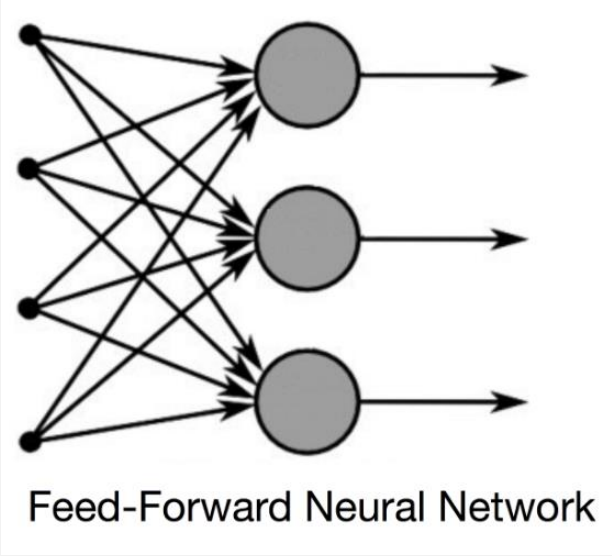
GENERATING MUSIC USING LSTM NEURAL NETWORKS

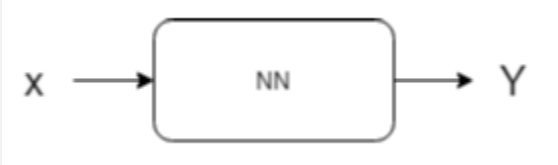
Author: Grzegorz Kazana

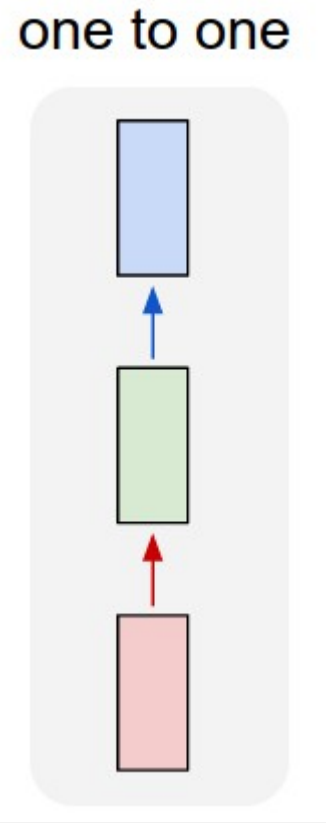
BIAI sem. 6

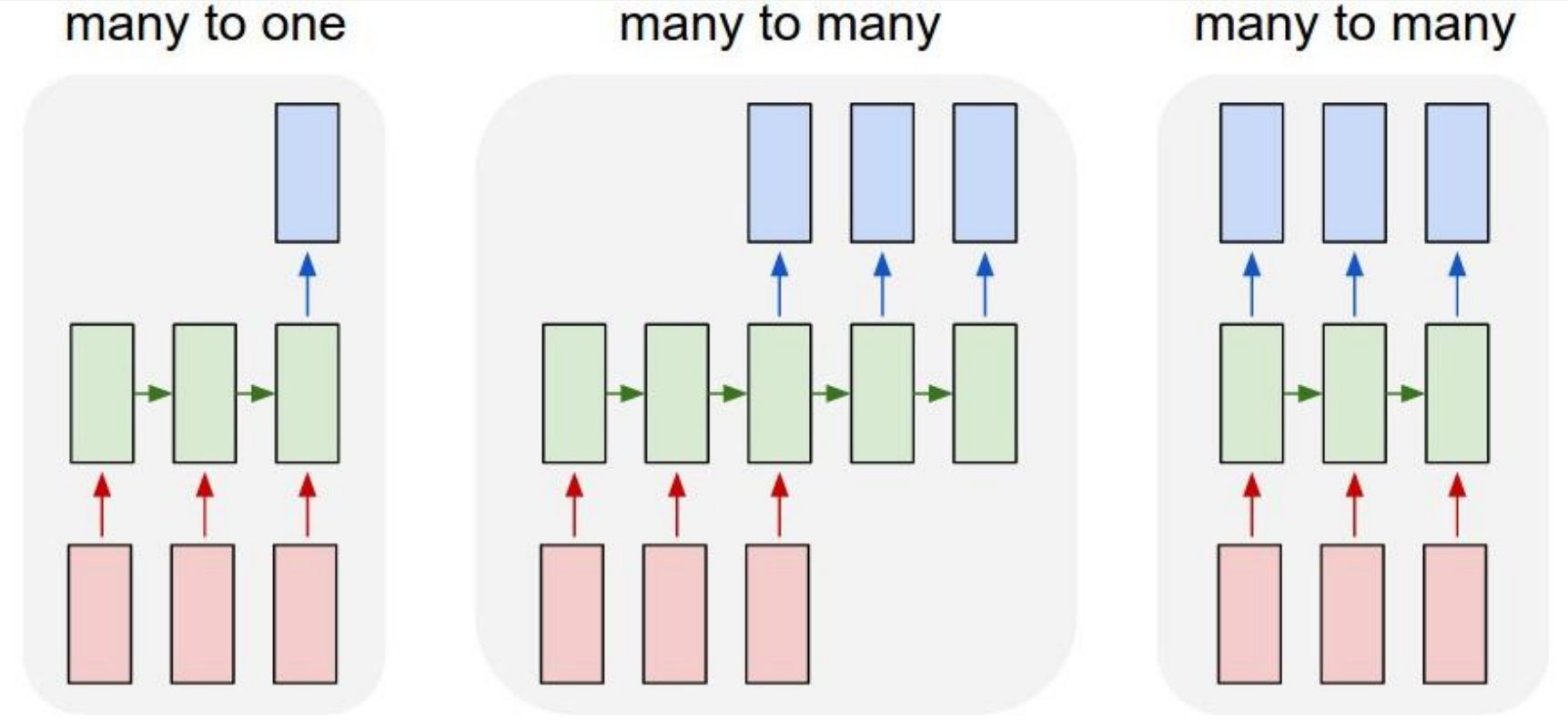
NEURAL NETWORKS AND SEQUENTIAL DATA



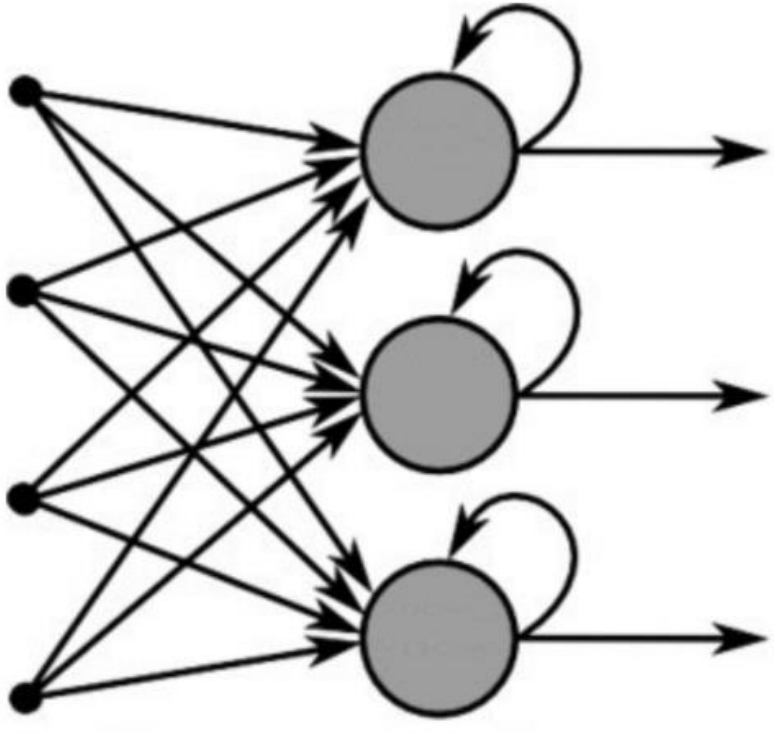
Feed-Forward Neural Network



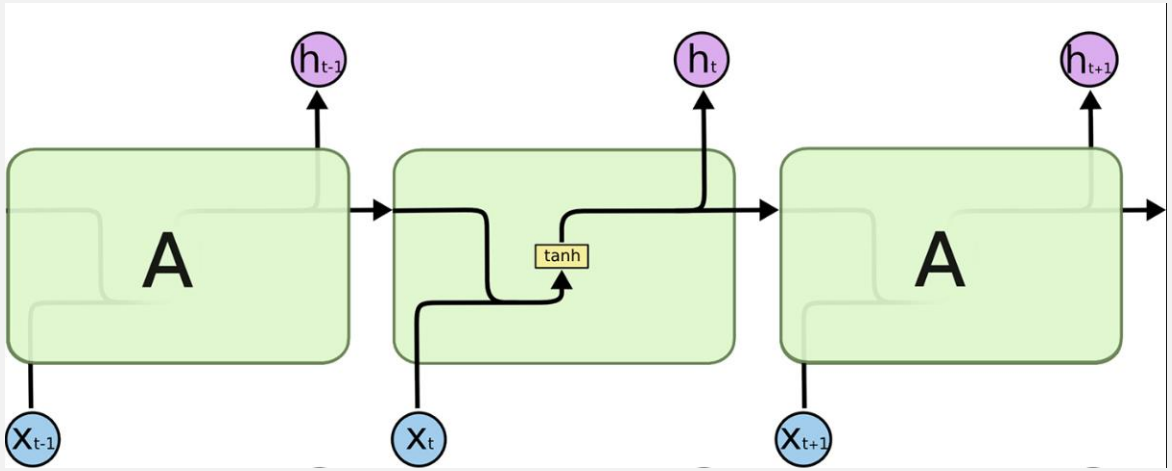



many to one
many to many
many to many

RNN

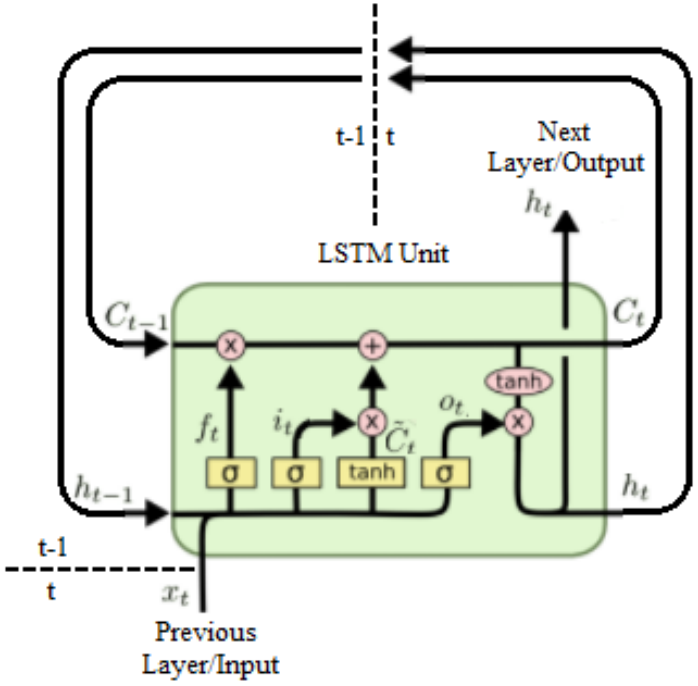


Recurrent Neural Network

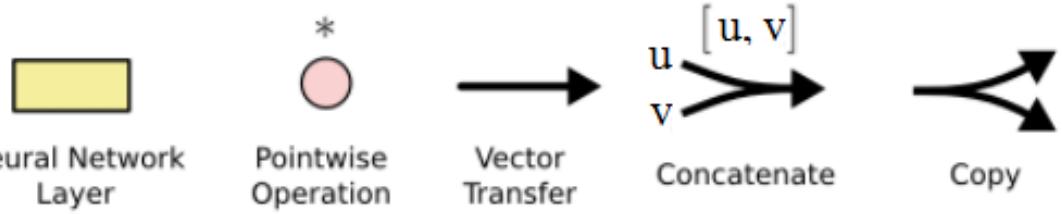


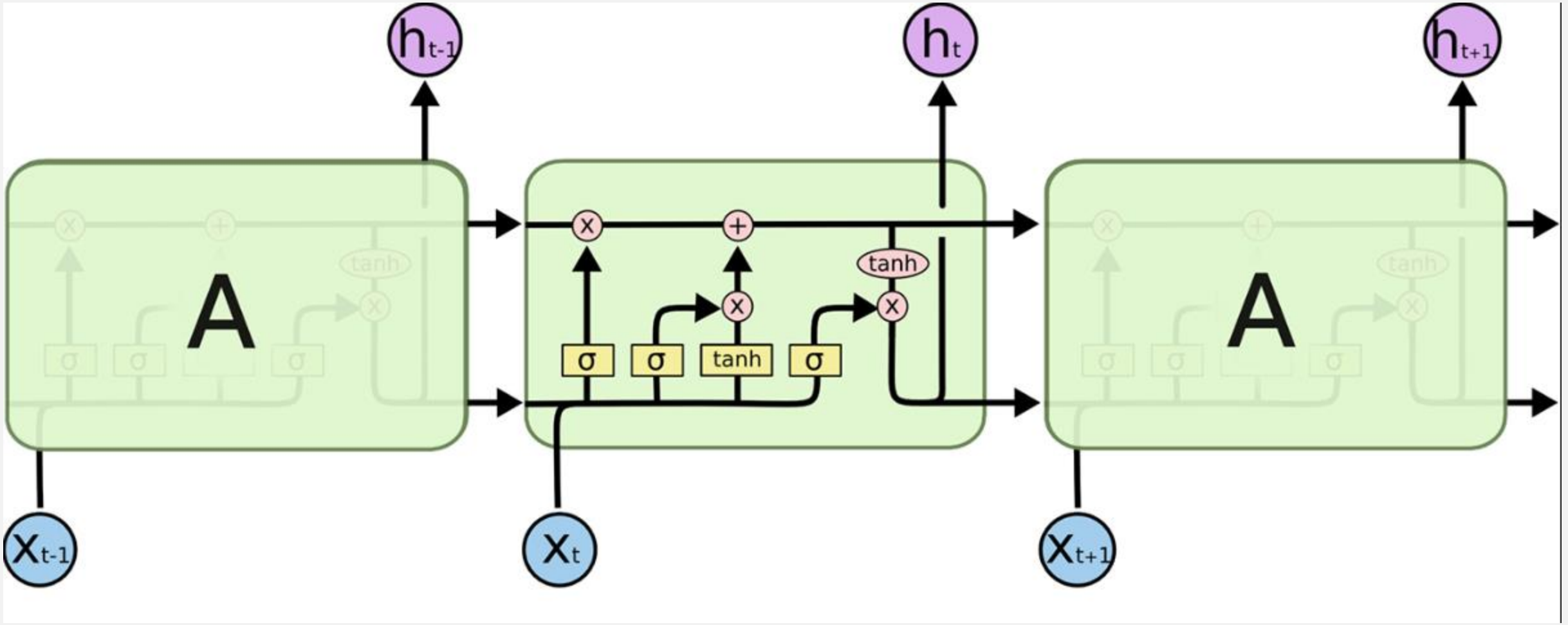
LONG-SHORT TERM MEMORY NN

Understanding LSTM Networks

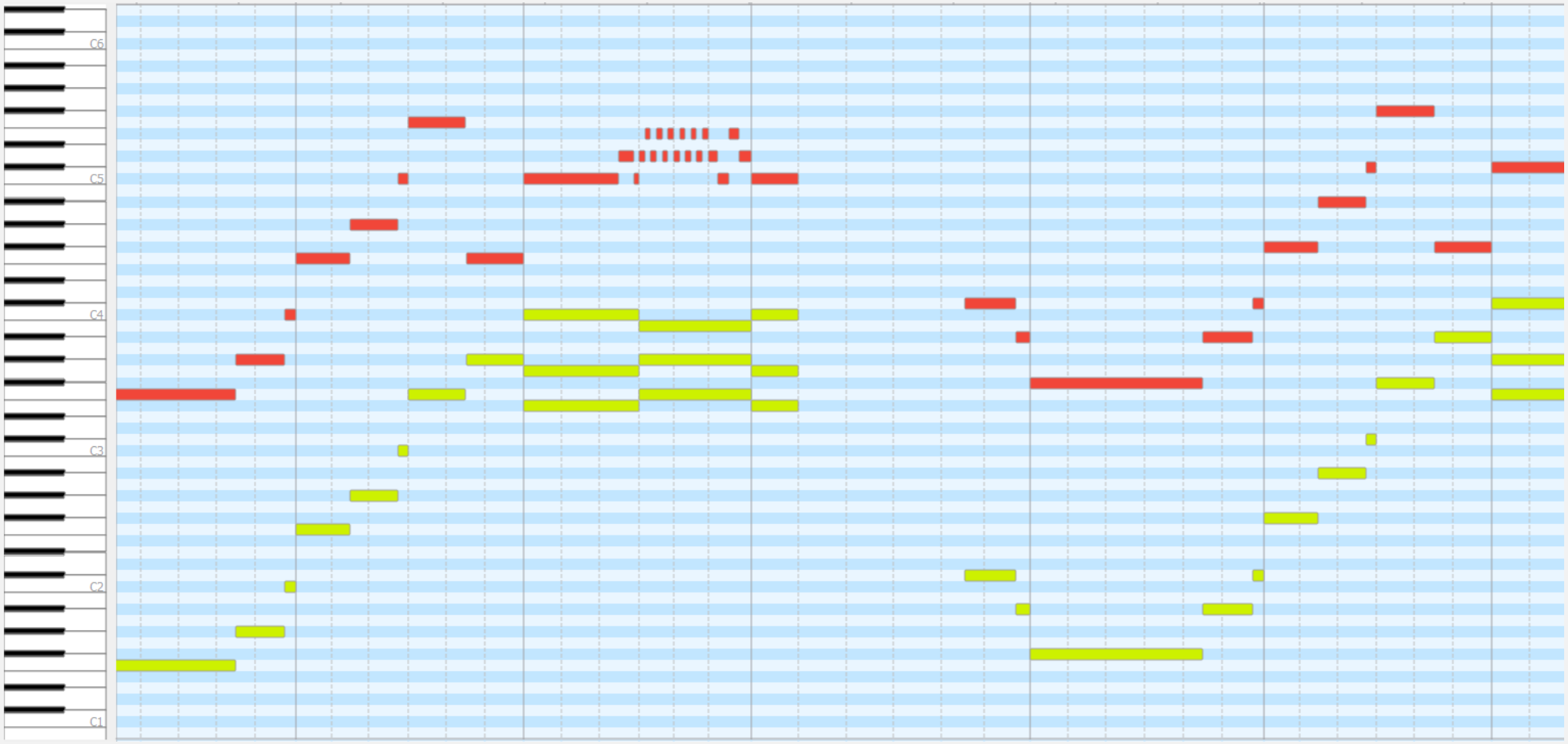


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$




h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}

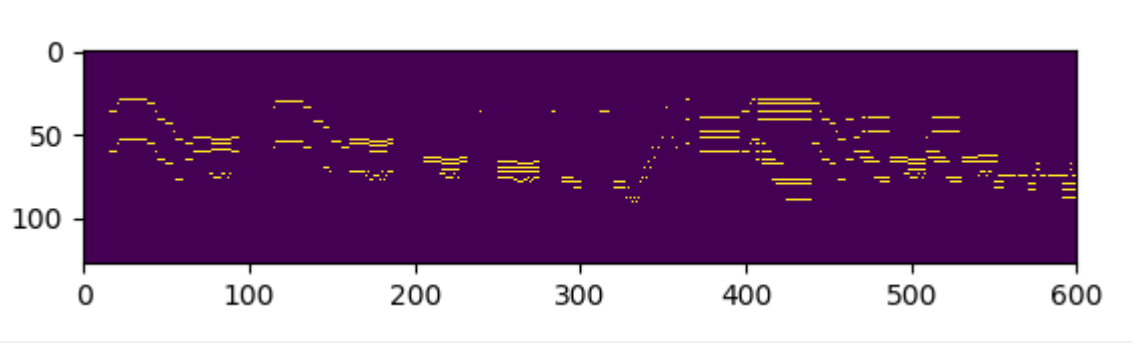
SEQUENTIAL DATA - MIDI



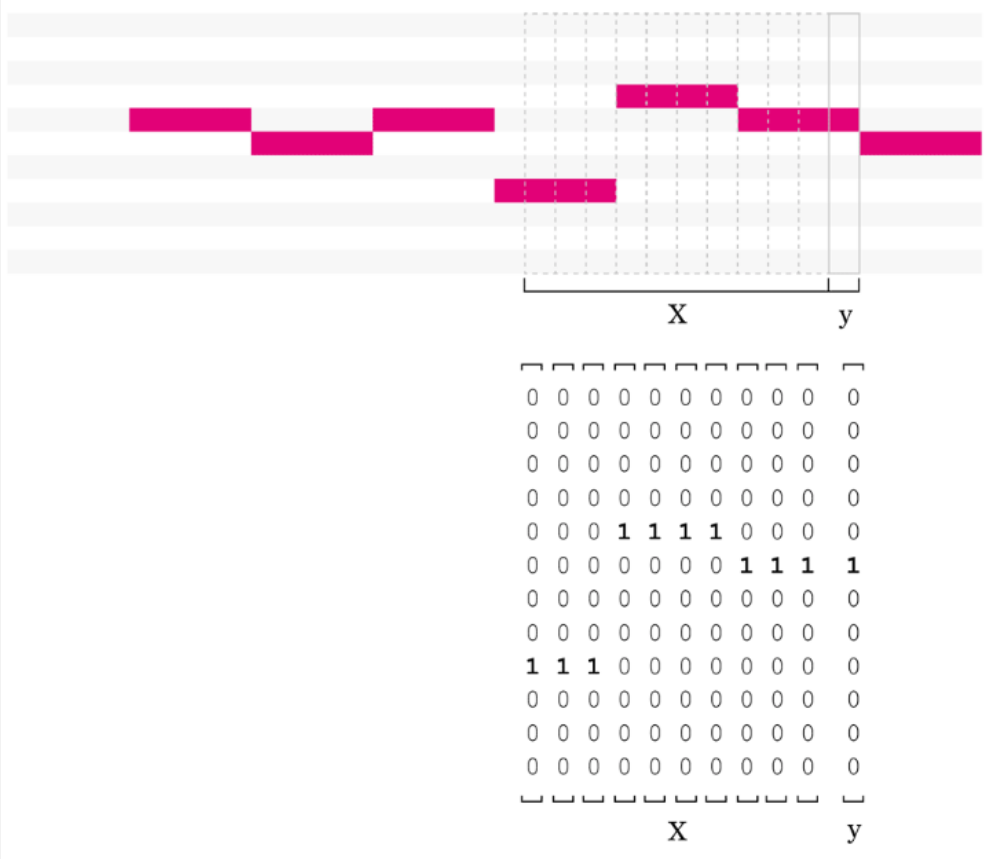
MIDI -> NUMPY

```
>>>
>>> for msg in mid.tracks[1][:100]:
...     print(msg)
...
<meta message track_name name='Piano right' time=0>
program_change channel=0 program=0 time=0
control_change channel=0 control=7 value=100 time=0
control_change channel=0 control=10 value=64 time=0
<meta message text text='bdca426d104a26ac9dcb070447587523' time=0>
control_change channel=0 control=91 value=127 time=0
note_on channel=0 note=60 velocity=35 time=2160
note_on channel=0 note=60 velocity=0 time=600
note_on channel=0 note=56 velocity=31 time=0
note_on channel=0 note=56 velocity=0 time=120
note_on channel=0 note=53 velocity=28 time=0
note_on channel=0 note=53 velocity=0 time=2160
note_on channel=0 note=56 velocity=35 time=0
note_on channel=0 note=56 velocity=0 time=600
note_on channel=0 note=60 velocity=30 time=0
note_on channel=0 note=60 velocity=0 time=120
note_on channel=0 note=65 velocity=35 time=0
note_on channel=0 note=65 velocity=0 time=720
note_on channel=0 note=68 velocity=35 time=0
note_on channel=0 note=68 velocity=0 time=600
note_on channel=0 note=72 velocity=30 time=0
```

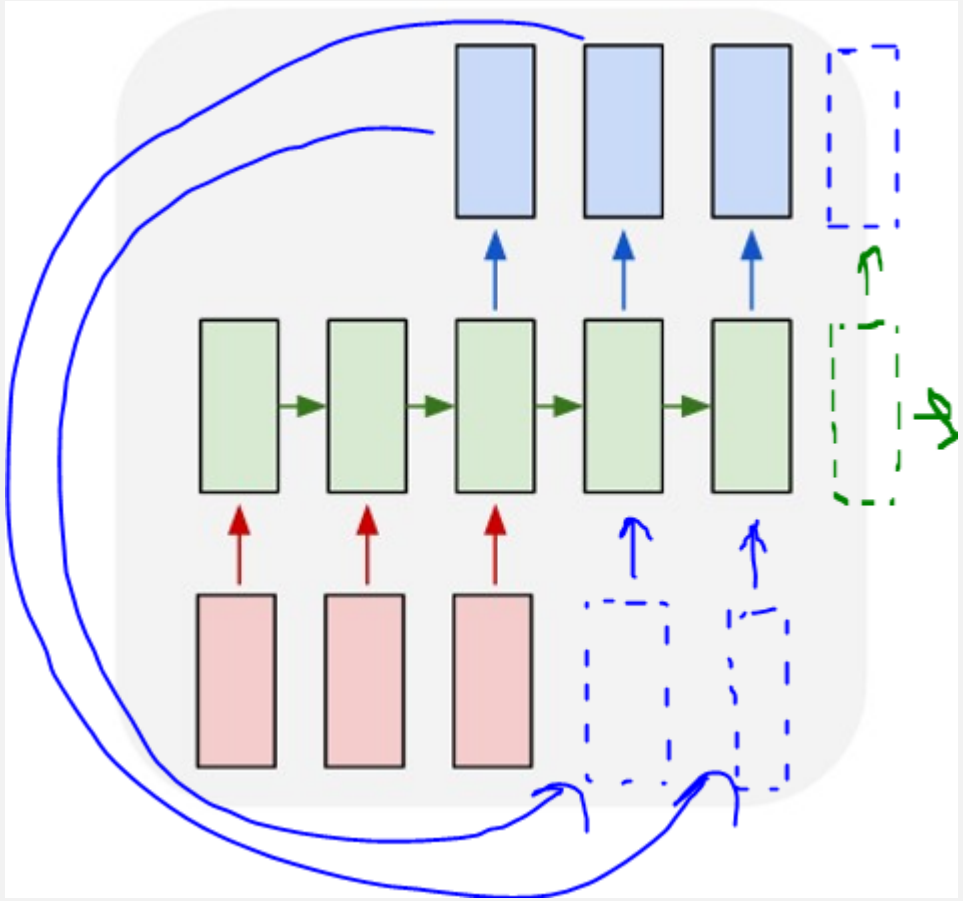
```
>>>
>>> track = midi_to_numpy_pipe(mid)
>>> track
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
>>>
>>> track.shape
(22355, 256)
>>>
>>>
>>> import matplotlib.pyplot as plt
>>> plt.imshow(track.T[:128, :600])
<matplotlib.image.AxesImage object at 0x00000165B30
>>> plt.show()
```



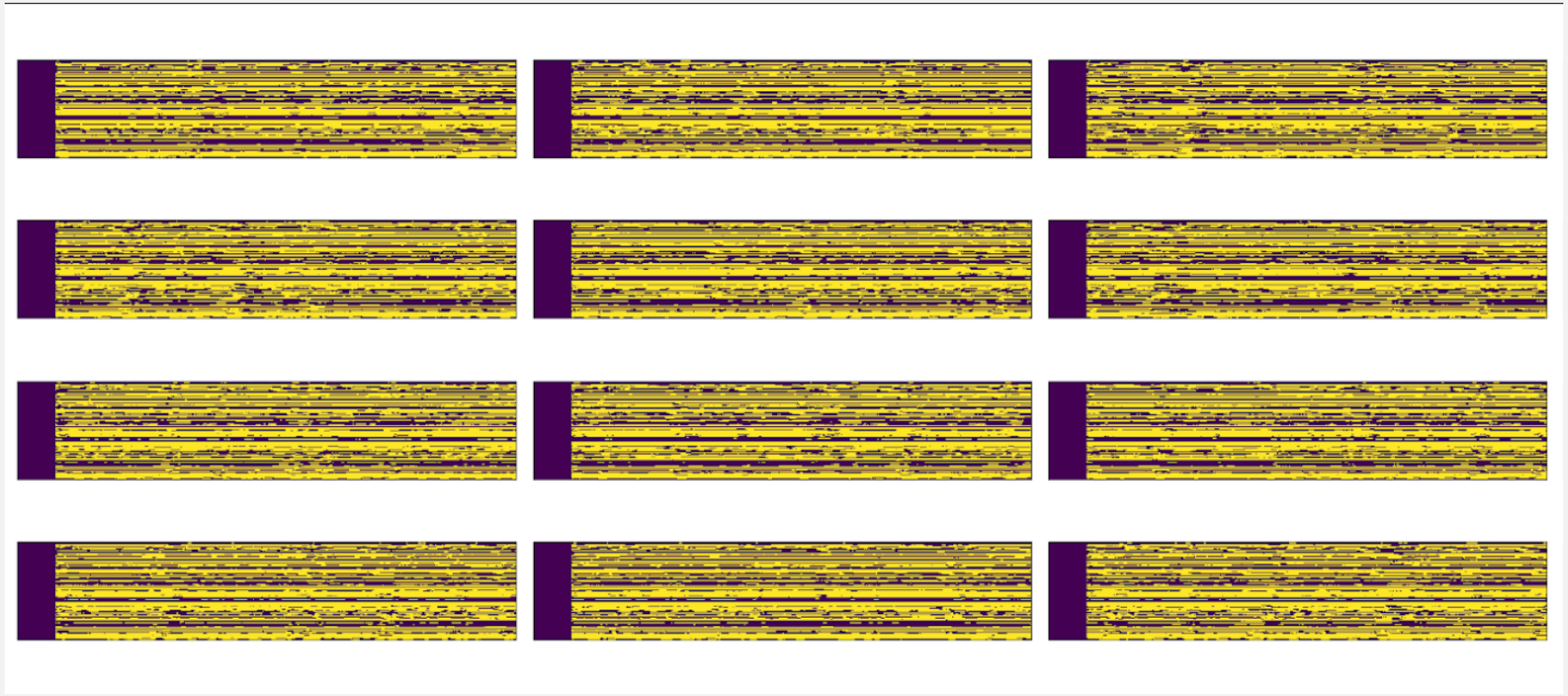
X AND Y


X
y
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
X
y

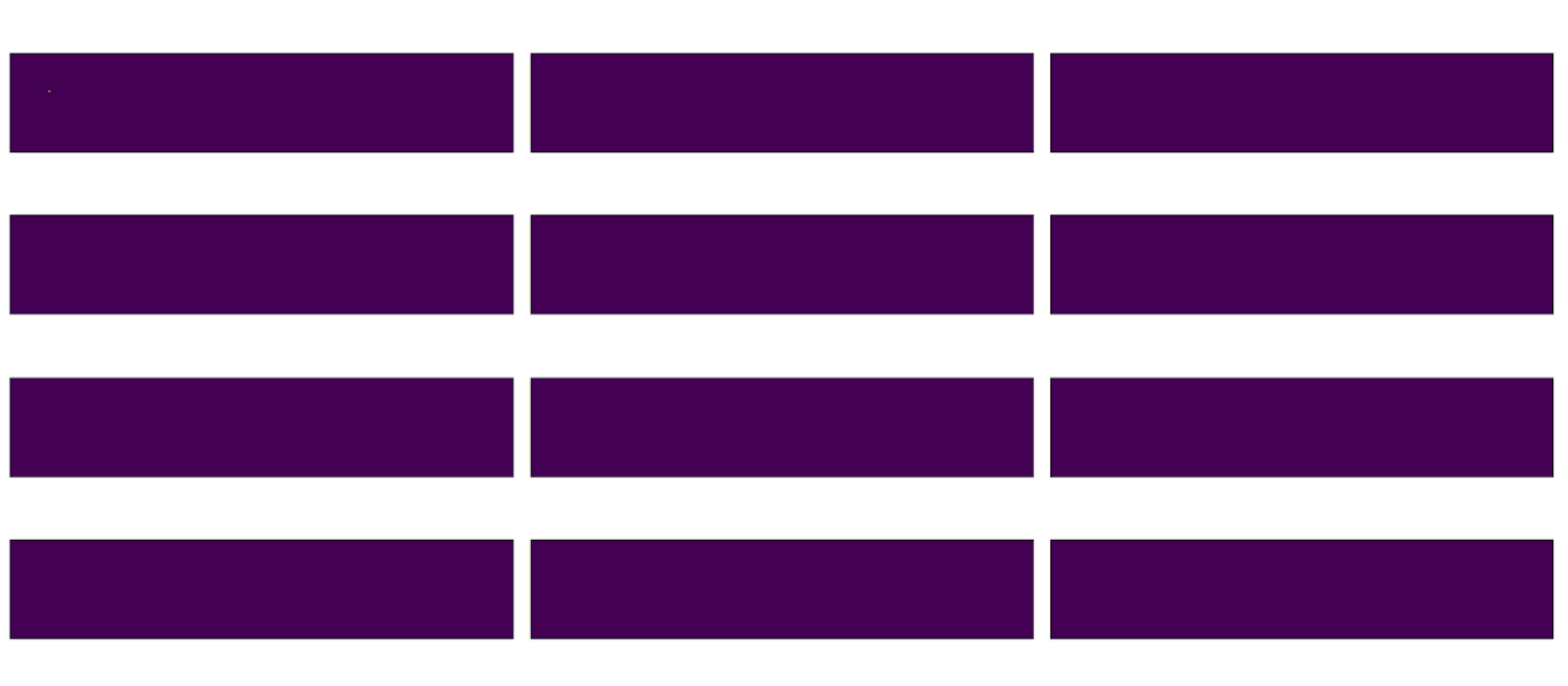
GENERATING NEW SAMPLES



LEARNING



Loss Function	Binary Crossentropy
Optimizer	Adam
Window size	50 (5s)
Hidden size	512
Samples per epoch	160000



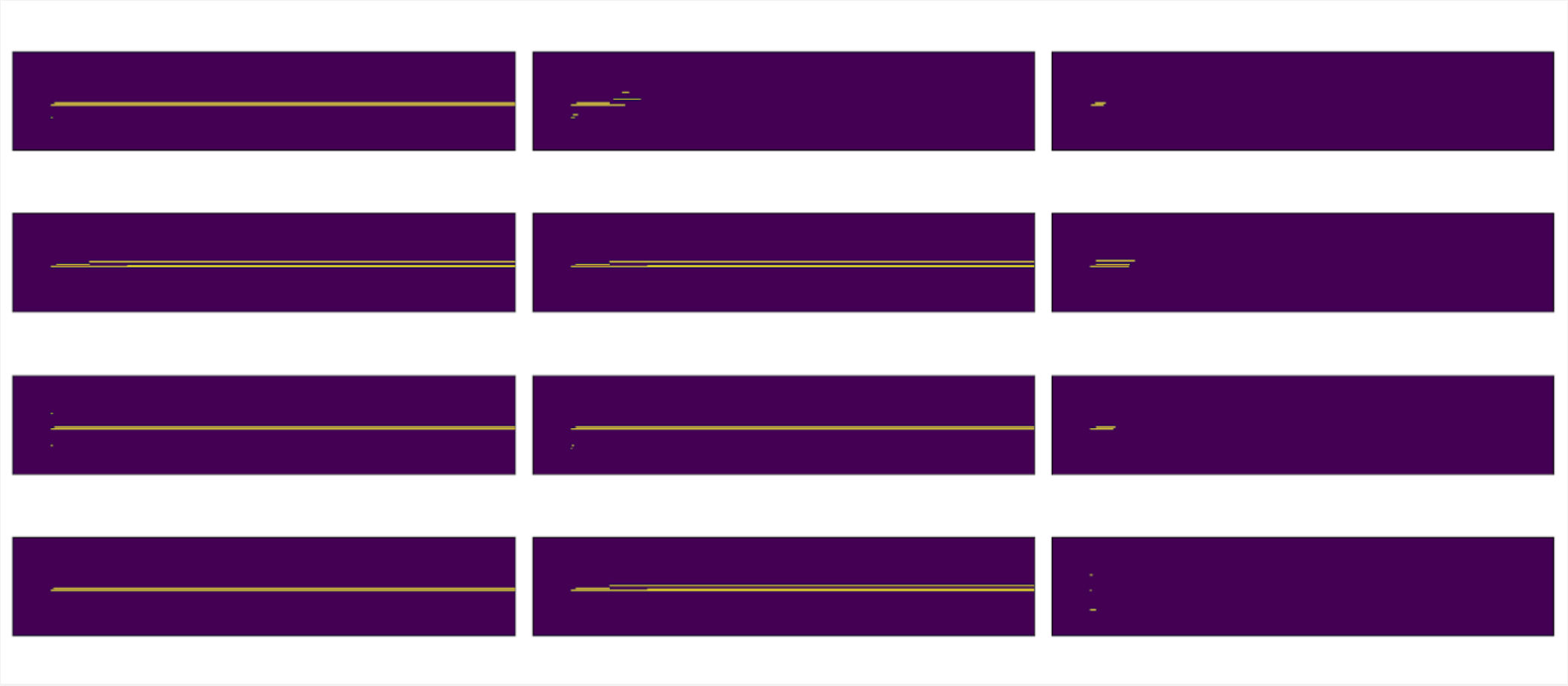
Epoch #1	
Loss	0.0535
Accuracy	0.9823
False Positives	8513
False Negatives	109658

Epoch #2	
Loss	0.0380
Accuracy	0.9867
False Positives	27069
False Negatives	65661

Epoch #3	
Loss	0.0380
Accuracy	0.9867
False Positives	27069
False Negatives	65661

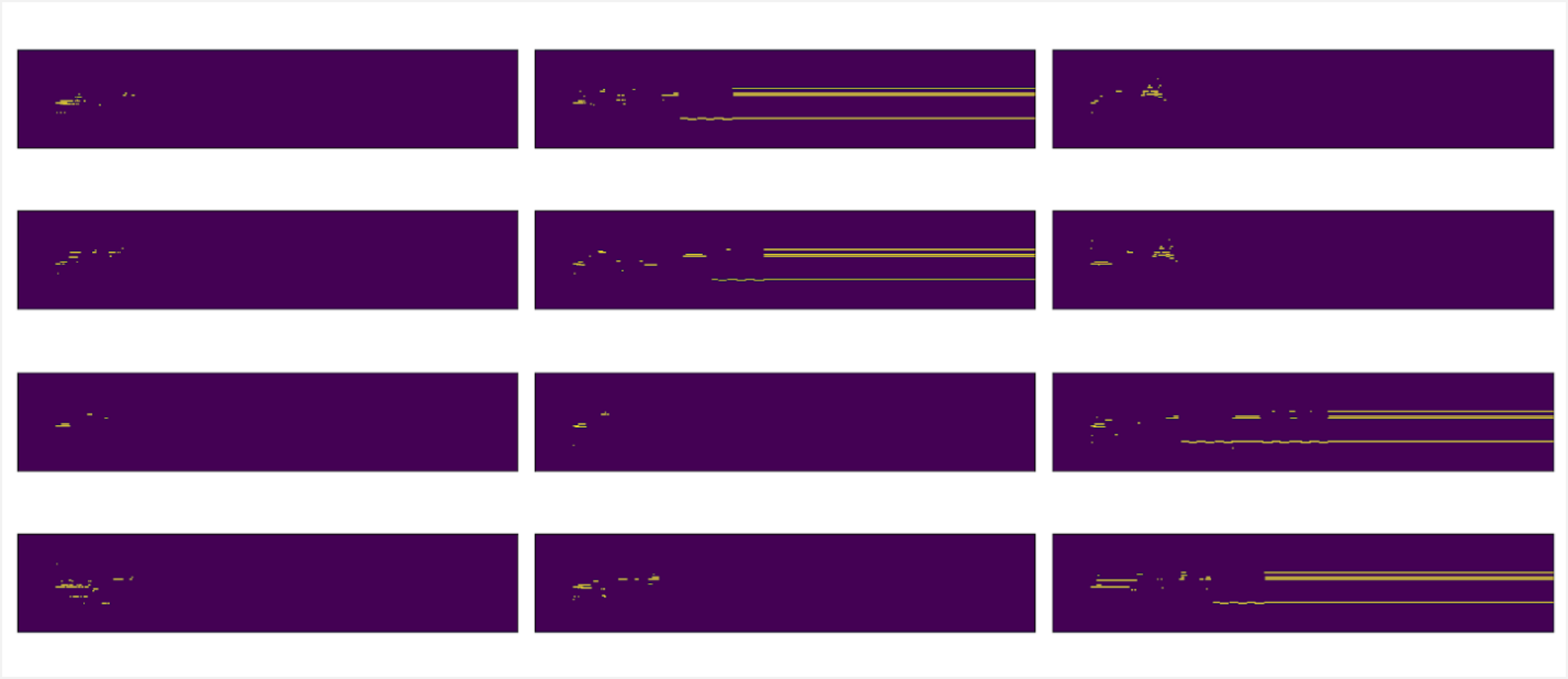
Epoch #4	
Loss	0.0284
Accuracy	0.9893
False Positives	23349
False Negatives	52003

Epoch #5	
Loss	0.0231
Accuracy	0.9914
False Positives	19882
False Negatives	43153

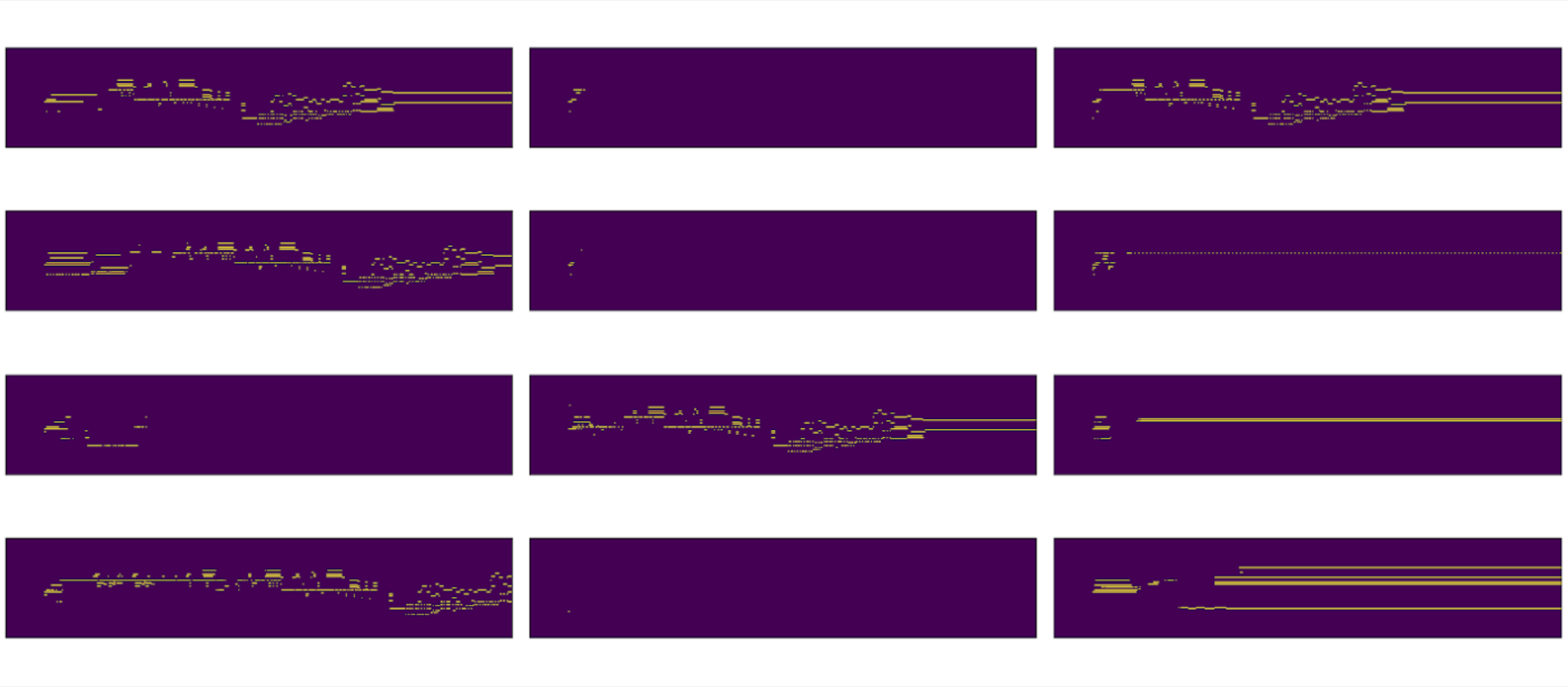


Epoch #6	
Loss	0.0194
Accuracy	0.9928
False Positives	14648
False Negatives	33459

Epoch #7	
Loss	0.0145
Accuracy	0.9951
False Positives	9852
False Negatives	23792

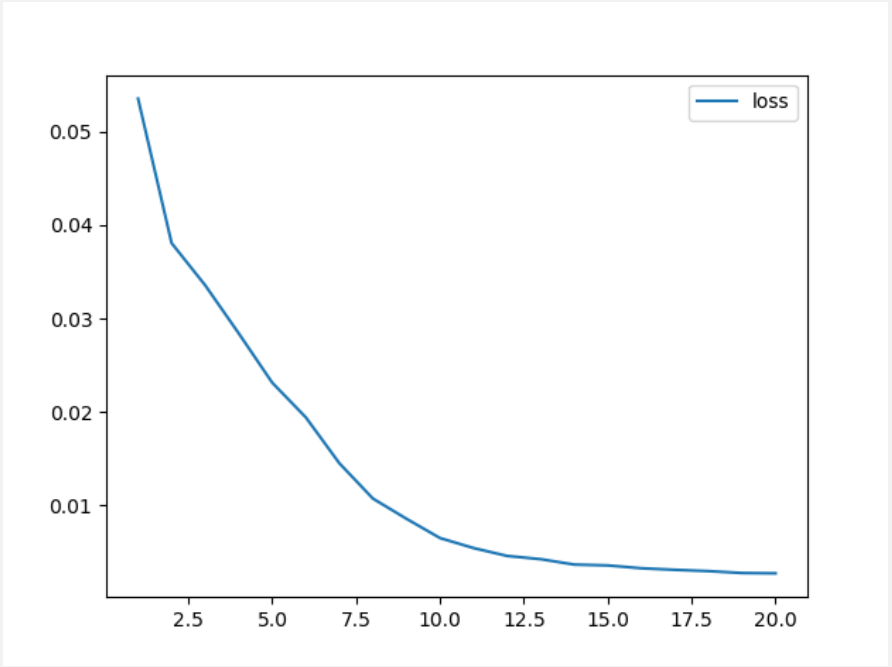


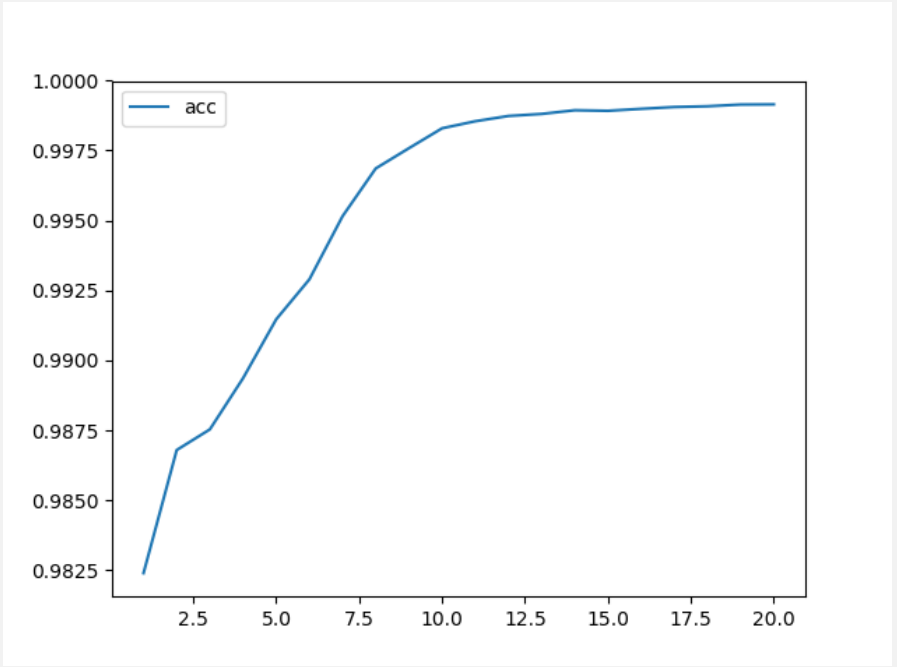
Epoch #10	
Loss	0.0065
Accuracy	0.9982
False Positives	3998
False Negatives	7885

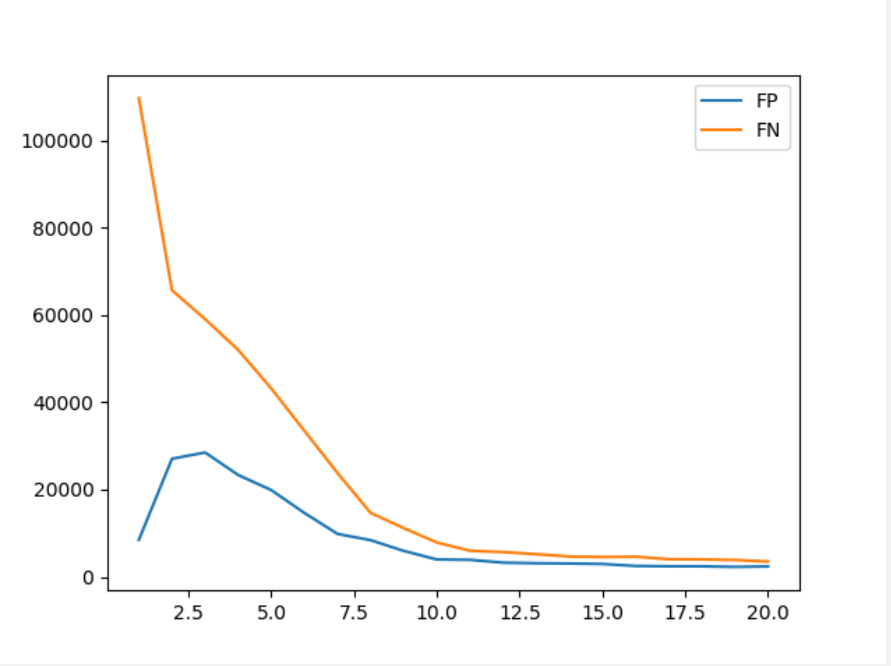


Epoch #14	
Loss	0.0036
Accuracy	0.9989
False Positives	3079
False Negatives	4675

Epoch #20	
Loss	0.0027
Accuracy	0.9991
False Positives	2404
False Negatives	3526

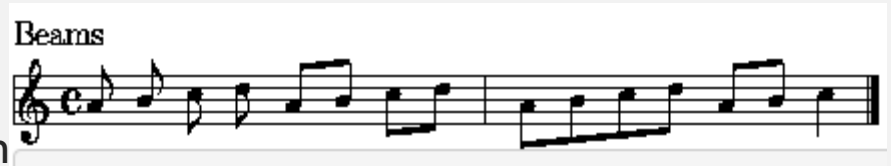

loss
0.05
0.04
0.03
0.02
0.01
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


acc
1.0000
0.9975
0.9950
0.9925
0.9900
0.9875
0.9850
0.9825
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


FP
FN
100000
80000
60000
40000
20000
0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

TO-DO

- Use more data
- Rethink Midi en
- Tune hyper-para
- Test different w
- Write custom loss function penalizing false negatives
- Add more layers
- Use music sheet .abc format




```
X:1
T:Beams
M:C
K:C
A B c d AB cd|ABcd ABc2|]
```

SOURCES

- https://towardsdatascience.com/back-to-basics-deriving-back-propagation-on-simple-rnn-lstm-feat-aidan-gomez-c7f286ba973d
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577