Iris Segmentation/Detecting GAN-generated Faces

Alston Shi, Hanna Gersten, Doug Bleek
netIDs: as13388, hg2537, dmb443

Problem Statement

The rapid increase of Generative Adversarial Networks (GANs) has had an incredible impact on today's society as artificial intelligence can fabricate images indistinguishable from real photographs. While this advancement has opened many new avenues for people, it has also raised concerns regarding the potential misuse of GAN-generated content, especially in the context of deception and misinformation. In response to the growing need for tools to discern between genuine and artificially generated images, our project focuses on the development of a deep learning model specifically designed to detect GAN-generated images.

We recognize that, despite their remarkable progress, GANs often struggle with achieving perfect symmetry, particularly in the representation of facial features such as eyes. Our approach revolves around leveraging these inherent limitations of GANs. By training a sophisticated neural network to scrutinize the subtle nuances of eye symmetry, we aim to find these inconsistencies that persist in GAN-generated images. Our code is available in the following GitHub repo: https://github.com/dbleek/Residual-Attention-Network-for-GAN-Image-Detection.

Literature Survey

GAN Detection

DNN-based methods detect artifacts or other properties of GAN-generated images, but the results are often unexplainable. Explainability is important to justify the classification decision–in other words, if the possibly GAN-generated image is created using AI, but the detection is also AI-based, why should the user trust one AI over the other?

There are four types of GAN detection classifiers (Wang et. al. 2023). Deep-learning based methods extract features from input images using a Deep Neural Network (CNN, Residual, etc.) to discern between real and GAN-generated images. They are effective, but the results can be difficult to explain to users. Physical based methods examine images for inconsistencies or artifacts. For example, a real image would be expected to have consistent light reflections in both eyes, whereas they may differ in GAN-generated images. These reflections can be detected and compared to each using a IoU operation, where a real image would be expected to have a high IoU value (Hu et. al. 2021). Relatedly, Physiological based methods examine the "semantic aspect of human faces," as in the qualities of the human face itself. These methods focus on pupil shape, iris color, symmetry between facial features, etc.. Both Physical and Physiological methods are limited by the visibility of specific features in an image (e.g., whether or not the eyes are occluded), but offer better interpretability of results (Wang et. al. 2023).

Lastly, Human Visual methods rely on humans to manually discriminate between real or GAN-generated images. As GANs become more realistic and less prone to artifacts, it is more difficult for humans to detect them visually. However, studies show that with training, visual detection of GAN-generated images improves significantly.

For our project, we decided to replicate the results of "Robust Attentive Deep Neural Network for Exposing GAN-generated Faces" (Guo et. al. 2022), which combines aspects of both Deep-learning and Physical based methods. In the paper, the authors train a Mask R-CNN to detect the segmentation masks of irises given an image of a face. They then train a Residual Attention Network (RAN) to learn features of the irises in real and GAN-generated images in order to be able to infer between the two types.

Residual Attention Networks

Residual Attention Networks (RANs) (Wang et. al. 2017) were proposed in 2017 and were a progression from residual networks (ResNets). As we learned in class, ResNets incorporate a residual function or the difference between the input and the output of a series of hidden layers. This is done by adding the input directly to the hidden layers' output. The paper hypothesizes that this allows layers to learn an identity mapping more easily if it would be beneficial for the net-

work performance. This improves the optimization landscape of the network allowing for deeper networks. ResNets exhibit lower training error than similar-sized "very deep" CNNs for this reason (He et. al. 2016).

A RAN improves upon ResNets by incorporating attention modules into the network. The attention module follows a U-Net architecture (Guo et. al. 2022), and are similar to small residual networks within themselves. They include two branches. The trunk branch, which consists of residual blocks, serving as an overarching skip connection from the beginning to the end of the module. The soft mask branch uses max-pooling layers to downsample the input and provide skip connections to the interpolation layers, which upsample the input back to the original resolution. As the input progresses up through the interpolation layers, the module learns how to group low- to high-level features in succession. These groupings in turn improve the model's learning performance (Wang et. al. 2017). Attention modules can be stacked in series, but it turns out that this can decrease performance. However, attention modules can be sandwiched between residual blocks, adding attention maps to the original feature maps, in turn allowing for deeper stacking (Guo et. al. 2022).

Datasets

To fine tune a Mask R-CNN to detect irises in photos, we used the UBIPR dataset, which provides 11,102 images of eyes along with segmentation masks for features such as the iris, eyebrow and sclera. To train a RAN to discern between real and fake images of faces, we followed the paper's method and used the Flicker Faces (FFHQ) dataset as well as images from This Person Does Not Exist, which were generated using StyleGAN2 (Guo et. al. 2022). We started with 5,000 GAN-generated images and 5,000 randomly sampled real images from the FFHQ dataset. From these, we were able to extract iris pairs from 4,795 GAN-generated images and 4,083 real images, which were used as inputs to the RAN.

Implementation

Iris Detection

To start the project, we implemented a dataset preparation pipeline using a custom UBIPRDataset class. This class facilitates the loading and alignment of images and masks, encoding instances into different colors and extracting unique object IDs. The color-encoded masks are converted into binary masks, and the bounding box coordinates for each mask are obtained. The dataset class wraps samples and targets into torchvision tv_tensors, ensuring compatibility for further processing.

For our iris detection/segmentation model, we utilized the Mask R-CNN architecture with a ResNet-50 backbone, pretrained on the COCO dataset, based on the Pytorch tutorial

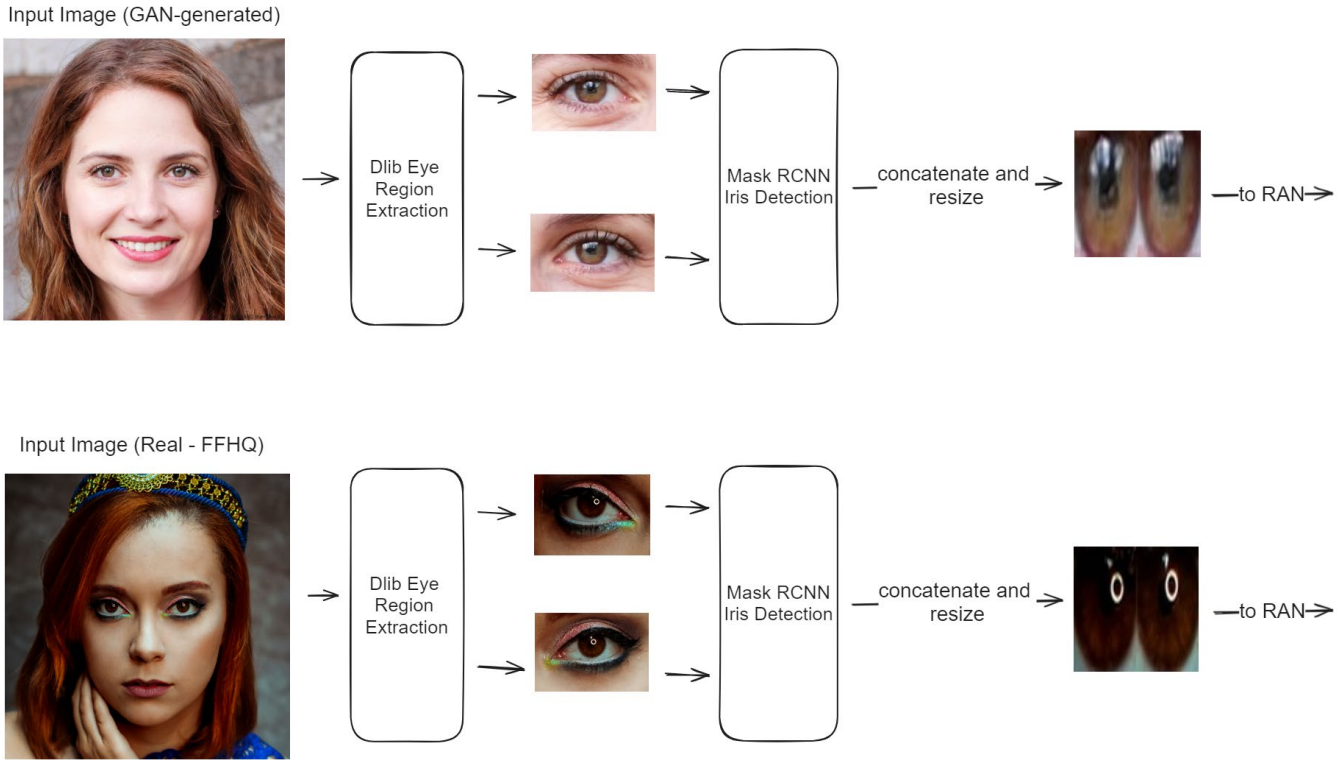

Figure 1: Example of Iris Pair Detection Pipeline for a GAN-generated and real image

for Object Detection fine-tuning (PyTorch 2023). The model was customized by replacing the pre-trained heads for box prediction and mask prediction. Training was performed over 5 epochs, employing utility functions for training and evaluation from the PyTorch vision repository. After training, the bounding box detection average precision and average recall were 0.837 and 0.862, respectively.

We implemented a function to crop the eyes based on detected facial landmarks using the Dlib library. The iris detection model was then loaded for inference on the crop ped eye images and we used the bounding boxes to crop the irises out of the image, when successfully detected. The iris pairs for each image were then concatenated side-by-side and each pair was resized to 96x96 pixels as input to the RAN. Figure 1 illustrates the iris pair detection pipeline.

RAN

In the next phase, we implemented a Residual Attention Network to discriminate between real or GAN-generated iris pairs. We adapted our code from a publicly available PyTorch implementation (Shaofeng 2018). Our implementation begins with importing key libraries, including PyTorch, torchvision, and other utilities for data handling and visualization. The Google Colab environment is set up, and the dataset containing the cropped iris pairs is unzipped and organized. Subsequently, a custom dataset class named IrisPairsDataset is defined. This class is responsible for loading image labels from a CSV file, managing image paths, and applying specified transformations.

The dataset is then instantiated with appropriate transformations. The mean and standard deviation for each RGB channel is calculated across the entire dataset for normalization, which we found improved the training of the network. The dataset is split into training and validation sets using the random_split function from PyTorch, with an 8:2 ratio. We used a batch size of 128, following the original paper.

As described in the literature survey, a RAN consists primarily of residual blocks and attention modules. Each residual block contains three convolutional layers, each followed by batch normalization and a ReLU activation function. The convolutional layers include a 1x1 kernel for dimensionality reduction, a 3x3 kernel with adjustable stride (for downsampling, if necessary) and padding, and a final 1x1 kernel for channel projection back to the original output channels. Additionally, an extra 1x1 convolution is integrated to handle scenarios where the input and output channel dimensions differ, allowing for the expansion of residual input dimensions. In the forward method, the input undergoes these convolutional operations, and the final output is obtained by summing the processed result with the residual input.

The RAN also needs the attention module designed to capture hierarchical features and enhance model performance. This module operates on input tensors of various sizes depending on the stage and consists of trunk and softmask branches, following Wang et. al. (2017). The trunk branch allows input features to bypass the module through a stack of two residual blocks, essentially as a skip connection. The softmask branch incorporates a multi-scale approach, employing max-pooling and interpolation layers to downsample and subsequently upsample the input tensor. The skip connections facilitate the flow of information across scales, following a U-Net structure. The output of the softmask branch is passed through convolutional layers, activated with a sigmoid, and then combined with the trunk branch using a Hadamard Product. As per Wang et. al. (2017), each attention module starts and ends with a single residual block. We used three stages of attention modules operating on 48x48, 24x24, and 12x12 input tensors respectively.

Following Guo et. al. (2022), our RAN begins with an initial convolutional layer followed by a max-pooling layer to reduce the feature map size. Subsequently, a sequence of alternating residual blocks and attention modules are stacked in series. After the final residual block, a global average pooling layer is employed to aggregate spatial information, and a fully connected layer with sigmoid activation is applied to produce the final classification output. The sigmoid activation ranges from [0, 1], with 1 indicating a GAN-generated image. The network's hierarchical structure, featuring residual blocks and attention modules, enables it to effectively learn hierarchical representations and focus on informative image regions for accurate classification. Table 1 below outlines the RAN architecture (Guo et. al. 2022).

Layer	Output Size	Network
Conv1	96×96×16	3×3, stride 1
Max pooling	48×48×16	3×3, stride 2
Residual block	48×48×64	$1 \times \begin{bmatrix} 1 \times 1, 16 \\ 3 \times 3, 16 \\ 1 \times 1, 64 \\ 1 \times 1, 64 \end{bmatrix}$
Attention Module	48×48×64	2× attention
Residual block	24×24×128	$1 \times \begin{bmatrix} 1 \times 1, 32 \\ 3 \times 3, 32 \\ 1 \times 1, 128 \\ 1 \times 1, 128 \end{bmatrix}$
Attention Module	24×24×128	2× attention
Residual block	12×12×256	$1 \times \begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \\ 1 \times 1, 256 \end{bmatrix}$
Attention Module	12×12×256	2× attention
Residual block	6×6×512	$3 \times \begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \\ 1 \times 1, 512 \end{bmatrix}$
Average pooling	1×1×512	6×6, stride 1
FC, Sigmoid	1	

Table 1: Model Details for RAN

Hyperparameters

To train the model, we used binary cross-entropy loss, stochastic gradient descent optimizer, and a learning rate scheduler. Following the original paper, the training loop runs for 100 epochs, using a learning rate of 0.001 and momentum of 0.9.

Results

In short, we were able to successfully match the metrics of the paper's model when using a comparable loss function. Table 2 compares our implementation's performance to that of the original paper.

In the paper, the authors used two separate loss functions: Binary Cross Entropy (BCE) and Binary Cross Entropy plus Area-under-curve (BCE +AUC). Although both perform well, the latter performs better, especially with imbalanced datasets where the number of real images outnumber the GAN-generated ones. We did not have time to implement a custom loss function and used standard BCE instead.

Method	Accuracy	Precision	Recall	F1 Score	AUC
Our Model	0.92	0.93	0.92	0.93	0.97
Authors' with BCE loss	0.92	0.92	0.92	0.92	0.98
Authors' with BCE + AUC loss	0.97	0.98	0.97	0.98	1.00

Table 2: Model Details for RAN

Figure 2 shows our model's training and validation loss. Validation loss stops improving around 20 epochs in this particular instance. During previous training runs, the model usually took about 60 epochs to reach its minimum loss, suggesting that the randomly initialized weights in this run were a particularly good starting point.

Figures 3 and 4 show our model's ROC curve and Precision-recall curve. Both graphs show how the model's metrics change with different threshold settings. We calculated the optimal threshold using the True Positive Rate (TPR) and False Positive Rate (FPR) from the ROC curve. The geometric mean is the square root of the TPR and Specificity (1 - FPR) of a classifier. By finding the geometric mean across all thresholds, the largest geometric mean corresponds with the optimal threshold setting (Brownlee 2021). In our case, the optimal threshold was 0.639.

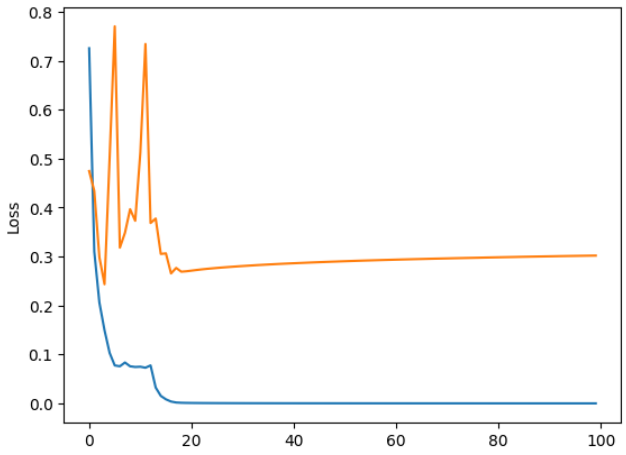

Figure 2: Loss vs. Epochs, where the orange line represents validation loss and the blue line represents training loss

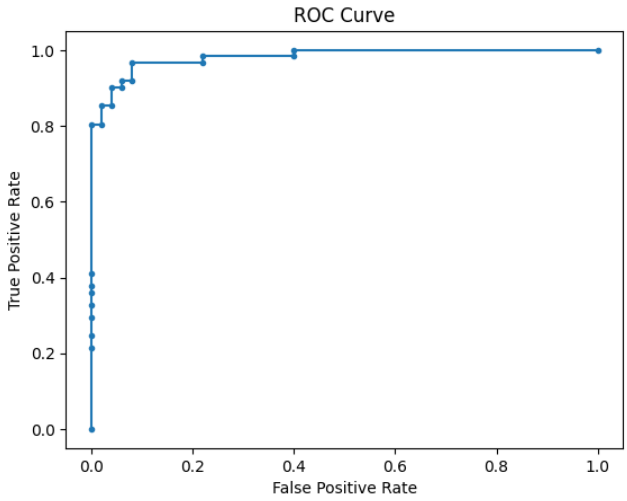

Figure 3: ROC Curve

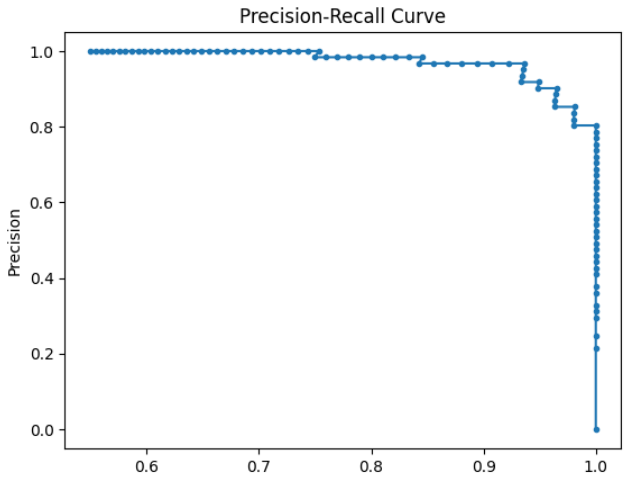

Figure 2: Precision-recall Curve

Lastly, figure 5 shows the confusion matrix of our model, which compares true positives to false positives and true negatives to false negatives.

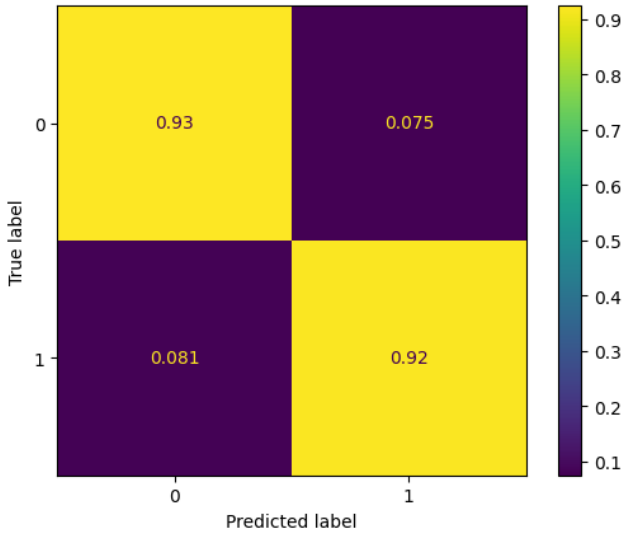

Figure 5: Confusion matrix

Conclusion

We were successful in recreating the results of the paper. Ideas for further improvements would include training the Mask RCNN iris detection model for longer to improve its performance. It sometimes detects irises incorrectly, usually mistaking a pupil or background object for an iris. Another idea would be to recreate the BCE + AUC loss function in PyTorch, as this performed better than BCE in the original paper.

References

Wang, X; Guo, H.; Hu, S.; Chang, M; and Lyu, S. 2023. GAN-generated Faces Detection: A Survey and New Perspectives. arXiv:1704.06904.

Guo, H.; Hu, S.; Wang, X.; Chang, M.; and Lyu, S. 2022. Robust Attentive Deep Neural Network for Exposing GAN-generated Faces. arXiv:2109.02167.

Hu, S.; Li, Y.; and Lyu, S. 2021. Exposing GAN-generated Faces Using Inconsistent Corneal Specular Highlights. arXiv:2009.11924.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; and Tang, X. 2017. Residual Attention Network for Image Classification. arXiv:1704.06904.

PyTorch. 2023. TorchVision Object Detection Finetuning Tutorial. https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html. Accessed: 2023-12-17.

Shaofeng, T. 2018. ResidualAttentionNetwork-pytorch. https://github.com/tengshaofeng/ResidualAttentionNetwork-pytorch. Accessed: 2023-12-17.

Brownlee, J. 2021. A Gentle Introduction to Threshold-Moving for Imbalanced Classification. 2021. https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification. Accessed: 2023-12-17.