# Scene classification with Convolutional Neural Networks

Josh King
jking9@stanford.edu

Vayu Kishore
vayu@stanford.edu

Filippo Ranalli
franalli@stanford.edu

## Abstract

*This paper approaches the problem of classifying scenes from the Places2 dataset using a simple baseline model, along with 18, 34, and 50 layer ResNet architectures. Model ablation is used to determine the effect of each of the architecture components on the overall performance on the task, and saliency maps are computed to interpret the model learning behavior when making a prediction. The results show that the 3 ResNets achieve very similar results on a downscaled dataset.*

## 1. Introduction

We investigate the problem of scene classification, in which scenes from photographs are categorically classified. Unlike object classification, which focuses on classifying prominent objects in the foreground, scene classification uses the layout of objects within the scene, in addition to the ambient context, for classification. The study of scene classification predates the use of convolutional neural networks (CNNs), where, for example, previous techniques include the use of codebooks and bag-of-word models[10].

In this paper, we examine using supervised learning to train convolutional neural networks to categorize scenes into a predefined set of categories.

## 2. Problem Statement

We investigate how to train a CNN that can classify scenes with high Top-1 and Top-5 accuracy. We use the Places365-Standard scene dataset, which contains 365 categories with 1.6 million labeled images in the training set, 50 images per category in the validation set, and 900 images per category in the test set[18].

In addition, through our exploration of training CNN's for this purpose, we seek to learn if there are certain neural net features that result in good performance for scene classification.

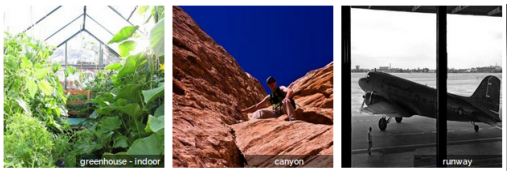


Figure 1. Sample labeled images from the Places2 dataset

### 2.1. Dataset Description

There are several different versions of the Places2 dataset. There is a Standard version, which consists of consists of 365 classes with 5,000 training images per class, and 100 validation images per class. There is also a Challenge version which has an additional 25,000 training examples per class. Both of these are available in low-resolution (256x256) and high-resolution (512+x512+). Examples of images and their labels can be seen in Fig.1.

### 2.2. Dataset Processing

While we use the entire Places365-Standard dataset for training our model, we reduce the resolution of the 256x256 images to 64x64 to allow us to run the images without running out of memory on a Google Cloud Instance (our training architecture requires us to have the entire training set in memory). This leads to a substantial decrease in the quality of the images. Manual visual inspection suggests it is much more difficult to classify the images when they are reduced to such low resolution.

## 3. Technical Approach

### 3.1. Baseline

A simple, but capable baseline was first deployed to gain insight on the problem and to have a solid benchmark for more powerful models. The architecture of our baseline is described in the flowchart in Fig.2. This model features a ReLu non-linearity preceded by batch normalization at every layer, 2x2 maxpooling after the first convolution and a series of progressively deeper and narrower CNN layers. The final layer is a dense layer of output size 4096, as can be seen in the VGG architecture [5]. The dense layer fea-

tures dropout to reduce overfitting. We expect this model to be harder to train than one that contains residual layers.

In the diagram, the term "Plain" refers to a layer that does not implement residuals, as opposed to "Residual" where the layer can indeed learn a margin (Fig.4)[4].

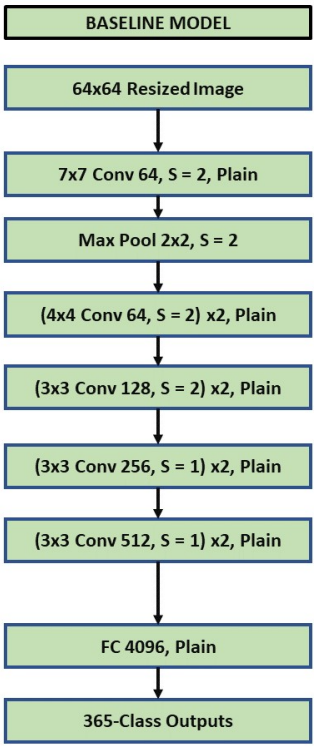


Figure 2. Baseline Architecture

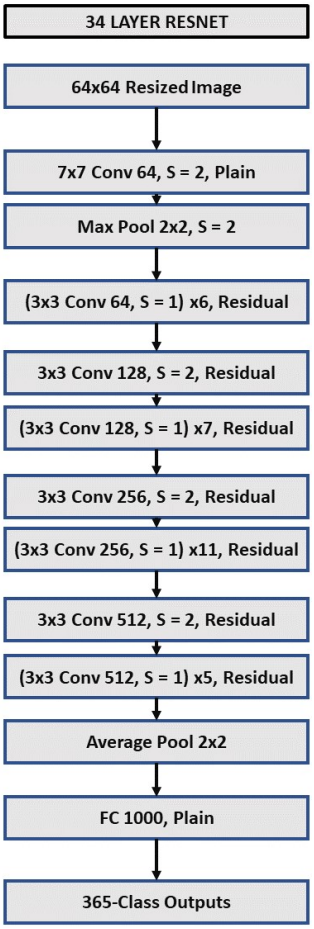


Figure 3. ResNet 34 Architecture

### 3.2. ResNet

In particular, for this report, we implemented and trained several variants of ResNet[5] with the hyperparameters summarized in Table 1, which utilizes residual modules to learn residuals between the output of a set of layers and their input. In the table, "Residual Layer Span" describes the number of convolutional layers that the residual layer learns residuals over. We implement the architecture described in [5] using TensorFlow[1]. Additionally, as suggested in the ResNet paper, in order to allow a residual layer to span convolutional layers with multiple dimensions, we zero-pad the channel dimension of the output of the layers to maintain a consistent size with the residual. When it comes to residuals between layers of different filter dimensions, we downsample the higher-dimensional residual with a max pool of stride and size of 2. Alternatively, we address the dimensionality shift in the residual layers with a projection matrix $W_s$ as described in the paper, as this yields slightly better performance than zero-padding the depth and downsampling the height and width. These projections are

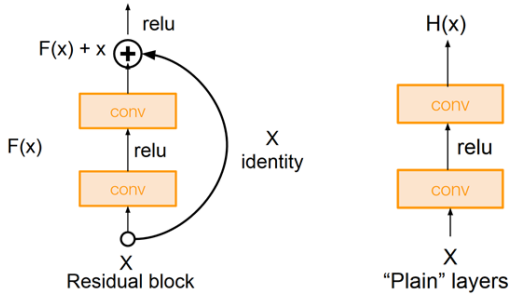


Figure 4. Plain VS Residual Layer

necessary when a residual has a different 2d spatial dimension from the output that it is being added to. For example, The first residual used in ResNet 34 is of size 15x15x64, but it is being added to an output of size 8x8x128. To take care of the difference between 15x15 and 8x8 there are two options. We can either do average pooling with a width of 2 and stride of 2 to turn the 15x15 into 8x8, or we could construct projection matrices of learnable parameters that will

| Hyperparameter | Value |
|---|---|
| Minibatch size | 100 |
| Initial Learning Rate | $10^{-4}$ |
| Gradients Clipped to | 5.0 |
| Batch Norm Momentum | 0.99 |
| L2 Strength | $10^4$ |
| Dropout | 0.85 |
| Residual Layer Span | 2 |

Table 1. Description of ResNet hyperparameters

project the 15x15x64 down into 8x8x64. If $R \in \mathbb{R}^{w \times h \times d}$ is the residual and $S \in \mathbb{R}^{w' \times h' \times d'}$ is the output of the next layer, then if $w' \neq w$ and $h' \neq h$ then two matrices $P_w \in \mathbb{R}^{w' \times w}, P_h \in \mathbb{R}^{h' \times h}$ of learnable parameters are constructed, and then a new Residual $R' \in \mathbb{R}^{w' \times h' \times d}$ can be constructed as:

$$R' = ((P_w R)^\intercal P_h)^\intercal$$

Where both transposes just switch the first and second dimensions. $R'$ is then of the correct area. If $d' \neq d$ then we zero pad $R'$ to make it $R' \in \mathbb{R}^{w' \times h' \times d'}$.

### 3.3. Dropout and L2 Regularization

Because the evaluated ResNet models tend to drastically overfit the training set, dropout[14] on the fully-connected layer was deployed, together with L2 weight regularization. At train time, dropout drops the activations of the FC with a probability $p$, whereas at test time it does not drop any of the units but it must scale the activations by multiplying by $p$ such that the expected output at train and test time match. While dropout helps with overfitting by reducing co-adaptations of features, L2 regularization pushes the weights $W$ to zero proportionally to its strength $\lambda$ for the layer $j$.

$$L \leftarrow L + \lambda ||W_j||^2 \tag{1}$$

### 3.4. Batch Normalization

To help accelerate the training and to allow higher learning rates, we implement batch normalization after the input and after every convolutional and fully connected layer. When a ReLu non-linearity is present, the batch normalization is placed right before it. Batch normalization behaves differently at train and test time, normalizing each minibatch in the former (Eq.2) while normalizing the entire data with accumulated statistics from the train data in the latter.

$$x_i \leftarrow \gamma * \frac{x_i - \mu_B}{\sqrt{\sigma_B + \epsilon}} + \beta \tag{2}$$

Here $\beta$ and $\gamma$ are the learnable scale and shift parameters for each mini-batch during the training phase, and $\epsilon$ helps with numerical stability. Moreover, $\mu_B$ and $\sigma_B$ are the mini-batch mean and standard deviation. At test time, the whole data is normalized once by the moving averages of all the training mini-batches.

### 3.5. Loss and Optimizer

The loss adopted for the task is a Softmax, defined by Eq.3.

$$L = \frac{1}{M} \sum_i^M (-s_{y_i} + log \sum_j^C e^{s_j}) \tag{3}$$

Where $M$ is the batch size, $C$ are the classes, $s_{y_i}$ is the correct label score and $s_j$ are the incorrect label scores for the data point $i$. The optimizer adopted to minimize the loss is Adam[7], and the learning rate is annealed according to an exponentially decaying schedule.

### 3.6. Saliency Maps

Saliency maps are a very useful tool to determine where the network is looking to make its classification decision. It is interesting to examine the saliency for the correct and incorrect Places2 predictions, as well as comparing saliency maps from networks trained on ImageNet to understand what pixel locations matter most for each problem. Mathematically, the saliency $S$ is the maximum of the absolute value of the gradient of the correct class score with respect to the image pixels, as seen in Eq.4.

$$\begin{aligned} s_j &= (WX)[j] \\ S &= max(abs(\nabla_X s_j)) \end{aligned} \tag{4}$$

Where $s_j$ is the correct class score for image X, $j$ is the correct class label index, and $WX$ is the forward pass output of the trained classifier network $W$ of dimension 365 (number of classes).

## 4. Results

We trained an 18-layer ResNet, 34-layer ResNet, and 50-layer ResNet in Tensorflow on the full downsampled training dataset, validating against the validation set. For reference the leaderboard of the Places365-Standard dataset currently reports a 50% best Top-1 accuracy.

| Model Feature | Accuracy % Increase |
|---|---|
| Batch Normalization | 1.83 |
| Dropout | 1.49 |
| L2 Regularization | 0.24 |
| Residual Learning | 4.12 |
| Residual Projection $W_s$ | 0.57 |

Table 2. Ablation Study on ResNet 34

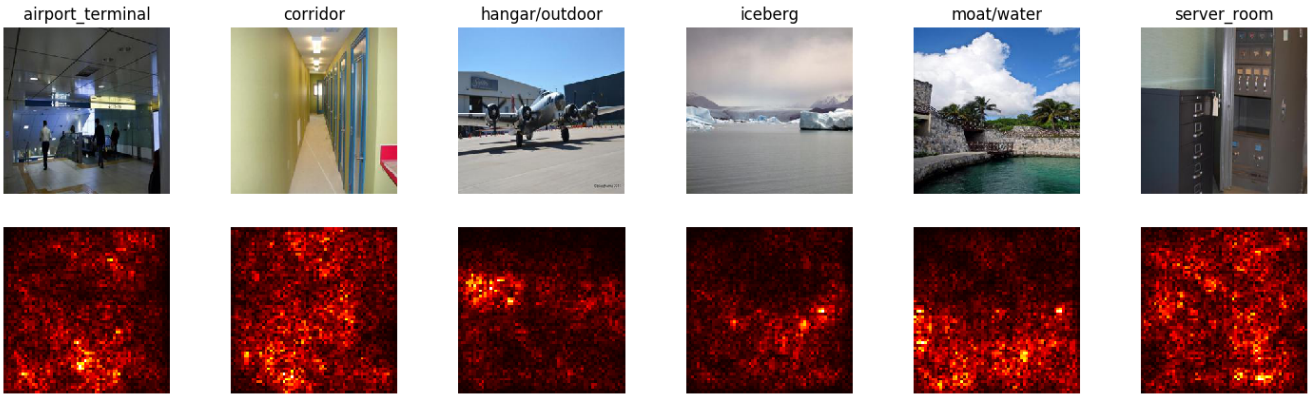


Figure 5. Examples of correctly classified scenes with associated saliency maps

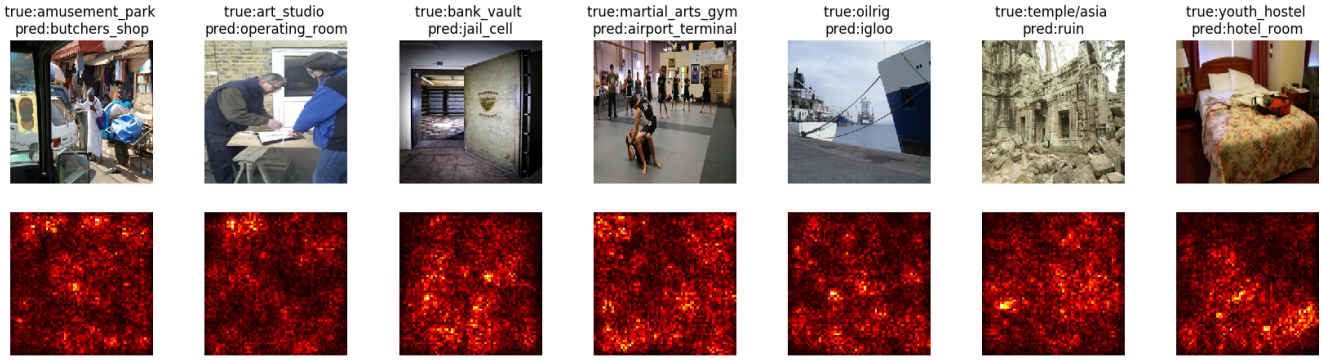


Figure 6. Examples of incorrectly classified scenes with associated saliency maps

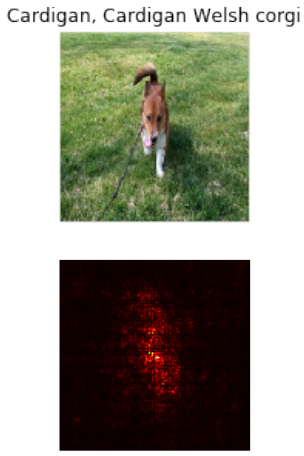


Figure 7. Example image of SqueezeNet saliency map on ImageNet image. Generated from [15]

### 4.1. Baseline Results

The baseline model performs reasonably well. The top-1 on the validation set is 28.5%. It is also useful to note that while it overfits slightly with a training accuracy of 30.9% it is by far the model that overfits the least. However, the training accuracy curve levels off, indicating that it is not powerful enough to capture more aspects of the dataset. Furthermore, this supports our expectation that the baseline model is more difficult to train since it does not use residual layers.

### 4.2. ResNet Results

For top-1 accuracy, all of the ResNet models performed similarly, but the 34 layer model was consistently the best performer with 38.8% accuracy on the validation set Table 3. However, as can be seen in Fig.11 Fig.10 Fig.9, all of the ResNet models suffer from significant overfitting but we were unable to improve performance with stronger dropout and L2 regularization. Furthermore, the training accuracy curves do not flatten out, indicating that our model is sufficiently powerful. We believe that if we were able to keep the images at 256x256 (or at the 512+x512+) resolution and were able to use the extra 25,000 images per class in the

Challenge dataset, our models would underfit less.

### 4.3. Ablation Results

As can be seen in Table 2, residual learning, batch normalization, and dropout are the features of the ResNet model that contribute most significantly to the accuracy of our model. With such overfitting, it is surprising that L2 regularization of fully connected layers is so irrelevant, but since there are only two fully connected layers, it may be that the majority of overfitting is happening in the convolutional layers.

### 4.4. Top-n Accuracy Results

The most commonly used metric on the Places2 dataset is top-5 accuracy where a prediction is deemed correct if the correct label is among the top-5 it predicts. There are several reasons for this metric being the most appropriate. First, the given class labels are hard labelings with only one correct answer. This is not an complete representation of the real world as there may be ambiguity in labellings. For example in Fig.6 ResNet predicted a picture as a ruin when the correct label is an Asian temple. Examination of the image itself indicates that both labels are correct, since it appears to be a ruined Asian temple, a composition of two scenes. In this case, it is very likely that Asian temple is among the top-5 labels predicted by the classifier and it would be wrong to penalize the classifier for this prediction. This ambiguity in Places2 is discussed by Wu et al.[17]. We extended this idea a bit to create top-n accuracy, where we count a prediction as correct if the the true label is in the top n predicted labels and we vary n. Results of this for $n = \{1, 2, 3, 4, 5\}$ can be seen in Fig.12. All of the models are very consistent in their performances on the various top-n accuracies. This is useful because while classifiers for Places2 are generally evaluated using top-5 accuracy, they are generally optimized using cross entropy loss which optimizes for top-1 accuracy. So implementing a loss function that optimizes for top-5 accuracy may yield some performance gains on top-5 accuracy. Also, the large increase in accuracy from top-1 to top-5 shows that most of the time, the correct label is very close to the top. Related to this idea of top-n accuracy Fig.13 shows the frequency of the predicted rank of the correct labels, e.g. 38% of the time the true label is rank 1 or the most likely predicted label. The graph shows that while the top-1 accuracy is only about 38% the true label is always quite close to the top, and almost always in the top 20. This indicate that when the classifiers mislabel an object, they are not too far off.

### 4.5. Saliency Maps

In Fig.5 and Fig.6, we show some example images from Places2 and their corresponding saliency maps. In contrast to saliency maps from a CNN classifier for ImageNet for object detection which focus on the individual objects that are the labels (Fig.7), the saliency maps generated by our model are sensitive to many different points around the scene. This suggests that scene classification has substantial differences from object classification. In some cases where an object defines a scene, our classifier exhibits behavior similar to classifier for object detection. For example, the map for the outdoor hangar is most sensitive to the pixels representing the hangar itself.

Furthermore, inspection of the saliency maps yields some insight into why the classifier chose its labels. For example, in Fig.6 the classifier is ignoring the people in the picture and instead is looking mostly at the patterns on the floor to predict this picture as an airport terminal. While this is an incorrect prediction, an example of a correctly predicted airport terminal can be seen in Fig.5. In this picture, the classifier is also mostly looking at the pattern on the floor which does look similar to the incorrectly classified picture. This is a case where we believe that higher resolution input images would help our model better distinguish between the two classes.

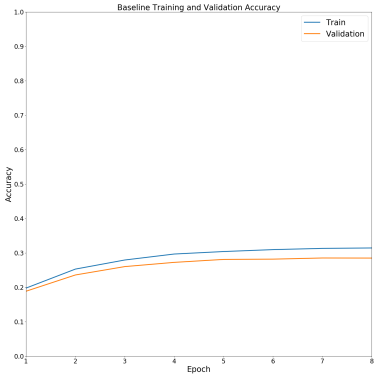


Figure 8. Top-1 Training Accuracy per Epoch for Baseline Model

| Architecture | Training | Validation |
|---|---|---|
| Baseline | 30.97 | 28.53 |
| ResNet 18 | 56.41 | 38.19 |
| ResNet 34 | 59.76 | 38.57 |
| ResNet 50 | 62.01 | 38.76 |

Table 3. Top-1 Accuracy all models after 8 training epochs

### 4.6. Data Quality

In inspecting misclassified results, we encountered some data quality issues. For example, the first image in Fig.6 is labeled as an amusement park, but does not seem to be one. This is an issue inherent to the dataset, and so do not expect this to impact our model's performance compared to other models which use the Places2 dataset.

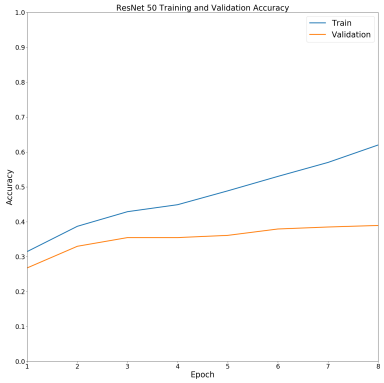


Figure 9. Top-1 Training Accuracy per Epoch for 50-Layer ResNet

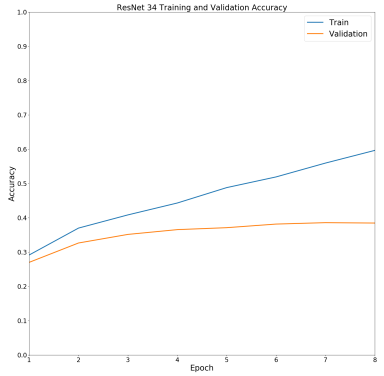


Figure 10. Top-1 Training Accuracy per Epoch for 34-Layer ResNet

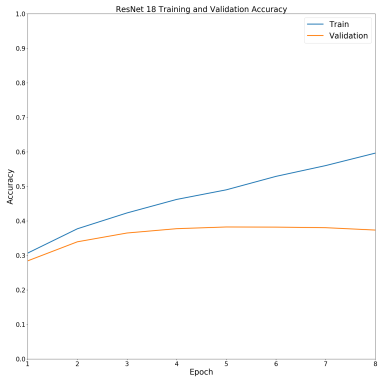


Figure 11. Top-1 Training Accuracy per Epoch for 18-Layer ResNet

## 5. Conclusion

Based on our results, though we perform well on the training set, our performance on the validation set is much lower due to overfitting. Other papers that have results from various models on Places2 do not seem to have similar issues with overfitting, but they do not downsample the images to 64x64 and most papers also train on the Challenge dataset so we suspect that our problems are mostly due to the downsampling of the images and only using the Standard dataset. After manually looking at examples of the images in 64x64 resolution and the corresponding predictions it is surprising how well the models perform.

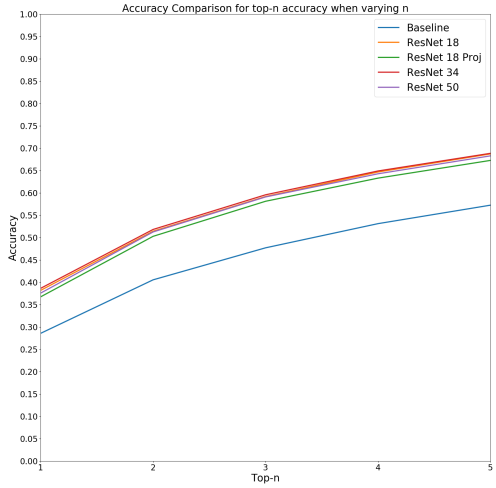


Figure 12. Top-n Validation Accuracy for all models across different n

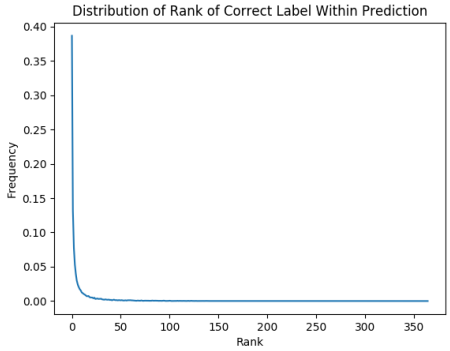


Figure 13. Frequency of the rank of the correct label within the predictions for ResNet 34

## 6. Future Work

Results of the same models on varying resolution images and varying the amount of training data would be interesting. It may also be interesting to compare our results and saliency maps to a model trained by transfer learning on a model for object detection.

Also, due to the differences between the saliency maps on ImageNet and on Places2 it might make sense for scene classification models to be more specialized rather then being as similar to object classification models as they cur-

rently are. Adopting techniques like attention and glimpsing to work on scene classification could take advantage of the unique challenges of scene classification.

More work could also be done on cleaning up incorrect labels and ambiguous labels in the Places2 dataset. As can be seen in Fig. 6 there are images in the dataset that are labeled incorrectly. When the model trains on these images it will learn an incorrect label making it not generalize as easily. Another problem is the ambiguous labels that were mentioned before. If Places2 were amended to allow several different correct labels per picture that would be more representative of the real world and might yield models that generalize better. However, both of these changes would require immense amounts of manual work and may not be feasible.

## References


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[2] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification using a hybrid generative/discriminative approach. *IEEE transactions on pattern analysis and machine intelligence*, 30(4):712–727, 2008.

[3] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[4] L. Fei-Fei, J. Johnson, and S. Yeung. Cs231n course lecture slides.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems*, pages 1378–1386, 2010.

[9] D. Lin, C. Lu, R. Liao, and J. Jia. Learning important spatial pooling regions for scene classification. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3726–3733, June 2014.

[10] N. Rasiwasia and N. Vasconcelos. Scene classification with low-dimensional semantic spaces and weak supervision. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[11] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *European Conference on Computer Vision*, pages 467–482. Springer, 2016.

[12] C. Siagian and L. Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE transactions on pattern analysis and machine intelligence*, 29(2), 2007.

[13] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[15] C. Staff. Cs231n assignment 3 code.

[16] L. Wang, S. Guo, W. Huang, Y. Xiong, and Y. Qiao. Knowledge guided disambiguation for large-scale scene classification with multi-resolution cnns. *IEEE Transactions on Image Processing*, 26(4):2055–2068, 2017.

[17] R. Wu, B. Wang, W. Wang, and Y. Yu. Harvesting discriminative meta objects with deep cnn features for scene classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1287–1295, Dec 2015.

[18] B. Zhou, A. Khosla, À. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *CoRR*, abs/1610.02055, 2016.