**Detecting Deep Fake Faces**

**Preprocessing**

CelebA dataset has 202,599 images and Fake Faces dataset has about a Million image of which 160,000 will be used in order to maintain a balanced dataset. The images from CelebA dataset and Fake Faces dataset are in different dimensions. Hence the images from Fake Faces dataset are resized into images of size 224\*224.

**Image Preprocessing**

**ImageDataGenerator class**

This class generates batches of tensor image data with real-time data augmentation. The data will be looped over in batches.

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

train\_gen = datagen.flow\_from\_dataframe(

train\_df,

target\_size=(224, 224),

batch\_size=64,

class\_mode='binary',

subset='training'

)

**Network Layers**

**Convo2D**

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

**Flatten**

Flattens a tensor and reshaped into 1-D array

**MaxPooling2D**

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. Max pooling operation for spatial data

**Dense**

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True).

**Dropout**

Applies Dropout layer to the input. Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

**Model Building**

The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the add() function to add layers to our model.

**Model -1**

First two layers are Convo2D layers, these will deal with the input images which are 2 dimensional matrices. Kernel size is the size of the filter size for our convolution. The activation function used for the first 2 layers is Rectified Linear Activation function. This function has been proven to work well with neural networks. Flatten layers serves as a connection between convolution and dense layers. Dense layer is the standard layer used in many types of neural networks. The Dense layer has 2 nodes because the goal of the neural network is the distinguish if an image is real or GAN generated. The activation is ‘SoftMax’. SoftMax makes the output sum up to 1 so the output can be interpreted as probabilities.

A screenshot of a cell phone

Description automatically generated

**Model -2**

First two layers are Convo2D layers, these will deal with the input images which are 2 dimensional matrices. Kernel size is the size of the filter size for our convolution. The activation function used for the first 2 layers is Rectified Linear Activation function. This function has been proven to work well with neural networks. Maxpooling operation for spatial data of the image and is followed by a dropout layer. The output is flattened and passed onto a dense layer. The last layer is a Dense layer with activation as ‘SoftMax’. SoftMax makes the output sum up to 1 so the output can be interpreted as probabilities.

