Towards Unconstrained Audio Splicing Detection and Localization with Neural Networks

Denise Moussa[1,2,3][0000−0002−1390−9198], Germans Hirsch[2,3], and Christian Riess[2][0000−0002−5556−5338]

[1] Federal Criminal Police Office (BKA), Germany
[2] Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
[3] Both authors contributed equally to this work.
{denise.moussa, christian.riess}@fau.de

Abstract. Freely available and easy-to-use audio editing tools make it straightforward to perform audio splicing. Convincing forgeries can be created by combining various speech samples from the same person. Detection of such splices is important both in the public sector when considering misinformation, and in a legal context to verify the integrity of evidence. Unfortunately, most existing detection algorithms for audio splicing use handcrafted features and make specific assumptions. However, criminal investigators are often faced with audio samples from unconstrained sources with unknown characteristics, which raises the need for more generally applicable methods.
With this work, we aim to take a first step towards unconstrained audio splicing detection to address this need. We simulate various attack scenarios in the form of post-processing operations that may disguise splicing. We propose a Transformer sequence-to-sequence (seq2seq) network for splicing detection and localization. Our extensive evaluation shows that the proposed method outperforms existing dedicated approaches for splicing detection [3, 10] as well as the general-purpose networks EfficientNet [28] and RegNet [25]. Our source code is available at: https://www.cs1.tf.fau.de/research/multimedia-security/code

Keywords: Audio Splicing Detection · Forensics · Deep Learning

1 Introduction

With steadily improving technical methods, it became increasingly easier to convincingly manipulate multimedia content. In the era of social media platforms, where huge amounts of images, audio snippets and videos are distributed, well-crafted pieces of maliciously manipulated content may quickly spread over the internet. Forged multimedia content also becomes increasingly relevant in a legal context, such as criminal investigations. Hereby, multimedia material may serve as clue or evidence. Hence, it is of increasing importance to research and to deploy methods for ensuring the integrity and authenticity of such data.

Plausible multimedia forgeries can be created with several deep learning (DL) methods. Examples are approaches for changing the attributes of a given image

of a human face [32], for swapping the face of persons in photographs [21], or for face reenactment [21, 29]. Another impressive demonstration was shown recently on video synthesis: given a speech recording and image of an individual's face, lip synthesis can even create a video of the individual that gives the speech [4]. To create fake audio, voice conversion and speech synthesis have been proposed. Voice conversion transforms the speech of one person to sound like the voice of another person [9]. Speech synthesis generates fully synthetic audio in the voice of a specific individual [11].

In this work, we analyze audio splicing, which describes the processes of inserting, deleting or concatenating audio signals. This tampering technique is arguably one of the easiest to perform, as it can also be done by laypeople using freely available audio editing tools like Audacity [1] or Oceanaudio [22]. Despite the simplicity of creation, audio splicing can pose a serious threat. In some situations, even small changes can already alter the semantics of statements in a speech. For example, removing the negation particle 'not' may flip a negated statement into a positive statement.

Many existing works use hand-crafted features to extract specific characteristics for audio splicing detection or localization (Sec. 2.1). One weakness of hand-crafted features is that they are typically only applicable within relatively narrow bounds around their designed purpose. For example, methods intended for finding differences in recording devices do not indicate splicing if the same device is employed. As a second example, classification of recording environments is unable to detect splicing from static surroundings. However, DL techniques make it feasible to learn features that fit more complex data distributions, and thereby to overcome the constraints of manual feature selection. Surprisingly, DL methods have so far barely been explored for audio splicing detection.

With this work, we aim at closing this gap and propose a Transformer [31] seq2seq architecture to make a first step towards unconstrained audio splicing detection and localization. We analyze different attack scenarios in the form of post-processing operations that may be applied to disguise splicing. This includes MP3 and AMR-NB single and multi compression, additive synthetic and real noise. Our forgery settings also cover audio compositions with samples from different environments, same environments and – particularly difficult to differentiate – anechoic environments. We compare to a recent method that uses handcrafted features [3], and to the neural network by Jadhav *et al.* [10], which to our knowledge is the only other end-to-end approach to forensic audio splicing detection. Our method also outperforms the state-of-the-art general-purpose models EfficientNet [28] and RegNet [25]. Our main contributions are:

- We present a robust end-to-end trainable seq2seq model for audio splicing detection and localization.
- For evaluation, we propose a challenging data set that covers a large variety of forgery scenarios for the task of unconstrained audio splicing detection and localization.

2 Related Work

We distinguish two types of methods for audio splicing detection and localization. First, we review methods that manually extract features in Sec. 2.1. Second, we review DL methods in Sec. 2.2.

2.1 Methods with Manual Feature Selection

Most related works on audio splicing detection rely on specific handcrafted features. As such, the feature response is inherently explainable, but the methods can typically only be used in very specific application scenarios. For example, Yang *et al.* focus specifically on MP3 encoded audio [34]. They localize splicing in the signals by identifying inconsistencies in the offsets of the encoded audio frames. Cooper *et al.* target uncompressed audio, where they search discontinuities in the high frequencies of the audio waveform [5].

Some other approaches detect splicing by analyzing noise levels in the audio signal. For instance, Pan *et al.* compute global and local noise levels of audio data and identify abnormal changes in the noise level [24]. Meng *et al.* adopt a similar approach and compute local noise levels w.r.t. each syllable in the speech signal [19]. Different from those works, Yan *et al.* [33] recently targeted composite audio with signals from different sources but with similar or equal signal-to-noise ratio (SNR). This work locates splicing points from the variances of Mel frequency cepstral coefficients (MFCCs).

Cuccovillo *et al.* [6] perform splicing detection based on microphone classification. They exploit the fact that source files from different devices exhibit characteristic traces in the signal.

Several works analyze the electric network frequency (ENF). Esquef *et al.* use ENF analysis to exclusively target splices in silent (non-voice-active) positions in the signal [8]. However, ENF traces are weak and may be occluded by high noise. Hence, Lin *et al.* apply a spectral phase analysis instead which exhibits better robustness towards noise corruptions [13]. Lin *et al.* also present a second approach to increase the robustness to noise [14]. They propose to amplify abnormal changes via wavelet-filtering the ENF signal, and then extract autoregressive coefficients for tampering detection.

Another line of work models acoustic environmental signatures for splicing detection. Zhao *et al.* compute features from the acoustic channel impulse response and ambient noise to model such an acoustic environmental signature [37, 38]. Rouniyar *et al.* refine the feature selection and increase performance by using both acoustic channel response and dynamic and static logarithmic spectral characteristics to identify and locate splicing [26].

Recently, Capoferri *et al.* proposed the reverberation time (RT) as single forensic trace to detect splicing [3]. The RT is assumed to differ in different surroundings. Hence, changes in the RT across different temporal windows of the audio signal are then used to localize splicing.

A shared drawback of all these approaches is that they are restricted by their specific assumptions. This includes specific audio compression formats or the

absence of compression [34, 5], differing recording devices in the forged audio [6] and changing recording environments, like changing noise [24, 19, 33], changing acoustic impulse [37, 38, 3] or indicative ENF patterns [8, 13, 14].

2.2 Deep Learning Methods

DL methods were successfully applied to various tasks in audio forensics, including double compression detection [16], audio recapture detection [15], speech presentation attack detection [12], and fake speech detection [35]. Surprisingly, the detection of audio splicing received little attention so far. Mao *et al.* [17] target audio tampering (including splicing) with a convolutional neural network (CNN) classifier for detection but without localization. The CNN does not directly operate on the audio samples, but instead on pre-defined ENF features. Zhang *et al.* [36] use an encoder-decoder architecture based on VGG-16 [27] to learn a segmentation mask for spliced audio samples. This method can be seen as a very preliminary splicing detector whose design is dependent on quite restrictive assumptions. Most notably, the architecture only permits the detection of splicings from snippets with 1s duration, where at the same time each snippet is a random crop from a different speaker.

Jadhav *et al.* [10] use the short-term Fourier transform of spliced audio signals directly as input to a shallow CNN architecture. They report that their method is robust under added white Gaussian noise and dynamic range compression. However, this work also does not consider the practically highly relevant case of splices from sources of the same speaker. Additionally, the evaluation of the method is somewhat limited to a relatively small, non-diverse test set.

In this work, we first explore the task of broadly applicable, robust audio splicing detection and localization. We propose a seq2seq Transformer [31] that operates on various representations of audio signals. Transformers excel in various natural language processing tasks [23] and efficiently exploit sequential context information which is also present in audio data. In addition, seq2seq methods have the specific benefit that, differently from CNN based methods [10, 36], they can very naturally process sequences of arbitrary input/output lengths.

Contrary to previous approaches [10, 36], we validate our method on a challenging, diverse dataset. We include single and multiple splices, as well as various post-processing operations that potentially disguise the splicing, such as multiple MP3 and AMR-NB compressions and added synthetic and real noise. Moreover, we consider forgeries that are created from audios that are recorded in different environments, as well as the more difficult case of identical environments. Additionally, we investigate the very challenging intersplicing scenario, where samples are assumed to be relatively anechoic and stem from the same recording in a static surrounding. With this diversity we also aim at mitigating the application constraints of previous methods that use handcrafted features (Sec. 2.1).

3 Methods

This section consists of four parts. In Sec. 3.1 we describe our setup for data generation, in Sec. 3.2 we formally define our task. Sec. 3.3 summarizes the Transformer design [31] and Sec. 3.4 describes our proposed architecture.

3.1 Data Generation Pipeline

The data for our experiments is generated with base data from the ACE [7] and the Hi-Fi TTS (TTS) [2] datasets. For the ACE dataset, we use the same data split as Capoferri *et al.* [3], thus 65 speech samples between 1.28 s and 97 s from 5 female and 9 male speakers. For the TTS dataset, we consider all 6 female and 4 male speakers and use the 10 longest utterances per person reading different books. The audio signals vary between 7 s and 20 s. We separate speakers in training, validation and test pools. For ACE, we take 10 speakers for training, 2 for validation, and reserve one female and one male speaker for test. For experiments where we train on ACE, we take all speakers of TTS for testing. When training on the latter, we split it into 7 speakers for training, and for validation and test we use 2 speakers (male and female) each.

For our experiments, we generate training, validation and test inputs from these disjoint speaker pools using our pipeline shown in Fig. 1. The pipeline operates as follows. First, we choose $N \in [1, 5]$ audio signals $\mathbf{a}_1(, \ldots, \mathbf{a}_N)$ from the same speaker randomly and allow multiple choices of the same sample. Each $\mathbf{a}_i$ is convolved with a room impulse response (RIR) randomly sampled from a set of 7 synthetic and 7 real RIRs (step 1) to simulate recordings from environments with various echoic characteristics. Here, we use the same RIRs as Capoferri *et al.* [3] and allow recordings to stem from equal environments. Note that we skip this step in one experiment without RIR cue (Sec. 4.3). We omit any other individual processing per sample $\mathbf{a}_i$ (*e.g.*, added noise or compression) to avoid that our model is accidentally biased to such overly specific features. In the second step, we compute all silent, *i.e.* non-voice-active, positions for each signal with a voice activation detector. We sample two random silent positions from each $\mathbf{a}_i$ and cut the sub-sequence between these points, which we denote as $\tilde{\mathbf{a}}_i$. The spliced sample $\tilde{\mathbf{a}}$ is then the concatenation of all $\tilde{\mathbf{a}}_1, \ldots, \tilde{\mathbf{a}}_N$ or equal to $\tilde{\mathbf{a}}_1$ if $n = 1$, *i.e.*, no splicing was sampled (step 3). Optionally, post-processing is applied to $\tilde{\mathbf{a}}$ which may mask splicing points. To this end, additive noise can first be added with a specified SNR (step 4). Then, compression can be applied once or multiple times with given format and strength (step 5). The length of the resulting audio samples varies between 3 s and 45 s. All specific post-processing parameters for each experiment are stated in Sec. 4.

As a last step, three representations of the resulting signal are computed via torchaudio [30] (step 6). We include the Mel spectrogram and MFCCs as two established features in the speech processing domain and additionally add spectral centroid features which describe the brightness of a sound [18]. Incorporating MFCCs and spectral centroid features additionally to the Mel spectrogram empirically showed to improve the performance in complex settings (Sec. 4). For

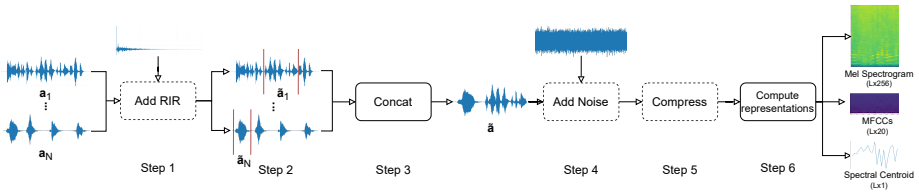


Fig. 1: Adaptive pipeline for generating spliced audio samples. Reverberation of environments is simulated prior to splicing, while additive noise and compression are post-processing operations that may disguise splicing points. Dotted lines indicate optional operations, allowing for adaptable splicing forgery scenarios.

computing the spectral features, we use a window size of 500 ms and a sample rate of 16 kHz. For the Mel spectrogram we additionally choose 256 Mel filter banks and then directly compute the MFCCs with `n_mfcc` = 20 from the result. All other parameters are used in their default settings.

3.2 Task Formulation

We model audio splicing detection/localization as a seq2seq task. The Mel Spectrogram of audio signals is our input sequence $\mathbf{S} \in [-1,1]^{w \times H}$ with fixed H, *i.e.*, fixed frequency resolution and variable width $w \in \mathbb{N}$ depending on the signal length in the time domain. We process $\mathbf{S}$ as a series of slices, $\mathbf{s}_i \in [-1,1]^H$, *i.e.*, column-by-column, which yields the input series $\tilde{\mathbf{S}} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_w]$. Our specific task is to translate $\tilde{\mathbf{S}}$ to a series of splicing points $\hat{P} = [\hat{p}_1, \hat{p}_2, ..., \hat{p}_L]$ of variable length L. Each splicing location $\hat{p}_j \in \mathcal{V}$ is provided in seconds from a vocabulary $\mathcal{V}$ of all valid splicing locations.

3.3 Transformers

Transformers [31] are neural network (NN) architectures with a scaled dot-product attention mechanism as core operation. They aim at grasping global dependencies between two sequences $\mathbf{S}_i$, $\mathbf{S}_j$. An important component is self-attention, where $\mathbf{S}_i = \mathbf{S}_j$. The attention function maps a matrix triple $(\mathbf{Q}, \mathbf{V}, \mathbf{K})$ to an output, with $\mathbf{Q}$,$\mathbf{V}$,$\mathbf{K}$ being a set of queries $\mathbf{Q}$, values $\mathbf{V}$, and keys $\mathbf{K}$. It is defined as

$$\mathrm{Att}(\mathbf{Q}, \mathbf{V}, \mathbf{K}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_K}}\mathbf{V}\right), \tag{1}$$

where d_K is the dimension of the queries/keys that normalizes the product. $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are obtained by projecting input sequences with learnable weight matrices $\mathbf{W}^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d_m \times d_k}$ and $\mathbf{W}^V \in \mathbb{R}^{d_m \times d_v}$, where d_m is the model dimension, *i.e.* the output dimension of the model's layers, and d_k, d_v are the keys' and values' dimensions, respectively. Typically, Transformers learn h

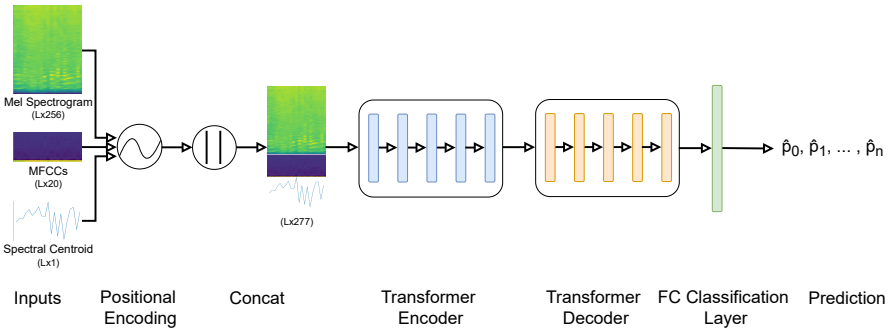


Fig. 2: Proposed network architecture. Different audio representations of length L (dependent on the audio sample length) are concatenated and processed by one encoder, which proved to be superior to separate encoders per representation. The encoded results are fed to the decoder and projected to the vocabulary space to yield the final predicted splicing points.

representations $(\mathbf{Q}, \mathbf{V}, \mathbf{K})$ and compute attention in parallel over those h triples. This is called multi-head attention and formally given by

$$\text{MultiHead}(\mathbf{Q}, \mathbf{V}, \mathbf{K}) = [\text{head}_1||...||\text{head}_h]\mathbf{W}^O, \tag{2}$$

where $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_m}$ and head_i (for $1 \leq i \leq h$) is

$$\text{head}_i = \text{Att}(\mathbf{Q}_i, \mathbf{V}_i, \mathbf{K}_i) \ . \tag{3}$$

A Transformer for seq2seq tasks consists of a transformer encoder and a transformer decoder network. The input sequence is processed by the encoder, which outputs the encoded result to the decoder. The latter calculates the prediction from that encoder memory and the sequence elements from all previous time steps. Both the encoder and the decoder network consist of several layers. Each encoder layer implements self-attention and fully-connected (FC) sub-layers. Each sub-layer includes layer-normalization as well as a residual connection around itself. The decoder is constructed similarly. However, additionally to self-attention, it also computes the encoder-decoder attention, where $\mathbf{K}$, $\mathbf{V}$ stem from the encoder output, while $\mathbf{Q}$ stems from the decoder. For an in-depth description of the Transformer architecture, we refer to Vaswani *et al.* [31].

3.4 Proposed Network Architecture

Our proposed network architecture is depicted in Fig. 2. It operates on three input representations, the Mel spectrogram, MFCCs and spectral centroid. All inputs are normalized to range $[-1, 1]$ to support training stability. They are then subjected to positional encoding [31], concatenated and fed to the encoder (blue). Fusing the inputs and encoding them empirically showed to perform better than

processing each input representation with a separate encoder and fusing the outputs. The encoder and decoder (orange) both consist of 5 layers with 8 heads. The model dimension $d_m = d_k = d_v$ is of size 256 and the dimension of each layer's FC sublayer is 512. The last FC layer (green) projects the output of the decoder to the vocabulary size $|\mathcal{V}|$. Our vocabulary $\mathcal{V} = \{\circ\} \cup \{0.5, ..., 44.5\} \cup \{<\text{pad}>, <\text{bos}>, <\text{eos}>\}$ consists of 93 items representing no splicing ($\circ$), all possible 89 splicing positions $[0.5, ..., 44.5]$ in 500 ms steps as well as 3 special tokens for seq2seq translation.

4 Evaluation

We perform a variety of experiments to show the robustness and general applicability of our method.

For single splicing, we report the top-n accuracy (acc) per sample for $n \in [1, 5]$. Thus, we report the relative frequency that all splicing points are predicted correctly within the n most likely predictions. To provide a more detailed insight on the quality of each prediction, we additionally report the Euclidean distance from the true splicing point d_{sp}. For multisplicing, we compute the Jaccard coefficient J between the predicted splicing points $\hat{P}_{\tilde{\mathbf{S}}}$ and ground truth $P_{\tilde{\mathbf{S}}}$ per sample $\tilde{\mathbf{S}}$. It is calculated as

$$\text{J} = \frac{|P_{\tilde{\mathbf{S}}}| \cap |\hat{P}_{\tilde{\mathbf{S}}}|}{|P_{\tilde{\mathbf{S}}}| \cup |\hat{P}_{\tilde{\mathbf{S}}}|} \ , \tag{4}$$

where 0 means no intersection with the ground truth of (non)-splicing points and 1 is the perfect score. Note that J is independent of the elements' order in $P_{\tilde{\mathbf{S}}}$ and $\hat{P}_{\tilde{\mathbf{S}}}$. We additionally report the recall R. For both metrics, we tolerate time errors for splicing points within a narrow window of size $w = 0.5\,\text{s}$. We also report softer variants J_w and R_w with larger tolerance windows of $w \in \{1\,\text{s}, 2\,\text{s}, 3\,\text{s}\}$.

4.1 Models and Training Procedures

We consider various baseline methods in our experiments. To the best of our knowledge, the only end-to-end trainable DL method for audio splicing detection was proposed by Jadhav *et al.* [10]. We do not compare to the method by Zhang *et al.* [36] due to its very limiting constraints (cf. Sec. 2.2), which are not straightforward to relax to our more general task. We re-implemented the method of Jadhav *et al.* [10] and filled unspecified network components with plausible settings. This includes the padding strategy, the kernel size k of the convolutional layers and the dimension d of the two FC layers, which we chose as "same" padding, $k = 3$ and $d = 1024$. We additionally compare to state-of-the-art EfficientNet B0 [28] (EffNet-B0) and RegNet-x-400mf [25] (RegNet-400m) recognition models [20]. All models, including our approach, are both evaluated with single-input, which includes only the Mel spectrogram as feature, as well as multi-input, which includes all three representations as shown in Fig. 1.

Among all methods, our model is the most parameter-efficient with $\sim$ 7M, followed by RegNet-400m with $\sim$ 29M, EffNet-B0 with $\sim$ 71M parameters and the work by Jadhav *et al.* with $\sim$ 197M parameters.

We adapt the baseline NNs to our task. Contrary to our model that generalizes naturally to varying lengths, the CNNs expect fixed input/output sizes. We thus pad all inputs to the maximum length of 90 and concatenate all three audio representations along the height. The input layer is thus set to $1 \times 90 \times 277$. For multisplicing, we extend the output FC layer from one to the maximum possible number N of splicing positions, predicting $\hat{p}_i \in [0.5, 1.0, \ldots, 44.5]$ with $i \in [1, N]$ or special token $\hat{p}_{N+1} = \circ$ if no splicing is detected. Given the corresponding ground truth, all models, including ours, are trained using the cross-entropy loss.

Per model, we increase the computational speed by enlarging the training batch size for as long as the training still convergences. We chose a batch size of 512 for our model, 256 for Jadhav *et al.* [10] and RegNet-400m [25], and 64 for EffNet-B0 [28]. Still, the Adam optimizer with learning rate $1e^{-4}$ proved to be most beneficial for all models. We train with early stopping on the validation loss with delta $\delta = 0.2$ and a patience of 10 epochs. Our model is trained with teacher forcing.

We also compare our method to Capoferri *et al.* [3], as a recently proposed method with handcrafted features for single splicing detection. For the first part of the comparative evaluation in Sec. 4.2, we include RIRs in step 1 of our data generation pipeline (cf. Fig. 1). Hence, this feature is present in our data and their analytic approach is applicable in this case. Note, that we consistently allow the same RIR to be sampled for audio sources to model both equal and different environmental sources.

4.2 Basic Single-Splicing Forgery Model

In our first experiments, we consider basic forgery scenarios that only cover single splices. We first conduct a cross-dataset validation (Sec. 4.2), where we omit any post-processing operations. In a second step, the scenario is more difficult with additive synthetic noise and single compression post-processing (Sec. 4.2) as easy mechanisms to disguise splicing.

Cross-Dataset Validation We perform a cross-dataset validation experiment between data from the ACE [7] and the TTS [2] dataset. For each, we generate a training set of 500 000 and multiple test sets of 30 000 samples as described in Sec. 3.1. For now, we omit post-processing operations (step 4,5) in the generation pipeline (cf. Fig. 1). This corresponds to the most basic forgery model, where new content is generated from samples of the same speaker with no camouflage operations to disguise the splicing points.

Figure 3 shows the results. The top row reports accuracy, the bottom row the distance d_{sp} to the ground truth splicing point. The four plots in one row show the four combinations of source datasets for training and testing.

Our method (violet) performs best in every case, while Jadhav *et al.* (yellow) is the best baseline. In general, our method in multi-input mode (solid lines)

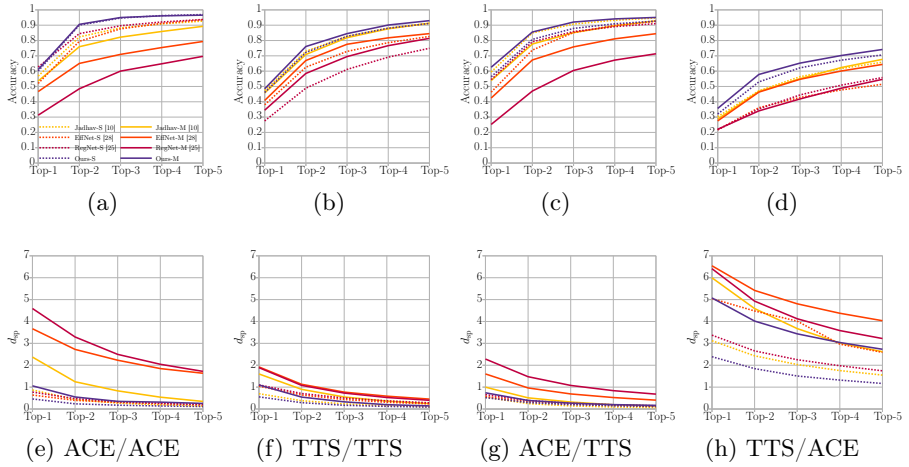
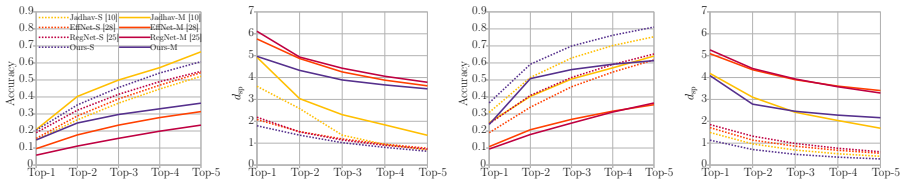


Fig. 3: Results for cross-data validation for the single- (dotted lines) and multi-input (solid lines) variants of the models. Each column portrays one train set/test set run. Top row: top-n accuracies. Bottom row: d_{sp}. We perform best for all combinations. Generalizing from TTS shows to be the most difficult.

achieves an equal (Fig. 3a) or higher accuracy (Fig. 3b-3d) than the single-input variant (dotted lines), even if d_{sp} is slightly larger (Fig. 3e-3h). Contrary, the CNN baselines [10, 25, 28] mostly perform worse when in multi-input mode. We assume a slight over adaptation due to the additional input representation information, such that training on the single (sparser) representation is beneficial in this case. Concerning the cross-dataset combinations, generalizing from TTS to ACE is the hardest for all models (Fig. 3d, Fig. 3h), while both the ACE/TTS (Fig. 3c, Fig. 3g) and the intra dataset experiment on TTS (Fig. 3b, Fig. 3f) exhibit significantly better results. We thus assume that TTS may contain specific characteristics to which the models adapt to, which results in a decreasing performance when testing on another dataset. For this reason, we take ACE/TTS as training/test combination for all following experiments.

We also run the method by Capoferri *et al.* [3] on both datasets. By design, it only provides top-1 accuracies. We report an accuracy of 4.9% and 6.8%, as well as a d_{sp} of 4.60 and 2.93 for ACE and TTS, which is below the other methods. We hypothesize that the discrepancy to the reported performances by Capoferri *et al.* is due to differences in the dataset construction. While Capoferri *et al.* [3] randomly splice signals, we only splice during silent points, which makes the detection task considerably more difficult.

Influence of Noise and Compression To disguise splicing points, a forger may take simple measures. Here, we consider additive synthetic noise, MP3 compression, and AMR-NB compression as post-processing operations on the forgery.



(a) (b)

Fig. 4: Results for robustness (Fig. 4a) and adaptability tests (Fig. 4b) for single- (dotted lines) and multi-input (solid lines) models. Overall, our model adapts best (Fig. 4b) and is second best w.r.t robustness (Fig. 4a). Contrary to most other experiments, the single-input accuracy is higher than multi-input accuracy.

Additive noise may be easily added with openly available audio editing tools [1, 22]. Encoding the audio data in a lossy compression format is even more straightforward. We consider white Gaussian noise as a general noise approximation and MP3 and AMR-NB as two popular lossy compression formats. We generate a test set from the TTS test split (Sec. 3.1) with 30 000 samples and consider weak to strong degradation influence (step 4,5 in Fig. 1). In detail, per sample, we first randomly chose a SNR $\in [-10, 50]$ db for noise. Then, randomly either MP3 or AMR-NB is chosen for compression with random $b_r \in [10, 128] \frac{kb}{s}$ for MP3 and $b_r \in \{4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2, 12.2\} \frac{kb}{s}$ for AMR-NB. In addition, we evaluate the adaptability of the models towards all degradations. For this, we fine-tune all NNs on a ACE training set of 500 000 samples with the same degradation pipeline and rerun on the TTS test set.

The NNs' results are reported in Fig. 4. Figure 4a shows the robustness of the models trained on the clean training set towards the degraded test set. With the exception of Jadhav *et al.* [10], the single-input models exhibit higher accuracy (Fig. 4a, left). Our single-input model performs second best w.r.t accuracy and has the lowest d_{sp} (Fig. 4a, right). When fine-tuned on the degradations, all models perform better as expected (Fig. 4b). Similar to the CNNs in the cross-dataset setting, the single-input variants are superior, with the difference that also our approach performs better here with single-input. This is, however, a transient effect: the multi-input variant consistently performs better for more complex training and test settings as will be shown in Sec. 4.3.

We also evaluate the handcrafted baseline by Capoferri *et al.* [3]. Its performance is below the NNs for all experiments with an accuracy of 7% and $d_{\text{sp}} = 2.9$. More in detail, the method achieves a good $d_{\text{sp}} \leq 1.5$ for 30% of the test samples. However, for the remaining 70%, the error is considerably larger.

4.3 Advanced Multi-Splicing Forgery Model

In our following experiments, we extend our forgery model to incorporate various multi-splicing scenarios for evaluating the robustness of the proposed method. To

this end, we generate a training set of 500000 samples from the ACE training split and fine tune the models previously trained on single splicing on it. We include $n \in [0, 5]$ splicing points in equal numbers, and apply all available processing steps, *i.e.*, RIRs, additive white Gaussian noise, and single MP3 or AMR-NB compression with the same SNR and b_r ranges as in Sec. 4.2.

For testing, we now assume a more elaborate forger and hence define several multi-splicing scenarios as they might be expected in the real world. For each scenario, we use test sets with 10 000 samples from the TTS test speaker pool and test the models' generalization ability to the diverse splicing situations that differ from the training setting. The method by Capoferri *et al.* does not support multi splicing, hence we exclude it from this evaluation.

Multi Compression In this experiment, we consider the difficult case of a forger who not only masks multiple splicing points with noise, but who also compresses the result multiple times afterwards. Note that multi compression may also occur during recurring up- and downloading from the internet.

We consider $n_c \in [0, 5]$ compression runs, which also includes uncompressed and single compressed files for reference. We generate 6 test sets, one for each n_c. The full data generation pipeline is applied with the same distortion parameters as for training, which includes RIRs as well as synthetic noise addition. However, the compression step is repeated n_c times with randomly sampled parameters.

The evaluation results are reported in Fig. 5. All baseline CNNs [10, 25, 28] perform comparably and fluctuate around 13.5% for single-input and 14% for multi-input along all n_c. We note that they show generalization problems towards this task and mostly collapse to predicting the same splicing points for most samples. We ran a series of experiments to tune the hyperparameters of these methods, but were not able to resolve this problem. Since our model did not show such problems, we attribute the advantage to the Transformer seq2seq design which is more naturally suited to the processing of sequential audio information than the per-position baseline classifiers.

As expected, the performance of our model decreases with increasing number of compression runs. Our multi-input method outperforms the single-input method except for the case of no compression, where the latter has a notable advantage of up to 10.0 percentage points (pp) for all metrics. For $w = 1\,\mathrm{s}, 2\,\mathrm{s}$, the multi-input model improves further by 5.7pp for J_{1s} and R_{1s}, and even by 10.7pp for J_{2s} and 10.5pp for R_{2s}.

For the most difficult case with $n_c = 5$ compression runs, our model still achieves $\mathrm{J}_{2\mathrm{s}} = 43.4\%$ and $\mathrm{R}_{2\mathrm{s}} = 46.6\%$ which we consider a notable generalization ability towards this difficult setting.

Intersplicing Intersplicing addresses a particularly difficult scenario, where a forgery is spliced together from a speaker recorded in a static (*i.e.*, unchanging) environment without significant impulse response cues. This can be expected from rather anechoic surroundings like open spaces. We omit RIRs and any further post-processing operations in this experiment and generate one test set

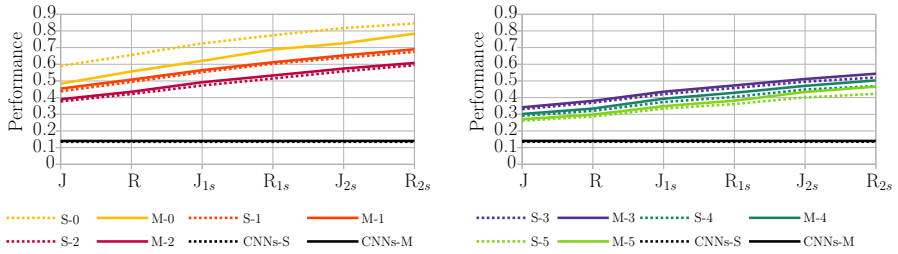


Fig. 5: Performance on $n_c \in [0, 2]$ (left) and $n_c \in [3, 5]$ (right) compression runs for our method trained on single- (dotted lines) and multiple-input (solid lines) representations. All baseline CNNs [10, 25, 28] have difficulties to generalize to the task and perform equally with minimal fluctuations. Our multi-input model exhibits greater robustness than the single-input variant except for $n_c = 0$.

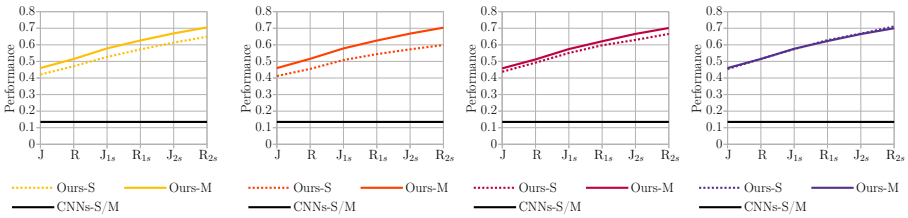
for each number of $n \in [0, 5]$ possible splicing points. Due to little cues and strong deviation from the training set distribution, this experiment requires particularly strong generalization abilities.

In fact, similarly to the phenomenon described in Sec. 4.3, no baseline CNN [10, 25, 28] can meet this requirement. All baseline methods default to a constant performance of 16.7% for J_w and R_w with $w \in \{0.5\,\text{s}, 1\,\text{s}, 2\,\text{s}\}$. Both our single and multi-input model are able to generalize to a certain degree. However, intersplicing proves to be the most challenging of all tested scenarios. The single-input variant achieves a J_w of 17.2%, 19.0% and 21.3% and a R_w of 17.9%, 20.0% and 22.7% for $w \in \{0.5\,\text{s}, 1\,\text{s}, 2\,\text{s}\}$. The multi-input variant performs comparably with a J_w of 16.7%, 17.5% and 18.8%, and a R_w of 17.0%, 18.0% and 19.4% for $w \in \{0.5\,\text{s}, 1\,\text{s}, 2\,\text{s}\}$, respectively.

Overall, we report that our Transformer seq2seq is the only model that generalizes towards this task. However, there is still room for robustness improvements towards data with such large deviations from the training distribution.

Real World Noise In training, we cover additive white Gaussian noise to approximate background noises. We now evaluate the robustness towards composite samples distorted by a forger with additive real noise to hide splicing points more convincingly. Real noise samples can be easily downloaded from the internet. For our experiments, we chose free ambiance sound samples featuring rain[1], a train passing by[2], an crowded exhibition hall[3], and a crowded boarding gate at the airport[4]. We generate one test set per noise type with SNRs uniformly drawn from the range $[-10, 50]$ db per sample. We also include RIRs and single compression post-processing as described in Sec. 4.3.

[1] https://freesound.org/people/straget/sounds/531947/
[2] https://freesound.org/people/theplax/sounds/615849/
[3] https://freesound.org/people/BockelSound/sounds/487600/
[4] https://freesound.org/people/arnaud%20coutancier/sounds/424362/



(a) airport (b) exhibition (c) rain (d) train

Fig. 6: Performance on 4 real world noise types. The CNN baselines [10, 25, 28] are unable to generalize. Our multi-input model (solid lines) is best except for train noise, where the single-input variant (dotted lines) performs comparably.

The results are reported in Fig. 6. As in all previous multi-splice experiments, the tested CNNs [10, 25, 28] are unable to generalize and yield a performance of 13.5% consistently for all metrics. Our multi-input model outperforms the single-input variant especially for the more complex airport and exhibition background noise (Fig. 6a, Fig. 6b). For the former noise, the average increase over all $w \in \{0.5\,\mathrm{s}, 1\,\mathrm{s}, 2\,\mathrm{s}\}$ is 4.9 and 5.3pp for J_w and R_w, respectively. For the latter noise, the average increase is even 7.1 and 8.2pp, respectively. Averaged over all real world noises, our best performing multi-input model shows satisfying robustness with $\mathrm{J} = 45.9\%$ and $\mathrm{R} = 51.4\%$ and for the 2 s windows even reaches a performance of $\mathrm{J}_{2\mathrm{s}} = 66.7\%$ and $\mathrm{R}_{2\mathrm{s}} = 70.3\%$.

5 Conclusion

This work investigates the robust detection and localization of single and multiple splices in audio forgeries under unconstrained settings. We propose a Transformer seq2seq network for this task. We perform extensive evaluations that cover basic and advanced forgery models, including splicing of samples from different/same, echoic/anechoic recording surroundings, and post-processing operations like additive synthetic/real noise and single/multiple compression runs that may disguise splicing. Our method clearly outperforms competing networks and CNN baselines, while requiring the smallest number of parameters. By design, it is also more universally applicable than methods with handcrafted features that exploit specific manipulation characteristics.

The proposed method generalizes well to challenging scenarios with multiple splices that cannot be solved by other CNNs. We hypothesize that this is due to the better suitability of a seq2seq model for processing sequential audio data.

After this first step into the direction of unconstrained audio splicing detection and localization we aim at further improving the robustness of our method. We expect that there is still room for improvement in particularly challenging situations like the generalization to intersplicing (Sec. 4.3) or a large number of post-processing compression steps (Sec. 4.3.)

References

1. Audacity: Audacity ® | Free, open source, cross-platform audio software for multi-track recording and editing (accessed: 2022-05-12), https://www.audacityteam.org/
2. Bakhturina, E., Lavrukhin, V., Ginsburg, B., Zhang, Y.: Hi-Fi Multi-Speaker English TTS Dataset. In: Proc. Interspeech. pp. 2776–2780 (2021). https://doi.org/10.21437/Interspeech.2021-1599
3. Capoferri, D., Borrelli, C., Bestagini, P., Antonacci, F., Sarti, A., Tubaro, S.: Speech Audio Splicing Detection and Localization Exploiting Reverberation Cues. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1–6. IEEE (2020)
4. Chen, L., Maddox, R.K., Duan, Z., Xu, C.: Hierarchical Cross-Modal Talking Face Generation with Dynamic Pixel-Wise Loss. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7832–7841 (2019)
5. Cooper, A.J.: Detecting Butt-Spliced Edits in Forensic Digital Audio Recordings. In: Audio Engineering Society Conference: 39th International Conference: Audio Forensics: Practices and Challenges. Audio Engineering Society (2010)
6. Cuccovillo, L., Mann, S., Tagliasacchi, M., Aichroth, P.: Audio Tampering Detection via Microphone Classification. In: 2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP). pp. 177–182. IEEE (2013)
7. Eaton, J., Gaubitch, N.D., Moore, A.H., Naylor, P.A.: Estimation of Room Acoustic Parameters: The ACE Challenge. IEEE/ACM Transactions on Audio, Speech, and Language Processing **24**(10), 1681–1693 (2016)
8. Esquef, P.A., Apolinário, J.A., Biscainho, L.W.: Improved Edit Detection in Speech via ENF Patterns. In: 2015 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1–6. IEEE (2015)
9. Gao, Y., Singh, R., Raj, B.: Voice Impersonation Using Generative Adversarial Networks. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2506–2510. IEEE (2018)
10. Jadhav, S., Patole, R., Rege, P.: Audio splicing detection using convolutional neural network. In: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–5. IEEE (2019)
11. Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Lopez Moreno, I., Wu, Y., et al.: Transfer Learning from Speaker Verification to Multispeaker Text-to-Speech Synthesis. Advances in Neural Information Processing Systems **31** (2018)
12. Korshunov, P., Gonçalves, A.R., Violato, R.P., Simões, F.O., Marcel, S.: On the Use of Convolutional Neural Networks for Speech Presentation Attack Detection. In: 2018 IEEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA). pp. 1–8. IEEE (2018)
13. Lin, X., Kang, X.: Exposing Speech Tampering via Spectral Phase Analysis. Digital Signal Processing **60**, 63–74 (2017)
14. Lin, X., Kang, X.: Supervised Audio Tampering Detection Using an Autoregressive Model. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2142–2146. IEEE (2017)
15. Luo, D., Wu, H., Huang, J.: Audio Recapture Detection Using Deep Learning. In: 2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP). pp. 478–482. IEEE (2015)

16. Luo, D., Yang, R., Huang, J.: Detecting Double Compressed AMR Audio Using Deep Learning. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2669–2673. IEEE (2014)
17. Mao, M., Xiao, Z., Kang, X., Li, X., Xiao, L.: Electric Network Frequency Based Audio Forensics using Convolutional Neural Networks. In: IFIP International Conference on Digital Forensics. pp. 253–270. Springer (2020)
18. McKinney, M., Breebaart, J.: Features for Audio and Music Classification (2003)
19. Meng, X., Li, C., Tian, L.: Detecting Audio Splicing Forgery Algorithm Based on Local Noise Level Estimation. In: 2018 5th International Conference on Systems and Informatics (ICSAI). pp. 861–865. IEEE (2018)
20. Models, P.V.: torchvision.models — Torchvision 0.11.0 documentation (accessed: 2022-03-02), https://pytorch.org/vision/stable/models.html
21. Nirkin, Y., Keller, Y., Hassner, T.: FSGAN: Subject Agnostic Face Swapping and Reenactment. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7184–7193 (2019)
22. Oceanaudio: ocenaudio (accessed: 2022-05-12), https://www.ocenaudio.com//
23. Otter, D.W., Medina, J.R., Kalita, J.K.: A Survey of the Usages of Deep Learning for Natural Language Processing. IEEE Transactions on Neural Networks and Learning Systems **32**(2), 604–624 (2020)
24. Pan, X., Zhang, X., Lyu, S.: Detecting Splicing in Digital Audios Using Local Noise Level Estimation. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1841–1844. IEEE (2012)
25. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing Network Design Spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10428–10436 (2020)
26. Rouniyar, S.K., Yingjuan, Y., Hu, Y.: Channel Response Based Multi-Feature Audio Splicing Forgery Detection and Localization. In: Proceedings of the 2018 International Conference on E-Business, Information Management and Computer Science. pp. 46–53 (2018)
27. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations (2015)
28. Tan, M., Le, Q.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: International Conference on Machine Learning. pp. 6105–6114 (2019)
29. Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., Nießner, M.: Face2Face: Real-Time Face Capture and Reenactment of RGB Videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2387–2395 (2016)
30. Torchaudio: torchaudio.transforms — Torchaudio 0.10.0 documentation (accessed: 2021-03-09), https://pytorch.org/audio/stable/
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention Is All You Need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)
32. Viazovetskyi, Y., Ivashkin, V., Kashin, E.: StyleGAN2 Distillation for Feed-Forward Image Manipulation. In: European Conference on Computer Vision. pp. 170–186. Springer (2020)
33. Yan, D., Dong, M., Gao, J.: Exposing Speech Transsplicing Forgery with Noise Level Inconsistency. Security and Communication Networks **2021** (2021)
34. Yang, R., Qu, Z., Huang, J.: Detecting Digital Audio Forgeries by Checking Frame Offsets. In: Proceedings of the 10th ACM Workshop on Multimedia and Security. pp. 21–26 (2008)

35. Zhang, Z., Yi, X., Zhao, X.: Fake speech detection using residual network with transformer encoder. In: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security. pp. 13–22 (2021)
36. Zhang, Z., Zhao, X., Yi, X.: Aslnet: An encoder-decoder architecture for audio splicing detection and localization. Security and Communication Networks **2022** (2022)
37. Zhao, H., Chen, Y., Wang, R., Malik, H.: Audio Source Authentication and Splicing Detection Using Acoustic Environmental Signature. In: Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security. pp. 159–164 (2014)
38. Zhao, H., Chen, Y., Wang, R., Malik, H.: Audio Splicing Detection and Localization Using Environmental Signature. Multimedia Tools and Applications **76**(12), 13897–13927 (2017)