Domain-incremental learning is when the task stays the same but the data distribution is changing, and the exact domain that a sample comes from is unknown. Class-incremental learning has to solve both the task and which task the sample comes from – e.g. accumulating an increasing number of categories to classify and having to distinguish between old and new categories.

Some continual learning methods such as LwF [1], LwF.MC [45], and DeepInversion [2] achieve data-free continual learning, where no past data is required to be available. Other methods require information that can only be obtained from the original dataset. Some use a subset of past data as exemplars (iCaRL [45]), or to estimate the importance of each network parameter (in the form of Fisher matrix in EWC [46], contribution to loss change in SI [47], and posterior of network weights in VCL [48]). Some use past data to train their representation (encoder [49], GAN [50, 51]) to help regenerate the training data. Some methods rely on network modifications, e.g. PackNet [52] uses past data to prune networks to make way for new knowledge, but a follow-up work Piggyback [53] achieves data-free task-incremental learning by learning different pruning masks for different tasks with only new task data.

Never-ending learning [54] also integrate knowledge over time. It focuses on building diverse knowledge and experience (e.g., by reading the web every day). Ruvolo et al. [55]

describe a method to efficiently add new tasks to a multitask system, co-training all tasks while using only new task data. The method assumes that weights for all classifiers and regression models can be linearly decomposed into a set of bases.

2.2 SAME TASK, DISTRIBUTION CHANGES

On the other hand, when the tasks are the same in both domains, but the joint input-label distribution is different (e.g. when one is in a lab environment, and one is in the wild), models' predictions become unreliable [6, 7]. Domain adaptation methods [56] come into play. We mainly discuss scenarios where the input space stays the same, whereas in other transductive transfer learning scenarios (see e.g. [9]) the input space can be different as well. We also assume covariate shift, i.e. the relationship between input and output ($P(y|x)$) stays the same between different domains. Domain adaptation (especially in deep learning) is a vibrant and active field. We refer our readers to the many surveys available [57, 58, 11] for a more complete picture.

In terms of data availability, supervised domain adaptation assumes the availability of some labeled samples in the target domain (e.g. [59]), but the annotations may be expensive to obtain. Unsupervised domain adaptation methods allow for all target sample labels to be missing [60, 61, 62, 63, 64, 65], but these potentially miss the opportunity to establish reliable semantic correspondence between domains using the ground truth. There are also methods that adapt without having the source domain available [10].

In terms of methods, most try to reduce the distribution difference between the source and target. We can model the distributions and explicitly minimize the difference [60], or use an adversarial loss to encourage the source and target representations to become indistinguishable to an adversarial network [65, 59]. The distribution matching usually happen on the feature space [60, 65], but can be done on the output space [61, 66] or the input images [64, 63] as well. Notably, the deep adaption network by Long et al. [60] matches the RKHS embedding of the deep representation of both source and target tasks to reduce domain bias. Tzeng et al. [59] encourage the shared deep representation to be indistinguishable to a domain classifier. Tsai et al. [66] applies an adversarial loss to the outputs of images from source and target domain images to make the predictions distribute similarly. CyCADA [63] uses a CycleGAN [67] to stylize synthetic images into real-looking images on which the main model is trained.

Some other methods deal with covariate shift by *reweighting samples*, e.g. using a domain classifier [68] to focus on those more similar to the target domain or increase the weights of misclassified target samples [69] similar to AdaBoost. These methods may assume some

source samples are close enough to the target distribution, but often in practice, the two distributions may overlap very little, making reweighting by itself insufficient. Some methods use a strategy similar to fine-tuning or multi-task learning, and adapt between domains by *reusing network weights*. Long et al. [70] assume that the modeled function is only different by an output transformation and add source domain-specific residual layers while reusing the backbone.

Domain adaptation requires data from the target domain, but in practice, the exact test environment can be hard to foresee [8]. *Domain generalization* [71, 72, 73] further aims to build models that generalize well on a previously unspecified domain, whose distribution can be different from all training domains. These models generally build a domain-invariant feature space [71] or a domain-invariant model [73], or factor models into domain-invariant and domain-specific parts [72]. These models all require multiple training domains to learn invariant representations, and often assume these source domains demonstrate specific aspects of the data that can potentially be different in the target domain.

When *the test scenario is completely unknown*, there are several fall-back lines of research to mitigate performance loss. One is to improve generalization, e.g. data augmentation [74, 75] or jittering [76], dropout [77], batch normalization [78], and weight decay. Hoffer et al. [79] propose better hyperparameter selection strategies for better generalization. Bagging [80], ensembles, and other model averaging techniques are also used prior to deep learning.

Another fall-back is to gauge epistemic uncertainty (uncertainty due to the lack of knowledge) and make best-possible predictions. For example, Bayesian methods [81, 82, 83] evaluate multiple models that all agree on the training data, and their disagreement on unfamiliar data can indicate uncertainty. Bayesian methods then produce an averaged output that reflects the best-possible prediction based on the uncertainty. However, these methods are usually very computationally intensive [6], limiting their practical application. Approximations for Bayesian methods can mitigate their low speed. Gal and Ghahramani [84] propose MC-dropout (using dropout for a Monte-Carlo sample of likelihood estimates) as a discrete approximation. Follow-up work [85] estimates aleatoric uncertainties as well. Multi-head networks [86, 87] approximate ensembles.

As another fail-safe, one can estimate whether the system is likely to fail and output a signal requesting external intervention [88, 89, 90, 91, 92], for example by looking at how close samples are to decision boundaries or estimating whether a test sample comes from the same distribution as training [93, 94, 95, 96, 97]. Typically, the motivation of these methods is to avoid making any prediction on suspect samples, even when users reasonably expect a prediction.

2.3 KNOWLEDGE TRANSFER BETWEEN NETWORKS

Our work also relates to *methods that transfer knowledge* between networks, for the same task and the same or similar data distribution. This idea dates back to when Breiman and Shang learned a single decision tree to approximate the outputs of multiple decision trees [98]. Similar ideas are explored in neural networks by Bucilua et al. [99], Ba and Caruana [100], and Hinton et al. [101]. Hinton et al. formulate the problem as "knowledge distillation," where a compact student mimics the output of expert teacher models [101]. The smaller network is trained using a modified cross-entropy loss that encourages both large and small probability masses of the original and new network to be similar. Its effectiveness and efficiency have been thereafter improved by novel techniques such as intermediate layer guidance [102], fast initialization [103], attention maps [104], auxiliary adversarial networks [105], relational losses [106], variational information losses [107], etc.. These methods enable teaching students with goals such as quantization [108, 109], compact neural network architecture design [102], semantic segmentation [110], self-distillation [111], and unsupervised or semi-supervised learning [112, 113, 114].

All methods above rely on images from the original dataset or one with a similar distribution. More recent research has explored data-free knowledge distillation. Lopes et al. [115] synthesized inputs based on pre-stored auxiliary layer-wise statistics of the teacher network. Chen et al. [116] trained a new generator network for image generation while treating the teacher network as a fixed discriminator. These methods shed light on generating synthetic versions of datasets with tiny images, such as MNIST and CIFAR.

CHAPTER 3: LEARNING WITHOUT FORGETTING AND DEEPINVERSION

3.1 INTRODUCTION AND RELATED WORK

Many practical vision applications require learning new visual capabilities while maintaining performance on existing ones. Ideally, the new tasks could be learned while sharing parameters from old ones, without suffering from catastrophic forgetting [16, 117] (degrading performance on old tasks) or having access to the old training data.

In our setting, a CNN has a set of shared parameters θ_s (e.g., five convolutional layers and two fully connected layers for AlexNet [74] architecture), task-specific parameters for previously learned tasks θ_o (e.g., the output layer for ImageNet [118] classification and corresponding weights), and randomly initialized task-specific parameters for new tasks θ_n (e.g., scene classifiers). It is useful to think of θ_o and θ_n as classifiers that operate on features parameterized by θ_s.

Currently, there are three common approaches (Figs. 3.1, 3.2(b-d)) to learning θ_n while benefiting from previously learned θ_s, and they differ mostly on which parameters are unchanged, and all have certain drawbacks:

Feature extraction (e.g., [17]): θ_s and θ_o are unchanged, while θ_n is learned anew.

Fine-tuning (e.g., [12]): θ_s and θ_n are optimized for the new task, while θ_o is fixed. Potentially, the original network could be duplicated and fine-tuned for each new task to create a set of specialized networks.

Joint training (e.g., [24]): All parameters θ_s, θ_o, θ_n are jointly optimized, for example by interleaving samples from each task.

Besides these commonly used approaches, methods [119, 120] have emerged that can continually add new prediction tasks by adapting shared parameters *without access to training data* for previously learned tasks.

	Fine Tuning	Duplicating and Fine Tuning	Feature Extraction	Joint Training	Learning without Forgetting
new task performance	good	good	X medium	best	best
original task performance	X bad	good	good	good	good
training efficiency	fast	fast	fast	X slow	fast
testing efficiency	fast	X slow	fast	fast	fast
storage requirement	medium	X large	medium	X large	medium
requires previous task data	no	no	no	X yes	no

Figure 3.1: This figure shows relative advantages of our method compared to commonly used methods.

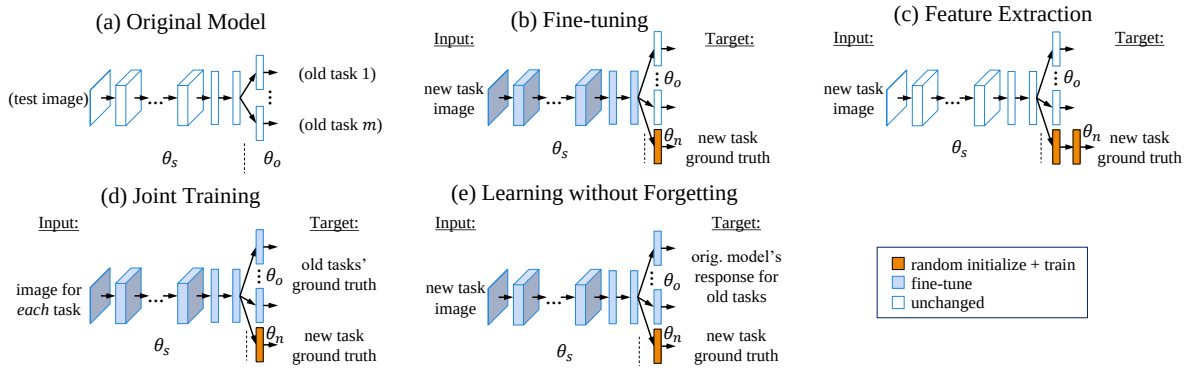


Figure 3.2: Illustration for our method (e) and methods we compare to (b-d). Images and labels used in training are shown. Data for different tasks are used in alternation in joint training.

A-LTM [119], developed independently, is nearly identical in method but has very different experiments and conclusions. The main differences of method are in the weight decay regularization used for training and the warm-up step that we use prior to full fine-tuning.

However, we use large datasets to train our initial network (e.g., ImageNet) and then extend to new tasks from smaller datasets (e.g., PASCAL VOC), while A-LTM uses small datasets for the old task and large datasets for the new task. The experiments in A-LTM [119] find much larger loss due to fine-tuning than we do, and the paper concludes that maintaining the data from the original task is necessary to maintain performance. Our experiments, in contrast, show that we can maintain good performance for the old task while performing as well or sometimes better than fine-tuning for the new task, without access to original task data. We believe the main difference is the choice of old-task new-task pairs and that we observe less of a drop in old-task performance from fine-tuning due to the choice (and in part to the warm-up step). We believe that our experiments, which start from a well-trained network and add tasks with less training data available, are better motivated from a practical perspective.

Less Forgetting Learning [120] is also a similar method, which preserves the old task performance by discouraging the shared representation to change (Fig. 3.2(e)). This method argues that the task-specific decision boundaries should not change, and the shared representation should not change. Therefore, LFL adds a L_2 loss that discourages the *output after* θ_s from changing for new task images, while θ_o remains as is. In comparison, our LwF method adds a loss that discourages the *old task output* to change for new task images, and jointly optimizes both the shared representation and all the final layers. We empirically show that our method outperforms Less Forgetting Learning on the new tasks.

Other work on incremental learning. Besides these two, to the best of our knowledge, only our LwF [1] method, LwF.MC [45] (a class-incremental variant), and Piggyback [53] achieve data-free continual learning. Other methods require information that can only be obtained from the original dataset, e.g., a subset of data (iCaRL [45]), parameter importance estimations (in the form of Fisher matrix in EWC [46], contribution to loss change in SI [47], posterior of network weights in VCL [48]), or training data representation (encoder [49], GAN [50, 51]). Some methods rely on network modifications e.g., Packnet [52] and Piggyback [53]; the latter is data-free but requires modifying network structures, and cannot evaluate all tasks at once. In comparison, our methods do not need network modifications or the original (meta-)data, and batch normalization statistics used in DeepInversion are inherent to neural networks.

3.2 LEARNING WITHOUT FORGETTING

3.2.1 Method

We propose our method **Learning without Forgetting** (LwF) [1]. Using only examples for the new task, we optimize both for high accuracy for the new task and for preservation of responses on the existing tasks from the original network. Our method is similar to joint training, except that our method does not need the old task's images and labels. Our network structure illustrated in Fig. 3.2(e).

Given a CNN with shared parameters θ_s and task-specific parameters θ_o (Fig. 3.2(a)), first we record responses $\mathbf{y}_o$ on each new task image from the original network for outputs on the old tasks (defined by θ_s and θ_o). Our experiments involve classification, so the responses are the set of label probabilities for each training image. Nodes for each new class are added to the output layer with randomly initialized weights θ_n, which takes up typically a very small percent of the total number of parameters.

Next, we train the network to minimize loss for all tasks and regularization $\mathcal{R}$ using stochastic gradient descent. The regularization $\mathcal{R}$ corresponds to a simple weight decay of 0.0005. For new tasks, the loss encourages predictions $\hat{\mathbf{y}}_n$ to be consistent with the ground truth $\mathbf{y}_n$, using a commonly used loss form (e.g. cross-entropy for multi-class classification, or binary cross-entropy for multi-label classification).

For each original task, we want the output probabilities for each image to be close to the recorded output from the original network. We use the knowledge distillation loss, which was found by Hinton et al. [101] to work well for encouraging the outputs of one network to approximate the outputs of another. This is a modified cross-entropy loss that increases the

weight for smaller probabilities:

$$\mathcal{L}_{old}(\mathbf{y}_o, \hat{\mathbf{y}}_o) = -H(\mathbf{y}'_o, \hat{\mathbf{y}}'_o) = -\sum_{i=1}^{l} y_o'^{(i)} \log \hat{y}_o'^{(i)} \tag{3.1}$$

where l is the number of labels and $y_o'^{(i)}$, $\hat{y}_o'^{(i)}$ are the modified versions of recorded and current probabilities $y_o^{(i)}$, $\hat{y}_o^{(i)}$ with higher temperatures $T = 2$ [101].

3.2.2 Experiments

Our experiments are designed to evaluate whether Learning without Forgetting (LwF) is an effective method to learn a new task while preserving performance on old tasks. We compare to common approaches of *feature extraction*, *fine-tuning*, and *fine-tuning FC*, and also *Less Forgetting Learning* (LFL) [120]. These methods leverage an existing network for a new task without requiring training data for the original tasks. Feature extraction maintains the exact performance on the original task. We also compare to *joint training* (sometimes called multitask learning) as an upper-bound on possible old task performance, since it uses old task data we assume unavailable.

We experiment on a variety of image classification problems with varying degrees of inter-task similarity. For the original ("old") task, we consider the ILSVRC 2012 subset of *ImageNet* [118] and the *Places365-standard* [121] dataset, both large in sample number since we assume we start from a well-trained network, which implies a large-scale dataset. For the new tasks, we consider PASCAL *VOC 2012 image classification* [122] ("VOC"), *Caltech-UCSD Birds-200-2011 fine-grained classification* [123] ("CUB"), and *MIT indoor scene classification* [124] ("Scenes"). In one experiment, we use MNIST [125] as the new task expecting our method to underperform, since the hand-written characters are completely unrelated to ImageNet classes.

We mainly use the AlexNet [74] network structure because it is fast to train and well-studied by the community [14, 15, 12]. We also verify that similar results hold using 16-layer VGGnet [126] on a smaller set of experiments. For both network structures, the final layer (`fc8`) is treated as task-specific, and the rest are shared (θ_s) unless otherwise specified. The original networks pre-trained on ImageNet and Places365-standard are obtained from public online sources. Due to the randomness within CNN training, we run our experiments three times, and report the mean performance.

(a) Using AlexNet structure (validation performance for ImageNet/Places365/VOC)

	ImageNet→VOC		ImageNet→CUB		ImageNet→Scenes		ImageNet→MNIST		Places365→VOC		Places365→CUB		Places365→Scenes		Places365→MNIST	
	old	new	old	new	old	new	old	new	old	new	old	new	old	new	old	new
LwF (ours)	56.2	76.1	54.7	57.7	55.9	64.5	49.8	99.3	50.6	70.2	47.9	34.8	50.9	75.2	38.3	99.2
Fine-tuning	-0.9	-0.3	-3.8	-0.7	-2.0	-0.8	-2.8	0.0	-2.2	0.1	-4.6	1.0	-2.1	-1.7	-0.9	0.1
LFL	0.0	-0.4	-1.9	-2.6	-0.3	-0.9	-2.9	-0.6	0.2	-0.7	0.7	-1.7	-0.2	-0.5	-0.4	-0.1
Fine-tune fc	0.5	-0.7	0.2	-3.9	0.6	-2.1	7.0	-0.2	0.5	-1.3	1.8	-4.9	0.3	-1.1	13.0	-0.2
Feat. Extraction	0.8	-0.5	2.3	-5.2	1.2	-3.3	7.3	-0.8	1.1	-1.4	3.8	-12.3	0.8	-1.7	13.3	-1.1
Joint Training	0.7	-0.2	0.6	-1.1	0.5	-0.6	7.2	-0.0	0.7	-0.0	2.3	1.5	0.3	-0.3	13.4	-0.1

(b) Test set performance

	Places365→VOC	
	old	new
LwF (ours)	50.6	73.7
Fine-tuning	-2.1	0.1
Feat. Extraction	1.3	-2.3
Joint Training	0.9	-0.1

(c) Using VGG structure

	ImageNet→CUB		ImageNet→Scenes	
	old	new	old	new
LwF (ours)	60.6	72.5	66.8	74.9
Fine-tuning	-9.9	0.6	-4.1	-0.3
LFL	0.3	-2.8	-0.0	-2.1
Fine-tune fc	3.2	-6.7	1.4	-2.4
Feat. Extraction	8.2	-8.6	1.9	-5.1
Joint Training	8.0	2.5	4.1	1.5

Table 3.1: Performance for the single new task scenario. For all tables, the **difference** of methods' performance with LwF (our method) is reported to facilitate comparison. Mean average precision (mAP) is reported for VOC and accuracy for all others. On the new task, LwF outperforms baselines in most scenarios, and performs comparably with joint training, which uses old task training data we consider unavailable for the other methods. On the old task, our method greatly outperforms fine-tuning and achieves slightly worse performance than joint training. An exception is the ImageNet-MNIST task where LwF does not perform well on the old task.

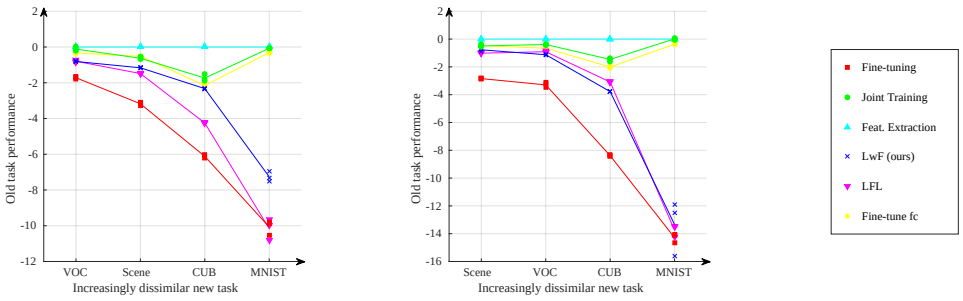
3.2.3 Results

We compare the results of learning one new task among different task pairs and different methods. Table 3.0(a), 3.0(b) shows the performance of our method, and the relative performance of other methods compared to it using AlexNet.

On the new task, our method consistently outperforms LFL, fine-tuning FC, and feature extraction, while outperforming fine-tuning on most task pairs except Places365→CUB, Places365→VOC, Places365→MNIST (similar performance), and ImageNet→MNIST (worse performance). The gain over fine-tuning was unexpected and indicates that preserving outputs on the old task is an effective regularizer. This finding motivates replacing fine-tuning with LwF as the standard approach for adapting a network to a new task.

On the old task, our method performs better than fine-tuning but often underperforms feature extraction, fine-tuning FC, and occasionally LFL. By changing shared parameters θ_s, fine-tuning significantly degrades performance on the task for which the original network was trained. By jointly adapting θ_s and θ_o to generate similar outputs to the original network on an old task similar to the new one, the performance loss is greatly reduced.

Our method usually performs similarly to joint training with AlexNet. Our method tends to slightly outperform joint training on the new task but underperform on the old task, which



(a) ImageNet as the old task (b) Places365 as the old task

Figure 3.3: Influence of new-old task similarity on old task performance preservation, related to the original old task performance. As the tasks becomes further irrelevant, the old task preservation drops.

we attribute to a different distribution in the two task datasets. Overall, the methods perform similarly (except on the extreme *→MNIST cases), a positive result since our method does not require access to the old task training data and is faster to train.

Dissimilar new tasks degrade old task performance more, visualized in Fig. 3.3. For example, CUB is very dissimilar task from Places365 [15], and adapting the network to CUB leads to a Places365 accuracy loss of 8.4% (3.8% + 4.6%) for fine-tuning, 3.8% for LwF, and 1.5% (3.8% − 2.3%) for joint training. In these cases, learning the new task causes considerable drift in the shared parameters, which cannot fully be accounted for by LwF because the distribution of CUB and Places365 images is very different. Even joint training leads to more accuracy loss on the old task because it cannot find a set of shared parameters that works well for both tasks. Our method does not outperform fine-tuning for Places365→CUB and, as expected, *→MNIST on the new task, since the hand-written characters provide poor indirect supervision for the old task.

Similar observations hold for both VGG and AlexNet structures, except that joint training outperforms consistently for VGG, and LwF performs worse than before on the old task. (Table 3.0(c)) This indicates that these results are likely to hold for other network structures as well, though joint training may have a larger benefit on networks with more representational power. Among these results, LFL diverges using stochastic gradient descent, so we tuned down the learning rate (0.5×) and used $\lambda_i = 0.2$ instead.

Please refer to our original paper [1] for extra experiments such as adding multiple tasks one by one, and the effect of architecture and some hyperparameter and design choices.

3.2.4 Advantage and Limitation

Requiring access to a pre-trained model's dataset can be restrictive, because they can be not only difficult to store, transfer, and manage, but also undesirable due to user privacy, security, proprietary concerns, or reducing competitive advantage. We address the problem of adapting a vision system to a new task while preserving performance on original tasks, without access to training data for the original tasks. We propose the Learning without Forgetting method for convolutional neural networks, which can be seen as a hybrid of knowledge distillation and fine-tuning, learning parameters that are discriminative for the new task while preserving outputs for the original tasks on the training data. We show the effectiveness of our method on a number of classification tasks, outperforming the popular transfer learning technique fine-tuning in most cases.

There are a few limitations to our method. First, it is worth pointing out that LwF operates on distinct tasks. Like many multitask learning methods, it cannot properly deal with domains that are continually changing on a spectrum, or when it needs to identify which task it should be evaluating. Second, in contrast to methods such as never ending learning [54], LwF requires all new task training data to be present before computing their old task responses. Third, the ability of LwF to incrementally learn new tasks is limited, as the performance of old tasks gradually drop. Finally, as observed in Section 3.2.3, the performance of LwF largely depends on how much the new task data resembles the old task's, as shown in Table 3.0(a), and visualized in Fig. 3.3. Much performance is sacrificed when the two datasets are very different from each other, yet this occurs in many practical scenarios where completely new classes are to be added to a pool of existing classes, e.g. in class-incremental learning. We discuss a possible remedy in the following sections.

3.3 IMPROVEMENTS WITH DEEPINVERSION

In the absence of prior data or metadata, an interesting question arises – can we somehow recover training data from the already trained model and use it for knowledge transfer? A few methods have attempted to visualize what a trained deep network expects to see in an image [127, 128, 129, 130]. The most popular and simple-to-use method is DeepDream [129]. It synthesizes or transforms an input image to yield high output responses for chosen classes in the output layer of a given classification model. This method optimizes the input (random noise or a natural image), possibly with some regularizers, while keeping the selected output activations fixed, but leaves intermediate representations constraint-free. The resulting "dreamed" images lack natural image statistics and can be quite easily identified as

unnatural. These images are also not very useful for transferring knowledge, as our extensive experiments in Section 3.3.2 show.

We make an important observation about deep networks that are widely used in practice – they all implicitly encode very rich information about prior training data. Almost all high-performing convolutional neural networks (CNNs) such as ResNets [131], DenseNets [132], or their variants, use the batch normalization layer [78]. These layers store running means and running variances of the activations at multiple layers. In essence, they store the history of previously seen data, at multiple levels of representation. By assuming that these intermediate activations follow a Gaussian distribution with mean and variance equal to the running statistics, we show that we can obtain "dreamed" images (Fig. 3.5) with high fidelity and realism at a high resolution, and much closer to the distribution of the training dataset as compared to prior work in this area, without requiring any training data or metadata.

Using our DeepInversion technique, we empower a new class of "data-free" applications of immense practical importance which need neither any natural image nor labeled data, greatly improving other data-free approaches. We demonstrate three knowledge transfer applications in the original paper [2]: network pruning, knowledge distillation (to a randomly initialized network), and class-incremental learning.

In the following sections, we show how DeepInversion-based class-incremental learning improves over Learning without Forgetting by introducing synthetic images that are more closely distributed to the original data than the proxy data used in Learning without Forgetting (i.e. the new task images).

Acknowledgement of contribution. The rest of this chapter is done in collaboration with Hongxu Yin and Pavlo Molchanov, first authors of the original Dreaming to Distill paper [2]. My contributions are (1) proposing the idea of DeepInversion with batch normalization statistics (excluding Adaptive DeepInversion), and (2) the experiments on class-incremental learning.

3.3.1 Method

Our new data-free knowledge distillation framework consists of two steps: (i) model inversion, and (ii) application-specific knowledge distillation. In this section, we briefly discuss the background and notation, and then introduce our *DeepInversion* methods.

DeepDream [129]. Originally formulated by Mordvintsev et al. to derive artistic effects on natural images, DeepDream is also suitable for optimizing noise into images. Given a randomly initialized input ($\hat{x} \in \mathcal{R}^{H \times W \times C}$, H, W, C being the height, width, and number of

color channels) and an arbitrary target label y, the image is synthesized by optimizing

$$\min_{\hat{x}} \mathcal{L}(\hat{x}, y) + \mathcal{R}(\hat{x}), \tag{3.2}$$

where $\mathcal{L}(\cdot)$ is a classification loss (e.g. cross-entropy), and $\mathcal{R}(\cdot)$ is a image regularization term. DeepDream uses an image prior [133, 128, 130, 134] to steer $\hat{x}$ away from unrealistic images with no discernible visual information:

$$\mathcal{R}_{\text{prior}}(\hat{x}) = \alpha_{\text{tv}}\mathcal{R}_{\text{TV}}(\hat{x}) + \alpha_{\ell_2}\mathcal{R}_{\ell_2}(\hat{x}), \tag{3.3}$$

where R_{TV} and R_{ℓ_2} penalize the total variance and ℓ_2 norm of $\hat{x}$, respectively, with scaling factors α_{tv}, α_{ℓ_2}. As both prior work [128, 129, 130] and we empirically observe, image prior regularization provides more stable convergence to valid images. However, these images still have a distribution far different from natural (or original training) images and thus lead to unsatisfactory knowledge distillation results.

DeepInversion. We improve DeepDream's image quality by extending the image regularization $\mathcal{R}(\hat{x})$ with a new feature distribution regularization term. The image prior term defined previously provides little guidance for obtaining a synthetic $\hat{x} \in \hat{\mathcal{X}}$ that contains similar low- and high-level features as $x \in \mathcal{X}$. To effectively enforce feature similarities at all levels, we propose to minimize the distance between feature map statistics for $\hat{x}$ and x. We assume that feature statistics follow the Gaussian distribution across batches and, therefore, can be defined by mean μ and variance σ^2. Then, the *feature distribution regularization* term can be formulated as:

$$\begin{aligned}\mathcal{R}_{\text{feature}}(\hat{x}) = &\sum_l || \ \mu_l(\hat{x}) - \mathbb{E}(\mu_l(x)|\mathcal{X}) \ ||_2 + \\ &\sum_l || \ \sigma_l^2(\hat{x}) - \mathbb{E}(\sigma_l^2(x)|\mathcal{X}) \ ||_2,\end{aligned} \tag{3.4}$$

where $\mu_l(\hat{x})$ and $\sigma_l^2(\hat{x})$ are the batch-wise mean and variance estimates of feature maps corresponding to the l^{th} convolutional layer. The $\mathbb{E}(\cdot)$ and $||\cdot||_2$ operators denote the expected value and ℓ_2 norm calculations, respectively.

It might seem as though a set of training images would be required to obtain $\mathbb{E}(\mu_l(x)|\mathcal{X})$ and $\mathbb{E}(\sigma_l^2(x)|\mathcal{X})$, but the running average statistics stored in the widely-used batchnorm (BN) layers are more than sufficient. A BN layer normalizes the feature maps during training to alleviate covariate shifts [78]. It implicitly captures the channel-wise means and variances during training, hence allows for estimation of the expectations in eq. 3.4 by:

$$\mathbb{E}\big(\mu_l(x)|\mathcal{X}\big) \simeq \text{BN}_l(\text{running\_mean}), \tag{3.5}$$

$$\mathbb{E}\big(\sigma_l^2(x)|\mathcal{X}\big) \simeq \text{BN}_l(\text{running\_variance}). \tag{3.6}$$

As we will show, this feature distribution regularization substantially improves the quality of the generated images. We refer to this model inversion method as *DeepInversion* - a generic approach that can be applied to any trained *deep* CNN classifier for the *inversion* of high-fidelity images. The regularization $R(\cdot)$ (corr. to eq. 3.2) can thus be expressed as

$$\mathcal{R}_{\text{DI}}(\hat{x}) = \mathcal{R}_{\text{prior}}(\hat{x}) + \alpha_{\text{f}}\mathcal{R}_{\text{feature}}(\hat{x}). \tag{3.7}$$

Adaptive DeepInversion. In addition to quality, diversity also plays a crucial role in avoiding repeated and redundant synthetic images. Please refer to the main paper [2] on Adaptive DeepInversion, which encourages synthesis of images that cause student-teacher disagreement. Its competitive and interactive nature gradually force new image features to emerge, leading to constantly-evolving students.

3.3.2 Experiments Outline

We demonstrate our inversion methods on datasets of increasing size and complexity. We perform a number of ablations to evaluate each component in our method on the simple CIFAR-10 dataset (32×32 pixels, 10 classes). Then, on the complex ImageNet dataset (224×224 pixels, 1000 classes), we show the success of our inversion methods on three different applications under the data-free setting - (a) pruning, (b) knowledge transfer, and (c) continual (class-incremental) learning. In all experiments, our image pixels are initialized i.i.d. from Gaussian noise of $\mu = 0$ and $\sigma = 1$.

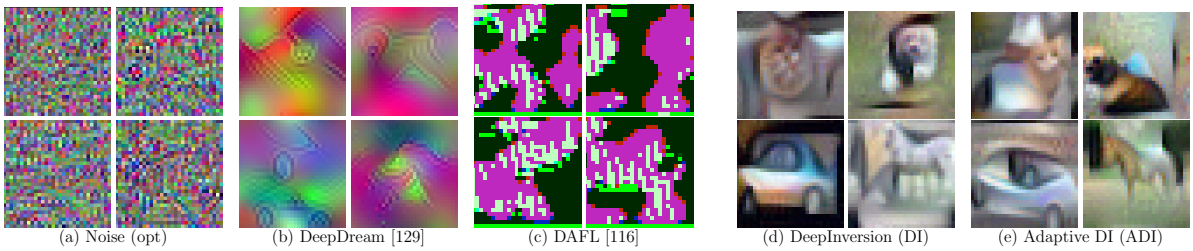
3.3.3 Synthesis Results on CIFAR-10

For validating our design choices, we consider the task of data-free knowledge distillation, where we teach a student network randomly initialized from scratch.

Implementation details. We use VGG-11-BN and ResNet-34 networks pretrained on CIFAR-10 as the teachers. For all image synthesis in this section, we optimize using Adam (learning rate 0.05). We generate 32×32 images in batches of 256. Each image batch requires 2k gradient updates. After a simple grid search optimizing for student accuracy, we found $\alpha_{\text{tv}} = 2.5\cdot10^{-5}, \alpha_{\ell_2} = 3\cdot10^{-8}$, and $\alpha_f = \{1.0, 5.0, 10.0, 100.0\}$ to work best for DeepInversion.

Teacher Network	VGG-11	VGG-11	ResNet-34
Student Network	VGG-11	ResNet-18	ResNet-18
Teacher accuracy	92.34%	92.34%	95.42%
Noise ($\mathcal{L}$)	13.55%	13.45%	13.61%
$+\mathcal{R}_{\text{prior}}$ (DeepDream [129])	36.59%	39.67%	29.98%
$+\mathcal{R}_{\text{feature}}$ (DeepInversion)	84.16%	83.82%	91.43%
$+\mathcal{R}_{\text{compete}}$ (ADI)	**90.78%**	**90.36%**	**93.26%**
DAFL [116]	–	–	92.22%

Table 3.2: Data-free knowledge transfer to various students on CIFAR-10. For ADI, we generate one new batch of images every 50 KD iterations and merge the newly generated images into the existing set of generated images.



(a) Noise (opt) (b) DeepDream [129] (c) DAFL [116] (d) DeepInversion (DI) (e) Adaptive DI (ADI)

Figure 3.4: Generated 32×32 images by inverting a ResNet-34 trained on CIFAR-10 with different methods. All images are correctly classified by the network, clockwise: cat, dog, horse, car.

We run each knowledge distillation experiment between the teacher and student networks for 250 epochs in all, with an initial learning rate of 0.1, decayed every 100 epochs with a multiplier of 0.1. One epoch corresponds to 195 gradient updates.

Baselines – Noise & DeepDream [129]. From Table 3.2, we observe that optimized noise, Noise ($\mathcal{L}$), does not provide any support for knowledge distillation - a drastic change in input distribution disrupts the teacher and impacts the validity of the transferred knowledge. Adding R_{prior}, like in DeepDream, slightly improves the student's accuracy.

Effectiveness of DeepInversion (R_{feature}). Upon adding R_{feature}, we immediately find large improvements in accuracy of $40\%-69\%$ across all the teaching scenarios. DeepInversion images (Fig. 3.4(d)) are vastly superior in realism, as compared to baselines (Fig. 3.4(a,b)).

Comparison with DAFL [116]. We further compare our method with DAFL [116], which trains a new generator network to convert noise into images while with a fixed teacher. As seen in Fig. 3.4(c), we notice that these images are "unrecognizable", reminiscent of "fooling images" [130]. Our method enables higher visual fidelity of images and eliminates the need of an additional generator network, while gaining higher student accuracy under the same setup.

Figure 3.5: Class-conditional 224×224 samples obtained by DeepInversion, given only a ResNet-50 classifier trained on ImageNet and no additional information. Random classes. Note that the images depict classes in contextually correct backgrounds, in realistic scenarios. Best viewed in color.

3.3.4 Synthesis Results on ImageNet

After successfully demonstrating our method's abilities on the small CIFAR-10 dataset, we move on to examine its effectiveness on the large-scale ImageNet dataset [118]. We first run DeepInversion on networks trained on ImageNet, and perform quantitative and qualitative analysis. Then, we show the effectiveness of synthesized images on three different tasks of immense importance: data-free pruning, data-free knowledge transfer, and data-free continual learning.

Implementation details. For all experiments in this section, we use the publicly available pretrained ResNet-{18, 50} from PyTorch as the fixed teacher network, with top-1 accuracy of $\{69.8\%, 76.1\%\}$. For image synthesis, we optimize using Adam (learning rate 0.05). We set $\alpha_{\text{tv}} = 1 \cdot 10^{-4}, \alpha_{\ell_2} = \{0, 1 \cdot 10^{-2}\}, \alpha_f = 1 \cdot 10^{-2}$ for DeepInversion. We synthesize 224×224 images in batches of $1,216$ using 8 NVIDIA V100 GPUs and automatic-mixed precision (AMP) [135] acceleration. Each image batch takes 20k updates over 2h.

The total variance regularization $\mathcal{R}_{\text{TV}}$ in eq. 3.3 is based on the sum of ℓ_2 norms between the base image and its shifted variants: (i) two diagonal shifts, (ii) one vertical shift, and (iii) one horizontal shift, all by one pixel. We apply random flipping and jitter (< 30 pixels) on the input before each forward pass. We use the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1 \cdot 10^{-8}$ given a constant learning rate of 0.05. We speed up the training process using half-precision floating point (FP16) via the APEX library.

Figure 3.6: Class-conditional 224 × 224 images obtained by DeepInversion given a ResNet50v1.5 classifier pretrained on ImageNet. Grouped by classes. Top to bottom: (left) daisy, volcano, quill, (right) cheeseburger, brown bear, trolleybus.

3.3.5 Analysis and Concerns of Synthesized Images

Fig. 3.5 shows images generated by DeepInversion from an ImageNet-pretrained ResNet-50 in random order, and Fig. 3.6 selectively shows synthetic images among six of the classes from a ResNet-50 v1.5. Remarkably, given just the model, we observe that DeepInversion is able to generate images with high fidelity and resolution. It also produces detailed image features and textures around the target object, e.g., clouds surrounding the target balloon, water around a catamaran, forest below the volcano, etc.. Images within the same class shows some variety, but the viewing angles seem somewhat fixed.

Generalizability. In order to verify that the generated images do not overfit to just the inverted model, we obtain predictions using four other ImageNet networks. As seen in Table 3.3, images generated using a ResNet-50 generalize to a range of models and are correctly classified. Further, DeepInversion outperforms DeepDream by a large margin. This indicates robustness of our generated images while transferring across networks.

Model	DeepDream top-1 acc. (%)	DeepInversion top-1 acc. (%)
ResNet-50	100	100
ResNet-18	28.0	**94.4**
Inception-V3	27.6	**92.7**
MobileNet-V2	13.9	**90.9**
VGG-11	6.7	**80.1**

Table 3.3: Classification accuracy of ResNet-50 synthesized images by other ImageNet-trained CNNs.

Method	Resolution	GAN	Inception Score
BigGAN [136]	256	✓	178.0 / 202.6^{+}
DeepInversion (**Ours**)	224		60.6
SAGAN [137]	128	✓	52.5
SNGAN [138]	128	✓	35.3
WGAN-GP [139]	128	✓	11.6
DeepDream [129]*	224		6.2

Table 3.4: Inception Score (IS) obtained by images synthesized by various methods on ImageNet. SNGAN ImageNet score from [140]. *: our implementation. $^{+}$: BigGAN-deep.

Inception score (IS). We also compare the IS [141] of our generated images with other methods in Table 3.4. Again, DeepInversion substantially outperforms DeepDream by an improvement of 54.2. Without sophisticated training, DeepInversion even beats multiple GAN baselines that have limited scalibility to high image resolutions.

Privacy. In general, model inversion techniques have raised privacy concerns that attackers can obtain sensitive information from a trained model. For example, when each class corresponds to an identifiable person, attackers have been able to reconstruct a person's face to some extent given the corresponding class index [142, 143].

However, in our case of object classification, this concern is mitigated since there is no single entity associated with one class, much like in prior non-data-free methods [144, 145, 127]. Any generated sample would be dissociated with actual entities in that class. The method may further mix up different entities' distinct features to form the same generated sample. This is because we model the entire distribution of the dataset, rather than generate actual identifiable members of a specific instantiation of that distribution. Figure 3.7 shows the nearest neighbors in the feature space of generated images in the real dataset. The generated images do not resemble any retrieved real data.

However, future work in model inversion attacks may well be able to reconstruct actual

Figure 3.7: Nearest real image neighbors of DeepInversion synthesized images in the ResNet-50-avgpool feature space for the ImageNet class "handheld computer".

members of the original training data, perhaps given some extra identifying information. We believe counter-measures against model inversion methods in general are valuable research directions. Our goal is to generate the training distribution without the aid of the dataset owner, but we cannot circumvent counter-measures placed by the owner. It is possible to prohibit image generation from the model in its license, and an easy counter-measure is to merge a batch normalization layer's running mean and variance with its weight and bias terms to destroy the statistics without changing its test-time behavior.

3.3.6 Results on Data-free Class-incremental Learning

In the class-incremental setting, the original data has classes $\mathcal{C}_o$, and new data $(x_k, y_k),\ y_k \in \mathcal{C}_k$. The resulting model is now required to make predictions in a combined output space $\mathcal{C}_o \cup \mathcal{C}_k$. Similar to previous sections, we take a trained network $\big($denoted $p_o(\cdot)$, effectively as a *teacher*$\big)$, make a copy $\big($denoted $p_k(\cdot)$, effectively as a *student*$\big)$, and then add new randomly initialized neurons to $p_k(\cdot)$'s final layer to output logits for the new classes. We train $p_k(\cdot)$

Method	Top-1 acc. (%)			
	Combined	ImageNet	CUB	Flowers
ImageNet + CUB (1000 → 1200 outputs)				
LwF.MC [45]	47.64	53.98	41.30	–
DeepDream [129]	63.00	56.02	**69.97**	–
DeepInversion (**Ours**)	**67.61**	**65.54**	69.68	–
Oracle (distill)	69.12	68.09	70.16	–
Oracle (classify)	68.17	67.18	69.16	–
ImageNet + Flowers (1000 → 1102 outputs)				
LwF.MC [45]	67.23	55.62	–	78.84
DeepDream [129]	79.84	65.69	–	**94.00**
DeepInversion (**Ours**)	**80.85**	**68.03**	–	93.67
Oracle (distill)	80.71	68.73	–	92.70
Oracle (classify)	79.42	67.59	–	91.25
ImageNet + CUB + Flowers (1000 → 1200 → 1302 outputs)				
LwF.MC [45]	41.72	40.51	26.63	58.01
DeepInversion (**Ours**)	**74.61**	**64.10**	**66.57**	**93.17**
Oracle (distill)	76.18	67.16	69.57	91.82
Oracle (classify)	74.67	66.25	66.64	91.14

Table 3.5: Continual learning results extending the network output space, adding new classes to ResNet-18. Accuracy over *combined* classes $\mathcal{C}_o \cup \mathcal{C}_k$ reported on individual datasets. Average over datasets also shown (datasets treated equally regardless of size, so ImageNet samples have less weight than CUB or Flowers samples).

to classify simultaneously over all classes, old and new, while network $p_o(\cdot)$ remains fixed.

Method	Top-1 acc. (%)			
	Combined	ImageNet	CUB	Flowers
ImageNet + CUB (1000 → 1200 outputs)				
LwF.MC [45]	47.43	64.38	30.48	–
DeepInversion (**Ours**)	**70.72**	**68.35**	**73.09**	–
Oracle (distill)	72.71	71.95	73.47	–
Oracle (classify)	72.03	71.20	72.85	–
ImageNet + Flowers (1000 → 1102 outputs)				
LwF.MC [45]	67.67	65.10	–	70.24
DeepInversion (**Ours**)	**82.47**	**72.11**	–	**92.83**
Oracle (distill)	83.07	72.84	–	93.30
Oracle (classify)	81.56	71.97	–	91.15

Table 3.6: Results on VGG-16-BN. Experiment setup same as Table 3.5

Continual learning loss. We formulate a new loss with DeepInversion images as follows. We use same-sized batches of DeepInversion data $(\hat{x}, p_o(\hat{x}))$ and new class real data (x_k, y_k)

for each training iteration. For $\hat{x}$, we use the original model to compute its soft labels $p_o(\hat{x})$, i.e., class probability among old classes, and then concatenate it with additional zeros as new class probabilities. We apply a KL-divergence loss between predictions $p_o(\hat{x})$ and $p_k(\hat{x})$ on DeepInversion images for prior memory, and a cross-entropy loss between one-hot y_k and prediction $p_k(x_k)$ on new class images for emerging knowledge. Similar to prior work [1, 45], we also apply a third KL-divergence term between the new class images' old class predictions $p_k(x_k|y \in \mathcal{C}_o)$ and their original model predictions $p_o(x_k)$. This forms the loss

$$\begin{aligned}\mathcal{L}_{\text{CL}} =& \text{KL}\big(p_o(\hat{x}), p_k(\hat{x})\big) + \mathcal{L}_{xent}\big(y_k, p_k(x_k)\big) \\ &+ \text{KL}\big(p_o(x_k|y \in \mathcal{C}_o), p_k(x_k|y \in \mathcal{C}_o)\big).\end{aligned} \tag{3.8}$$

Evaluation results. We add new classes from the CUB [123], Flowers [146], and both CUB and Flowers datasets to a ResNet-18 [131] classifier trained on ImageNet [118]. Prior to each step of addition of new classes, we generate 250 DeepInversion images per old category. We compare with prior class-incremental learning work LwF.MC [45] as opposed to the task-incremental LwF [1] that cannot distinguish between old and new classes. We further compare with oracle methods that break the data-free constraint: we use the same amount of real images from old datasets in place of $\hat{x}$, with either their ground truth for classification loss or their soft labels from $p_o(\cdot)$ for KL-divergence distillation loss. The third KL-divergence term in eq. 3.8 is omitted in this case. Implementation details are similar to previous sections and detailed in the original paper's supplemental material.

Results are shown in Table 3.5. Our method significantly outperforms LwF.MC in all cases and leads to consistent performance improvements over DeepDream in most scenarios. We are very close to the oracles (and occasionally outperform them), showing DeepInversion's efficacy in replacing ImageNet images for continual learning. We verify results on VGG-16 in Table 3.6. The observations are similar with a slightly larger gap between DeepInversion and oracle methods.

One method in particular, iCaRL [45], would be interesting but unfair to compare to. iCaRL uses stored samples to train the underlying representation. It also uses them to compute the class average features, and the one nearest to a test sample's feature is used for prediction. Experimentally, iCaRL learns many (5 to 100) groups of classes incrementally, splits the class groups randomly so each group is similar to another, and starts with a model trained using a small dataset. In comparison, we add relatively smaller numbers of classes to 1000 existing ImageNet classes in 2 to 3 groups, split groups by datasets so there are much more differences between samples from different groups, and start with a well-trained model. iCaRL would perform inferior to oracle methods, and we leave the comparison to

future work.

3.3.7 Discussions and Limitations

Strengths and novelty. The most significant departure from prior work such as EWC [46] is that our DeepInversion-based continual learning can operate on *any* regularly-trained model, given the widespread usage of batch normalization layers. Our method eliminates the need for any collaboration from the model provider, even when the model provider (1) is unwilling to share any data, (2) is reluctant to train specialized models for continual learning, or (3) does not have the know-how to support a downstream continual learning application. This gives machine learning practitioners more freedom and expands their options when adapting existing models to new usage scenarios, especially when data access is restricted.

Image synthesis time. Generating 215K ImageNet samples of 224×224 px for a ResNet-50 takes 2.8K NVIDIA V100 GPU-hours, or 22 hours on 128 GPUs. This time scales linearly with the number of images to synthesize. The multi-resolution scheme in our original paper [2] reduces this time by 10.7× (0.26K GPU-hours / 4 hours on 64 GPUs).

Image color and background similarity. We believe there are two possible reasons. 1) The method uncovers and visualizes the unique discriminative characteristics of a CNN classifier, which can guide future work on neural network understanding and interpretation. Post-training, the network learns to capture only the informative visual representations to make a correct classification. For example, the key features of a target object are retained, e.g. detailed bear heads as in Fig. 8 or the fur color/patterns of penguins and birds in Fig. 5, whereas the background information is mostly simplified, e.g., green for grass or blue for ocean. 2) For all the images synthesized in this section, we use a default Gaussian distribution with zero mean and unit variance to initialize all the pixels, hence may be subject to unimodal behaviors. We have also observed that the style varies with the choice of the optimization hyperparameters.

Continual learning class similarity. We implemented DeepInversion on iCIFAR and iILSVRC [45] (two splits), but we obtain statistically equivalent or slightly worse performance compared to LwF.MC. We suspect that our synthesized images are more effective in replacing old class images that are *different* from the new images, compared to the case where the old and new images are similar (e.g., random subsets of a pool of classes).

CHAPTER 4: TASK-ASSISTED DOMAIN ADAPTATION WITH ANCHOR TASKS

Collecting annotations is difficult for geometric tasks, such as predicting depth [147, 148], surface normals [29], and 3D pose [149], because it usually requires a specialized device and access to the scene. Synthetic images and their geometric labels are easily generated, but synthetically trained models often do not generalize well to real data. Unsupervised domain adaptation methods [150, 61, 62, 63] can help, but they often blindly minimize the domain distribution difference [65, 151, 64, 66] even when the ground truth distributions in source and target differ. How can we better adapt from synthetic to real data?

In this section, we propose to use *anchor tasks* as a guide for improving pixel-level domain transfer of the *main task*. The anchor task is a task labeled on both domains, whose annotations are already available or automatically generated. For example, in one experiment we improve transfer for surface normal prediction on faces by using facial keypoint detection as an anchor task, and the anchor task ground truth is estimated using an off-the-shelf model. We propose our HeadFreeze method that first trains the main task and anchor task on synthetic data, then freezes the top few layers and retrain the feature layers to perform the main task on synthetic data and the anchor task on both domains. The fully-supervised anchor task provides additional semantic and spatial context information to learn better feature representations for the real images (hence the name "anchor"), while the frozen top layers leverage learned "cross-task guidance" so that the main task on the target domain can be guided by the anchor task.

4.1 MOTIVATION

Our idea can be seen as a generalization of existing works that use a closely related auxiliary task to help adaptation. We call this family of methods "Task-assisted Domain Adaptation" (TADA). Prior work [152, 28, 153, 154] in the TADA family all rely on problem-specific, explicitly defined mappings between the auxiliary tasks and the main tasks (see Section 2 for details), and thus are restricted to their own task pair and problem settings. Unlike prior work, we require only that the anchor task has pixel labels, and HeadFreeze is applicable even when the main-anchor relationship lacks an explicit formulation (e.g. between facial keypoints and surface normal map). We demonstrate this with different anchor tasks for the same main task without changing the framework. Our experiments focus on geometric tasks with synthetic and real images as the source and target domains, but our approach also applies to other pixel labeling domain transfer problems.

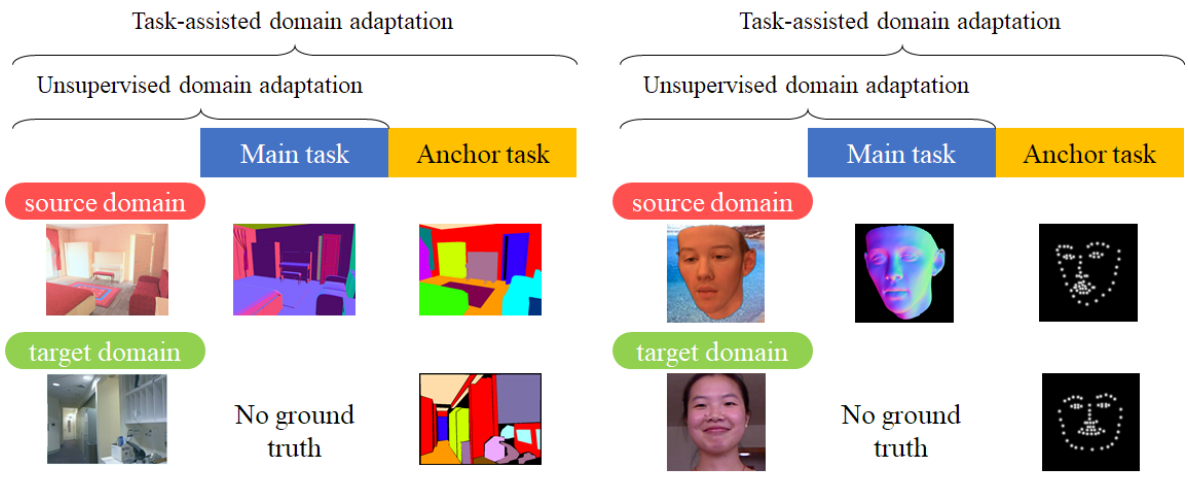


Figure 4.1: Illustration of our formulation compared to unsupervised domain adaptation. Although target domain main task labels are hard or expensive to obtain, we can use cheaper, easily available labels from an "anchor" task to help align the domains with clear correspondence between images and the anchor task label space.

The anchor task helps domain adaptation in two ways. First, learning shared features for the anchor task on both domains and the main task on the source domain encourages that the features are effective for the main task in the target domain. Second, there may be a multitask learning benefit, if the anchor task and main task have related labels (e.g. "ceiling" is always horizontal) or the same features are useful for both tasks.

Our HeadFreeze method strengthens the first benefit while being simple enough not to require domain knowledge on the task pair. Specifically, when our network finishes training on the source domain, its final layers have learned not to output unlikely main-anchor prediction pairs (e.g. flat nose, misaligned object edges) but to output likely pairs. We term this knowledge "**cross-task guidance**". But this guidance may be ignored if the network overfits to the target domain anchor task and outputs unlikely pairs at will. Freezing the final layers fixes the guidance, and ensures the main and anchor task classifiers continue to rely on the same features and the same mapping from feature space to label space.

4.2 RELATED WORK

Domain adaptation methods using auxiliary tasks that are constrained to specific task pairs (TADA methods). An emerging line of work recently is using multi-task learning or weakly supervised learning to help unsupervised domain adaptation. Gebru et al. [152] adapt fine-grain classification between an easy domain and in-the-wild images

with the help of classes' attributes. They adapt a consistency loss between attributes and classes and domain adaptation losses from Tzeng et al. [59]. Yang et al. [28] adapts lab-environment 3D human pose estimation for in-the-wild data with only 2D pose ground truth, by jointly training on 2D and 3D labels and aligning domains with a GAN-based discriminator. Fang et al. [154] adapts a robot grasping application from simulation to real images, and from the indiscriminate grasping task to instance-specific grasping. They perform joint training on all existing labels and the optional input of the instance mask. Inoue et al. [153] adapts image object detection to paintings by generating pseudo-labels which are filtered using auxiliary image-level labels.

Our method has two major differences from these prior work. (1) All prior work have very application-specific constraint formulation on the their task-pair relationship, making them inapplicable to nearly all other tasks. Gebru et al. [152] assumes the auxiliary and main annotation to have a known linear relationship. Yang et al. [28] constraints the 2D pose and 3D pose to use the same 2D pose output layer. Fang et al. [154] assumes both tasks' output are both binary prediction, share the same output neuron, and the tasks are differentiated by an extra input. Inoue et al. [153] must use a hard-coded procedure to filter erroneous outputs of the main detection task using the anchor task classification labels. In contrast, our work requires only that the two tasks' annotations are spatial, without any constraint on the output layers or the loss of each task – a much weaker assumption – and models the cross-task guidance without hard-coded domain knowledge. Our experiments show that the TADA formulation helps task transfer scenarios beyond these task-specific designs with explicit task relations. (2) We focus on tasks with pixel-wise outputs, such as surface normal estimation or keypoint detection (in the form of heatmaps for each keypoint).

Combining multi-task learning and domain adaptation [59, 155, 27] is topically similar. Besides the TADA works we mentioned earlier, most assume all tasks' labels from the target domain are available, and some still requires specially designed losses or constraints for the task pair (e.g. Tzeng et al. [59] constrains the all tasks to be classification tasks). Whether their formulation are still effective in our unsupervised case is beyond the scope of our paper.

Modeling output spatial structure [156, 66, 151] is related to how we preserve the cross-task guidance between two tasks' outputs. Mostajabi et al. [156] regularizes semantic segmentation by training an autoencoder on the *semantic labels*, and force the network to use the fixed decoder to output its prediction. We are inspired by these ideas, but are focused on how *two* tasks' output spaces interact, and generalizing across domains.

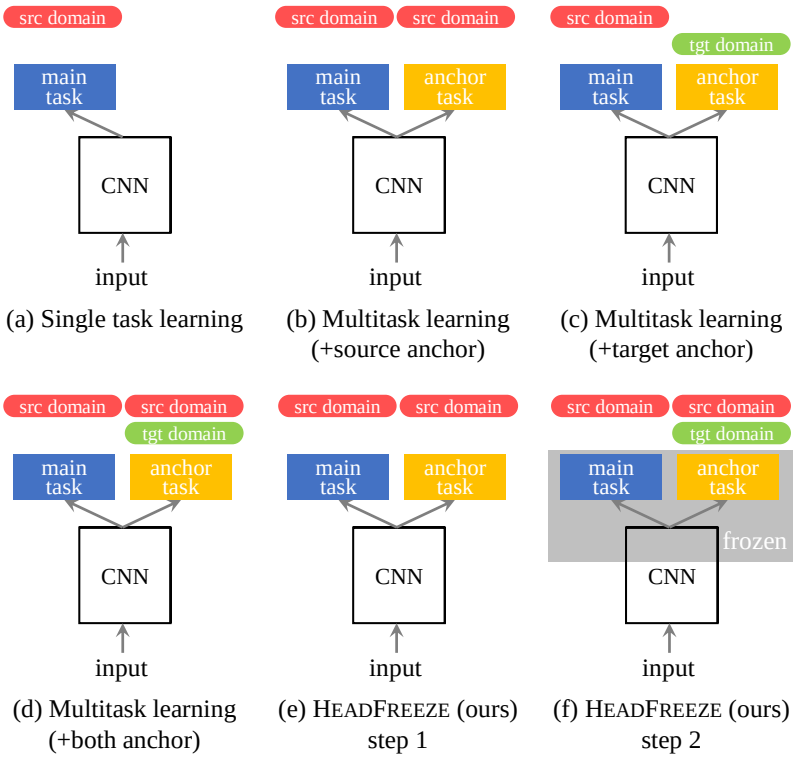


Figure 4.2: Illustration of various compared methods and their training label usage. TADA methods (d-f) uses the anchor task on *both* domains to establish clear correspondence in the anchor task annotation space. Our HeadFreeze method first trains only on the source domain, and then freezes the final network layers to consolidate the learned cross-task spatial and contextual guidance in the output.

4.3 METHOD

To formulate our Task-Assisted Domain Adaptation (TADA), we first start from a brief review of unsupervised domain adaptation (UDA). In UDA, we have labeled data in the source domain $(x_{\mathcal{S}}, y_{\mathcal{S}}) \in \mathcal{S}$, and unlabeled data in the target domain $(x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}$. However, only the test set in $\mathcal{T}$ may contain labels $y_{\mathcal{T}}$ for evaluation purposes, and in the train set, $(x_{\mathcal{T}}, \varnothing) \in S_{tr}$ is provided. A model, usually with the form of $\hat{y} = g(f(x))$, is trained on all available data, where $f(\cdot)$ is the network backbone for input-feature mapping, and $g(\cdot)$ is the network head for feature-prediction mapping. In this chapter, unless otherwise specified, we refer to the networks' second-to-last layer output as the "*features*".

Usually, to reduce the domain gap, features in both domains $f(x_{\mathcal{S}})$ and $f(x_{\mathcal{T}})$ are encouraged to follow the same distribution [65] (although this can also be done in output space $g(f(x.))$ as well [66]). However, it is usually not guaranteed that ground truths $y_{\mathcal{S}}$, $y_{\mathcal{T}}$ follow the same distribution. When the ground truths distribute differently, the ideal features and

outputs have to distribute differently too. Forcing either of them to distribute similarly would deviate the prediction from the ground truth.

In our Task-Assisted Domain Adaptation scenario, in addition to the main task, an anchor task is defined for both domains. The domains become $(x_{\mathcal{S}}, y_{\mathcal{S}m}, y_{\mathcal{S}a}) \in \mathcal{S}$, and $(x_{\mathcal{T}}, y_{\mathcal{T}m}, y_{\mathcal{T}a}) \in \mathcal{T}$. Here, m and a stand for the main and anchor tasks. In the train set of $\mathcal{T}$, only $(x_{\mathcal{T}}, \varnothing, y_{\mathcal{T}a}) \in T_{tr}$ is provided, while $y_{\mathcal{T}m}$ is unknown or unavailable. A model, usually with the form of $\hat{y}_m = g_m(f(x)),\ \hat{y}_a = g_a(f(x))$ is trained on those data, where $g_m(\cdot)$ and $g_a(\cdot)$ are sub-modules specific to each task. In this chapter, we focus on the popular formulation above where the two tasks share the same network backbone $f(\cdot)$.

In this chapter, we consider that the anchor task exists solely to aid the learning of the main task. We evaluate only on the target domain main task, not on the anchor. If the anchor task is important, one can always train a separate model for it using a variety of transfer learning methods.

4.3.1 MTL + Anchor for Effective Feature Learning

When prior work has performed multi-task learning (MTL), either all tasks are assumed to be in one domain, or one task is available in each domain (main in $\mathcal{S}$, anchor in $\mathcal{T}$). Formally, there can be three supervised losses in the TADA scenario:

$$\mathcal{L}_{\mathcal{S}m} = \mathcal{L}_m\left(y_{\mathcal{S}m}, \hat{y}_{\mathcal{S}m}\right), \tag{4.1}$$

$$\mathcal{L}_{\mathcal{S}a} = \mathcal{L}_a\left(y_{\mathcal{S}a}, \hat{y}_{\mathcal{S}a}\right), \tag{4.2}$$

$$\mathcal{L}_{\mathcal{T}a} = \mathcal{L}_a\left(y_{\mathcal{T}a}, \hat{y}_{\mathcal{T}a}\right). \tag{4.3}$$

In prior work, a multitask learning loss may only comprise of two of the three:

$$\mathcal{L}_{\text{MTL (+src anchor)}} = \mathcal{L}_{\mathcal{S}m} + \lambda\mathcal{L}_{\mathcal{S}a},\ \text{or} \tag{4.4}$$

$$\mathcal{L}_{\text{MTL (+tgt anchor)}} = \mathcal{L}_{\mathcal{S}m} + \lambda\mathcal{L}_{\mathcal{T}a}, \tag{4.5}$$

for "everything in source" and "one task per domain" respectively. We instead use the alternative baseline – MTL (+both anchor), which simply uses all the supervised losses.

$$\mathcal{L}_{\text{MTL (+both anchor)}} = \mathcal{L}_{\mathcal{S}m} + \lambda\mathcal{L}_{\mathcal{S}a} + \lambda\mathcal{L}_{\mathcal{T}a}. \tag{4.6}$$

Differences between these formulations are illustrated in Figure 4.2.

We suggest two ways of choosing the anchor task and obtaining its annotations. (1) Some

anchor annotations can be freely obtained, e.g. from very robust estimators that work across most domains, such as facial keypoint detectors. (2) Some anchor tasks can be popular tasks and already have labels in many datasets, such as semantic segmentation. It should be chosen so obtaining it is much easier than the main task annotation.

It may be tempting to hypothesize that the baseline losses in eq. 4.4, 4.5 will be enough for the TADA scenario, and that collecting anchor labels on both domains is not necessary. Maybe in eq. 4.4 the multitask learning aspect can already improve model generalization, and in eq. 4.5 the network is trained on the target domain, so it may be forced to adapt to perform well on the anchor task. One can also add an unsupervised domain adaptation loss to reduce the domain gap. We experimentally show that these baselines underperform MTL (+both anchor) and degrade performance.

In addition to any of these supervised losses, an unsupervised domain adaptation loss can be added. For example, adversarial losses (a.k.a. GAN losses) on the features or the output space are used in prior work [65, 66] with a discriminator network $d(\cdot)$ trained in a mini-max fashion:

$$\min_{f,g} \max_{d} \mathcal{L}(f, g) + \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}(f, g, d). \tag{4.7}$$

We refer our readers to the prior work [65, 66] for the exact formulation of $\mathcal{L}_{\text{adv}}$.

4.3.2 HeadFreeze for Preserving Cross-task Guidance

Building on MTL (+both anchor), we further propose our HeadFreeze method to leverage the cross-task guidance that can be used to guide the target domain main task based on the target anchor task.

The final layers of a trained multitask network can can be seen as a decoder from its input feature space to the joint label space of the two tasks. When we train these layers on the source domain to convergence, they have only learned to predict output pairs for main and anchor tasks that are *contextually and spatially coherent*, and have never learned to output incoherent pairs (such as misaligned object edges or shapes between tasks, and contradictory outputs like vertical ceilings or flat noses). We assume that the final layers can incorporate this coherency knowledge, and are more likely to predict coherent outputs. It follows that the coherency knowledge can act as a cross-task guidance, so training on target anchor task improves the target main task by ruling out incoherent predictions.

However, it is possible that the model overfits to the target domain anchor task, ignores or forgets any cross-task guidance learned in the source domain. We force the cross-task guidance to persist across domains with HeadFreeze. We first train the multitask network

on the source domain, using eq. 4.4. When it approaches convergence (or just before it overfits to one of the tasks), we freeze the parameters of its final layers. We then train only the lower layers jointly on all available labels using eq. 4.6, forcing their output to go through the pre-trained final layers. See Figure 4.2 (e,f) for an illustration. This procedure can be trained end-to-end by modifying the loss and optimizer's list of variables after the convergence of the first step, which is easy to do in modern frameworks such as PyTorch [157].

For implementation details such as network structure and the number of layers frozen, please see Section 4.4.4.

4.4 EXPERIMENT SETUP

We validate our methods and claims on two sets of experiments, facial images and indoor scenes, both adapting from synthetic data to real images – our motivating scenario.

4.4.1 Facial Images

We perform facial surface normal estimation as the main task, and for the anchor task we choose 3D facial keypoint detection with automatically generated ground truth. Intuitively, 3D keypoints can inform surface information, and thus is a good form of guidance. As 3D keypoints can currently be reliably generated by methods that generalize well across domains, we use this to show whether *free* anchor task labels can be helpful for another label-deprived task.

We adapt from synthetic data generated by Sengupta et al. [29] ("SfSsyn"), using 3DMM models [158]. The dataset provides facial images with surface normal ground truth, with synthetic faces both frontal and looking to the side. We change the reference frame of the surface normal to camera coordinates to follow the definition of all other datasets.

For the target domain, we use real data from FaceWarehouse [159] ("FaceWH"). The dataset provides facial models fitted using a morphable model followed by a laplacian-based mesh deformation without any PCA reduction, so the surface normals rendered from them are both clean and faithful to the raw RGBD scan.

None of these two datasets provide an official split. We split the subjects (separated by dataset folders) into 70% for training, and 15% each for validation and test.

We obtain the anchor annotations for free. On both datasets, we use state-of-the-art Bulat et al. [160] to extract both 3D keypoints and 2D keypoints using their separate models. 3D keypoints are used as anchor training ground truth. We compute the facial region mask from

the 2D keypoints for performing evaluation, which is a standard practice in facial surface normal estimation [161, 29].

During training, we use the standard losses for both tasks: for surface normal estimation, cosine loss (see [161]); for 3D keypoint detection, a heatmap regression for the 2D positions, and a vector regression for depth (see [160]). During evaluation of surface normal, we use five metrics in the literature. Specifically, the angular difference between predicted 3D surface normal and the ground truth is treated as the error and computed for each pixel. Then we aggregate the root mean square angular error (RMSE), mean of the error (Mean), median of the error (Median), and percentages of pixels with errors below 11.25° and 30°. Only valid regions are considered, so we ignore pixels outside the face or where there is no ground truth (e.g. where depth is missing and surface normal cannot be correctly estimated).

4.4.2 Indoor Scenes

We again perform surface normal estimation as the main task, but use semantic segmentation as the anchor task to demonstrate, since semantic segmentation has annotations available across many datasets. The semantic boundaries can inform discontinuities in surface normal space, and some categories such as ceilings have very constrained normal directions. Other categories with no fixed shape or expected direction can be hard to improve.

We adapt from the SUNCG dataset [147] with physically-based rendering [148], which provides images, semantic segmentation, and surface normal ground truth. We use NYUdv2 [162] as the target domain, with additional surface normal estimated from depth by Ladicky et al. [163]. We only use the labeled portion of the dataset.

SUNCG is large, so we use a 90%-5%-5% split for train, validation, and test. We use NYUdv2's official split. Normal estimation loss and metrics are the same as before, and semantic segmentation is trained using cross-entropy.

4.4.3 Compared Methods

Since we address the domain adaptation problem, we compare to unsupervised adaptation methods that applies either a multi-level version of Ganin et al. [65] or state-of-the-art Tsai et al. [66], either on the single task model (DA), or on our method for further improvement (HeadFreeze +DA). Adversarial training is brittle and not all configurations work too well. We implement our own version and perform hyperparameter tuning, and omit some of the underperforming combinations. We also compare to an oracle method that uses both tasks labels on both domains, including the target domain main task. This gauges how far

					Faces: SfSsyn→FaceWH					Indoor: SUNCG→NYUdv2				
	$y_{\mathcal{S}m}$	$y_{\mathcal{S}a}$	$y_{\mathcal{T}m}$	$y_{\mathcal{T}a}$	$< 11.25°$	$< 30°$	RMSE	Mean	Median	$< 11.25°$	$< 30°$	RMSE	Mean	Median
Baseline	✓				0.424	0.929	17.8	14.8	12.8	0.298	0.683	33.5	25.8	18.8
Baseline+DA	✓				0.456	0.937	17.2	14.2	12.1	**0.316**	0.703	33.3	25.2	**17.6**
HeadFreeze (ours)	✓	✓		✓	**0.519**	**0.954**	**15.8**	**12.9**	**10.9**	0.301	0.708	**31.8**	24.6	18.0
HeadFreeze (ours)+DA	✓	✓		✓	0.455	0.935	17.2	14.2	12.1	**0.316**	**0.715**	32.0	**24.4**	**17.4**
Oracle	✓	✓	✓	✓	0.907	0.995	7.8	6.2	5.2	0.340	0.734	30.4	23.1	16.5
SfSNet [29]	–	–	–	–	0.495*	0.965	15.2	12.9*	11.3*					

Table 4.1: Comparison in our two experimental settings. Unsupervised domain adaptation with Tsai et al. [66] is shown for indoor scenes, and with Ganin et al. [65] shown for faces, whereas the other combinations underperform. Our HeadFreeze method is comparable to surface normal estimated from SfSNet, without the use of a lighting model. HeadFreeze +DA performs closest to oracle in indoor scene, but domain adaptation methods fail to improve HeadFreeze for faces. Statistical significance computed from 3 runs. (*) denotes a method with domain knowledge performs *equal to or worse* than our best performing method.

					Faces: SfSsyn→FaceWH					Indoor: SUNCG→NYUdv2				
	$y_{\mathcal{S}m}$	$y_{\mathcal{S}a}$	$y_{\mathcal{T}m}$	$y_{\mathcal{T}a}$	$< 11.25°$	$< 30°$	RMSE	Mean	Median	$< 11.25°$	$< 30°$	RMSE	Mean	Median
Baseline	✓				0.424	0.929	17.8	14.8	12.8	0.298	0.683	33.5	25.8	18.8
MTL (+src anchor)	✓	✓			0.409	0.935	17.7	14.9	13.1	0.280	0.666	34.1	26.6	19.8
MTL (+tgt anchor)	✓			✓	0.162	0.791	24.3	21.8	20.4	0.260	0.662	32.9	26.2	20.6
MTL (+both anchor)	✓	✓		✓	0.492	**0.953**	16.0	13.3	11.4	0.275	0.675	32.4	25.7	19.8
HeadFreeze (ours)	✓	✓		✓	**0.519**	**0.954**	**15.8**	**12.9**	**10.9**	**0.301**	**0.708**	**31.8**	**24.6**	**18.0**

Table 4.2: Ablation studies. See Figure 4.2 for each method's formulation. Other MTL baselines underperform while MTL (+both anchor) outperforms, indicating the importance of shared anchor tasks. Our HeadFreeze technique further boosts MTL (+both anchor) performance. Statistical significance computed from 3 runs.

each method is from fully successful adaptation.

For facial surface normal, we compare to a recent and popular intrinsic decomposition method SfSNet [29], which produces surface normal based on extra domain knowledge (lighting model for unsupervised learning). We use their released model trained on synthetic data and on unsupervised CelebA [164], a much larger dataset. This comparison only serves to prove that our method is effective instead of being a controlled experiment, since neither our network structure or external knowledge is similar.

For ablation studies, we compare to baselines shown in Fig. 4.2: single task baseline, multitask with only one domain (MTL (+src anchor)), multitask with source main task and target anchor task (MTL (+tgt anchor)) as used in prior work such as Liu et al. [165], and MTL (+both anchor).

4.4.4 Implementation Details

We use a ResNet50 [131] with FPN [166] for our network backbone, with the ResNet pre-trained on ImageNet [118]. We use the variant with 3 upsampling layers with skip connection, and used a deconvolution layer as the output layer for both tasks, making the output 50% of the input resolution. For HeadFreeze, we freeze the layers after the second upsampling layer, including any skip connection weights. Some tasks require additional non-spatial outputs. A common practice of 3D keypoint estimation [160] is to output a heatmap for projected 2D positions, and a vector for 3D depth. We add a fully-connected branch of 2 layers with 256 hidden units after the global average pooling over the second upsampling layer's output. Batches of the same size are sampled from each domain for each iteration. We choose λ so losses from different domains and tasks have similar magnitudes.

The discriminator for the adversarial loss is based on DCGAN [167]. However, adversarial training is very delicate, and hyperparameters have to be carefully tuned to avoid artifacts that devastate performance. The discriminator is trained jointly with the main network. For stability, we make the following adjustments. We use a 2-layer discriminator with a batchnorm between layers for facial images, and a 5-layer discriminator for indoor scenes. We use a common training strategy of adding the adversarial loss after the training has nearly converged. The DCGAN is trained using a PatchGAN loss that reduces overfitting [64], and noise is added to the input, activation, and real-fake ground truth, as recommended by Salimans et al. [141]. Stochastic gradient descent is used for the adversarial network, while the main network uses Adam [168] with a lower learning rate. The surface normals are normalized to a magnitude of 1 before input to the discriminator. For faces, regions outside the facial area for both features and outputs are masked to zero. New parameters are initialized with He et al. [169].

Facial images are resized to 256×256 for input. Indoor scene images are randomly cropped to 256×256 during training, and resized to 50% resolution at test time. For data augmentation, images are subject to random color shift, but not transform or flipping due to the nature of surface normal estimation. All evaluations are against the original resolution ground truth. Due to memory limit, we subsample 12.5% of all pixels when computing the median metric for SfSsyn and SUNCG.

Hyperparameter tuning is hard in TADA, just like in any unsupervised domain adaptation, due to the lack of target domain main task ground truth in validation. Although by evaluating against available ground truth we can tune most hyperparameters (e.g. stop criteria, learning rate, layers to freeze), some parameters critical to target main task (e.g. discriminator network complexity, its learning rate and loss weights) may barely cause any

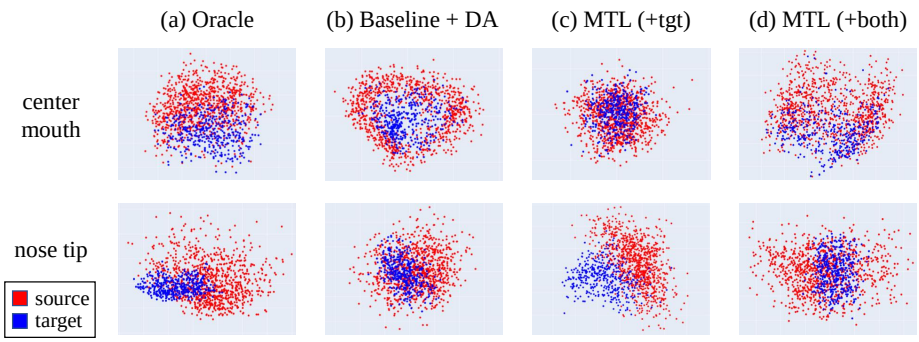


Figure 4.3: PCA visualization of the subtle differences between methods' feature space at different facial keypoint locations. (a) Oracle does not have fully overlapping domain due to systematic distribution differences. (b) Forcing domains' distributions to be similar can deviate the features from the oracle and hurt performance (top row). (c) Training MTL with one task per domain may encourage using separate feature space regions for different domains (bottom row). (d) MTL (+both anchor) produces feature distributions slightly more visually similar to the oracle. Disclaimer: the baseline's visualization (not shown) is also similar to the oracle, so this cannot indicate higher performance. Other facial locations may not exhibit observed behaviors as clearly. Best viewed in color.

change. We empirically find that the discriminator accuracy being very frequently lower than 55% and good qualitative results (absence of artifacts) are good indicators of successful adaptation, and tune the parameters accordingly.

4.5 RESULTS

Table 4.1, 4.2 shows our results and ablation studies.

Facial images. On SfSsyn to FaceWH adaptation, HeadFreeze outperforms the non-adaptation baseline. Adding domain adaptation [65] does improve baseline results, but still underperforms our HeadFreeze method. HeadFreeze is comparable to the popular SfSNet [29], which underperforms on Median and 11.25° but outperforms on RMSE and 30°. However, all methods are still quite far from the oracle method that uses target domain main task annotations.

Perhaps a very surprising observation is that the unsupervised domain adaptation methods added to HeadFreeze would *hurt* performance instead of improving them. In fact, HeadFreeze *without adaptation* is the best method apart from the oracle (and SfSNet). The adaptation puts HeadFreeze at the same level with DA [65], eliminating any advantage brought by the anchor task. We have vigorously tuned the adversarial loss hyperparameters, yet still cannot find a configuration that would not hurt performance. In comparison,

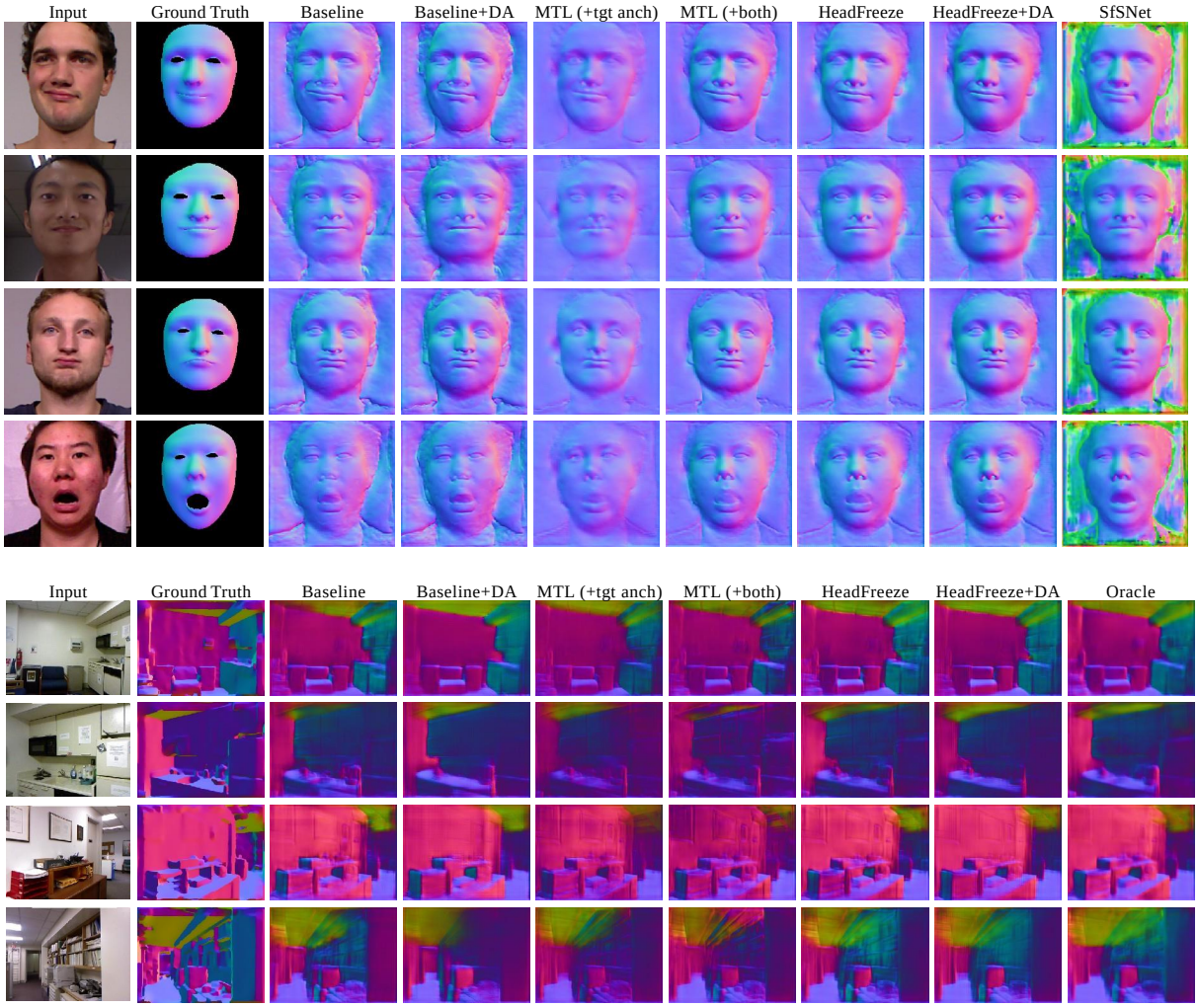


Figure 4.4: Qualitative results for compared methods. For domain adaptation, Ganin et al. [65] is shown for facial images (top), and Tsai et al. [66] is shown for indoor scenes (bottom). Best viewed in color.

HeadFreeze (and even MTL (+both anchor) in Table 4.2) work naturally. We analyze the reason for our robustness in Section 4.5.1.

In the ablation study, the baseline and MTL with anchor on either one domain all underperform. Two observations are interesting: (1) MTL (+src anchor) does not perform very differently from the baseline on the target domain, indicating that the effect of multi-task learning is limited here. (2) MTL (+tgt anchor) vastly underperforms the baseline when trained with one task per domain. We hypothesize that despite the network being trained on the target $\mathcal{T}$, the task performed on $\mathcal{T}$ is too different, which *encourages* the network to learn very different features for the tasks, harming adaptation. MTL (+both anchor) outperforms other MTL methods, indicating the importance of the anchor task being trained

	$< 11.25°$	$< 30°$	RMSE	Mean	Median
Baseline	0.418	0.913	18.6	15.3	13.0
Baseline+DA [65]	0.495	0.944	16.6	13.5	11.3
HeadFreeze	0.550	0.958	15.2	12.4	10.4
HeadFreeze +DA [65]	**0.573**	**0.963**	**14.7**	**11.9**	**10.0**

Table 4.3: Facial normal estimation, with SfSsyn-frontal as the source domain, which has head pose distribution similar to FaceWH. In this experiment, domain adaptation [65] always helps performance, indicating that systematic dataset difference is the reason distribution matching adaptation fails, which HeadFreeze is robust to.

on both domains, affirming our hypothesis. HeadFreeze further improves all criteria by a margin, implying that the cross-task guidance learned in the source domain can be helpful for the target domain as well.

Indoor scenes. We still see both HeadFreeze and domain adaptation [66] improve over the non-adaptation baseline, but it is inconclusive whether HeadFreeze outperforms domain adaptation [66]. But we observe that HeadFreeze +DA further improves the adaptation-only method, closing much of the gap between baseline and the oracle.

In Table 4.2's ablation study, all MTL variations suffer from negative transfer, i.e. main task performance degrades as the second task is jointly learned. We still observe that MTL (+both anchor) outperforms other MTL variants, indicating that it has an adaptation effect that other variants do not possess, despite the negative transfer. We also observe that HeadFreeze makes a larger improvement on MTL (+both anchor) than in the facial experiments.

4.5.1 Further Analysis

Failure of adaptation and face label distribution. We analyze why the compared domain adaptation methods fail to improve HeadFreeze in the SfSsyn-FaceWH experiment. After trials and errors, we found that the difference of head pose distributions between domains may be a major contributor. We generated a second version of the SfSsyn dataset with only frontal faces ("SfSsyn-front"), with rotation distribution closely following the estimated poses from the target dataset. We evaluate the domain adaptation methods with SfSsyn-front as the source domain in Table 4.3 instead, with all methods using the same hyperparameter.

The trends and conclusions are exactly the same, except that unsupervised domain adaptation always helps, making our HeadFreeze + DA the top method. This experiment

indicates that the distributional difference is indeed why adaptations [65, 66] fail. We conclude that while these prior works are effective, they would *hurt* performance when domain ground truths are differently distributed, whereas our methods are more robust to such differences. While these differences may sometimes be easily eliminated in data synthesis procedures, other times they may be expensive to eliminate, or difficult to pinpoint.

Impact on feature space. To better understand the impact of different methods on the feature distribution, we visualize their feature space for source and target domain. Since features at different spatial locations may encode information differently, we extract the feature at separate facial keypoint locations in the facial experiment. For each location (e.g. nose tip), we perform PCA and obtain the top two components, and visualize them in Fig. 4.3. Please refer to its caption for observations. This experiment resonates with our hypothesis that training MTL (+tgt anchor) with one task per domain would map source and target to different feature space regions, and that blindly matching feature distribution may be suboptimal.

Qualitative results are shown in Figure 4.4. For faces, the synthetic dataset has less facial expressions than FaceWarehouse, so baselines struggle with e.g. open mouths. Unsupervised adaptation [65] tends to erroneously force the cheeks and nose normals to the side to force the output look like side-facing faces locally. The ground truth is not extremely faithful to the image due to being fitted on RGBD scans, and both our HeadFreeze method and SfSNet [29] capture local details better than the ground truth, although SfSNet performs better with open mouths due to their usage of a lighting model on unlabeled real faces.

For indoor scenes, HeadFreeze improves the performance for shelves, cabinets, and ceilings more effectively than the facial datasets, possibly due to their semantic labels providing much information for their surface normal.

4.6 SUMMARY

In this work, we propose a strategy to extend prior Task-assisted Domain Adaptation methods by eliminating the need for task-specific relationship formulations. We use spatial information of a free or already available shared anchor task to align both features between domains and spatial prediction and context between tasks, and propose HeadFreeze to further leverage the cross-task guidance to improve main task on target domain. We show effectiveness and robustness of using anchor tasks against multitask baselines, and HeadFreeze against conventional domain adaptation methods.

We have demonstrated our method on two scenarios, facial images surface normals and indoor scene surface normals. But our method would be useful to other adaptation scenarios

where anchor tasks give hints about the underlying structure of an image. For example, using depth as the anchor for adapting road scene segmentation or detection in different weather conditions, using room layout and corner detection as the anchor for adapting from synthetic indoor scenes, and using video camera movement or optical flow prediction in adapting other forms of video tracking from synthetic videos. See Section 6.1.3 for other future work directions.

CHAPTER 5: IMPROVING CONFIDENCE ESTIMATES FOR UNFAMILIAR EXAMPLES

In research, the i.i.d. assumption, that train and test sets are sampled from the same distribution, is convenient and easily satisfied. In practice, the training and test images often come from different distributions, as developers often have access to a less diverse set of images than future samples observed by the deployed system. For example, the face images gathered by a company's employees may not have the racial or age diversity of the world's population. Scholars that study the impact of AI on society consider differently distributed samples to be a major risk [8]: "This is one form of epistemic uncertainty that is quite relevant to safety because training on a dataset from a different distribution can cause much harm." Indeed, high profile failures, such as a person being labeled as a gorilla [170] or a car driving through a tractor trailer [171], are due at least in part to failure to provide good confidence estimates for unfamiliar data.

In this chapter, our goal is to compare and evaluate several methods for improving confidence estimates for unfamiliar and familiar samples for the same task. We consider *familiar samples* to be drawn from the same distribution as the training, as is typically done when creating training and test sets for research. We term *unfamiliar samples* as drawn from a different but still applicable distribution under covariate shift. For example, for cat vs. dog classification, an image of a dog from a breed seen during training is familiar, while an image from a breed not seen during training is unfamiliar. The concept of cat vs. dog does not change from familiar to unfamiliar. We are not concerned with non-applicable "out of domain" images such as an image of a pizza for cat vs. dog classification, or changes in the concepts or conditional label distributions such as a dog having a cat ground truth label in the unfamiliar domain.

We propose familiar/unfamiliar splits for four image classification datasets and evaluate by measuring accuracy of predicted labels and confidences. One would expect that classifiers would be less accurate and less confident for unfamiliar samples. Our experiments confirm that deep network classifiers have lower prediction accuracy on unfamiliar samples but also show that wildly confident wrong predictions occur much more often, due to higher calibration error. A simple explanation is that classifiers minimize a loss based on $P(y|x)$, for label y and input features x, which is unregulated and unstable wherever $P(x) \sim 0$ in training. Empirical support for this explanation comes from Novak et al. [172] who show that neural networks are more robust to perturbations of inputs near the manifold of the training data. We examine the effectiveness of calibration (we use temperature scaling [173]) for improving confidence estimates and the potential for further improvement using uncertainty-sensitive

Figure 5.1: Deep networks often make highly confident mistakes when samples are drawn from outside the distribution observed during training. Examples shown have ages (top), breeds (middle), or species (bottom) that are not observed during training and are misclassified by a deep network model with high confidence. This chapter investigates the problem of overconfidence for unfamiliar samples and evaluates several potential methods for improving reliability of prediction confidences.

training [85], ensembles, and scaling based on novelty scores. Since calibrated ensembles perform best but are most computationally expensive, we also investigate distilling the ensemble from a mix of supervised and unsupervised data.

The key contributions of this chapter are: (1) highlight the problem of overconfident errors in practical settings where test data may be sampled differently than training; (2) propose a methodology to evaluate performance on unfamiliar and familiar samples; (3) demonstrate the importance of confidence calibration and compare several approaches to improve confidence predictions, including new ideas for incorporating novelty prediction and mixed supervision distillation.

5.1 RELATED WORK

Unreliability of prediction for unfamiliar samples have been recently observed by several works. Lakshminarayanan et al. [6] show that networks are unreliable when tested on semantically unrelated or out-of-domain samples, such as applying object classification to images of digits. They also show that a using the Brier score [174] (squared error of 1 minus confidence in true label) as a loss and training an ensemble of classifier improves

confidence calibration and reduces overconfident errors on out-of-domain samples. Ovadia et al. [7], in independent work concurrent to ours, also find that ensembles are most effective for skewed and out-of-domain samples, evaluating with Brier score, negative log likelihood of predictions, and expected calibration error (ECE). Our inclusion of Brier score and ECE is inspired by these methods. Our paper differs from these in the consideration of natural (not artificially distorted) samples from unfamiliar but semantically valid distributions, which is a common practical scenario when, for example, developers and users have access to different data. Roos et al. [175] distinguish between i.i.d. generalization error and off-training-set error and provide bounds based on repetition of input features. Extending their analysis to high dimensional continuous features is a worthwhile area of further study.

This chapter is also related to several lines of work outlined in Section 2.2, and we compare calibration, novelty detection, ensembles, distillation, and modeling uncertainty (Bayesian methods).

5.2 PROBLEM SETUP AND METHODS

In many commercial settings, the developers of an algorithm have access to data that may be limited by geography, demographics, or challenges of sampling in diverse environments, while the intended users, in aggregate, have much broader access. For example, developers of a face recognition algorithm may undersample children, elderly, or Inuits, due to their own demographics. Someone training a plant recognition algorithm may have difficulty collecting samples of species not locally native. A recognizer of construction equipment may be applied to vehicle models that came out after release of the classification model. To study and improve the robustness of classifiers in these settings we explore:

- How to organize data to simulate the familiar and unfamiliar test sets (Sec. 5.2.1)
- How to evaluate the quality of predictions (Sec. 5.2.2)
- What methods are good candidates to improve prediction quality on unfamiliar samples (Sec. 5.2.3)

5.2.1 Datasets and Familiar/Unfamiliar Split

We choose four classification tasks for evaluation, detailed below and shown in Figure 5.2. For each of the first three tasks, a dataset is first split into "familiar" and "unfamiliar" subsets according to an attribute or subcategory, simulating the case of training data not containing the full diversity of potential inputs. In the fourth task (object presence classification),

Figure 5.2: Familiar and unfamiliar samples from each dataset. To study how classifier performance varies with novelty, we create splits of unfamiliar and familiar samples that are task-relevant, where the split is defined by age, breed, species, or sampling date. The first three represent cases where the training distribution does not fully cover the test cases. The last represents a case of the minimal novelty achievable without independently sampling from the same image set.

two similar datasets are used for the same object categories, simulating similar sources but sampled at different times. The "familiar" samples $(\mathbf{x}_{\mathcal{F}}, y_{\mathcal{F}}) \sim \mathcal{F}$ are further split into training F_{tr}, validation F_{vl}, and test F_{ts} sets, while the "unfamiliar" samples $(\mathbf{x}_{\mathcal{U}}, y_{\mathcal{U}}) \sim \mathcal{U}$ are used only for testing. The inputs $\mathbf{x}_{\mathcal{F}}$ and $\mathbf{x}_{\mathcal{U}}$ may occupy different portions of the feature space, with a lot, little, or no overlap, but where they do overlap we assume $P_{\mathcal{F}}(y|\mathbf{x}) = P_{\mathcal{U}}(y|\mathbf{x})$. No sample from $\mathcal{U}$ is ever used in pre-training, training, or validation (parameter selection). In some cases, we use a dataset's standard validation set for testing (and not parameter tuning) so that we can compute additional metrics, as ground truth is not publicly available for some test sets.

Gender recognition: The extended Labeled Faces in the Wild (LFW+) dataset [176] with 15,699 faces is used. Samples are split into familiar $\mathcal{F}$ and unfamiliar $\mathcal{U}$ based on age annotations provided by Han et al. [177], with familiar ages 18-59 years and unfamiliar ages outside that range. The dataset comes with five preset folds; we use the first two for training, the third fold for validation, and the last two for testing.

Cat vs. dog recognition: Using the Pets dataset [178], the first 20 dog breeds and first 9 cat breeds are familiar, and the other 5 dog and 3 cat breeds are unfamiliar. The standard train/test splits are used (with training samples from $\mathcal{U}$ excluded).

Animal categorization: Four animal superclasses (mammals, birds, herptiles, and fishes) are derived from ImageNet [118], and different subclasses are used for familiar and unfamiliar sets. After sorting object classes within each superclass by their indices, the first half of classes are familiar $\mathcal{F}$, and the second half are unfamiliar $\mathcal{U}$. The data is also subsampled, so there are 800 training and 200 validation examples drawn from the ImageNet training set per superclass, and 400 examples drawn from the ImageNet validation set for each of the unfamiliar and familiar test sets.

Object presence classification: The PASCAL VOC 2012 dataset [122] is used as familiar, with the similar 20 classes in MS COCO [179] used as unfamiliar. `tvmonitor` is mapped to `tv`. Test samples are drawn from the VOC PASCAL and MS COCO validation sets. The familiar and unfamiliar samples in this task are more similar to each other since they vary, not by attribute or subclass, but by when and by whom the images were collected.

5.2.2 Evaluation Metrics

We use several error metrics to assess the quality of classifier predictions. We denote $P_m(y_i|\mathbf{x}_i)$ as the assigned confidence in the correct label for the i^{th} of N samples by a model m. In all metrics, lower is better.

NLL: Negative log likelihood (NLL) $\frac{1}{N}\sum_i \log P_m(y_i|\mathbf{x}_i)$ is a natural measure of prediction quality and commonly used as a loss for training classification models (often called "cross-entropy"), as it corresponds to the joint probability of predictions on independently drawn samples. The main drawback is that NLL is unbounded as confidence in the correct class approaches 0. To help remedy this, we clip the softmax probability estimates to $[0.001, 0.999]$ for all models before computing NLL.

Brier: The Brier score [174] measures the root mean squared difference between one and the confidence in the correct label: $\left(\frac{1}{N}\sum_i (1 - P_m(y_i|\mathbf{x}_i))^2\right)^{1/2}$. Similar to NLL, Brier is smallest when the correct label is predicted with high confidence, but the penalty for highly confident errors is bounded at 1, avoiding too much emphasis on a few large errors. We use RMS (root mean squared) instead of mean squared, as in the original, because we find it easier to interpret, and we call it "Brier error", since it should be minimized.

Label Error: Label error is measured as the percent of incorrect most likely labels, or 1 minus average precision. We use percent incorrect for all tasks except object presence classification, for which we use mean average precision, in accordance with community norms

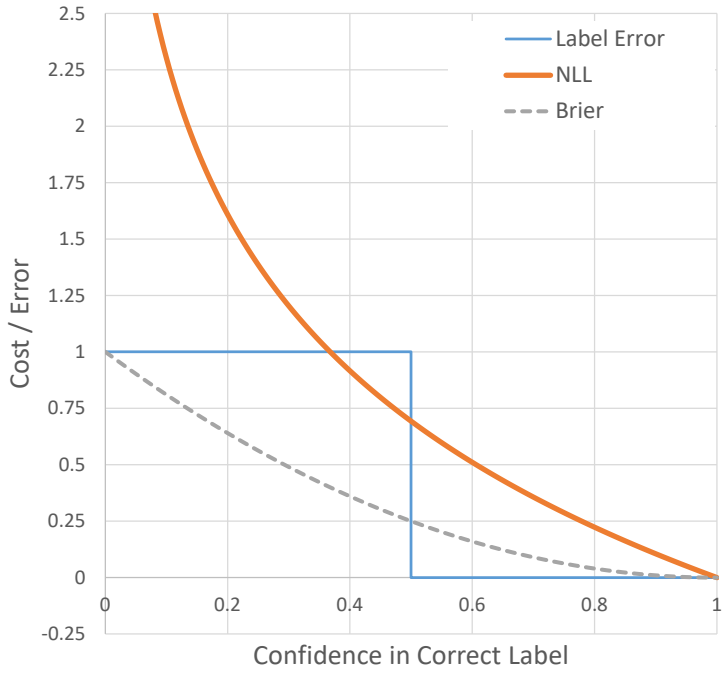


Figure 5.3: **Prediction quality metrics**: plot of error vs. confidence in correct label for 0-1 classification. NLL (negative log likelihood) strongly penalizes confidently wrong predictions, while Brier error penalties are constrained. Label error does not assess confidence beyond which label is most likely.

for reporting performance on these tasks.

ECE: Expected calibration error (ECE) measures whether the classifier "knows what it knows". Following the notation of [173], ECE is computed as $\sum_{j=1}^{J} \frac{|B_j|}{N} |\text{acc}(B_j) - \text{conf}(B_j)|$ where B_j is a set of predictions binned by confidence quantile, $\text{acc}(B_j)$ is the average accuracy of the B_j, and $\text{conf}(B_j)$ is the average confidence in the most likely label. We use 10 quantiles for binning.

E99: E99 is the error rate among the subset of samples that have at least 99% confidence in any label. If the classifier is well-calibrated, E99 should be less than 1%. We created E99 to directly measure a model's tendency to generate highly confident errors.

5.2.3 Compared Methods

Deep network classifiers are often overconfident, even on familiar samples [173]. On unfamiliar samples, the predictions are less accurate and even more overconfident, as our experiments show. We consider several tools to improve predictions: calibration, novelty detection, ensembles, and loss functions that account for uncertainty. Some methods provide better confidence calibration, while others (e.g. ensembles and Bayesian models) can also provide

more confidently accurate predictions.

T-scaling: Calibration aims to improve confidence estimates so that a classifier's expected accuracy matches its confidence. Among these, we use the temperature-scaling method described in Guo et al. [173]. At test time, all softmax logits are divided by temperature T. With $T > 1$, prediction confidence is decreased. T is a single parameter set to minimize NLL on the validation set. We then use this T on a network retrained on both training and validation sets.

Novelty-weighted scaling: We also consider novelty-weighted scaling, with the intuition that confidence should be lower for novel (i.e. unfamiliar) samples than for those well represented in training. We use the ODIN [95] model-free novelty detector. Since the novelty scores $novelty(\mathbf{x})$ often have a small range, we normalize them by linearly scaling the 5^{th} and 95^{th} percentile on training data to be 0 and 1 and clipping values outside $[0, 1]$. We then modify temperature scaling to set $T(\mathbf{x}) = T_0 + T_1 \cdot novelty(\mathbf{x})$, with T_0 and T_1 set by grid search on the validation set, so that temperature depends on novelty.

Ensemble methods consider both model parameter and data uncertainty by averaging over predictions. In areas of the feature space that are not well represented by training data, members of the ensemble may vary in their predictions, reducing confidence appropriately. In our experiments, members of the ensemble are trained with all training samples and differ due to varying initialization and stochastic optimization. We found this simple averaging approach to outperform bagging and bootstrapping. In prediction, the member confidences in a label y_i are averaged to yield the ensemble confidence: $P_m(y_i|\mathbf{x}_i) = \frac{1}{M}\sum_j^M P_{m_j}(y_i|\mathbf{x}_i)$, where M is the number of ensembles. $M = 10$ in our experiments.

Distillation: Our experiments show the ensemble is highly effective, but it is also M times more expensive for inference. We, thus, consider whether we can retain most of the benefit of the ensemble at lower compute cost using distillation [101]. After training the ensemble, the distilled model is trained by minimizing a weighted distillation loss (minimizing temperature-scaled cross-entropy of the ensemble's soft predictions with the distilled model's predictions) and a classification loss:

$$\mathcal{L} = \frac{1}{|F_{tr}|}\sum_{(\mathbf{x}_\mathcal{F}, y_\mathcal{F})\in F_{tr}} \big(\lambda_{cls}\mathcal{L}_{cls}(y_\mathcal{F}, f_{dis}(\mathbf{x}_\mathcal{F})) + \mathcal{L}_{dis}(f_{ens}(\mathbf{x}_\mathcal{F}), f_{dis}(\mathbf{x}_\mathcal{F}))\big) \tag{5.1}$$

where $\mathcal{L}_{cls}$ is the classification loss over the distilled model's soft predictions $f_{dis}(\mathbf{x}_\mathcal{F})$, $\mathcal{L}_{dis}$ is the distillation loss over the soft predictions of the distilled model and ensemble $f_{dis}(\mathbf{x}_\mathcal{F})$, and λ_{cls} is a weighting to balance classification and distillation losses ($\lambda_{cls} = 0.5$,

as recommended in [101]).

G-distillation: Under the standard distillation, the distilled model is guided to make similar predictions to the ensemble for the familiar distribution $\mathcal{F}$, but its predictions are still unconstrained for unfamiliar samples, potentially losing the benefit of the ensemble's averaging for samples from $\mathcal{U}$. Therefore, we propose G-distillation, a generalized distillation where the distillation loss is also computed over samples from an unsupervised distribution $\mathcal{G}$. In our experiments, we choose $\mathcal{G}$ to be related to the task, but make sure there is no overlap between specific examples in $\mathcal{G}$ and F_{ts} or $\mathcal{U}$. We use the following unsupervised datasets for $\mathcal{G}$ in our experiments: Gender, CelebA [164]; Broad animal, COCO [179]; Cat-dog, ILSVRC12 [118]; and Object presence, Places365-standard [121]. The images from $\mathcal{G}$ are disjoint with the datasets used to draw $\mathcal{F}$ and $\mathcal{U}$ for each respective task.

Bayesian model: Finally, we consider the Bayesian method of Kendall et al. [85], which accounts for uncertainty in model parameters (epistemic) and observations (aleatoric). To account for model parameter uncertainty, multiple predictions are made with Monte Carlo Dropout, and predictions are averaged. In this way, dropout is used to simulate an ensemble within a single network. In our implementation, we apply dropout to the second-to-last network layer with a rate of 0.2. Observation uncertainty is modeled with a training loss that includes a prediction of logit variance. The logits can then be sampled based on both dropout and logit variance, and samples are averaged to produce the final confidence. See [85] for details.

5.3 EXPERIMENTS

When comparing these methods, we aim to answer the following experimental questions:

- Do T-scaling calibration parameters learned from $\mathcal{F}$ also improve confidence estimates on $\mathcal{U}$?
- Does novelty-weighted scaling outperform the data agnostic T-scaling?
- Do ensembles learned on $\mathcal{F}$ also improve predictions on $\mathcal{U}$?
- Is distillation able to preserve ensemble performance on $\mathcal{F}$ and $\mathcal{U}$?
- Does adding the unsupervised set for distillation in G-distillation lead to better preservation?
- Does the Bayesian model that is specifically designed to manage model and observational uncertainty outperform more general alternatives?

(Spoiler alert: answers in order are yes, no, yes, no, yes, partially.)

5.3.1 Training and Testing Details

Training: For all experiments we use PyTorch with a ResNet-18 architecture and Adam gradient descent optimization with a momentum of 0.9. We initialize the final layer of our pre-trained network using Glorot initialization [180]. We perform hyper-parameter tuning for the learning rate and the number of epochs using a manual search on a validation split of the training data. When the performance on validation plateaus, we reduce the learning rate by a factor of 10 and run 1/3 as many additional epochs as completed up to that point. After fitting hyperparameters on the validation data, the models are retrained using both train and val sets. For data augmentation, we use a random crop and mirroring similar to Inception [181]. Places365-standard [121] dataset is used to pretrain the network, and the network is fine tuned separately for each task. When training G-distillation, we sample the image from G to be roughly $\frac{1}{4}$ the size of F_{tr}. We also verified that using a different architecture (DenseNet161 [132]) yields the same experimental conclusions.

Testing: At test time we evaluate on the center crop of the image. Due to the relatively high variance of NLL on U_{ts}, we run our experiments 10 times to ensure statistical significance (unpaired two-tail t-test with p=0.95 on model performance), but we run the ensemble method only once (variance estimated using ensemble member performance variance). Our 10 runs of the distillation methods use the same ensemble run.

5.3.2 Results

Our main table of results is shown in Table 5.1. The **baseline** is a single uncalibrated ResNet-18 network. The others correspond to the methods described in Sec. 5.2. For the baseline, we show the absolute error, and for the other methods, we show the percent reduction in error compared to the baseline (e.g. a drop from 0.10 to 0.09 is a 10% reduction) to facilitate comparison. The complete table with absolute error is included in the supplemental material. All methods except baseline use calibration.

Familiar vs. Unfamiliar Performance: Looking at baseline performance in Table 5.1, we see much higher error rates for unfamiliar samples, compared to familiar, for all tasks. The label error and calibration error are both higher, leading to much higher NLL and Brier error. *This means the baseline classifier is less accurate and has poor ability to detect its own inaccuracy on unfamiliar samples* — it does not know what it does not know. For example, in gender recognition, the label error increases from 2.8% for unfamiliar to 14.7%; the calibration error ECE increases from 0.013 to 0.109; and the NLL increases from 0.083 to 0.542. Figure 5.4 underscores the prevalence of confident errors, which are several times

	NLL		Brier		Label Error		ECE	
Gender	**fam.**	**unf.**	**fam.**	**unf.**	**fam.**	**unf.**	**fam.**	**unf.**
Baseline	0.083	0.542	0.147	0.352	0.028	**0.147**	0.013	0.109
T-scaling	12%	26%	2%	**4%**	0%	**0%**	**73%**	20%
Ensemble	**24%**	33%	**10%**	**6%**	**22%**	**0%**	36%	**29%**
Distill	8%	33%	3%	4%	3%	-7%	41%	21%
G-distill	13%	**38%**	5%	**6%**	9%	-5%	31%	**31%**
Bayesian	17%	26%	5%	4%	6%	**0%**	**77%**	19%
Cat vs. Dog								
Baseline	0.053	0.423	0.112	0.290	0.016	0.095	0.010	0.078
T-scaling	23%	30%	4%	5%	0%	0%	64%	23%
Ensemble	**40%**	**46%**	**17%**	**12%**	**22%**	**8%**	**79%**	**46%**
Distill	-13%	22%	-9%	1%	-18%	-4%	55%	26%
G-distill	-18%	27%	-14%	1%	-33%	-8%	41%	31%
Bayesian	17%	26%	3%	5%	0%	3%	42%	21%
Animals								
Baseline	0.326	1.128	0.199	0.341	0.104	0.291	0.048	0.187
T-scaling	13%	23%	3%	5%	0%	0%	**75%**	37%
Ensemble	**22%**	**32%**	**9%**	**8%**	**11%**	**6%**	50%	**57%**
Distill	7%	24%	1%	5%	-1%	0%	66%	45%
G-distill	14%	26%	5%	7%	7%	2%	56%	49%
Bayesian	16%	24%	5%	5%	4%	1%	**74%**	39%
Objects								
Baseline	0.086	0.128	0.154	0.186	0.195	0.455	0.005	0.010
T-scaling	0%	0%	0%	0%	0%	0%	2%	2%
Ensemble	**4%**	4%	**2%**	2%	**6%**	**3%**	**3%**	7%
Distill	-1%	**5%**	0%	**2%**	1%	0%	-31%	**10%**
G-distill	-2%	5%	-1%	2%	-2%	-1%	-41%	7%
Bayesian	0%	0%	0%	0%	0%	0%	3%	1%

Table 5.1: Performance of baseline (single model) for several metrics and percent reduction in error for other methods. All methods except baseline use T-scaling calibration. "T-scaling" is a single calibrated model.

more common for unfamiliar samples than familiar.

The differences between unfamiliar and familiar for object presence classification are substantial but smaller than other tasks, as expected, since VOC (familiar) and COCO (unfamiliar) images were both sampled from Flickr using similar methodologies [179]. The larger differences in mean AP (label error) may be due to lower frequency for a given object category in COCO.

Importance of calibration: Table 5.2 compares performance of the baseline and ensemble methods, both without and with T-scale calibration. Calibrated single models outperform uncalibrated models, and ensembles of calibrated models outperform ensembles of uncalibrated models. For example, in cat vs. dog recognition, the baseline NLL drops

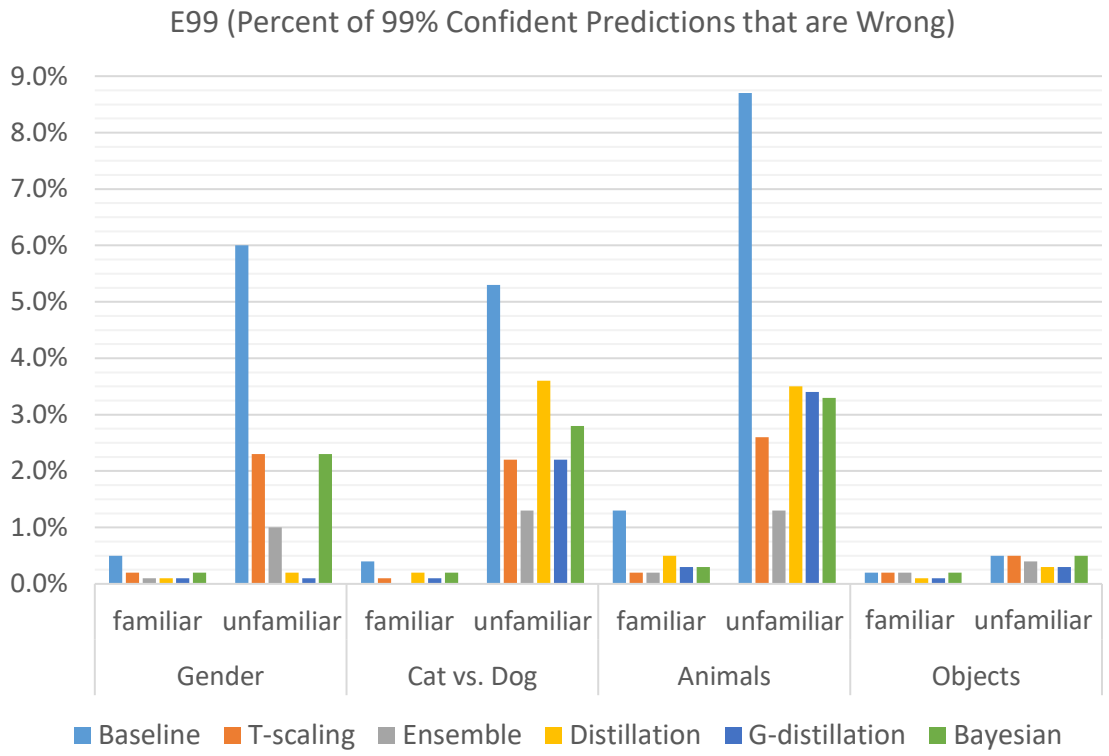


Figure 5.4: Classifiers are much more prone to confident errors when faced with unfamiliar samples. T-scaling calibration, among other methods, reduces the overconfidence, with ensembles of calibrated models providing consistent further improvement.

from 0.423 to 0.295, a 30% reduction; and the ensemble NLL drops from 0.286 to 0.229, a 20% reduction. Though not shown, a calibrated ensemble of uncalibrated models performs very similarly to an ensemble of calibrated models. For the object presence task, there is little effect of calibration because the classifier trained on the training samples was already well-calibrated for the familiar validation samples. We also found calibration to improve the Bayesian method [85]. Calibration has little effect for distillation and G-distillation, likely because distillation's fitting to soft labels makes it less confident. For those methods, we used calibration only when $T >= 1$, as setting $T < 1$ always made classifiers more over-confident. In Table 5.1, "T-scaling" refers to the T-scaled baseline, and T-scaling is used for all other non-baseline methods as well.

Given the benefits of T-scaling, we expected that novelty-weighted scaling, in which samples predicted to be unfamiliar have a greater temperature (reducing confidence more), would further improve results. However, we found the novelty weight T_1 was usually set to zero in validation, and, in any case, the novelty-weighted scaling performed similarly to T-scaling. The problem could be that the validation set does not have enough novelty to determine the

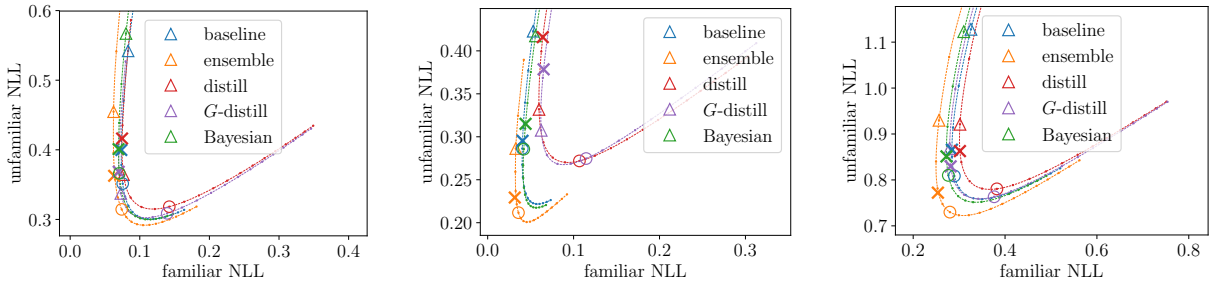
	NLL		Brier		ECE	
Gender	**fam.**	**unf.**	**fam.**	**unf.**	**fam.**	**unf.**
Single	0.083	0.542	0.148	0.352	0.013	0.109
Sin. T-scale	0.073	0.400	0.145	0.338	0.004	0.087
Ensemble	0.062	0.455	0.130	0.344	0.003	0.093
Ens. T-scale	0.063	0.363	0.130	0.333	0.009	0.077
Cat vs. Dog						
Single	0.053	0.423	0.110	0.290	0.010	0.078
Sin. T-scale	0.041	0.295	0.105	0.276	0.004	0.060
Ensemble	0.033	0.286	0.095	0.263	0.002	0.055
Ens. T-scale	0.032	0.229	0.095	0.255	0.002	0.042
Animals						
Single	0.326	1.128	0.200	0.341	0.048	0.187
Single T-scale	0.284	0.866	0.195	0.324	0.012	0.118
Ensemble	0.256	0.930	0.182	0.322	0.022	0.138
Ens. T-scale	0.254	0.772	0.182	0.311	0.024	0.080

Table 5.2: T-scaling calibration effectively reduces likelihood error (NLL, Brier) and calibration error (ECE) for many models across tasks for familiar and unfamiliar samples. Without calibration, using an ensemble reduces these errors, but an ensemble of calibrated models ("Ens. T-scale") performs best. Applying T-scaling to an ensemble of uncalibrated classifiers, and creating an ensemble of calibrated classifiers produces nearly identical results.

correct weights. If we "cheat" and use samples drawn from the unfamiliar distribution to set the two weights T_0 and T_1, the method performs quite well. For example, when tuning parameters on a mix of familiar and unfamiliar samples for gender recognition, novelty-weighted scaling performed best with 0.297 NLL compared to 0.328 for T-scaling and 0.313 for ensemble of calibrated classifiers that are tuned on the same data.

In Figure 5.5, we plot calibration curves of single networks, ensembles, distillation, G-distillation, and the Bayesian method with varying T. These curves allow us to peek at the best possible performance, if we were able to tune calibration parameters on unfamiliar and familiar test data. These curves allow a clearer view of which methods perform best. They also show that calibration on the familiar samples ('X' marks) leads to lower T values than is optimal for the unfamiliar samples. Generally, increasing T further would reduce likelihood error for unfamiliar samples without much adverse impact on likelihood error for familiar samples. On the object presence task (curve not shown), all models are well-calibrated (without T-scaling) for both unfamiliar and familiar categories.

Comparison of methods: Finally, considering Table 5.1, we see that the ensemble of T-scaled models dominates, consistently achieving the lowest label error, calibration error, NLL, and Brier error. The downside of the ensemble is higher training and inference com-



(a) Gender Recognition (b) Cats vs. Dogs (c) Animal Categorization

Figure 5.5: Familiar and unfamiliar NLL error while varying the T calibration parameter. Triangles mark the uncalibrated models; 'X' marks models calibrated on the validation set. Circles mark $T = 2$, with each rightward dot increasing by 0.25. Without calibration, classifiers are often overconfident even for familiar samples, so calibration reduces confidence to improve NLL for familiar and unfamiliar. Ensembles dominate the other methods, always achieving lower NLL for some T.

putational cost, 10x in our case since we test with an ensemble of 10 classifiers. Distillation and G-distillation offered hope of preserving some of the gains of ensembles without the cost, and we expected the performance of G-distillation at least to fall between T-scaling and the ensemble. However, while G-distillation, which uses unsupervised samples to better mimic ensemble behavior in the broader feature domain, slightly outperforms distillation, neither method consistently outperforms T-scaling — no pain, no gain.

The method of Kendall et al. [85], which we call "Bayesian", performs second best to the ensemble, with small reductions in label error and comparable calibration improvements to all methods except ensemble. The Bayesian method also requires generating multiple predictions via MC-Dropout at test time, so also incurs significant additional computational cost.

All methods reduce calibration and likelihood error. Reductions in NLL are larger than Brier (in relative terms), due to NLL's more aggressive penalty for larger errors, and the prevalence of overconfident errors on unfamiliar samples creates more opportunity.

We thank one of the paper reviewers for suggesting analysis of prediction entropy, which we include in Table 5.3. Prediction entropy measures the uncertainty of classification and is maximized if the classifier outputs uniform probabilities for each class. NLL, equivalent to cross-entropy when using hard labels, measures the uncertainty in the correct label. When entropy is lower than cross-entropy (i.e. more confident than confidently correct), the classifier is overconfident. The results show that calibration eliminates overconfidence for familiar samples but calibrated ensembles further reduce overconfidence on unfamiliar samples by increasing prediction uncertainty and improving accuracy.

	NLL		Entropy		NLL-Ent.	
Gender	**fam.**	**unf.**	**fam.**	**novel**	**fam.**	**unf.**
Single	0.083	0.542	0.036	0.089	0.047	0.453
Sin. T-scale	0.073	0.400	0.069	0.139	0.005	0.261
Ens. T-scale	0.063	0.363	0.079	0.158	-0.016	0.205
Cat vs. Dog						
Single	0.053	0.423	0.014	0.043	0.039	0.380
Sin. T-scale	0.041	0.295	0.033	0.090	0.007	0.206
Ens. T-scale	0.032	0.229	0.039	0.113	-0.007	0.116
Animals						
Single	0.326	1.128	0.146	0.263	0.180	0.864
Sin. T-scale	0.284	0.866	0.282	0.451	0.002	0.415
Ens. T-scale	0.254	0.772	0.311	0.504	-0.057	0.269

Table 5.3: When prediction entropy is lower than NLL (cross-entropy), the classifier is overconfident, i.e. more confident than confidently correct. We see, e.g., that single-model calibration eliminates overconfidence for familiar examples, but the calibrated ensemble achieves much further reduction in overconfidence for unfamiliar examples by increasing uncertainty and improving accuracy (lower NLL and label error).

Results on the gender task using DenseNet-161 is shown in Table 5.4. In this case Dropout was used for all layers of the network for "Bayesian". Ensemble of T-scaled Networks is still the clear leader for this architecture.

5.3.3 Findings

We summarize our findings:

- Unfamiliar samples lead to much higher calibration error and label error, which can make their behavior unreliable in applications for which inputs are sampled differently in training and deployment.
- T-scaling is effective in reducing likelihood and calibration error on familiar and unfamiliar samples.
- The simple ensemble, when applied to T-scaled models, is the best method overall, reducing all types of error for both unfamiliar and familiar samples. The method of Kendall et al. [85] is the only other tested method to consistently reduce labeling error.
- T-scaling, distillation, and G-distillation all perform much better than the baseline.

Our recommendation: developers of any application that relies on prediction confidences (e.g. deciding whether to return a label, or to sound an alarm) should calibrate their models

	NLL		Brier		Label Error		ECE		E99	
Gender	**fam.**	**unf.**	**fam.**	**unf.**	**fam.**	**unf.**	**fam.**	**unf.**	**fam.**	**unf.**
Single Model	0.0769	0.5608	0.1332	0.3499	0.0219	0.1430	0.0132	0.1139	0.0063	0.0658
Single + T-scaling	0.0611	0.3553	0.1291	0.3262	0.0219	0.1430	**0.0024**	0.0850	0.0015	0.0131
Ensemble	**0.0513**	0.4103	**0.1163**	0.3281	**0.0180**	**0.1342**	**0.0031**	0.0861	0.0022	0.0348
Ensemble of T-scaled models	**0.0507**	**0.2995**	**0.1165**	**0.3116**	**0.0185**	**0.1338**	0.0070	**0.0653**	**0.0003**	**0.0023**
Distill	0.0706	0.4117	0.1321	0.3347	0.0211	**0.1352**	0.0081	0.0951	0.0039	0.0245
Distill + T-scaling	0.0694	0.3947	0.1317	0.3326	0.0211	**0.1352**	0.0067	0.0920	0.0034	0.0186
G-distill	0.0645	0.3559	0.1265	0.3250	0.0196	0.1391	0.0049	0.0815	0.0030	0.0138
G-distill + T-scaling	0.0641	0.3477	0.1263	0.3235	0.0196	0.1391	0.0041	0.0791	0.0026	0.0118
Novelty scaling	0.0611	0.3553	0.1291	0.3262	0.0219	0.1430	**0.0024**	0.0850	0.0015	0.0131
Bayesian	0.0795	0.5930	0.1341	0.3512	0.0218	0.1416	0.0139	0.1155	0.0070	0.0738
Bayesian + T-scaling	0.0617	0.3934	0.1295	0.3324	0.0217	0.1412	0.0050	0.0932	0.0018	0.0206

Table 5.4: Performance for DenseNet-161 classifier. Best and within significance range of best is in bold.

or, better yet, use calibrated ensembles. Ensembles achieve higher accuracy and better calibration, but at additional computational expense. We suspect that ensembles of shallower networks may outperform single deeper networks with similar computation costs, though we leave confirmation to future work. Tuning calibration on a validation set that is i.i.d. with training leads to overestimates of confidence for unfamiliar samples, so to minimize likelihood error for both unfamiliar and familiar samples, it may be best to obtain a small differently-sampled validation set.

5.4 SUMMARY

We show that modern deep network classifiers are prone to overconfident errors, especially for unfamiliar but valid samples. We show that ensembles of T-scaled models are best able to reduce all kinds of prediction error. Our work is complementary to recent works on calibration of i.i.d. data (e.g., Guo et al. [173]) and artificially distorted data [7]. More work is needed to improve prediction reliability with a single model in the unfamiliar setting and to consolidate learnings from the multiple recent studies of calibration and generalization. Data augmentation and representation learning are other important ways to improve generalization, and it would be interesting to evaluate their effect on prediction for both familiar and unfamiliar samples.

CHAPTER 6: FUTURE WORK AND CONCLUSION

6.1 FUTURE WORK

Knowledge transfer, especially with incomplete data, is still a difficult and open problem. We identify three areas of possible future improvements: (1) more human-like knowledge representation, (2) better validation and hyperparameter selection philosophy, and (3) improvements to individual sections.

6.1.1 Better Knowledge Representation

To improve knowledge transfer in general, we have to take a step back and ask the question: what exactly counts as knowledge? Scholars such as philosophers and educators may disagree [182, 183, 184, 185] on the exact definition. But clearly, the types of knowledge explored in the vision literature is limited. In this work, either the dataset, a learned model, the input distribution, the interaction of ground truths of different tasks, or the ability to generalize are considered knowledge. But as humans, we understand much more.

Given an object or category, we know their properties, such as their 3D shape, location or typical location, their behavior in their environment, and other attributes – for example, cars are rigid, they drive on the road fast, and come in many colors. We understand how these attributes contribute to our understanding of these objects. We understand why these objects behave this way – cars are vehicles, so they are mostly rigid and deform in very limited ways; vehicles are man-made objects, so their color don't usually matter to the categorization as opposed to e.g. birds. By extension, we know if and how these types of knowledge apply to novel objects and objects in novel environments. And we know how they should affect our decision process via logic reasoning. How can we mimic these human capabilities in machine learning?

One possible step forward is to represent knowledge not just as data, but also as the behavior of data. How objects respond to interactions is useful but hard to learn for non-interactive vision tasks. But it is possible to model, for example, how a task's output should change if the input changes in a certain systematic way. To gather instantiations of these changes, one way is to systematically generate a set of input transformations, e.g. rotation, color mapping, morphing networks [186], or attribute disentanglement networks [187]. Another way is to make use of temporal data such as videos, and learn possible transformations in an unsupervised way.

If we can generate and validate category-restricted hypotheses of how the output should transform given an input transformation, we can extract such information from pre-trained networks. We can also investigate if a hypothesis is universal, or only applicable to one or a few classes, and the relationship between the hypothesis and the object category (or their hypernyms). Such investigation further allows us to infer if a hypothesis holds for a new object or environment. If such a system is in place, we can even connect it to knowledge described in human language or some propositional logic representation. They can be used to generate a prior for whether or not a hypothesis should hold.

6.1.2 Better Validation Philosophy

With incomplete data, one needs to be careful with handling validation data, which is often assumed to be inaccessible. If any validation set ground truth for the inaccessible data is used for the ablation study, model selection, or hyperparameter selection, then it is being accessed. In this thesis, we mostly use data that are available to compute metrics as proxies for those on unavailable data and use some heuristics for avoiding overfitting, but these measures can turn out inaccurate or incorrect.

Information leak can be detrimental to lines of research such as continual learning or weakly supervised learning [188, 189]. Apparent advances in the area are sometimes only due to selection of hyperparameters that lead to better performance on the inaccessible data. Further, in a sense, the research community as a whole can be seen as selecting models on the test set of such unavailable data, by publishing new papers one at a time. But there is a distinction from supervised methods' overfitting to test data as a community. In supervised training at least we can approximate test performance by using validation performance. But with parts of data unavailable, such validation data disappears, and there is no reliable way to estimate the test performance of a model candidate besides the final test.

Finding a more systematic way of validating models when the evaluation needs to use data considered unavailable may be a less "attractive" research direction, but an extremely practical and necessary one. Without this kind of work, we cannot easily tell if a method in research areas with data usage constraints can hold up in the long run. One way to gain insight into this problem is from existing discussions of scientific evaluation design [190, 191, 192].

6.1.3 Improvements to Individual Chapters

DeepInversion. It may be worthwhile to analyze how much performance drop in class-incremental learning is due to forgetting and old-new class confusion respectively. Of relevance is recently published work that explicitly deal with old-new class similarity [193, 194], and interestingly Wu et al. [193] use a calibration method very similar to that in Chapter 5. An analysis may shed insight on exactly why our method is failing on iCIFAR and iILSVRC, and potentially improve our results.

Recently, there are emerging techniques that instead of keeping a subset of images, keep only their features [195]. It would be interesting to see if the efficiency or performance can be improved if we generate features using the batch normalization statistics.

Task-assisted domain adaptation. In the anchor task work, we were modeling the relationship between tasks by training and freezing last layers of a network trained on the source domain, making it learn to only output pairs for both anchor and main tasks that are "valid". This assumes that the last network layers must have learned an information bottleneck that both tasks' predictions must rely on, and the bottleneck's dimensionality must be lower than that of both predictions combined. The effectiveness of task-label-relationship constraint is, therefore, (1) reliant on the bottleneck being learned, which is not guaranteed to happen, and (2) at odds with the performance of the main task, which may benefit from a design without such an information bottleneck. We conjecture that it may be better for performance to separate the task relationship modeling responsibility from the prediction network.

Recently there have been methods proposed to model inter-variable relationships, for example, using mutual information. Specifically, we can first train the mutual information network to model task label relationships. This can be done on either pairs of ground truths or pairs of predictions for both main and anchor tasks, using only source domain data. Then, as we add the target domain samples into training, we add a loss term to increase the mutual information of pairs of task predictions in the target domain. This would be a more principled way of constraining the two tasks' outputs to conform to a fixed relationship with some theoretical guarantees.

In our experiments, we can also try to eliminate the requirement that the indoor scene anchor task must be manually labeled, and experiment with generating semantic segmentation targets using a pre-trained network. We can also combine some other domain adaptation methods that do not try to match source and target distributions when they are supposed to be different.

Other open questions for the effect of anchor tasks include: how do we make sure the

main task gets information from the anchor task output directly? Would a design built on PAD-Net [27] work? Can we adapt multiple main tasks from only one anchor task to leverage all the rich labeling of synthetic data? How cheap can the anchor task be made? Can Taskonomy [19] help in choosing which anchor task to use?

Improving generalization to unfamiliar data. Applying the recommendations of our study to other tasks such as detection might not be straightforward. For example, the detection scores of object proposals (e.g. in RPN) are only sorted and the top proposals are taken, so methods such as T-scaling that do not change the order of sample scores become irrelevant. A further analysis may be required for tasks that have components other than classification.

Another useful analysis is how the definition of "unfamiliar" can affect the conclusion of this line of work. In domain adaptation, domain generalization methods, and our generalization to unknown domains work, the testing "unfamiliar" set of samples different from the familiar is defined and fixed in an experiment, often as a specific domain based on the dataset partitions that are available. In reality, the similarity of familiar and unfamiliar may come on a spectrum, ranging from barely noticeable to unrecognizable. An oft-ignored pitfall is methods may perform differently on different unfamiliar sets, e.g. T-scaling may perform the best since our split of familiar and unfamiliar are somewhat similar, while NCR may gain an advantage on unfamiliar samples that are more different by explicitly detecting outliers.

We can potentially adjust this degree of difference in two new ways. One is to split subcategories into familiar and unfamiliar by their WordNet hierarchy rather than random choice, to adjust the difference between familiar and unfamiliar sets, and analyze each compared method's performance in multiple settings. Another is to perform attribute classification but generalize from one kind of animal to another, with varying degrees of similarity between animals – but we need a set of attributes that are not directly determined by the species (unlike AwA).

6.2 CONCLUSION

In knowledge transfer applications for computer vision, portions of data necessary for performing the transfer can go missing. Past training data can become inaccessible due to privacy, legal, practical, or business leverage reasons. Future test distribution data can be unlabeled, or worse yet, unknown completely before model deployment.

In this thesis, we have shown that several ways can be explored to overcome these data incompleteness challenges. First, when training data of some learned capability is inaccessible,

we can synthesize proxy data in their place to maintain the performance of the capability. We can synthesize just the labels using Learning without Forgetting, or the input images as well using DeepInversion. Second, with unsupervised domain adaptation where target domain ground truth is unavailable, we can use an anchor task. Anchor task ground truth available on both source and target domains can serve as a bridge between them, and we show that modeling the implicit guidance between the ground truth of main and anchor tasks can further boost performance. Third, to improve generalization into target domains that are completely unknown before deployment, our study shows that a model properly calibrated on a validation set, or better yet, a calibrated ensemble, are the best methods for producing well-behaved probability predictions for classification tasks.

REFERENCES

[1] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] H. Yin, P. Molchanov, Z. Li, J. M. Alvarez, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via deepinversion," 2020.

[3] Z. Li, L. Luo, S. Tulyakov, Q. Dai, and D. Hoiem, "Task-assisted domain adaptation with anchor tasks," *arXiv preprint arXiv:1908.06079*, 2019.

[4] Z. Li and D. Hoiem, "Improving confidence estimates for unfamiliar examples," 2020.

[5] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," *CVPR*, pp. 1521–1528, 2011.

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6405–6416.

[7] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *NIPS*, 2019.

[8] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber-physical systems, decision sciences, and data products," *Big data*, vol. 5, no. 3, pp. 246–255, 2017.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.

[10] B. Chidlovskii, S. Clinchant, and G. Csurka, "Domain adaptation in the absence of source domain data," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 451–460.

[11] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2014.

[13] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[15] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic convnet representation," in *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2014.

[16] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of learning and motivation*, vol. 24, pp. 109–165, 1989.

[17] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International Conference in Machine Learning (ICML)*, 2014.

[18] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[19] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018.

[20] A. V. Terekhov, G. Montone, and J. K. O'Regan, "Knowledge transfer in deep block-modular neural networks," in *Biomimetic and Biohybrid Systems*. Springer, 2015, pp. 268–279.

[21] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[22] Y. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[23] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[24] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, Jul 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734

[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[26] I. Kokkinos, "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5454–5463, 2017.

[27] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 675–684, 2018.

[28] W. Yang, W. Ouyang, X. Wang, J. S. J. Ren, H. Li, and X. Wang, "3d human pose estimation in the wild by adversarial learning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5255–5264, 2018.

[29] S. Sengupta, A. Kanazawa, C. D. Castillo, and D. W. Jacobs, "Sfsnet: Learning shape, reflectance and illuminance of facesin the wild'," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6296–6305.

[30] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng, "Boosted multi-task learning," *Machine learning*, vol. 85, no. 1-2, pp. 149–173, 2011.

[31] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3994–4003.

[32] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, 08 2017.

[33] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019, pp. 1920–1929.

[34] R. Hu, P. Dollár, K. He, T. Darrell, and R. B. Girshick, "Learning to segment every thing," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4233–4241, 2018.

[35] J. Lei, Z. Guo, and Y. Wang, "Weakly supervised image classification with coarse and fine labels," *2017 14th Conference on Computer and Robot Vision (CRV)*, pp. 240–247, 2017.

[36] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.

[37] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, "Tracking emerges by colorizing videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 391–408.

[38] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[39] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[40] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin, "Unsupervised pre-training of image features on non-curated data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2959–2968.

[41] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1734–1747, 2015.

[42] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3733–3742.

[43] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.

[44] G. M. van de Ven and A. S. Tolias, "Generative replay with feedback connections as a general strategy for continual learning," *arXiv preprint arXiv:1809.10635*, 2018.

[45] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[46] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," in *Proc. National Academy Sciences*, 2017.

[47] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3987–3995.

[48] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *ICLR*, 2018.

[49] A. Triki Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," in *ICCV*, 2017.

[50] W. Hu, Z. Lin, B. Liu, C. Tao, J. Tao, J. Ma, D. Zhao, and R. Yan, "Overcoming catastrophic forgetting for continual learning via model adaptation," in *ICLR*, 2019.

[51] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *NeurIPS*, 2017.

[52] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *CVPR*, 2018.

[53] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *ECCV*, 2018.

[54] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[55] E. Eaton and P. L. Ruvolo, "Ella: An efficient lifelong learning algorithm," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 507–515.

[56] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[57] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[58] G. Csurka, *Domain adaptation in computer vision applications*. Springer, 2017, vol. 2.

[59] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.

[60] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*, 2015, pp. 97–105.

[61] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3723–3732.

[62] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Nam Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3752–3761.

[63] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle consistent adversarial domain adaptation," in *International Conference on Machine Learning (ICML)*, 2018.

[64] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2242–2251, 2017.

[65] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, 2015, pp. 1180–1189.

[66] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. K. Chandraker, "Learning to adapt structured output space for semantic segmentation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7472–7481, 2018.

[67] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[68] M. Dudík, S. J. Phillips, and R. E. Schapire, "Correcting sample selection bias in maximum entropy density estimation," in *Advances in neural information processing systems*, 2006, pp. 323–330.

[69] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 193–200.

[70] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in neural information processing systems*, 2016, pp. 136–144.

[71] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *International Conference on Machine Learning*, 2013, pp. 10–18.

[72] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5543–5551.

[73] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi, "Generalizing across domains via cross-gradient training," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[75] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 113–123.

[76] D. Pomerleau, "Neural network vision for robot driving," in *Intelligent Unmanned Ground Vehicles*. Springer, 1997, pp. 53–72.

[77] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[78] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[79] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1729–1739.

[80] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996. [Online]. Available: https://doi.org/10.1023/A:1018054314350

[81] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[82] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[83] J. M. Hernández-Lobato, Y. Li, M. Rowland, D. Hernández-Lobato, T. D. Bui, and R. E. Turner, "Black-box α-divergence minimization," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. JMLR. org, 2016, pp. 1511–1520.

[84] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[85] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, 2017, pp. 5580–5590.

[86] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why m heads are better than one: Training a diverse ensemble of deep networks," *arXiv preprint arXiv:1511.06314*, 2015.

[87] O. Makansi, E. Ilg, O. Cicek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7144–7153.

[88] W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1689–1695, 2011.

[89] G. Fumera and F. Roli, "Support vector machines with embedded reject option," in *Pattern recognition with support vector machines*. Springer, 2002, pp. 68–82.

[90] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in *Advances in neural information processing systems*, 2017, pp. 4885–4894.

[91] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh, "Predicting failures of vision systems," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3566–3573.

[92] P. Wang and N. Vasconcelos, "Towards realistic predictors," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 36–51.

[93] P. R. Devarakota, B. Mirbach, and B. Ottersten, "Confidence estimation in classification decision: A method for detecting unseen patterns," in *Advances In Pattern Recognition*. World Scientific, 2007, pp. 290–294.

[94] Y. J. Lee and K. Grauman, "Object-graphs for context-aware visual category discovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 346–358, 2012.

[95] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1VGkIxRZ

[96] D. Tax, "One-class classification," Ph.D. dissertation, TU Delft, Delft University of Technology, 2001.

[97] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," in *Irish Conference on Artificial Intelligence and Cognitive Science*. Springer, 2009, pp. 188–197.

[98] L. Breiman and N. Shang, "Born again trees," UC Berkeley, Tech. Rep., 1996.

[99] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *SIGKDD*, 2006.

[100] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *NeurIPS*, 2014.

[101] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Workshop*, 2014.

[102] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[103] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[104] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[105] Z. Xu, Y.-C. Hsu, and J. Huang, "Training shallow and thin networks for acceleration via knowledge distillation with conditional adversarial networks," in *ICLR Workshop*, 2018.

[106] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *CVPR*, 2019.

[107] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *CVPR*, 2019.

[108] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," in *ICLR*, 2018.

[109] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *ICLR*, 2018.

[110] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, "Structured knowledge distillation for semantic segmentation," in *CVPR*, 2019.

[111] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *ICML*, 2018.

[112] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," in *ICLR*, 2016.

[113] A. Pilzer, S. Lathuiliere, N. Sebe, and E. Ricci, "Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation," in *CVPR*, 2019.

[114] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *CVPR*, 2017.

[115] R. G. Lopes, S. Fenu, and T. Starner, "Data-free knowledge distillation for deep neural networks," *arXiv preprint arXiv:1710.07535*, 2017.

[116] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, C. Xu, and Q. Tian, "Data-free learning of student networks," in *ICCV*, 2019.

[117] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2014.

[118] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[119] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan, "Active long term memory networks," *CoRR*, vol. abs/1606.02355, 2016. [Online]. Available: http://arxiv.org/abs/1606.02355

[120] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetting learning in deep neural networks," *arXiv preprint arXiv:1607.00122*, 2016.

[121] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[122] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[123] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[124] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, pp. 413–420.

[125] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[126] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[127] K. Bhardwaj, N. Suda, and R. Marculescu, "Dream distillation: A data-independent model compression framework," in *ICML Workshop*, 2019.

[128] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *CVPR*, 2015.

[129] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[130] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015.

[131] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[132] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1 (2), 2017, p. 3.

[133] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *CVPR*, 2016.

[134] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[135] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," in *ICLR*, 2018.

[136] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *ICLR*, 2019.

[137] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *ICML*, 2019.

[138] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[139] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *NeurIPS*, 2017.

[140] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?" in *ECCV*, 2018.

[141] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NIPS*, 2016.

[142] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[143] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[144] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *NeurIPS*, 2016.

[145] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *CVPR*, 2017.

[146] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *ICCVGIP*, 2008.

[147] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. A. Funkhouser, "Semantic scene completion from a single depth image," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 190–198, 2017.

[148] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. A. Funkhouser, "Physically-based rendering for indoor scene understanding using convolutional neural networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5057–5065, 2017.

[149] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *CVPR*, 2017.

[150] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[151] Y.-H. Tsai, K. Sohn, S. Schulter, and M. K. Chandraker, "Domain adaptation for structured output via discriminative patch representations," *CoRR*, vol. abs/1901.05427, 2019.

[152] T. Gebru, J. Hoffman, and L. Fei-Fei, "Fine-grained recognition in the wild: A multi-task domain adaptation approach," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1358–1367, 2017.

[153] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5001–5009, 2018.

[154] K. Fang, Y. Bai, S. Hinterstoißer, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3516–3523, 2018.

[155] D. Fourure, R. Emonet, É. Fromont, D. Muselet, N. Neverova, A. Trémeau, and C. Wolf, "Multi-task, multi-domain learning: Application to semantic segmentation and pose regression," *Neurocomputing*, vol. 251, pp. 68–80, 2017.

[156] M. Mostajabi, M. Maire, and G. Shakhnarovich, "Regularizing deep networks by modeling and predicting label structure," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5629–5638, 2018.

[157] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[158] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *SIGGRAPH*, 1999.

[159] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou, "Facewarehouse: A 3d facial expression database for visual computing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 413–425, 2014.

[160] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks)," in *International Conference on Computer Vision*, 2017.

[161] G. Trigeorgis, P. Snape, I. Kokkinos, and S. P. Zafeiriou, "Face normals "in-the-wild" using fully convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 340–349, 2017.

[162] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[163] L. Ladicky, B. Zeisl, and M. Pollefeys, "Discriminatively trained dense surface normal estimation," in *ECCV*, 2014.

[164] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[165] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang, "Representation learning using multi-task deep neural networks for semantic classification and information retrieval," in *HLT-NAACL*, 2015.

[166] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[167] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2016.

[168] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[169] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[170] M. Zhang, "Google photos tags two african-americans as gorillas through facial recognition software," *Forbes*, 2015, news article. [Online]. Available: https://www.forbes.com/sites/mzhang/2015/07/01/google-photos-tags-two-african-americans-as-gorillas-through-facial-recognition-software/#766df498713d

[171] D. Yadron and D. Tynan, "Tesla driver dies in first fatal crash while using autopilot mode," *The Guardian*, 2016, news article. [Online]. Available: https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk

[172] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: an empirical study," *arXiv preprint arXiv:1802.08760*, 2018.

[173] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017, pp. 1321–1330.

[174] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly weather review*, vol. 78, no. 1, pp. 1–3, 1950.

[175] T. Roos, P. Grünwald, P. Myllymäki, and H. Tirri, "Generalization to unseen cases," in *Advances in neural information processing systems*, 2006, pp. 1129–1136.

[176] H. Han and A. K. Jain, "Age, gender and race estimation from unconstrained face images," Michigan State University, Tech. Rep. MSU-CSE-14-5, 2014.

[177] H. Han, A. K. Jain, S. Shan, and X. Chen, "Heterogeneous face attribute estimation: A deep multi-task learning approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.

[178] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[179] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[180] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–256.

[181] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, J.-H. Rick Chang et al., "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[182] J. J. Ichikawa and M. Steup, "The analysis of knowledge," in *The Stanford Encyclopedia of Philosophy*, summer 2018 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2018.

[183] M. Steup and R. Neta, "Epistemology," in *The Stanford Encyclopedia of Philosophy*, spring 2020 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2020.

[184] P. R. Pintrich, "The role of metacognitive knowledge in learning, teaching, and assessing," *Theory into practice*, vol. 41, no. 4, pp. 219–225, 2002.

[185] T. de Jong and M. G. Ferguson-Hessler, "Types and qualities of knowledge," *Educational Psychologist*, vol. 31, no. 2, pp. 105–113, 1996. [Online]. Available: https://doi.org/10.1207/s15326985ep3102_2

[186] D. Ji, J. Kwon, M. McFarland, and S. Savarese, "Deep view morphing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2155–2163.

[187] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 35–51.

[188] B. Pfülb and A. Gepperth, "A comprehensive, application-oriented study of catastrophic forgetting in dnns," 2019.

[189] J. Choe, S. J. Oh, S. Lee, S. Chun, Z. Akata, and H. Shim, "Evaluating weakly supervised object localization methods right," *arXiv preprint arXiv:2001.07437*, 2020.

[190] J. P. Ioannidis, "Why most published research findings are false," *PLos med*, vol. 2, no. 8, p. e124, 2005.

[191] R. Moonesinghe, M. J. Khoury, and A. C. J. Janssens, "Most published research findings are false—but a little replication goes a long way," *PLoS medicine*, vol. 4, no. 2, 2007.

[192] L. R. Jager and J. T. Leek, "Empirical estimates suggest most published medical research is true," *arXiv preprint arXiv:1301.3718*, 2013.

[193] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.

[194] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[195] A. Iscen, J. Zhang, S. Lazebnik, and C. Schmid, "Memory-efficient incremental learning through feature adaptation," *arXiv preprint arXiv:2004.00713*, 2020.