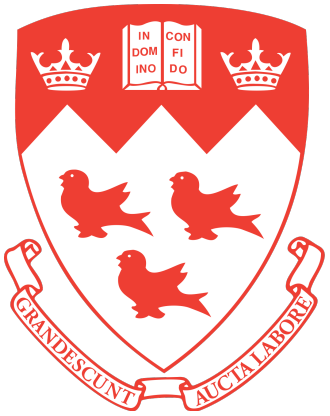
Multi-label Classification of Image Data

Saad Shahbaz : 260845253,
Ria Stevens : 260887564,
Navneet Kaur : 260779742



A project presented for the class of
COMP 551 : Machine Learning

Department of Computer Science
McGill University
Quebec,Canada
27 November, 2021

1 Abstract

This project implements Convolution Neural Networks(CNNs) to classify combo MNIST dataset. Based off of historically significant neural networks, we experimented with several CNN architectures [SSP03] such as LeNet-5, ResNet, VGG , semi-supervised image classification with unlabeled data [EOG19], and homespun sequential CNN models implemented both with Keras and PyTorch Our first approach to tackle the problem was by implementing LeNet-5 with 8 layers using PyTorch which performed relatively poor on our dataset giving an accuracy of merely 60.23%. After numerous other trials using various CNN architectures, we ultimately achieved the highest accuracy of 93.8% on the training set by implementing a modified version of sequential CNN with Keras, comprising of 13 total layers [KSH12]. This model accomplished an accuracy of 94.08 % on the Kaggle testset. Through experimentation it was also found that our CNN models performed significantly better with noise in the data [Kim].

2 Introduction

Convolutional Neural Networks are the standard form of neural network architecture for solving image associated problems. Solutions for image classification tasks such as pose estimation, object or face detection, number prediction, and more all utilize CNN architecture variants. CNNs apply a filter to the input data to create a feature map which summarizes the presence of detected features in the input.CNNs can be tremendously powerful because of their ability to learn the filters during training given the bounds of the prediction problem.

3 Dataset

The dataset comprises of 60,000 images for training the classifier of which 30,000 images are unlabelled. Each image has dimensions of 56 x 56 and comprises of two characters: one digit (from numbers 0-9) and one English alphabet. The size of the characters present in the images varies across the dataset. Further, several of the images also contain noise. Several of the CNN models were trained both on data before (as shown in Figure 2a) and after noise removal (as shown in Figure 3a and 3b). However, it was found that removing the background high intensity pixels from the images reduced the accuracy of our models. The accuracy of LeNet5 reduced from 89% to approximately 67% [Section 5]. Further, attempts at normalizing and re-scaling the images to various sizes also lead to no improvement in accuracy. Our model's performance was improvised by the process of data augmentation. Given images were rotated and translated to be augmented to the input data.

4 Results

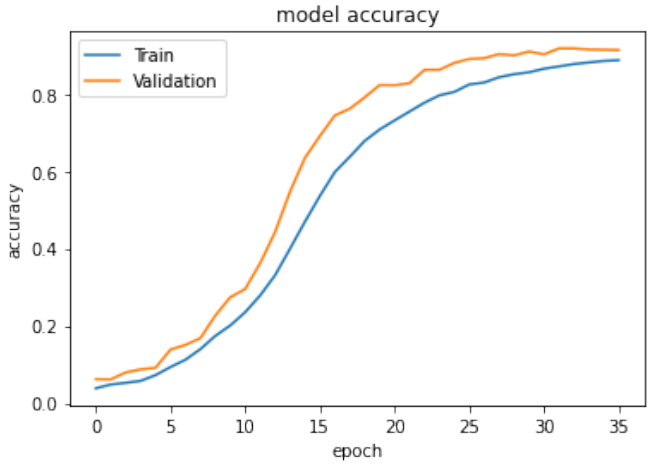
Our presented model utilizes 9 convolutional, 1 flattening and 3 fully-connected layers and takes as input 56 by 56 images for classification into one of 260 classes, with each class being a unique (number, letter) pairing. Each layer, safe the final dense layer, which uses softmax activation, uses a ReLU activation function [NH10]. We gradually increase the size of the convolutional layers by powers of 2 from 16 to 256, use 1-padding on each layer (for ease of development and to be able to maintain a wide network), and use a kernel size of 3×3 for every layer after the first, for which we use a kernel of size 5×5. To increase the convergence speed of our model, we include batch normalization after each convolutional layer. We also include some max pooling, to decrease the size of each feature map, and dropout functions, to regularize, after various layers. Their locations were determined through extensive experimentation of where they would produce the best results. Altogether, the first convolutional layers extract low-level features from the data, the later convolutional layers identify finer details of such features and the linear layers classify each image as belonging to a certain class based on its extracted features.

We trained our model on 30,000 training images from the Combo-MNIST over 30 epochs with a batch size of 15 and using cross-entropy loss. Each of these choices was deliberate: The validation accuracy of model converged after just over 30 epochs when the training data was split, so we elected to train it up to 30 epochs on the larger dataset; Models we trained using lower batch sizes did not demonstrate any ability to classify images after a reasonable number of epochs–they were achieving 0.004 accuracy,

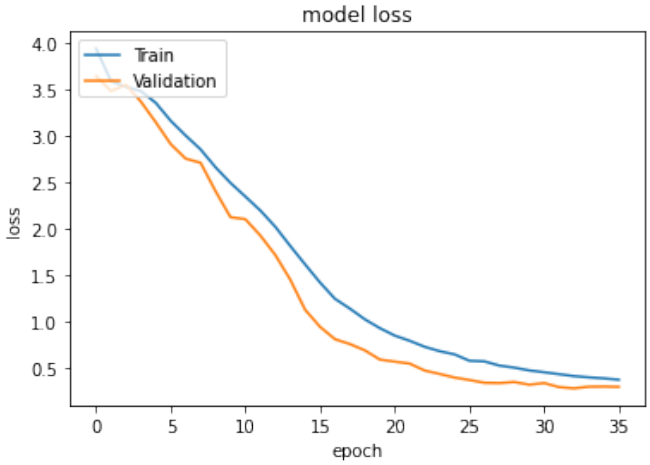
even with what one would expect using random-assignment–and higher batch sizes produced models that converged, but slowly, as a result of less randomness in their gradient computations; and cross-entropy loss was the natural choice for a classification task with a single 'true' label, such as our own.

We also experimented with augmenting our data and training our model to make predictions on unlabelled training data, then retraining it on the combined dataset of these results and the original training data. Our data augmentations consisted of rotations, zooms, translations and shears. In order to preserve the label of each letter, we limited the extent of these transformations, for example restricting rotations to be by no more than 15 degrees. Neither of these techniques performed significantly and consistently better than models trained without such changes.

We started with 36 classes, and had an accuracy of 89. Changing the classes to 260 gives an accuracy of 93. This happens since the loss function being utilized is cross-entropy loss, which performs best when the data has a single 'true' label.



(a) Training and test accuracy of 93.8 and 94.08 respectively are observed using the 13 layer CNN



(b) Figure shows cross entropy loss across test and validation dataset.

5 Discussion and Conclusion

The 22% drop in accuracy found after pre-processing of the images make an excellent point of further research [Kim]. Upon investigation, it was found that removing noise from a small dataset being utilized in CNN training can act as a source of reverse augmentation as the examples can be memorized by neural nets. Therefore, in certain CNN architectures, adding noise during the training process can reduce generalization error. Further, our attempts of training a VGG16 model on the given data were unsuccessful with the given resources. This is due to the fact that VGG16 is a very deep neural network and can therefore require tremendous resources.[KSH12].

6 Statement of Contributions

Ria Stevens, Saad Shahbaz, and Navneet Kaur collaborated for this project fairly. All members contributed to the best of their abilities.

References

[EOG19] Joseph Enguehard, Peter O'Halloran, and Ali Gholipour. Semi-supervised learning with deep embedded clustering for image classification and segmentation. *IEEE Access*, 7:11093–11104, 2019.

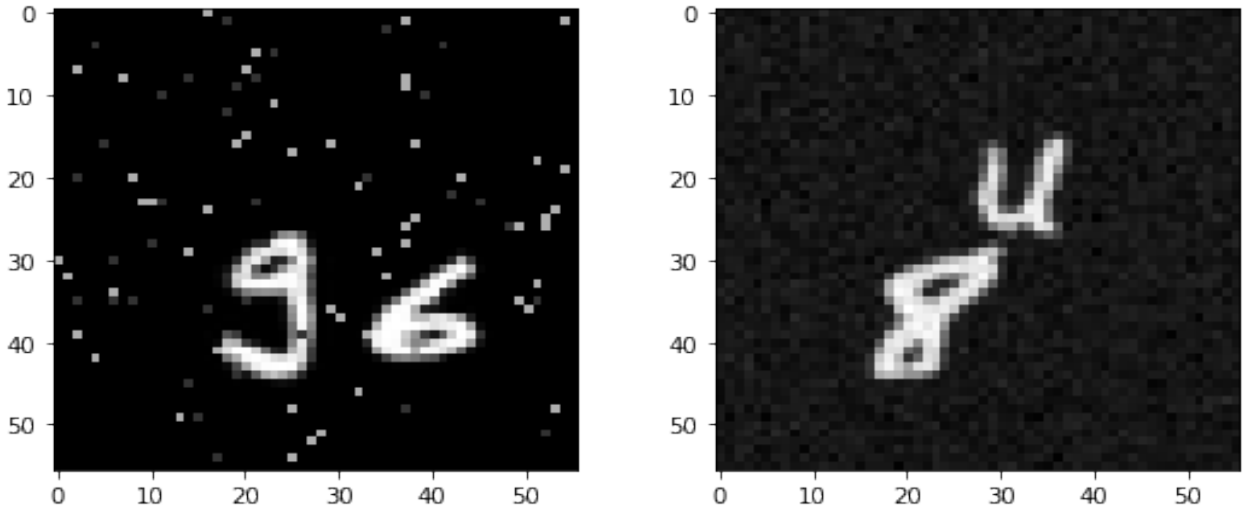
[Kim] Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[NH10] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

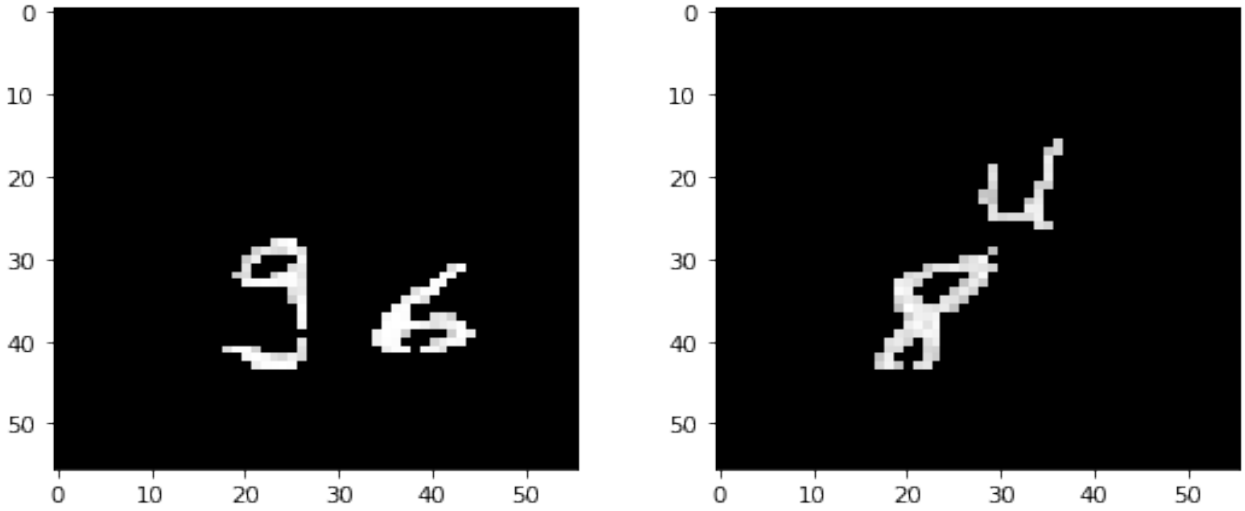
[SSP03] Patrice Simard, David Steinkraus, and John Platt. Best practices for convolutional neural networks applied to visual document analysis. pages 958–962, 01 2003.

7 Appendix



(a) (b)

Figure 2: The figure shows input dataset images before any pre-procesing.



(a) (b)

Figure 3: Figure shows input images after noise removal.

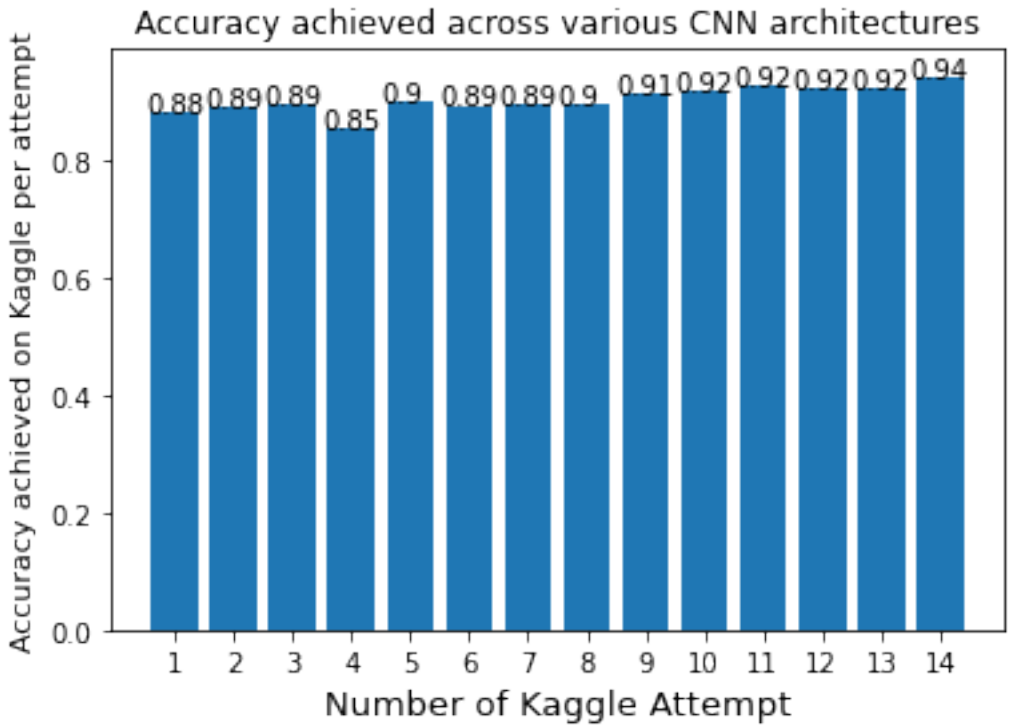


Figure 4: Figure above shows accuracies achieved across various CNN architecture trials for this project.