On the Detection of GAN-Generated Facial Imagery

Harshal Agrawal Ricky Parada Colin Sullivan

Abstract

The rise of generative AI models such as Sora and Midjourney that can create human-level images and video content poses a significant threat to the well-being of our democracy and the livelihoods of creators. Malicious actors can wrongly utilize these technologies to spread misinformation and impersonate others without their consent. In this research, we attempt to distinguish AI-generated synthetic content from real images, with a focus on facial imagery, as it is the most consequential for preventing wrongful impersonation. We approach this problem with three models: a single-layer fully connected neural network, a custom shallow neural network, and a fine-tuned ResNet50 architecture. Each model is trained on 64p facial images from the Kaggle-140k-Real-Fake-Faces dataset to output an image classification as real or fake. The models were trained on both augmented (randomly flipped, rotated, color adjusted, and normalized) and non-augmented normalized versions of images, as well as 2D discrete Fourier transformations of the images. Testing on three different test datasets suggests that no singular method outperforms any other, though the shallow CNN seems to perform the strongest overall.

1. Introduction

The AI revolution was (and is still) expected to threaten blue-collar work, but its largest impact thus far has hit an unlikely target — creators [24]. Artists, journalists, photographers, and even videographers have begun to face significant competition from generative AI models such as Midjourney, ChatGPT, and Sora, which can generate human-level image, text, and video content, respectively. These creators have shot back by suing the developers of these models for their use of the creators' work in training with varying degrees of success (e.g., NYTimes [18]). With policymakers scrambling to regulate this new technology, concern about the impact of misinformation (e.g., Twitter bots, deep fakes) on the upcoming election, and the alarming spread of AI-generated internet content, a crucial question arises: can we differentiate real from synthetic content?

1.1. Problem Statement

There exist many modes for which AI image content can be generated (e.g., cartoons, scenic photographs, paintings, facial imagery) and many potential methods across each of these modes (e.g., Generative Adversarial Networks (GANs), diffusion models). We restrict our focus in this project to facial imagery because it appears most consequential, considering the potential impact of impersonation on the upcoming election and the already-realized effect on people's lives (synthesized "revenge porn" [12], identity theft, etc.). We seek to distinguish AI-generated facial imagery from real facial imagery using the following trio of models:

1. Single Layer Fully Connected Neural Network (NN)
2. Custom Convolutional Neural Network (CNN)
3. Fine-tuned ResNet50 architecture. [8]

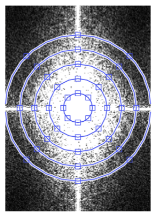
Each model acts as a discriminator, taking in a 64p facial image and outputting whether or not the image was generated by AI. The single layer and custom CNN models were additionally trained to take in an image in the Fourier domain (that is, the image after applying a 2D discrete Fourier transform) based on a related work's increased performance on transformed data (see the Related Work/Dataset and Preprocessing sections for more details).

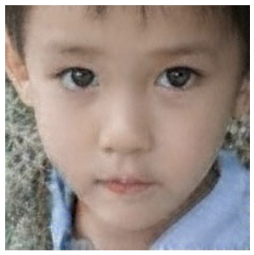
2. Related Work

In this section, we discuss several prior approaches to facial imagery discriminators and comment on how our work compares to the existing literature.

Image quality assessment based fake face detection [25] This work proposes a novel method to detect forged faces that trains a Random Forest (RF) classifier on Image Quality Assessment (IQA) based features. The approach is based on the hypothesis that the appearance of real and fake images is quite similar, so most of the discriminative information is available in the frequency and spatial domain of these images. As seen in Figure 1, visualizations of the difference in magnitude of the frequency domains between real and fake images appear to validate this idea. Further, the

(a) Real Face Image


(b) Magnitude of FFT of Real Face


(c) Fake Face Image

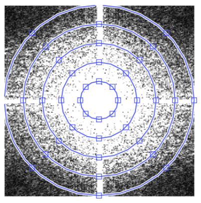

(d) Magnitude of FFT of Fake Face

Figure 1: Sample real/fake image and their respective Fast Fourier Transform (FFT) magnitudes [4]. We observe a significant difference between real and fake images in the magnitude of their respective frequency domains.

most utilized feature, or the feature with the highest Shapley additive explanation (SHAP value) [16] is based on the frequency domain of input images. The approach achieves a reported 99% accuracy on a varied combined dataset of real images from CASIA (celebrities) [32] + VGGFace2 (regular people of various professions) [5], and fakes images from iFakeFaceDB (pulled from the This Person Does Not Exist dataset and modified by a GAN finger print remover) [21]. Performance on wild data suggests that the approach generalizes well, which motivated our training on images in Fourier space.

GAN is a friend or foe?: a framework to detect various fake face images [29] As part of their NN based proposed framework for detecting adversarially constructed fake facial imagery, FakeFaceDetect, the authors try several classifier models, including three different shallow CNN architectures and several fine-tuned deeper models. The authors note that, surprisingly, the shallow CNN classifiers, ShallowNetV1/2/3, trained from scratch perform significantly better than deeper fine-tuned approaches such as XceptionNet [6] and VGG19 [26]. These larger models perform notably worse on the lower-fidelity 64p images than the proposed ShallowNet models. The data used in training and evaluation is pulled from the CelebA (celebrity images) [15] and PGGAN (PG-GAN generated celebrity images trained on the CelebA dataset) [10] datasets. To see if our findings match the authors, we also compare the results from a shallow CNN against a fine-tuned ResNet50 [8] model.

Fake Face Detection Methods: Can They Be Generalized? [13] This work provides a new dataset, Fake Face in the Wild (FFW), of 53k images from 150 videos, originating from multiple sources of digitally generated fakes, including CGI generation, and the commonly used Swap-Face application. The authors make various attempts to classify this data, using both pre-trained deep CNNs (AlexNet [14], VGG19, ResNet50, Xception, GoogLeNet/Inceptionv3 [28]) and an SVM with Local Binary Patterns (LBPs) as features [22]. While not particularly impressive, Xception appears to generalize best to new "unknown" data, but the LBP-featured approach also performs surprisingly well. The authors suggest that future research be directed toward the generalizability of fake facial imagery detection methods, suggesting that many current CNN-based approaches may fail to succeed in practice despite impressive experimental results. To determine how well our discriminator generalizes, we evaluate each model on the FFW dataset as a final evaluation.

3. Methods

Here we provide an overview of our loss function and the details of each proposed model architecture. To develop these models, PyTorch, which is an open-source machine learning library developed by Meta AI, and subsequent functions/packages such as torchvision were used [23].

3.1. BCE loss

We used Binary Cross Entropy (BCE) as our loss function during training for all three models: Fully Connected Linear NN, Custom Shallow CNN, and a Fine-tuned ResNet50. BCE is a great choice for our binary classification task, where the goal is to categorize images as either real or synthetic. The logarithmic component penalizes wrong predictions more heavily than correct ones, driving model improvement during training. BCE can be expressed as

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} -\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right], \quad (1)$$

where N is the number of samples in the dataset, p_i is the probability of sample i being AI generated (i.e. our model's output), and y_i is the actual label for sample i (0 if real, 1 if synthetic).

3.2. Fully Connected Linear NN

We train a fully connected (FC) single layer linear NN directly on the flattened pixel data as our baseline. The linear model can be expressed as

$$f(x_i, W, b) = \sigma(Wx_i + b), \tag{2}$$

where x_i is our flattened image of size $D = 64 \times 64 \times 3$ (3 channels: R, G, and B), W is our weight matrix of size $D \times D$, and b is a bias vector of size 1. The sigmoid function (σ) scales the output of the matrix operations into a final probability of being AI generated between [0, 1]. To produce a final decision value, we say that an image is real if the output is less than $\frac{1}{2}$ (i.e. closer to 0) or AI generated otherwise (closer to 1). The sigmoid activation is given by the following equation:

$$\sigma(x) = \frac{e^x}{1 + e^x}. \tag{3}$$

We decided to use the simplest barebone NN classifier as a baseline in order to either (a) identify any significant, easy-to-spot biases in our image data prior to tackling other approaches, or (b) provide a nice sanity check assuring us that the problem is non-trivial.

3.3. Custom shallow CNN

Taking inspiration from [29], our first approach at outperforming the baseline was to create our own custom shallow CNN. We settled on an architecture interleaving Convolutional [3], ReLU [20], Dropout [27], and Max Pool [19] layers in the following manner:

$$[((\text{CONV} - \text{RELU} - \text{DROP})^2 - \text{POOL})^2 - \text{FC}]. \tag{4}$$

At a high level, the convolutional layers use a large number of filters to detect local visual features, such as a facial edge or color clusters. Our first CONV layer has a kernel size of 7p, with the others being 3p. The zero padding was appropriately set to $P = \frac{F-1}{2}$, where F is the filter size, to ensure the input/output volumes have the same spacial size. Each CONV layer used a stride of $S = 1$, leaving the downsizing operations to the POOL layers.

The ReLU layers apply the activation function given in the following equation on each element:

$$f(x) = \max(0, x). \tag{5}$$

ReLU layers are extremely popular in CNN architectures because of their simplicity and effectiveness at introducing nonlinearities. We experimented with other intermediate activation functions such as Leaky ReLU [17] and ELU [7] during initial trainings and found that the performance increase over ReLU was negligible.

The Dropout layers serve as our model's form of regularization by only keeping neurons active with probability $p = 0.8$. The dropout probability was tuned to this value because of our model's tendency to overfit without the presence of dropout layers. Consequently, we aimed to generalize our model by reducing its complexity, thereby increasing its performance on unseen data.

Layer	Output Shape	Param #
Input	$3 \times 64 \times 64$	0
Conv2d	$32 \times 64 \times 64$	$4,736$
ReLU	$32 \times 64 \times 64$	0
Dropout2d	$32 \times 64 \times 64$	0
Conv2d	$64 \times 64 \times 64$	$18,496$
ReLU	$64 \times 64 \times 64$	0
Dropout2d	$64 \times 64 \times 64$	0
MaxPool2d	$64 \times 16 \times 16$	0
Conv2d	$64 \times 16 \times 16$	$36,928$
ReLU	$64 \times 16 \times 16$	0
Dropout2d	$64 \times 16 \times 16$	0
Conv2d	$128 \times 16 \times 16$	$73,856$
ReLU	$128 \times 16 \times 16$	0
Dropout2d	$128 \times 16 \times 16$	0
MaxPool2d	$128 \times 4 \times 4$	0
Flatten	2048	0
Linear	1	2049

Table 1: Shallow CNN parameter table ($136,065$ total parameters).

The max pooling layers were similarly added in order to reduce the complexity of our CNN. They come with the added benefit of reducing the model's spatial size, which reduces the amount of computation required to train our shallow CNN. We settled on max pool layers that use a kernel size of 4p with a stride of $S = 4$ for the same reasons as dropout. Namely, our hope is that the complexity reduction that comes with aggressive downsampling will make it harder for our CNN to overfit to the training set.

Finally, a FC layer followed by a sigmoid activation is added to round out the binary classifier. A table of our model parameters is shown in Table 1.

3.4. Fine-tuned ResNet50

Inspired by the promising results of the Xception model [6] in our related work discussion [13] [29], we experimented with fine tuning a state of the art architecture in ResNet50, developed by Kaiming et al [8]. As an overview, ResNet50 is a deep CNN with 34 layers of convolutions, skip connections, and batch normalizations [9]. The skip connections (also known as "shortcut connections") allow intermediate inputs to bypass subsequent layers and stack onto the output of other layers. Figure 2 provides an illustration of the skip layer building block.

One of the key reasons why ResNet exhibited stellar performance was its heavy use of batch normalization, which

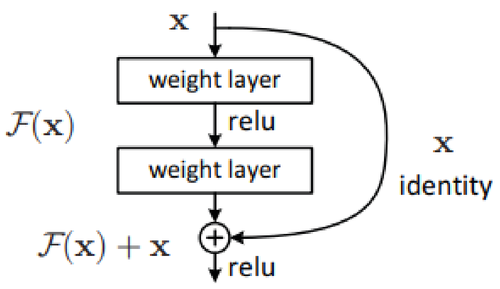

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2: ResNet50 skip connection example.

scales and shifts the outputs of each convolutional layer by the mean and variance in mini-batches. Specifically, given a mini-batch of size m denoted as $B = \{x_1, \ldots, x_m\}$, the values are scaled according to

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_B^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \\ \hat{x_i} &= \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\ y_i &= \gamma\hat{x_i} + \beta, \end{aligned} \tag{6}$$

where ϵ is a small constant (added for numerical stability), γ and β are learned parameters, and y_i is the new batch normalized value of sample i. Batch normalization has the added benefit of reducing gradient dependence, thereby mitigating the issues of exploding gradients during backpropagation.

We freeze the layers of ResNet50 and fine tune an additional fully connected layer with output dimension 512. We hypothesize that adding a single additional layer will be sufficient to transfer learn and obtain a facial image discriminator on par with our shallow CNN implementation.

3.5. Monitoring Progress

Given that our models were computationally intensive and our compute resources were limited, we chose to implement support for Tensorboard, which is a visualization toolkit provided by TensorFlow [1], to track and visualize key metrics such as loss and accuracy during training. By doing so, we were able to understand the flow of data through our model and debug various aspects of the training process, such as overfitting, vanishing gradients, or extreme learning rates. The real-time training monitoring provided by Tensorboard was critical to our hyperparameter tuning process.

4. Data

A perfectly balanced dataset of real and fake images with binary labels was used to perform binary classification with accuracy as our metric of success. The images are RGB and 64p. No additional image metadata was used to inform classification.

We used the Kaggle 140k Real and Fake Faces dataset [31] which consisted of 70k real faces from Nvidia's Flickr dataset and 70k fake faces generated by Nvidia's StyleGAN [11]. The latter, fake data, was generated and compiled by Bojan Tunguz as a Kaggle dataset of 1 million such samples at 1024p [30] before it was down-sampled to 256p and subsampled from to get the 70k images in the Kaggle dataset we use for this project. We note that this means our model will essentially act as a discriminator for the StyleGAN generator. This approach of training on both data generated by a model and the data that the model was trained on has been done [29] and reduces the potential for bias introduced in the image-gathering process, allowing our classifier to focus instead on the peculiarities of the fake image generation process.

We also believe this makes the problem tractable for us for the short duration of our project and that some of the findings we discover while trying several different approaches may generalize to facial imagery generated by other models. To further our model application, we tested on CelebA, a "wild" dataset which is a large-scale face attributes dataset with more than 200K celebrity images covering large pose variations and background clutter. [15]

4.1. Data Augmentation

We chose to train our models on both images with and without augmentation to see whether augmenting the images improves the model's generalization capabilities. We hypothesize that models trained on augmented images will outperform those unexposed to augmentation because the wider variety of examples will allow them to learn more robust features. The following augmentations were used:

Normalization: To normalize our data, we computed the average mean and standard deviation across all pixels for R, G, and B channels in our training data. The torchivsion.transforms.normalize function was used, which scales, shifts, and transforms data in the manner described in Equation 6 subequation 3, with $\epsilon = 0$. Every model was trained on normalized input data.

Random Horizontal Flip and Rotation: For the augmented data models, we introduce variance by utilizing torchvision.transforms.RandomHorizontalFlip and torchvision.transforms.RandomRotation to randomly flip and rotate certain images. This improves the model's ability to learn key features independently of image orientation.

Color Jitter: We further introduce variation in lighting and color distributions for the augmented models, which is key for facial recognition tasks. The following function was used: torchvision.transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1). The corresponding property of the image is randomly altered by +/- the input percent for each parameter in the function. As an example, a brightness of 0.2 means that the function will randomly alter the brightness of the image by +/- 20%. As a result of this augmentation, the model is less likely to overfit to any specific lighting conditions present in the training dataset.

2D Discrete Fourier Transformation While original images are represented by pixel intensity values which correspond to the spatial arrangement of visual features, through this augmentation, the Fast Fourier Transform (FFT) is used to convert an image from the spatial domain to the frequency domain. This allows the model to analyze the image based on its frequency components, which can provide complementary information to spatial features. Doing so increases the model's robustness to spatial variations such as flips and rotations.

4.2. Training Datasets

We first trained each of our three models (FC NN, CNN, and fine tuned ResNet50) on the normalized, unaugmented dataset. We then trained each model once again on the same dataset augmented with random flips/rotations and color jittering. Lastly, we trained the FC model and custom CNN on training examples in the Fourier domain, resulting in a total of eight unique models with the following labels that we will refer to throughout the rest of the paper: (FC, CNN, ResNet, FC augmented, CNN augmented, ResNet augmented, FC FFT, and CNN FFT).

5. Experimental Results

This section will outline our hyperparameter selection process and discuss experimental results in a quantitative and qualitative fashion. We will compare the performance of our eight models using the following metrics: visualized weights (FC model), visualized filters (custom CNN), and testing accuracy on three different unseen datasets.

5.1. Hyperparameter selection

To optimize our learning and tune our hyperparameters, we used the Adaptive Moment Estimation (Adam) optimizer which combines the advantages of the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam is an algorithmic optimizer that utilizes gradient descent with momentum, accelerating convergence by factoring in the exponentially weighted average of the gradients in the last n steps. This enables it to speed up convergence and skip over local minima. The learning rate for our training was set to $1e$-4 because the PyTorch default value of $1e$-3 proved too large during initial development training. The optimizer can be described quantitatively as follows:

$$\begin{aligned} w_{t+1} &= w_t - \alpha m_t \\ m_t &= \beta m_{t-1} + (1-\beta)\frac{\delta L}{\delta w_t} \end{aligned} \tag{7}$$

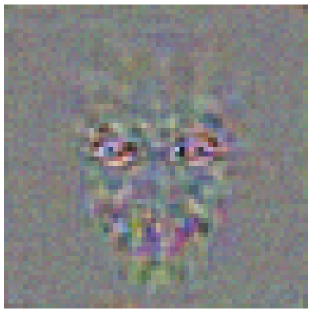
where m_t is the aggregate of gradients at time t, m_{t-1} is the aggregate of gradients at time $t-1$, W_t describes the weights at time t, W_{t+1} describes weights at time $t+1$, α_t is learning rate at time t, δL the derivative of our loss function, δW_t the derivative of our weights at time t, and β the moving average parameter with constant value 0.9.

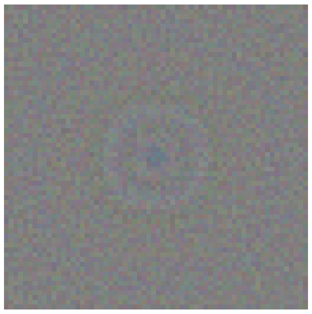
Additionally, we used a Step Learning Rate scheduler to adjust our learning rate during the training process. This helps the model explore the loss surface more effectively and avoid getting stuck in local minima to converge smoothly to a global minimum. With a gamma value of 0.2 and a step size of 3, our scheduler decreased the learning rate by a factor of 5 every 3 epochs. We used a batch size of 64 for the dataloader and set the total number of training epochs to 12 to allow for sufficient training. As another safeguard against overfitting, we set the early stopping criteria to 3, meaning our training terminates early if the validation accuracy decreases for 3 consecutive epochs.

5.2. FC Visualized Weights

We first evaluate the performance of the baseline linear model by visualizing its weight matrix. Visualizing these allows us to infer model performance because weights that appear smooth or resemble diverse features when visualized are a good indicator of successful training. On the other hand, weights that are visually incoherent or noisy tend to indicate an issue with the training, be it stemming from the input data or improperly chosen hyperparameters.

Looking at Figure 3(a), we see that the vanilla fully connected model emphasizes key facial features such as the eyes and mouth, which generative models struggle with replicating accurately. Investigating the synthetic images, we found that the eyes of AI-generated faces point in either direction in some cases, or have inconsistent lighting, features which aren't typically present in real facial imagery. Looking at the weight visualization for the augmented model in Figure 3(b), we see that the linear model struggles when images can be rotated up to 360 degrees, as indicated by the faint circular ring present within the noise. It's reasonable that the linear model can't handle data augmentation simply because of the limited number of parameters it possesses. Figure 3(c) demonstrates that the FFT model tends to emphasize higher frequency regions


(a) Linear Model


(b) Linear Model (Augmented Data)

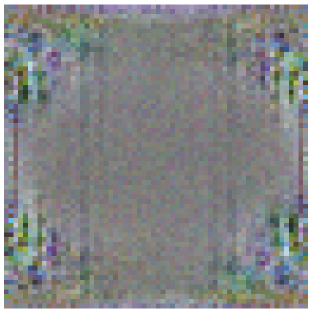

(c) Linear Model (Discrete Fourier-Transformed Data)

Figure 3: Linear models visualized weights.

(towards the corner of the image), which is consistent with the results of Figure 1.

5.3. CNN Visualized Filters

In the same vein as the previous subsection, we get an initial pulse on the shallow CNN's performance by visualizing its filters.

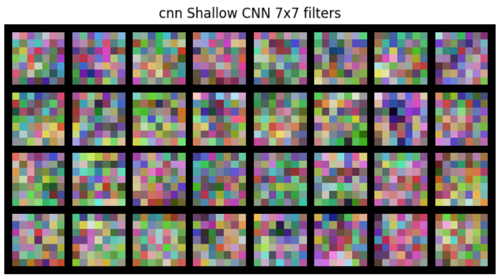
5.4. Testing accuracy

To evaluate the overal performance of each model, we compared the testing accuracy on a collection of three test sets:

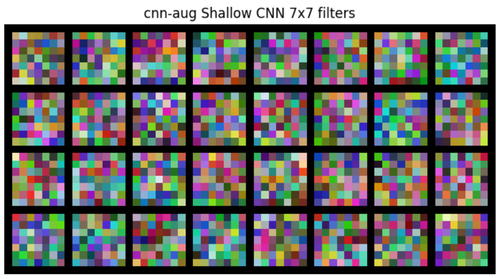
1. 1000 unseen images from CelebA [15]
2. 1000 unseen images from This Person Does not Exist [21]
3. 1000 unseen images from our ”local” test (140k Real and Fake Faces). [31]

The results are tabulated in Figure 5

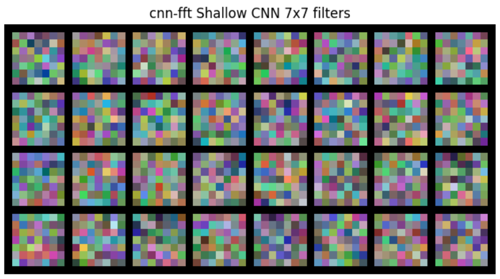
In general, we observe that the shallow convoluitional neural network performs best across most datasets. All models perform surprisingly well on real wild data, contrary to our expectations. Unfortunately, we observe that these results do not extend as well to fake wild data. It seems changing the generator does make the discriminative task more challenging as we had anticipated. We also not that, as expected, the large fine-tuned CNN was not up to the task of recognizing fake facial imagery. This failure could help to explain the indescribable patterns we see in the filters of our shallow CNN. It appears that there is more to fake facial image detection than meets the eye. We did not anticipate that the FFT-transformed data would so significantly under-



(a) CNN

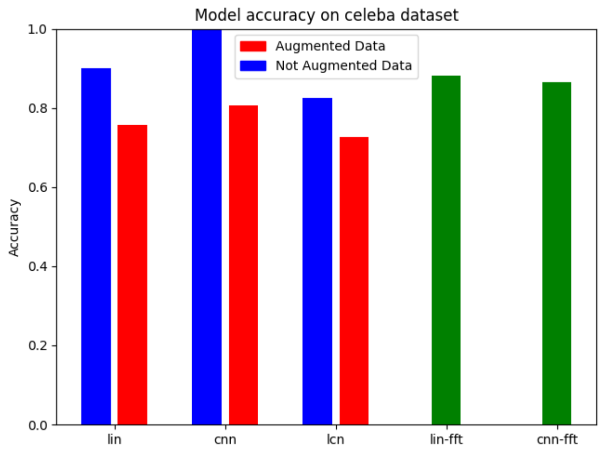


(b) CNN augmented

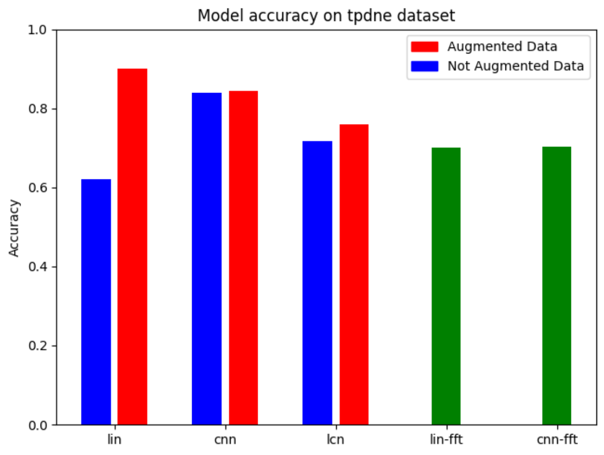


(c) CNN FFT

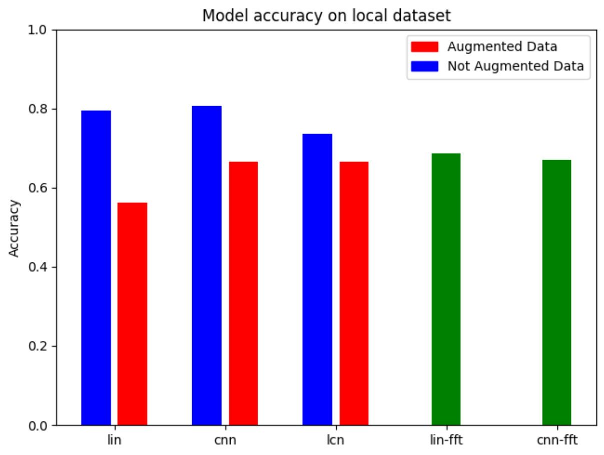
Figure 4: CNN models visualized filters.



(a) CelebA



(b) This Person Does not Exist



(c) 140k Real and Fake Faces

Figure 5: Final test accuracies for each of our proposed models on different test datasets. Labels with "lin" refer to our Linear NN (FC), "cnn" refer to our shallow CNN (CNN), and "lcn" refer to the fine tuned ResNet (ResNet).

perform. Perhaps using hand-crafted features as was done in previous work would be more beneficial.

6. Conclusion and Future Work

This project described our efforts in training a facial image AI discriminator, using three different models with a variety of inputs (normalized data, augmented data, and fourier domain data) to see how they stack up against one another. We found that the shallow CNN trained on unnormalized data outperformed the other models accross the board, save for on the "This Person Does not Exist" dataset, where the FC model trained on augmented data surprisingly performed the best. We noted that models trained on images in Fourier domain were outperformed repeatedly on two of the three test datasets, indicating that the Fourier domain data may lend itself better to certain datasets or image features such as those present in CelebA.

Random Forest of Features: An approach that we would like to try in the future is to concatenate the Fourier transform of our input image with its LBP features and train a random forest to perform classification. This approach is based on the finding from [25] that high frequency image data is helpful in differentiating between real and fake images combined with the semi-successful use in [13]. Random forests are additionally great for such a task because they can handle non-linearity between input and output data really well in addition to being able to provide feature-importance scores. From our visual analysis of fake images as well as in some interesting plots produced during training, we noticed consistently that AI generated images had eyes that were distinguishable as non-normal. Perhaps random forest would be able to better exploit this finding than our current approaches.

Deeper CNN Architecture: For the project, we opted to go with a shallower CNN architecture as deep networks have larger processing times, but in the future, we believe there is potential for deeper architectures to yield better results. While there is always the fear of overfitting with deep networks, there is also the potential for better learning as deep networks are able to capture high-level features and have larger receptive fields.

Increased Image Resolution: With higher-resolution images, models can better learn features and presumably better distinguish between real and fake. Unfortunately, this also increases training time, which is why we were not able to do so for this project, but there is potential for future work. Additionally, we also hypothesize that higher-resolution images would make any mistakes in artificially generated images more glaring and easily captureable.

Societal Implications: As researchers in industry continue to innovate and produce models that can generate more and more realistic human-level content, it is concerning that real or fake detection technology will not be able to keep up as rapidly. Models today already exist with the potential to create nearly indistinguishable deep-fakes of individuals performing actions, and it is only a matter of time before access to these models becomes commercialized and made widely available. We saw in the case of OpenAI's launch of GPT-3 just how catastrophic the implications were for K–12 education, where teachers were left unequipped to detect AI-generated work from student work. We hope that academia continues to focus on detection as well as generation for research in artificially generated human-level content.

7. Contributions and Acknowledgements

Harshal Agrawal took on responsibility for preprocessing. Ricky Parada started initial work on model development and Colin Sullivan built on Ricky's work to perform fine tuning and training. All three authors contributed equally to the write-up of this report and creation of the poster. Our code is available at the cited Github repository. [2]

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] H. Agrawal, R. Parada, and C. Sullivan. deepfakesonly. `https://github.com/SullivanC19/deepfakesonly`, 2024.

[3] J. Bouvrie. Notes on convolutional neural networks. 2006.

[4] R. N. Bracewell and R. N. Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[5] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age, 2018.

[6] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[7] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[10] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[11] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks, 2019.

[12] K. Kelleher. Revenge porn and deep fake technology: The latest iteration of online abuse. Boston University School of Law: Dome, 08 2023.

[13] A. Khodabakhsh, R. Ramachandra, K. Raja, P. Wasnik, and C. Busch. Fake face detection methods: Can they be generalized? In *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–6, 2018.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[15] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[16] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions, 2017.

[17] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[18] R. M. Michael M. Grynbaum. The times sues openai and microsoft over a.i. use of copyrighted work. NYTimes, 12 2023.

[19] N. Murray and F. Perronnin. Generalized max pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2473–2480, 2014.

[20] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[21] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proenca, and J. Fierrez. Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection. *IEEE Journal of Selected Topics in Signal Processing*, 14(5):1038–1048, Aug. 2020.

[22] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[24] A. Roy. Blue-collar jobs may weather raging ai storm better: experts. The Economic Times, 01 2024.

[25] K. S. and V. Masilamani. Image quality assessment based fake face detection. *Multimedia Tools and Applications*, 82, 01 2022.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision, 2015.

[29] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo. Gan is a friend or foe? a framework to detect various fake face images. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1296–1303, New York, NY, USA, 2019. Association for Computing Machinery.

[30] B. Tunguz. 1 million fake faces. Kaggle, 2019.

[31] xhlulu. 140k real and fake faces. Kaggle, 2020.

[32] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch, 2014.