**Artist Recognition for Fine Art Paintings with Deep Learning**

CS688: Pattern Recognition (Fall 2017)

Bhavika Tekwani, Jaecheon Jeong, Melanie Baker

**Abstract**

In this paper, we present a series of experiments to explore large scale classification of fine-art paintings from the Wikiart dataset. Our objectives are two-fold: first, we want to train classical machine learning models like Random Forests & Support Vector Machines to classify Impressionist paintings. Secondly, we want to train a convolutional neural network (CNN) end-to-end in order to use the deep learning approach that has gained popularity in the image classification domain. As we see later on, identifying paintings by artist is more challenging than detecting faces in images or classifying objects. We also explore model interpretability by exposing unlabeled images (not of paintings) to our trained models and trying to understand how machines perceive & recognize artistic style.

**Dataset**

We used the Wikiart dataset released by the authors of the ICIP2016 paper on deep convolutional networks for fine-art paintings classification [5]. The dataset in its entirety is 27 GB in size and contains over 80,000 images spanning 2108 artists and 27 genres.For our artist classification task, we use only the Impressionism genre (13060 images). Furthermore, to quickly prototype several methods and iterate on them, we limit all our experiments to 15 artists chosen in no particular order. For each artist, we select a set of 40 images. Therefore, our dataset for this task contains 600 images.

The complete list of artists is as follows:

|  |  |
| --- | --- |
| **Artist** | **Class Label** |
| Adam Baltatu | 0 |
| Alfred Sisley | 1 |
| Antoine Blanchard | 2 |
| Arkhip Kuindzhi | 3 |
| Armand Guillaumin | 4 |
| Auguste Rodin | 5 |
| Berthe Morisot | 6 |
| Camille Pissarro | 7 |
| Childe Hassam | 8 |
| Claude Monet | 9 |
| Constantin Artachino | 10 |
| Cornelis Vreedenburgh | 11 |
| Edgar Degas | 12 |
| Edouard Manet | 13 |
| Eugene Boudin | 14 |

**Setup**

Hardware:

We did not use any specialized hardware for running our experiments. Available memory ranged from 4 GB - 8 GB on i5 and i7 processors. Although we used a deep learning approach, we did not use GPUs to train our convolutional neural networks.

Software:

We used Python 3 for all our experiments. The libraries we used for machine learning & deep learning include *pandas*, *scikit-learn, numpy, PIL, OpenCV, Tensorflow, Keras & PyTorch*.

**Literature Review**

A literature review of several gave us insight into what features and methods have worked best for similar tasks. Those papers are below with our general findings.

A. Blessing and K. Wen’s paper was written in 2010, which explains why a deep learning network was not part of their architecture. The author’s initially implemented their solution using Naive Bayes to create their features, but found it wasn't sufficient enough and they went to an SVM. The main thing we got out of this was different features (ex. SIFT, HOG, color histograms, etc.) we could focus our algorithm on if we didn't implement our solution as a CNN [1].

In B. Saleh and A. Elgammal’s paper, the authors’ goal was to develop an algorithm that was “able to make aesthetic-related semantic-level judgments, such as predicting a painting’s style, genre,and artist, as well as providing similarity measures optimized based on the knowledge available in the domain of art historical interpretation”. The main takeaway, since our main goal is to identify paintings by artist, is that for Artist Classification, their CNN that learned using the Information-Theoretic Metric Learning (ITML) trained the fastest and had the best accuracy [2].

Y. Hong and J. Kim’s paper was published in 2017 and is very similar in our original approach to the artist classification problem. It talks about each of the different CNN architectures they came up with and their results. With CNNs doing so well in image recognition, it's clear this is the more modern approach and we think if we each developed a CNN we could improve our success on this task [3].

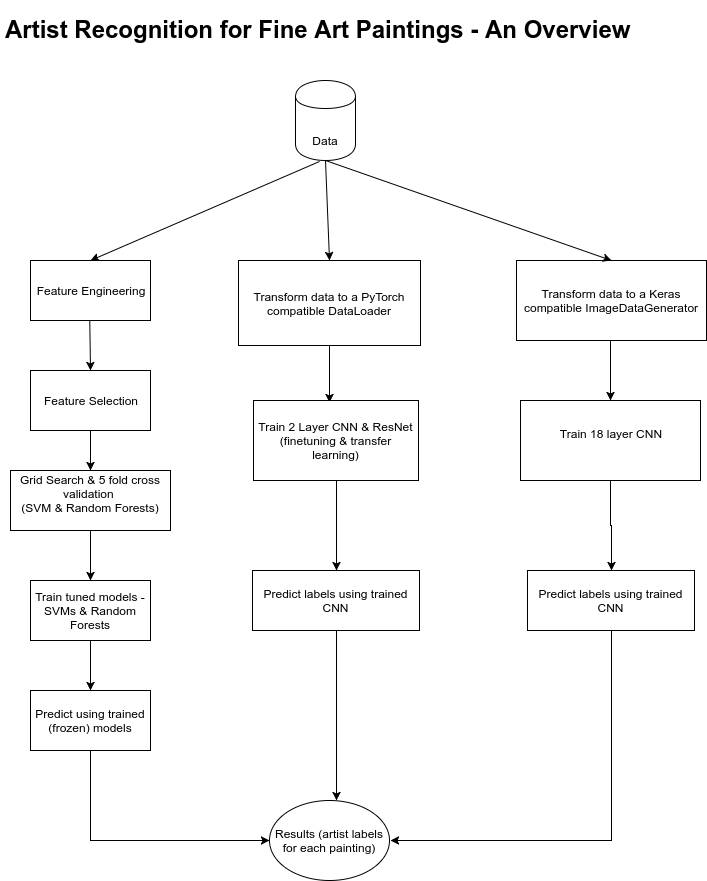
A. Elgammal’s paper shows another option for building a learning architecture if we didn't want to use python and focus more on classifying the more general painting style. This person was able to use the MATLAB image processing libraries and create an SVM that got up to 60% classification on painting style. It's worse than a CNN, but if we had time we could do an experiment [4].

After a review of the above, we conclude that our approach would be unique in how we architect our CNN and the features we could use for the SVMs & Random Forests. We also want to try different learning rates for the deep nets and see the effect on classification accuracy per class.

**Approach**

Our experimental setup is divided into 3 parts - one is machine learning with models like Random Forests, & SVMs, the second is an 18 Layer Convolutional Neural Network built in Tensorflow and Keras, the third is a 2 Layer Convolutional Neural Network built with PyTorch.

The diagram below shows the division of those 3 steps. As is evident from the 3 branches, using deep convolutional nets helps avoid certain steps like feature selection & cross validation. As fine art painting classification is a difficult task, we assume that feature engineering would require some creativity & domain knowledge. We were hoping deep learning would help in this case since no feature engineering is required.



For all our experiments, we keep the distribution of data constant. We have 600 paintings by 15 artists which we split into a train and test dataset with a 60-40 distribution. Our training data now has 360 paintings and unsequestered test data has 240 paintings.

Preprocessing

Feature Engineering & Selection

Models

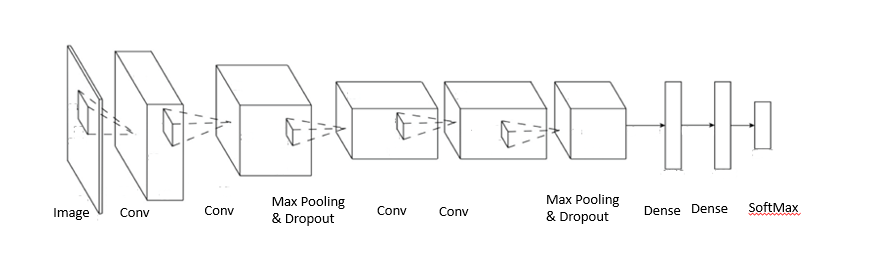
# Each experiment should contain details of the model, parameters (default & tuned), training time, loss function used, features used, cross validation

1. Random Forests
2. Support Vector Machines
3. Gradient Boosted Decision Trees (GBDT)
4. CNN - 12 Layer

In order to compare and possibly improve upon our 2-layer CNN accuracy, we wanted to experiment with a more complex network similar to ones we’ve seen used in research for classifying large data sets. Therefore, we utilized a pre-built 12 layer CNN architecture using Keras that had performed classification well for the CIFAR-10 dataset. The network architecture is detailed as such:

|  |
| --- |
| **Layer** |
| Conv2D [out=32, kernel = 3, stride = 1, padding = 0, act=relu] |
| Conv2D [out=32, kernel=3, stride=1, padding=0, act=relu] |
| MaxPool2D [kernel =2, stride = 2] |
| Dropout[rate =.25] |
| Conv2D [out=64, kernel = 3, stride = 1, padding = 0, act=relu] |
| Conv2D [out=64, kernel = 3, stride = 1, padding = 0, act=relu] |
| MaxPool2D [kernel =2, stride = 2] |
| Dropout[rate =.25] |
| Flatten[ ] |
| Linear [out = 512, act=relu] |
| Dropout[rate =.5] |
| Linear [out = 15, act=softmax] |

Visually, this neural net can be represented as shown below. We changed the input to be 210 x 140 sized images of the paintings and the network was trained using various learning rates of .01, .001, and .0001, a batch size of 15 and initially 100 epochs. The loss function used was cross-entropy loss. The .0001 learning rate produced the best model. I varied the number of epochs as well from 80, 100 and 125, but 100 proved best, achieving ~78% accuracy on the training set. Training time using the above parameters took ~3.5 hours.



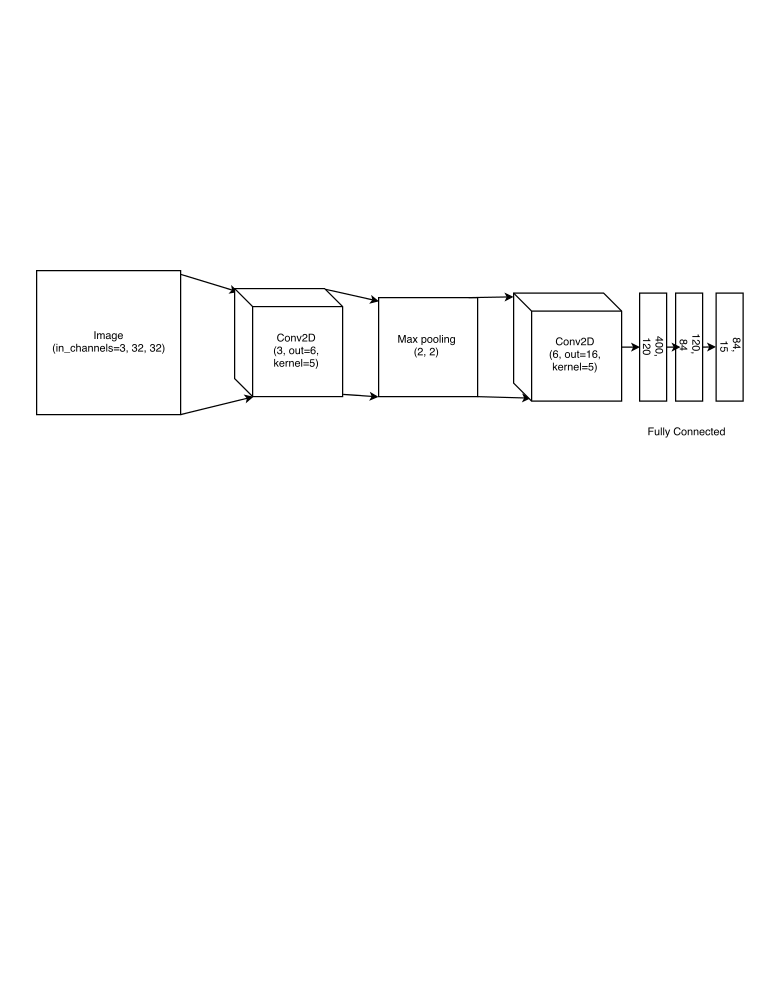
1. CNN - 2 Layer

We implemented a simple 2 layer convolutional neural network in PyTorch and trained it with different learning rates. Every image used as input to this network was resized to 32 \* 32. The network architecture is detailed below:

|  |
| --- |
| **Layer** |
| Conv2D [in=3, out=6, kernel=5, stride = 1, padding = 0] |
| MaxPool2D [kernel =2, stride = 2] |
| Conv2D [in=6, out=16, kernel=5, stride=1, padding=0] |
| Linear [in = 400, out = 120] |
| Linear [in = 120, out = 84] |
| Linear [in = 84, out = 15] |

Visually, this neural net can be represented as shown below. We use this 2 layer CNN as a baseline to measure other CNN architectures. The last fully connected layer emits predictions for 15 classes.

The loss function used is Cross Entropy Loss. We tune the weights in the CNN using the SGD optimizer with varying learning rates but keeping the momentum constant at 0.9. The learning rates were varied from 1e-5 to 1e-1, but the optimal learning rate for this network was found to be 0.001. Since PyTorch allows data to be processed in batches, we set the batch size to 4 since our training dataset was small & the network wasn’t too deep. The network was trained over 100 epochs which was excessive since we observed that the loss increases instead of decreasing after 50-60 epochs. Again, this can be attributed to the size of the training data.

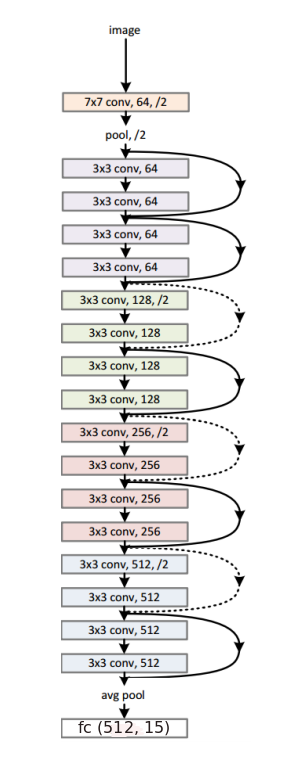


1. ResNet-18 (transfer learning)

Since it is quite uncommon to train complex deep architectures from scratch, we decided to experiment with some popular architectures that have been known to perform well on image classification problems. One such architecture is ResNet which has several variants that performed well on the ImageNet dataset that contained 1000 classes. Of the architectures available, we use the simplest one - ResNet-18. It has 18 convolutional layers & several pooling & dropout layers. For our task, we simply modify the last fully connected layer to give us the predictions for 15 classes instead of 1000.

ResNet-18 needs an input size of 224 \* 224, so we resized each image to these dimensions in order to make the network reusable. Other than the resizing, we do not perform any transformations on the original paintings.

The modified architecture with a change in the last fully connected layer is shown below.



1. ResNet-18 (fine tuned)

Another approach we tried was fine-tuning the ResNet-18 model to adjust the weights in each layer for our specific task of classifying paintings. To do this, we used the pretrained model in PyTorch and retrained it with different learning rates again varying them in the range 1e-5 to 1e-1.

The batch size was set to 4 & we trained the model over 100 epochs. As observed with the 2 layer CNN, the loss tends to oscillate between 0.007 and 0.004 once we reach 40-50 epochs. The best learning rate was found to be 0.001.

Model Interpretability

Deep learning models are often hard to debug & tend to appear as black boxes. For our task of fine art classification, we wanted to understand if some of the ways humans perceive art and objects depicted in art are reflected in deep learning models. To do this, we selected a few images that were not works of art & did not have any artist labels present in our training data. Once we trained our models, we used each model and asked it to predict an artist for one of these random images.

Below, we present the new test images (not actual paintings) and the labels our model predicted for them. What we hope to observe here is not accuracy but rather the similarity between the test image & some of the training images for the predicted artist. This similarity can be in terms of the subject of the painting (human, nature, physical object, animals, etc.), overall color, tone, etc.

# Jay’s demo images

# Bhavika’s demo images

Input 1:



**CNN - 2 Layer**

Predicted artist: Édouard Manet

**ResNet18 fine-tuned**

Predicted artist: Armand Guillaumin



Input 2:



**CNN - 2 Layer**

Predicted artist: Arkhip Kuindzhi



**ResNet-18 fine-tuned**

Predicted artist: Camille Pissarro  


Input 3:



**CNN - 2 Layer**

Predicted artist: Claude Monet



**ResNet-18 fine-tuned**

Predicted artist: Cornelis Vreedenburgh



As we observe from the demo inputs & paintings we trained on, both models seem to do well when it comes to understanding the subject of the painting - a human in input 1, nature/scenery in input 2 and

a water body & boat in input 3.

# Melanie’s demo images



**Results**

The table below shows the overall precision, recall & accuracy for each model we trained.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** |
| Random Forest | 38.33% |  |  |
| CNN - 2 Layer | 8.75% | 76.56% | 8.75% |
| CNN - 18 Layers | 32% | 39% | 32% |
| ResNet-18 (transfer learning) | 7.9% | 63.7% | 7.9% |
| ResNet-18 fine-tuned | 10.83% | 53.26% | 10.83% |

The table below shows classwise accuracy for artists listed in Table 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| Random Forest |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CNN - 2 Layer | 8 | 30 | 25 | 6 | 6 | 25 | 4 | 10 | 3 | 12 | 0 | 0 | 0 | 0 | 0 |
| CNN - 18 Layers | 19 | 38 | 100 | 6 | 13 | 38 | 19 | 56 | 31 | 19 | 25 | 0 | 56 | 38 | 44 |
| ResNet-18 (transfer learning) | 25 | 33 | 33 | 20 | 33 | 43 | 25 | 0 | 21 | 33 | 31 | 25 | 20 | 20 | 25 |
| ResNet-18 fine-tuned | 34 | 25 | 50 | 43 | 25 | 25 | 25 | 29 | 37 | 0 | 50 | 18 | 31 | 25 | 25 |

**Conclusions**

The highest accuracy we were able to achieve was 38% using random forest, with the next being 32% using the 12-layer CNN, and 7% using the 2-layer CNN. This is far away from the current state-of-the-art and given our results, we’ve concluded that artist recognition is an extremely difficult problem in machine learning that may require more data preparation than we used. To improve our model, we could have used paintings from only well known artists with unique styles or increased the number of training examples per artist. However, the latter could risk further model confusion.

**References**

[1] A. Blessing and K. Wen, Using machine learning for identification of art paintings,

Tech. Rep., 2010.

[2] B. Saleh and A. Elgammal, Large-scale classification of fine-art paintings: Learning

the right metric on the right feature, ArXiv Prepr. ArXiv150500855, 2015.

[3] Y. Hong and J. Kim, Art Painting Identification using Convolutional Neural

Network, Int. J. Appl. Eng. Res., vol. 12, no. 4, pp. 532–539, 2017.

[4] Creating Computer Vision and Machine Learning Algorithms That Can Analyze

Works of Art - MATLAB & Simulink. [Online]. Available:

<https://www.mathworks.com/company/newsletters/articles/creating-computer-vision-and-machine-learning-algorithms-that-can-analyze-works-of-art.html>. [Accessed: 18-Sep-2017].

[5] ICIP2016, Wei Ren Tan, Chee Seng Chan, Hernan E. Aguirre and Kiyoshi Tanaka, Ceci n'est pas une pipe: {A} deep convolutional network for fine-art paintings classification. <http://web.fsktm.um.edu.my/~cschan/doc/ICIP2016.pdf>