DART : Deep learning for ART

Maxime Dumonal *mdumonal@stanford.edu*
Yuze He *yuzehe@stanford.edu*
Prasad Kawthekar *pkawthek@stanford.edu*

1 Introduction

Art world places a significant value on the "brand" of artists behind art pieces [1]. This incentivizes art forgeries, wherein forgers attempt to imitate famous artists and sell the forgeries for large sums of money to museums and private art collectors. For instance, Han van Meegeren, a failed Dutch painter began forging famous painters such as 17th century artist Vermeer in early 20th century and sold forgeries worth $30 million before being caught [2]. The inability to catch art forgeries increases the risk of investment in art by art collectors, which in turn impacts preservation of art from an art history perspective.

We investigate the use of automated art forgery detection using deep supervised learning. During the training phase, the computational model learns to detect painter-specific features given digital copies of paintings and the corresponding painter labels. Then during evaluation, these models can then be used to discriminate whether two paintings are by the same painter or not. We test three different methods for this task that make use of deep convolutional neural networks for extracting painter-specific features from paintings, followed by evaluation on a reasonably sized dataset, and discussion of results and challenges therein.

2 Related Work

A wide variety of methods have been developed for visual art forgery detection. This includes physical methods such as carbon dating to detect anachronistic elements in forgeries, and X-ray diffraction techniques for pigment chemical analysis [3]. However, these methods require significant time and cost overhead and thus are difficult to scale. We refer the reader to [4] for a review of other physical methods for art forgery detection.

Recently, digital techniques have also been applied to classify digital images of paintings. Lombardi [5] used low level features of images such as texture, shadows, edges etc. to classify a small set of paintings into different styles and genres. Much work has also gone into designing features that incorporate brushwork analysis and then using these features to classify paintings into different styles [6]. Other image-level features such as SIFT have also been used for painting analysis, for instance in [7]. Our work is most closely related to the work in [10], which presented a convolutional neural network pre-trained for image categorization. It is also directly motivated by the work of [11, 12], in which the authors use a metric learning approach for painting classification. The interested reader is referred to [8, 9] for a comprehensive literature review of other similar techniques. To the best of our knowledge, few of the techniques applied for painting classification have relied on automatic featurization of paintings. Our intuition is that a deep learning method such as convolutional neural nets should be able to do this automatically and thus reduce the overhead required by developing domain specific features.

3 Dataset and Preprocessing

We obtain a supervised dataset of +80,000 paintings across a variety of genres, time periods and artists from Kaggle [13]. Most of the images of paintings in this dataset are originally taken from Visual Art Encyclopedia [14], while some others are obtained from contributions by artists to Kaggle. We select a subset from this original dataset by retaining all paintings corresponding to 100 random painters from the 1584 painters in the dataset. This subset contains 3529 total paintings, which we split into a train set of 2852 paintings (~80%), valid set of 318 paintings (~10%), and a test set of 358 paintings (~10%), such that the number of paintings for each painter in the train, valid and test set is proportional to that in the combined subset. Figure 1 shows the distribution of number of paintings for each of the 100 painters across the train

and test dataset.

The original dataset consists of paintings with varying scales and aspect ratios. We process the dataset to retain 64 x 64 patches of images, and use Lanczos filter to fit each image to this patch size [15]. For two of our methods, we also test our technique on 256 x 256 patches, which take significantly longer to process.

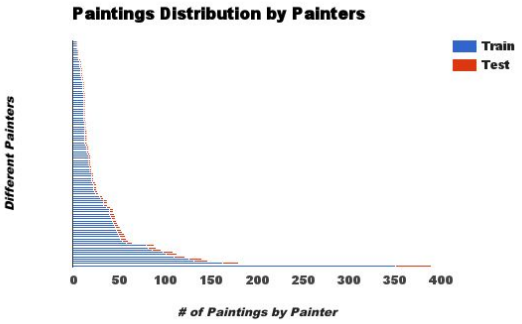

Figure 1. Painting distribution by painters in the dataset.

4 Methods

4.1 Deep CNN

Deep learning allows for a multi-layer neural network model to learn and extract high-level representations of data. A deep learning algorithm is usually composed of many layers, each layer further down in the neural network will learn a more and more abstract set of features. In the field of computer vision, deep learning models are usually combined with convoluted neural network models (CNN) to extract abstract features. A CNN model is simply a neural network model with convoluted layers wherein the neurons only have local receptive fields and consecutive convoluted layers do not have full connectivity. In each convoluted layer, there are multiple feature maps, wherein each feature map corresponds to a particular filter with a common set of parameters. Due to parameter sharing, each filter detects and learns only one feature across all spatial locations of the input images. Each convoluted layer would then undergo a nonlinear transformation in order to make the features linearly separable. In addition, CNN models usually incorporate max pooling layers by inserting max-pooling layers between convoluted layers, so that the more abstract features learned in later layers of the network are insensitive to local variations of the input images. The max pooling layers also serve to reduce the number of neurons in each subsequent layers as the number of feature maps multiplies.

For our task, we used a pretrained 16 layer CNN trained on object recognition tasks for ILSVC-14 [21]. We finetune this pretrained VGGNet by removing its fully connected classifier, attaching a fresh fully connected classifier on top, and calibrating this classifier and the last convolutional block of VGGNet on the multi-class classification task of painter detection. The original weights of VGGNet on the first 4 conv blocks can still be expected to contain valuable information as they are trained on large datasets for the ImageNet competition. The fully connected classifier consists of a two dense layers with 256 neurons each and ReLU activation in the first layer and softmax activation in the second layer. The final architecture of the VGGNet with the fully connected classifier on top is shown in Figure 2. We train this model to minimize the categorical cross entropy of classifying painters. The optimization is carried out using stochastic gradient descent with a learning rate of 0.01 (fine-tuned to promote early convergence.)

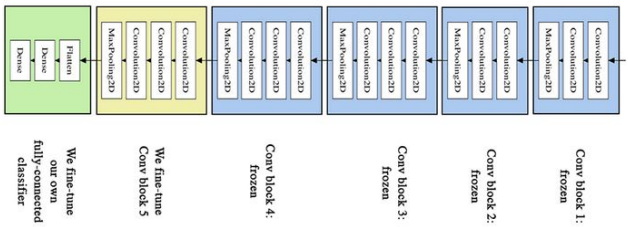

Figure 2. Fine-tuned VGGNet with fully connected classifier on top.

4.2 Deep CNN + SVM

This method utilized the CNN mentioned in Section 4.1 to extract features from the last layer of the fine-tuned VGGNet. These features serve as inputs to an SVM model which is trained for multi-class classification of predicting painters from paintings. We carried out this experiment to test the quality of features being extracted by Section 4.2, since a high performance for both SVM and a fully connected classifier using the same features as input would indicate better quality of features. We fine-tune the kernel of the SVM from amongst {rbf, polynomial, linear} and pick Linear kernel since it performs the best on our validation dataset.

4.3 Siamese Networks

The third method we implement is a siamese network due to Lecun et al. [16]. Siamese networks take a pair of inputs and directly predict whether the labels for the two inputs are the same or not. They have been used successfully for a variety of image verification tasks, such as face verification [17] and signature verification [18]. Consider a neural network that maps each input X to an embedding $G_W(X)$ using a mapping function G_W parametrized by weights W. The distance between the mappings of two inputs X_1 and X_2 can be measured as $D_W(X_1, X_2)$ using some distance metric, say Euclidean distance, as shown in the following equation-

$$D_W(\vec{X_1}, \vec{X_2}) = \|G_W(\vec{X_1}) - G_W(\vec{X_2})\|_2$$

The goal of the network is to minimize the distance $D_W(X_1, X_2)$ for input pairs X_1, X_2 that have the same label Y_1, Y_2 s.t $Y_1 = Y_2$; and to maximize the distance for input pairs for which $Y_1 != Y_2$. This can be achieved by describing a contrastive loss function of the predicted distance between the embeddings of a pair of inputs and the actual distance (which is 0 if $Y_1 = Y_2$ and 1 otherwise) as shown in the following equation-

$$L(W, Y, \vec{X_1}, \vec{X_2}) = \\ (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

The first term of the equation measures the penalty if $Y=0$ i.e if $Y_1 = Y_2$ and the second term measures the penalty if $Y=1$ i.e if $Y_1 = Y_2$. The margin m describes the minimum distance between dissimilar pairs of inputs for them not to contribute to the loss function. This loss function can then be minimized using (stochastic) gradient descent.

The final architecture of a siamese network is shown in Figure 3, and the convolutional network that we use to generate the embedding G_W is shown in Figure 4. The convolutional network consists of 3 convolutional layers with 32 filters each and a convolutional width of 3, followed by a max pooling layer with a pool width of 2, and a dropout layer with dropout probability of 0.05. This set of 3 convolutional layers, max pooling layer and a dropout layer is repeated twice. Finally a fully connected network consisting of two dense layers completes the convolutional network architecture. All layers use ReLU activation and are initialized randomly.

Hyperparameter Fine-tuning. The layer activation functions, margin for contrastive loss functions m and the total number of (3conv, 1max-pool, dropout) layer set was fine-tuned to minimize the ROC-AUC score of prediction on the test set, as described in Section 5.2.

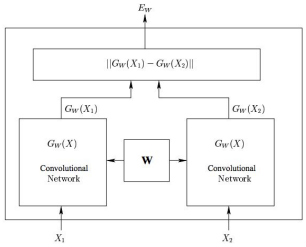

Figure 3. Siamese net architecture.

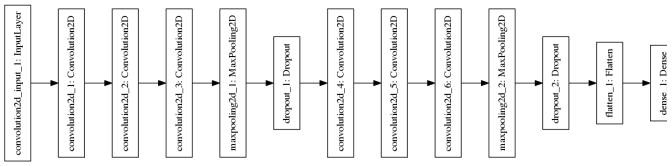

Figure 4. Convolutional network in siamese network.

5 Experiments

5.1 Experimental Setup

Method 1 and 2 (Deep CNN and Deep CNN + SVM) were trained on all paintings from the train set. Method 3 (Siamese Net) was trained on 102,400 randomly selected pairs of paintings from amongst ~8 million possible pairs of train paintings. 5000 pairs of test paintings were randomly selected from amongst the ~128,000 possible pairs of test paintings. Method 1 and 2 were used to generate class predictions for each of the paintings in these pairs and then the binary prediction of whether the painting was by same painter or not was computed using this multi-class output. Method 3 was used to directly retrieve the label of whether each test pair is by the same painter or not. The deep learning models were implemented using Keras library [20] and SVM was implemented using Scikit-Learn [22]. The experiments were carried out on AWS g2.2xlarge with an NVIDIA GPU.

5.2 Results

The train, precision, recall, accuracy, F1 and ROC AUC scores for each of the methods with different dimensions for image scaling are presented in Table 1. Due to the computational expense of running Deep CNN, it was only tested on 256 x 256 images (which were tested before testing 64 x 64 patches for all methods).

	Image Size	Test ROC AUC	Test F1	Test Accuracy	Test Precision	Test Recall
Deep CNN	256 x 256	0.547	0.671	0.547	0.526	0.925
Deep CNN + SVM	256 x 256	0.623	0.717	0.622	0.571	0.968
Deep CNN + SVM	64 x 64	0.867	0.882	0.867	0.793	0.993
Siamese Net	256 x 256	0.599	0.617	0.595	0.561	0.685
Siamese Net	64 x 64	0.800	0.833	0.800	0.714	0.995

5 Discussion

Surprisingly, for all patches, all methods work better for smaller patches than for larger patches. The reason behind this is that the computational overhead of training on larger patches is very high, so Method 1 and 3 were only trained on the large patch dataset for a few epochs which is not sufficient to extract accurate features from the images. Nonetheless, on 64 x 64 patches all methods achieve high high recall with reasonably high precision (except for Method 1).

Visualization. To investigate the performance of Method 1 further, we use the dimensionality reduction technique t-SNE to visualize the features extracted by the model [19]. The dataset we use also contains genre labels (abstract, landscape, portrait etc.) for each painting. Figure 7 corresponds to t-SNE applied to 750 paintings corresponding to Rembrandt and Picasso. Rembrandt and Picasso have different genre labels, and so unsurprisingly t-SNE is able to successfully discriminate between the two painters quite

successfully (different colors correspond to different painters). However, when we use t-SNE with 9 painters belonging to the same randomly selected genre, we obtain Figure 8, which is significantly less cleaner than Figure 7. Thus, our observation suggests that the Deep CNN is able to extract genre level features easily, but has difficulty differentiating painters from the same genre.

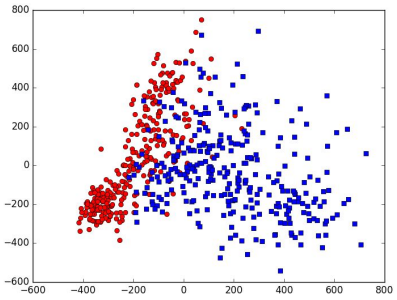

Figure 7. t-SNE applied to artists of different genre.

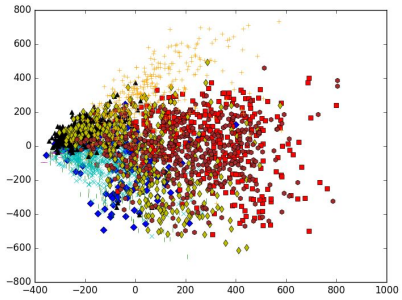

Figure 8. t-SNE applied to artists of same genre.

6 Conclusions and Future Work

We implemented three different methods for forgery detection of paintings using deep convolutional neural networks. The methods were trained on a dataset of +3000 paintings corresponding to 100 painters, and evaluated on a dataset of +350 paintings. Our results indicate that the computational overhead of this method is significant, since the methods do not converge when evaluated on larger image patches. Moreover, extracting painter specific features is difficult as shown by using t-SNE on the paintings with same genre labels. For future work, we hope to train larger models for more epochs in order to achieve better convergence and performance. Once an accurate method to distinguish painters is developed, it can be used for a variety of tasks, such as identifying and visualizing features that make some artists more skilled than others, or for style transfer of artists features to images.

Acknowledgements

We would like to thank Hao Sheng for his guidance, constructive reviews and helpful comments throughout the project. We are also grateful to the instructors and all TAs for organizing an excellent course.

References

[1] Mayyasi, A. (2015) *Why is Art Expensive?* [Web log post] Retrieved from https://priceonomics.com/why-is-art-expensive/

[2] Dolnick, E. (2008) *The Forger's Spell: A True Story of Vermeer, Nazis, and the Greatest Art Hoax of the Twentieth Century.* Harper.

[3] Ragai, J. (2013). *The Scientific Detection of Forgery in Paintings*. Proceedings of the *American Philosophical Society*, 157(2), 164.

[4] McHugh, M., DiFrancesco G., Gencarelli, J., Debenham, C. *Art Forgeries and Their Detection.* [Web log post] Retrieved from http://www.scientiareview.org/pdfs/197.pdf

[5] Lombardi, T. E. (2005). *The classification of style in fine-art paintings.* (Doctoral dissertation, Pace University).

[6] Johnson, C. Richard, Ella Hendriks, Igor J. Berezhnoy, Eugene Brevdo, Shannon M. Hughes, Ingrid Daubechies, Jia Li, Eric Postma, and James Z. Wang. *Image processing for artist identification.* IEEE Signal Processing Magazine 25, no. 4 (2008): 37-48.

[7] Khan, F. S., van de Weijer, J., & Vanrell, M. (2010). *Who painted this painting.* In 2010 CREATE Conference (pp. 329-333).

[8] D. G. Stork. (2009). *Computer vision and computer graphics analysis of paintings and drawings: An introduction to*

the literature. In Computer Analysis of Images and Patterns, pages 9–24. Springer.

[9] A. Bentkowska-Kafel and J. Coddington. (2010) *Computer Vision and Image Analysis of Art*. Proceedings of the SPIE Electronic Imaging Symposium.

[10] Y. Bar, N. Levy, and L. Wolf. *Classification of artistic styles using binarized features derived from a deep neural network*. 2014.

[11] Saleh, B., & Elgammal, A. (2015). *Large-scale Classification of Fine-Art Paintings: Learning The Right Metric on The Right Feature*. arXiv preprint arXiv:1505.00855.

[12] B. Saleh, K. Abe, and A. Elgammal. *Knowledge discovery of artistic influences: A metric learning approach*. In ICCC, 2014.

[13] Painters by Numbers. https://www.kaggle.com/c/painter-by-numbers. Kaggle competition.

[14] Visual Art Encyclopedia. https://www.wikiart.org/

[15] Lanczos, C. (1950). *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.* Los Angeles, CA: United States Governm. Press Office.

[16] Hadsell, R., Chopra, S., & LeCun, Y. (2006). *Dimensionality reduction by learning an invariant mapping*. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (Vol. 2, pp. 1735-1742). IEEE.

[17] Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). *Deepface: Closing the gap to human-level performance in face verification.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1701-1708).

[18] Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E. & Shah, R. (1993). *Signature verification using a "Siamese" time delay neural network*. International Journal of Pattern Recognition and Artificial Intelligence, 7(04), 669-688.

[19] Maaten, L. V. D., & Hinton, G. (2008). *Visualizing data using t-SNE*. Journal of Machine Learning Research, 9(Nov), 2579-2605.

[20] Keras Github Repository. https://github.com/fchollet/keras

[21] Karen Simonyan, Andrew Zisserman. (2014) *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Published as a conference paper at ICLR 2015

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.