RESTORING ANCIENT GREEK VASES WITH COMPUTER VISION

Desi DeVaul (dpd2136) and Nicky Khorasani (nck2130)

Columbia University

ABSTRACT

In this paper we contribute several new fine-tuned models and performance comparisons for the task of image restoration of ancient Greek vases. Ancient Greek vases are often decorated. After having survived for thousands of years, they sometimes accrue some damage, particularly to the decorated parts: either the paint has faded away in certain areas or otherwise parts of the actual vases have been been destroyed, removing that decoration. The study of these vases is important to the study of classics and by extension the study of the formation of Western civilization. To this end, models that generate possible restorations for these vases would be a useful contribution to classicists and archaeologists working on pottery and could be extended from ancient Greek pottery to other types of ancient artifacts. We first contribute several models that can restore suggest repairs for damaged decoration, and we do a comparison between them. We then try to extend this work to restoring the scenes, which are more complex, varied, and require more computation.

1. INTRODUCTION

Classics is the study of ancient civilizations, traditionally with a focus on the cultures and languages of the near Mediterranean, nominally ancient Greece and Rome. The study of classics is important because of the lasting impact of these ancient cultures on our world [1] [2] and the way these cultures have shaped our forebears. [3]. Part of classical studies relies upon the study of ancient pottery, with serious studies of Greek painted ceramics beginning in earnest during the 18th century." [4].

Many ancient Greek vases have both decorations (i.e. wreathes and palmettes which adorn the vase) and depictions of scenes. These scenes could be occurrences of daily life or festivals, or they could depict the gods. The study of these vases is instrumental in understanding many aspects of ancient Greek culture, from religious practices [5] to gender roles[6].

Although many examples of Greek pottery have survived since antiquity, many of them are reconstructions or have suffered some, if not significant, damage to their decoration and to the scenes which they depict. Manually restoring these paintings has been helpful to the field of classical archaeology, but it is a laborious and time-consuming process. Therefore, an efficient way to either suggest a restoration for either the damaged decoration, or the damaged scene, or both, would benefit the field. We have fine-tuned several models such that, given an image of a vase with a damaged area 'masked out', it will generate a new image with this mask filled in, thus 'restoring' the image.

2. BACKGROUND

Other groups have tried applying machine learning to solve similar problems: identifying pottery fragments [7], restoring ancient inscriptions [8] and restoring the pottery shapes [9]. However, no one has yet attempted to use machine learning or computer vision techniques to restore the actual *images* depicted on these vases. Our paper contributes several fine-tuned computer vision models specifically for this purpose.

A lot of work has been done in the field of computer vision toward creating models for image generation [10] [11] [12] [13]. Recently, diffusion models have begun to gain prominence as the predominant model for image generation [14]. Diffusion models are trained in a self-supervised manner, where noise is randomly added to an image to obfuscate it, and then the model learns to 'undo' this noise and return to the original image. For this reason, diffusion models might also be referred to as a "denoising" model. Stable diffusion is a particular style of diffusion model. It works based on the premise that images can be compressed to a smaller "latent space" – by virtue of the "Variational Auto Encoder" – at which point noise can be added that the model learns to be removed [15]. This is a more efficient process since its happening in a smaller latent space rather than the full image space. This is the type of model which we worked on. Of the many different tasks for such generative models, one stands out as particularly relevant to the task of partial image reconstruction: generative inpainting. Generative inpainting works in a self-supervised way: given some training images, randomly mask (i.e. replace the masked pixel values with all white values) a subsection of the image, and use some generative procedure to "fill-in" what has been masked [11]. Naturally, people have worked to combine diffusion models and the inpainting task by taking a pretrained diffusion model and further finetuning it on the inpainting task [13]. Multiple different methods exist for updating the weights of a model for better performance on a specific task. There is, of course,

finetuning, by which we explicitly mean the updating of *all* weights within the model using standard back propagation. This is precisely what we have learned in class.

Given the incredible size of some modern machine learning models, researchers have also created efficient ways of "finetuning" these models by adding an "adapter" to these models, a new tool which contains a much smaller amount of weights compared to the entire model, and then updating the weights only for this adapted. Thus, one can tailor the larger model to this specific task by only having to update the weights of the adapter. This process is called Low-Rank Adaptation (LoRA), and was orignally created for use with large language models [16]. However, researchers have also adapted LoRA to work with vision models [17]. Importantly, LoRA does *not* update the weights of the base model, it only learns the adapter weights, hence its efficiency.

Finally, work has been done on how to adapt vision models, specifically text to image models, for a specific individual object, which one might want the model to learn. This training process is called "Dreambooth" and it works to help a model very quickly learn how to incorporate a specific object [18]. Dreambooth takes only a handful of images (5-12 or so) for the model to learn this new object. It does this by associating this object with a particular input token, such that it can very quickly learn the association between the two. Dreambooth works under the assumption that these models already have a wealth of knowledge, and so adding an additional object to its "memory" should require relatively minimal effort if it properly utilizes what the model has learned already. However, one of the keys to diffusion is that the model has already learned the specific class associated with the particular object that it is learning. For instance, if you want it to learn object "X" as an instance of a particular class, say "Y", then the should a priori know the class "Y". For instance if it knows what a class "Cat" is, it can quickly learn a particular cat, e.g. "Fluffy."

3. METHODS

3.1. Dataset

We compiled multiple datasets using images that we downloaded from the BPAD. The BPAD is a large online repository of images of ancient Greek vases.

Our first dataset is a collection of vases who are entirely patterned. We began by selecting images whose "decoration" tag was "BLACK PATTERN." By limiting these images to the same style and color, we hoped to improve our chances at success. We further honed in the specificity of the task by only selecting specific types of vases. We selected these bases because they were a. similar, and b. had clean images, generally. The vase types we selected were: "LEKYTHOS, SQUAT", "ALABASTRON", "LEKYTHOS", "AMPHORA, NECK" and "ARYBALLOS."

This data set was on the smaller size (about 106 images after we removed by hand the bad ones), but because we our masking our images several times, each image can be used multiple times in training. We additionally cropped each image three different ways, so the effective size of our dataset was at least one order of magnitude larger, all things considered.

In order to try and expand our project, we also collected some scenes which we wished to try and train our models with. These scenes often contain people, animals, or gods interacting. We decided to collect all the images of the same style ("BLACK FIGURE") and of the same vase type: "AMPHORA, NECK". When we scraped all of these images, we had about 6,700 total images. Examples of such images can be seen in the appendix. Notably, we had less time to clean these images so this dataset overall was of a lower quality than the first dataset.

3.2. Models

For our models we used the Stability AI Stable Diffusion 2 Inpainting Model [13] as our base model, from which we further trained our own models. The Stable Diffusion 2 Inpainting Model is a text to image model, that takes in a textual prompt and generates an image conditioned on this input. As it says in the model card: this "model is resumed from stable-diffusion-2-base (512-base-ema.ckpt) and trained for another 200k steps."

3.3. Training Methods

3.3.1. Finetuning Stable Diffusion

We tried two approaches to fine-tuning. First, we tried "raw fine-tuning" of Stable-Diffusion-2 (which we describe below). When this did not work as well as expected, we decided to leverage a popular GitHub repository with tools available to help us fine-tune Stable-Diffusion-2 for in-painting.

Raw Finetuning Stable-Diffusion

Raw finetuning Stable-Diffusion-2 involved downloading the stable-diffusion inpainting pipelinefrom the huggingface diffusers library and creating a training loop to update the internal weights of the network.

Unfortunately, the stable-diffusion-inpainting pipeline does not support fine-tuning out of the box. To circumvent this, we attempted to apply network surgery to the diffusion model as taught in class.

The two primary blocks of an in-painting model are the Variational Auto Encoder (VAE), which creates a representation of the image in a lower dimensional "latent space", and the U-Net which is a Convolutional Neural Network (CNN) that takes as input the latents generated by the VAE and performs the diffusion process on them [MAYBE CITATION?]. For our network surgery, we decided to freeze the

Fig. 1. The original image (left), masked image (center) and generated image (right) with our "Raw Finetuned" model.

VAE weights and update the UNet weights because they are responsible for the diffusion process.

Our approach was to freeze the VAE weights and fine-tune only the UNet. This involved manually passing our masked images through the VAE and then passing these latents to the UNET. Once this was done, we decoded the UNET outputs before calculating mean squared error (MSE) loss between the generated image and the unmasked ground truth image and then used this loss to do back-propagation on the UNet.

Unfortunately, this proved to be an incredibly difficult task. The primary difficulty of this process was the latent tensors outputted by the VAE not matching the shape required UNet input shape. As a result, we tried to input dummy channels to the UNet. Adding dummy channels enabled us to run the training loop successfully, and our loss even decreased. However, when running inference, we were shocked to find that our output was all noise (See 1! We made many adjustments in an attempt to fix our training pipeline, but unfortunately, none succeeded. Therefore, we decided that this method of "raw fine-tuning" stable diffusion was outside this project's scope and pivoted towards using some established infrastructure for training in painting stable diffusion.

Using a Fork of Stable-Diffusion

We decided to pivot to forking this GitHub repository of Stable-Diffusion. We first needed to create masks for our dataset to finetune using this fork of stable diffusion using our pattern dataset. Then, we wrote a YAML config file describing the fine-tuning parameters. Finally, we ran the $main.py$ Python script using an A100 GPU on Google Colab. To mask our dataset, we evaluated a few options. For our "raw finetuning" experiment, we created masks out of a sequence of random squares. The center image of Figure 1 shows what this mask looks like. The second option was to use a script in the stable-diffusion repository called 'gen_mask_dataset.py' script. In the end, we opted to use the 'gen_mask_dataset.py' because it generates random masks of different shapes more robustly than our mask creation code. Additioanlly, it also creates three "crops" for each image (and a mask for each crop), expanding our pattern dataset by a factor of three. This gave us 300 training samples and 50 validation samples for finetuning with our pattern dataset.

For our first finetuning using this method we used this config

Fig. 2. Training and Validation Loss Per epoch in our first finetuning config

file. The key parameters for this finetuning were a "batch size = 4", "learning rate = $1 * e^{-6}$ ", and "max epochs = 50". Finetuning with this config led to our training loss and validation loss being very unstable, as shown in 2. Because of the instability of the train and validation loss in this run, we decided to finetune our model again with an "improved" config in order to increase the stability of the loss per epoch. This resulted in us finetuning with this config file. In this config file, we increased the batch size to 8, added gradient accumulation, decreased the learning rate, and added 10% dropout. We added each of these because they were discussed in class as effective ways to increase the stability of the loss.

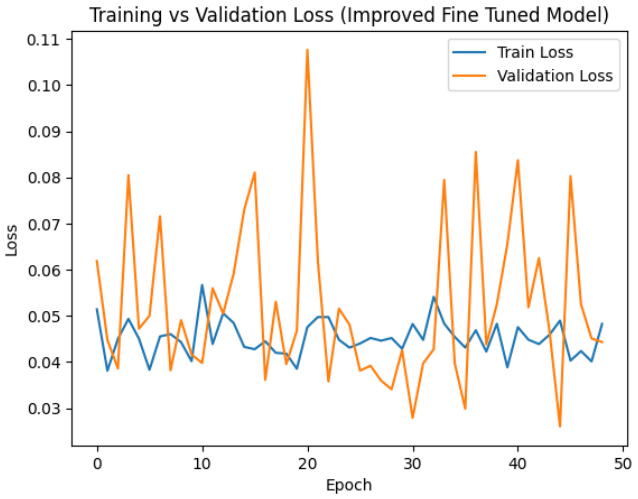


Fig. 3. Training and Validation Loss Per epoch with our "improved" config.

Unfortunately, as you can see in figure 4, the instabil-

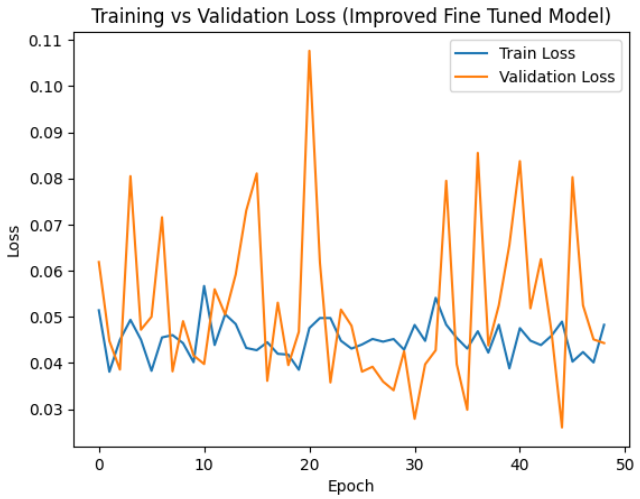


Fig. 4. Training and Validation Loss Per epoch with our "improved" finetune config

ity of our loss did not decrease with our "improved" config. Additionally, from 4, the validation loss became more unstable. This is probably due to using a small validation set for this finetuning. Despite the issue of unstable training and validation loss, we were pleased with the quantitative and qualitative results (which are discussed in the results section). We did try to overcome the issue of using a small validation set by using our large "scene vase" data set with this finetuning method. However, we quickly learned that finetuning stable-diffusion-2 with 1000+ images on a single A100 took on the order of days which would require resources (time and money) not available to use for this project.

3.3.2. LoRA

LoRA works by creating a low rank approximation of the normal weights matrix by decomposing this original weight matrix into two smaller matrices. These matrices form what is called an "adapter" because they work with the normal weights to create an augmented output. Thus, when one performs a LoRA training, one needs to only update these low rank approximation matrices and this should help the combination of the base model plus this adapter to perform better at the task.

For the pattern dataset, we trained 5 different LoRA adapters. They were of rank 4, 16, 32, 64 and 128, respectively. We ran each for 1000 training steps, a training batch of 1, accumulating the gradient for only 1 step, using a weight decay of 0.001 and a dropout rate of 0.01. We saved the models every 100 steps. We did not write this training code, we used the LoRA library.

What we did do, was take the masking code and modify it. Because the images that we had were of different sizes and shapes, I modified the automatic masking code. I created two different masking functions and copied the training code twice. On the "normal" masking function, I modified the coordinates at which the masks would be generated. I constrained where these would happen to the middle 50% of the image, because that is where the pattern or scene would be.

We then duplicated the training code to work with a different masking function. This masking function similarly centers the mask at somewhere within the middle region of the image, but it generates a singular large mask, that ranges in size between 64 and 128 pixels in size.

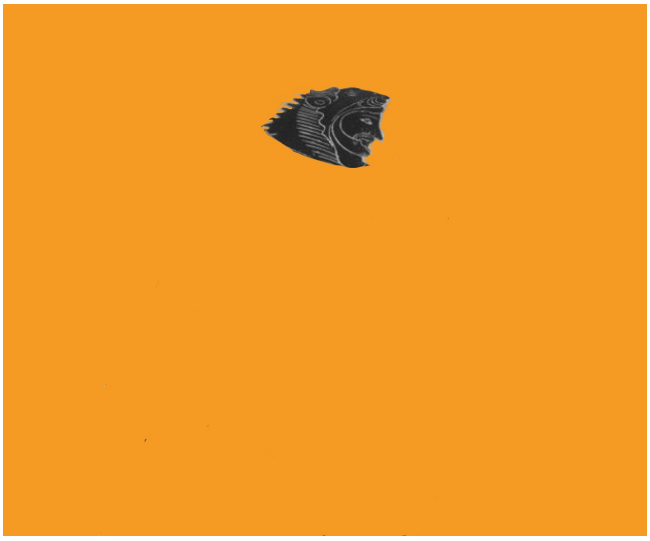
In total, we did 10 trainings of Lora: 5 different LoRA ranks and 2 different styles of masks. Each of these rank for 1000 steps. After every 100 steps, I saved the checkpoint and ran both the base model (which I did not finetune) and the finetuned LoRA model on the validation dataset to calculate the mean squared error against the base image. Note that each time the validation set was evaluated, masks were randomly generated and both the base model and the currently in progress LoRA model were evaluated on this dataset.

Because LoRA with rank 32 after 600 steps performed the best on the pattern dataset, we trained another version of this model on the scene dataset, in order to see if this procedure would expand to a more complicated dataset. We used the same hyperparameters and method of calulcating the validation loss which is described above.

3.3.3. Dreambooth

Dreambooth is designed to create a "personalized" Stable diffusion model. In essence, it is supposed to allow one to quickly adapt a stable diffusion model to learn about a particular object. One does this by taking an object and giving it a solid color background (see 5). Then, one passes in a prompt that designates this word with a specific word prefaced by the "sks" token. In essence, "sks" alerts the model's text encoder to pay attention to the fact that that the word following it is going to be associated with the input class. An application of this fine-tuning process would be to get a model to understand a specific figure or symbol or object that is common to Greek vase paintings. Then, if one saw one of these objects that was partially damaged or missing, they would be able to fill it in using this trained model. For instance, imagine they have a vase with a painting of Herakles, but his head is chipped. They could mask out the existing, damaged head, and then use a model which had learned what "sks Herakles' head" is to fill in the mask, thus "restoring" it. For the purposes of this final project we began by trying to get Dreambooth to understand Herakles' face. For the purposes of testing, we took 5 images of Herakles and colored out everything except for his face. We then ran Dreambooth training with the prompt "ancient greek vase of herakles sks face" as the prompt. We used quantized vectors of 16 bits

of precision, a training batch size of 1, and a learning rate of 5e-6. I then trained for 500 training steps, accumulating 2 gradient steps at a time. We did not write this training code, it comes from the Dreambooth library. We additionally trained



(a) Example data point for Dreambooth training

Fig. 5. This is an example of training image used for the Dreambooth finetuning model. Everything has been made monochromatic except for Herakles' face. The corresponding prompt was "ancient greek vase of herakles sks face", where 'sks' is a keyword for Dreambooth to alert it that it should pay attention and learn the following word ('face', in this case).

another Dreambooth model on images of legs from various vase paintings. Legs are a very simple shape that is also very consistent across paintings, therefore we thought it would be easy for the model to succeed at this task. We took 4 photos and colored in all of the background except for a singular leg, meaning we got multiple training images from one singular vase painting if the individuals shown had two images that were visible. We ended up with 11 images for training data of different legs. We used the same hyper parameters mentioned above.

4. RESULTS

4.1. Finetuning Stable Diffusion

Figures 6, 7, and 8 qualitatively show examples of the results for finetuning stable diffusion. As we can see, the original and finetuned models performed very similarly.

To qunatitatively evaluate the finetuned network we used the Structural Similarity Index (SSIM) (10) and Peak Signal To Noise Ratio (PSNR) (9 metrics to quantitatively evaluate the fine-tuning of stable diffusion. These are both standard metrics used to evaluate image generation models.

The PSNR of both our finetuned models outperforme out performed our base model on this test set. Additionally, we saw a slight boost in SSIM with our fine-tuned models. This is promising

Fig. 6. Test Image 0 Repair for the base, first finetuned, and "improved" model

Fig. 7. Test Image 2 Repair for the base, first finetuned, and "improved" model

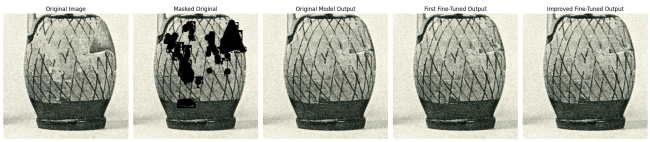


Fig. 8. Test Image 3 Repair for the base, first finetuned and, "improved" model

4.2. LoRA

In order to see how the LoRA models performed, we graphed their results against the validation dataset for every check point (see 11 and 12). We made two graphs, one for each type of masking that was performed. As can be seen in 11, most of the trainings were grouped together, though generally the best results were between 400 and 700 training steps. The best model overall was the model with a LoRA rank of 32 after it had run for 32 steps. This has an MSE of about 173. This was the lowest for any single model which I tested. The lowest base model had an MSE of 178 on this same validation set. Thus, this LoRA model **beat the base model performance** by about **2.8%** in terms of loss (not shown).

As for the LoRA models which we ran with the bigger masks, this actually made the models worse 12. None of them outperformed the base model, which had an MSE of 173 on one of the validation sets (not shown).

We have also shown in 13 an example of a repair done by the LoRA model. The LoRA model's suggestion is second to the left and the original image is on the right. The

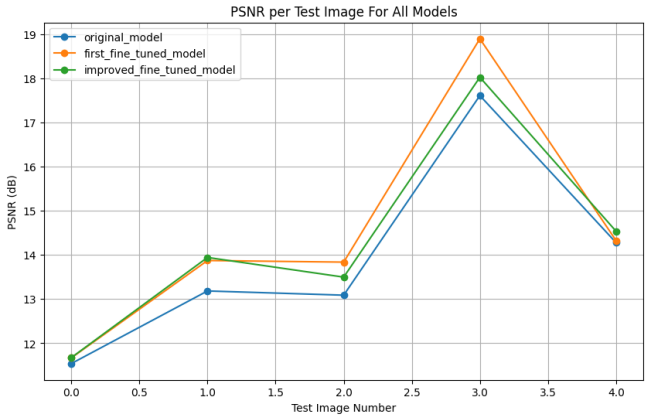


Fig. 9. PSNR of the base model, our first finetuned model, and our "improved" finetuned model on each of the 5 key test set samples

Fig. 10. SSIM of the base model, our first finetuned model, and our "improved" finetuned model on each of the 5 key test set samples

base model (not trained by us) model's suggestion is the farthest left. The mask is the second to the right. Clearly, in this example, LoRA appears to do the best job of creating a restoration.

We then also trained the best performing LoRA model on the scene dataset. We similarly ran this for 1000 training steps. As can be seen in the corresponding figure, we found that after around 400 steps, this LoRA model began to outperform the base model on the same datasets. We think that this shows this approach could be promising for future work specifically on restoring the scenes on ancient Greek vases.

4.2.1. Dreambooth

As for Dreambooth, we had mixed results. Because each validation required hand drawing over them to create the masks, we only made a couple of validation images for each iteration of Dreambooth. Because of this extremely small sample size, this section has to be conducted qualitatively. In short, Dreambooth did not seem to work very well. Case in point, the images pasted below. As you can see from 14, Herakles' leg does not get restored! Dreambooth training does not really seem let the model understand the features which were supposed to be included. We similarly trained Dreambooth to recognize "Herakles' face" but that performed equally poorly. Sometimes, it did in fact return somethingin the shape of a head, but it did not have the features of a head (namely eyes, nose, mouth, etc.).

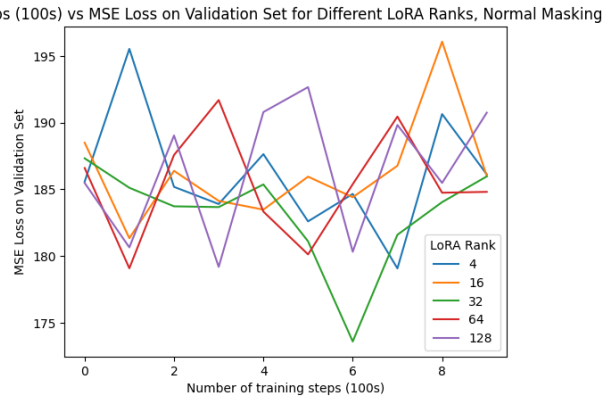


Fig. 11. Here we have the different validation loss values for each version of LoRA which we ran with the vanilla masking function. The number of training steps (in hundreds) is on the x-axis, and the MSE loss is on the y-axis. The lowest value for the base model is 178 (not shown.) The lowest loss for the LoRA models is 173 (shown in green.)

5. DISCUSSION

5.1. Finetuning

Despite the disappointing qualitative results of finetuning, the fact that the finetuned models had a better SSIM and PSNR is promising. This implies that with a larger training and validation set we could potentially finetune a model that is significantly better than the base model at our pottery restoration tasks. Unfortunately, we could not test this hypothesis with our larger scene vase dataset due to computational resource constraints.

5.2. LoRA

Overall, it seems like the best performing model outperformed the base model on the pattern dataset. Given the small increase in accuracy (approx. 3%), more testing is needed to really guarantee this result.

Additionally, the best models tended to reach optimal results in under 800 training steps, before they began to overfit.

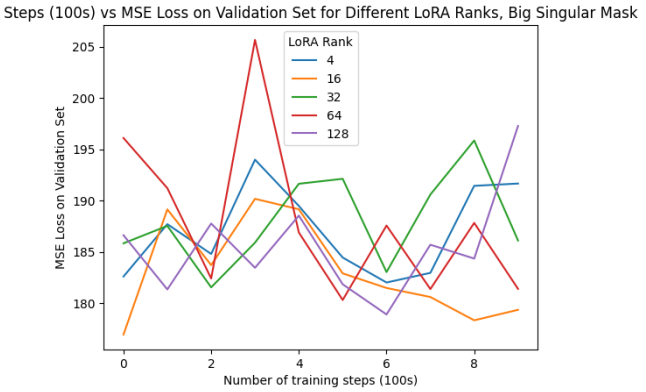


Fig. 12. Here we have the different validation loss values for each version of LoRA which we ran with the big, single masking function. The number of training steps (in hundreds) is on the x-axis, and the MSE loss is on the y-axis. The lowest value for the base model is 173 (not shown.) The lowest loss for the LoRA models is about 175 (shown in orange).

This is extremely efficient finetuning.

Finally, though we only trained one model on the scene dataset, it did outperform the baseline model. With more time, this would be an interesting extension of this project.

5.2.1. Dreambooth

Dreambooth had very disappointing results. It was very inconsistent and seemed to not quite be able to grasp the important key features of the objects it was tasked with learning. This perhaps makes sense in retrospect. Dreambooth is supposed to work by leveraging what the model has learned throughout its training *already*. Thus, the chances that it can already implicitly segment and understand "Herakles' face" or " a leg" is very low. It seems like the model was not able to learn any relationship between these images. I think perhaps this could be improved if more training images were gathered that were nearly identical (there was some variability in our training data), but that would likely lead to over-fitting. Overall, Dreambooth seems like an incorrect tool for this job.

6. CONCLUSION

In conclusion, we performed three different styles of training to the Stable Diffusion 2 Inpainting model to get it to work better on inpainting damaged images of ancient Greek vases. We trained these models for inpainting on both patterns and on scenes. Because we got positive results for creating better inpainting on our pattern data, and because our scene results (though scant) were also slightly positive, we think that there is room for further extension of this project to more robustly attempt this inpainting task on a wider variety of vases.

Fig. 13. Here we have the process of repairing an image. From left to right we have the image generated by the base model, the image generated by the LoRA rank 32 model after 100 steps, the mask applied to the original image, and then finally the original image. Note how the base model adds in some glare that shouldn't be there. The LoRA model's generated image looks fairly decent.

Fig. 14. Here is a result from the Dreambooth training. In the top left is the original image, in the bottom left is the mask applied to this image (the white park is what the model should fill in), and on the right is what the trained model provides. Clearly, this model can sometimes mess up quite badly.

6.1. Suggestions for Improvement

We think that one of the limitations of our approach for LoRA was using different validation sets because of the automatic masking each time. To make results easier to compare, a single validation dataset should have been created a priori and used every time. We also think that if we were to attempt this again, we would spend more time on the scene data trying to clean it. We noticed many damaged images that were less than ideal for training a model meant to undo this damage.

7. CONTRIBUTIONS

Desi: I finetuned the LoRA and Dreambooth models. I also found the data and wrote the basics of the code for scraping this data, but it was single threaded only. As for the report, I wrote the majority of the introduction and background sections. I wrote the methods section (where it applied to LoRA and Dreambooth) and the conclusions sections (where it applied to LoRA and Dreambooth), and I wrote my own results

/ discussion sections for LoRA/Dreambooth. I also wrote the suggestions for improvement section and the abstract. I edited together the video.
Nicky: I finetuned the stable diffusion model using the "raw method" and the stable-diffusion fork. I was responsible for scraping the data from the database and organizing and labeling it for all our models. I also designed and implemented an image masking scheme. I contributed to the introduction, background, and finetuning sections of the methods, results, and discussion.

8. REFERENCES

[1] Brian McGing, "Why bother with the classics in the twenty-first century?," *Classics Ireland*, vol. 26, pp. 142–157, 2019.

[2] Gilbert Highet, *The classical tradition: Greek and Roman influences on Western literature*, Oxford University Press, 2015.

[3] Paul MacKendrick, ""this rich source of delight": The classics and the founding fathers," *The Classical Journal*, vol. 72, no. 2, pp. 97–106, 1976.

[4] John H. Oakley, "Greek vase painting," *American Journal of Archaeology*, vol. 113, no. 4, pp. 599–627, 2009.

[5] Stefanos Gimatzidis, "Feasting and offering to the gods in early greek sanctuaries: Monumentalisation and miniaturisation in pottery," *Pallas*, , no. 86, pp. 75–96, 2011.

[6] Sue Blundell and Nancy Sorkin Rabinowitz, "Women's bonds, women's pots: Adornment scenes in attic vase-painting," *Phoenix*, vol. 62, no. 1/2, pp. 115–144, 2008.

[7] "An open system for collection and automatic recognition of pottery through neural network algorithms," *Heritage*, vol. 4, no. 1, pp. 140, 2021, Copyright - © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-11-24.

[8] Yannis Assael, Thea Sommerschield, Brendan Shillingford, Mahyar Bordbar, John Pavlopoulos, Marita Chatzipanagiotou, Ion Androutsopoulos, Jonathan Prag, and Nando de Freitas, "Restoring and attributing ancient texts using deep neural networks," *Nature*, vol. 603, no. 7900, pp. 280–283, Mar 2022.

[9] Pablo Navarro, Celia Cintas, Manuel Lucena, José Manuel Fuertes, Rafael Segura, Claudio Delrieux, and Rolando González-José, "Reconstruction of iberian ceramic potteries using generative adversarial networks," *Scientific Reports*, vol. 12, no. 1, pp. 10644, Jun 2022.

[10] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra, "Draw: A recurrent neural network for image generation," in *Proceedings of the 32nd International Conference on Machine Learning*, Francis Bach and David Blei, Eds., Lille, France, 07–09

Jul 2015, vol. 37 of *Proceedings of Machine Learning Research*, pp. 1462–1471, PMLR.

[11] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5505–5514.

[12] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. 2016, vol. 29, Curran Associates, Inc.

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10684–10695.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising diffusion probabilistic models," 2020.

[15] Onkar Mishra, "Stable diffusion explained," `https://medium.com/@onkarmishra/stable-diffusion-explained-1f101284484d`, 2024, Accessed: 2024-12-09.

[16] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "Lora: Low-rank adaptation of large language models," 2021.

[17] Cloneofsimo, "Lora," `https://github.com/cloneofsimo/lora`, n.d., Accessed: 2024-12-09.

[18] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 22500–22510.