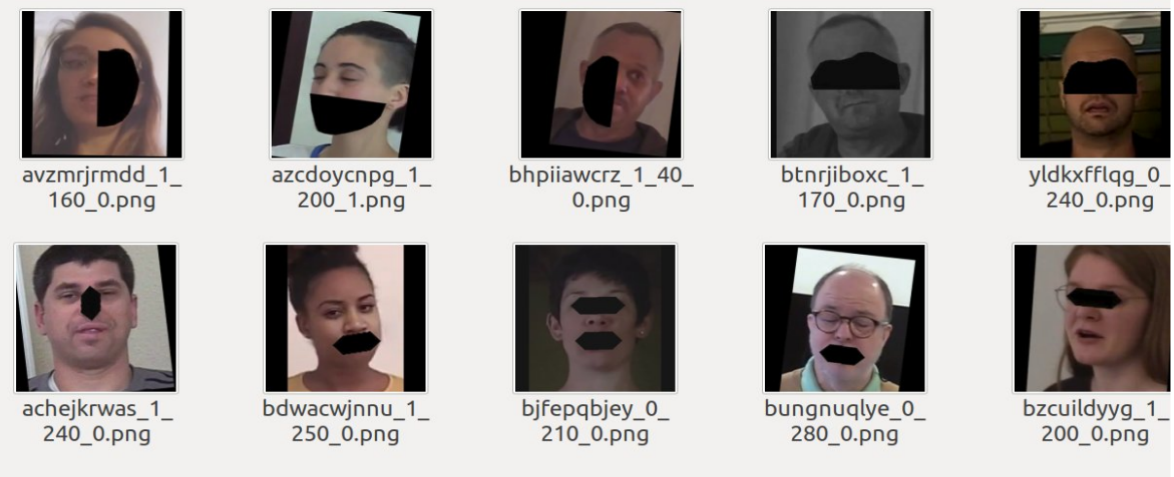
Brief summary of Data-Preprocessing meet held on 11th March, 10 CET

- We will have two separate teams working on data augmentation, one with Keras library and another with Albumentations library.
- Few augmentation parameters to be followed-
    - Size of image = (160, 160) (must be same for both teams)
    - Normalisation of image
    - Random rotation (0 to 360)
    - Horizontal and Vertical flip
    - Shear Range(0.2), zoom range(0.2) (can vary for both teams)
    - Height or width shift by 0.2 (can vary)
    - Blackout of either eyes, face or mouth (very imp)
    - Image compression
    - Gaussian Blurring (keep it's probability low, avoid using compression and blurring at same time)
    - Either Random brightness, saturation, hue contrast
      (Using too many augmentations can even degrade the dataset!)
- Generally, Gaussian blur has been used in papers, however there are other types of blur as well - Median, bilateral, glass blur, etc (options available in the Albumentations library). Try read about which works best for deepfake detection / image classification

- Blackout is a type of occlusion which has been proven very useful. However, there is no direct way to implement blackout of eyes, nose or mouth. Kaggle DFDC winner used MTCNN to get landmarks of eyes, nose and mouth and then using these coordinates, blackened certain regions.
- Find about its implementation method or try implementing on your own and share with both teams.

1. removed half face horisontally or vertically. Used dlib face convex hulls.
2. blacked out landmarks (eyes, nose or mouth). Used MTCNN landmarks for that.
3. blacked out half of the image. To be safe I checked that it will not delete highly confident difference from masks generated with SSIM.

Not only did he blackout eyes, nose or mouth, he also removed half face (vertically or horizontally)



Link - [Deepfake Detection Challenge | Kaggle]

- As of now, each team will create 30K real and 30K fake images (total 60K per team) from the available 10K real and 5K fake images.
- Each team to adopt different strategy to augment images, to have variation in data

(PDF) An Improved Dense CNN Architecture for Deepfake Image Detection (We will **try** to maintain balance by taking inspiration from their approach. Refer to section 5.B in this paper)

However, we are using full dataset of 10K images and 5K deepfakes.

- 10K real images → 5K from CelebA and 5K from FFHQ
    - Generate augmented data for each separately. 3 images to be generated per given image of CelebA and FFHQ, thus total 15K from each dataset and 30K total Real images combining the two.
    - Probably follow 80, 10, 10 ratio
    - Separate 3K images at random as “Real” test data
    - Out of 27K remaining images, select every 9th image as “Real” validation data
    - Remaining 24K as “Real” train data

- 5K deepfake images → 1K from each of 5 GANs
    - Generate augmented data for each separately. 6 images to be generated per given image of each GAN. That is, for each 1K image which we have by each GAN, we will augment it to 6K images per GAN. Hence total (6K * 5 = 30K deepfake images)
    - Same ratio as above, (80, 10, 10)
    - For each GAN → (we now have 6K images) separate 600 images as “deepfake” test data, out of 5400 remaining select each 9th image as “deepfake” validation data and remaining 4800 as “deepfake” train data.
    - Combined for 5GANs, we will have (600*5 = 3K) “deepfake” test images, (600*5 = 3K) “deepfake” validation data and (4800*5 = 24K) “deepfake” train data.

- No augmentation to be applied on the “Given” Test data which comprises of 7K images. They have already been augmented by people who conducted the challenge.

- At the end, create 3 folders (no sub-folder within each) -
  1. Augmented Train, (“real” train + “deepfake” train) total size = 48K
  2. Augmented Validation, (“real” validation + “deepfake” validation) total size = 6K
  3. Augmented Test (“real” test + “deepfake” test) total size = 6K

If this seems complex, we can discuss it in the meeting.

One more important goal - deepfake detectors fail to generalise well on unseen deep fakes generated by GANs other than the ones on which it was trained. If time permits, try to read about deepfake generation approaches different GANs follow and what can be done during augmentation to tackle this.

Keep in mind, we can not create new deep fakes by using other GANs as we have to compare benchmarks of our model, hence no “GAN-tampering” with the dataset.