Assignment_6.2b

July 15, 2021

[1]:
```
from keras.datasets import cifar10
from keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
import pandas as pd
import matplotlib.pyplot as plt

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
A local file was found, but it seems to be incomplete or outdated because the
auto file hash does not match the original value of
6d958be074577803d12ecdefd02955f39262c83c16fe9348329d7fe0b5c001ce so we will re-
download the data.
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 22s 0us/step
```

[2]:
```
x_train.shape, y_train.shape
```

[2]:
```
((50000, 32, 32, 3), (50000, 1))
```

[3]:
```
x_test.shape, y_test.shape
```

[3]:
```
((10000, 32, 32, 3), (10000, 1))
```

[4]:
```
# Preprocess the data (these are NumPy arrays)
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

[5]:
```
# Reserve 10,000 samples for validation
x_val = x_train[-10000:]
y_val = y_train[-10000:]
x_train_2 = x_train[:-10000]
y_train_2 = y_train[:-10000]
```

[6]:
```
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)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow(x_train_2, y_train_2,  batch_size=32)

validation_generator = train_datagen.flow(x_val, y_val, batch_size=32)
```

[7]:
```
#instantiate the model
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 64)          0
```

```
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 64)                16448
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 73,418
Trainable params: 73,418
Non-trainable params: 0
_________________________________________________________________
```

[8]:
```
from keras import optimizers

model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[9]:
```
history = model.fit_generator(train_generator,
                              steps_per_epoch=len(x_train_2) / 32,
                              epochs=30,
                              validation_data=validation_generator,
                              validation_steps=len(x_val) / 32)
```

```
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/30
1250/1250 [==============================] - 60s 47ms/step - loss: 2.2345 -
accuracy: 0.1506 - val_loss: 1.9912 - val_accuracy: 0.2555
Epoch 2/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.9917 -
accuracy: 0.2482 - val_loss: 1.9086 - val_accuracy: 0.2968
Epoch 3/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.9217 -
accuracy: 0.2808 - val_loss: 1.8425 - val_accuracy: 0.3159
Epoch 4/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.8709 -
accuracy: 0.3068 - val_loss: 1.8045 - val_accuracy: 0.3331
Epoch 5/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.8241 -
accuracy: 0.3224 - val_loss: 1.7681 - val_accuracy: 0.3492
Epoch 6/30
```

```
1250/1250 [==============================] - 58s 46ms/step - loss: 1.7955 -
accuracy: 0.3340 - val_loss: 1.7300 - val_accuracy: 0.3601
Epoch 7/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.7610 -
accuracy: 0.3522 - val_loss: 1.7076 - val_accuracy: 0.3737
Epoch 8/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.7422 -
accuracy: 0.3605 - val_loss: 1.7180 - val_accuracy: 0.3708
Epoch 9/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.7113 -
accuracy: 0.3688 - val_loss: 1.6650 - val_accuracy: 0.3983
Epoch 10/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.7001 -
accuracy: 0.3827 - val_loss: 1.6621 - val_accuracy: 0.3984
Epoch 11/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.6806 -
accuracy: 0.3853 - val_loss: 1.6346 - val_accuracy: 0.4102
Epoch 12/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6613 -
accuracy: 0.3946 - val_loss: 1.6068 - val_accuracy: 0.4207
Epoch 13/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.6567 -
accuracy: 0.4003 - val_loss: 1.5812 - val_accuracy: 0.4277
Epoch 14/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6438 -
accuracy: 0.4026 - val_loss: 1.5715 - val_accuracy: 0.4371
Epoch 15/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6283 -
accuracy: 0.4103 - val_loss: 1.5709 - val_accuracy: 0.4381
Epoch 16/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6083 -
accuracy: 0.4170 - val_loss: 1.5413 - val_accuracy: 0.4493
Epoch 17/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6014 -
accuracy: 0.4205 - val_loss: 1.5466 - val_accuracy: 0.4400
Epoch 18/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5938 -
accuracy: 0.4247 - val_loss: 1.5225 - val_accuracy: 0.4587
Epoch 19/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5859 -
accuracy: 0.4317 - val_loss: 1.5200 - val_accuracy: 0.4547
Epoch 20/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5902 -
accuracy: 0.4300 - val_loss: 1.5099 - val_accuracy: 0.4598
Epoch 21/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5650 -
accuracy: 0.4408 - val_loss: 1.4952 - val_accuracy: 0.4705
Epoch 22/30
```

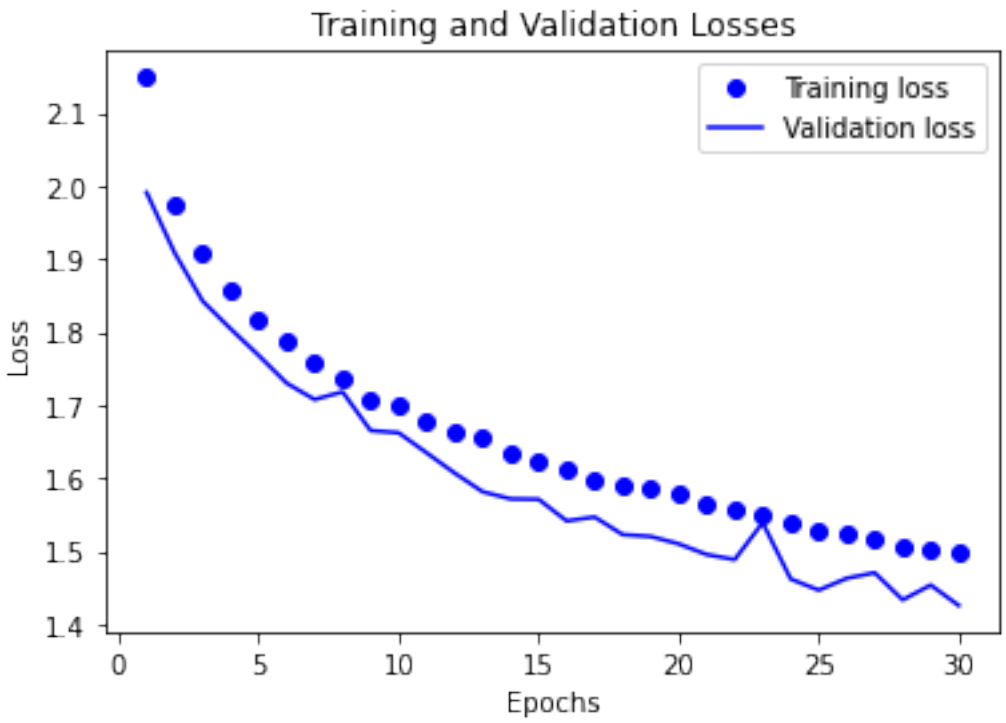
```
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5661 -
accuracy: 0.4425 - val_loss: 1.4884 - val_accuracy: 0.4624
Epoch 23/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5450 -
accuracy: 0.4460 - val_loss: 1.5389 - val_accuracy: 0.4531
Epoch 24/30
1250/1250 [==============================] - 57s 45ms/step - loss: 1.5398 -
accuracy: 0.4496 - val_loss: 1.4618 - val_accuracy: 0.4777
Epoch 25/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.5385 -
accuracy: 0.4493 - val_loss: 1.4465 - val_accuracy: 0.4853
Epoch 26/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.5231 -
accuracy: 0.4564 - val_loss: 1.4627 - val_accuracy: 0.4754
Epoch 27/30
1250/1250 [==============================] - 53s 43ms/step - loss: 1.5106 -
accuracy: 0.4621 - val_loss: 1.4703 - val_accuracy: 0.4792
Epoch 28/30
1250/1250 [==============================] - 43s 34ms/step - loss: 1.5147 -
accuracy: 0.4562 - val_loss: 1.4331 - val_accuracy: 0.4902
Epoch 29/30
1250/1250 [==============================] - 43s 34ms/step - loss: 1.5085 -
accuracy: 0.4626 - val_loss: 1.4535 - val_accuracy: 0.4820
Epoch 30/30
1250/1250 [==============================] - 43s 34ms/step - loss: 1.4960 -
accuracy: 0.4672 - val_loss: 1.4255 - val_accuracy: 0.4918
```

[10]:
```
train_loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(history.history['loss']) + 1)

plt.plot(epochs, train_loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and Validation Losses')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
plt.savefig('results/6_2b_lossplot.png')
```



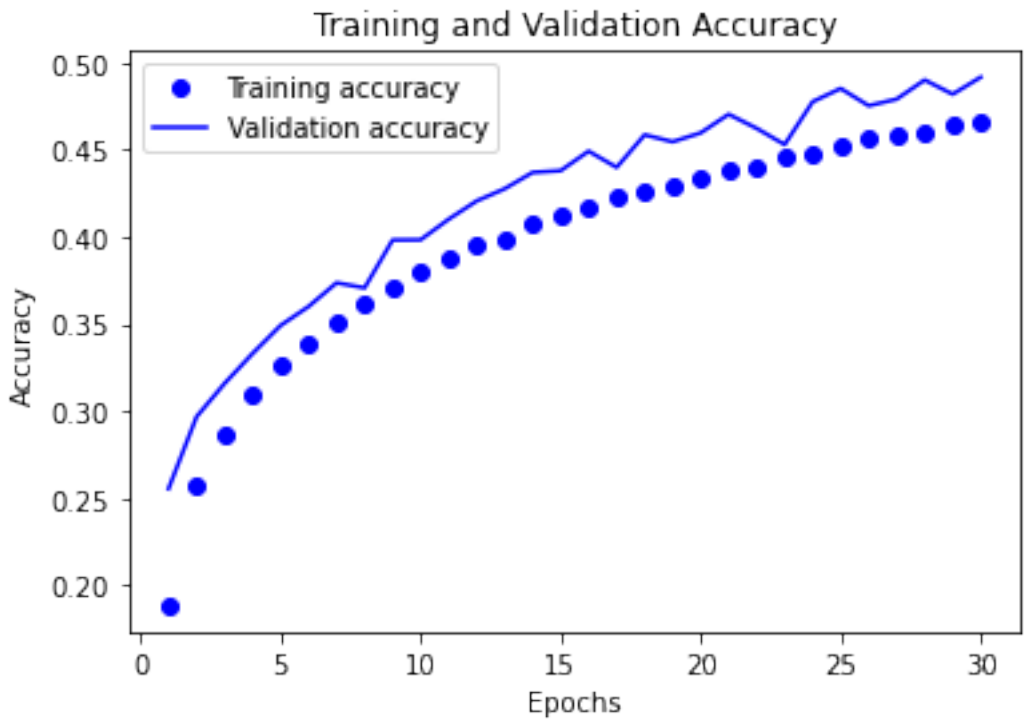

```
<Figure size 432x288 with 0 Axes>
```

[11]:
```
train_loss = history.history['accuracy']
val_loss = history.history['val_accuracy']

epochs = range(1, len(history.history['accuracy']) + 1)

plt.plot(epochs, train_loss, 'bo', label='Training accuracy')
plt.plot(epochs, val_loss, 'b', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
plt.savefig('results/6_2b_accplot.png')
```



<Figure size 432x288 with 0 Axes>

[12]:
```
#retrain the model and evaluate on test
train_generator = train_datagen.flow(x_train, y_train,  batch_size=32)

model.compile(optimizer=optimizers.RMSprop(lr=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

#16 epochs chosen based on graphs above
history = model.fit_generator(train_generator,
                              steps_per_epoch=len(x_train) / 32,
                              epochs=16)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/16
1562/1562 [==============================] - 45s 28ms/step - loss: 1.4980 -
accuracy: 0.4654
Epoch 2/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4841 -
accuracy: 0.4684
Epoch 3/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4770 -
accuracy: 0.4744
```

```
Epoch 4/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4596 -
accuracy: 0.4796
Epoch 5/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4613 -
accuracy: 0.4803
Epoch 6/16
1562/1562 [==============================] - 45s 28ms/step - loss: 1.4551 -
accuracy: 0.4803
Epoch 7/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4536 -
accuracy: 0.4842
Epoch 8/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4387 -
accuracy: 0.4878
Epoch 9/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4443 -
accuracy: 0.4859
Epoch 10/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4298 -
accuracy: 0.4911
Epoch 11/16
1562/1562 [==============================] - 45s 29ms/step - loss: 1.4355 -
accuracy: 0.4876
Epoch 12/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4225 -
accuracy: 0.4950
Epoch 13/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4078 -
accuracy: 0.4986
Epoch 14/16
1562/1562 [==============================] - 45s 29ms/step - loss: 1.4052 -
accuracy: 0.5009
Epoch 15/16
1562/1562 [==============================] - 44s 28ms/step - loss: 1.4082 -
accuracy: 0.4969
Epoch 16/16
1562/1562 [==============================] - 45s 28ms/step - loss: 1.4050 -
accuracy: 0.5023
313/313 [==============================] - 1s 4ms/step - loss: 257.6946 -
accuracy: 0.3655
```

[13]:
```
model.save('results/6_2b_model.h5')
```

[14]:
```
prediction_results = model.predict(x_test)
```

[15]:
```
#write metrics to file
with open('results/6_2b_metrics.txt', 'w') as f:
    f.write('Training Loss: {}'.format(str(history.history['loss'])))
    f.write('\nTraining Accuracy: {}'.format(str(history.history['accuracy'])))
    f.write('\nTest Loss: {}'.format(results[0]))
    f.write('\nTest Accuracy: {}'.format(results[1]))
```

[16]:
```
predictions = pd.DataFrame(prediction_results,␣
 ↪columns=['0','1','2','3','4','5','6','7','8','9'])
predictions.to_csv('results/6_2b_predictions.csv', index=False)
```

[ ]:

[ ]:

[ ]: