Assignment 6.1

In [1]:
```
from tensorflow.keras.datasets import mnist
from keras.utils import to_categorical
from keras import layers
from keras import models
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

Instantiating a Small Covnet

In [2]:
```
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'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

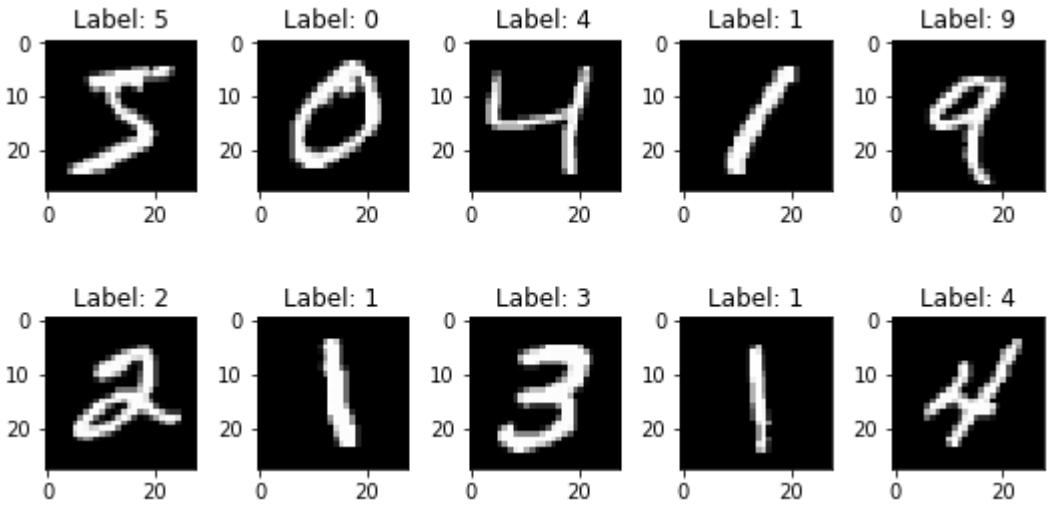
In [3]:
```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                36928
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

Load Data Set

In [4]:
```
# Loading the data into memory
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

In [5]:
```
fig, axes = plt.subplots(2, 5, figsize=(7.5,4))
for i in range(10):
    ax = axes[i//5, i%5]
    ax.imshow(train_images[i], cmap='gray')
    ax.set_title('Label: {}'.format(train_labels[i]))
plt.tight_layout()
plt.show()
```



Data Preparation

In [6]:
```
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)
```

Validation Dataset

In [7]:
```
# Splitting the data into validation and train set
train_images_val = train_images[:10000]
train_images = train_images[10000:]

train_labels_val = train_labels[:10000]
train_labels = train_labels[10000:]
```

Building a Neural Network

In [8]:
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Training the neural network with partial_x_train and partial_y_train
history = model.fit(train_images,
                    train_labels,
                    epochs=5,
                    batch_size=64,
                    validation_data = (train_images_val, train_labels_val),
                    verbose=True)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/5
50000/50000 [==============================] - 28s 569us/step - loss: 0.1912
- accuracy: 0.9397 - val_loss: 0.0783 - val_accuracy: 0.9768
Epoch 2/5
50000/50000 [==============================] - 29s 570us/step - loss: 0.0502
- accuracy: 0.9843 - val_loss: 0.0645 - val_accuracy: 0.9798
Epoch 3/5
50000/50000 [==============================] - 28s 567us/step - loss: 0.0350
- accuracy: 0.9886 - val_loss: 0.0387 - val_accuracy: 0.9888
Epoch 4/5
50000/50000 [==============================] - 29s 578us/step - loss: 0.0252
- accuracy: 0.9922 - val_loss: 0.0336 - val_accuracy: 0.9908
Epoch 5/5
50000/50000 [==============================] - 29s 575us/step - loss: 0.0199
- accuracy: 0.9940 - val_loss: 0.0417 - val_accuracy: 0.9892
```

In [9]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels)

test_acc
```

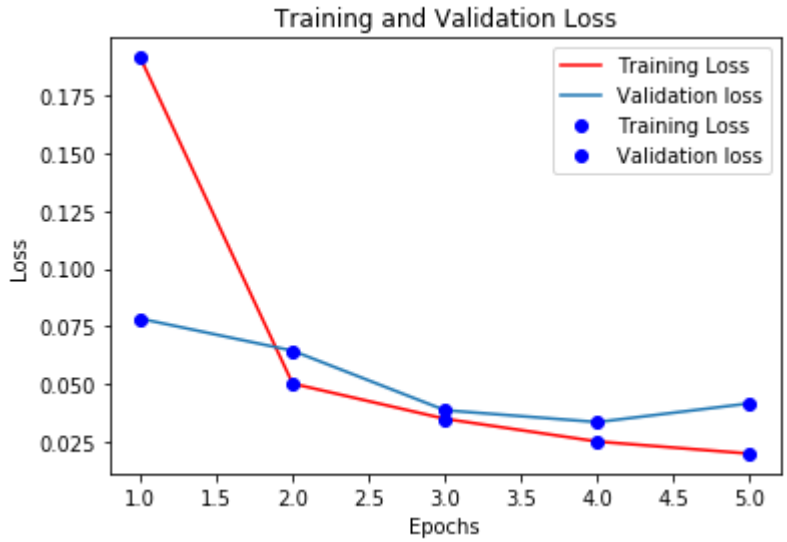
```
10000/10000 [==============================] - 2s 205us/step
```

Out[9]:
```
0.9900000095367432
```

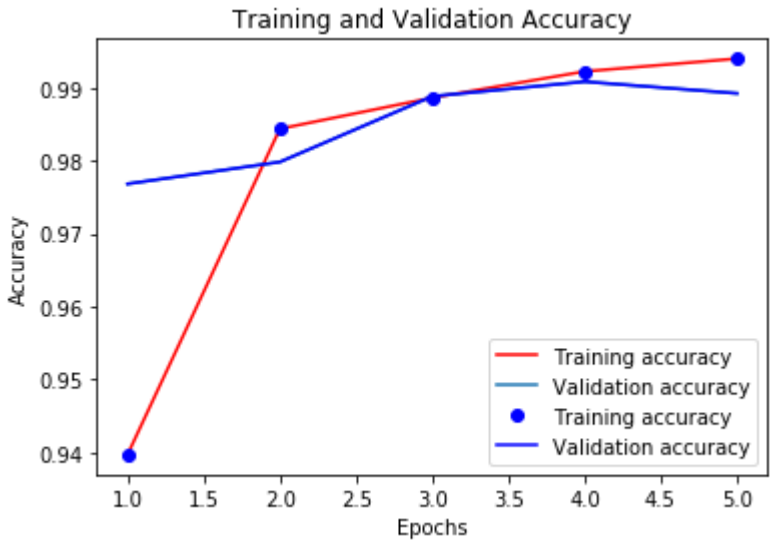
In [10]:
```
history_dict = history.history
history_dict.keys()
```

Out[10]:
```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

In [21]:
```
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1,len(acc)+ 1)

plt.plot(epochs, loss_values,  "r-", label = 'Training Loss')
plt.plot(epochs, val_loss_values, label = 'Validation loss')
plt.plot(epochs, loss_values,  'bo', label = 'Training Loss')
plt.plot(epochs, val_loss_values, 'bo', label = 'Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



In [22]:
```
plt.plot(epochs, acc,  "r-", label = 'Training accuracy')
plt.plot(epochs, val_acc, label = 'Validation accuracy')
plt.plot(epochs, acc,  'bo', label = 'Training accuracy')
plt.plot(epochs, val_acc, 'b',label = 'Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [ ]: