Deepfakes and How to Detect Them

Petr Grinberg
DLS 2025

Partially Based on HSE DLA Course: https://github.com/markovka17/dla

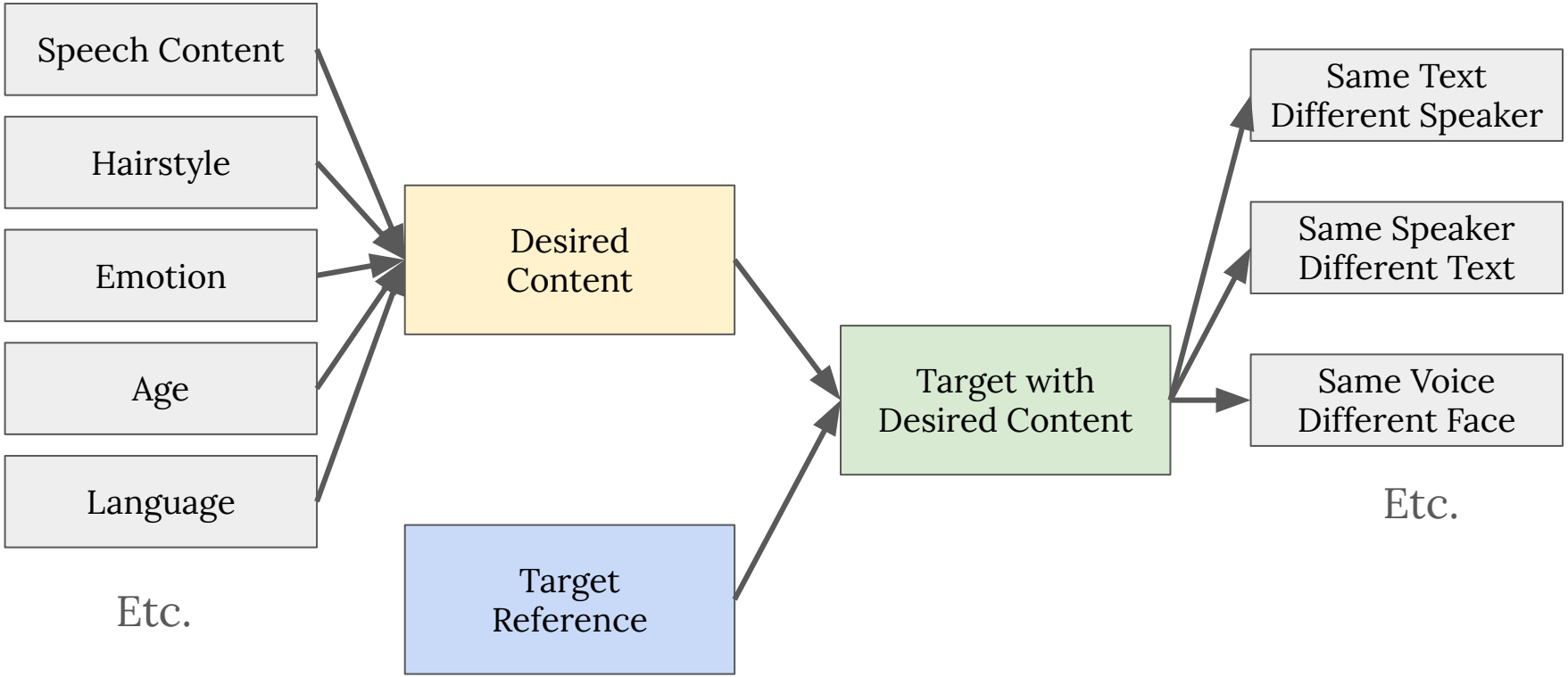
Welcome! I am Petr

- ***MSc EPFL, LauzHack Mentor***
- ***Lecturer at NRU HSE***
- ***Sony R&D***
- ***Ex. LCAV EPFL***
- ***Ex. Reality Defender Inc.***
- ***Ex. Samsung R&D***

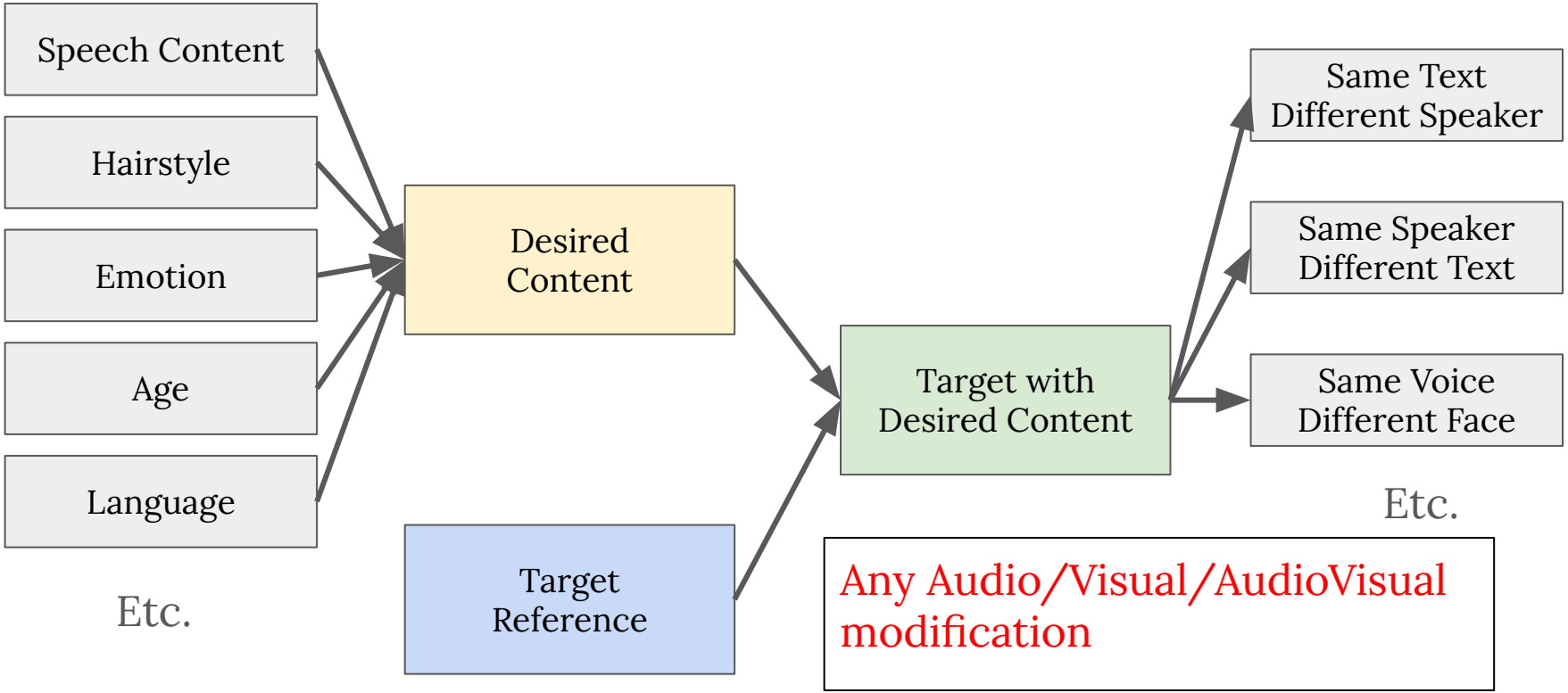
Follow me on GitHub (@Blinorot)

Deepfakes

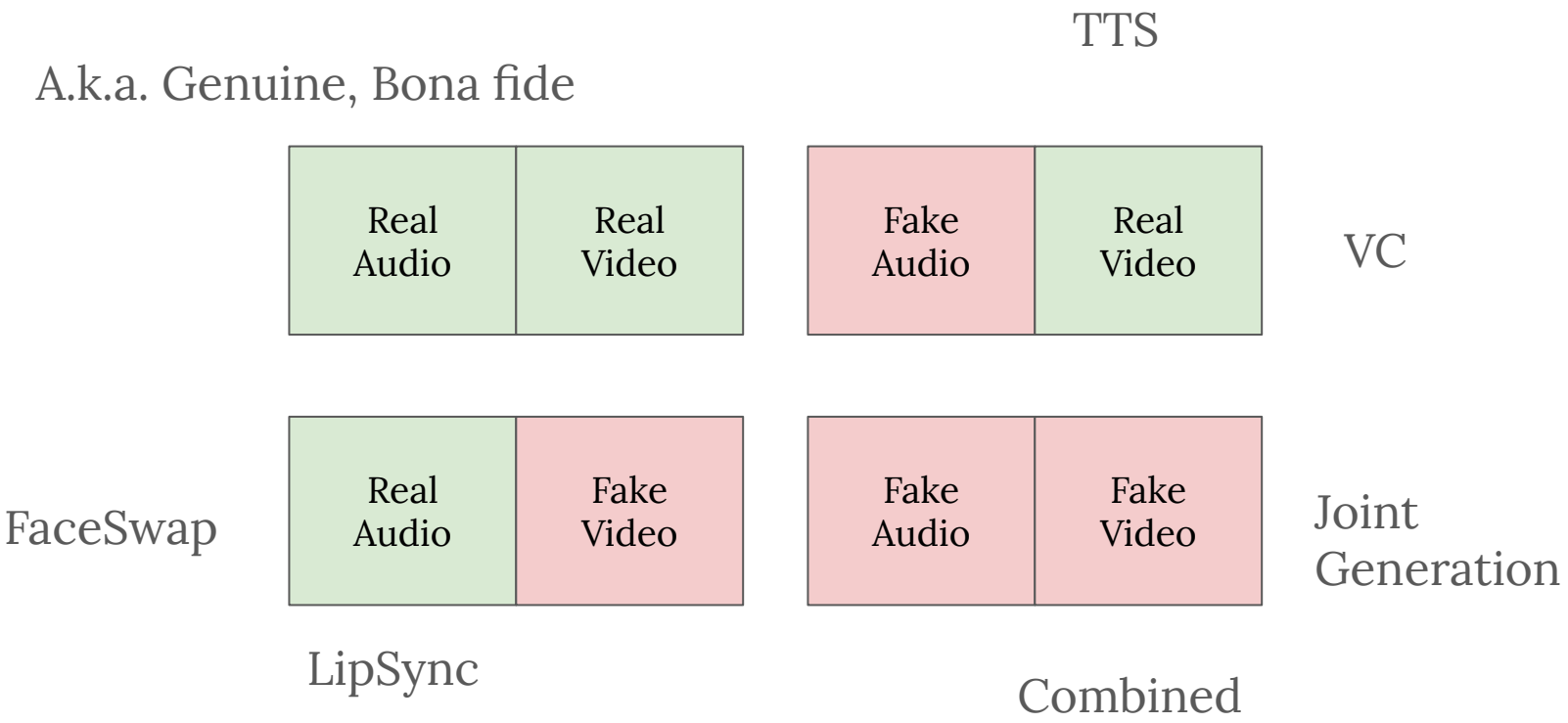
What is a deepfake?



What is a deepfake?



What is a deepfake?



Fake Audio

Text-To-Speech (Can also be expressive, e.g. with emotion or singing):

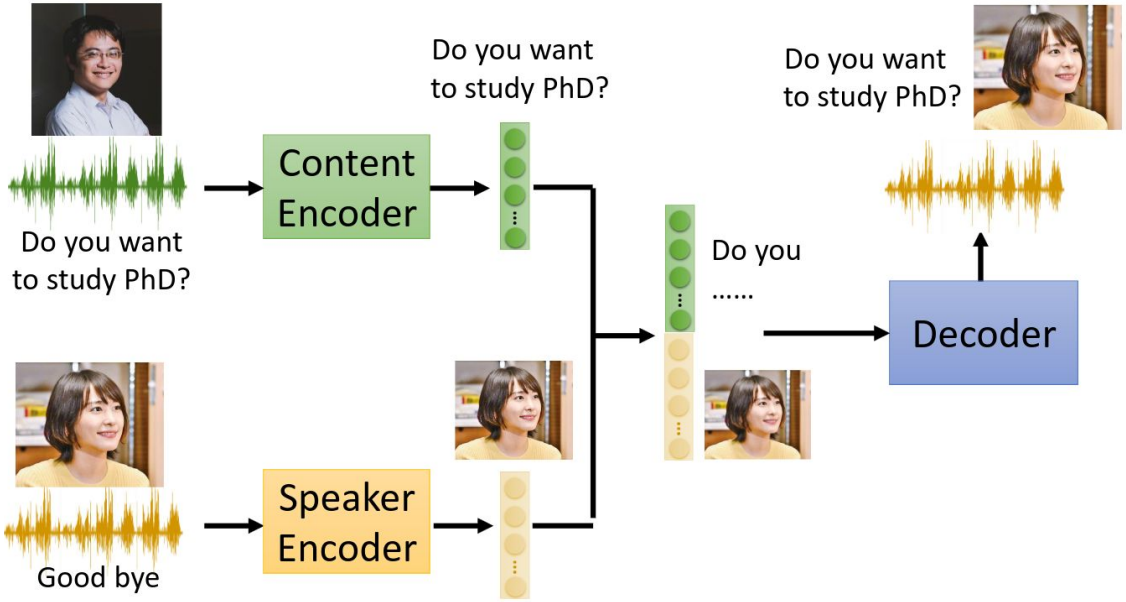
- Real-Mel + Vocoder
- Fake-Mel + Vocoder
- K-means on Latent Representation + Unit-Vocoder
- Neural Codecs

Can be

- GAN-Based
- Signal Processing-Based
- Diffusion-Based
- Autoregressive (LLM-like style)

Fake Audio

Voice Conversion / Voice Cloning

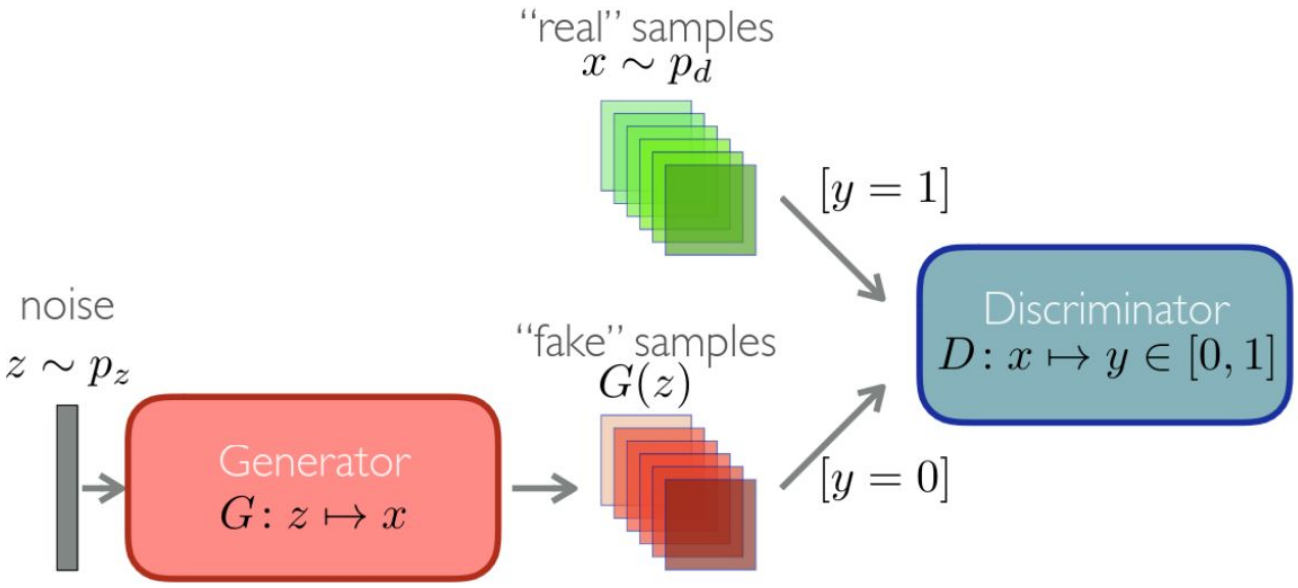


Remark: GANs

GAN consists of 2 networks: **Generator** and **Discriminator**

Discriminator solves binary classification task.

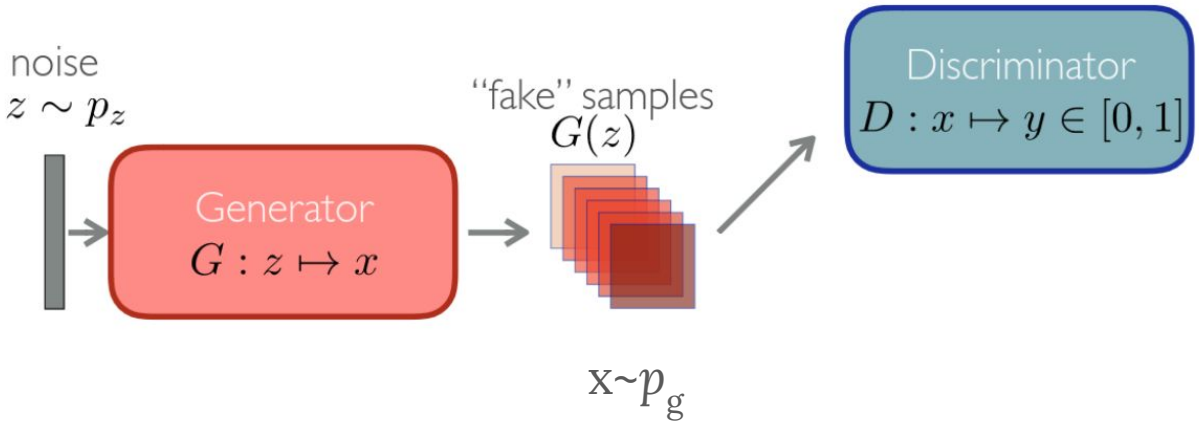
Given real and fake examples it predicts are they real or fake



Remark: GANs

GAN consists of 2 networks: **Generator** and **Discriminator**

Generator tries to generate realistic examples and fool the discriminator



Remark: GANs

There are alternative Loss Functions:
See LSGAN and WGAN

Two Networks play 2 players game:

$$\min_G \max_D \mathop{\mathbb{E}}_{x \sim p_d} [\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z} [\log(1 - D(G(z)))]$$

$$\mathscr{L}_D(G, D) = \max_D \mathop{\mathbb{E}}_{x \sim p_d} [\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z} [\log(1 - D(G(z)))]$$

$$\mathscr{L}_G(G, D) = \min_G \mathop{\mathbb{E}}_{z \sim p_z} [\log(1 - D(G(z)))]$$

$$:= (\text{in practice}) \max_G \mathop{\mathbb{E}}_{z \sim p_z} [\log(D(G(z)))]$$

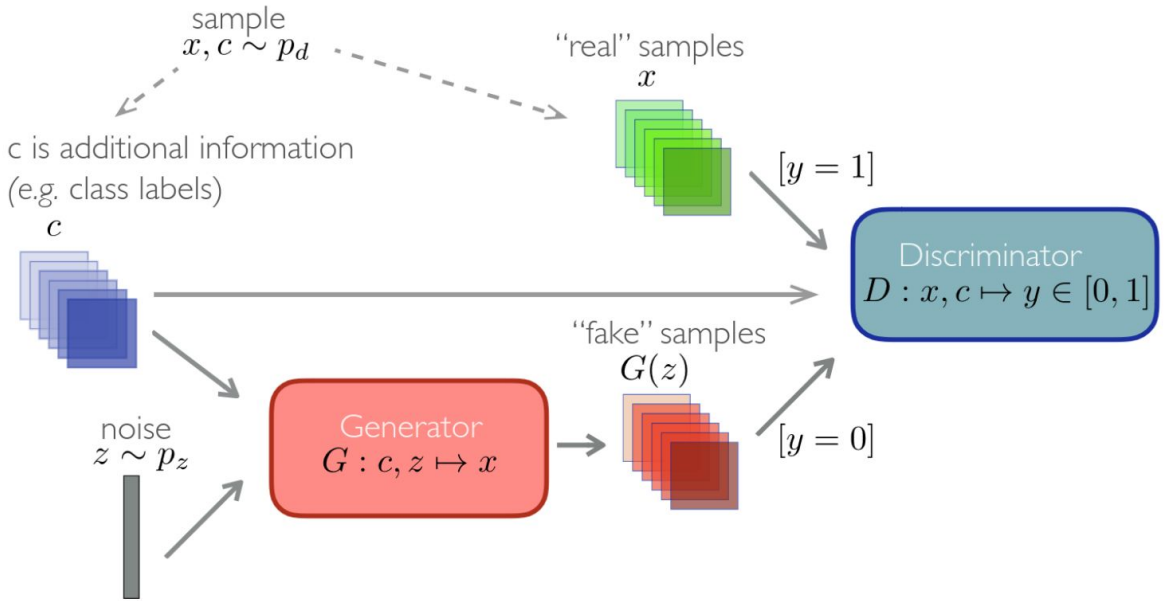
Remark: Training GANs

Train via alternating algorithm: Train Discriminator, then Generator

```
g_outputs = model.generate(**batch)
batch.update(g_outputs)

D_optimizer.zero_grad()

d_outputs = model.discriminate(generated_x=batch["generated_x"].detach(),
                               real_x=batch["real_x"])
batch.update(d_outputs)

D_loss = discriminator_criterion(**batch)
D_loss.backward()
D_optimizer.step()

G_optimizer.zero_grad()
d_outputs = model.discriminate(**batch)
batch.update(d_outputs)
G_loss = generator_criterion(**batch)
G_loss.backward()
```

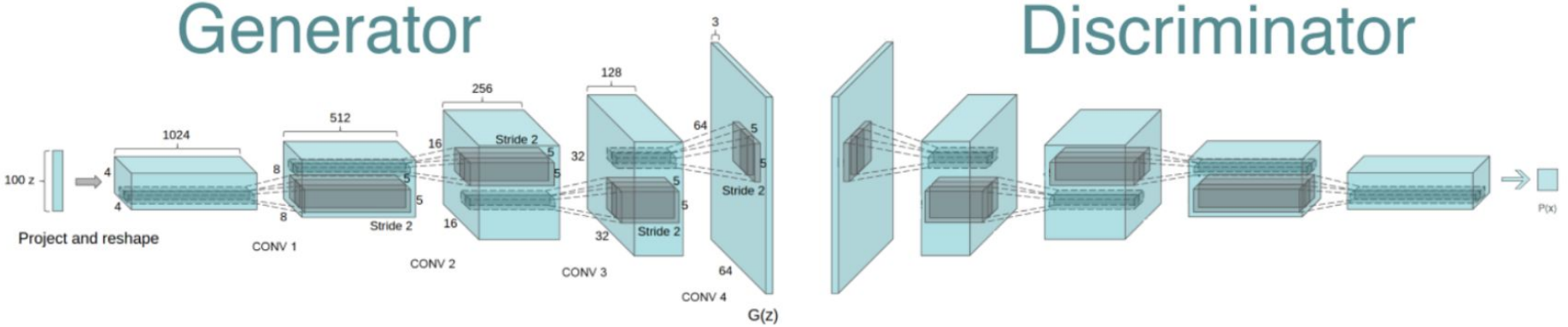
Remark: Conditional GANs (CGAN)

Conditional GANs have some other input to condition on apart from noise z

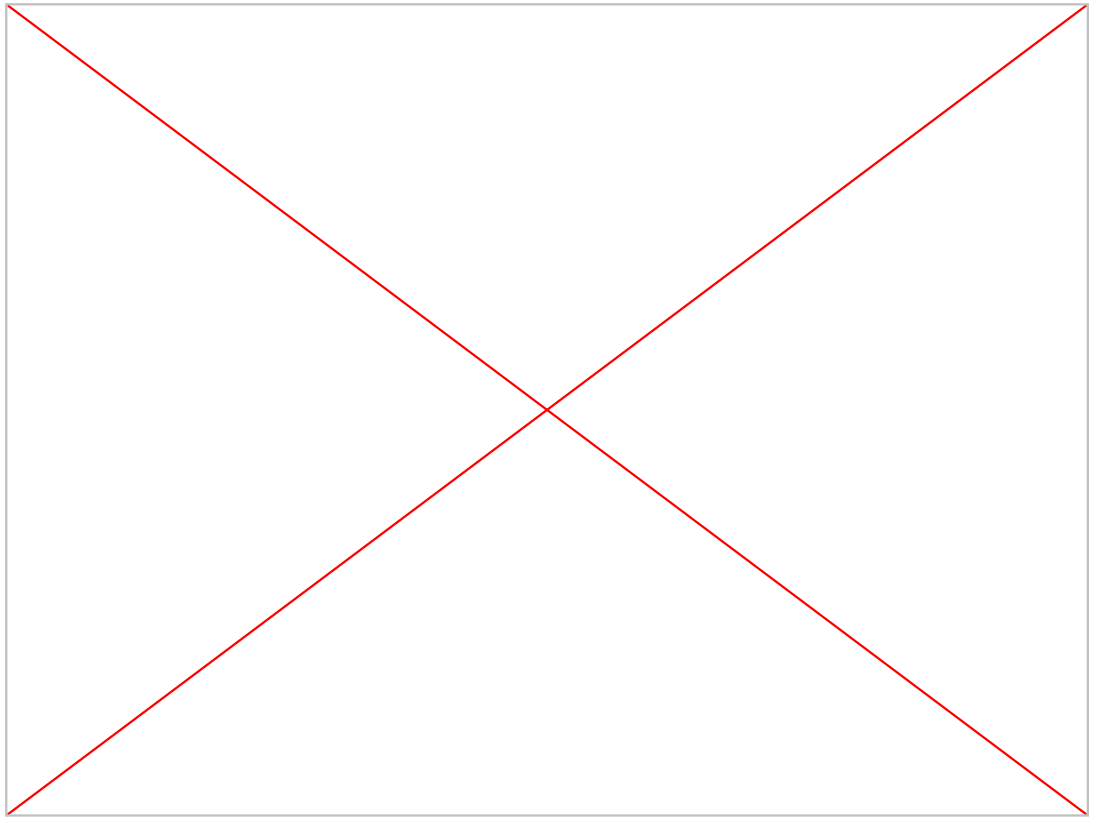


Remark: Deep Convolutional GAN (DCGAN)

Classic architecture DCGAN (Deep-Convolutional GAN)

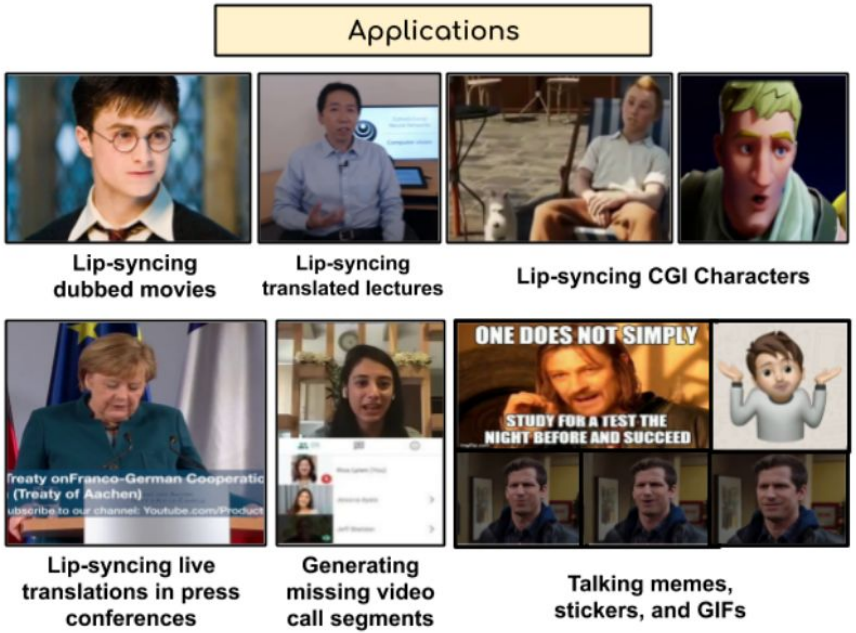


Deepfakes



Lip-Syncing

One of the most important things for realistic AV deepfakes is lip-syncing. However, AI-driven Lip-sync can help in various “more appropriate” tasks:



Complexity of the problem:

1. People can detect unsynced moments extremely fast (~0.1s)
2. Should be speaker-independent and not require extra data
3. Should work on unconstrained videos, not just on static images
4. Should handle tons of phonemes and their combinations

A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild (ACM Multimedia 2020)

Limitations of previous works:

To better understand why Wav2Lip works well, we need to understand the issues in design of previous models, such as LipGAN:

1. Reconstruction Loss for a face seems to be a good idea (to preserve pose, identity, etc.), however, **the lips are only 4% of the image** and DNN prefers other regions over the lips.
2. Temporal context should not be ignored for judging lip-sync quality.
3. Too diverse set of faces (poses, identities, etc) may cause discriminator to focus on visual GAN artifacts rather than lip-syncing quality.

A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild (ACM Multimedia 2020)

Lip-Sync Expert is all we need

Wav2Lip idea: "*Generating accurate lip-sync by learning from a well-trained lip-sync expert*"

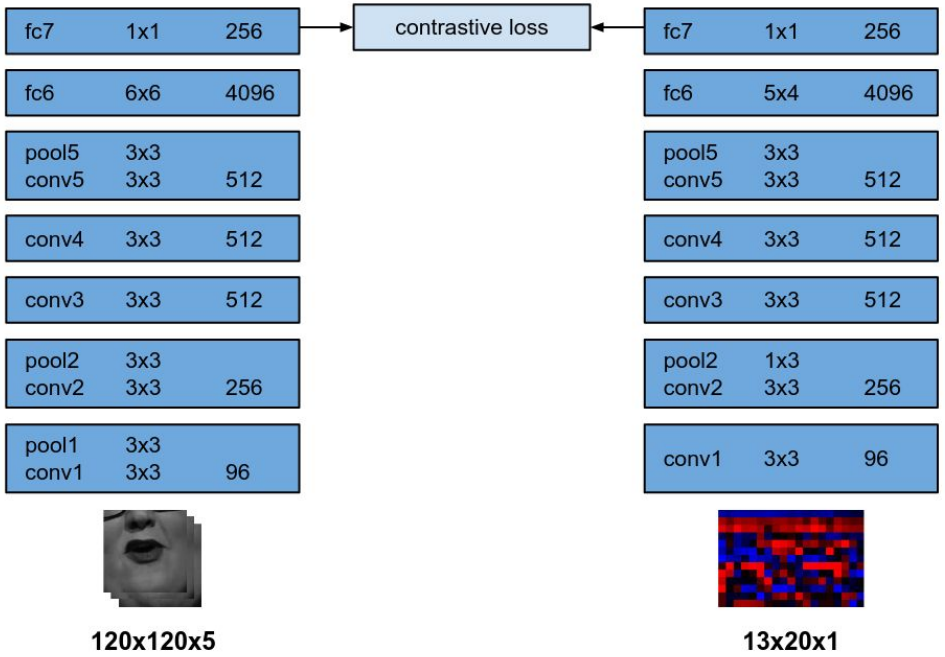
To mitigate the issues of previous works:

1. We need an **accurate** discriminator **pre-trained** on real video-audio pairs.
2. We have to **freeze** the expert to penalize for generating unsynced audio-video frames and not for anything else (pose, face, etc).
3. We have to add visual discriminator to improve generation quality

A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild (ACM Multimedia 2020)

SyncNet

We want to train video and audio encoders, so their embeddings are close when the video and audio are synced and far otherwise



Out of time: automated lip sync in the wild (ACCV 2016)

SyncNet

Dataset collection: given a video and corresponding audio

1. We take 5 frames and the corresponding audio segment – genuine pair
2. We randomly shift audio by up to 2s and take 5 video frames – false pair

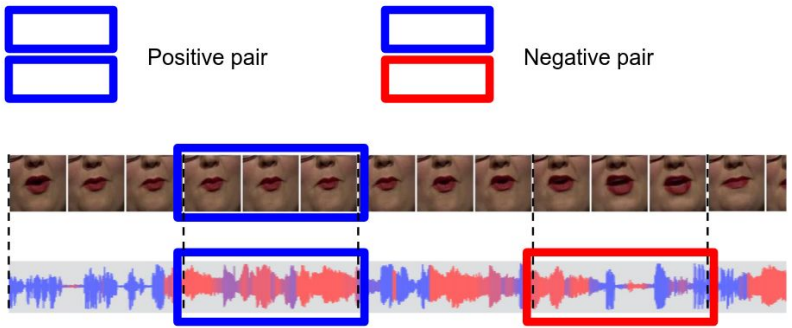


Fig. 5. The process of obtaining *genuine* and *false* audio-video pairs.

Out of time: automated lip sync in the wild (ACCV 2016)

Lip-Sync Expert: SyncNet

The discriminator is based on SyncNet with the following modifications:

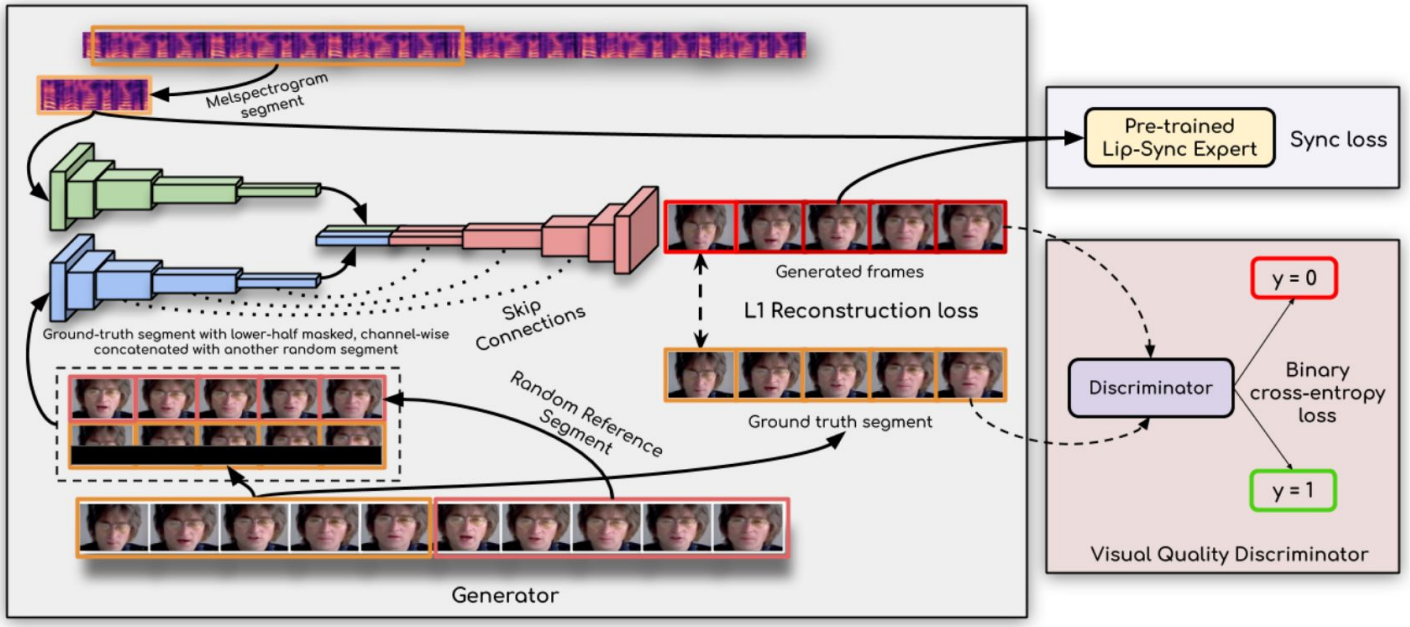
1. Colored frames instead of gray-scale ones
2. Deeper model with residual connections
3. Cosine similarity with Cross Entropy instead of L2 loss:

$$P_{\text{sync}} = \frac{v \cdot s}{max(\|v\|_2 \cdot \|s\|_2, \epsilon)}$$

A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild (ACM Multimedia 2020)

Wav2Lip

Three networks: Identity Encoder, Speech Encoder, and Decoder



Note 1:

Generator operates on a single frame independently

Note 2:
During inference Reference=Target (Target is just half-masked)

[A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild (ACM Multimedia 2020)]

Lip-Syncing

Let's look at the demo:

A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild (ACM Multimedia 2020)

How to Detect Deepfakes

Recap: Voice Biometric Systems

Automatic Speaker Verification System (ASV)

Defines whether a speaker is an owner or another **real** person.

Attack Types:

- Imposter Attacks

Countermeasure System (CM)

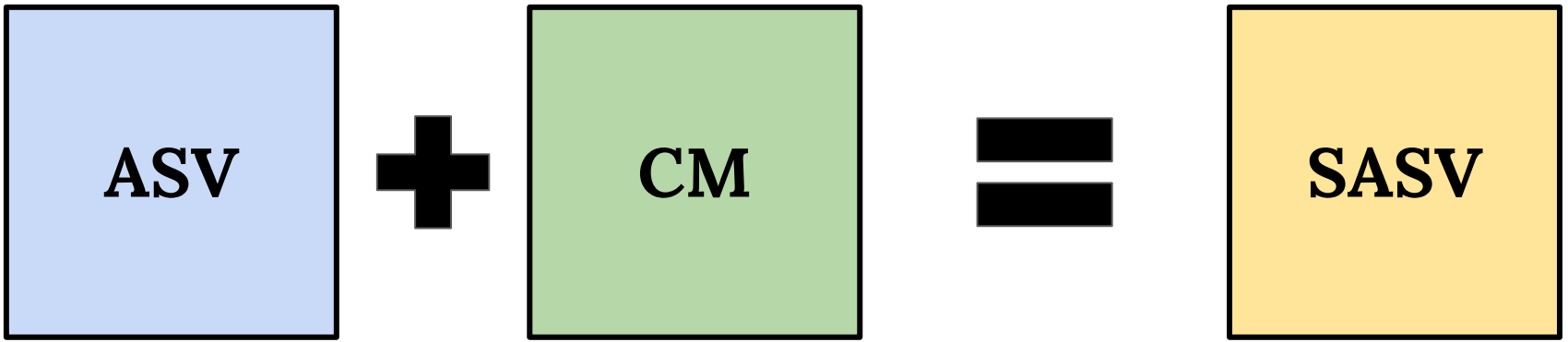
Defines whether speech is real (a.k.a. bona-fide, genuine) or not.

Attack Types:

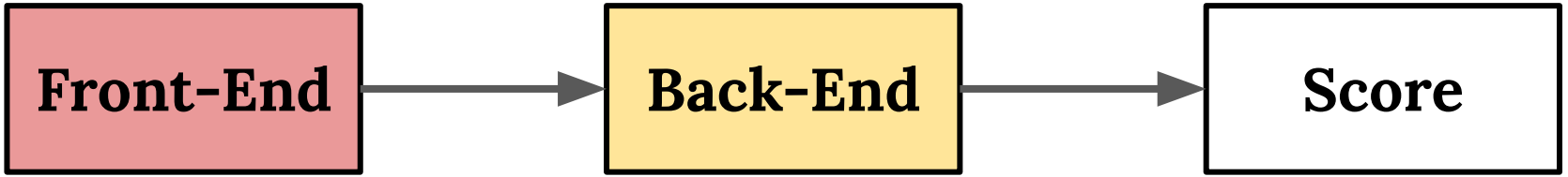
- Synthesized Speech
- Recorded Speech

Voice Biometric Systems: SASV

Complete authentication method is called Spoofing-Aware Speaker Verification System (SASV).



Voice Biometric Systems: Terminology



Input Features:

- Raw (X[n])
- Spectrogram
- Cepstral Coefficients
- Self-Supervised Embeddings

Architecture:

- Deep Neural Network
- Gaussian Mixture Model

Probability of audio spoof

Voice Biometric Systems: Metrics

- Equal Error Rate (EER). ↓

  False Acceptance – provide access to a spoofer
  False Rejection – reject access to the owner

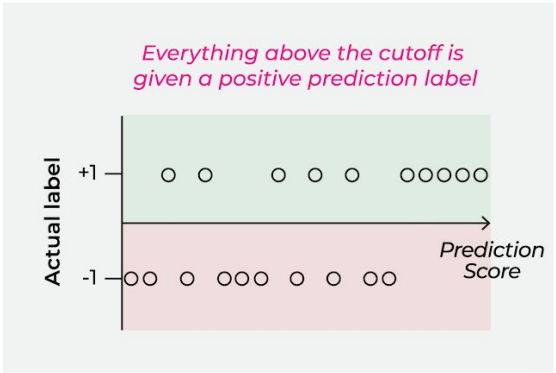
  Given a probability threshold s we can calculate $FAR(s)$ and $FRR(s)$. EER is equal to $FAR(s)$ with such s that $FAR(s) = FRR(s)$

- Detection Cost Function (DCF). Another approach. ↓

Voice Biometric Systems: Metrics

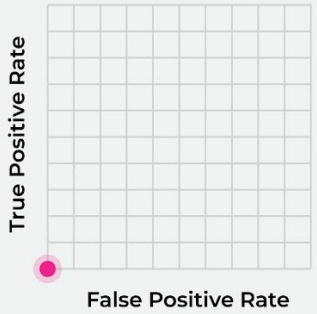
Area Under Curver (AUC) – Common metric in classification

True Positive Rate: 0/10	False Positive Rate: 0/10

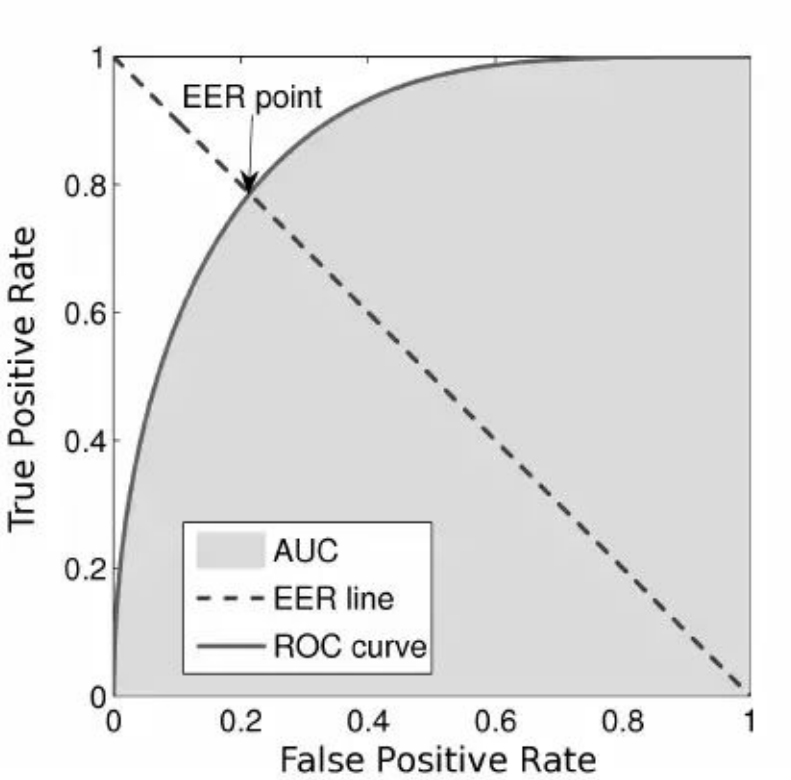


Prediction label

Actual label	POS (+)	NEG (-)
POS (+)	0	10
NEG (-)	0	10



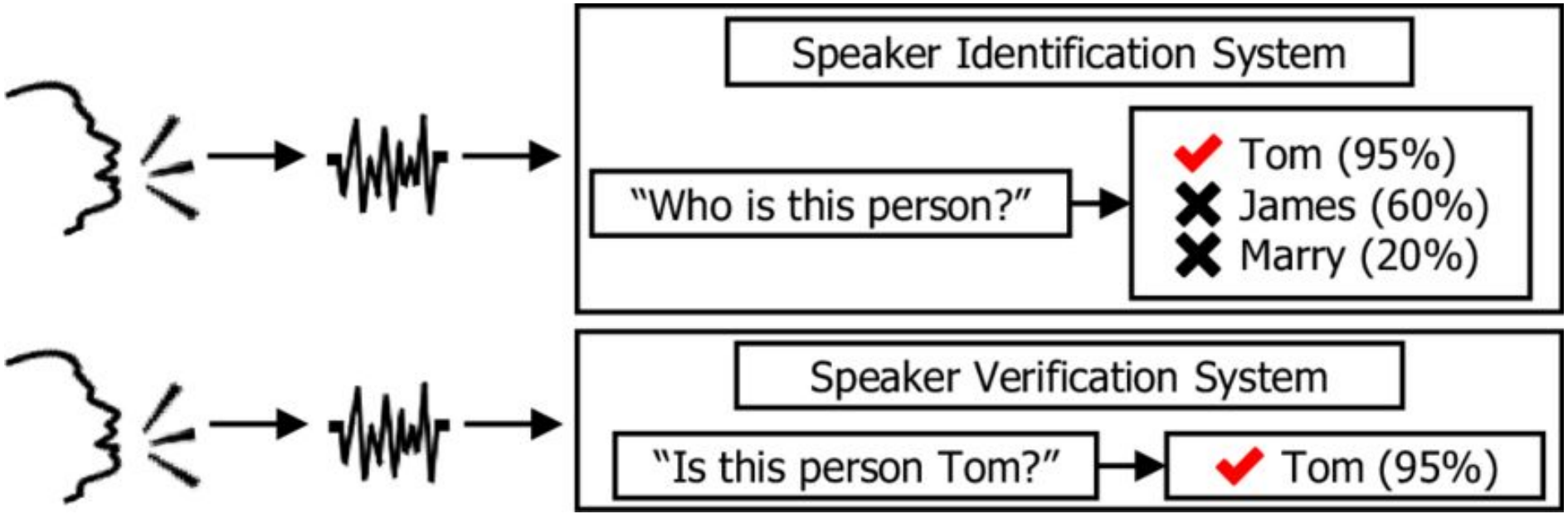
Voice Biometric Systems: Metrics



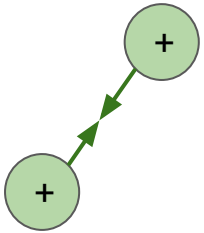
ASV

Speaker Verification Vs Recognition

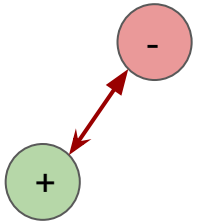
Speaker Recognition = Speaker Identification + Speaker Verification



Speaker Verification



“attract” everything that have to be close



“spread” everything that shouldn’t be close

General idea is to represent each utterance as an embedding in a such way that utterance embeddings of the same speaker are “close” and of different speakers are “distant”.

i-vectors Vs x-vectors and d-vectors

- i-vectors are the low-dimensional projection of high-dimensional statistics from universal background model (UBM).
  UBM is a fancy name for a special Gaussian Mixture Model.

- x-vectors (d-vectors) is a fancy name for embeddings obtained from a Deep Neural Network.

Similarity score

- Cosine similarity score

$$\text{similarity} = \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon)}$$

- Probabilistic Linear Discriminant Analysis (PLDA)

  Projects both embeddings into lower dimensional space where we find distance between them.

Loss functions

1. Softmax Loss, a.k.a. Cross-Entropy Loss

$$L_{\text{softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\mathbf{W}_{y_i}^T\mathbf{x}_i+\mathbf{b}_{y_i}}}{\sum_{j=1}^{c}e^{\mathbf{W}_j^T\mathbf{x}_i+\mathbf{b}_j}}$$

Softmax Loss penalizes on classification error, it does not explicitly enforce the similarity for intra-class samples and the diversity for inter-class samples.

Loss functions

2. Angular-Softmax Loss (A-Softmax Loss)

First, let's remove biases, so that b**i** = 0.
Then, the decision boundary will look like this:

$$(\mathbf{W}_1^T - \mathbf{W}_2^T)\mathbf{x} = 0 \quad \Longleftrightarrow \quad (\|\mathbf{W}_1\|\cos(\theta_1) - \|\mathbf{W}_2\|\cos(\theta_2))\|\mathbf{x}\| = 0$$

Second: let's normalize weights, so that ||W_i|| =1

$$(\|\mathbf{W}_1\|\cos(\theta_1) - \|\mathbf{W}_2\|\cos(\theta_2))\|\mathbf{x}\| = 0 \quad \Longleftrightarrow \quad \cos(\theta_1) - \cos(\theta_2) = 0$$

Loss functions

2. Angular-Softmax Loss (A-Softmax Loss)

$$L'_{\text{Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{||\mathbf{x}_i||\cos(\theta_{y_i,i})}}{\sum_{j=1}^{c}e^{||\mathbf{x}_i||\cos(\theta_{j,i})}}$$

Let's introduce a more restrictive condition:

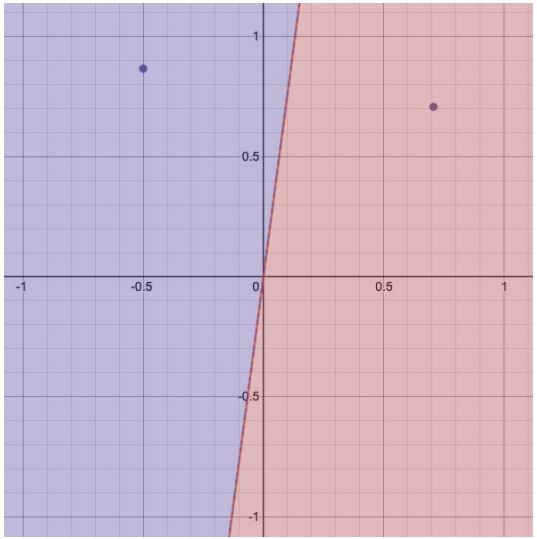
Class 1: $\cos(m\theta_1) > \cos(\theta_2)$

Class 2: $\cos(m\theta_2) > \cos(\theta_1)$

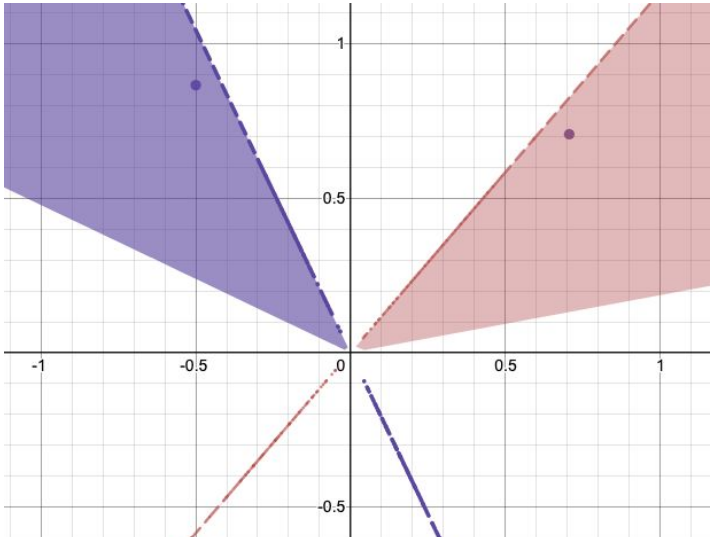
Idea: lower bound $\cos(\theta_1) > \cos(m\theta_1)$

Loss functions

2. Angular-Softmax Loss (A-Softmax Loss)



Softmax boundary



A-Softmax boundary

Loss functions: A-Softmax

$$L = \frac{1}{N}\sum_{n=1}^{N} -\log \frac{e^{\|\mathbf{x}^{(n)}\| \cos(m\theta_{y_n}^{(n)})}}{e^{\|\mathbf{x}^{(n)}\| \cos(m\theta_{y_n}^{(n)})} + \sum_{j \neq y_n} e^{\|\mathbf{x}^{(n)}\| \cos(\theta_j^{(n)})}}$$

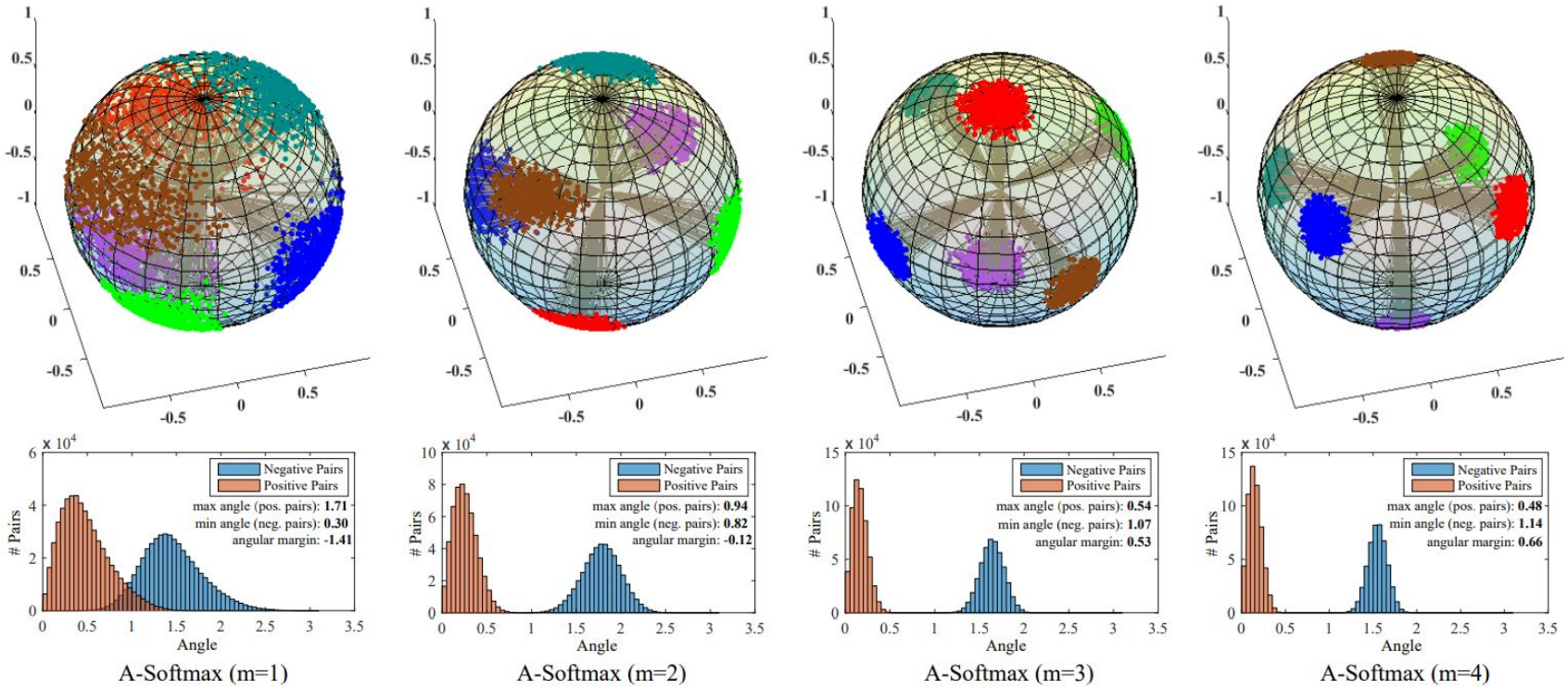
Note that in the above illustration, $\theta_{y_n}^{(n)}$ is supposed to be in $[0, \frac{\pi}{m}]$. To remove such restriction, a new function is defined to replace the cosine function as follows:

$$\phi(\theta_{y_n}^{(n)}) = (-1)^k \cos(m\theta_{y_n}^{(n)}) - 2k$$

for $\theta_{y_n}^{(n)} \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$ and $k \in [0, m-1]$. So the A-softmax loss function is finally defined as follow:

$$L = \frac{1}{N}\sum_{n=1}^{N} -\log \frac{e^{\|\mathbf{x}^{(n)}\| \phi(\theta_{y_n}^{(n)})}}{e^{\|\mathbf{x}^{(n)}\| \phi(\theta_{y_n}^{(n)})} + \sum_{j \neq y_n} e^{\|\mathbf{x}^{(n)}\| \cos(\theta_j^{(n)})}}$$

Loss functions: A-Softmax



Loss functions

3. Additive-Margin-Softmax Loss (AM-Softmax Loss)

Angular Softmax is hard to optimize. Basically, the main idea is having a lower bound in decision boundary. Let's do it by subtracting m.

$$\phi(\theta_{y_i,i}) \leq \cos(\theta_{y_i,i})$$

$$\phi(\theta_{y_i,i}) = \cos(\theta_{y_i,i}) - m$$

Loss functions

3. Additive-Margin-Softmax Loss (AM-Softmax Loss)

Let's normalize x as well and introduce new Loss:

$$L_{\text{AM-Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{Z}$$

where $Z = e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j=1,j\neq i}^{c} e^{s(\cos(\theta_{j,i}))}$ and s is a scaling factor used to make sure the gradient not too small during the training [18].

Loss functions

4. Additive-Angular-Margin-Softmax Loss (AAM-Softmax)

If x is normalized, x_i is a point on a hypersphere. The arc length between two points is a natural way to define the distance. Length depends on the angle. So a more natural definition is the following:

$$\phi(\theta_{y_i,i}) = \cos(\theta_{y_i,i} + m)$$

Loss functions

4. Additive-Angular-Margin-Softmax Loss (AAM-Softmax)

$$L_{\text{AAM-Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{Z}$$

where $Z = e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j=1,j\neq i}^{c} e^{s(\cos(\theta_{j,i}))}$.

ECAPA-TDNN

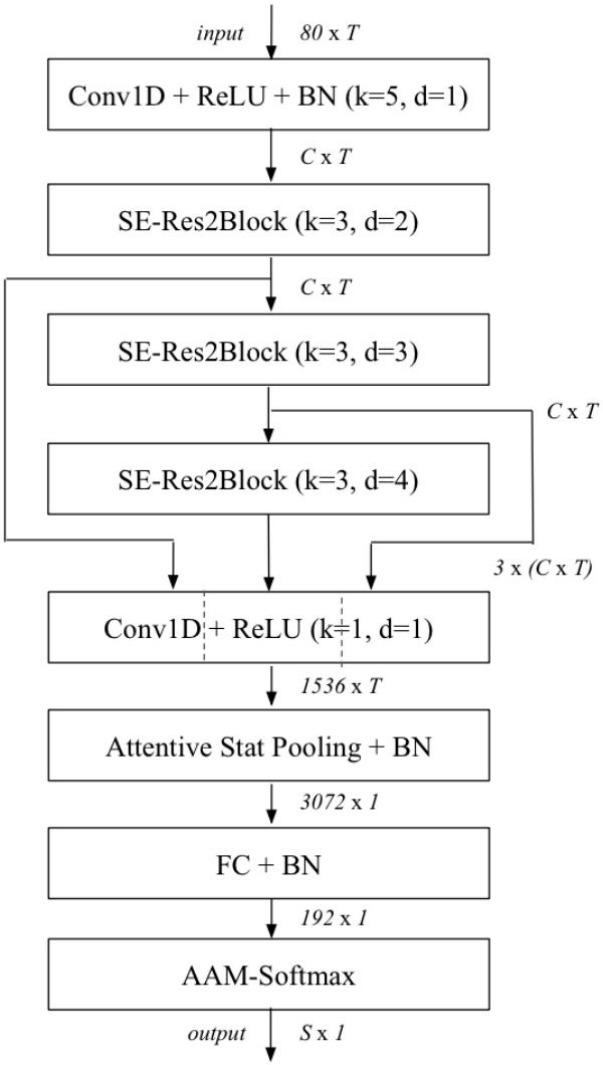
Classic ASV system (s.o.t.a. 2020)

Modern state-of-the-art systems are usually modifications of ECAPA-TDNN

Front-end:

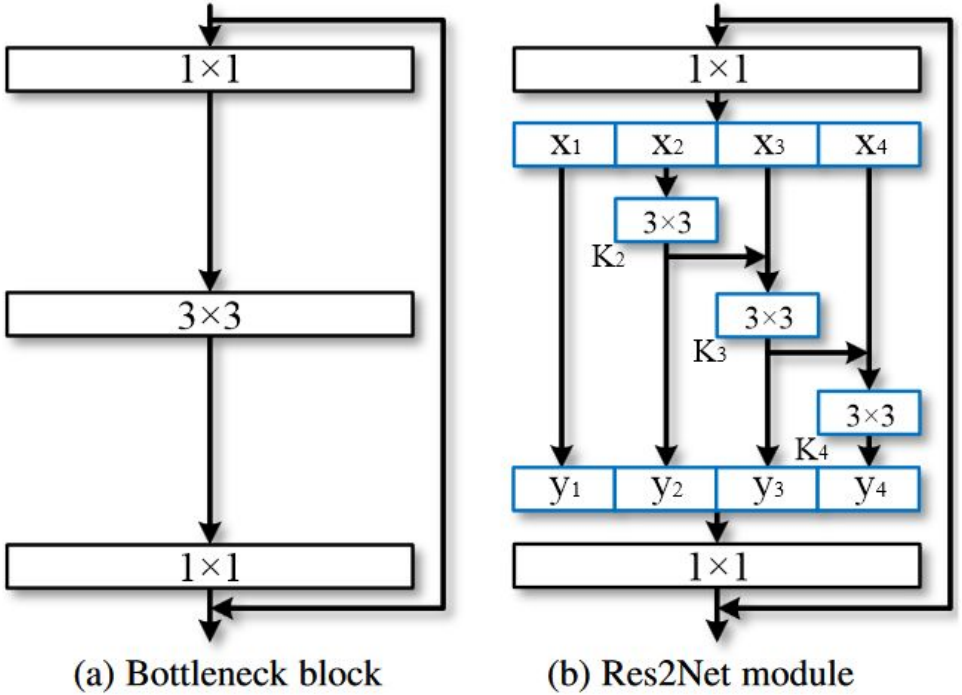
- MFCC (Originally)
- MelSpectrogram (Common)

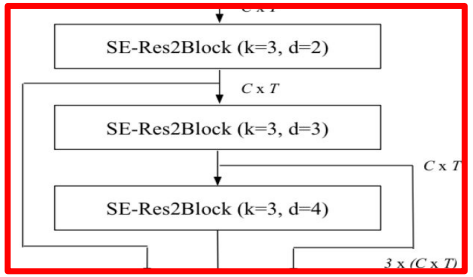
Spectrogram of size FxT is used as a 1-D input of length T with F channels

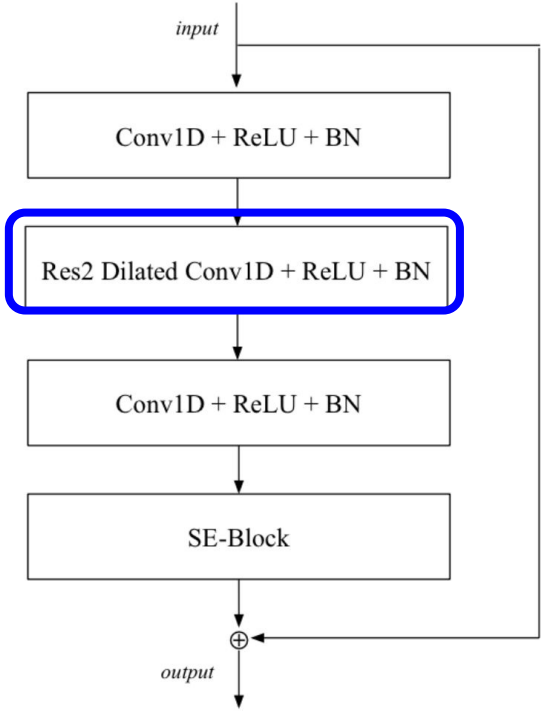


ECAPA-TDNN

Res2Net Block:







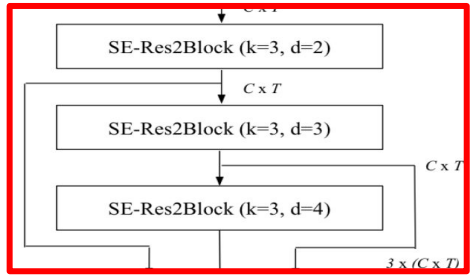
ECAPA-TDNN

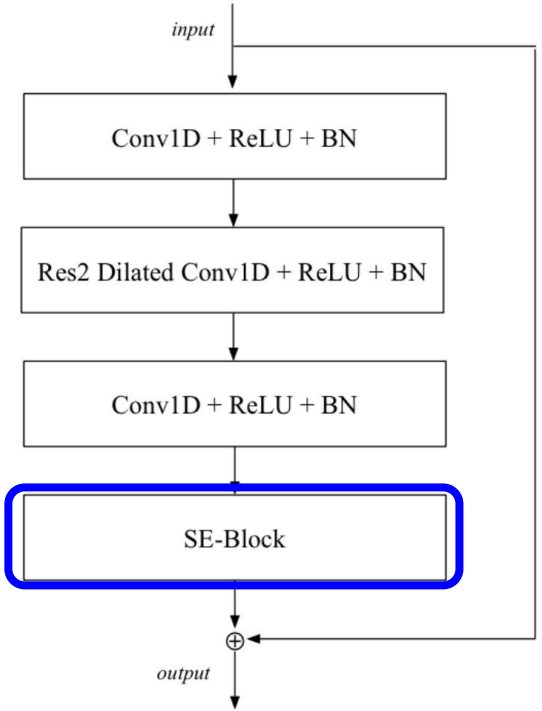
Squeeze-Excitation Block:

1. $$\boldsymbol{z} = \frac{1}{T}\sum_{t}^{T}\boldsymbol{h}_t$$

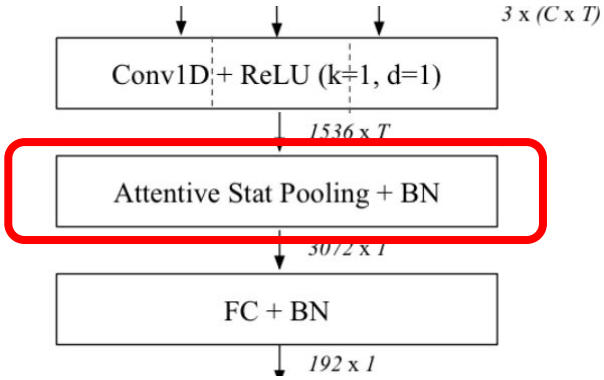
2. $$\boldsymbol{s} = \sigma(\boldsymbol{W}_2 f(\boldsymbol{W}_1 \boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$$

3. $$\tilde{\boldsymbol{h}}_c = s_c \boldsymbol{h}_c$$





ECAPA-TDNN



Attentive Statistic Pooling:

1. $e_{t,c} = \boldsymbol{v}_c^T f(\boldsymbol{W}\boldsymbol{h}_t + \boldsymbol{b}) + k_c$

2. $\alpha_{t,c} = \frac{\exp(e_{t,c})}{\sum_{\tau}^{T} \exp(e_{\tau,c})}$

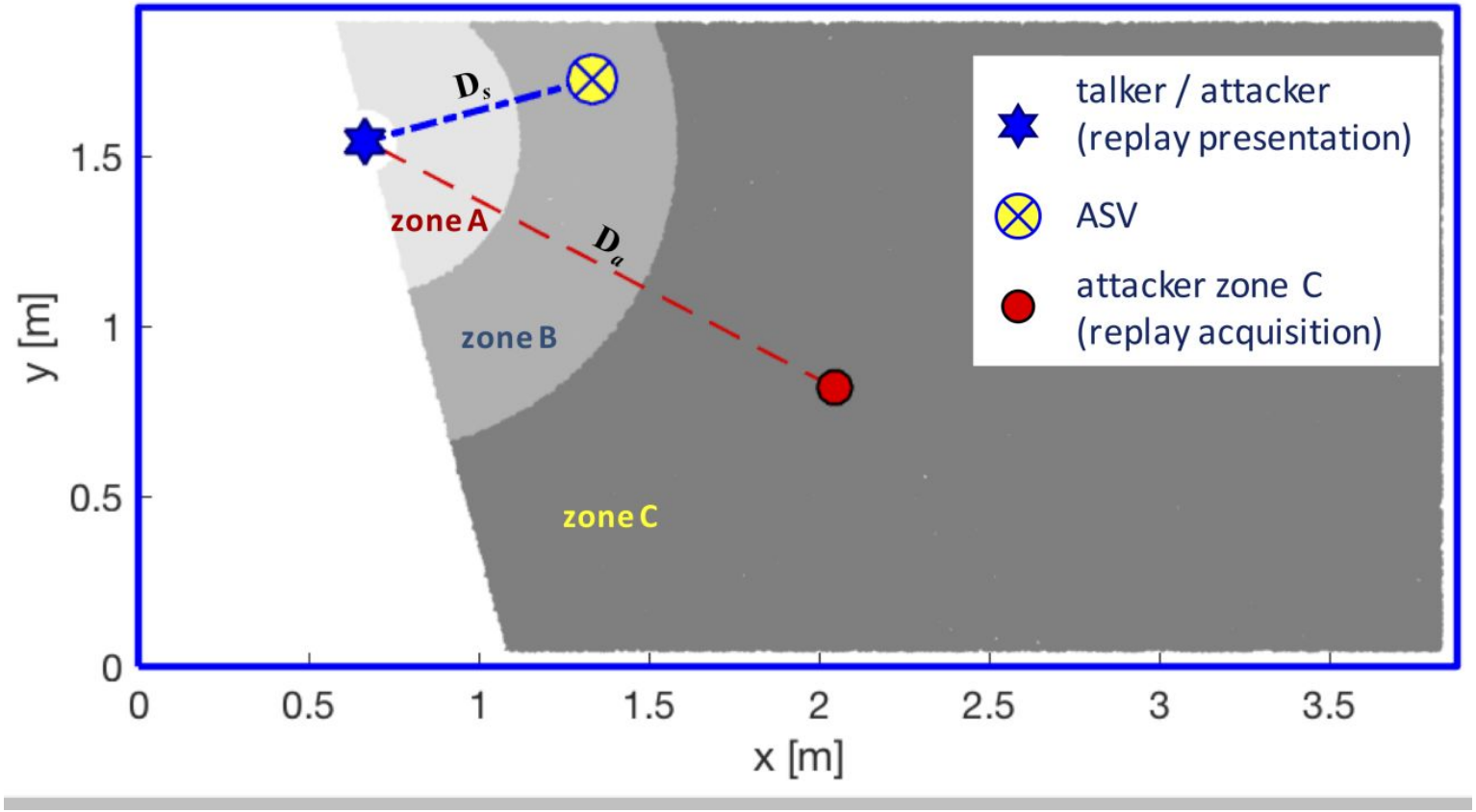
3. $\tilde{\mu}_c = \sum_{t}^{T} \alpha_{t,c} h_{t,c}$ $\tilde{\sigma}_c = \sqrt{\sum_{t}^{T} \alpha_{t,c} h_{t,c}^2 - \tilde{\mu}_c^2}$

Output: Concatenation of mu and sigma

Note: the temporal context is extended by concatenation of global mean and std to each h_t

CM

Recorded Speech, a.k.a. Replay Attack

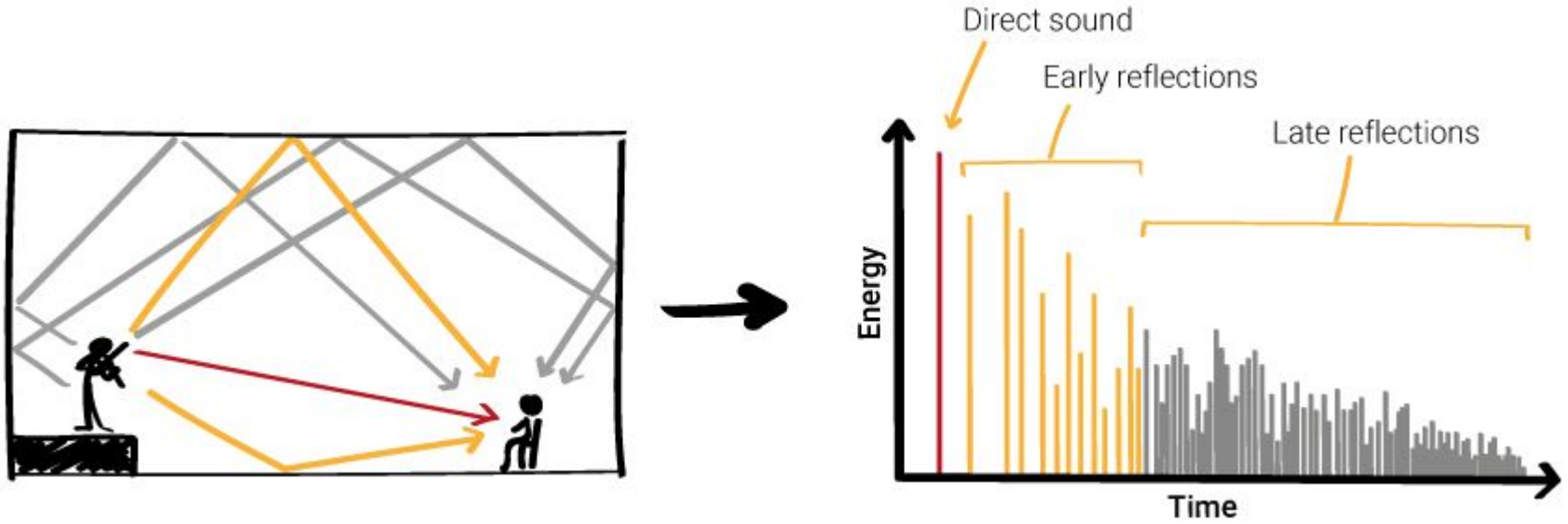


Recorded Speech, a.k.a. Replay Attack

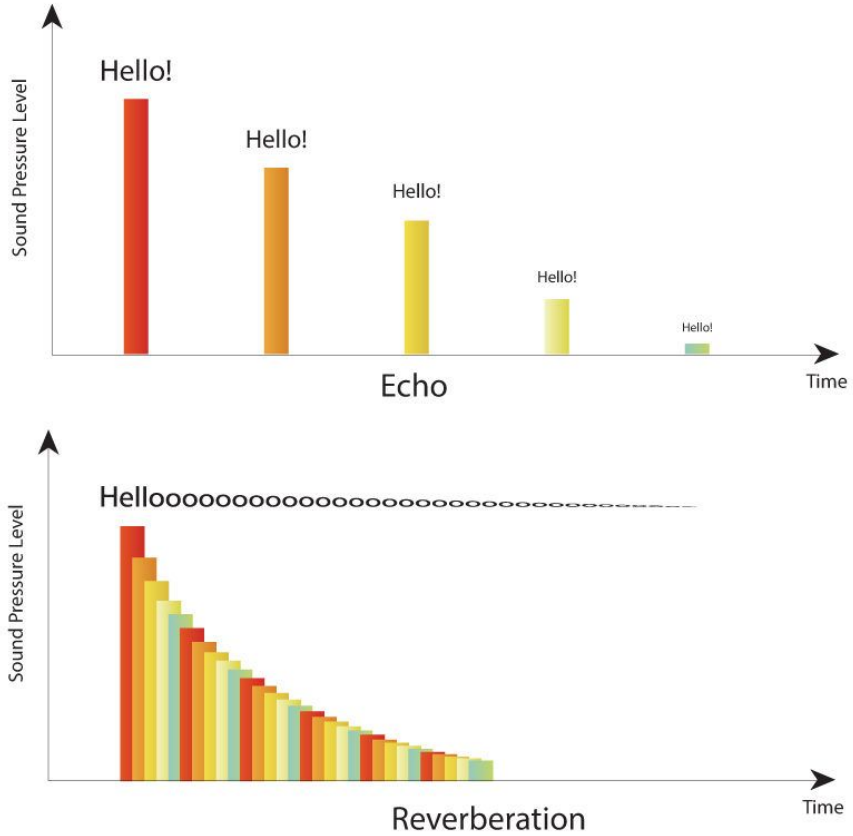
Table 2: Environment is defined as triplet (S,R,D_s), each element of which takes one value in set (a,b,c) as a categorical value.

Environment definition	labels		
	a	b	c
S: Room size (m^2)	2-5	5-10	10-20
R: T60 (ms)	50-200	200-600	600-1000
D_s: Talker-to-ASV distance (cm)	10-50	50-100	100-150

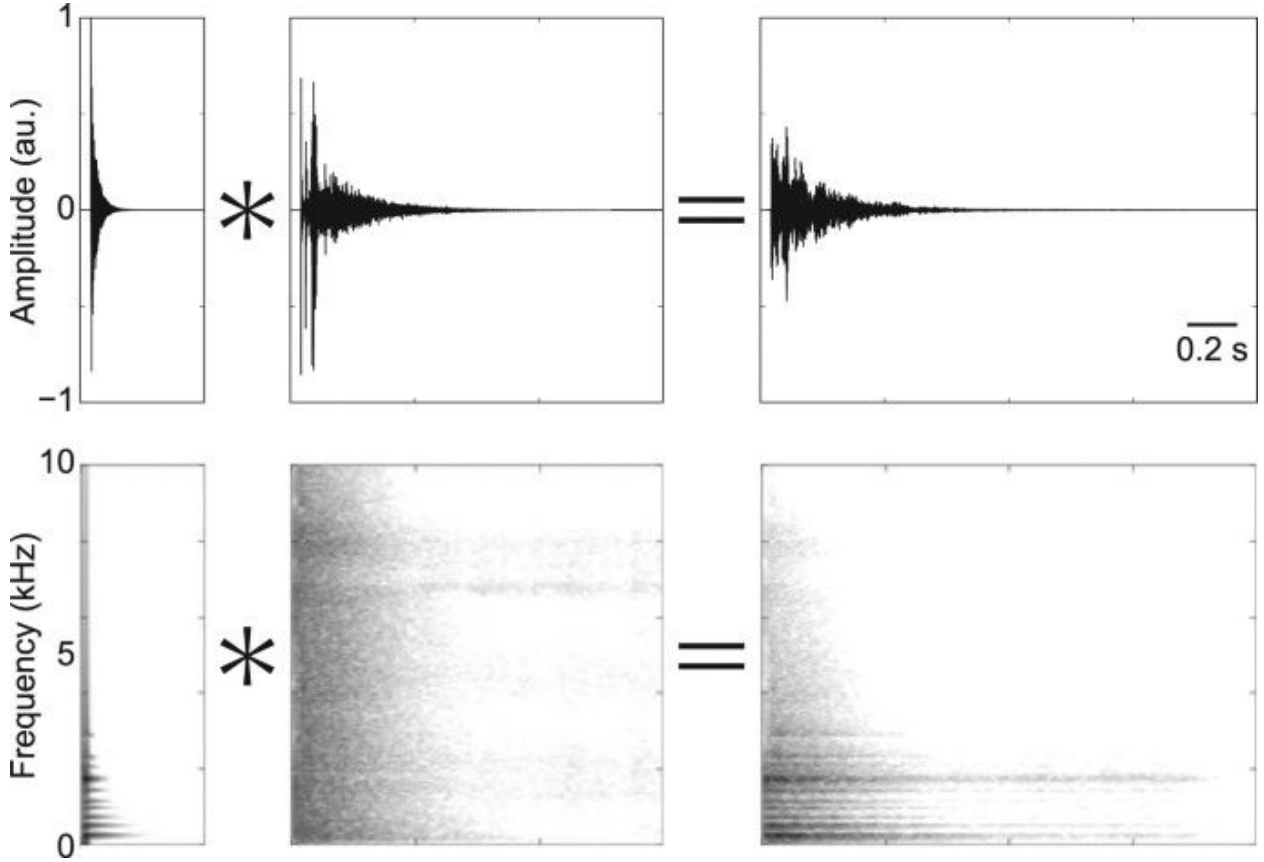
Recorded Speech, a.k.a. Replay Attack



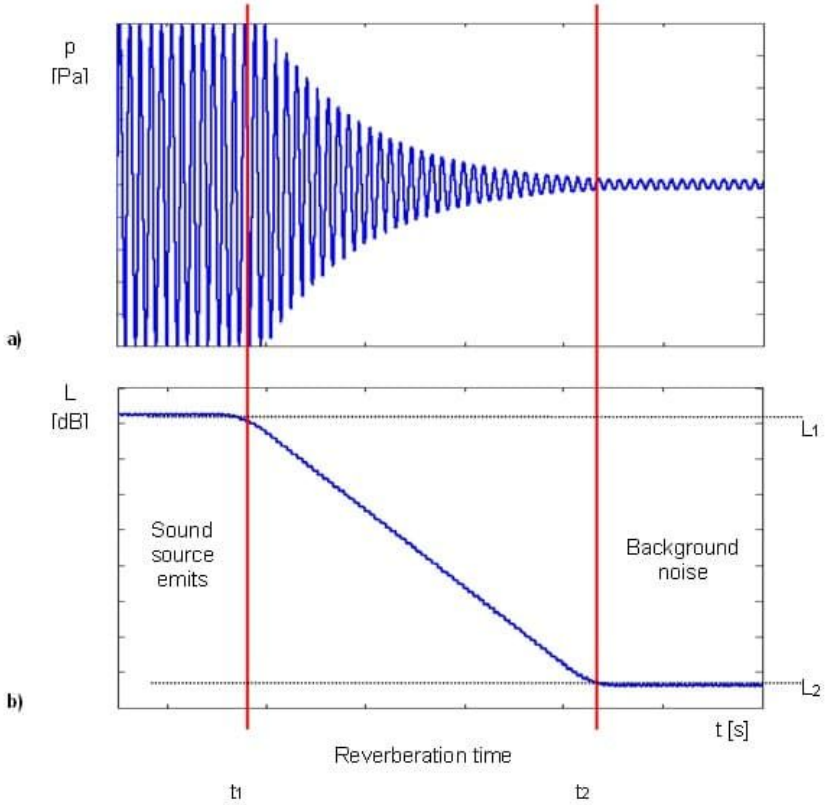
Recorded Speech, a.k.a. Replay Attack



Recorded Speech, a.k.a. Replay Attack



Recorded Speech, a.k.a. Replay Attack



Recorded Speech, a.k.a. Replay Attack

Table 3: Replay attack is defined as duple (D_a,Q), each element of which takes one value in set (A,B,C) as a categorical value.

Attack definition	labels		
	A	B	C
D_a: Attacker-to-talker distance (cm)	10-50	50-100	> 100
Q: Replay device quality	perfect	high	low

Recorded Speech, a.k.a. Replay Attack

Table 4: Definition of replay device quality (Q). OB refers to occupied bandwidth, minF is lower bound of the OB, LNLR is linear-to-non-linear power ratio.

Replay device quality	OB (kHZ)	minF (Hz)	LNLR (dB)
Perfect	inf	0	inf
High	> 10	< 600	> 100
Low	< 10	> 600	< 100

CM: Vocoder based Replay Channel Response Estimation

- Recorded utterance can be expressed as:

$$x(n) = s(n) * h(n) + v(n)$$

- Assuming we have a silent environment (v(n)=0), after STFT:

$$X(k,l) = S(k,l)H(k)$$

- Log-magnitude estimation:

$$log(|X(k,l)|) = log(|S(k,l)|) + log(|H(k)|)$$

CM: Vocoder based Replay Channel Response Estimation

- Filtration through **channel-independent** vocoder:

$$X_{vocoder}(k,l) = S(k,l)H_{vocoder}(k)H_{res}(k)$$

- After log:

$$log(|X_{vocoder}(k,l)|) = log(|S(k,l)|) + log(|H_{vocoder}(k)|) + log(|H_{res}(k)|)$$

CM: Vocoder based Replay Channel Response Estimation

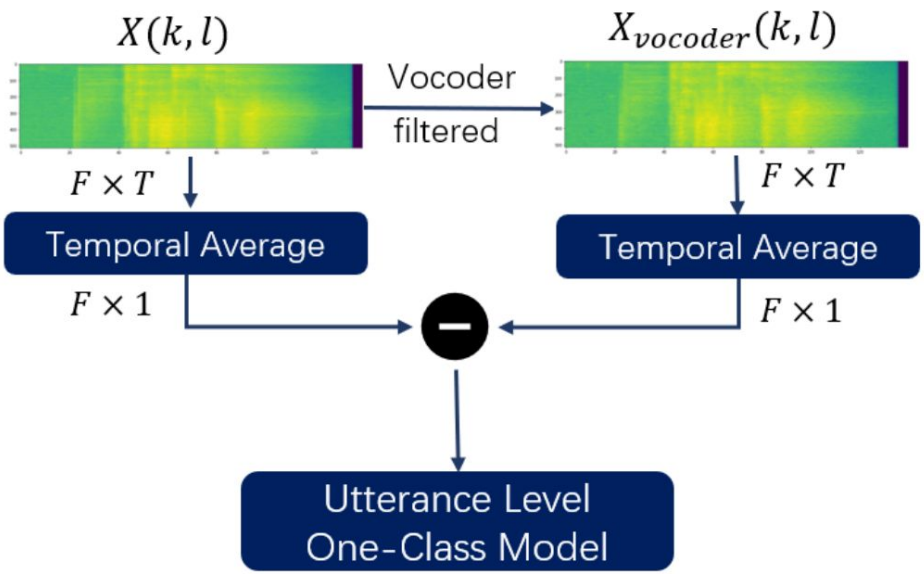
- Then we can define **frame-level** Vocoder-based channel estimation:

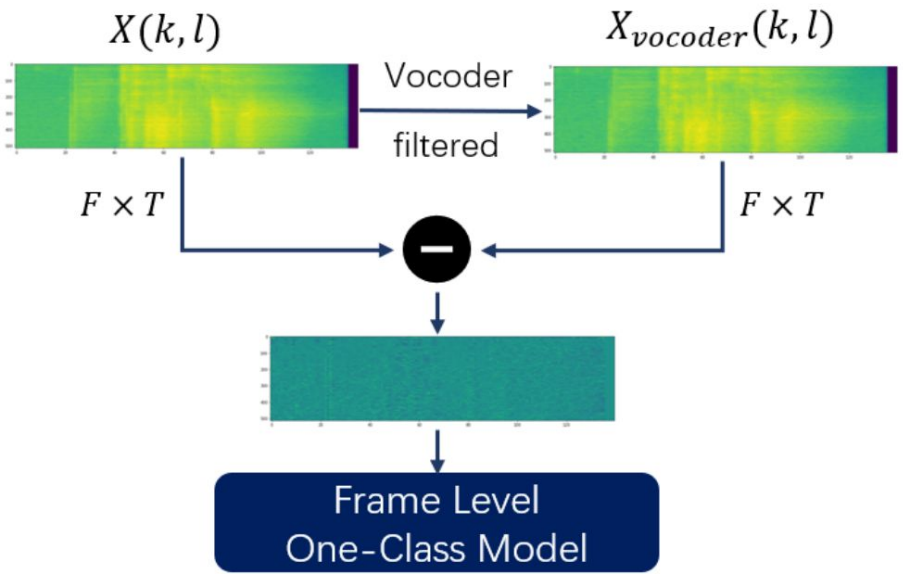
$$log(H_{vr(k,l)}) = log(|X_{vocoder}(k,l)|) - log(|X(k,l)|)$$
$$= log(|H_{vocoder}(k)|) + log(|H_{res}(k)|) - log(|H(k)|)$$

- And **utterance-level** version:

$$log(H_{vr}) = \frac{1}{L}\sum_{l=1}^{L} log(|X_{vocoder}(k,l)|) - \frac{1}{L}\sum_{l=1}^{L} log(|X(k,l)|)$$

CM: Vocoder based Replay Channel Response Estimation





CM: Vocoder based Replay Channel Response Estimation

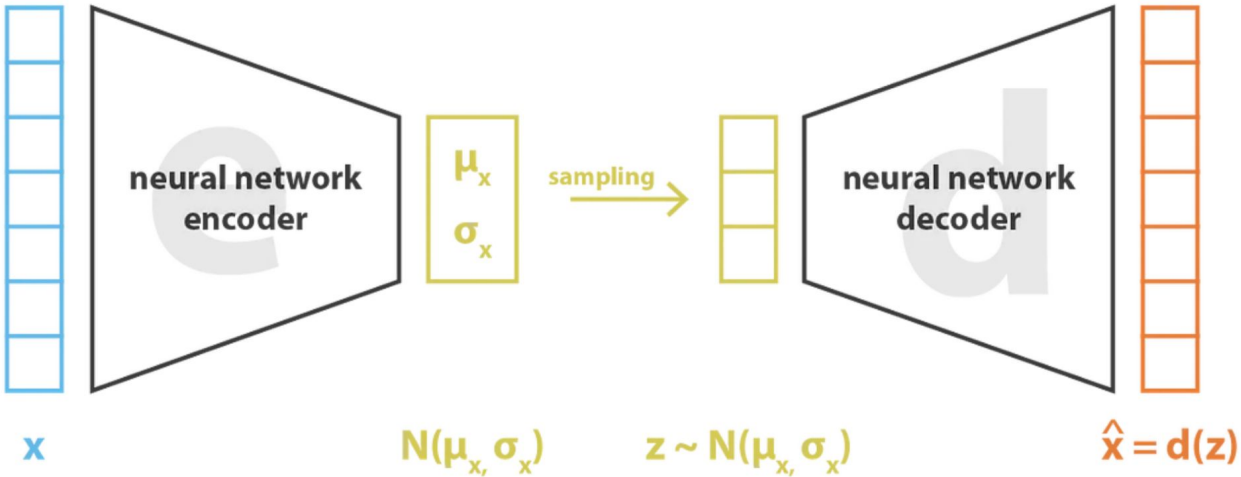
The first back-end is Gaussian Mixture Model (GMM).

Model is trained on bona-fide data via EM-algorithm. Final score is equal to the probability of one-class GMM.

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

CM: Vocoder based Replay Channel Response Estimation

The second back-end is Variational Auto-encoder (VAE).
Final score is equal to the reconstruction probability.



CM: Vocoder based Replay Channel Response Estimation

Algorithm 4 Variational autoencoder based anomaly detection algorithm

INPUT: Normal dataset X, Anomalous dataset $x^{(i)}$ $i = 1, \cdots, N$, threshold α

OUTPUT: reconstruction probability $p_\theta(x|\hat{x})$

$\phi, \theta \leftarrow$ train a variational autoencoder using the normal dataset X

for i=1 **to** N **do**

$\boldsymbol{\mu}_{\boldsymbol{z}^{(i)}}, \boldsymbol{\sigma}_{\boldsymbol{z}^{(i)}} = f_\theta(z|x^{(i)})$

draw L samples from $z \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{z}^{(i)}}, \boldsymbol{\sigma}_{\boldsymbol{z}^{(i)}})$

for l=1 **to** L **do**

$\boldsymbol{\mu}_{\hat{\boldsymbol{x}}^{(i,l)}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{x}}^{(i,l)}} = g_\phi(x|z^{(i,l)})$

end for

$reconstruction\ probability(i) = \frac{1}{L}\sum_{l=1}^{L} p_\theta(x^{(i)}|\boldsymbol{\mu}_{\hat{\boldsymbol{x}}^{(i,l)}}, \boldsymbol{\sigma}_{\hat{\boldsymbol{x}}^{(i,l)}})$

if $reconstruction\ probability(i) < \alpha$ **then**

$x^{(i)}$ is an anomaly

else

$x^{(i)}$ is not an anomaly

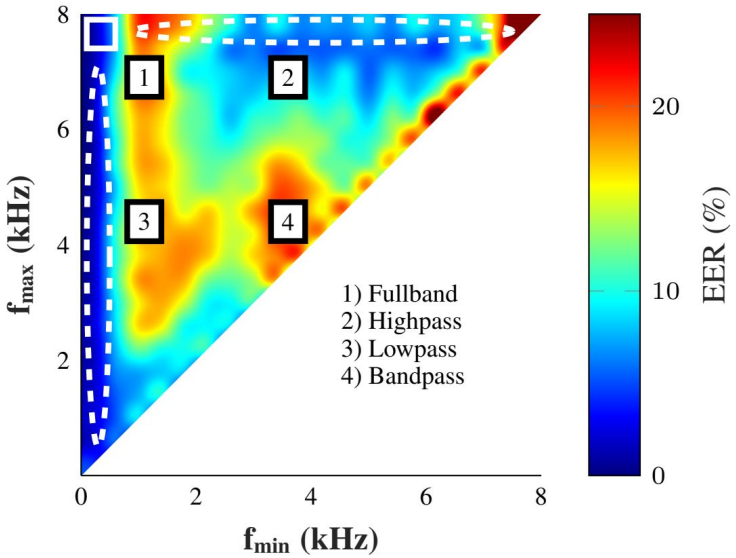
end if

end for

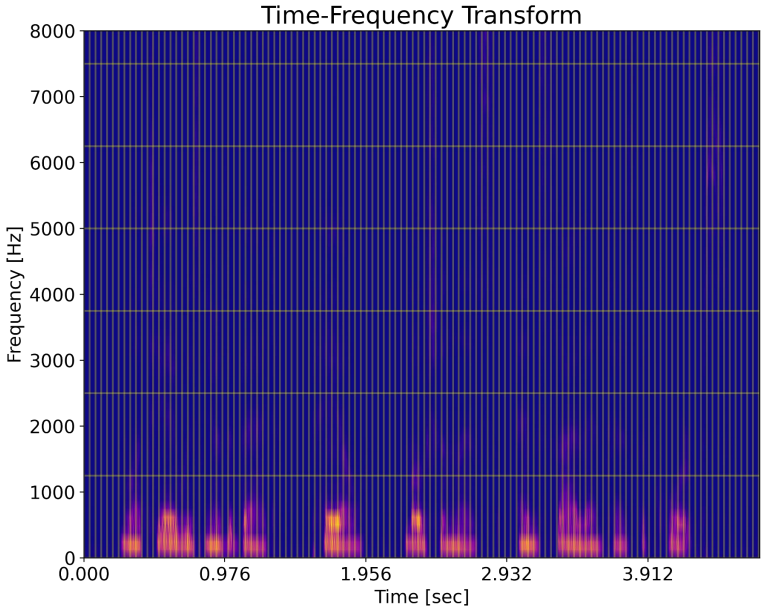
Synthesized Speech

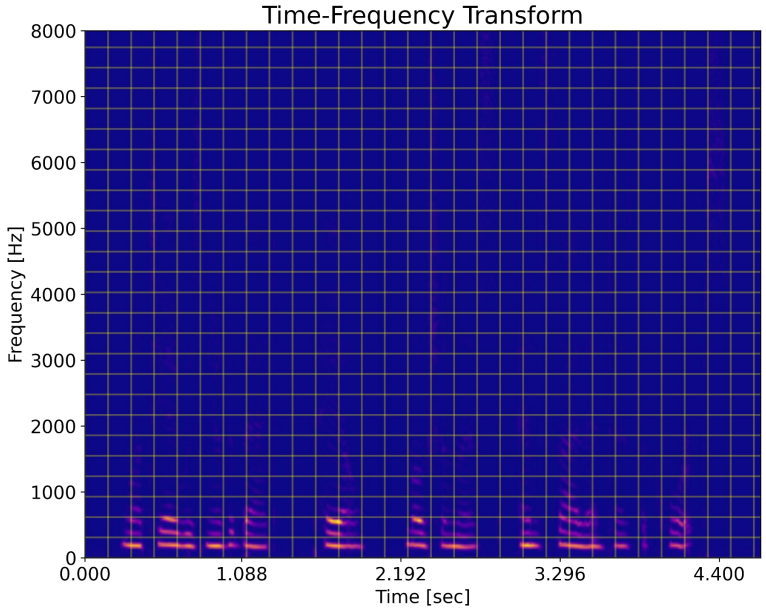
Synthesized speech is an utterance created via Text-To-Speech, Voice Conversion or both.

- Artefacts lay in sub-bands, not in the full-band
- Different front-ends emphasis information at different sub-bands

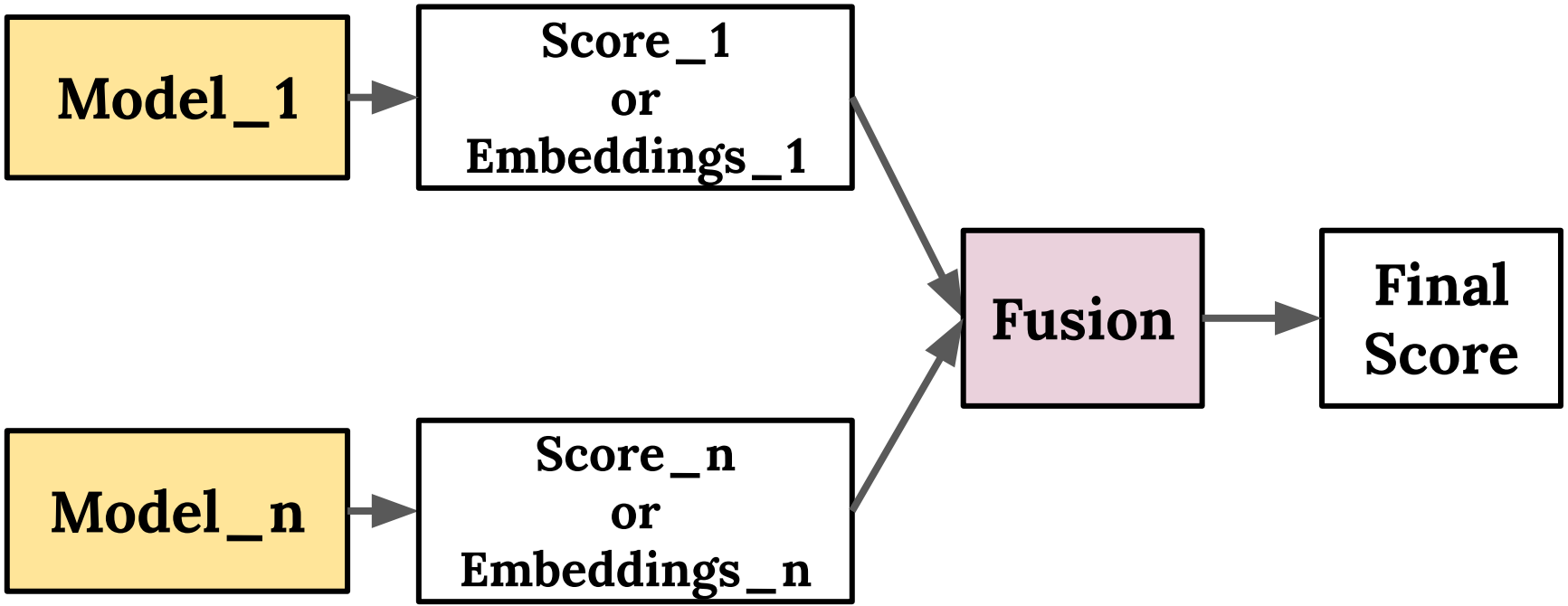


Synthesized Speech





Fusion, a.k.a Ensemble

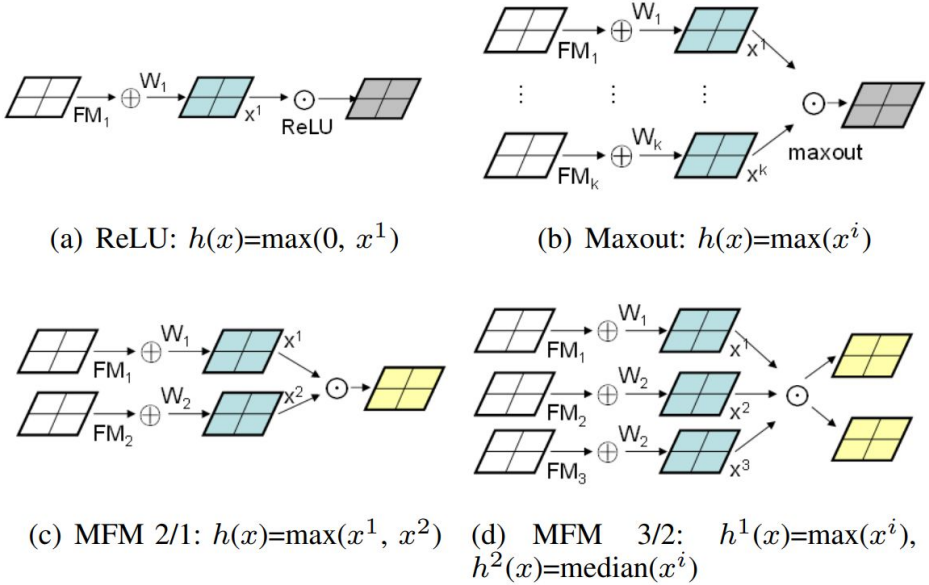


CM: LightCNN

Classic CNN with Max-Feature-Map activation and optional Network-In-Network layers (Linear layer instead of Convolution).

Front-End:

- LFCC
- MFCC
- CQCC
- STFT
- etc.



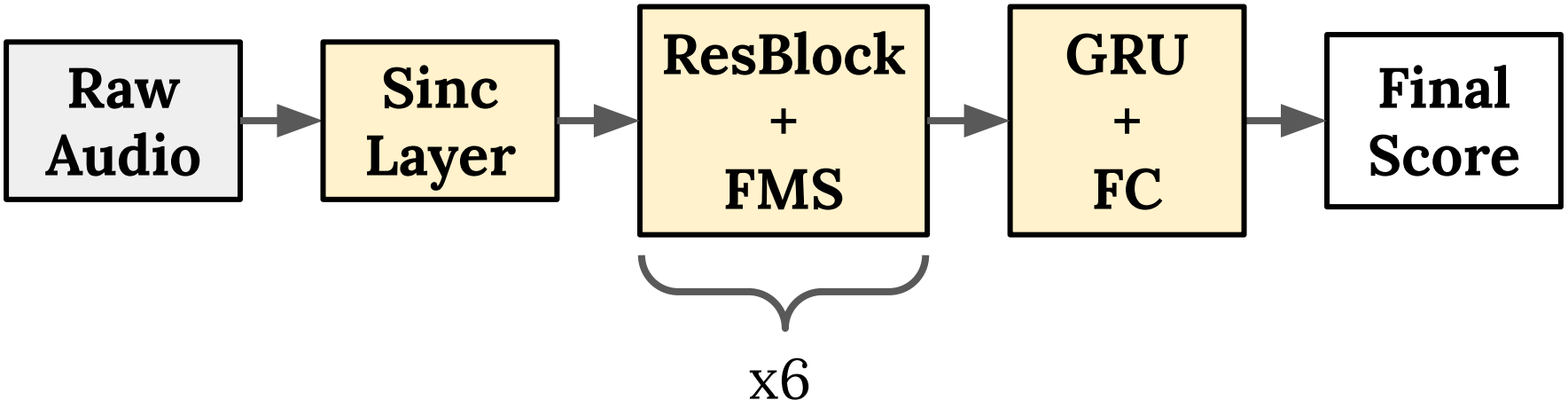
(a) ReLU: $h(x)$=max(0, x^1)

(b) Maxout: $h(x)$=max(x^i)

(c) MFM 2/1: $h(x)$=max(x^1, x^2)

(d) MFM 3/2: $h^1(x)$=max(x^i), $h^2(x)$=median(x^i)

CM: RawNet2



Sinc-Layer

Conventional Convolution-Layer with L filters:

$$y[n] = x[n] * h[n] = \sum_{l=0}^{L-1} x[l]h[n-l]$$

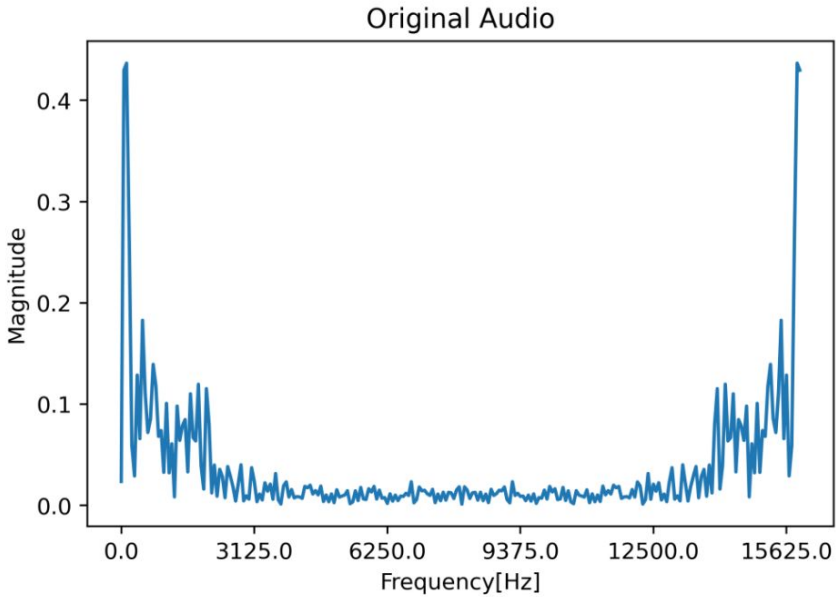
In Sinc-layer, we take the following g[n] instead of h[n]:

$$g[n, f_{max}, f_{min}] = 2f_{max}\text{sinc}(2\pi f_{max} n) - 2f_{min}\text{sinc}(2\pi f_{min} n)$$

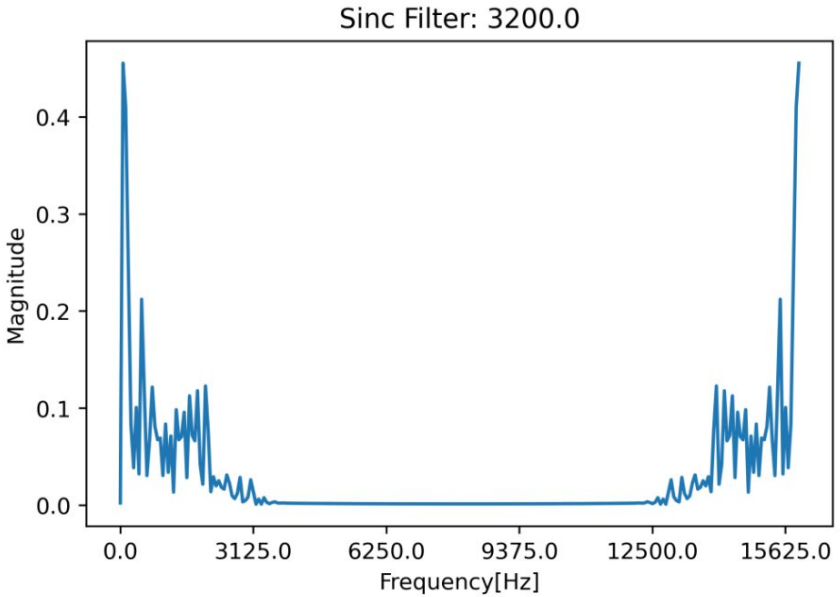
Note: g[n] is smoothed via Hamming window (g[n] = g[n] · w[n])

Filters Example

Sinc-Layer

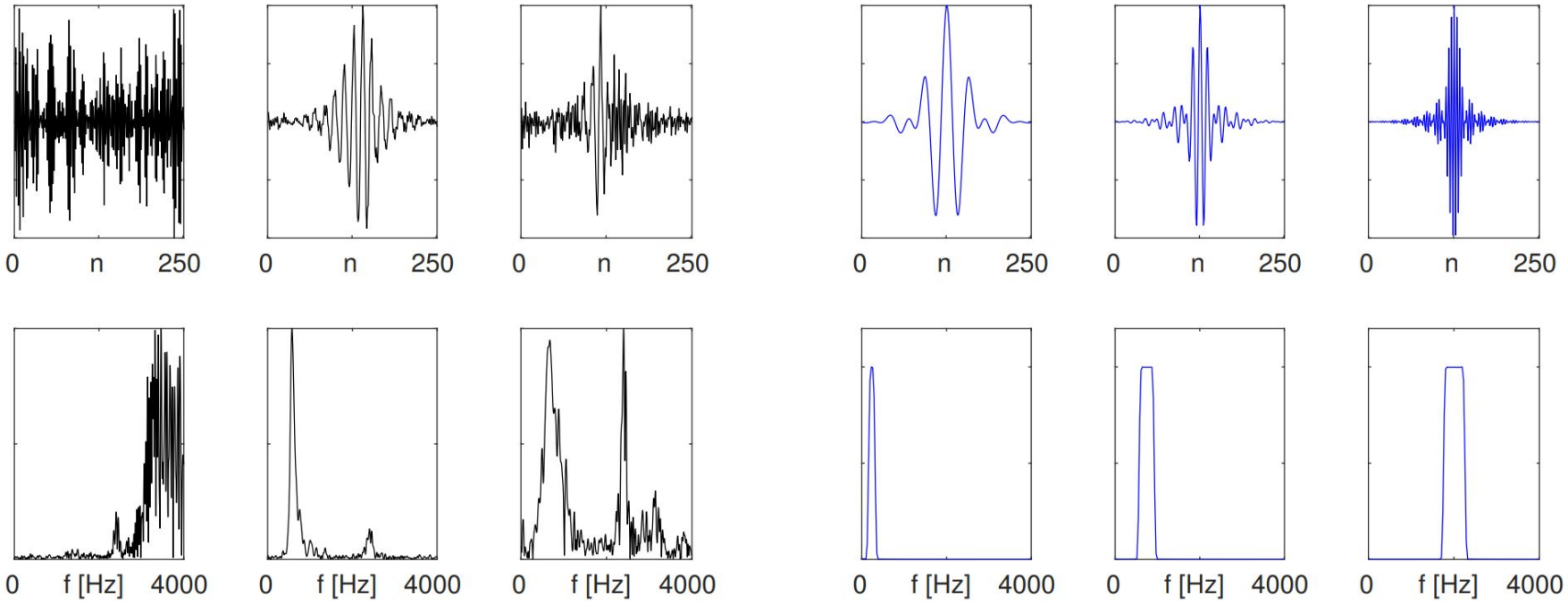


$x[n]$



$x[n] * \text{filter}[n]$

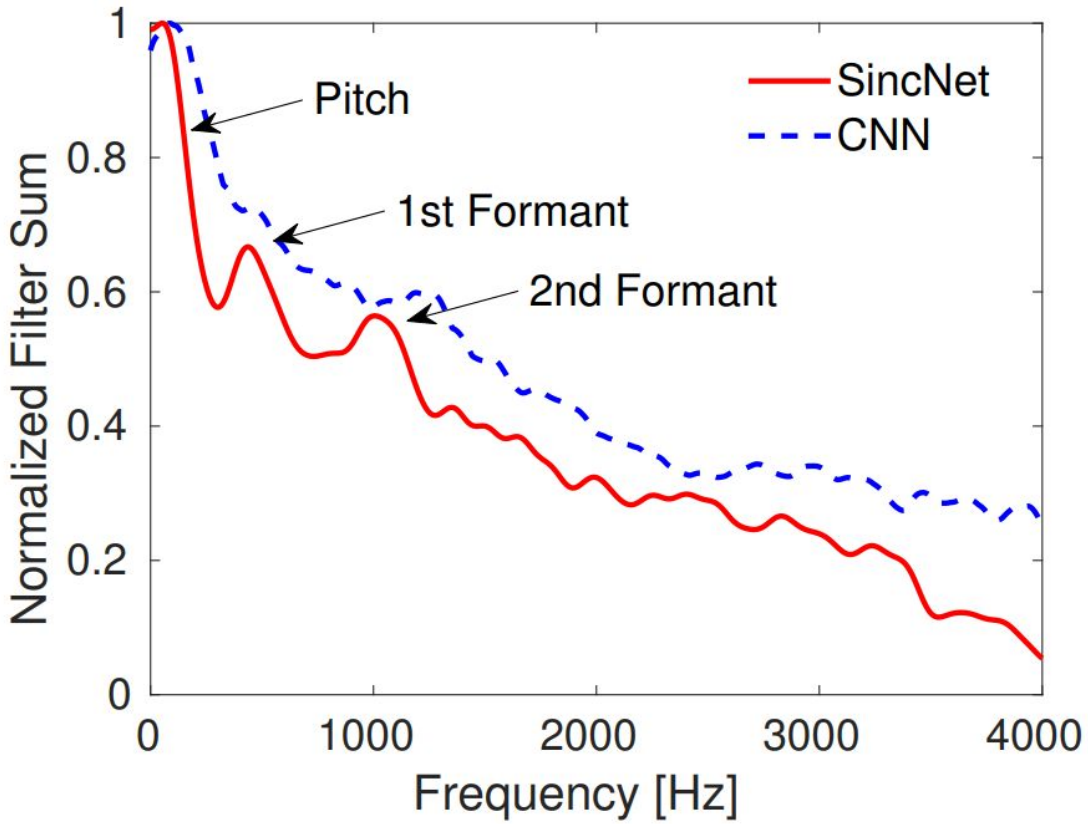
Sinc-Layer



(a) CNN Filters

(b) SincNet Filters

Sinc-Layer

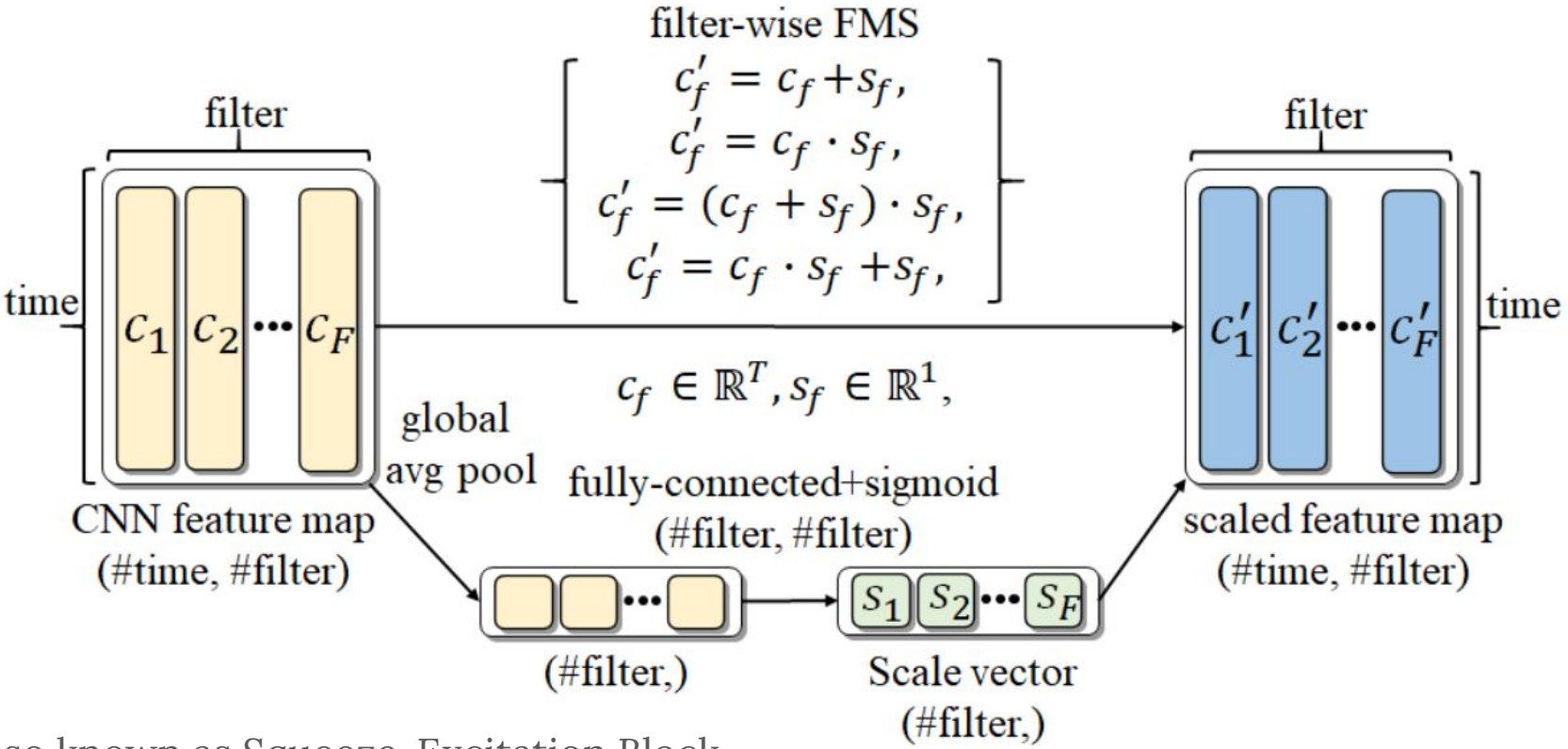

Normalized Filter Sum
Frequency [Hz]
SincNet
CNN
Pitch
1st Formant
2nd Formant
0
0.2
0.4
0.6
0.8
1
0
1000
2000
3000
4000

Sinc-Layer

Sinc-Layer advantages:

- 2 parameters per filter instead of L
- Physical interpretation
- Better convergence speed and learnt features

Feature-Map-Scaling (FMS)*



*Also known as Squeeze-Excitation Block

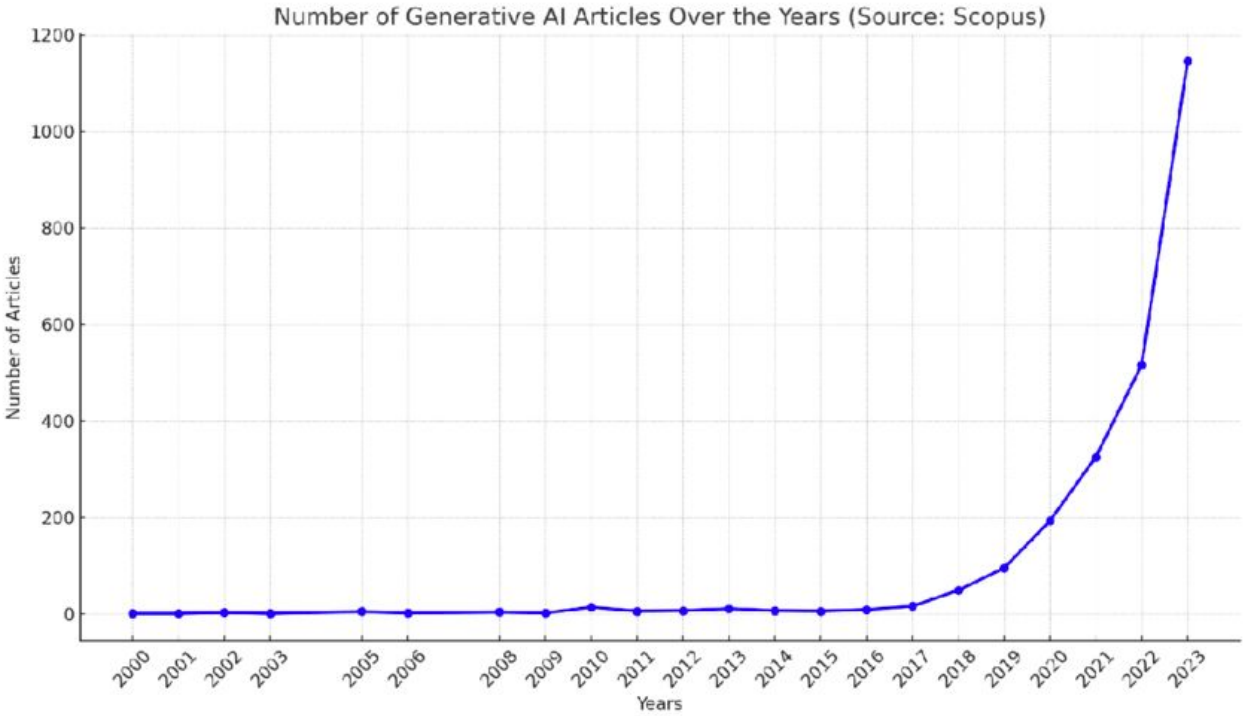
SASV

SASV Methods:

- Fusion of CM and ASV (Weighted Sum, ML models, Product, FiLM). Use either scores or embeddings.
- Cascade approach: one model after another.
- End-to-End solution.

Generalization
And
Video Modality

Main problem in Deepfake Detection



Generative artificial intelligence in education: Advancing adaptive and personalized learning (Acta Informatica Pragensia 2024)

Main problem in Deepfake Detection

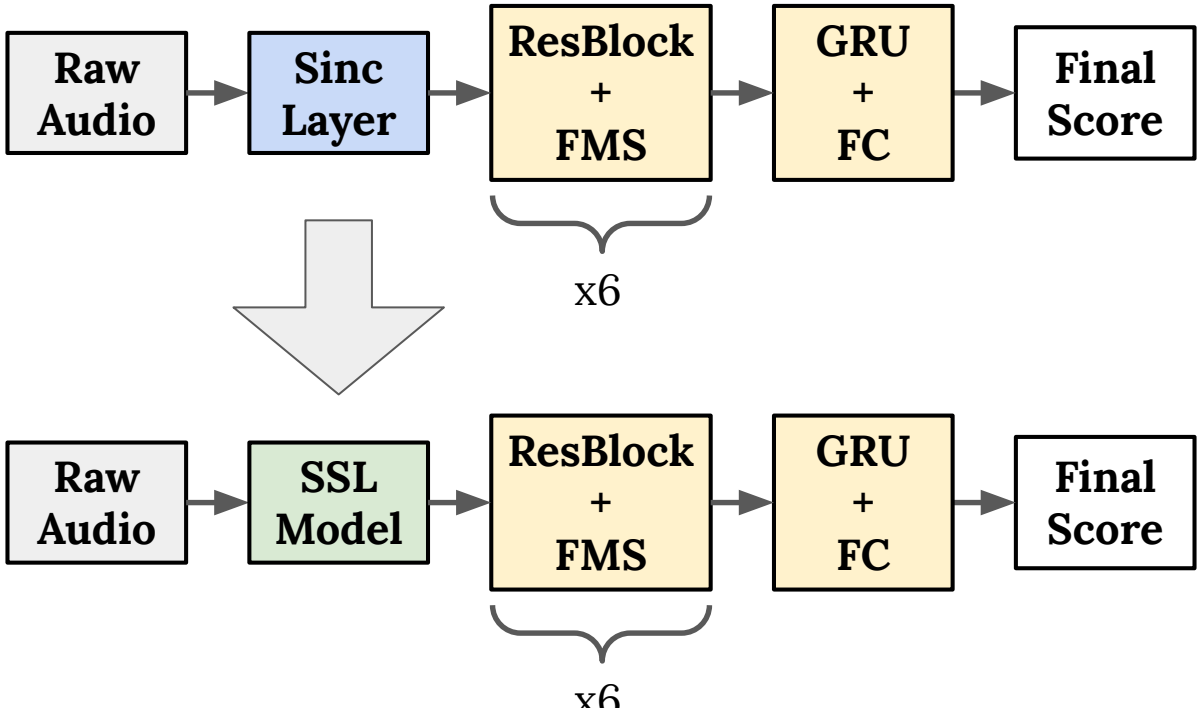
Generative AI is more accessible and of higher quality → more deepfakes and more variety of them

Deepfake Detector needs to generalize to unseen attacks (algorithms). We need:

- More data? Augmentation?
- Larger detectors?
- Pretrained models?

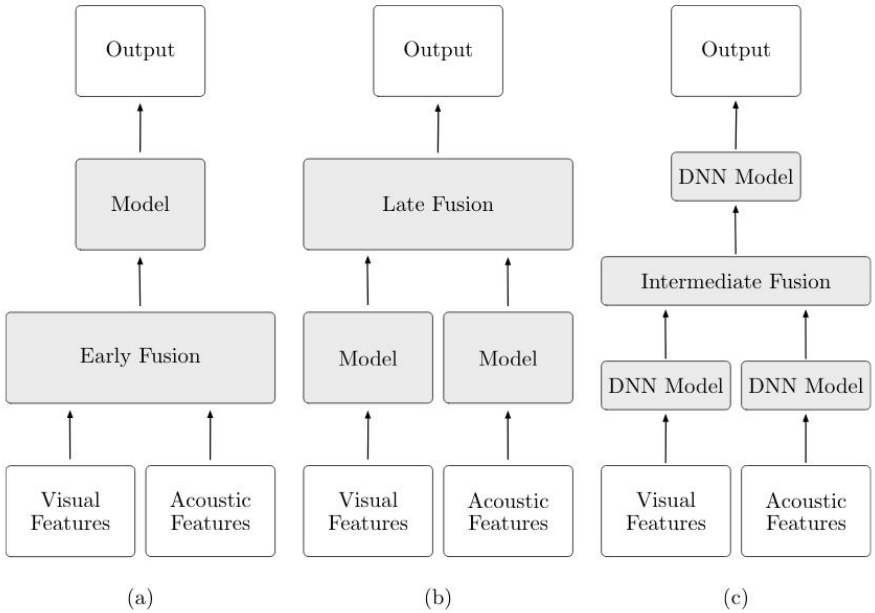
Pretraining to achieve generalization

One common way to enhance generalization is to use pretrained SSL models



What about video or audio video?

Solving just video is more or less the same as just audio. For audiovisual:



An Overview of Deep-Learning-Based Audio-Visual Speech Enhancement and Separation (TASLP 2021)

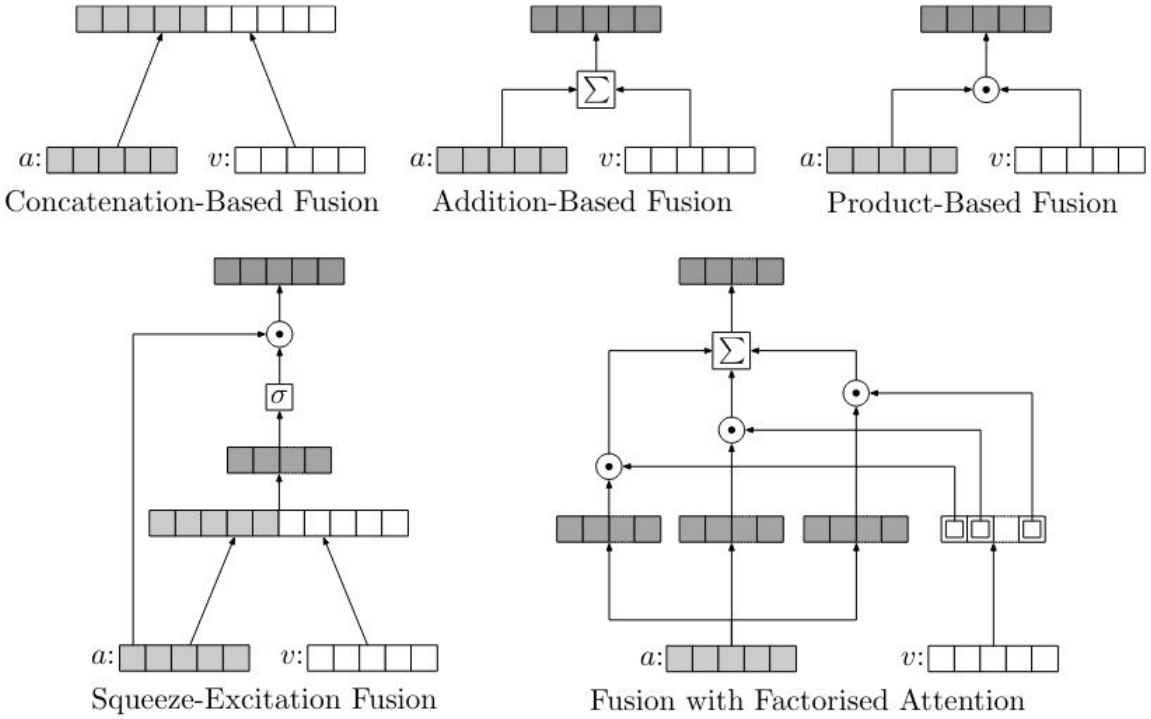
1) Early fusion: one model for both modalities

Advantage: That's how the brain works
Disadvantage: need to synchronize modalities, upsample, downsample, etc.

2) Late fusion: pre-process features with different models

Advantage: can use SOTA methods for both modalities as pre-processing step. Easier to fuse
Disadvantage: less connection between modalities and how the brain works

Fusion Methods



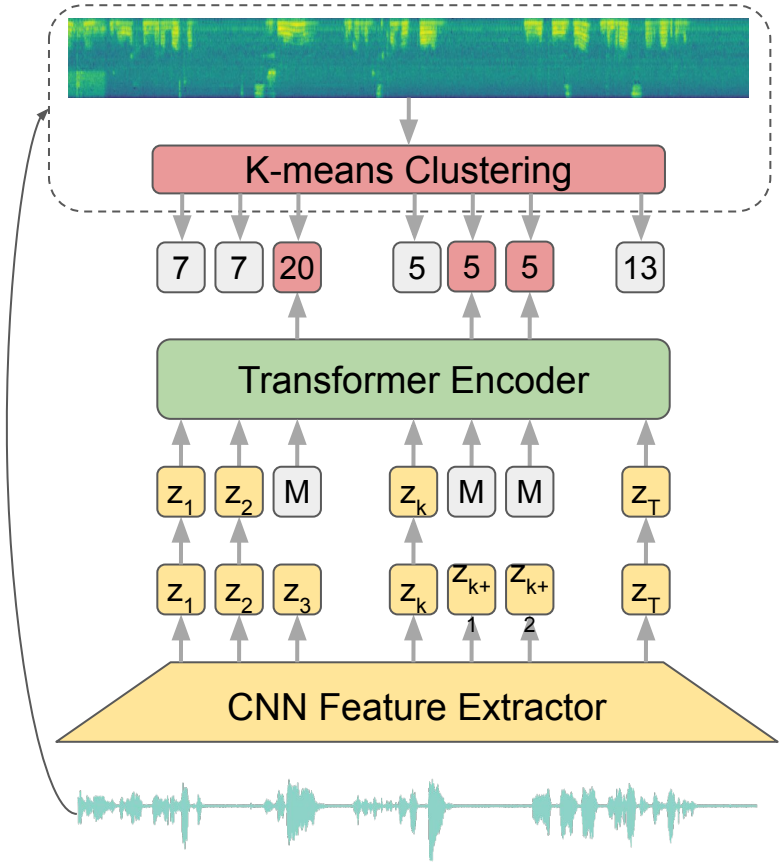
An Overview of Deep-Learning-Based Audio-Visual Speech Enhancement and Separation (TASLP 2021)

SSL Models for AV:

Self-supervised models can be extended to audio-visual domain. Let's recall how the Audio-HuBERT looked like:

Masked modeling using K-means on features:

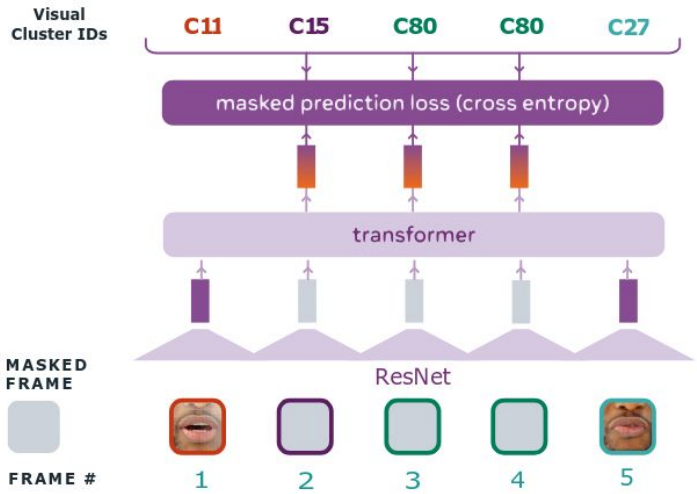
1. Start with MFCC for audio
2. Repeat with intermediate representations of HuBERT itself



HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units (TASLP 2021)

Visual HuBERT

HuBERT can be defined for visual domain, where instead of MFCC some other pre-processing can be done for the initial step (e.g. HoG)



(a) Single-modal visual HuBERT

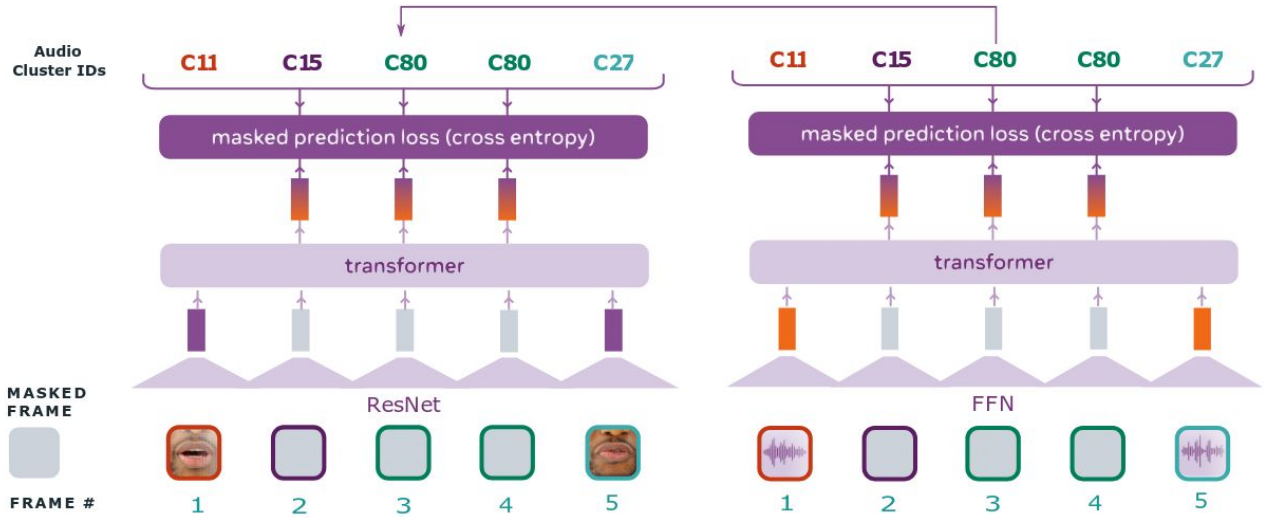
[Learning audio-visual speech representation by masked multimodal cluster prediction (ICLR 2022)]

Cross-modal Visual HuBERT

We can train visual and audio HuBERT simultaneously using audio features (MFCC and intermediate representations)

Alternating training:

1. Train visual model using audio features
2. Refine audio model using its features



(b) Cross-modal visual HuBERT

[Learning audio-visual speech representation by masked multimodal cluster prediction (ICLR 2022)]

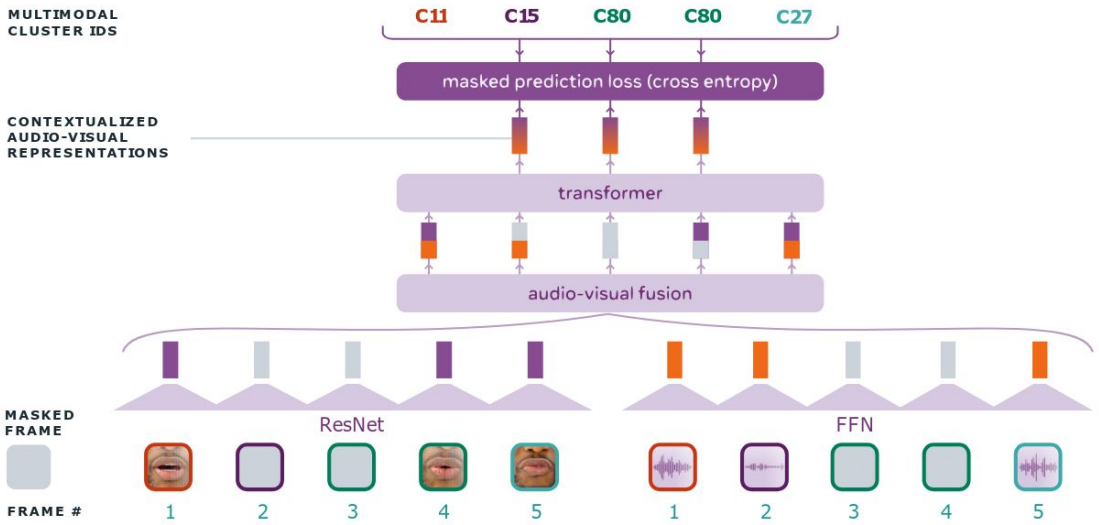
AV-HuBERT

Finally, we can improve modelling by using audio-visual fusion:

Tricks:

1. Modality dropout
2. AV clustering instead of AO (after 1st round)
3. Improved masking with “fake” frames
4. Different masking rates

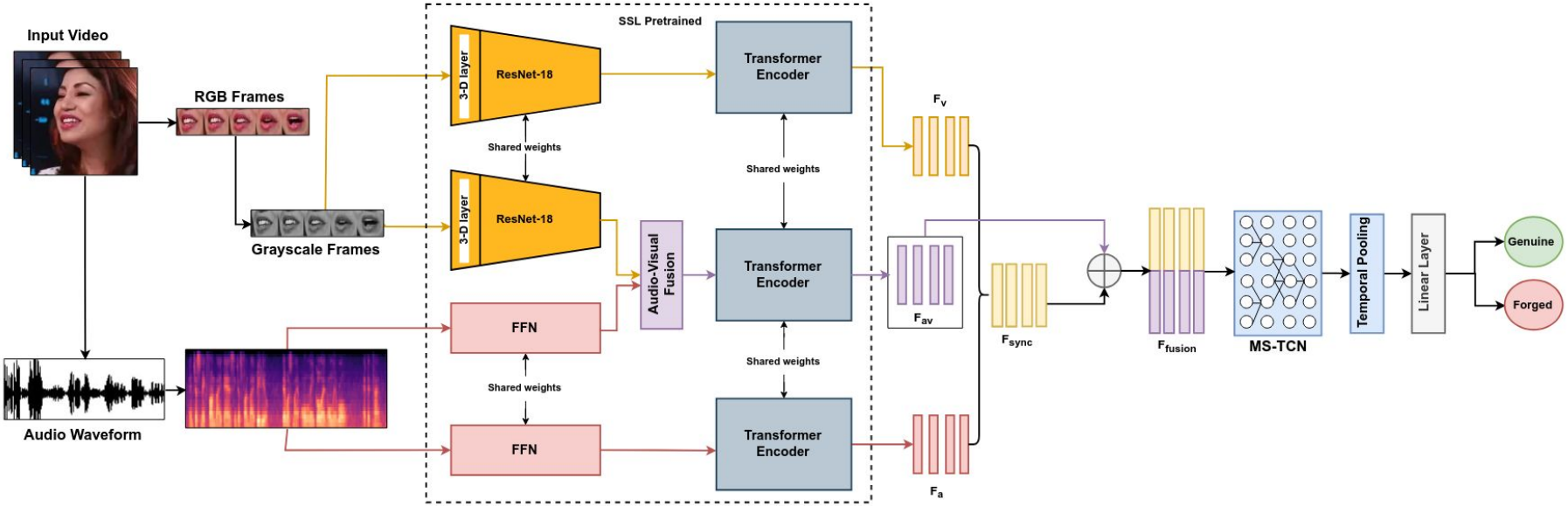
Trying to avoid audio domination



(c) Audio-visual HuBERT (proposed)

[Learning audio-visual speech representation by masked multimodal cluster prediction (ICLR 2022)]

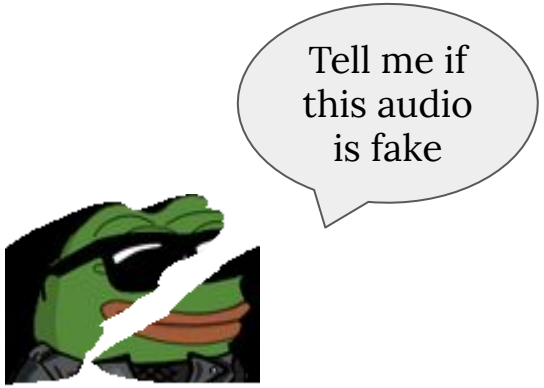
AV-HuBERT for Deepfake Detection



[AV-Lip-Sync+: Leveraging AV-HuBERT to exploit multimodal inconsistency for video deepfake detection (2023)]

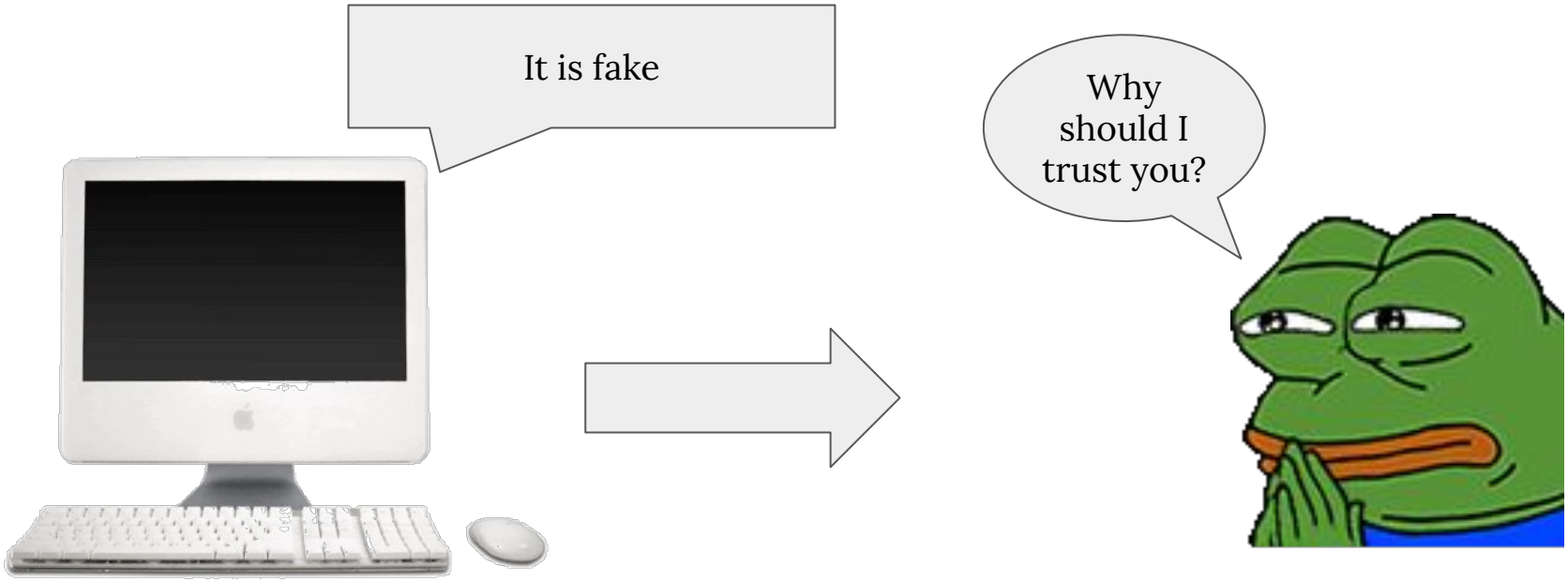
Explainable AI (XAI) / Model Interpretation

Why do we care?



Our SASV system

Why do we care?



Our SASV system

Why do we care?

- Why model made such a decision?
- What is its decision-making process?
- Does it actually understand what is a deepfake?
- Can we find its edge-cases and improve the model?
- If model makes a mistake, can we find why it did it?

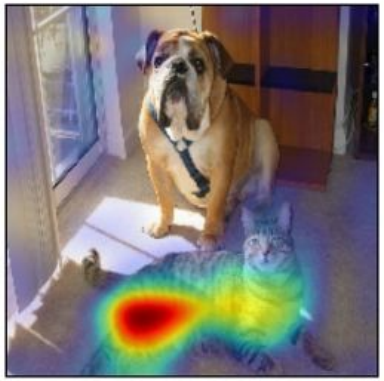
Enable Trust

Improve the model

Why do we care?

- Clear example for CV

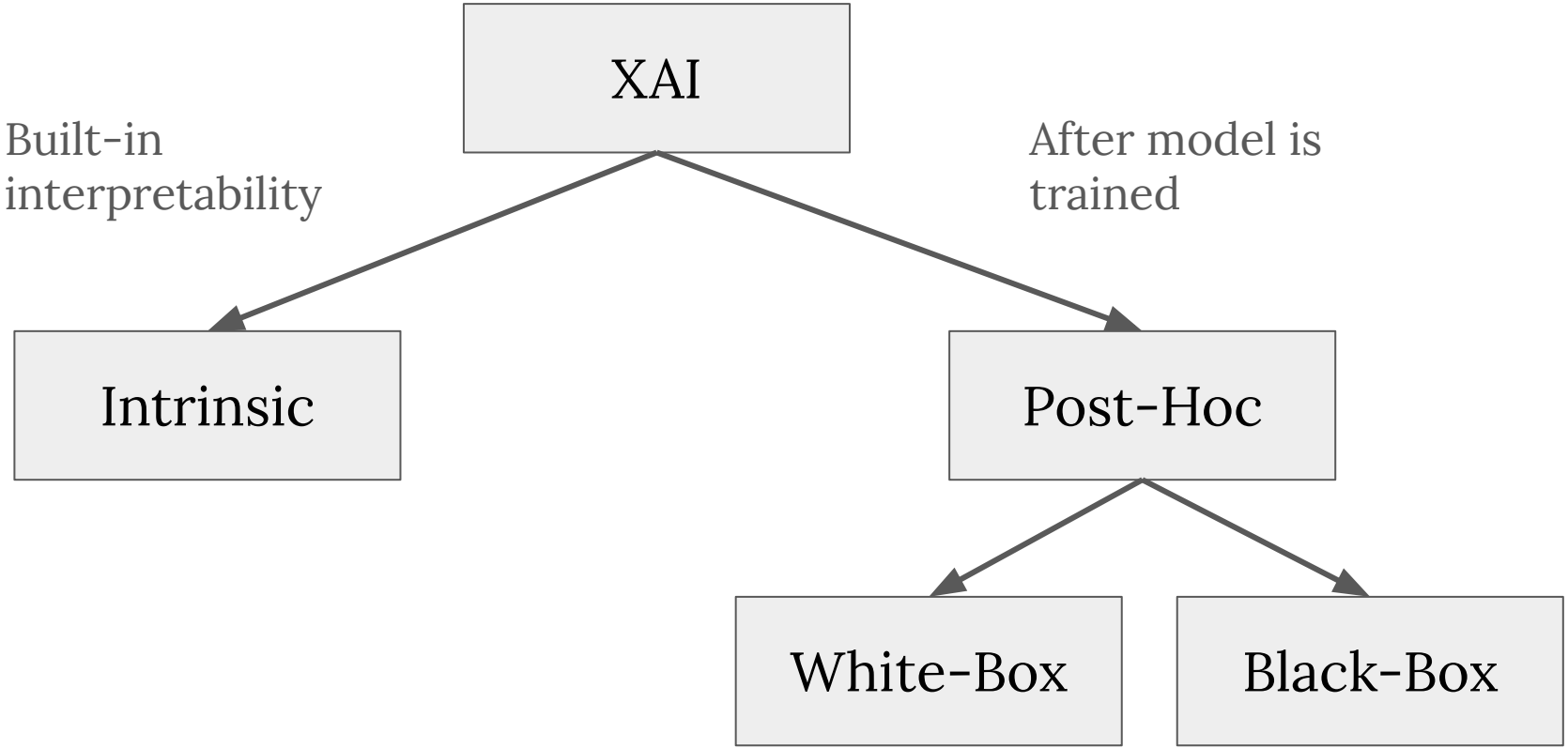
For good classifiers, XAI can also be used for weakly-supervised localization (i.e. localize object given only its label) and segmentation



(c) Grad-CAM ‘Cat’

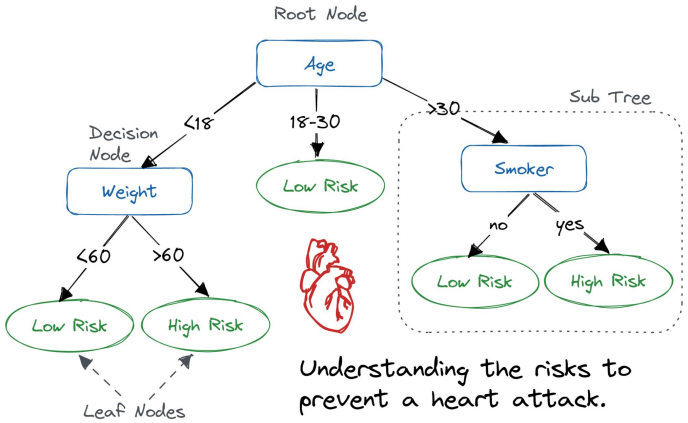
(i) Grad-CAM ‘Dog’

XAI Types


XAI
Built-in interpretability
After model is trained
Intrinsic
Post-Hoc
White-Box
Black-Box

Intrinsic XAI

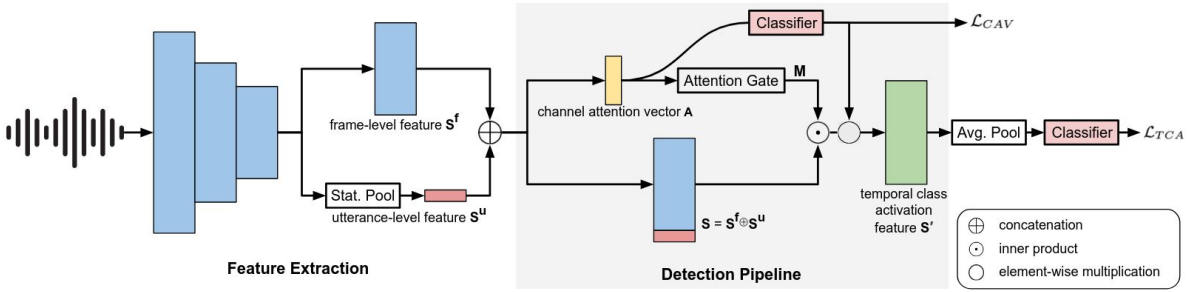
- Decision Trees (Feature > or < than a threshold – understandable)
- Linear Regression (Feature with specific weight – understandable)
- Loss-based interpretability*

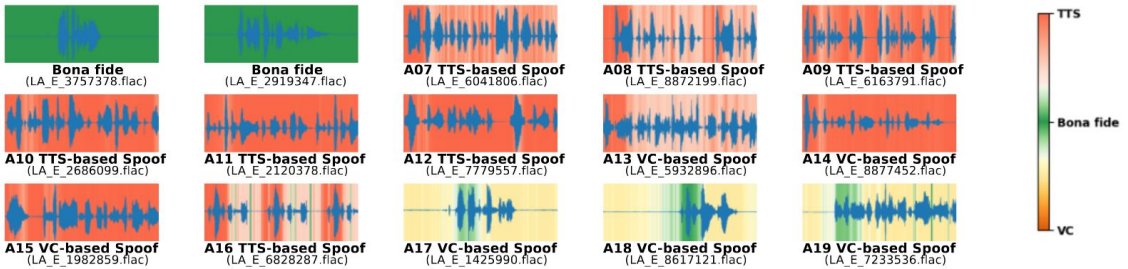


*Less trustworthy than architecture-based interpretability

Intrinsic XAI for Deepfake Detection

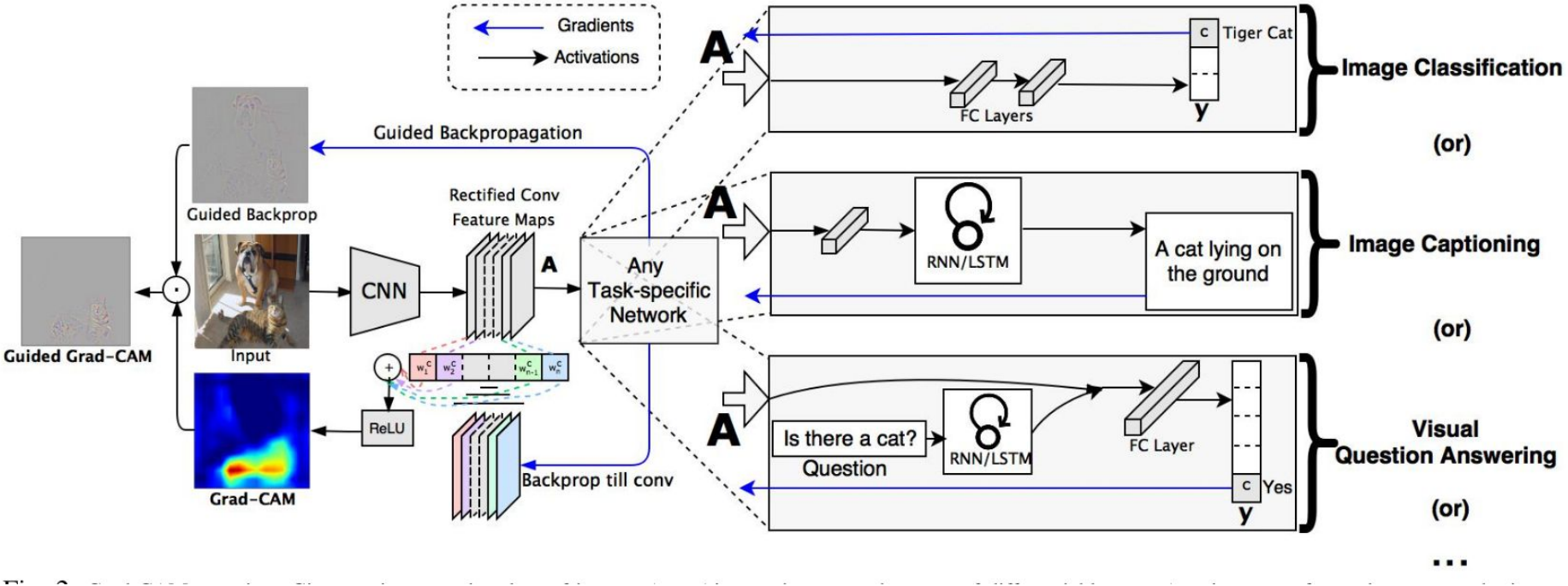
- Loss-based interpretability*





Interpretable Temporal Class Activation Representation for Audio Spoofing Detection (Interspeech 2024)

White-box Post-Hoc: Gradient-based methods



Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization (ICCV 17)

Black-box Post-Hoc: LIME

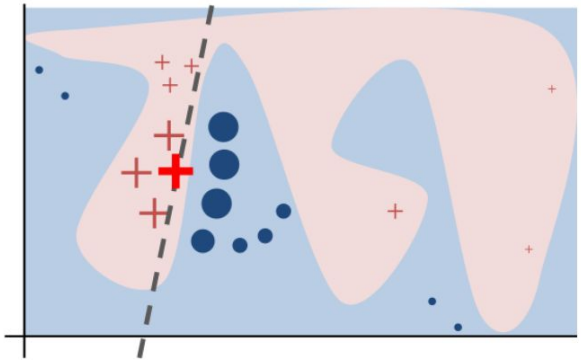
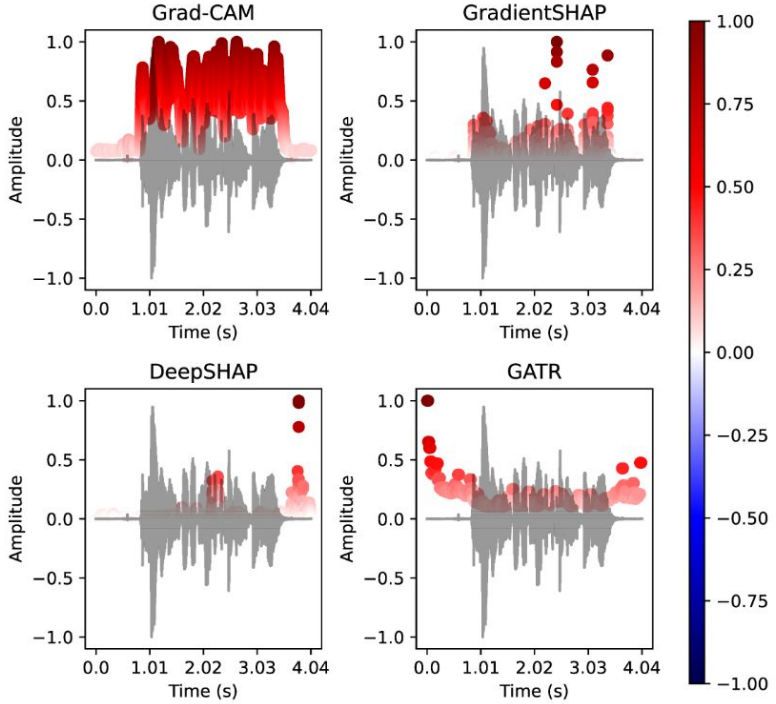


Figure 3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful.

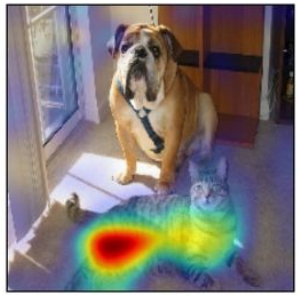
"Why Should I Trust You?": Explaining the Predictions of Any Classifier (KDD 2016)

Problem with post-hoc:



What Does an Audio Deepfake Detector Focus on? A Study in the Time Domain (ICASSP 2025)

Problem with XAI for deepfake detection:



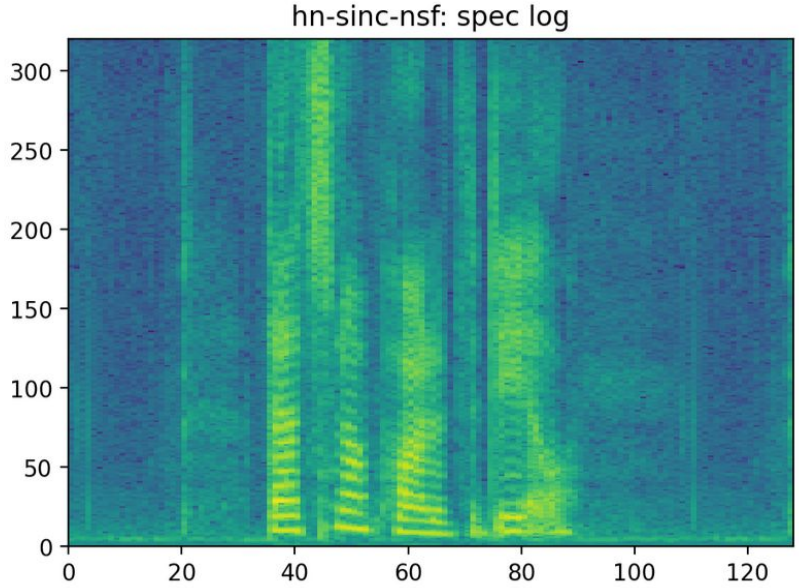
(c) Grad-CAM ‘Cat’

(i) Grad-CAM ‘Dog’

Easy to see if the XAI is correct

Absence of ground-truth explanations

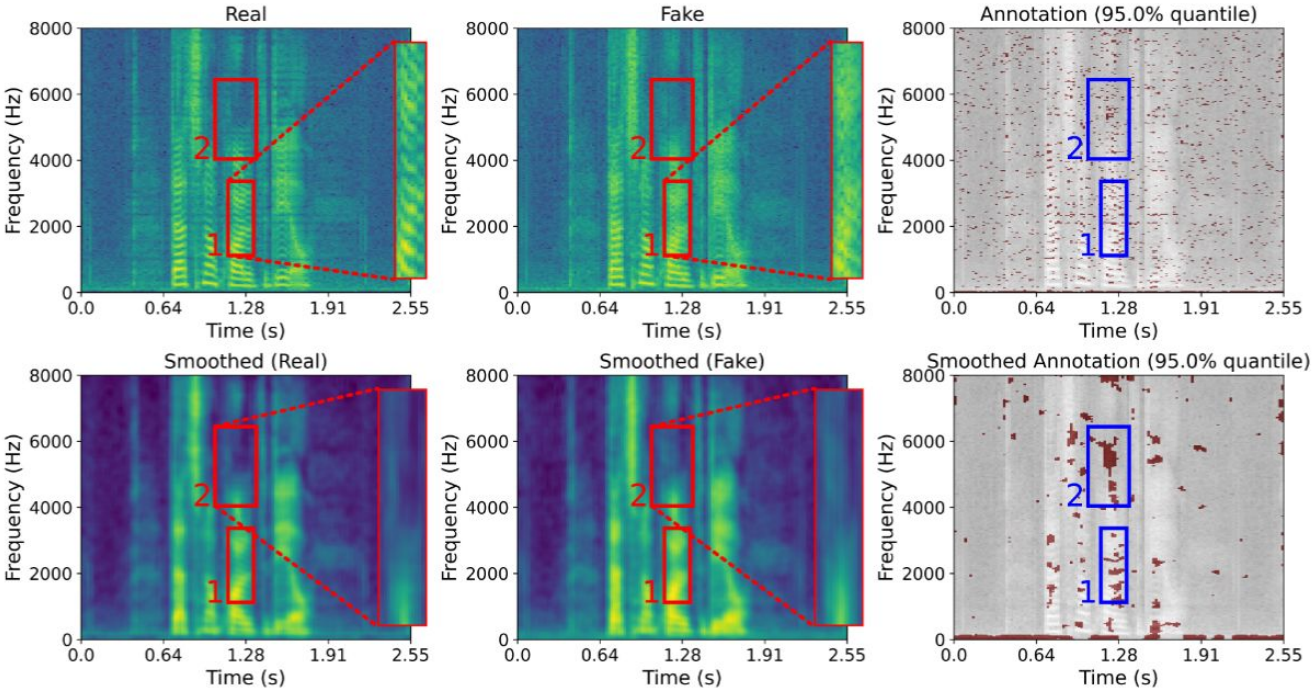
Hard: why is it a deepfake?



A Data-Driven Diffusion-based Approach for Audio Deepfake Explanations (Interspeech 2025)

Problem with XAI for deepfake detection:

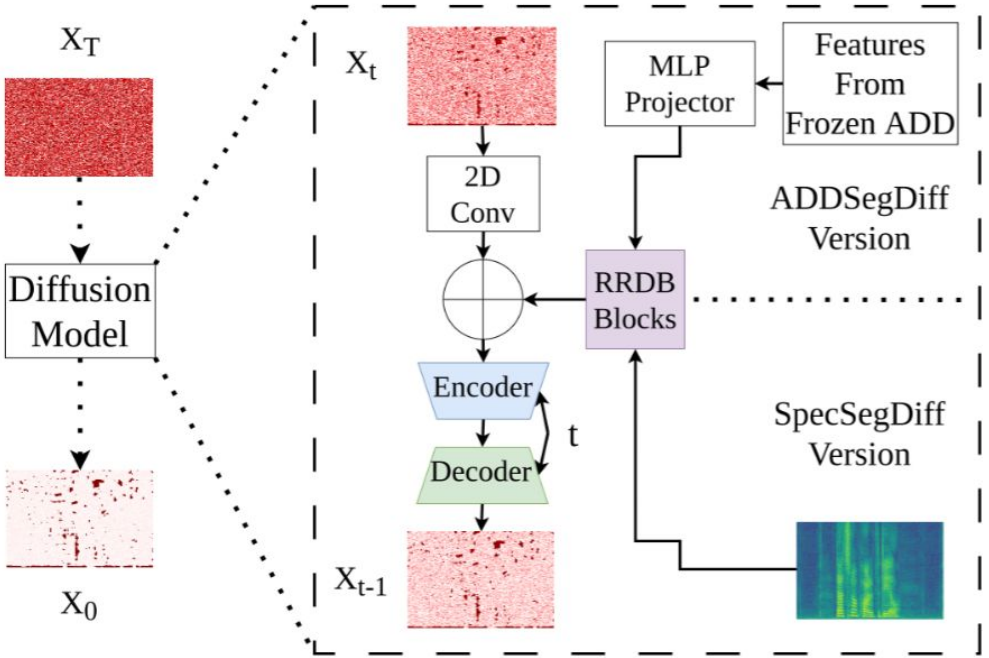
Easier if we know how it should be



[A Data-Driven Diffusion-based Approach for Audio Deepfake Explanations (Interspeech 2025)]

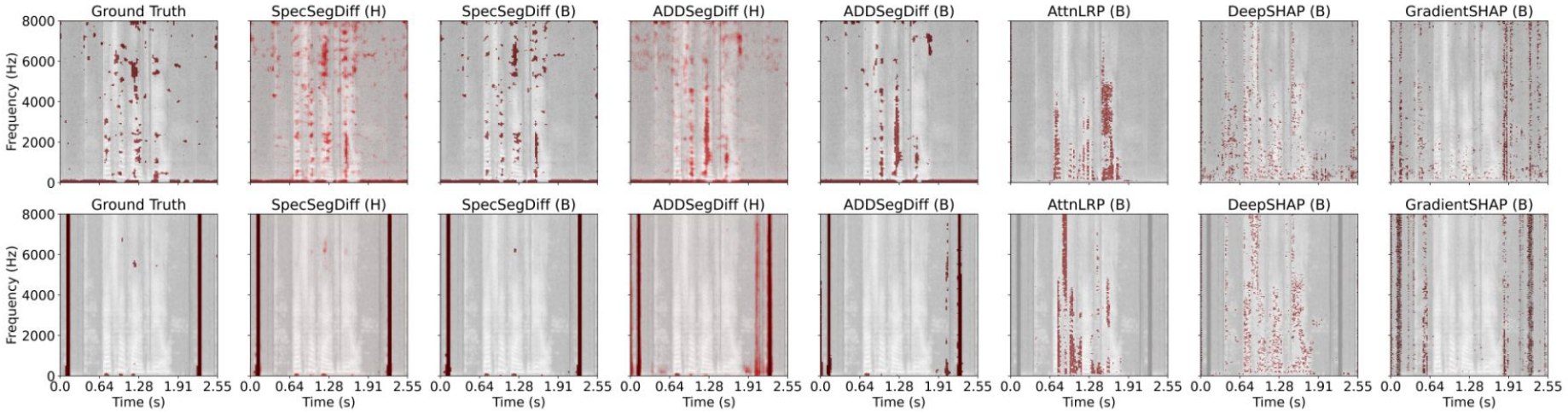
Data-Driven XAI

Can be used to create a data-drive n XAI



A Data-Driven Diffusion-based Approach for Audio Deepfake Explanations (Interspeech 2025)

Data-Driven XAI



A Data-Driven Diffusion-based Approach for Audio Deepfake Explanations (Interspeech 2025)