Neural Network

December 14, 2020

[3]:
```
## Used train.csv and test.csv which are csv files that comprise of all the␣
↪training
# and testing data from the MNIST handwritten Dataset
import numpy as np
import sklearn.preprocessing
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical, plot_model
import matplotlib.pyplot as plt
import seaborn as sns
# The training data has 42000 entries and 785 features
train_digits = pd.read_csv("train.csv")
```

[4]:
```
# Creating sections for training and testing data
# Creating training and test sets
# Splitting the data into train and test
A_matrix = train_digits.iloc[:, 1:]
Labels_A = train_digits.iloc[:, 0]

# Processing the images before analysis using scale function from sklearn.
↪preprocessing
# Scale is used to standardise the pixels(features) in the image. This needs to␣
↪be done because our dataset is sparse/widely soread out.
# Which can cause issues with our model being statistically skewed. Scaling the␣
↪data in this manner
# makes  the matrix entries have 0 mean and the standard variation of 1 in each␣
↪field.
A_matrix = sklearn.preprocessing.scale(A_matrix)

# Splitting the data with 42k datapoints into 30% training data and 10% testing␣
↪data
# The stratify property was used to ensure the data is split with even label␣
↪distribution
```

```
Train_A, Test_A, Train_y, Test_y = train_test_split(A_matrix, Labels_A,␣
↪test_size = 0.10, train_size=0.30, random_state=10, stratify=Labels_A)
```

[5]:
```
# ONEHOT ENCODING
Train_y= to_categorical(Train_y, 10)
Test_y = to_categorical(Test_y, 10)
```

0.1 Altering the Number of Hidden Layers in the Neural Network

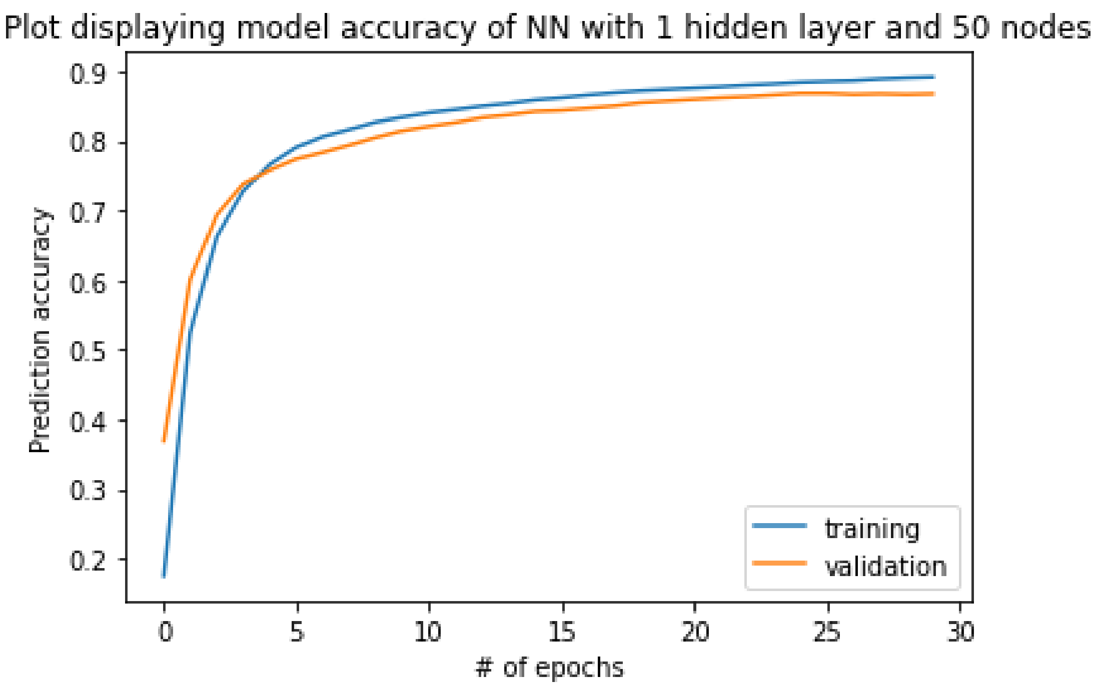
0.2 1 hidden layer with 50 nodes and sigmoid activation function

[5]:
```
# Using Sequential Keras model because we are creating a plain stack of layers
# Using Dense Layers so that our neural network is fully connected
# Using SGD as the Optimisation Algorithm because its the only one I fully␣
↪understand from this course
# Creating a single hidden layer with 50 nodes
# Because our model's output is one-hot encoded, I used softmax activation for␣
↪the output layer.
# This normalizes the values from the 10 output
# nodes such that the sum of all 10 values =1

## 1 hidden layers with 50 nodes
nn = Sequential([
    Dense(50, activation = 'sigmoid', input_shape = (784,)),
    Dense(10, activation = 'softmax')
])
```

[8]:
```
# Compiling previously defined model
nn.compile(metrics=['accuracy'], optimizer="sgd",␣
↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn.fit(Train_A, Train_y, batch_size=128, epochs=30, verbose=False,␣
↪validation_split=.1)
loss, prediction_correctness  = nn.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
```

```
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.888
```

0.3 Increasing to 2 hidden layers and 50 nodes

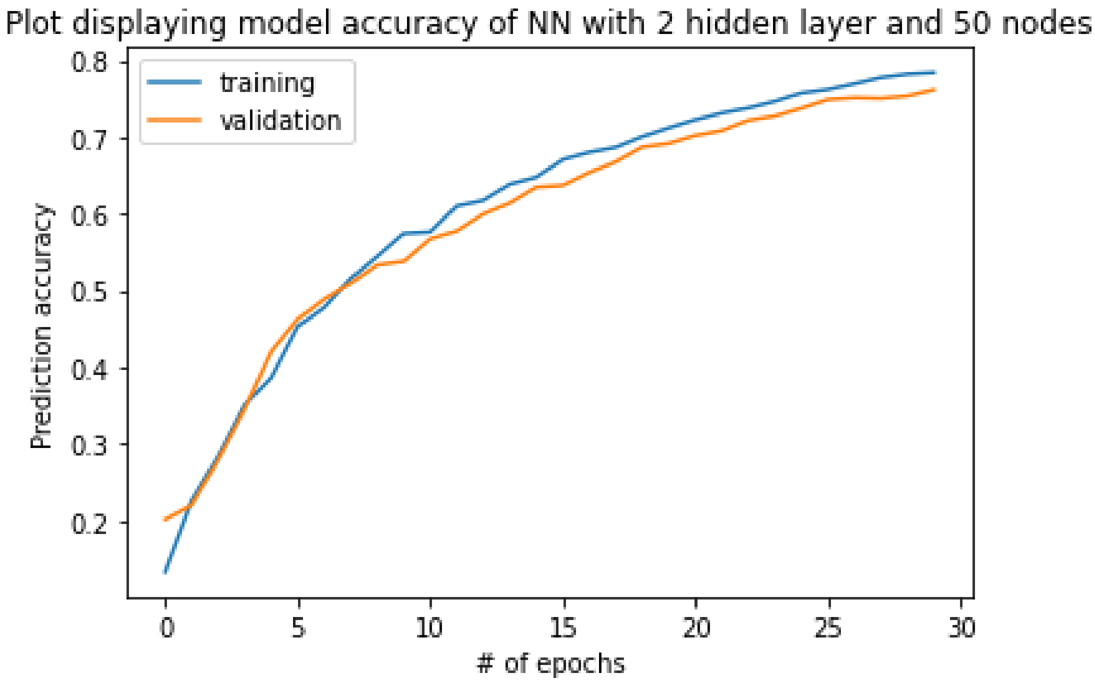
[9]:
```
# Creating a single hidden layer with 50 nodes
nn2 = Sequential()
## 2 hidden layers with 50 nodes each
nn2 = Sequential([
    Dense(50, activation = 'sigmoid', input_shape = (784,)),
    Dense(50, activation = 'sigmoid'),
    Dense(10, activation = 'softmax')
])

nn2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 50)                39250
_________________________________________________________________
dense_3 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_4 (Dense)              (None, 10)                510
=================================================================
Total params: 42,310
Trainable params: 42,310
Non-trainable params: 0
_________________________________________________________________
```

[10]:

```
# Compiling previously defined model
nn2.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn2.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn2.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.781
```

0.4 Increasing to 3 hidden layers and 50 nodes

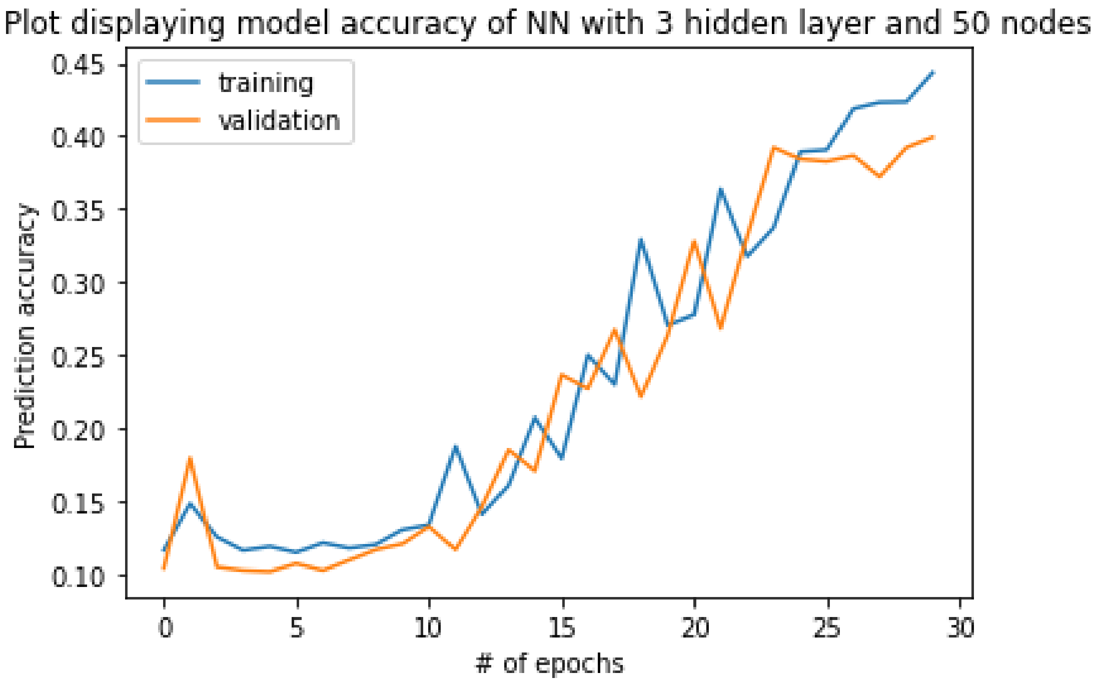
[6]:
```
# Creating a single hidden layer with 50 nodes
nn3 = Sequential()
## 3 hidden layers with 50 nodes each
nn3 = Sequential([
    Dense(50, activation = 'sigmoid', input_shape = (784,)),
    Dense(50, activation = 'sigmoid'),
    Dense(50, activation = 'sigmoid'),
    Dense(10, activation = 'softmax')
])
nn3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 50)                39250
_________________________________________________________________
dense_3 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_4 (Dense)              (None, 50)                2550
```

```
_________________________________________________________________
dense_5 (Dense)              (None, 10)                510
=================================================================
Total params: 44,860
Trainable params: 44,860
Non-trainable params: 0
_________________________________________________________________
```

[7]:
```
# Compiling previously defined model
nn3.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn3.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn3.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.421
```

1 CHANGING THE # of NODES INSTEAD

1.1 1 hidden layer with 100 nodes

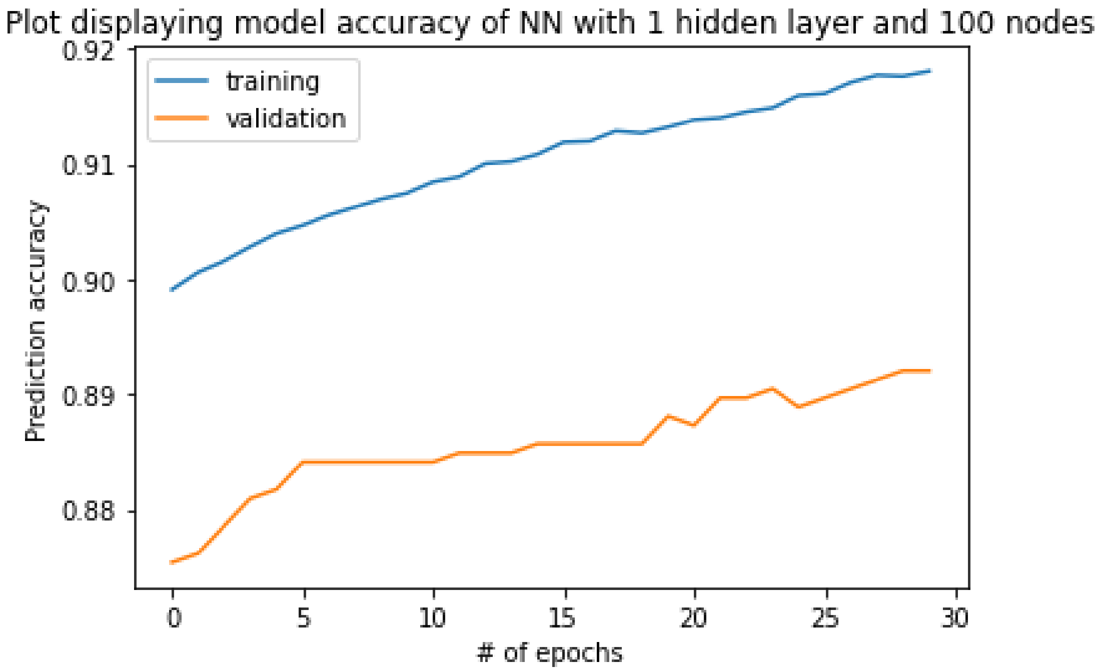
[8]:
```
# Creating a single hidden layer with 32 nodes
nn4 = Sequential()
## 2 hidden layers with 50 nodes each
nn4 = Sequential([
    Dense(100, activation = 'sigmoid', input_shape = (784,)),
    Dense(10, activation = 'softmax')
])
nn4.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 100)               78500
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1010
=================================================================
```

```
Total params: 79,510
Trainable params: 79,510
Non-trainable params: 0
_________________________________________________________________
```

[10]:
```
# Compiling previously defined model
nn4.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn4.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn4.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.911
```

1.2 1 hidden layer with 500 nodes

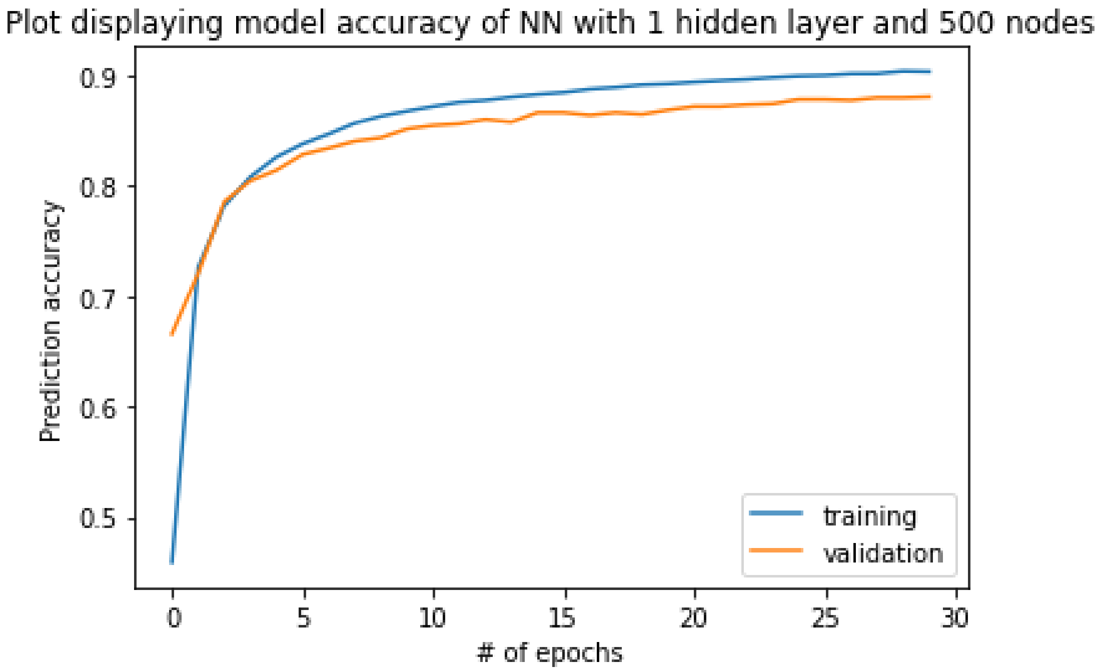
[11]:
```python
# Creating a single hidden layer with 500 nodes
nn5 = Sequential()
## 2 hidden layers with 50 nodes each
nn5 = Sequential([
    Dense(500, activation = 'sigmoid', input_shape = (784,)),
    Dense(10, activation = 'softmax')
])
nn5.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 500)               392500
_________________________________________________________________
dense_9 (Dense)              (None, 10)                5010
=================================================================
Total params: 397,510
Trainable params: 397,510
Non-trainable params: 0
```

----

[12]:
```python
# Compiling previously defined model
nn5.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn5.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn5.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.905
```

1.3 1 hidden layer with 1000 nodes

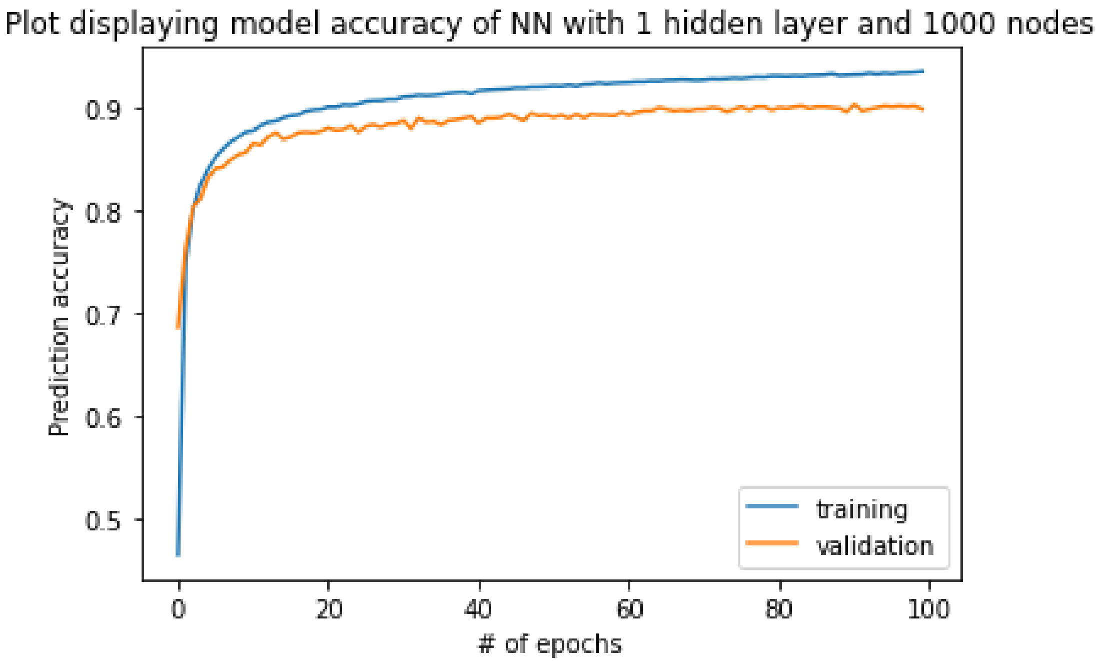
[6]:
```
# Creating a single hidden layer with 1000 nodes
nn6 = Sequential()
## 2 hidden layers with 50 nodes each
nn6 = Sequential([
    Dense(1000, activation = 'sigmoid', input_shape = (784,)),
    Dense(10, activation = 'softmax')
])
nn6.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1000)              785000
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10010
=================================================================
Total params: 795,010
Trainable params: 795,010
Non-trainable params: 0
```

----

[7]:
```
# Compiling previously defined model
nn6.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn6.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn6.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.919
```

1.4 Testing to find the best accuracy model

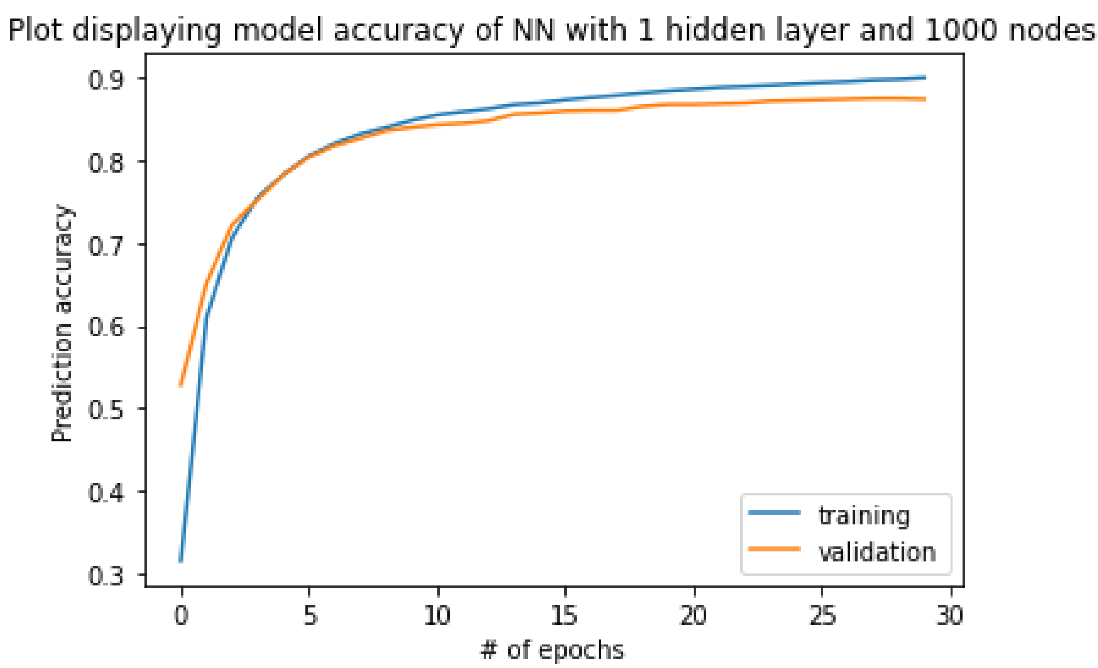
[12]:
```
# Creating a single hidden layer with 32 nodes
nn7 = Sequential()
## 2 hidden layers with 50 nodes each
nn7 = Sequential([
    Dense(90, activation = 'sigmoid', input_shape = (784,)),
    Dense(10, activation = 'softmax')
])
nn7.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 90)                70650
_________________________________________________________________
dense_5 (Dense)              (None, 10)                910
=================================================================
Total params: 71,560
Trainable params: 71,560
Non-trainable params: 0
_________________________________________________________________
```

[13]:

```
# Compiling previously defined model
nn7.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn7.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn7.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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



```
Test accuracy: 0.896
```

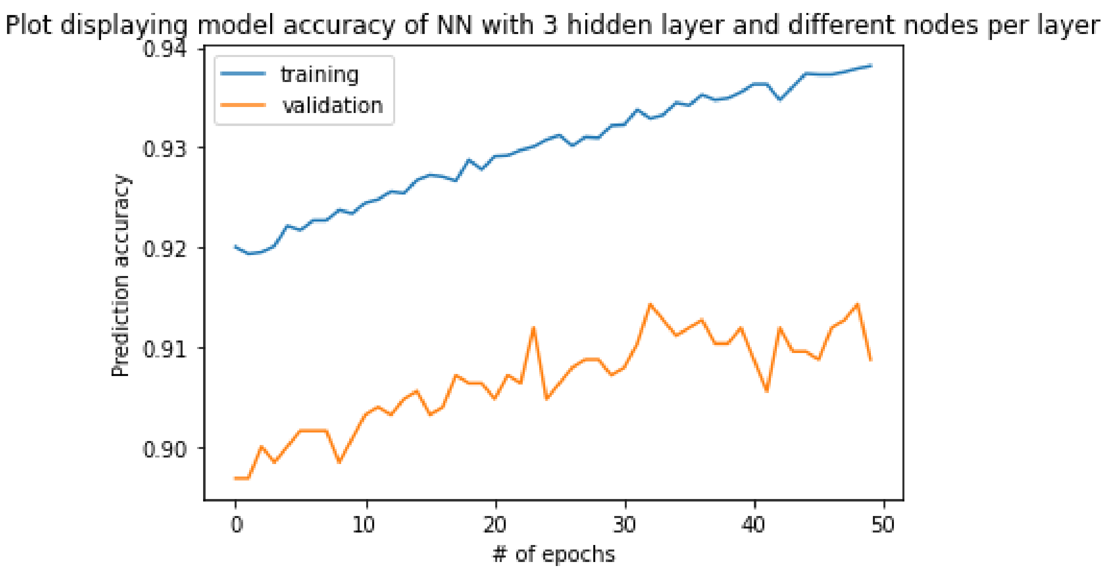
[16]:
```
# Creating a single hidden layer with 32 nodes
nn8 = Sequential()
## 3 hidden layers with different nodes each
nn8 = Sequential([
    Dense(800, activation = 'sigmoid', input_shape = (784,)),
    Dense(300, activation = 'relu', input_shape = (784,)),
    Dense(300, activation = 'sigmoid', input_shape = (784,)),
    Dense(10, activation = 'softmax')
])
nn8.summary()
```

```
Model: "sequential_12"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_17 (Dense)             (None, 800)               628000

_________________________________________________________________
dense_18 (Dense)             (None, 300)               240300

_________________________________________________________________
dense_19 (Dense)             (None, 300)               90300

_________________________________________________________________
dense_20 (Dense)             (None, 10)                3010
=================================================================
Total params: 961,610
Trainable params: 961,610
Non-trainable params: 0

_________________________________________________________________
```

[19]:
```
# Compiling previously defined model
nn8.compile(metrics=['accuracy'], optimizer="sgd",␣
 ↪loss='categorical_crossentropy')

#Using 10% of the sample size of testingvalidation
model_test = nn8.fit(Train_A, Train_y, batch_size=128, epochs=30,␣
 ↪verbose=False, validation_split=.1)
loss, prediction_correctness  = nn8.evaluate(Test_A, Test_y)

#Printing the keys of the test model dict to see what variables to plot
print(model_test.model_test.keys())


#Creating prediction and testing plot for comparison
plt.plot(model_test.model_test['accuracy'])
```

```
plt.plot(model_test.model_test['val_accuracy'])
plt.legend(['training', 'validation'])
plt.xlabel('# of epochs')
plt.ylabel('Prediction accuracy')
plt.title('Plot displaying model accuracy of NN with 1 hidden layer and 50␣
 ↪nodes')
plt.show()

#Printing accuracy of model
print('Test accuracy: ', np.round(prediction_correctness,3))
```

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




```
Test accuracy: 0.92
```