University of Southern California

EE660: Machine Learning from Signals: Foundations and Methods

CIFAR-10: Object Detection in Images

Submitted By:
Vinaykumar S. Hegde
Graduate Student
Department of Electrical Engineering.
University of Southern California, CA
USC ID: 4809250026
USC email: vinaykuh@usc.edu
Submission Date: Dec 8, 2015

Table of Contents

1 Project Homepage

CIFAR-10: Object Recognition in Images
Bigbucket Page:
`https://bitbucket.org/vinaykumarhs2020/ee660_project`

2 Abstract

CIFAR-10 is a tiny image dataset is provided by Krizhevsky and Hinton [5] of University of Toronto. This has 60,000 32x32 RBG images are divided into 50,000 train and 10,000 test images. This dataset is smaller compared to other object detection datasets, but equally challenging because of very small size of the images.

This project aims at applying machine learning techniques to classify images into 10 different categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. First major part of the project is feature extraction and next part is the model selection for machine learning algorithms. Features extractions like **H**istogram **o**f **G**radients (HOG),**B**ag **o**f **W**ords/**F**eatures (BOW/BOF) of **S**cale **I**nvariant **F**eature **T**ransform(SIFT) are tried with **S**upport **V**ector **M**achines (SVM) and **R**andom **F**orest techniques. In the last part of the project, **C**onvolutional **N**eural **N**etwork (CNN) is explored and seems to give very good result with this dataset.

Out of all the classifiers, CNN performs the best with 78% accuracy on testing data (and 82% accuracy on training data). Next best result is SVM classifier with HOG features. This accuracy is almost half of CNN accuracy. 42% on training set. SVM results are from obtained from cross validation and grid search over wide range of parameters.

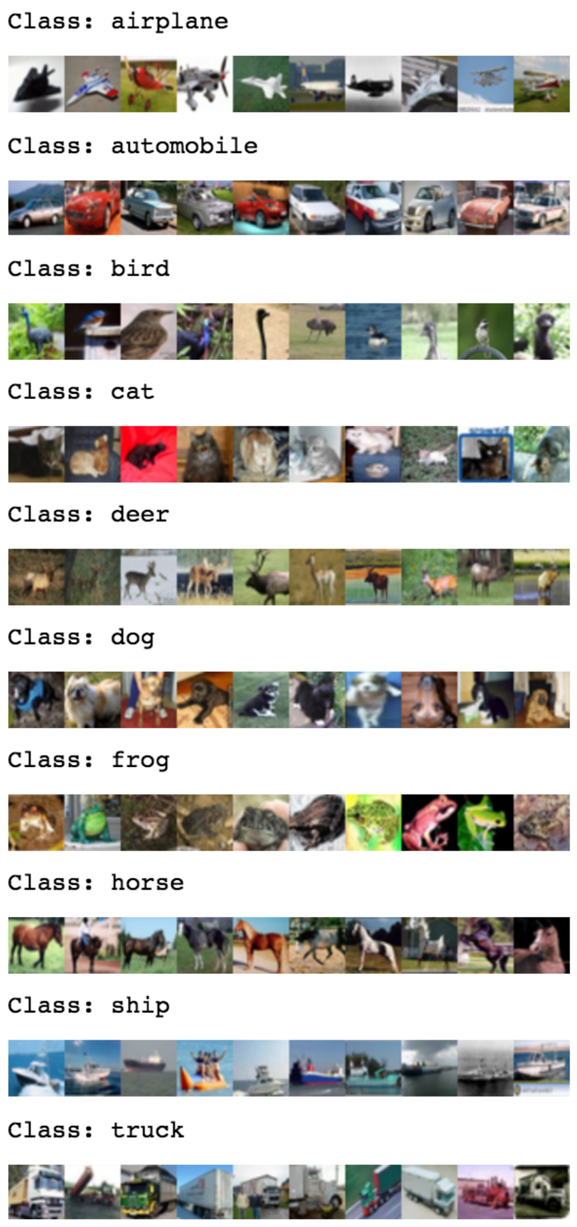


Figure 1: Example images in CIFAR-10 Dataset

3 Problem Statement and Goals

This project aims at evaluating various machine learning techniques for object detection in images, specifically in tiny images. Inherently image classification is non trivial because of various reasons. Pixel values themselves are not good set of features. Alex [5] shows that pixel values are higly correlated and need transformations or whitening operations to remove the correlation. Even then, number of pixels in image might exceed several thousands and burst the number of features in training.

These is an engineering problem and can be solved by better feature extraction techniques. Some of the common image features extractions depend on corner values, gradients and other variations in the image. Extracted features can be provided to classifiers, like SVM or Random Forest to classify the images.

Aim of this project is to compare the performance of these algorithms on features extracted from images and try to explore other methods to improve the performance.

4 Literature Review

Feature extraction form image is the crucial step of this project. Several image feature extraction techniques are explored and tested on SVM and Random Forest Classifiers.

4.1 Bag of Features (BOF)

This is derived from text classification literature (Bag of Words) and based on histogram of keywords in the document. Csurka et al. [3] showed the efficient method of creating bag of image keypoints and using the generated histogram as features for visual categorization. This method can be summarized as:

1. Find the image keypoints (SIFT, SURF or ORB)
2. Get the image descriptors for the detected keypoints
3. Cluster the keypoints into k clusters - representing k words.
4. Create a histogram of features and normalize it.

This histogram is used as features for image classification.

Image Keypoints and Descriptors

Popular keypoint descriptors are Scale Invariant Feature Descriptors (SIFT) [8], Speeded-Up Robust Features [1] and, Oriented FAST and Rotated BRIEF (ORB) [10]. All of these methods are explored and SIFT is the only method that is found to work faithfully. SURF and ORB need 32x32 window around the keypoints to define the descriptors. Since the CIFAR-10 dataset is tiny (32x32), extracting SURF and ORB descriptors seems to be difficult, unless we make some algorithmic changes.

Several parameters settigns in SIFT are explored and Number Octaves = 4, Contrast Threshold = 0.01, Edge Threshold = 20, Sigma=1.2 are found to work well. Default parameters proposed by Lowe [8] has Sigma = 1.6 and this configuration doesn't seem to work on our dataset. With default settings, 96 train images and 24 test images were without even single features. With my setting, only 2 train images had no features and all test images had at least one features.

4.2 Histogram of Gradients(HOG)

Histogram of Gradients (HOG) was proposed by Dalal and Triggs [4] in 2005 as an alternative to SIFT in pedestrian detection algorithm. This features can also be used for other application because of their versatility and dense nature. Unlike SIFT, these are dense features and whole image is considered while extracting features.

Image is divided down into several cells of fixed size and blocks are defined as some combination of these cells. Blocks stride may have overlap. Histogram of gradients are calculated for each of these blocks and quantized into 8 or 9 bin values corresponding to angle ranges. For this project, Scikit-Image [12] was used to extract HOG features and configurations used are: 9 orientations, 8x8 pixels per cell, 3x3 cells per block.

4.3 Convolutional Neural Network(CNN)

While the image feature extractions depend on handcrafted/user defied parameters, modern object detection techniques use features learnt from dataset using several iterations and convolution with trained filters. This technique was introduced by LeCun et al. [7] in 1989 and gained momentum after PASCAL and Imagenet challenges. AlexNet [6] and VGGNet [11] are two popular architectures which have won Imagenet competitions previously and used widely in object detection applications. AlexNet is dense, shorter and uses larger convolutional filters. VGGNet, on the other hand, uses smaller convolutions and much deeper. Both are known to produce very very good results compared to any conventional machine learning algorithms. It is also common to use trained VGGNet or AlexNet as feature extractors and tap the intermediate layer outputs to get the features for images. Forward propagation is very quick and can be a very efficient feature extractor. Figure 2 shows the architecture of AlexNet.

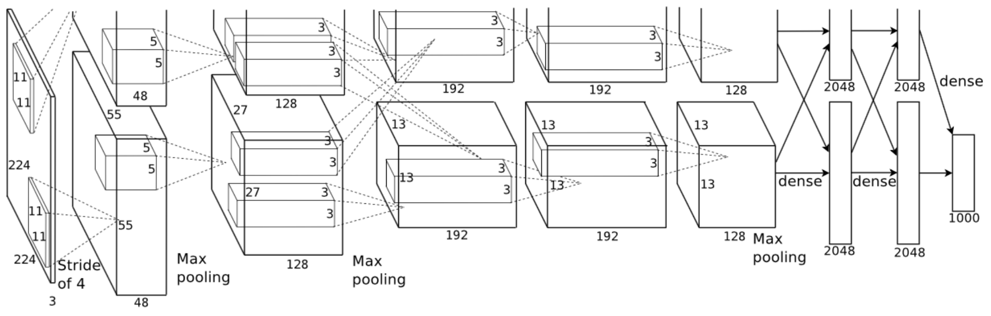


Figure 2: Architecture of AlexNet

5 Prior and Related Work

This project is done independently and does not coincide with any current or prior works.

6 Project Formulation and Setup

Algorithm details, parameters to tune, flowchart and formulas

6.1 Support Vector Machines (SVM)

SVM Classifiers are very well known in machine learning and found to yield very good results. Using appropriate kernels and parameters, we can achieve very good results. Idea of using this classifier for image object detection is very old and was also published by Dalal and Triggs [4] in 2005. This motivated me to try it on CIFAR-10 dataset and test the performance. Figure 3 shows the flowchart for my implementation.

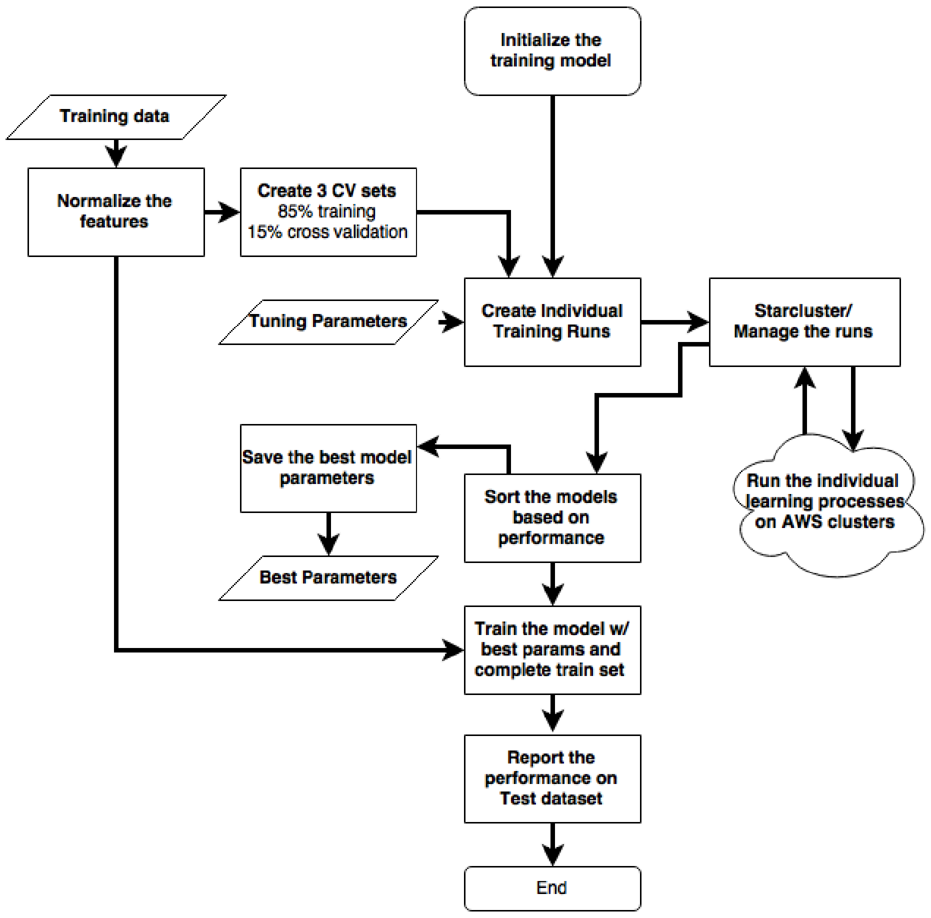

Initialize the training model
Training data
Normalize the features
Create 3 CV sets 85% training 15% cross validation
Tuning Parameters
Create Individual Training Runs
Starcluster/ Manage the runs
Save the best model parameters
Sort the models based on performance
Run the individual learning processes on AWS clusters
Best Parameters
Train the model w/ best params and complete train set
Report the performance on Test dataset
End

Figure 3: Flowchat - Model Selection Algorithm

Mathematic Formulation

Let y_i represent the class of image i represented by feature vector $\mathbf{x}_i$. Since we have 10 classes, we can use any multi-class approaches like one-vs-all method to classify them. This breakdown the problem to binary case and now let $y_i \in \{-1, 1\}$ represent if the example belongs to class i ($y_i = 1$) or not ($y_i = -1$). Let $\mathbf{w}$ be the weights and b be the constant term. Main objective in SVM is to minimize number of classification errors by finding a decision boundary that minimizes: $||\mathbf{w}||$ subject to $y_i(\mathbf{w}.\mathbf{x}_i - b) \geq 1$. If the data is not linearly separable, we can use some other kernels to classify the data.

Parameters to be tuned

In this project, I am using Scikit-Learn's [9] SVC implementation (which is a python wrapper over libSVM), to classify the data. Following parameters can be tuned to achieve better results:

- **C**: Penalty parameter of error term.
- **kernel**: Type of kernel to be used (linear, rbf)
- **gamma**: Kernel coefficient in rbf kernel

6.2 Random Forests and AdaBoost

Random Forests are well know ensemble learning algorithms where multiple decision trees are constructed during training and test data is classified based on mode of decisions given by individual decision trees. This uses the concept to bagging to select random features during training. AdaBoost, on the other hand, is a meta algorithm, which can be used in conjunction with any decision trees algorithms to classify the data. AdaBoost with decision trees is know best out-of-the-box classifier with very few parameters to be tuned.

This project aims to combine AdaBoost with Random Forest for image classification. Simplicity and quick results of these classifiers motivated me to use in the project. Figure 3 shows the flowchart of implementation.

Parameters to be tuned

Scikit-Learn provides python implementations of both, Random Forests and AdaBoost. Following are the parameters to be tuned:

- **max depth**: depth of decision trees
- **max features**: number of features to consider during split.
- **n estimators**: max number of estimators at which boosting is terminated.
- **learning rate**: shrinkage of contribution of individual classifier.

6.3 Convolutional Neural Networks

Figure 4 represents the cnn architecture used in this project. This is a very small network compared to any other popular networks like AlexNet, VGGNet or GoogLeNet. This architecture is borrowed from Keras[2]/Theano example implementation. Table 1 shows the layer parameters in this project and Figure 4 shows the layer connections.

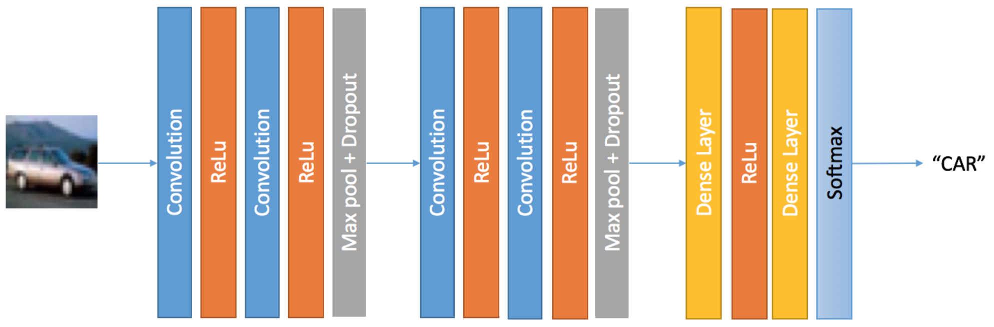


Figure 4: CNN Architecture

Layer	Description
Convolution	3x3 32 filters, 3 stack
ReLu	ReLu Activation
Convolution	3x3 32 filters 32 stack
ReLu	ReLu Activation
Max Pooling	2x2 Max Pooling
Dropout	p=0.25
Convolution	3x3 64 filters, 32 stack
ReLu	ReLu Activation
Convolution	3x3 64 filters 64 stack
ReLu	ReLu Activation
Max Pooling	2x2 Max Pooling
Dropout	p=0.25
Dense Layer	Linear Activation
ReLu	ReLu Activation
Dropout	p=0.5
Dense Layer	Linear Activation
SoftMax Layer	10 class output

Table 1: CNN Layer Description

Convolution Layer

This is a key layer in CNN which performs the convolution on the previous layer outputs with convolutional filters. Convolution filter parameters are learnt during the training process using back propagation. This, unlike hand picked parameters in SIFT/HOG feature extraction techniques, is learnt using training data and determines the best possible features. Each level of convolutional layers extract different types of features. For example, first convolution layer extracts low level features like edges, lines, color gradients.

ReLu Layer

ReLu stands for Rectified Linear Units ($f(x) = max(0, x)$). Earlier neural networks used Sigmoid ($f(x) = (1 + e^{-x})^{-1}$) or hyperbolic tangent functions. These functions have saturating properties. ReLu has non-saturating properties and found to converge faster.

Max Pool + Dropout

Pooling layers reduces the variance and perform sub-sampling operation in the neural network. These layers can pick the max, min, average or l2 norm of the given window of inputs. Dropout layer is reduced the overfitting in fully connected layers.

Softmax layer

This is similar to sigmoid function but with multi-class output in mind. This can be defined as: (x and w are vectors)

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

7 Methodology

Following sections explain the methodologies used in different classification algorithms.

AdaBoost - Random Forest and SVM

Figure 3 represents the approach in learning algorithm and model selection.

- Training data (HOG or BOF features) are normalized based mean and standard deviation of each features
- From normalized features for training data, 3 cross validation sets are created. Every time, X-train is randomly shuffled and 85% is named as new training portion and remaining 15% is named as validation data.

- These data are stored in numpy files on network mounted drives (to be easily accessed by clusters)
- IPython parallel and starcluster are used to submit the jobs to AWS cloud cluster and run the learning algorithms on the cloud in parallel.
- Once all the runs are completed, results are compared. Parameter with higher accuracy on validation data is considered best model. Overfitting is also handled by checking the difference between train and validation set accuracy numbers. Higher the difference, higher the overfitting.
- With the best parameters from previous step, complete original training data is used to train the model and test the performance on test data (which was used until this step!).

Convolutional Neural Networks

Convolutional Neural Networks are really compute intensive and we need to get the things right in the first run, otherwise the run time is wasted and if we run on AWS, we will be spending a lot of money as well. g2.xlarge AWS instance is the basic 1GPU 8VCPU instance in AWS and costs 70 cents/hour. One run for 200 epochs takes 26 hours and costs ˜$20.

Keras support CIFAR-10 datasets really well and has plugins to directly decode the dataset. Train and Test data are collected from keras numpy dataset and features are normalized to have zero mean and unit variance. These data are provided for training and Neural Network trains using back propagation and Stochastic Gradient Descent. This is carried out several times to fit the model better. Care should be taken to monitor the overfitting of the model.

8 Implementation

8.1 Feature Space

CIFAR-10 dataset has 60,000 images, each with 32x32 pixels and RGB channels. These images are split into 50,000 train and 10,000 test images. Each pixel is represented by an unsigned integer number (0-255). Therefore each image has 32x32x3 = 3072 features.

8.2 Preprocessing and Feature Extraction

3072 features directly obtained from image are not very useful, as they are mere rgb values and don't really contain the information about the object represented in the image. Very basic things, like a blue sky in most part of the image, might be helpful to classify airplanes (assuming that all airplanes pictures are taken while they are flying). In general, pixel values are highly correlated and don't account for scaling, rotation and translation of objects in the image. They don't even contain useful information about the shape of the object. This brings the necessity for alternate feature extraction techniques and look for robust features techniques to address these issues. SIFT and HOG are few standard feature extraction techniques in the computer vision literature and have their own advantages.

SIFT and Bag of Features

SIFT[8] features are invariant to scaling, rotation and translation. In the first step, keypoints are extracted in the gray scale image and in the latter step 128 dimensional descriptors are determined using 16x16 neighborhood around keypoint. So each SIFT keypoints have 128 dimensional and represent some key features in the image. The first step, keypoint extraction, is an important step which bring the invariant property to these features. Laplacian through Difference of Gaussian of image pyramids are used to detect keypoints. It is easy to match two images given their keypoints. Using these features directly is not a good approach, rather use bag of features approach suggested by Csurka et al. [3] and bag these features into 50 bags to get a 50 dimensional feature vector for each image.

Histogram of Gradients

Histogram of Gradients mentioned above in section 4.2 explains the theory behind HOG feature extraction technique. Using 9 orientations, 8x8 pixels per cell, 3x3 cells per block; we get 324 dimensional feature vector for each images.

8.3 Training, Validation and Model Selection

Using features mentioned in section 8.1, SVM and RF+AdaBoost models are trained. Test set is isolated from the training process. Out of 50,000 training samples, three set of training and cross validation samples are created. To do this, samples are randomly shuffled and separated into 85% training and 15% cross validation set. Three such sets are independently trained on all the parameters of training algorithms. Best performing parameters are picked based on performance on cross validation dataset and considered as suitable parameters for training process. These parameters are then used to train the algorithm on complete training dataset and evaluated on original testing dataset. Performance on testing dataset is evaluated and represented as confusion matrix and precision and recall scores for better clarity and judgment.

9 Final Results

9.1 Results with BOF features

This section illustrates the performance of different training algorithms on BOF features.

9.1.1 BOF with Random Forest

Figure 5 shows the variation of validation accuracy for every grid search parameters. First row represents the validation accuracy of number of estimators, second represents learning rate, third represents max depth, and the last one represents maximum number of features. From the plots, maximum number of features don't affect the accuracy a lot, whereas learning rate impacts the most. lower the learning rate, higher the accuracy.

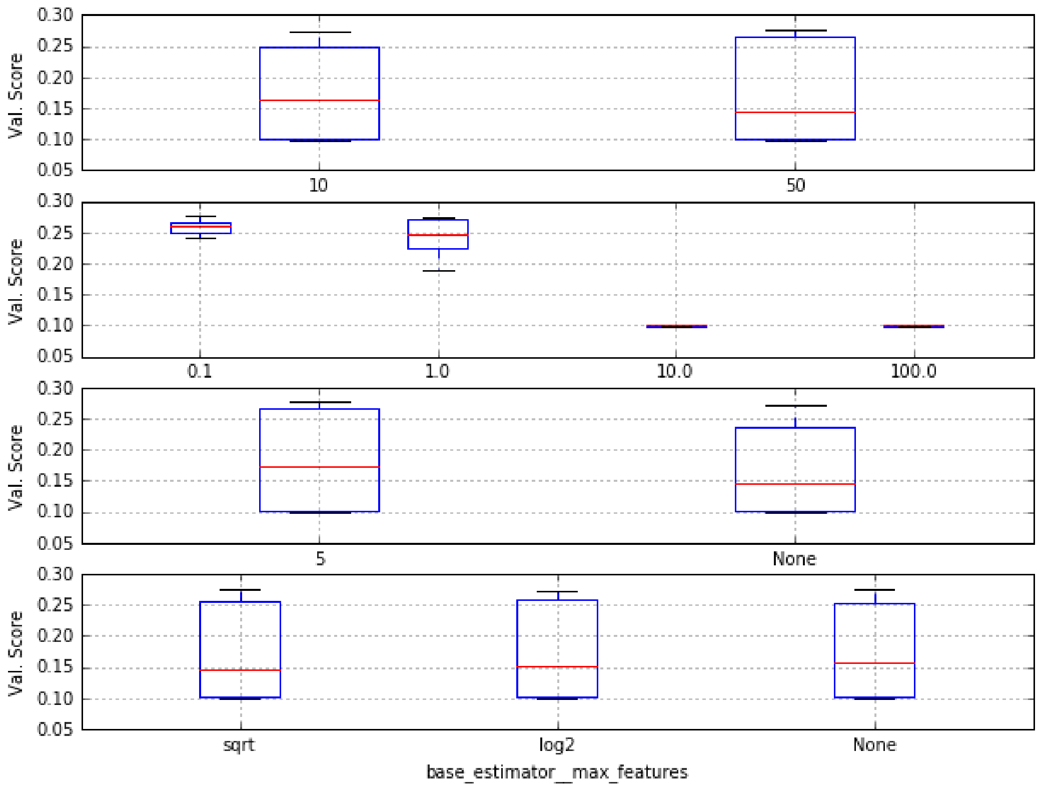


Figure 5: Grid Search Results for BOF with Random Forest

Figure 6 represents the confusion matrix of BOF features classified using random forest and adaboost. The performance is very very poor and diagonal entries are lighter than few of the off-diagonal entries. This mean a severe mis-classification.

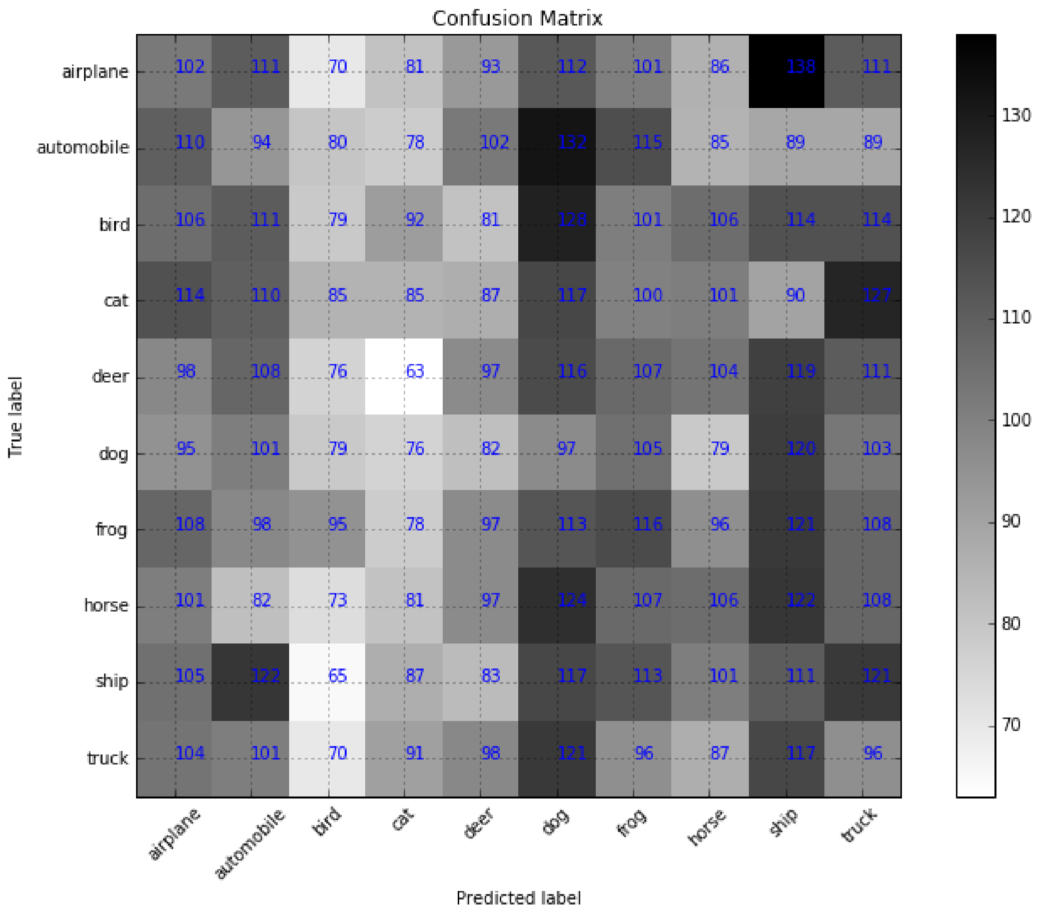


Figure 6: Confusion Matrix for BOF with Random Forest

Grid Search Results Following are the grid search results using BOF features with Random Forest and AdaBoost classifier. There is a clear sign of overfitting, train accuracy is much higher than validation accuracy. But, this was the best performing set out of all combinations and other iterations had poor accuracy as well as higher overfitting.

- validation set accuracy: 0.27173 (+/-0.00069)
- train set accuracy: 0.40913 (+/-0.00100)
- n estimators: 50
- learning rate: 1.0
- rf max depth: 5
- rf max features: sqrt

9.1.2 BOF with SVM

Figure 7 shows the variation of validation accuracy for every grid search parameters. First row represents the validation accuracy of kernels, second represents C and the last represents gamma. Interesting thing to note is that linear kernel overall performed better than rbf kernel. The best classifier noted below is one outlier in the batch.

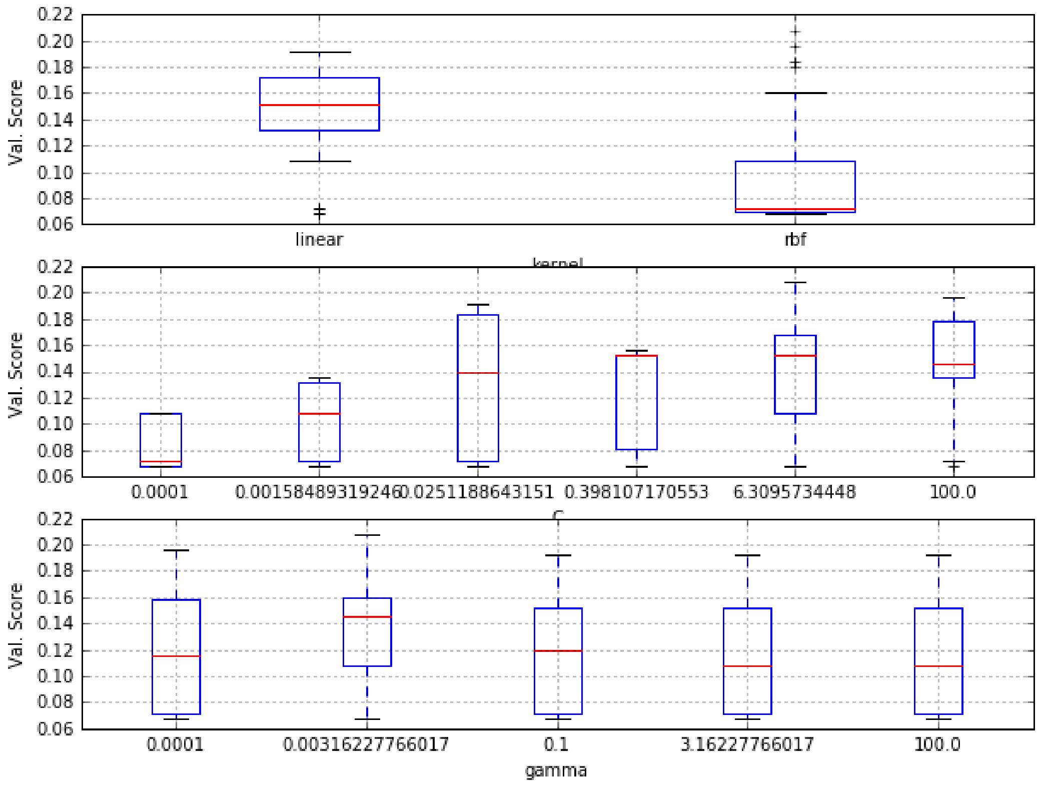


Figure 7: Grid Search Results for BOF with SVM

Grid Search Results

- validation set accuracy: 0.18800 (+/-0.00400)
- train set accuracy: 0.40311 (+/-0.00567)
- kernel: rbf
- C: 100.0
- gamma: 0.0001

9.2 Results with HOG features

This section illustrates the performance of different training algorithms on HOG features.

9.2.1 HOG with Random Forest

Figure 8 shows the variation of validation accuracy for every grid search parameters. First row represents the validation accuracy of number of estimators, second represents learning rate, third represents max depth, and the last one represents maximum number of features. From the plots, number of estimators don't affect the accuracy a lot, whereas learning rate impacts the most. Lower the learning rate, higher the accuracy.

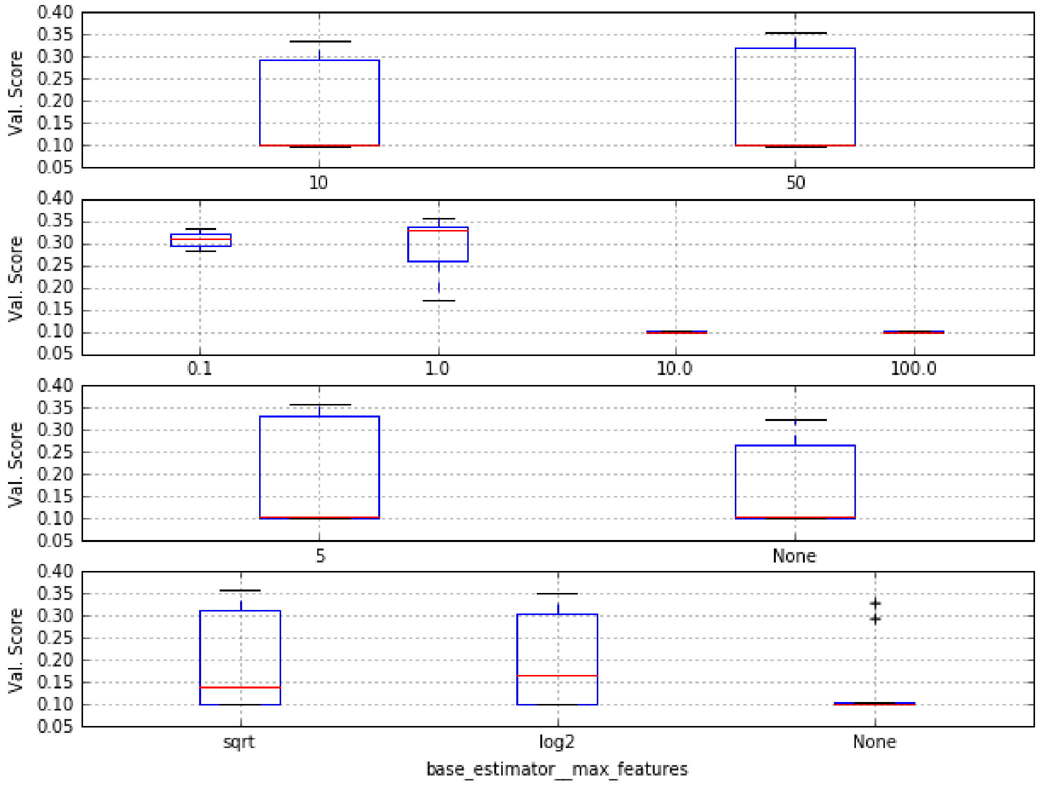


Figure 8: Grid Search Results for HOG with Random Forest

Figure 9 shows the confusion matrix for this classifier. Diagonal entries are darker than off-diagonal entries, which is a sign of good classification compared to earlier methods. But the test accuracy numbers mentioned below show that the performance is just around 35%.

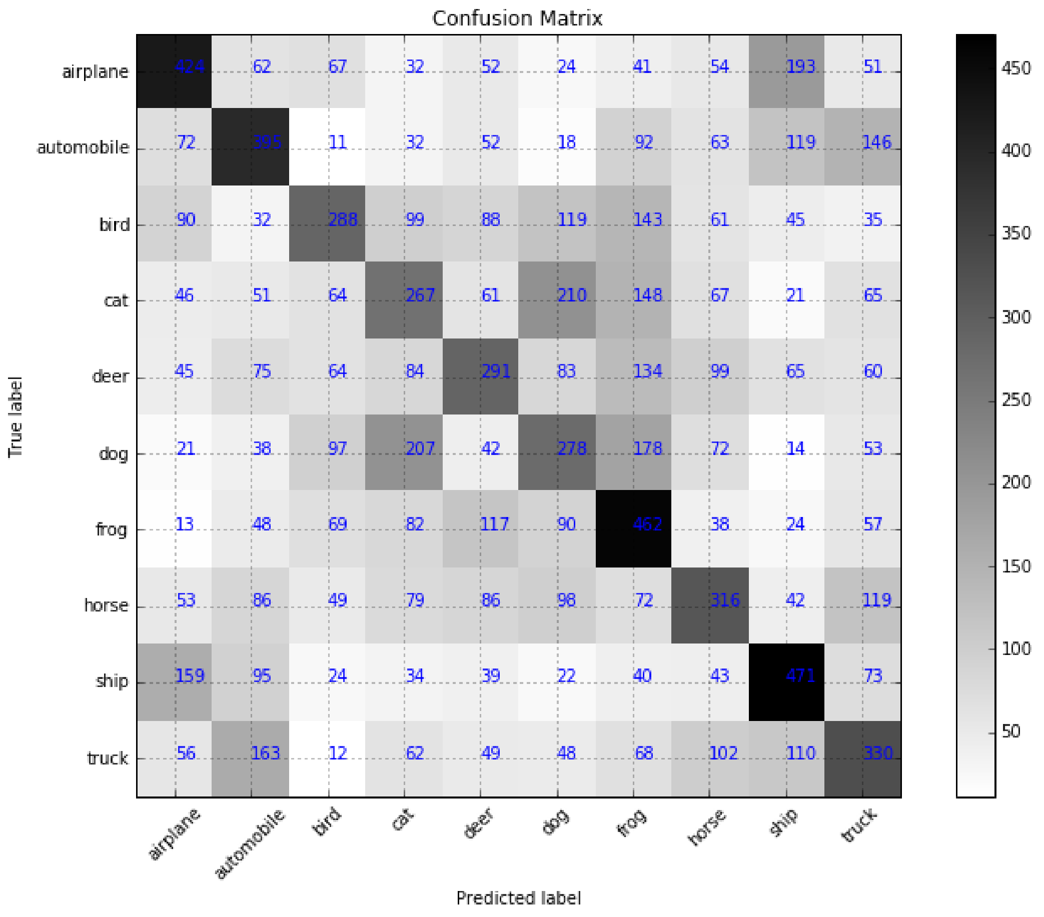


Figure 9: Confusion Matrix for HOG with Random Forest

Grid Search Results

- validation set accuracy: 0.34899 (+/-0.00395)
- train set accuracy: 0.49166 (+/-0.00054)
- n estimators: 50
- learning rate: 1.0
- rf max depth: 5
- rf max features: sqrt

Classification Statistics Following table shows the classification results from the best classifier (obtained form grid search)

	precision	recall	f1-score	support
airplane	0.43	0.42	0.43	1000
automobile	0.38	0.40	0.39	1000
bird	0.39	0.29	0.33	1000
cat	0.27	0.27	0.27	1000
deer	0.33	0.29	0.31	1000
dog	0.28	0.28	0.28	1000
frog	0.34	0.46	0.39	1000
horse	0.35	0.32	0.33	1000
ship	0.43	0.47	0.45	1000
truck	0.33	0.33	0.33	1000
avg / total	0.35	0.35	0.35	10000

Table 2: Classification result of Random Forest + AdaBoost with HOG features

9.2.2 HOG with SVM

Figure 10 shows the variation of validation accuracy for every grid search parameters. First row represents the validation accuracy of kernels, second represents C and the last represents gamma. Interesting thing to note is that linear kernel overall performed better than rbf kernel. The best classifier noted below is one outlier in the batch. One other interesting behavior to note is that gamma 0.0031 has a best performance and it drops on either sides of this value.

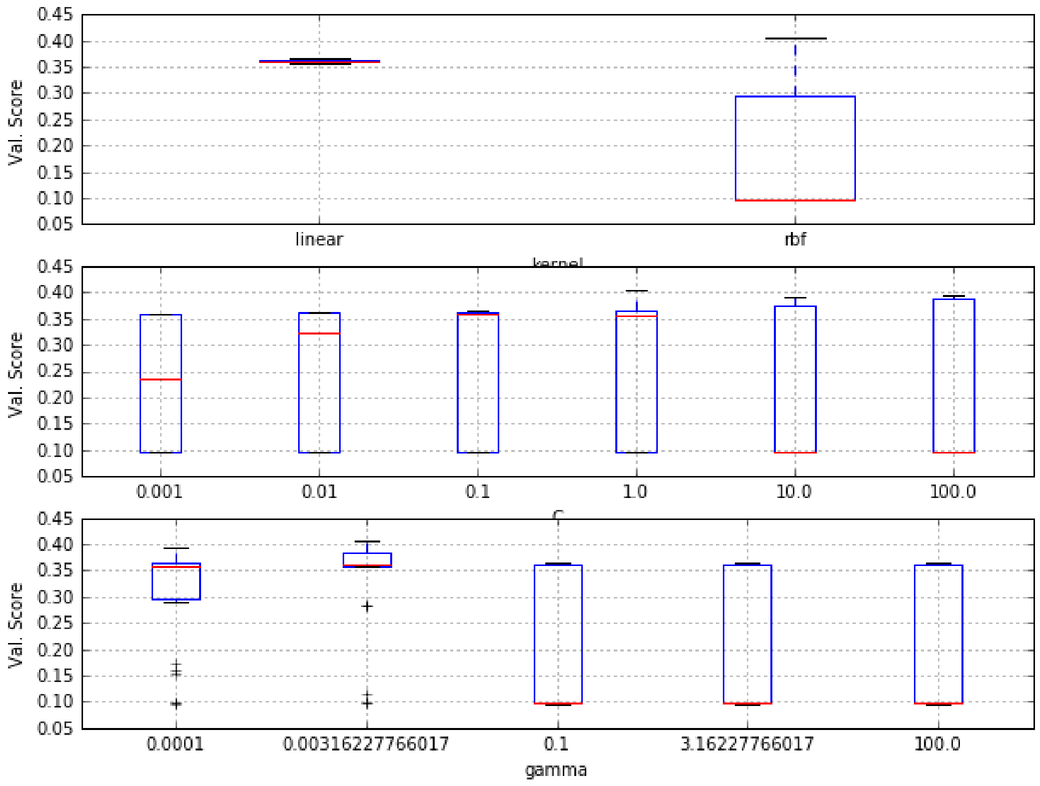


Figure 10: Grid Search Results for HOG with SVM

Figure 11 shows the confusion matrix for this classifier. The darker diagonal entries compared to off-diagonal entries are a good sign of performance and following section on classification statistics shows the 42% accuracy on test set.

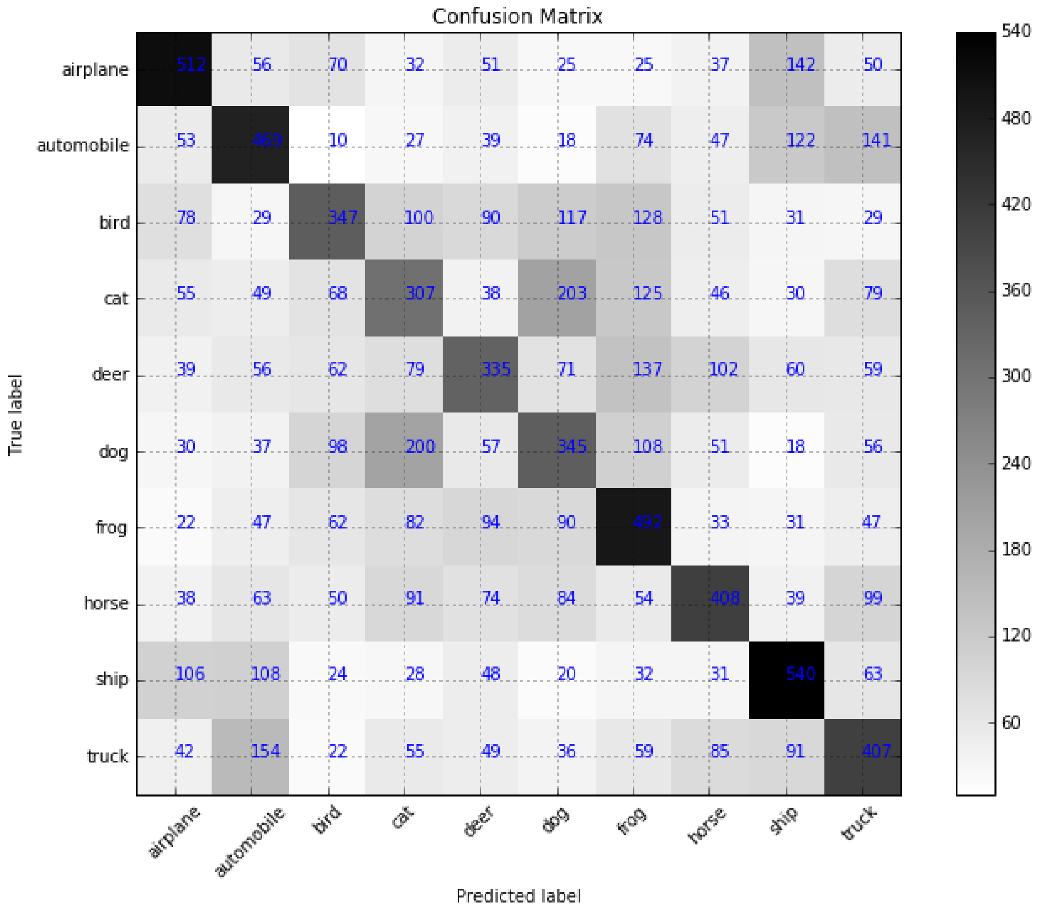


Figure 11: Confusion Matrix for HOG with SVM

Grid Search Results

- validation set accuracy: 0.40571 (+/-0.00014)
- train set accuracy: 0.63372 (+/-0.00097)
- kernel: rbf
- C: 1.0
- gamma: 0.00316

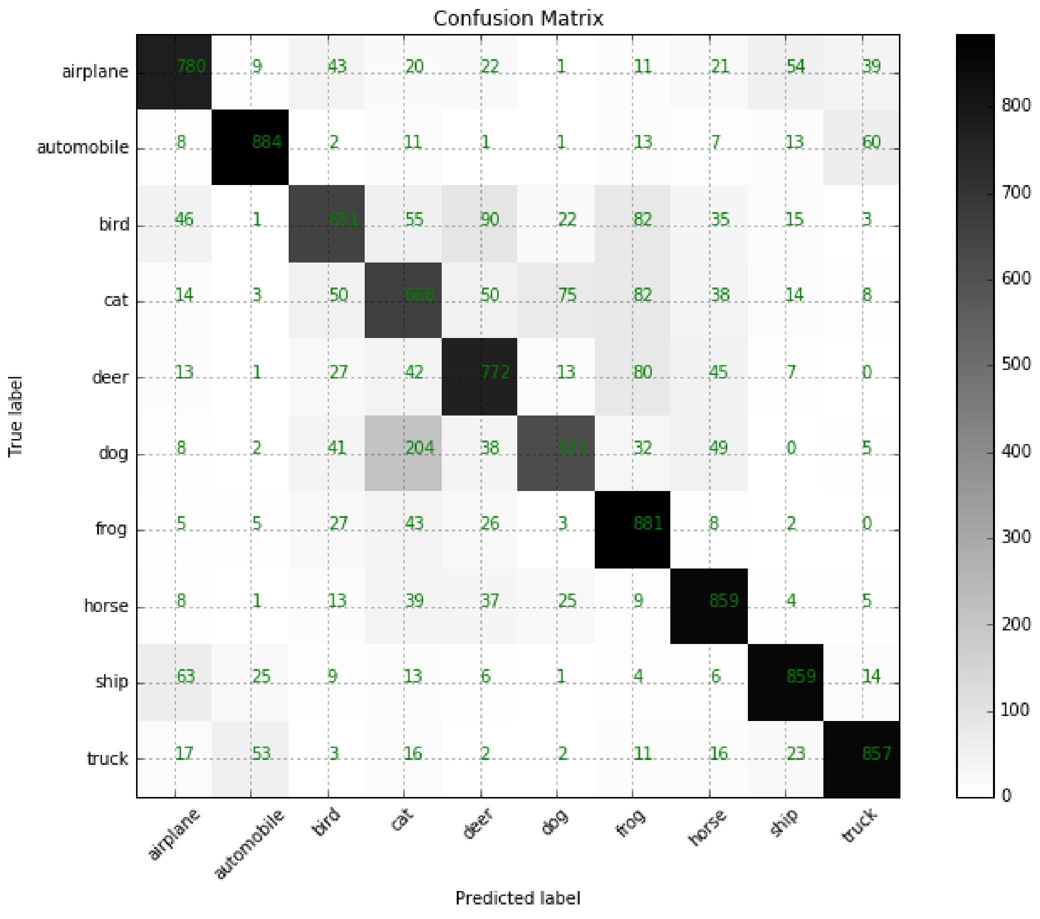
Classification Statistics Following table shows the classification results from the best classifier (obtained form grid search)

	precision	recall	f1-score	support
airplane	0.53	0.51	0.52	1000
automobile	0.44	0.47	0.45	1000
bird	0.43	0.35	0.38	1000
cat	0.31	0.31	0.31	1000
deer	0.38	0.34	0.36	1000
dog	0.34	0.34	0.34	1000
frog	0.40	0.49	0.44	1000
horse	0.46	0.41	0.43	1000
ship	0.49	0.54	0.51	1000
truck	0.40	0.41	0.40	1000
avg / total	0.42	0.42	0.41	10000

Table 3: Classification result of SVM with HOG features

9.3 Results with CNN

Figure 12 shows the confusion matrix for CNN classifier. As we can note that diagonal entries are much darker compared to any classifiers before, this represents a good classification. Classification report mentioned below reflects the same.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	780	9	43	20	22	1	11	21	54	39
automobile	8	884	2	11	1	1	13	7	13	60
bird	46	1	651	55	90	22	82	35	15	3
cat	14	3	50	666	50	75	82	38	14	8
deer	13	1	27	42	772	13	80	45	7	0
dog	8	2	41	204	38	621	32	49	0	5
frog	5	5	27	43	26	3	881	8	2	0
horse	8	1	13	39	37	25	9	859	4	5
ship	63	25	9	13	6	1	4	6	859	14
truck	17	53	3	16	2	2	11	16	23	857

Figure 12: Confusion Matrix for CNN 50 epochs

Classification Statistics Following table shows the classification results from CNN in 50 epochs:

	precision	recall	f1-score	support
airplane	0.81	0.78	0.80	1000
automobile	0.90	0.88	0.89	1000
bird	0.75	0.65	0.70	1000
cat	0.60	0.67	0.63	1000
deer	0.74	0.77	0.76	1000
dog	0.81	0.62	0.70	1000
frog	0.73	0.88	0.80	1000
horse	0.79	0.86	0.82	1000
ship	0.87	0.86	0.86	1000
truck	0.86	0.86	0.86	1000
avg / total	0.79	0.78	0.78	10000

Table 4: Classification results of CNN

Kaggle Competition Result CIFAR-10 Kaggle competition has ended last year and no new entries are accepted in the leader board. But the best performance result from this project can be placed 52 out of 231 entries in the leader board.

10 Interpretation

Section 9 shows the results of all the experiments. Since the dataset is huge, it takes quiet some time to run on local machines. Typical grid searches on random forest were taking 13-15 hours on MacBook pro laptop with i7 processor, 8GB RAM and SSD storage. SVM runs were taking very long time and crashed in between due to machine issues. Later, all the runs were run on 40-70 clusters on Amazon Web Services (AWS).

As Figure 6 shows, BOF features are very bad. This is mainly because of features used to create the bags. SURF and ORB features couldn't be extracted. SIFT features were extracted with lot of modifications to sigma and threshold parameters. I believe this is the reason for bad features and and hence worst performance on BOF features with AdaBoost+Random Forest as well as SVM classifiers.

HOG Dense features tend to perform much better than BOF features. As in table 2 and table 3, classifiers reach upto 42% accuracy on test dataset. AdaBoost+RF classifier gives 35% accuracy on test set and slightly below the SVM results. But training time for AdaBoost+RF classifier grid search was three times lesser than SVM grid search.

CNN results outperform all other results by factor of two. With 50 epochs, I gave got close to 78% accuracy on test set and clearly see a scope to improve this results with more processing power and GPUs. Just for a sake of curiosity, I saved the weights for all epochs and tested their performance on test sets. Test set error was 75% for 25 epochs, later improved to 76% at 35 epochs and finally to 78.3% for 50 epochs. Each epochs took 5800s for training and 940s for testing on CPUs, but took only 519s for training and 41s for testing on Nvidia K520 GPUs.

11 Summary and Conclusions

Performance and results clearly indicate that CNN is the best classifier for this case. It is possible to improve the performance even further by using real time data augmentations and deeper convolutional networks. Some of the top performers in Kaggle leader board have shared their network architectures and support my idea. Top performers on Kaggle use deeper VGG nets with ZCA whitened and normalized images to address features correlation in images and have achieved close to 95% accuracy.

Training deep neural networks might be time and compute intensive. So, some literature suggests to use trained models as feature extractors. The features might not be optimal, but assured to give better performance than existing feature extraction techniques. These features can be tapped from dense output or intermediate convolutional layers. Logistic regression or SVM on these features can give good results. I look forward to try these techniques in the winter break and compare them with my results.

On the other hand, this project is a great experience to handle big data and implementing machine learning algorithms efficiently. Many existing algorithms work well on a smaller dataset, but don't scale well with larger data. Thinking of how can it be implemented in parallel is crucial and great learning experience. Handling cloud based computing resources and open source tools are equally great experiences. Over the course of the project, I have fixed couple of bugs in open source tools and contributed to their betterment.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] Francois Chollet. Keras: a minimalist, highly modular neural networks library written on theano. https://github.com/fchollet/keras, 2015.

[3] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[5] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. ISSN 2167-8359. doi: 10.7717/peerj.453. URL `http://dx.doi.org/10.7717/peerj.453`.

Appendix

A Software Tools

Following softwares and toolsare used in this project:

- **Scikit-Learn**: Python based machine learning library. Has implementations of SVM, Adaboost, Random Forest etc,.
- **Scikit-Image**: Another python based library from Scikit-learn developers, useful to extract image features.
- **Keras**: Theano based deep learning library with python support. Has all basic layers, like convolutional layer, ReLu, Dropout etc,.
- **IPython and IPython Parallel**: Browser based interactive python shell. Supports cluster computing for parallel running jobs through ipcluster.
- **OpenCV**: Open Computer Vision library with Python support. Used to extract SIFT, SURF and ORB features.
- **Starcluster**: MIT open source tool to manage AWS clusters.
- **AWS**: Amaon Web Services. Cloud computing platform and useful to run compute intensive learning algorithms.
- **Bitbucket**: Code version management and repository.