Classifying Ancient Art with Neural Nets

Becky Desrosiers
School of Data Science
University of Virginia
Charlottesville, VA
rn7ena@virginia.edu

Alexandra Ferentinos
School of Data Science
University of Virginia
Charlottesville, VA
kzk8qq@virginia.edu

Peter Landers
School of Data Science
University of Virginia
Charlottesville, VA
bjs6pj@virginia.edu

Brendan Jalali
School of Data Science
University of Virginia
Charlottesville, VA
bdj9wf@virginia.edu

Fig. 1: Example of Egyptian Architecture from the late period

***Abstract*—This document describes our experiments using Python, TensorFlow and Keras to build a neural network to classify the culture, medium, and time period of various pictures of ancient Greek, Roman and Egyptian art. Our goal was to use deep convolutional networks, transfer learning, and advanced architecture to classify photos of artwork. In the end, we achieved modest results with about 50% accuracy in the final model.**

I. Motivation

This project seeks to solve the research question: Can Deep Learning methods be used to classify the culture, medium, and time period of ancient art? Our data is images of artwork, labeled with what culture and time period (e.g. early Greek) they are from as well as their medium (e.g. architecture). Our goal is to train a model to classify each image with all three tags. Our interest is primarily theoretical, instead of practical in application, and driven by an interest in middle-eastern, especially Grecian artwork. An example of a photo is included in Fig. 1.

II. Literature Survey

There are a few architectures with high performance on the ImageNet classification challenge and tasks similar to ours: AlexNet, Inception, ResNet, and SENet. A brief summary of their differences will be discussed. It is assumed that the reader has a basic understanding of a convolutional layer as well as fully-connected deep learning layers and pooling.

A. A Brief Overview of CNN Architectures

The main difference between AlexNet and the classic LeNet-5 architecture is that it stacks convolutional layers directly on top of each other, without pooling in between each layer (Aurelien 2019). It uses max pooling instead of average pooling, and rectified linear unit (ReLU) activation instead of hyperbolic tangent activation. Since AlexNet is large and deep, it employs dropout regularization to the 2 fully-connected layers at the end, before the output layer, to reduce overfitting(Aurelien 2019). When the AlexNet team won the ImageNet challenge in 2012, they also employed data augmentation in their training including random crops, mirror-images, and altered brightness to reduce overfitting. The benefit of AlexNet is a simple architecture that's easy to understand and implement. On the other hand, it has a lot more parameters than the following models (Aurelien 2019) and consequently is more prone to overfitting and may take longer to train.

Inception modules were first introduced in GoogLeNet, which won the ImageNet challenge in 2014. The fundamental idea of an inception layer is that there are multiple convolutional layers all wrapped up into one and concatenated to produce the output of a single layer (Aurelien 2019). An inception layer includes convolutional layers with different kernel sizes in order to capture both simple and complex patterns in the input. Layers with a 1x1 kernel capture cross-channel patterns as well (Aurelien 2019). GoogLeNet used global average pooling to flatten out the spatial layers and dropout regularization.

A residual network (ResNet) introduced a method of feeding raw inputs directly to future layers, skipping past some neurons. A ResNet is a compilation of residual units (RUs), where each RU is a small neural network with the original inputs added to the outputs (Aurelien 2019). ResNet architecture is visualized in Fig. 2. The purpose of RUs is to protect against stagnation in training, so that if one layer of the network

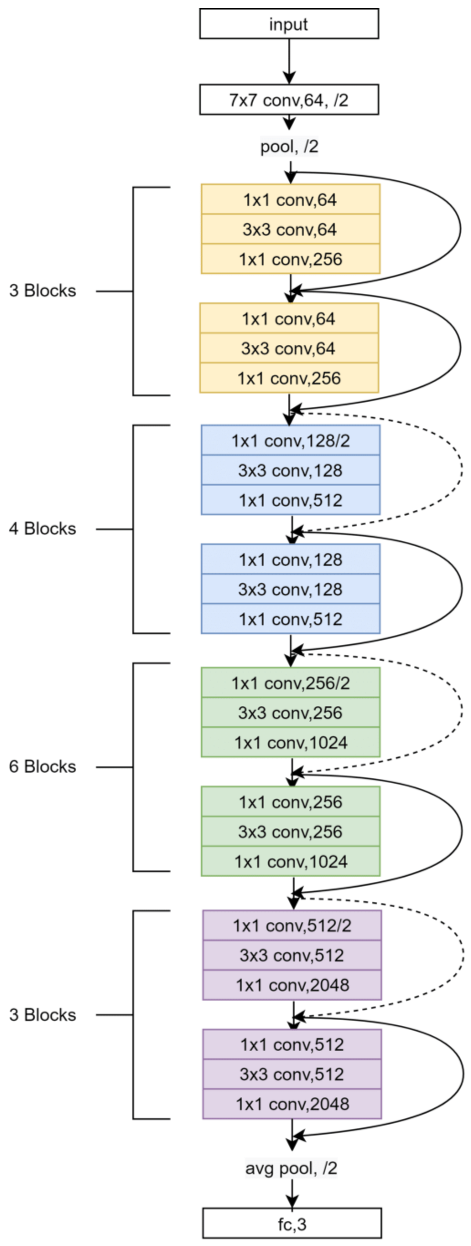


Fig. 2: ResNet50 [7]

is not learning, it can be "skipped" and that layer does not block the whole rest of the network from learning (Aurelien 2019). A deep, 152-layer version of ResNet won the ImageNet challenge in 2015 with a top-5 error rate under 3.6%.

Finally, the Squeeze-and-Excitation Network (SENet) introduced the SE block to CNN architecture. The SE block can be added after any other layer, and applies weights to each feature map (Aurelien 2019). The SE block includes a global average pooling layer, a fully-connected, ReLU-activated layer, and a fully-connected, sigmoid-activated layer to output the weights. These layers are visualized in Fig. 3. Adding an SE block can improve any existing architecture with very small computational cost (Jie Hu et al. 2018). SENet won the ImageNet challenge in 2017 with a 2.51% top-5 error rate.

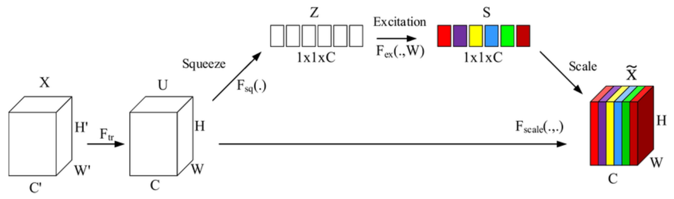


Fig. 3: Squeeze-and-Excitation (SE) Block [8]

Beyond the classical CNN architecture of LeNet-5, the strategies described above can be helpful for building the optimal CNN to solve our research question. It may be beneficial to stack convolutional layers on top of each other, use inception or residual modules, or SE blocks to improve the performance of our classifier.

B. *Image classification*

The fundamental task at hand is one of image classification. All of the models described in the preceding section have had high levels of success with image classification on the ImageNet dataset, and may be work mimicking. Other methods that might be helpful include data pre-processing: Y Zhu et al. (2019) found that, when working with pictures of paintings, resizing the images less than 256x256 made it impossible for their models to perform well in a 9-class style-classification problem. They tested 300x300 images with different models, then used 400x500 and finally 500x500 images with the most promising models to achieve an accuracy of 88.35%, better than expected for a human amateur. It may be useful for us to scale down the images for more efficient model training during initial testing phases, but then later scale them back up to achieve optimal performance on our selected model.

Manzoor et al. (2022) successfully used a simple AlexNet architecture to classify Roman coins with a 96.3% accuracy. Their model employed adaptive learning with ADAM optimizer and sparse categorical cross-entropy loss, so we will consider using these parameters in our model.

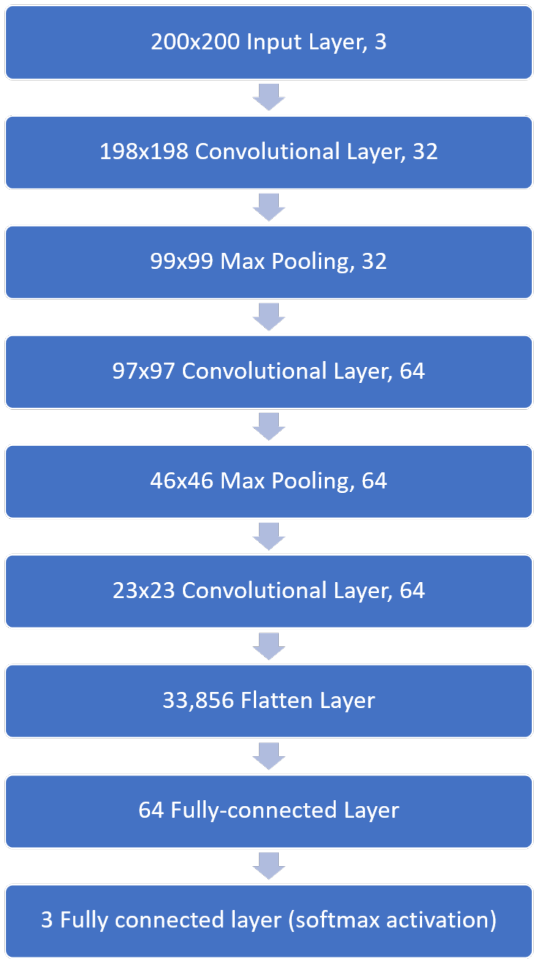
According to Yosinski et al. (2014), every CNN learns the same basic features in the first few filters. Regardless of the task, the first few layers show Gabor filters and color blobs, even in cases of unsupervised learning. Therefore, it is possible to use models that were pre-trained on much larger datasets to capture basic features that can be used by our model as well. Huh et al. agrees that the default way to train a computer-vision model is to use another model that was pre-trained on ImageNet, and then fine-tune the top layers for a specific task (2016). Mahajan et al. goes so far as to say that attempting to solve an image classification problem without using supervised pre-training would be "foolhardy" (2018). Therefore, it seems beneficial to explore the possibility of transfer learning in our task.

C. *Transfer learning*

Transfer learning provides the benefit of lower layers trained on an image dataset much larger than ours. In theory, training on a larger dataset helps a model generalize and recognize important features without overfitting to the training dataset. Mahajan et al. and other authors refer to ImageNet as the "de facto pretraining set," presumably because of its size and perhaps the diversity of its 1,000 classes. Interestingly, M. Huh, P. Agrawal, and A. Efros. found in their 2016 investigation that it may not be vital to have so many training observations and classes to develop a robust model for transfer learning. They found only a slight decrease in performance when using half of the ImageNet dataset, and similar results when using only 127 "coarse" classes, instead of the 1,000 (some of which are very similar, like different dog breeds). In some cases, pre-training on ImageNet with fewer classes actually led to improved performance on the transfer dataset. In short, blindly adding data to pre-training doesn't always improve performance.

J. Yosinski et al. found two distinct issues that might impact the effectiveness of transfer learning: neuron specialization and co-adapted neurons (2014). The first issue is that higher-level neurons adapt to their specific task and are not very useful for any other tasks. We can solve this problem by cutting off the top layers of a pre-trained network and adding our own layers to train to our specific task. The second issue occurs when this splitting is done: when the network was originally trained, every layer learned in the context of the other layers. The result is that some layers may be co-adapted, i.e. their efficacy in the model is dependent on each other. Therefore the transferability of a network may be significantly impacted by where the network is split. This problem can be minimized if lower-layer features are also able to be fine-tuned on the target dataset. With this knowledge, we will apply transfer learning to our research problem by using the lower layers of pre-trained models, adding and fine-tuning our own layers on top of them, then allowing the lower layers a few epochs to be fine-tuned as well.

In the 9-class, art-style classification problem explored by Zhu et al. in 2019, four models (Inception-V3, VGG-16, ResNet-152, and Inception ResNet) were trained using both pre-trained weights from ImageNet and randomly initialized weights. The authors hypothesize that the paintings are significantly different enough from the natural domain of the ImageNet dataset that the learned features were not helpful. Indeed, it has been found (and makes sense) that transfer learning will be less useful when the original training task differs greatly from the task at hand (Yosinski et al. 2014). Even so, the authors of the 2019 art classification study only replaced the final softmax layer of the pre-trained models. It may be that their models suffered from the problems described in the previous paragraphs, and could be improved greatly just by removing a few more layers of the pre-trained models. On the other hand, it's possible that ImageNet transfer learning may be less helpful to our research problem, given the domain of art.


200x200 Input Layer, 3
198x198 Convolutional Layer, 32
99x99 Max Pooling, 32
97x97 Convolutional Layer, 64
46x46 Max Pooling, 64
23x23 Convolutional Layer, 64
33,856 Flatten Layer
64 Fully-connected Layer
3 Fully connected layer (softmax activation)
Fig. 4: Preliminary Architecture

D. Conclusion

This literature review has provided a brief summary of possible useful architectural units for image classification: stacked convolutional layers, inception modules, residual units, and SE blocks. Other techniques that may help an image classification task include scaling images, using adaptive learning and sparse categorical cross-entropy, and transfer learning. We found that some issues in transfer learning can be overcome by removing top-level layers of pre-trained models and allowing for the lower-level parameters to be fine-tuned on the target dataset. We considered using all of these methods in our experiments.

III. Method

The high-level structure of our model is three separate simple classifier sub-models – one for culture, one for medium, and one for time period – all connected by a coordinator. The final model takes a 3-channel picture as input, feeds it to each sub-model, and gathers the three predictions into a single result. The three sub-models are trained separately so that they can each specialize in a single area, be it culture, medium, or time period. We started with the same baseline

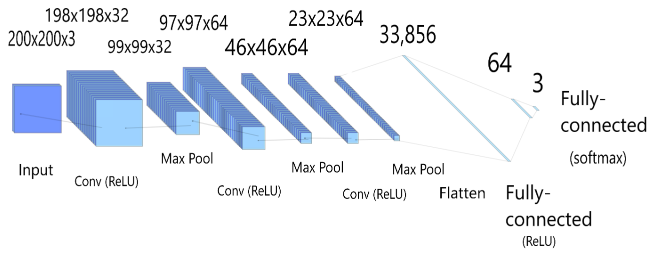


Fig. 5: Preliminary Architecture [9]

TABLE I: Results from Preliminary Architecture

	Training Accuracy (%)	Validation Accuracy (%)
Culture	37.66	31.96
Medium	40.34	39.3
Time	66.64	60.12

architecture for each model, and performed experiments to tune each sub-model to its particular task.

Our baseline model architecture (Fig. 4) is similar to a classical CNN using three convolutional layers sandwiched with max pooling, and the final layer being a fully-connected layer with softmax activation. From the past success of AlexNet and subsequent architectures, we used rectified linear unit (ReLU) activations instead of the classical hyperbolic tangent. The baseline model was optimized using Adam with a learning rate of 0.001. We chose this architecture as a baseline for simplicity in our preliminary experiments, then built off of it by adding more advanced machine learning techniques, described below. Another representation (Fig. 5) shows a more detailed version to help visualize the dimensionality of the problem. The preliminary model architecture led to the results displayed in Table 1.

A. *Data Pre-processing*

Data pre-processing is an important factor in machine learning. For our preliminary models, we standardized the size of the images to a $200\times200\times3$ array. For our transfer learning models, the pre-trained models expect input of size 224×224, so we used that size for many experiments and our final models. Since large inputs can be problematic in deep learning models, we standardized the inputs by dividing by 255. We also used caching, shuffle, batching, and pre-fetching, all with standard TensorFlow-provided functions.

In order to reduce overfitting to specific images and supplement our relatively small dataset of around 3.5k labeled images, we used random horizontal flips, rotations, and zooming in/out as well as small changes in brightness and contrast to augment the data. We used standard functions provided by keras for all data augmentation steps.

B. *Stabilization Techniques*

We introduced batch normalization using in order to stabilize the training at each block. We also used gradient clipping in some experiments to decrease the chance of gradient explosion.

TABLE II: Results from Preliminary Experiments

	Training Accuracy (%)	Validation Accuracy (%)
Culture	34.69	32.84
Medium	79.67	24.34
Time	67.72	20.82

C. *Training Optimization*

Doubling the number of filters in the convolutional layers at each step was attempted in order to increase the number and complexity of features the models could identify. We used max or global average pooling between convolutional layers in order to simplify the number of parameters. Different optimizers, including classic stochastic gradient descent and Adam, were used for experiments. We also tested different batch sizes and learning rates, including learning rates with decay, to optimize training.

D. *Regularization*

Regularization techniques employed in our experiments included dropout at different rates and early stopping.

E. *Transfer Learning*

In order to make use of much larger datasets and pre-trained models, we experimented with transfer learning using the following models that come with keras:

- ResNet
  - 50
  - 101
  - 152
- EfficientNetV2B0
- MobileNet
- Inception V3
- DenseNet
- NasNet

IV. Experiments

Our first experiments involved tweaking hyperparameters such as the number of filters, size of convolutional kernels, the presence and type of pooling (max or global average) and different levels of dropout. Using only our own training data, we ended up with the accuracies listed in Table 2. Though the validation accuracy of the culture submodel improved marginally, the other two had major issues with overfitting.

Overall, our model suffered from limited validation accuracy and overfitting. Our submodels each were limited by the amount of training data we were able to collect. However, we were able to successfully build a framework for training and testing models with different hyperparameters and architecture. In the next step, we tapped into the rich resource of transfer learning by using models pre-trained on the ImageNet dataset. We hoped that the pretrained models would be able to capture low-level features much more efficiently so that, when fine-tuned on our training data, our submodels would be able to focus on the high-level features that separate the artwork of different cultures, mediums, and time periods. We

TABLE III: Results from Final Experiments

	Training Accuracy (%)	Validation Accuracy (%)
Culture	36.19	46.63
Medium	40.76	45.16
Time	49.51	56.60
Full model		50.66

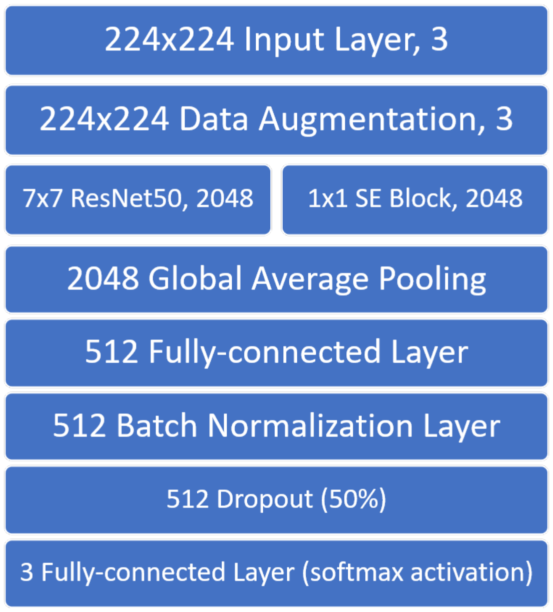

224x224 Input Layer, 3
224x224 Data Augmentation, 3
7x7 ResNet50, 2048
1x1 SE Block, 2048
2048 Global Average Pooling
512 Fully-connected Layer
512 Batch Normalization Layer
512 Dropout (50%)
3 Fully-connected Layer (softmax activation)

Fig. 6: Culture & Medium Submodel Architecture (refer to Fig. 2 & 3 for ResNet50 and SE block architecture)

also experimented with transformer units, but found that it increased training time dramatically without a great increase in performance.

V. Results

After experimenting with transfer learning, inception modules, residual units, and SE blocks, we came to our final models with results summarized in Table 3. The final architecture of the models is visualized in Fig. 6. The training and validation accuracies for all three models are visualized in Fig. 7. We can see that the models stagnated relatively early on in the training process.

We found that using transfer learning with ResNet50 architecture and an SE block worked the best out of all our experiments. The final models were trained for 20 epochs with the Adam optimizer and an exponential decay scheduled learning rate. Even though the final validation accuracy for the time submodel is a little lower than the preliminary model, it turned out that the preliminary model would always predict the middle time period, and so it wasn't useful. There were about 3 times as many data points in the middle time period as early or late, which adds up to explain the accuracy around 60%. The final time submodel had a lower accuracy, but was more useful overall. The full model accuracy in Table 3 (50.66%) represents the averaged accuracies across all 3 confusion matrices (Fig. 8). The final model got all three labels correct for about 11.25% of photos in the validation set.

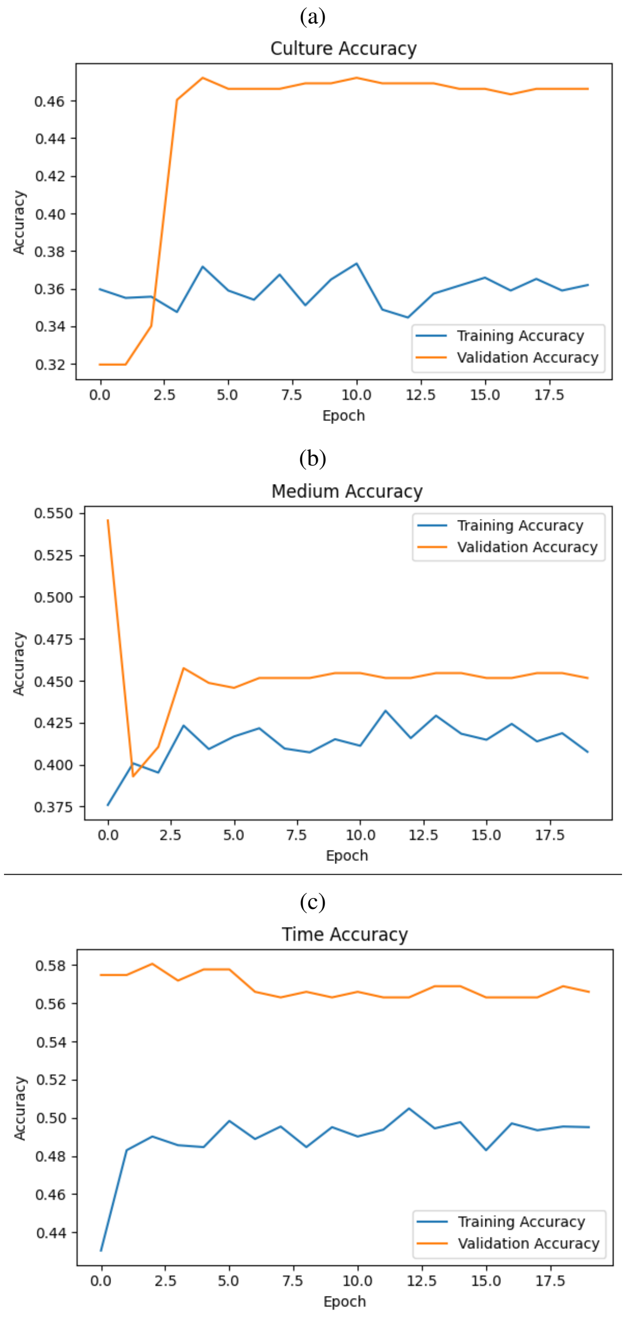


Fig. 7: Training and Validation Accuracies over 20 Epochs

From the confusion matrices in Fig. 8, we observe that the culture submodel was biased toward Egyptian, and almost never predicted Roman. This does not reflect the breakdown of labels in the data, where Greek artwork had the highest representation. The culture model had the hardest time identivying Roman artwork, with only 1 correct Roman identification. The medium submodel was biased toward paintings, even though there were about the same number of paintings and sculptures in the dataset, with about half as many instances of architecture. This bias showed up in low recall for both architecture

(a) Culture

Predicted	Egyptian	Greek	Roman
Actual			
Egyptian	114	6	0
Greek	67	37	0
Roman	83	21	1

(b) Medium

Predicted	Architecture	Painting	Sculpture
Actual			
Architecture	11	69	2
Painting	7	118	1
Sculpture	4	91	26

(c) Time

Predicted	early	late	middle
Actual			
early	21	17	31
late	20	26	27
middle	15	26	146

Fig. 8: Confusion Matrices for each category in the final model

and sculptures. As mentioned earlier, the preliminary model favored the middle time period, likely due to the large majority of photos that fell within the middle time period. The final time submodel predicted more evenly.

All code notebooks and data can be found in the PROJECT partition on Rivanna at absolute path /sfs/gpfs/tardis/project/SDS/instructional/DS-6050-DL/24F/ClassicalArt_Project. The coordinator between all three models and the final results can be found in wrapper.ipynb.

VI. Conclusion

In conclusion, we learned that it is very difficult to train deep neural networks for image classification tasks! Our results turned out to be underwhelming and our tool probably won't be very useful as-is for classifying artwork. We were able to confirm that adding an SE block to our model improved our results. The creators of the SE block described it as a compute-cheap way to improve any model, and it served our model well. We gained hands-on experience and a deeper understanding of advanced deep learning architecture as well as how to handle versioning challenges and save and load keras models to disk.

There are a few things that could be done in the future to improve our model. Our fine-tuning process was limited by the amount of data we had to train on. Using images as data proved to be very cumbersome and it was difficult to acquire and transport large numbers of photos at a time. Even so, we learned about how to transport data and create a custom TensorFlow dataset, and got experience with data preprocessing steps. In the real world, data is rarely tied up with a bow the way we receive it in class for practice. Through this experiment, we gained valuable insights into the challenges of data mining, cleaning, and transport.

Our experiment might benefit from some re-framing as well. Although we wanted a model that could classify a picture into three separate label categories, the 3-submodel plan might not be the best way to accomplish our task. Features from early Greek sculptures may bear similarity to late Roman paintings that confuse the model. It may be worth exploring a different paradigm entirely, or splitting the problem up by culture or medium.

VII. Members' Contribution

Becky: Organization/created GitHub project, created TensorFlow datasets from raw data, coded wrapper.ipynb, diagrammed model architecture, wrote up this report and drafted the final presentation.

Alexandra: Collected and organized raw data, authored the culture sub-model, and documented experiments with the culture sub-model.

Pete: Collected and organized raw data, authored the time sub-model, and documented experiments with the time sub-model.

Brendan: Collected and organized raw data, authored the medium sub-model, and documented experiments with the medium sub-model.

References

[1] Geron, Aurelien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O-Reilly Media, Inc. Sept 2019.
[2] Y Zhu, Y. Ji, Y. Zhang, L. Xiu, A.Zhou, and E. Chan. Machine: The New Art Connoisseur. arXiv e-prints, page arXiv:1911.10091, Dec 2019. https://arxiv.org/pdf/1911.10091
[3] Manzoor, Sehrish & Ali, Nouman & Raees, Muhammad & Khan, Awais & Ayub, Muhammad & Ahmed, Afzal. (2022). Ancient coin classification based on recent trends of deep learning.
[4] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems, pages 3320–3328, 2014. https://arxiv.org/pdf/1411.1792
[5] Huh, M., Agrawal, P., Efros, A.: What makes ImageNet good for transfer learning? arXiv:1608.08614 (2016)
[6] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in ECCV, 2018. https://arxiv.org/pdf/1805.00932
[7] Al-Humaidan, Norah & Prince, Master. (2021). A Classification of Arab Ethnicity Based on Face Image Using Deep Learning Approach. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3069022.
[8] Asker, Mehmet Emin. (2023). Hyperspectral image classification method based on squeeze-and-excitation networks, depthwise separable convolution and multibranch feature fusion. Earth Science Informatics. 16. 1-22. 10.1007/s12145-023-00982-0.
[9] LeNail, Alex. NN SVG. https://alexlenail.me/NN-SVG/