Assignment_6.2a

April 26, 2021

[1]:

```
from keras.datasets import cifar10
from keras.utils import to_categorical
import pandas as pd

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

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 19s 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") / 255
x_test = x_test.astype("float32") / 255

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

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

[5]:

```
x_val.shape, y_val.shape
```

[5]:

```
((10000, 32, 32, 3), (10000, 10))
```

[6]:

```
#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.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
_________________________________________________________________
dense (Dense)                (None, 64)                16448
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 73,418
Trainable params: 73,418
Non-trainable params: 0
_________________________________________________________________
```

[7]:

```
model.compile(optimizer='rmsprop',
             loss='categorical_crossentropy',
             metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=100, validation_data=(x_val,␣
 ↪y_val),verbose=0)
```

[8]:

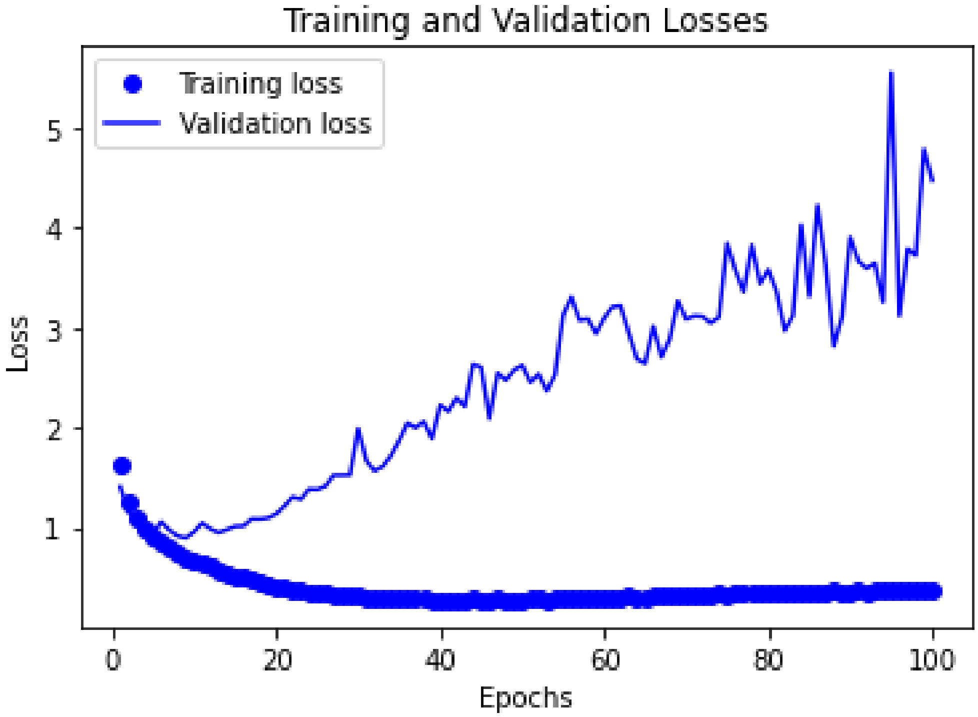
```python
import matplotlib.pyplot as plt

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_2a_lossplot.png')
```



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

[9]:

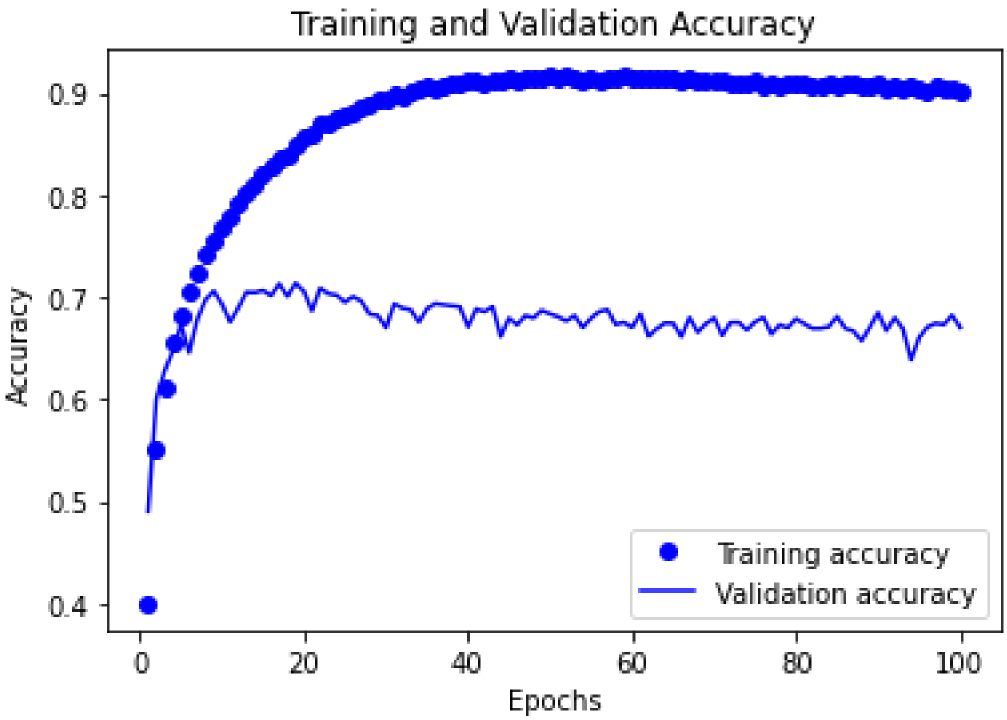
```python
import matplotlib.pyplot as plt

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_2a_accplot.png')
```



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

[10]:
```
#retrain the model and evaluate on test
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Preprocess the data (these are NumPy arrays)
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255

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

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=10)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/10
1563/1563 [==============================] - 37s 23ms/step - loss: 0.9409 -
accuracy: 0.7779
Epoch 2/10
1563/1563 [==============================] - 35s 22ms/step - loss: 0.7938 -
accuracy: 0.7757
Epoch 3/10
1563/1563 [==============================] - 31s 20ms/step - loss: 0.7645 -
accuracy: 0.7760
Epoch 4/10
1563/1563 [==============================] - 30s 19ms/step - loss: 0.6877 -
accuracy: 0.7903
Epoch 5/10
1563/1563 [==============================] - 30s 19ms/step - loss: 0.7088 -
accuracy: 0.7814
Epoch 6/10
1563/1563 [==============================] - 28s 18ms/step - loss: 0.6708 -
accuracy: 0.7922
Epoch 7/10
1563/1563 [==============================] - 27s 17ms/step - loss: 0.6484 -
accuracy: 0.7922
Epoch 8/10
1563/1563 [==============================] - 24s 15ms/step - loss: 0.6538 -
accuracy: 0.7947
Epoch 9/10
1563/1563 [==============================] - 18s 11ms/step - loss: 0.6403 -
accuracy: 0.7944
Epoch 10/10
1563/1563 [==============================] - 19s 12ms/step - loss: 0.6341 -
accuracy: 0.7963
313/313 [==============================] - 2s 6ms/step - loss: 1.3246 -
accuracy: 0.6260
```

[11]:

```
model.save('results/6_2a_model.h5')
```

[12]:

```
prediction_results = model.predict(x_test)
```

[13]:

```
#write metrics to file
with open('results/6_2a_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]))
```

[14]:

```
predictions = pd.DataFrame(prediction_results,␣
 ↪columns=['0','1','2','3','4','5','6','7','8','9'])
predictions.to_csv('results/6_2a_predictions.csv', index=False)
```

[ ]:

```
```