FakeCheck - Detecting and Classifying Whether an Image of Human Face Is Real or Fake

Praveen Kumar Sridhar, Kapil Sahu, Meghaana Tummapudi

February 18, 2023

1 Problem Statement

The widespread use of images on social media makes it easy to create and share fake images quickly, which poses a threat to the credibility of news and social communication. Social media id-theft is also on the rise, with scammers using fake profiles to deceive people. Image forgery/manipulation techniques contribute to unrealistic beauty ideals, leading to negative effects on mental health. Hence, monitoring and detecting fake facial images is important to maintain the truthfulness of the content.

2 Dataset

The dataset consists of all 70k REAL faces from the Flickr dataset collected by Nvidia, as well as 70k FAKE faces generated by StyleGAN. It is a balanced dataset. The images are of dimensions 256x256x3 representing the coloured images. The dataset is split into train, validation and test set. An example of real and fake images is shown below.

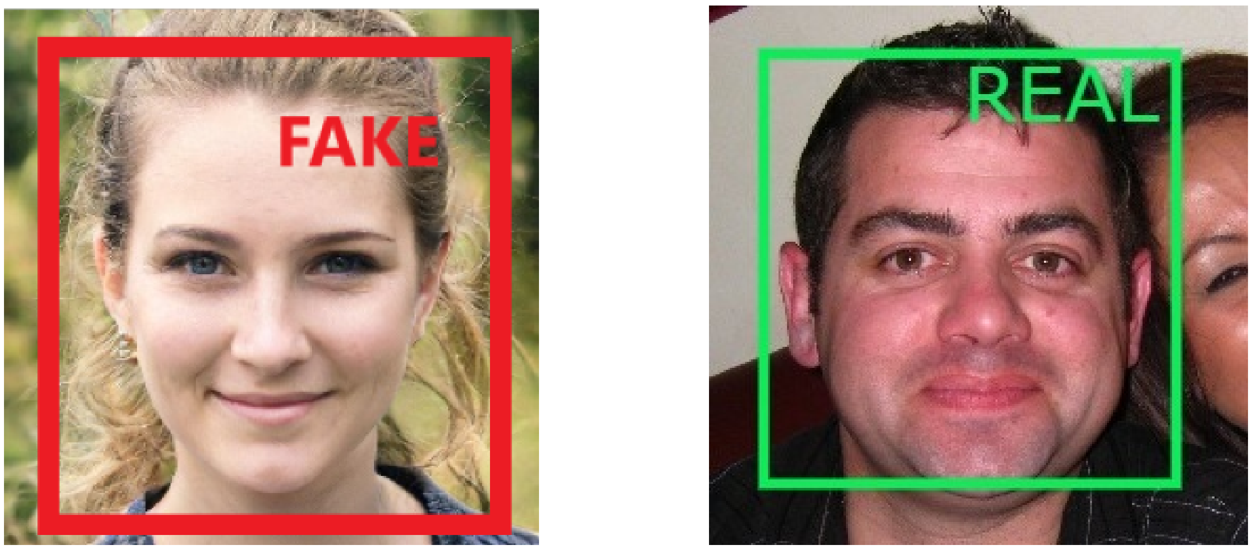


Figure 1: An example of Fake and Real images, indistinguishable to human eyes

3 Low-Risk Methodology & Results

As a part of low-risk goals, we have performed EDA, implemented Logistic Regression as a baseline model and Vanilla CNN to classify the REAL and FAKE images.

3.1 Exploratory Data Analysis

As a part of EDA, we have concentrated on examining the differences in the R, G, and B channels across the two classes (REAL vs. FAKE). This helped us understand the contribution of each color channel in these classes. The contribution of each channel is being calculated by merging the corresponding channel layers of all the images together and calculating the average intensity of each pixel after merging fig. 4. We concluded that the contribution of average intensity of R channel in pixel count is higher for Real images as compared to the Fake images fig. 2. This follows the fact that since Fake images are generated by a machine (it has a kind of a recipe which considers hard values of proportions of R, G, B channels to create an image). However, REAL images do not follow such a pattern.

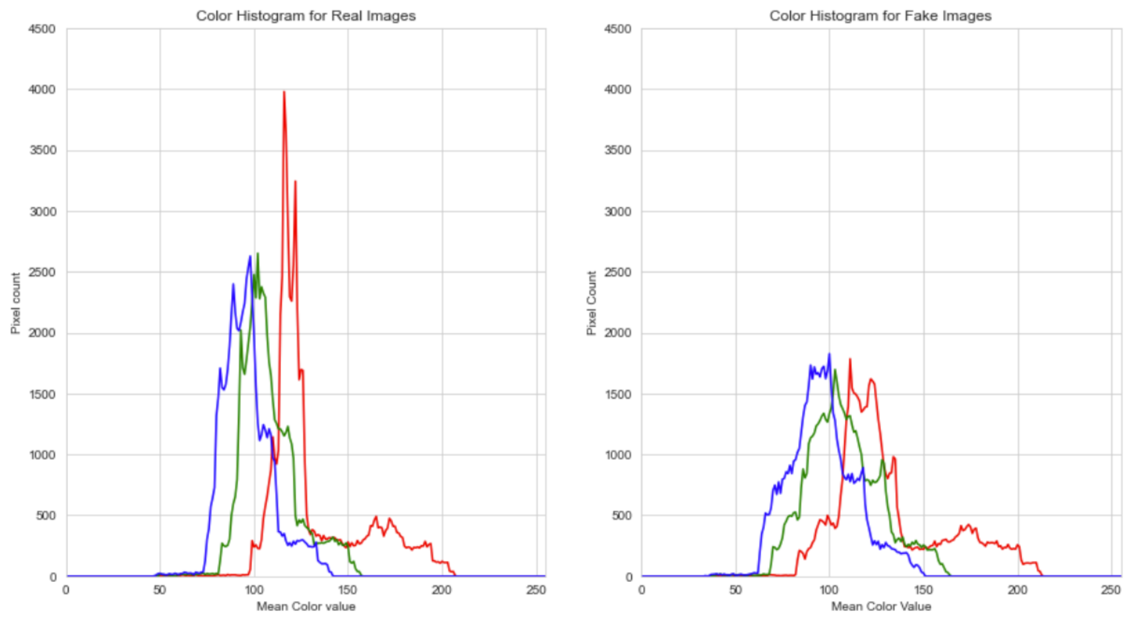


Figure 2: R, G, B Channels' Mean Intensity Value for Real and Fake Images

3.2 Logistic Regression

We implemented Logistic Regression as a baseline model. We chose this because ours is a binary image classification problem. Also because Logistic Regression is a simple model and due to its speed and ease of implementation. We used sklearn's implementation of logistic regression.

3.3 Vanilla CNN

Convolutional Neural Networks (CNNs) are a popular choice for image classification tasks as they are spatially invariant and able to capture the spatial structure of an image without requiring a large number of parameters or complex hyperparameters tuning. Our implementation of Vanilla CNN consists of 4 stacks of Convolution layers, 4 stacks of MaxPooling, a flattening layer, followed by a dense layer, finally passing to a sigmoid activation. The convolutional layers learn to identify local features in the image, while the dense layers use these features to make a final classification decision. The architecture used is visualized in fig. 5.

4 Medium-Risk Methodology & Results

As a part of medium risk goals we have experimented with VGG16, VGG19 and DenseNet-121 models. Finally, we have compared the performances of these models.

4.1 VGG

VGG(Visual Geometry Group) is a convolutional neural network known for its deep architecture (allows it to learn complex features from images) and one that is mostly used to solve the binary image classification problem for its Transfer Learning approach. The VGGs are pre-trained on the ImageNet dataset which consists of over a million images and has achieved great results on image classification tasks. Here, the VGG model is fine-tuned on our dataset. This allows the model to learn new features specific to our dataset, while still leveraging the knowledge learned from the pre-trained model. We experimented with both the VGG models (VGG16 & VGG19). The architectures of VGG16 and VGG19 are in fig. 6. VGG19 and VGG16 differ mainly in the number of convolutional layers, with VGG19 having three more convolutional layers than VGG16. This makes VGG19 a deeper architecture and allows it to learn more complex features in the input image. We used Keras' implementation of VGG.

4.2 Densenet

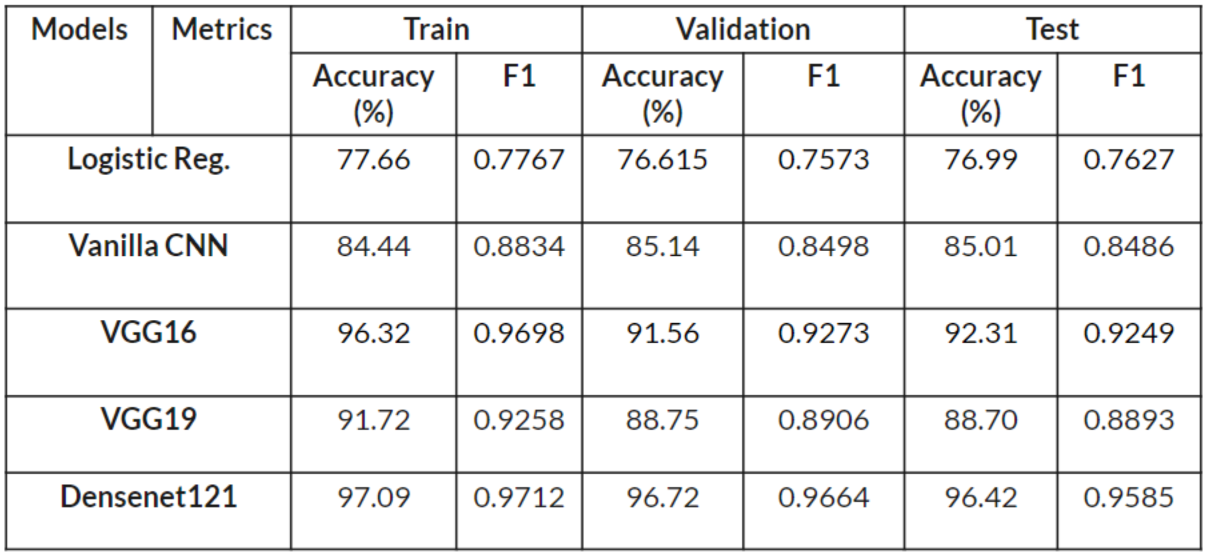
The next model we experimented with in this project is DenseNet, more specifically DenseNet-121. The network comprises L layers, each of which implements a non-linear transformation $H_l(.)$, where l is the index of the layer $H_l(.)$ can be a composite function of operations such as Batch Normalization (BN), rectified linear units (ReLU), Pooling, or Convolution (Conv). These are the dense blocks which can be seen on the fig. 7. This model has $\frac{L(L+1)}{2}$ direct connections, (i.e) connections from every dense block to every other subsequent dense block. We opted for this architecture due to several factors, including its ability to address the vanishing-gradient issue, enhanced feature propagation, promote feature reusability, and significantly decrease the parameter count. We need these advantages if we want to build any complex models with 100 or more convolution layers. We used TensorFlow to implement this model.

5 High-Risk Methodology & Results

Our high-risk goal is to evaluate our model's performance on an alternate dataset, engineer features and build a microservice (API) which could take multiple images as an input and return the likelihood of the image being fake or not. However, given the time constraint and complexity of the problem, we will be taking up these tasks in the next phase of our project.

6 Results

The chosen evaluation metrics for comparing the performance of the models are accuracy and f1 score. These metrics were chosen because the dataset is balanced and they provide a comprehensive view of the model's ability to correctly classify both real and fake images. The Densenet121 model performed very well with a high test accuracy of 96.42% and f1 score of 0.95, without any overfitting issues, as evidenced by the similar train, validation and test accuracies. Please refer the following table:



Models	Metrics	Train		Validation		Test	
		Accuracy (%)	F1	Accuracy (%)	F1	Accuracy (%)	F1
Logistic Reg.		77.66	0.7767	76.615	0.7573	76.99	0.7627
Vanilla CNN		84.44	0.8834	85.14	0.8498	85.01	0.8486
VGG16		96.32	0.9698	91.56	0.9273	92.31	0.9249
VGG19		91.72	0.9258	88.75	0.8906	88.70	0.8893
Densenet121		97.09	0.9712	96.72	0.9664	96.42	0.9585

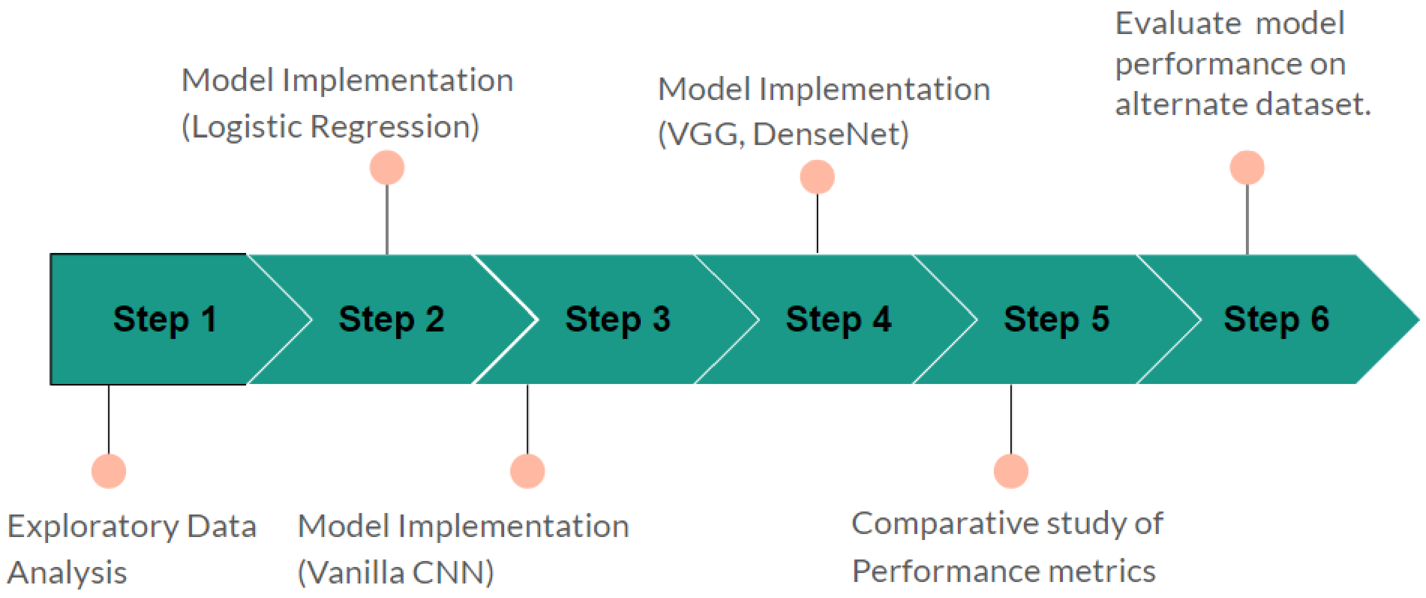
7 Future Work

For this phase we can conclude that our model performs well on the trian, validation and test sets. As a part of Phase-2, we intend to address the high-risk objectives, mentioned earlier.

8 References

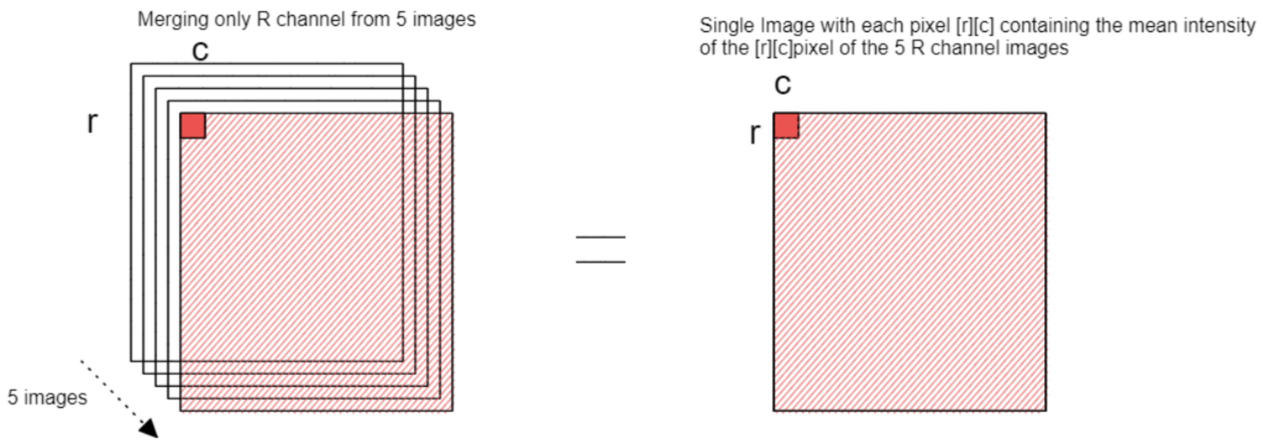
1. Dataset: 140k Real and Fake Faces
2. VGG: Very Deep Convolutional Networks For Large-Scale Image Recognition
3. DenseNet: Densely Connected Convolutional Networks
4. Project Source Code

9 Appendix



Model Implementation (Logistic Regression)
Model Implementation (VGG, DenseNet)
Evaluate model performance on alternate dataset.
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Exploratory Data Analysis
Model Implementation (Vanilla CNN)
Comparative study of Performance metrics

Figure 3: Workflow



Merging only R channel from 5 images
c
r
5 images
=
Single Image with each pixel [r][c] containing the mean intensity of the [r][c]pixel of the 5 R channel images
c
r

Figure 4: Merging R Channel Layers of Multiple Images

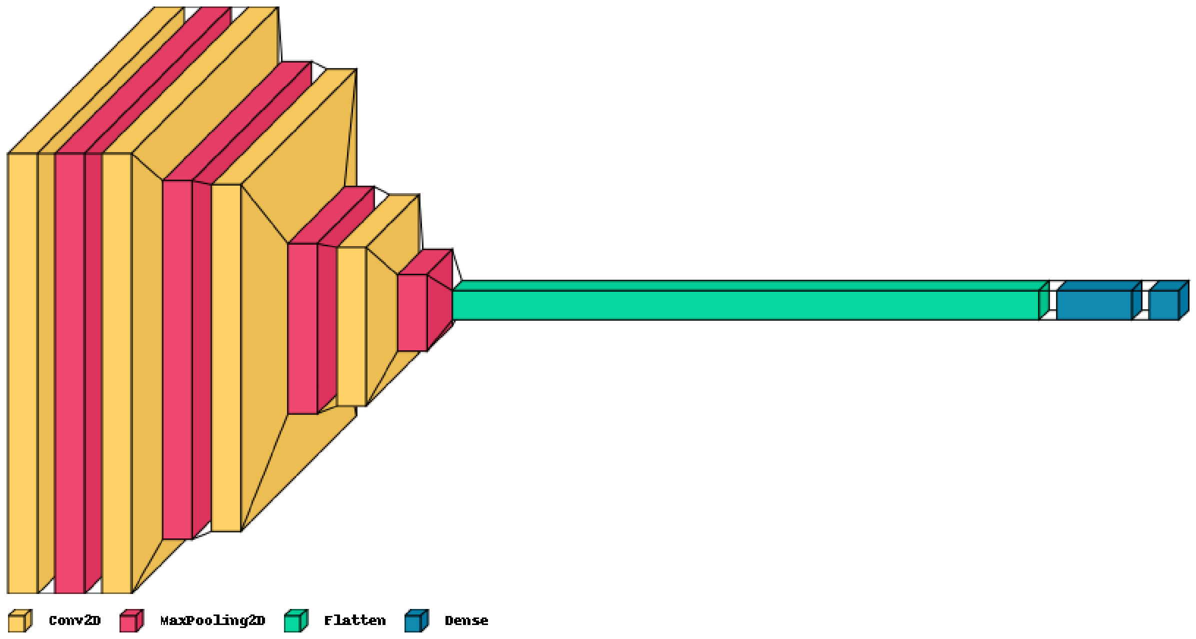


Figure 5: Vanilla CNN Architecture

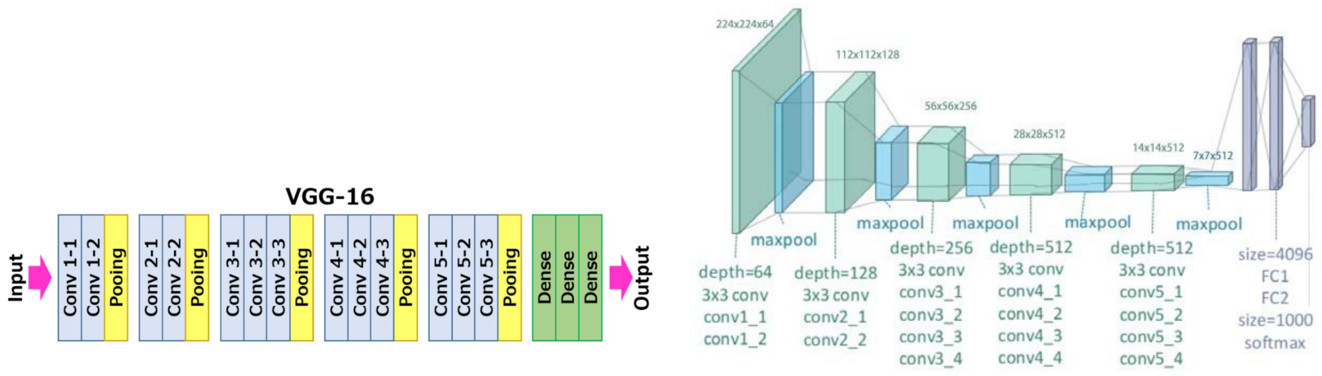


Figure 6: VGG16 and VGG19 Architecture

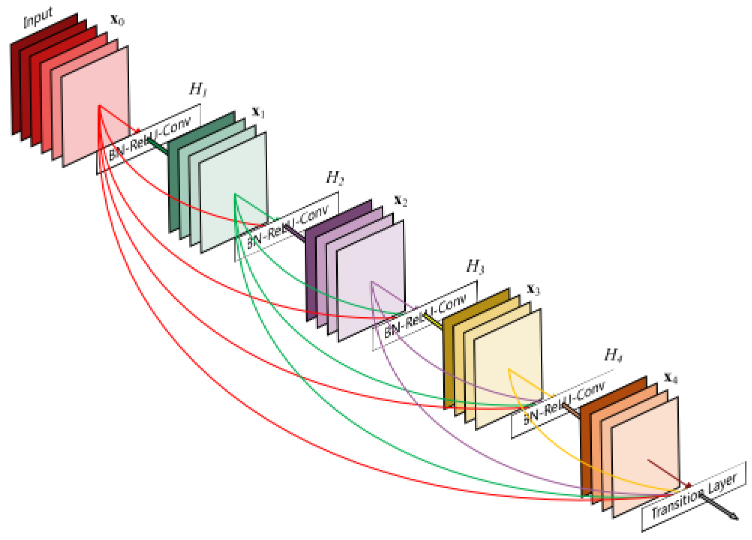


Figure 7: DenseNet Architecture