Homework Report

0. General

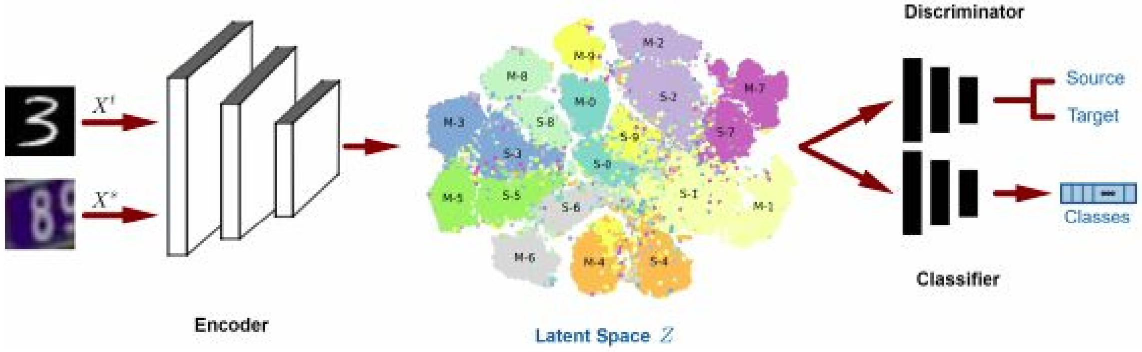
Author: Ruslan Sabirov
Group: B17-DS-02
Task description: HackMD
Code: Collab

1. Assumptions

All these results were achieved at seed = 5.

2. Model explanation

Model is usable for domain adaptation for different datasets. But for the project purposes I used only SVHN as a source domain and MNIST as a target domain.



(pic. Model architecture)

Architecture can be divided into three parts: encoder, discriminator and classifier. The encoder translates the images to embeddings in the latent space. The discriminator distinguishes if the latent representation is from source or target domain. The classifier finds most probable label for latent representation.
For the learning it uses combination of three losses (Classification loss, Discrimination loss, Separability loss).

Classification loss
$$\mathcal{L}_c(W^E, W^C) = \left(\lambda_s \sum_{x^s \in X^s} H(\hat{y}^s, y^s) + \lambda_t \sum_{x^t \in X^t} H(\hat{y}^t, y^t)\right),$$

Discrimination loss
$$\mathcal{L}_D(W^D) = -\sum_{z^s \in Z^s} \log(D(z^s)) - \sum_{z^t \in Z^t} \log(1 - D(z^t)),$$

Separability loss
$$\mathcal{L}_{sep}(W^E) = \left(\frac{\sum_{i \in Y} \sum_{z_{ij} \in Z_i} d(z_{ij}, \mu_i)}{\sum_{i \in Y} d(\mu_i, \mu)}\right) \times \lambda_{BF}, \quad \lambda_{BF} = \frac{\min_i |Y_i^t|}{\max_i |Y_i^t|},$$

Optimization
$$\mathcal{L} = \min_{W_D, W_C, W_E} \beta_C \mathcal{L}_C + \beta_P \mathcal{L}_P + \beta_{Sep} \mathcal{L}_{Sep},$$

3. Difference from first submission

This code is highly different from the one that was submitted on the first week. First, there was no domain adaptation (just datasets transformations). Second, architecture significantly differs: on the first submission there was just simple CNN for classification, here we have GAN modification (with encoder, classifier and discriminator parts). As a consequence, accuracy highly increased.

4. Solution source

I have taken the solution from paper Triplet Loss Network for Unsupervised Domain Adaptation (Imad Eddine Bekkouch, Youssef Youssry, Rustam Gafarov, Adil Mehmood Khan).
Solution was a little bit modified:
1) Unnecessary parts was erased (e.g. additional datasets)
2) Added plotting accuracies
2) Added plotting latent spaces

5. Solution for domain adaptation

The solution uses discrimination loss and pseudo labeling for DA purposes. By pseudo-labeling we provide pseudo-labels for the unlabeled samples from target domains (we should not use real-labels because we are interested in semi-supervised DA).
Task of the discriminator is to take the features of the image and try to guess is it real image or generated. Its loss helps to generator to produce samples that cannot be discriminated.

6. Hyperparameters

The most important hyperparameters are: β_{Sep}, β_C, β_P, λ_S, λ_T, *PL_Thresh*.

β_{Sep}, β_C, β_P - are balancing parameters for triplet loss.
λ_S, λ_T - are balancing parameters for classification loss.
PL_Thresh - is the minimum confidence level provided by the classifier for considering image in pseudo-labeling.

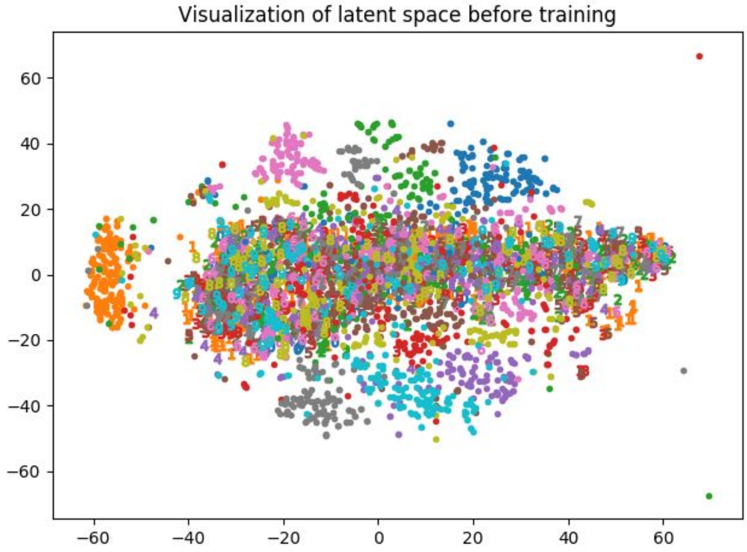
I have found the best hyperparameters combination for the model in the original paper. Despite this I tried to do grid CV search and I found out that hyperparameters from the papers are really the best ones.

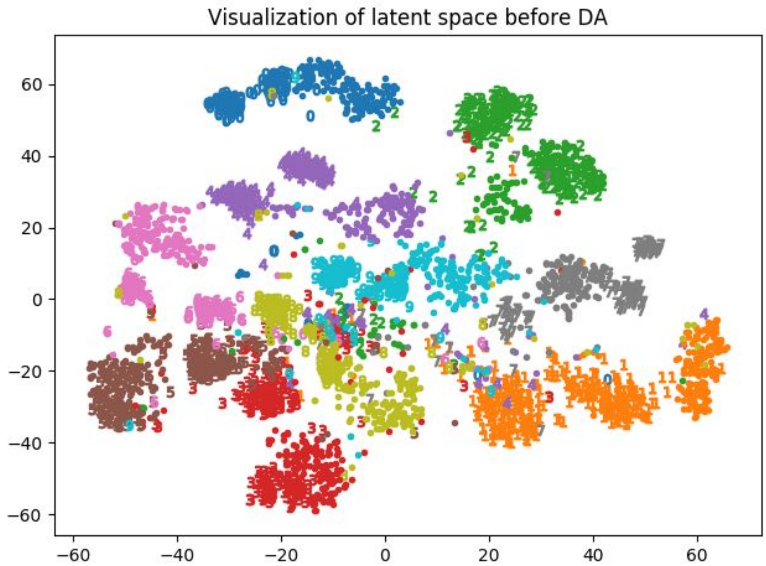
Experiments	β_{Sep}	β_C	β_P	λ_S	λ_T	PL_{Thresh}
SVHN → MNIST	1.5	1	4	0.5	0.8	0.999

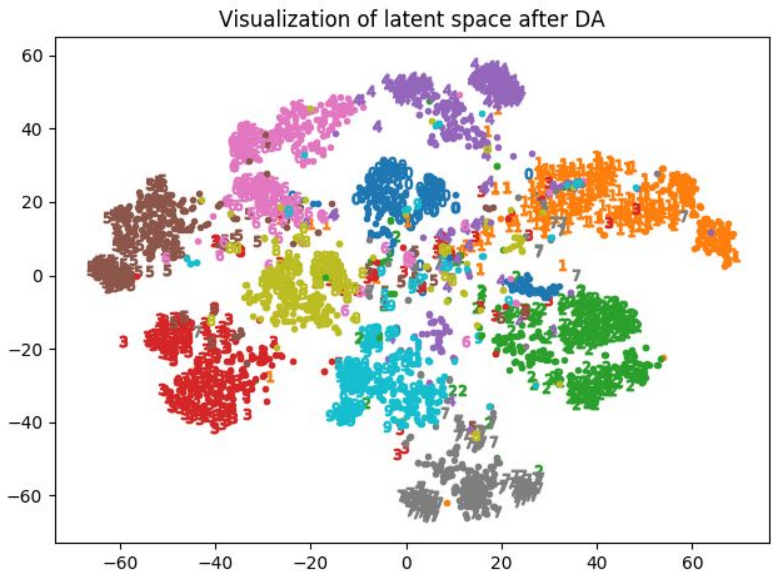
(pic. Hyperparameters form original paper)

7. Latent spaces

Used t-SNE. As we can see model is domain-informative and category-informative. Source colors are marked with their labels, target color are marked with dots.

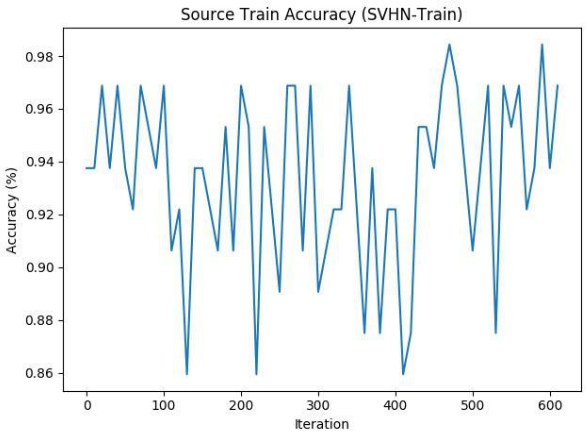


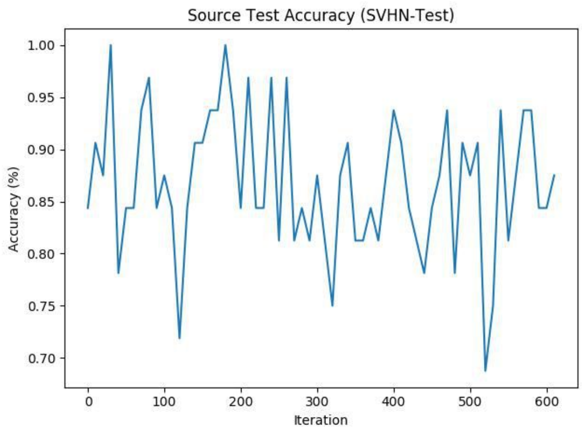


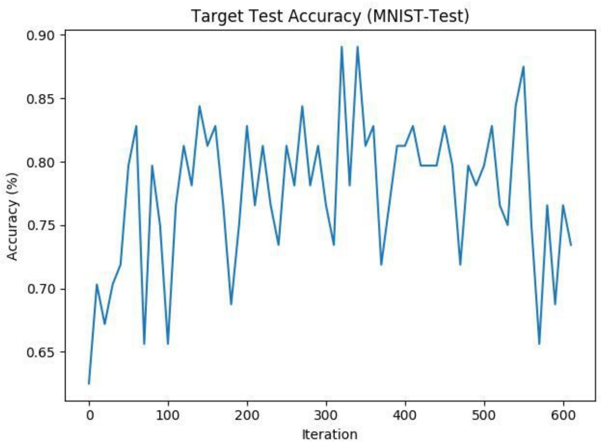


8. Accuracies

svhn-train	95.1
svhn-test	89.1
mnist-test	82.4







9. Personal thoughts

As I think GAN is really good Deep Learning approach, especially in domain adaptation tasks. They combine simplicity of idea (two contesting models) and hardness on implementation.

10. References

- Triplet Loss Network for Unsupervised Domain Adaptation (paper, code)
- https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html