GAN-based Domain Adaptation

236605 – Deep Learning on Computational Accelerators

Project Report

Based on the paper linked here

Project by:

Bahjat Kawar – 206989105 – bahjat.kawar@campus.technion.ac.il

Yishayahu Goodman – 203207881 - yishayahu@campus.technion.ac.il

Abstract

As the paper we're basing our work on notes, domain adaptation is an actively researched subject in computer vision and machine learning. Said paper suggests a domain adaptation approach that is based on Generative Adversarial Networks (GANs). This allows us to leverage unlabelled data to aid in the task of domain adaptation, and it differs from the traditional use for GANs, which is generating close-to-real data to retrain classifier models.

In the paper, they showed the example of digit classification, where they adapted learning from the SVHN dataset to the MNIST dataset. In our work, we attempted to achieve domain adaptation in the opposite direction: to adapt learning on the MNIST dataset to the SVHN dataset.

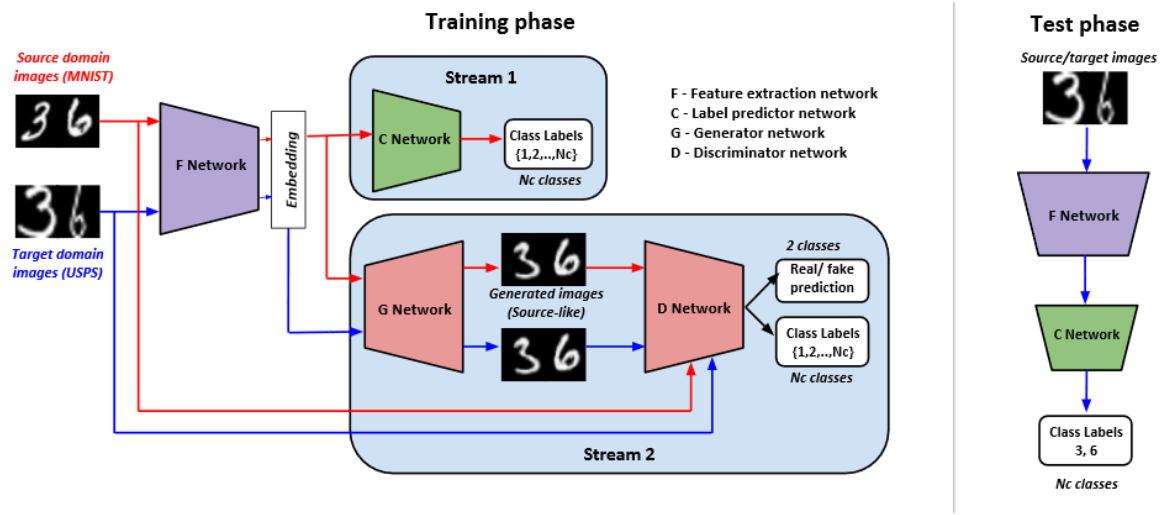
We experimented with different model architectures, while still adhering to the framework set by the paper, and we achieved more than 2x the accuracy score that the paper's original architecture did. We consider this a very significant improvement, but not enough to constitute a good classifier.

Introduction

Many modern machine learning and deep learning algorithms, such as Convolutional Neural Network (CNN)-based classifiers have been providing great performance in solving a lot of computer vision and image classification problems. Although labelled data is becoming increasingly available, it's still an expensive resource to obtain, and the algorithms require a lot of it. Another drawback of the need for labelled data, is that the actual real-world data sometimes differs (in its distribution) from the training data, which leads to a decrease in performance. This triggers the need for the use of domain adaptation, using labelled source data and unlabelled target data.

The paper we're studying discusses unsupervised domain adaptation. Their approach employs a Generative Adversarial Network (GAN), whose latent space (embedding space) is robust to the shift between the source and target distributions. They run a learning procedure using labelled data from the source distribution, and that procedure is guided using unsupervised data from the target distribution.

While there has been some prior work using adversarial methods for the purpose of domain adaptation, the novelty that this approach is that it uses a combination of classification loss and a GAN to learn the embedding. The paper shows that it achieves state-of-the-art results.



In the above illustration, the paper authors describe their general framework for their approach. During training, the source data is passed through the embedder network (F network), which outputs an embedding that is then used as an input to the classifier for predicting the label (C network), and also as an input for the generator network (G network), whose output is controlled by the discriminator (D network). The embedding is updated based on the back-propagated gradients from both routes.

When given unlabelled data from the target domain, the embedding is update using only the gradient from the adversarial route (G and D), because the labels are unavailable.

The main contribution of this paper is learning a joint feature space where the distance between the source and target distributions is minimized, using a GAN approach. Their experiments show better, state-of-the-art results than other approaches.

We used the paper's approach to tackle another domain adaptation task: given labelled data from the MNIST dataset, and unlabelled data from the SVHN dataset, we try to apply the domain adaptation process. In our experiments, we compared different possible architectures for the aforementioned F, C, G, and D networks, tuned their hyperparameters, and achieved a satisfactory improvement over the paper's original architecture.

The problem we are trying to tackle, which is adapting learning from the MNIST dataset to the SVHN dataset, is harder than the one solved in the paper. This is because the SVHN dataset models a real-world problem which is identifying numbers in street signs, whereas MNIST is more of a controlled lab dataset. Achieving good results here means that we can transfer learning done using easily labelled data to real-world samples that are not necessarily labelled.

Unfortunately, we did not achieve results that are good enough to be useful as a classifier, but we did achieve a significant improvement over the original paper's architecture. This improvement suggests that further progress may be made, using architectures from future work, under a similar framework to this one.

Methods

We based our code on an implementation of the code made public by the authors of the paper. We mostly used their code for running the experiments.

However, in order to improve performance on the task at hand, we modified the neural network architecture. We did that several times, in the hopes of achieving better results. We believe that the general framework given by the paper is good, but the network architectures need to be tweaked for each task.

In the beginning we ran the algorithm with the original architecture on the original task (from SVHN to MNIST) as a sanity check, and we made sure we got an accuracy score that is close to the one in the paper.

As for our task (from MNIST to SVHN), we decided to try to run it using the same architecture used for the opposite direction first, in order to get baseline metrics.
We also thought of running it using the other architecture described in the paper, because it was implemented and proven useful in different tasks in the paper.

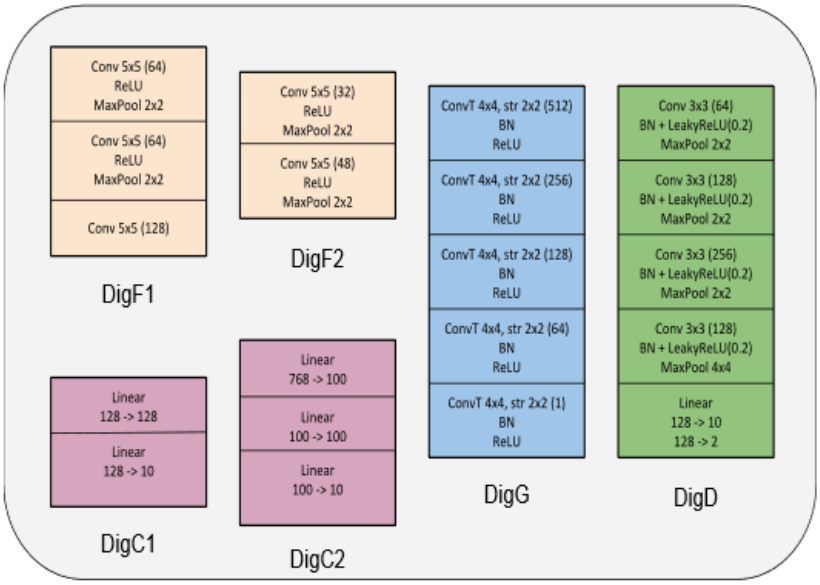
In addition, we found another paper that discusses using GANs for domain adaptation, and we decided to mimic their GAN network implementation into our framework (the G and D networks), and kept the original implementation for the F and C networks. The linked paper gives us theoretical motivation to try this approach. We believed that unifying approaches from both papers may yield better results. Another reason for conducting this experiment is that the original model performed too well on the source dataset, and too poorly on the target dataset, thus we deduced that the GAN was the weakest link in the setup, because its job is to make them similar.

As for evaluation, since we are performing domain adaptation from MNIST to SVHN, the training SVHN data is unlabelled. In practice, we do have labels for SVHN, which we will use in order to evaluate the resulting trained model's accuracy on the test set.

Experiments

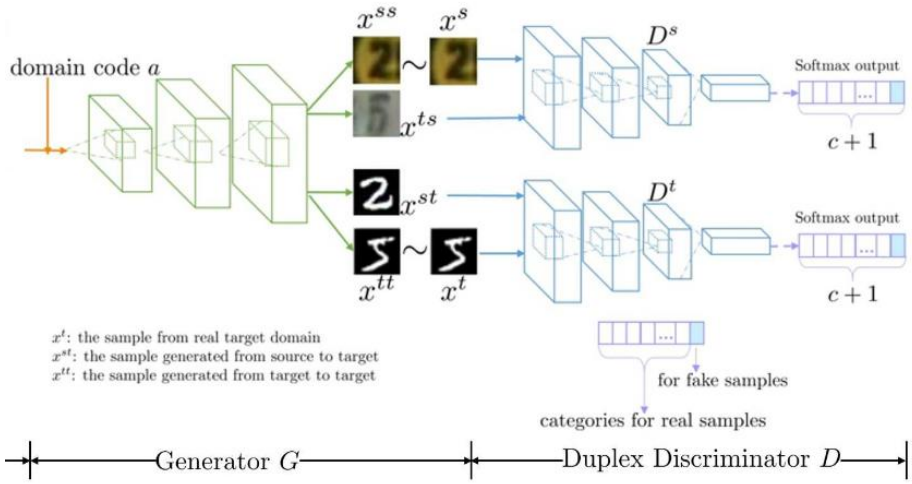
We ran all experiments using the same data: MNIST was used as the labelled source dataset, while SVHN was used as the unlabelled target dataset. Both are well known datasets for digit identification.

In the first experiment, we used the architectures that were used for the reverse task by the paper authors.
Namely: DigF1, DigC1, DigG, and DigD.



In the second experiment, we used the same architecture as in the first for the generator and discriminator, but we used DigF2 and DigC2 for the embedder and the classifier, respectively.

In the third experiment, we used DigF1 and DigC1 for the embedder and classifier, respectively. However, for the generator and discriminator, we implemented the following architecture:

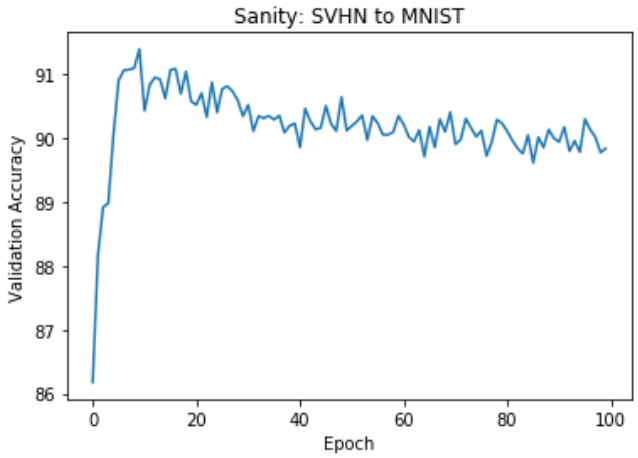


The model is based on the idea of using 2 discriminators, one for the source dataset and another for the target datasets, but only one generator. The idea is that identifying fake and real samples in each dataset is a different task.
The generator architecture consists of 4 transposed convolution layers which turn embedding vectors into generated samples.
The discriminator architecture consists of 4 convolutional layers. In the source discriminator, the output is one of 11 labels (one for each class, and one for fakes), and in the target discriminator, the output is only real or fake, since we don't have training labels for the target.
As in the original paper, both discriminators back-propagate losses. However, unlike the original paper, the discriminators will only produce one loss (instead of one for real/fake and another for the class), under the assumption that both can be combined in one loss.

After running all experiments, we evaluated results based, first and foremost, on the accuracy of classification on the test set, and also based on the speed of convergence, as shown in the graph for the accuracy on the validation set, every epoch, for each experiment.

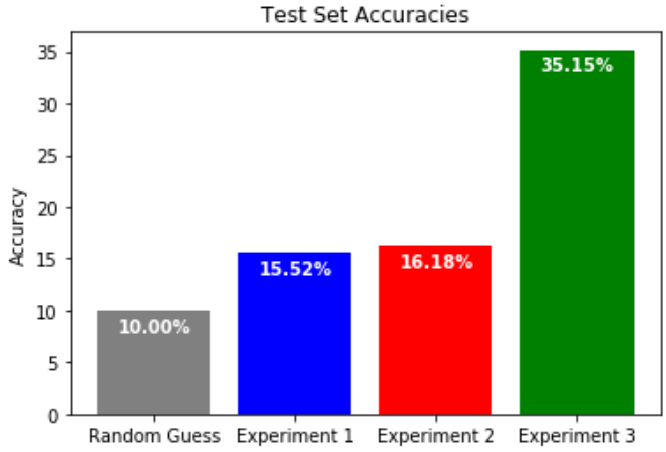
Results

Before running our experiments, we ran a sanity check on the same task the original paper authors ran: from SVHN to MNIST.



Based on the graph, we can see that the model converges quickly to an accuracy of about 90% on the validation set, which is close to the accuracy score reported in the paper. The model achieved a similar accuracy on the target test set as well.

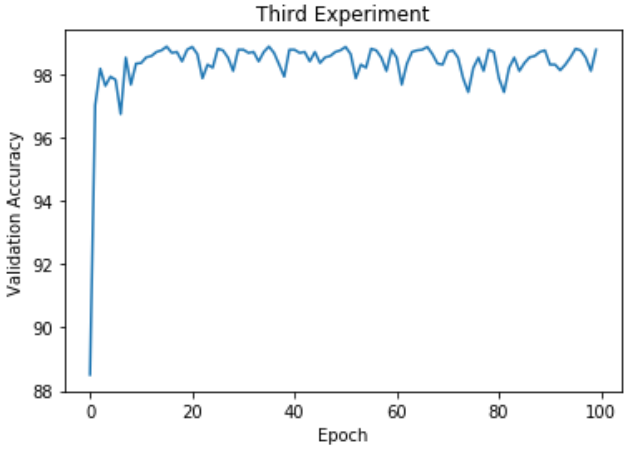
For the experiments on MNIST→SVHN, we got these results:



Random guess is the expected accuracy we would get if we just guess at random. Any classifier with less than 1/n_classes can be considered useless. We can see through these results that the architecture we proposed in experiment 3 achieves an improvement of 117.17% over experiment 2 in terms of accuracy on the test set.

The 1st and 2nd experiments used the original paper's model architectures. They achieved accuracies of 15.52% and 16.18%, respectively. These accuracies indicate that the models did learn *something*, but they are not good results (which may be the reason the original paper authors did not address this task). The poor performance makes sense, since the MNIST→SVHN problem is harder than the opposite direction, and models a real-world problem more strongly.

In the 3rd experiment, we introduced a new model architecture for the generator and discriminator, which vastly improved the performance. The test set accuracy more than doubled. The accuracy score is still not good enough for use, but we have made a significant improvement over the original architecture's numbers.



Based on the graph we can see that the model converges quickly to a little less than 99% on the source validation set.

By analysing the results of the first two experiments, we concluded that the model has learned the source dataset (MNIST) *too well*, to the point where it hardly learned anything on the target dataset. It achieves an accuracy of above 98% on MNIST. The third experiment achieved similar results on MNIST, but improved on SVHN.

We tried tweaking all 3 experiments' parameters, and try out different combinations of architectures, but did not get any better results than the ones shown here.

It is worth mentioning that in the paper we linked, they achieved an accuracy of 62.65% which is remarkable. This huge difference in numbers shows the quality of the linked paper, but we did not study it further because it's out of the scope of this project. 62.65% still does not make a good classifier, and it was reported in a paper as the best accuracy yet. This further emphasizes the difficulty of the problem that we attempted solving.

In conclusion, the architecture we introduced maintained the same level of accuracy on MNIST (the source dataset), while improving by 117.17% on SVHN (the target dataset). We did not achieve state-of-the-art results (because of the other paper), but we did improve upon the paper that we worked with.