Week 11 Report

CNN

37237256 薛丁銘

The code of this week is as shown in Figure 1.I tried to add 4 convolutional layers to the network but it showed error messages. I think the limit should be 3 because the network has to maintain its shape after pooling. The accuracy increased with epochs. The results are as shown in Figure 2.

```
school > Introduction to Machine Learning > Week11_exercise > CNN.py > ...
import matplotlib.pyplot as plt
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
print(x_train.shape)

# plt.imshow(x_train[1])
# plt.show()

x_train, x_test = x_train/255.0, x_test/255.0

## Regular NN
# model_NN = tf.keras.models.Sequential([
#     tf.keras.layers.Flatten(input_shape=(32,32,3)),
#     # tf.keras.layers.Dense(128, activation='sigmoid'),
#     # tf.keras.layers.Dense(128, activation='sigmoid'),
#     tf.keras.layers.Dense(128, activation='relu'),
#     tf.keras.layers.Dense(64, activation='relu'),
#     tf.keras.layers.Dense(64, activation='relu'),
#     tf.keras.layers.Dense(32, activation='relu'),
#     tf.keras.layers.Dense(100, activation='softmax')
# ])

# model_NN.compile(
#     optimizer = 'adam',
#     loss = 'sparse_categorical_crossentropy',
#     metrics = ['accuracy']
# )

# model_NN.fit(x_train, y_train, epochs = 10, validation_data = (x_test, y_test), batch_size = 128)

```

Figure 1.a Code

```
## CNN
model_CNN = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3), strides = (1,1)),
    tf.keras.layers.AveragePooling2D((2,2)),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu', strides = (1,1)),
    tf.keras.layers.AveragePooling2D((2,2)),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu', strides = (1,1)),
    tf.keras.layers.MaxPooling2D((2,2)),

    # tf.keras.layers.Conv2D(128, (3,3), activation='relu', strides = (1,1)),
    # tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_CNN.summary()
model_CNN.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)

# model_CNN.fit(x_train, y_train, epochs = 5, validation_data = (x_test, y_test), batch_size = 128)

history_CNN = model_CNN.fit(x_train, y_train, epochs = 20, validation_data = (x_test, y_test), batch_size = 128)
plt.plot(history_CNN.history['accuracy'], label = 'accuracy')
plt.plot(history_CNN.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5,1])
plt.legend(loc = 'lower right')
plt.show()
```

Figure 1.b Code

```
(50000, 32, 32, 3)
2024-01-06 18:43:05.858770: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with one
n performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 average_pooling2d (AverageP  (None, 15, 15, 32)       0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 average_pooling2d_1 (Averag  (None, 6, 6, 64)         0
 ePooling2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d (MaxPooling2D  (None, 2, 2, 128)        0
 )

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 32)                2080

 dense_4 (Dense)             (None, 10)                330

=================================================================
Total params: 186,090
```

Figure 2.a Results

```
=================================================================
Total params: 186,090
Trainable params: 186,090
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
391/391 [==============================] - 17s 42ms/step - loss: 1.8007 - accuracy: 0.3252 - val_loss: 1.5578 - val_accuracy: 0.4271
Epoch 2/20
391/391 [==============================] - 35s 90ms/step - loss: 1.4155 - accuracy: 0.4819 - val_loss: 1.3469 - val_accuracy: 0.5093
Epoch 3/20
391/391 [==============================] - 42s 108ms/step - loss: 1.2447 - accuracy: 0.5487 - val_loss: 1.2074 - val_accuracy: 0.5623
Epoch 4/20
391/391 [==============================] - 35s 88ms/step - loss: 1.1202 - accuracy: 0.6006 - val_loss: 1.1064 - val_accuracy: 0.6047
Epoch 5/20
391/391 [==============================] - 35s 90ms/step - loss: 1.0269 - accuracy: 0.6362 - val_loss: 1.0211 - val_accuracy: 0.6324
Epoch 6/20
391/391 [==============================] - 34s 86ms/step - loss: 0.9520 - accuracy: 0.6606 - val_loss: 1.0410 - val_accuracy: 0.6372
Epoch 7/20
391/391 [==============================] - 41s 105ms/step - loss: 0.8789 - accuracy: 0.6888 - val_loss: 0.9629 - val_accuracy: 0.6586
Epoch 8/20
391/391 [==============================] - 46s 118ms/step - loss: 0.8128 - accuracy: 0.7143 - val_loss: 0.9405 - val_accuracy: 0.6698
Epoch 9/20
391/391 [==============================] - 35s 89ms/step - loss: 0.7666 - accuracy: 0.7301 - val_loss: 0.9180 - val_accuracy: 0.6794
Epoch 10/20
391/391 [==============================] - 34s 87ms/step - loss: 0.7169 - accuracy: 0.7474 - val_loss: 0.9026 - val_accuracy: 0.6891
Epoch 11/20
391/391 [==============================] - 33s 85ms/step - loss: 0.6830 - accuracy: 0.7601 - val_loss: 0.8952 - val_accuracy: 0.6943
Epoch 12/20
391/391 [==============================] - 33s 85ms/step - loss: 0.6299 - accuracy: 0.7800 - val_loss: 0.9290 - val_accuracy: 0.6923
Epoch 13/20
391/391 [==============================] - 33s 84ms/step - loss: 0.5957 - accuracy: 0.7917 - val_loss: 0.8889 - val_accuracy: 0.7039
Epoch 14/20
391/391 [==============================] - 34s 88ms/step - loss: 0.5612 - accuracy: 0.8027 - val_loss: 0.9069 - val_accuracy: 0.7073
Epoch 15/20
391/391 [==============================] - 39s 99ms/step - loss: 0.5347 - accuracy: 0.8138 - val_loss: 0.8919 - val_accuracy: 0.7089
Epoch 16/20
391/391 [==============================] - 39s 101ms/step - loss: 0.4969 - accuracy: 0.8268 - val_loss: 0.9487 - val_accuracy: 0.7058
Epoch 17/20
391/391 [==============================] - 33s 85ms/step - loss: 0.4712 - accuracy: 0.8347 - val_loss: 0.9298 - val_accuracy: 0.7103
Epoch 18/20
391/391 [==============================] - 33s 84ms/step - loss: 0.4428 - accuracy: 0.8444 - val_loss: 0.9074 - val_accuracy: 0.7123
Epoch 19/20
391/391 [==============================] - 33s 83ms/step - loss: 0.4176 - accuracy: 0.8546 - val_loss: 0.9606 - val_accuracy: 0.7129
Epoch 20/20
391/391 [==============================] - 33s 85ms/step - loss: 0.3890 - accuracy: 0.8632 - val_loss: 0.9916 - val_accuracy: 0.7109
```

Figure 2.b Results

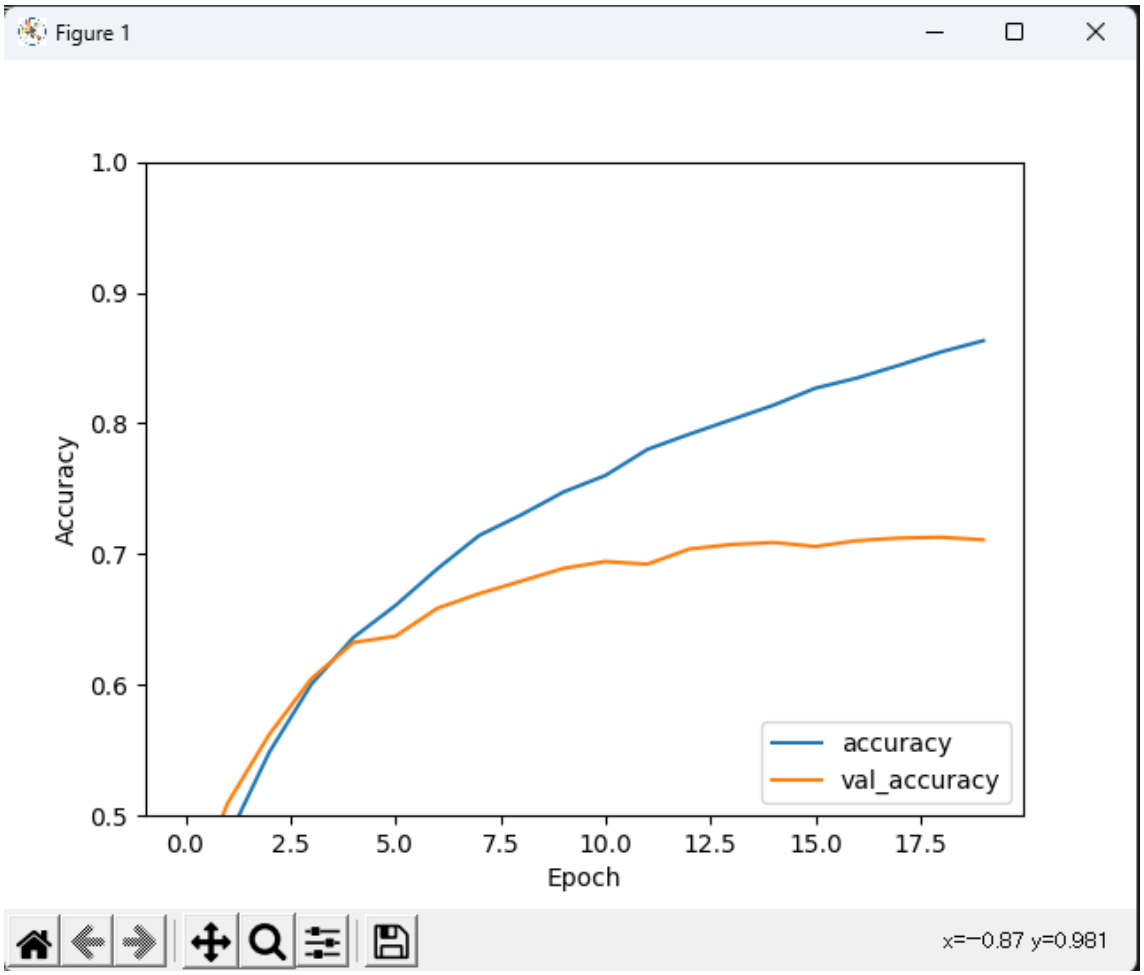

Figure 2.c Results