Finding Facial Forgery Artifacts with Parts-Based Detectors

Steven Schwarcz
University of Maryland
College Park, MD
schwarcz@umiacs.umd.edu

Rama Chellappa
Johns Hopkins University
Baltimore, MD
rchella4@jhu.edu

Abstract

Manipulated videos, especially those where the identity of an individual has been modified using deep neural networks, are becoming an increasingly relevant threat in the modern day. In this paper, we seek to develop a generalizable, explainable solution to detecting these manipulated videos. To achieve this, we design a series of forgery detection systems that each focus on one individual part of the face. These parts-based detection systems, which can be combined and used together in a single architecture, meet all of our desired criteria - they generalize effectively between datasets and give us valuable insights into what the network is looking at when making its decision. We thus use these detectors to perform detailed empirical analysis on the FaceForensics++, Celeb-DF, and Facebook Deepfake Detection Challenge datasets, examining not just what the detectors find but also collecting and analyzing useful related statistics on the datasets themselves.

1. Introduction

The past few years have seen a sharp rise in the availability of technology for creating and distributing digital forgeries, specifically those where one individual's identity is replaced with that of a another. These forgeries, known as "Deepfakes," represent an important and growing threat to information integrity on the web. A variety of techniques have been proposed to combat the risks inherent in the widespread availability of this technology, many of which are known to perform very well on the datasets on which they are trained, but perform worse when they are applied to different datasets, particularly those which generate fakes using different algorithms.

For this reason, it is vital that video forgery detection systems be able to generalize effectively to novel datasets. Improving performance in this regard requires a strong understanding of both the detectors used and the data itself. To this end, a lot of approaches have been explored [20, 5, 21].

In this paper, we seek to add one more approach to this list, analyzing the problem of digital forgery detection from another angle. Rather than focus on a single architecture for forgery detection, we break the problem down into smaller pieces, dividing the face into several distinct regions of interest and training separate classifiers for each one. Specifically, we divide the face into four main regions - the nose, mouth, eyes and chin - and learn separate detectors for the individual artifacts left behind in each region.

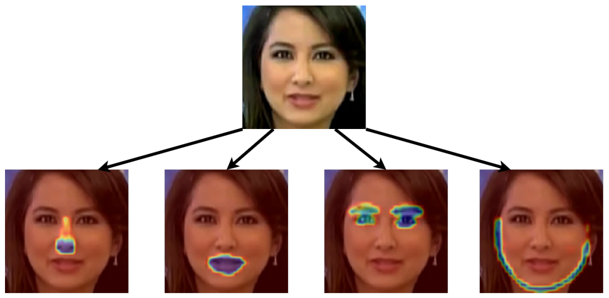
Figure 1. In order to detect manipulated images in an explainable way, we divide the face into four regions - nose, mouth, eyes and chin/jawline - and train separate classifiers for each region. The resultant classifiers generalize well and provide valuable insights into the image manipulation process.

Despite the fact that these classifiers are designed to restrict their attention only to their specific assigned regions, some of them still generalize quite well. This suggests that the regions we are training on may contain powerful clues for forgery detection, powerful enough to let these restricted classifiers out-perform more conventional systems.

These parts-based detectors are also fertile ground for deeper empirical analysis. They are explainable because they make decisions based on limited, easily understood criteria, and we can use their strengths or weaknesses to make inferences about the underlying data they are trained and evaluated on. They thus allow us to gain further insights into the comparative structure of different forgery algorithms.

Using FaceForensics++ [30] as our main dataset, our empirical analysis is broken down as follows: first, we train and evaluate the generalizability of parts-based detectors for each of the four regions of the face that we designate - nose, mouth, eyes and chin/jawline. We then perform the same experiments with a combined parts-detector, which uses multiple branches within the same network to compute parts-based detection separately before recombining. Having trained these parts detectors, we perform comparative analysis of their strengths and weaknesses, looking at both the performance of our detectors and the statistics of altered regions in manipulated images in an effort to develop new insights into the problem of forgery detection. We finish with cross-dataset analysis, comparing generalization performance not just between different algorithms of the FaceForensics++ dataset but also across datasets, preforming transfer experiments on the Celeb-DF [22] and Facebook Deepfake Detection Challenge [10] datasets.

2. Related Works

2.1. Deepfake Algorithms

Computers have long been used to generate forged imagery, but until recently most quality digital forgeries would need to be carefully designed by hand. However, since the invention and widespread use of Generative Adversarial Networks (GANs) [11], it has become considerably easier to use computers to convincingly modify images. In this paper, we are particularly concerned with digital forgery techniques that use deep learning or other modern techniques to transfer the face of one individual to another.

These forgeries, colloquially known as "Deepfakes," can be generated in a variety of ways. Some of these methods predate the invention of GANs, such as [13], which performs swaps by identifying landmarks on faces, and then warping one individual's face onto the other. Still, many Deepfake methods have GANs at their core. FSGAN [24] uses a GAN to reenact one person's pose with another's face, while the NTH model [34] produces Deepfakes by pre-training meta-variables and swapping faces in a few-shot setting. Similarly, techniques like StyleGAN [15] allow Deepfake generation by projecting one image into the latent space of another.

These techniques are especially easy to use because many of them have been made available as software packages. Software such as DeepFaceLab [25] and FaceSwap [1] are easy to download and use, making Deepfakes available to many people outside the research community.

2.2. Forgery/Deepfake Detection

As techniques for generating forged images have improved, so too have techniques for identifying them. Some techniques identified forgery by looking at metadata [14] or other low level artifacts. For instance, techniques such as [26, 33, 27, 21, 20] try to identify forgeries by looking for specific artifacts, e.g. by searching for repeated regions, or stitching or warping artifacts in transferring one face to another body. Patch-based techniques are also popular [35, 28]; for instance, [5] uses patch-based techniques to analyze and improve the generalizability of video forgery detection systems.

[1]https://github.com/deepfakes/faceswap

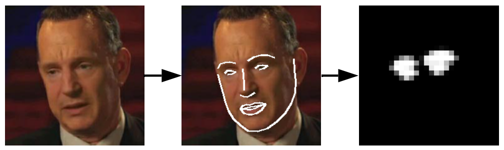
Figure 2. Mask generation pipeline. First parts are detected using dlib, then the masks are created from the convex hulls of the detected regions. Finally, masks are post-processed with morphological dilations and gaussian blur.

Many techniques take advantage of the strength of deep networks directly, by training image classifiers with various architectures to identify forgeries, either through RGB [4, 2, 23, 7] or optical flow [3]. Our work continues in the same vein as many of these other works, taking key intuitions from other methods. In particular we take inspiration from the patch-based approach of [5], extending their patch-based approach to train specific parts-based detectors.

There have also been many new datasets introduced to evaluate forgery detection systems. These range from the smaller but varied Deepfake TIMIT [18] and FaceForensics++ [30] datasets, to the larger, more recent Facebook Deepfake Detection Challenge (DFDC) [10] and Celeb-DF [22] datasets, the last three of which we use in this work.

3. Method

We seek to develop an explainable infrastructure for detecting manipulated images and videos. Below, we explain our approach for building separate part-based classifiers for each region of the face. These classifiers - which must make their decisions by looking at only a limited portion of the image - allow us to get the explainable detectors we desire, which will be critical in performing our analysis.

3.1. Creating the Parts Masks

In order to train parts-specific detectors, we must create individual masks for each region of the face so that they can be used during training. Figure 2 illustrates this straight-forward process. We first acquire facial landmarks using the dlib [16] software library. These detected landmarks are subsequently grouped into four categories: nose, mouth, eyes, and chin/jawline. To convert these landmarks into masks, we simply take the convex hulls of each region separately, except for the chin which we keep as a series of line segments. We then perform 8 iterations of morphological dilation and apply a gaussian filter to create the final high resolution masks. Since these masks will be used to train low-resolution filters within the network described below, the final step in our mask generation pipeline is to downsample the mask to a resolution that matches the maps produced by our network.

3.2. Parts Detection

Single Part-Based Classifier We first describe our detector in the context of a single fixed facial region R, such as the mouth or eyes, which we will refer to as an R-based detector. Since we are only interested in a small portion of the face at a time, we opt to use a patch-based detector with a small receptive field. Thus, as a backbone for our network we use a standard Xception [6] neural network, truncated after the second Xception block and given a single channel of output using a 1×1 convolution layer. This is the same architecture used in [5], and we adopt it in part because the authors of [5] demonstrated that a truncated Xception network generalizes more effectively than the full, deeper network.

Unlike [5], however, we do not directly classify from the output of the truncated Xception network. Instead, in the case of fake images, we use these truncated outputs in order to learn a mask for the region R. Specifically, if x is our input image, $f(x)$ is the output of our truncated neural network, and M_R is the binary mask for region R constructed as described in Section 3.1, then we train an R-based classifier by minimizing standard binary cross-entropy loss:

$$\mathcal{L}_M(x, M_R) = -\sum_{i,j} M_{R,ij} \log(\sigma(f(x)_{ij})) - \sum_{i,j} (1 - M_{R,ij}) \log(1 - \sigma(f(x)_{ij})) \tag{1}$$

where i and j are the horizontal and vertical indices of a given map, and $\sigma(\cdot)$ is the sigmoid function.

If the image being classified is real, then M_R in equation 1 is set to all zeros, teaching the network not to activate when given a real image. Thus, $\mathcal{L}_M$ teaches the network to only look for artifacts of forgery in the specific region targeted by M_R.

Once $f(x)$ has been generated, we produce a final classification label $\hat{y}$ by performing average pooling over $f(x)$ and feeding the result through a classification layer. We denote this operation as $\hat{y} = g(f(x))$. $\hat{y}$ is trained by minimizing binary cross entropy as well, this time over a single value only:

$$\mathcal{L}_C(x) = -y \log(\sigma(\hat{y})) - (1 - y) \log(1 - \sigma(\hat{y})) \tag{2}$$

where y is the ground truth label of the image. The fact that the network must make its prediction solely from the pooled results of the part detector ensures that only the activations of the part detector are used in the final classification.

Thus, for an R-based classifier over a single region, our final loss is:

$$\mathcal{L}_R(x) = \mathcal{L}_C(x) + \lambda \mathcal{L}_M(x, M_R) \tag{3}$$

where λ is a learning weight parameter set to 10 in all of our experiments.

We take a moment here to remark on the small receptive field of our classifier. [5] argued that this smaller receptive field helps with generalization, but it also provides another advantage in our analysis. Namely, it ensures that our network is only looking at the regions we want it to. If the receptive fields were large enough to see the whole image, then it would conceivably be possible that the network might perform classification in an implicit two step process that bypasses looking for artifacts in its assigned region: first identify if the image is real or manipulated, then detect the appropriate part. With a small receptive field, though, the network can make its decision only by looking in the area local to whichever region it is focused on classifying. Thus, in all our following analysis, we can be confident that the classifier is making its decision solely from observing the specific region we have trained it to examine.

Multiple Parts-Based Classifier In addition to performing experiments with single-parts based classifiers, we also perform analysis on detectors that combine multiple regions. Since we do not wish to train multiple networks from scratch in order to perform this recombination step, we seek a way to reuse as much computation as possible without having the networks interfere with each other.

Our proposed solution to this problem is illustrated in Figure 3. When using multiple parts, instead of simply truncating at the second block of the Xception architecture, we add an additional Xception block, identical to the previous one, for each of the four parts of the face we are detecting. None of these additional blocks share weights, thus ensuring that they are each performing their detections separately. We then average the results of all four maps to make our final predictions.

Our final loss for the multiple parts-based classifier is thus:

$$\mathcal{L}_{multi}(x) = \mathcal{L}_C(x) + \lambda \sum_R \mathcal{L}_M(x, M_R) \tag{4}$$

4. Experiments

4.1. Data

The main dataset we use in this paper is FaceForensics++ [30]. This dataset consists of 5000 videos broken

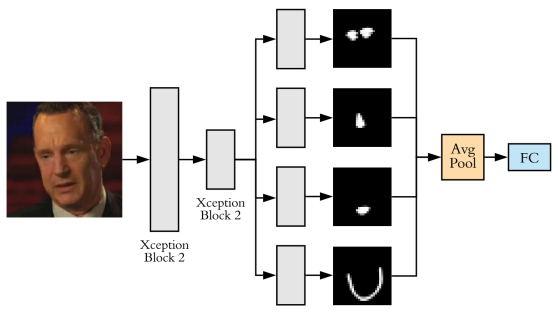

Figure 3. Overview of the architecture used in our experiments. We truncate the Xception architecture after two blocks, then divide the network into separate branches for each part of the face, learning a separate parts mask for each branch. We then concatenate and pool all of the masks into a single value. Finally, we feed this into a fully connected layer to make a prediction.

into three splits: a training split of 3600 videos (720 real and 2880 fake), a validation split of 700 videos (140 real and 560 fake), and a test split of 700 videos (140 real and 560 fake). FaceForensics++ labels the algorithms used to generate each manipulated video, and for each real video fakes are designed using one of the the following algorithms: FaceSwap[2] (FS), Deepfakes[3] (DF), Face2Face [32] (F2F), and NeuralTextures [31] (NT). For the bulk of our experiments, we will train a system on one of these algorithmic splits, and evaluate on the other three.

For experiments measuring the transfer between datasets, there are two other datasets we use. The first of these is the Celeb-DF v2 dataset [22], which contains 6229 videos, of which 518 are in the test split that we use in our experiments. These videos were generated using an improved version of the standard Deepfake synthesis algorithm from 590 different YouTube videos of celebrities.

The second is the Facebook Deepfake Detection Challenge dataset [10], which at the time of this writing is the largest publicly-available Deepfake dataset collected. We will only be performing our evaluation on the publicly available test set of 5000 videos, which were created using the algorithms DFAE [25], MM/NN Face Swap [13], NTH [34], and FSGAN [24].

4.2. Implementation Details

For all of our experiments, we use an Xception backbone [6], pretrained on ImageNet [8]. The networks are trained on 2 Nvidia GeForce RTX 2080 Ti GPUs for 40,000 steps each with Adam [17] using a batch size of 128, a β_1 value of 0.928, weight decay of 0.00005 and an initial learning rate of 0.0001. Every 10000 steps of training, the learning rate is reduced by a factor of 10. All of our code is written in the Tensorflow [1] python library for Deep Learning.

For every video in each dataset, we randomly select 40 frames to use for training. For all images, we use the Dual Shot Face Detector (DSFD) [19] method to detect the faces within each frame. We then take a crop of this face and resize it to 288×288 using nearest neighbor interpolation for all of our Xception training. When training the ResNet50 baselines, we use images of size 224×224 instead. No augmentation is used during training or testing, and all images are compressed as high-quality JPEGs before being fed through the system.

4.3. FaceForensics Generalization

We first explore the effectiveness of each of our parts detectors individually. With this analysis, the purpose is to see what can be achieved with networks trained only to identify fake imagery with respect to a specific location in the image, as described in Section 3.2.

Table 1 shows our performance on these methods. Here, our parts detectors are trained individually on each of the four algorithm-specific splits of the FaceForensics++ dataset, then evaluated on the other three algorithms. For each system we report the Area Under Curve (AUC) of the Receiver-Operating Characteristic (ROC) curve. We use AUC in part because we found that accuracy can be an unstable metric when measuring forgery detection generalization, since during transfer it is not uncommon for a system to be much better at classifying real images than fake images.

We compare our parts detector to 4 other baseline systems. Two of these systems are the standard Xception [6] and ResNet50 [12] architectures, each pretrained on ImageNet [9]. For each of these architectures, we also compare to a truncated variant, where we end the architecture after the second block (which is either a ResNet or Xception block) similar to [5], and compute our prediction as the average of all the logits. This is effectively the same as training the parts-based classifier without $\mathcal{L}_C$, and assigning M_R to be either all zero or all one if the image is real or fake, respectively.

We find from Table 1 that certain parts detectors are superior to others. For instance, the mouth-based detector rarely out-preforms the Xception based detectors, generally performing much worse. This can be seen, for instance, when observing the performance in transferring from Face2Face to Deepfakes or NeuralTextures. As we will see in Section 4.3.1, this may be correlated to different patterns of artifacts in those regions.

The eyes and chin-based detectors, on the other hand, perform considerably better, often out-performing or at least matching the baselines. In some cases, such as with transfer from FaceSwap to Face2Face, or from NeuralTextures

[2]https://github.com/MarekKowalski/FaceSwap
[3]https://github.com/deepfakes/faceswap

Model		DF	F2F	FS	NT		DF	F2F	FS	NT
ResNet50	DF	0.98	0.51	0.48	0.57	FS	0.5	0.54	0.99	**0.49**
ResNet50 Block 2	DF	0.99	0.55	0.47	0.68	FS	**0.54**	0.65	1	0.42
Xception	DF	0.98	0.52	**0.49**	0.61	FS	0.51	0.58	0.99	0.5
Xception Block 2	DF	0.91	0.6	0.43	0.79	FS	0.48	0.62	0.92	0.29
Nose	DF	0.97	**0.63**	0.33	0.81	FS	0.52	0.54	0.99	0.41
Mouth	DF	0.94	0.56	0.48	0.76	FS	0.45	0.63	0.96	0.22
Eyes	DF	0.96	0.6	0.4	0.84	FS	0.5	0.62	0.97	0.38
Chin	DF	0.96	0.59	0.33	**0.85**	FS	0.44	**0.73**	0.99	0.33
Eyes+Chin	DF	0.97	0.58	0.42	0.76	FS	0.48	0.71	0.99	0.41
Combined	DF	0.97	0.62	0.39	0.83	FS	0.52	0.56	0.98	0.32
ResNet50	F2F	0.57	0.98	0.5	0.56	NT	0.56	0.51	0.48	0.94
ResNet50 Block 2	F2F	0.66	0.99	0.54	0.65	NT	0.67	0.52	0.43	0.98
Xception	F2F	0.58	0.98	0.52	0.54	NT	0.59	0.6	0.5	1
Xception Block 2	F2F	0.7	0.94	0.64	0.74	NT	0.69	0.55	0.42	0.98
Nose	F2F	0.65	0.99	0.52	0.63	NT	0.67	0.63	**0.55**	0.98
Mouth	F2F	0.53	0.98	0.65	0.52	NT	0.64	0.63	0.54	0.99
Eyes	F2F	**0.76**	0.98	**0.66**	0.73	NT	0.64	0.51	0.47	0.99
Chin	F2F	0.75	0.95	0.65	0.74	NT	**0.84**	**0.68**	0.38	0.99
Eyes+Chin	F2F	0.56	0.98	0.62	0.48	NT	0.77	0.61	0.47	0.98
Combined	F2F	**0.76**	0.95	0.53	**0.77**	NT	0.66	0.62	0.53	0.98

Table 1. AUC for the ROC curves of the parts-based detectors for each of the four parts of the face, as well as for the combined detector. The second column indicates which split of FaceForensics++ was used to train the model, while the other columns show the performance on each of those splits. Baselines are above the dotted lines and our parts-based detectors are below. Best results for each run are in bold, while models evaluated on the same split they were trained with are in grey. The dataset abbreviations are as follows: FaceSwap (FS), Deepfakes (DF), Face2Face (F2F), and NeuralTextures (NT).

to Deepfakes, we see significant improvements in performance from using the chin detector alone. This strong performance suggests that important artifacts of the forgery process are left behind in these regions.

We observe as well that the nose-based detector falls somewhere in between the others. It often performs poorly, but on certain transfers it performs well, such as when transferring from Deepfakes to Face2Face, or from NeuralTextures to FaceSwap.

The rows marked "Combined" show the performance of our four-branch multi-headed parts detector using all branches. In many cases, we see that the combined detector falls somewhere in between the best and worst parts detectors for each given split.

Since the chin and eyes-based detectors tend to perform better than the other parts, we also include an "Eyes+Chin" detector, which uses two branches instead of the four used by the "Combined" method. Like the "Combined" method, however, we see that combining the detectors does not necessarily greatly improve performance.

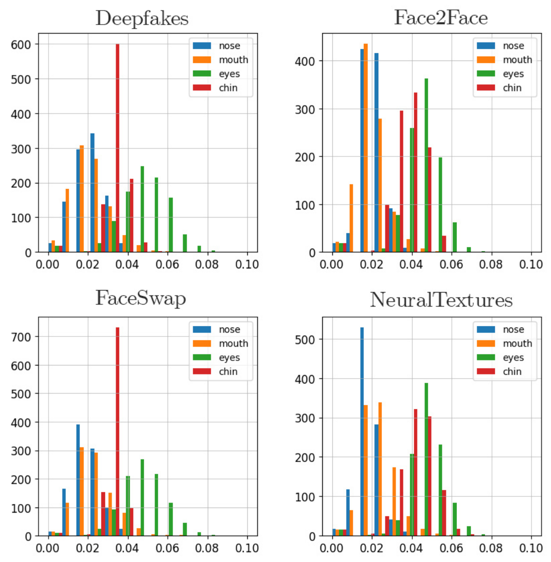

Figure 4. Histograms of absolute pixel difference in each of the four regions of the face used in our analysis, broken down by algorithmic split of the FaceForensics++ dataset.

4.3.1 Dataset Parts Breakdown

In order to better make sense of the performance discussed in Table 1, and in an effort to provide some intuition for certain results, we now perform pixel-level analysis on the FaceForensics++ dataset itself. To do this, we first compute, for one frame per video per algorithm split, the absolute difference between the manipulated image and its real counter-

	Nose	Mouth	Eyes	Chin
Deepfakes	0.0217	0.0218	0.0479	0.0325
Face2Face	0.0218	0.0206	0.0450	0.0371
FaceSwap	0.0206	0.0239	0.0470	0.0313
NeuralTextures	0.0196	0.0239	0.0469	0.0410

Table 2. Average normalized absolute difference between real and manipulated images, broken down by region of the face and manipulation algorithm used. We see that most algorithms have somewhat similar distributions, with the largest changes occurring in the eyes and chin.

Model	DF	F2F	FS	NT
Mean	0.601	0.486	0.643	0.597
Max	0.645	0.529	0.58	0.571
FC	0.573	0.519	0.583	0.555
Ensemble	0.587	0.462	0.576	0.61

Table 3. Different aggregation methods for the composite parts-based model. "Mean" is the method used in the rest of the paper which performs average pooling, "Max" performs max pooling, "FC" adds a fully connected layer, and "Ensemble" runs a separate network for each part, averaging the final logits.

part. We then take the same masks M_R used to generate the ground truth for the R-based part detectors, and multiply them by this computed absolute difference. Specifically, for each part R, given real image x_r and manipulated image x_f, we compute a map D_R as

$$D_{R,ij} = M_{R,ij} \left| x_{r,ij} - x_{f,ij} \right| . \tag{5}$$

In Figure 4 and Table 2, we provide statistical summaries of the maps D_R for each part R and each data split.

We posit that these statistics tell us something about where to find artifacts left over from the manipulation process. Although these summary statistics leave out a lot of important low-level information, some patterns do emerge when comparing certain results from Table 1 to the histograms in Figure 4. For instance, the Nose detector performs well on the transfer from NeuralTextures to FaceSwap (.55 vs a .42 truncated Xception baseline and a .5 Xception baseline), and we all see very similar histogram shapes for pixel differences in the nose region on those two datasets.

On the other hand, where distributions are very different, we see more discrepancy in performance. For instance, between Face2Face and Deepfakes we see very different distributions around the mouth regions, where Face2Face has a higher peak and Deepfakes is flatter. This may reflect the poor transfer performance of the mouth-based detector when trained on Face2Face and evaluated on Deepfakes (.53 vs a .7 baseline for truncated Xception).

Finally, when we look at Table 2 and observe the average absolute differences within regions, we see other distinct patterns. The most obvious pattern is that different parts have similar amounts of changes even between splits - there is far more variation between parts than between algorithms, with eyes changing the most while the nose and mouth change the least.

Beyond that, though, we observe other interesting patterns with respect to the performance of parts detectors. For instance, we see that in the NeuralTextures split the chin region changes far more than in the other splits, and in fact in Table 1 we see that the chin-based detector trained on NeuralTextures considerably out-performs the baseline in the Deepfake and Face2Face dataset splits (.84 and .68 vs .69 and .55, respectively). The one split where the chin-based detector consistently under-performs the baseline is FaceSwap (.38 vs .42), the split with the smallest changes - and therefore presumably the fewest artifacts - in the chin region.

4.3.2 Learned Masks

One advantage to using parts-based classifiers is their explainability. When a parts-based system identifies an image or video as real or fake, one need only look at the predicted masks (described in Section 3.2) for hints at what the system was used to make its prediction. In Figure 5 we provide two examples of these masks for each of the four facial regions. On the left we have the relatively easy scenario of identifying a fake from the same distribution as the network was trained on, in this case the Deepfakes split of FaceForensics++. We see that the system easily identifies all of the relevant regions as fakes.

On the right side of Figure 5, we have a sample of masks generated by parts-based detectors trained on the Deepfakes split and evaluated on Face2Face. Even transferring between splits, we see that the network is still able to find enough artifacts of the forgery process to be effective. For instance, the detector is still able to successfully pick up the small discoloration on the individual's nose, even if the masks are less clean and more likely to miss certain regions, such as the left eye.

4.3.3 Architecture Design

Choice of Aggregation Function In all of the experiments we explored above, we chose to use an average pooling layer for aggregation in order to aggregate the results of our multi-headed parts detector. In Table 3, we explore the choice of other aggregation methods, including a max pooling layer and a trained fully connected layer. These layers are all applied after the individual parts-maps are aggregated separately in the spatial domain using an average pooling layer. We also include one additional method of aggregation labeled "Ensemble", which is attained by training each part detector with a completely different network, and then averaging the logits in an ensemble. Though this

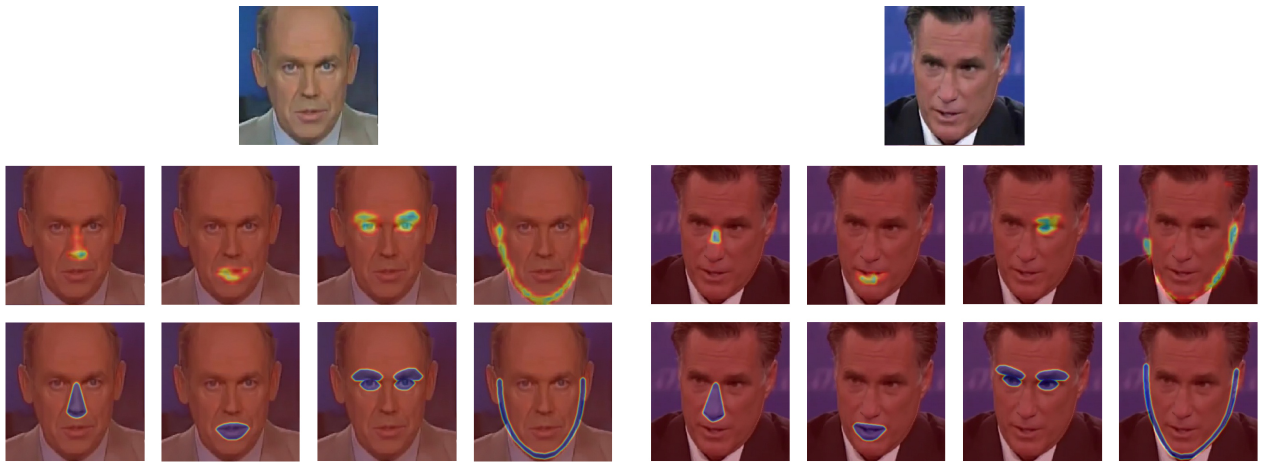

Figure 5. Masks learned by individual part detectors. The top row shows the masks learned by the various parts detectors, while the bottom row shows the ground truth masks constructed as described in Section 3.1 Left: a manipulated image from the Deepfakes FaceForensics++ split, evaluated by parts-based models trained on the Deepfakes split. Right: A manipulated image from the Face2Face split of FaceForensics++, evaluated on models trained on the Deepfakes split.

would of course be less practical since it involves training and running multiple networks, and the performance gains do not make it worthwhile, we do note that the final ensemble has only 1.6 million parameters and uses 14.6 billion FLOPS, which is still less than a full Xception architecture containing over 20 million parameters and using over 15 billion FLOPS.

In order to compute the values in Table 3, we train and evaluate each of our architectures on the same sixteen combinations of splits from the FaceForensics++ dataset that we used in Table 1. We then average all the results for a given split, ignoring results trained and evaluated on the same split. For example, for the Deepfakes split, we average the AUC results on the Deepfakes split for the architecture trained on Face2Face, FaceSwap, and NeuralTextures. This provides us with a good summary statistic for comparing the performance of all three architectural options. The full values obtained by all of these runs are included in the supplementary material.

We find that average and max pooling perform similarly, each out-performing the other in two out of the four categories. The fully-connected layer is generally an inferior form of aggregation, perhaps because it can encode biases for one part or another that do not transfer well between dataset splits.

Number of Additional Blocks Another axis of variation in our architecture is the choice of the number of additional Xception blocks used. Adding these extra blocks is an essential step, because they allow the different part detectors to operate separately. We explore this in Table 4, where we have trained the combined architecture using average-pooling aggregation over 0, 1, and 2 additional Xception blocks. Here, we aggregate values in the same manner as

Model	DF	F2F	FS	NT
0 Blocks	0.528	0.446	0.523	0.56
1 Block	0.601	0.486	0.643	0.597
2 Blocks	0.575	0.497	0.582	0.565

Table 4. Performance of the aggregated parts-based model, trained with different numbers of Xception blocks included after truncation.

we did in Table 3, and once again note that the full experiments can be found in the supplementary material. From this analysis, we find that adding one additional Xception block after truncation is optimal in most cases, with only a small loss with respect to the Face2Face split (from .497 to .486), which is easily outweighed by the larger gains made in the other three splits. The poor performance of using zero blocks in particular confirms our hypothesis that it is necessary to have at least some degree of separate processing for the individual parts-based models, as trying to detect all parts in a single branch will cause the detectors to interfere with one another and performance will degrade.

4.3.4 Failure Case Analysis

We can also find new insights by looking at the failure cases for a given parts-detector. Figure 6 shows some examples of false positives detected by our system for different part-based detectors. More such false positives can also be found in the supplementary material. While we find that these failures look generally how we would expect them to, showing activations in areas where they should not exist, for combined parts detectors the ability to observe these maps when the algorithm fails would help an observer understand which regions of the image caused the network to trigger in-

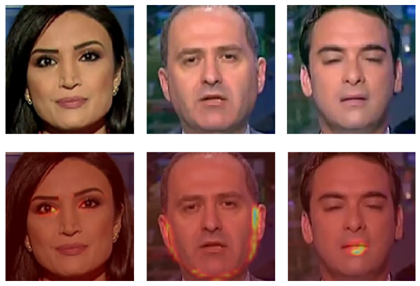
Figure 6. False positives for the eyes (left), chin (middle), and mouth (right) detectors. All three are incorrectly firing on regions within real images.

Model	FF++	Celeb-DF	DFDC
Xception	0.965	0.629	**0.673**
Xception Block 2	0.754	0.622	0.593
Nose	0.909	**0.667**	0.611
Mouth	0.914	0.658	0.617
Eyes	0.847	0.63	0.586
Chin	0.92	0.644	0.618
Average	0.931	0.633	0.627

Table 5. Performance of various systems when trained on the entire FaceForensics++ dataset and evaluated on the Celeb-DF and Facebook DFDC datasets. Our parts-based detectors are below the dotted line. Best results are in bold.

correctly.

4.4. Cross-Dataset Generalization

We have shown that parts-based detectors generalize well between splits of the FaceForensics++ datasets. In this section, we evaluate the generalization performance of parts-based detectors trained on the entirety of the Face-Froensics dataset and evaluated on two other datasets, Celeb-DF and Facebook DFDC. For both of these datasets, our evaluation is only on the publicly-available test splits.

The results of these experiments are shown in Table 5, using the same AUC metric used above. We observe that, with respect to the Celeb-DF dataset, parts-based detectors are generally superior to the Xception baselines, both truncated and otherwise. This shows that our method remains effective relative to other techniques even as the difference between training and evaluation grows.

However, for the DFDC dataset, we find that parts detectors are not sufficient to achieve state-of-the-art performance over the Xception baseline. However, we still observe that all parts-based methods out-perform the truncated Xception baseline, which itself outperformed the Xception baseline in the vast majority of other transfer tasks. Overall, this indicates that some of the data in the DFDC dataset might require a more global approach in order to perform proper detection, whereas more local approaches with smaller receptive fields, such as truncated Xception and parts-based classifiers, simply do not have sufficient receptive fields. This also opens up the possibility of future work into parts-based detectors that use much larger receptive fields, perhaps with variants of U-Net [29] or similar architectures.

5. Conclusion

In this work, we have shown that it is possible to use neural networks trained only to look in specific regions of the face to improve generalization performance between video manipulation algorithms. This suggests that these individual parts of the face may in some cases actually be more representative than the rest of the image as a whole, since restricting the classifier's attention to only these parts improves generalization. Having observed this, we used these parts-based detectors to perform extensive empirical analysis, analyzing which parts are most discriminative between datasets and examining the underlying distributions of our data.

6. Acknowledgements

This paper is based upon work supported, in whole or in part, by the Office of the Under Secretary of Defense for Intelligence and Security. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the University of Maryland, the UMD Applied Research Laboratory for Intelligence and Security (ARLIS), Office of the Under Secretary of Defense for Intelligence and Security, or the U.S. Government.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. Mesonet: a compact facial video forgery detection network. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, 2018.

[3] I. Amerini, L. Galteri, R. Caldelli, and A. Del Bimbo. Deepfake video detection through optical flow based cnn. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1205–1207, 2019.

[4] Belhassen Bayar and Matthew C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IHamp;MMSec '16, page 5–10, New York, NY, USA, 2016. Association for Computing Machinery.

[5] Lucy Chai, David Bau, Ser-Nam Lim, and Phillip Isola. What makes fake images detectable? Understanding properties that generalize. aug 2020.

[6] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[7] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, 2017.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge dataset, 2020.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] Dong Huang and Fernando De La Torre. Facial action transfer with personalized bilinear regression. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 144–158, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[14] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A. Efros. Fighting fake news: Image splice detection via learned self-consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[15] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.

[16] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[18] Pavel Korshunov and Sébastien Marcel. Deepfakes: a new threat to face recognition? assessment and detection, 12 2018.

[19] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang. Dsfd: Dual shot face detector. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5055–5064, 2019.

[20] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and B. Guo. Face x-ray for more general face forgery detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5000–5009, 2020.

[21] Yuezun Li and S. Lyu. Exposing deepfake videos by detecting face warping artifacts. In *CVPR Workshops*, 2019.

[22] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3204–3213, 2020.

[23] Huaxiao Mo, Bolin Chen, and Weiqi Luo. Fake faces identification via convolutional neural network. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, IHamp;MMSec '18, page 43–47, New York, NY, USA, 2018. Association for Computing Machinery.

[24] Y. Nirkin, Y. Keller, and T. Hassner. Fsgan: Subject agnostic face swapping and reenactment. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7183–7192, 2019.

[25] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang. Deepfacelab: A simple, flexible and extensible face swapping framework, 2020.

[26] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, 2005.

[27] A. C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Transactions on Signal Processing*, 53(10):3948–3959, 2005.

[28] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen. Distinguishing computer graphics from natural images using convolution neural networks. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2017.

[29] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[30] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to Detect Manipulated Facial Images. jan 2019.

[31] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), July 2019.

[32] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. *Commun. ACM*, 62(1):96–104, Dec. 2018.

[33] Weihong Wang and Hany Farid. Exposing digital forgeries in video by detecting duplication. In *Proceedings of the 9th Workshop on Multimedia amp; Security*, MMamp;Sec '07, page 35–42, New York, NY, USA, 2007. Association for Computing Machinery.

[34] E. Zakharov, A. Shysheya, E. Burkov, and V. Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9458–9467, 2019.

[35] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1831–1839, 2017.