EVALUATING *BiGAN* MODEL ON THE *MNIST* DATASET

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work we report our insights and results obtained while implementing and evaluating *Bidirectional Generative Adversarial Network (BiGAN)* model on the *MNIST* dataset. We try to reproduce results obtained in Donahue et al. (2016). We present qualitative and quantitative results of *BiGAN* model in terms of generative and reconstruction capabilities. Further, we extend the model by incorporating information about the classes with use of an additional triplet loss.

1 INTRODUCTION

The *Generative Adversarial Network (GAN)* learns a generator mapping samples from an arbitrary latent distribution to a data [Goodfellow et al. (2014)]. *Bidirectional Generative Adversarial Network (BiGAN)* learns an additional encoder, which inverts the generator [Donahue et al. (2016)]. This approach to unsupervised feature learning makes no assumptions about the underlying structure of the data. *Triplet learning* [Hoffer & Ailon (2014)] is a way to incorporate semi-supervised data into training so as to increase the quality of embeddings produced by encoder.

1.1 DATASET

We evaluate the model on permutation-invariant *MNIST*. In this setting, each image must be treated as an unstructured vector. We reshape each 28×28 image to be a $784D$ vector and scale it to $[0, 1]$.

2 *BiGAN*

The first goal of this work is to implement the *BiGAN* model and evaluate it on the *MNIST* dataset. We implement the *BiGAN* according to the method described in Donahue et al. (2016).
The code for all our models and evaluation is publicly available[1].

2.1 ARCHITECTURE

Generator, encoder and discriminator each consist of three dense layers. Each of the first two hidden layers consist of 1024 units. Batch normalization is applied to the second hidden layer.
Generator uses *ReLU* as an activation function. Encoder and discriminator each use *Leaky ReLU* with leak of 0.2. We use $50D$ continuous uniform distribution $[U(-1, 1)]^{50}$ as a latent distribution.

2.2 TRAINING

Weights are initialized with zero-mean normal distribution and standard deviation $stddev = 0.02$. Models are trained for 400 epochs using randomly selected batches of size 128. For optimization, we use *Adam* algorithm with initial learning rate $\alpha = 2 \times 10^{-4}$, $\beta 1 = 0.5$ and $\beta 1 = 0.999$. The learning rate decreases exponentially from the middle of the training, ending with $\alpha = 2 \times 10^{-6}$. For regularization, we apply l_2 weight decay of 2.5×10^{-5} to the weight matrices of dense layers. The training takes about 3 hours.

[1] `https://github.com/staffik/bigan_mnist`

2.3 EXPERIMENTS

We evaluate the *BiGAN* qualitatively by looking at samples produced by generator and reconstructions done by encoder.

Figure 1: *G(z)* - generated samples, *x* - real data (test set), *G(E(x))* - reconstructions done by encoder

Generated samples look like those from the dataset. In most cases, reconstructions are almost exact copies of the originals. The results are comparable to those from *Figure 2* of Donahue et al. (2016).

We visualize the latent representation z with the *T-SNE* plot. *T-SNE* algorithm ran for 250 epochs using Euclidean metric. Different colors correspond to different classes.

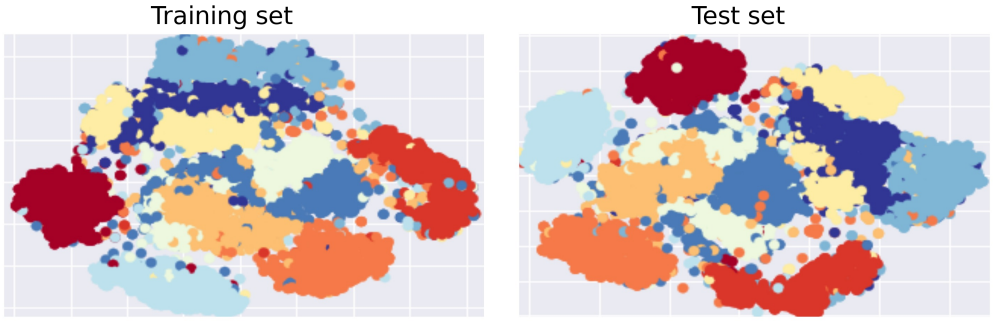

Figure 2: Clusterization results for latent representations

Results of clusterization are similar for samples from both training and test sets. A few classes seem to be separated from the others, but generally the classes are not well separated.

We evaluate the *BiGAN* quantitatively by calculating the *One Nearest Neighbors (1NN)* classification accuracy (%) on the test set in the feature space learned by *BiGAN*. The accuracy of our model is 93.75%, which is almost 4% worse than accuracies reported in *Table 1* of Donahue et al. (2016).

We noticed how the accuracy changes, when trying to use different hyperparameters than those proposed in Donahue et al. (2016):

- Use of 100-dimensional latent space (instead of 50) results in 0.5% better accuracy.
- Scaling dataset to $[-1, 1]$ (instead of $[0, 1]$) results in 1% worse accuracy.
- Not using l_2 weight regularization results in 1% worse accuracy.
- Using *Leaky ReLU* in generator (instead of *ReLU*) results in 1% better accuracy.

3 EXTENDED *BiGAN*

The second goal of this work is to extend the *BiGAN* model by incorporating additional information about the classes. We do that by using triplet learning procedure, similarly to how it is done in Zamorski & Zieba (2018).

Architecture, hyperparameters and learning procedure are the same as for simple *BiGAN* reported in *Section 2*, besides that:

- As a starting point we use weights of pretrained *BiGAN* reported in previous section.

- We use an additional triplet loss.
- Number of epochs is only 50.
- Learning rate is constant and equals 10^{-6}.

Also in Zamorski & Zieba (2018) learning starts with triplet loss disabled. Training the network from scratch behaved badly (discriminator had almost 100% accuracy all the time through the training).

For each digit, we randomly select 100 positive pairs and 100 negative examples. This procedure results in 10^5 triplets, which is not much, given that the size of the training set is 10^6.

3.1 EXPERIMENTS

We compare the results obtained by our model to the results obtained by simple *BiGAN* reported in previous section:

Figure 3: *G(z)* - generated samples, *x* - real data (test set), *G(E(x))* - reconstructions done by encoder

Qualitative results presented in *Figure 3* look similar to those presented in *Figure 1*, so this *BiGAN* extension keeps the generative and reconstruction capabilities.

How triplet learning procedure influences the quality of latent representation z? We investigate that by visualizing the latent representation z with the *T-SNE* plot as before:

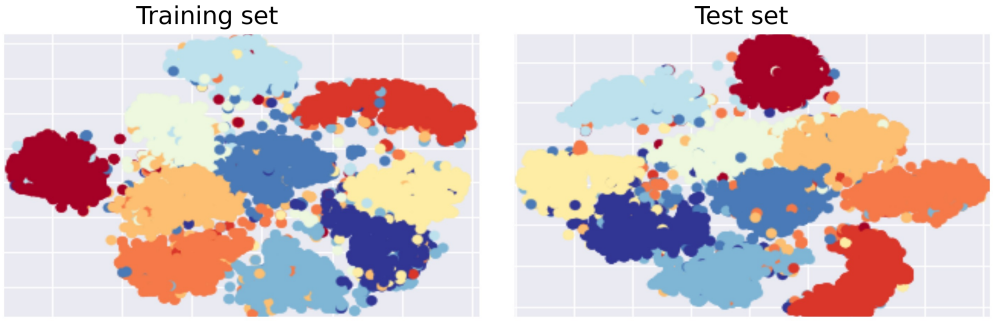

Figure 4: Clusterization results for embeddings given by encoder

Now, all classes are well separated in contrast to the situation from *Figure 2* for simple *BiGAN*. It means that triplet learning procedure succeeded in increasing the quality of embeddings. We check it also quantitatively by calculating the *One Nearest Neighbors (1NN)* classification accuracy, which is now 95.43%. It is almost 2% better than for simple *BiGAN* reported in *Section 2*.

We noticed that more triplets lead to better results, so there is potential in generating more triplets. Also, hard triplets mining could probably improve results (see Zieba & Wang (2017) for details).

Currently, there is a bug in Keras (which is our chosen framework), such that there is not possibility to use *BatchNormalization* layer in a triplet network[2]. We managed to do a simple workaround for that, but unfortunately, it hinders learning, so probably final results could be better.

[2] https://github.com/keras-team/keras/issues/11927

4 CONCLUSION

Extending the *BiGAN* with an additional triplet loss seems to be an easy way of incorporating a semi-supervised data into the training. This approach has advantages over ideas from conditional *BiGAN* (see Jaiswal et al. (2017) for details), such that it can be used to improve pretrained network (without modifying its internal structure) and it can be done in a semi-supervised way.

REFERENCES

J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. arXiv:1605.09782, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.

Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. 12 2014. doi: 10.1007/978-3-319-24261-3_7.

Ayush Jaiswal, Wael AbdAlmageed, Yue Wu, and Premkumar Natarajan. Bidirectional conditional generative adversarial networks. 11 2017.

Maciej Zamorski and Maciej Zieba. Semi-supervised learning with bidirectional gans. 11 2018.

Maciej Zieba and Lei Wang. Training triplet networks with gan. 04 2017.