Discussion 11

EE599 Deep Learning

Kuan-Wen (James) Huang, Arnab Sanyal

Spring 2020

RNN Layers in tf.keras

- SimpleRNN
  - `tf.keras.layers.SimpleRNN`
- Gated Recurrent Unit
  - `tf.keras.layers.GRU`
- Long Short-Term Memory (LSTM)
  - `tf.keras.layers.LSTM`

- Reference:
  1. GRU: https://arxiv.org/pdf/1406.1078.pdf
  2. LSTM: https://www.bioinf.jku.at/publications/older/2604.pdf

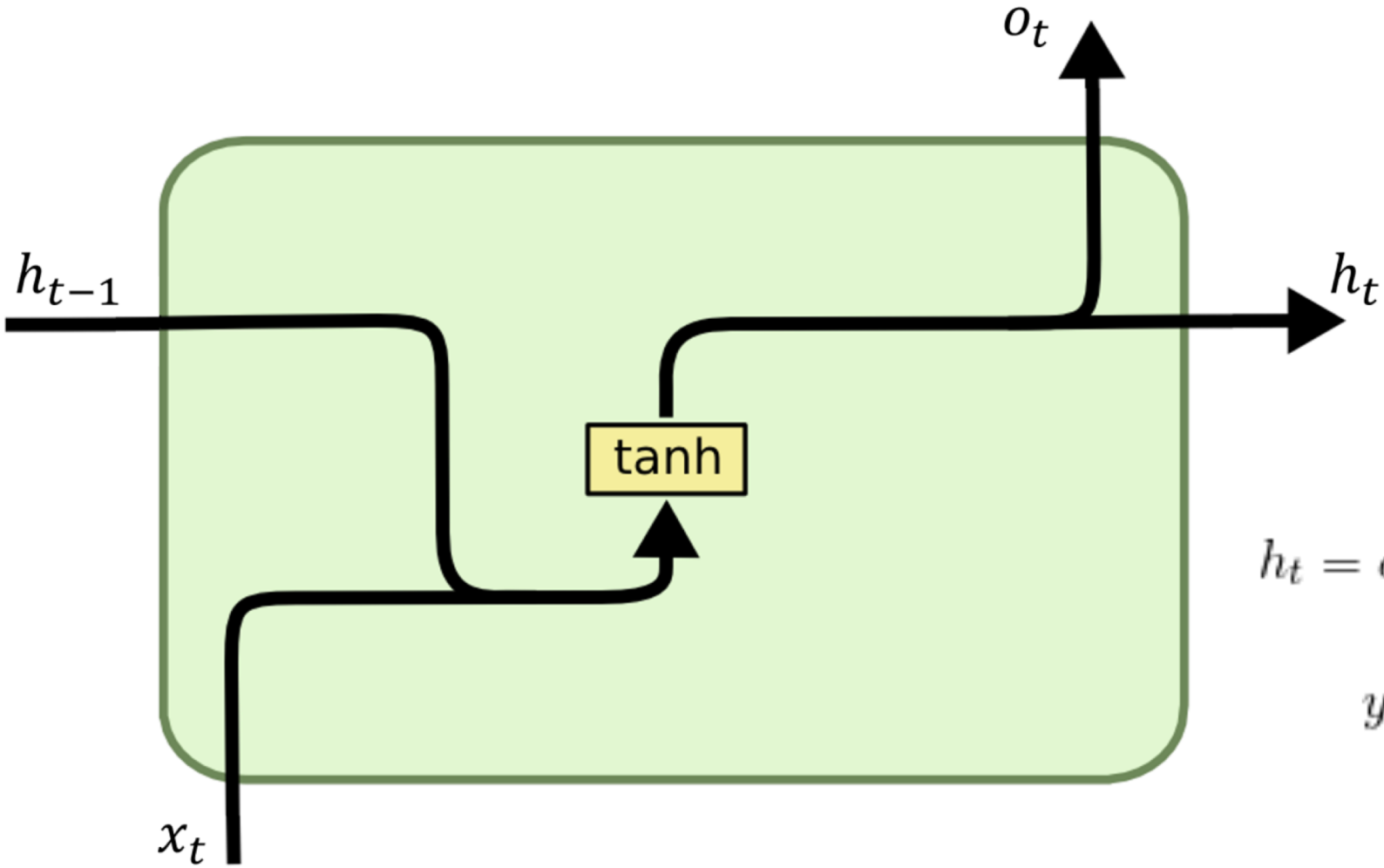
Difference between RNN layers

- Simple RNN can maintain only short-term memory (state). Therefore, if your signals (e.g. speech, text strings, consecutive snapshots) are long, its output at the end of signal has little to do with the early part of signal.

- Guide on RNN layers in tf.keras: https://www.tensorflow.org/guide/keras/rnn

Reference: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

SimpleRNN

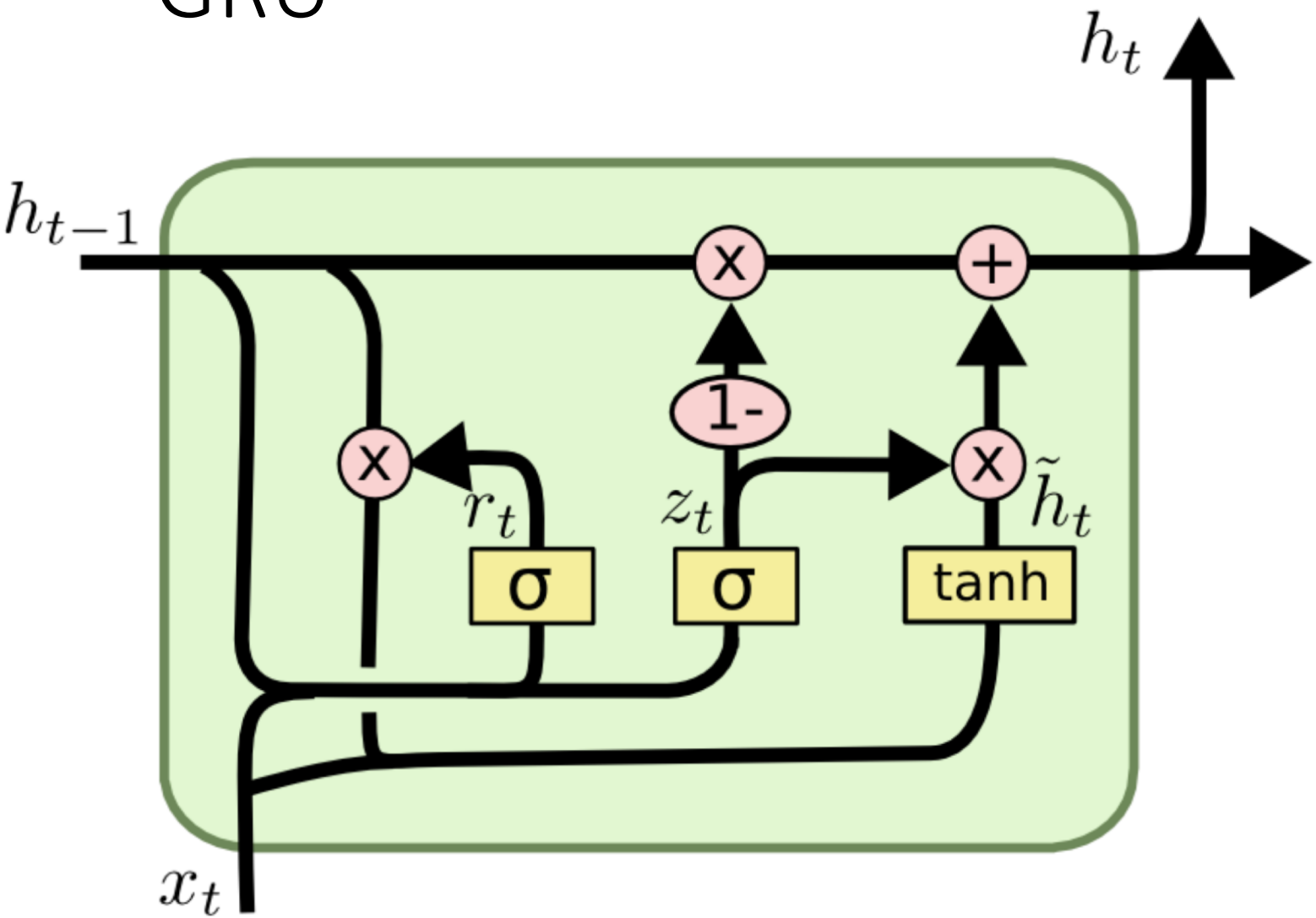


$$h_t = \sigma_h(i_t) = \sigma_h(U_h x_t + V_h h_{t-1} + b_h)$$

$$y_t = \sigma_y(a_t) = \sigma_y(W_y h_t + b_h)$$

Reference: http://dprogrammer.org/rnn-lstm-gru

GRU



Update gate: controls new info

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

Reset gate: controls old state

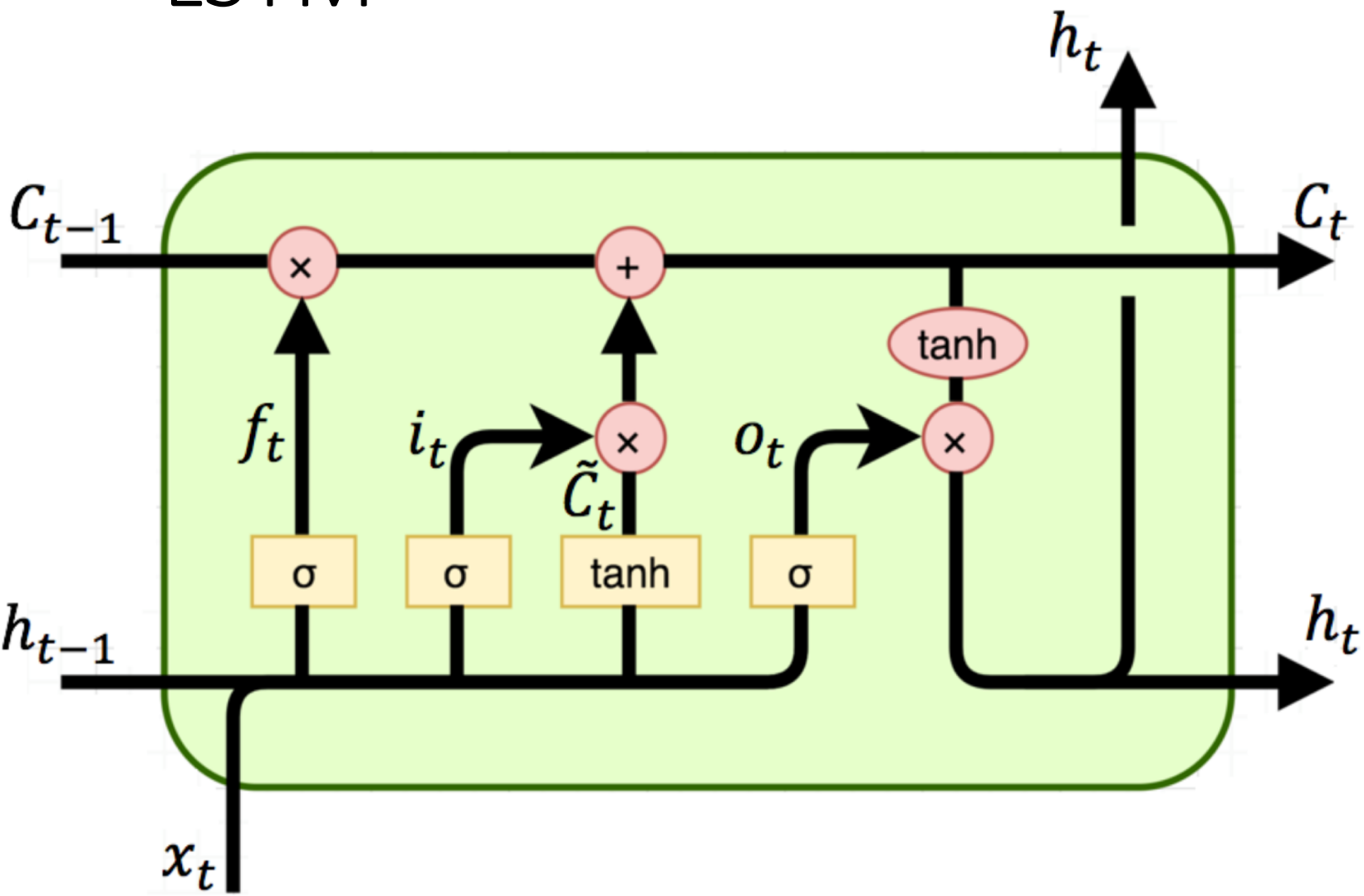
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$= \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

$$u_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Reference: http://dprogrammer.org/rnn-lstm-gru

LSTM



forget gate

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

input gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

output gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

cell state

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

hidden state (output)

$$h_t = o_t \odot \tanh(C_t)$$

Reference: http://dprogrammer.org/rnn-lstm-gru

Notes

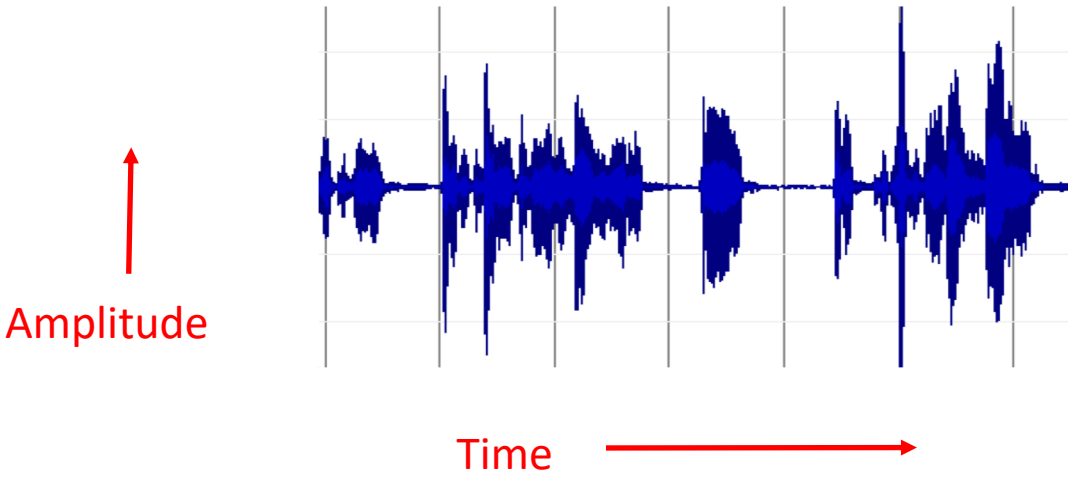
- Gate contains sigmoid activation. It acts like a control where 0 means to forget and 1 means to remember.
- GRU has one state while LSTM has two states.
- GRU has two gates while LSTM has three gates.

- For more, check the guide and `tf.keras.layers.SimpleRNN`, `tf.keras.layers.GRU`, `tf.keras.layers.LSTM`

Language Classification (HW5)

- Background – audio/speech signal processing
- Mel-frequency cepstrum coefficients (MFCCs)
- Build a RNN for spoken language classification

Audio Signal Processing

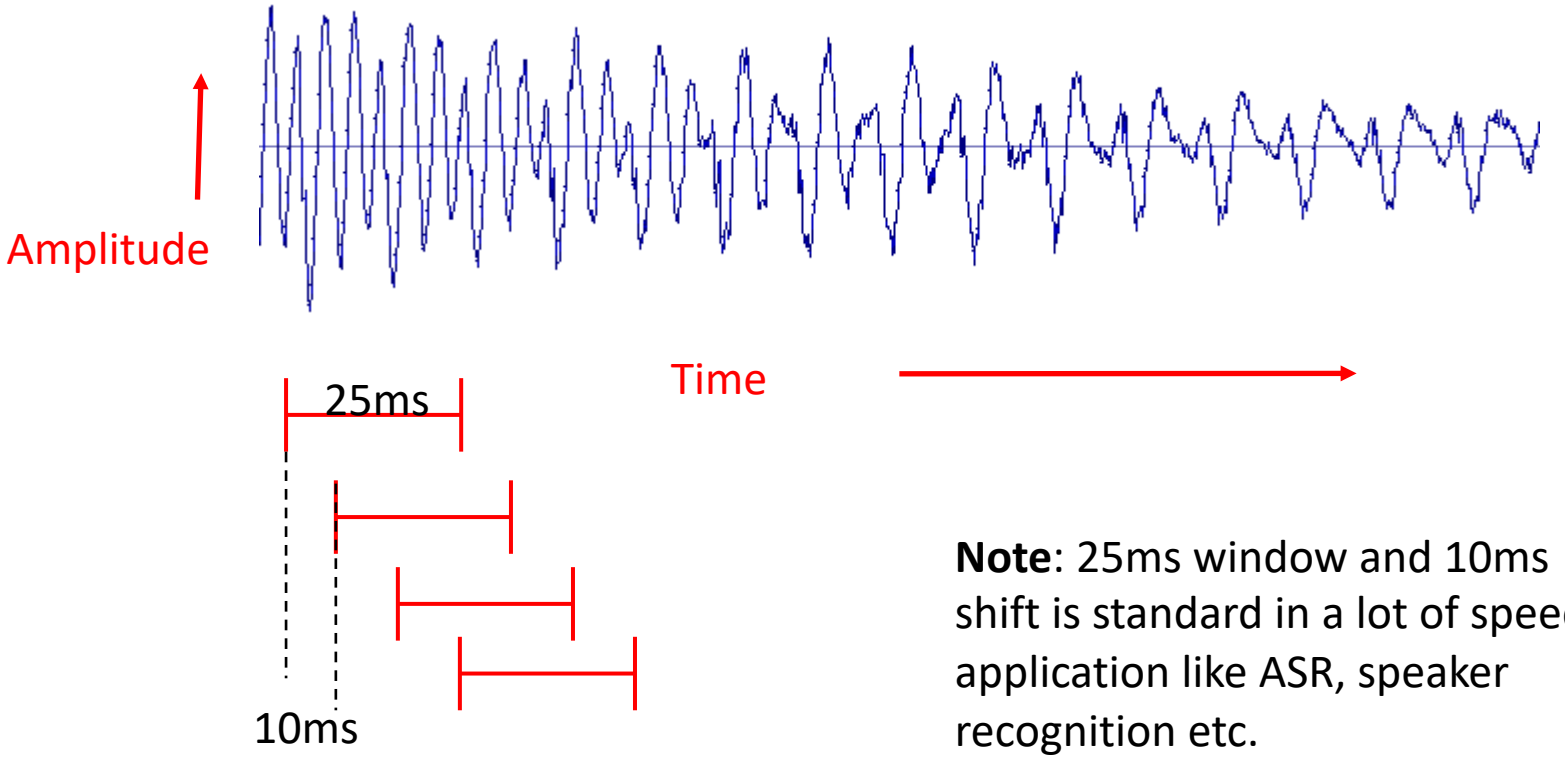


- We want to extract components in frequency domain, but audio signal is not stationary.

- Assume that the audio signal is stationary in a small window, we perform short-time Fourier transform on it.

Contributed by Arindam Jati

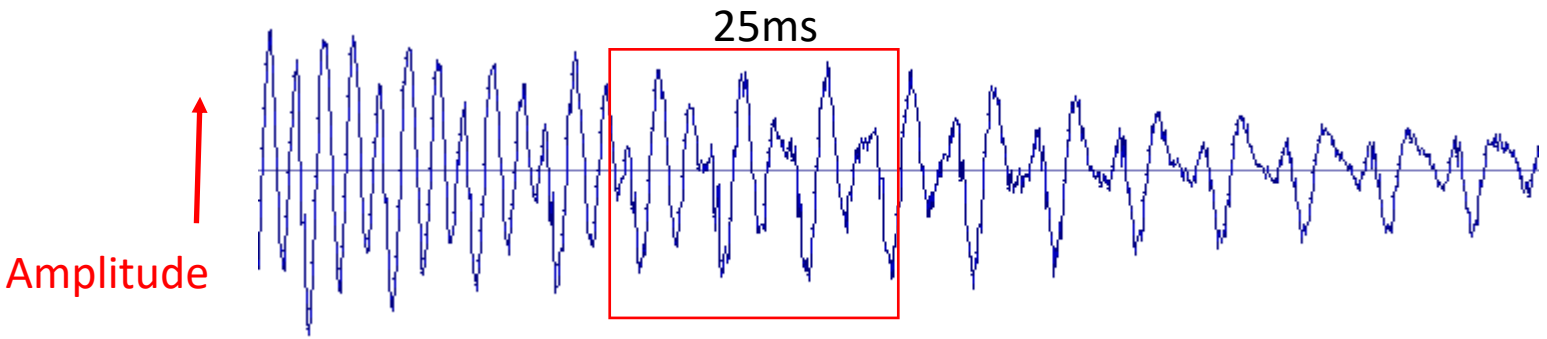
Short-time Analysis

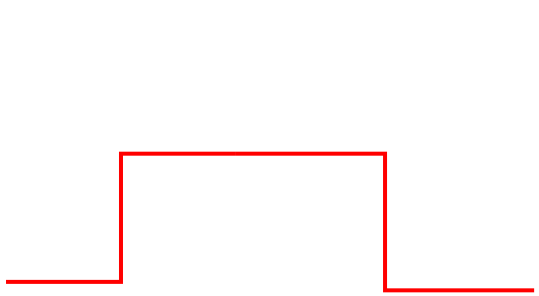


Note: 25ms window and 10ms shift is standard in a lot of speech application like ASR, speaker recognition etc.

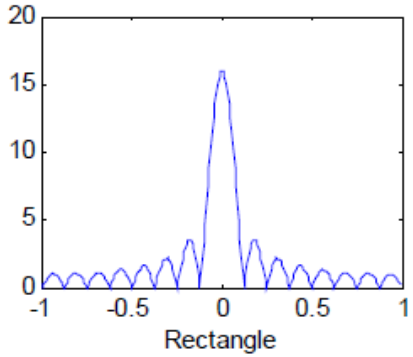
Contributed by Arindam Jati

Windowing





DTFT

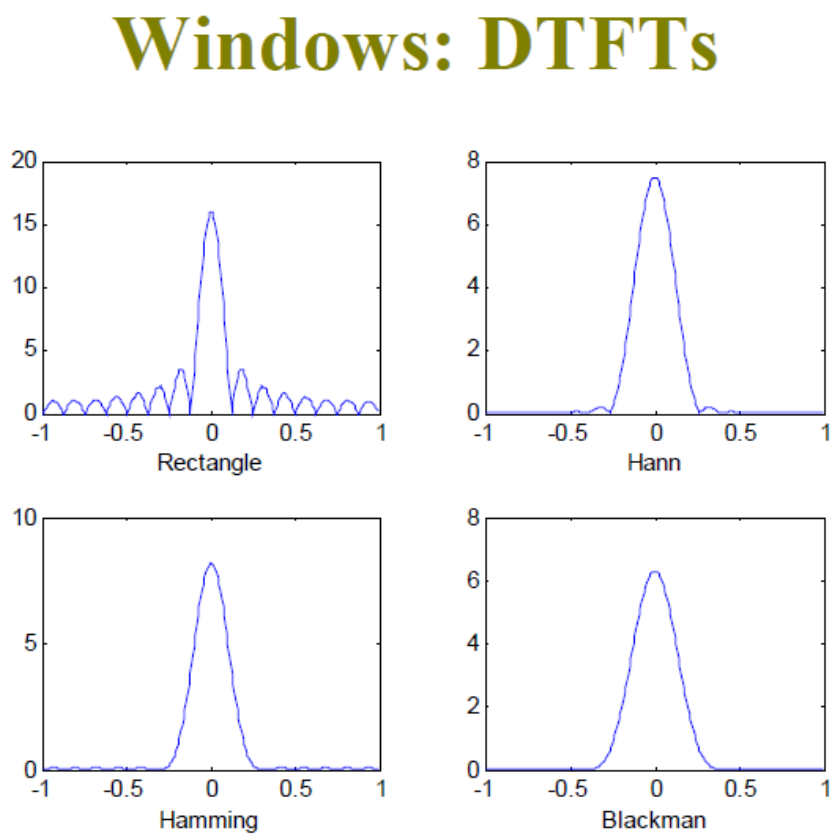


Higher sidelobe leakage

Contributed by Arindam Jati

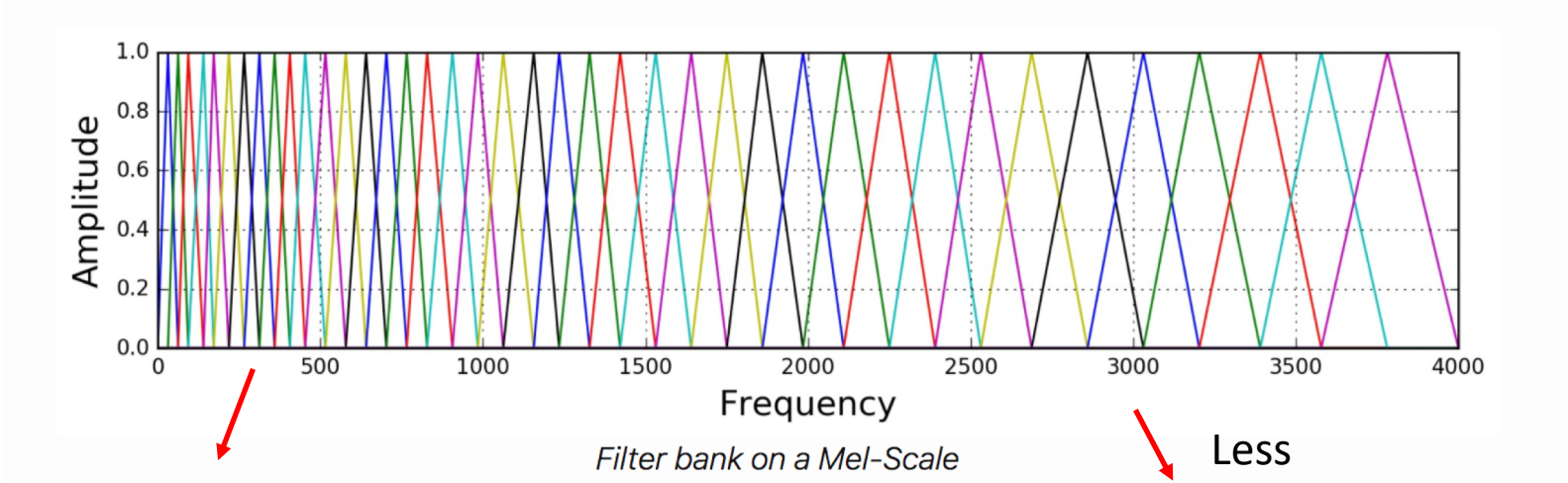
Windowing

- May want to choose other window function before FFT, e.g. Hann, Hamming, or Blackman window for small side-lobe energy.

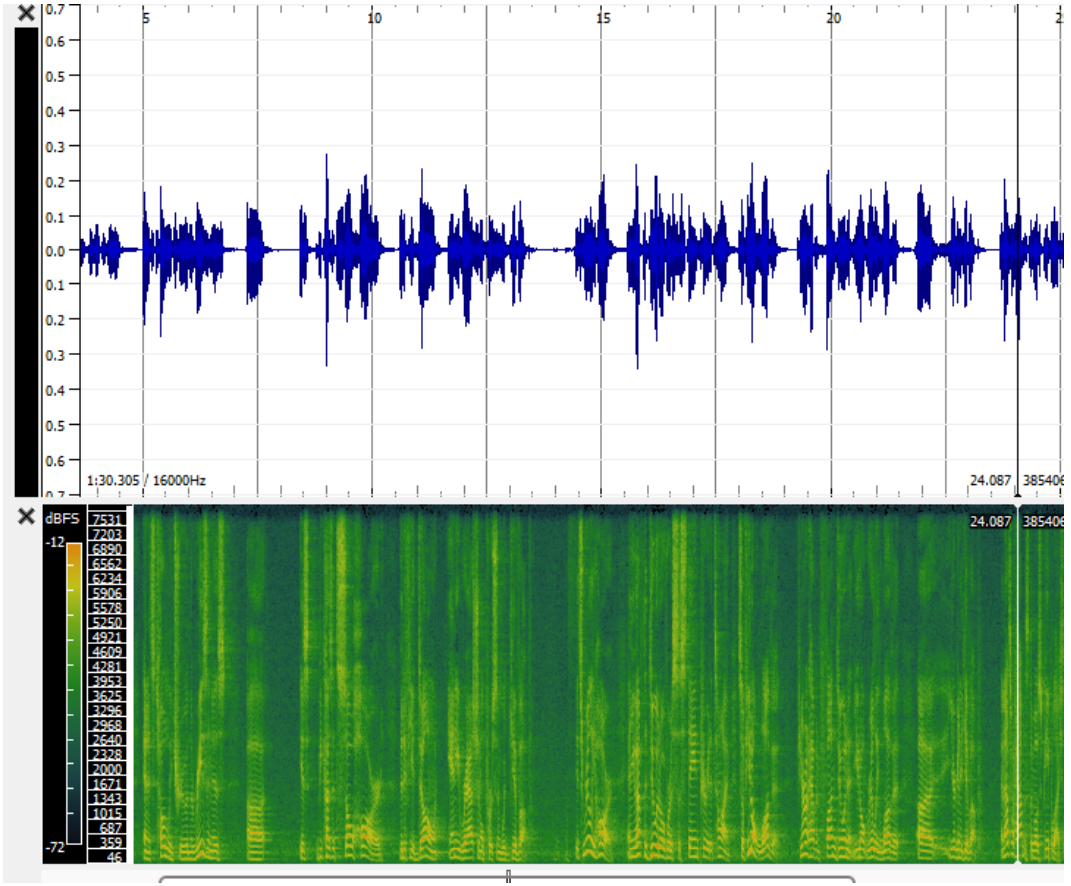


Mel Filter Banks

- Imitates human perception of speech
- Humans cannot properly distinguish two closely spaced frequencies, it becomes more noticeable as the frequency increases.

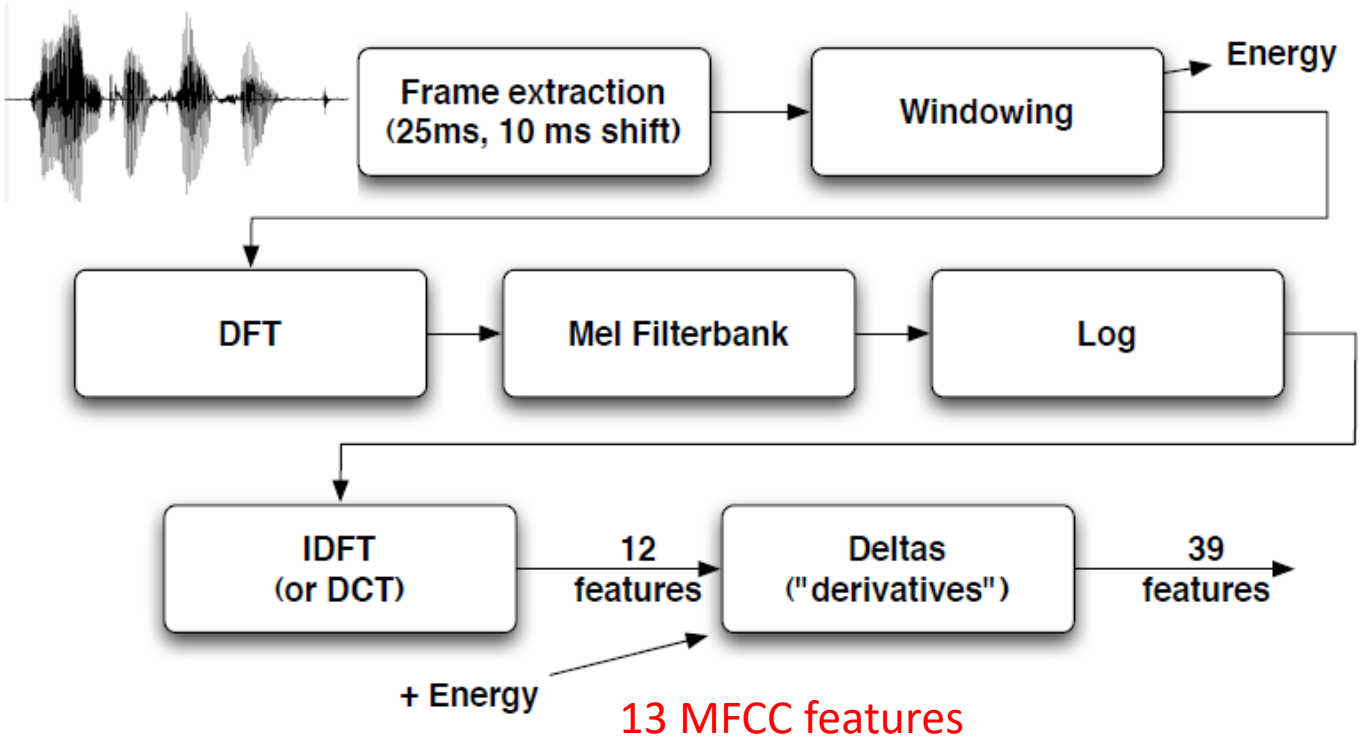


Spectrogram



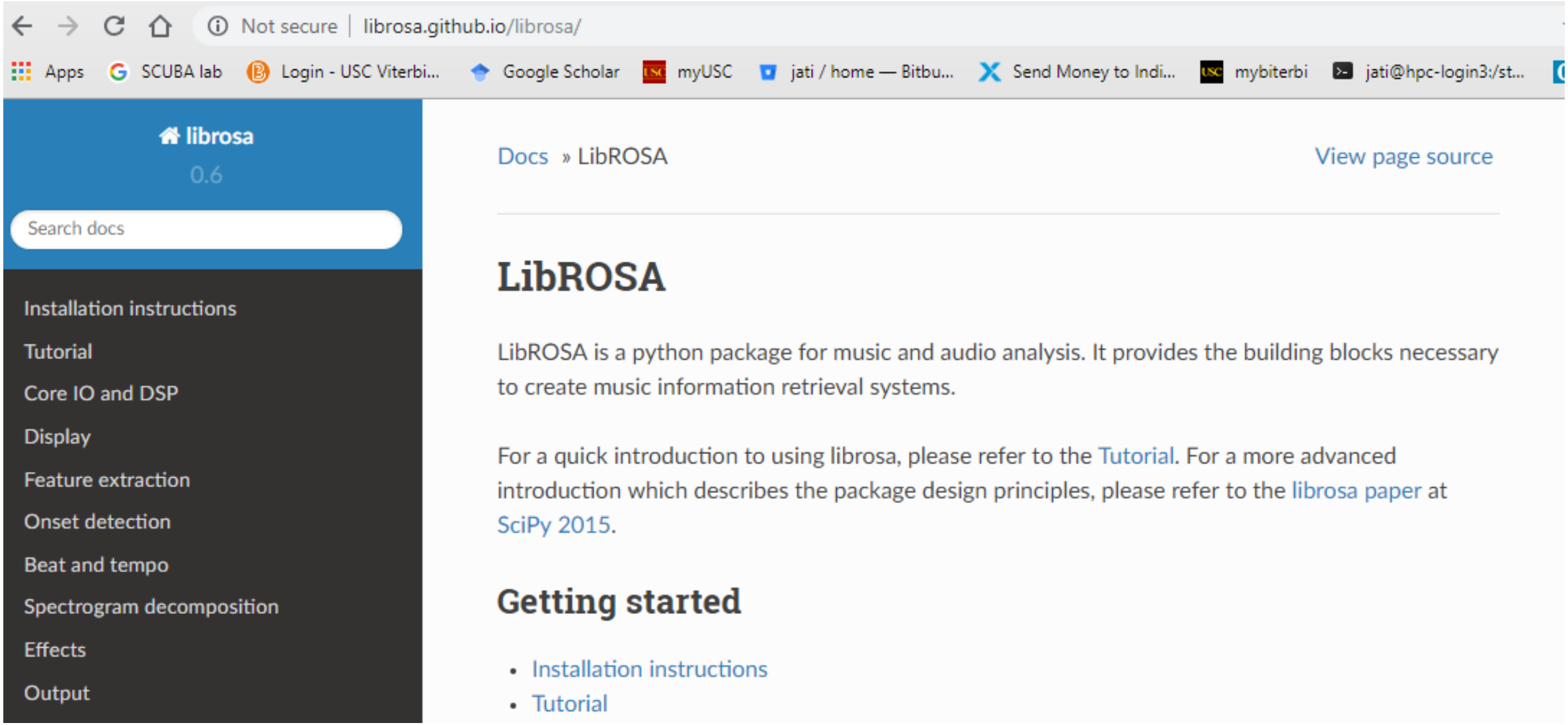
Contributed by Arindam Jati

From Raw Audio to MFCCs



- Reference: https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd

LibROSA


Not secure | librosa.github.io/librosa/
Apps
SCUBA lab
Login - USC Viterbi...
Google Scholar
myUSC
jati / home — Bitbu...
Send Money to Indi...
mybiterbi
jati@hpc-login3:/st...
librosa
0.6
Search docs
Installation instructions
Tutorial
Core IO and DSP
Display
Feature extraction
Onset detection
Beat and tempo
Spectrogram decomposition
Effects
Output
Docs » LibROSA
View page source
LibROSA
LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.
For a quick introduction to using librosa, please refer to the Tutorial. For a more advanced introduction which describes the package design principles, please refer to the librosa paper at SciPy 2015.
Getting started
Installation instructions
Tutorial

Using LibROSA

You can utilize **Librosa** (`https://librosa.github.io/librosa/index.html`) to extract 64 dimensional MFCC features for all utterances. A sample code snippet is provided below:

```
import librosa
y, sr = librosa.load('audio.wav',sr=16000)
#sr should return 16000, y returns the samples
mat = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=64, n_fft=int(sr*0.025), hop_length=
    int(sr*0.010))
print(y.shape, sr, mat.shape)
```

- This is configured for 25 m-sec frames and 10 m-sec skip.

LibROSA: Useful functions

- librosa.core.load
- librosa.core.to_mono
- librosa.core.stft
- librosa.feature.melspectrogram
- librosa.feature.mfcc

- ➢Load audio
- ➢Converts to single channel audio
- ➢STFT
- ➢Gives mel filterbank features
- ➢Gives MFCC features

and many more …