

Mini Places Challenge - Team Deepermind

Jeremy Ma

jeremy.ma@mit.edu

Laura Gustafson

lgustaf@mit.edu

Ivan Cisneros

icisneros@mit.edu

Jason Mahr

jasonmahr@college.harvard.edu

Abstract

*Our entry to the 2016 Miniplaces challenge used an ensemble of residual networks. We combined the outputs of three of these CNNs by averaging the outputs of the final softmax layer. Incorporating the outputs of the final 1000-dimensional layer of a state-of-the-art object recognition network trained on the ImageNet database was key to the success of our system. The softmax outputs of the scene and object networks were used as features for a Random Forest classifier resulting in a significant increase in accuracy. To combat overfitting due to the small size of the Miniplaces dataset, we implemented additional data augmentation: we added random rotations, translations, shear transforms and zooms. **Our final system achieved a Top-1 error of 0.4397 and Top-5 error of 0.1644 on the Miniplaces test set.***

1. Introduction

In recent years, the use of deep convolutional neural networks has significantly improved the performance of image classification and recognition. For our final project, we created a system using convolutional neural nets that would accurately classify images in the Miniplaces dataset into 100 different scene categories. To do this we used implementations of various state-of-the-art networks and modified them to work well with the Miniplaces data.

Our report is organized starting with a description of the architectures of the various CNNs in Section 2. In Section 3, we describe the structure of our final system as well as the values of the hyperparameters used to train our models. We report our experimental results on the validation set in Section 4 and summarise our findings in Section 5.

2. Related Work

2.1. Alexnet

The Alexnet architecture used to win the 2012 ILSVRC was the first successful use of convolutional neural net-

works in object recognition [5]. It uses a total of 5 convolutional layers, 3 fully connected layers, max pooling and local response normalisation. Key to the success of this network was the use of ReLU functions for the activation functions of each neuron which are much faster to train than sigmoid functions. Additionally, the architecture includes dropout layers to prevent over-fitting. This operation randomly drops out neurons from the network as it is trained, acting as a form of regularization.

2.2. VGG

The VGG networks, a set of convolutional neural net models developed at Oxford University, feature very small convolution filters (3×3) with depths of up to 16 to 19 weight layers [8]. The VGGS were the first to use smaller filters at a time when large convolutions were popular, as exemplified by the (9×9) or (11×11) filters of AlexNet. The key insight in these networks is that many (3×3) convolutions in sequence can match the performance benefits of larger receptive fields, has been used in more recent architectures such as ResNet and Inception, both discussed below. Fundamentally, it showed that deep networks are extremely important for good performance in image recognition.

2.3. Residual networks

The ResNet architecture by Microsoft Research [2] was another significant step forward in performance for deep convolutional networks. The key improvement made in this design was the use of residual blocks to improve the training of very deep architectures. The residual blocks are composed of a conv-relu-conv layer where the input x is added to the output of the convolutions. The motivation for this is that this 'residual mapping' is easier to learn than the regular unmodified mapping.

2.4. Inception Networks

The Inception architecture developed by Google [10] was designed to have good performance for a convolutional

neural net with memory and computation constraints [11]. In order to achieve this, the network uses an Inception module which performs multiple convolution filters on the inputs simultaneously and combines the results. The success of this network showed that the layers in a deep neural network do not necessarily have to be sequentially stacked for good performance.

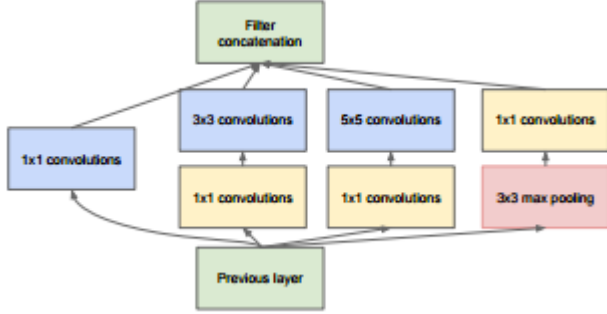


Figure 1. Diagram of an Inception module. [10]

2.5. Batch Normalization

Batch normalization is a method intended for dealing with covariate shifts, which are a naturally occurring effect of parameter tuning via back propagation in neural nets [3]. The effect of internal covariate shifts becomes problematic because of the parametrised, hierarchical nature of a neural net. Batch normalization remedies this by normalising the inputs based on mean and variance.

2.6. Leaky ReLU

Leaky ReLU is an activation function based on the commonly used Rectified Linear Unit (ReLU) [12, 7]. Using the standard ReLU function, the output is $\max(0, x)$. Once the weight becomes negative, and the output is 0, it will never activate. Leaky ReLU fixes this problem by changing the activation function to $\max(\alpha x, x)$ for a small value of α (generally around 0.01).

3. Our Approach

3.1. Overview

Figure 2 shows the overall design of our final system. We used the averaged output of three different neural networks trained on the Miniplaces dataset. In addition, a pretrained network used for object recognition (ImageNet) was also integrated into our system¹. For each test image (224×224), the outputs of the final layers of each neural net were used as feature vectors for a Random Forest Classifier [6].

¹We confirmed with Nick Hynes about the validity of this, and the following piazza post seems to allow this as well: <https://piazza.com/class/istai7i5411g3?cid=334>

The top-1 and top-5 predictions for the Random Forest were extracted using the probability outputs of the classifier.

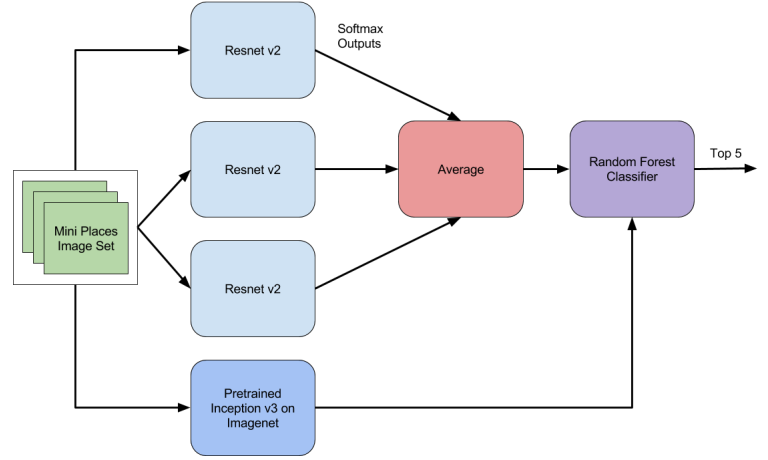


Figure 2. Diagram of our final design architecture.

3.2. Alexnet

A substantial amount of effort went into modifying the provided Alexnet code to match the baseline TA result. To this end, we implemented several improvements to the base Alexnet architecture. In particular we added batch normalization before each of the hidden layers. We also modified the activation functions to use Leaky ReLUs in order to improve the training of the network. Ultimately, the results of experimentation with this network architecture did not lead to good accuracy, so we did not integrate it into our final design.

3.3. ResNet

Our final design uses an average of the softmax outputs of several ResNet architectures trained on the Miniplaces data [2]. We chose to train resnet_50 (50 layers), as opposed to any of the deeper networks in order to reduce over-fitting and to meet GPU memory constraints. The batch size and initial learning rate were set to 100 and 0.001 respectively. We also trained the Inception ResNet architecture [9] using a learning rate of 0.0001, and a dropout probability of 0.8. These networks were trained using an Adam optimizer [4].

3.4. Pre-trained Neural Network

In order to improve our scene classification results, we augmented the results of our trained neural nets with the outputs of the final softmax layer from a pre-trained net on the ImageNet dataset for object detection. The model we used was a pre-trained instance of Inception-V3 trained on the ImageNet dataset. We chose to use Inception-V3 as it is a state of the art model with a low top-5 error (3.46%) and had a pre-trained model that was readily available [11]. In order to add the information from the object detection to

our existing system, we used the outputs of the final layer as features for a Random Forest classifier. Our hypothesis was that a classifier could learn how detected objects relate to each scene and therefore correct for any errors made by the networks trained on the miniplaces data.

3.5. Combining Neural Net Results

We used a Random Forest classifier with 1000 estimators to combine the results of each of our trained neural nets and the results of the pre-trained Inception-V3. The concatenated outputs of the final softmax layers from each of the nets were used as the feature vector for the model. The label was the correct scene classification. The classifier was then trained using the outputs of the neural nets on the 100,000 images in the training set. The top 5 scene labels were obtained from the 5 largest probabilities produced by the random forest.

We used a Random Forest classifier because they provide good out-of-the-box performance and are resistant to over-fitting [1]. They are also able to learn nonlinear decision boundaries, which was crucial for obtaining improvements in performance. When testing with linear classifiers such as an SVM with a linear kernel, we observed a reduction in accuracy, implying that the decision boundary was not linear. In addition, the sci-kit learn implementation of Random Forest has options to parallelize training making it quick and convenient to train. Given the time and computational resource constraints, this made the classifier preferable to other non-linear classifiers such as a Gaussian kernel SVM.

3.6. Data Augmentation

During training we observed significant over-fitting when training deeper networks (ResNet & Inception ResNet). This is demonstrated in Figure 3 where the training accuracy far exceeds the validation accuracy as the number of training iterations increases. To prevent this, we implemented further data augmentation in addition to the operations provided in the baseline code. The baseline code included random crops and horizontal flips in the images. We also implemented random rotations, shear transforms, zooms as well as translation. The effect of this procedure on training can be seen in Figure 4 where the training accuracy stays relatively close to the validation accuracy. The data augmentation operations were especially important for the training of the random forest classifier. This is because we used the outputs of the neural networks on the training set to train the Random Forest classifier. An over-fitted network would produce significantly fewer errors on the training set than on the validation and test sets. This would prevent the classifier from learning a good decision boundary that generalises to unseen images.

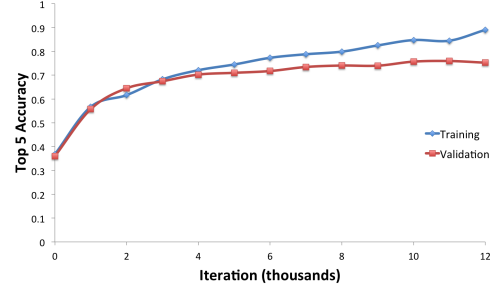


Figure 3. Top-5 accuracy for training and validation sets for ResNet v2 without data augmentation.

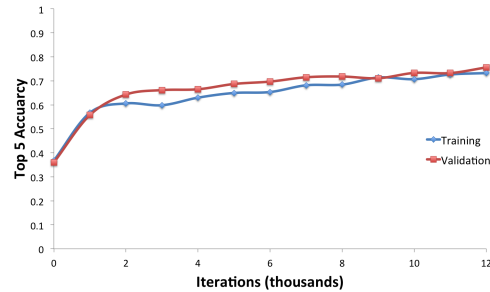


Figure 4. Top-5 accuracy for training and validation sets for ResNet v2 using the augmented data set.

4. Experimental Results

4.1. Software and Machines

Our system used TensorFlow in combination with publicly available implementations² of the state-of-the-art architectures described above. Tensorflow was chosen as it was used by the TA-provided baseline as well as the wide availability of development resources. To train our Random Forest classifier we used scikit-learn. We primarily used a workstation equipped with a Tesla K40c GPU as well as AWS g2 & p2 instances to develop and train our models.

4.2. Results

Table 1 summarizes the top-1 & top-5 accuracies of each of the neural networks trained. The best individual performance from each CNN was attained by an average of the outputs of 3 50-layer ResNet v2 models.

Table 1. Results on the Miniplaces Validation Set for individual Neural Networks

	Validation Accuracy	
	Top 1	Top 5
ResNet V2 (50) (Average of 3)	0.4987	0.7934
Inception Resnet V2	0.3807	0.6729
AlexNet	0.3634	0.6548

We observed a 5% increase in top-5 accuracy when the

²<https://github.com/tensorflow/models/blob/master/slim/nets/>

outputs of the object-recognition network trained on the ImageNet dataset were added as features as seen in Figure 2. Table 2 shows the increase in accuracy as the outputs of each neural net were added to the feature vector. Notably, the addition of outputs from AlexNet decreased the accuracy of the Random Forest Classifier. Ultimately, we found that averaging the outputs of multiple trained ResNet v2 architectures produced the best accuracy when combined with the object-recognition data.

Table 2. Results on the Miniplaces validation set for various ensemble configurations, including outputs from the pretrained ImageNet model

	Validation Accuracy	
	Top 1	Top 5
ResNet (RN; Average of 3)	0.5576	0.8394
RN + Inception RN	0.5121	0.8200
RN + Inception RN + AlexNet	0.4968	0.7957

4.3. Other Attempts

We attempted to train many other variants of networks. These included Inception v3, Inception v4, VGG a, VGG 16, VGG 19, ResNet v1 (50 layers) and CIFARNet. Apart from ResNet v1, which achieved a Top 5 validation accuracy of 0.67 over 20000 iterations, none of the other networks were close to the performance of our ResNet v2 (50), so we did not train these to convergence. Most of these were optimized for larger data sets, which partly explains why they did not work well for the Miniplaces challenge.

5. Conclusion

We found that an ensemble of well-trained neural networks provided the best classification results on the test and validation datasets. Including the outputs of a pre-trained object recognition network provided a significant boost in validation and test accuracy. This demonstrates that different modalities of data, specifically the objects in an image are important in scene recognition. The use of data augmentation was also important for preventing overfitting of the neural networks given the relatively small size of the miniplaces dataset. Another useful enhancement was averaging over the outputs of multiple networks. Lastly, the Random Forest Classifier we used was able to combine the outputs from both scene and object recognition networks to boost classification accuracy.

6. Further Work

Given more time, we would experiment further with the ResNet architecture. Due to time and computational constraints, we only trained the 50-layer networks. Adding or removing layers could improve training and increase performance. Additionally, more types of data augmentation

could further reduce the likelihood of over-fitting. For example we did not implement color jittering, a common data augmentation technique. In addition, performance could also be improved by using other more accurate pretrained network architectures to generate object data such as Inception v4.

7. Individual Contribution

I made several contributions to the final system used in our Miniplaces submissions. I performed the required modifications to the helper code used to train and test the networks. In addition, I was also responsible for the implementation of data augmentation, batch normalization in Alexnet as well as model averaging. I trained several Neural Networks with and without additional data augmentation including ResNet v2 and Inception ResNet. Finally I also came up with ideas for improving classification including incorporating features from the pretrained ImageNet model as well as model averaging.

References

- [1] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- [3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
- [4] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [6] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
- [7] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.
- [8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [9] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.
- [10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. 2015.
- [11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.
- [12] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.