# Pruning of an Audio Enhancing Deep Generative Neural Network

Simon Plain

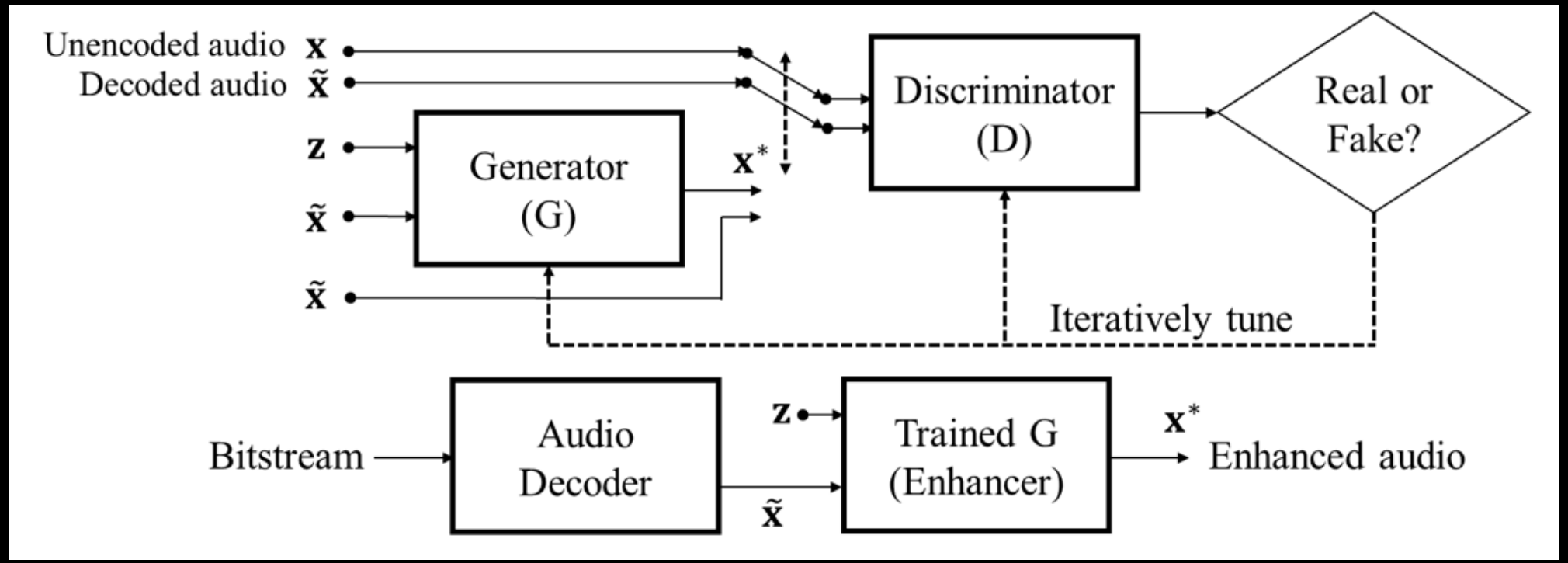
Arijit Biswas

# Paper Overview

- Generative Adversarial Network
- Coded Audio Enhancement
- Pruning/Sparsification
- Listening Test Results
- Discussion

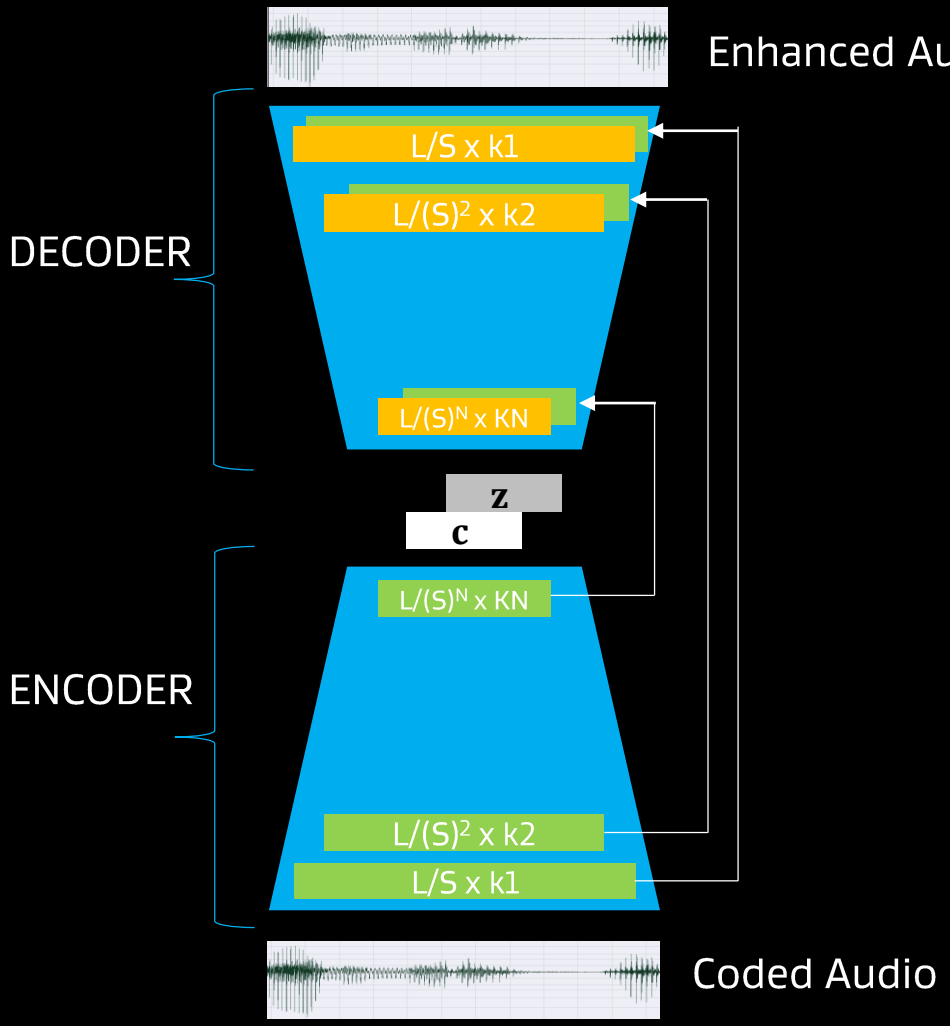
# Starting Point and Motivation

- Generative Adversarial Networks (GAN) for enhancing coded audio[1]
  - Demonstrated to enhance speech and applause



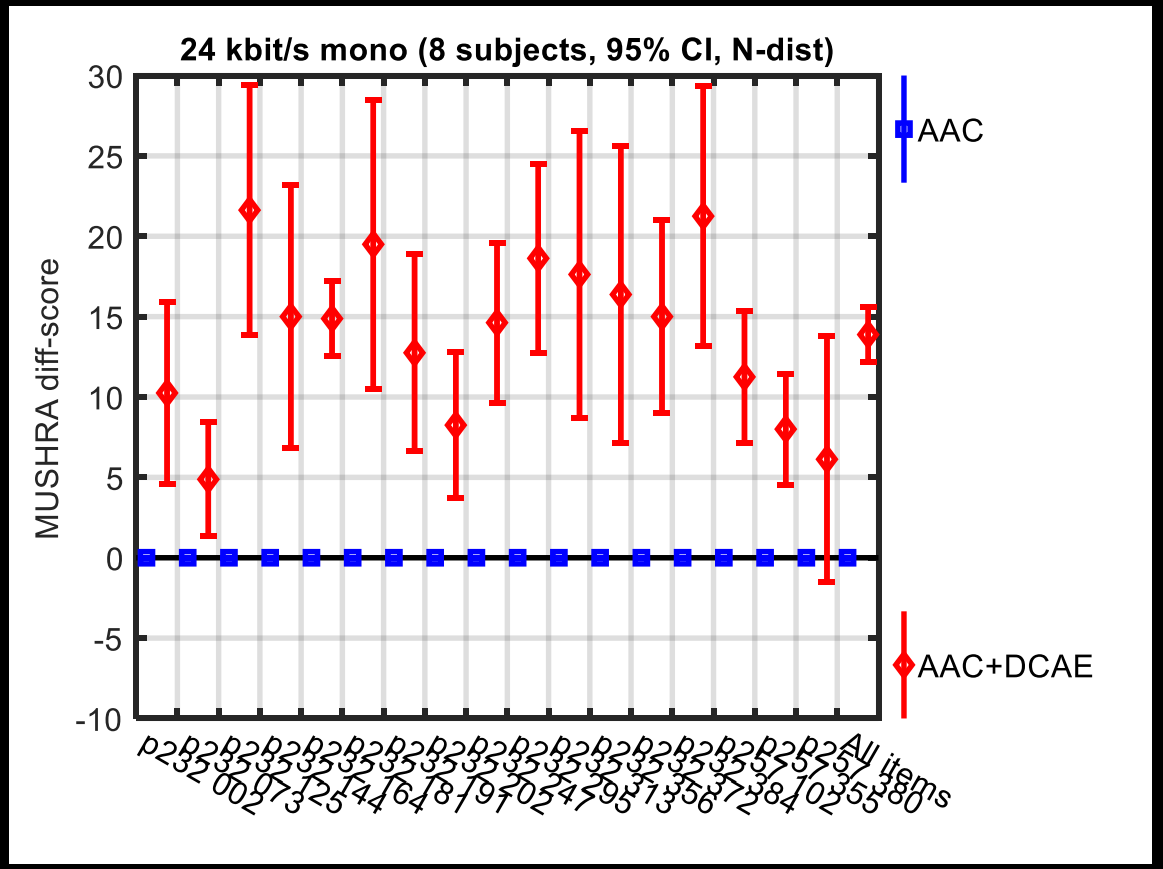

[1]A. Biswas and D. Jia, "Audio Codec Enhancement with Generative Adversarial Networks" *ICASSP 2020.*

# Generator – The Model to be Pruned/Sparsified




- 1D fully convolutional auto-encoder with non-linear activations
  - Bottleneck: $\mathbf{c}$
  - $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ concatenated at bottleneck: adds stochastic behavior to generator predictions
- Skip connections
  - Generated audio maintains fine structure of the coded audio

# Listening Test – AAC @ 24 kbit/s Mono (Speech – VCTK Test Set) Pre-Pruning




Trained on VCTK training set: 28 speakers (14 male, 14 female) with mix of regional English accents

# Why prune/sparsify networks?

Memory and processing requirements

- Achieving high accuracy requires deep and wide networks
- We want to be able to deploy Generator on complexity constrained resources

How compressible is our model?

- It is well-known that training a GAN is a difficult task

  → Pruning a Generator in a GAN setting should be even more challenging

- Literature seem to suggest that standard weight pruning techniques cannot be applied to GANs[1]

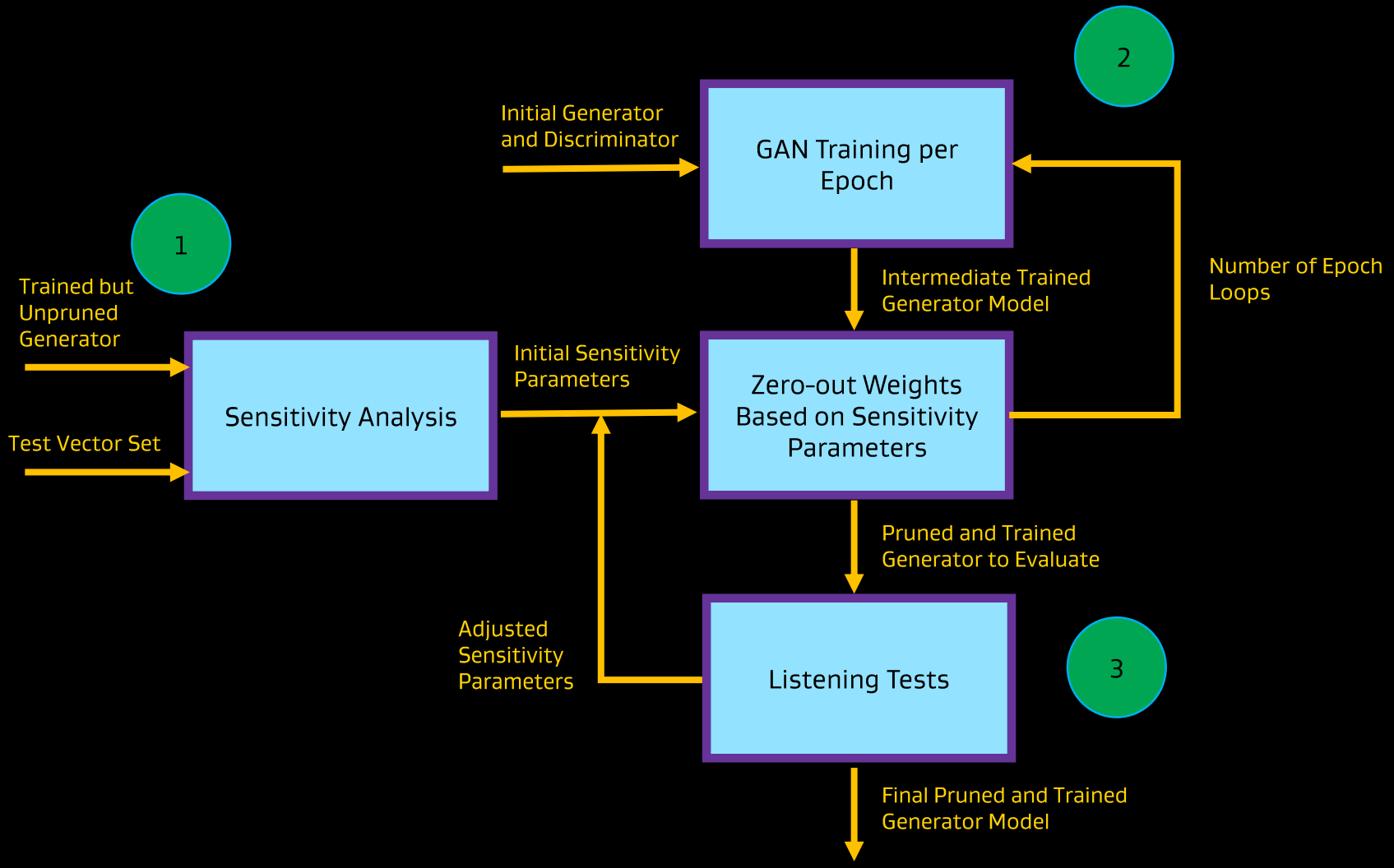
[1]C. Yu, J. Pool, and F. Xie, "Self-Supervised Generative Adversarial Networks Compression," *Nvidia GTC 2020.*

# Model Pruning – Reduce redundancy in the network

Post-training or during-training

- Set the value of unimportant parts of the model to zero
- Post-training: Fully train a model then analyze for potential pruning
    - Less effective, shorter training time
- During-training: Repeated training and pruning
    - More effective, longer training time
- Sensitivity Pruning: Technique Used

# Finding Parameters for Sensitivity Based Pruning



# Sensitivity Analysis

Where is the best place to remove weights?

- Automated pruning of weights across specified range, parameter-by-parameter
  - Report ideally indicates how compressible is each feature
- In our case turned out to be not useful
  - Lack of effective objective audio measure
  - Pruning parameters too aggressive
- Solution:
  - Intuitively setting the pruning configuration based on  deep learning theory and listening
    - Select sensitivity parameter in a way such that deeper layers gets pruned more than the outer layers, with the exception of the bottleneck layer.

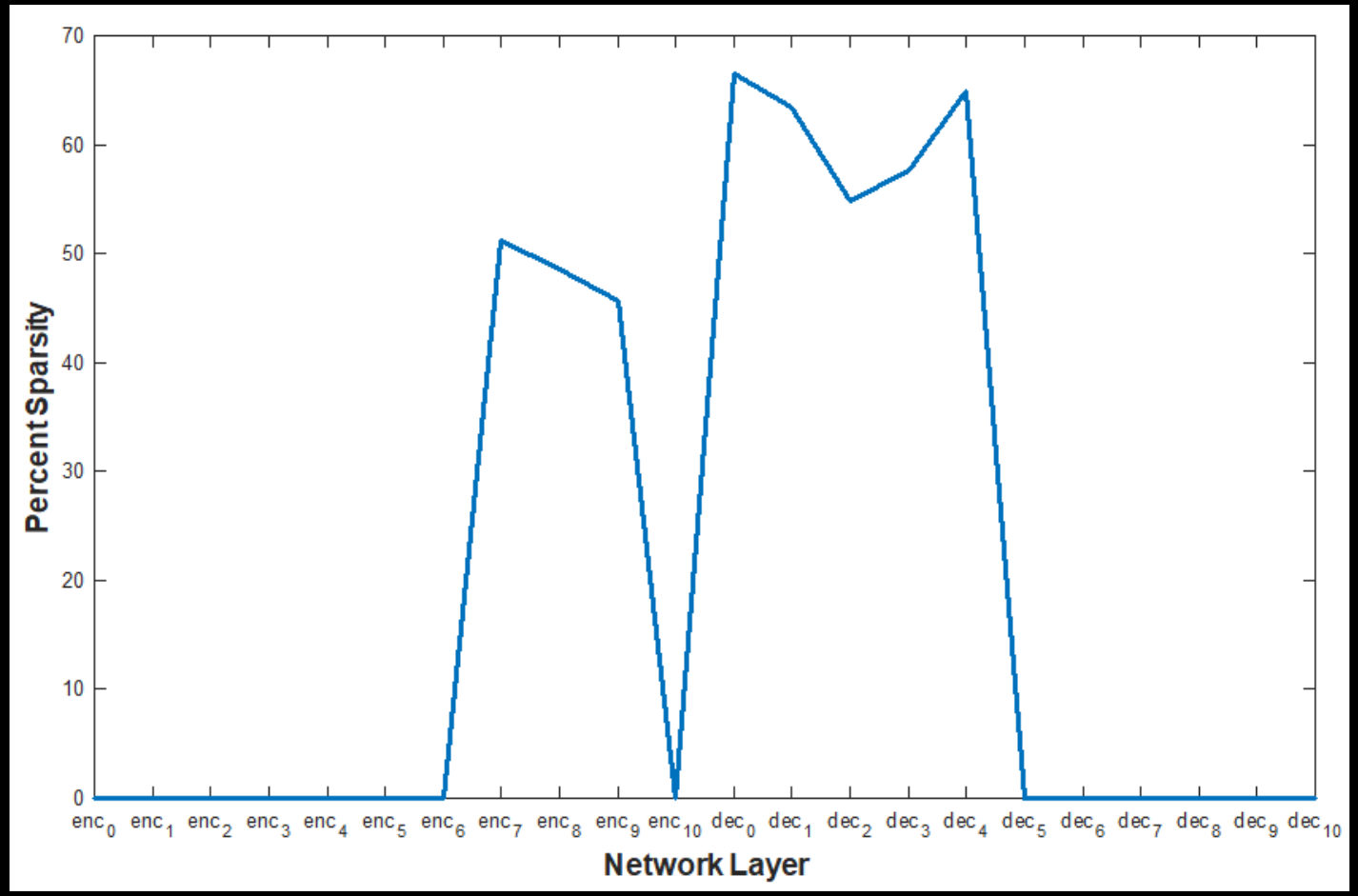
# Compression Schedule

Distiller Configuration File

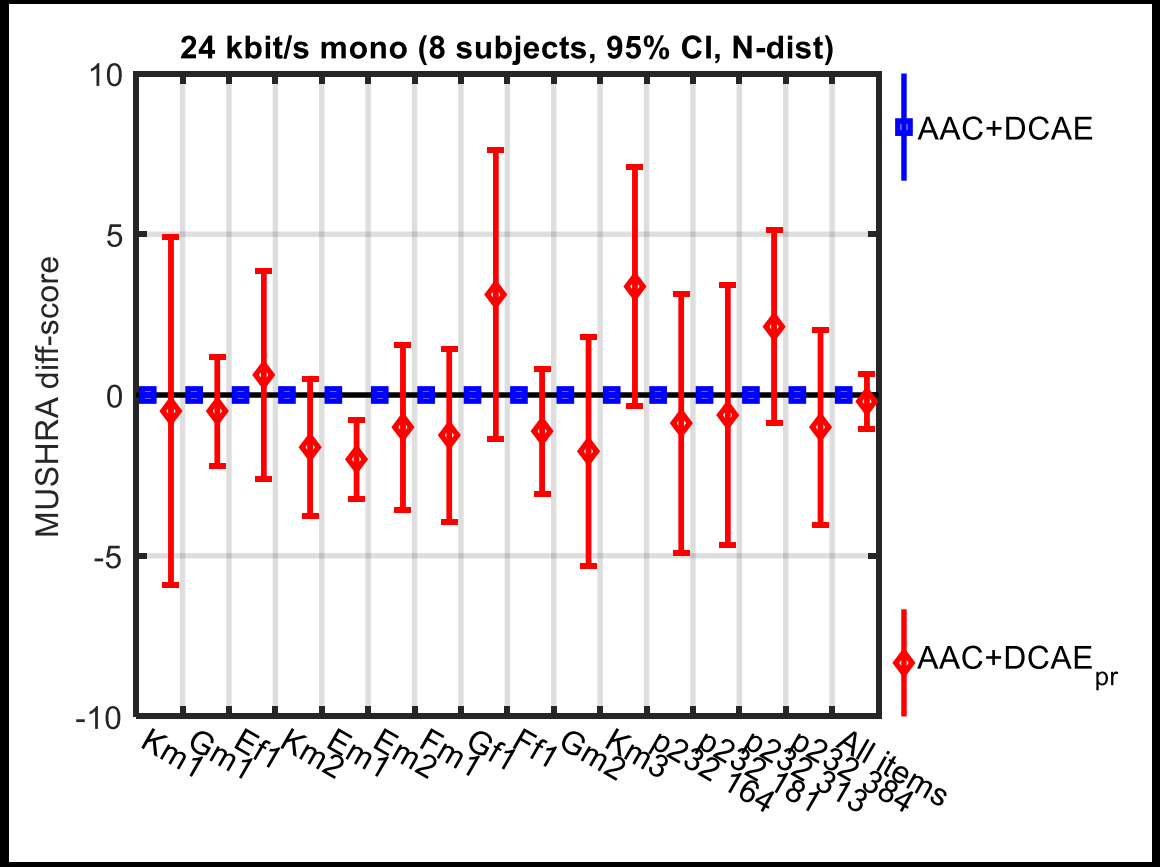
- Software hooks: Integrate compression into the training loop
- This configuration, the one used in our listening tests, returned a 47% reduction in non-zero network weights

```
pruners:
  pruner1:
class: 'SensitivityPruner'
    sensitivities:
      'G.enc_blocks.0.conv.weight': 0
      'G.enc_blocks.1.conv.weight': 0
      'G.enc_blocks.2.conv.weight': 0
      'G.enc_blocks.3.conv.weight': 0
      'G.enc_blocks.4.conv.weight': 0
      'G.enc_blocks.5.conv.weight': 0
      'G.enc_blocks.6.conv.weight': 0
      'G.enc_blocks.7.conv.weight': 0.3
      'G.enc_blocks.8.conv.weight': 0.3
      'G.enc_blocks.9.conv.weight': 0.3
      'G.enc_blocks.10.conv.weight': 0
      'G.dec_blocks.0.deconv.weight': 0.3
      'G.dec_blocks.1.deconv.weight': 0.3
      'G.dec_blocks.2.deconv.weight': 0.3
      'G.dec_blocks.3.deconv.weight': 0.3
      'G.dec_blocks.4.deconv.weight': 0.3
      'G.dec_blocks.5.deconv.weight': 0
      'G.dec_blocks.6.deconv.weight': 0
      'G.dec_blocks.7.deconv.weight': 0
      'G.dec_blocks.8.deconv.weight': 0
      'G.dec_blocks.9.deconv.weight': 0
      'G.dec_blocks.10.deconv.weight': 0
policies:
  - pruner:
      instance_name : 'pruner1'
    starting_epoch: 0
    ending_epoch: 129
    frequency: 2
```

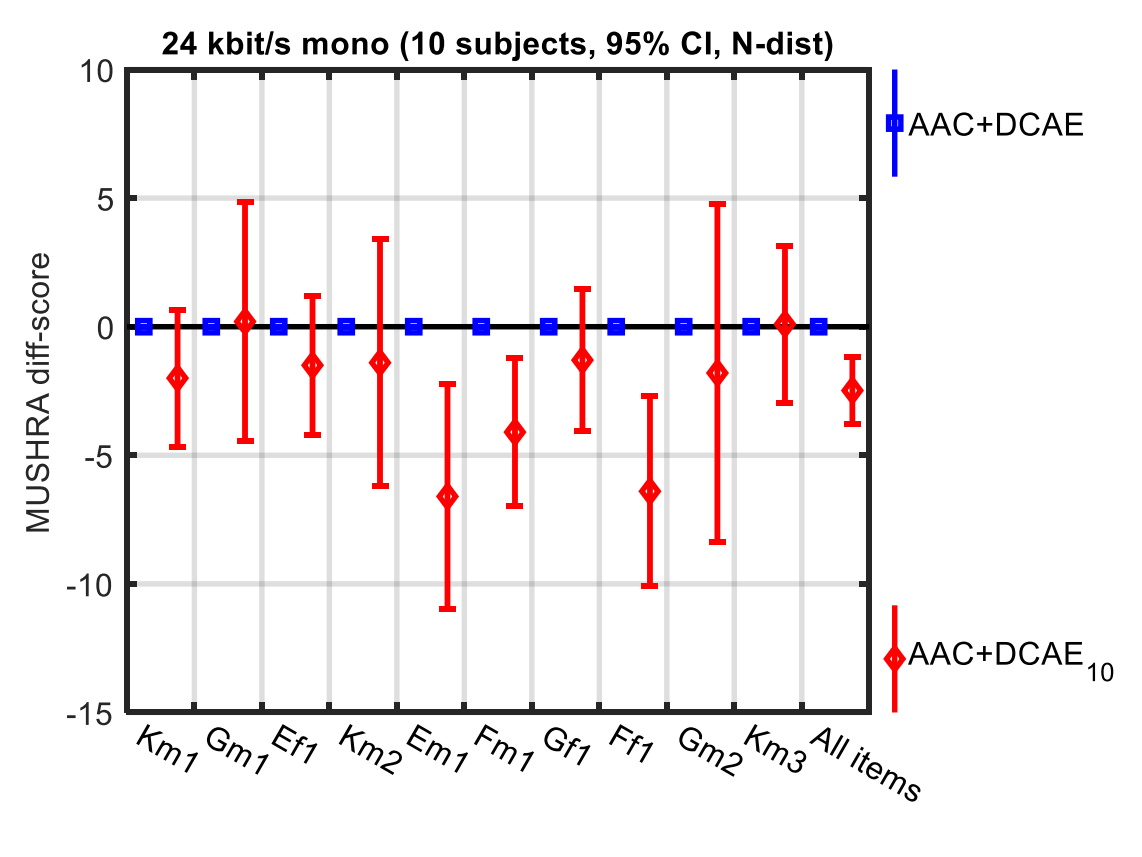
# Distribution of Sparsity Across Layers

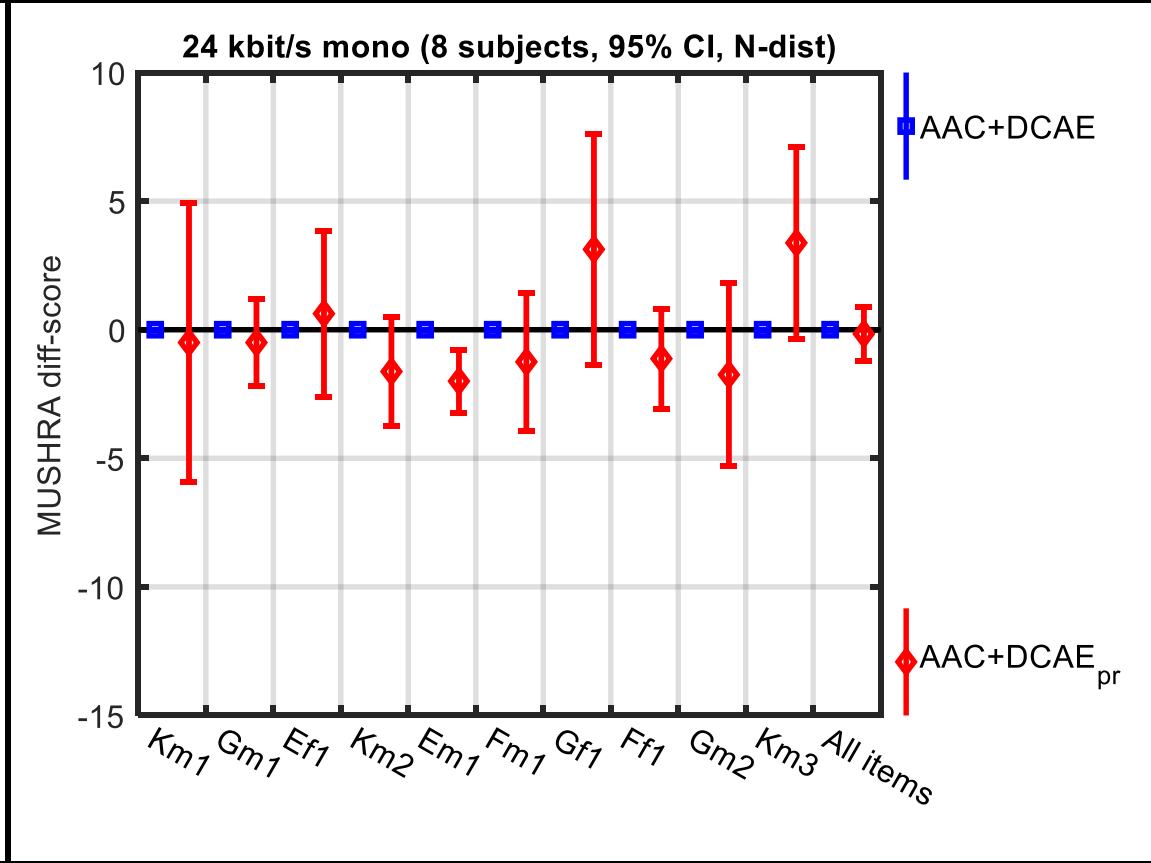


# Listening Test – AAC @ 24 kbit/s Mono
# Pruned vs Unpruned Models



# Comparison with an Unpruned Smaller Model



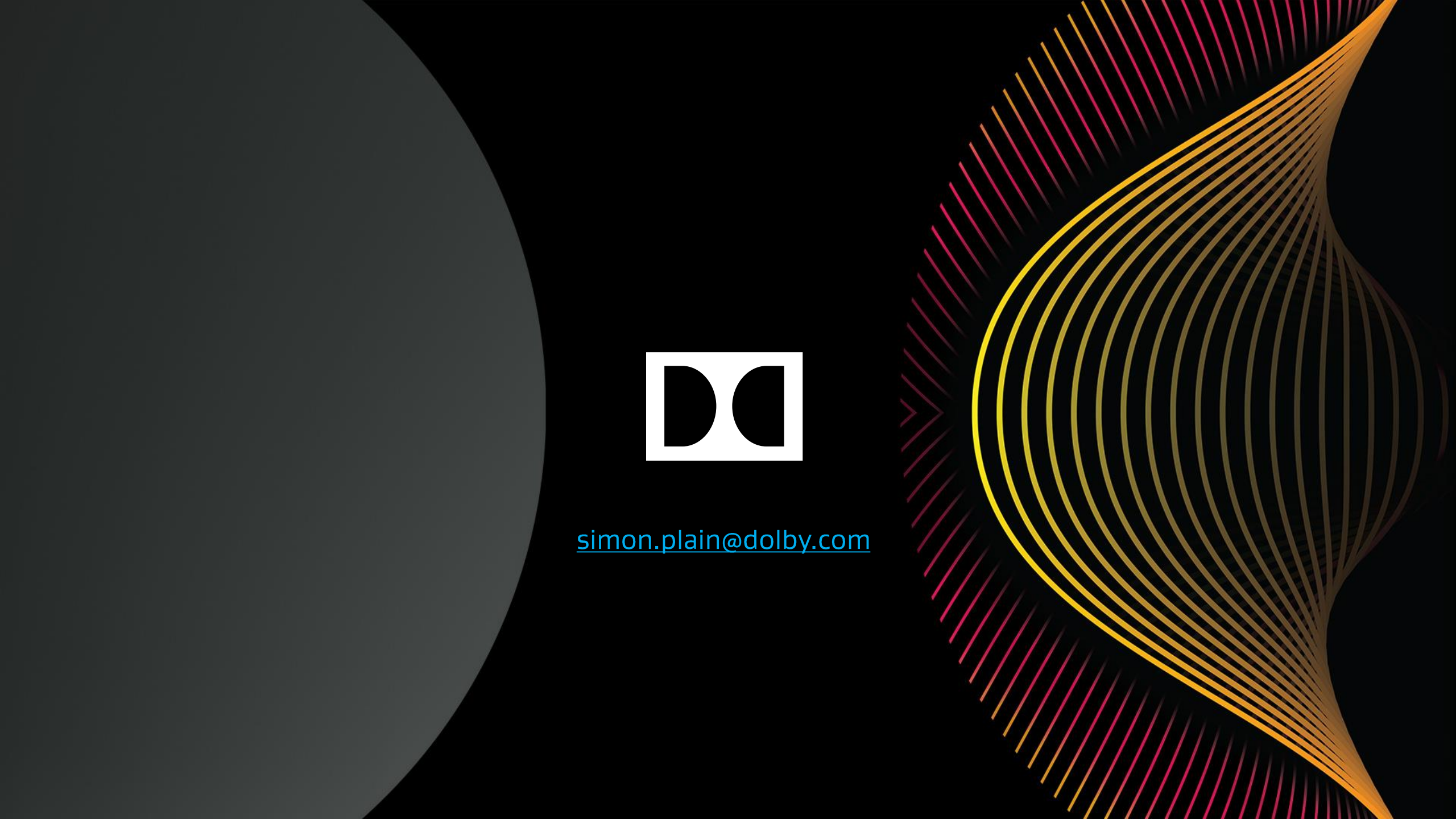




$DCAE_{10}$ has 11% fewer parameters than DCAE

$DCAE_{pr}$ has 47% fewer parameters than DCAE

# Conclusions

- We start with an audio enhancer for restoring signals with coding noise
- Our best procedures to date for constructing a pruning policy for a GAN
- Achieved nearly 50% reduction in non-zero weights with nearly negligible quality loss



simon.plain@dolby.com