Paintings To Artists

Achintya Gopal
Johns Hopkins University
agopal2@jhu.edu

William Watson
Johns Hopkins University
wwatso13@jhu.edu

Abstract

We looked into using machine learning for the purpose of predicting an artist given an image of a painting. In this paper, we analyze the performance of a multiclass SVM with multiple kernels trained on different feature vector encodings in comparison to training a convolutional neural network.

1 Introduction

We seek to create a model that can predict the artist of a given painting.

Since our data is a set of images, we needed to extract features for each painting and encode a vector for the purposes of training. For this, we choose three types of feature vector encodings to represent each image. The first encoding is a Color Histogram, which converts our images RGB color scheme into a n bit color system. The second encoding uses an approach known as the ”Bag of Words”, a vector encoding scheme that uses Oriented FAST and Rotated BRIEF (ORB) features as a ”vocabulary” to count the occurances of features. The third encoding is known as the Histogram of Oriented Gradients (HOG) which creates a vector of the distribution of edge directions and gradient orientations.

After feature extraction and encoding, we developed four algorithms to design an accurate model. Our first algorithm is a Dual One vs. All Support Vector Machine (Dual OvA SVM). Our second algorithm is a Dual Structured Support Vector Machine (Dual Structured SVM). We also developed a Convolutional Neural Network (CNN). We then designed a maching learning model that combined different machine learning models to increase the accuracy.

Since our algorithms use a Dual SVM, we will examine the effect of using various kernels. This will provide keen insight into the separability of our encodings. The two kernels examined included a Linear and a Radial Basis Function (**?**).

Current algorithms use Convolutional Neural Networks to train machine learning models to recognize objects in images (**?**). Convolutional Neural Networks have also been used to classify images with artists and recognize artistic styles. We tried to see how a Convolutional Neural Network would perform on a smaller dataset as most others use 100,000 - 1,000,000 images for training (**?**).

We will discuss the results of our methodology, along with an analysis of feature vector encodings, the accuracy of our algorithms, and any trends discovered throughout our analysis.

2 Data

The data set created for this project includes a collection of paintings from six artists: Dali, Monet, Turner, Rembrandt, Van Gogh, and Picasso. The training data includes 150 paintings for each artists; the testing set includes 50 paintings per artist. No painting is duplicated in the sets.

All images were resized so that the largest dimension is set to 500 pixels while preserving the aspect ratio of the original image. We did this in order to standardize the information being processed by our algorithms.

3 Feature Vector Encodings

In order to train our algorithms on a data set of images, we need to describe what a feature is and how can be encode our images as a vector of features. There have been many proposals as to what makes a good feature in an image, as well as the most efficent ways to describe these features into a set, known as a feature vector.

When detecting features, there are four important criteria that makes it a good feature. One is repeatability, where the same feature can be found in several images despite geometric and photometric tranformations. Another is saliency, where each feature has a distinctive description. Good features are compact and efficient, i.e. there should be fewer features than the number of pixels in an image. Finally, good features have locality, where a feature should occupy a relatively small area of the image, allowing the feature to be robust to clutter and occlusion.

Once we have detected features in an image, we must efficiently encode them into a set known as a feature vector. We will discuss three such algorithms that extract and encode a feature vector for each image efficiently.

3.1 Color Histogram

The purpose of the Color Histogram is to encode an artists choice in colors and the frequency of pixel values to encode an image. One recurrant theme that we visually discovered in our data sets was that each artists had a proclivity for the use of certain colors. For example, Rembrant used very dark colors throughout his paintings. Monet and Van Gogh used many variations of green and blue. In order to capitalize on this distinct difference, we decided to encode the frequency of there colors as a feature vector.

A Color Histogram is a representation of the distribution of colors in an image. For research purposes, we allowed there to be variation in the number of bits used to encode the RGB values. The size of the histograms are set to be 2^n, where n is the number of bits to encode. For our tests, we created Color Histograms that encoded for 9, 12, and 15 bits.

We use a simple voting-scheme for our histograms. For every image we create a histogram of size 2^n. For every pixel in the image, we encode it to create a specific ”bin” for that color. Then we increment the vote for that ”bin”. This will ensure that all similar colors share the same ”bin”.

3.2 Bag of Words (BoW)

The Bag of Words (BoW) model in computer vision seeks to treat image features as words, and then describe each image as a sparse vector of the occurance count of the words (**?**). The result is a sparse histogram over all the features in the ”bag.” There are several steps required to encode an image as a histogram.

The first step to building a BoW model is to extract from an image a set of keypoints and descriptors for those keypoints. For this, we used OpenCV’s implementation of the Oriented FAST and Rotated BRIEF (ORB) feature detection and description algorithm (**?**). ORB is a fusion of the FAST Keypoint Corner Detector and the BRIEF Descriptor to provide rotationally invariant descriptors for our keypoints.

However, each image has a variable number of descriptors, so the next step is to run a KMeans Clustering algorithm to find the centers and labels of the decriptors for all the images. This will be the ”dictionary” or ”visual vocabulary” for the encoding. The number of centers to create will be an optional parameter, and we will use 400, 800, and 1200 centers for testing purposes.

Then we represent our original images as a frequency of these ”visual words.” Hence our feature vector for an image is the histogram that describes the frequency of the ”visual words” for an image. The purpose for using a Bag of Words model is to find common features in an artists paintings and then create a set of histograms that included the frequencies of the features found from KMeans. This allows us to encode certain features that each artists used frequently in their works.

3.3 Histogram of Oriented Gradients (HOG)

The Histogram of Oriented Gradients approach is a feature descriptor representation that provides more reliable performance than using Scale Invariant Feature Tranformation (SIFT) feature descriptors (**?**).

The HOG Feature Extraction and Description algorithm is based on evaluating an image’s normalized local histograms of gradient orientations in a dense gird. The practical uses of HOG descriptors for our problem is that HOG capitalizes on the idea that an image can be characterized by its distribution of local intensity gradients or edge directions.

The image window is divided into small spatial regions that accumulates a local one dimentional histogram of gradient directions or edge orientations over all the pixels of the region. HOG improves the invariance to noise by contrast normalizing over overlapping spatial blocks. The normalized descriptors blocks form the HOG descriptors. For this project, we used OpenCV's implementation of the HOG algorithm.

The HOG representation of descriptors has several advantages that we want to exploit. The HOG algorithm captures edge or gradient structure of the image that is characteristic of local shape. It examines the "flow" of the image, allowing it to create a representation with a controlable degree of invariance to local geometric and photometric tranformations. By using HOG, we can encode the "flow" of a painting analytically, an use the resulting feature vector for training and testing purposes.

4 Support Vector Machines (SVM)

A Support Vector Machine (SVM) is a machine learning algorithm which finds the optimal hyperplane for separation of training data (**?**). The SVM algorithm amounts to the following optimization problem:

$$\min ||w||^2$$

subject to $y_i(w \cdot x_i + b) \geq 1$. This can be written as minimizing the expression

$$\sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b)) + \lambda ||w||^2$$

We implemented Pegasos to train the SVM (**?**). Since our implementation of SVM uses Stochastic Gradient Descent, at the start of each iteration (epoch), we shuffle the data.

The update rule for the Dual Pegasos was

$$\alpha_{t+1} = (1 - \frac{1}{t})\alpha_t$$

$$\alpha_{i,t+1} = \alpha_{i,t+1} + \eta_t \mathbb{1}[y_i \sum_{j=1}^{n} (\alpha_{jt} K(x_j, x_i)) < 1] y_i \alpha_{i,t+1}$$

where the first rule updates all the values of α and the second rule further updates the i-th α value in the t iteration.

We used a SVM because it has built in L2 regularization. We then converted the implementation of Pegasos into the Dual method so that kernels could be used.

The two kernels we used were the linear kernel,

$$K(x_i, x_j) = x_i \cdot x_j$$

and the radial basis function (RBF) kernel,

$$K(x_i, x_j) = e^{-\frac{|x_i - x_j|^2}{2\sigma^2}}$$

We used the linear kernel to compare the RBF kernel to the primal method. Since the linear method is a degenerative version of RBF, RBF will always do better in training than linear (**?**). However, RBF has a higher tendency to overfit. Since the RBF kernel has a parameter σ, which is the standard deviation of the Gaussian, we had to decide what would be the best standard deviation. For small σ, the RBF kernel has a higher change of overfitting since the kernel changes significantly for a slight variation in the input; for large σ, the RBF kernel could underfit since there is a very light change in the kernel when the input is varied slightly.

To use the SVM for multiclassification, we implemented a One vs All SVM (OVA SVM) and a Structured SVM. The One vs All SVM trains a Binary SVM for each class separate from each other and then predicts by finding which Binary SVM gives the highest score. However, the problem with this is the "confidence" scale might be different among the Binary SVMs and the distribution of positive and negative labels is highly skewed towards negative labels for each Binary SVM.

The structured SVM solves these problems since it only updates the SVM for the correct label and the SVM for the incorrect label per iteration, which solves the second problem of the OVA SVM. Since the updates are dependent on the scales of each SVM, the confidence scales are similar amongst the Binary SVMs, which solves the first problem.

5 Convolutional Neural Network (CNN)

For the Convolutional Neural Network, there are five types of layers: convolution layer, RELU layer, max pooling layer, dropout layer, and dense layer.

5.1 Convolutional Layer

The convolutional layer is a set of kernels (filters) used to convolve across the width and height of the input. The equation for a 2d convolution with a nxn kernel k on input x is

$$y_{rc} = \sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij} x_{(r+(n+1)/2-i)(c+(n+1)/2-j)}$$

The definition of a 3 dimensional convolution adds another summation over the third dimension of the kernel. Each result of a kernel being applied to input is stacked together to form the output.

For backpropagation through the convolutional layer, the derivative for kernel k of size nxnxc is

$$\frac{\partial L}{\partial x_{ijk}} = \sum_{l,p,r} \frac{\partial L}{\partial y^{l}_{(i+p-(n+1)/2)(j+r-(n+1)/2)}} k^{l}_{p,r,(c-k)}$$

where the sum is over the rows and columns of all the kernels. The derivative over the input w^{p}_{ijk} is

$$\frac{\partial L}{\partial w^{p}_{ijk}} = \sum_{a,b} \frac{\partial L}{\partial y^{p}_{a,b}} x_{(a-i+(n+1)/2)(b-j+(n+1)/2)(c-k)}$$

which is a sum over the input image.

The reason convolutional layers are useful is that the kernel is applied throughout the whole image which gives more local information.

5.2 RELU Layer

The RELU layer is an activation layer which applies $y = \max(0, x)$ to each value. The derivative used for backpropagation is $\frac{\partial y}{\partial x} = \mathbb{1}[x > 0]$.

5.3 Max Pool Layer

The max pooling layer is a way to downsample the input. It partitions the input into a set of non-overlapping windows and replaces each window with the maximum value in it. For backpropagation through the max pooling layer, the derivative is 0 through the values that are not the max, and 1 for the value that is the max.

The reason for the max pooling layer is to reduce the number of parameters needed to train, hence it reduces the amount of overfitting.

5.4 Dropout Layer

The dropout layer sets a fraction of the input units to 0. For backpropagation through the dropout layer, the derivative is 1 if the dropout did not affect the value or 0 if it set the input value to 0. Both the max pooling layer and the dropout layer are used to reduce overfitting.

5.5 Dense Layer

The dense layer is a fully connected layer usually added at the end of the network. Backpropagation through the dense layer is the same as propagated through a fully connected layer in a usual neural network. If there are n output nodes y_j (meaning n weight vectors w_j) and input x, the derivatives are

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial y_j} x$$

$$\frac{\partial L}{\partial x} = \sum_{j=1}^{n} \frac{\partial L}{\partial y_j} w_j$$

5.6 Implementation

We implemented the Convolutional Neural Network from scratch except for the dropout layer. Due to the massive sums for the backpropagation, we had to use a learning rate of 10^{-6}.

6 Accuracy Booster

After examining the images and the reasons for using different vector encoding, we predicted that the Color Histogram will do best on breaking the data into Rembrandt vs Monet/Turner/Van Gogh vs Dali/Picasso. The Histogram of Oriented Gradients will do better for a more global description that the Bag of Words model since Bag of Words only looks at the ORB features. The reason Color Histogram will do best in splitting the data into three separate classes is because the colors used by the artists in each ”class” are very similar.

For the accuracy booster, we created an algorithm that was trained on three classes (Rembrandt, Monet/Turner/Van Gogh, and Dali/Picasso). We then trained three structured SVMs with a RBF kernel using HOG based encoding for these three separate classes.

7 Analysis of Results

We first ran the One vs. All Dual SVM on the Bag Of Words encoding and the Color Histograms. However, due to the problems with using a OvA Dual SVM, we quickly switched to using a Stuctured SVM. We ran a Structured SVM on the Bag of Words enconding, Color Histograms, and HOG features with a linear kernel and the RBF kernel. Once we did this, we set up a simple convolutional neural network.

The OvA Dual SVM on the Bag of Words encoding converged to nearly perfect training accuracy in the beginning. This suggested to us that the algorithm was overtraining on the Bag of Words encoding.

The OvA Dual SVM on the Color Histogram after 20 iterations did not go past 80% training accuracy. This suggested to us that the Color Histograms were not linear separable and required a more complicated kernel to train the SVM. However, the Structured SVM on the Color Histograms also did similar in training accuracies, but achieved a 90% training accuracy for 15-bits. Even though the training accuracies were similar, the testing accuracy was better for the Structured SVM run on the Color Histograms than OvA, which suggests that the OvA Dual SVM is weaker.

The Structured SVM using a linear kernel when run with the Bag of Words encoding reaches about 100% training accuracy faster as the number of clusters increases (i.e. the more words there are to describe the image). The testing accuracy increases as the number of clusters increases, which makes sense since when you have more clusters, there are more ”words” describing the image, hence more information.

The Structured SVM using a linear kernel when run on the HOG encoding converges to nearly 100% training accuracy after 18-20 iterations. However, even as the training accuracy increased, the testing accuracy also increased instead of dropping off as a sign of overfitting. This would suggest that testing accuracy could be increased if we increased the training set.

The Structured SVM using a RBF kernel when run on the HOG encoding converges to over 95% after 16 iterations. As the training accuracy increased, our testing accuracy increased, but it converges to a consistent 50% testing accuracy after 8 iterations. The testing accuracy converged faster than the Linear Kernel Structured SVM, suggesting that the HOG encoding is more complex than just being linearly separable. If there was less testing accuracy, this would suggest that the SVM was overfitting, yet this is not the case.

The Structured SVM using a RBF kernel was also tested on the BOW encoding. The σ^2 value used for the RBF kernel on the BOW encoding was 750. This caused the training accuracy to converge to 100% and the testing accuracy increased as the number of iterations increased. Since the testing accuracy is still increasing, it is possible that if we increase the training size, the testing accuracy would also increase.

Another configuration was the Structured SVM using a RBF kernel when run on the Color Histogram encoding. upon inspection of the training accuracies, the training accuracy increases more quickly on the RBF kernel for the 9 bit Color Histogram than for 12 and 15 bit. Since the training accuracy did not converge as fast, it would suggest that the model is not overfitting, or at least overfitting occurs at a slower rate. This can be seen as the testing accuracy is better for the 12 and 15-bit Color Histogram instead of the 9 bit Color Histogram. Another reason they would do better than the 9-bit histograms is the fact that the higher number of bits results in a higher the range of colors to learn. However, the 12 and 15-bit testing accuracies are very similar, since it is impossible to paint one specific color smoothly across a surface.

The architecture of the Convolutional Neural Network we implemented was a convolutional layer with 6 kernels, a RELU layer, a max pooling layer which reduces the depth by a factor of two, a convolutional layer with 1 kernel, and then a dense layer with 6 outputs. We resized all the images to 128x128 so that the input was the same size. We used a learning rate of 10^{-6} and ran it for multiple iterations.

We knew our implementation of the Convolutional Neural Network was correct since the training accuracy increased on average over time. Even though the training accuracy increased, the testing accuracy never surpassed 25% for the first 10 iterations. The reason is that the data was most likely overfitting since there was not a dropout layer to further decrease overfitting and the data size was small. The CNN was run for 20 iterations, resulting in the training accuracy to jump between 20 and 50%. Our CNN converges to over 95% training accuracy after 32 iterations. However, this did not improve the testing accuracy at all.

Our accuracy booster used three Structured SVMs with RBF kernels. The first Strutured SVM classified if the image was Rembrandt vs Monet/Van Gogh/Turner vs Picasso/Dali based off of the Color Histograms. The other two Structured SVMs were used to distinguish from Monet vs Van Gogh vs Turner and Picasso vs Dali based off of the HOG feature vectors. After running this algorithm 12 times, the average testing accuracy was not much better than the testing accuracy for the other algorithms. Looking at which artists the algorithm got the most correct were Rembrandt at around 75%, Picasso and Dali around 45%, and the other three artists were around 70% for Monet and 0% for Van Gogh and Turner. The main issue was trying to classify Monet vs Van Gogh vs Turner since their styles and theme are very similar.

8 Comparision to Proposal

This project has evolved from our previous proposal. While our purpose remained the same, our approach was modified to reflect the challenges and difficulties we encountered.

We reduced the number of artists from 10 to 6. Our chosen artists were Dali, Monet, Turner, Rembrant, Van Gogh, and Picasso. We decided to have exactly 200 paintings per artist, split into 150 training and 50 testing images. Since our sets were large enough, we did not have to use cross validation.

In our proposal, we outlined three methods to train a model to predict the artist of a painting. However, we did not differentiate between the feature vector encoding process from the actual training. Our proposal mentioned a Color Histogram and a Bag of Words approach to encoding the data, yet we also implemented a HOG encoding scheme as well. We did not use the flattened feature vector approach since we did not train a perceptron-based neural network, instead we trained a convolutional neural network and so we did not have to flatten the image.

We then decided that a Dual SVM would yield more interesting results than in its primal form, allowing us to experiment with a linear and RBF kernel. This was not mentioned in the proposal. In addition, we did not specify whether our SVM would be a One vs. All or a Structured SVM.

Instead of constructing an ordinary Neural Network, we implemented a Convolutional Neural Network.

In terms of the milestones that we set forth in pur proposal, we achieved all stated goals and expanded upon the methods we researched. For the goals that set to achieve, we created the Color Histogram vector scheme and created two Dual SVM models, the OvA and the Structured SVM. In addition, we achieved two more vector encoding algorithms, the Bag of Words model and the Histogram of Oriented Gradients (HOG). We also implemented a Convolutional Neural Network. Therefore, we implemented all of the algorithms we would like to have achieved, and included other configurations to improve upon the research conducted.

9 Conclusion

9.1 Summary

In conclusion, we found that the feature vector encoding which gave the best results was the Histogram of Oriented Gradients with an RBF kernel.

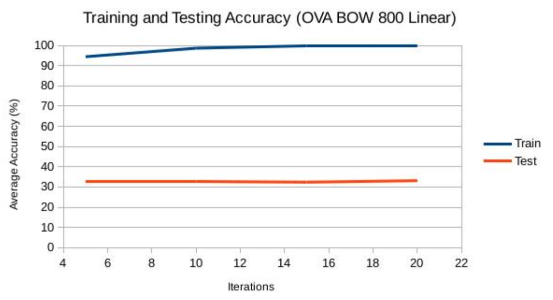
The RBF kernel was better than a linear kernel since the linear kernel is a weaker version of an RBF kernel.

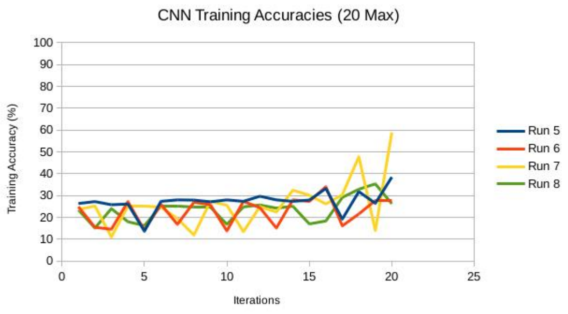
The Histogram of Oriented Gradients might have worked better than Bag of Words because Histogram of Oriented Gradients looks at the whole image; it might have done better than Color Histograms because it focus on the content of the paintings and not the colors used. This might be the reason it struggles to perfectly differentiate between Van Gogh, Monet and Turner since the content of their paintings are similar. The Histogram of Oriented Gradients worked better than the Convolutional Neural Network since the dataset was too small to train a CNN well.

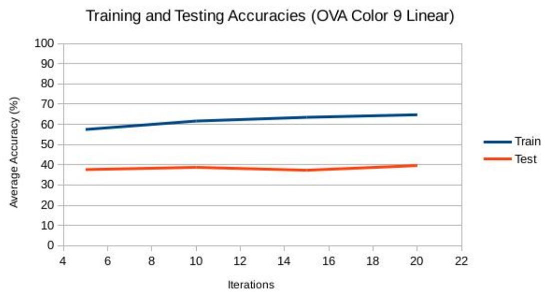
9.2 Going Forward

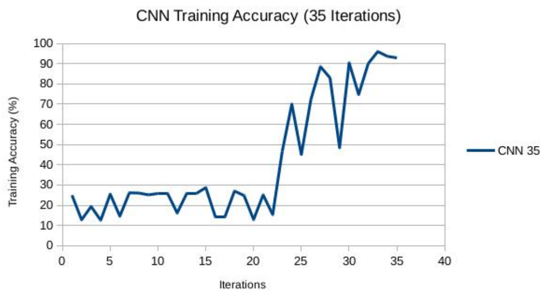
Going forward, since the Convolutional Neural Network did not do so well, to increase the accuracy, we would have to find a way to increase the data set and not resize the images in a way that ruins the aspect ratio. The way we would do this would be too keep the images such that the largest dimension is 500 pixels and the aspect ratio is kept the same, and then we would take non overlapping 100 by 100 tiles.

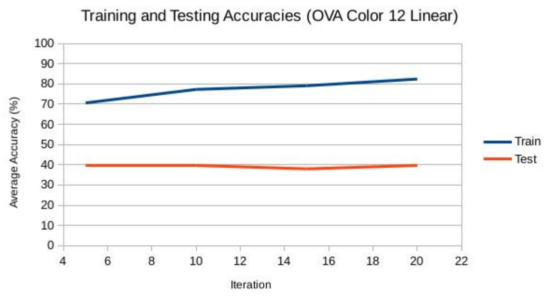
10 Appendix

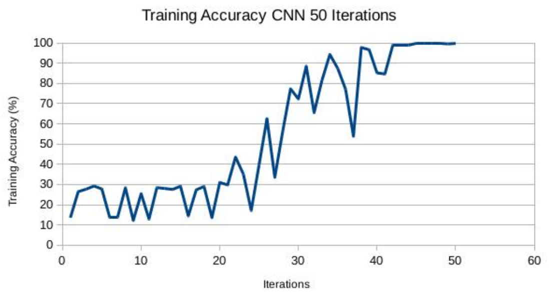

Training and Testing Accuracy (OVA BOW 800 Linear)
Average Accuracy (%)
Iterations
Train
Test

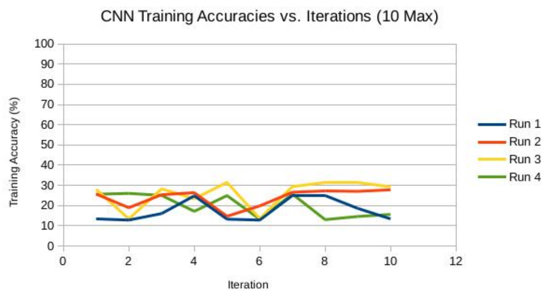

CNN Training Accuracies (20 Max)
Training Accuracy (%)
Iterations
Run 5
Run 6
Run 7
Run 8

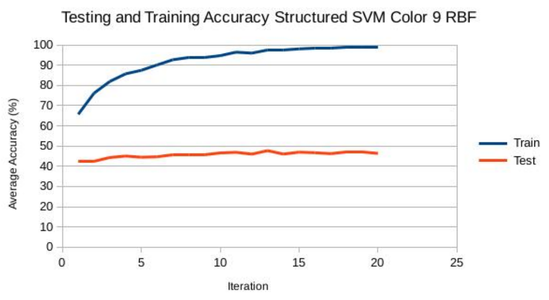

Training and Testing Accuracies (OVA Color 9 Linear)
Average Accuracy (%)
Iterations
Train
Test

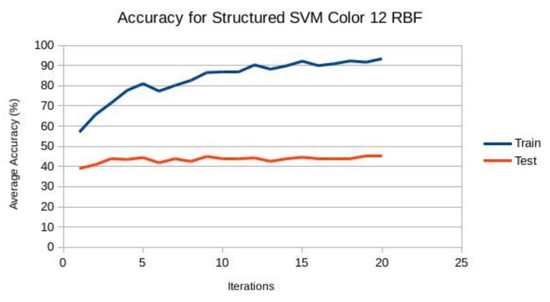

CNN Training Accuracy (35 Iterations)
Training Accuracy (%)
Iterations
CNN 35

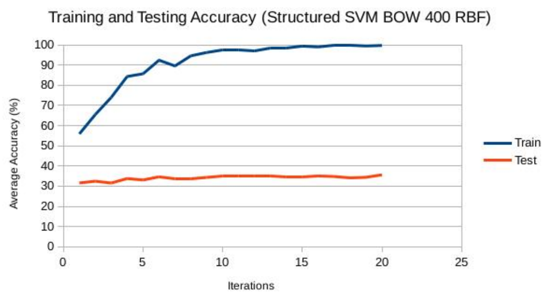

Training and Testing Accuracies (OVA Color 12 Linear)
Average Accuracy (%)
Iteration
Train
Test

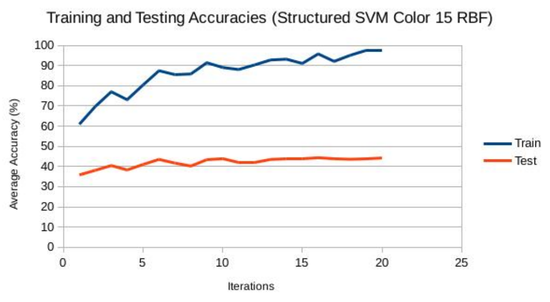

Training Accuracy CNN 50 Iterations
Training Accuracy (%)
Iterations

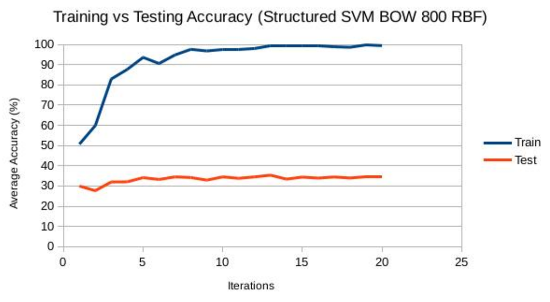

CNN Training Accuracies vs. Iterations (10 Max)
Training Accuracy (%)
Iteration
Run 1
Run 2
Run 3
Run 4

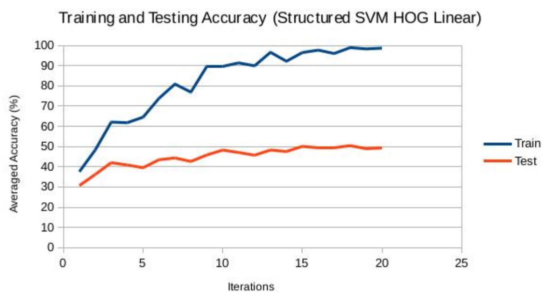

Testing and Training Accuracy Structured SVM Color 9 RBF
Average Accuracy (%)
Iteration
Train
Test

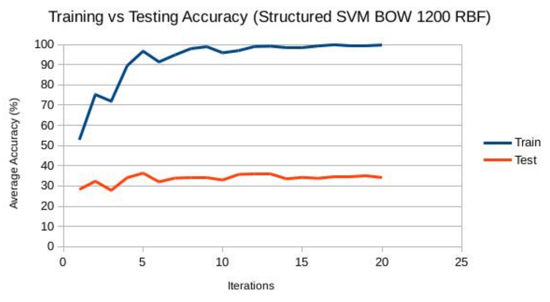

Accuracy for Structured SVM Color 12 RBF
Average Accuracy (%)
Iterations
Train
Test

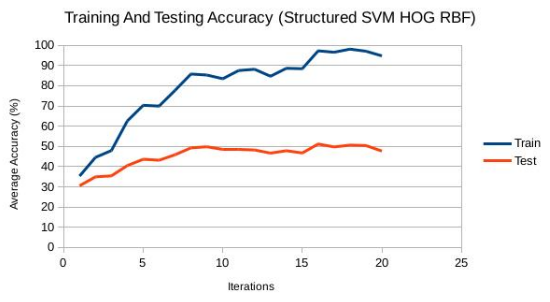

Training and Testing Accuracy (Structured SVM BOW 400 RBF)
Average Accuracy (%)
Iterations
Train
Test

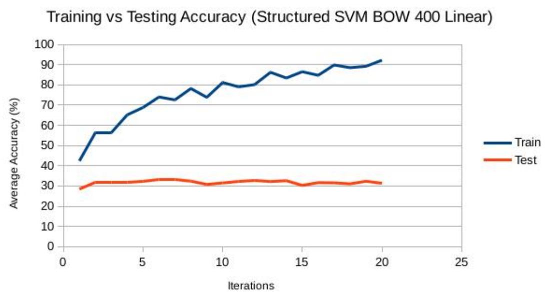

Training and Testing Accuracies (Structured SVM Color 15 RBF)
Average Accuracy (%)
Iterations
Train
Test

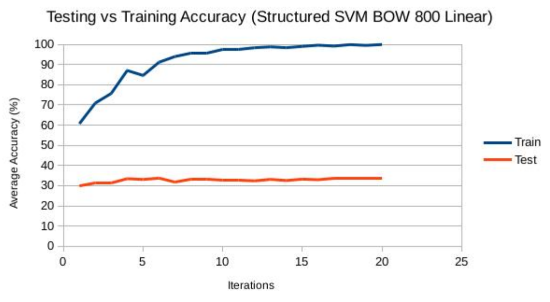

Training vs Testing Accuracy (Structured SVM BOW 800 RBF)
Average Accuracy (%)
Iterations
Train
Test

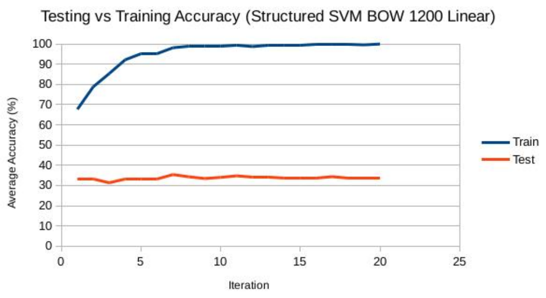

Training and Testing Accuracy (Structured SVM HOG Linear)
Averaged Accuracy (%)
Iterations
Train
Test

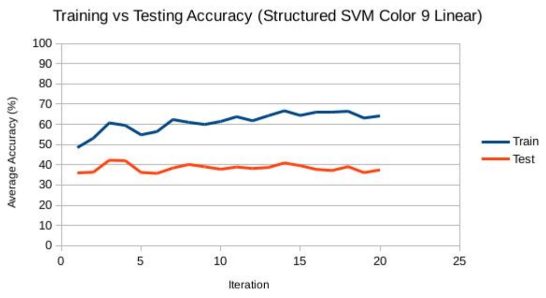

Training vs Testing Accuracy (Structured SVM BOW 1200 RBF)
Average Accuracy (%)
Iterations
Train
Test

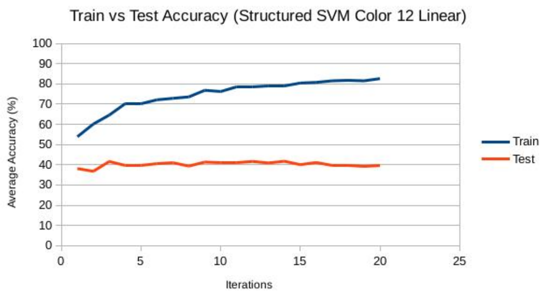

Training And Testing Accuracy (Structured SVM HOG RBF)
Average Accuracy (%)
Iterations
Train
Test

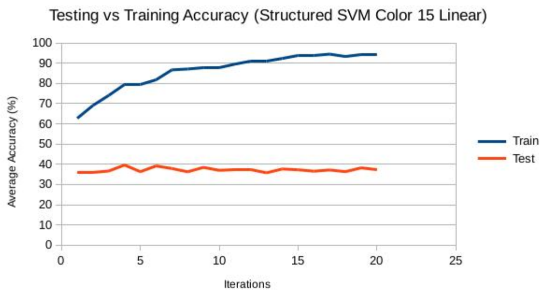

Training vs Testing Accuracy (Structured SVM BOW 400 Linear)
Average Accuracy (%)
Iterations
Train
Test


Testing vs Training Accuracy (Structured SVM BOW 800 Linear)
Average Accuracy (%)
Iterations
Train
Test


Testing vs Training Accuracy (Structured SVM BOW 1200 Linear)
Average Accuracy (%)
Iteration
Train
Test


Training vs Testing Accuracy (Structured SVM Color 9 Linear)
Average Accuracy (%)
Iteration
Train
Test


Train vs Test Accuracy (Structured SVM Color 12 Linear)
Average Accuracy (%)
Iterations
Train
Test


Testing vs Training Accuracy (Structured SVM Color 15 Linear)
Average Accuracy (%)
Iterations
Train
Test

References

[Blessing, 2010] Alexander Blessing and Kai Wen. 2010. *Using Machine Learning for Identification of Art Paintings.* Stanford University, Stanford, CA. Link

[Dala, 2005] Navneet Dalal and Bill Triggs. 2005. *Histogram of Oriented Gradients for Human Detection.* INRIA Rhone-Alps, Montbonnot 38334, France. Link

[Rublee, 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski. 2011. *ORB: An efficient alternative to SIFT or SURF.* Willow Garage, Menlo Park, CA. Link

[Cortes, 1995] Corinna Cortes and Vladimir Vapnik. 1995. *Support-Vector Networks.* Machine Learning. Link

[Shalev-Shwartz, 2011] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, Andrew Cotter. 2011. *Pegasos: Primal Estimated Sub-Gradient Solver for SVM.* Mathematical Programming. Link

[Keerthi, 2003] S. Sathiya Keerthi and Chih-Jen Lin. 2003. *Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel.* Neural Computation. Link

[Csurka, 2004] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cdric Bray. 2004. *Visual Categorization with Bags of Keypoints.* Xerox Research Centre Europe. Link

[Noord, 2015] Nanne van Noord, Ella Hendriks, and Eric Postma. 2015. *Towards Discovery of the Artists Style: Learning to Recognise Artists by their Artworks.* IEEE Signal Processing Magazine. Link

[Krizhevsky, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. *ImageNet Classification with Deep Convolutional Neural Networks.* Neural Information Processing Systems (NIPS) Conference. Link