Introduction to Deep learning

Assignment 3

Yuang Yuan

Wei Chen

ACM Reference Format:
Yuang Yuan and Wei Chen. 2022. Introduction to Deep learning: Assignment 3. In *Proceedings of Assignment 3 (Introduction to deep learning).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

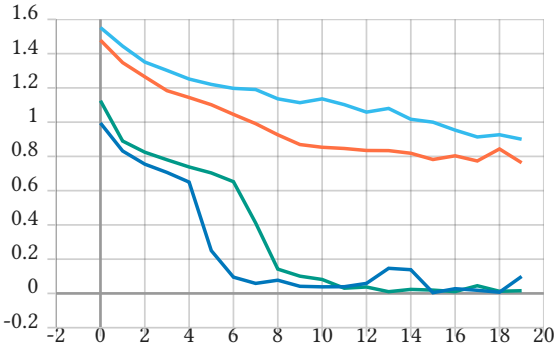
1 Task 1

In Task 1, we explore and develop encoder-decoder recurrent models that translate the result of addition and subtraction of integers between images and texts.

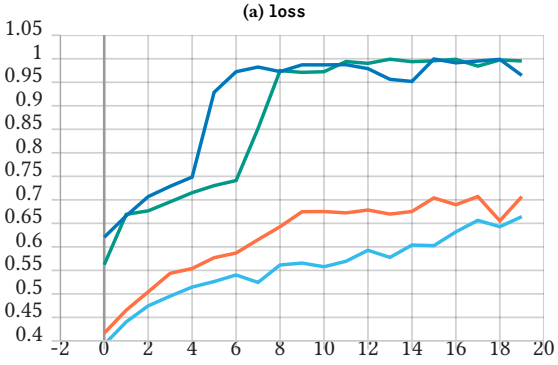
1.1 Sub Task 1: text to text model

In this sub task we experiment with a text to text model that structured as Figure **??**

1.1.1 Different spilts of dataset In this section, we try different spilts of dataset



(a) loss



(b) accuracy

Figure 1: the loss(a) and accuracy(b) curves of text to text model on different spilt percentages of datasets. [5% train - 95% test (light blue), 10% train - 90% test (orange), 50% train - 50% test (green), 80% train - 20% test (dark blue)]

1.1.2 Better model architectures

1.2 Sub Task 2: image to text model

1.3 Sub Task 3: text to image model

2 Task 2: Generative Models

The code in the notebook given is modified partially, where the text of the tutorial part is completely removed and some code related to latent space are added.

2.1 Dataset

The data set used for retraining is *Anime Avatar* [1]. For the source of the data set see the Source of Anime data set or click the next url to download the data set that we convert them into .npy file: Anime.npy

In addition, the image data in the data set look like ...(see Figure 2), where the resolution or size of each image is different from each other initially. We scale each image in the data set to 64×64 to be close to the size of the example data set used in tutorial.

Figure 2: Images in the anime data set [1]

2.2 VAE

2.2.1 Experiments and results of the VAE model retrained The anime Data set is fed into the VAE model for retraining, where the corresponding choice of (hyper)parameters is as shown in the Table 1.

The images generated by the decoder component of the VAE model is as the Figure 3 and 4 shown, which are more blurred than the real images in the data set. Then, in the 20th epoch, the clarity of the images is improved greatly (see Figure 4 (left)), while in the 40th epoch the images produced are also clearer than ever but not obvious (see Figure 4 (right)).

Definition of the (hyper)parameters					
Epoch	Batch size	Learning rate	Optimizer	Latent space dimension	seed
40	4	0.001	Adam	64	1024

Table 1: Selection of the (hyper)parameters of retraining the VAE model

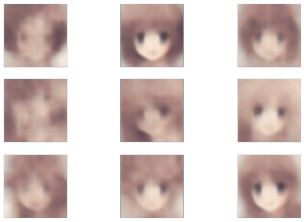


Figure 3: Images generated by the retrained VAE model in the 1st epoch

Figure 4: Images generated by the retrained VAE model in the 20th epoch (left) and the 40th epoch (right)

Figure 5: Two images generated by two random points (top) and ten images produced by the interpolated points on the line between the two random points (bottom), where the random seed is 4

However, the fake images generated in the last epoch are still relatively slightly less clear than the real images.

2.2.2 Interpolation in the latent space of the VAE model

We here try a total of twice for the experiment that randomly picks two points in the latent space of VAE, where the random seed is respectively set to 4 (see Figure 5) and then 999 (Figure 6). These two figures show the process that an image on the left side is as if smoothly transitioned to the other one on the right side.

Figure 6: Two images generated by two random points (top) and the ten images produced by the interpolated points on the line between the two random points (bottom), where the random seed is 999

2.2.3 Other uses for the latent data representations of VAE

One of the uses for the latent data representations of VAE, which can be to customize the style of avatar we want to generate. For instance, we are here interested in whether a character (avatar) with short and red hair can be changed to be with long and red hair simply using vector arithmetic.

The answer is yes. Firstly, we randomly choose 100 points in the latent space (here the random seed is set to 334) and plot a graph including the all images generated by the 100 random points, in a 10-by-10 grid pattern (see Figure 7). Next, We pick the three lists of feature by hand. For example, the indices (starting from 0) of an avatar with short and red hair, and with short and blue hair in this grid pattern can be 17 and 56 respectively. Therefore, a certain number of avatars (here we set the number to 3) are included in each list of hair features. Next, we average the three vectors for each list of hair features. The result we expect is to generate a new avatars by vector arithmetic:

$$\textit{Long red hair} = \textit{short red hair} - \textit{short blue hair} + \textit{long red hair}$$

where each term in the expression above represents an average vector calculated by its corresponding list of hair features.

Figure 7: 10-by-10 VAE-based grid graph generated by 100 random points, where the random seed is 334

As expected, the result corresponds to our assumption (see the first "term" in the Figure 8).

Figure 8: Visualization of the vector arithmetic based on VAE

2.2.4 Working principles of VAE

In general, Variational autoencoder (VAE) and autoencoder are similar in the principle and both include two components: the one is encoder network and the other is decoder network. The main difference between them is that the images generated by the points in the latent space of the autoencoder are probably vacuous and not understandable since it does not represent each latent variable of input as a probability distribution $p(Z)$ (Z denotes the all vectors or points in the continuous latent space).

The chief training process of the autoencoder is:
Firstly, suppose a data set is trained on an autoencoder model with a latent space (or variables) dimension of 32. Then the images will be passed to the encoder that compresses the dimension of the image of, say 4096, into 32. Finally, the decoder decompresses the vectors compressed in the decoder to images with their original dimension (4096). Therefore, the autoencoder is essentially a way of dimension reduction.

As stated before, the images generated by points in the latent space are mostly noise. To make the autoencoder model a generative model, we let Z $N(0, I)$, where I represents a unit matrix, and N is the multivariate normal distribution. Then the mean and log-variance parameters are introduced. The input are passed through the encoder network and then it outputs the parameters (mean μ and variance σ) of the approximate posterior distribution $q_\phi(z|x_i)$. Once the parameters μ and σ of the approximate posterior distribution are obtained, the points z_i in the latent space are fed into the decoder network. Next, we let the decoder network fit the conditional distribution $q_\theta(z|x_i)$ and output the parameters (mean μ' and variance σ') of this distribution. To train this model, back-propagation is needed. However, it is impossible to apply back-propagation to this kind of model architecture. Hence, the reparameterization trick is introduced. we sample the ϵ_i from the N(0, I) and approximate using the expression below:

$$z = \mu + \sigma \cdot \epsilon$$

Finally, the vae based on autoencoders can be trained.

2.3 GANs

2.3.1 Experiments and results of the GANs model retrained

The selection of the (hyper)parameters for retraining the GANs' model is similar to what I do in the experiments about VAE (see Table 2), where the seed is not defined, which means the grid graph generated in each epoch is different from each other

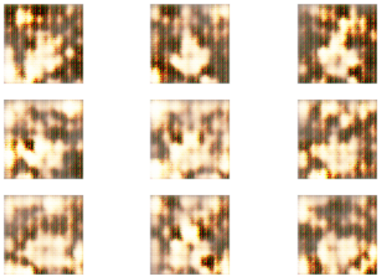


Figure 10: Images generated by the retrained GANs model in about the 10th epoch

Definition of the (hyper)parameters					
Epoch	Batch size	Learning rate	Optimizer	Latent space dimension	seed
400	64	0.001	Adam	128	None

Table 2: Selection of the (hyper)parameters of retraining the GANs' model

It can be seen from the Figure 9 that in the 1st epoch, the resulting images generated by the generator hardly show anything obvious, whereas, in the 3rd epoch, the extremely blurred profile emerges.

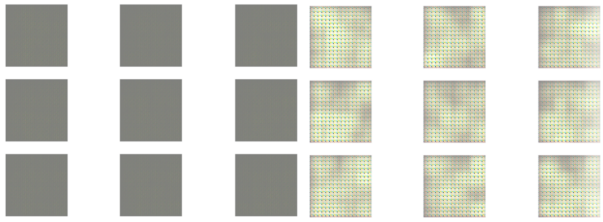


Figure 9: Images generated by the retrained GANs model in the 1st epoch (left) and 3rd epoch (right)

Obviously, GANs is different from VAE whose decoder component is capable of outputting images that normal human can recognize in the 1st iteration even though they are blurred in general. For this reason, we adjust up the epoch.

In about the 10th epoch, the lineament is not as blurred as we see in the 1st or 3rd epoch, and the feature of the face can be vaguely identified (see Figure 10).

Then in the 20th epoch or so, color is added to the face and hair soon (Figure 11). Also, the features, such as eye, mouth, and nose is formed gradually.

We experiment with the GANs model for 400 epochs, but it appears to have already reached its best performance in about the 200th epoch (Figure 12 (left)). Also, we found that the clarity of the images the model of GANs generates is evidently better than those generated by the decoder of VAE.

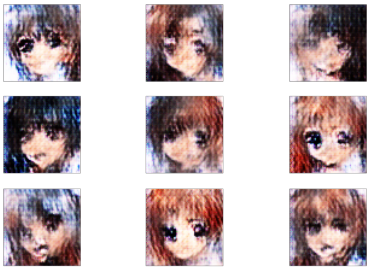


Figure 11: Images generated by the retrained GANs model in about the 20th epoch

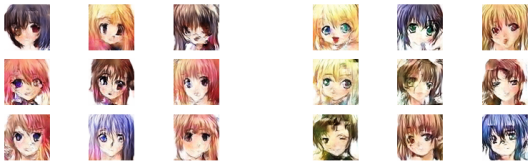


Figure 12: Images generated by the retrained GANs model in about the 20th epoch (left) and the 400th epoch (right).

Figure 13: Two images generated by two random points (top) and ten images produced by the interpolated points on the line between the two random points (bottom), where the random seed is 321

Figure 14: Two images generated by two random points (top) and ten images produced by the interpolated points on the line between the two random points (bottom), where the random seed is 5

Figure 15: 10-by-10 GANs-based grid graph generated by 100 random points, where the random seed is set to 334

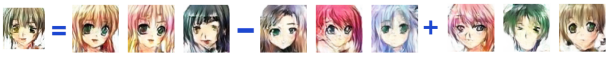


Figure 16: Visualization of the vector arithmetic based on GANs

2.3.2 Interpolation in the latent space of the GANs model

What we do here is exactly the same as we do in the VAE experiment: generate two random points (here the seed is 321) in the latent space and a function is defined to calculate ten points on the line between the two random points. Similarly, a total of twice experiment is performed (Figure 13 and 14).

By interpolation, the process that images morph from one avatar to another is implemented.

2.3.3 Other uses for the latent data representations of GANs

Due to the good clarity of the images generated by GANs, almost all features (e.g., mouth and eye) can be recognized by the naked eye. We, therefore, want to create a man's face with mouth opened by vector arithmetic. For convenience, we also pick three images (or more precisely points) for each list of mouth features, from the grid graph including one hundred avatar images (Figure 15), where the mouth features are females with mouth opened, females with mouth closed, and males with mouth closed.

The expression to compute the result(a man with a mouth opened) that we expect is:

$$female\ mouth\ opened - female\ mouth\ closed + male\ mouth\ closed$$

The visualization of the expression is shown as the Figure 16. It can be seen that the first term (result) is what we expect (a young boy that opens mouth).

2.3.4 Working principles of GANs

The models of GANs also generally have two components: generator and discriminator, where the generator is similar to the decoder

of VAE and the network of the architecture of the discriminator is similar to that of the encoder of VAE. While GANs and VAE are similar to a great extent, the difference is also obvious. The models of GANs represent a zero-sum game between two players, a generator and a discriminator. The generators try to generate images that are mistakenly thought to be the real images by the discriminator, while the discriminators need to determine which of the input images belong to the real dataset and which are the images generated by the generator.

When training on the model of GANs, generator and discriminator need to be trained separately. For example, when training on the generator model of GANs, we need to make the parameter matrix of the discriminator network untrainable and vice versa.

2.4 Conclusion

Essentially, VAE and GANs are both generative models and their patterns are both based on random noise. In addition, for both VAE and GANs when modeling the distribution, the measure of the difference between two probability distributions is needed.

A main difference between VAE and GANs is the loss function. VAE tries to maximize the ELBO, while GANs needs to minimize Jensen-Shannon Divergence. We believe that this difference is exactly the key reason why the models of GANs generally generate more photo-realistic images than the models of VAE. Take an example of the face avatar data set we use here, the images generated by VAE are more blurred than those generated by GANs, while the images generated by GANs sometimes have crooked mouths or noses.

Furthermore, VAEs are easier to train, whereas the models of GANs needs more epochs or time to generate photo-realistic images.

References

[1] Spencer Churchill. 2019. Anime Face Dataset.