INTRO TO AI COURSE 3160 PROJECT: HANDWRITTEN CHARACTER RECOGNITION USING ARTIFICIAL NEURAL NETWORK

WORK REPORT

Đinh Nguyễn Công Quý Quy.DNC214927@sis.hust.edu.vn

Vũ Tuấn Minh Minh.VT210597@sis.hust.edu.vn

Trần Đức Tuấn Kiên Kien.TDT214908@sis.hust.edu.vn

Bùi Minh Quang Quang.BM214925@sis.hust.edu.vn

DESCRIPTION OF THE PROBLEM

In various fields, there are demands for storing information in a computer storage from available data in handwritten documents. An approach is classifying each word then using combination of character segmentation and character recognition to store data directly in more convenient formats (doc ,docx, pdf, txt, …). In this project, we contribute to this approach with a method of applying Artificial Neural Network (ANN) to handwritten character recognition (HCR). It is worth mentioning that in this project, we mainly focus on analyzing our results in different settings. Therefore, instead of building ANN from scratch, we will utilize Python modules.

DETAILS OF SOLUTION

Our solution consists of a trained ANN that can take in an input image whose dimension is 28x28 pixels of 26 letters in the English alphabet and 10 digits (0-9). The ANNs will output a probability distribution and the letter or digit with the highest distribution will be the answer. This answer can be easily converted to document formats for further usage. This section only goes into details of the solution without explanation, further discussion will be in the following chapter.

1. **DATASET**

For this problem, we choose the EMNIST dataset, which is the best public dataset we can find so far for this problem. EMNIST (Extended MNIST) dataset is derived from NIST special database 19 which consists of the handwriting of 3,600 writers with more than 810,000 isolated character images. The difference between EMNIST and NIST special database 19 is that while images in the latter dataset have the dimension of 128x128 pixels, images in the former dataset have the dimension of 28x28 pixels.

EMNIST dataset is divided into 6 categories:

ByClass and ByMerge datasets

The full complement of the NIST Special Database 19 is available in the ByClass and ByMerge splits. These two datasets have the same image information but differ in the number of images in each class. Both datasets have an uneven number of images per class and there are more digits than letters. The number of letters roughly equates to the frequency of use in the English language.

Balanced dataset

The EMNIST Balanced dataset is meant to address the balance issues in the ByClass and ByMerge datasets. It is derived from the ByMerge dataset to reduce mis-classification errors due to capital and lowercase letters and also has an equal number of samples per class. This dataset is meant to be the most applicable.

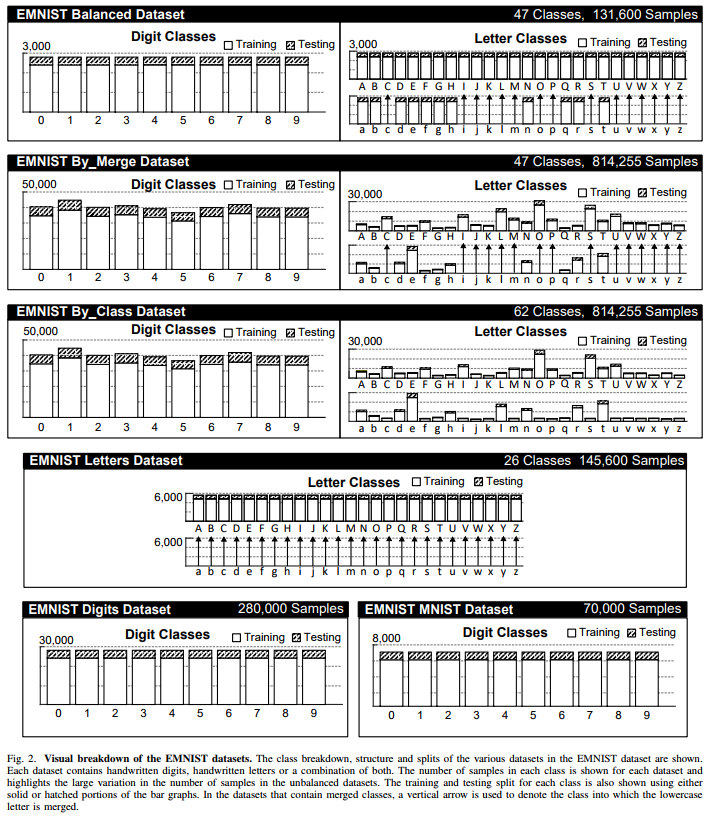
Letters dataset

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

Digits and MNIST datasets

The EMNIST Digits and EMNIST MNIST dataset provide balanced handwritten digit dataset directly compatible with the original MNIST dataset.

Visual breakdown of these datasets:



For more details of this dataset, please visit:

<https://www.kaggle.com/datasets/crawford/emnist>

<https://www.nist.gov/itl/products-and-services/emnist-dataset>

Since some characters have their uppercase form resembling their lowercase form, it is difficult to classify all 62 labels. Therefore, we only focus on ByMerge and Balanced datasets, both of which have 47 labels (15 characters have only one label instead of the usual two labels). Our final result is an ANN trained with the Balanced dataset.

1. **PROCESS OF PREPARING DATA AND TRAINING NETWORK.**
2. Data preparation

Balanced dataset is selected as we aim to train an ANN that can recognize all characters, without any bias caused by imbalanced data distribution.

Original EMNIST Balanced dataset has a total of 131,600 samples, 112,800 of which are devoted to training while the remaining 18,800 samples are used for testing. We have grouped data in these two groups and randomly redivided them again at 80:20 ratio (80% of data are used for training, the remaining 20% data are used for testing). In 80% data used for training , 10% of which are used for validation. In short, 72% of total data are used for training, 8% are used for validating, and 20% are used for testing

From experimental results, we also implement Min Max Scaler, further discussion will be in “discussion” sections.

In this problem, this dataset does not have any missing labels or data, so it is not necessary to perform any missing value handling.

Since all samples extracted from Balanced and ByMerge datasets are already in Numpy array format and already vectorized, no further data encoding is required.

1. TRAINING AND TESTING ANN.

All hyperparameters are selected based on the result of gridsearch, it is almost impossible to grid search all hyperparameters spontaneously so we have to perform gridsearch on each of hyperparameters individually. As a result, these hyperparameters do not constitute an optimal but a decent ANN with high overall accuracy of approximate 84%.

*Optimizer*

We choose Minibatch Gradient Descent with batch size = 32. The dataset has the size of 131,600 samples, 72% of which are training samples. Therefore, the number of training data for each epoch is 94,752 samples, 2,961 batches and 2,961 iterations per one epoch.

Initial learning rate is set at 0.01, with exponential decay rate of 0.9 every 10,000 steps

*Layers and units*

All hidden layers are dense and have Rectified Linear Unit (ReLU) activation function except for that last hidden layer which has Soft max activation function.

Input Layer: 784 units (equal the size of 28x28-pixel image)

First Hidden Layer: 256 units, 200,960 parameters (784x256 weights and 256 biases)

Second Hidden Layer: 256 units, 65,792 parameters (256x256 weights and 256 biases)

Last Hidden Layer (Output Layer): 47 units (47 classes), 12,079 parameters (47x256 weights and 47 biases).

In summary, there is one input layer and 3 hidden layers with a total of 278,831 parameters.

*Other hyperparameters*:

Dropout: We have experimented with Dropout on the first and second layers, but there is no significant improvement (less than 0.5% overall accuracy). Therefore, to keep the model simple, we decided to ignore Dropout.

Network Weight Initialization: there are many types of weight initialization. However, after having experimented with some types, we observed no significant improvement and to keep the simplicity of the model, we also ignored this hyperparameter.

Number of epochs: In all experiments with different optimizers, 30 to 40 epochs is a sufficient quantity of epochs for training while also preventing over-training.

Batch\_size: 32 samples. Although large batch-size makes less iterations per epoch, the ANN will require more epochs to learn. Batch size of 32 may not be optimal but it is good enough for this problem.

*Data preprocessing*:

Training and test data and labels are extracted from the module, they are then regrouped and divided again at 0.72:0.08:0.2 ratio for training, validation and testing respectively. Training, validation and test set are balanced (i.e. they have balance label distribution). All samples are preprocessed using Min Max Scaler.

*Building ANN*:

ANN model is built using Keras module. The input layer is built together with first hidden layers. The second hidden and output layer are built sequentially.

This built ANN structure is then compiled with mentioned hyperparameters to get ready for the training

*Training and testing:*

In one epoch, the ANN is trained with the training set. After one training minibatch is forwarded, back propagation occurs in which ANN adjusts parameters by calculating the derivative of loss function. As our training dataset contains 94,572 samples, 2,961 minibatches, there are 2,961 of this forward and back propagation process happen in one epoch.

After the ANN has completed the training process, validation process occurs in which the model is evaluated using the validation set. One epoch finishes once both training and validation processes finish. After all epoches have finished, the testing process begins where the model is evaluated using the test set.

SOFTWARE AND PACKAGE UTILIZATION

Our solution does not utilize any existing ANN model. Instead, we utilize several Python modules for data preprocessing, ANN model and result demonstration, these modules are listed below.

1. **EMNIST MODULE (DATASET MODULE)**

This module is used for importing dataset and is very simple to use. Two methods used are extracting samples, labels from Balanced training dataset and extracting samples, labels from Balanced test dataset. For more details of this module, please visit: https://pypi.org/project/emnist/

1. **SCIKIT-LEARN MODULE (DATA PREPROCESSING)**

This module assists us greatly in data preprocessing. Main methods used are splitting dataset into training and test dataset and classification report which helps getting accuracy for every label. For more details of this module, please visit: https://scikit-learn.org/stable/getting\_started.html

1. **KERAS (ANN)**

This module is used for building ANN, it offers users an easy way to build and adjust ANN. For more details of this module, please visit: <https://keras.io/getting_started/>

1. **OTHER MODULE**

**Numpy:** a Python module that is good for working with matrices . For more details of this module, please visit: <https://numpy.org/>

**Matplotlib, Seaborn:** a useful Python module for drawing graph resulst.For more details of this module, please visit: <https://matplotlib.org/>, <https://seaborn.pydata.org/>

EXPERIMENTAL RESULTS, ANALYSIS AND FUTURE WORK.

1. **ANN ACCURACY**

The trained ANN achieve 84% overall accuracy after 40 epochs with hyperparameter are configured as mentioned in previous section. However, there are significant differences among different labels. The confusion matrix below gives some insight into our result.

The row of the matrix corresponds to what the machine predicted and the column of the matrix corresponds to the true labels of the data. The main diagonal represents all the data that have been correctly classified labels, so the darker the main diagonal is, the more accurate the ANN is.

Chart, scatter chart

Description automatically generated

From classification report module of Sklearn, we observe that 10 labels have very poor accuracy metrics (at around 60% for precision, recall and f-1 score) are (0 - O), (1 - I - L), (9 - g - q), (F - f) while there are around 20 labels that constantly have the accuracy metric at around 90%.

1. **REMOVING SOME LABELS**

We remove all data and labels whose input resembles other labels (0, I, L, g, q, f) and the ANN achieves an expected significant improvement in accuracy, from 84% to around 92% in 40 epochs and may even reach higher accuracy. All the labels have their accuracy metrics at around 90%, 4 labels reach 0.97 f-1 score.

A picture containing chart

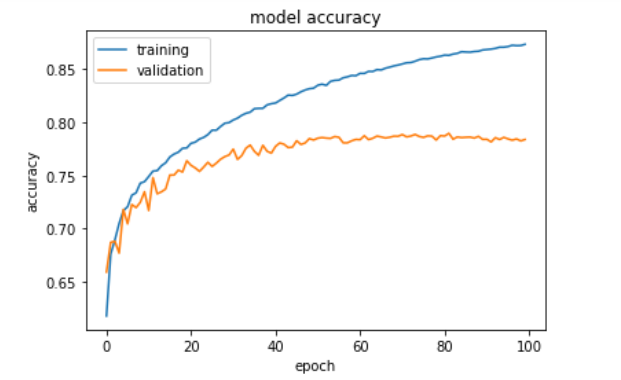
Description automatically generated A picture containing text, grate

Description automatically generated

The left figure is the 47x47 confusion matrix of ANN trained with the original dataset, the right figure is the confusion matrix of ANN trained with the new dataset where some labels are removed. While in the left figure there are some colored dots that are not on the main diagonal, in the right figure, all colored dots are on the main diagonal. From this experiment, we conclude that the ANN is sufficiently good to classify the majority of the labels, however, to classify 10 hard labels ((0 - O), (1 - I - L), (9 - g - q), (F - f)), some improvement must be made.

1. **DATA MIN MAX SCALER**

Two following graphs demonstrate ANN trained with unscaled data (first figure) and scaled data using Min Max Scaler (second figure).



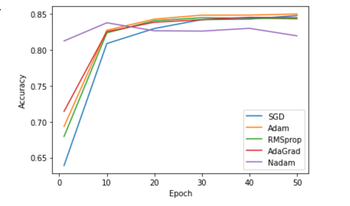
Chart

Description automatically generated

ANN achieves higher accuracy when data is already scaled (84% compared to 78%). Some other techniques such as standardization also help ANN achieve 84% accuracy, but to keep the simplicity of the model, we choose the simple Min Max Scaler technique. In conclusion, it is better to have the data feature scaled to increase the effectiveness of the back propagation process and improve accuracy.

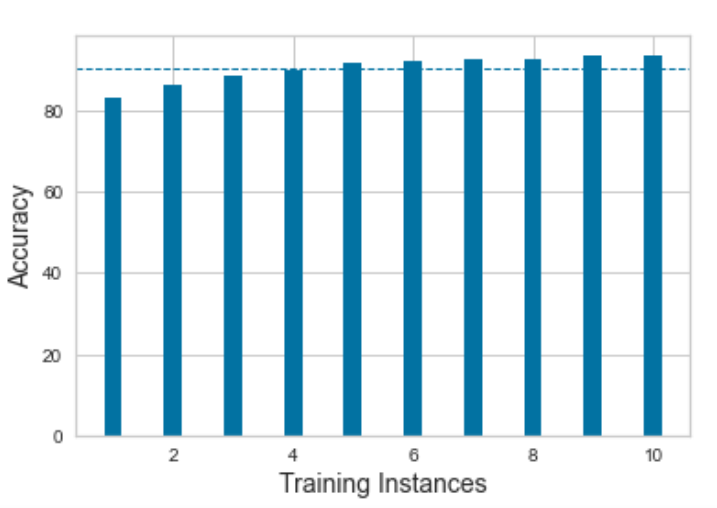
1. **OPTIMIZER EXPERIMENTS**

All Optimizers are tested using the same ANN settings that were mentioned in the previous chapter. After 50 epochs of training, except for Nadam which has approx. 82% accuracy, 4 other optimizers reach approx. 84% accuracy. We have also used girdsearch for different hyperparameters used for ANN such layers, units, weight initialization but there is no significant improvement.



1. **K-FOLD CROSS VALIDATION**

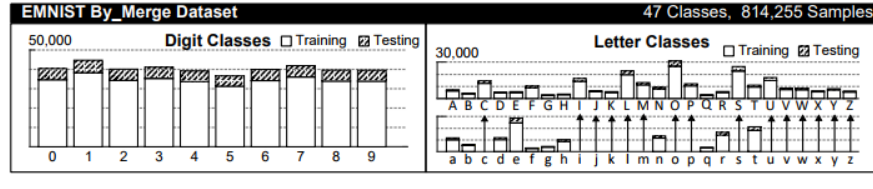
This technique is used in our project to make best use of the dataset , i.e, to give a thorough evaluation of how well the model would perform in different training and validation splits of data. Finally, we calculate the average accuracy of all folds to assess the overall performance of the model. In this case, we choose k = 10 which is a divisor of the size of the dataset so that the size of the groups in the sample would be sufficiently large and stratified.



The graph above shows that while the average accuracy is quite high, there are some splits for which our fitted classifier performs significantly less well. Although this procedure may increase the running cost by k times, it could prevent our model from overfitting. As there is no significant improvement, we do not inclue this in our final ANN

1. **BALANCED AND BYMERGE DATASET.**

ByMerge dataset has a considerably large numbers of samples, however, it has imbalance data disbituion. We have experimented with this dataset and achieved 88%, but this higher accuracy mainly results from imbalance data distribution.



5 hard labels are 9, g, q, and F, f have small ratio which makes the easier label contribute to more of the overall accuracy

1. **UNSOLVED PROBLEM AND FUTURE WORK.**

Hard labels:

10 hard labels (0 - O), (1 - I - L), (9 - g - q), (F - f) prove challenges to our ANN. In recent years, fine grained classification has emerged in Deep Learning as a potential technique to hard classification problem techniques (such as species of birds, flowers, or animals classification). To use this image must be in a larger dimension, therefore, it is suggested that readers may use NIST Special 19 Database). Although larger input image size leads to larger number of parameters, it is a promising solution to achieve higher accuracy.

Also more samples would probably increase overall accuracy. When experimenting with ByMerge dataset, due to the significantly increased number of samples in label "1" and "l", all accuracy metrics for these two labels generally improve by 8% consistently.