Text-To-Image-GAN

Alessa Grund, Tom Hatton, Jolan Nattebrede

March 2018

1 Task Description

For this task, we replicated a part of a Neural Network Architecture proposed by Reed et al. [5], which consists of a Text Embedding Network that translates detailed text descriptions into a text embedding; and a Generative Adversarial Network (GAN) that learns to generate images given the text embeddings. The goal is to generate plausible looking images that fit to the provided text descriptions. The dataset we use is called Oxford-102 Flowers dataset [1] that contains images of 150 classes of flowers and 10 text descriptions for every image. We only implement the GAN and not the Text Embedding Network, instead we use text embeddings for the flower dataset that are available. Our Network learned to generates images that can be recognized as flowers and relate to the text description in a relatively good manner. We do not manage to produce images of the quality produced in the original paper [5] although we trained our network more epochs then it was done in the original paper.

2 Related Work and similar approaches

By solving typical tasks like the one described above one issue remains which is not solely solved by deep learning, namely the multimodality. There are, for example, for each image description many plausible combinations of pixel values which would perfectly fit the given description. However, this conditional multimodality is handled by the adversarial competition between the generator and the discriminator and their conditioning on contextual information present in adversarial nets such as proposed by Goodfellow et al [2].

Multimodality in deep learning was more extensively investigated by Ngiam et al.[8]. Testing on classical deep learning, bimodal and crossmodal conditions they showed that deep autoencoders are able to achieve cross modality learning, i.e. having learned on one modality and being tested on another, by discovering better representations of one modality (video) when supplemented with additional information (audio). However, their bimodal autoencoders did not perform as well as their video-only autoencoders. This is due to the fact that the unimodal autoencoders only learn the features of one modality (video) which are also applicable for audio reconstruction whereas the bimodal autoencoders learn audio-only, video-only and modal invariant features which might

[1] http://www.robots.ox.ac.uk/ vgg/data/flowers/102/index.html

not be optimal for a task which has only one kind of input. Multimodal learning tackles the difficulty of not knowing beforehand what the appropriate features are for a given task and deep learning is a powerful method for discovering multimodal features.

Strongly motivated by Ngiam et al.[8] were the works of Srivastava and Salakhutdinov [4] about multimodal learning using deep Boltzmann machines. In contrast to Ngiam's et al. feed-forward autoencoder they used a probabilistic generative model for learning multimodal data representations. Srivastava and Salakhutdinov built a model that combines multiple data modalities into a unified representation which is able to capture certain features that can be used for classification and retrieval. Like the task of Reed et al. [5] that our task is based on, Srivastava Salakhutdinov applied their model also to Text-Image data, showing that it is able to generate captions to images. The sampling from conditional distributions over each available data modality enables the formation of representations even though some data modalities are missing. This is an addition to the model of Ngiam et al. [8] which exhibited impediment to performance when faced with unimodal input at training time. Srivastava and Salakhutdinov did not show this performance drawback, yet they showed that their multimodal model even supports classification and retrieval when only given unimodal input at test time.

For a GAN task also important is not solely the discriminative part and the multimodal feature learning but also the generating of new data, given the extracted feature information. Dosovitskiy, Springenberg and Brox [1] established evidence for the use of Convolutional Neural Networks (CNN) for generative purposes. Typical tasks solved by CNNs include a discriminative component. Dosovitskiy et al. take a standard discriminative CNN and modified it so that it is able to generate images of cairs given high-level information. This high-level information includes class information, perspective information as well as other informational parameters regarding size or colour. They showed that with supervised training it is possible to use CNNs for generative purposes and that the CNN does not solely learn to produce training images but rather is able to generalise well.

3 Theoretical Basis

3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) have been proposed by Goodfellow et al [2]. A GAN consists of a Generator G and a Discriminator D, both models are often implemented as Neural Networks. The Generator G gets a noise vector z as input and tries to output a plausible image G(z). The Discriminator D receives a training data sample x or a generated data sample G(z) as input and tries to classify the input correctly as being generated or real. The Discriminator D outputs D(v) which is the probability of an image v to be from the same distribution as the the training data. The underlying distribution of the training data is referred to as p_{data}. The generator attempts to learn to produce samples from p_{data}, the approximation of the distribution by the generator is referred

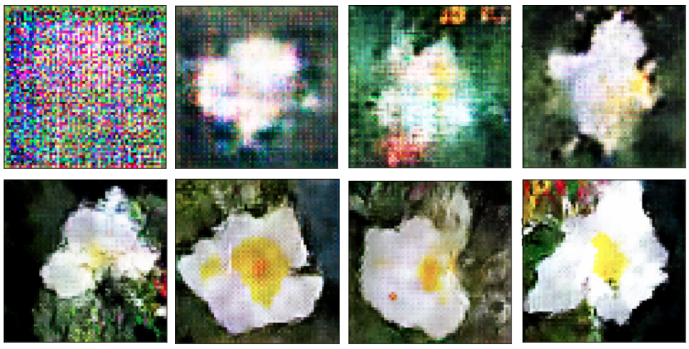

Figure 1: This figure shows an exemplified trainingprocess for one class of flowers. The Caption could read something like: "This flower has petals that are white with yellow stamen". Epochs are approx: (1, 80, 300, 380, 820, 1350, 1520, 1710)

to as p_g. By adjusting the parameters of the generator the approximation p_g gets more similar to real underlying distribution p_{data}. During training, the Discriminator D and the Generator G play a minimax game:

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

The minimization of V(D,G) forces G to generate images that are harder for D to differentiate from the training data. By maximizing V(G,D), the Discriminator D attempts to improve its performance of classifying training data and generated images correctly. The global optimum is reached iff $p_{data} = p_g$, so if the underlying real distribution p_{data} is the same as the approximated distribution p_g by the generator. If this is the case, the Discriminator is unable to differentiate between generated samples and real samples and outputs constant 0.5.

Generator and Discriminator are strongly dependent. A better Discriminator increases the loss of the Generator and a better Generator increases the loss of the Discriminator. During training the loss of the Generator is first backpropagated through the discriminator. Therefore, balancing those two network can be quite difficult. Improvements early on in training can be achieved by changing the loss function of the generator; instead of minimizing $log(1 - D(G(z)))$ to maximize $log(D(G(z)))$. An improvement providing stability during training was proposed by Radford et al [3]. D gets implemented as a Convolutional Neural Network and G as a Deconvolutional Neural Network. Both Networks upsample/downsample the data with stride 2 convolutions/deconvolutions and in the layers of the Generator the leaky relu is used as activation function.

An extension of the standard GAN is the Conditional GAN [7]. Conditional GANS make it possible to influence which data should be generated. Therefore, an additional vector with encoded features is provided to G and D. D learn also if the provided feature vector is compatible to the data. In many approaches D is not only trained with pairs of (real data, real feature) and (generated data,

Figure 2: This figure shows good results on the test set.

provided feature) but also with a third type of input (real data, mismatched features) which D must learn to score to fake. This assures that G will learn to take the information of the feature vectors into account to fool D.

3.2 Text embeddings

A specific text encoder has been proposed by Reed et al.[6], which transforms text in visually-discriminative vector representations. A visually-discriminative vector representation stores only visually relevant features in a vector e.g. colors, class of object, shapes, etc. Such a representation can be easily constructed by a image encoder given an image, whereas for a text encoder to learn such a representation can be more demanding. The text encoder must learn to focus on words that describe visually concepts only and to encode them in an appropriate way. Due to the fact, that such a visually-discriminative vector represenation is easily learned by a image encoder, we use a pretrained image encoder ϑ to train a text encoder ϕ. Training data for the text encoder consists of triples: images v_n, text description t_n, class y_n.The training process minimizes the difference between all text encodings $\phi(t_n)$ and all image encodings $\vartheta(v_n)$ that belong to the same class y_n.The loss function is the 0-1 loss between the real class y_n and the class assignment $f_v(v_n)$ and $f_t(v_t)$.

$$\frac{1}{N}\sum_{n=1}^{N} \triangle(y_n, f_v(v_n)) + \triangle(y_n, f_t(v_t)) \tag{2}$$

$$f_v(v) = \underset{y \in Y}{\operatorname{argmax}} E_{t \sim T(y)}[\phi(v)^T \varphi(t)] \tag{3}$$

$$f_t(t) = \underset{y \in Y}{\operatorname{argmax}} E_{v \sim T(y)}[\phi(v)^T \varphi(t)] \tag{4}$$

$f_v(v)$ is the class y_i of all texts t_i belonging to class y_i , which text embeddings $\phi(t_i)$ are in average most similar to the input image's embedding $\varphi(v)$. The idea is that we want the text embeddings and image embeddings from the same class to me more similar then text embeddings and image embeddings from

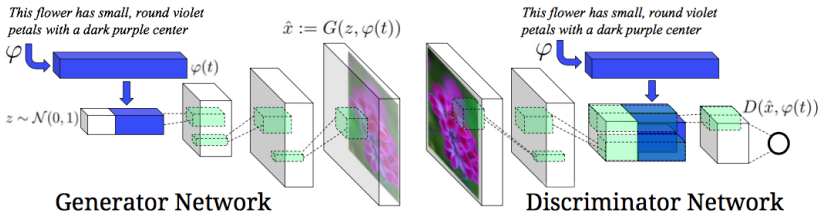

Figure 3: This figure shows the structure of the Generator and the Discriminator. Taken from [5]

different classes. $f_t(t)$ is the class y_i of the images v_i which encodings $\varphi(v_i)$ are most similar to the input text's embedding $\phi(t)$.Those two functions ensure that the text embedding encodes relevent visual information, which are highly useful when generating images given text descriptions.

4 Network Architecture

The structure proposed by Reed et al. [5] is comprised of two ANNs. A text encoder that yields visually-discriminative sentence embeddings given text captions and a GAN that generates images based on these sentence embeddings. In order to limit the scope of our task, we decided to only replicate the GAN. Fortunately, pretrained sentence embeddings for the oxford 102 flower dataset where available which we used for training and testing. The Generator G is given the random noise vector z sampled from a normal distribution and the sentence embeddings as an input. The embeddings go through a feed-forward layer of shape 1024x128 to yield a reduced embedding which are normalized by batch normalization and activated by a relu. The reduced embeddings and the noise vector are concatenated and represent the input to the first layer of the generator.

The first layer is a feed-forward layer as well and has the shape 228x16,348 but is immediately reshaped to yield 1024 feature maps of size 4x4. The second layer is a deconvolution layer and reduces the number of feature maps to 512 while increasing their size to 8x8. The third, fourth and fifth layers are deconvolution layers as well projects to 16x16x256 then to 32x32x128 and finally to 64x64x3 which is the intended shape of the generated images $\hat{x}$. All layers are normalized by batch normalization and activated by the relu. All deconvolution layers have a stride of 2x2 and a 5x5 filter.

The discriminator takes a batch of real images with the correct labels and a batch of real images with incorrect labels alongside the generated pictures $\hat{x}$ as input. In practice, we just sample a batch of real images with correct labels and sample a second batch of images and use the labels of the first batch. While it might seem intuitive to just concatenate all three batches of images and feed them to the discriminator. In reality, this is not a good idea and it kept our network from learning in the beginning. Batch normalization is applied in the

Figure 4: This figure shows bad results on the test set.

discriminator as well and a large concatenated batch would normalize over different types of images (generated and real). To avoid this it is better to feed all three batches independently into the discriminator and to reuse the weights[2].

The exact structure of the discriminator was not given by the paper, it was just mentioned that the implementation was based on the DcGAN [3] paper. We loosely followed this and build our structure as follows. The first layer of the discriminator gets batch normalized images of dimensions 64x64x3 as input. It increases the number of feature maps to 128 and reduces the size to 32x32. The second, third, and fourth layer are convolutional layers as well and reduce the size further while increasing the number of feature maps. From 32x32x128 the layers project to 16x16x256, then to 8x8x512 and finally to 4x4x1024. All convolutional kernels are of size 5x5 and have a stride of 2x2 also they are batch normalized and activated by the leaky relu.

The output of layer four is concatenated with reduced embeddings similar to the generator. The reduced embeddings are going through a feedforward layer that is batch normalized, activated by a relu and of size 1024x128. Again the output is directly reshaped into shape 4x4x8 and can be concatenated with the output of layer 4 to obtain the shape 4x4x32. The fifth layer is a 1x1 convolutional layer that just reduces the number of feature maps to 4. The output is reshaped so it can be fed into the final layer. The sixth and final layer is a feedforward layer of size 64x1 and is not batch normalized and activated by the sigmoid.

Later on, we added some alterations to this structure to improve the outcome. Based on recommendations online [3] we exchanged the last layer of the generator to be activated by the tanh function. As the tanh ranges between [-1, 1] we had to squash it between [0, 1] such that it resembles the values of an RGB picture. Also, we made the second and fourth layer of the generator dropout layers with a keep probability of 50%.

The objective function of the discriminator used in the paper [5] differs from the

[2]https://github.com/paarthneekhara/text-to-image
[3]https://github.com/soumith/ganhacks

original discriminator function due to the task. In order to train a generator to produce images out of captions, we need a discriminator to provide appropriate feedback. Therefore, the discriminator need to learn if a given image is related to a given caption. This is achieved by training the discriminator on pairs of real images and captions that do not describe the images. Those pairs and all generated images the discriminator should learn to score to zero. The loss function contains three scores: s_r is the mean activation of the discriminator if images and related captions are provided, s_w is the mean activation for images and unrelated captions and s_f is the mean activation for generated images and their captions. The loss is then calculated the following way:

$$L_{discriminator} = -(log(s_r) + (log(1 - s_w) + log(1 - s_f))/2) \tag{5}$$

$$L_{generator} = -(log(s_f)) \tag{6}$$

The loss of the generator is the same as in the original GAN paper, but gets implicitly changed due to the fact that the loss get first backpropagated through the discriminator. Therefore, the gradients also convey information how to change the image such that it fits to the text embeddings.

5 Results

Our network learns to generated images that show flowers, but they are far from being indistinguishable from real images. In Figure 1 the learning process is displayed. First the generator learns to generate images with a big color patch that is related to the caption. Then the generator seems to learn the concept of a second smaller color patch places in the middle of the big color patch. Further training seem to result in a more flower-like shape and in taking more and more detailed descriptions into account.

With GANs the evaluation is known to be problematic, the loss function does not provide a useful measurement in this case, because generator and discriminator are so interconnected. Figure 5 shows the two losses at the beginning of training. The discriminator loss is relatively high due to the fact that it is also trained to learn to differentiate between images with correct text embeddings and images with wrong text embeddings. While training, the two losses show parallel fluctuations but the mean values stays relatively stable over time. The quality of the generated images improves while the two losses stay stable, and therefore do not provide a useful measurement of performance.

Due to the fact that neither the loss nor the generated images could give a good approximation of performance it was impossible to determine if a small change in our network improved or worsen the overall performance. When we saw that our model trained 1700 epochs would not produce high quality samples as in the original paper in which the network was only trained 600 epochs, we added dropout and tanh as activation of the generators' last layer. This two methods are known to improve training of GANs [4]. Nevertheless the performance did not reach the performance described in the original paper.

[4] https://github.com/soumith/ganhacks

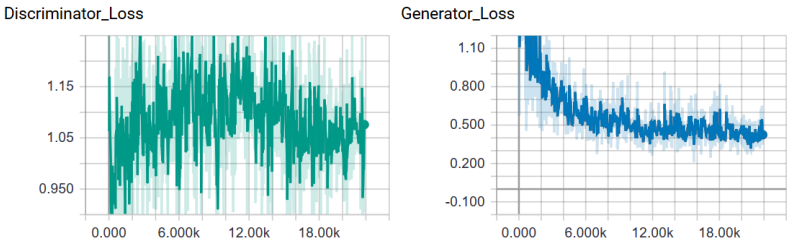

Figure 5: Loss of the Generator and Discriminator for the first 300 epochs

There are multiple reasons possible why we did not fully succeed in replicating the paper. In the paper, data augmentation is applied e.g. flipping the images which may result in better generalization. In the paper the architecture of the discriminator is not described in detail, so our discriminator may not have enough capacity. Furthermore, some information was not provided in the paper e.g. initialization and which 80 classes of flowers they used from the dataset. Also, we can not rule out that an error in our code causes our network to learn slowly. ART!!!hier

References

[1] Dosovitskiy, A., Springenberg, J. T., Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on (pp. 1538-1546). IEEE.

[2] Goodfellow,I., Pouget-Abadie, J.,Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., Generative adversarial nets, in NIPS, 2014, pp. 2672–2680.

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

[4] Srivastava, N., Salakhutdinov, R. R. (2012). Multimodal learning with deep boltzmann machines. In Advances in neural information processing systems (pp. 2222-2230).

[5] Reed, S.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; Lee, H. Generative Adversarial Text to Image Synthesis.arXiv:1605.05396. 2016.

[6] Reed, S., Akata, Z., Lee, H., and Schiele, B. Learning deep representations for fine-grained visual descriptions.In CVPR, 2016.

[7] Mirza, M., Osindero, S., Conditional Generative Adversarial Nets.arXiv:1411.1784, 2014.

[8] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A. Y. (2011). Multimodal deep learning. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 689-696).