Supplementary Material for
Lips Don’t Lie: A Generalisable and Robust Approach to Face Forgery Detection

Alexandros Haliassos\textsuperscript{1,†}, Konstantinos Vougioukas\textsuperscript{1}, Stavros Petridis\textsuperscript{1}, Maja Pantic\textsuperscript{1,2}

\textsuperscript{1}Imperial College London \hspace{1cm} \textsuperscript{2}Facebook London
\{alexandros.haliassos14,k.vougioukas,stavros.petridis04,m.pantic\}@imperial.ac.uk

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{ms-tcn.png}
\caption{Diagram of an MS-TCN block. Diagram of a single block in the multi-scale temporal convolutional network (MS-TCN) that we employ. The stride is 1 for all convolutions. The full temporal network we use consists of 4 such blocks. The dilation rate in the first block is equal to 1, and each subsequent block’s dilation rate is $2 \times$ the previous one’s.}
\end{figure}

1. More Implementation Details

1.1. Architecture details

A single block of the multi-scale temporal convolutional network [18] (MS-TCN) used in our architecture is shown in Figure 1. The abbreviations are defined as follows:

- Conv1D($x$, $y$): 1-D convolutional layer with $x$ output channels and kernel size $y$. All use “same” padding and stride of 1.


- PReLU: Parametric ReLU activation [8] with a separate learnable parameter for each input channel.

- Dropout($x$): Dropout layer [22] with probability $x$.

1.2. Datasets

FaceForensics++ (FF++) [20]. We download the dataset from the official webpage\textsuperscript{1}. We use the provided training/validation/test splits.

FaceShifter [14]. We download the FaceShifter samples (at c23 compression) from the same place as FF++, since these have been recently added to the webpage. Note that when we refer to FF++, we are referring to the version described in the FF++ paper, i.e., containing the 4 manipulation methods without FaceShifter. We use the same training/validation/test splits as in FF++.\textsuperscript{1}

DeeperForensics [12]. We download the dataset from the official webpage\textsuperscript{2}. We use the same training/validation/test splits as in FF++.\textsuperscript{2}

Celeb-DF-v2 [17]. We download the dataset from the official webpage\textsuperscript{3}. We use the test set, which consists of 518 videos.

DFDC [7]. We download the test set of the full DFDC dataset from the official webpage\textsuperscript{4}. Some videos feature more than one person. To remove ambiguities in preprocessing, we only use single-person videos. Further, many videos have been filmed in extreme conditions (lighting, poses, etc) and/or have been post-processed with aggressive corruptions. As such, we only use videos for which the face and landmark detectors did not fail.

\textsuperscript{1}\url{https://github.com/ondyari/FaceForensics}

\textsuperscript{2}\url{https://github.com/EndlessSora/DeeperForensics-1.0}

\textsuperscript{3}\url{https://github.com/yuezunli/celeb-deepfakeforensics}

\textsuperscript{4}\url{https://ai.facebook.com/datasets/dfdc}
1.3. Preprocessing

We use RetinaFace [5] to detect a face for each frame in the videos. As in [20], we only extract the largest face and use an enlarged crop, $1.3 \times$ the tight crop produced by the face detector. To crop the mouths for LipForensics, we compute 68 facial landmarks using FAN [2]. The landmarks are smoothed over 12 frames to account for motion jitter, and each frame is affine warped to the mean face via five landmarks (around the eyes and nose). The mouth is cropped in each frame by resizing the image and then extracting a fixed $96 \times 96$ region centred around the mean mouth landmark. We note that alignment is performed to remove translation, scale, and rotation variations; it does not affect the way the mouth moves.

1.4. Baselines

For the baselines we consider, we provide details on our implementations that are not given in the main text. Unless stated otherwise, Adam [13] optimisation is used with a learning rate of $2 \times 10^{-4}$ and batch size of 32.

Face X-ray [15]. To generate the blended images for training, we use provided code. In addition to the random mask deformation and colour correction operations described in the paper, the following augmentations are applied as per the code: random horizontal flipping, JPEG compression (with quality $\sim$ Uniform\{30, 31, …, 100\}), and pixelation (downscaling image by a factor $\sim$ Uniform\{0.2, 1\}), each with probability 0.5. For fair comparison with the other methods, we also train with samples from FaceForensics++ (FF++). Following the code, each image sampled during training is either a real FF++ frame or a fake sample, with probability 0.5. In turn, each fake sample is either a blended image or an FF++ fake frame, again with probability 0.5. The cropped faces are resized to $317 \times 317$ and then centre cropped to $256 \times 256$. The scaling factor, $\lambda$, corresponding to the segmentation loss is set to 100, as in the paper.

---

5https://github.com/biubug6/Pytorch_Retinaface

6https://github.com/1adrianb/face-alignment

7https://github.com/AlgoHunt/Face-Xray
CNN-aug [25]. We use the official code\(^8\). The cropped faces are resized to \(256 \times 256\). We use JPEG compression (with quality \(\sim\) Uniform\([60, 61, \ldots, 100]\)) and Gaussian blurring with standard deviation \(\sim\) Uniform\([0, 3]\), both with probability 0.1. We also use horizontal flipping with probability 0.5.

**Patch-based** [3]. We use the official code\(^9\). We train the model ourselves, since no provided pretrained model was trained on full FF++. The faces are aligned by affine warping them to the mean face and then resized to \(299 \times 299\). We use horizontal flipping with probability 0.5. As recommended in [21], we first train only the DenseNet-161 with learning rate \(1 \times 10^{-3}\) is used, as suggested in the paper.

**Xception** [20]. We use the official code\(^10\). The cropped faces are resized to \(299 \times 299\). We use horizontal flipping with probability 0.5.

**CNN-GRU** [21]. The cropped faces are resized to \(224 \times 224\). We use horizontal flipping with probability 0.5. As recommended in [21], we first train only the DenseNet-161 [10] (by adding a linear classifier). We then append a single-layer, bi-directional GRU [4] with hidden size 128 and train the whole network end-to-end.

**Multi-task** [19]. We use the official code\(^11\) and follow the paper recommendations for all hyperparameters. We use the “deep” version of the model. We train it ourselves since the provided pretrained model has only been trained on a subset of FF++. The cropped faces are resized to \(256 \times 256\). We use horizontal flipping with probability 0.5. Adam [13] with learning rate \(1 \times 10^{-3}\) is used, as suggested in the paper.

**DSP-FWA** [16]. We use the official code\(^12\) and pretrained model (on self-collected real faces), which uses a dual spatial pyramid approach. Each face is aligned and extracted at 10 different scales. They are all resized to \(224 \times 224\).

**R(2+1)D-18** [24] and **ip-CSN-152** [23]. We use the official code\(^13\) and finetune pretrained models. We perform the same preprocessing as for our LipForensics approach, except that RGB frames are used rather than grayscale, since the pretrained tasks use colour frames.

**SE-ResNet50** [9]. We use the ArcFace [6] code\(^14\) and finetune the backbone of the model pretrained on face recognition datasets. The cropped faces are resized to \(112 \times 112\), since this is the size used during pretraining. We use horizontal flipping with probability 0.5.

---

\(^8\)https://github.com/peterwang512/CNNDetection

\(^9\)https://github.com/chail/patch-forensics

\(^10\)https://github.com/ondyari/FaceForensics

\(^11\)https://github.com/niit-yamagishilab/ClassNSeg

\(^12\)https://github.com/yuezunli/DSP-FWA

\(^13\)https://github.com/facebookresearch/VMZ

\(^14\)https://github.com/TreB1eN/InsightFace_PyTorch

---

\(^15\)https://github.com/EndlessSora/DeeperForensics-1.0/tree/master/perturbation

### Table 1. Full face crops versus mouth crops

<table>
<thead>
<tr>
<th>Input type</th>
<th>Pretrain</th>
<th>Finetune</th>
<th>FSh</th>
<th>DFo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full face</td>
<td>none</td>
<td>whole</td>
<td>68.2</td>
<td>67.1</td>
</tr>
<tr>
<td>Full face</td>
<td>LRW</td>
<td>whole</td>
<td>82.9</td>
<td>85.2</td>
</tr>
<tr>
<td>Full face</td>
<td>LRW</td>
<td>temporal</td>
<td>84.3</td>
<td>90.0</td>
</tr>
<tr>
<td>Mouth</td>
<td>none</td>
<td>whole</td>
<td>62.5</td>
<td>61.4</td>
</tr>
<tr>
<td>Mouth</td>
<td>LRW</td>
<td>whole</td>
<td>83.2</td>
<td>84.6</td>
</tr>
<tr>
<td>Mouth</td>
<td>LRW</td>
<td>temporal</td>
<td><strong>87.5</strong></td>
<td><strong>90.4</strong></td>
</tr>
</tbody>
</table>

1.5. Robustness experiments

To apply the corruptions in our robustness experiments, we use the DeeperForensics code\(^15\). All considered corruptions at all severity levels are depicted in Figure 2.

### 2. Full Face Versus Mouth Crops

In the main text, we always use mouth crops for LipForensics. Here, we increase the crop from \(88 \times 88\) to \(112 \times 112\) (after random cropping) to also include the whole nose and eyes in the input. We pretrain a new model on LRW using this input. As shown in Table 1, when training from scratch, using full faces rather than mouth crops yields better generalisation to FaceShifter and DeeperForensics, but when using lipreading pretraining, mouth crops perform better. For both types of input, lipreading pretraining improves accuracy significantly.

### 3. Qualitative Analysis

#### 3.1. High-level mouth inconsistencies

Our approach targets high-level temporal inconsistencies related to the mouth region. We show examples of such anomalies in Figure 3. Notice that in some cases, the mouth does not sufficiently close, as noted in [1]. In other cases, subtle temporal inconsistencies in the shape of the mouth or its interior (e.g., teeth) are present.

#### 3.2. Failure cases

Examples of failure cases are given in Figure 4. In general, we noticed that many of the failure cases involve rapid head movements, poses that are uncommon in the training set (FF++), or very limited mouth movements.

#### 3.3. Occlusion sensitivity

We show more visualisation examples using the occlusion sensitivity approach discussed in the main text. This...
approach was introduced in [26]. It relies on systematically covering up different portions of the frames with a grey block and measuring the effect on the predictions of the model. We found that a block size of $40 \times 40 \times t$, where $t$ is the number of frames in the video, is suitable, as it is large enough to sufficiently occlude the mouth region. After each iteration, the block is displaced by 1 pixel, and the probability of predicting the correct class is recorded for each occluded pixel. Following this process, a heatmap can be created by averaging the probabilities at each pixel location. The heatmaps are finally normalised and overlaid on the first frame of the video.

We show visualisation examples for Xception [20] (see Figure 5) as well as for training the spatiotemporal network from scratch (see Figure 6) and LipForensics (see Figure 7). As mentioned in the main text, unlike Xception, LipForensics consistently relies on the mouth region. Interestingly, without lipreading pretraining, the network often seems to rely on regions other than the mouth (such as the nose), despite the (conservative) mouth crop. This is more the case for the face swapping methods, Deepfakes and FaceSwap.
Figure 3. **Examples of semantically high-level inconsistencies around the mouth region.** Rows 1-2 show mouths that do not sufficiently close; rows 3-7 show mouths with limited mouth movements but which still exhibit anomalous behaviour; rows 8-9 show inconsistencies in the teeth and lip shape; rows 10-11 show temporal irregularities in mouth shape (e.g., see frames 3 and 4 in row 10 and frame 3 in row 11). Subtle anomalies are more readily observed in video form.
Figure 4. **Failure cases.** Top two rows are real videos predicted as fake and bottom two are fake videos predicted as real.

Figure 5. **Visualisation examples for Xception.** We show examples for Deepfakes (DF), FaceSwap (FS), Face2Face (F2F), and Neural-Textures (NT).
Figure 6. **Visualisation examples for spatiotemporal network without lipreading pretraining.** We show examples for Deepfakes (DF), FaceSwap (FS), Face2Face (F2F), and NeuralTextures (NT).

Figure 7. **Visualisation examples for LipForensics.** We show examples for Deepfakes (DF), FaceSwap (FS), Face2Face (F2F), and NeuralTextures (NT).
References


