# Unsupervised Eye Gaze Estimation using Domain Adaptation

Project-I report submitted to
Indian Institute of Technology Kharagpur
in partial fulfilment for the award of the degree of
Bachelor of Technology
in
Electrical Engineering

**by**
**Sanskar Agrawal, Manu Maheshwari**
**(16EE10041, 16EE10062)**

**Under the supervision of**
**Prof Pabitra Mitra**

**Department of Computer Science and Engineering**
**Indian Institute of Technology Kharagpur**
**Spring Semester, 2019-20**
**April 30, 2020**

# DECLARATION

I certify that

(a) The work contained in this report has been done by us under the guidance of our supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, we have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 30, 2020 (Sanskar Agrawal, Manu Maheshwari)
Place: Kharagpur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

## *CERTIFICATE*

This is to certify that the project report entitled "**Unsupervised Eye Gaze Estimation using Domain Adaptation**" submitted by **Sanskar Agrawal, Manu Maheshwari** to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electrical Engineering is a record of bona fide work carried out by them under my supervision and guidance during Spring Semester, 2019-20.

Prof Pabitra Mitra
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: April 30, 2020

Place: Kharagpur

# Contents

# Chapter 1

# Introduction

## 1.1 Abstract

Domain Adaptation is an actively researched problem in Computer Vision. In this work, we propose an approach that leverages unsupervised data to bring the *source* and *target* distributions closer in a learned joint feature space. We accomplish this by inducing a symbiotic relationship between the learned embedding and a generative adversarial network. We test this approach on the task of eye gaze prediction in which Unity Eyes Dataset serves as the synthetic dataset and MPII Gaze dataset as the test dataset.

## 1.2 Challenges

The common theme across all approaches is the dependence on large amounts of labeled data. While labeled data is available and getting labeled data has been easier over the years, the lack of uniformity of label distributions across different domains results in suboptimal performance of even the most powerful CNN-based algorithms on realistic unseen test data. For example, labeled synthetic data is available in plenty but algorithms trained only on synthetic data perform poorly on real data. This is of vital importance in cases where labeled real data is unavailable. The use of such unlabeled *target* data to mitigate the shift between *source* and *target* distributions is the most useful direction among domain adaptation approaches. Hence we focus on the topic of unsupervised domain adaptation. In this work, we learn an embedding that is robust to the shift between *source* and *target* distributions. We achieve this by using unsupervised data sampled from the *target* distribution to guide the supervised learning procedure that uses data sampled from the *source* distribution. We propose an adversarial image generation approach to directly learn the shared feature embedding using labeled data from *source* and unlabeled data from the target. The novelty of the proposed approach is in using a joint generative discriminative method, the embeddings are learned using a combination of regression loss and a variant of Generative Adversarial Networks.

## 1.3 Related Works

The GAN framework learns two network at the same time with competing loses. It was first introduced by Goodfellow *et al.* [1], since then, many interesting applications and modifications has been proposed [2]. Wang and Gupta [3] learn surface normal and then combine it with StyleGAN. Im *et al.* [4] proposed recurrent generative model using adversarial training. CoGAN by Liu *et al.* [5] use GAN coupling to learn joint distribution from multiple modal images.

### 1.3.1 Unsupervised domain adaptation

Domain Adaptation on inherent features or *latent* space is an area of interest in recent findings of computer vision. Similar and closely related approach is the concept of *Domain Adversarial Networks* [6] by *Ganin et al.* to learn domain invariant features. Both *source* and *target* network shares few common layers for feature adaptation. Our approach in quite different than [6] in the sense that on the first step, we fix the *source* distribution and treat it as stationary, then we try to approximate with *target* distribution which is dynamic through adversarial training. We have more similarities with the original formulation of GAN [1] in which we expect generator to approximate a stationary and natural distribution. Kamnitsas *et al.* [7] also exploited similar approach of [6] for brain lesion segmentation with different dataset. They reported that simultaneous training of domain adversarial loss and *source* loss requires very specific scheduling of training each component.

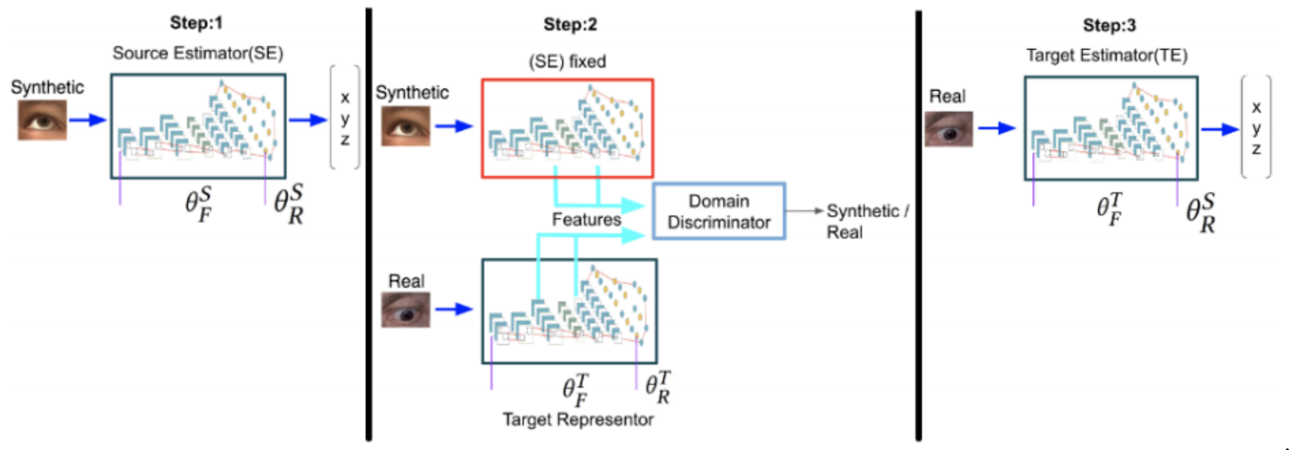


FIGURE 1.1: Three steps of our approach.

As depicted from the above figure, our approach is fairly straight forward. We don't need to treat each component to trigger any other component of training. Ghiffary *et al.* [8] further improved DANN by replacing maximization of domain classification loss by minimization of MMD(Maximum Mean Discrepancy) metric [9] between both domain *latent* space.
Other approaches for feature adaptation is to fix *latent* representations from both *source* and *target* domain and find some subspaces for domain alignment [10, 11]. This type of approach is also used on deep features by CORAL [12], they miminize Frobenius norm between a linear projection of covariance feature matrix of both domain covariance matrix.

### 1.3.2 Learning across synthetic and real domains

Application of domain adaptation using synthetic data has lately gained a lot of interest. Wood *et al.* used UNITY game engine to generate 1 million synthetic eye patch images. They used it to learn eye gaze estimator and achieved state-of-art performance on gaze estimation based on appearance. Now a days, synthetic data from video games are used widely to understand semantics of street videos [13, 14, 15]. This is very useful as collecting all these data is a cumbersome task and impossible sometimes. In [14], the authors simulated video game car crashes for real life crash prediction. All these papers proposed training on synthetic without having access to real data. Recently, Shrivastava *et al.* [16](SimGAN) proposed an adversarial pixel domain adaptation. They used *pixel level refiner* network to transform synthetic and annotated data from UnityEyes to be almost similar to real samples of MPIIGaze in terms of appearance. Any regression network trained on such transformed images are expected to perform better on real samples too. Bousmalis *et al.* [17] proposed pixel level domain adaptation with adversarial loss recently. Ours is a complementary approach to both [17, 16] in terms that we try to adapt *latent* representation of *source* and *target* domain instead of pixel space. We propose that, close similarities in visual appearance between two domains might not necessarily depict near performance of discriminative tasks [2]. Thus it is more intuitive to adapt *latent* features directly as they are discriminative instead of pixel space adaptation. We force *latent* features of both domains to be similar not only on the basis of appearance but also learn tast specific transferable features by utilizing labelled data on *source* domain.

## 1.4 Dataset

We usece then, many improvements and MPIIGaze Dataset for real eye images. It consists of a 213,659 images from 15 participants, while test subset contains 40k images. This dataset is created by asking participants to sit and gaze at a certain point on the screen and recorded corresponding label. Our whole approach in unsupervised and we use labels only for finding the angle error on test subset. For synthetic eye images, we use Unity based software which generates images of eye patch. The 3D gaze of these images are sampled from a normal distribution. We generated around 1 million images for *source* domain.

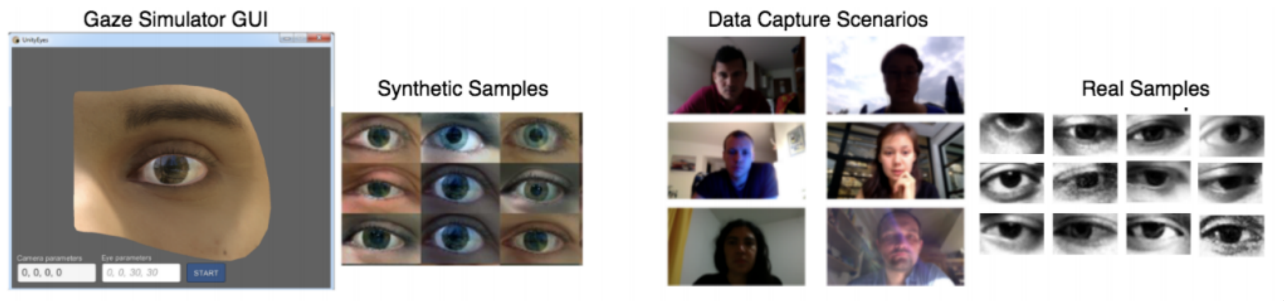


Figure 1.2: Synthetic and Real Images

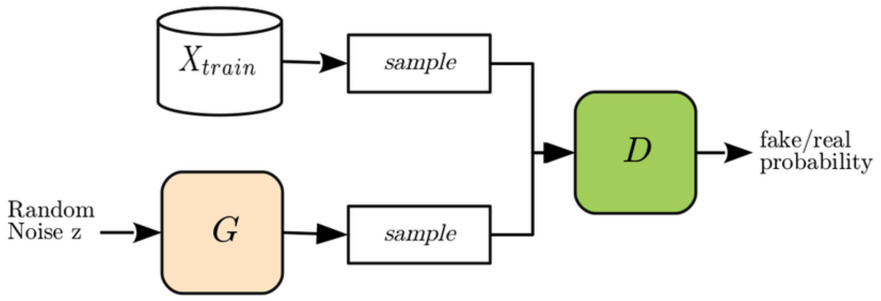
# Chapter 2

# GAN Variations

## 2.1 Generative Adversarial Networks

Generative Adversarial Networks or GANs are one of the most popular unsupervised Machine Learning algorithm. They are generative models which are used to learn the intrinsic underlying structure of the given data ***p(x)***, which allows us to generate synthetic data by sampling from the distribution whereas supervised approaches learn conditional probability distribution ***p(y—x)***. Generative Adversarial Networks are composed of two networks:

- First network is **Generator**, which is used to generate new fake data similar to the real data. It learns mapping from noise vector to the dataset distribution. Generator can be considered as art forger, which creates fake works of art.
- Second network is **Discriminator**, whose goal is to find whether an input data is real(belongs to original dataset) or fake(generated by Generator).



.

Figure 2.1: GAN

**Mathematical Modeling of GAN** We use neural networks as a function approximator for Generator and Discriminator. A neural network $\mathbf{G(z,\boldsymbol{\theta})}$ is used to model Generator Network. It's role is to map input noise variable $\mathbf{z}$ to the desired data space $\mathbf{x}$. Conversely, Discriminator is modelled as $\mathbf{D(x,\boldsymbol{\theta})}$ which outputs the probability that the data comes from dataset or is an output of Generator. In both cases, $\boldsymbol{\theta}$ represents the weights that define the neural network.

As a result, the Descriminator is trained to correctly classify data as *real* or *fake*. Weights are updated to maximise the probability that any *real* data is classified as *real*, and minimizes the probability for *fake* data to be classified as *real* Hence it maximises the function ***D(x)*** and minimizes ***D(G(x))***.

Whereas Generator is trained to fool the discriminator by generating realistic data as possible. It means, Generator's weights are updated such as it maximises probability of *fake* image to be classified as *real*. Formally, it means that it maximises ***D(G(x))***.

Since during training, both networks try to optimize the opposite loss function, it can be thought as two agents playing a minmax game with value function $\mathbf{V(G,\ D)}$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Figure 2.2: Value Function

## 2.2 GAN Stability

Training of a GAN is much more difficult than training of a discriminative network.

Training of a GAN has many problems namely-

- **Non convergence**-Either the generator or discriminator or both, oscillate and never converge.
- **Mode Collapse**-Discriminator is not able to force diversity into the samples generated by the generator, and it is as if the generator has collapsed to a single point.
- **Diminished Gradients**-The discriminator overpowers the generator as a consequence of which generator gradients diminish and it learns nothing.

- **Over-fitting**-Unbalance between the discriminator and generator which causes over-fitting.
- **Sensitivity to hyper-parameters**-The training of the GAN mode is highly sensitive to hyper-parameters like learning rate etc.

**Why mode collapse in GAN?**

Mode Collapse is one of the hardest problems in GAN training. Although a complete mode collapse is not that common but partial collapse happens often. In the case when the generator is able to generate a highly realistic image from the discriminator perspective, it is as if the most optimal image $x^*$ that fools discriminator is independent of z.

$$x^* = argmax_x D(x)$$

This way the mode collapses to a single point. The gradient of J wrt z become zero.

$$\frac{\partial J}{\partial z} \approx 0$$

Now since the generator is already very insensitive to z,it has high chances to get stuck in the next most vulnerable point. This leads to both generator and discriminator over-fitting to overcome the opponents weakness, and therefore the model will not converge.

## 2.3 More Stable GAN Variants

Vanilla GAN's are very unstable in training and especially when the *source* distribution p and *target* distribution q vary a lot, generator learns nothing and the problem of vanishing gradients take place.

To tackle this problem the concept of **Wasserstein GAN** [18] or WGAN came. **Wasserstein Distance** is a measure of the distance between two probability distributions. It is also called Earth Mover's distance, short for EM distance, because informally it can be interpreted as the minimum energy cost of moving and transforming a pile of dirt in the shape of one probability distribution to the shape of the other distribution. The cost is quantified by: the amount of dirt moved x the moving distance.

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

.

We can't exhaust all possible joint probability distribution of x and y, so the smart transformation of the above formula based on the Kantorovich-Rubinstein duality is:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$

.

This function demands K-Lipschitz continuous.

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

.

Suppose this function f comes from a family of K-Lipschitz continuous functions. In the modified Wasserstein-GAN, the "discriminator" model is used to learn w and the loss function is configured as measuring the Wasserstein distance between pr and pg.

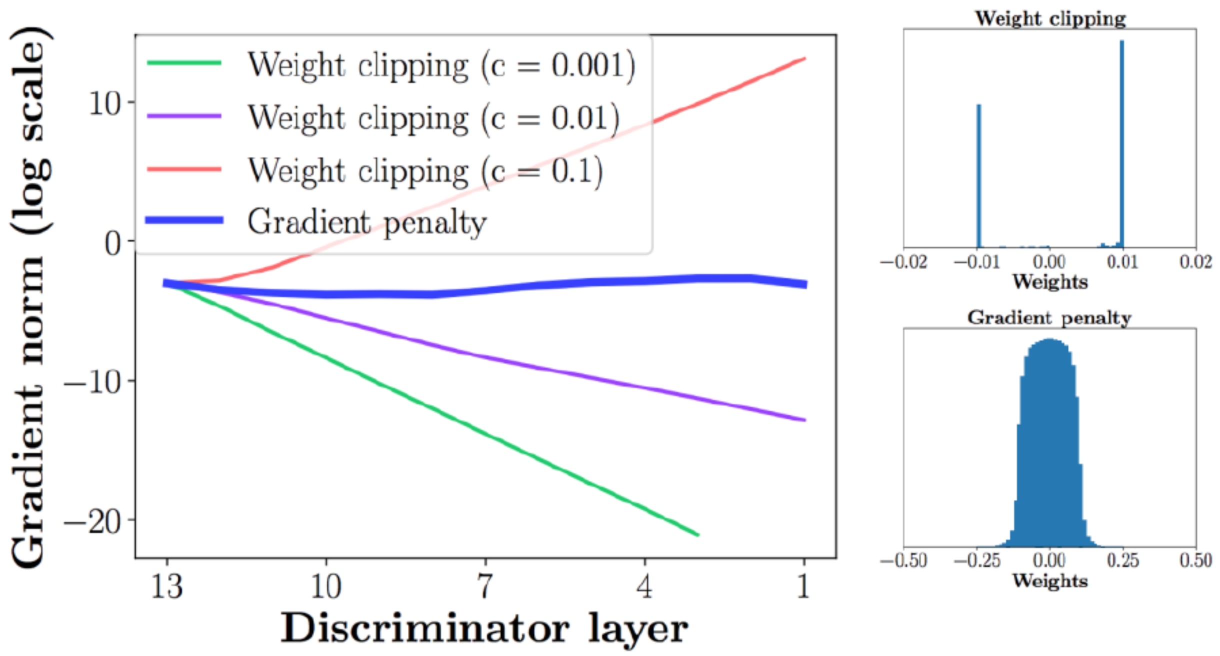
$$L(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x\sim p_r}[f_w(x)] - \mathbb{E}_{z\sim p_r(z)}[f_w(g_\theta(z))]$$

.

| | **Discriminator/Critic** | **Generator** |
|---|---|---|
| **GAN** | $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$ | $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$ |
| **WGAN** | $\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right)\right]$ | $\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$ |

.

Figure 2.3: WGAN cost function



$$w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$$
$$w \leftarrow \text{clip}(w, -c, c)$$

FIGURE 2.4: WGAN Gradient clipping

To ensure a linear variation of gradients WGAN enforces the Lipchitz constraint by gradient clipping which also acts as a weight regularizer. This results in a compact parameter space W and thus fw obtains its lower and upper bounds to preserve the Lipschitz continuity However, the training is very sensitive to the clipping parameters and as soon as the clipping parameter exceeds a particular value the gradients start to explode. Also the quality of image generated is worse than vanilla GAN still it provides a more stable training procedure with less chances of vanishing or exploding gradients if the clipping parameter is chosen wisely.

**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

.

Figure 2.5: WGAN Algorithm

However a better method of enforcing the Lipchitz constraint is Gradient Penalty. A penalty term is added to the critic or discriminator loss function whose norm is constrained to be one. This is termed as **WGAN-GP** [19]. This again gives image which are poorer in quality than vanilla GAN but the training is stable. Due to the stable training, a deeper generator and discriminator can be modeled which would give better image quality.

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}}\left[D(\tilde{\boldsymbol{x}})\right] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}}\left[D(\boldsymbol{x})\right]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

.

Figure 2.6: Gradient Penalty

## 2.3.1 Spectral Normalization

One of the main problems with Generative Adversarial networks is its instable training. So a normalization called spectral normalization [20] is used to normalize the weights to stabilize the training of the discriminator. This method is computationally very light. The spectral normalization normalizes the spectral norm of the weight matrix W of the discriminator so that it satisfies the Lipschitz constraint $\sigma(W) = 1$ :

$$\bar{W}_{SN}(W) := W/\sigma(W)$$

If we normalize each $W^l$ using the above equation, then $\sigma(\bar{W}_{SN}(W)) = 1$ and $||f||_{Lip}$ is bounded from above by one. This method is different from other spectral normalization methods as in this method the cost function is augmented with a sample data dependent regularization function in contrast to other methods which use sample data independent regularization function.

# Chapter 3

# Domain Adaptation

Domain Adaptation is a field of Computer Vision, where we try to learn mapping of labels from from *source* dataset and try to mimic the same performance on *target* dataset. It has a lot of application where we have very limited or no availability of labels. The main idea of Domain Adaptation is based on feature space. We use neural network to convert data into a feature space and finally classify the data. If we try to make the feature space of *target* domain similar to the *source* domain, then we get decent accuracy on *target* domain as well.

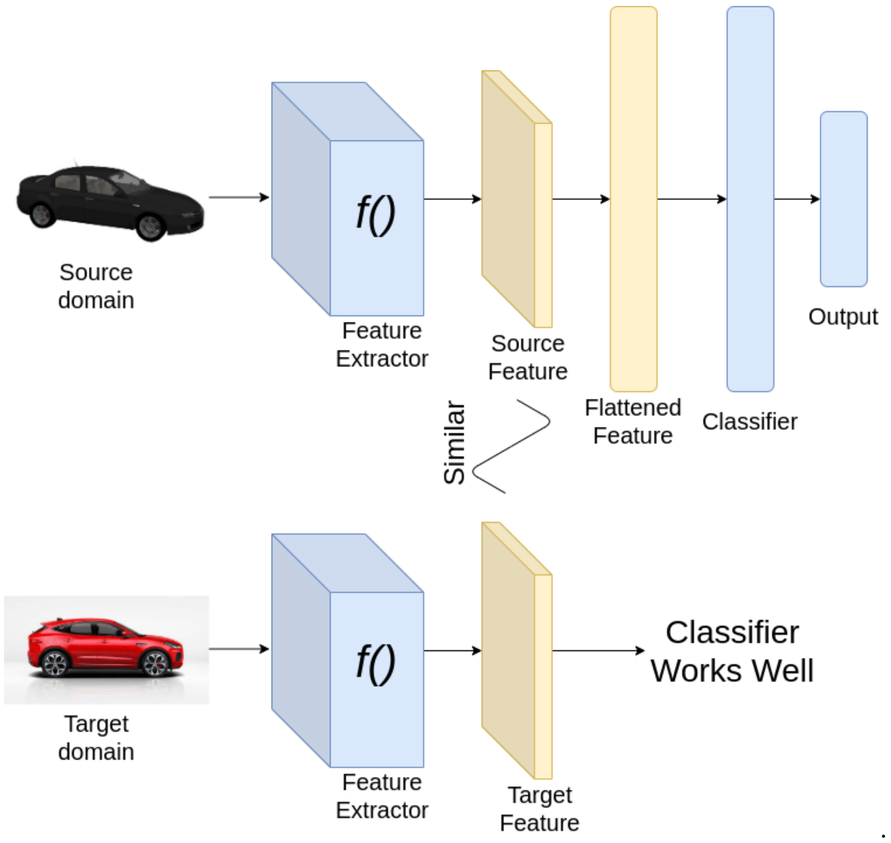


FIGURE 3.1: Domain Adaptation

There are three types of Domain Adaptation based on availability of data from *target* domain.

- **Supervised** We have labelled data available on *target* domain but the amount of data is much smaller as compared to *source* domain
- **Semi-supervised** Our *target* domain is partially labelled.
- **Unsupervised** Our *target* domain is completely unlabelled.

## 3.1 Techniques for Domain Adaptation

There are three type of techniques used for Domain Adaptation.

- Divergence based Domain Adaptation.
- Adversarial based Domain Adaptation.
- Reconstruction based Domain Adaptation.

### 3.1.1 Divergence based Domain Adpatation

Divergence based methods works on the principle of minimizing some divergence criterion between *source* and *target* distribution, hence leading to **domain invariant features**. Commonly used divergence based criterion are Contrastive Domain Discrepancy, Correlation Alignment, Maximum Mean Discrepancy, Wasserstein etc. For example in Maximum Mean Discrepancy(MMD), we find whether the given two samples belong to the same distribution or not. We say that two distribution are similar if their moments are similar. We apply a kernel to find first, second and third moments, and in *latent* space, we compute the difference between moments and average it.

$$\mathrm{MMD}(P, Q) = \|\mathbb{E}_{X\sim P}[\varphi(X)] - \mathbb{E}_{Y\sim Q}[\varphi(Y)]\|_{H}.$$

FIGURE 3.2: MMD critetion

The MMD divergence between two distributions **P** and **Q** over set **X** is defined by a feature map $\phi : \mathbf{X} \rightarrow \mathbf{H}$ , where H is called a reproducing Hilbert Space.

In correlation Alignment, we try to align the second order statistical correlation between the *source* and *target* domain. We have two different losses, one is classification loss and second is divergence based loss. All the divergences criterion are usually non-parametric and handcrafted mathematical formula which are not very specific to our problem. Hence this non parametric approach is not very specific to our problem. However, if the divergence can be learned based on our dataset then it is expected to perform better.

### 3.1.2 Adversarial based Domain Adaptation

This type of Domain Adaptation is based on Generative Adversarial Networks. Overall idea is to make feature space domain invariant. So, here our generator is the feature extractor from both the domains, while Discriminator distinguished between *target* and *source* domain. Playing this two player game, generator tries to fool the discriminator and in turn is expected to extract similar features.

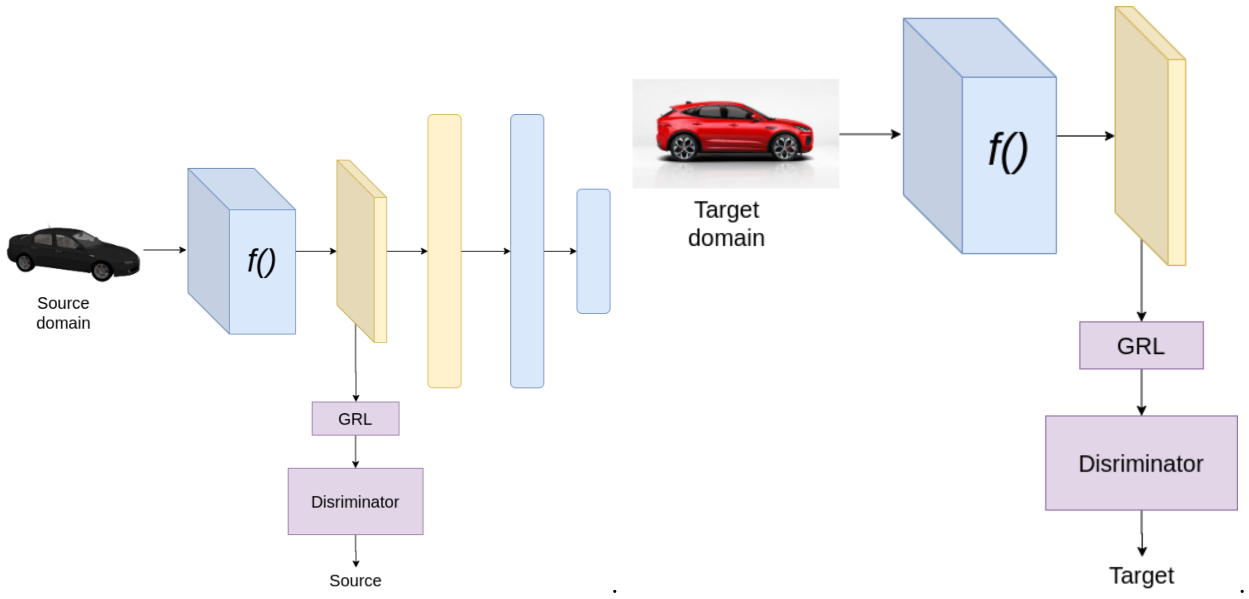


FIGURE 3.3: Adversarial based Domain Adaptation

### 3.1.3 Reconstruction based Domain Adaptation

This category belongs to the Image to Image translation. In the earlier two approaches, our objective was to make the feature space invariant to the *source*/*target* domain, but here we translate *target* data to look like the *source* data, so that classifier/regressor trained on soruce domain would perform good on the *target* domain as well. The simplest model for this type of

Domain Adaptation is Encoder-Decoder model along with a Discriminator which distinguish between decoder output and *source* data. Here Encoder-Decoder acts as a Generator which tries to generate data which looks similar to the *source* domain.

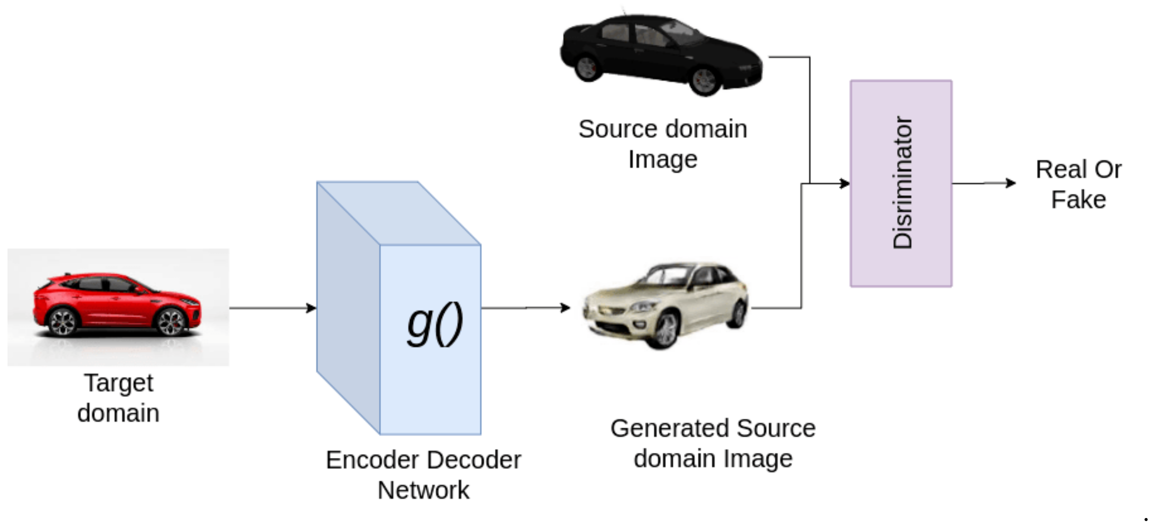


FIGURE 3.4: Image to Image translation

Another approach is to train a Cycle-GAN, where we use two neural network based encoder-decoder. One transform data from *source* to *target* domain, and other transform *target* to *source* domain. We simultaneously train both the networks and introduce a consistency loss to ensure consistency.

We have seen three different approach for domain adaptation. In our application of eye gaze estimation, the current state of art algorithm is based on approach involving Image-to Image translation. This type of methods gained a lot of popularity and are on the verge of saturation. We explored adversarial based Domain Adaptation and experimented many variations resolving the conflicts based on matching of *latent* space.

# Chapter 4

# Architectures Used

## 4.1 Adversarial Domain Adaptation

In this first architecture, we used simple adversarial adaptation to align the *latent* space of *source* and *target* domain. The regressor consists of 3 layered neural network and is trained on *latent* space of *source* domain. After alignment of *latent* spaces, the distribution of both *latent* spaces becomes almost similar and the regressor trained on *source* domain performs better as compared to the error before domain adaptation.

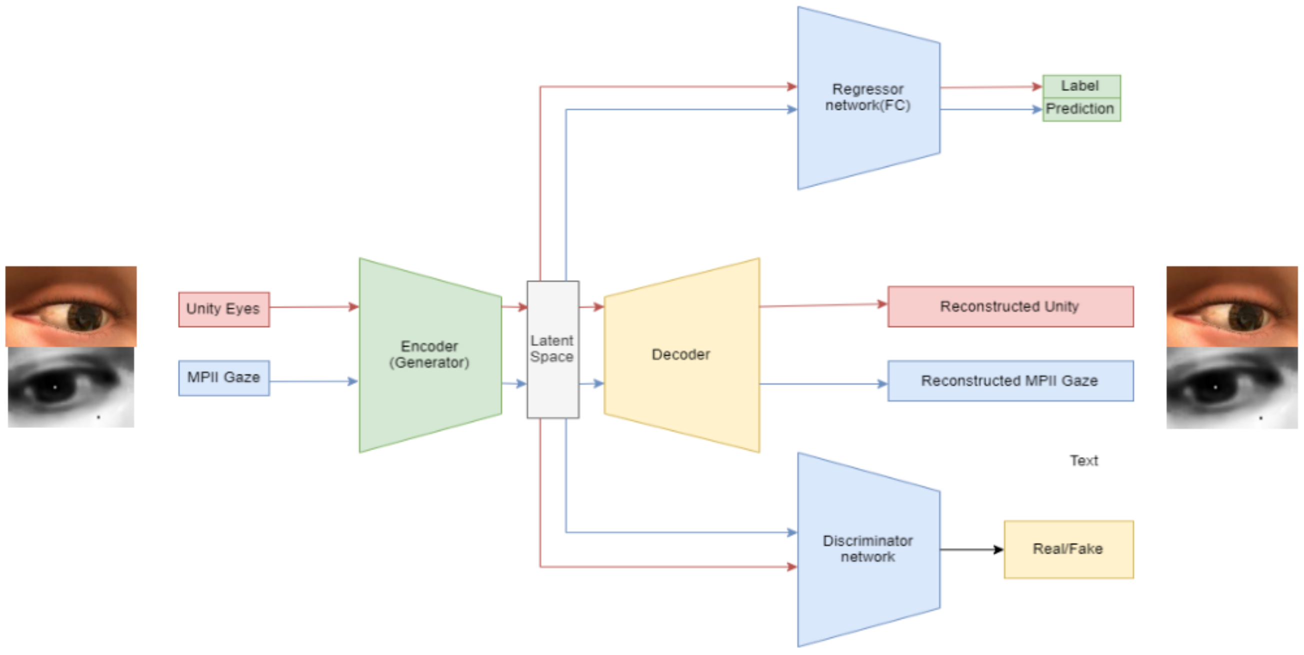


FIGURE 4.1: Adversarial Domain Adaptation Architecture

**Steps to train above architecture**

- First we train encoder-decoder network for *source* domain along with a regressor network parallel to the decoder for getting eye gaze angle.
- We use L2 norm for regressor angle error and reconstruction loss for decoder
- Then we freeze the layers of Regressor network.
- Afterwards, we train encoder-decoder network for *target* domain with L2 norm for reconstruction loss of decoder.
- Then we pre train Discriminator network to distinguish between *latent* spaces of *source* and *target* domain.
- Then we start GAN like training by loading the pretrained regressor and Discriminator weights. This makes *latent* space domain invariant.

## 4.2 Common Encoder Architecture

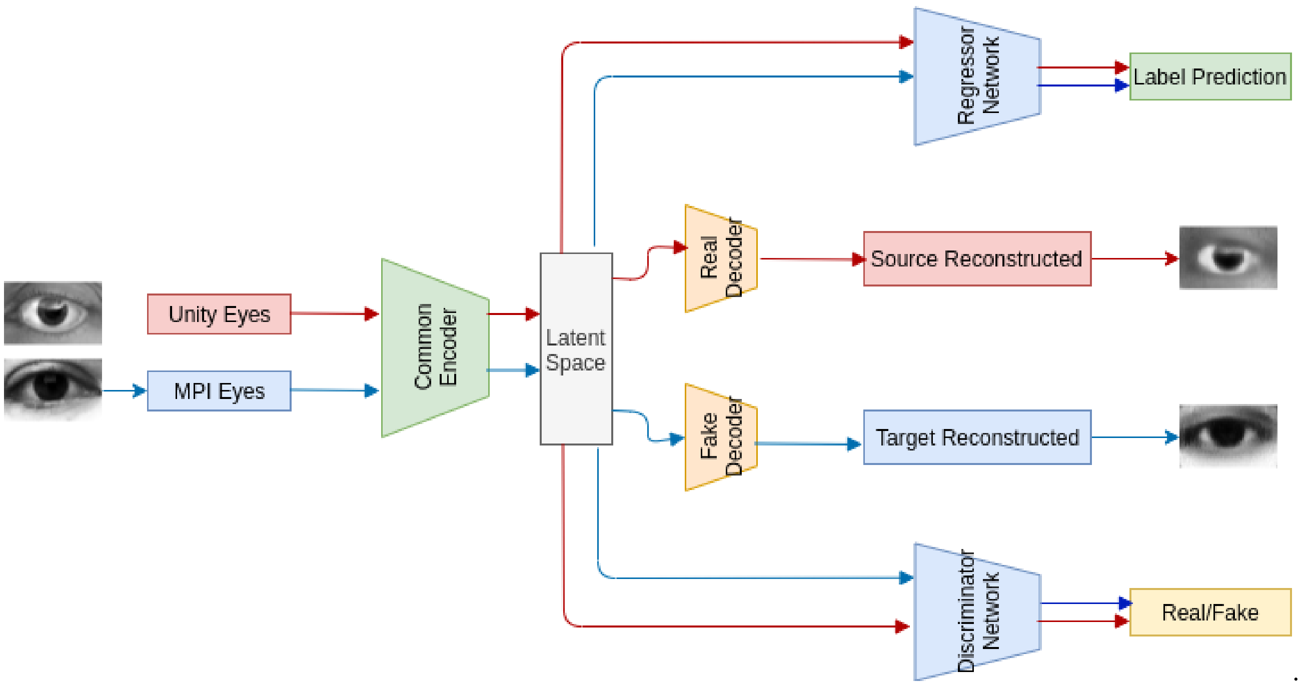


FIGURE 4.2: Common Encoder Architecture

This approach is almost similar to the previous one where we used two different encoders for *source* and *target* domains. While, in this approach, we use a single encoder for both the domains along with two separate decoder [21], Doing this, we force encoder to learn only the content part of the image and leave the texture and style for decoder to reconstruct. This makes *latent* space more similar to each other.

This method also forces both distributions to be in the same scale, hence it enables us to view the tsne visualization of *latent* space.Also since our purpose is to get both a good *latent* space distribution and a good reconstruction, the common encoder enables the network to learn common features for gaze prediction while the separate decoders help in reconstruction of each of the class of images.Common encoder helps since gradients for both the class of images pass and update the weights of the same encoder thereby learning common features.

**TSNE Visualizations** Common Encoder approach allows us to visualize *latent* spaces as they are in same scale for both domains.

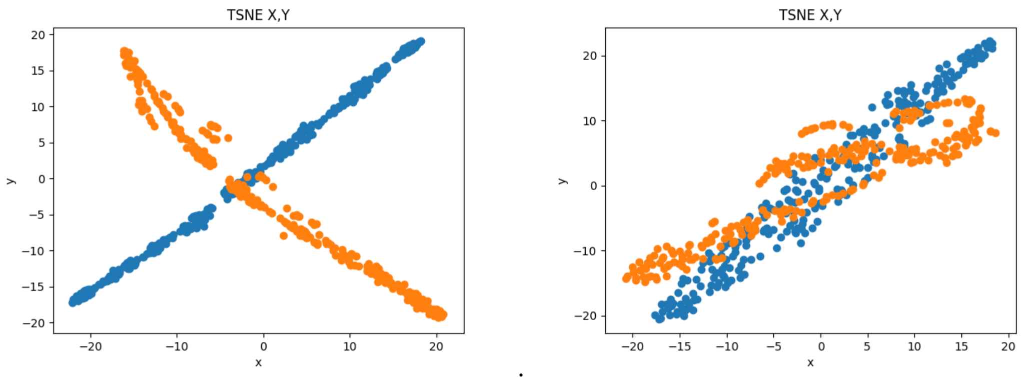


FIGURE 4.3: *latent* space Visualization

Here we can see that first image is *latent* space distribution while training only the AutoEncoder part. The common encoder forms two different clusters for both distributions. Now the task for GAN part would be to match these distributions. Whereas, the second image is the visualization after GAN like training for domain adaptation. Here we can see that encoder nearly matches both the distributions.

## 4.3 Style and Content Architecture

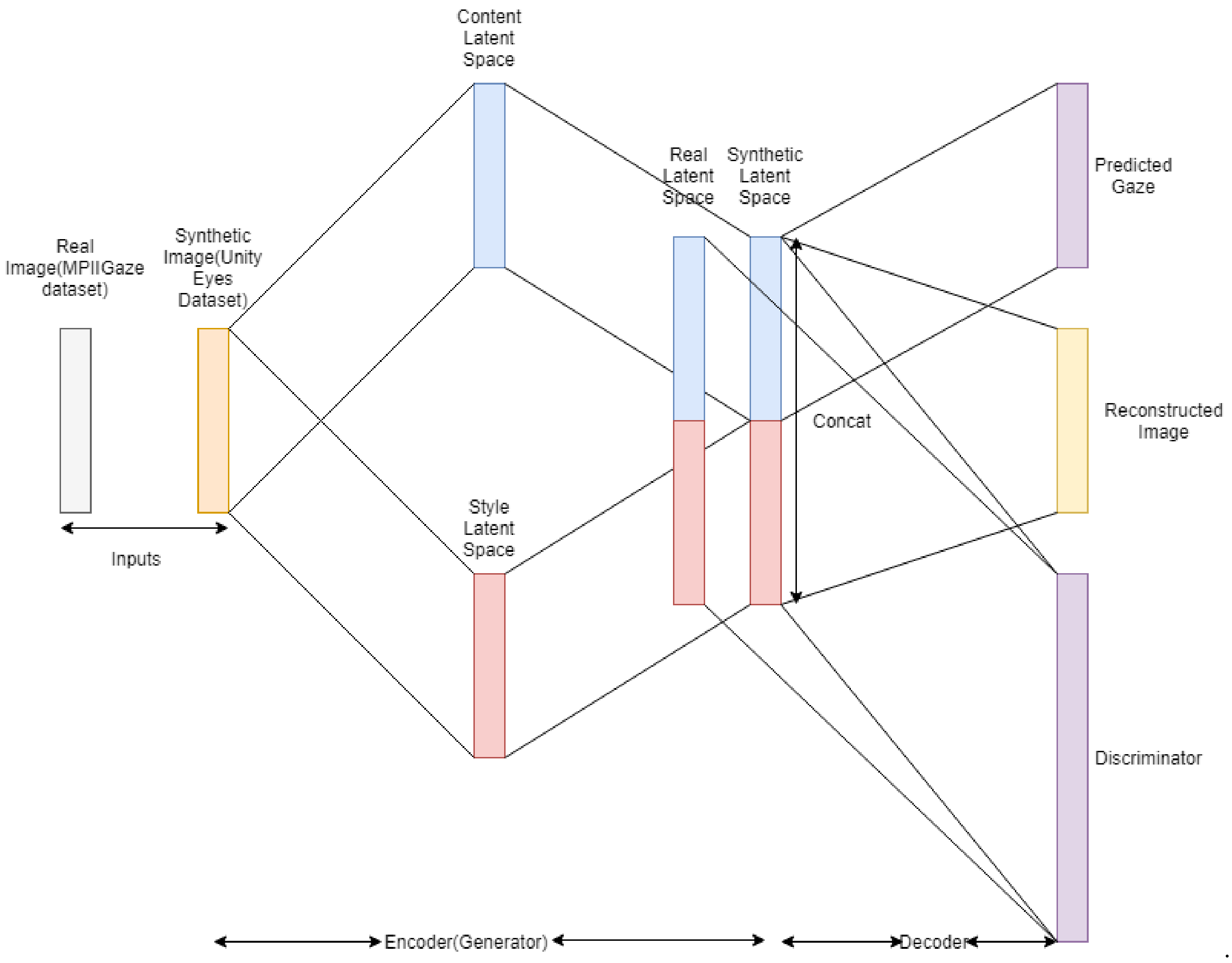


FIGURE 4.4: Style and Content Architecture

In this framework we assume that images from different domains of the same kind share similar attributes due to the common content but are different visually due to varying style contents [22].For example, one can tell by seeing digits in different fonts because of the common content which human brain perceives but these look visually very different to the eye due to different style contents.

In the traditional Encoder-Decoder architecture, due to common encoder the *latent* spaces may not be aligned properly due to the differing style part. In this architecture, images from both the synthetic and real dataset are passed through separate content and style encoders.The content part, which is assumed to be common between the synthetic and real datasets is used to predict the gaze for the real dataset(MPIIGaze dataset in our case) by passing through a fully connected neural network.For the reconstruction of the images from the *latent* spaces, the content *latent* space and the style *latent* space are concatenated, from which the reconstructed images are decoded.The discriminator tries to discriminate between the content *latent* space vectors of the real and the synthetic dataset.The encoder behaves as the generator.

Further as an experiment,it is seen that the *latent* space values can have varying ranges to match. So before moving on to the GAN training phase,the *latent* space is constrained to N(0,1) by adversarial training and sampling from unit normal distribution.Then the network is trained similarly.

## 4.4 Architectural Details

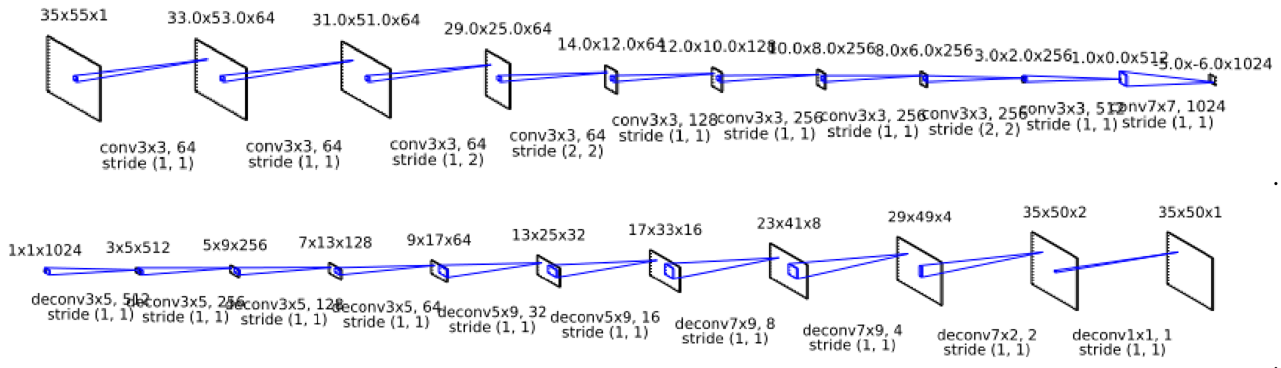

35x55x1
33.0x53.0x64
31.0x51.0x64
29.0x25.0x64
14.0x12.0x64
12.0x10.0x128
10.0x8.0x256
8.0x6.0x256
3.0x2.0x256
1.0x0.0x512
-5.0x-6.0x1024
conv3x3, 64 stride (1, 1)
conv3x3, 64 stride (1, 1)
conv3x3, 64 stride (1, 2)
conv3x3, 64 stride (2, 2)
conv3x3, 128 stride (1, 1)
conv3x3, 256 stride (1, 1)
conv3x3, 256 stride (1, 1)
conv3x3, 256 stride (2, 2)
conv3x3, 512 stride (1, 1)
conv7x7, 1024 stride (1, 1)
1x1x1024
3x5x512
5x9x256
7x13x128
9x17x64
13x25x32
17x33x16
23x41x8
29x49x4
35x50x2
35x50x1
deconv3x5, 512 stride (1, 1)
deconv3x5, 256 stride (1, 1)
deconv3x5, 128 stride (1, 1)
deconv3x5, 64 stride (1, 1)
deconv5x9, 32 stride (1, 1)
deconv5x9, 16 stride (1, 1)
deconv7x9, 8 stride (1, 1)
deconv7x9, 4 stride (1, 1)
deconv7x2, 2 stride (1, 1)
deconv1x1, 1 stride (1, 1)

FIGURE 4.5: AutoEncoder Architecture

Figure 4.5 shows the architecture for AutoEncoder network used. Our input image is of rectangular dimension 35x55. So we apply different stride and padding values across both dimensions. Also Encoder part converges to a *latent* space of 1024 length which is used by regressor network to predict 3 dimensions of gaze angle. Then we apply deconvolution layers to stretch this *latent* space into the original image dimension which we train with *L2norm* for reconstruction loss.

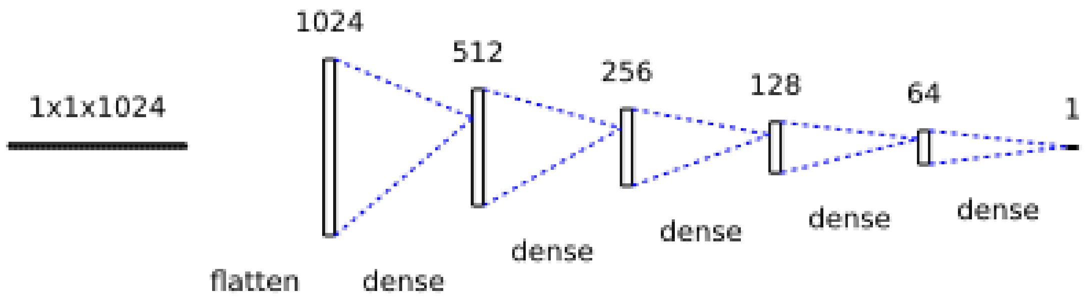


FIGURE 4.6: Discriminator Architecture

# Chapter 5

# Training

## 5.1 Preprocessing

The data was preprocessed and augmented in the following ways:-

- The size of the eye patches in both unity and MPIIGaze are resized to **35x55**.
- The input images are normalized to [-1,1] by the operation :-

$$img = 2 * (img/255.0) - 1$$

- Unity Eyes simulator generates images with right eye only but images from both eyes are present in MPIIGaze dataset, therefore left eye images are generates by randomly flipping the right eye images and their corresponding labels.

## 5.2 Loss Functions

Various kinds of losses are involved in the optimization equation for the various architectures stated in the previous section.

### 5.2.1 Centroid Loss

This loss is also named as *Semantic Loss*[23]. This is simply used in domain adaptation for classification tasks where we try to minimize the mean of *latent* vector for each class between *source* and *target* domain. This helps in domain alignment. But since we have a regression task, we divide continuous gaze angle into several bins and treat them as different classes.

**Intuition** We align the *source* and *target* distribution using adversarial Domain Adaptation. We use a Discriminator to make both distribution similar. There is an interent problem in this approach. The overall distribution might look same to the discriminator but it is quite possible that different class might have mapped to different parts of the *latent* space.

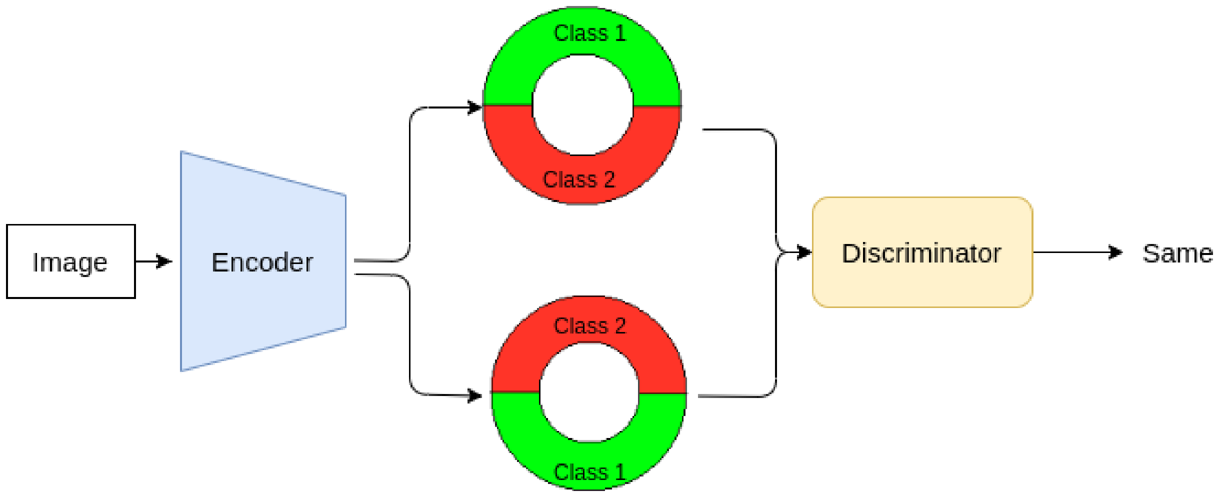


FIGURE 5.1: Overall distribution looks same to Discriminator.

Here we can see that class 1 is mapped to upper part and class 2 is mapped to lower part of the circle. Whereas it is exactly opposite in second distribution. Still overall distribution looks same and Discriminator cast distinguish between them. Generator only tries to fool the Discriminator, and makes overall distribution of *source* and *target* similar. So it is quite possible that different classes of eyes in *source* and *target* domain may lie on different part of *latent* space.

**Mathematical Modelling** This domain adaptation approach with semantic loss will make the two distribution similar but *latent* space of *target* domain may fail to perform on the regressor trained on *latent* space of *source* domain.

$$\mathcal{L} = \underbrace{\mathbb{E}_{(x,y)\sim D_S}[J(f(x),y)]}_{\mathcal{L}_C(\mathcal{X}_S,\mathcal{Y}_S)} + \lambda \underbrace{d(\mathcal{X}_S,\mathcal{X}_T)}_{\mathcal{L}_{DC}(\mathcal{X}_S,\mathcal{X}_T)}\ .$$

This loss function is *source* regression error and discrepancy between *source* and *target* domain. Here first term is the gaze angle error for synthetic UNITY images. $\lambda$ is the balance parameter $d(.,.)$ represents the divergence or adversarial loss between two domains. For aligning the domains, we use semantic transfer. For supervised domain adaptation, we can align the embeddings semantically by adding the following objective,

$$\mathcal{L}_{SM}^{SDA}(\mathcal{X}_S, \mathcal{X}_T, \mathcal{Y}_S, \mathcal{Y}_T) = \sum_{k=1}^{K} d(\mathcal{X}_S^k, \mathcal{X}_T^k) .$$

where $K$ is the number of classes. It means, we can match the distributions directly in supervised Domain Adaptation. But our objective is to train unsupervised way, we do not have label information from *target* domain. We find a workaround, since we don't have class labels, we try to create pseudo labels and make use of them. We first assign pseudo labels to data from *target* domain by training a classifier $f$ and we obtained a pseudo labelled *target* domain. Detailed approach used to improve this classification is discussed in the next Chapter.
To circumvent the harm caused by false pseudo labels, we use **centroid alignment**. While computing centroid for each class, the influence created by false labels is expected to have neutralized by correct pseudo labels. We use following semantic transfer function for unsupervised domain adaptation.

$$\mathcal{L}_{SM}^{UDA}(\mathcal{X}_S, \mathcal{Y}_S, \mathcal{X}_T) = \underbrace{\sum_{k=1}^{K} \Phi(C_S^k, C_T^k),}_{\mathcal{L}_{SM}(\mathcal{X}_S, \mathcal{Y}_S, \mathcal{X}_T)} .$$

where $C_S^k$ and $C_T^k$ are centroid for each class in *latent* space. $\phi(.,.)$ is an appropriate function which measure distance. In our experiments, we use squared Euclidean distance

$$\phi(x, x') = ||x - x'||^2)$$

We obtain K centroids for each domain. Through this explicit forcing the distance between same class to be smaller, we ensure that same class will be mapped nearby in tha *latent* space. Hence the regressor trained on *source* domain will perform good on *target* domain.

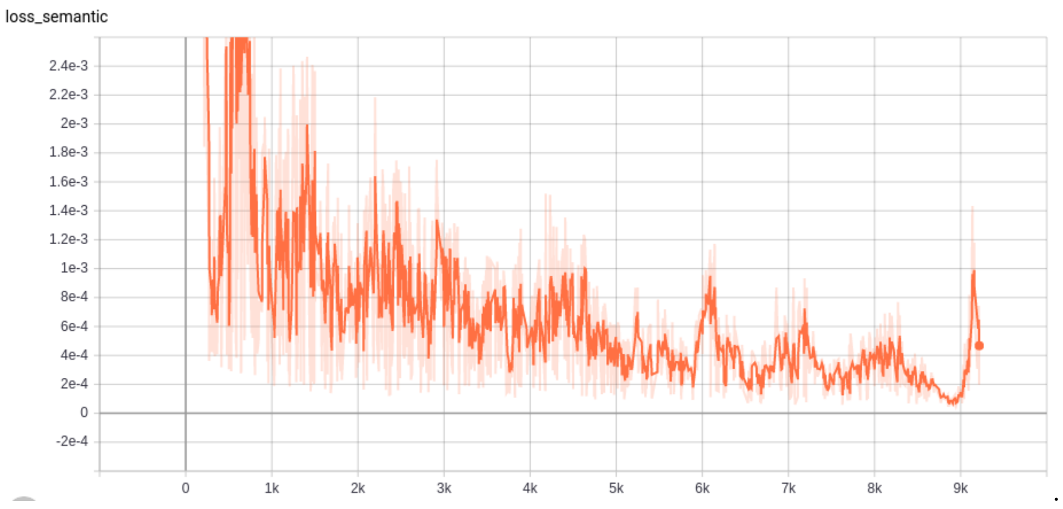


FIGURE 5.2: Semantic(centroid) Loss curve.

**Moving Semantic Transfer Network** The proposed objective function for centroid alignment suffers some problems, such as it is possible that some classes are missing in current mini batch of *target* data as batches are randomly selected. For smaller batch size, some false pseudo labels might create huge difference between true centroid and pseudo labelled centroid. Hence instead of aligning centroids in each batch separately, we maintain global centroids for each class. In each iteration, *source* centroids are updated by labelled data while *target* centroids are updated by pseudo labelled *target* samples.

$$C_S^k \longleftarrow \theta C_S^k + (1-\theta)C_S^k$$

$$C_T^k \longleftarrow \theta C_T^k + (1-\theta)C_T^k$$

This type of moving average works in a intuitive way. If a class is missing in current *source* batch, we can align the *target* centroid with the global centroid of that class updated in the last iteration. Thus the centroids change by a limited amount in each step. This also deals with the problem of some false pseudo labels. If there is a false pseudo label, moving average centroids can prevent the misalignment as it maintains global centroids from previous batches.

### 5.2.2 Adversarial Loss

The increasing popularity of Generative Adversarial Networks inspires us to use GANs measuring difference between two distributions. The main idea of GAN is to improvize both the generator and discriminator simultaneously through a two player min-max game.

#### 5.2.2.1 Generator Loss

Generator creates new data and tries to fine tune them in such a way that it looks realistic. It's goal is to fool the discriminator. Equation for loss function is shown below

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)}))) .$$

Loss function only contains how stongly it is able to fool the discriminator. Training loss curve is shown below.

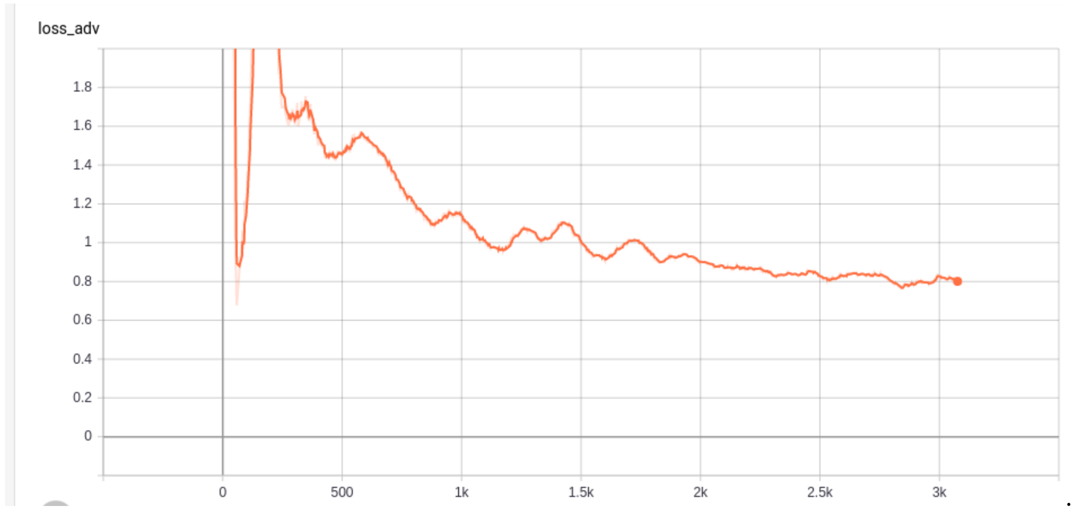


FIGURE 5.3: Training plot for generator.

#### 5.2.2.2 Discriminator Loss

Discriminator learns to distinguish between generator's output and real data. Loss function is shown below

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right] .$$

This loss function consists of two parts, first part force Discriminator to remember real data while second part asks it to reject generator's output. Training loss curve is shown below.

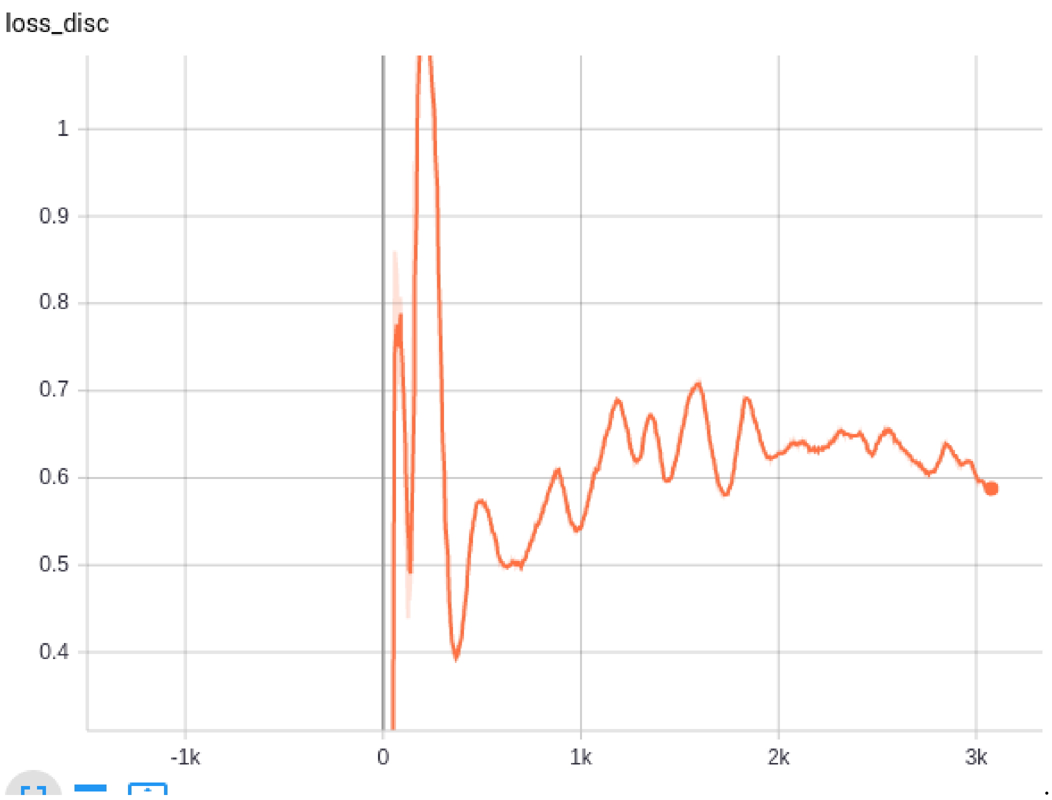


FIGURE 5.4: Training plot for Discriminator.

#### 5.2.2.3 Normal Distribution (N(0,1)) loss

This loss is applied to map *latent* space of *source* and *target* domain to Normal Distribution. We believe that if both the domains are in same scale, it will be much easier for Generator to align the domain properly. Hence this loss is applied on *latent* space of content part of the Encoder. Style part is only used for reconstruction of the image, hence there is no need for normalizing it.

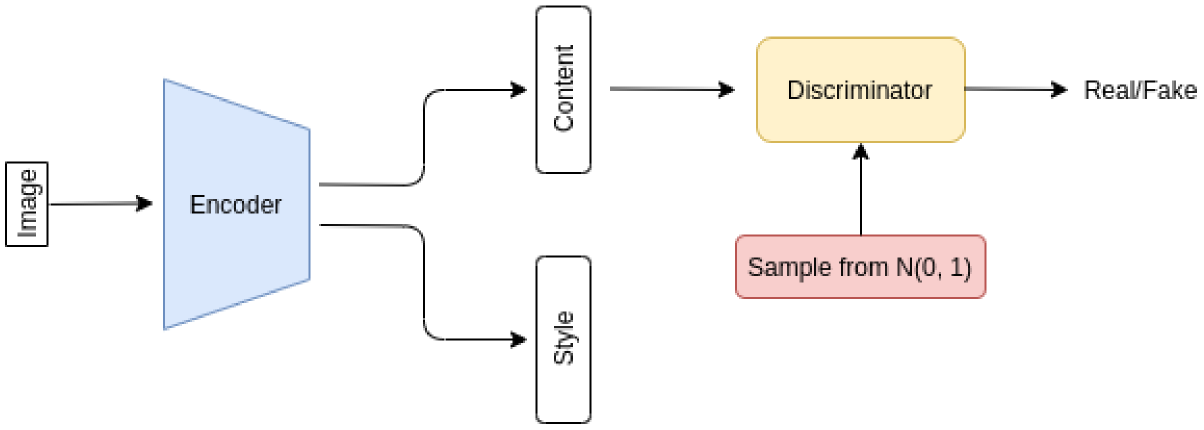


FIGURE 5.5: Mapping *latent* space to Normal Distribution.

This loss helps in getting both *latent* space in same scale and hence align them properly.

### 5.2.3 Reconstruction loss

Reconstruction loss is the L1-loss between the image tensor and the output tensor generated after passing the input image tensor through an autoencoder.

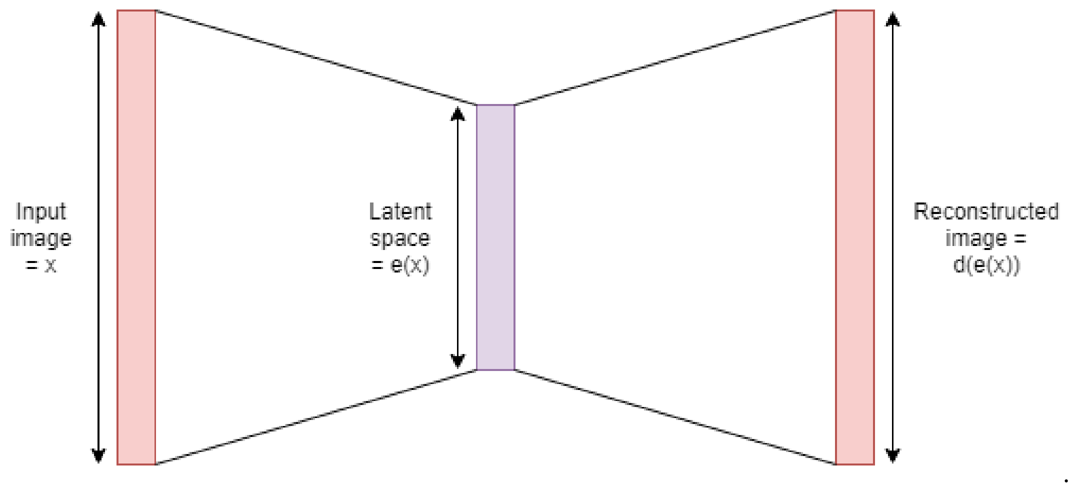


Figure 5.6: Reconstruction of image in an autoencoder

$$Reconstruction\ loss = \sum ||x - d(e(x))||$$

The backpropagation gradients flow from the decoder to the encoder. We observed that

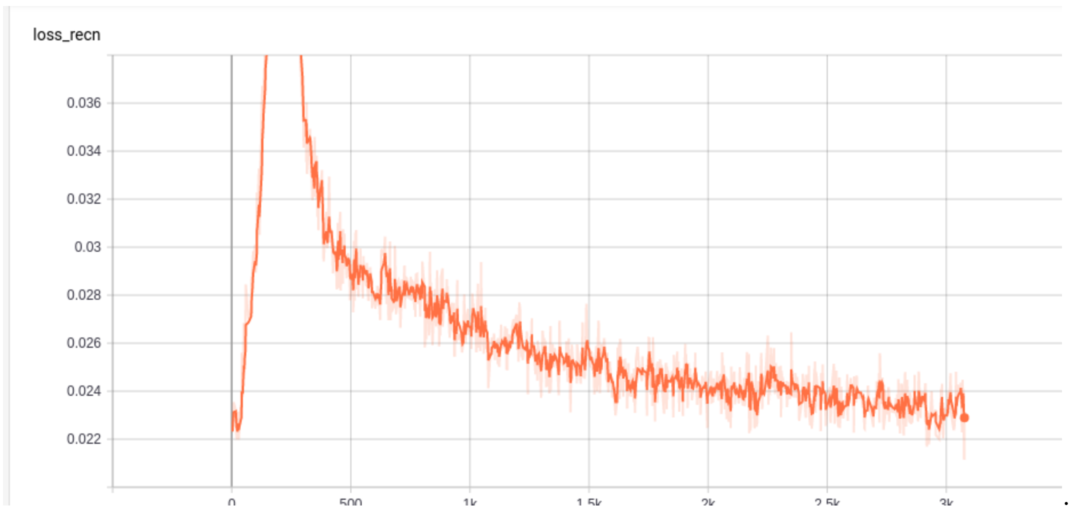
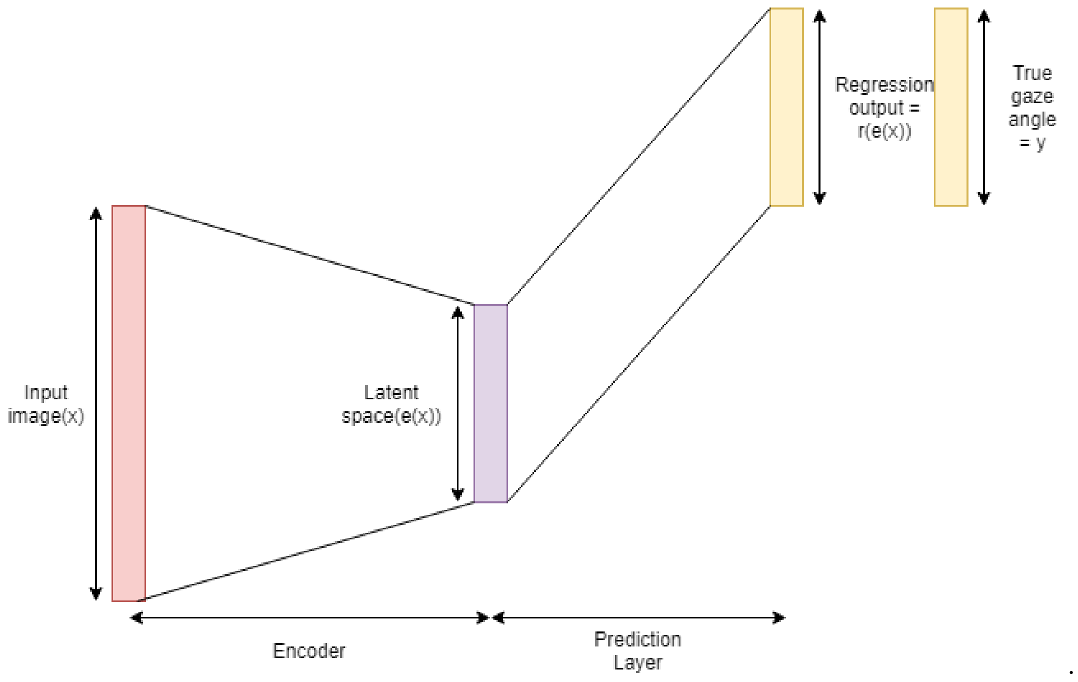


Figure 5.7: Training plot for Reconstruction loss.

### 5.2.4 Regression loss

While training the network with only unity images, the gaze angles of the images are available. So the encoder weights and the weights of the prediction layer from the *latent* space to the regression output are learned due to backpropagation of gradients available due to regression loss which is the L2 loss between true gaze direction and predicted gaze direction.



$$Regression\ loss = \sum ||y - r(e(x))||^2$$

## 5.3 Updating Discriminator with History

There are some more problems associated with adversarial training. Discriminator network gets updated by only focusing on latest batch of *latent* space. This lack of previous memory creates some problem:

- It may cause divergence of adversarial training. As discriminator sees current batch, after some iterations, it may forget completely about the previous batches.
- The generator network may introduce some artifacts that the discriminator has forgotten about.

During the entire training process, any *latent* space corresponding to the generator are fake to the discriminator. Hence discriminator should be able to classify all of them as fake. As pointed out in [24] we use a method to improve the stability of adversarial training. We update discriminator by using a history of previous *latent* spaces as compared to only few in the current batch. An overview of this approach is in Figure.

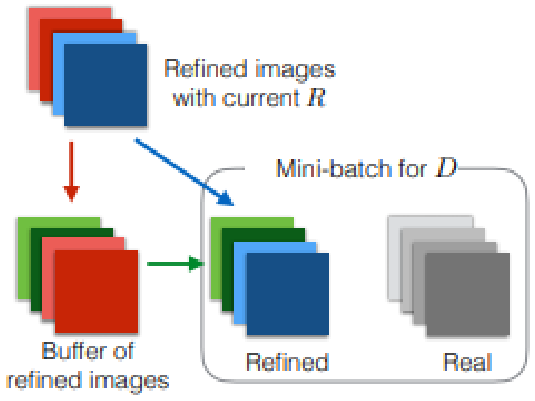


FIGURE 5.8: Illustration of using Discriminator with history.

Let $B$ be the Buffer size and $b$ be the size of mini-batch. At each iteration of the Discriminator training, we evaluate the loss function of discriminator by sampling exactly $b/2$ images from the current mini batch and rest $b/2$ images from the buffer to update Discriminator weights. Buffer size $B$ is kept fixed. In each iteration, we randomly selece $b/2$ images from current mini batch and replace them randomly with images in Buffer. In contrast to this approach, *Salimans et al.* [2] used a running average of model parameters to stabilize the training. These two approach can be used together complementing each other.

# Chapter 6

# Classifier

## 6.1 Introduction

Our problem at hand is a regression problem in which we have to predict the gaze angles in a continuous distribution. But before delving into the regression problem, it must be checked whether the network gives good classification results because after training of the unity images in the first place we freeze the regressor weights and then fine tune it to get good results for the MPIIGaze dataset using domain adaptation among the two *latent* spaces and thus making the *latent* spaces more informed. But if the regressor network is not trained well for unity it is highly unlikely that it would give good results for the MPIIGaze dataset. So we divide the gaze angles into different bins and train a classifier on them with unity images. And for our approach to work well it must give good classification results for unity as well as for MPIIGaze dataset even before starting the GAN training phase. We assume that if the bin predicted is correct for MPIIGaze, after domain adaptation the predicted gaze angles will move closer in that bin.

## 6.2 Resolution for bin division

Before training the network it must be decided what should be the number of classes, the regression space be divided. Since we have eye gaze angles in all x,y and z direction, naively the space can be divided into 8 buckets since each direction is divided into positive and negative space at zero and there are 3 directions so $2^3 = 8$. Also these spaces can be further divided into 2 spaces each, giving 64 bins. The finer the resolution the less accuracy of classifier is expected as we move closer to the continuous space.The division of the images into these spaces is calculated offline and the division goes as follows:-

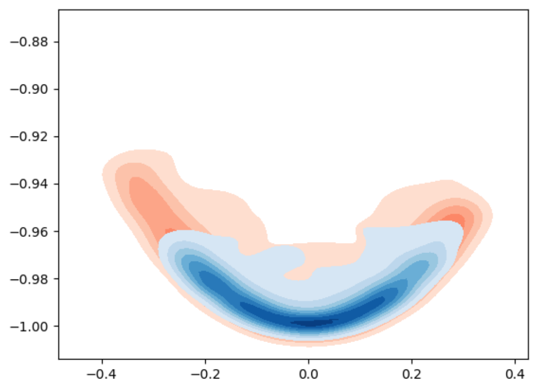
| Bin | X | Y | Z |
|---|---|---|---|
| 0 | - | - | - |
| 1 | - | - | + |
| 2 | - | + | - |
| 3 | - | + | + |
| 4 | + | - | - |
| 5 | + | - | + |
| 6 | + | + | - |
| 7 | + | + | + |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 55182 | 0 | 55600 | 0 | 54920 | 0 | 55473 | 0 |

It can be seen clearly that the distribution of images in the different buckets is not uniform. Since we use centroid loss in our domain adaptation optimization equation, we must know class information about our MPIIGaze images which we predict using our classifier. Since bin number 1,3,5 and 7 do not have any images in them, we discard them from our training. We notice that if we divide the gaze angles into 64 bins, only 27 of them have images from unity dataset and even lesser from MPII dataset, so it is not much useful.

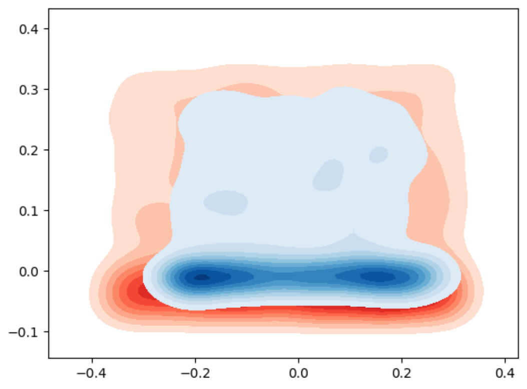
## 6.3 Comparison of range of gaze angles for Unity Eyes and MPIIGaze dataset

One of the assumption we make in domain adaptation is that the range of values of the two datasets should be same.Therefore, we analyze the range of gaze angles for the two datasets in all x-y,y-z and z-x direction. Here red colour depicts Unity Eyes dataset and blue colour depicts MPIIGaze dataset.



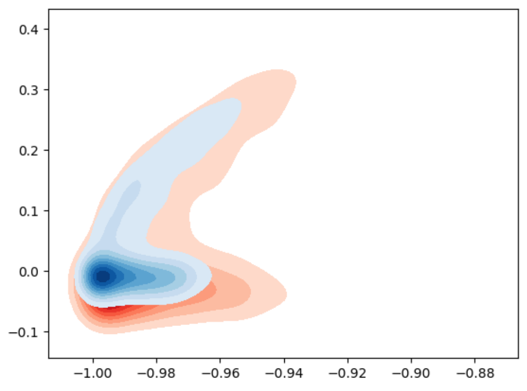
.

FIGURE 6.1: Distribution of images in x-y direction



.

FIGURE 6.2: Distribution of images in y-z direction



.

FIGURE 6.3: Distribution of images in z-x direction

So for validation of our assumption the Unity labels which lies in range of MPIIGaze images are taken into consideration. We found a minimum and a maximum value in all the three directions and removed all the images outside the range. But for this method to work in practice we need a good classifier to predict MPIIGaze labels since we are working in an unsupervised way.

## 6.4 Training

For the different architectures proposed in the previous chapters like separate encoder, common encoder, style and content and many more, first the classifier has to be trained to determine the accuracy of prediction on bins and to use the bins predicted for MPIIGaze dataset for training the second stage along with centroid loss which needs class information.

- **Simple neural network classifier** - Firstly we make a simple neural network with images as input as the bin to which the gaze belong as output.

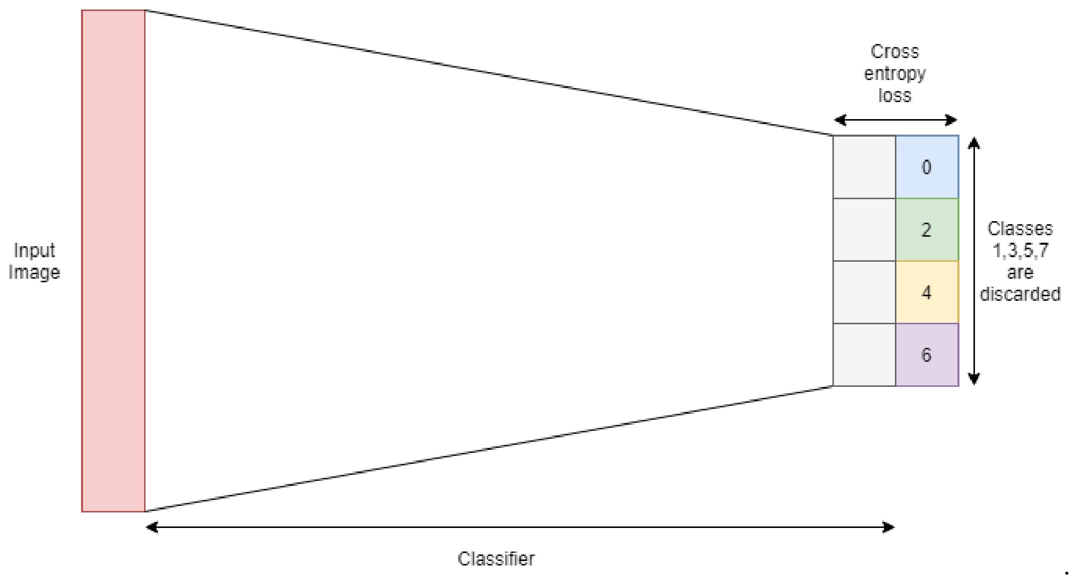


Figure 6.4: Simple Neural Network Classifier

- **Classifier trained with Domain Adaptation** - So only the encoder,decoder and classifier layer is trained for Unity Eyes images and then the accuracy of prediction for them is calculated. Then the same weights are used for the prediction of bins for MPIIGaze images and MPIIGaze labels are used to calculate accuracy. To increase the accuracy of the classifier for MPIIGaze, domain adaptation is started at this stage. Those weights are adapted to get a good classifier for MPIIGaze images.

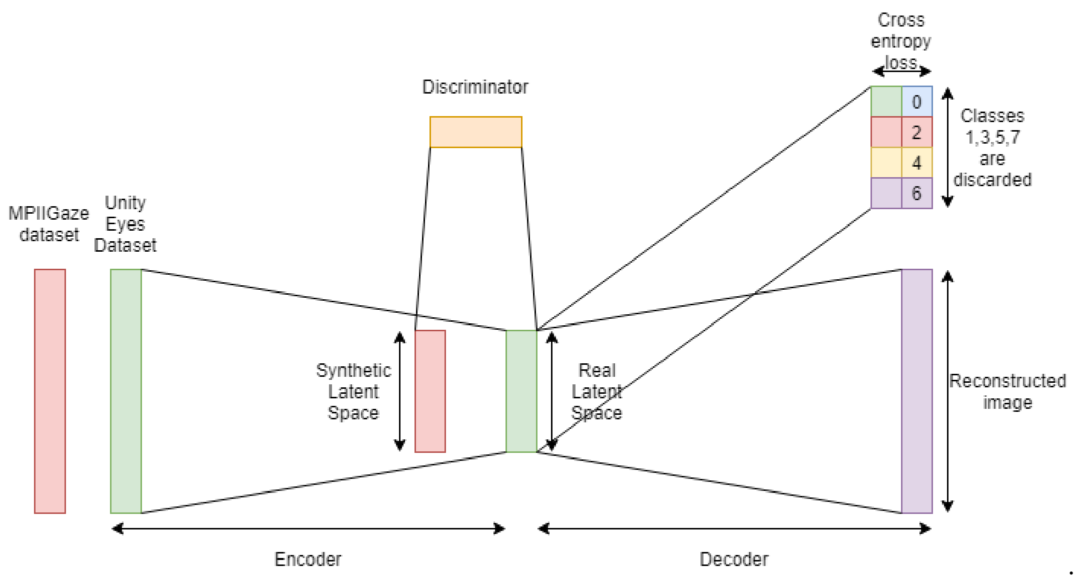


FIGURE 6.5: Classifier trained with domain adaptation

- **Classifier based on SimGAN[24]** - The distribution of Unity Eyes and MPIIGaze dataset vary a lot, so a classifier trained for Unity finds it difficult to adapt to MPIIGaze images due to difference in sharpness,contrast etc. So we train the SimGAN which converts Unity images to look like MPIIGaze images.Then we train a classifier on these MPII-lookalike Unity images and use that classifier to classify the MPII Images into bins.

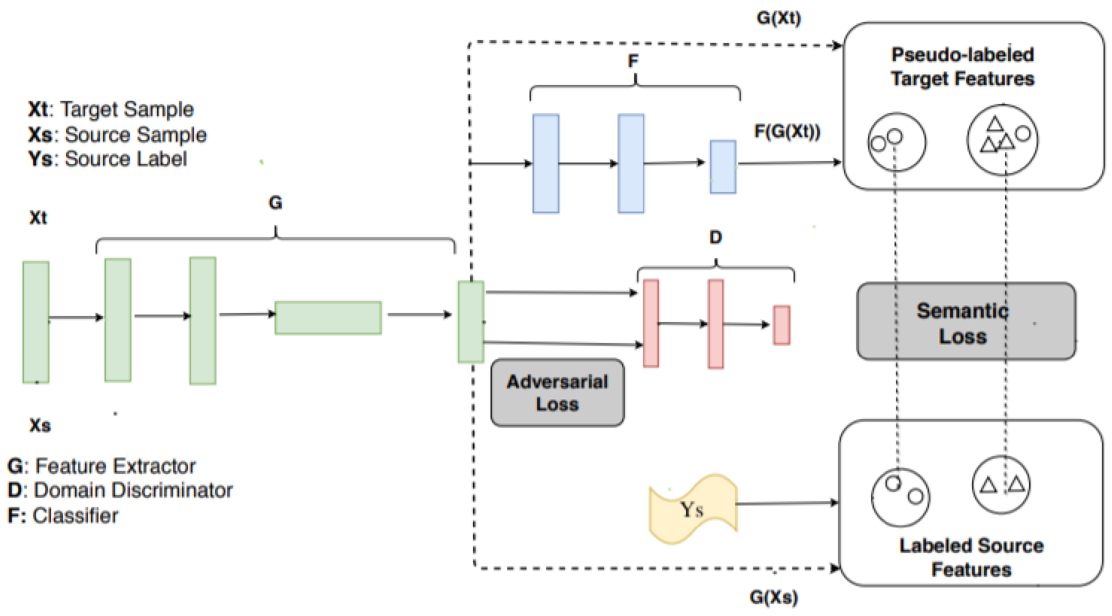


FIGURE 6.6: SimGAN

- **Classifier based on Perpetual Loss** - Firstly introduced as a concept in [25], perpetual loss has found use in style transfer [26] and many classification tasks. Due to the differences between MPIIGaze and Unity Eyes images, we intend to find the most structurally similar MPII image to a pool of Unity images and then give the unity image the label of that MPII image. For this we take a pre-trained VGG network with some fully connected layers removed, and use the features of its intermediate layers as the representation of similarity between the images.

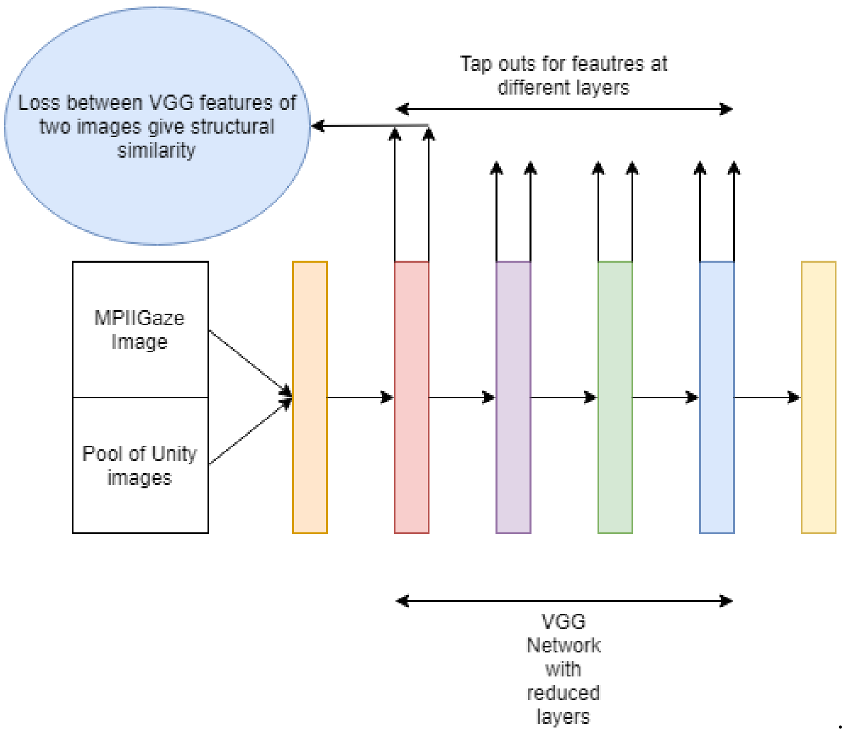

Loss between VGG features of two images give structural similarity
Tap outs for feautres at different layers
MPIIGaze Image
Pool of Unity images
VGG Network with reduced layers

FIGURE 6.7: Network for similarity prediction

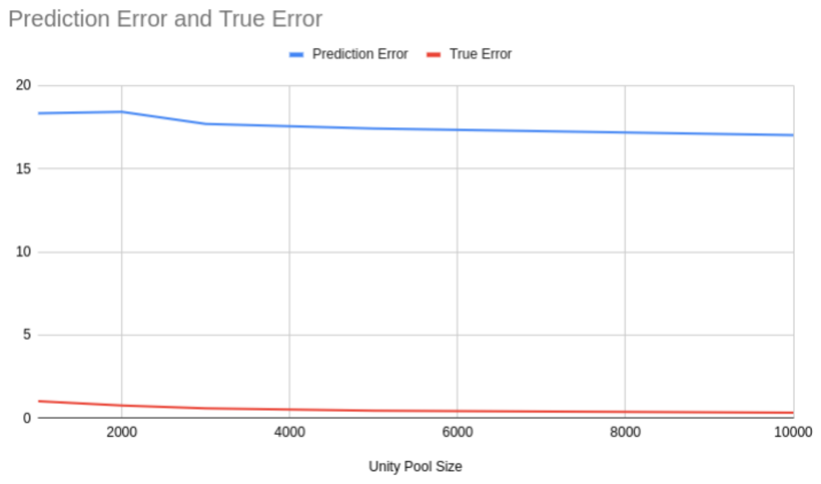


FIGURE 6.8: Performance of the methodology

| Unity Pool Size | Prediction Error | True Error |
|---|---|---|
| 1000 | 18.347 | 1.037 |
| 2000 | 18.43 | 0.77 |
| 3000 | 17.7 | 0.61 |
| 5000 | 17.43 | 0.47 |
| 10000 | 17.03 | 0.34 |

These results were not very promising so we did not train a classifier using this approach as it would not have given good results.

# Chapter 7

# Experiments and Results

## 7.1 Introduction

All the experiments that were performed were based on different architectures, additional losses but the training was performed following a common paradigm as follows:-

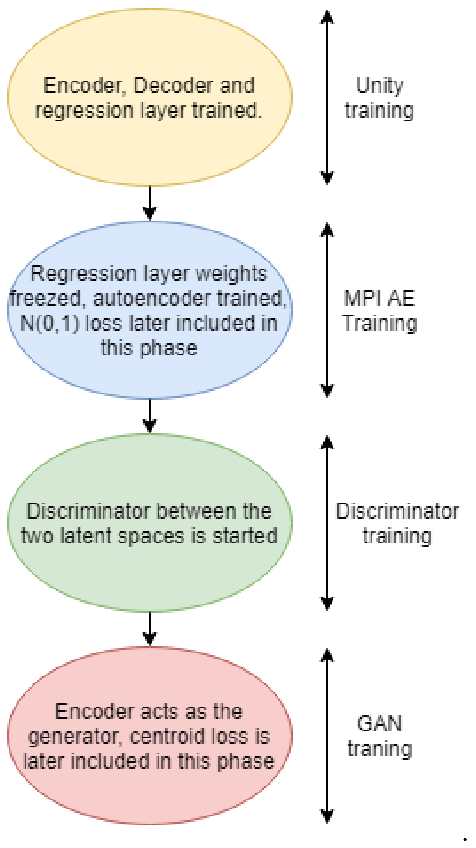

Encoder, Decoder and regression layer trained.
Unity training
Regression layer weights freezed, autoencoder trained, N(0,1) loss later included in this phase
MPI AE Training
Discriminator between the two latent spaces is started
Discriminator training
Encoder acts as the generator, centroid loss is later included in this phase
GAN traning

Figure 7.1: General training paradigm

## 7.2 Experiments

We will describe all the experiments we have performed.

### 7.2.1 Training *source* Domain (UNITY Images)

We trained AutoEncoder with labelled synthetic unity following usual supervised training methods. We used Adam optimizer[] to for mini batch gradient descent to optimize the parameters. Batch size was 512 and learning rate was kept at 0.001 along with a scheduler which decreases learning rate by 15% every 3 epochs.

| **Model** | **Angle Error(°)** |
|---|---|
| Simple AutoEncoder | 1.9 |
| Common Encoder | 2.1 |
| Style and Content | 2.0 |
| Style and Content with N(0, 1) loss | 3.8 |

Table 7.1: Angle Error for labelled *source* domain

We trained *source* domain with nearly 1 million unity images, separated a validation set consisting of 20 thousand images. We secured a decent angle error of 1.9 degrees. We now freeze these weights and try to bring target distribution closer to this.

### 7.2.2 target Domain performance before Adaptation

We load the weights of *source* domain and check the performance of target domain. This acts as a baseline to check if adaptation is working at all or not. We also finetune the *source* weights with reconstruction loss for *target* domain, freezing the gaze estimator layers.

| **Model** | **Angle Error(°)** |
|---|---|
| Simple AutoEncoder | 14.5 |
| Common Encoder | 12.3 |
| Style and Content | 12.1 |

Table 7.2: Before Adaptation performance of *target* domain

### 7.2.3 Perfomance after Adaptation

After finetuing of Decoder layers with reconstruction loss of *target* domain, we pretrain Discriminator for few epochs keeping AutoEncoder layers fixed. Then we load AutoEncoder and Discriminator weights and start adversarial training still keeping the layers fixed for gaze regressor.

| Model | WGAN | Gradient Penalty | Centroid Loss | N(0, 1) Loss, | Error(°) |
|---|---|---|---|---|---|
| Simple AutoEncoder | ✗ | ✗ | ✗ | ✗ | 11.3 |
| Simple AutoEncoder | ✓ | ✗ | ✗ | ✗ | 11 |
| Simple AutoEncoder | ✓ | ✓ | ✗ | ✗ | 10.8 |
| Common Encoder | ✗ | ✗ | ✗ | ✗ | 9.3 |
| Common Encoder | ✓ | ✗ | ✗ | ✗ | 9.8 |
| Common Encoder | ✓ | ✓ | ✗ | ✗ | 10 |
| Style and Content | ✗ | ✗ | ✓ | ✓ | **7.85** |
| Style and Content | ✗ | ✗ | ✓ | ✗ | 8.2 |
| Style and Content | ✗ | ✗ | ✗ | ✓ | 8.3 |
| Style and Content | ✗ | ✗ | ✗ | ✗ | 8.8 |
| Style and Content | ✓ | ✗ | ✓ | ✓ | 7.9 |
| Style and Content | ✓ | ✗ | ✓ | ✗ | 8.05 |
| Style and Content | ✓ | ✗ | ✗ | ✗ | 8.65 |
| Style and Content | ✓ | ✓ | ✓ | ✓ | 7.95 |
| Style and Content | ✓ | ✓ | ✓ | ✗ | 8.1 |
| Style and Content | ✓ | ✓ | ✗ | ✗ | 8.6 |

TABLE 7.3: Experiment Specifications

We performed many experiments with a combination comprising all three different architecture viz Autoencoder, Common Encoder, Style and Content, along with the different type of Generative Adversarial Networks. Error rates have been reported in Table 7.3 with an ablation study of different loss functions used in out approach.

## 7.3 Results

| Training Genre | Method | Angle Error(°) |
|---|---|---|
| Mannually Annotated Real Samples | Schneider *et al.*[27] | 16.5 |
| | Lu *et al.*[28] | 16.4 |
| | Sugano *et al.*[29] | 15.4 |
| | Zhang *et al.*[30] | 13.9 |
| Auto Annotated Synthetic Samples | Wood *et al.*[31] | 9.9 |
| | SimGAN[16] (Before Adaptation) | 11.2 |
| | SimGAN (After Adaptation) | 7.8 |
| | **Ours** (Before Adaptation) | **12.1** |
| | **Ours**(After Adaptation) | **7.85** |

Table 7.4: Comparison of Error in State-of-the-art algorithms.

**Comparision with State-of-the-art** In table 7.4, we compared performance of our approach with the recent state of the art methods on MPIIGaze test set. Results consists of two parts, first where those methods trained on mannually annotated gaze datasets. Our method does not involve any human intervention, it significantly beats fully supervised methods.
In second part, we compared our results with papers which use labels which are generated by some rendering engines like UnityEyes which was released by Wood *et al.* [31]. They achievec 9.9°error which is an improvement of 4°as compared to approaches involving supervised training. Current state-of-the-art is SimGan[16] with its adversarial pixel domain adaptation. It is the benchmark for gaze estimation on MPIIGaze test set. Our performance is almost similar to the current state-of-the-art SimGAN. SimGAN reported pre-adaptation error of 11.2°while the error reduces to 7.8°after adaptation making a relative improvement of 30% . We were not able to reproduce preadaptation error of 11.2°of SimGAN due to limited information made public. Before adaptation, we attained a mean error of 12.1°and after adaptation we show a relative improvement of 35.

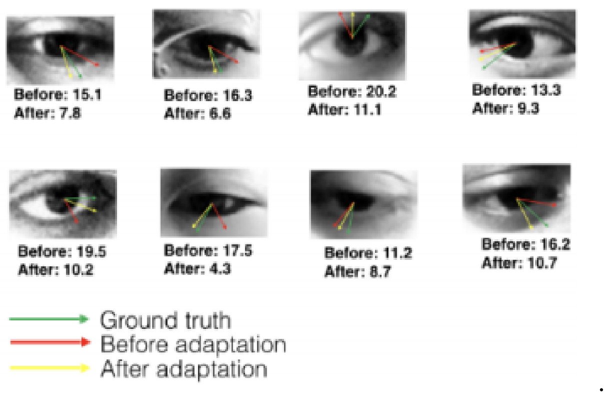


FIGURE 7.2: Visualization of gaze vector.

In fig 7.2 we visualize some examples, showing that vectors after adaptation moves closer to ground truth as compared to angle before adaptation.

# Chapter 8

# Conclusion

In this report, we presented an unsupervised approach for eye gaze estimation using domain adaptation for learning to predict in real life by using a large pool of simulated synthetic images generated from graphics engine. Our approach was more GAN like approach where we fix the *source* distribution and try to approximate this stationary distribution with a dynamic *target* distribution. This is contrary with the traditional trend of 'gradient reversal' genre of adversarial adaptation, where *source* and *target* distribution are both non stationary and updated simultaneously.

We also showed the importance of using semantic loss. It showed significant improvement in performance by much more efficient domain alignment. Our method achieved very close to benchmark performance ($7.85°$) compared to the current state-of-the-art SimGAN ($7.8°$). However it is interesting to note that we yield a relative improvement of 35% with respect to pre-adaptation performance as compared to SimGAN showing 30% relative improvement.
Our results suggest that it might be prudent to approach domain adaptation in feature/*latent* space as compared to pixel level domain adaptation done in SimGAN. This work in a first attempt of adversarial domain adaptation between UnityEyes and MPIIGaze. In future, we will combing this approach with pixel level adaptation appraoch of SimGAN. These methods are complementary to each other hence making joint optimization of pixel and features interesting to try out.

# Bibliography

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

[2] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” 2016.

[3] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” 2018.

[4] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, “Generating images with recurrent adversarial networks,” 2016.

[5] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” 2016.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” 2015.

[7] K. Kamnitsas, C. Baumgartner, C. Ledig, V. F. J. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, A. Nori, A. Criminisi, D. Rueckert, and B. Glocker, “Unsupervised domain adaptation in brain lesion segmentation with adversarial networks,” 2016.

[8] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, “Deep reconstruction-classification networks for unsupervised domain adaptation,” 2016.

[9] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012.

[10] R. Caseiro, J. F. Henriques, P. Martins, and J. Batista, “Beyond the shortest path : Unsupervised domain adaptation by sampling subspaces along the spline flow,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," *2011 International Conference on Computer Vision*, pp. 999–1006, 2011.

[12] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," 2015.

[13] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?," 2016.

[14] K. Lee, H. Kim, and C. Suh, "Crash to not crash: Playing video games to predict vehicle collisions," in *ICML 2017*, 2017.

[15] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," 2016.

[16] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," 2016.

[17] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," 2016.

[18] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," 2017.

[20] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018.

[21] Y. Zhang, Y. Zhang, Y. Wang, and Q. Tian, "Domain-invariant adversarial learning for unsupervised domain adaption," 2018.

[22] H. Hou, J. Huo, and Y. Gao, "Cross-domain adversarial auto-encoder," 2018.

[23] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. V. den Broeck, "A semantic loss function for deep learning with symbolic knowledge," 2017.

[24] S. Xie, Z. Zheng, L. Chen, and C. Chen, "Learning semantic representations for unsupervised domain adaptation," in *Proceedings of the 35th International Conference on Machine*

*Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 5423–5432, PMLR, 10–15 Jul 2018.

[25] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[26] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016.

[27] T. Schneider, B. Schauerte, and R. Stiefelhagen, "Manifold alignment for person independent appearance-based gaze estimation," in *Proceedings of the 2014 22nd International Conference on Pattern Recognition*, ICPR '14, (USA), p. 1167–1172, IEEE Computer Society, 2014.

[28] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Adaptive linear regression for appearance-based gaze estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 2033–2046, 2014.

[29] Y. Sugano, Y. Matsushita, and Y. Sato, "Learning-by-synthesis for appearance-based 3d gaze estimation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, (USA), p. 1821–1828, IEEE Computer Society, 2014.

[30] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

[31] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an appearance-based gaze estimator from one million synthesised images," in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research  Applications*, ETRA '16, (New York, NY, USA), p. 131–138, Association for Computing Machinery, 2016.