How Abstract Can You Get?
A Deep Learning Take on Art

Michael A. Alcorn

Introduction

Convolutional neural networks (CNNs) have seen great success in a number of different machine learning tasks, from computer vision (where they were originally developed; e.g. [1]), to natural language processing (e.g., [2]), to music information retrieval [3]. Many of the concepts learned by CNNs can be surprisingly abstract. Consider, for example, the CNN built by van den Oord et al. [3], which was capable of learning different subgenres of electronic dance music, among other seemingly complex musical concepts [4]. However, the question remains, “How abstract can they can get?”.

The human brain is capable of highly abstract thought due to its specific biological properties [5]. In order to fully understand the limitations of current artificial intelligence technology, we need to determine which areas of human abstraction are most difficult for machines to capture. One of the most defining abstractions of human consciousness is visual art [6]. Art movements, a concept used by art historians to group different works of art and artists based on the visual and philosophical qualities of the art, represent an even higher layer of abstraction. For my project, I trained three different CNNs to classify the movements and artists of paintings. In addition to this academic goal, I aimed to familiarize myself with the Theano framework and immerse myself in the workflow of a researcher/industry worker in computer vision. Although my CNNs performed much better than random classifiers (41.46% vs. 28.44% and 12.68% vs. 4.91% for movements and artists, respectively), there remains considerable room for improvement.

Methodology

The training data was obtained by downloading 25,518 images of paintings from the Web Gallery of Art (see “getImages.py”, “getImagesTimeout.py”, and “handleDuplicates.py”). The Web Gallery of Art provides the artist for each painting via a database. In all, 2,963 different artists were represented in the dataset; however, the vast majority of the artists were considerably underrepresented, so the models were only trained on the 100 most frequent artists (11,565 paintings; see “artistCounts.py”). The movements (24 total) for 15,580 of the paintings were obtained by cross-referencing the artist names from the Web Gallery of Art database with the artist names listed in each of the movements found on Artcyclopedia (see “getMovements.py”), but the models were only trained on the 14 most frequent movements (15,377 paintings), again, due to the uneven distribution of paintings (see “movementCounts.py”).

Cross-referencing the artist names ended up being a non-trivial task because the two websites differed in both their use of special characters (e.g., é vs e) and (occasionally) their spellings of artist names. To account for these anomalies, I wrote a program (see “artistMovements.py”) that found those artists from Artcyclopedia that had the closest Levenshtein distance to each artist found in the Web Gallery of Art database. I then manually inspected the results and, for each set of candidate matches, either selected the best match, or, if none of the candidates seemed like appropriate matches, rejected the matches and that painting was omitted from the movements dataset.

The CNNs were built using the Theano framework and modified from the template provided in the deeplearning.net tutorial on LeNet. In all, I tested three different networks (see “convArtAttemptOne.py”, .“convArtAttemptTwo.py”, and “convArtAttemptThree.py”). The architecture for the first CNN (C1) can be seen in **Figure 1**. The other two CNNs (C2 and C3) were nearly identical to C1 except that C2 used rectified linear units (ReLUs) instead of sigmoid units (which used the hyperbolic tangent [tanh] activation function) and C3 used 50 10x10 filters on the first convolutional layer rather than 20 5x5 filters. ReLUs have been shown to train much faster than sigmoidal units, which is necessary when dealing with large networks and datasets (e.g., [7], [1]). The input to each CNN was a 100x100 RGB digital image, which is represented as three overlapping squares (one for each channel) in the figure. Each image was initially scaled down so that the smaller of the two dimensions had a length of 100 pixels (while retaining the original aspect ratio). The image was then cropped equally on either side to bring its longer dimension down to a length of 100 pixels (see “resizeImagesArtists.py” and “resizeImagesMovements.py”).

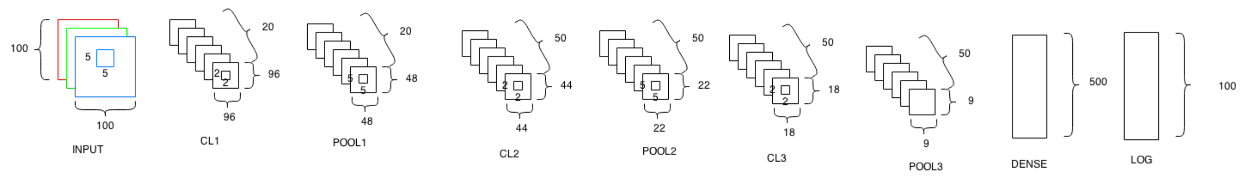

Figure 1. The architecture for C1.

The first convolutional layer contains 20 filters (i.e., feature detectors), each with 75 weights (5x5x3, or one set of 5x5 weights for each channel of the input image). The output of this first convolutional layer is 20 96x96 feature mappings. The output of each filter has dimensions of 96x96 because there are 96x96 overlapping 5x5 windows in a 100x100 grid. Essentially, each filter is determining its “confidence” that a single feature is present in each 5x5x3 window of the input image. In object recognition tasks (e.g., [1]), these filters often end up being edge detectors (i.e., each filter is detecting the presence or absence of an edge in each overlapping window of the input image). A 2x2 max-pooling function is then applied to each of the 20 feature maps resulting in 20 48x48 matrices.

The second convolutional layer has 50 filters, each with 500 weights (5x5x20, or one set of 5x5 weights for each feature map from the previous layer), resulting in an output of 50

44x44 feature mappings. A 2x2 max-pooling function is again applied to each of the 50 feature maps resulting in 50 22x22 matrices. The third convolutional layer also has 50 filters, each with 1,250 weights (5x5x50), resulting in an output of 50 18x18 feature mappings. A 2x2 max-pooling function is again applied to each of the 50 feature maps resulting in 50 9x9 matrices. The next layer is a fully-connected sigmoid layer containing 500 units. The final layer uses a logistic regression classifier to predict the output class. The cost function being minimized is the negative log likelihood of the models.

Results and Discussion

The results for each of the architectures can be seen in **Table 1** and **Table 2**. As you can see, C1 had the best testing accuracy for each classification task, although the performance between C1 and C2 seems comparable. C3, which used a 10x10 filter in the first convolutional layer, rather than a 5x5 filter, performed considerably worse than either C1 or C2. Interestingly, for the movement classification task, C3 never seemed to converge, while the other two networks did. In fact, most of the testing accuracies found for C3 during training were much worse than random, which suggests C3's best run (which is barely above random) might have "just been lucky". A 10x10 filter might be struggling to learn features from a 100x100 pixel representation of a painting because there might be too many low level features contained in such a window.

Table 1. Artist classification accuracy for each CNN. Each CNN was trained for 240 minutes on 11,565 paintings representing 100 different artists. A model selecting the most frequent artist in the training set would have a test accuracy of 4.82%.

Architecture	**Cross-Validation Accuracy**	**Test Accuracy**	**Best Time (min)**
C1	11.30%	12.68%	212
C2	12.05%	11.92%	238
C3	8.25%	9.44%	222

Table 2. Movement classification accuracy for each CNN. Each CNN was trained for 240 minutes on 15,360 paintings representing 14 different artists. A model selecting the most frequent artist in the training set would have a test accuracy of 28.06%.

Architecture	Cross-Validation Accuracy	Test Accuracy	Best Time (min)
C1	40.92%	41.46%	195
C2	41.00%	41.23%	191
C3	29.08%	28.49%	24

The filters learned by C1 for the first convolutional layer can be seen in **Figure 2.** Oftentimes, these low level filters can reveal features that are easy for humans to understand, such as edges and color gradients [1], but, in the case of my models, the filters do not seem to be easily interpretable.

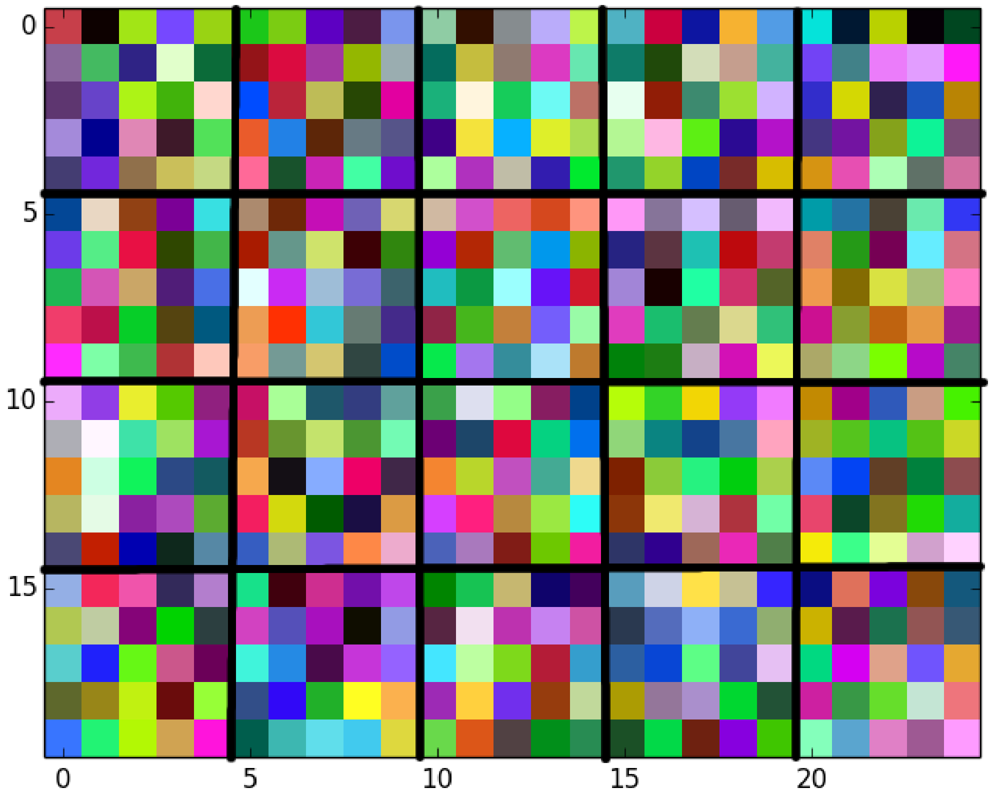

Figure 2. The 20 low level filters learned by C1.

As far as I am aware, CNNs have not been applied to art recognition (likely for good reason, as my experiments demonstrate), despite their success in other realms of computer vision, such as object recognition [1]. There are many nuances to art recognition that make it a much more difficult task than object recognition, but I believe progress can be made on the problem, though it will likely be difficult. To begin with, many artistic movements are defined, in part, by their stylistic approach. For example, Wikipedia describes Impressionist art as having:

> "Impressionist painting characteristics include relatively small, thin, yet visible brush strokes, open composition, emphasis on accurate depiction of light in its changing qualities (often accentuating the effects of the passage of time), ordinary subject matter, inclusion of movement as a crucial element of human perception and experience, and unusual visual angles."

To pick up on these stylistic subtleties, one would need to consider images of a much higher resolution than the 100x100 pixel images used in my models. Unfortunately, images of such high resolution would dramatically increase the computational time necessary to train the model, so sufficient hardware accommodations would be necessary.

In addition to their stylistic components, artistic movements are also often associated with certain philosophical ideas, or revolve around common thematic elements, which are frequently referenced in the works. For example, Baroque art typically showcases important mythological or historical moments. However, a model relying solely on a CNN trained on image pixels will be unable to incorporate this information. Adding a natural language processing component to the model may allow the model to learn the various abstract concepts used to classify art. An improved hypothetical model may incorporate three components: ***(1)*** a CNN trained on the artists/movements, like the one used in this paper, ***(2)*** a second CNN trained to recognize objects in the paintings, like the one found in [1], and ***(3)*** a natural language model capable of mapping objects found in a scene by ***(2)*** to historical or mythological moments, or, perhaps, scene descriptions. The rationale behind this hypothetical model is that humans typically consider two different realms of detail when categorizing works of art to an artist or movement, ***(1)*** the stylistic details (e.g., brushstrokes, use of color, “texture”) and ***(2)*** the narrative/thematic/philosophical content of the painting. Here, ***(1)*** would be used to determine the stylistic details of the work of art, while ***(2)*** and ***(3)*** would be used to describe the “content” of the painting. The style and content outputs would then be used to predict the artist/movement.

While my results are not particularly impressive, the fact that my models were able to learn anything about art after a few hours of training on my personal laptop (with measly integrated graphics) leaves me optimistic. I believe a company like Google, which has access to extremely high resolution digital images of art and state of the art hardware, would likely be able to build a high performing art classifier. This experiment has also revealed the importance of engineering in computer vision work. At first, I was somewhat discouraged by this revelation, but then I realized that the eye and visual cortex of animals is the result of billions of years of what amounts to essentially trial and error (i.e., evolution by natural selection). An interesting study might be to “evolve” (i.e., search) for architectures, and see what such an algorithm finds. Of course, such an experiment would be computationally expensive, but, after seeing the results of NELL, I think it would be a worthwhile endeavor.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012.
[2] N. Kalchbrenner, E. Grefenstette, and P. Blunson. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics 52*, 2014.

[3] A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, 2013.
[4] S. Dieleman. Recommending music on Spotify with deep learning. http://benanne.github.io/2014/08/05/spotify-cnns.html, 2014
[5] J. R. Binder, C. F. Westbury, K. A. McKiernan, E. T. Possing, and D. A. Medler. Distinct brain systems for processing concrete and abstract concepts. *Journal of Cognitive Neuroscience*, 17:905-917, 2005.
[6] G. M. Morriss-Kay. The evolution of human artistic creativity. *Journal of Anatomy*, 216:158-176, 2010.
[7] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.