

Neural Sinusoidal Modeling

From the Master Thesis of *Michail Raptakis*

Thesis Advisor: Professor *Yannis Stylianou*



University of Crete, Computer Science Department,
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Vocoders

➤ General Definition: A **vocoder** (short for “voice encoder”) is an audio processor (e.g., device, program, etc.) that encodes, analyses and **synthesizes human voice** signals.

➤ Application Examples:

- Radio (e.g., Digital Mobile Radio).
- Telephone networks (e.g., Voice over Internet Protocol).
- Musical and other artistic effects (e.g., feeding synthesizer outputs to a vocoder filter bank).
- Audio codecs (e.g., FLAC, MPEG-4).
- Encryption systems (e.g., NSA).
- Medical applications (e.g., cochlear implants).
- **Text-To-Speech synthesis** (e.g., voice assistants).

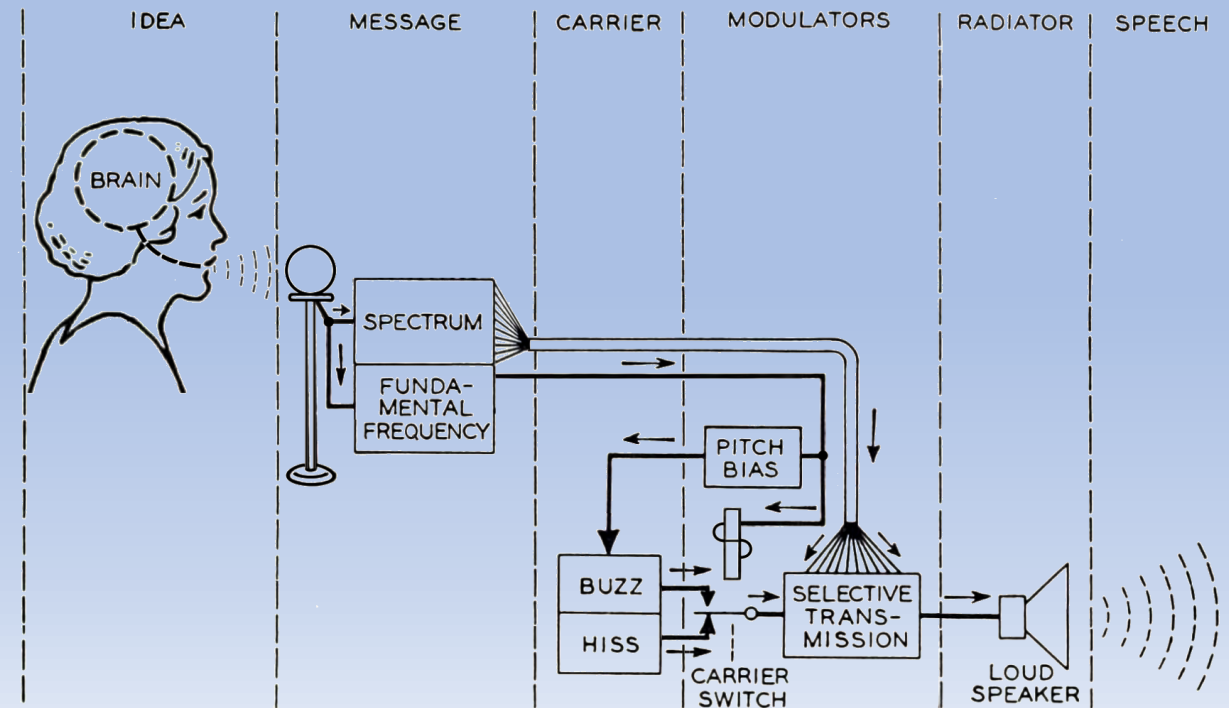
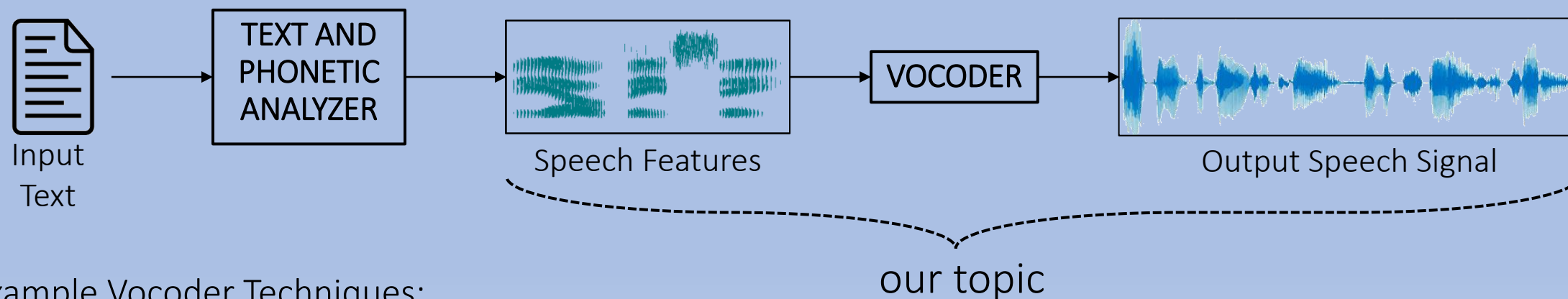


Fig. 7—Schematic circuit of the vocoder.

Text-To-Speech Synthesis

- Definition: A **Text-To-Speech (TTS) synthesis** system converts normal language text into audible speech output.
- We can break this problem into **two separate tasks**¹:
 - 1) Speech feature extraction (most commonly spectrograms) from the text.
 - 2) Synthesizing the artificial voice given these speech features (i.e., **vocoder's task** – our topic).



➤ Example Vocoder Techniques:

- Linear Predictive Coding (LPC).
- Griffin-Lin algorithm.
- Waveform-interpolative.
- Concatenative synthesis.
- **Sinusoidal representations.**
- **Neural-Based.**
- Combinations of the above etc.

¹ Models that do it in one pass, are called End-To-End (E2E).

The First Sinusoidal Vocoder

- Estimating speech using sums of linear AM sinusoids within short frames:

$$s[n] \approx \hat{s}[n] = \sum_{l=1}^{L(k)} \left[\hat{A}_l[n] \cos \left(\hat{\theta}_l[n] \right) \right].$$

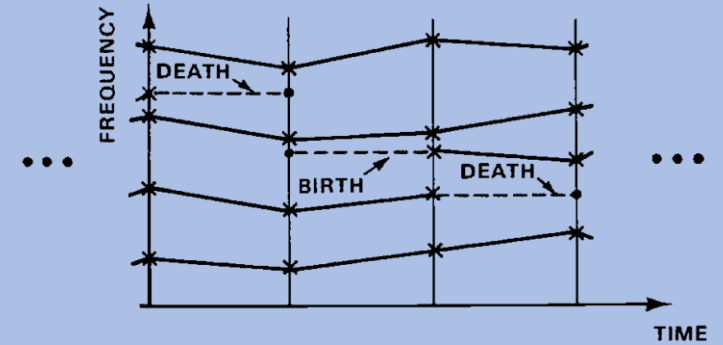


Fig. 4. Illustration of frequency tracks using the birth-death frequency tracker.

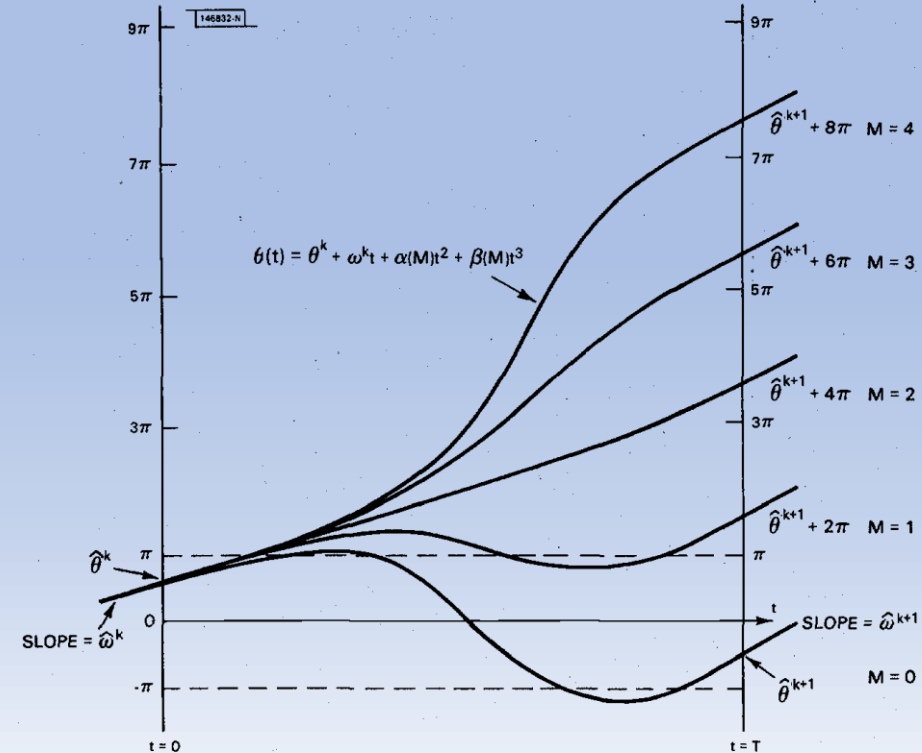
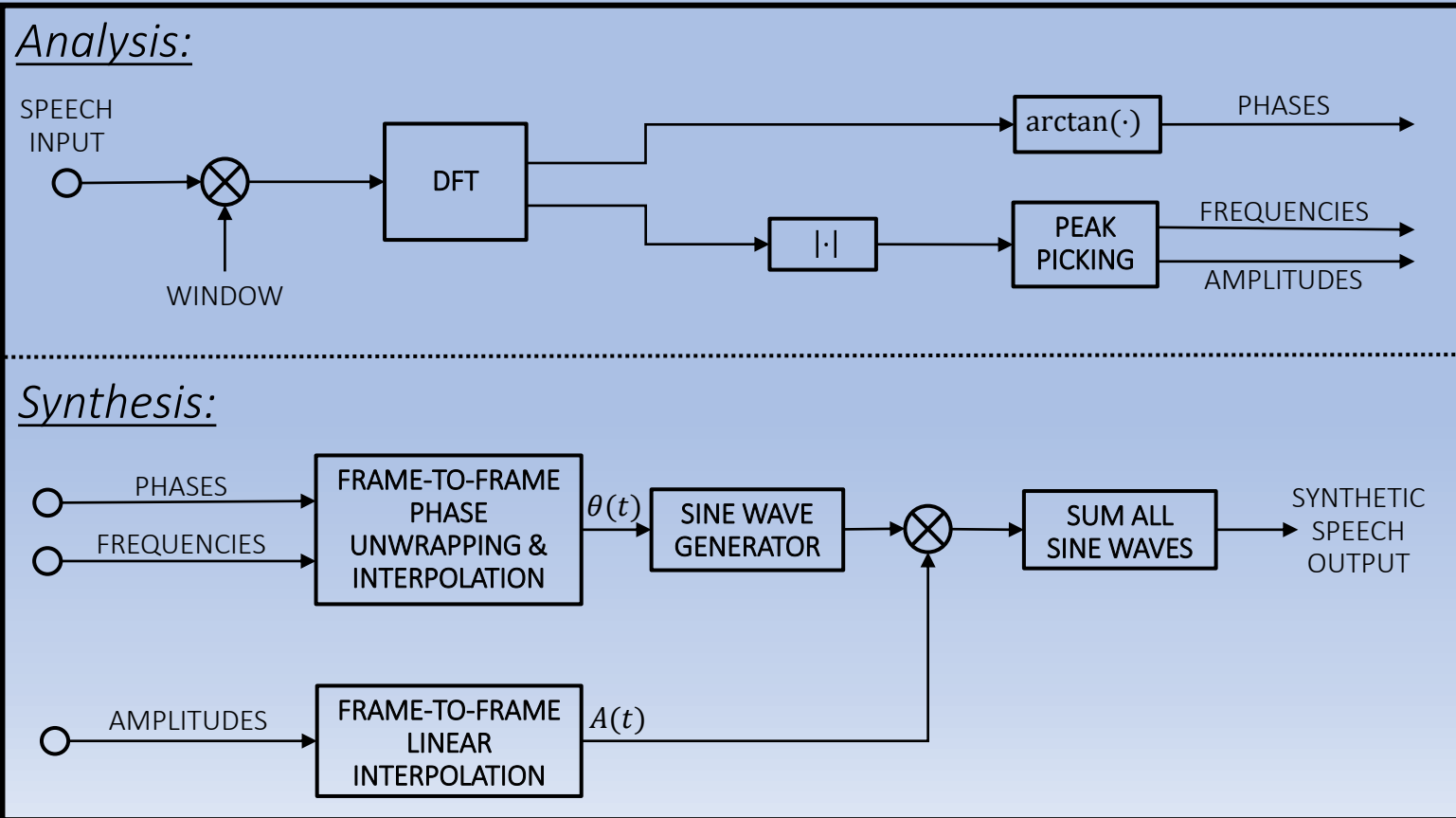


Fig. 6. Typical set of cubic phase interpolation functions.



Redefining the Problem: Phase and Amplitude

- Any generic **AM-FM** sinusoidal discrete-time wave can be written as:

$$\begin{aligned} & A[n] \sin \left(2\pi f_c \frac{n}{f_s} + \phi[n] \right) \\ &= A[n] \left(\sin \left(2\pi f_c \frac{n}{f_s} \right) \cos(\phi[n]) + \cos \left(2\pi f_c \frac{n}{f_s} \right) \sin(\phi[n]) \right) \\ &= \underline{A[n] \cos(\phi[n])} \sin \left(2\pi f_c \frac{n}{f_s} \right) + \underline{A[n] \sin(\phi[n])} \cos \left(2\pi f_c \frac{n}{f_s} \right) \\ &= \underline{\alpha[n]} \sin \left(2\pi f_c \frac{n}{f_s} \right) + \underline{\beta[n]} \cos \left(2\pi f_c \frac{n}{f_s} \right), \text{ where} \\ &\alpha[n] = A[n] \cos(\phi[n]), \text{ and } \beta[n] = A[n] \sin(\phi[n]). \end{aligned}$$

$\sin(a + b) = \sin(a) \cos(b) + \cos(a) \sin(b).$

- Therefore, a speech wave $s[n]$ can be approximated with AM-FM sinusoids as:

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi \hat{f}_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi \hat{f}_m \frac{n}{f_s} \right) \right].$$

f_s = Sampling rate

$s[n]$ = Input speech wave

$\hat{s}[n]$ = Predicted output speech wave

M = Total number of sinusoid pairs

f_m = Frequency of the m -th sinusoid pair

$\alpha_m[n]$ = Amplitudes of the m -th cosine wave

$\beta_m[n]$ = Amplitudes of the m -th sine wave

Redefining the Problem: Frequencies

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi \hat{f}_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi \hat{f}_m \frac{n}{f_s} \right) \right].$$

- In fact, we can represent **any arbitrary signal** using sums of **AM sinusoids**:

Let $h[n]$ be an arbitrary discrete-time signal. Choose frequencies f_1, f_2 s.t. $\text{LCM}(f_1, f_2) < f_s/2$. We can write $h[n]$ as:

$$\begin{aligned} h[n] &= h[n] \mathbb{1} \left\{ f_2 n = k \frac{f_s}{2} \right\} + h[n] \mathbb{1} \left\{ f_2 n \neq k \frac{f_s}{2} \right\}, \quad k \in \mathbb{N}. \\ &= \frac{h[n] \mathbb{1} \left\{ f_2 n = k \frac{f_s}{2} \right\}}{\cos \left(2\pi f_1 \frac{n}{f_s} \right)} \cos \left(2\pi f_1 \frac{n}{f_s} \right) + \frac{h[n] \mathbb{1} \left\{ f_2 n \neq k \frac{f_s}{2} \right\}}{\sin \left(2\pi f_2 \frac{n}{f_s} \right)} \sin \left(2\pi f_2 \frac{n}{f_s} \right), \quad k \in \mathbb{N} \\ &= \underline{\alpha_2[n]} \sin \left(2\pi f_2 \frac{n}{f_s} \right) + \underline{\beta_1[n]} \cos \left(2\pi f_1 \frac{n}{f_s} \right), \quad \text{where} \\ \alpha_2[n] &= \frac{h[n] \mathbb{1} \left\{ f_2 n \neq k \frac{f_s}{2} \right\}}{\sin \left(2\pi f_2 \frac{n}{f_s} \right)}, \quad \text{and} \quad \beta_1[n] = \frac{h[n] \mathbb{1} \left\{ f_2 n = k \frac{f_s}{2} \right\}}{\cos \left(2\pi f_1 \frac{n}{f_s} \right)}. \end{aligned}$$

$\mathbb{1}\{\cdot\}$ = Indicator function

- **Not** a actual scenario we want **in practice**, i.e., directly synthesize a speech wave using two AM signals.
- Ideally, **many sinusoids** should cooperate to make this an easier synthesis task (breaking it down).
 - This just shows the **limitless** representation **capabilities** from a theoretical point of view.
- In any case, this proves that we can work with **constant frequencies** over **longer frames**.

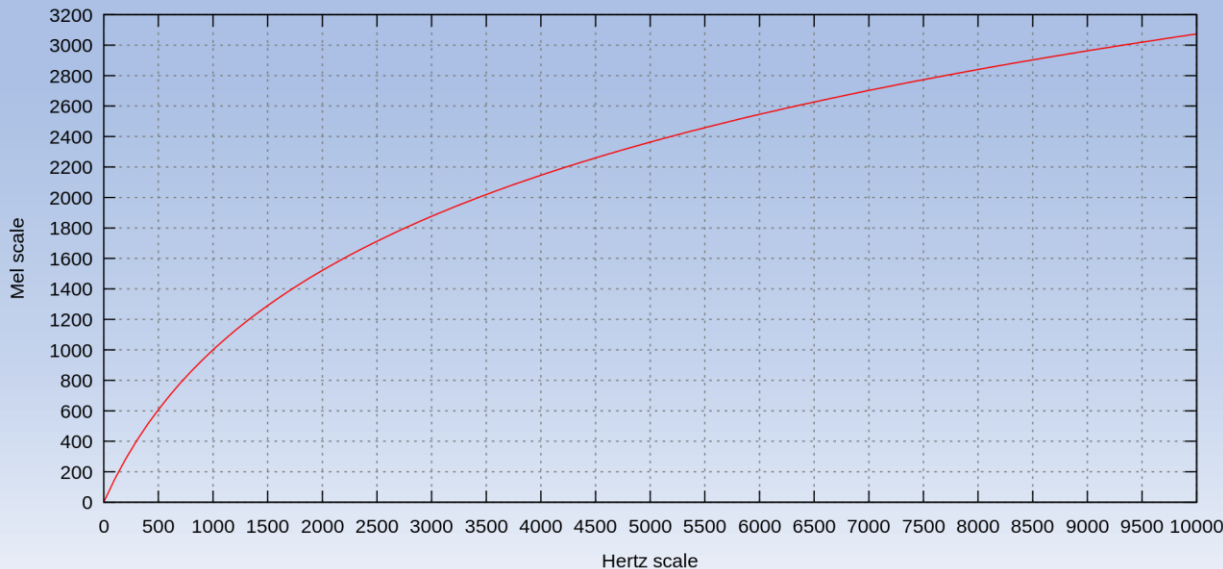
Redefining the Problem: Input

- Speech is non-stationary, therefore cannot be analyzed with a single DFT for long enough frames.
 - 💡 Discrete Short-Time Fourier Transform (DSTFT) → Spectrogram
- A linear spectrogram gives “equal importance” to all frequencies and requires **a lot of space**.
 - 💡 **Mel-Spectrogram** scales the frequencies **logarithmically** and unites them into **frequency bands**.

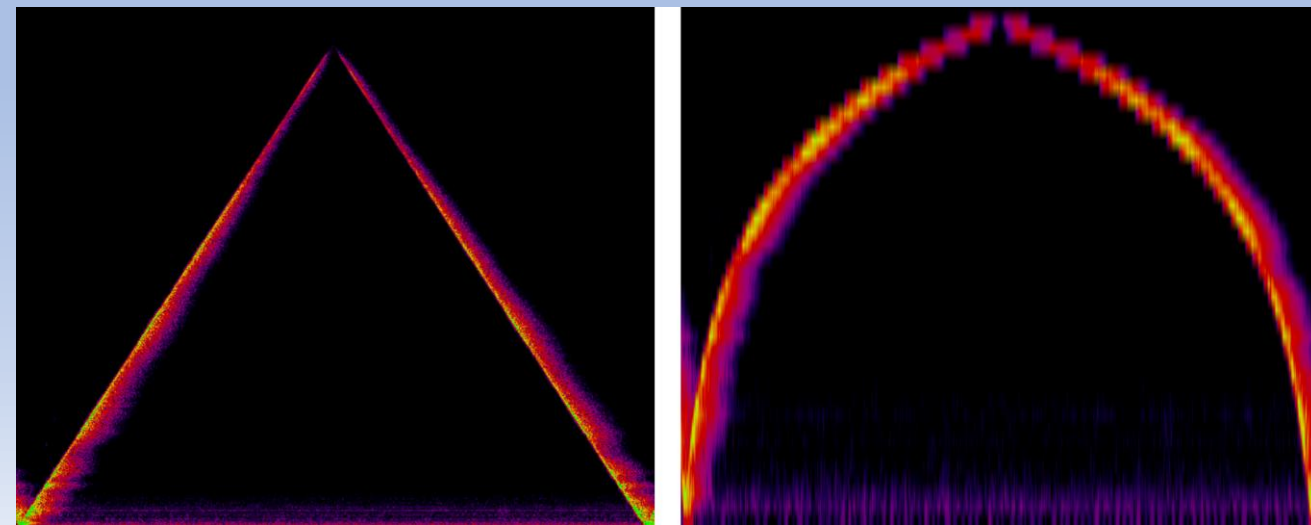
$$m(f) = 2595 \log_{10} \left(1 + \frac{f}{700} \right) = 1127 \ln \left(1 + \frac{f}{700} \right), \quad f \in \mathbb{R}^+. \quad (\text{Common conversion function from hertz to mel scale.})$$

$$f(m) = 700 \left(10^{\frac{m}{2595}} - 1 \right) = 700 \left(e^{\frac{m}{1127}} - 1 \right), \quad m \in \mathbb{R}^+. \quad (\text{Common conversion function from mel to hertz scale.})$$

Plot of frequencies in Mel versus Hertz scale:



An increasing and decreasing tone from 20Hz to 22kHz and back:



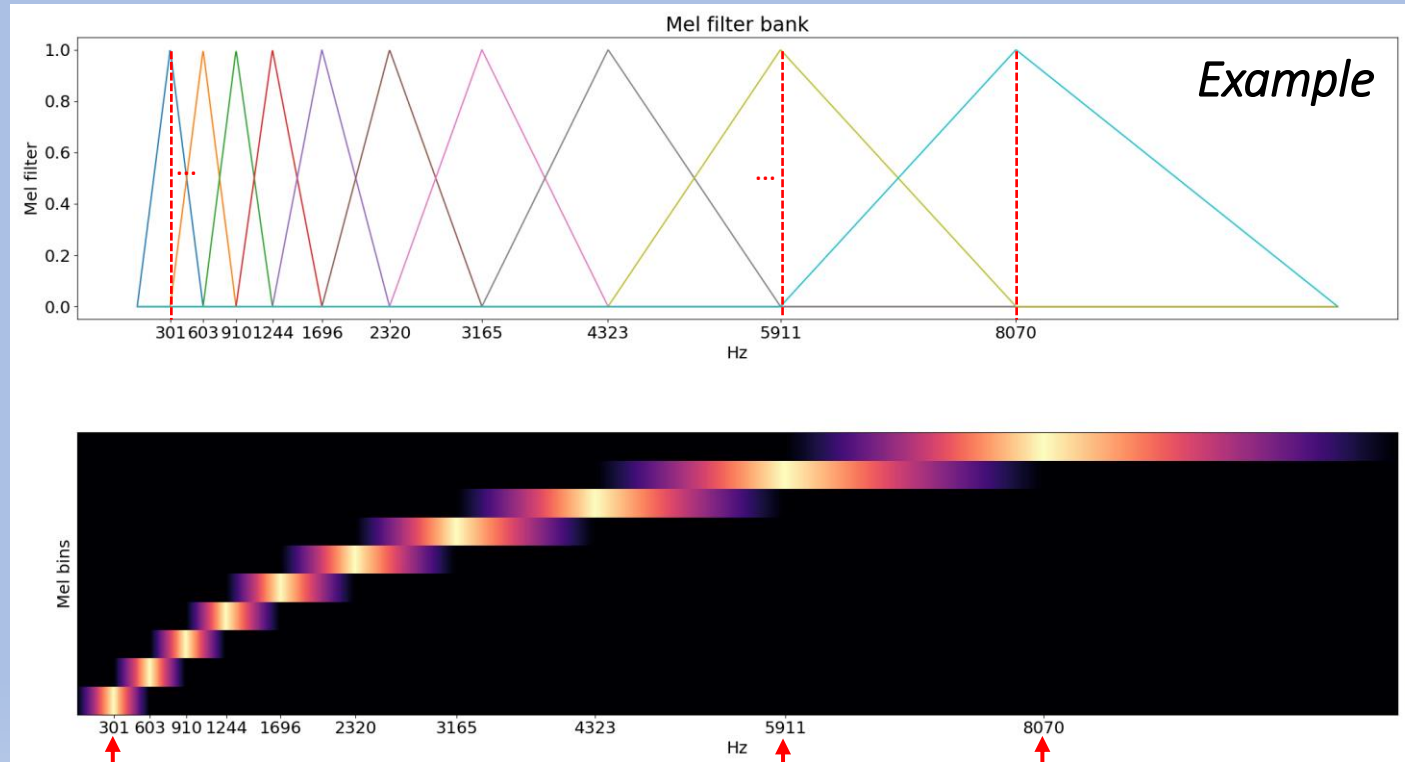
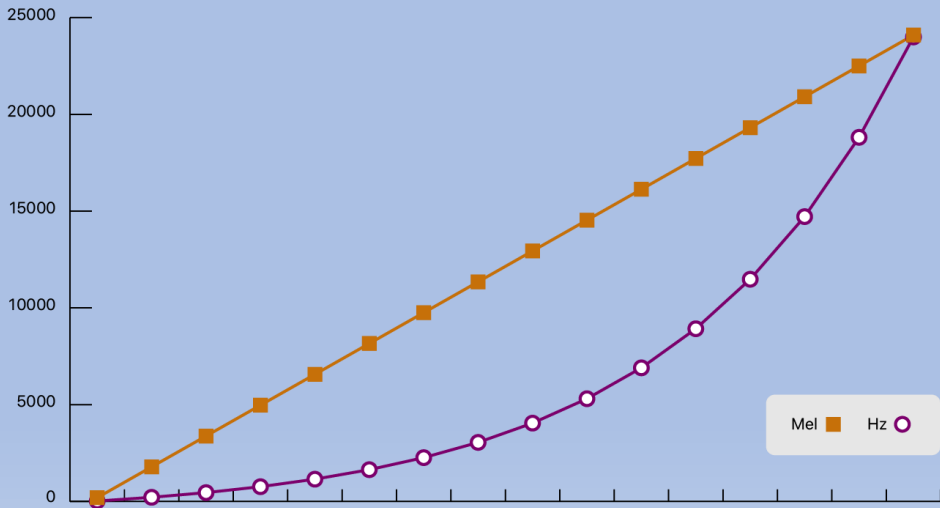
Spectrogram

Mel-Spectrogram

Redefining the Problem: Input and Frequencies

We create M equally spaced frequencies $\mathbf{m}_f = (m(f_{\min}), \dots, m(f_{\max}))$ and then convert them back into Hz scale $f(\mathbf{m}_f)$.

Then, to obtain the Mel-Spectrogram, we filter the linear spectrogram with M overlapping triangular filters whose centres are the $f(\mathbf{m}_f)$ frequencies:



These central band frequencies are the choice for our constant frequencies f_m of the model.

So, the **number** of AM sinusoid **pairs** in our model will be **equal** to the number of **mel bands** M that we choose for our input spectrogram.

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi f_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi f_m \frac{n}{f_s} \right) \right].$$

Redefining the Problem: Summary

➤ Original Sinusoidal Representation:

$$s[n] \approx \hat{s}[n] = \sum_{l=1}^{L(k)} \left[\hat{A}_l[n] \cos \left(\hat{\theta}_l[n] \right) \right].$$

➤ Proposed Sinusoidal Representation:

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi f_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi f_m \frac{n}{f_s} \right) \right].$$

➤ Main assumption differences:

- 1) Not constructing small frames with sums of AM sinusoids:
→ **longer frames** with sums of AM-FM sinusoids.
- 2) No alternating frequency estimation methods:
→ **constant frequencies** instead.
- 3) No separate phase or amplitude interpolation:
→ **$\hat{\alpha}[n], \hat{\beta}[n]$ amplitude modulators** compensate for them.
- 4) No DFT or linear spectrogram as input:
→ **Mel-Spectrogram** instead.
- 5) No analytical parameter estimation from the input:
→ **optimization** approach instead.

Redefining the Problem: Solution

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi f_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi f_m \frac{n}{f_s} \right) \right].$$

➤ Goal = **approximate** $s[n]$ with $\hat{s}[n]$ as closely as possible.

💡 Treat it as an **optimization problem**: **Minimize** the **distance** between $\hat{s}[n]$ and $s[n]$.

➤ Distance = **loss function** \mathcal{L}

- Numeric estimation of how close its inputs are.
- E.g., Mean Squared Error (MSE):

$$\mathcal{L}(s[n], \hat{s}[n]) = \frac{1}{N} \sum_{n=1}^N \left[s[n] - \hat{s}[n] \right]^2.$$

➤ Parametric model F with parameters θ

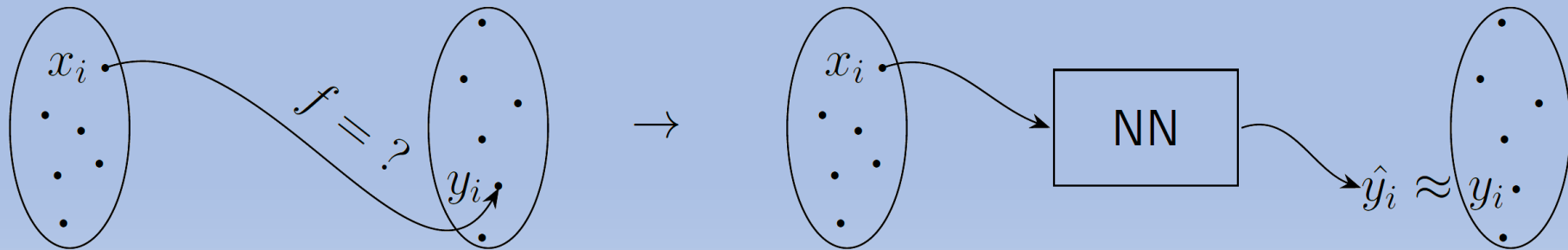
- Input = Mel-Spectrogram S
- Output = $\hat{\alpha}, \hat{\beta}$ that minimize \mathcal{L}

$$\begin{aligned} & \min_{\theta} \mathcal{L}(s, \hat{s}) \\ & \hat{\alpha}, \hat{\beta} = F(S, \theta). \end{aligned}$$

Optimization Approach: Artificial Neural Networks

$$\min_{\theta} \mathcal{L}(s, \hat{s})$$
$$\hat{\alpha}, \hat{\beta} = F(S, \theta).$$

- There exist many optimization methods depending on the problem's complexity/hardness, for instance, Linear, Convex, Message Passing, Belief Propagation, etc.
- From literature, we know that the hardness of our speech synthesis problem has shown the need for **Artificial Neural Networks (ANNs)**, or **Neural Networks (NNs)** for short.



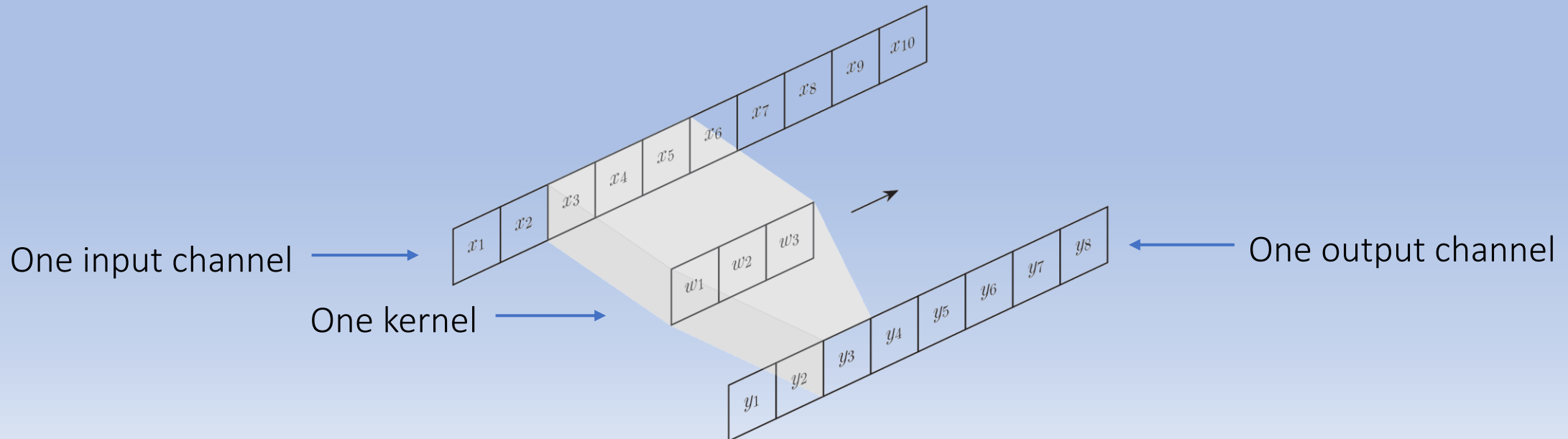
- Our function F will be an **artificial neural network** (or neural network, for short)
 - Neural networks are comprised of **layers**.
 - Each layer is made of **weights**, or neurons (i.e., the trainable parameters θ).
 - Each type of layer applies a different **operation** on its input to give its output.
- The main layer of interest for this lecture will be the 1D Convolutional Layer.

Neural Networks: 1D Convolutional Layer, 1D Input Case

- Equation for discrete 1D convolution (actually cross-correlation):
 - Input sequence x , Output sequence y , One kernel w :

$$y_j = \sum_{k=1}^K \left[w_k \cdot x_{j'} \right] + b, \quad j' = j - 1 + k.$$

- One can think of convolving as sliding a window over the input:

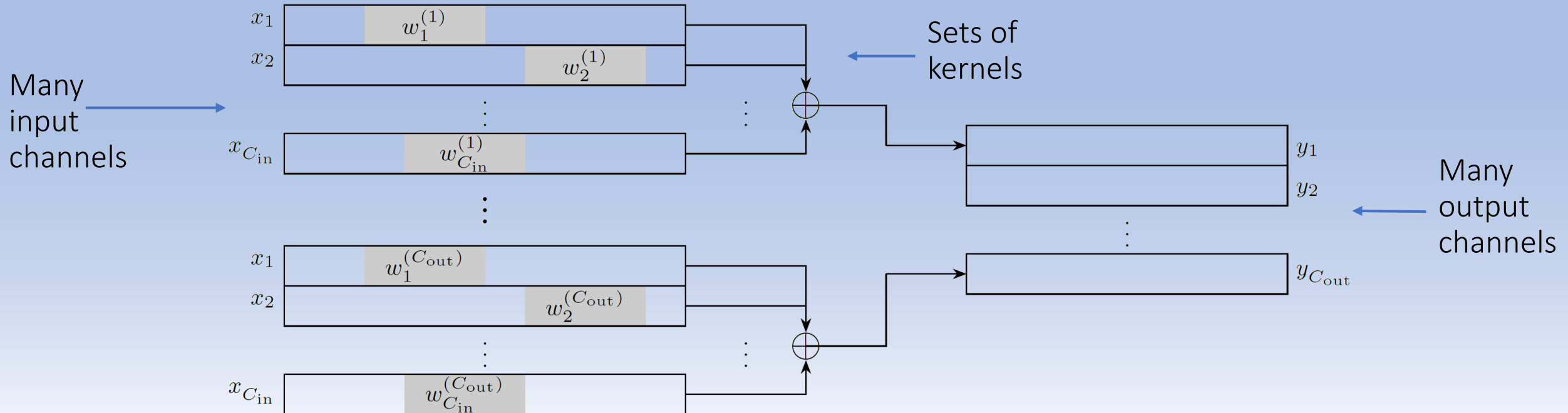


- Here, the **kernel** values w_k and the **bias** term b are **trainable parameters** ($\in \theta$).

Neural Networks: 1D Convolutional Layer, 2D Input Case

- Processing a 2D array, e.g., a Mel-Spectrogram with a 1D convolutional layer is possible:
 - We can treat **each input row**, e.g., frequency band, as an **input channel**.
- Getting a 2D output from a 1D convolutional layer is possible:
 - We can have **multiple kernels** operating over one input, thus resulting in a **different output sequence**, i.e., output channel, **per set of kernels**¹:

$$y_{i,j} = \sum_{i'=1}^{C_{in}} \sum_{k=1}^K \left[w_{i',k}^{(i)} \cdot x_{i',j'} \right] + b_i, \quad j' = S(j-1) + k.$$

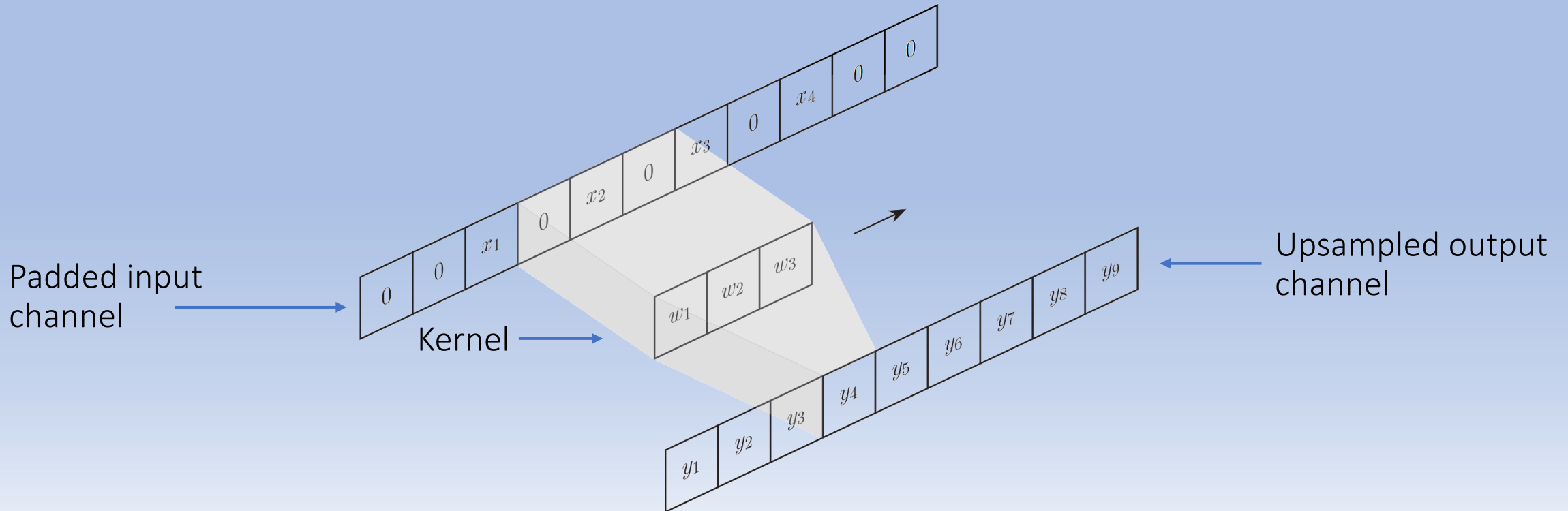


¹ The entire set of all kernels is known as the **filter** of the convolution.

Neural Networks: Transposed 1D Convolutional Layer

- **Upsampling** along the **x-axis** (time axis in our case) is also necessary.
 - It normally **convolves** the input sequence,
 - after introducing **padding** among the **input's** consecutive values:

$$x'_{i,j} = \begin{cases} x_{i,j'}, & j = S(j' - 1) + 1 \\ 0, & \text{otherwise.} \end{cases}$$



- There are more hyperparameters normally in a convolutional layer (e.g., dilation, initialization, etc.).

Neural Networks: Optimizer

- The algorithm for **updating the trainable parameters** (e.g., convolution kernels).
 - Iteratively registers updates to minimize our loss function (by trying to find the critical points).

Algorithm 1 Stochastic Gradient Descent

Input: Number of Iterations: $T \in \mathbb{N}$, Parametric Model: f , with Trainable Weights: w and Biases: b , Input Training Dataset: x , Corresponding Ground Truth: y , Loss Function: \mathcal{L} , Learning Rate: $\eta \in \mathbb{R}^+$, Weight Initialization: w_0 , Bias Initialization: b_0

Output: Trained parametric model f .

```
1:  $w \leftarrow w_0$            # Initialize the weights  $w$ .
2:  $b \leftarrow b_0$        # Initialize the biases  $b$ .
3: for  $i \in \llbracket 1, T \rrbracket$  do   # For all number of iterations given:
4:    $x_i \leftarrow \text{Sample}(x)$  # Fetch a sample  $x_i$  from  $x$ .
5:    $\hat{y}_i \leftarrow f(x_i)$   # Get the model's prediction  $\hat{y}_i$  for the input  $x_i$ .
6:    $L_i \leftarrow \mathcal{L}(\hat{y}_i, y_i)$  # Compute the loss  $L_i$  between  $\hat{y}_i$ , and  $y_i$ .
7:    $\Delta w \leftarrow -\nabla_w L_i$  # Compute the negative gradient  $-\nabla$  of  $L_i$  w.r.t.  $w$ .
8:    $\Delta b \leftarrow -\nabla_b L_i$  # Compute the negative gradient  $-\nabla$  of  $L_i$  w.r.t.  $b$ .
9:    $w \leftarrow w + \eta \Delta w$  # Scale  $\Delta w$  by  $\eta$ , and update the weights.
10:   $b \leftarrow b + \eta \Delta b$  # Scale  $\Delta b$  by  $\eta$ , and update the biases.
11: end for
12: return  $f$ 
```

- Improved versions of the above optimizer are actually used, but the core idea remains.
 - Examples: Mini-Batch SGD, RMSprop, AdaGrad, Adam, etc.

Neural Networks: Backpropagation

- **Backpropagation** is the **algorithm** for computing the **gradient**:
 - The outputs of the neural network are given during the **forward pass**.
 - Creating the neural network's **computational graph** is needed.
 - Then, the **backward pass** calculates the gradient of the computational graph.
 - This is done by applying the **chain rule** over and over:

$$\frac{\partial}{\partial x} (f \circ g) = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}.$$

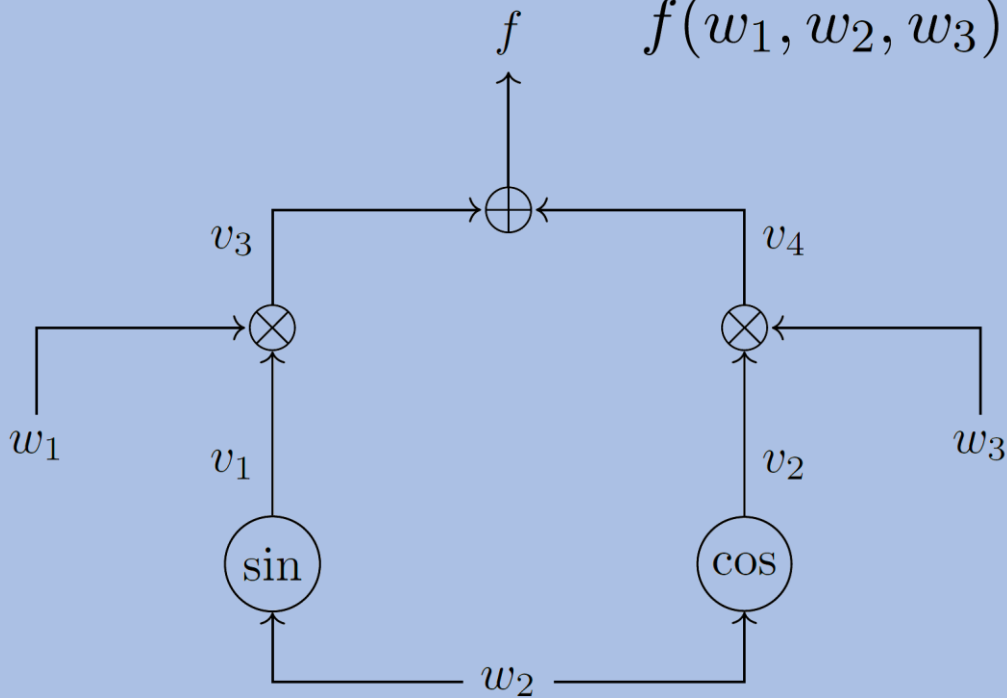
- It **always works** for any neural network, because in its core:
 - It is a combination of **matrix multiplications**,
 - and **differentiable**¹ **function compositions**:

$$f(x) = g^L (W^L g^{L-1} (W^{L-1} \dots g^2 (W^2 g^1 (W^1 x) \dots))) .$$

¹ Non-differentiable models, e.g., Boltzmann Machines are beyond our scope.

Backpropagation: Toy Example

$$f(w_1, w_2, w_3) = w_1 \sin(w_2) + w_3 \cos(w_2).$$



Forward Pass

$$v_1 = \sin(w_2)$$

$$v_2 = \cos(w_2)$$

$$v_3 = v_1 w_1$$

$$v_4 = v_2 w_3$$

$$f = v_3 + v_4$$

$$\frac{\partial}{\partial x} (f \circ g) = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}.$$

Backward Pass

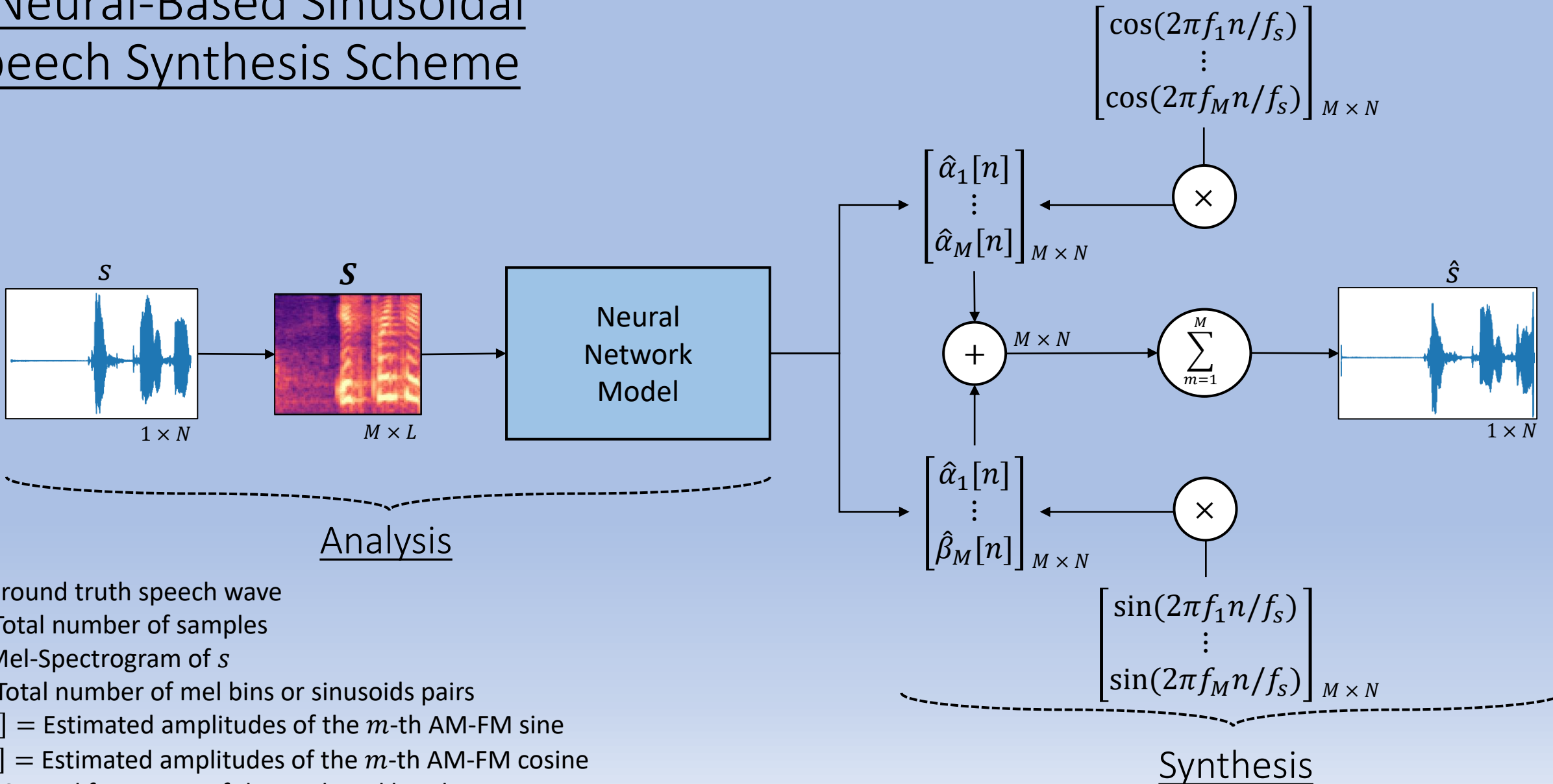
$$\bar{v}_3 = \frac{\partial f}{\partial v_3} = 1, \quad \bar{v}_1 = \frac{\partial f}{\partial v_3} \frac{\partial v_3}{\partial v_1} = \bar{v}_3 w_1 = w_1, \quad \bar{w}_1 = \frac{\partial f}{\partial v_3} \frac{\partial v_3}{\partial w_1} = \bar{v}_3 v_1 = v_1 = \sin(w_2).$$

$$\bar{v}_4 = \frac{\partial f}{\partial v_4} = 1, \quad \bar{v}_2 = \frac{\partial f}{\partial v_4} \frac{\partial v_4}{\partial v_2} = \bar{v}_4 w_3 = w_3, \quad \bar{w}_3 = \frac{\partial f}{\partial v_4} \frac{\partial v_4}{\partial w_3} = \bar{v}_4 v_2 = v_2 = \cos(w_2).$$

$$\bar{w}_2 = \frac{\partial f}{\partial v_1} \frac{\partial v_1}{\partial w_2} + \frac{\partial f}{\partial v_2} \frac{\partial v_2}{\partial w_2} = \bar{v}_1 \cos(w_2) - \bar{v}_2 \sin(w_2) = w_1 \cos(w_2) - w_3 \sin(w_2).$$

- Derive f analytically for sanity check: $\nabla_w f = \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial w_3} \right) = \left(\sin(w_2), w_1 \cos(w_2) - w_3 \sin(w_2), \cos(w_2) \right)$. ✓

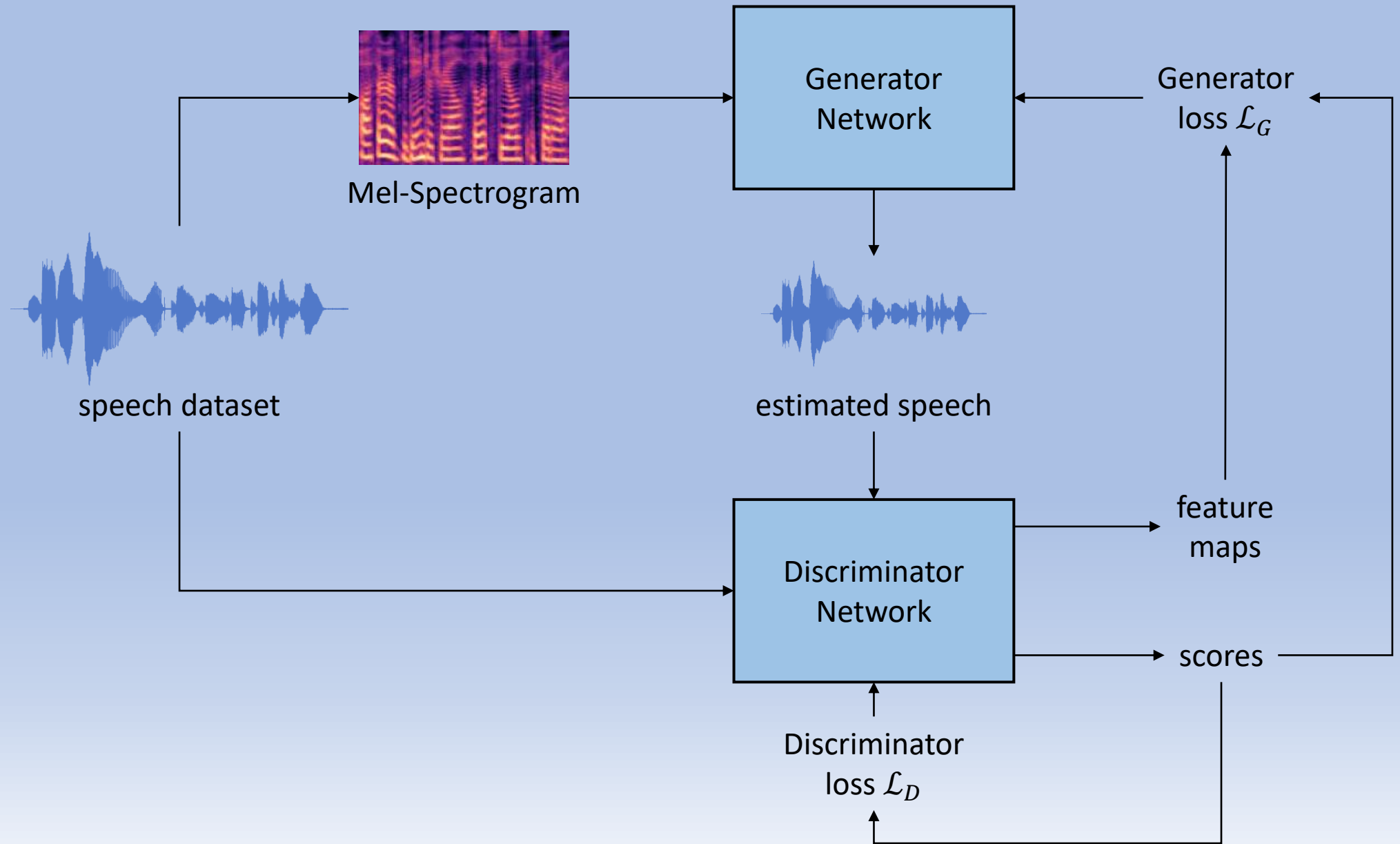
A Neural-Based Sinusoidal Speech Synthesis Scheme



s = Ground truth speech wave
 N = Total number of samples
 S = Mel-Spectrogram of s
 M = Total number of mel bins or sinusoids pairs
 $\hat{\alpha}_m[n]$ = Estimated amplitudes of the m -th AM-FM sine
 $\hat{\beta}_m[n]$ = Estimated amplitudes of the m -th AM-FM cosine
 f_m = Central frequency of the m -th mel band
 f_s = sampling rate
 \hat{s} = Estimated output speech wave

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi f_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi f_m \frac{n}{f_s} \right) \right].$$

Adversarial Training Scheme



Adversarial Loss Functions: Discriminator's Perspective

- Discriminator's output score:
 - Positive score $D_k(y) > 0 \Rightarrow y$ is real.
 - Negative score $D_k(y) < 0 \Rightarrow y$ is fake.
 - Higher score in magnitude \Rightarrow stronger belief about y (and vice versa).
- Discriminator's objective:
 - Maximize $D_k(y)$ (or minimize $-D_k(y)$) for all k when y is indeed real.
 - Minimize $D_k(y)$ for all k when y is indeed fake.

Hinge loss with $\Delta = 1$:

$$\mathcal{L}_D(D(s), D(G(\mathbf{S}))) = \sum_{k=1}^3 \max(0, 1 - D_k(s)) + \max(0, 1 + D_k(G(\mathbf{S}))).$$

y = A speech wave (either real or generated)

s = A ground truth (real) speech wave

\mathbf{S} = A ground truth Mel-Spectrogram

G = Generator neural network

D_k = k -th discriminator block

Adversarial Loss Functions: Generator's Perspective

➤ Discriminator's output score:

- Positive score $D_k(\mathbf{y}) > 0 \Rightarrow \mathbf{y}$ is real.
- Negative score $D_k(\mathbf{y}) < 0 \Rightarrow \mathbf{y}$ is fake.
- Higher score in magnitude \Rightarrow stronger belief about \mathbf{y} (and vice versa).

➤ Generator's objective:

- Maximize $D_k(\mathbf{y})$ (or minimize $-D_k(\mathbf{y})$) for all k when \mathbf{y} is indeed generated.
- Minimize the distance between real and generated audio feature maps.

Base Generator Loss:

$$\mathcal{L}_G^{\text{BASE}}(D(G(\mathbf{S}))) = -\sum_{k=1}^3 D_k(G(\mathbf{S})).$$

Feature Map Generator Loss¹:

$$\mathcal{L}_G^{\text{FMAP}}(D(s), D(G(\mathbf{S}))) = \sum_{i=1}^T \frac{1}{N_i} \left\| D_k^{(i)}(s) - D_k^{(i)}(G(\mathbf{S})) \right\|_1.$$

Total Generator Loss:

$$\mathcal{L}_G(D(s), D(G(\mathbf{S}))) = \mathcal{L}_G^{\text{BASE}}(D(G(\mathbf{S}))) + \lambda \cdot \mathcal{L}_G^{\text{FMAP}}(D(s), D(G(\mathbf{S}))).$$

y = A speech wave

s = A ground truth speech wave

\mathbf{S} = The Mel-Spectrogram of s

G = Generator neural network

D_k = k -th discriminator block

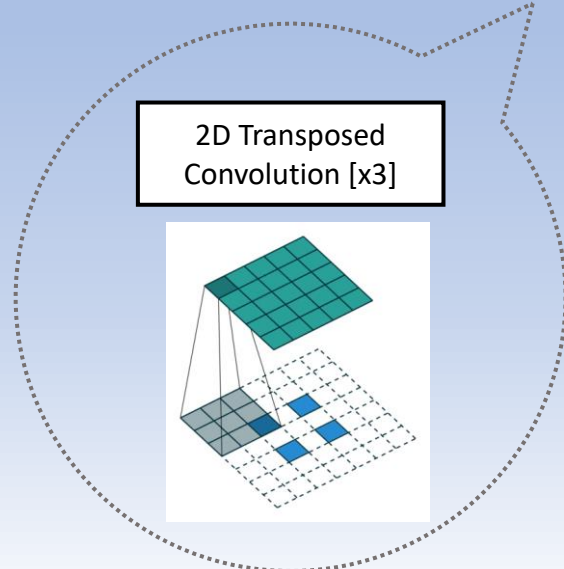
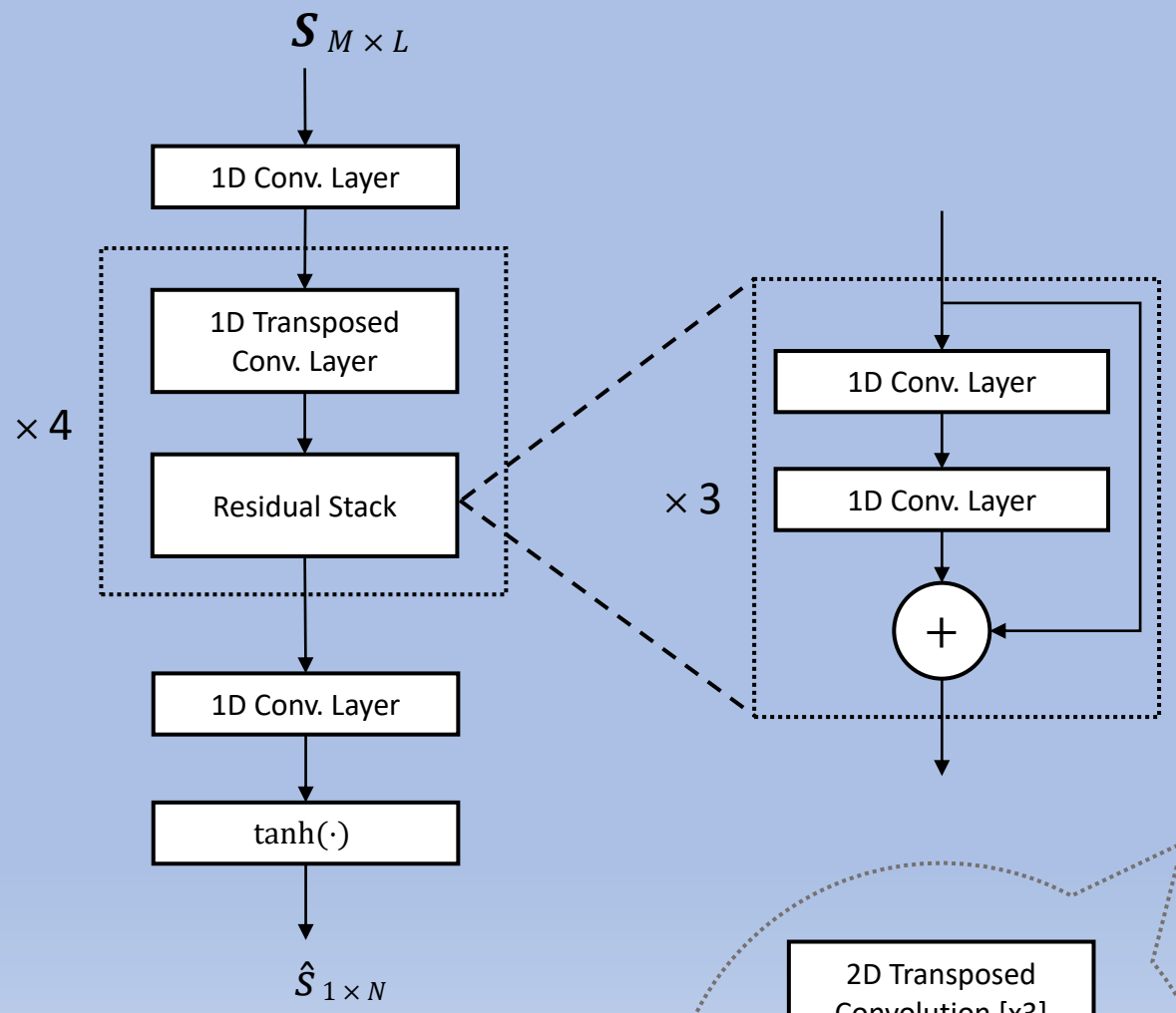
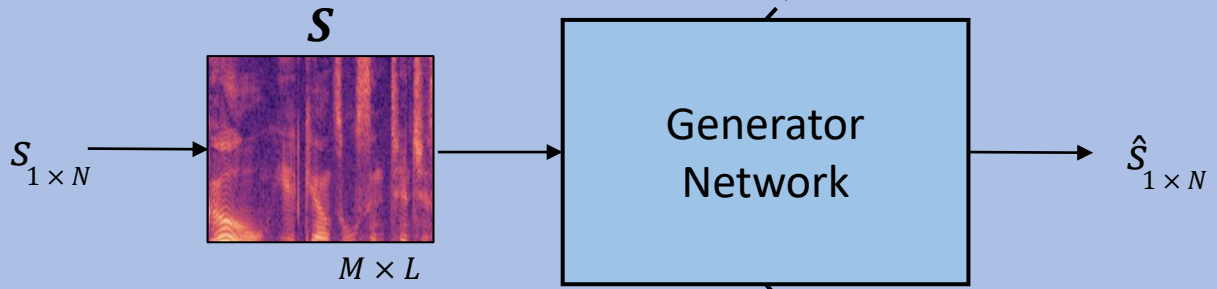
N_i = Number of units of the i -th layer

T = Number of feature maps

$\lambda = 10$, Regularization strength

¹ Similar to the perceptual loss in image processing.

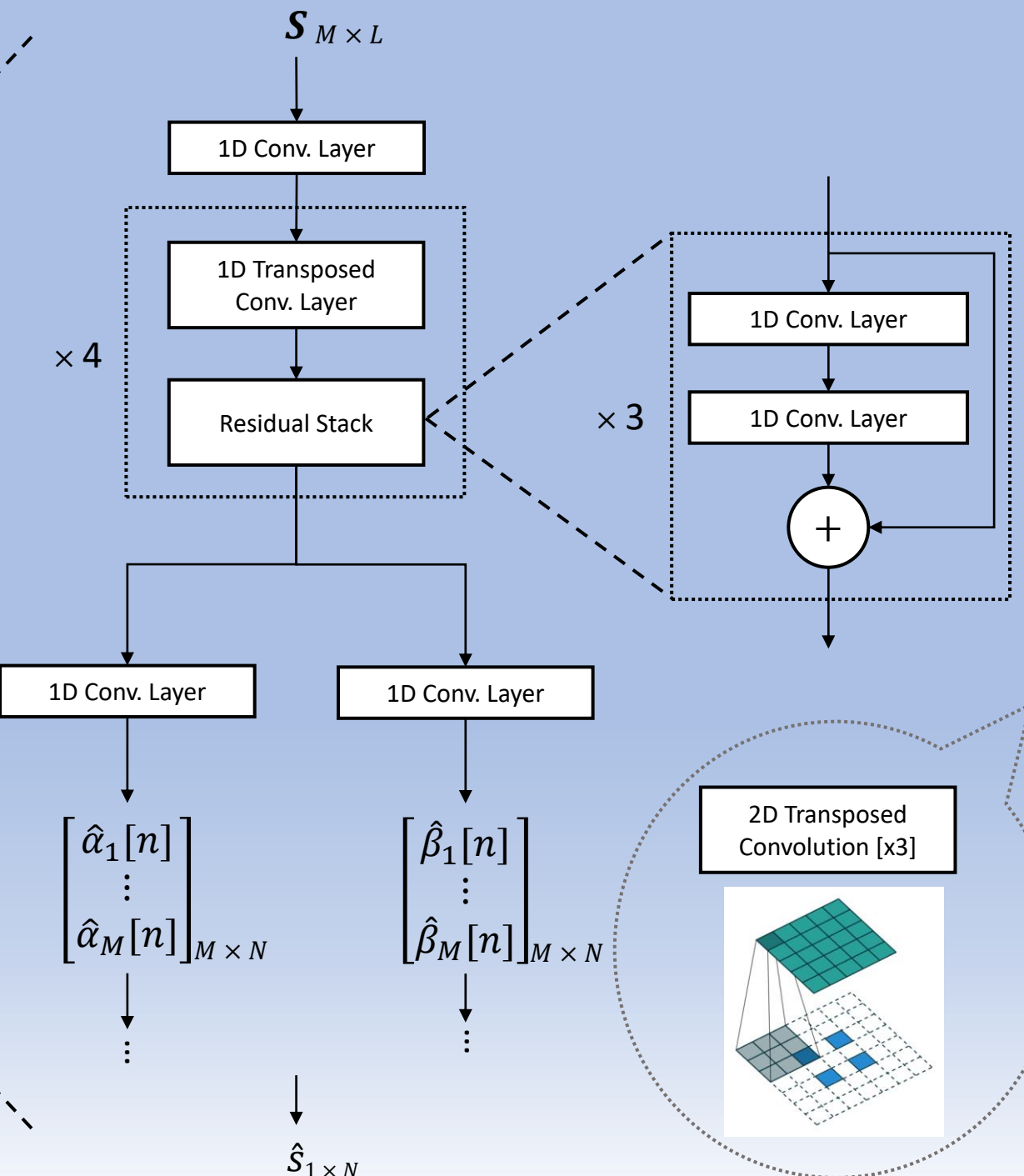
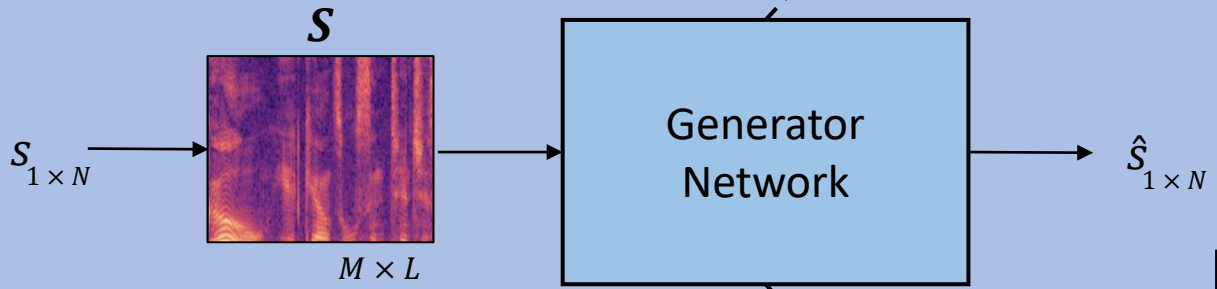
Neural Network Architecture: Extending MelGAN's Generator



➤ Depth is needed to capture long dependencies.

MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis, Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestein, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, Aaron Courville.

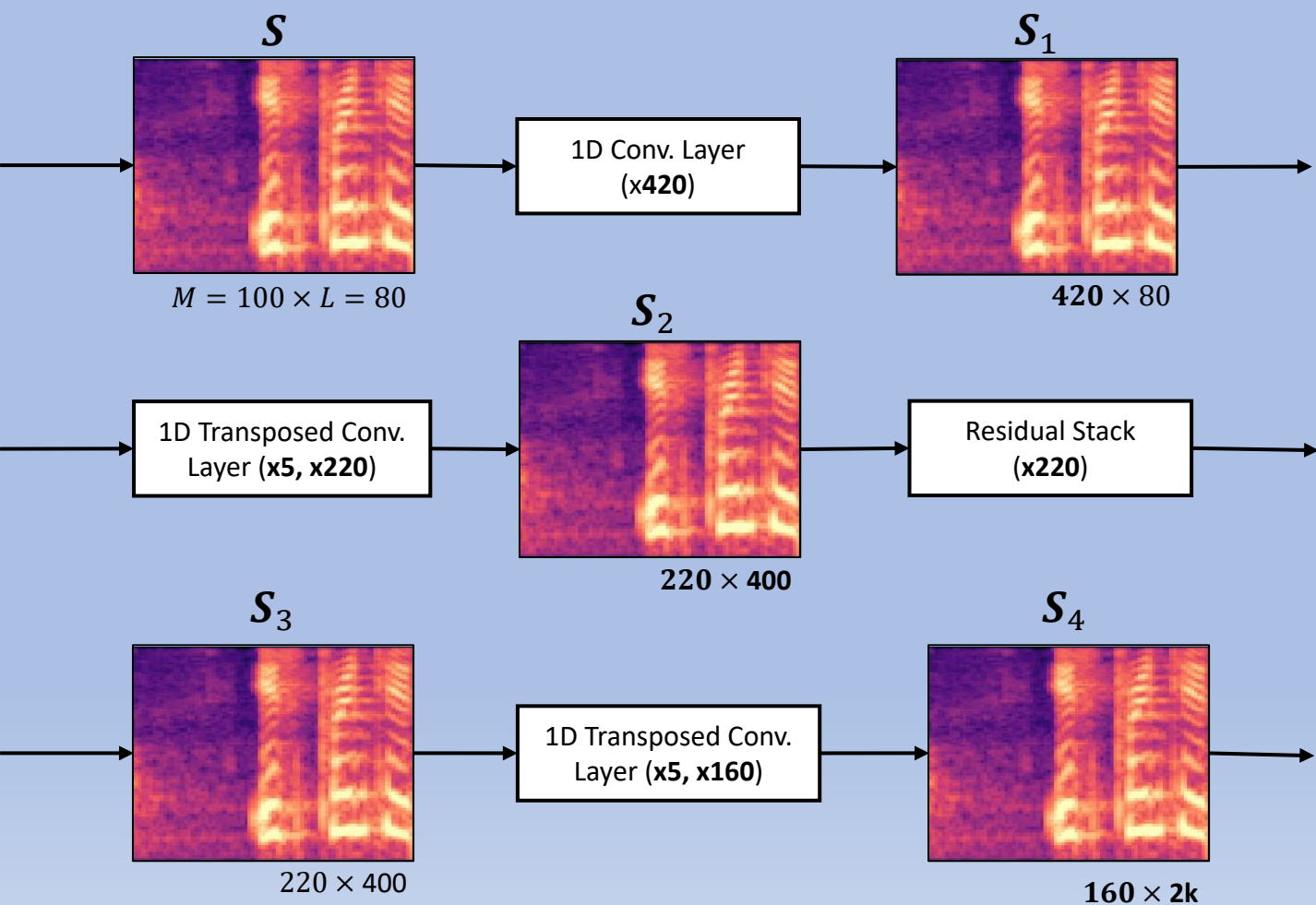
Neural Network Architecture: Extending MelGAN's Generator



➤ Depth is needed to capture long dependencies.

MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis, Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, Aaron Courville.

Example Through the Network

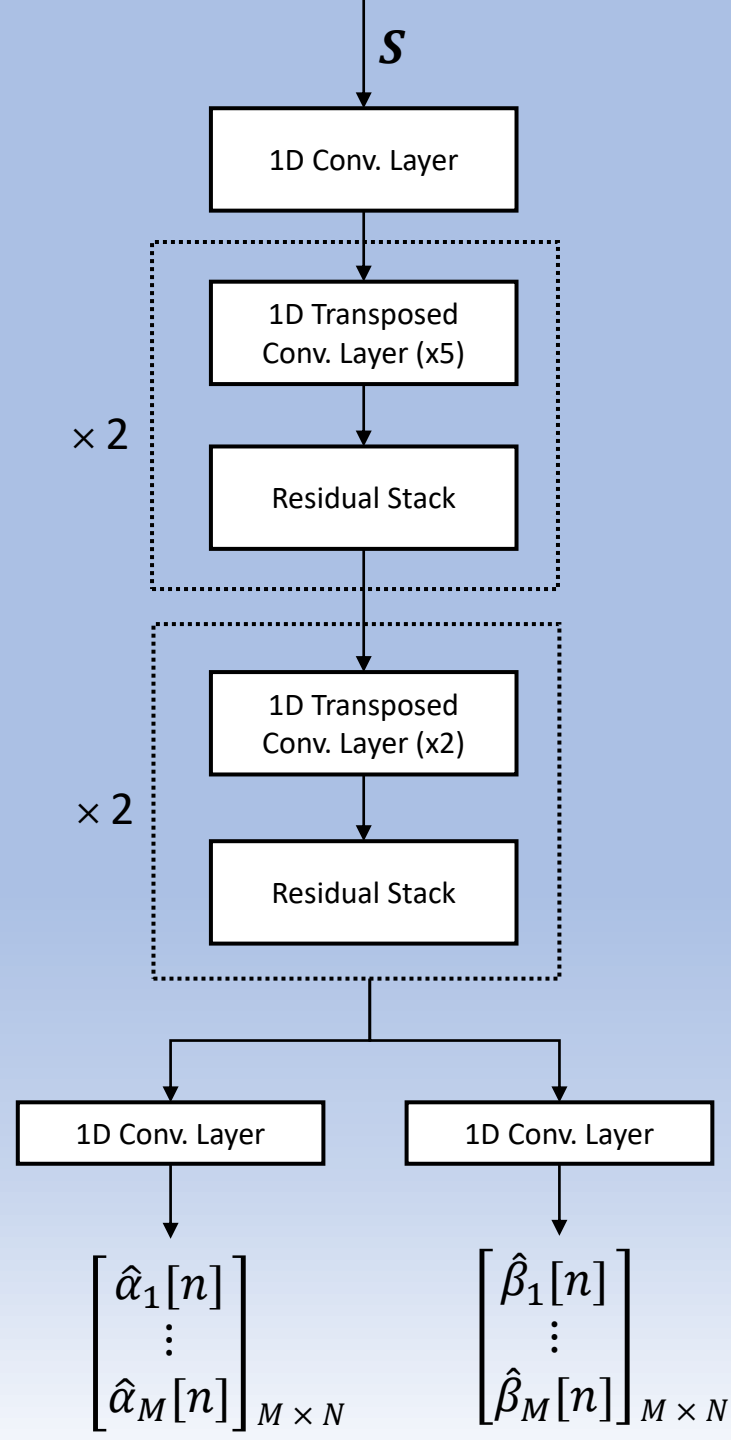


...

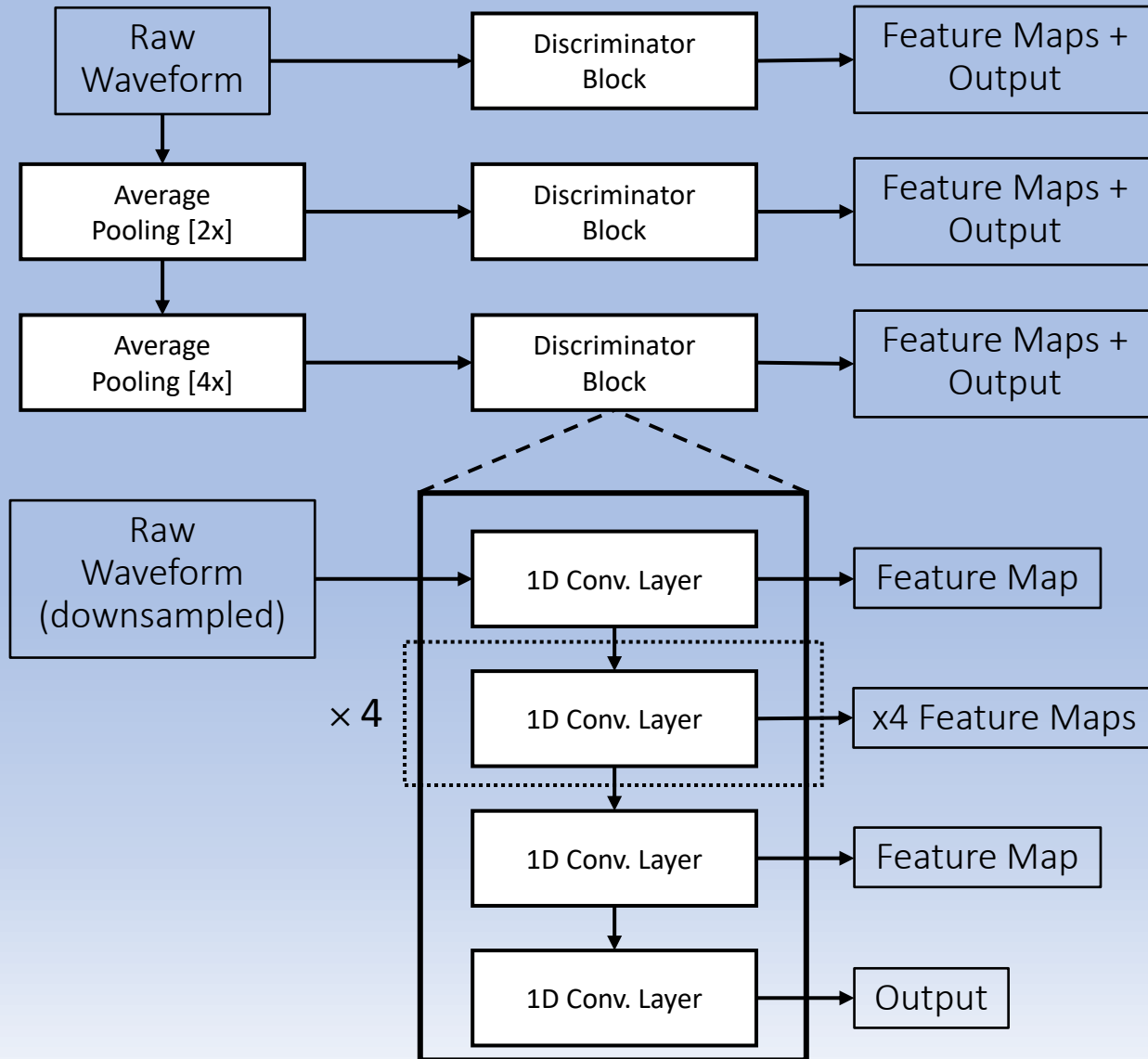
$$\begin{bmatrix} \hat{\alpha}_1[n] \\ \vdots \\ \hat{\alpha}_M[n] \end{bmatrix}_{M = 100 \times N = 8192}$$

$$\begin{bmatrix} \hat{\beta}_1[n] \\ \vdots \\ \hat{\beta}_M[n] \end{bmatrix}_{M = 100 \times N = 8192}$$

filters
420
220
160
140
$M = 100$



Neural Network Architecture: MelGAN's Discriminator



➤ Discriminator's Goal: Learn to differentiate between real (ground-truth) and fake (generated) samples.

➤ Multi-scale discriminator:

- Various scales with different downsampling ratios.
- Each scale focuses in different frequency ranges.
- Output = real number (score).
- Feature Map = layer output (the “why”).

➤ Advantages:

- ✓ Qualitative signals for training the generator.
- ✓ Discarded after the generator is trained.

➤ Disadvantages:

- ✗ Space efficiency (~16.9 MM trainable parameters).
- ✗ Training time overhead (× 2).

No GAN Approach: Spectral Loss Function Components

➤ Assumption: $s[n] \approx \hat{s}[n] \Leftrightarrow \mathbf{X}[k, m] \approx \hat{\mathbf{X}}[k, m]$

Spectral Convergence:

$$\mathcal{L}_{\text{sc}}(s[n], \hat{s}[n]) = \frac{\left\| \left| \mathbf{X}[k, m] \right| - \left| \hat{\mathbf{X}}[k, m] \right| \right\|_F}{\left\| \left| \mathbf{X}[k, m] \right| \right\|_F}.$$

- Accounts for the more **significant differences**.
- More important in the **early** stages of the training.
- Frobenius/Hilbert-Schmidt norm:

$$\|\mathbf{A}\|_F \triangleq \sqrt{\sum_{i=0}^{N-1} \sum_{j=0}^{K-1} |a_{i,j}|^2}, \quad \mathbf{A} \in \mathbb{C}^{N \times K}.$$

Logarithmic DSTFT Magnitude:

$$\mathcal{L}_{\text{lm}}(s[n], \hat{s}[n]) = \frac{1}{N} \left\| \ln |\mathbf{X}[k, m] + \epsilon| - \ln |\hat{\mathbf{X}}[k, m] + \epsilon| \right\|_1.$$

- Focuses more on **details**.
- More important in the **later** stages of the training.
- L_1 norm:

$$\|\mathbf{A}\|_1 \triangleq \sum_{i=0}^{N-1} \sum_{j=0}^{K-1} |a_{i,j}|, \quad \mathbf{A} \in \mathbb{C}^{N \times K}.$$

s = The ground truth speech wave

\hat{s} = The estimated speech wave

$\mathbf{X}[k, m]$ = The DSTFT of s

$\hat{\mathbf{X}}[k, m]$ = The DSTFT of \hat{s}

K = Number of DSTFT frequencies

N = Length of s or \hat{s} in samples

$\epsilon = 2$, Regularization strength

Spectral Loss Function

- Speech spectrograms reveal different information when analyzed in different resolutions:
- **Wideband** (small analysis window) gives more information about the **formants, bursts, excitation pulses**, etc.
 - **Narrowband** (long analysis window) gives more information about the **F_0 , harmonics, pitch**, etc.
 - We want our loss function to capture all these characteristics, so we use a **multi-resolution** loss.

Summing over different DSTFT parameters:

$$L_s(s, \hat{s}) = \sum_{i=1}^3 L_{sc}^{(i)}(s, \hat{s}) + \lambda \cdot L_{lm}^{(i)}(s, \hat{s}).$$

Entire Spectral Loss Function:

$$\mathcal{L}(s, \hat{s}) = L_s(s, \hat{s}) + L_s(\Delta s, \Delta \hat{s}).$$

DFT Size (K_i)	Window Size (L_i)	Hop Size (U_i)
2048	1200	240
1024	1024	256
512	240	50

s = The ground truth speech wave

\hat{s} = The estimated speech wave

$\mathbf{X}[k, m]$ = The DSTFT of s

$\hat{\mathbf{X}}[k, m]$ = The DSTFT of \hat{s}

$\lambda = 9$, Regularization strength

$$\Delta x[n] = x[n+1] - x[n], \quad n \in \llbracket 0, N-2 \rrbracket.$$

Difference in time acts like a high-pass filter:

$$\mathcal{F}\{x[n] - x[n-1]\} = \left(1 - e^{\frac{j2\pi k}{K}}\right) X[k].$$

Four Trained Vocoder Models

Generative Adversarial Network

MelGAN		SinGAN	
Generator ~4.26 MM trainable parameters	Discriminator ~16.9 MM trainable parameters	Generator ~4.14 MM trainable parameters	Discriminator ~16.9 MM trainable parameters
~21.16 MM total trainable parameters		~21.04 MM total trainable parameters	

Spectral Loss Function, no Discriminator

MelNoGAN	SinNoGAN
~4.26 MM total trainable parameters	~4.14 MM total trainable parameters

➤ Training Hyperparameters:

- 3000 epochs on the LJ speech dataset.
- 1 Epoch = all files have been sampled once.
- 1 sample = window taken from a file uniformly at random.
- 22.8 hours of speech for training, 1.2 hours for testing, 8 files excluded for validation purposes.
- More training details and hyperparameters on the thesis document.

Quality & Speed Assessment

➤ Mean Opinion Score (MOS) Experiment¹:

- 15 random samples inferred from all models + ground truth.
- 5-second long each + ground truth as the ideal reference sample.
- 1-5 integer evaluation score.
- 43 candidates in total.

<i>Average MOS Score</i>	MelGAN	Sinusoidal	Ground Truth
GAN	3.53 (± 0.83)	3.27 (± 0.83)	4.90 (± 0.29)
No GAN	1.76 (± 0.80)	2.01 (± 0.83)	4.90 (± 0.29)

➤ MOS Results:

- Sinusoidal slightly worse on the GAN experiment.
- Sinusoidal slightly better on the non-GAN experiment.





















<i>Average Training Speed</i>	MelGAN	Sinusoidal
GAN	236.67 ms/batch	515.10 ms/batch
No GAN	105.45 ms/batch	258.78 ms/batch
<i>Average Inference Speed</i>	2.34 min of speech/sec	1.52 min of speech/sec

➤ Speed Results:

- Discriminator adds a $\times 2$ computational overhead.
- Sinusoidal extension adds an additional $\times 2$ overhead.

¹ No established objective measures for speech quality assessment exist (still an open problem).

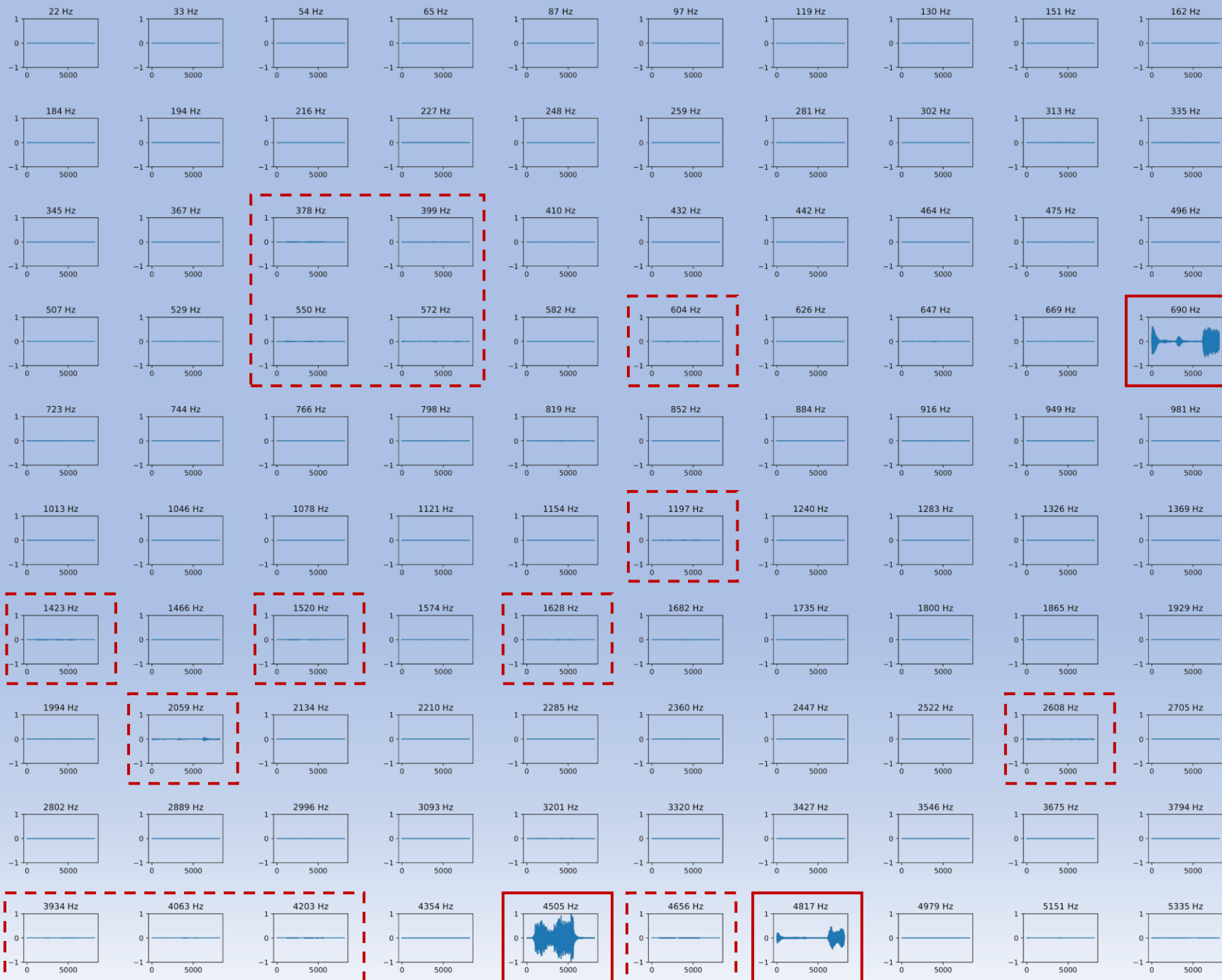
Audio Samples

Ground Truth	MeIGAN	SinGAN	MeINoGAN	SinNoGAN	Text
					"...dissimilarity of its protective functions to most activities of the department of the treasure..."
					"...the vice presidential vehicle, although not specially designed for that..."
					"...the service has experimented with the use of agents borrowed for short periods from..."
					"...which provides, quote, liquor, use of..."

Once can notice the GAN approaches producing very similar audio with few similar artifacts present.

Approaches without a discriminator are lower in quality due to similar artifacts plus a buzzing effect that is being generated throughout the waveform.

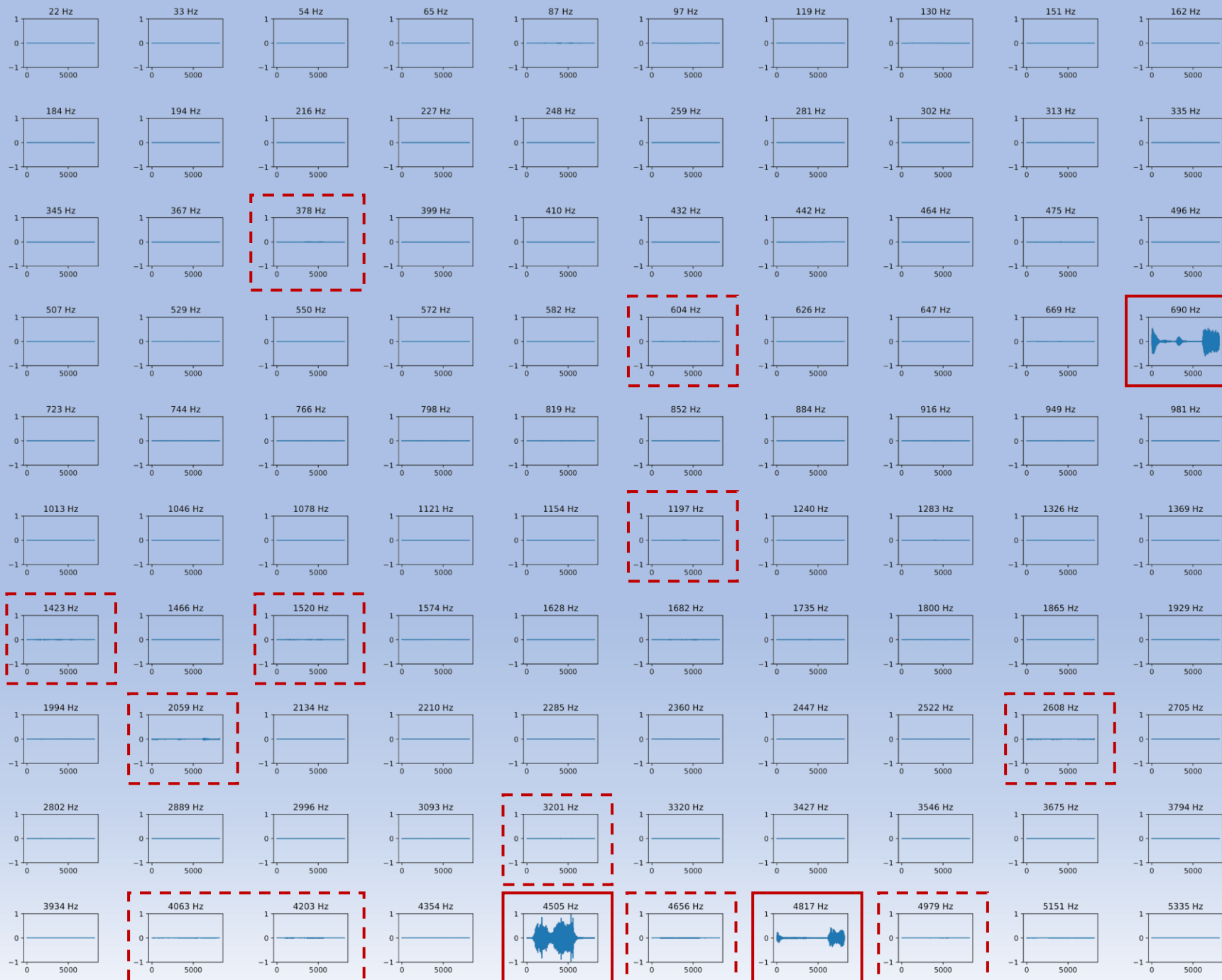
Amplitude Modulators



➤ SinGAN's $\hat{\alpha}_m[n]$ on random speech inputs:

- Very few ($\sim 3-4$) sinusoids used predominantly.
- Many sinusoids ($\sim 15-20$) with very small amplitude.

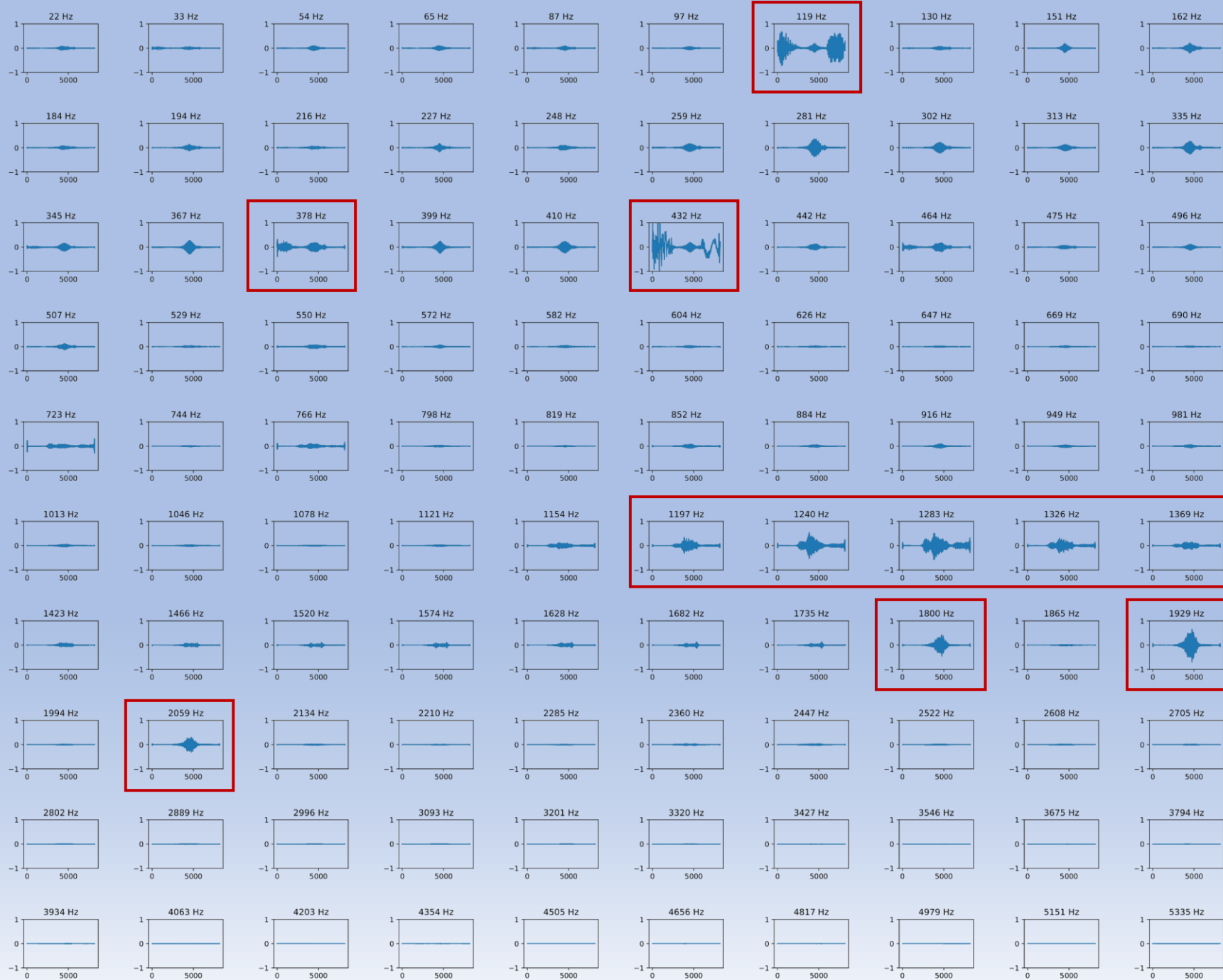
Amplitude Modulators



➤ SinGAN's $\hat{\beta}_m[n]$ on random speech inputs:

- Very few ($\sim 3-4$) sinusoids used predominantly.
- Many sinusoids ($\sim 15-20$) with very small amplitude
- Phase is indeed compensated using the sinusoid pairs.

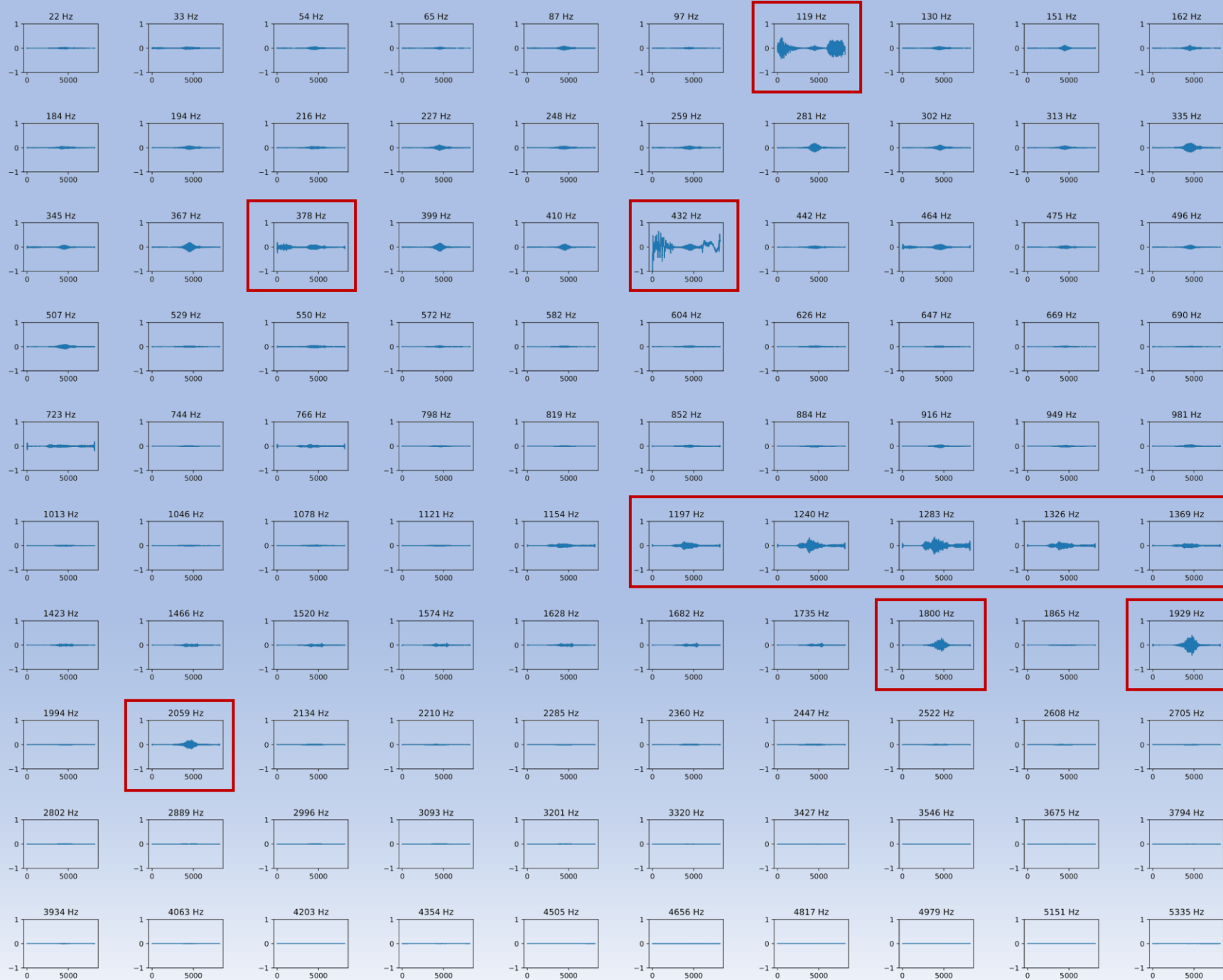
Amplitude Modulators



➤ SinNoGAN's $\hat{a}_m[n]$ on random speech inputs:

- Many more (~20) sinusoids used predominantly.
- Almost all frequencies are used to an extent, with lower ones used more as expected.

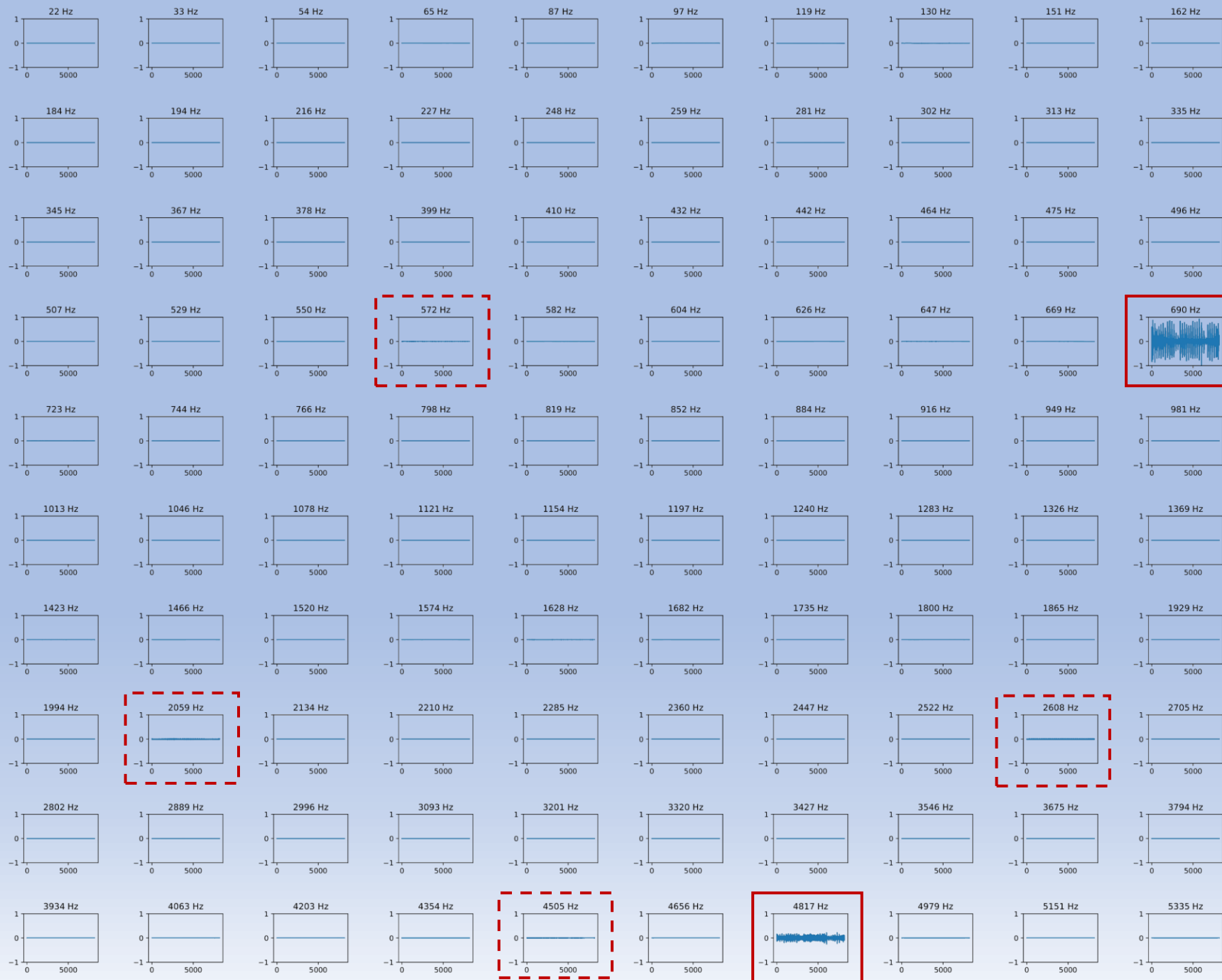
Amplitude Modulators



➤ SinNoGAN's $\hat{\beta}_m[n]$ on random speech inputs:

- Same frequencies used as in the previous figure.
- Slightly lower in amplitude than $\hat{\alpha}_m[n]$.
- Betas modulators are omitted for the following figures.

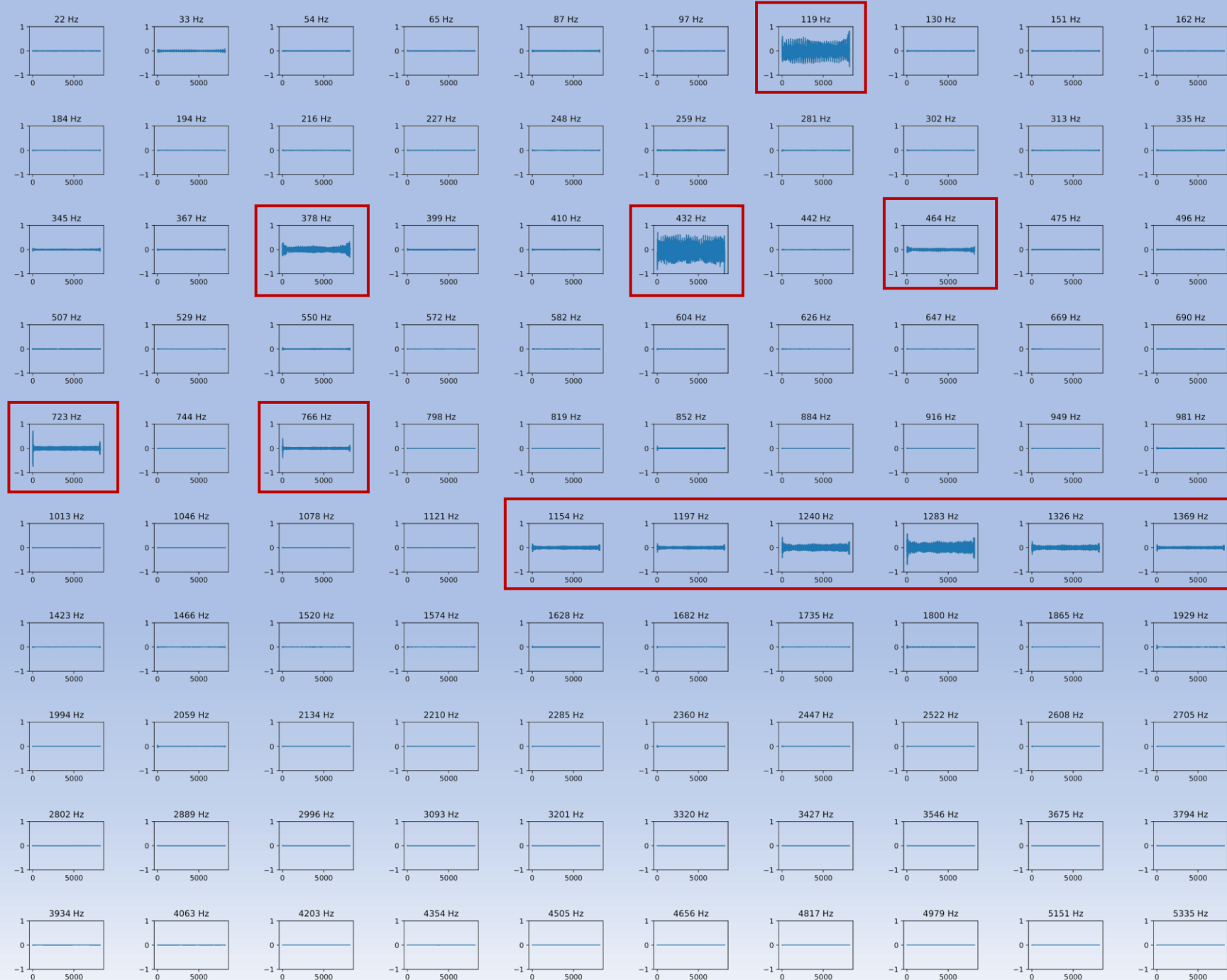
Amplitude Modulators



➤ SinGAN's $\hat{\alpha}_m[n]$ on a plain /a/ vowel signal:

- Only two frequencies used.
- Lower one being higher in amplitude because of /a/.

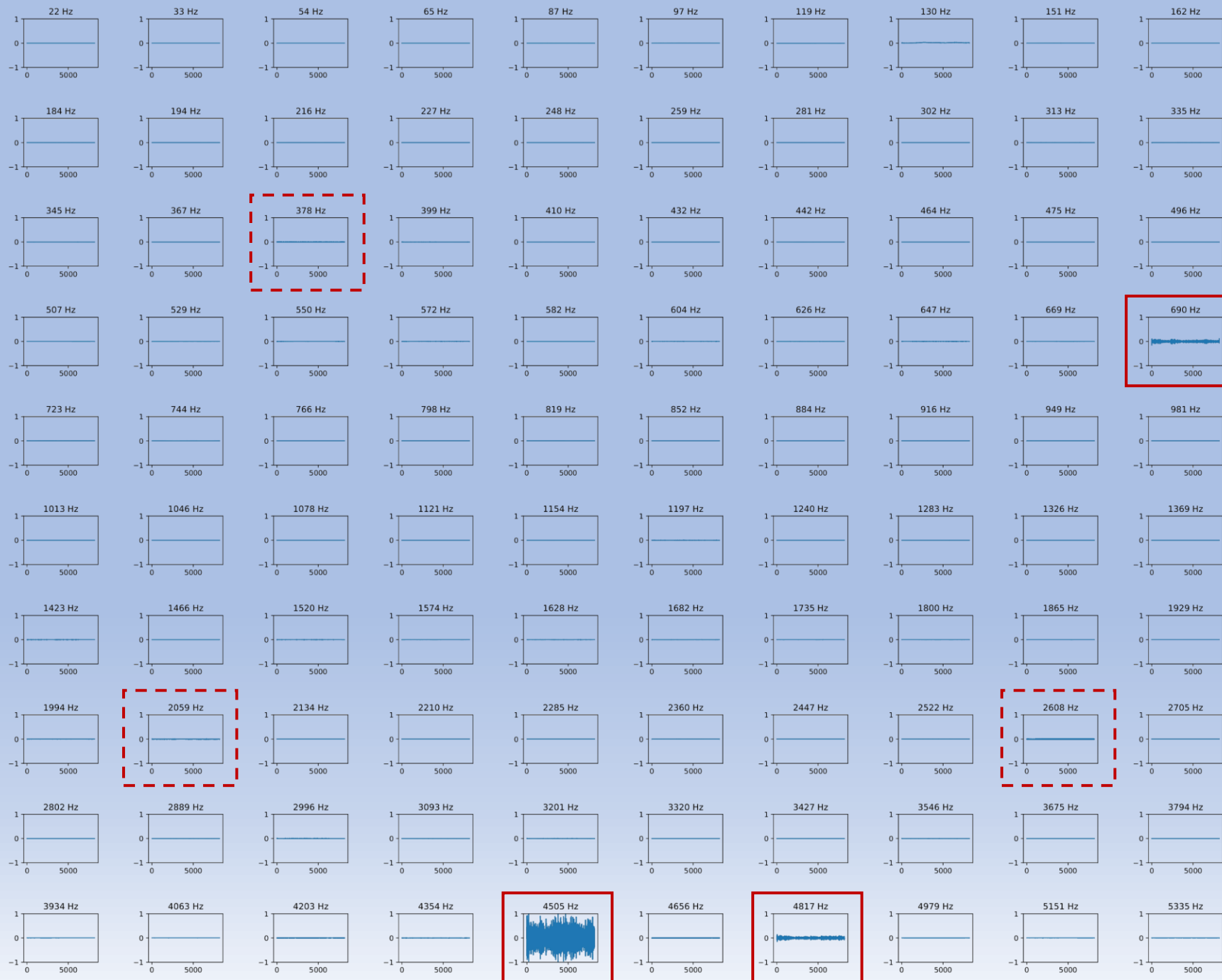
Amplitude Modulators



➤ SinNoGAN's $\hat{\alpha}_m[n]$ on a plain /a/ vowel signal:

- Less frequencies used (~ 10) as well.
- Lower ones are higher in amplitude as well.
- Still way more than SinGAN.

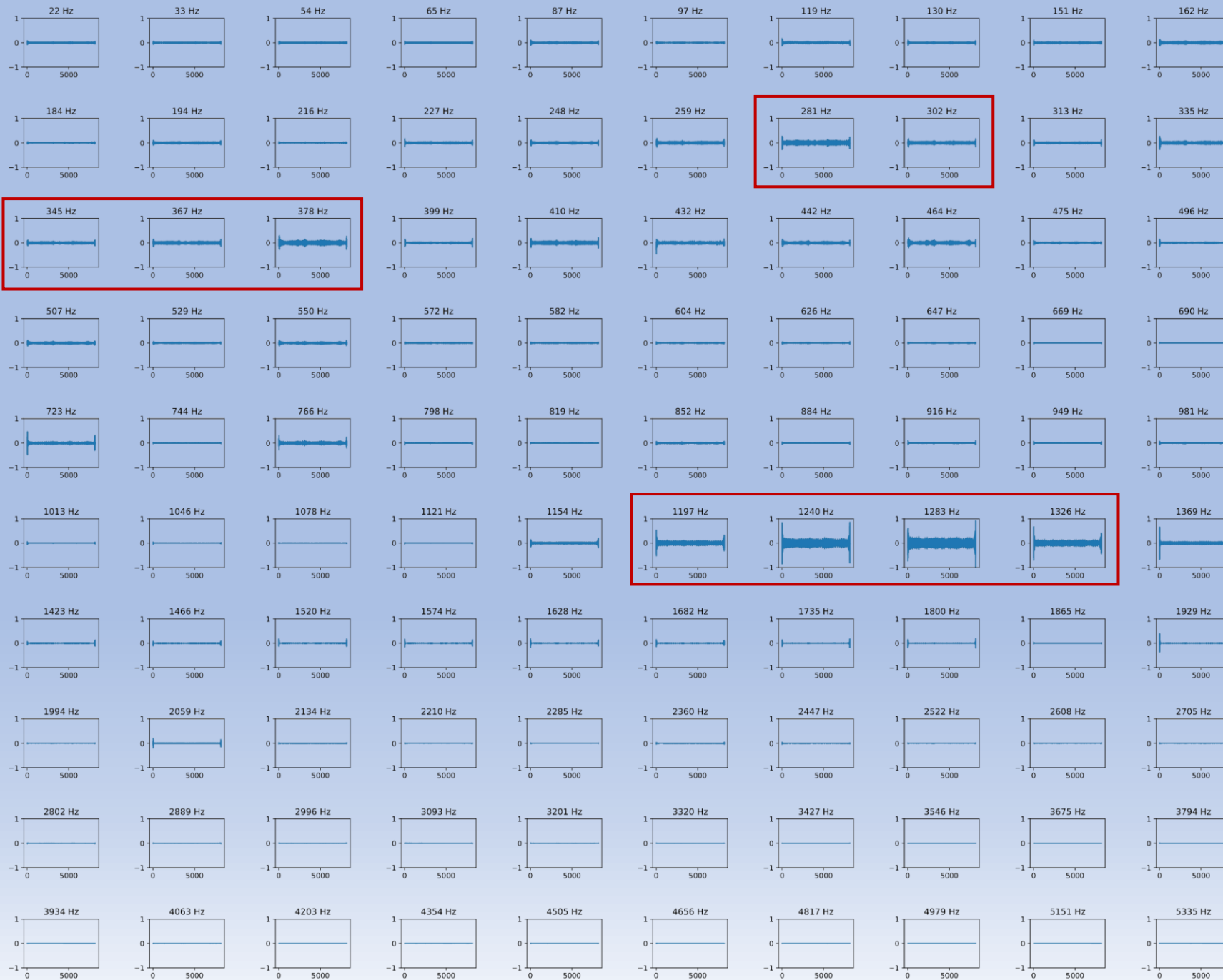
Amplitude Modulators



➤ SinGAN's $\hat{\alpha}_m[n]$ on a plain /s/ consonant signal:

- The same two-three frequencies mostly used.
- Higher ones are higher in amplitude due to /s/.

Amplitude Modulators



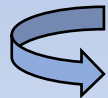
➤ SinNoGAN's $\hat{\alpha}_m[n]$ on a plain /s/ consonant signal:

- Less sinusoids needed compared to more complicated signals.
- Still a lot more than SinGAN.
- Higher frequencies are also higher in amplitude due to /s/.

Main Issues

➤ Number of sinusoids & quality:

- ✘ SinGAN does not take advantage of the sinusoidal approach to a satisfactory degree:
Most sinusoids are very low in amplitude, essentially not utilized.
- ✘ SinNoGAN needs a boost quality-wise.
- ✓ Non-GAN approach utilizes a lot more sinusoids in general.
- ✓ Spectral-based loss rewards the use of more frequencies more easily.
- 💡 Add spectral loss components in the GAN approach.
- 💡 Change the way the network learns to produce AM-FM waves.

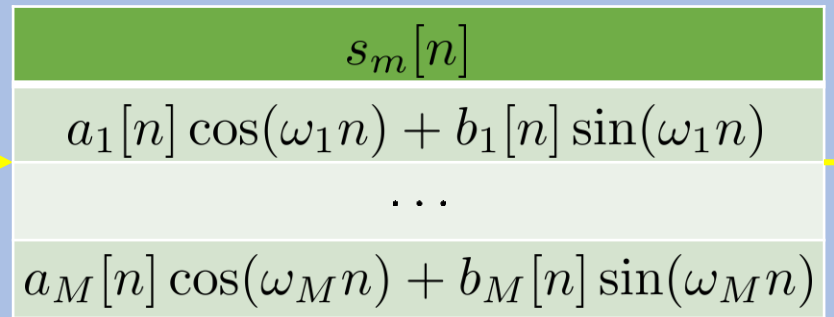
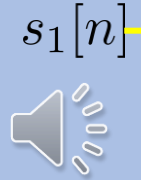
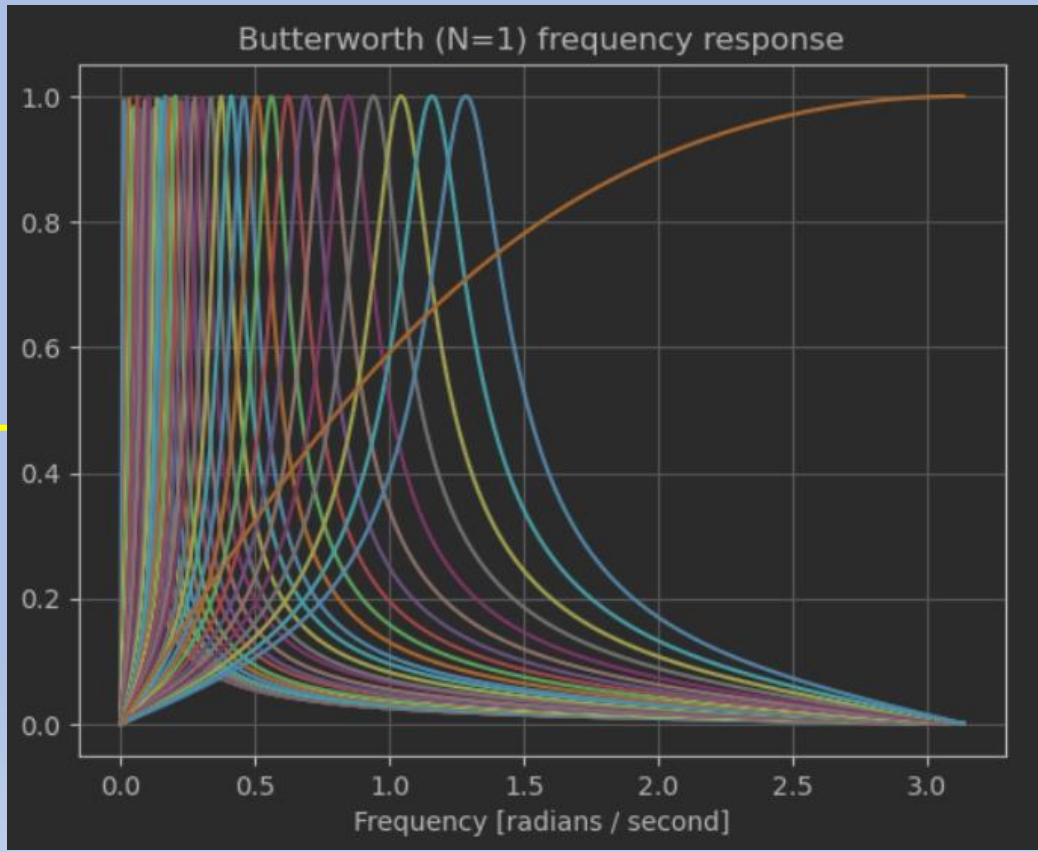


Can we somehow compute the ground truth amplitude modulators α , β ?

Step 1)

Getting the Ground Truth α, β

Butterworth Filterbank, BW: $[\omega_i, \omega_{i+2}]$



But we also need to separate the α from the β (Step 2).

Each filter is centered at a Mel frequency

$\frac{\sum}{\max}$

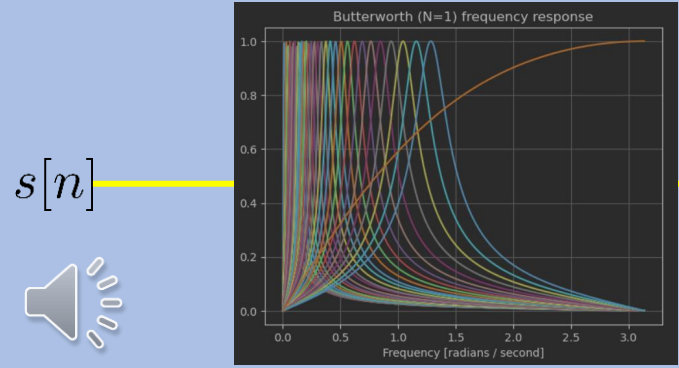
$\hat{s}_1[n]$



$RMSE(s[n], \hat{s}_1[n]) = 0.03822$

Step 1)

Getting the Ground Truth α, β



$s_m[n]$
$a_1[n] \cos(\omega_1 n) + b_1[n] \sin(\omega_1 n)$
...
$a_M[n] \cos(\omega_M n) + b_M[n] \sin(\omega_M n)$

Step 2)

Hilbert Transform*
 $s_m[n] \xrightarrow{H\{\cdot\}}$

$\Re\{H\{s_m[n]\}\}$	$\Im\{H\{s_m[n]\}\}$
$a_1[n] \cos(\omega_1 n)$	$b_1[n] \sin(\omega_1 n)$
...	...
$a_M[n] \cos(\omega_M n)$	$b_M[n] \sin(\omega_M n)$

We can use these AM-FM signals at the loss function now to train our estimated ones.

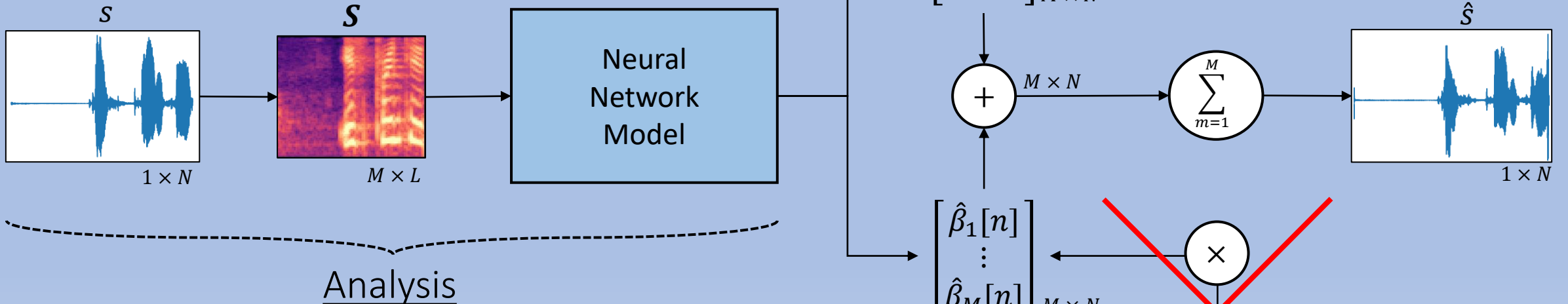
$\xrightarrow{\sum_{\max}} \hat{s}_2[n]$



RMSE($s[n], \hat{s}_2[n]$) = 0.09279

* $\mathcal{F}^{-1}(\mathcal{F}(x)2U) = x + jy$: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.hilbert.html>

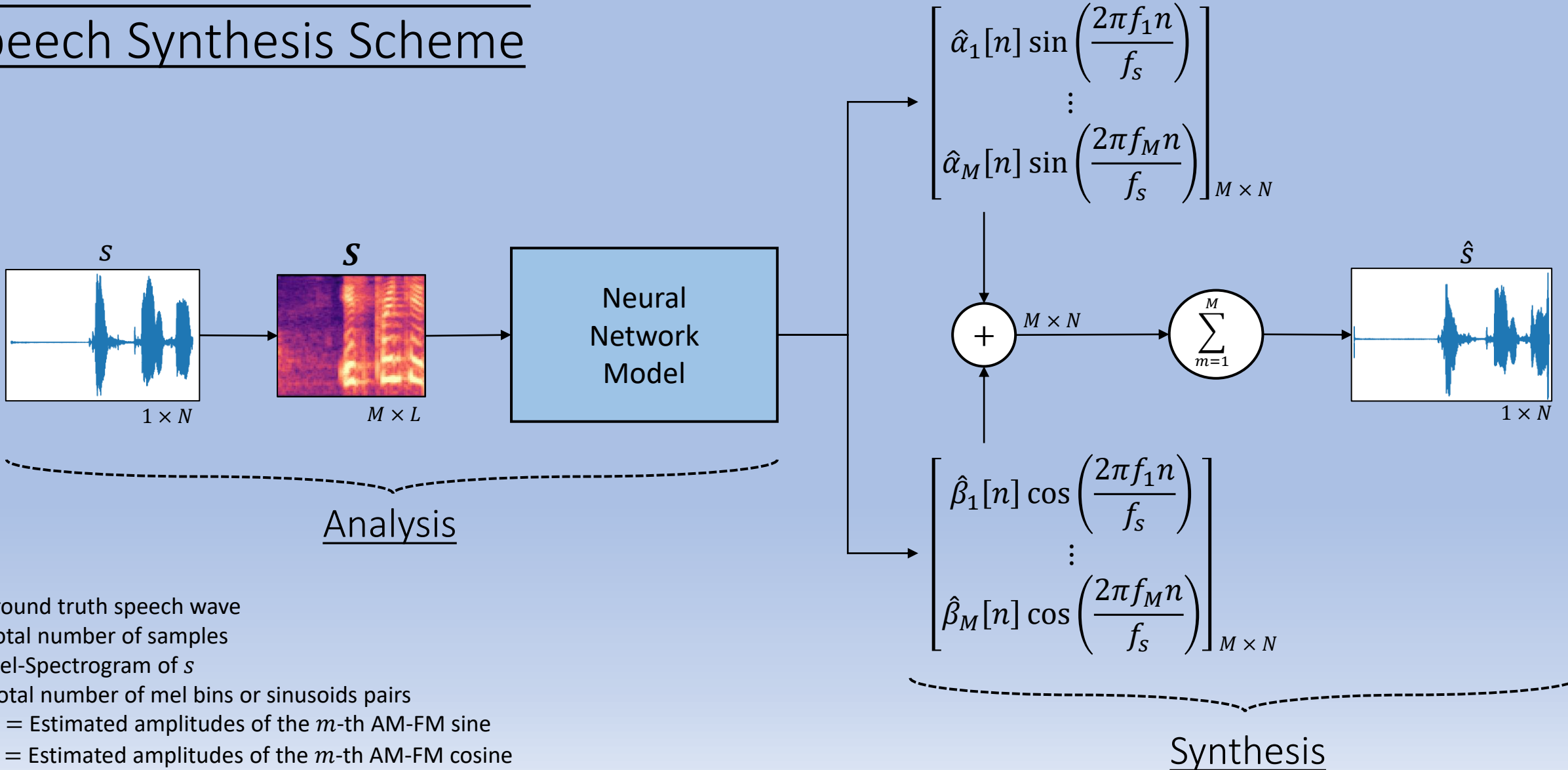
A Neural-Based Sinusoidal Speech Synthesis Scheme



s = Ground truth speech wave
 N = Total number of samples
 S = Mel-Spectrogram of s
 M = Total number of mel bins or sinusoids pairs
 $\hat{\alpha}_m[n]$ = Estimated amplitudes of the m -th AM-FM sine
 $\hat{\beta}_m[n]$ = Estimated amplitudes of the m -th AM-FM cosine
 f_m = Central frequency of the m -th mel band
 f_s = sampling rate
 \hat{s} = Estimated output speech wave

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi f_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi f_m \frac{n}{f_s} \right) \right].$$

A Neural-Based Sinusoidal Speech Synthesis Scheme



s = Ground truth speech wave
 N = Total number of samples
 \mathbf{S} = Mel-Spectrogram of s
 M = Total number of mel bins or sinusoids pairs
 $\hat{\alpha}_m[n]$ = Estimated amplitudes of the m -th AM-FM sine
 $\hat{\beta}_m[n]$ = Estimated amplitudes of the m -th AM-FM cosine
 f_m = Central frequency of the m -th mel band
 f_s = sampling rate
 \hat{s} = Estimated output speech wave

$$s[n] \approx \hat{s}[n] = \sum_{m=1}^M \left[\hat{\alpha}_m[n] \sin \left(2\pi f_m \frac{n}{f_s} \right) + \hat{\beta}_m[n] \cos \left(2\pi f_m \frac{n}{f_s} \right) \right].$$

Spectral Loss Function

- Same parameters, but also for the α , β , and no derivative-based extra loss.

$$L_s(s, \hat{s}) = \sum_{i=1}^3 L_{sc}^{(i)}(s, \hat{s}) + \lambda \cdot L_{lm}^{(i)}(s, \hat{s}).$$

$$L_{\alpha\beta}(\alpha\beta, \hat{\alpha}\hat{\beta}) = \sum_{i=1}^3 L_{sc}^{(i)}(\alpha\beta, \hat{\alpha}\hat{\beta}) + \lambda \cdot L_{lm}^{(i)}(\alpha\beta, \hat{\alpha}\hat{\beta}).$$

Entire Loss Function:

$$\mathcal{L}(s, \hat{s}) = L_s(s, \hat{s}) + L_{\alpha\beta}(\alpha\beta, \hat{\alpha}\hat{\beta}).$$

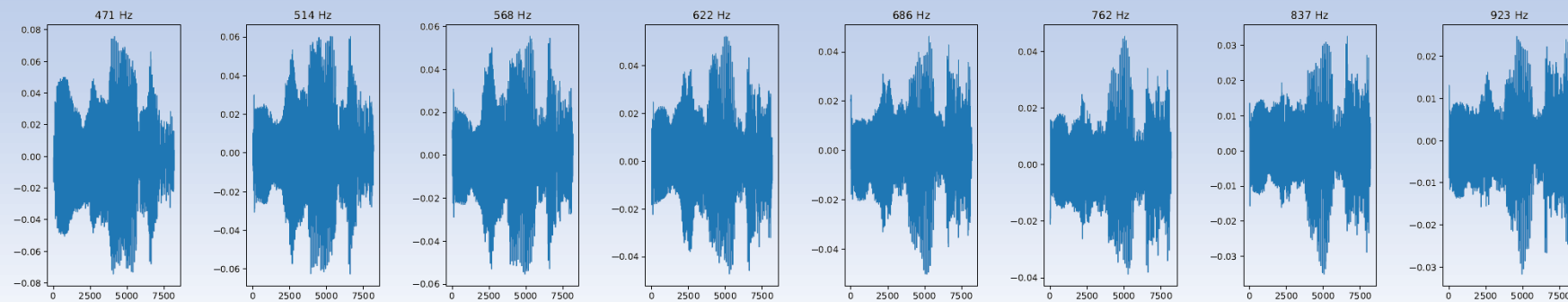
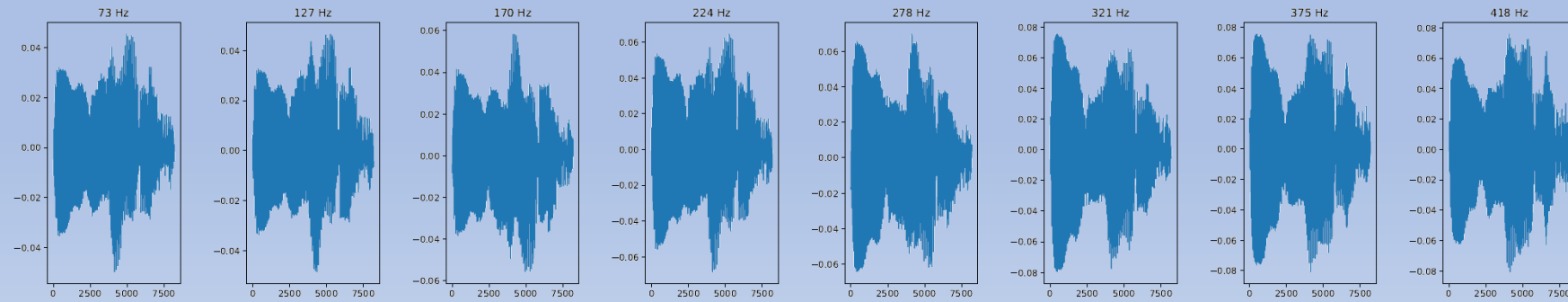
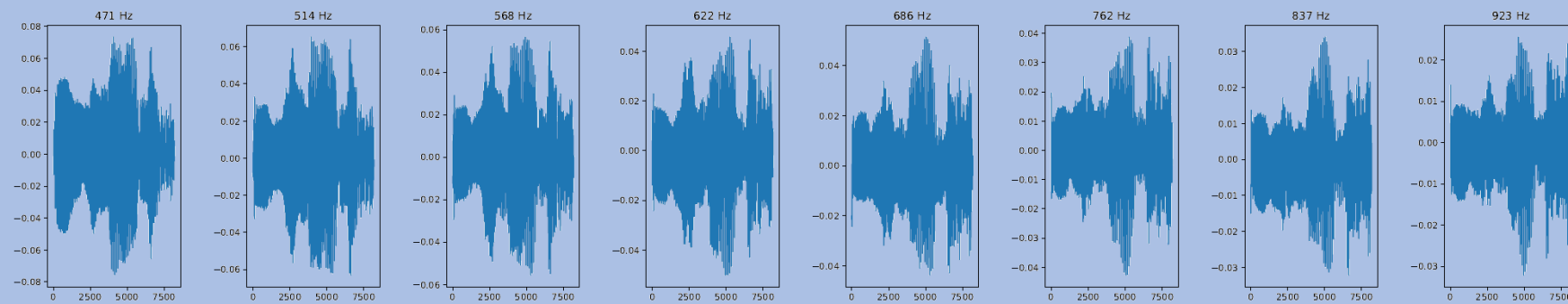
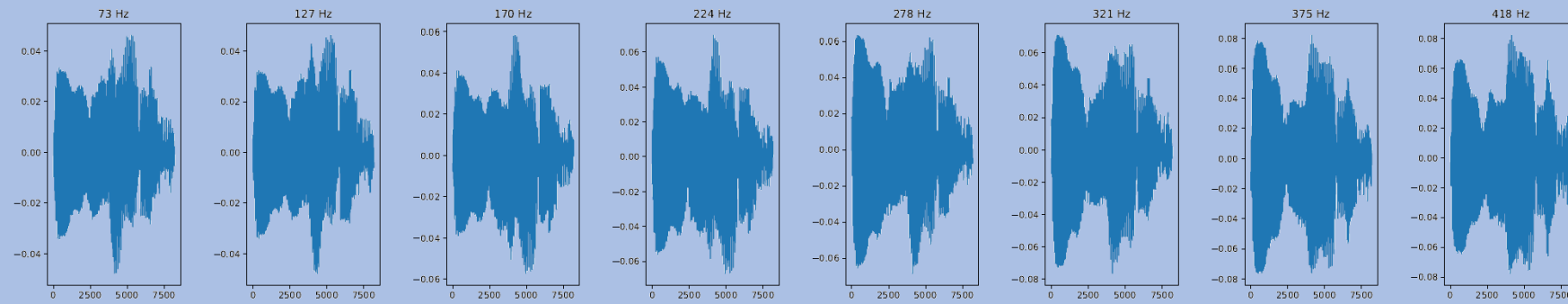
s = The ground truth speech wave
 \hat{s} = The estimated speech wave
 α, β = Ground truth AM-FM signals
computed via Butterworth & Hilbert
 $\hat{\alpha}, \hat{\beta}$ = Estimated AM-FM signals
 K = Number of DSTFT frequencies
 N = Length of s or \hat{s} in samples
 $\epsilon = 2, \lambda = 9$, Regularization strength

DFT Size (K_i)	Window Size (L_i)	Hop Size (U_i)
2048	1200	240
1024	1024	256
512	240	50

Results in Short

- Training with no Discriminator network (like SinNoGAN):
 - ✓ Our Butterworth & Hilbert technique for AM-FM speech synthesis is a novel approach.
 - ✓ No degradation in output speech quality (compared to SinNoGAN).
 - ✓ All AM-FM signals now are always used.
 - ✓ We now know exactly what our AM-FM signals represent and what the network tries to calculate (increased interpretability).
- ✗ Slower training due to increased overhead for computing the ground truth α , β spectra from the ground truth speech signal (i.e., using Butterworth & Hilbert) for the loss.

New Estimated AM-FM Signals



$$\hat{\alpha}_m[n]$$

- Butterworth & Hilbert approach, no GAN, random speech signal, first 16 AM-FM waves:

- Goal achieved: All frequencies are used.

$$\hat{\beta}_m[n]$$

Remaining Issue

➤ AM-FM waves are now utilized, but quality needs improvement:

- ✘ To justify complete removal of the discriminator, just gaining training speed & interpretability is not enough to compensate for much worse quality.
- ✘ Quality begs improving, but no established known loss function from literature can beat the strength of a Discriminator-based loss as of now. That is why GANs for speech synthesis are still a thing.
- 💡 One idea is to somehow involve a Discriminator with our new approach. We can have the Discriminator operate on the speech waves and our new spectral Butterworth & Hilbert loss operate on the alphas and betas. But, that will more than double the space and computations required during training.
- 💡 Perhaps, we can even somehow avoid the use of a Discriminator with yet another idea, named **Generalized Energy Distance** (GED). It has recently shown very promising results for the training of neural networks.

A New Hope: Generalized Energy Distance

- Rapid literature development on the GED, both theory and practice-wise. Needs many lectures to fully cover, but the core idea is:

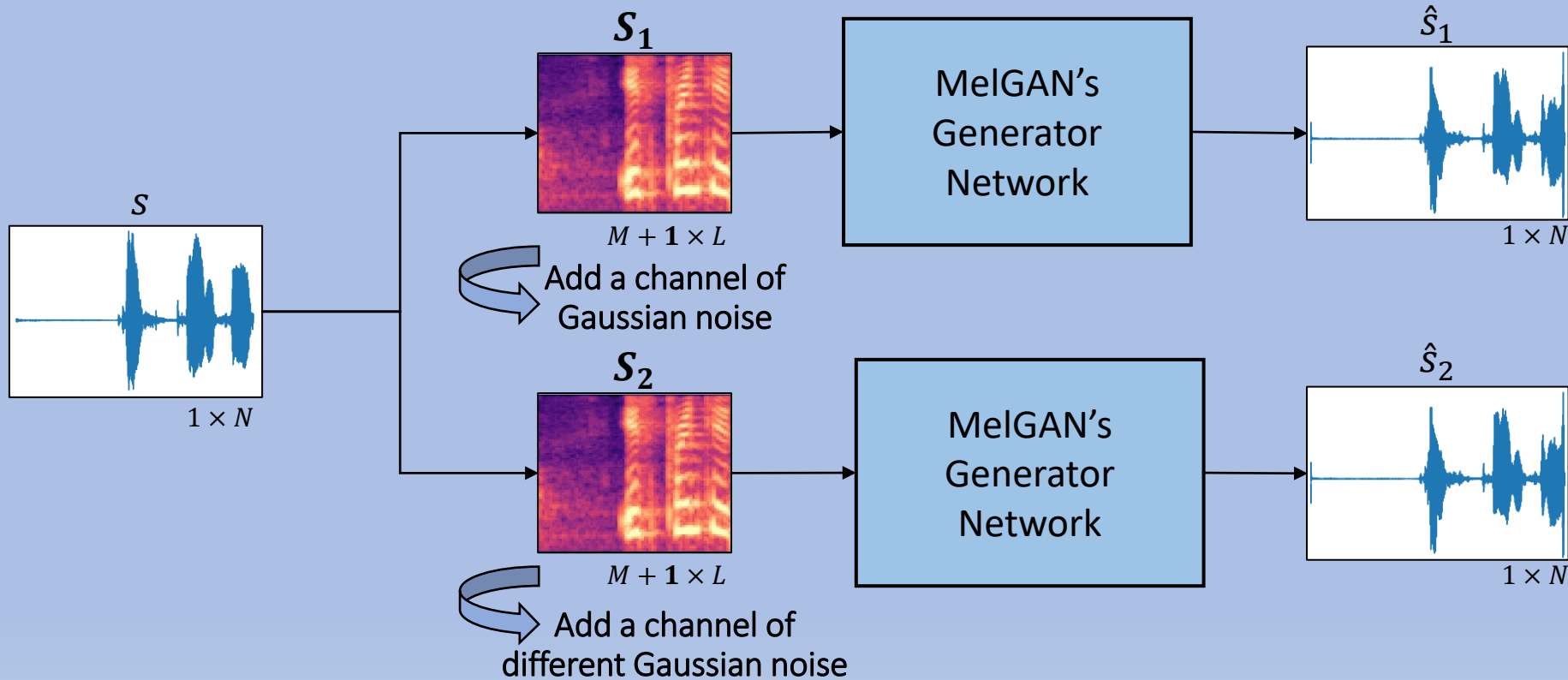


Get two different inferences from the model, then use a loss function with the following form:

$$\mathcal{L}(x, \hat{x}_1, \hat{x}_2) = \underbrace{L(x, \hat{x}_1) + L(x, \hat{x}_2)}_{\text{Attractive}} - \underbrace{L(\hat{x}_1, \hat{x}_2)}_{\text{Repulsive}}.$$

(make them the same) (make them different)

First Experiment: MelNoGAN with GED loss















$$\mathcal{L}(s, \hat{s}_1, \hat{s}_2) = L(s, \hat{s}_1) + L(s, \hat{s}_2) - L(\hat{s}_1, \hat{s}_2),$$

$$L(x[n], y[n]) = \left\| \left| \mathbf{X}[k, m] \right| - \left| \mathbf{Y}[k, m] \right| \right\|_1.$$

Results in Short

- Training with no Discriminator network (like SinNoGAN), GED loss, no AM-FM approach yet:
 - ✓ Tremendous quality upgrade, on par with the original MelGAN model.
 - ✓ Very fast training, lower resources needed, faster convergence.

Validation Samples

Ground Truth	MelGAN - Original	MelNoGAN – GED L1	Text
			“...Marguerite Oswald and Robert Oswald, were interviewed...”
			“...on individuals or groups, threatening to cause harm...”
			“...sources of danger to the president, in time...”
			“...under existing procedures, and did not know of his application.”

Future Work

➤ Loss functions:

- 💡 Use both a Discriminator and a GED loss to train the previous model for even better quality.
- 💡 Use a GED loss and/or a Discriminator to train the new AM-FM approach.
- 💡 Include more terms in the loss, e.g., a logarithmic spectral term like LM loss.
- 💡 Approach theoretically/analytically what features does the discriminator learn (future work in neural network research in general) to design better loss functions and models.

➤ Phase is neglected:

- Most neural networks used for speech processing learn phase information “from scratch”, since spectrograms inherently **do not contain** any information about the **phase**.
- 💡 Including some information about the phase in the input (or the loss function) might yield improvements. This is not an easy task, because, for instance, the phase spectrum is very noisy and sensitive to even tiny displacements in the time domain.

Conclusion

➤ Neural-based sinusoidal representations of speech are feasible:

- ✓ New solutions proposed for the problem of speech synthesis.
- ✓ Theoretically unlimited representation capabilities.
- ✓ Produced speech quality is on par with state-of-the-art models.
- ✓ Easily attachable extension to almost any architecture.
- ✓ Monitoring amplitude modulators offers very useful information about the model.
- ✓ More qualitative results with spectral-based loss functions.
- ✓ New idea of including the signal derivative in spectral losses can improve quality.
- ✓ Plenty of open room for future ideas for further improvements.

Neural Sinusoidal Modeling

From the Master Thesis of *Michail Raptakis*

Thesis Advisor: Professor *Yannis Stylianou*



University of Crete, Computer Science Department,
Voutes University Campus, 700 13 Heraklion, Crete, Greece