**Meeting 1**

---

**Meeting 2**

* Linux vs Mac/Windows:

Linux looks like windows but better, project is fine with either => we will be using Mac

* Frames:

PyTorch vs Tensorflow 2.0 => we are going with Tensorflow because of ST449

Yorgos can read tensorflow but more familiar with PyTorch

There is no point learning both

* Fabian? Same speed as Mac, no GPU but can use it
* Google Cloud might be a better option instead => we will start with that (should support Tensorflow
* Google Colab, Samsung’s resources are not available

Next steps:

1. Prepare for the presentation (present literature review and a vague plan)
2. Schedule next meeting by email

**Meeting 3**

* Train classifiers is the first step

Generate and train (maybe in the end try different kinds of networks)

Default options as part of models

* Colab to start with (Mac and Tensorflow are fine)
* Yorgos => getting initial samples with increasing level of difficulty
* Samsung? Personal link
* Timeline: aim to reproduce common practice before exams (May), end of february would be great
* Key papers are supplied by Yorgos
* Second Presentation (after Lent term): code, results (?))

**Meeting 4**

* The group should use data available publically

Look for data-loading code (time-consuming process), it should be available online (eg. search FaceForensics++ code)

Another source is a paper sent by Yorgos -> different generators to check performance + tricks during training stage

* Next step: read the sent papers and send a detailed plan to agree on what we propose: what kinds of experiments, then code after finished
* Implementation: see existing code on GitHub, can use modified code
* Can do the easiest example first => get a working model to test on Google Colab (8 hours cap)
* Aim: 3 weeks: plan + the first model
* The group has agreed to provide weekly progress reports/schedule weekly meetings with advisers
* Data in batches (64 or 1024), balanced => feature representation => loss function => output
* Paper sent by Yorgos talks about input preparation -> reading data on CPU, augmentation on CPU also, then loading to GPU, pass data through network
* Task: identifying the existing method for preprocessing step => we need to find data loaders (big role)
* GANs? We don’t have time, we will use already generated images/use existing code for GANs
* Lightweight NN (CPU?) => different models (not a first step, it’s an alternative direction)
* Potentially: heavy model as a teacher to lightweight NN

Next steps:

1. Send an update or a meeting

**Meeting 5**

* Simplest experiment is on CIFAR-10 dataset.
* Videos: might be relevant as temporal domain might be relevant.
* Training on compressed images: Yorgos thinks it will work. Not sure.
* Then building a pipeline
* Yorgos: ‘There are so many networks’
* Colab Pro might be something to consider
* Using a pre-traiened network is definitely not a bad idea
* Forget about Xception Inception, stick to ResNets (they are the best and the simplest) (18, 34, 50)
* Many implementations, also in Keras
* Yorgos: ‘Networks don’t matter too much?’
* **Stick to ResNets**
* Experiment on the side: CIFAR. **Greg**

For Kat and Laurens: Crop out only relevant part of video using the mask.

FFmpeg, figure out how it works

Pre-processing will take very long

Request resources via LSE

Write a quick report on both possible datasets (could be constrained by capacity) possibly consider the GAN dataset sent by Yorgos last meeting

Search data-loaders on the internet, should be able to find them

**Meeting 6**

* Presentation to give an update
* Uncompressed videos (5 types), extract frames (every 15th), crop the faces (opencv with face cascade)
* Falsely detected face? Check the size of the bounding box with the video, best: face recognition network -- try different face detector (cascade is the worst, need a different one -- will get more details from Yorgos)
* For GAN’s they have some built in algorithms to pre-crop original faces
* Use masks to crop the videos (consistent for all frames and videos)
* Important to process the faces in the same way (very close to what they used) -- preprocessing pipelines should be the same
* Maybe we can learn how faces have been manipulated differently and identify them as fakes then
* Not every method uses masks? Read what they do
* Do we train on all methods? **Pick the best one from the paper and use this for the beginning (consider other parameters as speed..?)**
* Sequential vs functional API? Maybe evaluating known architectures is okay (resnet, mobilenet etc)
* Pretrained vs training from scratch (for the weights: random initialisation or not)-- we can do both to try
* Dropout included (resnet does not need dropout)
* Find best suggestions in literature and try those
* Cosine scheduler for learning rate (Greg will be using adam) -- SGD with momentum? Momentum works very well but have to spend time tuning
* We will design an experiment and optimise that (if it generalises well or not)
* More novel? Lightweight networks, add more architectures, train on sparsity
* Maybe have 3 directions to explore
* **‘Bag of tricks’ (2019) paper**
* TensorflowLight? Simpler, tensorflowLight is too complicated, we can use architecture with less parameters (eg mobilenet) no difference in training but they are more friendly in terms of efficiency
* RNN? LSTM and classify videos, data loader for a bunch of frames

RNNs might be better for some problems if there is a difference how deepfakes vary from frame from frame in comparison to original -- further investigation in literature

* More computing power: apply for google research (Kostas to fill a form)