A VAE-GAN Model for Generating Mahjong Images

Bo Liu
Department of Statistical Science
Duke University
Durham, NC 27705
bl226@duke.edu

Linlin Li
Department of Statistical Science
Duke University
Durham, NC 27705
linlin.li434@duke.edu

Abstract

Through the project, we learned variational autoencoder (VAE) that extends the normal autoencoder with an ability to generate new objects. On basis of this, we knew the limitation on VAE and learned how generative adversarial networks (GAN) were combined with VAE to generate better images. We applied the algorithm on an image set of Mahjong tiles. The training results revealed some success but also showed some difficulties in the training process.

1 Background

1.1 variational autoencoder

The autoencoder [7] is a class of embedding methods implemented via a neural network. The key point is that a low dimensional layer, which is called a "bottleneck", is in the middle of the network. Thus, this neural network can be viewed as a combination of an encoder network and a decoder network. The encoder network maps original data to the output of the bottleneck, which represents some of the features that the original data possess. The decoder network, in the opposite, should be able to recover the original data as much as possible through this low dimensional vector. The target of the neural network is to minimize the difference between the input and the output.

The variational autoencoder [3, 6] further extends the autoencoder with the ability to generate new but similar outputs. This is achieved by adding randomness in the network. The bottleneck layer is modified that it now consists of two low dimensional vectors, namely a mean vector and a standard deviation vector. Vectors randomly sampled from the mean and standard deviation are given to the decoder to reconstruct the original data. Different from normal autoencoders that embed training data onto discrete points, variational autoencoders represent these data by probabilistic distributions. Therefore, we might be able to generate new objects through a random variable that follows such distributions.

Suppose the encoder maps a data sample $\boldsymbol{x}$ to a low-dimensional vector $\boldsymbol{z}$ and the decoder maps it back to $\widetilde{\boldsymbol{x}}$ in the data space. We expect that the reconstruction error to be small, and impose a regularization term on latent vector $\boldsymbol{z}$. The regularization term can be viewed as imposing a normal $\mathcal{N}(0, \mathbf{I})$ prior.

Let $\boldsymbol{z} \sim q(\boldsymbol{z} \mid \boldsymbol{x})$, $\widetilde{\boldsymbol{x}} \sim p(\boldsymbol{x}|\boldsymbol{z})$,

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{latent}} = -\mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})] + D_{\text{KL}}(q(\boldsymbol{z} \mid \boldsymbol{x})||p(\boldsymbol{z})),$$

where D_{KL} is the Kullback-Leibler divergence.

1.2 Generative adversarial network

A major shortcoming of VAE is that it measures the similarity between input and output in element-wise measures. In the topic of image reconstruction, an element-wise metric would measure the

Project Report for STA561 Probabilistic Machine Learning, Duke University.

pixel-by-pixel error between two images. Although by minimizing the pixel-wise error the network will generate quite similar reconstructions, it is not necessary for generating images that are similar to human eyes. For example, a human would barely percept a small translation on the image, while the pixel-wise error may become very large. Such a strict constraint greatly limits the variety of images it may produce and the neural network often takes longer to train.

A softer way to measure sample similarity is through a generative adversarial network (GAN) [2]. Instead of using a hard metric, GAN learns to distinguish generated images from actual training images. A GAN is composed of a generator network and a discriminator network. The generator takes a latent vector $\boldsymbol{z}$ as input and generates an object $\widetilde{\boldsymbol{x}} = \mathrm{Gen}(\boldsymbol{z})$ in the data space. Then, the discriminator assigns probability $y = \mathrm{Disc}(\widetilde{\boldsymbol{x}}) \in [0, 1]$ on each generated object. As the word *adversarial* shows, the object of GAN is to find the best binary classifier to discriminate from generated data and real data from the sample, while simultaneously training the generator to fit the true data distribution.

Define the discriminator loss for real sample $\boldsymbol{x}$ and fake sample $\mathrm{Gen}(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})$ respectively as

$$\mathcal{L}_{\text{disc}}^{\text{real}} = \log(\mathrm{Disc}(\boldsymbol{x})), \quad \mathcal{L}_{\text{disc}}^{\text{fake}} = \log(\mathrm{Disc}(\mathrm{Gen}(\boldsymbol{z}))).$$

We maximize this quantity with respect to Disc, and minimize it with respect to Gen.

2 Combination of VAE and GAN

2.1 Methodology

Larsen et al. [4] proposed an autoencoder which combines VAE and GAN to produce better image samples. Since the decoder network in VAE and the generator network in GAN both map a latent vector to an image, the combined network will share the two.

2.2 Algorithm

Some modifications should be adjusted to gain better training quality in practice [4]. First, the networks should not be trained simultaneously, since not all parameters should be updated to minimize the combined loss. Also, it is observed better not to backpropagate the error from GAN to the encoder network of VAE. Second, proper weighting should be done between VAE and GAN. Third, it performs better if we use samples from the encoder in addition to train the GAN.

The algorithm that we use is shown in Algorithm 1, adopted and modified from Larsen et al. [4].

Algorithm 1: Training the VAE/GAN model

$\boldsymbol{\theta}_{\text{Enc}}, \boldsymbol{\theta}_{\text{Gen}}, \boldsymbol{\theta}_{\text{Disc}} \longleftarrow$ initialize network parameters;
while *not converge* **do**
  $\boldsymbol{X} \longleftarrow$ random mini-batch from dataset;
  $\boldsymbol{Z} \longleftarrow \mathrm{Enc}(\boldsymbol{X})$;
  $\mathcal{L}_{\text{latent}} \longleftarrow D_{\text{KL}}(q(\boldsymbol{Z} \mid \boldsymbol{X}) || p(\boldsymbol{Z}))$;
  $\widetilde{\boldsymbol{X}} \longleftarrow \mathrm{Dec}(\boldsymbol{Z})$;
  $\mathcal{L}_{\text{recon}} \longleftarrow -\mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X})}[p(\boldsymbol{X} \mid \boldsymbol{Z})]$;
  $\boldsymbol{Z}_{\text{sample}} \longleftarrow$ samples from prior $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\mathrm{I}})$;
  $\boldsymbol{X}_{\text{sample}} \longleftarrow \mathrm{Gen}(\boldsymbol{Z}_{\text{sample}})$;
  $\mathcal{L}_{\text{GAN}} \longleftarrow \log(\mathrm{Disc}(\boldsymbol{X})) + \log(1 - \mathrm{Disc}(\widetilde{\boldsymbol{X}})) + \log(1 - \mathrm{Disc}(\boldsymbol{X}_{\text{sample}}))$;

  $\boldsymbol{\theta}_{\text{Enc}} \xleftarrow{+} -\nabla_{\boldsymbol{\theta}_{\text{Enc}}}(\mathcal{L}_{\text{latent}} + \mathcal{L}_{\text{recon}})$;
  $\boldsymbol{\theta}_{\text{Gen}} \xleftarrow{+} -\nabla_{\boldsymbol{\theta}_{\text{Gen}}}(\gamma \mathcal{L}_{\text{recon}} - \mathcal{L}_{\text{GAN}})$;
  $\boldsymbol{\theta}_{\text{Disc}} \xleftarrow{+} -\nabla_{\boldsymbol{\theta}_{\text{Disc}}} \mathcal{L}_{\text{GAN}}$;
end

3 Experiment

3.1 Mahjong tile images

We apply the algorithm to mahjong tile images from an opensource GitHub repository [8]. A part of this dataset consists of 628 images annotated with respective classes (see Figure 1 for information on these classes). In this experiment, we only used the images corresponding to suits (i.e. from 27 total classes).

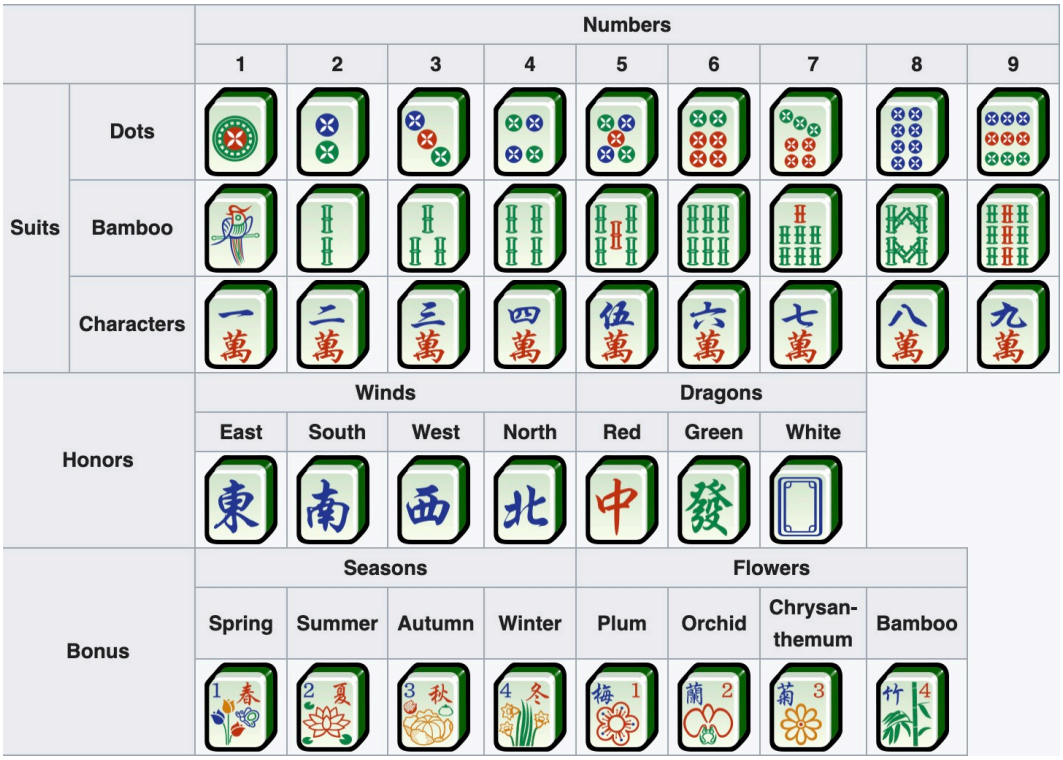


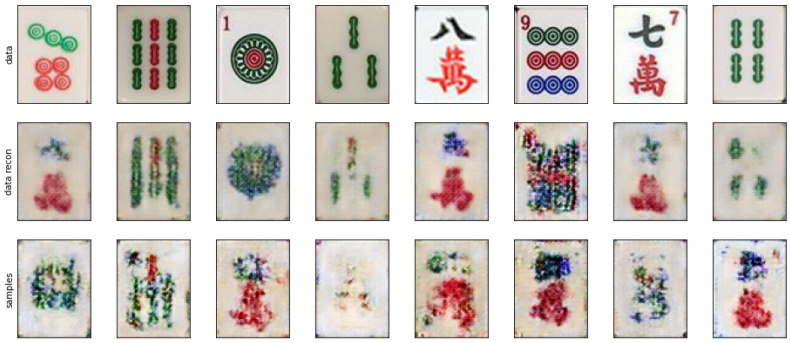
Figure 1: 43 various tile classes of a usual set of Mahjong

Each image is cropped and resized to 80×60 pixels. The format of the images is JPG which has three RGB channels. A total number of 435 images are used. The images were collected from online pictures/photos, so the style of each set of tiles may look slightly different (Figure 2).

Figure 2: Different styles of Bamboo 5 tiles in the dataset

3.2 Architecture

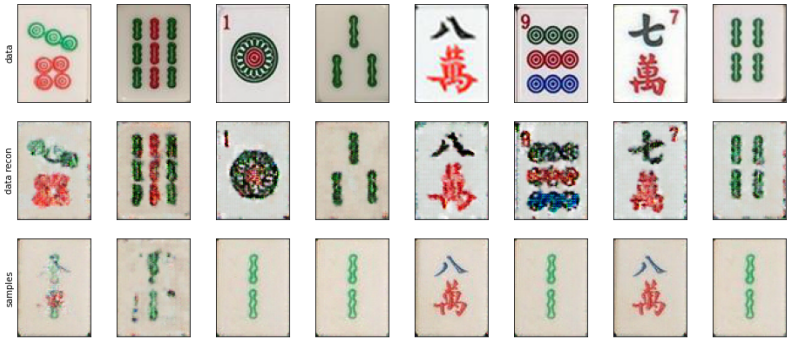
The encoder we use consists of two convolutional layers [(32, 3×3s2), (64, 3×3s2)], a flatten layer and a full connected layer to 128 dimensional mean and 128 dimensional standard deviation. The decoder/generator starts with a full connected layer with 900 neurons which are reshaped into (20,15,64). Three reverse convolutional layers [(64, 3×3s2), (32, 3×3s2), (3, 3×3)] then transform it back to the original shape. The discriminator has the same convolutional layers and flatten layer as the encoder, followed by a full connected layer with 512 neurons and finally a full connected layer to a single output.



(a) Epoch 1000

(b) Epoch 5000

(c) Epoch 10000



(d) Epoch 20000

Figure 3: First row: Random samples from training images. Second row: Reconstructed images. Last row: Images generated from random Gaussian vectors.

3.3 Results

We trained 20,000 epoches with the optimal set of hyperparameters we were able to find. Eight pairs of samples from the training set and the respective reconstructed images are shown to demonstrate the similarity between them. Additional 8 images shown are generated by the generator neural network, starting from a random Gaussian sample from $\mathcal{N}(0, \mathbf{I})$. The results are shown in Figure 3.

From the results, we can see that the VAE-GAN model is able to produce high quality reconstructed images compared to VAE (Figure 4). Also, it generates reasonable images from the true distribution. However, the generator tends to produce the same image when the discriminator fails to identify it. This phenomenon may last until the discriminator is trained enough from sampled and generated data. However, since the sigmoid function which we used as the last layer of generator reaches values close to 0 or 1 very fast, the gradient are almost zero for x not close to 0. This problem of vanishing gradient retards the convergence and the parameters are barely updated.

Figure 4: Images generated by VAE with the same hyperparameters and number of epoches

4 Discussion

VAE-GAN not only improves the performance of image reconstruction, but can also be extended to express semantic concepts using basic arithmetic [4, 5] or supervised tasks like image classification [4]. However, limited by the perfomance of computing devices accessible to us, we were not able to build a neural network capable of these tasks. Nonetheless, our neural network showed that the algorithm for recovering and generating images works surprisingly well even with a shallow network on a CPU machine. We might further attempt to avoid gradient vanishing by combining VAE with WGAN [1]. Also, we may try working on integrating VAE and Conditional GAN (CGAN) as the images have additional label information.

References

[1] M Arjovsky, S Chintala, and L Bottou. Wasserstein gan. arxiv 2017. *arXiv preprint arXiv:1701.07875*, 2019.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, 2014.

[4] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[6] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[7] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[8] Esmond Wong. mahjong-dataset: Computer vision dataset for chinese mahjong tiles, 2019. URL https://github.com/Camerash/mahjong-dataset.