Utilizing Normalizing Flows for Anime Face Generation - Journey Log

Allie (Alisher) Turubayev
Deep Learning - Final Project
Hasso-Plattner Institute, Summer 2022
turubayev@uni-potsdam.de

Abstract

Generative models in machine learning have become more ubiquitous with the introduction of Generative Adversarial Networks, or GANs. While GANs achieve impressive results in human faces, still images and even video generation, there are drawbacks of the architecture, such as the inability of exact density estimation and difficulty in training (due to training instability or mode collapse phenomena) (Saxena and Cao 2020). Normalizing flows have emerged as a possible candidate to address these limitations of GANs.

In this paper, the author will present the background information on the normalizing flows, utilize RealNVP (Dinh, Sohl-Dickstein, and Bengio 2016) to generate novel anime faces, and analyse the results in both qualitative and quantitative metrics.

1 Introduction

1.1 Motivation

Tasks involving generation of novel data (be it text, still images, videos, voice or music) have been at the forefront of machine learning research. There are several reasons for continuous interest in this particular challenge:

1. Creation of novel artifacts is seen as uniquely a human task - in other words, the concept of creativity is understood to be central to our general intelligence. Getting a computer to 'think creatively', therefore, is an important milestone in AI/ML research (Lamb 2021);
2. Generating novel artifacts without explicit human input can save resources in real-world scenarios, by either cutting the costs of hiring a human to generate the artifact or by streamlining the process;
3. Generated novel artifacts can be used to augment and complement currently available datasets, which can significantly increase the performance of a trained model (Lamb 2021).

Anime is a term that, at least outside of Japan, refers to animated works produced in Japan. Anime, like many other cultural products from Asia, has gained immense popularity outside of Japan and has been steadily entering the mainstream in the West. Anime style has several important differences to the traditional comic style of the West, which make it appealing to some: simplicity (either in the amount of detail in characters or in the limited animation), expressivity of features (eyes, hair color, or body attributes are often exaggerated) and of emotion (such as anger, infatuation, etc.).

With the growth of the market and cultural significance, anime experienced ever-increasing demand. In response to that, new animation techniques were introduced and adopted by the anime industry, such as the use of computer-generated imagery and 3D animation - particularly in action-filled scenes, where animation by hand is often impractical. However, the increasing costs of animation, coupled with the industry-wide shortage of animators (Margolis 2019) has induced interest in further computer aid to complement or even replace current workflows.

In particular, generation of novel faces in anime style may help in three ways:

1. Reduce the time to create character concept art - a novel generated face may act as an inspiration to the illustrator (Jin et al. 2017);
2. Allow non-professionals to try their hand at character design or animation (Jin et al. 2017);
3. Further popularize the genre by increasing visibility and acceptance of the anime.

1.2 Paper outline

The author will first start with the theory of normalizing flows in Section 2. The introduction of the normalizing flows, as well as the architecture of RealNVP (Dinh, Sohl-Dickstein, and Bengio 2016) will follow in the same section. Section 3 will contain the information on the related works in the field, with the particular focus on novel anime face generation - the task at hand. In Section 4, the dataset description, details of implementation in PyTorch of both RealNVP by (Mu 2019) & DCGAN by (Inkawhich 2018), and training observations will be presented. Finally, discussion and closing remarks will be presented in Section 5.

2 Theoretical Background

2.1 Generative Models

Generative modelling is a subset of unsupervised machine learning problems, where the input data represents a sample of real-world phenomena (such as all the faces of people, or all anime faces ever created).

Formally, generative models can be based on probabilistic or non-probabilistic frameworks (Lamb 2021). Probabilistic models, which are examined in this paper, are tasked with estimating the distribution of the input data $p(x)$, where x is the sample of real-world observations (in this case, a sample of anime faces) (Lamb 2021). Probabilistic models do this by learning an estimate distribution $q_\theta(x)$, where θ denotes a set of parameters (Lamb 2021). The divergence between the two models, denoted by $D(p||q)$, has the following properties:

$$D(p||q) : S \times S \to \mathbb{R}$$

$$D(p||q) \leq 1$$

$$D(p||q) \neq D(q||p)$$

$$D(p||q) = 0 \Leftrightarrow p = q$$

where S is a space of distributions (Lamb 2021).

Thus, the learning is an optimization of the probabilistic generative model with the loss function defined through the corresponding divergence (Lamb 2021):

$$L(\theta) = \operatorname*{argmax}_{\theta} D(p||q_\theta(x))$$

For non-probabilistic generative modelling (Lamb 2021), one can see Deep Photo Style Transfer by (Luan et al. 2017) or Creative Adversarial Networks by (Elgammal et al. 2017). As they are not a focus of this paper, they are skipped in this section.

2.2 Normalizing Flows

Normalizing flows are a framework based on transformations of a simple probability distribution (such as Gaussian) into a more complex distribution (Kobyzev, Prince, and Brubaker 2021). Notably, all transformations that are applied to the input distribution have to be invertible and differentiable (Kobyzev, Prince, and Brubaker 2021).

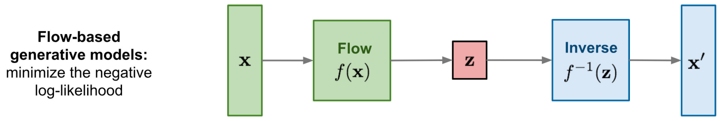
Flow-based generative models: minimize the negative log-likelihood
x
Flow $f(\mathbf{x})$
z
Inverse $f^{-1}(\mathbf{z})$
x'

Figure 1: The architecture of normalizing flows. Adapted from (Weng 2018).

Formally, let $z_0 \in \mathbb{R}$ be a random variable with known, tractable distribution (Kobyzev, Prince, and Brubaker 2021). Additionally, let $f : \mathbb{R} \mapsto \mathbb{R}$ be an invertible smooth mapping (Kosiorek 2018). One can transform the variable $z \sim p_{z_0}(z_0)$ using this mapping f, arriving at a random variable $z_1 \sim p_{z_1}(z_1)$ (Kosiorek 2018). Because of the Change of Variable theorem, which is further described in (Weng 2018), one can infer the probability density of z_1:

$$p_{z_1}(z_1) = p_{z_0}(z_0) \cdot |\det \frac{df}{dz_0}|$$

where $\det \frac{df}{dz_0}$ is a Jacobian determinant of a function f (Weng 2018). By applying several f_i transformations in succession, one could arrive at a complex probability density function $p(x)$, where x is the real-world data.

As mentioned previously, transformations f_i must be invertible, meaning that backward mapping (i.e. from x to z_0) is possible, as shown on the Figure 1.

For computational purposes, the transformations f_i must be chosen carefully to be:

1. easily invertible;
2. with easily calculable Jacobian determinants (Weng 2018).

2.2.1 RealNVP - *real-valued non-volume preserving* transformations

RealNVP brings several observations to the aforementioned normalizing flow background:

1. The determinant of a triangular matrix can be efficiently calculated (Dinh, Sohl-Dickstein, and Bengio 2016). Based on this observations, Dinh et al. propose *affine coupling layers*, a series of simple bijections that are easy to compute, but depend on a remainder of the input x in a complex way (Dinh, Sohl-Dickstein, and Bengio 2016).
2. By exploiting local correlation structure of images, Dinh et al. propose two masking techniques: checkerboard and channel-wise (Dinh, Sohl-Dickstein, and Bengio 2016). This also allows coupling layers to affect all components of the input (Dinh, Sohl-Dickstein, and Bengio 2016).
3. Finally, by implementing squeeze operations, where an input tensor of size $C \times H \times W$ is transformed into a tensor of size $C \cdot 4 \times \frac{H}{2} \times \frac{W}{2}$, effective trading can happen between spatial size and the number of channels can happen (Dinh, Sohl-Dickstein, and Bengio 2016).

2.3 Generative Adversarial Networks - Brisk Dive

Generative adversarial networks are a deep-learning based approach to generative modelling. They were first described in (Goodfellow et al. 2014). The model consists of two submodels - a generator and a discriminator. The task of the generator is to 'fool' the discriminator, who is tasked with distinguishing 'fake', or generated data from the real data.

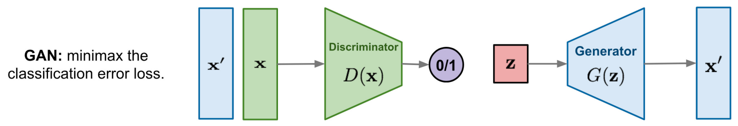

Figure 2: The architecture of GANs. Adapted from (Weng 2018).

More formally, $D(x)$ is a discriminator network, with x being the input. $D(x)$ outputs a probability that x comes from a training data rather than from the generator. In other terms, the $D(x)$ can be thought of as a traditional binary classificator, with the output of $D(x)$ being low when the input was generated, and high when the input is from the testing data.

The generator function, $G(z)$, maps the latent space z to the real data space x. The goal of the generator function is to estimate a real data's probability p_{data} so that it can generate new samples from the p_g.

The training process involves training the $D(x)$ to maximize the probability of correctly assigning the labels to training and generated data. Simultaneously, $G(z)$ is trained to minimize $\log(1 - D(G(z)))$. Therefore, both models are playing a game with a value function $V(D, G)$:

$$\min_G \max_D V(D, G) = E_{x \sim pdata(x)}[\log D(x)]$$

Deep convolutional generative adversarial networks, or DCGANs, were proposed by (Radford, Metz, and Chintala 2015), and explicitly use convolutional and convolutional-transpose layers.

3 Related works

3.1 GANs for Anime Face Generation

GANs are extensively used for generative modelling tasks. At the time of writing, several variations of the GAN architecture were used for anime face generation. For the purpose of this paper, two approaches with the subjectively best generative results were selected - StyleGAN 1/2 adapted to anime face generation by (Branwen 2021b) and DRAGAN (Jin et al. 2017).

StyleGAN, proposed by (Karras, Laine, and Aila 2018), is a GAN with an alternative generator architecture. Instead of providing latent space z once as an input to the generator, it is first mapped to an intermediate space W and is used to control the generator (see Figure 1 in (Karras, Laine, and Aila 2018)). Additionally, noise (Gaussian) is applied at each layer of the convolution, instead of being an implicit input. StyleGAN versions 2 and 3, released in 2020 (Karras, Aittala, Hellsten, et al. 2020) and 2021 (Karras, Aittala, Laine, et al. 2021), improve the stability of the model training and the internal representation.

While StyleGAN was not initially created to generate novel anime faces, (Branwen 2021b) has demonstrated excellent performance of a trained StyleGAN versions 1 and 2 on a custom dataset. Additionally, a further modified version, dubbed StyleGAN2-ext, was used to generate full images (not just the faces of anime characters)(Branwen 2021b).

Deep Regret Analytical GAN (DRAGAN) by (Jin et al. 2017) introduced labels to GAN architecture, as well as used gradient penalty scheme DRAGAN proposed by (Kodali et al. 2017). While their model underperforms in qualitative results in comparison to (Branwen 2021b), the authors of the study propose a different way to create a dataset - particularly by categorising input images by the release date of the associated product (see Figure 5 of (Jin et al. 2017)). By controlling for the release date during data collection, one may achieve a better-quality, more uniform dataset without variations described in 3.2 and (Jin et al. 2017).

Additionally, outside of the area of research, a generative model of unknown GAN architecture was discovered. Model is created by Preferred Networks, Inc., a Japanese company. The model supports face and full-body character generation, with subjectively impressive results. As explained in the 1.1, the author expects more commercial options to follow as the genre of anime becomes more and more ubiquitous.

Normalizing flows are not popular for anime face generation. Several reasons can be provided for a lack of publicly available implementations of normalizing flows for this task:

1. Relative unpopularity/unfamiliarity with normalizing flows - while a Google Scholar search of 'generative adversarial network/s' has returned over 21,000 results, a similar search has returned only about 4,000 results for 'normalizing flow/s'[1]. While not fully representative and certainly not as exhaustive as a full literature review, the results still could indicate relative unfamiliarity with normalizing flows.
2. Difficulty in training - particularly the size of the model. As reported by (Branwen 2021a), the size of a Glow model was about ~ 4.2 GB. This can pose difficulty, particularly while training on consumer-grade hardware or in Google Colaboratory[2]. Additional details on training issues are described in 4.2.
3. Their relatively poor performance in both quantitative and qualitative metrics. For quantitative metrics, see Table 1 in (Bond-Taylor et al. 2021). For quantitative results, one can examine the related sections in (Dinh, Sohl-Dickstein, and Bengio 2016) or (Kingma and Dhariwal 2018). While the analysis of qualitative results is subjective by nature, the results are still visibly worse than state-of-the-art GANs (such as StyleGAN 3 (Karras, Aittala, Laine, et al. 2021) or similarly performing models).

Finally, during the research for this paper, the author has found a directory of animation- and anime-related research (see (Hasegawa 2022)), which can be helpful in identifying other papers/repositories that are centered around anime face generation.

3.2 Current Issues with Anime Face Generation

There are several issues that impede the progress on the novel anime face generation.

The first of those is the wide variation in style and quality of the available anime face images. In particular, anime - like any art style - has matured and changed over the years, meaning that particular facial features (such as eyes, head shape, etc.) have changed. This can introduce noise into the dataset, decreasing the performance of the models by resulting in artifacts. Potential solutions include utilizing automated methods to estimate attributes of the images in the dataset, allowing for finer control over sample selection and enabling conditioned generation. This issue was originally noted in (Jin et al. 2017).

Secondly, there is a lack of representativeness in the datasets, which biases the models against people of color. Anime, which started as a domestic product of Japan and a unique phenomenon inside it, has always depicted characters of Asian descent. However, as anime internationalizes, the lack of diversity within the datasets can create problems to the generative models, particularly in style-transferring tasks. The problem is not easily solvable, too, as each anime face is a human-generated artifact. The possible solutions include increasing the representativeness of the samples by either commissioning a larger number of characters of color to be added to the genre or by augmenting the datasets by using novel images from models trained on datasets with only characters of color.

Finally, the quality of datasets can be varied, affecting overall performance of the models. While there are publicly available databases of anime characters, they can vary in quality of art and the target resolution (with some images compressed for better web performance). During research for

[1]Search was done with two keywords, either (”normalizing flows” OR ”normalizing flow”) or (”generative adversarial networks” OR ”generative adversarial network”), with results before 2021 filtered out on August 30, 2022.

[2]The author has experienced crashes when trying to train the Glow model on a free tier of Google Colaboratory due to the limited RAM/VRAM available.

this project, an approach was identified by (Branwen 2020), where Danbooru[3] image board was used for the dataset creation. In (Jin et al. 2017), Getchu was used. Finally, approaches overall differ across the field, as exemplified by (Hasegawa 2022) list of research articles: some articles re-use the datasets already available for consistency and reproducibility, while others create custom datasets (which, when lacking specific instructions on how to obtain make it harder to draw direct comparisons).

3.3 Future Outlook

With the increase of popularity and public acceptance of anime, it is expected that the novel anime face generative models are going to be further developed. While fairly good results in still image generation and style transfer for both still images and videos are achieved by commercially available products, the issues of high variativity of style and the lack of representativeness in the datasets are the major roadblocks to increasing the quality of generated images and bridging the gap between anime face and regular human face generation.

4 Dataset & Implementation Notes

4.1 Dataset

The dataset was downloaded from Kaggle (Churchill and Chao 2019). The dataset consists of 63,632 anime faces scraped from Getchu.com, a Japanese online database of anime-related CDs, games, etc. The reason for using Getchu for the dataset could be because of each page containing a profile for the characters of any given game, with a small, background-free avatar. Scraping those avatars is what allows the creation of the dataset.

While Getchu is a good database for the easy, quick dataset collection, it has problems, such as varying style of the anime faces (see 3.2).

Of these 63 thousand images, the author further removed all images with a low resolution (less than 64 pixels in any dimension). This was done to improve the quality of the model training. The script for image pruning used by the author is available on the Github repository for the project.

Images were transformed to a 64 pixels by 64 pixels resolution, center-cropped and (in the case of DCGAN) normalized with mean and standard deviation of 0.5 across all channels. For RealNVP, logit transform on top of normalization with dequantization was used.

4.2 Implementation in PyTorch & Training Log

The implementation of all models was done in Python 3.8. For the RealNVP implementation, the code from Fangzhou Mu (Mu 2019) was adapted to load and train on the selected dataset. The RealNVP model was trained with a variety of hyperparameters to test training time and associated performance. For additional details see accompanying Jupyter notebook.

In terms of multi-scale architecture described in (Dinh, Sohl-Dickstein, and Bengio 2016), two were tested: first one for CIFAR-10 (squeeze)

Because of the limitations of the Google Colaboratory environment[4], the training was done on a subset of randomly selected 100 batches of 64 images per batch - this ensured that the Random Access Memory (RAM) and Video-RAM limits were not exceeded.

Additionally, again due to the limitations of the Google Colaboratory environment, training was done in 50 epochs at a time[5]. After each training/validation run of 50 epochs, the model was saved and loaded again to be executed for another 50 epochs.

[3]Note: this website contains not safe for work (NSFW) images; refer to (Branwen 2020) for more information on filtering.

[4]At the time of completion of this project, the author did not possess a discrete CUDA-enabled video card, so they couldn't do the training locally.

[5]Google Colaboratory has a limit on continuous execution in free tier; while precise limits are not published by Google, the author observed about an hour to two hours of continuous execution at a time.

For DCGAN, an implementation by (Inkawhich 2018) was adapted to the dataset. Hyperparameters were set to the ones in the original implementation (see (Inkawhich 2018) for details). The DCGAN was trained over 500 epochs in a single run on Google Colaboratory. The training time for both RealNVP with varying parameters and DCGAN are shown in Table 1.

While Glow (Kingma and Dhariwal 2018) implementation in PyTorch by (Amersfoort and Carr 2019) was used to train a model, the general problems with training instability, extreme RAM/VRAM requirements and relatively poor performance let to the abandonment of the model and subsequent removal from the codebase. However, the generated images are available in the Github repository for the informative purposes. The samples for the other trained models, as well as training hyperparameters are available in the accompanying Jupyter notebook.

5 Discussion & Closing Remarks

5.1 Study Limitations

There are several study limitations:

1. The dataset used in this study is of questionable quality - while the images are background-free (reducing the noise within the dataset), the anime faces themselves are of varying style. This increases data variance and reduces the quality of the generated images (as the model cannot sufficiently learn the detail and instead relies on texture filling to generate images).
2. The Google Colaboratory environment has usage limits and is not very suitable for the task at hand. Using a paid tier of the Google Colaboratory or running the project locally on a machine with the sufficient computing power (in most of the papers and repositories, an Nvidia GTX 1080 TI was used) would be better.
3. While the PyTorch implementations used in this project were as close to the originals as possible, bugs exist (with the high probability introduced by the author). Code cleanup and further testing would be sufficient in making sure that the models train properly.

5.2 Further Research Opportunities

Despite aforementioned study limitations, normalizing flows still show further untapped potential. The author believes that the dataset selection remains a crucial roadblock in the anime face generation problem, and proposes that the future studies be based on a manually-curated dataset, rather than scraped from the publicly available databases.

5.3 Conclusion

This paper introduced normalizing flows for generative modelling, explained the task of the anime face generation, highlighted some of the issues that exist in the currently available datasets, showed the hyperparameters' influence on training time and performance of RealNVP (Dinh, Sohl-Dickstein, and Bengio 2016), and evaluated quantitative and qualitative performance in comparison to DCGAN (Radford, Metz, and Chintala 2015).

6 Acknowledgements

The author would like to thank Eshant English, Researcher/PhD Candidate at the Hasso-Plattner Institute, for the project supervision and continued guidance during the project.

References

[1] J. van Amersfoort and C. Carr. *Simple, extendable, easy to understand Glow implementation in PyTorch*. June 2019. URL: `https://github.com/y0ast/Glow-PyTorch`.

[2] B. Bond-Taylor et al. "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). DOI: 10.1109/tpami.2021.3116668. URL: https://doi.org/10.1109/tpami.2021.3116668.

[3] G. Branwen. *Anime Crop Datasets: Faces, Figures, & Hands*. Aug. 2020. URL: https://www.gwern.net/Crops.

[4] G. Branwen. *Anime Neural Net Graveyard*. Jan. 2021. URL: https://www.gwern.net/Faces-graveyard.

[5] G. Branwen. *Making Anime Faces With StyleGAN*. Jan. 2021. URL: https://www.gwern.net/Faces.

[6] S. Churchill and B. Chao. *Anime Face Dataset*. 2019. DOI: 10.34740/KAGGLE/DS/379764. URL: https://www.kaggle.com/ds/379764.

[7] L. Dinh, J. Sohl-Dickstein, and S. Bengio. *Density estimation using Real NVP*. 2016. DOI: 10.48550/ARXIV.1605.08803. URL: https://arxiv.org/abs/1605.08803.

[8] A. Elgammal et al. *CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms*. 2017. DOI: 10.48550/ARXIV.1706.07068. URL: https://arxiv.org/abs/1706.07068.

[9] I.J. Goodfellow et al. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

[10] S. Hasegawa. *AwesomeAnimeResearch*. Aug. 2022. URL: https://github.com/SerialLain3170/AwesomeAnimeResearch.

[11] N. Inkawhich. *DCGAN Tutorial*. Aug. 2018. URL: https://github.com/pytorch/tutorials/blob/master/beginner_source/dcgan_faces_tutorial.py.

[12] Y. Jin et al. *Towards the Automatic Anime Characters Creation with Generative Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1708.05509. URL: https://arxiv.org/abs/1708.05509.

[13] T. Karras, M. Aittala, J. Hellsten, et al. "Training Generative Adversarial Networks with Limited Data". In: *Proc. NeurIPS*. 2020.

[14] T. Karras, M. Aittala, S. Laine, et al. "Alias-Free Generative Adversarial Networks". In: *Proc. NeurIPS*. 2021.

[15] T. Karras, S. Laine, and T. Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. DOI: 10.48550/ARXIV.1812.04948. URL: https://arxiv.org/abs/1812.04948.

[16] D.P. Kingma and P. Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. DOI: 10.48550/ARXIV.1807.03039. URL: https://arxiv.org/abs/1807.03039.

[17] I. Kobyzev, S.J.D. Prince, and M.A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 3964–3979. DOI: 10.1109/tpami.2020.2992934. URL: https://doi.org/10.1109/tpami.2020.2992934.

[18] N. Kodali et al. *On Convergence and Stability of GANs*. 2017. DOI: 10.48550/ARXIV.1705.07215. URL: https://arxiv.org/abs/1705.07215.

[19] A. Kosiorek. *Normalizing Flows*. Apr. 2018. URL: http://akosiorek.github.io/ml/2018/04/03/norm_flows.html.

[20] A. Lamb. *A Brief Introduction to Generative Models*. 2021. DOI: 10.48550/ARXIV.2103.00265. URL: https://arxiv.org/abs/2103.00265.

[21] F. Luan et al. *Deep Photo Style Transfer*. 2017. DOI: 10.48550/ARXIV.1703.07511. URL: https://arxiv.org/abs/1703.07511.

[22] E. Margolis. *The Dark Side of Japan's Anime Industry*. July 2019. URL: https://www.vox.com/culture/2019/7/2/20677237/anime-industry-japan-artists-pay-labor-abuse-neon-genesis-evangelion-netflix.

[23] F. Mu. *PyTorch implementation of realNVP*. Jan. 2019. URL: https://github.com/fmu2/realNVP.

[24] A. Radford, L. Metz, and S. Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. DOI: 10.48550/ARXIV.1511.06434. URL: https://arxiv.org/abs/1511.06434.

[25] D. Saxena and J. Cao. *Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions*. 2020. DOI: 10.48550/ARXIV.2005.00065. URL: https://arxiv.org/abs/2005.00065.

[26] L. Weng. “Flow-based Deep Generative Models”. In: *lilianweng.github.io* (2018). URL: https://lilianweng.github.io/posts/2018-10-13-flow-models/.