Classification of Letters with Multilayer Perceptron

Feed Forward Neural Networks

Artur Sahakyan, Arbi Balaban

Machine Learning B
Instructor:
Elen Vardanyan

Zaven and Sonia Akian CSE
American University of Armenia
Armenia

Spring 2024

Abstract

Deep Learning has introduced many approaches to image classification with different architectures, such as Convolutional Neural Networks(CNN), ResNets, Visual Geometry Group(VGG), Inception Networks. Nevertheless, on simpler datasets, Multilayer Perceptrons can be used for classification, presenting some challenges depending on the objective. This paper focuses on our attempts at building a feed forward neural network, which can classify letters with high accuracy.

Contents

Chapter 1

Introduction

Our project is about letter classification with the help of feed forward neural nets. During this period we have worked on the implementation of a neural network, which classifies with 90%+ accuracy rate.

We looked through different activation methods, impact of size of hidden layers and stopped on an architecture with optimal parameters and sufficient performance on the classification of letters task.

Chapter 2

Data Pre-processing

The dataset for handwritten letters we chose is Extended MNIST.

The EMNIST Letters dataset merges a balanced set of the uppercase and lowercase letters into a single 26-class task.

Training and test sets are separately available by the authors. After loading it we analyzed the dataset. The first column was for labels, while the corresponding row for each label contained 784 pixels of a single image. The testing dataset readily available did not suit the MLP classifier that sklearn has(due to NaNs after one hot encoding) and our model would also have to skip many nans (after applying training set's one-hot transformer's fitting on testing set, the test set had 7 labels less than the training set). So we split the training dataset into train and test sets (71039 train images, 17760 test images). Labels were then one-hot encoded. The resulting CSV files were then loaded into a Pandas Dataframe object and shuffled.

The pixels have been normalized.

For showcasing using Matplotlib we experimented with the right angle and horizontal flip for the pixel data provided in dataset. For training images we flipped horizontally the rotated by 90 degrees matrix, which was obtained by reshaping each row to a 28×28 matrix. For testing images we also flipped horizontally and rotated by 90 degrees.

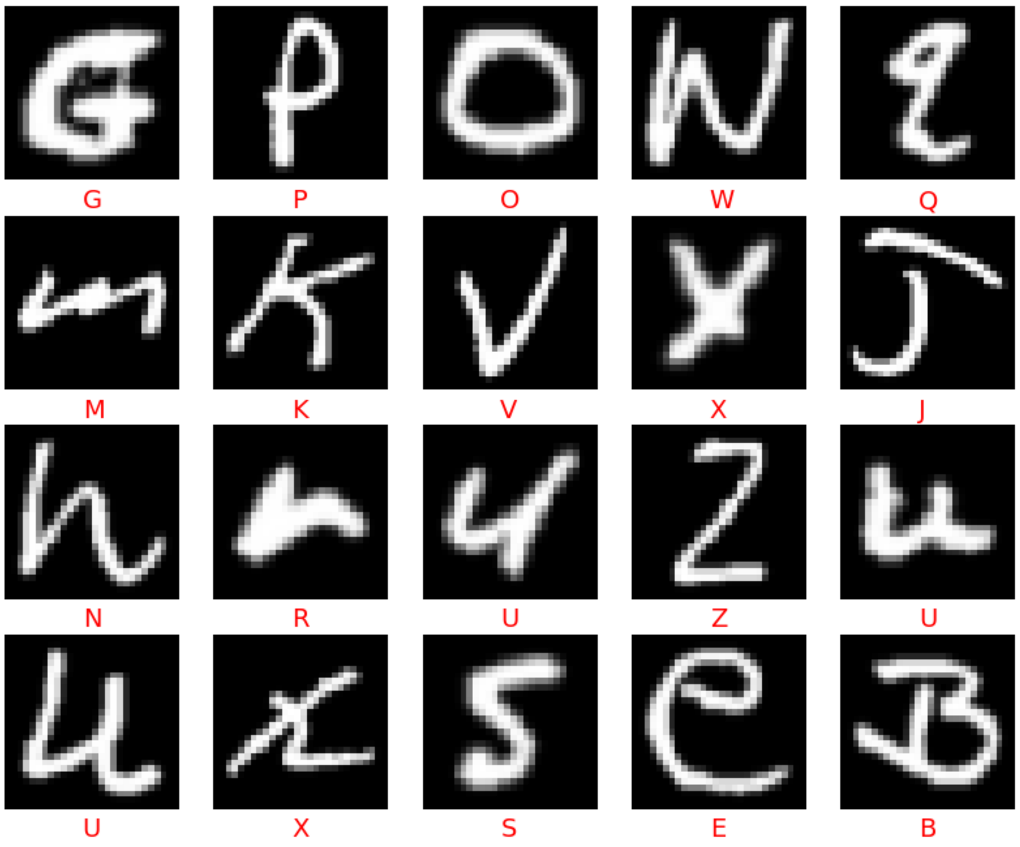

Figure 2.1: Displaying characters(letters) from the training dataset.

Chapter 3

Neural Network Architecture

3.1 Theoretical description

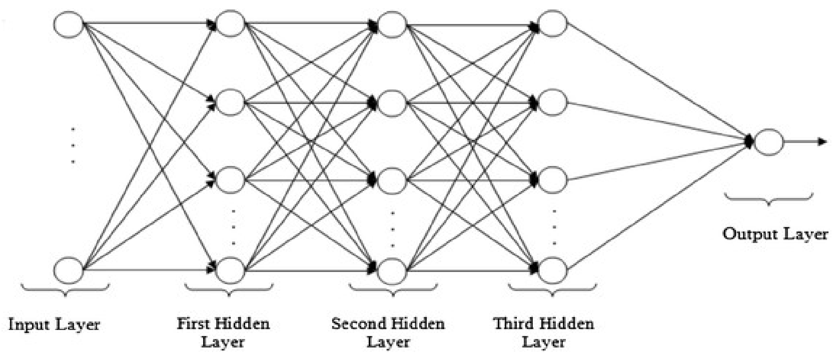

Figure 3.1: An illustration of a 3 hidden layer feed forward neural network

Since this is a multilayer perceptron, here the reader won't observe convolutional layers and more complex architectures. The architecture is composed of linear layers with activations after each [BG94].

More specifically, the feed forward neural network has 1 input layer, 3 hidden layers and 1 output layer.

The input size is determined by the number of pixels of each image, which in the case of letters dataset is 28×28.

The first hidden layer has size of 256, then decreasing with value of 128 in second hidden layer and 64 in the third hidden layer. The output size corresponds to the number of labels of dataset, that is 26.

Neurons of those layers are connected through weights, which are initialized from a random uniform distribution in range -0.5, 0.5 and have corresponding sizes for connecting the input to first hidden layer, then first hidden layer to second, second to third, and third hidden layer to output layer.

In forward propagation each layer has then been activated by Softplus activations.

Softplus is a smooth approximation to the ReLU activation function [Bis06] . Through Softplus we dealt with the ”vanishing” gradient problem in the back propagation.

$$f(x) = \ln(1 + e^x)$$

And it is generally better to have functions which have non-zero gradients as the input value gets larger.

For the back propagation we calculated the derivatives of the linear layers and by chain rule multiplied with the corresponding derivative of activation function Softplus.

$$f'(x) = \frac{d}{dx} \ln(1 + e^x) = \frac{1}{1 + e^x} \cdot e^x$$

3.2 Our Implementation

For the implementation of the project we used Python. For data preparation we used Pandas and Scikit-learn to perform one-hot encoding.

In the algorithm we used Numpy for computations.

The learned parameters, weights have been saved into .npy arrays, and then they are readily loaded and the classification loop displays the results.

3.2.1 Scikit-learn’s MLP classifier

Additionally, we have created an instance of MLP classifier object implemented in Scikit-learn and conducted training of 20 epochs. We initialized the object with the same hidden layer sizes and number that we implemented from scratch. Scikit-learn’s classifier uses ReLU activation by default, so we conclude that due to this difference(we used Softplus) on training set the Scikit-learn classifier has 1% higher accuracy on training (95%), while our model has 2% higher accuracy on testing set (87%).

Chapter 4

Experiments and Results

Our experiments were aimed at increasing the rate of accuracy during training and giving correct classifications during testing.

It was challenging in a way that the neural network is composed of only linear layers, not convolutional, and essentially there is no stack of complex blocks to gradually learn from high to low level features. Basically, the linear mappings have to be carefully planned, otherwise during training there doesn't occur any learning.

Here are some visual examples of the tests we conducted. For the given images their corresponding label has been learnt and predicted with highest probability.

We should note that the dataset available did not have separate labels for uppercase and lowercase letters, that is why, whether it is uppercase or lowercase we classify it with the label provided. For illustration purposes we used uppercase letters.

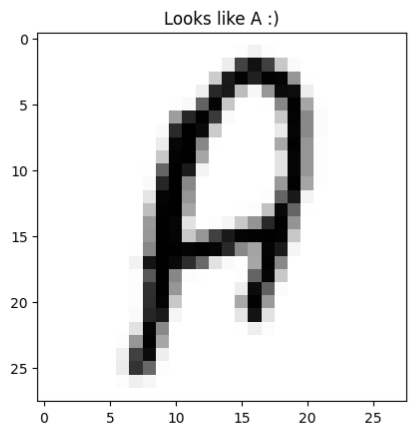

Figure 4.1: A has been correctly classified

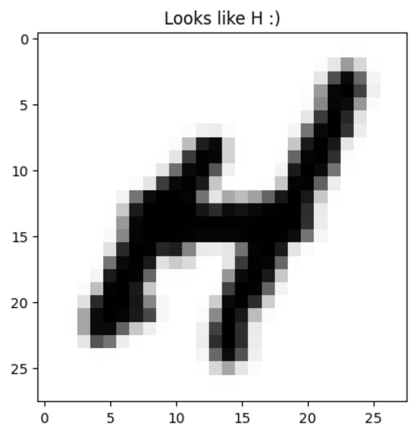

Figure 4.2: H has been correctly classified

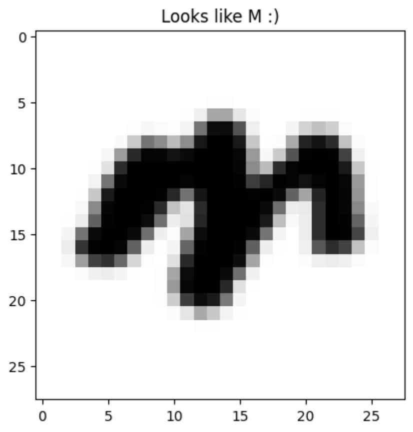

Figure 4.3: M has been correctly classified

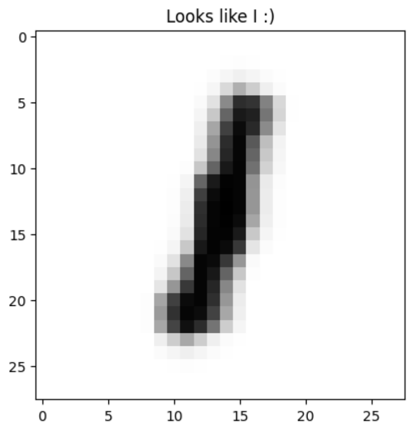

Figure 4.4: I has been correctly classified

Chapter 5

Conclusion

In summary, the project was interesting for us to implement and was fun to test.

This project presented an opportunity to apply the theoretical parts of multi layer perceptron on a real-life classification task, as well as review data science techniques during data preprocessing.

Bibliography

[BG94] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *Ieee Potentials*, 13(4):27–31, 1994.

[Bis06] Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:645–678, 2006.