VGG Adaptation for Small Scale Scene Recognition

Hansa Srinivasan
Massachusetts Institute of Technology
hansa@mit.edu

Kathryn Bartel
Massachusetts Institute of Technology
kwbartel@mit.edu

Abstract

The task of large-scale scene classification is closely related to the problem of object recognition. This work takes the successful ImageNet Large-Scale Visual Recognition Challenge CNNs, VGG-16 and GoogLeNet, architectures and adapts them for the smaller-scale Mini Places scene classification challenge. These adapted CNNs were used in conjunction with testing-time techniques, such as multi-crop and multiple net averaging, to achieve the lowest errors of the Mini Places challenge. Averaging two adaptations of VGG-16 and using multi-crop yielded the lowest top-5 error of 18.43% and a top-1 error of 45.32% on the competition test set.

1. Introduction

In the last several years, deep learning has emerged as the cutting-edge and most successful technique for object classification, as well as other visual classification. The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has provided a large, on the order of 107 images, object-based image dataset that has been used for establishing the top-performing convolutional neural net (CNN) architectures and their benchmarks, such as Alexnet, the VGG net, and GoogLeNet.

The problem of scene classification is related to that of object classification. The CNNs for both tasks need to learn to recognize various high level features and the combinations that result in certain classifications. The underlying similarity lends credence to the expectation that the deep learning techniques and net architectures that were successful in ILSVRC to be applicable and effective for scene classification. The Mini Places scene classification challenge presents another unique challenge due to the small size of the dataset. With 100,000 training images, 10,000 each of validation and testing images, the Mini Places challenge has a much smaller dataset than the ILSVRC dataset and the Places2 scene dataset. The key component of this paper is adapting the large VGG-16 net for the Mini Places challenge, and its resulting performance.

2. Related Work

This work on the Mini Places challenge builds on work done for ILSVRC. Krizhevsky *et al.*, achieved the best accuracy at the time on the ILSVRC-2012 dataset with one of the first successful large CNNs. This CNN, commonly referred to as Alexnet, established a now-standard architecture of multiple convolutional layers, interspersed with Max pooling layers and followed by fully connected layers. Their work also demonstrated the efficacy of using the Rectified Linear Unit (ReLU) neural model, which increases deep convolutional net training speeds several times faster than same architectures with tanh neural models. Krizhevsky *et al.* also experimented with two important techniques for reducing overfitting: data augmentation using multiple crops and RGB intensity modification, and dropout. Using dropout with a 0.5 probability significantly helped prevent overfitting in Alexnet, though it roughly doubled training time.

Simonyan *et al.* improve on the convolutional architecture proposed by Krizhevsky *et al.*, achieving a top ILSVRC 2014 submission error of top-1 23.7% and top-5 6.85%. The central ideas to their proposed VGG architecture is depth - more convolutional layers - and small kernels. In order to be able to construct a deeper net while keeping the number of parameters from being very large, they used 3x3 kernels in consecutive convolutional layers without pooling between. Multiple 3x3 kernel convolutional layers without pooling have two benefits: more rectified non-linearity, and the effect of a larger kernel with fewer parameters. Simonyan *et al.* also demonstrated the efficacy of using testing-time techniques such as multi-crop, dense evaluation and model combination to reduce error.

Szegedy *et al.* also use multi-cropping and multiple models in testing to achieve better performance. The significant contribution of their work is the idea of inception in the CNN architecture to achieve a deeper net while keeping the number of parameters low, 5 million, much lower than even that of VGG-16. This technique proved extremely effective, and their GoogLeNet architecture also achieved a top ILSVRC 2014 submission.

VGG Adaptations		
VGG-16	Deep WW4-VGG	Shallow WW4-VGG
16 weight layer	8 weight layers	10 weight layers
input (100 x 100 RGB image)		
conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool (2 x 2, stride: 2)		
conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool (2 x 2, stride: 2)		
conv3-256 conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256
maxpool (2 x 2, stride: 2)		
conv3-512 conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512
maxpool (2 x 2, stride: 2)		
conv3-512 conv3-512 conv3-512	conv3-512 conv3-512	
maxpool (2 x 2, stride: 2)		
FC-1024		
FC-100		
softmax		

Table 1. Model architectures. Side-by-side comparison of the original VGG-16 architecture and our adaptations.

3. Neural Net Architectures

Designing a neural network for the Mini Places Challege, one of our main priorities was keeping the number of parameters in the convolutional network low. Due to the success of VGG-16 on ILSVRC, and its comparatively small number of parameters, we decided to use it as the core network for our adaptation. Additionally, Zhou *et al.* was successfully able to use VGG-16 to achieve a top-1 error 52.4% on the full 10+ million image Places2 dataset, giving more reason to use it as the core network. While GoogLeNet performed even better than the VGG networks on ILSVRC and has even fewer parameters, it has a complex structure that is more difficult to adapt without compromising.

3.1. Adapted VGG-16 Architecture

For our adaptation, we wanted the number of parameters of the adapted VGG model to be on the order of Refnet. To accomplish this, we took the VGG 16 layer architecture from Simonyan *et al.* and removed 8 convolutional layers: the last three convolutional layers of size 512 output and

Model	Refnet	WW4-VGG	Deep WW4-VGG
Parameters	2,687,873	3,728,640	8,448,256

Table 2. The number of parameters for each model.

their max pooling, another 512 output convolutional layer, and a 256 size output convolutional layer. All convolutional layers have 3x3 kernels. We also removed 1 fully connected layer, and modified the output size of the last two fully connected layers to be 1024 and 100, respectively. We still have ReLU on all our convolutional layers as VGG, and the same parameters for our convolutional layers - each has a pad of 1, a learning rate multiplier of [1 2] and a decay multiplier of [1 0]. Our maxpooling also has the same parameters as VGG, with a kernel of 2 and stride of 2. We retained a dropout layer with probability 0.5 between our fully connected layers. The resulting adapted CNN, which we call WW4-VGG can be seen Table 1. WW4-VGG has roughly 3,700,000 parameters, which is on the order of Refnet's 2,600,000 parameters. The parameters of each of our models versus Refnet are shown in Table 2.

3.2. Deeper Adapted VGG-16 Architecture

Given the results from Simonyan *et al.*, showing deeper versions of VGG to have better accuracy, we decided to also try a deeper version of the WW4-VGG net. We constructed this deeper WW4-VGG by adding two more convolutional layers of output size 512 with a max pooling after them, as described in Table 1. This allowed us to investigate the effect of a deeper network, however it did cause a large increase in the number of parameters, roughly 2 times the parameters of the shallow WW4-VGG, as seen in Table 2.

4. Classification Framework

Our WW4-VGG architecture, described in the last section, was trained in a similar manner to VGG-16. We also improved our models' performance by using multi-croning and combining models for classification.

4.1. Training

4.1.1 WW4-VGG Training

We training the WW4-VGG model using nearly the same training parameters as Simonyan *et al.* and Zhou *et al.* use for VGG-16. Like both Simonyan *et al.* and Zhou *et al.*, we trained the CNN using mini-batch stochastic gradient descent with momentum of 0.9. We also had an initial learning rate of 0.1 which changed as a step function by a factor of 0.01 after a number of iterations referred to as step size. We different from both Simonyan *et al.* and Zhou *et al.* in our batch size and step size. We used a batch size of 64 due to computational limitations, versus a batch size of 256 and 128, respectively. We also trained two separate models of WW4-VGG. First we trained using a step size of 20,000 as done by Zhou *et al.* (where as Simonyan *et al.* decrease learning rate not after a number of iterations, but after validation accuracy ceases to improve), however, this net trained poorly, its loss function not decreasing steadily over 50,000 iterations from its initial value of 4.6. We then trained another WW4-VGG model from scratch, this time with a step size of 50,000, giving it more time to decrease to a state of non-improving accuracy. This model trained well, with a steadily decreasing loss function, and was trained for a total of 130,000 iterations. We believe the disparity in training success to be due to different initial weight initializations. As stated in Simonyan *et al.*, poor weight initialization can adversely affect learning due to the nature of the learning gradients in deep neural nets. Since the parameters of the competition did not allow fine-tuning off an existing model, we used random initialization where weights were sampled from a gaussian distribution whose standard deviation was 0.01, as done by Zhou *et al.*.

4.1.2 Deep WW4-VGG Fine Tuning

When training the deep WW4-VGG network we decided to fine-tune off the final weights of the trained WW4-VGG model in order to help avoid the problems of random weight. We trained the deep WW4-VGG model with the weights of its first 8 convolutional layers initialized to those of the shallow WW4-VGG model, and the last two convolutional layers as well as the fully connected layers initialized to random weights. We then fine-tuned two different model of the deep WW4-VGG model. The deep WW4-VGG-A model was fine-tuned with the same learning parameters as the final shallow WW4-VGG model described in the previous section, and the fine-tuning ran for 50,000 iterations. The deep WW4-VGG-B model was fine-tuned for 36,707 iterations with all the same parameters except the initial learning rate, which was set to a smaller value of 0.001, which was to address the possibility of overfitting the weights. Both the deep models, however, seemed to experience significant amounts of overfitting.

4.2. Testing

We tested our net using two techniques to improve classification accuracy: multi-crop and net combinations. For multi-crop, we took the image being classified and took 5 crops of size 100x100, one from each corner and then one from the center. We then also took the mirror of each of these crops, and the original image, resulting in a set of 11 images for the image being classified. We then ran each of the 11 images though the net, and averaged the 100-class confidence array, and selected the top 5 confidences from the averaged array as the 5 classifications. The idea from this came from the improvement both Simonyan *et al.* and Szegedy *et al.* saw from using multiple cropping at testing time.

We also combined multiple models to improve accuracy, again motivated by the improved performance reported by Simonyan *et al.* and Szegedy *et al.*. We did this by taking the averaged confidence array of the multiple crops from two or three nets, then averaging these multi-crop confidence arrays to make a multi-crop and multiple model confidence array. We then picked the top 5 classes with the highest confidence as the classifications.

4.3. Implementation

We created and trained our models using the public Caffe toolbox (branched September 2015) [CITE], training each model on single GPUs. All testing was done using the pycaffe interface, including the multi-cropping and model combining.

5. Experimental Results

All experimental results were performed using the Mini Places Challenge dataset, consisting of 100,000 training images across 100 categories, 10,000 validation images, and 10,000 test images. The test images had no classification labels and were used for the purpose of the Mini Places Challenge ranking. In this paper, we will report the results of our various classification strategies on the validation set. All models were exclusively trained on the training set, allowing us to use the validation set as a valid evaluation source.

5.1. Raw Image Evaluation

We tested 3 different models on the validation set: shallow WW4-VGG, deep WW4-VGG-A and deep WW4-VGG-B. We had very similar results for our three models, all within 1% of each other. The shallow WW4-VGG network had slightly better top-5 error than the other models with an error of 22.3%, and the deep WW4-VGG-A had a slightly better top-1 error of 51.43%. The top-1 and top-5 error for each model can be seen in Table 3. All models did significantly better than the Refnet model, achieving a top-1 and top-5 error of at least 11% better than Refnet. Also, despite a much smaller training set, our WW4-VGG models performed similarly to the VGG-16 model Zhou *et al.* trained on the full 10+ million Places 2 dataset, which achieved a top-1 error of 52.4%.

5.2. MultiCrop Evaluation

Our form of multi-crop evaluation significantly improved performance for both the shallow WW4-VGG and deep WW-4VGG-A model, reducing top-5 error by 3.31% and 3.65%, respectively. Multi-crop evaluation also reduced the shallow WW4-VGG model top-1 error by 1.83%, and significantly reduced the deep WW4-VGG-A model error by 4.00%.

Structure	Evaluation Method	Top-1 Error	Top-5 Error
Refnet	—	64.5%	35.1%
Shallow WW4-VGG	—	51.43%	22.3%
Deep WW4-VGG-A	—	51.16%	23.38%
Deep WW4-VGG-B	—	52.81%	23.66%
Shallow WW4-VGG	Multi-Crop	49.60%	18.99%
Deep WW4-VGG-A	Multi-Crop	47.16%	19.73%
Shallow and Deep-A nets	Combination and Multi-Crop	45.26%	17.78%
Shallow, Deep-A , Deep-B	Combination and Multi-Crop	45.06%	17.68%

Table 3. Results. Ours is better.

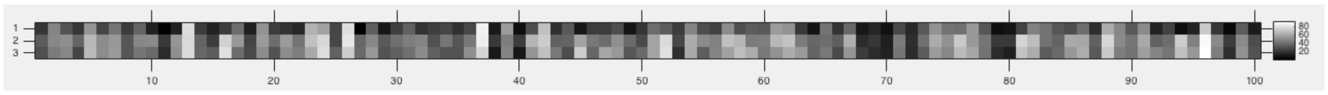

Figure 1. Each row represents the diagonal of a different model's confusion matrix. 1: WW4-VGG, 2: deep WW4-VGG-A, 2: deep WW4-VGG-B. The brightness represents the percentage that the model correctly guesses the class delineated by the number on the horizontal axis.

5.3. Net Combination Evaluation

Using multi-crop in conjunction with multiple nets gave the top classification performance. When combining the shallow WW4-VGG and deep WW4-VGG-A nets with multi-crop, we were able to classify scenes with a top-5 error of 17.78% and top-1 error of 45.26%. Adding the deep WW4-VGG-B net to make a 3-CNN combination classifier only slightly improved performance to a a top-5 error of 17.68% and top-1 error of 45.06%.

6. Conclusion and Future Work

We investigated the efficacy of two VGG-16 adaptations in classifying scenes, and the effect of multi-crop and net combinations at testing time on performance. An interested result was the lack of improvement of performance when using a deeper CNN. We believe the shallow WW4-VGG performed similarly to the deep WW4-VGG nets due to overfitting in the deeper nets. The deeper WW4-VGG nets had roughly 2 times the number of parameters, probably contributing to the overfitting that limited the deep net's performance.

Multi-cropping aided classification performance at test time as expected, and so did using multiple nets in combination. We believe the reason the 3-net combination did not perform significantly better than the 2-net combination is due to the fundamental similarity of deep WW4-VGG-A and deep WW4-VGG-B. From Figure 1. the classification similarities of the 2nd and 3rd models (deep WW4-VGG-A and deep WW4-VGG-B, respectively) can be seen. Figure X shows the diagonal of the confusion matrix for each of the three models, each box represents the percentage of the time that the models first guess is correctly the class on the horizontal axis. The similarity of the 2nd and 3rd rows show that both deep models share similar strengths, whereas they differ significantly, visually, from the first model (shallow WW4-VGG). This means when one of the deep nets is used in combination with the shallow, it provides useful additional information and strengths differing from the shallow net. However, adding another deep net with similar strengths as the existing deep net in the combination does not provide enough significantly different information to improve performance.