A tuning fork on a chalkboard: frequency space perturbations of images for adversarial dataset creation

Arjun Sharma
asharma3@caltech.edu

Abstract

Following the guidelines for project 1a of CS 148a, I produce a dataset of 100 images of digits with the adversarial intention of obstructing the inference-time performance of deep neural networks. Two perturbations are introduced to achieve this effect.

Firstly, directly following the guidelines, images are produced in a physical setting (negative space of chalk on a chalkboard) that is hypothesized to be out-of-distribution for a standard classifier. The second key perturbation — the main focus of this report — draws on previous literature on convolutional neural newtorks to perform frequency-domain modification of images, with the intention of introducing changes that are minimally perceptible to humans and maximally peceptible to the feature extractors of a deep classifier.

We divide frequency space into a finite number of buckets and assign tunable weights to each bucket. We then perform an approximation of gradient ascent on a dissimilarity metric between deep 'features' extracted from 'in-distribution' images and adversarial images. The 2-D Discrete Fourier Transform of adversarial images is taken in each iteration, and the magnitudes of the frequencies in this spectrum are multiplied by the current weight corresponding to their bucket. The image is reconstructed to the time (visual) domain through the Inverse Fourier Transform, deep features are extracted with an off-the-shelf feature extractor (MobileNet V2), and the similarity of these features to the ground-truth is computed over each digit.

This approach results in meaningful changes to the frequency space and the corresponding similarity between the features extracted, but causes no significant degradation to the recognizability of the digits in those images, which is guaranteed both through quantitative and qualitative means.

Unlike typical approaches to adversarial image cosntruction, we do not explicitly rely on the black-box outputs of classifiers as a supervision signal for constructing out-of-distribution images. Relatively conservative parameters are adopted for modification under significant computational constraints, with there being further scope to investigate the limits of this approach in destroying the performance of deep neural networks for image classification.

1 Introduction

Supervised deep neural network image classifiers are trained using examples of images from the provided classes, with the classical assumption that these training images, in aggregate, are a good approximation of the lower-dimensional manifold in image space where the true distribution of images of handwritten digits lives [6]. These networks implicitly learn representations of features in images provided to them during training and rely on extracting these features during inference to classify inputs. Since no explicit human intervention is part of the training process, classifiers networks may learn features that are sufficient to distinguish between classes but which are not easily interpretable to humans.

Convolutional neural networks (CNNs) are a common class of networks which achieve high performance on image tasks. Literature [9, 1, 8] has shown that these architectures have a tendency to find 'simple solutions' [9] for classification, which can be understood in the **frequency domain**. [8] argue that CNNs particularly capture high-frequency components, whose impact on the final image are almost imperceptible to a human. Learning high-frequency features such as texture, [9] argues, may be extremely beneficial in some settings during training. [8]they show through experiments that removing high-frequency features from an image meaningfully degrades the performance of cutting-edge networks while having negligible impact on the appearance of the image to humans.

In this report, I describe an attempt to perturb the frequency domain of images that could be used as a test set for a hypothetical CNN classifier of handwritten digits, while ensuring that a human could still be reasonably expected to identify them. This attack, unlike typical adversarial approaches, does not require access to the outputs of the target classifier; its strongest assumption, following literature [5], is that a generic CNN should learn similar features to the target CNN.

2 Method

Due to time constraints, I describe an algorithm without significant motivation or mathematical rigor here.

Let X be the space of grayscale images of size 128×128. Let G be the dataset of ground-truth images, which we assume to be 'in-distribution' for the classifier; and P be the images which we can modify, and try to make out-of-distribution. Let $F : X \to \mathbb{R}^n$ be the feature extractor, which maps from images to some continuous high-dimensional vector space. $\mathcal{F}$ is the 2-D discrete Fourier transform of an image, which maps it to frequency space, and $\mathcal{F}^{-1}$ is the inverse Fourier transform, which maps 2-D frequencies and magnitudes back to images.

Divide frequency space into B buckets based on radial distance from the DC component, using log-spaced boundaries. Initialize $W = [1, 1, \cdots, 1]$, a vector of length B. Let M_b denote the boolean mask selecting frequencies in bucket b. For each digit class $d \in \{0, \ldots, 9\}$, we optimize a separate weight vector $W^{(d)}$.

1. **Feature extraction.** Compute $F(g)$ for all ground-truth images $g \in G_d$ of digit d, yielding a set of feature vectors $\{F(g)\}_{g \in G_d}$.
2. **Frequency-domain modification.** For each image $p \in P_d$:
   (a) Compute $\hat{p} = \mathcal{F}(p)$, the 2-D DFT of p.
   (b) Decompose $\hat{p}$ into magnitude $|\hat{p}|$ and phase $\angle\hat{p}$.
   (c) For each bucket b, multiply magnitudes by the corresponding weight: $|\hat{p}'|[M_b] = W_b \cdot |\hat{p}|[M_b]$.
   (d) Reconstruct the modified spectrum: $\hat{p}' = |\hat{p}'| \cdot e^{i\angle\hat{p}}$.
   (e) Compute the modified grayscale image: $p' = \mathrm{Re}(\mathcal{F}^{-1}(\hat{p}'))$, clipped to $[0, 1]$.
3. **Dissimilarity computation.** Extract features $\{F(p')\}_{p \in P_d}$ from the modified images. Compute the Maximum Mean Discrepancy (MMD) between the two feature distributions [3].
4. **Gradient estimation via central finite differences.** For each bucket b:

$$\frac{\partial \mathrm{MMD}}{\partial W_b} \approx \frac{\mathrm{MMD}(W + \epsilon e_b) - \mathrm{MMD}(W - \epsilon e_b)}{2\epsilon}$$

   where e_b is the b-th standard basis vector and $\epsilon = 0.05$ [4].

5. **Gradient ascent.** Update weights to *maximize* dissimilarity:

$$W \leftarrow W + \eta \cdot \nabla_W \text{MMD}$$

where η follows a log-scale decay from $\eta_0 = 1.0$ to $\eta_T = 0.01$ over $T = 15$ iterations. Weights are clipped to $[0.1, 3.0]$.

6. **Iterate.** Repeat steps 2–5 for T iterations, storing the weight history and dissimilarity at each step.

The feature extractor F is MobileNetV2 [7] pretrained on ImageNet, with the classification head removed to yield 1280-dimensional feature vectors. Ground-truth MNIST images are upscaled from 28×28 to 128×128 using Lanczos interpolation to match the resolution of the custom images.

3 Experiments

Ten images each were collected for each of the digits 0-9, resulting in a **custom dataset** of 100 images. Efforts were made to make these images intrinsically adversarial to classifiers, even before frequency-domain modifications. A chalkboard was covered in chalk through running pieces of chalk[1] flat against it. A chalkboard eraser was used to create negative-space shapes of digits on the chalkboard. The motivation for this method was to reduce contrast in the images and noise the entire image with high-frequency patterns, with useful high-frequency features like edges not being distinctly visible. While the resulting negative space digits were clearly visually perceptible to humans, the empty space which created the digits is prominent in small, unstructured patches across the image due to inconsistency in the background chalk; therefore, a model which extracts local features should struggle to identify the components of a digit. Additional examples, details and evidence about data collection as well as postprocessing are in Appendix A.

Due to limited time, I include the key figures and statistical results from my experiments, with some qualitative discussion. A compressed codebase folder is available here[2].

We divide frequency space into the buckets $[0, 2, 4, 8, 16, 32, 64, 90]$. We use 75 iterations of updates, having tested 10 and 100 as well. Experiments take around 20 minutes on a 2025 M5 MacBook Pro, using `mps` acceleration for MobileNet inference with PyTorch and `numpy` for fourier transform operations.

Ground-truth 28×28 pixel MNIST images of handwritten digits were collected from an instance hosted on Kaggle [2]. These are treated as 'in-distribution', with the assumption that the target classifier has been trained on this or a similar dataset.

[1] With apologies to the chalk collection of the third floor of Linde Hall. Every effort was made to restore the room to its original state after.

[2] As discussed in Appendix B, large language models were used extensively to assist with the generation of code in this project, and their signature is visible in the linked codebase

3.1 Qualitative view of changes

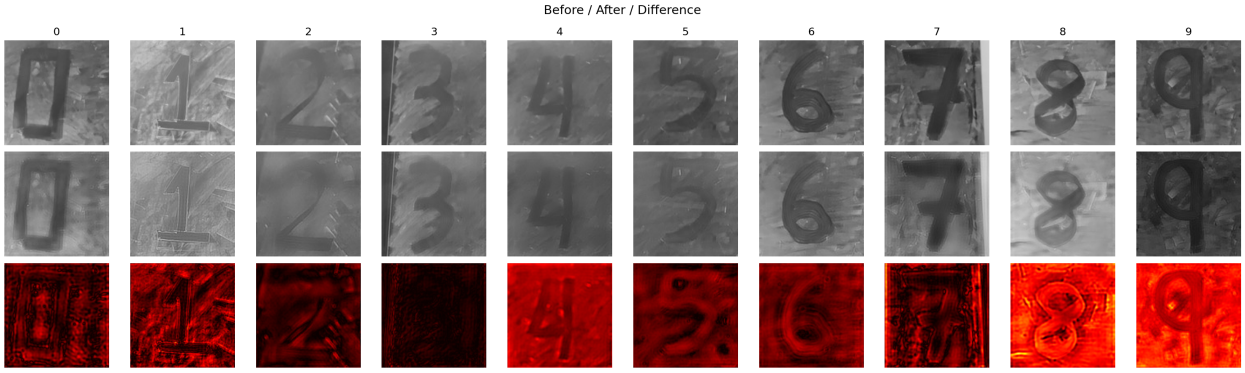
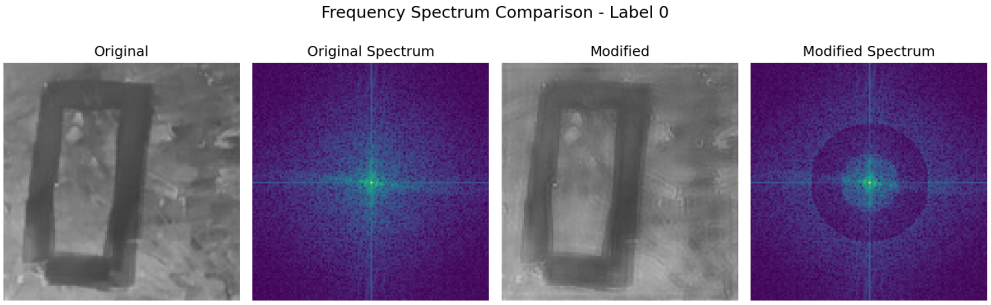
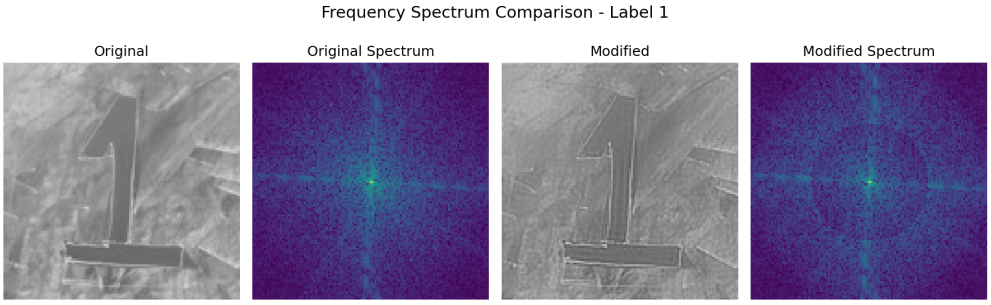


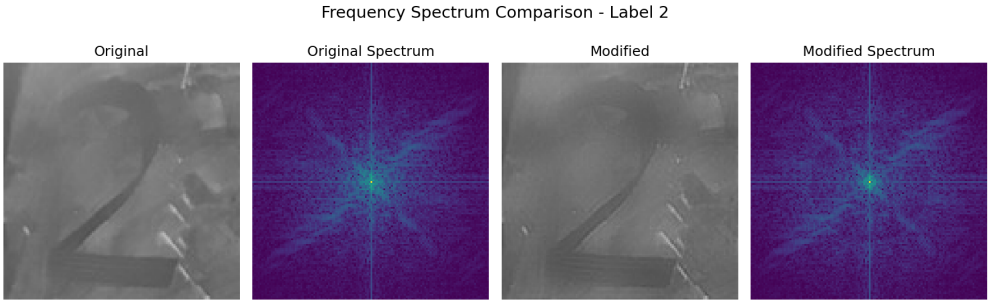
Figure 1: The top row is the original set of images photographed from the chalkboard, the middle row are the images after processing through the frequency perturbation pipeline, and the bottom row is an exaggerated visualization of the difference between the two images. Notice the minimal changes to the visible images but significant frequency domain change.



(a) Label 0



(b) Label 1



(c) Label 2

Figure 2: Example images before and after frequency-domain modifications. Minimal differences are visible in the time (spatial) domain, but significant changes have occurred in the frequency domain, with some frequency bucket weights (visible as concentric rings) having been pushed to near zero.

3.2 Quantitative view of changes

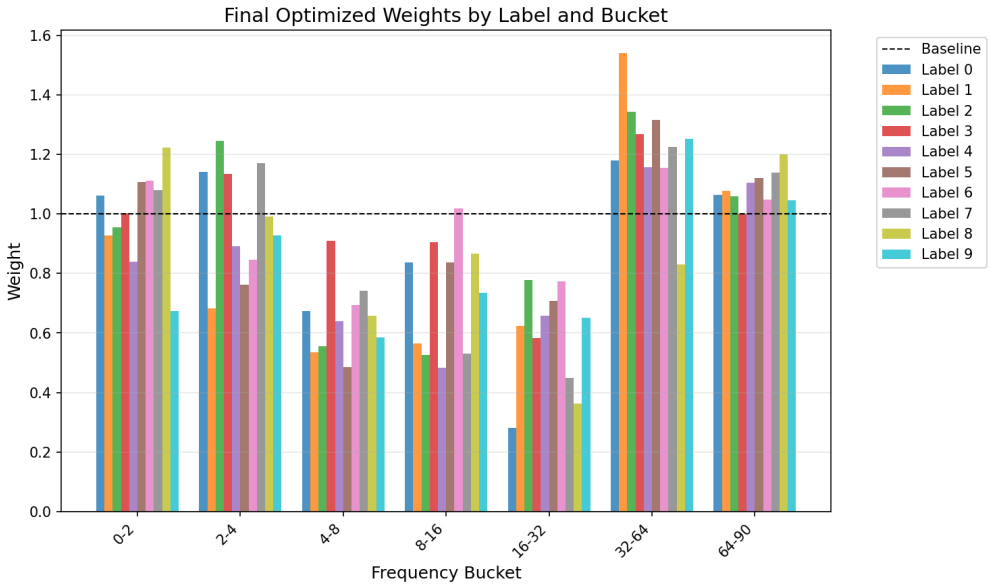


Figure 3: Weights over time. Significant deviations have occurred from the baseline. Note in particular that we converge naturally to large deviations to weights on higher frequencies (above 4) and minimal deviations to weights on low frequencies by trying to maximize dissimilarity against MobileNet features, which is exactly what previous literature predicts.

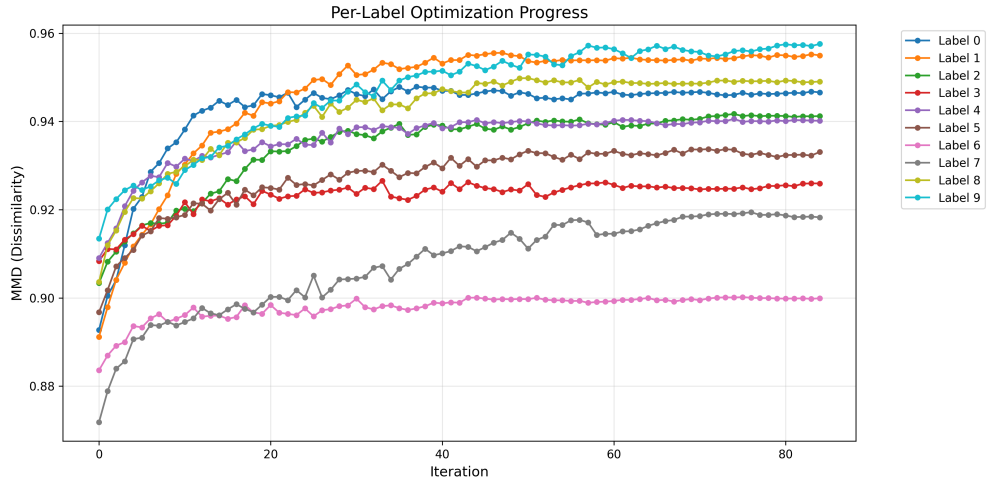


Figure 4: Dissimilarity (Maximum Mean Discrepancy) over over iterations of each image class, where MMD is taken between all the ground-truth images of a particular digit against all the perturbed images of the same digit. Lower MMD corresponds to more similar datasets.

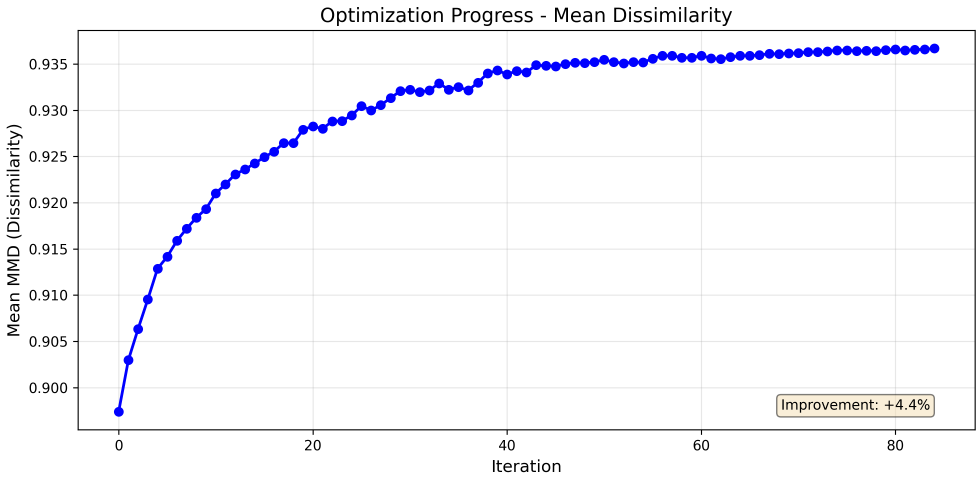


Figure 5: MMD reduces over time since we use a log-scale learning rate update. We achieve an overall 5% MMD shift

4 Conclusion

This approach allows us to make frquency perturbations that result in MobileNet, our representative CNN feature extractor, extracting different features of our images, even while they remain similar upon viewing by humans. Simple CNNs should hopefully struggle even more to identify similar features. Our approach requires no a priori access to the target CNN and is computationally cheap.

Remarkably, without enforcing the view that high frequency features are important to CNNs, our gradient ascent automatically changes the high frequency part of the spectrum most significantly, which exactly matches prior literture.

Still, our approach has limitations. We did not extensively explore dissimilarity comparisons, and MobileNet may not be the best representative feature extractor. We divided the frequency domain into a very small number of buckets and more detailed divisions would result in better perturbations. We were extremely conservative with perturbations, applying small updates over a small number of iterations. A reliable quantitative measure of 'human perceptibility' would have been valuable as that would have allowed joint optimization of both metrics, without having to abstractly worry about destroying human perceptibility. This idea of joint optimization may have parallels to contrastive learning.

We were extremely data-limited, with only 10 images per set, which restricted our ability to learn good frequency magnitudes for that digit. It is also likely that it would have been better to tune frequency magnitudes for each individual digit.

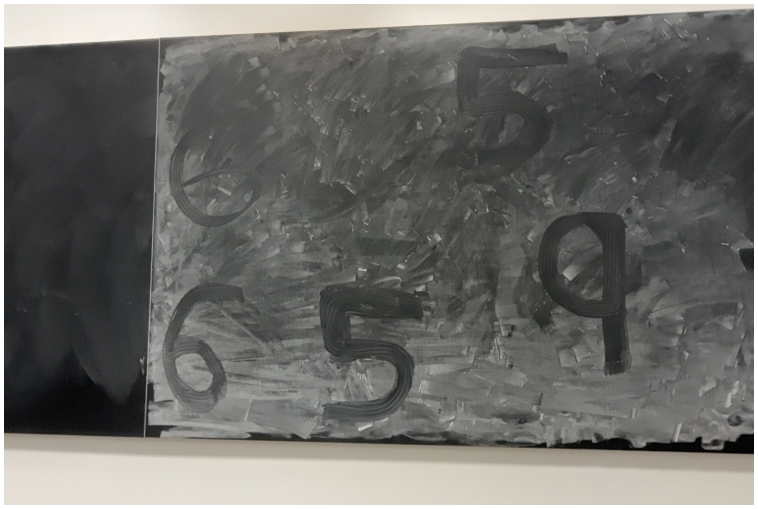


Figure 6: An example of digits drawn in the negative space on the chalkboard. Manual review of all images was conducted to ensure legibility when cropped, which resulted in the exclusion of, for example, the the two digits in the top half of this image.

A Image collection

It was possible to write 4-8 digits at a time on the third of the the chalkboard used for data collection. A Macbook Pro was placed on the stand in front of the chalkboard and its camera application (Photo Booth) was used to take photos of the chalkboard. Images were saved and each digit in the image was was extracted to a separate image and cropped to a square using the Preview app's inbuilt cropping tool [3]. The `sips` utility was used to perform aspect ratio-preserving resizing of these cropped images (which were all squares of variable lengths) to 128×128 pixels, with the following command:

```
sips -Z 128 *.jpg
```

B Use of large language models

I used Claude Opus 4.5 to implement and improve my plan for this project. I provided it a 375-word prompt describing my idea and motivation, and asked for suggestions with specific parts of the pipeline. I accepted its suggestions in many parts (most notably, I used its proposed method for gradient estimation using finite differences, and measuring dissimilarity against ground-truth by using MMD between MobileNet features). After this, I got it to generate the code I required for the project, providing it a description of the kinds of plots and safeguards I wanted.

I found its analysis and predictions to be rather lackluster and often incorrect; however, it was extremely useful in writing the large amount of code necessary for this project, as well as helping in surveying literature to identify tools that would best serve my requirements for in this project. Some of the code it suggested required modification. A significant component of the pipeline (backtracking upon significant drops in recognizability) that it included was removed upon noticing its negative impact on performance and negligible importance to recognizability.

No part of this document was written with a language model.

[3] Holding the shift key while creating a bounding box guarantees that the box is a perfect square, and Cmd + K crops the image into the bounding box.

All prompts I provided are listed below. The complete conversation log is attached here.

B.1 Exact prompts provided

Assume we have 2 directories 1. A 'ground-truth dataset', a directory with ground-truth MNIST images, divided into sub-directories for each digit. Each sub-directory contains M MNIST images of that digit. Assume for reasoning that M is ≈ 50-100 2. A 'custom dataset', a directory with new images, divided into sub-directories for each digit. Each sub-directory contains 10 images of that digit, so we have 100 total images.

I want to edit the frequency spectrum of the images in the custom dataset to maximize their dissimilarity to the ground truth.

My measure of dissimilarity for a given digit is the KL divergence between the top K principal components of the images for that digit in the ground truth vs the images in the custom dataset for that digit.

Segment the frequency spectrum into b buckets on an appropriate scale (help me pick a scale, eg. log or linear). For each i-th image in the custom dataset, define w_{ij} as the weight we multiply with all frequencies in the the j-th bucket of the frequency spectrum. Initialize these to 1.

This means that, for a given image, we have $10 \times b = 10b$ parameters we can modify. I want to update these weight parameters, calculate the inverse fourier transform of the modified frequency spectrum after rescaling the magnitudes of each frequency by the corresponding weight, and calculate similarity. I want to use some kind of minimization algorithm (propose suggestions - I'm trying to think about whether gradient descent would work here) with a fixed number of iterations (15) and step size (log scale, decreasing over time), to edit the frequency spectrum such that the similarity decreases.

The point of this is to create an adversarial dataset for a neural network trained on MNIST. Literature i've read suggests that removing high frequency components / amplifying low frequency components results in little change to human eyes but drastically hurts CNNs. I want to use this PCA similarity analysis to naturally modify the dataset to mimic the same adversarial behaviour

Help me think about: 1. How to segment the frequency spectrum - what's the right scale 2. An algorithm to reduce similarity by tuning the weights 3. Whether this approach should work

Then I'll need help writing a program - not right now.

Explain, programmatically if necessary, how estimating gradient with finite differences would work. Also, why do you take $w_{ij} \pm$ epsilon instead of just one sign? How does this impact the update you make?

Instead of KL divergence on PCA, I am willing to use feature space distance - suggest a realistic approach for calculating features with a pretrained CNN that I could evaluate in reasonable time on my M5 Macbook Pro, and a good way to compare distributions - FID sounds interesting.

Also suggest a measure for 'human recognizability' that isn't a classifier - the whole point is to make sure that classifiers perform poorly.

Suggest a plan for the entire pipeline. I'll edit and approve it, then we can work on code.

Per digit weights Take central differences Use MMD for feature distance Process digits independently Hard constraint with backtracking Assume I will use an .ipynb for running it, but organize code into other helper files / scripts as necessary. Everything should be called from the main ipynb.

Add good visualizations - after each iteration, for example, save the currently modified images. Show and save feature similarity (ie loss) curves.

Assume all the images will be 256x256 px, but make the code as generalizable as possible. I'll use something like the HWD dataset instead of MNIST. Looks good - go ahead and write the code

---

I need to collect these images first. I want to make the images intrinsically adversarial but I don't want my postprocessing with frequency edits to cancel out these intrinsic problems.

I'm thinking I"ll draw them on a chalkboard which has lots of white residue behind, and draw each digit with disconnected lines with lots of whitespace, that are labellable by humans but difficult for CNNs due to their locality.

Is this a good idea? What'll happen to this if we edit the frequency spectrum?

References

[1] Antonio A Abello, Roberto Hirata, and Zhangyang Wang. Dissecting the high-frequency bias in convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 863–871, 2021.

[2] Stuart Colianni. Mnist as .jpg. https://www.kaggle.com/datasets/scolianni/mnistasjpg, 2015.

[3] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

[4] Feng Kong. Python programming and numerical methods: A guide for engineers and scientists. https://pythonnumericalmethods.studentorg.berkeley.edu/notebooks/Index.html, 2020.

[5] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

[6] Christopher Olah. Neural networks, manifolds, and topology. https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[8] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8684–8694, 2020.

[9] Shunxin Wang, Raymond Veldhuis, Christoph Brune, and Nicola Strisciuglio. What do neural networks learn in image classification? a frequency shortcut perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1433–1442, 2023.