Audio Generation

Generative Artificial Intelligence

M. Sc. Angewandte Informatik

Prof. Dr. Marc Aubreville

Hochschule
Flensburg
University of
Applied Sciences

What's in this chapter

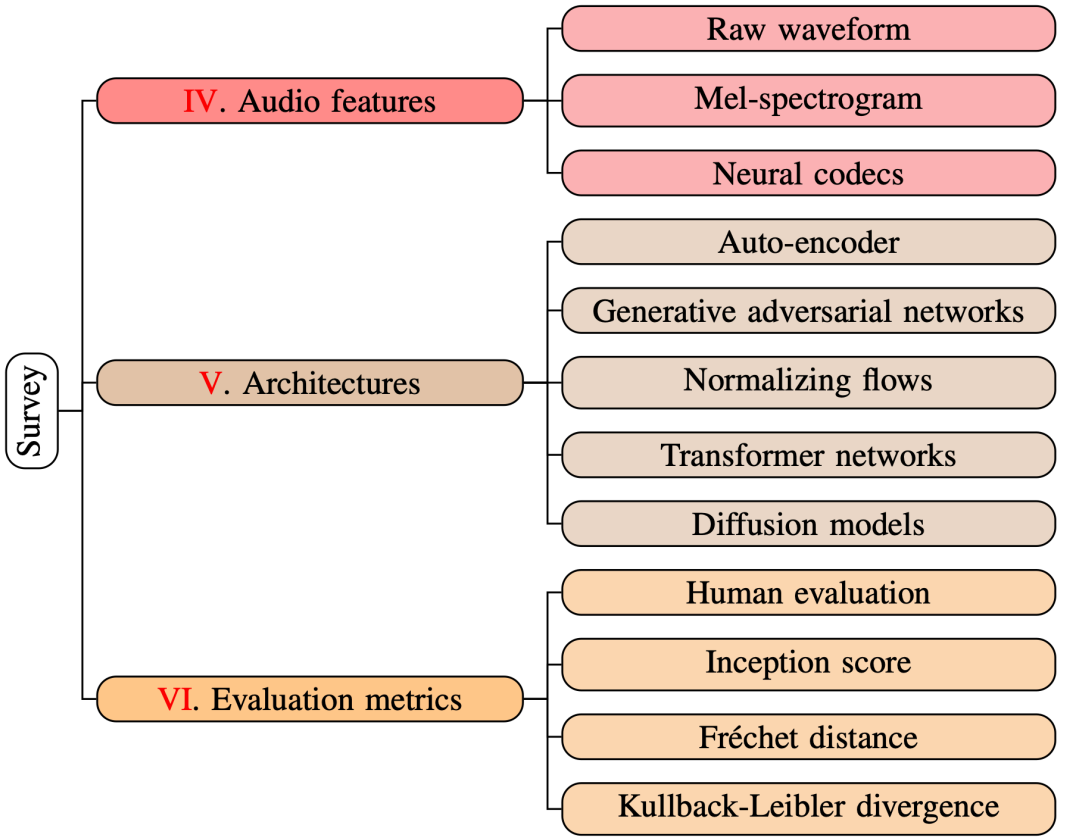
- Audio signals and their representation
  - Time signals
  - Frequency decomposition
  - Mel spectrogram representations
- Generation of audio
- Generation of speech / recognition of speech
- Generation of music

Hochschule Flensburg
University of Applied Sciences

Intro: Not so different, after all

Hochschule Flensburg
University of Applied Sciences

- In the field of audio, we meet with some already known concepts and methods.
- The same architectures we already know from image and text generation are also applicable here.
- However, we are working with a different modality.
- And this means, we need to dive into the peculiarities of this first.



https://arxiv.org/pdf/2406.00146

Use cases in audio generation

- The most prominent use cases are:
  - Text-to-Speech (TTS): Also called text-conditional speech synthesis
  - Music generation:
    - Midi representation generation
    - Raw waveform generation

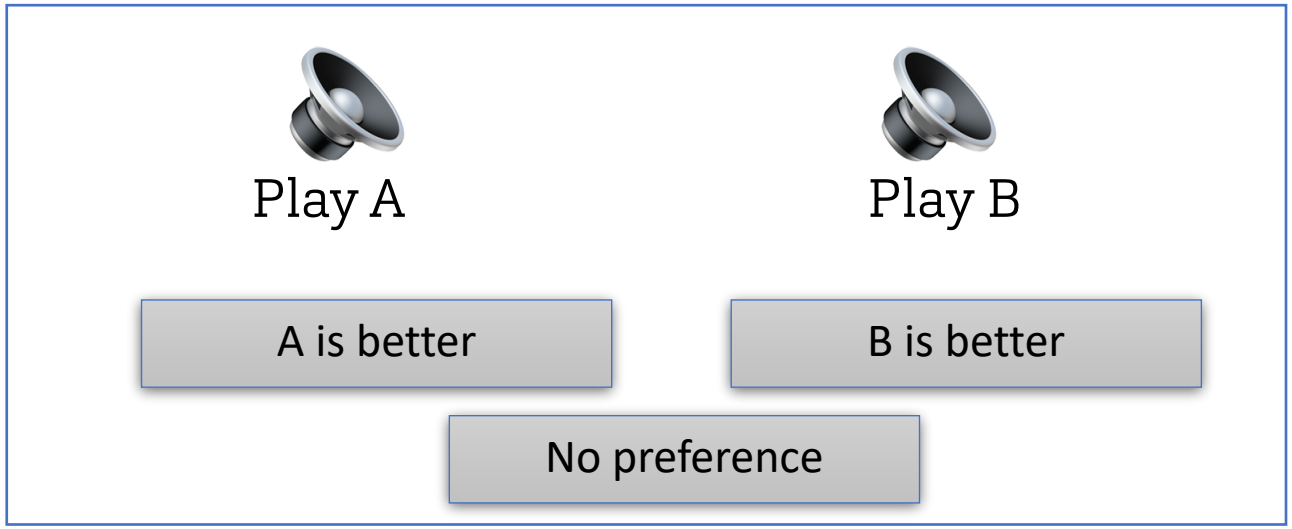
Hochschule
Flensburg
University of
Applied Sciences

Hochschule
Flensburg
University of
Applied Sciences

Metrics and Assessments

A/B Comparisons

- In audio generation and processing, we often ask subjects to rate audio samples blinded.



- This can be utilized to compare against original (unprocessed) samples, perfect audio signals, or even multiple processing schemes against each other.

Mean Opinion Score (MOS)

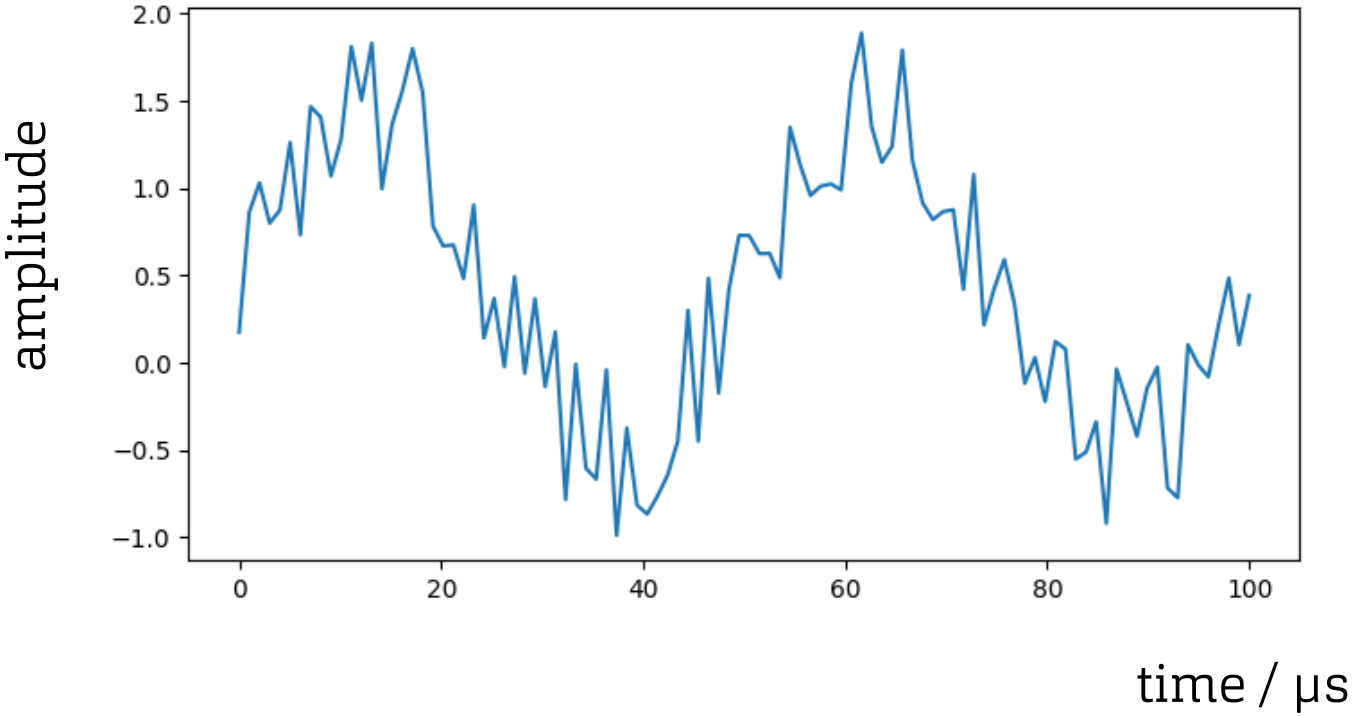
- The mean opinion score is also evaluated using subject tests.
- The subjects have to rate a signal on a Likert scale, typically between 1 and 5.

Hochschule
Flensburg
University of
Applied Sciences

Signals, Time and Frequency Domain

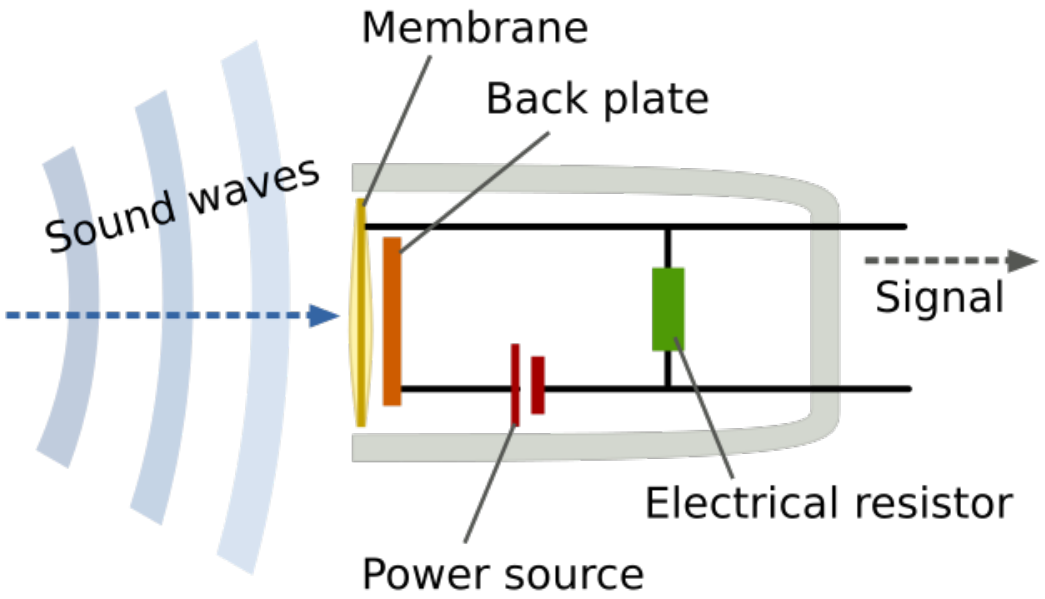
A time signal

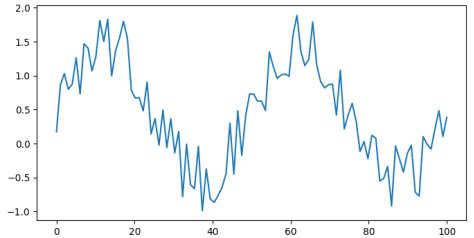
- In audio signal processing, we are dealing with time-dependent signals.
- Each signal comprises of samples that are recorded at a defined sampling rate.



Recording

- We record signals using microphones. These have a membrane that resonates with incoming sound.



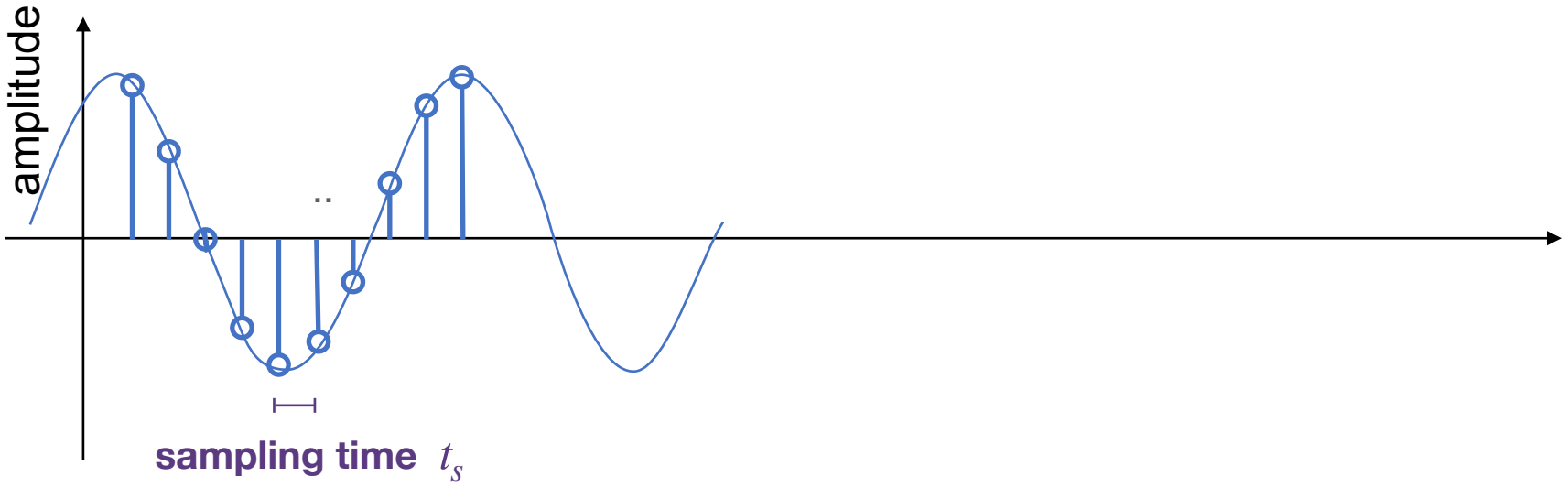


- The amplitude of the sound wave (effectively, a tiny change in air pressure) directly translates to the amplitude of the digital signal.

https://commons.wikimedia.org/wiki/File:Condenser_microphone.svg

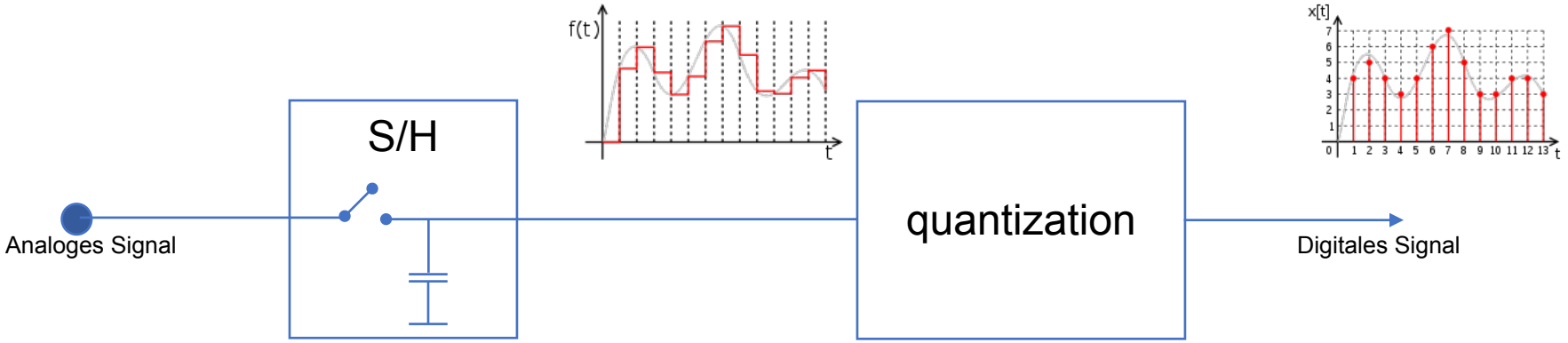
Sampling

- As we can't know the signal's amplitude for continuous time, we have to make regular measurements.
- We call this sampling of the signal. It typically also involves digitization (analog to digital conversion).
- The sampling is done at regular, fixed time intervals.



Analog to digital conversion (A2D)

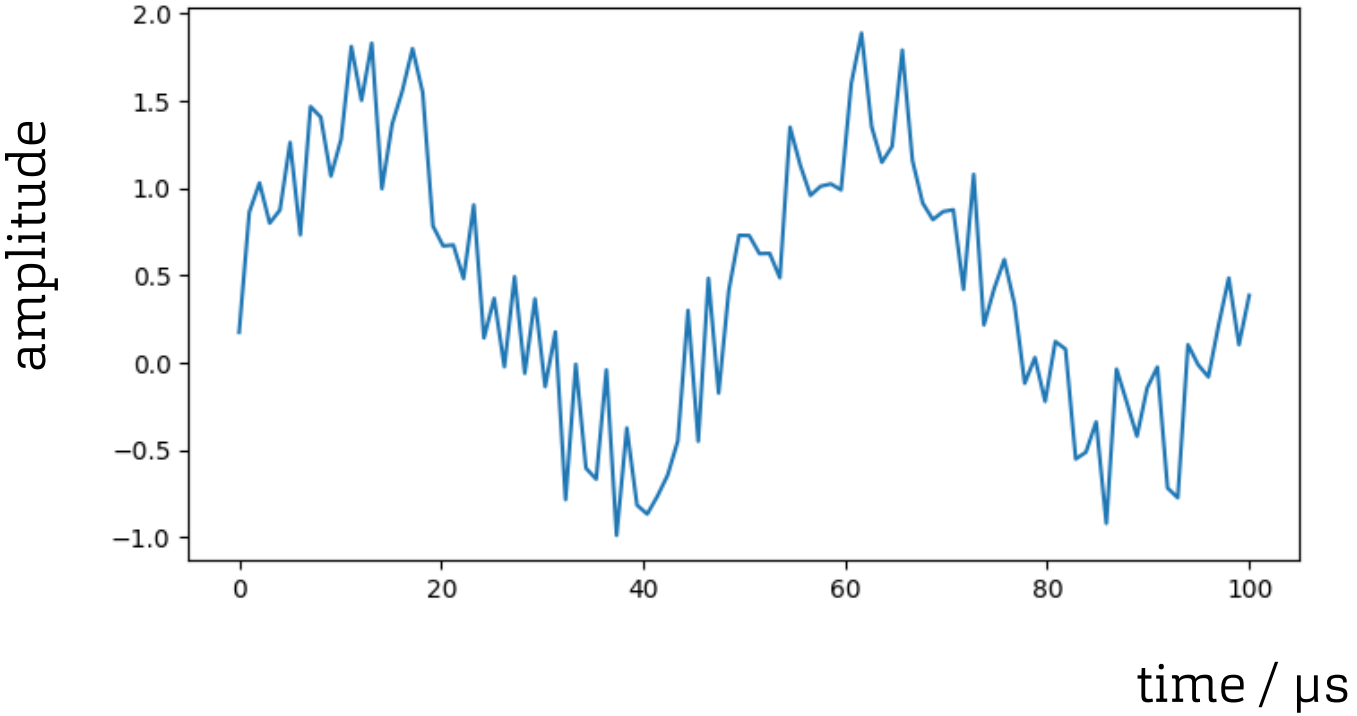
- Analog to digital conversion involves sampling and quantization.



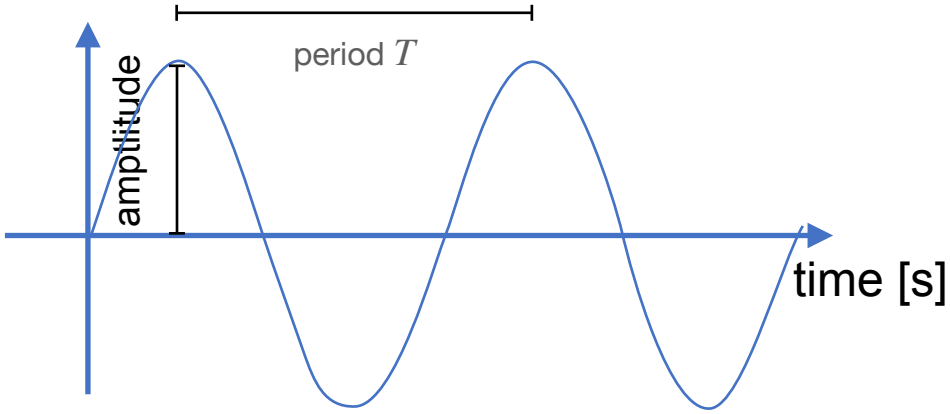
- **1st step: sample and hold** (S/H)
  - Voltage is being "frozen" at dedicated time points so we can measure it.
- **2nd step: quantization**
  - We compare the sampled voltage against thresholds to determine a quantized version of the sample.

Sampling rate

- The sampling rate is the frequency of sampling, i.e., how often per second a sample was recorded.



Some fundamental terms

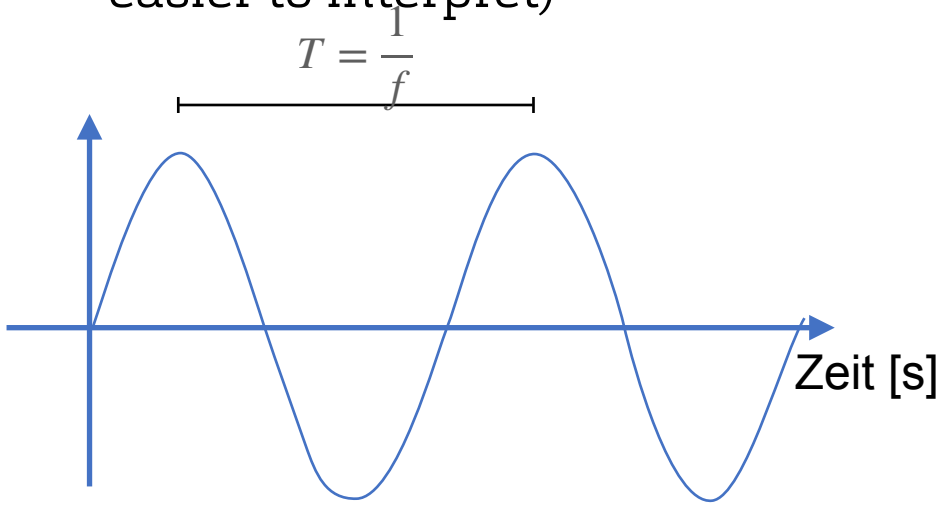


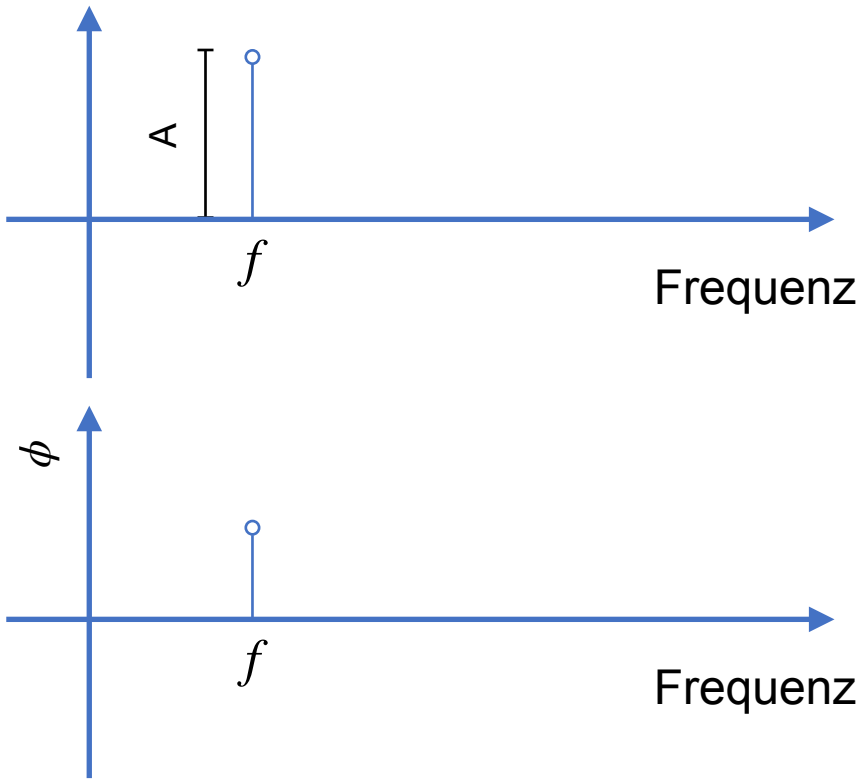
frequency $f = \frac{1}{T}$ Unit: Hertz [Hz]

- Let's take a sinus wave as an example now. We can explain certain terms on it:
  $x(t) = A\sin(2\pi ft + \phi)$
  whereas:
  - A: amplitude
  - f: frequency ($= \frac{1}{T}$)
  - ϕ :phase offset (shift of the sinusoid on the time axis)

Time-frequency dualism

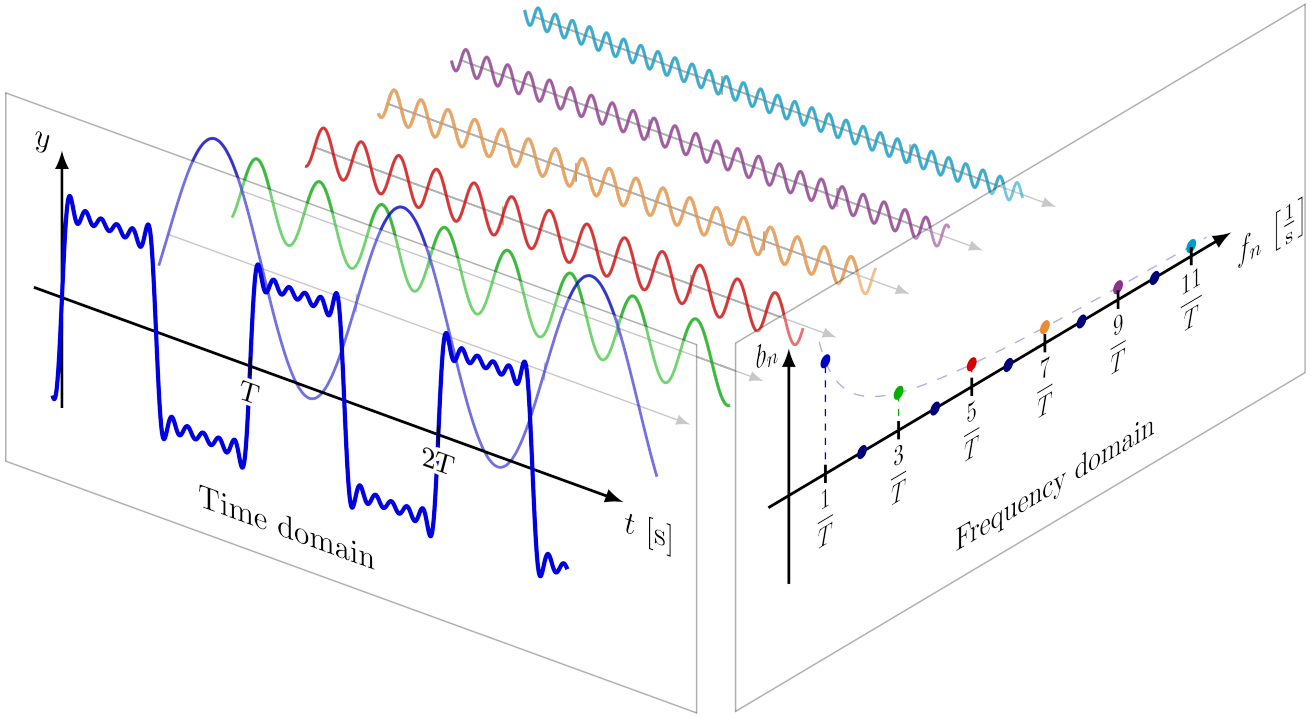
- Any sine wave is defined by the tuple (A, f, ϕ).
- We can alternatively find a representation over frequency.
  - We get the amplitude over the frequency and the phase over frequency.
  - This contains the same information (but is easier to interpret)





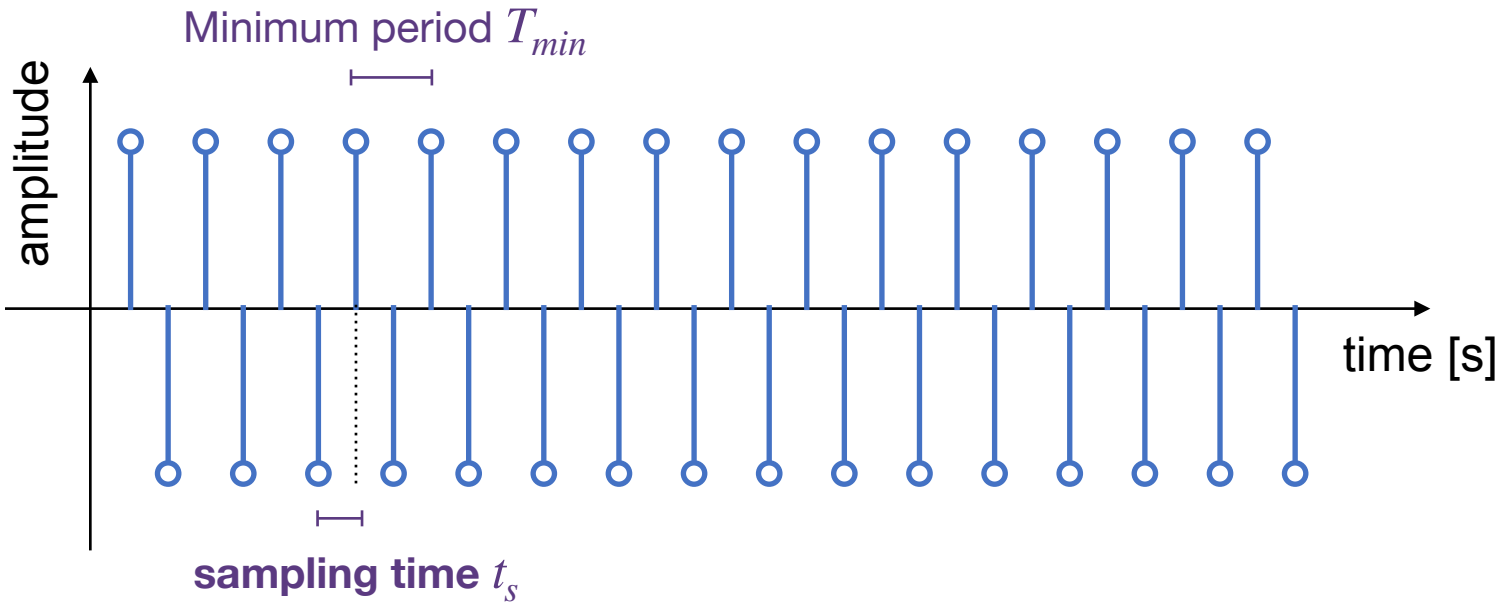
Fourier Decomposition

- Any signal can be constructed by a superposition of sine waves (with respective phase,frequency, amplitude).
- We can consequently also decompose any signal into the contained sine waves.



Maximum frequency in a signal

- Each signal can represent various frequencies.
- The maximum frequency contained in a sample is defined by a alternating impulses:



- From this, we can find the relationship: $T_{min} > 2t_s$

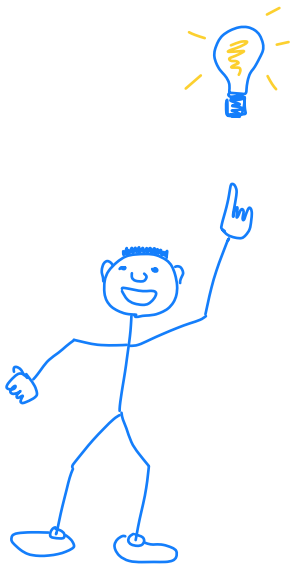
Sampling theorem (Shannon-Nyquist-Theorem)

Hochschule Flensburg
University of Applied Sciences

- We can reformulate $T_{min} > 2t_s$ into a maximum frequency f_{max}:

$$f_{max} = \frac{1}{T_{min}} < \frac{1}{2t_s} = \frac{1}{2} f_s$$

- or: $f_s > 2f_{max}$



A signal that contains no higher frequencies than f_{max}, can be sampled without information loss if $f_s > 2f_{max}$ is respected.

Shannon-Nyquist-Abtasttheorem

Hochschule
Flensburg
University of
Applied Sciences

Frequency decomposition

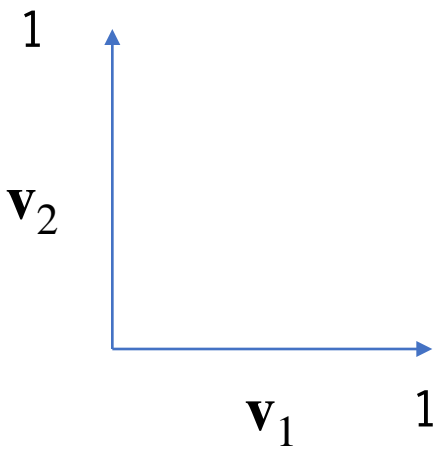
Jetzt packen wir die bisherigen Erkenntnisse zusammen.

A bit of math: Orthogonal vector space

- Let V be an euclidean vector space.
- A set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ in V is orthogonal, if and only if:

  $\langle v_i, v_j \rangle = 0 \quad$ for all $i \neq j, i, j \in \{1, \ldots, k\}$

  ($\langle \cdot, \cdot \rangle$ is a scalar product)



Example for an orthogonal vector space of dimension 2

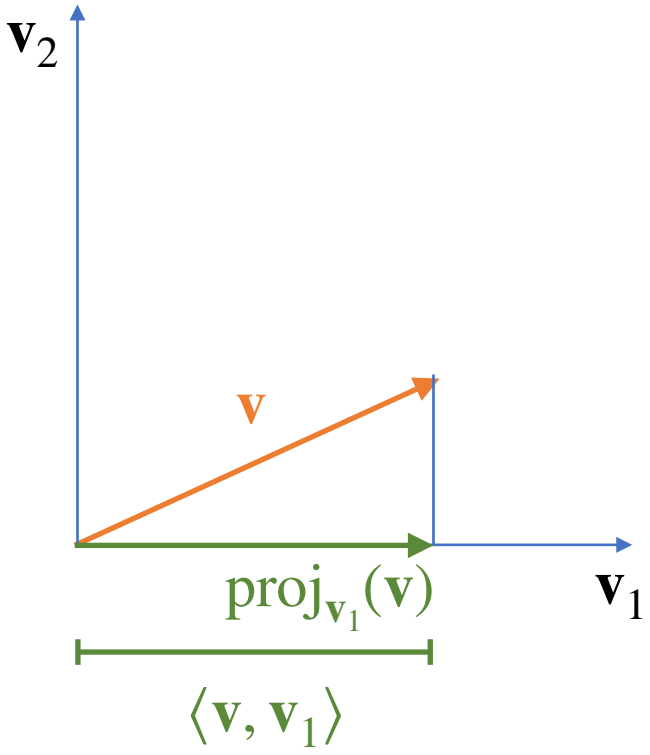
$$\mathbf{v}_2 = [0,1], \mathbf{v}_1 = [1,0]$$

Projection of a vector onto the orthogonal basis

- If we have a vector $\mathbf{v} \in V$ and an orthogonal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of V.
- Any vector v can be projected onto each of the basis vectors:

$$\text{proj}_{\mathbf{v}_i}(v) = \langle \mathbf{v}, \mathbf{v}_i \rangle \mathbf{v}_i$$

- Here:
  - $\langle \mathbf{v}, \mathbf{v}_i \rangle$ is the length of the projection
  - $\mathbf{v}_i$ is the direction o the basis vector

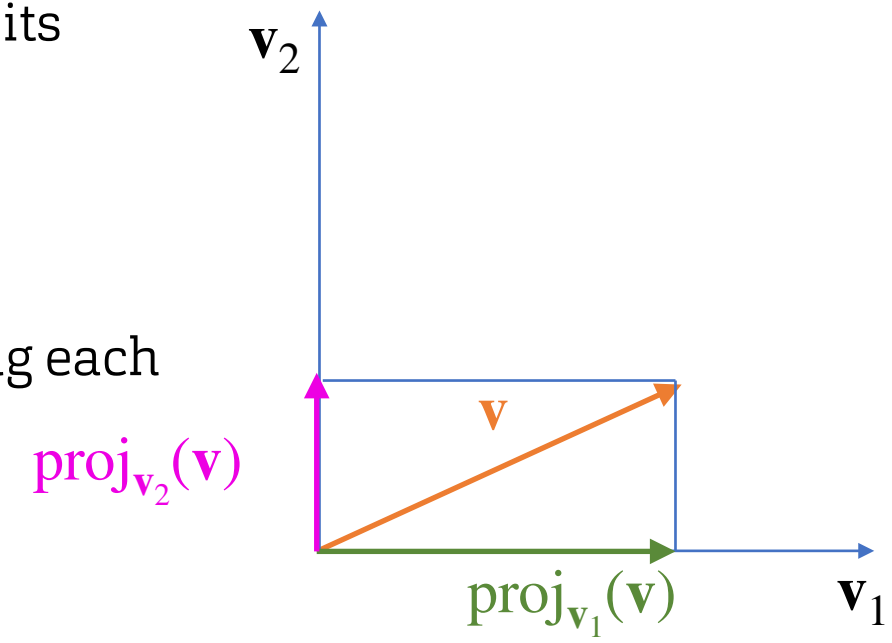


Decomposition into projections onto the basis vectors

- The vector $\mathbf{v}$ can then be written as the sum of its projections onto all basis vectors:

$$\mathbf{v} = \sum_{i=1}^{n} \text{proj}_{\mathbf{v}_i}(v) = \sum_{i=1}^{n} \langle \mathbf{v}, \mathbf{v}_i \rangle \mathbf{v}_i$$

- This allows us to analyze the contribution along each basis vector individually.

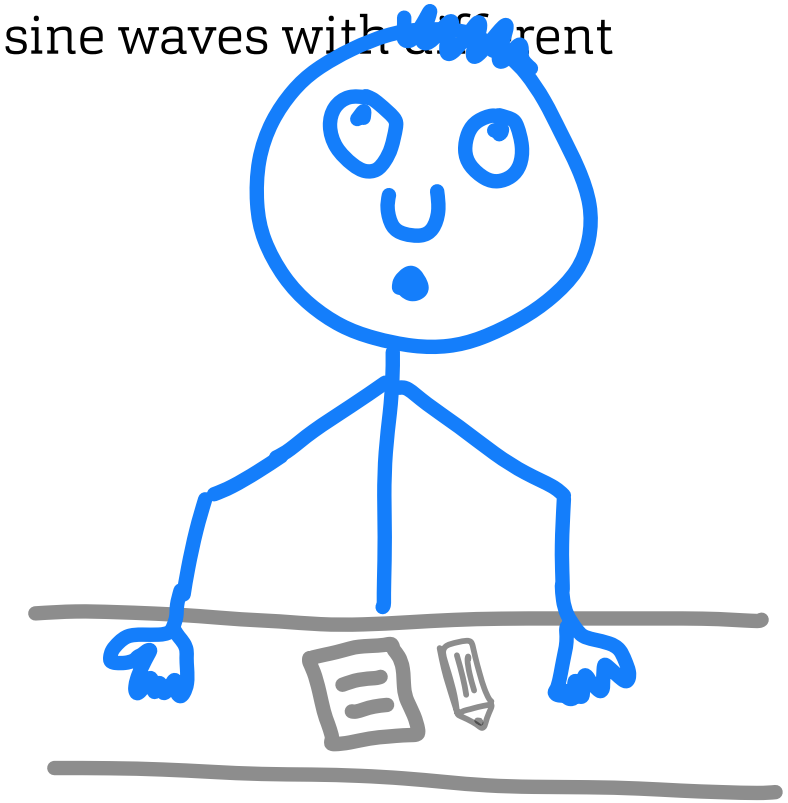


$$\mathbf{v} = \text{proj}_{\mathbf{v}_1} v + \text{proj}_{\mathbf{v}_2} v$$

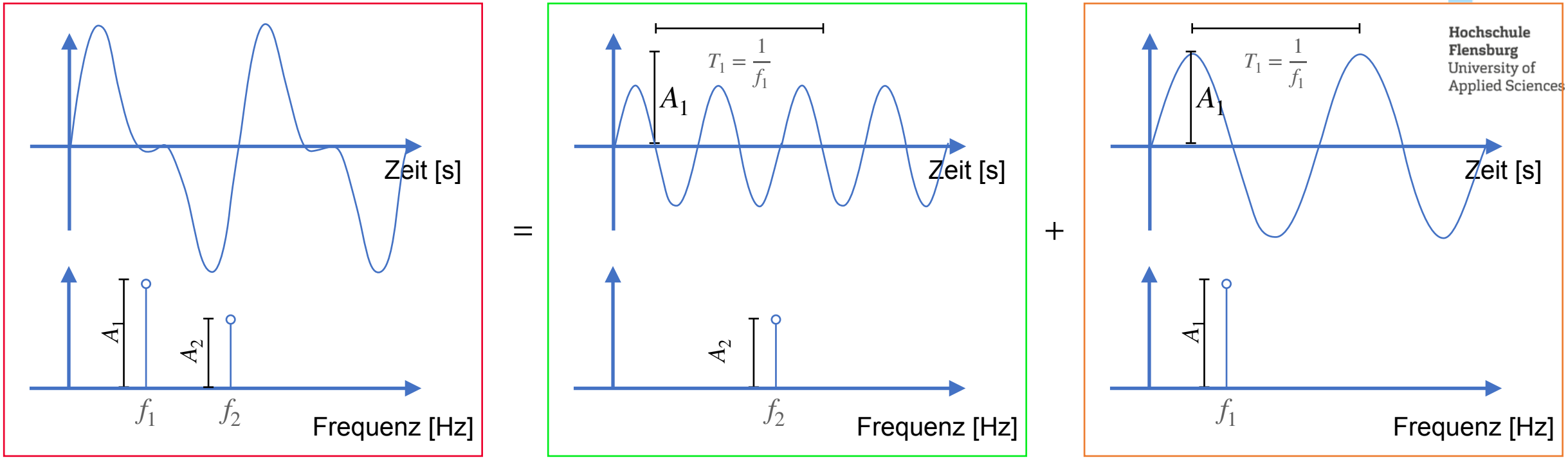
But why are we talking about this?

- All well and good—but why did we discuss this?
  - Let's briefly revisit our goal:
  - We want to break down any signal into its underlying sine waves with different frequencies, amplitudes, and phases.
  - In other words, we want to represent:

$$y(t) = \sum_{k=1}^{n} A_k \sin(2\pi f_k t + \phi_k)$$



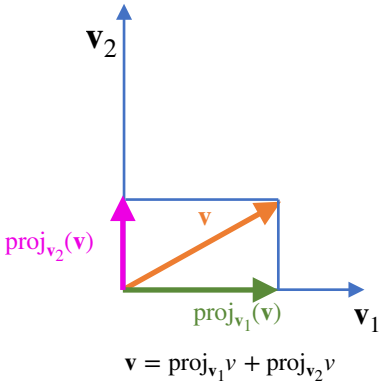
Decomposition into frequencies



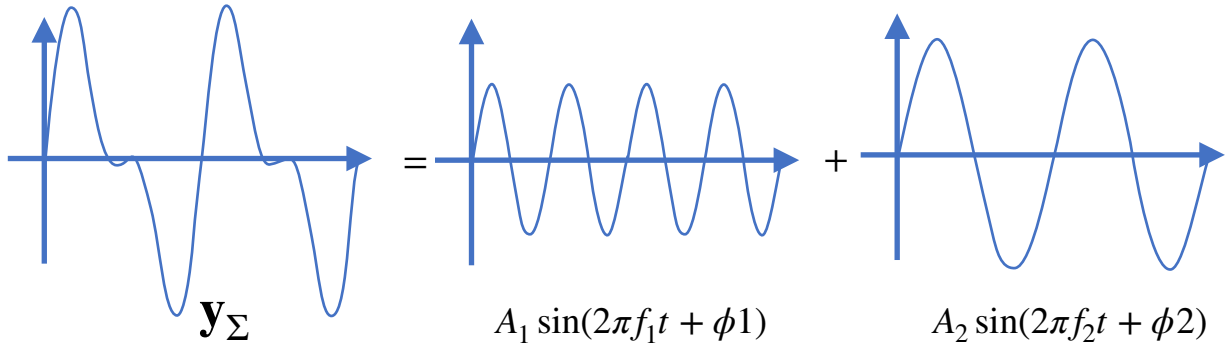
y_Σ $\quad$ $A_1 \sin(2\pi f_1 t + \phi_1)$ $\quad$ $A_2 \sin(2\pi f_2 t + \phi_2)$

- In other words: We want to break down the signal mixture (left) into its individual oscillations:
  $y_\Sigma = A_1 \sin(2\pi f_1 t + \phi_1) + A_2 \sin(2\pi f_2 t + \phi_2)$
- This gives us all the individual frequencies $f_1, f_2, \ldots$, the amplitudes $A_1, A_2, \ldots$, and phases ϕ_1, ϕ_2 of those frequencies.

But: is this the same?



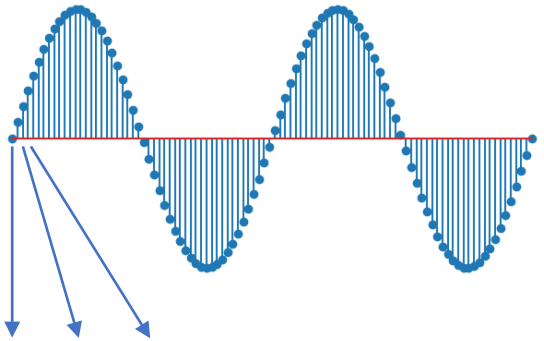
vs



- We now had:
  - on the one hand, the vectors in a basis V: $\mathbf{v} = [\ldots]$:
  - on the other hand, a (sum) signal $\mathbf{y}_\Sigma$
- We want to decompose it into:
  - on the one hand, projections into orthogonal basis vectors: $\text{proj}_{\mathbf{v}_i}\mathbf{v} = \langle \mathbf{v}, \mathbf{v}_i \rangle \mathbf{v}_i$
  - on the other hand, underlying sine waves: $A_i \sin(2\pi f_i + \phi_i)$
- Aren't these completely different things?

It's just vectors

- At the end, our signal y_Σ of oscillations is also just a vector:



[0.00,0.13,0.25,0.37,0.49,0.59,0.69,0.78,0.85,0.91,0.95,0.98,1.00,1.00,0.98,0.95, ...]

- And our basis functions $\mathbf{v}_i$ are also just vectors.
- We now need to check: Are the sine waves also an orthogonal basis?

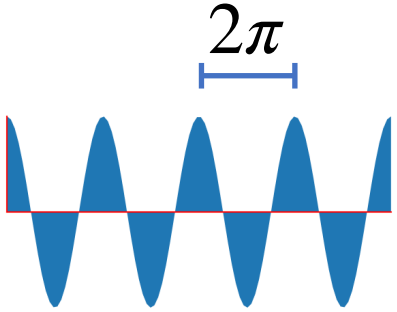
Sine waves as orthogonal basis

- So if:
  - The sine waves completely span the signal space, meaning that we can generate every signal as a linear combination of sine waves (we still need to check this, or at least assume it for now).
  - The sine waves form an orthonormal basis.
- Then we can decompose the signal into all its basis functions using projections with the basis functions. (The “length” of the vector product with the basis functions then gives us the amplitude).
- So we can then decompose each signal into its underlying sinusoidal oscillations.
- This tells us which frequencies occur in the signal and at what strength (amplitude).

Hochschule Flensburg
University of Applied Sciences

2π periodicity

- In the following, we assume that the signals we are considering are periodic. For sine functions, we find a periodicity with 2π, i.e., they repeat themselves at intervals of 2π.



- Formally expressed:
  $f(t + 2\pi) = f(t)$
- This works for all sine waves with integer circular frequencies $2\pi f = \omega$.
- Then it is sufficient for us to consider the interval $[-\pi, \pi]$.

Orthogonality of sine waves

- We calculate the scalar product (dot product) to check for orthogonality:

$$\langle \sin(\omega_1 t), \sin(\omega_2 t) \rangle = \int_{-\pi}^{\pi} \sin(\omega_1 t)\sin(\omega_2 t)dt$$

- We now state $\omega_1 \neq \omega_2$, then:

$$\int_{-\pi}^{\pi} \sin(\omega_1 t)\sin(\omega_2 t)dt = \int_{-\pi}^{\pi} cos((\omega_1 - \omega_2)t) - cos((\omega_1 + \omega_2)t)dt$$

$$= \int_{-\pi}^{\pi} cos((\omega_1 - \omega_2)t)dt - \int_{-\pi}^{\pi} cos((\omega_1 + \omega_2)t)dt$$

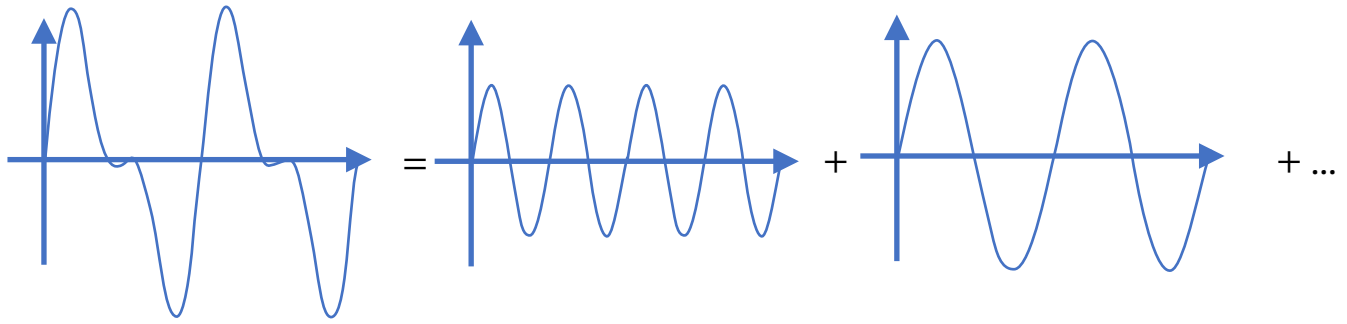
$$= 0$$

(there are always exactly as many positive as negative surface areas)

Orthogonality is given for all frequencies.

What does that mean?

- Sine functions of different frequencies are orthogonal.
- This means that their scalar product is always 0.
- The consequence of this is that
  - when we multiply a signal by a (normalized) sine function, the integral over this function will always give us the amplitude of this sine function in the signal.
  - This allows us to determine how “strong” this oscillation is in the signal.
  - If we know all the frequencies contained in the signal, we can use this to **decompose** the signal.



Let's do this now:

- If we have a signal mixture $y_\Sigma = A_1 \sin(2\pi f_1 t + \phi_1) + A_2 \sin(2\pi f_2 t + \phi_2)$

$$\langle y_\Sigma, \sin(2\pi f_1 t)\rangle = \int_{-\pi}^{\pi} y_\Sigma \sin(2\pi f_1 t) dt$$

$$= \int_{-\pi}^{\pi} \left(A_1 \sin(2\pi f_1 t + \phi_1) + A_2 \sin(2\pi f_2 t + \phi_2)\right) \sin(2\pi f_1 t) dt$$

$$= \int_{-\pi}^{\pi} A_1 \sin(2\pi f_1 t + \phi_1)\sin(2\pi f_1 t) dt + \int_{-\pi}^{\pi} A_2 \sin(2\pi f_2 t + \phi_2)\sin(2\pi f_1 t) dt$$

  - Case 1: $\phi_1 = 0$

$$= A_1 \int_{-\pi}^{\pi} \sin^2(2\pi f_1 t) dt + A_2 \int_{-\pi}^{\pi} \sin(2\pi f_2 t + \phi_2)\sin(2\pi f_1 t) dt$$

$$= 2\pi A_1$$

-> We the (scaled) amplitude, yay!

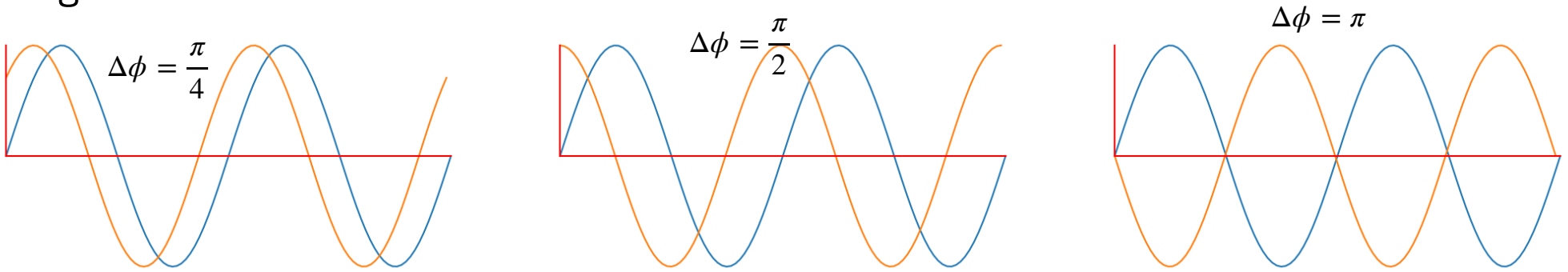
  - Case 2: $\phi_1 \neq 0$

$$= ?$$

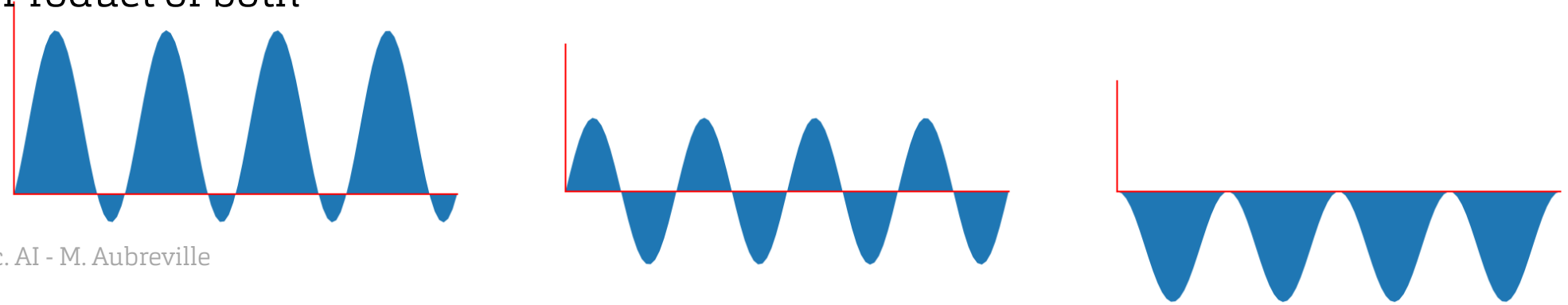
Case 2: Shown visually

- When we multiply two sine waves that have a phase difference of $\neq 0$:

Single oscillations



Product of both



Case 2: Conclusions

- Unfortunately, this means we end up with a dependency on the phase (or phase difference).
- But that bothers us. Ideally, we would like to obtain the result directly from the multiplication, which is the amplitude at the corresponding frequency.
- So unfortunately, we have to dig a little deeper.

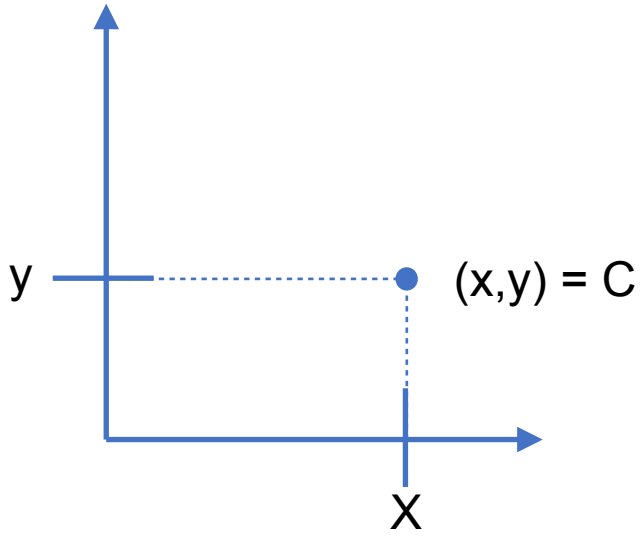
Hochschule
Flensburg
University of
Applied Sciences

Complex numbers

Yep, there wasn't enough math yet.

Complex numbers (1)

- A complex number is nothing but an extension of a scalar into two dimensions.



Notation:

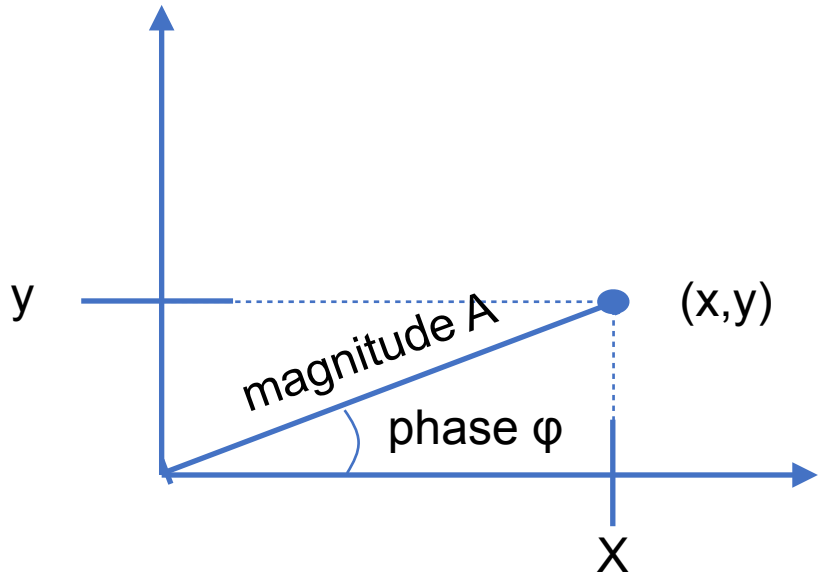
$$C = x + iy$$

imaginary number
(sometimes also j)

- The number now has two components: The **real part** x, and the **imaginary part** y.

Complex numbers (2)

- Let's briefly recall trigonometry: the real part and imaginary part can be understood as the tangents of a right-angled triangle.
- This allows us to calculate the hypotenuse (which we refer to here as the magnitude)



$$A = \sqrt{x^2 + y^2}$$

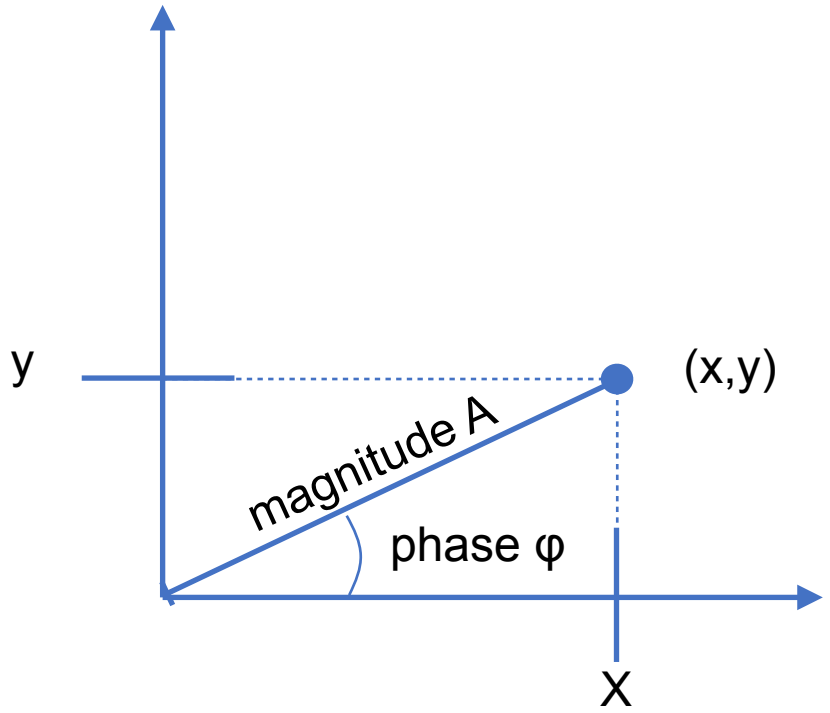
$$\phi = \arctan \frac{y}{x}$$

(if x>0)

- We can also determine the angle between the hypotenuse and the real part.
- We refer to this as the phase. We will see shortly why we use the same term for this as for the sine wave.

Complex numbers (3)

- From the absolute value A and the phase ϕ, we can reconstruct the real part x and the imaginary part y:



$$x = A\cos\phi$$

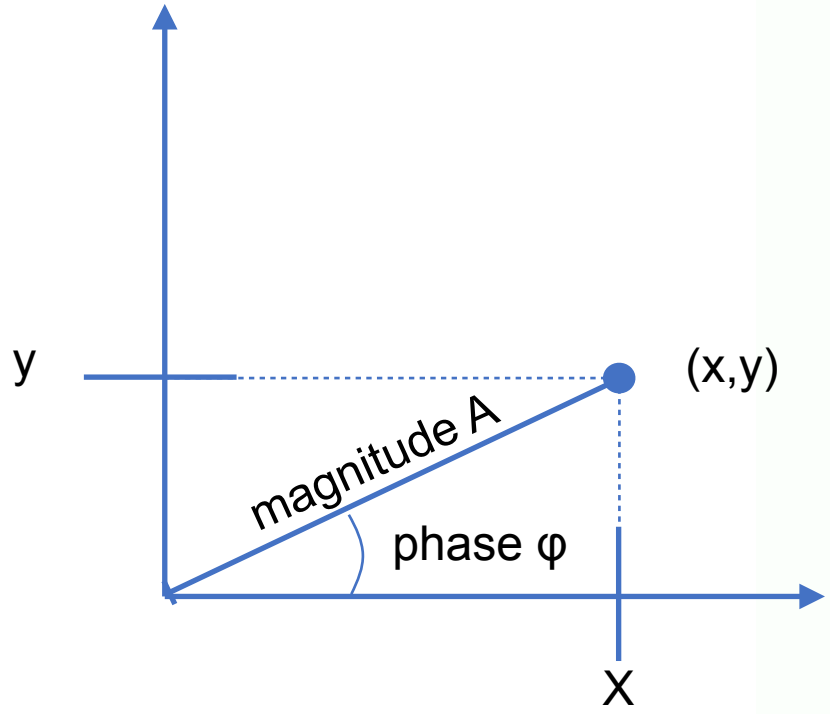
$$y = A\sin\phi$$

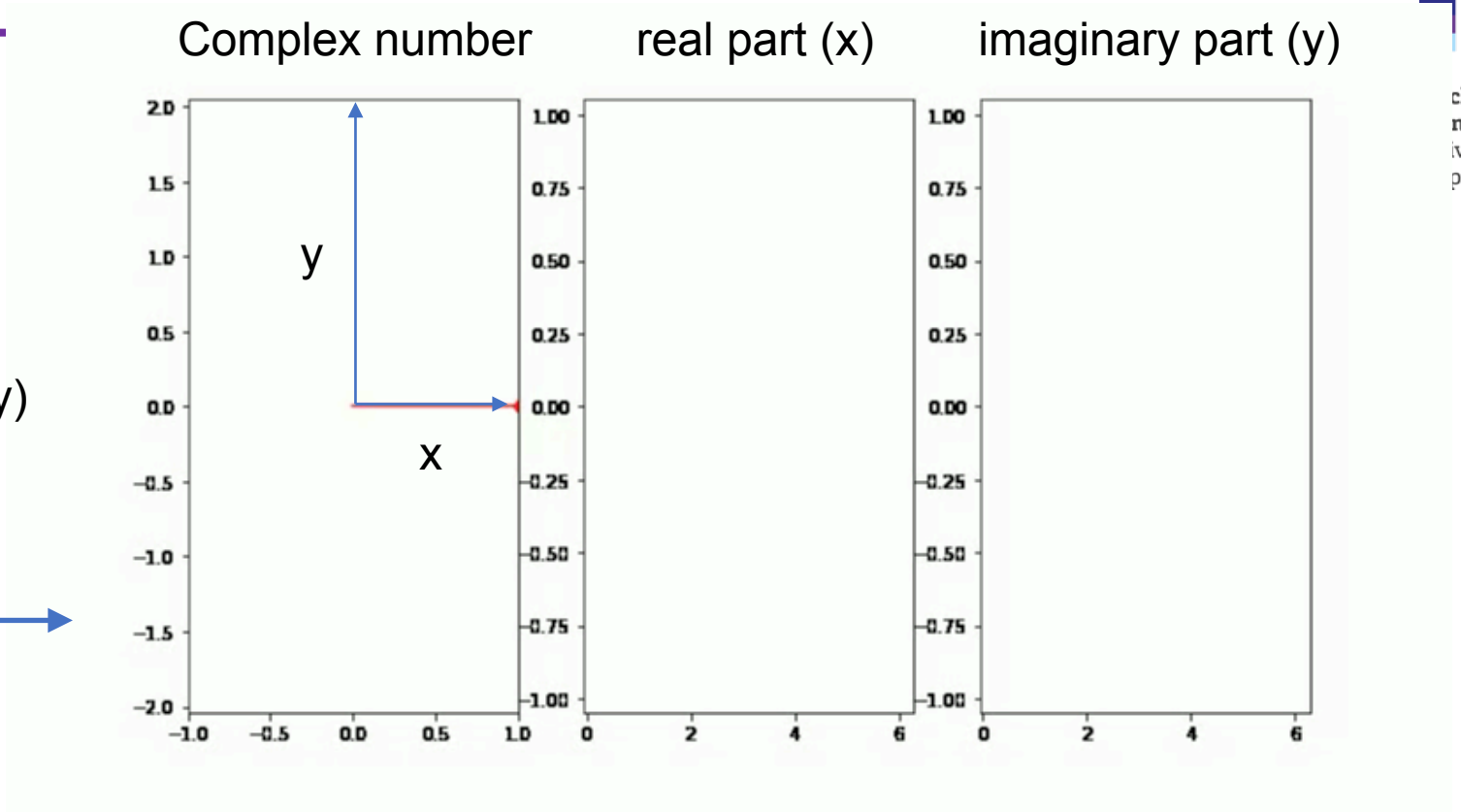
$$C = x + iy = A\left(\cos\phi + i\sin\phi\right)$$

$$= Ae^{i\phi} \quad \text{(Euler's formula)}$$

- Furthermore, we can convert this using Euler's formula to $C = Ae^{i\phi}$

Complex numbers (4)





- If we look at the real part and imaginary part with increasing phase ϕ, we see the sine/cosine oscillation again.
- We find it in $x = A\cos(\phi t)$ and $y = A\sin(\phi t)$

A complex sine wave

- We can express a complex-valued oscillation by “rotating” the phase:

$$f(t) = Ae^{i\phi_t} = Ae^{i\omega t} = Ae^{i2\pi ft}$$

Complex oscillation

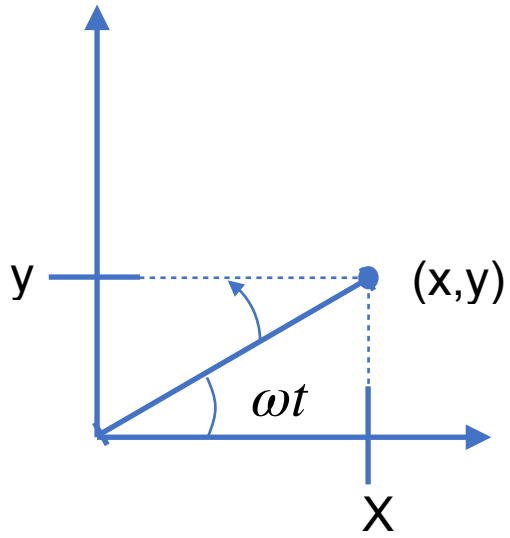
amplitude/
absolute value

(time-dependant) phase

circular frequency [1/s]

time

frequency [Hz]

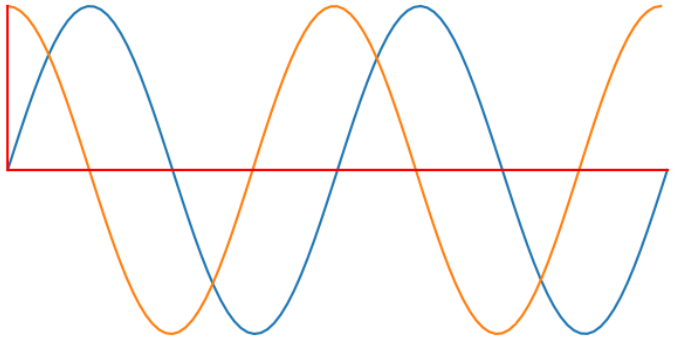


- The higher the frequency, the faster the rotating pointer moves in a circle.
- The higher the frequency, the faster the signal oscillates.

From real-valued to complex-valued oscillations (sine waves)

Hochschule Flensburg
University of Applied Sciences

- A complex-valued oscillation has the property that the real and imaginary parts are always offset by 90°.



- As we will see shortly, we can use this to make the product between our mixed signal and the oscillation (in magnitude) independent of the phase.
- The goal is still to break down the signal mixture into different frequencies.

Hochschule
Flensburg
University of
Applied Sciences

Fourier-Transformation

Let's

- With complex oscillation, we now have a tool that explicitly takes phase into account and allows us to break signals down into their frequency components.
- By using complex numbers, we can combine both the amplitude and phase of the oscillation in a single mathematical representation.
- Let's look at an example:

$$\langle y_\Sigma(t), e^{-i(2\pi f_1 t)} \rangle = \int_{-\infty}^{\infty} y_\Sigma(t) e^{-i(2\pi f_1 t)} dt$$

Hochschule Flensburg
University of Applied Sciences

Back to our "case 2"

- Let's take another look at what happens to the scalar product when one of the frequencies of the oscillation is included.
- For simplicity's sake, we will only look at the term where the frequency also matches; the other term will be 0 anyway (orthogonality).

$$\int_{-\infty}^{\infty} \left(A_1 sin(2\pi f_1 t + \phi)\right) e^{-i(2\pi f_1 t)} dt$$

- Let's calculate this now (next slides)

Hochschule Flensburg
University of Applied Sciences

Let's calculate this

- First, we can split the sine using Euler's formula into:

$$sin(2\pi f_1 t + \phi) = \frac{e^{i(2\pi f_1 t+\phi)} - e^{-i(2\pi f_1 t+\phi)}}{2i}$$

- This changes the previous formula to:

$$\langle A_1 sin(2\pi f_1 t + \phi), e^{-i2\pi f_1 t} \rangle = \int_{-\pi}^{\pi} A_1 \frac{e^{i(2\pi f_1 t+\phi)} - e^{-i(2\pi f_1 t+\phi)}}{2i} e^{-i2\pi f_1 t} dt$$

$$= \frac{A_1}{2i} \int_{-\pi}^{\pi} e^{i(2\pi f_1 t+\phi-2\pi f_1 t)} dt - \frac{A_1}{2i} \int_{-\pi}^{\pi} e^{-i(2\pi f_1 t+\phi)-i2\pi f_1 t} dt$$

$$= \frac{A_1}{2i} \int_{-\pi}^{\pi} e^{-i\phi} dt - \frac{A_1}{2i} \int_{-\pi}^{\pi} e^{i(4\pi f_1 t+\phi)} dt \qquad = \frac{\pi}{i} A_1 e^{i\phi}$$

Complex oscillation (with double frequency), integral is again 0

no longer dependent of time (t)

The result

- The result is thus:

$$= \frac{\pi}{i} A_1 e^{i\phi}$$

Phase of the (sine) oscillation

Amplitude of the oscillation

const. factor

- So, as a result, we have obtained a complex number that tells us exactly what phase and amplitude our oscillation (which was part of a mixture with other oscillations) had.

Frequency analysis

- Let's summarize these findings:
  - If we multiply our signal mixture with complex oscillations and integrate over it (scalar product), we obtain:
  - For frequencies that do not match, the result is 0.
  - For frequencies that match, we obtain the magnitude and phase of the oscillation, which allows us to clearly identify the oscillation in the mixture.
  - If we can do this for all frequencies present in the mixed signal, we can break down the signal into all the frequencies it contains.

Hochschule Flensburg
University of Applied Sciences

Fourier transform

- We refer to the generalization for all frequencies as the Fourier transform.
- It is defined as:

$$\mathcal{F}\left\{\hat{f}(t)\right\} = \int_{t=-\infty}^{\infty} \hat{f}(t) \quad e^{-i2\pi ft} \quad dt$$

A complex oscillation

Any mixture signal

Integral over time (generally over all time to ensure periodicity)

Hochschule Flensburg
University of Applied Sciences

Discrete Fourier Transform (DFT)

- Real signals are discretely sampled.
- We also cannot measure indefinitely. That is why we use:

$$X_k = \sum_{n=0}^{N} x_n \, e^{-i2\pi \frac{k}{N} n}$$

frequency

sum

any digital signal mixture

A complex oscillation with discrete frequncies

Hochschule Flensburg
University of Applied Sciences

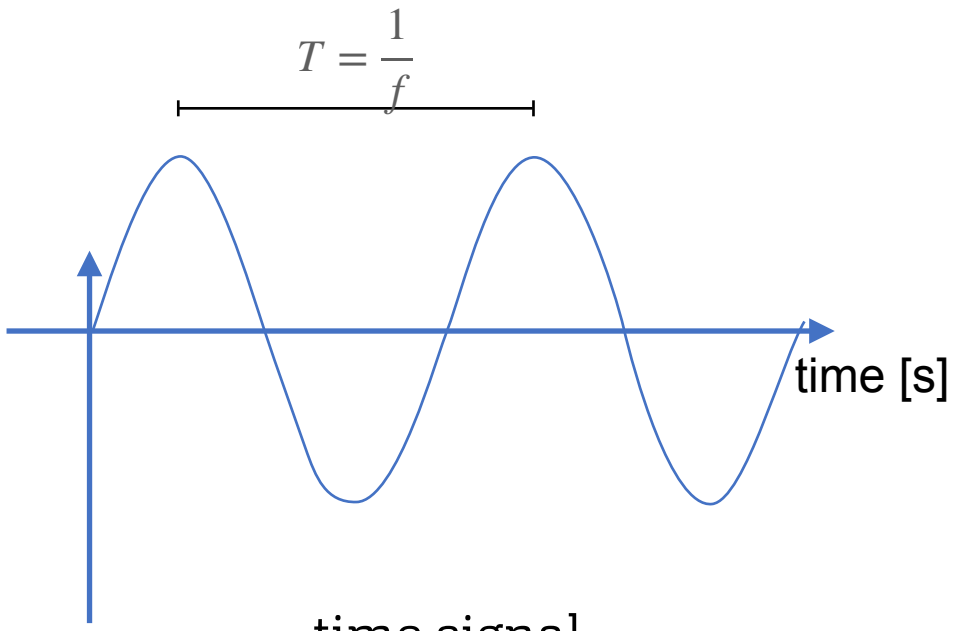
Discrete Fourier-Transformation (DFT)

- However, considering a time-discrete signal has a number of consequences:
  We cannot determine the frequency with any degree of accuracy. The frequency resolution depends on the length of the analyzed signal, i.e., on the number of sampling points. The longer the signal, the more accurately the frequency can be determined.
- Observing only a limited time window also has effects:
  If the signal is not exactly periodic or begins and ends abruptly, this can lead to distortions in the frequency spectrum (windowing effects).
- Nevertheless, the discrete Fourier transform is an invaluable tool, but we must be aware that we do not necessarily have perfect frequency resolution and that windowing effects can also occur.

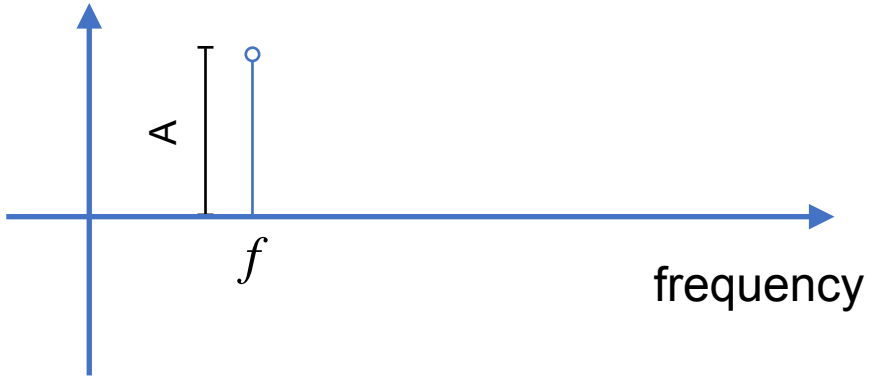
Hochschule
Flensburg
University of
Applied Sciences

Spectrum

- The spectrum of a signal is the absolute value of the Fourier Transform.
- It shows us the amplitudes of the sinusoids



time signal



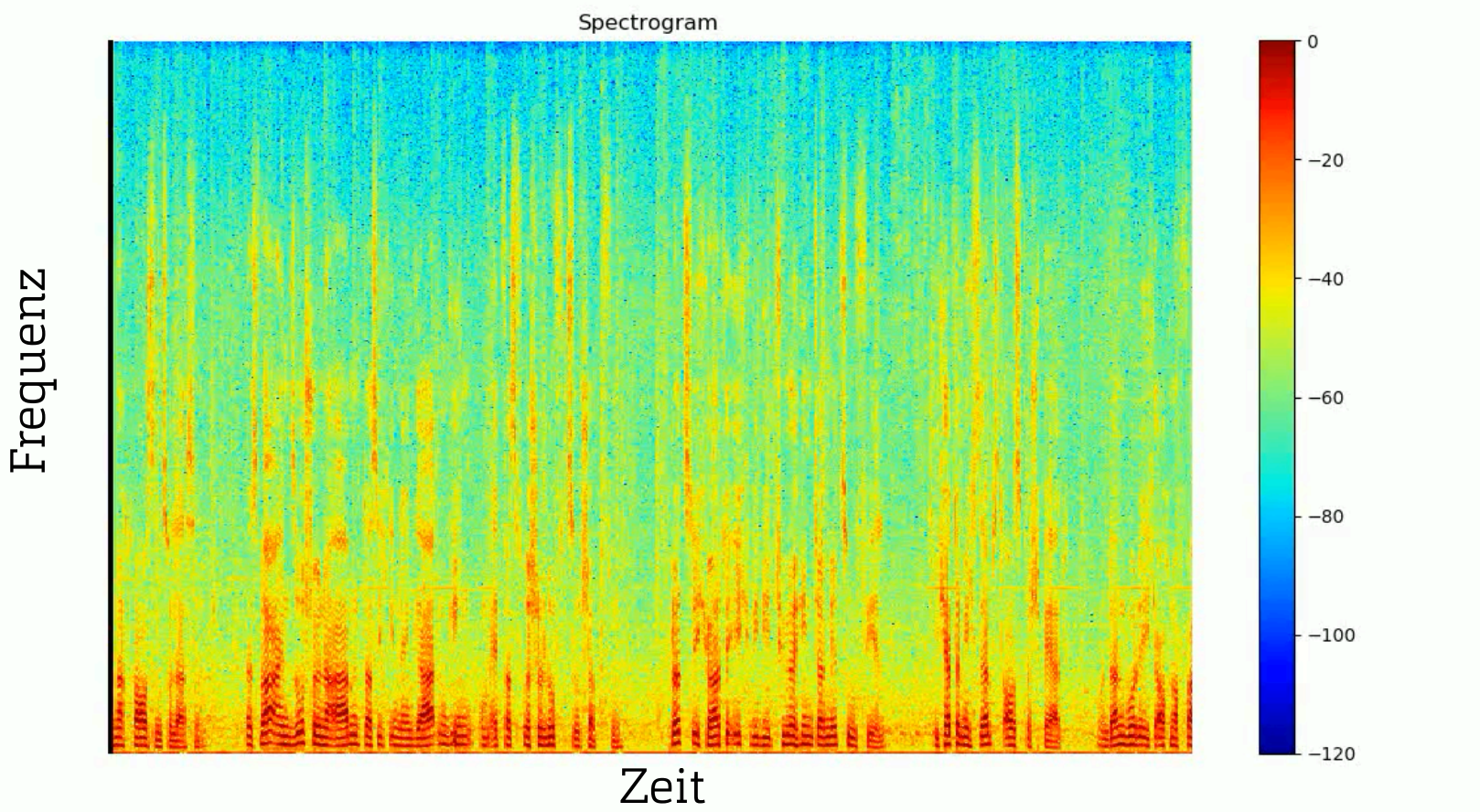
spectrum

Analysis of signal parts

- This gives us the frequency representation of the entire signal.
- It does not indicate when a spectral component was active.
- Wanted: A tool that provides us with a time-frequency representation.

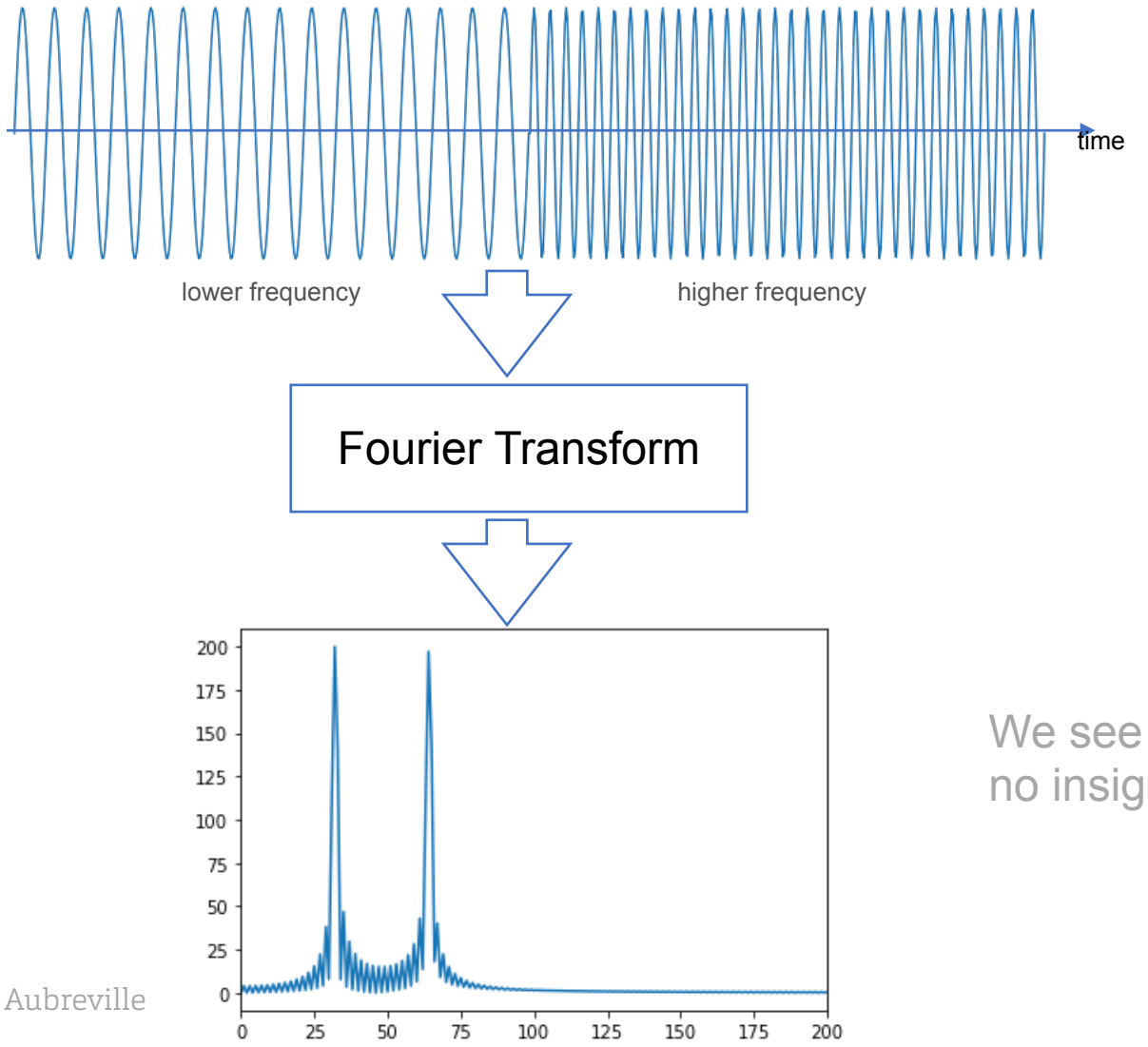
Spectrogram

- The spectrogram is the time-resolved version of the spectrum.
- It typically displays the amplitude over frequency and time.



Hochschule Flensburg
University of Applied Sciences

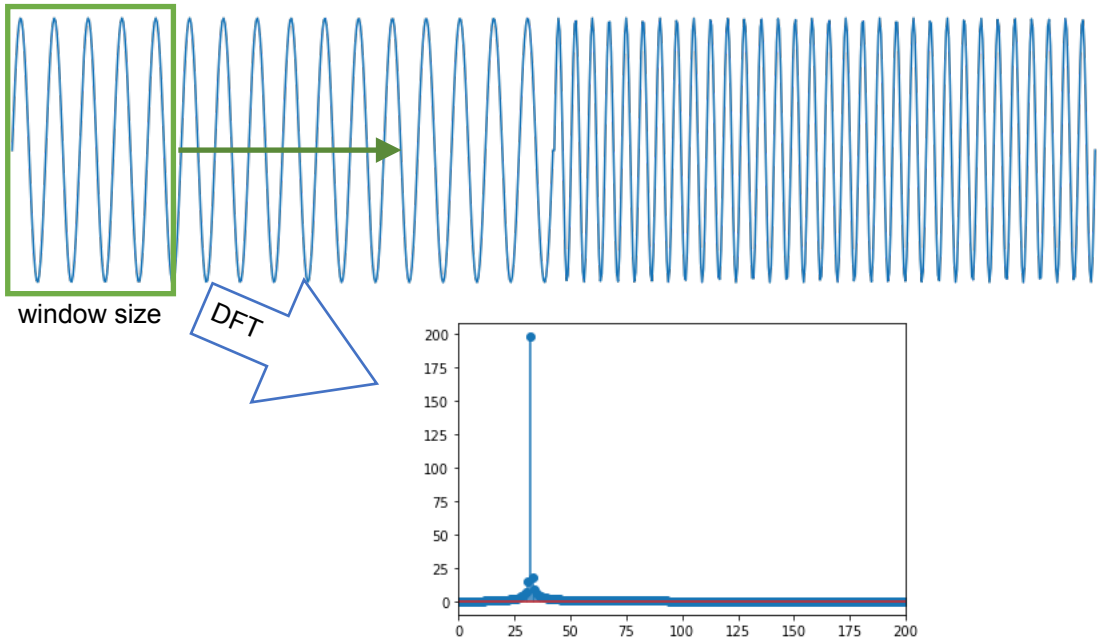
Calculating the spectrogram



We see both frequency components, but gain no insight into the temporal distribution.

Short-time Fourier transform (STFT)

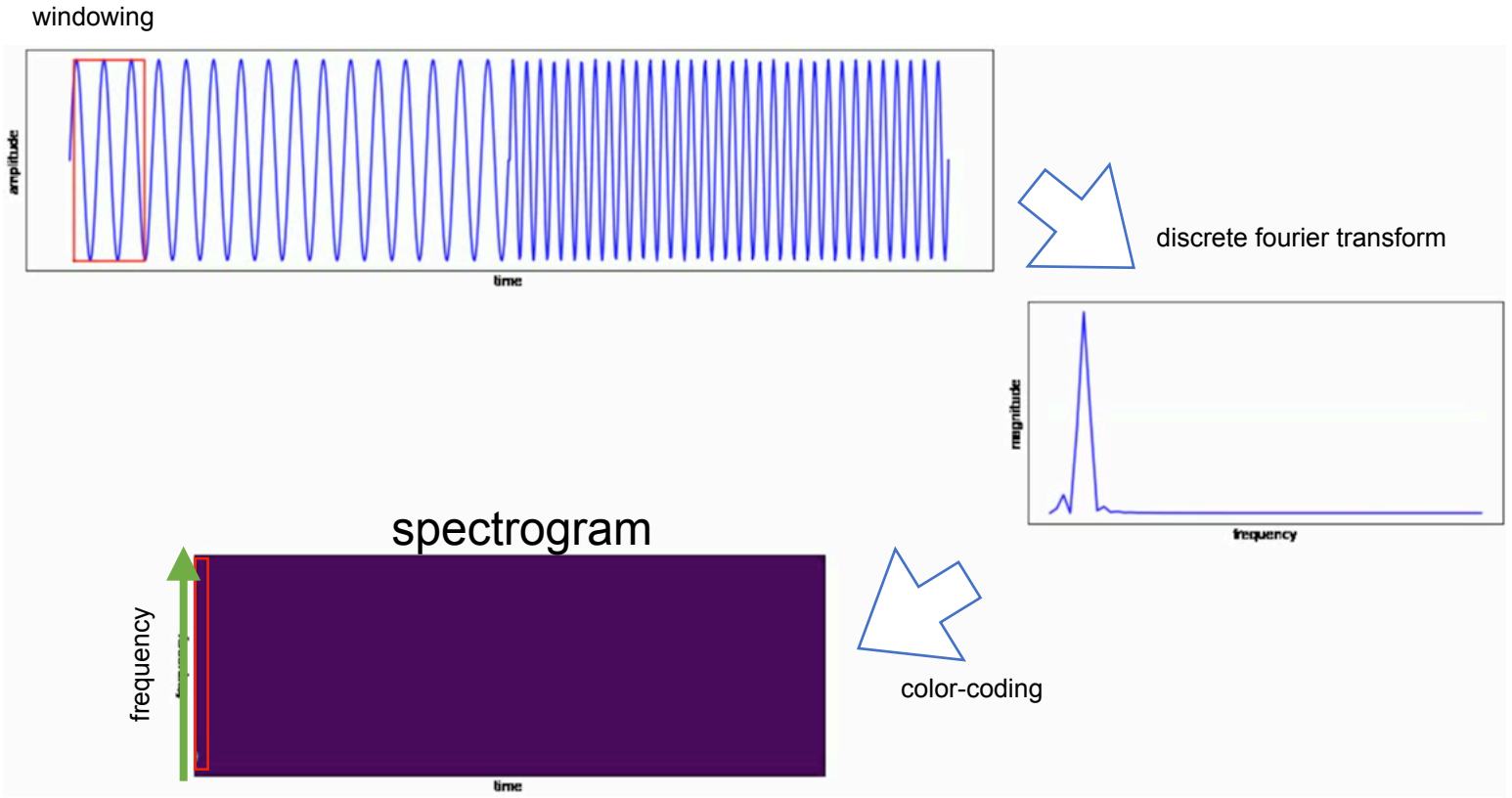
- Application of the discrete Fourier transform (DFT) to windows of the signal (windowing).



- The window size is decisive for the frequency resolution, but also for the time resolution.
- The amplitudes are displayed as a color-coded spectrogram.

Example for STFT

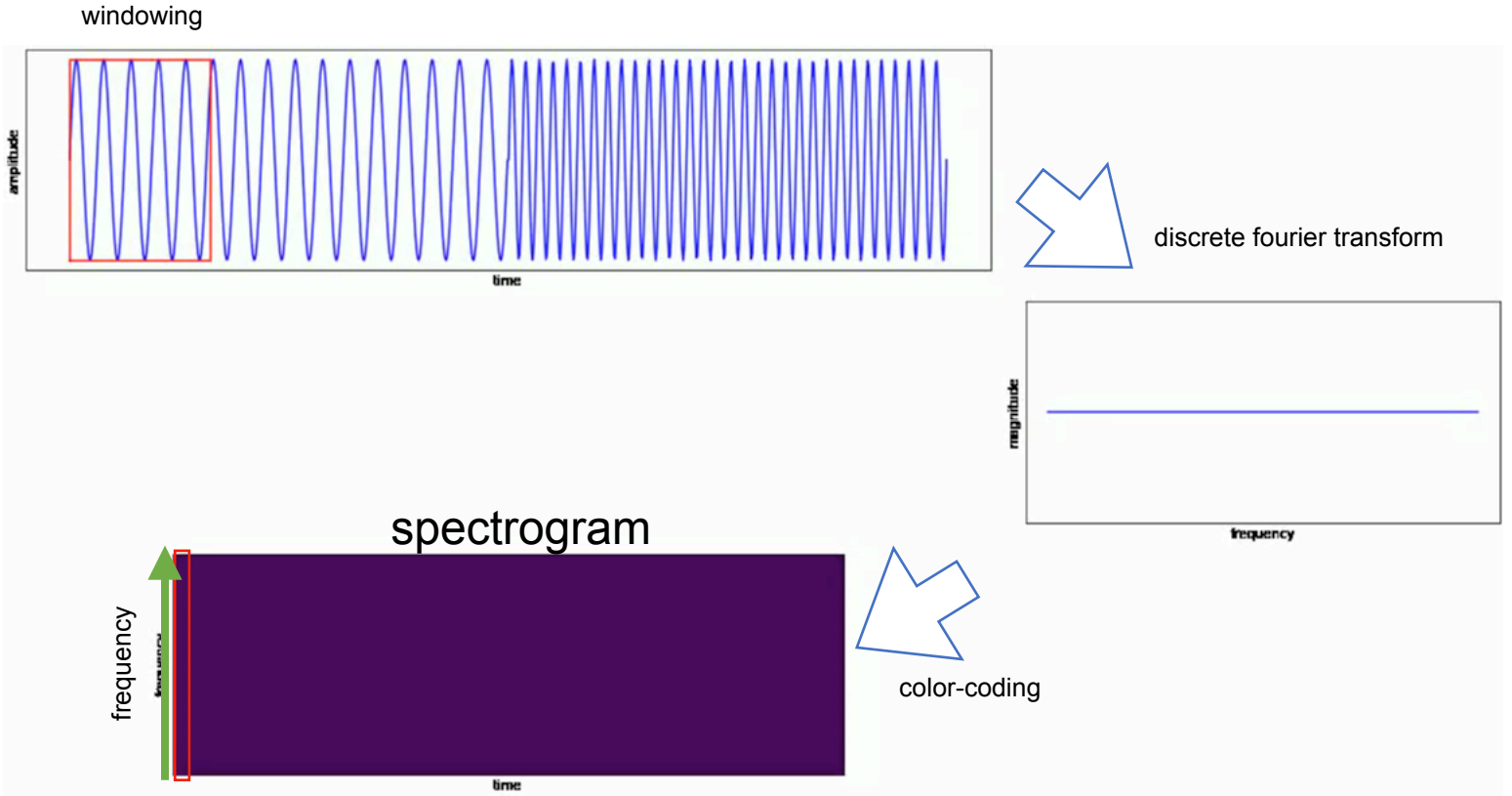
- The STFT performs a DFT on a moving window to obtain a time-frequency display.



https://github.com/maubreville/VL_Signalverarbeitung_Mustererkennung/blob/main/STFT.mp4

Experimenting with window size

- With a larger window



much "sharper"

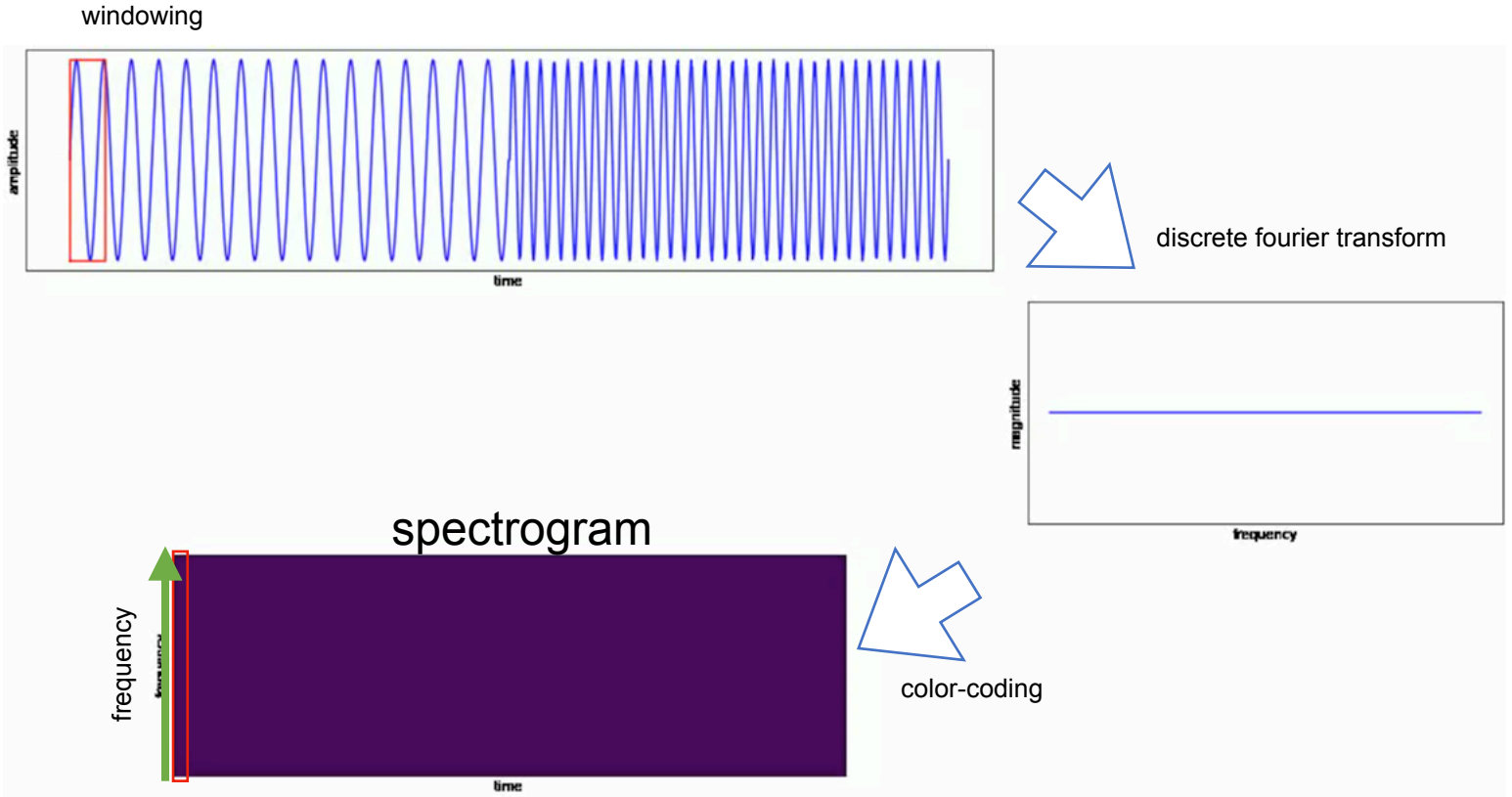
higher frequency resolution

but unfortunately with a reduced time resolution

https://github.com/maubreville/VL_Signalverarbeitung_Mustererkennung/blob/main/STFT_128.mp4

Experimenting with window size

- Using a smaller window



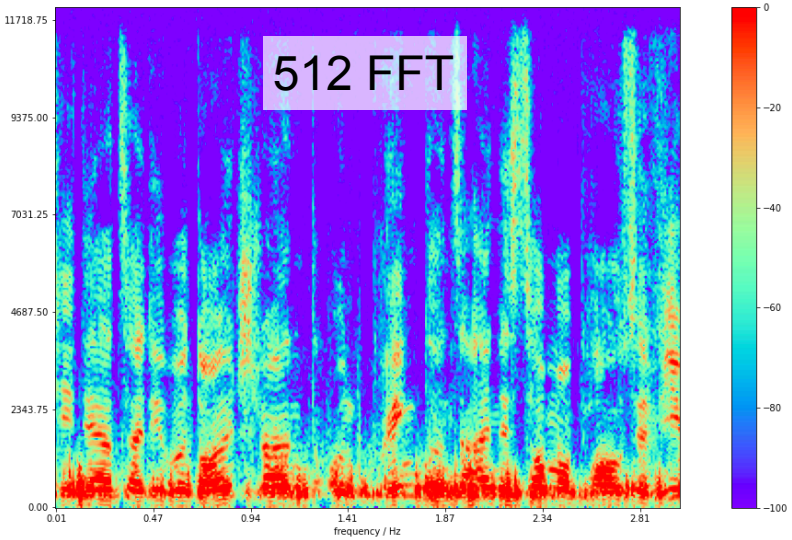
Reduced frequency resolution

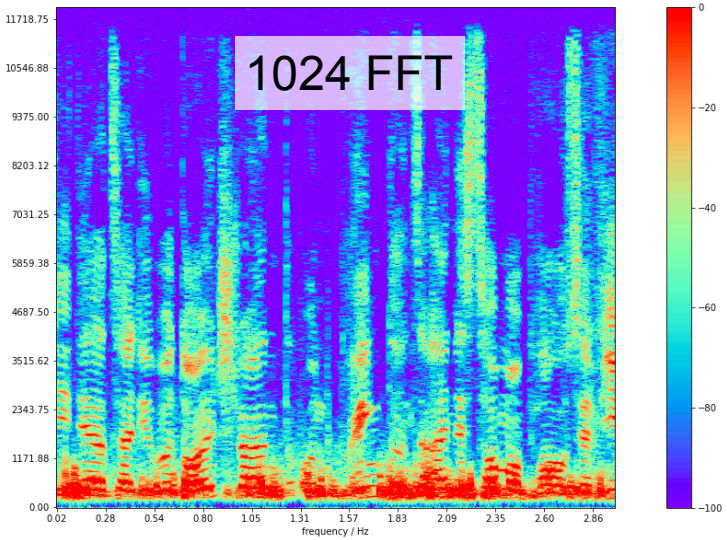
But much better time resolution

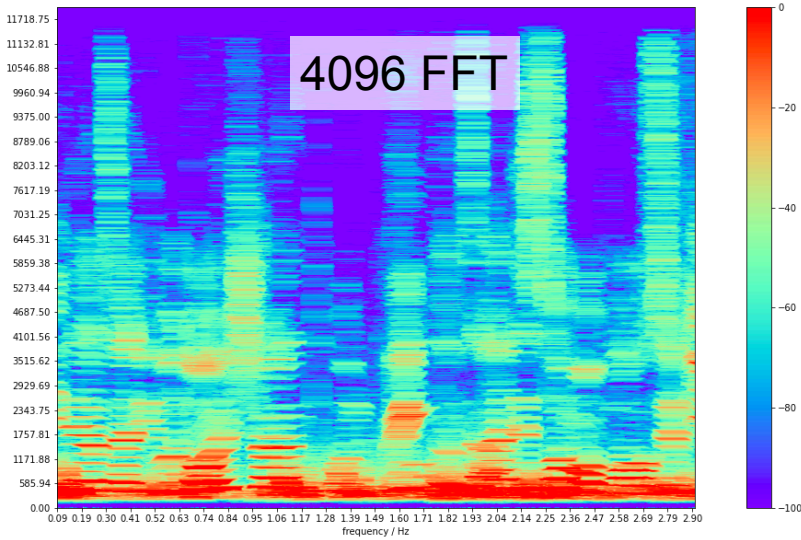
https://github.com/maubreville/VL_Signalverarbeitung_Mustererkennung/blob/main/STFT_32.mp4

Time-frequency trade-off in STFT

- Similar to Heisenberg's uncertainty principle, we cannot analyze exact frequency resolution or exact time resolution.



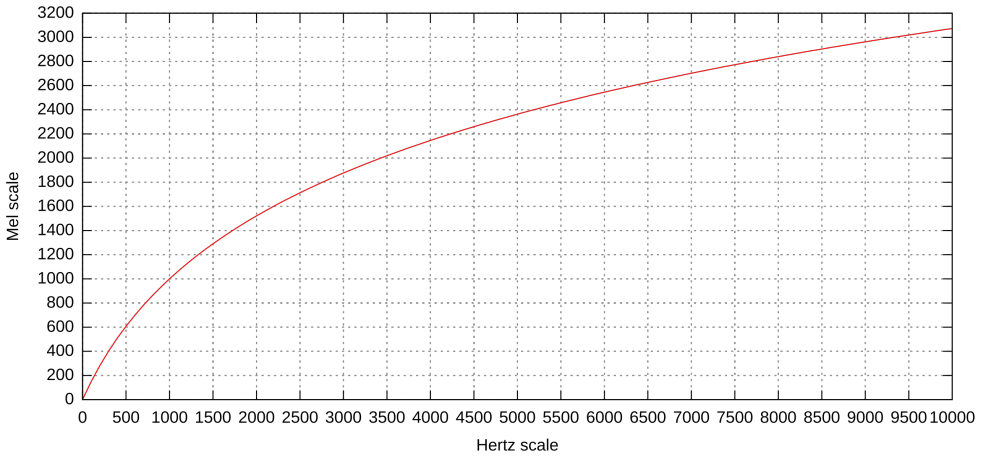




Hochschule
Flensburg
University of
Applied Sciences

Mel-spectrogram, cepstrum and MFCCs

Mel Scale



- Human perception of pitch is different from physical measurements of frequency.
- Pitch sensitivity is high at low frequencies, lower at high frequencies.
- The ear behaves approximately linear below ~1 kHz and logarithmic above it.
- The Mel scale models this by spacing frequencies according to perceived pitch, not Hz.
- Equal steps in Mel correspond to equal steps in perceived pitch.

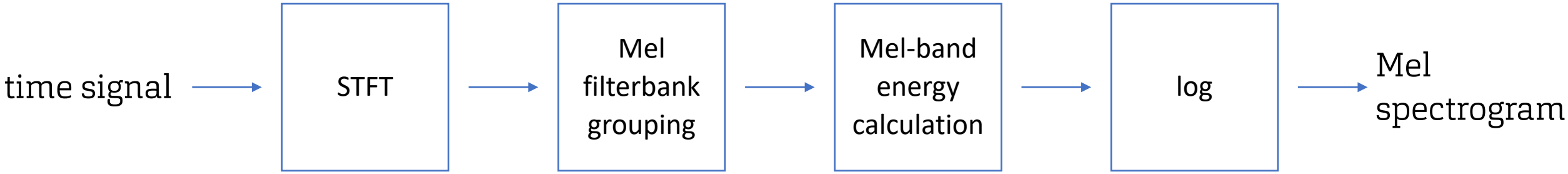
Hochschule Flensburg
University of Applied Sciences

https://en.wikipedia.org/wiki/Mel_scale

Mel Spectrograms

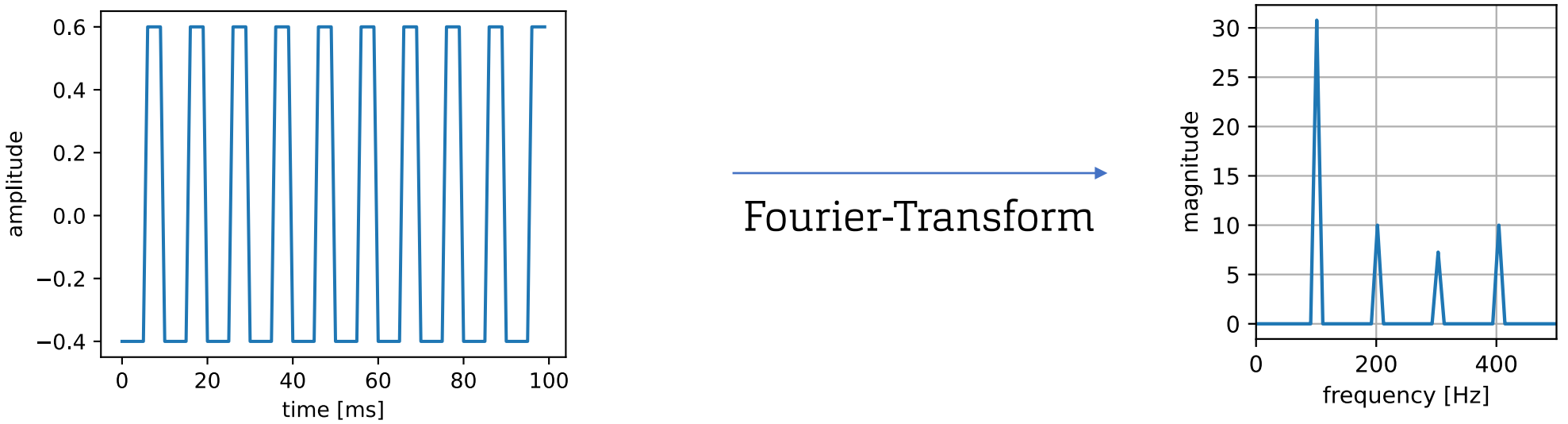
Hochschule Flensburg
University of Applied Sciences

- Human auditory perception has fine resolution at low frequencies, coarser at high frequencies.
- It makes sense to reflect this also in the analysis of signals.
- For this, we can group signal channels after the application of the STFT:
  - 1. Compute the Short-Time Fourier Transform (STFT)
  - 2. Apply Mel filterbanks, combining groups of neighboring STFT frequency bins according to the Mel (logarithmic) scale
  - 3. Compute Mel-band energies and calc the logarithm out of these (dB visualization).



Repeating patterns

- In each spectrum, we have - again - repetitive patterns.
- This is caused by signal harmonics, i.e., multiples of a fundamental frequency:

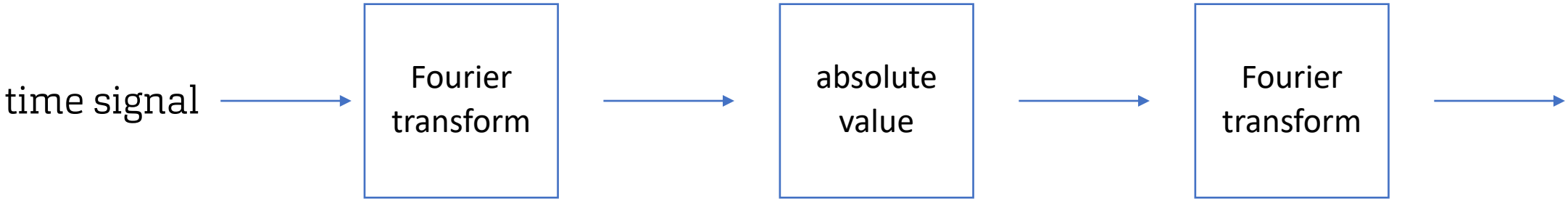


- This is very common for voice, music instruments, etc.
- But this makes the analysis of the signal more complicated.

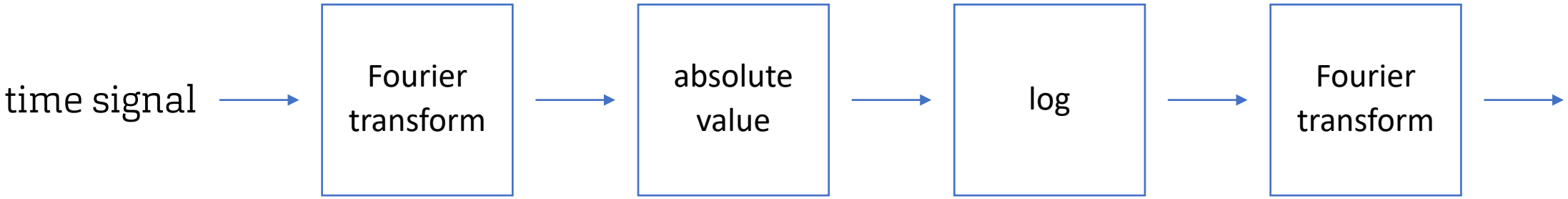
Cepstrum (Bogert, Healy und Tukey, 1963)

Hochschule Flensburg
University of Applied Sciences

- Since we already know that the FT is a tool to analyze repetitive patterns, we could just apply it again on the Fourier-transformed signal:



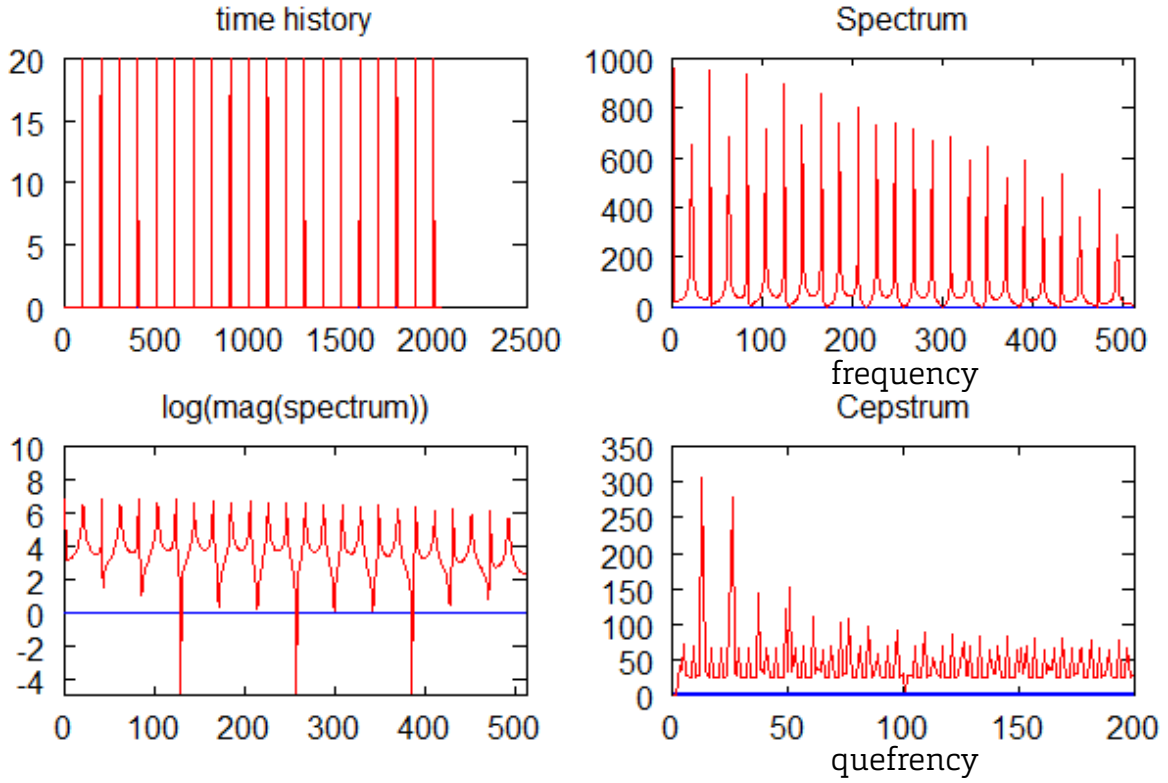
- We might have a multiplicative signal mixture of several sources, in this case, it's sensible to take the log before the analysis to transform multiplication into addition:



- The outcome of this analysis is called the Cepstrum.

Cepstrum

- The cepstrum can be seen as a frequency analysis of a frequency analysis.
- "Ceps"-trum is a word play, changing the order of letters in "spec"-trum.
- Funnily enough, the resulting frequency axis also has a different name: quefrency.

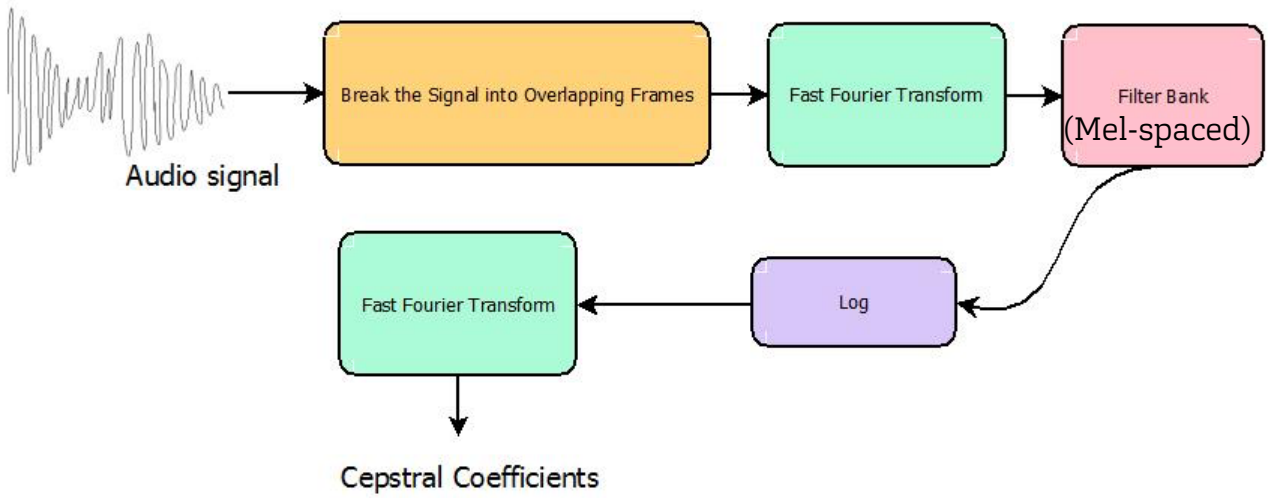


https://en.wikipedia.org/wiki/Cepstrum#/media/File:Cepstrum_signal_analysis.png

Mel-Frequency Cepstral Coefficients (MFCCs)

Hochschule Flensburg
University of Applied Sciences

- MFCCs can be seen as the analogous to the short-time Fourier transform (STFT).
- They are very often used for analysis in pattern recognition tasks.
- They're built from the cepstrum, but shaped to match how humans hear.
- Process:
  - Take the Fourier Transform of short frames of audio
  - Apply Mel-spaced filters to highlight perceptually important frequencies
  - Take the log of the filter outputs
  - Apply a Discrete Cosine Transform (DCT) or DFT to get compact coefficients.



https://mitll.github.io/pyslgr/MFCC_Features.html

Hochschule
Flensburg
University of
Applied Sciences

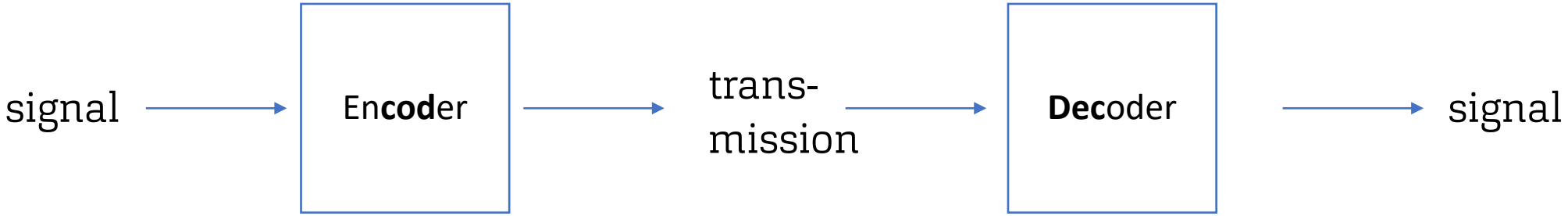
Generation of audio

Generation of audio signals

- We can generate audio signals in a multitude of different representations:
  - time signal (raw waveform)
  - time-frequency representation: spectogram or Mel spectrogram
  - in the representation space of a neural codec.

Audio Codecs

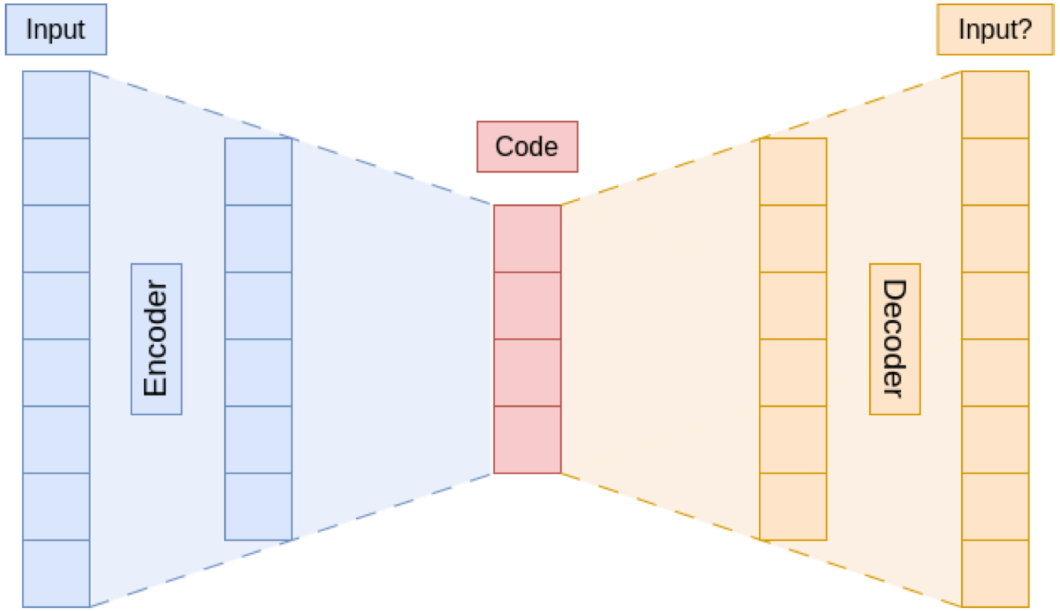
- Compress audio into discrete codes and reconstruct an approximation.



- Goal: minimize bitrate while keeping perceptual quality acceptable.
- Achieved by removing redundant or irrelevant signal information.
- Applications: communication, streaming (low latency), production (high quality).

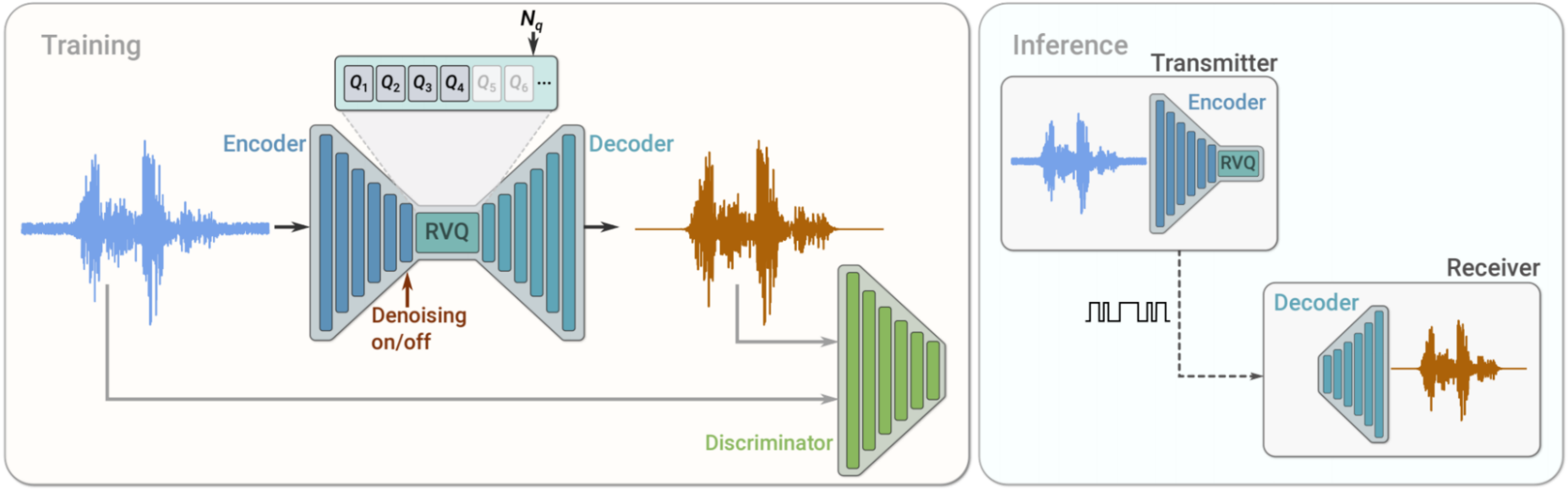
Neural Codecs

- The codec pipeline can also be interpreted as autoencoder.



- The "code" is then a compact, latent space representation of the signal.
- It is also a suitable domain for signal generation.
- We refer to audio codecs that are build on neural networks as "Neural Codecs".

SoundStream (Zeghidour et al., 2021)

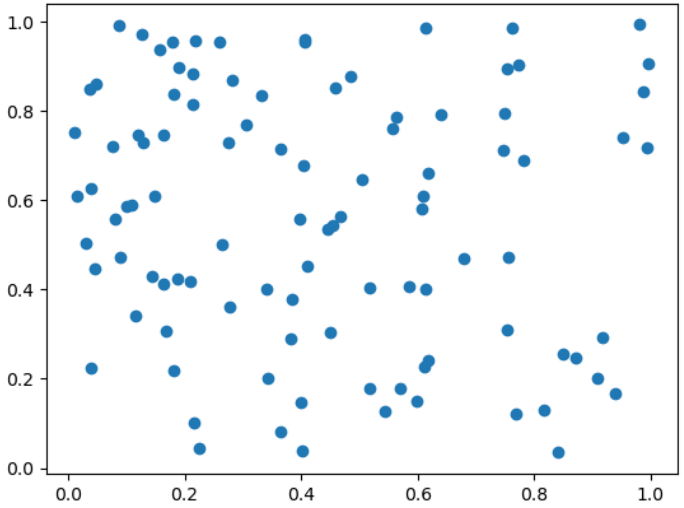


- Neural codec based on autoencoder setup
- Uses residual vector quantization (see next slides)

Residual Vector Quantization

Hochschule Flensburg
University of Applied Sciences

- The main goal of SoundStream is to have a bit-efficient representation of speech, so this can be transmitted over a data-restricted connection.
- The vector quantization is performed by a network that performs recursive quantization.
- Let's start with a 2D example (supposedly after the encoder of the autoencoder):



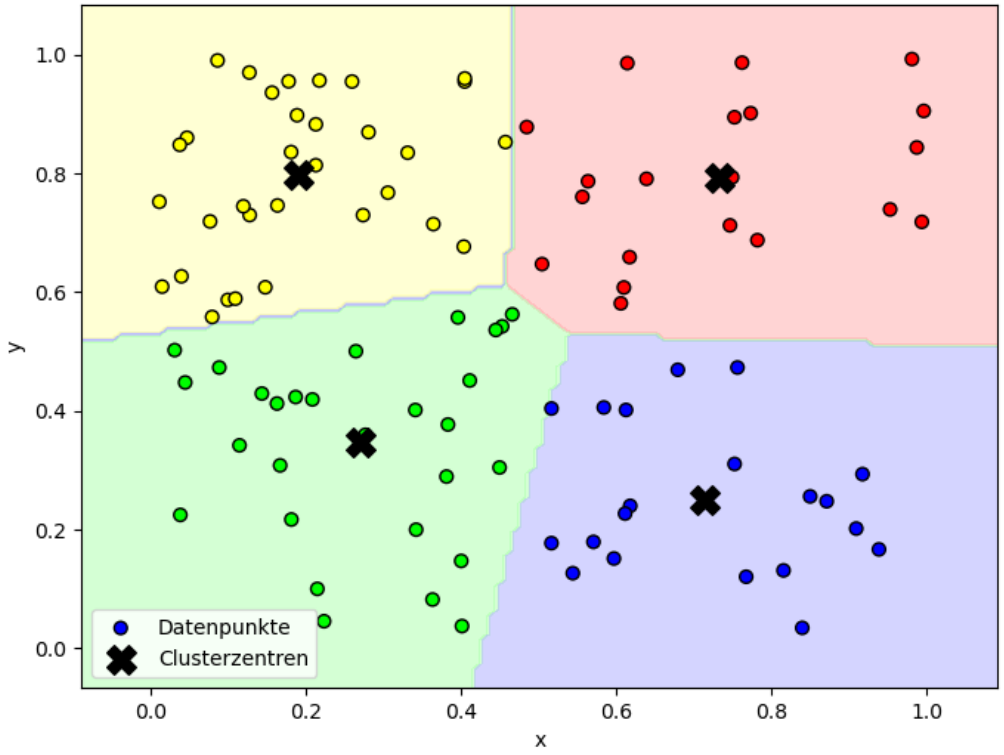
It is the idea of quantization to split up this 2D space into areas that represent symbols that can be transmitted.

Residual Vector Quantization (Zeghidour et al., 2021)

Hochschule Flensburg
University of Applied Sciences

- We can split this are up into 4 symbols:

[1,0]



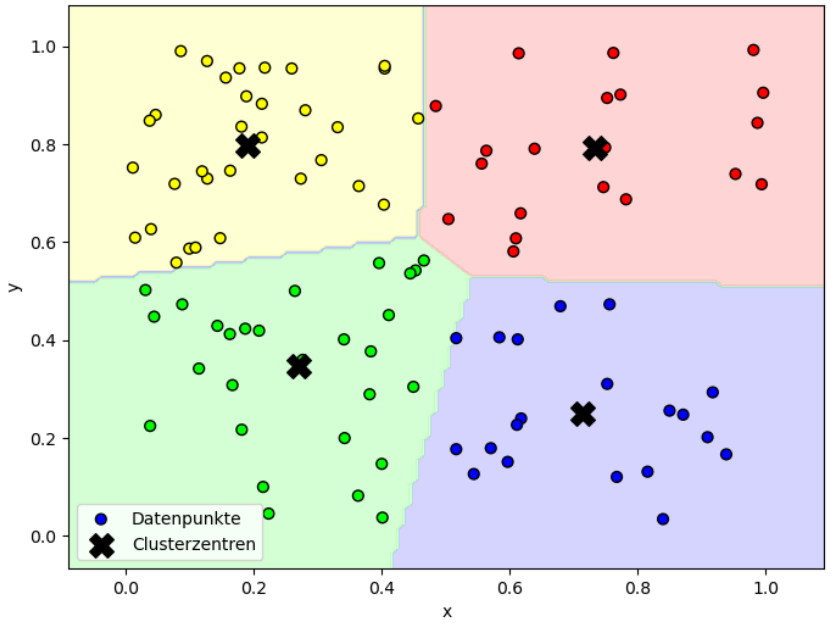
[0,0]

[1,1]

[0,1]

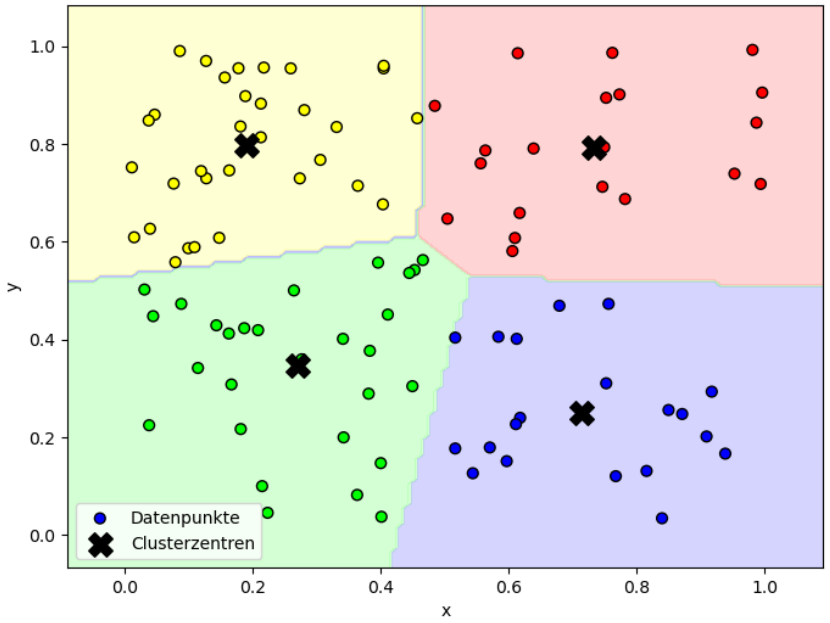
Codebooks

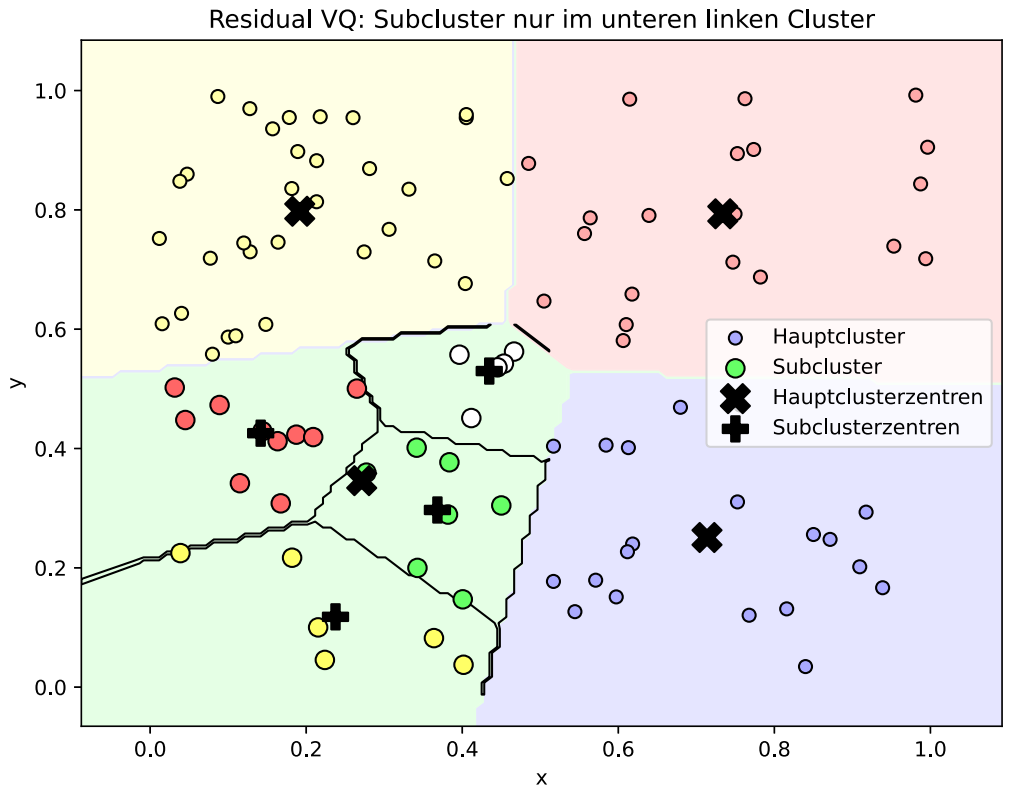
- There are many algorithms that we can use to divide the n-dimensional vector space into symbols (e.g., kmeans, fixed grid).
- Since the distribution is typically not uniform, it does not make sense to go with a fixed grid (would be inefficient).
- The idea of codebook vectors (which we know from VQGAN and VQVAE) is that we find N codebook entries that represent the data well (i.e., have a small distance).



Recursive codebooks

- We can now extend this to split up each field into four more fields (symbols).
- With each split, the residual error of quantization gets reduced

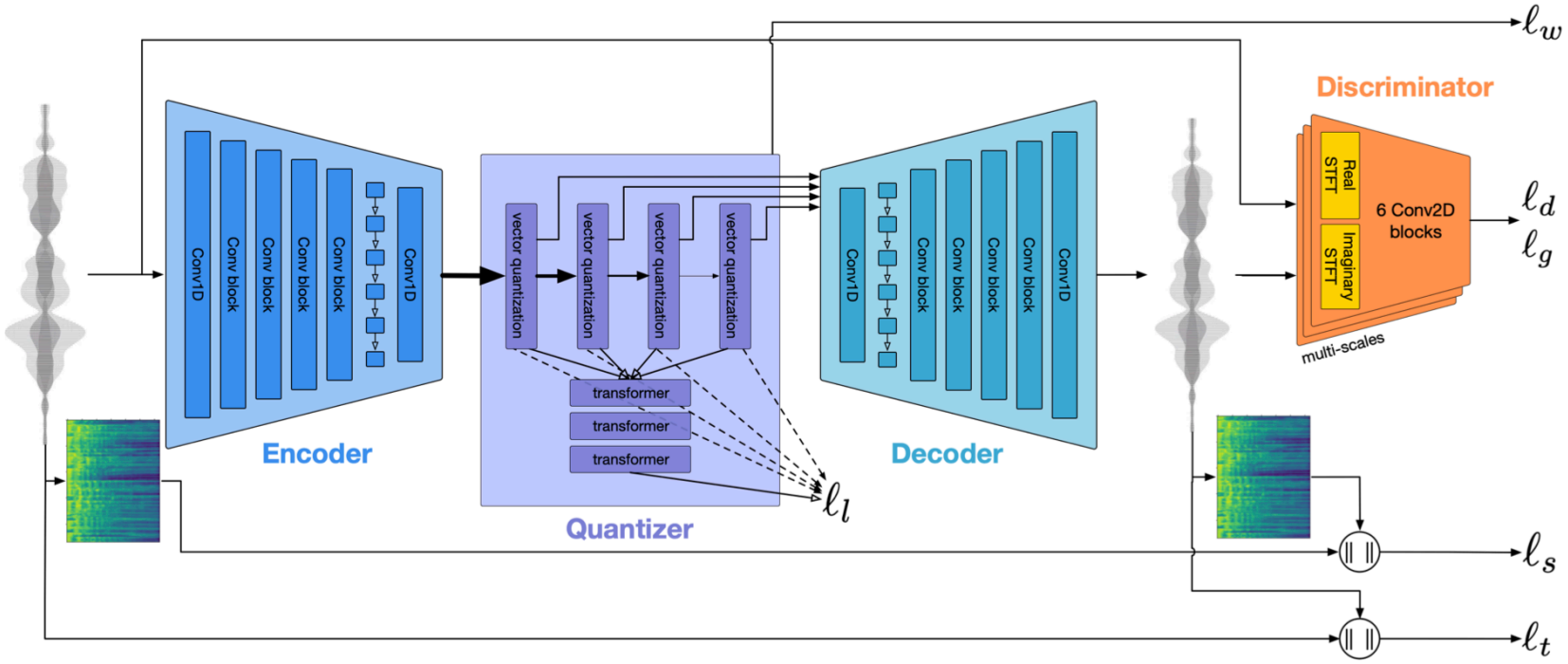




Residual Vector Quantization

- The first layer quantizes the code vectors with moderate resolution, and each of the following layers processes the residual error from the previous one.
- Using residual vector quantization, we can reduce the codebook size dramatically.
- The real beauty however is that the algorithm is scalable:
  - If less bandwidth is available, we can restrict the transmission to only the first vector quantization steps
  - If more bandwidth is available, we can transmit more bits (residual signal parts), improving the signal quality.

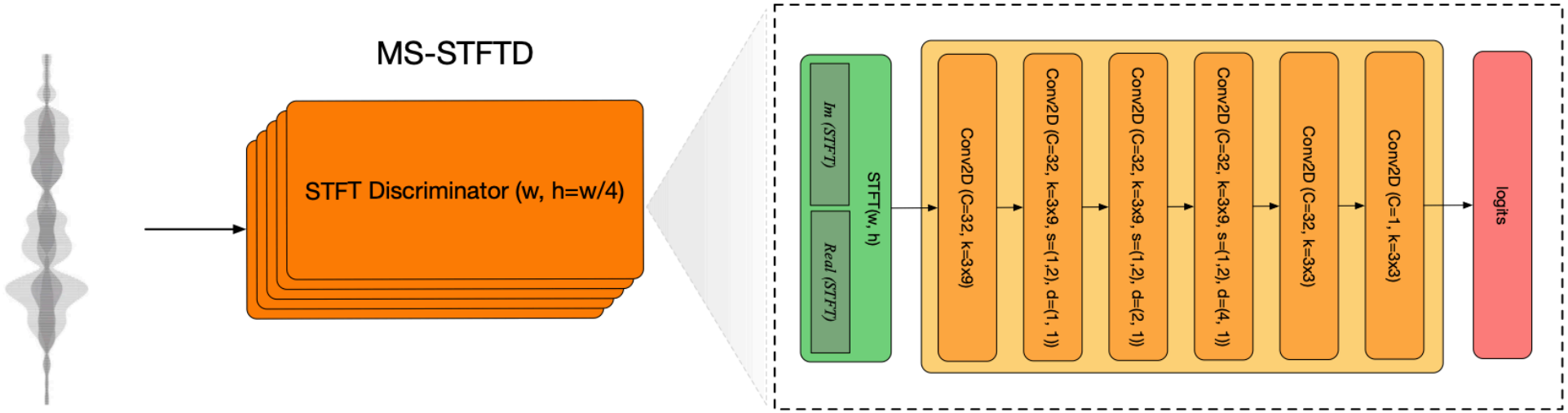
Encodec (Défossez et al., 2022)



- Very similar approach to SoundStream, but optimized for high-quality audio.
- Supports sampling rates up to 48kHz.
- Uses L1-Loss for auto-encoding, and a STFT-based GAN loss (see next slides).

https://arxiv.org/pdf/2210.13438

Encodec: STFT-based Discriminator



- Encoded uses a discriminator that utilizes a short Fourier transform-decomposed version (time-frequency representation) of the waveform.
- Therein the real and imaginary part are concatenated, followed by a couple of convolutional layers.

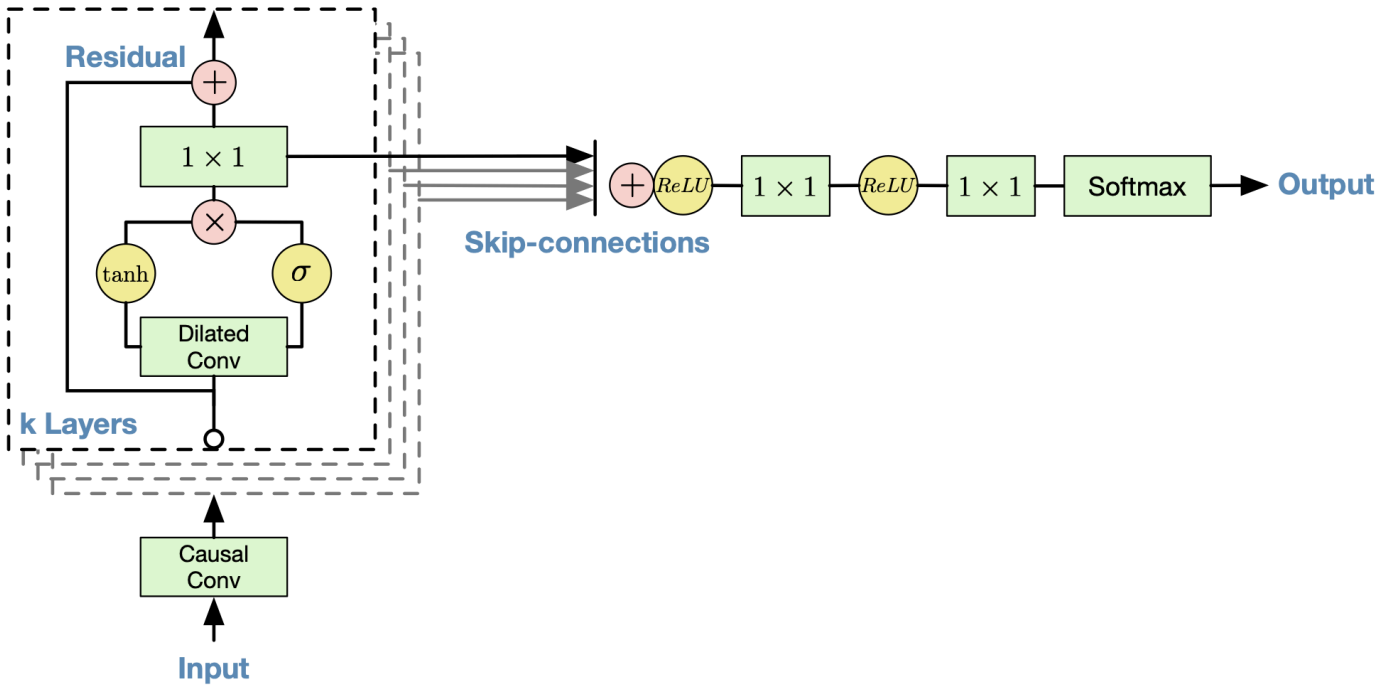
Hochschule
Flensburg
University of
Applied Sciences

Text-to-Speech and Speech-to-Text

WaveNet (van den Oord, 2016)

Hochschule Flensburg
University of Applied Sciences

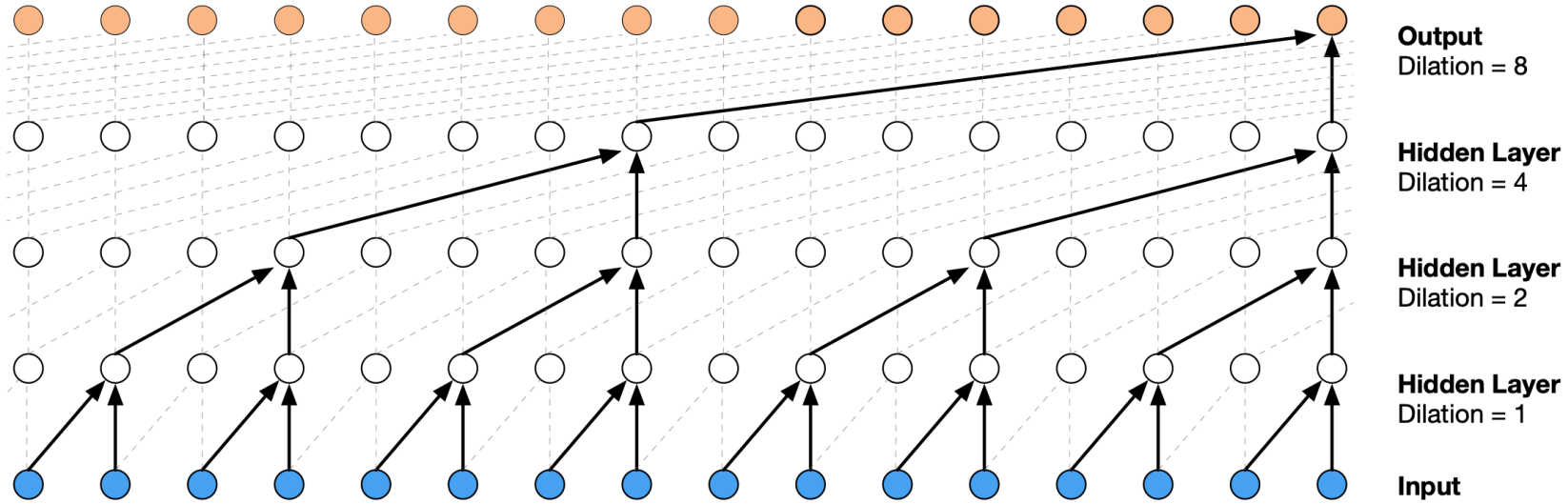
- Direct generation of audio waveforms, based on the PixelCNN architecture.
- Uses causal convolutions (utilizing samples from the past to predict the future)
- They introduce dilated convolutions (known from computer vision) to capture long-range dependencies.



https://arxiv.org/pdf/1609.03499

Causal Dilated Convolutions

- To enhance the receptive field of the PixelCNN-based architecture in WaveNet, the authors use dilation factors in the convolutions.
- These dilations efficiently enhance the long-range context for audio generation.

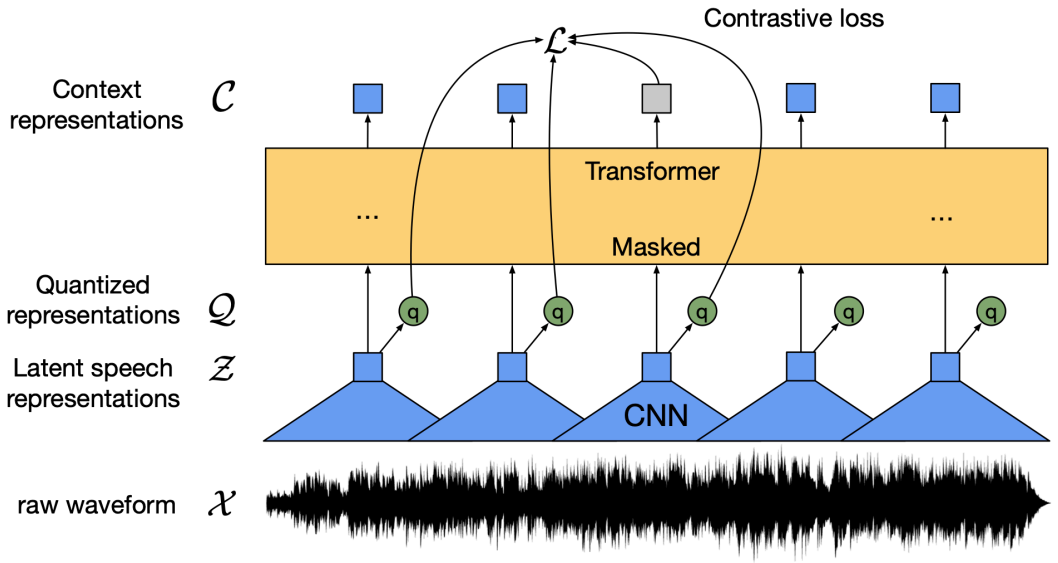


https://arxiv.org/pdf/1609.03499

wav2vec 2.0 (Baevski et al., NeurIPS, 2020)

Hochschule Flensburg
University of Applied Sciences

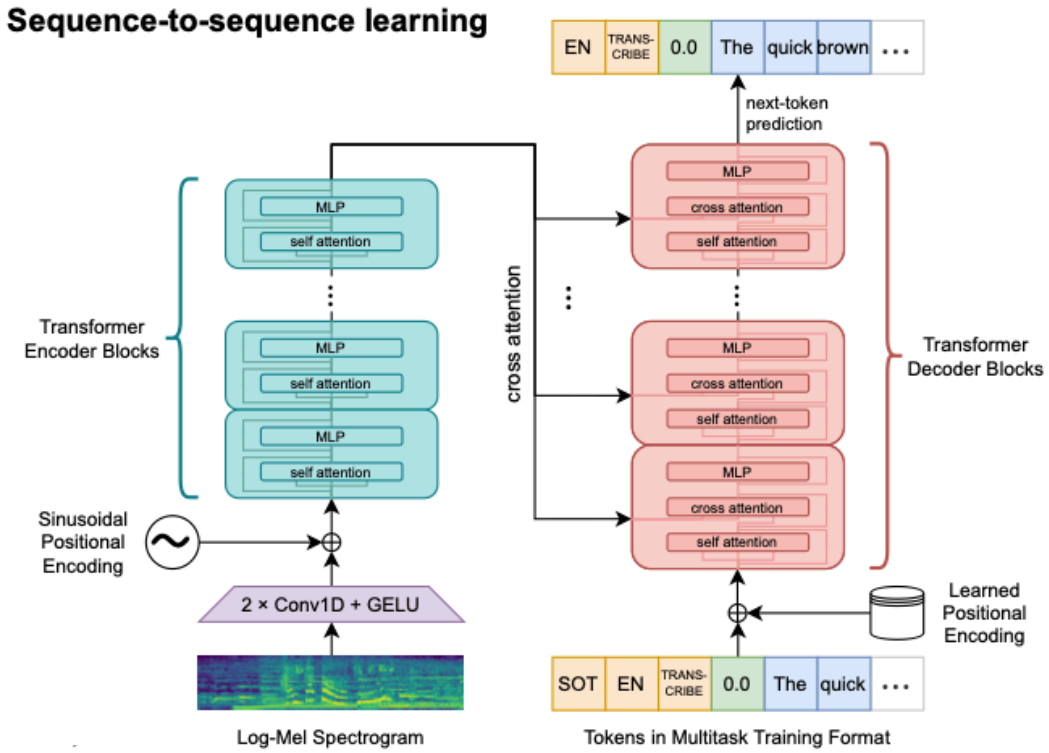
- Given the wide availability of unlabeled audio data, the authors propose a self-supervised training paradigm on speech.
- For this, they use the raw waveform as input, preprocessed by CNN kernels.
- They then use masked transformer training, where the model was tasked to predict the masked encoded tokens.
- The training objective contrasts the true quantized target with distractor samples (contrastive loss).



- This forces the model to learn meaningful speech units without labels.
- For fine-tuning, a small head is added for ASR, requiring much less labeled data.

Whisper (Radford et al., ICML 2023)

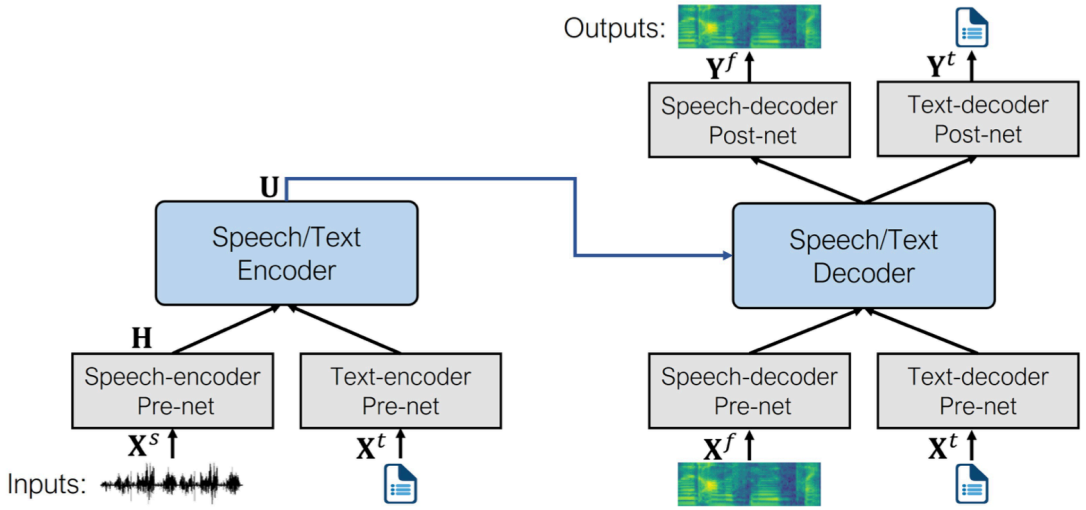
- Whisper is a speech-to-text approach, trained on a large-scale dataset from the internet (680k hours).
- It can perform various tasks beyond just transcription, including language identification, translation from the detected language to English, and voice activity detection (VAD).
- Whisper's training uses a mixed-task objective, where it is simultaneously trained to predict both the transcript (for transcription) and a special token sequence (for translation and language ID), allowing a single model to handle multiple tasks.



https://arxiv.org/pdf/2212.04356

SpeechT5 (Ao et al., ACL 2022)

- Based on the success of the T5 model, the authors of SpeechT5 also use large-scale pretraining in the domain of speech.
- Their architecture consists of an encoder-decoder architecture with multi-modal input and output blocks (pre-nets)

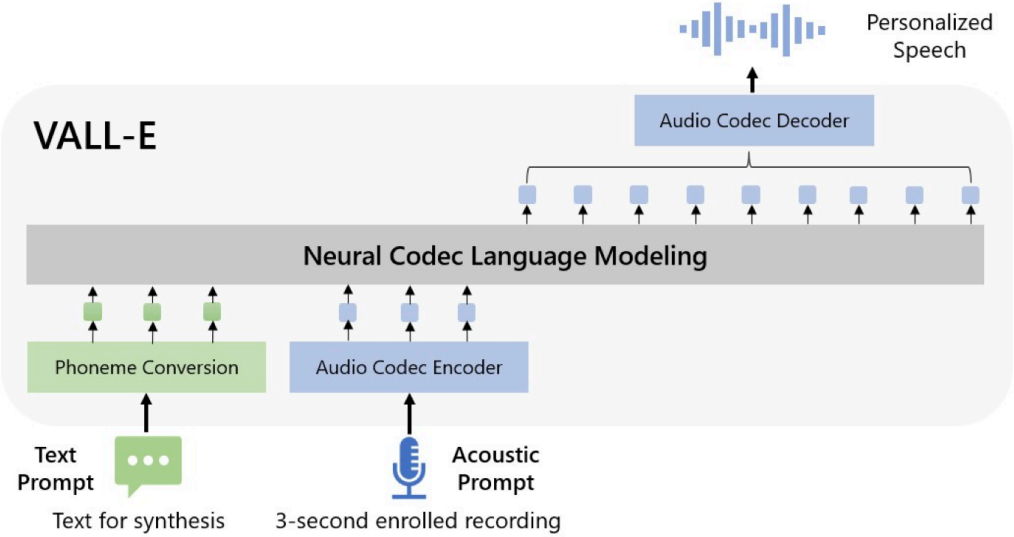


(a) The model architecture of SpeechT5

VALL-E (Wang et al., 2023)

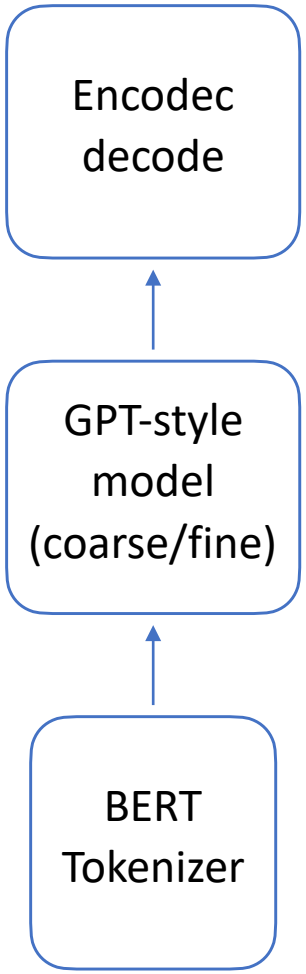
Hochschule Flensburg
University of Applied Sciences

- One important problem in text-to-speech is that the characteristics of the voice are sampled at random or fixed in typical TTS.
- VALL-E uses a short audio sample, referred to as acoustic prompt (e.g., 3 seconds) of an unseen speaker as an acoustic prompt to condition the TTS generation.
- It operates on discrete audio tokens produced by a neural codec (e.g., EnCodec).
- Speech is represented as a sequence of quantized codec codes, not spectrograms.
- Prediction is performed auto-regressively.



Bark 🐶 (Suno AI, 2023)

- Multilingual Text-to-Speech system
- Also features Non-speech audio generation:
  ambient noise, breathing, emotional cues.
- Highly expressive prosody, sounding more human and dynamic.
- Based on the GPT architecture
- Uses the BERT tokenizer
- Trained on a large, diverse audio-text dataset
- Generates tokens in the space of the Encodec model



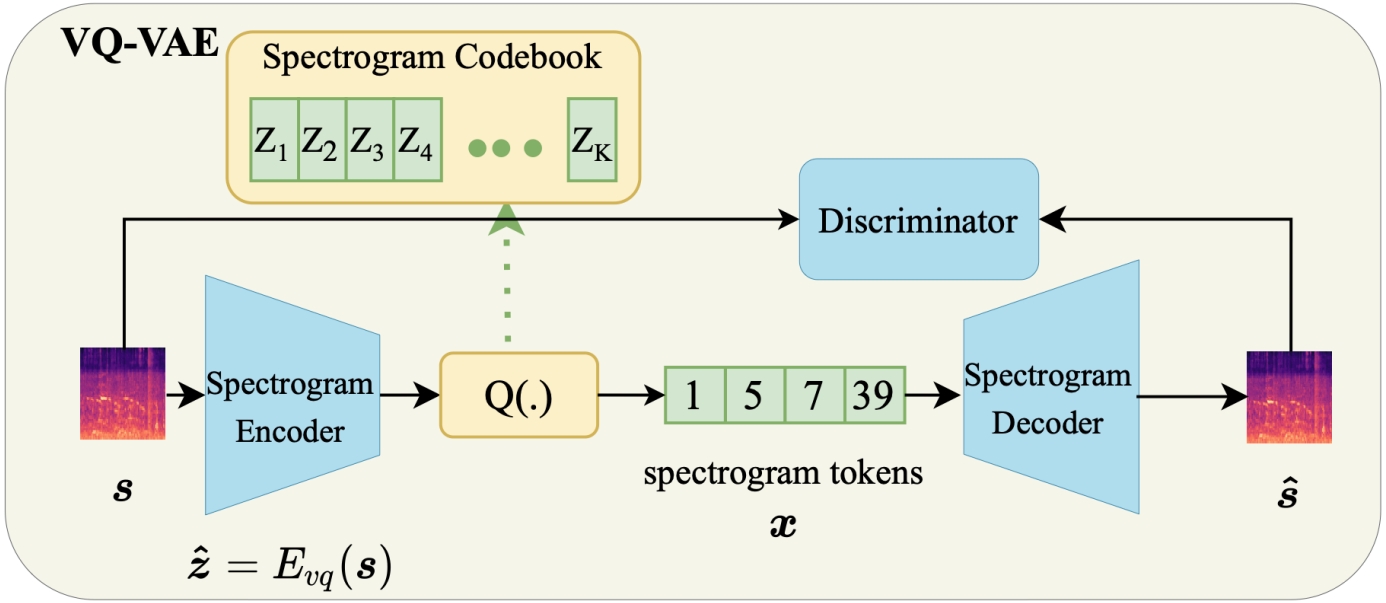
Hochschule Flensburg
University of Applied Sciences

https://github.com/suno-ai/bark/blob/main/bark/model.py

Hochschule
Flensburg
University of
Applied Sciences

Text-conditional Audio Generation (Text2Audio, TTA)

DiffSound (Yang et al., 2021)



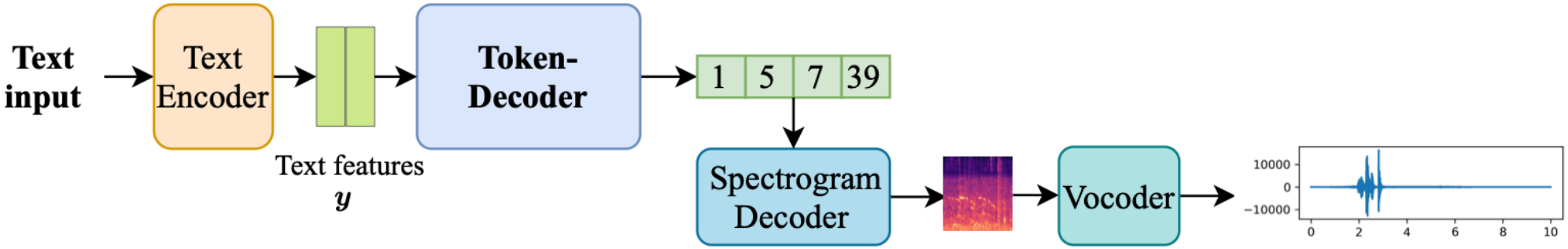
- The authors propose to train a diffusion model in the latent space of a VQ-VAE trained on Mel spectrograms.

https://arxiv.org/pdf/2207.09983

DiffSound

Hochschule Flensburg
University of Applied Sciences

- The overall architecture uses a text encoder as conditioning to the spectrogram decoder.

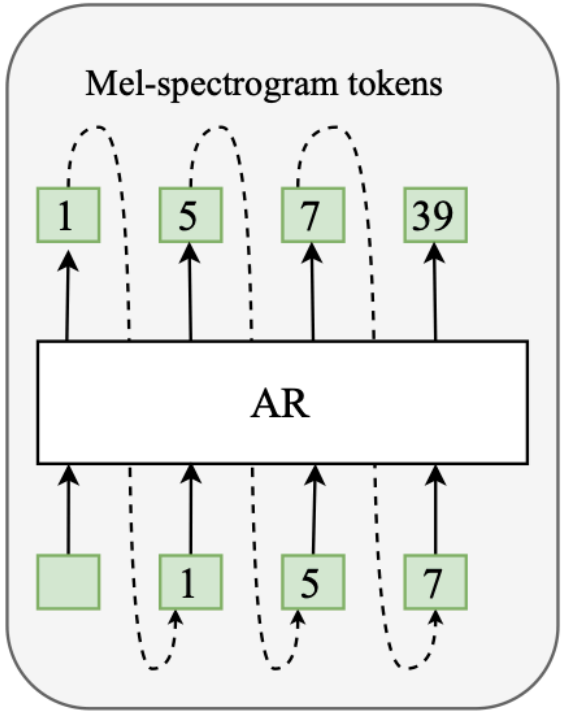


- The authors evaluate both auto-regressive and diffusion-based token generation, based on a transformer.
- The approach is not focused on speech, but on general audio (generated from descriptions)

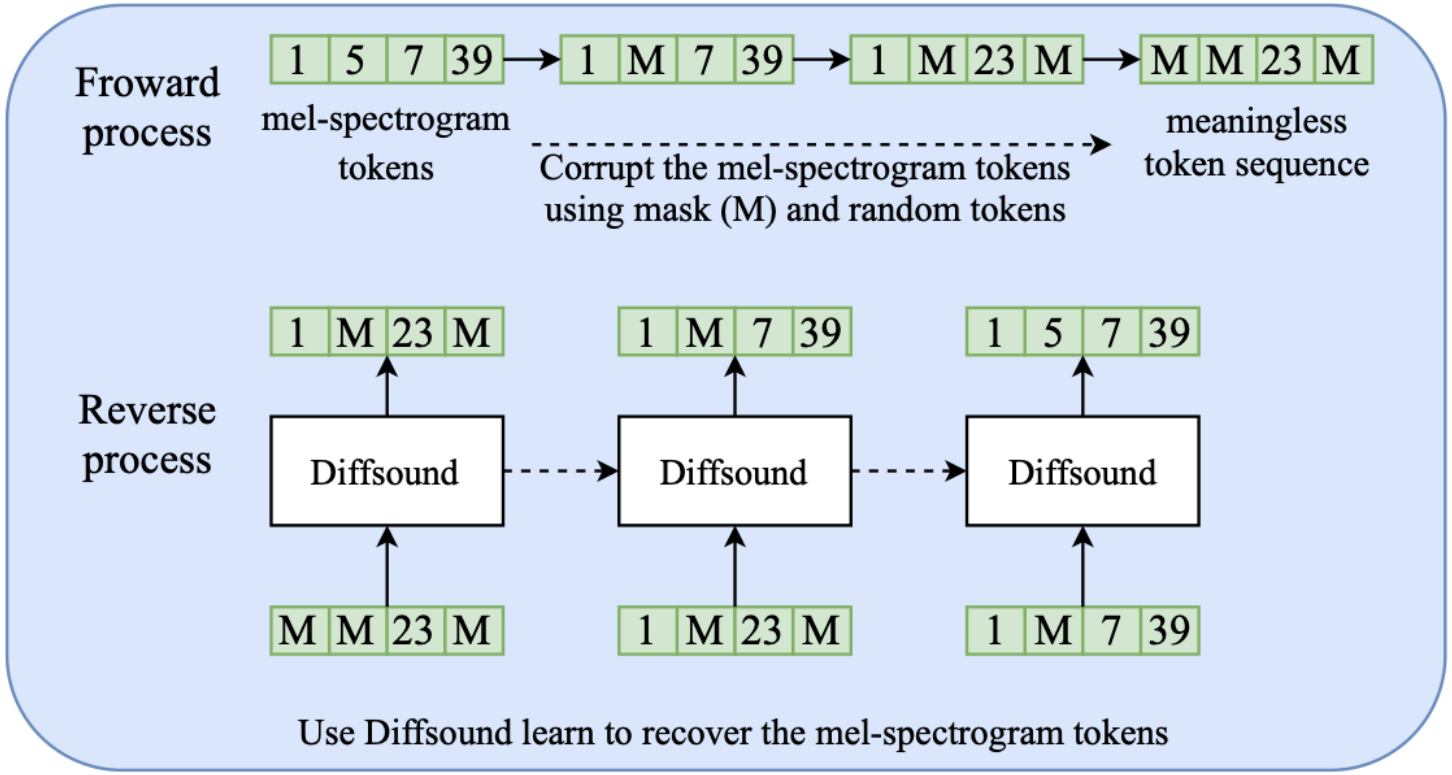
https://arxiv.org/pdf/2207.09983

DiffSound: Token prediction

Hochschule Flensburg
University of Applied Sciences



(b) An example of autoregressive spectrogram tokens generation



(c) An example of non-autoregressive spectrogram tokens generation.

https://arxiv.org/pdf/2207.09983

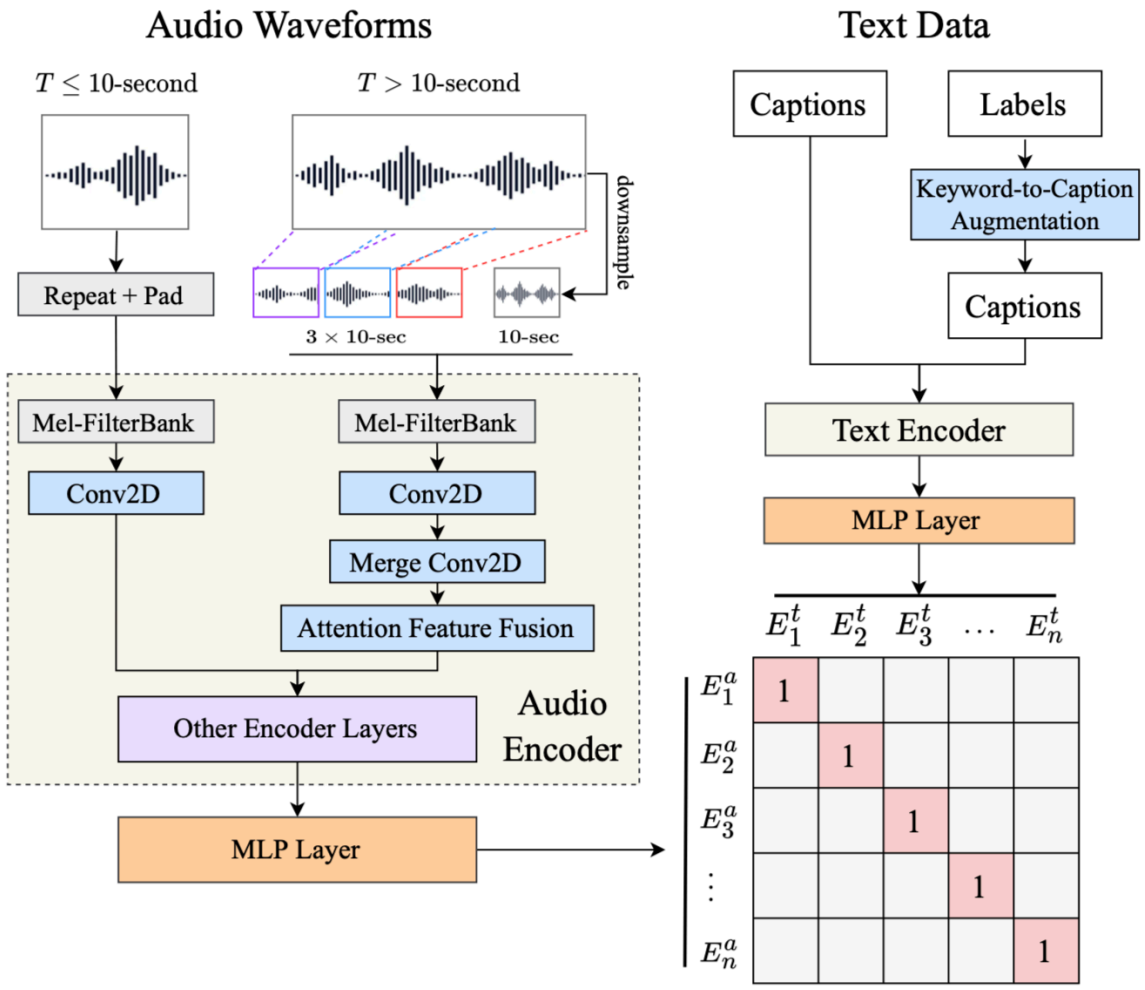
Examples:

Hochschule Flensburg
University of Applied Sciences

A dog barks and whimpers		
	autoregressive	diffusion
A person is snoring while sleeping		
	autoregressive	diffusion

https://dongchaoyang.top/text-to-sound-synthesis-demo/

Contrastive Language Audio Pretraining (CLAP, Wu et al., ICASSP 2023)

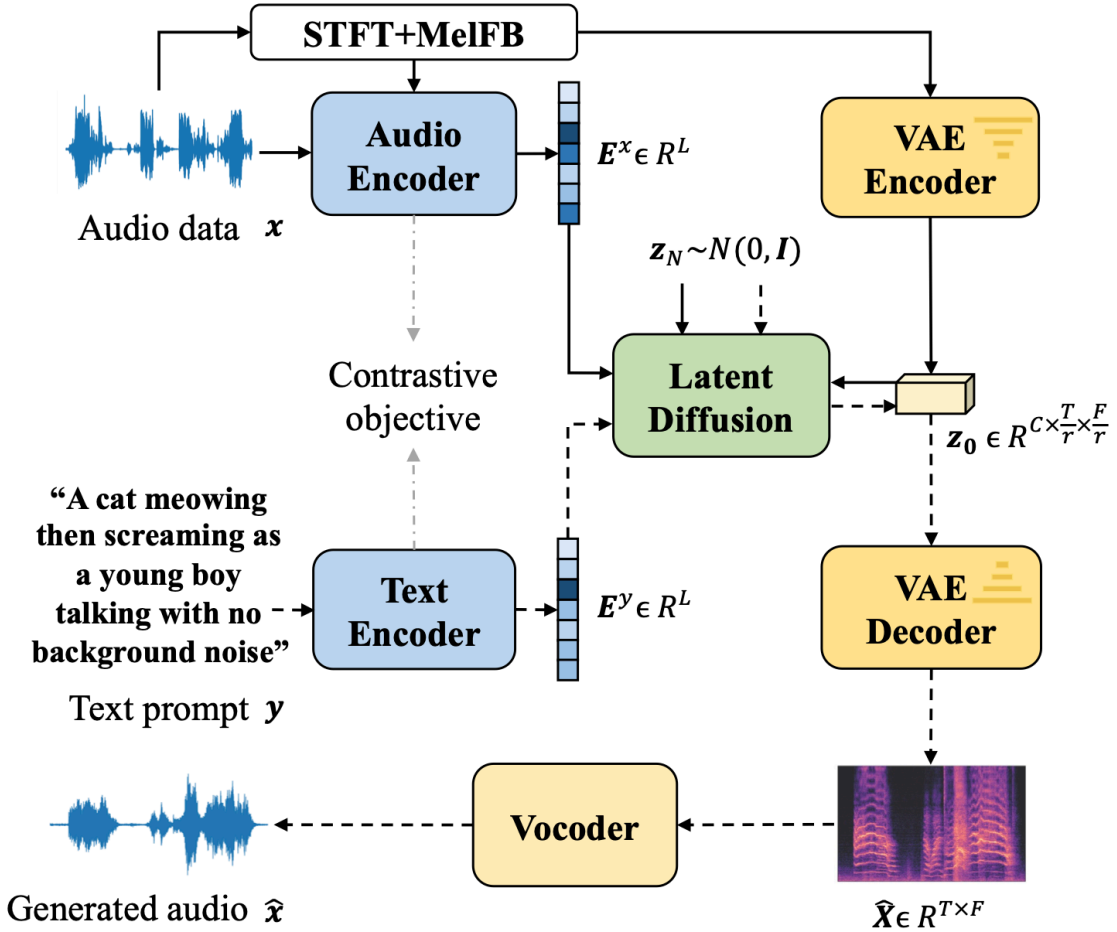


- Following the idea of CLIP in the image domain, the authors propose to use joint training of audio waveforms and text data to yield an aligned latent representation.
- They train on the LAION-Audio-630k dataset, a dataset of 633 thousand pairs of text and audio, with a total duration of ~4300 hours.
- Similarly to CLIP, they also show that this can be used for zero-shot classification of audio files.

https://arxiv.org/pdf/2301.12503

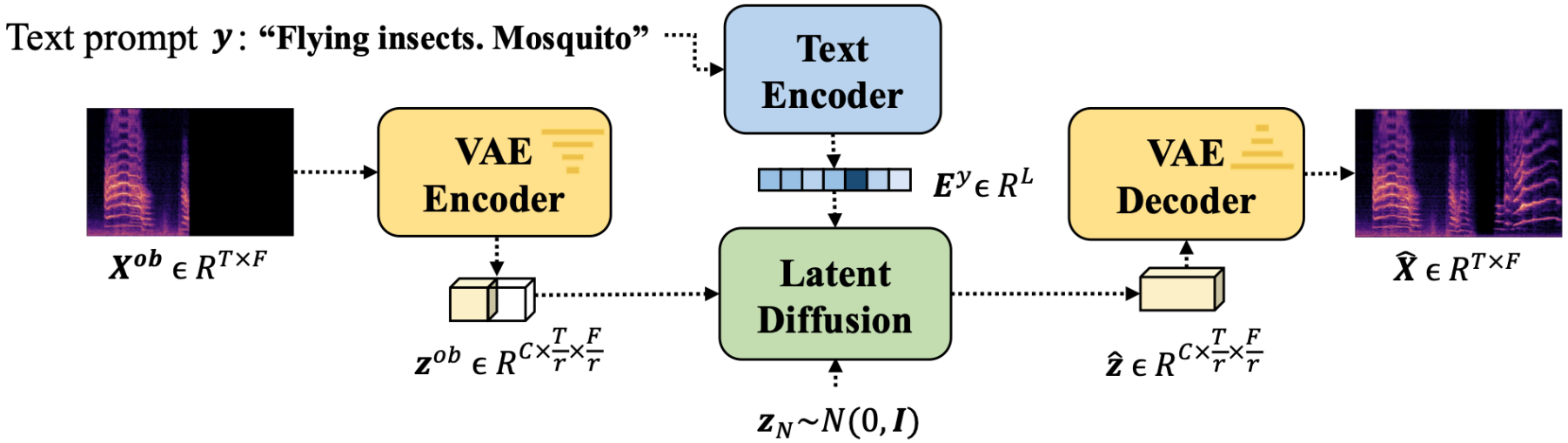
AudioLDM (Liu et al., ICML 2023)

- AudioLDM utilizes CLAP latents to generate new audio samples.
- The use a diffusion model that was conditioned on the CLAP latents.
- The diffusion model learns to denoise latent representations step-by-step toward realistic audio.
- Conditioning on CLAP latents guides the generative process toward the desired semantic content.
- Finally, a decoder reconstructs the full audio waveform from the denoised latent representation.



https://arxiv.org/pdf/2301.12503

AudioLDM: Inpainting

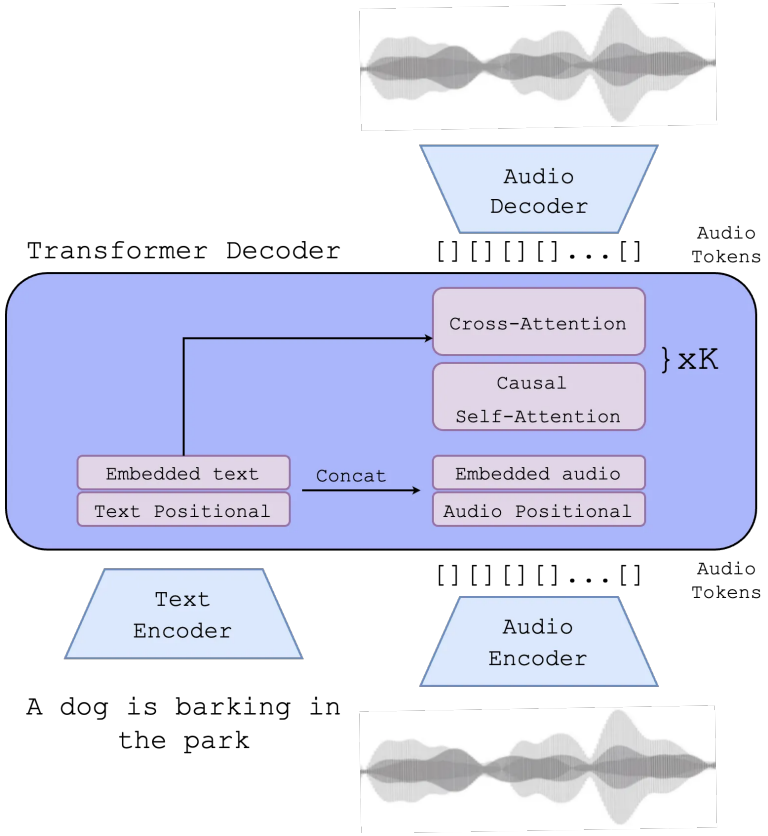


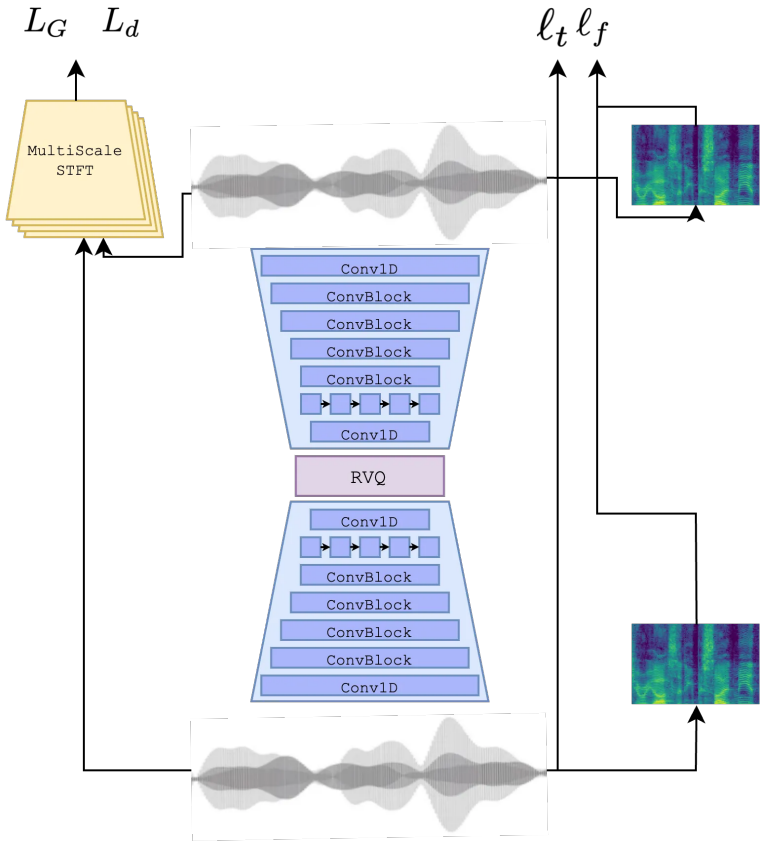
- The diffusion process also allows for inpainting of masked segments.
- In this, they reformulate the sampling process to overwrite the non-inpainted segments with their original value, with the masked samples starting from noise and being continuously denoised.

https://arxiv.org/pdf/2301.12503

AudioGen (Kreuk et al., ICLR 2023)

- AudioGen is a more recent text-conditional audio synthesis framework.
- They employ a special augmentation technique that mixes different audio samples to counter overfitting.
- They furthermore apply GAN-based training to improve the quality of the generated samples.





AudioGen: Examples

Hochschule Flensburg
University of Applied Sciences

male speech with horns honking in the background

SoundDiff

AudioGen

a baby continuously crying

SoundDiff

AudioGen

https://felixkreuk.github.io/audiogen/

Hochschule
Flensburg
University of
Applied Sciences

Music Generation

Modalities

- Symbolic music generation
  - Produces scores/MIDI (notes, tempo, chords).
  - Easier to structure and edit.
- Spectrogram generation
  - Needs additional synthesis step
  - Using approaches from computer vision
- Audio waveform generation
  - Direct creation of raw audio (e.g., diffusion models, GANs).
  - High fidelity but computationally intensive.

Pitch

Hochschule Flensburg
University of Applied Sciences

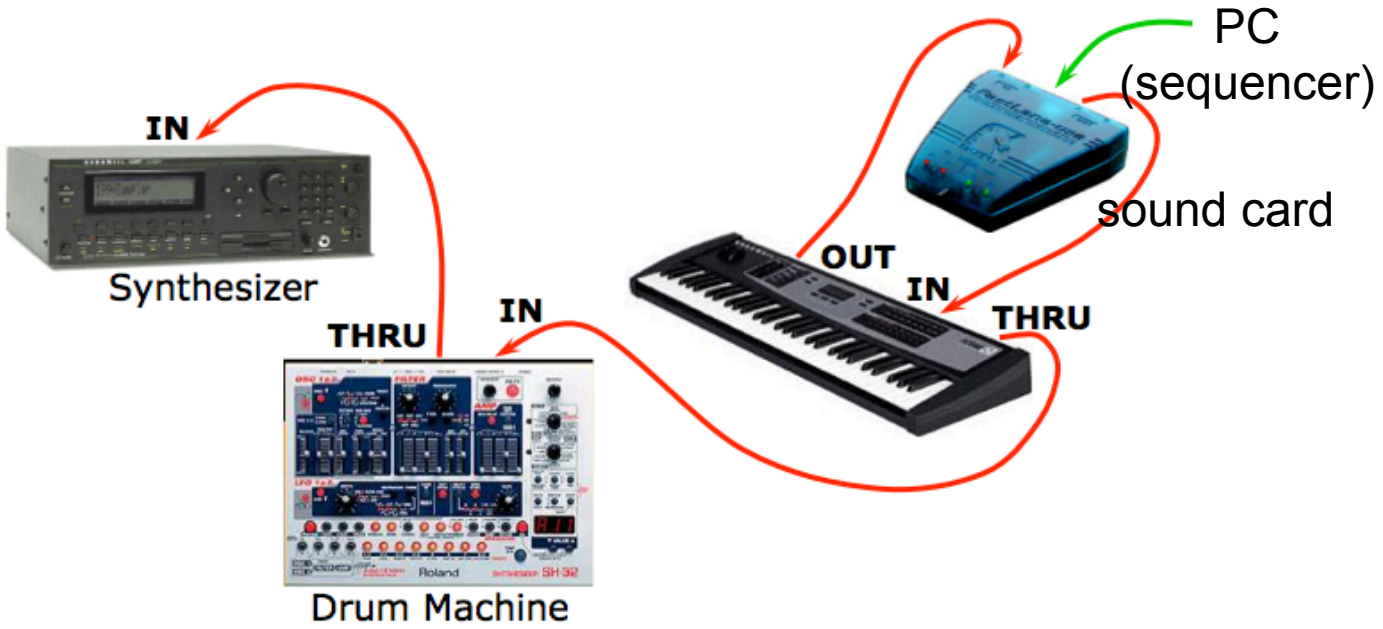
- Auditory sensation: At what frequency do we hear a sound?
- The pitch is commonly estimated as the fundamental frequency of a signal.
- For music instruments, there is a correspondence between pitch and note.
- Pitch can also be estimated for speech and singing.

Note	Sub-contra	Contra	Great	Small	One-lined	Two-lined	Three-lined	Four-lined	Five-lined
B♯/C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01
C♯/D♭	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64
D♯/E♭	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03
E/F♭	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04
E♯/F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65
F♯/G♭	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91
G	24.50	49.00	98.00	196.00	392.00	783.99	1567.99	3135.96	6271.93
G♯/A♭	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00
A♯/B♭	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62
B/C♭	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13

MIDI (Musical Instruments Digital Interface)

Hochschule Flensburg
University of Applied Sciences

- Technical standard for the connection of digital music instruments (and computers)



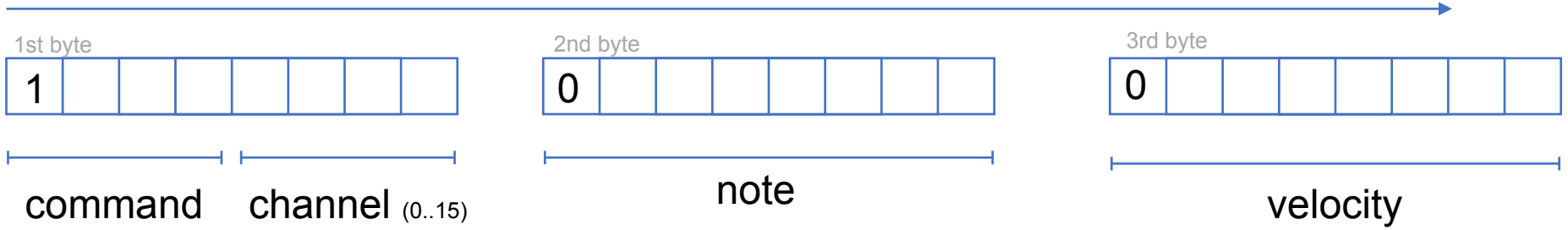
https://cecm.indiana.edu/361/midi-prev.html

standardized MIDI connector

- The actual sound wave is generated by the synthesizer

How does MIDI work?

- Over MIDI, an instrument can communicate several channels containing music notes (with volume and length)



how fast / strong is a note pressed

value	command
1000	Note Off
1001	Note On
1010	Polyphonic Pressure
1011	Control Change
1100	Programm Change
1101	Channel Pressure
1110	Pitch Bending
1111	System Exclusive

What's not included in MIDI: How does that instrument sound?

MIDI files

- The information from a MIDI stream can also be stored i
- Files have the extension .mid.
- Compared to wave files they are tiny.
- Information contained in MIDI files:
  - Playback speed (time division)
  - Number of tracks included
  - The actual tracks (including music instrument selection)

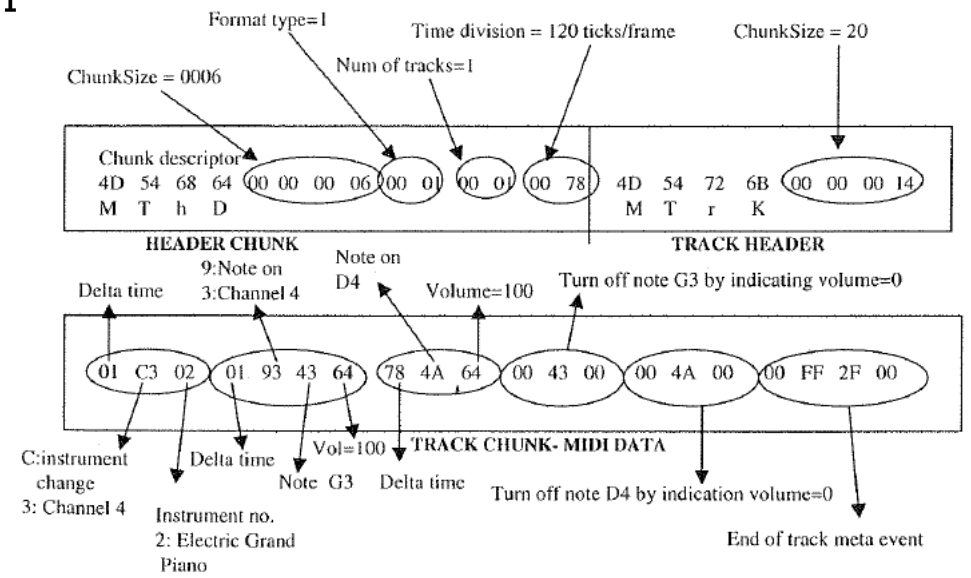


Figure 2.12. Explanation of the MIDI file in Example 2.2.

http://www.cs.uccs.edu/~cs525/midi/midi.html

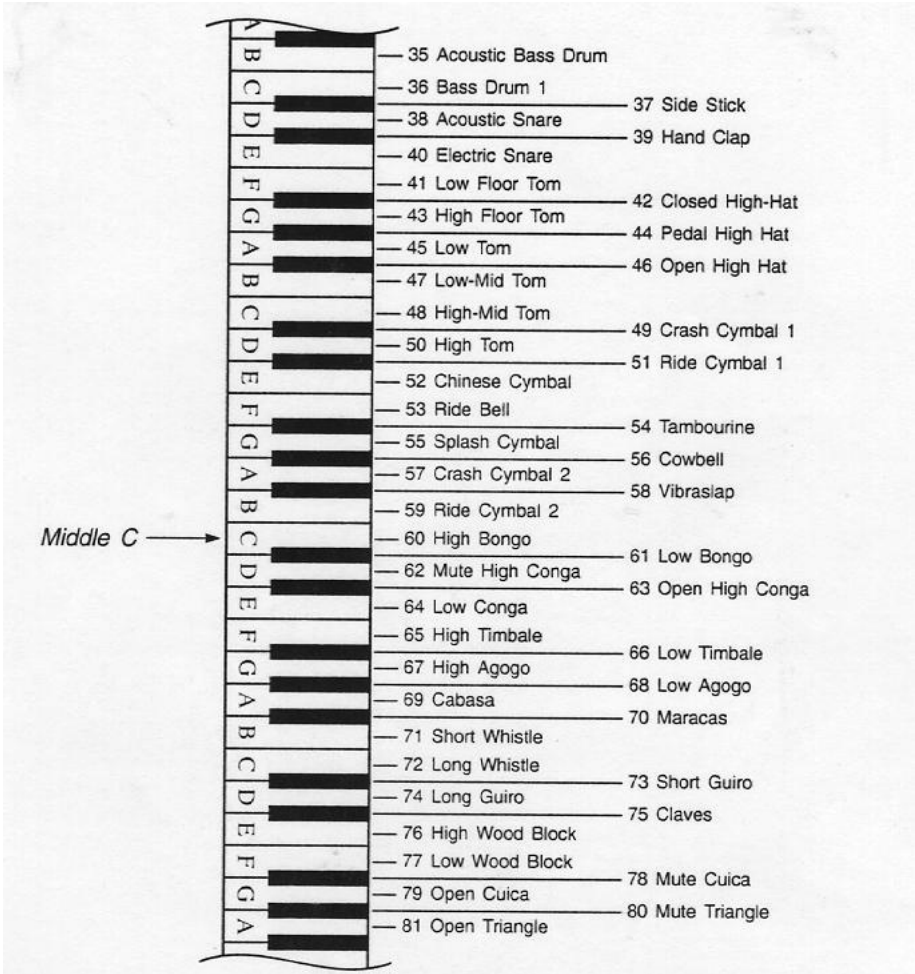
MIDI music instruments

standard music instruments

Piano	Chromatic Percussion	Organ	Guitar
0 Acoustic Grand Piano	8 Celesta	16 Hammond Organ	24 Acoustic Guitar (nylon)
1 Bright Acoustic Piano	9 Glockenspiel	17 Percussive Organ	25 Acoustic Guitar (steel)
2 Electric Grand Piano	10 Music box	18 Rock Organ	26 Electric Guitar (jazz)
3 Honky-tonk Piano	11 Vibraphone	19 Church Organ	27 Electric Guitar (clean)
4 Rhodes Piano	12 Marimba	20 Reed Organ	28 Electric Guitar (muted)
5 Chorused Piano	13 Xylophone	21 Accordion	29 Overdriven Guitar
6 Harpsichord	14 Tubular Bells	22 Harmonica	30 Distortion Guitar
7 Clavinet	15 Dulcimer	23 Tango Accordion	31 Guitar Harmonics
Bass	**Strings**	**Ensemble**	**Brass**
32 Acoustic Bass	40 Violin	48 String Ensemble 1	56 Trumpet
33 Electric Bass (finger)	41 Viola	49 String Ensemble 2	57 Trombone
34 Electric Bass (pick)	42 Cello	50 SynthStrings 1	58 Tuba
35 Fretless Bass	43 Contrabass	51 SynthStrings 2	59 Muted Trumpet
36 Slap Bass 1	44 Tremolo Strings	52 Choir Aahs	60 French Horn
37 Slap Bass 2	45 Pizzicato Strings	53 Voice Oohs	61 Brass Section
38 Synth Bass 1	46 Orchestral Harp	54 Synth Voice	62 Synth Brass 1
39 Synth Bass 2	47 Timpani	55 Orchestra Hit	63 Synth Brass 2
Reed	**Pipe**	**Synth Lead**	**Synth Pad**
64 Soprano Sax	72 Piccolo	80 Lead 1 (square)	88 Pad 1 (new age)
65 Alto Sax	73 Flute	81 Lead 2 (sawtooth)	89 Pad 2 (warm)
66 Tenor Sax	74 Recorder	82 Lead 3 (caliope lead)	90 Pad 3 (polysynth)
67 Baritone Sax	75 Pan Flute	83 Lead 4 (chiff lead)	91 Pad 4 (choir)
68 Oboe	76 Bottle Blow	84 Lead 5 (charang)	92 Pad 5 (bowed)
69 English Horn	77 Shakuhachi	85 Lead 6 (voice)	93 Pad 6 (metallic)
70 Bassoon	78 Whistle	86 Lead 7 (fifths)	94 Pad 7 (halo)
71 Clarinet	79 Ocarina	87 Lead 8 (brass + lead)	95 Pad 8 (sweep)
Synth Effects	**Ethnic**	**Percussive**	**Sound Effects**
96 FX 1 (rain)	104 Sitar	112 Tinkle Bell	120 Guitar Fret Noise
97 FX 2 (soundtrack)	105 Banjo	113 Agogo	121 Breath Noise
98 FX 3 (crystal)	106 Shamisen	114 Steel Drums	122 Seashore
99 FX 4 (atmosphere)	107 Koto	115 Woodblock	123 Bird Tweet
100 FX 5 (brightness)	108 Kalimba	116 Taiko Drum	124 Telephone Ring
101 FX 6 (goblins)	109 Bagpipe	117 Melodic Tom	125 Helicopter
102 FX 7 (echoes)	110 Fiddle	118 Synth Drum	126 Applause
103 FX 8 (sci-fi)	111 Shanai	119 Reverse Cymbal	127 Gunshot

http://www.cs.uccs.edu/~cs525/midi/midi.html

drum assignment according to notes

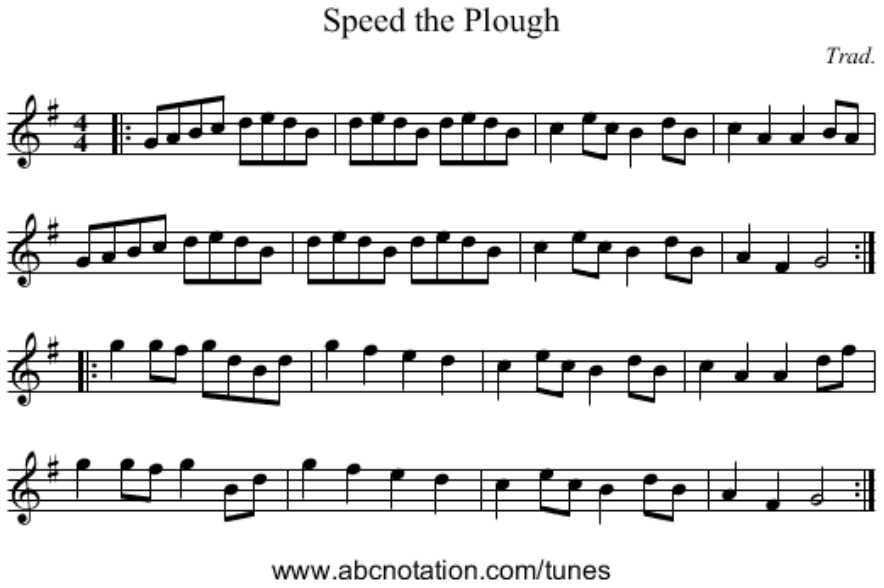


Standard MIDI key assignments for key-based percussion instruments.

ABC notation

- As an alternative to MIDI, the ABC notation was introduced.
- It uses simple ASCII characters to encode melodies, chords, and rhythms.
- It's human-readable and easy to edit, and can also be easily processed by a LLM.
- It also includes header fields for title/ryhthm/tune/etc.

```
X:1
T:Speed the Plough
M:4/4
C:Trad.
K:G
|:GABc dedB|dedB dedB|c2ec B2dB|c2A2 A2BA|
  GABc dedB|dedB dedB|c2ec B2dB|A2F2 G4:|
|:g2gf gdBd|g2f2 e2d2|c2ec B2dB|c2A2 A2df|
  g2gf g2Bd|g2f2 e2d2|c2ec B2dB|A2F2 G4:|
```



Text Conditioning

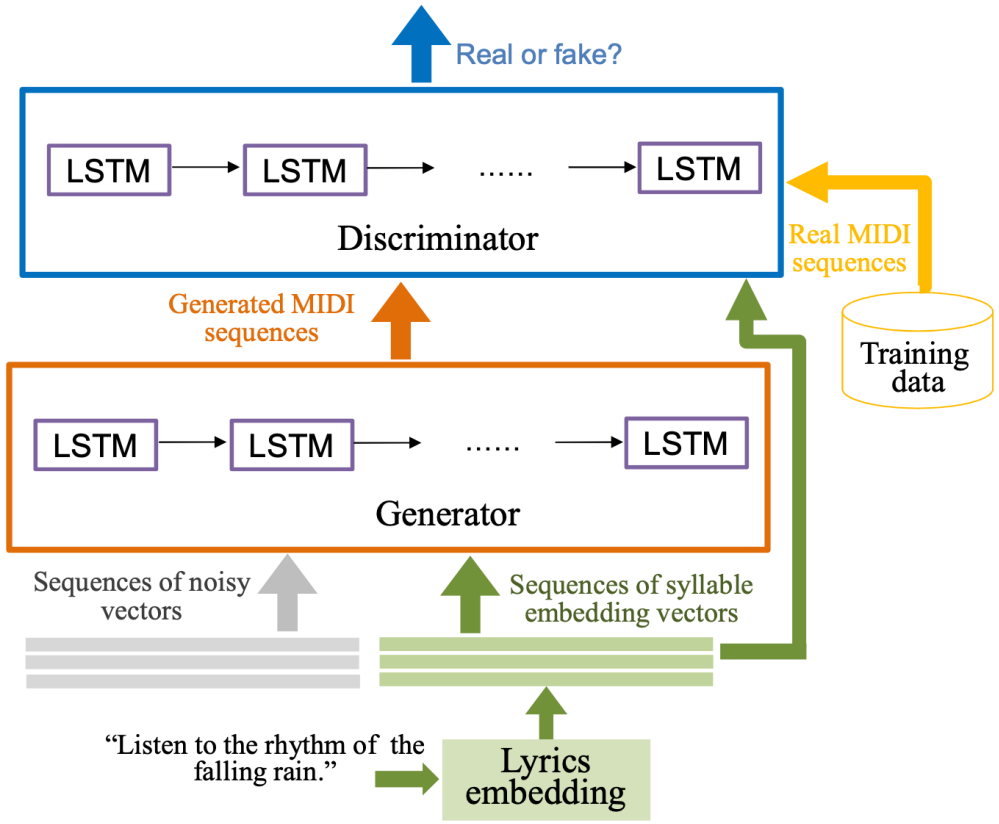
- Text conditioning can have various meanings in music generation:

	Lyrics	Musical attributes	Natural language description
e.g.,	"Let it be, let it be, ..."	120bpm, fortissimo	"Create a melody, filled with hope"

https://doi.org/10.3390/electronics14061197

LSTM-GAN: Melody generation from lyrics (Yu et al., 2021)

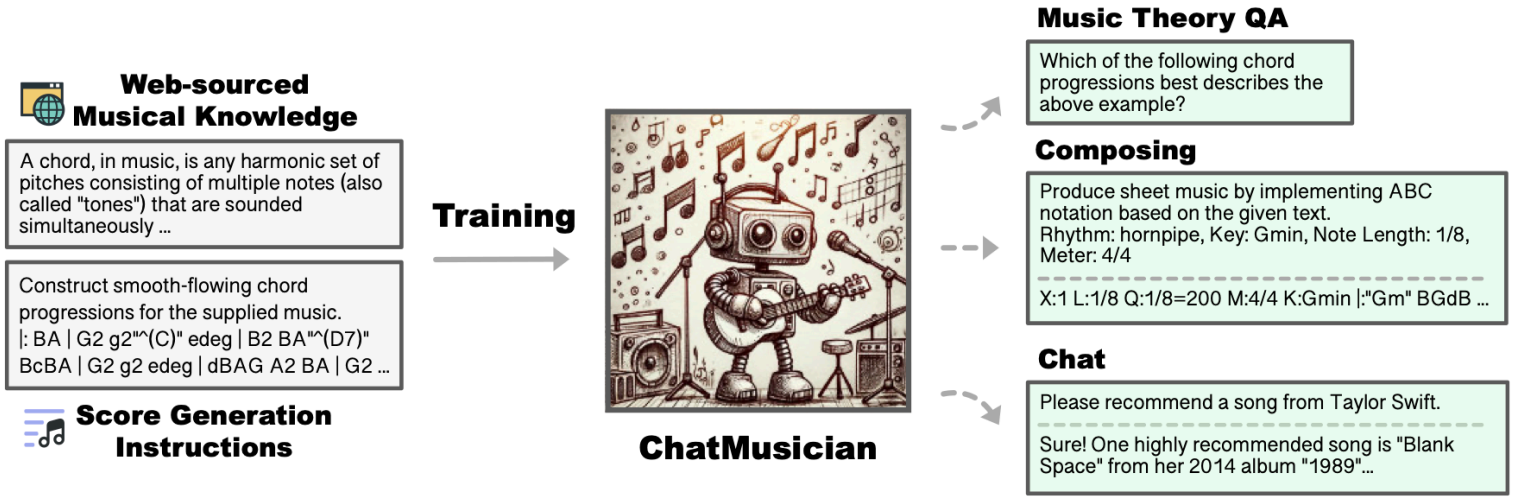
- Generation of symbolic representations for music
- Combination of LSTM and GAN
- Uses syllable embedding vectors converted from text lyrics
- Uses additional noise vector as entropy source



https://arxiv.org/pdf/1908.05551

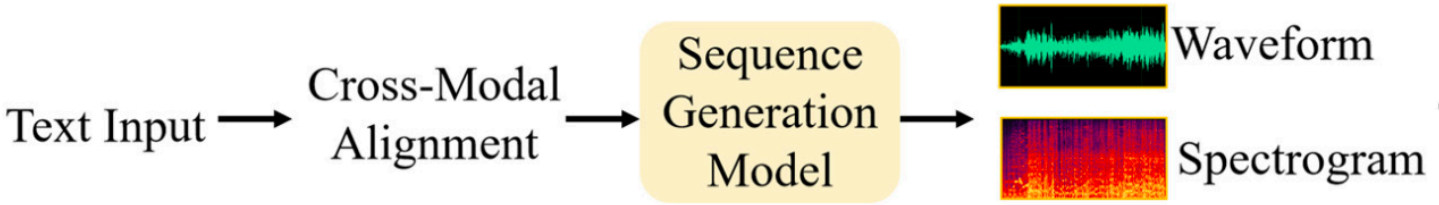
ChatMusician (Yuan et al., 2024)

- Idea: Use music as a further language for a LLM
- Based on the LLAMA2 model
- Uses ABC notation for music and text
- Was trained on 4B tokens, covering both music knowledge questions as well as synthetic music tokens.



https://arxiv.org/pdf/2402.16153

Signal-domain Generation



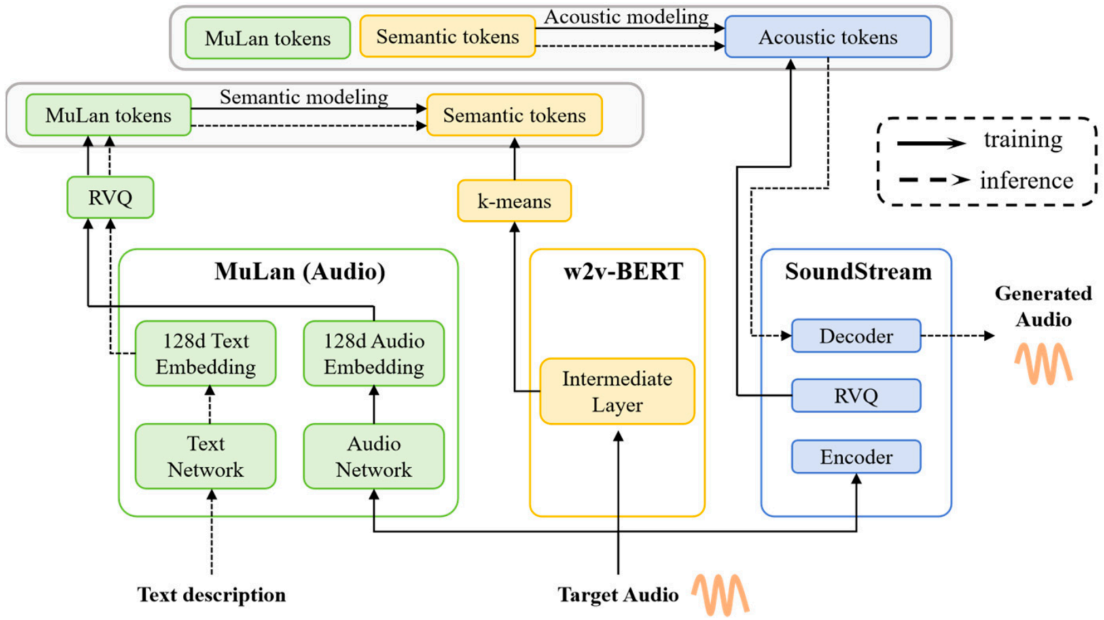
- Text-to-music generation is commonly done in two steps:
  - First, text input (e.g., natural language prompts or labels) is provided to a cross-modal alignment module, which interprets and maps the text into a representation suitable for audio generation
  - Next, a sequence generation model produces either raw waveforms or spectrogram information.

https://doi.org/10.3390/electronics14061197

MusicLM: Generating Music from Text (Agostinelli et al., 2023)

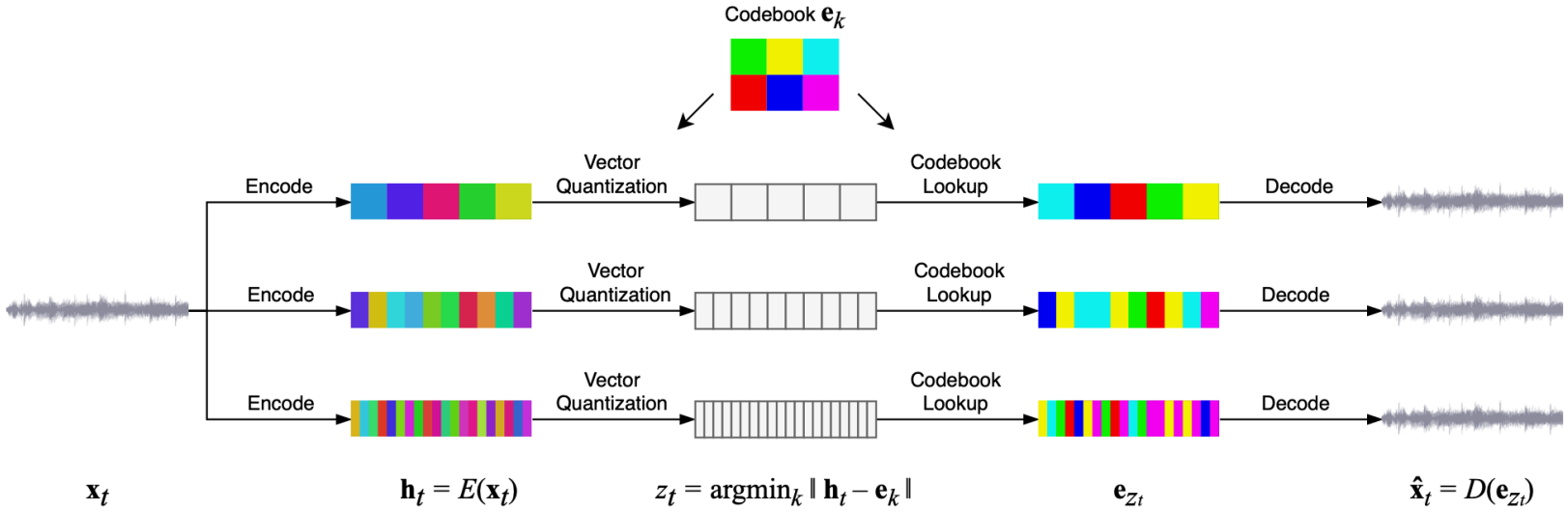
Hochschule Flensburg
University of Applied Sciences

- Utilization of a LLM for music generation.
- It generates music based on Audio conditioning, such as using a reference melody, style, or humming.
- The generation is based on three token hierarchies:
  - Joint embedding of text description with target audio tokens (pre-trained network MuLan)
  - Semantic tokens extracted from wave2vec-BERT
  - Acoustic tokens extracted from a SoundStream encoder.



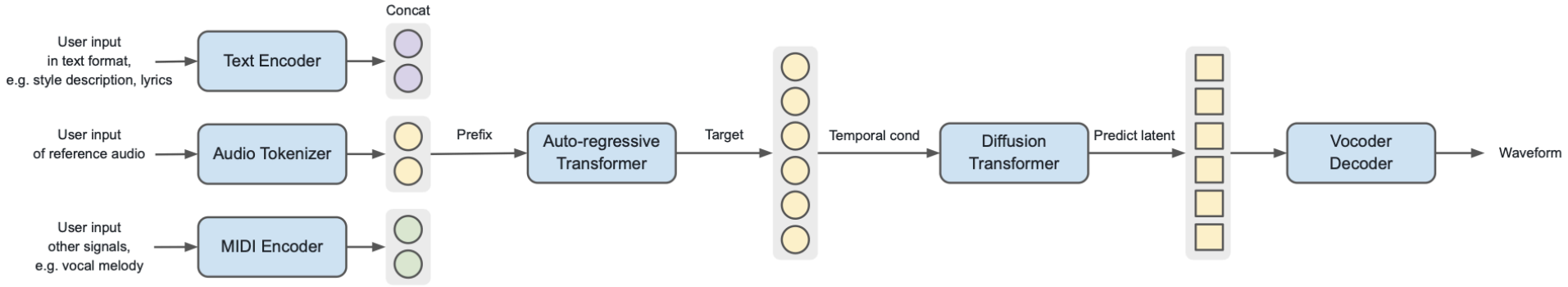
https://arxiv.org/pdf/2301.11325
https://doi.org/10.3390/electronics14061197

Jukebox (Dhariwal et al., 2020)



- Jukebox was one of the first approaches, directly modeling generation of entire songs in the waveform domain.
- The approach is based on three VQ-VAEs with different temporal resolutions.
- The upper one learns the most abstract representations, utilizing larger temporal blocks, while the lowest one yields the highest quality.

Seed-Music (Bai et al., 2024)



- Uses multiple conditionings to generate music using two stages of transformer networks (first: auto-regressive, second: DiT)
- Integrates autoregressive language modeling and diffusion methods to support two key workflows:
  - controlled music generation and
  - post-editing.

https://arxiv.org/pdf/2409.09214v1

Hochschule
Flensburg
University of
Applied Sciences

Summary

Generation of Audio: Summary

- Audio can be generated in different representations (waveform, MIDI, mel spectrogram, neural codec embedding space)
- Several methods that we already know from image and text generation are also applied in audio generation
  - GPT-style transformer models
  - Diffusion transformers
  - Adversarial loss functions
  - vector quantization
- Here, as well, the "magic ingredient" towards a great model is the scale and quality of data.