

Text-to-Image Generation Using Multi-Instance StackGAN

Alex Fu, Yiju Hou
Department of Computer Science
Stanford University
Stanford, CA 94305
{alexfu,yijuhou}@stanford.edu

Abstract

Synthesizing images from text descriptions remains a challenging problem in computer vision. Current AI systems have made remarkable breakthroughs in generating images of a single instance from a specific category. In this paper, we propose using an improved stacked generative adversarial networks (StackGAN) model to explore the text-to-image generation task with multiple instances from a broader variety of categories comparing with previous researches. Our model demonstrates the capability to generate images with complex scene composition consisting of multiple objects based on the semantics of the given input text. Our model proves the potential to generate high-resolution multi-instance images using the improved StackGAN model.

1. Introduction (0.5 -1 page)

The problem of generating images from text descriptions has recently gained interest in the research community and it has various potential applications, such as photo editing, video generation and digital design. In recent researches, the expressiveness of deep convolutional and recurrent networks enables capturing and generalizing the semantics of the input texts, which has improved the performance of text-to-image generation tasks significantly comparing with the traditional attribute representation approaches. Moreover, the recently developed stacked Generative Adversarial Networks (StackGAN) model has demonstrated its capability in generating high-resolution images with photo-realistic details [1].

In this paper, we are interested in translating a natural language caption into an image with one or multiple instances from different object categories. For example, the input can be an image caption like "men riding motorbikes on a dirt road on the countryside". The model first processes the text input and generates corresponding text embeddings to feed into the generative adversarial networks. This step is crucial because the text encoder not only abstracts the objects from the input, such as men, motorbikes, a dirt road, but also captures the complex interactions between the

instances, including relative sizes and positioning. Then the GAN will take the text embedding as input to generate a photo-realistic image of relevant visual information through two stages. The first stage generates a low-resolution image and the second stage improves the quality of the image generated from the previous stage (Figure 1).




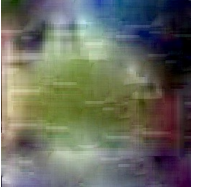


Input	Stage I image	Stage II image
A motor boat and a sail boat on the water in front of a big city.		
A man holding a baseball bat on a field.		
A girl on a board riding along a boat in the water		

Figure 1. Images generated by our improved StackGAN model from unseen text descriptions from the Microsoft COCO dataset [2].

2. Related work (0.5 -1 page, 15 references)

Most of the previous work in text-to-image generative models, including variants of Boltzmann Machines [3][4], Deep Belief Networks [5], are effective but they cannot scale to large datasets.

After the development of neural-network-based models, Mansimov et al. implemented a conditional alignDRAW model, a generative model of images from captions using a soft attention mechanism [6]. Built upon the Deep Recurrent Attention Writer (DRAW) [7], the alignDRAW model iteratively draws patches on a canvas, while attending to the relevant words in the description that

captures the main semantics. The generated images are further refined by a deterministic Laplacian pyramid adversarial network [8].

Another approach to solve the text-to-image generation problem is to use Generative Adversarial Networks (GAN). Reed et al. demonstrated that GAN can effectively generate images conditioned on text descriptions [9][10]. The GAN model successfully in generated photo-realistic images at the resolution of 64×64 , conditioned on text descriptions from the CUB and Oxford-102 dataset [11]. However, their synthesized images in many cases lack details and vivid object parts, e.g., beaks and eyes of birds.

Expanding on previous work for image quality assessment, Odena, et al. proposed the Auxiliary Classifier GAN (AC-GAN) model for conditional image synthesis of 128×128 resolution image samples [12]. The research demonstrates that high resolution samples provide class information not present in low resolution samples, as high-resolution samples are significantly more discriminable comparing with low-resolution samples. The auxiliary classifier discriminator in the AC-GAN model proposed a promising approach for synthesizing high-resolution images in the future [13].

Built upon GAN, the stackGAN model proposed by Zhang et al. decomposes the text-to-image generation problem into two more manageable sub-problems [14][15]. the stacked neural networks are able to rectify defects and refine details with a second GAN stage, which produces more plausible images than those generated by previous approaches. The StackGAN model reaches the state-of-art performance and achieves 28.47% and 20.30% improvements in terms of inception scores on the CUB and Oxford- 102.

Similar researches have been conducted on the tasks of generating chairs and human faces [16][17]. However, these researches are limited to generating one object instance from one category. However, in the real world, objects rarely appear in images. This limitation of the previous researches motivate us to explore multi-instance text-to image photo generation.

3. Methods (2 – 3 pages)

Our end-to-end architecture (Figure 2) includes a text encoder and decoder implemented with a word-level bidirectional recurrent neural network (RNN) consisting of two long short-term memory (LSTM) [18]. In section 3.1 and 3.2, we will give an overview on language model and then on the StackGAN architecture in Section 3.3.

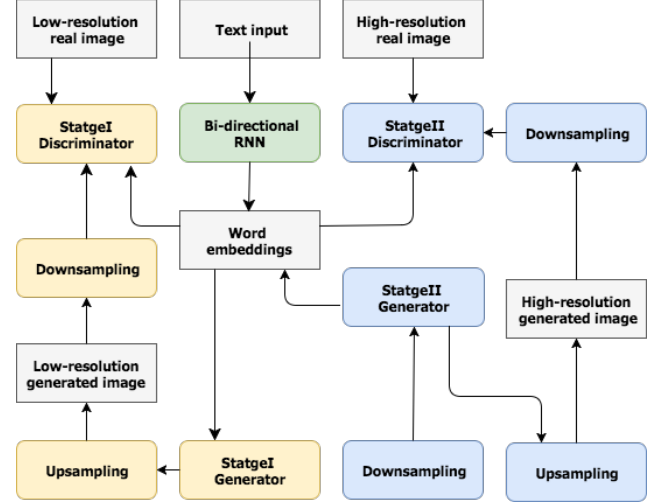


Figure 2: Multi-Instance StackGAN Model

3.1. Language model: bi-directional RNN with attention

In the bi-directional RNN language model, the two LSTMs with forget gates process the input sequence from both forward and backward directions [Gers et al., 2000]. We obtain the caption sentence representation by first transforming each word to an m-dimensional vector representation using the bi-directional RNN. During the transformation, the forward LSTM path computes the sequence of forward hidden, whereas the backward LSTM path computes the sequence of backward hidden states. Then the two hidden states are concatenated together into a sequence.

In our model, the "Inner-Attention" mechanism is employed to replace average pooling to better capture the semantics expressed in the given input [Y. Liu]. When reading a complex sentence, human readers often pay additional attention to certain words to perform deeper inference. The attention-based encoder captures the semantics of the input by simulating this attention behavior. We implemented the attention mechanism in the model as follows:

$$M = \tanh(W^y Y + W^h R_{ave} \oplus e_L)$$

$$a = \text{softmax}(w^T M)$$

$$R_{att} = Y \alpha^T$$

where Y is a matrix consisting of output vectors of the bidirectional LSTM, R_{ave} denotes the output of the mean pooling layer, α denotes the attention vector and R_{att} denotes the attention-weighted test input representation.

Different from the character-level language model used in the original StackGAN paper, the word-level language model used in the architecture helps the generative model to focus on rendering the most important objects in the image by giving them more pixel space and more details. Moreover, the language model can capture the interaction

between objects, which is used to determine the special relationship between objects later in the image generation process.

We also observe that the input text caption can vary significantly in length. To further reduce the training time, we adopt flexible RNN sequence length, which allows the runtime of encoding to be linearly correlated with the length of the question or context.

3.2. Sentinel and Trainable Null

The captions of the Microsoft COCO dataset contain some low-frequency words. Even with attention mechanism, most language models do not have good performance in tasks involving predicting using rare or previously unknown words. Therefore, we decide to handle the low-frequency words with sentinel vectors [19]. In LSTM, a hidden state has limited capacity in retrieving information from relevant previous hidden states, so the sentinel is proposed to increase hidden state capacity by providing a trainable vehicle that is not tied to time steps.

3.3. Multi-instance StackGan

The StackGan architecture was originally proposed by Zhang et al. to perform text-to-image photo generation task on the CUB and Oxford-102 datasets. Our model uses a similar architecture consisting of two stages of GAN – the first stage produces an image with correct object categories, rough shape and basic colors. The generated image from this stage is low resolution and some objects have discoloring, distortional shapes and missing object parts. On top of the first stage GAN, we stack Stage-II GAN to generate realistic high-resolution images conditioned on the low-resolution image and the previously generated text embeddings. The decomposed GAN model makes it possible to generate high-resolution images from captions possible because the training the stacked model is more feasible than training a GAN to generate the final high-resolution image directly. Each stage only need to focus one tasks – Stage I only needs to generate images with correct categories, shapes and colors, while Stage-II GAN only needs to focus on rendering more details and rectifying the defects in low resolution images.

Each GAN is composed of two models that are alternatively trained to compete with each other. The generator G is optimized to reproduce the true data distribution p_{data} by generating images that are difficult for the discriminator D to differentiate from real images. Meanwhile, D is optimized to distinguish real images and fake images generated by G . The training procedure of GANs is to optimize a two-player min-max game with the following objective function:

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

where x is a real image from the true data distribution p_{data} , and z is a noise vector sampled from distribution p_z . In our model, we sample from a Gaussian distribution for z , as suggested in Whites research [20].

Our model uses Conditional GAN, which is an extended model based on GAN where both the generator and discriminator receive additional conditioning variables c [17][21]. This formulation allows G to generate images conditioned on variables c , which is the word embedding in this case. Additionally, we replace the regular discriminator with auxiliary discriminator, which as classify the samples during training. We compute a cross-entropy loss between the predicted category and the ground truth. This technique helps speed up the training process.

4. Dataset and features (0.5 -1 page)

Data The datasets available for training neural networks on the task of image generation are limited. Previous researches have reached compelling performance on CUB and Oxford-102 datasets, which only contain single-instance images from very limited categories. The Microsoft COCO dataset contains more than 300,000 images with 5 captions per image, and these images have multiple objects per image from 80 object categories [Lin]. To our knowledge, Microsoft COCO is the largest publicly available dataset of multi-instance images with detailed image captions. The rich dataset of images and captions contains a variety of objects, background, scene composition, which makes the task of training a good generative model very challenging.

We first attempt to test the robustness of current state-of-art StackGan model on the Microsoft COCO dataset. We randomly select 20 text captions from the bird and flower category respectively from the Microsoft COCO dataset and then use the inputs to generate images using the StackGan models pretrained on the CUB and Oxford-102 datasets. As we have expected, the pretrained StackGan model has decent performance on generating images from a text input that is similar to its training set, which usually contains one instance that does not have complex interactions with its surroundings. For example, the images generated from input “a black bird with small amount of red and white on his wings” are visually plausible (Figure 3).



Figure 3. images generated by the StackGan model pretrained on the CUB dataset given input from input “a black bird with small amount of red and white on his wings” [1].

However, when the input contains multiple objects, StackGan fails to generate the correct number of instances with clear boundaries and spatial relationships. For example, the images generated from the input “two bird hanging out on the branch of a pine tree” do not match the text description (Figure 4).



Figure 4. images generated by the StackGan model pretrained on the CUB dataset given input from input “two bird hanging out on the branch of a pine tree” [1].

Moreover, when given an input that consists of complex interactions between the main object and its surroundings, StackGan fails to capture and express the interactions between the objects. For example, the images generated from the input “Vase with water holds a bunch of flowers in front of window” do not render the objects other than flowers (Figure 5). We have also run experiments to verify when the input does not contain instance from the bird or flower category, the pretrained StackGan models fail to generate the objects from the correct categories.



Figure 5. images generated by the StackGan model pretrained on the Oxford 102 dataset given input from input “Vase with water holds a bunch of flowers in front of window” [1].

As we learn from running the exprienment with StackGan, we conclude that a robust model that can performace well on the Microsoft COCO dataset must be able to handle the following features of the dataset:

1. Identify the catogotries of the instances described in the text input. The Microsoft COCO 2014 Train/Val object instances contains the multimodal category information for each image. In our model, we concatenate the category information with the corresponcing text input to generate text embeddings.
2. Capture the number of instances from each catogotry and render the instances with clear boundaries. The attention mechanism in the language model is desgin to assign more weight on information like amount and color.
3. Capture and express the interactions between objects. For example, one object can be inside another. The main objects should be larger in terms of pixel size than the objects in the background.

For the training of our Multi-instance StackGan model, we use the 2014 Val. Images, wich consists of 40,504 images and each image has at least 5 captions. Additionally, we also use the 2014 Train/Val object instances dataset to retrieve category label information of the images. We split the data set into a train set consisting of 32,910 images and 164,550 captions, a valuation set consisting of 5,063 images and 25,315 captions, and a test set consisting of 2,531 images and 12,655. Because the images from the Microsoft COCO dataset are different size, we preprocess each image to create a low-resolution sample of 76×76 and a high-resolution sample at 304×304 . We also parse all the label information using the Microsoft COCO API and concatenate the label information to the end of the text caption of each image.

5. Experiments and results (2 – 3 pages)

5.1. Evaluation

To demonstrate the effectiveness of our model, we use both quantitative and qualitative methods to evaluate the performance of the neural networks. For qualitative evaluation, we use human rank by human annotators. For quantitative evaluation, we use the Inception model proposed by Salimans et al. to evaluate samples to automatically [5]. The inception score calculated by the model is closely correlated with human evaluation. The metric used in the model is:

$$\exp(\mathbb{E}_x KL(p(y|x)||p(y)))$$

where $p(y|x)$ is the conditional label distribution and $p(y|x = G(Z))dz$ is the variation of the generated images. If a generated image contains objects from the correct categories as described in the caption, it should have a low entropy value in conditional label distribution. Moreover, a robust model should be able to generate varied images, so the marginal $\int p(y|x = G(Z))dz$ should have high entropy. As verified in previous research, a higher inception score indicates better human perception in terms of image quality. We evaluate a set of 1000 images generated using our Multi-Instance StackGan model and achieve an average inception score of 1.12 on average. However, we believe our model has the potential to achieve a much higher inception score if given more time and tuning in training Stage II. As we can see from the samples generated from Stage I, most of the images have the correct categories, color and rough shape. If the stage II process is able to render more plausible details, the generated images should look remarkably more photo realistic. Moreover, the inception model requires a large enough number of samples (i.e. 50k) to produce accurate evaluation. Our model might suffer from not having a large enough evaluation set.

5.2. Tune learning rate

As suggested in the original StackGan paper, the training process is most efficient when using stochastic gradient descent (SGD) for optimizing the Discriminator and ADAM for optimizing the Generator. Therefore, we set the learning rate to 0.001 for the ADAM optimizer and run the following experiment to find a suitable learning rate for SGD. In the first experiment, we start with the same training hyperparameters as used in training the original StackGan model on the Oxford 102 dataset. We run 4 experiments with different learning rates at 0.00005, 0.0001, 0.001 and 0.01 respectively (Figure 5).

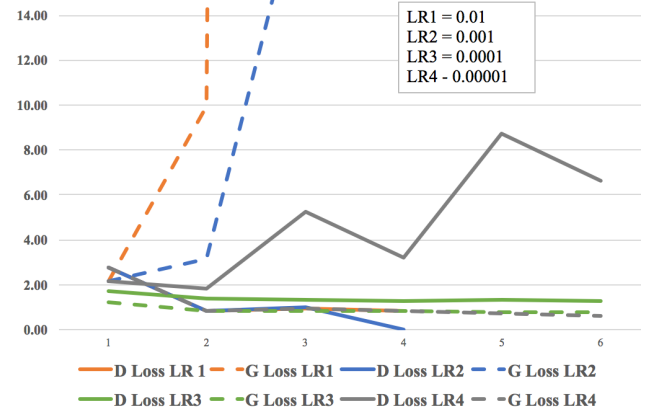


Figure 4. Training Multi-instance StackGan with learning rate at 0.00005, 0.0001, 0.001 and 0.01.

As we can see from the plot, when the learning rate is too high, the Generator loss increases too fast. One possible explanation is the Discriminator is learning exponentially faster than the Generator, so it outperforms the Generator. Consequently, the Generator is not able to generate any synthetic image to fool the Discriminator, and thereby the Generator can not learning useful information from the training.

When the learning rate is too low, we observe Discriminator loss has high variance and spikes while the loss of generator steadily decreases. In this case, the Discriminator is not learning as fast as the Generator. Therefore, the Generator is able to fool the Discriminator with any images, which is not benefit for training the Generator to synthesize photo-realistic images.

When the learning rate is set to 0.0001, both Discriminator loss and Generator loss decrease steadily. Moreover, both of the test and train samples generated from the last two epochs start showing blurry counter lines of objects instead of random noise. Therefore, we conclude from the experiment that we should keep search for the optimal learning rate around 0.0001.

In the next learning rate fine tuning experiment, we conclude that setting the learning rate at 0.0002 achieves the best training result, since both G loss and D loss decrease most steadily [Figure 5]. However, we observe that after epoch 11 the G loss plateaus while D loss keeps dropping.

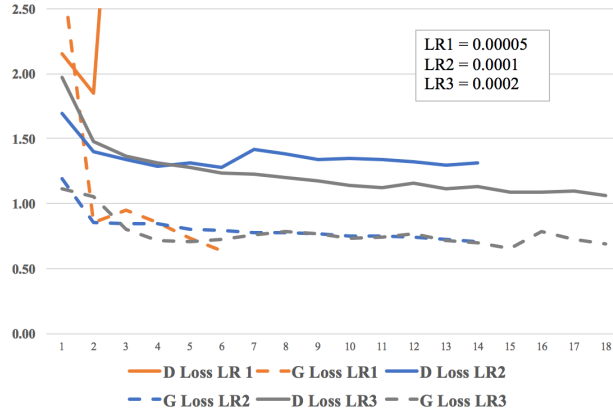


Figure 5. Training Multi-instance StackGan with learning rate at 0.00005, 0.0001 and 0.0002.

5.3. Regularization using Dropout

Our hypothesis is that the Generator is suffering from overfitting while the Discriminator keeps improving at differentiating. We decide to use dropout technique to resolve the overfitting problem. Dropout is a technique that randomly drops units from the neural network during training, preventing the units from overly co-adapting to training data [14]. Dropout also provides a more efficient alternative to approximately combining many different neural network architectures to improve the overall model performance. During training, we reload the snapshot from epoch 11 and add dropout at the rate of 0.2 into both directions of LSTM and after each convolutional layer in the Generator. The loss curve (Figure 6) indicates adopting the dropout technique is effective.

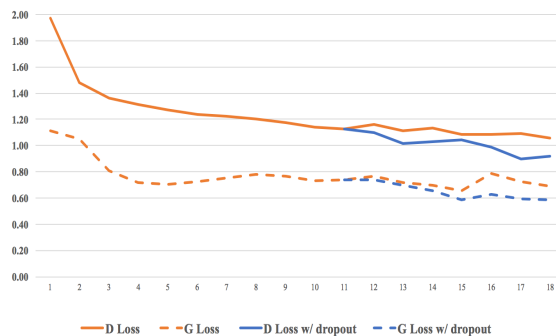


Figure 6. training with and without dropout

5.4. Batch normalization

In order to improve the robustness to bad initialization during training, we implemented the Batch Normalization layers immediately after each fully connected layers and before non-linearities [16]. Batch Normalization helps speed up the training process because it reset the distribution of each layer's inputs during training.

Therefore, we do not have to use low learning rates to converge the model.

5.5. Soft and noisy labels

Label Smoothing is another technique commonly used in neural network training. When we have two target labels: Real = 1 and Fake = 0, then for each incoming sample, if it is real, then replace the real label with a random number between 0.7 and 1.2, and similarly if it is a fake sample, replace the fake label with 0.0 and 0.3 [5]. During the training process, we make the labels the noisy for the discriminator and we also occasionally flip the labels when training the discriminator, as previous research has shown adding some artificial noise to inputs to D noise to every layer of generator helps increasing training efficiency.

6. Conclusion and future work

In this paper, we present an end-to-end text-to image generation system, in which we identify the key components crucial to abstracting the information of multiple instances from different categories to generate plausible scene composition. Moreover, we manage to verify the potential of our model through well-defined experiments. We observe that the gap between the loss decrease of the generator and the loss decrease of the discriminator remains reasonably narrow through training, which indicates the robustness of our training process.

In the future, we would like to improve our implementation from three perspectives. First, we want to spend more time training the Stage II GAN, as we believe our model has not reached its full capacity due to the time constrain of this project. Second, we plan to explore more options with the architecture, such as more flexible attention mechanisms and more expressive convolutional neural network models. Third, we are also constrained by time in design parameters exploration. We would like to experiment with more combinations of hyperparameters to quantify their impacts individually on the training process. Fourth, we would like to develop a more effective preprocessing script. We observe that many of the provided images or captions in the Microsoft COCO dataset rarely occur in real world. We would like to prune away these data points to improve the learning efficiency of our model.

References

- [1] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. arXiv:1612.03242, 2016.
- [2] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, C. L. Microsoft 'coco': Common objects in context. In ECCV. 2014.
- [3] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985.
- [4] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," in *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS'09)*, vol. 5, pp. 448–455, 2009.
- [5] G. E. Hinton and S. Osindero. A fast learning algorithm for deep belief nets. Improved techniques for training gans. In NIPS, 2016
- [6] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In NIPS, 2016
- [7] Y. Kim, Y. Jernite, D. Sontag, A. Rush. Character-Aware Neural Language Models. CoRR. abs/1508.06615, 2017
- [8] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee. Generative Adversarial Text to Image Synthesis. CoRR, abs/1605.05396, 2016.
- [9] 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.
- [10] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In NIPS, 2015
- [11] J. Gauthier. Conditional generative adversarial networks for convolutional face generation. Technical report, 2015.
- [12] Y. Liu, C. Sun, L. Lin, X. Wang. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. CoRR. abs/1605.09090, 2017
- [13] C. Xiong, V. Zhong, and R. Socher. Dynamic Coattention Networks For Question Answering. arXiv:1611.01604, 2016.
- [14] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photorealistic single image super-resolution using a generative adversarial network. arXiv:1609.04802, 2016.
- [15] C. K. Snderby, J. Caballero, L. Theis, W. Shi, and F. Huszr. Amortised map inference for image super-resolution. arXiv:1610.04490, 2016.
- [16] Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In CVPR, 2015.
- [17] S. Hochreiter, J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- [18] F. Gers, J. Schmidhuber, F. Cummins. Learning to Forget: Continual Prediction with LSTM Technical Report IDSIA-01-99 January, 1999.
- [19] T. White. Sampling Generative Networks: Notes on a Few Effective Techniques, CoRR, abs/1609.04468
- [20] C. Xiong, V. Zhong, and R. Socher. Dynamic Coattention Networks For Question Answering. arXiv:1611.01604, 2016.
- [21] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014) 1929-1958
- [22] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv:1411.1784, 2014. 3