Art Become Human: The Effect of Data Selection and Preprocessing on Artificially Generated Image Classification

Adam Klein
aklein4@stanford.edu

Matthew Villescas
mattjv22@stanford.edu

Alex Hodgges
alexh555@stanford.edu

I. Introduction

Generative AI image models are becoming increasingly prevalent and pervasive in society, be it through websites like The Night Cafe, apps such as Wombo AI, or models including MidJourney and DALL-E 2. The technology has been quick to take off, yet legislation and a solid understanding of the ramifications of such technology have been slower to follow. Currently, AI-generated art is not protected under US copyright, but there is still debate surrounding the use of copyrighted material in training sets of these models.

Consider Stable Diffusion, which uses the LAION dataset, a dataset of image-text pairs parsed from the Common-Crawl dataset. This dataset is fairly indiscriminate and contains content including copyrighted work, pornography, and violent images [1]. For-profit apps and platforms then use the Stable-Diffusion model, meaning that artists whose work was originally collected in the LAION dataset (a research dataset) can end up being used for commercial purposes. Real artists are already being affected, such as Greg Rutkowski, whose name has been used as a prompt over 93k times [2].

In the modern day, there exists little legislation to protect both artists and viewers. This leads to two main issues: first, artists have little to no say in how their work may be used to train AI models. Models tend to follow an opt-out scheme, if they offer the option at all. Secondly, AI-generated art can be misused for malicious purposes. AI-generated art isn't always disclosed as such, so viewers may face difficulty evaluating if a work is AI-generated or not. This can lead to ramifications in many contexts, ranging from the distribution and purchasing of art to art competitions and educational opportunities.

To address these issues, our goal is to contribute in identifying whether pieces were made by a human or an algorithm. Specifically, we aim to identify the best preprocessing techniques to create a robust system that can handle the wide range of quality of images and digital artifacts that may be present. We built and trained multiple convolutional neural networks using different datasets and evaluated their relative accuracy for classifying whether images were artificially generated. All code and referenced material can be found in the GitHub repository: *github.com/aklein4/ArtBecomeHuman/tree/gcloud-instance*.

II. Related Work

Much existing work has been done to identify fake images created with generative adversarial network (GAN) architecture. For instance, Wang et al. found that simple CNN models can be effective at detecting GAN-generated fake images [3]. Girish et al. further proposed a new algorithm, using a combination of CNN, WTA hashing, K-Means, and SVM classifiers, to achieve the same goal for unknown GANs [4]. However, there is not nearly as much literature on the identification of fake images created by other generation models, like Stable-Diffusion and DALL-E. We were only able to identify only one paper that did so, using a simple CNN for image-only detection and a binary classifier for hybrid detection (text and images) [5] However, unlike us, they focus on the model architecture more than the data used to train it. In addition to building these models, the study discovered that GAN-based detectors were unable to accurately detect diffusion-generated fake images, signifying that, in the context of image attribution, there are some unique properties of diffusion models.

Outside of academia, Hive AI is also working to address the issue of detecting AI-generated art. As of September 2022, they released an API that returns a classification label, confidence score, and the predicted source engine the image was generated from (for images classified as AI-generated) [6]. Details of the strategies and implementation behind the model are not currently available, but the release of this API confirms the Wang's claim that there exist some unique properties that differentiate real images from diffusion-generated images.

III. Dataset and Features

We drew our data (Fig. 1) from the Web Gallery of Art (WGA) [7], the Artbench Dataset [8], the Lexica Image Database [9], low-fidelity (LoFi) images we generated using Stable-Diffusion itself [10], and we drew a relatively small amount of images from the less-downloadable DALL-E Image Database [11].

The WGA is comprised of over 30,000 famous paintings of varying resolutions; the Artbench dataset contains 60,000 256x256 fair-use licensed images of 10 specific painting styles (Fig. 2); the Lexica Database houses thousands of art of varying resolution generated by Stable-Diffusion from various internet users, of which we drew 30,000 images using reverse

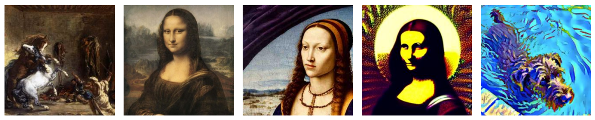

(a) WGA (b) ArtBench (c) Lexica (d) LoFi (e) DALL-E

Fig. 1: Dataset Sources (Resized to 256x256 Resolution)

image search on equally proportioned Artbench data; and we generated 60,000 Low-Fidelity (LoFi) Stable-Diffusion images using text-to-image on each image of the Artbench dataset, using ”[genre] painting of [image name]” as the text prompt. We also drew about 2,100 images from the DALL-E Image Database–a database similar to Lexica but utilizing the DALL-E model instead of Stable-Diffusion to generate AI images–for preliminary experiments. Due to difficulty in getting large quantities of images, this database would later be regulated to testing the robustness of models rather than training. Stable-Diffusion was ultimately preferred for its increased accessibility and search APIs. The proportions for our training/validation/testing data for our datasets varied across experiments. The specific proportions for each dataset were motivated by our both intuition and our planned preprocessing strategies as discussed below.

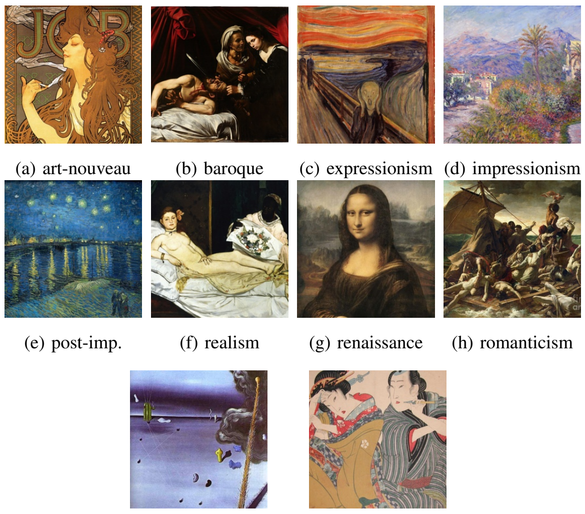

(a) art-nouveau (b) baroque (c) expressionism (d) impressionism (e) post-imp. (f) realism (g) renaissance (h) romanticism (i) surrealism (j) ukiyo-e

Fig. 2: Artbench Art Styles

Dataset 1: WGA + DALL-E

Our first dataset used 5,000 WGA images for our non-AI data and 5,000 DALL-E images for our AI data. This preliminary dataset was meant to assess the validity of using our Convolutional Neural Network infrastructure (discussed in further detail in the Methods section). Our early results returned a 100% validation accuracy on classifying both AI and non-AI images with very little training (one epoch). This exceptionally quick convergence made us suspect that the model was picking up some unknown bias in our data. Namely, we collected the AI and non-AI images from different sources with varying styles, and we only resized the non-AI images.

Dataset 2: Artbench + Low-Fidelity

For our second dataset, we tried to address these issues by using a larger dataset with matching AI and non-AI images. We used all 60,000 Artbench images as our non-AI data, and collected 60,000 Low-Fidelity images using Stable-Diffusion. We made sure that each Artbench image had a corresponding AI image, to minimize potential style biases or genre distribution differences in the datasets. The LoFi images did not need to be resized.

Dataset 3: Artbench + Mixed AI

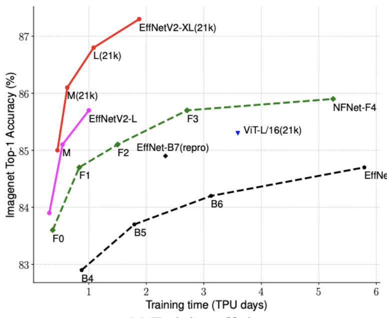
As with dataset 2, we used 60,000 Artbench images for our AI data. However, for this dataset our non-AI data came from two sources: our LoFi generation and the Lexica archive, contributing 30,000 images each. The Lexica images had to be resized to 256x256. The Lexica images provide the advantage of higher quality image generation parameters, but downloading them through the internet introduces compression artifacts.

Dataset 4: Mixed Non-AI + Mixed AI

With the hope of minimizing the significance of image artifacts from image scraping on the internet, we introduced lower quality images to the non-AI art training set, which derived from the WGA. Artifacts would be introduced either from the need to resize WGA images to 256x256 or naturally from image compression on the image at its initial upload time. In this way, we resolved the asymmetry in quality proportions between the non-AI and AI image datasets. The AI image dataset comprised of 55,000 AI images; equally proportioned between Lexica and LoFi images; and the non-AI image dataset comprised of 30,000 Artbench images (equally proportioned between style) and 20,000 resized images from the WGA.

IV. Methods

For our model, we utilize a Convolutional Neural Network (CNN) infrastructure. CNNs were originally inspired by the structure of the anatomical visual cortex. In particular, CNNs specialize in understanding the local relationships between pixels in an image [12], making them the ideal model for this image classification task. More specifically, we made use of EfficientNet, a family of CNNs that can quickly train models with efficient parameter-use [13]. Previous models have used the concept of progressive training by increasing the size of images as training continues. EfficientNet expands on this by also increasing the regularization alongside the images as to increase performance. Furthermore, since not all network stages need to be scaled to improve performance, EfficientNet uses non-uniform scaling for these stages. Finally, EfficientNet adds a scaling rule to limit image sizes, as to prevent memory issues. In particular, we made use of EfficientNetV2. Empirically, EfficientNetV2 has been shown to perform significantly better than other state-of-the-art models, as shown in Figure 4. For our training, we initialized with a pretrained model from torch vision, so the model already had a concept of lower-level features that would speed up training. The specific pretrained model we began with was EfficientNet_V2_S_Weights.IMAGENET1K_V1 from [14]



(a) Training efficiency.

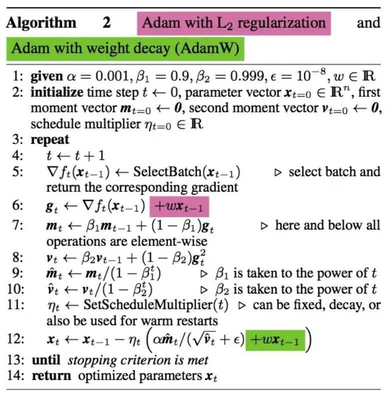
Fig. 3: Accuracy vs. Training Time

We used the special case of cross entropy loss known as log loss, which is a standard loss function for binary classification that measures the closeness of the prediction probability to the correct label in the training examples. A thorough treatment of log loss is presented in the lecture notes, starting at page 20.

Our model utilizes the optimizer algorithm AdamW, an augmentation of Adam [15]. Both algorithms are presented in Fig. 2. The original Adam algorithm incorporates gradient descent with the magnitude of updates based on the similarity of gradients in previous time steps. It does this by maintaining an exponentially moving average of both the gradients m_t and their squares v_t (we then scale these quantities with $1 - \beta_i^t$ to account for bias). Intuitively, Adam takes larger update steps when the gradient doesn't change much between time steps and smaller steps when it does (Recall that $\sqrt{\hat{v}_t} \approx \hat{m}$ if variance is close to 0, implying the gradients are close to constant). AdamW augments Adam by simply moving the regularization term, so that it isn't tracked in the moving average of the gradients. This resolves the asymmetry between the update of the gradient average and the squared gradient average. Empirically, the AdamW algorithm has been demonstrated to generalize much better than Adam, making it the superior optimizer for classification of a dataset that ranges as widely as art.

We kept the specifications of this model consistent for training on all of our datasets in order to adequately compare results (batch size 128, learning rate 1e-6, and default AdamW parameters worked well in every case). Instead of modifying the training model and method, we mostly focused on the preprocessing and distribution of the datasets in an attempt to extract useful and generalizable features from the data.

As an aside, we did try one modification of the achitecture: keeping the convolutional features constant and training a fully-connected network to classify images based on the encoded vector that the convolutional layers produce. To do



Algorithm 2 Adam with L_2 regularization and Adam with weight decay (AdamW)

```
1: given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, w \in \mathbb{R}$
2: initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{x}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
3: repeat
4:     $t \leftarrow t + 1$
5:     $\nabla f_t(\boldsymbol{x}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{x}_{t-1})$    ▷ select batch and return the corresponding gradient
6:     $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{x}_{t-1})$ $+w\boldsymbol{x}_{t-1}$
7:     $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$    ▷ here and below all operations are element-wise
8:     $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
9:     $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$    ▷ $\beta_1$ is taken to the power of $t$
10:    $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$    ▷ $\beta_2$ is taken to the power of $t$
11:    $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$    ▷ can be fixed, decay, or also be used for warm restarts
12:    $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \eta_t \left( \alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) +w\boldsymbol{x}_{t-1} \right)$
13: until stopping criterion is met
14: return optimized parameters $\boldsymbol{x}_t$
```

Fig. 4: Adam and AdamW algorithm [15]

this, we encoded each image into a 1280 element vector using the convolutional portion of pretrained EfficientNetV2_L and trained a fully connected network with 8 million parameters across 4 hidden layers. The theory was that the pretrained model would only encode high-level features, rather than the artifacts that our CNN was picking up. Unfortunately, this idea was abandoned when the model was unable to improve past 50% validation accuracy.

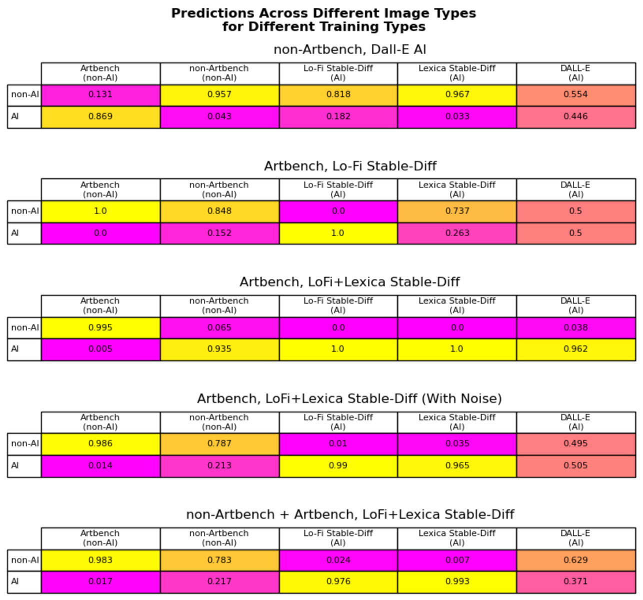
Another idea that we experimented with was converting the images to grayscale. This was intended to mitigate a difference in color saturation between the LoFi and the Artbench data that the model may have been overfitting to. However, this showed little difference in the models generalization ability and was not explored further.

Our final strategy to improve model generalization was to introduce blur and then random noise to the training images. We hoped that the noise and blur would obscure the compression artifacts and other low-level features that the model was overfitting to. We found that a large amount of noise (Fig. 6) was required for the strategy to work - so much that we needed to also validate and test using noisy images, otherwise the test and training datasets were too different for the learned featured to transfer. While using noise during the inference stage is not the typical strategy for computer vision, it turned out to greatly improve our results.

V. Results and Discussion

We found that a standard confusion matrix was inadequate for our analysis, so our primary evaluation metric was an extended confusion matrix (Fig. 5), which demonstrates the accuracy of the model under different testing sources. This is relevant since image quality and generation processes can vary significantly between the different sources. For Fig 6, we include the results for the best model checkpoint of each training method, based on the lowest validation loss.

Dataset 1 (non-Artbench, DALL-E AI) was where we conducted our preliminary experiments, but the trained model seems to reduce to an image quality classifier. This can be seen in Figure 5, where it identifies most images that were compressed and resized as non-AI, and the pristine Artbench images as AI. An interesting outlier is the DALL-E test


Predictions Across Different Image Types for Different Training Types

non-Artbench, Dall-E AI

	Artbench (non-AI)	non-Artbench (non-AI)	Lo-Fi Stable-Diff (AI)	Lexica Stable-Diff (AI)	DALL-E (AI)
non-AI	0.131	0.957	0.818	0.967	0.554
AI	0.869	0.043	0.182	0.033	0.446

Artbench, Lo-Fi Stable-Diff

	Artbench (non-AI)	non-Artbench (non-AI)	Lo-Fi Stable-Diff (AI)	Lexica Stable-Diff (AI)	DALL-E (AI)
non-AI	1.0	0.848	0.0	0.737	0.5
AI	0.0	0.152	1.0	0.263	0.5

Artbench, LoFi+Lexica Stable-Diff

	Artbench (non-AI)	non-Artbench (non-AI)	Lo-Fi Stable-Diff (AI)	Lexica Stable-Diff (AI)	DALL-E (AI)
non-AI	0.995	0.065	0.0	0.0	0.038
AI	0.005	0.935	1.0	1.0	0.962

Artbench, LoFi+Lexica Stable-Diff (With Noise)

	Artbench (non-AI)	non-Artbench (non-AI)	Lo-Fi Stable-Diff (AI)	Lexica Stable-Diff (AI)	DALL-E (AI)
non-AI	0.986	0.787	0.01	0.035	0.495
AI	0.014	0.213	0.99	0.965	0.505

non-Artbench + Artbench, LoFi+Lexica Stable-Diff

	Artbench (non-AI)	non-Artbench (non-AI)	Lo-Fi Stable-Diff (AI)	Lexica Stable-Diff (AI)	DALL-E (AI)
non-AI	0.983	0.783	0.024	0.007	0.629
AI	0.017	0.217	0.976	0.993	0.371

Fig. 5: Performance across models trained on different sources. Cells represent the percentage of column-typed images that were predicted to be of the row-class. The true class of each type is seen in parenthesis in the column headers.

set, which was split in accuracy, possibly due to DALL-E's inclusion in the training data. Also supporting the theory that image quality was the defining feature is the fact that training converged to 100 % training accuracy after 1 epoch, so the model must have learned 'obvious' features. However, these results are inconclusive since the dataset was considerably smaller than in future experiments, and image quality was much less controlled.

Dataset 2 (Artbench, LoFi Stable-Diffusion) was able to accurately identify all non-AI images and LoFi AI images. However, it inaccurately classified Lexica as non-AI and performed as guessing for DALL-E. In this way, the model seems to have again reduced to a fidelity classifier, assigning the AI label to the lowest fidelity images (the LoFi data), with everything else being given the non-AI label. This makes sense, since the non-AI image training data for this trial was considerably higher fidelity than the AI image training data. Furthermore, like the previous dataset, this training converged to perfect training accuracy in 1 epoch. Interestingly, the model performs on DALL-E about as well as the previous model, which implies that some of our DALL-E images were lower fidelity than we had realized.

Dataset 3 (Artbench, LoFi+Lexica Stable-Diffusion) only identified Artbench as non-AI and identified every other case as AI. Here, the model appears to have again reduced to a compression artifact classifier. Since the high-quality Artbench data was classified as non-AI and everything else as AI (DALL-E again being an outlier). This is the opposite of Dataset 1, which classified high-quality images as AI. Like datasets 1 and 2, this training converged after 1 epoch. The results of datasets 1 and 3 imply a need to account for the quality imbalances in the data that the model is overfitting to. However, it was promising that the model is no longer over fitting to fidelity after introducing more balanced AI data. We address the quality problem with two different preprocessing strategies, discussed in the next two datasets.

Dataset 3a (Artbench, LoFi+Lexica Stable-Diffusion with noise) performs considerably better than the previous datasets. As seen in figure 5, this model was accurate on most of the data types, while slightly underperforming on non-Artbench non-AI images and being inconsistent on DALL-E. The specific preprocessing used was Gaussian noise with $\sigma = 0.1$ and Gaussian blur with kernel size 7 and a standard deviation of 2. We chose these values since any more distortion leads to destruction of the image, while lower values were not able to fully obscure processing artifacts and did not prevent overfitting. It is notable that noise was applied during both training and inference time, as this made performance much more predicable. The success of this training method shows that we can prevent low-level overfitting using noise, and that forcing the model to look at high-level performance improves performance. We also note that this method still underperformed on DALL-E, which will be discussed further in the Discussion section.

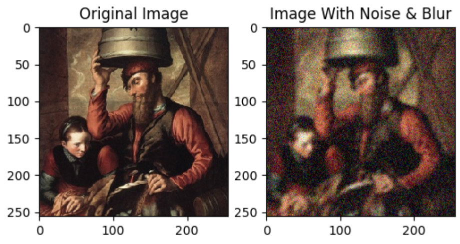


Fig. 6: Noise and Blur Image Comparisons

Dataset 4 (non-Artbench + Artbench, Lofi+Lexica Stable-Diffusion) applies no transformation to the images, but uses the most diverse training data - balancing quality and fidelity across AI and non-AI sub-sets. This supports our conclusion that quality balance and preventing low-level overfitting is key for the success of the classifier. More interestingly, this dataset's close similarity to the noisy Dataset 3a implies that data collection can be replaced by preprocessing transformations to prevent overfitting. What's also notable is its performance on the DALL-E dataset: all of the other datasets perform about as well as guessing on DALL-E, implying that DALL-E images have a unique structure compared to the others; this dataset led to a negative correlation between DALL-E images and AI classification, meaning that the features that this model learned are specific to stable-diffusion, and not generalizable (the opposite in fact) to other forms of generation. Another note is that this model's performance on DALL-E got worse with more training over time, reinforcing the idea that the model was fitting to features that correlate differently between Stable-Diffusion and DALL-E generation.

We noted the accuracy and loss of models that were trained on Datasets 3a (with blur/noise) and Dataset 4. For the former, training loss converged to 0.0297 and training accuracy converged to 0.989 (Fig 7). For the latter, training loss converged to 0.021 and training accuracy converged to 0.993 (Fig 8). The most pertinent observation is the smoothness of the data: although both have accuracy graphs that are relatively smooth throughout training, the model trained on Dataset 4 has considerably less stable loss before it converges while Dataset 3a is relatively smooth throughout. This is probably the case since Dataset 4's model had to directly reconcile with noise during training while Dataset 3a's model obfuscated all of it. The similarity of the accuracy curves, however, implies that both models are isolating the same features. A more specific analysis of this is aided by GradCAM.

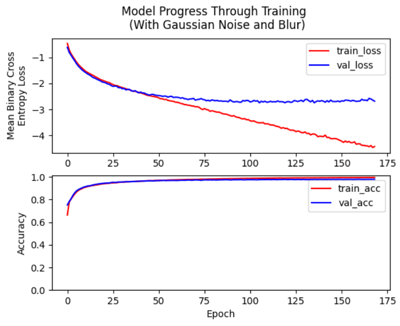


Fig. 7: Dataset 3 With Blur/Noise Accuracy and Loss

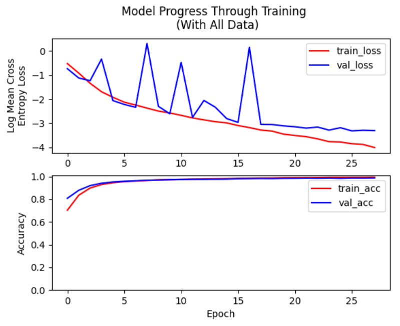
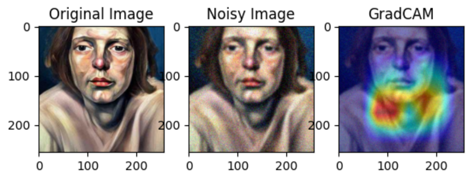


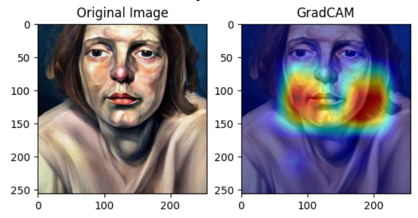
Fig. 8: Dataset 3 Without Blur/Noise Accuracy and Loss

Gradient-weighted Class Activation Mapping (GradCAM) is a visual explanation algorithm that used to make CNNs more transparent [16]. GradCAM uses gradients flowing into the last convolutional layer of the target classification label to produce a heat-map and identify the most important regions used in the classification. In particular, we used the EigenCam version. EigenCam focuses on the first principle component of the 2D activations. Although there's no class discrimination, it gives empirically strong results [17].

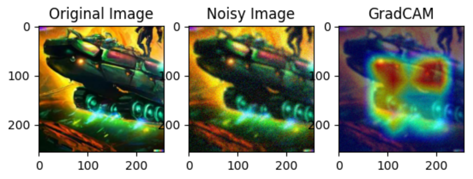
Comparing the GradCAM heat-maps produced for these models reveals that when both models classified an image in the same way, the GradCAM heat-maps tended to be very similar (i.e. Fig. 9). In the case where the models classified images differently, the heat-maps tended to identify different significant regions (Fig. 10). This implies that there are common features that both models are picking up on - likely features that are universally useful for picking out Stable-Diffusion images from real ones. The difference in predictions and heat-maps shows that the different data methods force the models to learn slightly different features - in the case of Dataset 4, features that correlate negatively to DALL-E.



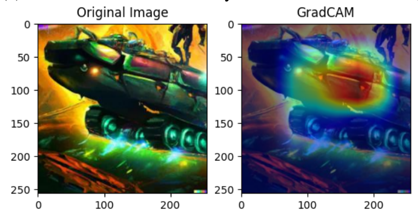
(a) Dataset 3a model correctly identifies Stable-Diffusion Image



(b) Dataset 4 model correctly identifies Stable-Diffusion Image
Fig. 9: GradCAM



(a) Dataset 3a model correctly identifies DALL-E Image



(b) Dataset 4 model misdentifies DALL-E Image
Fig. 10: GradCAM

VI. Conclusion and Future Work

We have shown that data selection preprocessing has significant impact on model performance. Specifically, we show that without consideration, models trained to classify whether images will AI generated tend to overfit to non-generalizable low-level features and artifacts. We then show 2 methods to mitigate this problem: introducing noise to obscure low-level features, and using large well-balanced training datasets that have symmetrical processing between classes. We also show that classifiers trained to identify images from one specific generation method do not necessarily generalize to all generation methods.

For future work, unsupervised learning approaches could be useful to collect large datasets. Additionally, we could expand the scope of our problem and train on images from multiple generation sources, possibly detecting the original model that created the AI-image, similar to Hive's API. Our project primarily focused on using a pretrained CNN, but with more time it might be valiable and valuable to train a new CNN from scratch or explore different NN architectures.

VII. Contributions

The standard CNN model architecture was primary strategized and designed by Adam Klein. Adam further worked on functions to accurately sort data for training, testing, and validating purposes. He also introduce the graphical tools used for analysis such as the plots and GradCam. The primary architecture for web scraping used for data collection was designed by Matthew Villescas. Matthew and Adam both worked on functions necessary for data collection from the various databases. The resizing architecture was built by Alex Hodges. Adam and Alex both focused on functions for general image preprocessing. Adam, Alex, and Matthew all contributed strategies to move forward with the classification and assisted in the analysis of the results. All members fought desperately to get Google Cloud to work properly.

VIII. References

[1] Birhane, Abeba, et al. ”Multimodal datasets: misogyny, pornography, and malignant stereotypes.” arXiv, 2021, https://doi.org/10.48550/arXiv.2110.01963. Accessed 9 Dec. 2022.

[2] Heikkilä, Melissa. “This Artist Is Dominating AI-Generated Art. and He’s Not Happy about It.” MIT Technology Review, MIT Technology Review, 23 Nov. 2022, https://www.technologyreview.com/2022/09/16/1059598/this-artist-is-dominating-ai-generated-art-and-hes-not-happy-about-it/.

[3] Wang, Sheng, et al. ”CNN-generated images are surprisingly easy to spot... for now.” arXiv, 2019, https://doi.org/10.48550/arXiv.1912.11035. Accessed 9 Dec. 2022.

[4] Girish, Sharath, et al. ”Towards discovery and attribution of open-world gan generated images.” Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

[5] Sha, Zeyang, et al. ”DE-FAKE: Detection and Attribution of Fake Images Generated by Text-to-Image Diffusion Models.” arXiv, 2022, https://doi.org/10.48550/arXiv.2210.06998. Accessed 9 Dec. 2022.

[6] Hive. “Detect and Moderate AI-Generated Artwork Using Hive’s New Api.” Hive, 2 Nov. 2022, https://thehive.ai/blog/detect-and-moderate-ai-generated-artwork-using-hives-new-classification-model.

[7] Krén , Emil, and Dániel Marx. “Web Gallery of Art.” Web Gallery of Art, Searchable Fine Arts Image Database, 1996, https://www.wga.hu/index_welcome.html.

[8] Liao, Peiyuan et al. ”The ArtBench Dataset: Benchmarking Generative Models with Artworks”. arXiv preprint arXiv:2206.11404. (2022).

[9] Lexica. 24 Aug. 2022. Web. Accessed 8 Dec. 2022.

[10] Robin Rombach, et al. ”High-Resolution Image Synthesis with Latent Diffusion Models.” (2021).

[11] DALL-E 2 Image Database. https://dalle2.gallery/search. Web. Accessed October 17.

[12] Saha, Sumit. “A Comprehensive Guide to Convolutional Neural Networks-the eli5 Way.” Medium, Towards Data Science, 16 Nov. 2022, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[13] Tan, Mingxing, and Le, Quoc. ”EfficientNetV2: Smaller Models and Faster Training.” arXiv, 2021, https://doi.org/10.48550/arXiv.2104.00298. Accessed 9 Dec. 2022.

[14] “Models and Pre-Trained Weights¶.” Models and Pre-Trained Weights - Torchvision Main Documentation, 2017, https://pytorch.org/vision/stable/models.html.

[15] Loshchilov, Ilya, and Hutter, Frank. ”Decoupled Weight Decay Regularization.” arXiv, 2017, https://doi.org/10.48550/arXiv.1711.05101. Accessed 9 Dec. 2022.

[16] Selvaraju, Ramprasaath, et al. ”Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.” arXiv, 2016, https://doi.org/10.1007/s11263-019-01228-7. Accessed 10 Dec. 2022.

[17] Gildenblat, Jacob. “PyTorch Library for CAM Methods.” GitHub, Github, 2021, https://github.com/jacobgil/pytorch-grad-cam.

[18] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[19] McKinney, W., others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).