Text to Image Synthesis

Bibek Shrestha*, Christian Becker†, Pia Schmidt‡
Department of Computer Science, University of Applied Sciences Karlsruhe
Moltkestr. 30
76133 Karlsruhe, Germany
Email: *shbi1012, †bech1029, ‡scpi1018 @h-ka.de

***Abstract*—In this work we describe how to utilize generative adversarial networks (GANs) for a text to image synthesis. We begin with very simple FMNIST data generation, continued to apply GANs on more complex CIFAR10 color images and aimed to also apply it with very complex COCO data set. We were successful in creating FMNIST images but couldn't generated fully realistic CIFAR10 images. Nevertheless we decided to switch from plain labels input to embedding inputs of fastText and DistilBERT to explore their influence the first two data sets. Our experiments show that using a more complex embedding even for only ten class label input yield better results for both FMNIST and CIFAR10.**

***Index Terms*—generative adversarial networks, text to image, embedding**

I. Introduction

After learning about computer vision, natural language processing and generative models, we were curious, if it would be possible to merge those three topics and build a new model that could create an image from a given random sentence. Not surprisingly, it was possible, and [1] had already proposed a model that uses all three branches, i.e., CV, NLP und GAN (Generative Adversarial Network) to synthesize realistic image from given text description.

The colour, class, shape and other information that describes the images when given as embeddings into the model might possibly limit the shape and illumination variation of the output when trained for different classes. To test it, we built a network based on the architecture proposed by [1] and trained it with different data sets. In our model, the generator and discriminator are implemented with DCGAN and embeddings are created using fastText and DistilBERT. We tested the quality and diversity of the generated images along with the generalization capability of the model. FMNIST, CIFAR10 and partly MS COCO are the chosen data sets as they cover a wide range of classes like clothes, automobiles, animals etc. We first trained our model on FMNIST and CIFAR10 data sets without text embeddings and after successful training we trained the models again with text embeddings. The text embeddings were just the labels of the respective classes and a variety of sentences describing the class labels.

With the images generated from FMNIST and CIFAR10, we will be able to answer the following significant questions:

1) Do the embeddings have any influence on result?
2) Does it make difference, if only labels are embedded instead of text?
3) Which hyperparameters are important for stable training?

II. Related Works and Backgrounds

The images are created using the model which is based on Generative Adversarial Networks proposed by [2]. The framework consists of two models: a generative model G that creates images from a prior noise sample and a discriminative model D that classifies the images as real or fake. The images from the training data are regarded as real whereas the one generated by the generator are regarded as fake images. The two models G and D are trained simultaneously, where G trains to maximize the probability of D making mistakes and D trains to correctly predict the probability that an image is real. The trainings for both models are done with backpropagation algorithm.

In the beginning the generative and discriminative model in GAN were implemented with Multi Layer Perceptron (MLP). MLPS are not reliable when it comes to computer vision task and thus convolutional neural networks (CNN) is the most popular model for computer vision applications.

Therefore, [2] introduced a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that implemented G and D using CNN and proposed a set of constraints on the architecture topology to make DCGAN stable to train in most settings. The proposed architecture guidelines for stable DCGANs are as follows:

- Pooling layers shall be replaced with strided convolutions in discriminator and fractional-strided convolutions (a.k.a. transposed convolutions) in generator.
- Batch normalization shall be used in both the generator and discriminator.
- Fully connected hidden layers shall not be used for deeper architectures.
- ReLu activation shall be used in generator for all layers except for the output. Output uses Tanh activation.
- LeakyReLu activation shall be used in the discriminator for all layers.

[1] introduced a model which successfully generated plausible images of birds and flowers from detailed text descriptions. The architecture used in the model is called text-conditional convolutional GAN and it uses text encoding in both G and D as shown in Fig.1. As with all GANs, a noise prior is first sampled in the generator. Then the text description is encoded using text encoder. The resulting embedding is compressed

using fully connected layer to small dimension layer (128 in paper) and passed through leaky ReLu. The output is finally concatenated to the noise vector and given as input to the fractional-strided convolutional (deconvolutional) layer. A fake image is generated via last layer of a generator. Several layer of stride 2 convolution with batch normalization followed by leaky ReLu is performed in the discriminator. The dimension of the resulting embedding is reduced in a separate fully connected layer which then is followed by ReLu. Once the spatial dimension of discriminator reaches 4x4, the description embedding is spatially replicated and depth concatenation is performed. The final prediction from discriminator is calculated by performing 1x1 convolution followed by a ReLu and a 4x4 convolution. As mentioned in DCGAN paper, batch normalization is done on all convolutional layers. The networks are updated using backpropagation algorithm. The text-conditional convolutional GAN was originally trained on the CUB data set of bird images [3], Oxford-102 data set of flower images [4] and MS COCO validation set [5].

III. Methodology

In order to examine the added value of using embeddings to generate images with GANs, it must first be defined which inputs are possible. TABLE I shows these input options and their associated y for calculating the loss of the generated image in the discriminator: The input for an embedding can be the string representation of label like described in C2. Usually a numeric label represents a certain category, e.g. “House” or “Car”. Alternatively, texts can be used that describe the picture like “The girl throws a blue ball” (C3). Furthermore, it is possible to train a GAN without any embedding as a baseline (C1). Basically, images can be black and white images or RGB images. Our concept provides applicability to all of the above

TABLE I
Training combinations

ID	generator: input x	discriminator: criterion for generated image y
C1	noise	label
C2	noise, embedding of label	embedding of label
C3	noise, embedding of sentence	embedding of sentence

input possibilities. For this purpose, we define the following requirements for our concept:

1) Applicability to a different number of color channels
2) Support of different word and sentence embedding concepts and their dimensionality.

In our approach, we are guided by the concept of [1], which has already been presented in II. In the following, our adaption for the application is explained based on this.

A. Embedding-Module

Since we are working with a finitely large training and test set, embeddings are generated and saved by this module before the actual training process. The advantages of this decoupling become particularly clear for a subset of the input if we distinguish the following scenarios.

1) *Input option 1:* The input consists of a fixed sequence of words. This includes labels which often correspond to sequences of length 1.
2) *Input option 2:* The input consists of any sequence of words of any length.

The advantage of case 1 is the reusability of the resulting embeddings, which have to be generated only once. These are saved in a suitable data structure so that they can be loaded via their corresponding numeric label ID. Additionally, any pre-trained language model will not be reloaded per training instance.
If several reference sentences are available for an image, they are stored under the same ID and randomly returned to learn different semantic representations of the text.

B. Architecture

We orientate ourselves on the architecture for convolutional generative opposing network (DCGAN) [6] and the concept of [1] as shown in Fig.1. The adjustments for our concept are described below.
The generator receives two inputs: samples from a uniform noise distribution Z and the embedding $t(s)$ of the label or text s. In the first step we transform the noise vector z into an 7x7-dimensional tensor. $t(s)$ is reduced in dimension by a fully connected layer. Then, z and $t(s)$ are concatenated and fed into a feed-forward CNN.
The discriminator receives the generated image as well as the embedding. The image is passed in a CNN and flatted to a two dimensional tensor afterwards. As above, the embedding is compressed again. The embedding is replicated seven times and is then concatenated seven times with the dimensionally reduced image.
These steps leads to an increase in the output dimension. We use multiple combinations of linear layers and LeakyReLu to reduce the output dimension again until it corresponds to 1x1xC with C as the width of the color channel.
We use LeakyReLu for all fully connected layers except the output layers, which uses tanh in the generator and sigmoid activation in the discriminator. For the data sets used, we use an adaption on each of the architecture described with regard to the depth of the respective generator and discriminator networks.

IV. Experiments and Results

The models and embeddings introduced earlier are evaluated in experiments described in the following.

A. Evaluation setup

For the execution of the experiments a GTX 1080 has been utilized in a laboratory setup. The machine learning framework used for training is PyTorch. Further to log the experiment history including parameters, metrics, images and a gif of the final training MLFlow is used. For comparison multiple GAN’s have been trained using different data sets, models and hyperparamter. The data sets used for training are Fashion-MNIST (FMNIST) and CIFAR10. The FMNIST data

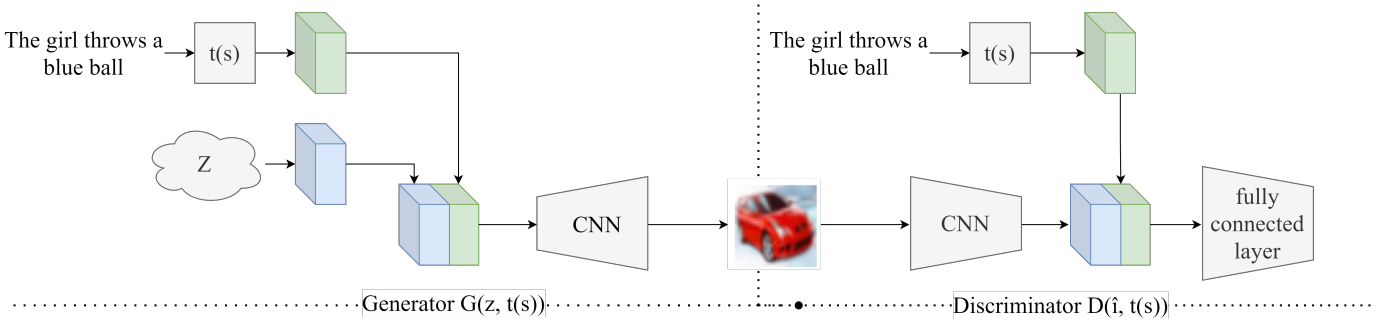

Fig. 1. text-conditional convolutional GAN architecture according to [1]

sets includes ten classes featuring simple pieces of clothing: t-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags and ankle boots. The latter data set feature color images of ten classes like birds, cats, airplanes. Thus, the range of data sets starts at simple 28x28 pixels grayscale images up to 32x32 pixel color images. All data sets have a history in research of being the default benchmark for deep learning use cases for a long time and thus are most suitable for our evaluation purposes. In order to be able to train on the data sets two models are necessary one for the one channel grayscale images of the MNIST data set and one for the three channel color images of the CIFAR10 data sets.

As embeddings models we use two different type of English models as comparison:

1) fastText (cc.en.300)
2) DistilBERT (distilbert-base-uncased[1])

Since we want to investigate the influence of each type of embedding input, we chose two as different types of embedding models as possible. While fastText [7], [8] is an N-Gram based word model trained with the CBOW method of Word2Vec [9], DistilBERT [10] uses Transformer architecture. DistilBERT is a lighter variant of BERT [11] that has been trained to reproduce the behavior of the teacher model using compression technique. fastText creates embeddings of dimension 300, while the resulting embeddings from DistilBERT are of dimension 768.

The training features multiple hyperparameters in order to fine tune training results. The learning rate of 0.0002, 50 epochs and a batch size of 32 are fixed value hyperparameters. Variable hyperparameters are listed in IV-A and include the number of steps to apply to the discriminator K, number of samples for training N and and latent vector size S. Additionally the discriminator can be trained on plain labels or on embeddings like fastText or DistilBERT embedding which is specified in the corresponding parameter.

B. Evaluation Methods

The evaluation of generated images is a wide and open topic in the area of GAN's [12]. In general there are two possible evaluation methods the manual evaluation by humans and

[1]Model by Hugging Face libary: https://huggingface.co/bert-base-uncased, visited on 26.07.2021

TABLE II
OVERVIEW OF HYPERPARAMETERS

parameters	value	description
learning rate	0.0002	Gradient update scale parameter
batch size	32	Number of samples taken into account in one batch
K	1	Number of steps to apply to the discriminator
N	32, 64, 128, 256	Number of samples for training
S	32, 64, 128, 256	Latent vector size
embedding	None, fastText or DistilBERT	Embedding used for label input

metrics calculated on the training data. One of the most widely used computed metrics is the inception score which tends to be very close to human assessment [13]. It's specifically designed to evaluate synthetic images and scores their quality and diversity. For calculation of the inception score the same pre trained model called inception_v3 is used. This allows to compare different GAN based on a scoring calculated on the very same baseline. Unfortunately the calculation of the inception score even on very simple CIFAR10 images is very computing intensive. Computing the inception score for just about 100 CIFAR10 images exceeds our memory of 20 GB RAM and 60 GB SWAP. Therefore we designed our own evaluation method to compare the results of multiple GANs. The *first sight scoring (FSS)* is applied on the syntactic samples generated by a GAN which are classified by one or many human experts. A human expert knows the original labels and sample images of it and therefore can intuitively classify the images without any hassle. The FSS than equals the number of items which can be identified and classified on first sight by an expert. The number of samples per GAN and the number of classifying experts are the same and thus the FSS can be used to compared the quality multiple GANs.

C. Evaluation

We have run multiple experiments in order to compare the usage of number of samples for training N and latent vector size S featuring the FMNIST and CIFAR10 data set. Additionally the experiments include the usage of the training

options from III: original labels for loss calculation (C1), fastText embeddings and DistilBERT embeddings (C2, C3).

D. Fashion MNIST experiments

The first experiment features FMNIST trained with default labels (C1), fastText embedding and DistilBERT embedding of the labels (C2). The results are depicted in Fig. 2. All three examples use the same number latent vector size of 256 and number of samples of 128. Both values have shown the best results for the FMNIST data set in our setup. The depiction show that embeddings do yield better results than training with labels only. The more advanced an embedding the better the results of the GAN. This can also be seen if taken a look at the FSS Diagramm 3 which has been evaluated by three experts over the 24 images per GAN as depicted in Fig. 2. The plain labels reach an average FSS of 5.66, where with fastText embedding an average of 7 are recognized and DistilBERT leads with 10 recognized examples. The average values also depict the opinion depicted for all the experts. Thus there is a correlation between the complexity of the input method and the quality of the images. The more complex the input the better the generated pictures are. This is the case because more complex embeddings hold more information about the input values.

Fig. 2. FMNIST traind with best hyper parameters in three settings: labels only without embeddings, fastText embedding and DistilBERT embedding.

E. CIFAR10 experiments

CIFAR10 is used for the second and third experiment. As in experiment one, experiment two uses the string representation of the labels as embeddings (C2). In contrast to experiment one, the training is here based on different configurations for the parameter S for fastText and DistilBERT embeddings. Fig. 7, 8 and 9, which can be found in the appendix, show in the first three columns an exemplary 24 images that were

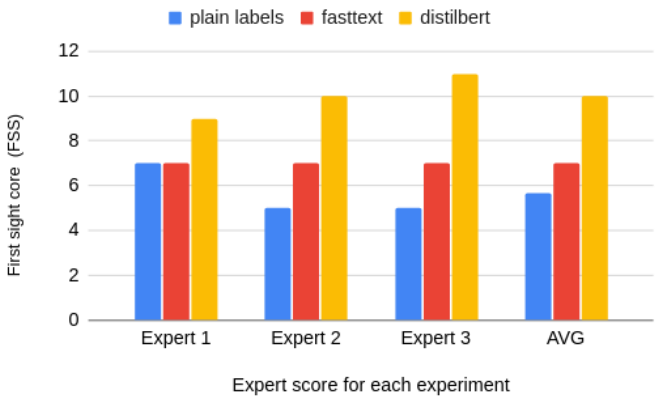

Fig. 3. FSS as chosen by three experts for three experiments of: labels only, fastText embedding and DistilBERT embedding.

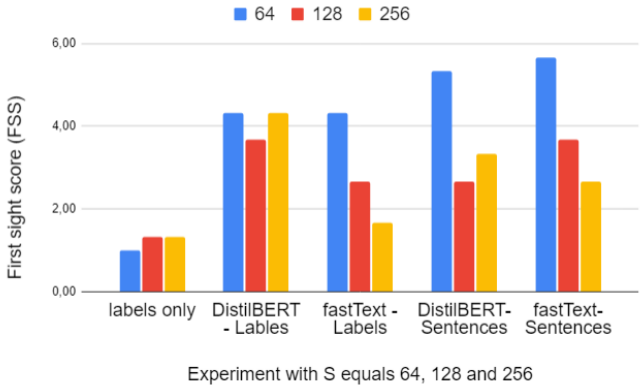

Fig. 4. FMNIST trained with best hyper parameters in three settings: labels only without embeddings, fastText embedding and DistilBERT embedding.

created per training iteration and evaluated using FSS. In particular, for $S = 128$, it can be seen that the colors of the images generated with embeddings are more diverse and vivid. In addition, the color areas are more clearly separated and color gradients appear more realistic. For example, cars, boats and birds can already be glimpsed here. An overview of the evaluation of the mean FSS is shown in Fig. 4. The two settings discussed here are shown from the left: “plain labels”, “fastText-labels” and “DistilBERT-labels”. S achieves better FSS with larger values for the embedding-free training, which nevertheless cannot stand comparison with the training including embeddings. On average over all configurations, 8.67 or 12.33 objects can be detected on the 24 images created with embeddings. For embedding-free images, the FSS is 3.67. DistilBERT embeddings generate equally good or better images than fastText embeddings for any, ascending sorted, value of S: FSS is 4.33, 3.67 and 4.33 for DistilBERT embeddings while it is 4.33, 2.67, 1.67 for fastText-Embeddings. If embeddings are used, $S = 64$ is superior or on par with $S = 256$ for DistilBERT. fastText embeddings perform worse with increasing latent features space.

Experiment three now uses whole sentences as embeddings (C3). Five sentences are available per class that describe the object depicted in semantically different ways, one of which is used randomly per epoch. Some of the records were taken from the COCO data record [5], which also provides five reference records for each image. COCO is divided into categories, which in eight cases are congruent with the classes of CIFAR10. For the classes "deer" and "frog" records were generated manually. Although this input corresponds to input option 2 from III, there is only a limited number of sentences in this setting. Therefore, the corresponding embeddings are also pre-generated. The decision to use the sentences in the restricted context of the CIFAR10 classes must be justified by the fact that this enables comparability with the other experiments. In addition, it was not possible for us to find a satisfactory architecture and configuration for the very diverse data set of the COCO data set in the limited time frame of this project. Examples of generated images are also shown in Fig. 7, 8 and 9. They can be found in the bottom two lines. The shapes of the generated objects look more realistic compared to the label embeddings. However, the colors are more reduced here. It can be assumed that a selection of 50 sentences, which among other things describe the color of the depicted object, limit the result here. Looking at the remaining FSS in Fig. 4, the sentence embedding leads to significantly better FSS only in one case, at $S = 64$. FSS is here 5.33 for DistilBERT and 5.67 for fastText. Comparing the embeddings, fastText is the better embedding in contrast to experiment two. Since the sentences used are no longer than ten words, it can be assumed that attention-based DistilBERT embedding still adds little value for coherent context here.

F. Convergence of CIFAR10 models

Since we chose an epoch number of 50 over all experiments, we want to compare the convergence behavior of the different CIFAR10 experiments at this point. For generator and discriminator a Nash equilibrium should be achieved in the convergence case. Fig. 5 shows the expected behaviour for the training without embeddings. However, for experiments C2 and C3, the loss was shown to converges at an early point and to diverge from that point for all training runs for the time being as depicted in Fig. 6. Our shown and discussed results are from the time of the strongest convergence. An investigation of this behavior and a possible convergence over a plurality of epochs provides scope for follow-up studies.

V. Conclusion

We were able to show that for the generation of images the use of embeddings achieves an improvement in the color and object representation. This is true for embeddings of labels as well as for the use of descriptive text. The use of text can achieve more interpretable results for humans using optimized hyperparameters such as the size of the latent features spaces of the noise vector. This was identified as a critical factor for successful training. A smaller value produced the better images for us. However, from the exemplary selection of embedding models used here, no clear recommendation can be made as to which is better. For label embeddings DistilBERT embeddings achieved higher FSS, for text embeddings fastText embeddings worked better.

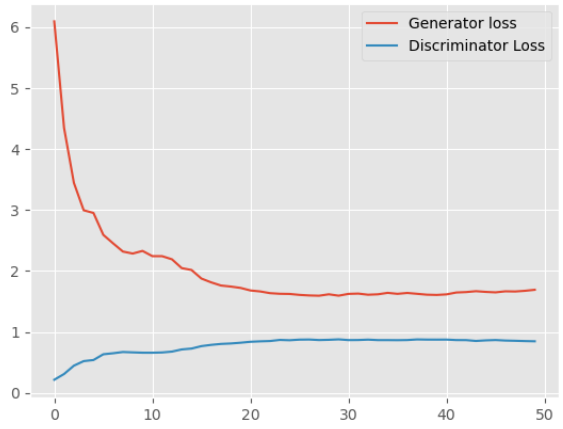

Fig. 5. Example for loss convergence of training without embeddings. The x-axis shows the epochs and the y-axis represents the loss for generator and discriminator

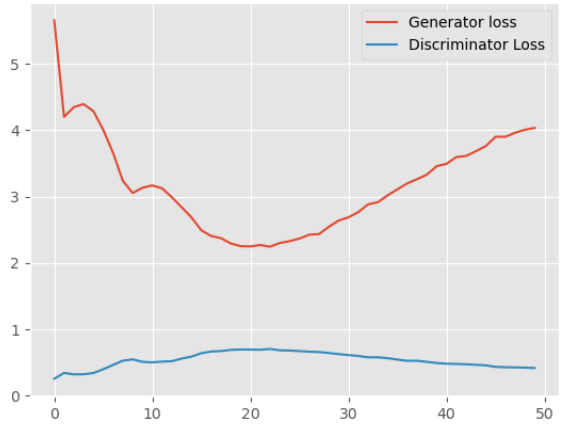

Fig. 6. Example for loss convergence of training with any embeddings

References

[1] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative Adversarial Text to Image Synthesis," in *International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1060–1069, iSSN: 1938-7228, visited on 25/06/2021. [Online]. Available: http://proceedings.mlr.press/v48/reed16.html

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv e-prints*, p. arXiv:1406.2661, Jun. 2014, visited on 28/05/2021.

[3] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010, visited on 27/06/2021.
[4] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008, visited on 27/06/2021.
[5] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014, visited on 28/05/2021. [Online]. Available: http://arxiv.org/abs/1405.0312
[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 11 2016.
[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *arXiv e-prints*, p. arXiv:1607.04606, Jul. 2016, visited on 27/06/2021.
[8] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv e-prints*, p. arXiv:1607.01759, Jul. 2016, visited on 27/06/2021.
[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv e-prints*, p. arXiv:1301.3781, Jan. 2013, visited on 28/06/2021.
[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019, visited on 27/06/2021. [Online]. Available: http://arxiv.org/abs/1910.01108
[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018, visited on 28/06/2021. [Online]. Available: http://arxiv.org/abs/1810.04805
[12] H. Alqahtani, M. Kavakli-Thorne, and D. G. Kumar Ahuja, "An analysis of evaluation metrics of gans," 07 2019.
[13] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016, visited on 28/06/2021.

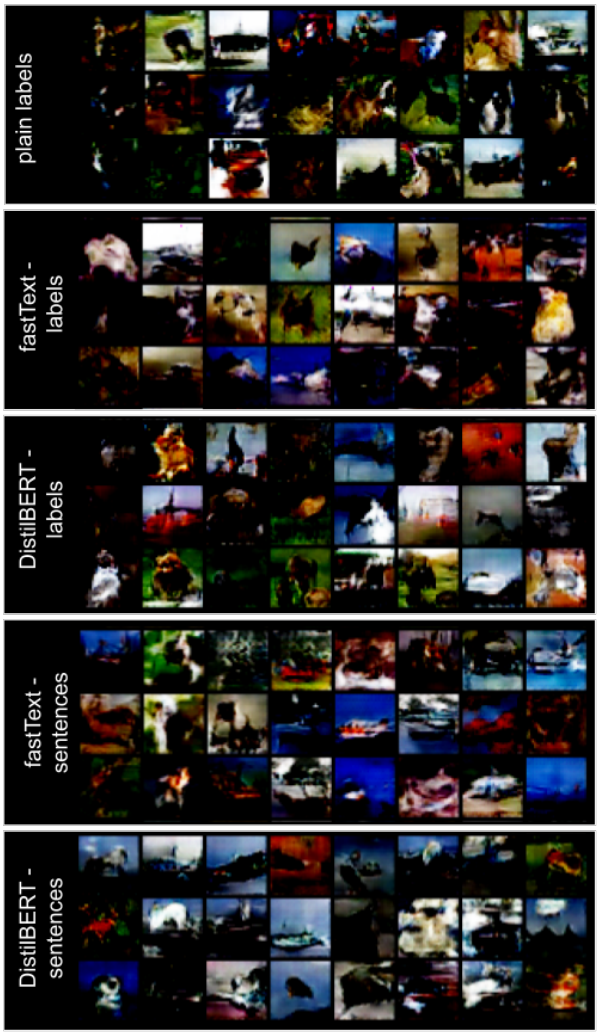

Fig. 7. Generated images on CIFAR10 dataset with best hyperparameter and a fixed latent feature space of 64

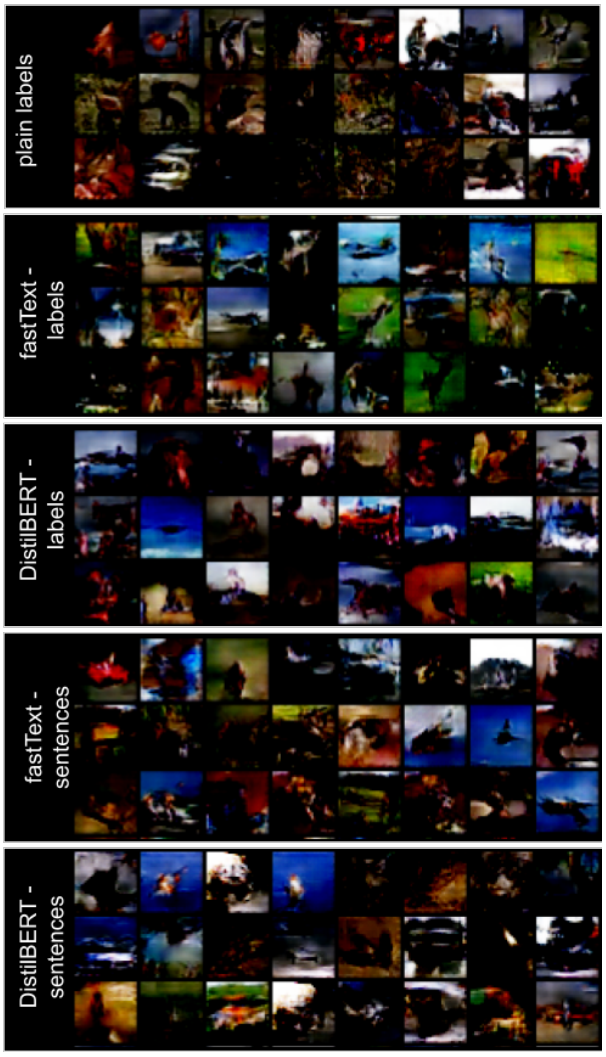

Fig. 8. Generated images on CIFAR10 dataset with best hyperparameter and a fixed latent feature space of 128

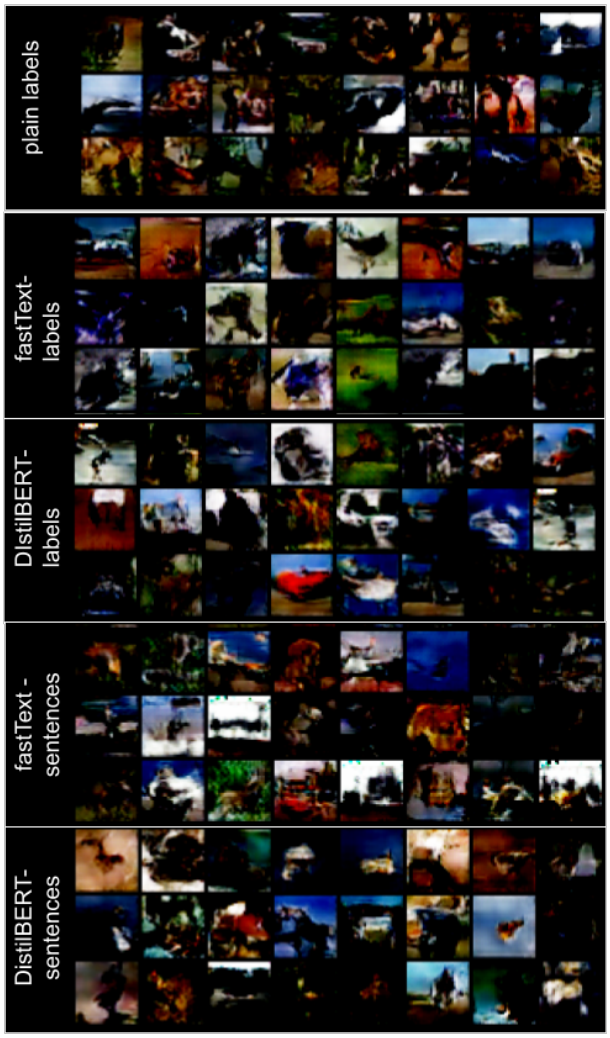

Fig. 9. Generated images on CIFAR10 dataset with best hyperparameter and a fixed feature space of 256

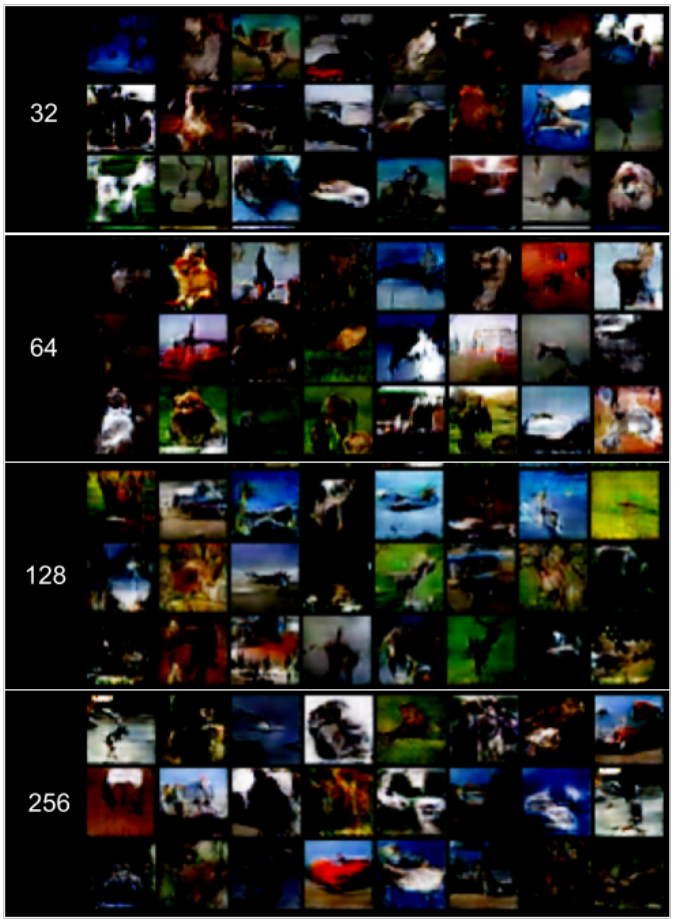

Fig. 10. Comparison of different latent feature space sizes. DistilBERT embeddings were used