# Domain Generalization in Numerical Datasets

Christopher Brokenshire
College of William and Mary
Email: cmbrokenshire@wm.edu

Anna Seiple
College of William and Mary
Email: asieple@wm.edu

Harry Choi
College of William and Mary
Email: hpchoi@wm.edu

## I. Introduction

The field of machine learning has made significant strides, particularly in image classification, a key area of focus for data scientists. This journey began with the creation of the MNIST dataset, comprising 60,000 hand-drawn digits, which laid the foundation for developing models capable of classifying relatively simple images. Over time, these models have evolved, achieving greater sophistication and accuracy, especially with the introduction of Convolutional Neural Networks (CNNs). This advancement led to groundbreaking performance, with classification accuracies reaching as high as 99.14 percent, a milestone marked by the development of models like LeNet.

However, despite these successes, a persistent challenge has been the ability of these models to generalize. When models trained on the MNIST dataset are tested on other similar datasets, such as USPS or SHVN, their performance significantly deteriorates. This issue underscores a fundamental limitation in current machine learning models: their inability to adapt to different representations of the same underlying concept, such as numerals presented in varying styles. The question that emerges is whether it is possible to develop models capable of adapting to new, unseen data. Addressing this challenge, the paper "Adversarial Discriminate Domain Adaptations" proposes an innovative solution. It suggests a methodology for building adaptable models that can transfer their learning from a known (source) domain to correctly classify data in an unknown (target) domain, even in the absence of labels. This approach marks a significant advancement in the pursuit of creating more flexible and robust machine learning models, capable of handling the diverse and dynamic nature of real-world data as we see in healthcare or even sports analytics.

In this paper, we will discuss the team's implementation of the ADDA model, following a rough code outline discovered online. Our discussion will cover our journey in understanding the conceptual ideas of the original paper, as well as the LeNet architecture. Our aim is to acquire a comprehensive understanding necessary to resolve the existing issues. The ultimate objective is to determine whether we can effectively adapt the USPS dataset for accurate classification by a model trained on the MNIST dataset.

## II. Adversarial Discriminative Domain Adaptation

Before breaking down the paper's solution, it is important to first define encoders and classifiers. An encoder is a component of a neural network that essentially encodes input data based on extracted features such that a classifier can then take that encoding and map it to an output class. For example, in the case where the input data is images, the encoder transforms the images into an abstract representation that attempts to capture the essential information that helps distinguish different pictures. Subsequently, the classifier receives the encoder's output and predicts the given classes of the input image based on the encoding.

Now transitioning to ADDA, with respect to digit image data, the whole premise is to develop models that are trained on one domain (source) of images and can still robustly classify images from a different domain (target) that does not contain labels. This can only be accomplished by aligning the source and target encoders to produce the same output for the same number. For example, let's say we have an image of a '3'. Ideally, the mapping of the source and target encoders should both be similar, so that the image is properly classified as a 3 by the classifier. However, if the '3' from the target domain has a completely different representation than the '3' from the source domain, it is unlikely that the respective encoders will produce similar encodings leading to misclassification. Thus, we want to minimize variations between the style in which the domains are encoded so the classifier can classify the representations of each digit, regardless of how different their representations original were. So how do we do this?

As the paper's authors write, domain adaptation is accomplished through adversarial training. After the pre-training of a source encoder CNN and the introduction of a target encoder, we introduce a domain discriminator. A discriminator is a neural network that attempts to distinguish which domain the images (representations) are from. Additionally, an adapter is a neural network that dynamically adjusts internally when there are changes to input data. Therefore, since the target encoder is an adaptive neural network, it learns to reduce the domain discriminator's ability to differentiate between source and target images; or, in other words, it learns to disguise target representations so that the discriminator is fooled into believing it is from the source encoder. [1]

The target encoder and domain discriminator are essentially pitted against each other, for they are trained simultaneously to optimize two competing objectives as you can see in the loss functions below.

Before continuing, let's lay out the key terms in both loss functions. Xs and Xt represent the source and target images,

[1]Eric Tzeng et al., "Adversarial Discriminative Domain Adaptation," arXiv:1702.05464 (2017): 3, https://arxiv.org/pdf/1702.05464.pdf.

$$
\begin{aligned}
\min_{D} \mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) = \\
&- \mathbb{E}_{\mathbf{x}_s \sim \mathbf{X}_s}[\log D(M_s(\mathbf{x}_s))] \\
&\quad - \mathbb{E}_{\mathbf{x}_t \sim \mathbf{X}_t}[\log(1 - D(M_t(\mathbf{x}_t)))] \\
\min_{M_s, M_t} \mathcal{L}_{\text{adv}_M}(\mathbf{X}_s, \mathbf{X}_t, D) = \\
&- \mathbb{E}_{\mathbf{x}_t \sim \mathbf{X}_t}[\log D(M_t(\mathbf{x}_t))].
\end{aligned}
$$

Fig. 1. Discriminator and Target Encoder Loss Functions

respectively; Ms and Mt are the encodings produced from each domain's respective encoder; and D is the discriminator that determines if the encoding are from the source encoder or not.[2]

The way adversarial training works is that iterates back and forth between optimizing the two loss functions. The target encoder learns to ”disguise” its encodings as coming from the source domain. On the other hand, the discriminator is optimized to correctly identify the encodings' respective encoder. In other words, the target encoder is trained to better fool the discriminator, while conversely the discriminator is updated to better identify the encoding discrepancies. Ideally, this adversarial process converges when the encoding representations from the target encoder are indistinguishable from the source encoders' representations, thus the discriminator can't distinguish between encodings of the target and source domains anymore. At this point, ADDA finally uses the source classifier, which was fixed and trained on the source encoder at the beginning, to make accurate predictions on the target domain with the adapted target encoder.[3]

## III. LeNet Model

Upon initial analysis of the base code, there were several bugs, mainly related to indexing errors, which had to be sorted through in order to completely run through the program. Once those bugs were fixed, the program was able to run end to end. However, the accuracy was extremely low, being only 6 percent for predicting figures from the USPS dataset. This indicated the need for tuning the neural networks of both the encoders and discriminator, however to do so properly we must first understand the intricacies of the architecture of each neural network.

The basic structure of the discriminant neural network is a set of fully connected layers. The NN begins by using the pytorch library to map the input to the first hidden layer, later applying ReLU to the hidden layers' output to create nonlinearity. It repeats this process twice lastly calling the LogSoftMax function on the output to normalize the results of the network to a probability distribution for the classification items.

Both the target and source encoders are convolutions neural networks, meaning they are built to learn image features extracted using convolutions.

The goal of a convolution is to extract the important features of an image to create feature maps. This is done by applying a kernel filter to the image, which slides around portions of the image, to detect the prominent qualities of the image. These kernels can be designed to detect different patterns, like edges, curves, or sharpening the image. Typically the convulution layer will apply several kernels to the input image, each kernel acting a feature map. Similar to a typical neural net, we then apply a ReLU function to these feature maps.

Once this convolution layer has outputted a set of feature maps, the next layer consists of max pooling.

Max pooling is used to identify the most important features within a set of feature maps, thus reducing the dimensions of the output. This is implemented by dividing the feature map into sections, based on input parameters, and selecting the most prominent features within each of those sections. The important features are detected by finding the values with the maximum activation value within each section. These values are then outputted creating a feature map which has smaller dimensions, yet has still retained the most needed features of the image.

These layers alternate, extracting prominent patterns within the image, until it outputs with a fully connected layer whose output is 500 dimensional vector which is the encoding the classifier uses to map the image to a class.

## IV. What We Did

As we can see from the results, the target encoder has not been able to learn a transformation on the target sets which allows the source classifier to correctly classify it. When we ran the test, we saw the discriminator was quite dominant. In such a case, the adapter is not really able to learn, since it is not able to extract viable information from the discriminator's gradients on how to adjust its weights to better be able to fool it. We reasoned that this was because the discriminator's NN ended with a LogSoftMax, which squishes all the inputs to between 0 and 1. When the discriminator makes a decision with very high confidence, it means the output of the activation function is near the extremes. At the extremes, the slope of the function is very small, which means very uninformative for the adaptive NN. As a result, at the beginning of the learning process when the adaptive encoder is just trying out random encoding, the discriminator is rejecting with high confidence and the gradient is very uninformative for the adaptive encoder so it begins falling behind. So we began by removing the LogSoftMax.

Although we saw some improvement, it was not substantial. There was something else going on.

The solution lies in the relationship between the source encoder and the target encoder. We want the target encoder to mimic the source encode. It does so by trial and error, yet how can we make the source encoder as mimicable as possible, and how can we make the source encoder's output reachable by the adaptive encoder? We do so by generalization.

An example may give a good explanation of why we may want to generalize the encoder. Let's consider two images of a three, one from the USPS data set and one from the MNIST dataset. Both represent the same abstract object, a three. When

[2]Tzeng et al., ”Adversarial Discriminative Domain Adaptation,” 3.

[3]Tzeng et al., ”Adversarial Discriminative Domain Adaptation,” 6.

we generalize an encoder, we make it so that it captures the domain invariant descriptions of a three, descriptions of the abstract object. As a result, when we want to make it such that the source encoder is generalized, its encoding depends on the domain invariant description of a MNIST three. When we use adversarial learning to train the target encoder, all the target encoder needs to do now is capture the domain invariant description of three in the USPS dataset. Since the domain invariant description of three is the same for all its representations, the better the target encoder is at abstracting the three, the closer it will get to the MNIST encoding. This methodology gives a clear path forward for the target encoder to fool the discriminator.

How do we generalize both the source and target encoder? We use two techniques: Batch Normalization and Dropout.

Batch Normalization works by normalizing the output of each layer's neurons to have a mean of zero and a variance of one. This normalization reduces the internal covariate shift, which is the change in the distribution of network activations due to the change in network parameters during training. By stabilizing the distribution of these activations, the network becomes more robust to changes in input data, allowing it to better capture domain-invariant attributes.

Dropout, on the other hand, randomly deactivates a subset of neurons during training. This prevents the network from overly relying on specific neurons or paths, effectively reducing overfitting. It forces the network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. This leads to the model learning more generalized, domain-invariant features, as it cannot depend on the presence of specific features in the training data. Both these techniques in tandem allow the adaptive NN to not only be competitive with the discriminator, but also move in the right direction: in the direction of abstracting numerals, and in the direction of mimicking the source encoder.

## V. Discussion

Upon finally achieving the desired results, our team was faced with many philosophical questions about the method. The first question we pondered was whether an adversarial model could ever classify the target domain better than the source domain. The second question was whether flipping the target domain and source domain leads to better accuracy. For the first question, we should consider the best possible scenario for the target encoder. Since the target encoder is meant to imitate, the best-case scenario is that it imitates the source encoder perfectly, which means it would have the same accuracy score. For the second question, as discussed earlier, the better the source encoder is at abstracting the idea of the numerals, the easier it is to mimic. This means that the dataset which gives the encoder the best chance at capturing the domain-invariant characteristics of the numerals, the easier that encoder will be to mimic, leading to higher accuracy scores for both parties. Hence it is possible to achieve a better score if we flip the source and target domain only in the instance that an encoder trained on the target domain will capture more abstract features.

Does the method of Adversarial Domain Adaptation work for all number datasets? So far in this project, we have focused on two datasets that are quite similar in style. However, the effectiveness of this method for datasets with vastly different numerical representations remains a question. When we applied this model to the SVHN dataset, we found that it performed moderately well, achieving a 70 percent success rate, which is not particularly impressive. What could be the issue? Our team hypothesizes that the presence of too many domain-specific features may hinder the neural network's ability to pick up abstract numerical qualities. So, how do we tackle this issue? This remains an open question in the machine learning community.

## VI. Conclusion

In concluding our exploration of Adversarial Discriminative Domain Adaptation (ADDA), this study has illuminated the potential of this method in the field of machine learning, particularly in the context of image classification. Our journey began with the ambitious goal of overcoming the limitations of domain specificity in neural networks, an objective that led us to delve into the intricacies of the ADDA model.

Throughout our research, we have seen firsthand the challenges and opportunities that come with attempting to generalize across different data domains. Our work with the MNIST and USPS datasets has been particularly revealing. It demonstrated not just the potential of ADDA in bridging the gap between these domains, but also the nuanced complexities involved in such an endeavor. The process of adapting and fine-tuning the ADDA model has been quite a challenge, revealing crucial insights into the dynamics of adversarial training, the importance of encoder generalization, and the delicate interplay between the discriminator and target encoder.

One of the key takeaways from our research is the critical role that both batch normalization and dropout play in the generalization of the encoders. These techniques were instrumental in enabling the target encoder to not only compete with the discriminator but also to move in the direction of effectively abstracting numerical representations. This led to more successful mimicry of the source encoder, a step crucial for the success of the ADDA model.

However, our study also uncovered limitations when applying the ADDA model to datasets with distinctly different domain-specific features, such as the SVHN dataset. This highlighted an important aspect of domain adaptation: the need for a careful consideration of the characteristics and compatibility of the source and target datasets. Our experiences suggest that the more domain-specific features present, the more challenging it is for the neural network to extract and learn abstract numerical qualities.

In light of these findings, we believe that the future of domain adaptation in machine learning is both promising and challenging. While our study attempts to make strides in demonstrating the practical applications and adaptations of the ADDA model, it also opens up new questions and avenues for further research. The exploration of more diverse and complex datasets, the refinement of adversarial training techniques, and the ongoing challenge to improve model adaptability and robustness are just a few areas that require further exploration.

## References

[1] Tzeng, Eric, Judy Hoffman, Kate Saenko, and Trevor Darrell. ”Adversarial Discriminative Domain Adaptation.” arXiv preprint arXiv:1702.05464 (2017): 1-8. https://doi.org/10.48550/arXiv.1702.05464