Information Retrieval

Expanding classic IR

- Recap
- Biological sound signals
- Frequency Transformations
- Classic Speech Recognition and IR

Development:
Moshe Friedman

Credits:
Yoav Goldberg, Ido Dagan, Reut Tsarfaty , Moshe Koppel, Wei Song, David Bamman, Ed Grefenstette, Chris Manning, Tsvi Kuflik, Hinrich Schütze, Christina Lioma, Wikipedia and more

Information Retrieval - administration

Moshe Friedman

Email: moshefr.teach@gmail.com

Reception time: before/after lesson/zoom with coordination

OCR

OCR - Recap

Classic OCR (before deep learning) is an Image Processing problem, not a Learning problem.

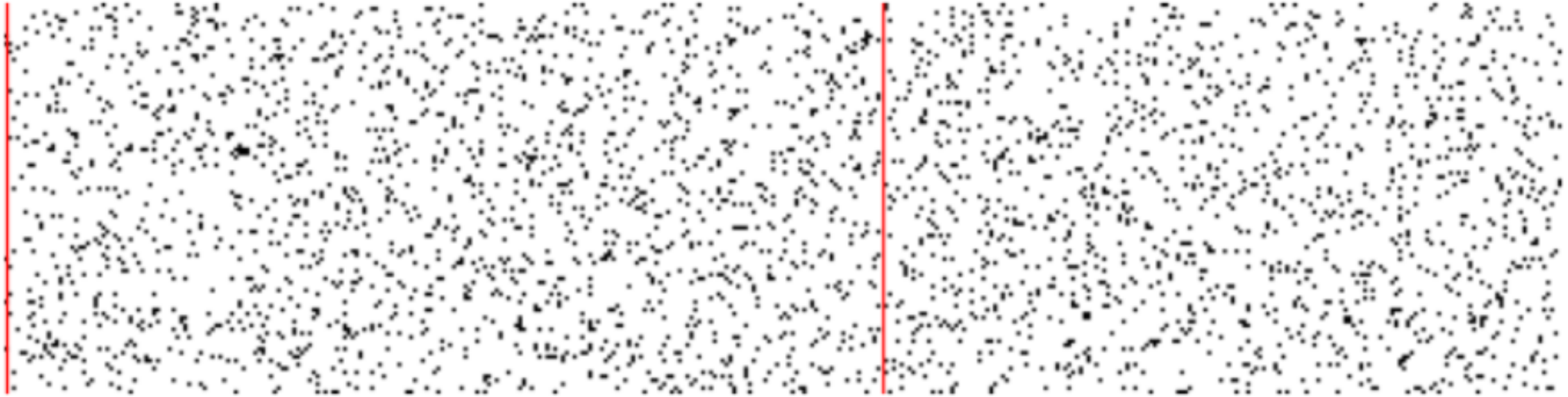
- It relies on geometry, projection profiles, and template matching.
- The IR systems then had to handle the inevitable typos ("teh" instead of "the", "1" instead of "l") using N-Gram (Fuzzy) Indexing.

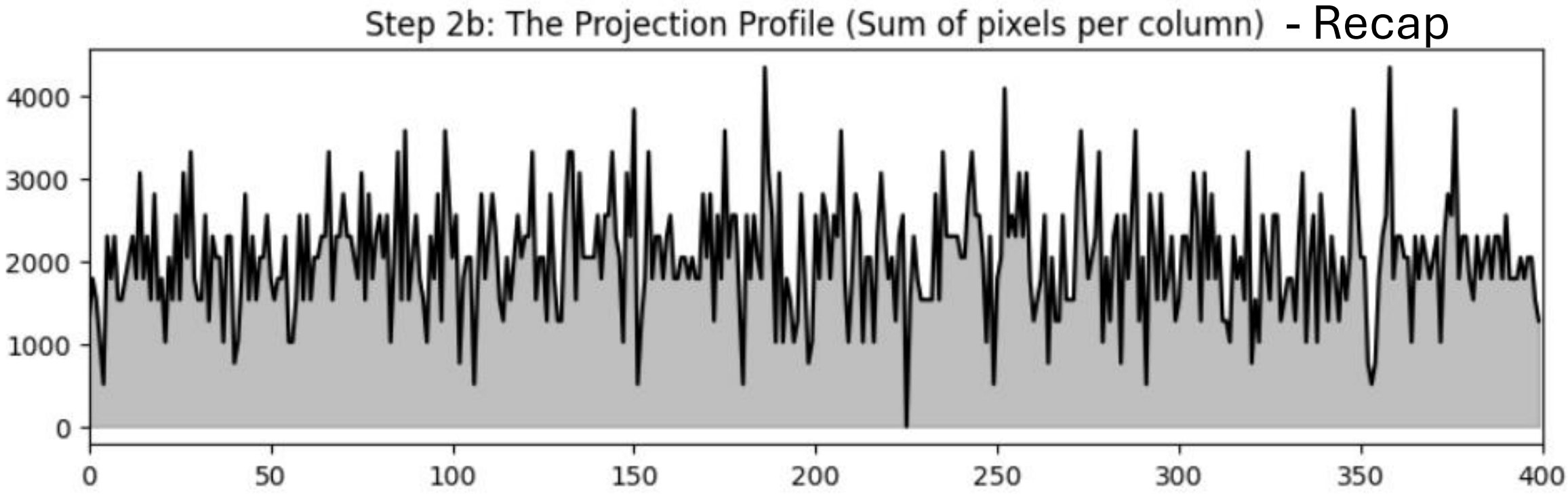
The Architecture - Recap

- **Image Pre-processing**: Binarization & Deskewing.
- **Segmentation**: Using Projection Profiles (histograms of pixel density) to cut out lines and characters.
- **Recognition**: Template Matching (Correlation) or Feature Engineering (HOG/Zoning).
- **Indexing**: N-Gram decomposition to make the search engine resilient to OCR errors.

Step 1: The 'Scanned' Noisy Document - Recap

Step 2a: Character Segmentation via Vertical Projection - Recap




Step 2b: The Projection Profile (Sum of pixels per column)
4000
3000
2000
1000
0
0
50
100
150
200
250
300
350
400

- Recap

Step 4: The IR Solution (N-Gram Indexing) - Recap

This is the critical IR component.

If the user searches for "Reference", but the database contains "Referenqe", a standard Exact Match fails.

Classic systems used Character N-Grams (q-grams).

Concept:

- **Word**: REFERENCE → `REF`, `EFE`, `FER`, `ERE`, `REN`, `ENC`, `NCE`
- **OCR**: REFERENQE → `REF`, `EFE`, `FER`, `ERE`, `REN`, `ENQ`, `NQE`
- **Overlap**: 5 out of 7 n-grams match. High similarity!

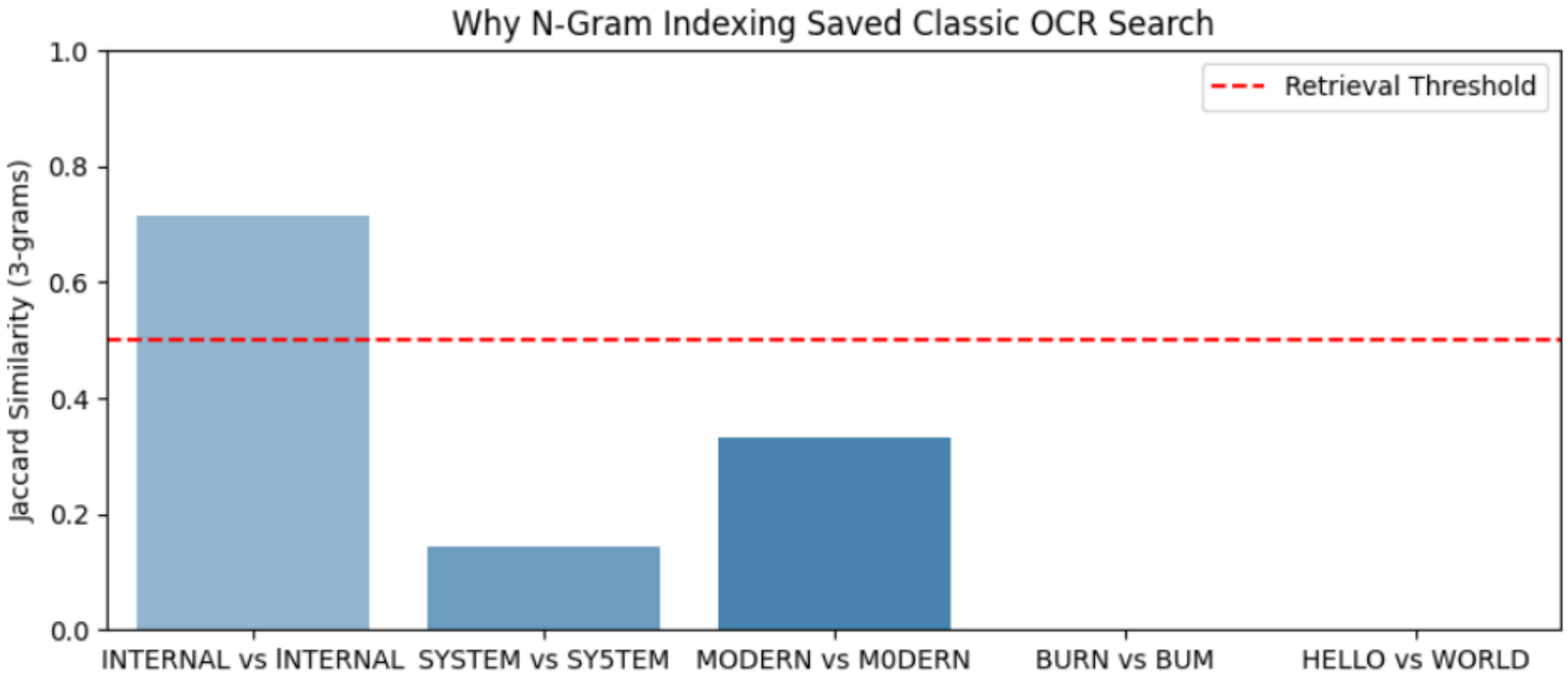
- Query n-grams: ['efe', 'enc', 'ere', 'fer', 'nce', 'ref', 'ren']
- Doc n-grams: ['efe', 'enq', 'ere', 'fer', 'nqe', 'ref', 'ren']
- Matching n-grams: ['efe', 'ere', 'fer', 'ref', 'ren']
- Jaccard Similarity Score: 0.56

→RESULT: Document Missed.

Step 5: Visualizing the "Robustness" Matrix- Recap

To verify this approach works for various **classic OCR errors**,
we can visualize the similarity matrix between correct words and common OCR misinterpretations.

- S ↔ 5
- l ↔ 1
- rn ↔ m



Summary of the Flow - Recap

- **Input**: Noisy image.
- **Segmentation**: Used Horizontal/Vertical Projections to chop the image into bits.
- **Recognition**: Used Template Matching (pixel overlap).
  This was brittle and produced "Dirty Text" (e.g., "SY5TEM").
- **IR Adaptation**: The search engine didn't index whole words.
  It indexed rolling 3-character windows (trigrams).
- **Result**: Even if rn became m, the other trigrams (bur, urn) would overlap enough to retrieve the document "Burn".

The Classic Computer Vision Flow (before CNNs) - Recap

Classic Computer vision is about "*Hand-Crafted Features*."

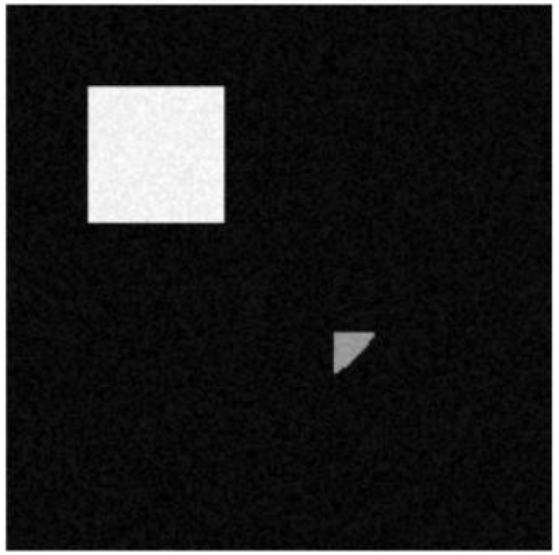
We didn't learn features; we mathematically defined them.

The Pipeline:

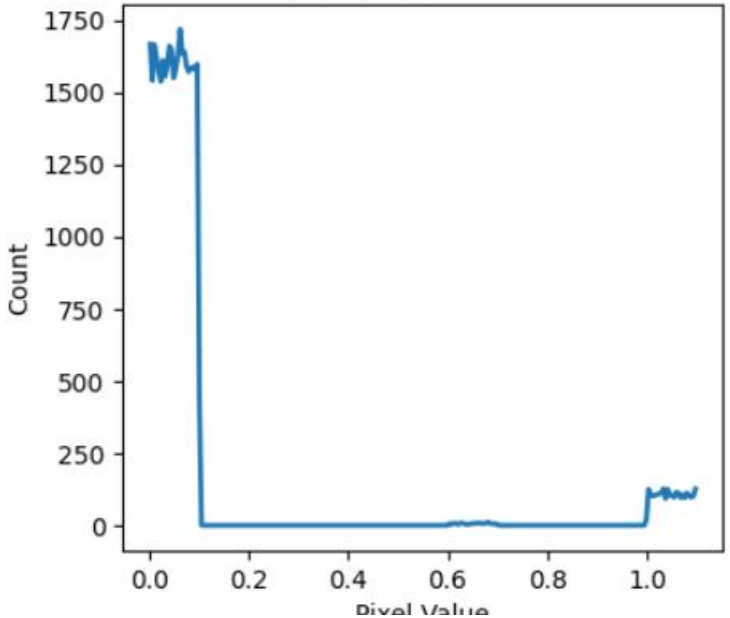
- **Preprocessing**: Grayscale/Normalization.
- **Global Analysis**: Pixel Intensity Histograms.
- **Edge Detection**: Finding structural boundaries (Canny/Sobel).

Feature Extraction: HOG (Histogram of Oriented Gradients).

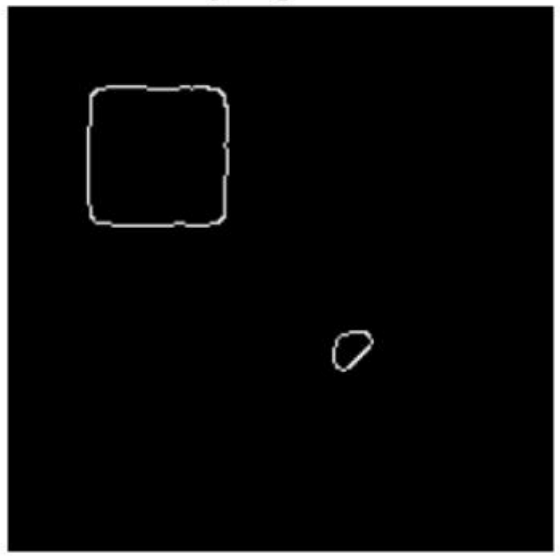
1. Raw Input (Noisy)



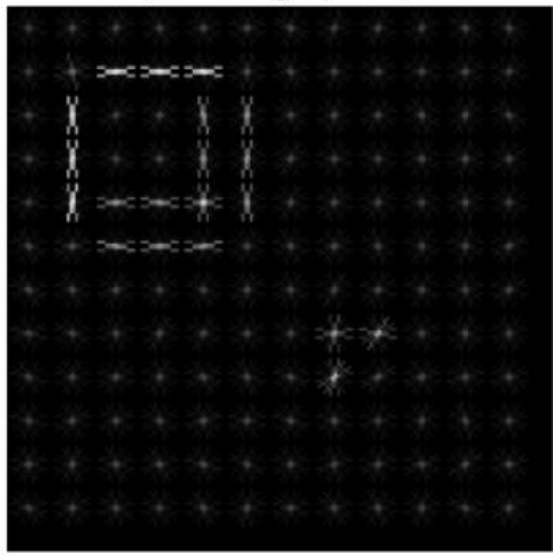
2. Intensity Histogram



3. Canny Edge Detection



4. HOG Features
(The 'Fingerprint')

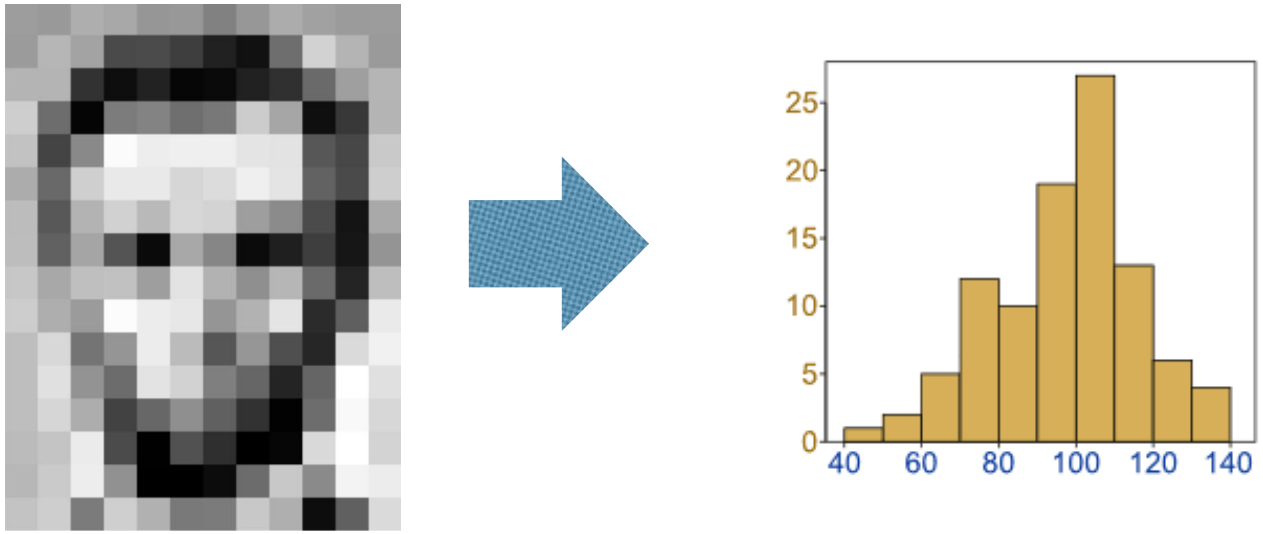


Explanation of Steps: - Recap

- **Histogram**: You see three peaks:
  The black background (near 0), the gray circle (near 0.6), and the white square (near 1.0).
  This is simple, but ignores location.
- **Edges**: The Canny filter calculates gradients.
  Note how it successfully ignores the random noise but highlights the borders of the shapes.
- **HOG**: Look closely at the last plot. You see "star-like" lines.
  These represent the dominant direction of the edges in that square.
  This grid of directions is the feature vector used to train classifiers (like SVMs) to recognize objects.

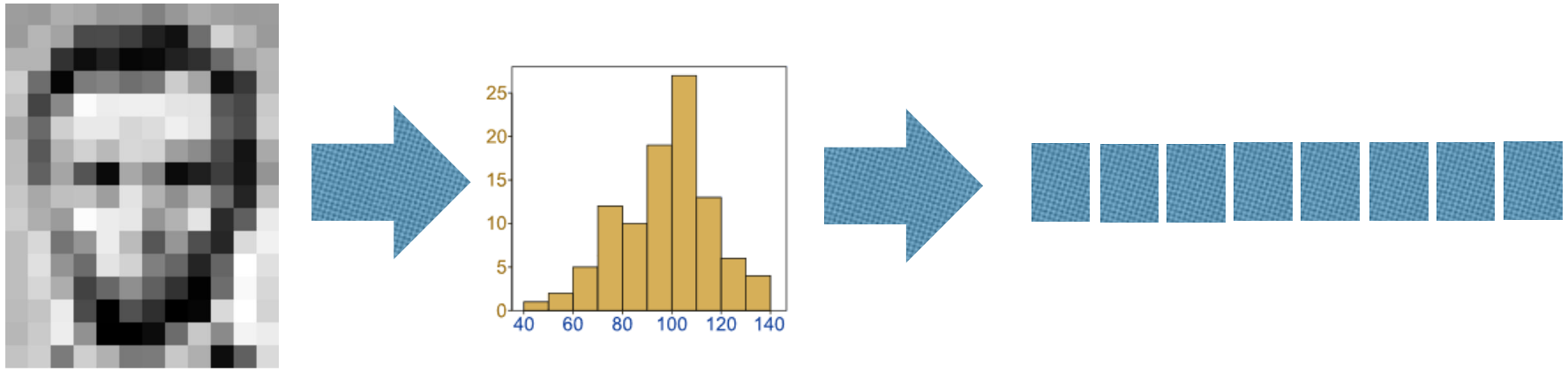
From image to features - Recap

Image Descriptors, e.g., Histograms



From image to features - Recap

Image Descriptors, e.g., Histograms



Traditional approach - Recap

Manually Identify key features in each category.

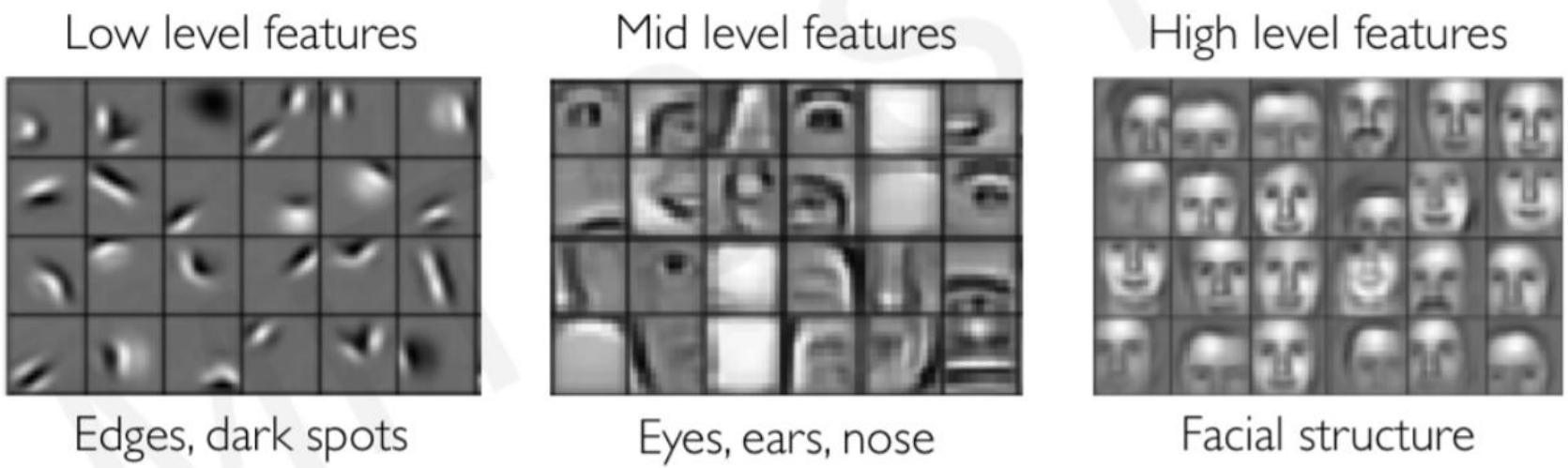
Nose
Eyes
Mouth

Wheels
License Plate
Headlights

Train ML classifier on these features.

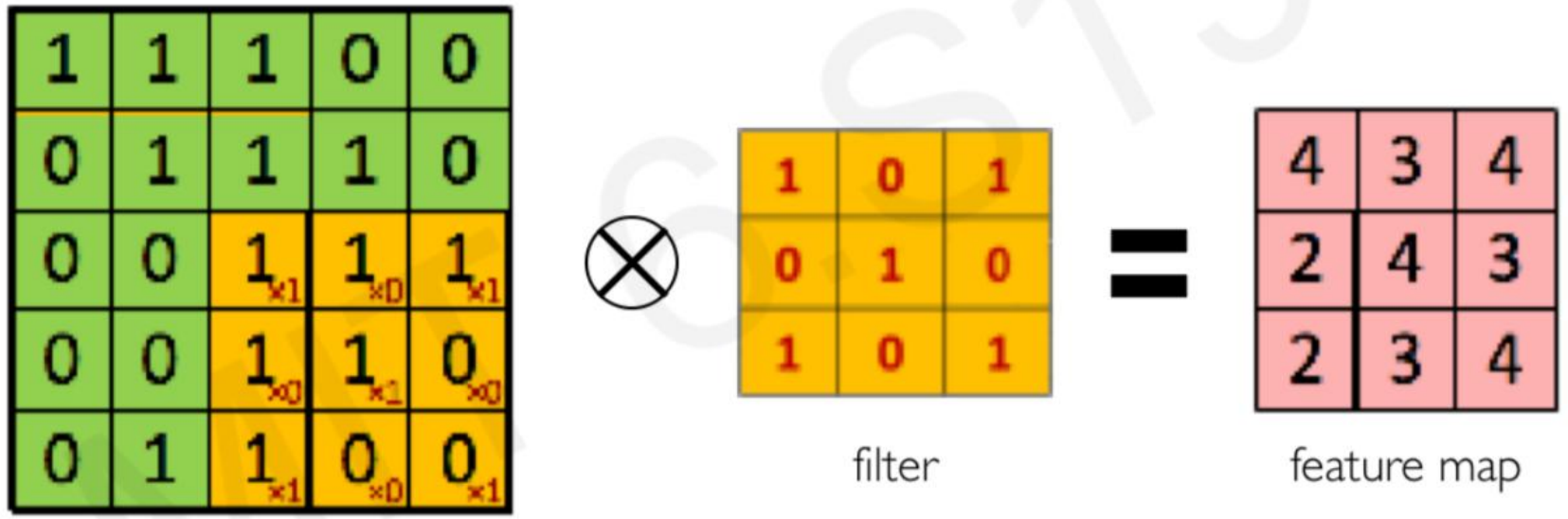
Automatic feature extraction - Recap

Can we learn hierarchy of features directly from the data?



convolutions - Recap

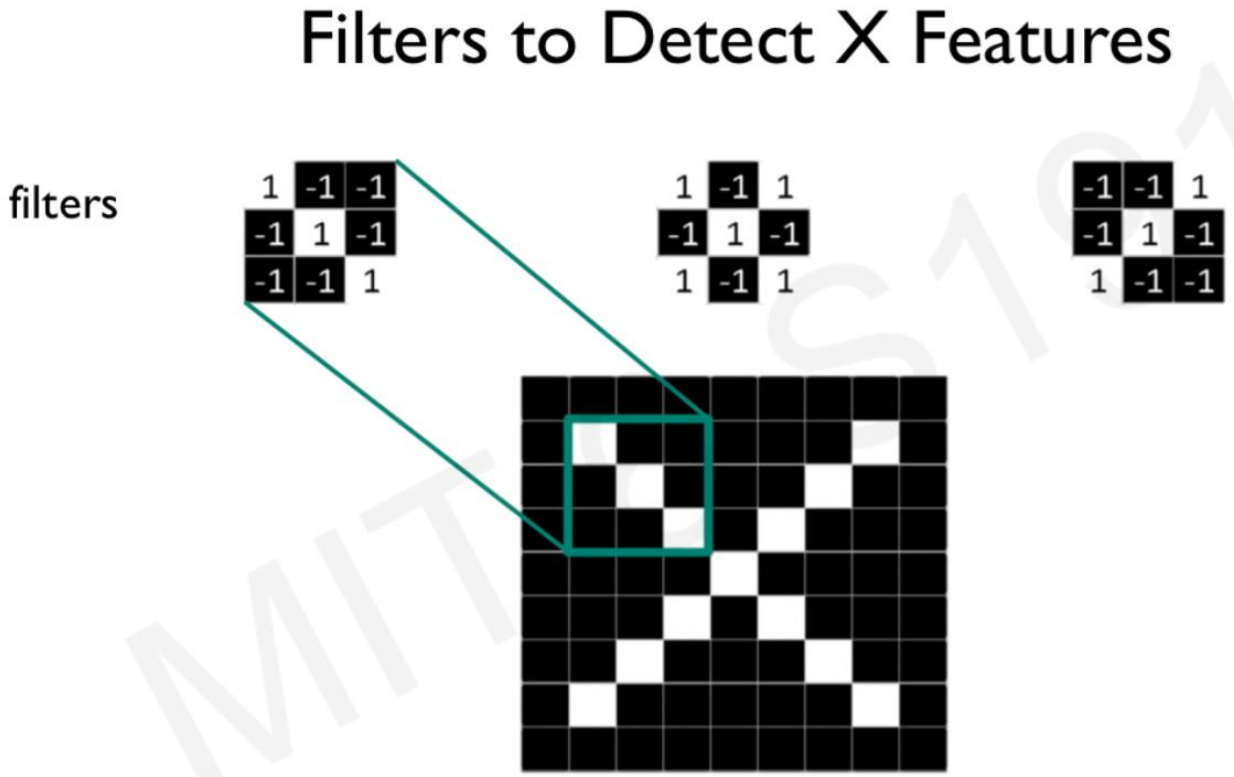
We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



Convolutions produce feature maps - Recap

Original | Sharpen | Edge Detect | "Strong" Edge Detect

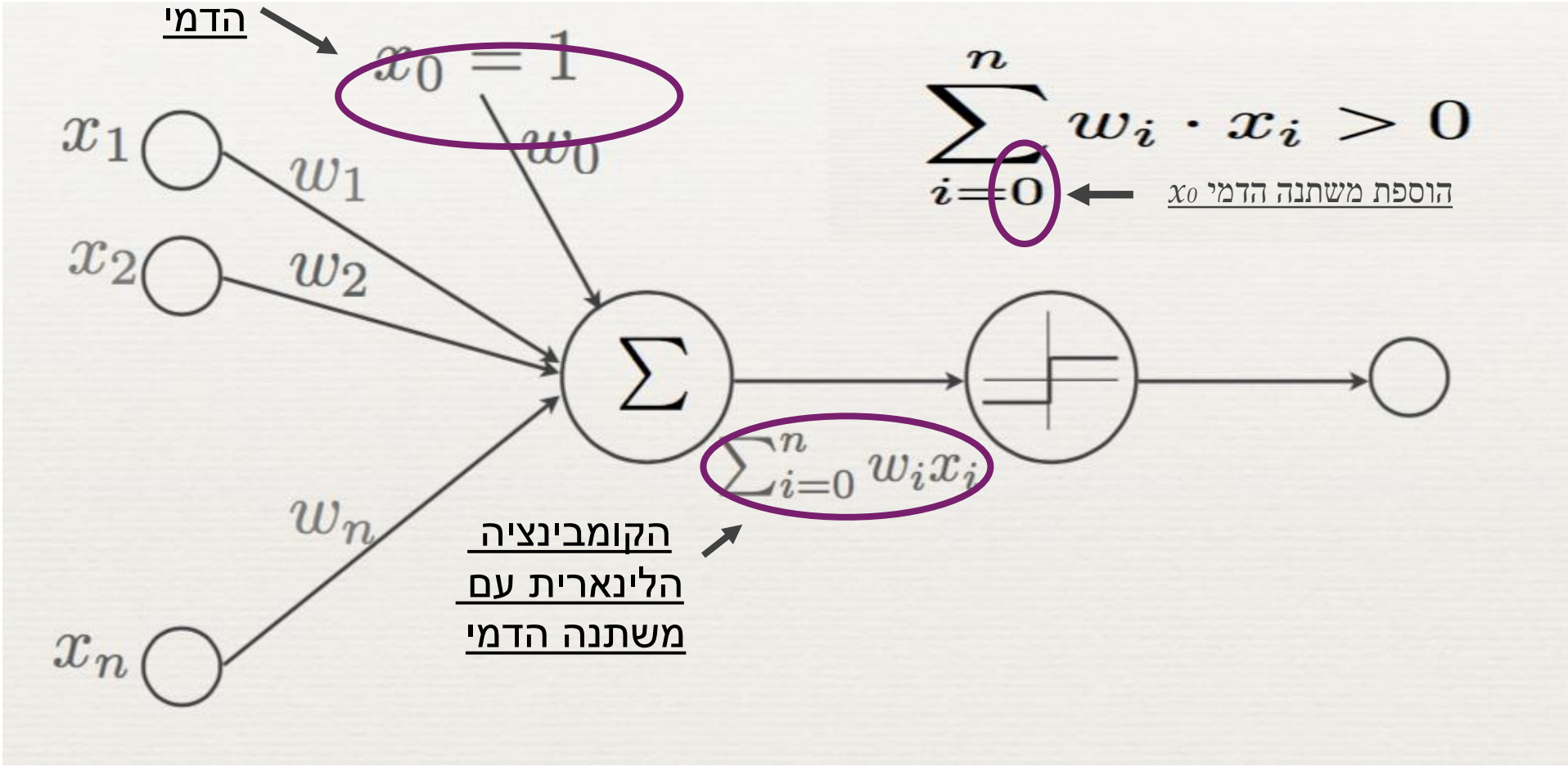
Convolutions large or small? - Recap



Convolutions on the low-level feature map produce mid-level features and so on

Perceptron - Recap

כדי לפשט את הביטוי, מוסיפים משתנה דמי ומקבלים: $x_0 = 1$



Training perceptron with sign function - recap

סימונים:

- η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)
- t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה
- o הערך שנותן ה נוירון עבור הדוגמא הנוכחית
- $\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה
- Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

אלגוריתם:

perceptron_train(training examples, η):

- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x}, t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta(t-o)x_i$
  - $w_i^{t+1} := w_i^t + \Delta w_i$

גרסה זו קיימת, למרות שפונקציית ה-sign, אינה גזירה ואינה דפרנציבלית

Gradient Descent for Logistic Regression training (with sigmoid function) - recap

```
E=1/2 sum((t[i]-o[i])^2
dJ/dw[j]=(t-o)*-1*(o*(1-o))*x[j]
=-(t-o)*(o*(1-o))*x[j]
delta(w[j])=delta(w[j])-eta*dJ/dw[j]
= delta(w[j])=delta(w[j])-[-(t-o)*(o*(1-o))*x[j]]
= delta(w[j])=delta(w[j])+t-o)*(o*(1-o))*x[j]
```

סימונים:

- η קבוע (קטן מ-1) הקובע את קצב ההתכנסות (בדרך כלל 0.1)
- t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה
- o הערך שנותן ה נוירון עבור הדוגמא הנוכחית
- $\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה
- Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

אלגוריתם:

Gradient Descent (training examples, η):

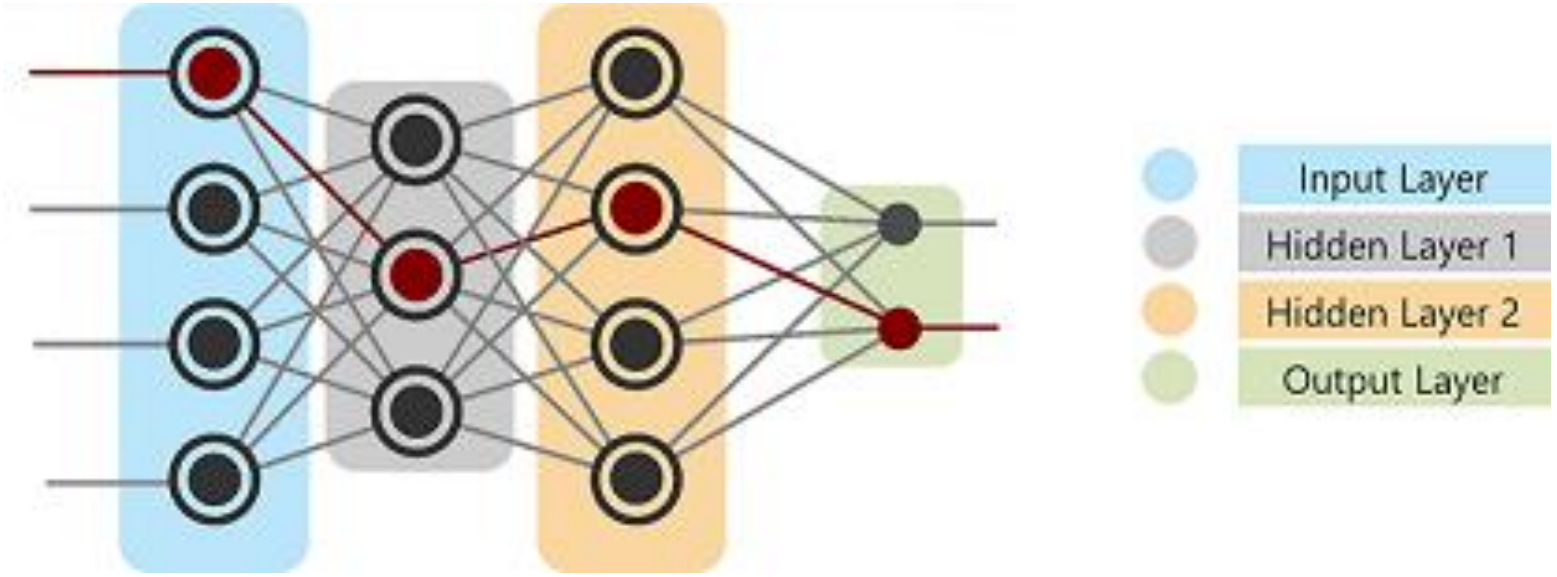
- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x}, t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta \cdot (t-o) \cdot o \cdot (1-o) \cdot x_i$
  - $w_i^{t+1} := w_i^t + \Delta w_i$

לעיתים קרובות מתעלמים מה-2 ע"י הכפלת פונקצית הטעות ב ½

J= $\sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2$

What’s next

Artificial Neural networks – part 2:
Multi-layer perceptrons / neurons



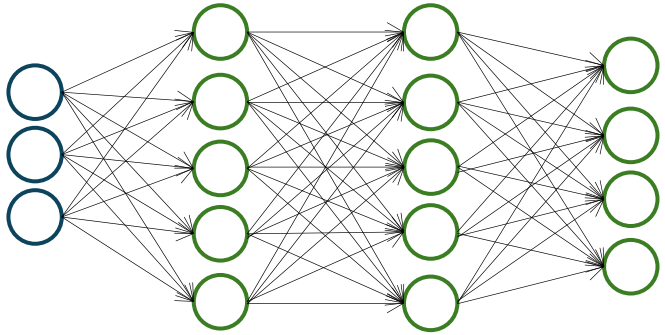
Artificial NN - recap

Pedestrian

Car

Motorcycle

Truck



$$h_w(x) \in \mathbb{R}^4$$

$$h_w(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

הולך רגל

$$h_w(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

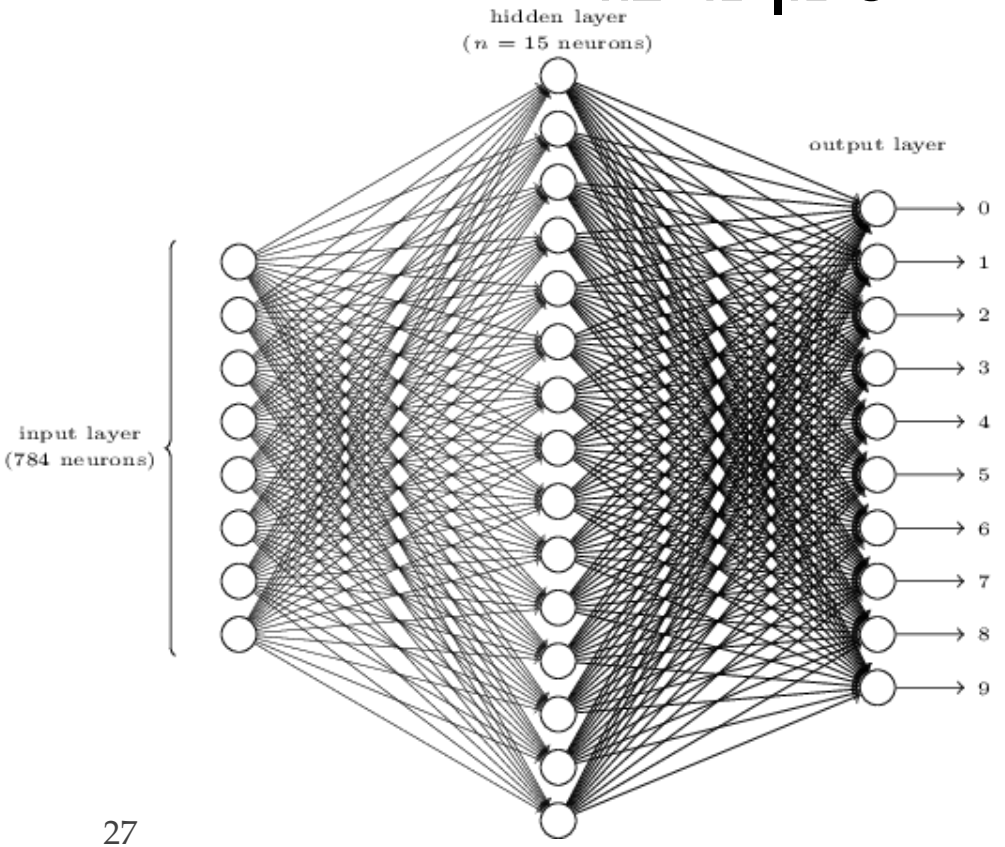
מכונית

$$h_w(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

אופנוע

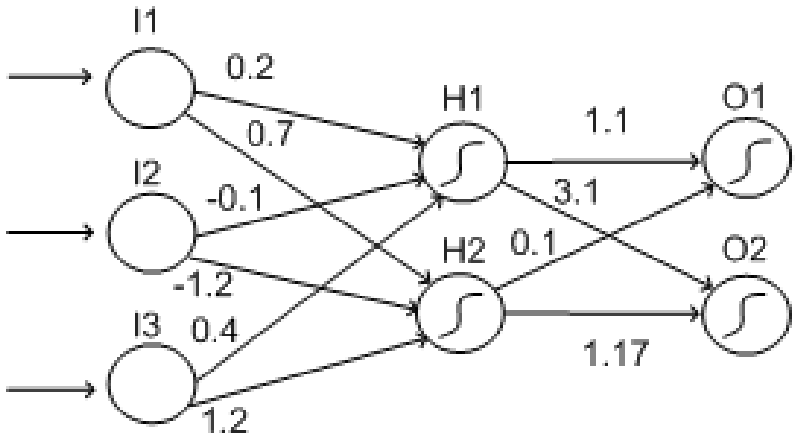
Artificial NN - recap

- אבחנה בין 10 סימנים שונים כדי להבין את הספרה שכל סימן מייצג



Artificial NN – Feed Forward - recap

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$



בצעו סימולציה לקריאת הרשת הבאה, הדוגמה אותה נרצה לסווג: (10,30,20)

אם היה מדובר בדוגמה חדשה:
פלט: O2 > O1 --> נסווג את הדוגמה כ-O2

שלב קריאת הרשת וחלחול הערך משכבת הקלט עד לשכבת הפלט נקרא Feed Forward

Input units		Hidden units			Output units		
Unit	Output	Unit	Weighted Sum Input	Output	Unit	Weighted Sum Input	Output
I1	10	H1	7	0.999	O1	1.0996	0.750
I2	30	H2	-5	0.0067	O2	3.1047	0.957
I3	20						

אם היה מדובר בדוגמה חדשה

Classic IR + Classic CV - recap

In the classic approach, searching for images

("Find me images that look like this sunset")

is treated as an Information Retrieval problem.

Bag of Visual Words - recap

The Concept: **"Bag of Visual Words" (BoVW)** or Global Color Histograms.

Just as Classic IR counts word frequencies (TF-IDF),

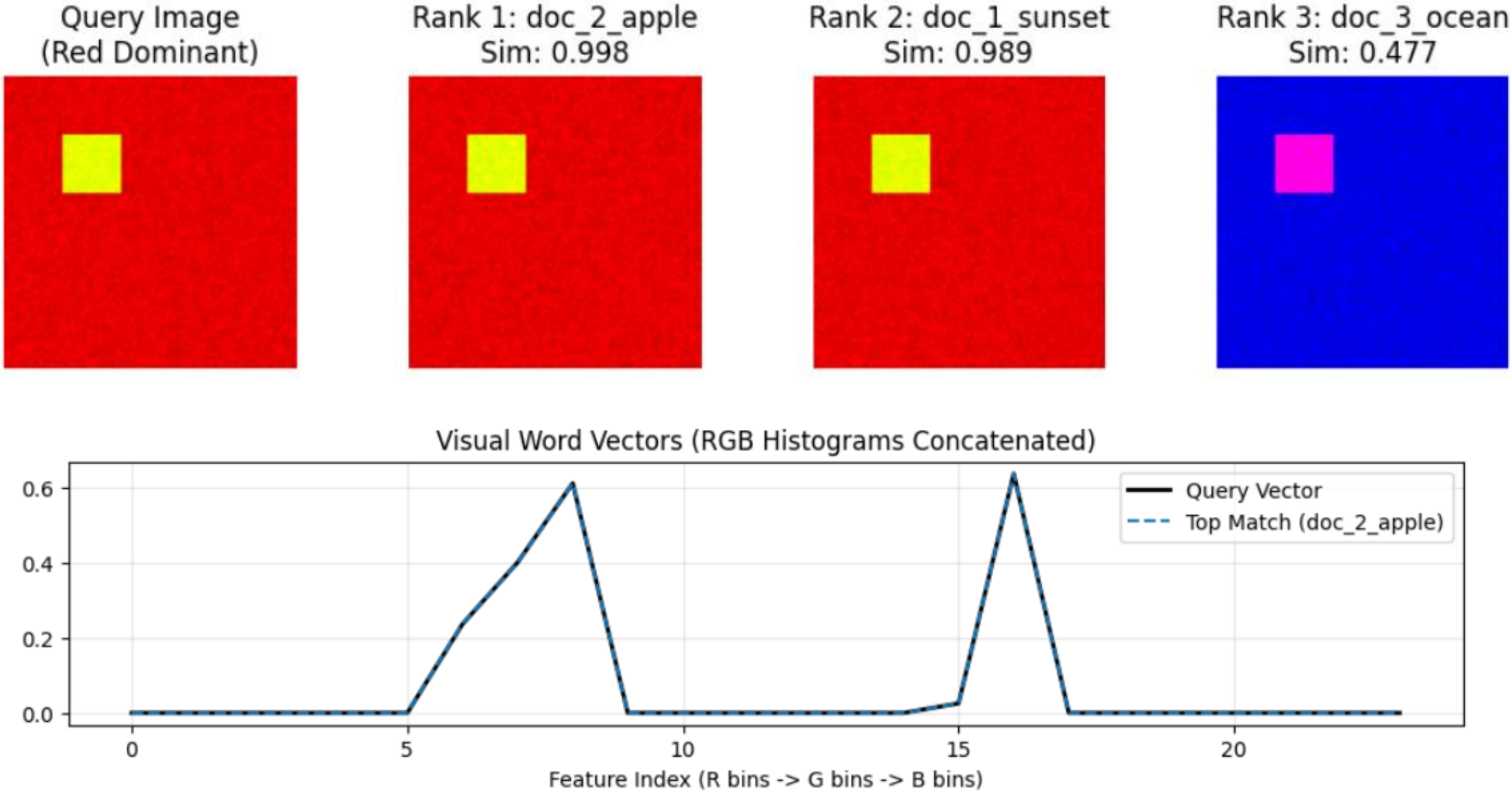
Classic CV counts Color Frequencies.

- **Word Dictionary** ≈ Color Palette (bins)
- **Document Vector** ≈ Color Histogram This technique, popularized by Swain & Ballard (1991), allows for incredibly fast image indexing.

  Demonstration: The Color Histogram Search Engine

  We will create a mini-database of images,

  them by their color distribution, and run a "**Visual Query**."


Query Image
(Red Dominant)
Rank 1: doc_2_apple
Sim: 0.998
Rank 2: doc_1_sunset
Sim: 0.989
Rank 3: doc_3_ocean
Sim: 0.477
Visual Word Vectors (RGB Histograms Concatenated)
Query Vector
Top Match (doc_2_apple)
0.6
0.4
0.2
0.0
0
5
10
15
20
Feature Index (R bins -> G bins -> B bins)

Understanding the IR+CV Flow: - recap

1. **Feature Vector**: Look at the bottom chart. The "Vector" is just a wavy line representing how much Red, Green, and Blue is in the image.
    - **Red Section** (Left): High peaks (because the image is red).
    - **Blue/Green Sections** (Right): Low/Flat (mostly background noise).
2. **Indexing**: The database stores these vectors, not the pixels.
3. **Retrieval**: The system compares the Query Vector to the Database Vectors.
    - **Result**: The "Ocean" image (Blue) has a completely different vector shape, so it gets a low score. The "Sunset" and "Apple" images have similar vectors to the query, so they are ranked high.

This is the grandfather of modern Reverse Image Search. While modern systems use semantic vectors (Deep Learning), the logic of Vector Space Retrieval remains the same.

Classic Multimedia Information Retrieval - MMIR (Before Deep Learning)

MMIR systems use Late Fusion.

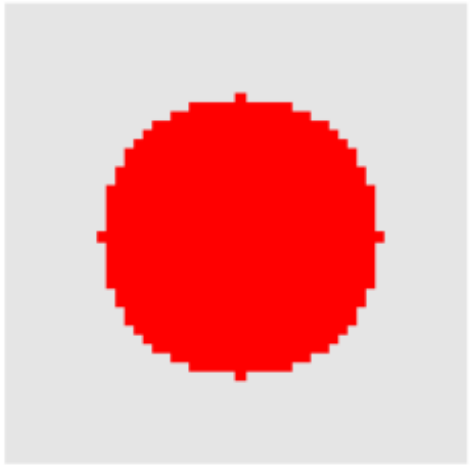
1. **Text Engine**: Solved the semantic matching (e.g., "Find me a car").
2. **Vision Engine**: Solved the stylistic matching (e.g., "Find me something red").
3. **Fusion**: A weighted mathematical combination of the two scores.

For example:

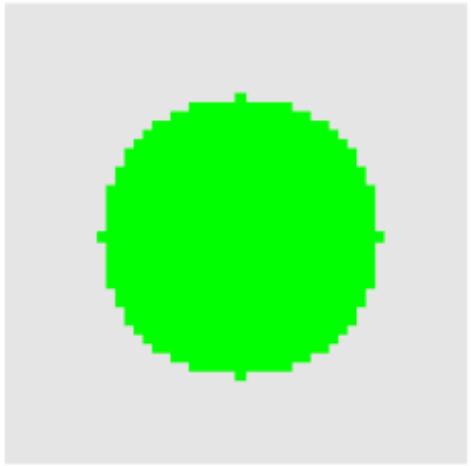
We build a "**Visual Reranking**" engine.

The user searches for "Apple", but provides a visual hint (a red color swatch) to imply they want Red Apples, not Green ones.

Doc 0
Fresh Red Apple

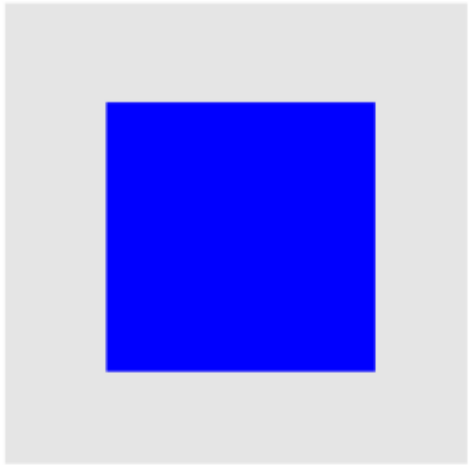


Doc 1
Green Granny Apple



Doc 2
Red Sports Car

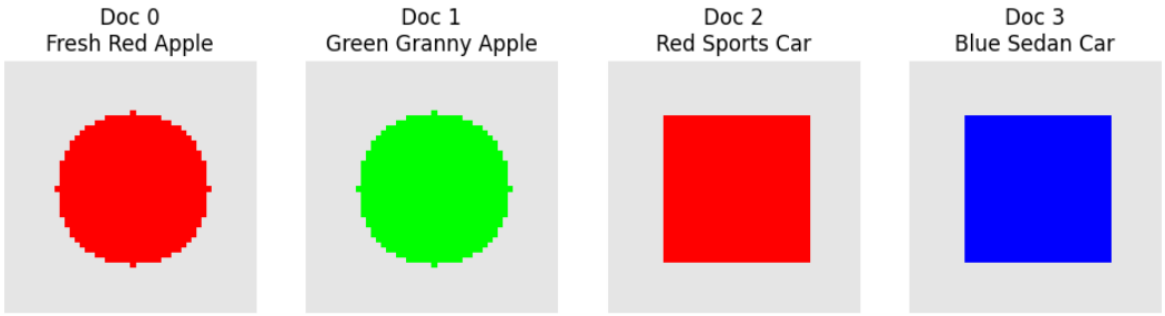
Doc 3
Blue Sedan Car



Step 2: The Text Pipeline (TF-IDF)

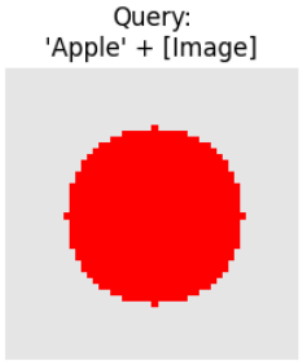
This handles the semantics.

It ensures that if the user asks for "Apple", we don't return a "Car" just because it's red.



--- Text Only Scores (TF-IDF) ---

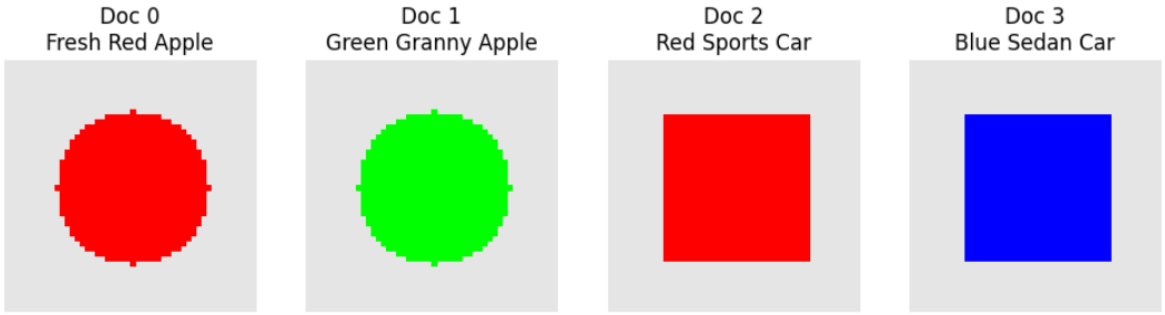
- Doc 0 (Fresh Red Apple): 0.5264
- Doc 1 (Green Granny Apple): 0.4869
- Doc 2 (Red Sports Car): 0.0000
- Doc 3 (Blue Sedan Car): 0.0000



Step 3: The Vision Pipeline (Color Histograms)

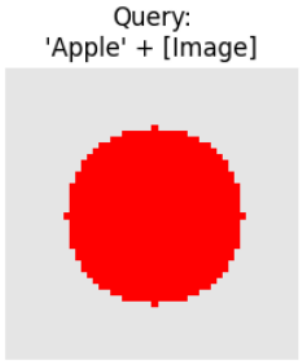
This handles the style/appearance.

We extract color features to find "Redness".



--- Visual Only Scores (Histogram) ---

- Doc 0 (Fresh Red Apple): 1.0000
- Doc 1 (Green Granny Apple): 0.9265
- Doc 2 (Red Sports Car): 0.9976
- Doc 3 (Blue Sedan Car): 0.9087

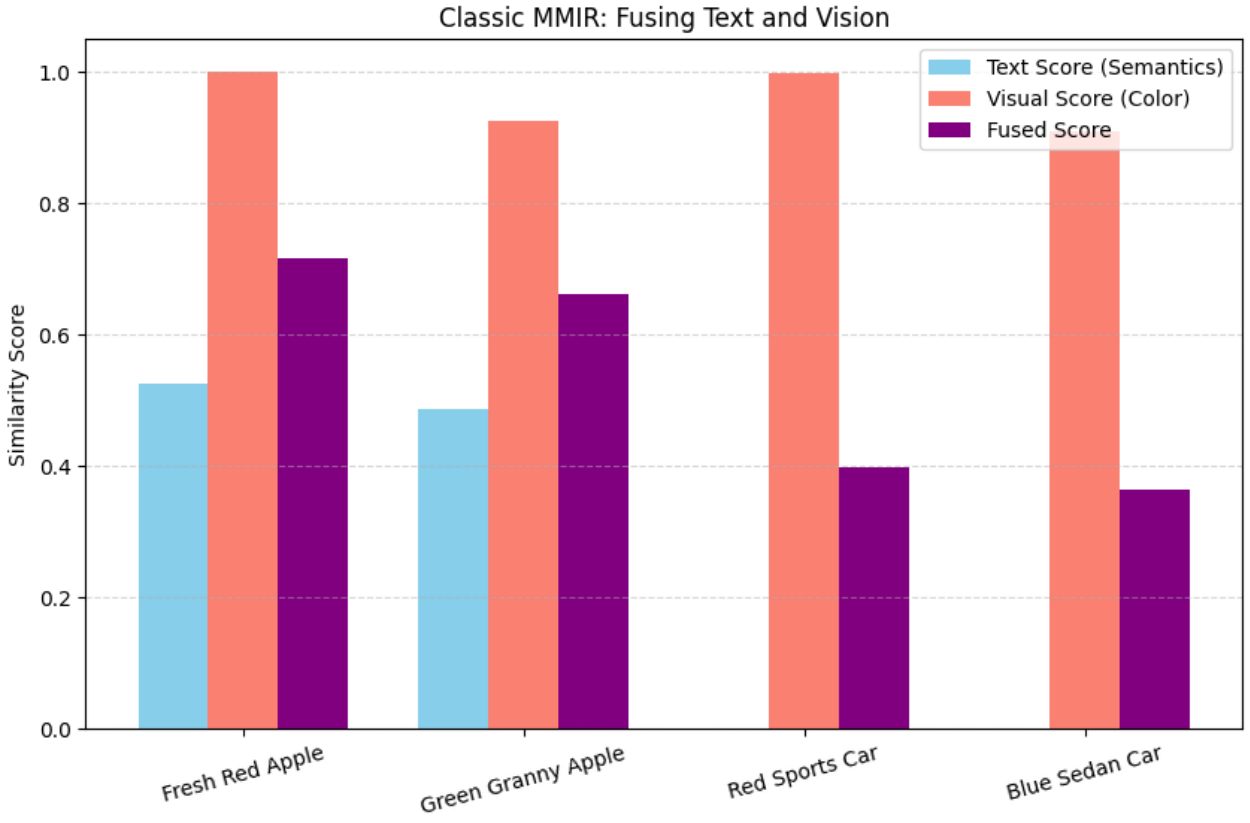


Step 4: Late Fusion (The "Mix")

We combine the scores using a weighting parameter (α).

$$\text{FinalScore} = \alpha \cdot \text{TextScore} + (1 - \alpha) \cdot \text{VisualScore}$$

In this scenario, we usually weight Text higher (to get the right object)
and use Vision to rerank (to get the right variant).



- --- Final Ranked Results ---
- Rank 1: Fresh Red Apple (Score: 0.7158)
- Rank 2: Green Granny Apple (Score: 0.6627)
- Rank 3: Red Sports Car (Score: 0.3990)
- Rank 4: Blue Sedan Car (Score: 0.3635

Analysis of the Plot

1. Red Apple (The Winner):
    - Text (Blue Bar): High. Matches "Apple".
    - Visual (Red Bar): High. Matches Red color.
    - Result: Highest Purple bar.
2. Green Granny Apple:
    - Text (Blue Bar): High. Matches "Apple".
    - Visual (Red Bar): Low. It's green!
    - Result: Pushed down to 2nd place. This is the "Visual Reranking" working.
3. Red Sports Car:
    - Text (Blue Bar): Zero. "Car" != "Apple".
    - Visual (Red Bar): High. It is red.
    - Result: Even though it looks right, the text filter kills it.

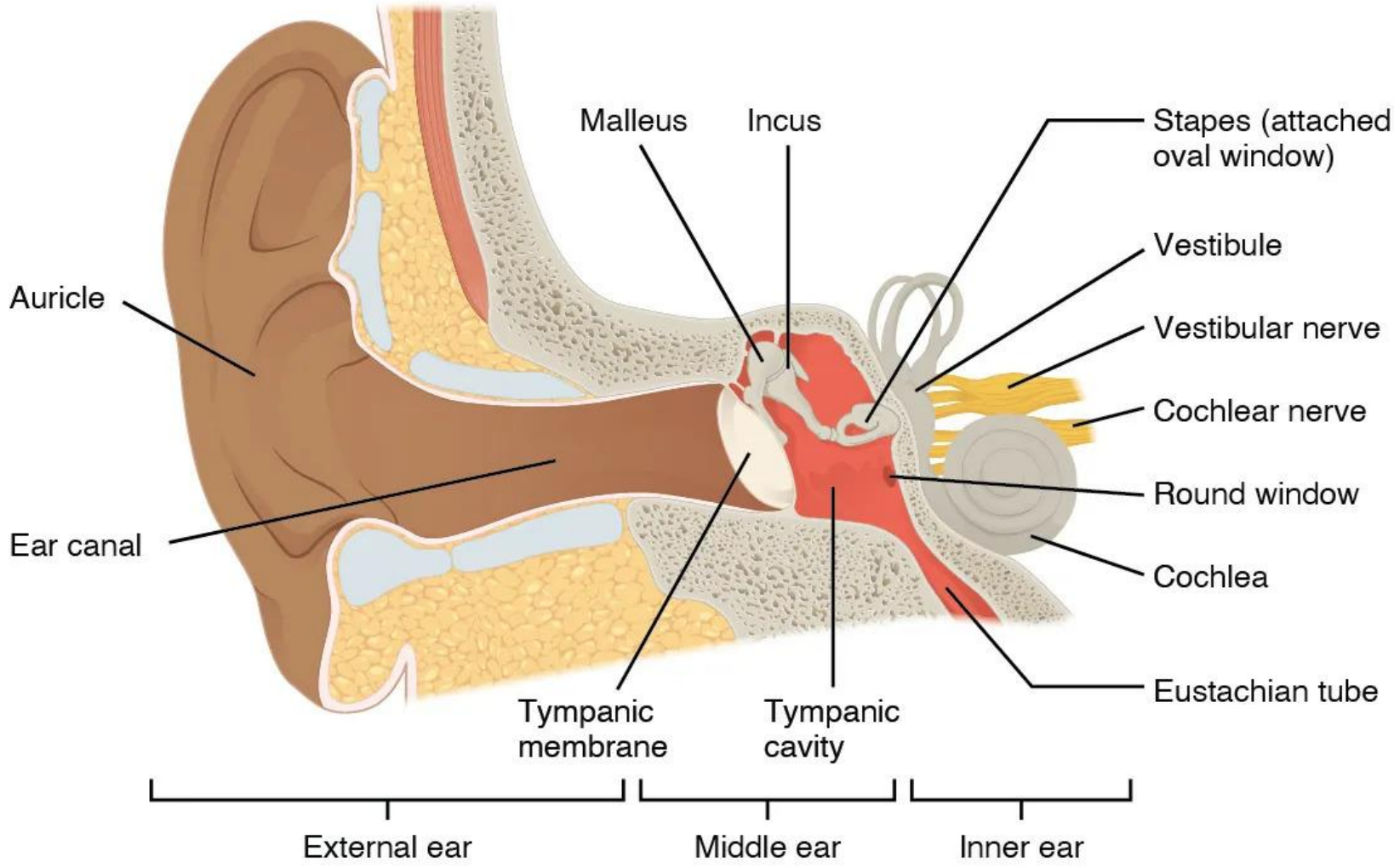
      This proves the system respects semantics.

Sound

- Physical definition
  - A **vibration** that propagates as an **acoustic wave**, through a transmission medium such as a gas, liquid or solid.

- Psychophysical definition
  - ***Reception*** of such acoustic waves and their ***perception*** by the brain.

Structures of the Ear

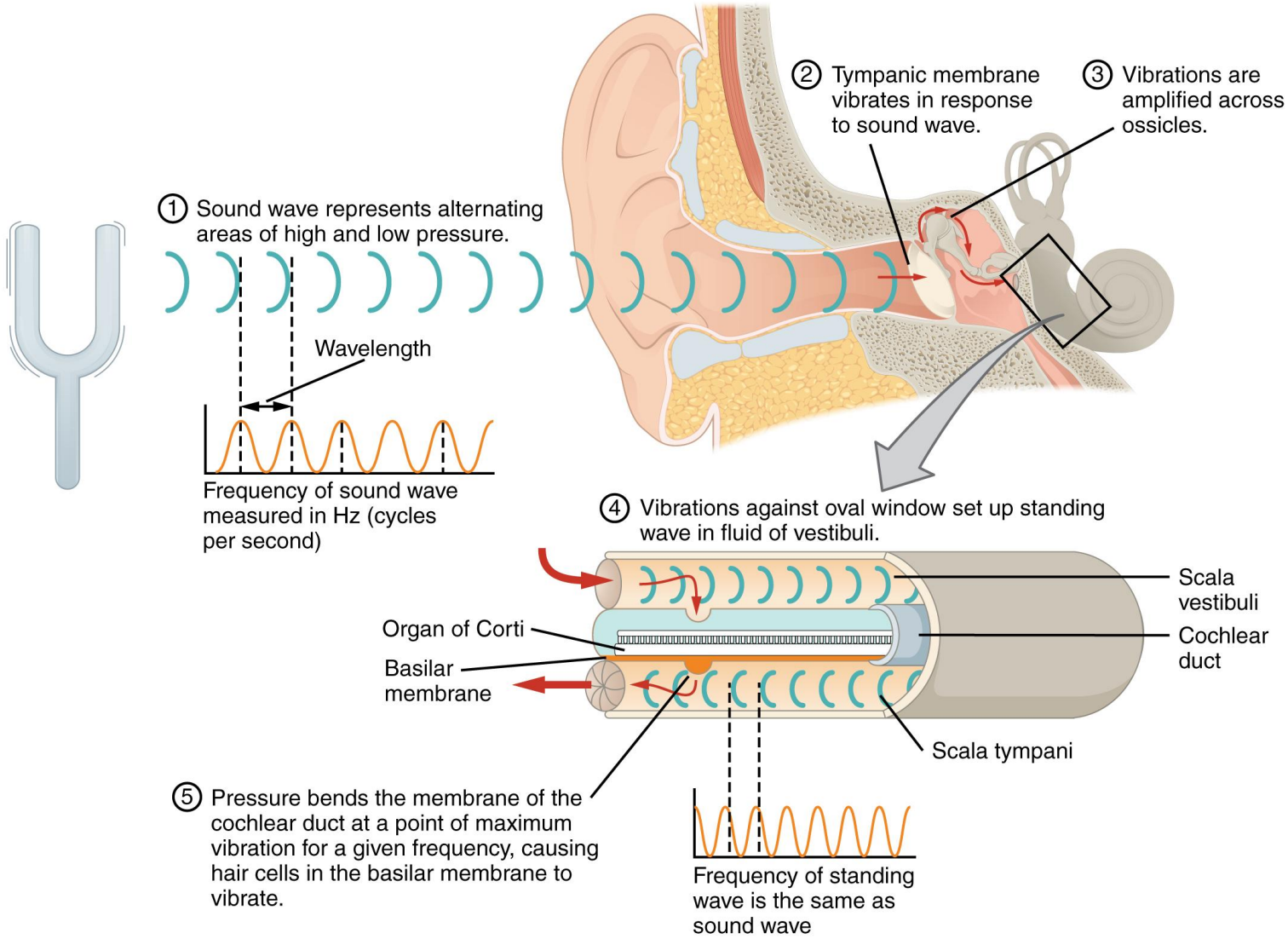


<u>The external ear</u> contains the auricle, ear canal, and tympanic membrane.

<u>The middle ear</u> contains the ossicles and is connected to the pharynx by the Eustachian tube.

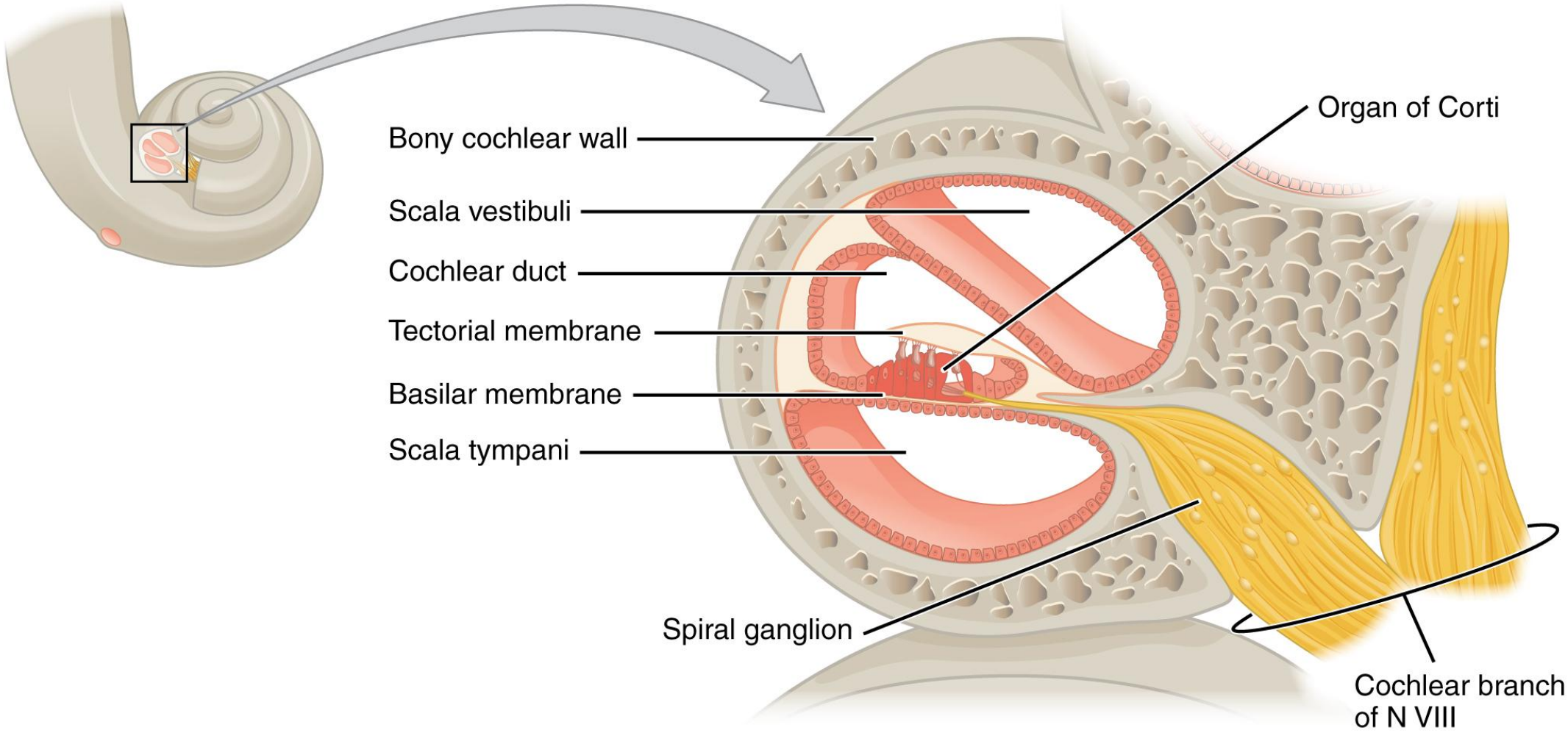
<u>The inner ear</u> contains the cochlea and vestibule, which are responsible for audition and equilibrium, respectively

Transmission of Sound Waves to Cochlea

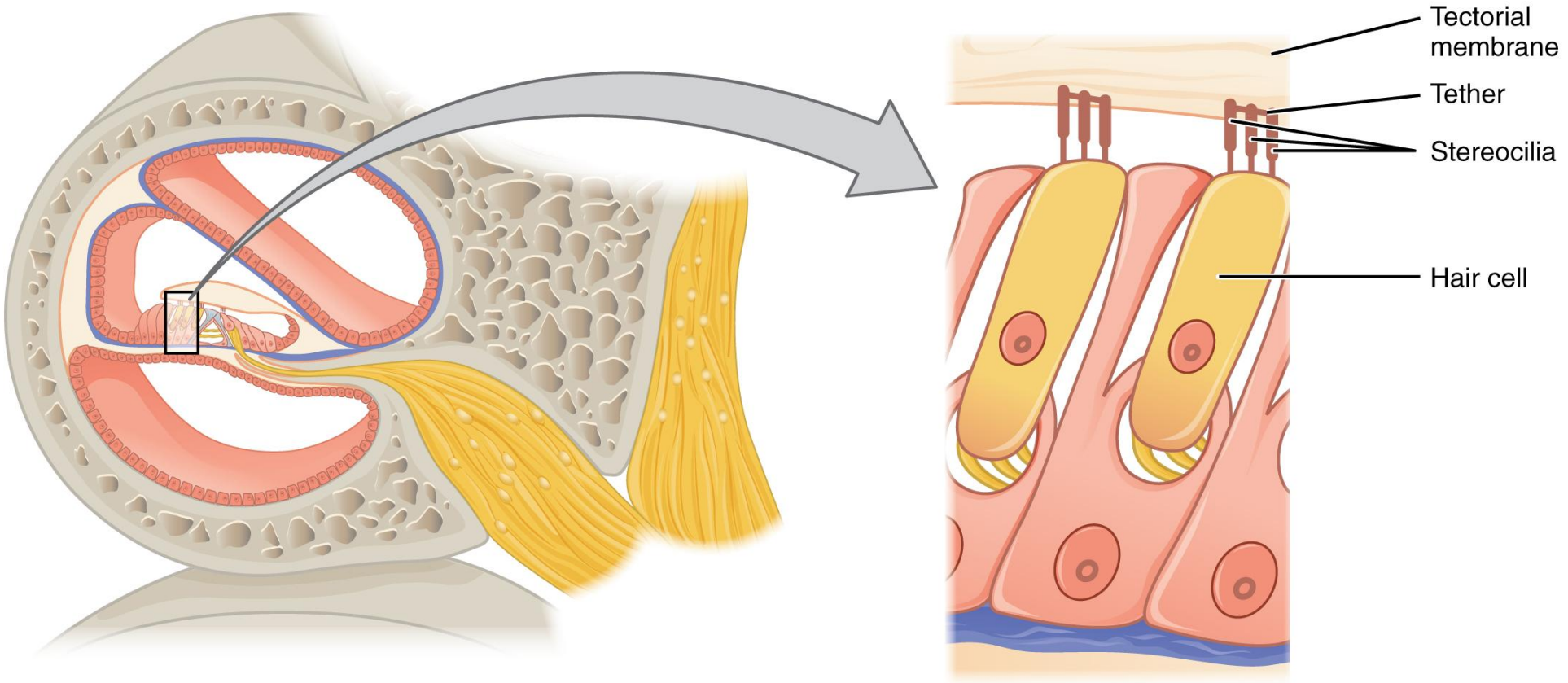


- *A sound wave causes the tympanic membrane to vibrate.*
- *This vibration is amplified as it moves across the malleus, incus, and stapes.*
- *The amplified vibration is picked up by the oval window causing pressure waves in the fluid of the scala vestibuli and scala tympani.*
- *The complexity of the pressure waves is determined by the changes in amplitude and frequency of the sound waves entering the ear.*

Cross Section of the Cochlea



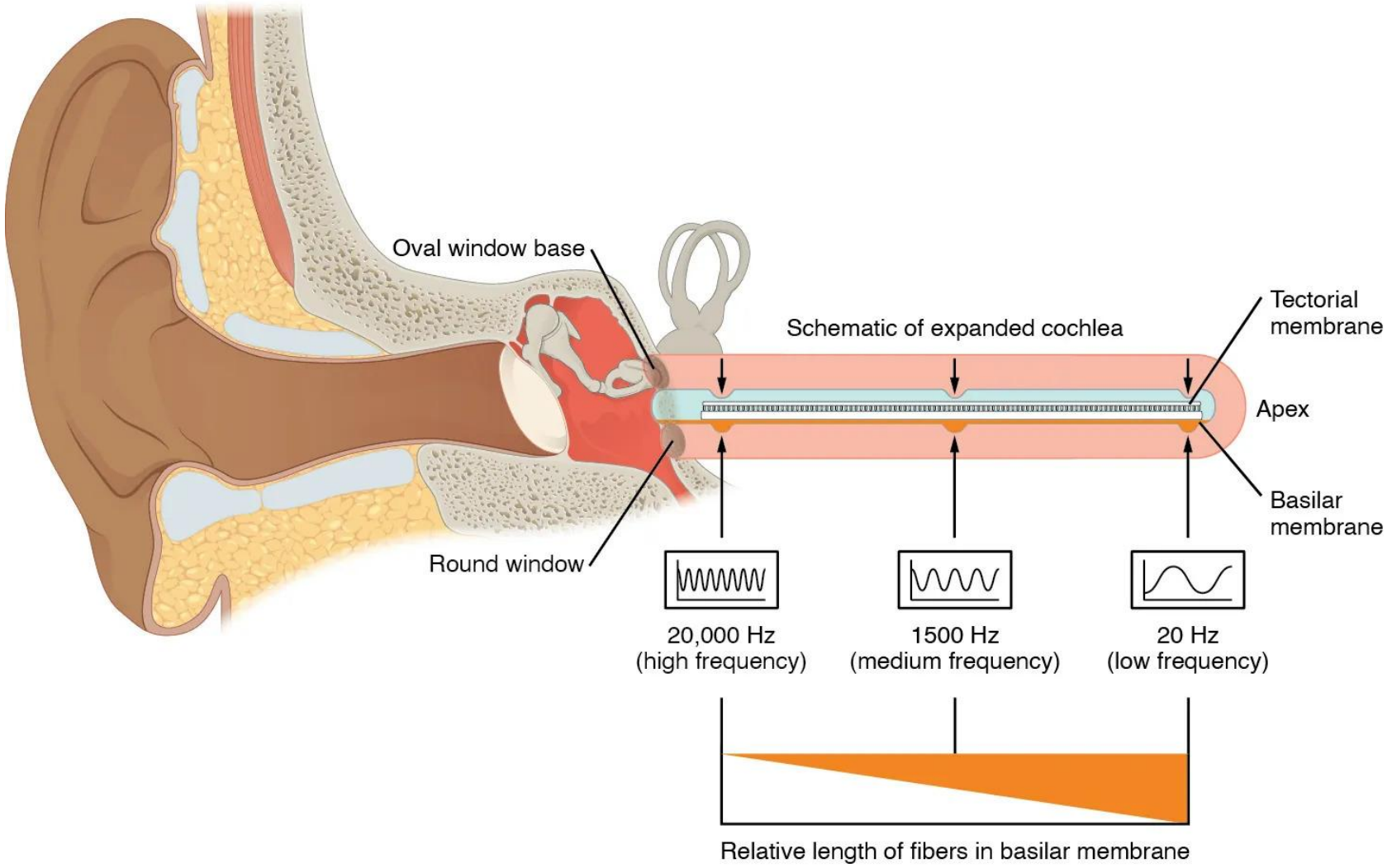
Hair Cell



The hair cell is a mechanoreceptor with an array of stereocilia emerging from its apical surface.

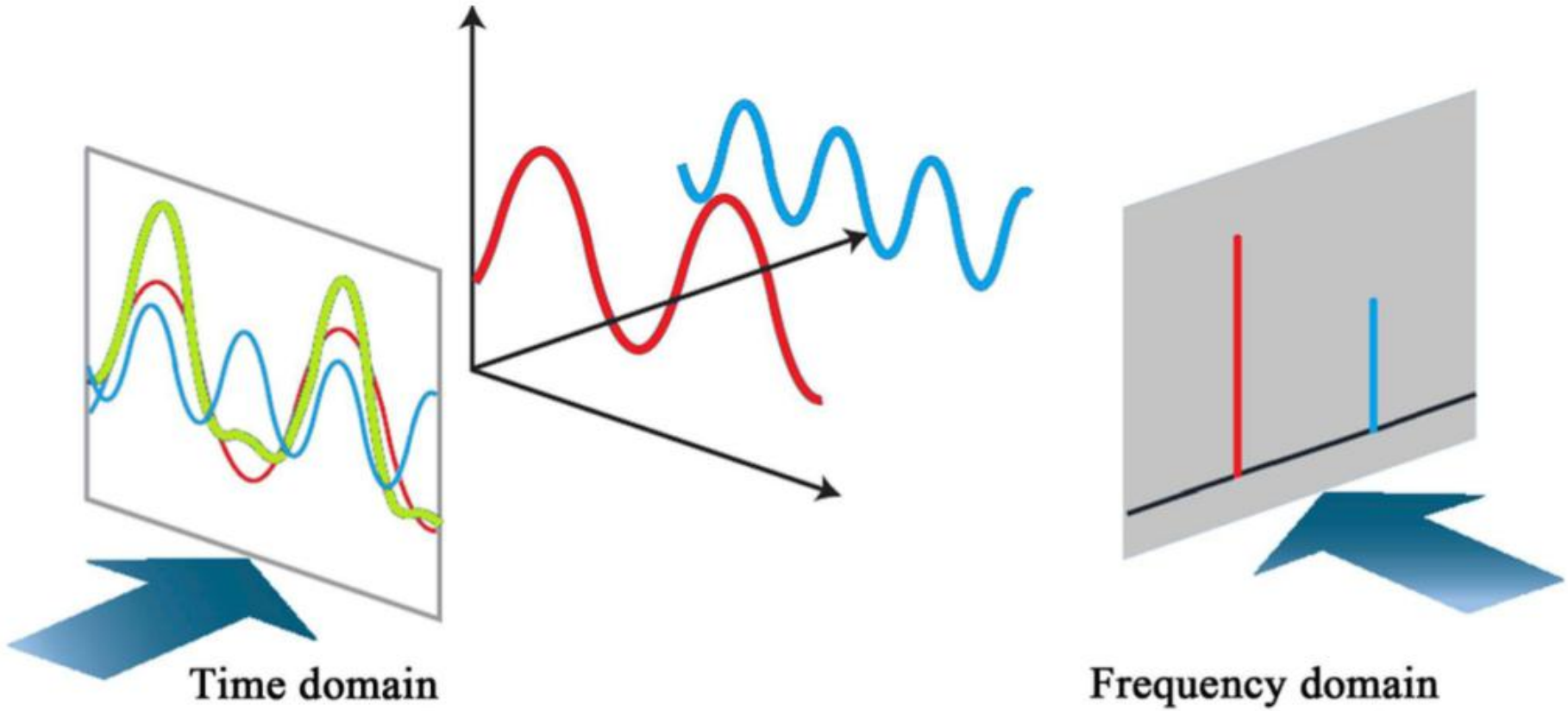
The stereocilia are tethered together by proteins that open ion channels when the array is bent toward the tallest member of their array, and closed when the array is bent toward the shortest member of their array.

Frequency Coding in the Cochlea



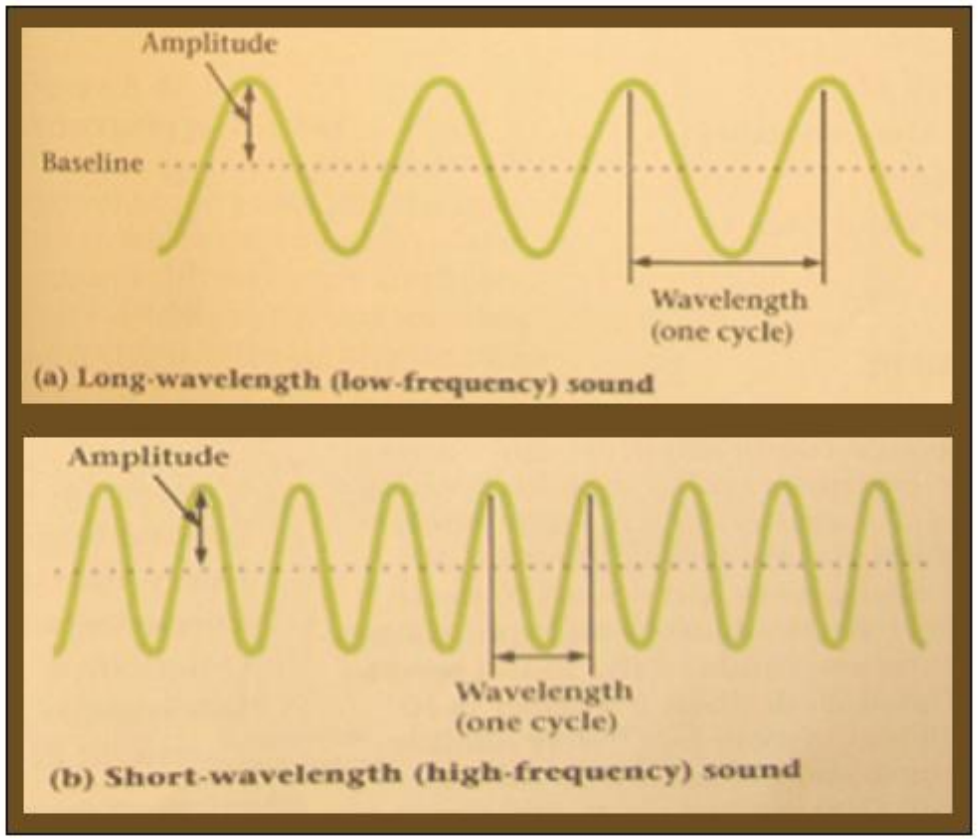
- *The standing sound wave generated in the cochlea by the movement of the oval window deflects the basilar membrane on the basis of the frequency of sound.*
- *Therefore, hair cells at the base of the cochlea are activated only by high frequencies, whereas those at the apex of the cochlea are activated only by low frequencies.*

Time domain vs frequency domain



Physical Dimensions

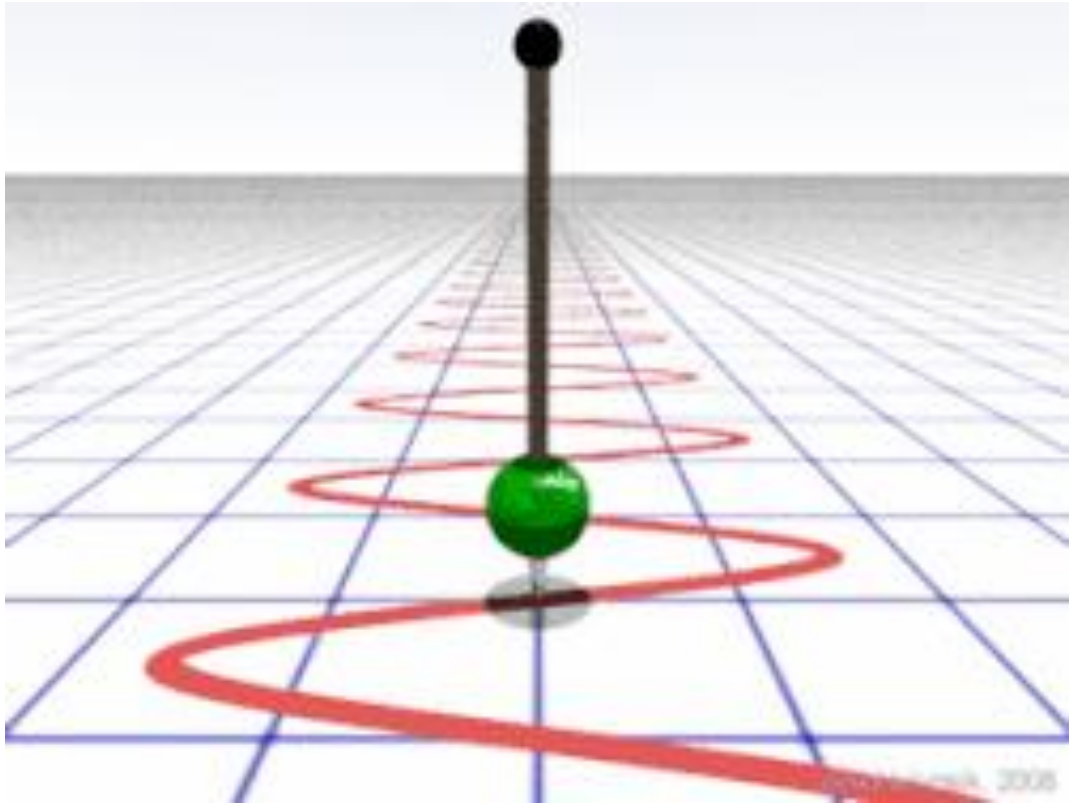
- Amplitude
  - height of a cycle
  - relates to loudness
- Wavelength (*w*)
  - distance between peaks
- Frequency (λ)
  - cycles per second
  - relates to pitch
  - λw = velocity
- Most sounds mix many frequencies & amplitudes

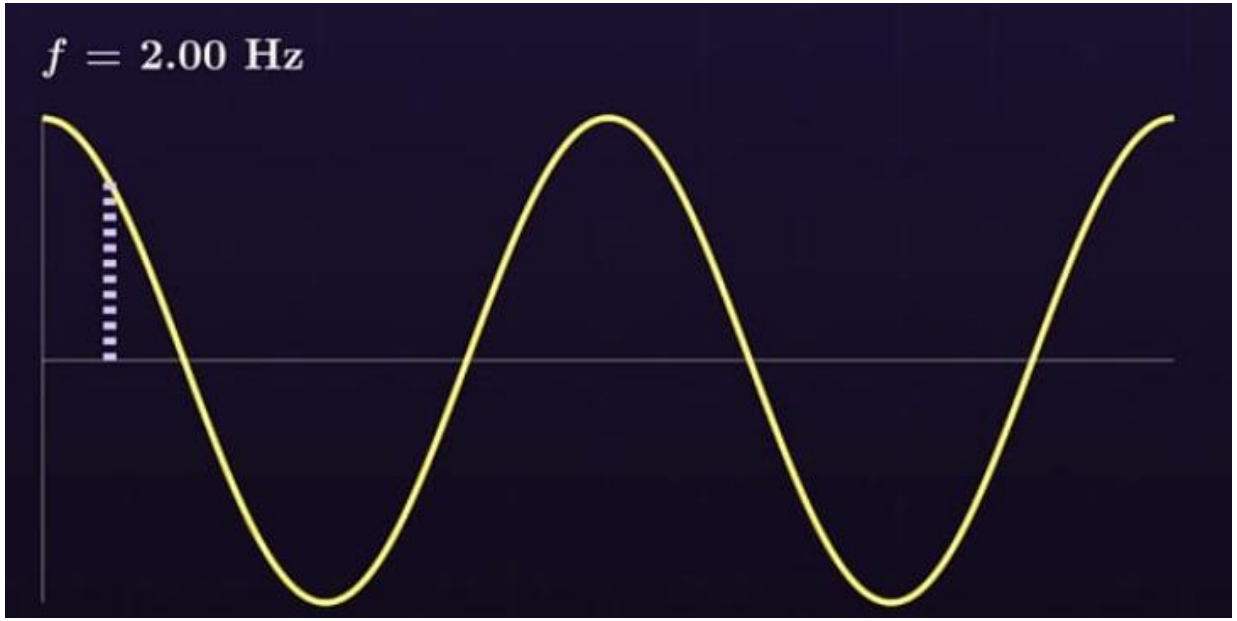


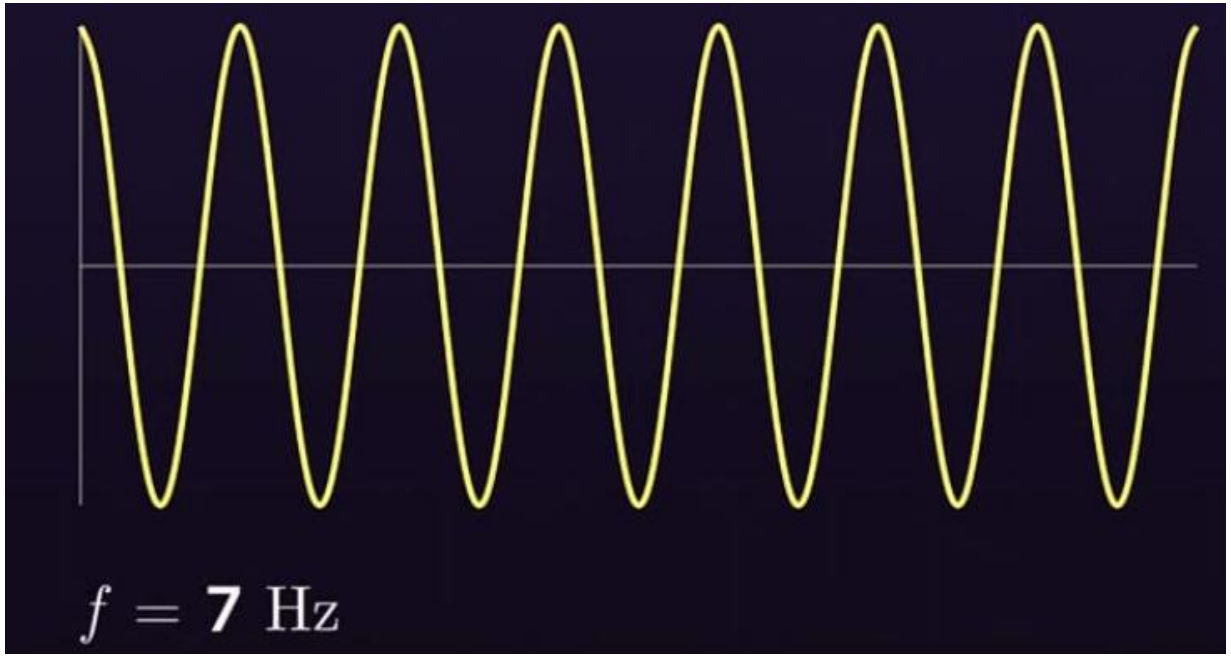
Sound is repetitive changes in air pressure over time

Frequency

Frequency is the number of occurrences of a repeating event per unit of time. Frequency is an important parameter used in science and engineering to specify the rate
of oscillatory and vibratory phen omena, such as mechanical vibrations, audio
signals (sound), radio waves, and light.

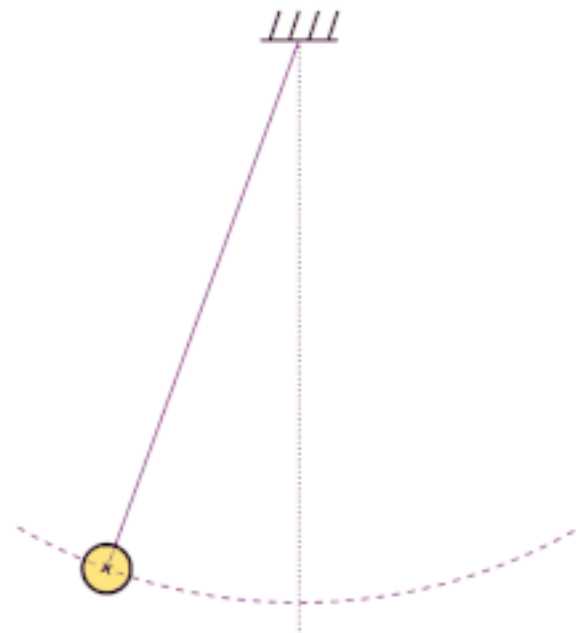



$f = 2.00$ Hz

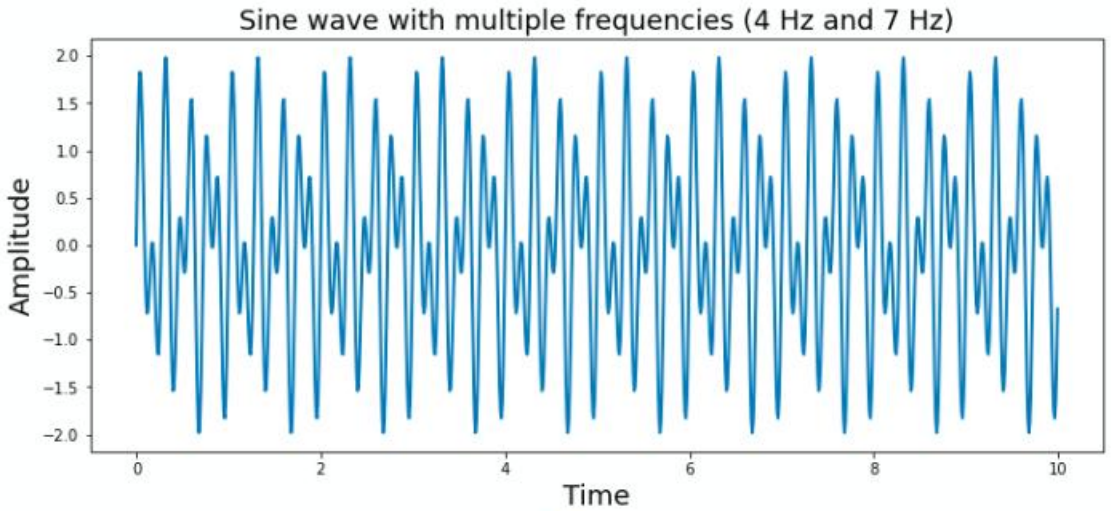

$f = 7$ Hz

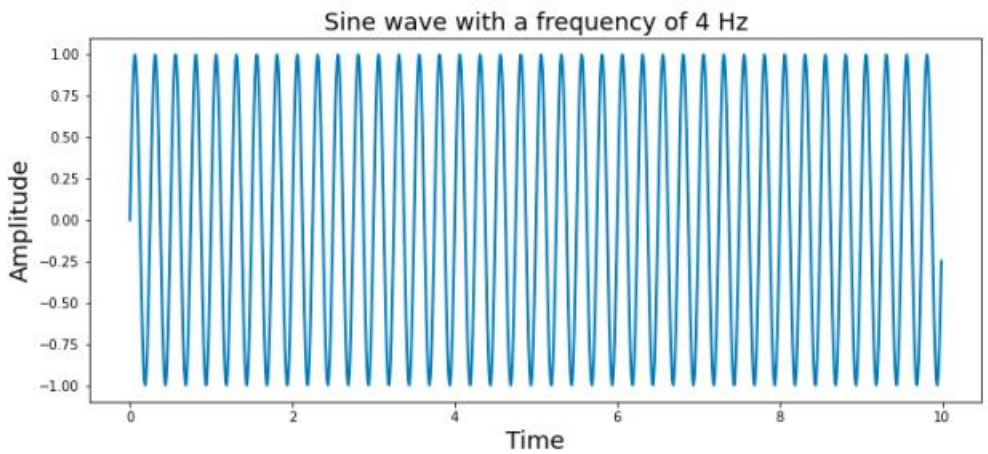
Key Formulas & Concepts

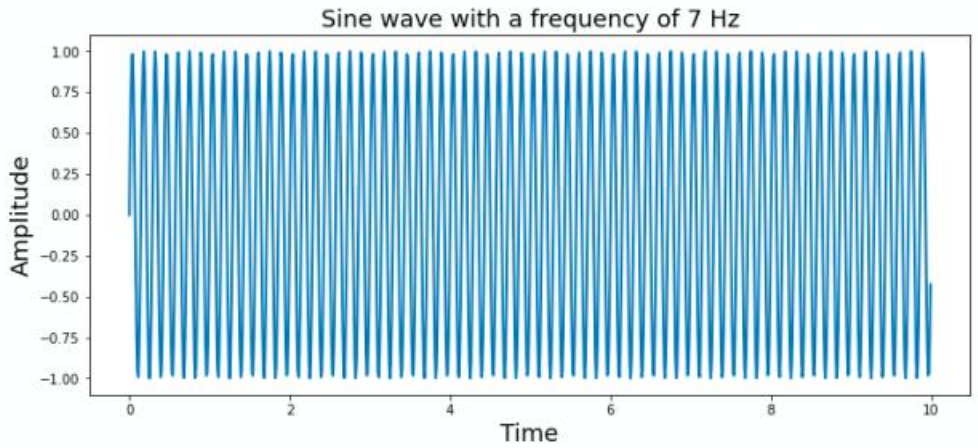
- **v = fλ**: The most common formula, where v (or c) is wave speed, f is frequency (Hz), and λ (lambda) is wavelength (m).
- **v = λ/T**: Another version, using the period T (seconds) instead of frequency, since $f = 1/T$.
- **Medium Dependence**: The velocity is set by the medium's physical characteristics (e.g., tension in a string, temperature in air, depth in water).
- **Energy Transfer**: It's the speed at which wave energy moves, not the speed of the particles in the medium (which oscillate locally).



A signal in time domain

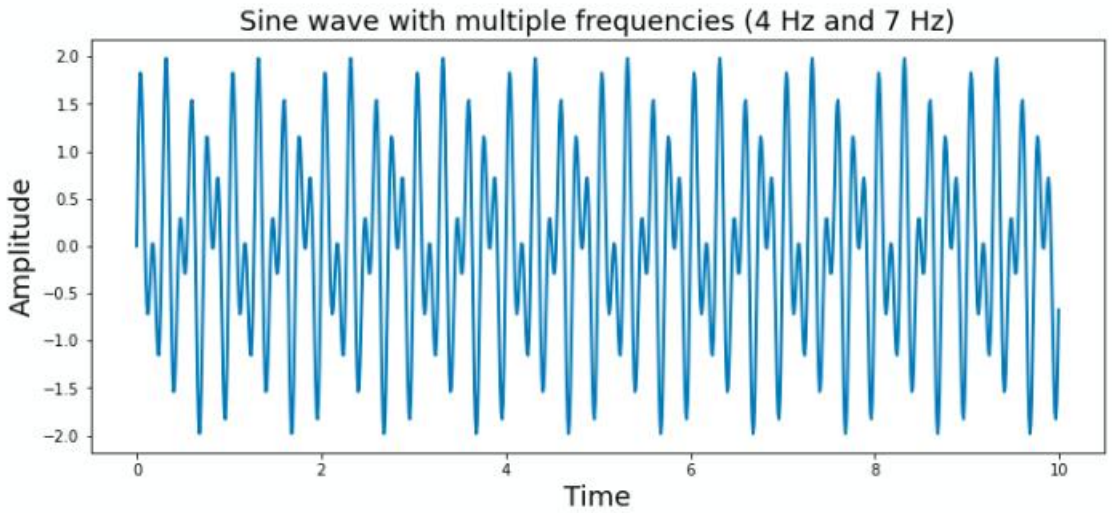


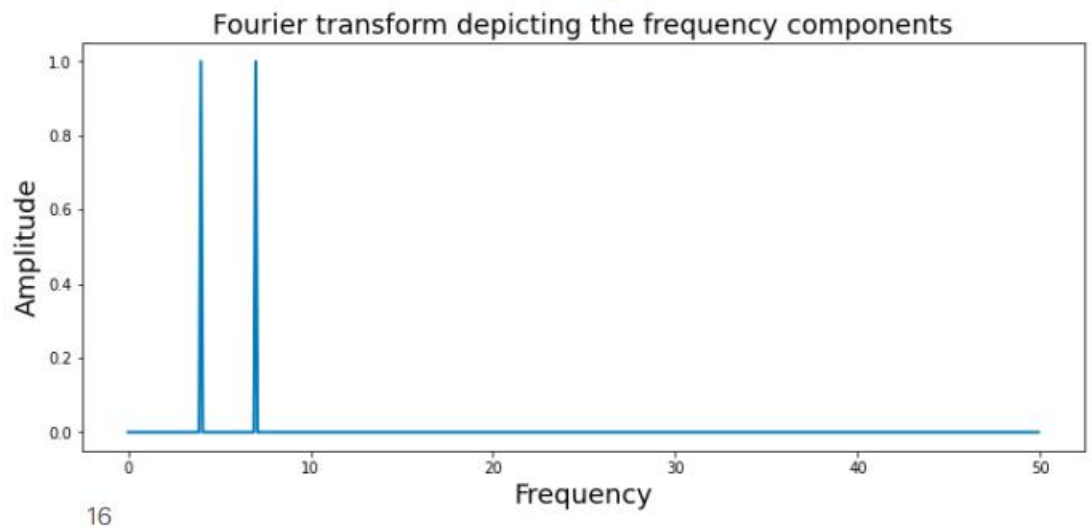




15

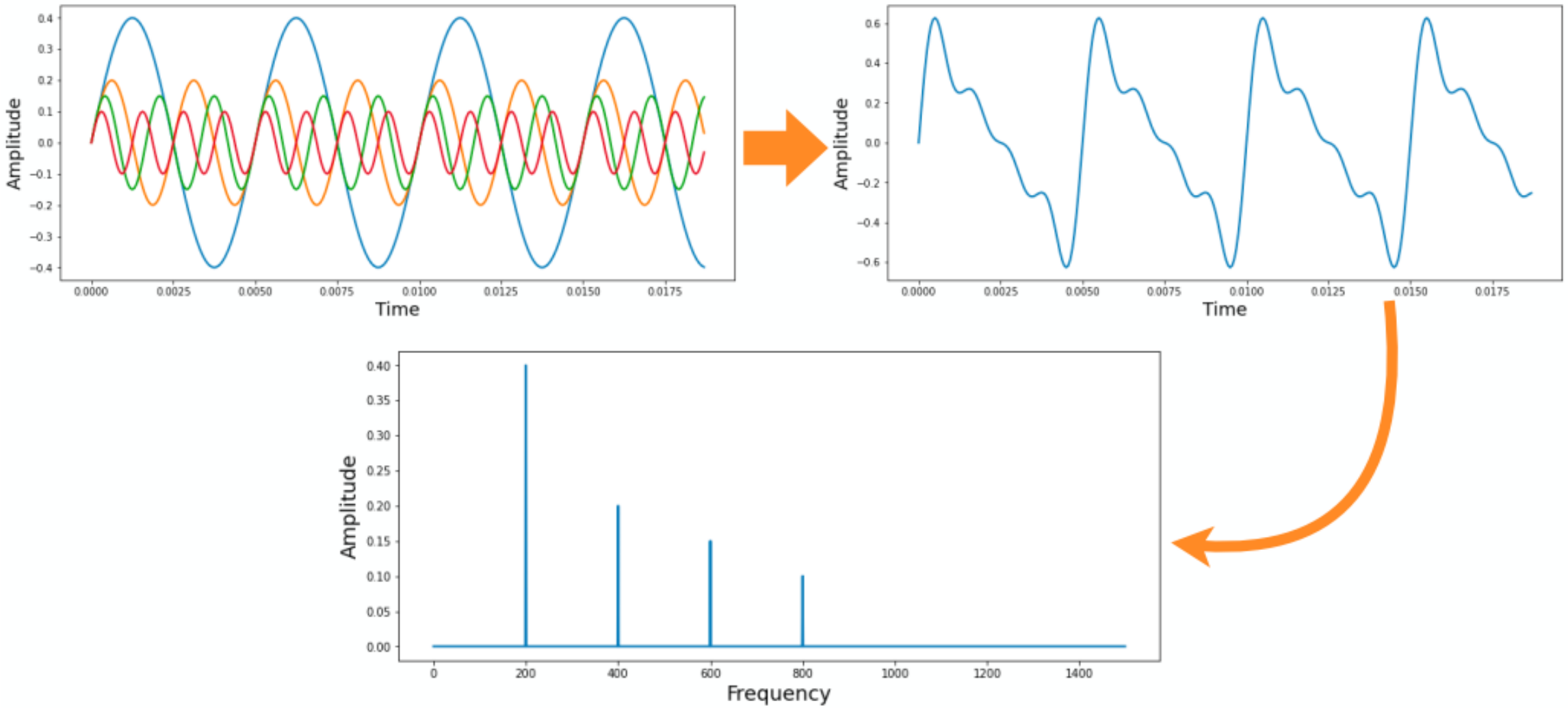
Frequency-domain representation





16

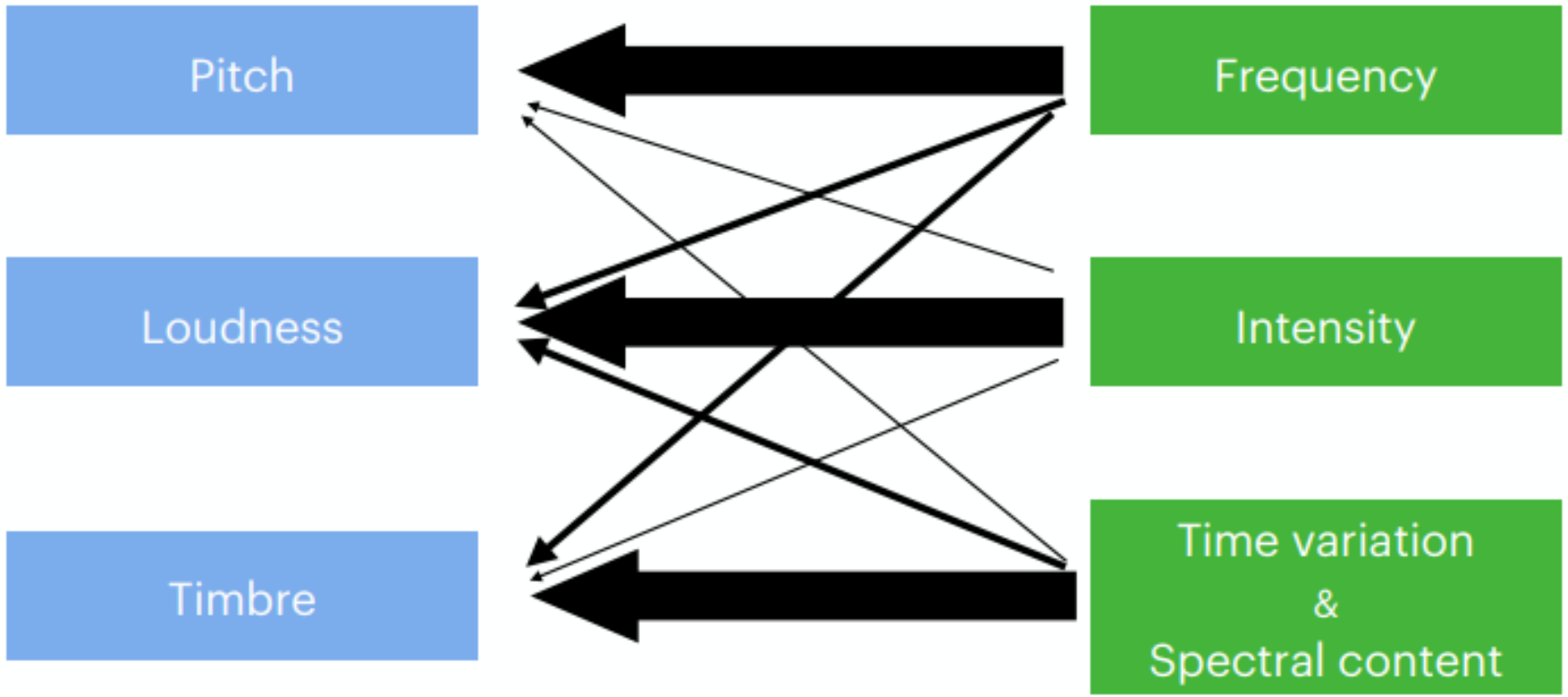
Frequency-domain representation



Quantifying sound

- Perceptual characteristics
  - Loudness
  - Pitch
  - Timbre (tone color)
- Physical characteristics
  - Intensity
  - Frequency
  - Time variation and harmonic spectrum

Physical property vs perceptual property



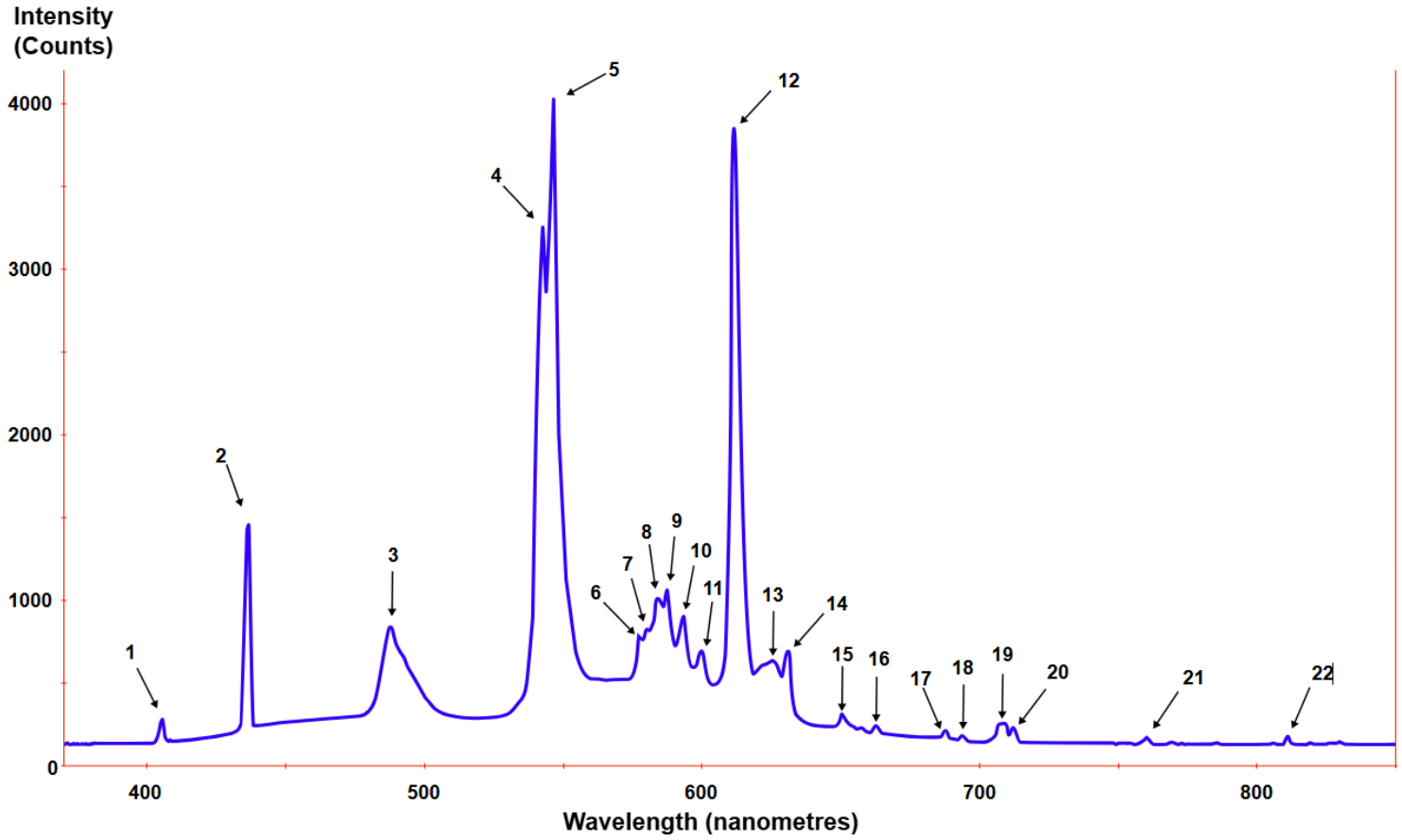
Psychological Dimensions

- Loudness
  - higher amplitude results in louder sounds
  - measured in decibels (db), 0 db represents hearing threshold
- Pitch
  - higher frequencies perceived as higher pitch
  - hear sounds in 20 Hz to 20,000 Hz range

Examples of Factors Affecting Velocity

- **Sound Waves**: Faster in solids than liquids, faster in liquids than gases; speed increases with temperature in air.
- **Water Waves**: Velocity changes with water depth (deep water waves have specific formulas).
- **Light/Electromagnetic Waves**: Velocity changes when passing through different transparent media (refraction).

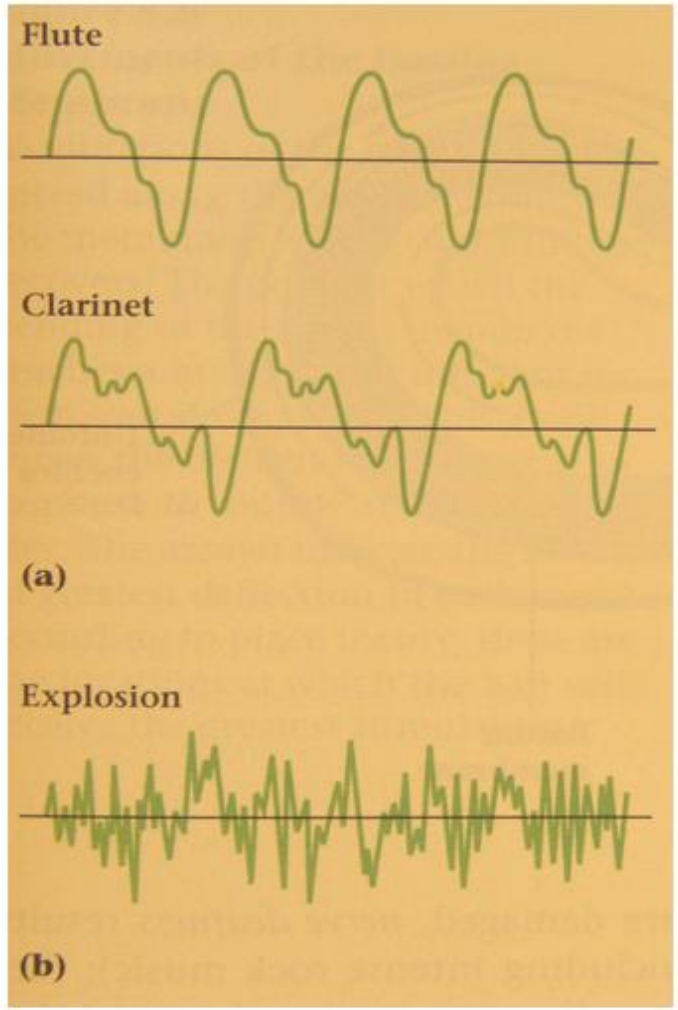
Spectral density



Spectral density (or **power spectrum**- the statistical average of the energy or power of any type of signal as analyzed in terms of its frequency content, is called its **spectral density**.

Psychological Dimensions (cont.)

- Timbre (tam-bre)
  - complex patterns added to the lowest, or *fundamental*, frequency of a sound, referred to as *spectra*
  - spectra enable us to distinguish musical instruments
- Multiples of fundamental frequency give music
- Multiples of unrelated frequencies give noise



Sound Intensity

- *Intensity* (I) of a wave is the rate at which sound energy flows through a unit area (A) perpendicular to the direction of travel

$$I = \frac{1}{A}\frac{\Delta E}{\Delta t} = \frac{P}{A}$$

P measured in watts (W), A measured in m^2

- *Threshold of hearing* is at 10^{-12} W/m^2
- *Threshold of pain* is at 1 W/m^2

Decibel (dB) Scale

- Describes intensity relative to threshold of hearing based on multiples of 10

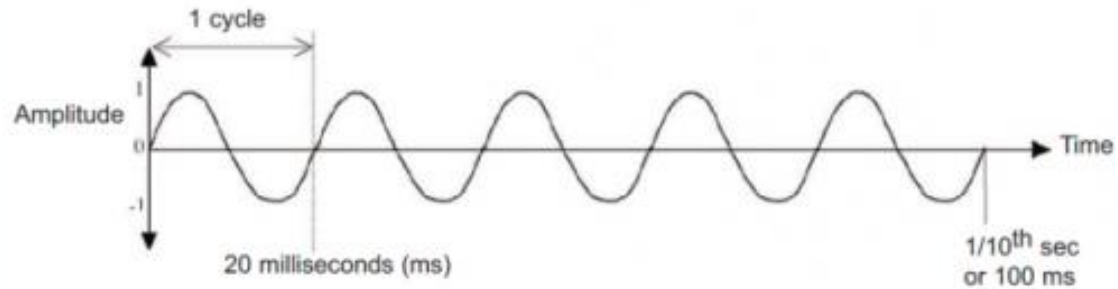
$$dB = 10 \log \frac{I}{I_0}$$

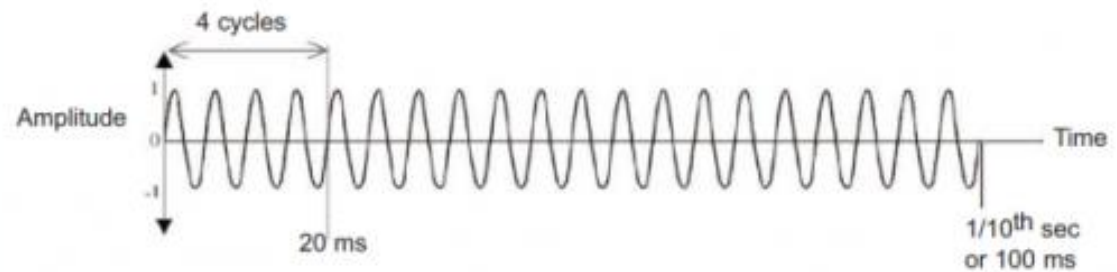
I_0 is reference level = 10^{-12} W/m^2

Frequency and pitch

- Pitch depends primarily (approximately) logarithmically on frequency
  - Pitch: Perceptual property
  - Frequency: Physical property
    - An expression of how frequently a periodic wave form or signal repeats itself at a given amplitude

$$f = \frac{1}{T}$$





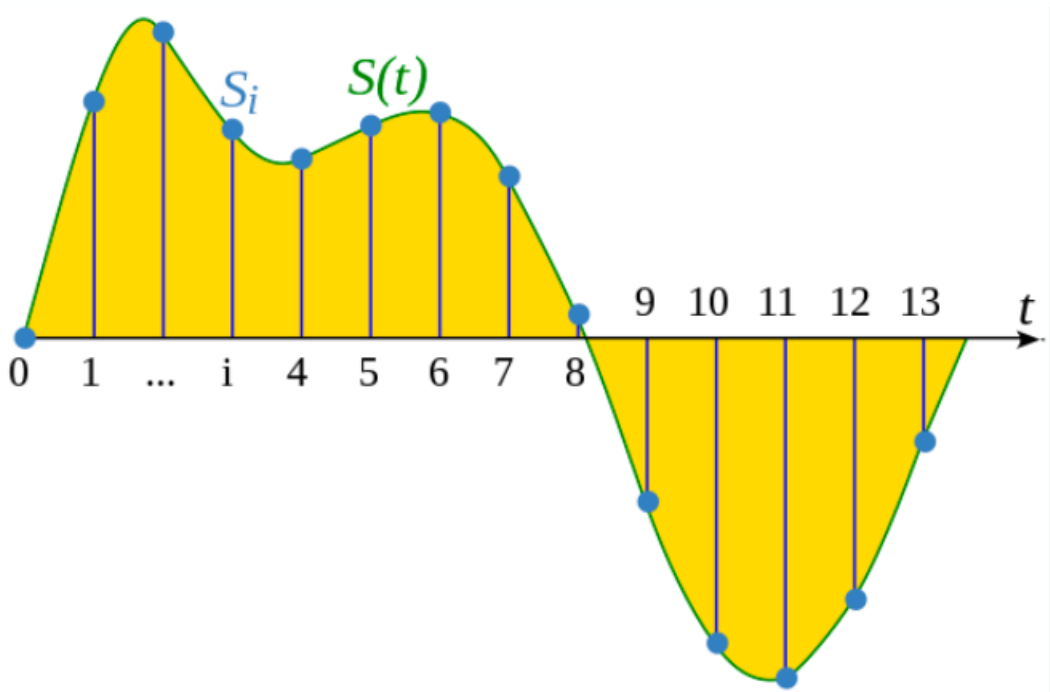
Spectral Analysis in Speech

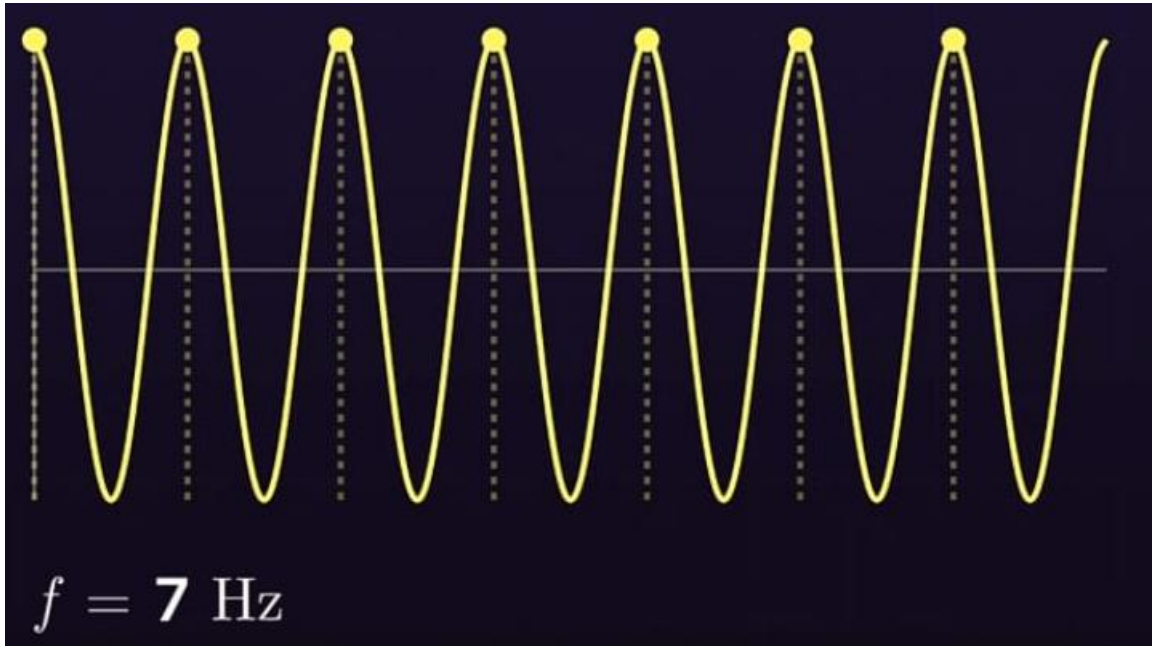
The core goal of spectral analysis in speech is to separate the source (vocal cords/pitch) from the filter (vocal tract/formants).

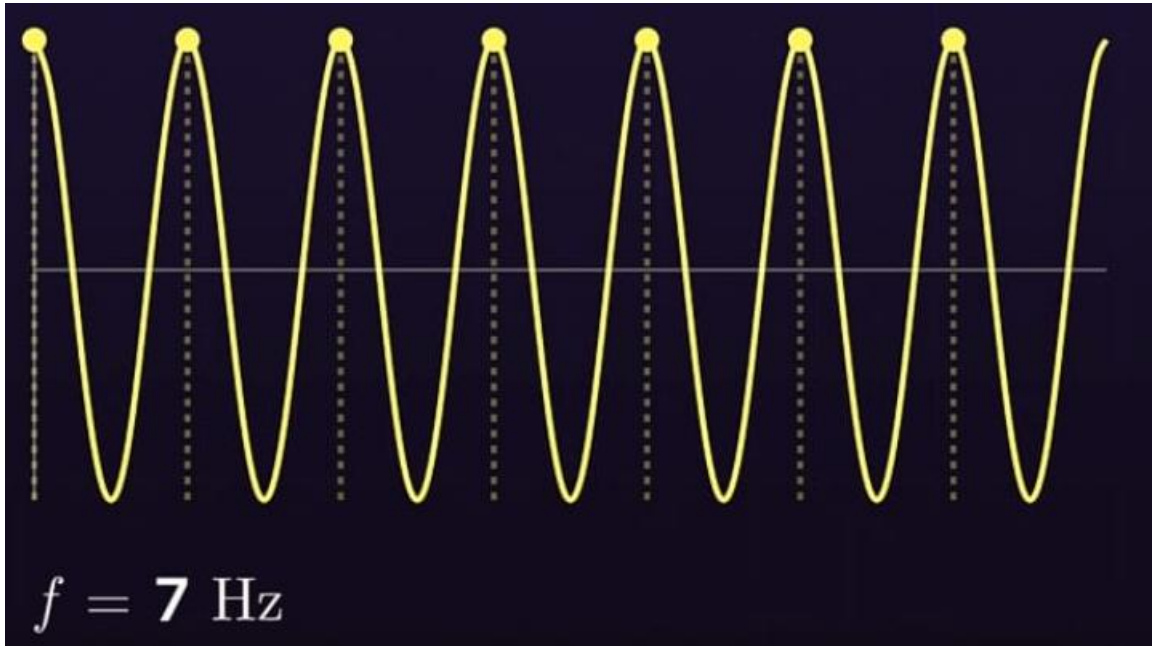
- **Source-Filter Theory**: Speech is modeled as a source signal (e.g., glottal pulses) passing through a filter (the throat and mouth).
- **Cepstrum Analysis**: By taking the `Inverse DFT (iDFT)` of the `Log-Magnitude Spectrum`, we move to the `Quefrency` domain. Here, the `slowly varying vocal tract shape (low quefrency)` and the `rapidly varying pitch (high quefrency)` become additive and separable.
- **LPC (Linear Predictive Coding)**: An alternative method that fits an "all-pole" model to the spectrum to estimate the spectral envelope (formants) directly.
- **MFCCs**: These are features optimized for human hearing (using the `Mel scale`) and are the standard for speech recognition.

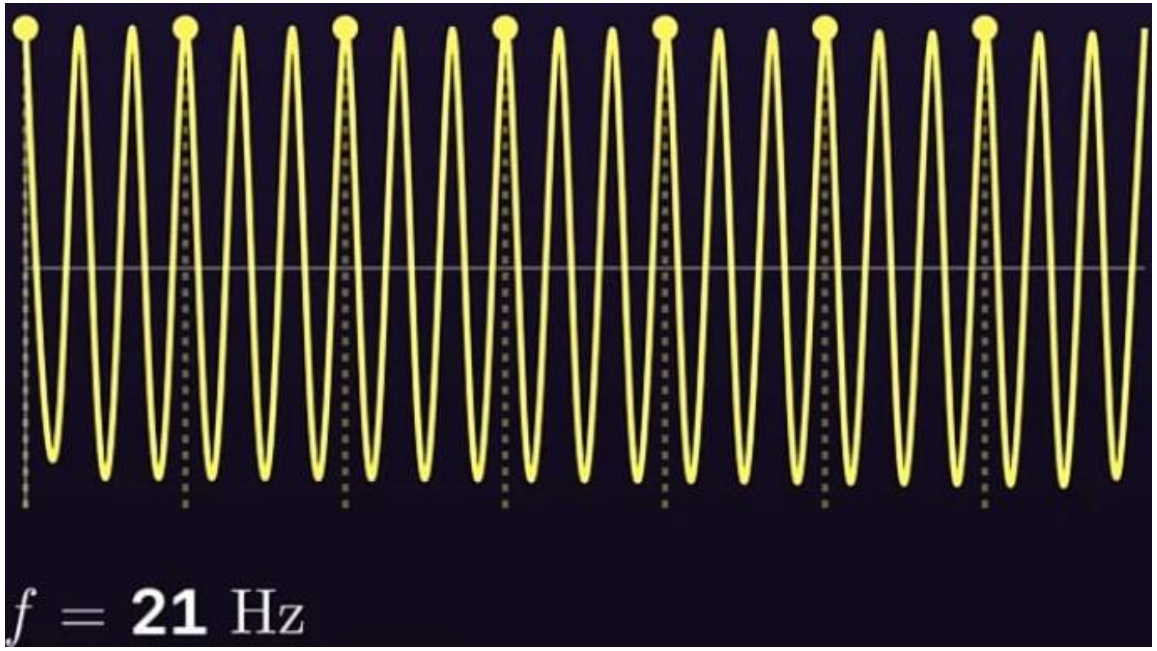
Analog signal to digital signal: Sampling

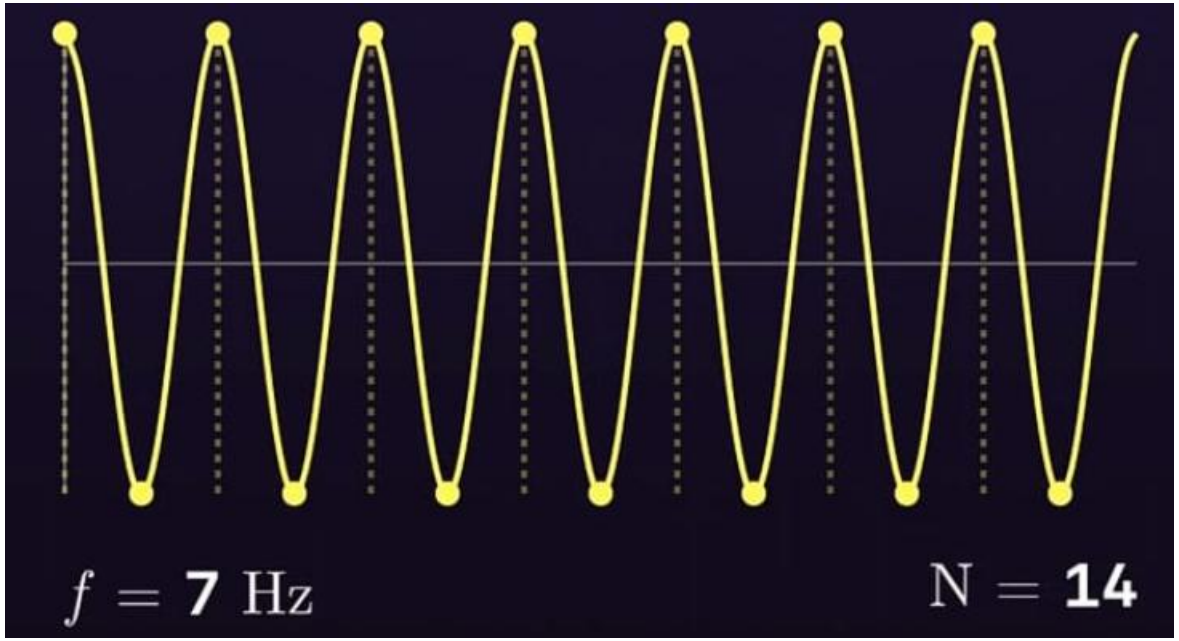
- Sampling period = 1/sampling rate (seconds)

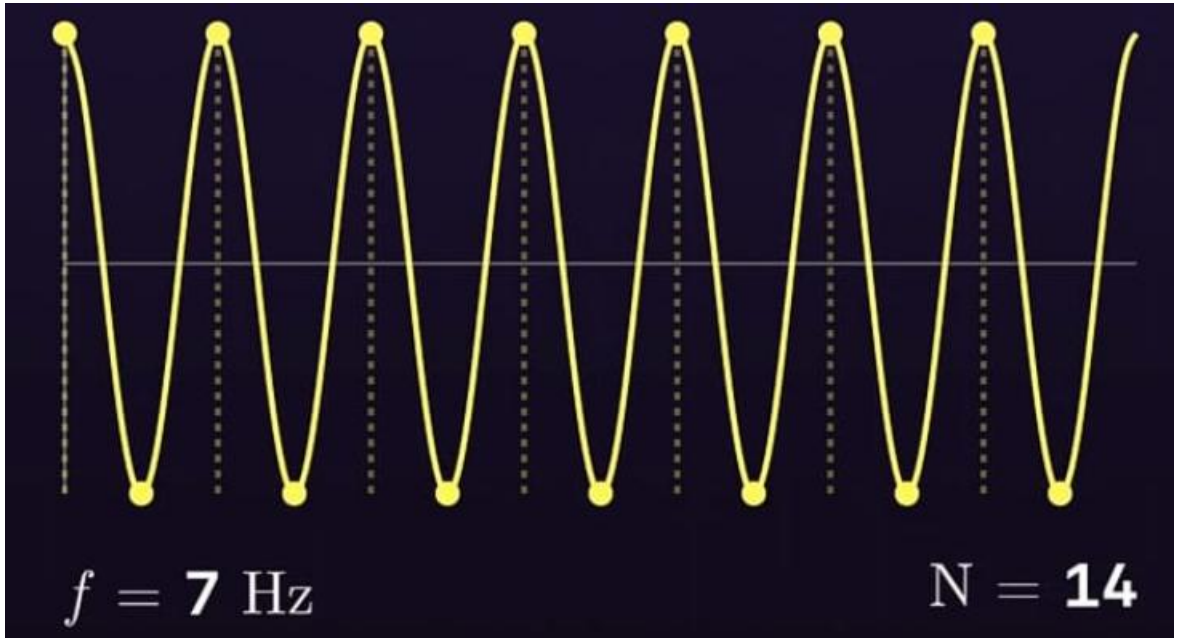


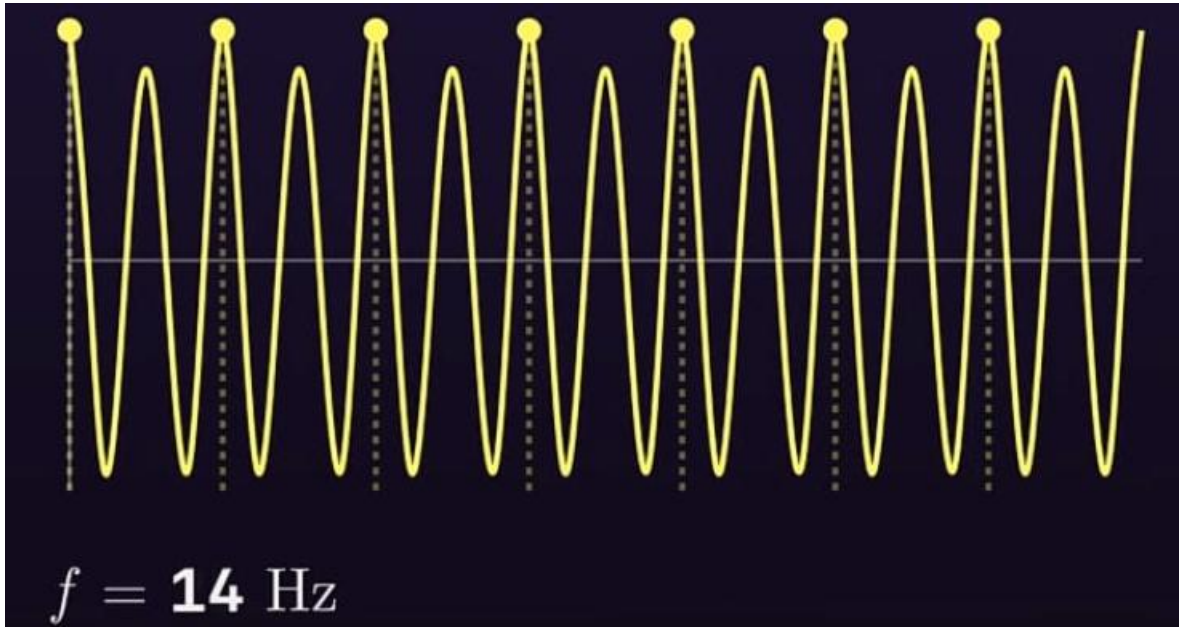

$f = 7$ Hz


$f = 7$ Hz


$f = 21$ Hz

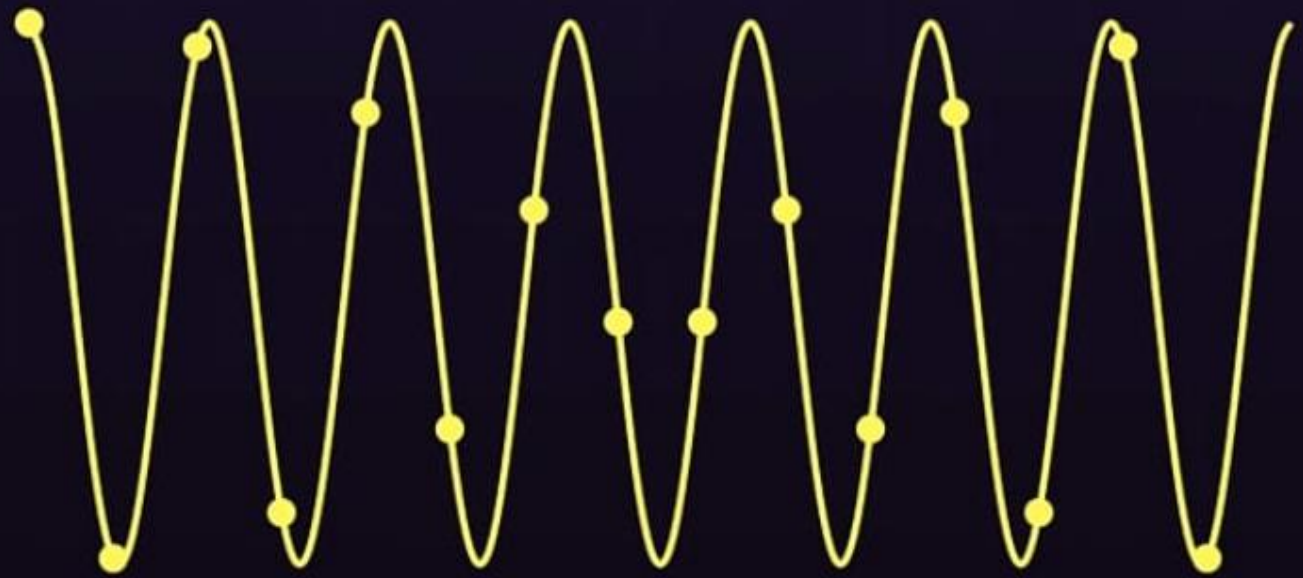

$f = 7$ Hz
$N = 14$

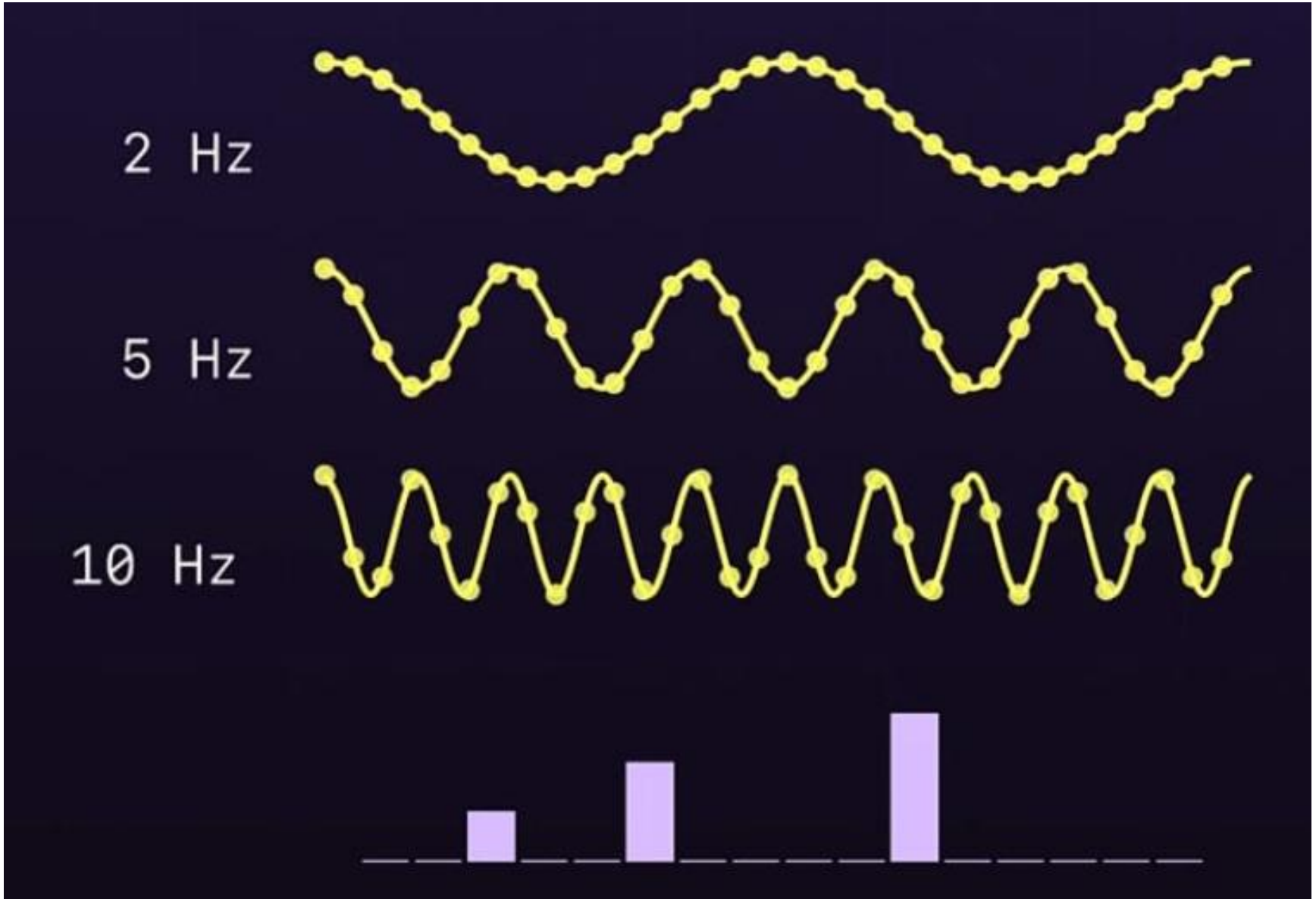

$f = 7$ Hz
N = 14

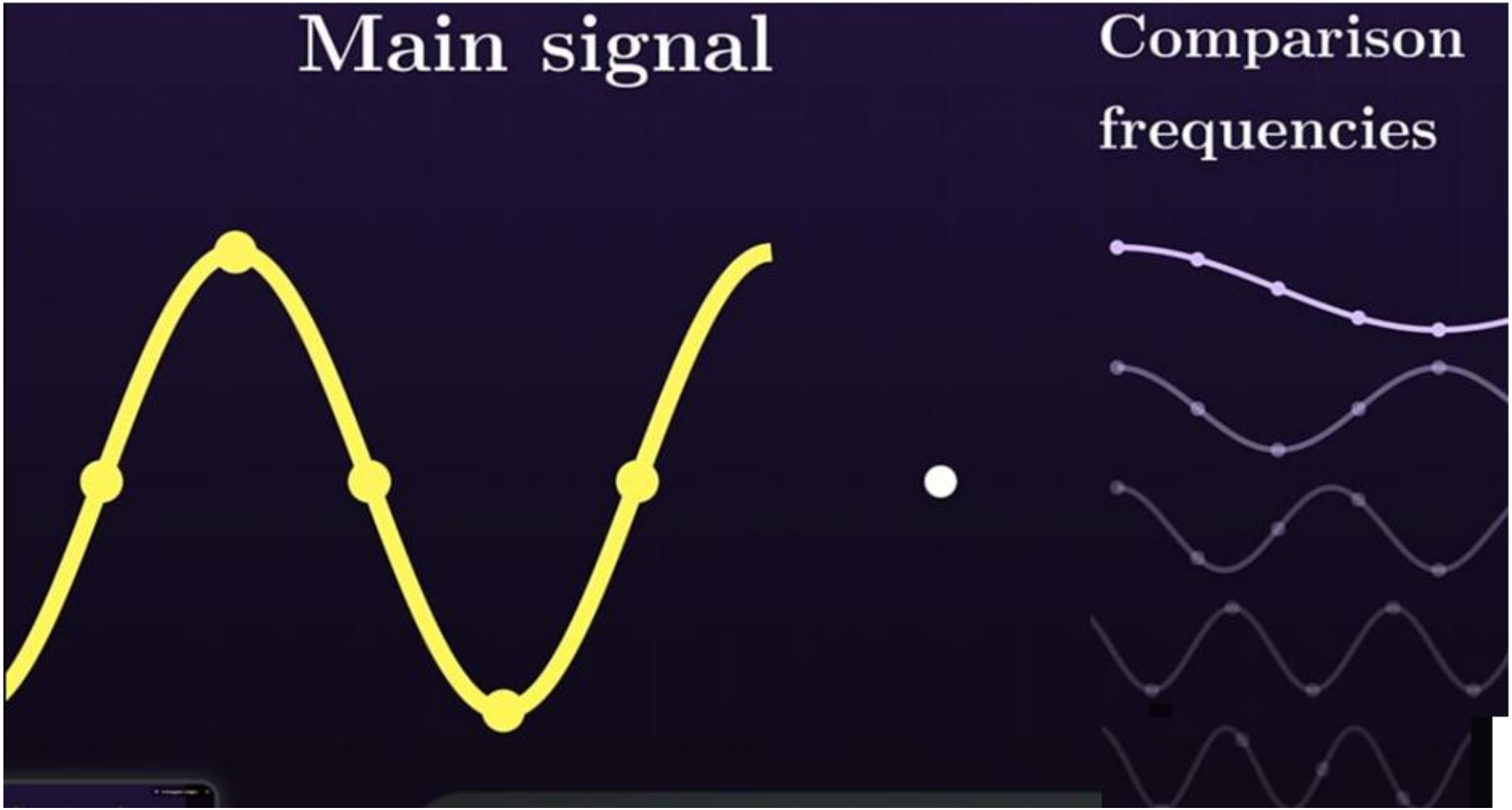

$f = 14$ Hz

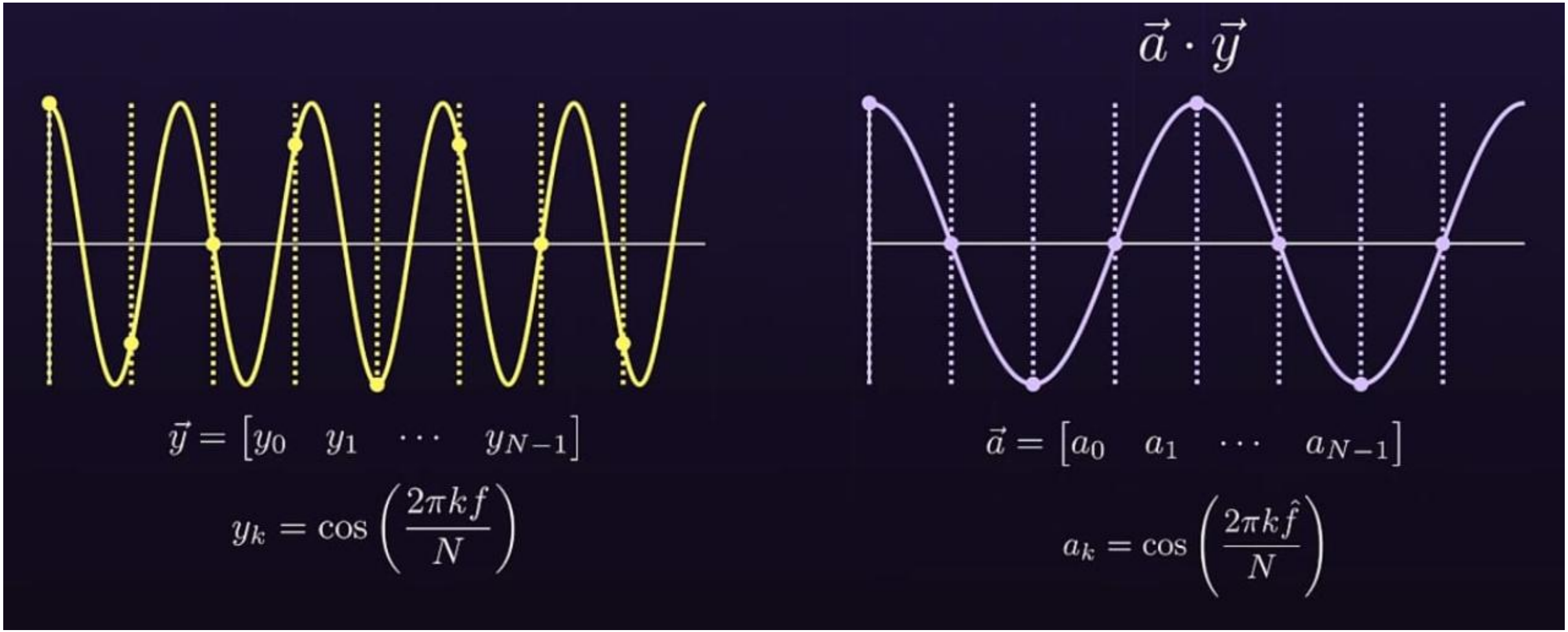
Shannon-Nyquist Theorem

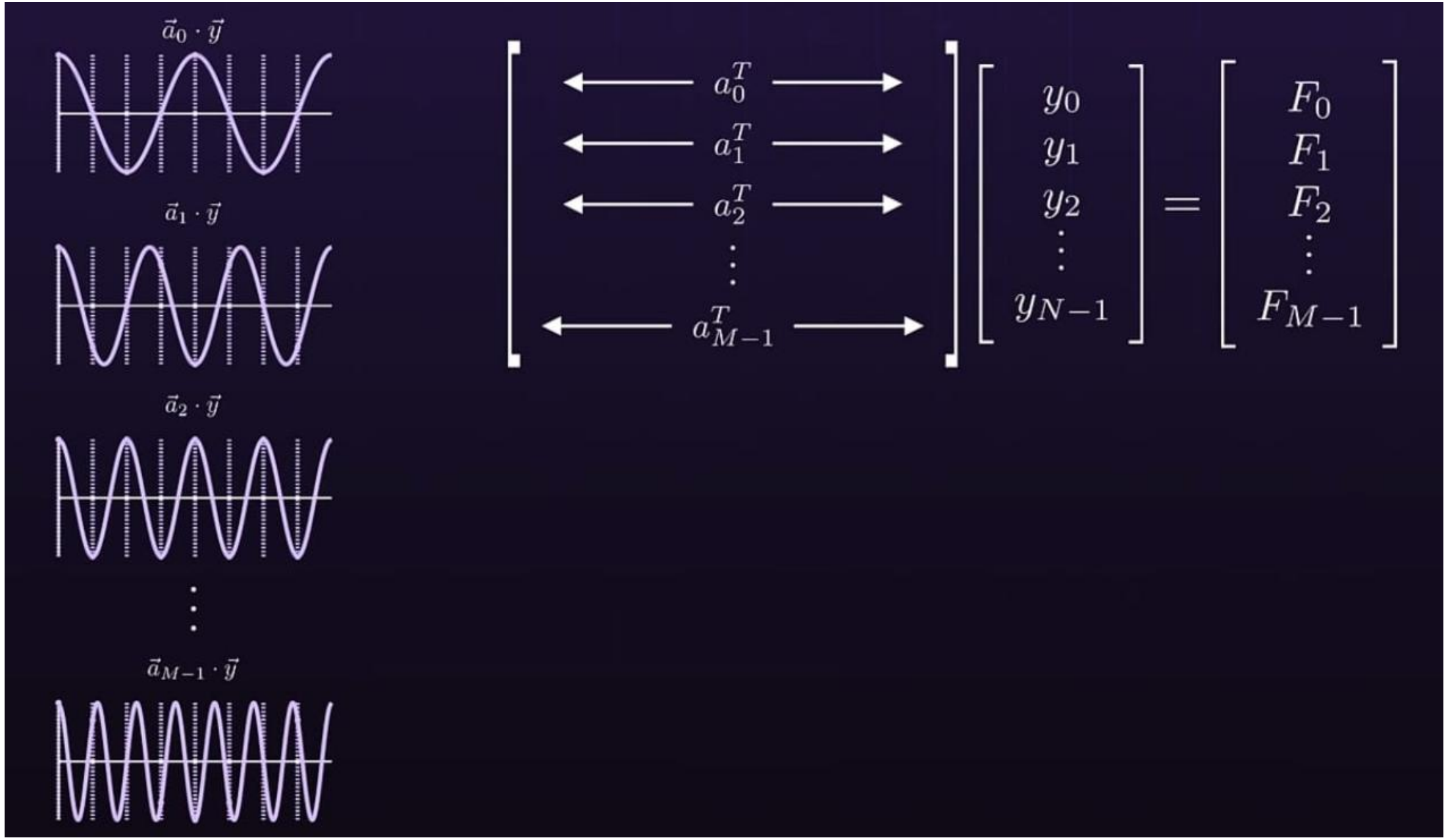
Given $f \Rightarrow N > 2 \cdot f$

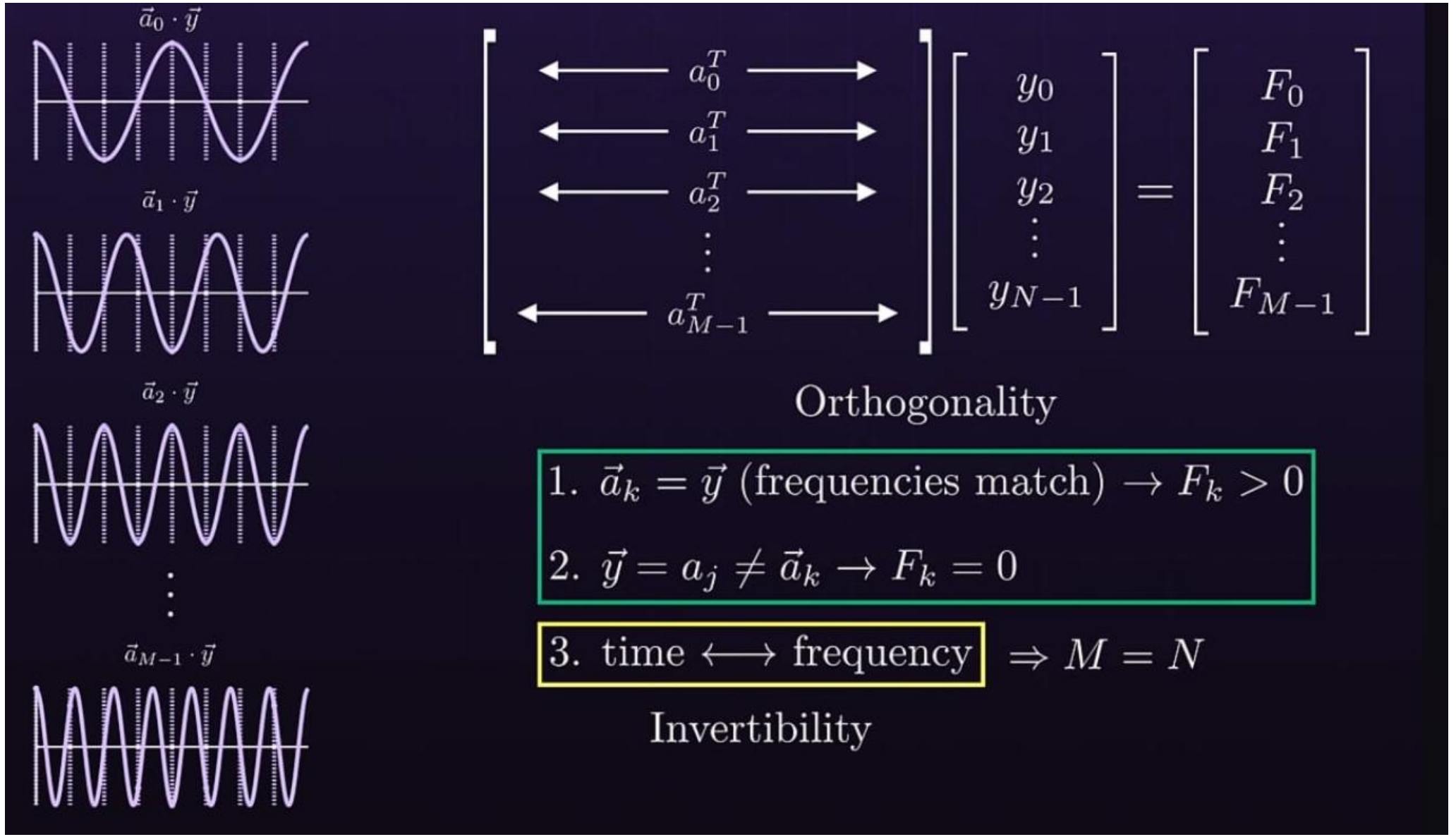


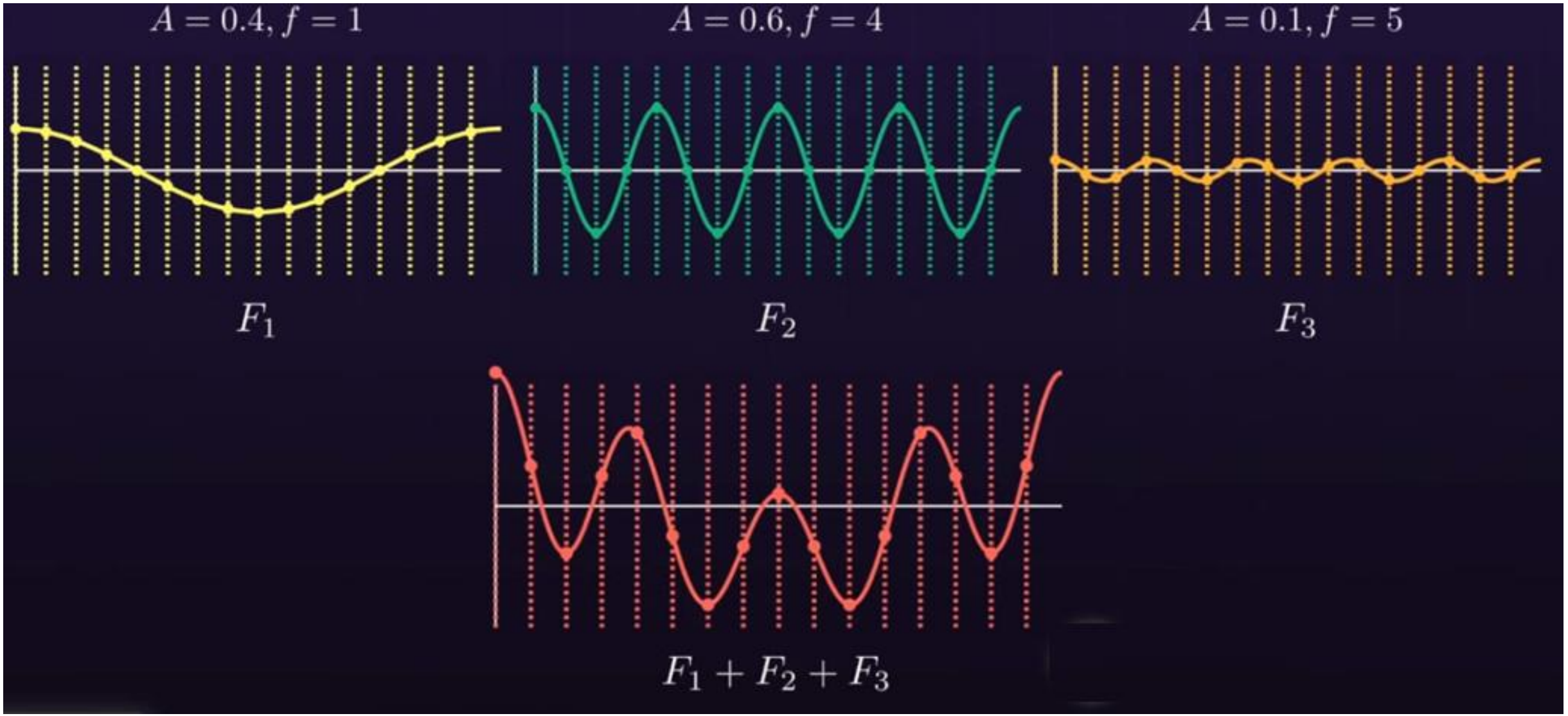

2 Hz
5 Hz
10 Hz

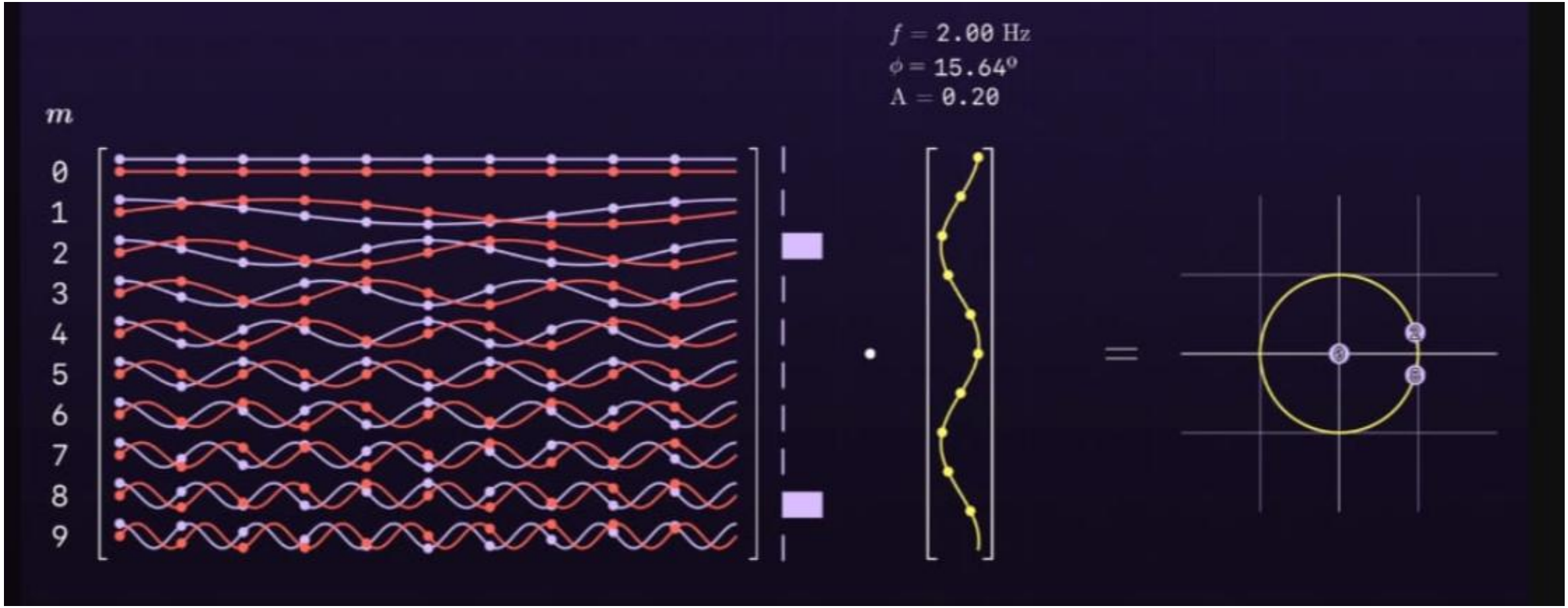

Main signal
Comparison frequencies


$\vec{a} \cdot \vec{y}$
$\vec{y} = [y_0 \quad y_1 \quad \cdots \quad y_{N-1}]$
$\vec{a} = [a_0 \quad a_1 \quad \cdots \quad a_{N-1}]$
$y_k = \cos\left(\frac{2\pi k f}{N}\right)$
$a_k = \cos\left(\frac{2\pi k \hat{f}}{N}\right)$


$\vec{a}_0 \cdot \vec{y}$
$\vec{a}_1 \cdot \vec{y}$
$\vec{a}_2 \cdot \vec{y}$
$\vec{a}_{M-1} \cdot \vec{y}$
a_0^T
a_1^T
a_2^T
a_{M-1}^T
y_0
y_1
y_2
y_{N-1}
$=$
F_0
F_1
F_2
F_{M-1}


$\vec{a}_0 \cdot \vec{y}$
$\vec{a}_1 \cdot \vec{y}$
$\vec{a}_2 \cdot \vec{y}$
$\vec{a}_{M-1} \cdot \vec{y}$
$\begin{bmatrix} \longleftarrow a_0^T \longrightarrow \\ \longleftarrow a_1^T \longrightarrow \\ \longleftarrow a_2^T \longrightarrow \\ \vdots \\ \longleftarrow a_{M-1}^T \longrightarrow \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ F_{M-1} \end{bmatrix}$
Orthogonality
1. $\vec{a}_k = \vec{y}$ (frequencies match) $\rightarrow F_k > 0$
2. $\vec{y} = a_j \neq \vec{a}_k \rightarrow F_k = 0$
3. time $\longleftrightarrow$ frequency $\Rightarrow M = N$
Invertibility

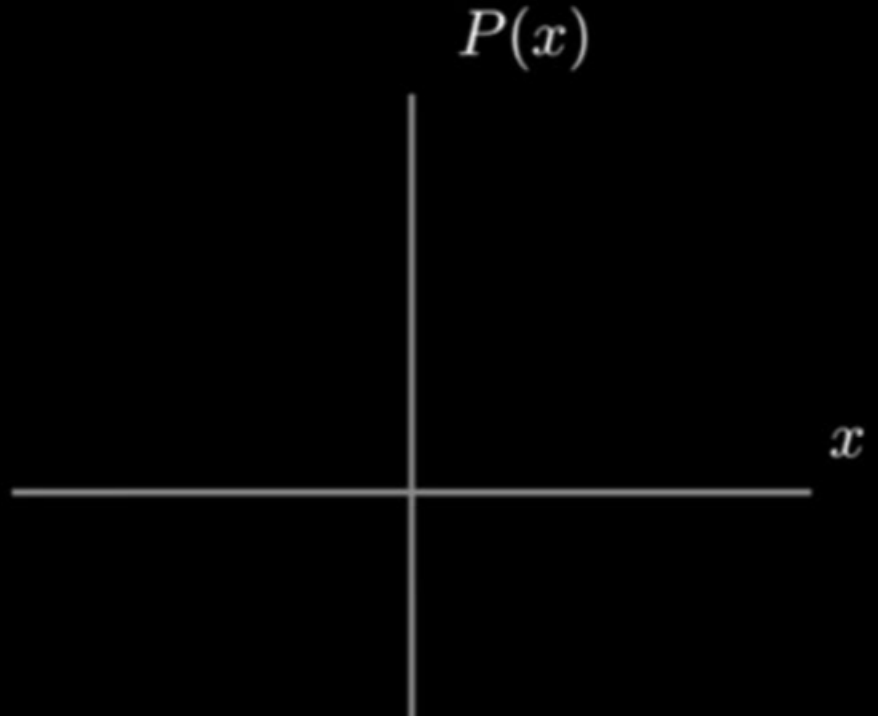

$A = 0.4, f = 1$
$A = 0.6, f = 4$
$A = 0.1, f = 5$
F_1
F_2
F_3
$F_1 + F_2 + F_3$

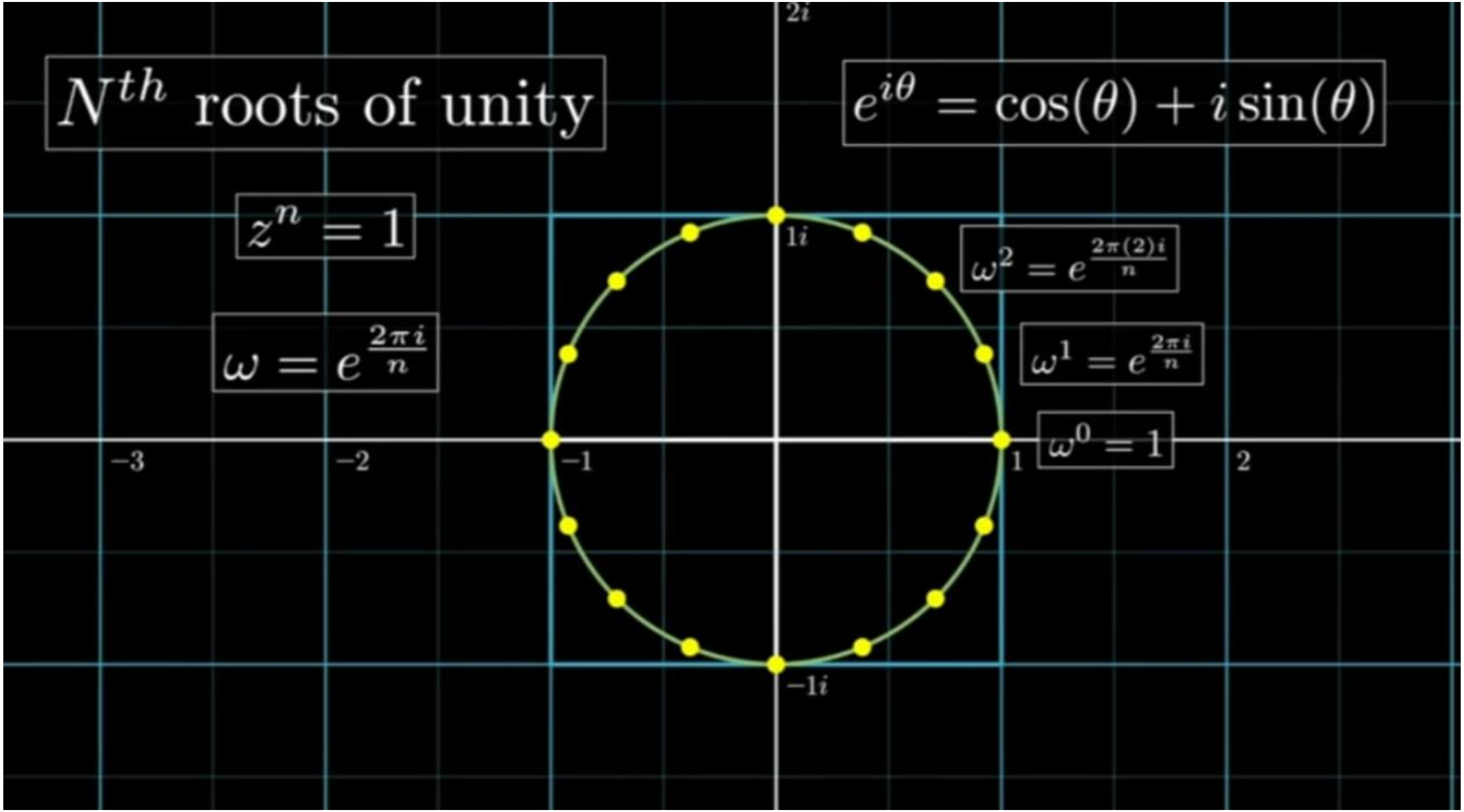

$f = 2.00$ Hz
$\phi = 15.64^{o}$
A = 0.20
m
0
1
2
3
4
5
6
7
8
9

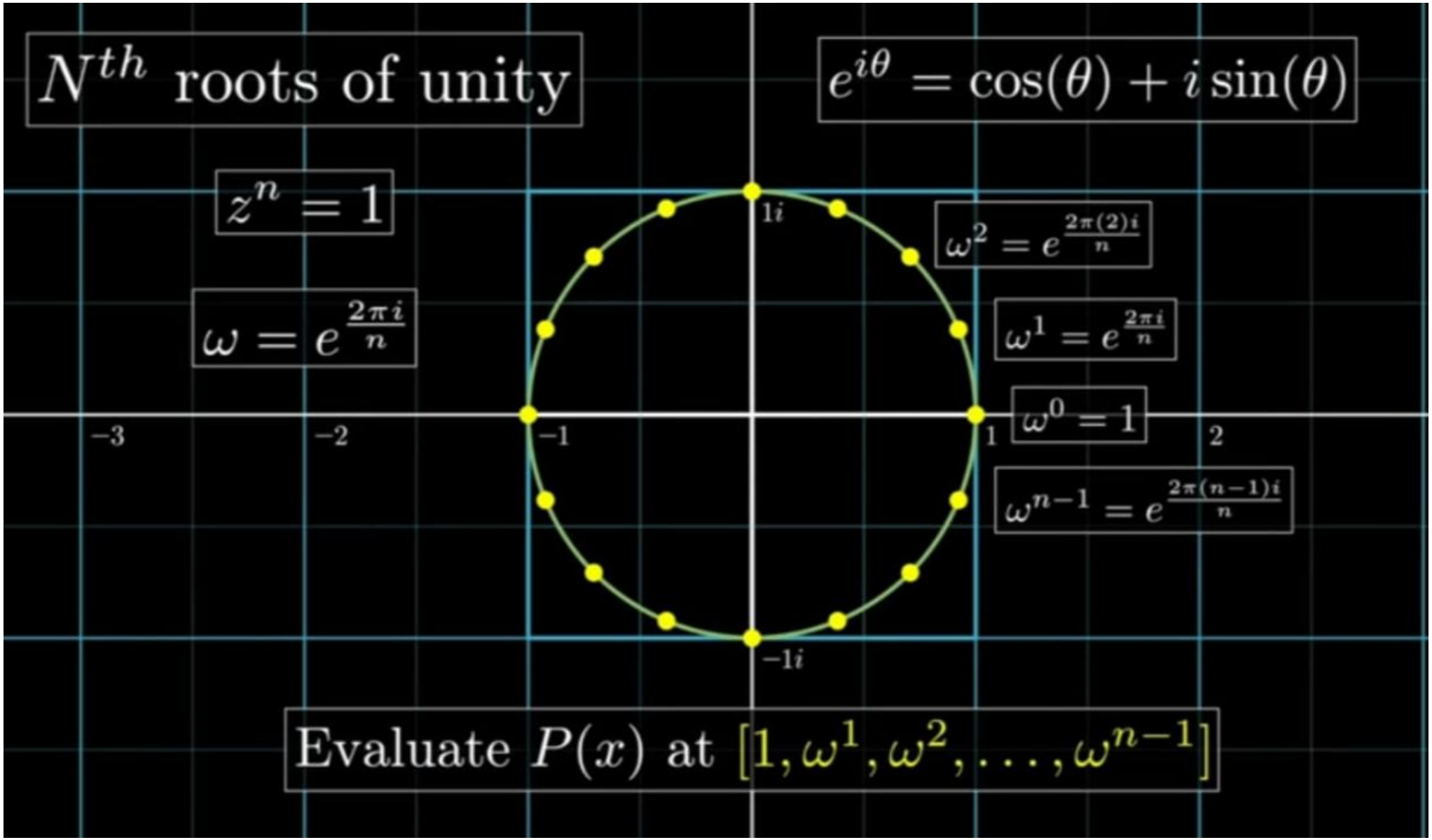
Evaluation

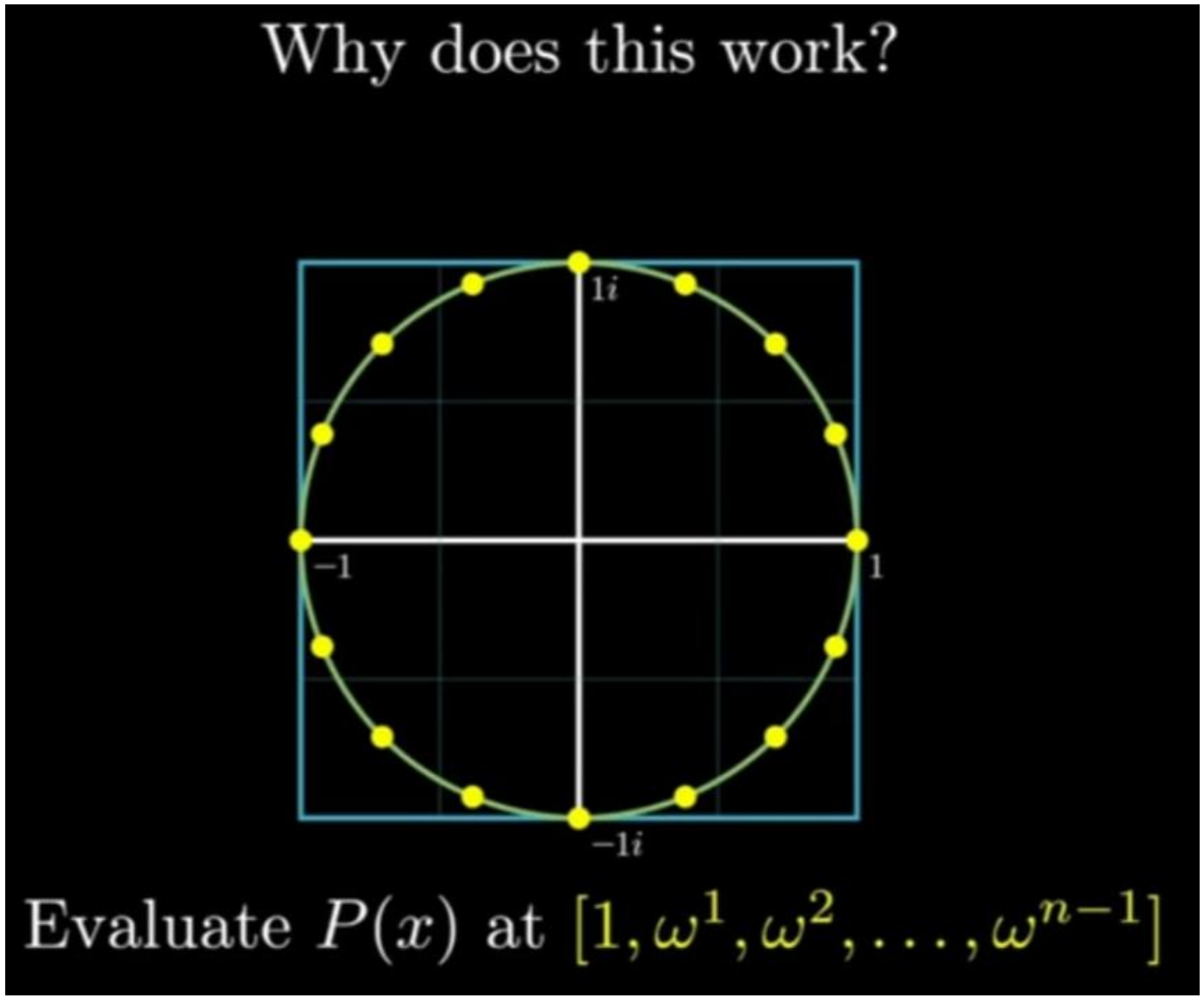
$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d$$

Evaluate at $n \geq d + 1$ points




N^{th} roots of unity
$e^{i\theta} = \cos(\theta) + i\sin(\theta)$
$z^n = 1$
$\omega = e^{\frac{2\pi i}{n}}$
$\omega^2 = e^{\frac{2\pi(2)i}{n}}$
$\omega^1 = e^{\frac{2\pi i}{n}}$
$\omega^0 = 1$
2i
1i
−1i
−3
−2
−1
1
2


N^{th} roots of unity
$e^{i\theta} = \cos(\theta) + i\sin(\theta)$
$z^n = 1$
$\omega = e^{\frac{2\pi i}{n}}$
$\omega^2 = e^{\frac{2\pi(2)i}{n}}$
$\omega^1 = e^{\frac{2\pi i}{n}}$
$\omega^0 = 1$
$\omega^{n-1} = e^{\frac{2\pi(n-1)i}{n}}$
1i
−1i
−3
−2
−1
1
2
Evaluate $P(x)$ at $[1, \omega^1, \omega^2, \ldots, \omega^{n-1}]$


Why does this work?
$1i$
-1
1
$-1i$
Evaluate $P(x)$ at $[1, \omega^1, \omega^2, \ldots, \omega^{n-1}]$

IFFT(<values>) $\Leftrightarrow$ FFT(<values>) with $\omega = \frac{1}{n} e^{\frac{-2\pi i}{n}}$

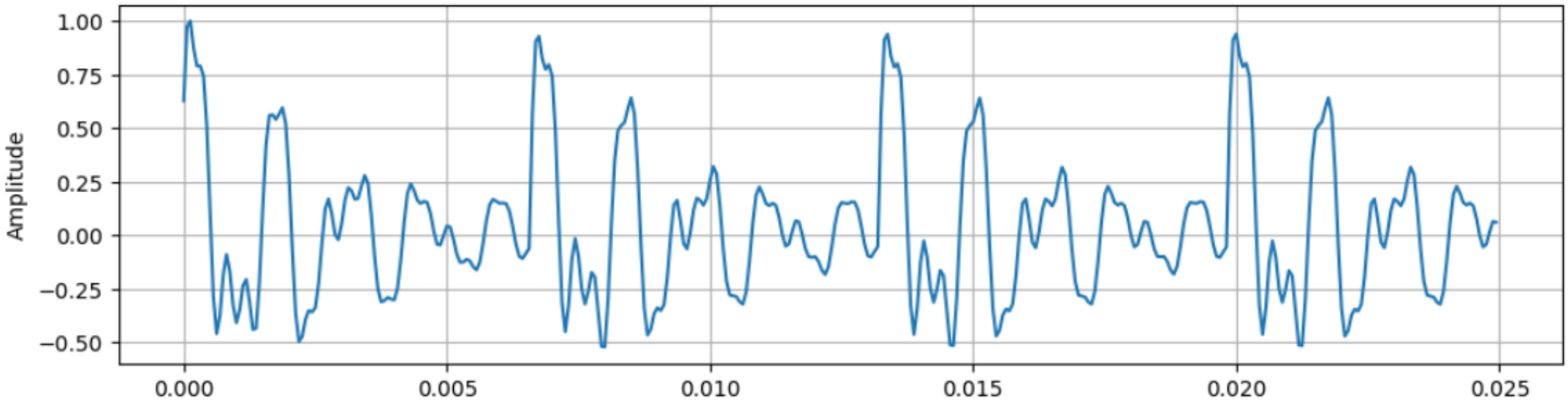
```
def FFT(P):
    # P - [p_0, p_1, ..., p_{n-1}] coeff rep
    n = len(P) # n is a power of 2
    if n == 1:
        return P
    ω = e^(2πi/n)
    P_e, P_o = P[::2], P[1::2]
    y_e, y_o = FFT(P_e), FFT(P_o)
    y = [0] * n
    for j in range(n/2):
        y[j] = y_e[j] + ω^j y_o[j]
        y[j + n/2] = y_e[j] - ω^j y_o[j]
    return y
```

```
def IFFT(P):
    # P - [P(ω^0), P(ω^1), ..., P(ω^{n-1})] value rep
    n = len(P) # n is a power of 2
    if n == 1:
        return P
    ω = (1/n) * e^(-2πi/n)
    P_e, P_o = P[::2], P[1::2]
    y_e, y_o = IFFT(P_e), IFFT(P_o)
    y = [0] * n
    for j in range(n/2):
        y[j] = y_e[j] + ω^j y_o[j]
        y[j + n/2] = y_e[j] - ω^j y_o[j]
    return y
```

Spectral Analysis in Speech

The core goal of spectral analysis in speech is to separate the source (vocal cords/pitch) from the filter (vocal tract/formants).

- **Source-Filter Theory**: Speech is modeled as a source signal (e.g., glottal pulses) passing through a filter (the throat and mouth).
- **Cepstrum Analysis**: By taking the `Inverse DFT (iDFT)` of the `Log-Magnitude Spectrum`, we move to the `Quefrency` domain. Here, the `slowly varying vocal tract shape (low quefrency)` and the `rapidly varying pitch (high quefrency)` become additive and separable.
- **LPC (Linear Predictive Coding)**: An alternative method that fits an "all-pole" model to the spectrum to estimate the spectral envelope (formants) directly.
- **MFCCs**: These are features optimized for human hearing (using the `Mel scale`) and are the standard for speech recognition.

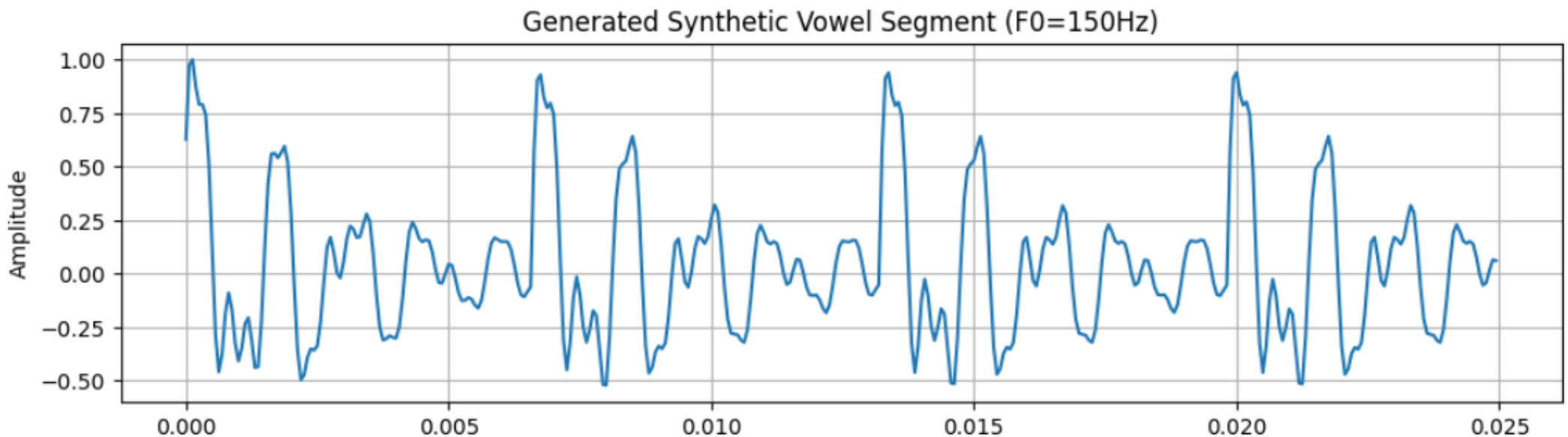


2. Pre-emphasis

Explanation:

- Speech typically has a `spectral tilt` where **high frequencies have lower energy**.
- `Pre-emphasis` is a `high-pass filter` that **boosts high frequencies** to balance the spectrum for better analysis.

Formula: $y[n] = x[n] - \alpha x[n-1]$ (typically $\alpha = 0.97$)

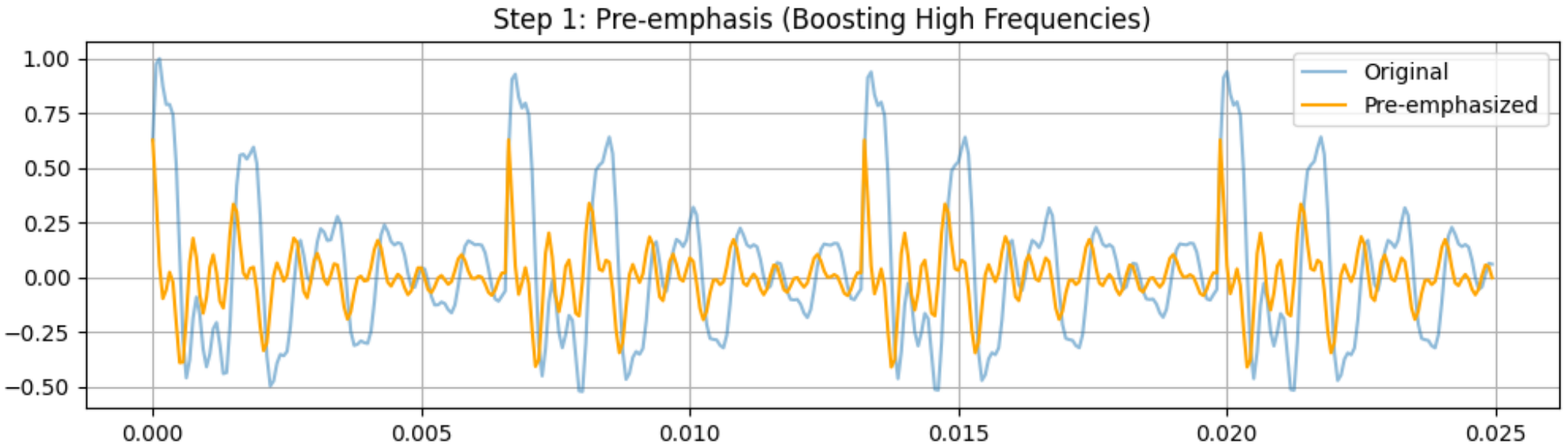


2. Pre-emphasis

Explanation:

- Speech typically has a `spectral tilt` where **high frequencies have lower energy**.
- `Pre-emphasis` is a `high-pass filter` that **boosts high frequencies** to balance the spectrum for better analysis.

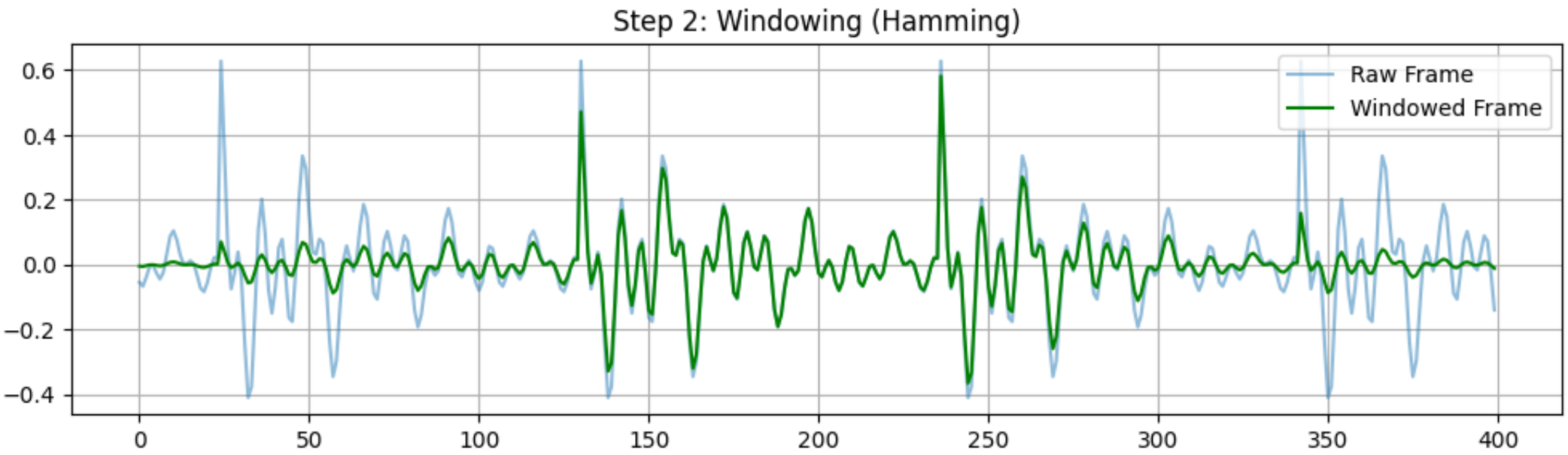
Formula: $y[n] = x[n] - \alpha x[n-1]$ (typically $\alpha = 0.97$)



3. Framing and Windowing

Explanation:

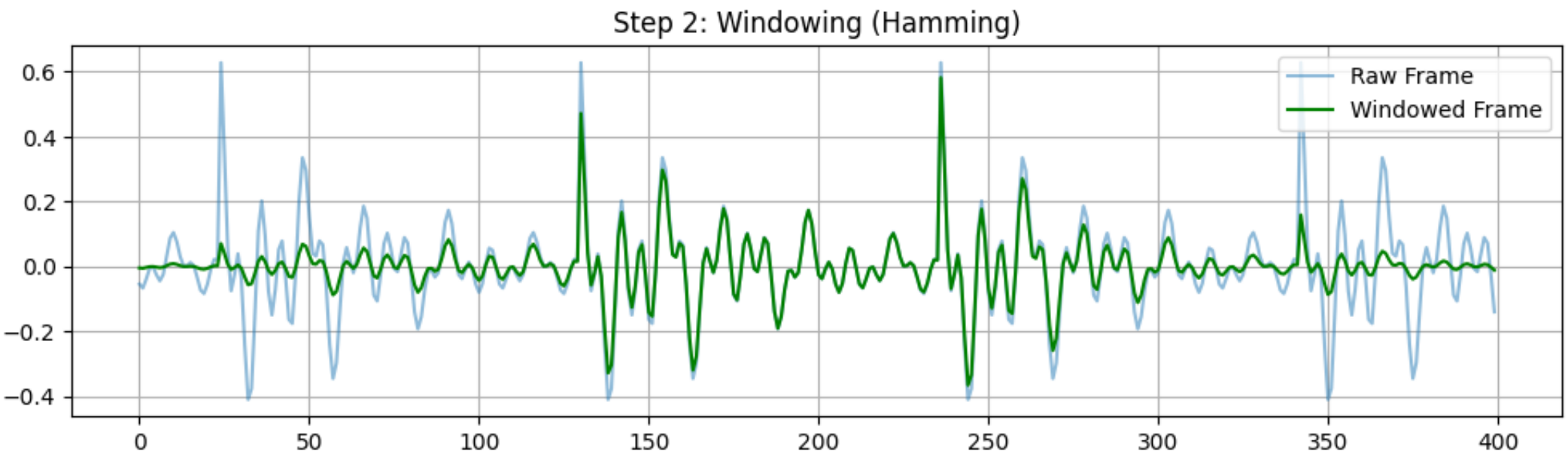
- Speech is **non-stationary** (changes over time).
- We **break it into short frames** (e.g., 25ms) where it is statistically stationary.
- We apply a `Hamming Window` to taper the edges to zero, reducing `spectral leakage` during FFT.



4. FFT (Fast Fourier Transform)

Explanation:

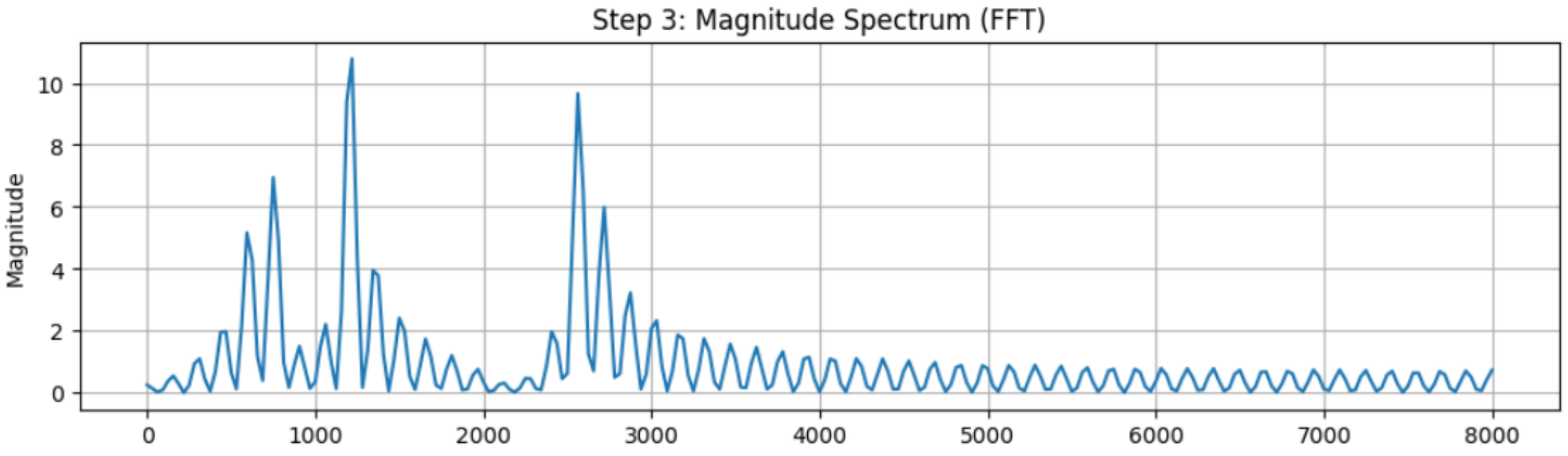
- Converts the **time-domain signal** into the **frequency domain**.
- We compute the **Magnitude Spectrum** to see how much energy is present at each frequency.



4. FFT (Fast Fourier Transform)

Explanation:

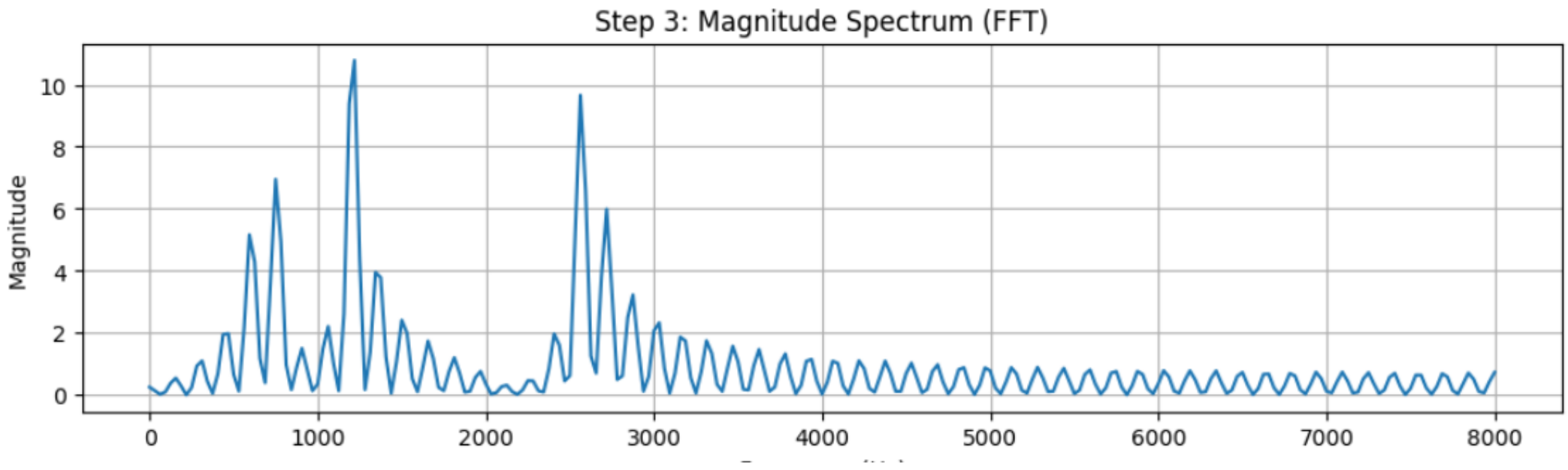
- Converts the **time-domain signal** into the **frequency domain**.
- We compute the **Magnitude Spectrum** to see how much energy is present at each frequency.



5. Apply Mel-Filterbank

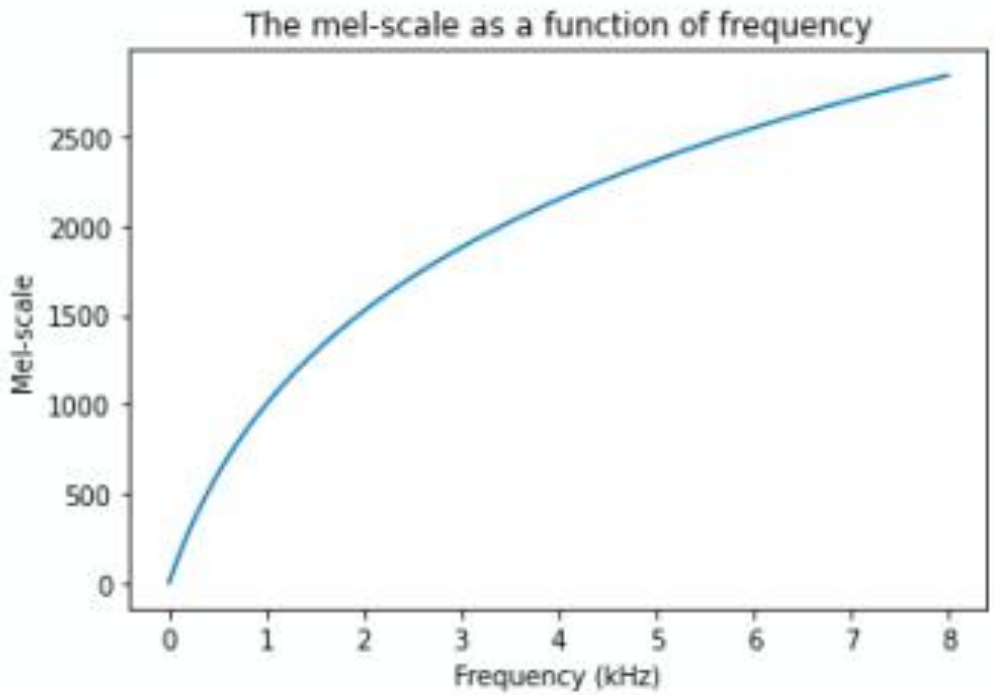
Explanation:

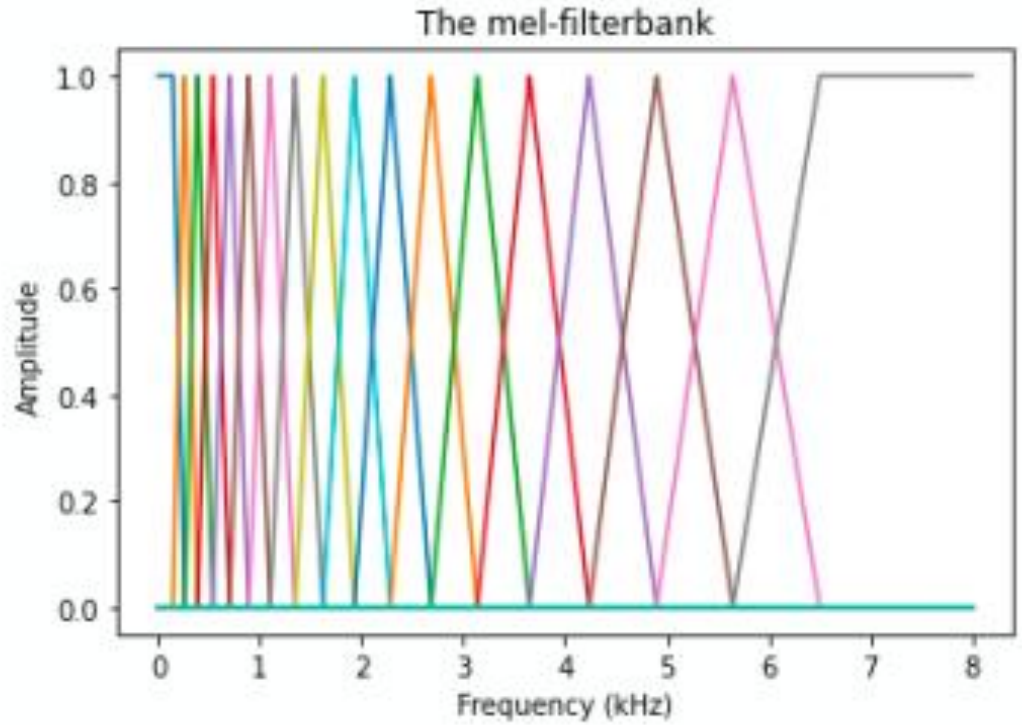
- Human hearing is not linear;
  we are **more sensitive to differences in low frequencies** than high frequencies.
- The **Mel Scale** mimics this.
  We multiply the spectrum by a bank of triangular filters
  (narrow at low freq, wide at high freq) to sum up energy in specific bands.



Mel-scale

- A scale that maps frequencies such that steps between tones align with our perception of steps

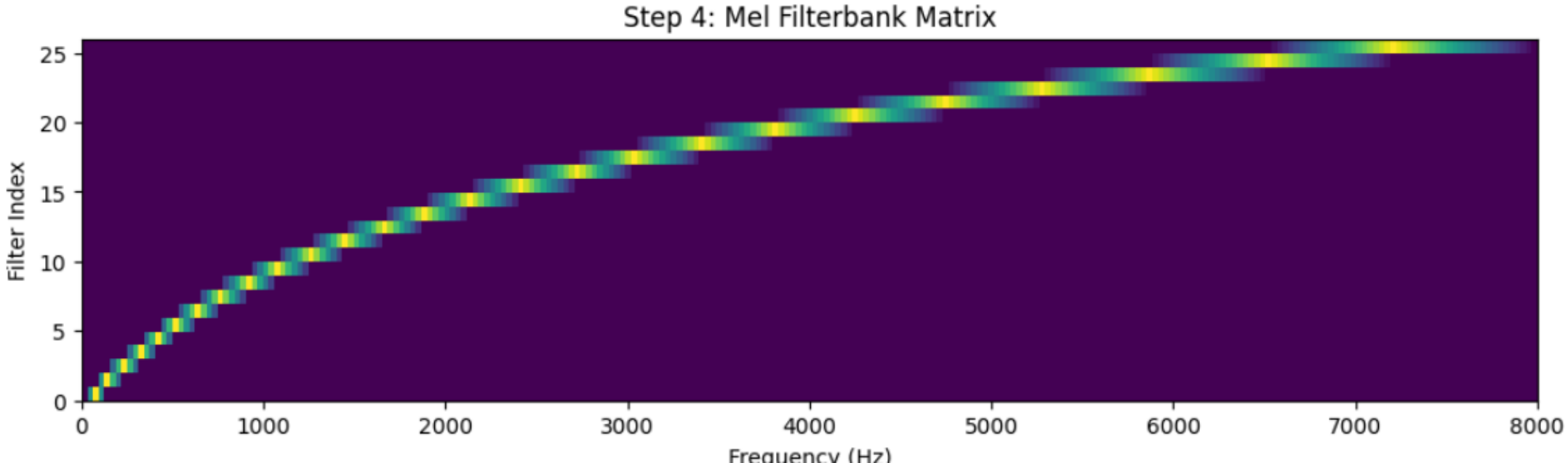




5. Apply Mel-Filterbank

Explanation:

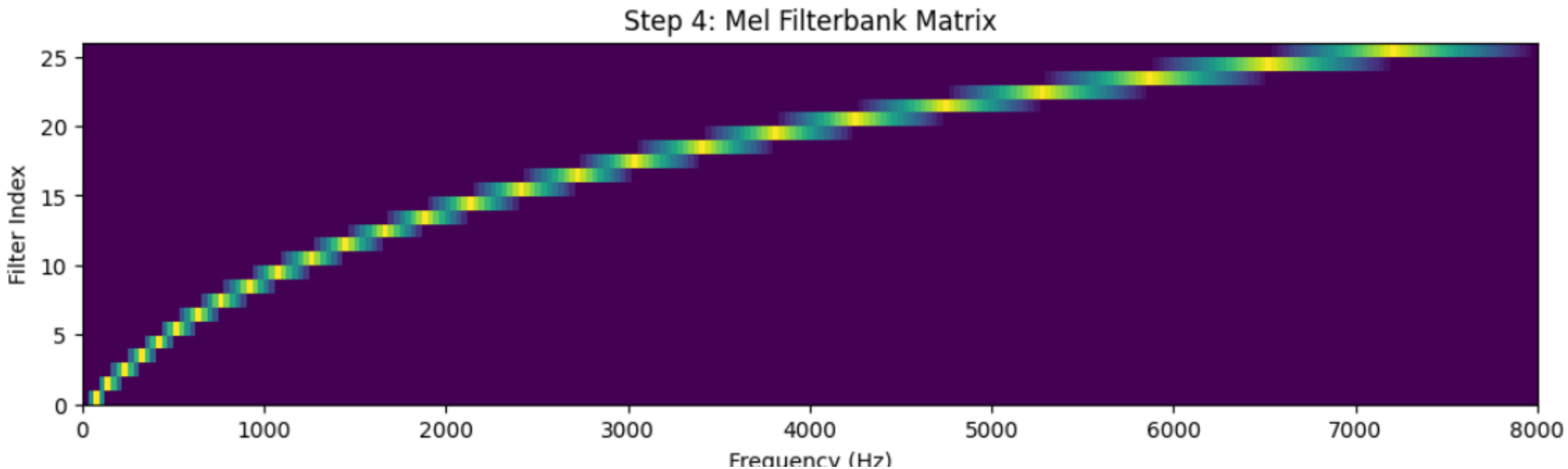
- Human hearing is not linear;
  we are **more sensitive to differences in low frequencies** than high frequencies.
- The **Mel Scale** mimics this.
  We multiply the spectrum by a bank of triangular filters
  (narrow at low freq, wide at high freq) to sum up energy in specific bands.



6. Logarithmic Amplitude

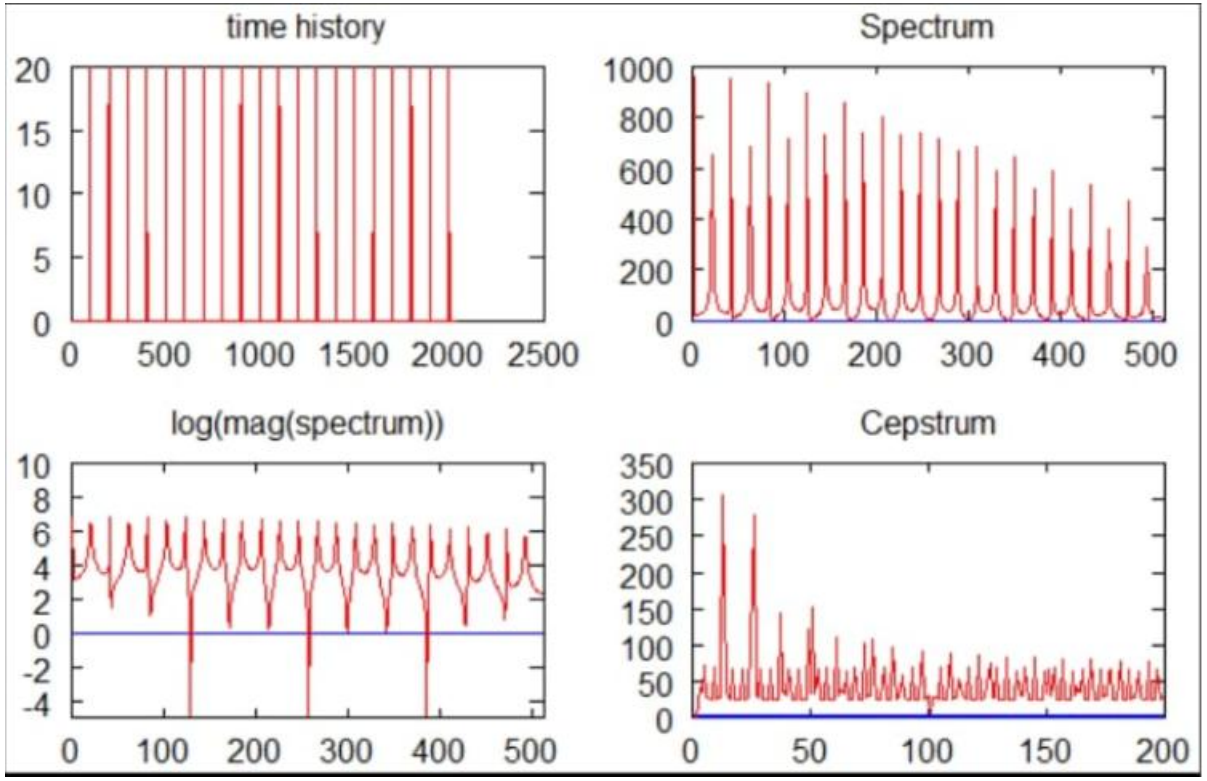
Explanation:

- We take the **logarithm of the filterbank energies**.
- This mimics the human ear's perception of loudness
  (which is logarithmic, measured in decibels)
  and **converts multiplicative noise (channel effects) into additive components**.



Cepstrum

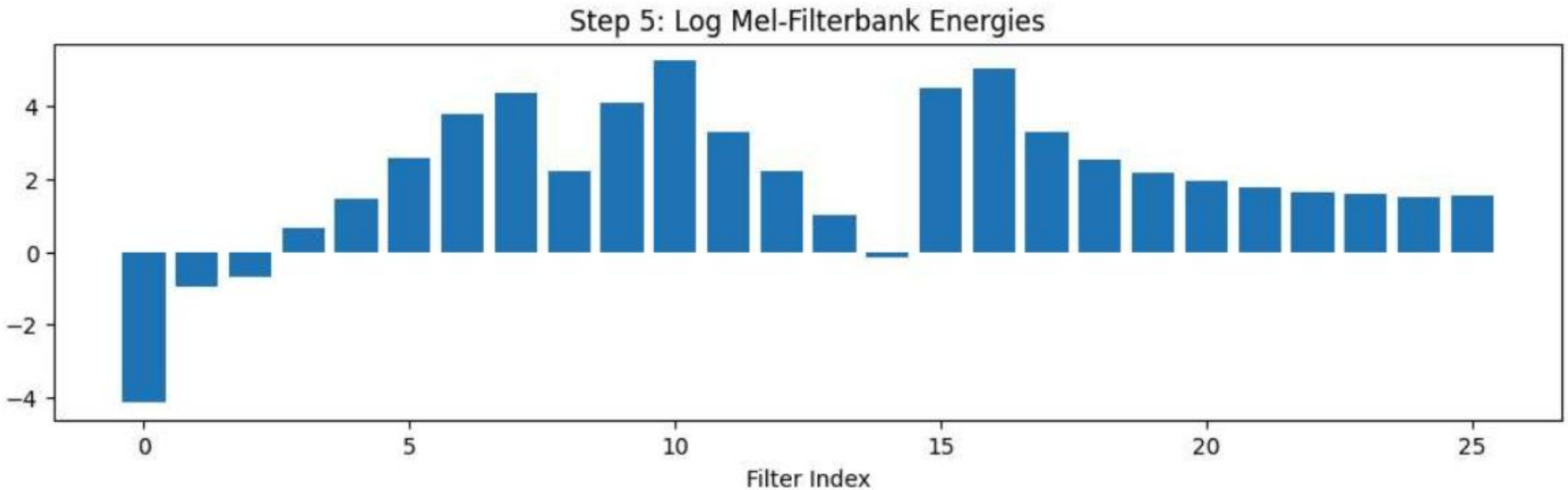
- Fourier analysis, the cepstrum is the result of computing the inverse Fourier transform (IFT) of the logarithm of the estimated signal spectrum.
- The method is a tool for investigating periodic structures in frequency spectra. The power cepstrum has applications in the analysis of human speech.



6. Logarithmic Amplitude

Explanation:

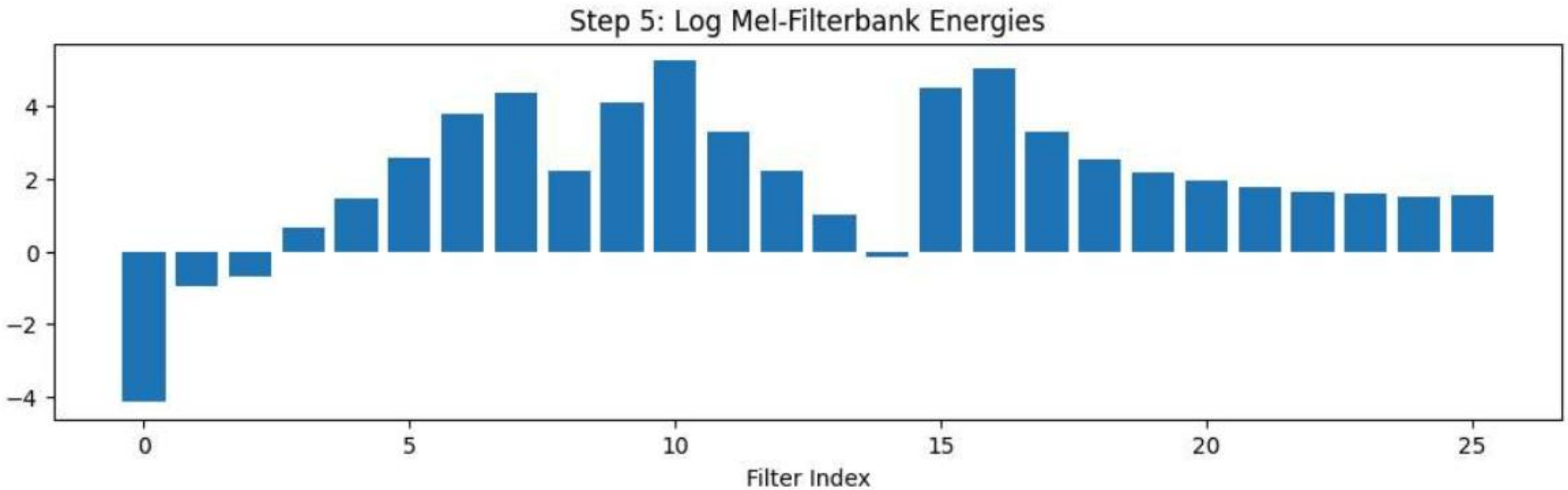
- We take the **logarithm of the filterbank energies**.
- This mimics the human ear's perception of loudness
  (which is logarithmic, measured in decibels)
  and **converts multiplicative noise (channel effects) into additive components**.



7. Discrete Cosine Transform (DCT) - "The MFCCs"

Explanation:

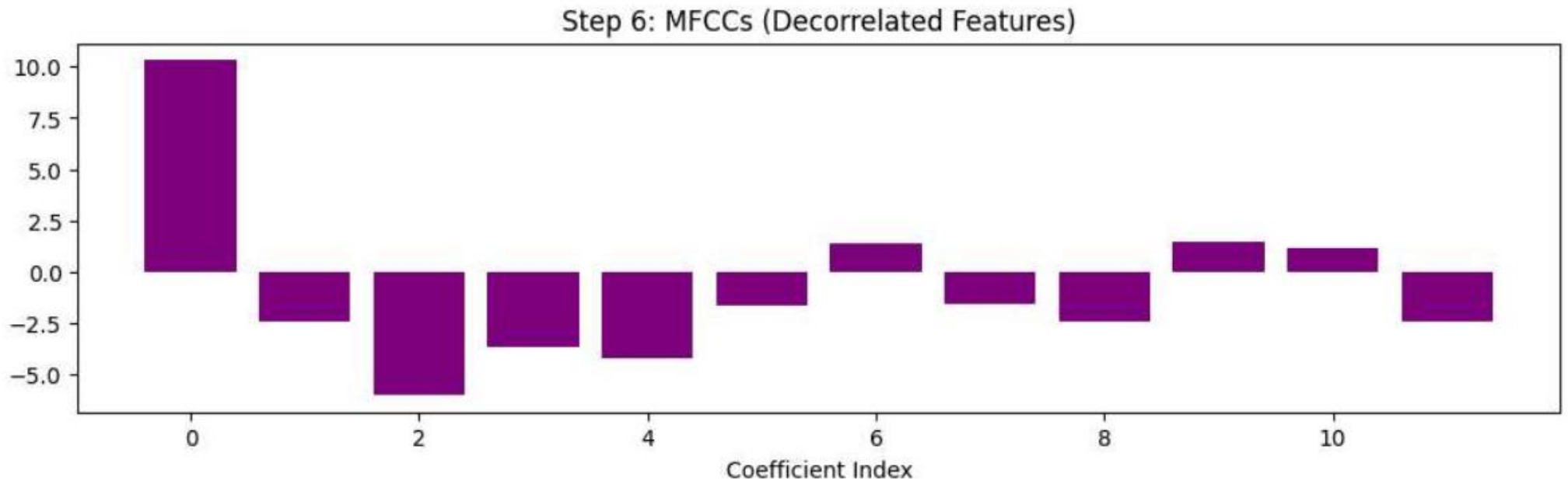
- The **log-filterbank energies** are **highly correlated** (overlapping filters).
- The **DCT decorrelates** them, compressing the information into the first few coefficients. These are the **MFCCs**.



7. Discrete Cosine Transform (DCT) - "The MFCCs"

Explanation:

- The **log-filterbank energies** are **highly correlated** (overlapping filters).
- The **DCT decorrelates** them, compressing the information into the first few coefficients. These are the **MFCCs**.

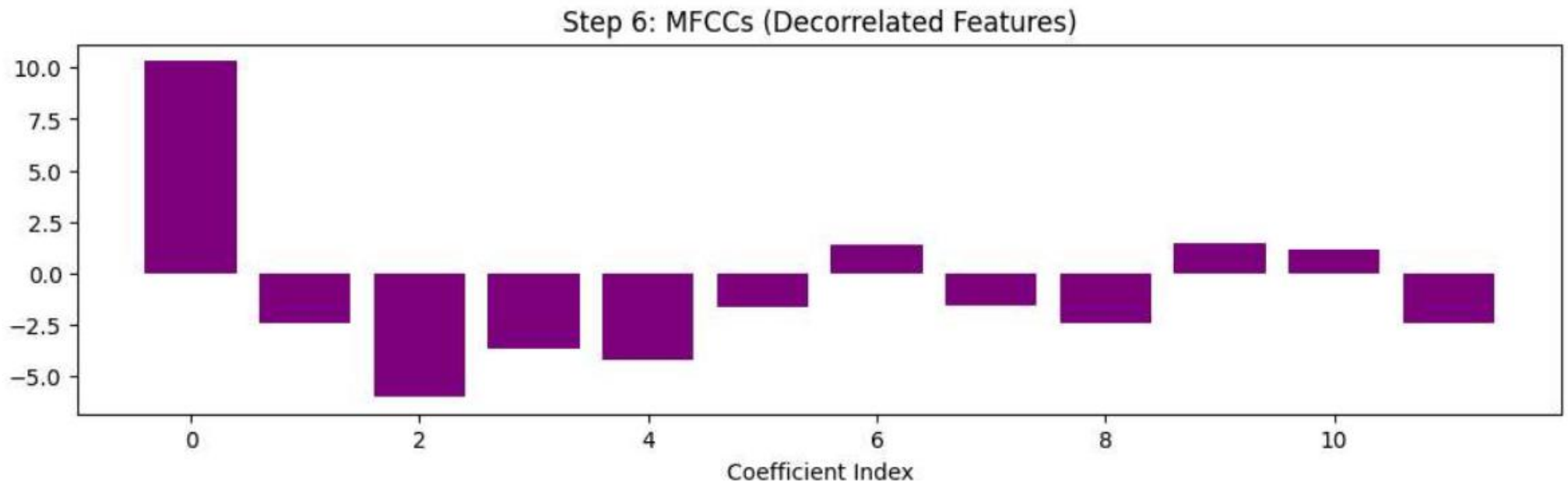


8. The "Cepstrum Smoothing" Flow (iDFT & Low Pass Filter)

This section specifically addresses the "**iDFT**" and "**Low pass filter**",
which are used to find the **spectral envelope (smooth curve)** rather than **MFCC** features.

Explanation:

- **Log Spectrum**: We start with the **Log Magnitude Spectrum** (from Step 5, but usually on the full FFT, not Mel bins).
- **iDFT**: Converts the frequency domain "wave" into the Cepstrum. The x-axis is now "**Quefrency**" (time).
- **Low Pass Filter (Lifter)**: The "**high frequency**" ripples in the **log spectrum** (caused by pitch harmonics) appear as peaks at high quefrency.
- The "**low frequency**" shape (vocal tract) appears at low quefrency.
- We apply a **Low-time Lifter** (keep only low coefficients) to extract the vocal tract.
- **DFT**: We transform back to get the smoothed spectrum.

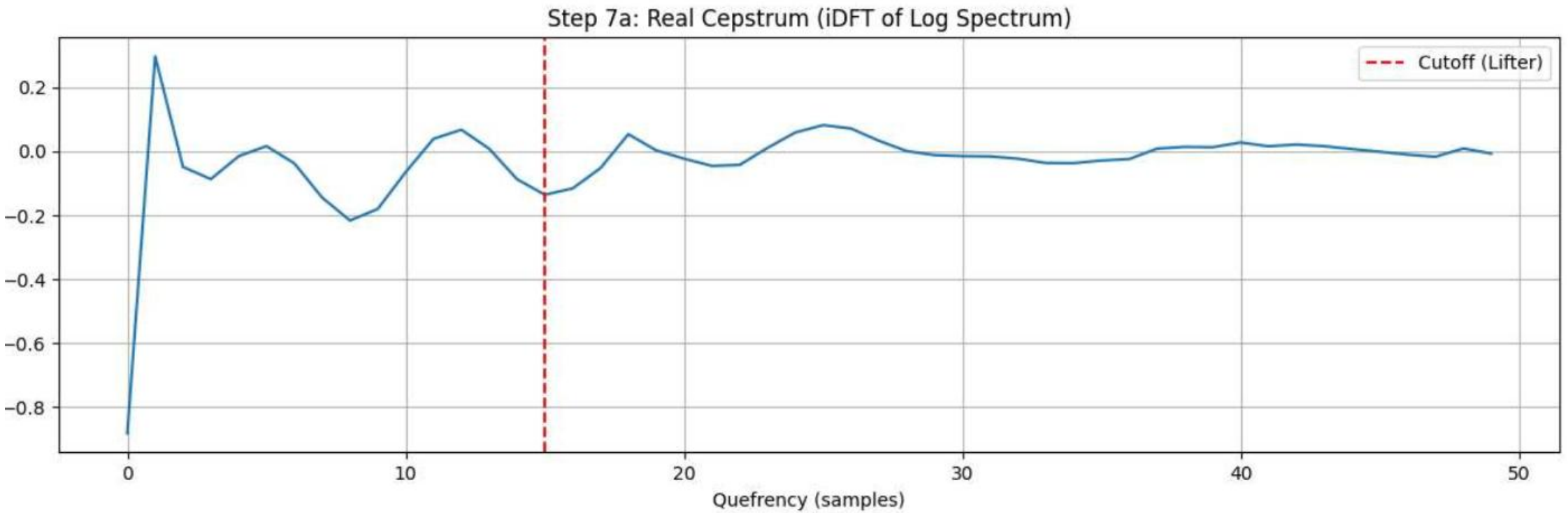


8. The "Cepstrum Smoothing" Flow (iDFT & Low Pass Filter)

This section specifically addresses the "**iDFT**" and "**Low pass filter**",
which are used to find the **spectral envelope (smooth curve)** rather than **MFCC** features.

Explanation:

- **Log Spectrum**: We start with the **Log Magnitude Spectrum** (from Step 5, but usually on the full FFT, not Mel bins).
- **iDFT**: Converts the frequency domain "wave" into the Cepstrum. The x-axis is now "**Quefrency**" (time).
- **Low Pass Filter (Lifter)**: The "**high frequency**" ripples in the **log spectrum** (caused by pitch harmonics) appear as peaks at high quefrency.
- The "**low frequency**" shape (vocal tract) appears at low quefrency.
- We apply a **Low-time Lifter** (keep only low coefficients) to extract the vocal tract.
- **DFT**: We transform back to get the smoothed spectrum.

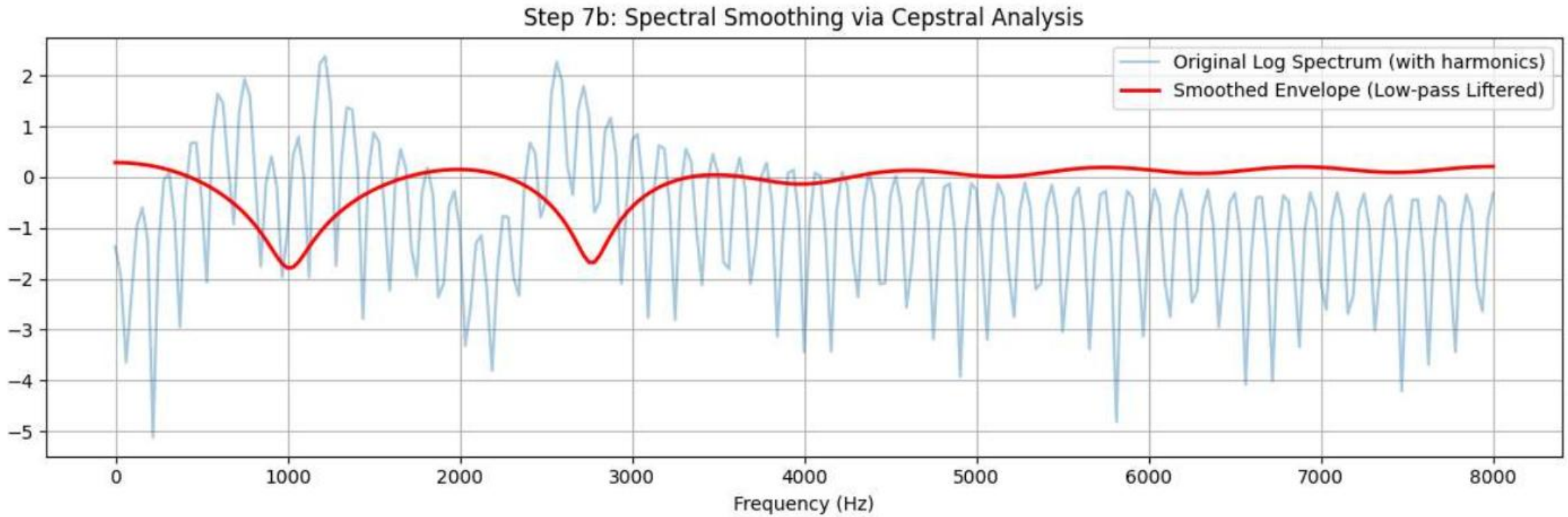


8. The "Cepstrum Smoothing" Flow (iDFT & Low Pass Filter)

This section specifically addresses the "**iDFT**" and "**Low pass filter**",
which are used to find the **spectral envelope (smooth curve)** rather than **MFCC** features.

Explanation:

- **Log Spectrum**: We start with the **Log Magnitude Spectrum** (from Step 5, but usually on the full FFT, not Mel bins).
- **iDFT**: Converts the frequency domain "wave" into the Cepstrum. The x-axis is now "**Quefrency**" (time).
- **Low Pass Filter (Lifter)**: The "**high frequency**" ripples in the **log spectrum** (caused by pitch harmonics) appear as peaks at high quefrency.
- The "**low frequency**" shape (vocal tract) appears at low quefrency.
- We apply a **Low-time Lifter** (keep only low coefficients) to extract the vocal tract.
- **DFT**: We transform back to get the smoothed spectrum.



Step 1: The Acoustic Model (GMM)

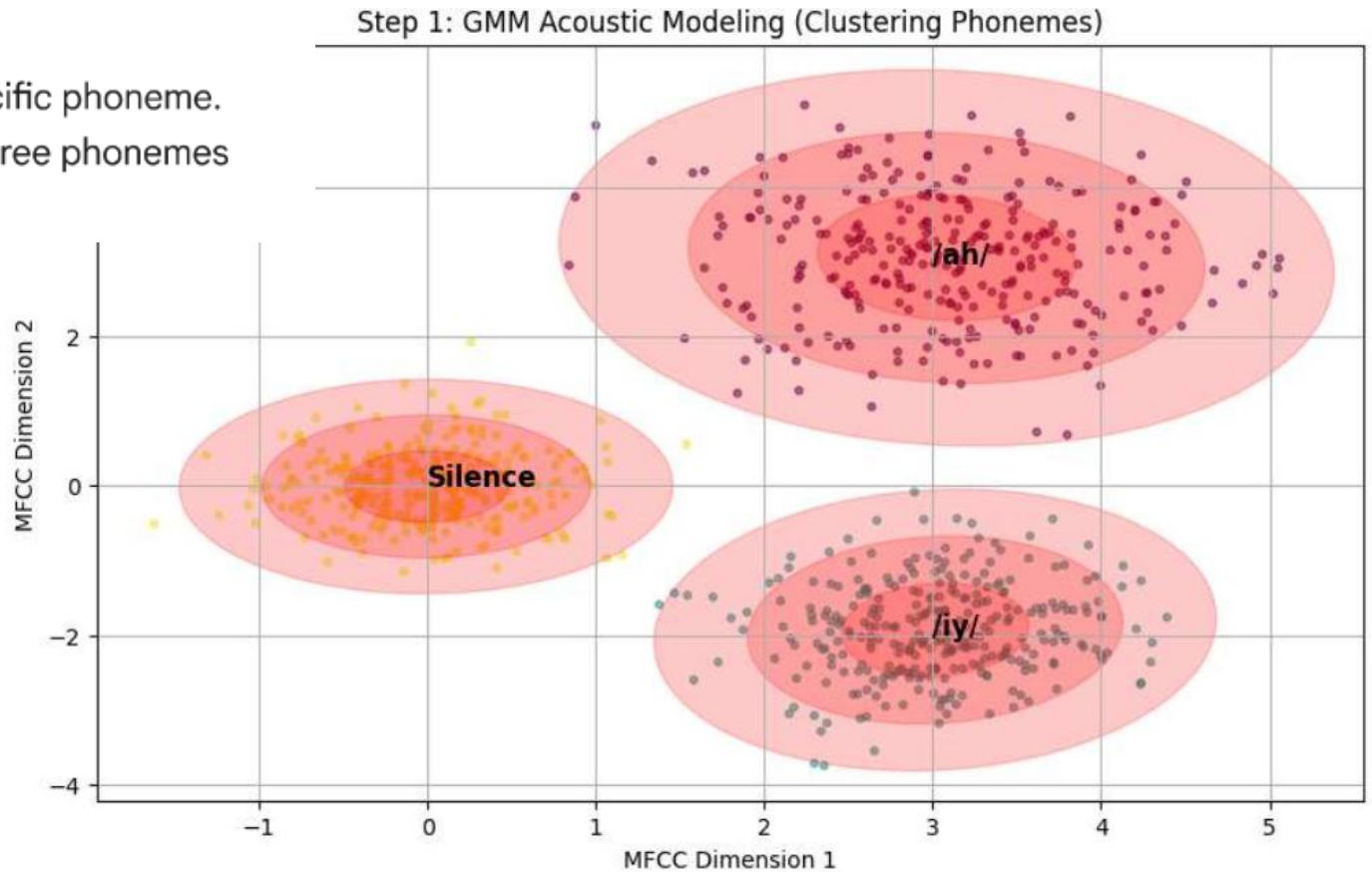
Explanation:

- **MFCCs** are just vectors of numbers.
- If you plot them, the same `phoneme` (e.g., "Ah") uttered by different people tends to cluster in the same area of vector space.
- A **Gaussian Mixture Model (GMM)** tries to fit "clouds" (Gaussian distributions) around these clusters.
    - **Input**: A single MFCC frame (e.g., vector of length 12).
    - **Output**: The probability that this frame belongs to a specific phoneme.
    - **Code**: We will generate **"toy" MFCC** data representing three phonemes (Silence, /ah/, /iy/) and train a GMM to recognize them.

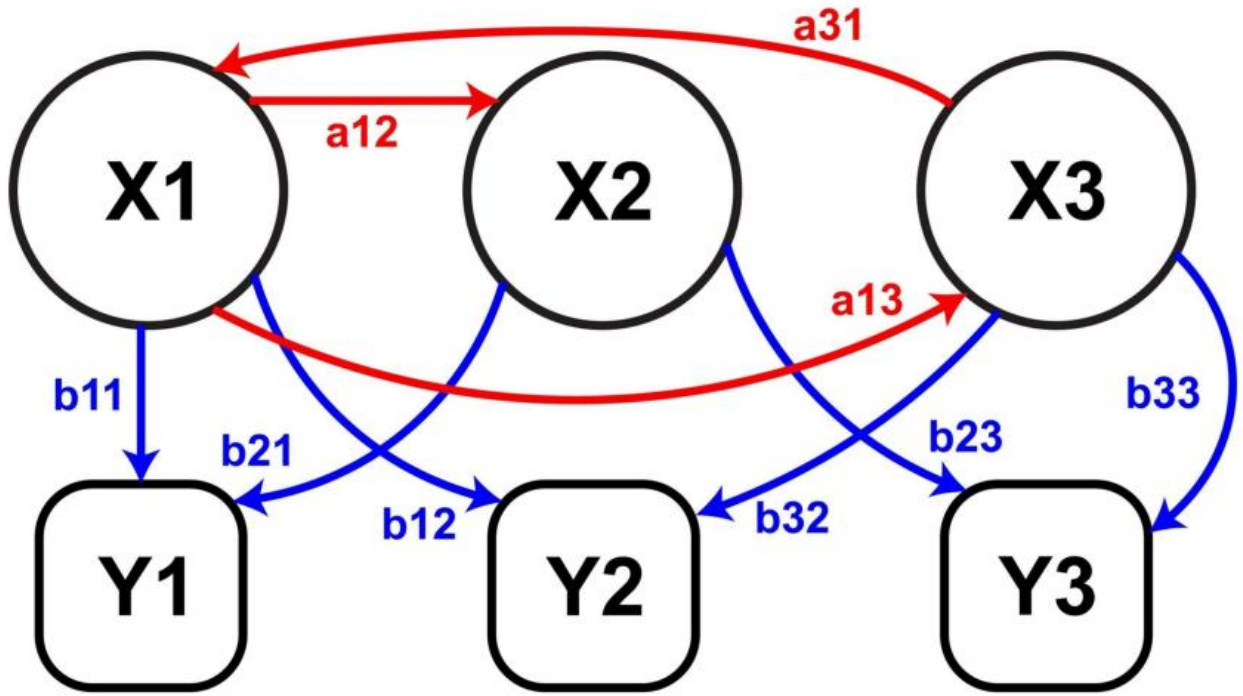
Step 1: The Acoustic Model (GMM)

Explanation:

- **MFCCs** are just vectors of numbers.
- If you plot them, the same `phoneme` (e.g., "Ah") uttered by different people tends to cluster in the same area of vector space.
- A **Gaussian Mixture Model (GMM)** tries to fit "clouds" (Gaussian distributions) around these clusters.
    - **Input**: A single MFCC frame (e.g., vector of length 12).
    - **Output**: The probability that this frame belongs to a specific phoneme.
    - **Code**: We will generate **"toy" MFCC** data representing three phonemes (Silence, /ah/, /iy/) and train a GMM to recognize them.



Step 2: The Sequence Model (HMM)



Hidden Markov model

HMM

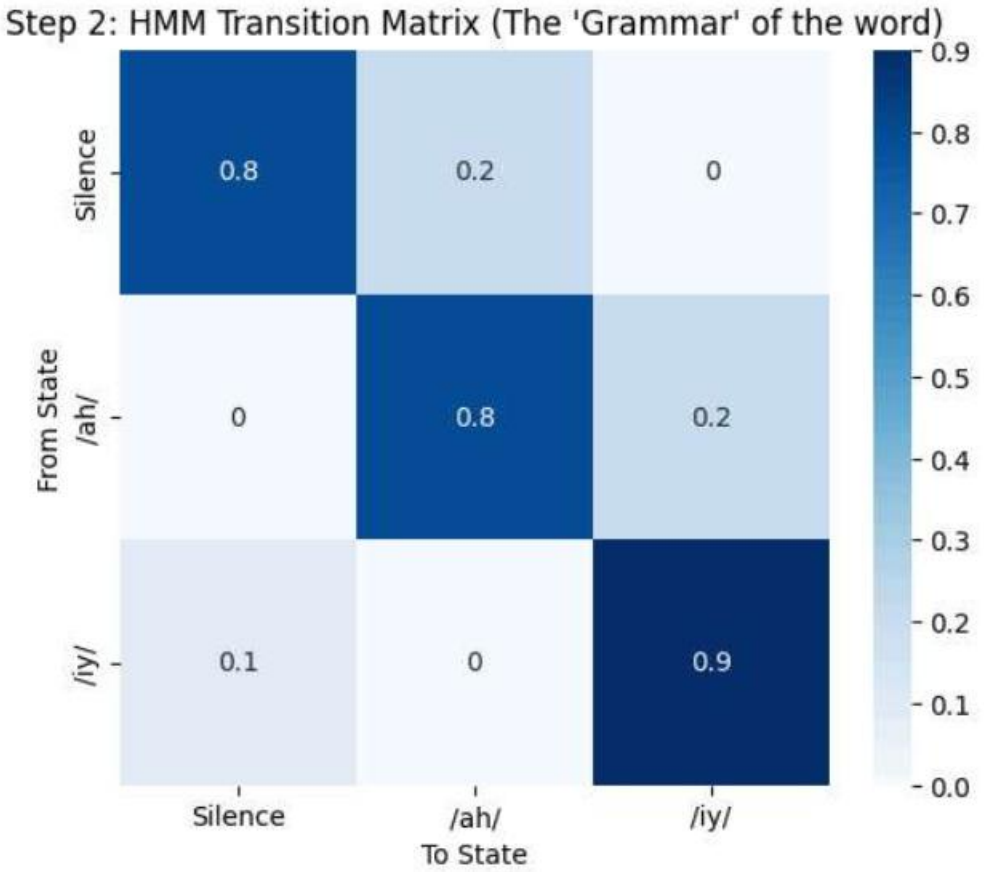
Explanation:

- Speech has rules.
  - You rarely go from "Silence" directly to the middle of a word without starting it.
- **The Hidden Markov Model (HMM)** manages these probabilities.
- **Transition Matrix (A)**: Probability of moving from state i to state j.
  - (e.g., $P(\text{Ah} \rightarrow \text{Silence})$).
- **Emission Probability (B)**: Probability that state i produced the observed MFCC vector (we get this from the GMM we just trained!).
- **Code**: We define the rules for a simple word "Hi" (Silence $\rightarrow$ /ah/ $\rightarrow$ /iy/ $\rightarrow$ Silence).

HMM

Explanation:

- Speech has rules.
  - You rarely go from "Silence" directly to the middle of a word without starting it.
- **The Hidden Markov Model (HMM)** manages these probabilities.
- **Transition Matrix (A)**: Probability of moving from state i to state j.
  - (e.g., $P(\text{Ah} \rightarrow \text{Silence})$).
- **Emission Probability (B)**: Probability that state i produced the observed MFCC vector (we get this from the GMM we just trained!).
- **Code**: We define the rules for a simple word "Hi" (Silence $\rightarrow$ /ah/ $\rightarrow$ /iy/ $\rightarrow$ Silence).

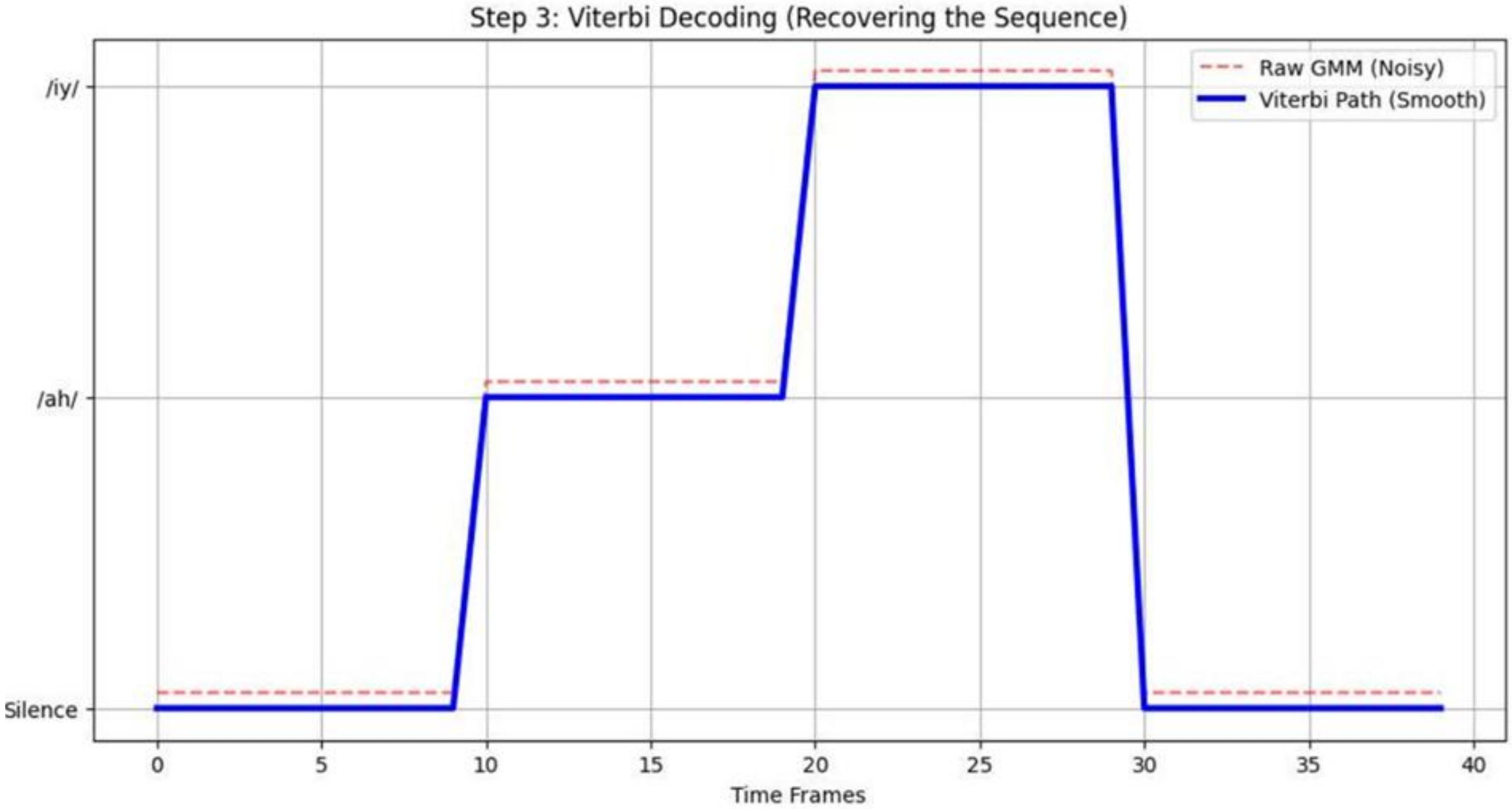


Step 3: Decoding (The Viterbi Algorithm)

Explanation:

- We now have a **sequence of observed MFCC**s.
- We want to find the **single most likely sequence of hidden states (Phonemes)** that produced them.
- If we just picked the **best GMM match for every frame**, the prediction would be noisy (e.g., "Silence, Silence, Ah, Silence, Ah, Ah...").
- **The Viterbi Algorithm l**ooks at the whole sequence globally to find the smoothest path that obeys the Transition Matrix.

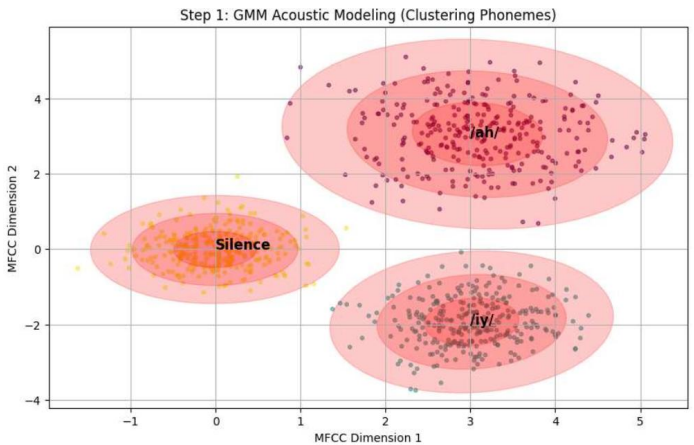
Code: We generate a simulated "spoken word" trajectory and decode it.

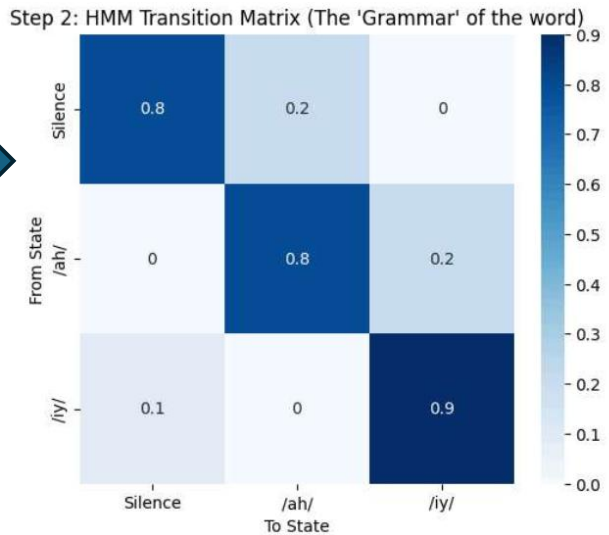


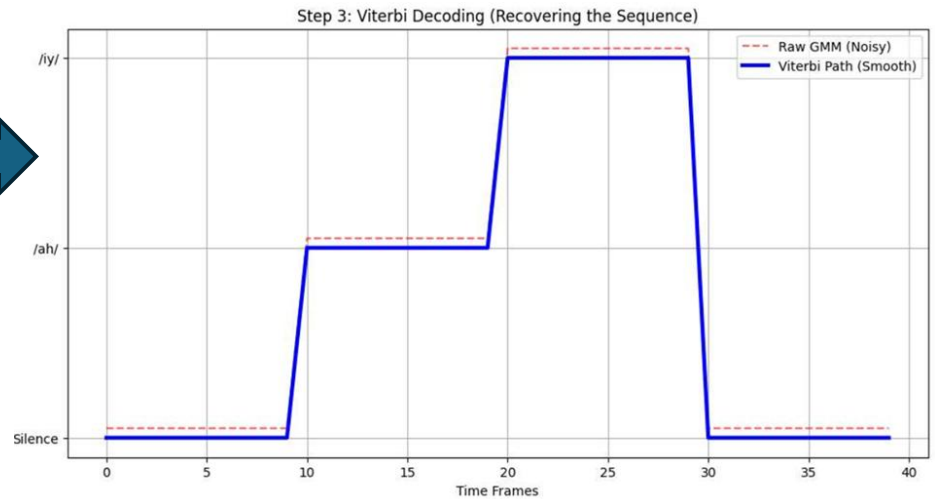
True Sequence Structure: Silence -> /ah/ -> /iy/ -> Silence

Summary of the Classic Flow

- **GMM**: Looked at the **features (MFCCs) spatially**.
  It asked: "Is this point close to the 'Ah' cluster?"
- **HMM**: Looked at the features temporally.
  It asked: "Does it make sense to jump from Silence to 'Ah' right now?"
- **Viterbi**: Connected the dots to draw the most logical line through the confusion.







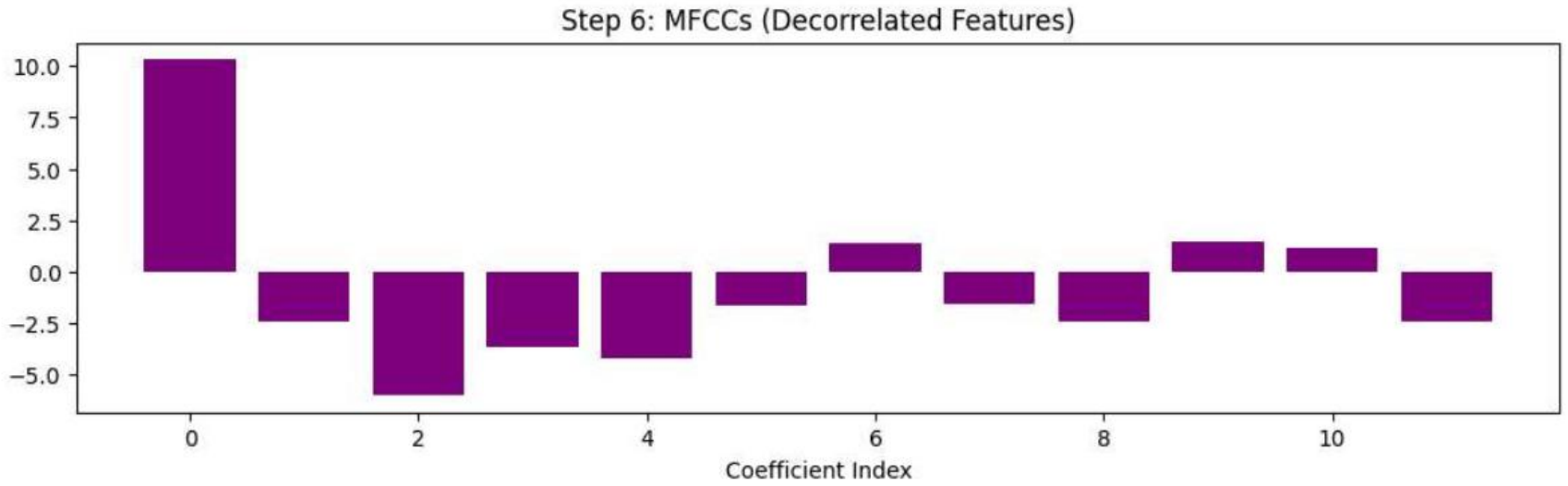
Example 1: Gender Classification (The Classic ML Approach)

- **Theory**: Biological differences in vocal folds and tract length result in distinct acoustic features.
- **Fundamental Frequency (F0/Pitch)**:
  - Male vocal folds vibrate slower (~85–180 Hz)
  - Female vocal folds vibrate faster (~165–255 Hz).
- **Timbre (MFCCs)**: Males typically have longer vocal tracts, shifting formants (resonant frequencies) lower.

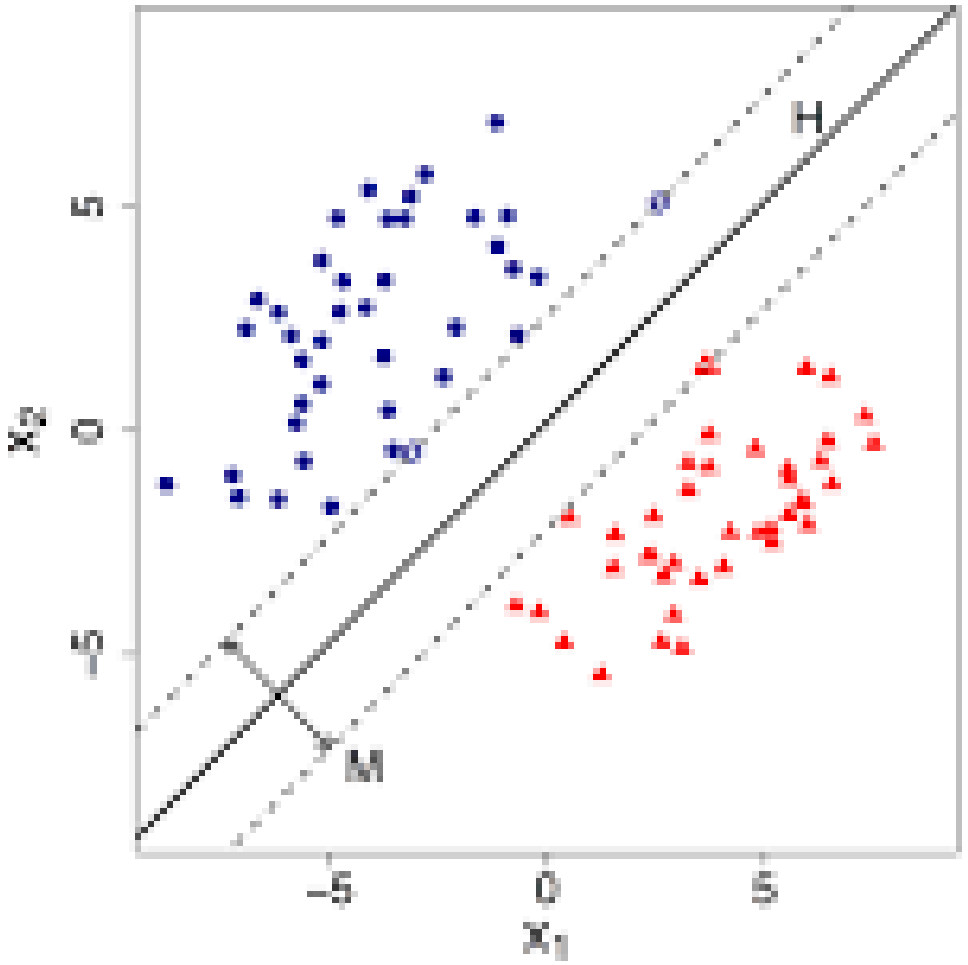
We will generate synthetic male/female data, extract features, and train a classifier.

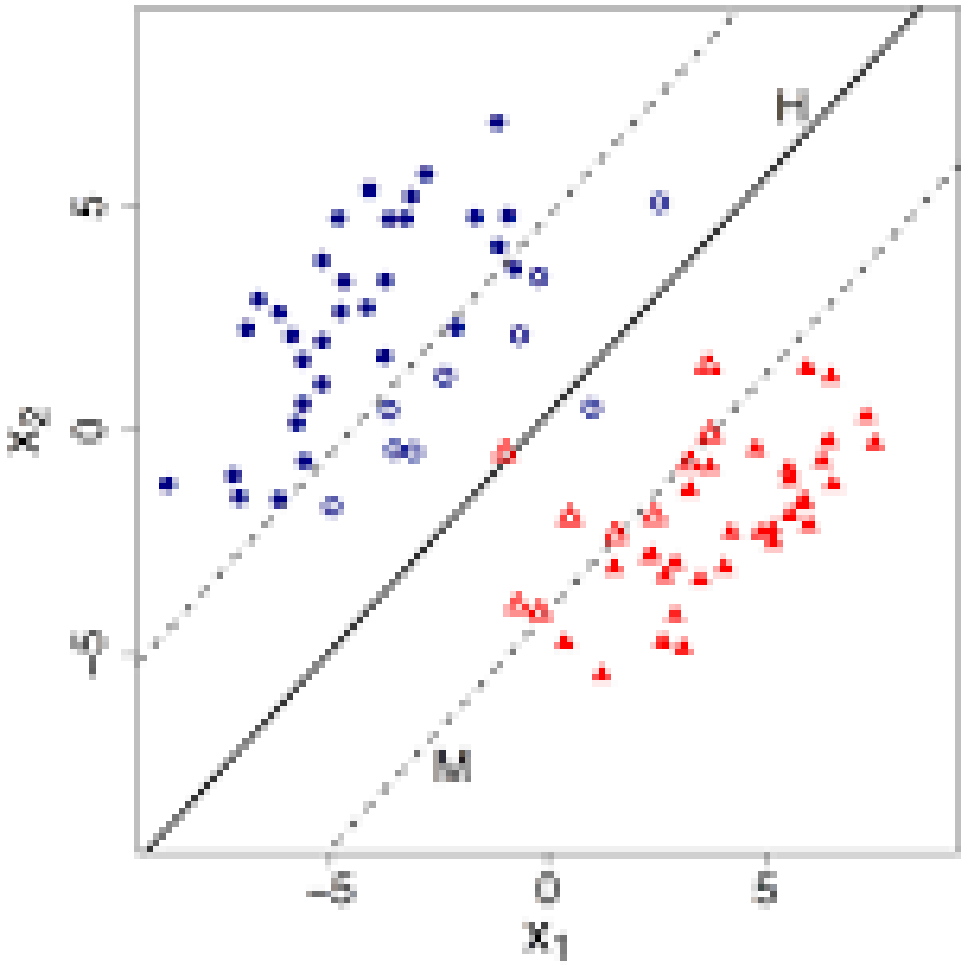
Step 2: Feature Extraction (Pitch & MFCC)

- We need to condense the audio into numbers.
- We will use **Cepstrum-based Pitch Detection** (as learned previously) and **MFCCs**.

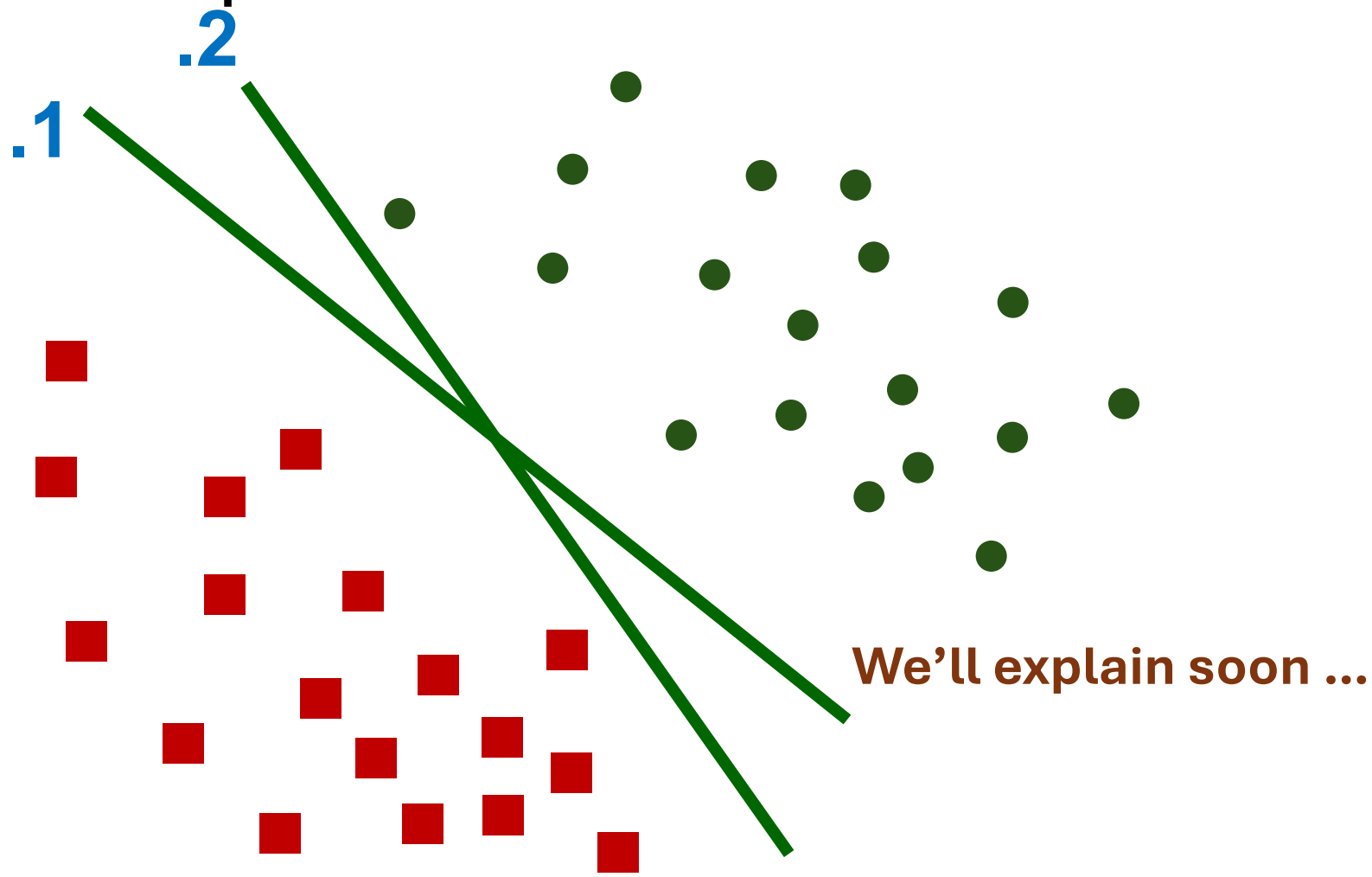


Introduction to SVM

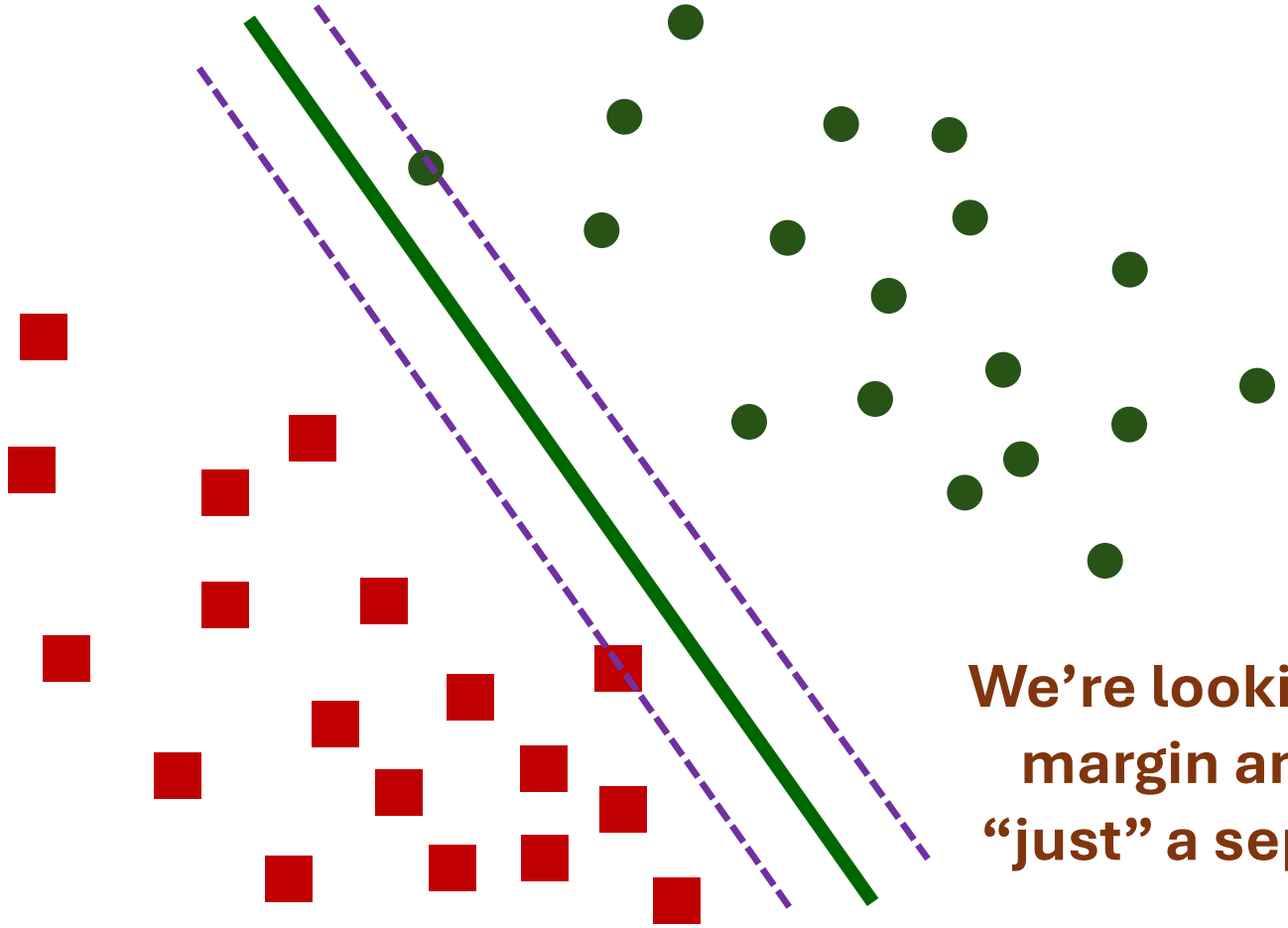




Which linear separator is better?

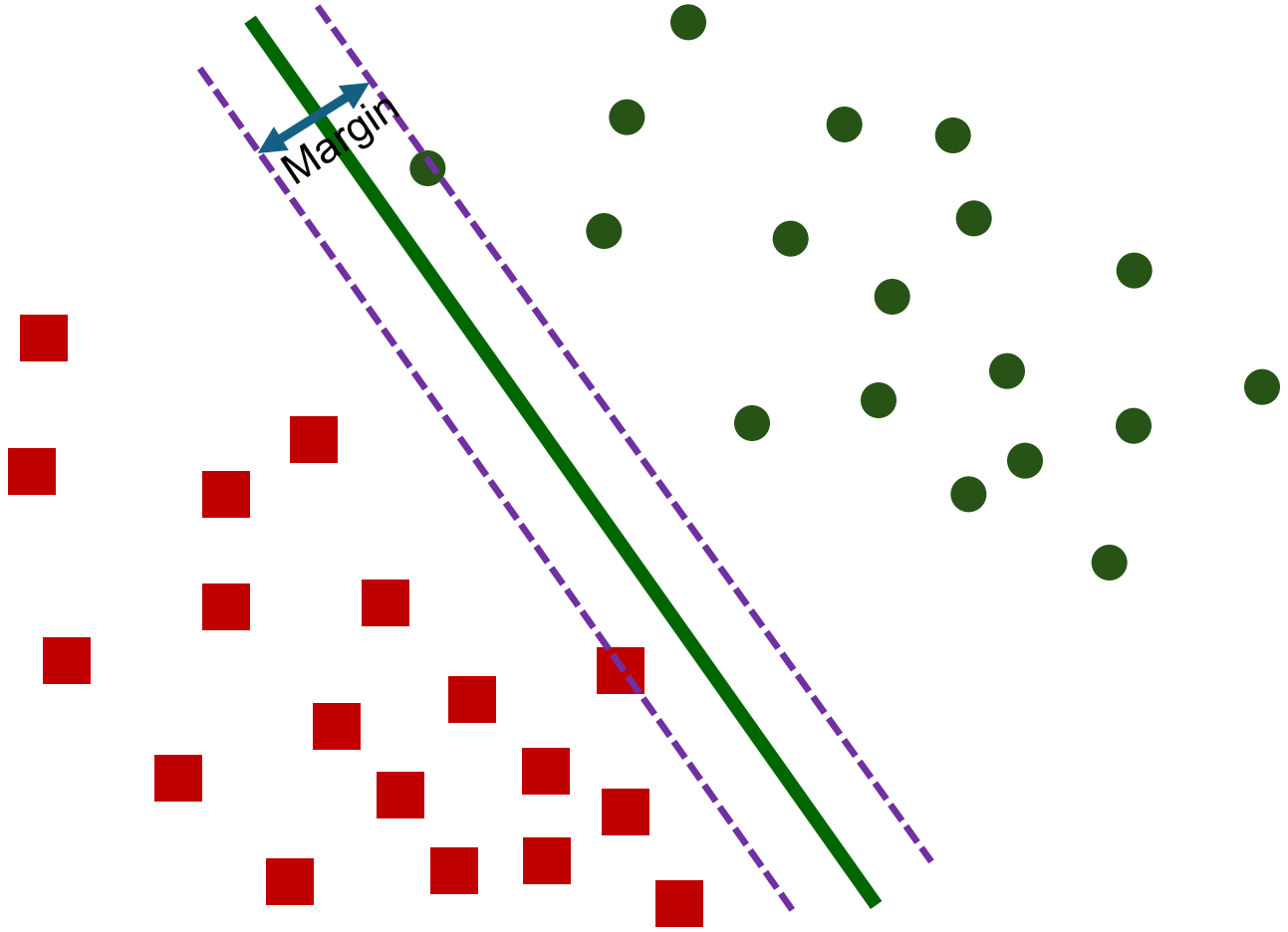


Which linear separator is better?

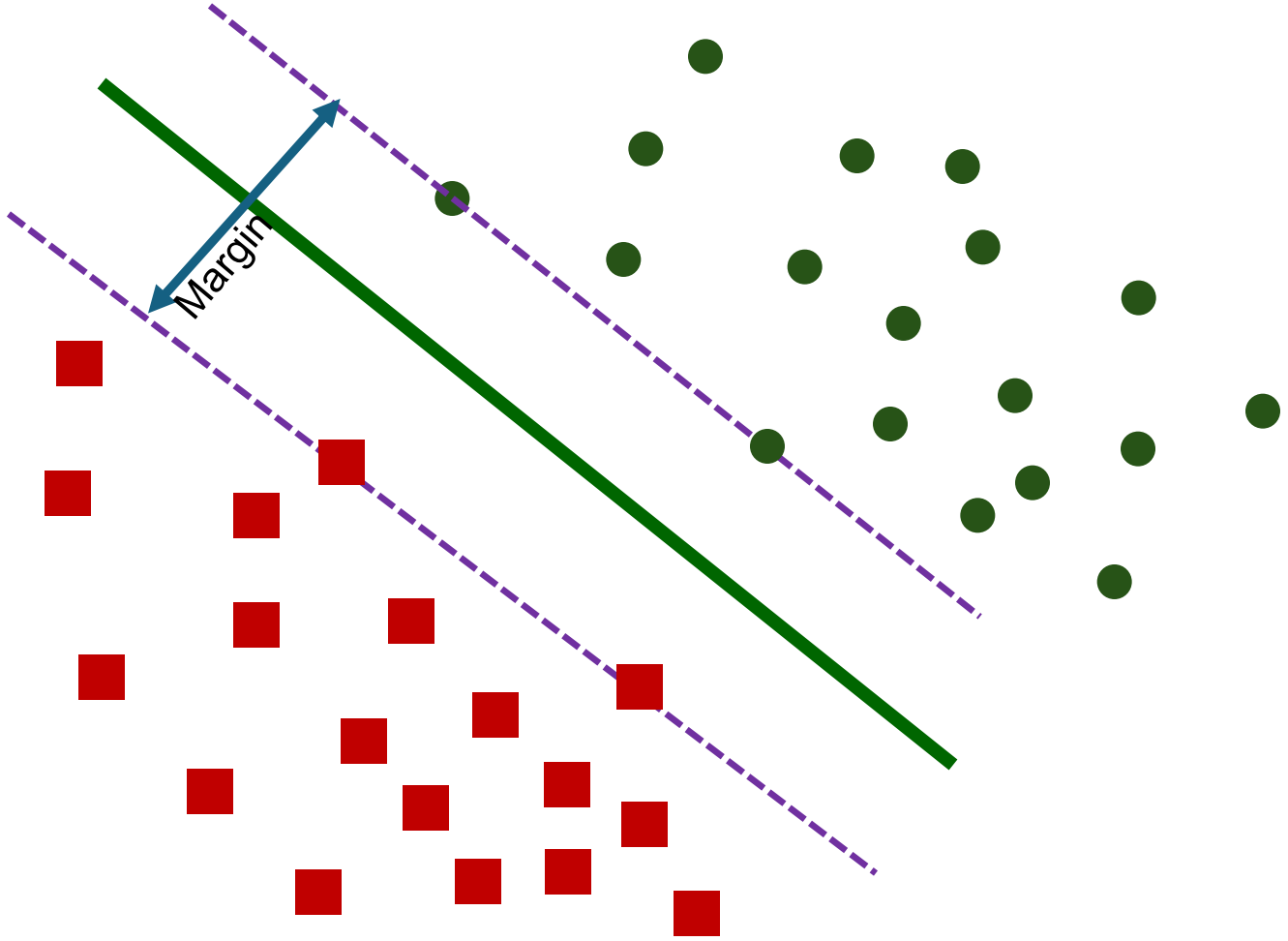


We’re looking for a margin and not “just” a separator

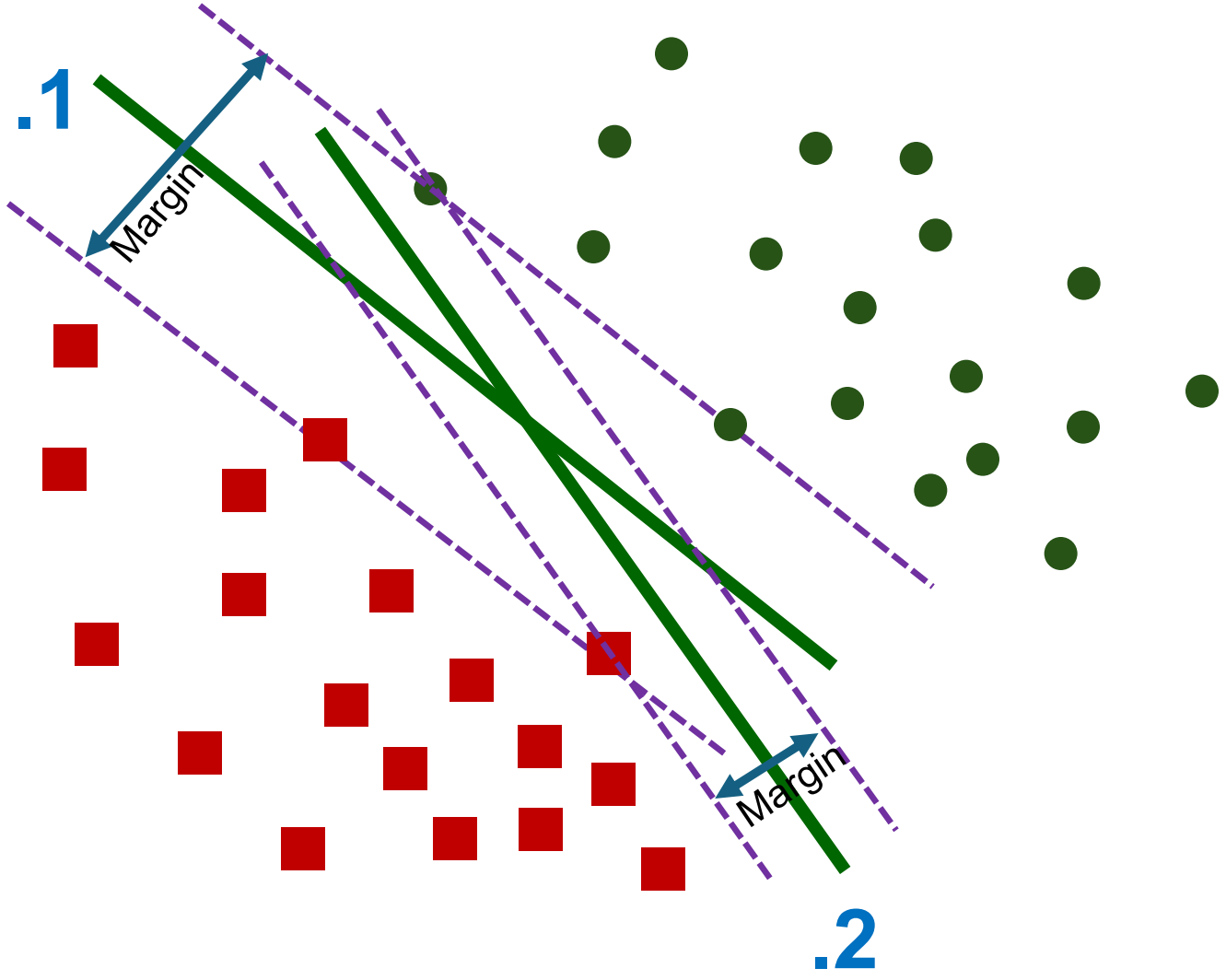
Different Margin options



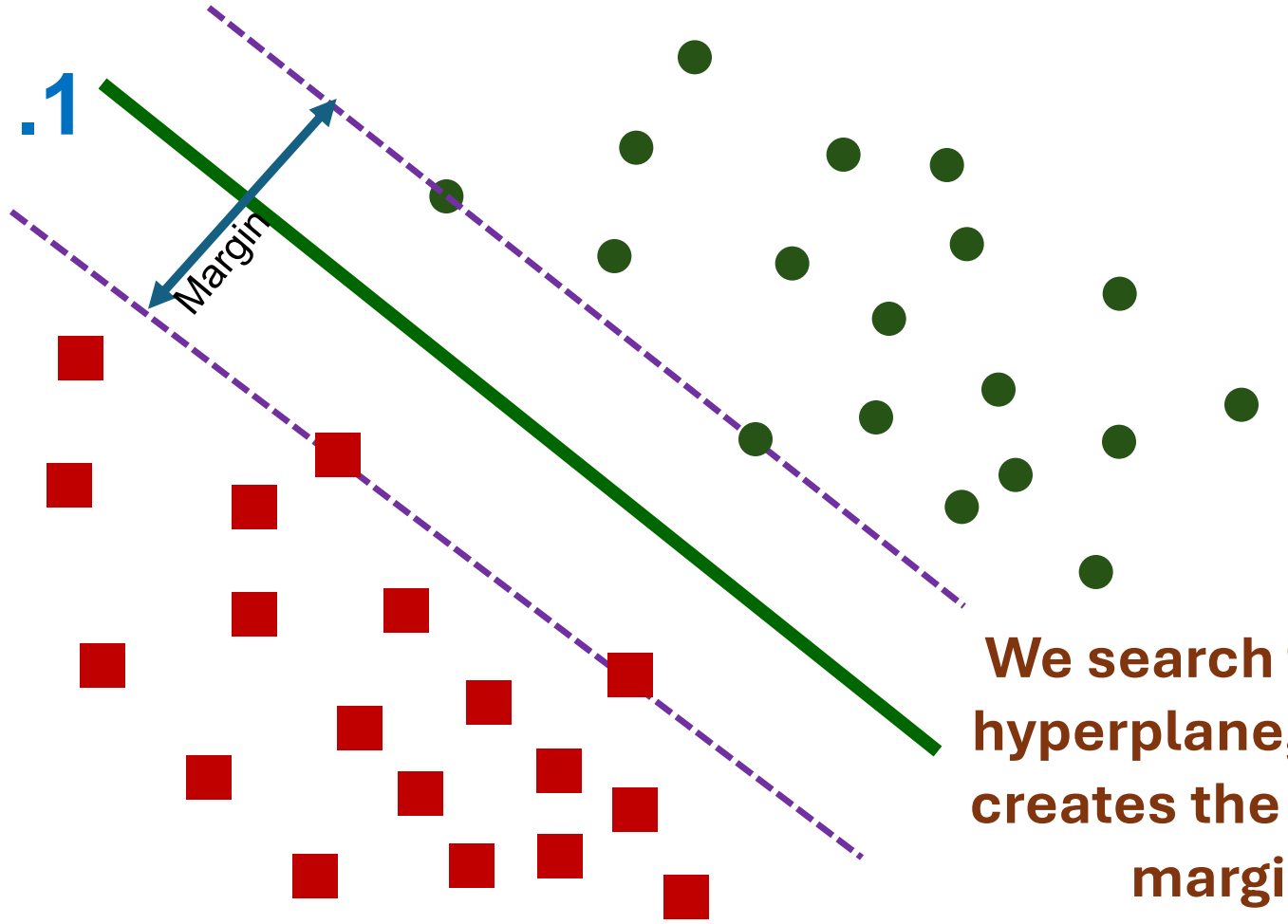
Different Margin options



Which Separating Margin is better?



Which Separating Margin is better?



We search for the hyperplane, which creates the largest margin

Calculating the Margin

Point-plain distances from the two margins to the origin:

$$d_+ = \frac{\left|\left(w \cdot 0\right) + b + 1\right|}{\|w\|}, d_- = \frac{\left|\left(w \cdot 0\right) + b - 1\right|}{\|w\|}$$

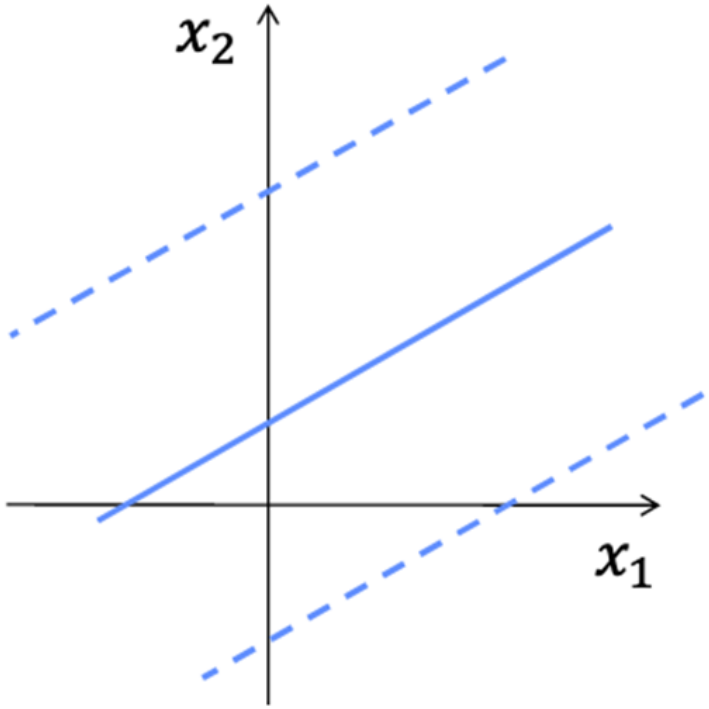
$$\Rightarrow \mathrm{M} = \frac{2}{\|w\|}$$

Margin Calculation

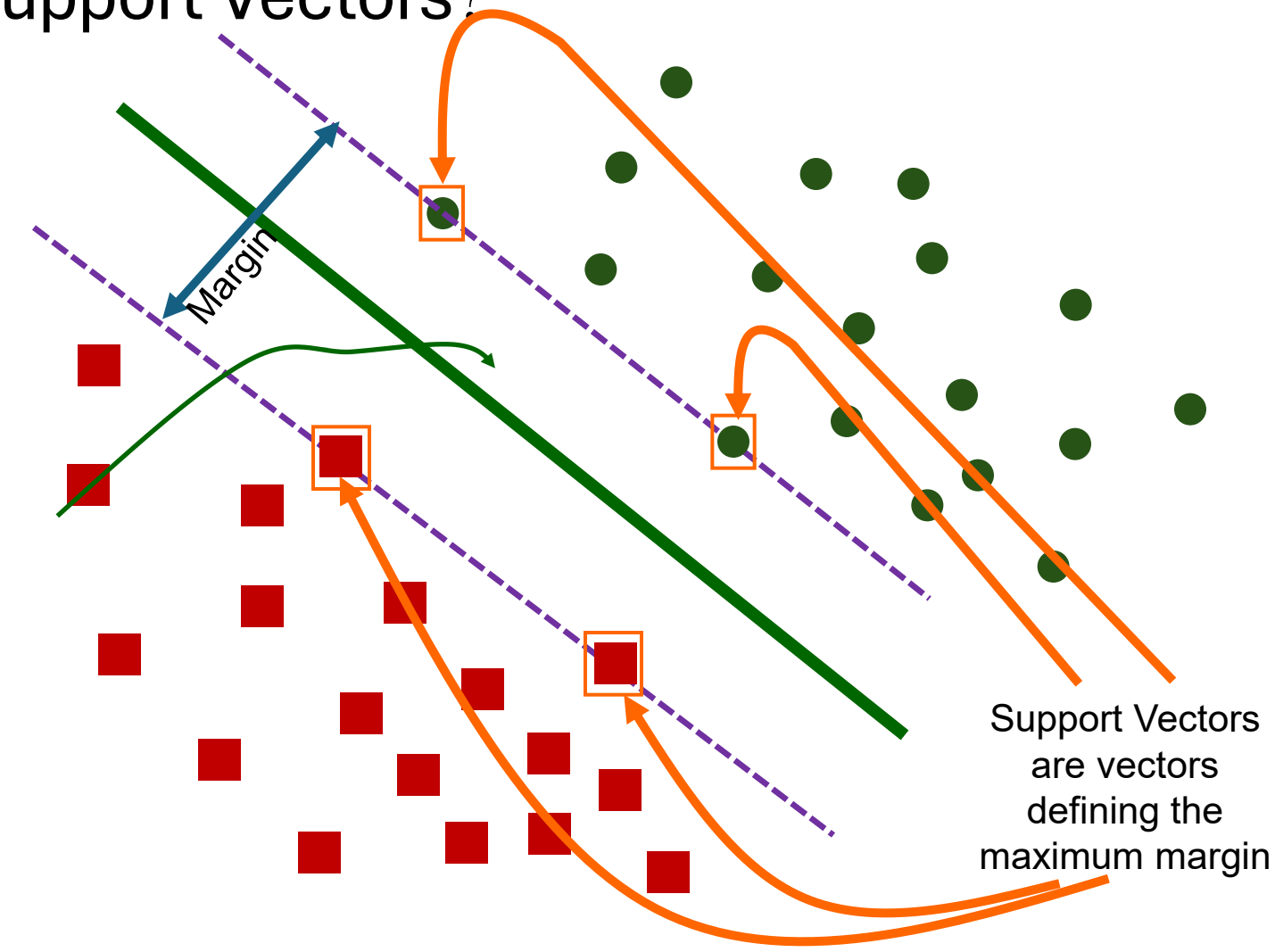
Objective:

$$max(\mathrm{M}) \rightarrow min(\|w\|^2)$$

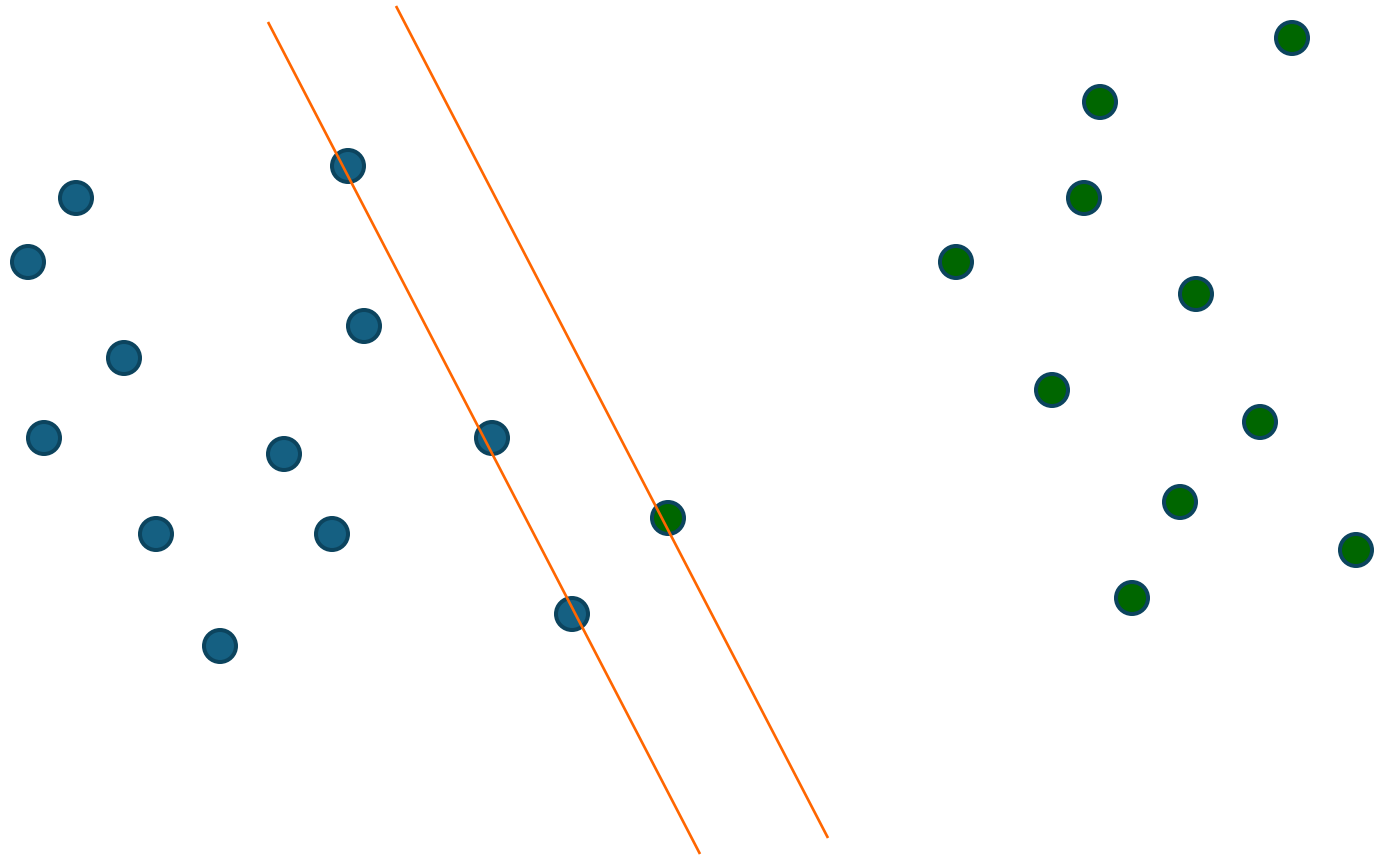
$$s.t.$$



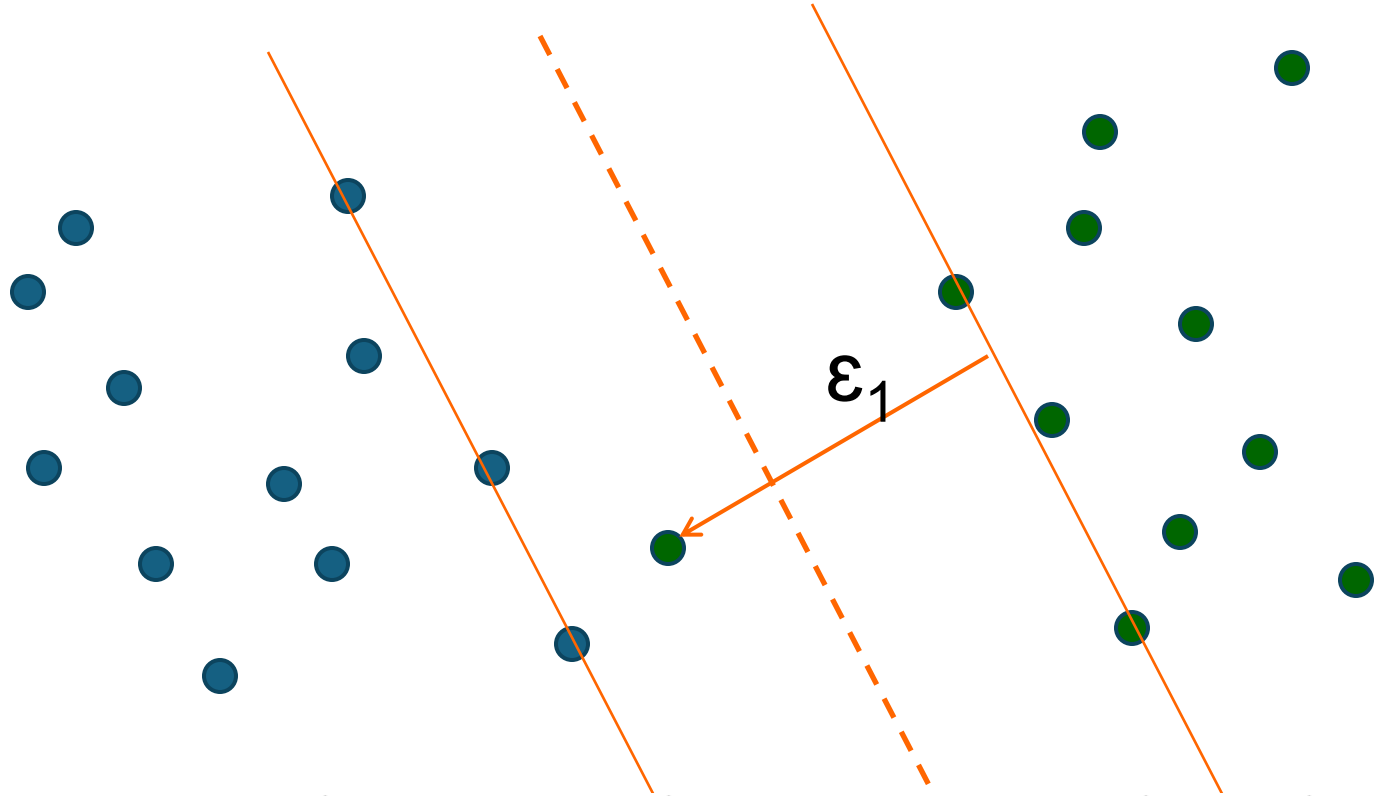
SVM – Support Vector Machines–
What are Support Vectors?



Hard Margin



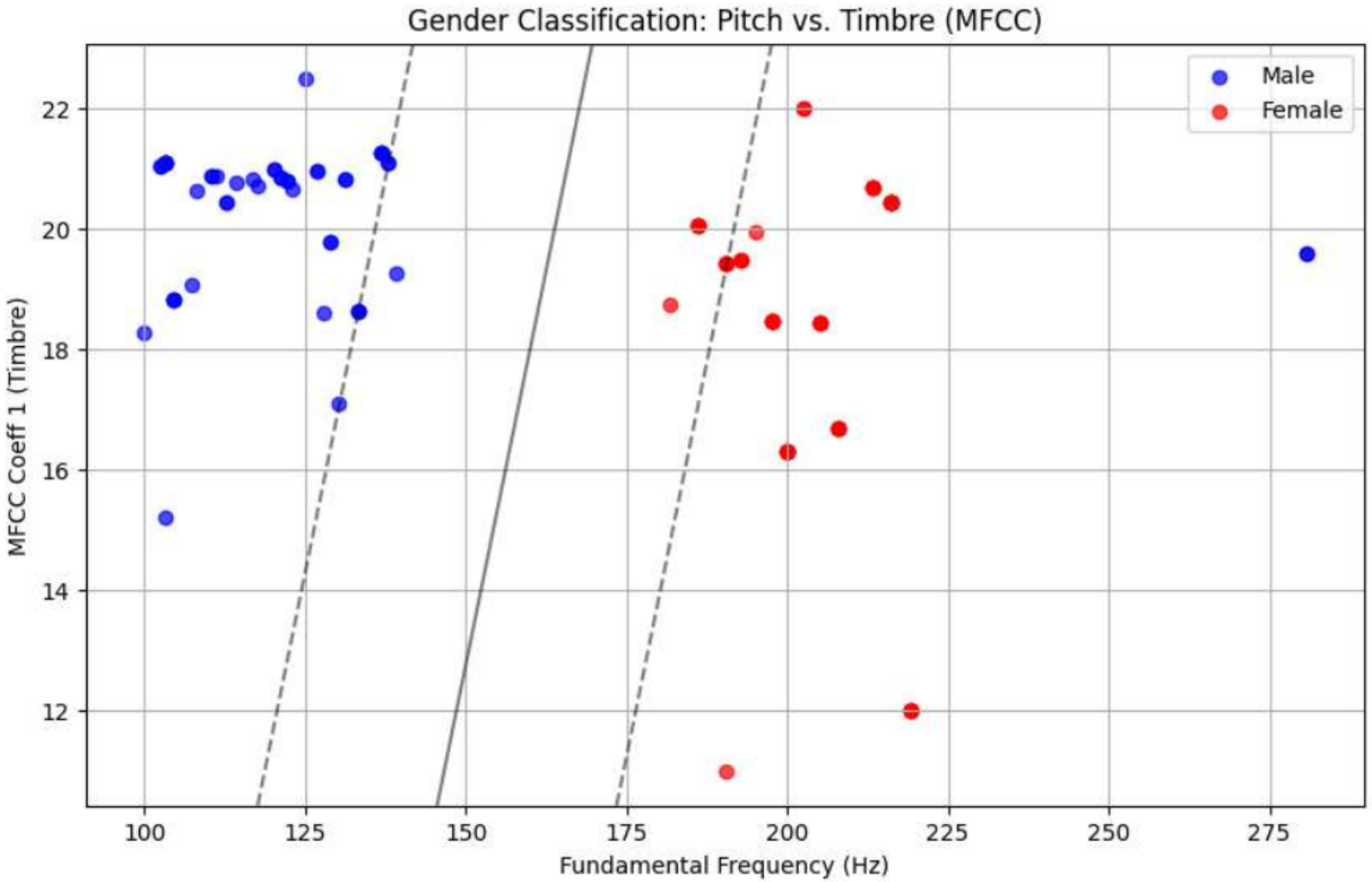
Soft Margin



$$\xi_j = \max\left(0, 1 - \left(w \cdot x_j + b\right) y_j\right)$$

Step 3: Classification

We use a Support Vector Machine (SVM) to find the line that divides Male/Female.



Example 2 - Speech recognition

- This is an end-to-end example of the **GMM-HMM (Gaussian Mixture Model - Hidden Markov Model)** architecture.
- We build a **specific** mathematical **model for every word (or phoneme)** we want to recognize.
    - **The Task**: "Isolated Digit Recognition"
    - **Example**: We will build a system that can distinguish between the spoken words "One" and "Two".

The Architecture

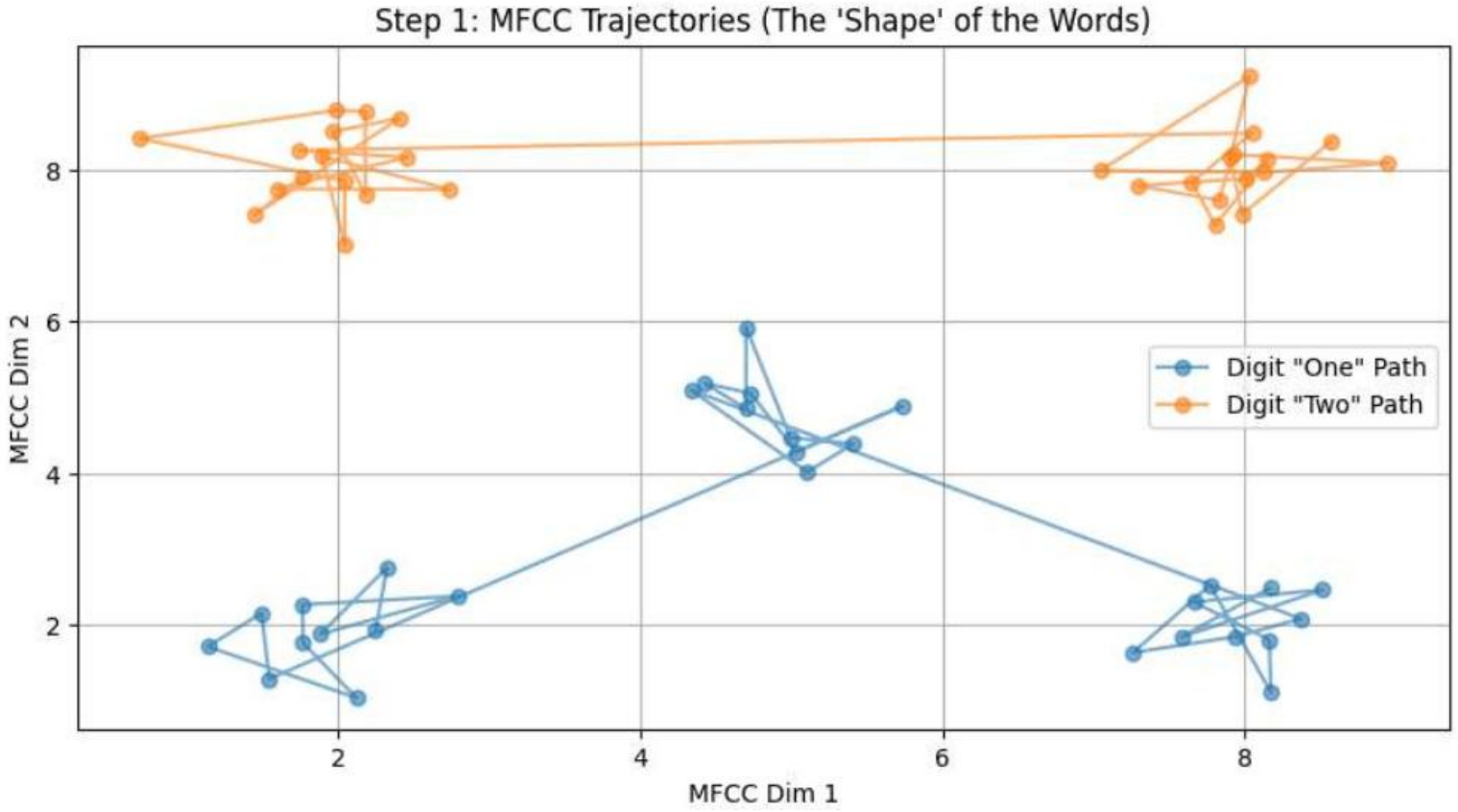
- **Feature Extraction**: MFCCs (Standard).Acoustic Model (GMM):
  Statistical models of the sounds inside the words (e.g., the "w" in "one").
- **Sequence Model (HMM)**: A "Left-to-Right" state machine that enforces the time order
  (Start → Middle → End).
- **Decoding**: We pass the audio through both models.
  Whichever model produces the higher probability score "wins".

Step 1: Generate Synthetic "Digit" Data

- Example: we will simulate the MFCC signatures of "One" and "Two".
  "One" ($/w/ \rightarrow / \wedge / \rightarrow /n/$):
- Starts with low energy, moves to a central vowel, ends with a nasal.
  "Two" ($/t/ \rightarrow /u/$):
- Starts with a burst (high freq), moves to a back vowel.

Step 1: Generate Synthetic "Digit" Data

- Example: we will simulate the MFCC signatures of "One" and "Two".
  "One" ($/w/ \rightarrow / \wedge / \rightarrow /n/$):
- Starts with low energy, moves to a central vowel, ends with a nasal.
  "Two" ($/t/ \rightarrow /u/$):
- Starts with a burst (high freq), moves to a back vowel.



Step 2: Defining the HMM Topology

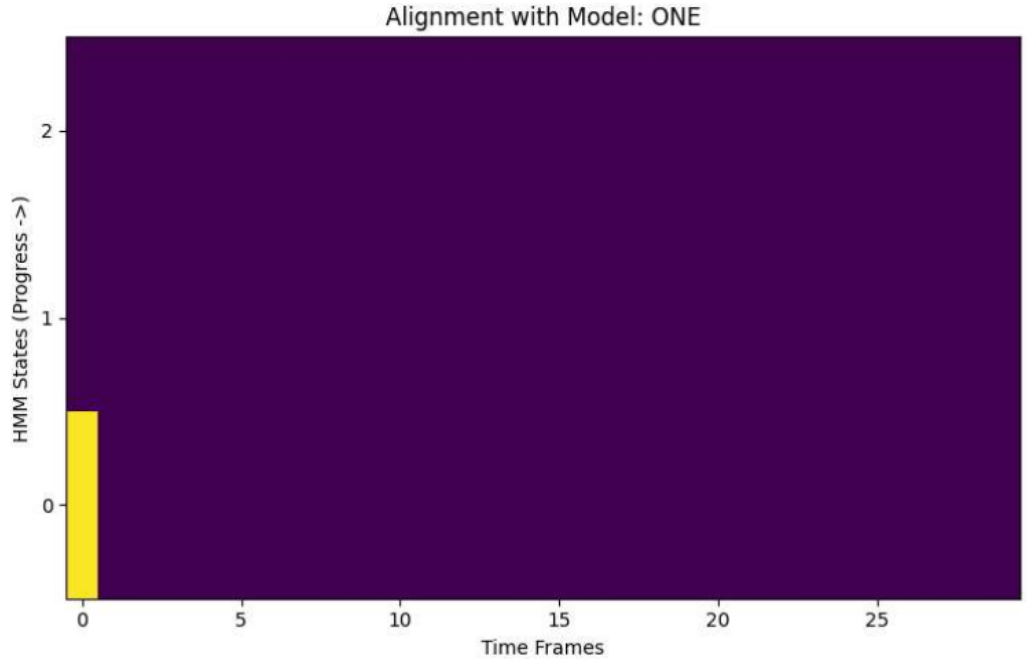
- In classic ASR, words are modeled as Left-to-Right HMMs.
- You can only move forward in time (or stay in the same state), never backward.
    - State 0: Start of word.
    - State 1: Middle of word.
    - State 2: End of word.
- We define a class that represents a single Word Model.

Step 2: Defining the HMM Topology

- In classic ASR, words are modeled as Left-to-Right HMMs.
- You can only move forward in time (or stay in the same state), never backward.
    - State 0: Start of word.
    - State 1: Middle of word.
    - State 2: End of word.
- We define a class that represents a single Word Model.

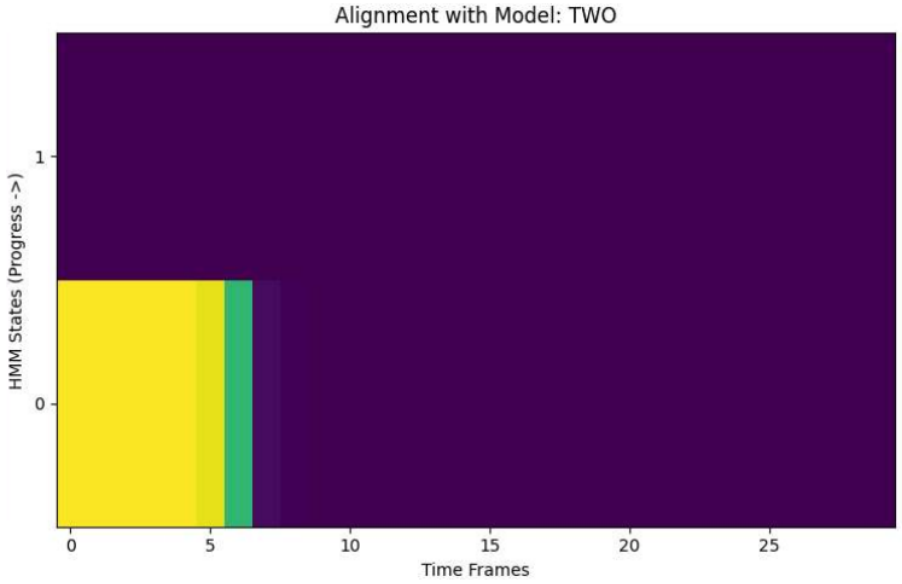
Step 3: Recognition (End-to-End Test)

- Now we generate a Test Signal.
- We don't tell the system what it is.
- The system will:
    - Feed the signal into Model "One".
    - Feed the signal into Model "Two".
    - Compare the scores.



Step 3: Recognition (End-to-End Test)

- Now we generate a Test Signal.
- We don't tell the system what it is.
- The system will:
  - Feed the signal into Model "One".
  - Feed the signal into Model "Two".
  - Compare the scores.




```
Likelihood for Model 'ONE': 9.64016e-309
Likelihood for Model 'TWO': 1.43801e-36

>>> RESULT: Recognized word 'TWO'
```

Summary of the "Classic" Era

- **Modularity**: Everything was broken down.
  We explicitly built a "One" model and a "Two" model.
- **Independence**: The Acoustic Model (GMM) and Sequence Model (HMM) were trained separately and then combined.
- **Search**: Recognition was purely a mathematical search (Viterbi/Forward) to find which graph path had the highest probability.

Signal Processing and Information Retrieval (IR)

- This field was known as **Spoken Document Retrieval (SDR)**.
  The challenge was that the output of an HMM is noisy.
  If the person said "Speech recognition" but the HMM heard "Peach wreck a nation," a standard keyword search for "Speech" would fail.

Here is how classic systems integrated HMM outputs into search engines, moving from the "Naive" approach to the "Robust" approach.

Signal Processing and Information Retrieval (IR)

- This field was known as **Spoken Document Retrieval (SDR)**.
  The challenge was that the output of an HMM is noisy.
  If the person said "Speech recognition" but the HMM heard "Peach wreck a nation," a standard keyword search for "Speech" would fail.

Here is how classic systems integrated HMM outputs into search engines, moving from the "Naive" approach to the "Robust" approach.

Approach 1: The "1-Best" Pipeline (Naive)

The simplest integration was to take the Viterbi path (the single best sequence of words) from the HMM, treat it as a text document, and index it using standard TF-IDF.

Problem: ASR Error Propagation. If the Word Error Rate (WER) is 30%, you lose 30% of your searchable terms.

Signal Processing and Information Retrieval (IR)

- This field was known as **Spoken Document Retrieval (SDR)**.
  The challenge was that the output of an HMM is noisy.
  If the person said "Speech recognition" but the HMM heard "Peach wreck a nation," a standard keyword search for "Speech" would fail.

Here is how classic systems integrated HMM outputs into search engines, moving from the "Naive" approach to the "Robust" approach.

Approach 2: Lattice-Based Indexing (Robust)

- This was the standard "Pro" technique in classic systems.
- Instead of saving just the "Winner" (Viterbi path), the system saved the entire confusing decision process of the HMM.
- This structure is called a Word Lattice (or Confusion Network).
- **The Lattice**: A graph where edges are words and weights are probabilities.
- **The Trick**: We index all likely words.
  - If the HMM was 60% sure of "Speech" and 40% sure of "Peach", both get indexed, but with different weights.

From HMM Lattice to Probabilistic Search

- We will simulate a scenario where an HMM processed an audio file and wasn't sure what it heard.
- We will then implement a Probabilistic TF-IDF search to find the document despite the error.

From HMM Lattice to Probabilistic Search

- We will simulate a scenario where an HMM processed an audio file and wasn't sure what it heard.
- We will then implement a Probabilistic TF-IDF search to find the document despite the error.

Step 1: Simulating the HMM Output (The Confusion Network)

Imagine we have two audio documents:

- **Doc A**: The user said "Global warming".
  - The HMM is clear.
- **Doc B**: The user said "Climate change".
  - The HMM is confused - it thinks it might be "Climb at change".

Step 1: Simulating the HMM Output (The Confusion Network)

Imagine we have two audio documents:

- **Doc A**: The user said "Global warming".
  - The HMM is clear.
- **Doc B**: The user said "Climate change".
  - The HMM is confused - it thinks it might be "Climb at change".

Step 2: The "1-Best" Failure

- If we just took the top scoring words (classic Viterbi)
  - --> We would index the wrong text for Doc B.

```
--- Standard Viterbi (1-Best) Transcripts ---
Doc A: 'global warming'
Doc B: 'climb at change'
```

Step 1: Simulating the HMM Output (The Confusion Network)

Imagine we have two audio documents:

- **Doc A**: The user said "Global warming".
  - The HMM is clear.
- **Doc B**: The user said "Climate change".
  - The HMM is confused - it thinks it might be "Climb at change".

Step 2: The "1-Best" Failure

- If we just took the top scoring words (classic Viterbi)
  - --> We would index the wrong text for Doc B.

```
--- Standard Viterbi (1-Best) Transcripts ---
Doc A: 'global warming'
Doc B: 'climb at change'
```

Observation: If you search for "Climate", you will NOT find Doc B.

Step 3: Probabilistic Indexing (The Solution)

- We calculate the Expected Term Frequency (Expected TF).
  - Instead of counting 1 if the word appears and 0 if not, we sum the probabilities.

$$\text{Expected TF}(t, d) = \sum_{slot \in d} P(t|slot)$$

```
--- Probabilistic Inverted Index ---
Entries for 'climate': [('Doc_B (Noisy)', 0.45)]
```

Step 3: Probabilistic Indexing (The Solution)

- We calculate the Expected Term Frequency (Expected TF).
    - Instead of counting 1 if the word appears and 0 if not, we sum the probabilities.

$$\text{Expected TF}(t, d) = \sum_{slot \in d} P(t|slot)$$

```
--- Probabilistic Inverted Index ---
Entries for 'climate': [('Doc_B (Noisy)', 0.45)]
```

Step 4: Execute the Search

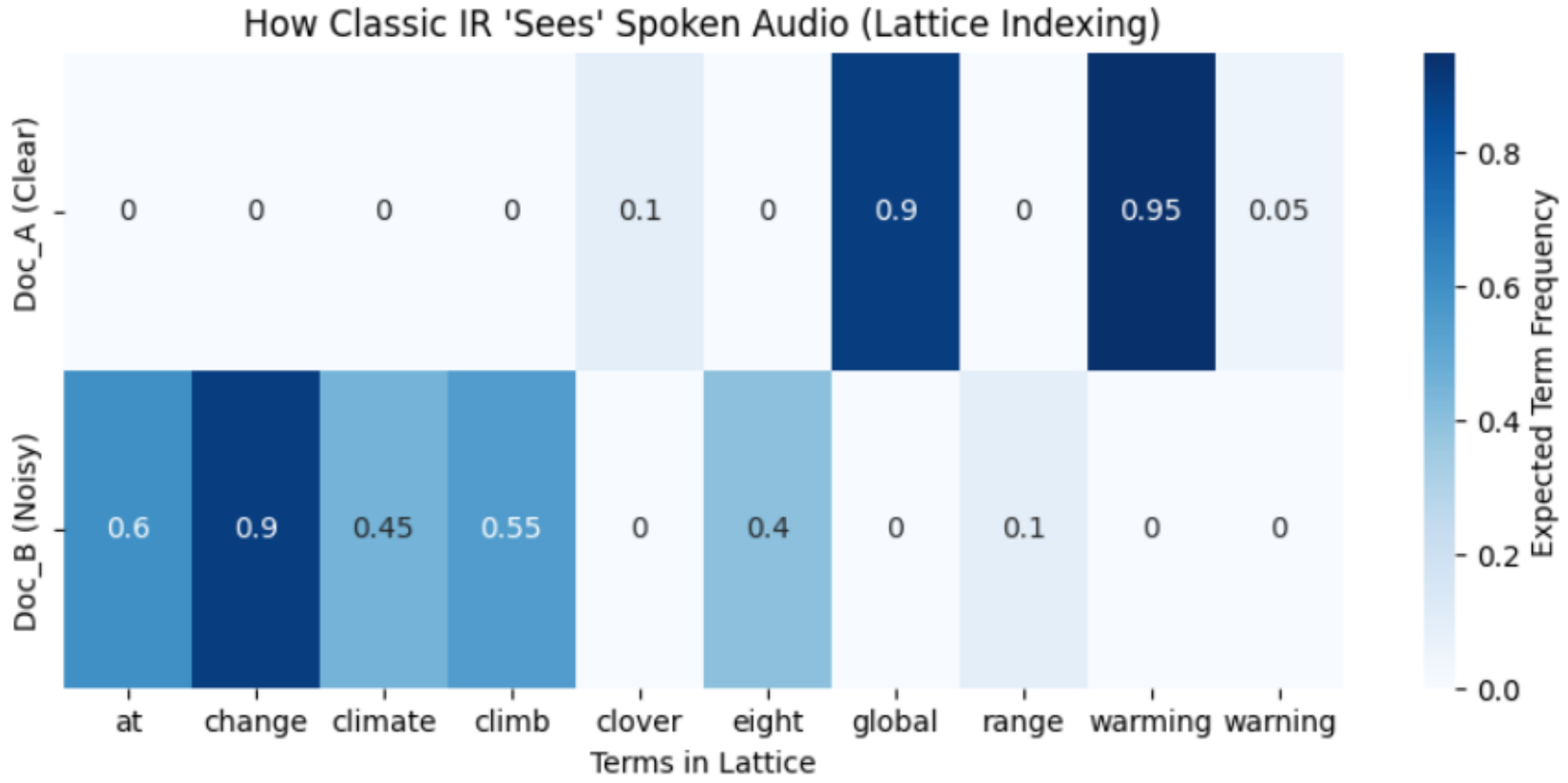
Now we search.

- Even though "Climb" won the probability contest in Doc B, "Climate" still has a score of 0.45.
- This allows the search engine to retrieve it.

```
--- Search Results for 'climate' ---
Found in Doc_B (Noisy) with relevance score: 0.4500
```

Visualizing the Integration

- This heatmap shows how IR sees the documents.
- It's no longer binary (Present/Absent), but a "cloud" of probability.



Summary: The "Phonetic" Integration

- **Lattices**: The IR system didn't trust the HMM blindly. It indexed the uncertainty.
- **Phonetic Expansion**: Another classic trick (not shown in code but conceptually similar) was Phonetic Hashing

If the user typed "Zelenskyy", the system converted the query to phonemes: Z EH L EH N S K IY.

- The system then searched the audio for that sequence of sounds, bypassing text entirely.
- This is critical for names and **Out-Of-Vocabulary** words that the **HMM dictionary** didn't contain.

classic IR & traits like Gender

- Such traits are treated as **Structured Metadata** (or "Fields") attached to the Unstructured Text.
    - Modern systems might use vector embeddings to capture "feminine style"
- Classic systems rely on a strict
  "Classify → Tag → Filter" pipeline.
- Here is the End-to-End workflow of how a Spoken Document Retrieval system
  handle specific traits like gender.

classic IR & traits like Gender

- Such traits are treated as **Structured Metadata** (or "Fields") attached to the Unstructured Text.
    - Modern systems might use vector embeddings to capture "feminine style"
- Classic systems rely on a strict
  "Classify → Tag → Filter" pipeline.
- Here is the End-to-End workflow of how a Spoken Document Retrieval system handle specific traits like gender.

The Architecture: The "Hybrid" Index

- Classic systems maintain two parallel data structures:
    - The Inverted Index:
      For the text (e.g., "climate" → Doc 1, Doc 2).
    - The Forward Index / Document Store:
      For the attributes (e.g., Doc 1 → Gender: Female).
- The search process was a boolean operation: Intersection.
  Query = (Text contains "Hello") AND (Gender == "Female")

Step 1: The Metadata Tagger (Simulation)

- **Before indexing**, the audio passes through the Gender Classifier (the GMM/SVM we built in the previous turn).
- **This classifier outputs** a Tag (Male/Female) which is attached to the document.
- We will simulate a dataset of spoken documents where the text comes from ASR and the gender comes from our Signal Processing classifier.

```
--- Input: Spoken Documents with Classified Metadata ---
   id                          text gender
0   0    global warming is a crisis      F
1   1     the weather is nice today      M
2   2   global markets are changing      M
3   3  warming trends in the arctic      F
4   4    warning signs of inflation      F
```

Step 2: Building the Hybrid Index

- We build a classic Inverted Index for the words,
  but we also build a BitMap or Field Store for the gender.
- This allows for lightning-fast filtering.

```
--- Internal Structure ---
Inverted Index for 'warming': [0, 3]
Gender Index for 'F': {0, 3, 4}
```

Step 3: The Search (Boolean Intersection)

- **The user asks**: "Show me documents about 'warming' spoken by a 'Female'."
- **In classic IR, this is a two-step** set operation:
    1. **Retrieve**: Get all docs containing "warming".
    2. **Filter**: Intersect that list with the list of "Female" docs.

```
--- Input: Spoken Documents with Classified Metadata ---
   id                          text gender
0   0    global warming is a crisis      F
1   1     the weather is nice today      M
2   2   global markets are changing      M
3   3  warming trends in the arctic      F
4   4    warning signs of inflation      F
```

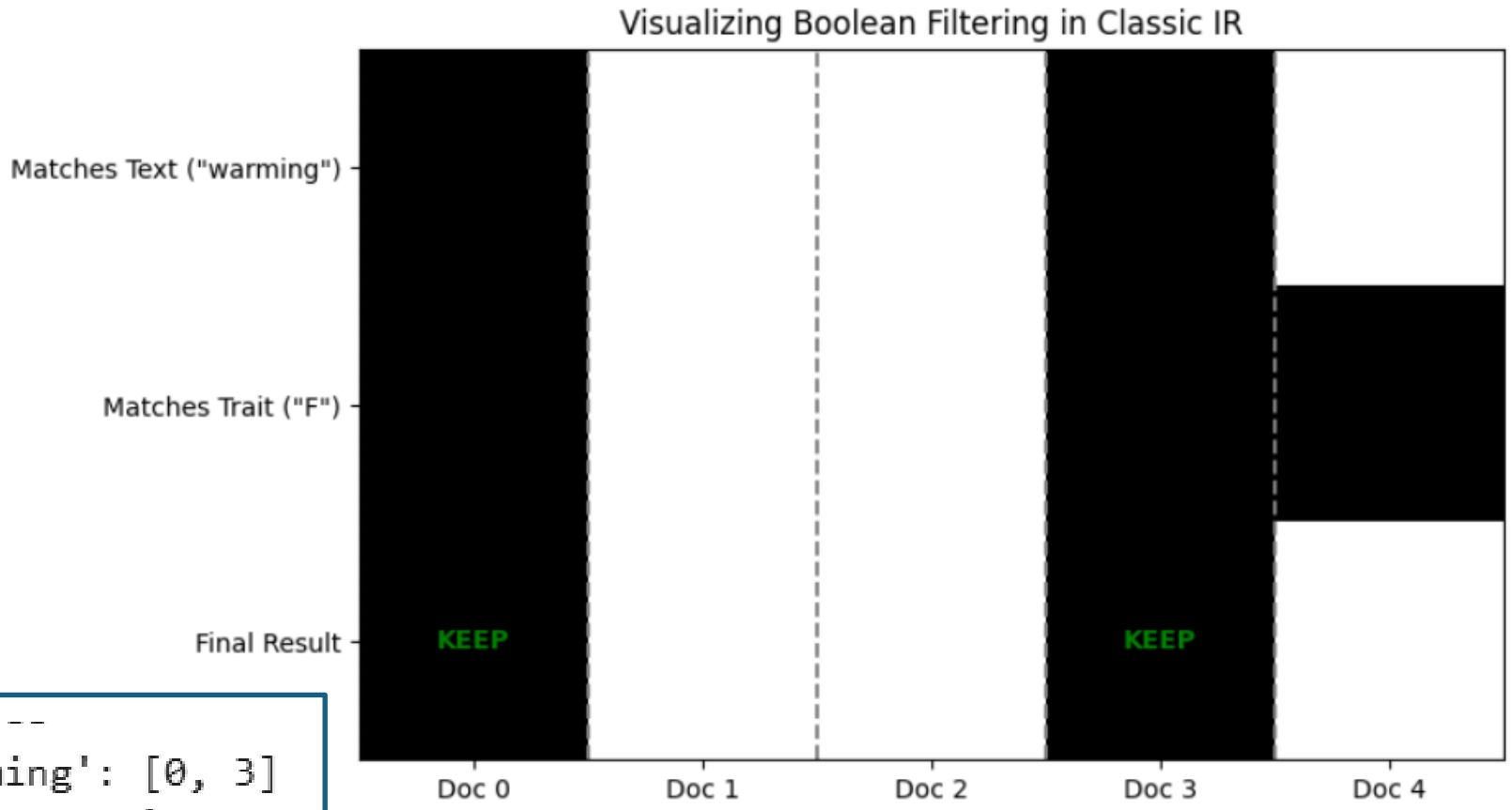
```
--- Internal Structure ---
Inverted Index for 'warming': [0, 3]
Gender Index for 'F': {0, 3, 4}
```

```
1. Text Matches for 'warming': {0, 3}
2. Gender Matches for 'F': {0, 3, 4}

>>> Final Results: Docs [0, 3]
    Doc 0: global warming is a crisis
    Doc 3: warming trends in the arctic
```

Visualization of the Search Logic

This visualization demonstrates exactly how the system effectively "slices" the database using the trait as a mask.




```
--- Internal Structure ---
Inverted Index for 'warming': [0, 3]
Gender Index for 'F': {0, 3, 4}
```

Step 4: Faceted Search (The "Aggregation" Feature)

- A hallmark of classic IR (popularized by e-commerce) was Faceting.
- If you search for "Global", the system tells you:
  "I found 2 docs. 1 by Male, 1 by Female."
- This was calculated by looking at the traits of the result set.

```
--- Input: Spoken Documents with Classified Metadata ---
   id                          text gender
0   0    global warming is a crisis      F
1   1     the weather is nice today      M
2   2   global markets are changing      M
3   3  warming trends in the arctic      F
4   4    warning signs of inflation      F
```

```
--- Internal Structure ---
Inverted Index for 'warming': [0, 3]
Gender Index for 'F': {0, 3, 4}
```

```
1. Text Matches for 'global': {0, 2}

--- Faceted Navigation for 'global' ---
Distribution: {'F': 1, 'M': 1}
```

Step 6: "Boosting" (Soft Filtering)

- Sometimes you don't want to exclude Males, but just rank Females higher.
- Classic IR handled this via *Boost Factors*.
  Formula: $Score = \text{TF-IDF} \times \text{BoostFactor}$
  If Gender == Female, Boost = 2.0. Else, Boost = 1.0.

The user asks: "Show me documents about "Global"spoken by a 'Female'."

```
--- Input: Spoken Documents with Classified Metadata ---
   id                          text gender
0   0    global warming is a crisis      F
1   1     the weather is nice today      M
2   2   global markets are changing      M
3   3  warming trends in the arctic      F
4   4    warning signs of inflation      F
```

```
 1. Text Matches for 'global': {0, 2}

 --- Faceted Navigation for 'global' ---
 Distribution: {'F': 1, 'M': 1}
```

```
--- Boosted Search (Prefer Female, but keep Male) ---
Doc 0 (F): Score 2.0
Doc 2 (M): Score 1.0
```

Summary

- In classic IR systems, traits like gender were not "learned" representations.

They were:

- **Classified upstream** (during the indexing phase) by a specific model (GMM/SVM).
- **Stored as explicit fields** (Categorical Data).
- **Utilized via Boolean Logic** (Filtering) **or Arithmetic** (Score Boosting).
- This is distinct from modern Vector Search, where "gender" might be implicit in the embedding space itself.

Classic Spoken Document Retrieval (SDR) system - a classic multimedia IR

The core challenge in a mixed Audio/Text system is the Mismatch:

- **Text Documents** consist of perfect words (e.g., "Weather").
- **Audio Documents (after ASR)** consist of noisy phonetic sequences (e.g., "W EH DH ER").

To fix this, classic systems uses a `Phonetic Bridge`.

Phonetic Bridge - Converte everything (both text and audio) into a common Phonetic Representation (often N-grams) before indexing.

Classic Spoken Document Retrieval (SDR) system - a classic multimedia IR

The core challenge in a mixed Audio/Text system is the Mismatch:

- **Text Documents** consist of perfect words (e.g., "Weather").
- **Audio Documents (after ASR)** consist of noisy phonetic sequences (e.g., "W EH DH ER").

To fix this, classic systems uses a `Phonetic Bridge`.
Phonetic Bridge - Converte everything (both text and audio) into a common Phonetic Representation (often N-grams) before indexing.

The Architecture

1. **Ingestion**:
    - **Text Docs**: Convert Text -> Phonemes (Grapheme-to-Phoneme / G2P).
    - **Audio Docs**: Convert Audio -> Phonemes (via HMM-ASR)
2. **Indexing**: Create a TF-IDF index of Phonetic N-Grams (e.g., `W-EH-DH`).
3. **Retrieval**: The user can type or speak. Both are converted to phonemes and matched against the index.

Step 1: The "Phonetic Bridge" Simulation

- we will simulate the two critical classic components:
    1. **G2P (Text-to-Phoneme)**: Perfect conversion.
    2. **ASR (Audio-to-Phoneme)**: Noisy conversion (simulating recognition errors).

Step 1: The "Phonetic Bridge" Simulation

- we will simulate the two critical classic components:
    1. **G2P (Text-to-Phoneme)**: Perfect conversion.
    2. **ASR (Audio-to-Phoneme)**: Noisy conversion (simulating recognition errors).

```
--- The Phonetic Bridge ---
[Text] Original: PROJECT ALPHA DEADLINE FRIDAY
       System Sees: P R AA JH EH K T AE L F AH D EH D L AY N F R AY D EY
------------------------------
[Audio] Original: URGENT CALL ME PROJECT ALPHA
       System Sees: ER JH AH N D G AO L M IY B R AA JH EH G T AE L TH AH
------------------------------
```

Observation: Notice that

- Doc_Audio_1 (Project Alpha) might have become `B R AA JH EH K T` (**P** became **B**).
- A strict text search would fail, but a **phonetic search** might survive.

Step 2: Indexing (Phonetic N-Grams)

- We don't index whole phoneme "words".
- We index **sliding windows** (trigrams) of phonemes.
    - This makes the system robust.
- If "**P**ROJECT" becomes "**B**ROJECT"
    - the trigrams `R-AA-JH`, `AA-JH-EH`, etc., still match!

```
--- The Phonetic Bridge ---
[Text] Original: PROJECT ALPHA DEADLINE FRIDAY
       System Sees: P R AA JH EH K T AE L F AH D EH D L AY N F R AY D EY
------------------------------
[Audio] Original: URGENT CALL ME PROJECT ALPHA
       System Sees: ER JH AH N D G AO L M IY B R AA JH EH G T AE L TH AH
------------------------------
```

```
Index Vocabulary Size: 36 Phonetic Trigrams
```

Step 3: The Mixed Query (Audio Input)

- The user speaks a query: "Project Alpha".
- Because of background noise or a different accent, the ASR hears it imperfectly.

```
User Said: 'PROJECT ALPHA'
ASR Heard: 'B R AA JH EH K T AE L F AH' (Note the P->B error)

--- Search Results ---
            id   type                          content     score
0   Doc_Text_1   Text  PROJECT ALPHA DEADLINE FRIDAY  0.552440
1  Doc_Audio_1  Audio   URGENT CALL ME PROJECT ALPHA  0.219461
```

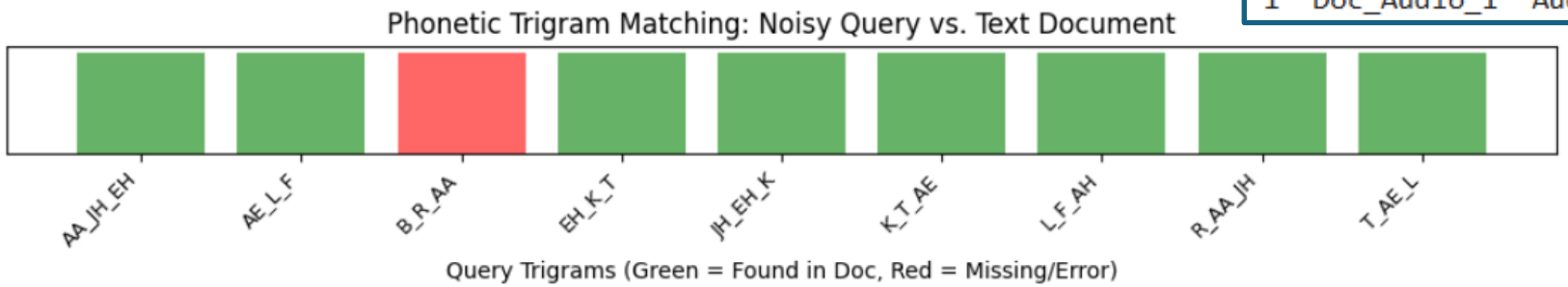
Step 3: The Mixed Query (Audio Input)

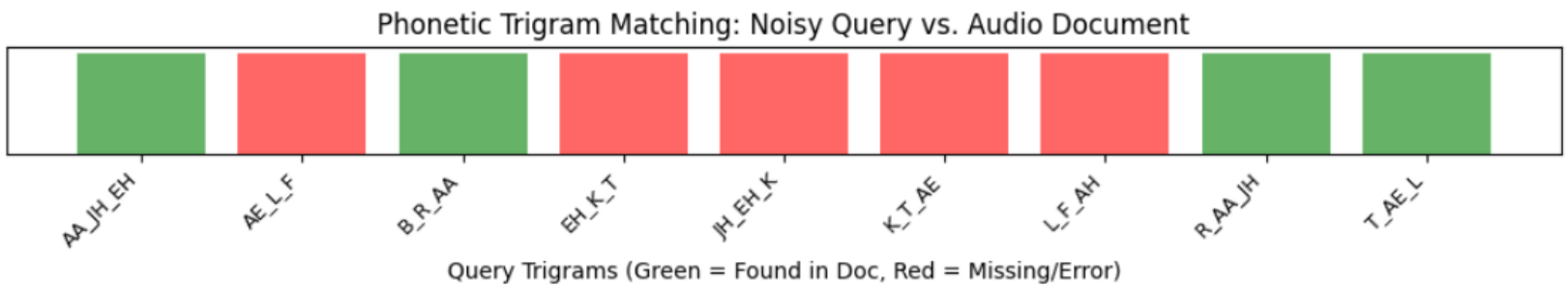
```
--- The Phonetic Bridge ---
[Text] Original: PROJECT ALPHA DEADLINE FRIDAY
       System Sees: P R AA JH EH K T AE L F AH D EH D L AY N F R AY D EY
------------------------------
[Audio] Original: URGENT CALL ME PROJECT ALPHA
       System Sees: ER JH AH N D G AO L M IY B R AA JH EH G T AE L TH AH
------------------------------
```

- The user speaks a query: "Project Alpha".
- the ASR hears it imperfectly.

```
User Said: 'PROJECT ALPHA'
ASR Heard: 'B R AA JH EH K T AE L F AH' (Note the P->B error)

--- Search Results ---
           id   type                          content     score
0   Doc_Text_1   Text  PROJECT ALPHA DEADLINE FRIDAY  0.552440
1  Doc_Audio_1  Audio   URGENT CALL ME PROJECT ALPHA  0.219461
```





Analysis of the Results

1. **The Bridge Works**: The query was Audio (converted to noisy text).
   Doc 1 was pure Text. They matched!
2. **Robustness**:
   - Look at the Green Bars in the visualization.
   - The user said "**P**roject" (Start with P).
     The ASR heard "**B**roject" (Start with B).
   - The trigram `B_R_AA` (Red/Miss) failed to match the document's `P_R_AA`.
   - However, the subsequent trigrams `R_AA_JH`, `AA_JH_EH`, `JH_EH_K` did match.
   - Because classic IR uses *voting* (Cosine Similarity of many trigrams), the document is retrieved despite the error at the start of the word.

This demonstrates the classic "**Phonetic N-Gram**" approach,
which allowed search engines to retrieve radio broadcasts,
voicemails, and text documents in a single unified result list.

Classic (pre-Deep Learning) Multimedia Information Retrieval (MMIR) system

- This is a comprehensive demonstration of a Classic Multimedia Information Retrieval (MMIR) system.
- In the classic era , these systems relied on **Late Fusion**.
- The system was essentially three separate search engines
  (Text, Audio, Vision) glued together.
    - **Text Engine**: Standard Vector Space Model (TF-IDF).
    - **Audio Engine**: Phonetic Lattice Indexing (handling ASR noise).
    - **Vision Engine**: Low-level features (Color/Texture Histograms).
- The **final relevance score** is a weighted sum:

$$S_{final} = w_t \cdot S_{text} + w_a \cdot S_{audio} + w_v \cdot S_{visual}$$

We will build a "News Archive Search" where documents can be:

- **Video** (Audio+Image)
- **Podcasts** (Audio only) or
- **Photos** (Image+Tags).

The Scenario: "The Ambiguous 'Bass'"

This is a classic Multimedia IR problem where a single word has multiple meanings, and we need Audio and Visual signals to disambiguate it.

User Query:

- **Text**: "Bass" (Ambiguous: Fish? Instrument? Frequency?)
- **Audio**: Input is a Low-Frequency Hum (User hums a low note).
- **Image**: Input is a Brown Wooden Texture (User uploads a photo of wood).
- **Intent**: User wants a Double Bass (Instrument), not a fish or a speaker.

User Query:

- **Text**: "Bass" (Ambiguous: Fish? Instrument? Frequency?)
- **Audio**: Input is a Low-Frequency Hum (User hums a low note).
- **Image**: Input is a Brown Wooden Texture (User uploads a photo of wood).
- **Intent**: User wants a Double Bass (Instrument), not a fish or a speaker.

The Database Candidates

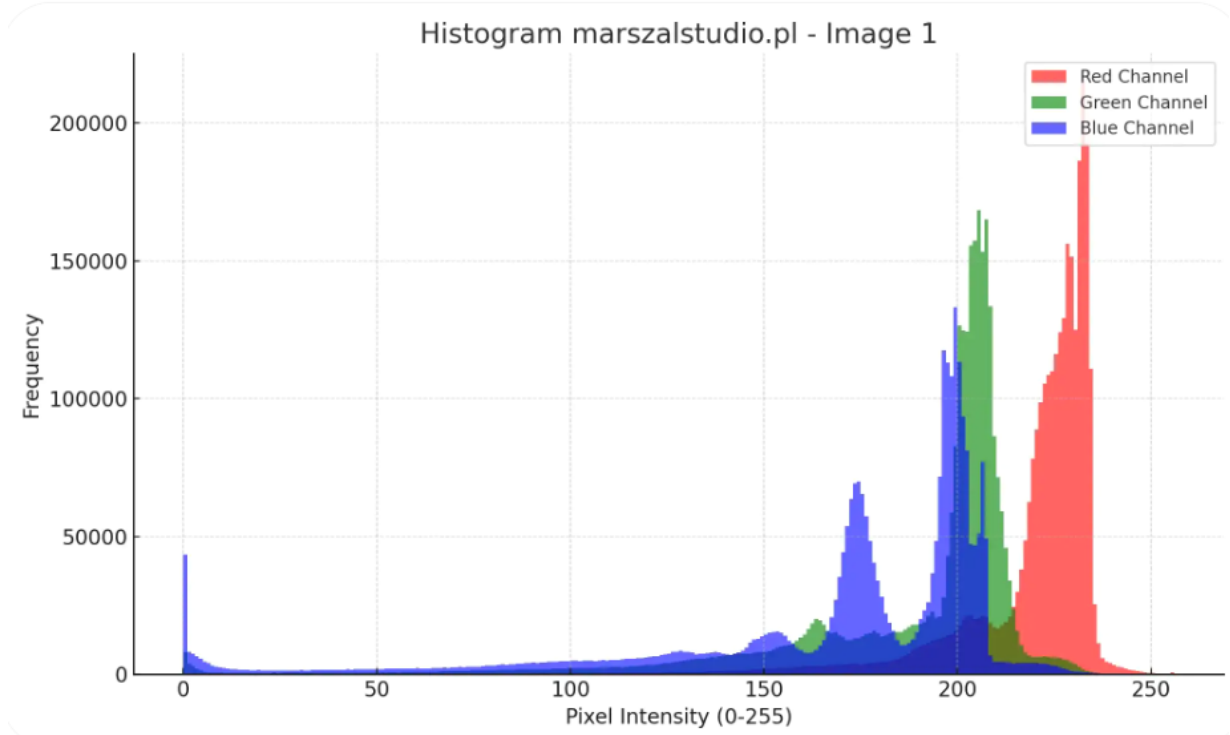
1. **Doc A (Fishing Blog)**:
    - *Text*: "Caught a giant Bass in the river." (Strong Text Match)
    - *Audio*: Sound of splashing water (High Pitch - Mismatch).
    - *Visual*: Photo of a Fish (Silver/Blue - Mismatch).
2. **Doc B (Club Music)**:
    - *Text*: "Drop the heavy beat." (Weak Text Match).
    - *Audio*: Electronic Sub-bass (Low Pitch - Strong Match).
    - *Visual*: Neon Lights (Bright Colors - Mismatch).
3. **Doc C (Jazz Concert)**:
    - *Text*: "The Bass player plucked the strings." (Strong Text Match).
    - *Audio*: Acoustic low note (Low Pitch - Strong Match).
    - *Visual*: Wooden Instrument (Brown - Strong Match).

Demonstration: 3-Way Late Fusion

Three "Classic" feature extractor simulation:

1. **Text**: **TF-IDF**.
2. **Audio: Zero Crossing Rate (ZCR)** or **Spectral Centroid**.
    - these were standard features to distinguish "Low/Thud" sounds from "High/Splash" sounds.
    - We will simulate a "Pitch/Frequency" score.
3. **Vision: Color Histograms** (RGB).

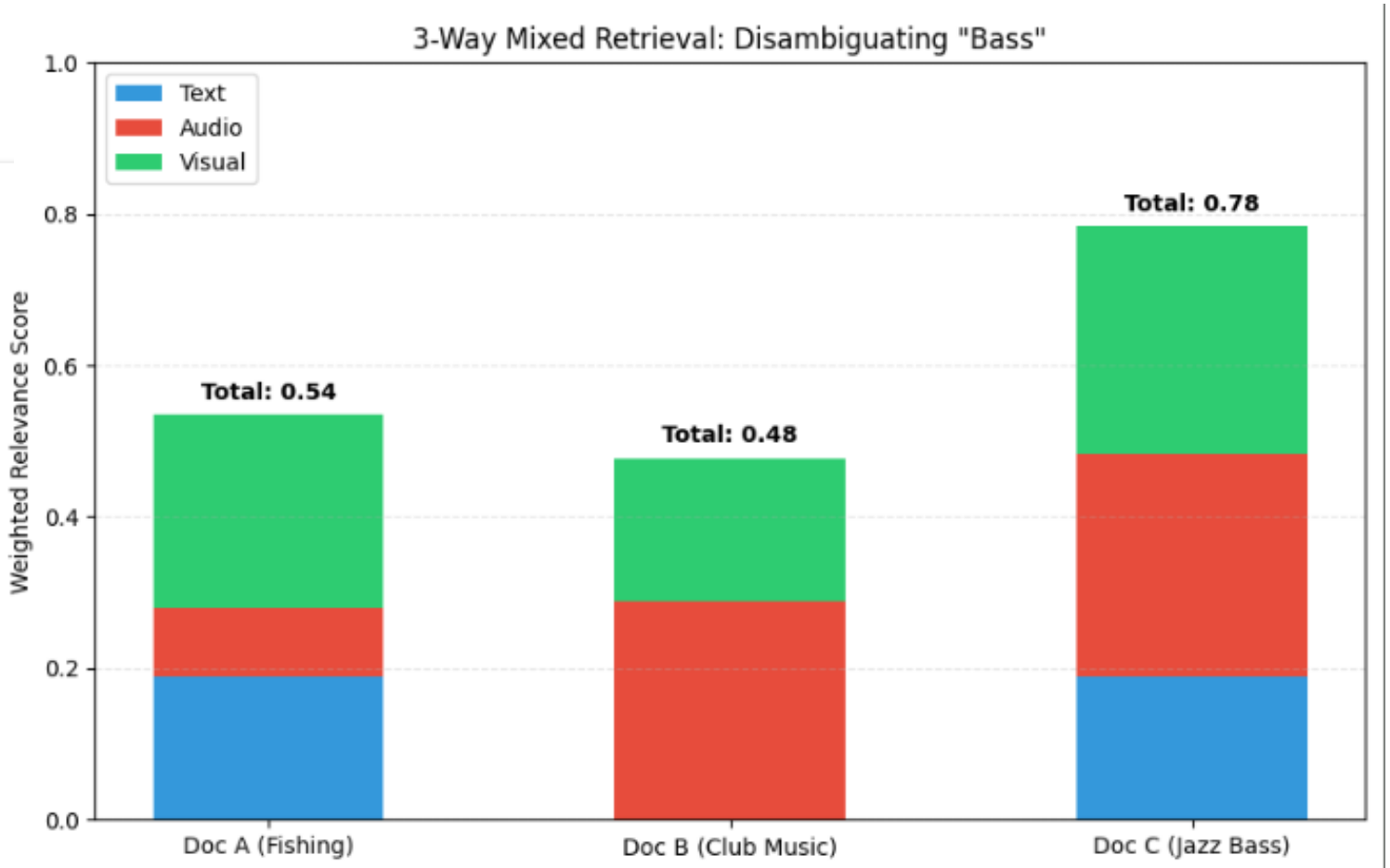
Vision – color histogram



Demonstration: 3-Way Late Fusion

Three "Classic" feature extractor simulation:

1. **Text**: **TF-IDF**.
2. **Audio: Zero Crossing Rate (ZCR)** or **Spectral Centroid**.
    - these were standard features to distinguish "Low/Thud" sounds from "High/Splash" sounds.
    - We will simulate a "Pitch/Frequency" score.
3. **Vision: Color Histograms** (RGB).



	Document	Text (40%)	Audio (30%)	Visual (30%)	Total Score
0	Doc A (Fishing)	0.189	0.090	0.256	0.536
1	Doc B (Club Music)	0.000	0.288	0.190	0.478
2	Doc C (Jazz Bass)	0.189	0.294	0.300	0.783

Analysis of the 3-Way Mix

- The stacked bar chart tells the complete story of how Late Fusion saves the day.

1. **Doc A (Fishing)**:
    - **Text (Blue)**: High score (Has "Bass").
    - **Audio (Red)**: Fail. Splashing water (800Hz) is too high-pitched compared to the user's hum (100Hz).
    - **Visual (Green)**: Fail. Silver fish color does not match brown wood.
    - ***Result***: It started strong with text but lost due to mismatched multimedia signals.
2. **Doc B (Club Music)**:
    - **Text (Blue)**: Fail. It doesn't contain the word "Bass" (it used "Beat").
    - **Audio (Red)**: Success. The low electronic thud (60Hz) is very close to the user's hum (100Hz).
    - **Visual (Green)**: Fail. Neon green doesn't look like wood.
    - ***Result***: A "Mixed" result. Good audio, bad everything else.
3. **Doc C (Jazz Bass) - The Winner**:
    - **Text (Blue)**: Success. Contains "Bass".
    - **Audio (Red)**: Success. The acoustic cello/bass (120Hz) matches the hum.
    - **Visual (Green)**: Success. The brown wood matches the user's visual hint.
    - ***Result***: All three signals align, creating the tallest bar.

Why this is "Classic" IR?

- **Explicit Feature Engineering**:
  - Manually (rule based) defined "Frequency" for audio and
  - Manually (rule based) defined "RGB" for vision.
  - **We didn't learn these features**.
- **Linear Combination**:
  - The final logic was just:

    0.4Text + 0.3Audio + 0.3Vision.
  - Modern systems would use a neural network to learn non-linear relationships
    - e.g., "Bass" + "Splash" -> "Fish".

Until the next time ☺