Name: Frank Gabel
Course: Applied Computer Science
Student number: 3537204
Date: September 9, 2018

Explainable Machine Learning Seminar

Generative Adversarial Text-to-Image Synthesis

Reed et al.

Frank Gabel

Contents

Figure 1: A very recent applications of GANs has been to generate photo-realistic images. These celebrity pictures are actually cherry-picked fakes. [5]

1 Introduction

This report aims at introducing the reader to the article "Generative Adversarial Text-to-image synthesis" by Scott Reed and coauthors from 2016 [1].

Automatic synthesis of realistic images from conditioning information such as text, speech or data from other domains would be interesting and useful, and recent AI systems have made large leaps towards this goal. Particularly, generic and powerful recurrent neural network architectures have been developed to learn discriminative text feature representations in recent years. Meanwhile, deep convolutional generative adversarial networks have begun to generate highly compelling images of specific categories, such as faces [2](see also Figure 1), "art" [3], and, classically, room interiors such as bedrooms [4]. In the seminal work of Reed et al. from 2016 [1], a deep architecture and GAN formulation has been developed to effectively bridge these advances in text and image modelling, translating visual concepts from characters to pixels.
In the following, we begin by classing the conditional image creation task in the context of explainability in machine learning, followed by briefly summarizing the GAN concept and proceeded by formalizeing a natural extension to it, the conditional GAN model.

1.1 A word on explainability

In many applications, neural networks are really good predictors, but mainly work as "black boxes". This is problematic for a number of reasons: How can trust in these methods be established? Will these methods contribute to the advancement of science, when they produce numbers from learning mere correlations, not (causal) insight? Is there legal ground to challenge predictions that have been made by a machine learning algorithm?
The paper "Generative Adversarial Text-to-image synthesis" adds to the explainabiltiy of neural networks as textual descriptions are fed in which are easy to understand for humans, making it possible to interpret and visualize implicit knowledge of a complex method.

1.2 Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GANs) have been introduced by Ian Goodfellow in his groundbreaking 2014 paper [6] and have seen rapid development[1] and many applications[2] since.

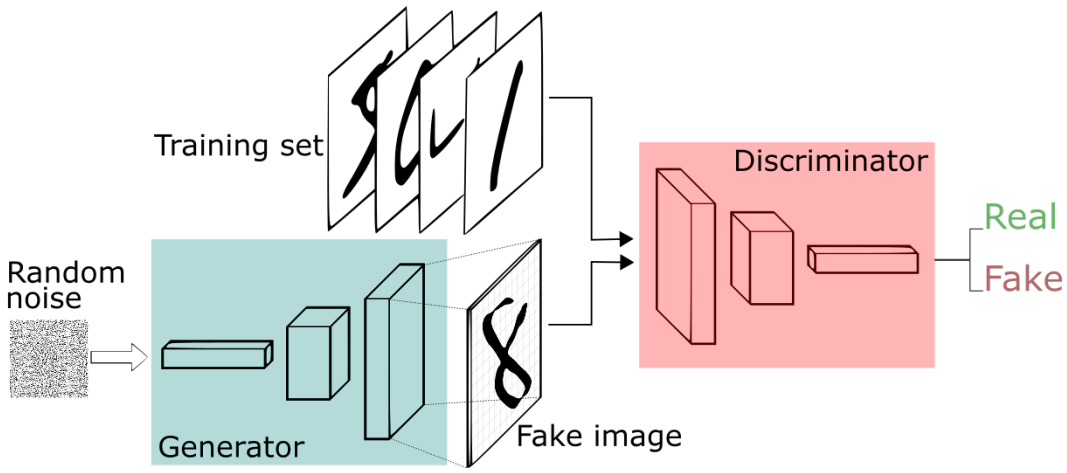

Figure 2: A conceptual overview of the GAN training procedure. The generator G creates images from random noise, while the discriminator D learns to differentiate between real images from the training set and fake images from the generator.

GANs consist of two neural networks - a so-called generator G and a so-called discriminator D that compete in a competitive process, allowing the derivation of gradient

[1] A timeline of GAN developments and extensions is available at https://github.com/dongb5/GAN-Timeline.
[2] A curated list of applications using GANs can be found at https://github.com/nightrome/really-awesome-gan.

signals: The discriminator is trained to distinguish real training data from synthesized images, and the generator is trained to synthesize images from white noise with the sole purpose of fooling the discriminator. More concisely, D and G play a minimax game on $V(D,G)$ that can be formalized as follows:

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

The optimization function of a naive GAN. The networks are optimized jointly (the two networks receive gradient information from one another).

A graphical sketch of the GAN training procedure is shown in Figure 2. The probability distributions of input and output spaces used in the above formulation can be subsumed as follows:

X - data space

representing an image output from the generator or input to the discriminator of normalized pixel values $X = [0,1]^W \times [0,1]^H \times C$, where W, H represent the resolution of the input images, and C usually depicts the set of distinct color channels in the input images. Using the images in the training data and their associated conditioning information, we can define a density function $p_{\text{data}}(x,y)$ of images. This is exactly the density function that GANs are encouraged to learn.

Z - noise space

used as a source of entropy, seeding the generative model. More formally, we get $Z = \mathbb{R}^{d_Z}$, where d_Z is a hyperparameter representing the dimensionality of the random input vector. Values $z \in Z$ are sampled from a noise distribution $p_z(z)$. Most commonly, p_z is set to be simple Gaussian noise.

Ian Goodfellow proved that this minimax game has a global optimum precisely when $p_g = p_{data}$, and that under mild conditions (particularly, G and D have enough learning capacity) p_g converges to p_{data}. In practice, in the start of training, samples from G are extremely poor and rejected by D with high confidence.

Using the above adversarial training procedure, generative adversarial networks (GANs) provide a way to learn deep representations without extensively annotated training data. They achieve this through deriving backpropagation signals from the competitive process described above. The representations that can be learned by GANs may be used in a variety of applications, including image synthesis, semantic image editing, style transfer, image super-resolution and classification. We will now begin to derive how to use GANs to generate realistic images from textual descriptions.

1.3 Motivation for Conditional Generative Adversarial Networks (cGANs)

Shortly after the rise of naive GANs, the idea of conditional GANs has been brought up by Mehdi Mirza and coauthor [7]. By establishing some arbitrary vectorized conditioning information y and restricting both the generator in its output and the discriminator in its expected input, we are able to encourage the generator to draw samples that reflect whatever was encoded in y (while the discriminator learns alongside the generator). Equivalently, we might think of this condition y as engaging both the generator and discriminator in a particular mode of generation or prediction.

Following this logic, the loss function changes in the following way:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y))] \quad (2)$$

The optimization function of a naive GAN. The networks are optimized jointly (the two networks receive gradient information from one another).

Here, X and Z keep their meaning of being data and noise space. In order to encode textual information, the data are conditioned on $y \in Y$, where Y is the embedding space.

Y - embedding space

used to condition the generative model on some external information such as text embeddings. Hereby, $Y = \mathbb{R}^{d_Y}$, where d_Y is a hyperparameter. Using condition information provided in the training data, we can define a density model $p_y(y)$. In the following chapter, we will talk about the abovementioned paper, using this technique to condition on textual information.

In the paper to be discussed in the next chapter, y well be an embedding of textual information (the particular technique used to vectorize sentences is “Deep symmetric structured joint embedding” - we refer to the original paper for further information on the technique[3]).

[3] An excellent introduction to word embeddings in general and Google’s *word2vec* in particular can be found on https://www.tensorflow.org/tutorials/representation/word2vec

2 Generative Adversarial Text to Image Synthesis

The contribution of the paper by Reed et al. [1] is to add text conditioning (particularly in the form of sentence embeddings) to the cGAN framework. To that end, their approach is to train a deep convolutional generative adversarial network (DC-GAN) conditioned on text features encoded by a hybrid character-level recurrent neural network. Both the generator network G and the discriminator network D perform feed-forward inference conditioned on the text feature. The approach of the authors is to train a deep convolutional generative adversarial network (DC-GAN).

2.1 Model architecture

The generator network can be denoted as $G : \mathbb{R}^Z \times \mathbb{R}^T \rightarrow \mathbb{R}^D$, the discriminator as $D : \mathbb{R}^D \times \mathbb{R}^T \rightarrow \{0, 1\}$, where T is the dimension of the text description embedding, D is the dimension of the image, and Z is the dimension of the noise input to G. The network architecture is illustrated in Figure 3.

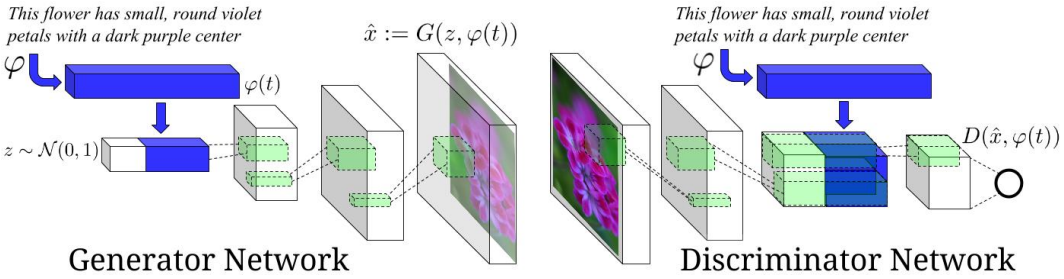

Figure 3: A conceptual overview of the GAN training procedure. The generator G creates images from random noise, while the discriminator D learns to differentiate between real images from the training set and fake images from the generator.

First, in the generator G, samples from the noise prior $z \in \mathbb{R}^Z \sim N(0, 1)$ are drawn and, using the text encoder ϕ, the text query t is encoded. The description embedding $\phi(t)$ is first compressed using a fully-connected layer to a small dimension (in practice, $T = 128$ is used) followed by leaky-ReLU and then concatenated to the noise vector z. Following this, inference proceeds as in a normal deconvolutional network: feed-forward the input signal through the generator G; a synthetic image $\hat{x}$ is generated via $\hat{x} \leftarrow G(z, \phi(t))$. Image generation corresponds to feed-forward inference in the generator G conditioned on query text and a noise sample. In the discriminator D, several layers of stride 2 convolution with spatial batch normalization [8] followed by leaky ReLU are performed. The dimensionality of the description embedding $\phi(t)$ is

again reduced in a (separate) fully-connected layer followed by rectification. When the spatial dimension of the discriminator is 4×4, the description embedding is replicated spatially and a depth concatenation is performed. Finally, a 1×1 convolution is applied, followed by rectification and a 4×4 convolution to compute the final score from D. Batch normalization is performed on all convolutional layers.

2.2 Training and Extensions

2.3 Matching-aware discriminator (GAN-CLS)

When training cGANs in a naive manner, the discriminator observes two kinds of inputs: real images with matching text, and synthetic images with arbitrary text. In that case, it must implicitly separate two error sources: images that just seem non-natural (irrespective of the textual description), and seemingly natural, realistic images that do not match the conditioning information. Based on the intuition that this may complicate learning dynamics, the authors modified the GAN training algorithm to separate these error sources. In addition to the usual real/fake inputs to the discriminator during training, a third type of input consisting of real images with mismatched text, which the discriminator must learn to score as fake, is added. By learning to optimize image/text matching in addition to the image realism, the discriminator can provide an additional signal to the generator.

2.4 Learning with manifold interpolation (GAN-INT)

It has been shown earlier that deep networks learn representations of inputs that have the nice property that interpolations between embedding pairs lie near the data manifold. This allows the generation of a large amount of additional text embeddings by simply interpolating between embeddings of training set captions. The fact that these text embeddings have no correspondance to human-written text (and thus, no corresponding training image) is not problematic - in the end, the discriminator D learns to output the probability of an image and a text embedding matching. If D does a good job at this, then by satisfying D on interpolated text embeddings G can learn to fill in gaps on the data manifold in between training points (practically, this corresponds to probabilities that are neither near 0 or 1).
This can be expressed mathematically as an additional minimization objective to the generator:

$$\mathbb{E}_{t_1,t_2\sim p_{data}}[log(1-D(G(z,\beta t_1+(1-\beta)t_2)))] \tag{3}$$

where z is drawn from the noise distribution and β is an interpolation hyperparameters between text embeddings t_1 and t_2. The authors argue that $\beta = 0.5$ works well in practice.

2.5 Inverting the generator for style transfer

If the text encoding $\phi(t)$ captures the image content (e.g. flower shape and colors), then, in order to generate a realistic image, the noise sample z should capture style factors such as background color and pose. With a trained GAN, one may wish to transfer the style of a query image onto the content of a particular text description. To achieve this, one can train a convolutional network to invert G to regress from samples $\hat{x} \leftarrow G(z, \phi(t))$ back onto z by using a simple square loss to train the style encoder

$$\mathbb{L}_{style} = \mathbb{E}_{t,z\sim\mathbb{N}(0,1)}||z - S(G(z, \phi(t)))||_2^2 \tag{4}$$

where S is the style encoder network. With a trained generator and style encoder, style transfer from a query image x onto text t proceeds as follows: $s \leftarrow S(x), \hat{x} \leftarrow G(s, \phi(t))$, where $\hat{x}$ is the result image and s is the predicted style.

3 Experiments

3.1 Training details

The authors mainly used three datasets for their experiments: the CUB dataset of bird images, Oxford-102 dataset of flowers (in this report, these results are omitted as they are similar to the CUB ones) and the MS COCO dataset of more general scenes. The same GAN architecture is used throughout all datasets. Training images were reshaped to $64\times 64 \times 3$. The text encoder produced 1024-dimensional embeddings that were projected to 128 dimensions in both the generator and discriminator before depth concatenation into convolutional feature maps. Generator and discriminator network are trained in alternating steps, using a learning rate of 0.0002 and the ADAM solver (Ba AND Kingma, 2015) with momentum 0.5. The generator noise was sampled from a 100-dimensional unit normal distribution.

3.2 Qualitative results

The authors compare the GAN baseline, GAN-CLS (i.e. with image-text matching discriminator), GAN-INT (i.e. learned with text manifold interpolation) and GAN-INT-CLS

which combines both. Results on CUB can be seen in Figure 4. Apparently, both GAN and GAN-CLS only get some color information right whole the images do not look real. If adding manifold interpolation (GAN-INT), the images become more realistic and usually match all or at least part of the caption.

Figure 5 shows results of a GAN-CLS on the more general MS-COCO dataset, displaying the generalization capability of the approach on a general set of images that contain multiple objects and diverse backgrounds. On first sight, the resulting samples are pretty sharp, similar to other GAN-based image synthesis models. Also, the model shows samples of large diversity when confronted with drawing multiple noise vectors and using the same fixed text encoding. However, even if these results are encouraging at first, upon close inspection it is clear that the generated scenes are not usually coherent; for example the human-like blobs in the baseball scenes lack clearly articulated parts. In future work, it may be interesting to add additional dependency constraints such as hierarchical structures into the image synthesis model in order to better handle complex multi-object scenes.

Figure 4: Zero-shot (i.e. conditioned on text from unseen test set categories) generated bird images using GAN, GAN-CLS, GAN-INT and GAN-INT-CLS. We found that interpolation regularizer was needed to reliably achieve visually-plausible results.

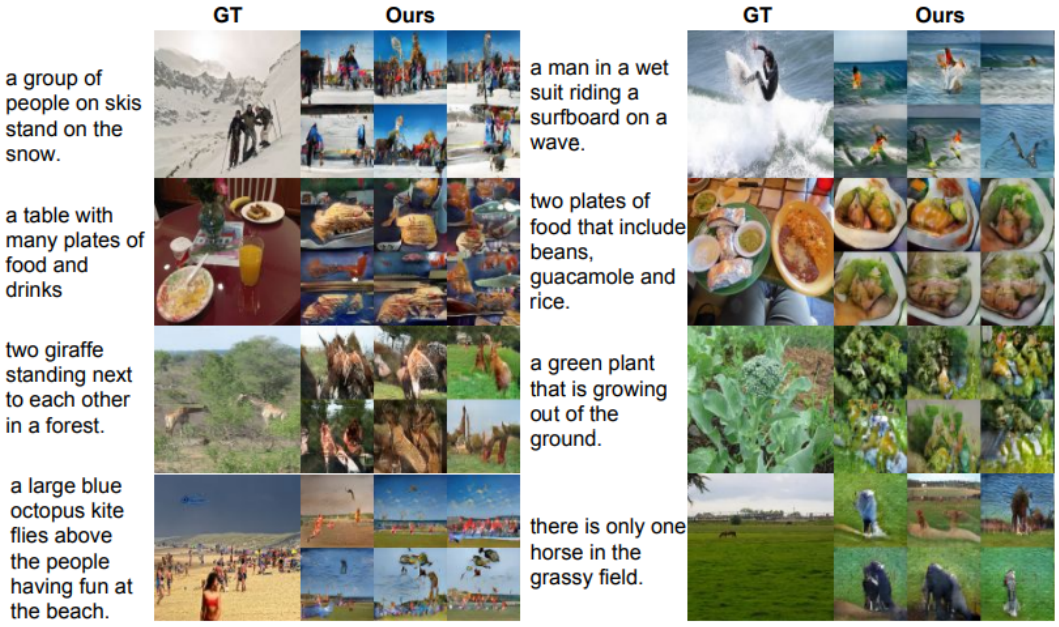

Figure 5: Generating images of general concepts using our GAN-CLS on the MS-COCO validation set. Unlike the case of CUB and Oxford-102, the network must (try to) handle multiple objects and diverse backgrounds.

3.2.1 Manifold Interpolation

Figure 7 shows a visual representation of the learned text manifold by interpolation. Although there is no ground-truth text for the intervening points (as these points correspond to sentences "in between" other sentences), the generated images appear plausible. It's important to keep the noise distribution the same so thatthe only changing factor within each row is the text embedding that we use. Notably, interpolations can accurately reflect color information, such as a bird changing from blue to red while the pose and background mostly stay the same.

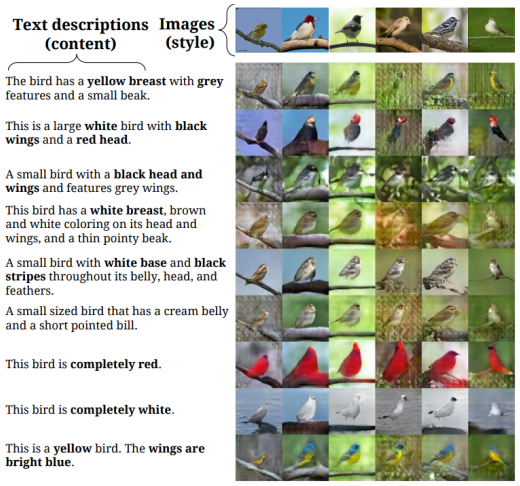

Figure 6: Transfering style from the top row (real) images to the content from the query text, with G acting as a deterministic decoder. The bottom three rows are captions made up by us.

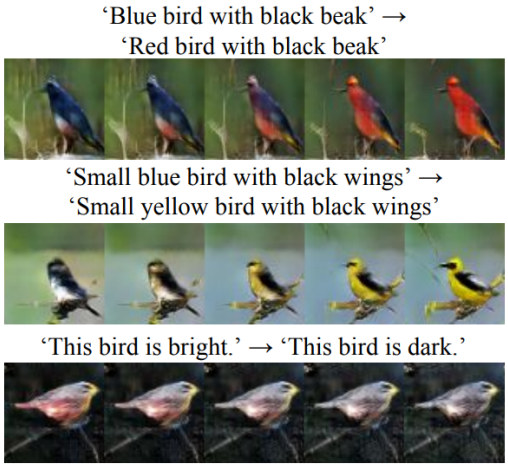

Figure 7: Generated bird images by interpolating between two sentences (within a row the noise is fixed).

3.2.2 Inverting the generator for style transfer

Figure 6 shows that GAN-INT-CLS with a trained style encoder can perform style transfer from an unseen query image onto a text description - apparently, the images generated using the inferred styles can accurately capture the pose information.

In several cases the style transfer preserves detailed background information such as the angle and texture of a tree branch upon which the bird is perched.

4 Recent further work and Outlook

The paper "Generative Adversarial Text to Image Synthesis", subsumed in this report, was one of the earliest papers on applications of cGANs. Shortly after, the authors extended more thorough ideas in another paper - "Learning What and Where to Draw" [9] where cGANs are equipped with information describing what content to draw in which location. Other more recent articles on text-to-image synthesis include TAC-GAN [10] and StackGAN [11], proposing a 2-stage process where one GAN sketches an image and a second corrects the defects and adds more details and resolution. Some of their results are pretty impressive (see figure 8).

Figure 8: A selection of results of a trained StackGAN [11] comprised of two stages - the resulting images already look rather realistic.

In the future, the recent advancements in "unconditional" image generation as shown in the introductory chapter may be combined with a cGAN architecture to improve the quality of generated images further. Some of the other most interesting use-cases span the likes of Font Generation, 3D Object Generation, Image Editing, Face Aging, Domain-transfer, Super-resolution as well as Synthetic Data Generation.

References

1. S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. “Generative Adversarial Text to Image Synthesis”. *CoRR* abs/1605.05396, 2016. arXiv: 1605.05396. URL: http://arxiv.org/abs/1605.05396.
2. J. Gauthier. “Conditional generative adversarial nets for convolutional face generation”. In: 2015.
3. A. M. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone. “CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms”. *CoRR* abs/1706.07068, 2017. arXiv: 1706.07068. URL: http://arxiv.org/abs/1706.07068.
4. A. Radford, L. Metz, and S. Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. *CoRR* abs/1511.06434, 2015. arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434.
5. T. Karras, T. Aila, S. Laine, and J. Lehtinen. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. *CoRR* abs/1710.10196, 2017. arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196.
6. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.
7. M. Mirza and S. Osindero. “Conditional Generative Adversarial Nets”. *CoRR* abs/1411.1784, 2014. arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.
8. S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. *CoRR* abs/1502.03167, 2015. arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.
9. S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. “Learning What and Where to Draw”. *CoRR* abs/1610.02454, 2016. arXiv: 1610.02454. URL: http://arxiv.org/abs/1610.02454.
10. A. Dash, J. C. B. Gamboa, S. Ahmed, M. Liwicki, and M. Z. Afzal. “TAC-GAN - Text Conditioned Auxiliary Classifier Generative Adversarial Network”. *CoRR* abs/1703.06412, 2017. arXiv: 1703.06412. URL: http://arxiv.org/abs/1703.06412.

11. H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. N. Metaxas. "Stack-GAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". *CoRR* abs/1612.03242, 2016. arXiv: 1612.03242. URL: http://arxiv.org/abs/1612.03242.