Assignment 3*

Xiang Yu-Ye
Mechanical Engineering
National Cheng Kung University
Tainan City East Dist. Daxue Rd., Taiwan
N16114445@gs.ncku.edu.tw

Abstract **-**使用LeNet對資料集分50個類別。

Index Terms – Deep Learning、LeNet5

I. INTRODUCTION

本次作業目標為使用Tensorflow or pytorch實現LeNet，所有code都放在 https://github.com/Jason890102/Deep-Learning/tree/main/Assignment_2_LeNetComputational_Graph。

II. METHOD

A. LeNet Architecture

Two Layer Net架構圖如Fig. 1.。Fig. 2.為trianing的Loss curve，以及trian&Test的accury，訓練300個epochs。

Layer (type)	Output Shape	Param #
conv2d_78 (Conv2D)	(None, 254, 254, 6)	168
conv2d_79 (Conv2D)	(None, 252, 252, 10)	550
max_pooling2d_43 (MaxPooling	(None, 126, 126, 10)	0
module_wrapper_38 (ModuleWra	(None, 126, 126, 10)	0
conv2d_80 (Conv2D)	(None, 124, 124, 10)	910
conv2d_81 (Conv2D)	(None, 122, 122, 10)	910
module_wrapper_39 (ModuleWra	(None, 122, 122, 10)	40
conv2d_82 (Conv2D)	(None, 120, 120, 16)	1456
max_pooling2d_44 (MaxPooling	(None, 60, 60, 16)	0
module_wrapper_40 (ModuleWra	(None, 60, 60, 16)	64
conv2d_83 (Conv2D)	(None, 58, 58, 16)	2320
max_pooling2d_45 (MaxPooling	(None, 29, 29, 16)	0
module_wrapper_41 (ModuleWra	(None, 29, 29, 16)	0
conv2d_84 (Conv2D)	(None, 27, 27, 16)	2320
max_pooling2d_46 (MaxPooling	(None, 13, 13, 16)	0
module_wrapper_42 (ModuleWra	(None, 13, 13, 16)	64
conv2d_85 (Conv2D)	(None, 11, 11, 16)	2320
module_wrapper_43 (ModuleWra	(None, 11, 11, 16)	0
conv2d_86 (Conv2D)	(None, 10, 10, 50)	3250
module_wrapper_44 (ModuleWra	(None, 10, 10, 50)	0
global_average_pooling2d_12	(None, 50)	0
dense_40 (Dense)	(None, 1200)	61200
dense_41 (Dense)	(None, 840)	1008840
dense_42 (Dense)	(None, 640)	538240
dense_43 (Dense)	(None, 200)	128200
dense_44 (Dense)	(None, 50)	10050

Total params: 1,760,902
Trainable params: 1,760,818
Non-trainable params: 84

Fig. 1. LeNet Architecture

```
model = Sequential()

model.add(Conv2D(6,(3,3),activation='relu', input_shape=(256,256,3),kernel_initializer = initializers.HeNormal()))
model.add(Conv2D(10,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(.4))
model.add(Conv2D(10,(3,3),activation='relu'))
model.add(Conv2D(10,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(16,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(16,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(.4))
model.add(Conv2D(16,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(16,(3,3),activation='relu'))
model.add(Dropout(.4))
model.add(Conv2D(50,(2,2),activation='relu'))
model.add(Dropout(.4))
model.add(GlobalAveragePooling2D())
model.add(Dense(1200,activation='relu'))
model.add(Dense(840,activation='relu'))
model.add(Dense(640,activation='relu'))
model.add(Dense(200,activation='relu'))
model.add(Dense(50,activation='softmax'))

# opt = tf.python.keras.optimizer_v2.adam(
#     learning_rate=1e-5)
# loss = tf.python.keras.losses.SparseCategoricalCrossentropy()
loss = tf.keras.losses.SparseCategoricalCrossentropy()


model.compile(optimizer = 'Adam',
              loss = loss,
              metrics=['accuracy'])

model.summary() #輸出訓練模型
```

Fig. 2. LeNet hyprpapramter

B. Train&validation accuracy

Fig. 3.為Train&Validation accuracy curve，從結果來看最高的準確度為32%，並且有點overfitting，Fig. 4.為手刻的LeNet5的Loss Curve、train&test accury，最高為3%。

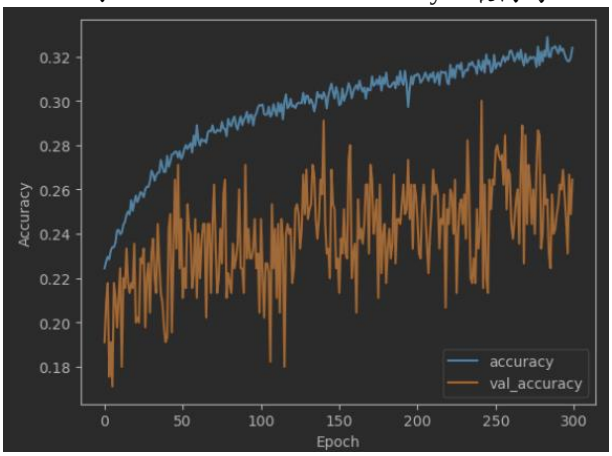

Fig. 3. Train&validation accury

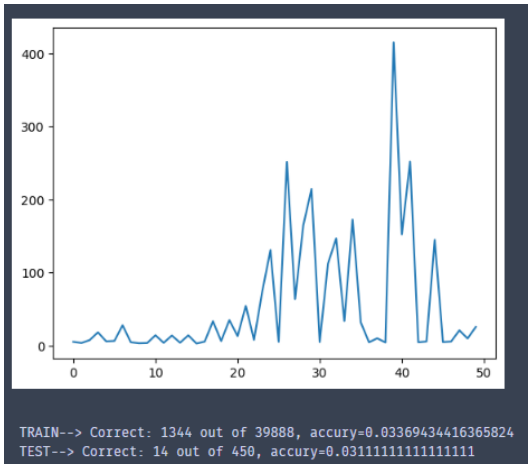

Fig. 4. LeNet5 Loss Curve、train&test accury

C. The predicted result on the validation set

Validation的accuracy如Fig. 5.，分為50個類別，Fig. 6. 為Test set的accuracy，可以看出Test set的accuracy稍微好一點，。

```
15/15 - 2s - loss: 2.6859 - accuracy: 0.2644
validation損失值: 2.685920238494873 validation精準度: 0.2644444406032562
```

Fig. 5. Validation predict accuracy

```
15/15 - 1s - loss: 2.6118 - accuracy: 0.2667
test損失值: 2.611841917037964 test精準度: 0.2666666805744171
```

Fig. 6. Test predict accuracy

III. CONCLUSION

在寫作業的過程中對於資料的前處理還是不太會導致正確率一直提升不上去，有試著增加層數及降低Learning rate，卻還是沒有超過40%。