CS698O: Unsupervised Pixel–Level Domain Adaptation with Generative Adversarial Networks

Vaibhav Nagar 14785 Ankur Kumar 14109

Group 10

Introduction

We build our models from some fixed source domain, but we wish to deploy them across one or more different target domains. Also labeled data (well-annotated image datasets) is often expensive to obtain, and frequently requires lots of human effort. An alternative is to render synthetic data where ground-truth annotations are generated automatically. Models trained purely on these rendered images often fail to generalize to real images having different distribution. To address this shortcoming, prior work introduced unsupervised domain adaptation algorithms that attempt to map representations between the two domains or learn to extract features that are domain–invariant. In this research paper "Unsupervised Pixel–Level Domain Adaptation with Generative Adversarial Networks"[1], authors present a new approach that learns, in an unsupervised manner, a transformation in the pixel space from one domain to the other based on generative adversarial network (GAN) which adapts source-domain images to appear as if drawn from the target domain.

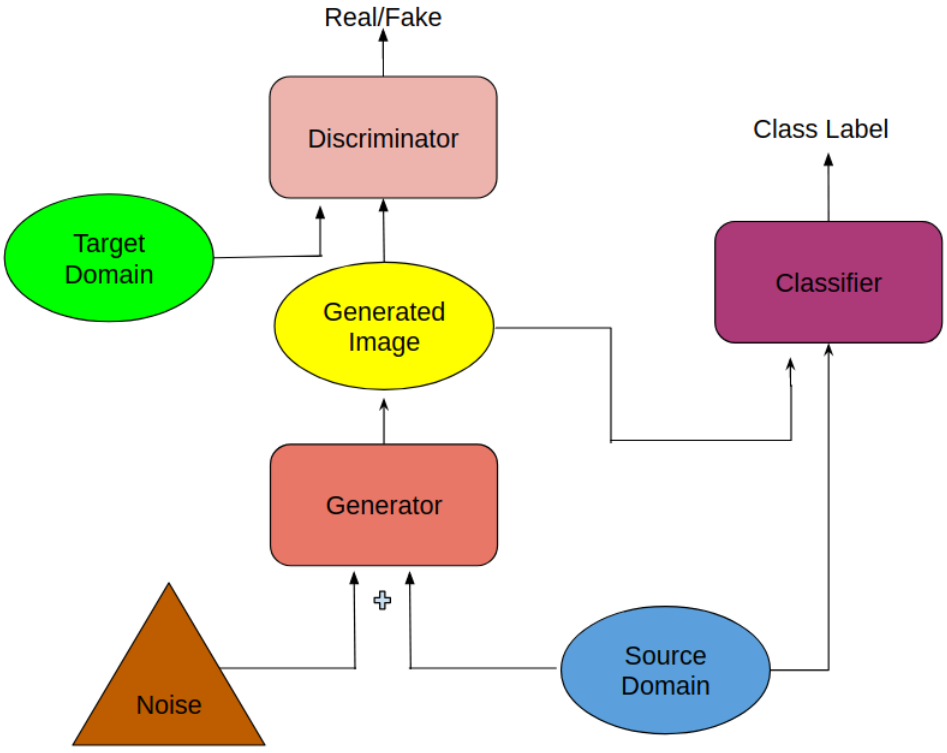


Figure 1: Proposed Model

Model Architecture

Model proposed by authors of the paper is not specific to any task. However for the sake of proper explanation, their work focuses on *classification* task. Therefore re-formed problem statement becomes: Given labeled dataset from source domain and unlabeled dataset from target domain, the goal is to learn a classifier trained on source data that generalizes well on target domain. Since the goal is to generalize well to unseen data, authors propose modular approach which will fit any task on targeted domain. Their model can be understood by the figure shown above. The idea of the model is as follows:

- *Generator block (G):* This part of the proposed model takes as input an image from source domain and some noise vector. The output is an image that belongs to targeted domain (a part of model training objective). One thing should be clear that target domain and source domain differ in noise, resolution or such minor variations. Data form these domains do not differ in high level constructs.
- *Discriminator block (D):* This is another part of *GAN* (Generative Adversarial Network). This module determines if an image belongs to source domain or target domain (second objective that model training tries to achieve). Discriminator should output *real* label if image belongs to source domain, *fake* otherwise.
- *Classifier (T):* Again, task need not be classification. Classifier classifies an input image into one of several classes. Since we are working with *MNIST, MNIST-M and USPS* dataset, there are 10 output classes: digits from 0 to 9.

Generator, with the help of discriminator, tries to output image which is as close to the target domain. Simultaneously, classifier forces the output of generator to be as relevant as possible.

Training

Our generative adversarial network aims at generating image that is as close to target domain as possible. Let x^s denote an image from source domain, x^f as generated image and x^t from target domain. Also, let z denote noise vector and θ_G, θ_D and θ_T denote parameters for *G, D* and *T* respectively. Therefore $G(x^s, z; \theta_G)$ denotes output image of generator block, x^f, given an input image from source domain along with some noise. $D(x; \theta_D)$ denotes output of discriminator (which can be a binary label or softmax value depending on model) given any input image. Note that discriminator will try to attach *real* label to x^t and *fake* label to x^f. Similarly for $T(x; \theta_T)$. So the overall training procedure unfolds as follows:

- Generator takes as input noise z and an image x^s from source dataset. Based on the input, it outputs an image x^f.
- Discriminator takes an image x^t from target dataset and update its weight based on the loss value (also known as *Domain Classification Loss*). Note that output label should be *real* in this case.
- Classifier is fed in x^s and it also updates its weight based on the loss value (also known as *Task-specific Loss*). Here also, classifier should output class which should be same as that of x^s (source dataset is labeled), otherwise it incurrs loss.
- Finally, generator updates its weights based on the output from discriminator and classifier when input to both of them is x^f. Note that discriminator should label x^f as *fake*. For classifier, x^f should belong to same class as that of x^s from which it was constructed.

This cycle continues till discriminator fails badly on generated image, i.e., it should not be able to distinguish between generated image and target domain images. As the classifier is being trained on generated images also (which we hope to be similar to target dataset), we expect the classifier to perform well on target domain as well.

In the training process, we encountered two types of losses:

- $L_d(D, G)$: Domain classification loss that discriminator and generator incurr. Discriminator incurrs loss when it labels x^t as *fake*. Generator incurs loss domain loss when discriminator labels x^f as *fake*.
- $L_t(T, G)$: Task-specific loss that task, here classifier, and generator incurr. Classifier incurs loss when it predicts wrong class for x^s. Generator incurs loss when classifier predicts wrong class for x^f.

So, the learning alogrithm has to optimize the following minimax objective:

$$\min_{\theta_G, \theta_T} \max_{\theta_D} \quad \alpha L_d(D, G) + \beta L_t(T, G)$$

Exact expression of *domain classification loss* depends on which *GAN* is used. The domain loss in case of PixelDA GAN has following expression:

$$L_d(D, G) \;=\; E_{x^t}[log\ D(x^t; \theta_D)] \;+\; E_{x^s, z}[log(1 - D(G(x^s, z; \theta_G); \theta_D))]$$

In other cases, exact expression varies slightly but gist remains the same. Similarly, exact expression of *task specific loss* depends on which classifier is used. We used MNIST classsifier (as we experimented with MNIST as source dataset) with *cross entropy loss* with *softmax logits.* We experimented with following *GANs* in our implementation:

- PixelDA GAN [1]
- DCGAN [2]
- WGAN [3]
- LSGAN [4]
- Softmax GAN [5]

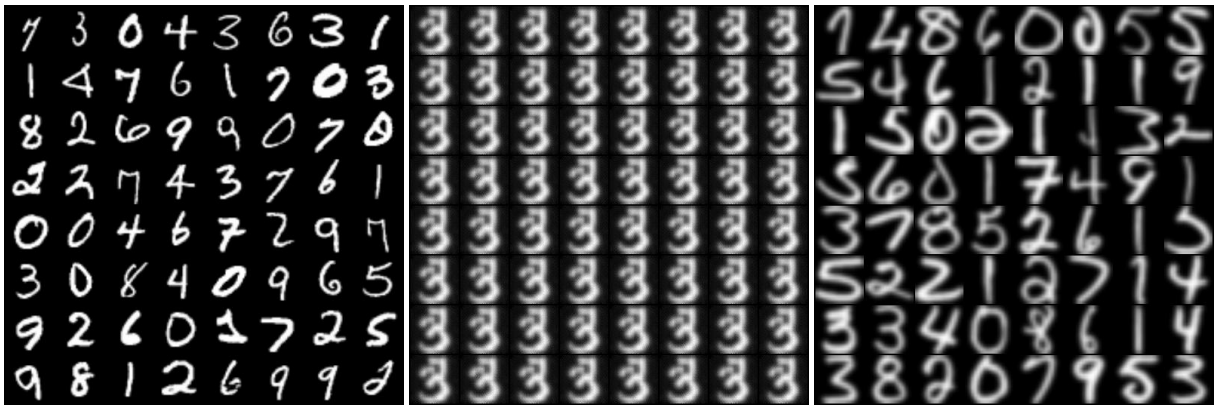
Experiments

We experimented with *MNIST* dataset as our source dataset. For target domain dataset, we used *MNIST-M* and *USPS* dataset. We evaluated our implementation on both these pairs, i.e. from *MNIST* to *MNIST-M* and from *MNIST* to *USPS*. We carried our evaluation both qualitatively and quantitatively. Qualitatively, we compare images generated from our implemented GAN with that of target and source domain images. Quantitatively, we calculated performance of our implemented model on these settings. Baseline comparison for this problem statement was performance of *source only* trained model, i.e. model trained on source dataset is tested on target dataset. We implemented baseline as well along with prior results for baseline.

MNIST to USPS

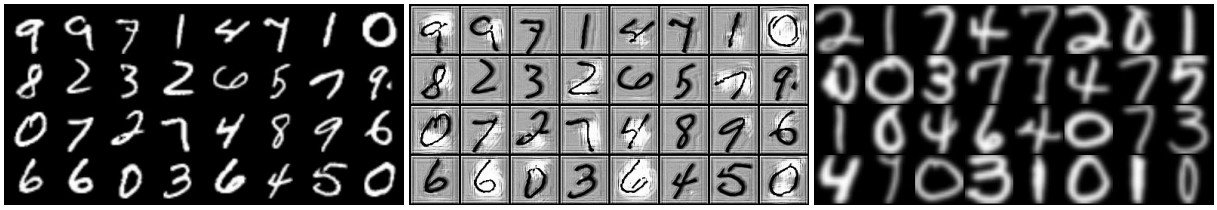
USPS dataset contains 7291 images for training and 2007 images for testing purpose with each digit class in proportion. The available USPS data was available in .mat format, each image normalized in -1 to 1. We experimented with normalized MNIST dataset as source domain when

target domain is USPS. We experimented with different GANs in this case only. Qualitative results obtained are as follows:



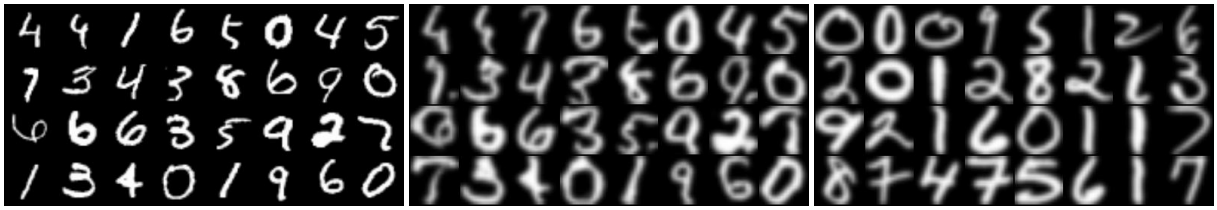
(a) Image Samples from MNIST dataset (source domain) (b) Generated Image Samples (c) Image Samples from USPS dataset (target domain)

Figure 2: DC-GAN



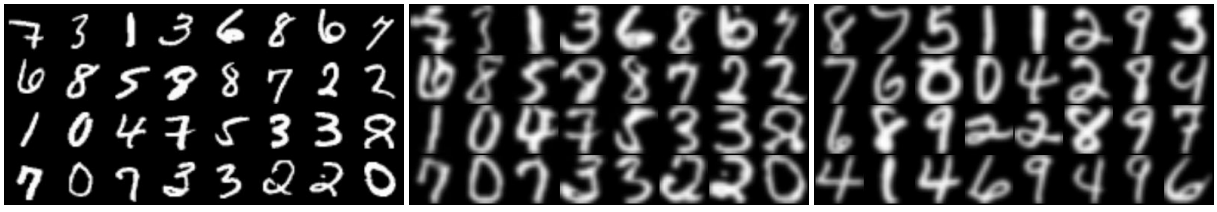
(a) Image Samples from MNIST dataset (source domain) (b) Generated Image Samples (c) Image Samples from USPS dataset (target domain)

Figure 3: WGAN



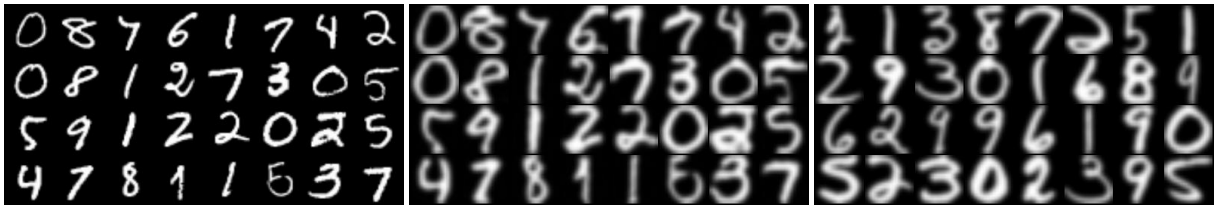
(a) Image Samples from MNIST dataset (source domain) (b) Generated Image Samples (c) Image Samples from USPS dataset (target domain)

Figure 4: Softmax GAN



(a) Image Samples from MNIST dataset (source domain) (b) Generated Image Samples (c) Image Samples from USPS dataset (target domain)

Figure 5: LS-GAN



(a) Image Samples from MNIST dataset (source domain) (b) Generated Image Samples (c) Image Samples from USPS dataset (target domain)

Figure 6: PixelDA-GAN

Model performances for different GANs are as follows:

Model	Performance (MNIST to USPS)
Source only	78.9% (56.8%)
DC-GAN	Mode collapse
W-GAN	37.2%
Softmax-GAN	90.3%
LS-GAN	**97.2%**
PixelDA-GAN	**96.01%**
Target-only	96.5% (99.0*%)

Table 1: Mean classification accuracy (%) for digit dataset (test only)- MNIST to USPS. The "Source-only" and "Target-only" rows are the results on the target domain when using no domain adaptation and training only on the source or the target domain respectively. We note that our Source and Target only baselines resulted in different numbers than published in the paper which we also indicate in parenthesis.

MNIST to MNIST-M

MNIST-M dataset was created primarily for unsupervised domain adaptation challenges. It was created from MNIST dataset by considering digit as binary mask and changing the remaining with some background patches. Images from this dataset contains 3 channels, whereas MNIST images contain single channel only. Therefore, we expanded MNIST images by simply stacking original image data three times. However, the approach taken in the paper differs. They introduce private layer which allows different channel data to share high level layers of the model. MNIST-M contains 59000 training images and 9000 test images. We evaluated our implementation for this target domain for PixelDA-GAN only. We obtained the following qualitative results:

(a) Image Samples from MNIST dataset (source domain) (b) Generated Image Samples (c) Image Samples from MNIST-M dataset (target domain)

Figure 7: PixelDA GAN

Model performances for PixelDA-GAN is as follows:

Model	Performance (MNIST to MNIST-M)
Source only	63.6%
PixelDA-GAN	**96.05%**
Target-only	95.9% (97.2%)

Table 2: Mean classification accuracy (%) for digit dataset (test only)- MNIST to MNIST-M. The "Source-only" and "Target-only" rows are the results on the target domain when using no domain adaptation and training only on the source or the target domain respectively. We note that our Source and Target only baselines resulted in different numbers than published in the paper which we also indicate in parenthesis.

Conclusion

We carried out experiments with different settings as mentioned above. From the result that we got, we can clearly say that the **proposed model outperforms the baseline**, which is a model trained on source only dataset, **and other prior works aimed at similar objective**. However in our experiments, we found that **replacing PixelDA GAN with LS-GAN resulted in even better performance**. In our implementation, we worked on only *classification* task. For testing how well does the model generalize, authors of the paper performed *classification* task and *pose estimation* task on unseen object categories, i.e. model was trained on only 6 classes and it was tested on rest of the 11 classes. The results in these cases suggest that model is able to adapt to target domain which is not specific to any task or training dataset. In our quest to implement this outstanding work, we observed that model gets into mode collapse when GAN part is replaced with DC-GAN (as evident from generated image for DC-GAN). Authors of the paper have argued that while task loss does not affect overall performance, they stabilize these adversarial models (instability of adversarial models has been key problem till date). However, it remains for us to find out the cause for mode collapse. Finally, one of the several objectives of the original work was to decouple domain adaptation problem from specific task like classification. It is evident from our implementation that once the model is trained for some target domain, we can easily replace the task classifier with any other task objective and the model (GAN part) need not be trained again.

Future Work

The work done by the authors of this paper is remarkable. It has indeed taken visual recognition community several steps forward with its contribution to domain adaptation as domain adaptation is one of the challenging problems in deploying systems. We recently came across a work, SBADA GAN[7], which takes this idea further by employing two way mapping, i.e. from source domain to target domain and target domain to source domain. The authors of this paper claim that this results in more robust and more general system. We can implement this work as well to become more familiar with the ideas involved in domain adaptation challenges.

Acknowledgements

We would like to thank our course instructor Prof. Vinay P. Namboodiri as he gave us opportunity to choose our own work and get it done our way. We would also like to thank course TAs and Vinod Kumar Kurmi who helped us, when we needed a guide the most, without worrying that we were interfering in their timetable frequently.

References

[1] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., & Krishnan, D. (2016). Unsupervised pixel-level domain adaptation with generative adversarial networks.

[2] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[3] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." arXiv preprint arXiv:1701.07875 (2017)

[4] Mao, Xudong, et al. "Least squares generative adversarial networks." arXiv preprint ArXiv:1611.04076 (2016).

[5] Lin, Min. "Softmax gan. arXiv preprint." arXiv preprint arXiv:1704.06191 (2017).

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[7] P. Russo, F. M. Carlucci, T. Tommasi, & B. Caputo (2017). From source to target and back: symmetric bi-directional adaptive GAN