Assignment 6.1

October 10, 2022

[1]:
```
#  Load Dataset
from keras.datasets import mnist
from keras.utils import to_categorical
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

[2]:
```
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

[3]:
```
# splitting the model into train and validation

x_val = train_images[:10000]
partial_x_train = train_images[10000:]
y_val = train_labels[:10000]
partial_y_train = train_labels[10000:]
```

[4]:
```
# Instantiating a small convnet
from keras import layers
from keras import models
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
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'))
```

[5]:
```
# Adding a classifier on top of the convnet
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

[6]:
```
# Compile the model
model.
 ↪compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
```

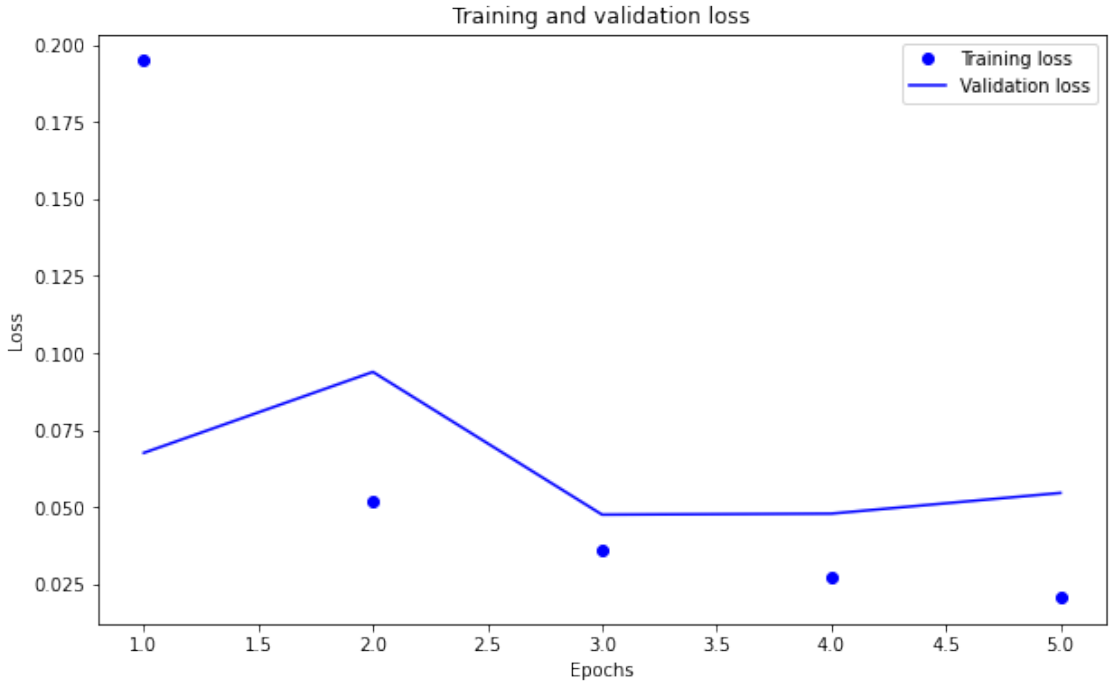
[7]:
```
# train the model

history = model.fit(partial_x_train, partial_y_train, epochs=5,␣
↪batch_size=64,validation_data=(x_val, y_val))
```

```
Epoch 1/5
782/782 [==============================] - 12s 16ms/step - loss: 0.1949 -
accuracy: 0.9388 - val_loss: 0.0675 - val_accuracy: 0.9791
Epoch 2/5
782/782 [==============================] - 12s 16ms/step - loss: 0.0518 -
accuracy: 0.9838 - val_loss: 0.0938 - val_accuracy: 0.9710
Epoch 3/5
782/782 [==============================] - 12s 15ms/step - loss: 0.0360 -
accuracy: 0.9890 - val_loss: 0.0474 - val_accuracy: 0.9861
Epoch 4/5
782/782 [==============================] - 12s 15ms/step - loss: 0.0274 -
accuracy: 0.9913 - val_loss: 0.0478 - val_accuracy: 0.9866
Epoch 5/5
782/782 [==============================] - 12s 15ms/step - loss: 0.0207 -
accuracy: 0.9937 - val_loss: 0.0545 - val_accuracy: 0.9858
```

[8]:
```
# Training history

history_dict = history.history
history_dict.keys()
```

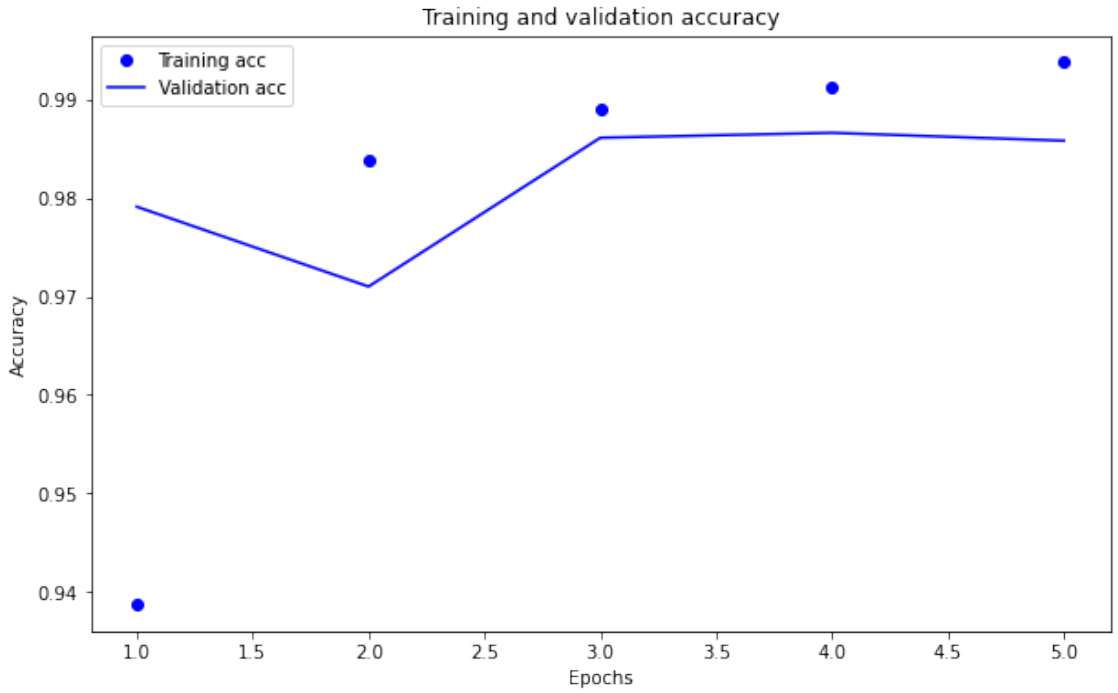
[8]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[9]:
```
# Training and validation loss

import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))

loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show();
```



[10]:
```
# Plot the training and validation accuracy

plt.clf()
plt.figure(figsize=(10,6))
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show();
```

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



[11]:
```
# evaluate the model

test_loss, test_acc = model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0386 -
accuracy: 0.9875
```

[12]:
```
print(f'Test accuracy: {test_acc*100:.1f}%')
print(f'Test loss: {test_loss:.3f}')
```

```
Test accuracy: 98.8%
Test loss: 0.039
```

[ ]:
```

```