Assignment_6.2b

April 26, 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()
```

[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 [==============================] - 62s 48ms/step - loss: 2.2082 -
accuracy: 0.1596 - val_loss: 1.9667 - val_accuracy: 0.2544
Epoch 2/30
1250/1250 [==============================] - 59s 47ms/step - loss: 1.9785 -
accuracy: 0.2525 - val_loss: 1.8877 - val_accuracy: 0.2911
Epoch 3/30
1250/1250 [==============================] - 59s 47ms/step - loss: 1.9106 -
accuracy: 0.2892 - val_loss: 1.8379 - val_accuracy: 0.3208
Epoch 4/30
1250/1250 [==============================] - 53s 42ms/step - loss: 1.8577 -
accuracy: 0.3084 - val_loss: 1.7903 - val_accuracy: 0.3439
Epoch 5/30
1250/1250 [==============================] - 52s 42ms/step - loss: 1.8123 -
accuracy: 0.3285 - val_loss: 1.7462 - val_accuracy: 0.3628
Epoch 6/30
1250/1250 [==============================] - 57s 45ms/step - loss: 1.7827 -
accuracy: 0.3428 - val_loss: 1.7158 - val_accuracy: 0.3694
Epoch 7/30
1250/1250 [==============================] - 59s 47ms/step - loss: 1.7572 -
accuracy: 0.3522 - val_loss: 1.6853 - val_accuracy: 0.3892
Epoch 8/30
1250/1250 [==============================] - 59s 47ms/step - loss: 1.7331 -
accuracy: 0.3684 - val_loss: 1.6670 - val_accuracy: 0.3890
```

```
Epoch 9/30
1250/1250 [==============================] - 59s 47ms/step - loss: 1.7041 -
accuracy: 0.3822 - val_loss: 1.6476 - val_accuracy: 0.4003
Epoch 10/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.6885 -
accuracy: 0.3819 - val_loss: 1.6212 - val_accuracy: 0.4112
Epoch 11/30
1250/1250 [==============================] - 52s 42ms/step - loss: 1.6760 -
accuracy: 0.3888 - val_loss: 1.6007 - val_accuracy: 0.4224
Epoch 12/30
1250/1250 [==============================] - 61s 49ms/step - loss: 1.6569 -
accuracy: 0.4025 - val_loss: 1.5912 - val_accuracy: 0.4229
Epoch 13/30
1250/1250 [==============================] - 64s 51ms/step - loss: 1.6493 -
accuracy: 0.4058 - val_loss: 1.5677 - val_accuracy: 0.4407
Epoch 14/30
1250/1250 [==============================] - 62s 50ms/step - loss: 1.6333 -
accuracy: 0.4057 - val_loss: 1.5599 - val_accuracy: 0.4383
Epoch 15/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.6091 -
accuracy: 0.4207 - val_loss: 1.5447 - val_accuracy: 0.4444
Epoch 16/30
1250/1250 [==============================] - 62s 49ms/step - loss: 1.5973 -
accuracy: 0.4176 - val_loss: 1.5398 - val_accuracy: 0.4459
Epoch 17/30
1250/1250 [==============================] - 55s 44ms/step - loss: 1.5933 -
accuracy: 0.4280 - val_loss: 1.5395 - val_accuracy: 0.4485
Epoch 18/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5850 -
accuracy: 0.4256 - val_loss: 1.5424 - val_accuracy: 0.4430
Epoch 19/30
1250/1250 [==============================] - 53s 42ms/step - loss: 1.5725 -
accuracy: 0.4339 - val_loss: 1.5086 - val_accuracy: 0.4629
Epoch 20/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5553 -
accuracy: 0.4439 - val_loss: 1.4927 - val_accuracy: 0.4689
Epoch 21/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.5422 -
accuracy: 0.4436 - val_loss: 1.4968 - val_accuracy: 0.4684
Epoch 22/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5437 -
accuracy: 0.4472 - val_loss: 1.4592 - val_accuracy: 0.4847
Epoch 23/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.5203 -
accuracy: 0.4505 - val_loss: 1.4661 - val_accuracy: 0.4827
Epoch 24/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.5254 -
accuracy: 0.4508 - val_loss: 1.4686 - val_accuracy: 0.4820
```

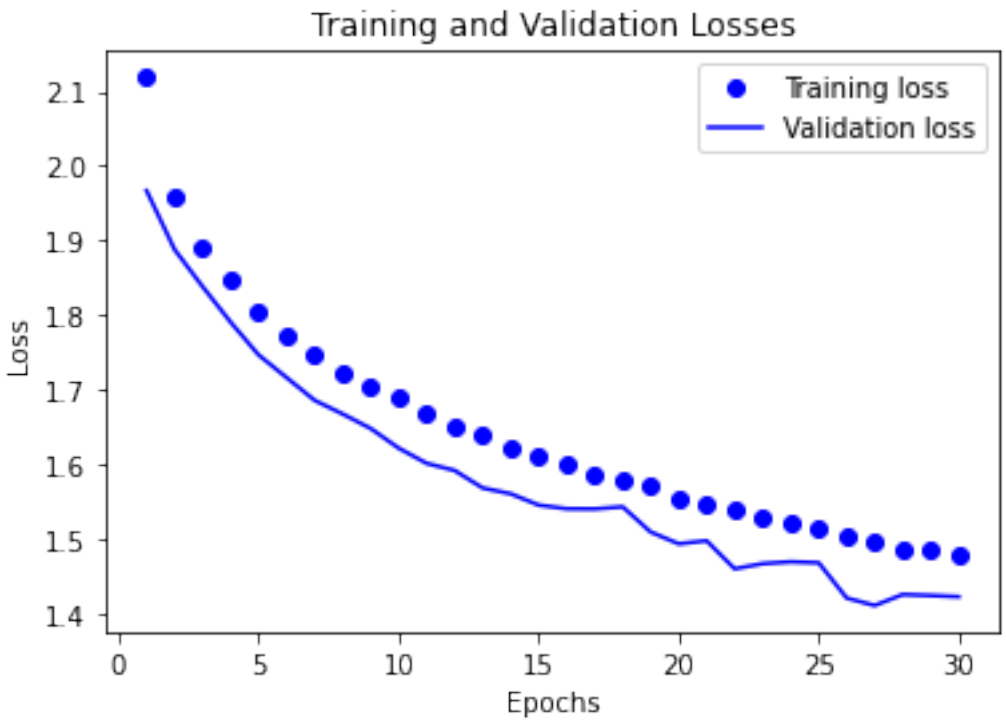
```
Epoch 25/30
1250/1250 [==============================] - 57s 46ms/step - loss: 1.5062 -
accuracy: 0.4621 - val_loss: 1.4673 - val_accuracy: 0.4751
Epoch 26/30
1250/1250 [==============================] - 58s 46ms/step - loss: 1.5126 -
accuracy: 0.4557 - val_loss: 1.4200 - val_accuracy: 0.4975
Epoch 27/30
1250/1250 [==============================] - 60s 48ms/step - loss: 1.4866 -
accuracy: 0.4683 - val_loss: 1.4100 - val_accuracy: 0.5010
Epoch 28/30
1250/1250 [==============================] - 59s 47ms/step - loss: 1.4840 -
accuracy: 0.4737 - val_loss: 1.4247 - val_accuracy: 0.4995
Epoch 29/30
1250/1250 [==============================] - 48s 38ms/step - loss: 1.4890 -
accuracy: 0.4650 - val_loss: 1.4234 - val_accuracy: 0.5041
Epoch 30/30
1250/1250 [==============================] - 48s 39ms/step - loss: 1.4588 -
accuracy: 0.4763 - val_loss: 1.4218 - val_accuracy: 0.4980
```

[10]:
```python
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')
```



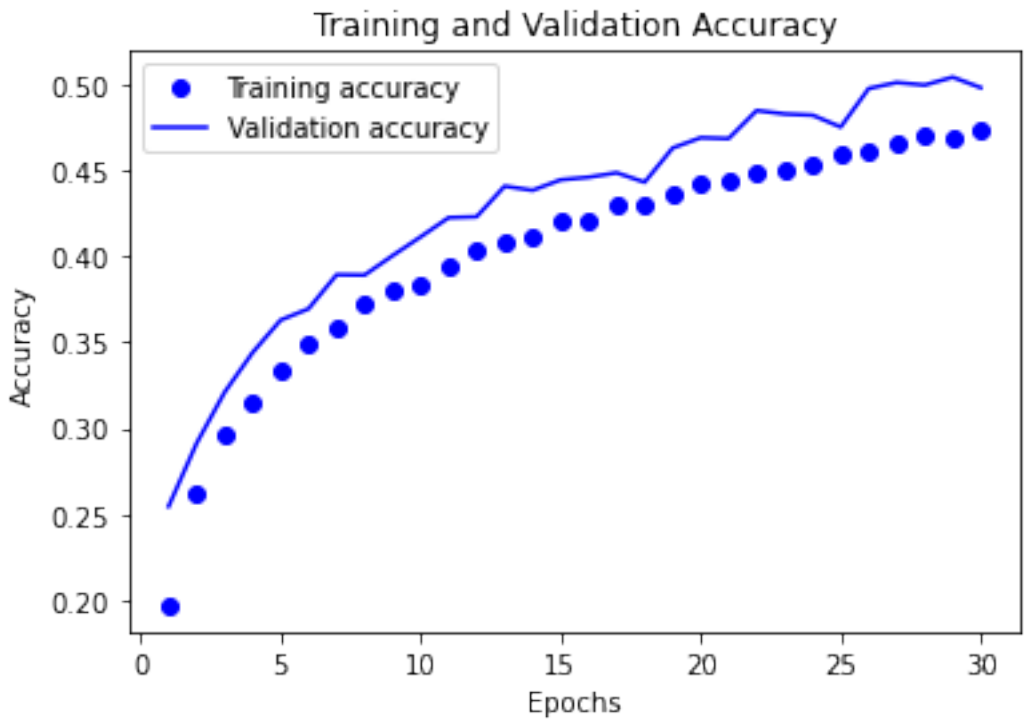

```
<Figure size 432x288 with 0 Axes>
```

[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 [==============================] - 50s 32ms/step - loss: 1.4728 -
accuracy: 0.4737
Epoch 2/16
1562/1562 [==============================] - 50s 32ms/step - loss: 1.4716 -
accuracy: 0.4775
Epoch 3/16
1562/1562 [==============================] - 49s 31ms/step - loss: 1.4582 -
accuracy: 0.4770
```

```
Epoch 4/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4484 -
accuracy: 0.4829
Epoch 5/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4299 -
accuracy: 0.4892
Epoch 6/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4192 -
accuracy: 0.4954
Epoch 7/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4200 -
accuracy: 0.4908
Epoch 8/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4249 -
accuracy: 0.4961
Epoch 9/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4180 -
accuracy: 0.4947
Epoch 10/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4130 -
accuracy: 0.4996
Epoch 11/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.3998 -
accuracy: 0.5034
Epoch 12/16
1562/1562 [==============================] - 48s 31ms/step - loss: 1.4081 -
accuracy: 0.4989
Epoch 13/16
1562/1562 [==============================] - 55s 35ms/step - loss: 1.3965 -
accuracy: 0.5019
Epoch 14/16
1562/1562 [==============================] - 60s 39ms/step - loss: 1.3851 -
accuracy: 0.5078
Epoch 15/16
1562/1562 [==============================] - 60s 39ms/step - loss: 1.3788 -
accuracy: 0.5068
Epoch 16/16
1562/1562 [==============================] - 61s 39ms/step - loss: 1.3887 -
accuracy: 0.5081
313/313 [==============================] - 2s 6ms/step - loss: 216.6761 -
accuracy: 0.4007
```

[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)
```

[ ]:
```

```