Title
Comapring Adversarial Unsupervised Domain Adaptation to Zero-Shot Classification in Contrastive Language-Image Pre-Training Embedding Space

Author
Deshpande, Kaustubh

Publication Date
2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Comapring Adversarial Unsupervised Domain Adaptation to Zero-Shot Classification in Contrastive Language-Image Pre-Training Embedding Space

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics and Data Science

by

Kaustubh Deshpande

2023

ABSTRACT OF THE THESIS

Comapring Adversarial Unsupervised Domain Adaptation to Zero-Shot Classification in Contrastive Language-Image Pre-Training Embedding Space

by

Kaustubh Deshpande
Master of Applied Statistics and Data Science
University of California, Los Angeles, 2023
Professor Yingnian Wu, Chair

This paper explores how Unsupervised Domain Adaptation (UDA) measures up to zero-shot classification using contrastive language-image pre-training (CLIP) models. We begin by introducing the concept of domain adaptation and its necessity in various real-world applications. Next, we introduce the ideas behind CLIP models followed by an introduction to the UDA method called Adversarial Discriminative Domain Adaptation (ADDA). Then, we conduct an experimental evaluation of this method by applying it to embeddings obtained from two different CLIP models, ViT-B-32 and ViT-g/14. Finally, we compare the performance of ADDA versus zero-shot classification (ZSC) using the same two CLIP models and provide insights into the implications of the results.

The thesis of Kaustubh Deshpande is approved.

Maryam Mahtash Esfandiari

Tao Gao

Yingnian Wu, Committee Chair

University of California, Los Angeles

2023

To my grandmother . . .

whose lifelong dedication to learning

set an example of intellectual curiosity for generations to come

May you rest in peace.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

In recent years, deep learning models have shown remarkable performance in various computer vision tasks such as image classification, object detection, and semantic segmentation. However, the performance of these models often degrades when they are applied to new or unseen domains, which differ in some way from the training data distribution. This problem, known as domain shift, is a significant challenge in real-world applications where the training and test data may come from different sources.

To address this challenge, domain adaptation techniques have been proposed to transfer knowledge from a source domain with labeled data to a target domain with unlabeled data. These techniques aim to mitigate the domain shift problem and improve the model's performance on the target domain. One common approach is domain alignment which aims to align the feature distributions of the source and target domains, such that the learned features are domain-invariant and generalize well on the target domain [1].

Recently, Contrastive Language-Image Pre-training (CLIP) models have been introduced, which jointly learn visual and textual representations in a self-supervised manner and have achieved state-of-the-art performance on a wide range of natural language understanding and image classification tasks [2]. These models can generate embeddings, which are mathematical representations of data that capture the underlying structure and relationships. These embeddings encompass both visual and textual semantics and are highly informative. They can be leveraged for a range of downstream tasks, including domain alignment, due to the richness of information they contain.

Motivated by this, our study aims to align domains directly in the CLIP embedding space using the Adversarial Discriminative Domain Adaptation (ADDA) methodology [3], which has shown promising results in aligning two domains for both classification and segmentation. First, we demonstrate the existence of a domain shift in our data set,

thus justifying the need for domain adaptation. Next, we perform our experimental evaluation of ADDA. Lastly, we perform zero-shot classification as a benchmark for our ADDA approach.

The remainder of the paper is organized as follows. In Section 2, a comprehensive background is provided on Domain Shift, Domain Alignment and CLIP. Section 3 outlines the main methodology employed in this study. The network architectures utilized for our experiments are introduced in Section 4. Detailed descriptions of the conducted experiments and their results are presented in Section 5. Section 6 delves into the discussion of the results and potential avenues for future research. Finally, Section 7 concludes the paper.

CHAPTER 2

Background

2.1 Domain Shift, Domain Adaptation and Domain Alignment

While working in the field of machine learning and pattern recognition, researchers often encounter the challenge of domain shift, which refers to the change or shift in statistical properties, data distribution, or feature representation between different domains. This shift can manifest as variations in lighting conditions, object appearances, data collection settings, or other factors, leading to differences in the characteristics of the data.

Referring to Figure 2.1, when a model trained on a source domain is applied to a target domain with significant domain shift, it can result in poor performance due to the presence of a domain gap. This gap represents the mismatch or differences between the source and target domains in terms of statistical characteristics and feature distributions. This discrepancy poses a challenge for models as the knowledge learned from the source domain may not be directly applicable to the target domain.

To address domain shift, researchers have developed various domain adaptation techniques. One such technique is domain alignment. This procedure aims to reduce the distributional differences and align the data or model representations between the source and target domains. It involves transforming the data from both domains to a common space where they share similar feature distributions. By aligning the domains, models are enabled to effectively leverage the shared information across domains and overcome the challenges posed by domain shift and discrepancy.

Referring to Figure 2.1 [4], after successfully aligning the domains, a cross-domain classifier can be trained to leverage the aligned representations. This cross-domain classifier effectively bridges the domain gap and enables accurate classification in the aligned domain space. As a result, The cross-domain classifier exhibits an improved performance

on the target domain in comparison the original source domain classifier [4].

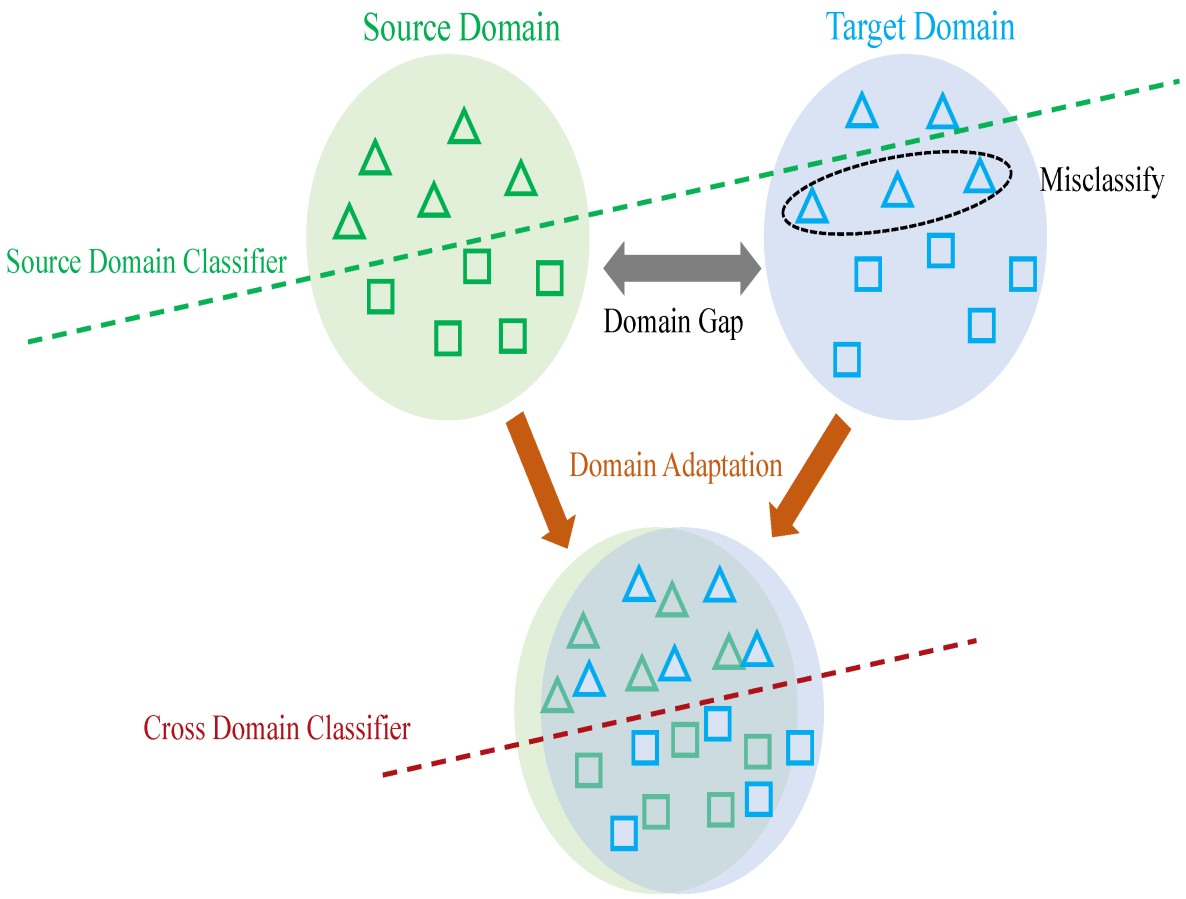


Figure 2.1: Domain Shift and Domain Adaptation

2.2 Contrastive Language-Image Pre-training (CLIP)

CLIP is a cutting-edge multimodal model that has garnered significant attention in the fields of computer vision and natural language processing. It is designed to jointly understand images and textual descriptions, enabling it to establish meaningful associations between visual and semantic representations.

Referring to Figure 2.2 [2], the training of CLIP involves a contrastive learning framework, which leverages large-scale image-text pairs. During training, the model learns to encode both images and text into meaningful representations. The key idea is to push the

embeddings of images and text belonging to the same class closer together in the joint embedding space, while simultaneously pushing embeddings from different classes further apart [5]. This process encourages the model to capture the shared semantics between the visual and textual modalities and learn a rich understanding of their associations.

By incorporating the contrastive learning objective, CLIP acquires a powerful ability to generalize across domains. It can perform tasks such as image classification, object detection, and even zero-shot learning, where it can classify images from unseen categories without specific training. Zero-shot learning is expanded on in detail later in this paper.

On the downside, training and utilizing CLIP models can be computationally expensive and require substantial computational resources. Additionally, fine-tuning CLIP for specific tasks may necessitate a considerable amount of labeled data. Furthermore, CLIP's performance may be challenged in tasks that heavily rely on fine-grained visual details or involve complex visual reasoning.

Despite these limitations, CLIP represents a significant advancement in multimodal learning. Its ability to understand and leverage both images and text opens up exciting possibilities for various applications, including computer vision, natural language understanding, and interdisciplinary tasks involving image-text interactions. The contrastive training approach employed by CLIP enhances its capability to capture the underlying associations between images and textual descriptions, making it a powerful tool in multimodal research and practical applications [5].

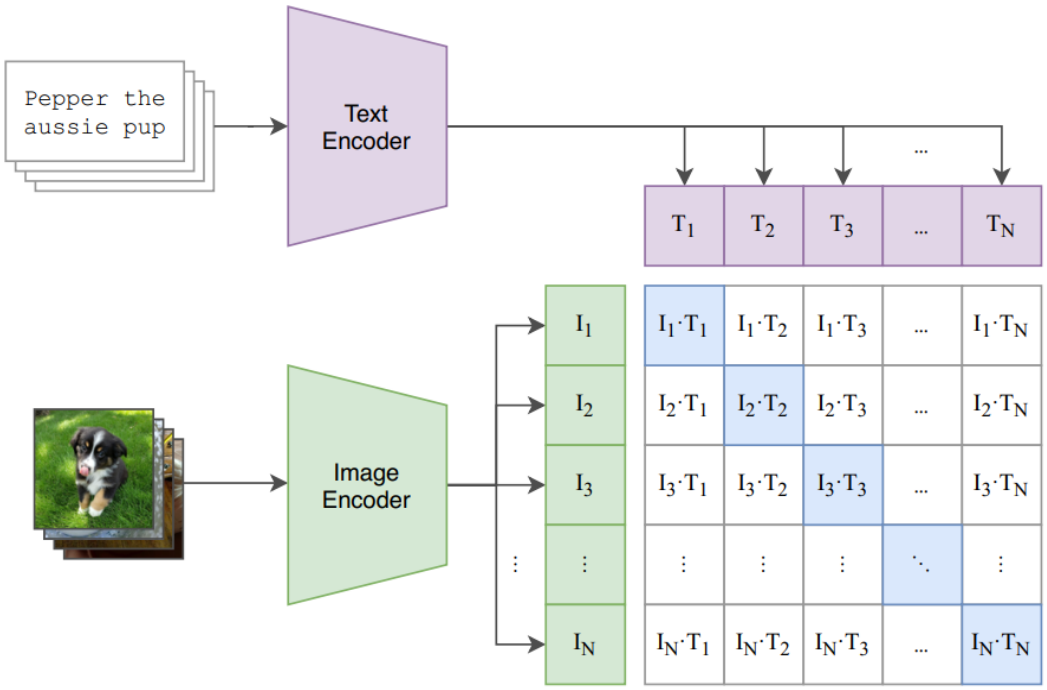


Figure 2.2: Contrastive Language-Image Pre-Training (CLIP)

CHAPTER 3

Methodology

In this section, we introduce the main methods used for our experiments. These are namely Adversarial Discriminative Domain Adaptation (ADDA) and Zero-shot Classification (ZSC). The architectures for networks are not emphasized here.

3.1 ADDA

ADDA is a popular methodology in the field of domain adaptation, specifically addressing the challenge of adapting a model from a source domain to a target domain with a significant domain shift. ADDA aims to reduce the discrepancy between the source and target domains by leveraging adversarial training [6].

Referring to Figure 3.1 [7], ADDA methodology consists of training three neural networks in parallel: a feature extractor (F), a domain discriminator (D) and a classifier (C). The feature extractor is responsible for extracting high-level representations from the data, while the domain classifier aims to distinguish between the source and target domains based on these representations. The goal is to learn domain-invariant features that are informative for the target domain despite the domain shift [6].

During training, ADDA employs an adversarial learning scheme [6]. The feature extractor and domain classifier are trained in an adversarial manner, with the discriminator trained to differentiate between the source embeddings and the target embeddings by minimizing binary cross-entropy loss: D_{ADV}. Simultaneously, the feature extractor is trained to minimize the binary cross-entropy loss on the discriminator's predictions of whether a target embedding, which should belong to the target domain, is misclassified as belonging to the source domain after passing through the extractor: F_{ADV}. Minimizing this loss encourages the extractor to align the domains [6]. Lastly, the feature extractor and

classifier are also jointly trained to minimize the cross-entropy loss on the classification task: C_{Task}. Minimizing this loss ensures that during training the extractor [6].

Here, F_{ADV} and D_{ADV} are computed as Binary Cross Entropy loss (BCE)

$$BCE = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{3.1}$$

Where n is the batch size, y_i is the ground truth domain label for the i-th sample, and $\hat{y}_i$ is the discriminator's predicted probability for the i-th sample. The value for y_i is as follows,

$$y_i = \left\{ \begin{array}{ll} 0, & \text{For } F_{ADV} \\ 0, & \text{For } D_{ADV} \text{ if } Sample_i \in Domain_{source} \\ 1, & \text{For } D_{ADV} \text{ if } Sample_i \in Domain_{target} \end{array} \right\}$$

C_{Task} is computed as Cross Entropy Loss (CE)

$$CE = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{3.2}$$

Where y_i is the ground truth class label and $\hat{y}$ is the predicted class label, and n is the batch size.

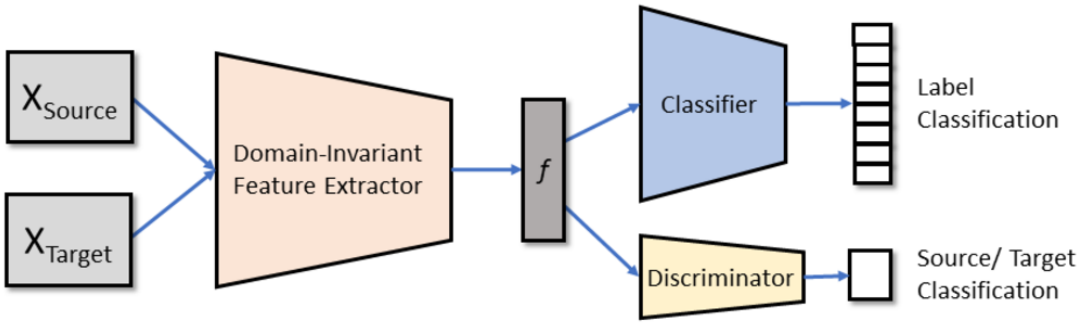


Figure 3.1: Adversarial Discriminative Domain Adaptation (ADDA)

3.2 Zero-Shot Classification

Zero-shot classification (ZSC) has emerged as a powerful approach in multimodal learning. It utilizes CLIP (Contrastive Language-Image Pretraining) models to recognize and classify images from unseen categories without specific training on those categories [2]. As mentioned previously, CLIP models are designed to jointly understand images and textual descriptions, allowing them to learn meaningful associations between visual and semantic representations. ZSC refers to the ability of CLIP models to used this learned associated to recognize and classify images from unseen categories without specific training on those categories [2].

Referring to Figure 3.2 [2], we perform ZSC by preparing text inputs in the format "a photo of a class" for each class present in our dataset. These textual descriptions are then encoded into text embeddings using a selected CLIP model. Simultaneously, we obtain image embeddings for each image in our dataset using the same CLIP model.

To determine the class prediction for a given image embedding, we compute the cosine similarity between the image embedding and each class-specific text embedding [2]. The

cosine similarity provides a measure of the alignment or resemblance between the image and the textual description associated with each class. We identify the class with the highest cosine similarity score as our "zero-shot prediction" for that particular image embedding.

Subsequently, we compare the zero-shot predictions with the ground truth labels assigned to the images in our dataset. By calculating the ZSC accuracy, we evaluate how effectively the chose CLIP model can classify images in our dataset with no direct prior training.

Mathematically, cosine similarity for two vectors is defined as:

$$\text{cosine} \quad \text{similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}||\mathbf{B}|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.3}$$

Here, $\mathbf{A}$ and $\mathbf{B}$ are the two vectors that we want to compute the cosine similarity between, n is the number of dimensions in the vectors, $\cdot$ denotes the dot product between the vectors, and $|\cdot|$ denotes the Euclidean norm.

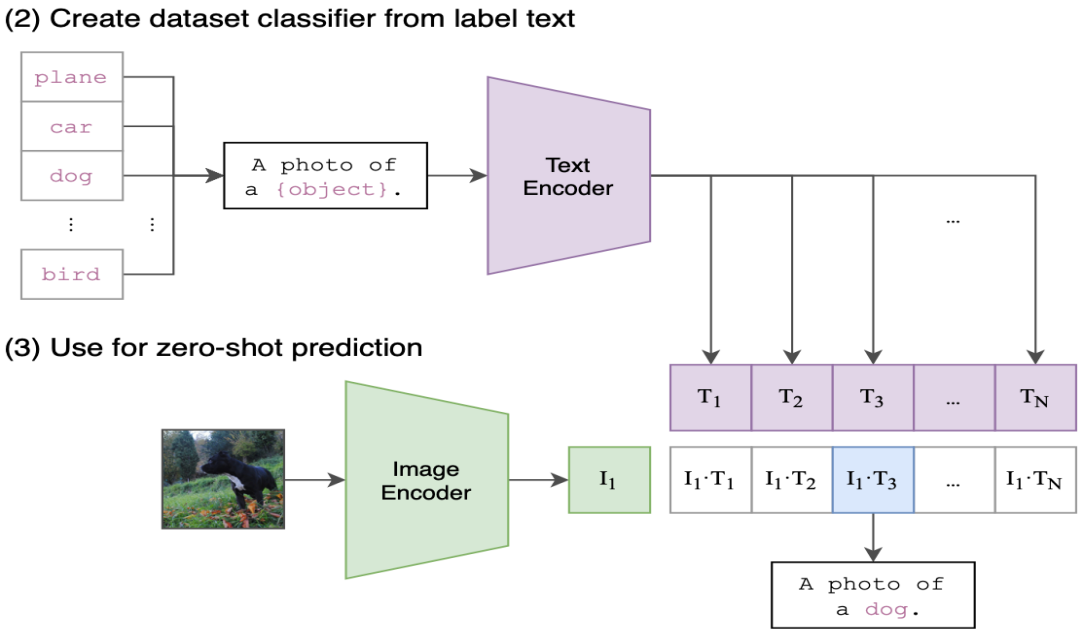


Figure 3.2: Zero Shot Classification

CHAPTER 4

Network Architectures

This section provides a comprehensive overview of the architectures employed in the ADDA setup, focusing on the neural networks and their constituent elements. It delves into the intricate details of the network architecture, including activation functions, dropout rates, batch normalization, and other relevant components. By presenting these details, this section offers insight into the design choices made for the neural networks in the ADDA framework, shedding light on the factors that contribute to its effectiveness in accomplishing domain adaptation.

4.1 Feature Extractor (F)

The feature extractor in our ADDA setup is a Neural Network responsible for extracting task-specific features from the CLIP embeddings. F is initially trained on the source domain data and learns to extract features that are relevant to the specific task at hand, which for our study is classification. Throughout the rest of the training process, F learns to map target embeddings to the source domain where class decision boundaries are preserved. This makes it easier for a downstream classifier to successfully classify target domain samples without having been trained on any labelled target domain data. For our study, the architecture of F is as follows,

```python
class F(nn.Module):
    def __init__(self, in_features, out_features):
        super(F, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(in_features, 768),
            nn.BatchNorm1d(num_features=768),
            nn.Mish(inplace=True),
            nn.Linear(768, 768),
            nn.BatchNorm1d(num_features=768),
            nn.Mish(inplace=True),
            nn.Linear(768, 512),
            nn.BatchNorm1d(num_features=512),
            nn.Mish(inplace=True),
            nn.Linear(512, out_features)
        )

    def forward(self, x):
        return self.model(x)
```

Figure 4.1: Feature Extractor Architecture

Referring to the Figure 4.1 , F consists of four fully connected layers with batch normalization to and Mish activation functions. The input and the output layers are tensors with a specified number of features. While, the hidden two layers have 768 output features. The final output from F is passed as input to C and D for predicting class labels and domain labels respectively.

4.1.1 Mish

Mish activation function [8] is used to introduce non-linearity into the feature extractor architecture and is defined as:

$$f(x) = x \tanh(\ln(1 + \exp(x)))$$

4.1.2 Batch Normalization

Batch normalization [9] is used to normalize the activations of the previous layer. Batch normalization reduces the effect of internal covariate shift, speeds up training, regularizes the network and prevents overfitting.

BatchNorm1D is defined in Pytorch as:

$$\mathrm{BN}(x_i) = \gamma \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}^2 + \epsilon} + \beta$$

Where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are the mean and standard deviation of the mini-batch $\mathcal{B}$, respectively. γ and β are learned scaling and shifting parameters, respectively. ϵ is a small constant added for numerical stability [10].

4.2 Domain Discriminator (D)

The Domain Discriminator in our ADDA set up is a neural network responsible for classifying whether a given output from F is from the source or target domain. The domain discriminator is trained using binary cross entropy loss, and its output is used to inform the F on how to produce features that are indistinguishable between the source and target domains. By doing so, F learns domain-invariant features and ensures that any downstream model, such as a classifier, generalizes well to the target domain.

```python
class D(nn.Module):
    def __init__(self, in_features):
        super(D, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(in_features, 128),
            nn.BatchNorm1d(128),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.5),
            nn.Linear(128, 64),
            nn.BatchNorm1d(64),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.5),
            nn.Linear(64, 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.model(x)
        return x.squeeze()
```

Figure 4.2: Discriminator Architecture

Referring to the Figure 3.2 , D consists of three fully connected layers with 128, 64, and 1 output units respectively, and batch normalization applied after each hidden layer. The LeakyReLU activation function, with a negative slope of 0.2, is used to introduce non-linearity into the network. Dropout with a rate of 0.5 is applied after each hidden layer to prevent overfitting and capture the help capture the essential discriminative patterns in the input data [6] [11]. The final output layer uses the sigmoid activation function, which maps the output to a probability between 0 and 1, indicating the likelihood that the input sample belongs to the target domain. The sigmoid function is commonly used in binary classification problems to convert the network's output into a probability score.

4.2.1 Leaky ReLU

The Leaky ReLU activation function [12] with a slope of 0.2 is defined as:

$$\text{LeakyReLU}(x) = \left\{ \begin{array}{ll} x, & \textbf{if } x > 0 \\ 0.2x, & \textbf{otherwise} \end{array} \right\}$$

4.2.2 Sigmoid

The sigmoid activation function [13] is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

4.3 Classifier (C)

The Classifier in our ADDA set up is a neural network responsible for performing the classification task on the input data, given the domain-invariant learned features from F. For our study, the architecture of C is as follows,

```
class C(nn.Module):
    def __init__(self):
        super().__init__()

        self.layer4 = nn.Linear(256, 128)
        self.layer5 = nn.Linear(128, 64)
        self.layer6 = nn.Linear(64, 12)
        self.mish = nn.Mish()

        self.C=nn.Sequential(self.layer4, self.mish, self.layer5, self.mish, self.layer6)

    def forward(self, x):
        return self.C(x)
```

Figure 4.3: Classifier Architecture

Referring to the Figure 3.3 , C consists of three fully connected layers with 128, 64, and 12 output units respectively, and the Mish activation function applied after each hidden layer. The Mish activation function is a non-linear activation function that has

been shown to improve the performance of deep neural networks. The final output layer maps the output to a 12-dimensional vector, which represents the probabilities of the input sample belonging to each of the 12 classes in our dataset. This is achieved using the softmax activation function, which ensures that the output probabilities sum to 1.

4.3.1 Softmax

The softmax activation function is defined as:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Where x_i is the i^{th} element of the input vector x, and the sum in the denominator is taken over all elements of x.

CHAPTER 5

Experiments

In this section, we delve into the practical implementation and evaluation of our proposed methodologies. For all experiments, we utilize embeddings from two distinct models, namely ViT-B-32 [5] and ViT-g/14 [2], with output embedding sizes of 512 and 1280, respectively.

We begin by providing an overview of the dataset employed in our study, highlighting its characteristics and composition. Subsequently, we present evidence of domain shift by evaluating the performance of a source-only model on the target domain. This analysis serves to highlight the challenges posed by domain shift and substantiates the need for domain adaptation techniques.

Finally, we present the results of our experiments, which demonstrate the effectiveness of ADDA in mitigating the impact of domain shift and achieving improved classification accuracy across both domains. Additionally, we compare these results with the zero-shot classification (ZSC) approach using the mentioned CLIP models. By examining the outcomes, we gain valuable insights into the capabilities and advantages of ADDA for domain adaptation and its performance relative to ZSC.

5.1 Dataset

For our experiment, we are working with the VisDA-2017 image classification data set, which poses a 12-class domain adaptation challenge. Referring to Figure 5.1 [14], the source domain is composed of 152,397 synthetic images, where 3D CAD models are rendered from various conditions . On the other hand, the target domain consists of 55,388 real images sourced from the MS-COCO data set. Lastly, the test set for the target domain consists of 72,372 real images [14]. The test set is a new real-image test

domain, different from the validation domain and without labels. By using different target domains for validation and testing, we can assess how well the proposed models perform as a domain adaptation tool in novel, real-world scenarios. This approach closely resembles actual deployment situations where the target domain is unknown during training, and it discourages algorithms that are designed to only work for specific target domains.

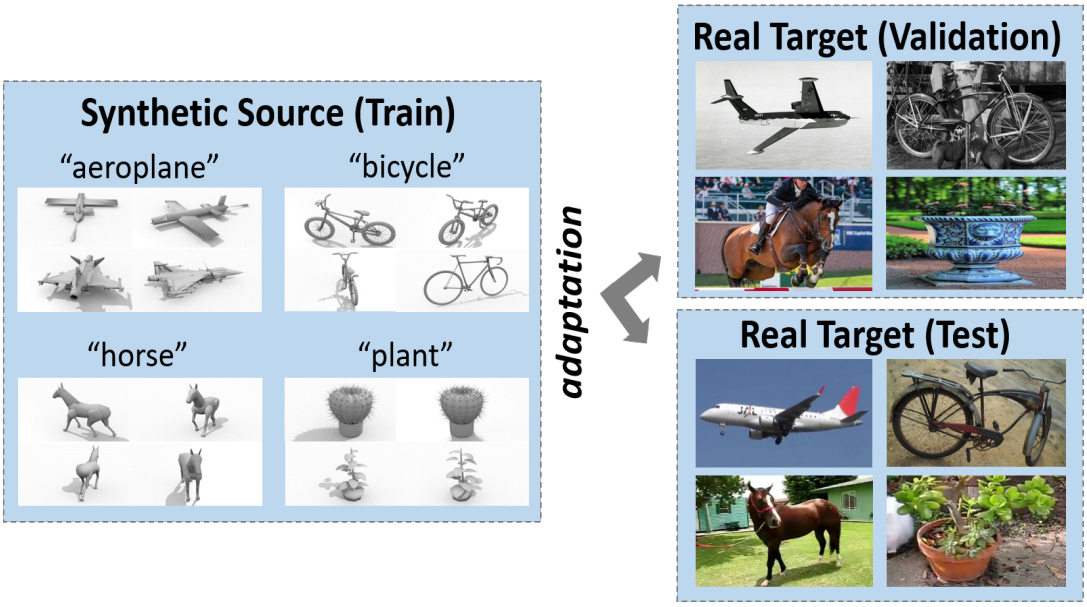
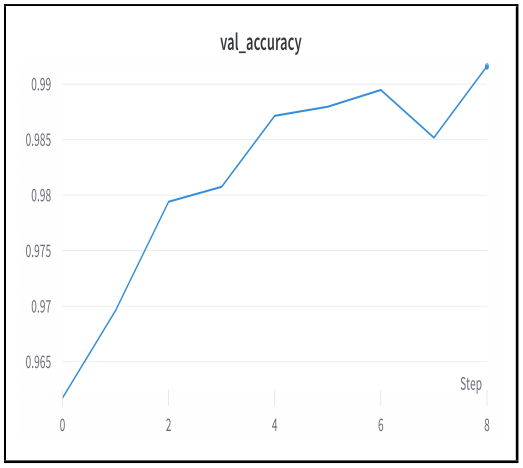


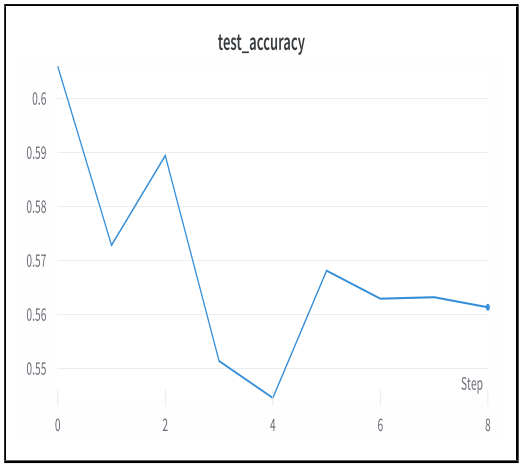
Figure 5.1: VisDA-2017 Dataset

5.2 Domain Shift

As mentioned previously, domain shift is a common issue that arises in computer vision tasks. Prior to tackling domain shift with ADDA we must first confirm its existence in our dataset. We do this by assessing how a model trained/validated on source domain performs on the target domain.

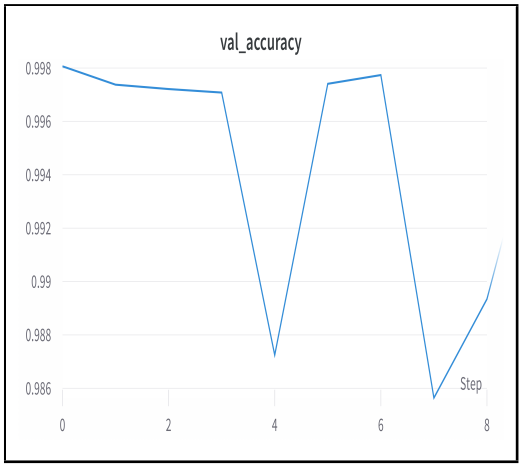


(a) Source Domain Validation Accuracy

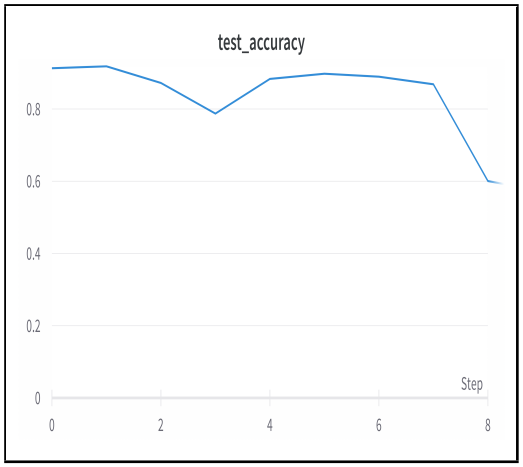


(b) Target Domain Test Accuracy

Figure 5.2: Domain Shift - ViT-B-32



(a) Source Domain Validation Accuracy



(b) Target Domain Test Accuracy

Figure 5.3: Domain Shift - ViT-g/14

Referring to Figure 4.2 and 4.3, classifiers trained on embeddings from both CLIP ViT-B-32 and CLIP ViT-g/14 models exhibited significant drops in performance when tested on the target domain compared to the source domain. While achieving an accuracy of 99% on the source domain, their performance on the target domain test set was only 61% and 82%, respectively. The superior performance of the ViT-g/14 embeddings on the target domain compared to ViT-B-32 can be attributed to its larger size and richer information content in the embeddings. These results confirm the presence of a domain

shift and encourage us to proceed with domain adaptation.

5.3 Result

Having confirmed domain shift using source only performance accuracy on the target domain test set, we proceeded with training the ADDA pipeline using embeddings from ViT-B-32 and ViT-g/14. We also computed zero-shot accuracy for both the CLIP models to compare against our domain adaptation performance.

	CLIP ViT-B-32	CLIP ViT-g/14
Source only	61.2	82.1
ADDA	73.5	91.2
ADDA + D multi-step	78.6	89.8
ADDA + weighted F_{ADV}	81.2	91.2
ADDA + D multi-step + weighted F_{ADV}	81.1	91.2
Zero-shot	**86.6**	**93.5**

Table 5.1: Results on Target Domain Test Set

Table 5.1 shows the results of different methods on the target domain test set. In the previous section we computed source only accuracy, which achieved an accuracy of 61.2% and 82.1% on CLIP ViT-B-32 and CLIP ViT-g/14, respectively.

While applying ADDA improved the accuracy to 73.5% and 91.2%, we noticed that the F_{ADV} loss was dropping too quickly, specifically when working with the smaller embeddings from CLIP ViT-B-32. We hypothesized that the root cause was a relatively weak discriminator that was unable to effectively distinguish between the source domain data and the target domain data. Consequently, allowing the feature extractor to easily fool the discriminator by overfitting on the source domain data or on non-domain-invariant features, instead of learning domain-invariant features that could generalize to the target domain. As a result, although the loss for the feature extractor decreased quickly, the

model's performance on the target domain did not improve as well as it could have and may have even degraded if we trained the pipeline for longer.

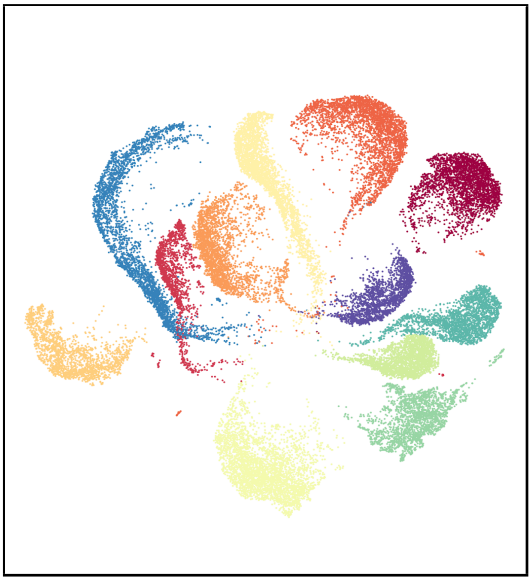
To address this issue, we attempted multi-step adversarial training [15], updating the discriminator twice for each update of the feature extractor during the first 10 epochs only. This method, ADDA + D multi-step in table 4.2, further improved accuracy for ViT-B-32 to 78.6% but slightly decreased accuracy for ViT-g/14 to 89.8%.

We also tried to weight down the F_{ADV} loss to prevent the feature extractor from overfitting on the source domain data. This method, ADDA + weighted F_{ADV} in table 4.2, achieved the best ADDA based performance for both Vit-32 and ViT-g/14 with accuracies of 81.2% and 91.2% respectively. We also tried both methods in parallel however that did not alter the performance much.

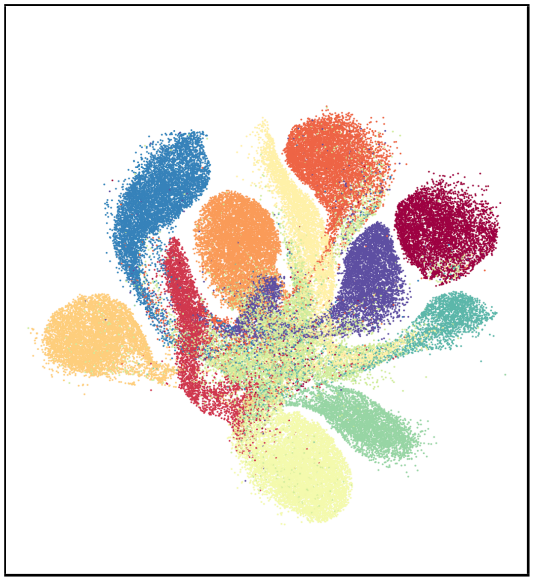
Finally, we evaluated the zero-shot approach, which achieved the highest accuracy of 86.6% and 93.5% for CLIP ViT-B-32 and CLIP ViT-g/14, respectively. These results demonstrate the effectiveness of ADDA in addressing the domain shift challenge but also highlight the power of CLIP models as they were able to outperform all ADDA methods with just zero shot predictions.

5.4 UMAP Visualizations

UMAP (Uniform Manifold Approximation and Projection) [16] is a powerful dimensionality reduction technique that enables the visualization of high-dimensional data in a lower-dimensional space. For qualitative analysis, we utilized UMAP to visualize the feature extractor output for both domains before and after domain adaptation. To ensure fair comparisons, we maintained consistent hyperparameters when generating all UMAP visualizations.

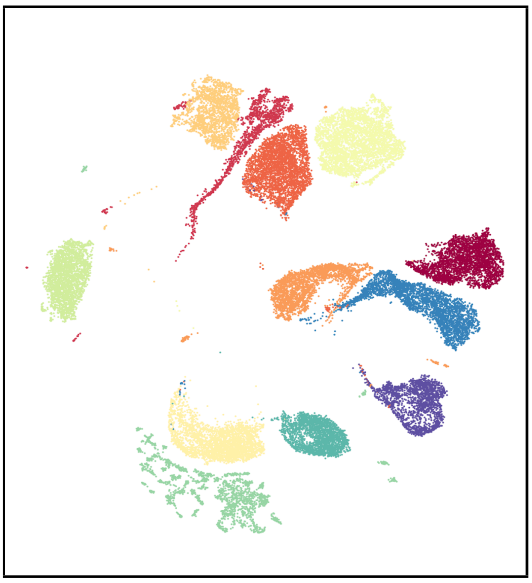


(a) Source Domain Features

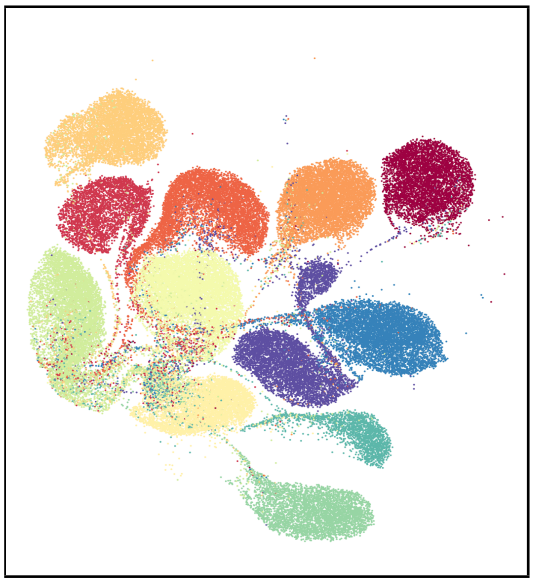


(b) Target Domain Features

Figure 5.4: Unaligned Features - ViT-g/14



(a) Source Domain Features



(b) Target Domain Features

Figure 5.5: Aligned Features - ViT-g/14

Referring to Figure 5.4, the UMAP visualization of the source domain prior to alignment reveals a noticeable degree of separation. This initial separation can be attributed to the fact that the feature extractor and classifier are jointly pre-trained with a supervised loss, enabling them to effectively extract and capture discriminative patterns associated

with the class labels. In contrast, the UMAP visualization of the target domain features initially lack clear separation due to the absence of labeled target domain data.

After applying ADDA, Figure 5.5 illustrates a noticeable improvement in the separation of both the source and target domain features in the UMAP visualizations. This enhancement in separation is consistent with our quantitative results, which demonstrate improved classification accuracy in the aligned domain space. Thus, ADDA successfully aligns the distributions of the source and target domains, enabling a cross-domain classifier to generalize effectively to the target domain samples.

Overall, the UMAP visualizations serve as visual evidence of the effectiveness of ADDA in mitigating the effects of domain shift and aligning the source and target domains. They provide a qualitative assessment of the success of the alignment process and reinforce the findings from our quantitative analysis.

CHAPTER 6

Discussion

The result section demonstrated the effectiveness of domain adaptation techniques for aligning feature representations between source and target domains. We observed a significant improvement in classification accuracy when applying unsupervised domain adaptation compared to using the source-only model, with the best-performing method achieving an accuracy of 81.2% and 91.2% for CLIP ViT-B-32 and CLIP ViT-g/14, respectively. Our UMAP visualizations also confirmed that applying resulted in improved separation for features in both domains.

The more defining result from our experiments, however, was the performance of the zero-shot approach, achieving the highest accuracy of 86.6% and 93.5% for CLIP ViT-B-32 and CLIP ViT-g/14, respectively. These numbers demonstrate the power of CLIP models and their ability to generalize to new and unseen domains without the need for domain-specific training data.

While these results suggest that zero-shot techniques could prove a viable replacement for unsupervised domain adaptation, there are several limitations that should be acknowledged. Firstly, the focus of the research is limited to a single task of classification, which may not capture the full extent of the UDA approach's capabilities across different tasks. Secondly, the investigation is conducted on a single dataset, potentially restricting the generalizability of the findings to other datasets with varying characteristics. Lastly, the study only explores two CLIP models, which might not represent the entire spectrum of CLIP model performance and could overlook other models that could potentially yield better results.

To gain a more comprehensive understanding of the UDA approach and its applicability, future research should aim to address these limitations by encompassing a wider range of tasks, datasets, and CLIP models. For alternative datasets, we recommend exploring

more niche or domain-specific datasets. Such data is more likely to be encountered in a production setting and is more difficult to zero shot on compared to commonplace objects such as trains, cars and bicycles which were a part of the VisDA-2017 data set used in this study.

As for testing other downstream tasks, we aim to explore instance segmentation, semantic segmentation [17], or multi-label classification. By expanding the scope of research beyond classification, we can gain a more comprehensive understanding of the capabilities and limitations of unsupervised domain adaptation and zero-shot techniques.

CHAPTER 7

Conclusion

This paper accomplishes two main objectives. Firstly, it implements an unsupervised domain adaptation (UDA) method named ADDA at the CLIP embedding level without relying on image-level augmentation. Secondly, it evaluates the performance of this approach by comparing it to zero-shot classification. Unsupervised domain adaptation is a popular approach for addressing significant domain shift commonly encountered in real-world scenarios. This approach aligns with existing methods in the field, employing conventional statistical techniques and neural network architectures. Notably, this approach shares similarities with the works of Shrivastava et al. (2017) [18] and Lee et al. (2019) [6], who explored Simulated+Unsupervised (S+U) learning using Generative Adversarial Networks (GANs) and adversarial dropout to enhance adaptation through learning discriminative features, respectively. Other notable methods in domain adaptation include training cyclic GANs [19] or Wasserstein GANs (WGANS), which aim to learn two-way mapping between domains and utilize Wasserstein loss, respectively [20]. While GANs have been widely explored, they are usually not the preferred choice in production settings due to challenges such as mode collapse and instability [21].

In this paper, we first introduced the challenge of domain shift accompanied by a detailed background of domain adaptation techniques, specifically, domain alignment to overcome this challenge. Next, we provided a theoretical introduction to CLIP models and their increasing popularity in the field of computer vision and natural language processing. This was followed by a detailed account of adversarial discriminative domain adaptation methodology. Which aims to align the embeddings from CLIP models in both the source and target domains into a shared space. By achieving alignment, we enabled the training of a cross-domain classifier that can generalize well to the target domain samples.

Our results demonstrated the effectiveness of these techniques for aligning feature representations between source and target domains, resulting in significant improvements in classification accuracy. We also compared the performance of UDA to the zero-shot (ZSC) approach and found that the latter achieved the highest accuracy. This result specifically stood out from our experimental evaluation and further highlighted the strength of CLIP models and their ability to generalize to new domains without the need for domain-specific training data.

While our study focused on classification tasks, further research can explore the performance of both techniques on other downstream tasks such as instance segmentation, semantic segmentation, or multi-label classification. Additionally, testing on more niche or domain-specific datasets is needed to obtain a fair comparison of UDA vs ZSC

In summary, this study presents promising results for UDA directly at a CLIP embedding level and highlights the potential of zero-shot approaches as a viable alternative. These findings contribute to the ongoing research on domain adaptation and zero-shot learning and can inform future developments in this field.

CHAPTER 8

Appendix

8.1 A Code Repository

Code used in this research project is available on GitHub in the repository *ADDA_vs_ZSC* by Kaustubh Deshpande. The repository contains the implementation of the algorithms and methods used for data analysis and model training. It can be accessed at: `https://github.com/kaus0399/ADDA_vs_ZSC`.

REFERENCES

[1] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *CoRR*, abs/1109.6341, 2011.

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[3] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017.

[4] Yongjie Shi, Xianghua Ying, and Jinfa Yang. Deep unsupervised domain adaptation with time series sensor data: A survey. *Sensors*, 22(15):5507, Jul 2022.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[6] Seungmin Lee, Dongwan Kim, Namil Kim, and Seong-Gyun Jeong. Drop to adapt: Learning discriminative features for unsupervised domain adaptation. *CoRR*, abs/1910.05562, 2019.

[7] S. Choudhuri. Adversarial domain adaptation with keras. `https://github.com/S-Choudhuri/Adversarial-Domain-Adaptation-with-Keras/tree/master`, Year the image was accessed. Accessed on Day Month Year.

[8] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[10] PyTorch Contributors. Pytorch documentation. `https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm1d.html`, Accessed on 2023-05-25.

[11] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[12] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[13] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[14] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1710.06924*, 2017.

[15] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[16] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[17] Timo Lüddecke and Alexander S. Ecker. Prompt-based multi-modal image segmentation. *CoRR*, abs/2112.10003, 2021.

[18] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016.

[19] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[20] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[21] Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. How to train your DRAGAN. *CoRR*, abs/1705.07215, 2017.