CS486 Project: Comparison of Image Recognition Algorithms

Mingkun Ni, Tianyi Zhang, Yuanhao Zhang
{m8ni, y2384zha, t296zhan}@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

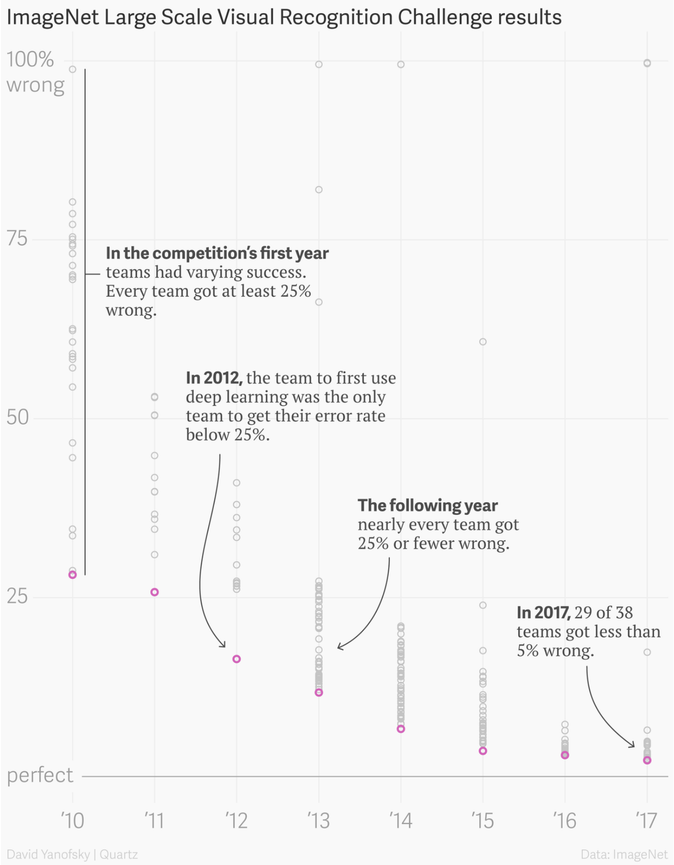
Abstract

Image recognition has been a popular field in Artificial Intelligence. Many real-world problems such as self-driving vehicles and medical diagnosis rely on image recognition. Researchers and programmers have came up with various neural network paradigms, such as Convolutional Neural Network(CNN), k-Nears-Neighbours, and Recurrent Neural Network(RNN). Both CNN and RNN are popular algorithms in classification problems. The main reason that RNN is chosen is due to its recurrent feature. RNN is able to discover the internal sequential relation between images. CNN is chosen since it is one of the most widely-used algorithm in the field of image recognition. Therefore, this paper will use Fashion-MNIST data set to run image recognition using CNN and RNN algorithms and compare the time complexity, memory usage, and test set accuracy of the results.

Introduction

Catalyzed by the creation of ImageNet, the image recognition technology has advanced remarkably in recent years. The image recognition technology is fundamental for many emerging applications such as autonomous vehicles, augmented reality, medical diagnosis, etc. As the technology becomes more mature, the image recognition market is rapidly expanding. According to Markets and Markets (MarketsAndMarkets 2017), the image recognition market is estimated to grow from 15.95 billion USD in 2016 to 38.92 billion USD in 2021. The fact that the market size would be more than doubled in five years shows the significance and potential of the image recognition technology.

Recent rapid development of the image recognition technology is largely thanks to the creation of an image dataset named ImageNet. The idea was conceived by a university professor Fei-fei Li in 2006, after she had realized all of the image datasets at the time were too limited in size and variety. She thought, to make advancements in image recognition, increasing the quality and quantity of training data might help more than improving on the algorithm alone. The ImageNet dataset was born in 2009, and an annual competition using the dataset as training data was started the next year. People quickly realized that their algorithms performed better when trained on ImageNet compared to when trained on traditional image datasets. Since ImageNet contains a large sample of images of a variety of objects, it accurately reflects the performance of algorithms in the real world. In the ImageNet competition, the error rate on the test set has decreased year over year (Figure 1), demonstrating the steady improvements of image recognition algorithms each year (Quartz 2017).



In the competition's first year teams had varying success. Every team got at least 25% wrong.

In 2012, the team to first use deep learning was the only team to get their error rate below 25%.

The following year nearly every team got 25% or fewer wrong.

In 2017, 29 of 38 teams got less than 5% wrong.

David Yanofsky | Quartz

Data: ImageNet

Figure 1: **Error Rate of Contestants of the ImageNet Competition over the years**

Image recognition is crucial for the development of many other technologies. Autonomous driving is a booming field with heavy reliance on image recognition. Image recognition enables vehicles to understand the surrounding environment in real time, thus allowing decision making and path finding. Developing an accurate and reliable image recognition algorithm for autonomous vehicles is no small feat. In the real world, a vehicle sees all sorts of objects, and the algorithm often has to classify these objects in motion and in real time. Some of the situations a vehicle would encounter cannot be anticipated by developers in a lab. For example, Google's self-driving car encountered possibly the weirdest road situation ever; it was stopped by a lady in a wheelchair chasing a bird on the road. (Self-driving cars 2017) Hence, accurate image recognition software that can identify all common objects in real time is necessary for the development of autonomous vehicles. Another field that has been gaining traction over recent years is augmented reality. Augmented reality needs quick and accurate image recognition because it superimposes information or visuals on the surroundings that users perceive. Augmented Reality has a broad range of applications including entertainment, medical applications, and military navigation and targeting. (Ronald T. Azuma 2006)

The aim of this project is to analyze the differences between two different image recognition algorithms, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The analysis makes use of the dataset MNIST Fashion. The purpose of this report is to illustrate the advantages and drawbacks of the two algorithms, so a better one can be chosen for image recognition tasks.

This paper compares the performance difference between two widely used neural networks over image recognition problem in detail. From our findings, CNN has huge advantage on time efficiency over RNN, but, unexpectedly, RNN is more space efficient.

Related Work

(Cios, Shin 1995; Xiao, Rasul, Vollgraf 2017) These two papers are focusing on the general topic and the data set of image recognition using neural network. In the first paper, the proposed neural network algorithm, namely image recognition neural network, is designed to classify objects through the attributes of the images. This algorithm will take a grey-scaled image as an input and return an appropriate output to which the algorithm believes the given image matches. As an extension of the first paper, the second paper discusses the dataset that we will use in this project, named Fashion-MNIST. It contains 70,000 grey-scaled images with a size of 28×28. This paper also states that Fashion-MNIST is considered as a better dataset for problem of image recognition using neural network compared to the old MNIST data set.

(Simonyan, Zisserman 2015; Hijazi, Kumar, and Rowen 2015; Razavian, Azizpour, Sullivan, Carlsson 2014; Chen, Han, Wang, Jeng, and Fan 2018; Hou, Wu 2016; Lo, Chan, Lin, Li, Freedman, Mun 1995) All of these paper discuss image classification and they all manage to train the model using CNN. CNN is widely used in pattern- and image-recognition problems because they have significant benefits over other neural network techniques. In general, CNN is a good choice for image recognition because of the following properties. First, it avoids the influence of image distortion on prediction result. Second, it uses less amount of memory compared to other Neural Network algorithm. Third, CNN reduces the number of parameters dramatically; thus, training time is significantly reduced. These papers describes the basic CNN algorithm and the hidden layers to be used for a CNN algorithm. Some of the other paper discuss different applications of CNN but in the field of image recognition and demonstrate how CNN is used in the real world to solve problems.

(Le, Jaitly, Hinton 2015; Zhang 2018) This paper talking about RNN and how RNN is used to solve problems. RNN is a powerful and natural method to map a sequence of input to a sequence of output. The second paper by Zhang discusses how CNN has been widely used in the field of image recognition, but there are still limitations of the current CNN image classification paradigm. For example, CNN cannot handle images with different sizes correctly. Thus, Zhang proposed a new paradigm named Scaled Recurrent Neural Network(SRNN) that is based on RNN embedded in the current paradigm, which is able to solve leftover problems of CNN.

Methodology

Due to fast development of artificial intelligence and neural network, multiple algorithms are applied in the field of image recognition. Performance can be different based on various implementation of algorithm and training environment. This project aims to analyze the performance of two different algorithms: Convocational Neural Networks(CNN) and Recurrent Neural Network(RNN).

The dataset we're using is Fashion-MNIST. Fashion-MNIST is a dataset of Zalando's article images, which consists of 60,000 examples and a test set of 10,000 examples(Han Xiao, 2017). Each image is a 28×28 grayscale image, and each has a label associated from different classes of clothing. This dataset is a great selection for image recognition since it differs from MNIST which is too easy and overused and doesn't represent modern computer vision tasks. In addition, Fashion-MNIST contains a large number of training and testing sets and should be able to yield a reasonable result in term of algorithm comparison.

The first algorithm is CNN, which is a fully connected network such that each neuron in one layer is connected to all neurons in the next layer. It simplifies complex patterns derived from training data. CNN has complex architectures and it is usually constructed with four types of layers: convolutional layers, pooling layers, non-linear layers, and fully connected layers (Hijazi et. al. 2015). The purpose of the convolutional layers is generally to extract the features of the input. According to Hijazi, the pooling layers are to reduce distortion and noise that may affect the process of image classification. Non-linear layers rely on activation function to match similar identification in each hidden layer. In the CNN algorithm we used in this project, the activation function used is ReLU. Lastly, the fully connected layer is the final layer that combines all the previous features and determines a target output based on the information (Hijazi et. al., 2015).

A simple and abstract mathematical formulation of CNN structure can be represented by the following:

$$x^1 \rightarrow w^1 \rightarrow ... \rightarrow x^{L-1} \rightarrow w^{L-1} \rightarrow x^L \rightarrow w^L \rightarrow z$$

This simple formulation demonstrates how CNN proceed forward layer by layer. x^1 is the initial input and it goes through process with weight w^1. This process will continue until it outputs x^L. In the CNN algorithm chosen for this paper, it will be using two built-in Conv2D layers provided by Keras (Simonyan, 2015). The first Conv2D layer creates a convolutional kernel with a 3-by-3 kernel size and uses ReLU as the activation function with the shape of input matching the Fashion-MNIST dataset. Following the first Conv2D layer, there is another Conv2D layer using ReLU activation function again. Then, the algorithm applies MaxPooling2D to find the max from a non-overlapping 2-by-2 block to filter out noise and distortion in the dataset while maintaining the features. A Dropout layer is used with a rate of 0.25 for the resulting neural network to have better generalization and be less likely affected by data overfitting. Then the algorithm will apply a Dense layer which is also known as a fully connected layer to connect and to combine previous features. and to produce an output. The following is sample code for Convolutional Neural Network obtained from Xiao:

```
num_classes = 10
input_shape = (1, img_rows, img_cols)
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation=’relu’,
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation=’
    relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation=’relu’))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation=’
    softmax’))
```

CNN is considered as a good choice for image recognition problems in general. It is widely applied in related problems; also, as stated in related work section, it has several known advantages regarding to image recognition problem, such that the influence of image distortion on prediction result will be avoided. Thus, we will evaluate its performance in this paper.

The second algorithm is RNN. RNN uses neural network to map an input sequence to an output sequence. In another word, it is designed to solve problems in term of predicting sequence. RNN is generally a good choice when dealing with classification problem. Since this report is working on image classification, RNN is chosen as one of the algorithm.

Although RNN is usually hard to train, it is able to store information in the form of internal states and it is able to learn sequential data. The reason for this is that, after feeding one datapoint, it will generate a state that contains some useful information from this datapoint, and use it as input when feeding the next datapoint. Thus, neural network will understand the sequential relation between datapoints, and generate its output accordingly.
Figure below is a mathematical representation of RNN retrieved from Marmot (Marmot 2019) :

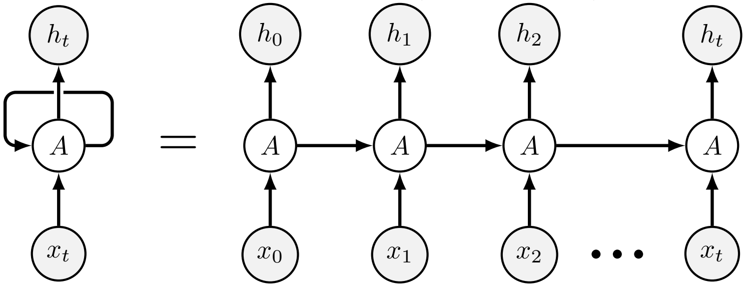


Figure 2: **Mathematical Representation of RNN**

RNN algorithm chosen in this paper will use one SimpleRNN Layer with 100 hidden units. It selects random normal distribution with standard deviation 0.001 as its initial weight distribution or kernel initializer and identity matrix with gain 1.0 as its recurrent initializer. Then, the algorithm will apply a Dense layer or fully-connected layer with 10 classes and activation function SoftMax. The following is sample code for Recurrent Neural Network:

```
num_classes = 10
hidden_units = 100
model.add(SimpleRNN(hidden_units,
kernel_initializer=initializers.RandomNormal
    (stddev=0.001),
recurrent_initializer=initializers.Identity(
    gain=1.0),
activation=’relu’,
input_shape=x_train.shape[1:]))
model.add(Dense(num_classes))
model.add(Activation(’softmax’))
```

RNN is considered in this paper because of its recurrent feature. RNN is able to discover the internal sequential relation between images. We believe that the history information or state will help us classify current image and improve recognition performance.

Two algorithm can be compared in lots of different aspects. For neural network algorithms, two aspects are relatively important indicators of performance: Accuracy and Resource Usage. For accuracy, this project will test the prediction accuracy using different values of parameters for each algorithm. For example, we will test the influence on accuracy if we change the number of training epochs each algorithm will run in test. Related result will be concluded by comparing two algorithm’s distribution of prediction accuracy. For resource usage, given the same training set and parameters, time efficiency and space efficiency will be considered and compared. For example, we will test the difference of time and memory used by running two algorithms with the same number of epochs.

We set up two test environments with Windows 10 operating system and MacOS Mojave. To test these two algorithm efficiently, we use TensorFlow with GPU support on Windows 10 and TensorFlow with CPU on MacOS while running python in bash terminal. Following are the configurations of Windows 10 and MacOS system respectively:

```
System: Windows 10
CPU: i7-6700hq Memory: 8G
GPU: GTX 960m
Video Memory: 2G
TensorFlow: tensorflow-gpu v1.14.0
CUDA: v10.0.130
cuDNN SDK: v7.6.1.34
Python: v3.7.3 64bit
```

For your information, CUDA and cuDNN SDK are softwares required to run TensorFlow with GPU support on Nividia graphic card in Windows 10 testing environment.

```
System: MacOS Mojave version 10.14.4
CPU: i5-7267u Memory: 8G
GPU: Intel Iris Plus Graphics 655
Video Memory: 1536MB
TensorFlow: tensorflow 1.13.1
Python: v3.7.3 64bit
```

Result

After running both CNN and RNN algorithms on the Fashion-MNIST dataset, we have obtained the following result. In this section, we will compare and contrast the results obtained for CNN and RNN in three aspects, test accuracy, time efficiency, and memory usage.

The first important performance indicator of a neural network is its test accuracy on test set. We run both algorithms with 30 epochs on both test environments, Windows and Mac OS. The graph below shows the change of validation accuracy over number of epochs. Test accuracy of CNN finally reaches 0.9274 on Windows environment and 0.9303 on MacOS environment Instead, test accuracy of RNN only reaches 0.5328 on Windows environment and 0.6083 on MacOS environment. By shown in Figure 3, Figure 4 and data above, we can clearly observe that CNN has significant advantage on accuracy over RNN when we run both algorithms with the same number of epochs. On the other hand, test accuracy of RNN can hardly reaches 0.7335 even after running 200 epochs, as shown in the Figure 5.

This result is foreseeable according to the features of each neural network. For CNN, the design of CNN algorithm is suitable to run stochastic or random data. Its convolutional layer and MaxPooling layer will extract the maximum of the characteristic of picture and use this information to predict. Also, its several Dense layers largely helped update the neural network after each epochs. This results in a quick increase of validation accuracy of CNN while the algorithm runs more epochs. However, for RNN, the historical information cannot be clearly utilized since data are shuffled at the first place. This means RNN will lost its advantage on utilizing past information to help current prediction. This results in a frequent fluctuation and slow increase of validation accuracy of RNN while the algorithm runs more epochs. However, after being trained with a significant large number of epochs, RNN can still reach a better test accuracy.

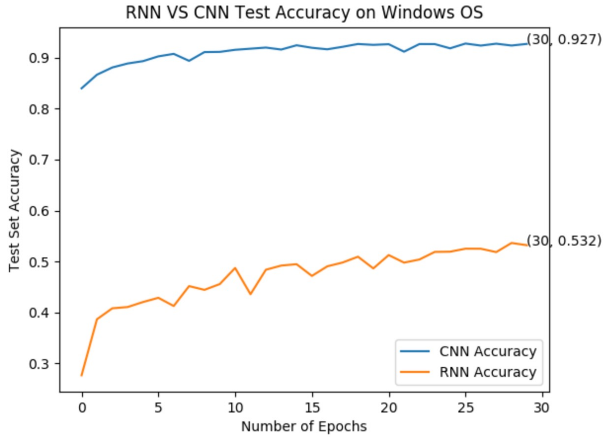


Figure 3: **RNN VS CNN Test Accuracy on Windows OS**

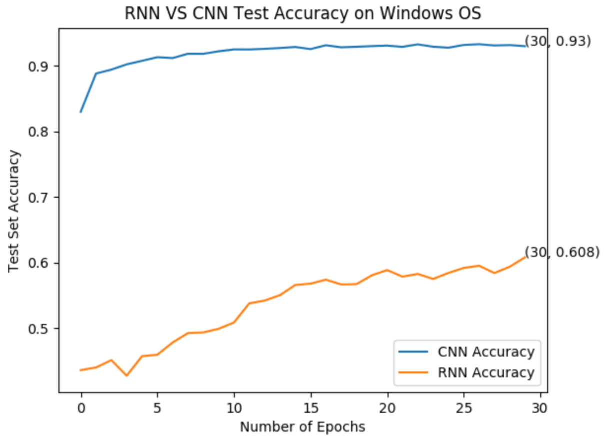


Figure 4: **RNN VS CNN Test Accuracy on MacOS**

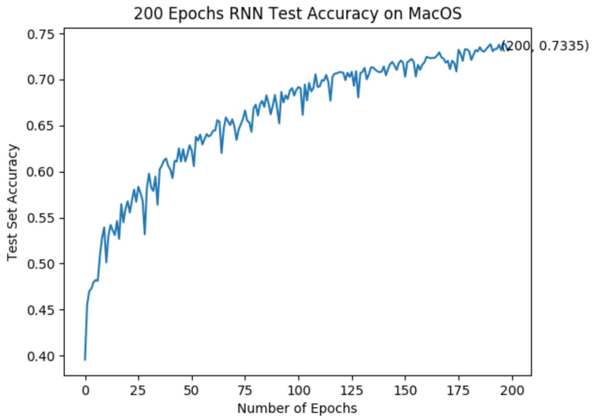
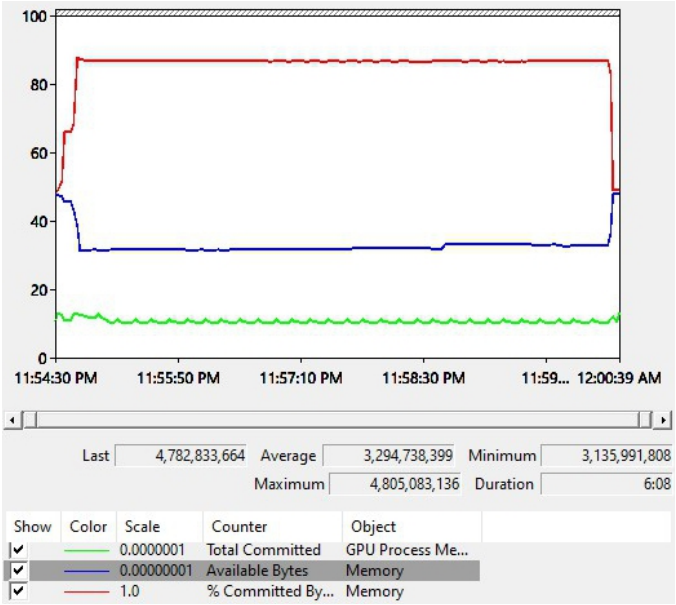


Figure 5: **Test accuracy of RNN after running 200 epochs**

The second important performance indicator is time efficiency. CNN takes 2555 seconds to be trained with 30 epochs on MacOS environment using TensorFlow-CPU, and approximately 80 seconds per epoch. Also, with support of TensorFlow GPU, CNN can finish training in only 350.45

seconds in total and around 11.67 seconds for each epoch on Windows environment. On the other hand, RNN has significant disadvantage on time efficiency. On MacOS environment, it takes 12150 seconds to run 30 epochs using TensorFlow-CPU, with each epoch around 405 seconds; on Windows environment, it takes 9031.26 seconds to run 30 epochs, which is clearly slower than CNN. To reach a useable accuracy using RNN, we trained it with 200 epochs on MacOS environment, and it takes nearly 18 hours to train, which is significantly inefficient in time scale.

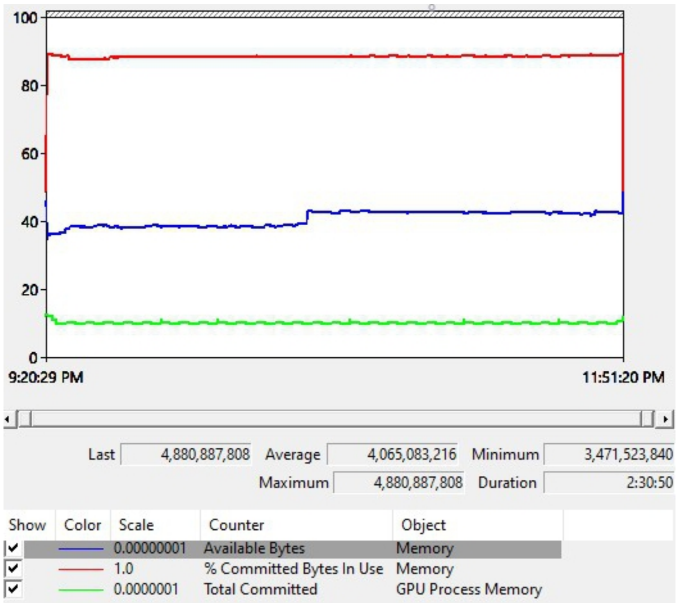
The third performance indicator is space efficiency or memory usage. Before we start analyze, there are three points need to know about Figure 6 and Figure 7 below. First, since we are testing both algorithm using TensorFlow with GPU, GPU memory used is also recorded. Second, to fit GPU memory usage(unit Byte), Available bytes (unit Byte), and percentage of Memory usage(unit 1 percent), data are scaled differently. Detailed information is listed in the legend. Third, since we start monitoring the data before we run the algorithm and end it after the algorithm finishes, there are some useless data at the begin and end of graph. Please treat them as the baseline of device, which is the regular memory usage without running other program. The baseline of memory occupation is around 48 percent.



Last 4,782,833,664 Average 3,294,738,399 Minimum 3,135,991,808
Maximum 4,805,083,136 Duration 6:08

Show	Color	Scale	Counter	Object
☑	🟩	0.0000001	Total Committed	GPU Process Me...
☑	🟦	0.00000001	Available Bytes	Memory
☑	🟥	1.0	% Committed By...	Memory

Figure 6: **Memory Data While Running CNN**



Last 4,880,887,808 Average 4,065,083,216 Minimum 3,471,523,840
Maximum 4,880,887,808 Duration 2:30:50

Show	Color	Scale	Counter	Object
☑	🟦	0.00000001	Available Bytes	Memory
☑	🟥	1.0	% Committed Bytes In Use	Memory
☑	🟩	0.0000001	Total Committed	GPU Process Memory

Figure 7: **Memory Data While Running CNN**

Figure 6 shows the memory usage of device while we run CNN or RNN over time. Note that numbers listed below reflects the percentage of total memory which includes all activities runs on computers. As we can observe from the Figure 6 and Figure 7, CNN has a 86.47 percent average memory usage, and it keeps around 86 percent for the most of time. The maximum usage of memory reaches 88.02 percent.On the other hand RNN has an average memory usage 88.36 percent and it keeps around 88 percent for the most of time. The maximum usage of memory reaches 89.37 percent. However, after we ran 14 epochs, available bites reserved for process used to run RNN program has a 7 percent increase. Regarding to the runtime of RNN, this will save memory in long running time. For GPU memory usage of CNN shown in Figure 6, the mean usage is around 101.72MB, and its maximum is around 124.01MB. For GPU memory usage of RNN shown in Figure 7, the mean usage is around 95.64MB, and its maximum is the same as the maximum in CNN cases.

As a conclusion, CNN has a better performance over RNN in all three aspects including test accuracy, time efficiency, and memory usage. Data above shows that CNN will become a better option of neural network algorithm to address image recognition problems in the most cases. However, because of the limitation of test device, we believe that on device with better performance, RNN will perform nearly the same as CNN. Thus, in the environment that device performance is comparatively better, RNN could be a backup option.

References

MarketsAndMarkets 2017 *Image Recognition Market*. Retrieved From: *https://www.marketsandmarkets.com/Market-Reports/image-recognition-market-222404611.html*

Dachis, A. 2019. *Google's EfficientNet Offers up to a 10x Boost in Image Analysis Efficiency*. Retrieved From: *https://www.extremetech.com/computing/ 292272-googles-efficientnet-offers-up-to-a-10x-boost-in-image-analysis-efficiency*

Figure 1: Quartz 2017 *Image Recognition Using Scale Recurrent Neural Network*. Retrieved From: *https://arxiv.org/pdf/1803.09218.pdf arXiv:1803.09218*

Google's self-driving car avoids hitting woman chasing a bird 2017 Retrieved From: *https://www.theguardian.com/technology/video/ 2017/mar/16/google-waymo-self-driving-car-video-woman-bird*

Ronald T. Azuma 2006 *A Survey of Augmented Reality*. Retrieved From: *https://www.mitpressjournals.org/doi/abs/ 10.1162/pres.1997.6.4.355?cookieSet=1*

Figure2: Marmot 2019 *Mathematical Representation of RNN*. Retrieved From: *https://tex.stackexchange.com/a/494148*

Xiao, H; Rasul, K; Vollgraf, R. 2017. *Fashion MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*.

Hijazi, S; Kumar, R; Rowen, C; IP Group; Cadenc 2015. *Using Convolutional Neural Networks for Image Recognition. arXiv:1708.07747*.

Cios, K.; Shin, I; 1995. *Image Recognition Neural Network: IRNN*.

Simonyan, K.; Zisserman, A.; 2015. *Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556*.

Razavian, A S.; Azizpour, H.; Sullivan, J.; Carlsson, S. 2014. *Very CNN Features off-the-shelf: an Astonishing Baseline for Recognition*. IEEE.

Chen, Y N.; Han, C C.; Wang, C T; Jeng and Fan. 2018 *The Application of a Convolutional Neural Network on Face and License Plate Detection*. IEEE.

Lo, S.; Chan, H.; Lin J.; Li H.; Freedman, M.; Mun, S.; 1995 *Artificial Convolutional Neural Network for Medical Image Pattern Recognition*. Elsevier Science Ltd.

Le, Q.; Jaitly, N.; Hinton, G.; 2015 *A Simple Way to Initialize Recurrent Neural Network of Rectified Linear Units. arXiv:1504.00941*

Hou, L.; Wu, Q.; 2016 *Fruit Recognition Based On Convolutional Neural Network*.

Xiao, H; Rasul, K; Vollgraf, R. 2017 *Fashion-MNIST*. Retrieved From: *https://github.com/zalandoresearch/fashion-mnist*

Dong-Qing Zhang 2018 *ImageNet Large Scale Visual Recognition Challenge Results*.