Joint Prediction of Style and Genre in the WikiArt Dataset

Wassim Mecheri

April 2025

1 Introduction

This project explores the joint prediction of artistic style and genre using the WikiArt dataset, a large-scale collection of paintings annotated with metadata such as artist, style, and genre. The objective is to develop a model capable of predicting both the style and genre of a painting simultaneously.

To tackle this multi-label classification task, a multi-head neural network was implemented, based on a ResNet-50 backbone. The model was fine-tuned specifically for this purpose, with one classification head dedicated to style prediction and the other to genre prediction. Prior to training, the dataset was carefully processed to ensure class balance and proper label remapping. Hyperparameter tuning was performed to optimize the model's performance.

In addition to the classification task, an interactive component was developed: a guessing game powered by a language model, TinyLlama. The game utilizes the classifier's predictions to generate descriptive hints for paintings, challenging users to guess whether the model or the LLM is attempting to deceive them.

This report details the dataset preparation, model architecture, training process, evaluation, and the design of the **interactive application**, along with insights gained throughout the project in each step.

2 Data Preparation

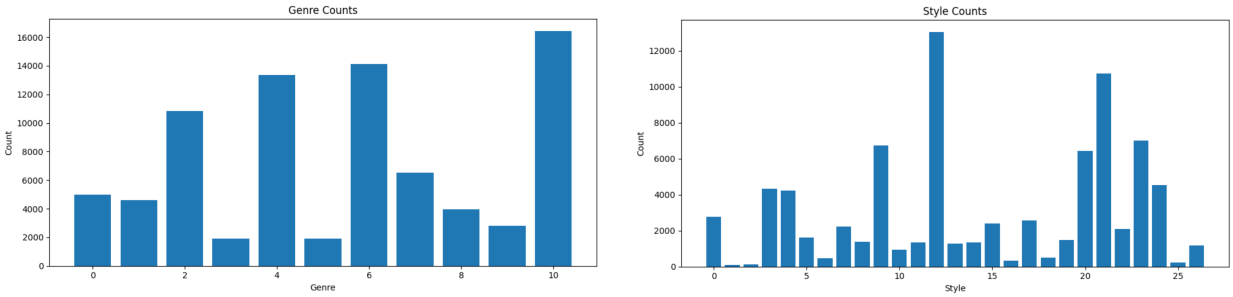
The first step of the project consisted of an in-depth exploration of the WikiArt dataset to understand its structure and identify any potential issues before model development. The dataset contains 81,444 artworks by various artists, collected from WikiArt.org, and includes labels for three main categories: 129 artist classes (including "Unknown Artist"), 11 genre classes (including "Unknown Genre"), and 27 style classes.

A random sample of images was visualized to ensure the integrity and visual quality of the data, and to gain an initial impression of the diversity across artistic styles and genres.

Figure 1: Sample images from the WikiArt dataset illustrating variation in style and composition.

Following this, the dataset was checked for missing values in the critical fields, namely the image paths, style labels, and genre labels. No missing entries were detected, confirming that the dataset was complete and could be used without the need for imputation or cleaning.

The distribution of genre and style labels was then analyzed to assess the level of class imbalance. The results revealed a significant skew, with certain genres and styles being far more represented than others. This imbalance posed an evident challenge for the classification task.



(a) Genre Label Distribution

(b) Style Label Distribution

Figure 2: Distribution of genre and style labels in the original WikiArt dataset.

To better understand the complexity of the task, the number of unique genre-style label pairs was also computed. This analysis revealed that some combinations were significantly more frequent than others, highlighting the sparsity and imbalance in the joint label distribution.

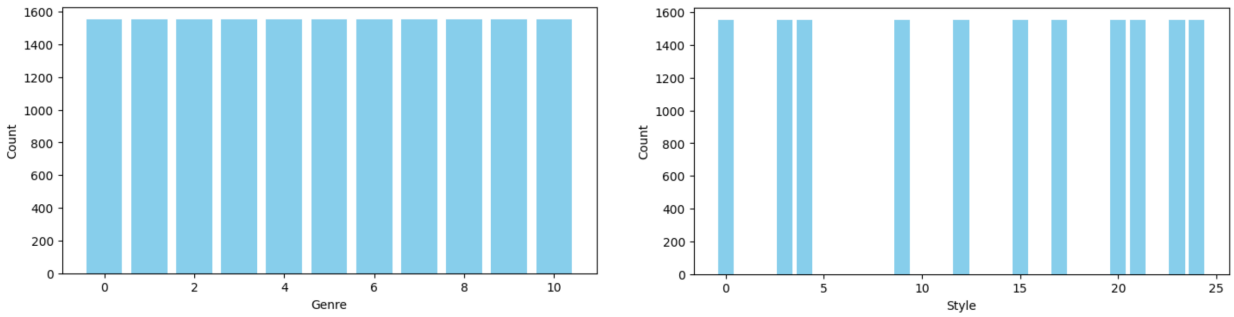
Since the model is designed to predict both genre and style simultaneously, it is essential to balance the dataset with respect to their joint occurrences. This means that instead of balancing each label independently, the goal is to select an appropriate number of samples from each genre-style pair to ensure a more uniform representation across the joint label space.

2.1 Dataset Balancing

To address the significant imbalance observed in the distribution of genre and style labels, a custom balancing procedure was implemented. The goal was to construct a subset of the dataset where each genre and style is equally represented, while preserving the joint relationships between them.

The approach involved analyzing the availability of samples for each genre-style pair, then filtering out styles that did not meet a minimum threshold of examples. A maximum matching strategy was used to determine how many samples could be evenly drawn from each pair, ensuring that all retained genres and styles would have an equal number of instances in the final dataset. This was achieved using a flow network formulation to verify the feasibility of the balancing, followed by uniform sampling across the selected pairs.

As a result, a balanced dataset was created where each genre and each style is represented by the same number of samples, while having the same number of genre and style. However, this balancing step introduced a new issue: while all genre labels were retained, a subset of style labels had to be excluded due to insufficient data. This led to gaps in the label indexing and motivated the next step: remapping the labels to ensure consistency and compatibility for model training.



(a) Balanced Genre Distribution

(b) Balanced Style Distribution

Figure 3: Balanced label distributions after preprocessing.

2.2 Label Remapping

To resolve the non-continuous labels, a remapping procedure was applied to the style labels. Each remaining style was assigned a new, consecutive integer index starting from zero to 10. This ensured that the style labels were compact and properly formatted for use in a classification setting.

The remapped dataset preserves the balanced distribution across both genre and style, and is now in a clean and consistent format suitable for training. Visual inspection of the new label distributions confirmed that the balancing was preserved, and all genre and style labels are now properly indexed.

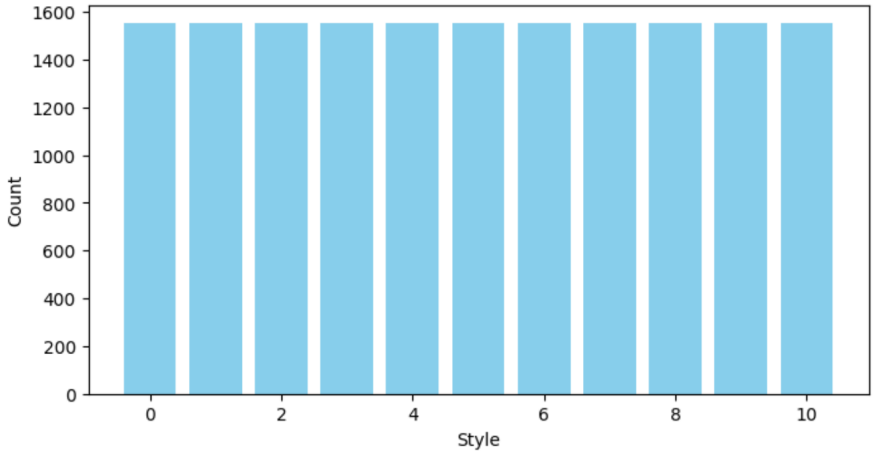


Figure 4: Style label distribution after remapping

3 Training and Evaluation

3.1 Final Dataset

As a result of the preprocessing and balancing steps described earlier, the final dataset used for training and evaluation consists of 17,061 images, 11 genre labels, and 11 style labels. These images are evenly distributed across the selected genre and style labels to ensure a fair learning environment for the model.

The dataset was split into 67.5% for training, 12.5% for validation (or 10% of training), and 20% for testing. This stratified split maintains the class balance across all subsets and ensures that the model is evaluated on examples it has never seen during training.

3.2 Baseline CNN Model

As an initial baseline, a simple convolutional neural network (CNN) was implemented with a multi-head structure to jointly predict the genre and style of paintings. The architecture consisted of three convolutional layers followed by fully connected classification heads for each label. The model was trained using the Adam optimizer with a learning rate of 1×10^{-3} and cross-entropy loss for both outputs. No hyperparameter tuning was performed. Data augmentation techniques such as random cropping, flipping, and RandAugment were applied during training.

The CNN was trained for 10 epochs using a split of 67.5% for training, 12.5% for validation, and 20% for testing. Despite its simplicity, the model achieved a genre accuracy of 33.9%, style accuracy of 27.6%, and joint prediction accuracy of 14.7% on the test set. While these results establish a starting point, they highlight the limitations of using shallow architectures for complex multi-label visual classification tasks.

3.3 ResNet-50

As a second baseline after the simple CNN, a ResNet-50 pretrained on ImageNet was employed, with all its weights frozen. Only two newly added classifier heads were trained: one for genre prediction and one for

style prediction. Each head consisted of a linear layer followed by ReLU activation and a final linear output layer.

The backbone served purely as a fixed feature extractor, with no layers updated during training. This setup allowed us to evaluate the raw transferability of features learned from natural images to the domain of paintings without fine-tuning.

The model was trained for 10 epochs using the Adam optimizer with a learning rate of 1×10^{-3} and a ReduceLROnPlateau scheduler. No hyperparameter tuning was performed. The dataset was split in the same way as described previously.

The model achieved moderate performance:

Table 1: Performance of the ResNet-50 model with frozen backbone on training, validation, and test sets.

Metric	**Train**	**Validation**	**Test**
Genre Accuracy	54.98%	56.92%	56.46%
Style Accuracy	46.87%	45.60%	46.70%
Joint Accuracy	31.55%	30.42%	30.65%
Loss	2.88	2.86	2.87

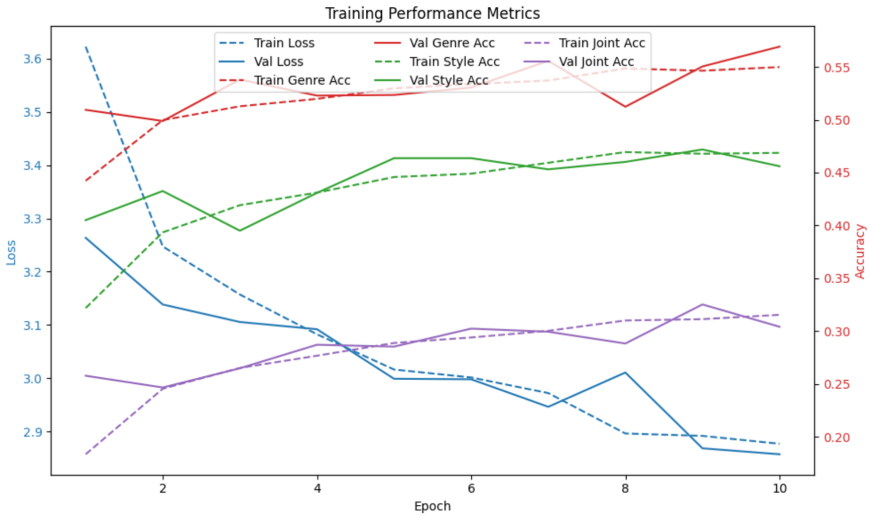


Figure 5: Training and validation metrics across epochs for ResNet-50.

This result demonstrates that the pretrained features alone provide a starting point, but are totally insufficient to fully capture the domain-specific visual cues required for high performance without further adaptation.

3.4 Fine-Tuned ResNet-50

To further improve upon the baseline, the same ResNet-50 architecture was fine-tuned by unfreezing its last four residual blocks while keeping the earlier layers frozen. The classification heads remained unchanged, with separate branches for genre and style prediction.

Unlike the frozen version, this model was able to adapt its high-level representations to the domain of fine art, allowing for better performance on the task. Hyperparameter tuning was performed over learning rate, batch size, weight decay, and dropout using a grid search strategy. The model was trained for 20 epochs using the Adam optimizer and the same ReduceLROnPlateau scheduler.

This model achieved much better performance:

Table 2: Best Hyperparameter Configuration (Fine-Tuned ResNet-50)

Learning Rate	1×10^{-4}
Batch Size	32
Dropout	0
Weight Decay	1×10^{-4}

Table 3: Performance of Fine-Tuned ResNet-50 (Epoch 18)

Metric	**Validation**	**Test**
Genre Accuracy	71.9%	70.6%
Style Accuracy	70.1%	68.2%
Joint Accuracy	56.4%	53.8%
Loss	2.12	2.01

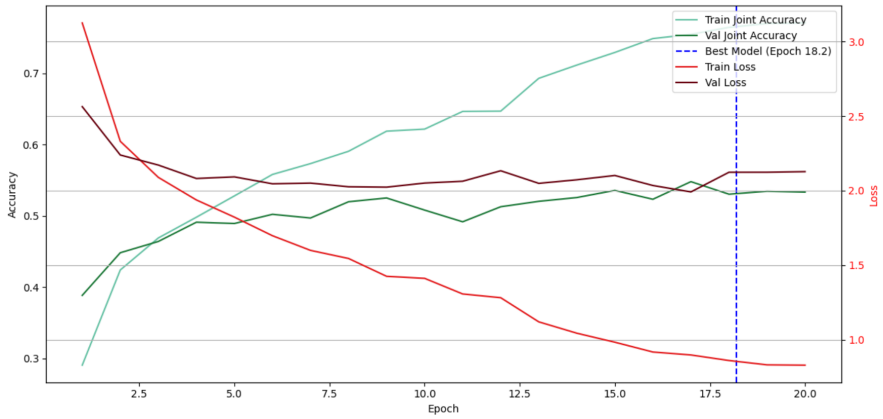


Figure 6: Training and validation metrics across epochs for the fine-tuned ResNet-50.

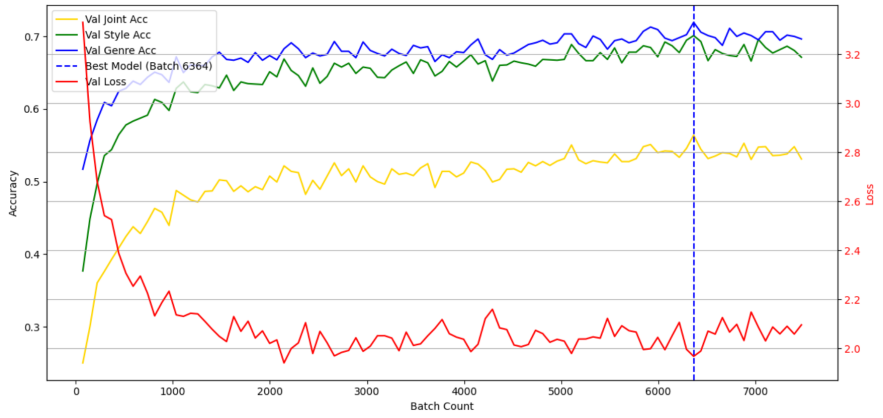


Figure 7: Validation metrics across evaluation batches during training of the fine-tuned ResNet-50.

These results demonstrate a clear improvement over both the CNN and the frozen ResNet-50 baselines. The ability to fine-tune higher-level layers significantly increased classification accuracy, especially in the joint prediction task, highlighting the value of domain adaptation through partial training of the pretrained backbone.

4 Evaluation and Testing

4.1 Test Performance

After identifying the best configuration through hyperparameter tuning, the fine-tuned ResNet-50 model was evaluated on the held-out test set to assess its generalization capabilities.

The model achieved the following results:

Table 4: Final test performance of the fine-tuned ResNet-50 model.

Metric	**Value**
Test Loss	2.0097
Genre Accuracy	70.6%
Style Accuracy	68.2%
Joint Accuracy	53.8%

These results indicate that the model performs well on both individual tasks and the more challenging joint prediction objective. Compared to the initial CNN baseline, this represents a substantial improvement in all metrics, highlighting the effectiveness of transfer learning and fine-tuning.

To further understand the model's strengths and weaknesses, confusion matrices and classification reports were generated for both genre and style predictions. They illustrate the overall distribution of correct and incorrect predictions. While the model shows consistent performance across several labels, certain classes remain difficult to distinguish, often due to visual or conceptual similarities between them.

Table 5: Genre Classification Report on Test Set

Genre	**Precision**	**Recall**	**F1-score**	**Support**
Abstract Painting	0.82	0.87	0.85	323
Cityscape	0.76	0.73	0.75	297
Genre Painting	0.56	0.57	0.56	320
Illustration	0.78	0.77	0.78	309
Landscape	0.70	0.69	0.70	332
Nude Painting	0.73	0.79	0.76	304
Portrait	0.71	0.70	0.71	296
Religious Painting	0.69	0.73	0.71	331
Sketch And Study	0.64	0.70	0.67	295
Still Life	0.81	0.83	0.82	298
Unknown Genre	0.51	0.39	0.44	308
Accuracy			0.71	3413
Macro Avg	0.70	0.71	0.70	3413
Weighted Avg	0.70	0.71	0.70	3413

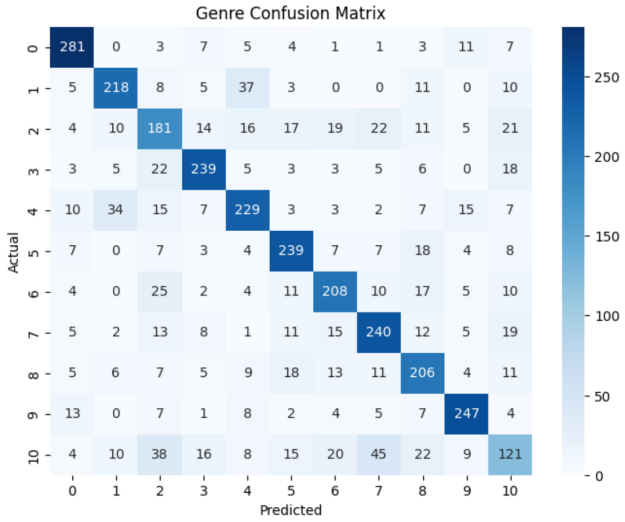


Figure 8: Confusion matrix for genre classification on the test set.

Table 6: Style Classification Report on Test Set

Style	**Precision**	**Recall**	**F1-score**	**Support**
Abstract Expressionism	0.83	0.88	0.85	327
Art Nouveau	0.74	0.66	0.69	309
Baroque	0.75	0.79	0.77	303
Expressionism	0.58	0.52	0.55	328
Impressionism	0.64	0.64	0.64	313
Naive Art Primitivism	0.72	0.79	0.75	318
Northern Renaissance	0.81	0.88	0.84	323
Post Impressionism	0.56	0.57	0.56	283
Realism	0.53	0.50	0.51	303
Romanticism	0.72	0.70	0.71	306
Symbolism	0.56	0.56	0.56	300
Accuracy			0.68	3413
Macro Avg	0.68	0.68	0.68	3413
Weighted Avg	0.68	0.68	0.68	3413

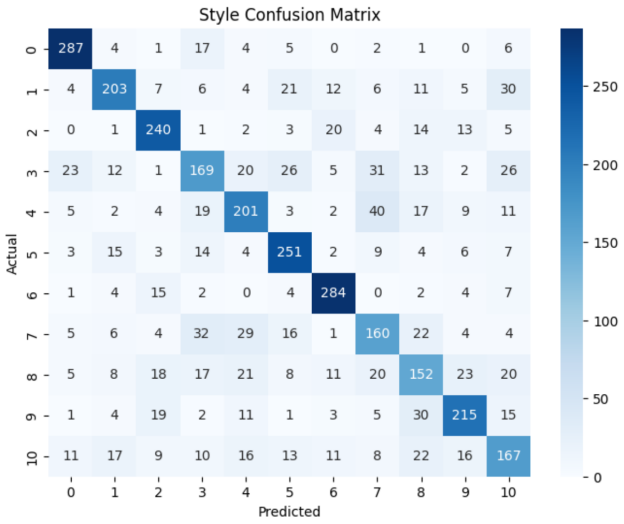


Figure 9: Confusion matrix for style classification on the test set.

4.2 Label Pair Bias and Distribution Imbalance

Although the dataset was balanced in terms of individual genre and style labels, the joint distribution of genre-style pairs remained highly imbalanced. This occurred as a result of the pair-wise sampling strategy used during dataset construction, which equalized counts per label but did not explicitly control the frequency of each genre-style combination.

The impact of this imbalance is evident in the style classification results when broken down by genre. Several genres are dominated by a single associated style in the training set, leading the model to overfit to those combinations. As a result, the model performs very well on common genre-style pairs but struggles on underrepresented ones.

Per-genre style classification reports reveal that for many genres, styles with low representation are consistently misclassified or ignored altogether. In extreme cases, some styles linked to a genre are never predicted correctly, showing that the model only learns a limited range of styles for each genre.

Good genre examples where this problem is highlighted are: Abstract Painting, where most predictions default to Abstract Expressionism; Nude Painting, where several styles receive little to no attention; and Unknown Genre, where predictions are dominated by just a few styles, with many others being completely ignored.

Table 7: Style Classification Report for Genre: Abstract Expressionism

Style	**Precision**	**Recall**	**F1-score**	**Support**
Abstract Expressionism	0.99	0.88	0.93	317
Expressionism	0.00	0.00	0.00	6
Others (all zero)	0.00	0.00	0.00	0
Accuracy			0.87	323
Macro Avg	0.11	0.10	0.10	323
Weighted Avg	0.97	0.87	0.91	323

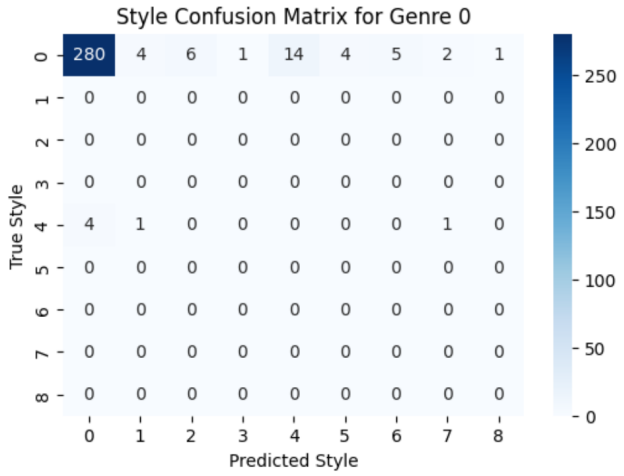


Figure 10: Style prediction confusion matrix for the genre *Abstract Painting*.

Table 8: Style Classification Report for Genre: Nude Painting

Style	**Precision**	**Recall**	**F1-score**	**Support**
Abstract Expressionism	0.20	0.33	0.25	3
Art Nouveau	0.60	0.69	0.64	13
Symbolism	0.00	0.00	0.00	12
Baroque	0.29	0.67	0.40	3
Expressionism	0.74	0.74	0.74	114
Impressionism	0.79	0.61	0.69	75
Naive Art Primitivism	0.25	0.80	0.38	5
Northern Renaissance	0.57	0.80	0.67	5
Post Impressionism	0.28	0.23	0.25	39
Realism	0.45	0.52	0.48	25
Romanticism	0.17	0.10	0.12	10
Accuracy			0.57	304
Macro Avg	0.39	0.50	0.42	304
Weighted Avg	0.59	0.57	0.57	304

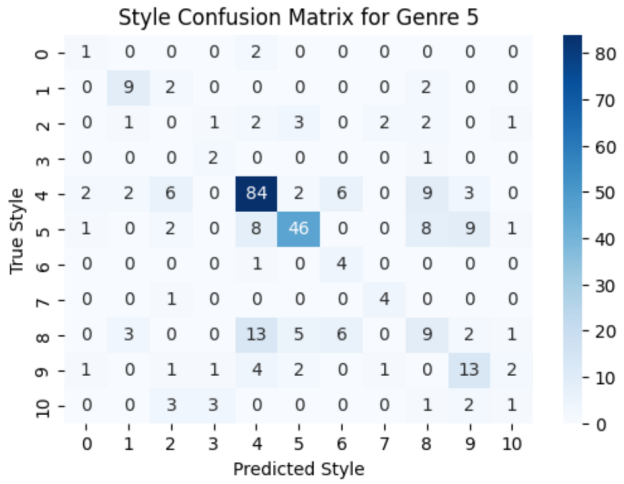


Figure 11: Style prediction confusion matrix for the genre *Nude Painting*.

Table 9: Style Classification Report for Genre: Unknown Genre

Style	**Precision**	**Recall**	**F1-score**	**Support**
Art Nouveau	0.87	0.59	0.71	69
Symbolism	0.72	0.53	0.61	55
Baroque	0.89	0.82	0.85	104
Northern Renaissance	0.88	0.85	0.87	80
Others (all zero)	0.00	0.00	0.00	0
Accuracy			0.72	308
Macro Avg	0.31	0.25	0.28	308
Weighted Avg	0.86	0.72	0.78	308

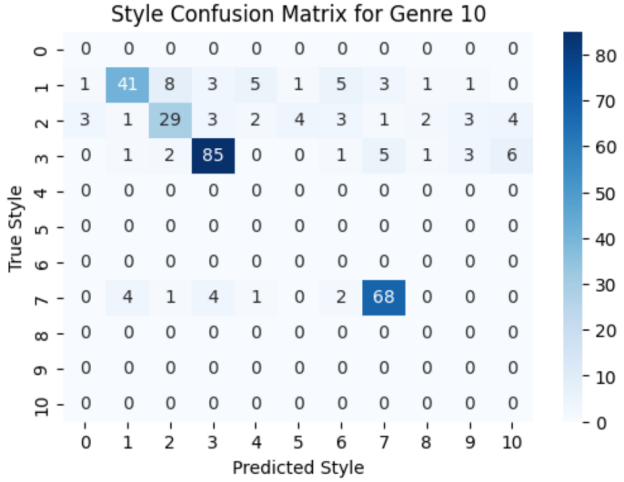


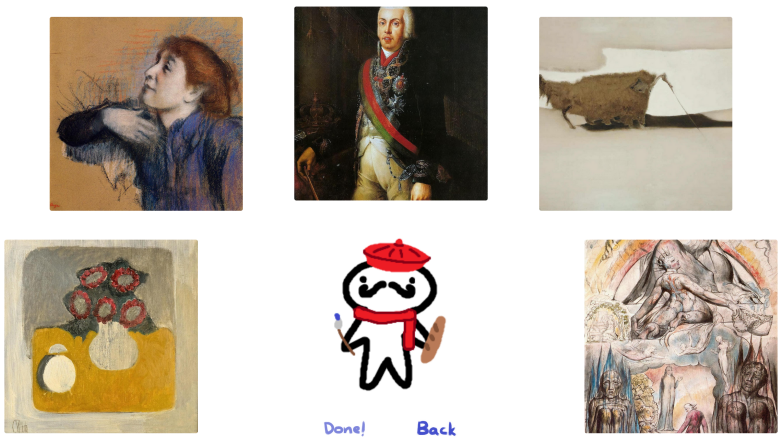
Figure 12: Style prediction confusion matrix for the genre *Unknown Genre*.

This highlights a key limitation of the current balancing strategy: while fairness is ensured at the genre and style level individually, the joint label space remains uneven. Addressing this issue may require alternative sampling techniques that balance properly the joint distribution, or the use of cost-sensitive learning and targeted augmentation strategies to emphasize underrepresented pairs.

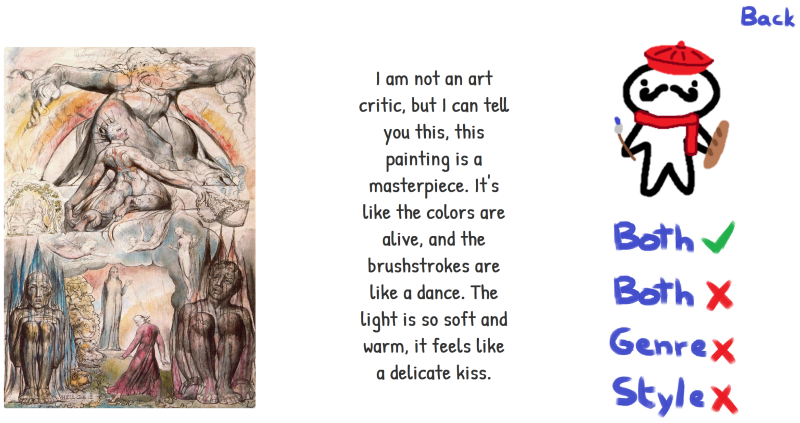
5 Interactive Application: Bob's Bluffing Gallery

In addition to the classification model, this project includes a playful interactive component designed to highlight the strengths and flaws of the predictive system in an engaging way. The result is **Bob's Bluffing Gallery**, a web-based game where users interact with an AI-powered character named Bob.

Bob is a novice tour guide working at a painting gallery. He is French, enthusiastic, and eager to impress visitors, but he's still learning the ropes. In each game session, the player is shown five different paintings. For each one, Bob provides a short description, attempting to identify the painting's genre and style. The twist: Bob's knowledge is entirely based on the predictions of the classification model as he technically can't see the painting, which means he can be wrong.



The user's objective is to decide whether Bob is bluffing or genuinely correct. After each guess, feedback is given to show whether Bob's prediction aligned with the ground truth. At the end of the game, a detailed score breakdown is presented to the user, showing the actual genre and style, the model's predictions, and the user's accuracy in detecting Bob's bluffs.



This concept takes advantage of the classification model's imperfections by turning them into a feature rather than a flaw. Much like a human with partial knowledge, the model makes confident but sometimes biased guesses, especially due to the label imbalance issues discussed earlier. By embracing this behavior, the model is used to simulate human-like reasoning, turning the game into a light-hearted commentary on AI bias and subjectivity in art interpretation.

5.1 Technologies Used

The application was built using the following technologies:

- **Flask (Python)**: Serves as the backend framework. It loads the trained fine-tuned ResNet-50 classifier and a local language model: TinyLlama, handles predictions and generates text descriptions.
- **HTML/CSS with Inline JavaScript**: The frontend is entirely handcrafted, with simple responsive layouts, interactive buttons, and sprite-based animations. JavaScript is used for rendering logic, transitions, and managing the game state.
- **Sprite Animation**: To add charm and personality to the game, a hand-drawn sprite-based animation system is implemented directly in the HTML interface. All the buttons are also hand made.

All model predictions are performed on real test or validation data samples, ensuring that users are interacting with genuinely challenging examples. This allows the game to serve both as an entertaining experience and an educational illustration of model behavior.

5.2 Language Model Integration

To generate Bob's descriptions, a local language model was integrated into the application: **TinyLlama-1.1B-Chat-v1.0**. The chat-specific version of the model was selected deliberately, with the idea that it would better simulate a conversational scenario, reflecting the natural dynamic of a visitor asking questions to a tour guide.

The setup is straightforward: the genre and style predictions from the classifier are used as input conditions for the LLM, which then generates a short description of the painting. However, to control the quality and tone of the outputs, a set of carefully crafted rules and formatting constraints were enforced. These were developed iteratively through trial and error, experimenting with prompt design and output patterns to achieve a believable and coherent voice for Bob.

In addition to prompt engineering, a number of post-processing steps were implemented to clean up the model's responses. These include filtering unwanted tokens, truncating when necessary, and hard-coded safeguards to avoid leakage from the original prompt or formatting artifacts.

Despite these efforts, integrating the language model posed several challenges:

- The model would often ignore the formatting rules, injecting irrelevant or unexpected content.
- Leakage from the prompt was frequent, especially when using higher temperatures or longer outputs.
- Attempts to correct this through stricter prompts or formatting constraints sometimes made the outputs worse, breaking fluency or leading to repetitive phrasing.
- The model having only the genre and style to work with make results unpredictable and often quite inaccurate.

One major factor in choosing TinyLlama was **speed**. Since the application is interactive, quick responses are essential to keep the user experience smooth and immersive. Several models were tested beforehand, but response times proved impractical for real-time interaction:

- **Mistral 7B (via Interpret 0.3):** $\sim$3 minutes
- **Phi-4:** $\sim$9 minutes
- **Phi-2:** $\sim$4 seconds
- **TinyLlama:** $\sim$3 seconds

TinyLlama ultimately provided the best balance between responsiveness and local deployability. While larger models may have generated more coherent or nuanced responses, they were simply too slow for the intended use case.

TinyLlama delivers acceptable results given the time constraints, but fine-tuning or upgrading to a faster, higher-quality model in the future could improve both reliability and immersion. For now, the imperfections in the responses even add to the charm, making Bob feel like an imperfect but enthusiastic human guide, rather than a perfectly scripted system.

6 Conclusion

This project explored the task of predicting both the artistic style and genre of paintings using the WikiArt dataset. After cleaning and balancing the data, a series of models were trained and evaluated, starting from a simple CNN to a frozen ResNet-50, and finally a fine-tuned version of ResNet-50.

The fine-tuned model showed clear improvements, achieving strong results on both individual tasks and the more challenging joint prediction. The results confirmed that fine-tuning a pretrained model is very effective for this kind of visual classification problem. At the same time, the evaluation showed some important limitations, especially regarding the imbalance in the joint distribution of labels, which led to biased predictions for less common genre-style combinations.

Instead of seeing these flaws as purely negative, the project took a creative turn by turning them into a feature. An interactive web application, *Bob's Bluffing Gallery*, was developed to showcase how the model behaves in a fun and engaging way. In this game, users meet Bob, a friendly but sometimes wrong gallery guide, who tries to describe paintings using only the model's predictions. The player has to guess whether Bob is bluffing or not.

The application is powered by TinyLlama, a lightweight local language model used to generate Bob's descriptions based on the classifier's predictions. The model was chosen for its fast response time, which is essential for a smooth game experience. Although larger models could generate better text, they were too slow for real-time use. TinyLlama offers a good balance between speed and quality, even if the results are not perfect and sometimes require extra tuning and filtering.

Overall, this project combines solid machine learning work with a creative application. It shows how even imperfect models can be used in meaningful and entertaining ways. There are many possible improvements for the future, including better prompt handling, fine-tuning the language model, or improving the joint label balancing. Improvements can also be made to the application itself, with new functionalities such as using an image-to-text model to get a precise description of the painting. This output could then be used to determine what the painting is and provide the user with insightful descriptions, rather than relying on our clumsy Bob.

But even in its current state, the system is both a useful classifier and a fun tool that invites people to explore how AI sees art.

References

[1] HuggingFace WikiArt Dataset. *https://huggingface.co/datasets/huggan/wikiart*

[2] PyTorch ResNet-50 Model Documentation.
https://pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html

[3] Mistral-7B-Instruct-v0.3. *https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3*

[4] Microsoft Phi-4. *https://huggingface.co/microsoft/phi-4*

[5] Microsoft Phi-2. *https://huggingface.co/microsoft/phi-2*

[6] TinyLlama-1.1B-Chat-v1.0. *https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0*

[7] WikiArt Classifier GitHub Repository. *https://github.com/mbellitti/wikiart-classifier*