Modified MNIST

Milad Ghanbari

Abstract

In this project, the problem of finding the largest digit in an image is tackled using several machine learning approaches. The provided dataset is a modified version of the MNIST dataset. There are multiple digits in each image, and these digits have been randomly rotated, resized, and stuck onto a randomly generated background. In the first attempt, the images were preprocessed in order to extract the largest digit from the image and to transform the modified MNIST problem to the typical MNIST problem. This report discusses and compares various approaches of image classification that fall within the machine learning paradigm such as Logistic Regression (LR), Support Vector Machines (SVM), Convolutional Neural Networks (CNNs), and Ensemble Method. The optimum values for each of the hyperparameters related to the classifiers were obtained by using a grid search approach. Furthermore, some very deep models such as VggNet, ResNet, and Xception were experimented on the provided data. K-fold cross-validation method was performed on the training data to achieve a reliable and robust model. In addition, dropout and data augmentation methods were utilized to prevent deep learning approaches from overfitting. Finally, through performing extensive experiments on the mentioned classifiers, the best model resulted in the highest performance was the ensemble of 10 Xception models with 98.433 % accuracy on the test data.

Keywords: *Deep Neural Network, Xception, Modified MNIST, Data Augmentation*

1 Introduction

The automatic classification of text in images has witnessed a huge interest in the last two decades, due to the increased availability of images in digital format and the ensuing needs of processing them to speed up information recognition. Text data presented in images play a significant role in the retrieval system as they contain plenty of useful information for automatic annotation, indexing, and structuring of images. One of the interesting topics in this area is the classification of handwritten digits from visual data. A standard entry point into this field is the classification of handwritten digits using the MNIST dataset [1, 2].

In this project, we aim to find the largest digit in the modified version of the MNIST dataset where each image containing multiple digits originating from the MNIST dataset. These digits have been randomly rotated, resized, and stuck onto a randomly generated background. As a benchmark, a dataset was provided which contains 40k labeled images for training and 10k unlabeled images for testing. In this project, different machine learning approaches were trained and evaluated on the 40k images in the training set, while the other 10k were utilized to obtain the accuracy of our final proposed model on the unseen examples. In order to propose a robust model with acceptable generalization to the unseen data, we used the K-fold cross-validation technique on the training data. In our first method, we took some preprocessing steps to clean the raw images before classification. For this end, some image processing algorithms such as background removal, finding image contours, shifting and resizing were performed respectively in the preprocessing phase.

Various approaches to image classification that fall within the machine learning paradigm were discussed and compared in this report. Consequently, the accuracy and performance of some machine learning based classification techniques such as Logistic Regression (LR), Support Vector Machines (SVM), Convolutional Neural Networks (CNNs), and Ensemble Method were evaluated. Besides the mentioned classifiers, some existing very deep models such as VggNet, ResNet and Xception were experimented as well. Among all the classifiers, the ensemble of 10 Xception models resulted in the highest accuracy on the training and the test set (98.43 %). It has to mention that one of the interesting findings was the fact that for deep learning approaches, higher accuracy can be achieved without preprocessing images (extracting the largest digit and removing other digits in each image). The highest accuracy of 96 % and 98.4 % was obtained for deep learning approach with and without preprocessing of images respectively. The reason is presumably due to the information missing during the preprocessing step. Furthermore, some methods such as dropout and data augmentation were utilized to prevent deep learning approaches from overfitting.

2 Related Work

In recent years, there has been a drastic growth in the number of studies related to visual recognition including digit recognition on MNIST dataset. Here, we briefly present some of the important ones which used Convolutional Neural Networks as their models and present some recent works utilized Xception.

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of Convolutional Neural Networks (CNNs) [3]. A CNN is a feed-forward network that can extract topological properties from an image. Like most other neural networks, they can be trained with a version of gradient descent learning e.g., back-propagation of error. During the last years, this paradigm showed a series of fascinating results. Especially in computer vision tasks, the CNNs became successful enough to be the basis for real-world suitable applications. For instance, deep convolutional neural networks were proposed to classify the 1.2 million high-resolution images (ImageNet dataset) into 1000 different classes. They obtained top-1 and top-5 error rates of 37.5% and 17.0% on the test data [4]. Regarding investigating MNIST dataset, the literature [5] is shown that properly trained wide and deep Deep Neural Networks (DNNs) can outperform all existed methods and is demonstrated that unsupervised initialization/pretraining is not necessary. Moreover, combining several DNN columns into a Multi-column DNN (MCDNN) further decreases the error rate by 30-40%. This study not only was the first to achieve near-human performance on the very competitive MNIST handwriting benchmark but also outperformed a traffic sign recognition human's benchmark by the factor of two. In [6], they focused on improving recognition rates using committees of neural networks, and they reported experiments using CNNs trained on MNIST as well as on a more challenging NIST SD 19 database. They reported $0.27\% \pm 0.02$ for a committee of seven deep CNNs trained on graphics cards, narrowing the gap to human performance on the MNIST handwriting recognition benchmark.

In [7], a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolutions is introduced and named Xception. It is required to mention that the Xception architecture has the same number of parameters as Inception V3; therefore, the performance gains are not due to increased capacity but rather to more efficient use of model parameters. Some of the most recent works which used this method are [8] and [9]. In [8], the aim of the research is to use deep learning algorithm combined with transfer learning to improve the generalization ability and the accuracy of the license plate recognition system. The Xception network is utilized to train license plate data with weights randomly initialized. Moreover, Transfer Xception model for image classification with weights trained on ImageNet to this task and train license plate data again, and then the accuracy of these two models were compared with different deep learning networks. In [9], a three Xception networks based Bangla handwritten digit classification scheme was presented and was evaluated on a hidden test set where it showed promising performance of 96.69% accuracy, F1 score of 97.14%.

3 Dataset and Setup

The MNIST dataset is a database of handwritten digits that is widely used in the field of machine learning to familiarize with pattern recognition and learning techniques [10]. The dataset provided for this project is a modified version of the MNIST dataset [11]. In this modified dataset, the digits rotated in a random orientation, the images contain more than one digit and every image has some background noise in it. The given dataset consists of 40k grayscale images of size 64×64 for training and 10k grayscale images with the same size as the test data. Labels included for the training dataset indicating the largest digit in each image.

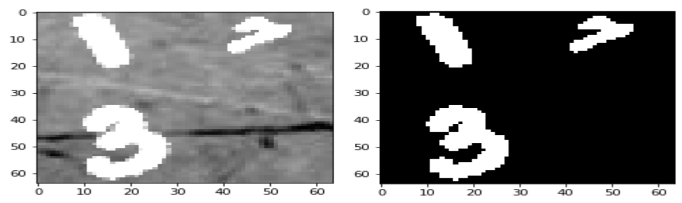
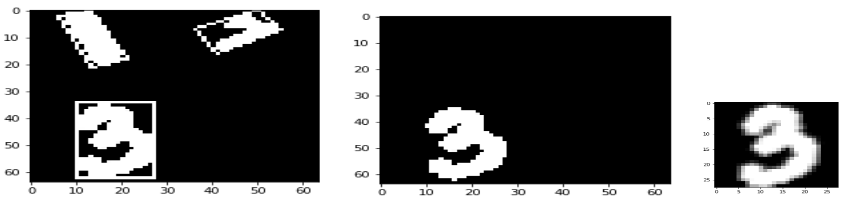


Figure 1: Background removal



(a) (b) (c)

Figure 2: Contours drawing, removing smaller digits, shifting and resizing procedures

4 Preprocessing Procedure

First of all, it is required to mention that the following preprocessing algorithm has not been applied to all of the employed machine learning approaches. Actually, the key idea behind this preprocessing was to transform the Modified MNIST problem to the typical MNIST problem. Therefore, some image processing algorithms have been utilized for finding the largest digit and removing the other digits in each image. In overall, the preprocessing procedure can be divided into the following steps:

1. **Background removal**: Binarization was applied as background removal in order to obtain clear images and to retrieve the digits in each image easily. The threshold adjusted for binarization was set to 200. The result of the binarization for one image is shown in Figure 1.

2. **Finding the largest digit**: The largest digit was detected by comparing the height of the smallest rectangular which encompasses each digit. These contours were found by using OpenCV library [12]. As an example, the contour of all digits for one image can be seen in Figure 2(a).

3. **Removing other digits:** Once the largest digit is detected, we remove the other digits from the image, as shown in Figure 2(b).

4. **Shifting the largest digit to the center of the image:** In this step, the largest digit is shifted to the center of an image. This process is performed by finding the centroid of the largest component and shifting it to the image's middle point.

5. **Resizing:** Finally, the image was resized from (64, 64) to (28, 28) dimensions since this is the size of an image in typical MNIST. The result of this procedure is shown in Figure 2(c).

5 Applied Approaches

As mentioned before, several classifiers were evaluated for this problem, and the following subsections are the concise description of applied classifiers. There were some hyperparameters in classifiers (e.g., kernel and the value of C in SVM or batch size, kernel size, and the number of layers in CNN) which provided the degree of freedom for selecting the final model. We used the grid search technique to obtain the optimal value for each of the hyperparameters of the different

Table 1: The best hyperparameters obtained for final CNNs architecture

Hyperparameter	**Value**
Number of convolution layers	3
Number of feature maps	32, 64, 128
Kernel size	3*3
Activation	ReLU
Dropout	0.3
Batch size	150
Learning rate	0.001
Number of epochs	50
Optimizer	Adam with categorical cross entropy

models. These optimal values are supposed to maximize an objective function which is defined as the accuracy of our model on the validation data.

5.1 Logistic Regression

Logistic Regression (LR) was chosen as our first classification algorithm. The sklearn module was utilized to construct the LR. The grid search resulted in C=1 as the optimized hyperparameter for this classifier.

5.2 Support Vector Machines

Support Vector Machines (SVM) find a linear separating hyperplane with the maximal margin in the higher dimensional space. The sklearn module was also used to construct the SVM. Hyperparameter C is the cost or penalty term for misclassification. Polynomial kernel with the degree of 5 and $C = 1000$ found as the best parameter selection for this classifier.

5.3 Convolutional Neural Networks

Many studies have shown that Convolutional Neural Networks (CNNs) are profoundly powerful classifiers for image recognition tasks [13, 14]. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing. As compared to Feed-Forward Neural Networks, CNNs make the explicit assumption that the input dataset consists of images. Utilizing this assumption, CNNs are able to extract and combine local features, which are fed into a set of alternating convolutional and pooling layers. CNNs have an advantage over traditional neural network architectures because they have fewer connections and fewer parameters; since the neurons of a layer are learning only local connections the total number of connections is reduced; the smaller number of parameters is a result of parameter sharing. These factors cause CNNs less computationally expensive to train. We performed an extensive implementation of various architectures for CNNs. We created CNN using Keras framework in Python. The activation for the convolution was rectified linear unit (ReLU). The max pooling layer pool size is set to (2, 2). We experimented with the dropout rate of dropout layer, with values from 0.25 to 0.5. Finally, we added a flattening layer, a fully connected layer, and a softmax layer to classify into one of the 10 digit classes. We performed hyperparameter tuning in order to find the best architecture of CNN by varying parameters. The parameters of the final architecture selected based on hyperparameter tuning as shown in Table 1.

5.4 Ensemble Learning

Majority voting has been shown to improve the performance of an ensemble of the same network trained multiple times [5]. Consequently, an ensemble of the same network trained with different training and validation data (by using

Table 2: Comparing Performance of Applied Classifiers on Modified MNIST dataset

Classifier	Pre-processing	Training Accuracy	Validation Accuracy
Linear Regression	Yes	43.5	39.5
SVM	Yes	78.0	74.4
CNN	Yes	95.5	93.6
Xception	No	99.0	97.6

splitting of data randomly) was used to improve prediction performance.

5.5 Very Deep Neural Networks

Regarding the Kaggle competition, we make an attempt to use existing very deep models that have shown satisfying results on the ImageNet dataset. We employ the VggNet [15], the ResNet [16] and the Xception [7] models. From our experimental results, we found that the Xception model outperforms the VggNet and the ResNet models. The initial learning rate for the Xception model is set to 0.005, and we decay the learning rate by 0.94 every two epochs. We use online data augmentation to improve the accuracy of the model and to prevent it from overfitting. In this regard, we randomly shift images up to 5 pixels in horizontal or vertical directions and randomly rotate images up to 45 degrees clockwise or counterclockwise. Furthermore, We utilize the ensemble method with 10 and 20 Xception models trained with 90% and 95% of the training set respectively. It is worth mentioning that we also apply online data augmentation for the ensemble methods.

The images used for training the Xception model should be three channels with the minimum size of 71×71 pixels. Therefore, we reshape our dataset form 64×64 to 71×71 and convert them from one channel to three channels using OpenCV library [12]. We also use the batch size of 50 images for training the model. The One-hot encoder is applied on Labels, and all pixel values are downscaled from (0:255) to (0:1). For final validation and test results, all models are trained for 100 epochs. It should be noted that we use Adam optimizer with *categorical_crossentropy* and include the top layer of the Xception model with 10 classes for classification part.

6 Results

This section shows classification accuracy achieved by different classifiers and compares the performance between different model configurations. We split the training set into two portions (i.e., 90 % as training data and 10 % as validation data) to train our models on the training portion and test them on the validation set.

6.1 Comparing Classifiers Performance

Accuracy on training and validation datasets of all used classifiers are shown in Table 2. As indicated in Table 2, preprocessing used for all the classifiers but Xception. It is worth to mention that matrix elements of each image were used as features for LR and SVM. Furthermore, for the reported CNNs performance, the ensemble of 15 networks was utilized. Finally, as can be seen from Table 2, the best performance among all methods we experimented is given by Xception. The obtained result was expected due to the fact that Convolutional Neural Networks have shown leading to the highest performance on MNIST dataset based on previous literature.

6.2 More on Xception Model

Table 3 shows the results for the aforementioned models exploiting the Xception model. According to Table 3, online data augmentation has a positive impact on the validation accuracy, and it prevents the model from overfitting. Furthermore, we can observe from Table 3 that the ensemble method improves the accuracy by 0.8%.

Table 3: Accuracy results over training, validation and test sets for different Xception models

Model	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
Xception	99.00	97.60	97.333
Xception with data augmentation	98.95	97.75	97.866
10-Xception	98.94	97.75	98.433
20-Xception	98.72	97.36	98.400

While the Xception model uses the learning rate of 0.045 for the ImageNet dataset [7], our experimental results show that the mentioned learning rate is not suitable for our dataset, and the Xception model fails to learn the training data. We found that the learning rate values between 0.001 and 0.005 are the best ones for our dataset. Figure 3 shows the training accuracy of Xception model over learning rates of 0.005 and 0.001.

The Xception model learns the training dataset completely (i.e., with 100% accuracy) and overfits the dataset. To solve this issue, we used online data augmentation. Before each epoch begins, the training dataset is manipulated (e.g., each image is rotated and is shifted) and a new dataset is generated. In other words, data augmentation helps us to generate an infinite number of training data. Performance of the Xception model on training data with and without data augmentation is shown in Figure 4. The Xception model reaches the accuracy of 100% without data augmentation and overfits the training data. On the contrary, the accuracy of the Xception model on training data remains below 98% when using data augmentation, which is close to the accuracy of test dataset. For instance, the training accuracy of the Xception model without data augmentation constantly improves while the validation accuracy remains below 97.75% as demonstrated in Figure 5.

There is an element of randomness in the way classifications change (e.g., near the decision boundary) when changes are made to the parameters of the model. This may affect the validation accuracy more than the training accuracy (see Figure 5). The reason is that during training, the small changes made to the parameters are specifically designed to optimize a training objective which is a good proxy for training accuracy. Therefore, the effect is less like a random perturbation of the parameters. In fact, it is more like pushing classifications for training examples in a consistent direction. The fluctuations in validation accuracy could also just be because of the small size of the validation set. The validation accuracy in Figure 5 is obtained from the ensemble method with 20 Xception models where the validation set is only 5% of the whole dataset. When the validation size is too small, any small changes in the output could cause large fluctuations in the validation error (or accuracy).

7 Discussion and Conclusion

As a first step, we tried to preprocess the modified MNIST to achieve a dataset similar to typical MNIST. This preprocessing was performed by extracting the largest digit and removing the other digits in each image. However, due to the fact that images were rotated and had harsher and extrapolated contours, hence preprocessed images did not lead to the typical MNIST dataset perfectly. This allows us to acquire the accuracy close to 93.6%, which is not the best accuracy obtained for typical MNIST (99%) based on literature [5]. Therefore, we tried to train deep learning models without any preprocessing to prevent them from losing any information due to the preprocessing. As a result, higher performance was achieved. The ensemble of 10 Xception model with online data augmentation achieved the highest accuracy among all implemented designs with the accuracy of 98.433%. It is shown that data augmentation techniques can prevent overfitting and increase the performance of the Xception model. In this work, we used the rotation and shifts properties of online data augmentation provided by Keras library. There is still another interesting data augmentation such as zooming, shearing, and feature-wise or center-wise standard normalization. For future work, we intend to use the mentioned data augmentation techniques to investigate their impacts on the training of our model and the accuracy of the test dataset. Furthermore, it would be an interesting study to evaluate the performance of different DNNs outputs ensembles (instead of using only one model).

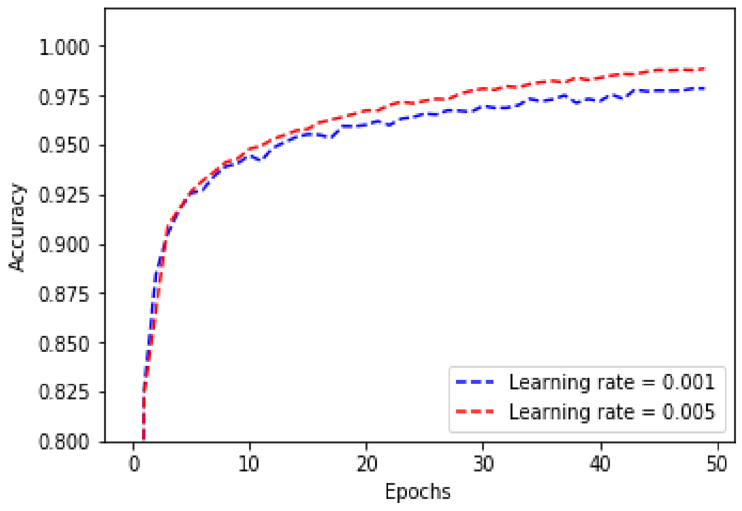


Figure 3: Training accuracy of Xception model over two different learning rates.

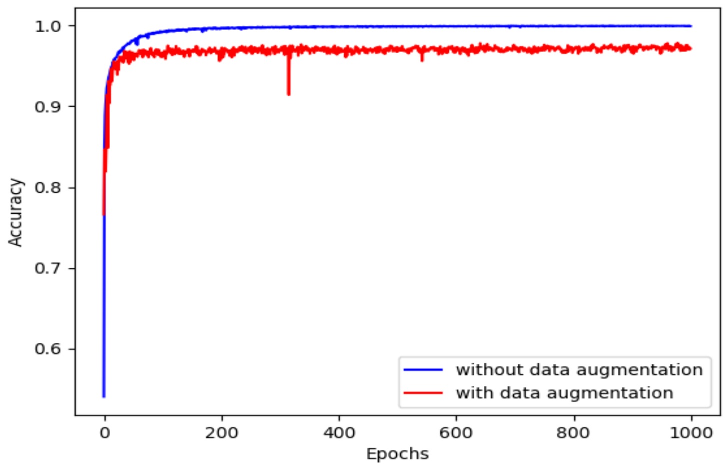


Figure 4: Training accuracy of Xception model with and without data augmentation for 1000 epochs.

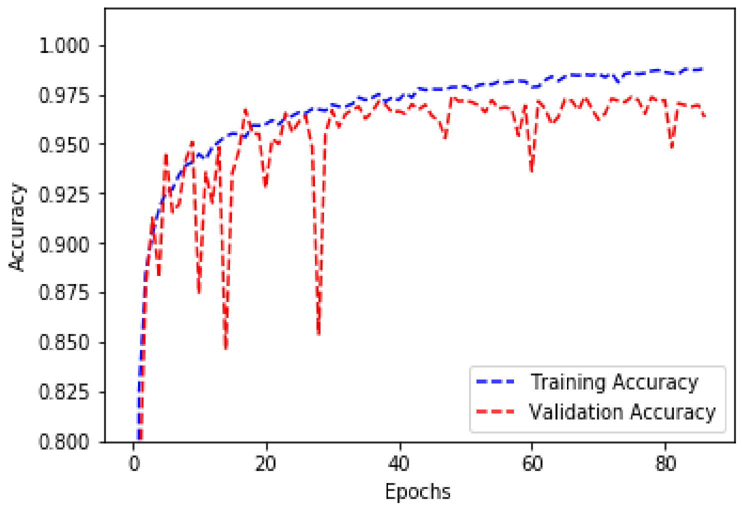


Figure 5: Validation and training accuracy results of Xception model for 100 epochs.

References

[1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Graph Neural Processes: Towards Bayesian Graph Neural Networks”, Advances in Neural Information Processing Systems, 1990, pp. 396–404.

[2] LeCun et al., “The MNIST Dataset Of Handwritten Digits (Images)”, 1999.

[3] Y. LeCun, Y. Bengio, and others, “Convolutional networks for images, speech, and time series”, 1995.

[4] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, Advances in Neural Information Processing Systems 25, 2012, pp. 1097-1105.

[5] D. C. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column Deep Neural Networks for Image Classification”, CoRR, vol. 1202.2745, 2012.

[6] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Convolutional Neural Network Committees for Handwritten Character Classification”, in International Conference on Document Analysis and Recognition, 2011, pp. 1135-1139.

[7] F. Chollet, “Xception: Deep Learning With Depthwise Separable Convolutions”, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[8] Z. Zhen, G. Pan, and S. Songlin, “License Plate Recognition System Based on Transfer Learning”, Signal and Information Processing, Networking and Computers, Singapore: Springer Singapore, 2019, pp. 42-49

[9] M. Rahaman Mamun, Z. Al Nazi, and M. Salah Uddin Yusuf, “Bangla Handwritten Digit Recognition Approach with an Ensemble of Deep Residual Networks”, in International Conference on Bangla Speech and Language Processing (ICBSLP), 2018, pp. 1-4.

[10] MNIST database: `https://en.wikipedia.org/wiki/MNIST_database`

[11] Modified MNIST database: `https://www.kaggle.com/c/comp-551-w2019-project-3-modified-mnist/data`

[12] OpenCV-Python Tutorials: `https://opencv-python-tutroals.readthedocs.io/en/latest/`

[13] K. Labusch, E. Barth, and T. Martinetz, “Simple Method for High-Performance Digit Recognition Based on Sparse Coding”, IEEE transactions on neural networks, vol. 19, no. 11, 2008, pp. 1985-1989.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, vol. 86, no. 11, 1998, pp. 2278-2324.

[15] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, CoRR, vol. 1409.1556, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, CoRR, vol. 1512.03385, 2015.