Understanding Weight Transformation using VGGNet Image Classification Models on CIFAR-10 Dataset

Vipul Chalotra

June 23, 2024

Abstract

This report provides an analysis of the effects of weight initialization on neural networks and their transformation over the training period. Specifically, it summarizes observations from models inspired by VGGNet, a well-known convolutional neural network (CNN). The CIFAR-10 dataset, provided by the University of Toronto, is used for training, validation, and testing of these models. The analysis focuses on two VGGNet models trained for the CIFAR-10 image classification task: one initialized with pre-trained ImageNet weights and the other trained from scratch.

The objective is to understand the training process by examining the models' weights throughout the training period. Data collection involves recording the weights of the models at various stages of training. The study compares the differences in training processes, training times, and model weights to understand the impact of weight initialization. The methodologies employed include data augmentation, learning rate scheduling, CNN model training and evaluation, data collection and preprocessing, and various statistical methods. This comprehensive approach aims to provide insights into how weight initialization affects model performance and how weight transforms during training.

The results indicate ...

1 Introduction

In this project, the aim is to examine the performance of VGGNet-inspired models on the CIFAR-10 classification task. The primary objectives are to achieve acceptable classification results and to analyze the weights of the models after the training process to record the differences in the VGGNet "black box" with and without weight initialization. This work is important because it enhances the understanding of the significance of weight initialization, the role it plays in the training process, and how the weights transform over the training period.

2 Model Training

This experiment was initiated by training two VGGNet models to achieve acceptable classification results. In this section, the dataset, the models, and their training processes are described in detail to provide an insight into the overall process.

2.1 Training Data Description & Preprocessing

The data used for model training was obtained from the University of Toronto and is known as the CIFAR-10 dataset[1]. It contains 60,000 32x32 color images across 10 different classes, with 6,000 images per class, as shown in Figure 1. The dataset was divided into 50,000 images for training and 5,000 images each for validation and testing. The classes were equally distributed across the training, validation, and testing datasets, resulting in an initial distribution of approximately 84%, 8%, and 8% for training, validation, and testing datasets, respectively, which was considered suitable for the experiment.

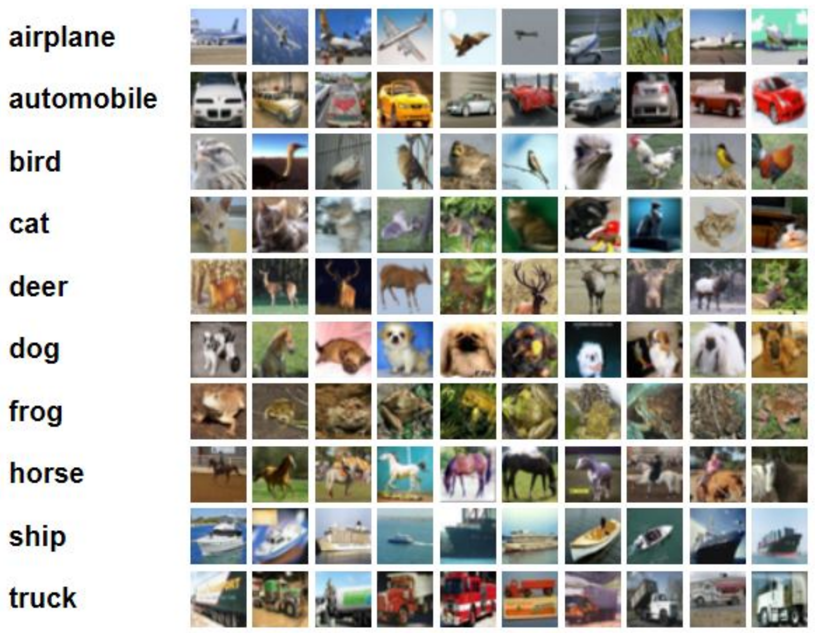

Figure 1: CIFAR-10 Dataset: 10 Random Images from each Class [1]

To expand the training dataset, data augmentation techniques were applied, increasing its size to 200,000 images. The first augmentation involved horizontally flipping the images, doubling the training set from 50,000 to 100,000 images. This was further supplemented with random data augmentations, including a rotation range of 20 degrees and width shift, height shift, shear, and zoom ranges of 0.2 each, resulting in the final augmented training dataset of 200,000 images.

Some interesting insight into the training data is that the CIFAR-10 is a labeled subset of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. ”Automobile” includes sedans, SUVs, things of that sort. ”Truck” includes only big trucks. Neither includes pickup trucks[1].

2.2 VGGNet Inspired Model Structure

The VGGNet models used in this experiment were based on the VGG19 architecture[2], excluding the final dense and classification layers. The convolutional network structure of the VGG19 model comprises five blocks: the first two blocks contain two convolutional layers each, followed by a max pooling layer, while the last three blocks contain four convolutional layers each, also followed by a max pooling layer.

Following the base VGG19 structure, batch normalization, global average pooling, and dropout layers were added. These were further followed by four pairs of dense and dropout layers, culminating in a final dense layer for predictions. The two models trained in this study are based on this VGG19-inspired architecture: one initialized with pre-trained ImageNet weights and the other trained from scratch. According to VGGNet naming conventions, these models can be referred to as VGG22, reflecting the 22 layers with weights. For a detailed view of the model structures, please refer to the associated GitHub repository.

2.3 Pre-Trained Model: Preload VGG22

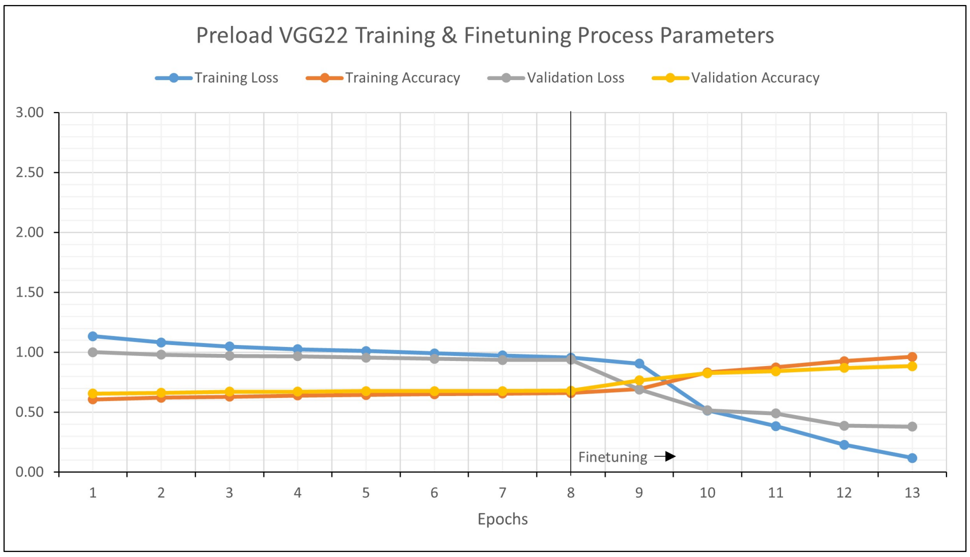

Figure 2: Preload VGG22 Training & Validation Metrics per Epoch

			Training		Validation	
Ep	**Time**	**Learning Rate**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
01	770	1.00E-04	1.1346	0.6071	1.0021	0.6552
02	1189	1.00E-04	1.0818	0.6219	0.9810	0.6618
03	885	1.00E-04	1.0491	0.6299	0.9708	0.6722
04	774	1.00E-04	1.0251	0.6392	0.9666	0.6724
05	847	1.00E-04	1.0107	0.6435	0.9553	0.6762
06	842	1.00E-04	0.9919	0.6500	0.9463	0.6764
07	837	1.00E-04	0.9733	0.6557	0.9365	0.6764
08	901	1.00E-04	0.9551	0.6606	0.9380	0.6798
09	3382	5.00E-05	0.9051	0.6929	0.6896	0.7658
10	3304	5.00E-05	0.5141	0.8331	0.5156	0.8266
11	3355	5.00E-05	0.3851	0.8750	0.4902	0.8420
12	3741	2.50E-05	0.2293	0.9268	0.3878	0.8702
13	3189	7.50E-06	0.1177	0.9631	0.3795	0.8852

Table 1: Preload VGG22 Training & Validation Metrics per Epoch

The first of the two models is the one whose VGG19 base structure was initialized with pre-trained ImageNet weights. For ease of documentation, this model is named 'Preload VGG22'. Additionally, it is referred to as the pre-trained model or the model with weight initialization.

The training process for this model was conducted in two stages. In the first stage, only the final layers added to the base VGG19 structure were trained, while the VGG19 base layers were frozen. This stage was performed for 8 epochs with a learning rate of 1.00E-04, resulting in approximately 67% accuracy. In the second stage, the entire model was trained for five additional epochs with the parameters shown in Table 1. This final training resulted in a testing loss and accuracy of 0.3684 and 0.8840, respectively.

2.4 Model Trained from Scratch: Custom VGG22

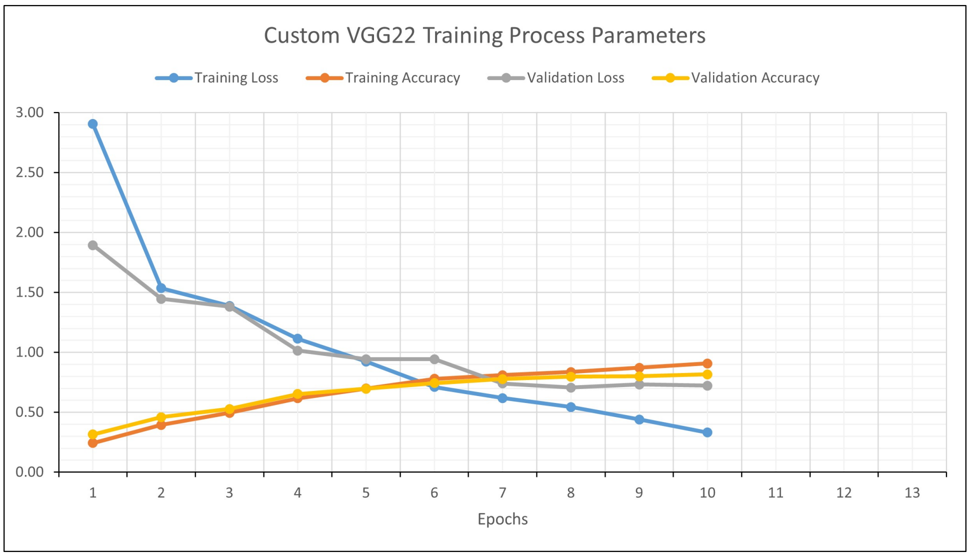

Figure 3: Custom VGG22 Training & Validation Metrics per Epoch

			Training		Validation	
Ep	**Time**	**Learning Rate**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
01	3893	1.00E-04	2.9057	0.2438	1.8941	0.3144
02	3331	1.00E-04	1.5347	0.3949	1.4455	0.4598
03	3805	1.00E-04	1.3873	0.4952	1.3813	0.5268
04	3705	1.00E-04	1.1130	0.6173	1.0152	0.6526
05	3420	1.00E-04	0.9242	0.6980	0.9430	0.6970
06	3402	5.00E-05	0.7113	0.7789	0.9422	0.7426
07	3217	5.00E-05	0.6186	0.8106	0.7402	0.7768
08	3227	5.00E-05	0.5441	0.8361	0.7060	0.7964
09	4049	2.50E-05	0.4404	0.8713	0.7314	0.8008
10	3376	7.50E-06	0.3306	0.9080	0.7224	0.8170

Table 2: Custom VGG22 Training & Validation Metrics per Epoch

The second model is the one that was trained from scratch with random weight initialization instead of pre-trained ImageNet weights. For ease of documentation, this model is named 'Custom VGG22'. Additionally, it is referred to as the model trained from scratch or the model without weight initialization.

The training process for this model was conducted similarly to the Preload VGG22. As shown in Table 2, the first five epochs were trained with a learning rate of 1.00E-04 to achieve results comparable to the initial training stage of the Preload VGG22. Subsequently, the final five epochs were trained in the same manner as the second training stage of the Preload VGG22. This approach was implemented to ensure that the training processes for both models were as similar as possible.

2.5 Preload VGG22 vs Custom VGG22

In this section, we delve into the performance details of the two models based on their training and testing metrics. Observing these differences is crucial as they highlight the benefits of transfer learning in machine learning, a concept well-supported by previous studies and experiments. As shown in Table 3, the Preload VGG22 outperforms the Custom VGG22 across all metrics.

Both models exhibit overfitting to the training data due to very small learning rates in the later stages of training. While a generic VGGNet lacks sufficient regularization to address overfitting, dropout layers were incorporated towards the end of the model architecture to mitigate this issue. Achieving the best possible results without altering the base VGG19 structure involved several weeks of fine-tuning.

The results of this extensive training process are summarized in Tables 1, 2, and 3.

Metric	Preload VGG22	Custom VGG22
Total Epochs	13	10
Total Training Time	06:40:16	09:50:25
Final Training Loss	0.1177	0.3306
Final Training Accuracy	0.9631	0.9080
Final Validation Loss	0.3795	0.7224
Final Validation Accuracy	0.8852	0.8170
Testing Loss	0.3684	0.7031
Testing Accuracy	0.8840	0.8094

Table 3: Comparison of Overall Metrics for Preload VGG22 & Custom VGG22

3 Data Collection & Selection

The experimental approach adopted in this research is fundamental. Given the fact that the CIFAR-10 data was completely structured and properly preprocessed as explained above, the main steps involved were structuring and training the VGGNet models as described in Section 2 and using proper callbacks to record the weights along the training process before further statistical analysis of the weights to gain insights into the learning behavior of VGG22 models.

The data under examination in this project are the weights and the biases of Preload VGG22 and Custom VGG22 models at different times through the training process. Data collection was carried out by an in-built TensorFlow Model Checkpoint callback function. This specific callback allowed storing weights of all the layers throughout the training process. Additionally, learning rate scheduling and manual intervention allowed for a smooth learning process.

After training both the models, four checkpoints were established for model comparison analysis. The first checkpoint was set before any training was performed. The second checkpoint was determined based on the final training point of the Preload VGG22 with a frozen base VGG19 model, which corresponds closely to Epoch 05 of Custom VGG22. The third checkpoint was identified when both models were slightly overfitting after using a learning rate of 5.00E-05. The final checkpoint was the last epoch of training that achieved the best possible validation accuracy despite overfitting.

			Training		**Validation**	
Checkpoints	**Model**	**Epoch**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
Checkpoint 1	Preload VGG22	00	-	-	-	-
Checkpoint 1	Custom VGG22	00	-	-	-	-
Checkpoint 2	Preload VGG22	08	0.9551	0.6606	0.9380	0.6798
Checkpoint 2	Custom VGG22	05	0.9242	0.6980	0.9430	0.6970
Checkpoint 3	Preload VGG22	10	0.5141	0.8331	0.5156	0.8266
Checkpoint 3	Custom VGG22	08	0.5441	0.8361	0.7060	0.7964
Checkpoint 4	Preload VGG22	13	0.1177	0.9631	0.3795	0.8852
Checkpoint 4	Custom VGG22	10	0.3306	0.9080	0.7224	0.8170

Table 4: Comparison of Training & Validation Metrics at Different Checkpoints

Basic statistics from these checkpoints are summarized in Table 4. The weights and biases from the two models at these four different checkpoints in the training process form the dataset for investigating the main objectives of our study.

4 Methodology

In order to understand potential similarities in the weights of the models across the four defined checkpoints, we employed a range of statistical approaches and visualizations. While the methods used are not exhaustive, they are adequate for providing valuable insights into our objectives.

4.1 Descriptive Analysis of Model Weights & Biases

In this section, we performed a basic descriptive analysis on the weights and biases of the two models across all layers. We calculated the means, standard deviations, and ranges at each checkpoint to uncover any significant trends in the weights and biases over time. The results are visualized as heatmaps in Figures 4, 5, and 6.

As shown in the figures, the trends within each model are consistent. However, there are no apparent associations between the two models when comparing their layer-by-layer basic descriptive statistics of weights and biases.

In particular, the Custom VGG22 model exhibits consistently small values for the mean, standard deviation, and range of weights and biases throughout the training period.

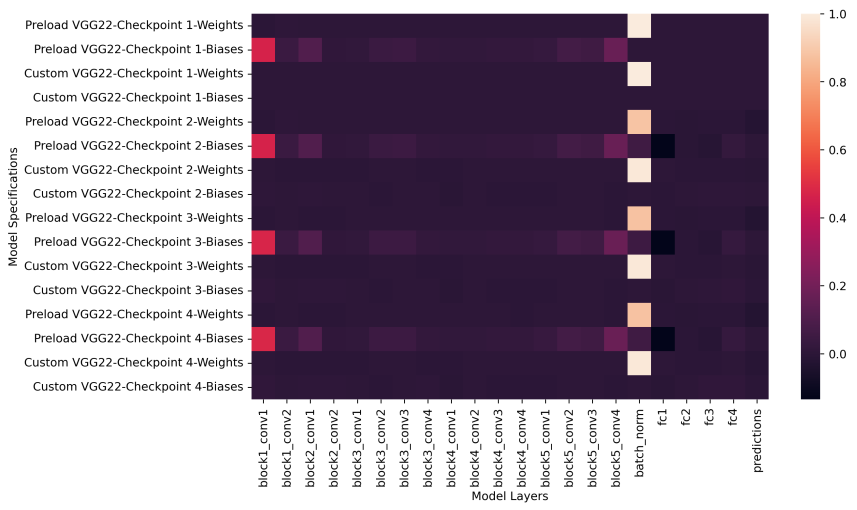

Figure 4: Calculated Mean Values Heatmap

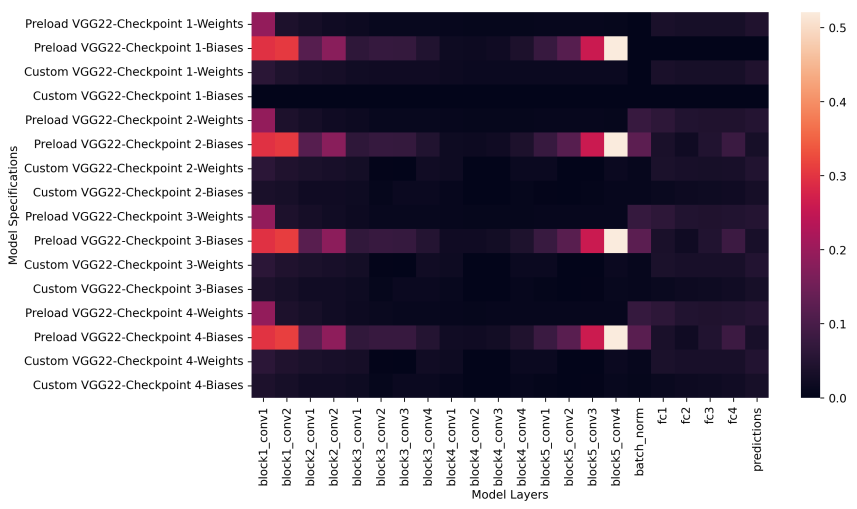

Figure 5: Calculated Standard Deviation Values Heatmap

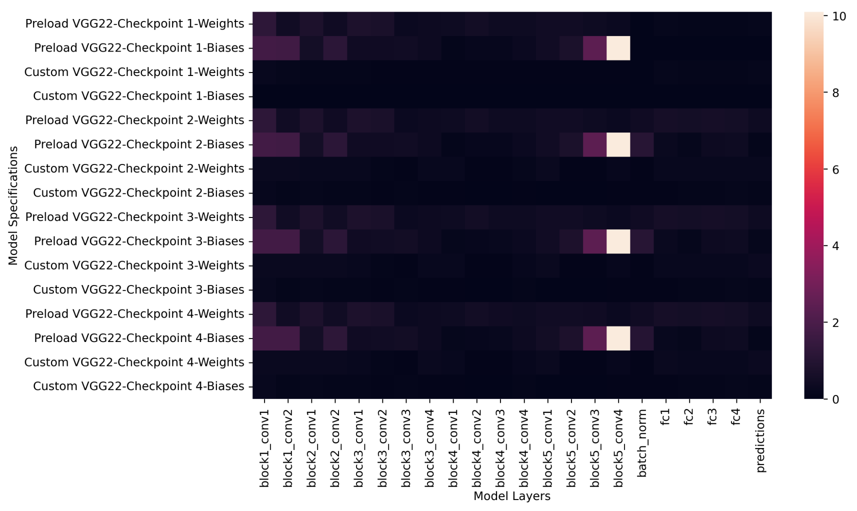

Figure 6: Calculated Range Values Heatmap

Similarly, the magnitude of these statistics for the Preload VGG22 model does not change significantly either, indicating that the initialized weights and biases undergo minimal changes during the training process.

The only notable difference is observed from Checkpoint 1 to Checkpoint 2 in the Preload VGG22 model's final dense layers, which is expected since only the final layers were trained during this interval.

Across the two models, the Preload VGG22 displays higher mean, standard deviation, and range values for certain layers throughout the training process. This suggests that the ImageNet weights retained some learned information that was deemed necessary even for this specific task, which the Custom VGG22 model was not able to learn.

4.2 Correlation Analysis of Model Weights & Biases

4.2.1 Correlation Analysis across Models

In this part of our analysis, we calculated correlations to observe any similarities between the two models based on their weights and biases. We flattened the weights array to create weights vector. The correlation calculations were conducted at the layer level for each model, allowing us to calculate the correlation between the corresponding weight and bias vectors from each model. Both Pearson and Spearman correlation coefficients were computed. The results are visualized in a heatmap in Figure 7.

The analysis indicates no significant correlation between the weights and biases of the two models at each checkpoint for all layers, except for the prediction layer biases as we approach the end of training. Additionally, the white space observed in the heatmap corresponds to all biases and the batch normalization layer weights at Checkpoint 1. This is due to the biases being initialized to zero and the batch normalization weights being initialized to one. Correlation calculations would be insignificant under the condition.

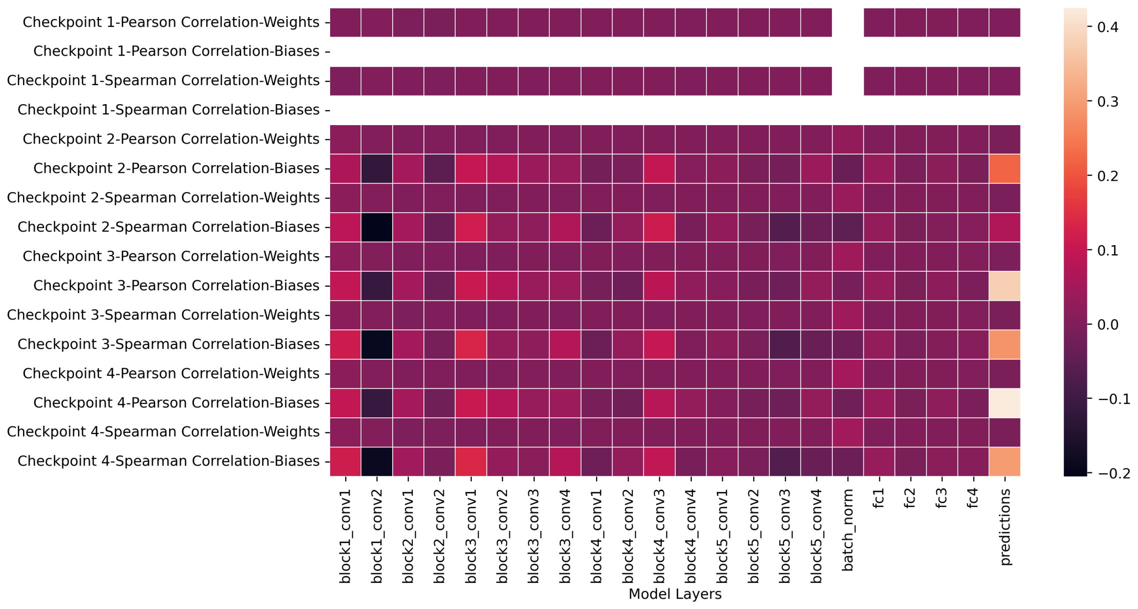

Figure 7: Calculated Correlation Coefficient Values across Models Heatmap

5 Results

The results are summarized in ...

6 Discussion

...

7 Conclusion

...

References

[1] Alex Krizhevsky, Learning Multiple Layers of Features from Tiny Images, 2009.

[2] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014.