Introduction to Deep Learning Assignment 2

Group 53

Chenyu Shi (s3500063); Shupei Li (s3430863); Shuang Fan (s3505847)

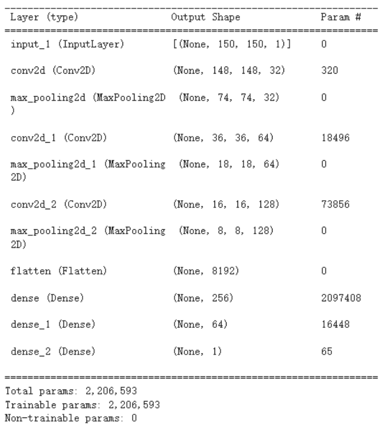
November 21, 2022

Task 1

Task 2

2.1 Regression

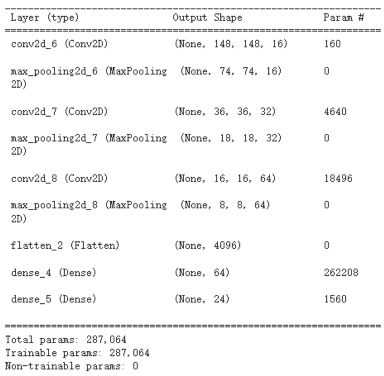
The regression model structure is shown in Figure 1. And its corresponding results are shown in Table 1. Because this is a regression task, we use MSE as loss function and Adam as optimizer. This model takes 2207593 trainable parameers, which is relatively high comparing to other models in task2. However, the final common sense loss is still a little bit high, which is 0.7053 hour(about 42.3 minutes). This is mainly caused by the problem of this kind of labels. Representing time in this way doesn't obey common sense. For example, 11:55 will be represented as 11.917 while 0:05 will be represented as 0.0833. Even though in common sense, the difference between 11:55 and 0:05 is only 10 minutes, their reformulated labels 11.917 and 0.0833 have a very high MSE. Therefore, this regression model has a limitted final performance.



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 1)]	0
conv2d (Conv2D)	(None, 148, 148, 32)	320
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 36, 36, 64)	18496
max_pooling2d_1 (MaxPooling 2D)	(None, 18, 18, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73856
max_pooling2d_2 (MaxPooling 2D)	(None, 8, 8, 128)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 256)	2097408
dense_1 (Dense)	(None, 64)	16448
dense_2 (Dense)	(None, 1)	65

Total params: 2,206,593
Trainable params: 2,206,593
Non-trainable params: 0

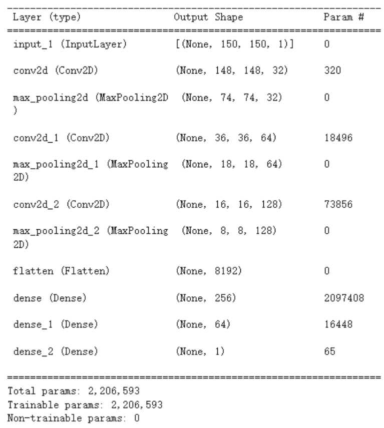
Figure 1: Regression model



Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 148, 148, 16)	160
max_pooling2d_6 (MaxPooling 2D)	(None, 74, 74, 16)	0
conv2d_7 (Conv2D)	(None, 36, 36, 32)	4640
max_pooling2d_7 (MaxPooling 2D)	(None, 18, 18, 32)	0
conv2d_8 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_8 (MaxPooling 2D)	(None, 8, 8, 64)	0
flatten_2 (Flatten)	(None, 4096)	0
dense_4 (Dense)	(None, 64)	262208
dense_5 (Dense)	(None, 24)	1560

Total params: 287,064
Trainable params: 287,064
Non-trainable params: 0

Figure 2: Classification model



Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 1)]	0
conv2d (Conv2D)	(None, 148, 148, 32)	320
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 36, 36, 64)	18496
max_pooling2d_1 (MaxPooling 2D)	(None, 18, 18, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73856
max_pooling2d_2 (MaxPooling 2D)	(None, 8, 8, 128)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 256)	2097408
dense_1 (Dense)	(None, 64)	16448
dense_2 (Dense)	(None, 1)	65

Total params: 2,206,593
Trainable params: 2,206,593
Non-trainable params: 0

Figure 3: Multi-head model

Table 1: Results and Algorithm Comparison

Model	Common Sense Loss(hours)
Regression	0.7053
Classification(24 classes)	0.6206
Classification(72 classes)	0.9517
Classification(720 classes)	3.0309
Multi-head	0.8634

2.2 Classification

The regression model structure is shown in Figure 2. And its corresponding results are shown in Table 1. Thereinto, model structures of 24 classes, 72 classes and 720 classes are only different in last output layers. Because this is a multi-class classification task, we use cross entropy as loss function adn Adam as optimizer.

And we can see, although classification model uses much fewer trainable parameters, the final result for 24 classes is better than regression model with a performance of 0.6206 hours(about 37.2 minutes). However, when we increase the number of categories, the performance of model become poorer. And the model of 720 classes even didn't coverge and has a very high common sense loss. That's because this kind of label representation also has its problems. The first problem is that this label representation cannot measure "how much difference two categories have". For example, in 24 classes model, 0:01 and 0:31 are in different categories, and 0:01 and 6:00 are also in different categories. Although the common sense loss between 0:01 and 6:00 are much larger than that between 0:01 and 0:31, the cross entropy loss function in classfication task will give them same loss values. The second big problem is that it's hard for us to balance sampling interval and samples number. If we have large sampling interval along with a large samples number, the model can get well trained because there are sufficient training data in each class, but the common sense loss within each class becomes rather high. For example, in 24 classes model, 0:01 and 0:29 will be classified into same class even they are almost half an hour apart. On the contrary, if we have small sampling interval along with a small samples number, the common sense loss within each class becomes rather small, but the model itself cannot be well trained due to lack of training data for each class. For example, in 720 classes model, each class will only have 25 pictures for training, validation and test set. That's not a sufficient number for training CNN in a regular way.

Task 3

3.3 Datasets

We use two datasets in Task 3. Firstly, we explore the performance of different model architectures and the effects of different parameters with MNIST data. After that, we leverage the power of generative models on Butterfly & Moth data.

We directly call Tensorflow API to download the MNIST dataset. However, the original dataset is also available on https://deepai.org/dataset/mnist. MNIST dataset contains 70,000 grayscale images ($28 \times 28 \times 1$), whose content is handwritten numbers.

Butterfly & Moth is an open source dataset on Kaggle. There are 13,639 RGB images ($224 \times 224 \times 3$) composed of 100 butterfly or moth species. Link of the dataset is https://www.kaggle.com/datasets/gpiosenka/butterfly-images40-species?resource=download.

3.4 Experimental Set-up

All experiments are deployed on two servers. Server 1 has an Intel(R) Xeon(R) Platinum 8358P CPU and a RTX A5000 GPU, while server 2 has an Intel(R) Xeon(R) E5-2680 v4 CPU and a TITAN Xp GPU. Table 2 summarizes the hyperparameter values in the experiments. The following describes the process of our experiments.

Table 2: Hyperparameter Settings

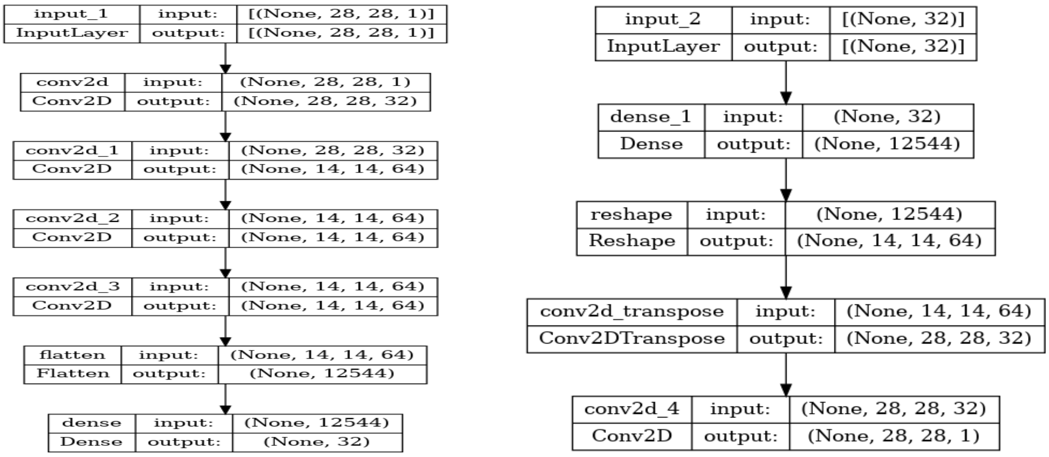
Parameter	Value	Meaning
cae_latent_dim	32	Dimensions of the latent space in CAEs.
cae_epoch	10	The number of training epochs in CAEs.
vae_latent_dim	32	Dimensions of the latent space in VAEs.
vae_epoch	20 (MNIST) / 100 (Butterfly & Moth)	The number of training epochs in VAEs.
gan_latent_dim	256	Dimensions of the latent space in GANs.
gan_epoch	20 (MNIST) / 250 (Butterfly & Moth)	The number of training epochs in GANs.

3.4.1 MNIST

We modify the model architecture to decrease the model complexity and match the data better. Specially, we build the basic convolutional network with four Conv2D layers and construct the basic deconvolutional network with one Conv2DTranspose layer. Figure 4 illustrates our model settings. There is no need to resize the images due to the modification of the model.

3.4.2 Butterfly & Moth

We rescale the images to $64 \times 64 \times 3$ and directly apply the model architecture provided in the notebook when working with Butterfly & Moth data.

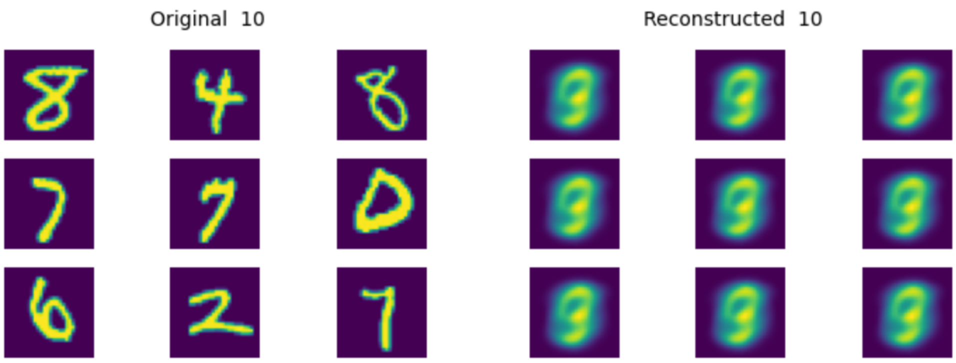


(a) Convolutional network **(b)** Deconvolutional network

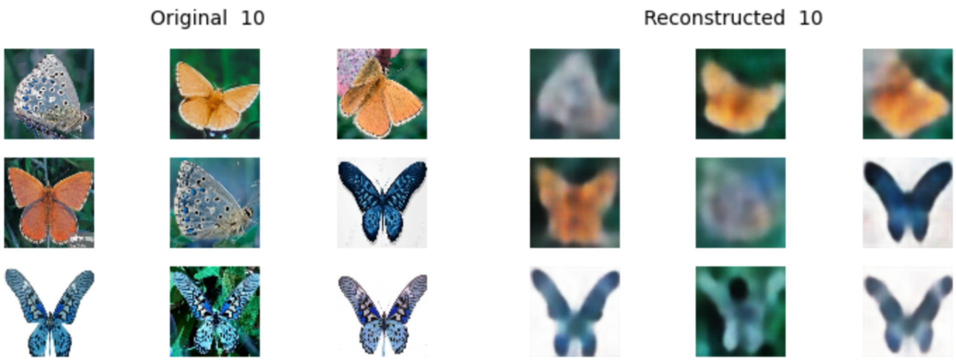
Figure 4: Model Structure

3.5 Results

CAEs. Figure 5 shows the reconstructed images from CAEs. It is easy to see that CAEs have captured the main features in the original images.



(a) MNIST: Original Images **(b)** MNIST: Reconstructed Images



(c) Butterfly & Moth: Original Images **(d)** Butterfly & Moth: Reconstructed Images

Figure 5: Results of CAEs

VAEs. We explore the learned latent space with linear interpolation technique. Firstly, we sample a point from the latenty space by generating its coordinates from a standard normal distribution. Then, we change one or two coordinates along the straight line in the latent space, while keep other coordinates unchanged. For MNIST dataset, we apply linear interpolation on the 10th and 27th coordinates simultaneously, which are related to the shape of the number. Figure 6 shows the visualization results. As for Butterfly & Moth data, we linearly interpolate the 6th and 29th coordinates, whose outputs are presented in Figure 7. According to Figure 7, the 6th coordinate is related to the color of wings, while the 29th coordinate is concerned with the width of wings.

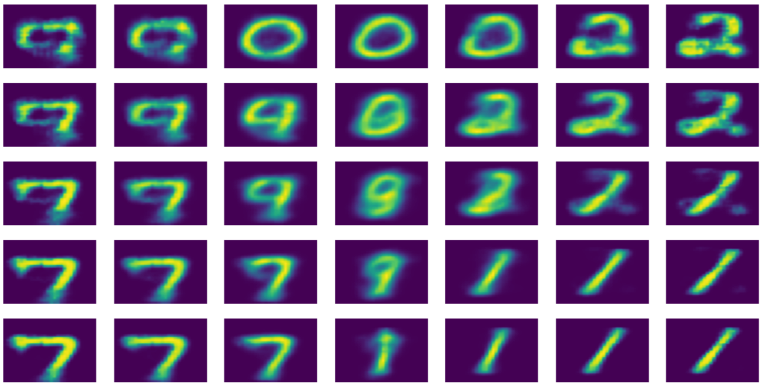


Figure 6: VAEs: MNIST Data

Figure 7: VAEs: Butterfly & Moth Data

GANs. We visualize the outputs of linear interpolation in the same way as VAEs. We change the 1st, 120th, and 126th coordinates for MNIST data to obtain different numbers. Figure 8 shows the generated images. For Butterfly & Moth data, we change the 40th coordinate, which is related to the color and the posture of the butterfly. Figure 9 shows the generated butterfly.

Figure 8: GANs: MNIST Data

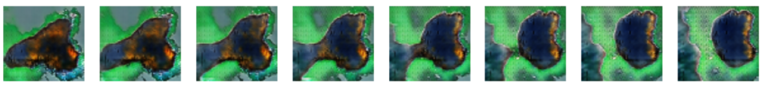


Figure 9: GANs: Butterfly & Moth Data

3.6 Discussion: Model Comparison

CAEs are autoencoders that apply the CNN architecture to compress the input images into a lower dimensional latent space and reconstruct the images from the learned latent space. Using an encoder/decoder structure enables CAEs to capture as much information about data as possible, even without labels. However, CAEs are deterministic. That means there is a one-to-one relationship between the input and output in CAEs. Therefore, CAEs can't generate new samples. Based on the architecture of the traditional autoencoder, VAEs introduce randomness into the model by assuming a prior distribution of latent space and infering the posterior distribution during the training process. In most cases, we choose the standard Gaussian distribution as prior distribution, which helps the latent space to be complete and continuous. The probabilistic nature allows VAEs to generate new images from random noise.

GANs are designed for generating new samples. Instead of inferring the distribution of latent space, GANs sample from random noise and learn a transformation to imitate the real distribution of data. Gans improve the quality of imitations by training a discriminator to distinguish between generated samples and real samples.

In summary, VAEs sample from a prior distribution and infer the real distribution of latent space, while GANs sample from random noise and learn the data transformation by encouraging the competition between generator and discriminator.

Contributions

Name	**Contribution**
Chenyu Shi	Task 2 code, Task 2 report.
Shupei Li	Task 3 code, Task 3 report, Task2 code.
Shuang Fan	