Exploring Lightweight VGG Architectures for Image Classification: A Comparative Study

David Chen Salas
Queens College, City University of New York

Abstract

In this paper, our objective is to investigate the performance of lightweight versions of VGG16. Due to the depth and large number of parameters in VGG16, the model has relatively high computational complexity compared to simpler architectures. In some smaller applications, VGG16 may be considered too bulky. Therefore, our goal is to explore the potential of a lightweight version of VGG16 by reducing its computational layers. T examine the impact of each layer and determine the minimum number of layers required to maintain performance levels simultaneously. This paper presents experiments involving training 5 different VGG models: VGG4, VGG5, VGG6, VGG7 and VGG8. All models are trained for an image classification task aimed at predicting 5 categories. The results show that there is an average increase in prediction accuracy by 9% from each increase layer.

1 Introduction

Deep learning has revolutionized various fields, especially in computer vision tasks such as image classification, object detection, and segmentation. The VGG16 convolutional neural network is known for being simple, performing really well, and being able to learn from other tasks. However, its extensive depth and large number of parameters pose challenges on computational complexity, which makes it tough to use in places where there isn't a lot of computing power.

During my small-scale image classification project involving the classification of only 5 distinct object classes, I faced challenges while trying to use a VGG16 model due to limited computing capabilities. The complex heavy calculations involved in training up the VGG16 overwhelmed my computer system, resulting in frequent crashes. As a result, I started searching for possible solutions and came across research papers discussing lighter versions of the VGG16 model that aim to achieve both performance and computational efficiency. These lightweight models provide a direction in resolving my hardware limitations.

Motivated by this, the present study aims to investigate the performance of lightweight variants of VGG16, namely VGG4, VGG5, VGG6, VGG7, and VGG8, in the context of image classification tasks. The objective is to

systematically reduce the number of layers in the original VGG16 network while maintaining high performance and reducing computational complexity. By exploring the trade-off between model complexity and performance in image classification tasks, this paper aims to seek strategies for designing leaner and more efficient models tailored to specific application scenarios.

2 Literature review

The application of the VGG16 lightweight model can be roughly divided into two approaches: 1) Reduce the number of convolutional layers. This modification is straightforward. We just need to reduce the number of convolutional layers in the VGG16 model. Since the pooling part has 5 blocks, the number of convolution layers in each part is different, such as 2->2->3->3->3 in the original VGG16 model. This way we can try out many possible combinations by selecting the layer to cut. 2) Use only the first part of the VGG16 model structure. VGG16 is mainly divided into two parts: convolutional layer and fully connected layer. In this approach, fully connected layers are removed or adapted to different technologies to build a hybrid architecture model. The main purpose is to obtain feature maps through convolutional layers and fine-tune the feature maps to suit specific tasks.

The approach from the paper 'A Lightweight Model of VGG-16 for Remote Sensing Image Classification'[2] employs a reduced-layer approach. Their model modifies the original VGG16 by changing the three layers into two, which can effectively reduce the parameter of the model, not reduce the accuracy of model recognition and classification, but also improve the training speed of the model. Furthermore, the fully connected layers in their model remain unchanged, with minimal parameter adjustments. The experimental results of their model achieved an accuracy of 95%, surpassing the baseline VGG16 model's accuracy of 79%.

In Ewe et al. (2022) [1] study, they employed the lightweight VGG16 model's second method to enhance hand gesture recognition. They extracted features from convolutional layers and fed the features map into an ensemble classifier, achieving an exceptional accuracy of 99.97%. This surpassed the performance of competing methods at the time. Likewise, Jae Kyu Suhr and Ho Gi Jung[3] adopted a comparable approach to develop a hybrid model for parking slot detection. Their method achieved an impressive accuracy of 99.77%, outperforming existing models for this task available at the time.

3 Methodology

3.1 Method and model

In this experiment, we will adopt the first approaches mentioned earlier to develop five distinct lightweight variations of the VGG16 model, namely VGG4,

VGG5, VGG6, VGG7, and VGG8. The performance of each model will be evaluated through an image classification task aimed at distinguishing between five classes of objects. Further specifics of the task will be provided below. It's noteworthy that the architecture of the fully connected layers in all models remains consistent with that of the base VGG16, except with a reduction in the dense unit to 1024 for lower computational complexity. Figure 1 shows the illustrated architecture for each model. All models have convolution layers of 3x3 filter with stride 1 and always use the same padding and maxpool layer of 2x2 filter of stride 2. The filter number for each block of layer are 16, 32, and 64 depending on the depth of the layer. For example figure 1(e), there are two convolution layers in the first block and each layer performs 16 filters. Second block will perform 32 filter and the last will perform 64 filter to the layer.

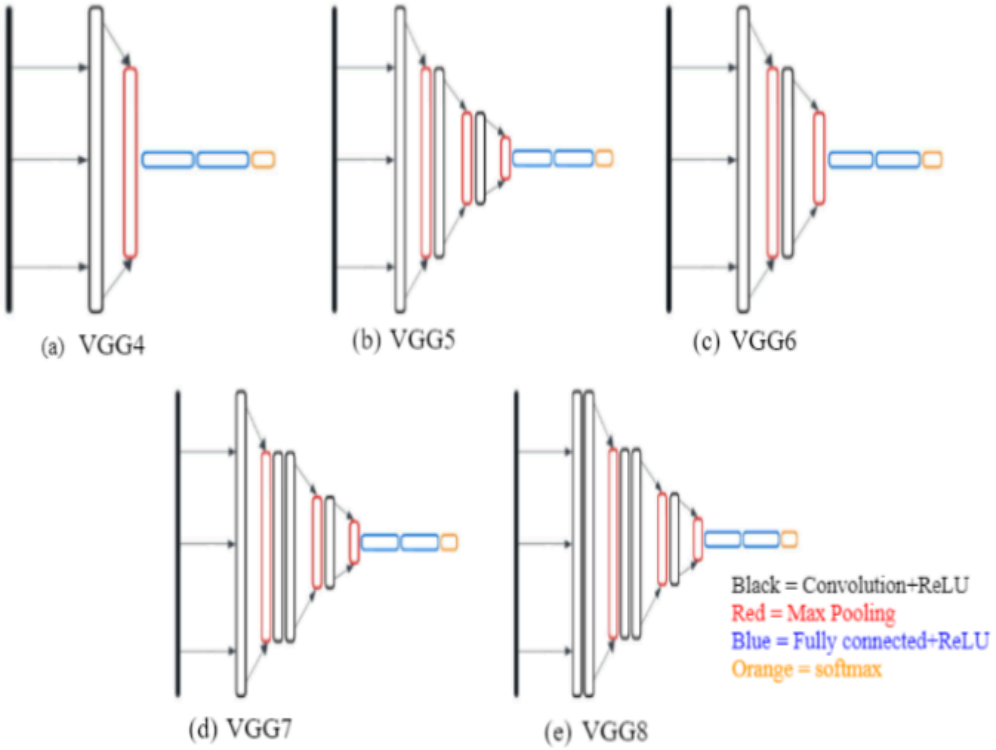

Figure 1: VGG model architecture

3.2 Task and algorithm

Each of the models undergoes training to do an image classification task, tasked with accurately classifying images into one of five distinct categories: cat, dog, cow, sheep, and horse. By evaluating the accuracy and performance metrics of each model across the same set of experimental conditions, aiming to illuminate the relationship between the model's architectural depth and its performance. A valuable insight on how variations in the number of layers within the model architecture influence its overall performance is expected to see. The whole experiment process will describe as below algorithm:

1) Build all models as shown on Figure 1.
2) Train all models with the same set of train data.
3) All models perform classification tasks with the same set of test data.
4) Compute the accuracy of each model in the task
5) Compare the accuracy to find the relationship between layer depth and performance

4 Implementation & Experimentation

4.1 Dataset

The training dataset[4] in this paper experiment was published on the kaggle website by Corrado Alessio. The dataset contains 10 different animal images, but only 5 types are picked for the training since our task is to distinguish the 5 categories, cat, dog, cow, sheep, and horse only. This training dataset contains a total 12840 images, which are composed of 1668 of cat images, 4863 of dog images, 1866 of cow images, 1820 of sheep images, and 2623 of horse images.

Train Dataset

Cat	Dog	Cow	Sheep	Horse
1668	4863	1866	1820	2623

The Pascal VOC2012 dataset[5] will be the test data for the models. But only part of the data set was selectively used for experiments. Images from the 2008-2011 dataset were selected because the validation text files in the dataset only contain label information associated with the 2008-2011 images. There are a total 14205 images and there are only 1746 images belonging to our target class.

Test Dataset
Total: 14205

Cat	Dog	Cow	Sheep	Horse
541	654	152	154	245

4.2 Measurement

We'll primarily focus on comparing the accuracy of each model. Given that our model is trained on a relatively small dataset, there's a chance it might incorrectly classify some irrelevant images as belonging to the target classes. To keep things

simple and since our main goal is to explore how the depth of layers affects performance, we'll only consider recording true positive classifications for computing accuracy. This way, we can better understand how well each model performs in correctly identifying the images that truly belong to the specified classes. Let p(class) as the classify accuracy for the class, and A(m) as the model accuracy computed by the sum of p(class) dividing by the number of classes. Below shown the equation:

$$A(m) = \Sigma p(class) \,/\, number\ of\ class$$
$$p(class) = true\ positive \,/\, total\ \#images\ in\ class$$

4.3 Validation process

The main objective of the validation process is to identify true positive predictions made by the model. The validation process utilizes a test dataset that includes a validation file containing the class label information for each image. This validation file serves as a reference to match the predictions generated by the classification task. If a prediction made by the model matches the label specified in the validation file, it is considered a true positive prediction. Figure 2 illustrates the validation process.

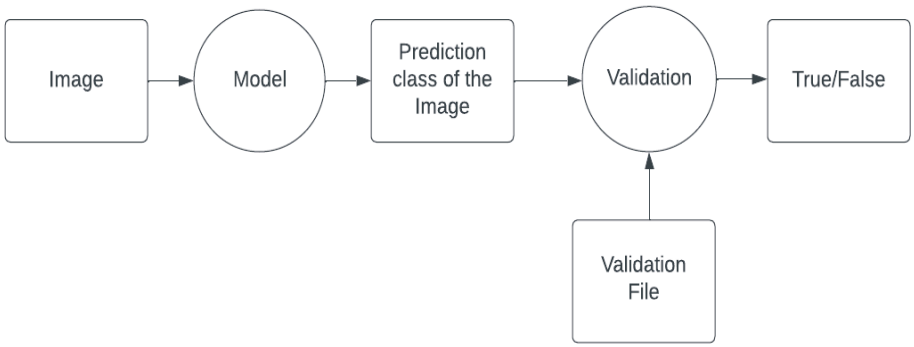

Figure 2: Validation process

4.4 Result

The results show that the accuracy of all models is surprisingly low. However, this result is in line with the overall expected trend, with models with deeper layers tending to achieve higher accuracy. The highest accuracy computed from the VGG8 model has only 35.4% accuracy. The prediction rate p() for each class are 15%, 46.3%, 31.6%, 48%, and 36% respectively for cat, dog, cow, sheep and horse. Table 1 provides a detailed insight about each model's accuracy and prediction accuracy on different classification. By comparing only specific class accuracy, there is no pattern observed based on the statistic. It's hard to tell which model performs better without comparing the average accuracy. On average, each increase in layer depth improves accuracy by approximately 9%. However, it is worth noting that the accuracy of VGG5 is slightly lower compared to VGG4.

This unexpected trend prompted further research into the specific architectural configurations. Despite this, the overall trend highlights the positive impact of increasing layer depth on model performance.

Table 1: Model Accuracy

	Cat	Dog	Cow	Sheep	Horse	Average
VGG4	22.90%	29.60%	9.20%	59.70%	35.90%	31.50%
VGG5	23.80%	27.70%	15.10%	63.60%	26.90%	31.40%
VGG6	18.30%	43.10%	39.40%	24%	35.90%	32.10%
VGG7	14%	39.60%	23%	57.80%	38.30%	34.50%
VGG8	15%	46.30%	31.60%	48%	36%	35.40%

5 Discussion and conclusion

It is known from experiments that more convolutional layers generally lead to better performance, but the results of this experiment are quite disappointing and may not be suitable for drawing conclusions or making inferences. Clearly, larger data sets are needed to further refine the model training process. Additionally, the strategy for selecting which convolutional layers to reduce should be re-evaluated. It is worth noting that there is a significant difference in the performance gain between VGG6 and VGG7 compared to VGG7 and VGG8. This difference is mainly due to architectural differences between these pairs. Specifically, the reduction from VGG7 to VGG6 involves omitting a block of pooling layers, while the reduction from VGG7 to VGG8 involves reducing one layer within a block of pooling layers. Therefore, exploring various combinations of layer reduction in specific VGG model architectures is expected to achieve better optimization and fine-tuning results in future work.

Reference

[1] Ewe, E.L.R.; Lee, C.P.; Kwek, L.C.; Lim, K.M. Hand Gesture Recognition via Lightweight VGG16 and Ensemble Classifier. Appl. Sci. 2022, 12, 7643. https://doi.org/ 10.3390/app12157643

[2] M. Ye *et al*., "A Lightweight Model of VGG-16 for Remote Sensing Image Classification," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6916-6922, 2021, doi: 10.1109/JSTARS.2021.3090085.

[3] J. K. Suhr and H. G. Jung, "End-to-End Trainable One-Stage Parking Slot Detection Integrating Global and Local Information," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 5, pp. 4570-4582, May 2022, doi: 10.1109/TITS.2020.3046039.

[4] Dataset Name: Animals-10, Authors: Corrado Alessio, Publication Year: 2020, URL: https://www.kaggle.com/datasets/alessiocorrado99/animals10?resource=download

[5] The PASCAL VOC2012 dataset, URL:http://host.robots.ox.ac.uk/pascal/VOC/voc2012/