

Assignment 3

CSC 411/2515

Team

Infinite Improbability Drive

Jean-Francois Lafleche

1003727363

Shayne Lin

998701542

Table of Contents

I. Introduction.....	1
II. Training Data Analysis.....	1
Ambiguity and errors	1
Class Imbalance	2
III. SIFT features	2
IV. Inception-v3 CNN model	3
Transfer learning	3
CNN + SoftMax	4
CNN + KNN.....	5
CNN + SVM (with PCA)	5
V. Ensemble methods	6
VI. Comparison between methods.....	6
Features.....	6
Classifiers.....	6
VII. Final submission model	6
Works Cited.....	7

I. Introduction

Scene classification is one of the most difficult problems in the machine learning community and is often used as a benchmark for algorithm performance. One of the big challenges associated with scene prediction is that these images are classified semantically and analyzing individual pixels will often yield poor results. One approach to address the problem involves generating low level features from these images and perform classification on these features. Two popular features that are often used in scene classifications are GIST and SIFT features (Li Zhou). The former detects gradients (magnitude and orientations) in different parts of an image while the latter extracts blob features at different scales. While these features have significant improvements over pixel representation, they only perform well on object detection and classifications with a small number of classes. In addition, we also considered Google's Inception v3 CNN model, which was trained on over 1,000,000 images and 1,000 classes from ImageNet. The number of training examples and variety of classes make CNNs trained on ImageNet particularly well suited for the assignment 3 task as the detectors trained are very likely to produce outputs that will be helpful in classifying the images to their respective class. In the project, we explored two types of features: 1) SIFT features and 2) features generated from part of the pre-trained inception-v3 model. The classification models we focused on were KNN, SVM and a simple neural network that produces the softmax probability of each class. Different ensemble classifiers were also explored to see if results could be further improved. Since labels were only given for the training images, most analyses were done by separating the training set into an actual training set (90%) and a validation set (10%). Both sets had similar proportions for each class.

II. Training Data Analysis



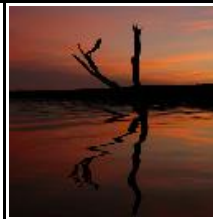


Ambiguity and errors

The training set contains images that are difficult for even a human to classify (Table 1). There were also other images that could conceivably belong to more than one class (Table 2). A few labels were also mislabelled (e.g. image 74), and all these factors could lead to prediction errors.

Table 1. Ambiguous images from training set

				
32 2 - Indoor	49 5 - Plant Life	77 2 - Indoor	84 4 - Animal	232 5 - Plant Life

Table 2. Training images that could fall in multiple categories

				
3 3 - People 8 - Sea	4 1 - Structure 8 - Sea	44 5 - Plant Life 8 - Sea	74 1 - Structure 5 - Plant Life 8 - Sea	113 2 - Indoor 3 - People

Class Imbalance

The number of examples in each class is greatly imbalanced, leading to worries about performance for the under-represented classes. The classes are distributed as follows:

Table 3. Histogram of class labels in the training set

Class	1	2	3	4	5	6	7	8
Size	2114	1951	700	406	1671	91	19	48

III. SIFT features

The original SIFT feature detection uses difference of gaussian to locate keypoints where the features are the most prominent compared to its neighboring pixels. The features at the keypoints are subsequently described using Histogram of Oriented Gradients (HOG). However, many papers have suggested an alternative method where image are divided into small overlapping grids and HOG is applied on each grid. This is commonly known as dense SIFT feature detection and is often used in scene classification. In our project, the dense SIFT technique was adopted and 256 grids of size 16x16 pixels were used to generate SIFT features on each image. All images were converted into gray images before feature detection. Since each image contained a large number of features, principal component analysis was applied to reduce the dimension of the input features. KNN was used to compare with the baseline method and SVM was also used to see if a different classifier would yield a better result. For both methods, the number of PCA components was varied initially with a default set of parameters to determine the optimal number of components (figure not shown). Each classifier was subsequently tested with different parameters as shown in Table 1.

Table 4. KNN and SVM parameters

Method	Parameters	Values	# of PCA components
KNN	# of nearest neighbors	3, 7, 11,...,49, 53, 57	10
SVM	Kernel type	'linear','rbf'	50
	C (penalty term)	1, 3, 5, 7, 10, 20	
	gamma	1e-1, 1e-2, 1e-3, 1e-4	

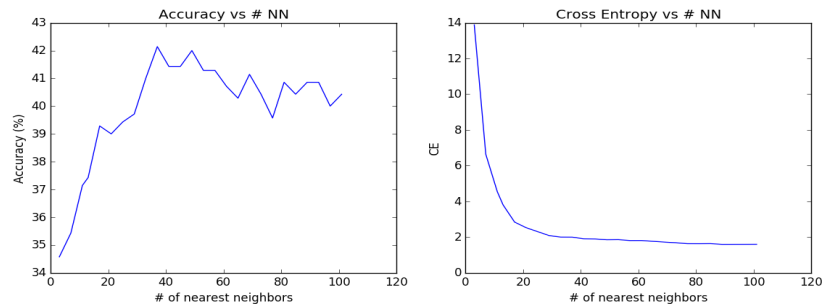


Figure 1. Accuracy and cross entropy as a function of number of nearest neighbors

KNN achieved its highest accuracy at 42% when the number of nearest neighbors (NN) is 35. The accuracy was lower when the number of NN is small, which seemed to suggest that the SIFT features were not very good at separating classes and more neighbors were needed for the correct class label to achieve majority. However, we noticed that most of the predictions were either class 1, 2 or 5, which corresponded to the three largest classes in the training set. This was not surprising since other classes had substantially fewer samples and it would be very unlikely that the majority of 35 neighbors belong to one of the minority classes no matter where the input features are located.

Therefore, the real reason why accuracy increased with higher NN was that the model could only predict one of the three classes, and the chance of guessing it correctly was higher since these classes made up the majority of the validation set. As the number of NN increases past 35, accuracy decreased gradually even though but cross entropy was still decreasing. However, it would be unreasonable to consider NN of over 100 since some samples would no longer be neighbors.

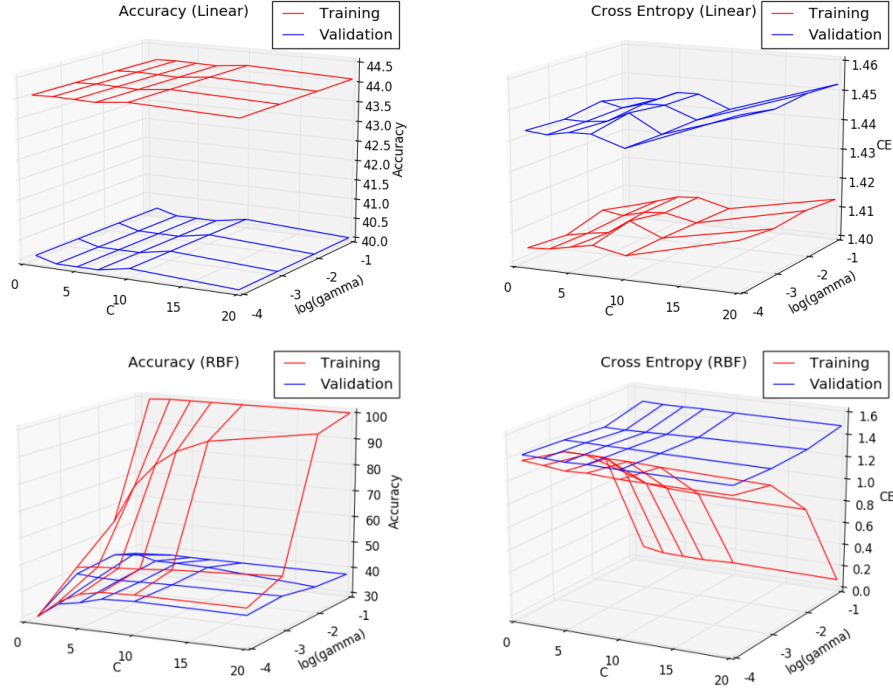


Figure 2. Accuracy and cross entropy across different parameters

In the linear SVM model, the maximum validation accuracy achieved was 40% (Figure 3). Training accuracy was slightly higher but still below 45%, which suggested that the decision boundaries were not linearly separable. The hyperparameters had a subtle effect on cross entropy but very little impact on accuracy. In the case where the radial basis function (RBF) kernel was used, the maximum validation accuracy was approximately 45%. Greater gamma value resulted in higher training accuracy but had little effect on validation accuracy, indicating that the model was overfitting the training data. The “C” parameter was the penalty term that effectively regularized the weights of the SVM model to minimize training errors. Accuracy was low when C was small since the penalty was not significant enough to affect the overall loss function. However, both training and validation accuracy plateaued as C approached 20. Note that contrary to the linear model, training accuracy in the RBF model did converge towards 100% due to the flexibility of the decision boundary.

IV. Inception-v3 CNN model

Transfer learning

In order to benefit from the performance of the inception-v3 CNN model from Google’s TensorFlow (TensorFlow), a transfer learning technique was used where the final layer of the pre-trained CNN model was removed (leaving the trained detectors) and replaced by a new classifier. After running each image through the CNN, the features, known as “bottleneck” features, were saved for future use. In order to determine how separable the features were after this step, the high dimensional features were processed through T-SNE in order to view them on a two dimensional plane. The results are shown below:

From the T-SNE graph, it can be observed that classes 1 to 6 are generally separated while classes 7 and 8 are mostly embedded in other clusters. Some interesting interactions are also observed, such as many in class 3 mixing with classes 1 and 2. This is rather not surprising, as images of people will often be in environments that could be classified as indoors or near structures. There are many class 1 examples interspersed within class 2, which again is not surprising as indoor scenes have many similarities with structures. After some initial tests, three classifiers were chosen as good candidates: SoftMax, KNN, and SVM (with PCA).

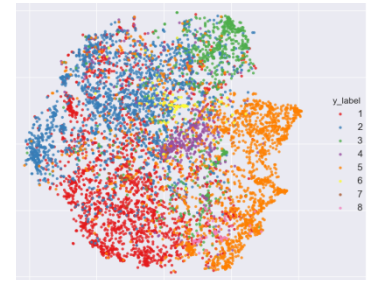


Figure 3. T-SNE graph for “bottleneck” features

CNN + SoftMax

TensorFlow’s default softmax retraining example was found to be very robust, and the existing code was utilised with minimal changes. Certain parameters were kept constant in order to allow for a fair comparison of the hyperparameters. The holdout validation size (only tested after training has ended) was kept at 500 examples which are selected proportionally from each class. The results of the hyperparameter selection is shown below (hyperparameter under investigation for each row in **bold**):

Table 5. Hyperparameters tested for CNN + softmax

Training Batch Size	Number of Steps	Learning Rate	Final Accuracy	Accuracy Graph	Cross Entropy Graph
Baseline					
100	4000	0.01	78%		
Analysing Training Batch Size					
50	4000	0.01	73%		
200	4000	0.01	73%		
Analysing Learning Rate					
100	400	0.1	72%		
100	4000	0.001	82%		
100	8000	0.0005	79%		

As shown in Table 4, various hyper-parameters were analysed to arrive at an optimal model. The most significant concern was the degree of overfitting to the training data observed in most cases. This resulted in decreasing testing accuracy. The final solution utilised slower learning over more iterations to ensure that overfitting was minimized. The CNN model with SoftMax was the first submission to Kaggle and obtained a score of 79%. The python code was inspired by the work done by (Thompson). This was an encouraging first attempt! Having just learned about Ensemble methods from lecture, the team formulated the idea of utilizing the bottleneck features to train other types of classifiers and then using a simple bagging method to output predictions based on more than one classifier. Many classifiers were tested against a constant validation holdout (10% of the training data for each class) set to identify the best performers to use in the final Ensemble method.

CNN + KNN

KNN was the first and simplest classifier to use with together with the bottleneck features, and it performed reasonably well. The 'k' parameter (number of neighbours) was optimized to 9.

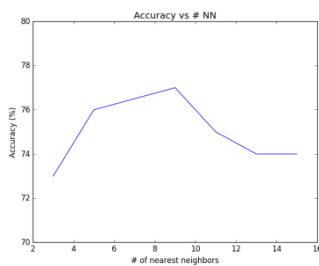
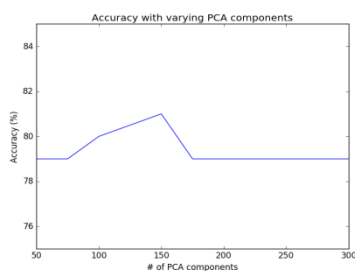


Figure 4. Nearest neighbor optimization

CNN + SVM (with PCA)

Another interesting classifier to experiment with is SVM. Due to the time required for training and to reduce unwanted noise from the data, the bottleneck features were first processed with PCA. Various hyperparameters were optimized as follows:



Kernel	Validation Accuracy
RBF	81% (slightly higher)
Linear	81%
Polynomial (degree 3)	77%

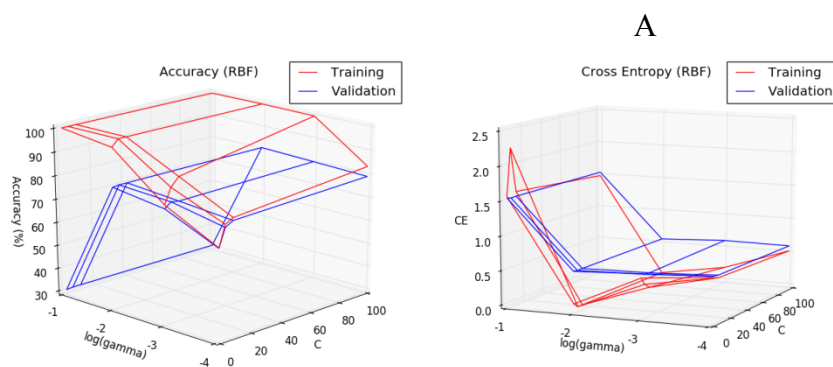


Figure 5. Hyperparameter optimization for SVM. Final parameters used were: PCA (150), Kernel (RBF), C(5), gamma (0.01)

V. Ensemble methods

To further improve the overall performance, several ensemble methods were used build on the methods that had already been explored.

- 1) Majority vote: Since the three methods used in section IV all achieve similar performance on the validation set, this was a simple approach that could be effective if all three methods make different errors.
- 2) Soft majority vote: This method was used as an improvement over hard majority vote since it also considered the confidence of each method on each label.
- 3) Bagging method: Bagging method was chosen as a way to mitigate the effect of outliers, which could be abundant given the diversity of the images in the training set. Bagging was performed on both the KNN and SVM model and the fraction of samples that were tested were 0.7, 0.8 and 0.9 of the total training samples.

Table 6. Validation accuracy

Method	Majority vote	Soft majority vote	Bagging (KNN)	Bagging (SVM)
Accuracy	80%	81%	78%	80%

VI. Comparison between methods

Features

From KNN and SVM, it was evident that dense SIFT features along with PCA were not able to separate features between different classes. One possible explanation could be the loss in spatial information when SIFT features were generated. Different objects may contain similar SIFT features but very distinct configurations, and SIFT features would not be able to distinguish between these objects. The same argument held true for GIST features, which explained why both features yielded similar results. On the other hand, inception-v3 model was able to detect low level features, capture the spatial relationship and transform into more general features at each stage of the model, which largely explained the substantial improvement over GIST and SIFT features. The result demonstrated the power of CNN and justified why it is the state-of-the-art model for complex image classification.

Classifiers

When SIFT features were used as the input, all classifiers yielded similar result since the input was poorly constructed. When the inception model was used, the SVM and softmax method performed slightly better than KNN. The result indicated that the inception model was able to distinguish different classes very well and the discrepancy of decision boundaries between from the three methods did not have a significant effect on overall performance. None of the ensemble methods improved the model, which suggested that any attempts to improve training accuracy resulted in overfitting and the models became less generalizable.

VII. Final submission model

The final submission model made use of transfer learning through the use of Google's Inception v3 CNN together with Softmax, KNN and SVM classifiers. The results of each classifier were combined through a hard-voting ensemble method to make use of each classifier's unique interpretation of the CNN features being used as input and improve on the performance of any individual approach. The top performance reached on the test set was a score of 0.81649.

Works Cited

- Li Zhou, Zongtan Zhou, Dewen Hu. *Scene classification using a multi-resolution bag-of-features model*. 14 June 2012. <file:///Users/jfl/Documents/CSC2515/Papers/2013%20(46)%20-%20Scene%20classification%20using%20multi-resolution%20bag-of-features%20model.pdf>.
- TensorFlow. *How to Retrain Inception's Final Layer for New Categories*. n.d. <https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/how_tos/image_retraining/index.md>.
- Thompson, Scott. *Using Transfer Learning to Classify Images with TensorFlow*. n.d. <<https://medium.com/@st553/using-transfer-learning-to-classify-images-with-tensorflow-b0f3142b9366#.auhk1zei2>>.
- Scikit-learn: *Machine Learning in Python*, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.