XDNet: A Few-Shot Meta-Learning Approach for Cross-Domain Visual Inspection

Xian Yeow Lee*, Lasitha Vidyaratne*, Mahbubul Alam, Ahmed Farahat, Dipanjan Ghosh,
Teresa Gonzalez Diaz, Chetan Gupta
Industrial AI Lab, Hitachi America Ltd., R&D
{xian.lee,lasitha.vidyaratne,mahbubul.alam,ahmed.farahat,dipanjan.ghosh,
teresa.gonzalezdiaz, chetan.gupta}@hal.hitachi.com

Abstract

Automated visual inspection has the potential to improve the efficiency and accuracy of inspection tasks across various industries. Deep learning models have been at the forefront of many automated visual inspection technologies. In this work, we focus on a specific instance of a visual inspection problem: the defect detection and classification problem. Training a deep learning model from scratch to detect defects is challenging due to the scarcity of labeled images with defects. Moreover, it is progressively more challenging to adapt a deep learning model across different domains using limited labeled data. We propose a cross-domain meta-learning framework, XDNet, to solve the defect classification problem using a few labeled samples. XDNet is inspired by recent advancements in pre-trained backbone models as general feature extractors and meta-learning frameworks, which adapt across different domains using non-parametric classifiers under limited computational resources. We demonstrate the efficacy of XDNet using a benchmark anomaly detection dataset which we re-formulate as a defect detection and classification problem. Experimental results suggest that XDNet performs significantly better ($\approx$ 17%) than the existing state-of-the-art and baseline models. Additionally, we perform an ablation study to identify the important components that contribute to the improved performance of the proposed framework. Finally, we conduct a data domain-specific analysis to understand the potential strengths and drawbacks of XDNet on different types of defects.

1. Introduction

Visual inspection methods are increasingly being applied in many downstream applications across various industries [1–6]. One important downstream visual inspection task is defect detection and classification, where the task is to identify defects from visual data. Automated defect detection and classification is especially appealing in manufacturing and maintenance settings where manual inspection is often time-consuming, labor-intensive, and error-prone. Hence, automated solutions are crucial to maintaining product quality and efficiently identifying potential issues. The visual defect detection problem is also commonly formulated as an anomaly detection problem if the assumption is that most of the inspections are free of defects and the task is to identify when a defect is present. One of the most prevalent methods for automated defect or anomaly detection is to leverage deep learning-based methods. The combination of advancement in deep learning methods, such as better model architectures and training paradigms, the availability of pre-trained models, and increasingly cheaper computational resources, has all played a major role in the widespread adoption of deep learning-based methods for automated visual inspection.

Nevertheless, training a deep-learning model that achieves acceptable performance is data-intensive. This issue is further exacerbated in industrial settings, where data is often sparse and expensive to obtain, especially if the problem extends beyond just anomaly detection to classifying types of defects. Another major bottleneck of deploying deep learning-based models for defect detection and classification is that traditional deep-learning approaches do not often generalize across domains, i.e., a new set of data needs to be collected to train a model from scratch for every new application domain. For example, a model trained to detect cracks in glass will not be very useful for detecting similar defects in electrical components. This study demonstrates that traditional model adaptation techniques, such as transfer learning, are sub-optimal for such situations. Collecting a new dataset and training a new model is also often time-consuming and computationally expensive. Thus, this further inhibits the adoption of deep learning-based defect

*These authors contributed equally to this work detection and classification methods.

In this work, we propose a cross-domain meta-learning framework that utilizes techniques in meta-learning, unsupervised learning, and feature representation to circumvent the challenges of sparse data and high-compute resource requirements needed to train new deep learning models for defect detection and classification across multiple domains. To demonstrate the efficacy of our proposed method, we use the MVTec dataset [7] as benchmark data and compare it with transfer learning and meta-learning approaches. While the MVTec dataset is commonly used in an anomaly detection setting, we have re-purposed the dataset for a n-way k-shot cross-domain meta-learning approach in this work. We selected the MVTec dataset for our experimentation due to the fact that it consists of data from multiple domains, thus enabling us to validate cross-domain generalization efficacy. Additionally, each domain of data in the datasets also consists of data with multiple classes of defects in an industrial setting that is commonly used as a benchmark in industrial visual inspection tasks. Furthermore, the task of defect classification in this dataset is also especially challenging as many of the defects occur at the pixel scale, with most of the features in the images being largely similar. In contrast, meta-learning approaches are typically evaluated on benchmarks with images from natural domains. Hence, re-purposing the MVTec data also serves to evaluate the effectiveness of meta-learning on defect classification in an industrial setting. As such, the contributions of this work are: **(1)** We propose a n-way, k-shot cross-domain meta-learning framework, XDNet, that leverages squeeze-excitation modules and anti-aliasing filters, contrastive learning and efficient non-parametric classifiers for visual inspection. **(2)** We repurposed an anomaly detection benchmark dataset as a cross-domain industrial defect classification dataset and compare XDNet with common methods for defect detection and classification[1]. **(3)** We show XDNet performs better and is more compute efficient than common transfer learning methods and more robust than a popular meta-learning method. **(4)** We analyze the main components of XDNet that contribute to the overall performance and present a data domain specific study to understand the robustness of XDNet for specific defects.

2. Related Work

In this section, we provide a brief primer on relevant literature. Since the MVTec dataset is a benchmark for anomaly detection, most work developed have followed the setting of anomaly detection, i.e., the model is typically only provided data that has no defects during training, and the model's performance is measured by its capability to detect anomalies during test time when presented with defective data. Among the anomaly detection methods that have been developed are generative methods [8, 9], unsupervised and self-supervised methods [10–12] and transfer learning methods [13, 14]. Contrary to the assumption that no anomalous data is available, we assume that a few samples of data with anomalies can typically be obtained, but with a constraint on the amount of computational resources available to repeatedly train new models for different domains. As such, we frame the problem as a *cross-domain meta-learning* problem, where the objective is to have a model that performs well across multiple domains using only a few labeled examples. A rich amount of work in literature has developed algorithms in the context of meta-learning [15]. The main objective of meta-learning is to develop an algorithm that can adapt to a variety of tasks. Meta-learning approaches can be broadly categorized into optimization-based, metric-learning, and model-based methods, with many approaches belonging to one or more categories. Optimization-based methods operate by re-casting the meta-learning problem as an optimization problem, which can be solved with traditional gradient-based methods. Some examples include Model-Agnostic Meta-Learning (MAML) [16] and Reptile [17]. In contrast, metric learning-based methods typically learn a metric function for each new task and leverage non-parametric techniques for predicting the labels of the test data. Popular examples of metric learning-based methods are Prototypical Networks [18], Matching Networks [19] and Relation Networks [20]. Finally, model-based methods aim to encode the meta-learning process itself into a model for fast adaptation to new tasks and include algorithms such as Memory Augmented Neural Networks [21] and Meta-Networks [22]. Note that while some of the methods described are task-agnostic and can generalize to various tasks (e.g., classification, regression, reinforcement learning), other methods are more specifically designed for classification. Nonetheless, most of these methods have largely been developed in the context of image classification for natural images. As such, these methods have not been benchmarked against industrial-type images with pixel-level defects in a cross-domain setting. Other related areas of research can also be applied for cross-domain defect detection & classification using small or no amount of samples, such as domain adaptation methods [23] and zero-shot learning methods [24]. Nonetheless, each of these methods often comes with its respective constraints. For example, supervised domain adaptation methods often require the domain of transfer to be known *a priori*. Meanwhile, zero-shot learning methods such as CLIP [25] may not generalize well and also require the class labels of new tasks to be described concisely. In defect classification for industrial applications, this requirement is especially restrictive as it is often challenging to

[1]Modified dataset can be found at: https://github.com/xylhal/XDNet-Dataset

accurately describe different types of defects.

3. Methods

3.1. Background and Formulation

Formally, we define the cross-domain meta-learning problem as follows: Let $\mathcal{D}$ define the set of datasets $\{D_1, D_2, D_3 \dots D_m\}$, with D_i, $i = 1 \dots m$ denoting datasets from separate domains. Furthermore, in this work, we focus on a specific instance of meta-learning for classification. Hence, let $\mathcal{T}_j^i$ denote a n-way, k-shot task with data sampled from D_i that consists of a support set and a query set. The support set, $\mathcal{S}_j$, represents labeled data (specifically with k data points from n classes), while the query set $\mathcal{Q}_j$, represents unlabelled data from n classes with an undetermined number of samples. In this work, we also assume that $|S_j| \lll |Q_j|$, though the violation of this assumption does not affect our proposed framework. Hence, the objective of a cross-domain meta-learning framework is to obtain a model $\mathcal{M}$ that is potentially trained on $\mathcal{D}_{\text{Train}} = \{D_1, D_2, D_3 \dots D_k\}$ where $k < m$, and performs well on Q_j when given S_j to adapt on. Note that in this setting, S_j and Q_j are tasks sampled from $\mathcal{D}_{\text{Test}} = \{D_{k+1}, D_{k+2}, D_{k+3} \dots D_m\}$ and $\mathcal{D}_{\text{Train}} \cap \mathcal{D}_{\text{Test}} = \varnothing$

3.2. Framework

In this section, we describe our proposed framework in detail. Fig. 1 illustrates the overall concept of the framework. Inspired by the efficacy of another meta-learning algorithm MetaDelta [26], our framework consists of two main components, an ensemble of pre-trained feature extractors, parameterized with deep neural networks that are trained during the meta-training phase and a meta-learner that adapts to tasks from new domains during inference or deployment. In the setting of cross-domain meta-learning, we hypothesize that the critical component of obtaining a good model is to ensure that the features learned are generalizable across multiple domains without over-fitting to a single domain. During the meta-training phase, we train the feature extractor on $\mathcal{D}_{\text{Train}}$ using a combination of cross-entropy loss and contrastive loss. In practice, we combine the total number of classes in all the datasets in $\mathcal{D}_{\text{Train}}$ and attach a multi-layer perceptron to the feature extractor to enable a classification loss to be obtained. As such, the feature extractor's overall loss is defined as :

$$L = \lambda_1 L_{\text{CrossEntropy}} + \lambda_2 L_{\text{Contrastive}} \quad (1)$$

where λ_1 and λ_2 denote the weighting factors of the cross-entropy and contrastive loss respectively. Furthermore, we use the triplet-margin loss as an instantiation of the contrastive loss. In terms of the neural network architecture, we have empirically found that a ResNext101 architecture with Squeeze Excitation Blocks [27] and anti-aliasing filters [28] works the best.

In the context of training a feature extractor that is capable of extracting generalizable features, we believe that the contrastive loss, squeeze-excitation modules, and anti-aliasing filters can all contribute to prevent the model from learning features that over-fit to the data in $\mathcal{D}_{\text{Train}}$. To improve generalizability, we leverage an ensembling technique such that the extracted feature maps (embeddings) are further regularized. To circumvent the computational cost of training multiple deep neural networks, we use snapshot ensembling [29] to save snapshots of the weights of the top-m performing versions of the models on the validation set during meta-training.

Next, we describe the components found in the meta-testing phase (adaptation phase). The efficacy of a meta-learning algorithm is validated in this phase, where the model is presented with a small amount of labeled data from the support set that is different from the domains of the meta-training data and is scored based on its performance on the query set. We pass the images from the support set through the ensemble of feature extractors to obtain multiple embeddings and concatenate them together. These concatenated embeddings are then passed to a parameter-less soft k-means-based transductive decoder as used in [30, 31]. The decoder iteratively generates the prototypes of the support set embeddings based on the class labels to produce a distribution of Euclidean distance over the output classes for a given query sample. To further generate more separable prototypes and subsequently better assignment of the query samples, we use the self-optimal transport (SOT) feature transform [32] to augment the concatenated embeddings before applying the iterative soft k-means. Our hypothesis here in using a parameter-less method for classification is that the model will less likely over-fit to the support set's labeled embeddings as compared to a parameterized model, and the SOT will further facilitate the assignment of query samples to prototypes. While using a parameter-less decoder and techniques such as SOT is not novel in itself, we hypothesize that including such techniques in conjunction is critical in cross-domain meta-learning, especially for defect classification applications where the classes in a task often contain highly overlapping features.

3.3. Experimental setup

To establish the efficacy of our framework, we use the MVTec benchmark data [7] as an instance of a defect classification problem. As the MVTec dataset is originally designed as an anomaly segmentation task, we make several modifications to pose it as a benchmark for cross-domain defect detection and classification problem. The MVTec consists of 15 unique datasets of textures and objects, with each dataset having training data with only defect-free images and testing data with defect-free and (mostly multi-class) defective images. We select 12 of the 15 datasets

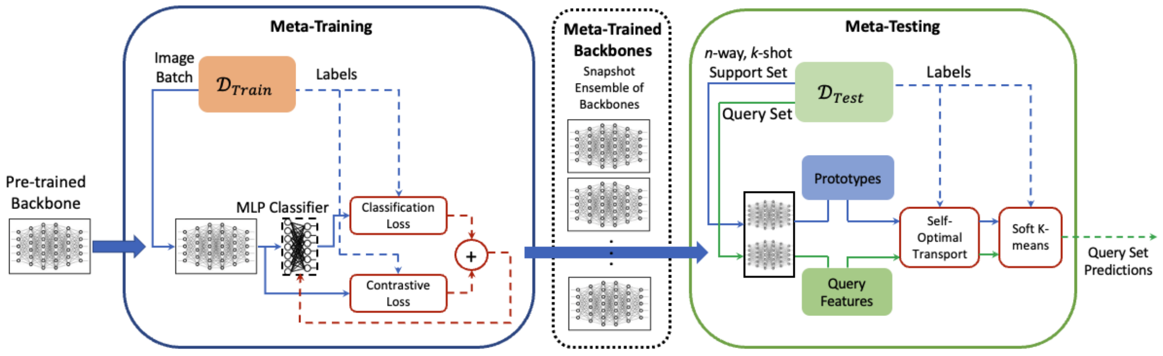

Figure 1. Illustration of the proposed XDNet for few-shot cross-domain meta-learning.

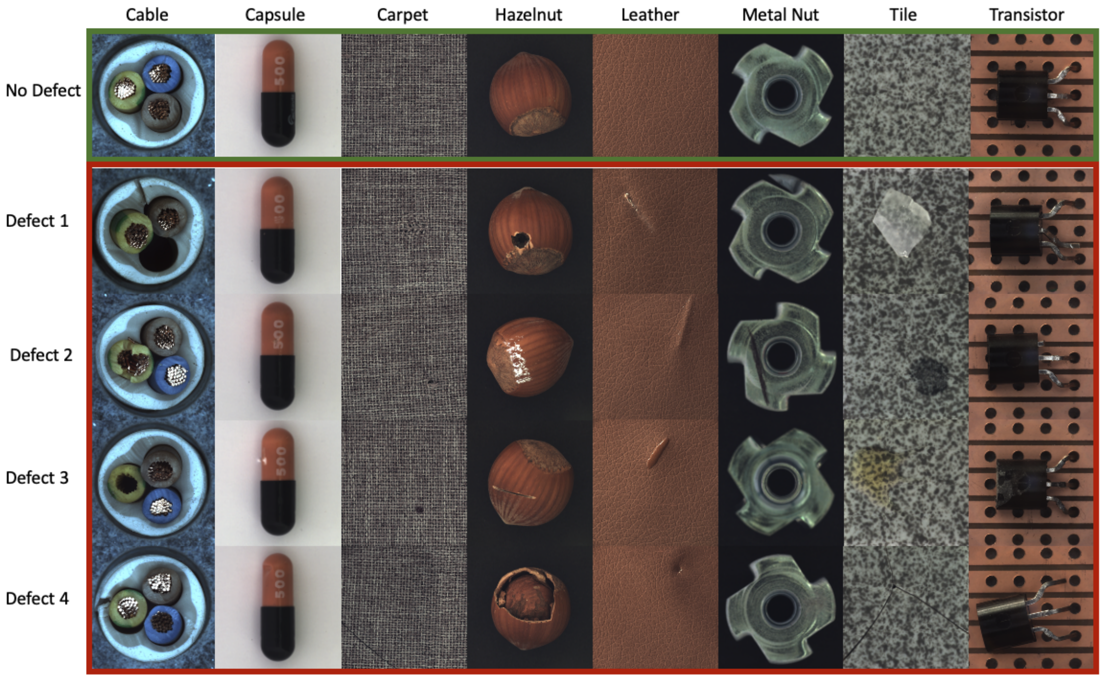

Figure 2. Some example images of the MVTec data sampled from $\mathcal{D}_{\text{Test}}$ to generate tasks during meta-testing phase. The top row represents images with no defects while the bottom five rows represents images with defects.

and discard the datasets with only gray-scale images as we conduct our experiments using colored (RGB) images for consistency across dataset domains. To utilize this dataset as a cross-domain meta-learning benchmark, we randomly split the list of datasets into $\mathcal{D}_{\text{Train}}$ and $\mathcal{D}_{\text{Test}}$. During the meta-training phase, we concatenate the defect-free and defective data together from all the datasets in $\mathcal{D}_{\text{Train}}$ and train the model via the loss in Eq. 1. During the meta-testing phase, for every dataset in $\mathcal{D}_{\text{Test}}$, we sample a fixed number of randomly generated tasks, with the support set in each task having two to five classes, $n \in [2, 5]$ and having one to five labeled data points per class, i.e., $k \in [1, 5]$. Additionally, for every task, we ensure that the query set also has the same n-classes with the corresponding support set and five unlabeled data points per class, $|Q_j| = 5$, to evaluate the accuracy of the framework.

To demonstrate the efficacy of XDNet, we compare XDNet with several popular transfer learning and meta-learning methods. As one of the main constraints in this work is the computational cost, we select only algorithms with comparable computational costs. To reduce the overall meta-training time, we initialize the feature extractors with weights from models trained on ImageNet and only update the weights of the final two layers of the feature extractor during meta-training while keeping the rest of the layers frozen.[2]

Training from scratch: As a preliminary baseline, this method uses a single backbone feature extractor initialized with weights trained on ImageNet and directly trains the model on tasks sampled from $\mathcal{D}_{\text{Test}}$. In this setting, only the final two layers of the feature extractor and a task-specific multi-layer perceptron layer are trained for each task using a classification loss during the meta-testing phase, and there is no meta-training phase.

Transfer Learning: This baseline is architecturally similar to training from scratch, with the exception that the model undergoes a meta-training phase on $\mathcal{D}_{\text{Train}}$ first before being evaluated on the adaptation capability on $\mathcal{D}_{\text{Test}}$. Hence, in this setting, we study if pre-training the model on industrial images from different domains is beneficial compared to just using the initial weights from natural images.

Prototypical Networks: Protonet trained in an episodic manner during the meta-training phase; i.e., the model is trained sequentially on tasks randomly sampled from any single domain in $\mathcal{D}_{\text{Train}}$, in contrast to training on the combination of all the datasets in $\mathcal{D}_{\text{Train}}$. During meta-training, the model generates prototypes of each class (by projecting the support set into l-dimensional vectors and computing class-specific mean vectors). The model is then trained to maximize the probability of the query set being assigned to the true class. Specifically, the model is trained to learn generalizable representations that map the support set to these prototypes such that the distances between the query set and the prototypes correspond well to the class membership of the query set. During meta-testing, the class of a query sample is determined by aggregating the distances between the query sample and the prototypes for each class and choosing the class with the minimum distance.

XDNet: The proposed framework in this paper. In summary, our method consists of two backbone models acting as feature extractors trained with the contrastive and supervised loss on $\mathcal{D}_{\text{Train}}$. During meta-testing, we use the two models to extract feature vectors from the data sampled from support and query set in $\mathcal{D}_{\text{Test}}$, concatenate the feature vectors, and post-process using self-optimal feature transform. The transformed feature vectors of the support set are used to form the prototypes, and the query set is classified using the iterative soft k-means decoder.

4. Results

4.1. Method comparisons

We first present the results of the baselines and XDNet. Fig. 3 illustrates the comparison of the performance across 14 different random folds of $\mathcal{D}_{\text{Train}}$ and $\mathcal{D}_{\text{Test}}$. In each fold, $\mathcal{D}_{\text{Test}}$ consists of datasets from four different domains, while the rest of the datasets are included in $\mathcal{D}_{\text{Train}}$. These 14 folds have been generated to ensure that there are no more than two overlapping datasets in $\mathcal{D}_{\text{Test}}$ between all the folds. Additionally, we have also reported the average normalized accuracy in the vertical axis of Fig. 3. The normalized accuracy is commonly used to measure performance in n-way, k-shot tasks [33] and accounts for the fact that the tasks generated during meta-testing time may have varying values of n and k. Formally, the normalized accuracy is defined as:

$$\text{Acc}_{\text{normalized}} = \frac{\text{Acc}_{\text{class}} - \text{Acc}_{\text{rc}}}{1 - \text{Acc}_{\text{rc}}} \tag{2}$$

where $\text{Acc}_{\text{class}}$ is the average accuracy per class of a specific task and Acc_{rc} is the accuracy of a uniform random classifier that depends on the number of classes (n-way) present in a specific task. As such, a normalized accuracy of 0 denotes a model that is on par with a random classifier, while a normalized accuracy > 0 represents a performance that is better than a random classifier. In our experiments, during meta-testing, we have set the number of n-way, k-shot tasks generated for each domain in $\mathcal{D}_{\text{Test}}$ to be 50, resulting in a total of 200 tasks per fold.

From Fig. 3, we can observe that XDNet consistently outperforms all the other baselines across all random folds. Averaged across all folds, our method performs the best, followed by Protonet, transfer learning, and finally, training from scratch. Although Protonet performs the second best on average, we can also observe that this is not the case across all folds, specifically in Folds 4, 10, and 13, where it performs worse than the other two baselines. Thus, we conclude that, while on average, Protonet may be a better solution than training from scratch or transfer learning, it is also highly sensitive to the domain of data it is adapting to. Considering that the same backbone is used between Protonet and our framework, we hypothesize that this could be attributed to Protonet being trained in an episodic mode and only using features extracted from a single feature extractor.

Comparing transfer learning versus training from scratch, we observe that transfer learning is also consistently better than or as good as the strategy of learning from scratch, with the exception of Fold 7. Recall that in our implementation, both transfer learning and learning from scratch models are initialized with model weights trained on ImageNet and would already have learned robust image features. This observation highlights that there are potential benefits to meta-training a model on datasets that may belong to the same larger overarching domain of data (objects with defects from various domains in this case), despite the downstream tasks being from different domains (such as defects in cables, capsules, transistors, etc.).

Finally, we tabulate the average amount of time required per task for the different methods to adapt and classify the

[2]All the experiments were conducted on a workstation with a six-core Intel CPU and an NVIDIA TITAN Xp GPU with 12 GB memory.

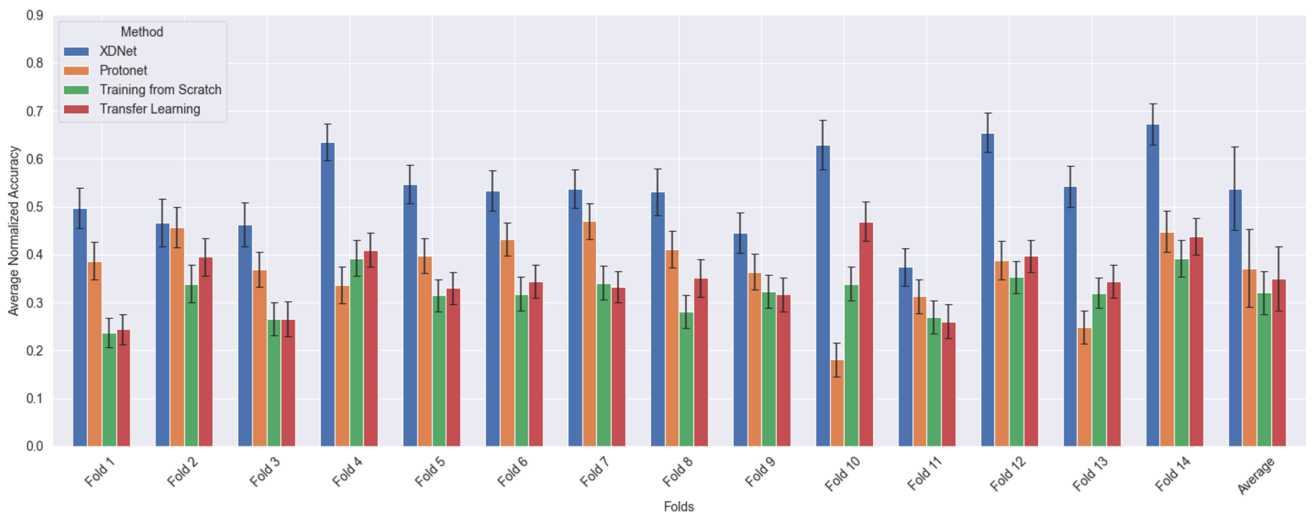

Figure 3. Comparison of our proposed framework with the other three baseline methods across 14 random folds of $\mathcal{D}_{\text{Test}}$. Our proposed method consistently outperforms the rest of the baselines across all folds.

Method	Average Time/Task (s)
Training from scratch	12.57
Transfer learning	12.52
Protonet	2.58
XDNet	2.97

Table 1. Time per task of each method during meta-testing phase, averaged over 200 tasks. While XDNet is slightly slower than Protonet, the performance gain is significantly and consistently stronger, as shown in Fig. 3.

defects when presented with support and query sets from new domains. As observed in Table 1, training from scratch and transfer learning requires an order of magnitude higher amount of time, since both methods essentially have to train one or more parameterized layers. In contrast, Protonet and XDNet are significantly faster due to the classification being computed by a parameter-less approach. While XDNet is slightly slower than Protonet, we believe it is worth trading off a small amount of computational efficiency for the additional robustness and accuracy performance of XDNet.

4.2. Ablation study

Next, we perform an ablation study on the proposed method to isolate and identify the components that contribute to the performance of the framework most significantly. Specifically, we repeat the experiments across all folds of data by removing the following components:

1. **Squeeze-excitation architecture and anti-aliasing filters**: This experiment forces the backbone feature extractors to default to the basic ResNet101 architecture without the squeeze-excitation modules and anti-aliasing filters.
2. **Self-optimal feature transform**: This ablation removes the SOT feature transform during the meta-testing phase and forwards the raw embeddings extracted by the ensemble of backbone feature extractors directly to the soft k-means-based transductive decoder to generate the prototypes and classify the query set.
3. **Ensemble of feature extractors**: This experiment removes the snapshot ensembling technique during the meta-training phase. As a result, only one feature extractor is used during the meta-testing phase to extract the features from the input.
4. **Contrastive loss**: This ablation removes the contrastive loss term during the meta-training phase of the feature extractors and only trains the feature extractors using the classification loss.

From Fig. 4, we observe that, on average, removing the squeeze-excitation modules and anti-aliasing filters results in the largest drop in performance, followed by ensembling, contrastive loss, and the use of SOT. Nevertheless, we observe no consistent trends across the 14 random folds of $\mathcal{D}_{\text{Test}}$. On the contrary, removing certain components sometimes resulted in an overall better performance. For example, removing the contrastive loss significantly improved the normalized accuracy for Folds 2, 6, and 7, while the presence of the contrastive loss improved performance in Folds 1, 10, and 12.

4.3. Domain-specific analysis

In this section, we present the dataset-specific normalized accuracy for every fold of the data to analyze the effects of different domains of data on the generalizability of

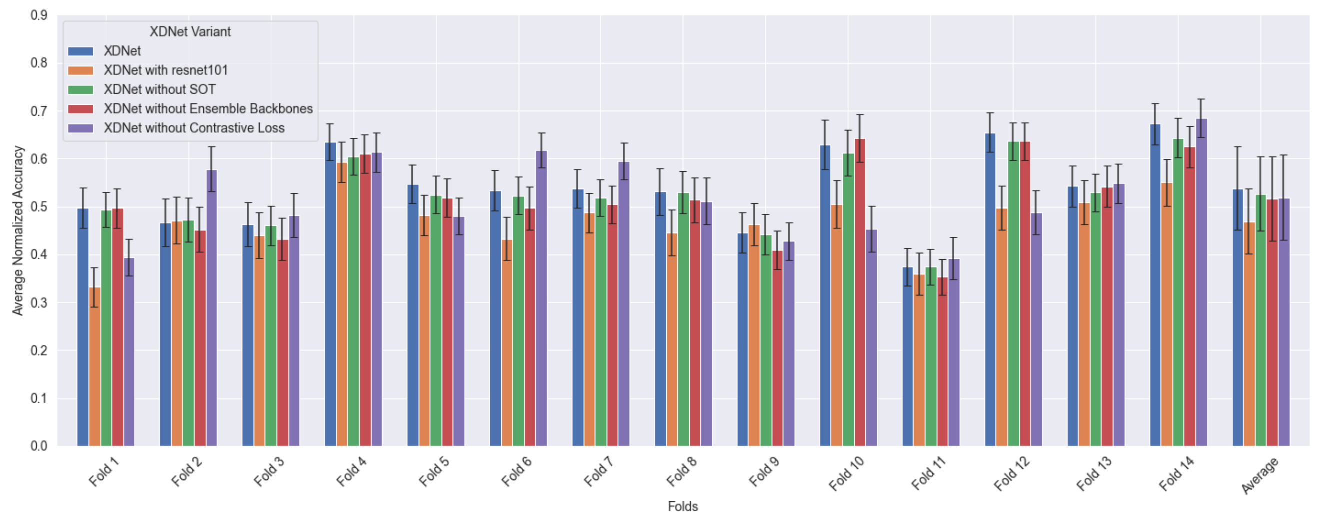

Figure 4. Ablation study to identify the main components that contribute to the performance of XDNet. The squeeze-excitation modules and anti-aliasing filters have the most significant effect, followed ensembling, contrastive loss during meta-training and SOT.

	Cable	Capsule	Carpet	Hazelnut	Leather	Metal Nut	Tile	Transistor
Fold 1	0.58 ± 0.06	0.28 ± 0.08	0.61 ± 0.07	0.50 ± 0.07	-	-	-	-
Fold 2	0.47 ± 0.07	0.21 ± 0.08	-	-	0.80 ± 0.05	0.37 ± 0.10	-	-
Fold 3	0.48 ± 0.08	0.22 ± 0.06	-	-	-	-	0.76 ± 0.06	0.37 ± 0.08
Fold 4	0.50 ± 0.06	-	0.43 ± 0.08	-	0.84 ± 0.04	-	0.76 ± 0.05	-
Fold 5	0.58 ± 0.06	-	0.62 ± 0.08	-	-	0.53 ± 0.08	-	0.44 ± 0.08
Fold 6	0.53 ± 0.06	-	-	0.36 ± 0.05	0.84 ± 0.04	-	-	0.38 ± 0.09
Fold 7	0.56 ± 0.06	-	-	0.33 ± 0.05	-	0.46 ± 0.07	0.78 ± 0.06	-
Fold 8	-	0.20 ± 0.08	0.60 ± 0.07	-	0.85 ± 0.04	-	-	0.45 ± 0.08
Fold 9	-	0.20 ± 0.08	0.42 ± 0.07	-	-	0.45 ± 0.07	0.70 ± 0.05	-
Fold 10	-	0.19 ± 0.08	-	0.56 ± 0.07	0.86 ± 0.04	-	0.89 ± 0.04	-
Fold 11	-	0.24 ± 0.08	-	0.39 ± 0.07	-	0.41 ± 0.07	-	0.45 ± 0.09
Fold 12	-	-	0.60 ± 0.08	0.65 ± 0.07	0.86 ± 0.04	0.49 ± 0.08	-	-
Fold 13	-	-	0.53 ± 0.07	0.33 ± 0.06	-	-	0.80 ± 0.06	0.49 ± 0.08
Fold 14	-	-	-	-	0.86 ± 0.04	0.49 ± 0.09	0.80 ± 0.07	0.52 ± 0.07
Average	0.53 ± 0.04	0.22 ± 0.03	0.54 ± 0.08	0.45 ± 0.12	0.84 ± 0.02	0.46 ± 0.05	0.78 ± 0.05	0.44 ± 0.05

Table 2. Averaged normalized accuracy and standard deviation for individual data domains within $\mathcal{D}_{\text{Test}}$ for all folds.

	Cable	Capsule	Carpet	Hazelnut	Leather	Metal Nut	Tile	Transistor
Training from scratch	0.28 ± 0.00	0.13 ± 0.01	0.28 ± 0.02	0.29 ± 0.01	0.48 ± 0.03	0.37 ± 0.03	0.45 ± 0.02	0.25 ± 0.04
Transfer learning	0.28 ± 0.03	0.13 ± 0.02	0.30 ± 0.04	0.35 ± 0.08	0.58 ± 0.05	0.39 ± 0.04	0.48 ± 0.05	0.26 ± 0.02
Protonet	0.40 ± 0.05	0.24 ± 0.03	0.29 ± 0.08	0.33 ± 0.18	0.52 ± 0.14	0.40 ± 0.06	0.49 ± 0.18	0.26 ± 0.06
XDNet	0.53 ± 0.04	0.22 ± 0.03	0.54 ± 0.08	0.45 ± 0.12	0.84 ± 0.02	0.46 ± 0.05	0.78 ± 0.05	0.44 ± 0.05

Table 3. Comparison of domain specific average normalized accuracy between our proposed method and the other baselines.

XDNet. The average normalized accuracy across all tasks $\mathcal{D}_{\text{Test}}$ for each dataset and each fold is shown in Table 2. The first observation we make is that the normalized accuracy of XDNet for every domain remains largely consistent across all folds of $\mathcal{D}_{\text{Test}}$ despite having different n-way, k-shot tasks being generated during meta-testing. We observe that the model performs particularly well on domains such as leather and tile while performing moderately better than a random classifier in the cable, carpet, hazelnut, metal nut, and transistor datasets. Most notably, the model performs just marginally better than a random classifier in the capsule dataset. Additionally, the top three domains in which XDNet performed well (leather, tile, and carpet) are all from textured domains, while the rest of the domains consist of an object with a background (as seen in Fig. 2). We hypothesize that domains with objects often consist of more prominent features, thus making the task of classifying different defects more challenging.

In Table 3, we compare the domain-specific average normalized accuracy of the proposed framework against the other baselines that we implemented. Once again, we observe that XDNet consistently outperforms the rest of the baseline methods across all domains of data. Interestingly, our hypothesis that defects in texture domains are less challenging to classify does not hold as strongly for the rest of the methods. Specifically, the leather and tile datasets still remain as the top two domains in which the other methods perform the best. Nevertheless, the carpet dataset is not the third-best domain for the rest of the baselines. This observation suggests that while the images in the carpet dataset are textured, the defects are significantly more challenging to identify as compared to defects in tile and leather. Referring to Fig. 2 once more, we can confirm that the defects in the carpet are visually less obvious when compared to defects in the other two texture datasets.

4.4. Effects of k and n on performance

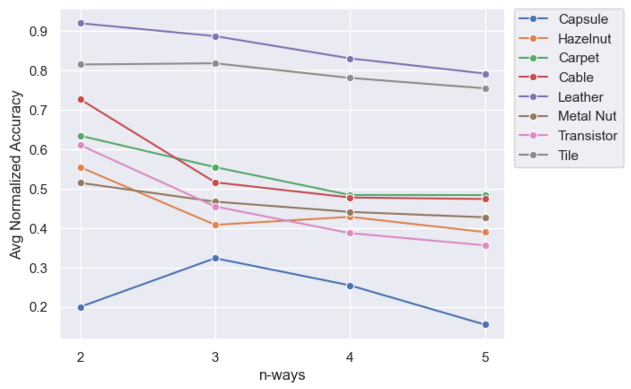

Figure 5. Effect of increasing n in the support set during the meta-testing phase on the average normalized accuracy of XDNet.

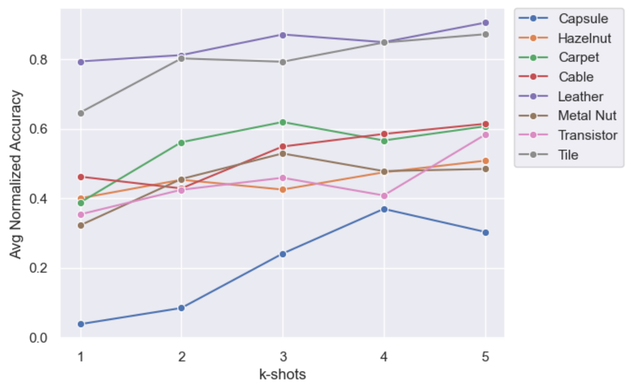

Figure 6. Effect of increasing k in the support set during the meta-testing phase on the average normalized accuracy of XDNet.

Finally, we present analysis on the effects of n and k on the average normalized accuracy of XDNet. Fig. 5 illustrates the performance of XDNet on each domain aggregated over all folds of $\mathcal{D}_{\text{Test}}$ when n in the task increases. As expected, almost all domains exhibit a monotonic decrease in performance when the value of n in a task increases, with the exception of the hazelnut and capsule datasets. However, the overall trend still shows a decrease as n increases for these domains. Another observation is that for the leather and tile domains, the model's performance is relatively robust to an increasing value of n when compared to other domains, where the model performance experiences a relatively sharper decrease when n increases. In Fig. 6, we show the effect of increasing the number of label examples in the support set during the meta-testing phase (averaged over all values of n) on the average normalized accuracy of XDNet. While the model displayed a trend that shows performance increases with the number of labeled examples, the overall trend is less monotonic than expected. In other words, in this regime of few-shot samples ($k \in [1, 5]$), the model is still sensitive to the number of labeled examples, and increasing the size of the support set with 1 or 2 samples does not necessarily improve overall performance. Furthermore, we also observe that for the capsule dataset, increasing k from 2 to 4 results in a significant gain in normalized accuracy, while the same phenomenon is not observed in other domains. We hypothesize this is because when k is small, the prototypes generated by the soft k-means-based decoder are relatively noisy, especially when n is high. The results from Fig. 5 and 6 suggest that when deploying the framework to a new domain, it is imperative to consider that each domain may have a unique number of required k in order to observe a significant increase in performance. Future work will also further investigate the effects of k in higher regimes to determine if monotonic improvements can be observed.

5. Conclusion

This work leverages advancements in unsupervised learning, model architectures, training strategies, and feature processing methods to propose a cross-domain meta-learning framework for defect detection and classification tasks. We demonstrate the efficacy of our approach with several baselines on MVTec dataset. This work reformulates the defect detection problem of MVTec into a more challenging multi-class classification problem to investigate the n-way k-shot performance of XDNet. We show that XDNet is more effective than other baselines in cross-domain settings with few-shot examples. Despite performing better than other baselines, classification of the defects in several domains remain challenging. Additionally, we performed an ablation study and discover that while most of the advanced techniques contribute to an increase in performance, changes to the model's architecture has the most significant effect. Future directions include incorporating more advanced backbone feature extractors specifically tailored towards identifying pixel-level defects while reducing the computational complexity of the method during meta-testing.

References

[1] X. Sun, J. Gu, S. Tang, and J. Li, "Research progress of visual inspection technology of steel products—a review," *Applied Sciences*, vol. 8, no. 11, p. 2195, 2018. 1

[2] R. Jenssen, D. Roverso *et al.*, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *International Journal of Electrical Power & Energy Systems*, vol. 99, pp. 107–120, 2018. 1

[3] S. Liu, Q. Wang, and Y. Luo, "A review of applications of visual inspection technology based on image processing in the railway industry," *Transportation Safety and Environment*, vol. 1, no. 3, pp. 185–204, 2019. 1

[4] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ digital medicine*, vol. 4, no. 1, p. 5, 2021. 1

[5] L. Zhu, P. Spachos, E. Pensini, and K. N. Plataniotis, "Deep learning and machine vision for food processing: A survey," *Current Research in Food Science*, vol. 4, pp. 233–249, 2021. 1

[6] Y. D. Yasuda, F. A. Cappabianco, L. E. G. Martins, and J. A. Gripp, "Aircraft visual inspection: A systematic literature review," *Computers in Industry*, vol. 141, p. 103695, 2022. 1

[7] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9592–9600. 2, 3

[8] T.-W. Tang, W.-H. Kuo, J.-H. Lan, C.-F. Ding, H. Hsu, and H.-T. Young, "Anomaly detection neural network with dual auto-encoders gan and its industrial inspection applications," *Sensors*, vol. 20, no. 12, p. 3336, 2020. 2

[9] V. Zavrtanik, M. Kristan, and D. Skočaj, "Draem-a discriminatively trained reconstruction embedding for surface anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8330–8339. 2

[10] B. A. ugli Olimov, K. C. Veluvolu, A. Paul, and J. Kim, "Uzadl: Anomaly detection and localization using graph laplacian matrix-based unsupervised learning method," *Computers & Industrial Engineering*, vol. 171, p. 108313, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835222003709 2

[11] S. Yoa, S. Lee, C. Kim, and H. J. Kim, "Self-supervised learning for anomaly detection with dynamic local augmentation," *IEEE Access*, vol. 9, pp. 147 201–147 211, 2021. 2

[12] J.-C. Wu, D.-J. Chen, C.-S. Fuh, and T.-L. Liu, "Learning unsupervised metaformer for anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 4369–4378. 2

[13] O. Rippel and D. Merhof, "Leveraging pre-trained segmentation networks for anomaly segmentation," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 01–04. 2

[14] O. Rippel, A. Chavan, C. Lei, and D. Merhof, "Transfer learning gaussian anomaly detection by fine-tuning representations," *CoRR*, vol. abs/2108.04116, 2021. [Online]. Available: https://arxiv.org/abs/2108.04116 2

[15] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021. 2

[16] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135. 2

[17] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018. 2

[18] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017. 2

[19] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016. 2

[20] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208. 2

[21] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1842–1850. 2

[22] T. Munkhdalai and H. Yu, "Meta networks," *CoRR*, vol. abs/1703.00837, 2017. [Online]. Available: http://arxiv.org/abs/1703.00837 2

[23] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018. 2

[24] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019. 2

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763. 2

[26] Y. Chen, C. Guan, Z. Wei, X. Wang, and W. Zhu, "Metadelta: A meta-learning system for few-shot image classification," in *AAAI Workshop on Meta-Learning and MetaDL Challenge*. PMLR, 2021, pp. 17–28. 3

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141. 3

[28] R. Zhang, "Making convolutional networks shift-invariant again," in *International conference on machine learning*. PMLR, 2019, pp. 7324–7334. 3

[29] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, “Snapshot ensembles: Train 1, get m for free,” *arXiv preprint arXiv:1704.00109*, 2017. 3

[30] Y. Chen, C. Guan, Z. Wei, X. Wang, and W. Zhu, “Metadelta: A meta-learning system for few-shot image classification,” in *AAAI Workshop on Meta-Learning and MetaDL Challenge*. PMLR, 2021, pp. 17–28. 3

[31] S. M. Kye, H. B. Lee, H. Kim, and S. J. Hwang, “Meta-learned confidence for few-shot learning,” *arXiv preprint arXiv:2002.12017*, 2020. 3

[32] D. Shalam and S. Korman, “The self-optimal-transport feature transform,” *arXiv preprint arXiv:2204.03065*, 2022. 3

[33] D. Carrión-Ojeda, H. Chen, A. El Baz, S. Escalera, C. Guan, I. Guyon, I. Ullah, X. Wang, and W. Zhu, “Neurips’22 cross-domain metadl competition: Design and baseline results,” in *ECMLPKDD Workshop on Meta-Knowledge Transfer*. PMLR, 2022, pp. 24–37. 5