Art Movements Classification

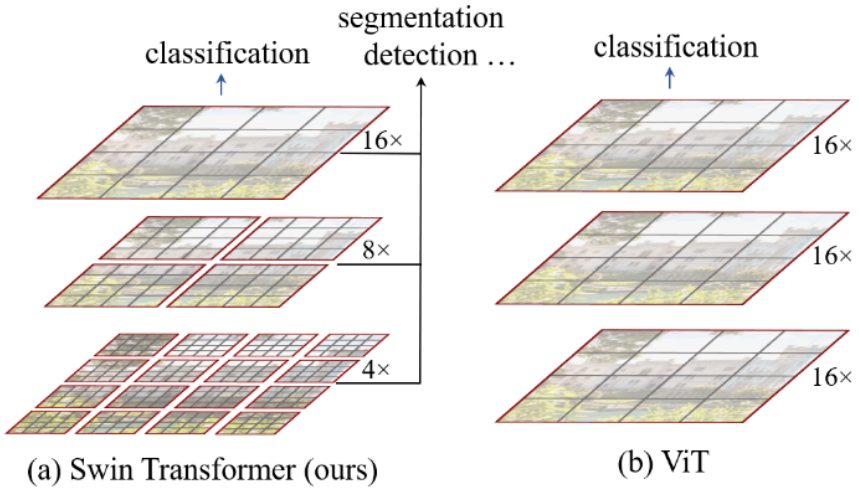
Neema Jafari (neemaj@uci.edu), Karissa Pratt (krpratt@uci.edu), Ian Wang (wangin@uci.edu), Andre Wu (andrecw@uci.edu)

Problem Statement

For our group project, we will be using a Swin Transformer to help classify the movement/style of an art piece given an image of it at any resolution. This model can be helpful in art education and analysis as students, critics, and ordinary people can gain deeper insight into their favorite works. Much of art is a conversation between artists in different time periods. Knowing the movement and style of any given piece places it in the wider span and evolution of art as a whole.

Citations

While searching for a suitable image classifier, we consulted with a TA, who directed us to Swin Transformers, mentioning how it would be a good fit for our project. Swin Transformers were first introduced in 2021, in the paper “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”(Liu Z. et al.,) to address the quadratic running time with respect to image size that Vision Transformers have (Dosovitskiy, A. et al.). The reason for Vision Transformer’s quadratic running time is mostly because of how a Transformer’s attention mechanism is applied to images. In the process of learning spatial relationships, each pixel attends to every other pixel in the image which gives Vision Transformers its quadratic running time. The Swin Transformer paper addresses this by limiting the self-attention mechanism to non-overlapping windows containing patches that gradually merge as the image progresses deeper through the architecture. This change gave the architecture linear running time, a major performance improvement over Vision Transformers.



A visual representation of the Swin and Vision Transformer’s (ViT) hierarchical feature maps and self-attention mechanism (Liu Z. et al.)

The group also referenced “Attention in Transformers”(Sanderson), a YouTube video that was important for members to understand the attention mechanisms in Swin and Vision Transformers.

Additionally, when writing the training loop we found it useful to reference PyTorch’s article on this subject called “Training with PyTorch” (Luis, J).

Data

For our dataset, we turned to Kaggle where we found Sivar Azadi's "WikiArt Art Movements/Styles." This dataset consisted of 42.5 thousand total jpg files with art labeled as 1 of 13 different movements. These movements were Academic Art, Art Nouveau, Baroque, Expressionism, Japanese Art, Neoclassicism, Primitivism, Realism, Renaissance, Rococo, Romanticism, Symbolism, and Western Medieval. Works varied wildly in terms of size and aspect ratio, so we resized all images as a baseline in our preprocessing. Above are examples of resized art from each movement to paint a better picture of our dataset.

Methodology

Throughout most of our experiments, we fine-tuned Microsoft's "swin-tiny-patch4-window7-224" model using the Timm library. This model was initially trained on the ImageNet-1k data set.

The constant hyperparameters used that we didn't experiment on were as follows. We used a random seed of 99 so we would have reproducible results. The train, validation, and test data split we used on our dataset was 80-10-10. We used cross-entropy loss as our loss function and used the AdamW optimizer as our preliminary research suggested they were good options for fine-tuning a Swin Transformer. We also used a batch size of 32 and kept our weight decay at 1e-4. Additionally, all of our runs had the base data augmentation step of resizing to 224x224 pixels so they would be accepted by the pre-trained model we were using.

During some of our experiments, we performed a horizontal flip on a portion of our images. A horizontal flip is the least destructive data augmentation that reduces overfitting by changing the direction people are looking, for example. We also had experiments which utilized "Medium Augmentation" and "Heavy Augmentation," as depicted in the table found in the Results section.

Our medium augmentation experiment consisted of adding padding around the image to prevent a black border. 25% of the images got distorted ±10% by changing the viewing perspective. Following this, images were rotated ±4 degrees, sheared ±4 degrees, and scaled

±5%. At this point, half of the images were horizontally flipped. The images then had their brightness value changed by ±3%, contrast by ±3%, saturation by ±3%, and hue by ±2%. Lastly, the image is brought down to 224x224 dimensions through a center crop.

The heavy augmentation experiment was modelled after the medium augmentation experiment but with drastically changed values. After the resizing and padding, images had a 49% chance to be distorted by ±20%, then were rotated ±10 degrees, sheared ±10 degree, scaled ±10%, horizontally flipped half the time, brightness adjusted by ±7%, contrast by ±7%, saturation by ±7%, then hue by ±5%. A further explanation of each specific augmentation is provided below.

Data Augmentations

Pad - Adds a border to the image, increasing its size. "Reflect" mode causes the border to be created by mirroring the edges of the image. This removes any potential black borders around the image that could skew results.

Geometric Transformations

Center Crop - Some transforms change the size of the image. This function crops the picture back to the original 224x224 size, allowing the Swin Transformer to accept the data.
Resize - Scale images to set dimensions. 224x224 was the standard we used. Each image had to be uniformly sized for the model to accept it as valid input.
Random Perspective - Shears the image by moving the corners, as if the image was projected onto a trapezoid or rhombus. By moving the image around, the model doesn't overfit to faces that look identical and in the same spot, for example.
Random Affine - Can rotate, shear, and scale the image by random amounts. This similarly prevents overfitting by moving the image slightly.
Random Horizontal Flip - A percentage of the images are flipped horizontally. This makes it so the model doesn't get biased towards people facing to the left, for example.

Color-Based Augmentations

Color Jitter - Randomly adjusts the brightness, contrast, saturation, and hue of the image. This eliminates the model from drawing connections between specific colors and styles of images. For example, most of the images from the Japanese style feature an off-white background that the model would be biased towards.

Examples of images from the "Medium Augmentation" trial.

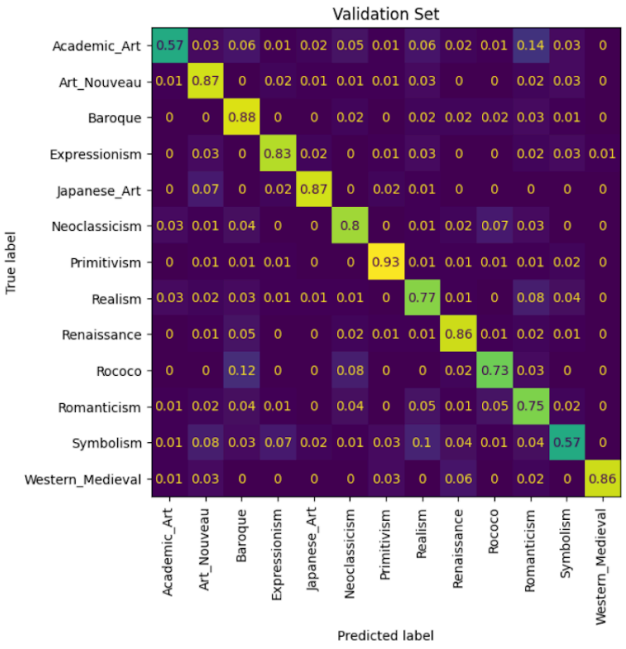


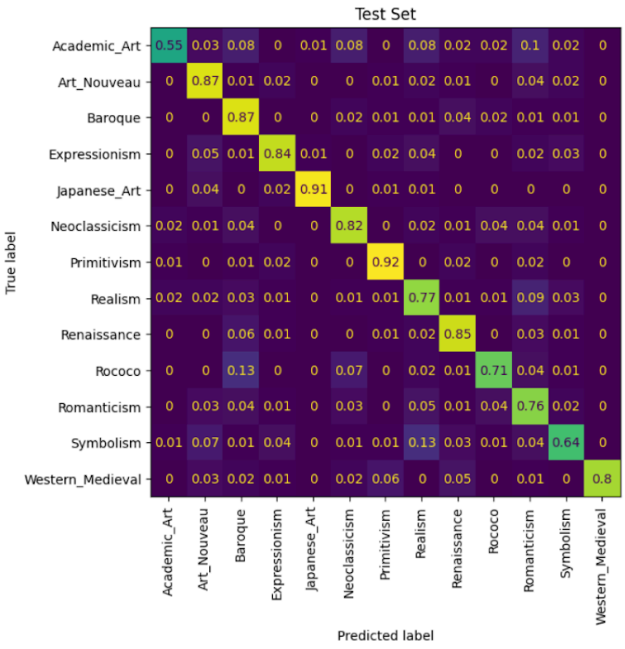
Examples of images from the "Heavy Augmentation" trial.



Experience in coding it up

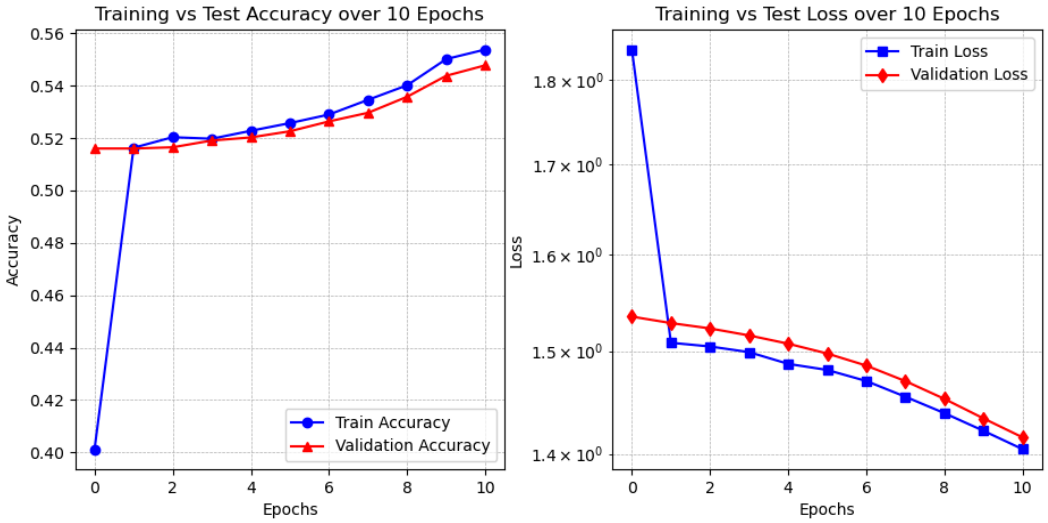
After arranging the image dataset and getting the Swin Transformer to work, the first run we did was on a pre-trained model with no adjustments to it. We were tuning every stage of the transformer as well as the head classifier and used a learning rate of 5e-5, which is small enough to not drastically affect the pre-trained model. Without any data augmentation, the model overfits. Below is the confusion matrix from this first run. The model was noticeably bad at classifying images of the Symbolism style, most prominently confusing it with images of the Realism style.

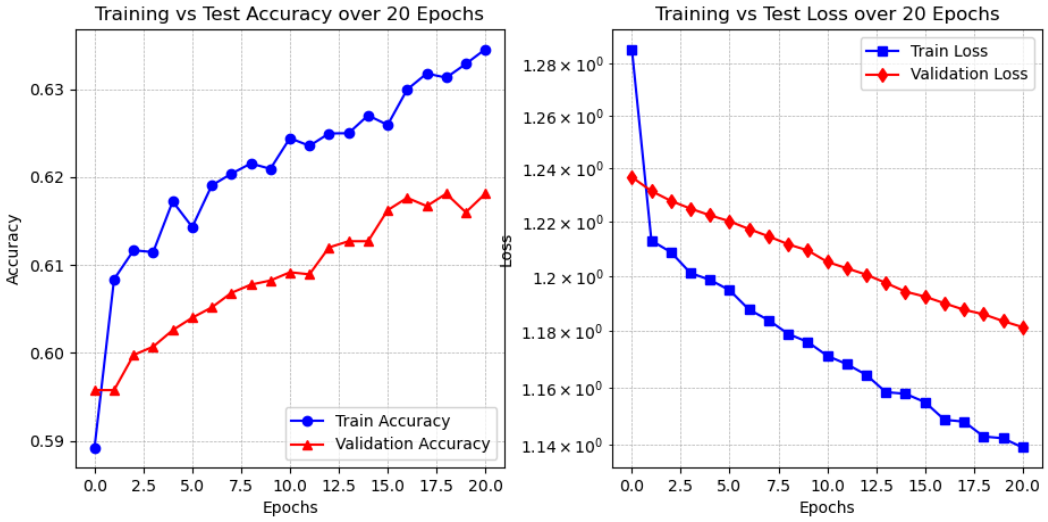




After this initial run, we experimented with freezing all stages of the model aside from the head classifier. We figured training only the head would be efficient as we don't overwrite the pre-trained model's knowledge, and instead spend the computing power tuning the model to our specific dataset. Unfortunately, training only the head led to underfitting and the model trained exceedingly slowly, which didn't allow us to wait until the model converged. We first ran the

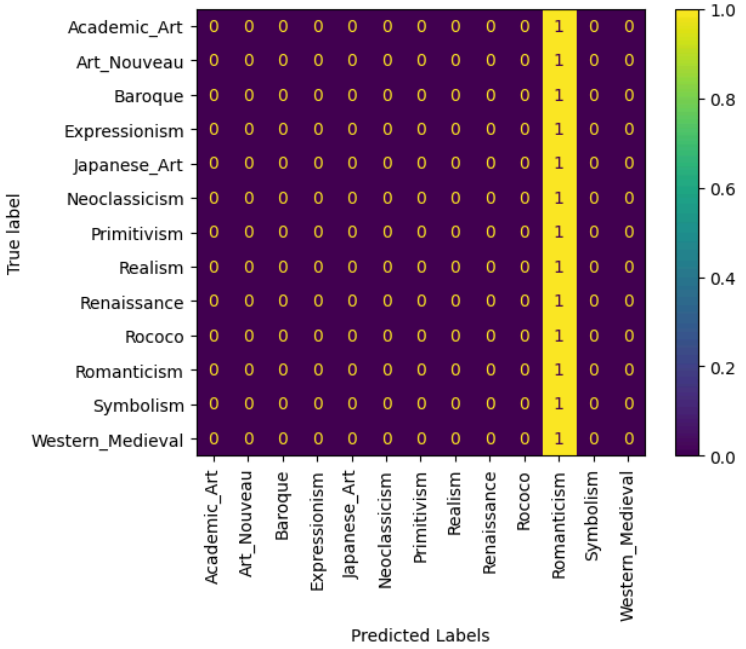
model for 10 epochs before stopping the run. After we noticed that the validation and test accuracy have not yet converged, we ran the model again using the .pth file for 20 more epochs. The graphs below show that the validation and test accuracy have not converged after 30 epochs, at which point we stopped training the model.





At this point, we decided that it may be best to train the model from scratch with our dataset of 42.5k images, allowing the model to be perfectly tuned to our needs and not have any unrelated holdover information from the pre-trained model. We allowed the model to run without any pre-training for a day, but realized that we did not have enough time nor data to train a classification model from scratch. We learned that it is better to fine-tune from a pre-trained model, as it allows the model to focus on the aspects of our dataset, whereas training the model from scratch requires the model to learn basic patterns in images from nothing. As seen in the confusion matrix below, the model got stuck in a local maximum where the model would

exclusively vote Romanticism. This is because the Romanticism group has twice the number of pictures as any other group, so the model would always be 16% accurate.

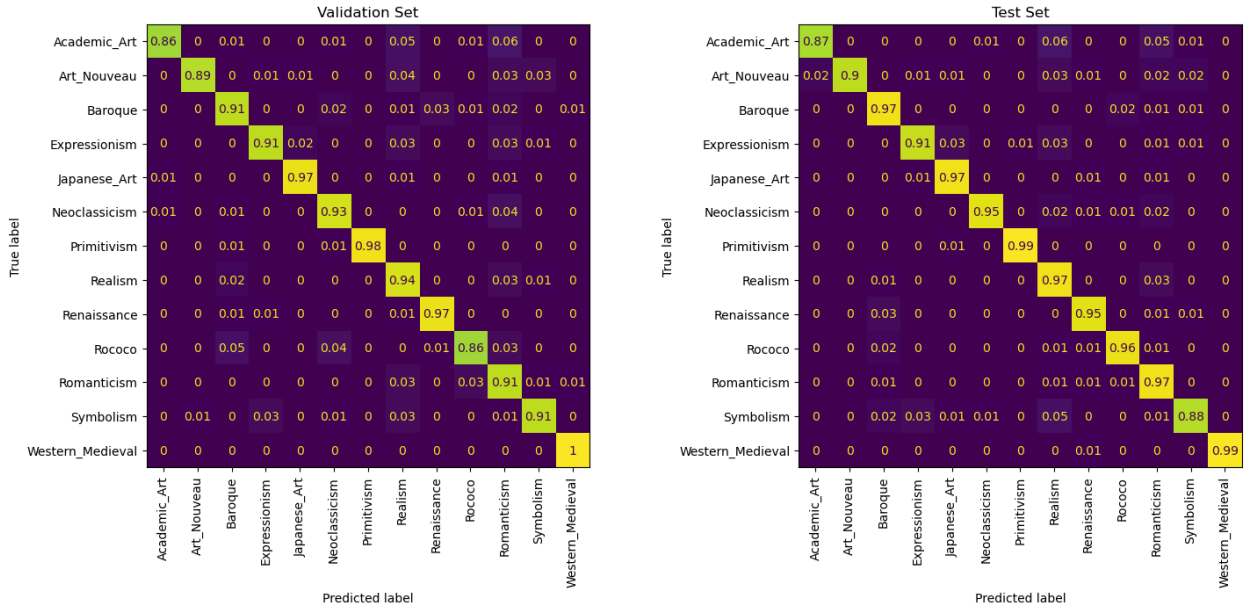


After this, we decided to increase the number of layers at the head stage to effectively turn the head of our model into a small neural network. The model performed slightly better than a run where we froze everything but the head, but it wasn't a large enough improvement, so we scrapped the idea of freezing all stages of the model.

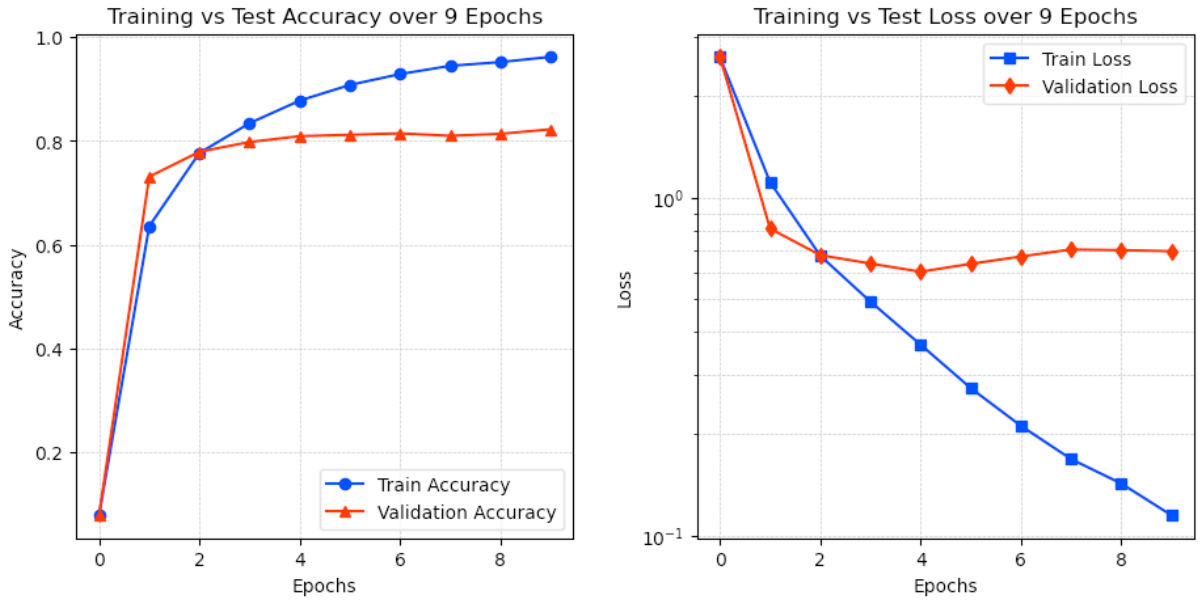
To deal with overfitting from our baseline run, we performed a run with light data augmentation where 30% of the images in our dataset would be horizontally flipped. To further reduce bias, we adjusted the amount of images in each class to be equal. This should've prevented the model from falling into a local maximum where it would vote for a single style because it was the most represented. Unfortunately, this run performed worse compared to our baseline.

We then experimented with a higher learning rate, allowing the model to learn faster from the dataset and converge earlier. We also unfroze the last couple of stages of the pre-trained Swin Transformer so the validation accuracy could increase without overfitting. We first tried a learning rate of 1e-3 with stage 4 of the Swin Transformer architecture unfrozen, but this led to an unstable validation accuracy between epochs. In another run where we kept the old learning rate but unfroze stage 3 of the Swin Transformer onwards, we experienced an anomaly where the validation accuracy was an expected 81% while the test accuracy was 94%. In the confusion matrix below, we also had our highest accuracy in detecting the Symbolism class here. This impacted future runs where we would try to recreate this result by unfreezing stage 3 onwards but we did not achieve the same results, leading us to believe this run to be an

outlier.

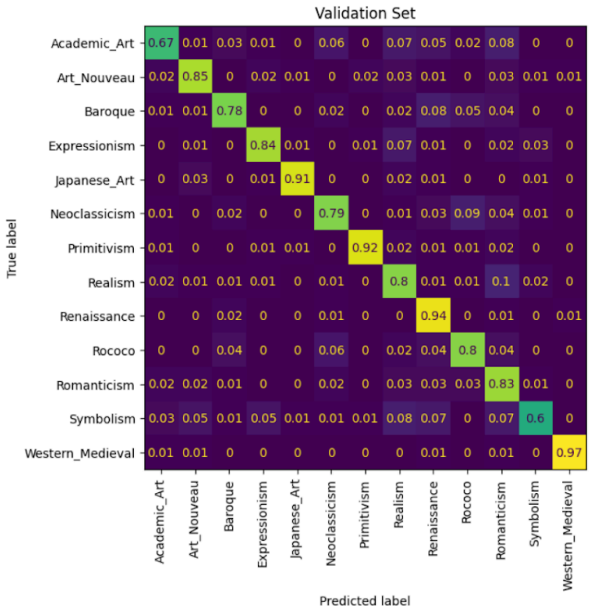


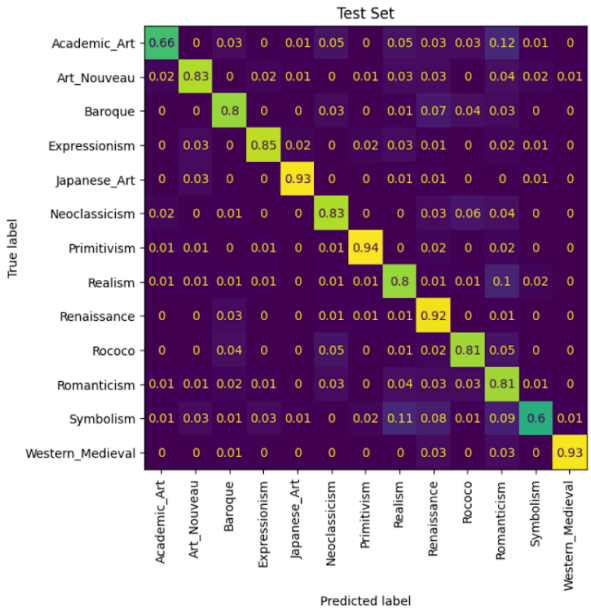
As the model was still overfitting, we created more aggressive data augmentation methods. We ran our most successful model with "Medium Augmentations" and "Heavy Augmentations" (as depicted in the methodology section above). These experiments plateaued relatively early at epoch 4, much like the previous experiments where we unfroze layer 3. While overfitting was still prevalent as can be seen in the graphs below, we were optimistic in exploring further augmentations from this experiment as validation accuracy was higher than it ever was in any of our runs.

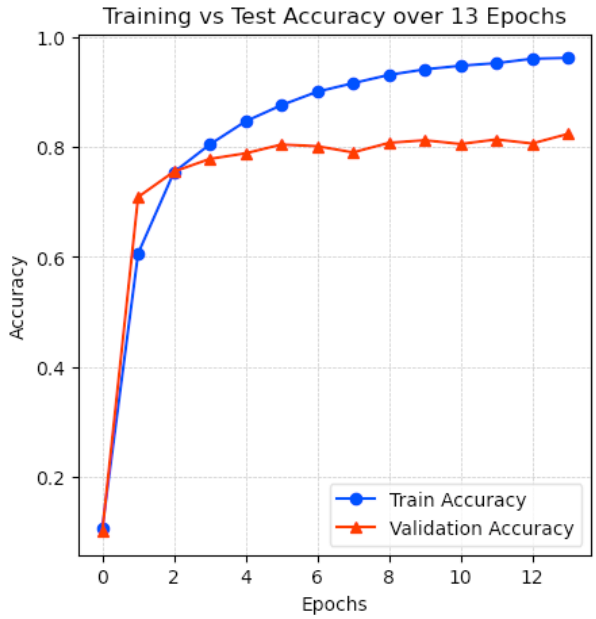


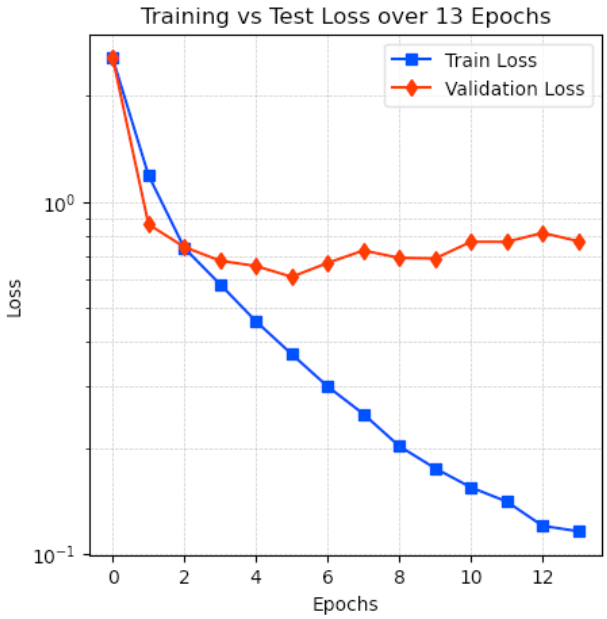
Following the success of the previous experiment, a "Heavy Augmentation" run was devised with the thought process that more augmentation leads to increased validation accuracy. Our hypothesis was confirmed, as this run had the best validation and test accuracy

(not counting the outlier) thus far at about 83%. We unfortunately didn't solve the problem of lower Symbolism classification accuracy rates as can be seen in our confusion matrix below.









Results

Below is a table listing the details of all our experiments along with the accuracy and loss of the final models that resulted from them after the given number of epochs. Train and validation statistics were calculated at every epoch, while test statistics were calculated after the model was done training.

While writing this paper, our model files got corrupted for one of our experiments, meaning we can't get test accuracy or loss for our even split light augmentation experiment. This shouldn't be a big issue as we expect this number to be somewhere close to the validation accuracy and loss as it was for most other experiments we performed. Missing data is denoted with "???" in our results table.

Nickname	Pretrained?	Unfrozen Layers	Learning Rate	Epochs	Augmentations	Even Data Split?	# Head Layers	Train Acc	Validation Acc	Test Acc	Train Loss	Validation Loss	Test Loss
Initial Run	Yes	All	5.00E-05	10	None	No	1	0.97876	0.80965	0.81153	0.06741	0.93544	0.8858
Freeze all but head	Yes	Head	5.00E-05	10	None	No	1	0.55379	0.54776	0.56682	1.4043	1.41548	1.38173
Freeze all but head-30	Yes	Head	5.00E-05	30	None	No	1	0.63456	0.61812	0.63059	1.13904	1.18148	1.15318
From Scratch	No	All	5.00E-05	10	None	No	1	0.15662	0.16047	0.16024	2.40522	2.4027	2.40236
Neural Net End	Yes	Head	5.00E-05	10	None	No	3	0.655	0.63059	0.64894	1.03786	1.09399	1.07325
Even Split Light Aug	Yes	All	5.00E-05	20	Horizontal Flip on 30%	Yes	1	0.98655	0.77092	???	0.0428	1.15286	???
Unfreeze Stage 4, high lr	Yes	Stage 4, Head	1.00E-03	26	None	No	1	0.20932	0.23412	0.19953	2.40242	2.28476	2.28717
Unfreeze Stage 3	Yes	Stage 3, 4, Head	5.00E-05	7	None	No	1	0.97368	0.80988	0.94352	0.08355	0.81881	0.20341
Unfreeze Stage 4, medium lr, light aug	Yes	Stage 4, Head	1.00E-04	13	Horizontal Flip on 50%	No	1	0.96974	0.80729	0.79059	0.09017	0.92215	0.65949
Freeze all but head, medium lr	Yes	Head	1.00E-04	27	None	No	1	0.63953	0.62071	0.63341	1.12051	1.16893	1.13904
Unfreeze Stage 3, medium aug	Yes	Stage 3, 4, Head	5.00E-05	9	Medium Augmentation	No	1	0.96232	0.82259	0.82612	0.11488	0.69453	0.68377
Unfreeze Stage 3, heavy aug	Yes	Stage 3, 4, Head	5.00E-05	13	Heavy Augmentation	No	1	0.96159	0.82353	0.82988	0.11585	0.77327	0.70521

Final Results table from experimentation (corrupted data is denoted with ???)

Our results show that unfreezing the last two Swin Transformer stages of a pre-trained model along with heavy augmentations gave us the best performance.

Conclusion

Our group's approach involved using a pre-trained Swin Transformer model as a base, where we then customized and tuned it to fit our specific usage of classifying the style of an image. As seen in our experience in the coding it up section, we implemented many methods of experimentation such as hyperparameter tuning, data augmentation, layer freezing, and training from scratch to obtain the best model performance. These methods of experimentation were mainly implemented to address the overfitting found in the initial training session but they mostly failed to improve the model's ability to generalize. In the end, we found that freezing stages 1 and 2 of our pre-trained Swin Transformer model while leaving stages 3, 4, and a single layer classifier head trainable gave us our best validation and test accuracy results. We believe our layer freezing methods helped keep the more generalized knowledge the model learned during pre-training, while leaving a few stages open to further improvement helped pick up the nuances of our project domain. As our model was still overfitting with this configuration and our data augmentation steps showed promise in closing this gap, we believe that further, and more aggressive data augmentation techniques would be an important avenue to explore to improve our model's performance in the future.

Works Cited

Azadi, S. (2023, January 3). WikiArt art movements/styles. Kaggle. https://www.kaggle.com/datasets/sivarazadi/wikiart-art-movementsstyles

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021, August 17). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv. https://arxiv.org/pdf/2103.14030

Luis, J. (2021, November 30). Training with pytorch. Training with PyTorch - PyTorch Tutorials 2.6.0+cu124 documentation. https://pytorch.org/tutorials/beginner/introyt/trainingyt.html

Microsoft/Swin-Tiny-patch4-window7-224 · hugging face. microsoft/swin-tiny-patch4-window7-224 · Hugging Face. (n.d.). https://huggingface.co/microsoft/swin-tiny-patch4-window7-224

Sanderson, G. (2024, April 7). Attention in transformers, step-by-step | DL6. YouTube. http://www.youtube.com/watch?v=eMlx5fFNoYc