Modified MNIST Calssification

COMP 551 - Mini Project 3

Mohammad Hamed Azizi (260812541) Saeed Shoaraye Nejati ((260890049)
Paniz Bertsch(260862054)

March 2019

Abstract

The objective of this project is to perform a supervised image classification task using convolutional neural network (CNN) on a modified MNIST dataset. Our dataset consists of $50,000$ gray-scale images of handwritten digits with 64x64 pixel resolution. Each image in dataset contains more than one digit and a background that contains noise. MNIST images are preprocessed using OpenCV library functions so that images can be classified based on the digit that occupies the most space. Keras library with Tensorflow backend, is used to build a CNN (Convolutional Neural Network) and also, experiment with pretrained models. Our CNN model is trained and validated using $37,500$ and $2,500$ labeled images respectively, and our best model is used to classify $10,000$ test images into 10 groups (0-9) with test results to be submitted to Kaggle website. This project was in form of a in-class competition on the Kaggle website. We achieved an accuracy of 0.95566 on the Kaggle for our test set.

1 Introduction

In this project, we design a supervised classification model for a Modified MNIST prediction task. Our dataset consists of $50,000$ grayscale images with 64x64 pixel resolution, containing MNIST handwritten digits, and each image may contain multiple digits and background noise. Each digit in an image, has a different size and has been transformed (rotated, translated, etc.), therefore image preprocessing tasks are performed prior to feeding the data to our model. OpenCV and numpy libraries are used for preprocessing to find and crop the largest contour in each image, and normalize the pixel values. Also, our classification model is a CNN (Convolutional Neural Network) implemented using Keras library with Tensorflow backend and written in Python. Training dataset consists of $37,500$ images labeled based on the digit that occupies the most space in the image, and is fed to our CNN for training the model. Validation data consists of $2,500$ labeled images and is used to fine-tune hyperparameters and select the best model. Our test dataset contains $10,000$ unlabeled images and best CNN model is used to classify these data into 10 classes (0-9). Finally test labels are submitted to class competition website on Kaggle which resulted in an accuracy of 0.95566 for prediction of labels on test data.

MNIST database (Modified National Institute of Standards and Technology database) is a collection of handwritten digits used for training and testing of various image processing systems [1]. We use a model based on Convolutional Neural Network (CNN) to detect the largest digit present in each image in our MNIST dataset.

OpenCV is an Open-source Computer Vision library designed for computational efficiency and with a strong focus on real-time application. It is written in optimized C/C++ and has become popular worldwide for use in image processing, robotics and real-time video and image applications [2].

Convolutional Neural Network (CNN) is a state-of-the-art image recognition technique, used mainly in computer vision and image analysis, and is part of a larger family of modern machine learning methods called deep learning. CNN based models consist of networks of fully and partially connected layers designed to recognize visual patterns directly from pixel images with minimal preprocessing. These networks can recognize patterns with extreme variability such as handwritten character and digits (MNIST), and are robust systems with high accuracy even in presence of distortions and simple geometric transformations [3].

Keras is a high-level neural network API, written in Python and can run on top of other libraries such as Tensorflow and Theano. Keras is highly popular and has a large community mainly due to offering a single API across many popular neural network libraries, and because of enabling fast experimentation [4]. For reasons mentioned above, we have selected Keras to implement our CNN.

Tensorflow is also an open-source software library for neural network developed originally by Google researchers and engineers working on Google Brain team to conduct machine learning and deep neural network research [5].

2 Related Work

LeNet was the first successful application of CNN developed in 1990s to read zip codes, digits, etc. AlexNet was the first work that popularized Convolutional Networks in Computer Vision. AlexNet was similar to LeNet but deeper and bigger with multiple convolutional layers. ZF Net, was an improvement to AlexNet by tweaking hyperparameters. GoogleLeNet main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network [6, 7]. VGG developed by University of Oxford, used a smaller kernel size and is very popular [6]. ResNet avoids the problem of vanishing gradients, by reusing activations from a previous layer until the adjacent layer learns its weights[8]. Best classification of MNIST handwritten digits are achieved by using multicolumn deep CNN with width normalization in 2012, and regularization of neural networks using DropConnect with error rate of 0.23% and 0.21% respectively [9, 1, 3]. Also due to a wide range of application of AI driven computer vision, many universities and companies are active in this research area such as facial recognition, real-time classification and segmentation, driverless cars ,robotics and more.

3 Dataset and Setup

For this project we use $50,000$ MNIST data in grayscale format and 64x64 pixel resolution. Each image in our dataset contains multiple digits with different sizes and a noisy background. Also, digits in each image has been rotated and transformed, which requires preprocessing steps to acquire the largest bounding box containing a digit, remove extra noise and normalization of pixel values. Training and validation sets, consist of $37,500$ and $2,500$ images respectively, labeled based on the digit occupying the most area in the image. Therefore we have 10 classes from 0 to 9. Remaining $10,000$ images are unlabeled test data to be classified using our model into 10 classes. We use OpenCV and numpy library for preprocessing images. A max threshold with value of 254 has been applied to all pixel values and fed to an OpenCV library function to find all contours in a given image (A contour is a curve joining all the continuous points along the boundary). Largest contour in each image is then selected and saved in a new array. We find the minimum area of contours, to count for rotation of digits in an image [2] and also, find X & Y coordinates of contours which can be used to crop smallest rectangle containing the digit from an image. Also, we first remove extra black pixels from the boundaries of each

contour before cropping the digit out of original image. We reshape all arrays containing image data, to have a dimension of 28x28 pixels by adding extra padding along the smaller edges when required. Finally, we normalize all values (dividing by 255) and convert our data to 3 dimensional (28x28x1) numpy arrays with float32 data type. All labels in training and validation datasets are converted to binary class matrix to be used with categorical-crossentropy.

4 Proposed Approach

Convolutional Neural Networks (CNN) are biologically-inspired variants of Multilayer perceptrons (MLP)[3] which exploit spatially-local correlation in images by enforcing a local connectivity pattern between neurons of adjacent layers [7]. A CNN consists of an input and an output layer as well as multiple hidden layers. These hidden layers consists of convolutional layers, ReLU layers (an activation function), pooling layers (e.g. MaxPool), fully connected layers and/or normalization layers [10, 11]. CNNs are trained using back propagation. They are made up of neurons that have learnable weights and biases, and have a loss function on the last (fully-connected) layer such as Sigmoid for binary and Softmax for multi-class classification tasks. We implement 2 CNN models based on figures 1 and 2 , using Keras library, and optimize hyperparameters to select the best performing model on our validation dataset. We also experiment with pretrained models VGG16 and ResNet. Following section describes steps and layers used in our CNNs.

4.1 Data augmentation

Data augmentation is done using ImageDataGenerator module to improve generalization. Small random rotation, shifting horizontally and vertically, and zooming is applied to training data to help with capturing important features in our training dataset [4].

4.2 Callback Functions

Callback functions from Keras library, used to save internal states and statistics of the model during training at the end of an epoch if accuracy of validation set has increased, and also to reduce learning rate when model accuracy has stopped improving.(we used two functions, ReduceLROnPlateau and LearningRateScheduler)

4.3 Convolutional Layer

3x3 kernel size and ReLU activation used at each convolutional output. Activation layer controls how the signal flows from one layer to another. Output signals which are strongly associated with past references would activate more neurons, enabling signals to be propagated more efficiently for identification. The most common activation function is Rectified Linear Unit because of faster training speed (ReLU = Max(0,x)) [7, 4].

4.4 Max-pooling

2x2 max-pooling with strides of 2x2 is used at the end of each block. Max-pooling is a form of non-linear down-sampling and outputs the maximum value. Max-pooling is useful in vision for two reasons: By eliminating non-maximal values, it reduces computation for upper layers. Since it provides additional robustness to position, max-pooling is a “smart” way of reducing the dimensionality of intermediate representations. [10, 6]

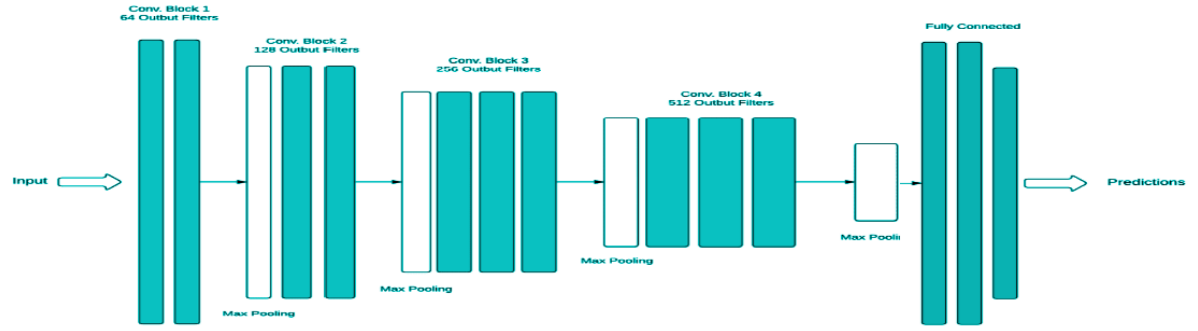

Figure 1: CNN1

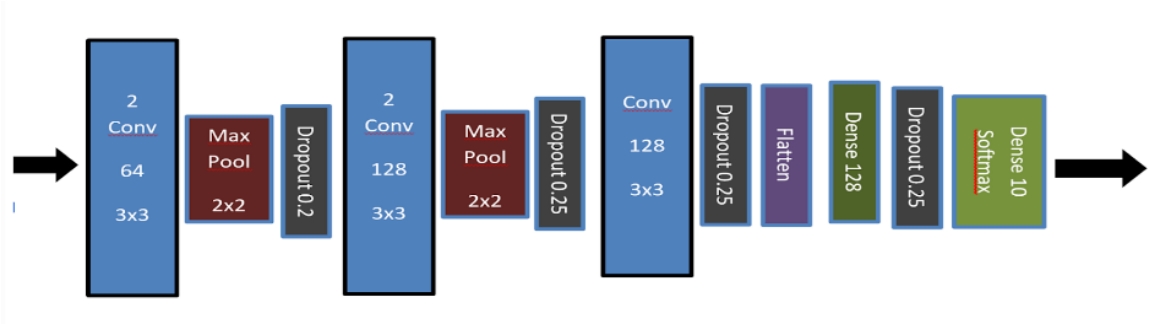

Figure 2: CNN2

4.5 Dropout

Dropouts applied by randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting[4].

4.6 Output Layer

Softmax activation function is used at the output layer with 10 classes [7].

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

4.7 Batch Size & Number of Epochs

Our final CNN is trained on data generated in batches of 128 and 30 epochs, chosen based on accuracy and computational efficiency.

4.8 Compilation

CNN is compiled with Adam optimizer, accuracy metric and categorical-crossentropy loss. Adam is a computationally efficient algorithm for gradient-based optimization aimed towards machine learning problems with large datasets and/or high-dimensional parameter spaces [12].

4.9 Validation Dataset

Prediction on validation data is used to evaluate the accuracy of the network. This will help to fine-tune hyperparameters, prevent overfitting on the training data, and lower the learning rate

if necessary.

4.10 VGG16 & ResNet

A pre-trained network is a saved network that was previously trained on a large on a large-scale image-classification task. We adapted architecture of VGG16 for one CNN, used an ensemble of our CNN and VGG, and also experimented with ResNet model[6, 8].

5 Results

After increasing the number of epoch in models, the chance of overfitting increases and we observed that augmentation in the prepossessed part had a great impact on the accuracy. Deeper net based on VGG, had slow convergence rate on validation and after 10 epoch, model reduced the learning rate to 0.0005000000237487257. The results of CNN with 2 convolutional layers and VGG with 5 layers was ensembled to achieve an accuracy of 0.95566 on test data. we observed only 1% improvement in accuracy using ensemble method of different CNN models(cnn with three layers & VGG5). Table **??** summarizes our results on validation set and figure 3 shows convergence of best model.

Model	Accuracy	Epoch
CNN1	0.94200	20
CNN2	0.94840	50
Restnet	0.8700	20
Ensemble (CNN+VGG)	0.955	20

Table 1: Validation Set Results

6 Discussion and Conclusion

Objective of this project was to implement a supervised classification task using convolutional neural networks. Our Training and validation datasets contained $40,000$ labeled images, and the remaining $10,000$ data was used for testing our model. We used Keras libraries with Tensorflow backend in Python, to build 2 CNN models and experimented with ResNet and VGG5 pretrained models. We chose our CNN with 3 convolutional layers to predict test labels due to small increase in accuracy using other methods. This project was in a form of in-class Kaggle competition and we submitted our best CNN's prediction for test data to Kaggle website. We achieved an accuracy of 0.95566 on test data. A larger dataset and experimenting with VGG19 can improve the accuracy of our prediction. Also having higher processing power and storage could help with hyperparameter optimization and training our CNN. Also, it is almost impossible to reach 100% accuracy since some of handwritten digits are too illegible and even people will have a difficult time to judge their specific meanings.

7 Statement of Contributions

All members of the group contributed equally.

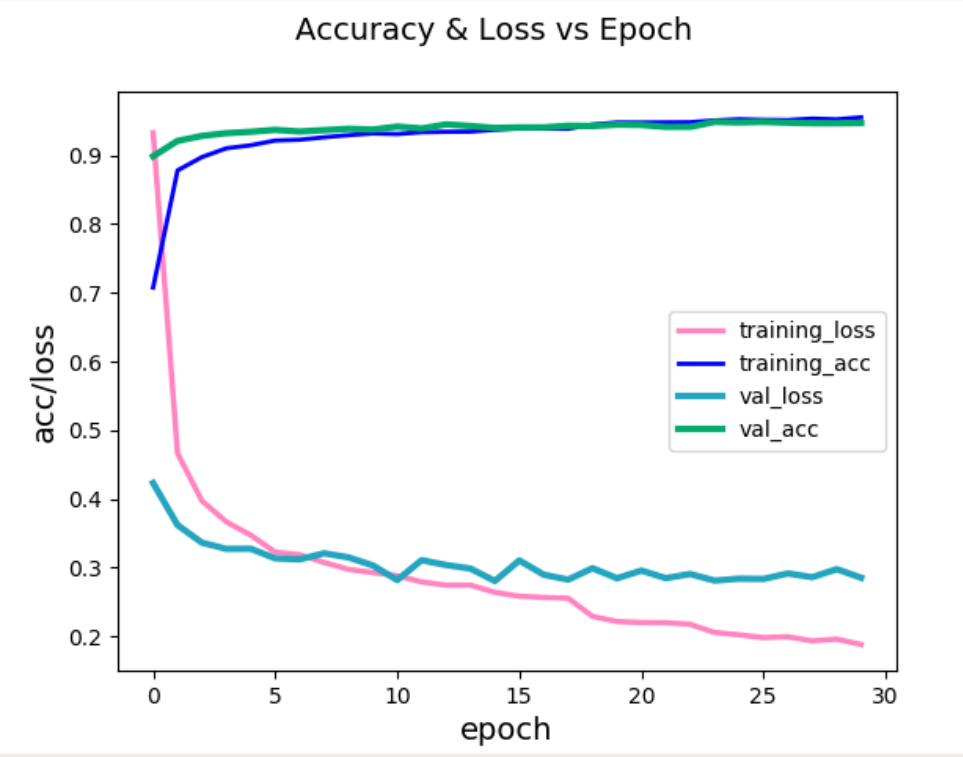

Figure 3: Accuracy & Loss at each Epoch

References

[1] Christopher J.C. Burges Yann Lecun, Corinna Cortes. The mnist database.

[2] https://opencv.org/.

[3] Convolutional neural networks (lenet).

[4] Keras documentation.

[5] Tensorflow github.

[6] University of Oxford. Visual geometry group.

[7] Stanford University. Cnn for visual recognition.

[8] Xiangyu; Ren Shaoqing; Sun Jian He, Kaiming; Zhang. Deep residual learning for image recognition. *Cornell University*, 1512(3385), 2015.

[9] Yann Lecun. Lenet-5, convolutional neural networks.

[10] Jurgen Schmidhuber Dan Ciresan, Ueli Meier. Multi-column deep neural networks for image classification. *arVXiv*, 1(275):891–921, 2012.

[11] Yu Qiao. Handwriting database.

[12] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *arXiv*, 9(1412), 2015.