

# Semi-supervised learning with Bidirectional GANs

Maciej Zamorski and Maciej Zięba

<sup>1</sup> Faculty of Computer Science and Management  
Wrocław University of Science and Technology  
maciej.zamorski@pwr.edu.pl, maciej.zieba@pwr.edu.pl  
<sup>2</sup> Tooploox Ltd.

**Abstract.** In this work we introduce a novel approach to train Bidirectional Generative Adversarial Model (BiGAN) in a semi-supervised manner. The presented method utilizes triplet loss function as an additional component of the objective function used to train discriminative data representation in the latent space of the BiGAN model. This representation can be further used as a seed for generating artificial images, but also as a good feature embedding for classification and image retrieval tasks. We evaluate the quality of the proposed method in the two mentioned challenging tasks using two benchmark datasets: CIFAR10 and SVHN.

**Keywords:** Generative models · Triplet learning · Generative Adversarial Networks · Image retrieval

## 1 Introduction

One of the most common and important tasks of machine learning is building generative models, that can capture and learn a wide variety of data distributions. Recent developments in generative modeling concentrate around two major areas of research: variational autoencoders (VAE) [5] that aim at capturing latent representations of data, while simultaneously keeping it restricted to known distribution (e.g., normal distribution) and generative adversarial networks (GANs) [3,8] with grounds in game theory, having strong emphasis on creating realistic samples from underlying distributions.

These kind of models are not only known from generating data from the distribution represented by data examples but are also used to train informative and discriminative feature embeddings. It can be obtained either only with unsupervised data by using good discriminative properties of the GAN's discriminator achieved during adversarial training [8,12] or using some subset of labeled data and incorporating semi-supervised mechanisms during training the generative model [9,13].

In this work we concentrate on obtaining better feature representation for image data using semi-supervised learning with a model based on Bidirectional

Generative Adversarial Networks (BiGANs) [1] / Adversarially Learned Inference (ALI) [2]. In order to incorporate semi-supervised data into training procedure, we propose to enrich the primary training objective with an additional triplet loss term [4] that operates on the labeled examples.

Our approach is inspired by the work [13] where the triplet loss was used to increase the quality of features representation in the discriminator. Contrary to this approach, we make use of an additional model in BiGAN architecture - encoder and aim at increasing the quality of feature representation in the coding space, that is further used by a generator to create artificial samples. Practically, it means that the feature representation can be used not only for classification and retrieval purposes but also for generating artificial images similar to existing.

The contribution of the paper is twofold. We introduce a new GAN training procedure for learning latent representations that extends the models presented in [1,2] and inspired by [13] for semi-supervised learning. We show that Triplet BiGAN will result in superior scores in classification and image retrieval tasks.

This work is organized as follows. In Sec. 2 we present the basic concepts related to GAN models and triplet learning. In Sec. 3 we describe our approach - Triplet BiGAN model. In Sec. 4 we provide the results obtained by Triplet BiGAN on two challenging tasks: image classification and image retrieval. This work is summarized in Sec. 5.

## 2 Related works

### 2.1 Generative Adversarial Networks

Since their inception, Generative Adversarial Networks (GANs) [3] have become one of the most popular models in a field of generative computer vision. Their main advantages come from their straightforward architecture and ability to produce state-of-the-art results. Studies performed in recent years propose many performance, stability and usage improvements to the original version, with Deep Convolutional GAN (DCGAN) [8] and Improved GAN [9] being used most often as architectural baselines in pure image generation learning tasks.

The main idea of GAN is based on game theory and assumes training of two competing networks: generator  $G(\mathbf{z})$  and discriminator  $D(\mathbf{x})$ . The goal of GANs is to train generator  $G$  to sample from the data distribution  $p_{data}(\mathbf{x})$  by transforming the vector of noise  $\mathbf{z}$  to the data space. The discriminator  $D$  is trained to distinguish the samples generated by  $G$  from the samples from  $p_{data}(\mathbf{x})$ . The training problem formulation is as follows:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z})))] \quad (1)$$

where:  $p_{data}$  - true data distribution,  $p_z$  - prior to the data space.

The model is usually trained with the gradient-based approaches by taking minibatch of fake images generated by transforming random vectors sampled from  $p_z(\mathbf{z})$  via the generator and minibatch of data samples from  $p_{data}(\mathbf{x})$ . They are used to maximize  $V(D, G)$  with respect to parameters of  $D$  by assuming a

constant  $G$ , and then minimizing  $V(D, G)$  with respect to parameters of  $G$  by assuming a constant  $D$ .

## 2.2 Bidirectional Generative Adversarial Networks

BiGAN model, presented in [1,2] extends the original GAN model by an additional encoder module  $E(\mathbf{x})$ , that maps the examples from data space  $\mathbf{x}$  to the latent space  $\mathbf{z}$ . By incorporating the encoder into the GAN architecture, we can code examples in the same space that is used as a seed for generating artificial samples.

The objective function that is used to train the BiGAN model can be defined in the following manner:

$$V(G, D, E) = \mathbb{E}_{x \sim p_x} [\log D(x, E(x))] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z), z))]. \quad (2)$$

The adversarial paradigm applied to train the model BiGAN is analogous as for GAN model. The goal of training is to solve the min-max problem stated below:

$$\min_{G, E} \max_D V(G, D, E). \quad (3)$$

Practically, the model is trained in alternating procedure, where the parameters of discriminator  $D$  are updated by optimizing the following loss function:

$$L_D = \mathbb{E}_{x \sim p_x} [\log D(x, E(x))] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z), z))], \quad (4)$$

and the parameters of generator  $G$  and encoder  $E$  are jointly trained by optimizing the following loss:

$$L_{EG} = \mathbb{E}_{z \sim p_z} [\log D(G(z), z)] + \mathbb{E}_{x \sim p_x} [\log(1 - D(x, E(x)))]. \quad (5)$$

Since (as authors showed in [1]) the encoder, in order to be an optimal one, learns to invert the examples from true data distribution, the same loss can be applied to the encoder and the generator parameters.

Experiments show that the encoder despite learning in a pure unsupervised way was able to embed meaningful features, which later show during reconstruction. The inclusion of additional module raises a question about the quality of this feature representation for classification and image retrieval tasks. The approach of combining objectives seems promising, as the encoder module is explicitly trained for feature embedding, as opposed to the discriminator, which main task is to categorize samples into real and fake.

## 2.3 Triplet Networks

Triplet networks [4] are among the most used methods for deep metric learning [6,10,11]. Triplet networks consist of three instances of the same neural network, that share parameters among themselves. During training, the triplet model  $T$

receives three examples from the training data: the reference sample  $x^q$ , the positive sample (the sample that is in some way similar to the reference sample, f.e. it belongs to the same class)  $x^+$  and the negative sample (that is dissimilar to the reference sample)  $x^-$ . The goal is to train the triplet network  $T$  in such a way, that the distance  $d^-$  between encoded query example  $T(x^q)$  to the encoded negative example  $T(x^-)$  is greater than the distance  $d^+$  from the encoded query example  $T(x^q)$  to the encoded positive example  $T(x^+)$ . In general case, this distances are computed as a L2-norm between feature vectors, i.e. :  $d^- = \|T(x) - T(x^-)\|_2$  and  $d^+ = \|T(x) - T(x^+)\|_2$ .

During the training the triplet model makes use of the probability  $p_T$  that the distance of the query example to the negative example is greater than its distance to the positive one which can be defined in the following way:

$$p_T = \frac{\exp(d^-)}{\exp(d^+) + \exp(d^-)} \quad (6)$$

We formulate the objective function for a single triplet  $(x, x^+, x^-)$  in a following manner [13]:

$$L_T = -\log(p_{T(x, x^+, x^-)}) \quad (7)$$

The parameters of the model  $T$  are updated according to the gradient-based approach that is used to optimize the objective function  $L_T$  by utilizing the minibatches of triplets  $(x, x^+, x^-)$  selected from data. Usually, the procedure of triplet selection is performed randomly (assuming that  $x, x^+$  are closer than  $x, x^-$ ) but there are some other approaches that speed-up the training process. The most popular is to construct the triplets for training taking under consideration the hardest negative samples  $x^-$ , which are the closest to the currently selected reference sample  $x$ .

### 3 Triplet BiGANs

In this work we introduce Triplet BiGAN model that combines the benefits of using BiGAN in terms of learning interesting representations in latent space and the superior power of the triplet model that trains well using supervised data. The core idea of our approach is to incorporate the encoder model of BiGAN to act as triplet network on the labeled part of training data (see fig. 1).

In terms of training the Triplet BiGAN we simply modify the  $L_{EG}$  (see eq. (5)) criterion by incorporating an additional triplet term:

$$L_{TEG} = L_{EG} + \lambda \cdot \mathbb{E}_{(x, x^+, x^-) \sim p_{triplet}} [-\log(p_{T(x, x^+, x^-)})], \quad (8)$$

where  $p_{T(x, x^+, x^-)}$  is triplet loss defined by eq. (7),  $\lambda$  is a hyperparameter that represents the impact of the triplet loss on the global criterion and  $p_{triplet}$  is the distribution that generates triplets, where  $x, x^+$  are from the same class and  $x, x^-$  are from different classes.

Triplet BiGAN model is dedicated to solving semi-supervised problems, where only some portion of labeled data is available. Practically, we do not have access

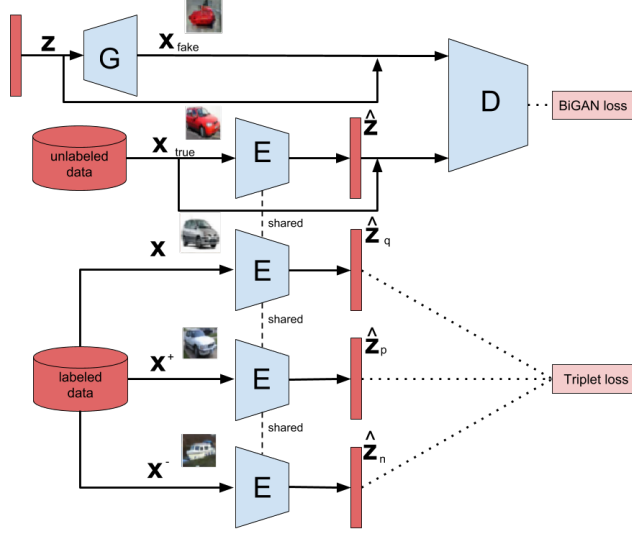


Fig. 1: The scheme presents the architecture of Triplet BiGAN model. We modify the BiGAN architecture by incorporating the encoding model in additional triplet task. We use the labeled part of the training data to construct triplets  $(x, x^+, x^-)$ , where we assume that  $x$  and  $x^+$  are from the same class and  $x$  and  $x^-$  are from different classes. Each of component of the triplet  $(x, x^+, x^-)$  is passed thru the encoding module to obtain the corresponding coding vectors  $(z, z^+, z^-)$  that are further used to construct the triplet loss. The remaining part of the model operates on the unsupervised data as in basic BiGAN model.

to  $p_{\text{triplet}}$ , therefore we are sampling the triplets  $(x, x^+, x^-)$  from some portion of an available labeled dataset,  $X_q$ .

The training procedure for the model is described in alg. 1. We assume that  $G$ ,  $E$ ,  $D$  are neural networks, that are described by the parameters  $\theta_G$ ,  $\theta_E$ ,  $\theta_D$  respectively. For the training procedure we assume, that we have access to unsupervised data  $X$  and some portion of supervised data  $X_q$ . For each training iteration, we randomly sample noise vector  $z$  from the normal distribution, pass it thru generator  $G$  to obtain the fake sample  $\hat{x}$ . We select  $x$  from unlabeled data  $X$  and triplet  $(x, x_+, x_-)$  from labeled data  $X_q$ . Using encoder  $E$  we receive the coding vector  $z$  corresponding to the sample  $x$ . Next, we update the parameters  $\theta_D$  of discriminator  $D$  by optimizing the criterion  $L_D$ . During the same iteration we update the parameters of generator  $G$  and encoder  $E$  by optimizing  $L_{TEG} = L_{EG} + L_T$ . The procedure is repeated until convergence.

In practical implementation, we make use stochastic gradient optimization techniques and perform gradient updates using ADAM method. We also initialize parameters of the Triplet BiGAN by training simple BiGAN without triplet term for given number of epochs without triplet term ( $\lambda = 0$ ).

The motivation behind this approach is to increase the discriminative capabilities of the codes obtained from latent space for BiGAN model using some

```

1  $\theta_G, \theta_E, \theta_D \leftarrow$  initialize network parameters
2 while converging do
3   for  $(x, x_q)$  in  $(X, X_q)$  do
4      $z \sim \mathcal{N}(0, I)$  // Sample feature data from normal distribution
5      $\hat{x} \leftarrow G(z)$  // Generate images using sampled data
6      $\hat{z} \leftarrow E(x)$  // Encode images from a minibatch
7      $x_+ \leftarrow$  random  $x'_q \in X_q$ , same class as  $x_q$ 
8      $x_- \leftarrow$  random  $x'_q \in X_q$ , different class than  $x_q$ 
9      $d_+ \leftarrow \|E(x_q) - E(x_+)\|_2$ 
10     $d_- \leftarrow \|E(x_q) - E(x_-)\|_2$ 
11     $L_D \leftarrow \log D(x, \hat{z}) + \log(1 - D(\hat{x}, z))$  // GAN loss for  $D$ 
12     $L_{EG} \leftarrow \log D(\hat{x}, z) + \log(1 - D(x, \hat{z}))$  // GAN loss for  $G, E$ 
13     $L_T \leftarrow -\log \frac{\exp(d_-)}{\exp(d_+) + \exp(d_-)}$  // Triplet loss for  $E$ 
14     $\theta_D \leftarrow \theta_D - \nabla_{\theta_D} L_D$ 
15     $\theta_G \leftarrow \theta_G - \nabla_{\theta_G} L_{EG}$ 
16     $\theta_E \leftarrow \theta_E - \nabla_{\theta_E} (L_{EG} + L_T)$ 
17  end
18 end

```

**Algorithm 1:** Training procedure for Triplet BiGAN.

portion of labeled examples involved in triplet training. As a result, we obtain the encoding model  $E$ , that is not only capable of coding the data examples for further reconstruction but also can be used as good quality feature embedding for the tasks like image classification or retrieval.

## 4 Experiments

The goal of the experiments is to evaluate the discriminative properties of the encoder in two challenging tasks: image retrieval and classification. We compare the results with the two reference approaches: triplet network trained only with supervised data and simple BiGAN model, where the latent representation of encoder is used for evaluation.

**Datasets** The model was trained on two datasets: Street View House Numbers (SVHN) and CIFAR10. In each dataset, 50 last examples of each class were used as a validation set and were not used for training the models. During training only selected portion of training set have assigned labels. The next subsection presents results obtained when using only 100, 200, 300, 400 or 500 labeled examples per class. For testing purposes, we trained classifier only on the images from the training split, that were given a label for triplet training.

**Metrics** Retrieval evaluation was done with accuracy and mean average precision (mAP). For classification, 9-nearest neighbors classifier was used with weighted by the distance-based importance of neighbors. Mean average precision

was calculated at length of encoded data. Cluster visualization was performed by applying t-SNE[7] with Euclidean metric, perplexity 30 and Barnes-Hut approximation for 1000 iterations.

**Architecture** The architectures of discriminator, encoder, and generator were as presented in [2].

The encoder network  $E$  is a 7-layer convolutional neural network, that learns the mapping from image space  $X$  to feature space  $z$ . After each convolutional layer (excluding the last), a batch normalization is performed, and the output is passed through a leaky relu activation function. After penultimate convolutional block (meaning convolutional layer with normalization and activation function) a reparametrization trick [5] is performed.

The generator network  $G$  is a neural network with seven convolution transposition layers. After each layer (except the last) a batch normalization is performed, and the output is passed through a leaky relu activation function. After the last convolution-transposition layer, we squash the features to  $(0,1)$  range with the sigmoid function.

The discriminator part  $D$  consists of three neural networks –  $D_x$  discriminates in the image space,  $D_z$  discriminates in the encoding space. Both of them map their inputs into a discriminative latent space and each of them returns a same-size vector. Third network  $D_{xz}$  takes concatenation of said vectors as an input and returns a decision, whether an input tuple (image, encoding) comes from encoding or generative part of the Triplet BiGAN network. Image discriminator  $D_x$  is made of five convolution layers with Leaky Relu nonlinearity after each of them. Encoding discriminator is represented as two convolution layers with Leaky Relu nonlinearity after each of them and Joint discriminator  $D_{xz}$  is another three convolutional layers with Leaky Relu between them and the sigmoid nonlinearity at the end.

#### 4.1 Results

**Classification** For assessing classification accuracy quality, the experiments were done to test the influence of feature vector size and images per class taken for semi-supervised learning. For each of the model, the experiments were done, when the feature vector consisted of either 16, 32, 64 or 128 (256 for SVHN) variables and 500 labeled images per class were taken. On the other hand, using feature vector size of 64, the experiments measured the impact of a number of labeled examples available during training, with possible values being 100, 200, 400 and 500 (only for Cifar10). The experiments were conducted on Cifar10 and SVHN datasets.

**Retrieval** For assessing image retrieval quality, the experiments were made to test an influence of feature vector size and images per class taken for semi-supervised learning. For each sample in the testing data, an algorithm sorts the images from the training dataset from closest to most further. Distances are calculated basing on Euclidean distances between images' feature vectors to

Table 1: Classification results on CIFAR10 dataset for different sizes of encoder feature vector using 500 labeled examples per class in the training set.  $m$  - vector size. Only labeled samples from training set used.

Model	<b>m=16</b>	<b>m=32</b>	<b>m=64</b>	<b>m=128</b>
Triplet	44.42	45.56	52.32	46.15
BiGAN	41.30	43.81	48.50	49.13
Triplet BiGAN	61.08	62.40	63.14	53.92

Table 2: Classification results on SVHN dataset for different sizes of encoder feature vector using 200 labeled examples per class in the training set.  $m$  - vector size. Only labeled samples from training set used.

Model	<b>m=16</b>	<b>m=32</b>	<b>m=64</b>	<b>m=256</b>
Triplet	66.12	69.96	71.11	71.54
BiGAN	44.65	53.80	62.35	17.48
Triplet BiGAN	71.43	75.65	79.12	78.86

Table 3: Classification results on CIFAR10 dataset for different portions of labeled samples per class and feature vector size  $m = 64$ .  $n$  - number of labeled samples per class. Only labeled samples from training set used.

Model	<b>n=100</b>	<b>n=200</b>	<b>n=400</b>	<b>n=500</b>
Triplet	28.80	34.81	40.12	52.32
BiGAN	42.73	45.06	47.67	41.30
Triplet BiGAN	53.15	52.02	57.45	63.14

Table 4: Classification results on SVHN dataset for different portions of labeled samples per class and feature vector size  $m = 64$ .  $m$  - vector size. Only labeled samples from training set used.

Model	<b>n=100</b>	<b>n=200</b>	<b>n=400</b>
Triplet	66.48	71.11	73.76
BiGAN	61.16	62.35	58.94
Triplet BiGAN	72.40	79.12	82.21

check if samples that are close to each other in data space (images belong to the same class) are close to each other in feature space (their representation vectors are similar). In ideal type situation ( $mAP = 1$ ), all of the relevant training images would be put first and only then training images that belong to the same class. With 10 classes in each dataset  $mAP = 0.1$  may be considered random ordering, as it roughly means that on average, only every tenth image was of the same class as the test image.

Results presented in tables below indicate the increased average precision of image retrieval when using Triplet BiGAN method as opposed to using only labeled examples by 0.05 - 0.15 in all, but one experiment.

Figure 2 presents a visualization of the closest images from the restricted training set (used only 500 examples per class from the original training set,



Table 5: Image retrieval results on CIFAR10 dataset for different sizes of encoder feature vector using 500 labeled examples per class in the training set.  $m$  - vector size. Only labeled samples from training set used.

Model	<b>m=16</b>	<b>m=32</b>	<b>m=64</b>	<b>m=128</b>
Triplet	0.4993	0.5134	0.5235	0.5197
BiGAN	0.1458	0.1433	0.1634	0.1620
Triplet BiGAN	0.6292	0.6457	0.6528	0.3748

Table 6: Image retrieval results on SVHN dataset for different sizes of encoder feature vector using 200 labeled examples per class in the training set.  $m$  - vector size. Only labeled samples from training set used.

Model	<b>m=16</b>	<b>m=32</b>	<b>m=64</b>	<b>m=256</b>
Triplet	0.6855	0.7224	0.7474	0.6989
BiGAN	0.1492	0.1582	0.1748	0.1633
Triplet BiGAN	0.7201	0.7633	0.8002	0.7931

Table 7: Image retrieval results on CIFAR10 dataset for different portions of labeled samples per class and feature vector size  $m = 64$ .  $n$  - number of labeled samples per class. Only labeled samples from training set used.

Model	<b>n=100</b>	<b>n=200</b>	<b>n=400</b>	<b>n=500</b>
Triplet	0.3732	0.4027	0.4778	0.5235
BiGAN	0.1681	0.1666	0.1645	0.1634
Triplet BiGAN	0.5628	0.5487	0.5966	0.6528

Table 8: Image retrieval results on SVHN dataset for different portions of labeled samples per class and feature vector size  $m = 64$ .  $n$  - number of labeled samples per class. Only labeled samples from training set used.

Model	<b>n=100</b>	<b>n=200</b>	<b>n=400</b>
Triplet	0.6323	0.7474	0.7490
BiGAN	0.1737	0.1748	0.1655
Triplet BiGAN	0.7300	0.8002	0.8198

the same examples that were used in semi-supervised learning). The closeness of the image was decided between each image from the randomly chosen sample of 5 images from the test set and each image from the restricted training set. The distance between images was calculated by encoding each image to feature vector form and calculating Euclidean distance between selected test and training images.

As seen in the visualization, BiGAN model, despite the fact of being trained in a pure unsupervised way, can still embed similar images to similar vectors. However, the closest pictures tend to contain occasional errors, which is not the case with retrieval using triplet models, that tend to contain errors sparingly.

The notable example, showing better results of Triplet BiGAN in comparison to regular Triplet model is the 4th image from the selected test pictures (grey frog). Using 32 and 64 size features vectors Triplet BiGAN was able to retrieve other frog and toad images correctly. The same image caused problems for the original Triplet model, not to mention BiGAN. This shows that additional unsupervised learning of underlying data architecture is indeed beneficial to finding subtle differences in images and can improve the quality of feature embedding.

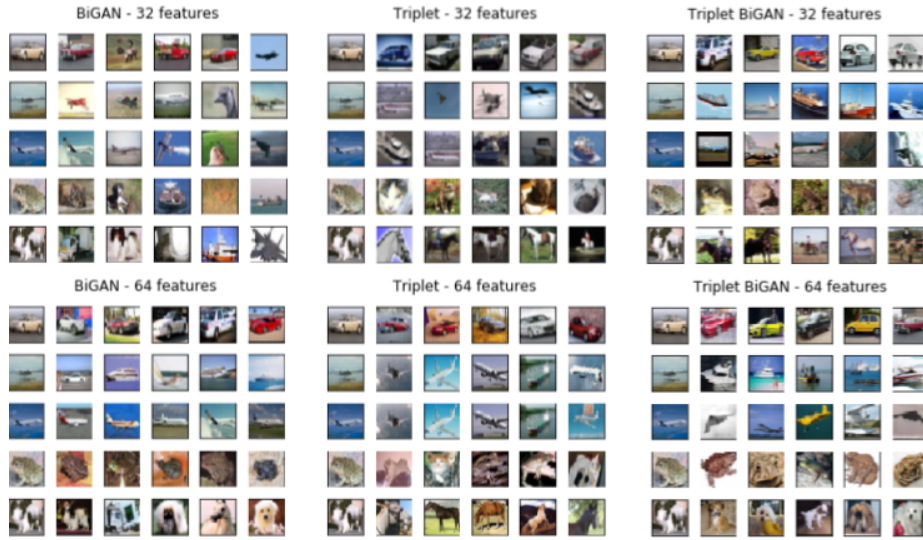


Fig. 2: Visualization presenting image retrieval results. Left-most column of each section contains five randomly chosen images from the Cifar10 test set. Columns from second-left to right-most present images from the restricted (500 examples per class) training set, from the closest to 5th-closest image.

**Clustering** Figures below show visualizations of embedding quality of tested models. Each sub-figure present embedding mapped to 2 dimensions using the t-SNE algorithm with one of the three models: Triplet, BiGAN and Triplet BiGAN on the training set and the test set. For Triplet and Triplet BiGAN models, 500 labeled samples per class were used. Two experiments were performed: one with the feature vector of size 32, and one with the feature vector of size 64, as mentioned in figure captions. T-SNE algorithm ran for 1000 epochs using perplexity of 30 and Euclidean metric for distance calculation. In the visualization each class was marked with own color, that was preserved through all sub-figures.

**Discussion** In classification and retrieval experiments Triplet BiGAN achieved worse results (tables 1, 3, 5, 7) than a Triplet GAN presented in [13]. However,

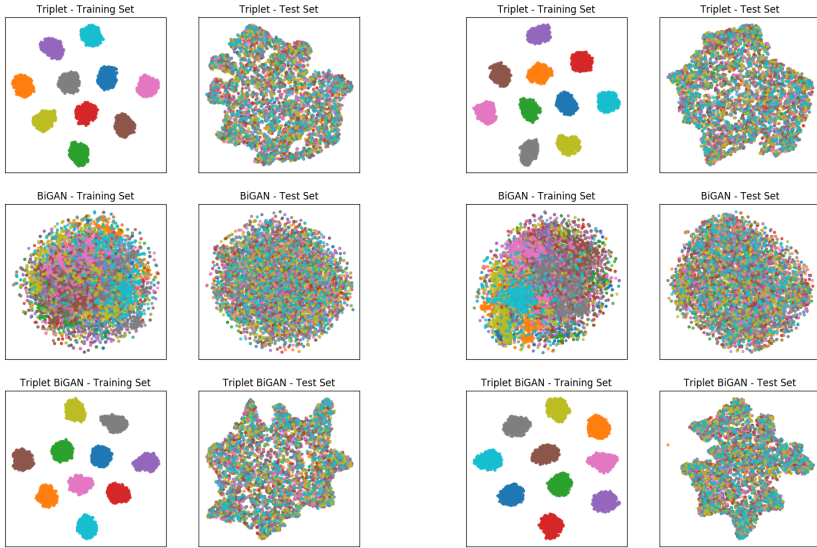


Fig. 3: a) Clusterization results for models with feature vector size  $m = 32$  and trained (Triplet and Triplet BiGAN) using 500 labeled examples per class.  
b) Clusterization results for models with feature vector size  $m = 64$  and trained (Triplet and Triplet BiGAN) using 500 labeled examples per class.

we believe than our proposed model has still several advantages in comparison to the reference method. Since in Triplet BiGAN, we perform metric learning on the Encoder (unlike in [13] where metric learning is done on the Discriminator features)

As visualizations suggest, both Triplet and Triplet BiGAN models did not have any problems with learning clusterization on training sets. The output from the t-SNE clearly shows separate group for each class of the samples for this models. This is not the case in BiGAN model. However, while BiGAN was trained without distance-based objective, one can still spot concentration of particular colors. This aligns with observations [2] that the encoder learns to embed meaningful features into the feature vector, including those, that are somewhat characteristic for specific classes.

Regarding test sets, Triplet and Triplet BiGAN did not generalize to create perfect separations of classes. The models learn to rather bind particular classes into small, homogeneous groups, which are not clearly visible on visualizations but are enough to perform classification using the nearest neighbor algorithm. In the case of BiGAN model the embedding features from the training set do not translate well to the test set, creating a somewhat chaotic collection of points, that is able to generate image retrieval results that are close to random.

## 5 Summary

This work presents the Triplet BiGAN model that uses joint optimizing criteria: to learn to generate and encode images and to be able to recognize the similarity of given objects. Experiments show that features extracted by an encoder, despite learning only on true data (in opposition to features learned by the discriminator, that learns on real and generated data), may be used as a basis of image classifier, retrieval, grouping or autoencoder model.

Also included in this work are descriptions of the models that were essential milestones in the field of generative models and distance learning models and an inspiration for creating the presented framework.

## References

1. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)
2. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016)
3. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
4. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International Workshop on Similarity-Based Pattern Recognition. pp. 84–92. Springer (2015)
5. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
6. Kumar, B., Carneiro, G., Reid, I., et al.: Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5385–5394 (2016)
7. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
8. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
9. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems. pp. 2234–2242 (2016)
10. Yao, T., Long, F., Mei, T., Rui, Y.: Deep semantic-preserving and ranking-based hashing for image retrieval. In: IJCAI. pp. 3931–3937 (2016)
11. Zhuang, B., Lin, G., Shen, C., Reid, I.: Fast training of triplet-based deep binary embedding networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5955–5964 (2016)
12. Zieba, M., Semberecki, P., El-Gaaly, T., Trzcinski, T.: Bingan: Learning compact binary descriptors with a regularized gan. arXiv preprint arXiv:1806.06778 (2018)
13. Zieba, M., Wang, L.: Training triplet networks with gan. arXiv preprint arXiv:1704.02227 (2017)