SqueezeNet Model Reproduction

Elio Abi Younes, Yasmine Bougueche, Laith Mubaslat

Abstract

Reproducing works done in machine learning is becoming much sought after. Many of the well cited works were not reproduced on different datasets and/or tested under modifying the parameters. In this project we reproduce the baseline of the paper in [4]. The main finding of the paper is building a SqueezNet netowrk from the famous AlexNet squeleton but downsampling the number of estimated parameters while still achieving the same accuracy on ImageNet dataset. We had three main approaches. First, we tested the methods proposed in the paper by building and training the SqueezNet on a subset of the ImageNet and we compared its accuracy with the one of AlexNet; each of the networks achieved 70% accuracy. Second, significant results are shown when comparing the performance of AlexNet and SqueezNet on MNIST and CIFAR-10 datasets. Finally, we modified some parameters in the SqueezeNet architecture to assess the effect of each on the overall accuracy and number of estimated parameters of the Network. This was done on SqueezeNet itself and the proposed reduction techniques was applied to VGG19. Our main finding is that reducing the number of parameters while maintaining the same accuracy is an achievable goal that depends on the kind of task the model is tackeling.

1 Introduction

Increasing the accuracy level of deep convolutional neural networks (CNNs) is a pervasive problem. Often the goal is to build CNNs that are relevant to solve a particular problem such as classifying images, object recognition and many other aspects of visual data. Each CNN aims to have the highest prediction/ classification accuracy . The paper we are tackling in this work is entiteled ”SqueezNet: AlexNet-Level Accuracy With 50$\times$ Fewer Parameters And $<$ 0.5MB Model Size”[4]. It introduces a smaller CNN architecture of AlexNet called SqueezeNet with fewer parameters but equivalent accuracy. This smaller model can be as important for performing a particular task (such as classification with high accuracy) as it can be for understanding the underlying advantages. Smaller CNNs advantages, as underlined in the original paper, lie behind requiring less communication among servers and making frequent updates more feasible. SqueezeNet showed an excellent performance when compared to AlexNet as it was able to achieve same accuracy level when trained on ImageNet dataset with 50 times fewer parameters. The original paper also introduces a compression technique where SqueezeNet was compressed to less than 0.5MB that is 510 times smaller than AlexNet.

In this work, we first reproduce the SqueezeNet model to address this problem. We test this model on different datasets namely MNIST, ImageNet (subset due to computational restrictions) and CIFAR-10. The results we obtained showed that SqueezNet achieve similar accuracy to other state-of-the-art models(namely AlexNet), and proves the applicability of this network to other classification and recognition tasks. We also modified the model by removing or reparametrizing different components and check how it impacts performance. Finally, as VGGNet has been shown to be a powerful method for learning to extract features from images [2], our main contribution is in applying the compressed SqueezeNet architecture on VGGNet19 and showing that it is able to match AlexNet's accuracy on different datasets.

The rest of the report is organized as follows. In Section 2, we review the work related to SqueezeNet and other advanced models. Section 3 summarizes the original paper in all its aspects including a detailed description of the model's design. Then in Section 4 we evaluate the SqueezNet on the three datasets and compare our results to the original AlexNet model. We describe the VGGNet architecture and our modified model and compare their accuracy in Section 5. Section 6 concludes the report. For completeness, we provide some models' architecture and datasets examples in the Appendix.

2 Related Works

Existing deep neural network models are computationally expensive and memory intensive. During the past few years, tremendous progress has been made to perform model compression and acceleration in deep networks without significantly decreasing the model performance.

Han *et al.* introduced a three stage pipeline: pruning, quantization and Huffman encoding which combined reduce the storage requirement of neural networks by $35\times$ to $49\times$ without affecting their accuracy [3]. The first step consists of pruning the network by learning only the important connections which reduces the number of connections by $9\times$ to $13\times$. The next step is to quantize the weights to enforce weight sharing followed by Huffman encoding which reduce the number of bits that represent each connection from 32 to 5. This work was tested on the ImageNet dataset, reducing the storage required by AlexNet by $35\times$, from 240MB to 6.9MB, without loss of accuracy.

Another similar filter pruning strategy to compress CNN models was introduced in [9]. The importance of each filter is evaluated by the proposed entropy-based method, then several unimportant filters are discarded to get a smaller CNN model followed by fine-tuning to recover its generalization ability. Evaluated on the ILSVRC-12 benchmark, the entropy-based method achieves 3.3x speed-up and 16.64x compression on VGG-16, 1.54x acceleration and 1.47x compression on ResNet-50, both with about 1% top-5 accuracy decrease.

A work that inspired us to apply SqueezeNet on VGGNet is SqueezeNext introduced in [2]. The work done in this paper was guided by the paper we're tackling. Using SqueezeNet architechture as a baseline, SqueezeNext make the following changes: it applies a two-stage squeeze module which reduces the total number of parameters used with the 3×3 convolutions, then uses a separable 3×3 convolutions to reduce the model size, followed by an element-wise addition skip connection to avoid the vanishing gradient problem. SkeezeNext matched AlexNet's accuracy on the ImageNet benchmark with $112\times$ fewer parameters, and one of its deeper variants is able to achieve VGG-19 accuracy with only 4.4 Million parameters, that is $31\times$ smaller than VGG-19.

3 Background: AlexNet

AlexNet is one of the strongest performing CNN's in ImageNet classification competition [7]. The network is composed mainly of convolutional layers and fully connected layers. Relu activation function is applied after every convolutional and fully connected layer.The network has a dropout in the first and the second fully connected layers. The Netowrk takes as input RGB images with size 227×227 or 224×224. The full architecture is showed in the Appendix.

4 SqueezeNet

In this section, a detailed summary of the paper we're reproducing is provided. The paper first propose the SqueezeNet architecture and evaluate its performance with and without model compression. Then, the impact of design space exploration on the SqueezeNet-like CNN microarchitecture and macroarchitecture is explored [4].

4.1 Strategies and Fire Module

Three main strategies are outlined by the authors for reducing parameter size while maximizing accuracy. The first strategy consists of making the network smaller by replacing 3×3 filters with 1×1 filters which reduces the number of parameters 9×.
The second strategy reduces the number of inputs for the remaining 3x3 filters. This consists of reducing the number of parameters by just using fewer filters. This is done in a systematic way by feeding "squeeze" layers into "expand" layers as termed in the paper. The "squeeze" layers are convolution layers that are made up of only 1×1 filters and "expand" layers are convolution layers with a mix of 1×1 and 3×3 filters. Reducing the number of filters in the "squeeze" layer feeding into the "expand" layer, leads to reducing the number of connections entering these 3×3 filters thus reducing the total number of parameters. This architecture is shown in figure 1 and called the "fire module"; it serves as the basic building block for the SqueezeNet architecture.
The third strategy performs downsampling in the network by decreasing the stride with later convolution layers and thus creating a larger activation/feature map.

4.2. SqueezeNet Architecture

The SqueezeNet architecture takes advantage of the aforementioned "fire module" and chains a bunch of them to obtain a smaller model. It begins with one standalone convolutional layer (denoted by conv1), followed by several fire modules (mainly 8 denoted by fire2 till fire8), then another standalone convolution layer (denoted by conv10). Max-pooling with a stride of 2 is performed after each of conv1, fire4 and fire8, and global average pooling after conv10. The architecture ends with a softmax activation function (Check the Appendix for an illustration of this architecture).

4.3. SqueezeNet Evaluation

To evaluate SqueezeNet, AlexNet5 is used with the associated model compression results as a basis for

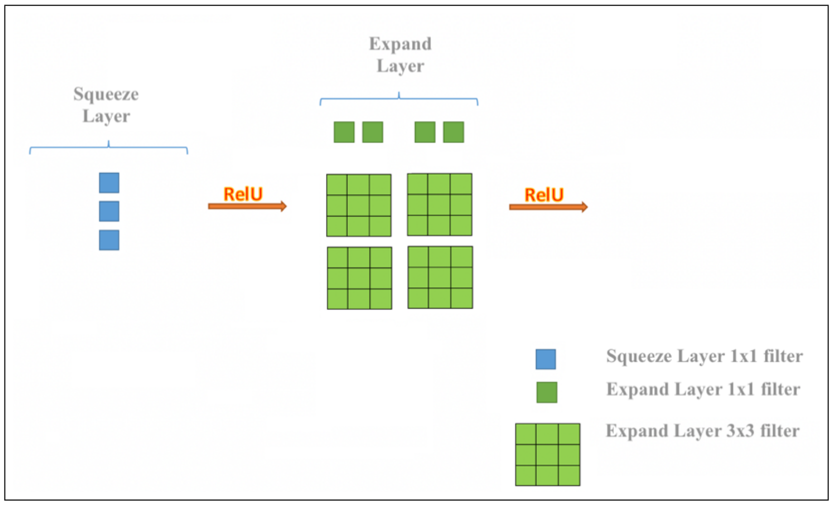


Figure 1: Example of a Fire Module

comparison. The results are impressive. SqueezeNet architecture achieves a 50× reduction in model size compared to AlexNet while meeting or exceeding the top-1 and top-5 accuracy of AlexNet. The most interesting part is the application of deep compression to the smaller model which creates a model that is 510× smaller than AlexNet.

4.4.Design Space Exploration

Several aspects of the design space are explored and classified into two main parts: the microarchitectural exploration deals with the per-module layer dimensions while the macroarchitectural exploration tackles the high-level end-to-end organization of layers.

4.4.1. CNN Microarchitecture

With the goal of understanding the impact of CNN architectural choices on model size and accuracy, the authors defined a set of metaparameters which control the dimensions of all Fire modules.
First, a squeeze ratio (SR) is defined by $SR = \frac{\text{\# filters in squeeze layers}}{\text{\# filters in expand layers}}$. Multiple modelsSR beyond 0.125 increases ImageNet top-5 accuracy from 80.3% i.e AlexNet level with 4.8MB, to 86.0% with a 19MB model. Second, the proportion of 1×1 and 3×3 filters in each Fire module's expand layer is modified form "mostly 1×1" to "mostly 3×3". Results show that increasing the percentage of 3×3 filters leads to a larger model size but shows no improvement in accuracy on ImageNet.

4.4.2. CNN Macroarchitecture

The high-level connections among Fire modules are tackled by comparing the following networks:
(1) Without residual connections (i.e. the prior model in Sections 4.1-4.3)
(2) With residual connections between modules of same dimensionality
(3) With residual connections between all modules (except pooling layers) using 1x1 convolution layer (instead of identity functions) where needed
Adding residual connections (2) improved top-1 accuracy from 57.5% to 60.4% without any modification

in model size. However, adding complex residual connections (3) decreased top-1 accuracy again to 58.8%, while increasing model size.

5 Model Reproducibility

As part of reproducing the above results, we first use precisely the architecture proposed by the authors (explained in Section 3.1 and 3.2).

5.1 Datasets

For model comparison, performance of each of AlexNet and SqueezeNet models on the following three datasets is evaluated.

5.1.1 MNIST

This data contains a set of hand-written digits. It consists of 40,000 labeled and 10,000 unlabeled images. These images are grey-scale of dimension 64×64. The task at hand is to recognize the digits (labeled 0 to 9) [1].

5.1.2 ImageNet

ImageNet is a dataset of over 15 million labeled images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. The data we are using is a subset of ImageNet with only 2 classes, bike and ship [7].

5.1.3 CIFAR-10

The CIFAR-10 dataset consists of 60000, 32×32 colour images. It contains ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, with 6000 images per class. There are 50000 training images and 10000 test images [6].

5.2 Results

Table 1 summarizes the results we obtainsed on training both SqueezNet and AlexNet on the previously mentioned datasets. We can see that, even though SqueezeNet is significantly smaller in terms of size and number of parameters estimated, its performance is very close to AlexNet (the difference ranges between 3% to 6%). This shows that indeed, SqueezeNet proposed in the paper can compete with AlexNet on different classification tasks and with various datasets. It is important to mention that our experimentation were limited by our computational capacity, and if stronger GPU's were provided , we would have investigated training SqueezeNet with more epochs and bigger datasets. An educated guess after our experiences is that the accuracy of SqueezeNet would converge to that of AlexNet [11].

Dataset	AlexNet Top-1 accuracy	SqueezeNet Top-1 accuracy
MNIST	50%	44%
ImageNet	78%	75%
CIFAR-10	67%	67%

Table 1: Performance of Alexnet and SqueezeNet on the three datasets

6 Model Variations

As part of validating the performance of the authors' SqueezeNet model and modifying its structure to assess and have a detailed understanding of the model in hand, sections 6.1 to 6.5 provides a brief description of all the modifications we made. The results are shown in section 6.6.

6.1 Data Preprocessing

The performance of both Alexnet and SqueezeNet was computed for CIFAR-10 both with and without major preprocessing. Image normalization was common to both tests. CIFAR-10 preprocessing involved augmenting the data with random horizontal flips and random cropping with a padding of 4 [8]. The preprocessed data was then used in the following tests.

6.2 Traditional Convolutional Neural Network (CNN)

A normal CNN architecture was introduced to put the performance of squeeznet in perspective when compared to models of similar size. The model consists of two convolution blocks with each block consisting of a 5x5 convolutions followed by a 3x3 maxpooling layer. The model utilized RelU activation and the blocks were followed by 3 fully connected layers [8].

6.3 Batch Normalization

Batch Normalization acts as a regulizer in machine learning models allowing the use of much higher learning rates while being less careful about initialization [5]. The authors do not appear to have used batch normalization. In order to evaluate its impact, batch normalization was introduced at different locations in the model mainly to the convolutional layers inside the fire module in addition to after conv1 in the original model.

6.4 Removal of Fire Modules

To evaluate the impact of changing the original structure of the SqueezeNet model, removing model components and assessing model performance is a common practice in machine learning. Our test involved the removal of fire modules fire2 and fire5 described in section 4.2.

6.5. Modified SqueezeNet Architecture

The design space exploration attempted by the original paper involved the implementation of a number of strategies. The tests done by the authors tackled the squeeze ratio between the squeeze and expand layers of the fire module and the ratio of 1×1 to 3×3 filters in the expand layer as per section 4.4.1. Our modifications however aim at expanding strategies 1 and 2 to be implemented over the entire structure of the model. This is done by creating a squeeze fire module and an expand fire module, thus mirroring the effect provided by the original fire module (parameter reduction at an acceptable accuracy) on the larger scope of the entire model. The expand layer of the squeeze and expand fire modules are dominated by 1×1 and 3×3 filters respectively. Additionally the number of convolutions for the squeeze Fire module was reduced by a factor which was varied for model size and accuracy. Following, the overall model structure was changed by replacing each pair of the original fire modules by a squeeze fire module followed by an expand module. Figure 2 visualizes our squeeze and expand fire modules and their connections.

The rational behind this test is two folds. First, it would explore part of the design space left unexplored by the authors (i.e. the notion of utilizing different fire module structures instead of the same module throughout the model). Second, it would serve to test the effect of strategies 1 and 2 when implemented over the model as a whole. Namely, the effect of the squeeze fire module reducing the number of input channels to the expand fire module by means of having its size reduced and being dominated by filters of size 1×1. Interesting results are shown in the next section.

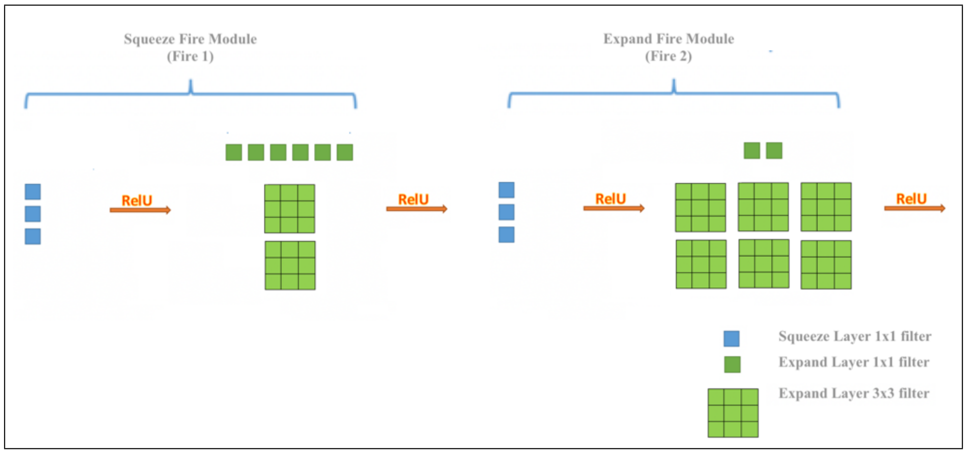


Figure 2: Squeeze and Expand Fire Modules

6.6. Results

CIFAR-10 is the dataset most relevant to the problem statement in the original paper in both nature and multi-class formulation. Accordingly, the performance of the different models discussed before was evaluated using this dataset.

Over the course of 12 epochs, Squeezenet reported an accuracy of 67% and 70% without and with preprocessing respectively. That agree with the results of the Alexnet implementation. The traditional CNN implementation accuracy topped up at 55%.

Significant reductions in accuracy are noticed when introducing batch normalization; the most severe of which (a resulting 10% accuracy) is when normalizing the conventional layers inside the fire module.

The models described in sections 6.4 and 6.5 were attempted over 25 epochs. Epoch count was increased considering how the differences were not as obvious over initially compared to the earlier tests. The baseline performance of the unmodified Squeezenet over that number of epochs was 80%.

The initial introduction the squeeze, expand fire modules described in figure 2 produces an accuracy of 78% for the same number of parameters. Following that with a reduction in the squeeze fire module by a factor of 4 produced an accuracy of 76%. Model size is reduced to 0.66× of the original (The Appendix sgows detailed number of parameters). Furthermore, no change in the accuracy is observed after the removal of fire units 2 and 5.

7 VGGNet

VGG is one of the top 5 best performing networks in the ImageNet competition. It has three versions, 13, 16 and 19 and the pretrained version of it is used as a defaqto in current classification tasks [12].

7.1. Original Architecture

VGG is a deep NN builds using 3×3 kernel size convolutional layers, maxpooling of 2×2 and fully connected layers at the end of the network. It started with a total of 16 layers with VGG16 and VGG19 with 19 layers. The original NN was desgined to have RGB images with 224×224 size [12]. The exact architecture of VGG19 is demonstrated in the Appendix.

7.2. Proposed Approach

In our project, we used VGG 19 as it's the deepest of all VGG versions and hence has the highest number of parameters. We wanted to test the concept of the reducing the kernel size and how it affects both number of estimated parameters and the performance of the network. For this we used the first part of the fire function by reducing the kernel size of all convolutional layers except the last ones to a one of 1×1 [2]. We then compared the performance of the Squeezed VGG with the full VGG trained on the CIFAR-10 dataset to assess the effect of downsizing the kernels on performance of deep neural networks. The results are shown in the upcoming section.

7.3. Results

The squzeed VGG 19 was trained on the CIFAR-10 dataset with 10 epochs without any image preprocessing. It gave an accuracy of 75% which is a remarkable accuracy compared to 79% with the trained VGG19. The number of parameters dropped by 18M compared to the full VGG which is very significant based on the modifications we made. We can see that downsizing only the Kernel reduces the number of parameters, yet it maintains a very closed accuracy to the full VGG19.

8 Discussion and Conclusion

This project evaluated the SqueezeNet model architecture and design criteria. The results can be seen to be consistent over the three datasets utilized in training the SqueezeNet model in contrast to the original ALexNet formulation. Further expanding on the design criteria of the SqueezeNet as in section 6.5 can be seen to have the same effect of preserving the accuracy while reducing the number of parameters. That serves to further validate the design methodology of the original paper. Additionally, SqueezeNet can be seen to provide results that are of a way more superior nature when compared to traditional CNN model formulations. The removal of fire units can be seen to have no effect on the accuracy of Squeezenet when trained on the CIFAR-10 dataset. Considering the goal of the original paper was to reduce the number of parameters while maintaining prediction accuracy. It can be fairly assumed that the authors of the original paper attempted this modification and obtained non satisfying results. That leaves us with the conclusion that while tests on datasets other than ImageNet might provide an insight into the performance of the model and its variations, more concrete results can only be obtained through tests done on the full ImageNet dataset.

SqueezeNet utilizes a single building block(i.e. the fire module). Accordingly, and based on our tests, we recommend that future work investigates the merit of utilizing different modules in creating an even smaller and more efficient variation of SqueezeNet [10].

9 Statement of Contributions

Team members contributed almost equally in reproducing the original paper, modifying the existing model, trying to improve upon it as well as writing the report.

References

[1] DeVries, Terrance, and Graham W. Taylor. "Dataset augmentation in feature space." arXiv preprint arXiv:1702.05538 (2017).

[2] Gholami, Amir, et al. "Squeezenext: Hardware-aware neural network design." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.

[3] Han, Song, Huizi Mao, and William J. Dally. "A deep neural network compression pipeline: Pruning, quantization, huffman encoding." arXiv preprint arXiv:1510.00149 10 (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] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[6] Krizhevsky, Alex, and Geoff Hinton. "Convolutional deep belief networks on cifar-10." Unpublished manuscript 40.7 (2010).

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[8] Katariya, Yash. Applying Convolutional Neural Network on the MNIST Dataset. yashk2810.github.io/Applying-Convolutional-Neural-Network-on-the-MNIST-dataset/.

[9] Luo, Jian-Hao, and Jianxin Wu. "An entropy-based pruning method for cnn compression." arXiv preprint arXiv:1706.05791 (2017).

[10] Qian, Xingzhi, et al. "Machine Learning on Cataracts Classification Using SqueezeNet." 2018 4th International Conference on Universal Village (UV). IEEE, 2018.

[11] Santos, Aline Gondim, et al. "Reducing squeezenet storage size with depthwise separable convolutions." 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.

[12] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

Appendix

Figure 3 shows the architecture of each of the models from the original paper in [4]. Three models are described as follows. The first one is the normal SqueezeNet while the the other two are compressed versions.

Figure 4 provides AlexNet architecture as designed in [7]. It's the main complex structure with a very large number of parameters.

Figure 5 visualizes VGG19 architecture as mentioned in section 7.

The numnber of parameters for each model described in scetion 6 and 7 appear in figure 6 and 7.

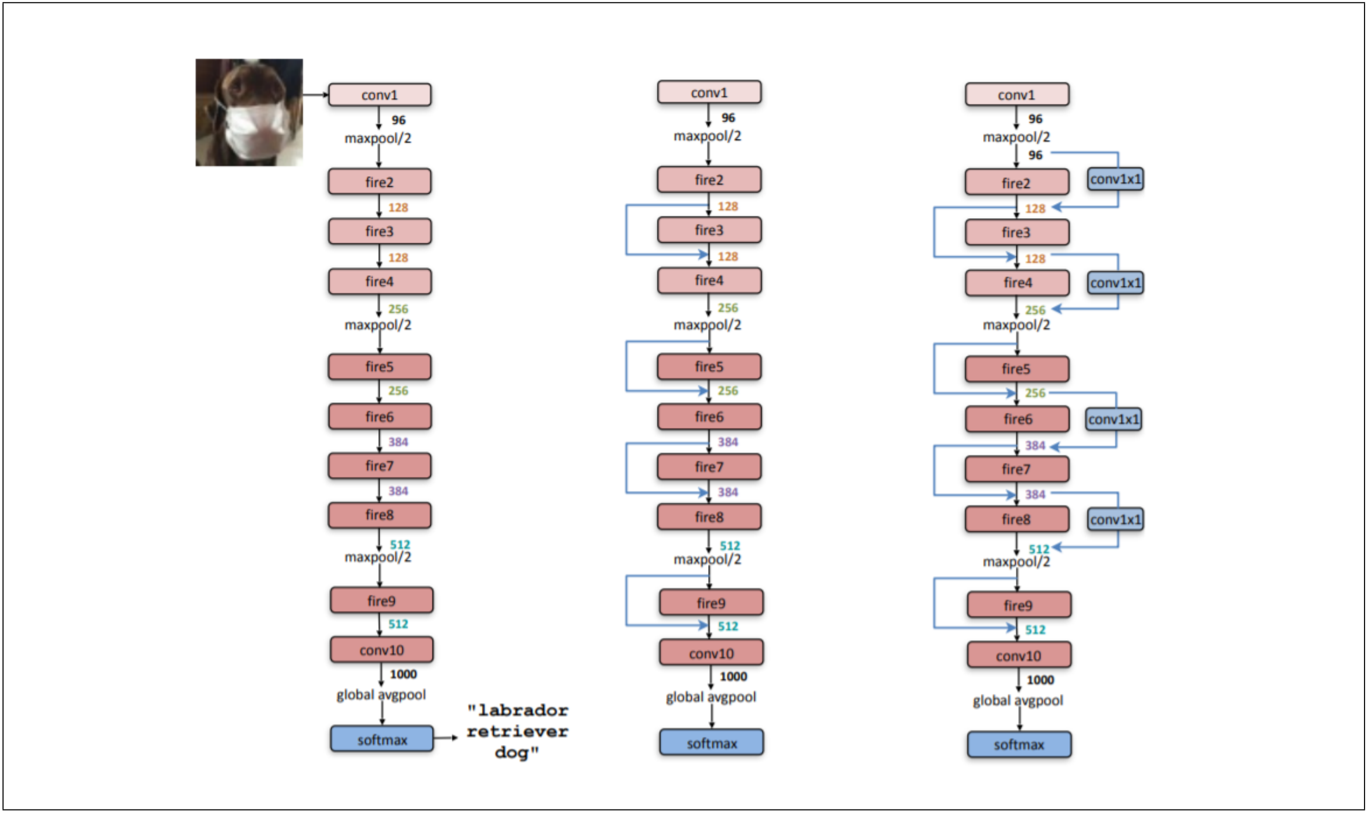
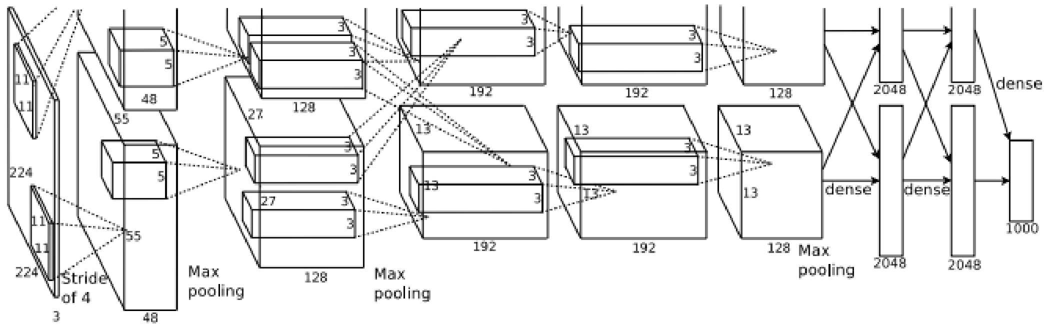


Figure 3: Image from paper; "Left: SqueezeNet; Middle: SqueezeNet with simple bypass; Right: SqueezeNet with complex bypass. Image from paper."



net.png net.png

Figure 4: AlexNet Architechture

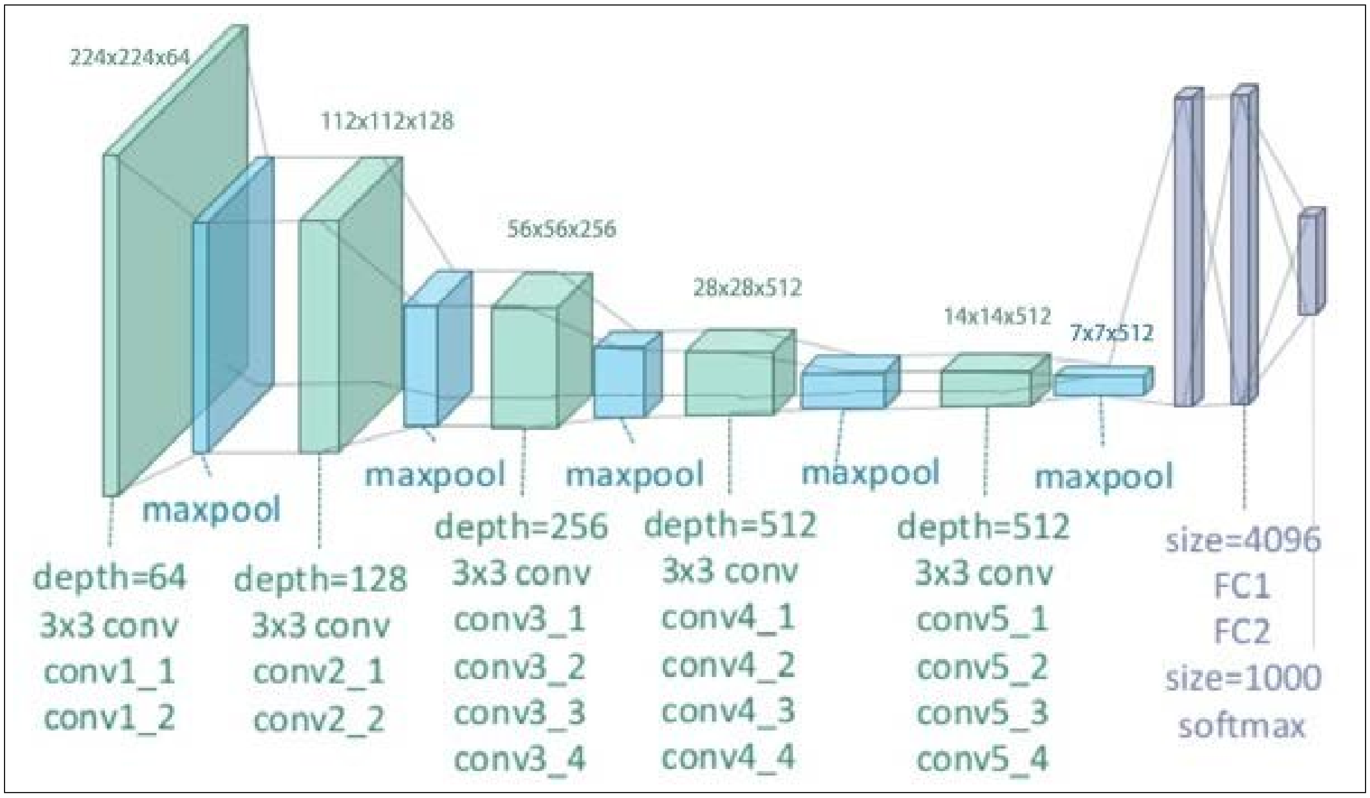


Figure 5: VGG19 Architecture

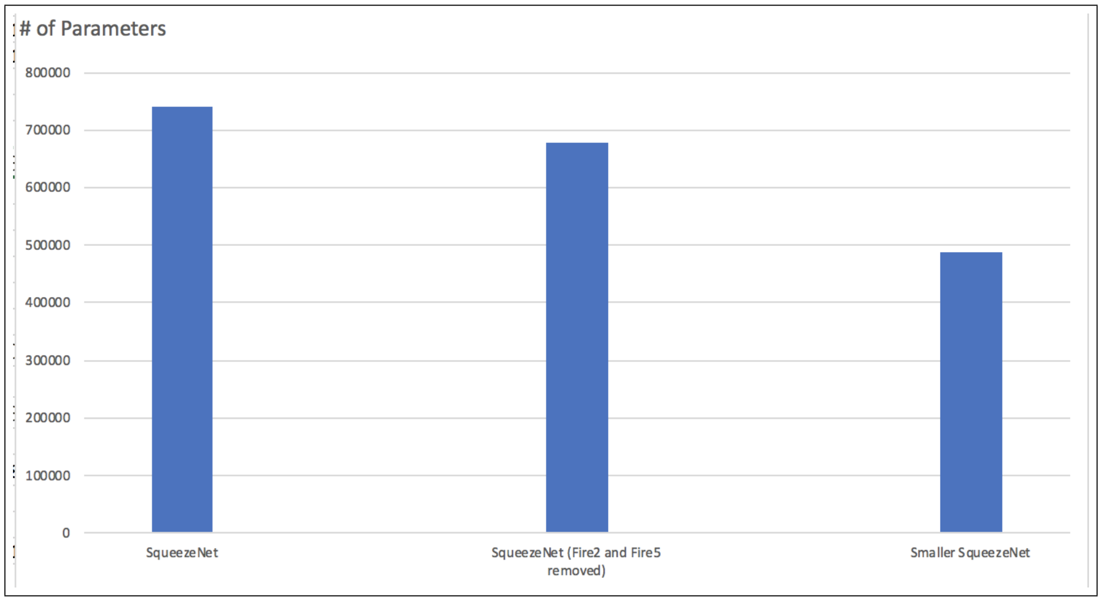


Figure 6: Number of parameters for the modified SqueezeNet as per section 6

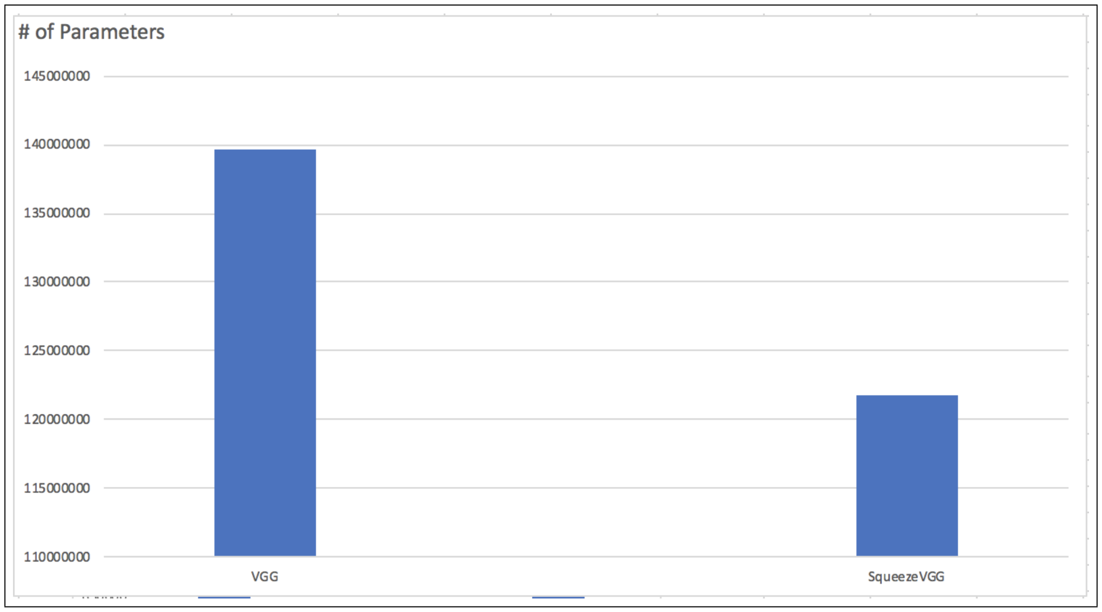


Figure 7: Number of parameters for the VGG model as per section 7