Random ideas

1. Instead of limiting data as input or using smaller networks to train because we’re worried it’ll take too long, maybe lets just :
   1. Test on larger strides (leading to smaller feature maps and a neural net that will train faster).
   2. Or even better, apply pooling to more pixels (maybe use 4x4 windows instead of 2x2 windows).
   3. Or can pool after every convolutional layer instead of having 3 conv layers before pooling
   4. Or can use 1x1 convolutions which don’t change the feature map but we can reduce the dimensionality/number of channels before applying the next convolution
   5. Use a higher dropout rate - which will help deal with overfitting - but we must remember to apply the dropout **after** the activation function UNLESS we are using relus in which case we must apply dropout **before** the activation function because the result will be the same but computation will be faster. Side note: we mustn’t forget to scale up during training if we use dropout
2. Don’t just use softmax as the activation in the final layer (experiment with SVM and others)
3. Completely out the scope of our Capstone - but using other factors to determine if something is a DeepFake (appearances elsewhere online, surrounding environment etc.)
4. We should look at images of the features each layer of our neural net is extracting. This way we can adjust the width and depth of it to detect more or less features as required
5. Domain specific knowledge seems important - if we can look at most DeepFake methods and identify the way they are produced, maybe we can transform our inputs to make searching for those features much easier
6. When doing the write-up we should use ANN visualizer to clearly explain our architecture
7. On the topic of lightweight neural networks, we should reduce layers and width of our network. We will get a lower accuracy, but we should build the model so that it obtains a very low FNR (don’t want to miss DeepFakes). And then we can perform a deep neural net on the subset of DeepFakes. So essentially applying two neural nets, an initial very lightweight one just to make subset the data.

Kat’s mess #1

Look into using a pretrained convnet? “Here’s the list of image-classification models (all pretrained on the ImageNet dataset) that are available as part of keras .applications: Xception, Inception V3, ResNet50, VGG16, VGG19, MobileNet” Reuse convolutional base and do fine-tuning?

**It might be cool to visualise intermediate activations or heatmaps etc. similar to analysis in the Keras book (page 160) to prove (or not) that deepfake detection models pick up low-level subtle features**

On data preprocessing (Kat’s mess #2)

* Is there a particular reason we are resizing all frames to 32x32 in one of the .py files?

Should I skip the resizing step when reading into tensors

* Our initial approach is quite naive: we are getting frames, treating each frame individually getting a prediction for each, then, after running the network average out the scores to get the prediction per video (at least that is the simplest in terms of implementation as of now, grouping frames per video before cnn overcomplicates the data conversion).

That works for the first model, we would just have to create a dataframe where each image has a true label, prediction, and the video it corresponds to, it’s an extra step after the network.

But, this method does not take into account that certain frames are related (independent treatment) and their order in the sequence is dismissed as well. If we wanted to explore further, we could look into multi instance learning which utilises set functions; apparently, there have been some recent publications stating that there are particular network architectures that work well with it. [can get more resources on it if interested]

More random thoughts (Greg):

1. Use a Kalman filter to get rid of noise in deepfake frames and use this as inputs to network
2. Digital media forensics - they try to detect computationally cheap manipulations eg. dropped por duplicate frames. This might be a nice addition to the neural network (before or after - we can try both)
3. I think we should include a section in the report describing the different manipulation methods
4. **An average measure might work i.e. use a variety of techniques that we try and look at the majority vote**
5. We should used balanced splitting i.e. 0.5 fake and originals in training and test data
6. Sometimes the manipulation is not for the whole video - how to deal with this? One solution is RNN where we use continuous subsamples