Building a Neural Historical Art Classifier with GLCM Texture Features and Attention Feature Extraction

Eish Maheshwari
Department of Computer Science
Stanford University
eishm@stanford.edu

Michael Y. Cao
Department of Computer Science
Stanford University
caom@stanford.edu

Abstract

Evaluating historical art is expensive and time-consuming since it requires both contextualized objective information and subjective opinion from professionally-trained appraisers. We implement a novel fine-tuned deep learning model to predict art style from digital painting images. We address class imbalances and over-fitting in artwork data through multiple data augmentation strategies, account for intra-class variation using custom texture-based features, and leverage deep discriminative representation learning with attention mapping (DDRL-AM) to better learn inter-class dissimilarities. Our models use a fine-tuned ResNet50 with geometric image transformations, texture features computed with gray-level co-occurrence matrices (GLCMs), and DDRL-AM to improve accuracy by 6.6% compared to the best previously published benchmarks on this task. We also learn that attention mapping with Grad-CAM does not add significant predictive power in determining art style.

1. Introduction

As art becomes rapidly digitized, there is an increasing need to extract information to analyze and classify art pieces [22]. Evaluating historical art is especially challenging since it requires both contextualized objective information and subjective opinion. Relying on professionally-trained appraisers is expensive and time-consuming. By training a deep learning model on digital artwork images, the evaluation process can be streamlined. In this project, we aim to efficiently classify art style/genre, as this provides significant contextual information for each piece.

This problem is especially difficult due to historical artwork having large intra-class variation and small inter-class differences, which is noted in work by Merchant et al [18]. For example, even within prominent art genres such as Baroque, each artist has their own individual style, and differences between genres like Early Renaissance and High Renaissance are subtle. Additionally, the amount of available artwork is highly dependent on the time period, resulting in a dataset with a heavy class imbalance. Existing modeling approaches use a handful of standard CNN baselines, but struggle to perform well. We aim to improve current modeling techniques with a three-pronged approach.

1. Fixing class imbalances and reduce over-fitting through data augmentation.
2. Using custom texture-based features to better account for intra-class variation.
3. Use deep discriminative representation learning with attention mapping (DDRL-AM) to better learn inter-class dissimilarities.

The input to our model is a digital image of a artpiece. We then use a modified-CNN architecture to output which one of 12 historical art styles/genres an image is from. Our experiments yield accuracy scores that outperform all published baselines on this task.

2. Related Work

2.1. Data Augmentation

Data augmentation is a powerful strategy for creating synthetic samples from existing training data. This helps reduce over-fitting and address large class imbalances. Previous work by Wong et al explored augmentation in the data space (raw pixel values) compared to the feature space (after CNN-extraction) and found a 27% lower error using data-space augmentation [21]. However, this was limited to the well-established MNIST handwriting classification task; it is unclear how augmentation generalizes to more complex artwork classification. We examine more recent work by Hong et al which using geometric transformations (ex: projection, rotation, scaling, etc.) to generate synthetic images of paintings with improved classification potential [10]. However, gains were minimal, only a 3% increase in accuracy, and thus, a more effective strategy is needed. Preliminary results

by Bianconi explored color-space augmentation in painting classification (ex: hue/chromatic shifts) and found virtually no benefit despite significant computational overhead [5].

2.2. Gray-Level Co-Occurrence Matrices (GLCM)

GLCM is a statistical approach that gives information about positions of pixels with similar gray level values. GLCMs were first introduced by Haralick as a $N \times N$ matrix G where (i, j) of G represents the number of occasions a pixel with intensity i is adjacent to a pixel with intensity j [7]. Adjacency can be horizontal, vertical, or diagonal, and GLCMs can be generated for each of these adjacency directions. Cetinic et al used GLCMs to extract texture features from work by 20 different artists, which were then used to classify the painter and achieved 75% accuracy [6]. This helps address intraclass variation by learning higher level texture-based representations. However, there is limited work on combining high-level GLCM texture-features with deep neural methods. Recent work by Haryanto et al was some of the first to use GLCM with a 5-layer feed-forward neural network and observed a 6.9% increase in classification accuracy on colon histopathology images [8]. However, there is yet to be work on aggregating GLCM with CNN-extracted features, especially given the importance of textures in art classification.

2.3. Deep Discriminative Representation Learning with Attention Mapping (DDRL-AM)

DDRL-AM uses a fine-tuned CNN and Grad-CAM to generate attention maps, extract discriminative features from these maps using a spacial feature transformer (SFT) network [12], and combine these features with existing features to train another neural model. Li et al used DDRL-AM to address small inter-class dissimilarity and diverse objects/themes in satellite images. [17]

Saliency maps and Grad-CAM are both good tools for network visualization, as they highlight what parts of an image the model is focusing on, but Li et al [17] take inspiration from the attention mechanism [20] and use attention maps as a model input to increase the discriminative power of the model— especially useful for their remote-sensing use-case. With their task, the objects they are looking for only take up a fraction of the overall image, so attention can help the model focus on only the relevant parts of the image. They explain that in contrast to existing approaches that utilize attention mechanisms, their approach makes use of fine-tuning pretrained models such as AlexNet and ResNet and uses class-specific attention maps as a model input as supervisory signals for model training. Li et al. use a novel feature fusion method to combine features from both the attention maps as well as the last layer outputs of a traditional CNN, and they also define a new center-based cross entropy loss to better accommodate the fused features. This work was limited to object detection/classification in satellite images [17]. We hope to explore how attention mappings affect the art classification problem which does not have defined objects.

2.4. Prediction Task

Current state of the art methods on the art style classification task involve fine-tuning pretrained ImageNet models. Previous work by Howell fine-tuned a pretrained VGG11 CNN-based architecture to classify art style using the WikiArt dataset (fine art pieces spanning fifteen centuries) and achieved 48% accuracy on predicting art piece time-period [11]. However, using VGG11 was suboptimal due to a shallow feature representation which makes accounting for intra-and inter-class variation challenging. Kovalev et al fine-tuned ResNet with geometric data augmentation to classify painting style with 51.5% accuracy [15]. However, they only considered 5 different art styles. Thus, a more precise model that can distinguish between more classes is needed.

3. Methodology

3.1. Baseline Model Selection

We used transfer learning through a pre-training framework to architect our baseline, motivated by the success of Howell et al [11]. We evaluated the performance of three baseline parameter initializations (all from PyTorch): AlexNet, ResNet18, and ResNet50. AlextNet was one of the first deep CNNs to achieve considerable accuracy on the ImageNet challenge and popularizes the convolutional-normalization-pooling layer paradigm [16]. However, AlexNet had only 5 convolutional layers. ResNet expands on AlexNet with many more layers (ResNet18 has 18, ResNet50 has 50), and introduces skip connections to address vanishing gradients [9]. These baselines were chosen since they are designed to be fine-tuned for image classification, having been pre-trained on millions of images across 1000 ImageNet categories. For each model, we evaluated accuracy on our task after fine-tuning on artwork images. The best baseline was used in all future experiments.

3.2. Data Augmentation

To address overfitting from small sample size and class imbalances, we tried several geometric augmentation strategies on the training data space (implementations based on `torchvisions.transforms` [1]). We used geometric transformations since they preserve overall texture patterns to identify art style. We implemented horizontal/vertical flips, perspective shifts, random cropping, and rotations. We developed a hybrid-strategy where these techniques were randomly applied to each image. These transformations were applied on raw pixel data as recommended by Wong et

al [21]. We did not use any color-based strategies as Bianconi found minimal benefits in artwork classification [5]. Only classes with fewer examples were augmented up until there was no class imbalance. The augmented training data was used to fine-tune baseline, GLCM, and DDRL-AM neural models.

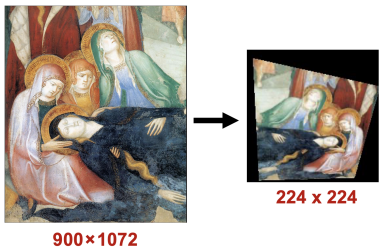

Figure 1. Example data augmentation pipeline with resizing and geometric transformations

3.3. GLCM Net

Texture features from GLCMs were generated from painting images to combine with deep CNN-features. This is motivated by the importance of textures in identifying a particular artwork style. Incorporating higher-level textures should help address intra-class variation since art of a certain style are likely to share textural features. We calculate multiple GLCMs to extract 6 features: dissimilarity, correlation, contrast homogeneity, ASM, and energy (implementation based off `scikit-image` [2]). Features were generated for each augmented image using the following formulas.

Texture feature	Equation
Contrast	$\sum_{i=1}^{N}\sum_{j=1}^{N}(i-j)^2 P(i,j)$
Entropy	$-\sum_{i=1}^{N}\sum_{j=1}^{N}P(i,j)\,lgP(i,j)$
Correlation	$\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(i-\bar{x})(j-\bar{y})P(i,j)}{\sigma_x\sigma_y}$
Energy	$\sum_{i=1}^{N}\sum_{j=1}^{N}P(i,j)^2$

Figure 2. GLCM feature computation. Each (i, j) represents pixels with intensity i adjacent to a pixels with intensity j [13]

After values for each GLCM feature were computed, they were concatenated into a single 6×1 GLCM feature vector. Rather than solely using GLCMs as features for deep models like previous work [8], we developed a novel GLCM-Net which incorporates GLCM features with deep features extracted from a fine-tuned CNN. The intuition is that GLCMs focus specifically on textures to address intra-class variation while CNN features will focus on other colors and patterns for a comprehensive analysis.

The CNN feature extractor is a pre-trained baseline ImageNet model with the final fully connected layer removed (downloaded from PyTorch [3]. This consists of several convolutional, batch-norm, and pooling layers combined in sequence to transform raw image pixels into a feature vector. It takes in a 224×224 image and transforms it into a $512 \times 7 \times 7$ feature vector. This feature vector can then be used for classification.

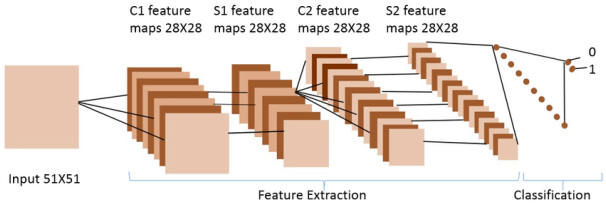

Figure 3. Example CNN feature extractor with final fully connected classification layer [14]

In GLCM-Net, images are simulatenously fed through both a pre-trained CNN and GLCM feature extractor to create two independent sets of features. Both of these output features are then flattened, concatenated, and linearly projected into a lower feature space (using a linear layer) to create a hybrid feature. This is then run through a 2-layer feedforward network with dropout and ReLU activation for classification. Our loss function (for all models) is cross-entropy loss:

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(\hat{y}_i)$$

where $y_i \in \{0, 1\}$ are true labels for the i-th class and $\hat{y}_i$ are predicted labels for the i-th class over all m examples.

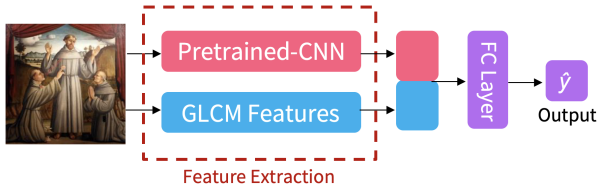

Figure 4. Architecture of the GLCMNet. The entire model is trained end-to-end where pre-trained weights are fine-tuned and FC layer is learned simultaneously.

3.4. DDRL-AM

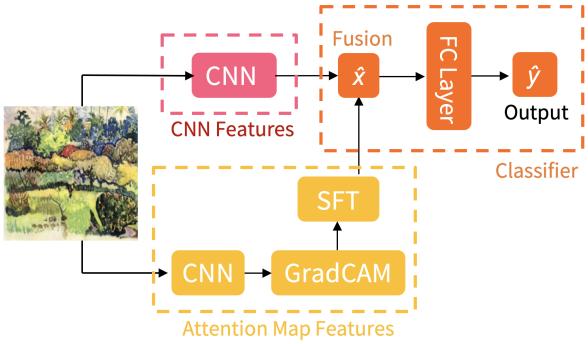

Figure 5. Architecture of DDRL-AM. We extract feature maps from the original image as well as attention maps and combine them via multiplicative fusion.

For our task, we follow Li et al [17] and generate features from attention maps as well as features from CNNs for each image. We follow three steps to generate these features:

1. We first fine-tune a pretrained ResNet model on our Historic Art dataset so that we can generate Grad-CAM attention maps from this model's weights and gradients.
2. We instantiate another pretrained ResNet model as well as a SFT model (which takes the fine-tuned ResNet model as an input) and extract features from the RGB stream and the attention maps, respectively.
3. We fuse the features with multiplicative fusion, and we feed the fused features into a linear classification layer.

3.4.1 Grad-CAM Attention Maps

Grad-CAM uses the gradients of a specific class to produce a saliency map that highlights important regions of the image. We cut our fine-tuned ResNet model off right before the final fully-connected layer so that it outputs K feature maps of size $W \times H$. From this, we calculate the total gradient flow for the k-th channel feature map for the c-th class.

$$a_k{}^c = \sum_i \sum_j \frac{\partial y^c}{\partial a_i j^k}$$

We then combine the k feature maps and use ReLU to remove negative values to receive our final attention map:

$$M = RELU(\sum_k a_k{}^c A^k)$$

Note that the final attention map is of dimension 7×7, so we upsample the map to the original image size of 224×224.

3.4.2 SFT Model

The spatial feature transformer, or SFT, is designed to extract features from the attention maps. We model our SFT after Li et al [17] and use a combination of convolutional layers and batchnorm layers (to reduce overfitting). The final output shape matches that of the penultimate layer of ResNet. The implementation details are listed in the table below.

Table 1. SFT Architecture Settings

layer	num	kernel size	stride	padding
Conv2D	64	7	2	3
BatchNorm	64	-	-	-
MaxPool2d	-	3	2	1
Conv2D	128	3	2	1
BatchNorm	128	-	-	-
Conv2D	256	3	2	1
BatchNorm	256	-	-	-
Conv2D	512	3	2	1
BatchNorm	512	-	-	-

3.4.3 Multiplicative Feature Fusion

The outputs of both the truncated CNN and the SFT are $(N, 512, 7, 7)$. Following Li et al [17], we perform multiplicative fusion to combine the two feature maps:

$$\text{out}_d{}^{\text{fused}} = \text{out}_{i,j,d}{}^{\text{cnn}} \times \text{out}_{i,j,d}{}^{\text{attn}}$$

A way to interpret multiplicative fusion is we take the dot product between the 7×7 mini feature maps from the pretrained CNN and the SFT. The final output is the same shape as the two inputs, which is $(N, 512, 7, 7)$.

3.4.4 Classifier

After we generate the fused feature map, we pass the map through an average pooling layer and then finally a fully connected layer to compute the class output probabilities.

4. Experiments/Results

4.1. Dataset and Features

Our classification task uses the Historic Art dataset, which contains 45.6k images of European artworks from the 3rd to 19th century, gathered from the Web Gallery of Art [4]. We chose this dataset because it includes both images and rich metadata. There are 12 classes, each corresponding to a major European art period.

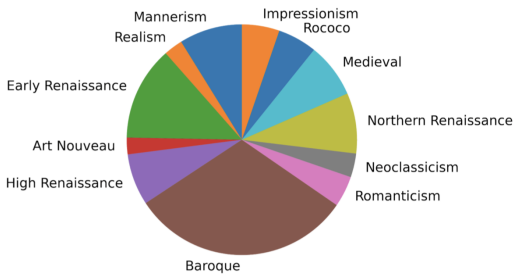

Figure 6. Proportion of each class (art style) in the training data set

After data augmentation, we had a total of 60k images, with 5k images per class to eliminate the class imbalance. The labeled data was divided into a train, dev, and test set with an 90-5-5 split resulting in 53,000 train images, 3000 dev images, and 3000 test images. To feed our images into the pretrained baseline models, we resized them to 224x224, normalized pixel values to be in range $[0, 1]$, and ensured all images were in RGB. Features were computed using the CNN/GLCM methods described previously.

4.2. Evaluation Methods

Previous art classification work by Kovalev et al, Li et al, and Howell et al all use accuracy as the main evaluation metric [15] [11] [17]. Therefore, we use accuracy as the main quantitative metric. In addition, we examined confusion matrices to evaluate model precision and recall. Qualitatively, we used Grad-CAM to highlight important regions in an image for predicting a given concept [19]. We also visualized and analyzed model weights for fully connected layers to examine feature importance. Lastly, we looked at common qualitative errors across different models.

4.3. Experimental Details

4.3.1 Baselines and Data Augmentation

Previous work on deep CNNs for art classification found best results using the Adam optimizer [18] [15]. Thus, we used Adam to train for 10 epochs with a batch size of 100, which is the maximum size we could fit on our machine. We adjusted the number of epochs needed to train until loss plateaued, using manual early stopping as needed. In addition, we performed a hyperparameter grid search to find optimal learning rate (3e-5).

4.3.2 GLCM

We used a similar strategy to tune hyperparameters of GLCM Net using the augmented dataset, using Adam with a batch size of 100 and learning rate of 3e-5. In addition to the parameters already discussed, we also tuned the optimal L2 regularization / weight decay (1e-2), and size of projection/hidden layers after feature extraction (512) through a grid search. The dropout rate for the feedforward layer was modeled after the original ResNet ($p = 0.2$) [9].

4.3.3 DDRL-AM

Similar to GLCM, to train our DDRL-AM model we use the Adam optimizer with a batch size of 100. For our hyperparameters, we started off with the same parameters that were optimal for the GLCM model, which were learning rate of 3e-5 and weight decay of 1e-2. We experimented with a few different weight decay values to test different regularization strengths (which is further explained in our results section) and found that 1e-3 yielded the best results.

4.4. Results

4.4.1 Baseline Model Selection

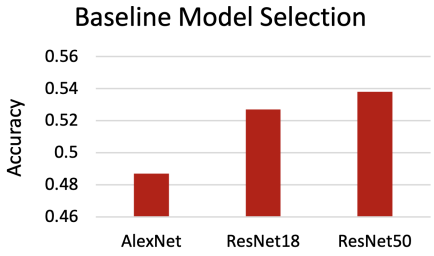

Figure 7. Performance of fine-tuned baseline models

ResNet50 and ResNet18 had similar accuracy (0.538 and 0.527 respectively), which both performed better than AlexNet (0.487). While current benchmarks by Kovalev performed comparably, they predicted on only 5 classes whereas our model achieved this accuracy over 12 different classes [15]. Thus, our model showed improved performance over the current best baseline. ResNet performed better, likely since it is a deeper model, uses batch normalization, and uses short skip connections to reduce overfitting while having a deeper, more nuanced network. Thus, we chose ResNet50 as the backbone for all future models (i.e. for DDRL-AM and SFT).

4.4.2 Data Augmentation

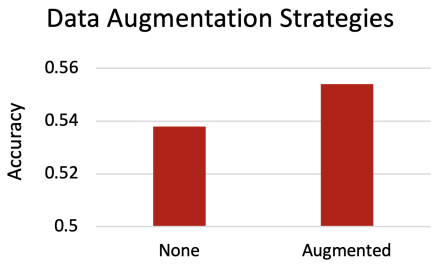

Figure 8. Performance of data augmentation on fine-tuned ResNet50

Using data augmentation had a 2.97% increase in dev accuracy on a fine-tuned ResNet50 (baseline) compared to no augmentation, even though train accuracy went down by 1.43%. This gave a final accuracy of 0.554. This suggests that adding geometric noise through transformations reduced over-fitting and helped address initial class imbalances. We noticed when training that the loss would first decrease then began to increase again. To combat this, we decreased the learning rate to 1e-5 and increased the number of epochs to 30 which fixed the optimization issue. We still noticed discrepancies between train and dev accuracy, so we opted to add stronger weight decay (L2 penalty to the loss) for future models to further reduce over-fitting.

4.5. GLCM

Projection Layer Size	Weight Decay	Dev. Accuracy
64	1e-2	0.517
256	1e-2	0.541
512	1e-1	0.522
512	**1e-2**	**0.549**
512	1e-3	0.513
512	1e-4	0.488
1024	1e-2	0.548

Table 2. Hyperparameter tuning of projection layer size and weight decay rate

Ideal performance was found with a projection layer size of 512 and weight decay of 1e-2. Across the board, we saw heavy evidence of over-fitting with dev accuracies ≈ 0.5 while train accuracies were closer to ≈ 0.7. As a result, adding weight decay helped reduce over-fitting. Reducing projection layer size below 512 reduced expressivity, causing accuracy to decrease. This is likely since important information features are lost when reducing the $512 \times 7 \times 7$ CNN feature into a smaller dimension space.

The best GLCM-Net model gave a 0.9% reduction in accuracy compared to the augmented ResNet50 model. Overall, there is minimal improvement on performance, suggesting that concatenating GLCM texture features with CNN-features does not reveal new associations that increase predictive power. This is likely due to the high dimensionality of CNN-features (512) over-powering the low dimensionality of GLCM features (6), resulting in minimal benefit from the feature.

To further analyze why GLCM features did not add predictive power, we analyzed weights of the fully connected layer directly after concatenation of GLCM and CNN features. The weight matrix W for this linear layer was partitioned into W_{CNN} and W_{GLCM} which correspond to the columns which contribute to the CNN (x_{CNN}) and GLCM (x_{GLCM}) components of the concatenated feature respectively (see figure below). To determine how CNN and GLCM features were weighted, we took the sum of the absolute value of each individual weight, and normalized by the size of the corresponding weight matrix. This showed, on average, how large in magnitude the weights for CNN and GLCM features were, in other words, how the feedforward layer differently weights CNN and GLCM features. We computed this average impact (I) of each $n \times m$ weight matrix as

$$I = \frac{1}{nm}|W|_1$$

where $|W|_1$ is the L1 norm. W_{CNN} had an average impact of 0.0175, whereas W_{GLCM} had an average impact of 0.00071, meaning GLCM features were weighted only 4.1% as much as CNN features. This further supports that GLCM features did not extract new texture information that the CNN-feature extractor did not already know.

Figure 9. Fully connected layer after CNN and GLCM feature concatenation, showing how much CNN and GLCM features are being weighted.

4.6. DDRL-AM

Our best performing DDRL-AM model after hyperparameter tuning achieved a dev accuracy of 0.402—this model used a learning rate of 3e-5 and a weight decay of 1e-3.

Weight Decay	Dev. Accuracy
1e-1	0.395
1e-3	**0.402**
5e-4	0.386

Table 3. DDRL-AM Hyperparameter tuning of weight decay

We noticed that DDRL-AM models achieved very high train accuracies, much higher than other models, within a few epochs. However, the overall dev accuracy was still lower than our augmented baseline. This suggests heavy over-fitting, likely due to the dramatic increase in number of parameters. We tried adjusted regularization strength through weight decay and dropout of the feedforward layer; however, this just reduced overall accuracy. Additionally, because even our augmented fine-tuned ResNet model only had an accuracy of 0.554, the attention maps generated through GradCAM from this model were not the best at highlighting the most salient parts of images. Because DDRL-AM depends on these scores, this may have amplified errors, resulting in suboptimal performance.

5. Analysis

5.1. Final Model Error Analysis

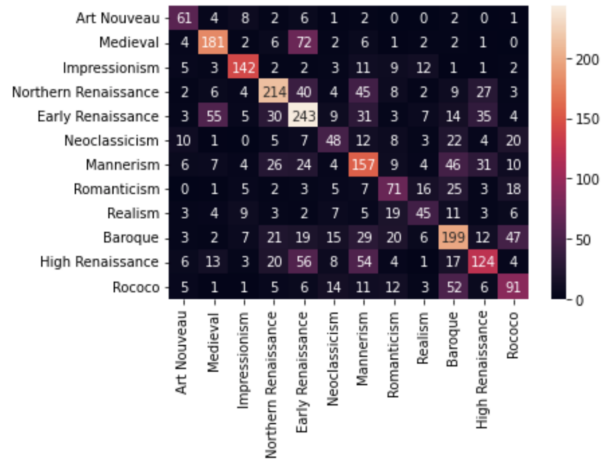

Figure 10. Confusion matrix for best model. The Y axis represents the true label, while the X axis represents the predicted label. Predictions represented as counts on a total 3000 sample dev set. Overall accuracy for this model was 0.554.

Our best performing model (accuracy: 0.554) used a fine-tuned ResNet50 baseline with geometric data augmentation only. Looking at the confusion matrix, the rows represent false negatives and the columns represent false positives (excluding diagonal entries which are true hits). Examining the rows, we see that some periods are classified quite accurately, such as Art Nouveau and Impressionism, while art from the different Renaissance periods, Mannerism, and Rococo are often misclassified. If we examine the columns, we see the most false positives with the Renaissance genres once again, with Mannerism, Baroque and Rococo following close behind. This makes sense historically as artwork during the different Renaissance periods, and between the Rococo and Baroque period, were quite similar. The similarity in time-period explains why the model frequently confused them. Additionally, the interclass dissimilarities between these genres are subtle, so it is hard to visually distinguish between them. Thus, the low overall accuracy of our models was likely since the expressive power of our models was not enough to correctly distinguish these more difficult genres.

5.2. Grad-CAM

We will examine model predictions using Grad-CAM visualization on the best performing model, a fine-tuned ResNet50 with data augmentation. We analyzed which pixels the model was focusing on for both correctly and incorrectly classified samples.

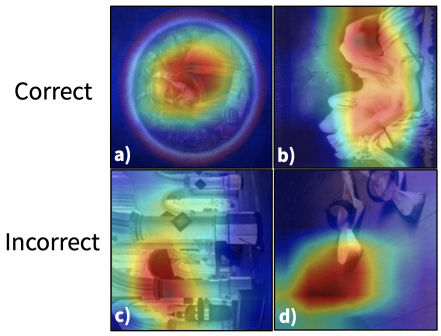

Figure 11. GradCAM visualizations for fine-tuned ResNet50 w/ augmentation. **a)** Medieval **b)** Early Renaissance, **c)** Predicted: High Renaissance, Actual: Baroque, **d)** Predicted: Neoclassicism, Actual: Northern Renaissance,

The model performs well when there is a clear, defining object that can be attributed to a style, such as the people in a and b. However, when there are general patterns, such as c and d, the model does not know what to focus on. Because ResNet was pre-trained on the ImageNet task, it performs well in classification tasks that involve identifying and locating objects. However, because art style classification requires analyzing overall patterns/themes, rather than specific objects, these baselines may not have been optimal. Out of all the correctly predicted samples, the most common label correctly classified was Medieval. Images in this art style typically contain a few objects with a simple background, for instance, portraits. Therefore, it makes sense that our model performs well on this class because the ResNet baseline is best for identifying objects, as evident by GradCAM visu-

alizations. Overall, we see that focusing attention on parts of an image is not ideal for genre classification. It is more beneficial to look at the entire piece as a whole to analyze broader textural and color patterns.

5.3. Augmented ResNet vs DDRL-AM

We will examine specific discrepancies between the augmented fine-tuned ResNet50 vs DDRL-AM models to analyze the effect of GradCAM-based attention.

Image 1: DDRL-AM Correct, Data Aug Incorrect

Figure 12. Original image shown next to Grad-CAM visualization.

Label: Northern Renaissance
DDRL-AM Prediction: Northern Renaissance
Data Aug Prediction: Early Renaissance

This image was correctly classified by the DDRL-AM model but incorrectly by the data aug model. There is a clear, defined object (a person), which is attended to, as seen in the Grad-CAM visualization. This further shows that attention mapping is beneficial to focus on specific image regions indicative of an art style (ex: a portrait). Especially since Early Renaissance and Northern Renaissance are very similar styles, the extra attention on the area of interest helps DDRL-AM correctly classify the image, which the data aug model cannot do.

Image 2: Data Aug Correct, DDRL-AM Incorrect

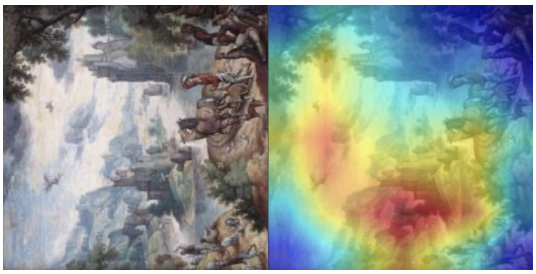

Figure 13. Original image shown next to Grad-CAM visualization.

Label: Mannerism
DDRL-AM Prediction: Northern Renaissance
Data Aug Prediction: Mannerism

This image was correctly classified by the data aug model but incorrectly by DDRL-AM model. This is a complex image with several objects, backgrounds, and shapes. Thus, it is hard to attend to a specific region. The Grad-CAM reveals the model struggling to focus on a relevant portion. Thus, attention scores in DDRL-AM may further confuse the model, causing it to make an incorrect prediction. The simplicity of the data aug model allows it to be correct.

6. Conclusion/Future Work

We saw that augmenting our dataset with geometric transformations led to a 6.6% increase in accuracy from the previous best published model [15], with an overall accuracy of 0.554. This was our best model built upon a fine-tuned ResNet50 baseline. Furthermore, we saw that GLCM texture features did not improve model performance, suggesting that the 6 texture characteristics do not add predictive power to existing CNN-extracted features.

Most notably, we saw a significant performance decline from our novel DDRL-AM framework which uses Grad-CAM scores to attend to specific image regions. By analyzing confusion matrices and Grad-CAM visualizations, we learned that attention mapping is beneficial when there is a clear, defining object indicative of an art style. Otherwise, the model attends to regions with little predictive power, which only confuses it. Because historical artwork heavily emphasizes overall textures and color patterns, rather than detecting objects, attending to specific regions is not optimal to better learn inter-class dissimilarities.

In the future, we would like to use a different pre-trained baseline, one not focused on object recognition, to better account for textural and color patterns. Because DDRL-AM was heavily impacted by over-fitting, we would also like to try more advanced data augmentation strategies such as kernel filters, mixing image, and random erasing. Finding a larger dataset would also be beneficial. Lastly, we can improve the classification layer of GLCM and DDRL-AM models beyond a simple fully-connected network and implement additional measures to guard against overfitting.

7. Contributions/Acknowledgements

Eish worked on developing the idea behind the GLCM features and the network architecture for integrating these features into the classifier. This involved modifying a pre-trained PyTorch CNN model with new features. Additionally, he wrote code for experimenting with numerous different data augmentation techniques, implemented the modified NN architecture, and ran hyperparameter-tuning experimeents. Michael worked on developing and implementing the baseline model and training infrastructure as well as

the DDRL-AM technique. He also created a novel feature fusion technique by training models, saving weight parameters, and developing attention mechanisms. He implemented Grad-CAM visualizations for qualitative metrics. He also analyzed different baseline models to find optimal parameter initializations. All of us contributed equally to the proposal, milestone, paper, and poster. We have no project sharing, and as far as external collaboration, to generate and visualize GradCAM saliency maps we used the GitHub repository pytorch-grad-cam (https://github.com/jacobgil/pytorch-grad-cam).

References

[1] Transforming and augmenting images. *PyTorch*, 2017. 2

[2] Glcm texture features. *scikit-image*, 2020. 3

[3] Models and pre-trained weights. *https://pytorch.org/vision/stable/models.html*, 2020. 3

[4] Historic art. *Kaggle*, 2022. 4

[5] Francesco Bianconi. Experimental analysis of colour constancy and colour augmentation for painting classification by artistic genre: preliminary results. *IOP Conf. Ser.: Mater. Sci. Eng*, 2020. 2, 3

[6] Eva Cetinic and Sonja Grgic. Automated painter recognition based on image feature extraction. 2013. 2

[7] Robert Haralick, Kumarasamy Shanmugam, and Itshak Dinstein. Textural features for image classification. *IEEE Trans. on Systems, Man and Cybernetics*, SMC–3(6):610–621, 1973. 2

[8] Toto Haryanto, Adib Pratama, Heru Suhartanto, Aniati Murni, Kusmardi Kusmardi, and Jan Pidanic. Multipatch-glcm for texture feature extraction on classification of the colon histopathology images using deep neural network with gpu acceleration. *Journal of Computer Science*, 2020. 2, 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. 2, 5

[10] Yiyu Hong and Jongweon Kim. Art painting identification using convolutional neural network. *International Journal of Applied Engineering Research*, 2017. 1

[11] Blake Howell. Using convolutional neural networks to predict completion year of fine art paintings. 2017. 2, 5

[12] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuogl. Spatial transformer networks. *Google DeepMind*, 2016. 2

[13] Sutasinee Jitanan and Pawat Chimlek. Quality grading of soybean seeds using image analysis. *International Journal of Electrical and Computer Engineering (IJECE)*, 9, 2019. 3

[14] Mina Khoshdeli, Richard Cong, and Bahram Parvin. Detection of nuclei in he stained sections using convolutional neural networks. 02 2017. 3

[15] Valentin Yu. Kovalev and Alexei G. Shishkin. Painting style classification using deep neural networks. pages 334–337, 2020. 2, 5, 8

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. 25, 2012. 2

[17] Jun Li, Daoyu Lin, Yang Wang, Guangluan Xu, Yunyan Zhang, and Chibiao Ding. Deep discriminative representation learning with attention map for scene classification. 2020. 2, 4, 5

[18] Daniel Merchant, Sean Yang Bum Mook Oh, Bill Howe, and Jevin West. Classifying digitized art type and time period. *University of Washington Seattle*, 2018. 1, 5

[19] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. 5

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2

[21] Sebastien Wong, Adam Gatt, Victor Stamatescu, and Mark McDonnell. Understanding data augmentation for classification: when to warp? *Computational Learning Systems Laboratory*, 2016. 1, 3

[22] Wentao Zhao, Dalin Zhou, Xinguo Qiu, and Wei Jiang. Compare the performance of the models in art classification. *PLOS ONE*, 2021. 1