Genetic Algorithms and Deep Learning for Automatic Painter Classification

Erez Levy
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
erezlevy@gmail.com

Omid E. David
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
mail@omiddavid.com

Nathan S. Netanyahu*
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
nathan@cs.biu.ac.il

ABSTRACT

In this paper we describe the problem of painter classification, and propose a novel hybrid approach incorporating genetic algorithms (GA) and deep restricted Boltzmann machines (RBM). Given a painting, we extract features using both generic image processing (IP) functions (e.g., fractal dimension, Fourier spectra coefficients, texture coefficients, etc.) and unsupervised deep learning (using deep RBMs). We subsequently compare several supervised learning techniques for classification using the extracted features as input. The results show that the weighted nearest neighbor (WNN) method, for which the weights are evolved using GA, outperforms both a support vector machine (SVM) classifier and a standard nearest neighbor classifier, achieving over 90% classification accuracy for the 3-painter problem (an improvement of over 10% relatively to previous results due to standard feature extraction only).

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

General Terms

Algorithms

Keywords

Genetic Algorithms, Deep Learning, Painter Classification, Restricted Boltzmann Machines, Deep Belief Network

1. INTRODUCTION

Art forgery, which dates back more than two thousand years, has played a key role in the development of painting authentication. This task has been usually performed manually by art experts who have dedicated their lives to this profession. Their expertise amounted to using various characteristics other than what the human eye can see, including chemical analysis, spectrometry, and infrared or X-ray imaging. The infamous Vermeer forgery [14] attests, perhaps, most vividly to the challenges presented by painting authentication. Han van Meegeren used historical canvasses and managed to deceive art experts into believing that his painting was an authentic Vermeer. Only after being charged with treason and sentenced to death for selling another (forged) Vermeer, did he confess and was forced to create another painting to prove himself innocent of treason. A more recent case of painting authenticity involves the Pollock paintings found a decade ago in a storage locker in Wainscott, NY. The authenticity of these paintings was compromised on the basis of computer analysis of the paintings' fractal dimension [21]. This claim was subsequently disputed by analyzing childlike drawings that supposedly have the same fractal dimension as the Pollock paintings [8].

In this paper we address the closely related problem of painting classification, i.e., the task of assigning a specific artist to a given painting (from a dataset of paintings by several artists). Note that the image authentication problem can be viewed as a binary image classification problem (i.e., determine whether or not a given painting was painted by a certain artist). Recent developments for both problem types have focused on preprocessing techniques of reducing the high dimensionality of visual data to low-dimensional representations which can be manipulated towards image understanding. This process is essentially accomplished upon the transition from image space to feature space. That is, painting classification consists of two main stages: (1) Performing feature extraction on the painting (i.e., obtaining a low-dimensional vector of feature values for each image), and (2) performing supervised classification given the feature vectors.

Levy *et al.* [13] applied feature extraction to paintings using generic image processing (IP) functions (e.g., fractal dimension, Fourier spectra coefficients, texture coefficients, etc.), followed by genetic algorithms (GA)-based learning of the weights of a weighted nearest neighbor (WNN) classifier [18]. Their approach achieved 80% accuracy for the 3-painter classification problem.

In recent years *deep learning* (DL) methods, in particular, deep *restricted Boltzmann machines* (RBM) [6] have proven highly successful in unsupervised feature extraction

*Nathan Netanyahu is also with the Gonda Brain Research Center at Bar-Ilan University, and the Center for Automation Research, University of Maryland, College Park, MD 20742 (email: nathan@cfar.umd.edu).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
Copyright 2014 ACM 978-1-4503-2662-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2576768.2598287 .

tasks, improving the state-of-the-art of numerous classification problems in image processing and computer vision [11].

In this paper we present the problem of painter classification and briefly survey recent research that has been conducted in the field. We then describe our approach which combines feature extraction due to standard image processing techniques and deep RBMs, and compare several classification methods which use the augmented set of extracted features as input. Specifically, we compare support vector machine (SVM), standard nearest neighbor (NN), and GA-based weighted nearest neighbor (WNN) classifiers. The results presented here show that the GA-based WNN classifier using features extracted by standard IP, as well as deep RBMs outperforms the other classifiers; specifically, it achieves over 90% classification accuracy, improving previous results in [13] by over 10%.

2. BACKGROUND

Image authentication is the task of determining whether or not a given painting was painted by a specific artist. The related task addressed by us, though, is image classification, i.e., the task of determining the artist of a given painting (from a certain group of artists). The input to our problem consists of painting images of the group of artists (several paintings of each artist), and our objective is to automatically classify a given painting. One of the difficulties in solving this problem is that we cannot define a certain set of rules that the painting has to conform to in order to classify it to the subgroup corresponding to the correct artist. For this reason, computer vision techniques which are capable of identifying shapes and objects in an image are not sufficiently effective for solving the problem.

Formerly there have been attempts to harness the strength of image analysis tools to classify historical art paintings into categories of artists or genres. Levy *et al.* [13] used GA-based WNN with a set of 78 prevalent image features for classifying paintings by Rembrandt, Renoir, and van Gogh, obtaining 80% classification accuracy. Herik and Postma [22] surveyed image features relevant to the historic art domain and concluded that neural network techniques combined with domain knowledge were most suitable to the task of automatic image classification. Under-drawing strokes in infrared reflectograms were analyzed by Kammerer *et al.* [9] in order to classify how and by what tools paintings are painted. Natural language processing techniques using a naive-Bayes classifier and the coefficients of a discrete cosine transform (DCT) were used by Keren [10] in order to classify local features in an image. Kroner *et al.* [12] classified drawings by using image histograms and pattern recognition methods.

The above past research focused on specific image processing features tailored for specific datasets (such as ink paintings, infrared reflectograms, or black and white sketches). This domain-specific knowledge facilitates the exploitation of various characteristics of the painting-specific domain. In the next section we present our generic approach which does not rely on any domain-specific knowledge. Specifically, the features are extracted using both generic image processing functions (not specialized for paintings) and deep RBM neural networks.

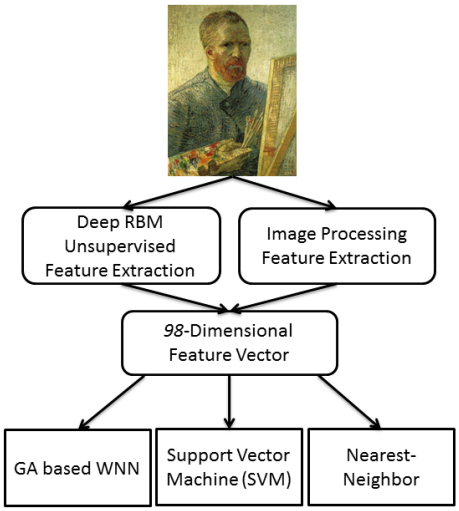

Figure 1: Basic structure of the research conducted, feature extraction techniques used and the learning methods compared.

3. FEATURE EXTRACTION

In this section we describe two feature extraction methods, using (1) generic IP functions and (2) deep RBM neural networks. The number of features extracted by these methods is 78 and 20, respectively. Namely, our module extracts a total of 98 features for each painting. Section 4 describes, in detail, the GA-based WNN module using these 98 features for painter classification.

3.1 Feature Extraction Using Image Processing Functions

We use 78 features extracted by standard IP techniques, as described in [13]. These features include also color uniqueness features, defined in terms of unique color values in the two color representations considered. We review below some of these feature types.

3.1.1 Fractal Dimension and Texture Features

Fractal dimension measures essentially the self similarity of an image, and is an index for characterizing fractal patterns or sets by quantifying their complexity as a ratio of the change in detail to the change in scale. A common approximation to the fractal dimension is the Hurst exponent [16, 17], which is related to the fractal dimension by $D = 2 - H_E$, where D is the fractal dimension and H_E is the Hurst exponent. Fractal dimension is widely used in various areas of applied mathematics (e.g., fractals and chaos theory), biophysics, hydrology, etc.

Additionally, we extract texture features, which are important when identifying objects or regions in an image. The texture of an image can be captured by a gray level co-occurrence matrix which provides the distribution of co-occurring values for a given offset. Haralick's texture features (defined in [5]) are extracted and used in our experiments.

3.1.2 Statistical Descriptors

Statistical descriptors are among the most trivial features that may be extracted in the preprocessing stage. Nonetheless, they may be highly valuable to painter classification. Specifically, we employ various standard statistical descriptors, e.g., standard deviation, skewness, kurtosis, and the minimum, maximum, mean and median values for each of the values of the RGB and HSV representations, and of the amplitudes of the Fourier transform.

Let $I(x, y)$ denote an image value (at a particular pixel) of an $N \times M$ image, for all of the image bands considered (i.e., gray-scale, as well as RGB and HSV bands). We compute the following parameters:

$$mean = \bar{I} = \frac{\sum_{x=1}^{N}\sum_{y=1}^{M} I(x,y)}{NM} \quad (1)$$

$$stddev = \sigma = \frac{\sqrt{\sum_{x=1}^{N}\sum_{y=1}^{M}(I(x,y)-\bar{I})^2}}{NM-1} \quad (2)$$

$$skewness = \frac{\sum_{x=1}^{N}\sum_{y=1}^{M}(\frac{I(x,y)-\bar{I}}{\sigma})^3}{NM} \quad (3)$$

$$kurtosis = \frac{\sum_{x=1}^{N}\sum_{y=1}^{M}(\frac{I(x,y)-\bar{I}}{\sigma})^4}{NM} - 3 \quad (4)$$

3.1.3 Steerable Filters and Color Histograms

A steerable filter is an orientation-selective convolution kernel used for image enhancement and feature extraction. It can be expressed via a linear combination of a small set of rotated versions of itself. A steerable Gaussian filter evaluates the first directional derivative of an image. The filter was implemented as outlined in [4] with successive angles separated by 15°, with an expectation that the angles contain the data of the brush strokes used by the painter while painting.

Additionally, color histograms were extracted separately for six image bands (i.e., red, green, blue, hue, saturation, value) using 256 depth sized bins. Afterwards, statistical descriptors were applied to the histogram vectors, producing feature values. RGB values were translated to HSV representation as outlined in [19].

3.1.4 Fourier Spectra

Fourier's transform is one of the most applied transformations in signal processing. Upon transforming an image matrix, the transform produces a spectral representation of frequencies that form the image. Sine waveforms of amplitudes and phases constitute an image when joined together, and can be inversely transformed to the original image. The transform and its inverse for a vector of length N are computed by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)} \quad (5)$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)} \quad (6)$$

where $\omega_N = e^{(-2\pi i)/N}$ is an N-th root of unity. Amplitudes for the transform are computed by applying an absolute operator on the transformation's matrix.

3.2 Feature Extraction Using Deep RBMs

Deep learning, in general, and deep RBMs, in particular, have proven recently rather effective in several machine learning and computer vision benchmarks [3]. The idea is to enable learning via deep networks in a manner that was not possible before (due to the learning complexity of these deep networks). The concept of using restricted Boltzmann machines and stacking them on top of each other in order to learn one layer at a time, was proved to be quite effective for unsupervised feature extraction.

We now provide a brief overview of RBMs, and describe how they are used to provide the 20 additional features (to the 78 features extracted due to standard IP functions).

3.2.1 Restricted Boltzmann Machines

An RBM uses a network of stochastic binary units arranged in two layers; a visible layer and a hidden one. Units $\mathbf{v}$ in the visible layers are fully connected via weights W to units $\mathbf{h}$ in the hidden layer. There are no connections within the visible layer or the hidden layer (the only connections are between the layers). The configuration of visible and hidden units has the energy:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in visible} a_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} v_i h_i w_{ij} \quad (7)$$

where v_i and h_j are the states of visible and hidden units i and j, w_{ij} denotes the weight between visible unit i and hidden unit j, and a_i and b_j are bias terms. The network assigns probabilities to all pairs of a visible and a hidden units via the function:

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})} \quad (8)$$

where the partition function Z is calculated by summing all possible pairs of visible and hidden vectors, i.e.,

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \quad (9)$$

Since RBMs lack connections between units within a layer, the conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$ have a convenient form:

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i w_{ij} v_i) \quad (10)$$

$$p(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_j w_{ij} h_j) \quad (11)$$

where $\sigma(x)$ is the logistic sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (12)$$

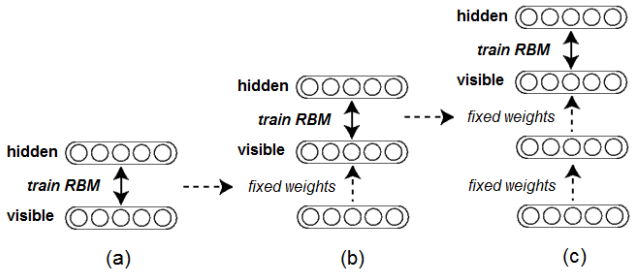

Figure 2: Composition of a 3-layer deep network subsequent to learning stacks of RBMs: (a) One layer of RBM is trained, (b) learned weights are fixed (can no longer be modified) and new RBM layer is added on top of previous one, and (c) weights of second layer are fixed, and a third RBM layer is added.

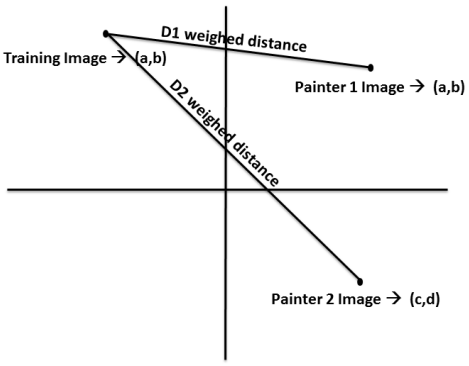

Figure 3: Simplified two-dimensional feature space, illustrating the notion of weighted distance between a (point representing a) training image and two points representing images by two painters.

The weights w_{ij} and the biases a_i and b_j can be updated during training using the so-called contrastive divergence algorithm of Hinton *et al.* [6].

3.2.2 Deep RBMs

RBMs can be stacked on top of each other to form a deep neural network (so-called *deep belief network*), as proposed by Hinton *et al.* [6]. The idea is to train, one layer at a time, a stack of RBM, after which the entire stack can be viewed as a single probabilistic model. This is achieved by first learning the initial RBM layer, then freezing the values of the weights (can no longer be modified), and finally adding another layer on top of the current one. Thus, the output values of the first layer form the input values for the next layer in the hierarchy (see Figure 2). We implemented deep RBMs using *dropouts* [7, 2, 20], which regularize the network and prevent overfitting by randomly disabling some neurons in the hidden layer during training. The deep RBM network provides 20 features in our implementation (see Figure 4).

4. PROPOSED CLASSIFICATION APPROACH

We now map the classification of the data acquired during preprocessing (i.e., feature extraction due to standard IP and deep RBMs) to GA-based learning of the weights of a WNN classifier. As mentioned previously, each painting in the dataset, which consists of $(3 \times 40 =)$ 120 images, is represented by a $(78 + 20 =)$ 98-dimensional feature vector. The 78 features (due to standard IP) consist of $(5 \times 6 =)$ 30 values of the mean, median, maximum, minimum and the number of histogram bins with non-zero frequency of each band of the RGB and HSV color representations, $(4 \times 2 =)$ 8 values of the standard deviation, skewness, kurtosis and mean of the gray-scale and FFT images, as well as the number of unique RGB color triplets, the Hurst exponent, 24-bin histogram of directional derivatives, and Haralick's texture features (in the gray-scale image).

Adding the 20 features produced by the deep RBM to the above 78 features, we obtain chromosomes of 98 real-valued weights. We then train the GA-based scheme on these weight vectors to produce the best chromosome, i.e., to evolve the best weights for the WNN classifier. The above chromosome weights are tested during validation and the overall performance of this hybrid approach is evaluated. The basic framework is as follows:

1. Produce 98-feature vectors values for each painting in the dataset.

2. Initialize vectors to be used with the nearest neighbor algorithm.

3. Train a genetic algorithm using a fitness function based on the previously produced vectors and the weighted nearest neighbor algorithm.

4. Compare the performance of the best chromosome to that of an ordinary (unweighted) nearest neighbor algorithm and an SVM classifier.

Our training is divided to $(36 \times 3 =)$ 108 training images and $(4 \times 3 =)$ 12 validation images (one tenth of the dataset, as in 10-fold cross-validation). Additionally, the image subset for training is divided to $(5 \times 3 =)$ 15 paintings used for NN classification (as explained in the next subsection), and $(31 \times 3 =)$ 93 paintings used to train the GA. We have tested a 3-way classification for paintings of Rembrandt, Renoir, and van Gough. We elaborate below on each of the above stages.

4.1 Feature Vector Extraction

As previously discussed, 78 features are extracted due to standard IP techniques. Additionally, a network of stacked RBMs is trained in an unsupervised manner using the dataset images. Running these images through the stacked RBMs produces 20 additional features for a total of 98 features per image. The structure of the deep RBM network (see Figure 4) consists of $(35 \times 35 \times 3 =)$ 3675 nodes in the visible layer. Note that each image is resized to 35×35 pixels (per each color band). The hidden layers contain 2000, 1000, 500, 250, 100, 50, and 20 nodes. Each layer was trained for 1000 epochs, during which the weights were tuned.

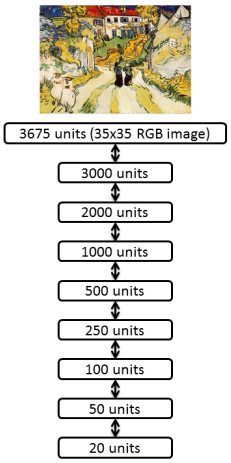

Figure 4: Structure of RBM network: Visible layer is of size $35 \times 35 \times 3$ (for RGB representation). Number of nodes in each hidden layer is indicated. Final 20-node layer provides 20 features for each image representation.

```
initialize population randomly
evaluate fitness of each chromosome
repeat
        select best-ranked individuals to reproduce
        mate pairs at random
        apply crossover operator
        apply mutation operator
        evaluate fitness of each chromosome
until enough generations have passed
```

Figure 5: Pseudo-code of genetic algorithm.

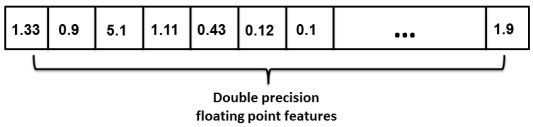

Figure 6: An example of a chromosome containing 98 feature values. Every cell contains the coefficient of its appropriate feature, and the 98-value chromosome is a solution for the classification weights applied by the weighted nearest neighbor method.

4.2 Nearest Neighbor Initialization

A portion of the training data is used for an initialization of vectors that are used by the nearest neighbor algorithm. A simple two-dimensional example is depicted in Figure 3. Suppose that the vectors, "Painter 1 Image" and "Painter 2 Image", are added to the feature space and are subsequently utilized in the fitness evaluation of the chromosomes and in the verification stage. During this process, weighted distances with respect to a given chromosome are calculated between the initialized vectors and image vectors used to train the GA.

The nearest neighbor algorithm is a special case of the general k-nearest neighbor algorithm (where $k = 1$). It is a method for classifying objects based on the closest training examples in feature space. The training stage consists of storing labeled training vectors, and the classification is accomplished by calculating which vector label is the most frequent among the k closest vectors to a feature vector in question. Based on our experiments, we chose k to be 1, using Euclidean distance as a distance metric.

4.3 Training of Genetic Algorithm

The training phase of the genetic algorithm employs the representation of the problem as a chromosome, and the definition of several evolutionary operators that are used during the genetic simulation performed by the genetic algorithm. These operators include crossover and mutation (see below). Also, the evolution is customized by fine-tuning parameters and flags such as crossover/mutation rates and elitism.

A chromosome represents double precision values that are vector components in feature space. Upon calculating the fitness for each chromosome we insert it into the nearest neighbor algorithm discussed earlier, and calculate weighted distances (according to Eq. (13)). Based on the weighted distances calculated, the nearest neighbor algorithm classifies the paintings. The classification accuracy is used as the chromosome's fitness value. The weighted distance is dependent on the chromosomes' parameters as weights, and is defined by:

$$NN_w = \sqrt{\sum_{i=1}^{d} w_i (y_i - z_i)^2} \tag{13}$$

where $Y = (y_1, y_2, ..., y_d)$ and $Z = (z_1, z_2, ...z_d)$ are painting representations in feature space, and $w_i (i = 1, .., d)$ are the weights corresponding to a given chromosome ($d = 98$ in our case). Figure 6 illustrates a chromosome structure, which consists of the various weights used by the nearest neighbor classifier.

Additional GA properties used are double-point location crossover operator, Gaussian additive mutation operator, 80% crossover occurrence, 40% mutation rate and elitism.

```
int calculate_fitness (chrom, training_data){
    fitness = 0;
    for each (image in training_data){
        weighted_vec = get_weighted_vec(img, chrom);
        if (check_nearest_neighbor(weighted_vec))
            fitness += 1;
    }
    return fitness;
}
```

Figure 7: Pseudo-code of fitness evaluation during training.

4.4 Performance Validation

Subsequent to the training phase of the genetic algorithm, validation (and cross-validation) is performed on $(4 \times 3) = 12$ images of the dataset by applying the best chromosome obtained, i.e., the one with the highest fitness value after 40 generations. Each vector (chosen for validation) is compared against all of the vectors in the dataset (stored during the initialization phase), by applying the weights of the best

chromosome to find the vector's (weighted) nearest neighbor. The vector is classified according to its (weighted) nearest neighbor label. We can thus classify all of the "validation vectors" and evaluate the performance according to the number of correctly classified vectors.

Aside from evaluating the GA-based WNN and deep RBM-based classification, we compare its performance to that of a GA-based WNN classifier without the features produced by deep learning (i.e., just the module described by Levy *et al.* [13]), and a standard (unweighted) nearest neighbor classifier. Additionally, we compare all the results to those obtained by using an SVM classifier.

5. EXPERIMENTAL RESULTS

The dataset for the conducted experiments is identical to that used by Levy *et al.* [13] in their experiments. It consists of $(3 \times 40 =)$ 120 digital reproductions of paintings by Rembrandt, Renoir, and van Gough, downloaded from the Web-museum [15]. The acquisition process is uniform across the images, ensuring that the classification is based on painter characteristics rather than artifacts of electronic devices or the digitization process. Artifacts due to JPEG compression are uniform across all images and are relatively negligible. Thus, they do not pose a problem. The images have 24-bit color depth with varying resolutions averaged approximately at 1000×1000 pixels, and compressed as JPEG formatted files. We have resampled the images and normalized them to 35×35 pixels for the deep RBM learning process. The Appendix contains the painting titles of the images used in our experiments.

We have implemented our scheme using [1] as a platform for the genetic algorithm and MATLAB as an infrastructure for calculating feature values. Additionally, deep RBM learning was implemented in Python. As described in Section 3, 98 feature values were calculated for each image. The dataset experimented with consists of $(40 \times 3) = 120$ images of paintings by Rembrandt, Renoir, and van Gough. 10 random runs were invoked, where each run was restarted with random initial chromosomes. The best chromosome over these 10 runs was chosen for a given subset of the training images. Additionally, cross-validation was conducted by shuffling (at random) the role of the images and averaging the results, i.e., images previously used for training were used for validation in a subsequent run and vice versa. This way we utilized effectively our dataset.

The experiments were conducted using three feature subsets: 78 features due to standard IP, 20 features due to deep RBMs, and the augmented set of 98 features due to both. The classification algorithms tested were a basic NN classifier, an SVM classifier, and the GA-based WNN classifier. Using only standard IP features, these classifiers achieved, respectively, 65.71%, 68.33%, and 78.33% accuracy.

The accuracy results obtained by the above classifiers using only the 20 features (from the deep RBMs) were 64.41%, 77.50%, and 73.92%, respectively. However, for the entire feature set the accuracy results were 68.71%, 71.66%, and 90.44%, respectively. Namely, the improvement obtained by the GA-based WNN classifier is substantial with respect to the 80% accuracy reported by Levy *et al.* [13]. Examining the output chromosome reveals that every feature is crucial for the painter classification, as small feature values are rarely encountered in practice. The graph in Figure 8 presents the fitness function (classification accuracy) associated with the best and average chromosomes in a single run over 40 generations. The evolutionary improvement is clearly evident.

Table 1 provides a summary of the classification performance obtained using different methods and features. IP, DL, and IP + DL stand for features obtained due to standard IP techniques, deep learning, and the combination of both, respectively.

	IP	DL	IP & DL
GA-based WNN	78.33%	73.92%	90.44%
Basic NN	65.71%	64.41%	68.71%
SVM	68.33%	77.50%	71.66%

Table 1: Classification accuracy for GA-based WNN, basic NN, and SVM, using features due to standard IP, deep learning (DL), and the combination of both (IP + DL).

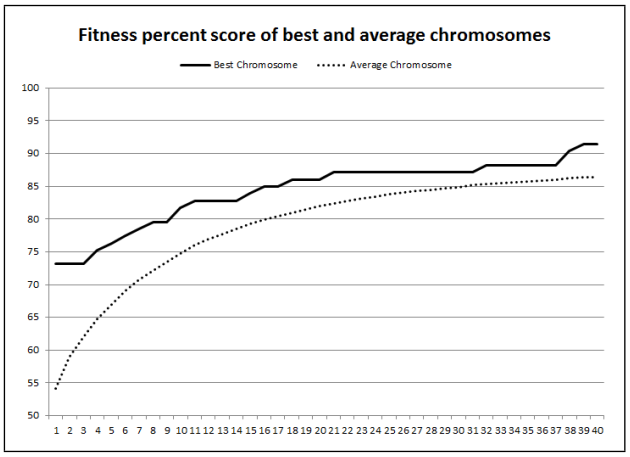

Figure 8: Accuracy scores of best and average chromosomes over 40 generations: The vertical axis represents the fitness score (classification accuracy) in each generation and the horizontal axis represents the generation number; a chromosome's fitness is defined as the classification accuracy using that chromosome's weights.

6. CONCLUSION

Automatic painter classification has gained much attention over the past decades, and much progress has been made with regards to both relevant preprocessing techniques and classification algorithms. Still, the problem of painter classification remains a complex task that requires more sophisticated techniques.

Deep learning techniques, in particular, those based on deep RBMs have gained popularity in recent years as powerful tools for unsupervised feature extraction. These methods have proven successful in improving the state-of-the-art of numerous classification tasks in computer vision..

The results presented in this paper show that deep learning methods can yield substantial improvements when integrated with standard, existing techniques. While the clas-

sification rate (using either standard IP features or deep learning features) does not exceed 80%, employing the augmented set of features due to both methods results in an enhanced performance of over 90% accuracy.

Additionally, during the supervised classification we observed that while SVM performs well on a smaller feature set, its performance deteriorates on a larger number of features (98 in our case). Also, while the basic NN classifier is consistently inferior to SVM, the performance of its weighted version improves dramatically (from 68.71% to over 90% accuracy) by evolving the feature weights due to GA. This attests to the power of genetic algorithms for parameter tuning. The hybrid IP+DL for unsupervised feature extraction and GA+NN (WNN) for supervised classification underlines the vast potential in combining several methods, resulting in a hybrid approach which substantially outperforms each method separately.

The presented hybrid approach is generic, and does not rely on any domain-specific assumptions. As such, it may be ported easily to additional classification domains.

7. REFERENCES

[1] S. Adcock. Gaul - Genetic Algorithms Utility Library. `http://gaul.sourceforge.net`, 2000.

[2] J. Ba and B. Frey. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, pages 3084–3092, 2013.

[3] L. Deng and D. Yu. *Deep learning: Methods and applications*. Now Publishers, 2014.

[4] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[5] R.M. Haralick, K. Shanmugam, and I.H. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, (6):610–621, 1973.

[6] G.E Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[7] G.E Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[8] K. Jones-Smith and H. Mathur. Fractal analysis: revisiting Pollock's drip paintings. *Nature*, 444(7119):E9–E10, 2006.

[9] P. Kammerer, M. Lettner, E. Zolda, and R. Sablatnig. Identification of drawing tools by classification of textural and boundary features of strokes. *Pattern Recognition Letters*, 28(6):710–718, 2007.

[10] D. Keren. Painter identification using local features and naive Bayes. In *Proceedings of the IEEE International Conference on Pattern Recognition*, volume 2, pages 474–477, 2002.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[12] S. Kroner and A. Lattner. Authentication of free hand drawings by pattern recognition methods. In *Proceedings of the IEEE 14th International Conference on Pattern Recognition*, volume 1, pages 462–464, 1998.

[13] E. Levy, O.E. David, and N.S. Netanyahu. Painter classification using genetic algorithms. In *IEEE Congress on Evolutionary Computation*, pages 3027–3034, 2013.

[14] D. Phillips. How do forgers deceive art critics? *The Artful Eye*. R. Gregory, J. Harris, P. Heard, and D. Rose, Eds., Oxford University Press, pages 372–388, 1995.

[15] N. Pioch. Webmuseum. `http://webmuseum.meulie.net/wm/`, 1994.

[16] J.C. Russ. Surface characterization: Fractal dimensions, Hurst coefficients, and frequency transforms. *Journal of Computer-Assisted Microscopy*, 2(3):161–183, 1990.

[17] B. Schiele and J. Crowley. Object recognition using multidimensional receptive field histograms. *In Proceedings of the European Conference on Computer Vision*, pages 610–619, 1996.

[18] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10(5):335–347, 1989.

[19] A.R. Smith. Color gamut transform pairs. In *ACM Siggraph Computer Graphics*, volume 12, pages 12–19, 1978.

[20] N. Srivastava. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

[21] R.P. Taylor, R. Guzman, T.P. Martin, G.D.R. Hall, A.P. Micolich, D. Jonas, B.C. Scannell, M.S. Fairbanks, and C.A. Marlow. Authenticating Pollock paintings using fractal geometry. *Pattern Recognition Letters*, 28(6):695–702, 2007.

[22] H.J. van den Herik and E.O. Postma. Discovering the visual signature of painters. *Future Directions for Intelligent Systems and Information Sciences*. N. Kasabov, Ed., Physica-Verlag, Heidelberg, pages 129–147, 2000.

APPENDIX

This appendix lists the $(40 \times 3) = 120$ titles of the paintings experimented with by van Gogh, Rembrandt, and Renoir.

#	van Gogh	Rembrandt	Renoir
1	bandaged-ear	abraham	apres-bain
2	berceuse	anslo	baigneuses
3	cordeville	aristotle-homer	bathers-1887
4	corridor-asylum	artemis	bathers-1918
5	cypress-star	bathing-river	bougival
6	cypresses	bathsheba	canoeist
7	flower-beds-holland	belshazzar	chocquet
8	green-vineyard	children	city
9	green-wheat-field	danae	country
10	house-ploughman	david	dancer
11	mme-trabuc	descent	durieux
12	mr-trabuc	emmaus	flowers
13	old-mill	hendrickje	gabrielle
14	old-vineyard	holy-family	girl-seated
15	olive-alpilles	jan-six	jugglers
16	olive-trees	magn-glass	lady-piano
17	orchard-bloom-poplars	meditation	laundress
18	orchard-plum-trees	mill	loge
19	poppies	music-party	lucie-berard
20	red-vineyard	nicolaes-tulp	near-lake
21	reminiscences	old-man	fournaise
22	road-menders	ostrich	horsewoman
23	roulin	potiphar	meadow
24	self-1	prodigal-son	moulin-galette
25	self-2	raising-lazarus	nini
26	self-easel	.1640	parapluies
27	self-gauguin	.1661	premiere-sortie
28	self-orsay	.1669	promenade
29	self-whitney	.night-watch	ride
30	skull-cigarette	return-prodigal-son	romain-lacaux
31	sun-cloud	ruts	sisley-wife
32	threatening-skies	samson	women
33	trees-asylum	scholar	seashore
34	trees-ivy-asylum	self-1629	seated-bather
35	village-stairs	self-1634	sewing
36	wheat-field	self-1660	sisley
37	wheat-rising-sun	slaughtered-ox	swing
38	willows	staalmeesters	terrace
39	peasant	stofells	watercan
40	woman-arles	tobias	woman-veil