# Convolution Neural Network for modified MNIST dataset

Han Zhou [1], Hao Shu[2] and Jiewen Liu[3]

***Abstract*— In this project, we have practiced the build-up process of Convolutional Neural Network (CNN) on the Google Colab platform, and used the CNN model we selected and built with package mainly TensorFlow and Keras to practice in predicting a set of modifies MNIST dataset. After the investigation of different CNN models available to us, we found the best option for this task: VGG-16, and customized the layers based on the model preset to optimize its performance. The competition results show its outstanding performance among all candidates we chose and achieved an accuracy of 97.3% in the Kaggle competition. In this project, we have gathered knowledge of implementing CNN model in practice and predict the dataset from the real world, proved this knowledge is effective and can be used as instructions for the model building and modifying afterwards.**

## I. INTRODUCTION

### A. Preliminary

With the rise of computational power, neural has been a popular machine learning algorithm in solving a variety of issues. And derived from it, by introducing the idea of convolution and pooling, Convolutional Neural Network(CNN) is created. The recent practice of CNN has garnered tremendous success in a variety of domains in computer vision, namely object localization, object detection, image segmentation, and image captioning [1]. The Neural Network acts in a similar way of Biologic Neural Network and consisted up by multiple layers of perceptions. The nature of its structure makes it prone to overfitting issues that overcomes a big challenge in machine learning.

### B. Related work

Convolutional neural networks have been the master algorithm in computer vision in recent years, and developing recipes for designing them has been a subject of considerable attention [2]. The history of convolutional neural network started with LeNet-style models [3], which were simple stacks of convolutions for feature extraction for spatial sub-sampling. In 2012, these ideas were refined into the AlexNet [4]: Alex Krizhevesky et al. proposed a deeper and wider CNN model compared to LeNet and won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge [5] AlexNet achieved state-of-the-art recognition accuracy against all the traditional machine learning and computer vision approaches [6]. It was a significant breakthrough in the field of machine learning and computer vision for visual recognition and classification tasks. However, AlexNet consists of fully connected layers which are extremely computationally expensive, so it is not discussed in this paper.

The interest in deep learning increased rapidly after AlexNet was proposed, and a large number of architectures have been invented. Among all of these structures, some of the architectures are designed especially for large scale data analysis (such as ResNet [7][8]), whereas the VGG [9] network is considered a general architecture. Some of the architectures are dense in terms of connectivity, such as DenseNet [10], and some of the architectures are based on an inverted residual structure, such as MobileNet [11]. At this point, the Inception architecture was also introduced as GoogLeNet [12], by Szegedy et al. in 2014. Later GoogLeNet was refined as InceptionV3 [13] and most recently Inception-ResNet in 2016 [14]. In 2017, Francois Chollet proposed Xception [2], a new architecture based on the inception architecture, which significantly outperforms InceptionV3 on a large image classification dataset.

### C. Task introduction

The task of this project is to train a multi-class classification model using the knowledge of the Convolutional Neural Network. After feeding training data, the model should be able to find out the largest digit within the given images of size 128∗128. The dataset used in this project is a set of Modified MNIST dataset.
The main challenge of this project is to find an efficient way of training models by edit and arranging proper layers in a convolutional neural network model.

## II. DATASET AND SETUP

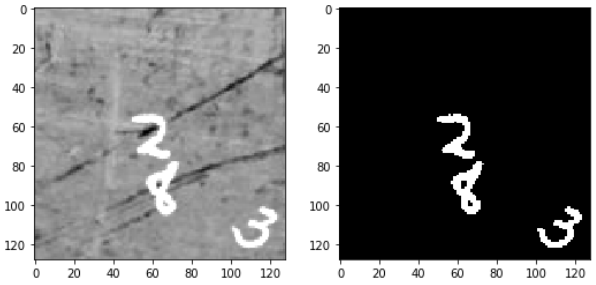
### A. Dataset acquiring

The provided data for this training is a modified MNIST dataset which consists of 60,000 gray-scale images, each is of size 128∗128 and contains multiple hand-written digits 0-9 on each sample. To increase the difficulty of classification, most of them are also filled with random noise pattern. 50000 of them comes with a tag indicating the largest number in that sample, which can be used as the training set. The other 10000 samples, waiting to be predicted, will be used as a test dataset to compete for the prediction accuracy among teams.

[1]**Han Zhou** Third year Software Engineering student at McGill University.
[2]**Hao Shu** Third year Software Engineering student at McGill University.
[3]**Jiewen Liu** Second year Computer Science student at McGill University

### B. Data preprocess

While attempting to improve the performance of the classifiers, a preprocessing method is implemented. This preprocessing method takes advantage of the regularities observed in the data: the digits in an image, which are the elements of interest, were all white, while the background and noise were gray and black, as shown in Figure.1(a).



**(a)** Before processing **(b)** After processing

**Fig. 1:** Comparison between the original MNIST sample and the preprocessed MNIST sample

To remove the noise in the input image as much possible while keeping the informative data unaffected, we set up a threshold for pixel color to filter out the non-informative patterns: if the grey-scale value of a pixel is higher than 220, we classified the color of the pixel as informative pixel and set the grey-scale value to 255, which is white. For the pixel with a grey-scale value of less than 220, we will set them to zero, which is black. More formally, we alter the grey-scale value of pixels in the image with a new value by the following rule:

$$RGB_{new} = \begin{cases} 0, & \text{if RGB} < 220 \\ 255, & \text{otherwise} \end{cases} \tag{1}$$

With this method, the pixels of the informative handwritten digits are set highlighted, and all the other pixels are set to black, as shown in Fig.1(b). This filter method can significantly reduce the biases caused by the noise, and enhance edge detection at the feature learning process, leading to better classification performance. In practice, this action increased both the speed of local minimum convergence at each epoch as well as the final accuracy of the model.

### C. Ethic consideration

It's worth noticing that this type of model can be used in the recognition of verification code, which provides a convenient way of surpassing the defense system of websites trying to guard them against intense data crawling using verification code. As a programmer responsible for social safety, we should not use this algorithm to escape the guard of the website, as they may lead to the heavy load of the target website, and other potentially illegal practices.

## III. EXPERIMENT APPROACH

### A. Environment setup

In this experiment, we first built up a general code framework using packages, mainly TensorFlow and Keras for data processing, model training and predicting. The code is run on the Google Colab platform, which provides around 13 GiB CPU resources and 15 GiB GPU resources. This limited the selection of model as the one requires extra large memory space or GPU memory will not be able to train on this platform. However, it is enough for most of the available choices of the model provided by Keras.

### B. Model choose and modification

After acquiring information on basic features and expected performance, a list of CNN Models was chosen as candidates for the data training:

*1)* ***ResNet 152 &101****:* ResNet is presented in 2017 with a depth of up to 152 layers. It can reduce the training error while deepening the depth of the network, and solve the problem of gradient dispersion, improving network performance [8]. Most importantly, ResNet can not only be very deep but also has a very simple structure. It is a very small single module piled up, its unit module block is shown in Fig. 2, , where x means the input and F(x) means the output of the weight layer, the final output is the sum of F(x) and x. For this task, we pick the ResNet 152 and 101 as a training candidate and set their initial learning rate high to overcome the slow learning issue.

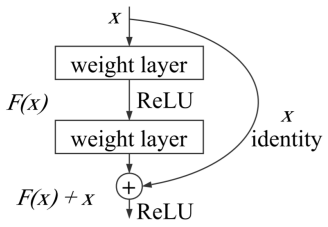



**Fig. 2:** Unit module block for ResNet [7]

*2)* ***DenseNet 169 & 201****:* DenseNets have several compelling advantages: they alleviate the vanishing gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters [10].It has a more dense structure, which requires more memory resources in the training process comparing to ResNet, however, the required resource is under a reasonable range. In this lab, we chose DenseNet 169 and 201 based on their layer identity as the candidate for model training.

*3)* ***VGG-16****:* VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford [9]. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of

over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. It improves AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 33 kernel-sized filters one after another [9]. Due to the model preset structure, VGG has a default output layer with 1000 nodes, our model used to make this it adaptive to the user input by adding a convolution layer to shrink it into the required size. While implementing the model, we insert another hidden layer that still serves the purpose of the shrinking size of the data, but makes the process goes more gradually. It proves that the extra layer helps in the prediction accuracy.

*4) **MobileNet**:* MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to lightweight deep neural networks [15], it is a small and fast model, with demonstrated effectiveness when applied to a wide variety of tasks including large scale geolocalization, fine-grained recognition, Face attributes and objects detection. We selected this model base on its outstanding static on image attributes recognition.

*5) **InceptionV3**:* InceptionV3 attempts to learn filters in a 3D space, with 2 spatial dimensions (width and height) and a channel dimension; thus a single convolution kernel is tasked with simultaneously mapping cross-channel correlations and spatial correlations [13]. The idea behind the Inception module is to make this process more efficient by explicitly factoring it into a series of operations that would independently look at cross-channel correlations and at spatial correlations. More precisely, the typical Inception module first looks at cross-channel correlations via a set of 1x1 convolutions, mapping the input data into 3 or 4 separate spaces that are smaller than the original input space, and then maps all correlations in these smaller 3D space [2], shown in Figure. 3. We selected this model base on its outstanding computational efficiency.

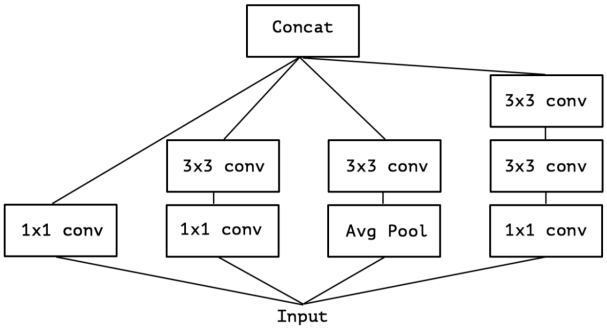



**Fig. 3:** A canonical Inception module for InceptionV3 [13]

*6) **InceptionRestNet**:* This hybrid inception module was inspired by the performance of the ResNet. InceptionRestNet has a similar computational cost to Inception-v3 but a dramatically improved training speed due to the residual connections [14]. We select this model since it's an improved model based on InceptionV3 and ResNet, which has a high training speed and outstanding classification performance.

*7) **Xception**:* Xception stands for "Extreme Inception", which is a convolutional neural network architecture based entirely on depthwise separable convolution layers [2]. In effect, the person who proposed Xception made the following hypothesis: that the mapping of cross-channel correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled [2]. This hypothesis is a stronger version of the hypothesis underlying the Inception of architecture. Compared to Inception V3, Xception shows small gains in classification performance on the ImageNet dataset and large gains on the JFT dataset, which means that Xception significantly outperforms Inception V3 on the larger image dataset. We select this model since it's an improved version of InceptionV3, with stronger classification performance on the large datasets.

### C. Training strategy

In the data training process, we took the approach by descending of learning rate after each epoch sets, inspired by the idea of deep learning. Namely, we save the model after each epoch of training and validate the performance model by a small group of validation data that did not participate in the training process. After a set of epochs of training, we picked models with the highest validation accuracy and the lowest loss to be the models of the next generation's training. As the generation grows, the learning rate of each generation descends to a smaller value, usually 1/5 or 1/10 for fine-tuning. A model usually has a stable loss and accuracy performance after 4 generations of training where each generation took about 10 epochs. This approach ensures multiple local minimums is first detected in a large span of directions by the large learning rate, approached with smaller learning rate, and finally fine-tuned till the local minimum is reached.
Here's an example of the model performance of descending learning rate training: Figure.4

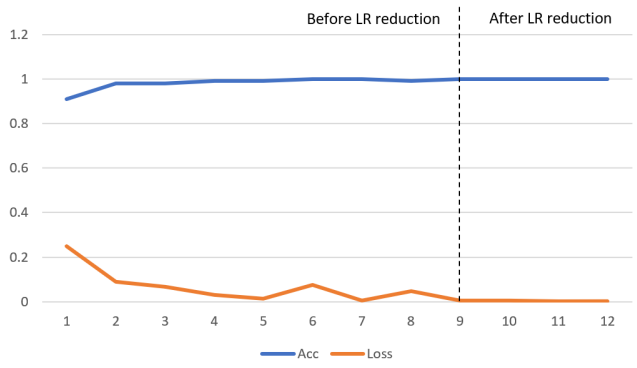



**Fig. 4:** The effect of descending learning rate training in validation loss and accuracy of VGG16

The model trained in this example shows how VGG16 model is trained and fine-tuned in the last two generations, we can see that the loss can be further decreased while the learning rate is adjusted.

## IV. RESULTS

After all model has been trained, we listed and compared their final accuracy (Figure.5) and loss (Figure.6)

After analyzing and comparing the overall performance, we have decided that the VGG16 model with modification is the most robust model we trained for this task. The score of this model on the submission to Kaggle is 97.3%, which is higher than any of others, or any stack voting result we have tried which combining this class with other classes.

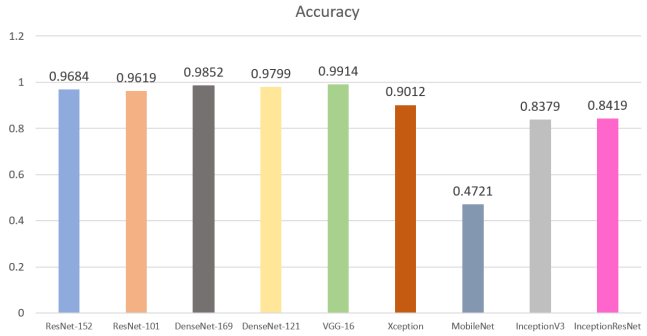



**Fig. 5:** Comparison between multiple training model final accuracy

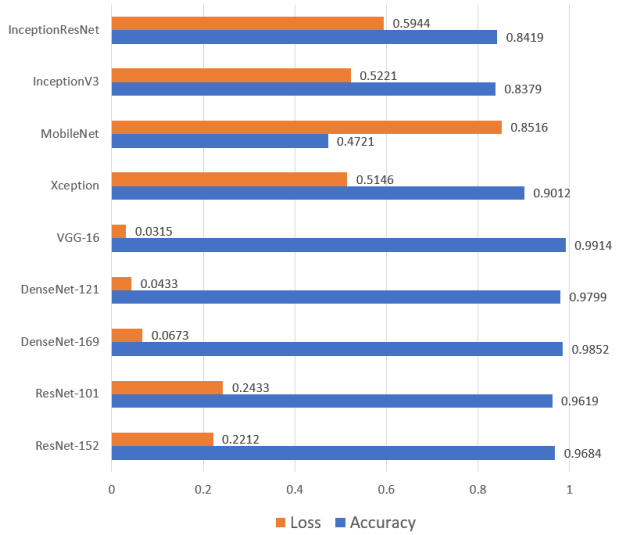



**Fig. 6:** Comparison between multiple training model final accuracy and loss

## V. DISCUSSION AND CONCLUSION

### A. Conclusion and Takeaway

In this project, we have discovered lots of useful knowledge for CNN model building and training. Generally the acquired dataset for CNN model training doesn't need any preprocess since some informative features might be lost during this process, but in this case, as we found the noise follows a fixed pattern, filtering out noise is proved really helpful in CNN model training. During the selection of the model, we've found that Since most of the common existing models use the image input channel as three, we can have either duplicated the grey-scale image by 3 times to make it fit the input size. Alternatively, we can also add another convolution layer to reshape the input. After comparing the result of each method, we found that the first method provides better performances. This is probably due to the fact that convolutional layers often lose some informative features of the graphic during transformation. Another Interesting phenomenon we have observed is that under similar levels of losses, each model performs consistently different from each other, which means some models have a better performance comparing to other models under the same level of loss. For example, DenseNet169 generally has a better performance than VGG16 at the same level of loss. Moreover, the rate of the convergence of the loss is different between models, as VGG16 converges faster than any others in training. This information can potentially become instructions on the persuading of a better image recognition model later.

### B. Further development

There are some potential solutions that can be used in this task that we haven't got time to tryout. For example, the preprocessing on a grey-scale image, we could spend time find a better way to fit the model with the 1 channel structure, as well as augmenting the final convergence process which shrinks the number of output nodes from 1000 to 10. Trying more starting points would also help in discovering some better local minimums. There are also some limitations of our model, for other MNIST sets where the informative data is not insulated from the noise, the temptation of filtering out the noise could cause the loss of the informative features during the process.

## VI. STATEMENT OF CONTRIBUTIONS

All members have made significant contributions towards this project. The distribution of work for each member is described as follows:

**Jiewen Liu** : Model train and modification, framework build-up, report writing.

**Han Zhou** : Model training and testing, stats organization, report writing.

**Hao Shu** : Data organization, model testing, report writing.

## REFERENCES


[1] S. Maji and J. Malik, "Fast and accurate digit classification," Tech. Rep. UCB/EECS-2009-159, EECS Department, University of California, Berkeley, Nov 2009.

[2] C. Francois, "Xception: Deep learning with depthwise separable convolutions," Tech. Rep. arXiv:1610.02357, arXiv.org, 2017.

[3] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. M. E. Sackinger, P. Simard, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," *online:* http://yann.lecun.com/exdb/publis/pdf/lecun-95a.pdf, 1995.
[4] K. Alex, S. Ilya, and H. Geoffrey, "Imagenet classification with deep convolutional neural networks," Tech. Rep. pages 1097–1105, Advances in neural information processing systems, 2012.
[5] D. Jia, D. Wei, S. Richard, L. Li-Jia, L. Kai, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," tech. rep., 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.
[6] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, M. Hasan, B. Esesn, A. Awwal, and V. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," tech. rep., arXiv.org, 2018.
[7] C. Feiyang, C. Nan, M. Hanyang, and H. Hanlin, "Assessing four neural networks on handwritten digit recognition dataset," *online:* https://arxiv.org/pdf/1811.08278.pdf, 2018.
[8] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep residual learning for image recognition," Tech. Rep. arXiv:1512.03385 [cs.CV], arXiv.org, 2015.
[9] S. Karen and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," Tech. Rep. arXiv:1409.1556, arXiv.org, 2014.
[10] H. Gao, L. Zhuang, M. Laurens, and W. Kilian, "Densely connected convolutional networks," *online:* https://arxiv.org/pdf/1608.06993.pdf, 2018.
[11] S. Mark, H. Andrew, Z. Menglong, Z. Andrey, and C. Liang-Chieh, "Mobilenetv2: Inverted residuals and linear bottlenecks," Tech. Rep. pp. 4510-4520, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[12] S. Christian, L. Wei, J. Yangqing, S. Pierre, R. Scott, A. Dragomir, E. Dumitru, V. Vincent, and R. Andrew, "Going deeper with convolutions," Tech. Rep. arXiv:1409.4842, arXiv.org, 2014.
[13] S. Christian, V. Vincent, I. Sergey, S. Jon, and W. Zbigniew, "Rethinking the inception architecture for computer vision," Tech. Rep. arXiv:1512.00567 [cs.CV], arXiv.org, 2015.
[14] S. Christian, I. Sergey, V. Vincent, and A. Alex, "Inception-v4, inception-resnet and the impact of residual connections on learning," Tech. Rep. arXiv:1602.07261 [cs.CV], arXiv.org, 2016.
[15] H. Andrew, Z. Menglong, C. Bo, K. Dmitry, W. Weijun, W. Tobias, A. Marco, and A. Hartwig, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," Tech. Rep. arXiv:1704.04861 [cs.CV], arXiv.org, 2017.