MaskMark: Robust Neural Watermarking for Real and Synthetic Speech

Patrick O'Reilly[1], Zeyu Jin[2], Jiaqi Su[2], Bryan Pardo[1]

IEEE ICASSP 2024

1. Northwestern University
2. Adobe Research

MaskMark: Robust Neural Watermarking for Real and Synthetic Speech

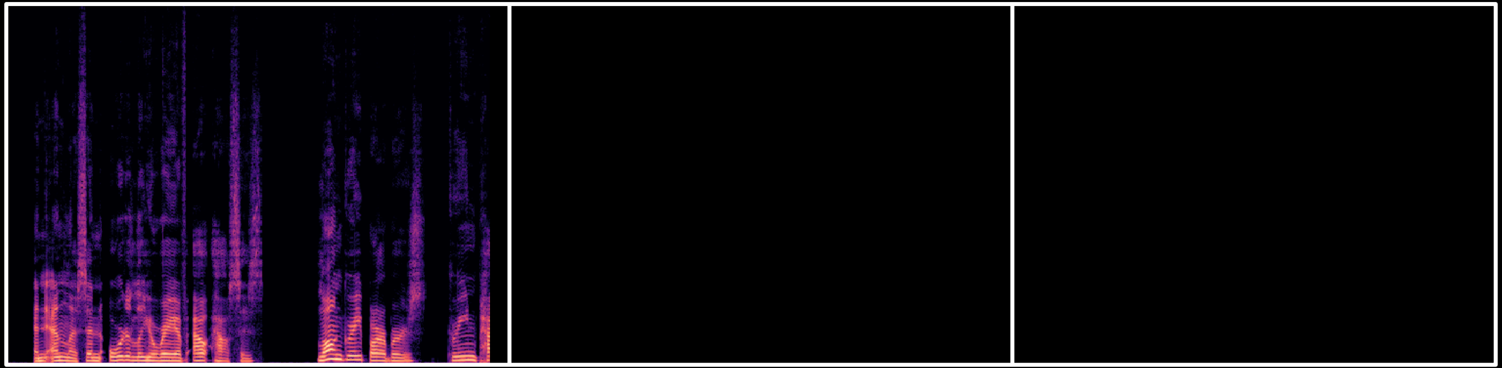
(Listening examples)

TL;DR:

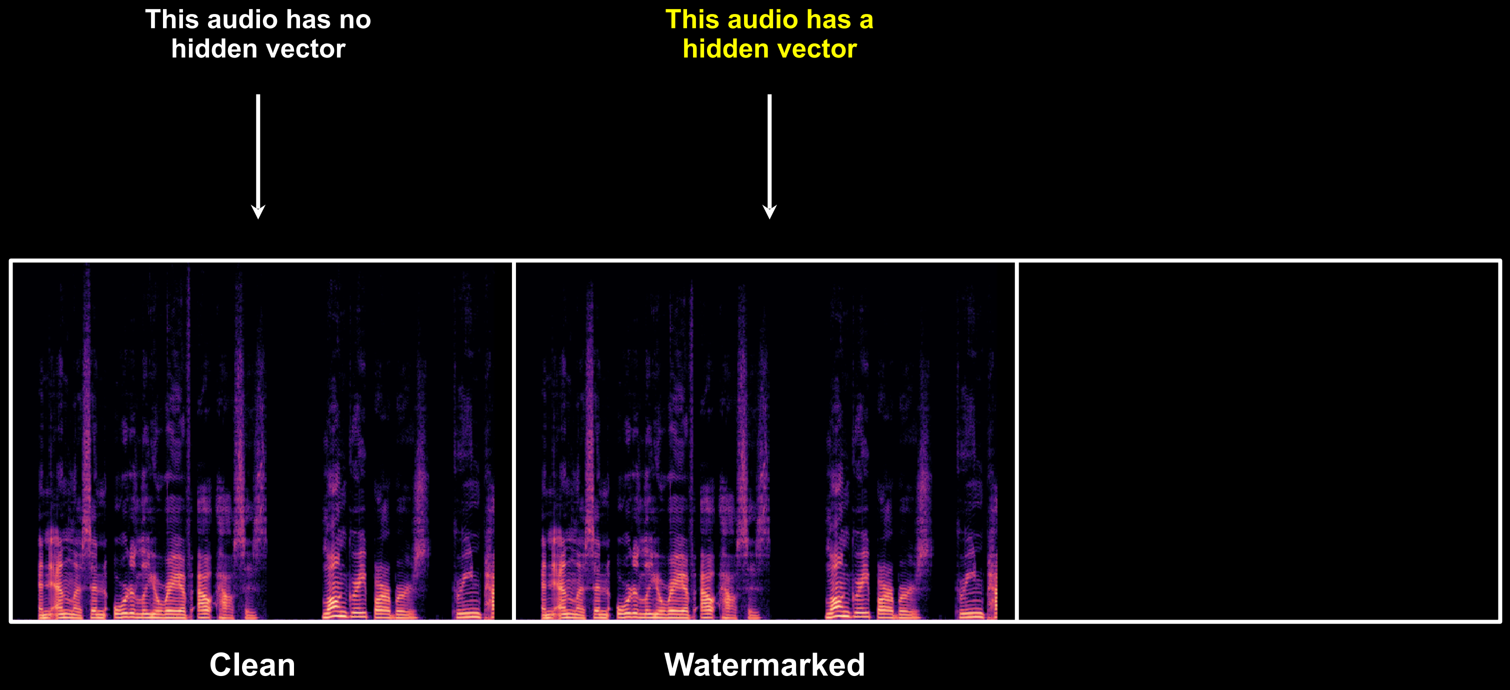
In this work, we show how to hide a binary vector in audio that can be recovered even when the audio has been altered significantly.

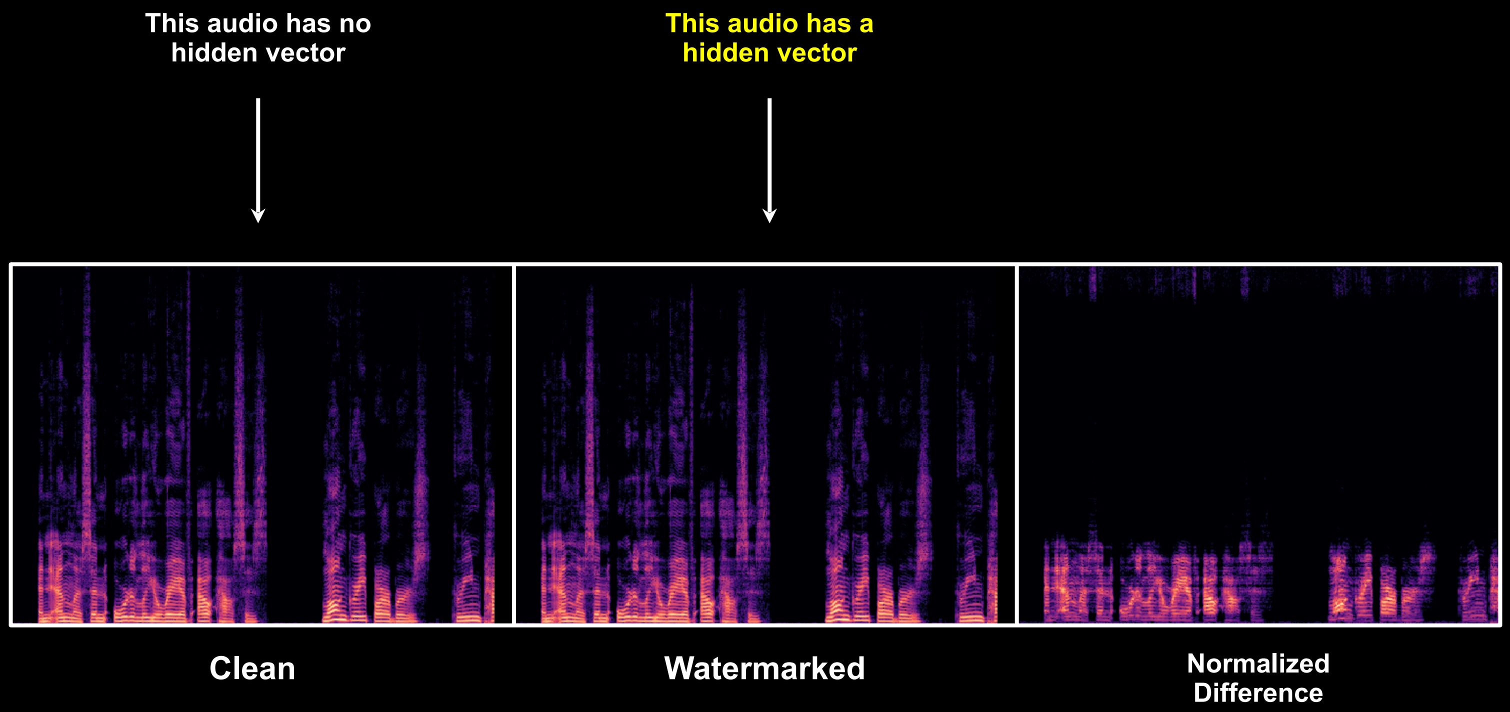
Let’s look at some examples.

This audio has no
hidden vector



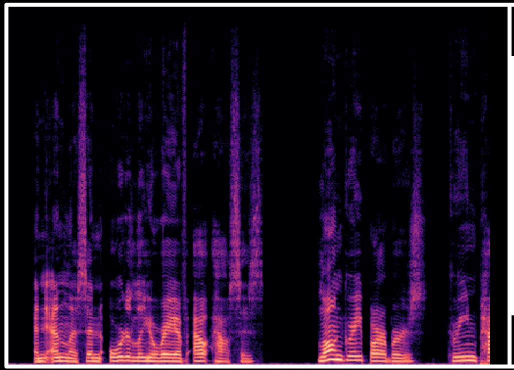
Clean


This audio has no hidden vector
This audio has a hidden vector
Clean
Watermarked

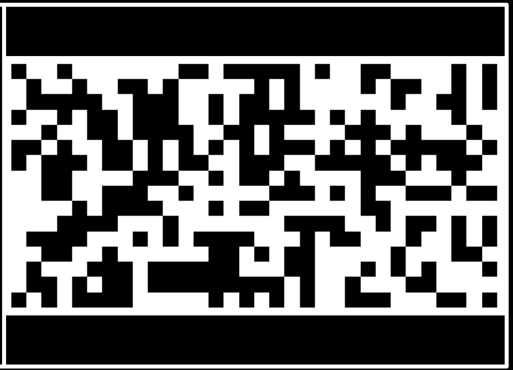

This audio has no
hidden vector
This audio has a
hidden vector
Clean
Watermarked
Normalized
Difference

Watermarked

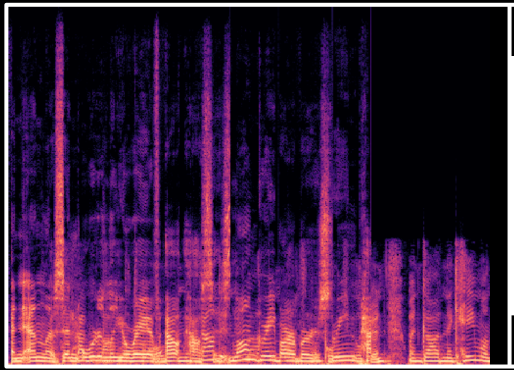
Embedded key vector



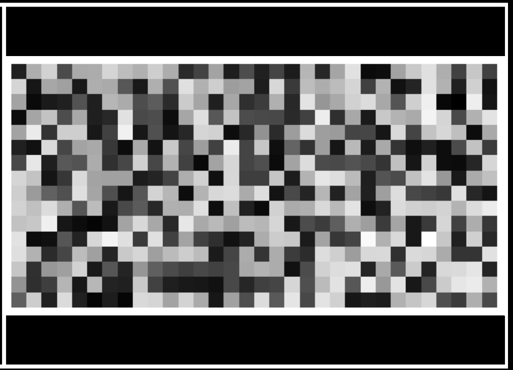
Watermarked



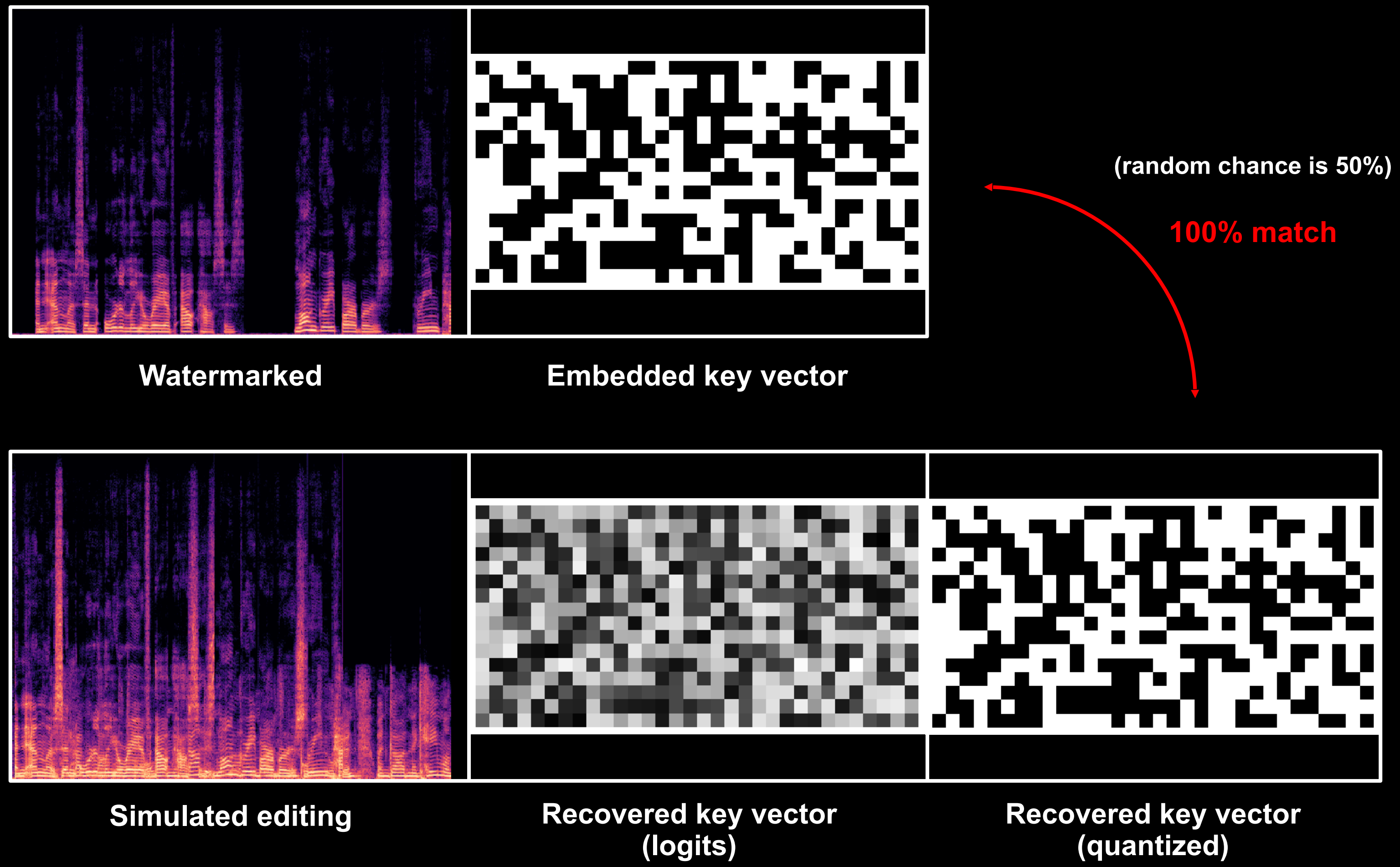
Embedded key vector

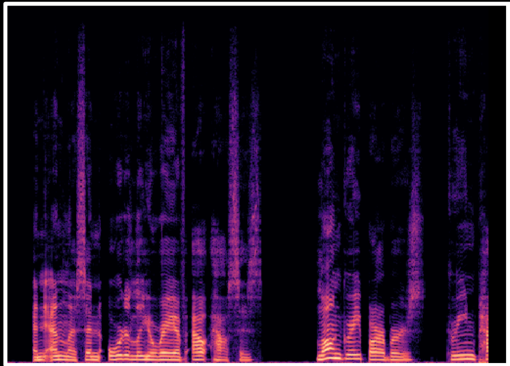


Simulated editing

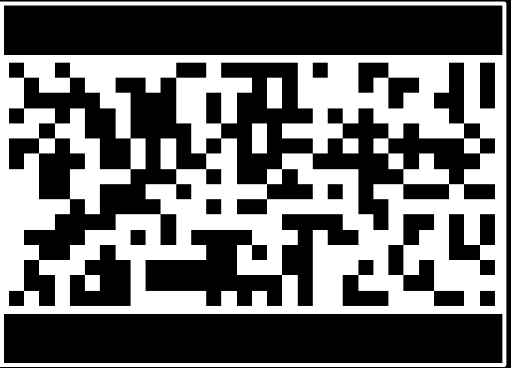


Recovered key vector
(logits)

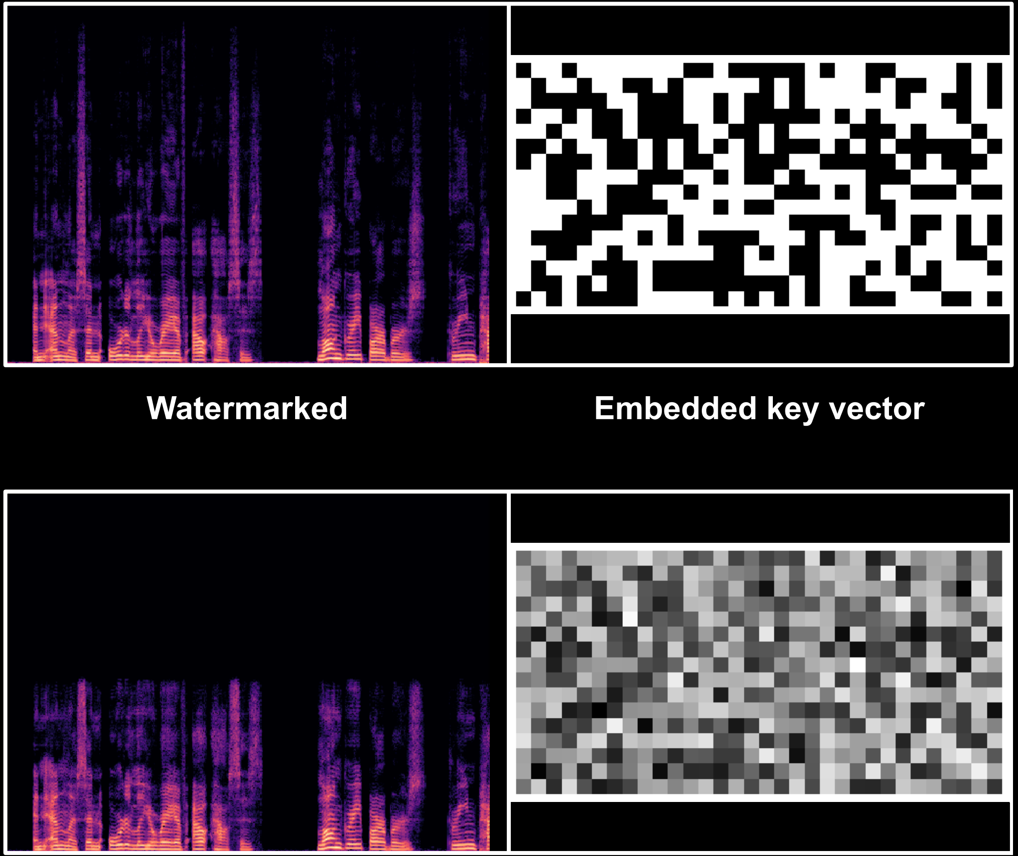

(random chance is 50%)
100% match
Watermarked
Embedded key vector
Simulated editing
Recovered key vector
(logits)
Recovered key vector
(quantized)



Watermarked



Embedded key vector

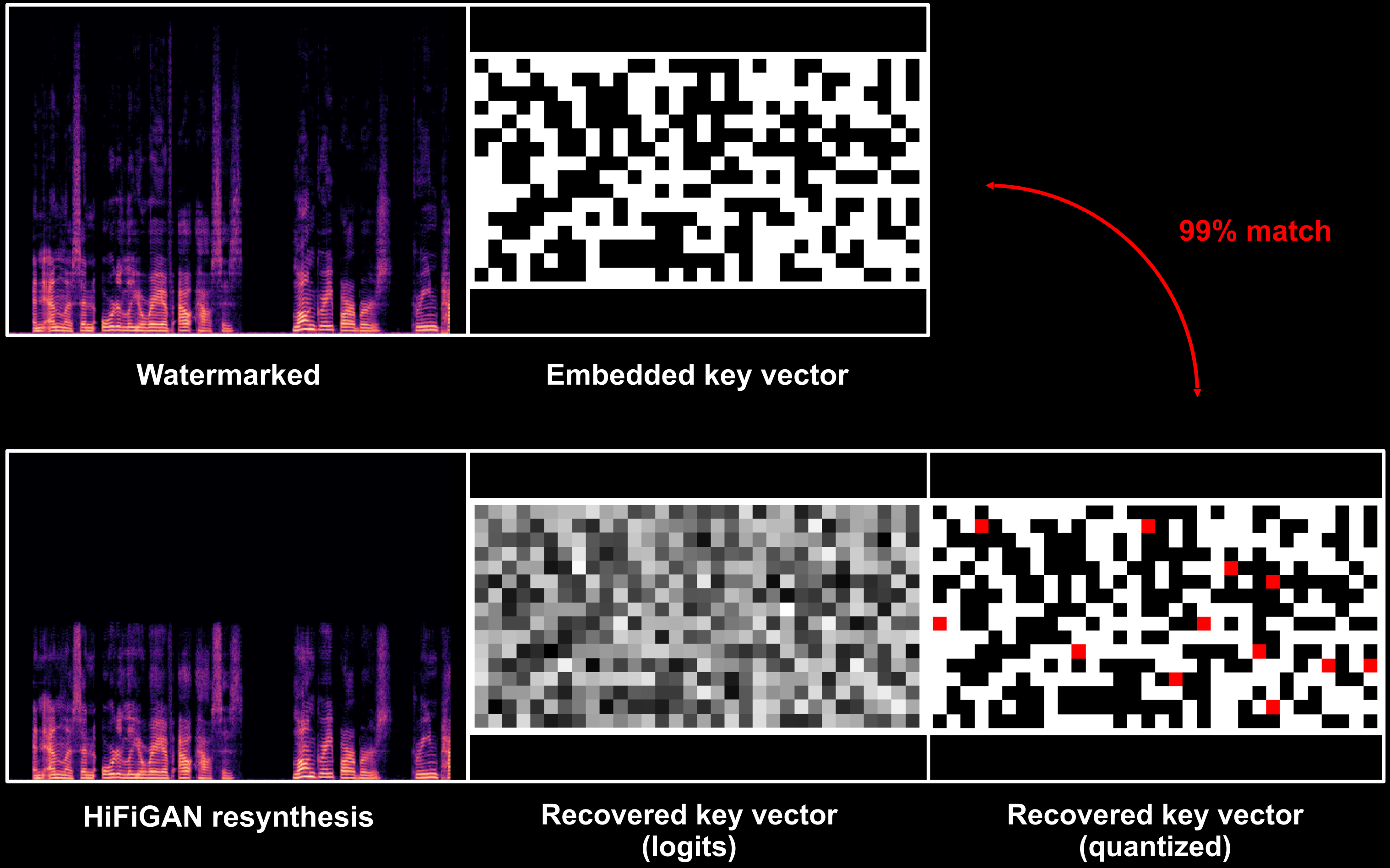


Watermarked

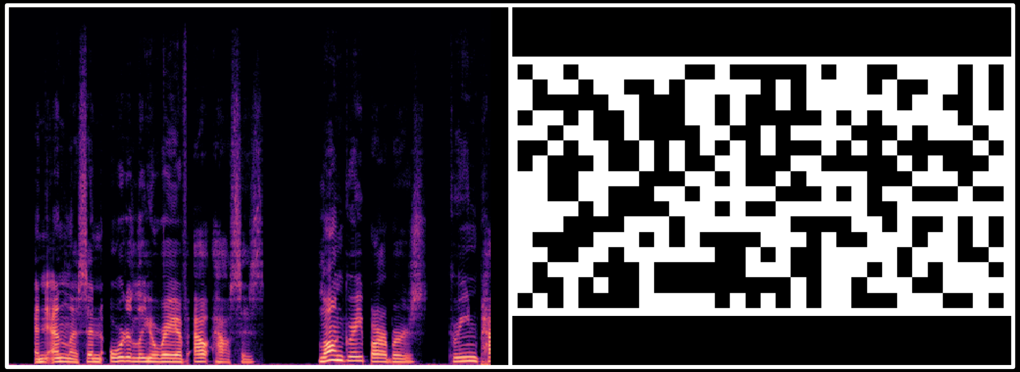
Embedded key vector

HiFiGAN resynthesis

Recovered key vector
(logits)

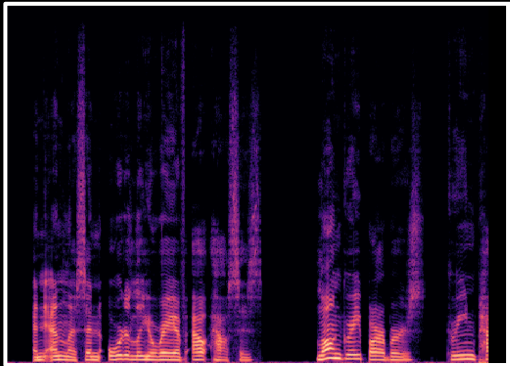


99% match
Watermarked
Embedded key vector
HiFiGAN resynthesis
Recovered key vector
(logits)
Recovered key vector
(quantized)

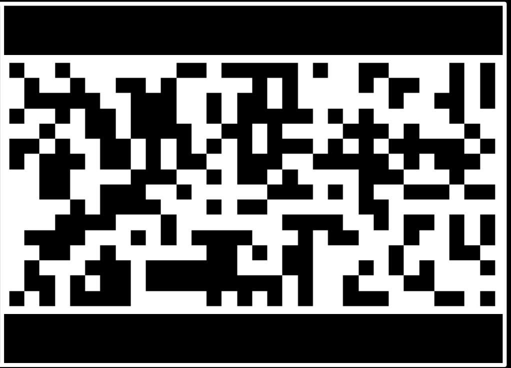


Watermarked

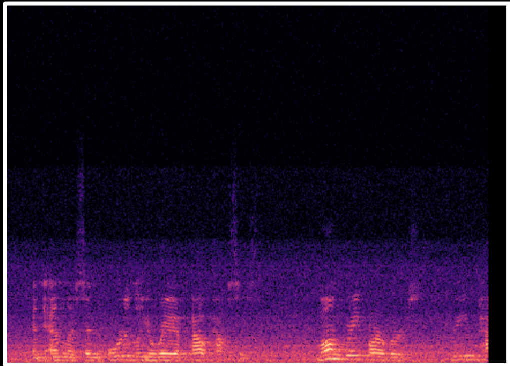
Embedded key vector



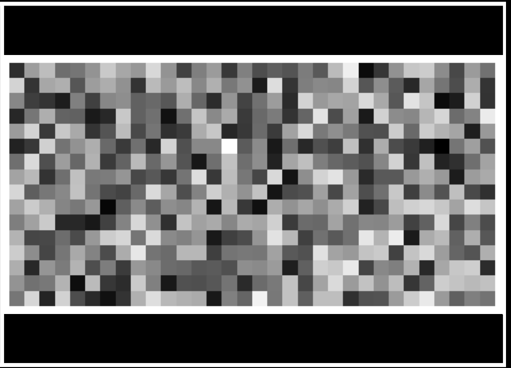
Watermarked



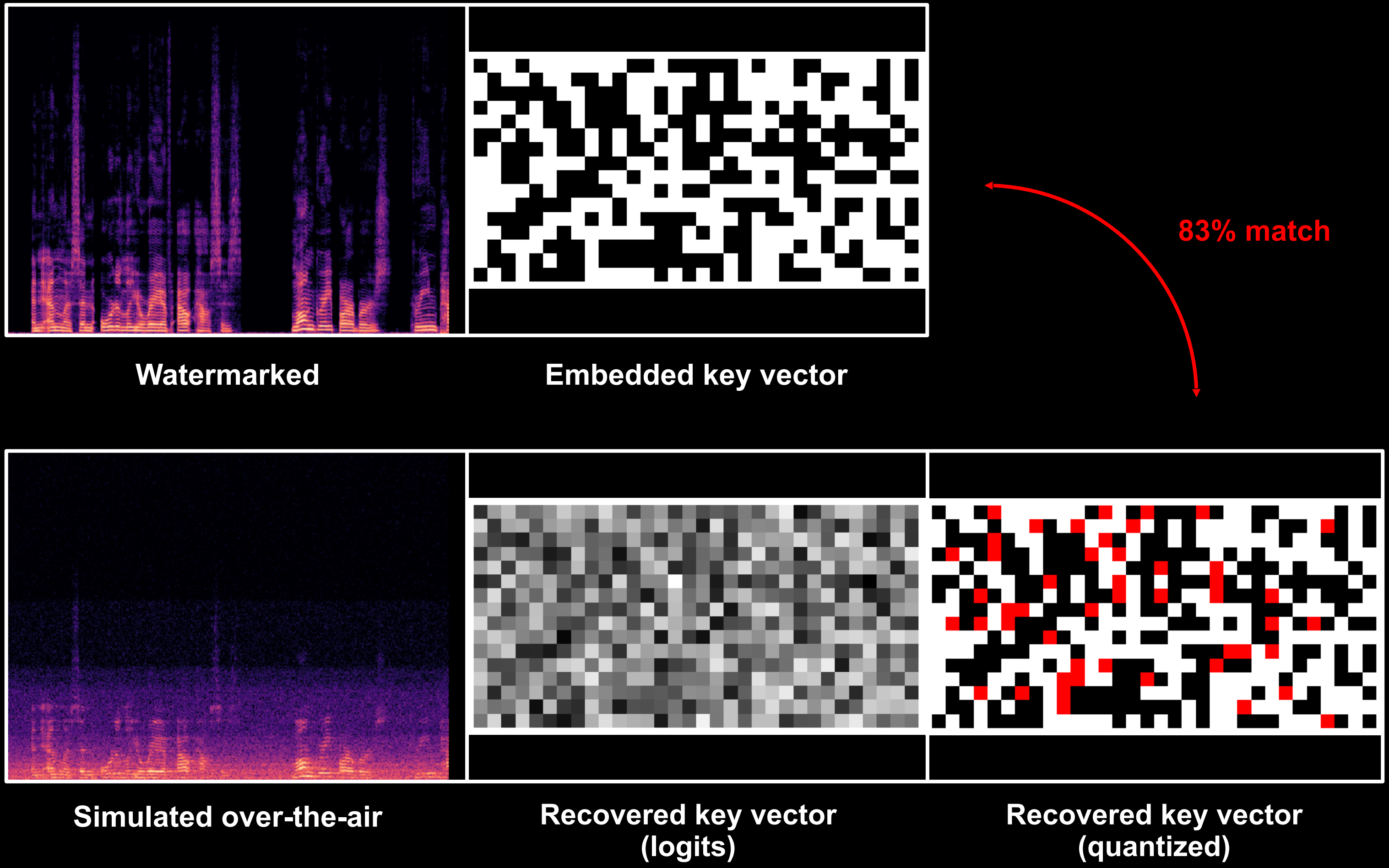
Embedded key vector



Simulated over-the-air



Recovered key vector
(logits)


83% match
Watermarked
Embedded key vector
Simulated over-the-air
Recovered key vector (logits)
Recovered key vector (quantized)

Why should we care about hiding binary vectors in audio clips?

2016: WaveNet

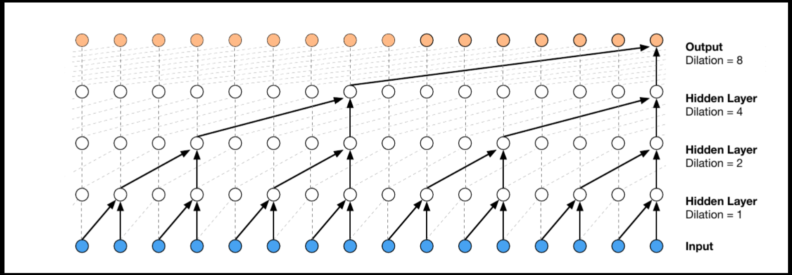
Expertise + compute + a large single-speaker dataset + lots of time

=

2016: WaveNet

Expertise + compute + a large single-speaker dataset + lots of time

=



2023: Suno Bark

$0 + 1-10 min. audio + 5 min. editing

=

2023: Suno Bark

$0 + 1-10 min. audio + 5 min. editing

=

ARTIFICIAL INTELLIGENCE / TECH / CREATORS

4chan users embrace AI voice clone tool to generate celebrity hatespeech

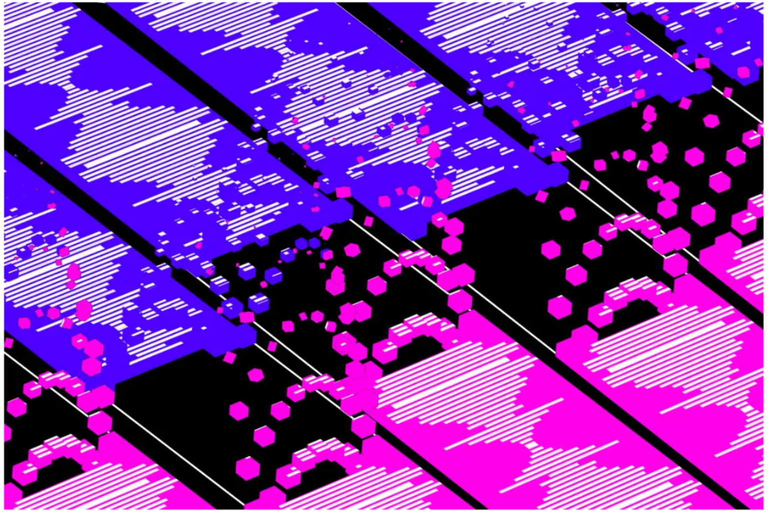


Illustration by Alex Castro / The Verge

/ Free AI voice cloning technology from startup ElevenLabs has been used by trolls to imitate the voices of celebrities. The generated audio ranges in content from memes and erotica to virulent hatespeech.

By James Vincent, a senior reporter who has covered AI, robotics, and more for eight years at The Verge.

Jan 31, 2023, 5:00 AM PST | 7 Comments / 7 New

Attacks/Breaches | 5 MIN READ | NEWS

AI-Enabled Voice Cloning Anchors Deepfaked Kidnapping

Virtual kidnapping is just one of many new artificial intelligence attack types that threat actors have begun deploying, as voice cloning emerges as a potent new imposter tool.

What can speech synthesis providers do?

ElevenLabs

TTS

Microsoft Azure

bing

RESEMBLE.AI

We can hide a message in all the audio we generate

We can hide a message in all the audio we generate

We can check any audio for the message

We can hide a message in all the audio we generate

We can check any audio for the message

If we find the message, the audio was generated by our system

“embed”

We can **hide** a **message** in all the audio we generate

We can **check** any audio for the message

If we find the message, the audio was generated by our system

“embed”

We can hide a message in all the audio we generate

“watermark key”

We can check any audio for the message

If we find the message, the audio was generated by our system

“embed”

We can **hide** a **message** in all the audio we generate

“watermark key”

[0, 1, 1, 0, 0, 1, …]

$\underbrace{\qquad\qquad\qquad}_{n \text{ bits}}$

We can **check** any audio for the message

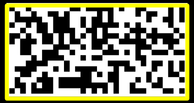
If we find the message, the audio was generated by our system

“embed”

We can **hide** a **message** in all the audio we generate

“watermark key”

$\underbrace{\texttt{[0, 1, 1, 0, 0, 1, …]}}_{n \text{ bits}}$



We can **check** any audio for the message

If we find the message, the audio was generated by our system

“embed”

We can hide a message in all the audio we generate

“watermark key”

`[0, 1, 1, 0, 0, 1, … ]`

n bits

We can check any audio for the message

If we find the message, the audio was generated by our system

“embed”

We can **hide** a **message** in all the audio we generate

“watermark key”

`[0, 1, 1, 0, 0, 1, … ]`

n bits

We can **check** any audio for the message

“detect”

If we find the message, the audio was generated by our system

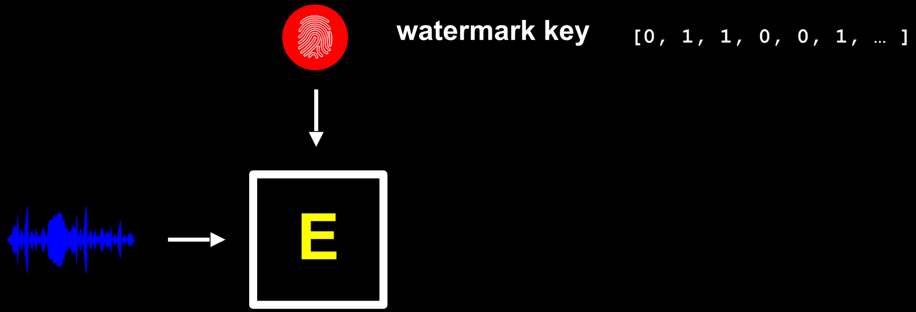
Watermarking

Watermarking

Embed the watermark

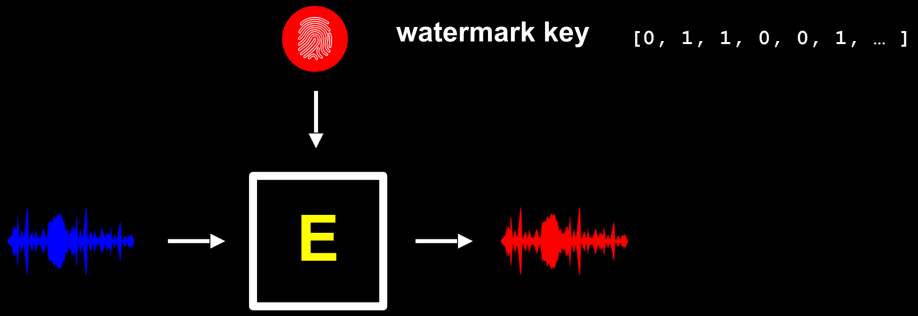
Watermarking

Embed the watermark

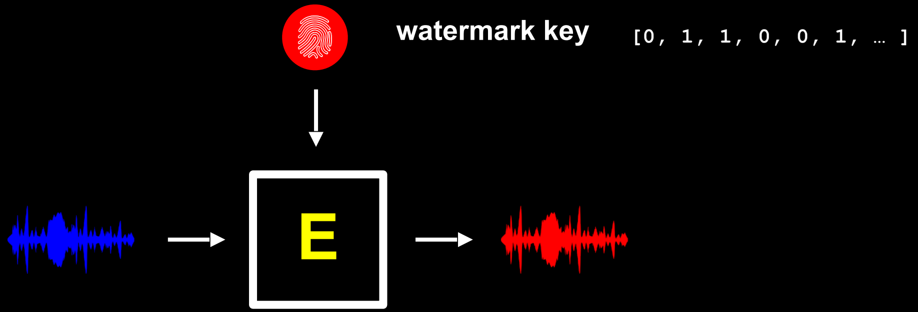


Watermarking

Embed the watermark



Watermarking



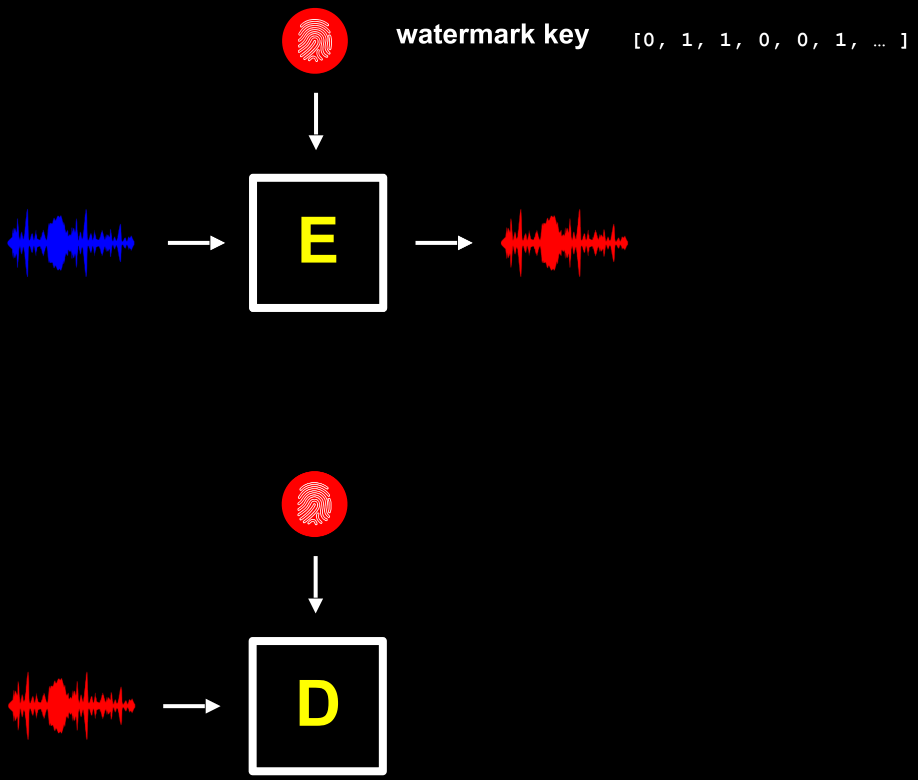
Embed the watermark

Detect the watermark

Watermarking

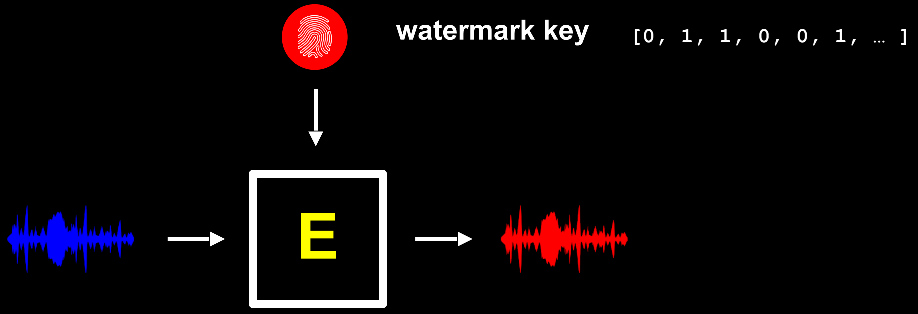
Embed the watermark

Detect the watermark

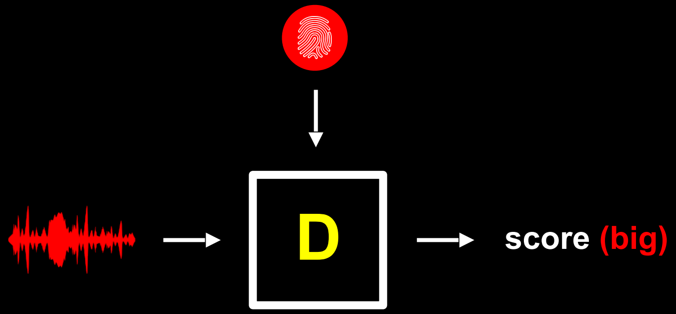


Watermarking

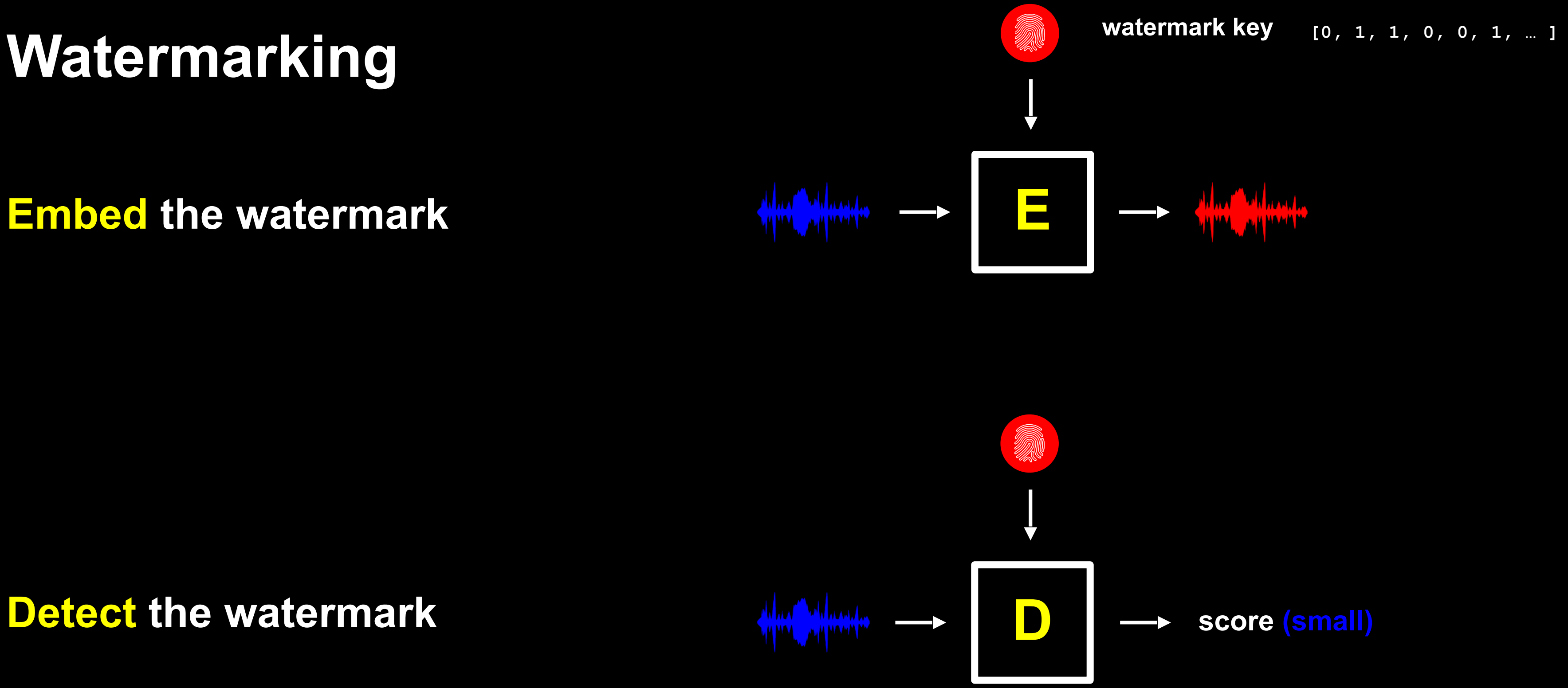
Embed the watermark



Detect the watermark



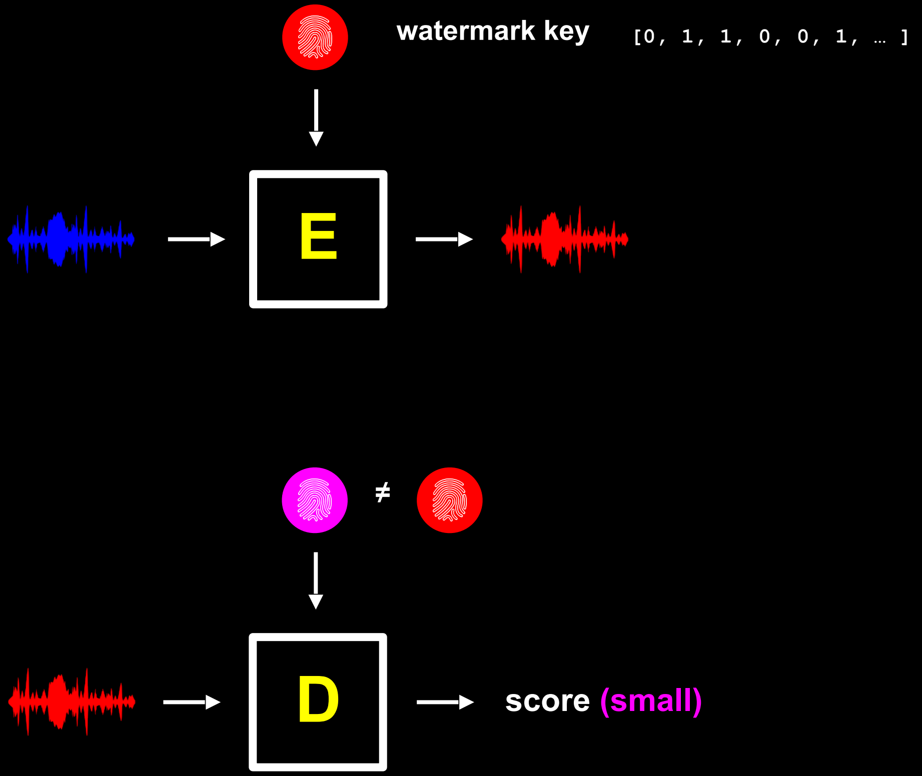
Watermarking



Watermarking

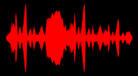
Embed the watermark

Detect the watermark

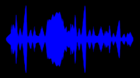
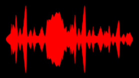


Desiderata

Desiderata

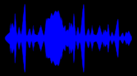
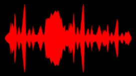
1.  ≈ 

Desiderata

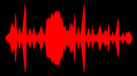
1.  ≈ 

2. $\left|\left\{\right.\right.$  $\left.\left.\right\}\right|$ is big

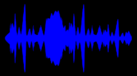
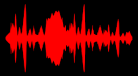
Desiderata

1.  ≈ 

2. $\left| \left\{ \right.$  $\left. \right\} \right|$ is big

3.  is hard to remove from 

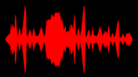
Desiderata

1.  ≈ 

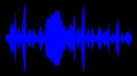
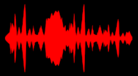
Perceptual transparency

watermark doesn’t ruin user experience

2.  is big

3.  is hard to remove from 

Desiderata

1.  ≈ 

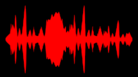
Perceptual transparency

watermark doesn’t ruin user experience

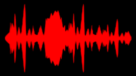
2.  is big

Capacity

can hide info like user IDs in the watermark

3.  is hard to remove from 

Desiderata

1.  ≈ 

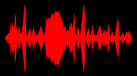
Perceptual transparency

watermark doesn’t ruin user experience

2. |{  }| is big

Capacity

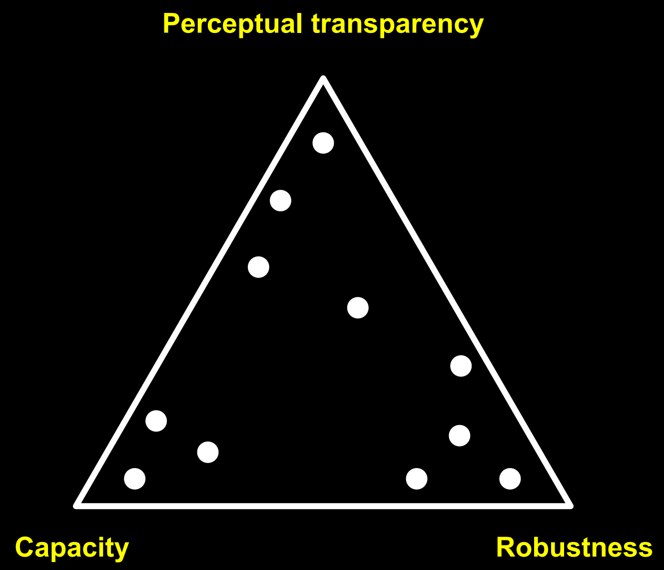
can hide info like user IDs in the watermark

3.  is hard to remove from 

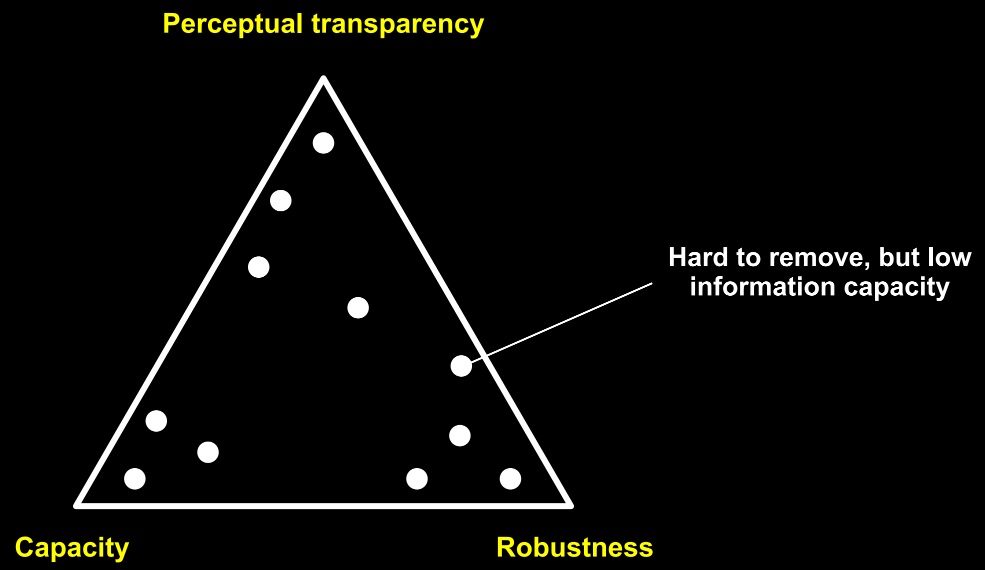
Robustness

watermark works under realistic conditions

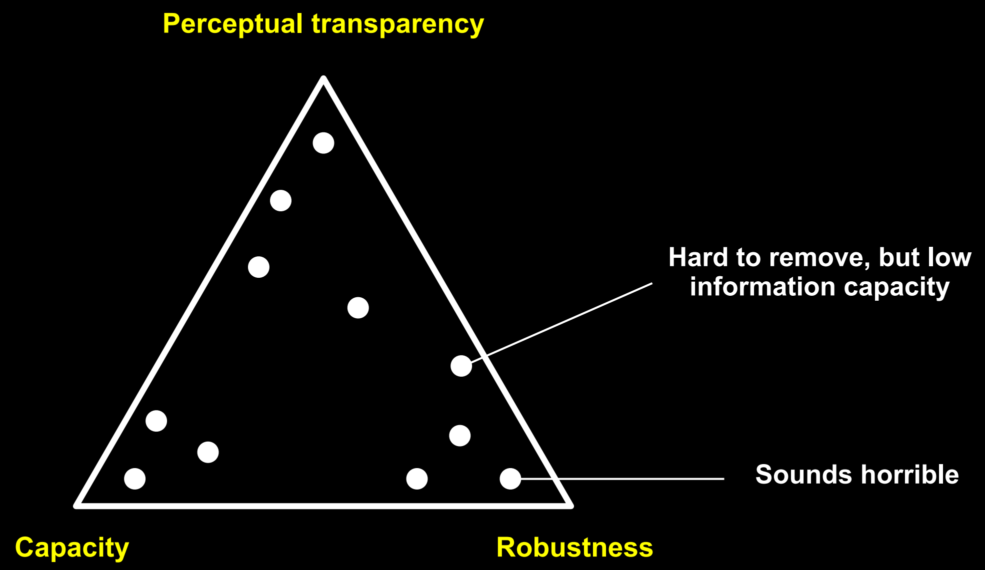
Desiderata



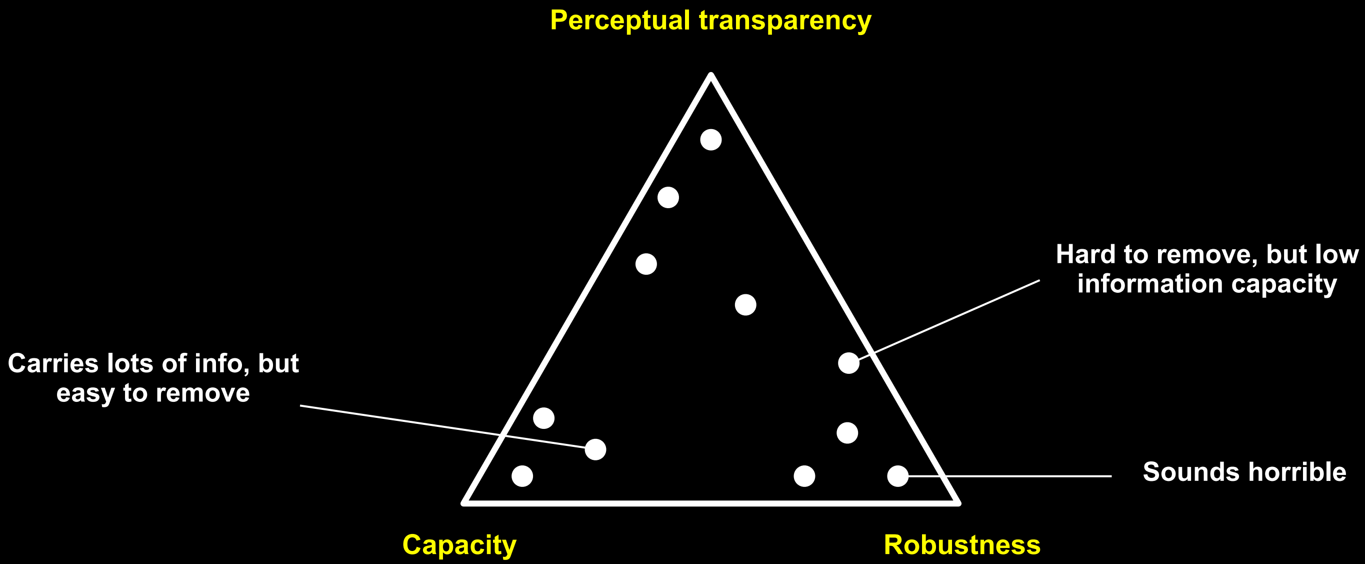
Desiderata


Perceptual transparency
Hard to remove, but low information capacity
Capacity
Robustness

Desiderata

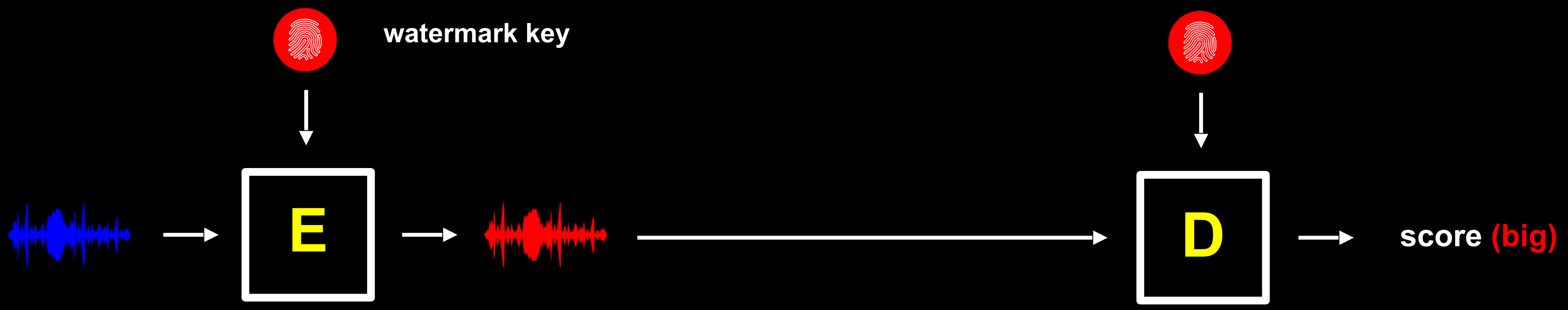

Perceptual transparency
Hard to remove, but low information capacity
Sounds horrible
Capacity
Robustness

Desiderata

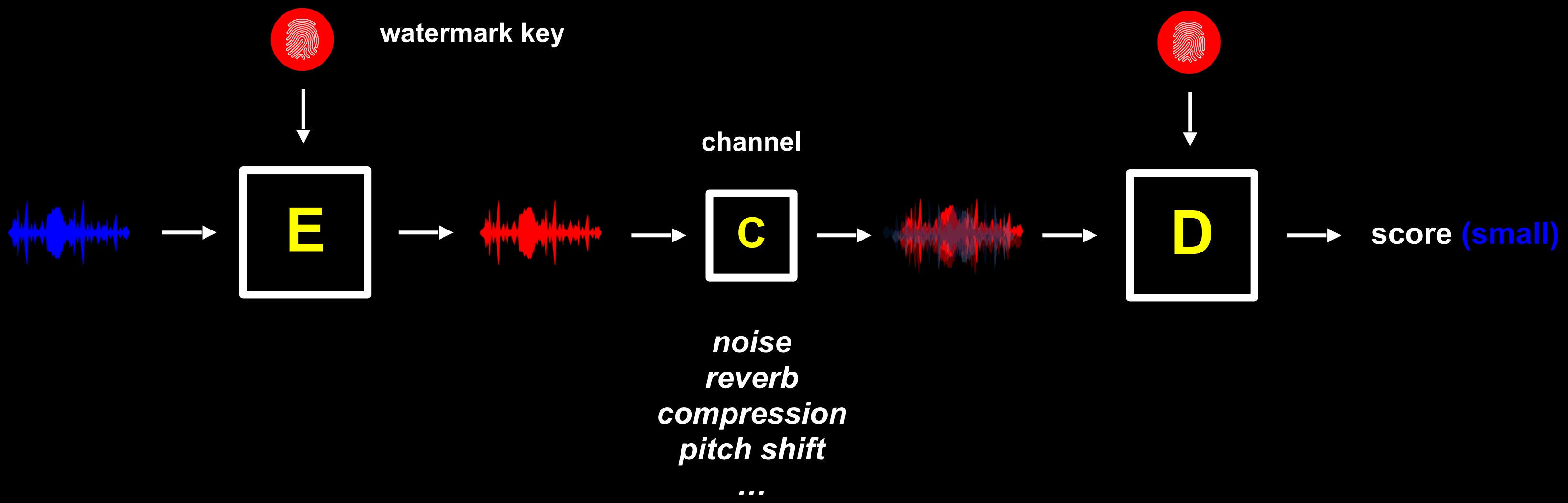

Perceptual transparency
Hard to remove, but low information capacity
Carries lots of info, but easy to remove
Sounds horrible
Capacity
Robustness

Balancing these is hard!

Robustness

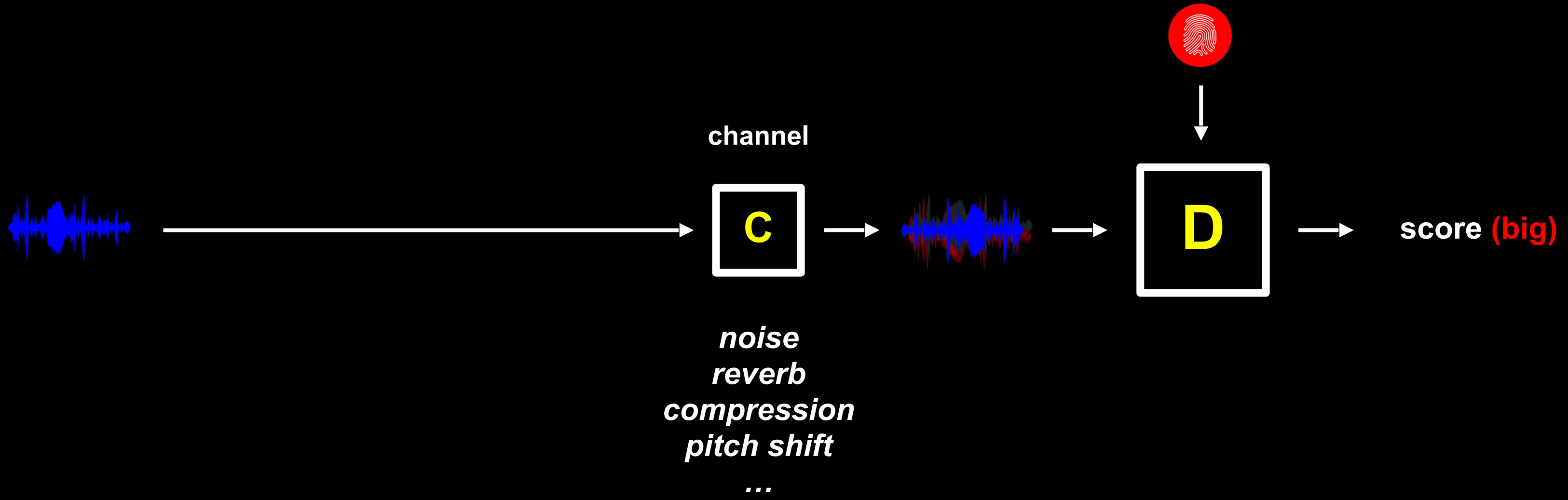


Robustness



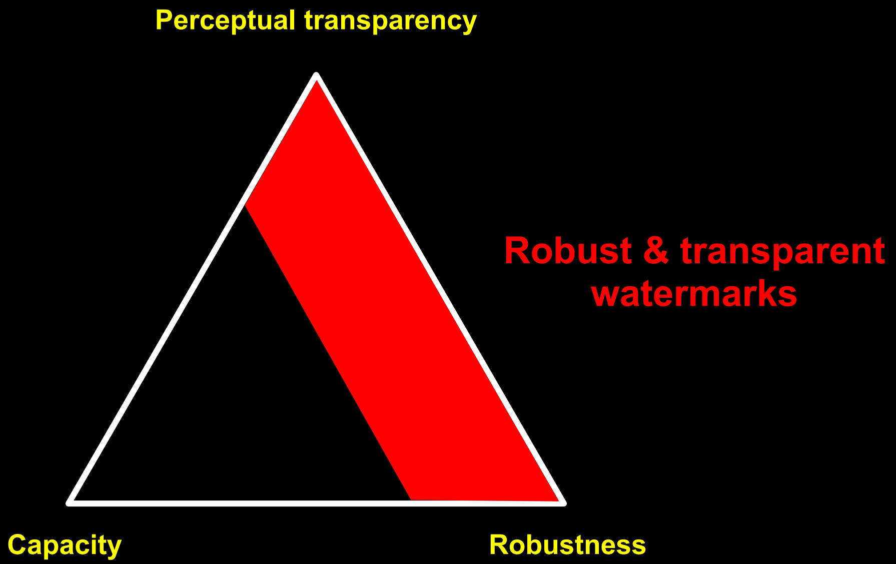
watermark key
channel
E
C
D
score (small)
noise
reverb
compression
pitch shift
...

Robustness



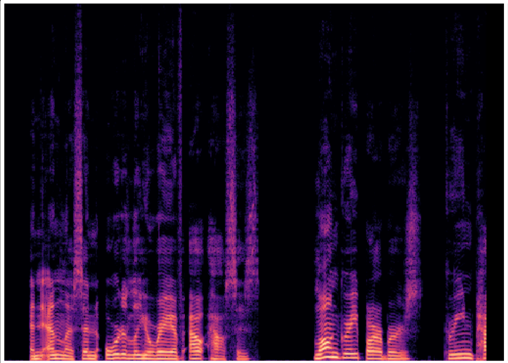
We only need 1 bit to answer “fake or not?”

Desiderata


Perceptual transparency
Robust & transparent
watermarks
Capacity
Robustness

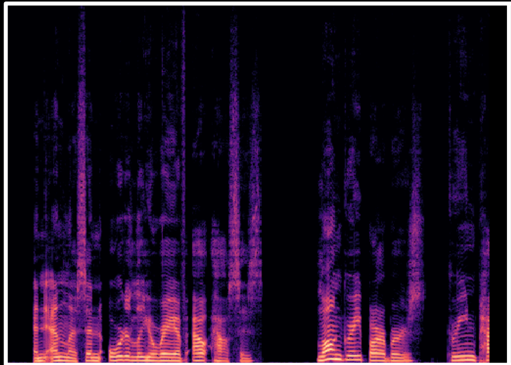
How can we robustly and transparently hide a little information in audio?

“EigenWatermark” (Tai & Mansour 2019)

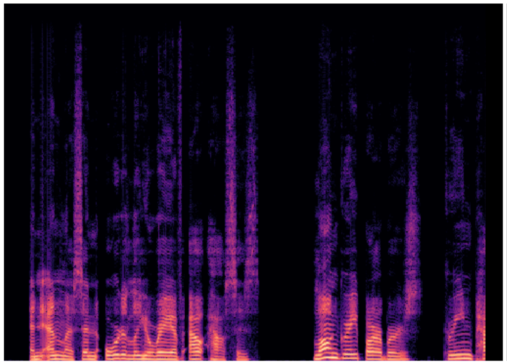


Clean

“EigenWatermark” (Tai & Mansour 2019)

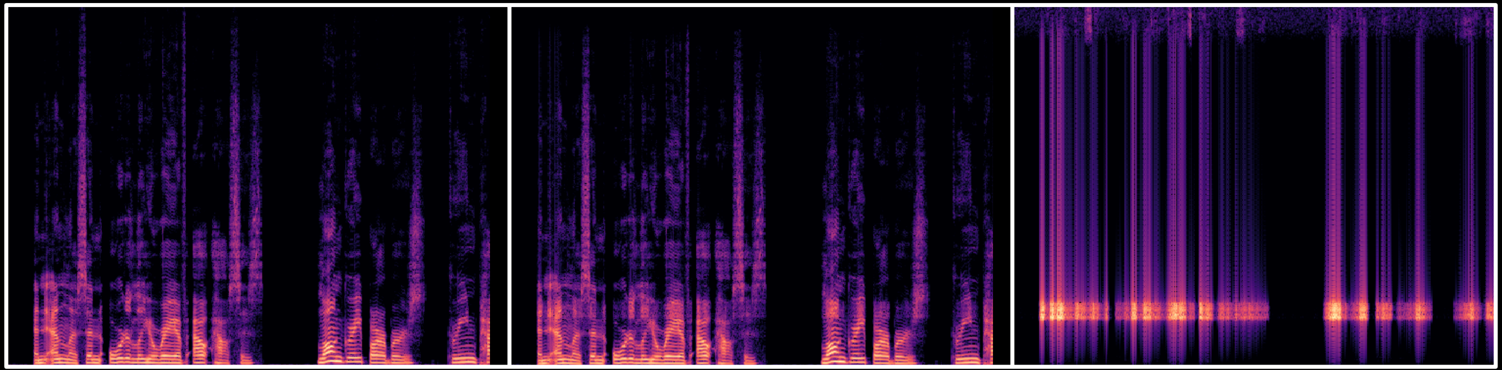


Clean



Watermarked

“EigenWatermark” (Tai & Mansour 2019)

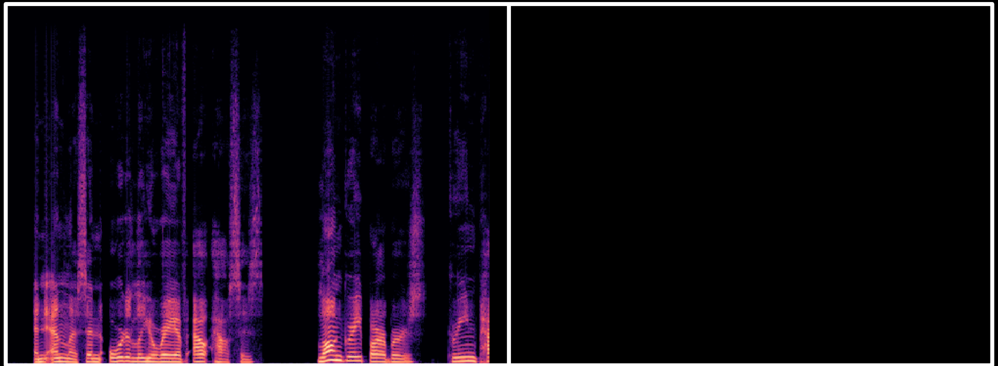


Clean

Watermarked

Normalized Difference

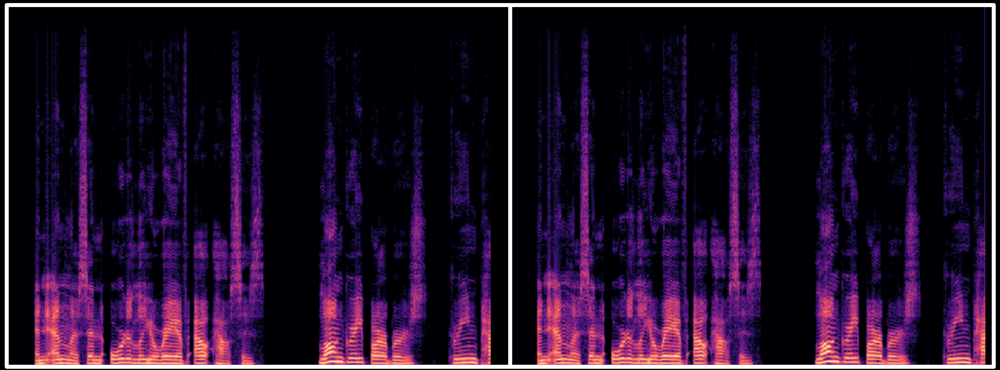
“EigenWatermark” (Tai & Mansour 2019)



Watermarked

Completely breaks the watermark!

“EigenWatermark” (Tai & Mansour 2019)



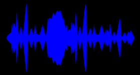
Watermarked | Speed up 2%

Completely breaks the watermark!

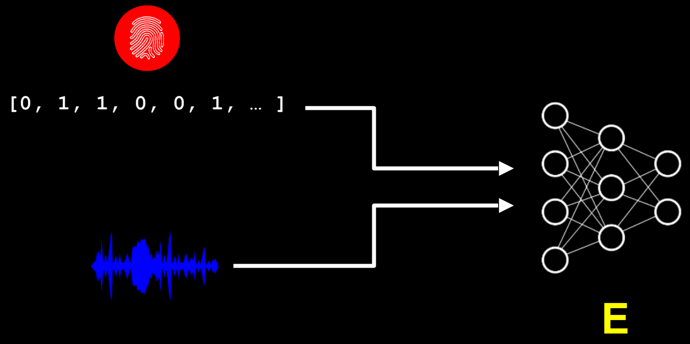
How can we robustly and transparently hide a little information in audio?

Let’s make $\boxed{E}$ and $\boxed{D}$ neural networks

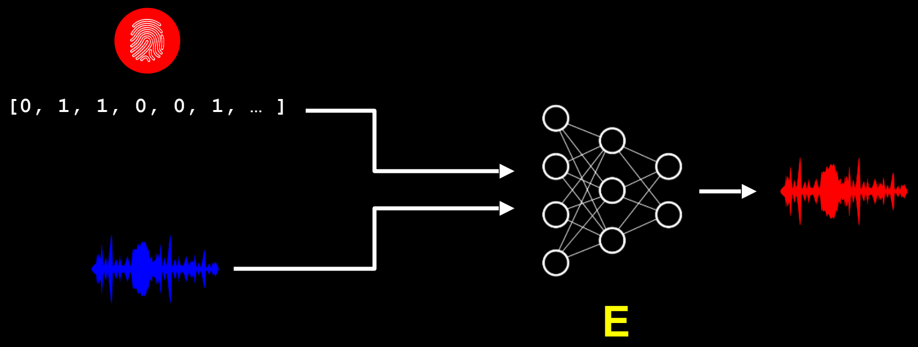
[0, 1, 1, 0, 0, 1, …]



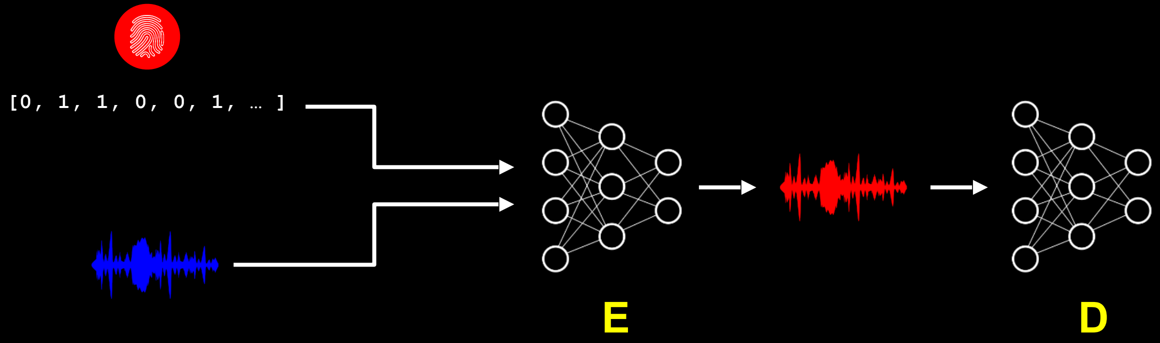
Let's make $\boxed{E}$ and $\boxed{D}$ neural networks



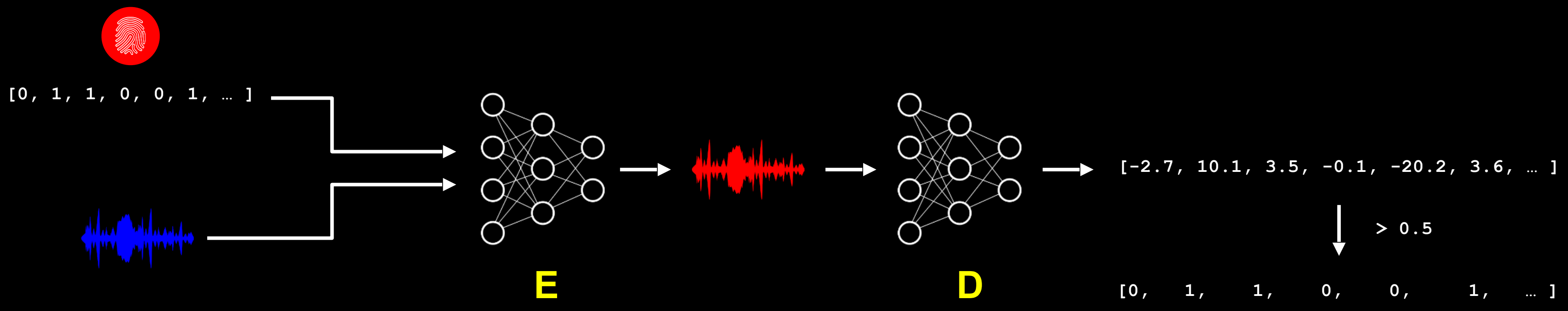
Let's make E and D neural networks



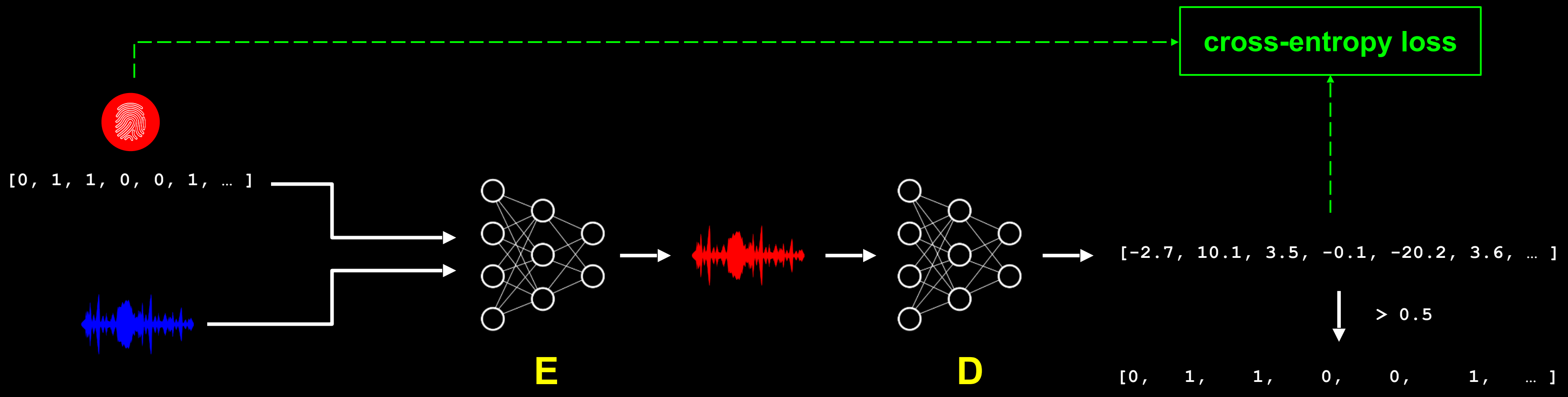
Let’s make $\boxed{E}$ and $\boxed{D}$ neural networks



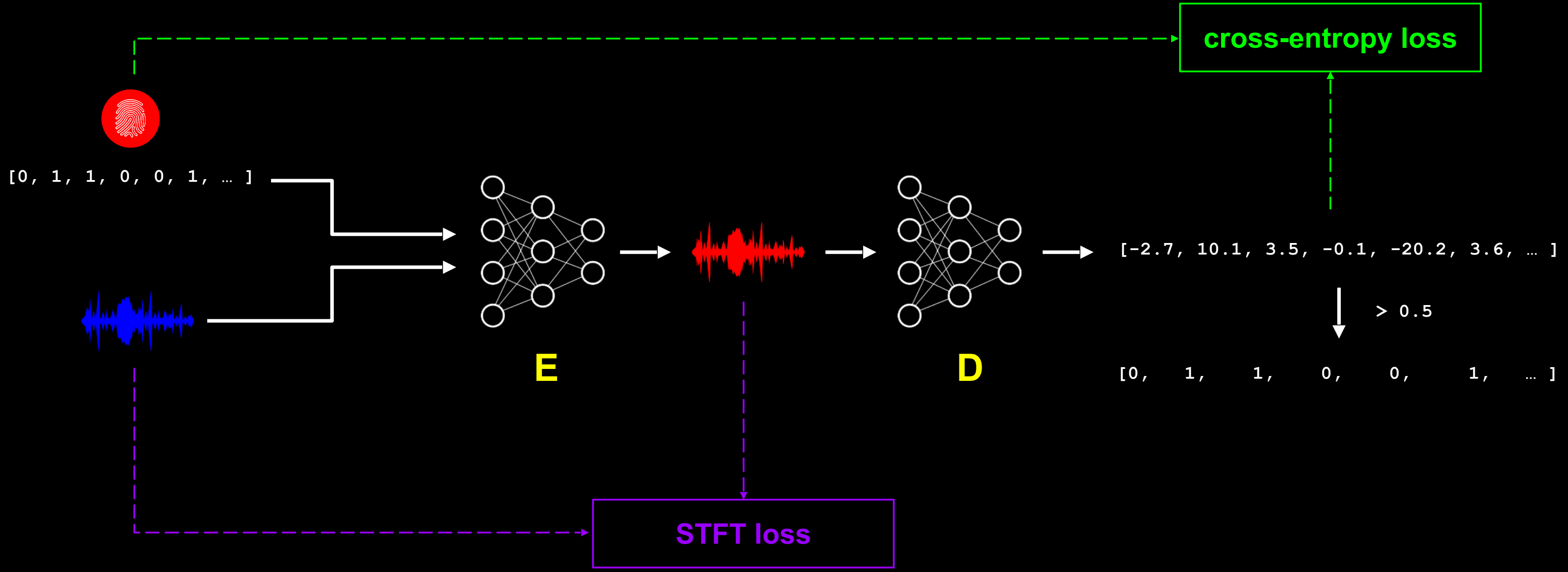
Let's make E and D neural networks



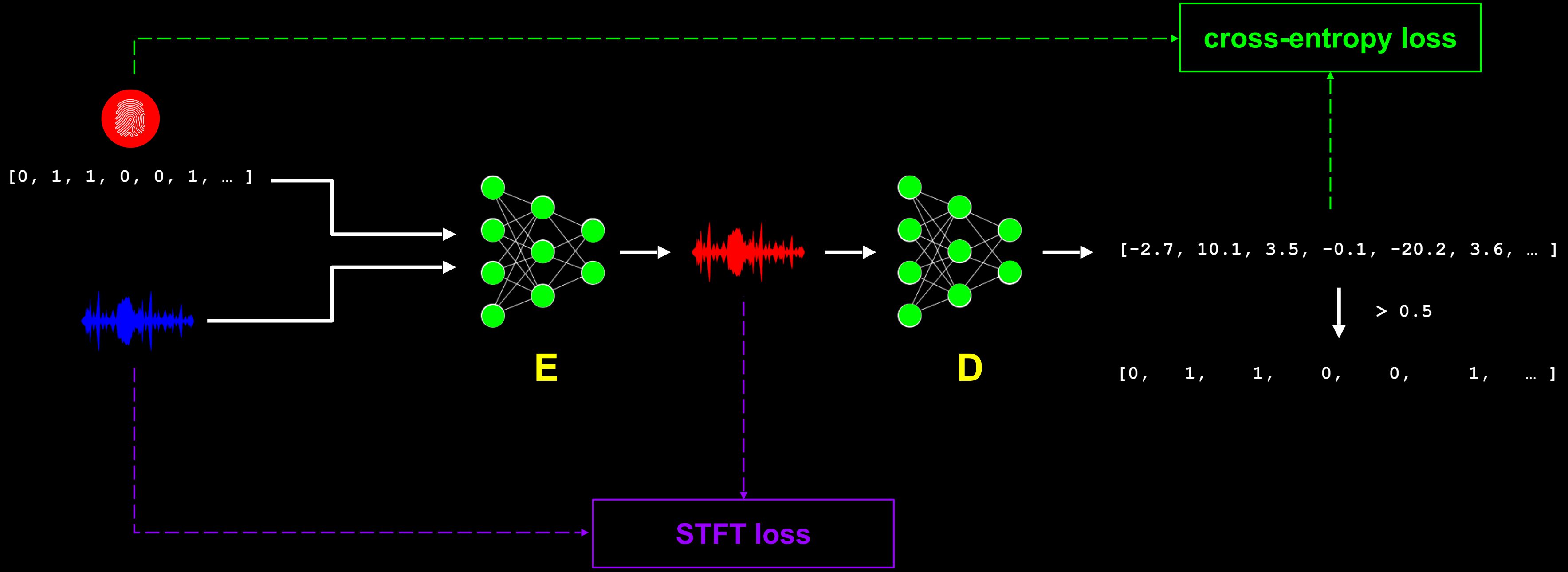
Let's make E and D neural networks



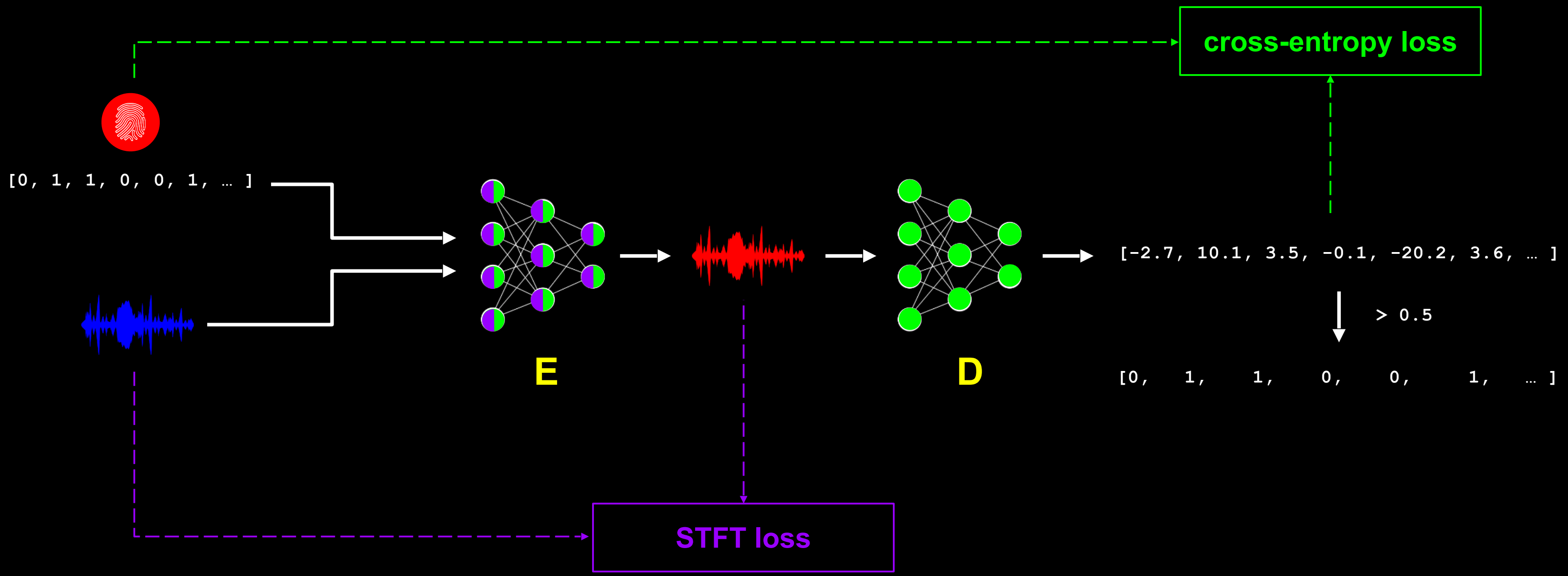
Let's make E and D neural networks



Let's make E and D neural networks

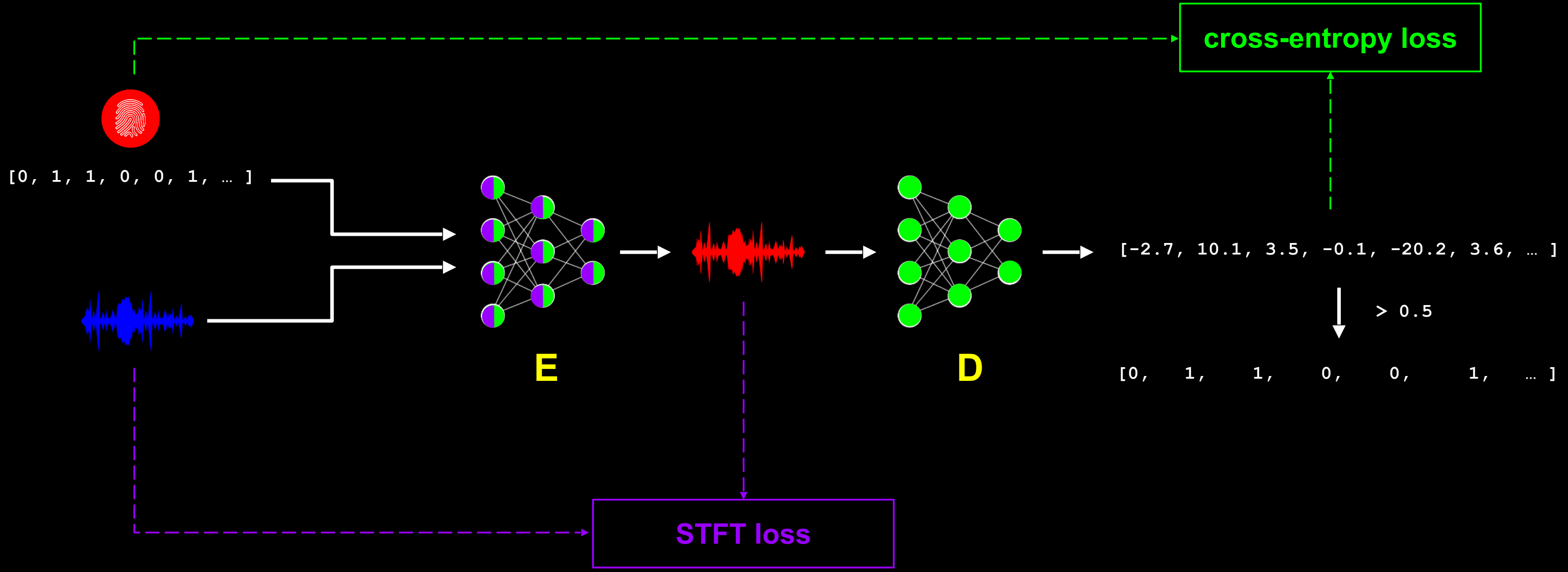


Let's make E and D neural networks

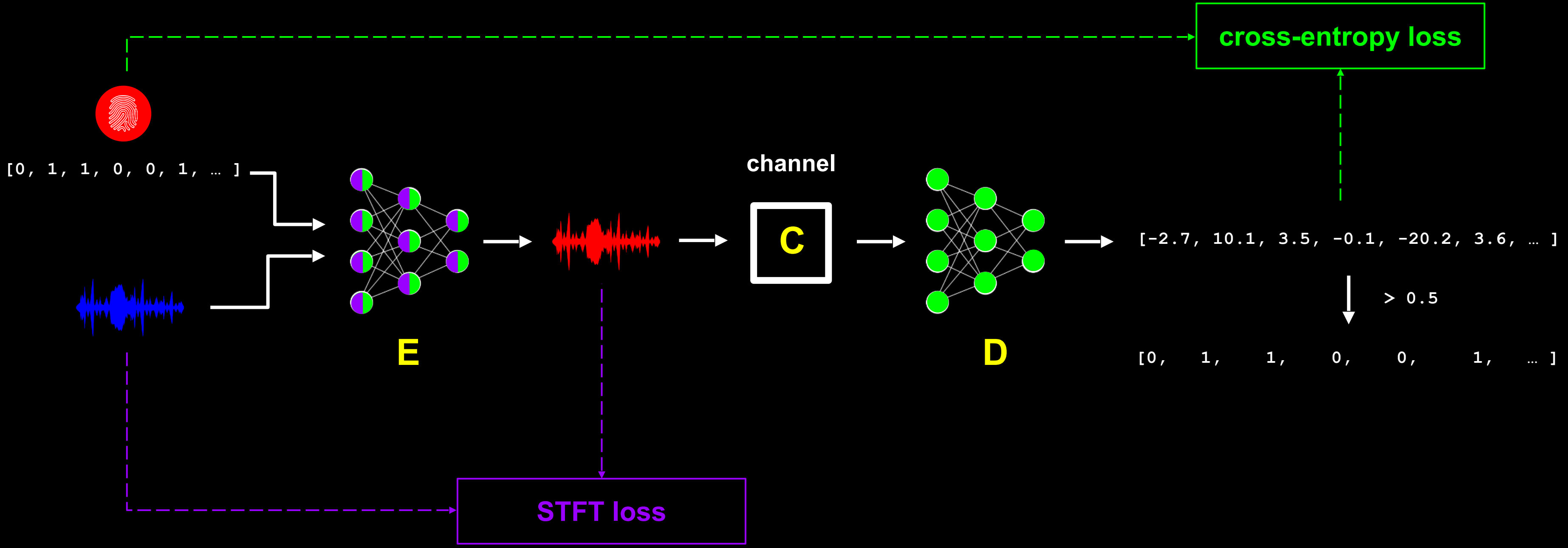


What are we missing?

Let's make E and D neural networks

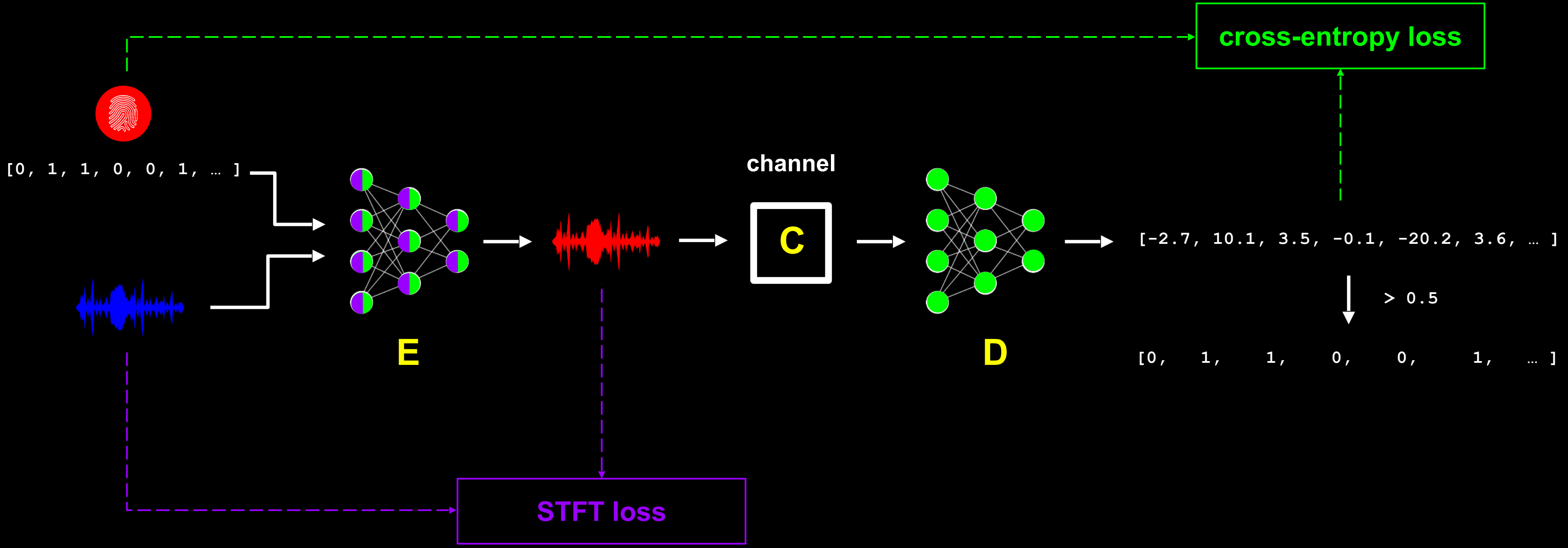


Let’s make E and D neural networks

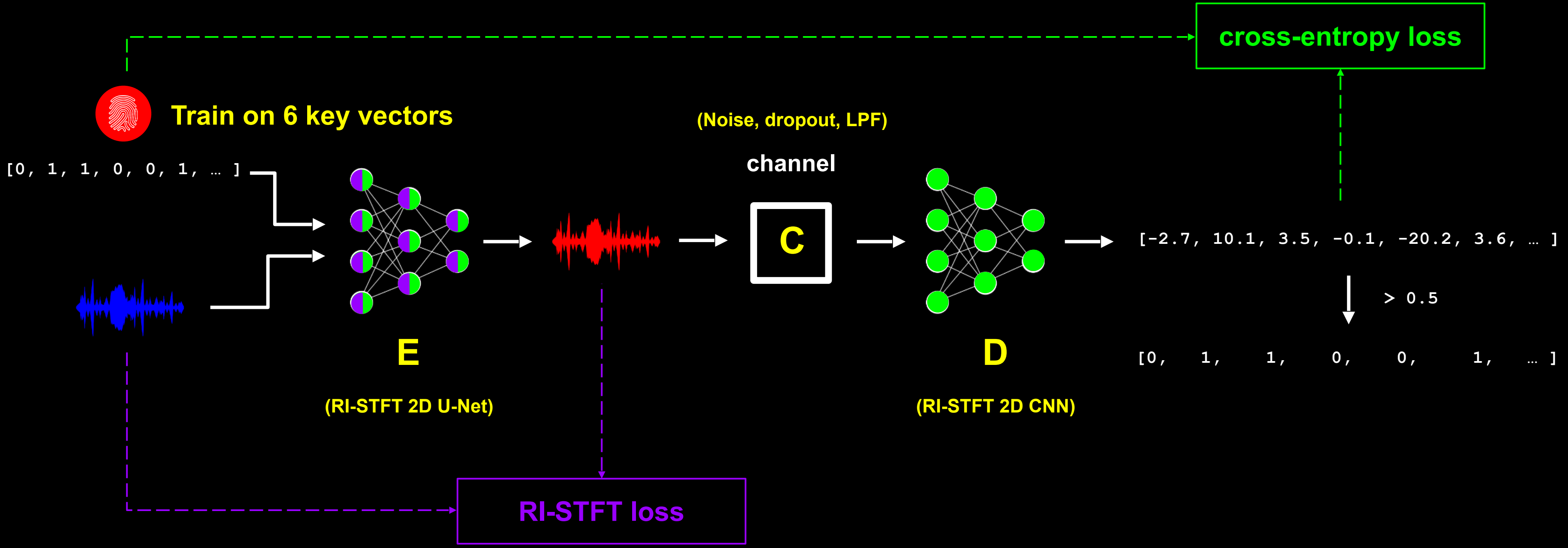


People have tried this!

DNN-A, Pavlovic et al. (2022)



DNN-A, Pavlovic et al. (2022)



Less robust — More robust

	DNN-A (Pavlovic et al. 2022)	EigenWatermark (Tai & Mansour 2019)
Sample rate	16kHz	44.1kHz
Required audio length	2s	1s
Robustness		
Signal-processing		
Neural audio codec		
Neural vocoder		
Neural denoiser		

Less robust → More robust	DNN-A (Pavlovic et al. 2022)	EigenWatermark (Tai & Mansour 2019)
Sample rate	16kHz	44.1kHz
Required audio length	2s	1s
Robustness		
Signal-processing	TPR @ 1% FPR = 0.00	
Neural audio codec		
Neural vocoder		
Neural denoiser		

Less robust — More robust

	DNN-A (Pavlovic et al. 2022)	EigenWatermark (Tai & Mansour 2019)
Sample rate	16kHz	44.1kHz
Required audio length	2s	1s
Robustness		
Signal-processing	TPR @ 1% FPR = 0.00	
Neural audio codec	0.00	
Neural vocoder	0.00	
Neural denoiser	0.00	

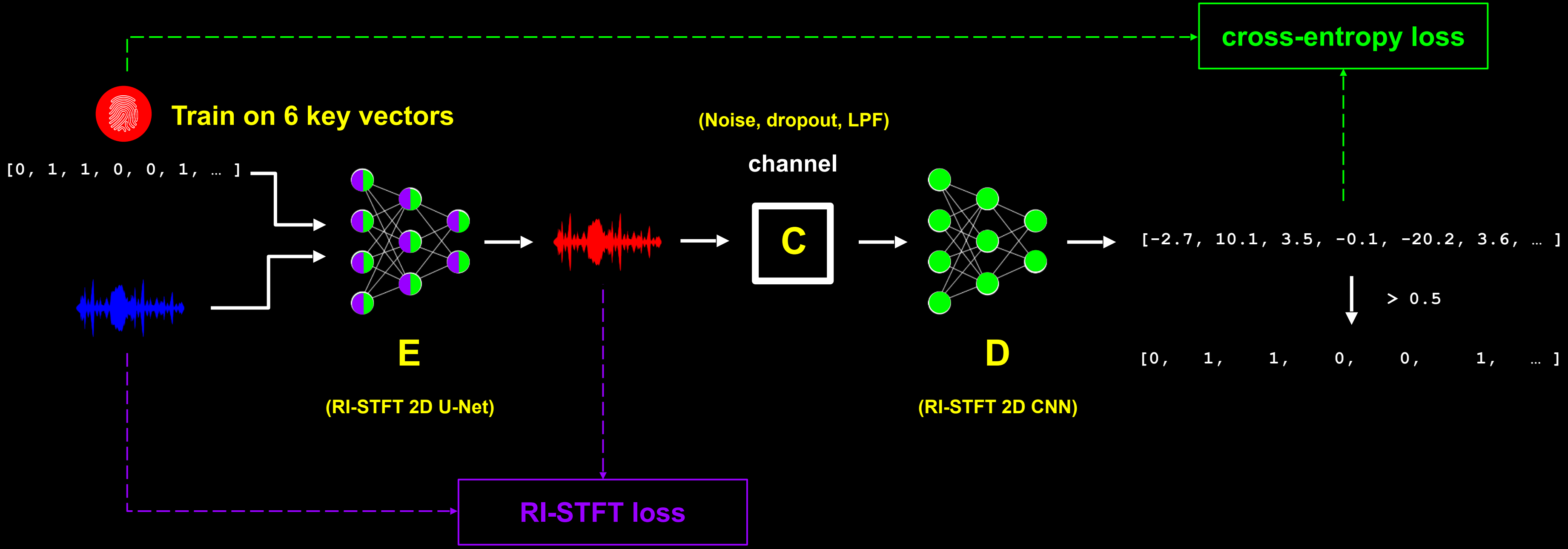
Less robust — More robust

	DNN-A (Pavlovic et al. 2022)	EigenWatermark (Tai & Mansour 2019)
Sample rate	16kHz	44.1kHz
Required audio length	2s	1s
Robustness		
Signal-processing	TPR @ 1% FPR = 0.00	0.73
Neural audio codec	0.00	0.39
Neural vocoder	0.00	0.01
Neural denoiser	0.00	1.00

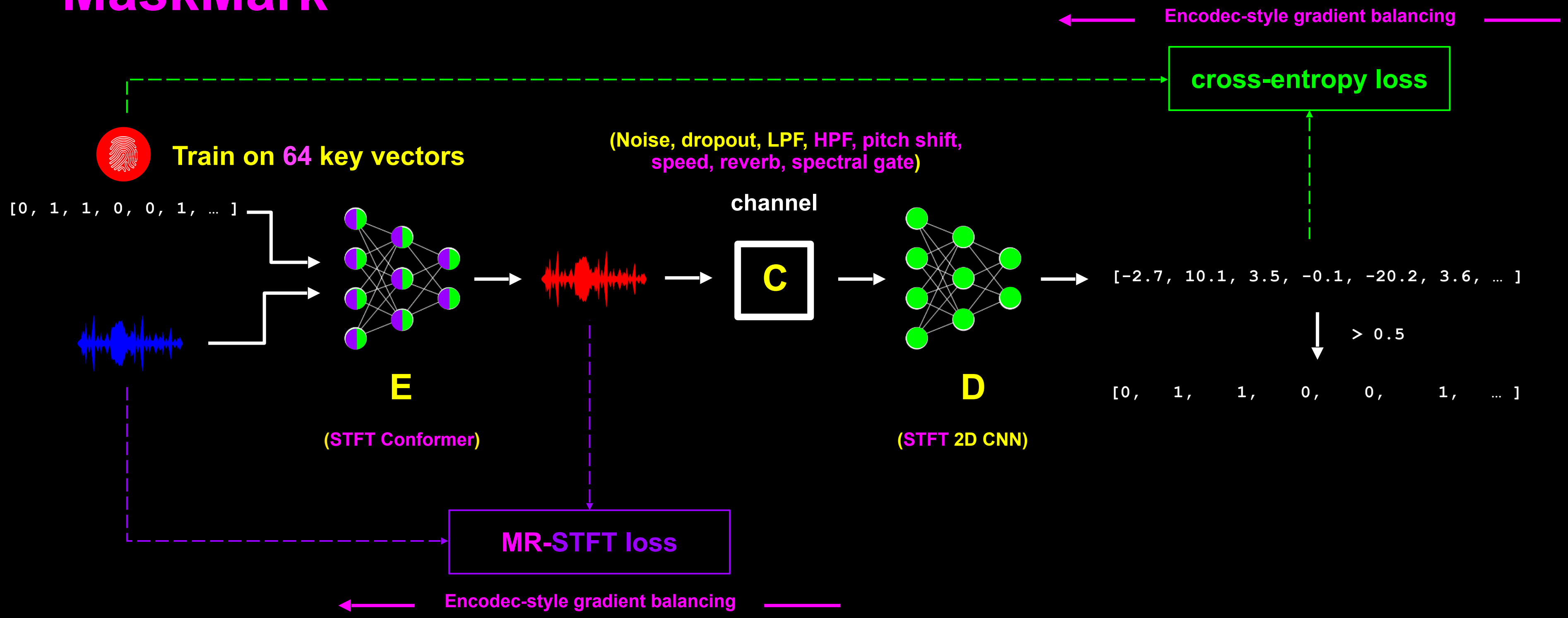
Less robust — More robust

	DNN-A	Eigen	MaskMark
Sample rate	16kHz	44.1kHz	48kHz
Required audio length	2s	1s	1s
Robustness			
Signal-processing	TPR @ 1% FPR = 0.00	0.73	0.97
Neural audio codec	0.00	0.39	0.45
Neural vocoder	0.00	0.01	0.82
Neural denoiser	0.00	1.00	0.99

DNN-A, Pavlovic et al. (2022)

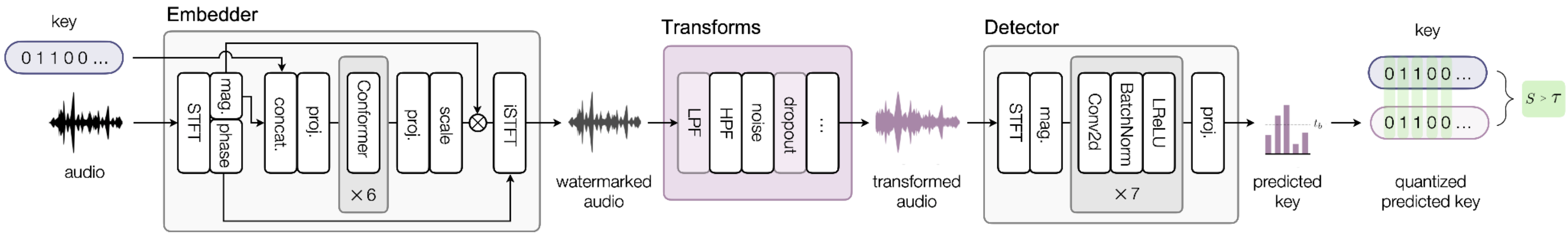


MaskMark

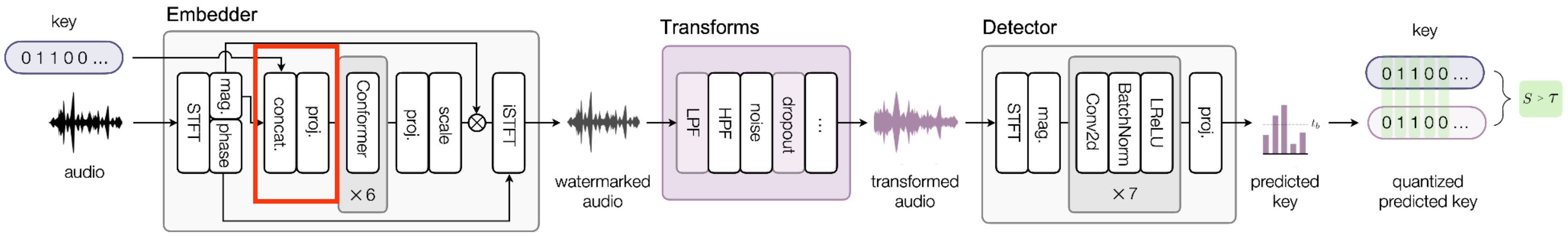


Encodec-style gradient balancing
cross-entropy loss
Train on 64 key vectors
(Noise, dropout, LPF, HPF, pitch shift,
speed, reverb, spectral gate)
channel
[0, 1, 1, 0, 0, 1, …]
C
[-2.7, 10.1, 3.5, -0.1, -20.2, 3.6, …]
> 0.5
[0, 1, 1, 0, 0, 1, …]
E
D
(STFT Conformer)
(STFT 2D CNN)
MR-STFT loss
Encodec-style gradient balancing

Architecture details

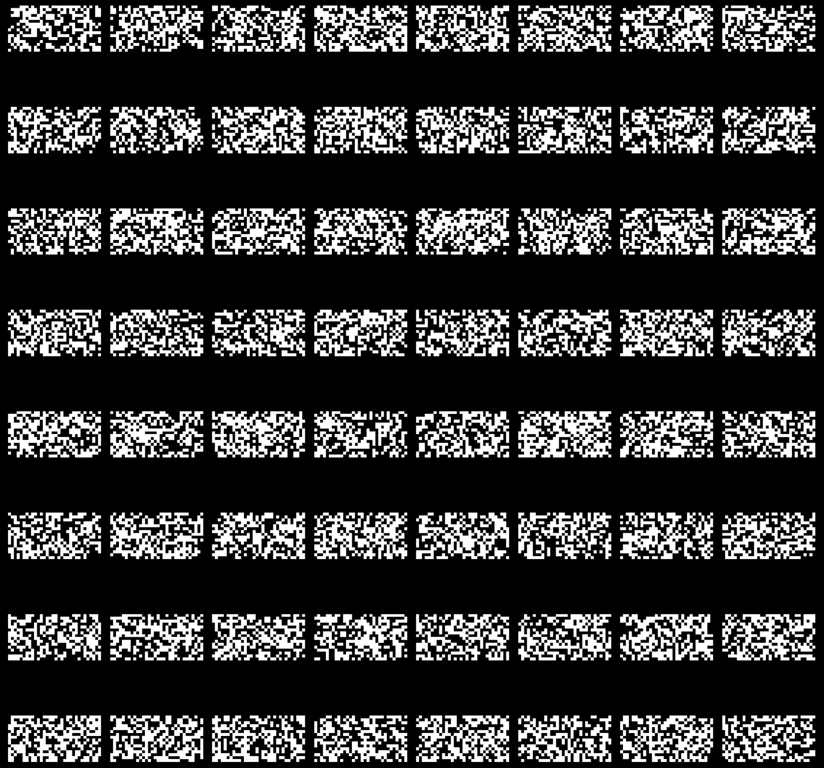


Architecture details



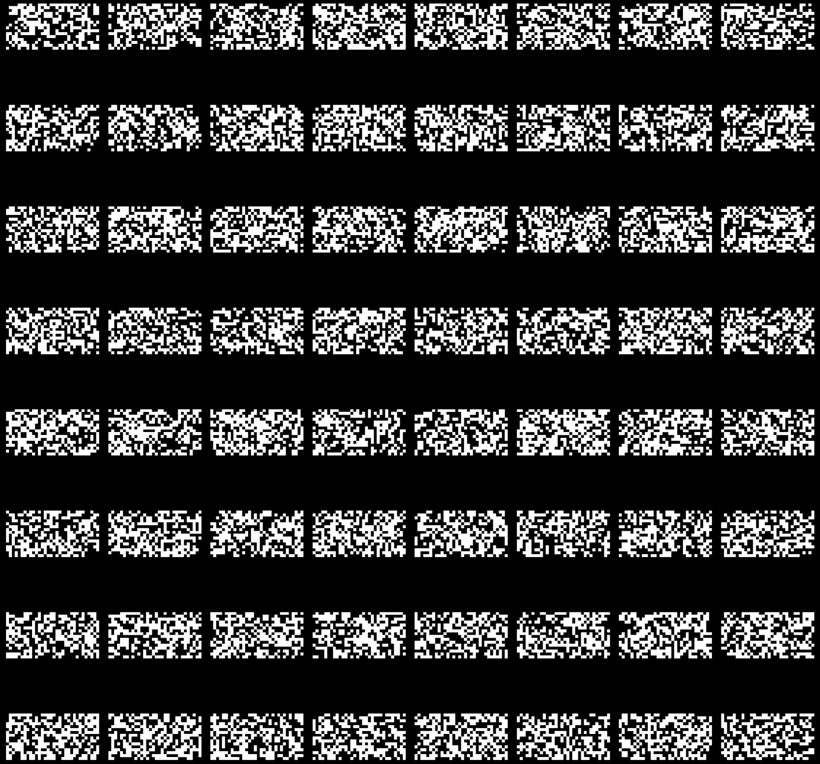
We hide the same watermark information in every frame

No explicit learning of an “un-watermarked” class!



64 learned watermarks

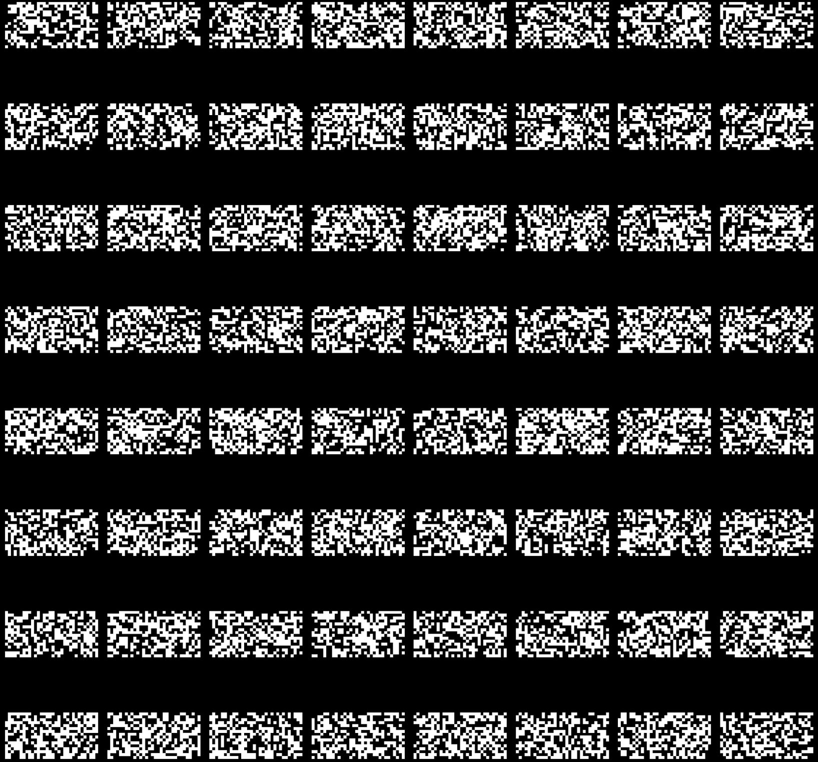
Detector prediction for **unwatermarked** audio (essentially random)



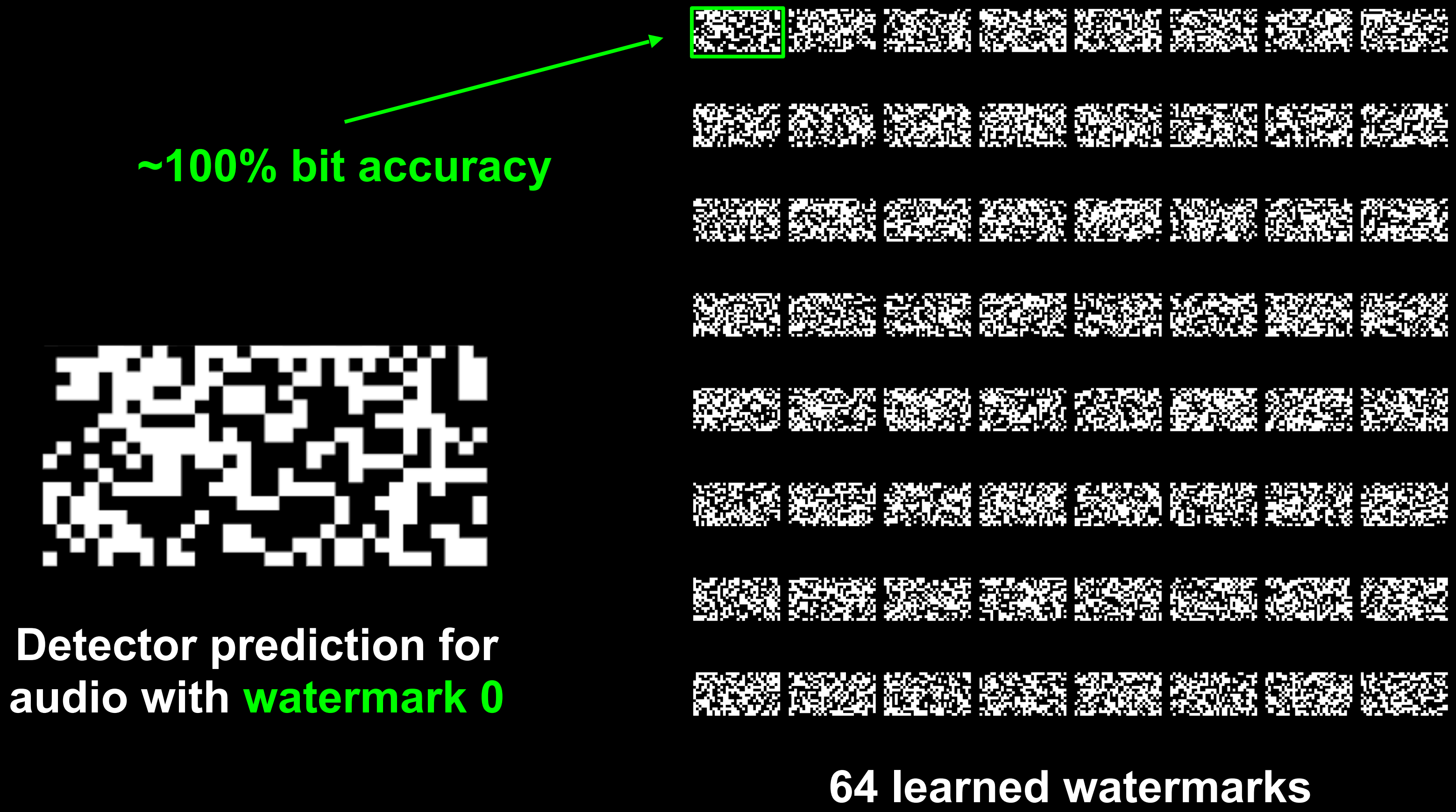
64 learned watermarks

~50% bit accuracy

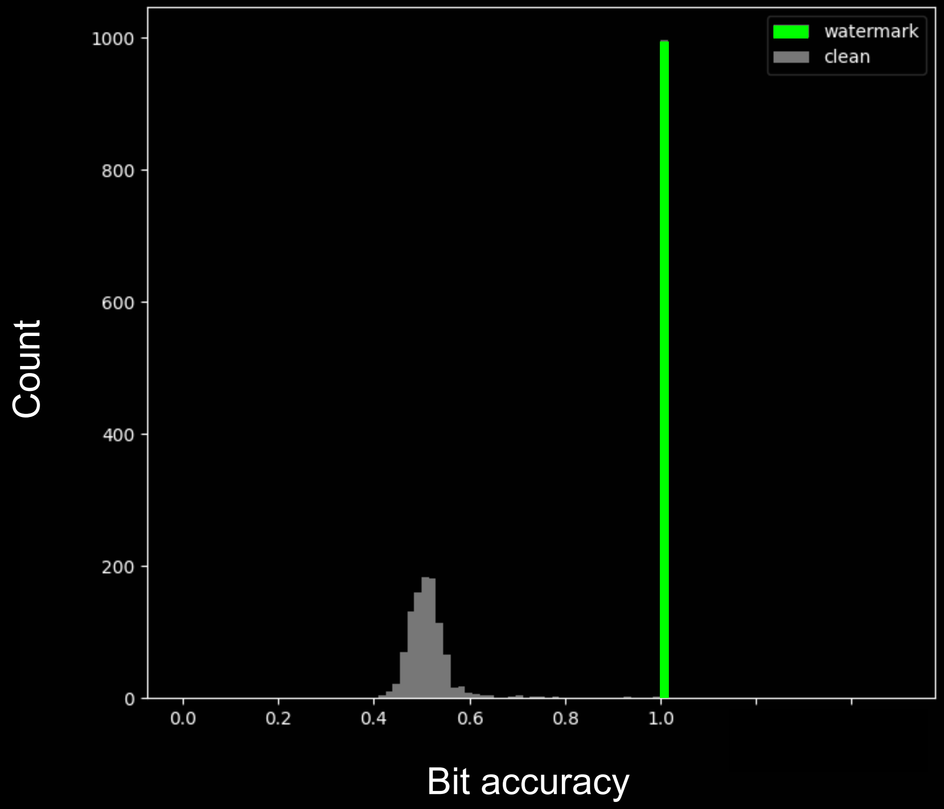
Detector prediction for unwatermarked audio (essentially random)



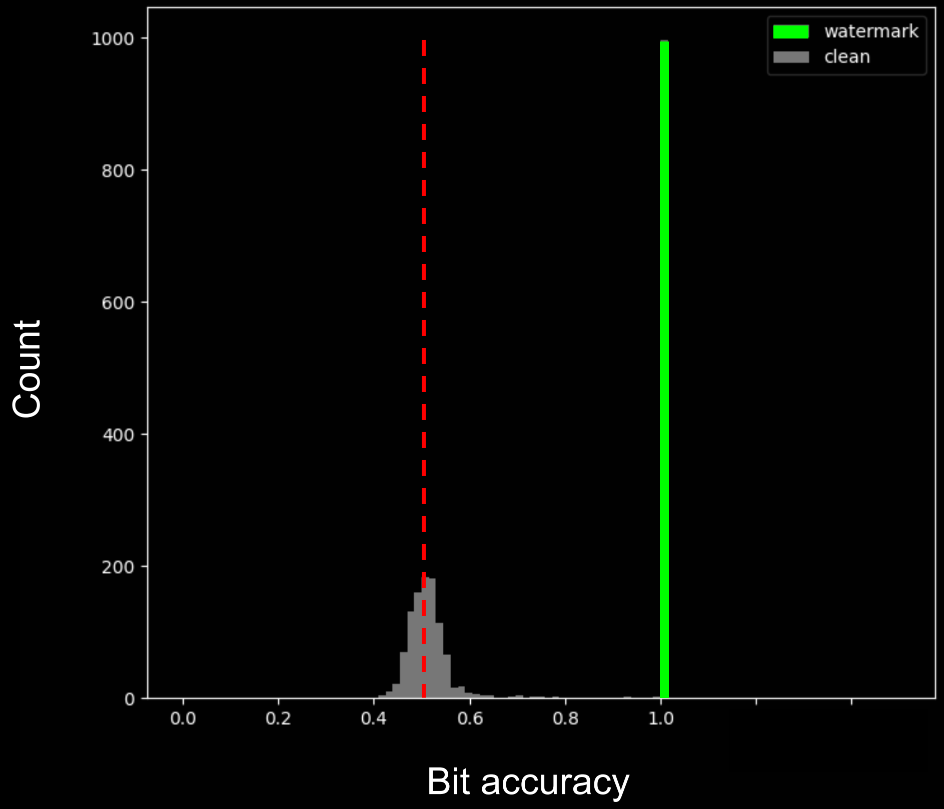
64 learned watermarks


~100% bit accuracy
Detector prediction for
audio with watermark 0
64 learned watermarks

Bit accuracy vs. known key vector, watermarked & un-watermarked audio



Bit accuracy vs. known key vector, watermarked & un-watermarked audio



~50% bit accuracy for unwatermarked

~100% bit accuracy for watermarked

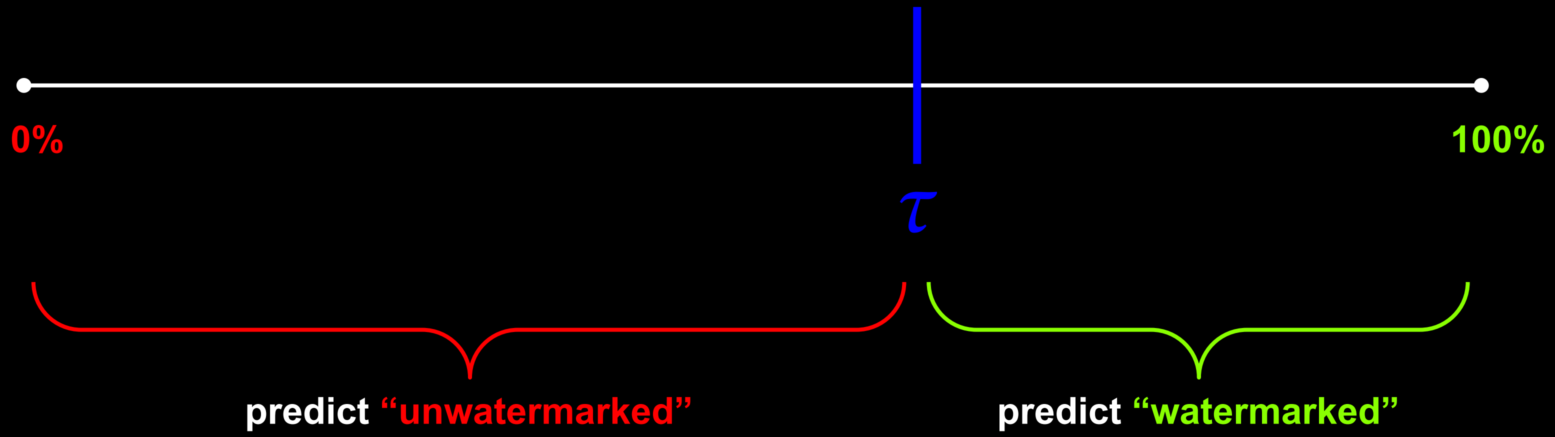
This lets us distinguish between watermarked and unwatermarked audio using bit accuracy

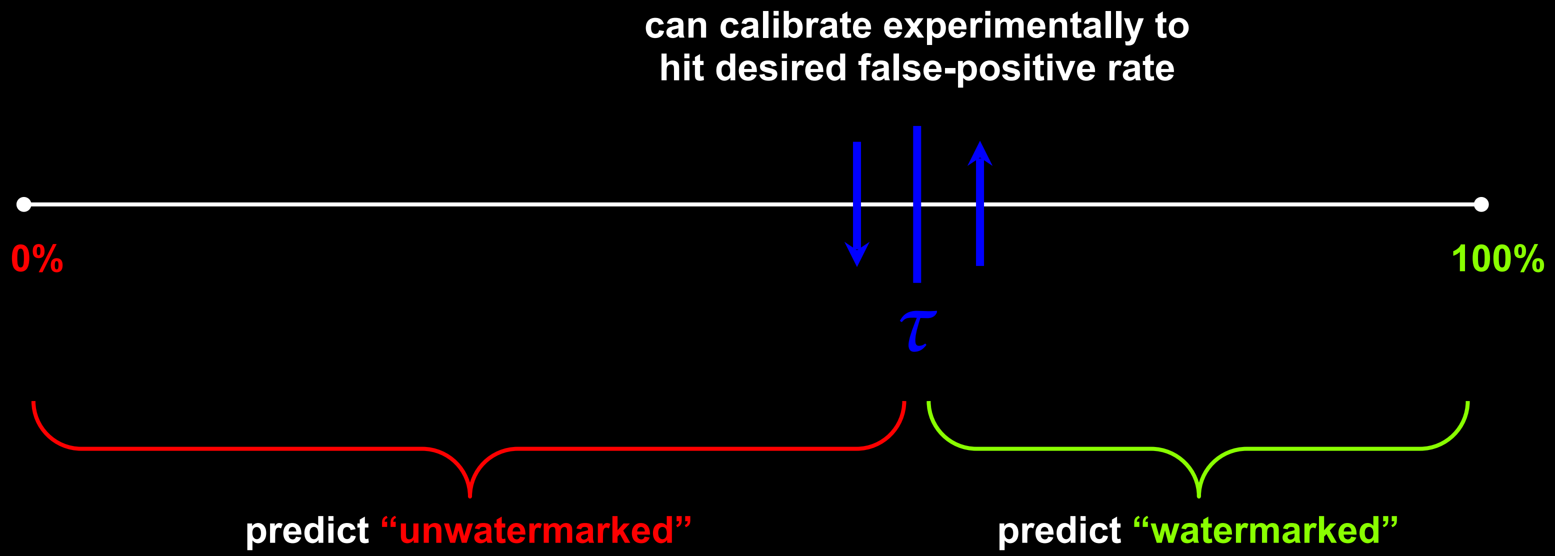
This lets us distinguish between watermarked and unwatermarked audio using bit accuracy

0% 100%

This lets us distinguish between watermarked and unwatermarked audio using bit accuracy

This lets us distinguish between watermarked and unwatermarked audio using bit accuracy



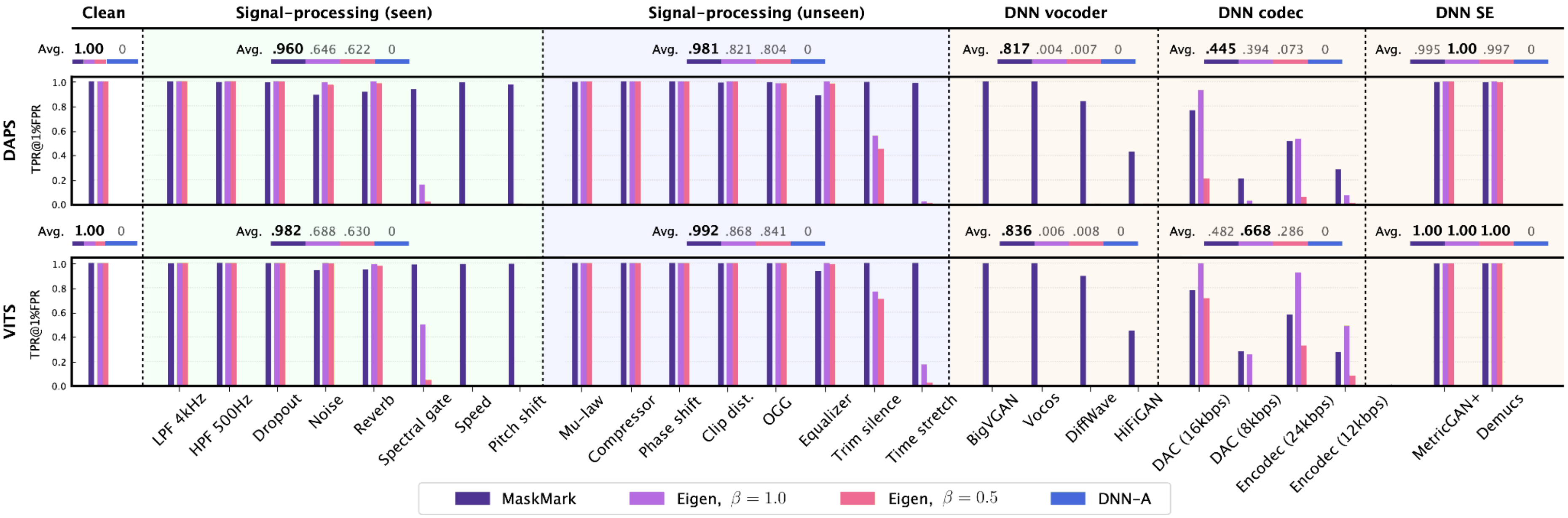


can calibrate experimentally to
hit desired false-positive rate
0%
100%
τ
predict "unwatermarked"
predict "watermarked"

When targeting a low (1%) FPR, our approach outperforms recent signal-processing and neural-network watermarks!

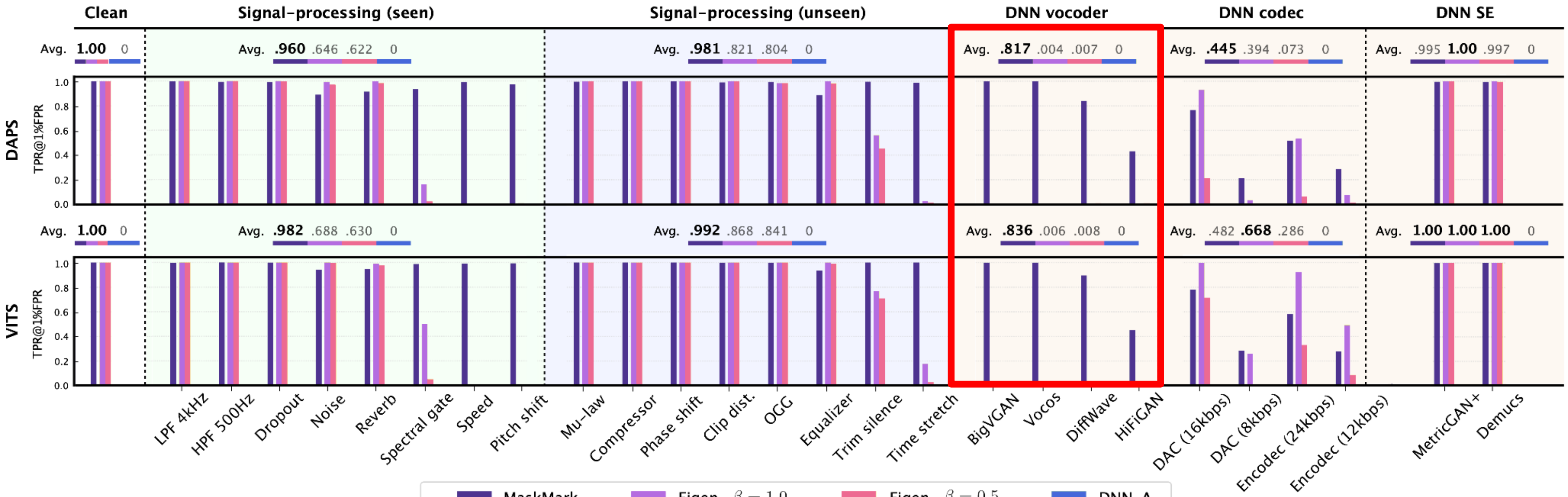
Less robust — More robust

	DNN-A	Eigen	Proposed
Sample rate	16kHz	44.1kHz	48kHz
Required audio length	2s	1s	1s
Robustness			
Signal-processing	TPR @ 1% FPR = 0.00	0.73	0.97
Neural audio codec	0.00	0.39	0.45
Neural vocoder	0.00	0.01	0.82
Neural denoiser	0.00	1.00	0.99

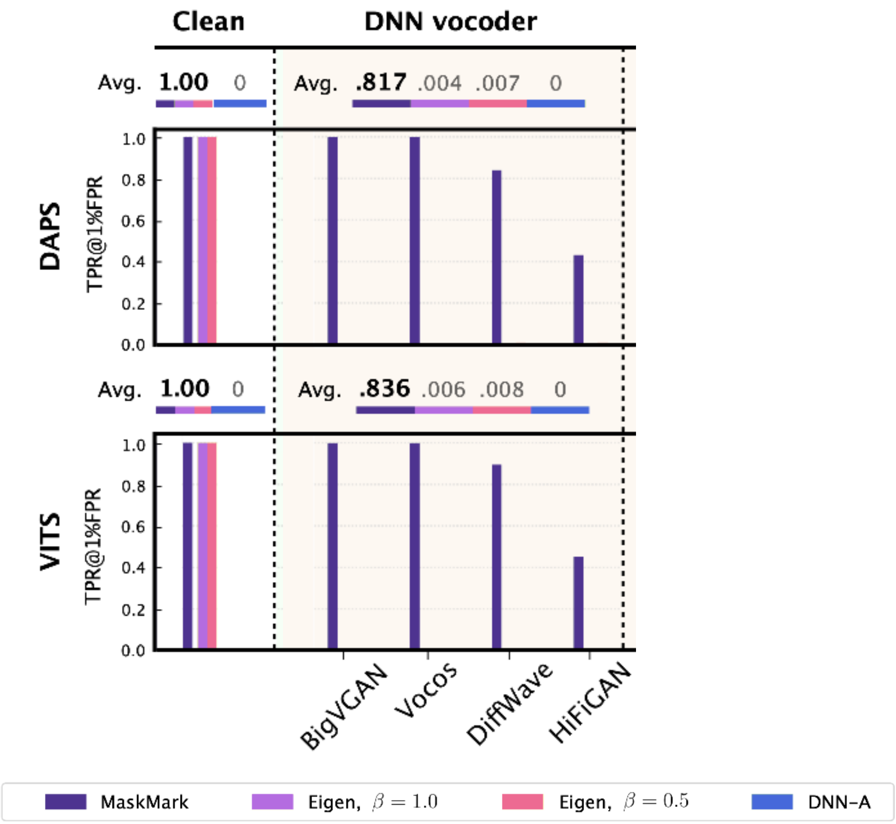


Clean
Signal-processing (seen)
Signal-processing (unseen)
DNN vocoder
DNN codec
DNN SE
DAPS
VITS
TPR@1%FPR
Avg. 1.00 0
Avg. .960 .646 .622 0
Avg. .981 .821 .804 0
Avg. .817 .004 .007 0
Avg. .445 .394 .073 0
Avg. .995 1.00 .997 0
Avg. 1.00 0
Avg. .982 .688 .630 0
Avg. .992 .868 .841 0
Avg. .836 .006 .008 0
Avg. .482 .668 .286 0
Avg. 1.00 1.00 1.00 0
LPF 4kHZ
HPF 500Hz
Dropout
Noise
Reverb
Spectral gate
Speed
Pitch shift
Mu-law
Compressor
Phase shift
Clip dist.
OGG
Equalizer
Trim silence
Time stretch
BigVGAN
Vocos
DiffWave
HiFiGAN
DAC (16kbps)
DAC (8kbps)
Encodec (24kbps)
Encodec (12kbps)
MetricGAN+
Demucs
MaskMark
Eigen, $\beta = 1.0$
Eigen, $\beta = 0.5$
DNN-A

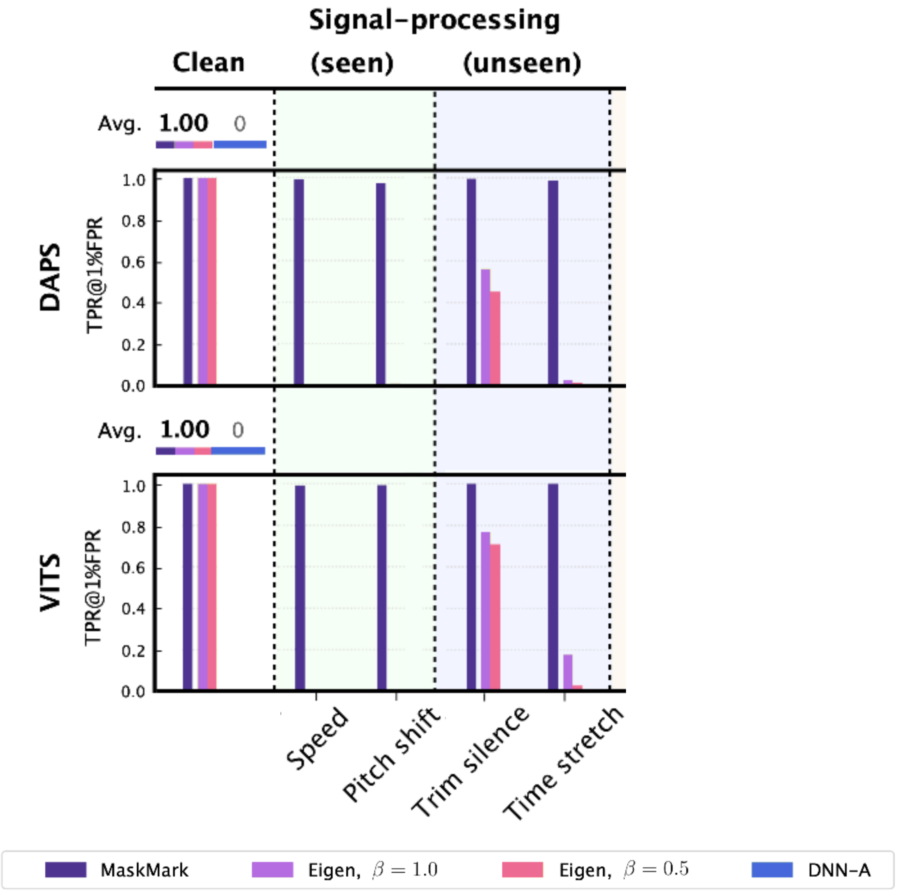
Neural vocoders can wipe out other watermarks while maintaining high audio quality!



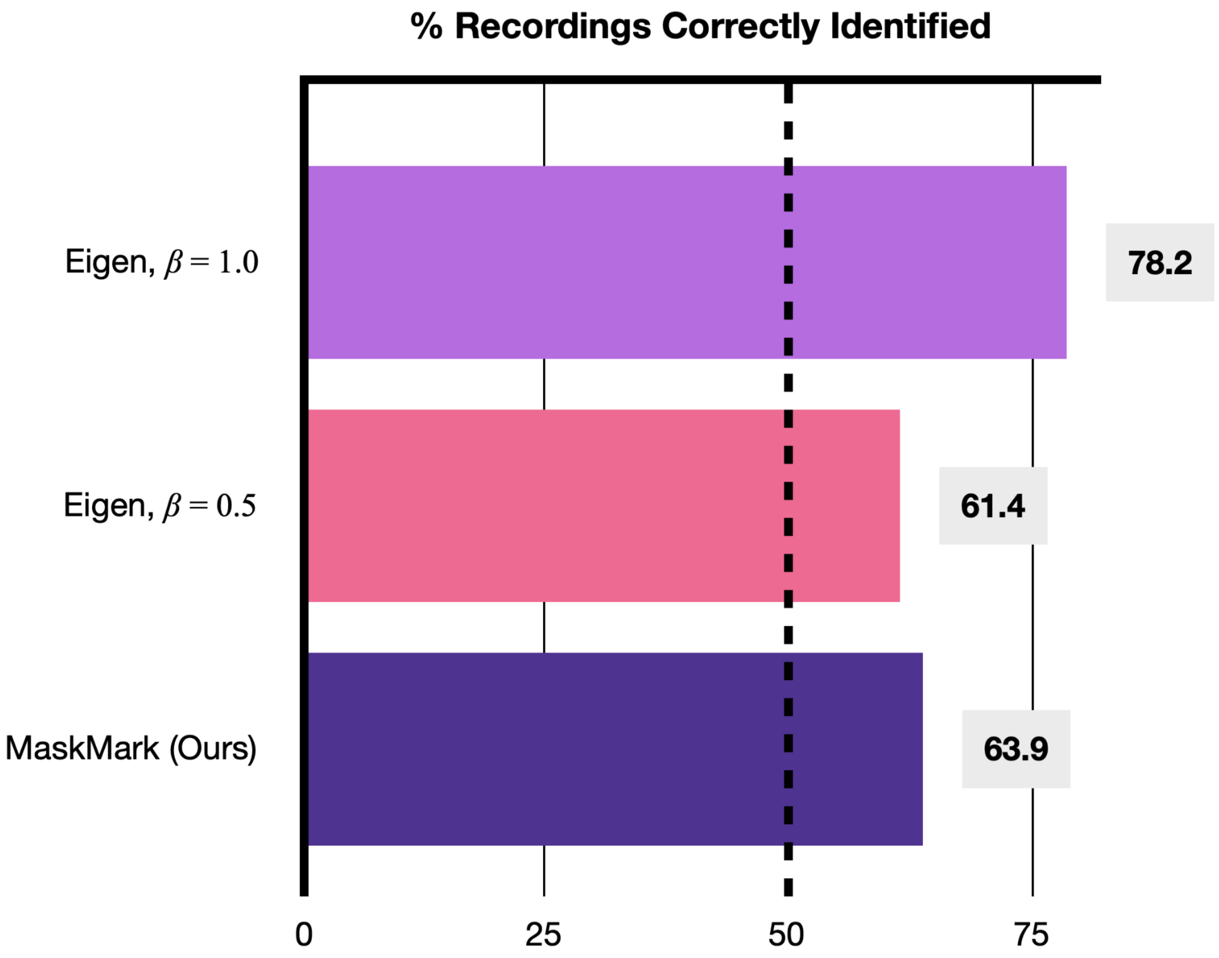
Neural vocoders can wipe out other watermarks while maintaining high audio quality!



Redundant frame-level embedding helps against pitch- and time-scale modification



Our approach preserves audio quality as rated by human listeners.


% Recordings Correctly Identified
Eigen, $\beta = 1.0$
Eigen, $\beta = 0.5$
MaskMark (Ours)
78.2
61.4
63.9
0
25
50
75

Concurrent works:

- **Timbre Watermark** (Liu et al. 2024) uses a similar network design and also demonstrates robustness against neural network-based transformations

- **WavMark** (Chen et al. 2023) uses invertible neural networks to achieve a higher watermark capacity, but considers a narrower and “gentler” set of transformations

- **AudioSeal** (Roman et al. 2024) embeds a residual signal in the time domain and likewise considers a narrower set of transformations

Future directions:

- Improved robustness to neural network-based transformations
- Robustness to adversarial (optimization-based) attacks
- Increased information capacity

MaskMark: Robust Neural Watermarking for Real and Synthetic Speech

Patrick O'Reilly[1], Zeyu Jin[2], Jiaqi Su[2], Bryan Pardo[1]

IEEE ICASSP 2024

1. Northwestern University
2. Adobe Research

https://oreillyp.github.io/maskmark/

MaskMark: Robust Neural Watermarking for Real and Synthetic Speech

(Listening examples)