Research Center for Technology and Art

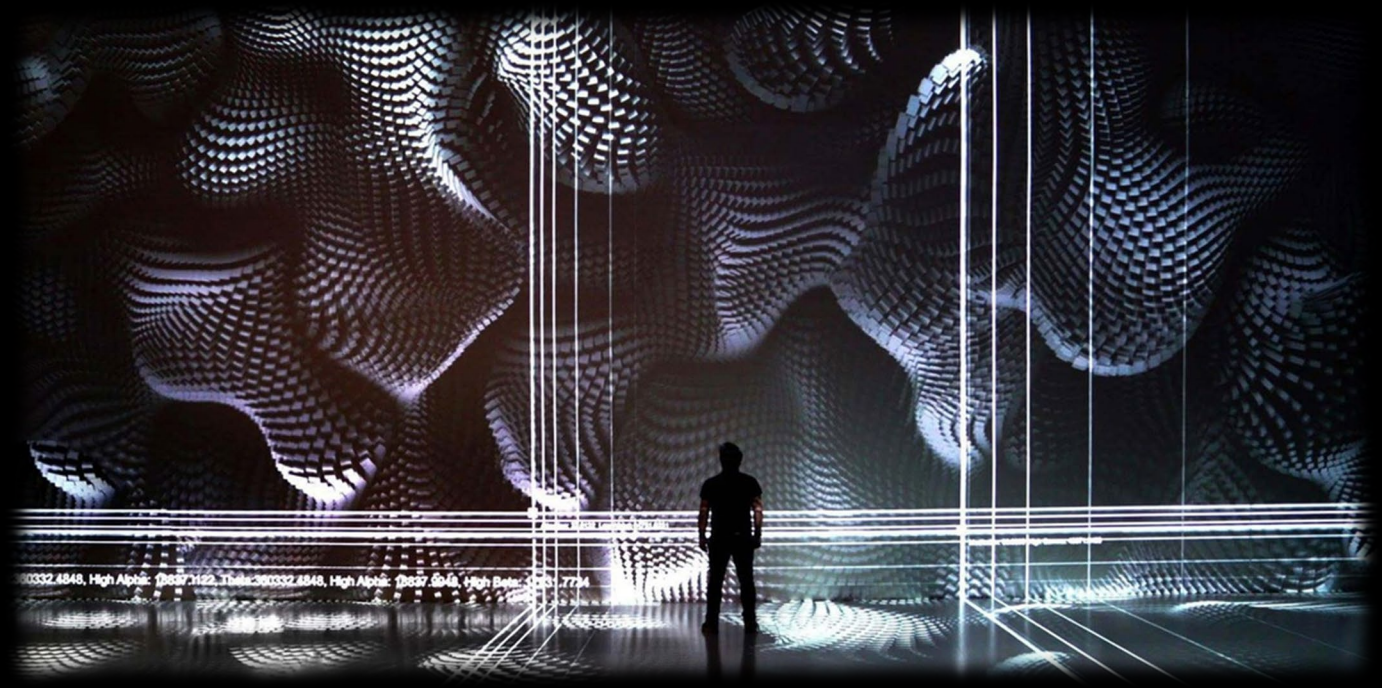
Seminar Presentation

AI
Artificial Intelligence

ARS Electronica
Festival 2019

Digital Music & Sound Art
- Voices from AI in Experimental Improvisation

[Award of Distinction]



https://soundcloud.com/tomomibot

Digital Music & Sound Art

- The prizes in the categories “Interactive Art”, “Digital Communities”, “Digital Musics & Sound Art” and “Artificial Intelligence & Life Art” are not awarded annually but alternately every two years
- The “Digital Musics & Sound Art” winners present their works in the form of concerts conceived as part of the programme of Ars Electronica Nightlines.

Authors

Tomomi Adachi is a performer/composer, sound poet, instrument builder and visual artist. Known for his versatile style, he has performed his voice and electronics pieces, sound poetry, improvised music and contemporary music in site-specific compositions, classical ensembles and groups of untrained musicians.

Andreas Dzialocha is an artist and developer. His work consists of digital and physical environments, spaces, festivals or software for participants and listeners. The computer itself serves as an artistic, political, social or philosophical medium.

Marcello Lussana combines music, philosophy and technology, particularly the interaction between music and human movement, where body and computer are connected through a complex understanding of body perception and interfaces.

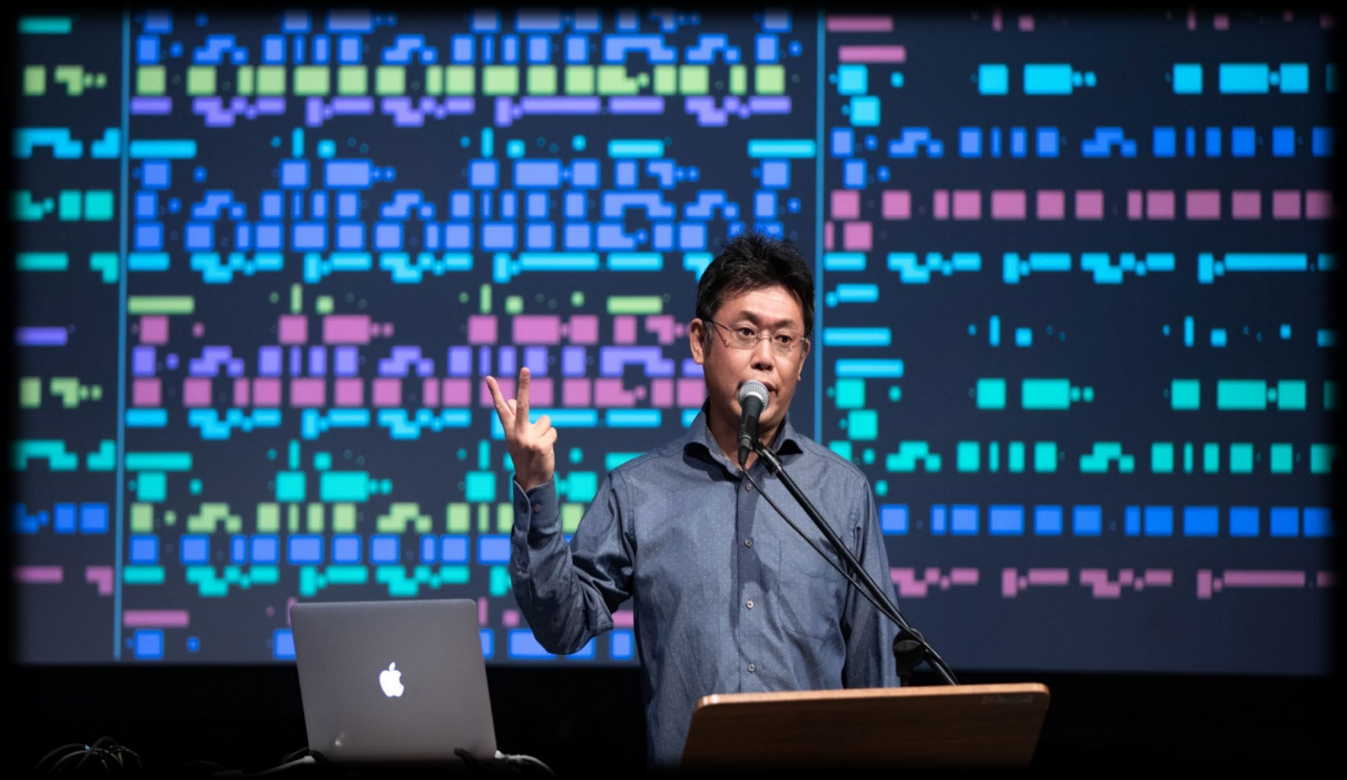
Project Introduction

Voices from AI in Experimental Improvisation is an attempt to improvise and interact with a computer software which “learns” about the performer’s voice and musical behavior. The program named “ tomomibot ” is based on artificial intelligence (AI) algorithms and enables a voice performer, Tomomi Adachi (human), to perform with his AI learning independently over time from his past performances.

Artistic direction, performance:
Tomomi Adachi

AI programming, concept:
Andreas Dzialocha

Programming, concept:
Marcello Lussana



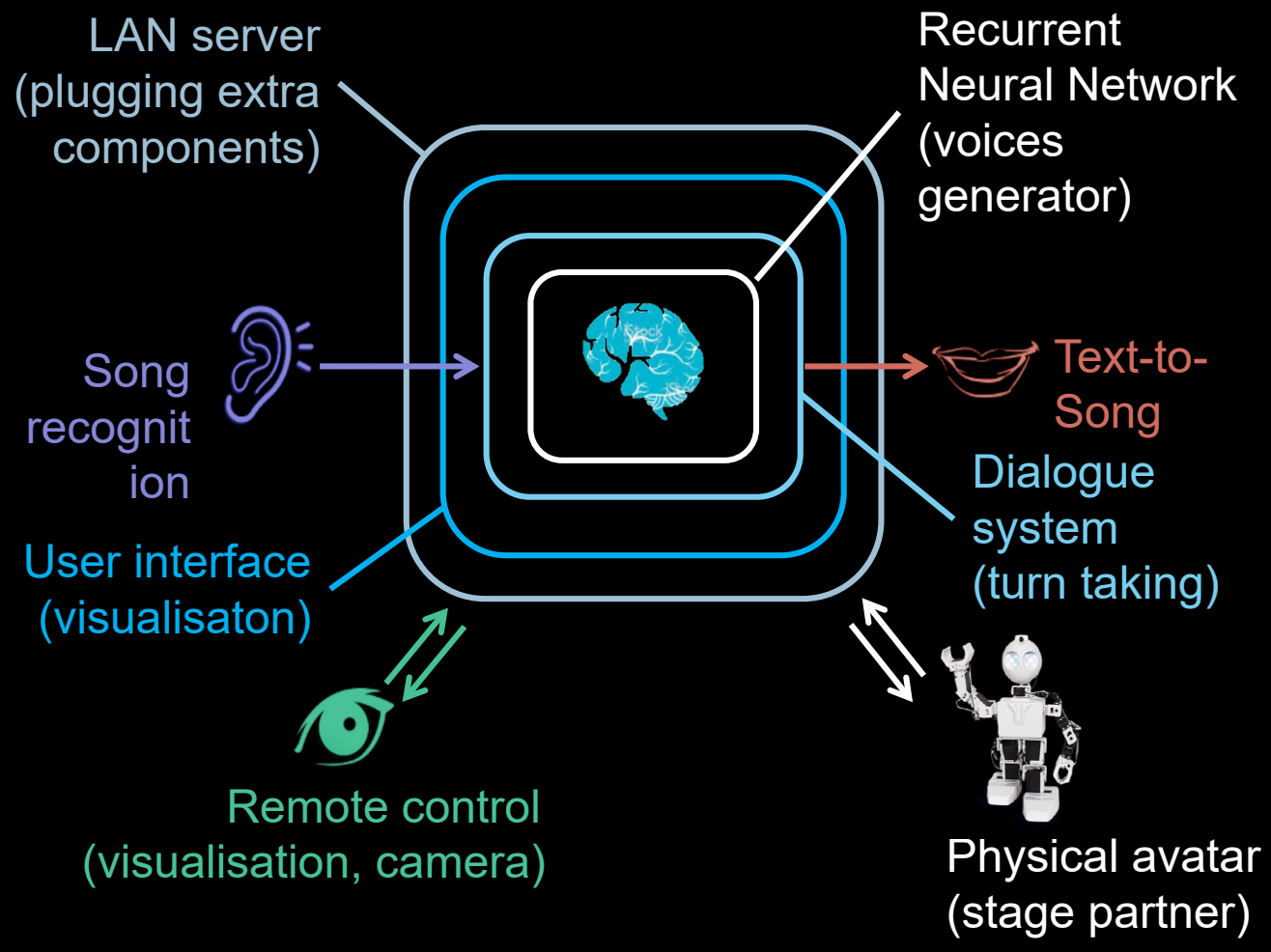
Background

Tomomibot tries it out by interacting in real time with the sound artist Tomomi Adachi, using deep learning techniques. The Artificial Intelligence (AI) “learns” the artist’s individual style via voice recordings and directly confronts him with the newly generated material. Their joint performance shows how interactive technology and AI can influence a (vocal) style.

The work was exhibited at Neue Musik Berlin e.V. in 2018.

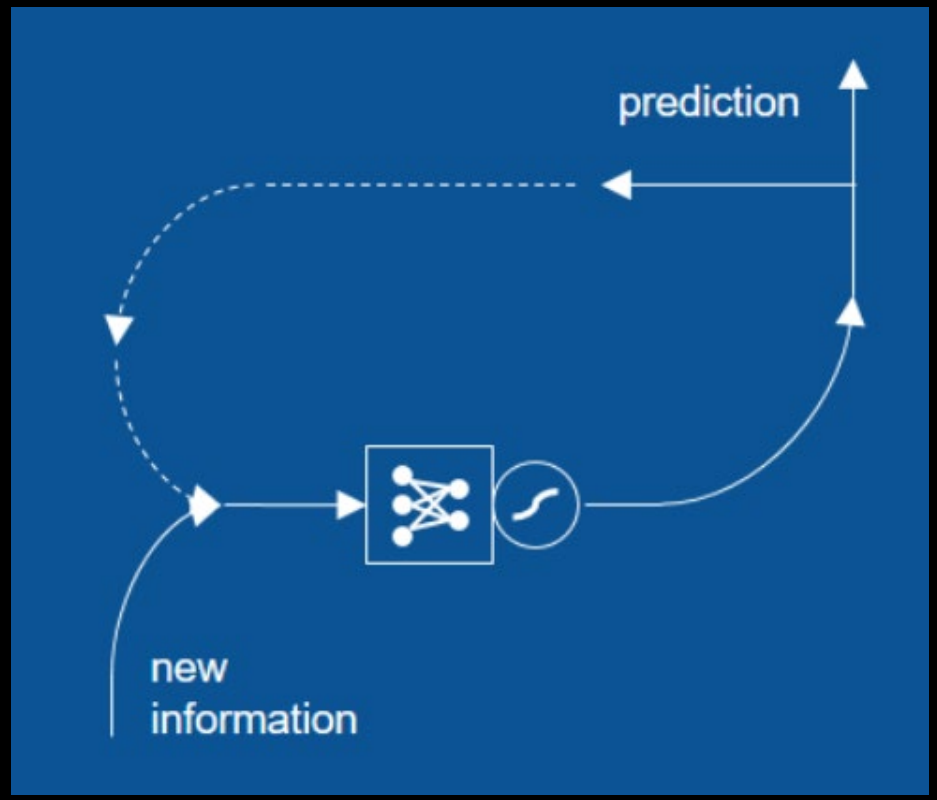
Tomomibot

Tomomibot is a software based on a sequential neural network algorithm, LSTM (Long short-term Memory), a form of sequential neural networks, deciding on which sound to play next, based on which live sounds it heard before. The software was designed and developed by Andreas Dzialocha. Experimenting with AI sound synthesis algorithms (WaveNet, WaveRNN, FFTNet) the developer Marcello Lussana generated a large database of sounds which sound like Tomomi. These sounds serve as the sound vocabulary “tomomibot” uses to improvise with human Tomomi.

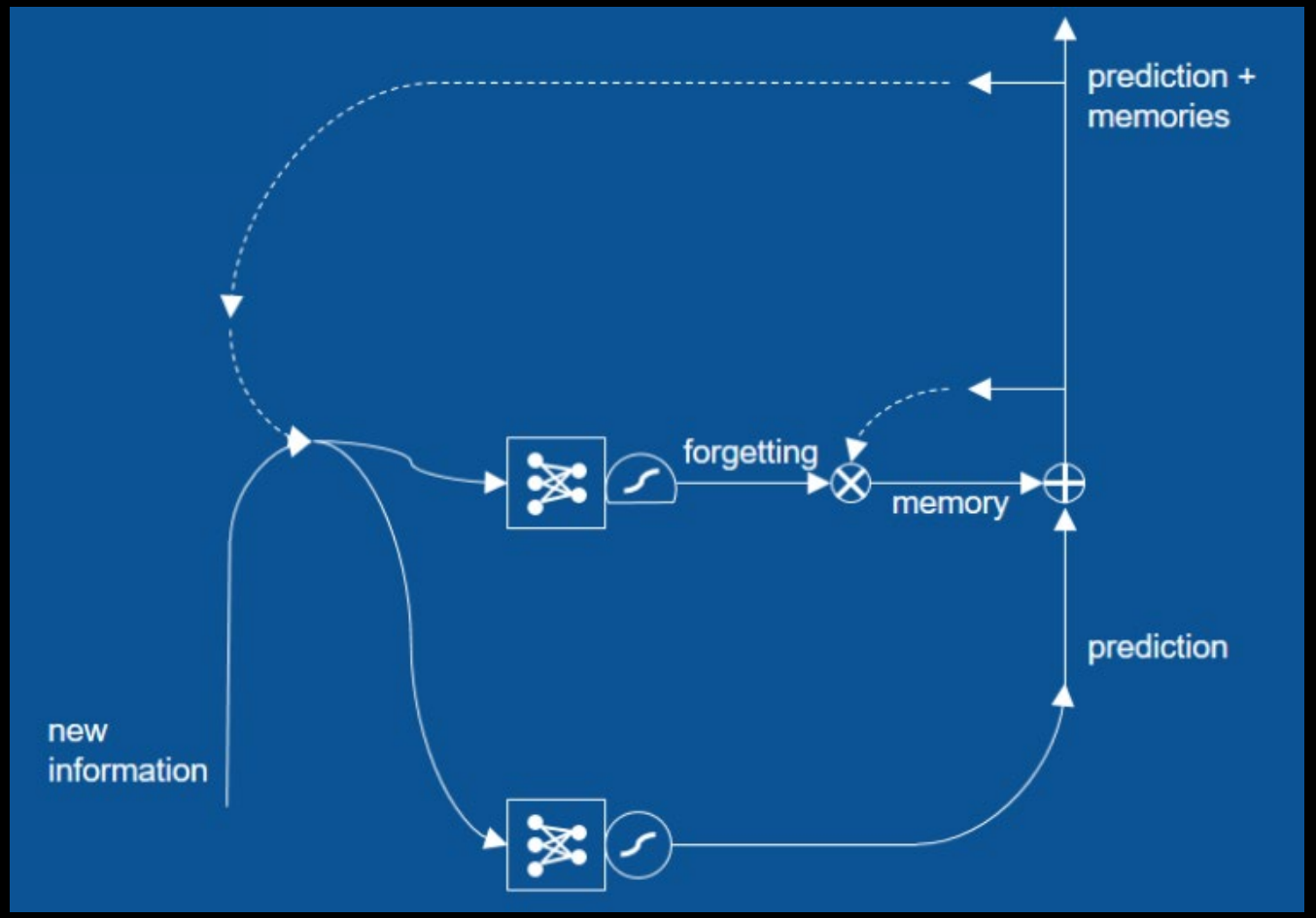


Sequential Neural Network

Recurrent Neural Network (RNN)



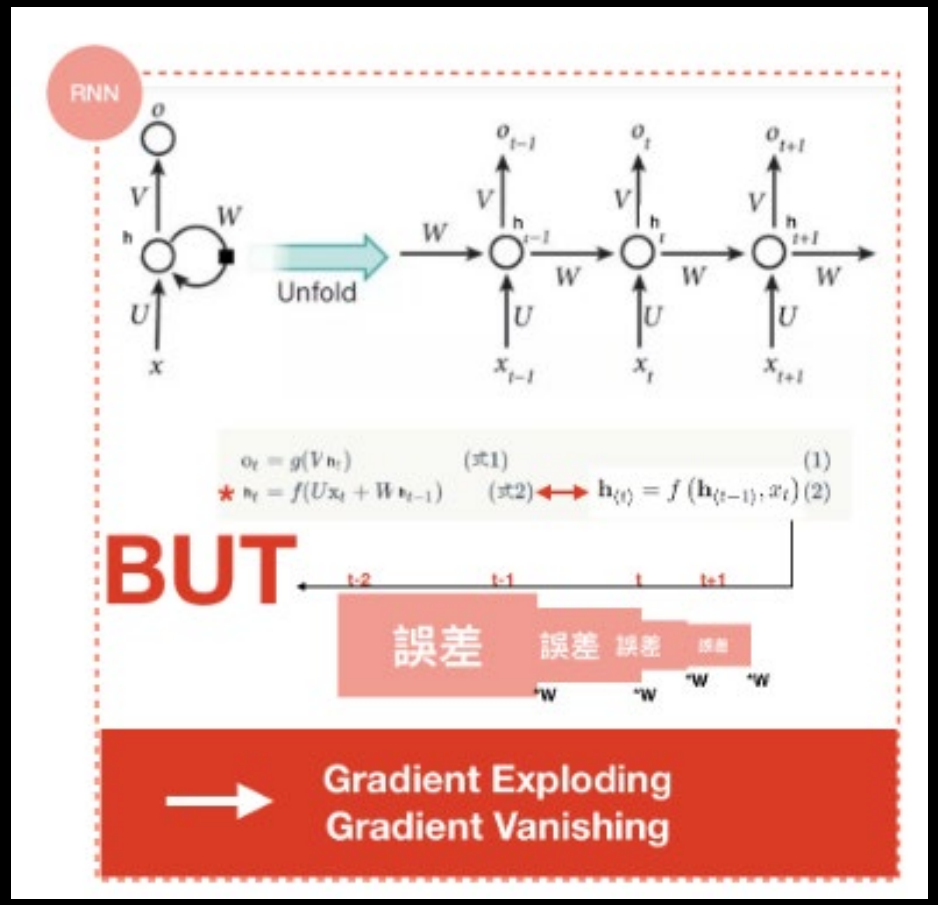
Long Short-Term Memory Network (LSTM)



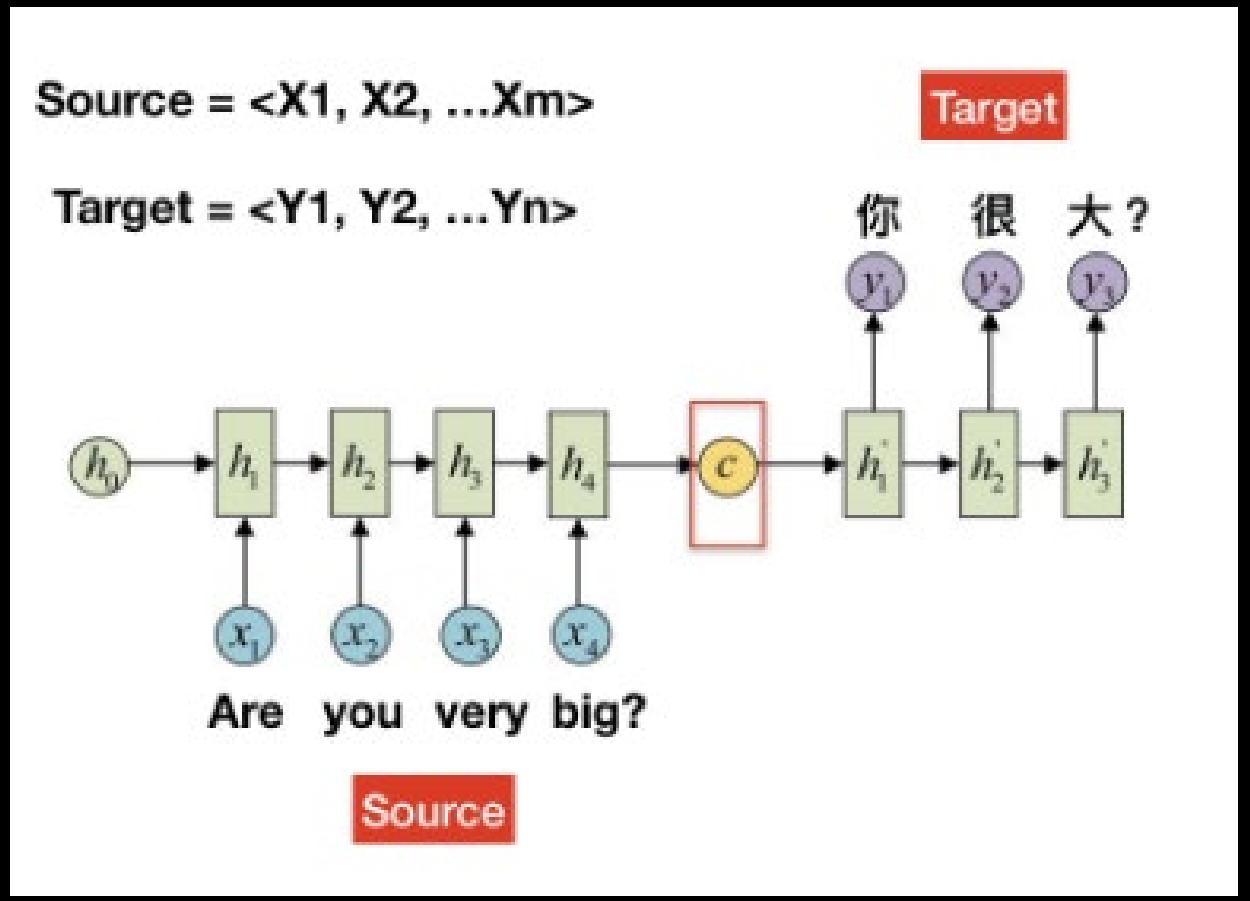
Reference: https://towardsdatascience.com/how-to-implement-seq2seq-lstm-model-in-keras-shortcutnlp-6f355f3e5639

Sequential Neural Network

Recurrent Neural Network (RNN)

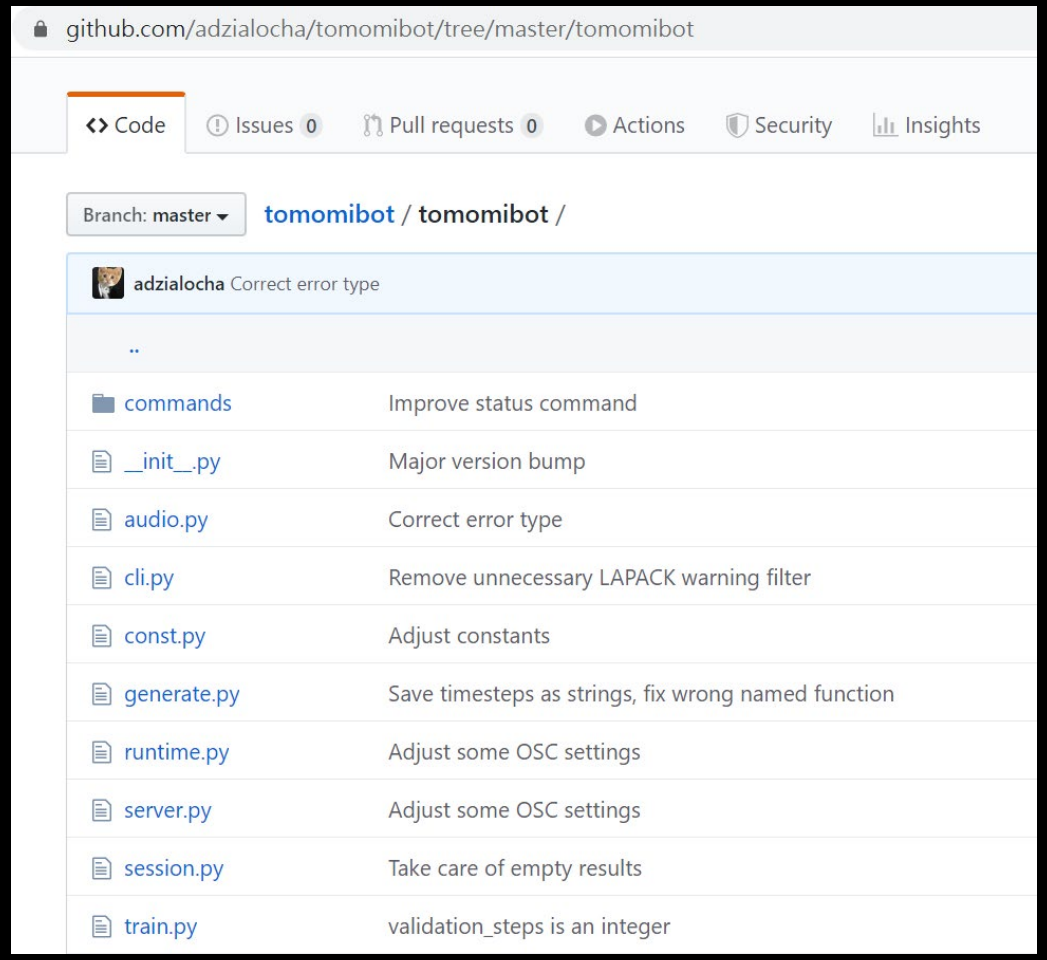


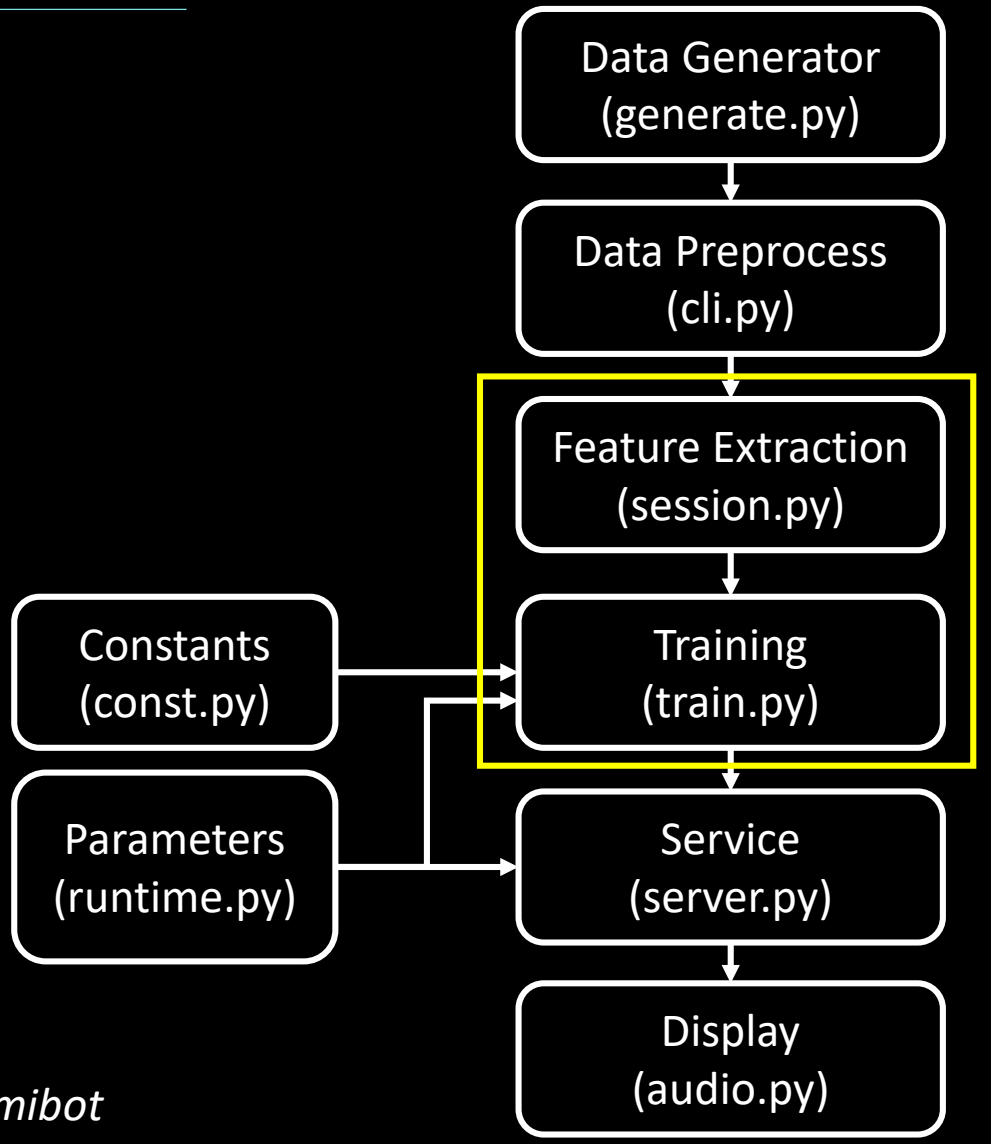
Long Short-Term Memory Network (LSTM)



Reference: https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-1-%E4%B8%AD%E6%96%87%E7%89%88-2714bbd92727

Architecture of tomomibot





https://github.com/adzialocha/tomomibot/tree/master/tomomibot

Feature Extraction

```
# Analyze and categorize slices
for y in slices:
    y_slice = y[0]

    # Calculate MFCCs
    try:
        mfcc = mfcc_features(y_slice, self.samplerate)
    except RuntimeError:
        self.ctx.vlog(
            'Not enough sample data for MFCC analysis')
    else:
        # Calculate RMS
        rms_data = librosa.feature.rms(y=y_slice) / self._voice.rms_max
        rms = np.float32(np.max(rms_data)).item()
```

MFCC (Mel-Frequency Cepstral Coefficients)

Step1: Pre-emphasis

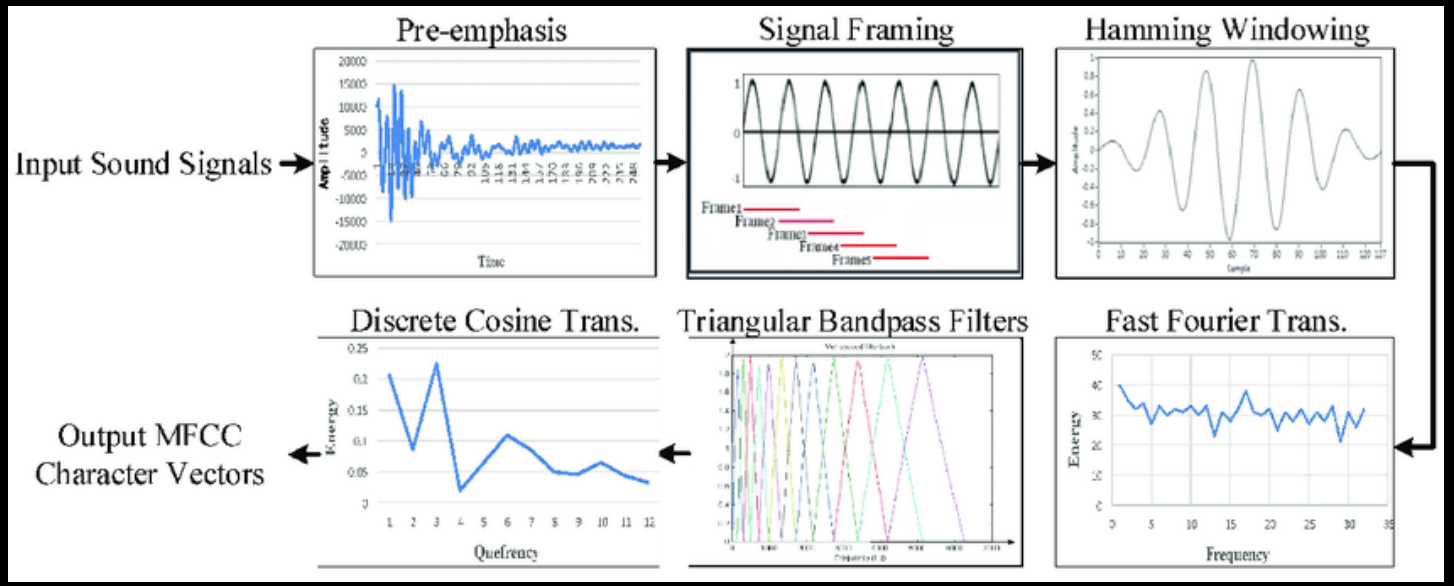
s2(n) = s(n) - a*s(n-1)
a: 0~1

Step2: Signal Framing

N: 256 or 512 (set of sample)
M: N/3 (set of overlap sample)

Step3: Hamming Windowing

W(n, a) = (1 - a) - a cos(2pn/(N-1)),
0≦n≦N-1



Step6: Discrete Cosine Trans.

$C_m=\Sigma_{k=1}^{N}\cos[m*(k-0.5)*\pi/N]*E_k$,
m=1,2, ..., L

Step5: Triangular Bandpass Filters

mel(f)=2595*log10(1+f/700)

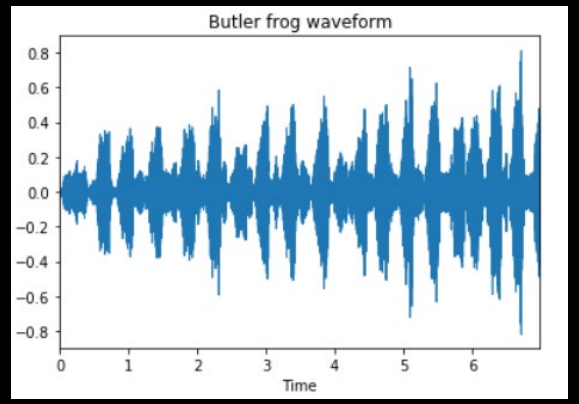
Step4: Fast Fourier Transform

$F(x) = \sum_{n-0}^{N-1} f(n) e^{-i2\pi(x\frac{n}{N})}$

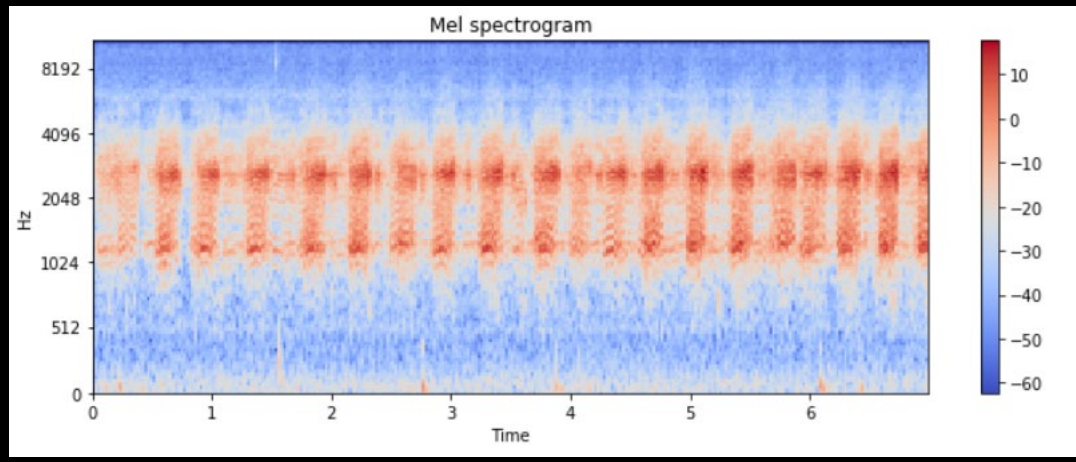
$f(n) = \frac{1}{N}\sum_{n-0}^{N-1} F(x) e^{i2\pi(x\frac{n}{N})}$

Feature Extraction

MFCC (Mel-Frequency Cepstral Coefficients) - Taiwan's Frog Sounds Classification



MFCC



Feature

Model	Initial Model	Model turning	Testing
SVM	0.8	0.94	0.99
Random Forst	0.6	0.90	0.95
XGBoost	0.75	0.85	0.75
LightGBM	0.85	0.85	0.85

Train

	0	1	2	3	4	5	6	7	8	9	...	200	201	202	
0	-185.626883	107.772851	-9.412268	46.641109	-31.252501	26.115870	0.743796	16.494876	12.095498	-9.613577	...	0.430407	51.993177	21.858606	7.6
1	-195.073173	99.788199	-11.642600	49.751983	-28.479792	22.598996	1.311489	12.465497	14.098283	-8.353313	...	0.632943	49.751168	19.857528	4.0
2	-195.435764	95.834447	-9.763945	50.097418	-31.189562	25.782336	-0.952654	14.067396	13.743476	-8.822381	...	0.482606	43.245856	20.145879	6.8
3	-351.180482	92.030936	3.365378	27.404095	1.362222	15.229861	-5.240146	26.568555	-8.659325	16.669641	...	1.358107	46.366270	16.790002	6.8
4	-335.960816	97.861751	-1.966739	19.760014	0.298002	16.324501	-10.484625	28.557586	-5.481404	15.082006	...	0.825509	38.168811	16.063708	6.6

5 rows × 210 columns

Reference: https://github.com/Yfyangd/frog

Training Model: LSTM

4 LSTM with 4 dropout function, 1 dense layer (256 to 99 classes)
Param count: 2,152,035

```
model = Sequential()
model.add(layers.Embedding(input_dim=num_classes,
                           output_dim=num_units,
                           input_length=seq_len))
for n in range(num_layers - 1):
    model.add(layers.LSTM(num_units, return_sequences=True))
    if dropout > 0.0:
        model.add(layers.Dropout(dropout))
model.add(layers.LSTM(num_units))
if dropout > 0.0:
    model.add(layers.Dropout(dropout))
model.add(layers.Dense(num_classes, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['acc'])

model.summary()
```

num_layers = 5
num_classes = 99

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, 3, 256)	25344
LSTM_1 (LSTM)	(None, 3, 256)	525312
dropout_6 (Dropout)	(None, 3, 256)	
LSTM_2 (LSTM)	(None, 3, 256)	525312
dropout_7 (Dropout)	(None, 3, 256)	
LSTM_3 (LSTM)	(None, 3, 256)	525312
dropout_8 (Dropout)	(None, 3, 256)	
LSTM_4 (LSTM)	(None, 3, 256)	525312
dropout_9 (Dropout)	(None, 3, 256)	
dense_1 (Dense)	(None, 3, 99)	25443
Total params: 2,152,035		2,152,035
Trainable params: 2,152,035		2,152,035
Non-trainable params: 0		0

Presentation/Demo

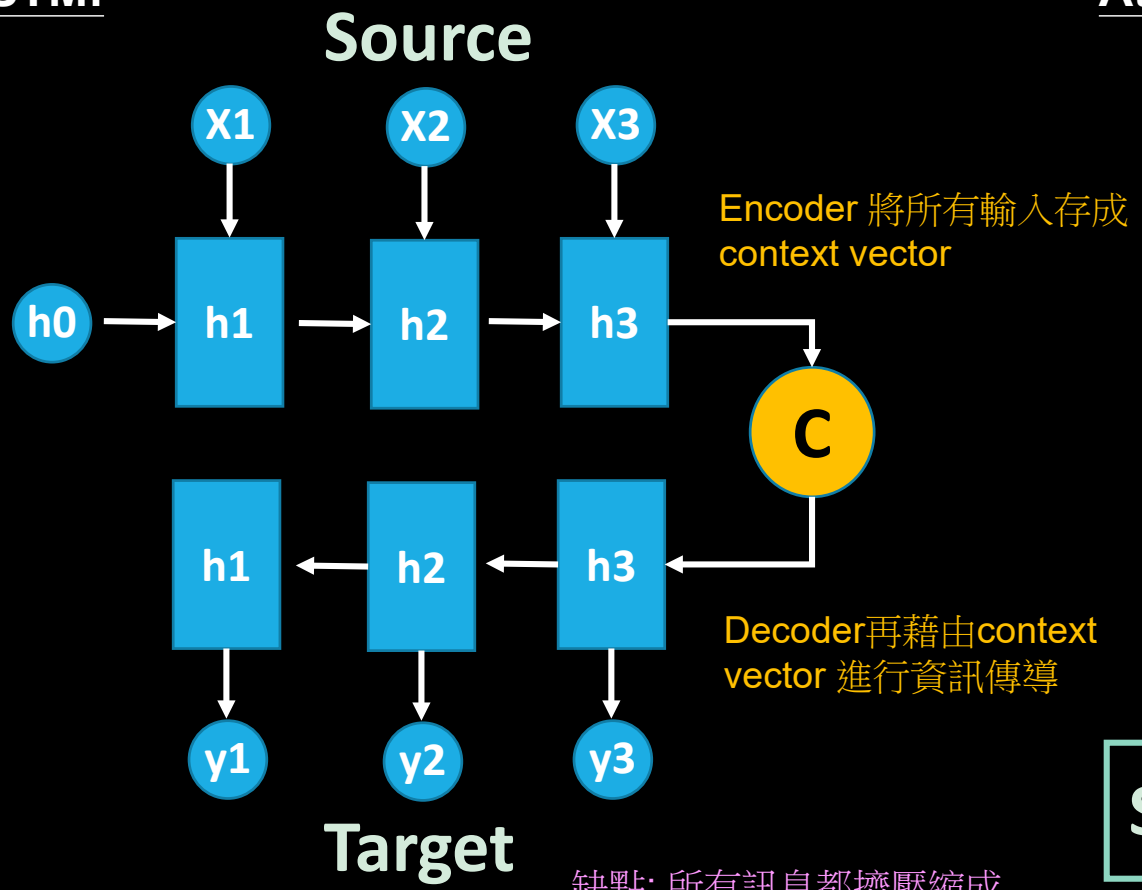
https://www.youtube.com/watch?v=8aKSWf-QpVs

Connection

- The project is not only a musical experiment with a non-human performer, but also an undertaking to make computer culture “audible.”
- The performance raises questions about the logic and politics of computers in relation to human culture.
- The Artificial Intelligence (AI) “learns” the artist’s individual style via voice recordings and directly confronts him with the newly generated material. Their joint performance shows how interactive technology and AI can influence a (vocal) style. However, this dialogue also makes clear that the artist will always be more creative and unpredictable than his mechanical counterpart.
- The state of the art method: Attention Mechanism

LSTM vs Attention Mechanism

LSTM:

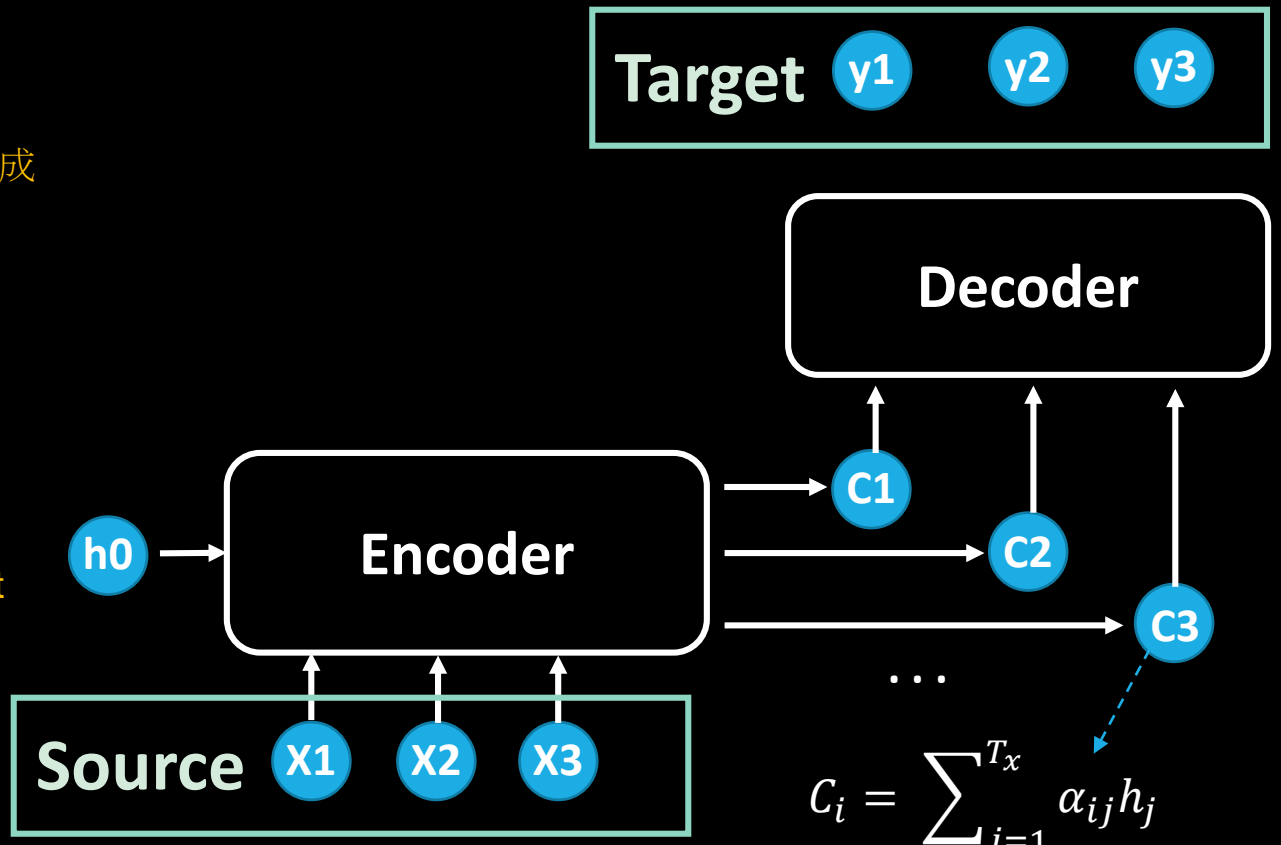


$$C = F(X_1 + X_2 + \ldots + X_m)$$

$$Y_i = G(C, Y_1, Y_2, \ldots, Y_{i-1})$$

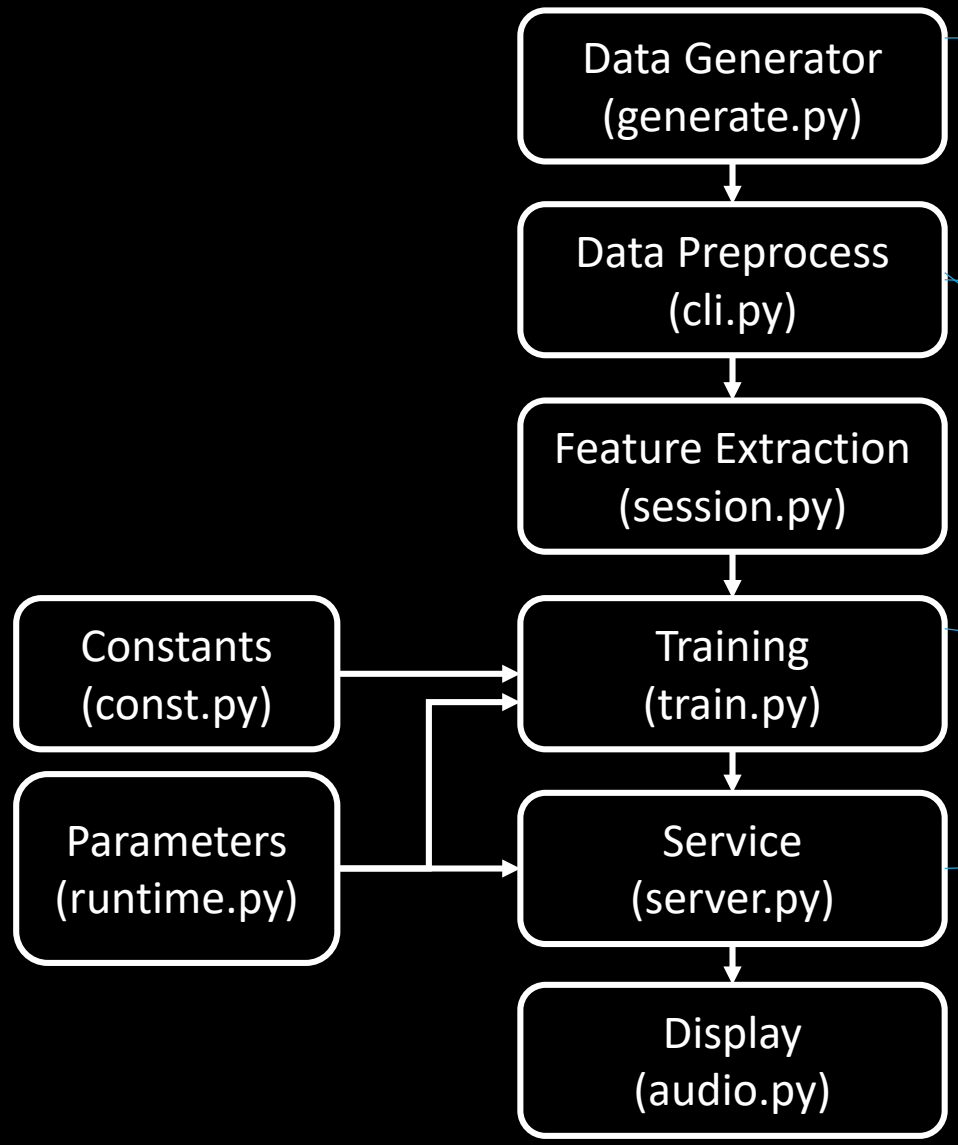
缺點: 所有訊息都擠壓縮成 one context vector, 造成:
(1) 無法表達序列訊息
(2) 句子太長會丟失訊息

Attention Mechanism:



Reference: Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473

Training Processes



Input: midi

```
stream = converter.parse('midi/jjj.mid')
print('parts:')
parts = instrument.partitionByInstrument(stream)
for i in parts:
    print(i)
```

```
parts:
<music21.stream.Part Flute>
<music21.stream.Part Piano>
<music21.stream.Part Electric Guitar>
<music21.stream.Part Organ>
<music21.stream.Part Electric Bass>
```

Encoding

```
note_to_int
```

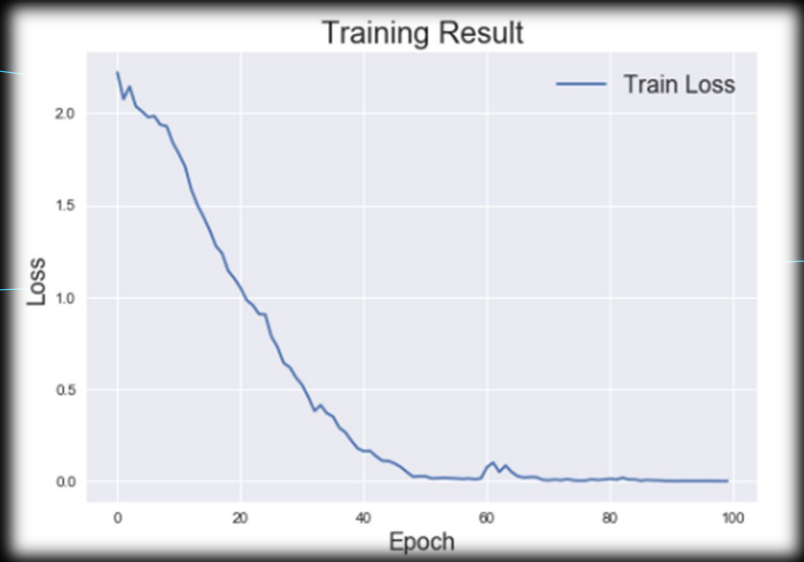
```
{'A4': 0,
 'B-4': 1,
 'B4': 2,
 'C#5': 3,
 'D5': 4,
 'E5': 5,
 'F#4': 6,
 'F#5': 7,
 'G4': 8,
 'G5': 9}
```

Data Set

```
sorted(set(item for item in notes))
```

```
['A4', 'B-4', 'B4', 'C#5', 'D5', 'E5', 'F#4', 'F#5', 'G4', 'G5']
```

Model Training



Predict: Data Generator

```
output_notes
```

```
<music21.note.Note A>,
<music21.note.Note A>,
<music21.chord.Chord D F A>,
<music21.chord.Chord C E>,
<music21.note.Note F>,
<music21.note.Note A>,
<music21.note.Note A>,
<music21.note.Note D>,
<music21.note.Note D>,
<music21.note.Note C>,
<music21.note.Note A>,
<music21.note.Note D>,
<music21.note.Note D>,
<music21.note.Note A>,
<music21.chord.Chord A C>,
<music21.note.Note D>,
```

Attention Mechanism Demo

Original Music:

Learning Result: