# COMP 551 - MiniProject 4

John Flores, Luis Pinto, John McGowan

April 18, 2019

**Abstract**

In this paper, the SqueezeNet [1] architecture is modified and tested on the CIFAR10 dataset [2]. Our first modified model, named Sq-1, adds two linear layers with dropout to the end of SqueezeNet; the other modified model, named Fire-Next, replaces one Fire module of SqueezeNet with a modified SqNxt block [3]. Both models perform similarly, with Sq-1 having 797K parameters to learn and the Fire-Next model having 691K parameters to learn. Hyperparameters are tuned in search of better accuracy. The top-1 accuracy of these models are 86.7 % for Sq-1 and 86.5 % for Fire-Next.

## 1 Introduction

The SqueezeNet CNN architecture was introduced in 2017 and touted "AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size" [1]. SqueezeNet achieves this through three main strategies: replacing 3x3 filters with 1x1 filters, decreasing the number of input channels to 3x3 filters, and downsampling late in the network. SqueezeNet also incorporates 3x3 filters in tandem with 1x1 filters in so called "fire modules". With these strategies, ReLU activations, dropout, and other design choices SqueezeNet achieves 57.5% Top-1 accuracy on ImageNet [4] with a model size of 4.8MB, backing up the aforementioned claim. Through model compression techniques, SqueezeNet can be reduced even further in size. In this paper, the SqueezeNet architecture is discussed and the related literature is reviewed. Variations of SqueezeNet are tested on the CIFAR10 dataset [2] and hyperparameters are fine tuned in search of better accuracy on the dataset from the model presented by Iandola et al [1]. All experiments were performed using the Pytorch deep learning platform [5].

## 2 Related Work

### 2.1 Image Recognition using Convolutional Neural Networks

Image recognition with convolutional neural networks dates back to 1989 with LeCun's backpropagation algorithm applied to identifying hand written zip codes [6]. The field exploded when a CNN named AlexNET drastically outperformed the competition on the ImageNet challenge [7]. While AlexNET ahieved high performance on the ImageNet challenge, the uncompressed model is 240MB and contains some 61 million parameters. The immediate trend after AlexNET was deeper models with longer training times trained over multiple GPUs. As models are trained on more GPUs, communication between nodes becomes a significant cost of training. In [8], Iandola et al. demonstrate that this cost is proportional to the number of model parameters. Beyond the cost of training, the size of models limits the application of the models. For example, FPGAs seldom have over 10MB of memory, rendering AlexNET unfeasible on an FPGA [1]. These concerns have motivated research smaller CNNs that do not sacrifice accuracy when limiting the number of parameters.

A significant step in decreasing the complexity of CNNs came with ResNET [9], which won the ILSVRC challenge in 2015. ResNET incorporated residual connections, which connect a layer not only to the layer directly preceeding it, but to multiple previous layers. In doing this, ResNET acheived a lower completxity than VGGNet despite a far greater depth. While ResNET attempted to reduce complexity by adding more connections between layers of 3x3 fileters, GoogLeNet [10] attempted to tackle the problem of model complexity with more carefully designed "inception layers". Inception layers include multiple convolutions

of different size and concatenate the output of these layers before the next layer. These are engineered with the idea that salient features may vary in size from image to image. GoogLeNet won the ILSVRC challenge in 2014.

### 2.2 SqueezeNet

This paper specifically examines SqueezeNet [1]. Similar to GoogLeNet's inception layer, SqueezeNet makes use of fire modules, shown in Figure 1, in which the output from multiple 1x1 filters are fed into multiple 1x1 and 3x3 filters. The output of these layers are then concatenated and the ReLU activation function is applied before the output is sent to the 1x1 filters of the next fire module. The purpose of these fire modules was to decrease the number of input channels to 3x3 filters, thus decreasing the number of parameters in the system. Squeeze1.1, which this paper focuses on, consists of 8 fire modules sandwiched between convolutional layers. Maxpool is applied after the 4th and 8th fire modules and an averaging layer is used after the final convolutional layer, as shown again in Figure 1. This model achieved AlexNet top-1 and top-5 accuracy on ImageNet [4] (ILSVRC 2012) dataset, with 50x fewer parameters than [7].

Since its development, SqueezeNet has been used in many applications, especially on mobile devices where memory is limited. For example, SqueezeDet [11], a model based on SqueezeNet with additional layers, has been proposed for use in self driving cars. SqueezeDet is only 8MB and achieved the best average precision in the category of cyclist detection in the KITTI object detection challenge [12]. Of course, SqueezeNet was just the beginning of CNNs for small applications. Since SqueezeNet, SqueezeNext [3] has achieved AlexNet accuracy with 120x fewer parameters. This is achieved with separable 3x3 filters and skip connections among other improvements. Other notable works in accurate CNNs of small size include, but are not limited to, DarkNet [13], MobileNet [14], and ShuffleNet [15]. These papers all use different strategies to reduce model size are referenced for the interested reader.

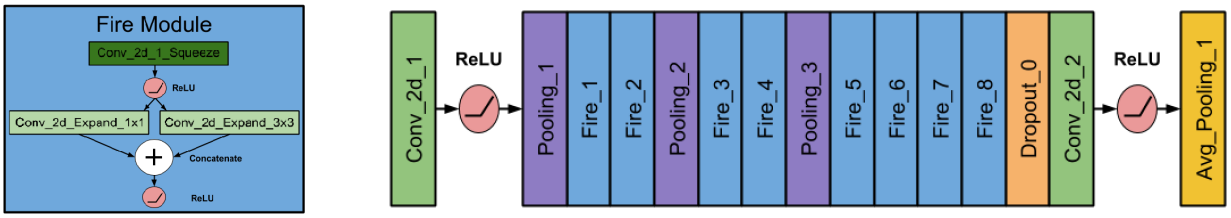


Figure 1: Fire Module and Squeezenet Architectures. (Left): Fire Module: 1x1 Convolution Filters (shown in dark green) feed into two layers, 1x1 convolution filters and 3x3 convolution filters, which are concatenated together. (Right): Squeezenet Architecture. A preprocessed image of size 224 x 224 x 3 is fed into the first convolutional layer, while the pooling layer gives the final classification.

## 3 Datasets

Originally, SqueezeNet was trained on the ILSVRC-2012 dataset, which was comprised of 1.2 million images of 1000 different categories. This paper will focus on using SqueezeNet and its variations on the CIFAR10 dataset [2]. The CIFAR10 dataset is made up of 60000 32x32 colour images split evenly into 10 classes. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Example images from each class are shown in Figure 2. Note that the classes are mutually exclusive, meaning that there is no overlap between labels in the images.

## 4 Preprocessing

In order to train the models on the CIFAR10 dataset, the CIFAR images were resized from 32x32 to 256x256 and then central cropped to 224x224 to match the models' input size. Moreover, the images are normalized

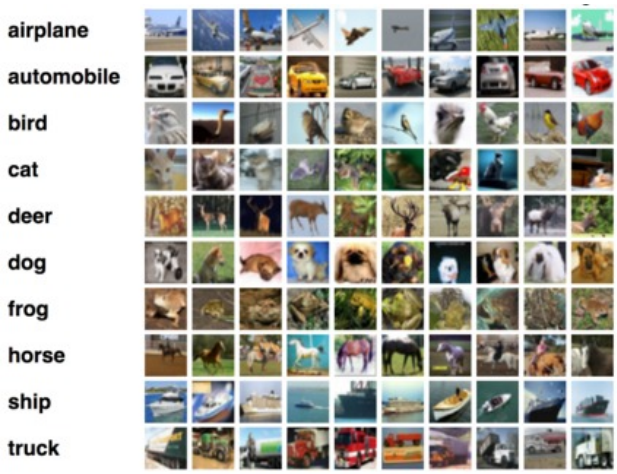


Figure 2: Sample images from the CIFAR10 dataset

across each input channel (i.e. subtract the mean of the channel and the result is divided by its standard deviation).

# 5 Benchmarking

We use AlexNet's Top-1 accuracy [7] as a basis for comparison as in the original SqueezeNet paper [1]. First, the pre-trained model on ImageNet (with 1000 classes) was imported and, because of the nature of the dataset, we changed the last layer of AlexNet to match the required number of classes. Then, the model was re-trained freezing all layers of the CNN except for the last layer. Consequently, the number of parameters that this model needs gets reduced from 61.9M to 57M.

# 6 Models

## 6.1 Original Architecture

In this project, we decided to use the pre-trained SqueezeNetv1.1 model on ImageNet instead of training it from scratch because of time and computation power limitations. As mentioned above, the last layer of the CNN needs to be adapted to the new dataset, and thus fine tuned by freezing all layers except for that one. After changing the last layer, the number of parameters got reduced to 727K from the original 773K.

## 6.2 Modified Architectures

### 6.2.1 Sq-1 Model

To achieve better results, two linear layers were added to the end of the original SqueezeNet architecture, as shown in Figure 3. After each linear layer, dropout was added to reduce the likelihood of overfitting from adding more layers. Moreover, the output of the last convolutional layer was also changed to have a decreasing number of channels starting from that layer. This CNN will be terms the Sq-1 Model.

### 6.2.2 Fire-Next Model

Inspired by the SqueezeNext architecture, we decided to remove the last Fire module from the original architecture and replace it by a SqNxt block [3]. Then, the last two layers, SqNxt block and classifier, were re-trained while keeping the other layers' weights frozen. This changes the total amount of parameters from 773K to 691K and showed an increase in performance as shown in Table 2. The architecture of the model

can be seen in Figure 4 and the SqNxt block is visualized and explained in Appendix B. It is important to note that we did not adopt the element-wise addition skip connection that characterizes the SqNxt block.

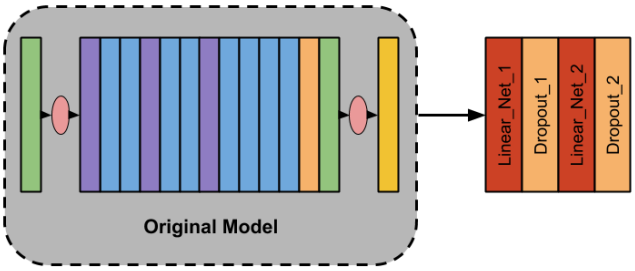


Figure 3: Sq-1 Architecture

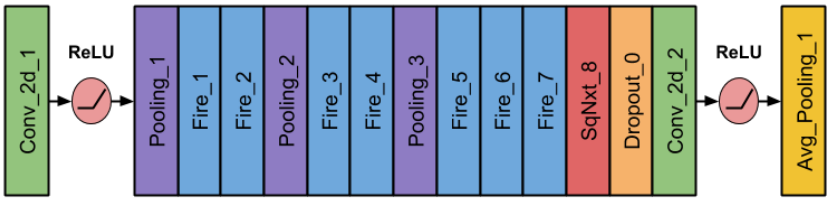


Figure 4: Fire-Next Architecture

## 6.3 Experimental Procedure

For all our experiments, we assume that the hyperparameters affect the validation accuracy independently. Thus, we could reduce the number of tests to perform and save resources. The hyperparameters to be tested are: the learning rate, the number of input/outputs channels of the layers added and the probability of the dropouts (in case of Sq-1). Due to lack of resources, we limited ourselves to do an extensive optimization of one of the previous mentioned models: Sq-1; and a less rigorous optimization of the other model: Fire-Next. Finally, the total number of parameters for our models was calculated by adding the required parameters for each layer.

### 6.3.1 Sq-1 optimization

We had to decide upon a model with base hyperparameters that will be varied independently, excluding dropouts. All combinations of dropouts between 0.1 and 0.4 were tested. Based on preliminary tests with a small subset of random values, we chose hyperparameters that seemed to give decent results. Therefore, the base hyperparameters we use for our linear model were learning rate of 0.001, a mid-layer size of 64, dropout probabilities of 0.3 and 0.1 after the first and second added linear layers, respectively, max pooling after the second and fourth fire modules and average pooling before the two linear layers. The base model to be modified is summarized in Table 1. In addition to these, the type of poolings used by the original model were considered. However, we only change the second, third, and final poolings (i.e. Pooling_2, Pooling_3, and Avg_Pooling_1 in the original SqueezeNet architecture in Figure 3). Note that the type of pooling is part of the architecture as opposed to a hyperparameter.

### 6.3.2 Fire-Next optimization

For this model, we decided to test only the learning rate and the output size of the SqNxt block due to lack of resources. With respect to the learning rate optimization, we decided to test five values raging from 0.005 to 0.00002 for 50 epochs. Using the value that gave us the best results, we proceeded to test different values of the output size (which in turn is the input of the last convolutional layer): 100, 128, 192 and 256.

| Base Linear Model Hyperparameters | |
|---|---|
| Learning Rate | 0.00100 |
| First Layer Size | 64 |
| Dropout_1 | 0.3 |
| Dropout_2 | 0.1 |
| Pooling Configuration | Max/Max/Average |
| **Base Accuracy:** | 86.1% |

Table 1: Hyperparameters of our base model. Each value will be varied independently while the others are kept constant for each set of experiments. This will isolate the effect of each of these hyperparameters on the model's accuracy.

# 7 Results

We trained our two models and compared their best accuracy and model size on CIFAR10 in Table 2. The two models used pre-trained weights on ImageNet (ILSVRC 2012) dataset and the new layers were trained using Adam optimizer.

| Model | Top-1 test loss | Top-1 accuracy | Parameters |
|---|---|---|---|
| AlexNet (pre-trained model) | 0.511 | 82.7 | 57M |
| SqueezeNet (pre-trained model) | 0.490 | 83.1 | 773K |
| Sq-1 (ours) | 0.403 | 86.7 | 797K |
| Fire-Next (ours) | 0.404 | 86.5 | 691K |

Table 2: Performance of variations of the SqueezeNet architecture after optimization.

## 7.1 Sq-1 Hyperparameters

Over the course of testing, the parameters of the original SqueezeNet were modified one at a time and independently of the others. Different dropout probabilities were tested at both layers, with the best performance of 86.3% accuracy on CIFAR10 achieved with dropout probability = 0.1 on both layers, as shown in Figure 5. Other combinations of dropouts would lead to a decline in accuracy. Similarly, combinations of different pooling techniques in the three pooling layers are tested, with the original combination of max pooling layers after the second and fourth fire modules and average pooling before the linear layers achieving the highest validation accuracy. These results are shown in Figure 7. Finally, a 32-node mid-layer is found to outperform the original 64-node layer, as shown in Figure 10 in Appendix A. These parameters are summarized in Table 3. The effect of learning rate on validation accuracy is more pronounced, with values lower than 0.001 performing similarly, as shown in Figure 6. Performance begins to suffer at values 0.002 and larger.

| Final Linear Model Hyperparameters | |
|---|---|
| Learning Rate | 0.0004 |
| First Layer Size | 32 |
| Dropout_1 | 0.1 |
| Dropout_2 | 0.1 |
| Pooling Configuration | Max/Max/Average |
| **Final Accuracy:** | 86.7% |

Table 3: Hyperparameters of our final model. These parameters amounted to a 0.6 % increase in accuracy over the base model.

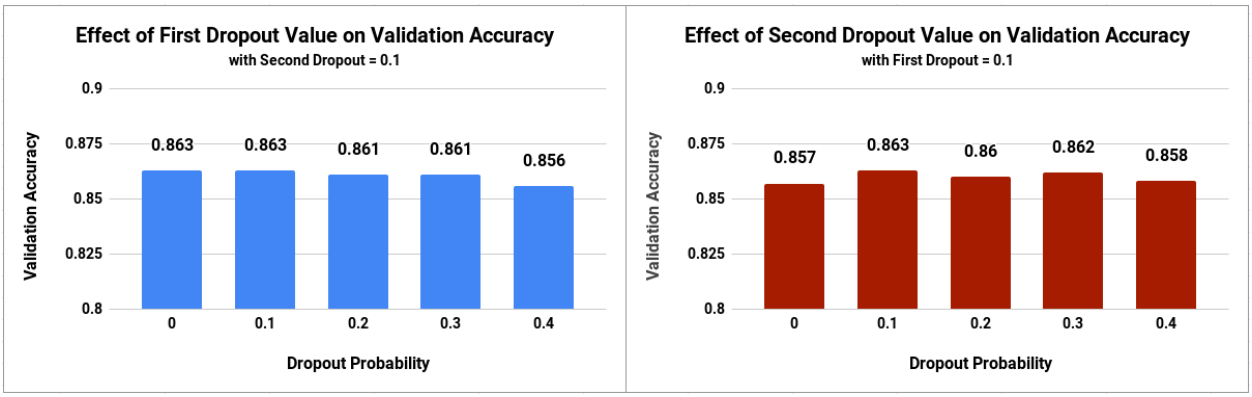


Figure 5: Validation Accuracy with changing dropout probabilities. Optimal performance was achieved with both dropout probabilities being set to 0.1.

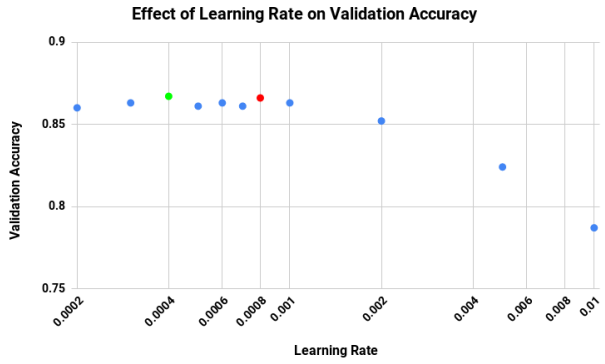


Figure 6: The effect of learning rate on our Sq-1 model. At learning rates higher than 0.001, performance suffers significantly. Optimal performance is achieved at a learning rate of 0.0004 with a validation accuracy of 86.7%, shown in green. A learning rate of 0.0008, shown in red, gives a comparable performance with a validation accuracy of 86.6 %.

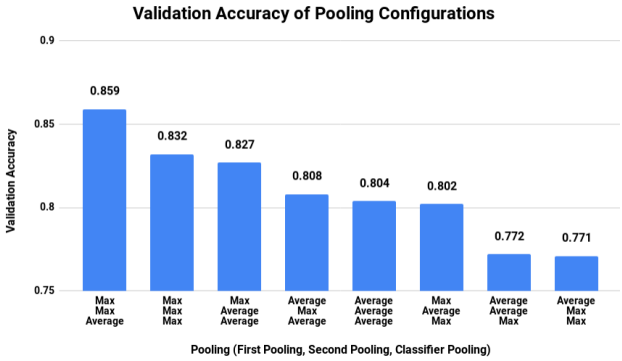


Figure 7: The effect of differing pooling configurations on our Sq-1 model. The original configuration of Max/Max/Average performs best with 85.9 % accuracy. Changing the second pooling from Max to Average had the most effect on the accuracy, decreasing performance to 80.2 %.

## 7.2 Fire-Next Hyperparameters

We focus the optimization of this model on two hyperparameters: learning rate and output size of the SqNxt block. As shown in Figure 8, we found that a learning rate of 0.001 gave us the best results. We can make two remarks based on that plot: the learning rate of 0.005 got the lowest accuracy which suggests that the we need a small learning rate to fine tune this model. Also, the learning rate of 0.001 converged to the highest accuracy up to five times faster than the rest

Moreover, we tested different values for the output size of the SqNxt block. The trends in Figure 9 show that they all converge to the same accuracy 86.5 % but only the output size of 128 got a test loss reduced to 0.404.

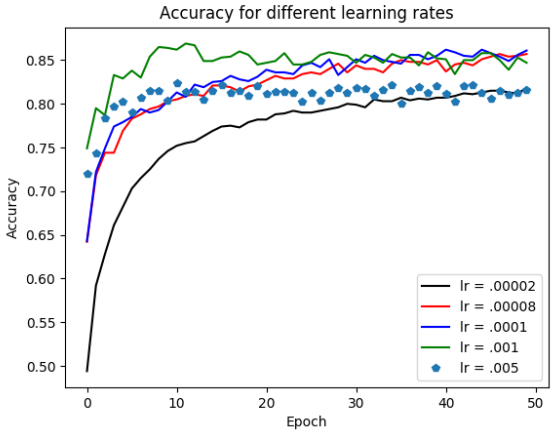


Figure 8: The effect of learning rate on our Fire-Next model over training epochs. Learning rates 0.00008, 0.0001, and 0.01 converge to similar values, while values 0.00002 and 0.005 converge to similar, but lower accuracies.

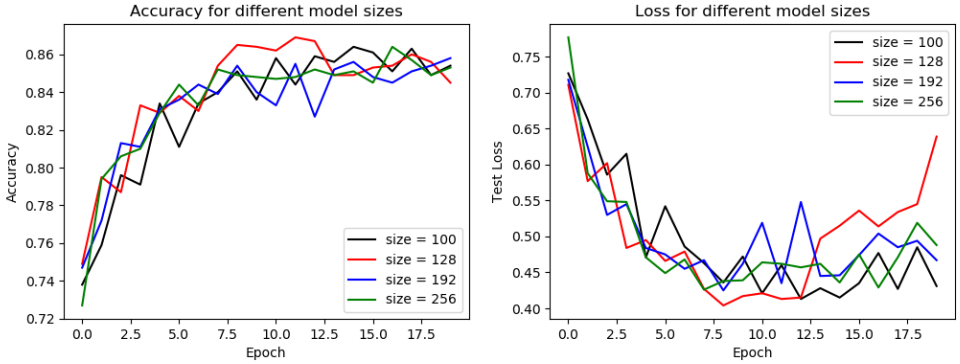


Figure 9: The effect of output size on our Fire-Next model. All models converge to the same accuracy, with output size = 128 performing the best.

# 8 Discussion and Conclusion

In this work, we present two variations of SqueezeNet, Sq-1 and Fire-Next, that were able to outperform AlexNet's and SqueezeNet's top-1 accuracy on the CIFAR10 dataset. The first of our modified models, Sq-1, consists of the original SqueezeNet architecture with two linear layers with dropouts added at the end to control overfitting. The hyperparameters that seemed to contribute most to our results were learning rate and pooling configuration. As discussed above, learning rates above 0.002 proved to decrease performance rapidly; this non-convergence is most likely due to the model "bouncing" between two configurations with the local minima in the middle. However, we note that the learning rate = 0.0004 trial and the final model have similar validation accuracies. That trial, with base dropout probabilities and base first layer sizes, performing comparably to our final linear model suggests that the layer size (32 vs 64 nodes) and the first dropout probability (0.3 vs 0.1) are hyperparameters that do not significantly affect our results. In fact, this linear model achieves the same validation accuracy with dropout probabilities of 0 and 0.1. This suggests that the first dropout layer may not be necessary. However, the second dropout value shows a 0.6 % increase in accuracy between dropout probabilities of 0 and 0.1, suggesting that the second dropout layer may increase performance.

Our second model, Fire-Next, uses the same architecture as SqueezeNet but replaces the last Fire module with a SqNxt block. This reduces the number of parameters to train by 80K with respect to SqueezeNet and

achieves a ∼3 % increase in accuracy. We optimized this model with respect to the learning rate and the output size of the SqNxt block. We find that a learning rate of 0.001 and an output size of 128 converges after ∼8 epochs and gets the highest accuracy and lowest test loss.

In conclusion, as a result of our optimizations, both models have ∼3 % higher accuracies and test loss reductions of ∼0.1 compared to AlexNet and SqueezeNet on the CIFAR10 dataset. Furthermore, in terms of the number of parameters, the Sq-1 model added only 20K more parameters compared to SqueezeNet, using x71 fewer parameters than AlexNet; in the case of the Fire-Next model, 80K less parameters are utilized compared to SqueezeNet and x82 fewer parameters than AlexNet. Like the original Squeezenet, these models use significantly fewer parameters than AlexNet while producing better results. However, Fire-Next, using 100K less parameters than our Sq-1 model, represents a ∼10% reduction in parameters over Squeezenet.

To further improve the accuracy of our SqueezeNet variations, experiments could be done using skip connections as they are demonstrated to be effective in other SqueezeNet models [1] and in SqueezeNext [3]. SqueezeNext and other recent models such as MobileNets [14] have also made use of separable convolutions, in which a convolution filter is expressed as the product of two vectors, in order to save on parameters. Much work has been done since the original SqueezeNet paper towards achieving higher accuracy with fewer parameters. One must only consult the literature to see a great range of possibilities for further experimentation with the SqueezeNet model presented here.

This research was enabled in part by support provided by CalculQuebec (http://www.calculquebec.ca) and Compute Canada (www.computecanada.ca).

# 9 Statement of Contributions

The work was split evenly among the group. The models were written mainly by Luis Pinto. John McGowan and John Flores participated in training and testing of the models and the write-up was shared between the members.

# References


[1] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.

[2] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

[3] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. Squeezenext: Hardware-aware neural network design, 2018.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[6] J.S. Denker D. Henderson R.E. Howard W. Hubbard L.D. Jackel Y. LeCun, B. Boser. Backpropagation applied to handwritten zip code recognition, 1989.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Forrest N. Iandola, Khalid Ashraf, Matthew W. Moskewicz, and Kurt Keutzer. Firecaffe: near-linear acceleration of deep neural network training on compute clusters, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[11] Bichen Wu, Alvin Wan, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving, 2016.

[12] Andreas Geiger and Philip Lenz. Are we ready for autonomous driving? the kitti vision benchmark suite, 2012.

[13] J. Redmon. Darknet: Open source neural networks in c.

[14] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[15] C-L Ng, S-W Seo, and Hisashi Kobayashi. Performance analysis of generalized multihop shuffle networks. *IEEE Computer and Communications Societies, Annual Joint Conference of the*, 0:842, 01 1997.

# Appendix A Additional Plots of Results

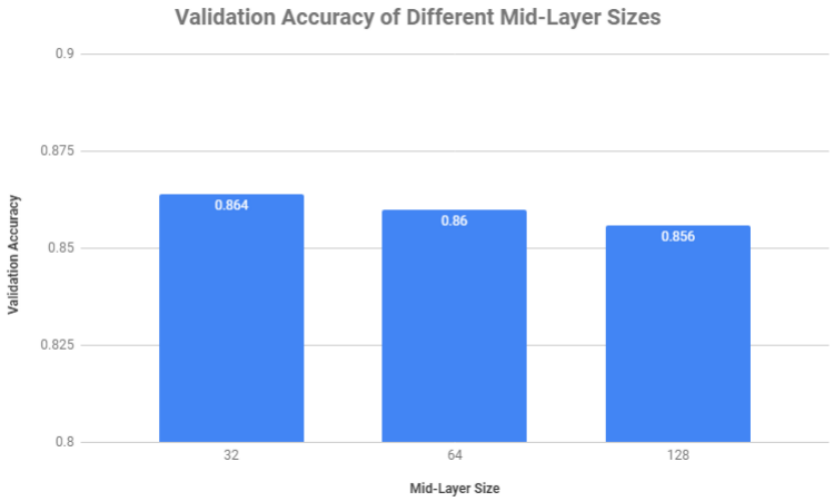


Figure 10: Validation accuracy as a function of mid-layer size. Accuracy is highest with a 32-node layer, with accuracy decreasing with layer size.

# Appendix B SqueezeNext Module

SqueezeNext is another deep neural net module developed by Gholami et al [3], much like the Fire module shown in Figure 1. The module utilizes two squeezing layers, designed to half the number of inputs each time. The 3x3 expanding layer is now replaced with two, with kernels 3x1 and 1x3. Finally, the expansion layers are now sequential, as opposed to being parallel and then concatenated. In a SqNxt module, the end result of all convolutions is concatenated to the original input.

The purpose of the SqueezeNext block (SqNxt) is to reduce the number of parameters the module must learn. Three major changes were made when compared to the Fire module. First, it reduces the number of parameters used with 3x3 convolutions: squeezing twice further reduces the necessary parameters. Second, the use of 3x1 and 1x3 reduces the size further when compared to a 3x3 kernel. Finally, the use of an element-wise addition reduces the effect of vanishing gradients.

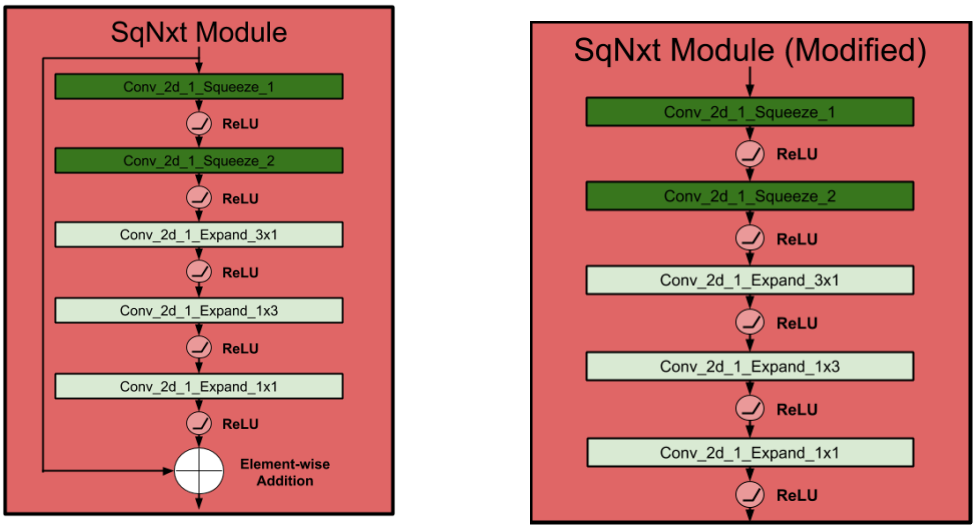


Figure 11: (Left) SqNxt block as developed by Gholami et al. Convolutional layers are stacked sequentialy and added element-wise to the original input. (Right) SqNxt block used in our Fire-Next model. Note the lack of element-wise addition.