Domain Disentangled Meta-Learning

Xin Zhang†, Yanhua Li†, Ziming Zhang†, Zhi-Li Zhang$
Worcester Polytechnic Institute, USA†, University of Minnesota, USA$
{xzhang17,yli15,zzhang15}@wpi.edu, zhzhang@cs.umn.edu

Abstract

A key challenge with supervised learning (*e.g.*, image classification) is the shift of data distribution and domain from training to testing datasets, so-called "domain shift" (or "distribution shift"), which usually leads to a reduction of model accuracy. Various meta-learning approaches have been proposed to prevent the accuracy loss by learning an adaptable model with training data, and adapting it to test time data from a new data domain. However, when the domain shift occurs in multiple *domain dimensions* (*e.g.*, images may be transformed by rotations, transitions, and expansions), the average predictive power of the adapted model will deteriorate. To tackle this problem, we propose a domain disentangled meta-learning (DDML) framework. DDML disentangles the data domain by dimensions, learns the representations of domain dimensions independently, and adapts to the domain of test time data. We evaluate our DDML on image classification problems using three datasets with distribution shifts over multiple domain dimensions. Comparing to various baselines in meta-learning and empirical risk minimization, our DDML approach achieves consistently higher classification accuracy with the test time data. These results demonstrate that domain disentanglement reduces the complexity of the model adaptation, thus increases the model generalizability, and prevents it from overfitting.

1 Introduction

Machine learning models (*e.g.*, trained for image classification and object recognition) have many real-world applications, for example, teaching children to learn zoo animals, enabling a self-driving car to travel safely on road networks, or assisting customers to find products in grocery stores. However, a practical challenge in these applications is that the statistical distribution of the data to be classified differs (sometimes significantly) from that of the training data. For example, a well-trained handwriting digit classifier may not work well when applied to a new user with a different writing style from the data used in the training process [1].

Such a challenge is called a *domain shift* (or *distribution shift*) [1–4], which usually leads to a reduction of classification accuracy. Various meta-learning approaches have been proposed to prevent or mitigate the accuracy loss by domain shift, such as optimization based model MAML [5], black-box model MBB [6–8], and ARM [9]. All these works assume that training tasks with different (shifted) data domains are provided, and each testing task represents a set of data from a new domain. To tackle the domain shift problem, these meta-learning approaches aim to learn an *adaptable* model that are able to deal with the shift by adapting to the test time data domain. At test time, MAML and MBB adapt the model to a batch of labeled data, where the model adaptation by ARM is based on a batch of unlabeled data.

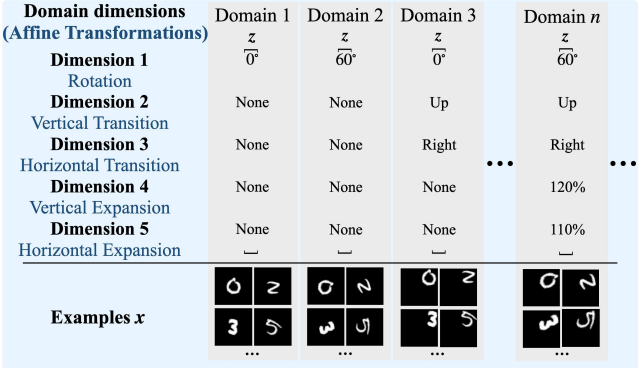


Figure 1: Illustration of domain shift (*i.e.*, distribution shift). Domain dimensions are independent. (z: domain shift vector; x: domain data examples.)

Motivation. All the above literature works consider the domain shift along one unified domain dimension, and adapt the model by extracting and integrating a task representation from the test time data. However, the shift may occur in a high-dimensional domain space. Taking the affNIST dataset [10] as an example, as shown in Fig. 1, the domain shift to MNIST images involves five different affine transformations (as shifting over domain dimensions), such as the image rotation, vertical and horizontal transitions, and vertical and horizontal expansions. It is well-known that, with a fixed number of training samples, the average (expected) predictive power of the extracted test time data representation deteriorates (*a.k.a.* overfitting), as the number of domain dimensions (*i.e.*, "features") increases [11, 12]. As validated in [13–16], disentangling and learning representations of the factors of variation from the high-dimensional input data space is effective in improving the

model generalizability, by reducing the model complexity. In this work, we are motivated to develop a model to disentangle the data domain by domain dimensions, for meta-learning to better adapt to the test time data domain.

Our DDML. We focus on the problem of domain shift in high-dimensional domain space, and propose domain disentangled meta-learning (DDML) by extracting and adapting domain disentangled representations from test time data. Our key contributions are summarized below:

- We are the *first* to introduce the domain independence assumption, that is widely valid in many computer vision tasks such as image classification, into meta-learning to reduce the model complexity for better generalization;
- We propose a domain disentangled meta-learning framework to learn an adaptable meta-model from training tasks with a domain disentanglement module, and adapt it to the domain disentangled representations of test time data;
- We present evaluation results of DDML in comparison to various state-of-the-art meta-learning algorithms on three benchmark image datasets with high-dimensional domain shift, *i.e.*, rotated MNIST [9], affNIST [10], and rotated Tiny ImageNet-C [17]. **We make our code available to contribute to the research community**[1]**.**

2 Preliminaries and Problem Definition

2.1 Preliminaries

A supervised learning task, τ, is defined by a set of labeled training data points $\{(\boldsymbol{x}, y)\}$ sampled i.i.d. from an unknown distribution $p(\boldsymbol{x}, y)$. The task aims to learn a model $g(\cdot;\theta) : \mathcal{X} \to \mathcal{Y}$ parameterized by θ, from the training dataset $\{(\boldsymbol{x}, y)\}$, such that the learned model can accurately predict a label $\hat{y} = g(\boldsymbol{x};\theta) \in \mathcal{Y}$ for each input $\boldsymbol{x} \in \mathcal{X}$ at the test time. The task objective is $\min_\theta \mathbb{E}_{p_{\boldsymbol{x}\boldsymbol{y}}}[\ell(g(\boldsymbol{x};\theta), y)]$, with $\ell(\hat{y}, y)$ as the loss function quantifying the difference between predicted vs ground-truth labels.

A meta-learning problem is defined as a set of supervised learning tasks $\{\tau\}$, with each $\tau \in \mathcal{T}$ sampled i.i.d. from a task distribution $p(\tau)$. The goal is to optimize a model from $\{\tau\}$, such that it can quickly adapt to each new test task $\tau \in \mathcal{T}$, also sampled i.i.d. from $p(\tau)$. Overall, as summarized in Eq. (2.1), where $\{(\boldsymbol{x}, y)\}$ denotes the labeled dataset from a task, meta-learning aims to optimize both θ and ϕ, so that the adaptation function h can quickly adapt the prediction function g to achieve high prediction accuracy on each new test task with a small set of data:

$$\min_{\theta,\phi} \mathbb{E}_{p_\tau}\left[\mathbb{E}_{p_{\boldsymbol{x}\boldsymbol{y}|\tau}}[\ell(g(\boldsymbol{x};\theta'), y)]\right], \text{ s.t. } \theta' = h(\theta, \{(\boldsymbol{x}, y)\}; \phi). \tag{2.1}$$

In the literature, there are different paradigms in implementing the adaptation function h. For instance, MAML [5], MBB [6–8], and ARM [9] employ gradient-based fine-tuning and deep neural network models, respectively.

2.2 Problem Definition

A domain shift problem. Meta-learning focuses on tasks $\{\tau\}$ all sampled from the same data domain. A domain shift problem is defined by a set of labeled training data sampled from different data domains. Each data point $(\boldsymbol{x}, y, \boldsymbol{z})$ has an observed domain identifier (*i.e.*, a vector) $\boldsymbol{z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_M]$ representing the associated data domain with M dimensions. For example, in affNIST dataset [10], $\boldsymbol{z}_m$ could represent the rotation degree, offsets in vertical and horizontal transitions and expansions of the MNIST images. Hence, in the domain shift problem, the task sampling process is governed by a domain sampling distribution, $\boldsymbol{z} \sim p(\boldsymbol{z})$, and a task sampling distribution $\tau \sim p(\tau|\boldsymbol{z})$. The goal is to learn a model $g(\cdot;\theta) : \mathcal{X} \to \mathcal{Y}$ from training tasks i.i.d. sampled from the different data domains. At test time, the learned model $g(\cdot;\theta)$ can quickly adapt to a test task with a batch of K data points, without domain identifier $\boldsymbol{z}$ observed. By adaptation, it updates the model parameters to θ', *i.e.*, $\hat{y} = g(\cdot;\theta')$, that predicts a label $\hat{y} \in \mathcal{Y}$ for an input $\boldsymbol{x} \in \mathcal{X}$ from the same test task. Hence a domain shift problem is modeled as follows[2]

$$\begin{aligned} &\min_{\theta,\phi} \mathbb{E}_{p_{\boldsymbol{z}}}\left[\mathbb{E}_{p_{\tau|\boldsymbol{z}}}\left[\mathbb{E}_{p_{\boldsymbol{x}\boldsymbol{y}|\tau\boldsymbol{z}}}\ell(g(\boldsymbol{x};\theta'), y)\right]\right], \\ &\text{s.t. } \theta' = h(\theta, \{\boldsymbol{x}\}; \phi). \end{aligned} \tag{2.2}$$

Such a formulation determines that the network architectures in learning have to be sequential, *i.e.*, one module taking the output of another module as its input. Theorem 1 below explains the solution to Eq. (2.2) from the probability perspective, and is proved using the Jensen's inequality [3] as in Appendix.

Theorem 1. *The solution to Eq.* (2.2) *represents a tight upper bound of the maximum log-likelihood of the observed data* $\{(\boldsymbol{x}, y)\}$.

[1] The code is accessible from `https://github.com/XinZhang525/SDM-DDML`.

[2] The data for adaptation can be either labeled $\{(\boldsymbol{x}, y)\}$ or unlabeled $\{\boldsymbol{x}\}$. Our presentation will focus on unlabeled test time data, and our proposed approach can naturally adapt to the setting with labeled test time data.

[3] See more details in `https://en.wikipedia.org/wiki/Jensen%27s_inequality`.

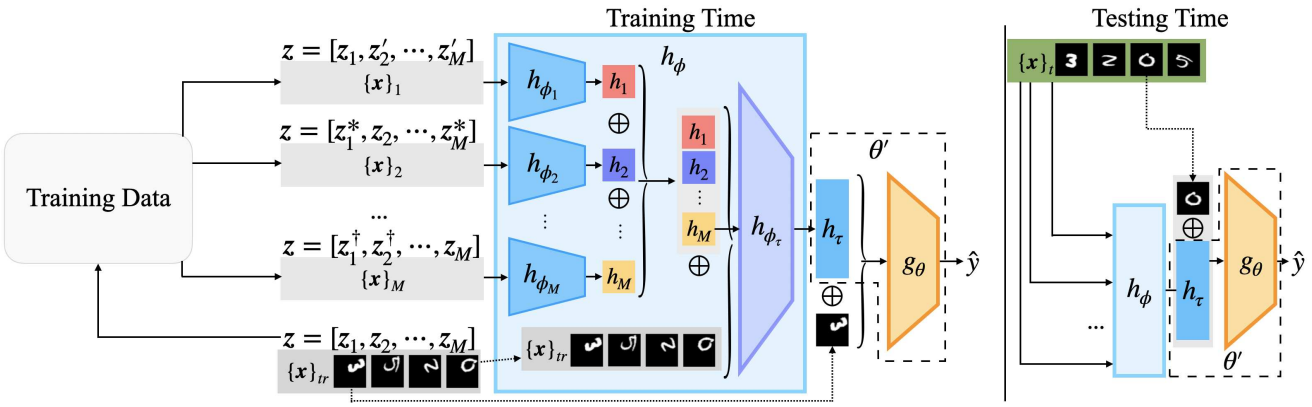


Figure 2: DDML architecture, with $\bigoplus$ denoting the feature concatenation operation.

Proof. The objective of meta-learning in Eq. 2.1 is equivalent to maximizing the probability of given data $\{(\boldsymbol{x}, y)\}$, *i.e.* $\max_{\omega\in\Omega} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{X}\times\mathcal{Y}}[p(\boldsymbol{x}, y; \omega)]$, with $p(\boldsymbol{x}, y; \omega)$ representing the probability distribution of observing a data point $(\boldsymbol{x}, y)$ parameterized by ω. By introducing latent variables $\boldsymbol{z} \in \mathcal{Z}, \tau \in \mathcal{T}$ representing domain and task information, we have

$$\begin{aligned}
&\max_{\omega\in\Omega} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{X}\times\mathcal{Y}}[p(\boldsymbol{x}, y; \omega)] \\
&= \int p_{\boldsymbol{x}y\tau\boldsymbol{z}} p(\boldsymbol{x}, y, \tau, \boldsymbol{z}; \omega) d(\boldsymbol{x}, y, \tau, \boldsymbol{z}) \\
&= \iint p_{\tau\boldsymbol{z}} p_{\boldsymbol{x}y|\tau\boldsymbol{z}} p(\boldsymbol{x}, y; \omega|\tau, \boldsymbol{z}) p(\tau, \boldsymbol{z}) d(\boldsymbol{x}, y) d(\tau, \boldsymbol{z}) \\
&\equiv \mathbb{E}_{p_{\tau\boldsymbol{z}}} \left[\mathbb{E}_{p_{\boldsymbol{x}y|\tau\boldsymbol{z}}} p(\boldsymbol{x}, y; \omega|\tau, \boldsymbol{z})\right] \\
&= \mathbb{E}_{p_{\boldsymbol{z}}} \left[\mathbb{E}_{p_{\tau|\boldsymbol{z}}} \left[\mathbb{E}_{p_{\boldsymbol{x}y|\tau\boldsymbol{z}}} p(\boldsymbol{x}, y; \omega|\tau, \boldsymbol{z})\right]\right].
\end{aligned}$$

Suppose that probability $p(\boldsymbol{x}, y; w|\tau, \boldsymbol{z})$ follows Gaussian, *i.e.* $p(\cdot) \equiv \exp(-\ell(\cdot))$ with some loss function ℓ, then a tight lower bound of maximum log-likelihood of the data $(\boldsymbol{x}, y)$ exists (by Jensen's inequality), *i.e.*,

$$\begin{aligned}
&\max_{\omega\in\Omega} \mathbb{E}_{p_{\boldsymbol{z}}} \left[\mathbb{E}_{p_{\tau|\boldsymbol{z}}} \left[\mathbb{E}_{p_{\boldsymbol{x}y|\tau\boldsymbol{z}}} p(\boldsymbol{x}, y; \omega|\tau, \boldsymbol{z})\right]\right] \\
&\leq \min_{\omega\in\Omega} \mathbb{E}_{p_{\boldsymbol{z}}} \left[\mathbb{E}_{p_{\tau|\boldsymbol{z}}} \left[\mathbb{E}_{p_{\boldsymbol{x}y|\tau\boldsymbol{z}}} \ell(\boldsymbol{x}, y; \omega|\tau, \boldsymbol{z})\right]\right],
\end{aligned} \tag{2.3}$$

where the loss function in meta-learning follows $\ell(\boldsymbol{x}, y; \omega|\tau, \boldsymbol{z}) = \ell(g(\boldsymbol{x}; \theta'), y)$ with $\theta' = h(\theta, \{\boldsymbol{x}\}_{\tau\boldsymbol{z}}; \phi)$, that is, $\omega = [\theta, \phi]$. □

The Challenges of high-dimensional data domains. In the literature, the meta-learning methods [5, 6, 8, 18–24] have been validated extensively on solving the problem of domain shift in Eq. (2.2) over one unified domain space [5, 6, 8, 21, 25]. However, when the domain shift occurs in a high-dimensional domain space, *e.g.*, with shifting over rotation, expansion, transition, occlusions, and more, by "the curse of dimensionality", the adaptability of function h will deteriorate (*a.k.a* overfitting). To tackle this problem, we are motivated to advance the model h by disentangling the task domain dimensions to reduce the model complexity, thus improving the model generalizability and effectiveness of g, when adapting to the test task data. The rational of this disentanglement has been validated in [13–16].

Domain Disentangled Meta-Learning (DDML). The domain disentanglement is done by assuming the domain dimensions to be *independent*, *i.e.*, $p(\boldsymbol{z}) \equiv \prod_{m=1}^{M} p(\boldsymbol{z}_m)$. In fact, our assumption in general holds in reality, *i.e.*, each domain dimension $\boldsymbol{z}_m$ can be well-defined to represent one independent aspect of the data points. For example, in affNIST dataset [10], each affine transformation (as a domain dimension), *i.e.*, rotation, transition, or expansion, has no impact on other dimensions. Therefore, from the perspective of log-likelihood maximization we have the following equivalent expectation where at the right-hand side, the expectation operators are exchangeable:

$$\mathbb{E}_{p_{\boldsymbol{z}}} \equiv \mathbb{E}_{p_{\boldsymbol{z}_1}} \circ \mathbb{E}_{p_{\boldsymbol{z}_2}} \circ \cdots \circ \mathbb{E}_{p_{\boldsymbol{z}_M}}. \tag{2.4}$$

This decomposition allows us to design networks where the input of each module for a domain can be independent and parallel, *i.e.*, domain disentanglement. Based on Eq. (2.2) and Eq. (2.4), our DDML objective function with domain disentanglement can be further written as follows:

$$\begin{aligned}
&\min_{\theta,\phi} \mathbb{E}_{p_{\boldsymbol{z}_1}} \circ \cdots \circ \mathbb{E}_{p_{\boldsymbol{z}_M}} \left[\mathbb{E}_{p_{\tau|\boldsymbol{z}}} \left[\mathbb{E}_{p_{\boldsymbol{x}y|\tau\boldsymbol{z}}} [\ell(g(\boldsymbol{x}; \theta'), y)]\right]\right], \\
&\text{s.t.} \theta' = h(\theta, \{\boldsymbol{x}\}; \phi).
\end{aligned} \tag{2.5}$$

Why domain disentanglement works? Intuitively, let us consider the information flows (*i.e.* paths) in a deep neural network: Given M modules, where each module is a layer of $N_m (\forall m \in [M])$ neurons, the fully connected M layers induce $\prod_m N_m$ paths in the deep neural network, while our disentangled modules in Eq. (2.5) only induce

Algorithm 1 Domain Disentangled Meta-Learning

1: // Training procedure
Require: Training data $\{\mathcal{D}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in\mathcal{Z}}$, M-dimensional domain $\mathcal{Z}$, the number of training steps T, learning rate η, batch size K.
2: Randomly initialize θ and ϕ.
3: **for** $t = 1, \cdots, T$ **do**
4: Sample a domain $\mathcal{D}_{\boldsymbol{z}}$, by uniformly sampling $\boldsymbol{z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_M]$ from $\mathcal{Z}$.
5: Sample a training task τ_{tr} uniformly from $\mathcal{D}_{\boldsymbol{z}}$.
6: Extract M disentangled tasks $\{\boldsymbol{x}\}_1, \cdots, \{\boldsymbol{x}\}_M$ via Alg. 2.
7: $\theta' = [\theta, h_\tau] \leftarrow h_\phi(\theta, \{\boldsymbol{x}\}_{tr}, \{\boldsymbol{x}\}_1, \cdots, \{\boldsymbol{x}\}_M; \phi)$
8: $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \sum_{(\boldsymbol{x},y)\in\tau_{tr}} \ell(g(\boldsymbol{x}; \theta'), y))$.
9: **end for**
10: // Testing procedure
Require: θ, ϕ, test task $\tau_t = \{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(K)}\}$.
11: $\theta' = [\theta, h_\tau] \leftarrow h_\phi(\theta, \{\boldsymbol{x}\}_t, \{\boldsymbol{x}\}_t, \cdots, \{\boldsymbol{x}\}_t; \phi)$.
12: $\hat{y}_k \leftarrow g(\boldsymbol{x}^{(k)}; \theta')$ for $k = 1, \cdots, K$.

$\sum_m N_m$ paths, which is *exponentially* smaller in terms of model complexity. Similar to the observations in [13–16], disentangling the domain dimensions reduces the function complexity, thus increases its generalizability and mitigates the potential overfitting problem. In contrast, without domain disentanglement, the state-of-the-art meta-learning works [5,6,8,9,21] learn a unifying adaptation function h to extract the task representation, thus implicitly characterizes the potential correlations among the domain dimensions, which significantly increases the model complexity, thus downgrades the model generalizability, and is prone to overfitting.

3 Domain Disentangled Meta-Learning

From the DDML objective in Eq. (2.5), there are two functions to learn, *i.e.*, the prediction function g, and the adaptation function h. We model both functions with deep neural networks parameterized by θ and ϕ respectively. In this section, we will tackle two key challenges, including *i)* how to design an algorithm to jointly learn h_ϕ and g_θ that solve the DDML objective in Eq. (2.5); and *ii)* how to design h_ϕ to learn and adapt a task from its disentangled domain space. Fig. 2 shows our DDML architecture.

3.1 DDML Algorithm

The goal of DDML algorithm is to jointly meta-learn the adaptation network h_ϕ and g_θ, such that h_ϕ can learn representations of a given task in its disentangled domain dimensions, and adapt the prediction network g_θ to the task. Hence, we need to specify a domain distribution

Algorithm 2 Domain Disentanglement

Require: A task τ_{tr}, domain $\boldsymbol{z}$, training data $\{\mathcal{D}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in\mathcal{Z}}$.

1: **for** $m = 1, \cdots, M$ **do**
2: Randomly sample $\boldsymbol{z}^{(m)} \in \mathcal{Z}$, with $\boldsymbol{z}^{(m)}_m = \boldsymbol{z}_m$.
3: Sample $\{\boldsymbol{x}\}_m$ uniformly from $\mathcal{D}_{\boldsymbol{z}^{(m)}}$.
4: **end for**

$p_{\boldsymbol{z}}$ as a prior, for sampling task domains in the training stage. However, we have no knowledge how the test time domain sample distribution $p_{\boldsymbol{z}}$ looks like. Fortunately, the state-of-the-art works in deep learning have shown that uniformly sampling over a quantity of interest, can achieve robust performance to that quantity [9, 26–28]. Hence, in DDML algorithm, we sample training tasks τ_{tr}'s uniformly from the domain space $\mathcal{Z}$. We introduce the proposed DDML algorithm in both training and testing processes below, with the pseudo-code presented in Alg. 1.

Training procedure. For each iteration in training, one domain $\boldsymbol{z}$ is uniformly sampled from $\mathcal{Z}$, and one training task τ_{tr} is uniformly sampled from the domain data $\mathcal{D}_{\boldsymbol{z}}$ (line 4–5). Then, based on the training task dataset τ_{tr}, we prepare M disentangled task datasets, *i.e.*, $\{\boldsymbol{x}\}_1, \cdots, \{\boldsymbol{x}\}_M$, with each $\{\boldsymbol{x}\}_m$ preserving the dimension $\boldsymbol{z}_m$ from τ_{tr}, while randomly changing all other $M-1$ domain dimensions (line 6). See more details of preparing all $\{\boldsymbol{x}\}_m$'s in Sec. 3.2 and Alg. 2. Taking the training task dataset $\{\boldsymbol{x}\}_{tr}$, and its M disentangled task datasets $\{\boldsymbol{x}\}_1, \cdots, \{\boldsymbol{x}\}_M$ as input, h_ϕ network (line 7 and Fig. 2) outputs a representation h_τ of the training task τ_{tr}, which as an input is fed into the prediction network g_θ. Then, θ and ϕ are both updated by gradient descent to minimize the objective in Eq. (2.5) (line 8).

Testing procedure. During the test time, given a task τ_t as a batch of data points, h_ϕ takes its test dataset $\{\boldsymbol{x}\}_t$ as input to extract the test task representation h_τ (line 11). Then, taking h_τ as input, the prediction network g_θ is adapted to the test task, and can perform prediction for any data point from the the test task τ_t (line 12).

3.2 Domain Disentanglement with Network h_ϕ

In the training process, we disentangle the M domain dimensions of a task τ_{tr}, using M deep neural networks, *i.e.*, $h_{\phi_1}, \cdots, h_{\phi_M}$ (see Fig. 2). Each h_{ϕ_m} with $1 \leq m \leq M$ learns a representation h_m of the domain dimension $\boldsymbol{z}_m$ from the task τ_{tr}. Given a training task τ_{tr}, M disentangled task datasets to τ_{tr} are prepared, *i.e.*, $\{\boldsymbol{x}\}_1, \cdots, \{\boldsymbol{x}\}_M$, where each $\{\boldsymbol{x}\}_m$ preserves the dimension $\boldsymbol{z}_m$ from τ_{tr}, while randomly changing all other $M-1$ domain dimensions. The process of

preparing M disentangled tasks for τ_{tr} is summarized in Alg. 2. The data points of each disentangled task $\{\boldsymbol{x}\}_m$ is used to train the corresponding disentangled adaptation network h_{ϕ_m}, to guarantee that h_{ϕ_m} only extracts a representation for $\boldsymbol{z}_m$. Then, the concatenation of all h_m's ($m = 1, \cdots, M$) and the task data samples $\{\boldsymbol{x}\}_{tr}$ together are input into the task adaptation network h_{ϕ_τ}, which extracts the representation h_τ of the task τ_{tr}. As a result, the adaptation network h_ϕ consists of $M+1$ networks $h_{\phi_1}, \cdots, h_{\phi_M}$, and h_{ϕ_τ}, with the parameter vector $\phi = [\phi_1, \cdots, \phi_M, \phi_\tau]$.

In the testing process, no domain identifier $\boldsymbol{z}$ is provided for a test task τ_t. Hence, the same dataset $\{\boldsymbol{x}\}_t$ of τ_t will be used as an input to all adaptation networks $h_{\phi_1}, \cdots, h_{\phi_M}$, and h_{ϕ_τ} to extract the task representation h_{τ_t} for prediction network g_θ to adapt.

4 Experiments

We evaluate our proposed DDML by comparing it with baselines on three image classification benchmarks, which naturally lie in high-dimension data domains, including the rotated MNIST [9], affNIST [10] and rotated Tiny ImageNet-C [17]. In this section, we introduce the experiment settings, and present the experimental results on the three datasets to show: *i)* our DDML outperforms baselines on test tasks with domain shifts; *ii)* with the domain disentanglement module, DDML can extract higher quality representations for individual domain dimensions than other baselines; *iii)* DDML converges faster than baselines, and it is a robust approach to various model parameters, such as the training data size, the number of disentangled domain dimensions, and the size of the domain representation vector.

4.1 Experiment settings

We evaluate and compare DDML with four baselines below in meta-learning and empirical risk minimization models [5–9].

- *ERM* [29] (empirical risk minimization) is a zero-shot generalization approach. It assumes that the training and testing data follow the same underlying distribution and strives to train a model with minimal average error over the training data.
- *ARM* [9] (adaptive risk minimization) is a SOTA domain adaptation approach in tackling the domain shift problem by meta-learning. It uses unlabeled data adaptation for test time adaptation.
- *MAML* [5] represents the family of model-agnostic meta-learners, which is an optimization based meta-learning approach. It adapts to test tasks by minimizing the classification loss on test task support data.
- *MBB* [7, 8] is a black-box meta-learning approach, which aims to learn task-specific parameters to be used for test time prediction, from an LSTM trained with a sequence of labeled training data.

Note that ERM and ARM are baselines for adaptation with unlabeled test data, where MAML and MBB are designed to adapt to labeled test data. Our proposed DDML can be naturally extended to adapt labeled test data, by taking labeled support set data as input to adaptation function h_ϕ, and evaluate the classification loss on a labeled query set data on $g_{\theta'}$. As a result, we compare our proposed DDML with all the above baselines on three image classification benchmark datasets below.

Rotated MNIST. We modify the MNIST by rotating the images with 5° increments from 0° to 350° to generate the training data. We consider two (2) domain dimensions for tasks on this dataset, including i) the rotation degrees and ii) the number of image classes (N) and the number of copies (k) for a class in a task, namely, this dimension defines an N-way-K-shot problem. For the N-way-K-shot dimension, we only choose four settings, including 5-way-1-shot, 5-way-5-shot, 10-way-1-shot, and 10-way-5-shot.

AffNIST Dataset [10] imposes different affine transformations on the MNIST data. The affine transformations includes counter-clockwise rotations (from $-20°$ to 20°), vertical and horizontal transitions, and vertical and horizontal expansion. In the experiment, we consider five (5) domain dimensions for tasks on this dataset, including i) the rotation degrees, ii) transition, iii) vertical expansion, iv) horizontal expansion, and v) the N-way-K-shot setting.

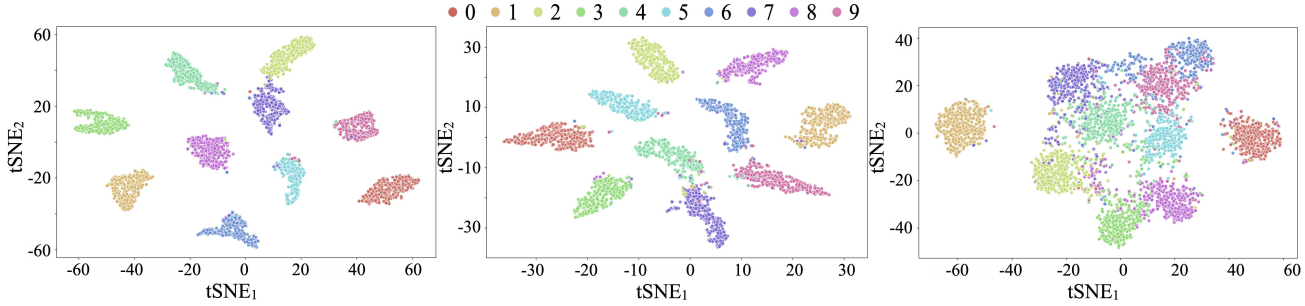
For the rotation degree dimension, we divide the rotations into 8 disjoint equal rotation degree bins. For the transition dimension, we consider 4 quadrant directions as four possible choices. For the vertical and horizontal expansion dimensions, we equally divide the expansion degree from 0.8 to 1.2 into 4 equal bins. For the N-way-K-shot dimension, we only choose four settings, including 5-way-1-shot, 5-way-5-shot, 10-way-1-shot, and 10-way-5-shot.

Rotated Corrupted Image Datasets. We evaluate DDML and all baselines on a rotated Tiny ImageNet-C dataset [17]. This dataset augments the Tiny ImageNet dataset using different types of image corruptions and the level of severity. In the experiment, we consider four (4) domain dimensions for tasks on this dataset, including i) the rotation degrees, ii) the type of corruption, iii) the severity level of the corruption, and iv) the N-way-K-shot setting.

For the domain dimension i) of the rotation, we consider 4 rotation degrees including 10° and 30° for

Table 1: State-of-the-art performance comparison on testing accuracy (with standard deviations in parentheses) over domains with metrics BCA (best case accuracy), AA (average accuracy) and WCA (worst case accuracy).

Method & Setup	Rotated MNIST		AffNIST		Rotated Tiny ImageNet-C		
	AA	WCA	AA	WCA	BCA	AA	WCA
ERM	81.9 (0.2)	62.0 (0.8)	92.2 (0.2)	83.5 (0.2)	56.7 (0.7)	50.3 (0.7)	37.3 (1.7)
ARM	97.1 (0.3)	94.0 (0.7)	96.3 (0.2)	90.5 (0.5)	58.2 (0.6)	51.3 (0.6)	35.0 (1.2)
MBB	92.8 (0.2)	80.0 (1.0)	94.0 (1.1)	82.3 (1.4)	58.0 (0.7)	51.0 (0.6)	34.0 (1.7)
MAML	92.8 (0.4)	87.2 (1.2)	93.6 (0.4)	80.2 (0.9)	57.2 (0.5)	49.8 (0.5)	30.7 (1.0)
DDML (Ours)	**98.1 (0.1)**	**95.9 (0.6)**	**98.3 (0.2)**	**93.4 (0.7)**	**62.7 (0.9)**	**54.1 (0.8)**	**42.7 (1.0)**



(a) DDML (Ours) on rotated MNIST. (b) ARM on rotated MNIST. (c) ERM on rotated MNIST.

Figure 3: tSNE results of ARM, DDML and ERM by learned hidden features on rotated MNIST.

the training data, and 20° and 40° for the test data. For the domain dimension ii) of the corruption type, we consider seven different corruption types from [17]. For the domain dimension iii) of the severity level, we consider two severity levels (*i.e.*, high and low). For iv) the N-way-K-shot dimension, we consider four settings, including 10-way-1-shot, 10-way-5-shot, 20-way-1-shot, and 20-way-5-shot.

For all the experiments, we report the average accuracy (AA) over all domains, the worst case accuracy (WCA) on the worst performed domain, and best case accuracy (BCA) on the best-performed domain. We delegate more implementation details and model architectures in Appendix.

4.2 DDML Performance

Tab. 1 summarizes our state-of-the-art result comparison, where our DDML model outperforms baselines over all datasets in the best case accuracy (BCA), average accuracy (AA), and worst case accuracy (WCA). This shows that the proposed DDML has good generalization ability over all domain shifts at test time. Below, we use tSNE[4] to visualize and understand the learned hidden features in the prediction model, and show how model accuracy evolves when the test domain shifts along different domain dimensions. For brevity, we present results for the setting of unlabeled adaptation, where similar results for labeled adaptation are in Appendix.

Looking over different N-way-K-shot tasks of a fixed dataset, Fig. 3 shows the tSNE results on learned hidden features of test images sampled from different test domains on rotated MNIST. Comparing with baselines ARM and ERM, we observe that the hidden features learned by DDML (captured from the last hidden layer in the prediction network g_θ) generate more compact data clusters with clearer cluster separations. This echoes and explains the classification accuracy results in Tab. 1, that the adapted models by DDML can better capture key features to classify the input images than other baselines. For other datasets, we obtained similar results, and we delegate them to Appendix.

Next, we evaluate how the accuracy changes, when the test domain shifts along different domain dimensions. Fig. 4 and Fig. 5 show the comparison results to all baselines on the rotated MNIST and affNIST datasets, respectively. Our DDML outperforms all baselines in different settings.

4.3 How well is the domain disentangled?

Fig. 6 and Fig. 7 show the tSNE results on learned domain representations of each test task with ARM [9] and our model DDML on the rotated MNIST and the

[4] tSNE is a dimension reduction approach for the visualization of a high-dimensional data. See details in [30].

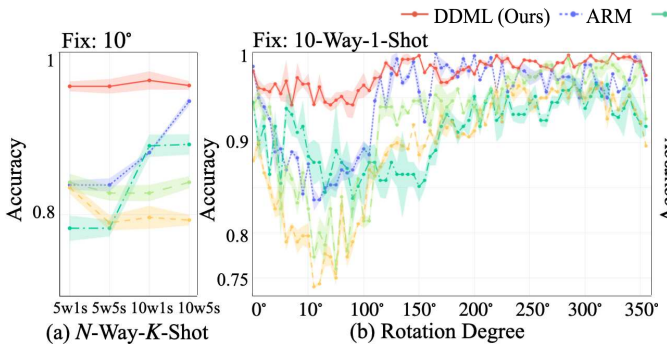


Figure 4: Rotated MNIST results.

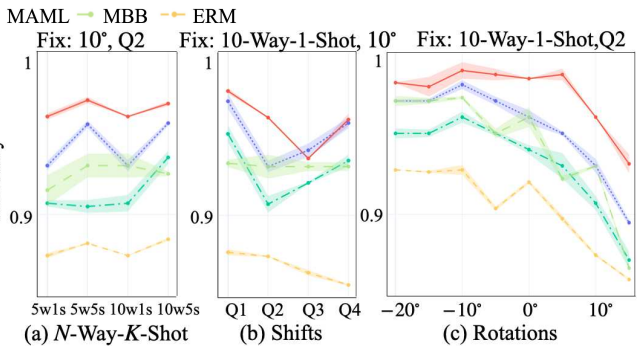


Figure 5: AffNIST results.

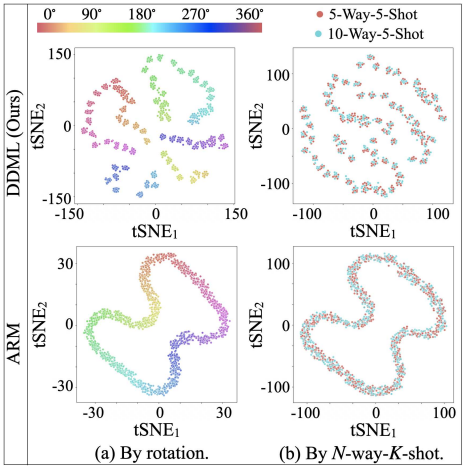


Figure 6: tSNE of learned domain representations on rotated MNIST.

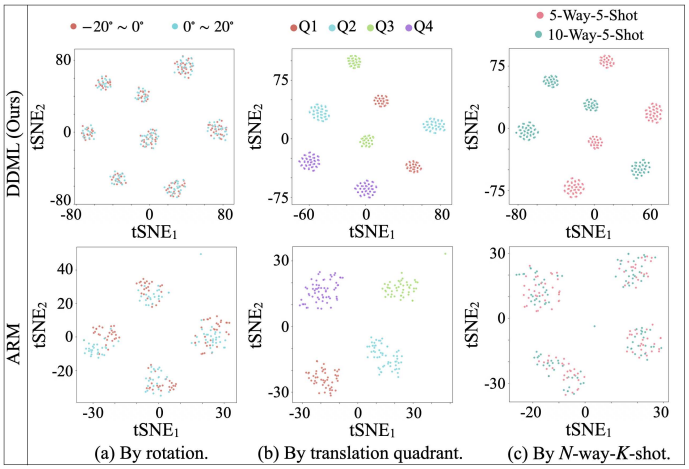
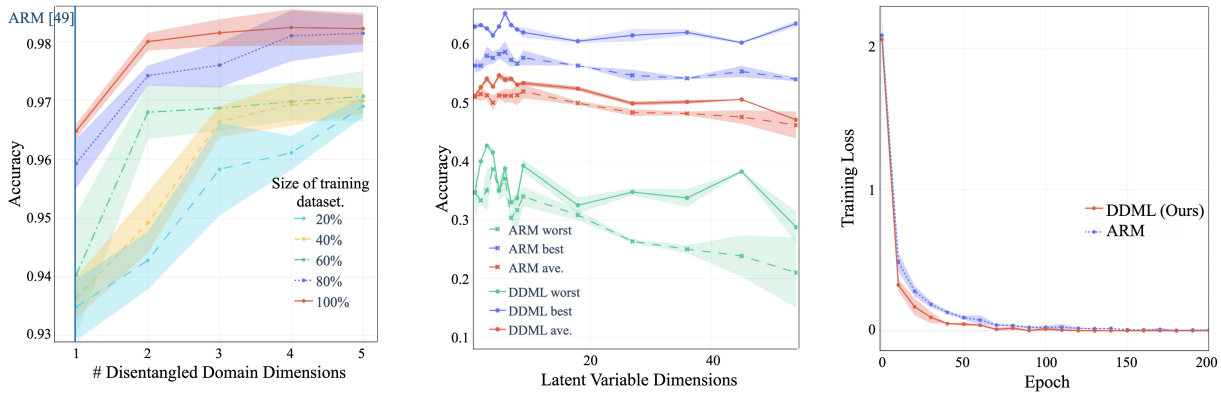


Figure 7: tSNE of learned domain representations on affNIST.

affNIST data respectively. Note that the domain representation in DDML is a concatenation of $[h_1, \cdots, h_M]$ from M adaptation networks, while that of ARM is the output vector from its adaptation network. Based on the data compactness within a cluster and data distance across clusters, we observe that DDML extracts higher quality domain representations than ARM. In rotated MNIST, we first apply tSNE on the domain representations from ARM and DDML by coloring the tasks by their rotation degrees. As shown in Fig. 6a, the domain representations learned from DDML can better capture different rotation degrees, as tasks with the same rotation degree are clearly clustered. In contrast, with ARM, the tasks with different rotation degrees are not well clustered. In Fig. 6b, we show tSNE results by coloring test tasks in the two N-way-K-shot settings. We observe that there is no clear cluster structure for the domain dimension of N-way-K-shot. This shows that the rotation degree is a dominant domain dimension on rotated MNIST.

Fig. 7a shows tSNE results on test tasks from affNIST data, colored by their rotation degrees. Clearly, test tasks are not clustered by their rotation degrees. On the other hand, looking at the domain dimensions of N-way-K-shot and quadrant translation, we observe from Fig. 7b that these two domain dimensions dominate the clustering structure of test tasks. It shows that our model DDML extracts eight task clusters with each cluster representing a combination of N-way-K-shot and quadrant translation. However, for ARM in Fig. 7b, the test tasks are only grouped in four clusters by different quadrant-transitions.

These observations show that DDML can extract higher quality domain representations than ARM. As a result, with the disentangled domain representation, DDML can thus better adapt the prediction function g_θ to test tasks.



(a) Accuracy w/ domain disentanglement.

(b) Accuracy w/ domain representation size.

(c) AffNIST training curves.

Figure 8: Performance with different training data size and disentangled domain dimension number on affNIST (a), with different domain representation size on rotated Tiny ImageNet-C (b), and the training curve on affNIST (c).

4.4 Ablation Studies & Convergence Analysis

In this section, we investigate how various hyper-parameters affect the performances of the proposed DDML, including i) the size of training data, ii) the number of domain dimensions disentangled, and iii) the size of domain representation vector. We also conduct convergence analysis of our DDML.

• *Impact of training data size.* Now, we investigate how the training data size affects the performance of DDML. We do experiments on the affNIST dataset with either the entire dataset as training dataset or a portion of it, *i.e.*, 100% (all), 80%, 60%, 40%, and 20% of the entire dataset. Fig. 8a shows the results. The x-axis represents the DDML models with different numbers of domain dimensions disentangled. The y-axis shows the average test accuracy, The curves in different colors represent the results obtained with different training data sizes. Clearly, when increasing the training data size, the DDML performance increases accordingly.

• *Impact of the number of disentangled domain dimensions.* Now, we examine how the number of disentangled domain dimensions affect the DDML performances. We conduct experiments on the affNIST dataset by using different numbers (from 1 to 5) of (parallel) adaptation networks to extract the domain representation. Note that when only one adaptation network is used, it is equivalent to the ARM approach. Fig. 8a shows the average accuracy over test tasks. It shows a clear increase in average model performance when more domain dimensions are disentangled. Moreover, with different training data sizes, Fig. 8a shows that there is more performance space to improve when smaller training datasets are used. For example, when using 20% (blue) and 40% (yellow) of training data, DDML (with 5 disentangled domain dimensions) can improve the classification accuracy by 4%, comparing to 2% when (100%) all training data are used.

• *Impact of the size of domain representation.* We further investigate how the size of domain representation affects the performance in DDML vs ARM [9]. Fig. 8b shows the performance changes (*i.e.*, average accuracy over test domains, the best case accuracy, and the worst case accuracy) with respect to the latent variable dimensions, namely, the number of channels, using the rotated Tiny ImageNet-C data. In general, we observe that in both ARM and DDML, increasing the latent dimension to be large (*i.e.*, more than 5) does not promote performance. This is likely due to the meta-overfitting of a large model [25].

• *Convergence rate.* Fig. 8c shows the training curve of both ARM and DDML on the affNIST dataset. We only show the two training curves (of DDML and ARM), because they have the same objective, thus are comparable. From Fig. 8c, we observe that DDML shows a faster convergence rate than ARM. We observe similar convergence properties on all other datasets, and we delegate their results to Appendix.

5 Conclusion

In this paper, we study the problem of domain shift (or distribution shift) in a high-dimensional domain space, namely, the test data may shift significantly from the training data across a high-dimensional data domain, which significantly decreases the average predictive power of the learned model. We propose a domain disentangled meta-learning (DDML) framework that learns an adaptable meta-model from training tasks with a domain disentanglement module, and adapts it to the

domain disentangled representations of test-time data. Our results on three image classification benchmark datasets, including rotated MNIST, affNIST, rotated Tiny ImageNet-C, show promising results, comparing to state-of-the-art baseline approaches in meta-learning and empirical risk minimization.

6 Acknowledgement

Xin Zhang and Yanhua Li were supported in part by NSF grants IIS-1942680 (CAREER), CNS-1952085, CMMI-1831140, and DGE-2021871. Ziming Zhang was supported by NSF grant CCF-2006738. Zhi-Li Zhang was supported in part by NSF grants CNS-1901103 and CCF-221231.

References

[1] J. Q. Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, "Dataset shift in machine learning," *The MIT Press*, vol. 1, p. 5, 2009.

[2] D. Lazer, R. Kennedy, G. King, and A. Vespignani, "The parable of google flu: traps in big data analysis," *Science*, vol. 343, no. 6176, pp. 1203–1205, 2014.

[3] J. C. Duchi, T. Hashimoto, and H. Namkoong, "Distributionally robust losses against mixture covariate shifts," *Under review*, 2019.

[4] M. Sugiyama and A. J. Storkey, "Mixture regression for covariate shift," in *Advances in Neural Information Processing Systems*, pp. 1337–1344, 2007.

[5] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[6] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, pp. 1842–1850, 2016.

[7] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *International Conference on Artificial Neural Networks*, pp. 87–94, Springer, 2001.

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

[9] M. Zhang, H. Marklund, A. Gupta, S. Levine, and C. Finn, "Adaptive risk minimization: A meta-learning approach for tackling group shift," *4th Workshop on Meta-Learning at NeurIPS*, 2020.

[10] T. Tieleman, "Affnist," *URL https://www. cs. toronto. edu/˜ tijmen/affNIST/, Dataset URL https://www. cs. toronto. edu/˜ tijmen/affNIST/.[Accessed on: 2018-05-08]*, 2013.

[11] K. Koutroumbas and S. Theodoridis, "Pattern recognition 2nd edition," 2018.

[12] G. Houghes, "On the mean accuracy of statistical pattern recognition," *IEEE Trans. Inform. Theory*, vol. 14, no. 1, pp. 55–63, 1968.

[13] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[14] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1947–1980, 2018.

[15] N. Dvornik, C. Schmid, and J. Mairal, "Selecting relevant features from a multi-domain representation for few-shot classification," in *European Conference on Computer Vision*, pp. 769–786, Springer, 2020.

[16] L. Liu, W. Hamilton, G. Long, J. Jiang, and H. Larochelle, "A universal representation transformer layer for few-shot image classification," *arXiv preprint arXiv:2006.11702*, 2020.

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

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

[19] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, Lille, 2015.

[20] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," 2017.

[21] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[22] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*, pp. 1180–1189, PMLR, 2015.

[23] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176, 2017.

[24] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," *arXiv preprint arXiv:1710.03463*, 2017.

[25] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, 2020.

[26] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," in *European conference on computer vision*, pp. 467–482, Springer, 2016.

[27] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[28] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks," in *International Conference on Learning Representations*, 2019.

[29] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[30] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.