ARTIST IDENTIFICATION
A COMPARISON BETWEEN ALEXNET, GOOGLENET AND RESNEXT

COGNITIVE SERVICES COURSE
COMPUTER SCIENCE MASTER DEGREE
UNIVERSITY OF PADOVA

Mattia Bottaro
Dipartimento di Matematica
Università di Padova
`mattia.bottaro@studenti.unipd.it`

Mauro Carlin
Dipartimento di Matematica
Università di Padova
`mauro.carlin@studenti.unipd.it`

July 21, 2019

ABSTRACT

In this essay we present our work for the project of the Cognitive Services course. The problem we face is the artist identification, that is the ability to recognize the author of a painting, given an image of it. Our dataset consists of about 10.000 paintings by 23 different artists. In order to resolve this task, we exploited some famous Convolutional Neural Networks (CNNs), i.e. AlexNet, GoogLeNet and ResNeXt, which come with Pytorch library. Those networks have been trained according to two distinct approaches: training them from scratch and exploiting pre-trained models using transfer learning technique. We used a virtual machine (VM) hosted by Google Cloud Platform to perform our experiments.
What we have achieved is a set of results that are in line or even better with the state-of-the-art, confirming that CNNs are suitable to solve this type of task.

***K*eywords** Convolutional Neural Network · Artist Identification · ResNeXt · GoogLeNet · AlexNet · Transfer Learning

1 Introduction

Artist identification is the task of recognizing the author of a painting given only an image of it, without any other information about it. It is still nowadays a difficult problem to solve, for the following reasons:

- there are not many contributions in the state-of-the-art, moreover the majority of them regards style identification and relies on old deep learning approaches. Indeed, only [5] is concerned with artist identification through the use of CNNs;
- the available datasets are not as large as those used by models which solve more common tasks, such as image classification (e.g. Imagenet [16]);
- an artist's style could change a lot through his works (intra-class variation), or it could be influenced by other painters (inter-class similarity). Therefore, to correctly classify a painting can be extremely complicated.

This task is usually carried out by human experts because of its difficulty, so our contribution could help to automatize cataloging of artworks, which is more and more important due to the increasing number of digital archives. In addition, our work could be useful in art forgeries detection.
Our approach involves 3 different CNNs, namely AlexNet ([13]), GoogLeNet ([12]) and ResNeXt ([11]), trained from scratch and with transfer learning in order to compare these opposite techniques. Our experiments yielded a top-1 accuracy of 82% and a top-3 accuracy of 94% as best result.

The rest of the essay is structured as follows: in section 2 we discuss similarities and differences between previous works and ours. In section 3 we explain the dataset we used, how it was created and which preprocessing and data augmentation techniques we exploited. In section4 we discuss our approach to solve the problem we have just described. In 5 we show some details regarding the *GCloud Compute Engine*'s configuration, the metrics used to evaluate the goodness of the models and the detailed results of our experiments. Finally, in section 6 we draw some conclusions and outline some possible future works which rely on our result.

2 Related Works

In this section we show some previous works related to ours. In addition, to define the references of our experiments, the following should further convince the reader of present-day difficulties in achieving good precision in artist identification.
Recently CNNs have become widespread, because of the good performance and results obtained in tasks dealing with visual recognition. Previous works not regarding CNNs use some famous approaches of features detection, such as Scale Invariant Feature Transforms (SIFT) and Histograms of oriented gradients (HOG) ([2], [4], [7], [8]), to extract some particular characteristics of the painting that prove to be useful for its classification. What these works have in common is the use of an SVM 1-vs-rest classifier, except for [8], which uses both SVM 1-vs-rest and Naïve Bayes classifier, and [7], which uses k-nearest neighbors and hierarchical clustering. However, [7] and [8] face up the problem of style recognition, which is different than artist identification, but using similar approaches.
Furthermore, some of these works perform on small or not so heterogeneous datasets, such as [4] which uses the artworks of a single museum, or [8] which focuses on only 5 artists. Instead, in our work we set our sights on recognizing 23 different artists, with 450 paintings for each of them as our dataset.
In any case, these techniques are known to be not so effective due to the fact that features characterizing an artist's style can be many and difficult to engineer, such as texture, shape, colour, tone, variety, proportion, pattern and brush strokes.

One of the results achieved in [1] shows an improvement of performance using CNNs up to 13.6 percentage points more than SIFT, HOG or similar. This work aims to detect artwork that violates copyright terms in television programs, TV series, movies or similar scenarios. The authors use different techniques to distort the images on their dataset in order to simulate potential situation that would appear on TV programs, photographs or movies. This is not suitable for our context because distortion would be harmful for artist identification.
They have experimented 3 different CNNs inspired by AlexNet and VGG, training them from scratch with a weights initialization according to a Gaussian distribution, in our work, on the contrary, we use pre-existing CNNs without modifying their architectures. [1] is one of the few works that does not use SVM as classifier, but 3 fully connected layers with softmax classifier.
Other works like [3] and [9] use a CNN to generate features, then use an SVM 1-vs-rest classifier on top of it. [5] exploit one of the most recent CNNs, i.e. ResNet ([10]). It is an ancestor of ResNeXt ([11]), one of the CNNs used in our experiment. In [5] either training from scratch and training through transfer learning are studied, as in our work.

In the latest years, the number of CNNs for visual recognition has continued to grow, as well as their contributions in the field of computer vision. Instead of hand-crafting features, CNNs are able to learn features representation of an image, making this kind of task easier and more accurate. What is missing is a performance comparison between the latter models in the field of artist identification and, more generally, in art-recognition tasks. This comparison is the purpose of our work.

3 Dataset

3.1 Overview

The first requirement to train a CNN for artist identification is a dataset, concerning paintings of different artists, which should be as large as possible. For our project we obtain a dataset from [6], which is a subset of the WikiArt Dataset. The dataset contains about 19,000 images of 23 different artists, regarding very different styles, genres and periods. There are also 3 *.csv* files that labelled all the paintings for genre, style, and artist: for our work we only used the last one. In order to obtain a balanced dataset, we decided to randomly select 450 paintings for each artist, so in the end our dataset contains of 10,350 total images.
The next step was to divide the entire dataset in training, validation and test sets. [6] suggests a possible partitioning, but we decided to create our own because it did not provide a test set, and some paintings included in the training set

are also in the validation set; this fact could create a distorted evaluation of our model during training. Therefore, we split the dataset in training, validation and test sets using a 80-10-10 division per artist, obtaining in this way a training set of 360 paintings per artist, and a validation and test sets each of 45 paintings per artist.

3.2 Preprocessing and Data augmentation

Since CNNs require a precise size for their input images, we had to modify the paintings in the dataset, considering that they vary widely in shape and size.
So, first, we took a 224×224 crop of each input images, then we zero-centered and normalized them, with mean and standard deviation calculated for each of the three channels red, green and blue (they are RGB images).
During training we also randomly horizontally flipped the image with a probability of 50%, and also took a random crop of it (not always the central pixels), for essentially two reasons:

- randomness creates variety and can avoid overfit;
- we assumed that the style of an artist is present everywhere in the painting, so taking random crops do not influence the performance of our models.

By contrast, for validation and test sets we always took a central crop, in order to obtain stable results comparable between different epochs.

4 Method

In our work we considered three different CNNs: AlexNet, GoogLeNet and ResNeXt. Every network takes as input a $3 \times 224 \times 224$ RGB image and outputs the scores for each of the 23 artists present in our dataset.
For all of our networks, we used a softmax classifier with cross-entropy loss:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{1}$$

Where L_i represent the loss for the example i in the minibatch, f the output of the network for all 23 classes, y_i the ground truth class of the example i, and j is one of the possible class. We chose this loss function because it is the most common one in multi-class classification problem.

First, we trained the networks from scratch, in order to allow them to learn features solely for artist identification.
Afterwards, starting from networks with weights pre-trained on ImageNet dataset, we trained them using transfer learning to understand if a features representation from ImageNet can be a valuable starting point for our problem.
In both approaches, we replaced the last fully-connected layer of the CNNs with a new one to calculate a score for each artist in our dataset. We used [14] as a guide to replace the fully-connected layer, and to perform transfer learning on the pre-trained networks.

4.1 AlexNet

AlexNet ([13]) was the winning entry in ILSVRC 2012. It solves the problem of image classification where the input is an image of one of 1000 different classes (e.g. cats, dogs etc.) and the output is a vector of 1000 numbers, achieving a top-5 error of 16.4%. Architecture (Figure 1) is made up of 8 layers, 5 convolutional, some of them followed by max pooling layers, and 3 fully-connected, with ReLU as activation function.
AlexNet also uses dropout ([15]) to reduce overfitting. In this strategy, a neuron is dropped from the network with a probability of 50%. When a neuron is dropped, it does not contribute to either forward or backward propagation. As a result, the learnt weight parameters are more robust and do not get overfitted easily.
We decided to include AlexNet in our work because it represent one of the most important starting point for image classification with CNN. So, it would be interesting to compare its performance on our task with more recent, complex and deep networks.

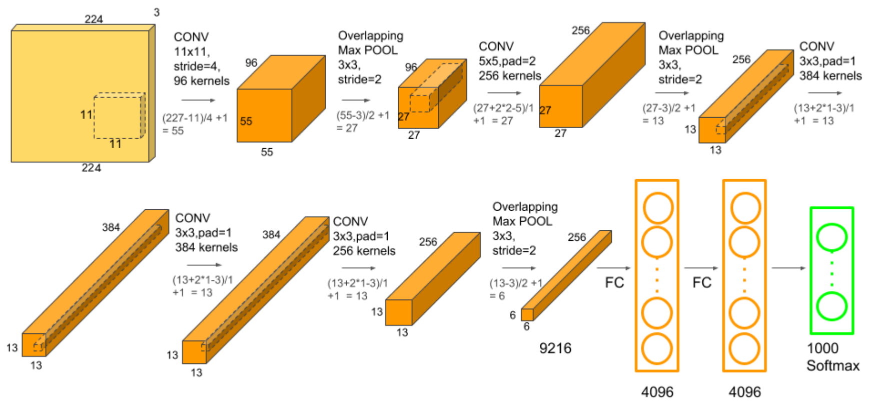


Figure 1: AlexNet's architecture

4.2 GoogLeNet

GoogLeNet ([12]) won the ILSVRC 2014 achieving 6.67% top-5 error on ImageNet. It is composed of 22 layers, with different innovations regarding the fully-connected layers, the vanishing gradient problem and a new inception module. An interesting addition to the architecture is to change the second last fully-connected layer with an average pooling layer. This layer spatially averages the feature map, converting 7x7x1024 input to 1x1x1024, reducing the computation and the number of parameters, by a factor of 49. This average pooling layer is finally followed by a normal fully-connected layer with 1000 neurons (and 1024x1000 parameters), for the 1000 ImageNet classes. During training, to address the vanishing gradient problem, special extra structures are added to the network (these are removed during testing). These are auxiliary classifiers attached to intermediate layers. All the losses from each classifier get added up, taking contribution from the auxiliary classifier lower than the main one, during training. The gradient from the main classifier that would have otherwise become very small, and thus slowing training, can in this way reach the lower initial layers, receiving gradient from the auxiliary classifiers, and thus the net gradient becomes big enough to allow training to progress.

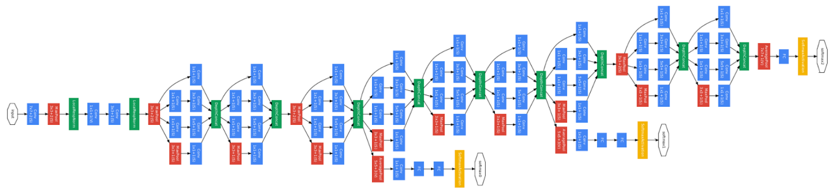


Figure 2: GoogLeNet's architecture

Inception module

The Inception modules were designed to solve the problem of computational expense, as well as overfitting, among other issues. The solution, in short, is to take multiple filter sizes within the CNN, and rather than stacking them sequentially, ordering them to operate on the same level.
The most simplified version of an inception module works by performing a convolution on an input with not one, but three different sizes of filters (1x1, 3x3, 5x5). Also, max pooling is performed. Then, the resulting outputs are concatenated and sent to the next layer. A method to reduce the number of computation is dimensionality reduction. This involves convolutions with 1x1 filters before convolutions with bigger filters (Figure 3).

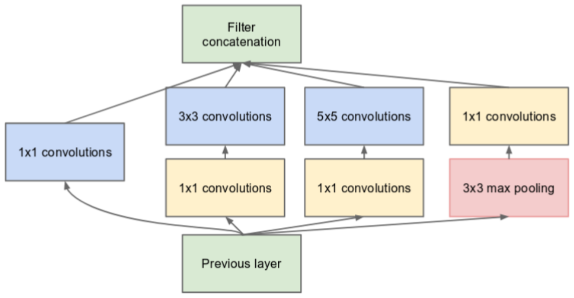


Figure 3: Inception module with dimension reduction

4.3 ResNeXt

ResNeXt ([11]) was the first runner up in ILSVRC 2016, achieving 3.03% top-5 error rate, which is a better result than human precision on the same dataset (5.1% error). The model's name, ResNeXt, contains Next. It means the next dimension, on top of the ResNet, namely the winner in ILSVRC 2015. This next dimension is called the cardinality dimension.
The architecture adopts VGG/ResNets' strategy of repeating layers, while exploiting the split-transform-merge strategy (for example inception modules have the same approach, that is, input is split into a few lower-dimensional embeddings, transformed by a set of specialized filters, and merged by concatenation). A module in this network performs a set of transformations, each on a low-dimensional embedding, whose outputs are aggregated by summation. The transformations to be aggregated are all of the same topology (Figure 4).

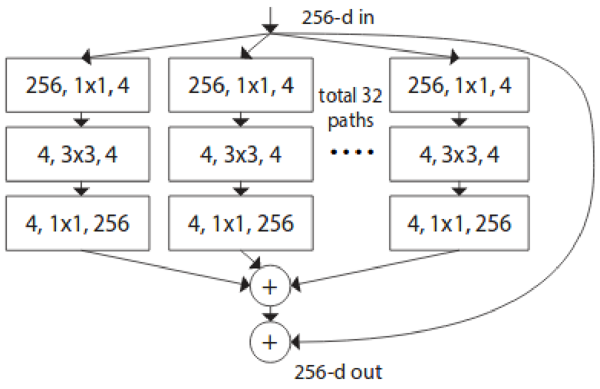
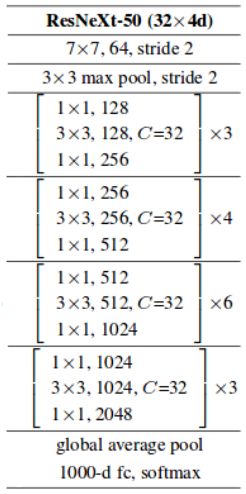


Figure 4: A block of ResNeXt with cardinality=32. A layer is shown as (# in channels, filter size, # out channels)

Experiments in [11] demonstrate that increasing cardinality (the size of the set of transformations) is a more effective way of gaining accuracy than going deeper or wider, especially when depth and width starts to give diminishing returns for existing models.
In our work we use ResNeXt-50 32×4d (Figure 5), that represents a network with four bottleneck, and each layer having cardinality of 32.



ResNeXt-50 (32×4d)
7×7, 64, stride 2
3×3 max pool, stride 2
$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{bmatrix} \times 3$
$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{bmatrix} \times 4$
$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C=32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
global average pool 1000-d fc, softmax

Figure 5: ResNeXt-50 (32×4)'s architecture

5 Experiments

5.1 Setup

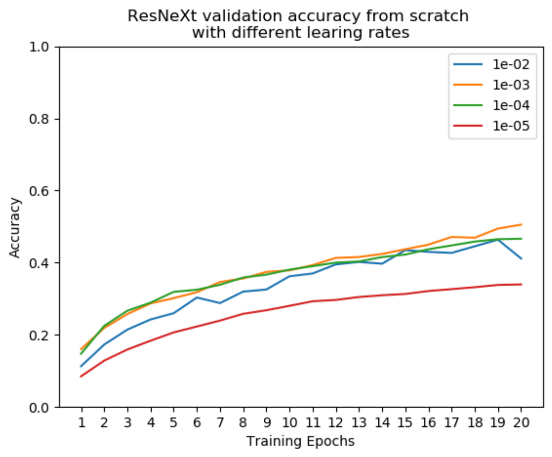
Both the built models and the experiments have been implemented in Pytorch ([17]). The reason why we chose this library is because of the presence of models, pre-trained and not, that we wanted to study.
The training of the models and the experiments were performed on a virtual machine hosted by *Compute Engine*, one of the services provided by Google Cloud Platform ([19]). Its configuration, shown in Table 1, is a pre-setting provided by the service itself and designed for deep learning applications ([18]). The dataset was stored in a *Google Cloud Storage*'s bucket, another service provided by Google Cloud Platform.

Component	Specification
OS	Debian 9
vCPU	2
RAM	13GB
GPU	1X NVIDIA Tesla K80
Disk	100GB
Disk Type	SSD
Availability Zone	us-west1-b

Table 1: VM specifications

5.2 Implementation Details

For each CNN, we tested their performance with different learning rate values ($10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$). We have observed that 10^{-3} is the value that allows to converge faster to the best results, thus all experiments we performed use this learning rate.
In Figure 6 we report ResNeXt's performance based on these different values, where it's clear that the lowest value of learning rate slows down the training phase, while higher values (10^{-2}) bring more instability in the accuracy trend.
We chose the Adam's update rule because experiments using SGD obtained worse performances. Learning rate was set to 10^{-3}, following experiments shown previously, while for the other parameters we used the default values, i.e. $\beta_1 = 0.9$ and $\beta_2 = 0.999$.
In transfer learning training, we set respectively the number of epochs and batch size to 16 and 32. For each model chosen, Pytorch makes available their final weights trained on the ImageNet dataset ([16]), which makes transfer learning very easy to implement.
We first started training the network updating the weights only in the fully-connected layers, but the results obtained were not satisfactory.
With the aim of achieving higher accuracy, as experimented in [5], we decided to allow the update of the weights in



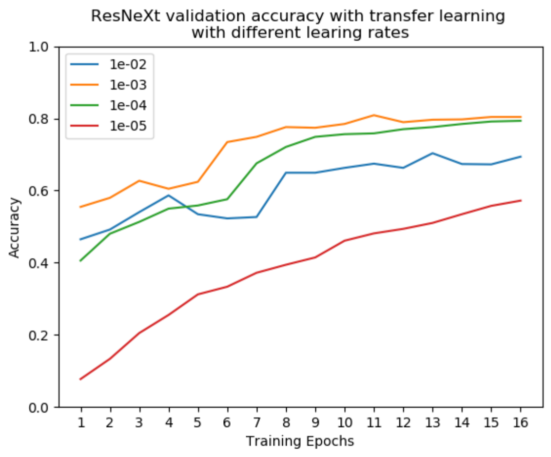


Figure 6: Top-1 accuracies for ResNeXt's validation set with different learning rates

the whole network when, for 3 consecutive epochs, the validation set's accuracy does not improve by 3 percentage points. When this condition occurs, learning rate is decreased by a factor of 10. This strong decrease is motivated by the hypothesis that pre-trained weights are already "close" to those we are looking for, so it is better to update them with a smaller step-size.

In training from scratch, the number of epochs is set to 20, while the batch size is still 32.
We experimented what in Pytorch is called *scheduler*, i.e. every 7 epochs the learning rate is decreased by a factor of 10. This strategy, however, has not led to improvements; indeed, the results exposed were obtained without the use of this technique.

In order to evaluate all the models that we study, we calculated the accuracy on training and validation sets for each epochs during training, and the final accuracy on test set using the best weights resulted from training. The accuracy is intended as top-1, i.e. the fraction of paintings whose artists are identify correctly, and top-3, i.e. the fraction of paintings whose artists are included in the 3 to whom the model has assigned the highest score.

Network	**Training from scratch**				**Training with transfer learning**			
	Top-1		**Top-3**		**Top-1**		**Top-3**	
	Train acc	Test acc	Train acc	Test acc	Train acc	Test acc	Train acc	Test acc
AlexNet	0.23103	0.27439	0.46159	0.51304	0.7134	0.68695	0.88925	0.87439
GoogLeNet	0.42669	0.46956	0.68285	0.70628	0.87536	0.80579	0.96787	0.92492
ResNeXt-50	**0.50531**	**0.50048**	**0.74275**	**0.74782**	**0.92125**	**0.82318**	**0.98695**	**0.94879**
ResNet-18 ref	0.516	0.511	0.737	0.710	0.907	0.777	0.973	0.898
1-vs-all SVM [2]	-	0.5776	-	-	-	-	-	-
1-vs-all SVM [4]	-	0.591	-	-	-	-	-	-

Table 2: Results of our experiments, compared with some previous works.

5.3 Results

In Table 2 we report the overall results obtained with training from scratch and transfer learning for each CNN, including the results obtained in [5], [2] and [4], too.

The first thing we can observe, comparing the CNNs, is that AlexNet's performances are much worse than other CNNs for all experiments, thus we can deduce how network's complexity allows to achieve better results. In fact, GoogLeNet and ResNeXt are successors of AlexNet, so this outcome should not be surprising. In particular, in training from scratch, AlexNet is 20 percentage points worse than others CNNs, while in training with transfer learning this distance is halved. This result shows that transfer learning performs very well in order to achieve quite good accuracy, even with not so complex networks. Indeed, AlexNet has obtained 87% of top-3 accuracy in the test set.
For all the networks, we can see how training with transfer learning achieved results with 40 percentage points more than training from scratch. So, we can understand that CNNs already trained on ImageNet dataset, which is wider than ours, are a valuable starting point for artist identification.

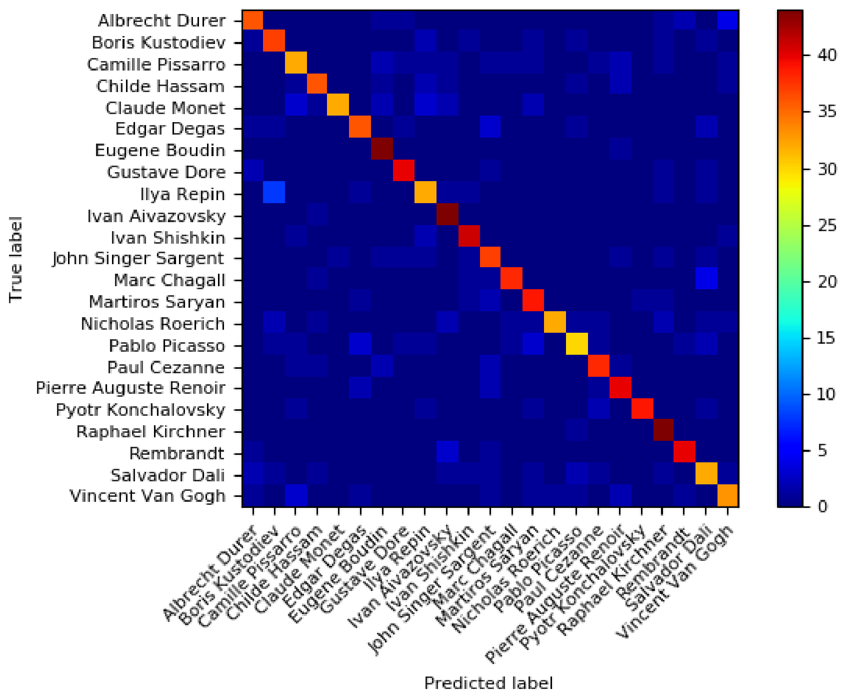


Figure 7: Confusion matrix for top-1 classification on ResNeXt test set

The most performing CNN is ResNeXt trained with transfer learning. Its top-1 accuracy reaches 82% on test set, while its top-3 accuracy is roughly 94%. This mean that it can narrow down the artist to three in the vast majority of cases.

Figure 7 shows the confusion matrix concerning our best network's classification on test set. Each row represents the true artist and each column represents the predicted one. Ideally, we want as many predictions as possible to be concentrated on the diagonal as this means that the network correctly predicted the true artist. The maximum possible value for a cell in the confusion matrix is 45, since there are 45 paintings per artist in the test set.
In general, we can easily see how the majority of artifacts are well-classified thanks to the colours (from yellow to brown) on the diagonal cell, i.e. more than 30 painting for each artist are well classified.

In particular, the artist better classified (44 out of 45) are Eugene Bodin, Ivan Aivazovsky and Raphael Kirchner. Indeed, even though these artists may belong to different artistic movements and styles, their painting tends to represent the world very close to reality.
For instance, Ivan Aivazovsky produced many portraits and landscapes, over half of all of his paintings are realistic depictions of coastal scenes and seascapes. The manner of painting real scenario or objects makes artist classification very similar to image classification. Indeed, from our results we note that artists with this characteristic obtained a higher accuracy.

As for the artists whose paintings are often misclassified, we noticed some common traits which follow.
Some of them belong to movements such as surrealism, cubism or expressionism, thus their artefacts are not depicting real scenarios, making them harder to classify. For example, Pablo Picasso, who founded the cubism movement, is the most misclassified artist (30 correctly classified out of 45).
The second reason leading to misclassification is the similarity between different artists. The most striking example regards Ilya Repin. In fact, as we can see in Figure 7 (9th row, 2nd column), the model has classified Boris Kustodiev as authors of 8 Repin's artworks. This is not so surprising since Repin was Kustodiev's teacher, influencing significantly his style.
Another reason concerns artists who, during their life, changed their art style significantly, producing very various type of paintings. For example in Figure 10 we can see 2 very different Van Gogh's artworks. This complicates furthermore the classification, leading to worse performance (accuracy on Van Gogh is 33 out of 45).

In Figure 8, for each CNN trained from scratch, we report the top-1 accuracy trends on training and validation set. What we can notice is that both trends are very close one to the other, proving a very low level of overfitting. From this fact, we deduced that with a wider dataset and an higher number of epochs, we could obtain better performances. Therefore, we decided to test once our best network for 30 epochs. We did not extend this kind of experiment to all networks due to our limited Google Cloud's resources. As we expected, ResNeXt's accuracy increased without introducing overfitting, improving both accuracies by 8 percentage points on test set.

In Figure 9, instead, we see the trends for each CNN trained with transfer learning. In all of them, after few epochs, an increase in the trend's slope is noted. This corresponds to the moment when the network is allowed to update not only the fully-connected layers' weights, but also those of all layers as previous explained in 5.2, adapting the whole CNN to artist identification task.
Moreover, we can see that GoogLeNet and ResNeXt, after roughly 10 epochs, are affected by overfitting. Indeed, there is an increasing difference between training accuracy and validation accuracy due to the stabilization of the latter, stating that these networks could have been trained for fewer epochs. However, our results achieved higher accuracies than [5], [2] and [4].

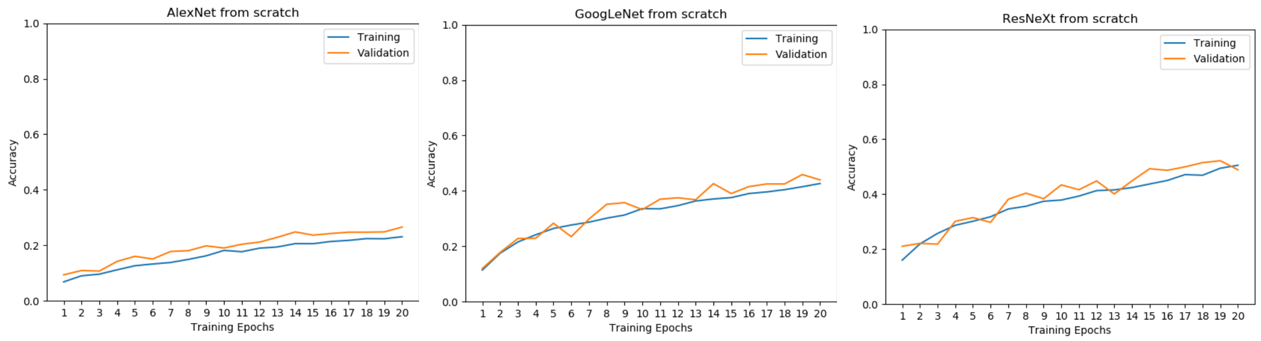


Figure 8: Top-1 accuracy for CNNs trained from scratch

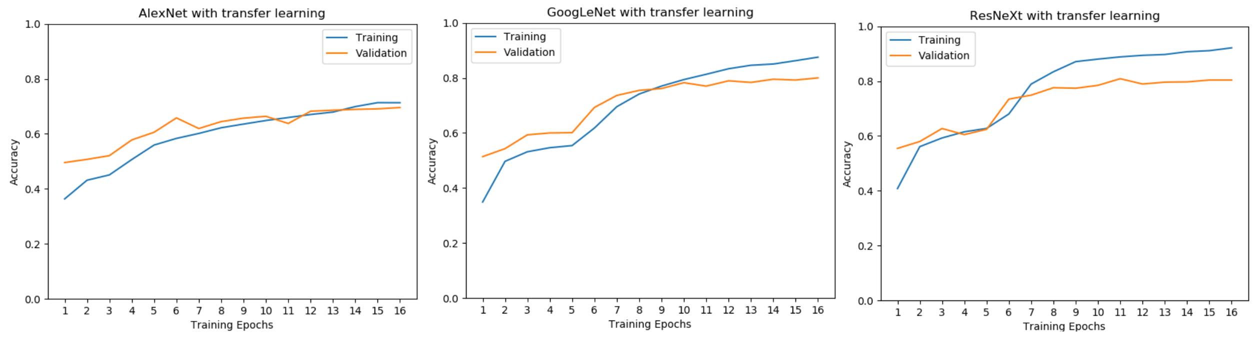


Figure 9: Top-1 accuracy for CNNs trained with transfer learning

6 Conclusions and future works

In this essay we have presented the methods and approaches we exploited to deal with artist identification. Once again we mark that it is a complex task, lacking many previous works. In our work we were able to reach a top-1 final accuracy of 82% using ResNeXt with transfer learning, pre-trained on ImageNet, outperforming traditional feature-based approaches by a significant margin, claiming that learning features exploiting the latest CNNs is more performing than hand-crafting them. With the aim of improving our work we think that a larger dataset, both in number of paintings and artists, would allow a significantly increment in accuracy, particularly in training from scratch.

In future work, we would like to deal with a task that could be considered related to this, namely the recognition of forgeries in art. Our belief is that the representation of style and content learned by our networks would be an interesting starting point to create a system able to assign a value of authenticity to a painting, because we think that, though is possible to replicate precisely an artifact, the style of an artist is somehow unique and possibly captured by a CNN.

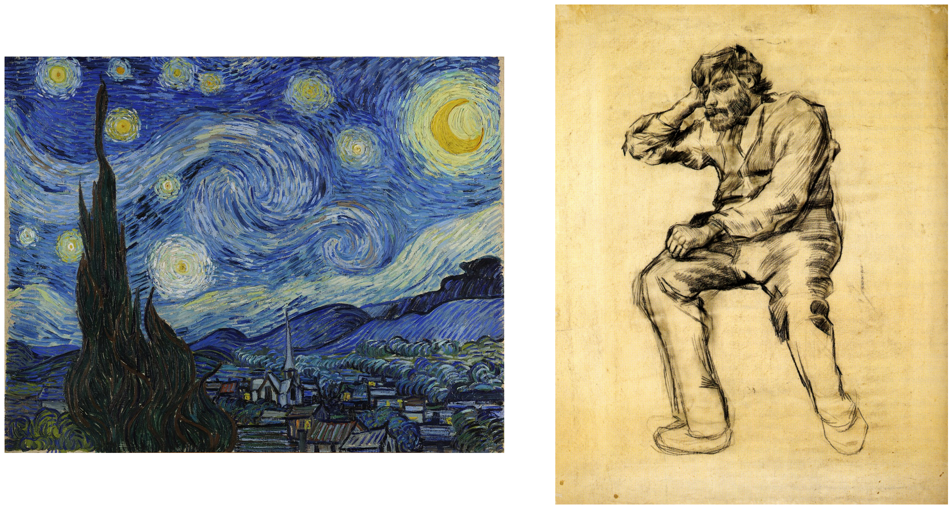


Figure 10: Two different Van Gogh's artworks

References

[1] Yiyu Hong, Jongweon Kim, Art Painting Identification using Convolutional Neural Network, International Journal of Applied Engineering Research, 2017

[2] B. Saleh and A. M. Elgammal.Large-scale classification of fine-art paintings: Learning the right metric on the right feature. CoRR, abs/1505.00855, 2015.

[3] W. L. Bar Y., Levy N. Classification of artistic styles using binarized features derived from a deep neural network. ECCV 2014.

[4] T. Mensink and J. van Gemert. The rijksmuseum challenge: Museum-centered visual recognition. 2014

[5] Nitin Viswanathan, Standford University, Artist Identification with Convolutional Neural Networks, 2017

[6] Dataset: `https://github.com/cs-chan/ArtGAN/tree/master/WikiArtDataset`

[7] T. E. Lombardi. The classification of style in fine-art painting. ETD Collection for Pace University, 2005.

[8] J. Jou and S. Agrawal. Artist identification for renaissance paintings, 2011.

[9] A. H. S. J. C. S. Sharif Razavian, A. Cnn features off-theshelf: An astounding baseline for recognition, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Microsoft Research. Deep Residual Learning for Image Recognition. 2015

[11] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, Kaiming He. UC San Diego - Facebook AI Research. Aggregated Residual Transformations for Deep Neural Networks, 2017.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going Deeper with Convolutions. 2014

[13] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. 2012

[14] Sasank Chilamkurthy. `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`

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

[16] ImageNet: `http://www.image-net.org/`

[17] Pytorch: `https://pytorch.org/`

[18] Virtual Machine pre-settings: `https://console.cloud.google.com/marketplace/details/click-to-deploy-images/deeplearning`

[19] Google Cloud Platform: `https://cloud.google.com/`