Automatic classification of works of art

Anissa EIF — Celio B — Jean G — Myriam L

***Abstract*—We present a deep learning pipeline for unsupervised clustering of paintings according to their artistic movements. Our approach combines both local and global visual feature representations. At the local level, we extract handcrafted features such as DoG and gradient-based descriptors, which are then encoded using an InfoVAE to capture low-level structural information. At the global level, we fine-tune a Vision Transformer using a self-supervised learning approach to learn semantic representations from entire images. These features are concatenated to form a joint embedding, which is used as input to a GMM for clustering. This work explores the potential of exploiting deep generative models for structure discovery in large-scale art collections.**

I. Introduction

We are provided with a set of artworks (8000 from 300 painters). The aim of this project is to cluster these artworks in a manner comparable to classifications found in art treatises. This work focuses on leveraging deep features to construct and iteratively refine clusters.

This classification can have several practical applications :

- Facilitating cataloging pieces of art or creating thematic exhibitions for museums.
- Discovering unexpected stylistic groupings and generating new hypotheses on the evolution of styles for researchers.
- Opening the way to new cultural mediation tools and democratizing access to culture.

II. Related work

Somepalli et al. present a framework for understanding and extracting style descriptors from images [1] with the scope of measuring style similarity in images generated by diffusion models.

With archetypal analysis, Wynen et al. introduced an unsupervised learning approach based to automatically discover, summarize, and manipulate artistic styles from large collections of paintings [2].

Vision Transformers (ViTs) have recently emerged as a competitive alternative to CNNs for image analysis. By processing images as sequences of patches through self-attention, ViTs capture long-range dependencies across the entire image. This can make them relevant for artwork analysis, where meaningful visual relationships often extend beyond local regions [3].

III. Tests and experiments

A. Graph clustering

Given the nature of the problem we first thought of using GCNs because :

- We could use painters' names for a good initialization.
- There are stylistic links between paintings of the same style.

Thus, the graph structure appeared well suited for the problem [4]. We tried using a Variational Graph Auto-Encoder to encode the links between the different paintings [5]. We used paintings belonging to the same painter as initialization and features extracted with ResNet-50 trained on ImageNet.

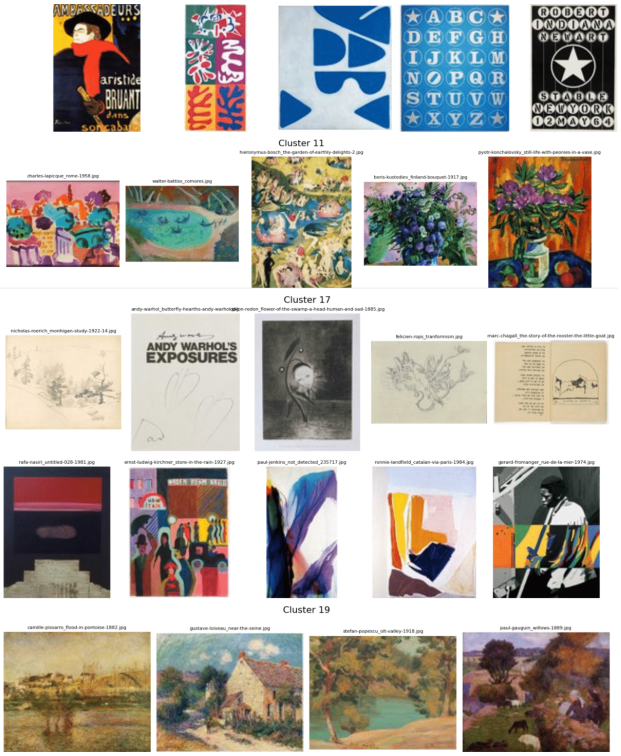

Fig. 1. Clustering results obtained with GCN architecture

However we had two issues:

- The models relied too much on initialization, various artists didn't keep the same style for all their paintings, Picasso for instance.
- We realized extracting features from the last layer of a model trained on ImageNet was a bad idea when considering style descriptors extraction, as it was trained to detect objects (Fig. 1. shows it links black and white paintings with text and gray sketches).

B. Gram matrices

Another idea explored was to take pre-trained neural networks to extract features from our images and then perform a classic clustering. To do so, we took the ResNet model and extracted the layers 8, 10, 15 and 22. To make sure we cluster

on the style of the image and not on the object represented in the image (we do not want to have all the paintings of dog together for instance), we are using Gram matrices to get rid of spacial information. The global pipeline of this method is the following:

- Extract the feature maps of the selected layer for a given image. All the feature maps are stored in a matrix.
- Calculate the gram matrix of the image matrix and then flatten it.
- Clustering using K-means.

We tested this method on a small subset of the data and we observed that the results were unsatisfying. Indeed, some clusters were populated by only one painter but some others were with paintings from many various styles.
This result can be explained by the fact that the features extracted are not pertinent or that the pipeline is too simple to capture the whole style of an image.

C. Auto-Encoders for local features

Another idea we had was to look for patterns in small details and ways to reconstruct them. While an AE could indeed recreate an image from low dimension (fig. 2) the latent space would not be suited for any type of clustering.

The architecture used to obtain the reconstructions shown in figure 2 is:

- Encoder: 3× (Conv2d: 3→64→128→256, kernel 4, stride 2, padding 1, ReLU) → Flatten → Linear (256×32×32 → $latent\_dim$).
- Decoder: Linear ($latent\_dim$ → 256×32×32) → Unflatten → 3× (ConvTranspose2d: 256→128→64→3, kernel 4, stride 2, padding 1, ReLU) → Sigmoid.

Fig. 2. Reconstruction with AE_local_patch - $latent\_dim = 256$

As observed, the reconstruction os really unsatisfying. In order to obtain a better built latent space, we then explored VAEs, used further in the *Local Features* section.

D. Self-supervised learning

We had the idea of searching for a pre-trained model on a large dataset and then fine-tune it on our images. Thus, we used simCLR, a framework for contrastive learning (Fig. 3). The core idea is to train the model so that different augmentations of the same image produce similar representations in the latent space. In the case of artworks, it is interesting to look at augmentations that preserve style (e.g. not affecting colors too much).

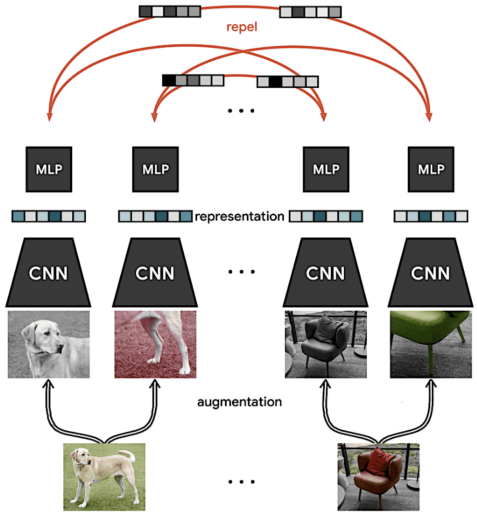

Fig. 3. Credit to Ting Chen et al.

To try new ideas on a smaller and more interpretable dataset, we used DomainNet to try to cluster between the different simpler styles in this dataset.

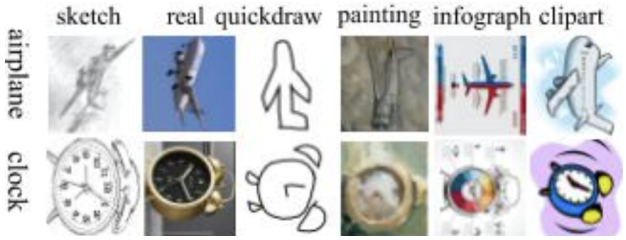

Fig. 4. Example of the 6 styles available in DomainNet

- Using simCLR we managed to improve results: Table I.

Architecture	ViT	ViT + SSL	ViT + curated SSL
Accuracy	0.78	0.83	0.83

TABLE I
RESULTS ON A SMALL SUBSET OF DOMAINNET*

**Subset of DomainNet, same object, different styles*
Acc = $\frac{1}{N}\sum(c_i = l_i)$.

Here we do not see any improvement with curated SSL but Somepalli et al. showed that its efficiency for style related tasks on larger datasets [1].

IV. PROPOSED METHOD

We combine these findings to achieve an innovative method combining both local and global features for painting clustering (Fig.5).

To our knowledge no other work has been made for considering both local and global features for style identification regarding paintings.

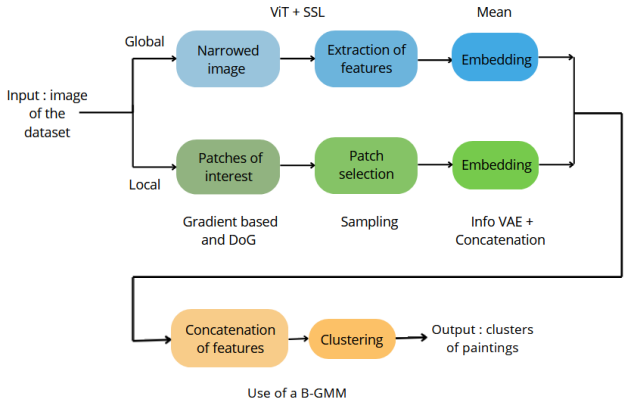

Fig. 5. Proposed pipeline

V. Model architecture

An acceptable clustering is only possible if we have access to great features. While it may be tempting to use pre-trained networks such as resNet-X, those are trained on imageNet, making them efficient for the classification of **objects** but not for style identification. Here, we are looking for specific details, such as mixtures of colors, brushes specific to an artist, the 'vibe' of a painting. In order to find such features, we choose to consider two different scales.

A. Global features

Images are narrowed to a 224x224 size and passed into the encoder part of a ViT. It splits the image into same-size patches, which are then flattened and projected into embeddings. These are combined with positional encodings and passed through a transformer encoder made of multi-head self-attention and feedforward layers. Unlike CNNs, a ViT models long-range dependencies between patches from the *entire* image, which is better suited for considering style, a task different from just detecting objects [6, 3].

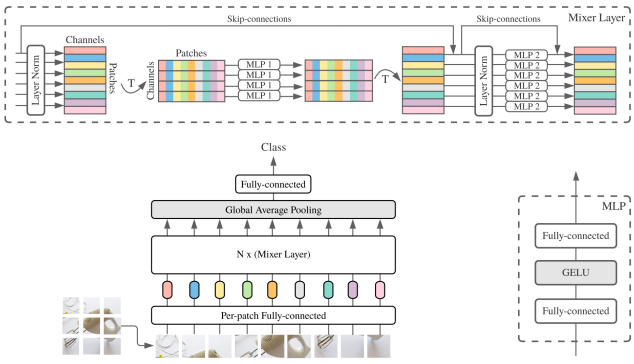

Fig. 6. ViT architecture, source: google

We use weights of the model trained on Contra-Styles dataset [1]. This allows the extraction of style specific features.

Yet we couldn't get proper results on real paintings with sole style features extraction. Experiments on DomainNet showed that the model could cluster images of different styles but having the same content (e.g. painting or real **planes**) but struggled to provide a good clustering when we considered different objects.

That means we can have some information about the style, but the influence of the object over the latent description of images is too strong for our task. This gave us the intuition of using local features to get pass these object detection issues.

Model	ViT native weights	ViT CSD weights
Accuracy	0.74	0.78

TABLE II
CHANGE IN RESULTS WHEN USING PRETRAINED WEIGHTS

B. Local features

The core of this part of the architecture is to extract pertinent patches of the image to capture the small details that define the artist's style before extracting the embedding of those extracted parts.

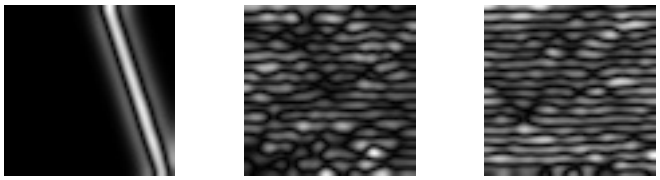

Fig. 7. Examples of patches extracted

1) Patch extraction: .
To extract the interesting parts, we rely on a score capturing the amount of details in a patch. To do so, we combine gradient-based patch selection with Difference of Gaussians (DoG) filtering to extract pertinent local features from our images.

Gradient Computation
The first step is to calculate the gradient of our image. We start by converting our image to grayscale thanks to luminance conversion. We allow ourselves to get rid of the color information since it will already be encoded in the global part.

Then, the gradient magnitude at each pixel is computed using the Sobel operator. The horizontal and vertical gradients are calculated as:

$$G_x(x,y) = I_{\text{gray}} * S_x$$
$$G_y(x,y) = I_{\text{gray}} * S_y$$

where S_x, S_y are the Sobel kernels and

$$I_{\text{gray}}$$

is our image in grayscale.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The gradient magnitude is then computed as:

$$|\nabla I|(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

Patch Extraction and Scoring

Once the gradient is computed, we divide the image into overlapping patches $P_{i,j}$ of size 64×64 pixels, where (i,j) represents the top-left corner coordinates. The patches are extracted with a stride 32 to ensure 50 % overlap.

For each patch $P_{i,j}$, we compute a gradient-based score:

$$\text{Score}(P_{i,j}) = \frac{1}{s^2} \sum_{x=i}^{i+s-1} \sum_{y=j}^{j+s-1} |\nabla I|(x,y)$$

This score represents the average gradient magnitude within the patch, serving as a measure of local texture complexity and edge density.

Top Patch Selection

Let $\mathcal{S} = \{\text{Score}(P_{i,j}) : (i,j) \in \mathcal{P}\}$ be the set of all patch scores. We select the top 10% patches based on their gradient scores.

Weighted Random Sampling

From the top patches $\mathcal{P}_{\text{top}}$, we perform weighted random sampling to select 5 patches for final processing. The probability of selecting patch $P_{i,j}$ is proportional to its gradient score:

$$p(P_{i,j}) = \frac{\text{Score}(P_{i,j})}{\sum_{P_{m,n} \in \mathcal{P}_{\text{top}}} \text{Score}(P_{m,n})}$$

This ensures that patches with higher gradient activity have a higher probability of being selected while maintaining diversity in the final patch set. We could have selected the 5 best patches but during experimentation we realized that by doing so we tended to select patches that were close geographically. By adding some randomness we ensure that the patches are well distributed on the points of interest in the image.

Difference of Gaussians (DoG) Filtering

The patches we have extracted are colored parts of the original image. Since the goal of local patches is to detect small details we want to extract what is essential in the patch. to do so we only keep edges information in the image :

For each selected patch $P_{i,j}$, we apply a Difference of Gaussians filter to enhance edge information. The DoG filter is defined as:

$$\text{DoG}(P_{i,j}) = G_{\sigma_1} * P_{i,j} - G_{\sigma_2} * P_{i,j}$$

where G_σ represents a Gaussian kernel with standard deviation σ:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

The parameters σ_1 and σ_2 satisfy $\sigma_1 < \sigma_2$, typically set to $\sigma_1 = 3.0$ and $\sigma_2 = 5.0$. This configuration creates a band-pass filter that emphasizes edges and texture details while suppressing both high-frequency noise and low-frequency illumination variations.

The DoG operation can be mathematically expressed as:

$$\text{DoG}(P_{i,j}) = \left(\frac{1}{2\pi\sigma_1^2} e^{\left(-\frac{r^2}{2\sigma_1^2}\right)} - \frac{1}{2\pi\sigma_2^2} e^{\left(-\frac{r^2}{2\sigma_2^2}\right)} \right) * P_{i,j}$$

where $r^2 = x^2 + y^2$.

Normalization

The final DoG-filtered patches are normalized to the range $[0, 255]$ to match classical image representations:

$$P_{\text{final}}(x,y) = 255 \cdot \frac{|\text{DoG}(P_{i,j})(x,y)| - \min(|\text{DoG}(P_{i,j})|)}{\max(|\text{DoG}(P_{i,j})|) - \min(|\text{DoG}(P_{i,j})|)}$$

The absolute value ensures that both positive and negative edges are preserved as intensity variations.

Pipeline Summary

The complete pipeline can be summarized as the following sequence of operations:

- 1. **Gradient Computation:** $I \to |\nabla I|$
- 2. **Patch Scoring:** $\{P_{i,j}\} \to \{\text{Score}(P_{i,j})\}$
- 3. **Top Selection:** $\mathcal{S} \to \mathcal{P}_{\text{top}}$ (top 10
- 4. **Weighted Sampling:** $\mathcal{P}_{\text{top}} \to \{P_1, P_2, \ldots, P_n\}$
- 5. **DoG Filtering:** $P_k \to \text{DoG}(P_k)$ for $k = 1, \ldots, n$
- 6. **Normalization:** $\text{DoG}(P_k) \to P_{\text{final},k}$

This approach ensures that the selected patches contain rich texture information and edge details, which are crucial for distinguishing between different artistic styles and techniques.

2) Embedding: .

Once the 5 patches were selected, the goal was to represent them in a well-structured latent space. It has been observed that Variational Autoencoders (VAEs) generate better latent codes than traditional Autoencoders. However, VAEs also have their drawbacks, one of which is posterior collapse, meaning that the decoder becomes sufficiently powerful to ignore the latent representation. Furthermore, the ELBO loss of VAEs is the sum of two terms: a log likelihood (reconstruction) term $L_{AE}(x)$ and a regularization term $L_{REG}(x)$:

$$\mathcal{L}_{\text{ELBO}}(x) = L_{\text{AE}}(x) + L_{\text{REG}}(x) \tag{1}$$

$$\equiv \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathcal{D}_{\text{KL}}(q_\phi(z|x) \parallel p(z)) \tag{2}$$

As a result, optimizing this loss does not guarantee good inference performance. This is why Info-VAEs were introduced. Their primary purpose is to generate a well-structured latent space with strong inference capabilities. This is achieved by adding terms to the ELBO loss, specifically the mutual information between z (the latent code) and x (the reconstruction). Zhao et al. in [7] optains the following loss :

$$\mathcal{L}_{\text{infoVAE}}(x) = \mathbb{E}_{p_{\mathcal{D}}(x)}\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \quad (3)$$

$$(1-\alpha)\mathbb{E}_{p_{\mathcal{D}}(x)}[\mathcal{D}_{\text{KL}}(q_\phi(z|x)||p(z))] - \quad (4)$$

$$(\alpha+\lambda-1)\mathcal{D}_{\text{KL}}(q_\phi(z)||p(z)) \quad (5)$$

This helped avoiding the issue of posterior collapse and leads to improved inference performance, as demonstrated in the paper of Zhao et al. . The first two terms can be optimized using the reparameterization trick. The last one demands more efforts to be optimized, so we used the approximation proposed by Zhao et al. .

In order to train our Info-VAE architecture, we extracted the top 10% patches from each painting, using the protocol described above, and we randomized the data in this new dataset.

We used a common encoder architecture consisting of 4 blocks of Conv2d (with kernel $= 3$, padding $= 1$, stride $= 2$ and respectively n_channels $= 32, 64, 128, 256$) $\rightarrow$ BatchNorm $\rightarrow$ leaky ReLU followed by a fully connected layer with a latent dimension of 64. The decoder follows a symmetrical architecture, mirroring the structure of the encoder.

Then, we used the AdamW optimizer with a learning rate of 10^{-3} and a weight decay of 10^{-4} on 5 epochs.

We obtained at the end of epoch 5 a training loss of 0.3619 and a similar testing loss, indicating that the model did not overfit.

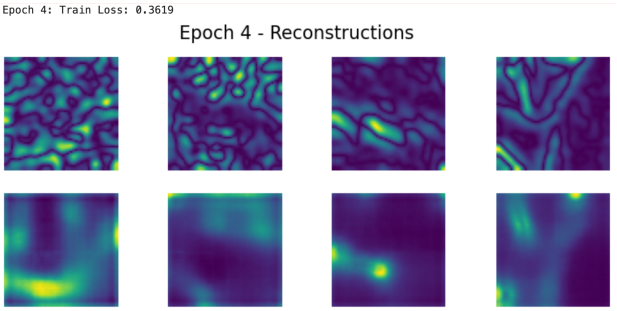

Fig. 8. At the top : the original data, at the bottom : the reconstructions at epoch 5 (here noted 4 because the first epoch is 0)

We can see that the reconstruction is poor. Nevertheless, we hope that the model has created good latent representations. A simple VAE with ELBO loss would certainly have a better result, but our info-VAE should create a better latent space.

VI. Merging and Clustering

A. Initial idea

Once features were extracted, we tried to combine them with an encoder architecture. Xie et al. introduced *DEC* [8], a method that learns a mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective.

B. Method retained

The method we finally chose to retain is to concatenate the different embeddings and then apply a Bayesian Gaussian Mixture of Model to cluster the artworks.

VII. Results

Our latest clustering did not provide successful results. Automatically determining the number of cluster was not doable because of their low quality. Paintings were mixed together without any logic. For example those two paintings below were in the same cluster :

Fig. 9. Same cluster but different styles (Watts - Watteau)

We could think of different explanations of why our work was not conclusive :

- Bad embedding (from local and global): we struggled to capture the style of an artwork and thus to cluster it. This could be explained by many factors. The most plausible ones are that the embeddings created in our local method were inadequate or that the transformer used in the global method failed to capture the aspect of the image.
- Bad merging : as expected, merging embeddings solely using a concatenation was not sufficient. This could lead to some losses in data or to creation of artifacts misleading the clustering.

VIII. Limitations

- Local features: when considering local patches, we look at same size ones (64x64). However, if we consider two paintings of very different physical sizes — for example, Mona Lisa by Leonardo da Vinci $0.4m^2$ and The Wedding Feast at Cana by Paolo Veronese $66m^2$ both resized to a 1024×1024 image, the level of detail captured in a 64×64 patch will not be comparable between the two works.
- The highest available online resolution for The Wedding Feast at Cana is 4214×2863 pixels, meaning that a 64×64 patch corresponds to approximately a 15cm square of the original painting. In contrast, the highest resolution for Mona Lisa is 7479×11146 pixels. Moreover, our dataset includes images at various resolutions. A potential improvement would be to standardize or adjust image resolutions in the dataset to ensure more consistent scaling between images, enabling fairer and more meaningful patch-based comparisons.

References

[1] Gowthami Somepalli et al. *Measuring Style Similarity in Diffusion Models*. 2024. arXiv: 2404.01292 [cs.CV]. URL: https://arxiv.org/abs/2404.01292.

[2] Daan Wynen, Cordelia Schmid, and Julien Mairal. *Unsupervised Learning of Artistic Styles with Archetypal Style Analysis*. 2018. arXiv: 1805.11155 [stat.ML]. URL: https://arxiv.org/abs/1805.11155.

[3] Alexander Kolesnikov et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021.

[4] Jie Zhou et al. *Graph Neural Networks: A Review of Methods and Applications*. 2021. arXiv: 1812.08434 [cs.LG]. URL: https://arxiv.org/abs/1812.08434.

[5] Thomas N. Kipf and Max Welling. *Variational Graph Auto-Encoders*. 2016. arXiv: 1611.07308 [stat.ML]. URL: https://arxiv.org/abs/1611.07308.

[6] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[7] Shengjia Zhao, Jiaming Song, and Stefano Ermon. *InfoVAE: Information Maximizing Variational Autoencoders*. 2018. arXiv: 1706.02262 [cs.LG]. URL: https://arxiv.org/abs/1706.02262.

[8] Junyuan Xie, Ross Girshick, and Ali Farhadi. *Unsupervised Deep Embedding for Clustering Analysis*. 2016.