INDIAN INSTITUTE OF TECHNOLOGY, MADRAS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CS6910: FUNDAMENTALS OF DEEP LEARNING

COURSE PROJECT

Image to Image Translation for Domain Adaptation [1]

' *Prash Goel - CS20D019*

Anirudh Ajith - CS18B070

Arihant Samar - CS18B052

January 31, 2021

Introduction

The success of CNN models can be attributed to the presence of a large amounts of labelled data which allows us to train very complex models on these data and predict the labels (be it for classification, segmentation, detection or any other problem). The availability of the labelled data is critical to train huge models that work extremely well on these tasks. The same task (for example classification of digits or of objects) can require different data for different entities or individuals based on the domain of the images in their test use case. But, gathering such annotated data for every new task would be challenging, expensive and time consuming. So, it is worthwhile to look at techniques where we can adapt a classifier trained to classify images in one domain (**source domain**) to classify images of a different domain (**target domain**). This would help in transferring the learned knowledge from source to target domain, and at the same time, reduces the annotation work for target domain. Obviously, applying a classifier trained on source domain to the target domain directly would not produce impressive results because of different distribution of the data in the two domains.
The authors of the paper that we are implementing ([1]) try to combat this problem by using an intermediate **shared domain** which facilitates the conversion between the source and the target domain. This shared domain should have certain characteristics so that the domain adaptation produces decent results:

- The shared domain feature vectors should be domain agnostic i.e they should not contain domain-specific characteristics.
- We should be able to reconstruct the image in the original domain from the shared domain.
- Cycle consistency should be maintained.

The framework of this paper (Figure 1) is constructed in a way that all these characteristics can be achieved. Also, it is able to work for the domain adaptation and image to image translation problems simultaneously.

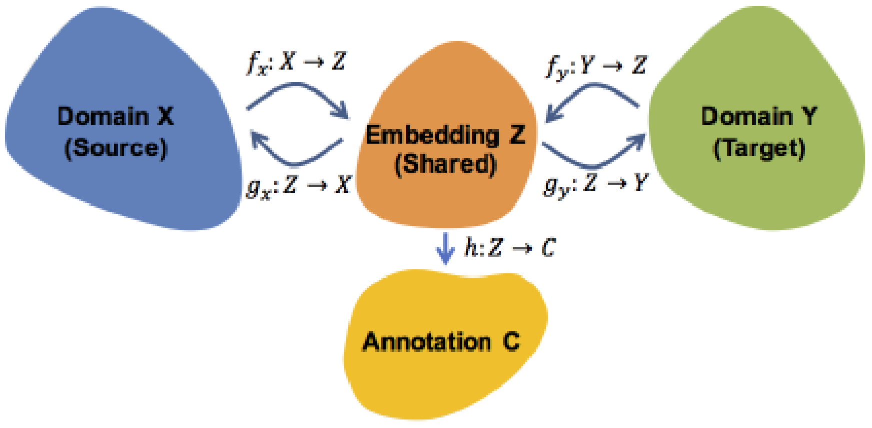

Figure 1: An overview of the framework used by Murez et al.[1]. Figure taken from the same paper.

Implementation Details

The code of the paper was not available. So, we implemented it ourselves from scratch. We followed the implementation details given in Sections 3 and 5 in the paper. For the sake of completeness of the report, we briefly discuss give a high level overview of the model here as well. Figure 2 shows the architecture of the model.

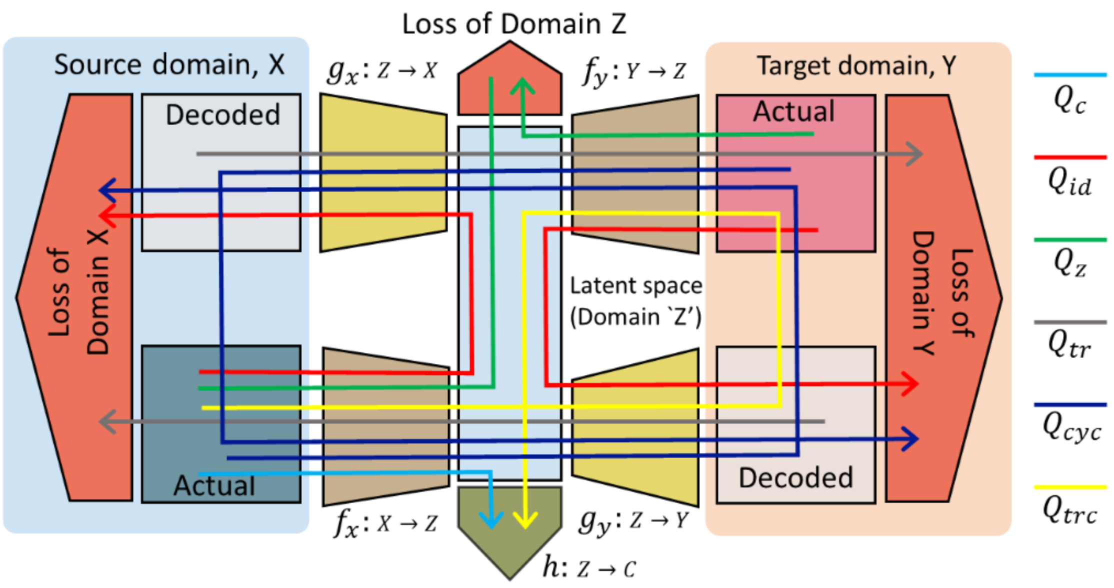

Figure 2: The architecture of the domain adaptation network used by Murez et al.[1]. Figure taken from the same paper.

The source domain X is the one for which we have the annotated classes as well. The target domain Y is that domain for which the annotations are not available. The main goal is to find a shared latent space Z where the representations of both X and Y domains are domain agnostic. To achieve this objective along with the characteristics discussed in the last section, this paper proposes a solution that combines the use of following models into a single framework:

- **Encoders** $f_x : X \mapsto Z$ and $f_y : Y \mapsto Z$. They convert the images from source and target domains to a feature vector in the shared domain. These are implemented using CNNs.
- **Classifier** $h : Z \mapsto C$ that maps the feature vectors in the latent space to the annotations. It is implemented using a simple Feed Forward Neural Network.
- **Decoders** $g_x : Z \mapsto X$ and $g_y : Z \mapsto Y$ that reconstruct the source and target domain images from the feature vectors in the shared domain. This will ensure that the feature vectors in the shared domain are able to capture important information from both the domains needed to reconstruct the images back. These are implemented using the transposed-convolutional layers.
- **Discriminator**: d_z which classifies whether a feature vector was generated from the source or the target domain. The encoders are trained to maximise the error that this discrim-

inator makes, while this discriminator tries to minimise its error. This way, the feature vectors in the shared domain are constrained to be domain agnostic.

- **Discriminators** $d_x : X \mapsto \{c_x, c_y\}$ and $d_y : Y \mapsto \{c_x, c_y\}$ which try to identify if an image is real or fake (generated using the feature vector of other domain). All the discriminators are implemented using simple feed forward neural networks.

To train all these models, we require appropriate losses as well. The complete detail of all the loss functions used is mentioned in Section 3 of [1]. We give a brief overview of the different loss functions used below:

- A cross entropy loss for classifier h to classify source images to the corresponding annotations.
- A pixel wise image reconstruction loss to ensure that reconstructed images are close to original images.
- Cross entropy losses for the discriminators to correctly classify an image in one of the two classes as described above.
- Cycle consistency loss which are again pixel wise image reconstruction loss.
- There is one additional classification loss for classifier h which classifies the feature vector generated from target domain (where the image in target domain is constructed from source domain images).
- All these losses are added together after multiplying with suitable weightage hyperparameters as mentioned in the paper.

Adam optimizer is used to train the models with discriminators being trained alternately with encoders, decoders and classifier. While training the domain discriminators (d_x and d_y, the paper has recommended to use **Improved Wasserstein Method** [2]. We used PyTorch for the implementation and the corresponding code is attached with this submission. To train the models on a GPU, we used the free services offered by Google Colab.

Experiments and Results

We tested our implementation of this paper on two sets of benchmark datasets for the domain adaptation problem, namely, **Digits** and **Office**. A brief description of these datasets is given below.

Digits Dataset

We used MNIST, USPS and SVHN domains out of the Digits dataset. MNIST dataset consists of 70,000 images in total. These images represent hand written digits from 0 to 9. They are binary images of size 28 × 28. USPS dataset contains 9300 images which are grayscale and of size 16 × 16. Again, these images are of hand written digits. SVHN dataset contains about

1,00,000 RGB images of size 32 × 32. They were converted to grayscale for this experiment. Further, all images were resized to a size of 32 × 32, and normalized to $[-1, 1]$. Sample images from all 3 domains of Digits datasets are shown in Figure 3.

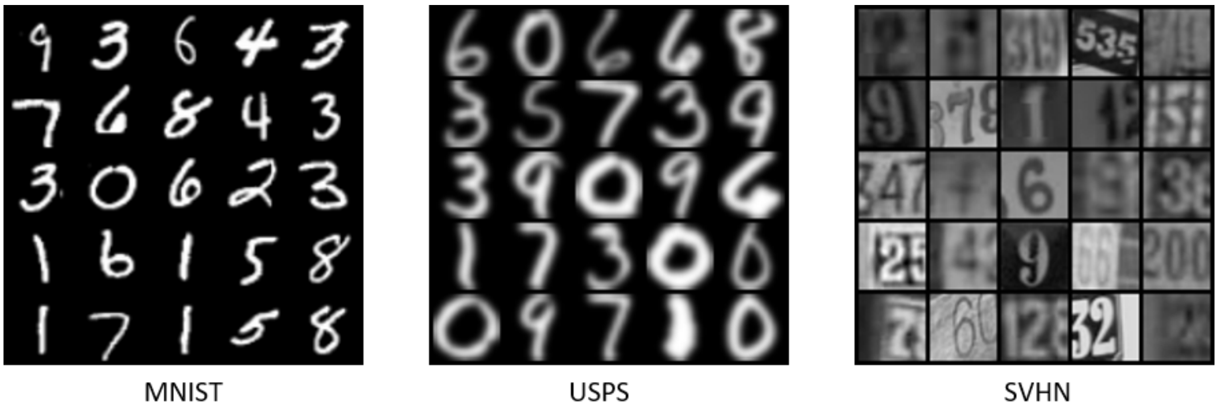

Figure 3: Sample images from all 3 domains of the Digits datasets.

Office Dataset

This is a relatively smaller dataset and with larger number of classes. It consists of images in 3 domains and classified in 31 classes. The domains are Amazon, DSLR and Webcam. These images are resized to 256 × 256, and then a random crop of size 224 × 224 is selected out of it. The different domains of Office dataset are shown in Figure 4.

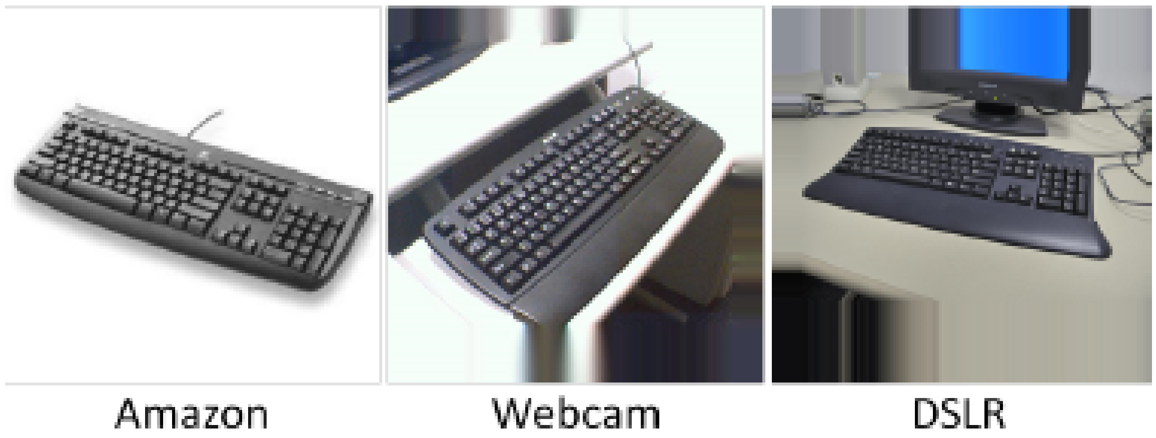

Figure 4: The original images in all 3 domains of the Office dataset.

GTA5 and Cityscapes Dataset

The paper runs the experiment on this dataset as well but this dataset is huge (50GB!), and due to unavailability of compute power to train on such a huge dataset, we haven't used this for our experiments. We got prior permission from the TA (Mr. Gouthaman KV) for this.

Results on Digits Dataset

We tried with the 3 combinations of source-target domain pairs (same as mentioned in the paper), and obtained similar results as in the paper. Table 1 compares the performance of our implementation with that of the paper and the numbers are comparable to the ones in the paper, even exceeding the accuracy of paper for USPS $\rightarrow$ MNIST task. Figure 5 shows the translated images from one domain to another and as can be seen the images look close to real images in that domain. Notice how the images generated in SVHN are able to reconstruct the specific pattern that is common across all images in SVHN domain.

Method	Our implementation	Results from paper
MNIST $\rightarrow$ USPS	91.42	**95.1**
USPS $\rightarrow$ MNIST	**93.08**	92.2
SVHN $\rightarrow$ MNIST	89.7	**92.1**

Table 1: Performance of our implementation and the results in paper on the different combinations of domains in Digits dataset. This performance indicates accuracy (in %) for digit classification task.

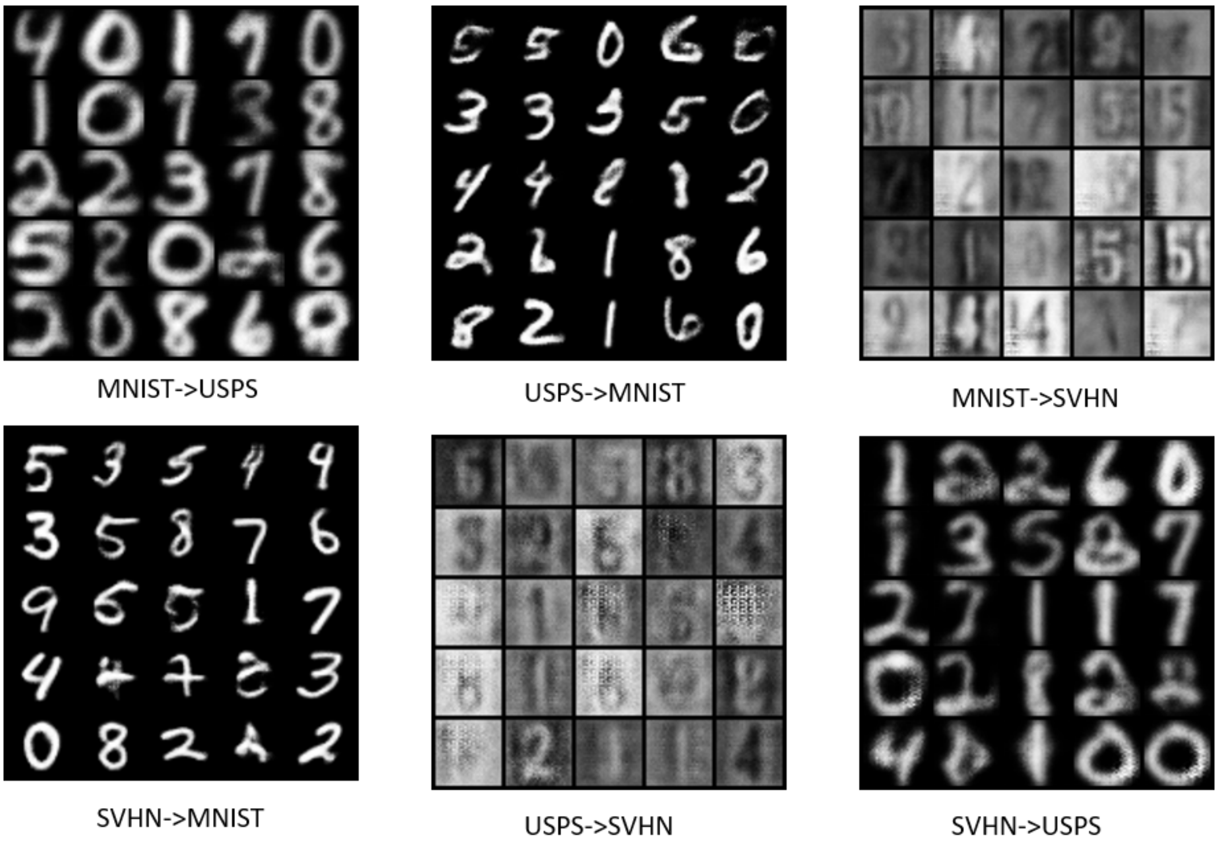

Figure 5: The translated images from one domain to the other in Digits dataset. For example, MNIST $\rightarrow$ USPS indicates that the corresponding image was generated in USPS domain by translating an image from MNIST domain.

Results on Office Dataset

We experimented with all the 6 combinations of the 3 domains in Office dataset,and were able to reproduce the results of the paper. As can be seen in Table 2, we were able to achieve better results than the implementation of paper for 2 out of 6 tasks. For the remaining 4 tasks, our accuracy was close enough to the ones mentioned in the paper. We found that training the model longer helps improve the accuracy, but due to lack of computational resources, we couldn't train the models further to improve their accuracy.
We also tried with a better pixel loss for images ,the **MS-SSIM** loss but we did not find any improved performance .

Method	**Our implementation**	**Results from paper**
Amazon $\rightarrow$ Webcam	**76.35**	75.3
Webcam $\rightarrow$ Amazon	**52.21**	52.1
Webcam $\rightarrow$ DSLR	96.18	**99.0**
Amazon $\rightarrow$ DSLR	69.9	**71.1**
DSLR $\rightarrow$ Webcam	94.3	**96**
DSLR $\rightarrow$ Amazon	47.2	**50.0**

Table 2: Performance of our implementation and the results in paper on the 6 combinations of domains in Office dataset. This performance indicates accuracy (in %) for image classification task.

Limitations of this paper

While experimenting with the models, we found out a few issues with this approach. The same are being presented here.

1. **Dependence on weight initialization** - The target domain images generated after translating the source domain images via the shared domain (or vice versa) have some artefacts like presence of regularly spaced white spots. Figure 6 shows clearly that the white spots forms a pattern in all the 25 images of one instance. We trained the same model multiple times to check if these are consistent but noticed that these white spots were absent in some of the trials. This led us to conclude that the model and the architecture depends heavily on how the weights are initialized.

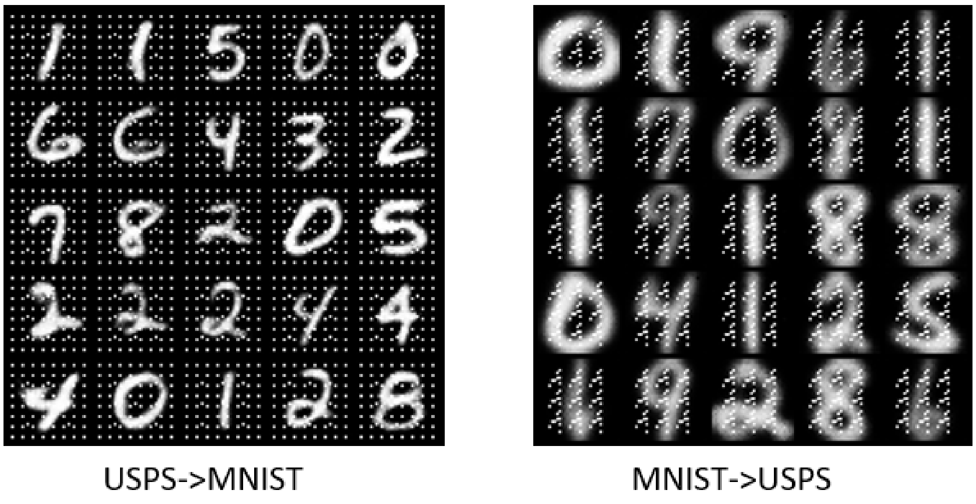

Figure 6: The images generated after translating source domain images to shared domain or vice versa.

2. **Inability to generate proper images for Office dataset**: We checked the translated images for Office dataset as well, but the results were not very impressive (Figure 7). We double checked our implementation, and the numerical results even match with those in the paper (Table 2). Unfortunately, the paper as well doesn't provide generated samples for this dataset. One reason for this might be that this dataset was very small (2000 images), which might not be enough to train a complex model to generate images well. Although, quantitatively, Table 3 in [1] claims that this approach was best in 4 out of the 6 tasks on this dataset.

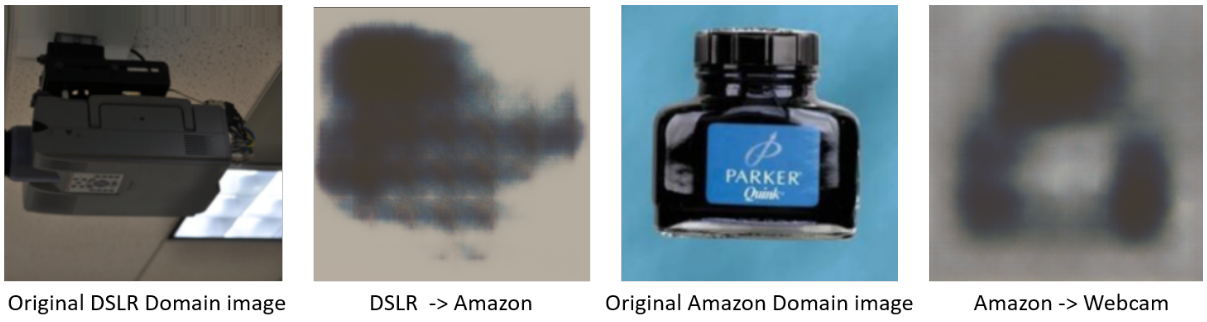

Figure 7: The original image and the image translated image in a different domain in Office dataset.

3. **Unable to perform well when generating RGB images**: As an extension of the experiments on Digits dataset, we tried out using the RGB images of SVHN dataset instead of converting them to grayscale. Even though, the generated images were not very bad, but the accuracy of digit classification when SVHN images were in target set was very poor (10%).

Conclusions

We implemented the paper [1] from scratch and tested the model on 2 datasets : Digits and Office. Since we did not have any reference implementation of this paper, we had to implement it from scratch and since the implementation was a bit involved, it took good amount of effort. But, eventually we were able to reproduce the results of this paper successfully (even exceeding the accuracy of paper in some cases), and even went a step ahead to experiment a few things which helped us in finding some limitations in the technique of this paper.

References

[1] Zak Murez et al. Image to image translation for domain adaptation. *CoRR*, 2017.

[2] Gulrajani et al. Improved training of wasserstein gans. *CoRR*, 2017.