**List of Papers:**

1. Deeptrace, The State of Deepfakes: Landscape, Threat and Impact (Kat)
2. FaceForensics++ (Laurens)
3. Fake Faces Identification via CNN (Kat)
4. Deep Learning for Deepfake Creation and Recognition (Kat)
5. MesoNet: a Compact Facial Video Forgery Detection Network (Kat)
6. CNN-generated images are surprisingly easy to spot... for now (Kat)
7. New convolutional layer (Laurens)
8. Computer graphics versus Natural images (Laurens)
9. Deep Learning with Python (Greg)

## Xception: Deep Learning with Depthwise Separable Convolutions (Greg)

1. Densely connected convolutional networks (Greg)
2. Useful FaceForensics notes (Greg)
3. Deep residual learning for image recognition (Greg)
4. Multitask Learning (Kat)
5. Face Wrapping Artifacts (Kat)
6. Bag of Tricks (Laurens)
7. Simple features (Laurens)
8. Optical flow (Laurens)
9. Kaggle public second place (Laurens)
10. Kaggle pubic third place (Laurens)
11. Deepfake Video Detection Using Recurrent Neural Networks (Greg)
12. The Deepfake Detection Challenge Dataset (Greg)
13. Mixup: Training Augmentation (Kat)

**Deeptrace, The State of Deepfakes: Landscape, Threat and Impact** (Kat)

Foreword:

* “Deepfakes are here to stay, and their impact is already being felt on a global scale. We hope this report stimulates further discussion on the topic, and emphasizes the importance of developing a range of countermeasures to protect individuals and organizations from the harmful applications of deepfakes.”

The State of deepfakes: an Overview:

* Huge increase in deepfake material: “Our findings revealed that the total number of deepfake videos online is rapidly increasing, with this measurement representing an almost 100% increase based on our previous measurement (7,964) taken in December 2018.” (page 1); 96% of all deepfakes online is pornography
* Deepfake pornography does not affect different groups equally: 100% female, nationality and profession breakdowns differs from non-pornographic deepfakes

Commodification of Deepfakes:

* “Currently, the most popular of these techniques is the Generative Adversarial Network (GAN) due to its flexible applications and realistic outputs. … [The increased number of mentions of GAN in the academic papers] provides an indirect indication of how fast GAN quality is improving.” (page 3)
* 20 deepfake creation community websites and forums, activity on github is rising
* “The faceswap source code from the anonymous Deepfakes creator was donated to the open-source community and uploaded on GitHub. Since then, development has been driven by many project forks, with programmers contributing to improved quality, efficiency, and usability of these new code libraries.” (page 4)
* There are computer apps, service portals and marketplace services to create deepfakes

Deepfake pornography:

* 10 individuals most targeted (figure, page 7)
* A case study, DeepNude (page 8)

Politics and Cybersecurity:

* “growing awareness of deepfakes is already dal discourse, and undermining the perceived objectivity of recorded video in politically charged contexts.” (page 10)
* “We observed two cases where realistic synthetic photos of non-existent people were used on fake social media profiles, in an attempt to deceive other users and extract information. ” (page 13)
* “There have been several reported cases where synthetic voice audio has allegedly been used to defraud companies. While no concrete evidence has been provided to support claims that the audio was synthetic, the cases illustrate how synthetic voice cloning could be used to enhance existing fraud practices against businesses and individuals.” (page 14)

## FaceForensics++ (Laurens)

Photos are manipulated using Face2Face, FaceSwap, DeepFakes, and NeuralTextures. Paper examines the realism of state-of-the-art image manipulations. Propose an automated benchmark for facial manipulation detection. The dataset is an order of magnitude larger than comparable, publicly available, forgery datasets. The use of additional domain-specific knowledge improves forgery detection to unprecedented accuracy, even in the presence of a strong compression.

The reconstruction and tracking of human faces is a well-examined field in computer vision. There are two facial manipulation method categories: facial expression manipulation and facial identity manipulation. Face2Face enables the transfer of facial expressions of one person to another person using commodity hardware. “Synthesizing Obama” is able to animate the face of a person based on an audio input sequence. Identity manipulation is known as face swapping. DeepFakes performs this via deep learning. The paper can automatically and reliably detect such manipulations, and outperforms humans by a significant margin. Use Convolutional Neural Networks (CNN).

As the digital media forensics field lacks a benchmark for forgery detection, we propose an automated benchmark that considers the four manipulation methods in a realistic scenario, i.e., with random compression and random dimensions.

The paper intersects several fields in computer vision and digital multimedia forensics.

Face manipulation methods: Comprehensive state-of-the-art report published by Zollhofer. Recently, several face image synthesis approaches using deep learning techniques have been proposed. Generative Adversial Networks are used to apply face aging. Most techniques suffer from low image resolutions. Recently, this has been improved by using progressive growing of GANs, producing high-quality synthesis of faces.

Multimedia forensics: this aims to ensure authenticity, origin and provenance of an image or video without the help of an embedded security scheme. Recent literature concentrates on CNN-bsaed solutions, both through supervised and unsupervised learning. For videos, the main body of work focuses on manipulations created with relatively low effort, such as dropped or duplicated frames, varying interpolation types, copy-move manipulations, and chroma-key compositions. Some work focsues on distinguishing computer generated faces from natural ones. Robustness issues often remain unaddressed. (such as resizing and compression).

Forensic analysis datasets: Most have been created with significant manual effort under very controlled conditions. 1.8 million images from 4000 fake videos.

3. Large-scale facial forgery database. Applied 1000 pristine videos downloaded from the internet. Early experiments with al manipulation methods showed that the target face had to be nearly front-facing to prevent the manipulation methods from failing or producing strong artifacts. Manual screening performed. They use state-of-the-art video editing methods to work fully automatically. Use two graphics based approaches (Face2Face and FaceSwap) and two learning-based approaches (DeepFakes and NeuralTextures). Ground truth masks were computed to indicate whether a pixel has been modified or not.

FaceSwap: transfer the face region from a source video to a target video.

DeepFakes: term deepfakes has become a synonym for face replacement based deep learning.

Face2Face: facial reenactment system that transfers the expressions of a source video to a target video while maintaining the identity of the target person.

NeuralTextures: facial reenactment. Uses the original video data to learn a neural texture of the target person, indulging a rendering network.

Videos are compressed using the H.264 codec.

4. Forgery detection of human observers.

204 participants consisting mostly of computer science university students. Baseline for automated forgery detection methods. 50:50 split pristine and fake. 2, 4 or 6 seconds. Collection of 12240 decisions. User accuracy on Face2Face and NeuralTextures (< 40%) is bad, on LQ pristine images too. Reason is that both do not introduce a strong semantic change.

4.2 AFDM: since the goal is to detect forgeries in facial imagery, they use additional domain-specific information that can be used to extract the input sequences. This information is using face tracking, and a conservative crop. (Thies et al.) Various variants of the approach were evaluated using different state-of-the-art classification methods. Learning-based methods used in the forensic community for generic manipulation detection, computer-generated vs natural image detection, and face tampering detection. The classification based on XceptionNet outperforms all other variants.

Detection based on Setanalysis features: employs hand-crafted features. Features are co-occurrences on 4 pixels patterns along the horizontal and vertical direction on the high-pass images for a total feature length of 162. Used to train a SVM classifier. 128x128 central crop-out was input. Method *struggles with compression.*

Detection based on Learned features: Five network architectures were evaluated. Paper (1) cast Steganalysis to a CNN-based network. Use data set to train CNN. The constrained convolutional layer is specifically designed to suppress the high-level content of the image.

XceptionNet is a traditional CNN trained on ImageNet based on separable convolutions with residual connections. Transferred to their task by replacing the final fully connected layer with two outputs. And three others.

Comparisons of the forgery detection variants: All approaches achieve very high performance on raw input data. Detectors outperform human performance by a large margin. Domain-specific information in combination with a XceptionNet classifier shows the best performance in each test. We use this network to further understand the influence of the training corpus size and its ability to distinguish between the different manipulation methods.

The experiments show that all detection approaches achieve a lower accuracy on the GAN-based NeuralTextures approach. NeuralTextures trains a unique model for every manipulation which results in a higher variation of possible artifacts. Training corpus size plays an important role in testMSE.

Benchmark:

Specifically, we collect a set of 1000 images, each image randomly taken from either the manipulation methods or the original footage. Note that we do not necessarily have an equal split of pristine and fake images nor an equal split of the used manipulation methods. The ground truth labels are hidden and are used on our host server to evaluate the classification accuracy of the submitted models. The automated benchmark allows submissions every two weeks from a single submitter to prevent overfitting.

**Fake Faces Identification via CNN** (Kat)

Introduction:

* Generative Adversarial Network (GAN) is a powerful model used to produce fake face images with high quality
* Paper proposes a CNN-based method to identify fakes with an average performance accuracy of over 99.4% (“with particular attention to the high pass filter for the input image, the number of layer groups and the activation function” (page 43))
* “Some novel models in deep learning such as CNN and GAN have been extensively studied and have achieved great success in many image related applications, such as image style transfer [8, 11], image super-resolution [13, 18], image inpainting [10, 22] and image steganalysis [6, 21].” (page 43)

Face Generation with GAN:

* GAN “learns the distribution form high-dimension data and produces novel sample” (page 43)
* GAN consists of two parts: “The generator learns to create fake data indistinguishable from the real data, while the discriminator learns to determine whether the input data is real or fake. They contest against each other during training until the generator can produce high quality fake data.” (page 43)
* Currently, there are methods to produce fake images of high resolution, 1024x1024

The Proposed Detection Method:

* Input: “RGB color image with size M × M × 3” (page 44), output: probability
* “the main difference between the two kinds of images would be reflected on the residual domain according to the previous research” (page 44) -> high pass filter transforms the input images into residuals
* Next, three layer groups (“each group includes a convolutional layer (3 × 3 size, 1 × 1 stride) equipped with LReLu and a max pooling layer (2 × 2 size, 2 × 2 stride)” (page 44))
* The output feature map number doubles after each group
* Then, 2 fully connected layers with LReLu, units are 1024 then 512
* Lastly, softmax layer outputs probability
* Implementation: Tensorflow, learning rate is 0.0001, “All the weights are initialized using a truncated Gaussian distribution with zero mean and standard deviation of 0.01. The biases is initialized as zero. L2 regularization is enabled in the fully-connected layer with the λ of 0.0005. In the training stage, we use a batch size of 64 and train the proposed CNN for 20 epochs. In addition, we shuffle the training data between epochs.” (page 44)

Experimental Results

* Data set: 30000 true (CELEBA-HQ dataset) and 30000 fake face images (GAN-generated), all 1024x1024, PNG (images are resized and compressed)
* Identification: “some background regions in some fake face images looks unnatural, which may help increase the detection performance. To reduce the influence of image backgrounds, we crop a small segment (128×128) from every image in the original image set (256 × 256), and ensure that each cropped segment mainly includes some facial key-points (such as eyes, nose, and mouth) ... Finally, we obtain two different image data set for experiments, that is original ones including face and background, and the cropped ones just including main facial region.” (page 45)
* Results: “on the test set, and obtain accuracies of over 99.4% and 96.3% on the original images and cropped images respectively” (page 45)
* Model’s features: 3 different high-pass filters were tested, layered groups: removing one decreases accuracy but adding one does not make a difference, activation function: “six commonly used activation functions are considered in the proposed model. They are TanH, ReLu, and four variants of ReLu, including PReLu, LReLu, ELU and ReLu6 … LReLu obtains the best performance” (page 45)

**Deep Learning for Deepfake Creation and Recognition** (Kat)

Abstract:

* “this study provides a comprehensive overview of deepfake techniques and facilitates the development of new and more robust methods to deal with the increasingly challenging deepfakes” (page 1)

Introduction:

* “Deepfakes therefore can be abused to cause political or religion tensions between countries, to fool public and affect results in election campaigns, or create chaos in financial markets by creating fake news.” (page 1) “software called DeepNude shows more disturbing threats as it can transform a person to a non-consensual porn” (page 2)
* “the United States Defense Advanced Research Projects Agency (DARPA) initiated a research scheme in media forensics (named Media Forensics or MediFor) ... Recently, Facebook Inc. teaming up with Microsoft Corp and the Partnership on AI coalition have launched the Deepfake Detection Challenge” (page 2)

Deepfake Creation:

* Table of notable deepfake tools (page 2)
* Facial expression manipulation: two encoder-decoder pairs with the same encoder network, relatively easy because “because faces normally have similar features such as eyes, nose, mouth positions” (page 2)

Deepfake Detection (fake image detection subsection only, fake video detection subsections are omitted):

* “Zhang et al. [46] used the bag of words method to extract a set of compact features and fed it into various classifiers such as SVM [47], random forest (RF) [48] and multi-layer perceptrons (MLP) [49]” (page 4)
* “Among deep learning- generated images, those synthesised by GAN models are probably most difficult to detect as they are realistic and high-quality based on GAN’s capability to learn distribution of the complex input data and generate new outputs with similar input distribution.” (page 4)
* “Xuan et al. [50] used an image preprocessing step, e.g. Gaussian blur and Gaussian noise, to remove low level high frequency clues of GAN images. This increases the pixel level statistical similarity between real images and fake images and requires the forensic classifier to learn more intrinsic and meaningful features, which has better generalization capability than previous image forensics methods [51,52] or image steganalysis networks [53].” (page 4)
* Alternative approach: “Agarwal and Varshney [54] cast the GAN-based deepfake detection as a hypothesis testing problem where a statistical framework was introduced using the information-theoretic study of authentication [55]. The minimum distance between distributions of legitimate images and images generated by a particular GAN is defined, namely the oracle error. The analytic results show that this distance increases when the GAN is less accurate, and in this case, it is easier to detect deepfakes.” page (4)
* Also, Hsu [56] used a two-phase deep learning method, the first phase is “a feature extractor based on the common fake feature network (CFFN)” (page 4), the second phase is “a small CNN concatenated to the last convolutional layer of CFFN” (page 5)
* CFFN: has several dense units (3-5, number of channels is up to a few hundred, each has a varying number of dense blocks), extracts “discriminative features between the fake and real images, i.e. pairwise information” (page 5)
* Hsu’s method is better than other methods: “Experimental results show the superior performance of the proposed method against its competing methods such as those introduced in [69–72].” (page 5)

Discussions, Conclusions …

* The arising problem is that most detection methods rely on the drawbacks of the deepfake generating methods, but the code for a given generating method is not always available: “current detection methods mostly focus on drawbacks of the deepfake generation pipelines, i.e. finding weakness of the competitors to attack them. This kind of information and knowledge is not always available in adversarial environments where attackers commonly attempt not to reveal such deepfake creation technologies. This is a real challenge for detection method development and a future research needs to focus on introducing more robust, scalable and generalizable methods.” (page 9)
* Further research: to integrate deepfake detection methods on various media platforms etc.

**MesoNet: a Compact Facial Video Forgery Detection Network** (Kat)

Abstract:

* ‘a method to to automatically and efficiently detect face tampering in videos’ using videos manipulated with Deepfake and Face2Face
* “a very successful detection rate with more than 98% for Deepfake and 95% for Face2Face.”

Introduction:

* What helps to spot fake images: “There have been several approaches to detect image forgeries [8, 19], most of them either analyze inconsistencies relatively to what a normal camera pipeline would be or rely on the extraction of specific image alterations in the resulting image. Among others, image noise [11] has been shown to be a good indicator to detect splicing (copy-paste from an image to another). The detection of image compression artifacts [2] also presents some precious hints about image manipulation.”
* Fake video detection is still a very challenging task because fake image detection methods (more progress) can not be extended to video ones efficiently (“the strong degradation of the frames after video compression”)

1.1 Deepfake:

* “parallel training of two auto-encoders … The last step is to take the target video, extract and align the target face from each frame, use the modified auto-encoder to generate another face with the same illumination and expression, and then merge it back in the video.”
* Potential giveaways: “the extraction of faces and their reintegration can fail, especially in the case of face occlusions: some frames can end up with no facial reenactment or with a large blurred area or a doubled facial contour. … autoencoders tend to poorly reconstruct fine details because of the compression of the input data on a limited encoding space, the result thus often appears a bit blurry. A larger encoding space does not work properly since while the fine details are certainly better approximated, on the other hand, the resulting face loses realism as it tends to resemble the input face”

1.2 Face2Face:

* Changing facial expressions by “a photorealistic and markerless facial reenactment in real-time from a simple RGB-camera”

2 Proposed method:

* “microscopic analyses based on image noise cannot be applied in a compressed video context where the image noise is strongly degraded. Similarly, at a higher semantic level, human eye struggles to distinguish forged images [21], especially when the image depicts a human face [1, 7]. That is why we propose to adopt an intermediate approach using a deep neural network with a small number of layers”
* Meso-4: “a sequence of four layers of successive convolutions and pooling, and is followed by a dense network with one hidden layer. To improve generalization, the convolutional layers use ReLU activation functions that introduce non-linearities and Batch Normalization [10] to regularize their output and prevent the vanishing gradient effect, and the fully-connected layers use Dropout [24] to regularize and improve their robustness. … 27,977 trainable parameters”
* MesoInception-4: “replacing the first two convolutional layers of Meso4 by a variant of the inception module introduced by Szegedy et al [25]. The idea of the module is to stack the output of several convolutional layers with different kernel shapes and thus increase the function space in which the model is optimized. Instead of the 5 × 5 convolutions of the original module, we propose to use 3 × 3 dilated convolutions [30] in order to avoid high semantic. … We have added 1×1 convolutions before dilated convolutions for dimension reduction and an extra 1×1 convolution in parallel that acts as skip-connection between successive modules. … 28,615 trainable parameters”

3 Experiments:

* “175 rushes of forged videos have been collected from different platforms. Their duration ranges from two seconds to three minutes and have a minimum resolution of 854 × 480 pixels. All videos are compressed using the H.264 codec but with different compression levels, which puts us in real conditions of analysis.”
* “Python 3.5using the Keras 2.1.5 module … Weights optimization of the network is achieved with successive batches of 75 images of size 256 × 256 × 3 using ADAM [13] with default parameters (β1 = 0.9 and β2 = 0.999). The initial learning rate of 10^(−3) is divided by 10 every 1000 iterations down to 10^(−6)”
* Image classification: Deepfake with Meso-4 is 0.891, with MesoInception-4 is 0.917; Face2face with Meso-4 is 0.832, with MesoInception-4 is 0.813 (each frame independently)
* Image aggregation: 0.969, 0.984, 0.953, 0.953 respectively
* “Deepfake-generated faces tend to be blurry, or at least to lack details, compared to the rest of the image that was left untouched”

Conclusion:

“These days, the dangers of face tampering in video are widely recognized. This paper provides two possible network architectures to detect such forgeries efficiently and with a low computational cost. Our experiments show that our method has an average detection rate of 98% for Deepfake videos and 95% for Face2Face videos under real conditions of diffusion on the internet. Moreover, the visualization and study of the layers and filters shows that the eyes

and mouth play a paramount role in the detection of faces forged with Deepfake.”

**CNN-generated images are surprisingly easy to spot... for now** (Kat)

Abstract:

* Paper explores whether it is possible to put together a universal detector of fake images
* Data was generated using 11 different methods: “ProGAN, StyleGAN, Big- GAN, CycleGAN, StarGAN, GauGAN, DeepFakes, cascaded refinement networks, implicit maximum likelihood estimation, second-order attention super-resolution, seeing- in-the-dark”

“with careful pre- and post-processing and data augmentation, a standard image classifier trained on only one specific CNN generator (Pro- GAN) is able to generalize surprisingly well to unseen architectures, datasets, and training methods”

Introduction:

* “It is natural, therefore, to ask whether today’s CNN- generated images contain common artifacts, e.g., some kind of detectable CNN fingerprints, that would allow a classifier to generalize to an entire family of generation methods, rather than a single one.”
* “In this paper, we show that, contrary to this current understanding, classifiers trained to detect CNN-generated images can exhibit a surprising amount of generalization ability across datasets, architectures, and tasks.”
* “we create a new dataset of CNN- generated images, the ForenSynths dataset”
* “We find that data augmentation, in the form of common image post- processing operations, is critical for generalization, even when the target images are not post-processed themselves. … We show that when the correct steps are taken, classifiers are indeed robust to common opera- tions such as JPEG compression, blurring, and resizing.”

“In summary, our main contributions are: 1) we show that forensics models trained on CNN-generated images exhibit a surprising amount of generalization to other CNN synthe- sis methods; 2) we propose a new dataset and evaluation metric for detecting CNN-generated images; 3) we exper- imentally analyze the factors that account for cross-model generalization.”

Related work:

* Ro ̈ssler et al. [38] – “evaluated methods for detecting face manipulation techniques, including CNN- based face and mouth replacement methods”
* Marra et al. [28] – “likewise showed that simple classifiers can detect images created by an image translation network [18], but did not consider cross-model transfer.”
* Cozzolino et al. [13] – bad translation between models
* “Zhang et al. [48] finds that classifiers generalize poorly between GAN models.”
* “Researchers have also proposed methods for identifying which, of several, known GANs generated a given image [29, 45]. ”
* “This line of work has found, like us, that simple, su- pervised classifiers are often effective at detecting manipu- lations [49, 42]. ”
* “Researchers have shown, recently, that common CNN designs contain artifacts that reduce their representational power. ”

A dataset of CNN generated models:

* “All of these models have an upsampling- convolutional structure”

“**Perceptual loss** We consider models that directly opti- mize a perceptual loss [19], with no adversarial training. This includes Cascaded Refinement Networks (CRN) [10], which synthesizes images in a coarse-to-fine manner, and the recent Implicit Maximum Likelihood Estimation (IMLE) conditional image translation model [25].”

* Included: low-level vision, similar to long-exposure photography
* Also, evaluating models on FaceForensics++ dataset

Detecting CNN synthesized images:

* “The main idea of our experiments is to train a “real-or-fake” classifier on this ProGAN dataset, and evaluate how well the model generalizes to other CNN-synthesized images.”
* “All of our models are trained with images that are randomly left-right lipped and cropped to 224 pixels. We evaluate several additional augmentation variants:” no augmentation, Gaussian blur, JPEG, Blur+JPEG (0.5), Blur+JPEG (0.1)
* Regarding testing dataset: “During testing, each image is center-cropped without resizing in order to match the post-processing pipeline used by models during training. No data augmentation is included during testing”
* Augmentation generally improves generalization and robustness
* “Image diversity improves performance”

4.7 Qualitative analysis:

* studying whether models pick up “subtle low-level features generate by CNN architectures, or high-level features such as visual quality” by fakeness (the 0th, 25th, 50th, 75th, 100th percentile)
* results: “In most datasets, we observe little noticeable correlation between the model predictions and the visual quality of the synthesized images. However, there is a weak correlation in the BigGAN and StarGAN datasets” => models learn low-level features
* “Artifacts of CNN image synthesis”: “While the real image spectra generally look alike (with minor variations due to differences in the datasets), there are distinct patterns visible in images generated by different CNN models. … Interestingly, the most effective unconditional GANs (BigGAN, ProGAN) contain relatively few such artifacts. Also, DeepFake images do not contain obvious artifacts.”

## **A Deep Learning Approach To Universal Image Manipulation Detection Using A New Convolutional Layer**

In this paper, we propose a universal forensic approach to performing manipulation detection us- ing deep learning. In their current form, convolutional neural networks will learn features that capture an image’s content as opposed to manipulation detection features. To overcome this issue, we *develop a new form of convolutional layer that is specifically designed to suppress an image’s content and adaptively learn manipulation detection features.*

Recent experimental evidence has shown that tools initially devel- oped to perform steganalysis are capable of detecting a wide variety of image editing operations These tools from steganalysis operate by building local models of pixel depen- dencies by analyzing the joint distribution of pixel value pre- diction errors, then extracting detection features from these joint distributions [16, 4].

A convolutional layer is typically followed by a pooling layer whose purpose is to reduce the dimensionality of the feature maps. This reduces the computational cost associated with training the network and decreases the chances of over-fitting.

in their existing form, CNNs will tend to learn features that represent an image’s content rather than manipulation detection features. This effect has recently been observed by Chen et al. during their efforts train a CNN to perform median filtering detection [2].

The key idea behind developing this layer is that certain local struc- tural relationships exist between pixels independent of an image’s content. Manipulations will alter these local rela- tionships in a detectable way. As a result, manipulation detection feature extractors must learn the relationship be- tween a pixel and its local neighborhood while simultane- ously suppressing the content of the image so that content dependent features are not learned. For this to occur, the first convolutional layer must not be allowed freely evolve into any set of filters. Instead, it must be constrained to evolve filters with the desired properties described above.

Make constrained convolutional layers. convRes is the constrained layer.

## **Distinguishing Computer Graphics from Natural Images Using Convolution Neural Networks**

## Distinguishing computer generated graphics from real photographs. Uses a CNN with a custom pooling layer to optimise the current best-performing algorithms feature extraction scheme.

## 

## Some recent advances in image processing, like the real- time facial reenactment face2face [2], show the importance of having some tools to distinguish computer graphics (CG) from natural photographic images (PG). Although the distinction between CG and PG depends not only on image properties, but also on cognitive characteristics of viewers [3], people show inaccurate skills at differentiating between altered and non-altered images [4].

## 

## Ng et al. [6] is the first paper to mention the expected difference between CG and PG images, mainly consisting in the image smoothness due to triangles. Dirik et al. [7] consider that this difference better resides in the statistical noise properties of the image. Wang et al. [8] consider that the image edges are more relevant for this problem. For more detailed overviews, the reader can refer to Tokuda et al. [16], as well as to Wang et al. [8] and Ng and Chang [17]. In the rest of the paper, we will mainly compare our method to Wu et al. [12], i.e. an histogram based method derived from [13], known as one of the most effective from the current state of the art.

## 

## Statistical properties of filtered images are good discrimi- nators for distinguishing CG from PG, whether computations involve image gradient [12] or more sophisticated wavelet transformations [9]–[11]. For all these methods, the question of using the best filters is central, i.e. finding the ones that will extract the most meaningful information.

## 

## Convolution layers model filters while densely connected layers can replace efficiently other classification methods. Motivated by those observations, we implemented a special pooling layer to extract statistical features within a CNN framework and trained the network to distinguish CG from PG.

## 

## Split the input images in Np tiles of resolution 100 x 100. Trade-off between execution time and statistical meaningfulness. Then comes filtering, statistical feature extraction and classification. Computes Nf filtered images, extract Ns statistical quantities for each filtered image and feeds Nf x Ns long feature vector to a usual classifier which estimates the posterior probability of each class. Training found 32 different filters from the first convolution layer of the Stats-2L model. Tried different paramters for the number of layers and number of kernels per layer.

## 

## aThis Section explains how statistical information is ex- tracted from the convoluted images. In a deep-learning context, this operation can be viewed as a pooling layer. Usually, after convolution layers, a local maximum pooling is computed to reduce the dimension of the data representation before classification. As for image forensics, other global statistical quantities are known to be more useful. Thus a special pooling layer is developed for adapting neural nets to this particular task.

## 

## Either use four statistics or histogram Guassian with 11 bins. For classification use either LDA, SVM, Neural Network. Neural Networks include multi-layer perceptrons, which are an example of model free classifying methods. 1024 ReLU activated neurons and a read-out layer with 2 SoftMax activated neurons.

## 

## We also use dropout on the hidden layer as described by Strivastava et al. [25] to avoid over-fitting. The loss to minimize is the cross-entropy function for the two-classes problem, which can be interpreted as the minus log-likelihood of the label data under our model. Finally, we used Adam algorithm [26] to optimize synchronously MLP’s and convolution layers’ weights.

## 

## First filtering, then feature extraction (both can be done in many ways) and then classification (not everything have to be NN). For 3600 images, 3 databases were constructed. First, the green channel of each image was selected, then divided into 70, 20, 10 TTV. Implemented using TF. Parameter tuning is described. Training the models took 20+ hours. Stats-2L performs best.

## 

## Optimising the filtering step improves significantly the accruacy. Plugging an SVM or LDA on top of the feature extraction instead of MLP leads to less accurate classification.

## 

## 

## 

## Deep Learning with Python (Greg)

Some introductory points from the first 5 chapters:

First, interesting, but not really essential to know for Capstone:

1. Kernel: that maps any two points in your initial space to the distance between these points in your target representation space → saves computation by having to actually calculate all the coordinates in the new space (SVM uses this). Problem with this is that these useful representations have to be found manually which is called feature engineering which is tough (neural networks are great because this feature engineering occurs automatically in the model)
2. Gradient boosting techniques are the best current methods for non-perceptual data, however, we are dealing with perceptual data
3. Kaggle is dominated by gradient boosting machines and deep learning
4. Deep learning has become popular of late due to high accuracy as a result of:
   1. Better activation functions
   2. Better initializations of weight (something for us to consider)
   3. Better optimisation schemes (we should probably use RMSProp or Adam)
5. Dense and fully connected layer mean the same thing
6. Tensors are just multi dimesional arrays (matrices are a subset of tensors)
7. In short, all the neural net is doing is geometrically transforming (in a complicated high dimension) some input tensor into a representation that can be separated (think of two pieces of crumpled paper, one blue and one red. Each layer uncrumples the paper a little bit so the pieces of paper can finally be separated)
8. Simplified process of building a neural net:
   1. Draw a batch of training samples x and corresponding targets y.
   2. Run the network on x to obtain predictions y\_pred.
   3. Compute the loss of the network on the batch, a measure of the mismatch  
      between y\_pred and y.
   4. Compute the gradient of the loss with regard to the network’s parameters (a  
      *backward pass*).
   5. Move the parameters a little in the opposite direction from the gradient—for  
      example W -= step \* gradient—thus reducing the loss on the batch a bi
9. Basic components of neural nets:
   1. *Layers*: these are combined into a *network*
   2. The *input data* corresponding *targets*
   3. The *loss function*: defines the feedback signal used for learning
   4. The *optimizer*: which determines how learning proceeds
10. Activation functions purpose: Introduce non-linearity into the model
11. Feature engineering: using non-learnt domain specific knowledge that is hardcoded as transformations to the data before feeding it to the neural net - should we do any of this? Try think how humans would solve the problem and transform the input data accordingly (hands on a clock example). Need fewer resources and less data

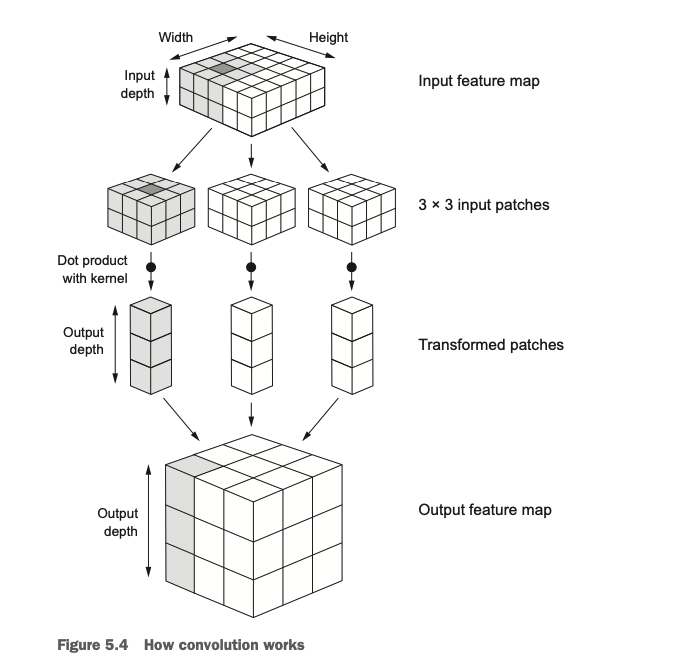
Some more useful points:

1. Default shape of tensors for images in tensorflow is: (samples, height, width, color\_depth)
2. And for videos: (samples, frames, height, width, color\_depth)
3. Remember that all the standard operators (+ - \* /) will perform element wise, need to specify if we want to do tensor multiplication
4. The step used by the optimiser is important:
   1. Too large - you might miss min and end up at random points on the curve
   2. Too small - you will need many iterations and might get stuck in a local min
5. Momentum helps optimisation a lot - faster convergence and helps avoid getting caught in local minima (uses previous gradients to store a velocity and goes past the local min with this velocity to check if there might be another better min further down the line)
6. Epoch: an iteration over the entire dataset - this will control the total number of iterations when updating weights so it is NB to choose a number that will prevent overfitting
7. Which neural nets to use where:
   1. Sequence data: Recurrent layers such as LSTM layers
   2. Image data (stored in 4D tensors): 2D convolution layers
8. Choosing the right topology (width and depth) is NB: theres no real science behind it, just need to try things and have a good feel for neural net architecture. There are some guidelines to follow though
9. Can use sequential class (only for linear stacks of layers) when programming or use the functional API which allows for custom architecture (eg to build directed acyclic graphs) - sequential should be fine for us
10. Binary Cross entropy is the best loss function when dealing with output probabilities which is what we’ll be doing. It comes from information theory: measures the distance between probability distributions or, in the case of neural nets, the distance between the ground-truth distribution and our predictions
11. Pre-processing is NB - we should start here and explore the options that Yorgos was telling about because it would save us a lot of time if this code already exists. We must also see if we need to scale each feature, if the features have different ranges. Some NB points to consider:
    1. We need to ensure the data is in tensor format so we can feed it into the neural network
    2. Normalization: We probably need to convert these values into float32 digits between 0 and 1 (or something else if more appropriate but 0 and 1 is good because it is more likely to converge. Should usually also normalise to 0 mean and SD 1). Data should have similar ranges and be fairly homogenous. It can prevent convergence.
    3. Missing values - usually its fine to encode missing values as 0. But if the test data has missing values, make sure you have missing values in the training data and if not, artificially generate them because the neural net needs to learn how to deal with them!
12. Relus seem best for image classification with our last layer being a softmax/sigmoid activation function to normalise the output to a probability
13. It seems like RMSProp or Adam are the best optimisers
14. Too wide a network will take too long to train, if it’s not wide enough then we will lose some info resulting in poorer accuracy
15. General rule: avoid having hidden layers with fewer dimensions than that of the input data eg. if the input data has 40 features, each layer in the neural net should have more than 40 nodes. But not too many as this will overfit the data
16. If we use one-hot encoding, we must use categorical\_crossentropy and if we encode with integers we need to use sparse\_categorical\_crossentropy
17. For regression problems (not our case), you need to standardise each column to mean 0, SD 1. Dont use any activation function in the final step
18. General rule: less data means overfitting is more likely → so mitigate this by using a smaller model
19. Autoencoders: the input is the target
20. When doing mini batch stochastic gradient descent, we should feed in multiples of 2 as the mini batches as this works better with the GPUs memory and will make the process run faster
21. Need to figure out how to split our data between training, validation and test. Two NB points to consider:
    1. Data representativeness: we need to choose data that represents the entire dataset when training (and testing). A random shuffle usually achieves this but it might be a good idea to check the data to ensure that a variety of images are included in both sets
    2. Redundancy: make sure we dont have repeats! Because then we might have test data that we already trained on
22. If we dont have enough data, we need to use iterated k-fold validations with shuffling (performed really well on Kaggle). It just entails performing k-fold several times and shuffling the data before each iteration of the k-fold CV
23. We should use the k-fold to tune our hyperparameters, then retrain again on ALL the training data and only run the test data after those two steps
24. Ways to prevent overfitting (we have enough data I think, it might be our contraints on resources thats a problem and so we’d have to use less data and then we must implement the below techniques):
    1. Use more trainig data
    2. Reduce network size - massive networks can learn the training data perfectly as they have a lot of *memorization capacity*. We must start with few layers and slowly increase until our validation set stops seeing diminishing loss
    3. Regularize weights with constraints - adjust loss function with L1 or L2 (keep in mind that this loss is only added at training time in Keras so it is likely that *test* loss will be lower than training loss if you do this)
    4. Dropout - randomly set some of the outputs of a layer to zero. Dropout rate is the fraction of units that are dropped out. At test time no values are dropped out - all weights are just scaled down by the dropout weight (or just scale up at training time). Hinton and bank teller experience conspiracies inspired him
25. Steps to follow when attempt to train data:
    1. Ensure input data has predictive info for the target e.g. don’t try predict stock prices with recent price history. Decide what the problem is - scalar regression/ classification/…
    2. Choose measure of success: precision/recall/ROC/AUC etc.
    3. Choose evaluation protocol: K-fold CV/Iterated K-fold CV
    4. Prepare data:
       1. Must be in tensor format
       2. Must be scaled to small values - usual [-1,1] or [0,1]
       3. Different features must have similar ranges otherwise normalise them
       4. Try feature engineering (transforming data so the neural net can learn easier) especially if you have small amounts of data
       5. Choose correct last-layer activation and loss function. It’s sigmoid and binary\_crossentropy for Binary Classification tasks
       6. Try things and use validation set to test:
          1. Try different architectures (change number of layers and width)
          2. Train for more epochs
          3. Add dropout or L1/L2 regularization
          4. Try different hyperparameters (learning rate etc.)
          5. Feature engineering - add or remove relevant or irrelevant features to the input data

Useful points specific to computer vision (will be difficult to understand without having read the chapter) - Super useful link to understand CNNs if you don’t have enough time:

<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

1. Dense layers vs conv layers: dense layers learn global input features (eg patterns involving all pixels in the MNIST case) whereas conv layers learn local patterns (patterns found in small 2D windows)
2. The above means convnets:
   1. Learn translation invariant patterns - a learnt pattern in a small 2D area can be recognised anywhere on the image whereas a dense layer would have to relearn it which is less efficient
   2. Learn spacial hierarchies - eg. the first layer learns about images edges, the next layer learns about patterns of the previous layer’s features etc.
3. Convolutions operate over 3D tensors, called *feature maps*, with two spatial axes (*height* and *width*) as well as a *depth* axis (also called the *channels* axis). For an RGB image, the dimension of the depth axis is 3, because the image has three color channels: red, green, and blue. For a black-and-white picture, like the MNIST digits, the depth is 1 (levels of gray). The convolution operation extracts patches from its input feature map and applies the same transformation to all of these patches, producing an *output feature map*. This output feature map is still a 3D tensor: it has a width and a height. Its depth can be arbitrary, because the output depth is a parameter of the layer, and the different channels in that depth axis no longer stand for specific colors as in RGB input; rather, they stand for *filters*. Filters encode specific aspects of the input data: at a high level, a single filter could encode the concept “presence of a face in the input,” for instance.
4. Two key parameters:
   1. Size of the patches extracted from the inputs - usually 5x5 or 3x3
   2. Depth of the output feature map i.e. number of filters
   3. Conv2D(output\_depth, (window\_height, window\_width)) → in keras



1. More filters (depth) means our net will be better at recognising patterns in images
2. Padding: allows output feature map to be of same dimension as input feature map after conv layer by adding rows and columns to input feature map - allows us to fit convolution windows around every input tile
3. Strides: distance between windows which is 1 by default
4. Pooling: Downsamples (reduces dimension) of feature maps (so do strides but this is better). Extracts windows from input feature map and outputs max value of each channel (usually done with 2x2 window which downsamples by a factor of 2). So, in essence, the conv layers linearly transform each window of input map by a learned linear transformation whereas pooling transforms via a hardcoded max tensor operation
5. Note that the level of generality (and therefore reusability) of the representations extracted by specific convolution layers depends on the depth of the layer in the model. Layers that come earlier in the model extract local, highly generic feature maps (such as visual edges, colors, and textures), whereas layers that are higher up extract more-abstract concepts (such as “cat ear” or “dog eye”). So if your new dataset differs a lot from the dataset on which the original model was trained, you may be bet- ter off using only the first few layers of the model to do feature extraction, rather than using the entire convolutional base.
6. Activations of higher layers carry less and less information about the specific input being seen, and more and more infor- mation about the target. Earlier layers contain more general information which can be leveraged to a wide range of classification tasks

## Xception: Deep Learning with Depthwise Separable Convolutions (Greg)

This paper discusses an architecture which has an extremely high recognition accuracy

1. Architecture name: Xception
2. In short, it improves on the Inception model: (also a successful network architecture)
   1. 27 layers deep
   2. Problem: Conv net kernel (the matrix that you multiply by to extract the edges, blurriness etc.) needs to map cross channel correlations and cross spatial ones. Solution: Inception first looks at cross-channel correlations by using a *set* of 1x1 convs and then 3x3 or 5x5 after to look at cross spatial correlations. Underlying assumption: Cross-channel correlations and spatial correlations are sufficiently decoupled
   3. Introduces the inception layer: Uses filters with multiple sizes that operate on the same level making the model wider rather than deeper (i.e. 3x3, 5x5 convs and max pooling are all performed on the same layer (this is relevant to understanding the Inception framework, but not so NB to understand how Xception builds on it)
3. Xception improves on the above by using depthwise separable convolutions - they make the assumption that cross-channel correlations and spatial correlations can be ENTIRELY decoupled
4. The big difference: Inception has spatial convolutions that operate on non-overlapping segments of the output channels (i.e. the output of the 1x1 convolution). Xception (Xtreme Inception) has a convolution for each and every segment of the output channel
5. Parameters used: L2 weight decay of 1e-5, 0.5 dropout rate
6. Time taken: 3 days on Imagenet data, 3 months on Imagenet (350mil pics) → they used 60 NVIDIA GPUs
7. Results: Improvements over Inception Res-net (latest inception neural network)
8. Potential to explore: Spectrum between regular convolutions and depthwise separable convs

DenseNet (Greg)

1. Include connections between non-subsequent layers - more efficient as there are fewer parameters (no need to relearn redundant feature maps), prevents information loss if there are many layers and alleviates vanishing gradients problem.
2. Can use stochastic dropping to randomly drop layers allowing for better gradient flow.
3. Different to resnet: instead of combining features through summation before inputting into a layer, they are concatenated

FaceForensics (Greg):

1. Important for the pre-processing: we need to identify the face and then look at a slightly larger area – this will massively improve the detection performance (FaceForensics). Look into Thies et al for the paper that describes this method. They refer to this as using domain specific knowledge.

2. MesoInception-4: They propose mean squared loss instead of cross entropy

3. If you try DenseNet, initialize with the weights they have obtained through training on some other dataset

4. READ THE APPENDIX FOR EXACT DETAILS ON HOW THEY TRAINED VARIOUS COMPONENTS OF THE NETS (SPECIFICALLY PART C)

5. Training size is important, especially for low quality images

Deep residual learning for image recognition (ResNet) (Greg):

1. A lot of detail but in summary: Deep is better – but the problem of degradation arose where deeper led to lower accuracies (degradation is when deeper networks lead to higher training and test error that is not a result of overfitting). ResNets solve this problem and proceed to build networks with 100s of layers.

2. Maps a residual mapping: F(x) = H(x) – x where H(x) is the desired mapping and F(x) is supposedly much easier to optimise. It does this by adding identity shortcut mappings (between non-subsequent nodes) which don’t add any parameters to the model

3. THE BEST MODELS ARE ENSEMBLES I.E. COMBINED MODELS. WE SHOULD DO THIS

Use batch normalisation: Adjust hidden layer outputs by subtracting mean and dividing by standard deviation

Seems like cross validation is never used – just a validation set

Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos

Abstract:

* *“locating manipulated regions (i.e., performing segmentation), which are mostly created by three commonly used attacks: removal, copy-move, and splicing” -- from intro the citations are [6, 32, 7]*
* Main contribution: “*We have designed a convolutional neural network that uses the multi-task learning approach to simultaneously detect manipulated images and videos and locate the manipulated regions for each query. Information gained by performing one task is shared with the other task and thereby enhances the performance of both tasks.*”
* Performance on data: “*Experiments using the FaceForensics and FaceForensics++ databases demonstrated the networks effec- tiveness against facial reenactment attacks and face swapping attacks as well as its ability to deal with the mismatch condition for previously seen attacks.*” + fine-tuning to improve network’s performance on unseen attacks

Introduction:

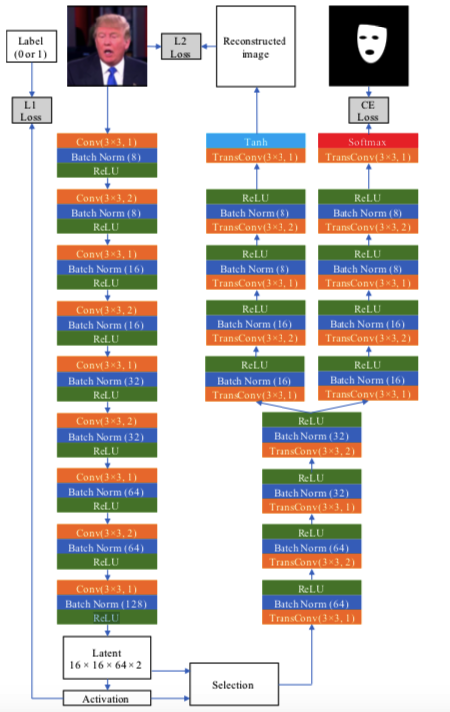
* Dangerous because: manipulation can be done in real time and “thanks to advances in speech synthesis and voice conversion [19], an attacker can also clone a person’s voice (only a few minutes of speech are needed) and synchronize it with the visual component”
* Main available databases: “the CGvsPhoto database [25], the Deepfakes databases [2, 16, 17], and the FaceForensics/FaceForensics++ databases [26, 27]”
* Previous work on transferability by Coz- zolino et al. [11]: “developed an autoencoder-like architecture that supports generalization and can be easily adapted to a new domain with simple fine-tuning”
* Locating manipulated regions: full-scale images required; Previous work by Rahmouni et al. [25], Nguyen et al. [21] and Rossler et al. [26]; + approach effectively segments manipulated regions in Face2Face images; - “these methods need to score many overlapped windows by using a spoofing detection method, which takes a lot of computation power”
* Perform classification, segmentation, and reconstruction while sharing information between all three tasks

Related work:

* On manipulating videos (visual material and voice)
* Detection: “The noise-based method proposed by Fridrich and Kodovsky [12]” -- improved version for CNN shows that automatic feature extraction is effective, fine-tuning and transfer learning work with pre-trained models, other methods: “a constrained convolutional layer [8], using a statistical pooling layer [25], using a two-stream network [31], using a lightweight CNN network [2], and using two cascaded convolutional layers at the bottom of a CNN [23]”, Lastly, “Li et al. [...] developed a network for detecting eye blinking, which is not well reproduced in fake videos [17]”
* Locating manipulated regions:

1. segmenting the entire input image: “ commonly used to detect removal, copy-move, and splicing attacks [6, 7]” (“A slightly different segmentation approach is to return the boxes that represent the boundaries of the manipulated regions instead of returning segmentation masks”) - for this paper this method is used but segmentation is only on facial areas -- major reduction in computations

2. “repeatedly performing binary classification using a sliding window”: “used for detecting spoofing regions generated by a computer to create spoof images or videos”, binary classification on each position of the sliding window, (non-overlapped) [25] or (overlapped) [21, 26]) - not used in this paper

Proposed Method:

* Output: “The autoencoder outputs the reconstructed version of the input image (which is used only in training), the probability of the input image having been spoofed, and the segmentation map corresponding to this input image”
* “Video inputs are treated as a set of frames”
* Preprocessing done earlier, images are 256x256
* “For video inputs, we average the probabilities of all frames”
* Y-shaped autoencoder and 3 types of loss (activation, segmentation and reconstruction)
* All three tasks are equally important: “Unlike Cozzolino et al. [11], we set the three weights equal to each other”

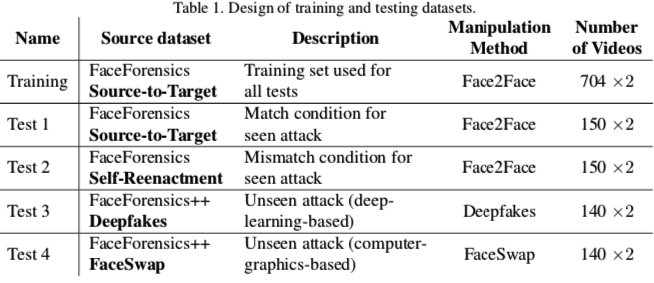
Network architecture:

“For simplicity, we directly feed normalized images into the autoencoder without con- verting them into residual images [11]. Further work will focus on investigating the benefits of using residual images in the classification and segmentation tasks.”;

“ADAM optimizer [15] with a learning rate of 0.001, a batch size of 64, betas of 0.9 and 0.999, and epsilon equal to 10−8.”

Experiments:

* two databases: FaceForensics FaceForensics++ (segmentation masks included); for each dataset, light compression (quantiza- tion = 23)
* First 200 frames from a training video and first 10 frames from a validation/testing video
* “normalization with mean = (0.485,0.456,0.406) and standarddeviation = (0.229, 0.224, 0.225)”
* No data augmentation for training

“we trained the shallower ones with 100 epochs and the deeper ones with 50 epochs. For each method, the training stage with the highest accuracy for the classifica- tion task and a reasonable segmentation loss (if available) was used to perform all the tests described in this section”

Results:

* “Although degraded, the segmentation accuracies were still high [...] When dealing with unseen attacks, this segmentation information could thus be an important clue”
* New weight settings are better because “*Proposed New* methods, which used the new weighting settings, had higher accuracy than the *Proposed Old* method”
* Validation set used for fine-tuning; 10 frames per video, 100 videos for training and 40 for testing; 50 epochs
* After fine-tuning: “Their classification and segmentation accuracies increased around 25% and 8%, respectively, which are re- markable compared with the small amount of data used. The one exception was the *Proposed Old* method – its seg- mentation accuracy did not improve.”

Conclusion:

* Y-shaped autoencoder is good for classification as well as segmentation (segmenting facial areas and no sliding window)
* Segmenting facial areas only is a novel contribution which worked well and massively decreased computation
* “Information sharing among the classification, segmentation, and re- construction tasks improved the network’s overall perfor- mance, especially for the mismatch condition for seen attacks”
* Fine-tuning is effective despite small training dataset size

Exposing DeepFake Videos By Detecting Face Warping Artifacts

Abstract: (Note: in this paper, “DeepFake” == “*AI-generated fake”*)

* Main idea: “*current DeepFake algorithm can only generate images of limited resolutions, which need to be further warped to match the original faces in the source video. Such transforms leave distinctive artifacts in the resulting DeepFake videos*”
* Advantages of proposed method:

1. Does not require DeepFake generated images (simple processing operations are enough to make it an example of a fake image) - computationally quicker and easier
2. Artifacts exist in DeepFake videos from different sources -- hence authors’ method is more robust than others

Introduction:

* Manipulating becomes more sophisticated due to “large-volume training data and high-throughput computing power, but more to the growth of machine learn- ing and computer vision techniques that eliminate the need for manual editing steps”
* DeepFake videos “With proper post-processing, [...] can achieve a high level of realism.”
* Authors’ proposition: compare “the generated face areas and their surrounding regions”
* “Based on our collected real face images from Internet and corresponding created negative data, we train four CNN models: VGG16 [31], ResNet50, ResNet101 and ResNet152 [11]. We demonstrate the effectiveness of our method on a DeepFake dataset from [20] and test several fake videos on YouTube.”

Related Works:

* Manipulating methods: “The GAN model inspired many subsequent works for image synthesis, such as [8, 28, 2, 13, 32, 30, 21, 36, 3, 5]. Liu *et al.* [21] proposed an unsupervised image to image translation framework based on coupled GANs [...] basis for the DeepFake algorithm.”
* Recycle-GAN [3], StarGAN [5]
* Resampling detection: “detecting transforms or the underly- ing resampling algorithm has been extensively studied, *e.g.*, [25, 26, 22, 15, 16, 17, 7, 24, 27, 12, 4]”, “the performance of these methods are affected by the post- processing steps, such as image/video compression [...] not subject to simple modeling. Besides, these methods usually aim to estimate the exact resampling operation from whole images” -- Authors believe that a simple solution is compare facial regions with the rest of the image
* Image/video detection: “two-stream CNN for face tampering de- tection” by Zhou *et al.* [35], “CNN model to trace de- vice fingerprints for forgery detection” by NoisePrint [6], lack of eye blinking by Li *et al.* [20], “inconsistency in head pose” by Yang *et al.* [34], “[19] exploited the color disparity between GAN generated images and real images in non-RGB color spaces” ([23] also did that), MesoNet and incorporating RNN to CNN

Methods:

* Main idea: “the facial region and surrounding regions in the original image/video frame will present artifacts, the resolution inconsistency due to such transforms after the subsequent compression step to generate the final image or video frames”
* training on “face images collected from the Internet.” (24, 442 JPEG face images)
* Negative images are generated by “simulating the affine face wrapping step directly”:
  + “extract the face region using software package dlib [14]”
  + “align faces into multiple scales and randomly pick one scale, which is then smoothed by a Gaussian blur with kernel size (5 × 5)” -- “simulate different kinds of resolution inconsistency”
  + “smoothed face undergoes an affine warp back to the same sizes of original faces”
* For all training images: augmentation on colour plus “change the shape of affine warped face area to simulate different post-processing procedure in DeepFake pipeline”
* As part of pre-processing, the faces (in bounding boxes with a varying amount of additional space are cropped out) -- not squares!! -- then resized to 224x224
* Models: VGG16 [31], ResNet50, ResNet101 and ResNet152 [11]
* To improve robustness each example is copied and cropped independently 10 times, then the prediction is averaged

Experiements:

* Negative examples are generated during training
* Set up: “batch size as 64, learning rate starting from 0.001 and decay 0.95 after each 1000 steps. We use SGD optimization method and the training process will be terminated until it reaches the maximum epoch.”
* VGG is trained from scratch for 100 epochs, others rely on pre-trained on ImageNet models (20 epochs)
* After for 20 epochs models are “fine-tuned using hard mining strategy”

Evaluations of UADFV:

* easy dataset (98 videos (balanced). “Each video has one subject and lasts approximate 11 seconds.”) -> 83.3% (VGG) ,97.4%,95.4%,93.8% -- ResNet is better “due to the residual connections” which makes learning more efficient, however for ResNet “as the depth of network increases, the classification-relevant information diminishes”
* For video detection: “we feed all frames of the video to the CNN based model and then return average the top third of the output score as the overall output of the video” (results 84.5% (VGG), 98.7%, 99.1%, 97.8%)

Evaluations on DeepfakeTIMIT:

* Dataset has LQ (64x64) and HQ (128x128) ~4 seconds-long videos, “Each fake video set has 32 subjects, where each subject has 10 videos with faces swapped”; original videos from VidTIMIT dataset [29]; “LQ and HQ video sets are 84.6%,99.9%,97.6%,99.4% and 57.4%, 93.2%, 86.9%, 91.2%”
* Plus tested on youtube videos -> effective

Comparison with State-of-the-arts (two-stream NN, Meso-4 and MesoInception-4): noteable outperformance on HQ video

Conclusion: “First, we would like to evaluate and improve the robustness of our de- tection method with regards to multiple video compression. Second, we currently using predesigned network structure for this task (*e.g.*, resnet or VGG), but for more efficient de- tection, we would like to explore dedicated network struc- ture for the detection of DeepFake videos.”

**Bag of Tricks**

Baseline training: sample images, decode into 32-bit floating point raw pixel values. Flip with 0.5 probability, scale hue, saturation, and brightness with coefficients [0.6,1.4], add PCA noise with a coefficient sampled from normal distribution, normalise RGB channels. Initiliase weights using Xavier algorithm. Nesterov Accelerated Gradient (NAG) descent is used for training. The learning rate is initialized to 0.1 and divided by 10 at the 30th, 60th, and 90th epochs. It is now more efficient to use lower numerical precision and larger batch sizes during training.

Efficient training for large-batch training: linearly increase the learning rate with the batch size. Learning rate warmup also possible. No bias decay. FP16 could help (instead of FP32). Compared to the baseline with batch size 256 and FP32, using a larger 1024 batch size and FP16 reduces the training time for ResNet-50 from 13.3-min per epoch to 4.4- min per epoch. Increased accuracy compared to baseline model.

Section 4 is just on ResNet improvements, not relevant.

Section 5: training refinements: cosine learning rate decay (!) Label smoothing: in a normal setting, the loss function encourages the output scores to be dramatically distinctive which potentially leads to overfitting. Knowledge Distillation (KD), Mixup training: augmentation before training: each time two examples are sampled, a new example is formed by a weighted linear interpolation of these two examples, where lambda (weight parameter) is sampled from the Beta() distribution. Only the new example is used.

**Simple features**

Despite the fact that many face editing algorithms seem to produce realistic human faces, upon closer examination, they do exhibit artifacts in certain domains which are often hidden to the naked eye.

Method is based on a classical frequency domain analysis, followed by a basic classifier.

Deep generative model are used extensively to produce artificial images with realistic appearance. Necessity: approximate the true data distribution of a given training set.

Pipeline: input, preprocessing: [grey-scale, feature extraction (DFT), amplitude spectrum 2D, Azimuth averaging, Amplitude spectrum 1D, training: [classifier].

Introduces a new data set: facesHQ. (and celebA).

This algorithm works well on medium to high-resolution images. 91% on benchmark ff++.

In the same vein, [13] introduces deep forgery discriminator with a contrastive loss function and [12] incorporates temporal domain information by employing Recurrent Neural Networks (RNNs) upon CNNs. While deep learning methods show promising performance, a key concern is that all these methods can be easily learnt by the GAN. In particular, by incorporating them in to the GAN’s discriminator, the generator can be fine-tuned to learn a countermeasure for any differentiable forensic.

Transform every sample from the spatial domain to the 1D frequency do- main, reducing 1024x1024x3 high quality color images to 722 features (1D Power Spectrum).

*Ideas:* combine multiple inputs, normal preprocessing and preprocessing into 1D power spectrum. OR combine this technique with RNN, see if temporal element visible in 1D power spectrum.

For ff++, Finally, we compute the average classification rate per video, applying a simple majority vote over the single frame classifications: 91% with a SVM.

**Optical Flow**

Proposing the adaption of optical flow fields to exploit possible inter-frame dissimilarities.

Optical flow [4, 3] is a vector field which is computed on two consecutive frame f(t) and f(t + 1) to extract apparent motion between the observer and the scene itself. In particular, our hypothesis is that the optical flow is able to exploit discrepancies in motion across frames syn- thetically created with respect to those naturally generated by a video camera.

So, for this reason, for each frame f(t), at a certain time t, a forward flow OF(f (t), f (t + 1)) is extracted us- ing the CNN model for optical flow called PWC-Net [13]. Successively (see Figure 1), the computed forward flow OF(f (t), f (t + 1)) is given as input to a semi-trainable CNN named Flow-CNN, based on some pre-trained network. In our experiments we have tested VGG16 [12] and ResNet50 [6] as backbones. To exploit existing implementations and pre-trained networks trained on raw RGB images, the optical flow is transformed to a 3-channel image using a fixed color-coding approach. The color of pixels is determined by the angle between the flow vector and the horizontal axis, while the intensity of the motion is encoded by the saturation of the color.

Binary detection accuracy of VGG16 is 81.61%. They are obtained on the whole testset of FaceForensics++ for the manipulation Face2Face and witness that the method is able to distinguish the two kinds of videos. **Worthless.**

**Public second place:**

Augmentations, face margin, model capacity (efficientnet-b4), multi-task learning (making use of mask information), ensemble, more frames per video.

<http://shuoyang1213.me/WIDERFACE/WiderFace_Results.html> where retinaFace performed best.

Unet for multi-task learning?

*Augmentation*: Shift, scale, rotate, rgbshift, brightness, contrast, hue, saturation, value. Also, referencing dfdc preview paper( https://arxiv.org/abs/1910.08854 ), I added JpegCompression and Downscale augmentation too. added extensive augmentations including noise and blur, and increased the degree of augmentations. Score jumped to 0.269. I did harder augmentations but it didn't improve, so I stopped tuning augmentations.

*What Didn't Work for LB*

• Different learning rate between encoder - decoder

• Pad rather than resize when feeding the image

• CNN-LSTM architecture

• Construct model so that it uses statistical information across frames (ex. mean, std)

• Tune segmentation loss weight

• Guarantee fake-original pair to exist in each batch

• Use scse option in decoder

• Decrease face margin

• Use Vggface2 pretrained inceptionresnetv1 provided by FaceNet pytorch as encoder

• Validating with compress/downscaled validation set

• Split validation set with actors grouped by face encoding + KMeans

• One cycle learning rate scheduler

• Integrate retinaface confidence score as additional feature

• Stacking across frames with LGBM

*What Worked for LB*

• Tune learning rate considering batch size (batch size 24 - lr 0.0002)

• RAdam + ReduceLROnPlateau

• Tune ratio of decreasing the learning rate when plateau (0.1->0.3)

• Use different frames of each video every epoch (#0->#7->#14->#1->#8->…)

• Increase resolution

• Use conservative fix: multiply constant(<1) to the logits than take sigmoid. It helps improve logloss when the training and testing distributions differ.

**Public third place:**

7xB7 (different seeds) with more conservative avergaing heursitic and trained with hardcore augmentations - 3rd place on private. I used 32 frames for each video.

*Augmentation:* compression, noise, blur, resize with different interpolations, color jittering, scaling and rotations. Function in notebook.

Uses face warping artefacts. 2500 iterations per epoch. SGD, momentum=0.9, weight decay=1e-4. PolyLR with 0.01 starting LR. 75k iterations. APEX WITH MIXED PRECISION? LABEL SMOOTHING? UNet did not help. 3

Deepfake Video Detection Using Recurrent Neural Networks

**Main idea:** There are temporal elements that uniquely identify deepfake videos - CNNs cannot pick these up but RNNs can. So use a CNN to extract frame level features and stack an RNN on top to pick up the temporal signs that an image is a deepfake

**CNN+RNN architecture:** InceptionV3 with fully connected top layer removed - no fine tuning done, weights are adopted. Output after the last pooling layers are used as LSTM inputs.

The LSTM takes a sequence of the CNN outputs and maps them to ‘fake’ or ‘original.’ The sequence length of the RNN is tuned (20, 40 and 80 frames are attempted)

**More ideas:**

1. Use digital media forensics as an additional technique - things like dropped or duplicate frames are indicative of deepfakes
2. Ensure that the training, validation and test data is split in a balanced way
3. Sometimes there is manipulation in only part of the video - therefore they extract continuous subsequences to deal with this (I would argue that this is not so important as humans would instantly recognise a deepfake that is not there for the whole video)

The Deepfake Detection Challenge Dataset (Greg) - NB paper - include in write-up as well

The paper mainly discusses how the created the dataset. Some of it is relevant for the paper but the valuable stuff for network architecture is from Section 6 onwards.

Important notes:

1. More training data the better (David J Sturman - A brief history of motion capture...) - I think we should train on more data
2. FF+ is a first gen dataset meaning there are few identities - leads to overfitting on these identities. From the paper: “due to the small scale, models trained on datasets such as FaceForensics++ usually do not generalize to real Deepfake videos” (Rayhane Mama - Towards deepfake detection that actually works)
3. DFDC Dataset: Basically, its just a better dataset in every regard - include in write-up
   1. DFAE, MM, 3 types of GANs used to create fakes - done in proportion ot techniques used in the wild so the model generalises
   2. MM faceswap: Obviously deepfake videos but are hard to detect on a per frame basis - this can be tough for the model to detect **(motivation for RNNs!)**
   3. 83.9% of training data are fakes. Public test set was used for live leaderboard - private test set was used to rank models for prizes. Test sets were 50% fakes.
   4. Augmentations used: distrators (overlay objects onto videos and facial filters) and augmenters (geometric and colour transforms - brightening, flipping, adding noise etc, frame rate changes etc.)
4. Metric: log loss is used as relative model performance is important. Can also look at weighted PR which is a better indication of performance out in the wild
5. Results: This section just discusses the performance of different Deepfake algos. A common theme is that alot of the algos dont perform well in darker settings.
6. Architecture tips: Refer to links in paper that lead to code and descriptions of the best performing models. But briefly, the top models had this in common:
   1. Clever augmentations: drop parts of the face, use landmarks etc.
   2. Architectures: ensembles of EfficientNets
   3. No forensic methods: non-learned pixel level techniques either dont perform well or just weren’t used by those who entered the competition

**Mixup: BEYOND EMPIRICAL RISK MINIMIZATION**

Abstract:

* Problem “Large deep neural networks [...] exhibit undesirable behaviors such as memorization and sensitivity to adversarial example” -- proposed solution is mixup (‘trains a neural network on convex combinations of pairs of examples and their labels’, “a simple and data-agnostic data augmenta- tion routine”)
* Mixup “improves the generalization of state-of-the-art neural network architectures [...],reduces the memorization of corrupt labels, increases the robustness to adversarial examples, and stabilizes the training of generative adversarial networks”

Introduction:

* ERM = minimization of average error over the training data
* “Strikingly, a classical result in learning theory (Vapnik & Chervonenkis, 1971) tells us that the convergence of ERM is guaranteed as long as the size of the learning machine (e.g., the neural network) does not increase with the number of training data.”
* “ERM allows large neural networks to *memorize* (instead of *generalize* from) the training data even in the presence of strong regularization [...]. Neural networks trained with ERM change their predictions drastically when evaluated on examples just outside the training distribution (Szegedy et al., 2014), also known as *adversarial examples*. This evidence suggests that ERM is unable to explain or provide generalization on testing distributions that differ *only slightly* from the training data.”

Contribution:

* “Mixup extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets”
* Easy implementation, introduces little additional computation, improves robustness and generalization on adversarial examples

From ERM to mixup: mathematically, mixup is a “generic vicinal distribution”

“What is *mixup* doing? The *mixup* vicinal distribution can be understood as a form of data aug- mentation that encourages the model f to behave linearly in-between training examples. We argue that this linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples. Also, linearity is a good inductive bias from the perspective of Occam’s razor, since it is one of the simplest possible behaviors. Figure 1b shows that *mixup* leads to decision boundaries that transition linearly from class to class, providing a smoother estimate of uncertainty.”

Experiments

* ImageNet-2012 classification dataset (1,000 classes), “standard data augmentation practices: scale and aspect ratio distortions, random crops, and horizontal flips (Goyal et al., 2017). During evaluation, only the 224 × 224 central crop of each image is tested.”
* on the CIFAR-10 and CIFAR-100 datasets + speech data

3.4 MEMORIZATION OF CORRUPTED LABELS

* “Following Zhang et al. (2017), we evaluate the robustness of ERM and *mixup* models against randomly corrupted labels. We hypothesize that increasing the strength of *mixup* interpolation α should generate virtual examples further from the training examples, making memorization more difficult to achieve”
* “we compare in these experiments *mixup*, dropout, *mixup* + dropout, and ERM”
* “*mixup* + dropout performs the best of all, showing that the two methods are compatible”

3.5 ROBUSTNESS TO ADVERSARIAL EXAMPLES

* “Among the several methods aiming to solve this problem, some have proposed to penalize the norm of the Jacobian of the model to control its Lipschitz constant (Drucker & Le Cun, 1992; Cisse et al., 2017; Bartlett et al., 2017; Hein & Andriushchenko, 2017). Other approaches perform data augmentation by producing and training on adversarial examples (Goodfellow et al., 2015). Unfortunately, all of these methods add significant computational overhead to ERM. Here, we show that *mixup* can significantly improve the robustness of neural networks without hindering the speed of ERM by penalizing the norm of the gradient of the loss w.r.t a given input along the most plausible directions (e.g. the directions to other training points).”
* three ResNet-101 models, two are trained with ERM on ImageNet-2012, third -- with mixup
* Two experiments: white and black box attacks
* “Overall, mixup produces neural networks that are significantly more robust than ERM against adversarial examples in white box and black settings without additional overhead compared to ERM”

3.6 Tabular Data: omitted (non-image data experiments)

3.7 STABILIZATION OF GANS: “the stabilizing effect of *mixup* the training of GAN”

3.8 ABLATION STUDIES

* *“mixup* is a data augmentation method that consists of only two parts: random convex combination of raw inputs, and correspondingly, convex combination of one-hot label encodings”
* Multiple options for interpolation

“First, *mixup* is the best data augmentation method we test, and is significantly better than the second best method (mix input + label smoothing).

Second, the effect of regularization can be seen by comparing the test error with a small weight decay (10−4) with a large one (5 × 10−4). For example, for ERM a large weight decay works better, whereas for *mixup* a small weight decay is preferred, confirming its regularization effects. We also see an increasing advantage of large weight decay when interpolating in higher layers of latent representations, indicating decreasing strength of regularization.

Among all the input interpolation methods, mixing random pairs from all classes (AC + RP) has the strongest regularization effect. Label smoothing and adding Gaussian noise have a relatively small regularization effect. Finally, we note that the SMOTE algorithm (Chawla et al., 2002) does not lead to a noticeable gain in performance.”

Related work:

“In image classification, for example, one routinely uses rotation, translation, cropping, resizing, flipping (Lecun et al., 2001; Simonyan & Zisserman, 2015), and random erasing (Zhong et al., 2017) to enforce visually plausible invariances in the model through the training data.”

“More related to *mixup*, Chawla et al. (2002) propose to augment the rare class in an imbalanced dataset by interpolating the nearest neighbors; DeVries & Taylor (2017) show that interpolation and extrapolation the nearest neighbors of the same class in feature space can improve generalization.” -- mixup is an extension

“Recent approaches have also proposed to regularize the output distribution of a neural network by label smoothing (Szegedy et al., 2016), or penalizing high-confidence softmax distributions (Pereyra et al., 2017).”

“Like the method of DeVries & Taylor (2017), it does not require significant domain knowledge. Like label smoothing, the supervision of every example is not overly dominated by the ground-truth label. Unlike both of these approaches, the *mixup* transformation establishes a linear relationship between data augmentation and the supervision signal. We believe that this leads to a strong regularizer that improves generalization as demonstrated by our experiments.”

Discussion:

“In our experiments, the following trend is consistent: with increasingly large α, the training error on real data increases, while the generalization gap decreases. This sustains our hypothesis that *mixup* implicitly controls model complexity. However, we do not yet have a good theory for understanding the ‘sweet spot’ of this bias-variance trade-off.”