Experimental Analysis of VGG Neural Networks

Maya Madhavan, Sean Zhou, Phudish Prateepamornkul

May 2024

Abstract

The Visual Geometry Group (VGG) neural networks have become a cornerstone of deep learning due to their impressive performance in image recognition tasks. We use the MNIST dataset to perform an experimental analysis of these networks, focusing on their memory capacity, generalizability, and resilience. Our findings reveal that the models theoretically have the same Memory-Equivalent Capacity (MEC) and a generalization ratio less than 1, indicating potential overfitting issues. To address this, we reduce the number of hidden layers and discover that with an MEC that is 1/128th of the original, the models can still achieve perfect accuracy. Despite having the same MEC, the models show varying sensitivity to noise across different VGG configurations. We empirically demonstrate that resilience is not consistent across all VGG architectures, providing insights into how these models can be further optimized.

1 Introduction

In 2015, Simoyan and Zisserman introduced the Visual Geometry Group (VGG) set of models in their paper Very Deep Convolution Networks For Large Scale Image Recognition [3]. The VGG framework popularized using modules within Neural Networks (NNs) by introducing kernel chaining to reduce the overall number of parameters in NNs. Multiple Convolutional Network (ConvNet) configurations are discussed in the paper. Given VGG's place in the field of Deep Learning, there is a need to understand the experimental design of these models. This paper therefore provides an review of the memory capacity, generalizability, and other relevant aspects of the models.

2 VGG Architecture

The original paper includes multiple architectures, which we will refer to from here on as models 1-6. VGG frameworks generally use 3x3 convolution layers and max pooling to reduce dimensions, along with ReLU activation functions. They conclude with three fully connected layers made up of 4096, 4096, and 1000 neurons respectively. The configurations primarily vary in the number of convolutional layers they contain per block, as specified below:

Model	Layers	Total Parameters (Millions)
1	11	133
2	11 (Includes LRN)	133
3	13	133
4	16 (Uses 1x1 filters)	134
5	16	138
6	19	144

Table 1: Summary of Model Configurations

A table from the original paper with details on the specific model structures is available in the appendix.

3 Dataset Evaluation

We perform our assessment on the popular MNIST dataset [1], a solely visual and atemporal set of hand-drawn digits between 0-9 and their corresponding label. The architecture predicts what number is written. Because it is a famous dataset that has been used and reviewed on numerous occasions, the labelling on this dataset has perfect accuracy and we expect close to 100% accuracy in order to claim a successful model. The annotator agreement is unknown for the MNIST data. There are 60,000 rows in the training dataset and 10,000 rows in the test dataset with a relatively balanced distribution of classes in both.

Noise in the data primarily arises from differences in digit drawing styles, as these are quirks that are unique to individual handwriting. The images themselves are normalized, centered, and in grayscale, so we do not expect additional background noise. That said, there could be an overall bias given that this dataset was collected in American High Schools, as those who did not learn English as their first language may write numbers differently.

There may also be data drift as people move towards typing and using tablets to write. However, in general, numbers have stayed the same over centuries so we do not expect major changes over time to handwritten numbers.

We add a 0 padding of size 2 to the original 28x28 images for model compatibility. However, the 0 padding contributes no information to the data. There are thus maximally 28x28x60,000=47,040,000 bits of information in the training dataset, and maximally 28x28x10,000=7,840,000 bits in the testing set. We calculate the memory equivalent capacity of the linear portion of the network in section 4.1.

4 Model Evaluation

We evaluate the original VGG methodologies by comparing the information capacity of our MNIST images, the compression rates of the convolutional layers, and the Memory Equivalent Capacity (MEC) of the feed-forward neural network layer [2].

4.1 Memorization Capacity

VGG models are CNNs with many convolution layers, meaning that images are repeatedly compressed to a smaller dimension then fed into the fully connected linear layers. Despite the distinct VGG model configurations, the input dimension ($28 \times 28 \times 1$) and output dimension ($512 \times 1 \times 1$) of the convolutional layers remain consistent across all configurations. This leads to the same compression ratio of roughly $\frac{28\times28}{512} \approx 1.53 : 1$. Thus, the information content of our train data after compression is $47,040,000 \div 1.53 \approx 30,745,098$ bits.

Furthermore, we evaluate the model's capacity to memorize by calculating its MEC. Again, all VGG specifications have the same three-layer fully-connected network, thus the same total MEC [2]:

- MEC for individual linear layer i, accounting for dropouts ($i = 2, 3, \ldots, k$)

$$\text{MEC}_i = \begin{cases} (d_{\text{in,i}} + 1) \times d_{\text{out,i}}, & \text{for } i = 1 \\ \min\left(\lfloor d_{\text{in,i}} \times p_{\text{drop,i}} \rfloor, (d_{\text{in,i}} + 1) \times d_{\text{out,i}}\right), & \text{else} \end{cases}$$

- Final MEC for the entire network

$$\text{MEC}_{\text{total}} = \sum_{i=1}^{k} \text{MEC}_i$$

In all VGG models, $\text{MEC}_{\text{total}}$ with $k = 3$ linear layers is $2,105,344$ bits. We discover that the first and second model can be trained to achieve full memorization, reaching almost 100% accuracy. This holds true for the larger models when the learning rates are lower. The result is not surprising, as models with the same MEC should be able to achieve memorization with data that has equivalent information content. In section 4.2, we experiment with much smaller hidden layer dimensions.

4.2 Generalizability

We use Generalization metric G given by

$$G(\vec{k}, \vec{p}) = -\frac{\sum_{i=1}^{c} k_i \log_2 p_i}{MEC}$$

Here $c = 10$ is the number of image classes in our dataset, k_i is the number of correctly classified instances in class i, p_i is the probability of the class being i with $i = 1, 2...10$. MEC is the memory-equivalent capacity of our feed-forward neural network. We observe that the numerator is largely dependent on $\vec{k}$, or our overall model accuracy; the denominator is a function of the size and structure of the neural net. Thus the goal is to maximize G by maintaining high prediction accuracy across all classes with the smallest possible linear layer [2].

Hidden layer size	Train accuracy (%)	Test accuracy (%)	MEC (bits)	G (bits/bit)
8	11.24	11.35	4112	5.17
16	62.32	62.90	8224	14.98
32	95.73	96.37	16448	11.59
64	98.74	98.63	32896	5.98
128	98.64	98.35	65792	2.99
256	98.84	98.69	131584	1.50
512	95.16	95.01	263168	0.72
1024	99.18	98.92	526336	0.38
2048	98.97	98.66	1052672	0.19
4096	98.76	98.57	2105344	0.09

Table 2: VGG Generalizability Comparisons

The original VGG model 1 architecture has 4096 neurons in hidden layer and only achieves a generalization metric of 0.09 on our training dataset. This indicates that the neural net potentially uses more parameters than what is needed to memorize the training data, leading to overfitting issues. On the other hand, setting the size of the hidden layer to 32 achieves a 96% testing accuracy with a drastically smaller model that is 1/128th of the original MEC. Shrinking the hidden layer further might not be ideal as it still gives relatively high generalizability but at the cost of accuracy.

Figure 1 showcases the accuracy vs capacity curve for the train and test data, while the generalization results are available in Table 2. Note that the train and test data trends are the same across the MEC values. As long as the MEC is greater than 10,000, we see full memorization or 100% accuracy.

At the point where we chose to stop training our model, the accuracy was 96% and the generalization was 14.9809 for VGG-16. Given this ratio, employing other machine learners is not necessary. Additionally, we calculated the average model resilience that 1 $\frac{bits}{bit}$ of G is equal about 6 dB of added average noise [2]. The result of this calculation is in Table 3. We will first analyze the last 3 hidden layers where the Generalization value is less than 1. In these cases, each of the samples inherently possesses a degree of predictive uncertainty even before any noise is introduced. The result shows the average amount of noise in dB added to each validation instance before the prediction changes.

Hidden layer size	Average Model Resilience (dB)
8	31.02
16	89.88
32	69.54
64	35.88
128	17.94
256	9

Table 3: VGG Average Model Resilience

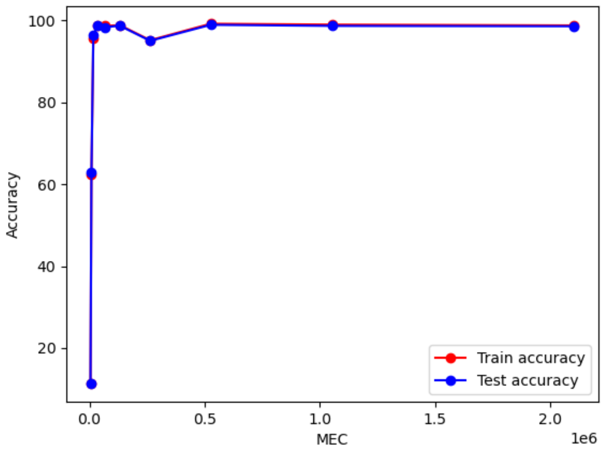


Figure 1: Accuracy Results

4.3 Resilience

To assess resilience, we introduce noise to the data and determine whether the models fails to generalize. We experiment on all model specifications (1-6) and report the results in Table 4 below. We inject values from the standard normal distribution into the original data.

Model	Mean	STD	Train Accuracy (%)	Test Accuracy (%)
1	0	1.0	11.23	11.35
2	0	1.0	11.23	11.35
3	0	1.0	11.23	11.35
4	0	1.0	97.80	97.96
5	0	1.0	97.81	97.78
6	0	1.0	11.236	11.35

Table 4: Accuracy of models under noise conditions

As specified in section 4.2, we anticipate that all original VGG models have $G \ll 1$, meaning that they would be sensitive to noise and overfitting which we

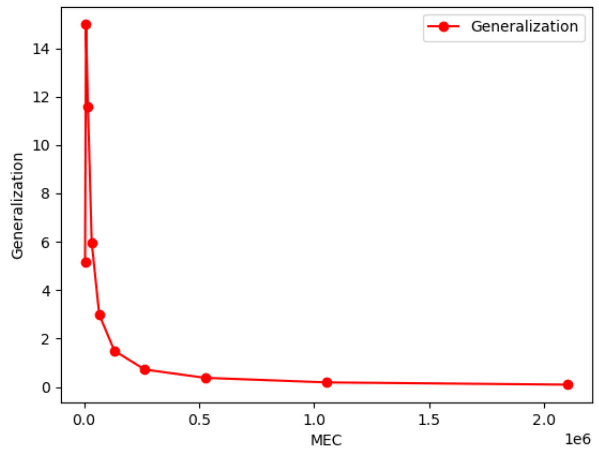


Figure 2: Generalization Results

think that given the low value for generalization and the sensitivity to the noise the resilience for this hidden size is not enough for this task. Additionally, if we changed the hidden size to 64 instead of 4096 as in the paper then the resislence will be enough for this task. Contrary to expectations, we find that we do not get the same results across all models. Specifically, Models 4 and 5 are far more resilient to the added noise and have a higher than expected generalization capacity the other models. This warrants further investigation to understand this result.

As specified in Section 4.2, we anticipate that all original VGG models, with $G \ll 1$, are sensitive to noise and overfitting. This suggests that their resilience is insufficient for the task at hand due to low generalization. However, contrary to expectations, we find inconsistent results across models; specifically, Models 4 and 5 exhibit higher resilience and generalization capacity compared to others. This discrepancy warrants further investigation to understand the underlying reasons. We propose reducing the hidden layer size from 4096, as used in the original study, to 64, which we believe could enhance model resilience.

5 Capacity Progression

We used the Algorithm 11 [2] to obtain the capacity progression and the result is show in Figure 3. Comparing this graph to the Figure 12.1 in the [2] book page 265 we can say that our data is somewhat generalizing which mean that if we want to achieve perfect 100% accuracy then we would require more data.

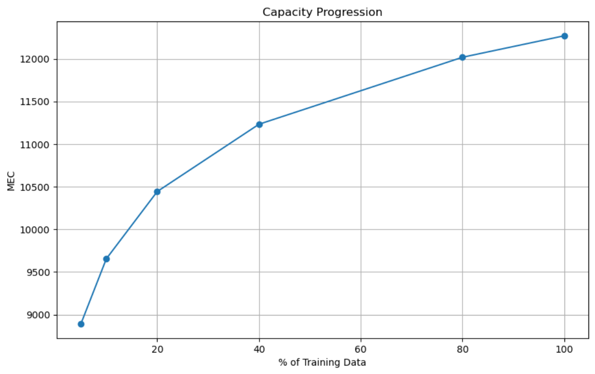


Figure 3: Capacity Progression

6 Conclusion

In this report, we calculate the MEC and generalization ratios for multiple VGG configurations. We further assess the impact of changes in the number of hidden layers on the accuracy, and discover that a significantly lowered MEC maintains perfect accuracy. Lastly, we investigate the effect of adding noise to the different configurations, finding inconsistencies in resilience in two of our models.

While this report focuses on the overarching MEC and generalizability of VGG models, further experimentation around regularization and thresholding may be employed to expand our analysis. Additionally, we could investigate the performances of custom configurations that are even closer to optimal MEC. Since the MNIST dataset is popular and commonly used, we could explore the effects of introducing alternative types of noise to the data.

To reproduce our results, we have made our model weights available here. Additionally, the experimentation process can be found here.

Contribution

Phudish Prateepamornkul worked on implementation of the VGG Architecture, getting the visualization and getting the resilience results. Maya Madhavan worked on the memory equivalent,dataset evaluation and capacity progression. Sean Zhou worked on generalizability and resilence.

References

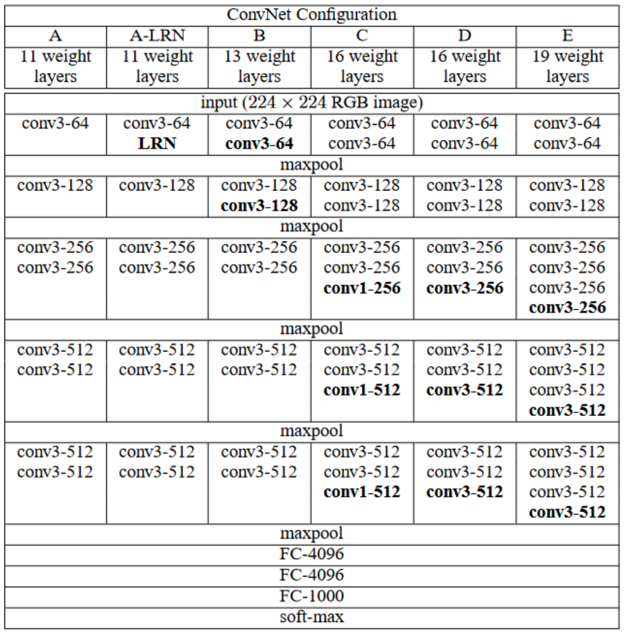
[1] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[2] Gerald Friedland. *Information-Driven Machine Learning: Data Science as an Engineering Discipline*. Springer, 2024.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2015.

Appendix

Figure 4 refer to the model architectures from original paper.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4: Model architectures from original paper