**Project Report: Deep Fake Detection**

**Problem Statement and Background:**

Deepfakes can distort our perception of the truth and we need to develop a strategy to improve their detection. Deep Fakes are increasingly detrimental to privacy, social security, and democracy. We plan to achieve better accuracy in predicting real and fake videos.

For an instance, Recently a video on social media has shown that a high ranked U.S legislator declared his own support for an enormous tax increase. At this point, people might tend to react accordingly because the video is exactly the same as the person by looks and voice. This way, DeepFake content can be used to manipulate people’s opinions. So, Deepfakes detection plays a prominent role in identifying fake content on social media and other forms of media.

Relevant work:

* Blink detection network using CNN and LSTM - <https://arxiv.org/pdf/1806.02877.pdf>
* Recurrent Convolutional Strategies for Face Manipulation Detection in Videos - <https://arxiv.org/pdf/1905.00582.pdf>
* Deep Learning Based Computer Generated Face Identification Using Convolutional Neural Network(CGFace) - <https://www.mdpi.com/2076-3417/8/12/2610/htm>
* MesoNet: a Compact Facial Video Forgery Detection Network - <https://hal-upec-upem.archives-ouvertes.fr/hal-01867298/document>

**Methods:**

**Dataset:**

We plan to use Detect fake videos using “DeepFake detection” challenge dataset of Kaggle.

Dataset: <https://www.kaggle.com/c/deepfake-detection-challenge/data>

The full dataset contains 470 GB of video files(training and testing) and a metadata file for each video. We plan to use 100 videos with ground truth, split them as 70% training, and 20% test and evaluate models using this. We plan to build a model that generalizes well.

**Columns in metadata file:**

filename - the filename of the video.

label - whether the video is real or fake.

original - in the case that a train set video is fake, the original video is listed here.

split - this is always equal to "train

**Preprocessing:**

**V**ideos to frames Conversion - Captured frames using Vedio\_Capture class of cv2 library from a video.

Individual Video length (8 seconds) → 300 Frames

* Frames to Faces - We explored dlib and Facenet to detect faces in frames and saved the face images resized to 86\*86 with both RGB and Grayscale formats. We hope that faces are important features to identify fake and real images.
* To leverage discrepancies across frames, we saved each video frame image sequentially inside a single folder across the pipeline of data preprocessing, So that we could use it for LSTM if needed. For CNN and GAN’s this doesn’t really matter.
* We resized images to different sizes like 256\*256 with entire frame, 128\*128 face only, 64\*64 face only, 86\*86 face only and trained them to pick the best configuration.
* We also explored training on RGB images and gray scale images. GAN had a performance improvement on generating high quality images in lesser epochs for Gray scale images, and it was intuitive that RGB would take time to learn complex features as the dimensions increased three times. But we went ahead and trained models for RGB as it captures high resolution as is more applicable in practice.

**Baseline model:**

We are using a single neuron with sigmoid activation function as a baseline model to classify images.

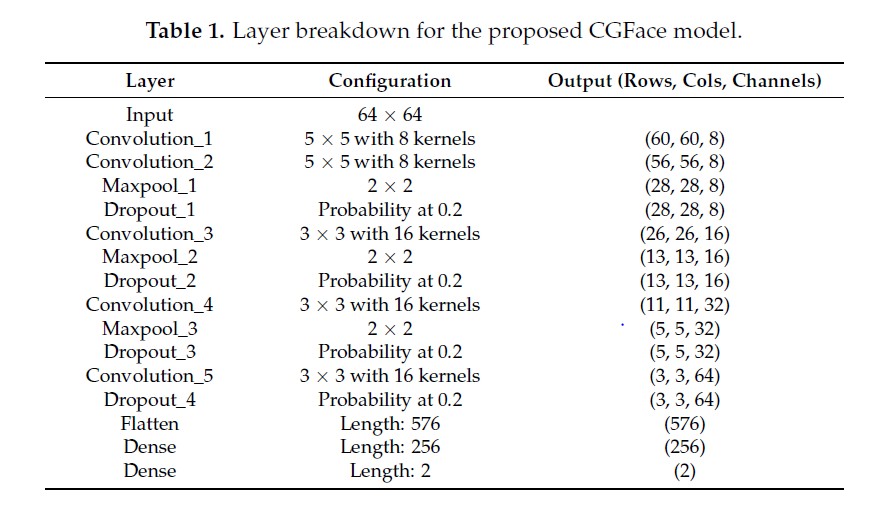
**CGFace model**

**CGFace:** It is a computer-generated face detection task by customizing the number of convolutional layers, so it performs well in detecting computer-generated face images. Adding to that an imbalanced framework (IF-CGFace) is created by altering CGFace’s layer structure to adjust to the imbalanced data issue by extracting features from CGFace layers and using them to train AdaBoost.

**Batch Normalization**: Before the fully connected layers, one batch normalization layer was added. The reason was to improve optimization by introducing some noise into the network so that it can regularize the model besides the dropout layers.

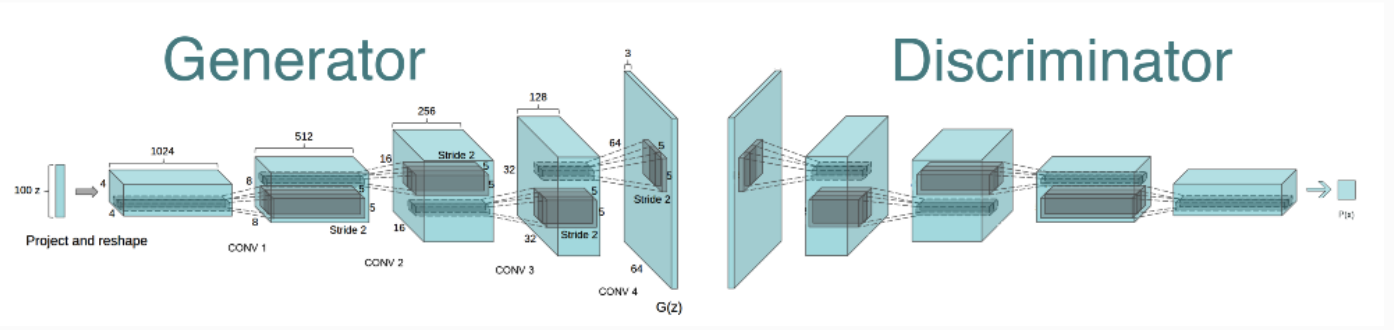
**Optimization algorithm**:Adam, **learning rate** : 0.001, **batch size**: 32, and 50 epochs

**Model building:**

****

We modified above architecture to accept input image of 84\*84 RGB. We used 32 kernels instead of 5 in the first two convolution layers to learn more dense features initially and then encode along the way. Increasing number of kernels slowed the optimization process, but increased the accuracy greatly.

**DCGAN model**

****

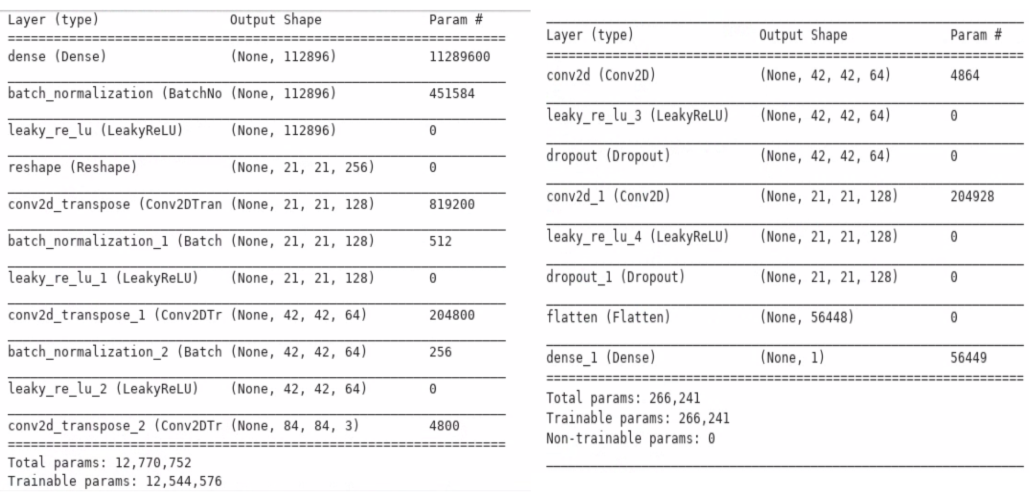
Methods for Deep Convolutional GANs

* Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
* Use batchnorm in both the generator and the discriminator.
* Remove fully connected hidden layers for deeper architectures.
* Use ReLU activation in the generator for all layers except for the output, which uses Tanh.
* Uses LeakyReLU activation in the discriminator for all layers.

We modified the architecture with different kernel sizes and number of kernels, for processing the face image that we have of size 84\*84\*3.

**Generator**: We gave a 100 dimension noise vector, this was based on other research papers that have successfully implemented GAN’s and other variants of GANS. For the number of kernels, and filter size, we went backward from last layer which is 84\*84 image, and tried many parameters and ran a few 100 epochs to find the better parameter. It was challenging to tune hyper parameters for GAN as it took more time.

**Discriminator**: We built it to accept the 84\*84\*3 image with 2 convolution layers, and a fully connected layer activated by Leaky ReLu to make predictions. **Generator**  **Discriminator**



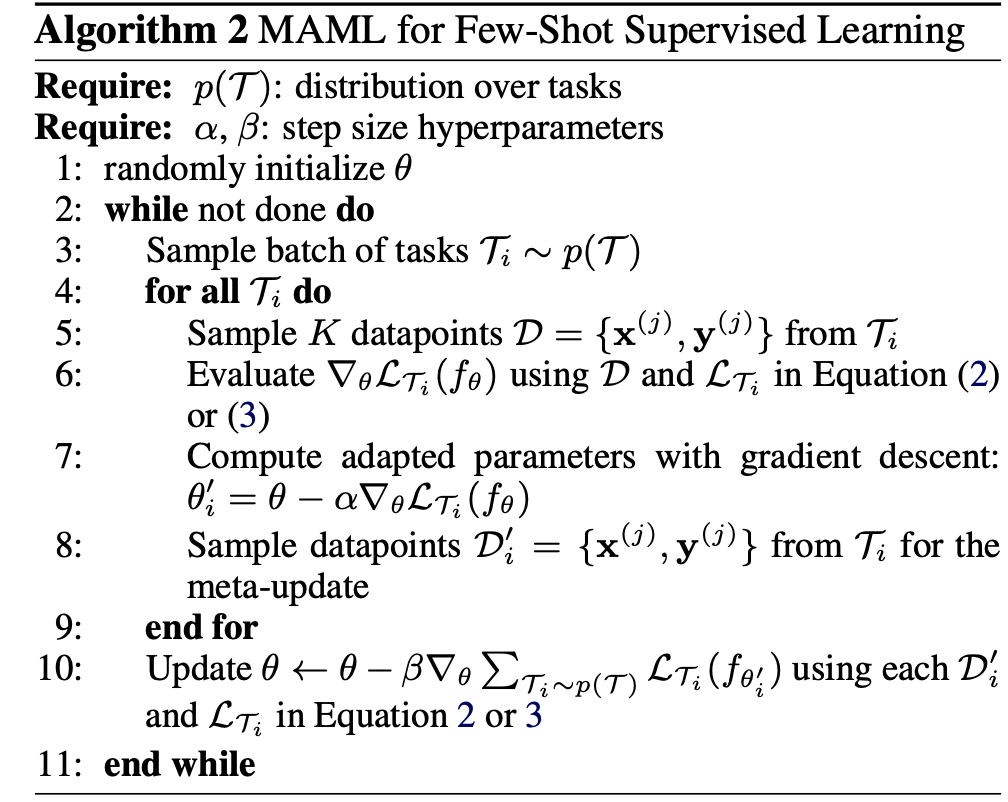
**MAML based CNN classifier:**

We referred to a MAML paper mentioned in reference, and implemented a CNN classifier by using less dataset (300-shot 2-way).

We computed loss, gradients and updated weights as specified in the algorithm using tensorflow.

Meta training tasks: As our task involved facial features, we thought of adding gender classification tasks, emotion recognition tasks, human vs horse classification tasks as meta training tasks. But we did not see any improvement in the accuracy and loss pattern.

We thought it could be due to less amount of sample tasks that we had. So, we tried using the entire imagenet task as our meta training tasks, and implemented it. But the RAM usage was very high and the computation was not enough. Because of this, google vm instance got terminated and we lost gan results. So, we decided to give up on the MAML based CNN classifier and take it as a future project for summer possibly.

****

**Results:**

We planned to achieve better accuracy for the real vs fake prediction task. This is our primary goal of the project. We initially planned to use MAML, so that it would generalize well to un-seen samples, and it can be trained online with few samples. As it did not work, we chose convolution based classifiers, and GAN has our primary solutions.

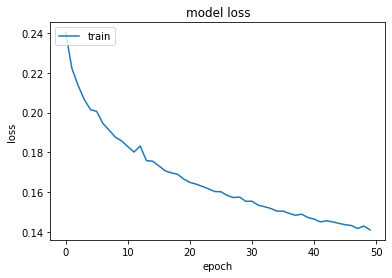
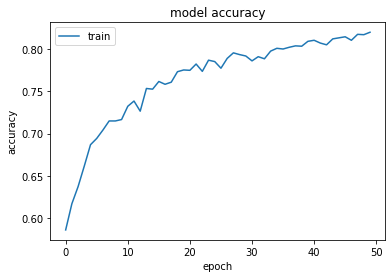
**Baseline solution vs Primary solution:** Baseline solution doesn’t encode the pixels and learn the image features. Primary solution does that by using convolution layers. So, the baseline solution has a better possibility to generalize to new unseen samples.

CGFace - Took 70 to 80 seconds per epoch on a CPU machine, i7, 8 GB RAM. We ran it for a total of 50 epochs

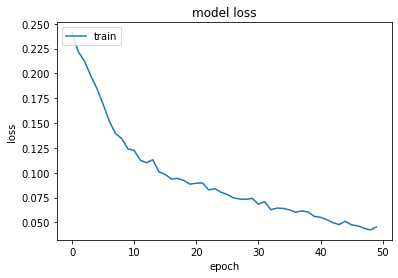
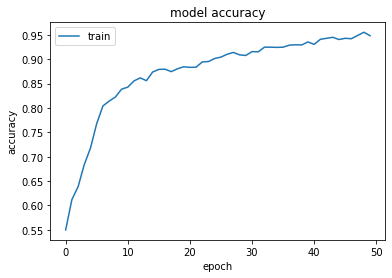
GAN - Took about 15 to 20 seconds per epoch on a Google deep learning vm instance, 13 GB 2 vCPUs, 1 x NVIDIA Tesla K80. We ran it for a total of 1500 epochs.

**Visualization:**

Base Model - Accuracy and Loss vs epoch



CGFace - Accuracy and Loss vs epoch



DCGAN - Generated images at 1000th epoch



We understand that the equilibrium between discrinimator and generator is not achieved yet and it might take several epochs for it. We lost the model weights of 2500+ epochs in the middle and we had to restart and interrupt the training process with the time we had. But we believe important facial features are being learnt and the training process is on the right track. If it learns the colour and other complex shapes and features of the shape, we hope it would be able to predict the real and fake images with reasonable accuracy. Currently, DCGAN predicts all images as real.

|  |  |  |
| --- | --- | --- |
| Model | Training Accuracy | Testing Accuracy |
| Baseline model | 82.022 | 62.9333 |
| CGFace Model | 94.822 | 68.2777 |
| GAN Model | NA | 50 |

**Tools:**

Python - Programming language

Dlib, Facenet, MTCNN - Face detection

CV2 - Image and Video processing

Tensorflow - Deep learning library

Keras - Deep learning library

Machine configuration for training: Google deep learning VM instance 13 GB RAM, 500 GB storage, 2 vCPUs, 1 x NVIDIA Tesla K80

**Lessons learned:**

CNN classifier learns better than naive classifier or other fully connected networks because of its ability to learn different image features with different settings of kernels. We also reduce the dimensions greatly in a meaningful way to speed up the optimization process.

We hoped meta learning would require less parameter tuning and simple models would perform well. But our assumption turns out to be wrong as for meta-training to go well, we might have to tune the parameters well and we might have to do it dynamically too to achieve best performance. We learnt that MAML++ approach overcomes this limitation to a certain extent.

For GAN’s to perform well with high dimensional images, and to train directly on videos using conv3d layers, we need a lot of computing resources. For videos, a single epoch could take up to days. It is always a best practice to save weight and create checkpoints during training, we learned it the hard way as we lost our gan weights on the last day!

**Team Contribution:**

We coordinated and worked on the tasks equally from day 1 when we started exploration. So, We would say each of us have equal effort. We had teams meeting on the daily basis once the class went remote and had working sessions.

**Aditi:** Main focus on MAML implementation. Worked on other two models, pre-processing and visualization as well.

**Selva:** Main focus on GAN. Worked on other two models, pre-processing and visualization as well.

**Swayanshu:** Main focus on CGFace. Worked on other two models, pre-processing and visualization as well.

**References:**

1. CGFace - <https://www.mdpi.com/2076-3417/8/12/2610/htm>
2. DCGAN - <https://arxiv.org/pdf/1511.06434.pdf>
3. MAML - <https://arxiv.org/pdf/1703.03400.pdf>
4. Blink detection network using CNN and LSTM - <https://arxiv.org/pdf/1806.02877.pdf>
5. Recurrent Convolutional Strategies for Face Manipulation Detection in Videos - <https://arxiv.org/pdf/1905.00582.pdf>
6. Deep Learning Based Computer Generated Face Identification Using Convolutional Neural Network(CGFace) - <https://www.mdpi.com/2076-3417/8/12/2610/htm>
7. MesoNet: a Compact Facial Video Forgery Detection Network - <https://hal-upec-upem.archives-ouvertes.fr/hal-01867298/document>