SqueezeNet Architecture

Krutheeka R K J
Dept. of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, India
200070038@iitb.ac.in

Pranoti Pravin More
Dept. of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, India
200070058@iitb.ac.in

***Abstract*—A small CNN architecture named SqueezeNet has been used to classify the images of wild cats into five species. This architecture offers three advantages: (1) less communication across servers. (2) less bandwidth to export from the cloud. (3) feasible to deploy on FPGAs and hardware with limited memory. The model achieved expected classification accuracy, demonstrating its effectiveness.**

***Index Terms*—Deep learning, squeezenet, convolutional neural network**

I. Introduction

Convolution Neural Networks (CNN) is a type of deep learning model used for processing and analyzing images. Much of the recent work on CNNs has been focused on increasing the accuracy of image vision datasets and decreasing the size of the model. In today's world, almost everything is controlled remotely, like self-driving cars, which require constant communication with the server to work efficiently. It is good to have a small model to deploy easily in the cloud. Small CNNs require less communication between servers during distributed learning. They are more feasible to implement on FPGAs and other hardware with memory constraints. However, the lesser size would often lead to compromise in the accuracy. So, we need an architecture with lesser size and also provides accuracy as high as any other architecture. *SqueezeNet* is a CNN model with a smaller number of parameters but a high level of accuracy as compared to a well-known model. We implement this model for multi-class classification and analyze the accuracy results.

II. Approach

We employ three main design strategies for building CNN architectures with fewer parameters:

- **Strategy-1. Replace 3x3 filters with 1x1 filters.** We choose the majority of the convolution filters to be 1x1 since it has 9 times lesser number of parameters compared to 3x3.
- **Strategy-2. Decrease the number of input channels to 3x3 filters.** Since the number of parameters increases with the number of input channels, we decrease it for the 3x3 filters. We do it using *squeeze layers.*
- **Strategy-3. Downsample late in the network so that convolution layers have larger activation maps.** It is claimed that large activation maps can lead to higher classification accuracy.

Strategy 1 and 2 deal with decreasing the number of parameters of the model while strategy 3 aims to increase the accuracy of the model.

A. FIRE MODULE

It is comprised of two layers: a *squeeze* convolution layer made up of only 1x1 filters, feeding into the *expand* convolution layer which is a mix of 1x1 and 3x3 filters. We have three tunable hyperparameters: s_{1x1}, e_{1x1} and e_{3x3} which represents the number of filters in the squeeze layer, the number of 1x1 filters in expand layer and the number of 3x3 filters in expand layer, respectively.

B. ARCHITECTURE

SqueezeNet begins with a convolution layer, followed by Fire modules, ending with a final convolution layer. Batch normalization is done after each convolution since the model didn't train without it. There are max-pooling layers with stride 2 included between some fire modules. Average global pooling is done after the final convolution layer. A dropout layer with a ratio of 0.2 is added to the architecture. ReLU is applied to activation from squeeze and expand layers. The expand layer is implemented with two separate convolution layers: a layer with

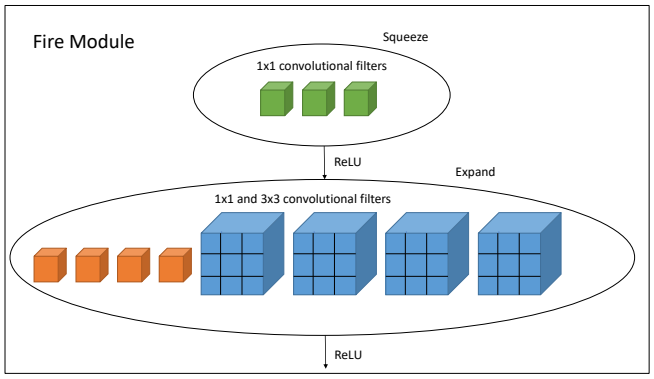

Fig. 1. Fire module consists of squeeze layer with 1x1 convolutional filters and expand layer with 1x1 and 3x3 convolutional filters. The output of the squeeze layer is passed to the 1x1 and 3x3 convolutional filters and then concatenated along the channel dimension. The activation used is ReLU. Batch normalization is performed after each convolution.

1x1 filters and a layer with 3x3 layers. They are concatenated along the channel dimension to produce the expand layer. This model doesn't contain fully-connected layers.

C. SQUEEZENET METAPARAMETERS

We define $base_e$ as the number of expand filters in the first Fire module in a CNN. After every $freq$ fire module, we increase expand filters by $incr_e$. Therefore, the number of expand filters is,

$$e_i = base_e + (incr_e * [\frac{i}{freq}])$$

In the expand layer, we have both 1x1 and 3x3 layers. We define $e_i = e_{i,1x1} + e_{i,3x3}$ and pct_{3x3} as the percentage of expand filters that are 3x3. Therefore,

$$e_{i,3x3} = e_i * pct_{3x3}$$

$$e_{i,1x1} = e_i * (1 - pct_{3x3})$$

We define the ratio of the number of filters in the squeeze layer to the number of filters in expand layer as *squeeze ratio.*

III. EXPERIMENT

We are using the dataset consisting of images of various wild big cats. The goal is to develop a model that can classify these images into five categories based on the species of the big cat: Caracal, Cheetah, lion, puma, and tiger. Each class has about 240 images for training. The training dataset contains 1031 images, the validation dataset contains 141 images and the test dataset contains 50 images. Each image is of dimensions $224 \times 224 \times 3$. To achieve the goal, we must preprocess the dataset by resizing and normalizing the images. The data is augmented as well. With the help of the SqueezeNet model, we perform the classification task.

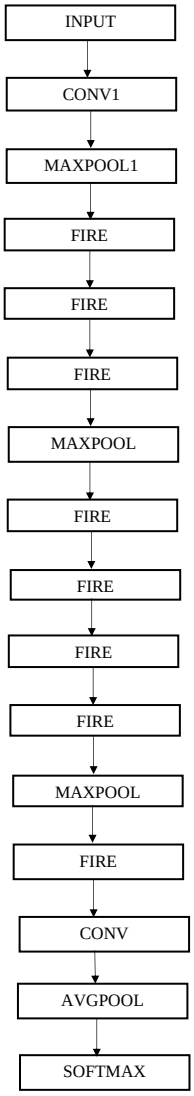

Fig. 2. An example of the model architecture.
The input is first convolved, followed by a combination of fire modules and pooling layers, followed by convolution, average pooling, and softmax activation.

IV. Results

The following results are obtained when the model is trained with squeeze ratio SR = 0.5, $base_e$ = 32, $incr_e$ = 32, $freq$ = 2, pct_{3x3} = 0.5, momentum = 0.9, dropout = 0.2, number of filters in the first convolutional layer (num_filters) = 48, and the number of repetitions of the block (Max pooling + fire module + fire module) = 3

training: loss: 0.1225 - accuracy: 0.9855
validation: loss: 0.6358 - accuracy: 0.7801
testing: loss: 0.5242 - accuracy: 0.8000

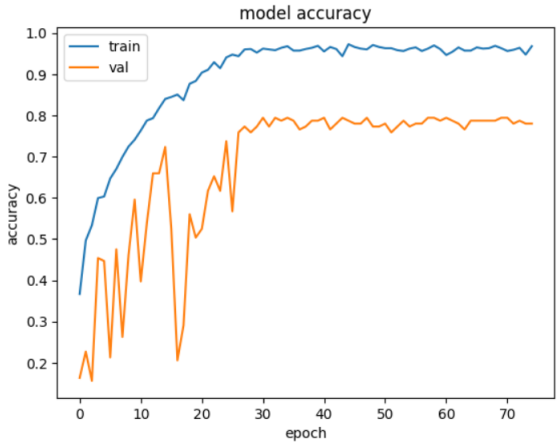

Fig. 3. Training and validation accuracy versus epochs. Squeeze ratio SR = 0.5, base$_e$ = 32, incr$_e$ = 32, freq = 2pct3x3 = 0.5, momentum = 0.9, dropout = 0.2, num_filters = 48

TABLE I
Classification report for Squeeze ratio SR = 0.5, base$_e$ = 32, incr$_e$ = 32, freq = 2pct3x3 = 0.5, momentum = 0.9, dropout = 0.2, num_filters = 48

Class	Precision	Recall	F1-score	Support
0	0.89	0.80	0.84	10
1	0.91	1.00	0.95	10
2	0.75	0.60	0.67	10
3	0.57	0.80	0.67	10
4	1.00	0.80	0.89	10
micro avg	0.80	0.80	0.80	50
macro avg	0.82	0.80	0.80	50
weighted avg	0.82	0.80	0.80	50
samples avg	0.80	0.80	0.80	50

Here is the link to the results by varying the hyperparameters:
Experiment and results

- The accuracy was almost constant for greater than 40 epochs
- The accuracy we got (80 %) was comparable to the accuracy mentioned in the paper
- Batch normalization is crucial for training the model. Without batch normalization, the model was unable to train even on a small dataset (10 images per class)
- The model took very less time ($<$ 3 minutes) to train

V. Conclusion

SqueezeNet architecture was implemented for the classification of wild cat images. The model provided around 78% accuracy for the test and validation dataset. The time taken for the model to train is also less ($<$ 3 minutes) when trained with GPU.

VI. Links

- Link to the Video
- Github link
- Experiments and results

References

[1] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5MB model size. ArXiv. /abs/1602.07360

[2] Gerry. 10 Big Cats of the Wild - Image Classification, Retrieved April 26, 2023, from https://www.kaggle.com/datasets/gpiosenka/cats-in-the-wild-image-classification.

[3] Saife Uddin. "SqueezeNet:—Model Size of 0.5MB, Is It True?" Kaggle, May 26, 2020. https://www.kaggle.com/code/saife245/squeezenet-model-size-of-0-5mb-is-it-true/notebook.