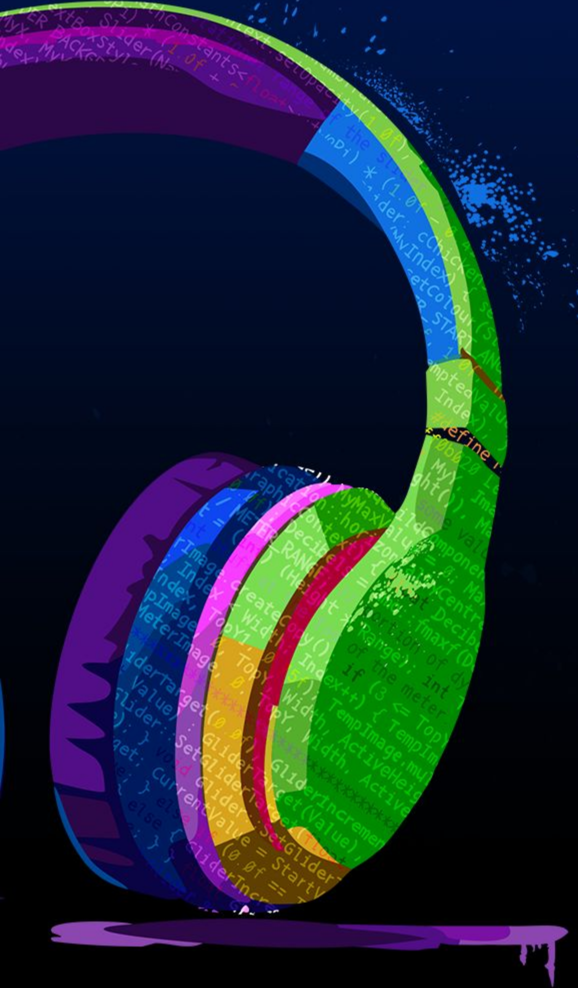


ADC24 Bristol

RESPONSIBLE AI FOR OFFLINE PLUGINS

TAMPER-RESISTANT NEURAL AUDIO WATERMARKING

KANRU HUA

DREAMTONICS

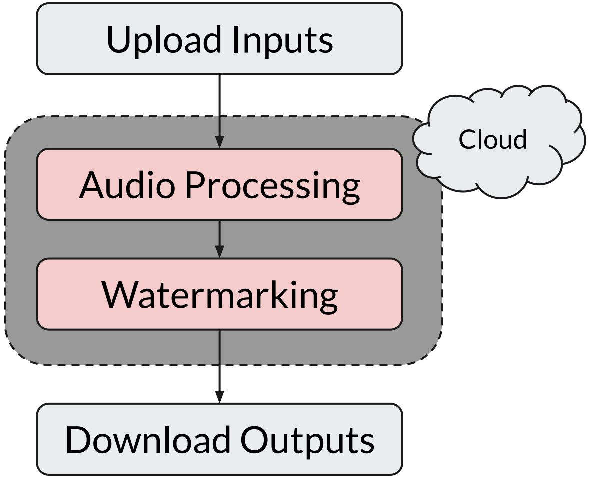
So we are talking about audio watermarks.

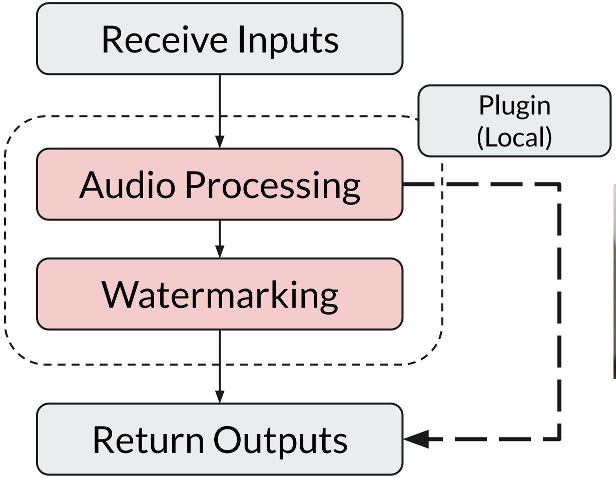
- Add watermarks as a deterrent to unlawful usage.
- Add watermarks to make pirated copies traceable.
- Add watermarks to comply with generative AI regulations.

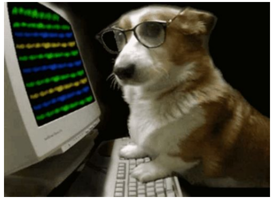
We are not talking about this kind of watermark.

- Too weak
  - Flip the least significant bits
  - Inject “meaningful noises” into inaudible frequencies
- Not traceable
  - Watermark just for verifying if the audio is synthesized

Online vs Offline







Bypassed!

Requirements

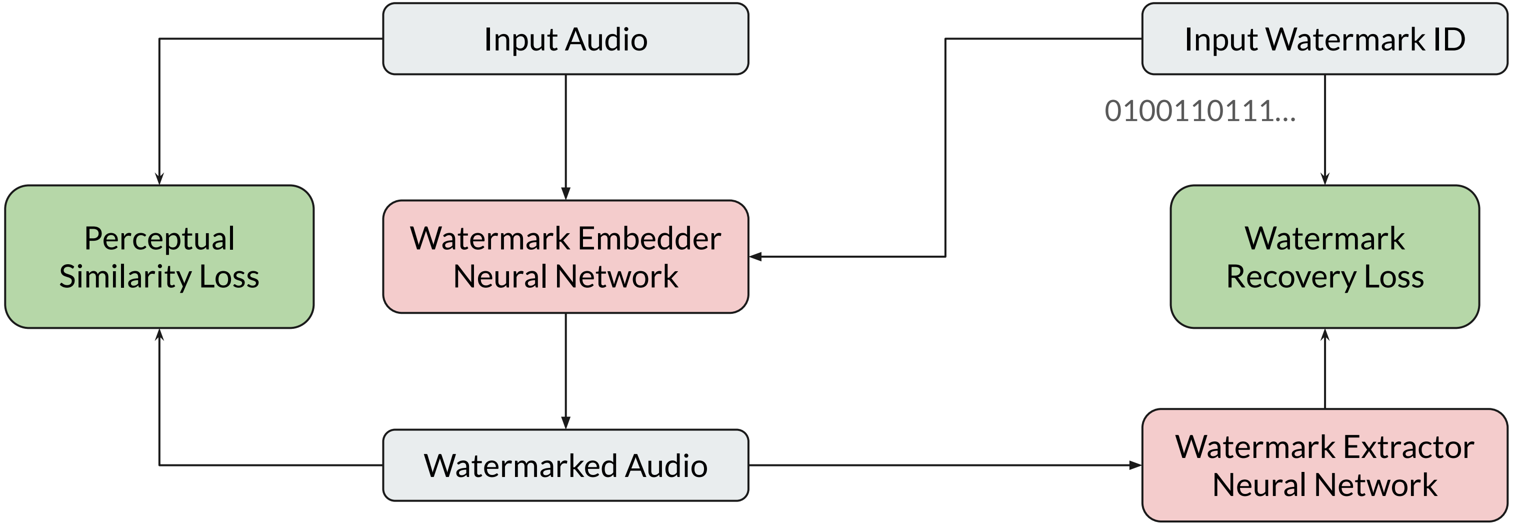
- Uniqueness - each user gets a different ID
- Robustness against audio modification
  - Lossy compression, downsampling, EQ, reverb, distortion
  - Noise and mixing with background music
  - Playback through a speaker and record it with your phone
- Program security
- Low audio latency

Our Starting Point

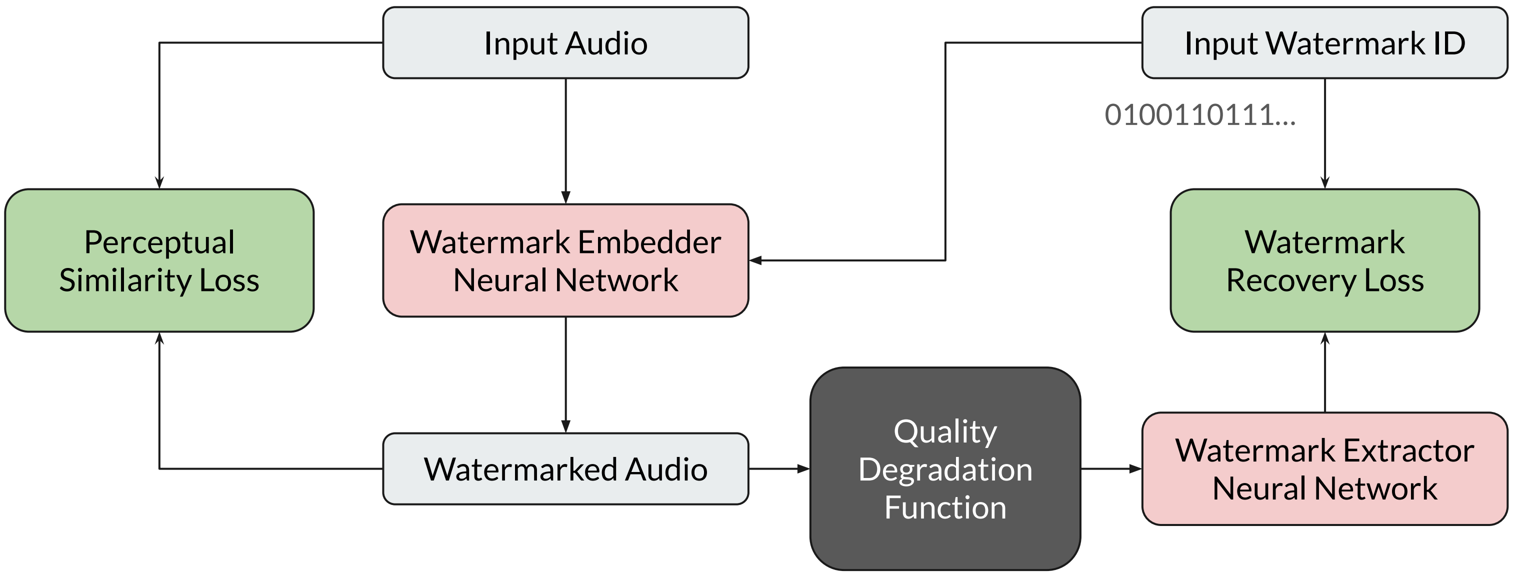
Existing studies on using a neural network to embed watermarks into speech/audio:

- Robust speech watermarking by a jointly trained embedder and detector using a DNN (Pavlović et. al., 2022)
- WavMark: Watermarking for Audio Generation (Chen et. al., 2023)
- DeAR: A Deep-Learning-Based Audio Re-recording Resilient Watermarking (Liu et. al., 2023)

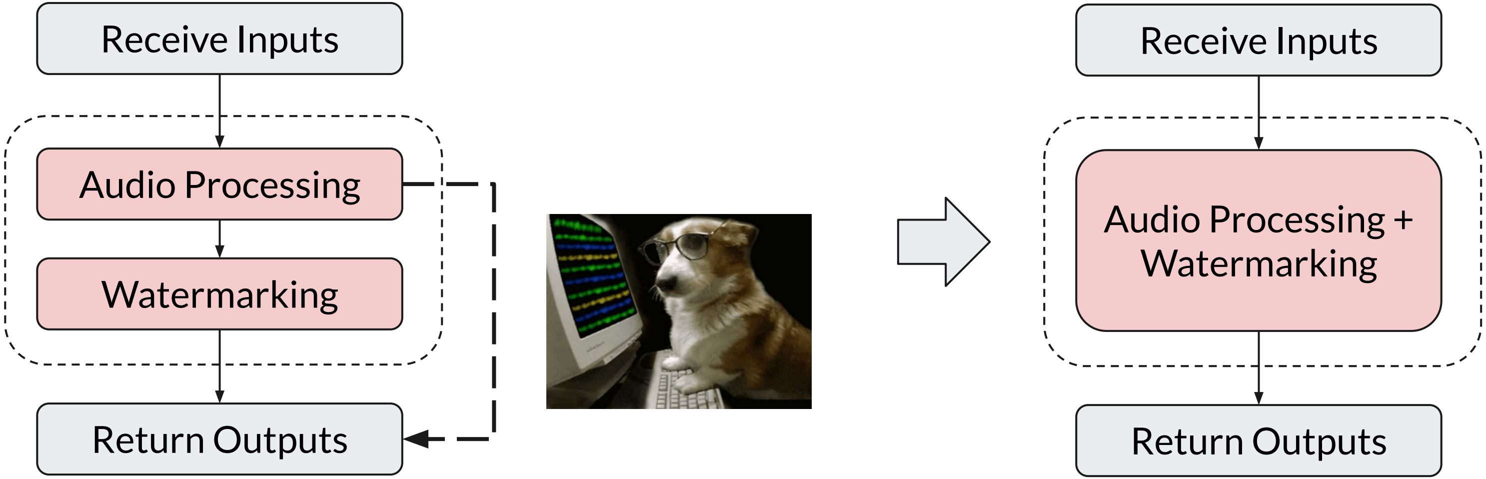
Train Neural Networks for Adding Watermarks!



Train Neural Networks for Adding Watermarks!

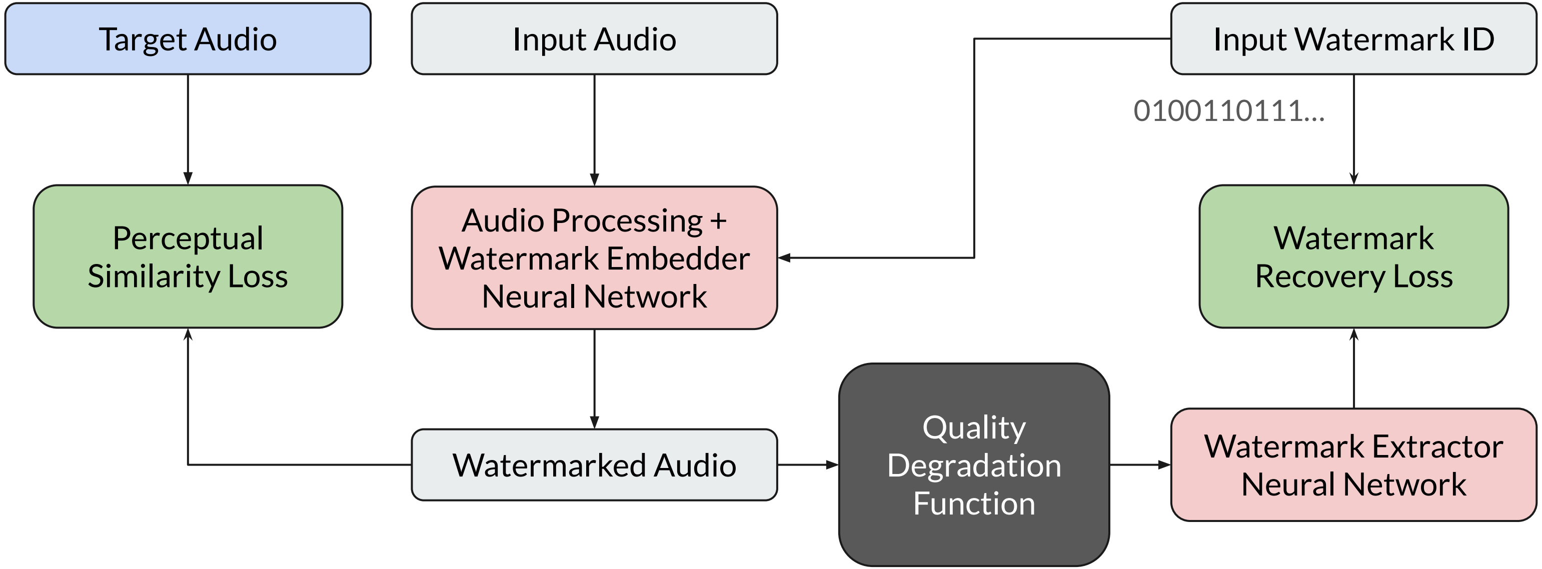


Fuse the Watermark NN with Audio Processing NN

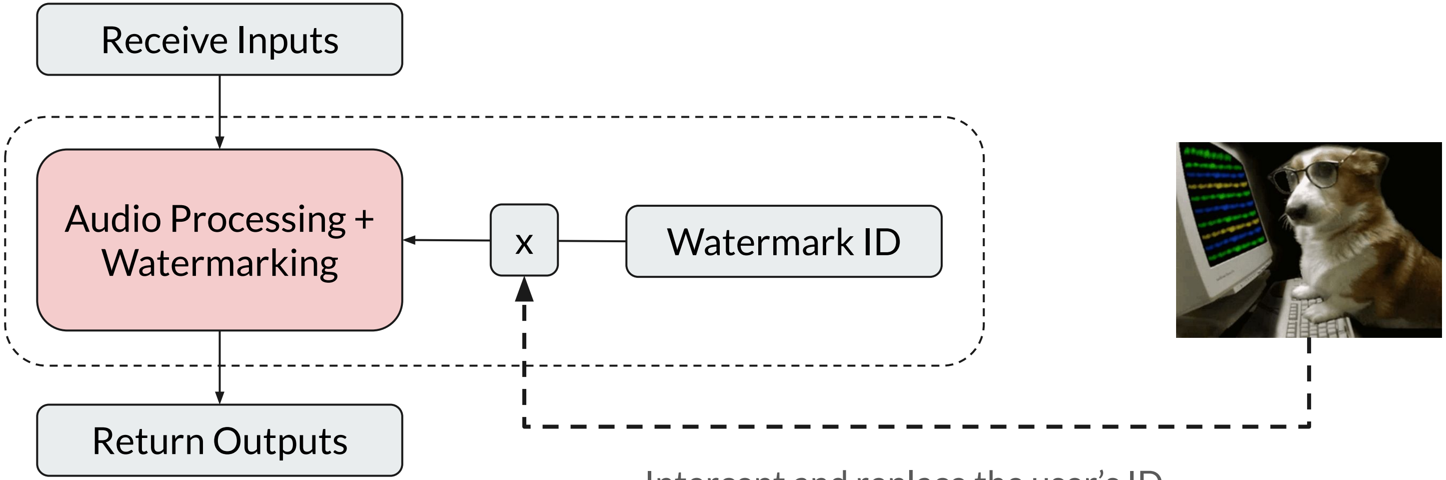


DREAMTONICS

Fuse the Watermark NN with Audio Processing NN

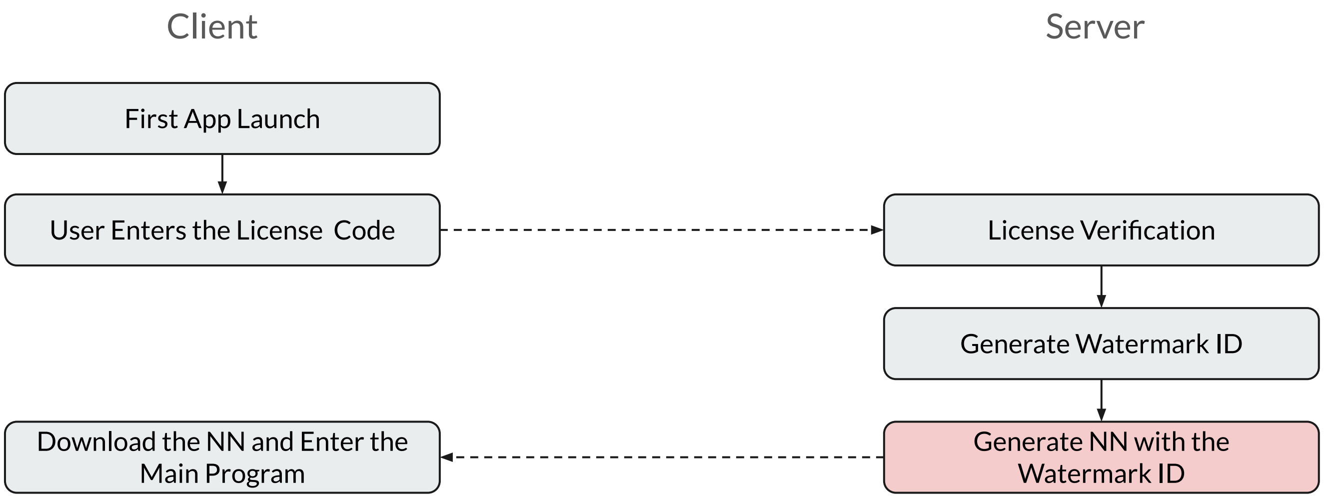


Fusing does not solve all problems...

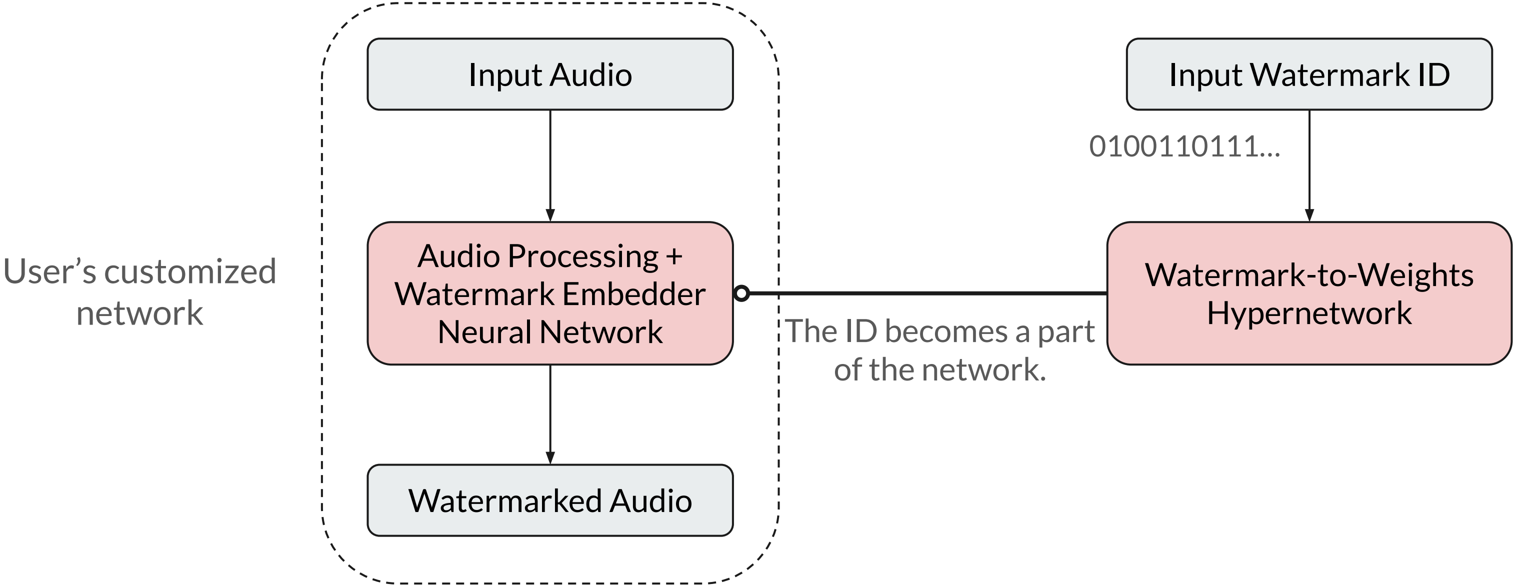


Intercept and replace the user’s ID

Personalized NNs



Fuse Each User's ID into the Network



A close-up look at the Hypernetwork

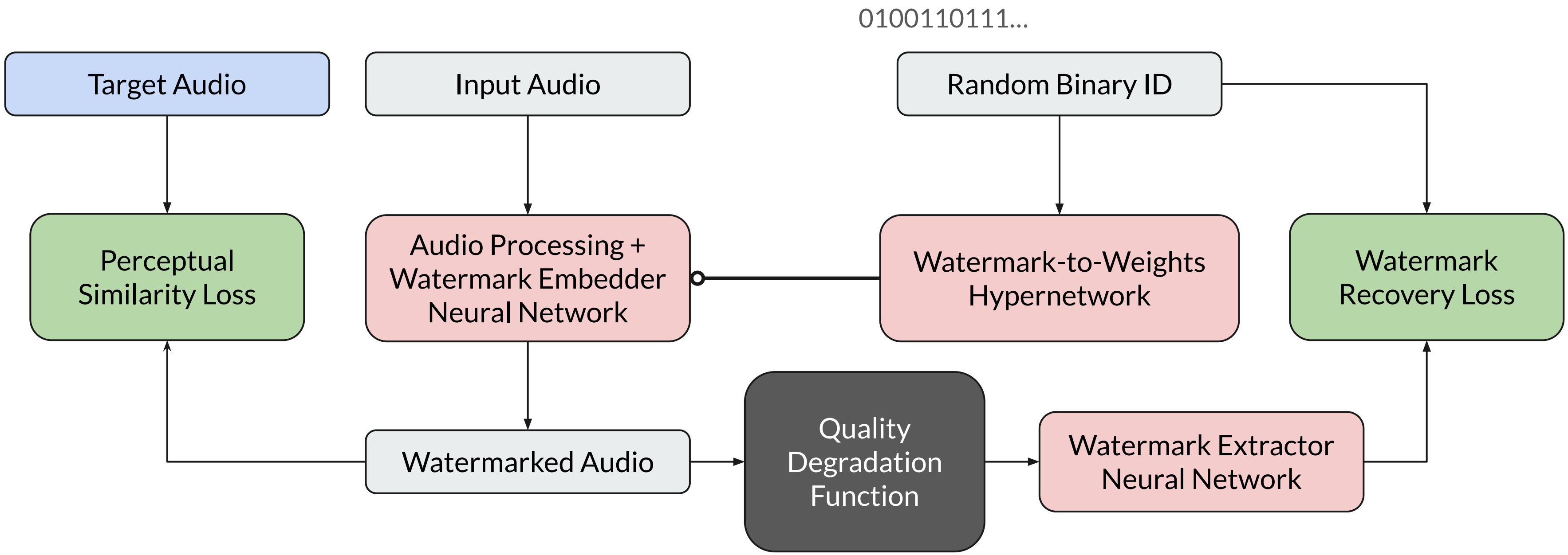
Fully connected layer: $y = \sigma(Wx + b)$

- x: input, y: output
- W: weights matrix, b: bias vector
- σ: non-linear function

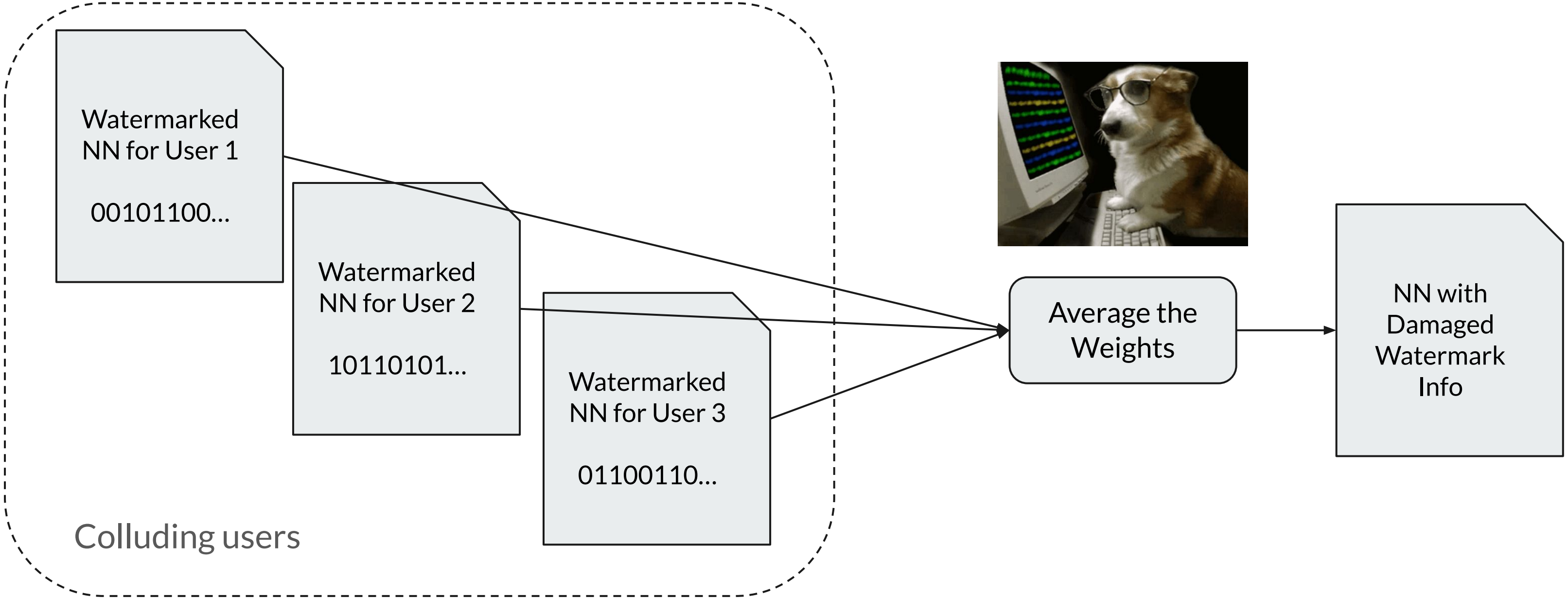
Fully connected layer parameterized by a hypernetwork: $y = \sigma\left((W_0 + f_{\mathrm{hyp}}(m))x + b\right)$

- m: binary ID sequence
- f_hyp: weights-generating hypernetwork
- W_0: “default” weights

Training the Hypernetwork



Collusion (not Collision!) Attack



Anti-Collusion Codes

Mathematically construct a list of IDs such that out of a total of N users,

- Any combination (using the logical AND operator) of IDs from up to a subset of K users can still be uniquely identified.

Example (N = K = 4)

- 1110
- 1101
- 1011
- 0111

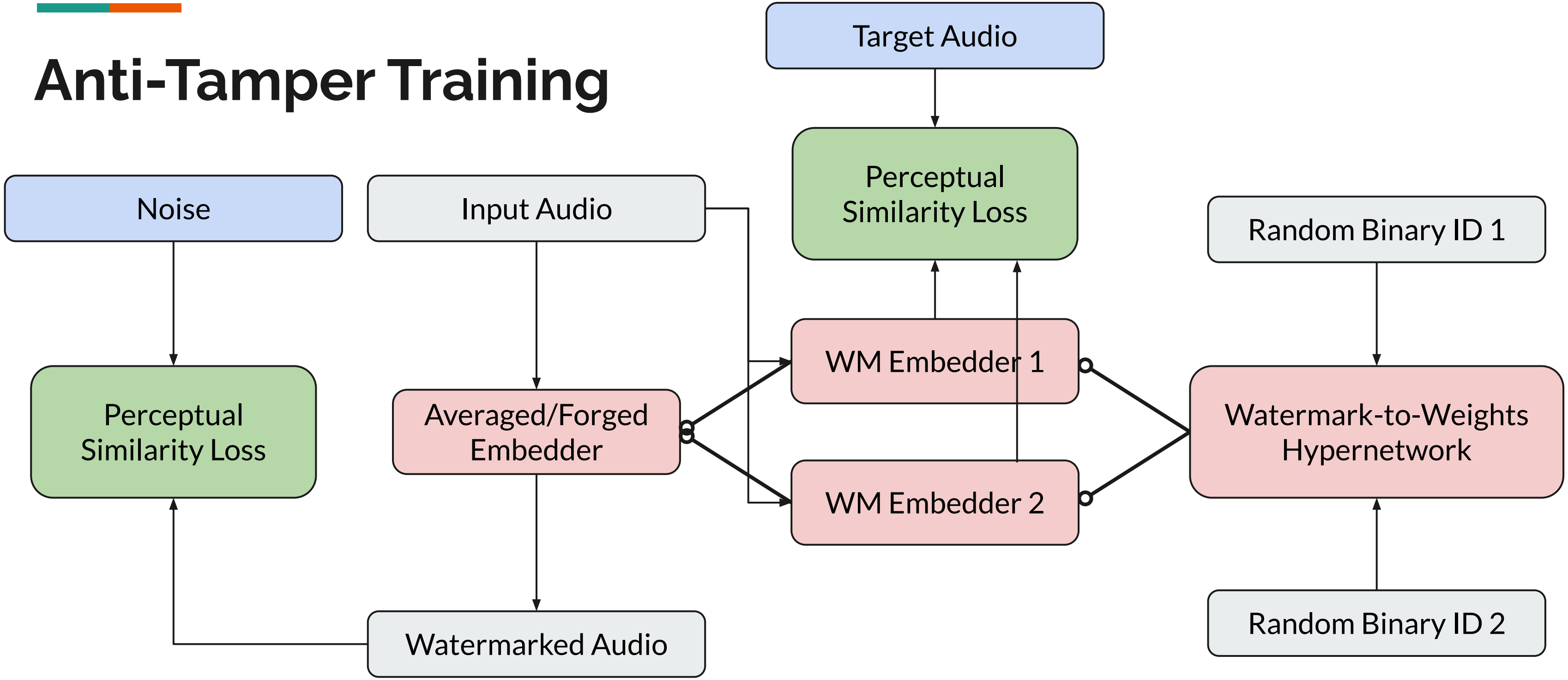
Techniques such as BIBD-AND-ACC (W. Trappe, M. Wu, Z.J. Wang, and K.J.R. Liu, 2003) offer protection using $O(\sqrt{N})$ bits.

Anti-Tamper Training

Train our all-in-one NN to contaminate the output when its weights are modified.

If you damage the watermark, you'll damage the audio as well.

Anti-Tamper Training



Connecting the Dots

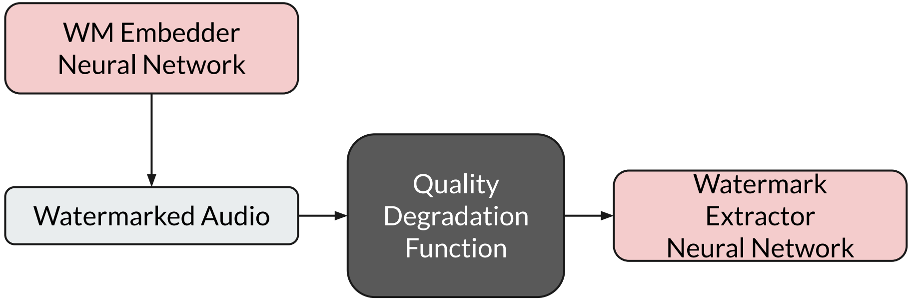
- Introduce differentiable quality degradations during training
- Fuse the audio processing/generation network with the watermark embedder network
- Internalize the unique watermark ID using a hypernetwork
- Anti-collusion codes for watermark ID generation
- Anti-tamper training

Implementation Tips

- For real-time applications, make use of causal operators in the watermarking NN to minimize latency. However the watermark extractor NN does not have to be causal.
    - Lightweight watermarking NN + Heavyweight extractor NN
- Anti-tamper training can be unstable due to its sensitivity to small differences in the weights.
    - First train without the anti-tamper objective, and then fine-tune.
- The weights generating hypernetwork can be prohibitively huge and won't fit in GPU ram.
    - Make it very, very sparse.

How robust does it need to be?

- Under heavy noises and distortions, 100% watermark recovery is not possible.
- Make watermarks too strong and the audio quality will suffer.
- When designing and testing our watermarking system, what is an acceptable recovery rate?
- How many bits do we need?



Goals for Watermark Recovery Accuracy

Out of a total of N users, to uniquely identify a user at a given confidence level, how many bits out of a watermark ID of B bits need to be accurately recovered?

- What is the probability for the user associated with the ID to get more bits recovered than all the other N - 1 users?

For this to not happen for all the rest of the users.

$$\mathrm{P}_{\mathrm{minD}} = \mathrm{E}_{\mathbf{k} \sim \mathrm{Binom}(B, \mathrm{p_{rec}})} \left[\mathrm{P}^{N-1}(\mathbf{X} < \mathbf{k} | \mathbf{X} \sim \mathrm{Binom}(B, \frac{1}{2})) \right]$$

The case when k out of B bits are accurately recovered

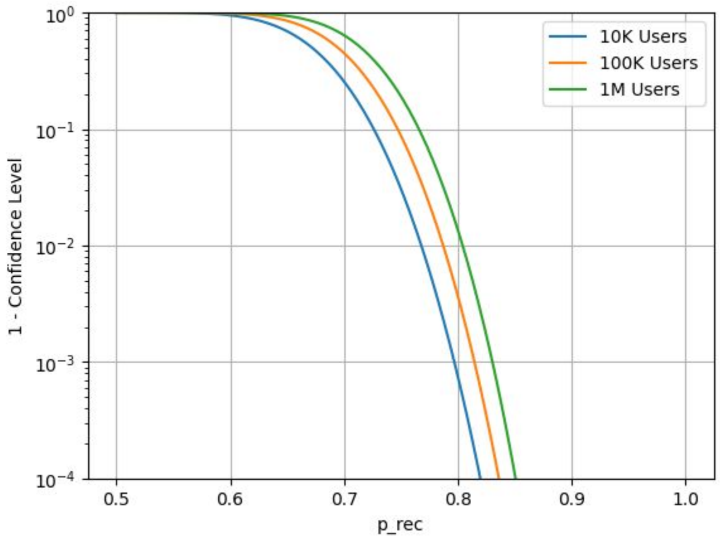
The chance for an unrelated ID to not match (out of pure luck)

Goals for Watermark Recovery Accuracy

Example:

- B = 128 bits (16 bytes)
- N = 10K / 100K/ 1M users

Recover more than 85% of the bits and you'd be pretty safe!

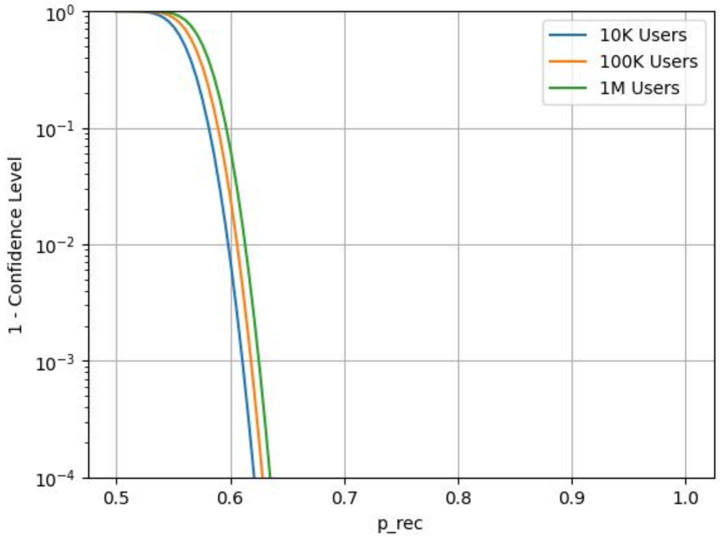


Goals for Watermark Recovery Accuracy

Example:

- B = 1024 bits (128 bytes)
- N = 10K / 100K/ 1M users

You only need it to perform barely better than random, however at the cost of needing to hide a lot more information.



Why? And what to do next?

This is probably the first documented approach of designing an offline-deployable, user-unique, anti-tamper audio watermarking system.

We want your engineers to take this as an inspiration and design your own watermarking system.

However, *don't copy it in verbatim*.

The more diversity there is among our approaches, the harder it is to get targeted & bypassed while being treated as the same class of methods.