Machine Learning Engineer Nanodegree

Capstone Project

Hussein Al-barazanchi
July 4th, 2017

I. Definition

Project Overview

Most of the research on image classification using CNN (Convolutional Neural Networks) concentrate on enhancing the accuracy in the classification task. However, the common CNN architectures need a lot of memory and computation to run and also require moderate storage size. These reasons make CNN very difficult to use on devices with limited resources such as smartphones. Just few research were dealing with the issues of CNN related to the memory and storage requirements. One of the early attempts to compress CNN was by Denton [1]. After that, Han [2] developed a popular technique known as deep compression. Also, Han [3] presented another method less popular before deep compression. The mentioned methods try to compress CNN models by reducing the floating point operations.

The other direction of research is trying to compress CNN by reducing the number of parameters used in the network. The famous network in this direction is SqueezeNet developed by Andola [4]. SqueezeNet depends on architectural design choices to minimize the number of parameters by using convolutional filters of reception field 1. Also, it depends on squeezing and expanding the number of filters in the consecutive convolutional layers in a module named Fire. SqueezeNet was able to achieve an accuracy very similar to AlexNet network [5]. The number of parameters used in SqueezeNet is less than AlexNet by 50 times.

Problem Statement

One of the problems with SqueezeNet is the required time for training. SqueezeNet is slow to train comparing with the small number of parameters it has. So, I will work on improving the architecture of SqueezeNet to decrease the training time and also to improve the accuracy resulted.

After studying the architecture of SqueezeNet, I think the reason for slow training is the branching in the Expand layer in the Fire module and also the use of the concatenation layer. So, I will try to fix this by removing the concatenation layer and one of the Expand layers. This will let data flow straightforward without going to branches so it will be faster. In this case, the number of parameters will be decreased and hence a drop in the accuracy is expected. To fix the possible accuracy dropout, I will increase the number of parameters in the Squeeze layer.

Metrics

As the task is image classification, the evaluation metrics used is accuracy score. Where accuracy is calculated by computing the number of correctly predicted labels to the total number of images. This score will be denoted as Top-1 accuracy. Another measure of accuracy is Top-5 accuracy. Where accuracy will be computed as the number of labels predicted correctly within the top 5 most likely labels produced by the network to the total number of images.

The selection of these two metrics is based on the goal of this project which is to improve SqueezeNet network. And to benchmark the suggested improvements, I have to use the same metrics used by the SqueezeNet to measure the improvement rate. These two metrics as explained above are Top-1 and Top-5 accuracies. These two metrics best express the performance as the task image classification.

II. Analysis

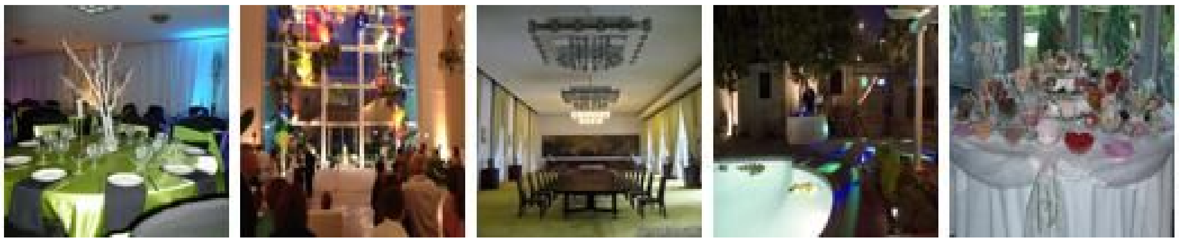
Dataset Exploration

The authors of SqueezeNet used a large scale image dataset which is ImageNet in their experiments. I do not have access to ImageNet at the current time. So, I need to test my proposed improvement on SqueezeNet on large scale image dataset too. For more robustness of results, I decided to use two large scale image datasets. Below sections cover a brief overview of the selected datasets.

The first dataset is MIT places scene 205 [6]. The Scene 205 contains 205 different scene category with 2.5 million image. Each image is associated with a specific label. The second dataset that will be used is MIT Places 2 [7]. This dataset contains 365 different category with about 1.8 million images. The images in the two datasets come in different resolutions. However, this will be solved in the preprocessing section.

Aquatic Theater

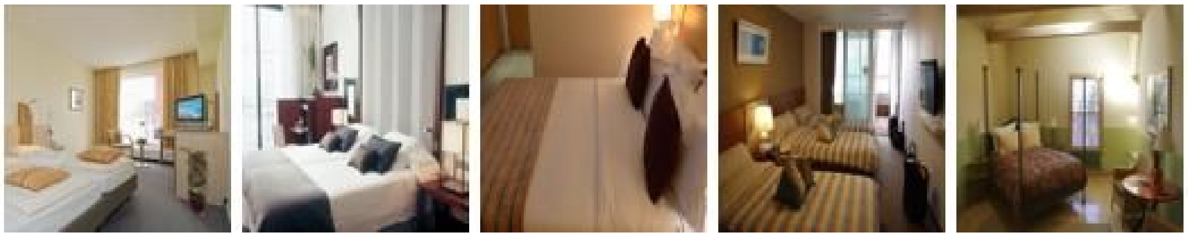
Banquet Hall



Cathedral Outdoor

Forest Path

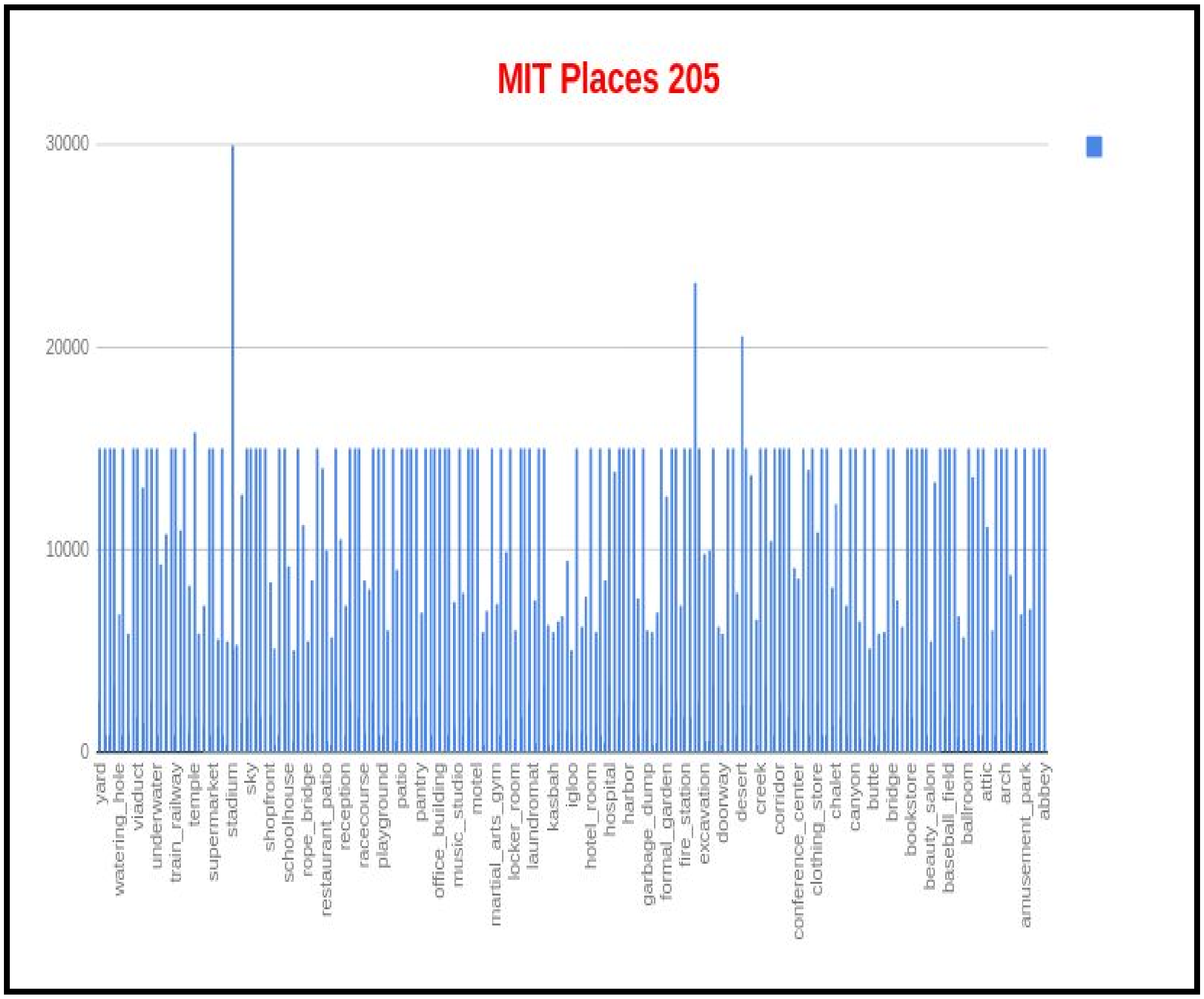
Hotel Room

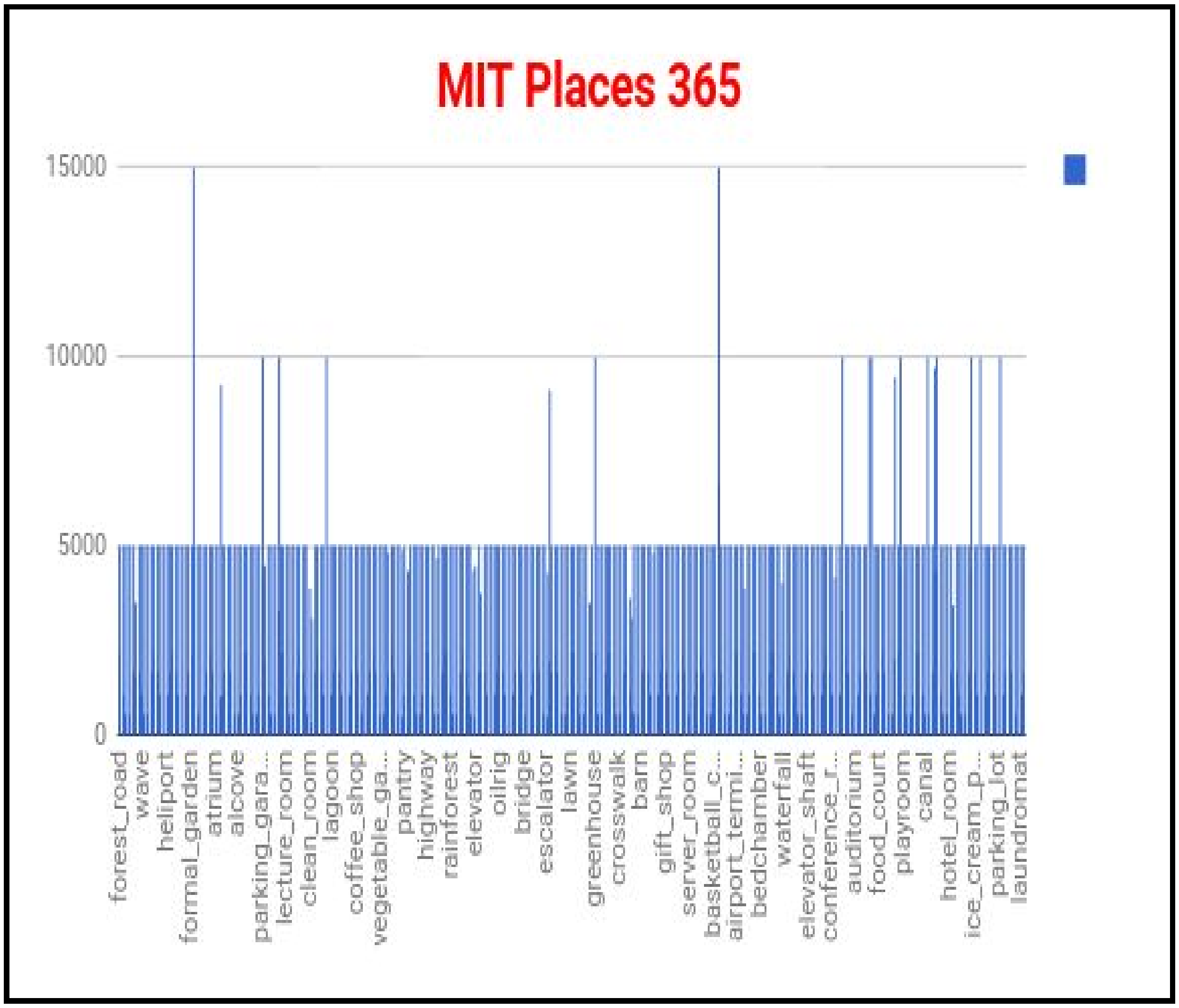


The number of images across labels is not equal so they are not balanced datasets. The division of the data into training and testing will take into consideration the unbalanced nature of the data. The figure above gives an overview about the dataset. The image samples selected randomly from few different categories for the demonstration purposes. The selected images and categories are from MIT Places 205 dataset.

Exploratory Visualization

The following two figures illustrate the distribution of images across the categories of MIT Places 205 and MIT Places 365 consecutively. The test set for MIT Places 205 consists from 20500 image where 100 image is assigned for each category. The test set for MIT Places 365 consists from 36500 image where also 100 image is assigned for each category.





Benchmark

The following table gives the Top-1 and Top-5 accuracies on MIT Places 205 and MIT Places 2. The results show the performance of SqueezeNet on the Two datasets. These results will serve as the benchmarking threshold to compare the improvement of SqueezeNet with.

	MIT Places 205		**MIT Places 2**	
	Top-1	**Top-5**	**Top-1**	**Top-5**
SqueezeNet	48.1415	78.7316	47.4459	78.1687

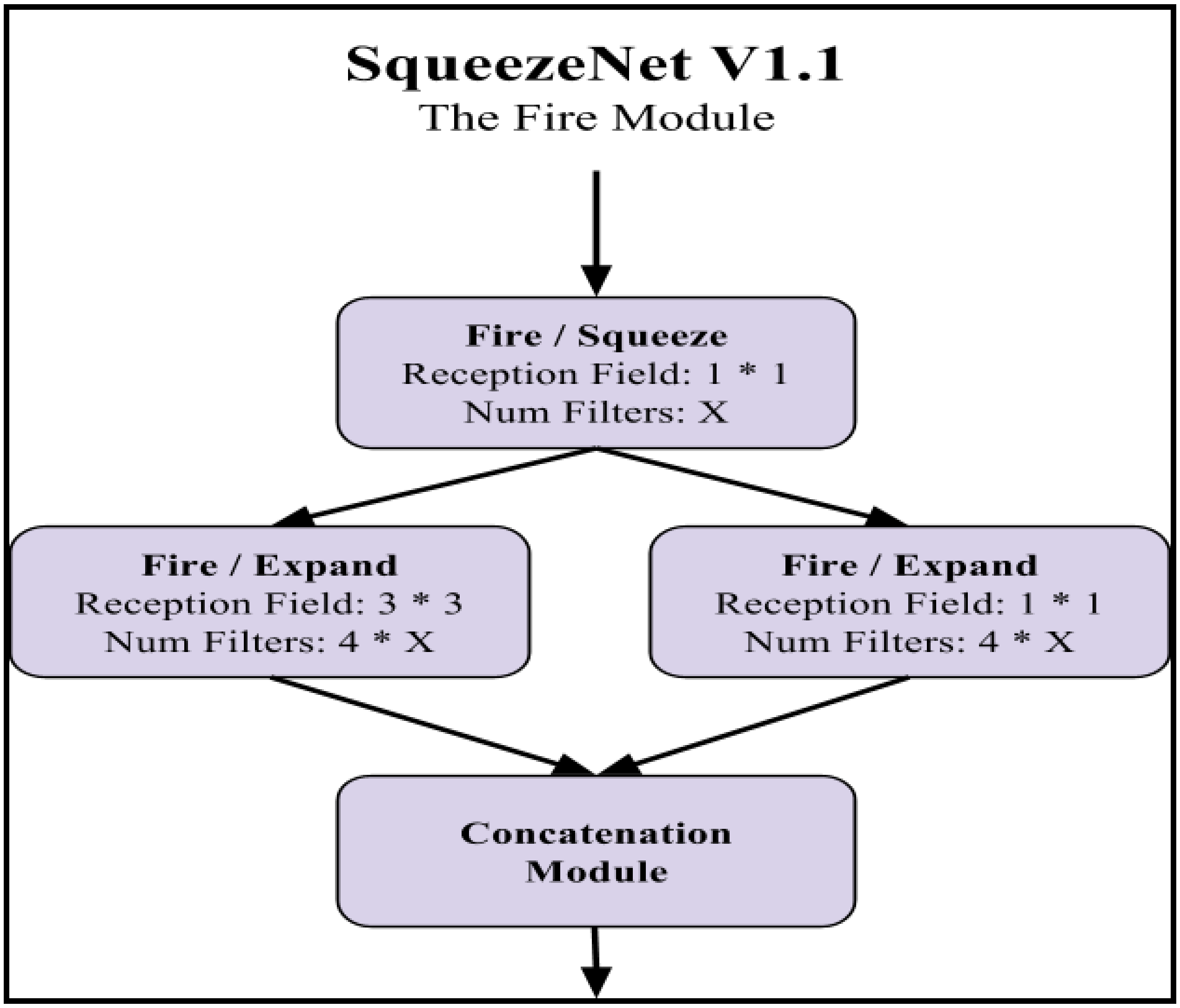
III. Methodology

Data Preprocessing

- The used framework is Nvidia Digits
- The used underlying deep learning library is Caffe
- All images will be resized to 256 * 256
- The mean pixel for each color channel will be subtracted from all images
- 80% of the images will be used for training and 20% for testing
- The division of training and testing will be as a percentage in regard to the total number of images per label
- All images will be cropped randomly to be 227 * 227

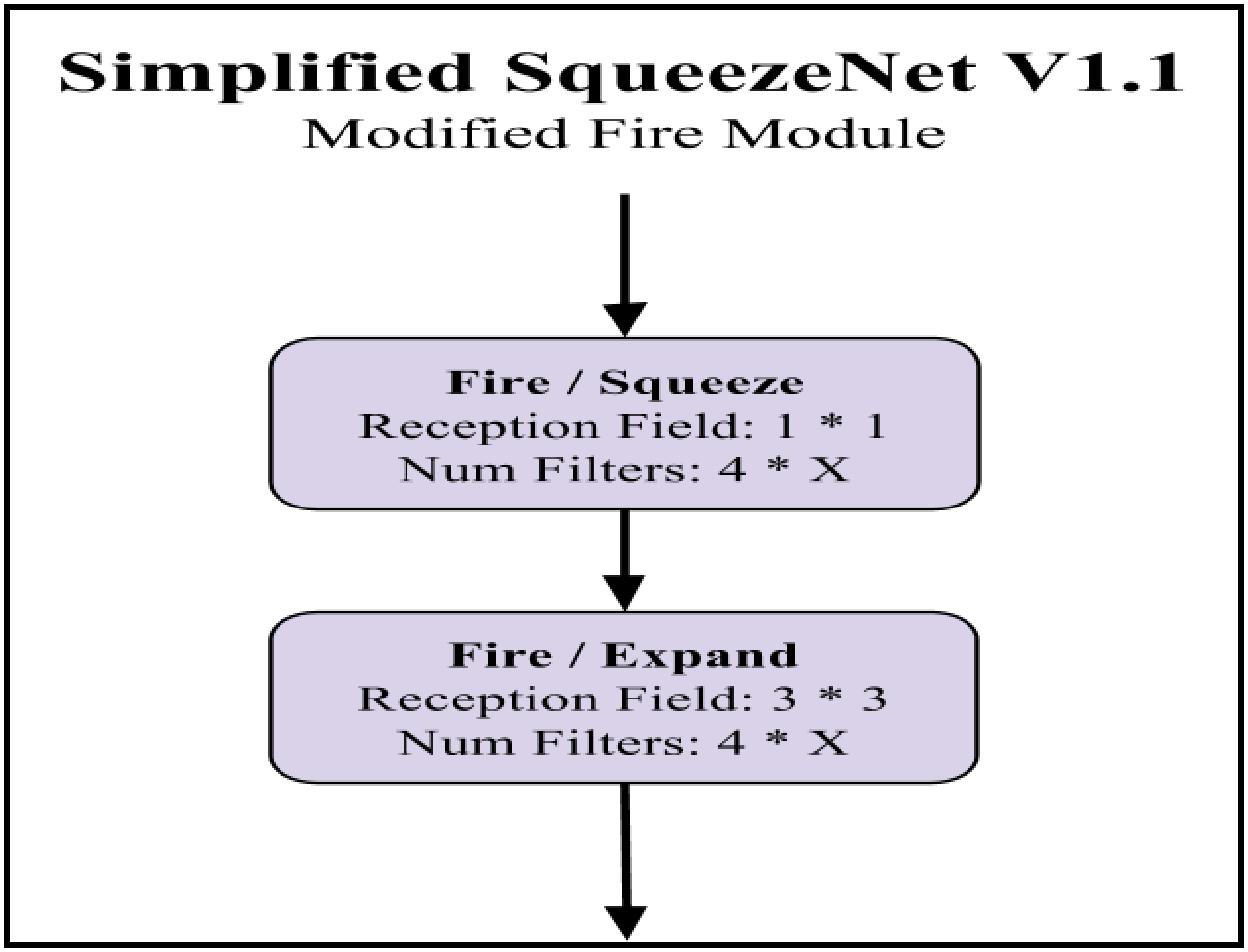
SqueezeNet

The fundamental idea behind SqueezeNet is to reduce the number of parameters through architecture design choices. The intuitive solution is to reduce the number of convolutional kernels of reception field 3*3. To implement this technique, they suggested a basic module named Fire. Fire module is inspired from Google Inception module [8]. The fire module consists from 3 parts. The first part is Fire/Squeeze which consists from convolutional layer of reception field 1. This layer is followed by the second part which is two parallel convolutional layers named Fire/Expand. The first layer reception field is 3*3 while the other layer is 1*1. The outputs from these two layers are combined through the third part which is a module named concatenation. The CNN network will be constructed from this basic module. The figure below shows a basic illustration of the Fire module.



Proposed Simplified-SqueezeNet

The slow training speed of SqueezeNet is because the use of two parallel convolutional layers in the expand level followed by the concatenation module. So, my suggested solution to this issue is to remove the Fire/Expand layer of reception field 1*1 so there will be no need for the concatenation module. The network will start training faster. However another issue will arise which is the performance will degrade because the number of layer is reduced. To solve this, I will increase the number of filers in the Fire/Squeeze layer. I tried two settings for the number of parameter in the Fire/Squeeze. The details are described in the results section.



Training and Testing

1. I used the SqueezeNet V1.1 network provided by its author from this repository [9]
2. I trained it on the MIT Places 205 and MIT Places 2.
3. The top-1 and top-5 accuracies reported in the Benchmark section.
4. I will use my suggested modifications mentioned in the solution statement above.
5. It is very expensive (my budget is very limited) to try many hyper-parameters configurations. So, I will conduct two experiments on each dataset using the new architecture as follow:
    A. Use the same hyper parameters provided by SqueezeNet V1.1.
    B. Increase the number of filters in Fire/Squeeze to be Equal to the number of filters in Fire/Expand.
6. The top-1 and top-5 accuracies will be reported and compared with the original networks.

IV. Results

The authors of SqueezeNet did not benchmark their network on MIT Places 205 or MIT Places 2. So, I run the SqueezeNet V1.1 on the two datasets and obtained the results shown in the Benchmark section and also in the following table. The second set of experiments conducted using the proposed new network with using the original hyperparameters provided by SqueezeNet. The results of this set of experiments denoted as A. Simplified SqueezeNet and shown in the following table. The training speed of this network was about 8 times faster than the original SqueezeNet. However it is clear the degradation in the accuracy.

To improve the accuracy, I increased the number of convolutional kernels in the Fire/Squeeze to be equal to the number of kernels in the Fire/Expand. This will slow the training but it will keep being faster than the original SqueezeNet. The results of this configuration is reported in the following table under B. Simplified SqueezeNet. The training speed of this configuration is about 4 times faster than the original SqueezeNet. The accuracy gain is very clear and it is higher than the accuracy of SqueezeNet and A. Simplified SqueezeNet on both datasets.

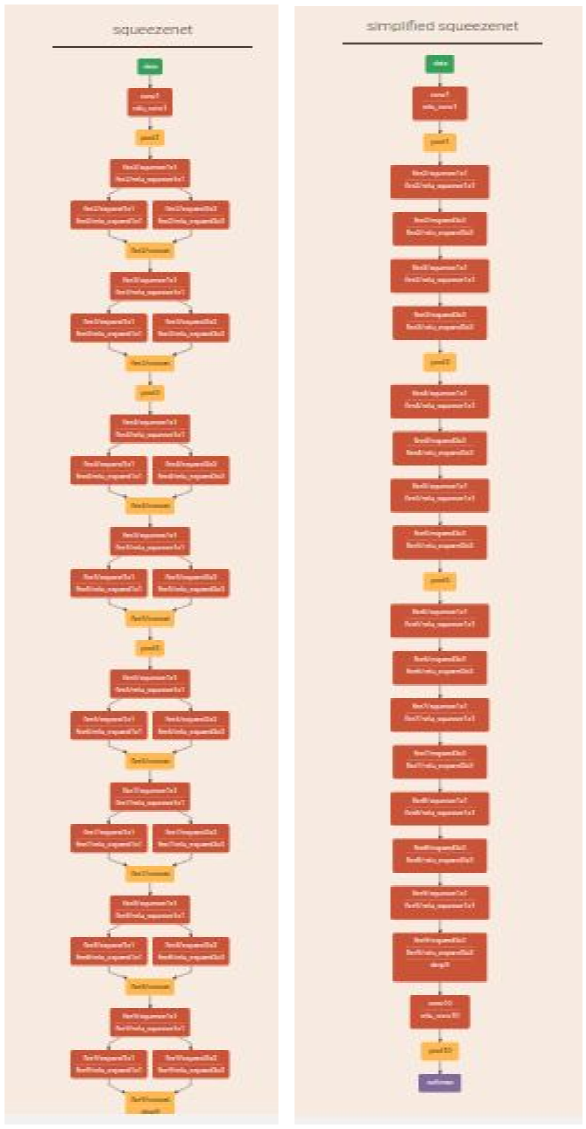
	MIT Places 205		**MIT Places 2**	
	Top-1	**Top-5**	**Top-1**	**Top-5**
SqueezeNet	48.1415	78.7316	47.4459	78.1687
A. Simplified SqueezeNet	45.3319	76.1792	45.2360	76.1364
B. Simplified SqueezeNet	50.9753	81.3373	48.7352	79.357

The results obtained are **robust** for two reasons. First the test data used has equal number of images for all classes. This means all categories in the two datasets got the same importance. So if the network overfit on categories that has large number of images in the training data, it will have to generalize well to get good results on the test data. This makes the result obtained from testing robust to overfitting. Second, the suggested network passed the original SqueezeNet in the Top-1 and Top-5 on both datasets on training data and test data. This means that the network is generalizing better than the original network. Getting higher Top-1 and Top-5 on two data sets on training and testing data means the suggested network performed better in all 8 results. This means the results are **robust** enough to be considered reliable.

The trained networks on both datasets for SqueezeNet and B. Simplified SqueezeNet are open sourced on the following link: https://github.com/NidabaSystems/Simplified_SqueezeNet

V. Conclusion and Future Work

SqueezeNet network proposed a new direction in compressing convolutional neural networks. It is performance is similar to the performance obtained by AlexNet network [5] with 50x times fewer parameters. With the small size of SqueezeNet, the training stage should be fast. However from experiments, this is not the case. I tried to address this issue in this project here. My suggested modification achieved increasing in the training speed and the accuracy. However, it came with a drawback too.



The drawback of Simplified SqueezeNet is the size of the network is 3 times bigger than the original SqueezeNet V1.1. The reason for this increase in the model size is the increase that I employed in the squeeze layer which increase the number of parameters in the expand layer of reception field 3*3. This issue is not very critical and solving it will be in future work. One suggested solution for this issue is to use Separable Convolutional layers [10] instead of the regular convolutional layers. These layers will reduce the number of parameters without affecting the accuracy.

The two figures above show the general overview of the SqueezeNet (left figure) and Simplified SqueezeNet (right figure). One quality of Simplified SqueezeNet over SqueezeNet is reduction of network architecture complexity. Where the suggested network simplify the network and follow straightforward pattern. This allowed the network to train faster even though it has more parameters than SqueezeNet.

My reflection on this project is it was very challenging in all aspects. Starting from preparing datasets ending with network setup and training. Where the size of the zipped MIT Places 205 is 132GB while the MIT Places 365 is 107GB. Downloading Places 205 took about 2 days while Places 365 took about 1 day and half. The reason is slow speed of the host server where the datasets are hosted. After downloading the data, I had to extract the images from the zipped files where the extraction took couple days too. It took me about 2 days to run many experiments to check the integrity of the two datasets.

As the used framework is Nvidia Digits, I had to convert the image data to the format used by Nvidia Digits. The conversion operation was easy because the used frameworks provides this functionality but the conversion operation took about 5 days in total for both sets. The deep learning library used is Caffe. SqueezeNet authors build their network using Caffe so I thought I will just copy and paste their network and will start training. Unfortunately, This wasn't the case. As Nvidia Digits used a modified version of caffe and there were no documentation available for the changes. So, I had to change the SqueezeNet network caffe files to be compatible with Nvidia Digits. I followed trial and error to fix this issue.

The training of the network was time consuming where the training took couple days just to train the SqueezeNet on a single dataset to get the benchmark results. As it is clear, trying to test many hyper parameters or network configuration will require very long time and also more money to run the experiments and my budget was running at this stage. So, I have to find a good modification that will run and get good results. Deciding which configuration to try wasn't easy but I leaned heavily on available research and on my experience training neural networks. In summary, designing, training and testing deep learning models is not easy as it looks as it comes with so many uncertainty about the performance of the new models.

VI. References

1. Denton, Emily L., et al. "Exploiting linear structure within convolutional networks for efficient evaluation." Advances in Neural Information Processing Systems. 2014.
2. Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).
3. Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in Neural Information Processing Systems. 2015.
4. Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv:1602.07360 (2016).
5. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
6. Zhou, Bolei, et al. "Learning deep features for scene recognition using places database." Advances in neural information processing systems. 2014.
7. Zhou, Bolei, et al. "Places: An image database for deep scene understanding." arXiv preprint arXiv:1610.02055 (2016).
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-9).
9. Retrieved from Github by April 9, 2017, https://github.com/DeepScale/SqueezeNet.
10. V. Vanhoucke. Learning visual representations at scale. ICLR, 2014.