Image Classification Using Convolutional Neural Network

Student ID:230161210

Student name: Qing Zhao

1. Convolutional Neural Network Review

Convolutional networks, also known as convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels.The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution.

A typical layer of a convolutional network consists of three stages.In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is run through a nonlinear activation function, such as the rectified linear activation function.In the third stage, we use a pooling function to modify the output of the layer further.

2. Problems Formulation and Image Classification Tasks

CNN has broad applications in image and video recognition, image classification,image segmentation,medical image analysis,natural language processing and financial time series. In the report I will use CNN to sovle two image classification tasks, the first one is to multi classify fixed and small size images. However, in the real world, images have different sizes and usually the pixies are much higher, so in the second task,I choose a dataset which contains different sizes of colorful images, and implement more complex techniques to solve this binary task.

3. Task one_Multi-Classification for Fashion-mnist Dataset

3.1 Explore the dataset

Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The total numbers of samples in each class are the same which means it is a balanced dataset. Figure 1 demonstrates the example from each class.

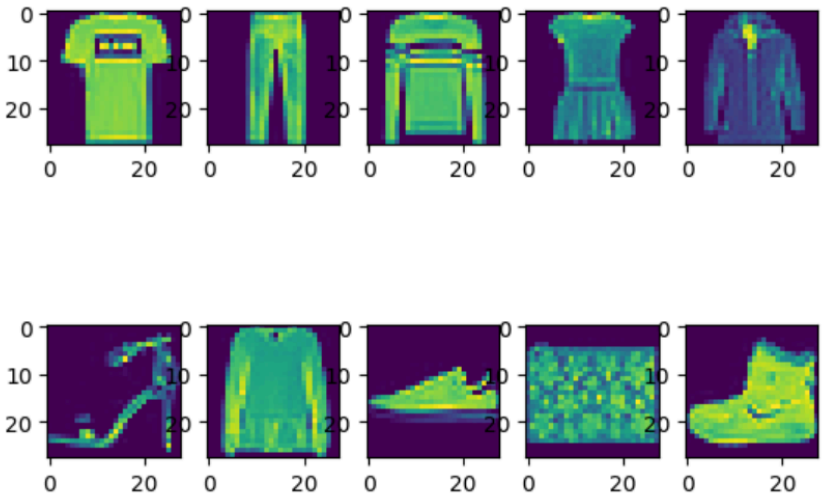

Figure 1: the example from each class in fashion-mnist dataset

3.2 Preprocess the dataset

Images in this dataset are represented as pixel values, which can range from 0 to 255. Before feeding into the model, I normalize the data using min-max scaling normalization techique. This is a crucial step for deep learning models, which can address several critical issues. It standardizes the input values, improves gradient descent optimization, ensures activation functions work effectively, and ultimately leads to faster training, better generalization, and potentially improved model performance.

3.3 Build a simple CNN model

In this task I build a CNN model which contains 8 layers, implementing twice convolution operations before connecting into fully connected layers. Figure 2 shows the summary architecture of the model.

Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_25 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_26 (Conv2D)	(None, 11, 11, 32)	9248
max_pooling2d_26 (MaxPooling2D)	(None, 5, 5, 32)	0
flatten_13 (Flatten)	(None, 800)	0
dense_26 (Dense)	(None, 64)	51264
dropout_12 (Dropout)	(None, 64)	0
dense_27 (Dense)	(None, 10)	650

Total params: 61482 (240.16 KB)
Trainable params: 61482 (240.16 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 2: the summary architecture of the model

In the two convolutional operations, I choose 32 filters to extract the features from each image, the size of each filter is 3*3, then followed by a 2*2 max pooling operation. After twice convolution operations, the original 28*28 image has been converts to 5*5*32 feature map representing the image. Finally, in the full connection layer, I choose 64 units to calculate the weights. As for the activation function, “Relu " is the most commonly used in CNN. Since this task is a multi classification task, the softmax function should be used in the last dense layer to convert the output into the probability for each class . To avoid overfitting, I added a dropout layer after the dense layer which will lead to randomly close 20% units in the dense layer per batch during the trainning process.

3.4 Train Model

Using ‘sparse_categorical_crossentropy’ loss function, ’Adam’ as optimizer, I trained the model with 20 epochs. With 20% samples splitted into validation, after 20 epochs, the accuracy of training achieved 0.94, while the accuracy of validation is 0.91. Figure 3 demonstrates the loss and accuracy of training and validation.

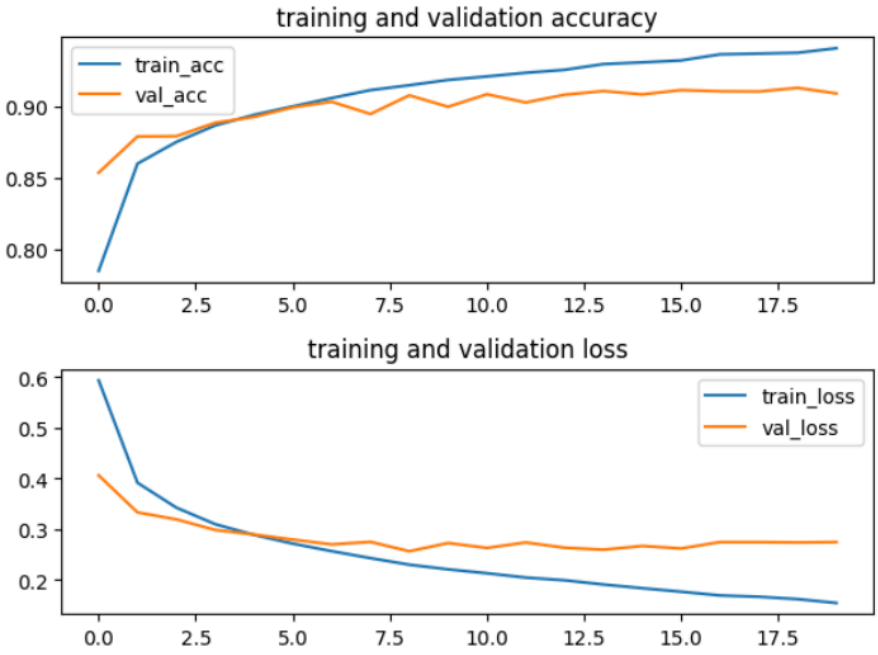

Figure 3: the loss and accuracy of training and validation

From the diagram above, we can see that the model still has an overfitting problem to some extent. Although the loss keeps going down, the loss of validation remains unchanged after about 10 epochs. The accuracy diagram shows the same tendancy. So the best training epochs is 10 rather than 20.

3.5 Model Prediction

Finally, I applied the model on the testing dataset, which contains 10000 samples with 1000 for each class. The accuracy of the testing dataset is 0.91, indicating the model does well in correctly classifying these fashion images. Figure 4 is the confusion matrix. We can see the model does the worst in correctly detecting the class of shirt (only 715 out of 1000 shirts is detected and mixing up with T-shirt is the main misclassification),while it did the best in classifying bags, only 18 out of 1000 was misclassified.

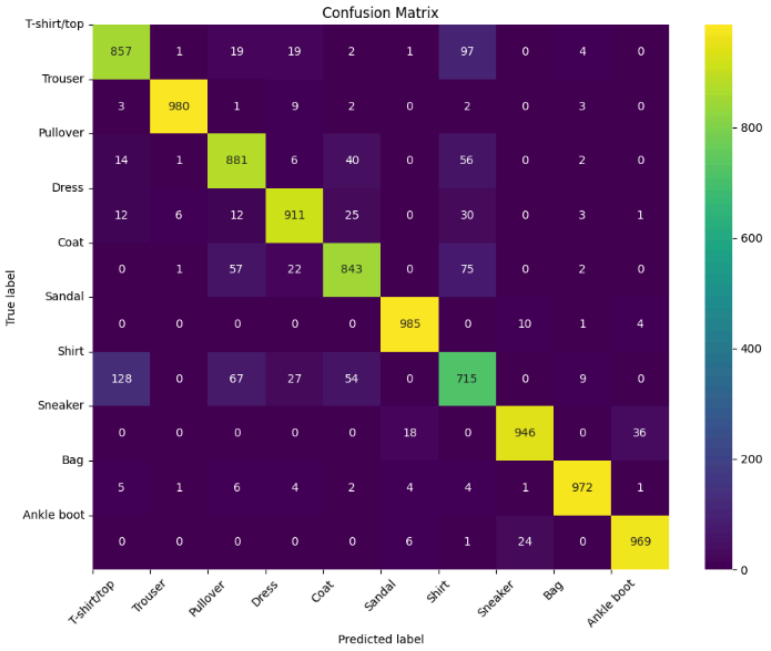

Figure 4: the confusion matrix of testing dataset

4. Task two_Image Classification on real world images

In the real world, images have different sizes and usually the pixies are much higher, so in the second task, I choose a dataset which contains different sizes of colorful images of cheetahs or lions from the real world. To solve this problem I will implement ImageDataGenerator which is a powerful API in tensorflow.

4.1 Explore the dataset

The cheetahs_vs_lions dataset is a combination of two dataset from kaggle. There are 300 images for training and 88 images for validation, each class has the same amount of samples. Below are some examples of the dataset.

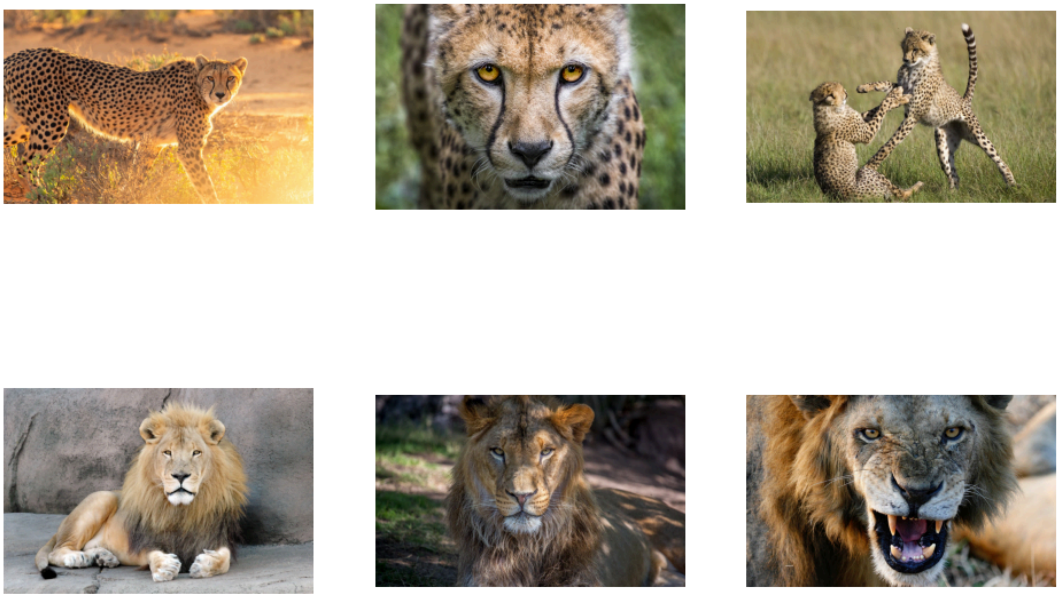

Figure 5: some examples of the cheetahs_vs_lions dataset

We can see pictures of lions and cheetahs taken from different angles, different resolutions, and even multiple cheetahs or lions in one picture. All these increase the difficulty of model learning and recognition.

4.2 Data Augmentation

ImageDataGenerator is a powerful class In tensorflow.keras, used for image data augmentation. It provides functionalities to manipulate and transform the image data on-the-fly during training. Since these transformations happen on-the-fly, the original images are not modified. The generator creates new variations in memory for each batch.By introducing these variations, it essentially increases the size and diversity of the training data, which can help to prevent overfitting and improve model robustness. In this project, I used augmentation techniques like rotations, zooms, and shears etc. figure 6 demonstrate the parameters of the ImageDataGenerator used in the data augmentation step.

implement Data augmentation

```
[ ]  train_datagen=ImageDataGenerator(rescale= 1./255.,
                                      rotation_range=40,
                                      width_shift_range=0.2,
                                      height_shift_range=0.2,
                                      shear_range=0.2,
                                      zoom_range=0.2,
                                      horizontal_flip=True,
                                      fill_mode='nearest')

     val_datagen=ImageDataGenerator(rescale=1./255.)
```

```
[ ]  train_generator=train_datagen.flow_from_directory(
         train_dir,
         target_size=(150,150),
         batch_size=30,
         class_mode='binary'
     )

     val_generator=val_datagen.flow_from_directory(
         val_dir,
         target_size=(150,150),
         batch_size=11,
         class_mode='binary'
     )
```

```
Found 300 images belonging to 2 classes.
Found 88 images belonging to 2 classes.
```

Figure 6: the parameters of the ImageDataGenerator used in the data augmentation

4.3 Build a CNN model

To solve this binary task, firstly, I build a deep learning model, which contains three convolutional layers along with three max pooling layers, followed by a fully connected layer with 512 units. The total parameters for this model is 3472449.

Layer (type)	Output Shape	Param #
conv2d_94 (Conv2D)	(None, 148, 148, 64)	1792
max_pooling2d_4 (MaxPooling2D)	(None, 74, 74, 64)	0
conv2d_95 (Conv2D)	(None, 72, 72, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_96 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_6 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_97 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_7 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_2 (Flatten)	(None, 6272)	0
dense_4 (Dense)	(None, 512)	3211776
dense_5 (Dense)	(None, 1)	513

Total params: 3472449 (13.25 MB)
Trainable params: 3472449 (13.25 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 7: the summary architecture of the CNN model

In order to find the optimal learning rate for the optimizer, I tried 100 epochs with the callback of LearningRateScheduler, setting 0.0001 as the final learning rate. In the following training step, although I tried different combinations of parameters, the accuracy of validation is consistently less than 75% and fluctuates widely between epochs. Figure 8 shows the loss and accuracy of the training and validation dataset.

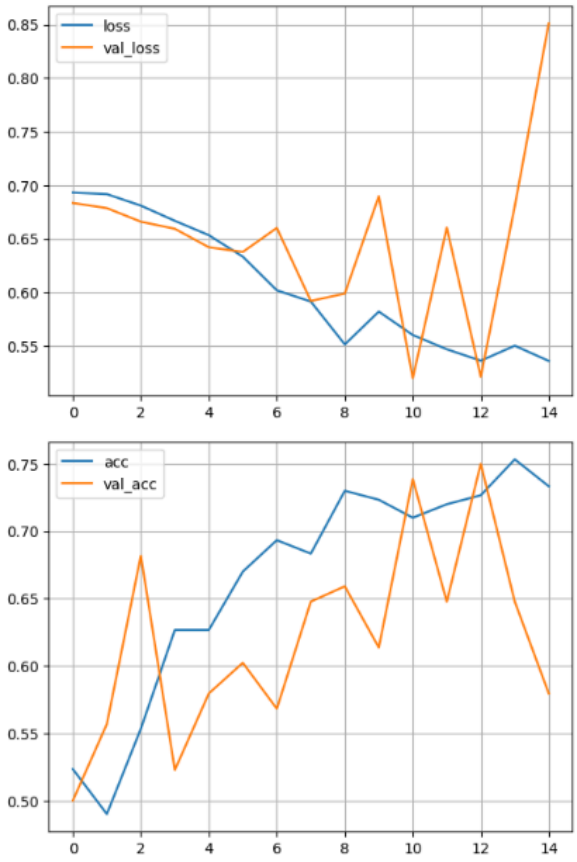

Figure 8: the loss and accuracy of the training and validation dataset.

4.4 Apply Transfer Learning to improve the model performance

To improve the model performance, I applied the transfer learning to this task, using the convolution layers of InceptionV3 architecture as my base model. By fetching the pretrained weights of the InceptionV3model and replacing its fully connected layer with my customized layers, the accuracy of both training dataset and validation dataset achieved 98%.

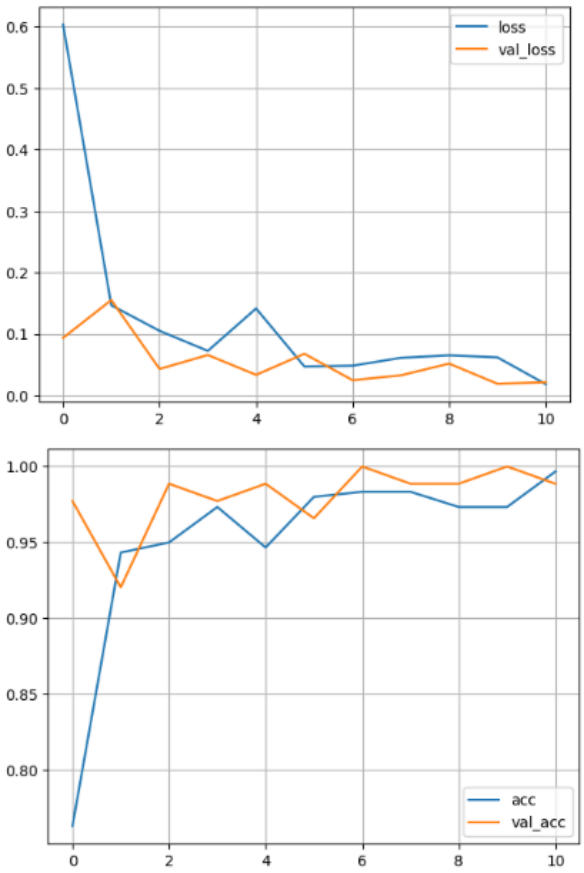

Figure 9: the loss and accuracy of the transfer learning customized model.

References

1. dataset:https://www.kaggle.com/datasets/mikoajfish99/lions-or-cheetahs-image-classification/data

2. dataset: https://www.kaggle.com/datasets/patriciabrezeanu/big-cats-image-classification-dataset

3. course video:MIT 6.S191: Convolutional Neural Networks

4. course video:Convolutional Neural Networks in TensorFlow

5. textbook:Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville