Team Orange (Team 5)

Final Project Progress Report

Ruoshui Li, Yi Feng, Jennie Sun, Shangwen Yan, Michael Tang

Project Title

Text Detection And Extraction Using OCR (Optical Character Recognition)

Goals of the Project

We plan to implement a Convolutional Neural Network (CNN) algorithm and apply image recognition and classification for OCR for this project by following a machine learning pipeline that includes model training, image preprocessing, text detection, character segmentation, and character classification. Currently, we have preprocessed image data and trained a few classifiers to predict the class characters 0-9, a-z, and A-Z using different machine learning approaches and compared model performance.

The dataset that we used for model exploring is the EMNIST dataset, which is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28 x 28 pixel image format and dataset structure that directly matches the MNIST dataset. There are six different splits provided in this dataset, including ByClass, ByMerge, Balanced, Letters, Digits, and MNIST. Specifically, we trained five different models using the Balanced subset of the EMINIST data that includes 131,600 characters, which further splits into 112,800 characters for the train set, and 18,000 characters for the test set. This dataset has 47 balanced classes in total. We decided to train the EMNIST Balanced dataset because it is meant to be the most applicable, as it addresses 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.

At the current stage, we have evaluated our model performance by comparing the accuracy scores and computational speed across five different models. As we continue to work on the project, we will adopt other model evaluation metrics by either counting how many characters that were detected correctly (character-level accuracy) or how many words that were recognized correctly (word-level accuracy).

Initial Results

We trained 3 traditional machine learning models and 2 CNN models on the EMNIST Balanced dataset. Below summarizes the training process and results from those models.

1. Support Vector Machine

Support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. In order to use SVM for a multi-class problem, we use the SVC function from scikit-learn svm model. After that, we train the model on the EMNIST balanced training dataset and use default values for all parameters. While the overall accuracy (which is defined as the number correctly predicted observations divided by total observations, for all classes) on the test data is 0.84, the model performance varies across different classes. More specifically, the model has the best performance on classifying digits. When it comes to letters, our model has lower performance in general. It is also worth noting that since “o” and “0” look very similar, our model has particularly low precision on classifying these two classes. Similarly, classes “1”, “I” and “q”, “9” have the same problem.

2. Multi-Class Logistic Regression

Logistic function generally deals only with binary classes, but when there are multiple labels in the dataset that need to be handled, an extended multinomial logistic regression is needed. Instead of the sigmoid function used in generalized logistic regression, multinomial logistic regression uses a softmax function to compute the probability that one of the observations belongs to a specific class. The loss function, similar to generalized logistic regression, applies the concept of cross-entropy, which basically takes the calculated probabilities from softmax function and the known labels to calculate the loss.

Data preprocessing includes: 1). Standardizing the image data by dividing pixel values by 255 (maximum pixel value) to make sure each pixel has a value between 0 and 1. This reduces the varying degree of features’ magnitude and range. 2). Since the output of the softmax function is a vector of probabilities of each class label, for instance in our case, the softmax function for each observation provides a 47 by 1 matrix, the known label vector corresponding to each observation is converted to the one-hot encoded vector for calculating the loss function.

After loading the EMNIST dataset and preprocessing the data, we deployed the Tensorflow package to build the multinomial classifier. The key parameter needs to be tuned here is the learning rate of the model. To figure out the best possible learning rate for achieving maximum classification accuracy, we randomly assigned 10 learning rates value between 0.00001 and 10, and calculated the overall accuracy and loss function value of the training dataset. For each run, there are 200 training epochs. The results showed that training dataset accuracy reaches the maximum when the learning rate hits 0.001307, which we then choose to be our final learning rate. The final model accuracy on the test dataset is 0.75.

3. Multilayer Perceptrons (MLP)

A multilayer perceptrons model was proposed for classifying hand-written digits and letters in the EMNIST dataset, as this type of model is often applied to supervised learning problems and is helpful in distinguishing data that are not linearly separable. We trained our multilayer perceptrons algorithm on a set of input-out pairs, which learned to model the correlation between those inputs and outputs. We started by loading the datasets from the tensorflow package and extracted the predefined train and test data in the Balanced dataset. During the data preprocessing stage, we first flattened the data by reshaping the training data from (112800, 28, 28) to (112800, 784), and performing the same steps for the testing data. The data were then rescaled using the same approach as described in the logistic regression above.

During model training, we used two dense layers, meaning that every node from the previous layer is connected to each node in the current layer. We used 256 nodes in the first layer, added a dropout with rate 0.2, and used 64 nodes in the second layer. We applied ‘ReLU’ as our activation function in both layers. In the output layer, we applied softmax as our activation function to normalized output of the network to a probability distribution over predicted output classes. We then compiled our model by defining a set of hyperparameters and applied early stopping to our algorithm to prevent overfit. The model was trained on a local device as the process only took a few minutes to complete. The model had 220,528 trainable parameters in total. With 50 epochs and a batch size of 128, the MLP model achieved a test accuracy of 0.85.

4. LeNet-5

LeNet-5 was a simple Convolutional Neural Network-based classifier that was initially proposed by Yann LeCun in 1989 specifically for recognizing hand-written digits. Despite being one of the first published Neural Networks for tackling computer vision jobs, this model achieved a remarkable result on the MNIST dataset, and had demonstrated superior performance over the Support Vector Machines, which was back then considered the state-of-the-art classifier.

LeNet consisted of 2 main components - a set of convolutional layers and a set of fully connected dense layers. More specifically, the original network accepted an input image of size (32, 32). The EMNIST data are of shape (28, 28), which is left unchanged before feeding into the network. The work starts by convolving 6 kernels of size (5, 5) across the input image to capture low level details such as edges and lines. For the non-linear activation function, we used 'ReLU' instead of the originally proposed 'Sigmoid/Tanh' function for better performance and to avoid the vanishing gradient problem. Next, an average pooling layer with size (2, 2) is then applied to condense information. (The original LeNet architecture used a stride of 2 for average pooling). The convolutional step is then repeated with 16 kernels of size (5, 5, 6) to further extract higher level information from the original image, and followed by another average pool layer. The output shape of the first component of our network is (4, 4, 16). The second component comprises 2 fully connected layers of size 120 and 84 respectively. And the final layer is a single neuron with a softmax activation function to output probabilities for 47 classes.

Our LeNet model had a total of 47571 trainable parameters, which is significantly less than most modern architectures. This resulted in a much faster training time (~30 seconds/epoch on a local dual-core system without any GPU performance boosting). More concretely, we used Adam as our training optimizer, and Categorical Crossentropy and Categorical Accuracy as the loss function and metrics , respectively. The training was performed for over 50 epochs, and we achieved an accuracy of 0.86 on the training set, and 0.88 on the test set.

5. AlexNet

As one of the pioneer models in the deep learning domain, AlexNet was developed by computer scientist Alex Krizhevsky and his team in 2012. Compared to previous convolutional neural network models, AlexNet has several distinguishing features: 1) used Rectified Linear Units as the activation function 2) overlapping max pooling layers 3) used data augmentation techniques and dropout to reduce overfitting.

The input shape is (28, 28), which is convolved 96 times using a (11, 11) filter. The output from this layer is then convolved 259 times using a (5, 5) filter , after which it goes through a max pool layer. The next layer uses a (3, 3) filter 384 times and then feeds it to another max pool layer. Then the output is passed to a (3, 3) filter 384 times and 256 times, and sequentially goes through the last max pool layer. Convolved results are then flattened and fed to three fully connected layers with 4096, 4096, and 1000 nodes accordingly. Lastly, softmax is applied on the output layer to predict the probability of sample belonging to each of the 47 classes.

The architecture of the AlexNet includes 5 convolutional layers, 3 max pool layers, 3 fully connected layers, and 1 output/softmax layer. In total, the model has 25, 723, 229 trainable parameters. Using a learning rate of 0.001 and bath size of 320, after 10 epochs, the model achieved a training accuracy of 0.86 and validation accuracy of 0.84. The accuracy on the test data was 0.84.

Summary Table of Test Accuracy by Model

Model	SVM	Logistic Regression	MLP	LeNet	AlexNet
Accuracy	0.84	0.75	0.85	0.88	0.84

Major Challenges

As we have dived deeper into this domain, several challenges have arised. First of all, different models have shown to have highly varied training time and complexity. This is seen in the difference between the AlexNet and LeNet. Convolutional Neural Network models, although show better model performance, can have thousands of more trainable parameters than linear models, and can thus require a significant amount of training time and hyperparameter tuning. In this case, we will consider using the model that not only has high performance, but also with superior computation efficiency. Besides, running models using GCP or AWS will also be feasible solutions.

The second challenge that we have faced is cross-domain comparison. By far, we have trained and tested all models on the same dataset -- EMNIST. Given our interests in testing the generalizability and applicability of each model in different domains, we will need to carefully design and choose datasets such that the results are meaningful for comparison.

The third challenge is metrics incongruency. We find out that it is harder than expected to use the same reasonable, standard metrics to evaluate performances. Additionally, current evaluations were performed on the test set predefined by the MNIST. As previously proposed, we also plan to carry out train-test split on the training set to further fine tune model hyperparameters and conduct cross data performance evaluation.

References

EMNIST: an extension of MNIST to handwritten letters
https://arxiv.org/pdf/1702.05373v1.pdf

AlexNet:
https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

C-Support Vector Classification:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC