Introduction to Deep Learning Assignment 2

Group 53

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

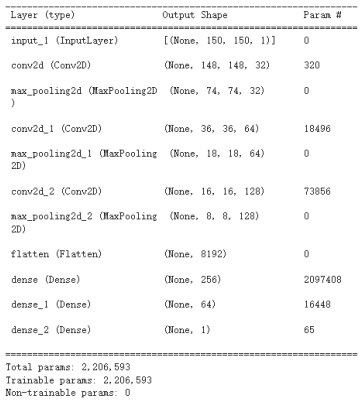
November 21, 2022

Task 1

Task 2

2.1 Regression

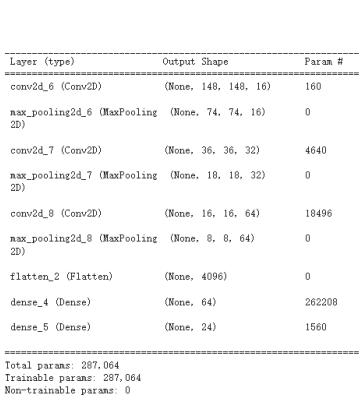
The regression model structure is shown in Figure 1. And its corresponding results are shown in Table 1. Because this is a regression task, we use MSE as loss function and Adam as optimizer. This model takes 2207593 trainable parameers, which is relatively high comparing to other models in task2. However, 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

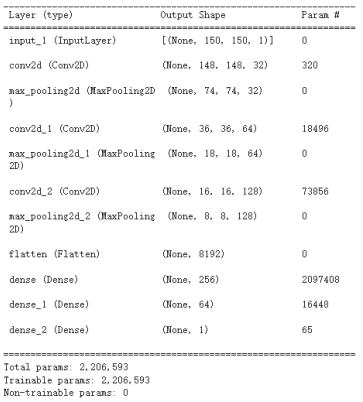
Figure 1: Regression model



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 & Moths 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 & Moths 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

3.4.1 MNIST

We modify the

3.5 Results

3.6 Discussion: Model Comparison

Contributions