Deep Learning Lab Session

First Lab Session - 3 Hours

Artificial Neural Networks for Handwritten Digits Recognition

Student 1: Alessandro Gaballo **Student 2:** Jonas Wacker

The aim of this session is to practice with Artificial Neural Networks. Answers and experiments should be made by groups of one or two students. Each group should fill and run appropriate notebook cells.

To generate your final report, use print as PDF (Ctrl+P). Do not forget to run all your cells before generating your final report and do not forget to include the names of all participants in the group. The lab session should be completed by April 7th 2017.

Introduction

In this session, your will implement, train and test a Neural Network for the Handwritten Digits Recognition problem [1] (http://yann.lecun.com/exdb/mnist/) with different settings of hyper parameters. You will use the MNIST dataset which was constructed from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST). Images of digits were taken from a variety of scanned documents, normalized in size and centered.

Figure 1: MNIST digits examples

This assignment includes a written part of programms to help you understand how to build and train your neural net and then to test your code and get restults.

1. NeuralNetwork.py (NeuralNetwork.py)
2. transfer_functions.py (transfer_functions.py)
3. utils.py (utils.py)

Functions defined inside the python files mentionned above can be imported using the python command : from filename import *

You will use the following libraries:

1. numpy (http://cs231n.github.io/python-numpy-tutorial/): for creating arrays and using methods to manipulate arrays.
2. matplotlib (http://matplotlib.org/): for making plots

Section 1 : My First Neural Network

Part 1: Before designing and writing your code, you will first work on a neural network by hand. Consider the above Neural network with two inputs $X = (x1, x2)$, one hidden layers and a single output unit (y). The initial weights are set to random values. Neurons 6 and 7 represent the bias. Bias values are equal to 1.
Training sample, X = (0.8, 0.2), whose class label is Y=0.4.

Assume that the neurons have a Sigmoid activation function $f(x) = \frac{1}{(1+e^{-x})}$ and the learning rate μ=1

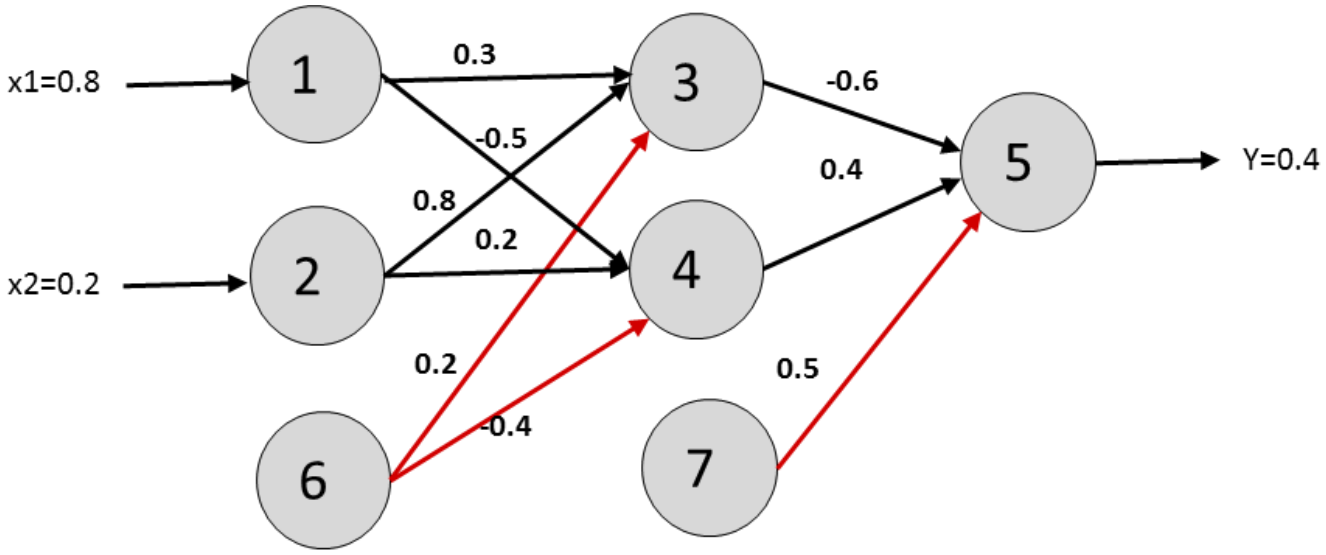


Figure 2: Neural network

Question 1.1.1: Compute the new values of weights $w_{i,j}$ after a forward pass and a backward pass. $w_{i,j}$ is the weight of the connexion between neuron i and neuron j.

#Your answer goes here :

$w_{1,3} = 0.3043$

$w_{1,4} = -0.5027$

$w_{2,3} = 0.8011$

$w_{2,4} = 0.1993$

$w_{6,3} = 0.2054$

$w_{6,4} = -0.4034$

$w_{3,5} = 0.6254$

$w_{4,5} = 0.3875$

$w_{7,5} = 0.4606$

Part 2: Neural Network Implementation

Please read all source files carefully and understand the data structures and all functions. You are to complete the missing code. First you should define the neural network (using the NeuralNetwork class, see in the NeuralNetwork.py (NeuralNetwork.py) file) and reinitialise weights. Then you will to complete the Feed Forward and the Back-propagation functions.

Question 1.2.1: Define the neural network corresponding to the one in part 1

In [9]:

```
from NeuralNetwork import *
import numpy as np
#create the network
my_first_net = NeuralNetwork(2, 2, 1, learning_rate=1)
```

In [12]:

```
#Data preparation
X=[0.8,0.2]
Y=[0.4]
data=[]
data.append(X)
data.append(Y)

#initialize weights
wi=np.array([[0.3,-0.5],[0.8,0.2],[0.2,-0.4]])
wo=np.array([[-0.6],[0.4],[0.5]])
my_first_net.weights_initialisation(wi,wo)
print(my_first_net.W_input_to_hidden)
print(my_first_net.W_hidden_to_output)
```

```
[[ 0.3 -0.5]
 [ 0.8  0.2]
 [ 0.2 -0.4]]
[[-0.6]
 [ 0.4]
 [ 0.5]]
```

Question 1.2.2: Implement the Feed Forward function (feedForward(X) in the NeuralNetwork.py file)

In [3]:

```
# Implement it in the NeuralNetwork.py file and when finalised copy and paste your
def transfer_function(self, x):
    return 1.0 / (1.0 + np.exp(-x))

def feedForward(self, inputs):
    self.a_input = np.append(inputs, 1)
    a_hidden_without_bias = np.dot(self.a_input, self.W_input_to_hidden)
    self.a_hidden = np.append(a_hidden_without_bias, 0)
    self.o_hidden = self.transfer_function(self.a_hidden)
    self.o_hidden[-1] = 1
    self.a_output = np.dot(self.o_hidden, self.W_hidden_to_output)
    self.o_output = self.transfer_function(self.a_output)
    return self.o_output
```

Check your network outputs the expected value (the one you computed in question 1.1)

In [13]:

```
#test my  Feed Forward function
Output_activation=my_first_net.feedForward(X)
print("output activation =%.3f" %(Output_activation))
```

```
output activation =0.560
```

Question 1.2.3: Implement the Back-propagation Algorithm (backPropagate(Y) in the NeuralNetwork.py file)

In [5]:

```
# Implement it in the NeuralNetwork.py file and when finalised copy and paste your
def backPropagate(self, targets):

    # calculate error terms for output
    self.errors = self.o_output - targets
    delta_e_u_output = self.errors * self.o_output * (1 - self.o_output)
    delta_e_u_horizontal = np.matrix(delta_e_u_output)
    o_hidden_vertical = np.matrix(self.o_hidden).T

    delta_e_w_output = np.dot(o_hidden_vertical, delta_e_u_horizontal)

    # calculate error terms for hidden
    delta_e_u_hidden = np.dot(self.W_hidden_to_output, delta_e_u_output) * self.o_h
    delta_e_u_horizontal = np.matrix(delta_e_u_hidden)
    o_input_vertical = np.matrix(self.a_input).T
    delta_e_w_hidden = np.dot(o_input_vertical, delta_e_u_horizontal)
    # delete last column
    # delta_e_w_hidden = delta_e_w_hidden[:,0:delta_e_w_hidden.shape[1]-1]
    delta_e_w_hidden = np.delete(delta_e_w_hidden, -1, 1)
    # update output weights
    self.W_hidden_to_output -= self.learning_rate * delta_e_w_output
    # update input weights
    self.W_input_to_hidden -= self.learning_rate * delta_e_w_hidden

    return np.square(self.errors).sum()/2
```

Check the gradient values and weight updates are correct (similar to the ones you computed in question 1.1)

In [14]:

```
#test my  Back-propagation function
my_first_net.backPropagate(Y)
#Print weights after backpropagation
print('New input weights\n', my_first_net.W_input_to_hidden)
print('New output weights\n', my_first_net.W_hidden_to_output)
```

```
New input weights
 [[ 0.30432265 -0.50273473]
 [ 0.80108066  0.19931632]
 [ 0.20540332 -0.40341841]]
New output weights
 [[-0.62541468]
 [ 0.38745727]
 [ 0.46063746]]
```

Your Feed Forward and Back-Propagation implementations are working, Great!! Let's tackle a real world problem.

Section 2 : The MNIST Challenge!

Data Preparation

The MNIST dataset consists of handwritten digit images it contains 60,000 examples for the training set and 10,000 examples for testing. In this Lab Session, the official training set of 60,000 is divided into an actual training set of 50,000 examples, 10,000 validation examples and 10,000 examples for test. All digit images have been size-normalized and centered in a fixed size image of 28 x 28 pixels. The images are stored in byte form you will use the NumPy python library to read the data files into NumPy arrays that we will use to train the ANN.

The MNIST dataset is available in the Data folder. To get the training, testing and validation data, run the the load_data() function.

In [2]:

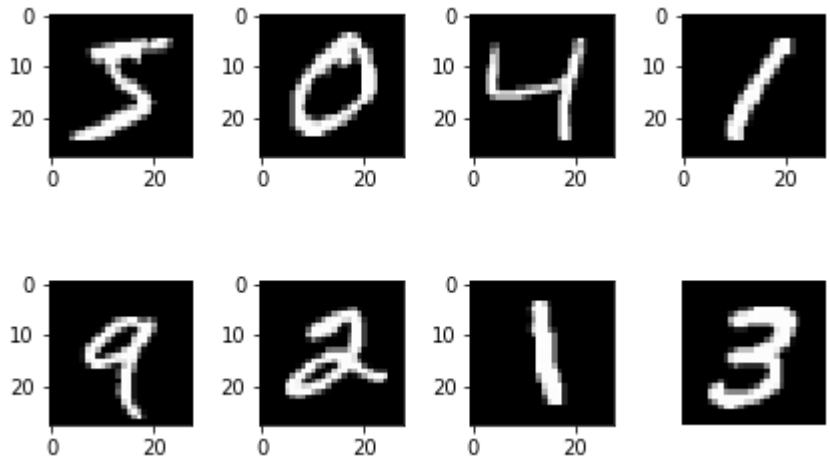
```
from utils import *
training_data, validation_data, test_data=load_data()
```

```
Loading MNIST data .....
Done.
```

MNIST Dataset Digits Visualisation

In [3]:

```
ROW = 2
COLUMN = 4
for i in range(ROW * COLUMN):
    # train[i][0] is i-th image data with size 28x28
    image = training_data[i][0].reshape(28, 28)
    plt.subplot(ROW, COLUMN, i+1)
    plt.imshow(image, cmap='gray')  # cmap='gray' is for black and white picture.
plt.axis('off')  # do not show axis value
plt.tight_layout()   # automatic padding between subplots
plt.show()
```



Part 1: Creating the Neural Networks

The input layer of the neural network contains neurons encoding the values of the input pixels. The training data for the network will consist of many 28 by 28 pixel images of scanned handwritten digits, and so the input layer contains 784=28×28 neurons. The second layer of the network is a hidden layer, we set the neuron number in the hidden layer to 30. The output layer contains 10 neurons.

Question 2.1.1: Create the network described above using the NeuralNetwork class

In [4]:

```
#create the network
from NeuralNetwork import NeuralNetwork

input_nodes = 784
hidden_nodes = 30
output_nodes = 10

my_mnist_net = NeuralNetwork(input_nodes, hidden_nodes, output_nodes)
```

Question 2.1.2: Add the information about the performance of the neural network on the test set at each epoch

In [5]:

```
test_accuracy=my_mnist_net.predict(test_data) / 100
```

In [6]:

```
print('Test_Accuracy [epoch = 0]  %-2.2f' % test_accuracy)
```

```
Test_Accuracy [epoch = 0]  10.09
```

Question 2.1.3: Train the Neural Network and comment your findings

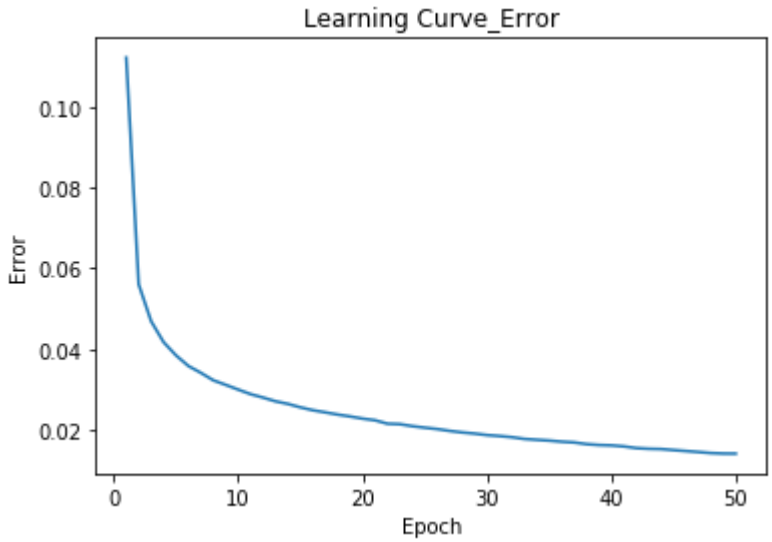
In [7]:

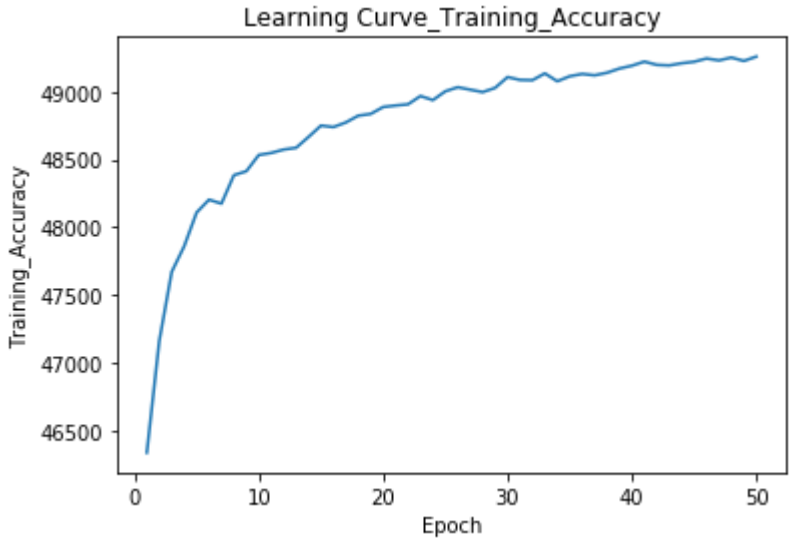
```
test_accuracy = []
#train your network
val_accuracies = my_mnist_net.train(training_data,validation_data)
#save your model in Models/ using a distinguishing name for your model (architectur
```

```
Iteration:  1/50[==============] -Error: 0.1123316059  -Training_Accu
racy:  92.67  -time: 35.69
Validation accuracy 92.7
Iteration:  2/50[==============] -Error: 0.0560563344  -Training_Accu
racy:  94.32  -time: 73.15
Validation accuracy 93.65
Iteration:  3/50[==============] -Error: 0.0468937200  -Training_Accu
racy:  95.34  -time: 110.38
Validation accuracy 94.5
Iteration:  4/50[==============] -Error: 0.0417135517  -Training_Accu
racy:  95.72  -time: 146.80
Validation accuracy 94.96
Iteration:  5/50[==============] -Error: 0.0384278051  -Training_Accu
racy:  96.22  -time: 183.48
Validation accuracy 95.17
Iteration:  6/50[==============] -Error: 0.0357778918  -Training_Accu
racy:  96.41  -time: 221.13
Validation accuracy 95.21
Iteration:  7/50[==============] -Error: 0.0340840941  -Training_Accu
racy:  96.35  -time: 259.54
Validation accuracy 95.03
Iteration:  8/50[==============] -Error: 0.0322334574  -Training_Accu
racy:  96.77  -time: 294.98
Validation accuracy 95.57
Iteration:  9/50[==============] -Error: 0.0311007917  -Training_Accu
racy:  96.83  -time: 332.65
Validation accuracy 95.64
Iteration: 10/50[==============] -Error: 0.0299295064  -Training_Accu
racy:  97.07  -time: 371.07
Validation accuracy 95.72
Iteration: 11/50[==============] -Error: 0.0287945642  -Training_Accu
racy:  97.10  -time: 409.15
Validation accuracy 95.69
Iteration: 12/50[==============] -Error: 0.0279465594  -Training_Accu
racy:  97.15  -time: 445.02
Validation accuracy 95.69
Iteration: 13/50[==============] -Error: 0.0270195310  -Training_Accu
racy:  97.17  -time: 484.16
Validation accuracy 95.66
Iteration: 14/50[==============] -Error: 0.0263753463  -Training_Accu
racy:  97.33  -time: 525.65
Validation accuracy 95.89999999999999
Iteration: 15/50[==============] -Error: 0.0255188896  -Training_Accu
racy:  97.50  -time: 563.54
Validation accuracy 96.09
Iteration: 16/50[==============] -Error: 0.0248074819  -Training_Accu
racy:  97.48  -time: 602.14
Validation accuracy 95.88
Iteration: 17/50[==============] -Error: 0.0242756412  -Training_Accu
racy:  97.55  -time: 640.06
Validation accuracy 95.78
Iteration: 18/50[==============] -Error: 0.0237267106  -Training_Accu
racy:  97.65  -time: 680.77
Validation accuracy 95.91
```

```
Iteration: 19/50[==============] -Error: 0.0232591624  -Training_Accu
racy:  97.67  -time: 720.22
Validation accuracy 95.99
Iteration: 20/50[==============] -Error: 0.0227244074  -Training_Accu
racy:  97.77  -time: 758.08
Validation accuracy 95.98
Iteration: 21/50[==============] -Error: 0.0223126635  -Training_Accu
racy:  97.80  -time: 795.99
Validation accuracy 96.02000000000001
Iteration: 22/50[==============] -Error: 0.0214598691  -Training_Accu
racy:  97.81  -time: 835.07
Validation accuracy 95.93
Iteration: 23/50[==============] -Error: 0.0213472628  -Training_Accu
racy:  97.94  -time: 877.37
Validation accuracy 96.00999999999999
Iteration: 24/50[==============] -Error: 0.0208509002  -Training_Accu
racy:  97.88  -time: 915.94
Validation accuracy 96.00999999999999
Iteration: 25/50[==============] -Error: 0.0204703033  -Training_Accu
racy:  98.01  -time: 956.01
Validation accuracy 96.00999999999999
Iteration: 26/50[==============] -Error: 0.0201351799  -Training_Accu
racy:  98.07  -time: 995.54
Validation accuracy 95.94
Iteration: 27/50[==============] -Error: 0.0196720400  -Training_Accu
racy:  98.03  -time: 1033.14
Validation accuracy 95.8
Iteration: 28/50[==============] -Error: 0.0192977011  -Training_Accu
racy:  97.99  -time: 1073.86
Validation accuracy 95.88
Iteration: 29/50[==============] -Error: 0.0190109381  -Training_Accu
racy:  98.06  -time: 1115.23
Validation accuracy 95.8
Iteration: 30/50[==============] -Error: 0.0186355214  -Training_Accu
racy:  98.22  -time: 1153.09
Validation accuracy 95.92
Iteration: 31/50[==============] -Error: 0.0184047452  -Training_Accu
racy:  98.17  -time: 1191.87
Validation accuracy 95.88
Iteration: 32/50[==============] -Error: 0.0181160958  -Training_Accu
racy:  98.17  -time: 1231.64
Validation accuracy 95.96000000000001
Iteration: 33/50[==============] -Error: 0.0176798823  -Training_Accu
racy:  98.27  -time: 1269.55
Validation accuracy 95.94
Iteration: 34/50[==============] -Error: 0.0174679701  -Training_Accu
racy:  98.15  -time: 1311.36
Validation accuracy 95.74000000000001
Iteration: 35/50[==============] -Error: 0.0172696712  -Training_Accu
racy:  98.23  -time: 1349.66
Validation accuracy 95.88
Iteration: 36/50[==============] -Error: 0.0169945119  -Training_Accu
racy:  98.26  -time: 1387.44
Validation accuracy 95.86
Iteration: 37/50[==============] -Error: 0.0168154400  -Training_Accu
racy:  98.24  -time: 1425.70
Validation accuracy 95.87
Iteration: 38/50[==============] -Error: 0.0163923258  -Training_Accu
racy:  98.28  -time: 1464.93
Validation accuracy 95.81
Iteration: 39/50[==============] -Error: 0.0161731920  -Training_Accu
```

```
racy:  98.34  -time: 1506.44
Validation accuracy 96.0
Iteration: 40/50[==============] -Error: 0.0160767319  -Training_Accu
racy:  98.38  -time: 1544.38
Validation accuracy 95.84
Iteration: 41/50[==============] -Error: 0.0158286172  -Training_Accu
racy:  98.44  -time: 1584.74
Validation accuracy 95.84
Iteration: 42/50[==============] -Error: 0.0154059630  -Training_Accu
racy:  98.40  -time: 1622.21
Validation accuracy 95.98
Iteration: 43/50[==============] -Error: 0.0152376159  -Training_Accu
racy:  98.39  -time: 1660.04
Validation accuracy 95.91
Iteration: 44/50[==============] -Error: 0.0151676867  -Training_Accu
racy:  98.42  -time: 1703.18
Validation accuracy 95.72
Iteration: 45/50[==============] -Error: 0.0149125588  -Training_Accu
racy:  98.44  -time: 1741.35
Validation accuracy 95.78
Iteration: 46/50[==============] -Error: 0.0146728505  -Training_Accu
racy:  98.49  -time: 1778.95
Validation accuracy 95.98
Iteration: 47/50[==============] -Error: 0.0144329537  -Training_Accu
racy:  98.46  -time: 1817.12
Validation accuracy 95.86
Iteration: 48/50[==============] -Error: 0.0141737024  -Training_Accu
racy:  98.50  -time: 1854.84
Validation accuracy 95.95
Iteration: 49/50[==============] -Error: 0.0140509201  -Training_Accu
racy:  98.45  -time: 1896.82
Validation accuracy 95.89999999999999
Iteration: 50/50[==============] -Error: 0.0140293003  -Training_Accu
racy:  98.52  -time: 1936.84
Validation accuracy 95.97
```





In [8]:

```
my_mnist_net.save('saved_models/NN_MNIST_default_params')
test_accuracy.append(my_mnist_net.predict(test_data)/100)
print('Test_Accuracy  %-2.2f' % test_accuracy[-1])
```

```
Test_Accuracy  96.07
```

We obtain a test accuracy of 96%, which is a very decent result. Especially the test accuracy is just 2% less than the training accuracy, wich means that we did not overfit the model. The validation accuracy also follows the trend of the training accuracy.

Question 2.1.4: Guess digit, Implement and test a python function that predict the class of a digit (the folder images_test contains some examples of images of digits)

In [9]:

```
#Your implementation goes here

#DON'T KNOW HOW TO READ THE IMG
import os
from scipy import misc
def guess_digit(nn, sample):
    prediction = nn.feedForward(sample)
    return np.argmax(prediction)

dirname = 'Images_test'

for image in os.listdir(dirname):
    data = misc.imread(os.path.join(dirname, image), flatten=True)
    resized = misc.imresize(data, (28,28))
    print(guess_digit(my_mnist_net, resized))
```

```
5
5
3
```

The training has been made on images that have a black background and a white digit, in the folder

Images_test only one picture respects that format, that's the main reason of the wrong classification. A possible solution is to recognize the images with a white background (histogram of colors) and invert the colors feeding them into the prediction procedure.

Part 2: Change the neural network structure and parameters to optimize performance

Question 2.2.1: Change the learning rate (0.001, 0.1, 1.0 , 10). Train the new neural nets with the original specifications (Part 2.1), for 50 iterations. Plot test accuracy vs iteration for each learning rate on the same graph. Report the maximum test accuracy achieved for each learning rate. Which one achieves the maximum test accuracy?

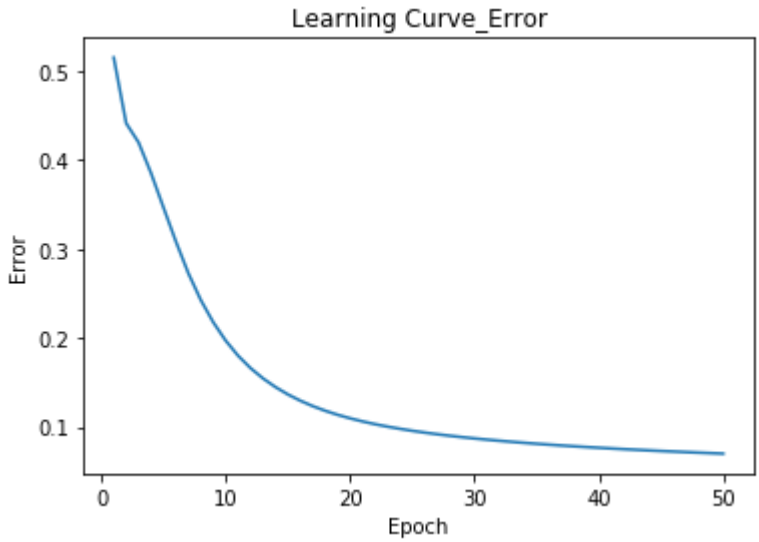
In [10]:

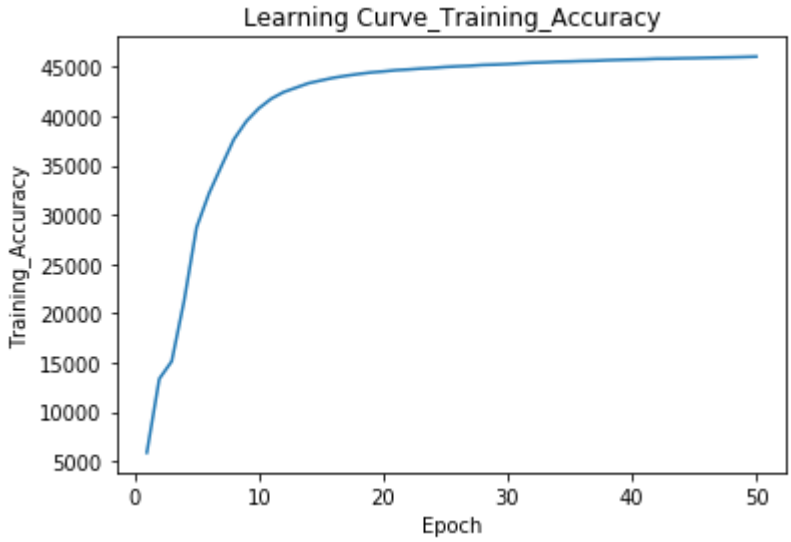
```
#Your implementation with a learning rate of 0.001 goes here

learning_rate = 0.001

my_mnist_net_1 = NeuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_ra
val_accuracies_1 = my_mnist_net_1.train(training_data,validation_data)
test_accuracy.append(my_mnist_net_1.predict(test_data)/100)

print('Learning rate %f, Test_Accuracy  %-2.2f' % (learning_rate, test_accuracy[-1]

my_mnist_net_1.save('saved_models/NN_MNIST_0-001')
```

```
Iteration:  1/50[==============] -Error: 0.5148340507  -Training_Accu
racy:  11.75  -time: 36.50
Validation accuracy 11.83
Iteration:  2/50[==============] -Error: 0.4411256002  -Training_Accu
racy:  26.72  -time: 74.22
Validation accuracy 27.339999999999996
Iteration:  3/50[==============] -Error: 0.4195128437  -Training_Accu
racy:  30.29  -time: 112.64
Validation accuracy 30.19
Iteration:  4/50[==============] -Error: 0.3850714665  -Training_Accu
racy:  42.61  -time: 154.43
Validation accuracy 41.980000000000004
Iteration:  5/50[==============] -Error: 0.3464431099  -Training_Accu
racy:  57.47  -time: 193.38
Validation accuracy 57.989999999999995
Iteration:  6/50[==============] -Error: 0.3082212685  -Training_Accu
racy:  64.36  -time: 233.46
Validation accuracy 65.27
Iteration:  7/50[==============] -Error: 0.2726447611  -Training_Accu
racy:  69.91  -time: 270.59
Validation accuracy 70.67
Iteration:  8/50[==============] -Error: 0.2425447972  -Training_Accu
racy:  75.33  -time: 308.19
Validation accuracy 76.21
Iteration:  9/50[==============] -Error: 0.2176486932  -Training_Accu
racy:  78.99  -time: 347.98
Validation accuracy 79.99000000000001
Iteration: 10/50[==============] -Error: 0.1970255513  -Training_Accu
racy:  81.60  -time: 389.94
Validation accuracy 82.77
Iteration: 11/50[==============] -Error: 0.1800372474  -Training_Accu
racy:  83.57  -time: 429.44
Validation accuracy 84.72
Iteration: 12/50[==============] -Error: 0.1661365512  -Training_Accu
racy:  84.92  -time: 467.61
Validation accuracy 86.0
Iteration: 13/50[==============] -Error: 0.1546045304  -Training_Accu
racy:  85.81  -time: 505.21
Validation accuracy 86.85000000000001
Iteration: 14/50[==============] -Error: 0.1448947866  -Training_Accu
racy:  86.71  -time: 545.00
Validation accuracy 87.75
Iteration: 15/50[==============] -Error: 0.1366147244  -Training_Accu
racy:  87.27  -time: 585.41
Validation accuracy 88.31
Iteration: 16/50[==============] -Error: 0.1295555611  -Training_Accu
racy:  87.81  -time: 624.93
```

```
Validation accuracy 88.79
Iteration: 17/50[==============] -Error: 0.1234759969  -Training_Accu
racy:  88.22  -time: 662.61
Validation accuracy 89.0
Iteration: 18/50[==============] -Error: 0.1182571586  -Training_Accu
racy:  88.57  -time: 700.90
Validation accuracy 89.32
Iteration: 19/50[==============] -Error: 0.1137342192  -Training_Accu
racy:  88.89  -time: 740.31
Validation accuracy 89.53
Iteration: 20/50[==============] -Error: 0.1097803210  -Training_Accu
racy:  89.10  -time: 779.79
Validation accuracy 89.77000000000001
Iteration: 21/50[==============] -Error: 0.1062927505  -Training_Accu
racy:  89.35  -time: 822.27
Validation accuracy 90.03
Iteration: 22/50[==============] -Error: 0.1032154907  -Training_Accu
racy:  89.50  -time: 859.80
Validation accuracy 90.18
Iteration: 23/50[==============] -Error: 0.1004660866  -Training_Accu
racy:  89.68  -time: 897.44
Validation accuracy 90.32
Iteration: 24/50[==============] -Error: 0.0980139300  -Training_Accu
racy:  89.82  -time: 936.16
Validation accuracy 90.46
Iteration: 25/50[==============] -Error: 0.0957712345  -Training_Accu
racy:  90.01  -time: 974.15
Validation accuracy 90.62
Iteration: 26/50[==============] -Error: 0.0937563102  -Training_Accu
racy:  90.14  -time: 1015.16
Validation accuracy 90.75999999999999
Iteration: 27/50[==============] -Error: 0.0919150824  -Training_Accu
racy:  90.24  -time: 1055.69
Validation accuracy 90.8
Iteration: 28/50[==============] -Error: 0.0902086937  -Training_Accu
racy:  90.42  -time: 1093.18
Validation accuracy 90.97
Iteration: 29/50[==============] -Error: 0.0886370856  -Training_Accu
racy:  90.49  -time: 1130.87
Validation accuracy 91.05
Iteration: 30/50[==============] -Error: 0.0871803041  -Training_Accu
racy:  90.59  -time: 1169.19
Validation accuracy 91.14999999999999
Iteration: 31/50[==============] -Error: 0.0858303605  -Training_Accu
racy:  90.74  -time: 1209.97
Validation accuracy 91.32000000000001
Iteration: 32/50[==============] -Error: 0.0845633628  -Training_Accu
racy:  90.87  -time: 1248.60
Validation accuracy 91.33
Iteration: 33/50[==============] -Error: 0.0833753009  -Training_Accu
racy:  90.95  -time: 1286.34
Validation accuracy 91.44
Iteration: 34/50[==============] -Error: 0.0822532879  -Training_Accu
racy:  91.07  -time: 1324.77
Validation accuracy 91.51
Iteration: 35/50[==============] -Error: 0.0811988362  -Training_Accu
racy:  91.12  -time: 1364.44
Validation accuracy 91.60000000000001
Iteration: 36/50[==============] -Error: 0.0802099417  -Training_Accu
racy:  91.22  -time: 1404.52
Validation accuracy 91.66
```

```
Iteration: 37/50[==============] -Error: 0.0792574433  -Training_Accu
racy:  91.27  -time: 1446.35
Validation accuracy 91.84
Iteration: 38/50[==============] -Error: 0.0783652283  -Training_Accu
racy:  91.38  -time: 1483.93
Validation accuracy 91.91
Iteration: 39/50[==============] -Error: 0.0775050401  -Training_Accu
racy:  91.45  -time: 1523.51
Validation accuracy 91.97999999999999
Iteration: 40/50[==============] -Error: 0.0766982378  -Training_Accu
racy:  91.52  -time: 1561.84
Validation accuracy 92.0
Iteration: 41/50[==============] -Error: 0.0759169297  -Training_Accu
racy:  91.57  -time: 1599.51
Validation accuracy 92.10000000000001
Iteration: 42/50[==============] -Error: 0.0751525636  -Training_Accu
racy:  91.68  -time: 1642.36
Validation accuracy 92.11
Iteration: 43/50[==============] -Error: 0.0744562316  -Training_Accu
racy:  91.69  -time: 1681.57
Validation accuracy 92.24
Iteration: 44/50[==============] -Error: 0.0737551024  -Training_Accu
racy:  91.77  -time: 1721.02
Validation accuracy 92.25
Iteration: 45/50[==============] -Error: 0.0730697829  -Training_Accu
racy:  91.80  -time: 1758.57
Validation accuracy 92.38
Iteration: 46/50[==============] -Error: 0.0724636865  -Training_Accu
racy:  91.87  -time: 1796.77
Validation accuracy 92.36
Iteration: 47/50[==============] -Error: 0.0718453459  -Training_Accu
racy:  91.92  -time: 1836.49
Validation accuracy 92.39
Iteration: 48/50[==============] -Error: 0.0712479252  -Training_Accu
racy:  91.97  -time: 1878.40
Validation accuracy 92.44
Iteration: 49/50[==============] -Error: 0.0706691698  -Training_Accu
racy:  92.03  -time: 1916.41
Validation accuracy 92.54
Iteration: 50/50[==============] -Error: 0.0701060324  -Training_Accu
racy:  92.11  -time: 1953.84
Validation accuracy 92.60000000000001
```

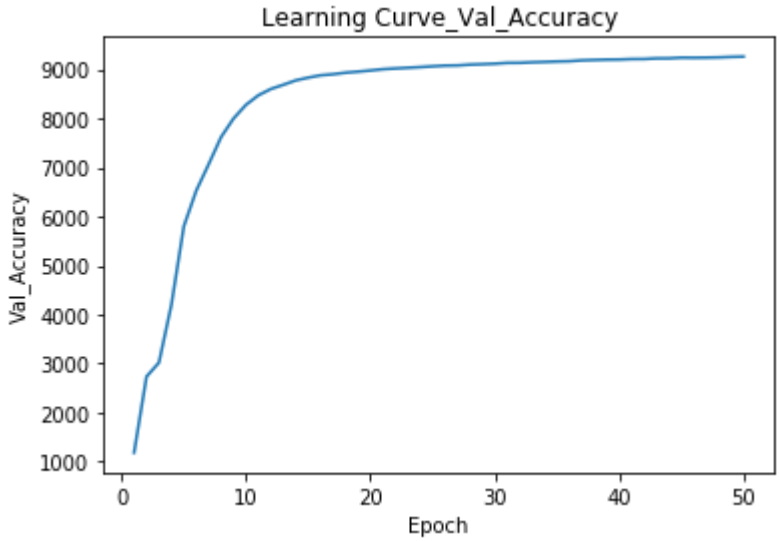





```
Learning rate 0.001000, Test_Accuracy  92.81
```

In [11]:

```
plot_curve(range(1,my_mnist_net_1.iterations+1), val_accuracies_1, "Val_Accuracy")
```



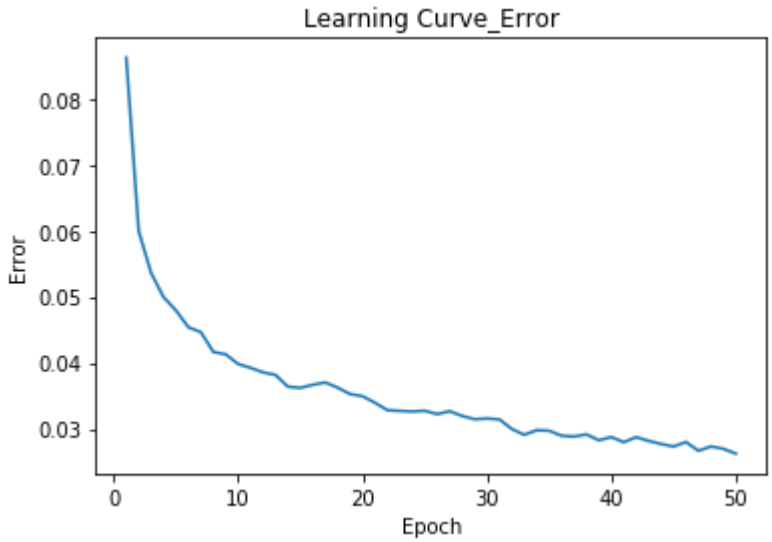
In [12]:

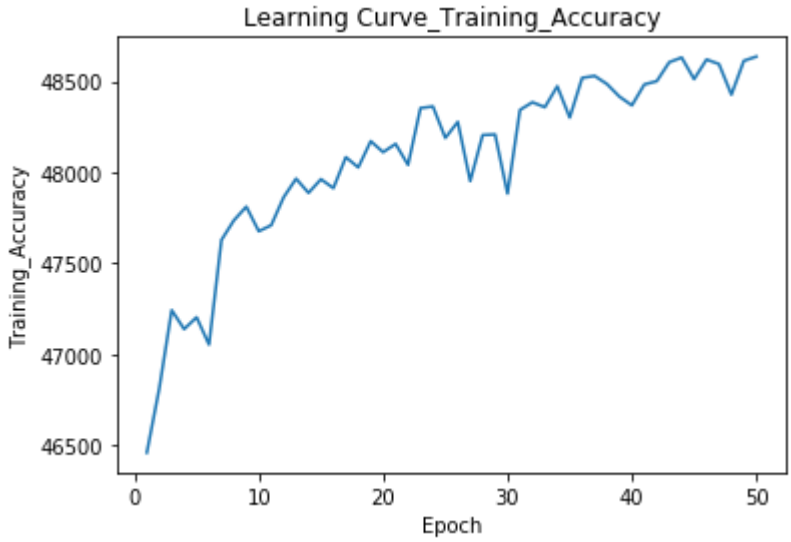
```
#Your implementation with a learning rate of 1.0 goes here

learning_rate = 1.0

my_mnist_net_2 = NeuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_ra
val_accuracies_2 = my_mnist_net_2.train(training_data,validation_data)
test_accuracy.append(my_mnist_net_2.predict(test_data)/100)

print('Learning rate %f, Test_Accuracy  %-2.2f' % (learning_rate, test_accuracy[-1]

my_mnist_net_2.save('saved_models/NN_MNIST_1-00')
```

```
Iteration:  1/50[==============] -Error: 0.0862356325  -Training_Accu
racy:  92.92  -time: 36.89
Validation accuracy 93.03
Iteration:  2/50[==============] -Error: 0.0599729304  -Training_Accu
racy:  93.62  -time: 76.06
Validation accuracy 93.19
Iteration:  3/50[==============] -Error: 0.0536076778  -Training_Accu
racy:  94.48  -time: 119.15
Validation accuracy 93.74
Iteration:  4/50[==============] -Error: 0.0499437011  -Training_Accu
racy:  94.27  -time: 156.78
Validation accuracy 93.95
Iteration:  5/50[==============] -Error: 0.0479279663  -Training_Accu
racy:  94.40  -time: 194.73
Validation accuracy 93.96
Iteration:  6/50[==============] -Error: 0.0453834845  -Training_Accu
racy:  94.10  -time: 232.20
Validation accuracy 93.78999999999999
Iteration:  7/50[==============] -Error: 0.0446461275  -Training_Accu
racy:  95.25  -time: 271.64
Validation accuracy 94.73
Iteration:  8/50[==============] -Error: 0.0416390341  -Training_Accu
racy:  95.47  -time: 311.79
Validation accuracy 94.93
Iteration:  9/50[==============] -Error: 0.0412828293  -Training_Accu
racy:  95.62  -time: 353.11
Validation accuracy 94.82000000000001
Iteration: 10/50[==============] -Error: 0.0398151083  -Training_Accu
racy:  95.35  -time: 390.52
Validation accuracy 94.77
Iteration: 11/50[==============] -Error: 0.0392298400  -Training_Accu
racy:  95.42  -time: 429.96
Validation accuracy 94.1
Iteration: 12/50[==============] -Error: 0.0385487414  -Training_Accu
racy:  95.72  -time: 467.04
Validation accuracy 94.96
Iteration: 13/50[==============] -Error: 0.0381629814  -Training_Accu
racy:  95.93  -time: 504.50
Validation accuracy 95.07
Iteration: 14/50[==============] -Error: 0.0363854761  -Training_Accu
racy:  95.77  -time: 545.59
Validation accuracy 95.11
Iteration: 15/50[==============] -Error: 0.0362018910  -Training_Accu
racy:  95.92  -time: 586.34
Validation accuracy 95.25
Iteration: 16/50[==============] -Error: 0.0366658783  -Training_Accu
```

```
racy:  95.82  -time: 623.65
Validation accuracy 94.75
Iteration: 17/50[==============] -Error: 0.0370168213  -Training_Accu
racy:  96.16  -time: 661.96
Validation accuracy 95.26
Iteration: 18/50[==============] -Error: 0.0362384246  -Training_Accu
racy:  96.05  -time: 699.36
Validation accuracy 94.88
Iteration: 19/50[==============] -Error: 0.0352528047  -Training_Accu
racy:  96.34  -time: 737.39
Validation accuracy 95.07
Iteration: 20/50[==============] -Error: 0.0349380833  -Training_Accu
racy:  96.22  -time: 778.73
Validation accuracy 95.16
Iteration: 21/50[==============] -Error: 0.0339371758  -Training_Accu
racy:  96.31  -time: 816.42
Validation accuracy 95.24000000000001
Iteration: 22/50[==============] -Error: 0.0328153817  -Training_Accu
racy:  96.08  -time: 855.87
Validation accuracy 95.1
Iteration: 23/50[==============] -Error: 0.0327116693  -Training_Accu
racy:  96.70  -time: 895.84
Validation accuracy 95.5
Iteration: 24/50[==============] -Error: 0.0326234890  -Training_Accu
racy:  96.72  -time: 933.20
Validation accuracy 95.67
Iteration: 25/50[==============] -Error: 0.0327332417  -Training_Accu
racy:  96.38  -time: 972.72
Validation accuracy 95.22
Iteration: 26/50[==============] -Error: 0.0322216656  -Training_Accu
racy:  96.55  -time: 1012.25
Validation accuracy 95.33
Iteration: 27/50[==============] -Error: 0.0326687935  -Training_Accu
racy:  95.90  -time: 1050.31
Validation accuracy 94.58
Iteration: 28/50[==============] -Error: 0.0319495006  -Training_Accu
racy:  96.41  -time: 1087.56
Validation accuracy 95.41
Iteration: 29/50[==============] -Error: 0.0314499470  -Training_Accu
racy:  96.41  -time: 1125.32
Validation accuracy 95.13000000000001
Iteration: 30/50[==============] -Error: 0.0315598042  -Training_Accu
racy:  95.76  -time: 1165.82
Validation accuracy 94.58
Iteration: 31/50[==============] -Error: 0.0314258597  -Training_Accu
racy:  96.68  -time: 1206.59
Validation accuracy 95.43
Iteration: 32/50[==============] -Error: 0.0299724799  -Training_Accu
racy:  96.77  -time: 1246.28
Validation accuracy 95.45
Iteration: 33/50[==============] -Error: 0.0290891965  -Training_Accu
racy:  96.71  -time: 1286.26
Validation accuracy 95.22
Iteration: 34/50[==============] -Error: 0.0297894472  -Training_Accu
racy:  96.94  -time: 1323.53
Validation accuracy 95.57
Iteration: 35/50[==============] -Error: 0.0297017160  -Training_Accu
racy:  96.60  -time: 1360.97
Validation accuracy 95.33
Iteration: 36/50[==============] -Error: 0.0289517659  -Training_Accu
racy:  97.04  -time: 1402.85
```

```
Validation accuracy 95.62
Iteration: 37/50[==============] -Error: 0.0288205752  -Training_Accu
racy:  97.06  -time: 1441.13
Validation accuracy 95.66
Iteration: 38/50[==============] -Error: 0.0291505446  -Training_Accu
racy:  96.97  -time: 1480.65
Validation accuracy 95.47
Iteration: 39/50[==============] -Error: 0.0282609811  -Training_Accu
racy:  96.83  -time: 1518.66
Validation accuracy 95.53
Iteration: 40/50[==============] -Error: 0.0287447322  -Training_Accu
racy:  96.73  -time: 1556.19
Validation accuracy 95.55
Iteration: 41/50[==============] -Error: 0.0279722710  -Training_Accu
racy:  96.96  -time: 1594.94
Validation accuracy 95.84
Iteration: 42/50[==============] -Error: 0.0287332328  -Training_Accu
racy:  97.00  -time: 1638.17
Validation accuracy 95.58
Iteration: 43/50[==============] -Error: 0.0281622868  -Training_Accu
racy:  97.21  -time: 1675.53
Validation accuracy 95.64
Iteration: 44/50[==============] -Error: 0.0276980624  -Training_Accu
racy:  97.26  -time: 1712.96
Validation accuracy 95.64
Iteration: 45/50[==============] -Error: 0.0273129487  -Training_Accu
racy:  97.02  -time: 1752.63
Validation accuracy 95.34
Iteration: 46/50[==============] -Error: 0.0279976717  -Training_Accu
racy:  97.24  -time: 1792.08
Validation accuracy 95.65
Iteration: 47/50[==============] -Error: 0.0266650619  -Training_Accu
racy:  97.19  -time: 1831.68
Validation accuracy 95.73
Iteration: 48/50[==============] -Error: 0.0273257724  -Training_Accu
racy:  96.85  -time: 1873.89
Validation accuracy 95.46
Iteration: 49/50[==============] -Error: 0.0269769196  -Training_Accu
racy:  97.22  -time: 1911.35
Validation accuracy 95.56
Iteration: 50/50[==============] -Error: 0.0262357139  -Training_Accu
racy:  97.27  -time: 1949.11
Validation accuracy 95.65
```

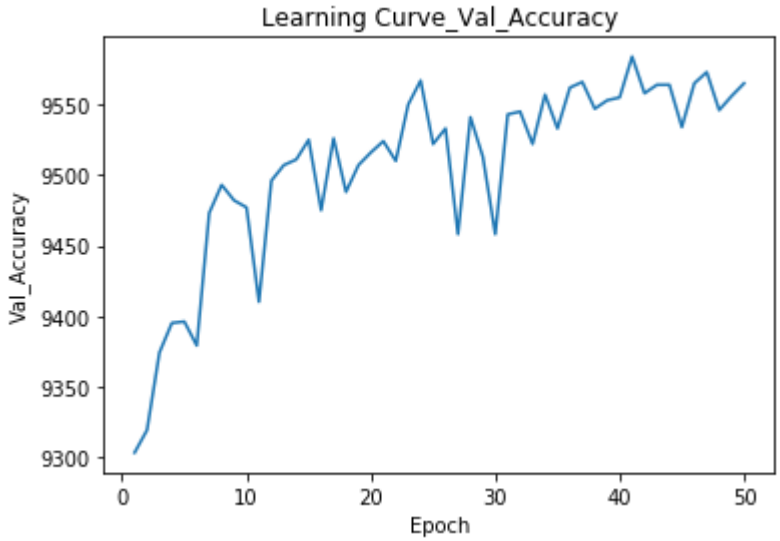





```
Learning rate 1.000000, Test_Accuracy  95.70
```

In [13]:

```
plot_curve(range(1,my_mnist_net_2.iterations+1), val_accuracies_2, "Val_Accuracy")
```



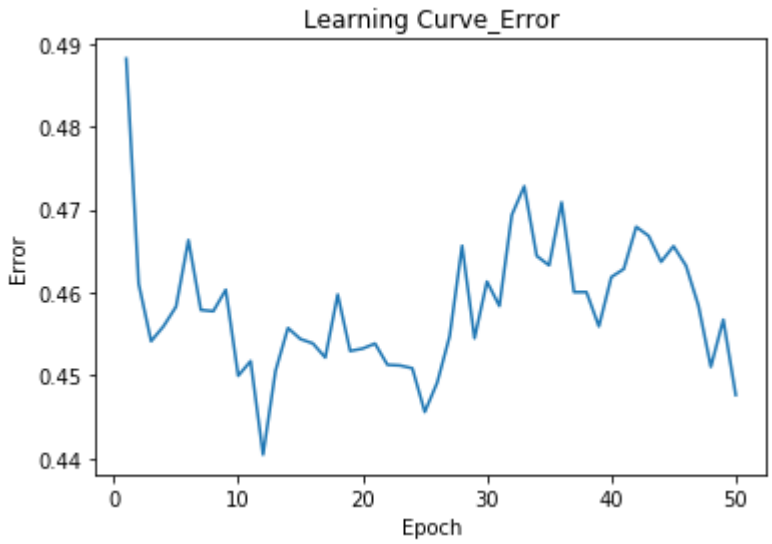
In [14]:

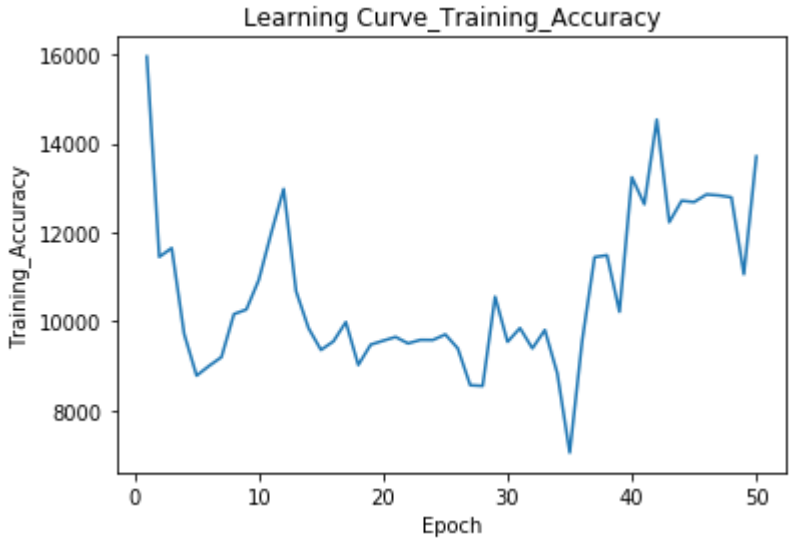
```
#Your implementation with a learning rate of 10 goes here
learning_rate = 10

my_mnist_net_3 = NeuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_ra
val_accuracies_3 = my_mnist_net_3.train(training_data,validation_data)
test_accuracy.append(my_mnist_net_3.predict(test_data)/100)
print('Learning rate %f, Test_Accuracy  %-2.2f' % (learning_rate, test_accuracy[-1]
my_mnist_net_3.save('saved_models/NN_MNIST_10-0')
```

```
Iteration:  1/50[==============] -Error: 0.4882105748  -Training_Accu
racy:  31.90  -time: 38.77
Validation accuracy 32.550000000000004
Iteration:  2/50[==============] -Error: 0.4610481917  -Training_Accu
racy:  22.89  -time: 78.49
Validation accuracy 23.22
Iteration:  3/50[==============] -Error: 0.4541295311  -Training_Accu
racy:  23.29  -time: 119.98
Validation accuracy 23.44
Iteration:  4/50[==============] -Error: 0.4559398002  -Training_Accu
racy:  19.43  -time: 159.80
Validation accuracy 19.54
Iteration:  5/50[==============] -Error: 0.4582842583  -Training_Accu
racy:  17.55  -time: 197.89
Validation accuracy 17.76
Iteration:  6/50[==============] -Error: 0.4663215677  -Training_Accu
racy:  17.99  -time: 238.03
Validation accuracy 17.59
Iteration:  7/50[==============] -Error: 0.4578936873  -Training_Accu
racy:  18.38  -time: 277.43
Validation accuracy 18.23
Iteration:  8/50[==============] -Error: 0.4577466750  -Training_Accu
racy:  20.32  -time: 321.00
Validation accuracy 19.89
Iteration:  9/50[==============] -Error: 0.4603335406  -Training_Accu
racy:  20.52  -time: 359.52
Validation accuracy 20.369999999999997
Iteration: 10/50[==============] -Error: 0.4499614225  -Training_Accu
racy:  21.85  -time: 400.35
Validation accuracy 22.39
Iteration: 11/50[==============] -Error: 0.4517253801  -Training_Accu
racy:  23.96  -time: 443.10
Validation accuracy 23.94
Iteration: 12/50[==============] -Error: 0.4404472550  -Training_Accu
racy:  25.95  -time: 498.72
Validation accuracy 26.19
Iteration: 13/50[==============] -Error: 0.4505511532  -Training_Accu
racy:  21.35  -time: 547.80
Validation accuracy 21.05
Iteration: 14/50[==============] -Error: 0.4557225503  -Training_Accu
racy:  19.69  -time: 608.02
Validation accuracy 19.89
Iteration: 15/50[==============] -Error: 0.4543909011  -Training_Accu
racy:  18.71  -time: 659.26
Validation accuracy 18.91
Iteration: 16/50[==============] -Error: 0.4538647860  -Training_Accu
racy:  19.09  -time: 710.73
Validation accuracy 19.27
Iteration: 17/50[==============] -Error: 0.4521645140  -Training_Accu
racy:  19.96  -time: 754.55
```

```
Validation accuracy 19.86
Iteration: 18/50[==============] -Error: 0.4597598192  -Training_Accu
racy:  18.03  -time: 796.56
Validation accuracy 17.82
Iteration: 19/50[==============] -Error: 0.4529481143  -Training_Accu
racy:  18.95  -time: 839.24
Validation accuracy 18.94
Iteration: 20/50[==============] -Error: 0.4532349656  -Training_Accu
racy:  19.12  -time: 880.68
Validation accuracy 19.189999999999998
Iteration: 21/50[==============] -Error: 0.4538417939  -Training_Accu
racy:  19.29  -time: 924.01
Validation accuracy 19.220000000000002
Iteration: 22/50[==============] -Error: 0.4512818786  -Training_Accu
racy:  19.00  -time: 967.99
Validation accuracy 19.0
Iteration: 23/50[==============] -Error: 0.4512037792  -Training_Accu
racy:  19.16  -time: 1009.46
Validation accuracy 19.18
Iteration: 24/50[==============] -Error: 0.4508660079  -Training_Accu
racy:  19.15  -time: 1050.88
Validation accuracy 19.18
Iteration: 25/50[==============] -Error: 0.4455977192  -Training_Accu
racy:  19.40  -time: 1089.65
Validation accuracy 19.46
Iteration: 26/50[==============] -Error: 0.4491395946  -Training_Accu
racy:  18.78  -time: 1131.37
Validation accuracy 18.86
Iteration: 27/50[==============] -Error: 0.4547287523  -Training_Accu
racy:  17.12  -time: 1169.89
Validation accuracy 17.19
Iteration: 28/50[==============] -Error: 0.4656372429  -Training_Accu
racy:  17.09  -time: 1208.63
Validation accuracy 17.16
Iteration: 29/50[==============] -Error: 0.4544827607  -Training_Accu
racy:  21.10  -time: 1246.19
Validation accuracy 20.94
Iteration: 30/50[==============] -Error: 0.4613144187  -Training_Accu
racy:  19.07  -time: 1286.96
Validation accuracy 18.26
Iteration: 31/50[==============] -Error: 0.4583782157  -Training_Accu
racy:  19.69  -time: 1325.86
Validation accuracy 19.650000000000002
Iteration: 32/50[==============] -Error: 0.4693517877  -Training_Accu
racy:  18.78  -time: 1363.61
Validation accuracy 18.34
Iteration: 33/50[==============] -Error: 0.4728123888  -Training_Accu
racy:  19.60  -time: 1404.55
Validation accuracy 19.29
Iteration: 34/50[==============] -Error: 0.4644112943  -Training_Accu
racy:  17.66  -time: 1443.44
Validation accuracy 17.66
Iteration: 35/50[==============] -Error: 0.4632774224  -Training_Accu
racy:  14.09  -time: 1480.85
Validation accuracy 14.34
Iteration: 36/50[==============] -Error: 0.4708774480  -Training_Accu
racy:  19.12  -time: 1520.38
Validation accuracy 19.1
Iteration: 37/50[==============] -Error: 0.4600473788  -Training_Accu
racy:  22.88  -time: 1560.71
Validation accuracy 22.18
```

```
Iteration: 38/50[==============] -Error: 0.4600538618  -Training_Accu
racy:  22.97  -time: 1599.79
Validation accuracy 22.97
Iteration: 39/50[==============] -Error: 0.4559174448  -Training_Accu
racy:  20.42  -time: 1638.61
Validation accuracy 20.26
Iteration: 40/50[==============] -Error: 0.4618667596  -Training_Accu
racy:  26.47  -time: 1678.01
Validation accuracy 26.729999999999997
Iteration: 41/50[==============] -Error: 0.4628373178  -Training_Accu
racy:  25.26  -time: 1717.23
Validation accuracy 25.94
Iteration: 42/50[==============] -Error: 0.4679239671  -Training_Accu
racy:  29.06  -time: 1754.13
Validation accuracy 29.65
Iteration: 43/50[==============] -Error: 0.4668289274  -Training_Accu
racy:  24.45  -time: 1794.86
Validation accuracy 24.8
Iteration: 44/50[==============] -Error: 0.4636969855  -Training_Accu
racy:  25.42  -time: 1834.62
Validation accuracy 24.81
Iteration: 45/50[==============] -Error: 0.4655915793  -Training_Accu
racy:  25.36  -time: 1873.71
Validation accuracy 24.89
Iteration: 46/50[==============] -Error: 0.4632374452  -Training_Accu
racy:  25.70  -time: 1912.86
Validation accuracy 25.230000000000004
Iteration: 47/50[==============] -Error: 0.4584310458  -Training_Accu
racy:  25.66  -time: 1956.60
Validation accuracy 25.15
Iteration: 48/50[==============] -Error: 0.4510370594  -Training_Accu
racy:  25.57  -time: 1995.38
Validation accuracy 25.080000000000002
Iteration: 49/50[==============] -Error: 0.4567230888  -Training_Accu
racy:  22.11  -time: 2034.86
Validation accuracy 21.66
Iteration: 50/50[==============] -Error: 0.4476436669  -Training_Accu
racy:  27.41  -time: 2074.45
Validation accuracy 26.740000000000002
```

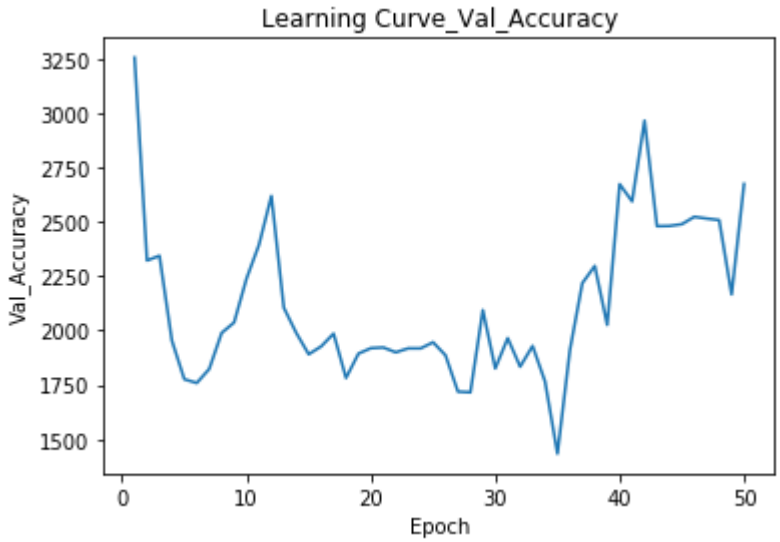


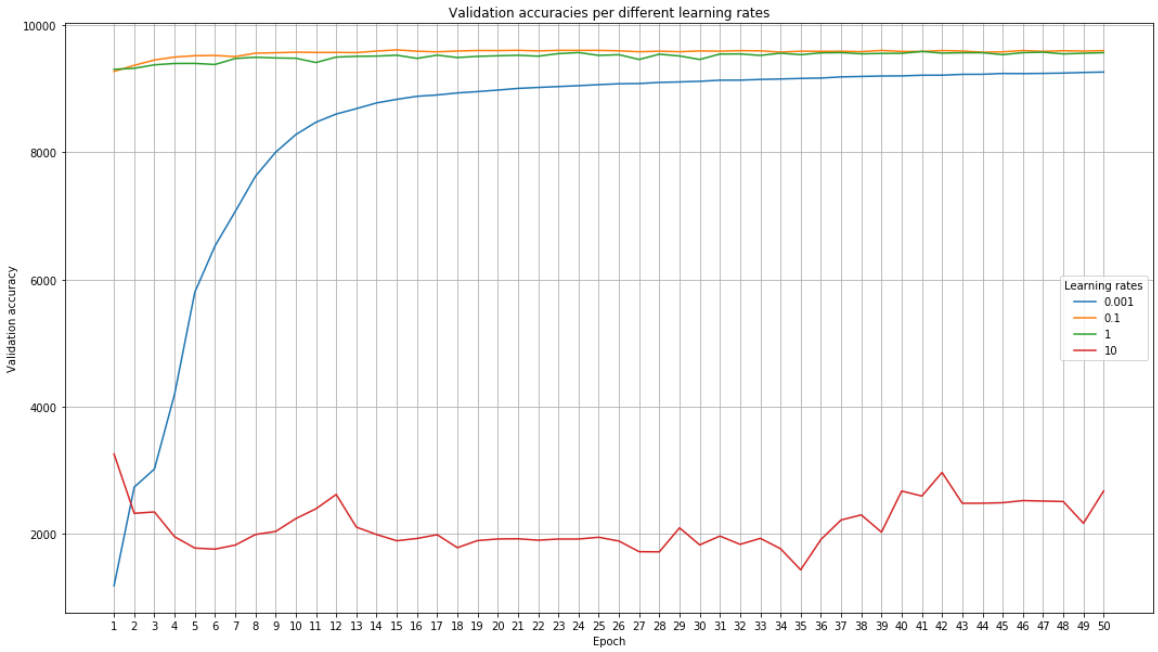



```
Learning rate 10.000000, Test_Accuracy  27.43
```

In [32]:

```
plot_curve(range(1,my_mnist_net_3.iterations+1), val_accuracies_3, "Val_Accuracy")


#plotting different validation accuracies in the same plot
x = np.arange(1,51)
plt.figure(figsize=(18,10))
plt.xlabel('Epoch')
plt.ylabel('Validation accuracy')
plt.title('Validation accuracies per different learning rates')
plt.xticks(x)
plt.grid()
plt.plot(x,val_accuracies_1, label=0.001)
plt.plot(x,val_accuracies, label=0.1)
plt.plot(x,val_accuracies_2, label=1)
plt.plot(x,val_accuracies_3, label=10)
plt.legend(title="Learning rates")
plt.show()

rates = [0.1, 0.001, 1, 10]

best_rate = rates[np.argmax(test_accuracy)]
print("Best learning rate = %f" % (best_rate))
```





```
Best learning rate = 0.100000
```

Comment

Increasing the learning rate increases the step size at which our weights get updated. From the plot above we can see very well that the learning rates 0.1 and 1 produce the best validation accuracies. However, the 0.1 learning rate converges the fastest and also produces the best test accuracy. Therefore, we consider it the best parameter value. A learning rate of 10 is definitely too high. The validation accuracy does not improve, instead it keeps fluctuating. This is because the weight updates do not properly follow the gradient direction towards the ground of small "valleys" on the error surface. We either "jump" across them or we keep on bouncing back and forth inside the valley.

Question 2.2.2 : initialize all weights to 0. Plot the training accuracy curve. Comment your results

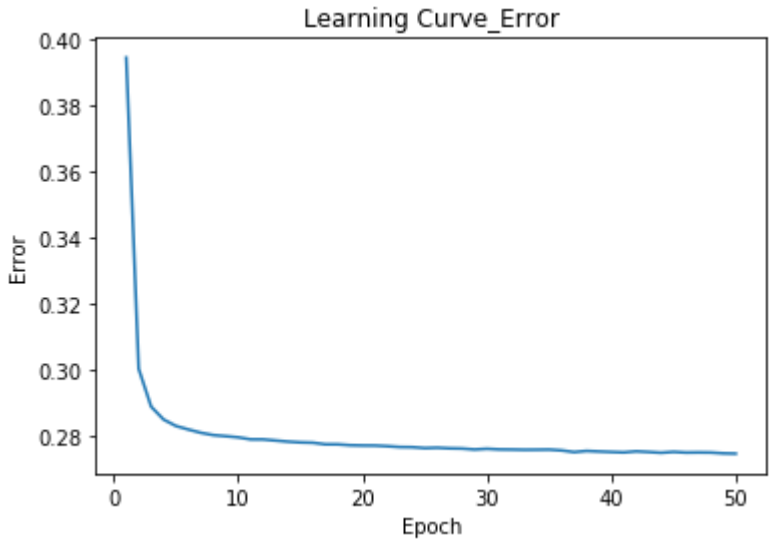
In [25]:

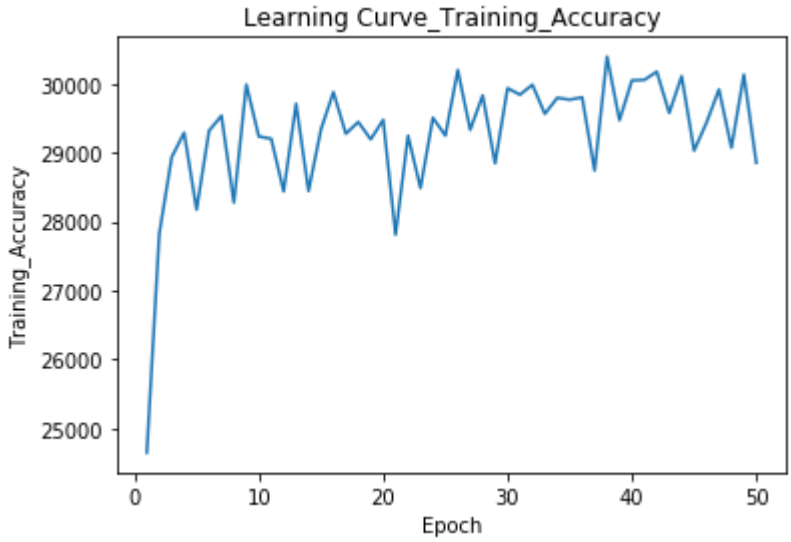
```
#Your implementation goes here
my_mnist_net_zeros = NeuralNetwork(input_nodes, hidden_nodes, output_nodes)

input_hidden_weights = np.zeros((my_mnist_net_zeros.input, my_mnist_net_zeros.hidde
hidden_output_weights = np.zeros((my_mnist_net_zeros.hidden, my_mnist_net_zeros.out

my_mnist_net_zeros.weights_initialisation(input_hidden_weights, hidden_output_weigh

validation_zeros = my_mnist_net_zeros.train(training_data,validation_data)
test_accuracy_zeros = my_mnist_net_zeros.predict(test_data)/100
print('Test_Accuracy [zeros]  %-2.2f' % (test_accuracy_zeros))
my_mnist_net_zeros.save('saved_models/NN_MNIST_DEFAULT_0s')
```

```
Iteration:  1/50[==============] -Error: 0.3944418670  -Training_Accu
racy:  49.30  -time: 42.47
Validation accuracy 50.019999999999996
Iteration:  2/50[==============] -Error: 0.3002768536  -Training_Accu
racy:  55.65  -time: 86.71
Validation accuracy 55.769999999999996
Iteration:  3/50[==============] -Error: 0.2888264688  -Training_Accu
racy:  57.87  -time: 127.11
Validation accuracy 58.040000000000006
Iteration:  4/50[==============] -Error: 0.2849249187  -Training_Accu
racy:  58.58  -time: 169.63
Validation accuracy 57.9
Iteration:  5/50[==============] -Error: 0.2829657162  -Training_Accu
racy:  56.35  -time: 207.85
Validation accuracy 56.230000000000004
Iteration:  6/50[==============] -Error: 0.2818992711  -Training_Accu
racy:  58.64  -time: 244.12
Validation accuracy 58.45
Iteration:  7/50[==============] -Error: 0.2809120987  -Training_Accu
racy:  59.07  -time: 281.01
Validation accuracy 59.07
Iteration:  8/50[==============] -Error: 0.2802082076  -Training_Accu
racy:  56.55  -time: 317.03
Validation accuracy 56.81
Iteration:  9/50[==============] -Error: 0.2798840755  -Training_Accu
racy:  59.98  -time: 352.85
Validation accuracy 59.809999999999995
Iteration: 10/50[==============] -Error: 0.2795321069  -Training_Accu
racy:  58.47  -time: 389.46
Validation accuracy 58.08
Iteration: 11/50[==============] -Error: 0.2789165000  -Training_Accu
racy:  58.41  -time: 426.38
Validation accuracy 57.9
Iteration: 12/50[==============] -Error: 0.2788981985  -Training_Accu
racy:  56.88  -time: 462.78
Validation accuracy 56.81
Iteration: 13/50[==============] -Error: 0.2785632172  -Training_Accu
racy:  59.42  -time: 499.11
Validation accuracy 59.29
Iteration: 14/50[==============] -Error: 0.2782286098  -Training_Accu
racy:  56.89  -time: 536.29
Validation accuracy 56.64
Iteration: 15/50[==============] -Error: 0.2780018660  -Training_Accu
racy:  58.68  -time: 572.56
Validation accuracy 58.35
Iteration: 16/50[==============] -Error: 0.2779022939  -Training_Accu
```

```
racy:  59.76  -time: 608.91
Validation accuracy 59.63
Iteration: 17/50[==============] -Error: 0.2774224113  -Training_Accu
racy:  58.55  -time: 646.19
Validation accuracy 58.120000000000005
Iteration: 18/50[==============] -Error: 0.2774098505  -Training_Accu
racy:  58.89  -time: 682.15
Validation accuracy 58.8
Iteration: 19/50[==============] -Error: 0.2771471771  -Training_Accu
racy:  58.39  -time: 718.44
Validation accuracy 58.120000000000005
Iteration: 20/50[==============] -Error: 0.2770659903  -Training_Accu
racy:  58.95  -time: 755.34
Validation accuracy 58.57
Iteration: 21/50[==============] -Error: 0.2770408051  -Training_Accu
racy:  55.62  -time: 791.49
Validation accuracy 54.86
Iteration: 22/50[==============] -Error: 0.2768545449  -Training_Accu
racy:  58.49  -time: 827.56
Validation accuracy 57.809999999999995
Iteration: 23/50[==============] -Error: 0.2765981811  -Training_Accu
racy:  56.98  -time: 864.21
Validation accuracy 56.36
Iteration: 24/50[==============] -Error: 0.2765520517  -Training_Accu
racy:  59.01  -time: 901.19
Validation accuracy 58.51
Iteration: 25/50[==============] -Error: 0.2762777797  -Training_Accu
racy:  58.50  -time: 937.29
Validation accuracy 58.01
Iteration: 26/50[==============] -Error: 0.2763817840  -Training_Accu
racy:  60.40  -time: 973.93
Validation accuracy 59.709999999999994
Iteration: 27/50[==============] -Error: 0.2762021474  -Training_Accu
racy:  58.67  -time: 1010.72
Validation accuracy 58.29
Iteration: 28/50[==============] -Error: 0.2761388793  -Training_Accu
racy:  59.66  -time: 1046.94
Validation accuracy 59.08
Iteration: 29/50[==============] -Error: 0.2758522697  -Training_Accu
racy:  57.69  -time: 1082.93
Validation accuracy 57.31
Iteration: 30/50[==============] -Error: 0.2760678748  -Training_Accu
racy:  59.87  -time: 1119.96
Validation accuracy 59.599999999999994
Iteration: 31/50[==============] -Error: 0.2758703251  -Training_Accu
racy:  59.68  -time: 1156.05
Validation accuracy 59.13
Iteration: 32/50[==============] -Error: 0.2758316718  -Training_Accu
racy:  59.97  -time: 1192.10
Validation accuracy 59.98
Iteration: 33/50[==============] -Error: 0.2757659814  -Training_Accu
racy:  59.13  -time: 1228.32
Validation accuracy 58.940000000000005
Iteration: 34/50[==============] -Error: 0.2757852272  -Training_Accu
racy:  59.59  -time: 1265.20
Validation accuracy 59.099999999999994
Iteration: 35/50[==============] -Error: 0.2758197468  -Training_Accu
racy:  59.54  -time: 1301.28
Validation accuracy 59.48
Iteration: 36/50[==============] -Error: 0.2755742328  -Training_Accu
racy:  59.61  -time: 1337.38
```

```
Validation accuracy 59.07
Iteration: 37/50[==============] -Error: 0.2750658399  -Training_Accu
racy:  57.48  -time: 1374.18
Validation accuracy 57.46
Iteration: 38/50[==============] -Error: 0.2754128547  -Training_Accu
racy:  60.79  -time: 1409.87
Validation accuracy 60.18
Iteration: 39/50[==============] -Error: 0.2752446102  -Training_Accu
racy:  58.94  -time: 1446.03
Validation accuracy 58.89
Iteration: 40/50[==============] -Error: 0.2750998157  -Training_Accu
racy:  60.10  -time: 1482.55
Validation accuracy 59.699999999999996
Iteration: 41/50[==============] -Error: 0.2749747333  -Training_Accu
racy:  60.11  -time: 1518.91
Validation accuracy 59.95
Iteration: 42/50[==============] -Error: 0.2752910871  -Training_Accu
racy:  60.35  -time: 1554.82
Validation accuracy 59.519999999999996
Iteration: 43/50[==============] -Error: 0.2750971396  -Training_Accu
racy:  59.16  -time: 1591.71
Validation accuracy 58.57
Iteration: 44/50[==============] -Error: 0.2748683508  -Training_Accu
racy:  60.22  -time: 1627.91
Validation accuracy 59.79
Iteration: 45/50[==============] -Error: 0.2751317140  -Training_Accu
racy:  58.06  -time: 1663.98
Validation accuracy 58.13
Iteration: 46/50[==============] -Error: 0.2749221124  -Training_Accu
racy:  58.88  -time: 1700.05
Validation accuracy 58.660000000000004
Iteration: 47/50[==============] -Error: 0.2749639640  -Training_Accu
racy:  59.84  -time: 1737.24
Validation accuracy 59.519999999999996
Iteration: 48/50[==============] -Error: 0.2749265388  -Training_Accu
racy:  58.15  -time: 1773.49
Validation accuracy 57.830000000000005
Iteration: 49/50[==============] -Error: 0.2746863887  -Training_Accu
racy:  60.27  -time: 1809.90
Validation accuracy 59.86
Iteration: 50/50[==============] -Error: 0.2745970478  -Training_Accu
racy:  57.71  -time: 1846.74
Validation accuracy 57.63
```






```
Test_Accuracy [zeros]  57.12
```

The test accuracy is very low (57.12%). This is because symmetric weight upates prevent the network from learning. Weights need to be randomized to break this symmetrie.

Question 2.2.3 : Try with a different transfer function (such as tanh). File transfer_functions.py provides you the python implementation of the tanh function and its derivative

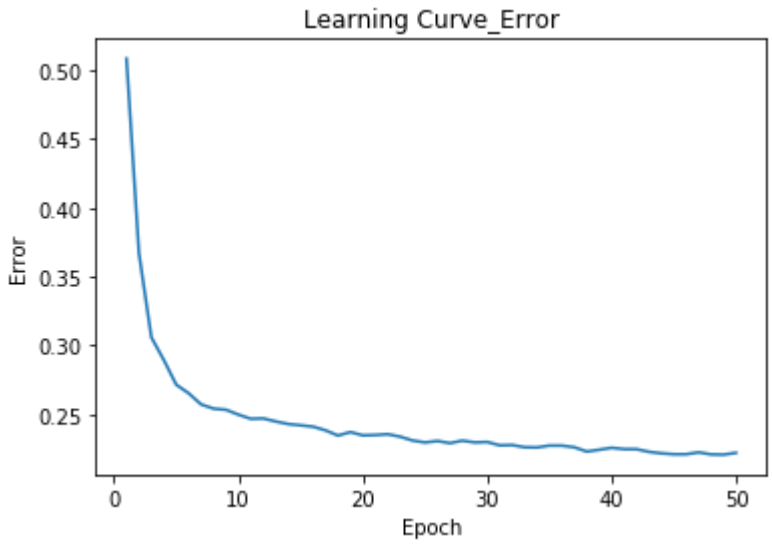
In [26]:

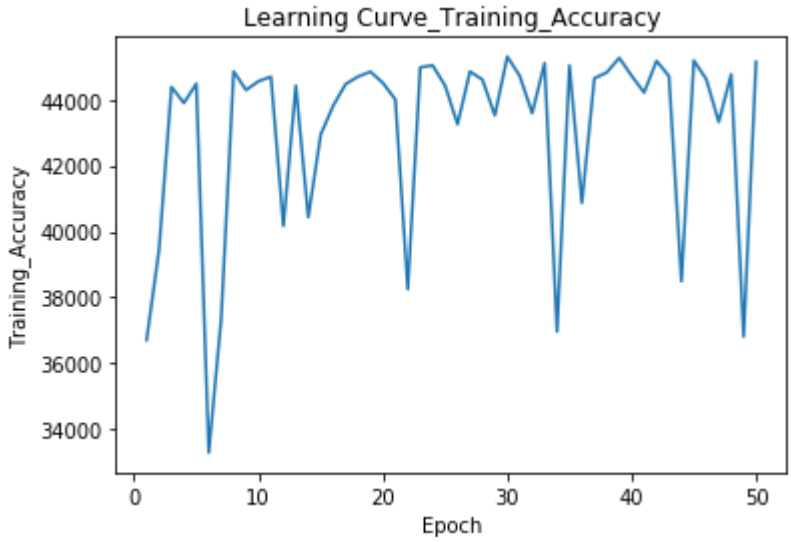
```
#Your implementation goes here

my_mnist_net_tanh = NeuralNetwork(input_nodes, hidden_nodes, output_nodes, transfer
#Your implementation goes here
validaion_tanh = my_mnist_net_tanh.train(training_data,validation_data)
test_accuracy_tanh = my_mnist_net_tanh.predict(test_data)/100
print('Test_Accuracy [tanh]  %-2.2f' % (test_accuracy_tanh))
my_mnist_net_zeros.save('saved_models/NN_MNIST_DEFAULT_tanh')
```

```
Iteration:  1/50[==============] -Error: 0.5085838336  -Training_Accu
racy:  73.41  -time: 31.80
Validation accuracy 72.66
Iteration:  2/50[==============] -Error: 0.3673537104  -Training_Accu
racy:  78.88  -time: 64.40
Validation accuracy 78.7
Iteration:  3/50[==============] -Error: 0.3059682351  -Training_Accu
racy:  88.76  -time: 97.30
Validation accuracy 89.38000000000001
Iteration:  4/50[==============] -Error: 0.2895039859  -Training_Accu
racy:  87.79  -time: 130.99
Validation accuracy 87.59
Iteration:  5/50[==============] -Error: 0.2715567574  -Training_Accu
racy:  88.96  -time: 163.68
Validation accuracy 89.21
Iteration:  6/50[==============] -Error: 0.2652897194  -Training_Accu
racy:  66.59  -time: 196.75
Validation accuracy 66.8
Iteration:  7/50[==============] -Error: 0.2572294795  -Training_Accu
racy:  74.73  -time: 230.36
Validation accuracy 75.26
Iteration:  8/50[==============] -Error: 0.2541554068  -Training_Accu
racy:  89.70  -time: 263.16
Validation accuracy 89.96
Iteration:  9/50[==============] -Error: 0.2533331663  -Training_Accu
racy:  88.58  -time: 295.91
Validation accuracy 88.32
Iteration: 10/50[==============] -Error: 0.2497735892  -Training_Accu
racy:  89.11  -time: 328.78
Validation accuracy 89.19
Iteration: 11/50[==============] -Error: 0.2467558199  -Training_Accu
racy:  89.38  -time: 362.34
Validation accuracy 89.41
Iteration: 12/50[==============] -Error: 0.2470200957  -Training_Accu
racy:  80.34  -time: 395.16
Validation accuracy 80.07
Iteration: 13/50[==============] -Error: 0.2448189539  -Training_Accu
racy:  88.84  -time: 427.99
Validation accuracy 88.61
Iteration: 14/50[==============] -Error: 0.2429629734  -Training_Accu
racy:  80.87  -time: 461.25
Validation accuracy 81.24
Iteration: 15/50[==============] -Error: 0.2420487842  -Training_Accu
racy:  85.88  -time: 494.93
Validation accuracy 85.33
Iteration: 16/50[==============] -Error: 0.2410381778  -Training_Accu
racy:  87.62  -time: 527.80
Validation accuracy 87.36
Iteration: 17/50[==============] -Error: 0.2383785755  -Training_Accu
racy:  88.93  -time: 560.72
```

```
Validation accuracy 88.48
Iteration: 18/50[==============] -Error: 0.2345942730  -Training_Accu
racy:  89.39  -time: 594.20
Validation accuracy 89.03
Iteration: 19/50[==============] -Error: 0.2371064791  -Training_Accu
racy:  89.69  -time: 627.09
Validation accuracy 89.12
Iteration: 20/50[==============] -Error: 0.2347393385  -Training_Accu
racy:  89.01  -time: 659.96
Validation accuracy 88.59
Iteration: 21/50[==============] -Error: 0.2349769277  -Training_Accu
racy:  88.01  -time: 693.01
Validation accuracy 87.26
Iteration: 22/50[==============] -Error: 0.2355610508  -Training_Accu
racy:  76.50  -time: 726.71
Validation accuracy 75.8
Iteration: 23/50[==============] -Error: 0.2337554799  -Training_Accu
racy:  89.95  -time: 759.57
Validation accuracy 89.4
Iteration: 24/50[==============] -Error: 0.2309771655  -Training_Accu
racy:  90.07  -time: 792.29
Validation accuracy 89.42999999999999
Iteration: 25/50[==============] -Error: 0.2295799330  -Training_Accu
racy:  88.84  -time: 825.87
Validation accuracy 88.6
Iteration: 26/50[==============] -Error: 0.2306281751  -Training_Accu
racy:  86.50  -time: 858.58
Validation accuracy 85.67
Iteration: 27/50[==============] -Error: 0.2290878847  -Training_Accu
racy:  89.70  -time: 891.31
Validation accuracy 89.05999999999999
Iteration: 28/50[==============] -Error: 0.2308811063  -Training_Accu
racy:  89.21  -time: 924.28
Validation accuracy 88.9
Iteration: 29/50[==============] -Error: 0.2296004688  -Training_Accu
racy:  87.04  -time: 958.01
Validation accuracy 87.1
Iteration: 30/50[==============] -Error: 0.2299055666  -Training_Accu
racy:  90.60  -time: 990.94
Validation accuracy 89.62
Iteration: 31/50[==============] -Error: 0.2274767065  -Training_Accu
racy:  89.44  -time: 1023.82
Validation accuracy 88.85
Iteration: 32/50[==============] -Error: 0.2277029374  -Training_Accu
racy:  87.18  -time: 1056.75
Validation accuracy 86.55000000000001
Iteration: 33/50[==============] -Error: 0.2261079129  -Training_Accu
racy:  90.21  -time: 1090.32
Validation accuracy 89.3
Iteration: 34/50[==============] -Error: 0.2259335923  -Training_Accu
racy:  73.93  -time: 1123.01
Validation accuracy 73.31
Iteration: 35/50[==============] -Error: 0.2273045705  -Training_Accu
racy:  90.07  -time: 1156.03
Validation accuracy 89.32
Iteration: 36/50[==============] -Error: 0.2272675179  -Training_Accu
racy:  81.73  -time: 1189.49
Validation accuracy 81.17
Iteration: 37/50[==============] -Error: 0.2260849116  -Training_Accu
racy:  89.29  -time: 1222.29
Validation accuracy 88.44
```

```
Iteration: 38/50[==============] -Error: 0.2229750733  -Training_Accu
racy:  89.64  -time: 1255.05
Validation accuracy 88.9
Iteration: 39/50[==============] -Error: 0.2242682466  -Training_Accu
racy:  90.53  -time: 1288.02
Validation accuracy 89.7
Iteration: 40/50[==============] -Error: 0.2256879697  -Training_Accu
racy:  89.44  -time: 1321.62
Validation accuracy 88.53999999999999
Iteration: 41/50[==============] -Error: 0.2247478596  -Training_Accu
racy:  88.42  -time: 1354.61
Validation accuracy 87.48
Iteration: 42/50[==============] -Error: 0.2247002893  -Training_Accu
racy:  90.34  -time: 1387.41
Validation accuracy 89.05999999999999
Iteration: 43/50[==============] -Error: 0.2227572239  -Training_Accu
racy:  89.42  -time: 1420.48
Validation accuracy 88.58
Iteration: 44/50[==============] -Error: 0.2216742715  -Training_Accu
racy:  76.98  -time: 1454.01
Validation accuracy 76.11
Iteration: 45/50[==============] -Error: 0.2210043456  -Training_Accu
racy:  90.36  -time: 1486.91
Validation accuracy 89.49000000000001
Iteration: 46/50[==============] -Error: 0.2210046642  -Training_Accu
racy:  89.23  -time: 1520.05
Validation accuracy 88.08
Iteration: 47/50[==============] -Error: 0.2223766359  -Training_Accu
racy:  86.64  -time: 1553.69
Validation accuracy 85.25
Iteration: 48/50[==============] -Error: 0.2208815162  -Training_Accu
racy:  89.53  -time: 1586.68
Validation accuracy 89.12
Iteration: 49/50[==============] -Error: 0.2207115863  -Training_Accu
racy:  73.63  -time: 1619.59
Validation accuracy 73.07000000000001
Iteration: 50/50[==============] -Error: 0.2220251862  -Training_Accu
racy:  90.28  -time: 1652.60
Validation accuracy 89.4
```






```
Test_Accuracy [tanh]  89.57
```

The range of values of *tanh* is (-1, 1) but our labels are only 0 and 1. Therefore, the labels only cover a part a part of the image of the tanh function. To be precise, training our network with 0,1 labels constraints the activation at the output layer to be positive in order to produce an output between 0 and 1. This also implies a constraint on the respective weights that now cannot be set as freely as with a sigmoid activation. A smaller search space on the optimal weight values results in a worse performance on the test accuracy (we lose about 5%).

Question 2.2.4 : Add more neurons in the hidden layer (try with 100, 200, 300). Plot the curve representing the validation accuracy versus the number of neurons in the hidden layer. (Choose and justify other hyper-parameters)

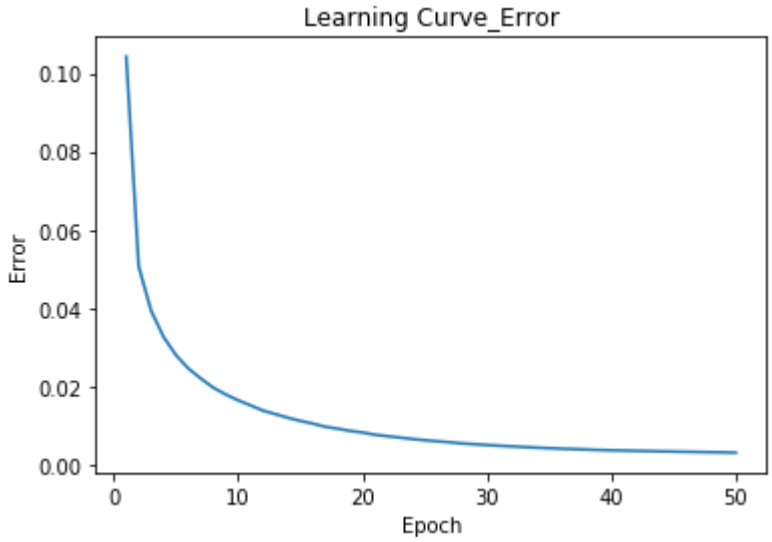
In [27]:

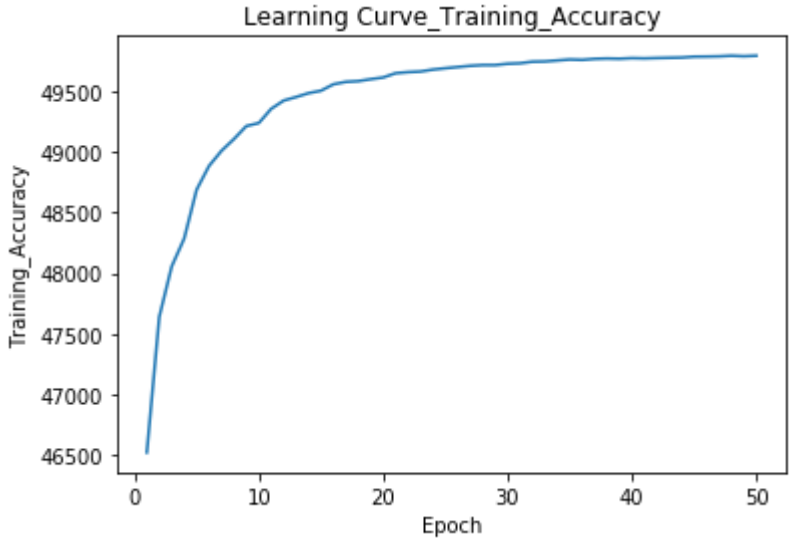
```
#Your implementation goes here

learning_rate = best_rate
#using the best rate found before
my_mnist_net_100_hidden = NeuralNetwork(input_nodes, 100, output_nodes, learning_ra
val_accuracies_100 = my_mnist_net_100_hidden.train(training_data,validation_data)
print(val_accuracies_100[-1])

my_mnist_net_200_hidden = NeuralNetwork(input_nodes, 200, output_nodes, learning_ra
val_accuracies_200 = my_mnist_net_200_hidden.train(training_data,validation_data)
print(val_accuracies_200[-1])

my_mnist_net_300_hidden = NeuralNetwork(input_nodes, 300, output_nodes, learning_ra
val_accuracies_300 = my_mnist_net_300_hidden.train(training_data,validation_data)
print(val_accuracies_300[-1])
```

```
Iteration:  1/50[==============] -Error: 0.1042457508  -Training_Accu
racy:  93.05  -time: 58.60
Validation accuracy 93.13
Iteration:  2/50[==============] -Error: 0.0506479369  -Training_Accu
racy:  95.28  -time: 118.03
Validation accuracy 95.12
Iteration:  3/50[==============] -Error: 0.0393141904  -Training_Accu
racy:  96.11  -time: 177.83
Validation accuracy 95.71
Iteration:  4/50[==============] -Error: 0.0326241688  -Training_Accu
racy:  96.57  -time: 236.81
Validation accuracy 95.69
Iteration:  5/50[==============] -Error: 0.0279573126  -Training_Accu
racy:  97.37  -time: 296.95
Validation accuracy 96.6
Iteration:  6/50[==============] -Error: 0.0245479952  -Training_Accu
racy:  97.77  -time: 356.59
Validation accuracy 96.87
Iteration:  7/50[==============] -Error: 0.0219899221  -Training_Accu
racy:  98.01  -time: 418.70
Validation accuracy 97.00999999999999
Iteration:  8/50[==============] -Error: 0.0196508788  -Training_Accu
racy:  98.21  -time: 480.62
Validation accuracy 97.1
Iteration:  9/50[==============] -Error: 0.0179200252  -Training_Accu
racy:  98.42  -time: 545.28
Validation accuracy 97.25
Iteration: 10/50[==============] -Error: 0.0164263995  -Training_Accu
racy:  98.47  -time: 610.11
Validation accuracy 97.27
Iteration: 11/50[==============] -Error: 0.0151239245  -Training_Accu
racy:  98.71  -time: 672.00
Validation accuracy 97.41
Iteration: 12/50[==============] -Error: 0.0137950257  -Training_Accu
racy:  98.84  -time: 734.38
Validation accuracy 97.37
Iteration: 13/50[==============] -Error: 0.0128876766  -Training_Accu
racy:  98.90  -time: 794.61
Validation accuracy 97.46000000000001
Iteration: 14/50[==============] -Error: 0.0119681845  -Training_Accu
racy:  98.96  -time: 856.82
Validation accuracy 97.48
Iteration: 15/50[==============] -Error: 0.0111748247  -Training_Accu
```

```
racy:  99.00  -time: 918.44
Validation accuracy 97.49
Iteration: 16/50[==============] -Error: 0.0104791168  -Training_Accu
racy:  99.11  -time: 980.99
Validation accuracy 97.56
Iteration: 17/50[==============] -Error: 0.0096664711  -Training_Accu
racy:  99.15  -time: 1042.56
Validation accuracy 97.42
Iteration: 18/50[==============] -Error: 0.0091813853  -Training_Accu
racy:  99.16  -time: 1104.90
Validation accuracy 97.56
Iteration: 19/50[==============] -Error: 0.0086185342  -Training_Accu
racy:  99.19  -time: 1166.57
Validation accuracy 97.50999999999999
Iteration: 20/50[==============] -Error: 0.0082035278  -Training_Accu
racy:  99.22  -time: 1228.93
Validation accuracy 97.42
Iteration: 21/50[==============] -Error: 0.0076346688  -Training_Accu
racy:  99.29  -time: 1291.03
Validation accuracy 97.52
Iteration: 22/50[==============] -Error: 0.0072668998  -Training_Accu
racy:  99.31  -time: 1353.72
Validation accuracy 97.57000000000001
Iteration: 23/50[==============] -Error: 0.0069076237  -Training_Accu
racy:  99.32  -time: 1415.57
Validation accuracy 97.54
Iteration: 24/50[==============] -Error: 0.0065390693  -Training_Accu
racy:  99.35  -time: 1478.46
Validation accuracy 97.52
Iteration: 25/50[==============] -Error: 0.0062351792  -Training_Accu
racy:  99.37  -time: 1540.55
Validation accuracy 97.6
Iteration: 26/50[==============] -Error: 0.0059602638  -Training_Accu
racy:  99.39  -time: 1602.78
Validation accuracy 97.55
Iteration: 27/50[==============] -Error: 0.0057109180  -Training_Accu
racy:  99.41  -time: 1663.05
Validation accuracy 97.68
Iteration: 28/50[==============] -Error: 0.0054068667  -Training_Accu
racy:  99.43  -time: 1724.40
Validation accuracy 97.64
Iteration: 29/50[==============] -Error: 0.0052152863  -Training_Accu
racy:  99.43  -time: 1785.88
Validation accuracy 97.56
Iteration: 30/50[==============] -Error: 0.0050039089  -Training_Accu
racy:  99.45  -time: 1848.33
Validation accuracy 97.69
Iteration: 31/50[==============] -Error: 0.0048308175  -Training_Accu
racy:  99.46  -time: 1910.24
Validation accuracy 97.72999999999999
Iteration: 32/50[==============] -Error: 0.0046537108  -Training_Accu
racy:  99.48  -time: 1974.50
Validation accuracy 97.65
Iteration: 33/50[==============] -Error: 0.0044985758  -Training_Accu
racy:  99.49  -time: 2036.34
Validation accuracy 97.68
Iteration: 34/50[==============] -Error: 0.0043319941  -Training_Accu
racy:  99.50  -time: 2098.40
Validation accuracy 97.68
Iteration: 35/50[==============] -Error: 0.0042035820  -Training_Accu
racy:  99.52  -time: 2160.34
```

```
Validation accuracy 97.74000000000001
Iteration: 36/50[==============] -Error: 0.0040694997  -Training_Accu
racy:  99.51  -time: 2222.85
Validation accuracy 97.77
Iteration: 37/50[==============] -Error: 0.0039564171  -Training_Accu
racy:  99.53  -time: 2284.84
Validation accuracy 97.78
Iteration: 38/50[==============] -Error: 0.0038451222  -Training_Accu
racy:  99.53  -time: 2347.43
Validation accuracy 97.71
Iteration: 39/50[==============] -Error: 0.0037357974  -Training_Accu
racy:  99.53  -time: 2409.99
Validation accuracy 97.69
Iteration: 40/50[==============] -Error: 0.0036574691  -Training_Accu
racy:  99.54  -time: 2472.09
Validation accuracy 97.72999999999999
Iteration: 41/50[==============] -Error: 0.0035934655  -Training_Accu
racy:  99.54  -time: 2534.01
Validation accuracy 97.68
Iteration: 42/50[==============] -Error: 0.0035207324  -Training_Accu
racy:  99.54  -time: 2593.68
Validation accuracy 97.77
Iteration: 43/50[==============] -Error: 0.0034610671  -Training_Accu
racy:  99.55  -time: 2655.07
Validation accuracy 97.78999999999999
Iteration: 44/50[==============] -Error: 0.0033932341  -Training_Accu
racy:  99.55  -time: 2716.62
Validation accuracy 97.72999999999999
Iteration: 45/50[==============] -Error: 0.0033368580  -Training_Accu
racy:  99.56  -time: 2779.30
Validation accuracy 97.77
Iteration: 46/50[==============] -Error: 0.0032729459  -Training_Accu
racy:  99.57  -time: 2841.04
Validation accuracy 97.83
Iteration: 47/50[==============] -Error: 0.0032266193  -Training_Accu
racy:  99.57  -time: 2905.31
Validation accuracy 97.8
Iteration: 48/50[==============] -Error: 0.0031753337  -Training_Accu
racy:  99.58  -time: 2967.18
Validation accuracy 97.8
Iteration: 49/50[==============] -Error: 0.0031232514  -Training_Accu
racy:  99.57  -time: 3029.90
Validation accuracy 97.77
Iteration: 50/50[==============] -Error: 0.0030596516  -Training_Accu
racy:  99.58  -time: 3091.97
Validation accuracy 97.82
```



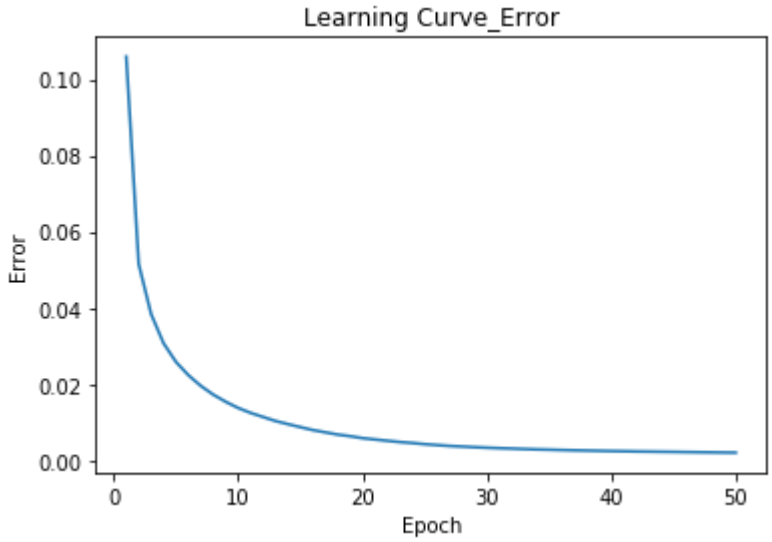


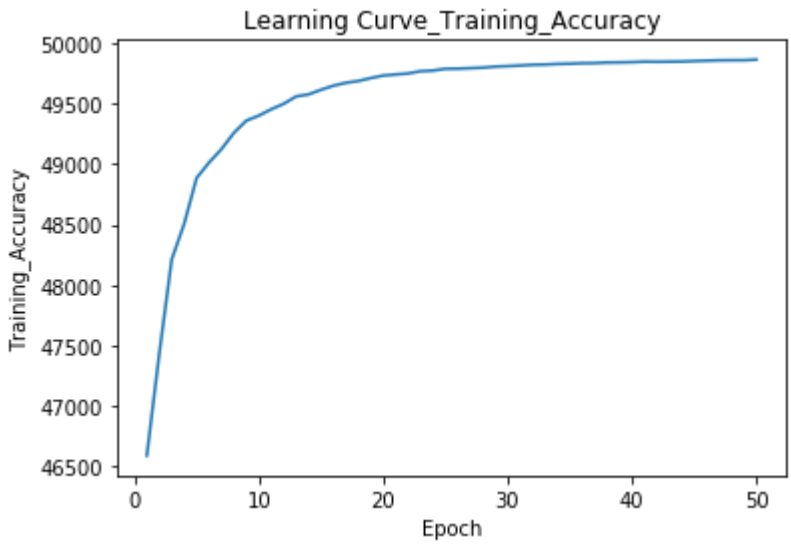

```
9782.0
Iteration:  1/50[==============] -Error: 0.1059214423  -Training_Accu
racy:  93.17  -time: 99.31
Validation accuracy 93.34
Iteration:  2/50[==============] -Error: 0.0515270623  -Training_Accu
racy:  94.86  -time: 201.57
Validation accuracy 94.71000000000001
Iteration:  3/50[==============] -Error: 0.0383784849  -Training_Accu
racy:  96.42  -time: 301.74
Validation accuracy 96.1
Iteration:  4/50[==============] -Error: 0.0307268811  -Training_Accu
racy:  97.01  -time: 404.04
Validation accuracy 96.46000000000001
Iteration:  5/50[==============] -Error: 0.0258218786  -Training_Accu
racy:  97.77  -time: 503.86
Validation accuracy 97.06
Iteration:  6/50[==============] -Error: 0.0223467948  -Training_Accu
racy:  98.03  -time: 608.22
Validation accuracy 97.25
Iteration:  7/50[==============] -Error: 0.0195369628  -Training_Accu
racy:  98.25  -time: 707.75
Validation accuracy 97.18
Iteration:  8/50[==============] -Error: 0.0172501375  -Training_Accu
racy:  98.52  -time: 810.36
Validation accuracy 97.43
```

```
Iteration:  9/50[==============] -Error: 0.0153779518  -Training_Accu
racy:  98.72  -time: 910.80
Validation accuracy 97.59
Iteration: 10/50[==============] -Error: 0.0137596647  -Training_Accu
racy:  98.80  -time: 1010.88
Validation accuracy 97.61999999999999
Iteration: 11/50[==============] -Error: 0.0124666206  -Training_Accu
racy:  98.91  -time: 1113.00
Validation accuracy 97.78
Iteration: 12/50[==============] -Error: 0.0113977039  -Training_Accu
racy:  99.00  -time: 1213.75
Validation accuracy 97.74000000000001
Iteration: 13/50[==============] -Error: 0.0103408522  -Training_Accu
racy:  99.12  -time: 1316.36
Validation accuracy 97.92
Iteration: 14/50[==============] -Error: 0.0095251676  -Training_Accu
racy:  99.15  -time: 1420.37
Validation accuracy 97.81
Iteration: 15/50[==============] -Error: 0.0087361404  -Training_Accu
racy:  99.23  -time: 1522.61
Validation accuracy 97.72999999999999
Iteration: 16/50[==============] -Error: 0.0079672669  -Training_Accu
racy:  99.30  -time: 1622.83
Validation accuracy 97.78999999999999
Iteration: 17/50[==============] -Error: 0.0073748638  -Training_Accu
racy:  99.35  -time: 1723.42
Validation accuracy 97.87
Iteration: 18/50[==============] -Error: 0.0067819910  -Training_Accu
racy:  99.38  -time: 1825.53
Validation accuracy 97.92
Iteration: 19/50[==============] -Error: 0.0063600064  -Training_Accu
racy:  99.42  -time: 1926.13
Validation accuracy 97.91
Iteration: 20/50[==============] -Error: 0.0058380806  -Training_Accu
racy:  99.47  -time: 2026.37
Validation accuracy 97.97
Iteration: 21/50[==============] -Error: 0.0054943980  -Training_Accu
racy:  99.48  -time: 2129.08
Validation accuracy 97.95
Iteration: 22/50[==============] -Error: 0.0051535523  -Training_Accu
racy:  99.50  -time: 2230.73
Validation accuracy 97.97
Iteration: 23/50[==============] -Error: 0.0048187460  -Training_Accu
racy:  99.54  -time: 2328.72
Validation accuracy 97.99
Iteration: 24/50[==============] -Error: 0.0045726991  -Training_Accu
racy:  99.55  -time: 2428.48
Validation accuracy 98.02
Iteration: 25/50[==============] -Error: 0.0042593024  -Training_Accu
racy:  99.58  -time: 2529.12
Validation accuracy 98.07000000000001
Iteration: 26/50[==============] -Error: 0.0040298637  -Training_Accu
racy:  99.58  -time: 2627.47
Validation accuracy 98.08
Iteration: 27/50[==============] -Error: 0.0037963059  -Training_Accu
racy:  99.59  -time: 2730.64
Validation accuracy 98.00999999999999
Iteration: 28/50[==============] -Error: 0.0036436219  -Training_Accu
racy:  99.60  -time: 2832.11
Validation accuracy 98.09
Iteration: 29/50[==============] -Error: 0.0034744344  -Training_Accu
```

```
racy:  99.61  -time: 2934.71
Validation accuracy 98.07000000000001
Iteration: 30/50[==============] -Error: 0.0033353458  -Training_Accu
racy:  99.62  -time: 3037.26
Validation accuracy 98.07000000000001
Iteration: 31/50[==============] -Error: 0.0032118758  -Training_Accu
racy:  99.63  -time: 3137.26
Validation accuracy 98.04
Iteration: 32/50[==============] -Error: 0.0031120599  -Training_Accu
racy:  99.64  -time: 3235.66
Validation accuracy 98.09
Iteration: 33/50[==============] -Error: 0.0029935771  -Training_Accu
racy:  99.65  -time: 3335.70
Validation accuracy 98.06
Iteration: 34/50[==============] -Error: 0.0028913529  -Training_Accu
racy:  99.66  -time: 3438.46
Validation accuracy 98.06
Iteration: 35/50[==============] -Error: 0.0028198740  -Training_Accu
racy:  99.66  -time: 3538.77
Validation accuracy 98.06
Iteration: 36/50[==============] -Error: 0.0027286862  -Training_Accu
racy:  99.67  -time: 3637.07
Validation accuracy 98.09
Iteration: 37/50[==============] -Error: 0.0026536167  -Training_Accu
racy:  99.67  -time: 3737.10
Validation accuracy 98.05
Iteration: 38/50[==============] -Error: 0.0025879550  -Training_Accu
racy:  99.68  -time: 3836.96
Validation accuracy 98.1
Iteration: 39/50[==============] -Error: 0.0025226469  -Training_Accu
racy:  99.68  -time: 3937.39
Validation accuracy 98.11
Iteration: 40/50[==============] -Error: 0.0024642357  -Training_Accu
racy:  99.69  -time: 4041.25
Validation accuracy 98.08
Iteration: 41/50[==============] -Error: 0.0024162062  -Training_Accu
racy:  99.70  -time: 4142.32
Validation accuracy 98.09
Iteration: 42/50[==============] -Error: 0.0023562364  -Training_Accu
racy:  99.69  -time: 4242.65
Validation accuracy 98.09
Iteration: 43/50[==============] -Error: 0.0023220710  -Training_Accu
racy:  99.70  -time: 4341.89
Validation accuracy 98.06
Iteration: 44/50[==============] -Error: 0.0022805246  -Training_Accu
racy:  99.70  -time: 4441.93
Validation accuracy 98.11999999999999
Iteration: 45/50[==============] -Error: 0.0022404254  -Training_Accu
racy:  99.71  -time: 4542.26
Validation accuracy 98.06
Iteration: 46/50[==============] -Error: 0.0021936029  -Training_Accu
racy:  99.71  -time: 4642.70
Validation accuracy 98.09
Iteration: 47/50[==============] -Error: 0.0021525965  -Training_Accu
racy:  99.72  -time: 4746.02
Validation accuracy 98.1
Iteration: 48/50[==============] -Error: 0.0021141793  -Training_Accu
racy:  99.72  -time: 4846.55
Validation accuracy 98.09
Iteration: 49/50[==============] -Error: 0.0020768998  -Training_Accu
racy:  99.72  -time: 4948.21
```

```
Validation accuracy 98.05
Iteration: 50/50[==============] -Error: 0.0020428318  -Training_Accu
racy:  99.73  -time: 5048.40
Validation accuracy 98.11999999999999
```



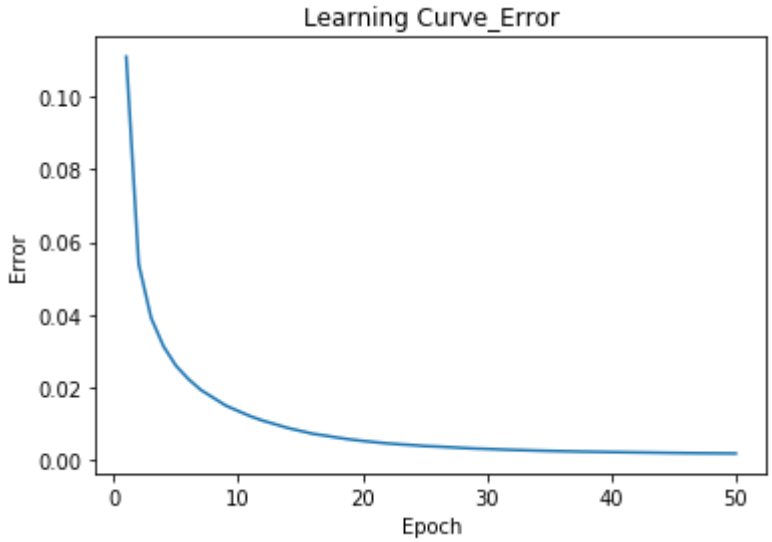


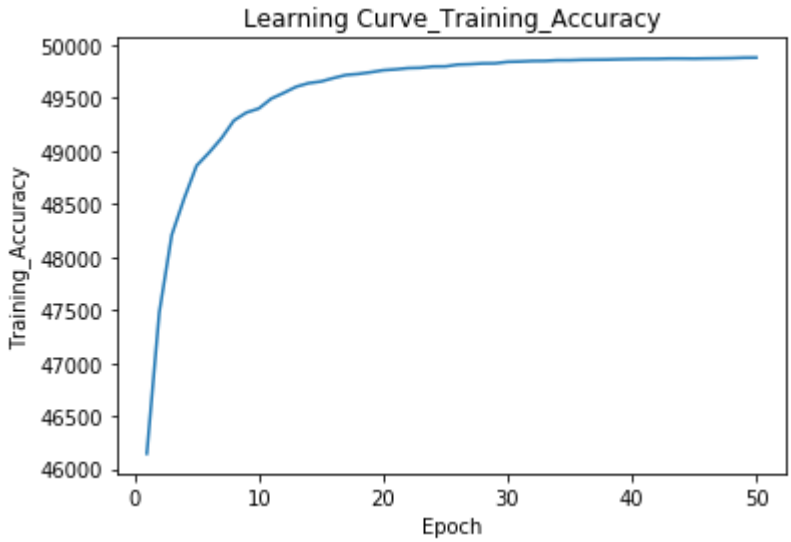

```
9812.0
Iteration:  1/50[==============] -Error: 0.1109705403  -Training_Accu
racy:  92.29  -time: 147.58
Validation accuracy 92.28
Iteration:  2/50[==============] -Error: 0.0539136829  -Training_Accu
racy:  94.95  -time: 294.49
Validation accuracy 94.92
Iteration:  3/50[==============] -Error: 0.0390772051  -Training_Accu
racy:  96.41  -time: 441.84
```

```
Validation accuracy 96.3
Iteration:  4/50[==============] -Error: 0.0312298663  -Training_Accu
racy:  97.10  -time: 589.34
Validation accuracy 96.57
Iteration:  5/50[==============] -Error: 0.0259170025  -Training_Accu
racy:  97.72  -time: 737.00
Validation accuracy 97.02
Iteration:  6/50[==============] -Error: 0.0222465027  -Training_Accu
racy:  97.97  -time: 885.89
Validation accuracy 97.08
Iteration:  7/50[==============] -Error: 0.0192613549  -Training_Accu
racy:  98.24  -time: 1032.69
Validation accuracy 97.32
Iteration:  8/50[==============] -Error: 0.0171085852  -Training_Accu
racy:  98.57  -time: 1178.98
Validation accuracy 97.57000000000001
Iteration:  9/50[==============] -Error: 0.0149639784  -Training_Accu
racy:  98.72  -time: 1327.50
Validation accuracy 97.58
Iteration: 10/50[==============] -Error: 0.0134554138  -Training_Accu
racy:  98.80  -time: 1476.39
Validation accuracy 97.68
Iteration: 11/50[==============] -Error: 0.0119953733  -Training_Accu
racy:  98.98  -time: 1624.39
Validation accuracy 97.78999999999999
Iteration: 12/50[==============] -Error: 0.0107769206  -Training_Accu
racy:  99.09  -time: 1772.30
Validation accuracy 97.87
Iteration: 13/50[==============] -Error: 0.0097954244  -Training_Accu
racy:  99.21  -time: 1920.55
Validation accuracy 97.96000000000001
Iteration: 14/50[==============] -Error: 0.0087739138  -Training_Accu
racy:  99.28  -time: 2069.44
Validation accuracy 97.92999999999999
Iteration: 15/50[==============] -Error: 0.0079855296  -Training_Accu
racy:  99.31  -time: 2217.40
Validation accuracy 97.85000000000001
Iteration: 16/50[==============] -Error: 0.0071976091  -Training_Accu
racy:  99.37  -time: 2366.09
Validation accuracy 98.08
Iteration: 17/50[==============] -Error: 0.0066845226  -Training_Accu
racy:  99.43  -time: 2514.18
Validation accuracy 97.98
Iteration: 18/50[==============] -Error: 0.0061230620  -Training_Accu
racy:  99.45  -time: 2661.78
Validation accuracy 97.97
Iteration: 19/50[==============] -Error: 0.0056315743  -Training_Accu
racy:  99.48  -time: 2809.68
Validation accuracy 98.13
Iteration: 20/50[==============] -Error: 0.0052356051  -Training_Accu
racy:  99.52  -time: 2957.74
Validation accuracy 98.17
Iteration: 21/50[==============] -Error: 0.0048758748  -Training_Accu
racy:  99.54  -time: 3104.71
Validation accuracy 98.19
Iteration: 22/50[==============] -Error: 0.0045495806  -Training_Accu
racy:  99.56  -time: 3252.55
Validation accuracy 98.13
Iteration: 23/50[==============] -Error: 0.0042927591  -Training_Accu
racy:  99.57  -time: 3400.93
Validation accuracy 98.18
```

```
Iteration: 24/50[==============] -Error: 0.0040755150  -Training_Accu
racy:  99.59  -time: 3548.45
Validation accuracy 98.15
Iteration: 25/50[==============] -Error: 0.0038144778  -Training_Accu
racy:  99.59  -time: 3695.30
Validation accuracy 98.05
Iteration: 26/50[==============] -Error: 0.0036715662  -Training_Accu
racy:  99.63  -time: 3842.84
Validation accuracy 98.13
Iteration: 27/50[==============] -Error: 0.0035010397  -Training_Accu
racy:  99.63  -time: 3991.42
Validation accuracy 98.22
Iteration: 28/50[==============] -Error: 0.0032981301  -Training_Accu
racy:  99.65  -time: 4138.76
Validation accuracy 98.11999999999999
Iteration: 29/50[==============] -Error: 0.0031294499  -Training_Accu
racy:  99.65  -time: 4286.80
Validation accuracy 98.16
Iteration: 30/50[==============] -Error: 0.0030247706  -Training_Accu
racy:  99.68  -time: 4434.38
Validation accuracy 98.15
Iteration: 31/50[==============] -Error: 0.0028820509  -Training_Accu
racy:  99.68  -time: 4582.65
Validation accuracy 98.17
Iteration: 32/50[==============] -Error: 0.0027578719  -Training_Accu
racy:  99.69  -time: 4730.69
Validation accuracy 98.02
Iteration: 33/50[==============] -Error: 0.0026633824  -Training_Accu
racy:  99.70  -time: 4878.26
Validation accuracy 98.16
Iteration: 34/50[==============] -Error: 0.0025620839  -Training_Accu
racy:  99.71  -time: 5026.43
Validation accuracy 98.14
Iteration: 35/50[==============] -Error: 0.0024733469  -Training_Accu
racy:  99.71  -time: 5174.30
Validation accuracy 98.22
Iteration: 36/50[==============] -Error: 0.0023944286  -Training_Accu
racy:  99.72  -time: 5322.97
Validation accuracy 98.13
Iteration: 37/50[==============] -Error: 0.0023246938  -Training_Accu
racy:  99.72  -time: 5470.69
Validation accuracy 98.1
Iteration: 38/50[==============] -Error: 0.0022629101  -Training_Accu
racy:  99.72  -time: 5618.25
Validation accuracy 98.15
Iteration: 39/50[==============] -Error: 0.0022103686  -Training_Accu
racy:  99.73  -time: 5766.19
Validation accuracy 98.09
Iteration: 40/50[==============] -Error: 0.0021548077  -Training_Accu
racy:  99.73  -time: 5915.03
Validation accuracy 98.13
Iteration: 41/50[==============] -Error: 0.0021085282  -Training_Accu
racy:  99.73  -time: 6063.88
Validation accuracy 98.15
Iteration: 42/50[==============] -Error: 0.0020567283  -Training_Accu
racy:  99.73  -time: 6211.81
Validation accuracy 98.13
Iteration: 43/50[==============] -Error: 0.0020220495  -Training_Accu
racy:  99.74  -time: 6360.10
Validation accuracy 98.13
Iteration: 44/50[==============] -Error: 0.0019639215  -Training_Accu
```

```
racy:  99.74  -time: 6508.56
Validation accuracy 98.19
Iteration: 45/50[==============] -Error: 0.0019184973  -Training_Accu
racy:  99.74  -time: 6656.35
Validation accuracy 98.08
Iteration: 46/50[==============] -Error: 0.0018784104  -Training_Accu
racy:  99.74  -time: 6804.33
Validation accuracy 98.15
Iteration: 47/50[==============] -Error: 0.0018502466  -Training_Accu
racy:  99.74  -time: 6952.24
Validation accuracy 98.13
Iteration: 48/50[==============] -Error: 0.0018212858  -Training_Accu
racy:  99.75  -time: 7099.20
Validation accuracy 98.13
Iteration: 49/50[==============] -Error: 0.0017982556  -Training_Accu
racy:  99.76  -time: 7247.18
Validation accuracy 98.1
Iteration: 50/50[==============] -Error: 0.0017692966  -Training_Accu
racy:  99.76  -time: 7395.36
Validation accuracy 98.24000000000001
```





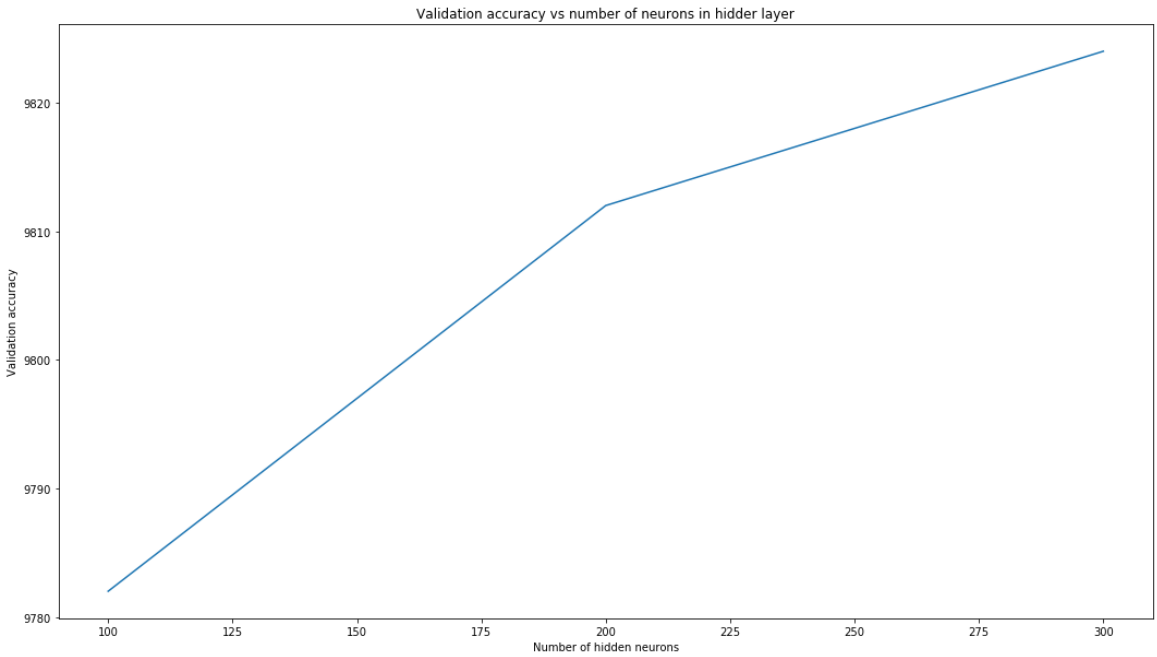

```
9824.0
```

In [33]:

```
my_mnist_net_100_hidden.save('saved_models/NN_MNIST_100_hidden_DEF')
my_mnist_net_200_hidden.save('saved_models/NN_MNIST_200_hidden_DEF')
my_mnist_net_300_hidden.save('saved_models/NN_MNIST_300_hidden_DEF')
```

In [30]:

```
plt.figure(figsize=(18,10))
plt.xlabel("Number of hidden neurons")
plt.ylabel("Validation accuracy")
plt.title("Validation accuracy vs number of neurons in hidder layer")
plt.plot([100, 200, 300], [val_accuracies_100[-1], val_accuracies_200[-1], val_accu
plt.show()
```



Looking at the plot above we can quickly see that 300 neurons perform the best among the tested configurations (the validation accuracy is the highest while the learning rate was left constant - the one that we found to be the best before).

Question 2.2.5 : Add one additionnal hidden layers and train your network, discuss your results with different setting.

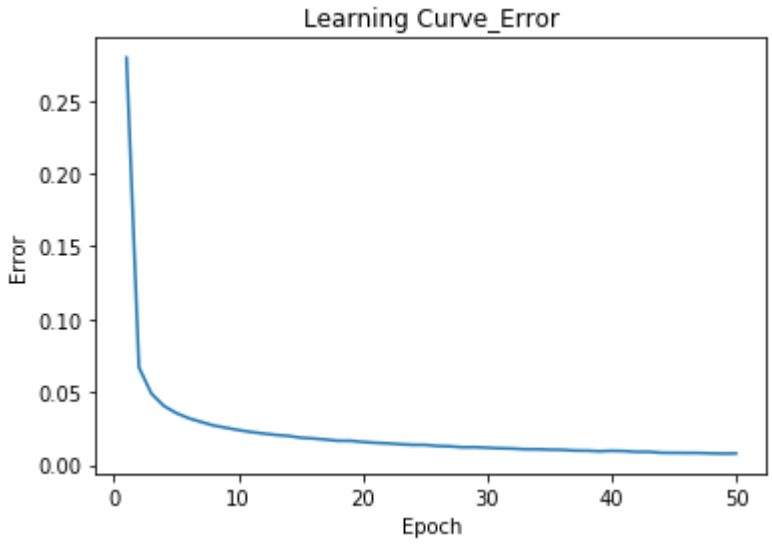
In [31]:

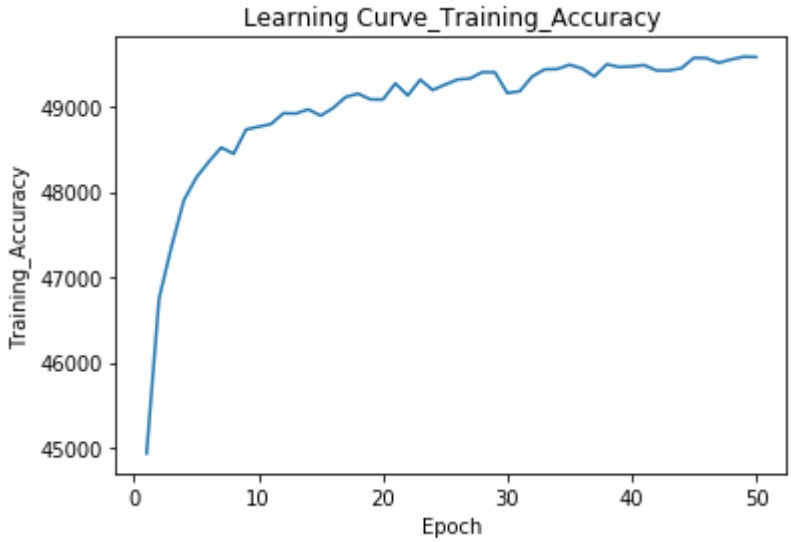
```
lementation goes here
alNetwork2 import NeuralNetwork2

he network

es = 784
des_1 = 30
des_2 = 30
des = 10

net = NeuralNetwork2(input_nodes, hidden_nodes_1, hidden_nodes_2, output_nodes)
 = my_mnist_net.train(training_data,validation_data)
y_mnist_net.predict(test_data)/100
curacy with default learning rate = %f" % acc_1)

rate = 1
net_2 = my_mnist_net = NeuralNetwork2(input_nodes, hidden_nodes_1, hidden_nodes_2, 
 = my_mnist_net_2.train(training_data, validation_data)
y_mnist_net_2.predict(test_data)/100
aring rate = %f, accuracy = %f" % (learning_rate, acc_2))
```

```
Iteration:  1/50[==============] -Error: 0.2794236659  -Training_Accu
racy:  89.87  -time: 36.95
Iteration:  2/50[==============] -Error: 0.0670139700  -Training_Accu
racy:  93.49  -time: 79.16
Iteration:  3/50[==============] -Error: 0.0490478529  -Training_Accu
racy:  94.71  -time: 133.62
Iteration:  4/50[==============] -Error: 0.0405936985  -Training_Accu
racy:  95.80  -time: 172.40
Iteration:  5/50[==============] -Error: 0.0356868360  -Training_Accu
racy:  96.33  -time: 213.34
Iteration:  6/50[==============] -Error: 0.0321592929  -Training_Accu
racy:  96.70  -time: 254.66
Iteration:  7/50[==============] -Error: 0.0295883383  -Training_Accu
racy:  97.04  -time: 295.28
Iteration:  8/50[==============] -Error: 0.0271886497  -Training_Accu
racy:  96.89  -time: 338.94
Iteration:  9/50[==============] -Error: 0.0255985267  -Training_Accu
racy:  97.46  -time: 380.90
Iteration: 10/50[==============] -Error: 0.0240409727  -Training_Accu
racy:  97.52  -time: 421.49
Iteration: 11/50[==============] -Error: 0.0227053358  -Training_Accu
racy:  97.59  -time: 462.54
Iteration: 12/50[==============] -Error: 0.0216514846  -Training_Accu
racy:  97.84  -time: 519.31
Iteration: 13/50[==============] -Error: 0.0206833104  -Training_Accu
racy:  97.83  -time: 562.69
Iteration: 14/50[==============] -Error: 0.0200511510  -Training_Accu
racy:  97.93  -time: 616.99
Iteration: 15/50[==============] -Error: 0.0187757610  -Training_Accu
racy:  97.78  -time: 657.35
Iteration: 16/50[==============] -Error: 0.0182692819  -Training_Accu
racy:  97.96  -time: 714.95
Iteration: 17/50[==============] -Error: 0.0175153730  -Training_Accu
racy:  98.22  -time: 759.05
Iteration: 18/50[==============] -Error: 0.0166820688  -Training_Accu
racy:  98.30  -time: 808.66
```

```
Iteration: 19/50[==============] -Error: 0.0166641841  -Training_Accu
racy:  98.16  -time: 858.33
Iteration: 20/50[==============] -Error: 0.0158684372  -Training_Accu
racy:  98.16  -time: 904.04
Iteration: 21/50[==============] -Error: 0.0152971227  -Training_Accu
racy:  98.54  -time: 947.41
Iteration: 22/50[==============] -Error: 0.0149235978  -Training_Accu
racy:  98.26  -time: 999.78
Iteration: 23/50[==============] -Error: 0.0143282756  -Training_Accu
racy:  98.63  -time: 1046.49
Iteration: 24/50[==============] -Error: 0.0139048971  -Training_Accu
racy:  98.38  -time: 1092.30
Iteration: 25/50[==============] -Error: 0.0138614484  -Training_Accu
racy:  98.51  -time: 1142.13
Iteration: 26/50[==============] -Error: 0.0130942154  -Training_Accu
racy:  98.63  -time: 1196.48
Iteration: 27/50[==============] -Error: 0.0128778333  -Training_Accu
racy:  98.65  -time: 1245.75
Iteration: 28/50[==============] -Error: 0.0122478541  -Training_Accu
racy:  98.80  -time: 1291.20
Iteration: 29/50[==============] -Error: 0.0122878322  -Training_Accu
racy:  98.80  -time: 1337.26
Iteration: 30/50[==============] -Error: 0.0119129242  -Training_Accu
racy:  98.31  -time: 1382.65
Iteration: 31/50[==============] -Error: 0.0115341710  -Training_Accu
racy:  98.35  -time: 1429.38
Iteration: 32/50[==============] -Error: 0.0113410412  -Training_Accu
racy:  98.70  -time: 1480.62
Iteration: 33/50[==============] -Error: 0.0108938225  -Training_Accu
racy:  98.87  -time: 1523.05
Iteration: 34/50[==============] -Error: 0.0108642168  -Training_Accu
racy:  98.87  -time: 1567.30
Iteration: 35/50[==============] -Error: 0.0106036668  -Training_Accu
racy:  98.98  -time: 1611.70
Iteration: 36/50[==============] -Error: 0.0104815881  -Training_Accu
racy:  98.89  -time: 1656.52
Iteration: 37/50[==============] -Error: 0.0099562523  -Training_Accu
racy:  98.70  -time: 1711.12
Iteration: 38/50[==============] -Error: 0.0098983011  -Training_Accu
racy:  98.99  -time: 1791.18
Iteration: 39/50[==============] -Error: 0.0095054373  -Training_Accu
racy:  98.92  -time: 1875.05
Iteration: 40/50[==============] -Error: 0.0098076036  -Training_Accu
racy:  98.94  -time: 1934.68
Iteration: 41/50[==============] -Error: 0.0096275943  -Training_Accu
racy:  98.97  -time: 1983.41
Iteration: 42/50[==============] -Error: 0.0090763788  -Training_Accu
racy:  98.84  -time: 2028.68
Iteration: 43/50[==============] -Error: 0.0091255570  -Training_Accu
racy:  98.84  -time: 2083.64
Iteration: 44/50[==============] -Error: 0.0084579227  -Training_Accu
racy:  98.89  -time: 2150.97
Iteration: 45/50[==============] -Error: 0.0083798834  -Training_Accu
racy:  99.14  -time: 2197.31
Iteration: 46/50[==============] -Error: 0.0083340243  -Training_Accu
racy:  99.13  -time: 2247.07
Iteration: 47/50[==============] -Error: 0.0083071795  -Training_Accu
racy:  99.02  -time: 2296.19
Iteration: 48/50[==============] -Error: 0.0080452325  -Training_Accu
racy:  99.10  -time: 2343.54
Iteration: 49/50[==============] -Error: 0.0078572695  -Training_Accu
```

```
racy:  99.16  -time: 2389.03
Iteration: 50/50[==============] -Error: 0.0080175108  -Training_Accu
racy:  99.16  -time: 2444.83
```

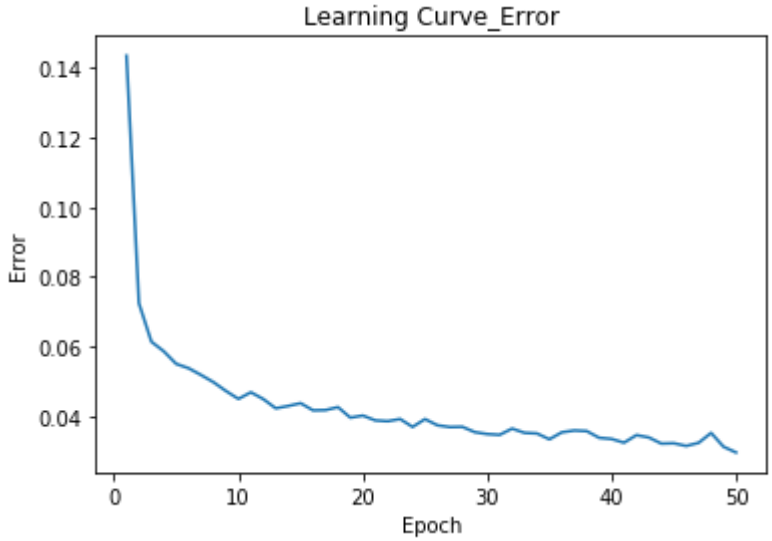


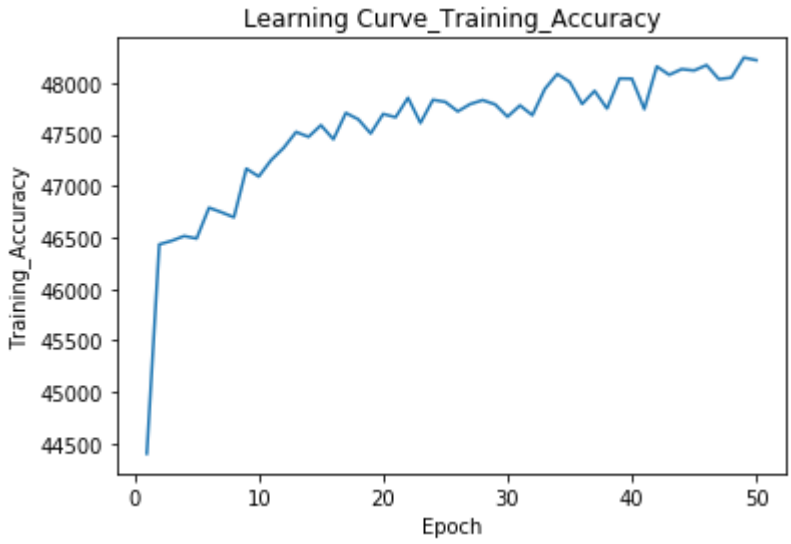



```
Accuracy with default learning rate = 96.450000
Iteration:  1/50[==============] -Error: 0.1434018567  -Training_Accu
racy:  88.80  -time: 59.77
Iteration:  2/50[==============] -Error: 0.0724579397  -Training_Accu
racy:  92.87  -time: 133.23
Iteration:  3/50[==============] -Error: 0.0614478681  -Training_Accu
racy:  92.94  -time: 188.54
Iteration:  4/50[==============] -Error: 0.0586879438  -Training_Accu
racy:  93.03  -time: 251.20
Iteration:  5/50[==============] -Error: 0.0550862141  -Training_Accu
racy:  92.98  -time: 298.27
Iteration:  6/50[==============] -Error: 0.0538291738  -Training_Accu
racy:  93.58  -time: 359.54
Iteration:  7/50[==============] -Error: 0.0518926217  -Training_Accu
racy:  93.49  -time: 404.16
Iteration:  8/50[==============] -Error: 0.0498259405  -Training_Accu
racy:  93.39  -time: 450.08
Iteration:  9/50[==============] -Error: 0.0473083150  -Training_Accu
racy:  94.34  -time: 497.68
Iteration: 10/50[==============] -Error: 0.0450449853  -Training_Accu
racy:  94.18  -time: 547.62
Iteration: 11/50[==============] -Error: 0.0469454029  -Training_Accu
```

```
racy:  94.50  -time: 599.33
Iteration: 12/50[==============] -Error: 0.0449948155  -Training_Accu
racy:  94.74  -time: 647.84
Iteration: 13/50[==============] -Error: 0.0423519886  -Training_Accu
racy:  95.05  -time: 714.94
Iteration: 14/50[==============] -Error: 0.0430194555  -Training_Accu
racy:  94.95  -time: 769.72
Iteration: 15/50[==============] -Error: 0.0438157453  -Training_Accu
racy:  95.18  -time: 823.06
Iteration: 16/50[==============] -Error: 0.0417687525  -Training_Accu
racy:  94.91  -time: 868.48
Iteration: 17/50[==============] -Error: 0.0418386556  -Training_Accu
racy:  95.42  -time: 915.76
Iteration: 18/50[==============] -Error: 0.0426896224  -Training_Accu
racy:  95.29  -time: 963.26
Iteration: 19/50[==============] -Error: 0.0396712078  -Training_Accu
racy:  95.02  -time: 1007.19
Iteration: 20/50[==============] -Error: 0.0402731641  -Training_Accu
racy:  95.40  -time: 1054.86
Iteration: 21/50[==============] -Error: 0.0388866615  -Training_Accu
racy:  95.33  -time: 1101.27
Iteration: 22/50[==============] -Error: 0.0386609251  -Training_Accu
racy:  95.71  -time: 1148.11
Iteration: 23/50[==============] -Error: 0.0392769739  -Training_Accu
racy:  95.22  -time: 1197.05
Iteration: 24/50[==============] -Error: 0.0370152321  -Training_Accu
racy:  95.67  -time: 1240.03
Iteration: 25/50[==============] -Error: 0.0392559376  -Training_Accu
racy:  95.63  -time: 1283.32
Iteration: 26/50[==============] -Error: 0.0374844273  -Training_Accu
racy:  95.45  -time: 1329.13
Iteration: 27/50[==============] -Error: 0.0369786955  -Training_Accu
racy:  95.59  -time: 1372.00
Iteration: 28/50[==============] -Error: 0.0370488658  -Training_Accu
racy:  95.67  -time: 1419.14
Iteration: 29/50[==============] -Error: 0.0355222735  -Training_Accu
racy:  95.58  -time: 1463.31
Iteration: 30/50[==============] -Error: 0.0349676884  -Training_Accu
racy:  95.34  -time: 1505.55
Iteration: 31/50[==============] -Error: 0.0347479866  -Training_Accu
racy:  95.56  -time: 1552.30
Iteration: 32/50[==============] -Error: 0.0365337114  -Training_Accu
racy:  95.37  -time: 1594.21
Iteration: 33/50[==============] -Error: 0.0353218593  -Training_Accu
racy:  95.88  -time: 1637.16
Iteration: 34/50[==============] -Error: 0.0351470617  -Training_Accu
racy:  96.17  -time: 1683.34
Iteration: 35/50[==============] -Error: 0.0334686563  -Training_Accu
racy:  96.02  -time: 1725.43
Iteration: 36/50[==============] -Error: 0.0354799648  -Training_Accu
racy:  95.59  -time: 1770.31
Iteration: 37/50[==============] -Error: 0.0359958504  -Training_Accu
racy:  95.85  -time: 1814.99
Iteration: 38/50[==============] -Error: 0.0358263921  -Training_Accu
racy:  95.50  -time: 1857.40
Iteration: 39/50[==============] -Error: 0.0338975743  -Training_Accu
racy:  96.09  -time: 1904.47
Iteration: 40/50[==============] -Error: 0.0335704925  -Training_Accu
racy:  96.08  -time: 1947.96
Iteration: 41/50[==============] -Error: 0.0324956029  -Training_Accu
racy:  95.49  -time: 1990.37
```

```
Iteration: 42/50[==============] -Error: 0.0346704035  -Training_Accu
racy:  96.32  -time: 2046.75
Iteration: 43/50[==============] -Error: 0.0339883545  -Training_Accu
racy:  96.16  -time: 2091.53
Iteration: 44/50[==============] -Error: 0.0322370246  -Training_Accu
racy:  96.27  -time: 2137.40
Iteration: 45/50[==============] -Error: 0.0323357858  -Training_Accu
racy:  96.24  -time: 2183.93
Iteration: 46/50[==============] -Error: 0.0315695196  -Training_Accu
racy:  96.35  -time: 2228.40
Iteration: 47/50[==============] -Error: 0.0324596413  -Training_Accu
racy:  96.07  -time: 2274.82
Iteration: 48/50[==============] -Error: 0.0352979434  -Training_Accu
racy:  96.10  -time: 2320.56
Iteration: 49/50[==============] -Error: 0.0313380003  -Training_Accu
racy:  96.49  -time: 2378.84
Iteration: 50/50[==============] -Error: 0.0296932186  -Training_Accu
racy:  96.44  -time: 2432.44
```






```
Learing rate = 1.000000, accuracy = 95.650000
```

In [34]:

```
my_mnist_net.save('saved_models/NN_MNIST_2_HIDDEN_LAYERS_DEF')
my_mnist_net_2.save('saved_models/NN_MNIST_2_HIDDEN_LAYERS_1-0')
```

Keeping the best learning rate from before (0.1) and adding an additional hidden layer that has the same shape as the one before, improves the test accuracy by around 0.4%. This result is still better than the one with a learning rate of 1.0. Therefore, we can say that adding one hidden layer only leads to a slight improvement in test accuracy in this particular case. For a learning rate of 1 the result even worsened by 0.05%. So for now, adding more neurons had a bigger impact than adding one additional layer. However, it remains to be seen what happens if we further increase the number of layers.

In [ ]:

```

```