Fake Face Detection

Tanishq Sharma, Nandhana K S, Sakshi Tiwari

[1]Indian Institute of Science Education and Research, Bhopal

Abstract

In this report, we are trying various traditional Machine Learning as well as Deep Learning methods to classify real and manually photoshopped fake faces. The main challenge here, is that the manually photoshopped fake images are harder and trickier to classify than normal AI-generated fake images. The lack of a good and big dataset also adds up to the problem.

Introduction

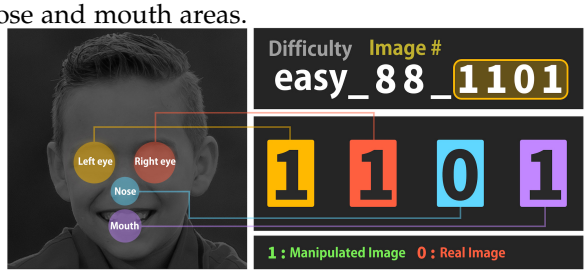
In the digital age, the ability to distinguish between real and fake human faces has become increasingly crucial. As advancements in artificial intelligence (AI) and image manipulation technologies continue to evolve, fake faces—whether generated by sophisticated AI systems or manually altered by photo editing software—are becoming more difficult to detect. The rapid proliferation of such fake images poses significant challenges in various sectors, including security, privacy, social media, and law enforcement. As a result, real and fake face detection has emerged as a vital area of research to safeguard the authenticity of visual content and mitigate the risks associated with malicious use of fabricated images.

The Dataset

The dataset we use is available on the Kaggle website namely as Real and Fake Face Detection. The dataset contains expert-generated high-quality photoshopped face images. The images are composite of different faces, separated by eyes, nose, mouth, or whole face. You may wonder why we need these expensive images other than images automatically generated by computers. Say we want to train a classifier for real and fake face images. In case of generative models like Generative Adversarial Networks (GAN), it is very easy to generate fake face images. Then, a classifier can be trained using those images, and they do great job discriminating real and generated face images. We can easily assume that the classifier learns some kind of pattern between images generated by GANs. However, those patterns can be futile in front of human experts, since exquisite counterfeits by experts are created in completely different process. Thus, we had to create our own dataset with expert level fake face photos.

Directory and File Information

Inside the parent directory, real and spoof contains real and fake face photos, respectively. In case of fake photos, we have three groups; easy, mid, and hard (these groups are separated subjectively, so we do not recommend using them as explicit categories). Also, you can use the filenames of fake images to see which part of faces are replaced (refer to the image below). All the images are photoshopped at the eyes, nose and mouth areas.



Background and Literature Review

A significant portion of the research has been dedicated to AI-generated faces, with numerous studies investigating the capabilities of AI tools in creating highly realistic human-like images. However, while research into AI-generated fake faces is extensive, there is considerably less attention given to the challenge of detecting manually photoshopped images, which are often more difficult to detect due to their tailored nature. Photoshop manipulations may be subtle, such as altering facial features, backgrounds, or expressions, and are not as easily identified by traditional AI detection systems designed for detecting synthetic faces.

Recent papers have explored various aspects of

fake face detection, with a few key works examining methods applicable to both AI-generated and photoshopped faces. For example, [Author1 et al., 20XX] presented a deep learning-based approach to identify AI-generated fake faces using a CNN architecture. Meanwhile, [Author2 et al., 20YY] investigated the challenges of detecting manually photoshopped images using statistical image forensics methods like mean, variance and fourier transform. However, we have tried to focus solely on the photoshopped fake face classification.

Methodologies

Exploratory Data Analysis

Firstly, for the dataset we performed Exploratory Data Analysis (EDA) which include the plot of class distribution of each class (real and fake), the size of images in each class and plotting some examples of images in each class.

Basic Pre-processing Methods

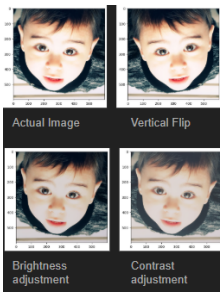
Before feeding into any model, we firstly resize the image to (72,72) from (600, 600), normalize the image between (0, 1) and add a gaussian blur to remove unneccesary noise.

Failure of Edge Detectors

We try various types of Feature extraction methods such as Edge Detectors like **Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP) and Canny Edge detectors.** These are standard feature extraction methods in an image classification problem but they do not seem to work in this case. My interpretation is that these work great in image classification problems where the two classes are very distinct in their own. For example, cat vs dog or animal vs human classification. As in this problem, we have two human images to classify for, the edge detection does not really do anything. Some of the hard level fake images are difficult even for humans to classify.

Data Augmentation

As the dataset in itself is very small (only 2000 images in total), A basic technique to increase the dataset would be **Data Augmentation**. This includes rotating, flipping or adjusting brightness and contrast in the images and adding them to the dataset which in total increases the size of the dataset. Here, we try three different augmentation techniques to increase the data samples in number, that are, Vertical Flip, Brightness Adjustment and Contrast Adjustment. As the images can be of various illuminations, the contrast and brightness adjustment works very well for the dataset and a simple vertical flip adds more variety to the dataset for the model to learn better. For this problem, the rotation of images actually, reduces the performance of the models. I tried some more data augmentation techniques but they reduced the performance of the models like for SVM, from 84% to 77%. Therefore, very specific data augmentation techniques work here.



A novel approach for feature extraction

Because the dataset is very specific in spoofing human face images, a simple trick can be used to exploit this. The making of fake face images in the dataset include two real human faces, extracting of a part of one human face like eyes, nose and mouth and putting it successfully on the other human face. All the manipulation happens only on eyes, nose and mouth which gives the opportunity to just use them as features. **What we did was, extracting the eyes, nose and mouth from the image using a pretrained face landmark extractor and with the use of a library named dlib, we get cropped images of the left eye, right eye, nose and mouth from the face images.** What dlib and pre-trained landmark does is to extract all the landmarks from the images and make them extractable via numbers like the image shown below. After extracting the facial crops, we would get something like this.

After cropping the eyes, nose and mouth from the image, we resize all the 4 images to (70,70) and then concatenate them after flattening it. This in total gets us (70*70*4)=19,600 features for each image. After flattening the image, the spatial structure of image would fall down, this is right. But we already had extracted the features from the images, now flattening would be useful to feed the image into the model. Let say, you ran a CNN, after the feature extraction, you need to flatten the image the feed into the neural network. This is the same process.

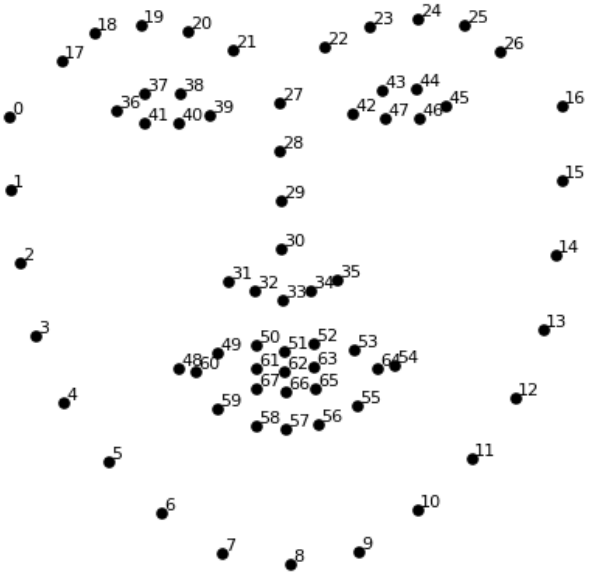


Figure 1: *dlib with facial landmark extraction*

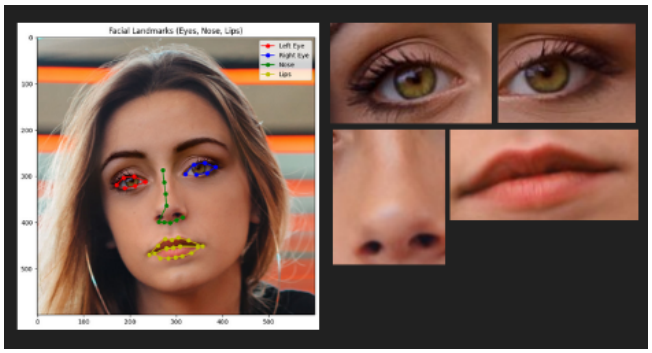


Figure 2: *Facial cropping on a sample image*

Reducing Dimensions with PCA

19,600 features are in total too much for the model to process from each image. With the help of **Principal Component Analysis**, we reduce to the number of features that include the 95% of the explained variance of the features. We can do this by using the scikit-learn's PCA method with n_components=0.95. After performing PCA, we get around 325 features which we eventually feed in the models.

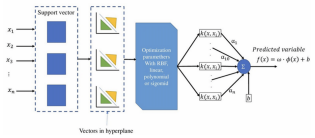
Types of Models used

We have tried various traditional Machine learning models and some deep learning models with all sorts of augmentation and feature extraction techniques.

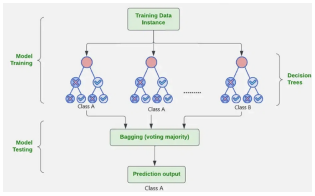
1. Support Vector Machines (SVM)
2. Random Forests (RF)
3. Convolutional Neural Network (VGG16)
4. Hybrid Model (RF + CNN)
5. Pre-trained ResNet-50

Architecture of Models

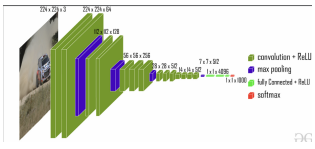
1. Support Vector Machines (SVM)
   SVM is a supervised learning model used for classification and regression. It finds the optimal hyperplane that separates classes in a high-dimensional space. Kernels like RBF or linear help handle non-linear separations. Support vectors are key data points defining the hyperplane. It's efficient for small datasets with clear class boundaries



2. Random Forests (RF)
   Random Forest is an ensemble method combining multiple decision trees for classification or regression. It uses bagging (bootstrap aggregation) and feature randomness for diversity and robustness. The final output is derived from majority voting or averaging. It is highly interpretable and reduces overfitting. Ideal for tabular and structured dataset

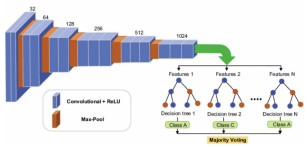


3. Convolutional Neural Network (VGG16)
   VGG16 is a CNNarchitecture with 16 layers, designed for image classification. It uses small 3x3 kernels in deep convolutional layers and has a consistent structure. Fully connected layers and a softmax layer perform the final classification. It excels in feature extraction but requires significant computational power. Commonly used for transfer learning

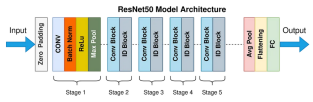


4. Hybrid Model (RF + ResNet)
   This model combines ResNet for feature extraction and Random Forests for decision-making. Features from CNN's layers are fed into a Random Forest for classification or regression. It leverages CNN's spatial feature extraction and Random Forest's robustness. The approach reduces overfitting and enhances interpretability.

Suitable for applications requiring structured decisions from visual data



5. Pre-trained ResNet-50
   ResNet50 is a 50-layer deep CNN that introduces residual connections to solve vanishing gradient issues. Its bottleneck structure with 1x1 convolutions improves efficiency. Residual blocks allow gradients to flow directly, enabling very deep architectures. It is powerful for image-related tasks like classification and segmentation. ResNet models are widely used for advanced computer vision problems



Results

The results include the performance of 12 different combinations with 5 models in total with data augmentation and facial feature extraction. Support Vector Machines with the linear and kernel functions, a Random Forest, a Pre-trained ResNet-50, a Custom VGG-16 and a Hybrid model that include a Pre-trained Resnet50 for feature extraction and Random Forest as a classifier.

I have used some short forms in the table. These are the full forms of them.

- SVM - Support Vector Machines
- Facial FE - Facial Feature Extraction
- AUG - Data Augmentation
- PCA - Principal Component Analysis
- ResNet - Residual Neural Networks
- VGG - Very Deep Convolutional Neural Networks

Table 1: *Comparison of Models and Their Results*

Model	Macro Avg Precision	Macro Avg Recall	Macro Avg F1-Score	Accuracy
Linear SVM	56%	56%	56%	56.03%
Linear SVM + AUG	95.4%	94.7%	96.0%	95.3%
Linear SVM + AUG + Facial FE + PCA	70%	70%	70%	69.8%
Kernel SVM	63%	62%	61%	63.3%
Kernel SVM + AUG	84%	84%	84%	84%
Kernel SVM + AUG + Facial FE + PCA	88%	88%	88%	88.1%
Random Forest	60%	59%	58%	59.65%
Random Forest + AUG	88%	88%	88%	88%
Random Forest + AUG + Facial FE + PCA	89%	89%	89%	**88.67%**
Pre-trained ResNet-50	72%	72%	72%	72.3%
Custom VGG-16	95.4%	94.7%	96.0%	95.3%
Pre-trained ResNet + Random Forest	59%	59%	59%	61.3%

Future Work

Although the current model for real vs fake face detection shows promise, several improvements can be made:

- **Hyperparameter Tuning:** Future work could explore a more extensive hyperparameter search, including adjustments to learning rates, batch sizes, and optimizers (e.g., Adam, SGD) to improve model performance.
- **Model Architecture:** We plan to experiment with more advanced CNN architectures such as ResNet, InceptionV3, or EfficientNet, and consider fine-tuning pre-trained models using transfer learning.
- **Data Augmentation:** Techniques such as random rotations, zooming, cropping, and color jittering can be applied to increase dataset diversity. Additionally, generating synthetic data using Generative Adversarial Networks (GANs) could augment the training process.
- **Batch Normalization and Dropout:** Incorporating Batch Normalization layers could accelerate training, while adding Dropout layers would help prevent overfitting and improve generalization.
- **Dataset Expansion:** Increasing the size and diversity of the dataset, especially by collecting more fake face samples or using publicly available data sets like CelebA or LFW, will enhance the robustness of the model.
- **Evaluation Metrics:** Implementing advanced evaluation metrics such as ROC-AUC, F1-score, and confusion matrices will provide a deeper analysis of model performance.

These improvements will help make the system more accurate, generalizable, and applicable in real-world settings such as deepfake detection and facial recognition security.