Learning Transferable Data Representations Using Deep Generative Models

by

Jose Miguel Eusebio

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2018 by the
Graduate Supervisory Committee:

Sethuraman Panchanathan, Chair
Hasan Davulcu
Hemanth Venkateswara

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

Machine learning models convert raw data in the form of video, images, audio, text, etc. into feature representations that are convenient for computational processing. Deep neural networks have proven to be very efficient feature extractors for a variety of machine learning tasks. Generative models based on deep neural networks introduce constraints on the feature space to learn transferable and disentangled representations. Transferable feature representations help in training machine learning models that are robust across different distributions of data. For example, with the application of transferable features in domain adaptation, models trained on a source distribution can be applied to a data from a target distribution even though the distributions may be different. In style transfer and image-to-image translation, disentangled representations allow for the separation of style and content when translating images.

This thesis examines learning transferable data representations in novel deep generative models. The Semi-Supervised Adversarial Translator (SAT) utilizes adversarial methods and cross-domain weight sharing in a neural network to extract transferable representations. These transferable interpretations can then be decoded into the original image or a similar image in another domain. The Explicit Disentangling Network (EDN) utilizes generative methods to disentangle images into their core attributes and then segments sets of related attributes. The EDN can separate these attributes by controlling the flow of information using a novel combination of losses and network architecture. This separation of attributes allows precise modifications to specific components of the data representation, boosting the performance of machine learning tasks. The effectiveness of these models is evaluated across domain adaptation, style transfer, and image-to-image translation tasks.

DEDICATION

For my family: Jesse, Suzette, Francis, and Isabel, whose unending support has provided me with confidence, wisdom, and great joy.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The rise of big data and the development of discrete graphics cards has allowed researchers and scientists to create and test a wide variety of new machine learning models that are increasingly capable. In some cases, these models are more capable than people. The machine learning model AlphaGo Zero learns to play the game of Go by competing against itself in order to learn new, optimal ways of performing tasks. With several weeks of training, an AlphaGo Zero model is capable of competing with and beating the world's greatest players of Go who have decades of experience and knowledge[1]. Similarly, other models and approaches are capable of performing tasks like opening doors, shooting a basketball, and driving a car. In a small subset of tasks, artificial intelligence has been created that is comparable or better than humans. However, AI still struggles to compete with human intelligence in general.

One aspect of human intelligence that is still challenging for machine learning models is the ability to adapt and transfer knowledge across different contexts. When a human learns to perform some task, they understand not only which actions to take, but also the the context of the task and the reasons for performing specific actions. This abstract understanding allows people to generalize knowledge they learn for a specific task and utilize it in other settings. For example, a tennis player, with their knowledge of racket placement and ball spin, is able to easily transfer this knowledge to another racket sport like ping pong and gain a competitive edge over a beginner. Harnessing this ability to transfer knowledge into a machine learning model would allow the creation of tools that can process and understand new, unlabeled data and

[1]`https://deepmind.com/blog/alphago-zero-learning-scratch/`

utilize this knowledge for applications in assistive technology, robotics, and healthcare.

In machine learning, the traditional feed forward neural network is able to learn higher-level concepts of data when learning to classify it. The problem is that this knowledge is highly specific to the data used during training. The singular objective for the network is to learn to classify the training data. In order to create highly transferable data representations, researchers must develop new methods and network architectures to represent the data and its abstract attributes.

This thesis focuses on models that learn transferable data representations by leveraging different generative models. Richard Feynman said, "What I cannot create, I do not understand."[2] This quotation alludes to the idea that a deep understanding of an object is necessary before one can create it. The act of creation, or generation, necessitates an understanding of the component parts as well as their interactions with each other. By using models that learn to generate data, researchers can further understand how data can be broken down and recomposed. With this deeper understanding, researchers can also learn to modify these components in such a way that they can be reused and adapted for more applications, tasks, and contexts.

1.1 Goals and Motivations

The goal of this thesis is to propose machine learning models with transferable data representations that can be used for machine learning tasks such as domain adaptation and style transfer. It seeks to highlight knowledge transfer in deep generative machine learning models and summarizes the related literature. This thesis seeks to directly contribute to two larger challenges: labeling big data and reverse-engineering the brain.

1. *Labeling Big Data*: Creating infrastructures to store and handle the data as well

[2]`http://archives-dc.library.caltech.edu/islandora/object/ct1:483`

as analyzing and understanding the large amount of data are both large problems in an age of technology that generates massive amounts of data. Although there are some aspects of data that are easy to understand like the origin of data or the date of creation, there are many useful attributes of data that are harder to extract. Learning powerful ways to label these attributes in new data would make the data more usable and valuable for important problems.

2. *Reverse-Engineering the Brain*: The National Academy of Engineering (NAE) has laid down 14 challenges for the 21st century [3] that have far-reaching impacts. One of these challenges is to reverse-engineer the brain. The goal of this challenge is to create machines capable of emulating human intelligence. Because of the capability of the human brain, many believe that as the research progresses, the performance of machine learning tasks will drastically increase. New technology and machine learning models will allow great advances in healthcare, manufacturing, and communications. The particular part of this challenge that this thesis focuses on is the ability to transfer previously gained knowledge and adapt to a wide range of data inputs, an important ability of human intelligence.

1.2 Thesis Outline

The thesis is structured in the following manner:

Chapter 2 provides an overview of different concepts covered in this thesis including disentanglement, transfer learning, domain adaptation, and image-to-image translation. The first section introduces the concept of domains of data and disentangling generative factors of a domain. The second section introduces the problem of transfer learning, domain adaptation, and the relevant notation. The discussion of domain adaptation includes the explanation of two problem scenarios, semi-supervised and

[3] `http://www.engineeringchallenges.org/challenges.aspx`

unsupervised, as well as an overview of several domain adaptation methods. The third section introduces generative models. Specifically, this section discusses the Variational Autoencoder (VAE) (Kingma and Welling (2013)), the Generative Adversarial Network(GAN) (Goodfellow *et al.* (2014)), and the application of generative models for image-to-image translation.

Chapter 3 is a survey on the research in deep generative models relevant to transferable data representations. The first section introduces current deep generative models and key models that drastically progressed the capabilities of the original GAN and VAE models. This section also highlights important generative models used for disentangling generative factors. The second section introduces extensions of generative models for the use of machine learning tasks such as image-to-image translation, style transfer, and domain adaptation.

Chapter 4 proposes the Semi-Supervised Adversarial Translator (SAT). The chapter describes the model, which utilizes a set of coupled VAE-GANS to map data from multiple domains into a shared latent space. The network achieves this shared latent space through a combination of adversarial training, cross-domain weight sharing, and classification losses. Using a shared latent space, the network is able to perform cross-domain image-to-image translation as well as domain adaptation. Throughout the chapter, the model, the network architecture, the applications, and the contributions are detailed and evaluated.

Chapter 5 discusses the Explicit Disentangling Network (EDN), which seeks to simplify the task of transferable data representations through explicitly disentangling subsets of the generative factors. As a direct result to lessons learned from the SAT, the EDN learns to organize generative factors of data into two sets, content-related factors and style-related factors. This separation allows the network to handle each set of factors differently, resulting in more efficient performance in domain adaptation and image-to-image translation. Throughout the chapter, the model, the network architecture, the applications, and the contributions are detailed and evaluated. **Chapter 6** concludes the thesis by summarizing the results and contributions.

1.3 Contributions

The contributions of this thesis as well as other graduate work are as follows.

1. A novel semi-supervised deep generative model, the Semi-Supervised Adversarial Translator, is proposed that accomplishes semi-supervised image-to-image translation and domain adaptation using unexplored combinations of domains. The model extends previous models by incorporating a modified discriminator and losses relating to the sparse target labels.

2. A novel model, the Explicit Disentangling Network, is proposed that exemplifies explicit disentanglement and demonstrates the performance and possibilities of the new approach. By separating content-related and style-related factors through a specific network architecture and novel combination of losses, the model learns a powerful, transferable representation of data. This work will be submitted as an article to the IEEE Transactions on Neural Networks and Learning Systems journal.

3. Creation of the Office-Home dataset, a deep learning dataset aimed to evaluate domain adaptation algorithms. It consists of nearly 15,500 images from 65 different categories and 4 domains: art, clipart, product, and real-world. This work, along with other research was published in "Deep Hashing Network for Unsupervised Domain Adaptation" Venkateswara *et al.* (2017).

Chapter 2

BACKGROUND

This chapter provides an overview of different concepts, models, and tasks covered in this thesis including disentanglement, generative models, transfer learning, domain adaptation, and image-to-image translation. Section (2.1) introduces the the task of disentangling generative factors of a domain. Section (2.2) discusses the major generative models, the VAE and GAN, and their methods, network architectures, and capabilities. Section (2.3) introduces concepts concerning transferable representations including transfer learning, domain adaptation, and image-to-image translation. The discussion of domain adaptation includes the explanation of two problem scenarios, semi-supervised and unsupervised, as well as an overview of several relevant domain adaptation methods.

2.1 Disentangling Generative Factors

One way to analyze data is to model it as a collection of independent generative factors (Bengio (2012)). Each generative factor relates to one of the many independent attributes found in a domain, such as the color or shape of an object. A domain, or collection of data, could likewise be modeled as a collection of possible variations of a set of generative factors. These factors interact in intricate ways, complicating machine learning tasks. For example, when the lighting of an object changes from one angle to another, the many different shadows change shape and size and result in a drastically different image, often causing problems for tasks like object classification. Disentangling the generative factors relating to the shadow and an object allows machine learning models to separate the two and focus solely on the object for more effective classification. In general, disentanglement leads to a clear understanding of the core features and attributes of the data and brings greater control and performance in machine learning tasks .

2.2 Generative Models

Given training data X that is distributed by $P(X)$, the goal of generative models is to learn a model $\hat{P}(X)$, which can generate new samples similar to the ones in X. The generation of new data and the process of learning $\hat{P}(X)$ are both powerful, useful tools. Generative models have demonstrated to be able to create new, realistic art, increase the resolution of blurry images, and color black and white images. Outside of working with images, generative models for time-series data can be used for many reinforcement learning applications. The latent representations learned by generative models are useful themselves. The data representations of generative models can be adapted and then utilized for new tasks. Recently, two generative models have risen in popularity, the VAE (Kingma and Welling (2013)) and the GAN (Goodfellow *et al.* (2014)) which both make use a differentiable generator network.

2.2.1 Variational Autoencoders

The VAE is a generative model that approximates the data distribution $P(X)$ of input from X into a low dimensional representation by using a decoding generative network and an encoding inference network paired together in an autoencoder(Kingma and Welling (2013)). The VAE model utilizes an inference network as a probabilistic encoder $q_\phi(z|x)$ that produces a distribution over the possible values of latent code z. The decoding network $p_\theta(z|x)$, given z, produces a distribution over possible values of x that could have produced z. VAEs are trained by maximizing the variational lower bound L_{VAE} associated with data x:

$$L_{VAE} = \mathbb{E}_{z \sim q(z|x)}[-\log p_G(x|z)] + \mathrm{KL}(q(z|x)||p(z)) \tag{2.1}$$

The first term in the equation can be seen as the log-likelihood reconstruction and ensures the autoencoder network is able to reconstruct input x. The second term acts as a regularizer and aims to minimize the Kullback-Leibler divergence between the approximate posterior distribution $q(z|x)$ and the model prior $p(z)$, encouraging the approximate posterior to be closer to prior $p(z)$. Typically, $p(z)$ is chosen to be a centered isotropic multivariate Gaussian, i.e. $p(z) = \mathcal{N}(z, 0, I)$. Encouraging this Gaussian distribution forces the encoder network to find efficient variables and features in order to represent data in the latent space. In some cases, the different variables within the latent space can correspond to disentangled generative factors (Higgins *et al.* (2016)).

Normally, backpropagation cannot flow through a random node, which is present in a VAE when sampling z. VAEs avoid this limitation by reparameterizing z as a function that takes a noise vector ϵ as well as the outputs of the encoder network μ and σ:

$$z \sim \mathcal{N}(\mu, \Sigma) \text{ where } z = \mu + \sigma\epsilon,\ \epsilon \sim \mathcal{N}(0, I),\ \Sigma = \sigma\sigma^T \tag{2.2}$$

2.2.2 Generative Adversarial Networks

GANs pair a generator network with a discriminator network to generate high-fidelity images(Goodfellow *et al.* (2014)). GANs learn in a game theoretic scenario where the generator and discriminator network compete against each other. The generator network generates samples $x = g(z; \theta^{(g)})$ with z typically being a random noise vector. The discriminator, on the other hand, learns to classify the source of incoming samples, whether they be taken from the original training data or generated samples. This binary classification is given by $d(x;\theta^{(d)})$. This competition between the two networks can be seen as a zero-sum game, with function $v(\theta^{(g)},\theta^{(d)})$ acting as the utility of the game, and can be modeled by:

$$v(\theta^{(g)},\theta^{(d)}) = \mathbb{E}_{x\sim p_{data}} \log d(x) + \mathbb{E}_{x\sim p_{model}} \log(1-d(x)) \tag{2.3}$$

The first term in the equation relates to the success of the discriminator in classifying original data as original data while the second term positively relates to the ability for the generator to create data classified as original by the discriminator. Since these two terms are intrinsically opposed, the model trains in an oscillating manner. At each training step, if either network improves, the other network learns to improve in the next step. Convergence in a successful GAN occurs when the model hits a saddle point where the oscillation becomes negligible. At this point, the generator creates data so similar to the original data that the discriminator can only correctly identify the generated images half of the time.

Although GANs have the capability for high fidelity image generation, they still suffer from stabilization issues. In many cases the generator or discriminator of a model becomes unpropotionally better at the zero-sum game. Because the GAN relies heavily on the balance of generation and discrimination, these mismatched networks perform sub-optimally. Thus, is it often necessary to finely tune the model architecture as well as the model's hyper-parameters. Extensions to the GAN, such as a deep convolution GAN (DCGAN) (Radford *et al.* (2015)), have worked to further stabilize and improve GANs, but complete stabilization is still an open problem. In a DCGAN, the discriminator is set up similarly to a deep convolutional neural network (CNN) (LeCun *et al.* (1995)) and utilizes powerful convolutional layers to learn highly discriminatory hierarchical features that downsample input to be more compact and abstract at each layer. The generator, on the other hand, utilizes transposed convolutional, or deconvolutional, layers (Zeiler *et al.* (2011)) that upsample a compact, abstract vector at each layer until the last layer, which outputs an image.

2.3 Transfer Learning

A large number of machine learning methods work well when the training and test data are taken from the same distribution and feature space. In most models, changing the test data to some other dataset dramatically affects the success of the task because these models are trained to specifically handle one type of data. Transfer learning aims to produce machine learning methods that reduce this degradation by learning transferable data representations that can be utilized in multiple settings and tasks by learning the underlying, hidden structure of the data.

In machine learning, A domain D is composed of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$. In a domain, $X = x_1, ...x_n \subset \mathcal{X}$ is the set of samples found in the feature space. This thesis includes multiple datasets and variations on these datasets. Each variation of dataset is considered a different domain because they differ to various extents in their feature spaces and marginal probability distributions. A task T is composed of a label space Y and $f(\cdot)$, a function of $f : X \to Y$. This function is originally unknown and must learn to predict the correct label of an input x. Given a source domain D_S with an accompanying task T_S and a target domain D_T with task T_T, transfer learning aims to improve the target predictive function $f_T(\cdot)$ in D_T using knowledge from D_S and T_S (Pan and Yang (2010)).

2.3.1 Domain Adaptation

One type of transfer learning is domain adaptation. In domain adaptation, there is a single task T shared by a source D_S and target D_T domain. In most cases, the label spaces of D_S and D_T are the same, but the two domains differ in their joint probability distributions of $P_S(X,Y)$ and $P_T(X,Y)$ (Bruzzone and Marconcini (2010)). Typically, data in the source domain is labeled, and the target domain has a little to no label information, leaving no straightforward method of estimating $\hat{P}_T(X,Y)$. Since the two domains share a label space, they usually share many attributes and features, especially higher level ones, leaving the possibility to approximate $\hat{P}_T(X,Y)$ by adapting $\hat{P}_S(X,Y)$.

Within domain adaptation, there are two problem scenarios: semi-supervised and unsupervised (Chapelle *et al.* (2009)). In the semi-supervised setting, the source domain contains labeled data $D_S = \{x_i^s, y_i^s\}_{i=1}^{n_s}$ and the target domain contains some labeled and some unlabeled data $D_T = \{x_i^t, y_i^t\}_{i=1}^{n_T} \cup \{x_i^t\}_{i=n_t+1}^{n_t+n_u}$. In the target domain, there are a sparse number of n_t labeled data points and a much larger number of n_u unlabeled points. Due to the sparse number of labeled data points in D_T, $\hat{P}_T(X,Y)$ cannot be estimated without severe overfitting. Semi-supervised methods rely heavily on $\hat{P}_S(X,Y)$ to estimate $\hat{P}_T(X,Y)$, but also make use of the labeled target data to assist in the alignment of $\hat{P}_T(X,Y)$ with $P_T(X,Y)$.

In the unsupervised scenario, the source domain contains labeled data $D_S = \{x_i^s, y_i^s\}_{i=1}^{n_s}$ and the target domain contains only unlabeled data $D_t = \{x_i^t\}_{i=1}^{n_t}$. Since there is no information in the target domain, the estimation of $\hat{P}_T(X,Y)$ solely relies on information gleaned from $\hat{P}_S(X,Y)$. This results in a much harder problem than semi-supervised learning but is much more common, with no requirement for labels in the target domain. Unsupervised models rely on matching and adapting knowledge in the source and target dataset.

Currently, the best performing algorithms utilize deep learning approaches because of the powerful discriminatory features found in their convolutional layers (Goodfellow *et al.* (2016)). As these deep networks get trained, the convolutional layers learn to find the most useful features for whichever loss they are given rather than utilizing handcrafted feature extractors. This automatic process allows deep networks to specialize their layers to extract powerful, specialized features. Another quality about deep neural networks is that they learn to extract hierarchical features, meaning that their earlier layers extract lower-level spaces and find simpler features. In images, this would include edges or curves. Later layers of a deep network extract higher-level, abstract features that are combinations of their lower level features. These abstract features can end up representing complicated shapes, the orientation of an object, and much more.

The powerful features found using deep neural networks are great for the domain adaptation task. Alexnet (Krizhevsky *et al.* (2012)), a deep CNN trained for the challenging ImageNet classification task, learns to extract features in order to classify millions of images as one of a thousand categories. Even though these features are trained for a specific task, the features, to some extent, can be used for other tasks such as domain adaptation (Razavian *et al.* (2014)). Similarly, a target domain can be directly input to a deep network trained on a source domain to some success, called nave domain adaptation. This form of domain adaptation can be further improved by fine-tuning specific layers of the CNN(Oquab *et al.* (2014)).

Another set of deep learning domain adaptation techniques involve aligning domain-invariant features. In networks like the deep adaptation network(DAN) Long *et al.* (2015), the domain adaptive hash (DAH) network (Venkateswara *et al.* (2017)), and the residual transfer network (Long *et al.* (2016)), the models use various losses like maximum mean discrepancy (MMD) loss and domain confusion loss(Tzeng *et al.* (2014)) to align the features found when analyzing both source and target data. Since the features found in source and target data are aligned, a classifier trained on the source domain will have some level of success in the target domain. In a similar fashion, disentangling generative factors can be used for domain adaptation. Since the label space is the same for both domains in domain adaptation, data from both domains typically share most factors that relate to the labels. Another method of feature alignment is to first separate the domain-invariant factors from the domain-variant factors. This separation allows the model to focus only on the domain-invariant factors when aligning domains and performing classification(Bousmalis *et al.* (2016)).

Aside from adapting or transferring features, domain adaptation can be accomplished through adapting or transferring images themselves. With the rise of generative models and new ways to synthesize data, images themselves can be transferred, or translated, into another domain. These translated images can be used to help train a classifier or be classified themselves (Liu *et al.* (2017)).

2.3.2 Image-to-Image Translation and Style Transfer

In the context of transferable data representations, one important application of generative models is image-to-image translation. In this translation, the content of an inputted image is captured and represented in a new, generated image in another domain, resulting in an image that has the same content but a new style. Content, in this context, relates to the main objects of an image and typically relate to an image's label. The style of an image relates to the specifics of how the content is portrayed and can include attributes like color, size, and rotation.

Style transfer can be seen as a version of image-to-image translation that only focuses on conserving and transferring the style elements a single image to another (Gatys *et al.* (2015)). Typically, images going through image-to-image translation may have many plausible outcomes because of the range of styles in a domain, similar to languages that have multiple ways of saying the same thing. On the other hand, style transfer typically transfers the stylistic elements, or the textures, of a singular image onto another (Gatys *et al.* (2016)). To accomplish style transfer, early techniques like (Efros and Leung (1999)) transfer only the low-level features which contain the textures found in images. Deep CNNs expand on early style transfer approaches by adding the capability to achieve some high-level content conservation due to the hierarchical nature of features found in CNNs. With deeper image representations, and sufficient amount of labels, convolutional style transfer provided better looking results.

The goal of cross-domain image-to-image translation is to synthesize image x_T that contains a similar content as image x_S from domain D_S but looks similar to images in domain D_T. This version of image translation requires a comprehensive understanding of the content and style from both domains. The CycleGAN (Zhu *et al.* (2017)), an early approach for cross-domain image-to-image translation, does not explicitly understand the content and style of images it transfers. Instead a CycleGAN utilizes a cyclical loss to ensure that the translation of some image to a new domain and back would result in an image similar to the original image. Due to this lack of an explicit method of content conservation, the domains that the CycleGAN can accurately translate are severely limited. CycleGANs, due to the lack of explicit content conservation and reliance on only a cyclical loss, are unable to perform geometric transformations in image translation. Similar to style transfer techniques, CycleGANs simply transfer textures when translating from one domain to another. More recent approaches to cross-domain image-to-image translation like UNIT (Liu *et al.* (2017)) and CyCADA (Hoffman *et al.* (2017)) utilize semantic loss functions in order to explicitly understand the content of original and translated images. Although these content conservation loss functions severely complicate training, they allow the models to correctly translate images. The better a model is at understanding the content and style of an image, the better the translation.

Chapter 3

LITERATURE SURVEY

3.1 Generative Models

In order to create something, one must first understand how it is put together. With a motto that creation leads to understanding, there has recently been much work to use deep machine learning models to understand data by learning how to generate it.

3.1.1 Generative Adversarial Networks

Generative Adversarial Networks (Goodfellow *et al.* (2014)) are a subset of generative models that utilize adversarial learning to generate data and have exploded in popularity and performance in recent years. A traditional GAN has two components, a generator and discriminator, that are set up to oppose each other. The discriminator acts as a classifier of samples as either real or fake, having come from the original distribution or not. While the discriminator is learning to classify the domains of images, the generator learns how to create realistic images. While working against each other, the discriminator and generator also work together and learn from each other by constantly outdoing the other.

Many works extend the original GAN in order to expand the range of applications, increase performance, and improve training stability. DCGANs improve on the original GAN by utilizing deep convolutional networks for the generator and discriminator (Radford *et al.* (2015)). This shift to deep modeling improves image quality and stability. DCGANs are also shown to model a subset of the generative factors when learning to generate new images. DCGANs are able to find these factors by learning a hierarchy of representations of the data in both the generator and discriminator similar to those found in Convolutional Neural Networks (CNNs).

Another important extension of GAN is the conditional GAN (Mirza and Osindero (2014), which introduces a conditional input to GANs. Prior to adding a controlled variable in the input, the output of a GAN was solely correlated to a random noise vector, resulting in a relationship that was difficult to analyze. The controlled input gives conditional GANs the ability to directly correlate the input of a GAN with the output, resulting in more powerful applications. One model that utilizes these control variables is the information maximizing GAN, or InfoGAN (Chen *et al.* (2016)). InfoGANs are able to map previously unknown attributes of synthesized images from a GAN to interpretable variables in the input in an unsupervised manner. An InfoGAN finds this mapping by maximizing the mutual information in sets of generated images.

3.1.2 Variational Autoencoders

The VAE is a variational inference method that understands images by finding a set of compact, normally distributed latent variables that can describe all of the data in a dataset (Kingma and Welling (2013)). Each VAE is made of two smaller networks, an encoder, that maps data into a compressed latent space, and a decoder, which upsamples the latent variables back into an image. Because of the constraint of compression, the representation of the data in the latent space becomes efficient at modeling the key attributes of the data. One weakness of the VAE is that the resulting images tend to be blurry due to the compression in the latent space.

VAEs and GANs are both successful generative models with their own benefits and deficiencies. In the work on the VAE-GAN (Larsen *et al.* (2015)), researchers combine the two models in order to make a model with the strengths of both. This work combines the powerful representations learned by VAEs with the image quality of GANs resulting in better image quality than the traditional VAE, better training stability than a GAN, and more variation in generated images than typical GANs.

3.1.3 Disentangling Generative Factors

Recently, there has been much work using machine learning models to disentangle the generative factors of data. These generative factors, or factors of variation(Bengio (2012)), relate to different attributes of data that vary independently. If a model learns to understand the different factors of a set of data and also how these factors work together, the model can break down data into its different core aspects for further analysis and modification.

An InfoGAN network disentangles the generative factors by maximizing the mutual information between the observations and a subset of latents (Chen *et al.* (2016)). This information maximization allows the network to map specific generative factors found in the generated images to interpretable variables, effectively disentangling these interpretable variables.

The VAE can be seen as a disentangling network that disentangles its generative factors in the latent space to a certain degree. The disentanglement occurs because the compression forces the VAE to use efficient data representations, and the most efficient data representation comes in the form of disentangled independent generative factors. β-VAE (Higgins *et al.* (2016)) is a modified version of VAE that focuses on the degree of disentanglement of a VAE by varying the weight of the loss based on the KL divergence in order to limit the information flow of the latent space, forcing more efficient and more disentangled factors.

The disentanglement of a β-VAE provides a simple, powerful solution to understanding the composition of various domains and environments. The Symobl-Concept

Association Network, or SCAN (Higgins *et al.* (2017b)), utilizes the disentangled generative factors from a β-VAE to learn a hierarchical composition of the visual concepts found within a domain. Using a small subset of image-symbol pairs, SCAN can also be used to learn the visual concepts associated with certain symbols. The Disentangled Representation Learning Agent (DARLA) (Higgins *et al.* (2017a)) utilizes β-VAE to learn a disentangled representation of observations in a reinforcement learning environment. This robust understanding of the environment allows DARLA to learn a source policy capable of zero-shot domain adaptation.

Another disentangling model is the Adversarial Autoencoder (AAE) (Makhzani *et al.* (2015)), which can learn to disentangle the content and style of an image. Although it does not disentangle at the level of generative factors, an AAE disentangles sets of generative factors. An AAE is set up similar to a VAE, except for the use of an adversarial training procedure in the AAE. This procedure is used to impose a prior distribution on the hidden code rather than the KL divergence used in VAEs. AAEs can learn to disentangle in supervised settings by providing a label vector to the generator of the model, making the content of the generated image rely solely on this inputted vector and having the rest of the AAE handle the style.

3.2 Applications of Deep Generative Models

The rise of deep generative models like GAN and VAE and their various extensions have brought a wide array of new methods to understand and represent data. Generative models can utilize their deep understandings to excel in many challenging tasks.

3.2.1 Image-to-Image Translation

Many models based off of the original GAN have been utilized for cross-domain image-to-image translation. Pix2Pix (Isola *et al.* (2017)) learns to perform supervised image-to-image translation by comparing a generated image with the known translated version of the image. The goal of Pix2Pix is to perform unsupervised translation at a similar level of supervised translation without the requirement of having prior translated pairs of images. Unsupervised image-to-image translation is a tougher task, because there is no straightforward way to check if an image is translated correctly without labels in both domains. The CycleGAN(Zhu *et al.* (2017)) performs unsupervised translation without pairing through a pixel-level cyclical loss that compared an original image with a version of the image that was translated to another domain and then translated back. CycleGANs can demonstrate high-quality unsupervised translation, but the datasets that it is capable of successfully translating are quite limited, especially when geometric transformations are necessary for translation. The Domain Transfer Network (DTN) (Taigman *et al.* (2016)) utilizes an f-constancy term, which encourages an original image and the generated image translation to have similar higher-level features. Compared to the pixel-wise loss of CycleGAN, this loss based on feature-consistency allowed the translation of more domains. The unsupervised image-to-image translator, or UNIT (Liu *et al.* (2017)), achieves unsupervised image-to-image translation by utilizing a pair of coupled VAE-GANs, with each VAE-GAN handling the encoding and decoding of images in its domain. UNIT assumes that the two translated domains can share a common latent space. Because of this assumption, UNIT utilizes weight sharing of the higher-level layers of the two VAE-GANs, forcing both subnetworks to map the data into a shared latent space. Because of this shared latent space, images from both domains can be realized as an image from either domain. Most recently, the Cycle-Consistent Adversarial Domain Adaptation model, or CyCADA(Hoffman *et al.* (2017)) extends the CycleGAN by adding feature consistency losses as well as reclassification losses, similar to the DTN. The resulting model is able to achieve state-of-the art results, but requires much fine-tuning in order to balance the complicated objective loss.

3.2.2 Style Transfer

Style transfer typically refers to a type of image-to-image translation that focuses on transferring the artistic style of one image to another i.e. morphing a cityscape photograph to look like a Van Gogh oil painting. Recent style transfer techniques utilize a combination of convolutional layers and Gram matrices originally proposed by in Gatys *et al.* (2015). Their method combines the content of an image with the style of another by jointly minimizing a content loss based off of the squared-error loss between higher level layers as well as a style loss which is done by comparing Gram matrix statistics. In typical style transfer images, the main shapes of the original is kept consistent while everything else changes.

3.2.3 Domain Adaptation

Aside from generative approaches, two deep domain adaptation methods have risen in popularity: discrepancy-based and discriminative models. The Deep Adaptation Network (DAN) (Long *et al.* (2015)) learns to reduce the domain discrepancy in multiple fully connected layers, making these features more transferable. The Adversarial Discriminative Domain Adaptation network(Tzeng *et al.* (2017)), or ADDA, proposes a discriminative adversarial network that improves unsupervised domain adaptation by combining adversarial learning with discriminative feature learning. ADDA does this by learning a discriminative mapping of target images into the source feature space that must also look realistic to a domain discriminator. The Domain Adversarial Neural Network (DANN) (Ganin and Lempitsky (2014)) added a domain discriminator to the traditional convolutional neural network framework that works to ensure the feature distributions from both the source and target domain would be made similar, resulting in the network finding more domain-invariant features and better classification of the target domain using these features. Another domain adaptation model is the Domain Separation Network, or DSN (Bousmalis *et al.* (2016), which explicitly separates domain-variant and domain-invariant generative factors using various similarity and difference metrics. The isolated domain-invariant factors can then be used to train a classifier using the source domain and labels. This classifier can also be used on the target domain with the assumption that the two domains share features relating to classification.

Recently, deep generative models have raised the standard of unsupervised domain adaptation. Capable unsupervised image-to-image translation networks like DTN, UNIT, and CyCADA can be utilized to perform unsupervised domain adaptation. Rather than adapting layers of a classification network, translation networks adapt target images into the source domain which can then be classified with a classifier network trained on the source dataset.

Chapter 4

SEMI-SUPERVISED ADVERSARIAL TRANSLATOR

Recently, many unsupervised image translation techniques have been proposed including: Pix2Pix, CycleGAN, DTN, and UNIT. One consistent deficiency of these techniques is the ability to handle image-to-image translation of drastic domain shifts when the source domain contains a much simpler feature space than the target domain i.e. black and white vs. color or two-dimensional vs. three-dimensional images. In these cases, the way that the models represents and understands the source domain does not have the capacity to also fully understand the more complicated target domain. Chapter (4), introduces a novel semi-supervised approach to the image-to-image translation task. With the addition of several semi-supervised losses, the semi-supervised adversarial translator (SAT) explores the benefits and complications arising from the addition of a limited number of target labels to image-to-image translation.

In this chapter, the SAT, its architecture, its methods, and its applications are outlined. Section (4.1) introduces the model as well as the notation of the model. Section (4.2) details the setup of the network and provides explanations for different architectural decisions. Section (4.3) details two applications of the SAT, image-to-image translation and domain adaptation. In section (4.4), the model and its performance in different applications are evaluated and discussed. Lastly, section (4.5) summarizes the model, details the contributions, and proposes several extensions.

4.1 Overview

Currently, unsupervised image-to-image translation models still have certain limits. One notable limitation is the translation of simpler domains such as MNIST into a more complicated domain like SVHN. In order to help overcome this obstacle, the SAT utilizes a limited number of labels in the target domain. The SAT is a semi-supervised generative adversarial model for image translation across two domains, a labeled source domain, X_S, and a sparsely labeled target domain, X_T. The variables x_S and x_T represent images taken from their respective domains and are used as inputs to the network with $x_S \sim X_S$ and $x_T \sim X_T$. Every image from the source domain has an associated one-hot label vector y_S taken from label space Y_S with K categories where $y_S \in \{0,1\}^K$ while a limited amount of images in the target domain are associated with a one-hot label vector y_T taken from label space Y_T where $y_T \in \{0,1\}^K$. The network is composed of two coupled VAE-GANs, similar to the UNIT network(Liu *et al.* (2017)), with each VAE-GAN handling the interpretation and synthesis of images from either the source or target domains. For the source domain, the model contains an encoder E_S, a decoder G_S, a domain discriminator $D_{S,D}$, and a category discriminator $D_{S,C}$ subnetwork. Similarly, the VAE-GAN that handles the target domain is composed of an encoder E_T, a decoder G_T, a domain discriminator $D_{T,D}$, and a category discriminator $D_{T,C}$ subnetwork.

For each domain, the encoders take their respective inputs x_S and x_T and maps them into a shared, compressed latent space Z as latent variables z_S and z_T. These variables can then be utilized by either decoder to generate a corresponding image in either domain as $\tilde{x}_S$ and $\tilde{x}_T$. The decoders each are tasked with two roles, reconstruction and translation. When the decoders generate an image in the domain of the original input, they can be seen as a reconstructor with outputs of $\tilde{x}_S^{S\to S}$ and $\tilde{x}_T^{T\to T}$. If

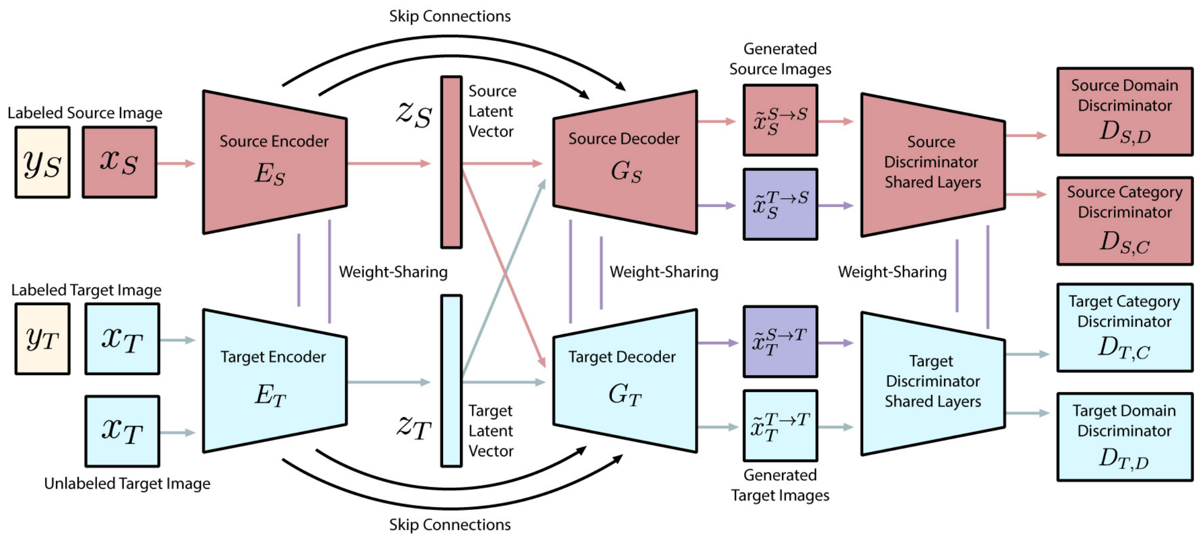

Figure 4.1: The proposed Semi-Supervised Adversarial Translator framework. Labeled source images x_S are encoded into latent vector z_S by source encoder E_S and sparsely labeled target images x_T are encoded into latent vector z_T. The latent vectors z_S and z_T, because of weight sharing throughout the network, can be decoded by either G_S into a source image or by G_T into a target image. Vector z_S can be decoded into $\tilde{x}_S^{S\to S}$ in the source domain and $\tilde{x}_T^{S\to T}$ in the target domain. Similarly, z_T can be decoded into $\tilde{x}_S^{T\to S}$ in the source domain and $\tilde{x}_T^{T\to T}$ in the target domain. Cross-domain image-to-image translation occurs when these latent vectors are decoded into the other domain ($\tilde{x}_S^{T\to S}$ and $\tilde{x}_T^{S\to T}$). Each domain has a multi-purpose discriminator that branches into a category discriminator($D_{S,C}$ and $D_{T,C}$) and a domain discriminator ($D_{S,D}$ and $D_{T,D}$) that ensure that the model is able to synthesize images that conserve the original contents while being stylistically similar to the new domain.

the generated image is in the other domain, the decoders are viewed as a cross-domain translator with outputs of $\tilde{x}_T^{S\to T}$ and $\tilde{x}_S^{T\to S}$. The discriminators, $D_{S,C}$,$D_{T,C}$,$D_{S,D}$ and $D_{T,D}$, are used for classification of an image's label and domain.

4.2 Architecture

4.2.1 Variational Autoencoder

In each domain, the paired encoder and decoder can be viewed as a VAE. For the source domain, E_S and G_S together form VAE_S, which reconstructs image x_S into $\tilde{x}_S^{S\to S}$. The encoder E_S is a deep neural network that acts as a variational inference network, mapping input x_S to latent code z_S in the latent space Z, which is assumed to be conditionally independent and have a normal distribution. Due to the nature of variational autoencoders, this latent code inherently contains compressed information on the content and style of the input image. The encoder E_S, given input x_S, outputs a mean vector $E_{S,\mu}(x_S)$ and variance vector $E_{S,\sigma^2}(x_S)$ which are representative of the latent code. These outputs are then used as the parameters of distribution $q_S(z_S|x_S)$, which is sampled from in order to obtain $z_S \sim q_S(z_S|x_S)$. The decoder G_S is a deep neural network that then takes latent code z_S and attempts to upsample the latent code to reconstruct the original input x_S denoted as $\tilde{x}_S^{S\to S} = G_S(z_S)$. VAE_S is trained by minimizing the variational upper bound of the objective function given by

$$L_{\text{VAE}_S} = \gamma_1 \mathbb{E}_{x_S \sim X_S, z_S \sim q_S(z_S|x_S)}[-\log p_{G_S}(x_S|z_S)] + \gamma_2 \text{KL}(q_S(z_S|x_S)||p(Z)) \qquad (4.1)$$

In the objective function, $p(Z)$ is the prior distribution of Z, which is set to a standard isotropic Gaussian prior $\mathcal{N}(0, I)$. The expression, $\mathbb{E}_{z_S \sim q_S(z_S|x_S)}[-\text{log} p_{G_S}(x_S|z_S)]$, is equivalent to minimizing the Euclidean distance between the original and reconstructed image. The KL stands for Kullback-Leibler divergence, which is a measure of disparity between the goal probability distribution of $p(Z)$ and the encoder's distribution of $q_S(z_S|x_S)$. The hyper-parameters γ control relative weights of the reconstruction and divergence losses and are varied throughout experimentation.

A similar procedure occurs for VAE_T in the target domain, which has an objective function given by

$$L_{\text{VAE}_T} = \gamma_1 \mathbb{E}_{x_T \sim X_T, z_T \sim q_T(z_T|x_T)}[-\log p_{G_T}(x_T|z_T)] + \gamma_2 \text{KL}(q_T(z_T|x_T)||p(Z)) \qquad (4.2)$$

4.2.2 Domain Coupling

This model also utilizes weight sharing, or coupling, between the respective subnetworks of each domain. Domains being translated using this model are assumed to

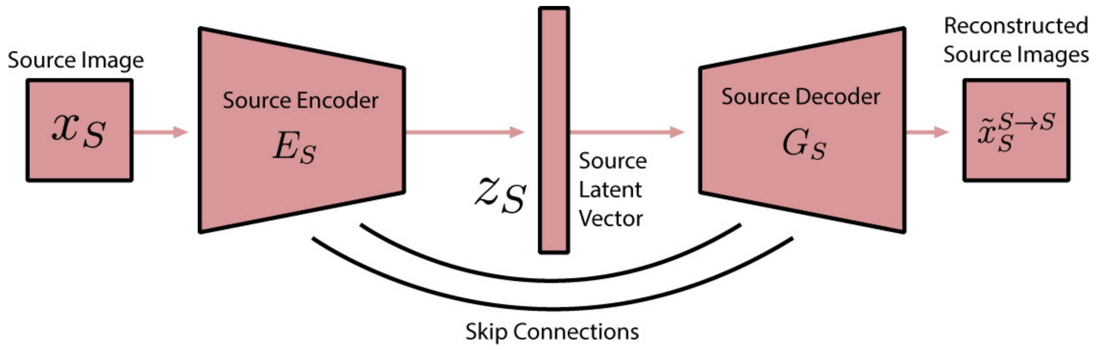

Figure 4.2: A section of the proposed SAT framework that illustrates the source VAE subnetwork. The source and target VAEs act as the generators of the SAT. Similar to traditional VAEs, the latent vector is pushed to have an isotropic Gaussian distribution and a reconstruction loss is utilized to ensure correct original images are correctly represented in the latent vector. A similar subnetwork handling target data can also be found in the SAT

share some characteristics like having similar and corresponding classes in each domain. This similarity allows the assumption that the domains used also share many higher level features that correspond to the category and overall structure of the image. Because of this assumption, the subnetworks from each domain share their higher level layers with their corresponding subnetworks from the other domain. This weight sharing forces the higher level features that are evaluated to be similar in each domain. Each VAE also makes use of skip connections between some of the shared layers of the encoder and decoder. Skip connections allow information to bypass the normal flow of data and send outputs of one layer directly to a layer further down the network. These skip connections allow some lower level information to bypass the information bottleneck found in the compressed latent vector. These skip connections are ultimately able to preserve some features specific to each domain and ultimately allow more faithful reconstructions and translations.

4.2.3 Generative Adversarial Network

Attached to each VAE is a multi-purpose discriminator that provides feedback to the generator concerning both the realism of the generated image as well as its content. The discriminator is a deep convolutional network that branches out into two subnetworks, a domain discriminator and a categorical discriminator. Each branch shares the weights of their early layers but also utilize unique layers used to perform their specific function.

4.2.3.1 Domain Discriminator

The domain discriminator has the same role as the discriminator in Goodfellow's GAN (Goodfellow *et al.* (2014)). In the source domain, the network learns to discriminate x_S, which come from the discriminator's domain, from images reconstructed or translated into the source domain, $\tilde{x}_S^{S\to S}$ and $\tilde{x}_S^{T\to S}$. In the SAT model, each domain discriminator forces it's associated decoder to generate domain-like images from compressed latent vectors originating from either domains. The source domain discriminator accomplishes this by utilizing the objective function given by

$$\begin{aligned} L_{GAN_{S,D}} =& \gamma_3 \mathbb{E}_{x_S \sim X_S}[\log D_{S,D}(x_S)] \\ &+ \gamma_4 \mathbb{E}_{z_S \sim q_S(z_S|x_S)}[\log(1 - D_{S,D}(G_S(z_S)))] \\ &+ \gamma_4 \mathbb{E}_{z_T \sim q_T(z_T|x_T)}[\log(1 - D_{S,D}(G_S(z_T)))] \end{aligned} \tag{4.3}$$

The first term ensures that inputs from the original source domain are categorized as source images. The second and third term tries to maximize the probability that generated images originating from the source as well as target domain are classified as source images. Similarly, the target domain discriminator is trained with an objective function given by

$$\begin{aligned} L_{GAN_{T,D}} =& \gamma_3 \mathbb{E}_{z_T \sim p_{X_1}}[\log D_{T,D}(x_T)] \\ &+ \gamma_4 \mathbb{E}_{z_T \sim q_T(z_T|x_T)}[\log(1 - D_{T,D}(G_S(z_T)))] \\ &+ \gamma_4 \mathbb{E}_{z_S \sim q_S(z_S|x_S)}[\log(1 - D_{T,D}(G_S(z_S)))] \end{aligned} \tag{4.4}$$

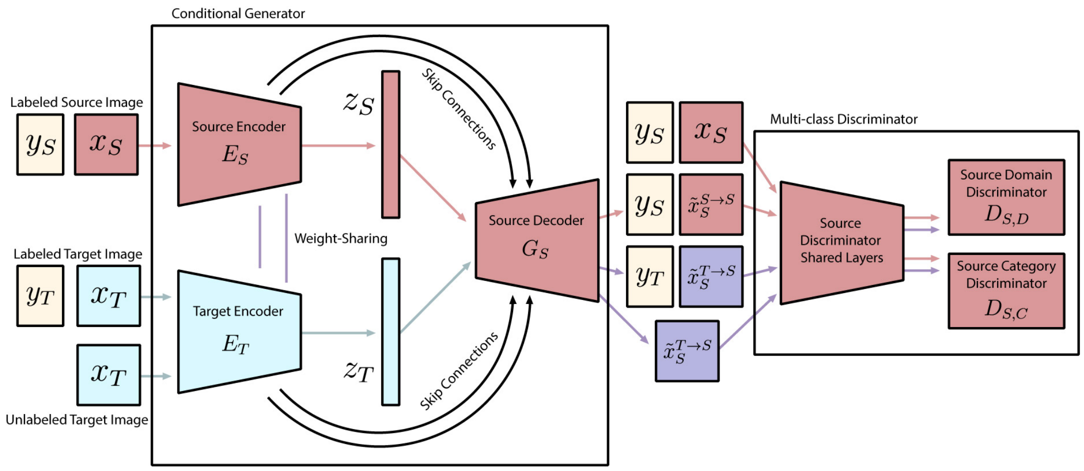

Figure 4.3: Source Multi-class Conditional Generative Adversarial subnetwork found in the proposed SAT framework. The conditional source generator, composed of both encoders and the source decoder of the SAT network, acts as the generator in the typical GAN setup opposed to a source discriminator that classifies both the category and domain of generated images in the source domain. A similar subnetwork handling target data can also be found in the SAT

4.2.3.2 Category Discriminator

The category discriminators classify the category of the input image for their associated domain. The outputs of the category discriminator are used to force the variational autoencoders to maintain the original input's class through reconstruction or translation. In the source domain, the category discriminator is trained to correctly classify original source images through minimizing a cross-entropy term between the predicted conditional distribution $\mathbf{p}(y_S|x_S, D_{S,C})$ and the true label y_S given by

$$\text{CE}[y_S, \mathbf{p}(y_S|x_S, D_{S,C})] = -\sum_{i=1}^{K} y_{S,i} \log p(y_{S,i}|x_S, D_{S,C}) \tag{4.5}$$

Where $\mathbf{p}(y_S|x_S, D_{S,C}) = [p(y_{S,1}|x_S, D_{S,C}), p(y_{S,2}|x_S, D_{S,C}), ..., p(y_{S,K}|x_S, D_{S,C})]^T$. Also, $y_{S,i} = 1$ if the label of x_S is i. Otherwise, $y_{S,i} = 0$. The cross entropy equation is then put into a larger objective function designed to force the generators and discriminator to generate images that are correctly classified and is given by

$$\begin{aligned} L_{GAN_{S,C}} =& \gamma_5 \mathbb{E}_{x_S \sim X_S, y_S \sim Y_S}\Big[\mathrm{CE}[y_S, \mathbf{p}(y_S|x_S, D_{S,C})]\Big] \\ &+ \gamma_6 \mathbb{E}_{x_T \sim X_T, y_T \sim Y_T, z_T \sim q_T(z_T|x_T)}\Big[\mathrm{CE}[y_T, \mathbf{p}(y_T|G_S(z_T), D_{S,C})]\Big] \\ &+ \gamma_7 \mathbb{E}_{x_T \sim X_T, \tilde{y_T}, z_T \sim q_T(z_T|x_T)}\Big[\mathrm{CE}[y_{\tilde{T}}, \mathbf{p}(y_T|G_S(z_T), D_{S,C})]\Big] \end{aligned} \tag{4.6}$$

The first term of the equation forces the categorical discriminator correctly classifies images taken from the original source domain. The second term pushes the model to translate labeled images from the target domain into the source domain with the correct category. The last term, which usually has a weighting much lower than the previous terms, obtains a pseudo-label $\tilde{y}_T$ of an unlabeled target image by using the target's categorical discriminator and then pushes the model to translate the target image into an source-like image classified as the pseudo-label.

The target's categorical discriminator is trained similarly to the source categorical discriminator with the exception of the use of pseudo-labeled target images. Due to the nature of the semi-supervised problem, there are few labels to train the target categorical discriminator. This is taken into account when assigning weights to the components of the objective function. The objective function is given by

$$\begin{aligned} L_{GAN_{T,C}} =& \gamma_5 \mathbb{E}_{x_T \sim X_T, y_T \sim Y_T}\Big[\mathrm{CE}[y_T, \mathbf{p}(y_T|x_T, D_{T,C})]\Big] \\ &+ \gamma_6 \mathbb{E}_{x_S \sim X_S, y_S \sim Y_S, z_S \sim q_T(z_S|x_S)}\Big[\mathrm{CE}[y_S, \mathbf{p}(y_S|G_T(z_S), D_{T,C})]\Big] \end{aligned} \tag{4.7}$$

4.3 Semi-Supervised Image-to-Image Translation

The SAT model is able to translate images between the source and target domains by using the encoder of the original domain and the decoder of the opposite domain.

Translation using the SAT is achieved with a compound objective function utilizing losses from the variational autoencoders and the domain and category discriminators as well as the use of weight sharing between corresponding subnetworks.

In general, successful image-to-image translation has two requirements. The first is to maintain content consistency between the original input and the generated image. The second requirement is to create an image that could have originated from the target domain. The losses of the SAT work together to maximize both of these requirements . The reconstruction loss and regularization loss ensure that encoders and decoders are learning to encode and decode inputs from their respective domain. When reconstructing, the domain discriminators push the decoders to produce higher fidelity images. When translating, the domain discriminators push the decoders to learn to decode encoded images from another domain and translate these into pictures the decoder's domain. Using the regularization, reconstruction, and discriminator losses results in an unsupervised method of translating images similar to Liu *et al.* (2017). However, like most adversarial networks, training is a highly volatile training environment, with different losses pulling the model into opposing directions, causing failures like mode-collapse or a loss of class-consistency during training. Utilizing categorical losses in the generators for both the source and target domains helps push the cross-domain translator in the right direction. By associating images with their labels, the categorical losses ensure that labeled images are always classified as their original label despite the domain. Without the domain coupling or categorical discriminator, mixing these subnetworks results in unintelligible or class-inconsistent outputs.

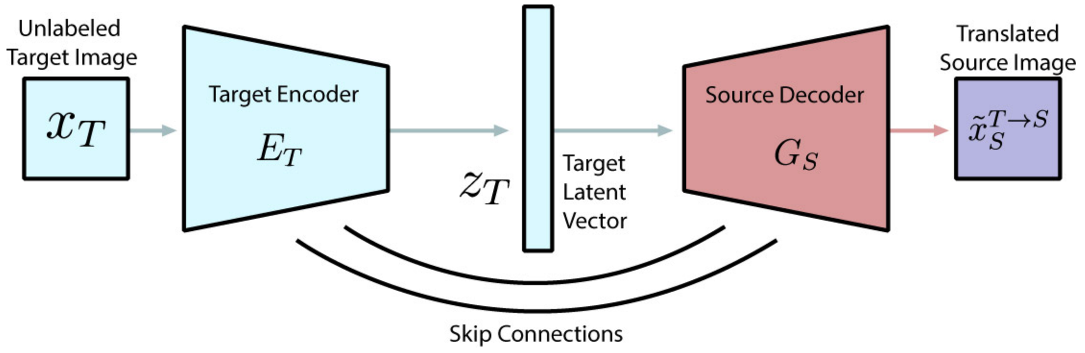

Figure 4.4: Illustration of the process of cross-domain image-to-image translation from a target domain into a source domain. In order to perform image-to-image translation, an image is first mapped to its latent vector. The decoder then decodes a combination of the latent vector and information from the skip connects into an image in new domain.

4.4 Semi-Supervised Domain Adaptation

The translation model can also be used for semi-supervised domain adaptation on the target domain. Using only the sparse amount of labeled target samples results in severe over-fitting and poor performance in a classifier. However, the larger number of labeled source images can be utilized for the target domain by translating target data into the source domain for classification.

4.5 Training

Training the SAT network is achieved through jointly optimizing the compound objective function from the VAE and discriminator subnetworks. In a similar fashion to a GAN, the discriminators are trained to maximize the probabilities of assigning the correct class and domain while training the generator networks made of the encoders and decoders to be labeled correctly and fool the discriminator. This is achieved by having the discriminator and generator play a two-player minimax game with a value function V(E_S,E_T,G_S,G_T,$D_{S,D}$,$D_{T,D}$,$D_{S,C}$,$D_{T,C}$):

$$\begin{aligned}\min_{[E_S,E_T,G_S,G_T,D_{S,C},D_{T,C}]}\ \max_{[D_{S,D},D_{T,D}]} V(E_S,E_T,G_S,G_T,D_{S,D},D_{T,D},D_{S,C},D_{T,C}) = \\ L_{\text{VAE}_S}+L_{\text{VAE}_T}+L_{GAN_{S,D}}+L_{GAN_{T,D}}+L_{GAN_{S,C}}+L_{GAN_{T,C}}\end{aligned} \tag{4.8}$$

The model is trained similarly to the original GAN in a repeating two-step procedure Goodfellow *et al.* (2014). In the first step, the parameters of the discriminators are adjusted to optimize their part of the objective function. In the next step, the parameters of the encoders and decoders are trained.

Loss	**Affected Subnetworks**	**Role**
Reconstruction	Encoders and Decoders	Ensures decoders are faithful to the original input
Kullback-Leibler Divergence	Encoders	Ensures latent feature space maintains a normal distribution
Domain	Encoders, Decoders, and Discriminators	Ensures reconstructions and translations look realistic
Categorization	Encoders, Decoders, and Discriminators	Ensures the content of the original input is preserved

Table 4.1: Descriptions of the different losses of the SAT Model

4.6 Experiments

This section outlines some of the experiments conducted to test the SAT model. The experiments were conducted using Tensorflow 1.2.1 on an Intel Core-i5 3.9 GHz paired with an Nvidia GTX 1070.

4.6.1 Datasets

The datasets used are cropped SVHN, MNIST, and MNIST-V, a modified MNIST. Digit datasets are used for because they contain relatively simple images and have a consistent amount of categories. SVHN, or the Street View House Numbers Dataset, contains real-world images of house numbers taken from Google Street View. SVHN contains the most complex pictures of the tested domains because of the variation found in real-world images(color,font, angle, size, background, clarity). MNIST contains normalized and fixed-size handwritten digits. This normalization causes there to be much less variation in the domain. MNIST-V contains every image from MNIST with random scale shifts and random pixel value multipliers. MNIST-V is used to examine the changes in translation and domain adaptation caused by the similarity and complexity of domains.

4.6.2 Semi-Supervised Translation

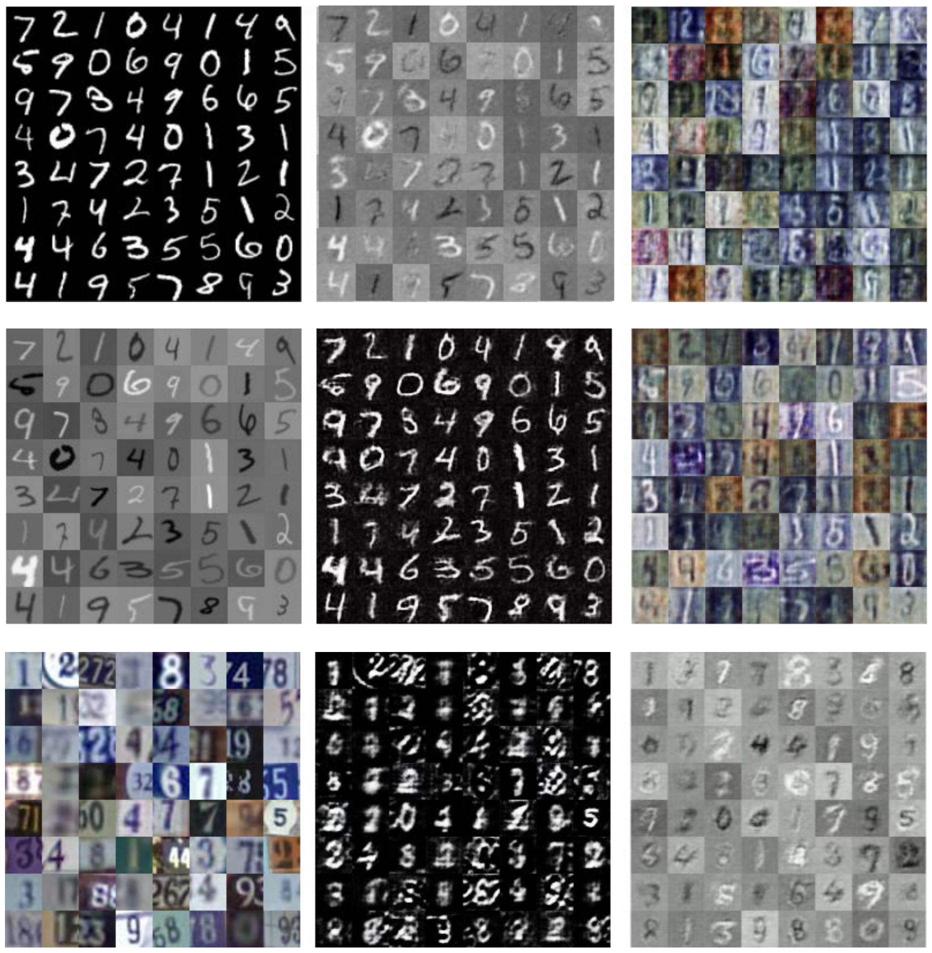
Figure 4.5: Original and translated versions of images obtained using the SAT. Top row: MNIST(left) translated into MNIST-V(middle) and SVHN (right). Middle row: MNIST-V(left) translated into MNIST(middle) and SVHN(right). Bottom row: SVHN(left) translated into MNIST(middle) and MNIST-V(right)

To evaluate the translation capabilities of the SAT model, an SAT network is trained to translate a pair of the MNIST, MNIST-V, and SVHN datasets. In these results, images from the source domain are translated into a target domain that had 100 labels, or 10 labels for each category. Each combination of the three datasets provides some meaningful insights into the way the model performs as the choice of source and target domains dramatically affects translation performance.

The first set of results found in Figure 4.5 are the translations from source MNIST images into the MNIST-V and SVHN domains. Translating from MNIST, the domain with the least variation and complexity, into MNIST-V yields generated images that exhibit the same digit as the original. Similar to the MNIST-V dataset, the new images exhibit a range of background colors and colors of the digits. Translating MNIST into SVHN achieves relatively good content consistency, but translated images do not necessarily look like images from the SVHN domain. The colors of the translated images are consistent with other images of the domain, but the style of the numbers are still more similar to an MNIST image. Perfect content consistency and domain consistency may be achievable with the model, but would require extensive fine-tuning the various parameters of the network. Throughout experimentation, content consistency is given priority when setting the parameters and result is a visible trade-off with the domain consistency or image fidelity. This priority is because this content consistency is not present in current unsupervised image-to-image translation models. Typically, these models can produce higher quality images, but with the severe trade-off of not correctly translating the original image. The translation of MNIST images into SVHN images is the most difficult of these sets of translations because of the steep domain shift between the simple source domain and complicated target domain. SVHN labels are much more valuable than MNIST labels because each SVHN label is associated with images that contain much more information and variation. But with the little amount of SVHN labels, the SAT learns to decently preserve the content. Similar unsupervised solutions, like the UNIT model, will generate very realistic images, but with incorrect digits. An example of this content inconsistency during translation is shown in Figure 4.6

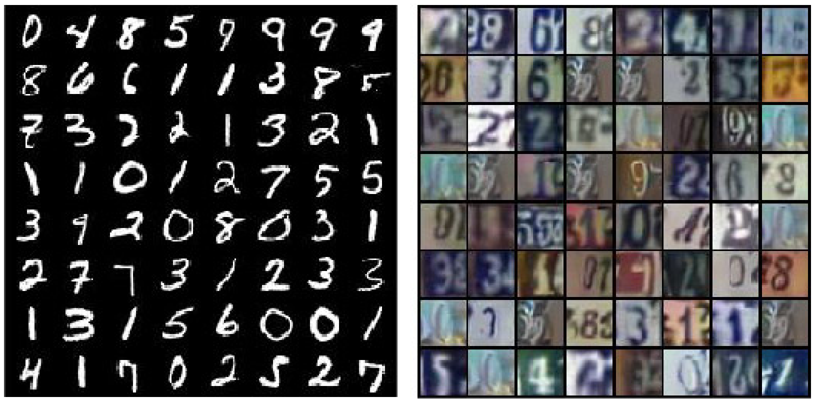

Figure 4.6: Translation of MNIST data(left) into SVHN domain(right) using a UNIT network.

The next set of results are the translations from MNIST-V into MNIST and SVHN. Translating from MNIST-V into MNIST, for the most part yields great results. The digits in the generated images look similar to the digits of the original image except with a black background and white digit. These results are the best example of translation because the source is more complex than the target and the domain shift between MNIST-V and MNIST is relatively small compared to the shifts between other sets of domains. Translating images from MNIST-V into SVHN yields higher quality images than the translation of MNIST into SVHN because of the added complexity in MNIST-V. The variety in MNIST-V that is introduced when adding noise to the original MNIST dataset forces the SAT to learn a deeper understanding of MNIST-V, which in turn makes the data representation more robust and better suited to understand the target SVHN dataset.

The last set of translation results are the translations from SVHN into MNIST and MNIST-V. Translating from SVHN into MNIST yields some great images that preserve the original content, but a subset of these translations contain large amounts of artifacting. In this case, the target decoder is not able to handle the large number of variations of the SVHN dataset's background and foreground color. Since the MNIST dataset only contains images with a black background and a white digit, the SAT model learns to handle one variation of image, resulting in the artifacting present in the translated images. Translating from SVHN into MNIST-V yields images that preserve the original digit and contain much less artifacting than the translations into MNIST. The SAT achieves better results because the variation and complexity found in the MNIST-V dataset is able to express much more of the variation found in the SVHN dataset.

In these different translation scenarios, the SAT is able to find a shared latent space from which it is able to generate reconstructions or translations of images. This success shows that the latent variables are able to contain powerful data representation that can be used for generating new images that exhibit the same digit, but with a style more similar to another domain. Because of this shared representation, the SAT is also able to transfer knowledge of labels in the source domain into the target domain, allowing category consistency to be present in image translation. Without this transfered knowledge, these generated translations would not know which digit is being portrayed. These translations, however, are not perfect. In training, it was found that image fidelity and category preservation were hard to achieve simultaneously. A further increase of the weight of the losses relating image fidelity would result in more realistic images but a decline in the content-consistency. On the other hand, increasing the weight on categorical preservation would result in worsened image quality. Overall, the quality of translated images diminished with the addition of a categorical loss in the target domain, but the content-consistency was drastically better, especially in situations of a complicated target domain.

4.6.3 Semi-Supervised Domain Adaptation

Target Domain	Target Labels	CNN	Deep Features	SAT(MNIST)	SAT(MNIST-V)	SAT(SVHN)
MNIST	50	60.34	61.45	-	**92.40**	67.14
MNIST	100	73.29	68.23	-	**92.66**	76.27
MNIST	200	78.73	79.40	-	**93.15**	82.59
MNIST	1000	88.65	90.42	-	**94.29**	90.67
MNIST-V	50	25.90	51.03	50.79	-	**57.84**
MNIST-V	100	31.63	58.35	55.23	-	**64.28**
MNIST-V	200	41.10	67.85	59.16	-	**71.04**
MNIST-V	1000	77.70	85.30	**85.62**	-	80.47
SVHN	50	16.51	19.05	23.8	**48.33**	-
SVHN	100	20.28	28.03	27.2	**52.55**	-
SVHN	200	33.08	30.66	37.0	**54.97**	-
SVHN	1000	57.72	44.12	62.6	**69.31**	-

Table 4.2: Recognition accuracies (%) on the MNIST, MNIST-V, and SVHN datasets across multiple semi-supervised algorithms with varying amounts of labeled target data. CNN is a Convolutional Neural Network with an architecture similar to LeNet. Deep Features trains a CNN using deep features of the input obtained from VGG-16 trained on imagenet. SAT is the proposed Semi-Supervised Adversarial Translator utilizing various source domains. SAT(MNIST) implies MNIST is the source domain. The best results are emboldened.

To evaluate the domain adaptation capabilities of the SAT model, an SAT network was trained to translate a pair of the MNIST, MNIST-V, and SVHN datasets. In these results, images from the target domain are translated into a source domain. These translated In the experimentation, the entire source dataset had associated labels while the target dataset had a varying number of labels.

Since there is currently little work on semi-supervised domain adaptation in generative models, the classification results of the SAT on target data is compared to a classifier based on a convolutional neural network trained with similar numbers of labels as well as a classifier that utilizes deep features obtained from a pre-trained imagenet network and a similar number of labels. Results from the SAT consistently outperform these two methods.

Further domain adaptation comparisons were made with the unsupervised UNIT model to analyze the effects of the addition of target labels and losses in Table 4.3. For instances of target domains being much more complicated than the source like MNIST $\rightarrow$ SVHN and MNIST-V $\rightarrow$ SVHN, there are very clear improvements on performances. However, in other situations, the semi-supervised domain adaptation of the SAT is worse than the unsupervised domain adaptation of the UNIT network. This decrease in performance can be attributed to the additional complexity introduced by the target labels. Adding more losses to an already complicated model gives more sub-tasks to each component of the network. In some cases, this helps the network model more complicated domain shifts, but in many cases, this pulls the weights in different directions.

Source Domain	**Target Domain**	**UNIT**	**SAT-50**	**SAT-200**
MNIST	MNIST-V	98.07	50.79	59.16
MNIST	SVHN	23.46	23.82	37.20
MNIST-V	MNIST	99.01	92.40	93.15
MNIST-V	SVHN	29.00	48.33	52.55
SVHN	MNIST	83.52	51.03	82.59
SVHN	MNIST-V	85.23	57.84	71.04

Table 4.3: Recognition accuracies (%) of the UNIT and SAT models on images from the target domain while utilizing knowledge and labels from the source domain. The numbers appended to SAT reference the total number of labels used during training.

4.7 Conclusions and Summary

Although there are a limited amount of semi-supervised generative models to compare to, the experimental analysis indicate that the SAT is capable of handling the tasks of image-to-image translation and domain adaptation with various complexities of datasets. Because of a compound objective function composed of a variety of losses, the SAT is able to, with some level of success, represent data from multiple domains using a shared latent space and then utilize this space to translate and reconstruct images from either domain.

Because of inherent differences in domains, attempting to fully represent the different generative factors of multiple domains in a compressed latent space can be a challenging task. As seen in the evaluation on semi-supervised translation, despite the use of labels in a target domain, perfect domain consistency and label consistency were never able to simultaneously be present. This indicates that there is some conflict between these two objectives stemming from how they are represented in the latent space. Fully containing the information necessary to recreate two drastically different domains in a single shared latent space can be challenging and requires much fine-tuning. In the next chapter, an approach is analyzed where the task of transferable data representation is made easier by splitting up the representation into related sets.

Chapter 5

EXPLICIT DISENTANGLING NETWORK

As a direct response of the complications that arose in the SAT, the explicit disentangling network (EDN) was proposed to simplify how data is represented in a generative model while maintaining and improving on the transferability. By removing unstable adversarial training and explicitly segmenting images into multiple components, the EDN successfully learns transferable representations of multiple domains of data that can be applied to various machine learning tasks.

In this chapter, the EDN, its architecture, its methods, and its applications are outlined. Section (4.1) introduces the model and its key ideas. Section (4.2) details the setup of the network and provides explanations for different architectural decisions. Section (4.3) details two applications of the EDN, image-to-image translation and domain adaptation. In section (4.4), the model and performance in the different applications are evaluated and discussed. Section (4.5) summarizes the model, details the contributions, and proposes several extensions.

5.1 Overview

The performance of a machine learning task heavily depends on how data is represented (Bengio *et al.* (2013)). This is because the method of data representation has a profound impact on how we interact with the data. A single piece of data can be represented in an exponential number of ways. For example, a simple image can be broken down into countless parts, each describing a characteristic of the image. If this image is described to a person as its specific RGB values, they will have no idea what the image looks like. Whereas, if the image is represented by describing the image's different characteristics, they could have a clear picture of the image in their head. Both representations describe the same data, but the form of the representation can drastically limit the ways of interacting with the data as well as the potential uses of the created representation.

A recent approach to represent data more effectively in machine learning models has been to disentangle the underlying generative factors of data. These generative factors are the independent attributes found within a domain of data. A successful disentangled representation means that the features in the data relating to a specific generative factor are separated from unrelated features. The Symbol-Concept Association Network (Higgins *et al.* (2017b)), or SCAN, disentangles data using a β-VAE (Higgins *et al.* (2016)) and then learns to associate compositions of these disentangled generative factors with visual concepts. This association allows SCAN to describe data using different visual concepts as well as create new data with specified combinations. The Disentangled Representation Learning Agent (Higgins *et al.* (2017a)), or Darla, learns to create a disentangled representation of its environment that it leverages to outperform conventional baselines in different zero-shot domain adaptation scenarios. InfoGAN (Chen *et al.* (2016)) is an extension of the GAN that is capable of disentangling a subset of the generative factors. InfoGAN disentangles these factors by maximizing the mutual information between a fixed small subset of the GAN's noise variables and the observations in the generated image.

An explicitly disentangled representation extends the disentangled representation by separating generative factors in such a way that a model can perform tasks on specific factors. This allows models to specialize their architecture and methods in order to best work with each separated set of factors. Unlike SCAN and DARLA, explicit disentanglement requires previous knowledge of the data in order to know which factors to separate, but this knowledge is utilized to increase stability and expand the potential uses of the disentangled factors.

One instance of explicit disentanglement is separating content-related generative factors from style-related generative factors in images. The content of an image is the information related to the ideas expressed in the data, and the style of images relates to how the idea is expressed in the image. Either set of generative factors may be more useful for different aspects of tasks. For example, the content related data is strongly related to the data needed for the classification task. Separating the two sets of factors allows specific objectives and parameters to be assigned to each set in order to more efficiently utilize the data.

This chapter introduces the Explicit Disentangling Network (EDN), a novel method that explicitly disentangles the content and style of images. The EDN contains two VAEs to process content and style related factors as well as a separate decoder to combine the outputs of both. The EDN constricts and restricts the flow of information in the model using various losses and a specific network architecture. This control allows the EDN to separate content and style factors into two different pathways in the network and also allows each set of factors to be independently modified. The performance of this explicit disentanglement is demonstrated by constructing new images generated using a combination of different disentangled style and content vectors. Since content-related generative factors are domain-invariant in similar domains, an EDN is able to disentangle the content and style from data originating from multiple domains in both supervised and unsupervised settings. The performance of the EDN in multiple tasks with multiple domains is demonstrated by utilizing the model for cross-domain image-to-image translation as well as unsupervised domain adaptation. Section 2 details the architecture and methods of the EDN. Section 3 includes the experiments on the measure of disentanglement, the style-transfer capabilities, and domain adaptation performance of the DVN. Section 4 discusses conclusions and directions for future works.

5.2 Architecture

The EDN is a generative model that is capable of disentangling the content and style of images from multiple domains. Each EDN contains two VAE-like subnetworks that do not reconstruct their inputs themselves. Rather, the subnetworks have their outputs combined into a final decoder that then generates a final, new image. These VAEs learn how to deconstruct images into a content vector and a style vector, which can be recombined to reconstruct the original image or combined with other vectors to generate a new image. The main feature of this model is that the styles of the inputted images become disentangled and unpaired from the content of the image, allowing the model to mix content vectors from source and target inputs with randomly generated style vectors or style vectors extracted from an image in the target domain. Moreover, after training, the network for content extraction can also be used for unsupervised domain adaptation and other machine learning tasks.

The EDN model utilizes multiple specialized variational autoencoders to separate data. The model takes images x_S and one-hot labels $y_S \in \{0,1\}^K$ from a source domain X_S and label space Y_S, with K referring to the number of categories in the label space. The model also takes images x_T from a target domain X_T and with labels $y_S \in \{0,1\}^K$ from label space Y_T. In the model, there is a variational autoencoder specialized for content, VAE$_C$, that learns to extract the content from the images in the source and target domain and a variational autoencoder for style, VAE$_S$, that extracts information concerning the specific style of target images. The outputs from these VAEs can be combined and used by a target decoder, D_T, which will generate a final target image, $\tilde{x}_T^{v_1,v_2\rightarrow T}$. In this notation, v_1 represents the source of the content vector, v_2 represents the source of the style vector, and T represents the target domain.

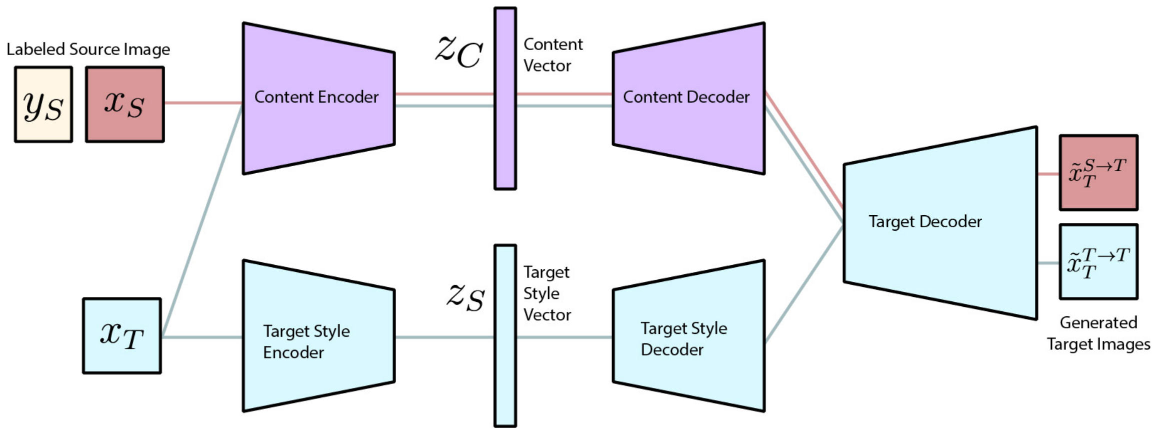

Figure 5.1: The proposed Explicit Disentangling Network framework. The content encoder explicitly separates the content-related generative factors of source images x_S or target images x_T and encodes them into a content vector z_C. The style encoder extracts the style of target image x_T and encodes it into style vector z_S. The decoders can take any combination of available content and style vectors and create an image with content similar to the input to the content encoder and style similar to the input to the style encoder. This ability to generate any combination of vectors is due to the explicit disentanglement of content factors from style factors. The disentanglement is possible because of the branching network architecture as well as several losses constricting and restricting the flow of information.

5.2.1 Variational Autoencoder

There are two variational autoencoders in the EDN model. Each are similar to a typical VAE in all aspects other than the structure of the decoder, which has two distinct stages. The first stage contains network layers specific to each VAE and the second stage merges the layers of the two VAEs. The content VAE, VAE_C, is made of an encoder E_C and decoder D_C and the style VAE, VAE$_S$, is made of encoder E_S and decoder D_S. Encoder E_C takes x_S or x_T and encodes them into content vector $z_{C,S}$ or $z_{C,T}$, depending on the input domain, and E_S takes in x_T and encodes it into style vector $z_{S,T}$. The content encoder E_C, given source data x_S as input for the content vector, outputs a mean vector $E_{C,\mu}(x_S)$ and variance vector $E_{C,\sigma^2}(x_S)$ which are representative of the latent, compressed content vector. These outputs are then used as the parameters of distribution $q_C(z_{S,C}|x_S)$, which is sampled from in order to obtain $z_{S,C} \sim q_S(z_{S,C}|x_S)$. Similarly, E_C can be given target data x_T to find mean vector $E_{C,\mu}(x_T)$ and variance vector $E_{C,\sigma^2}(x_T)$ and $q_C(z_{T,C}|x_T)$. The style encoder E_S, can similarly be given target data x_T to find mean vector $E_{S,\mu}(x_T)$ and variance vector $E_{S,\sigma^2}(x_T)$ and $q_C(z_{T,S}|x_T)$. The decoders then take their related input vectors and upsample them into a higher dimensional space using deconvolutional layers. Upsampled vectors from both decoders are then used by the final decoder, D_T, to create either $\tilde{x}_T^{S\to T}$ or $\tilde{x}_T^{T\to T}$, depending on the domains of the style and content vectors.

Inherent to VAE networks is a compound objective function composed of a KL divergence component and a reconstruction component. The first component aims to minimize the KL divergence of between the goal distribution of $p(Z)$, which is set to a standard isotropic Gaussian prior $\mathcal{N}(0, I)$, and the distributions from the outputs of E_C and E_S. The reconstruction component aims to minimize the pixel-wise difference of x_T and $\tilde{x}_T^{T\to T}$. The resulting objective function specific to VAE losses is given by

$$\begin{aligned} L_{VAE} =& \gamma_1 \mathbb{E}_{x_T \sim X_T, z_{S,T} \sim q_S(z_{S,T}|x_T), z_{C,T} \sim q_C(z_{C,T}|x_T)}[-\log p_{D_T}(\tilde{x}_T^{x_T, x_T\to T}|z_{S,T}, z_{C,T})] \\ &+ \gamma_2 \text{KL}(q_C(z_{C,T}|x_T)||p(Z)) \\ &+ \gamma_2 \text{KL}(q_C(z_{C,S}|x_S)||p(Z)) \\ &+ \gamma_2 \text{KL}(q_S(z_{S,T}|x_T)||p(Z)) \end{aligned} \tag{5.1}$$

5.2.2 Disentangling Style and Content

Separating the style and content of images is achieved through a specific network architecture, and an objective loss containing reconstruction and content consistency losses. First, separate networks, VAE$_C$ and VAE$_S$ are used to extract and encode the content and style respectively. VAE$_C$, in addition to the overall VAE losses, is tasked with classifying x_S and associated y_S, with an objective function given by:

$$L_{Classification} = \gamma_3 \mathbb{E}\Big[\text{CE}[y_S, \mathbf{p}(y_S|x_S)]\Big]. \tag{5.2}$$

In the equation, CE$[y_S, \mathbf{p}(y_S|x_S, D_{S,C})]$ relates to the cross entropy between the input's associated label y_S and the prediction of the network,$\mathbf{p}(y_S|x_S)$, which is set up to be $\mathbf{p}(y_S|x_S, D_{S,C}) = [p(y_{S,1}|x_S, D_{S,C}), p(y_{S,2}|x_S, D_{S,C}), ..., p(y_{S,K}|x_S, D_{S,C})]^T$. The prediction is obtained by adding a classification layer that branches off the second to last layer of E_C. Because of this classification loss, the content that is being recognized relies heavily on features needed for classifying images in the source domain. This also means that the success of this content and style extraction rely heavily on the fact that features used for classification can be used for both source and target domains, an assumption common in domain adaptation.

In order to make sure this content information is kept consistent throughout the image generation process, a content consistency loss is used. During training, a target image is constructed using target image x_{T1} as the source of the style and target image x_{T2} as the source for the content. When combined, the EDN generates $\tilde{x}_T^{x_{T1},x_{T2}\rightarrow T}$, which is then subsequently sent back into the content encoder to obtain the activations at a specified layer of the content encoder, $F(\tilde{x}_T^{x_{T1},x_{T2}\rightarrow T})$. These activations are then compared to the activations of the original input of the content encoder, $F(x_{T1})$, with $d(F(x_T), F(\tilde{x}_T^{x_{T1},x_{T2}\rightarrow T}))$ representing the Euclidean distance between the activations from the two inputs. The resulting objective function is then given by the total difference between the features of the original input and the reconstruction:

$$L_{Content} = \gamma_4 \sum_{x_{T1}.x_{T2}\sim X_T} d(F(x_{T1}), F(\tilde{x}_T^{x_{T1},x_{T2}\rightarrow T})) \tag{5.3}$$

This content consistency loss ensures that the content of the image is only dependent on the information from VAE$_C$ by enforcing that the content of generated images be consistent despite changing the style vector.

In the EDN model, there is no explicit style loss. Instead, the network learns to model the style of the image due to the restriction and constriction of the information flow throughout the network. First, the weight, γ_3, of the above classification loss in the content encoder is set to be much higher than the weight of the reconstruction loss, γ_1, allowing little information beyond what is necessary for classification to flow through VAE$_C$. This restricts style-related information from flowing through the content encoder and forces information not pertaining to the content to flow through VAE$_S$. In order to ensure VAE$_S$ does not completely take over the reconstruction task and contain information for both the content and style, the size and capability of the network must be restricted so that reconstruction relies on both VAE$_S$ and VAE$_C$.

5.3 Training

The training of EDN occurs by iteratively minimizing a compound loss made of the three KL divergence losses, the target image reconstruction loss, the classification loss, and the content consistency loss. The complete objective function is given by:

$$\begin{aligned}
L_{EDN} =& \gamma_1 \mathbb{E}_{z_{S,T} \sim q_S(z_{S,T}|x_T), z_{C,T} \sim q_C(z_{C,T}|x_T)}[-\log p_{D_T}(\tilde{x}_T^{x_T, x_T \to T} | z_{S,T}, z_{C,T})] \\
&+ \gamma_2 \text{KL}(q_C(z_{C,T}|x_T)||p(Z)) \\
&+ \gamma_2 \text{KL}(q_C(z_{C,S}|x_S)||p(Z)) \\
&+ \gamma_2 \text{KL}(q_S(z_{S,T}|x_T)||p(Z)) \\
&+ \gamma_3 \mathbb{E}\Big[\text{CE}[y_S, \mathbf{p}(y_S|x_S)]\Big] \\
&+ \gamma_4 \sum_{x_{T1}.x_{T2} \sim X_T} d(F(x_{T1}), F(\tilde{x}_T^{x_{T1}, x_{T2} \to T}))
\end{aligned} \tag{5.4}$$

Throughout the different contexts during the evaluations of the EDN, the weights,

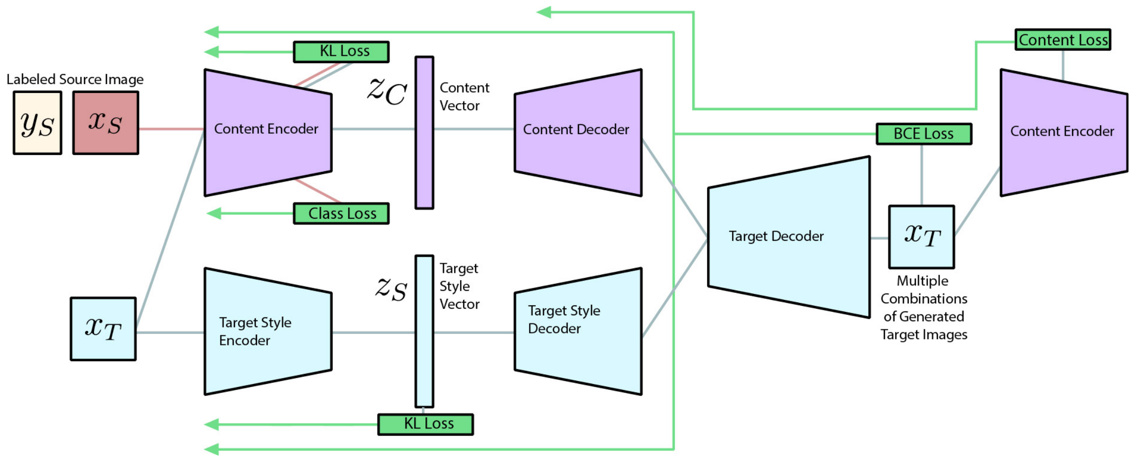

Figure 5.2: Illustration of flow of various losses in the EDN. The Kullback Leibler divergence losses ensure that the encoders generate content and style vectors that are normally distributed. The class loss ensures that the content encoder learns to extract and find content-related features and also restricts the flow of style information. The Binomial Cross-Entropy loss (BCE) acts as the reconstruction loss and ensures a target image can be reconstructed if input as the content and style inputs and affects the whole network. The content loss ensures that the content of images synthesized with various combinations of content and style vectors is conserved. The content loss is found by comparing the activations of the original image and synthesized image in the content encoder.

γ_1,γ_2,γ_3,γ_4 are kept constant. What must change is the size of the latent style vector. In order to achieve the correct flow of information, the vector must be adjusted so that it is large enough to contain the generative factors of the target domain, but must also be limited so that the classification information does not flow through the style encoder.

5.4 Unsupervised Domain Adaptation

In order to disentangle the content and style of images, the EDN requires a powerful content extractor. This extractor can also be used for great performance in unsupervised domain adaptation. VAE$_S$ specializes in reading and extracting portions of the input directly related to classification. Labels in the source domain allow E_C to find the features necessary for classification of source images. However,

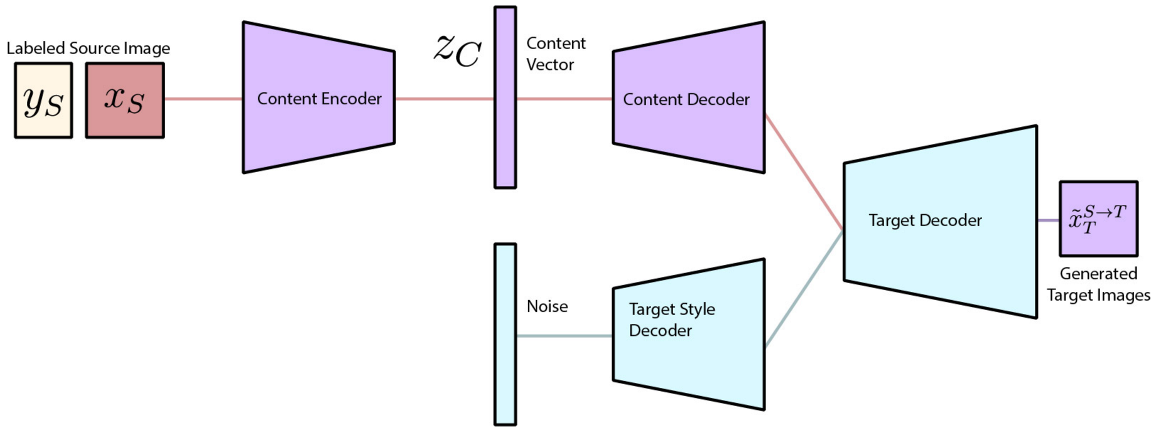

Figure 5.3: Translating source images into target images using random noise. An alternative to using images as inputs to the encoders is to use randomly sampled vectors. This is possible because the KL Divergence losses ensure that these vectors are normally distributed.

there will always be some level of domain shift between two different domains. The EDN model addresses this discrepancy by applying KL divergence losses on both the source and target data that are sent through the content encoder. This loss forces the content-related features of the network to model the high level features of both source and target data with similar normal distributions, further aligning the higher level features of both domains. Aside from content extraction, the VAE$_S$ must also help in image generation. Because of the model's overall reconstruction loss, E_C is required to extract the content information of target data required for re-creating target images. Because of these different factors, E_C can learn to understand the content of both the source and target domains despite the lack of target labels to some success.

5.5 Image-to-Image Translation

The EDN model approaches the image-to-image translation task by breaking it up into smaller problems, content and style transfer. Content transfer is achieved through VAE$_C$, which specializes in reading and extracting the parts of the input directly related to the classification of the labeled source images. The style transfer aspect of image-to-image translation relates to creating an image that could have originated from a different domain and is achieved through VAE$_S$.

Image-to-image translation can occur in a EDN in two ways. Given a trained EDN network and x_S, an image from a source domain, the extracted content from x_S from the VAE$_C$ network can be paired with a randomly sampled style vector or an extracted style vector from x_T, an image from the target domain. Utilizing this content and style vector for image generation results in a translated version of x_S that has the style contained in style vector. Because of this change in style, along with a transfer of content, the newly synthesized image can be considered transfered to the target domain. This method is notably different than the SAT, UNIT, CycleGAN, and DTN translation methods by allowing a single image to be translated into a new domain in multiple ways because of the explicitly disentangled understanding of content and style.

Unsupervised image-to-image translation can be performed in the EDN by using a single classification loss for the source domain in E_C with some success. For more consistent success in translation, another classification loss for the target domain may be added, although then the classification of the task must switch to supervised image-to-image translation.

5.6 Evaluation

This section outlines some of the experiments conducted to test the EDN model. The experiments were conducted using Tensorflow 1.2.1 on an Intel Core-i5 3.9 GHz paired with an Nvidia GTX 1070.

5.6.1 Datasets

With a focus on finding relationships across image domains, our experiments focus on two large digit datasets, MNIST[1], which contains 70000 handwritten digits that are size-normalized, centered, and black and white, as well as a cropped version of SVHN[2], which contains over 600000 digits taken from color photos of house numbers. MNIST is a simpler dataset, with no background noise and simpler digits while the SVHN datasets contains images with different, multi-colored backgrounds and a variety of fonts and styles. Along with the original versions of MNIST and SVHN, the EDN is evaluated on the MNIST-N and MNIST-R datasets. MNIST-N is a copy of MNIST with added noise in the form of inverting half of the images to introduce more variation and complexity. MNIST-R[3], is a version of MNIST whose digits are randomly rotated.

These datasets were chosen for multiple reasons. The first reason is that they all share a common label space of the digits from 0-9. A variety of datasets and variations are used in order to examine how the network performs under different degrees and directions of domain shift. Lastly, these datasets were chosen due to their large number of data. Domains consisting of a wider array of images and labels, like the Office-Home dataset, were considered, but no deep generative transfer models have been successful converge during training with these datasets due to the lack of associated data for each label.

5.6.2 Disentangling Performance

In order to demonstrate that the content-related information of a trained EDN is being sent through VAE$_C$ and the style-related information is being sent through

[1]`http://yann.lecun.com/exdb/mnist/`

[2]`http://ufldl.stanford.edu/housenumbers/`

[3]`http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations`

VAE_S, the content and style was classified in images synthesized using different combinations of inputs. Complete disentanglement would lead to the content of the generated image being solely determined by the content input and the style to be solely determined by the style input.

In this evaluation, the content label was classified by inputting a synthesized image with specific content and style inputs into a CNN trained to classify the domain of the input. The style accuracy was evaluated by using a nearest neighbor classifier based on euclidean distances to a set of 10 original style images. Aside from using images from the domain, randomized content and style vectors were also used during the evaluation process.

Content Input	**Style Input**	**Classification Accuracy using A's digit**	**Classification Accuracy using A's style**
A	B	80.82%	10.80%
A	R	82.12%	8.97%
B	A	12.44%	100.00%
R	A	11.79%	100.00%

Table 5.1: Classification of style and digit of a generated images with various inputs. A and B are subsets of SVHN was used while R indicates a random vector was used during generation. Digit accuracy was evaluated using a simple neural network trained to classify SVHN while style accuracy was evaluated using a nearest neighbor classifier based off of euclidean distances to the set of original style images.

Table 5.1 demonstrates that successful classification of the class label is only present if labeling the class of the content input and not the style input. Otherwise, the classification scores are similar to randomly guessing. Similarly, the style is found to be solely dependent on the style vector. Since this network is shown to explicitly disentangle the content and style related generative factors, this model can further utilize its powerful transferable data representation for other useful tasks.

5.6.3 Unsupervised Domain Adaptation

Because E_C of the EDN is able to perform explicit inference of content-related features, E_C can perform efficient unsupervised domain adaptation. VAE$_C$ is trained to handle three different tasks, classification of source data, content extraction for reconstruction of target data, and reclassification of generated data. When the VAE$_C$ learns to handle input from multiple domains simultaneously, it finds a saddle point where these three tasks perform well. In this middle-ground, the extractor learns to find the domain-invariant features for content extraction without the use of labels in the target domain. Classification of unlabeled target images can occur by creating a classifier on the domain-invariant content features of the source domain and then use this classifier on images from the target domain.

In Table 5.2, results of unsupervised domain adaptation are compared between EDN and other unsupervised domain adaptation methods using different combinations of of digit datasets to evaluate the different model's performances in different scenarios. In most of the combinations of source and target domains, the EDN performs better or comparable to the other methods.

The most recent of the baseline comparisons is the UNIT network, which uses coupled VAE-GANs that attempt to model multiple domains using a single latent space. The two large differences between EDN and UNIT is the simplicity of the EDN due to a lack of adversarial training as EDN's capability of explicitly segmenting generative factors in order to increase performance. The separation of generative factors in the EDN allows it to learn more efficient data representations of separate style and content factors that perform well in unsupervised domain adaptation. The EDN performs especially well in comparison to other models when source domain is more simple than the target domain. When transferring knowledge, the ability to

Source	**Target**	**EDN(proposed)**	**UNIT**	**DANN**	**DAN**
M	MN	67.60%	**98.07**%	57.38%	66.72%
M	MR	**61.72**%	15.70%	22.14%	41.29%
M	SVHN	**53.84**%	23.46%	17.81%	27.85%
MN	M	97.67%	**99.01**%	93.59%	98.59%
MN	MR	**37.18**%	24.99%	20.10%	35.07%
MN	SVHN	**55.89**%	29.00%	22.64%	47.07%
MR	M	**95.44**%	93.07%	42.38%	93.22%
MR	MN	57.03%	**70.26**%	23.49%	53.05%
MR	SVHN	**25.14**%	12.04%	18.56%	19.92%
SVHN	M	**83.50**%	57.29%	33.38%	59.20%
SVHN	MN	85.20%	**88.25**%	46.69%	62.30%
SVHN	MR	**30.80**%	17.25%	24.37%	26.54%

Table 5.2: Classification accuracies (%) on the MNIST, MNIST-N, MNIST-R and SVHN datasets across multiple unsupervised domain adaptation algorithms. CNN is a Convolutional Neural Network with an architecture similar to LeNet. Deep Features trains a CNN using deep features of the input obtained from VGG-16 trained on imagenet. SAT is the proposed Semi-Supervised Adversarial Translator utilizing various source domains. SAT(MNIST) implies MNIST is the source domain. The best results are emboldened.

disentangle the content related vectors, which are more domain-invariant, reduces the domain-shift between the domains of the factors relating to the classification task. This reduction in domain-shift is more necessary in the challenging simple domain to more complicated domain situation, resulting in the EDN performing relatively well.

5.6.4 Image-to-Image Translation

Another application of the EDN is image-to-image translation, where the network learns to transfer the style of images by sending the original image into the content input and sending an image with the desired style through the style input. When transferring the style of images from the same domain, the content encoder only

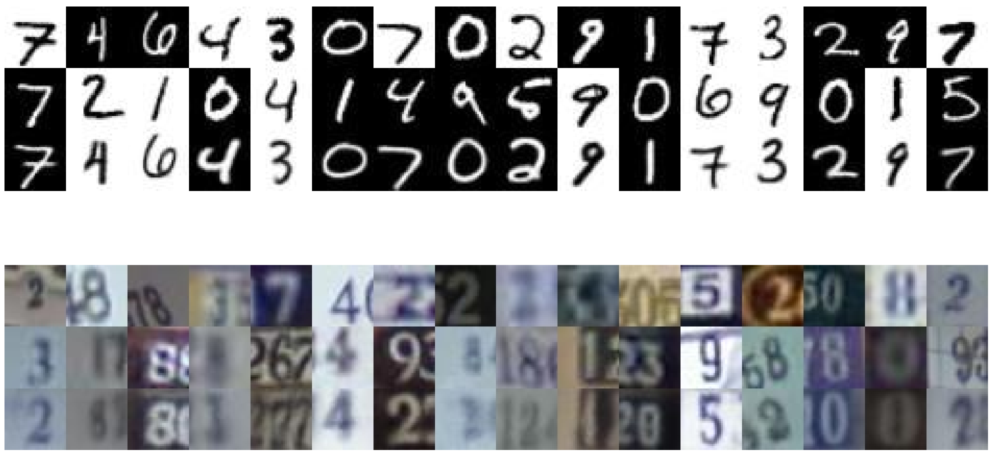
Figure 5.4: Style transfer between images in the same domain. The top image illustrates style transfer in the MNIST-N dataset and the lower image illustrates style transfer in the SVHN dataset. Top row: Source content image. Middle row: Target style image. Bottom row: Generated image

encounters a single dataset, allowing it to have strong performance in its roles as a classifier as well as an encoder for image generation and reconstruction.

The EDN also has the capability of performing image-to-image translation on images from the same domain as well as two different domains. In cross-domain image-to-image translation, the content encoder is required to extract the content of images from two different datasets. The generated images of both experiments show that content and style are both carried over from their input. Although some of the generated images are blurrier than images typically generated from an adversarial network, the images are clearer than a typical VAE because of the presence of a dedicated stylization subnetwork VAE_S.

In the results of supervised intradomain translation found in figure 5.4, the results of successful disentanglement as well as the ability to swap and change style and content can be seen. Consistent with the results of disentanglement performance, the generated images keep the label of the content image consistent while changing the overall style of the image. This is best observed in the SVHN results, where translated

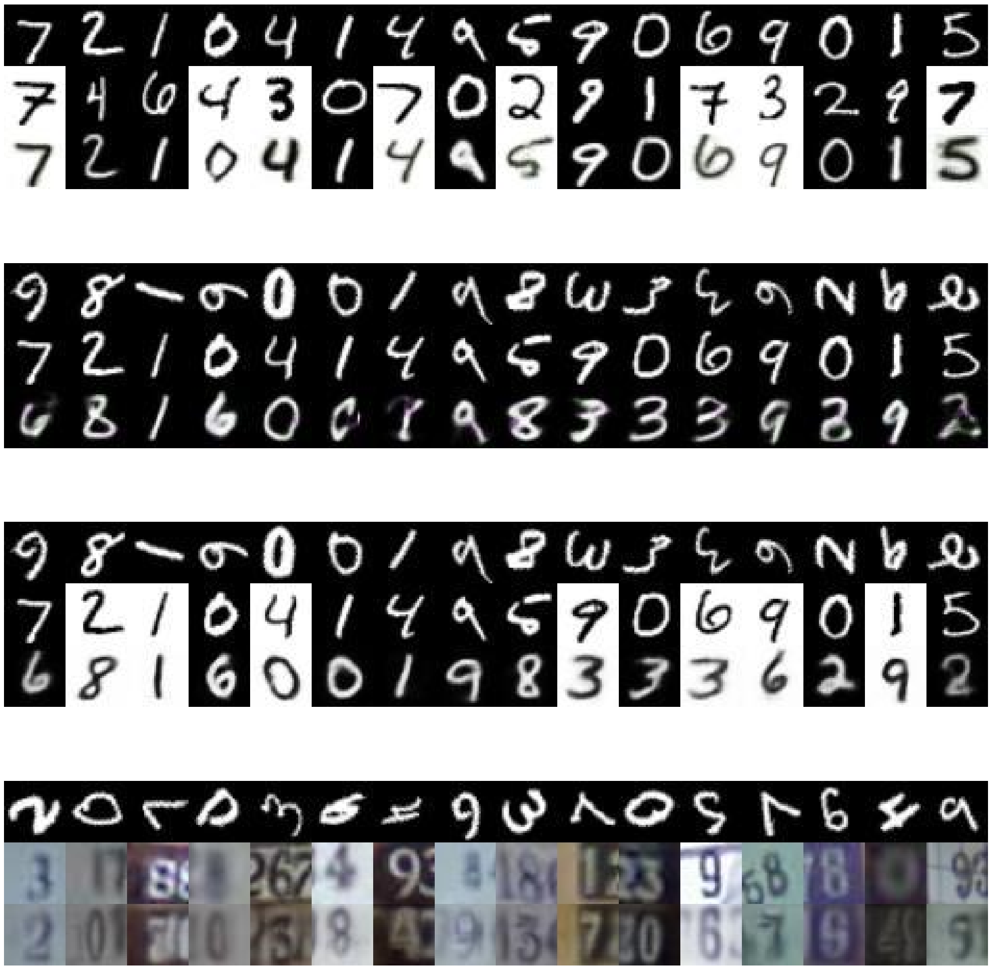
Figure 5.5: Style transfer between images in the different domains. Source and target datasets from top to bottom: MNIST and MNIST-N, MNIST-R and MNIST, MNIST-R and MNIST-N, MNIST-R and SVHN, SVHN and MNIST, SVHN and MNIST-N. Top row: Source content image. Middle row: Target style image. Bottom row: Generated image

images look very similar to the original style image aside from the change in digit. In the MNIST-N dataset, the style transfer can be seen in the preservation of the colors used for the background and digit, but the origin of other aspects of style of the digit is not as easy to infer. These images also demonstrate the EDN's ability to translate images into more than one output. Rather than correlating the edges of an image into another domain, similar to CycleGAN, this translation is able to understand the

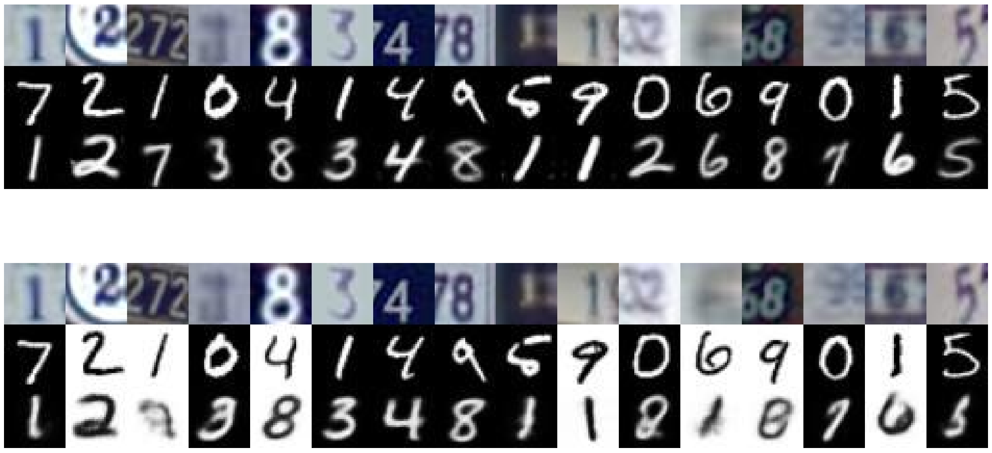
Figure 5.6: Style transfer between images in the different domains. Source and target datasets from top to bottom: MNIST and MNIST-N, MNIST-R and MNIST, MNIST-R and MNIST-N, MNIST-R and SVHN, SVHN and MNIST, SVHN and MNIST-N. Top row: Source content image. Middle row: Target style image. Bottom row: Generated image

content and style of images and knows how to transfer and change both.

Inter-domain translation is a more challenging task, but can be appropriately handled by the EDN if given labels in the target domain. The main difference between inter-domain and intra-domain translation is that E_C must interpret the content of data from multiple domains in a single latent space. Compared to other techniques, the explicit disentangling on the domain-invariant content-related generative factors lets E_C focus on examining the shared factors of the two domains. It must be noted that this translation is successful because it is completely supervised. Training without target labels typically leads to varied levels of mode collapse present in challenging translation problems like MNIST $\rightarrow$ SVHN, a problem found in all current translation models.

5.7 Conclusion

This work presents a new model, the EDN, that explicitly disentangles the generative factors relating to content and style. This disentanglement allows the EDN to perform specific actions on each set of factors, resulting in new, more efficient ways of handling data. The EDN accomplishes this explicit disentanglement by utilizing two unique subnetworks with their own losses and parameters, with one specializing in content extraction and the other in style extraction. This work demonstrates that the proposed model can manipulate these sets of factors to perform unsupervised domain adaptation and supervised image-to-image translation. The domain adaptation performance is comparable and even better than other recently proposed models, and the image-to-image translation results demonstrate some powerful benefits of explicit disentanglements. Although the scope of this works evaluations is limited, this method of explicit disentanglement may be extended to increase performance of more unsupervised problems, handle more domains and data types, and utilized for more types of machine learning tasks.

The next step of the EDN is to segment generative factors in new ways. An interesting idea would be to separate the style extractor of the EDN into two parts, one which would be similar to the current extractor and the other would be similar to work on style-transfer using Gramm matrices. This addition could allow more control and insight into the style aspect of the data. Another method could be to separate the factors based on similarity between multiple domains, similar to DSN(Bousmalis *et al.* (2016)), resulting in explicitly separating domain-variant and domain-invariant factors. Finding different ways to segment data can also lead to different types of data being used, like finance data or datasets modeling human behavior.

In the larger picture, further development of the explicit disentangling approach can allow better overall transfer learning. By categorizing different parts of deconstructed data, future models have a new way to interact and modify data.

Chapter 6

CONCLUSIONS

This thesis aimed to find better ways of learning transferable data representations in machine learning models. It proposed two models, the SAT and EDN, which explored two different methods. Each of these models were evaluated in two tasks that require transferring and adapting knowledge, image-to-image translation and domain adaptation.

The SAT utilized coupled VAE-GANs to learn to represent data from two domains in a common latent space. This common latent space meant that the SAT would interpret data from multiple domains in a similar fashion, giving the model the ability to decode data from both domains into either domain. It extended past research by evaluating the addition of target labels to an unsupervised image-to-image translation model in order to help the model align features relating to categorization. In semi-supervised cross-domain image-to-image tasks, the SAT was able to successfully conserve content in even the most difficult of the domain combinations presented at the expense of requiring some target labels as well as lessened image quality. In semi-supervised domain adaptation tasks, the SAT was demonstrated to improve upon simple semi-supervised domain adaptation methods but only had some marginal improvements. In some cases, the SAT was worse than similar deep unsupervised methods. From experimentation, it was found that the addition of target labels did increase performance of transfer tasks with complicated target domains. However, along with this benefit came the consequence of adding another level of complexity, which is unnecessary in simpler contexts and can lead to degradation in performance.

The EDN, as a direct response to the deficiencies of the SAT model, attempted to simplify previous models. This simplification was achieved by explicitly disentangling incoming data into content-related and style-related generative factors. Without the use of an adversarial loss, an EDN can disentangle these factors, put these factors together to reconstruct inputs, and combine different sets of factors to create a images. The results from the evaluations of the disentanglement, unsupervised domain adaptation, and supervised cross-domain image-to-image translation all demonstrate that the EDN and approach of explicit disentanglement work well and deserve further research. Immediate future plans for the EDN and explicit disentanglement include fine-tuning parameters and network architecture for increased performance, expanding the domains of data and types of data used to include domains like the Office-Home dataset, and expanding types of tasks that explicit disentanglement can work on. Explicit disentanglement can be used by many machine learning models in order to separate certain generative factors in order to apply specialized actions and modifications to specific parts of the data. By having machine learning models separate data into more abstract and more useful concepts, transfer learning will progress further and further and might eventually compete with human intelligence in more and more tasks.

BIBLIOGRAPHY

Bengio, Y., “Deep learning of representations for unsupervised and transfer learning”, in “Proceedings of ICML Workshop on Unsupervised and Transfer Learning”, pp. 17–36 (2012).

Bengio, Y., A. Courville and P. Vincent, “Representation learning: A review and new perspectives”, IEEE transactions on pattern analysis and machine intelligence **35**, 8, 1798–1828 (2013).

Bousmalis, K., G. Trigeorgis, N. Silberman, D. Krishnan and D. Erhan, “Domain separation networks”, in “Advances in Neural Information Processing Systems”, pp. 343–351 (2016).

Bruzzone, L. and M. Marconcini, “Domain adaptation problems: A dasvm classification technique and a circular validation strategy”, IEEE transactions on pattern analysis and machine intelligence **32**, 5, 770–787 (2010).

Chapelle, O., B. Scholkopf and A. Zien, “Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]”, IEEE Transactions on Neural Networks **20**, 3, 542–542 (2009).

Chen, X., Y. Duan, R. Houthooft, J. Schulman, I. Sutskever and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”, in “Advances in Neural Information Processing Systems”, pp. 2172–2180 (2016).

Efros, A. A. and T. K. Leung, “Texture synthesis by non-parametric sampling”, in “Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on”, vol. 2, pp. 1033–1038 (IEEE, 1999).

Ganin, Y. and V. Lempitsky, “Unsupervised domain adaptation by backpropagation”, arXiv preprint arXiv:1409.7495 (2014).

Gatys, L. A., A. S. Ecker and M. Bethge, “A neural algorithm of artistic style”, arXiv preprint arXiv:1508.06576 (2015).

Gatys, L. A., A. S. Ecker and M. Bethge, “Image style transfer using convolutional neural networks”, in “Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on”, pp. 2414–2423 (IEEE, 2016).

Goodfellow, I., Y. Bengio, A. Courville and Y. Bengio, *Deep learning*, vol. 1 (MIT press Cambridge, 2016).

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, “Generative adversarial nets”, in “Advances in neural information processing systems”, pp. 2672–2680 (2014).

Higgins, I., L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed and A. Lerchner, “beta-vae: Learning basic visual concepts with a constrained variational framework”, (2016).

Higgins, I., A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell and A. Lerchner, “Darla: Improving zero-shot transfer in reinforcement learning”, arXiv preprint arXiv:1707.08475 (2017a).

Higgins, I., N. Sonnerat, L. Matthey, A. Pal, C. P. Burgess, M. Botvinick, D. Hassabis and A. Lerchner, “Scan: learning abstract hierarchical compositional visual concepts”, arXiv preprint arXiv:1707.03389 (2017b).

Hoffman, J., E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation”, arXiv preprint arXiv:1711.03213 (2017).

Isola, P., J.-Y. Zhu, T. Zhou and A. A. Efros, “Image-to-image translation with conditional adversarial networks”, CVPR (2017).

Kingma, D. P. and M. Welling, “Auto-encoding variational bayes”, arXiv preprint arXiv:1312.6114 (2013).

Krizhevsky, A., I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in “Advances in neural information processing systems”, pp. 1097–1105 (2012).

Larsen, A. B. L., S. K. Sønderby, H. Larochelle and O. Winther, “Autoencoding beyond pixels using a learned similarity metric”, arXiv preprint arXiv:1512.09300 (2015).

LeCun, Y., Y. Bengio *et al.*, “Convolutional networks for images, speech, and time series”, The handbook of brain theory and neural networks **3361**, 10, 1995 (1995).

Liu, M.-Y., T. Breuel and J. Kautz, “Unsupervised image-to-image translation networks”, arXiv preprint arXiv:1703.00848 (2017).

Long, M., Y. Cao, J. Wang and M. I. Jordan, “Learning transferable features with deep adaptation networks”, arXiv preprint arXiv:1502.02791 (2015).

Long, M., H. Zhu, J. Wang and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks”, in “Advances in Neural Information Processing Systems”, pp. 136–144 (2016).

Makhzani, A., J. Shlens, N. Jaitly, I. Goodfellow and B. Frey, “Adversarial autoencoders”, arXiv preprint arXiv:1511.05644 (2015).

Mirza, M. and S. Osindero, “Conditional generative adversarial nets”, arXiv preprint arXiv:1411.1784 (2014).

Oquab, M., L. Bottou, I. Laptev and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks”, in “Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on”, pp. 1717–1724 (IEEE, 2014).

Pan, S. J. and Q. Yang, “A survey on transfer learning”, IEEE Transactions on knowledge and data engineering **22**, 10, 1345–1359 (2010).

Radford, A., L. Metz and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks”, arXiv preprint arXiv:1511.06434 (2015).

Razavian, A. S., H. Azizpour, J. Sullivan and S. Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition”, in “Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on”, pp. 512–519 (IEEE, 2014).

Taigman, Y., A. Polyak and L. Wolf, “Unsupervised cross-domain image generation”, arXiv preprint arXiv:1611.02200 (2016).

Tzeng, E., J. Hoffman, K. Saenko and T. Darrell, “Adversarial discriminative domain adaptation”, in “Computer Vision and Pattern Recognition (CVPR)”, vol. 1, p. 4 (2017).

Tzeng, E., J. Hoffman, N. Zhang, K. Saenko and T. Darrell, “Deep domain confusion: Maximizing for domain invariance”, arXiv preprint arXiv:1412.3474 (2014).

Venkateswara, H., J. Eusebio, S. Chakraborty and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation”, arXiv preprint arXiv:1706.07522 (2017).

Zeiler, M. D., G. W. Taylor and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning”, in “Computer Vision (ICCV), 2011 IEEE International Conference on”, pp. 2018–2025 (IEEE, 2011).

Zhu, J.-Y., T. Park, P. Isola and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks”, arXiv preprint arXiv:1703.10593 (2017).

APPENDIX A

NETWORK DETAILS

The following section utilize this notation:

Notation	Meaning
CONV	Convolutional layer
DCONV	Transposed Convolutional layer
ReLU	Rectified Linear Unit
CONCAT	Concatenation
N	Neurons
K	Kernel Size
S	Stride
{}	parameters
()	inputs
A $\rightarrow$ B	evaluation of model with A as the source domain and B as the target domain

Table A.1: Notation used when describing the networks

Semi-Supervised Adversarial Translator

Layer	Encoder	Input	Weight-Sharing
E1	CONV{N64,K5,S2}, BatchNorm, LeakyReLU	Input Image	No
E2	CONV{N128,K5,S1}, BatchNorm, LeakyReLU	E1	Yes
E3	CONV{N256,K8,S1}, BatchNorm, LeakyReLU	E2	Yes
E4	CONV{N512,K1,S1}, BatchNorm, LeakyReLU	E3	Yes
Eσ	CONV{N128,K1,S1}, LeakyReLU, softplus	E4	Yes
Eμ	CONV{N128,K1,S1}	E4	Yes
Layer	**Decoder**	**Input**	**Weight-Sharing**
D1	CONV{N512,K4,S2}, BatchNorm, LeakyReLU	Sampling Eσ and Eμ	Yes
D2	CONV{N256,K4,S2}, BatchNorm, LeakyReLU	D1	Yes
D3	CONV{N128,K4,S2}, BatchNorm, LeakyReLU	D2	Yes
D4	CONV{N164,K4,S2)} BatchNorm, LeakyReLU	D3	No
Output	CONV{N3,K1,S1)} BatchNorm, LeakyReLU	D4	No
Layer	**Discriminator**	**Input**	**Weight-Sharing**
DS1	CONV{N128,K5,S1}, ReLU, MaxpoolK2,S2,dropout.3	Output, x_S, x_T	No
DS2	CONV{N256,K5,S1}, ReLU, MaxpoolK2,S2,dropout.5	DS1	Yes
DS3	CONV{N512,K5,S1}, ReLU, MaxpoolK2,S2,dropout.5	DS2	Yes
DS4	CONV{N1024,K5,S1}, ReLU, MaxpoolK2,S2,dropout.3	DS3	Yes
DSC5	CONV{N1024,K2,S1}	DSC4	Yes
DSC6	CONV{N1,K1,S1}	DSC5	Yes
DSD5	CONV{N1024,K2,S1}	DS4	Yes
DSD6	CONV{N10,K1,S1}	DSD5	Yes

Table A.2: Network architecture for evaluation of SAT

Loss	Weight	Input
Target KL	.1	SE$\sigma(x_T)$,SE$\mu(x_T)$
Source KL	.1	CE$\sigma(x_T)$,SE$\mu(x_T)$
Target Reconstruction MSE	1	x_T, Output$\{x_T \to T\}$
Source Reconstruction MSE	1	x_S, Output$\{x_S \to S\}$
S→S Domain Discriminator	.5	Output$\{x_S \to S\}$
T→T Domain Discriminator	.5	Output$\{x_T \to T\}$
S→T Domain Discriminator	.5	Output$\{x_S \to T\}$
T→S Domain Discriminator	.5	Output$\{x_T \to S\}$
S→S Class Discriminator	.5	Output$\{x_S \to S\}$
T→T Pseudo Class Discriminator	.2	Output$\{x_T \to T\}$
S→T Class Discriminator	.5	Output$\{x_S \to T\}$
T→S Pseudo Class Discriminator	.2	Output$\{x_T \to S\}$

Table A.3: Objective loss hyper-parameters for training the autoencoders in experiments with SAT model

Loss	Weight	Input
Target Image Domain	1	SE$\sigma(x_T)$,SE$\mu(x_T)$
Source Image Domain	1	CE$\sigma(x_T)$,SE$\mu(x_T)$
S→S Domain	.5	Output$\{x_T \to T\}$
T→T Domain	.5	Output$\{x_T \to T\}$
S→T Domain	.5	Output$\{x_T \to T\}$
T→S Domain	.5	Output$\{x_T \to T\}$
Source Classifier	.25	Output$\{x_T \to T\}$
Target Classifier	.25	Output$\{x_T \to T\}$
S→T Classifier	.1	Output$\{x_T \to T\}$

Table A.4: Objective loss hyper-parameters for training the autoencoders in experiments with SAT model

Layer	Content Encoder	Input
CE1	CONV{N64,K5,S1}, ReLU, maxpool(S2)	Content Image
CE2	CONV{N128,K5,S1}, ReLU, maxpool(S2)	CE1
CE3	CONV{N256,K5,S1}, ReLU, maxpool(S2)	CE2
CE4	CONV{N512,K5,S1}, ReLU, maxpool(S2)	CE3
CE5	CONV{1024,K5,S1}, ReLU, maxpool(S2)	CE4
CEσ	CONV{N128,K1,S1}, ReLU, softplus	CE5
CEμ	CONV{N128,K1,S1}	CE5
Classifier	CONV{N10,K1,S1}	CE5
Layer	Style Encoder	Input
SE1	CONV{N16,K5,S2}, BatchNorm, LeakyReLU	Style Image
SE2	CONV{N32,K5,S2}, BatchNorm, LeakyReLU	CE1
SE3	CONV{N64,K8,S1}, BatchNorm, LeakyReLU	CE2
SE4	CONV{N64,K1,S1)} BatchNorm, LeakyReLU	CE3
SEσ	CONV{N32,K1,S1}, ReLU, softplus	CE5
SEμ	CONV{N32,K1,S1}	CE5
Layer	Content Decoder	Input
CD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling CEσ and CEμ
CD2	DCONV{N256,K4,S2}, LeakyReLU	CD1
CD3	DCONV{N128,K4,S2}, LeakyReLU	CD2
Layer	Style Decoder	Input
SD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling SEσ and SEμ
SD2	DCONV{N256,K4,S2}, LeakyReLU	SD1
SD3	DCONV{N128,K4,S2}, LeakyReLU	SD2
Layer	Combined Decoder	Input
D1	DCONV{N64,K4,S2}, LeakyReLU	CONCAT(CD3,SD3)
Output	DCONV{N3,K1,S1}, TANH	D1

Table A.5: Network architecture for experiments with the EDN model with source domains of MNIST, MNIST-N, MNIST-R, SVHN and target domains of MNIST, MNIST-N, MNIST-R

Loss	Weight	Input
Target Style KL	.01	SE$\sigma(x_T)$,SE$\mu(x_T)$
Target Content KL	.01	CE$\sigma(x_T)$,SE$\mu(x_T)$
Source Content KL	.1	CE$\sigma(x_S)$,SE$\mu(x_S)$
Reconstruction MSE	2	Output(Content:x_T, Style:x_S)
Source Feature Consistency MSE	35	CE4(x_S), CE4(Output Content:x_S, Style:x_T)
Target Feature Consistency MSE	35	CE4(x_T), CE4(Output Content:x_T, Style:x_T)
Source Classification CE	.2	Classifier(x_S), y_S

Table A.6: Objective loss hyper-parameters for experiments with the EDN model with source domains of MNIST, MNIST-N, MNIST-R and target domains of MNIST, MNIST-N, MNIST-R

Explicit Disentangling Network

SVHN → SVHN

Layer	Content Encoder	Input
CE1	CONV{N64,K5,S1}, ReLU, maxpool(S2)	Content Image
CE2	CONV{N128,K5,S1}, ReLU, maxpool(S2)	CE1
CE3	CONV{N256,K5,S1}, ReLU, maxpool(S2)	CE2
CE4	CONV{N512,K5,S1}, ReLU, maxpool(S2)	CE3
CE5	CONV{1024,K5,S1}, ReLU, maxpool(S2)	CE4
CEσ	CONV{N128,K1,S1}, ReLU, softplus	CE5
CEμ	CONV{N128,K1,S1}	CE5
Classifier	CONV{N10,K1,S1}	CE5
Layer	Style Encoder	Input
SE1	CONV{N28,K5,S2}, BatchNorm, LeakyReLU	Style Image
SE2	CONV{N56,K5,S2}, BatchNorm, LeakyReLU	CE1
SE3	CONV{N112,K8,S1}, BatchNorm, LeakyReLU	CE2
SE4	CONV{N112,K1,S1)} BatchNorm, LeakyReLU	CE3
SEσ	CONV{N28,K1,S1}, ReLU, softplus	CE5
SEμ	CONV{N28,K1,S1}	CE5
Layer	Content Decoder	Input
CD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling CEσ and CEμ
CD2	DCONV{N256,K4,S2}, LeakyReLU	CD1
CD3	DCONV{N128,K4,S2}, LeakyReLU	CD2
Layer	Style Decoder	Input
SD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling SEσ and SEμ
SD2	DCONV{N256,K4,S2}, LeakyReLU	SD1
SD3	DCONV{N128,K4,S2}, LeakyReLU	SD2
Layer	Combined Decoder	Input
D1	DCONV{N64,K4,S2}, LeakyReLU	CONCAT(CD3,SD3)
Output	DCONV{N3,K1,S1}, TANH	D1

Table A.7: Network architecture for experiments with the EDN model with a source domain of SVHN and target domain of SVHN

Loss	Weight	Input
Target Style KL	.01	SE$\sigma(x_T)$,SE$\mu(x_T)$
Target Content KL	.01	CE$\sigma(x_T)$,SE$\mu(x_T)$
Source Content KL	.1	CE$\sigma(x_S)$,SE$\mu(x_S)$
Reconstruction MSE	5	Output(Content:x_T, Style:x_S)
Source Feature Consistency MSE	15	CE4(x_S), CE4(Output Content:x_S, Style:x_T)
Target Feature Consistency MSE	15	CE4(x_T), CE4(Output Content:x_T, Style:x_T)
Source Classification CE	.2	Classifier(x_S), y_S

Table A.8: Objective loss hyper-parameters for experiments with the EDN model with a source domain of SVHN and target domainof SVHN

MNIST, MNIST-N, MNIST-R, SVHN $\rightarrow$ MNIST, MNIST-N, MNIST-R

Layer	Content Encoder	Input
CE1	CONV{N64,K5,S1}, ReLU, maxpool(S2)	Content Image
CE2	CONV{N128,K5,S1}, ReLU, maxpool(S2)	CE1
CE3	CONV{N256,K5,S1}, ReLU, maxpool(S2)	CE2
CE4	CONV{N512,K5,S1}, ReLU, maxpool(S2)	CE3
CE5	CONV{1024,K5,S1}, ReLU, maxpool(S2)	CE4
CEσ	CONV{N128,K1,S1}, ReLU, softplus	CE5
CEμ	CONV{N128,K1,S1}	CE5
Classifier	CONV{N10,K1,S1}	CE5
Layer	Style Encoder	Input
SE1	CONV{N16,K5,S2}, BatchNorm, LeakyReLU	Style Image
SE2	CONV{N32,K5,S2}, BatchNorm, LeakyReLU	CE1
SE3	CONV{N64,K8,S1}, BatchNorm, LeakyReLU	CE2
SE4	CONV{N64,K1,S1)} BatchNorm, LeakyReLU	CE3
SEσ	CONV{N32,K1,S1}, ReLU, softplus	CE5
SEμ	CONV{N32,K1,S1}	CE5
Layer	Content Decoder	Input
CD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling CEσ and CEμ
CD2	DCONV{N256,K4,S2}, LeakyReLU	CD1
CD3	DCONV{N128,K4,S2}, LeakyReLU	CD2
Layer	Style Decoder	Input
SD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling SEσ and SEμ
SD2	DCONV{N256,K4,S2}, LeakyReLU	SD1
SD3	DCONV{N128,K4,S2}, LeakyReLU	SD2
Layer	Combined Decoder	Input
D1	DCONV{N64,K4,S2}, LeakyReLU	CONCAT(CD3,SD3)
Output	DCONV{N3,K1,S1}, TANH	D1

Table A.9: Network architecture for experiments with the EDN model with source domains of MNIST, MNIST-N, MNIST-R, SVHN and target domains of MNIST, MNIST-N, MNIST-R

Loss	Weight	Input
Target Style KL	.01	SE$\sigma(x_T)$,SE$\mu(x_T)$
Target Content KL	.01	CE$\sigma(x_T)$,SE$\mu(x_T)$
Source Content KL	.1	CE$\sigma(x_S)$,SE$\mu(x_S)$
Reconstruction MSE	2	Output(Content:x_T, Style:x_S)
Source Feature Consistency MSE	35	CE4(x_S), CE4(Output Content:x_S, Style:x_T)
Target Feature Consistency MSE	35	CE4(x_T), CE4(Output Content:x_T, Style:x_T)
Source Classification CE	.2	Classifier(x_S), y_S

Table A.10: Objective loss hyper-parameters for experiments with the EDN model with source domains of MNIST, MNIST-N, MNIST-R and target domains of MNIST, MNIST-N, MNIST-R

MNIST, MNIST-N, and MNIST-R $\rightarrow$ SVHN

Layer	Content Encoder	Input
CE1	CONV{N64,K5,S1}, ReLU, maxpool(S2)	Content Image
CE2	CONV{N128,K5,S1}, ReLU, maxpool(S2)	CE1
CE3	CONV{N256,K5,S1}, ReLU, maxpool(S2)	CE2
CE4	CONV{N512,K5,S1}, ReLU, maxpool(S2)	CE3
CE5	CONV{1024,K5,S1}, ReLU, maxpool(S2)	CE4
CEσ	CONV{N512,K1,S1}, ReLU, softplus	CE5
CEμ	CONV{N512,K1,S1}	CE5
Classifier	CONV{N10,K1,S1}	CE5
Layer	**Style Encoder**	**Input**
SE1	CONV{N32,K5,S2}, BatchNorm, LeakyReLU	Style Image
SE2	CONV{N64,K5,S2}, BatchNorm, LeakyReLU	CE1
SE3	CONV{N128,K8,S1}, BatchNorm, LeakyReLU	CE2
SE4	CONV{N128,K1,S1)} BatchNorm, LeakyReLU	CE3
SEσ	CONV{N128,K1,S1}, ReLU, softplus	CE5
SEμ	CONV{N128,K1,S1}	CE5
Layer	**Content Decoder**	**Input**
CD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling CEσ and CEμ
CD2	DCONV{N256,K4,S2}, LeakyReLU	CD1
CD3	DCONV{N128,K4,S2}, LeakyReLU	CD2
Layer	**Style Decoder**	**Input**
SD1	DCONV{N512,K4,S2}, LeakyReLU	Sampling SEσ and SEμ
SD2	DCONV{N256,K4,S2}, LeakyReLU	SD1
SD3	DCONV{N128,K4,S2}, LeakyReLU	SD2
Layer	**Combined Decoder**	**Input**
D1	DCONV{N64,K4,S2}, LeakyReLU	CONCAT(CD3,SD3)
Output	DCONV{N3,K1,S1}, TANH	D1

Table A.11: Network architecture for experiments with the EDN model with a source domain of SVHN and target domains of MNIST, MNIST-N, and MNIST-R

Loss	Weight	Input
Target Style KL	.01	SE$\sigma(x_T)$,SE$\mu(x_T)$
Target Content KL	.01	CE$\sigma(x_T)$,SE$\mu(x_T)$
Source Content KL	.01	CE$\sigma(x_S)$,SE$\mu(x_S)$
Reconstruction MSE	8	Output(Content:x_T, Style:x_S)
Source Feature Consistency MSE	20	CE4(x_S), CE4(Output Content:x_S, Style:x_T)
Target Feature Consistency MSE	20	CE4(x_T), CE4(Output Content:x_T, Style:x_T)
Source Classification CE	100	Classifier(x_S), y_S

Table A.12: Objective loss hyper-parameters for experiments with the EDN model with a source domain of SVHN and target domains of MNIST, MNIST-N, and MNIST-R

APPENDIX B

OFFICE-HOME DATASET

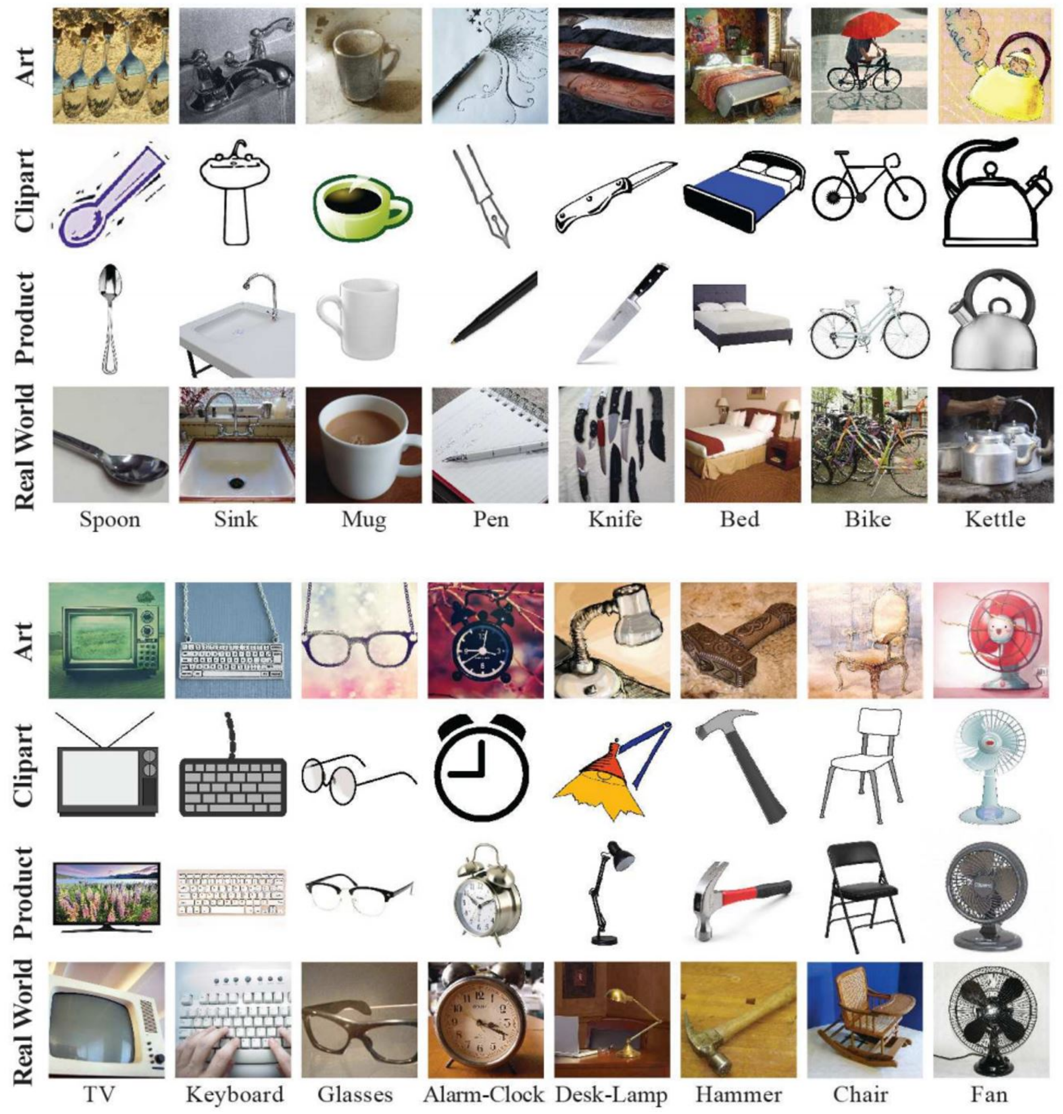

Figure B.1: Examples from the Office-Home Dataset

A new deep learning domain adaptation dataset called the Office-Home dataset was created in conjunction with the other work presented in the thesis. This dataset, along with other work, was published in the paper ”Deep Hashing Network for Unsupervised Domain Adaptation” (Venkateswara *et al.* (2017)). The Office-Home dataset consists of around 15,500 images from 65 categories from 4 domains (art, clipart, product, and real-world) and was created due to the lack of domain adaptation datasets with enough images, domains, and categories to fully train deep domain adaptation models.