MNIST handwritten digit recognition with different CNN architectures

Lead Ming Seng
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
tp054850@mail.apu.edu.my

Brennan Bang Chen Chiang
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
tp050473@mail.apu.edu.my

Zailan Arabee Abdul Salam
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
zailan@apu.edu.my

Gwo Yih Tan
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
tp051104@mail.apu.edu.my

Hui Tong Chai
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
tp054680@mail.apu.edu.my

***Abstract*—Handwritten digit recognition has long been a popular research topic in computer vision and pattern recognition. Recognizing handwritten digits used to be challenging but thanks to many machine learning techniques nowadays, the problem is no longer. In this research, we looked into the MNIST database using fast.ai and trained the CNN ResNet-18 model to recognize handwritten digits. We then modified the architecture with different pre-trained models. For this work, we implemented five PyTorch's pre-trained models, which are GoogLeNet, MobileNet v2, ResNet-50, ResNeXt-50, Wide ResNet-50. The purpose of this paper is to reveal the most accurate architecture for handwritten digits recognition. Also, we provide comparisons of training time, top-1 error, top-5 error and model size on all five models.**

Keywords— Convolutional Neural Networks (CNN), CNN Architectures, Image Classification, Handwritten Digit Recognition

I. INTRODUCTION

Handwritten recognition is the ability of machines to recognize input handwritten by human. The variety of handwriting styles, spacing variations and handwriting inconsistencies all make it a much more challenging task for the machine. Nevertheless, machine learning models have evolved significantly in recent years and are still growing. Many state-of-the-art models are able to achieve high performance and a very high accuracy. With this success, this technology is now used in many ways i.e. reading postal address, bank check processing, form data entry, etc.

Convolutional Neural Networks (CNNs) are widely and conveniently used for these image recognition and classification tasks. CNN is a special type of Neural Network capable of taking in an input image, assigning importance to various aspects and being able to distinguish one from another. Recent research works have seen convolutional neural networks being applied for facial recognition, document analyses, speech detection and license plate recognition [1].

Different models are built and trained using convolution operation, but achieving high accuracy depends on many factors such as dataset used or network architecture. Our experiment set out to see how different architectures can affect the accuracy of handwritten digit recognition using the same dataset which is the MNIST dataset. MNIST is a large database of handwritten digits that contains 70,000 grayscale images, each of 28×28 pixels. Altogether there are 10 classes representing numbers from 0 to 9. The images of digits are normalized in size and centred which makes it an excellent dataset for evaluation. The train-test distribution differs for this project as 42,000 images are used in the training set and 28,000 images in the test set.

We first began by implementing the ResNet-18 architecture and trained the model using the training dataset. After getting the results, we then modified the architecture with predefined architectures from PyTorch. The models implemented are GoogLeNet, MobileNet v2, ResNet-50, ResNeXt-50 and Wide ResNet-50. All the models are trained on the MNIST and the CIFAR-10 dataset to see the which is the most accurate of all.

II. LITERATURE REVIEW

A. Similar projects

Many researches have been conducted on handwritten digit classification with different algorithms and classifiers. For Convolutional Neural Network, many models are also available to train the algorithm to achieve a better result. Some of the models include ResNeXt-50, ResNet-50 and GoogLeNet.

Saining et al [2] presented a simpler neural network focused on aggregating transformations of the same topology called ResNeXt based on VGG and ResNet. Cardinality, which is the size of the set of transformations, is increased and experimented on vs Depth and Width. ResNeXt-50 and ResNeXt-10 compared with the ResNet-50 and ResNet-101 models, have successfully reduced error rates by 3.2% and 2.3%. This shows that complex models that are deeper and

wider are not always better as it may take more time but return similar results. The problem of ResNet is diminishing feature reuse, which is it does not force to go through the residual block, and it can avoid learning. So, most of the block is not contributing or contributing a little to the final goal. Besides that, when comparing width to depth, the complexity of width is higher than the depth, so ResNet is made as thin as possible to increase the depth and have less parameters.

Zagoruyko and Komodakis [3] proposed the idea of decrease depth and increase width of residual network The parameters are tested to know how deep and how wide in ResNet to be able to optimize it. The Wide ResNet-40-4 that has fewer parameters is able to have a lower error rate compared to the 1001-layer Pre-Activation ResNet. The Wide ResNet-16-8 and Wide ResNet-28-10 achieve lower error rates and they are shallower and wider than Wide ResNet-40-4. The training time of shallower is shorter because the GPUs perform parallel computations. With dropout, the model is also able to have consistent gain and reduce overfitting.

Basri et al [4] observed and compared the performance contributed by different networks, the four models being discussed in the paper are AlexNet, MobileNet, GoogLeNet and CapsuleNet. While considering the result from normal data in the same condition, the error rate of relevant models are GoogLeNet (Inception V3) 7%, AlexNet 8%, CapsuleNet 8.7% and MobileNet 20.3%. After adding the augmentation dataset in the training process, the error rate had been reduced to AlexNet 0.99%, GoogLeNet 1.49%, CapsuleNet 7.76% and MobileNet 16.42%. For the computation time, AlexNet was fastest 1.14s, CapsuleNet 3.86s, MobileNet 12.52s and GoogLeNet 22.53s. This implies that the structure of models result in different performance and computation time, it also emphasizes the importance of augmented datasets.

B. Methodology / Approach

- *Dataset*

The dataset used in this paper is the MNIST database of handwritten digits. The dataset contains total 70,000 grayscales, each 28×28 pixels of size. Altogether there are 10 different classes, depicting the number 0 to 9. Normally the dataset is split into 60,000 and 10,000 for training set and test set respectively. The training dataset is to teach the model how every digit looks like by including the labels. Then the test dataset is used to test the model by feeding it only the images to let it predict data it has never seen before.

- *CNN*

Convolutional neural networks combine artificial neural networks with the recent methods of deep learning. They have been used for years in image recognition tasks, like handwritten digit recognition, which is addressed in this paper. CNNs are thought to be the first deep learning approach with robustness that is successful in using multilayer hierarchical structure networks. CNNs can reduce the number of trainable network parameters to improve the back-propagation algorithm deficiency of forward propagation networks.

They are particularly suitable for image processing and understanding due to the close link and spatial formation between the levels and can extract the rich correlative characteristics from the images [1].

- *Fast.Ai*

Fast.Ai, a deep learning library provides high-level components that can deliver state-of-the-art results quickly in standard deep learning domains and provides low-level components to be mixed and matched for building new approaches. Fast.Ai leverages the dynamism of Python language and the flexibility of PyTorch library. It offers a new type of dispatch system for Python with a semantic type hierarchy for tensors, a GPU-optimized computer vision library, a new data block API, etc. [5]. Overall, Fast.Ai is easily approachable and rapidly productive.

C. Conclusion / Recommendation

In a nutshell, CNN is widely used for image classification problems. With the ever-growing advancements of technology and complexity of datasets, more new and efficient CNN architectures are developed. Spoiled by the abundance of choice of CNN architectures, we need to pick the right architecture for the right problem. Therefore, comparisons of different architectures on datasets are done to evaluate which one suits best.

III. Algorithm Implementation

A. Data

The database utilized in the project is MNIST database, it has separated to training dataset, testing dataset and saved in train.csv, test.csv respectively. Both datasets contain plenty of grayscale images from number zero to nine. The images in the training dataset are labelled according to their classes, while images in testing dataset are not labelled.

B. Preparation

```
# the following three lines are suggested by the fast.ai course
%reload_ext autoreload
%autoreload 2
%matplotlib inline

# hide warnings
import warnings
warnings.simplefilter('ignore')

# the fast.ai library, used to easily build neural networks and train them
from fastai import *
from fastai.vision import *
```

Fig. 1. Importing libraries

```
# transforms-make some manipulation
tfms = get_transforms(do_flip=False)

data = ImageDataBunch.from_folder(
    path = TRAIN,
    test = TEST,
    valid_pct = 0.3,
    bs = 16,#1 batch 16 pictures
    size = 28,
    #num_workers = 0,
    ds_tfms = tfms
)

data.normalize(mnist_stats)
```

Fig. 2. Data transformation

Before starting the session, there is some preparation required to be done to make the process go smoothly. First of all, the programmer set up the environment by importing required libraries for further usage of functions. Due to fast.ai only accept images input, the data is stored in variables and reshape to the desired dimensions of images (28 x 28). Before load into the data bunch, the data has been transformed and normalized with the aid of mnist_stats function. In the transformation, the flip action has been restricted to prevent confusion.

C. Training

The implemented algorithm and architecture is CNN Resnet-18. The images data will be input and the model will

output the prediction result. Thus, the predicted result will be verified by the actual result and it will feedback to the model for improvement, which is called learning.

```
learn = cnn_learner(data, base_arch=models.resnet18, metrics=accuracy, model_dir="/tmp/models", callback_
learn.fit_one_cycle(cyc_len=5)
```

Fig. 3. Setting up the configurations of the architecture

The learner function cnn_learner has been set up according to the environment and the function fit_one_cycle is utilized as a support. It shows the condition of the training including loss, accuracy and time consumed.

D. Evaluation

```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_top_losses(9, figsize=(7, 7))
```

Fig. 4. Visualizing top losses

The object Classification Interpretation is created for evaluation and the images of top 9 loss are plotted. Confusion matrix are one of the suitable ways to present the data because it is easy to understand. From the figure and confusion matrix, it is clearly shown that the confusion among the digits is usually caused due to the similar pattern of the digit.

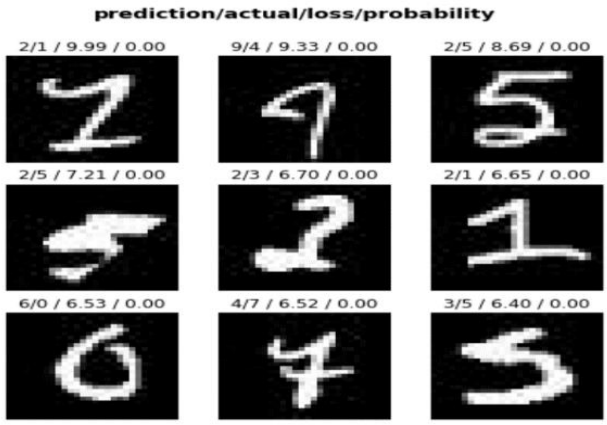

Fig. 5. Top-9 top losses

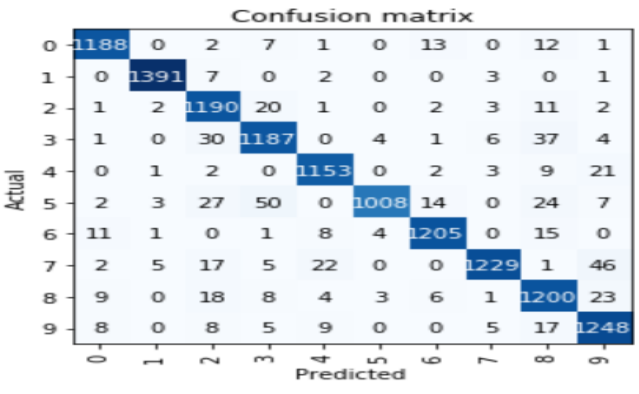

Fig. 6. Confusion matrix

E. Prediction

```
class_score, y = learn.get_preds(DatasetType.Test)
probabilities = class_score[0].tolist()
[f"{index}: {probabilities[index]}" for index in range(len(probabilities))]
class_score = np.argmax(class_score, axis=1)
```

Fig. 7. Evaluating performance

The testing data is fed to evaluate the performance of the learning model. The model will rate the possibility of each possible class and the class with highest probability will be the result of the images. It was found that with ResNet-18, the accuracy on the MNIST dataset average around 96% with a training time of 874 seconds. Original code from Kaggle [6].

IV. Result and Discussion

A. Discussion on implementation

The aim is to propose a more accurate and faster architecture for solving the MNIST handwritten digit image classification problem. We trained different architectures on the same dataset to compare and evaluate the time and accuracy. Using the Fast.Ai's default modification and some of the more popular PyTorch's default pre-trained models, we train with a one cycle policy along with 10 epochs for each architecture. The pre-trained models that we have used are GoogLeNet, MobileNet v2, ResNet-50, ResNeXt-50, Wide ResNet-50.

B. Experiments on MNIST Dataset

TABLE I. Comparisons of different architectures on the MNIST dataset. Top-1 and Top-5 error has been obtained by using the average over 3 runs

Model	Top-1 error (%)	Top-5 error (%)	Training Time (s)	Model Size (MB)
GoogLeNet	0.5317	0.0397	512	49.7
MobileNet v2	0.5754	0.0079	498	13.6
ResNet-50	0.6190	0.0159	510	97.8
ResNeXt-50	0.5794	0.0119	549	95.8
Wide ResNet-50	0.5278	0.0079	540	132.0

Table I shows that the performance of the models on the MNIST dataset appears to saturate. We argue that this is because of the complexity of the dataset being regarded as one of the simplest and is mostly used as a baseline for image recognition. Nevertheless, it is still possible to see which model is best suited for a smaller and simpler dataset such as the MNIST dataset.

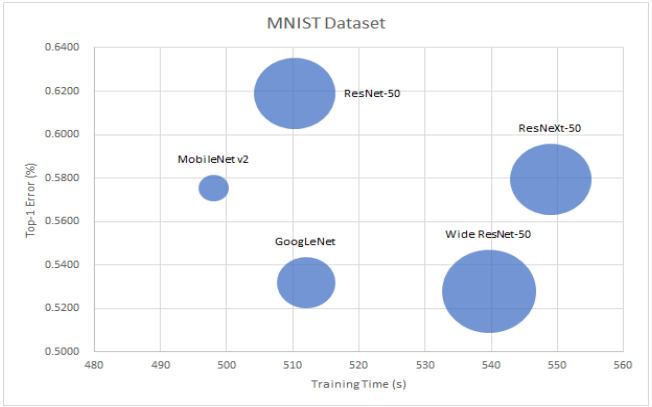

Fig. 8. Bubble Chart of MNIST Dataset comparing the Top-1 error, training time and the size of the model. The model size is represented by the size of the bubble.

The model with the lowest Top-1 error is Wide ResNet-50 at 0.5278% and a Top-5 error of 0.0079%. We also note that MobileNet v2 has achieved the 3rd best Top-1 error of 0.5754% and the best Top-5 error alongside Wide ResNet-50 at 0.0079% despite being 10x smaller than the model size of

Wide ResNet-50 at 13.6MB. It is also noted that MobileNet v2 has the fastest training time among the models at 498 seconds.

C. *Experiment on Cifar 10 Dataset*

Due to the saturated results from the MNIST dataset experiment, we conducted more experiments on more complex datasets using the same configurations from the previous experiment, notably the CIFAR-10 dataset.

TABLE II. COMPARISONS OF DIFFERENT ARCHITECTURES ON THE CIFAR 10 DATASET. TOP-1 AND TOP-5 ERROR HAS BEEN OBTAINED BY USING THE AVERAGE OVER 5 RUNS

Model	Top-1 error (%)	Top-5 error (%)	Training Time(s)
GoogLeNet	15.4500	0.7800	843
MobileNet v2	15.2780	0.5380	826
ResNet-50	19.0580	0.9440	850
ResNeXt-50	14.0460	0.5300	901
Wide ResNet-50	20.3620	1.0720	952

From Table II, we found that the model that performed the best is ResNeXt-50 with a Top-1 error of 14.0460% and a Top-5 error of 0.5300%. The model that performed exceptionally well was MobileNet v2 with a Top-1 error of 15.2780% and a Top-5 error of 0.5380% while being 7 times smaller in size as well as being 75 seconds faster in training time compared to ResNeXt-50, slightly outperforming GoogLeNet. It is also noted that MobileNet v2 has one of the fastest training times among the models.

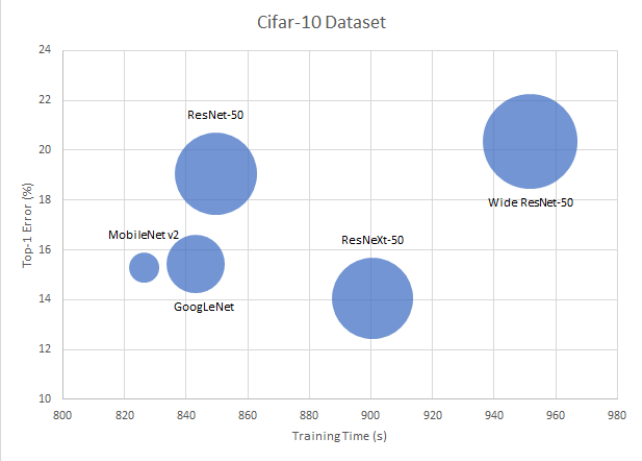

Fig. 9. Bubble Chart of Cifar Dataset comparing the Top-1 error, training time and the size of the model. The model size is represented by the size of the bubble.

V. CONCLUSION

The task of image recognition is still growing and developing as researchers alike develop progressively sophisticated neural networks. After conducting research, it became clear that the performance of models would differ from the task being performed on. Accuracy and error rate, while being important factors in determining the suitability of models on a task, are not the only factors that we look at. Training time plays an important role as a factor alongside accuracy and error rate as the complexity and size of a dataset grows, the more crucial training time becomes. The decision on picking the right model on solving the task at hand is best determined by these 3 factors, and we have thus concluded that based on our research, MobileNet v2 is the best among these 5 models for the MNIST dataset problem

REFERENCES

[1] X. Han., and Y. Li., "The Application of Convolution Neural Networks in Handwritten Numeral Recognition," International Journal of Database Theory and Application, Vol.8, No.3 (2015), pp.367-376.

[2] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He., "The Aggregated residual transformations for deep neural networks," ´ CVPR, 2016.

[3] S. Zagoruyko and N. Komodakis., "Wide residual networks. Arxiv," SERGEY ZAGORUYKO AND NIKOS KOMODAKIS: WIDE RESIDUAL NETWORKS 1, 2016.

[4] Basri, R. & Akter, M. (2020). Bangla Handwritten Digit Recognition Using Deep Convolutional Neural Network | Proceedings of the International Conference on Computing Advancements. [Online]. 2020. Doi.org. Available at https://doi.org/10.1145/3377049.3377077

[5] fast.ai. 2020. Welcome To Fastai | Fastai. [online] Available at: <https://docs.fast.ai/> [Accessed 31 August 2020].

[6] kaggle.com. (n.d.). Beginners guide to MNIST with fast.ai. [online] Available at: https://www.kaggle.com/christianwallenwein/beginners-guide-to-mnist-with-fast-ai