Deep Art
Mark A. Martinez II
May 29, 2017

Abstract

Art style is something that defines the overall character of a piece of art. Style is often synonymous with era, but also contains information such as the content of a piece of art and other rich information. I devised a convolutional neural network that was trained on fragments of pieces of art to detect the broad art style of a piece. I used a subset of art totaling 25957 pieces and spanning 37 (and in some experiments 21) styles of art to classify what style a piece of art may be. I compared this deep learning approach with a traditional texture classification approach and using transfer learning to retrain a network trained on ImageNet data. The transfer learning net performed the best with a peak accuracy on a test set with 30.78%, eleven times better than chance, correct total classification likely because of using objects in the images to help classify style. The best CNNs made from art data only used small convolutional filters that were repeated one after another. This likely helped with detecting fine detail in an image.

1 Introduction

Deep Neural Networks have revolutionized the field of machine learning and have also seismically altered the field of computer vision. Various complicated techniques used in, for example, detecting objects or locating faces in an image have been completely bested by deep convolutional neural networks (CNN) in the previous years. One area that has not gained much traction is general understanding of an image. CNNs have isolated objects and labeled what is happening in pictures, but overall style is still something that is not fully fleshed out.

Art is a domain that is seen as a field that is immune to machine learning and artificial intelligence, but in this specific example of isolating art style there are often formulaic rules that a person can follow to figure out the style. For example, if a painting has swirly brush strokes and pastel colors it is very likely to be an impressionist painting.

Here I attempt to use a CNN to identify the overall style of a piece of art. I use a CNN that is trained on either 37 or 21 different art styles that vary in era of creation and culture of origin. I will compare this output with a traditional method of texture classification that evaluates adjacent pixel intensities to see if there is a real advantage to using CNNs.

Figure 1: Examples of art used in classification

(a) Cubism  (b) Modern  (c) Ukiyo-e
2 Methods

2.1 Data Pre-Processing

The data was collected from a Kaggle Competition [Kag] that originally had datasets from Wikiart.org in a competition to detect when a piece of art was likely to be from the same artist. This dataset had near complete descriptors for each of the pieces of art that included style, date of creation, and artist. I re-purposed this dataset and only retained the style information for each piece. The complete dataset consisted of 65534 unique images of which I used 25957 images for the training set and 37760 images for the test set.

After downloading the data I analyzed the dataset to find the most populated styles to use in my analysis. I initially used a threshold that each style must have at least 400 unique images, but later on in testing I made the testing more stringent such that there had to be at least 1000 images in the style for me to be able to use it. I restricted the styles to only the most popular categories because some of the styles were extremely sparsely populated and would only turn into noise when used in an algorithm. In the first round of testing I used 37 different art styles and the second round I used 21 styles. The styles used in the experiments are shown below in the style specific classification found in the results section.

I utilized the Python Imaging Library (PIL) to modify both the training and test sets to make it easier and more efficient to test using neural networks and other machine learning algorithms. One issue with the data is that each of the images has a different size. This makes a straight comparison of each image impossible using most algorithms. One option I could have used was to shrink each image down to a consistent size, but I chose against this option because transforming each of the images like this would degrade the quality of the image and in most cases severely alter the image and destroy the intricate style information that would be encoded in the full image.

Figure 2: Examples of art used in classification

(a) Full Image  (b) Cropped, Saturated, and Sharpened Image

Instead I chose to work with another approach. I would crop each of the images to a set size randomly from the image. Style is largely something that exists as a whole for an image, but it is possible to understand style from small portions of an image. Brush strokes of a painter, color, or general types of objects (such as humans, cherub, or fruit) would be highly indicative of a particular genre to a human viewer. As such, I believed that using smaller subsections from an image would give adequate information to an algorithm when trained over many different images and and styles. I took five random crops of either 227x227 pixels or 100x100 from each of the training and test images. This approach helped with allowing me to use stylistically unaltered images in the algorithms as well as increase the number of examples I could use in the test and training sets. To help the algorithms notice things such as brush strokes and color in the images I increased the sharpness metric of each metric. And to make the color contrasts in the image to be more apparent I also increased the saturation of the images.
2.2 Traditional Texture Classification

In order to compare the results of the neural networks I also incorporated a traditional texture classification technique to compare with the neural network. The analysis I chose to use came from professor Kung’s textbook Digital Neural Networks [Kun93]. The example used in Kung’s textbook creates a histogram for each pixel in the image that finds the local texture surrounding that pixel.

Each pixel that is surrounded completely by 8 other pixels gets a corresponding vector where the values of the vector correspond to

\[ x = \text{number of pixels of lower intensity than the center pixel} \]
\[ y = \text{number of pixels of equal intensity than the center pixel} \]
\[ z = \text{number of pixels of greater intensity than the center pixel} \]

There are 45 total combinations of the x,y,z values for the vectors and each pixel’s vector will put 1 input into the histogram of one of those combinations.

Each image (or crop of an image) has a corresponding histogram that that can be used to as describe the texture of the image. Using this as the data point and the style as the label I took histograms with corresponding style labels and ran these 45 dimensional vectors into a support vector machine (SVM) to classify the images.

I used several sizes for creating the histogram vectors. I initially used the entire image, but that caused the SVM to overfit so I experimented with different random crop sizes of the image. The images used in this texture based SVM were the same crops from the training set collected for the neural network. Instead of using the entire dataset though, I varied the number of samples used to train the SVM to make sure that the model did not overfit the data.

2.3 Convolutional Neural Network

I constructed and trained Convolutional Neural Networks (CNN) using a Caffe framework on the image samples I collected. The training set numbered 116205 data points for training that spanned 21 styles and 129785 data points that spanned 37 styles. The test set numbered 33595 data points that spanned 21 styles and 37760 data points for the 37 styles.

The general structure that I used for the network was based off of the structure used by Alexnet.[Kri12] When using 227x227 images the network contained 5 convolutional layers, 3 pooling layers, and 3 fully connected layers along with relu and normalization layers interspersed when necessary. When using 100x100 images the network contained 4 convolutional layers, 3 pooling layers, and 3 fully connected layers along with relu and normalization layers interspersed when necessary. For all networks I implemented a 50% dropout when reaching a fully connected layer during the training time.

I decided to shrink the image size down from 227x227 to 100x100 because I feared that I was capturing too much information in the larger crop sizes of the images. Because this CNN is supposed to tell when something is a particular style by training on too much specific detail of each individual image I believed that it would not be as good at generalizing things such as brush stroke and instead capture large scale features such as objects in the image. By starting with a smaller image I had to use smaller convolutional filters and more of them in succession. This had the added benefit of looking at detail more so than when using larger filters.

The meta-parameters I used when training the network were as such

<table>
<thead>
<tr>
<th>Meta-Parameters</th>
<th>227x227</th>
<th>227x227 ImageNet Transfer</th>
<th>100x100 v1</th>
<th>100x100 v2</th>
</tr>
</thead>
<tbody>
<tr>
<td>base learning rate</td>
<td>0.00005</td>
<td>0.0015</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>gamma</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>stepsize</td>
<td>5000</td>
<td>20000</td>
<td>8000</td>
<td>8000</td>
</tr>
<tr>
<td>momentum</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>weight decay</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
<tr>
<td>max iterations</td>
<td>20,000</td>
<td>100,000</td>
<td>20,000</td>
<td>20,000</td>
</tr>
</tbody>
</table>
Figure 3: The pixels surrounding the center pixel have 3 pixels that are of lower value, 3 pixels that are the same value, and 2 pixels that are of greater value. The vector for this center pixel is then created accordingly.

Figure 4: This shows that the pixel grouping is derived from a texture.

Figure 5: This shows examples of the different histograms that are used as inputs to the SVM.
Figure 6: The basic structure of Alexnet was used as the template for this network. When testing on 227x227 images I left the structure intact as it was the same image input size. For 100x100 images I modified the sizes of the convolutional and pooling layers as well as removed one convolutional layer.

2.4 Transfer Learning

I wanted to see if it were possible to transfer the concept of object recognition from networks trained on this particular task to detecting art style. Because so much of style is also based on what is in the image as well I figured that transfer learning would provide interesting results. I trained the same dataset on the final fully connected layer of the BVLC Caffenet.

The original network was trained on the ImageNet dataset and learned on 1,000,000 images to classify 1,000 classes.

The network started training on it's finalized weights, but the final fully connected layer was restructured so that the weights were randomized and the learning rate was increased dramatically for this layer only. To get adequate training on the initial layers and the last fully connected layer this network was made to run much longer than the other networks.

This network also happened to use 227x227 images so the original 37 styles were used instead of the truncated list that used the smaller number of styles.

3 Results

3.1 Traditional Texture Classification

I tested a number of different hyper parameters, image sizes, and number of styles when testing the efficacy of the SVMs based off of the pixel based histograms.

From the graph we see that the 100x100 crops produced the best results compared to using the full image size of 227x227 or 32x32 crops. A possible reason for this is that by focusing on specific smaller regions of an image instead of larger chunks it is possible to get a lower level pattern understanding of an art style, but not so small that the algorithm over generalizes.

The best results happened with a training size of 6500 images and 14.71% accuracy. This is better than random chance which would sit at 4.76%, but still far lower than what I had expected to be the outcome.

3.2 Convolutional Neural Network

The results from the loss plotted against the number of iterations are shown in the 100x100 v2 CNN and only the training loss is shown against the 100x100 v2 CNN. Only the training loss is shown for v1 because the test loss was not captured at this phase of creating this net.

Among the CNNs constructed from the art data alone the configuration for CNN 100x100 v2 gave the best performance in classifying the images correctly. The combination of having a learning rate that decayed quickly along with smaller filters made capturing fine details in paintings easier. [Sim14] training loss for this configuration was 0.15 at its lowest levels and achieved 20.84% classification of the test set.
3.3 Transfer Learning

The transfer learned neural network achieved at peak accuracy of 30.78% classification of the same test set used in the other neural networks. This network performed the best among all of the different networks. The figure that shows the test and training loss is shown in figure 10.

I tested on both the weights from both the 20000 iteration and the 100000 iterations of this transfer learned network. I used the 20000 because it had the lowest test loss on a different set of test data and I used the 100000 because it was the most finetuned network. The results of 30.78% correct classification was achieved with the 100000 iteration network, while the 20000 iteration achieved a slightly lower 28%. I believe that the network might have had the lowest loss on a lower iteration, but was not as generalizable as a more finetuned network.

<table>
<thead>
<tr>
<th>Table 2: Meta- Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>227x227</td>
</tr>
<tr>
<td>Transfer Learned 227x227</td>
</tr>
<tr>
<td>100x100 V1</td>
</tr>
<tr>
<td>100x100 V2</td>
</tr>
</tbody>
</table>

4 Discussion

4.1 Traditional Texture Analysis

Although the results for the traditional texture analysis were quite low, the highest performing machine learning algorithm peaking at 30.78% accuracy, it was not far worse than an optimized neural network. The simplicity and speed of this method makes it appealing if it were possible to push the accuracy to higher values. This algorithm still performs several times higher than random chance, but would be far worse than a person trained on this task specific given examples to learn from.

Ways that this approach can be improved would be to optimize the SVM and to find the best size of the data to test on.
Figure 8: Loss plotted against number of iterations from the 100x100 CNN V1 training

Figure 9: Loss plotted against number of iterations from the 100x100 CNN V2 training
4.2 Convolutional Neural Network

The best performing CNN was the 100x100 v1. Considering that the only difference in this configuration from the other 100x100 network was a far higher gamma value, it would seem that decreasing the learning rate at a much faster pace would give the best results in this type of a problem.

The smaller convolutional filters seem to really help the performance of the algorithm. What would be useful would be to stack more small filters on top of each other to really try and find the limit of where finding more detail in a painting is impossible and where more detail would only hinder the algorithm.

4.3 Transfer Learning

In a bit of a surprise Transfer Learning performed by far the best of the classifiers. It had several things going for it that the other algorithms did not have. This network was trained on far more data and was able to distinguish things such as objects with high accuracy. This network likely used this specialty of already detecting objects to use the object detecting ability to help distinguish among different styles. On top of this this network likely already was sensitive to color in ways that helped the classification between styles that were very highly color dependent.

5 Conclusion

The far best technique in identifying different art styles was to use a network that was pretrained on ImageNet data and used for classification of objects. The takeaways that can be used for more research is that having large quantities of data is extremely important. It seems as if that the over 100,000 samples used in the training were insufficient to actually training a network to the best classifying power. On top of that using a quickly dropping learning rate helped in the discrimination. It might be interesting to see if a network that is only trained on art data, but with many more data samples, would compare to a transfer learned network on object detection. It might continue to be the case that having diversity of data, not just art, may actually make discrimination easier.
References


