

# 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

➤ 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 the 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)

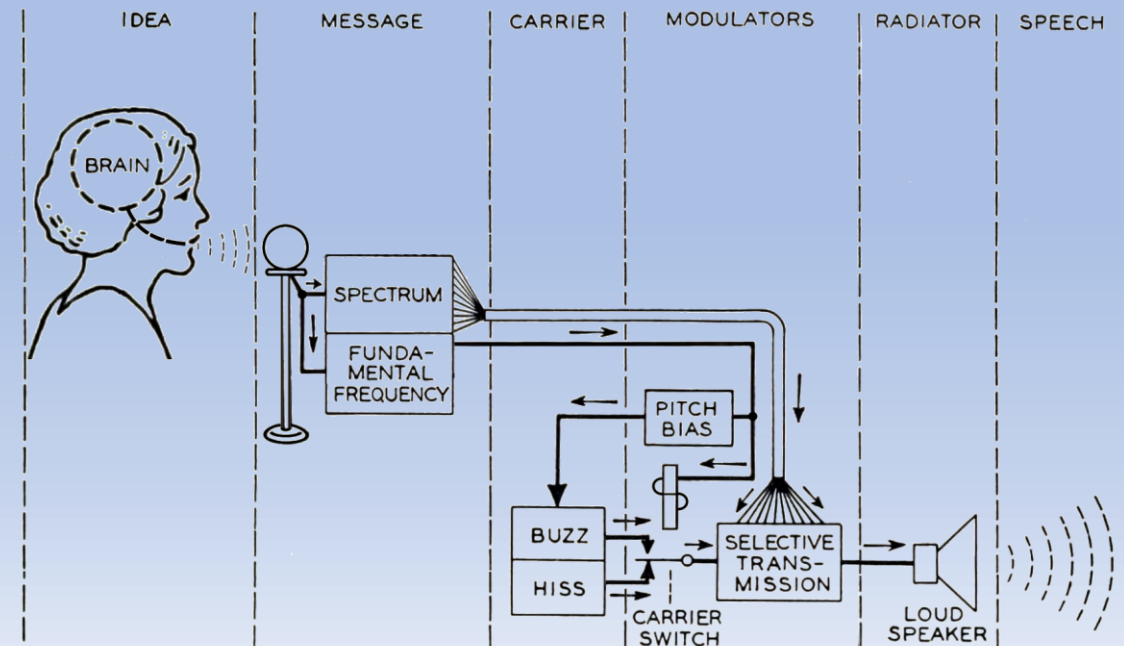
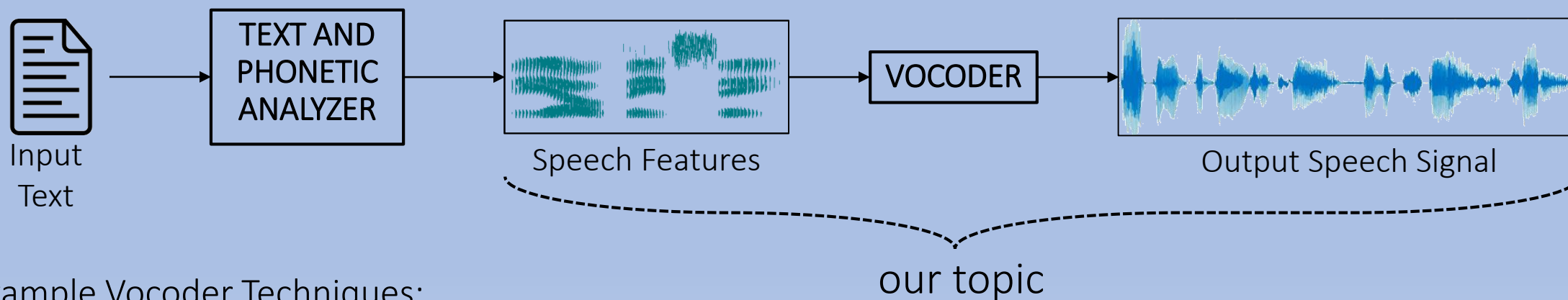


Fig. 7—Schematic circuit of the vocoder.

# Text-To-Speech Synthesis

- Definition: A **Text-To-Speech (TTS) synthesis** system converts normal language text into audible speech output.
- This problem is usually broken down into **two separate tasks**:
  - 1) Speech feature extraction (most commonly spectrograms) from the text
  - 2) Synthesizing the corresponding artificial voice given these speech features (**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 et al.

# The First Sinusoidal Vocoder

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

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

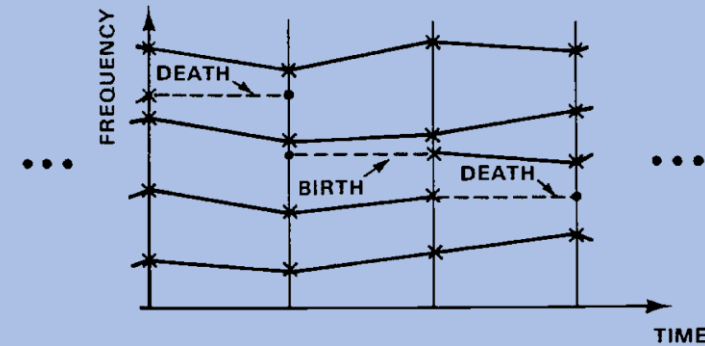
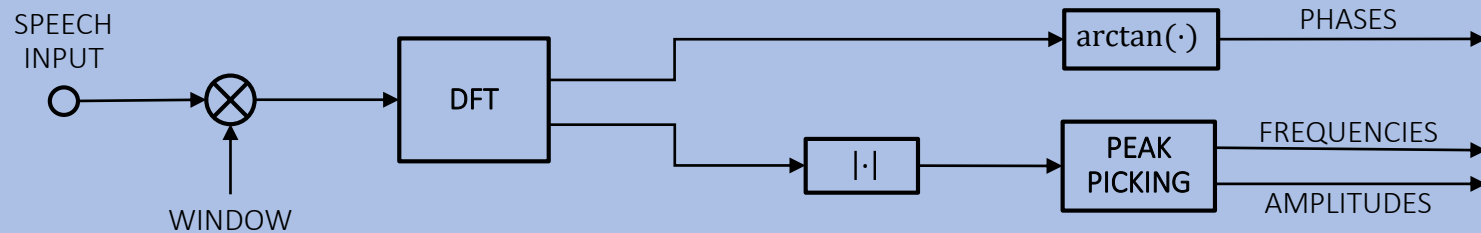


Fig. 4. Illustration of frequency tracks using the birth-death frequency tracker.

## Analysis:



## Synthesis:

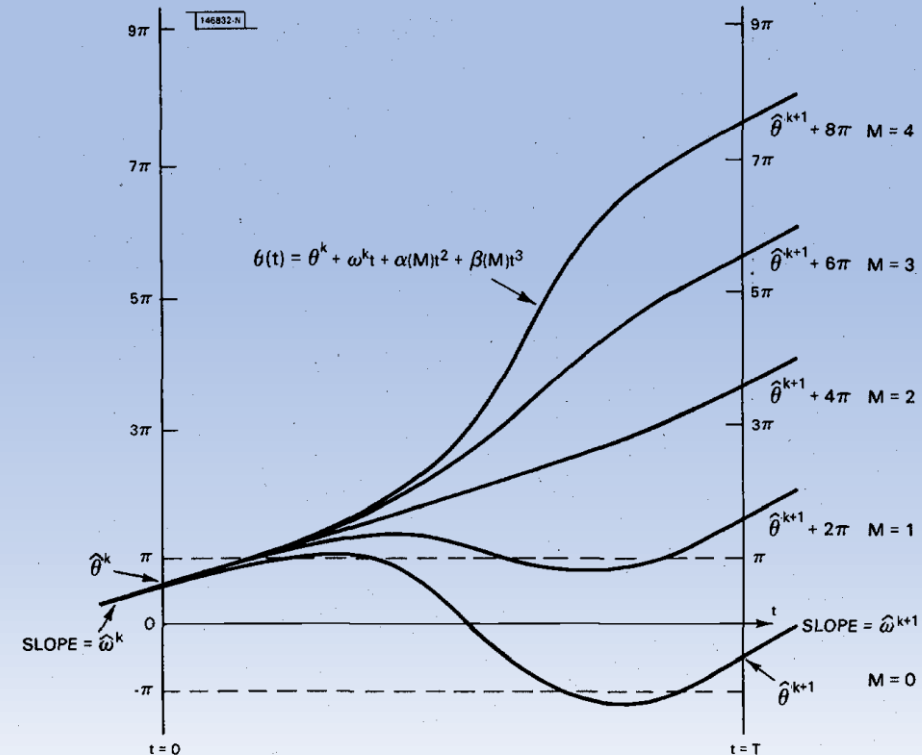
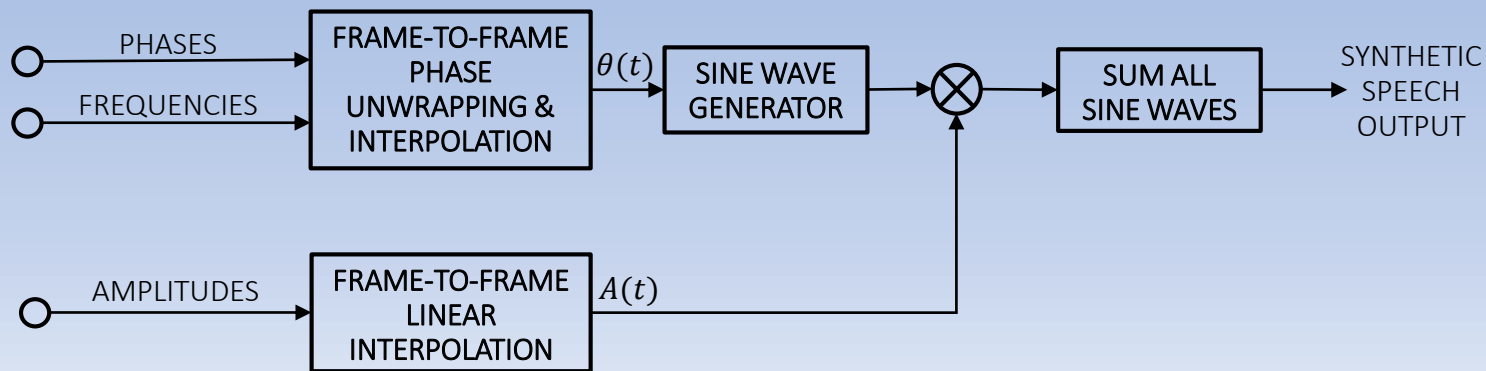


Fig. 6. Typical set of cubic phase interpolation functions.

# 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:

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

$f_s$  = 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

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

- 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:

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

$\mathbb{1}\{\cdot\}$  = 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 this an easier task
  - This shows the **limitless** representation **capabilities** from a theoretical point of view
- Hence, we can work with **constant frequencies** over **longer frames**.

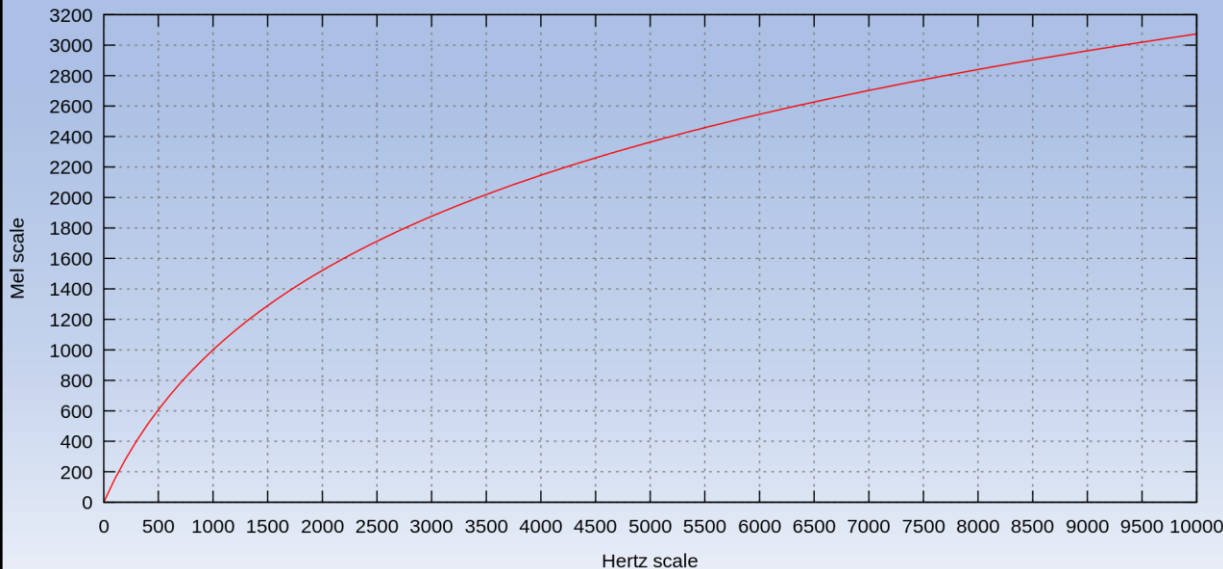
# Redefining the problem: Input

- 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**

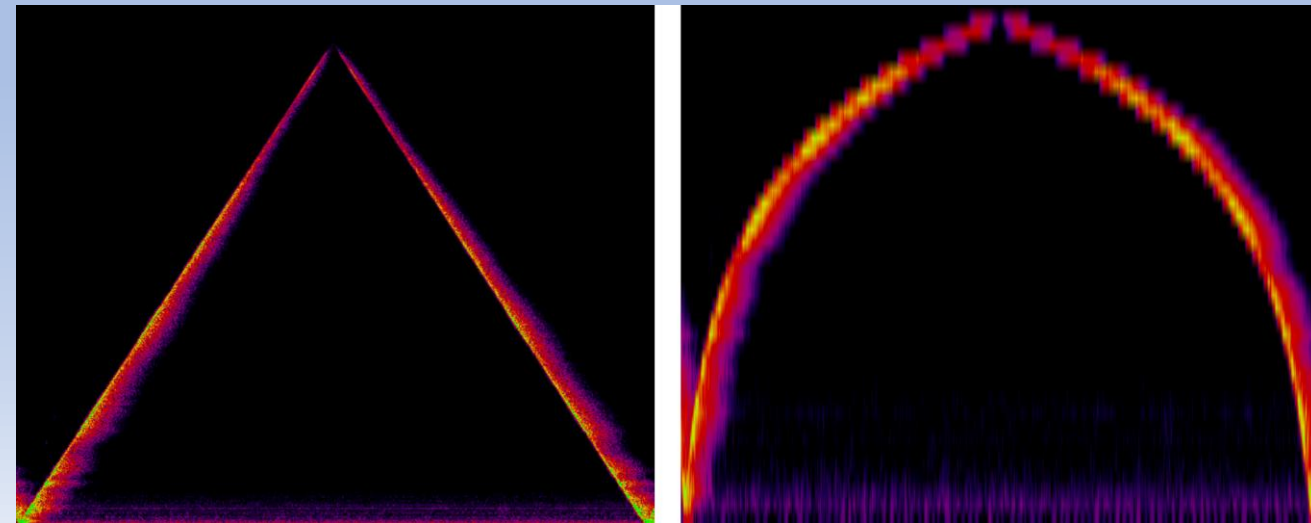
$$m(f) = 2595 \log_{10} \left( 1 + \frac{f}{700} \right) = 1127 \ln \left( 1 + \frac{f}{700} \right), \quad f \in \mathbb{R}^+. \quad (\text{Common conversion function from hertz to mel scale.})$$

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

Plot of frequencies in Mel versus Hertz scale:



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



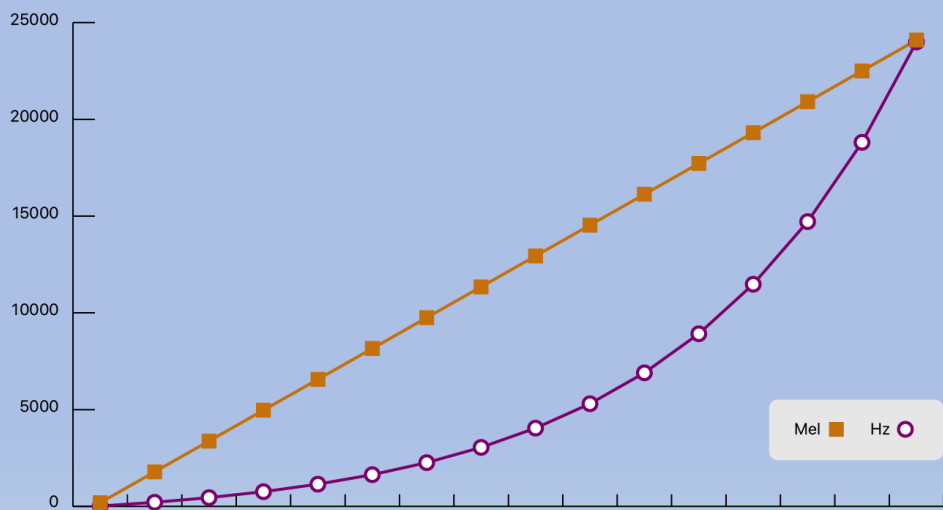
Spectrogram

Mel-Spectrogram

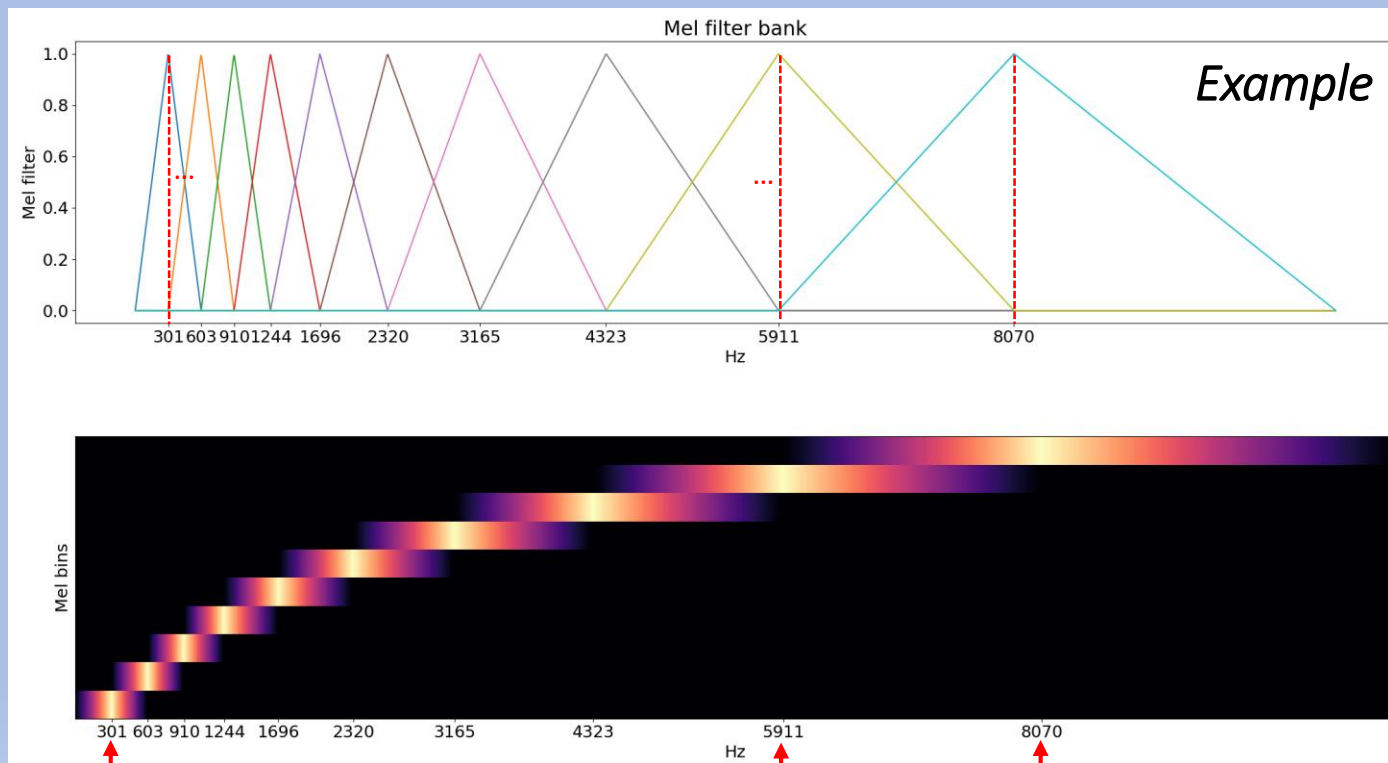
# Redefining the problem: Input and Frequencies

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

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



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



These central band frequencies are the choice for our constant frequencies  $f_m$  in the model

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



# 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:

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

➤ Main assumption differences:

- 1) Not constructing small frames with sums of 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:  
→  **$\alpha[n]$ ,  $\beta[n]$  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

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

➤ 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  $\theta$

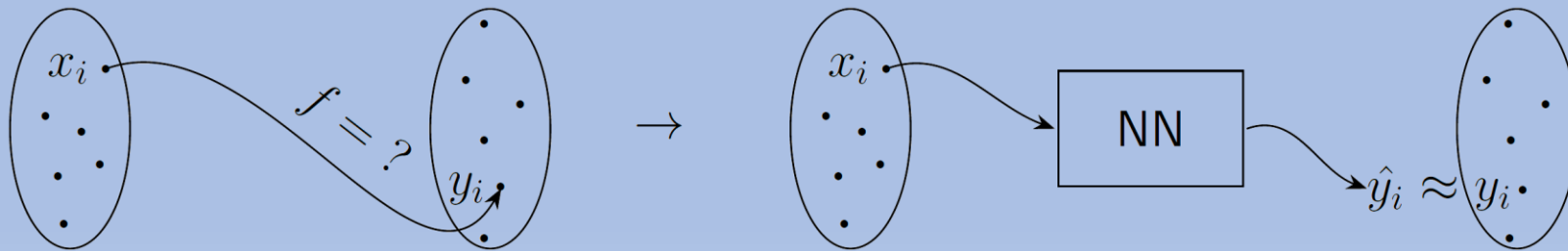
- Input = Mel-Spectrogram  $S$
- Output =  $\alpha, \beta$  that minimize  $\mathcal{L}$

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

# Optimization Approach: Artificial Neural Networks

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

- There exist many optimization methods depending on the problem's complexity/hardness, for instance, Linear, Convex, Message Passing, Belief Propagation, et al.
- The hardness of this problem, i.e., speech synthesis, has shown the need for **Artificial Neural Networks**



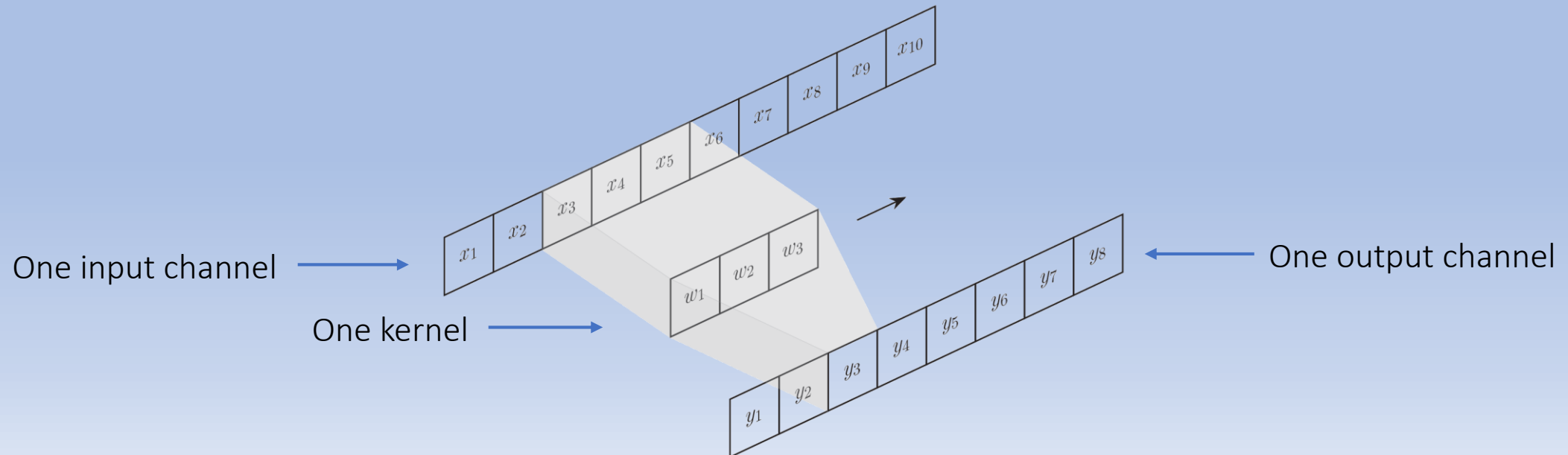
- Our function  $F$  will be an **artificial neural network** (or neural network, for short)
  - Neural networks are comprised of **layers**
  - Each layer is made of **weights**, or neurons (i.e., the trainable parameters  $\theta$ )
  - Each type of layer applies a different **operation** on its input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

# Neural Networks: 1D Convolutional Layer

- 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:

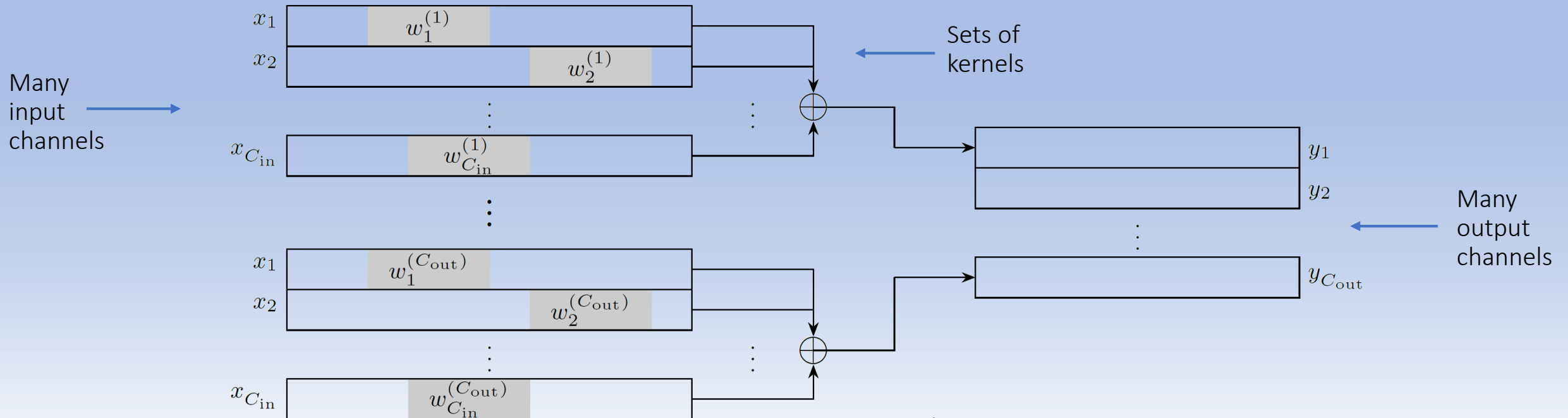


- Here, the **kernel** values  $w_k$  and the **bias** term  $b$  are **trainable parameters** ( $\in \theta$ ).

# Neural Networks: 1D Convolutional Layer

- 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**<sup>1</sup>:

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

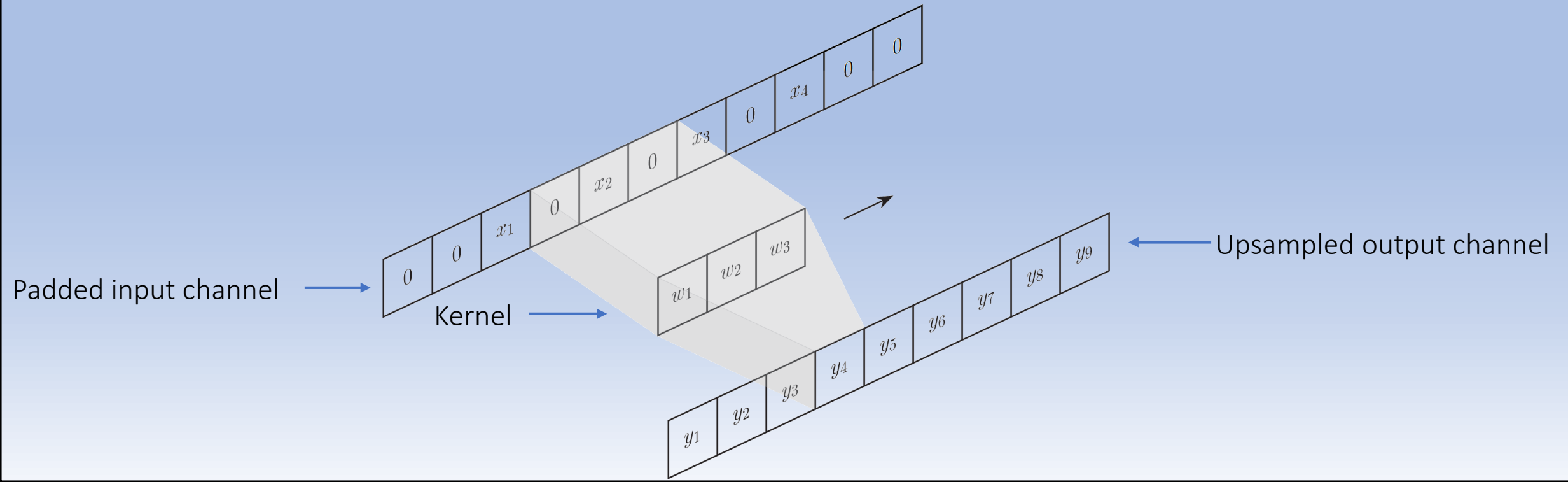


<sup>1</sup> The entire set of all kernels is known as the **filter** of the convolution.

# Neural Networks: Transposed 1D Convolutional Layer

- **Upsampling** along the **x-axis** (time axis in our case) is also necessary
  - Normally **convolves** the input sequence
  - After introducing **padding** among the **input's** consecutive values:

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



# Neural Networks: Optimizer

- The algorithm for **updating** the trainable parameters (e.g., convolution kernels)
  - Iteratively registering updates that minimize our defined loss function (finding 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 version of the above optimizer are actually used, but the core idea remains the same
  - Examples: Mini-Batch SGD, RMSprop, AdaGrad, Adam, et al.

# Neural Networks: Backpropagation

- **Backpropagation** is the algorithm for computing the gradient:
  - The outputs are given during the **forward pass**
  - Creates the function's **computational graph**
  - **Backward pass** calculates the gradient
  - This is done using the **chain rule**:

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

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

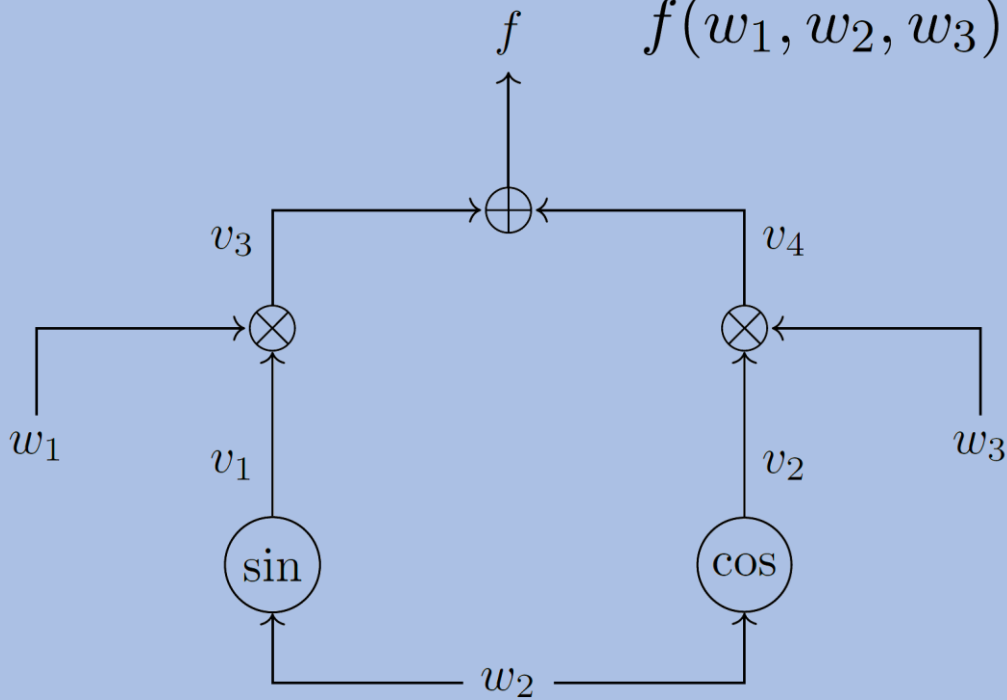
$$f(x) = g^L (W^L g^{L-1} (W^{L-1} \dots g^2 (W^2 g^1 (W^1 x) \dots))) .$$

<sup>1</sup> 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

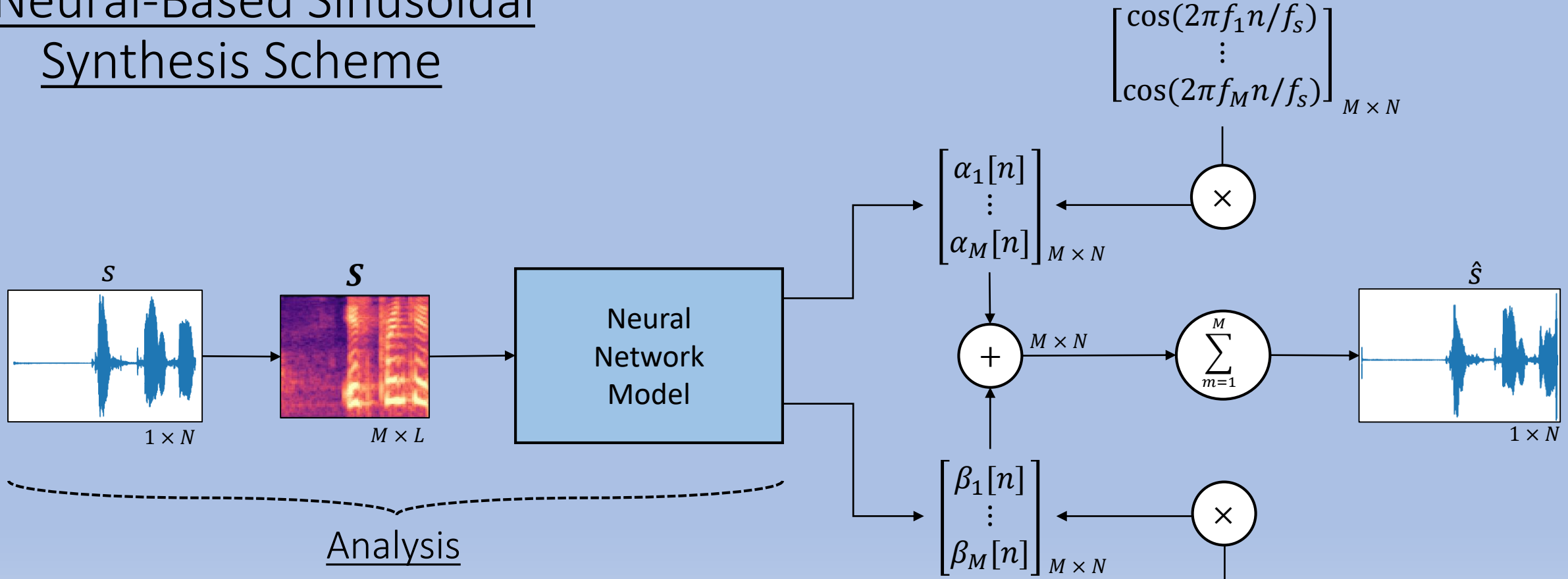
$$\bar{v}_3 = \frac{\partial f}{\partial v_3} = 1, \quad \bar{v}_1 = \frac{\partial f}{\partial v_3} \frac{\partial v_3}{\partial v_1} = \bar{v}_3 w_1 = w_1, \quad \bar{w}_1 = \frac{\partial f}{\partial v_3} \frac{\partial v_3}{\partial w_1} = \bar{v}_3 v_1 = v_1 = \sin(w_2).$$

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

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

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

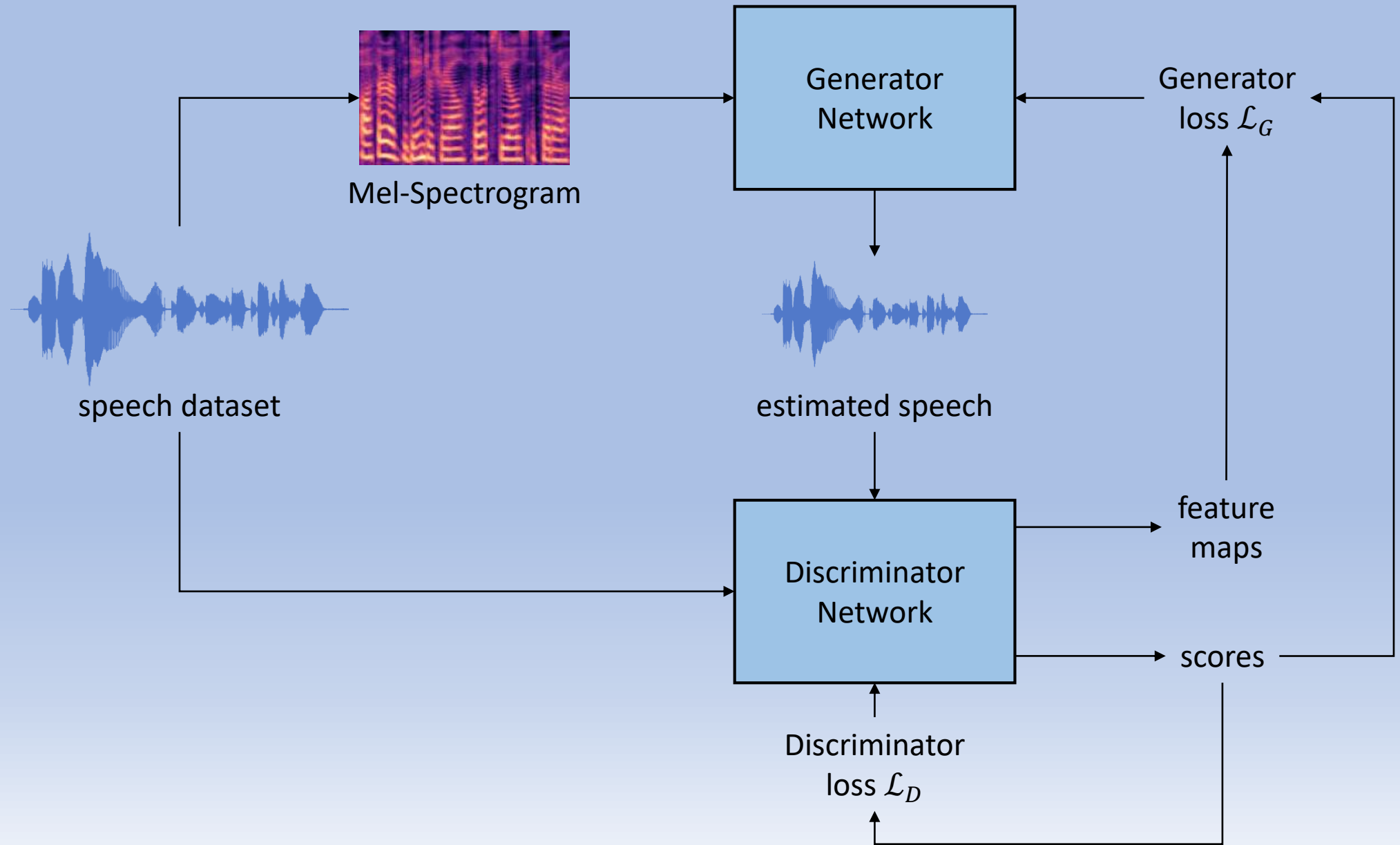
# A Neural-Based Sinusoidal Synthesis Scheme



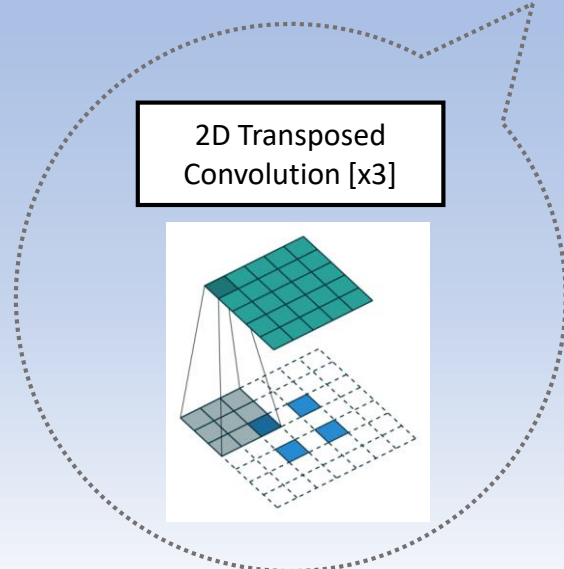
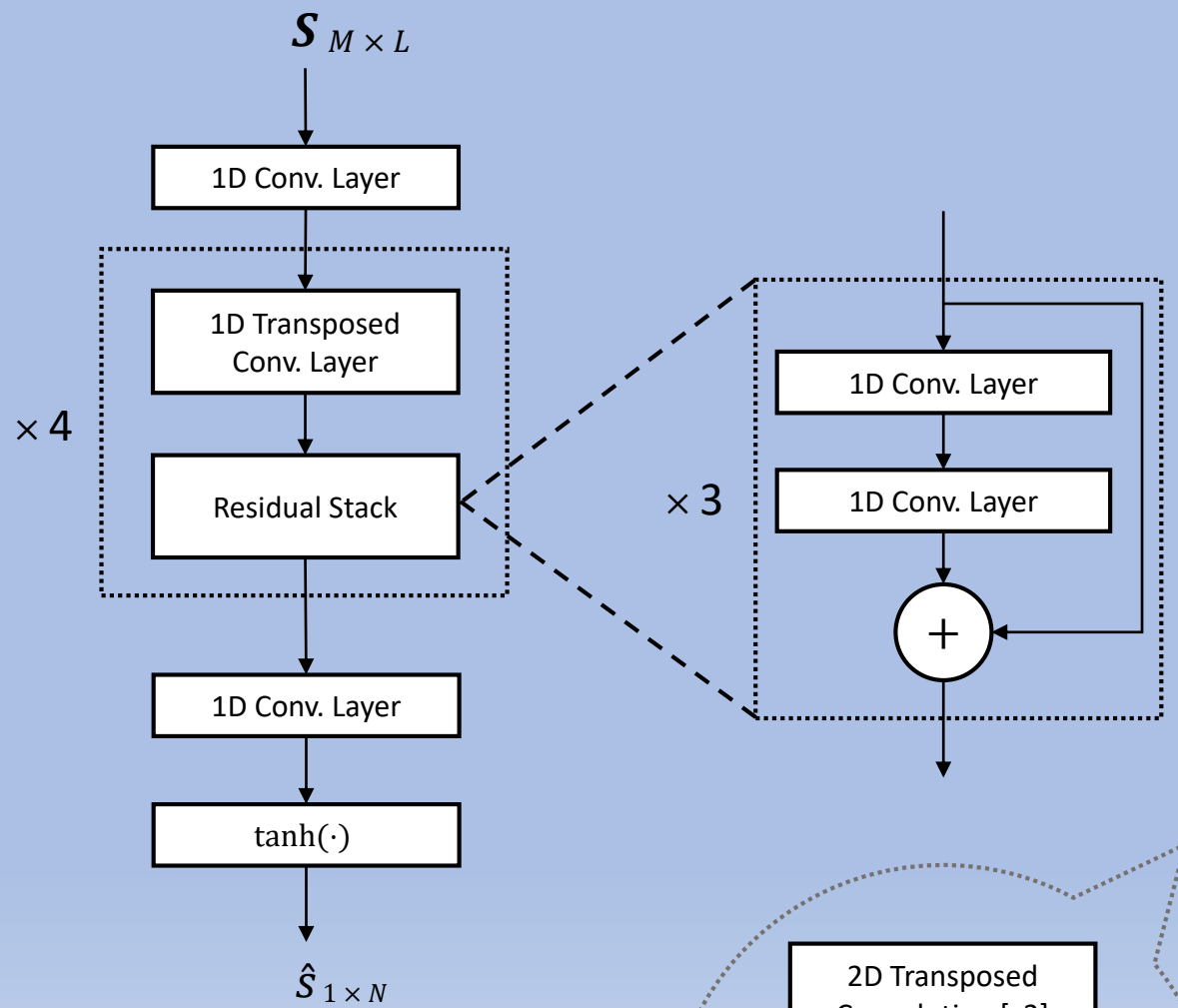
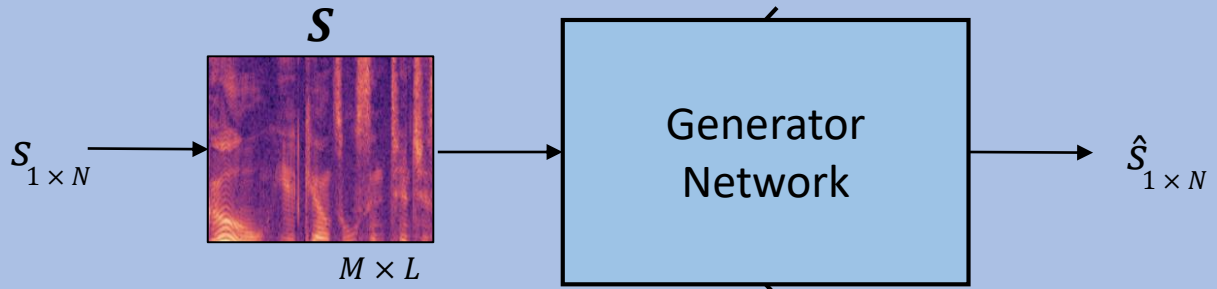
$s$  = Input speech wave  
 $N$  = Total number of samples  
 $S$  = Mel-Spectrogram of  $s$   
 $M$  = Total number of mel bins or sinusoids pairs  
 $\alpha_m[n]$  = Amplitudes of the  $m$ -th AM-FM cosine  
 $\beta_m[n]$  = Amplitudes of the  $m$ -th AM-FM sine  
 $f_m$  = Central frequency of the  $m$ -th mel band  
 $f_s$  = sampling rate  
 $\hat{s}$  = Predicted output speech wave

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

# Adversarial Training Scheme

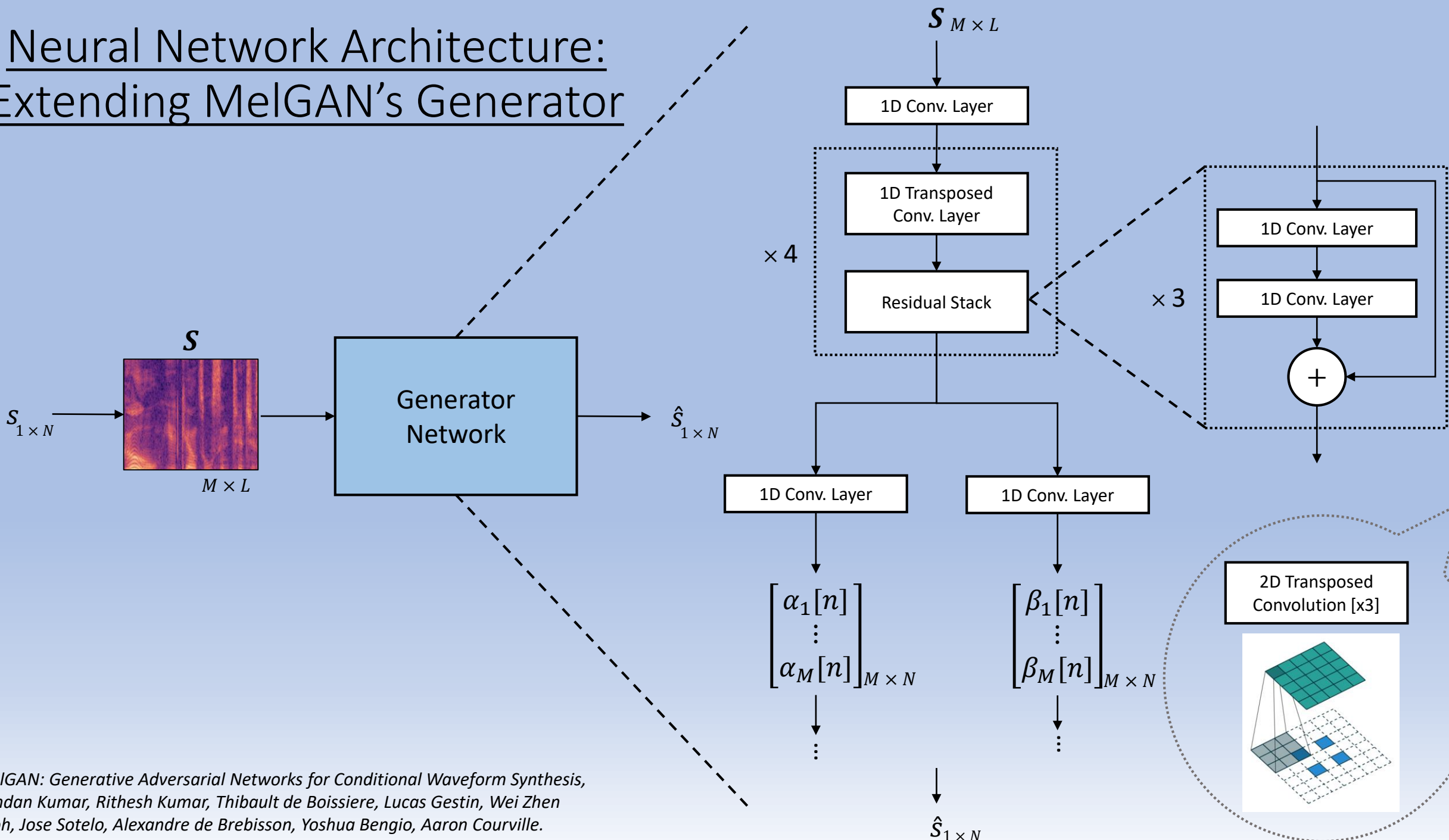


# Neural Network Architecture: Extending MelGAN's Generator



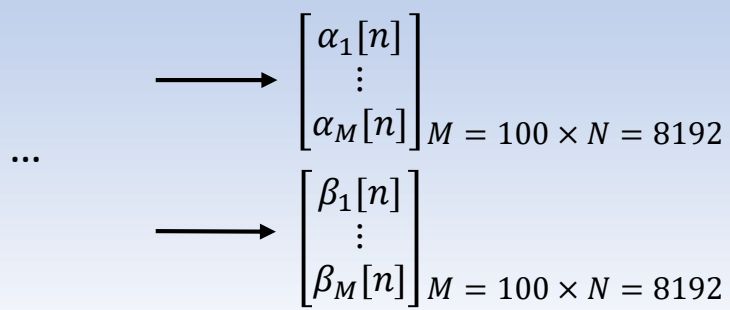
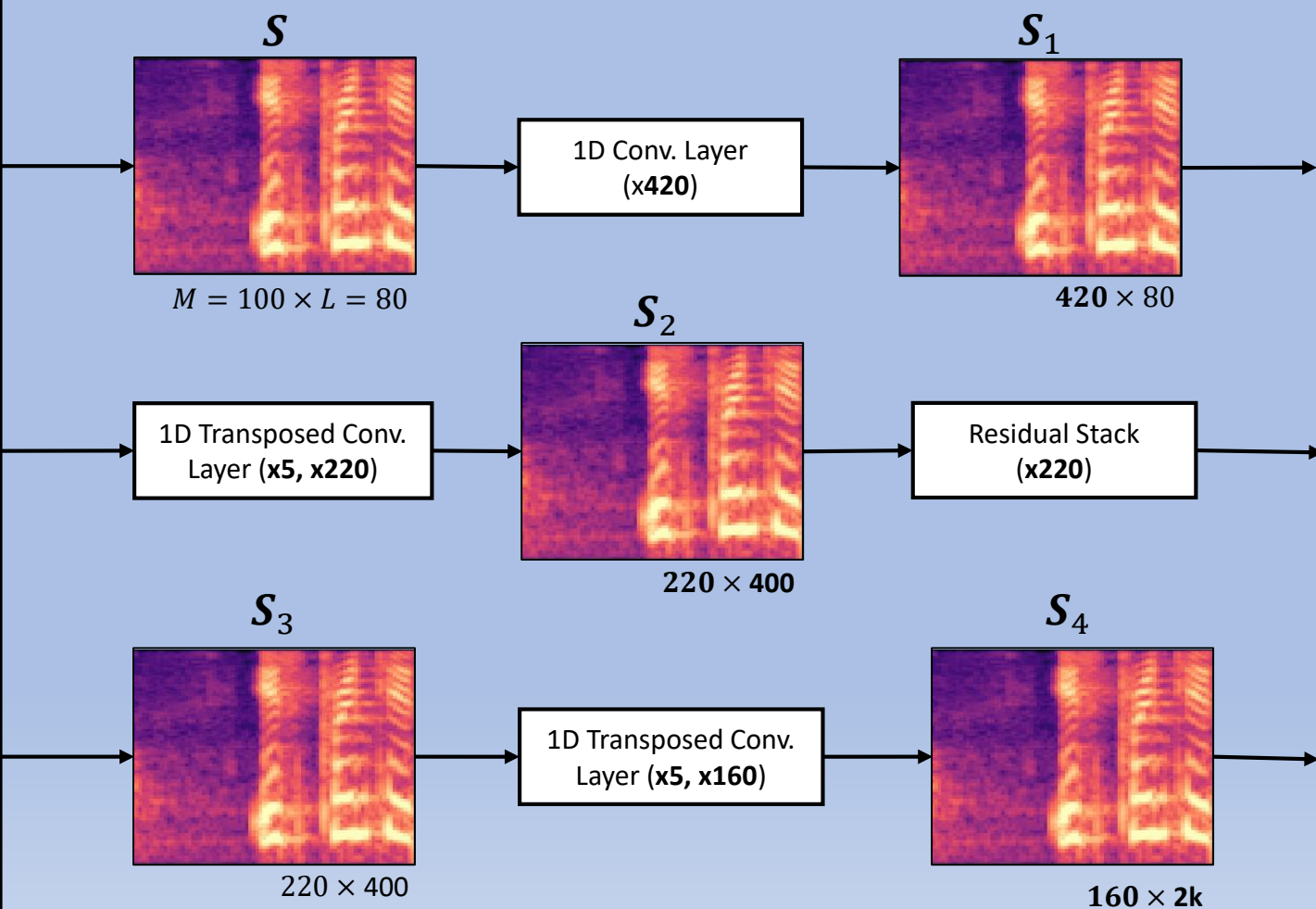
MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis,  
Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestein, Wei Zhen  
Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, Aaron Courville.

# Neural Network Architecture: Extending MelGAN's Generator

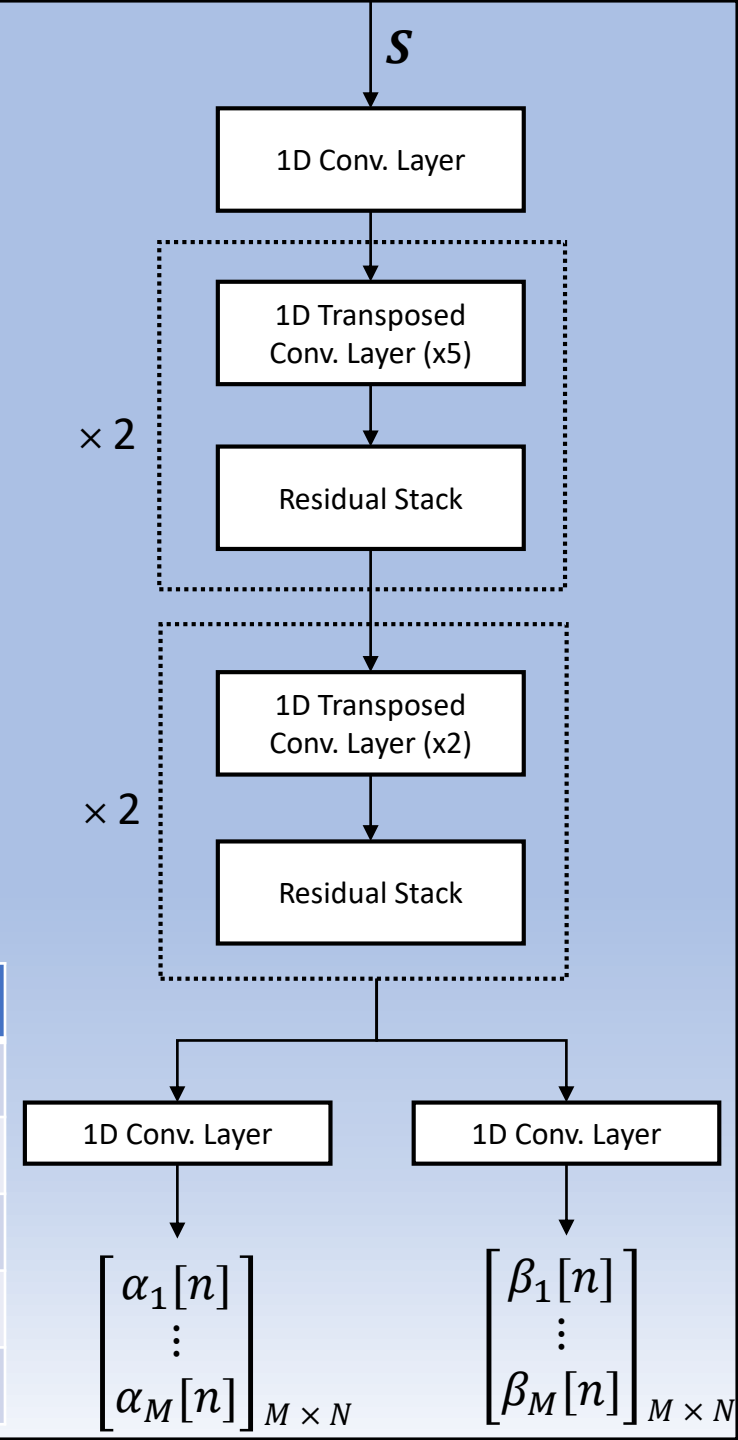


MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis, Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestein, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, Aaron Courville.

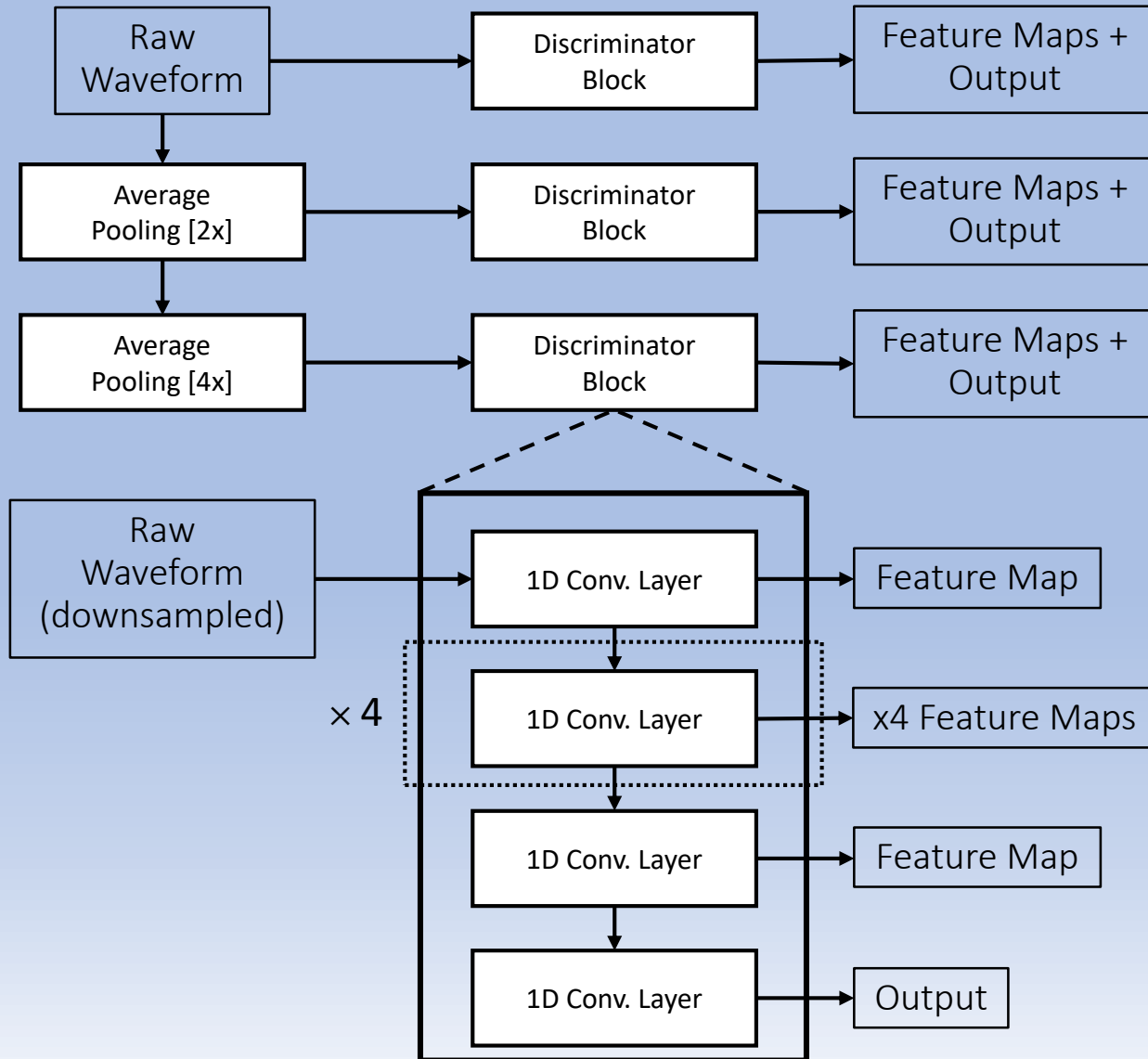
# Example Through the Network



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's training
- Disadvantages:
  - ✗ Space efficiency (~16.9 MM trainable parameters)
  - ✗ Training time overhead (× 2)

# 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

$s$  = A ground truth 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 Loss<sup>1</sup>:

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

### 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

<sup>1</sup> Similar to the perceptual loss in image processing.

# Spectral Loss Function

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

Spectral Convergence:

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

Logarithmic DSTFT Magnitude:

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

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

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

- 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  $\hat{s}$

$X[k]$  = The DFT of  $x[n]$

$K$  = Number of DSTFT frequencies

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

$\epsilon = 2, \lambda = 9$ , Regularization strength

# Spectral Loss Function

- Speech reveals different information when analyzed on different resolutions
  - **Wideband** (small analysis window) gives more information about the **formants, bursts, excitation pulses**, et al.
  - **Narrowband** (long analysis window) gives more information about the  **$F_0$ , harmonics, pitch**, et al.
  - We want our spectral loss function to capture all these characteristics

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  $\hat{s}$

$X[k]$  = The DFT of  $x[n]$

$K$  = Number of DSTFT frequencies

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

$\epsilon = 2, \lambda = 9$ , 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 approach

<b>MelGAN</b>		<b>SinGAN</b>	
<b>Generator</b> ~4.26 MM trainable parameters	<b>Discriminator</b> ~16.9 MM trainable parameters	<b>Generator</b> ~4.14 MM trainable parameters	<b>Discriminator</b> ~16.9 MM trainable parameters
~21.16 MM total trainable parameters		~21.04 MM total trainable parameters	

## Spectral loss function and no discriminator

<b>MelNoGAN</b>	<b>SinNoGAN</b>
~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) Experiment:

- 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>	<b>MelGAN</b>	<b>Sinusoidal</b>	<b>Ground Truth</b>
GAN	<b>3.53</b> ( $\pm 0.83$ )	3.27 ( $\pm 0.83$ )	4.90 ( $\pm 0.29$ )
No GAN	1.76 ( $\pm 0.80$ )	<b>2.01</b> ( $\pm 0.83$ )	4.90 ( $\pm 0.29$ )

## ➤ MOS Results:





















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

<i>Average Training Speed</i>	<b>MelGAN</b>	<b>Sinusoidal</b>
GAN	<b>236.67</b> ms/batch	515.10 ms/batch
No GAN	<b>105.45</b> ms/batch	258.78 ms/batch
<i>Average Inference Speed</i>	<b>2.34</b> 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

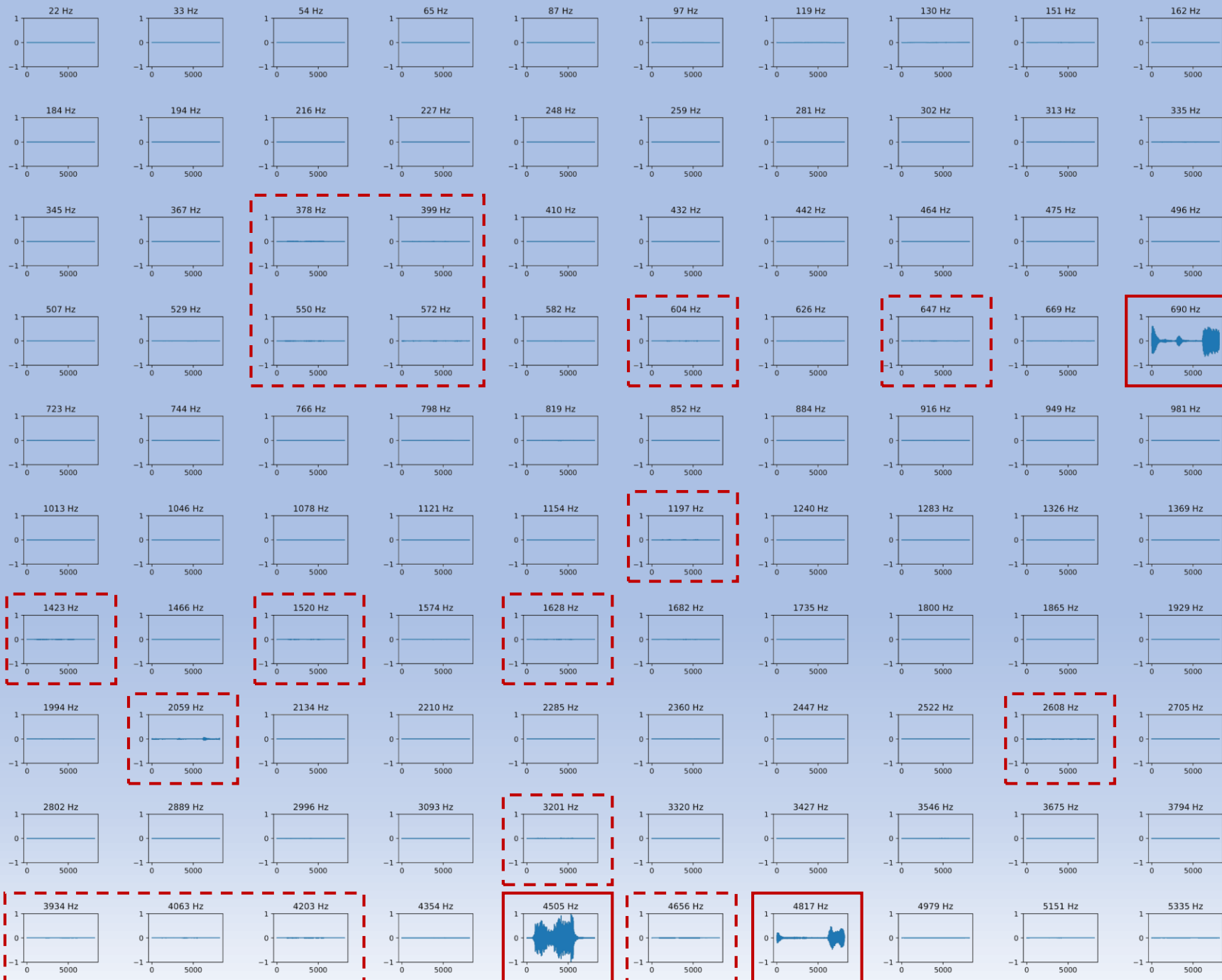
# Audio Samples

Ground Truth	MelGAN	SinGAN	MelNoGAN	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.

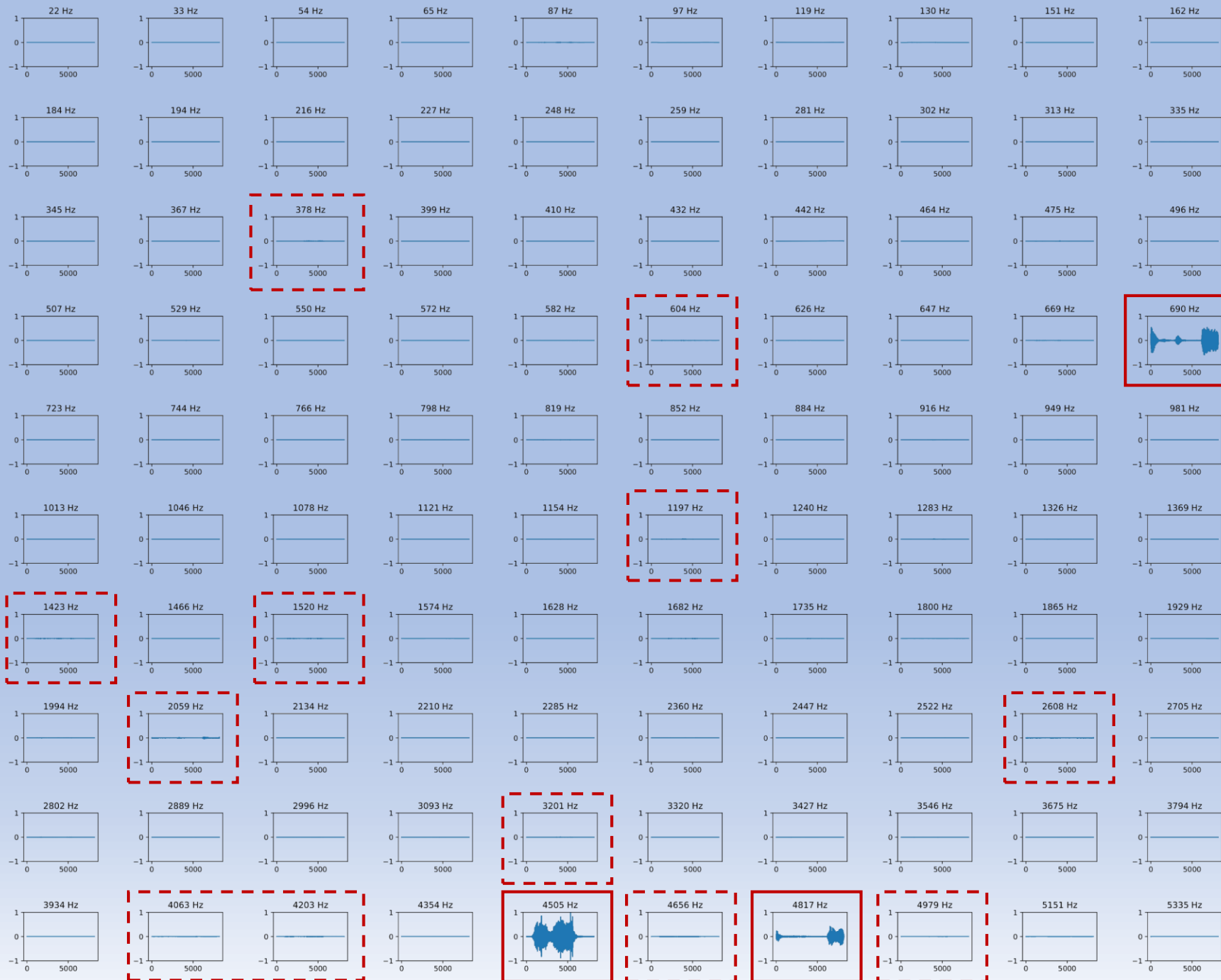
# Amplitude Modulators



➤ SinGAN's  $\alpha_m[n]$  on random speech inputs:

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

# Amplitude Modulators

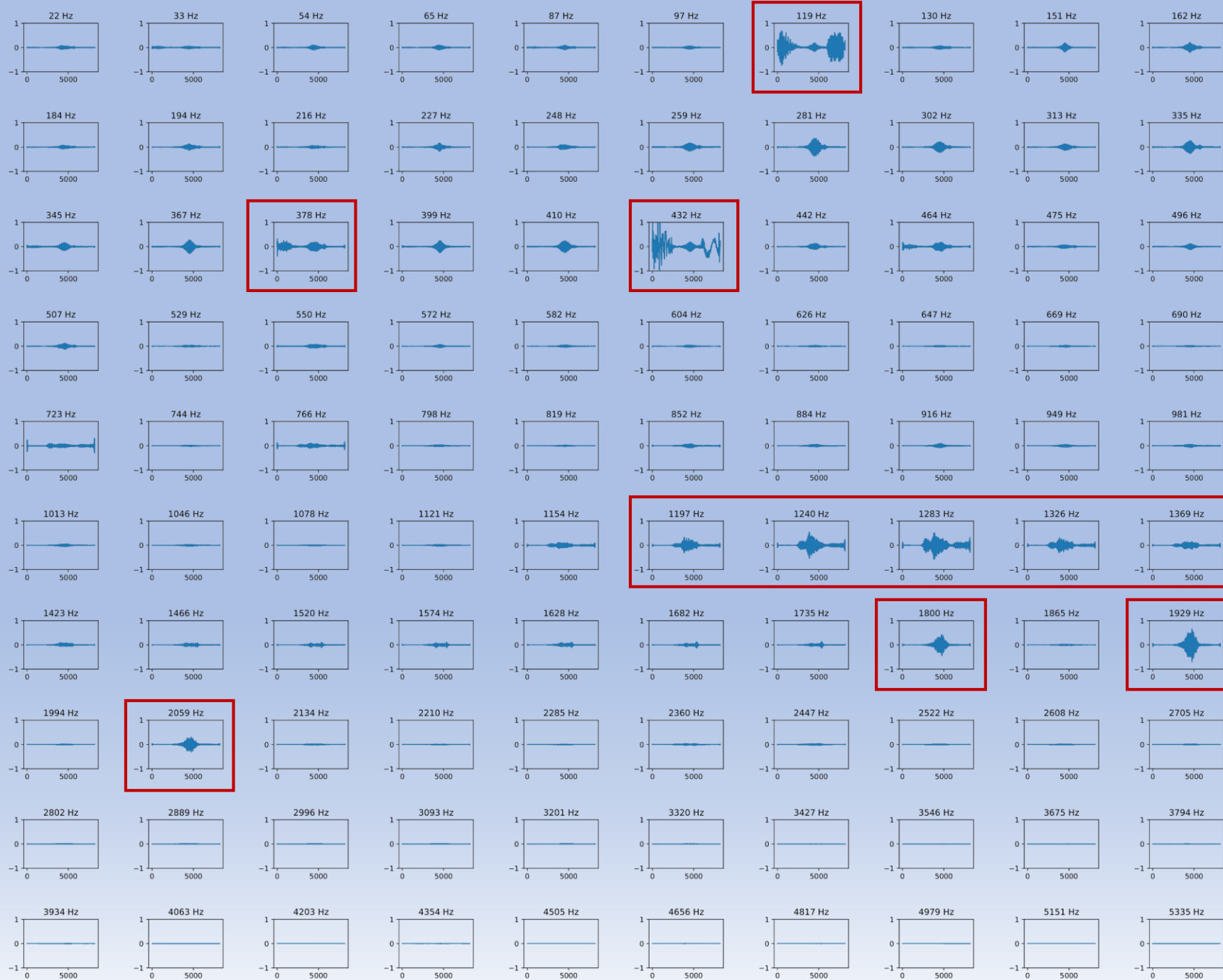


➤ SinGAN's  $\beta_m[n]$  on random speech inputs:

- 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



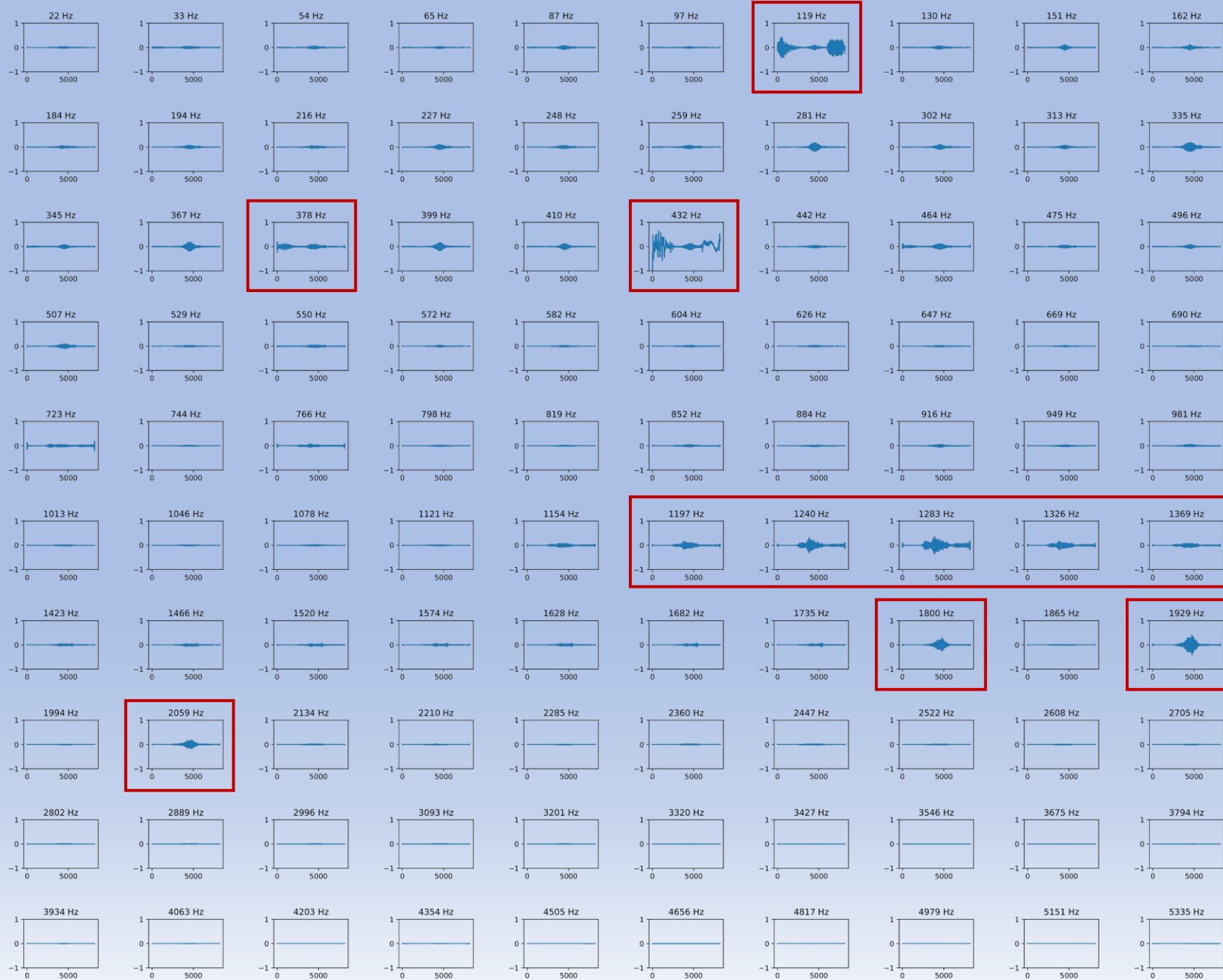
# Amplitude Modulators



➤ SinNoGAN's  $\alpha_m[n]$  on random speech inputs:

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

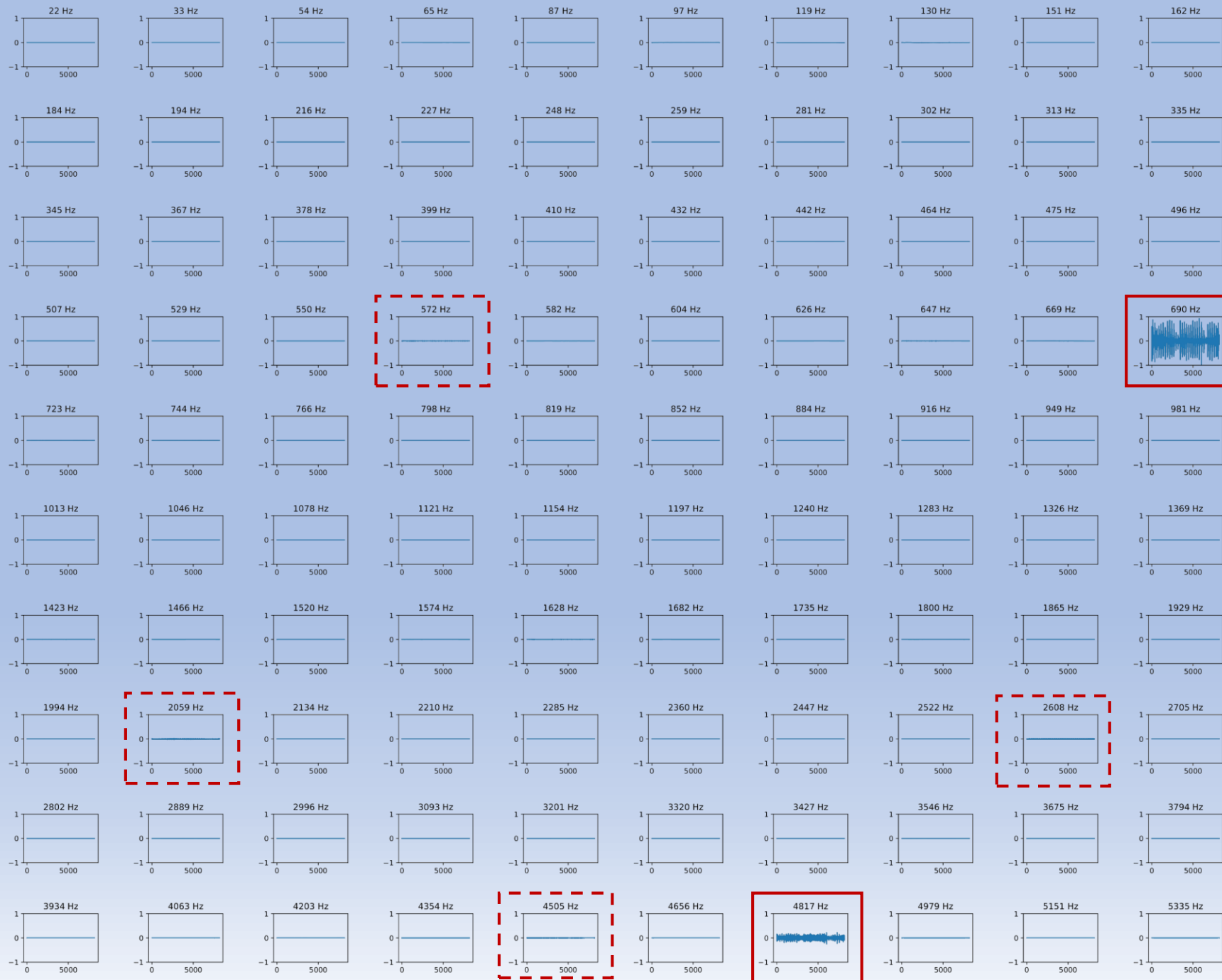
# Amplitude Modulators



➤ SinNoGAN's  $\beta_m[n]$  on random speech inputs:

- Same frequencies used as in the previous figure
- Slightly lower in amplitude than  $\alpha_m[n]$
- Betas modulators are omitted for the following figures

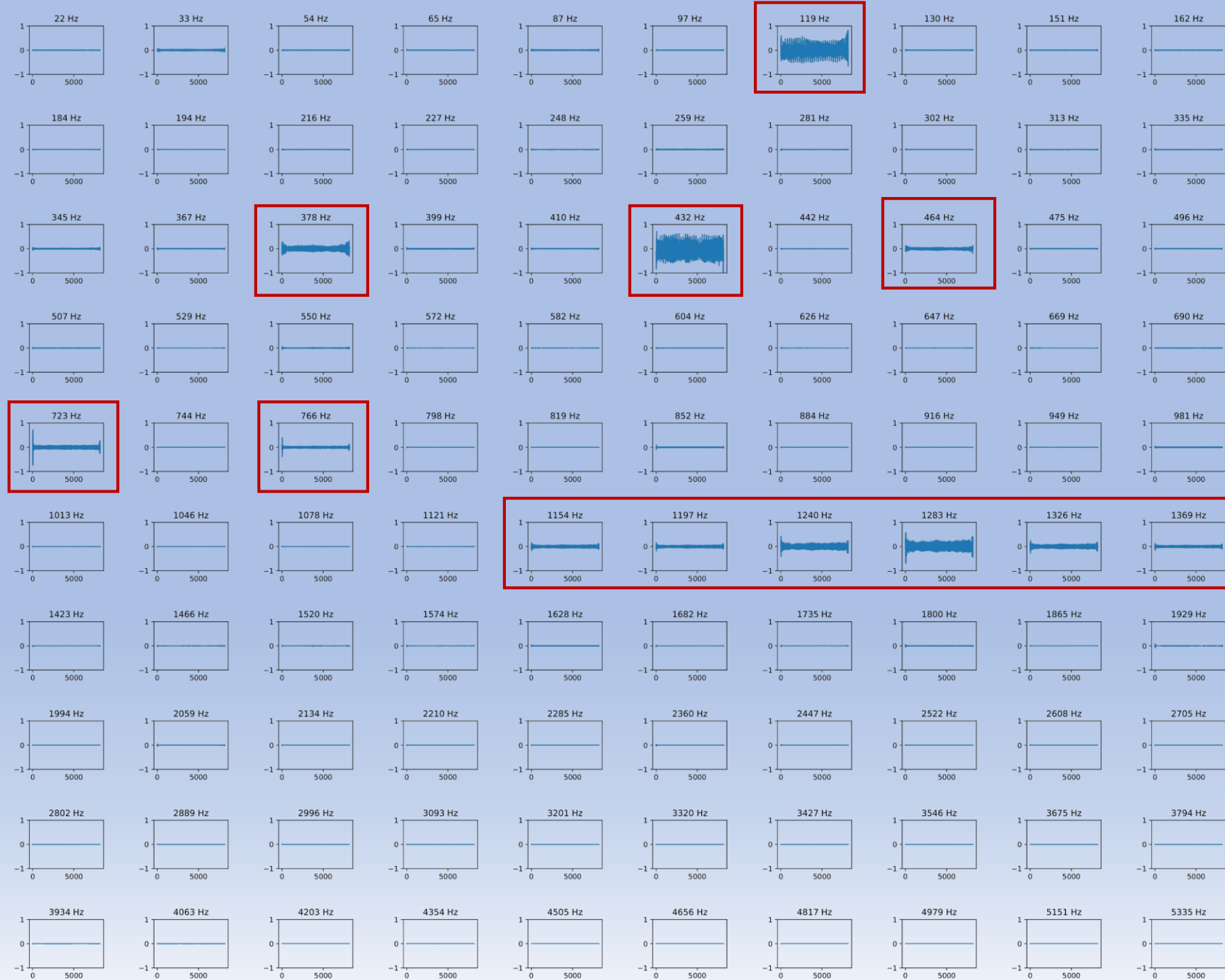
# Amplitude Modulators



➤ SinGAN's  $\alpha_m[n]$  on a plain /a/ vowel signal:

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

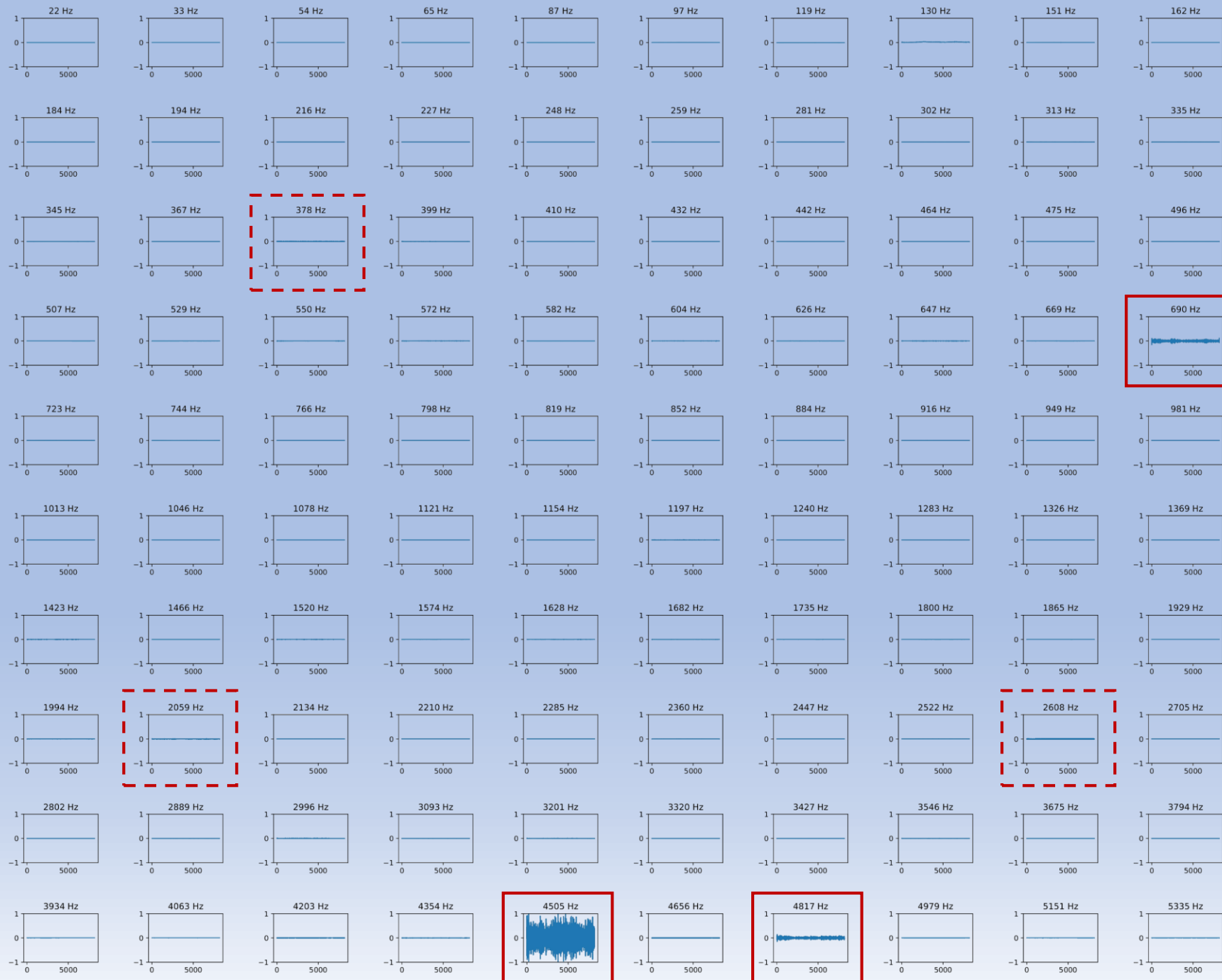
# Amplitude Modulators



➤ SinNoGAN's  $\alpha_m[n]$  on a plain /a/ vowel signal:

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

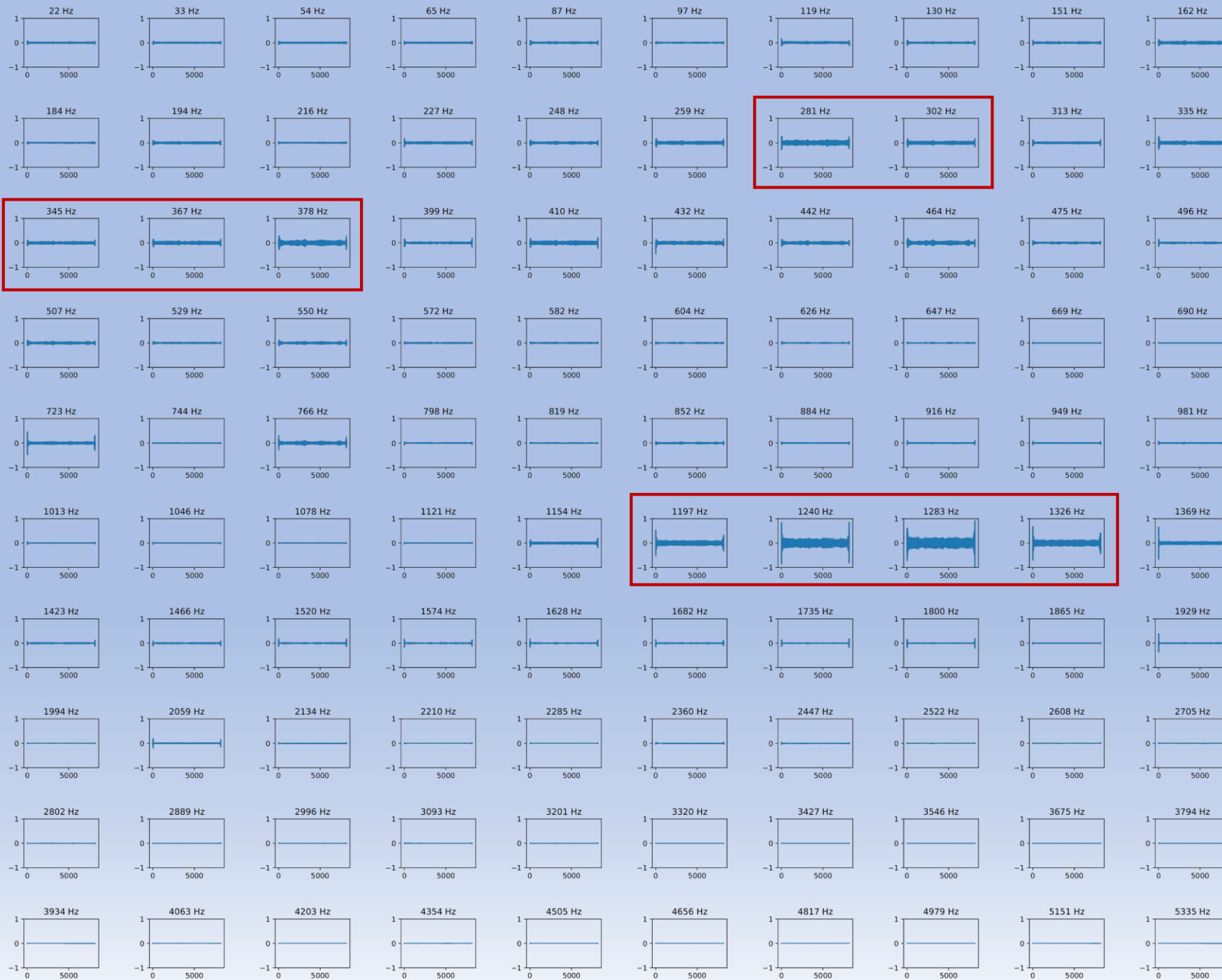
# Amplitude Modulators



➤ SinGAN's  $\alpha_m[n]$  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  $\alpha_m[n]$  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/

# Future Work

## ➤ Number of sinusoids utilized:

- ✘ SinGAN does not take advantage of the sinusoidal approach to a satisfactory degree
- ✓ 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

## ➤ Small-amplitude sinusoids:

- Especially in the non-GAN model, many sinusoids were observed to have very little amplitude
- ? Those may contribute to noisy artifacts in the output
  - 💡 Discard small-amplitude sinusoids and compare the resulting quality
  - 💡 Improvement  $\Rightarrow$  automatically better model
  - 💡 Improvement/Same  $\Rightarrow$  retrain with the noise-inducing sinusoids excluded (parameters--, speed++, quality++)
  - 💡 Deterioration  $\Rightarrow$  models indeed use more sinusoids that we previously thought of

# Future Work

## ➤ Loss function in non-GAN approach:

- ✘ Cannot match the discriminator in terms of quality – misses some speech features for better quality
- 💡 Scale by a hyperparameter which subsequent loss components are more important
- 💡 Add features extracted from the discrete-time domain
- 💡 Approach analytically what features the discriminator learns (future work in neural network research in general)

## ➤ Spectrogram Input:

- Neural networks learn the phase information “from scratch” since spectrograms do not contain phase information
- 💡 Including some information about the phase in the input or the loss function might yield improvements



## Conclusion

- Neural-based sinusoidal representations of speech are feasible:
  - ✓ New solution 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 improves quality further
  - ✓ Plenty of ideas for further quality and speed 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