Artist identification using deep multibranch neural network

Michele Cazzola
Politecnico di Torino
s323270@studenti.polito.it

Giuseppe Arbore
Politecnico di Torino
s329535@studenti.polito.it

Abstract

This project addresses the challenge of categorization in digital paintings, by extracting distinctive features to differentiate artists. We adopted a deep learning-based multibranch neural network approach, using a Spatial Transformer Network (STN) to identify key regions within paintings, followed by feature extraction using a ResNet-like architecture. To enhance feature representation, we incorporate hand-crafted features, specifically the Histogram of Oriented Gradients (HOG), alongside neural features. This combined strategy improves the model's ability to capture intricate artistic patterns. We evaluated multiple backbone architectures employed as Spatial Transformer Network, including ResNet, MobileNet, and RegNet, to determine their effectiveness in artist classification. In addition, an ablation study is conducted to assess the impact of individual components of the model on performance. The results highlight the advantages of combining deep learning with hand-crafted features for deeper networks, whereas they prove to be harmful when using lighter ones. The code of our project is available at `https://github.com/MicheleCazzola/artist-identification`.

1. Introduction

Our project deals with the categorization of paintings, by automatically predicting the corresponding artist, given only the digital painting itself. This research is motivated by the "Artist identification" competition of Kaggle [1], which has exactly the same goal as previously described. The main challenge of this kind of task is to build a model capable to capture intra-class and inter-class feature variations: indeed the same art subject, activity, or pose can be represented in many different ways across different artists, whereas the same artist can evolve itself during its life, realizing paintings with several differences.

1.1. Related work

ConvNets. Many architectures have been proposed to solve image classification tasks: AlexNet [12] represents the beginning of CNN research, followed by better and deeper architectures, known to be good feature extractors [18, 17]. ResNet [7] introduced skip connections, leading to the design of deeper architectures and improving the performance on ImageNet dataset [6]. In the next years, several architectures have been designed with the focus on optimizing both model performance and latency: MobileNet [9, 16, 8] has a light-weight design, to have low latency also on CPUs; RegNet [15] is realized using a novel paradigm and obtains comparable (or even better) results than deep versions of ResNet, while using fewer parameters and having lower latency.

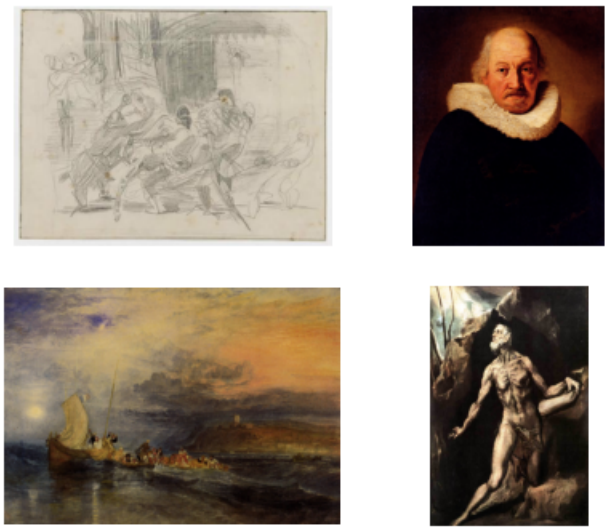

Figure 1. Sample images from the dataset: it is possible to observe their variability in size and shape.

Hand-crafted features. Hand-crafted features were widely used in earlier approaches, such as [4] and [10], since they are effective feature extractors. Recent works rely instead on convolutional neural networks and use fully connected layers as classifiers. Among earlier methods, the Histogram of Oriented Gradients (HOG) has been widely used since [5], especially in object detection tasks. Its purpose is to describe the appearance of objects in images using the distri-

bution of intensity gradients, dividing the image into regular cells, and computing the histograms inside each one. The cells are then grouped into blocks and normalized, ensuring better invariance to changes in illumination and shadowing. The result is a *feature vector*, which encodes local features and is invariant to geometric and photometric transformations.

1.2. Our approach

The main goal of our project is to train a model capable of correctly predicting the artist from the digital representation of a painting. Currently, automatic categorization of paintings is gaining increasing attention due to the greater amount of data available and the progressive digital transition that museums and art institutes are making, for both cataloging and interactive purposes [14, 10].

Our approach follows what was done by [2]: we used a multibranch deep neural network, based on residual blocks. Moreover, we also integrated the usage of hand-crafted features to better capture the salient patterns in paintings. Since only a high-level description of the network is provided, we implemented and trained our model from scratch.

2. Data

The dataset used in this project is collected from the competition [1], and is derived from a refined version of WikiArt, described in [19]. We performed some pre-processing operations, saving the result at `https://github.com/MicheleCazzola/artist-identification-dataset`.

2.1. Dataset insights

The dataset contains 25,180 RGB images that depict paintings created by 161 artists. It is divided into a training set and a test set: the first contains 21,220 labeled samples, whereas the second has 3960 unlabeled paintings and it is used only for evaluation purposes in the context of the competition.

The collection spans a wide range of artists and historical periods. The images vary significantly in both dimensions and quality, going from 70K to 44M pixels. The class distribution among the 161 artists is uneven: some artists contribute more than 400 paintings, while others are represented by fewer than 100, as illustrated in Figure 2.

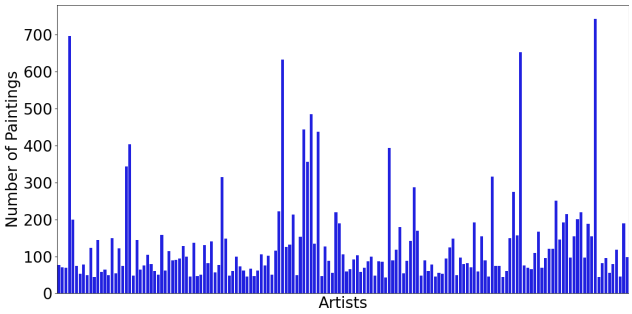

Figure 2. Distribution of paintings across the 161 artists in the dataset: the number of samples has wide variations across different artists.

2.2. Data handling and preprocessing

Dataset preparation. We moved the official dataset to our Github repository to facilitate access when using Google Colab. We divided labeled and unlabeled datasets into two different directories; paintings from the first portion are then grouped into several folders, one for each author. Moreover, we split the labeled dataset in a static way, taking 70% of it for training, 15% for validation, and 15% for testing, saving the image names into plain text files; this operation was performed with label stratification, to ensure an equal distribution of samples among artists in each split.

Dataset downsampling. We implemented functionalities to downsample the dataset at load time. This approach allowed us to test the pipeline without dealing with the entire dataset, ensuring that the main operations worked as expected before moving on to training with the full dataset.

Data preprocessing. Before passing through the model, each image is downsampled and cropped to a size of 512x512; then, input normalization is applied, with respect to the mean and variance per channel, computed in the training set. The goal is to feed the model with homogeneous samples, to account for variations in shape and feature scales within the dataset. Data augmentation is applied in this pipeline, before normalization.

2.3. Data augmentation

To cope with the small size of the dataset, with respect to the number of classes, we used some data augmentation techniques:

1. *Color jitter*: random variation of the brightness, contrast and saturation of the image.
2. *Lighting noise*: adds PCA-based lighting noise to the image; it is based on the eigenvalues of the RGB pixel distribution in the image, as introduced by [12].
3. Gaussian blur: application of a Gaussian kernel on the image to make it blurred.
4. Geometric transformation: random change in the scale and aspect ratio of the image.

These techniques have also been applied by [2] with the goal of making the model more robust to variations between the distribution of training and evaluation samples, enhancing its generalization capability.

3. Method

We implemented the approach described by [2], with the goal of processing different versions of the same image in parallel, to produce different low-level features. Then, they are fed to a final deep residual branch, for further extraction and classification, together with hand-crafted features. As hand-crafted features, we used Histogram of Oriented Gradients (HOG), to detect local patterns and capture salient information in patches of the images.

3.1. Model

The model is based on a multibranch structure, as done by [2] and shown in Figure 3. The main components are the following:

1. *ROI proposal* (3.1.1): it is implemented using a Spatial Transformer Network (STN) or by downsampling the input image.
2. *Multibranch neural network* (3.1.2: it is composed of three branches, each working on an independent version of the image. The outputs are then joined and fed to a deeper residual branch. The final classification layer takes both neural and hand-crafted features.
3. *Hand-crafted features* (3.1.3): we used Histogram of Oriented Gradients to detect local patterns in the image, as previously described.

3.1.1 ROI proposal

The selection of the three images to process independently to one another is carried out with a Spatial Transformer Network or by downsampling the input image. In both cases, the result is cropped to a size of 224x224, as needed when using a ResNet-like architecture [7], such as this one.

Downsampling. This method is used to allow the model to focus on coarse features, as a low-resolution image is provided. Here, it is used only in the third branch.

Spatial Transformer Network. This method is used to transform the input image, using only translation and scaling. Since the optimal transformations are not known beforehand, it is beneficial to embed the transformation parameters inside the model: due to the differentiability of traslation and scaling operations, it is possible to include them in backpropagation and optimize these parameters during training. A Spatial Transformer Network is composed of three stages:

1. *Localization network*: extracts the transformation parameters from the input image; here, it is implemented with a convolutional neural network (ResNet18 pretrained on ImageNet in the baseline version [2]) with the classification layer adapted to learn the parameters $\theta = f_{loc}(x)$, where x is a sample image.
2. *Parametrized Sampling Grid*: takes the learned parameters θ and produces a sampling grid of a given size, 224x224 in our case.
3. *Bilinear Sampler*: takes the sample image and the grid from the previous steps and produces a transformed image, using bilinear interpolation to perform the spatial warping.

Therefore, this module transforms an image using translation and rescaling in a completely differentiable way: this allows us to train it jointly with the rest of the network.

3.1.2 Multibranch neural network

The multibranch structure takes three images from the ROI proposal and processes them independently. The general description of the layers is given in Table 1.

Residual block. The residual block used in the network is a variant of the bottleneck block described by [7]: it allows to design deeper networks, due to the higher throughput of information, while maintaining a lower computational complexity and memory usage with respect to the basic block. The variant designed in [2] and shown in Figure 4 has the final *BatchNorm + ReLU* moved after the skip connection, since it has been shown to perform better in their experiments: due to task similarity and since we are using the same network structure, we decided to implement our network with the same type of residual block.

Branch structure. Each branch is designed to process an image (or feature map) independently: first, it is downsampled using *Conv7 + BatchNorm + ReLU + MaxPool*, where *Conv7* is a convolutional layer using a 7x7 kernel, to obtain a 112x112x64 feature map, starting from a 224x224 RGB image; then it is further downsampled using three residual blocks, to obtain a 56x56x256 feature map. This pattern is the same used in residual networks, with an initial aggressive downsampling, coupled with a huge increase of channels.

Branch fusion. The feature maps coming from the three branches are fused: first, they are concatenated on the channel dimension, to produce a 56x56x768 feature map; then, it is passed through a Join Residual Block, designed as in Figure 4, with output channels three times fewer than input ones. Therefore, it is necessary to linearly project the skip connection in the new space, using a convolutional layer, as described by [7].

Final branch. The final branch is a deep concatenation of 13 residual blocks. They are organized in three stages, each one with a first block that performs a strided convolution, to halve the feature map size and double the channel depth, followed by two, five and three residual blocks respectively, which do not perform any modification on the feature map dimensionality. The result is a 7x7x2048 fea-

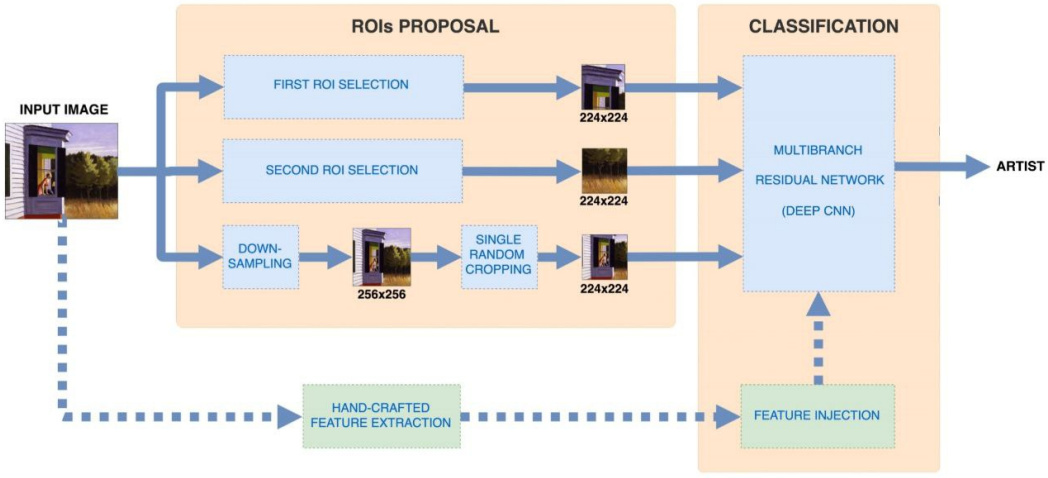

Figure 3. Multibranch residual network.

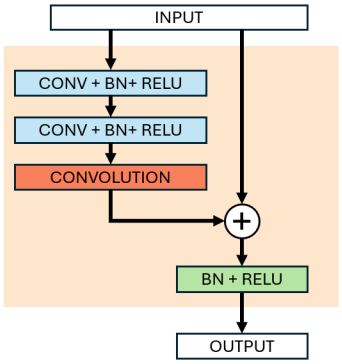

Figure 4. The residual block used in the network.

ture map, which is downsampled using global average pooling, obtaining a 1x1x2048 feature map.

Classification. The result of the feature extraction is flattened and used in a fully connected layer to perform the final classification, together with the hand-crafted features, that are concatenated to it.

3.1.3 Hand-crafted features

As mentioned previously, Histogram of Oriented Gradients is used; the goal is to detect local patterns that could be relevant for discriminating among artists. Since the network is the same and the datasets are similar, we applied HOG with the same setting as [2]:

- 81-dimensional HOG is computed, dividing each image in a 3x3 grid and computing 9 histograms with 9 bins each.
- Before HOG application, the image is converted in grayscale using the following formula: $L = 0.299R + 0.587G + 0.114B$, where L is the converted image and R, G, B are red, green, and blue channels, respectively.
- All features are l^2-normalized.

Before HOG application, the image is downsampled to 256x256, then cropped to 224x224; the goal is to reduce the computational complexity of the problem, since HOG works on CPU, processing one image at a time.

Output Size	**Layers**		
	Branch 1	**Branch 2**	**Branch 3**
112x112x64	Conv7 BatchNorm ReLU MaxPool	Conv7 BatchNorm ReLU MaxPool	Conv7 BatchNorm ReLU MaxPool
56x56x256	3× ResBlock	3× ResBlock	3× ResBlock
56x56x768	Concatenation (channel dimension)		
56x56x256	Join ResBlock, stride 1		
28x28x512	ResBlock, stride 2 2× ResBlock		
14x14x1024	ResBlock, stride 2 5× ResBlock		
7x7x2048	ResBlock, stride 2 3× ResBlock		
1x1x2048	AvgPool		
Num. classes	FC-161		

Table 1. Architecture description

3.2. Loss

We used the cross-entropy loss, which is particularly suitable for classification tasks as it measures the error between the predicted probability distribution and the ground-truth labels. Mathematically, it can be expressed as:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \quad (1)$$

where C is the number of classes, y_i is the ground-truth

(one-hot encoded), and $\hat{y}_i$ is the predicted probability for class i. By minimizing this loss, the model optimizes its ability to distinguish between classes effectively.

3.3. Training process

The training process is carried out in the same way for several configurations, related to hyperparameters and architecture variations.

Since we perform a mini-batch gradient descent, back-propagation and parameters optimization are applied every time a mini-batch is passed through the model. For each forward pass, predictions are used to compute the cross-entropy loss on both training and validation splits, as defined by the expression (1), and the evaluation metrics (3.5) on the validation set. Finally, we compute the loss of the best model (with respect to epochs) and the metrics on the test set.

The Spatial Transformer Network is a pretrained convolutional neural network (a ResNet18 in the base version of our project): due to the difference between ImageNet and our dataset images (the former contains real-world objects, whereas the latter contains paintings), we decided to fine-tune the STN, without freezing any layer.

3.3.1 Hyperparameters

The model is trained for a fixed number of epochs, determined empirically based on learning curves and execution time. Hyperparameters are tuned by training a few epochs at a time, to identify the optimal configuration: significant effort have been dedicated to the tune learning rate, weight decay and learning rate scheduler parameters.

3.3.2 Augmentations

We employed an online data augmentation approach, applying random transformations at load time exclusively to the training set. This virtually increases the size of the training set and enables the model to see a different randomized version of every image at each epoch, improving its generalization capability. Each data augmentation is applied independently to one another, with its own application probability: therefore, it is theoretically possible to configure any subset of applicable augmentations.

3.3.3 Miscellaneous

Due to the limited computational budget available, we split the training process into multiple parts; to ensure correct execution, we saved and resumed model and optimizer checkpoints. Moreover, we manually set random seeds and generators, to guarantee reproducibility of the problem, particularly useful when tuning parameters and advancing few epochs at a time.

3.4. Challenges

The implementation of the proposed approach presented several challenges. First, the entire system was developed from scratch, without using pretrained weights or external code. This has required manual implementation of each component, including a custom learning rate scheduler. Additionally, the limited size of the dataset led to early overfitting, even when employing various data augmentation techniques to increase the diversity of training samples. Another challenge was represented by the slow training process and high latency of the models. This issue was further amplified by the model's complexity, which, even without integrating a Spatial Transformer Network, reached a depth comparable to ResNet18, resulting in substantial computational demands and prolonged training time.

3.5. Evaluation process

Evaluation metrics. We used several metrics to evaluate our models:

1. *Top-1 accuracy*: fraction of correctly classified samples.
2. *Top-5 accuracy*: fraction of samples whose ground truth is in the first five positions of their predicted ranking.
3. *Mean Class Accuracy (MCA)*: average of top-1 accuracy, as measured independently for each class; it allows to consider the performance on under-represented classes.
4. *Confusion matrix*: fraction of samples predicted for class *p* and belonging to class *c*, for each pair (p, c).
5. *Weighted Top-5 MCA (WT5-MCA)*: it has been defined by the competition organizers and works as follows:
   (a) For each painting, the top-5 predictions are computed: each prediction carries a score equal to $1/k$, if the ground-truth label is the $k - th$ class predicted, otherwise it is zero.
   (b) Paintings are grouped per artist, and the average score is computed.
   (c) MCA is computed on the average scores from the previous point.

Application. The evaluation process is applied twice. At the end of each epoch, the weighted top-5 MCA is computed on the validation split. The best model identified during this phase is then evaluated on the labeled test split, where all metrics previously defined are computed.

Submission. Finally, we submitted the best performing model to the competition, in order to virtually rank us among other participants. This process involved computing the top-5 predictions on the unlabeled test set provided

to us and then submitting the corresponding file to the competition.

4. Experiments

We carried out several experiments to find the best working configuration with the baseline model, composed of a ResNet18 in the Spatial Transformer Network and using Histogram of Oriented Gradients. We also considered the usage of different convolutional neural networks instead of ResNet18: MobileNetV3-Small [8], a lightweight model, and RegNetX-400MF [15], a more recent network that combines higher quality and lower latency. The results of this part are described at 4.1.

Then, we performed an ablation study (4.2) to assess the contribution of the different parts of the model to its performance.

Finally, we submitted our best model to the competition (4.3), to evaluate our work in a real setting.

In the experiments performed, we used AdamW as optimizer [13], both due to empirical results and because it represents a good trade-off between the faster convergence of Adam [11] and the better generalization capability of SGD. We set the augmentation probability to 0.5, for all the transformations described in 2.3. We used a custom scheduler to decrease the learning rate following a multi-step trend, by a possibly different scaling factor each time. All models are trained for 30 epochs and with batch size 16, due to the memory limitations of the GPU used (15 GB); the results shown refer to the performance of the best model across the epochs, as measured on the test split.

4.1. Results

We trained three models, one for each localization network considered: as expected, ResNet is the slowest model; MobileNetV3-S is faster but loses about 1–2% in performance, depending on the metric; RegNetX-400MF achieves the best results across all metrics, reaching a weighted top-5 MCA of 43.84% at epoch 29. It is also slightly faster than ResNet, which it outperforms by 1–2%, depending on the metric. These results are summarized in Table 2. HOG features are used in all cases.

Learning curves and accuracy trends are shown in Figure 5. It is possible to see how these models have similar trends, beginning to overfit around epochs 25–30.

The confusion matrix of the best model (RegNetX-400MF) is shown in figure 6: it is possible to see the presence of a good prediction trend, where the diagonal presents more occurrences than non-diagonal elements.

These three experiments have been carried out using a weight decay of 10^{-4} and a learning rate of 10^{-4}, decayed with a multi-step scheduling policy. The hyperparameters have been empirically chosen after careful tuning. More details can be found in the Github repository of the project.

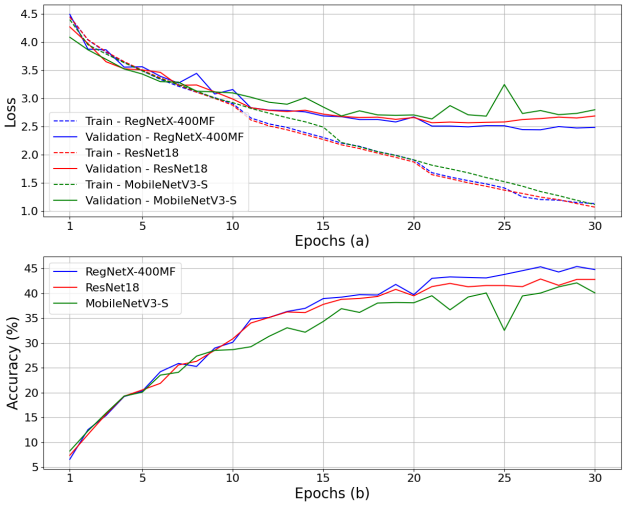

Figure 5. Comparison of different STN-based models: (a) Training (dashed) and validation losses, (b) accuracy on validation set.

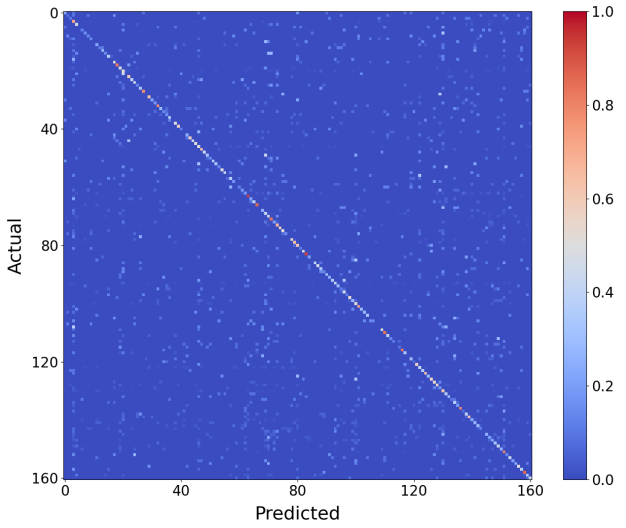

Figure 6. Confusion matrix of RegNetX-400MF, trained with HOG features.

4.2. Ablation study

Our ablation study focuses on two aspects of the model architecture:

- Using a random crop region proposal instead of a learnable STN (4.2.1).
- Removing HOG feature injection (4.2.2).

4.2.1 Ablation on ROI proposal

We replaced the learnable STN with a random crop. We expected the quality to decrease, due to the lack of data guidance for focusing on details. We trained this model for 30 epochs, still using HOG features in this part.

We tuned again learning rate and weight decay, finding

STN	acc@1 (%)	acc@5 (%)	MCA (%)	WT5-MCA (%)	Training time (min)
ResNet18	41.66	66.16	32.83	42.31	330
MobileNetV3-S	40.00	65.88	30.87	41.07	**257**
RegNetX-400MF	**42.69**	**68.20**	**33.69**	**43.84**	324

Table 2. Comparison among different models, trained with HOG features: RegNetX-400MF achieves best results.

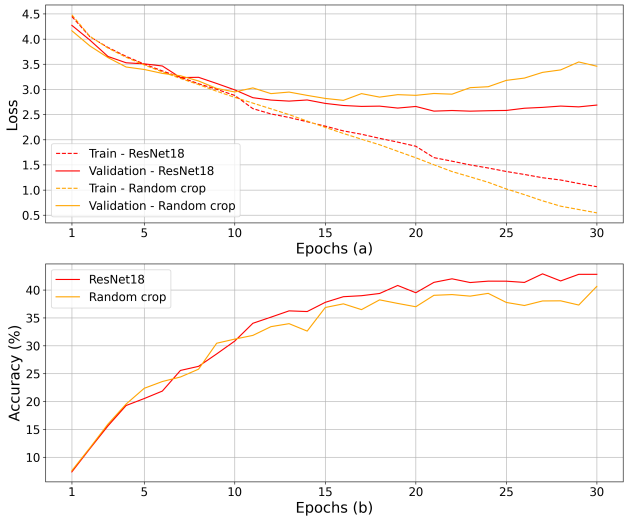

Figure 7. Comparison between ResNet18-STN and random ROI proposal: (a) Training (dashed) and validation losses, (b) accuracy on validation set.

the same optimal values as 4.1, with minor variations of scheduler parameters.

As shown in Table 3, where the comparison is limited to ResNet18 for simplicity, removing the learnable STN leads to an expected decrease in model quality (by 4–6%, depending on the metric), while reducing training time by approximately 10%. Specifically, this model achieves only a 38% on weighted top-5 MCA, at epoch 30. The comparison between learning curves and accuracy is shown in Figure 7.

4.2.2 Ablation on hand-crafted features

We removed HOG feature injection, while maintaining the learnable STN. We trained three different models for 30 epochs, as done in 4.1; we tuned the learning rate and weight decay, finding the same optimal values as in 4.1, with minor variations in the scheduler parameters.

The results show an expected performance decrease for ResNet18, with a gap of less than 1%. However, as much as networks become lighter, this trend reverses: the RegNetX-400MF version without HOG performs comparably or slightly better than the one with HOG; in case of MobileNetV3-S, removing HOG leads to a 5–6% improvement across metrics. This unexpected behavior suggests that HOG features may be effective primarily in deeper networks, such as ResNet18.

For completeness, we also trained a model without HOG

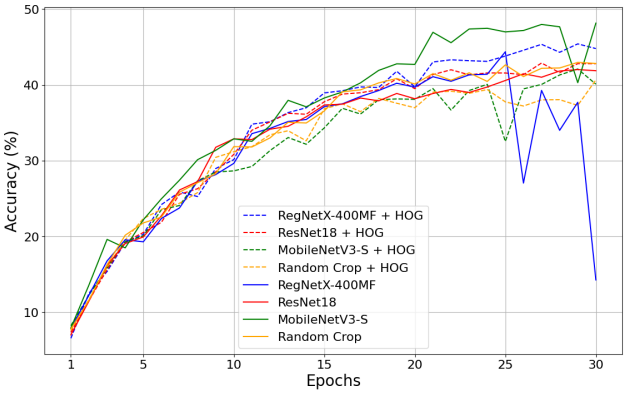

Figure 8. Accuracy comparison among models, with or without HOG.

feature injection or the STN backbone for 30 epochs. This experiment aligns with the work described by [3]. Once again, we observed that removing HOG led to a 2–4%, improvement in model performance, depending on the metric.

Furthermore, removing HOG feature injection also leads to faster training, with speed improvements ranging from 5% to 15%, depending on the model. This is expected, as computing HOG requires processing each image without GPU parallelization, resulting in slower operations.

These results are compared in Table 4, and the accuracy trends are shown in Figure 8. More details can be found in the Github repository of the project.

4.3. Kaggle submission

We submitted our best model (MobileNetV3-S without HOG) and obtained a score of 43.6%, which represents the weighted top-5 MCA on the official unlabeled test set.

This result virtually places us in fourth place of the challenge: we can consider it a good result, since we used only 70% of labeled data for training, due to the necessity of having a labeled test split, which we used to compute model performance.

5. Conclusion

We implemented a multibranch deep convolutional neural network to address a painting classification problem, with the main goal of studying the impact of a multibranch architecture on feature extraction. We combined this approach with hand-crafted feature injection, to detect local features invariant to geometric transformations.

We faced challenges related to implementing and training a deep CNN from scratch, as well as the limited size

STN	acc@1 (%)	acc@5 (%)	MCA (%)	WT5-MCA (%)	Training time (min)
ResNet18	**41.66**	**66.16**	**32.83**	**42.31**	330
Random crop	36.00	60.79	28.80	38.00	**290**

Table 3. Comparison between ResNet18-STN and random crop. Both are trained with HOG features.

Backbone Model	acc@1 (%)	acc@5 (%)	MCA (%)	WT5-MCA (%)	Training time (min)
ResNet18	41.25	65.69	31.66	41.35	292
ResNet18 + HOG	41.66	66.16	32.83	42.31	330
MobileNetV3-S	**45.39**	**70.90**	**36.99**	**46.97**	**241**
MobileNetV3-S + HOG	40.00	65.88	30.87	41.07	257
RegNetX-400MF	43.50	69.00	33.46	43.75	276
RegNetX-400MF + HOG	42.69	68.20	33.69	43.84	324
Random crop	40.59	64.78	32.11	40.93	244
Random crop + HOG	36.00	60.79	28.80	38.00	290

Table 4. Comparison among different models with or without HOG features. In bold, the best performing model overall.

of the dataset and the high latency of the models. Despite issues, we achieved a peak performance exceeding 45%, in terms of weighted top-5 MCA.

Further research in this field can be pursued, by exploring more advanced models to be used as Spatial Transformer Network, scaling up the number of branches, or using alternative hand-crafted feature extractors.

References

[1] BYU AI Association. Artist identification. `https://kaggle.com/competitions/artist-identification`, 2024. Kaggle.

[2] Simone Bianco, Davide Mazzini, Paolo Napoletano, and Raimondo Schettini. Multitask painting categorization by deep multibranch neural network, 2018.

[3] Simone Bianco, Davide Mazzini, and Raimondo Schettini. Deep multibranch neural network for painting categorization. In Sebastiano Battiato, Giovanni Gallo, Raimondo Schettini, and Filippo Stanco, editors, *Image Analysis and Processing - ICIAP 2017*, pages 414–423, Cham, 2017. Springer International Publishing.

[4] Gustavo Carneiro, Nuno Pinho da Silva, Alessio Del Bue, and João Paulo Costeira. Artistic image classification: An analysis on the printart database. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 143–157, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[10] Fahad Khan, Shida Beigpour, Joost Weijer, and Michael Felsberg. Painting-91: A large scale database for computational painting categorization. *Machine Vision and Applications*, 25:1385–1397, 08 2014.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[14] Hui Mao, Ming Cheung, and James She. Deepart: Learning joint representations of visual arts. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 1183–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[15] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces, 2020.

[16] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[19] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Improved artgan for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2019.