Modified MNIST Digit Recognition

Elio Abi Younes, Samuel Grégoire, Laith Mubaslat

COMP 551, McGill University

March 18, 2019

Abstract

This work attempts to design and compare the performance of several supervised learning classification algorithms on a modified version of the well known MNIST dataset of handwritten digits. After preprocessing, the models were trained and tested on the dataset. Models were then evaluated based on their respective accuracy over a left out validation set. Ensemble CNN has shown a significant improvement over the other approaches reporting a test accuracy of 0.96333 on Kaggle.

1 Introduction

Handwritten digit recognition is an important task that comes under the umbrella of optical character recognition and has shown its practical applications in many areas including financial implications [10]. This project explores preprocessing of handwritten digits, including segmentation, normalization and data augmentation which were found to significantly improve the results. Additionally, the performance of different classifiers was evaluated, namely ANN, CNN, pre-trained VGG and Ensemble CNN on a modified version of the MNIST dataset. An 80/20 training, validation split was used to evaluate the accuracy of the different models. An 8-layer CNN was found to perform best as a single model. Stacking 15 such models in an ensemble produced the best performing classifier with an accuracy of 96.33% and a considerable running time.

2 Related work

Handwritten digit recognition is among the most popular and useful work in machine learning. Researchers have studied numerous algorithms to improve this task. In particular, Niu and Suen have proposed a very simple, yet modified convolution neural network model called hybrid CNN–SVM model for handwritten digit recognition. This model is based on two tasks, that is, retrieving features based on the CNN architecture and recognizing the unknown pattern based on the SVM recognizer. The model was tested on the MNIST dataset and has shown the best recognition results with an error rate of only 0.19% when compared to other methods including CNN (error rate 0.40%) and KNN (error rate 0.52%) [10].

In addition, Lauer, Suen and Bloch also explored the MNIST dataset, designed and compared three different algorithms. The first suggested model is LeNet5, a convolutional neural network, composed of

seven layers, namely three convolutional layers, two subsampling layers, one fully connected layer and the output layer. The second model is a simple SVM trained with linear kernel and a polynomial kernel of degree 5. The last proposed approach is based on the so called trainable feature extractor (TFE) which consists of replacing the last two LeNet5 layers by single linear output layer as follows: for a 10-class problem, the 120 outputs of the last convolutional layer are trained to be linearly separable and can be used as features for any other classifier. The designed algorithm is labelled TFE-SVM since it's composed of the TFE connected to a multitude of binary SVMs. The results show that TFE-SVM method outperforms both LeNet5 and SVM [8].

3 Dataset and Setup

3.1 Dataset

The dataset is a modified variation of the well known MNIST hand-written digit dataset. It consists of 40,000 labeled and 10,000 unlabeled images. These images are grey-scale of dimension 64×64. Contrary to the standard MNIST dataset, each image contains more than one digit which are of varying sizes. The task at hand is to classify the digit occupying the largest bounding box in an image.

3.2 Preprocessing

Extracting the digit with largest bounding box would simplify the classifier's task into that of a MNIST hand written digit recognition. Accordingly, two methods were utilized in accomplishing that task. Both involved first setting all image pixel values below a certain threshold to zero. Following that the indices for islands of connected components of similar intensity were evaluated and then compared for their respective bounding box size. The first method involved manually transforming each image into a graph and then utilizing DFS [16] in computing these islands while the second involved using the standard $connectedcomponentswithstats$ function of the *OpenCV* library [6]. The latter was used in the implementation of the proposed approaches considering it is significantly faster.

Next, several preprocessing techniques standard to image recognition tasks were used to improve the performance of the different classifiers. The grey scale pixel values ranging between 0 and 255 were scaled to reduce the effect of illumination's differences. The pixel values were normalized to the range 0 and 1 by dividing each value by the maximum of 255 [7].

To fit a multi-layer perceptron model the image size was reshaped into a vector of pixels. Thus, the dimensionality of each image sample vector is 64×64 pixel input values. Finally, a one hot encoding of the class values was used to transform the vector of class integers (0 to 9) into a binary matrix. The final resulting dataset was divided using an 80/20 training, validation split.

4 Proposed Approach

After preprocessing the data, four machine learning classification techniques were fitted to train the data and recognize the handwritten digits. Models were then compared to select the one with the best performance.

These models were built using the *Keras* package [2]. Parameter assignment in the following models was made based on empirical testing and the results of the study in [4].

4.1 ANN

A simple neural network model with one hidden layer is first designed. A rectifier (RELU) activation function is used for the neurons in the hidden layer and a softmax activation function for the output layer. The classification is based on the maximum probability values of the outputs. To train the model, a logarithmic loss is used which calculates the error rate between the predicted value and the original value.To learn the weights, ADAM optimizer is used since it has shown to have an improvement over Stochastic Gradient Descent [17].

4.2 5-Layer CNN

A convolutional neural network is a feed-forward network used to extract features from the raw image in its first layers and classify the pattern with its last layers. It is trained in a similar way to a standard neural network by back propagation but is of a more complicated nature [8]. The following is the architecture of the 5-Layer CNN composed of 5 layers including the input and output layers. The input layer expects images of size 64×64 with 1 channel (i.e. a greyscale image). The next layer is a convolutional layer with stride 2 composed of 32 filters with a 3×3 kernel size. A rectified linear unit (ReLU) is used as the activation function. It's the most used activation function since it reduces training time and prevents the problem of vanishing gradients [13]. Next is a subsampling layer using the MaxPooling technique with 6 feature maps and a 2×2 kernel. MaxPooling layer is usually used to enable the model to make assumptions about the features and to reduce over-fitting and the number of parameters to learn which as a result reduces the training time [7]. The convolutional layers are then flattened so that they become the input to the next 128 units Dense layer which classify the features learned by the model. In order to control overfitting, the Dropout method is used with 40% neurons disable rate [7].The last fully connected layer is composed of a Dense layer with 10 neurons equal to the number of classes (10 digits). The Softmax Activation is used to predict the output based on maximum probabilities. As before, the model is trained using logarithmic loss and the ADAM optimizer.

4.3 8-Layer CNN

A larger CNN architecture is defined, expanding on the earlier network, with additional convolutional, pooling and fully connected layers. The two additional convolutional layers are used with stride 2, 64 feature maps of size 5×5 and 3×3 respectively while the sub-sampling layer is used with 16 feature maps and a 3×3 kernel. Here, BatchNormalization is used to normalize the matrix after the convolutional layer in order to maintain the same scale of each dimension thus reducing the training time [7]. All the convolutional layers are also flattened. The remaining dropout and output layers are used as before and the model is trained similar to the 5-Layer CNN.

4.4 8-Layer CNN with Data Augmentation

Expanding the handwritten digit dataset by altering the training data is a technique called Data Augmentation and has been proved [9] to improve the learning algorithms. The idea used here is to randomly rotate the digits up to 10 degrees, zooming-in as well as shifting the images vertically and horizontally by 0.1 and flipping them in order to reproduce the variations occurring in the handwriting digits. The complex CNN model described in 4.3 is used to train the augmented data thus creating a robust model.

4.5 Transfer-Learning (VGG Pre-trained Model)

Transfer learning pertains to the utilization of pre-trained models in building newer models [11]. The base test model consisted of a 16 Layer pre-trained VGG convolutional neural network connected to two dense layers on the output side of sizes 256 and 10 respectively [15]. The input size for the VGG network was altered to fit our dataset while its output layer was removed. Several iterations of this model were attempted under a batch size of 160. Only one epoch was attempted due to the high computational demand of this approach. For each iteration of the model the number of trainable parameters was varied. That was achieved by deciding either on which VGG layers were to be removed, just freezing their respective weights or varying the depth and width of layers connected to the VGG output side. Non of the models attempted produced an accuracy tempting enough to justify the computational (and time) cost of training it beyond the count of 1 epoch.

4.6 Ensemble CNN

To obtain better classifier performance, 15 models were used in a bagging ensemble, each being the 8-Layer CNN. The models were then trained independently on separate datasets. These datasets were obtained by randomly sampling the training set with replacement[1]. We further utilized the difference in initial weights as a difference in model structure and designed a different ensemble by stacking the 15 models and training them on the same training set[14]. The final output for each of the two ensembles is then obtained by averaging the outputs of the 15 networks trained before.

5 Results

All the approaches proposed previously were fitted and evaluated over a different number of epochs and batch sizes. A batch size of 64 was empirically found to best accommodate the available computational resources and desired gain in accuracy. We found that no significant improvement was obtained by training the models with an epoch count beyond 10 for that batch size [12]. In table 1 the accuracy of every proposed model on the validation and Kaggle test sets is reported. In general, ANN and VGG did not perform well on this dataset as compared to other CNN models. Even though VGG showed a very high validation accuracy, it ended up being the least performing model on kaggle test set. The performance of the VGG model could be attributed to lack of training (i.e. high bias). However, that is contradictory to the relatively high validation and training accuracy contrary to the low test accuracy which indicate overfitting (i.e. high variance). The 5 and 8 Layer CNNs performed significantly better and gave similar results. We found

that data augmentation positively influenced the performance of our 8-Layer CNN. Additionally, we tried to normalize the pixel values to the range (0,0.99) by dividing each value by 257 which showed a better performance increasing the test accuracy to 0.954. The two variations of the ensemble model (stacking and bagging) showed a significant improvement where they scored on Kaggle 0.9633 and 0.9620 respectively. Clearly the best performing model was the stacking ensemble CNN.

Regarding the runtime of our models, we noticed that there is a tradeoff between model accuracy and runtime. The depth and complexity of the model, whereas could positively influence the accuracy, would drastically increase the run time of a neural feed-forward network [5]. Following is the average runtime of our models by increasing order: ANN < 5-Layer CNN < 8-Layer CNN < Ensemble CNN < VGG.

Classifier	Validation Accuracy	Test Accuracy
ANN	0.8969	-
5-Layer CNN	0.9140	0.92433
8-Layer CNN	0.9325	0.93633
8-Layer CNN with Data Augmentation	0.9435	0.94600
8-Layer CNN with Data Augmentation (Norm /257)	0.9570	0.95400
Stacking Ensemble CNN	0.95900	0.96333
VGG	0.9714	0.88033

Table 1: Model accuracy on the validation and test set

6 Discussion and Conclusion

Handwritten digit recognition is an active research area. This project presented several classifiers commonly used in handwriting digits recognition and has been tested on a modified version of the well known MNIST dataset. The stacking ensemble CNN method outperformed all the other approaches on the test set. It should be noted that some handwritten digits are ambiguous even to the human reader. That could in part explain the error obtained during our experiment. However, the error can be better explained due to our implementation having not explored the entirety of machine learning algorithms and having not pushed our implementation to its limit (i.e. acquiring the most optimal bias-variance trade off for a certain architecture). Additionally, bagging produces a lower test accuracy when compared to stacking. That, combined with the low epoch count, confirms the previous hypothesis that our model is one of high bias (i.e. underfit)[18]. Thus, training it further should yield more positive results especially if used in a boosting ensemble [9].

For future work, we would like to consider the problem of feature extraction and test the performance of the TFE based on LeNet5 approach introduced in [8]. We would also like to train the hybrid CNN-SVM model mentioned previously in the related work section and introduced by Niu and Suen [10]. An important challenge that we faced in our work and would like to further investigate is the model accuracy and runtime in deep learning. A new learning rate strategy introduced in [5] proposes reducing the mini-batch size per learner which can effectively improve runtime performance and achieve good model accuracy.

7 Statement of Contributions

Laith and Elio were equally responsible for the data preprocessing, classifiers design and algorithms, and report writing. Samuel worked on part of the preprocessing and part of the data augmentation.

8 References

[1] Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.

[2] Chollet, Francois. Keras, 2015, keras.io/?fbclid=IwAR2GtEFV7xMS-tFjV0dASFGlKDElkm-bqihwlam$_r$0t7svj.

[3] Deng, Li. "The MNIST database of handwritten digit images for machine learning research [best of the web]." IEEE Signal Processing Magazine 29.6 (2012): 141-142.

[4] Deotte, Chris. "How to Choose CNN Architecture MNIST." Kaggle, www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist.

[5] Gupta, Suyog, Wei Zhang, and Fei Wang. "Model accuracy and runtime tradeoff in distributed deep learning: A systematic study." 2016 IEEE 16th International Conference on Data Mining (ICDM).

[6] Howse, Joseph. OpenCV computer vision with python. Packt Publishing Ltd, 2013.

[7] Katariya, Yash. Applying Convolutional Neural Network on the MNIST Dataset. yashk2810.github.io/Applying-Convolutional-Neural-Network-on-the-MNIST-dataset/.

[8] Lauer, Fabien, Ching Y. Suen, and Gérard Bloch. "A trainable feature extractor for handwritten digit recognition." Pattern Recognition 40.6 (2007): 1816-1824.

[9] LeCun, Yann, et al. "Learning algorithms for classification: A comparison on handwritten digit recognition." Neural networks: the statistical mechanics perspective 261 (1995): 276.

[10] Niu, Xiao-Xiao, and Ching Y. Suen. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits." Pattern Recognition 45.4 (2012): 1318-1325.

[11] Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering 22.10 (2010): 1345-1359.

[12] Radiuk, Pavlo M. "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets." Information Technology and Management Science 20.1 (2017): 20-24.

[13] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).

[14] Rokach, Lior. "Ensemble-based classifiers." Artificial Intelligence Review 33.1-2 (2010): 1-39.

[15] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to Algorithms, Second Edition". MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp. 540–549.

[17] Valdenegro-Toro, Matias. "Best practices in convolutional networks for forward-looking sonar image recognition." OCEANS 2017-Aberdeen. IEEE, 2017.

[18] Wolpert, David H., and William G. Macready. "An efficient method to estimate bagging's generalization error." Machine Learning 35.1 (1999): 41-55.