AI for the Media

Week 6, Generating Sequences

~ Vít Růžička

Overview

Generating Sequences (*pre-recorded lecture*):

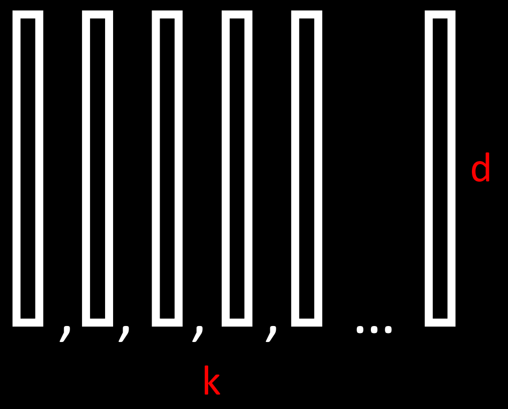
- **Recap**, and few examples of using sequential models for generating *new* data
- **In-depth look** at units and loss functions
- **Dataset** preparation

Practical session (*during the live session*):

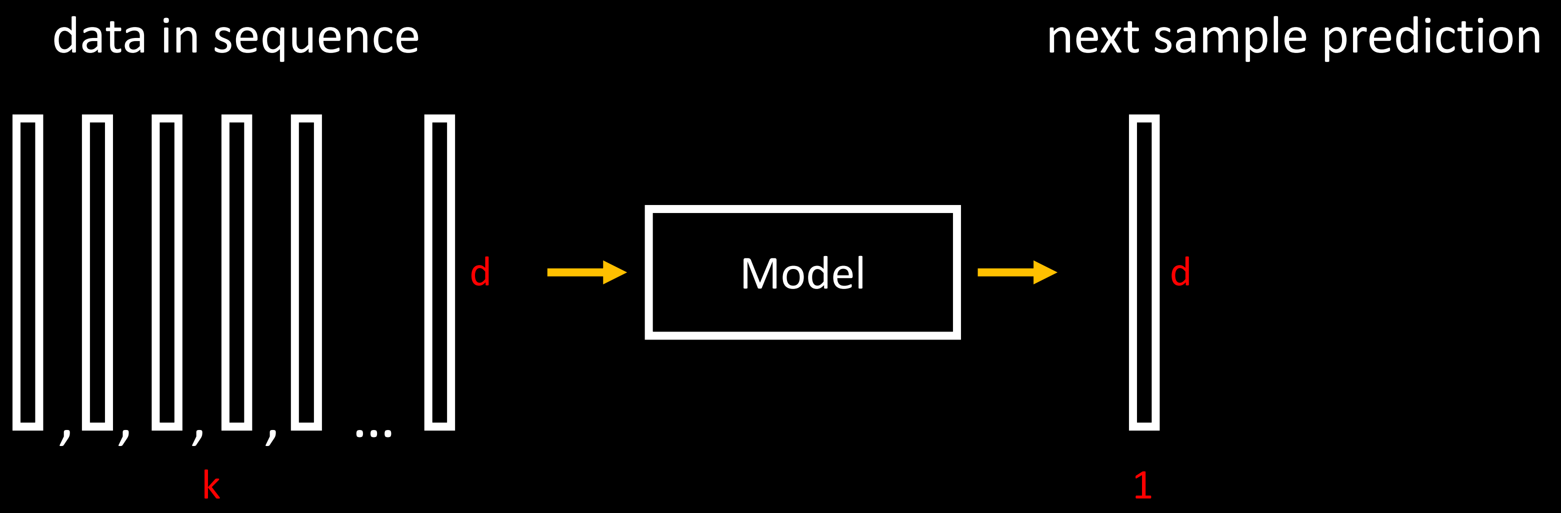
- **Code**: generative music using custom LSTM model

Motivation for sequential models

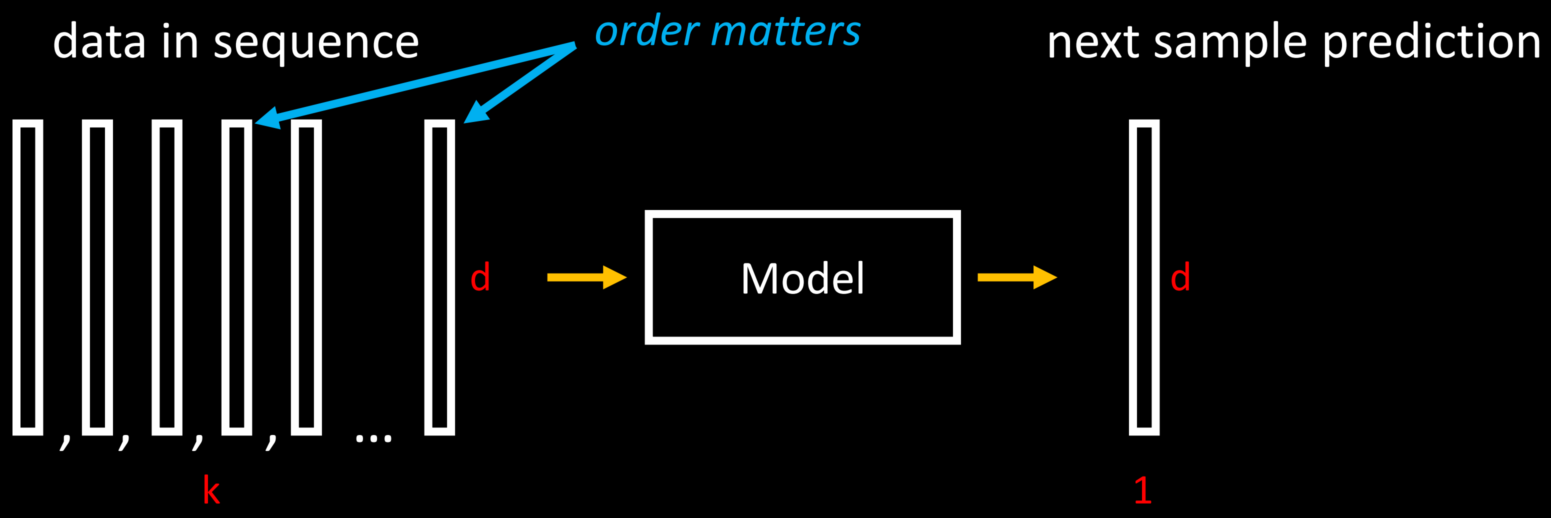
data in sequence



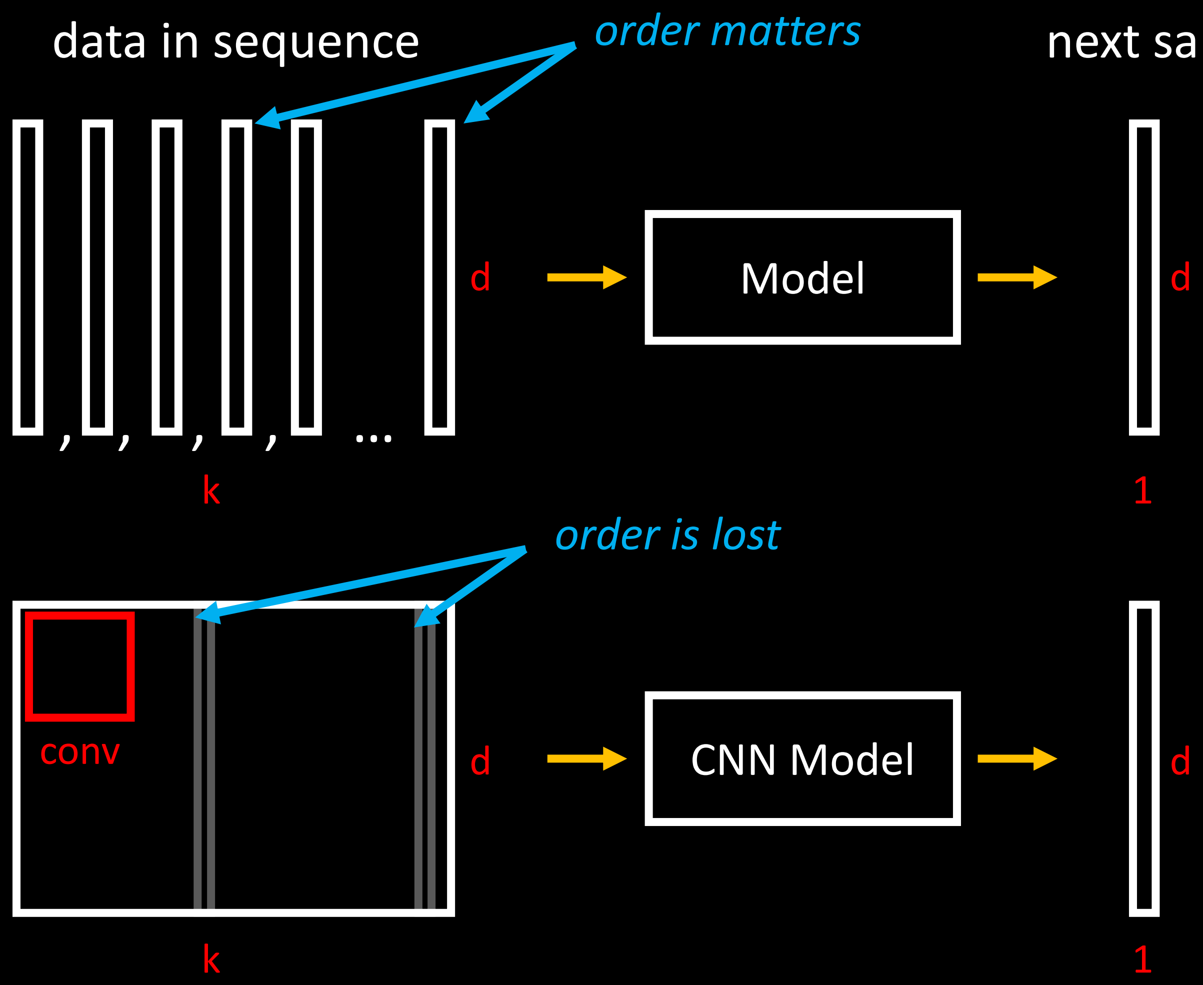
Motivation for sequential models



Motivation for sequential models

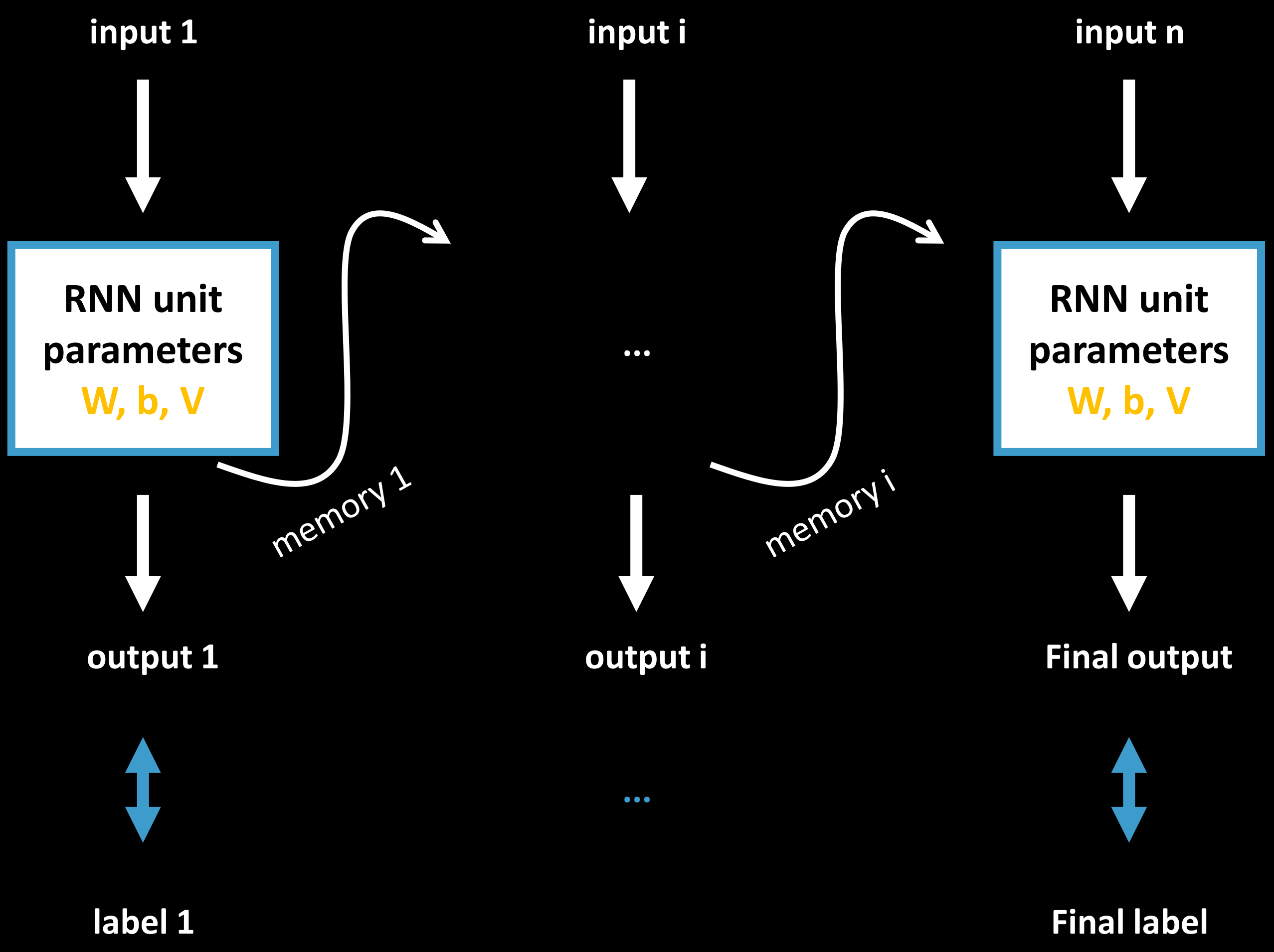


Motivation for sequential models

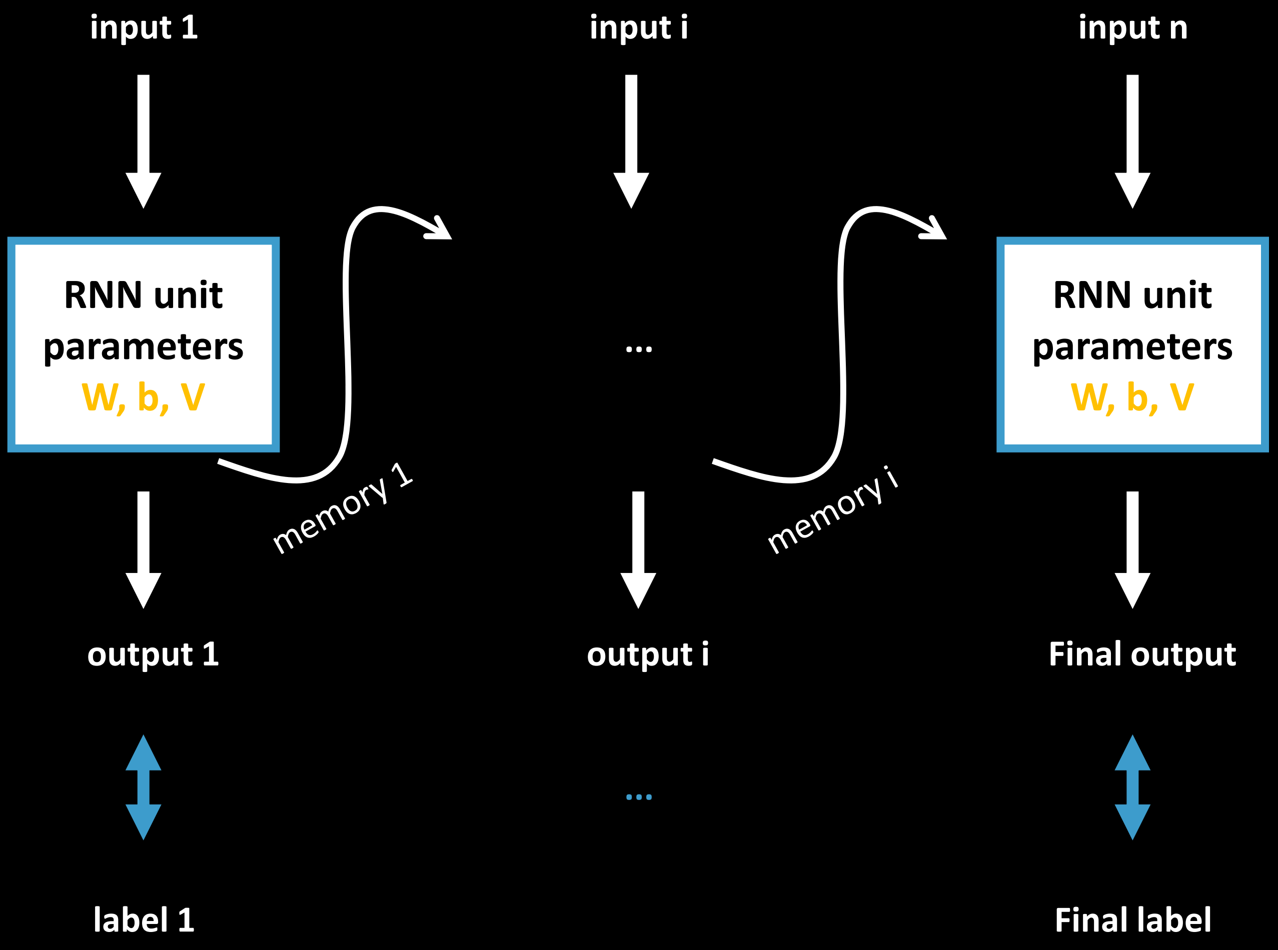


We could use any model to map this data – but there is a **type of models which is designed with sequential data in mind** and maps this sequentiality better.

Recap: Sequential model

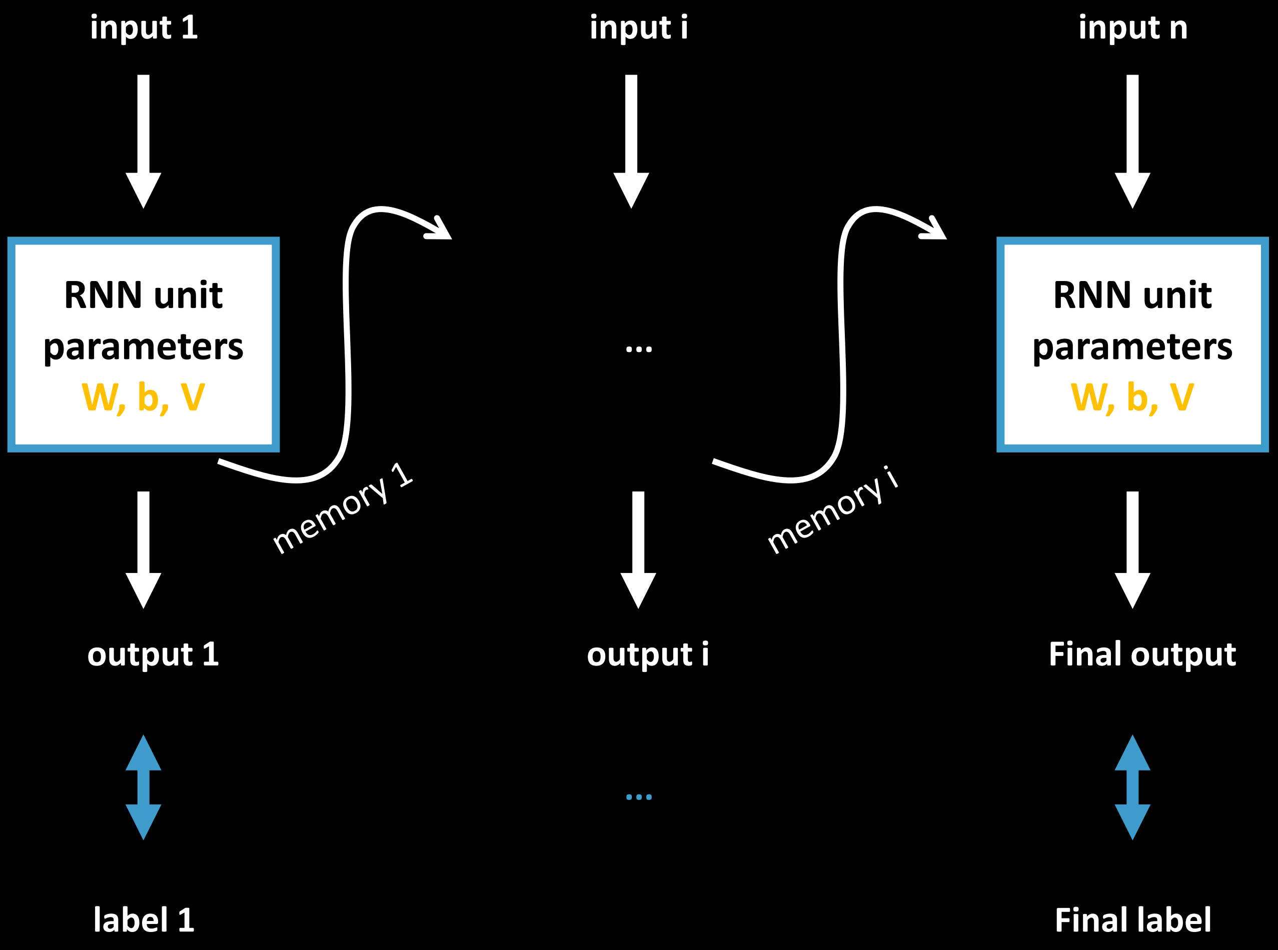


Recap: Sequential model



- Allows us to **feed in data sequentially**
- Learned **parameters** which influence **a.)** how we **transform the input data**, **b.)** what we **keep in the memory** for the later inputs
- We **train** these parameters using a **loss** and optimization algorithm

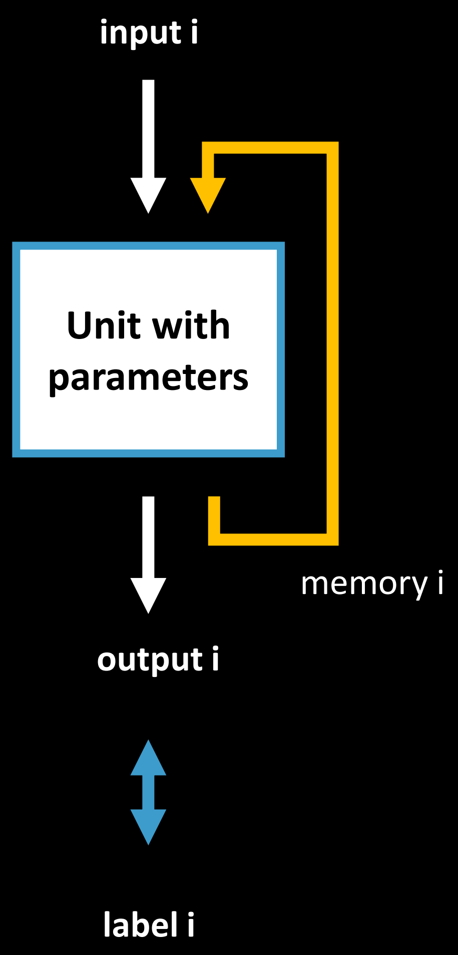
Recap: Sequential model



Today:

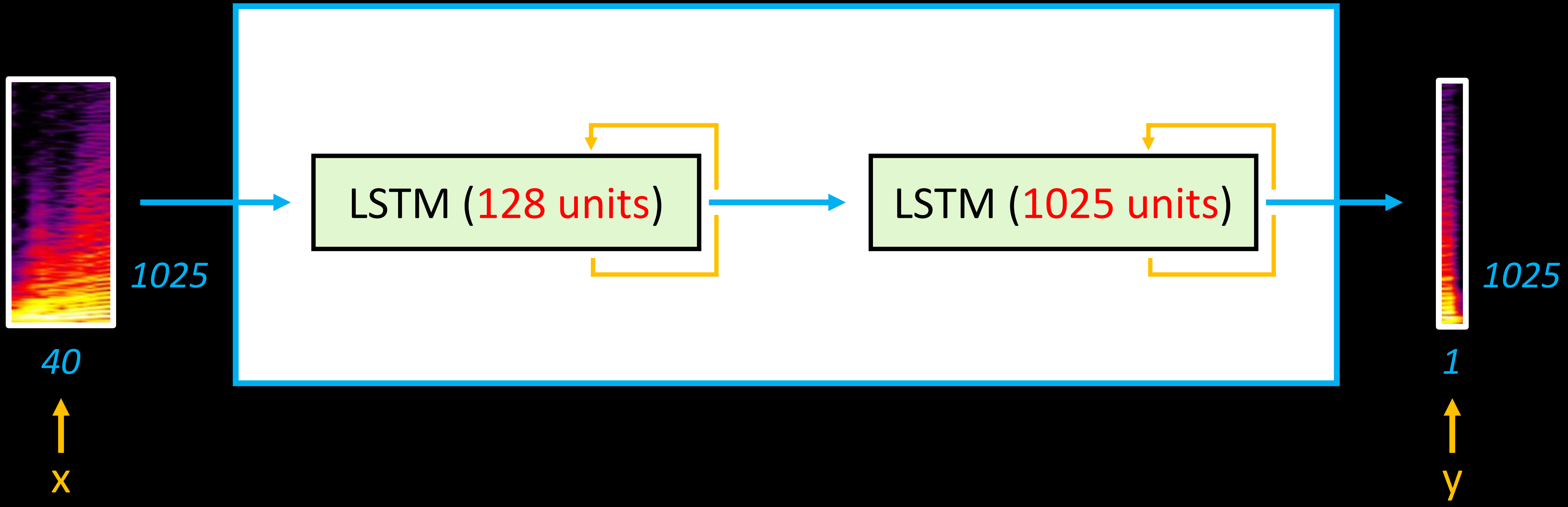
- Closer look into the **unit design**
- How can this be used for **data generation**

Recap: Sequential model



PS: We can use a more concise illustration >

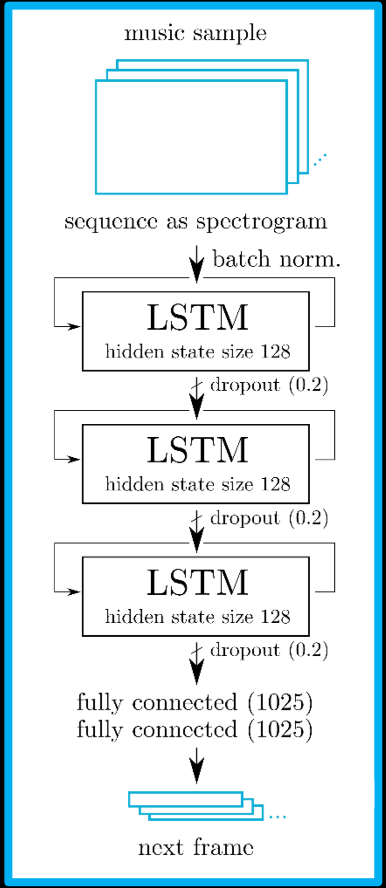
Example 1: Music Generation



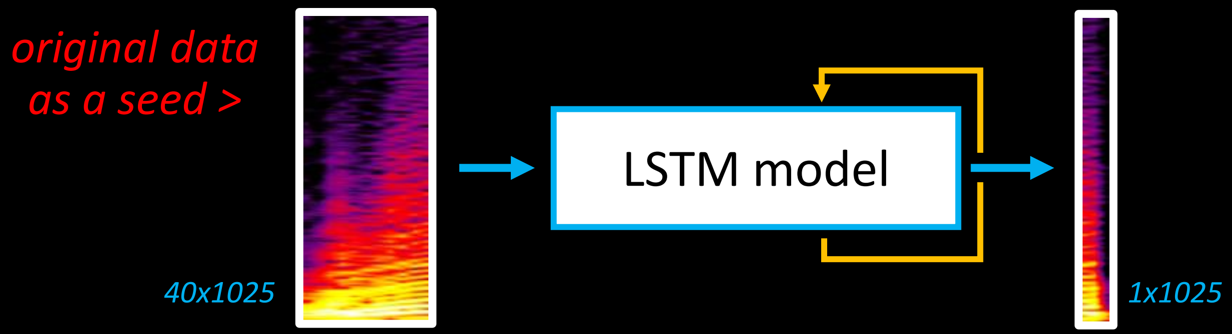
- Example of a possible model with two layers of LSTMs. We prepared our dataset as **sequences of 40 frames predicting the next 1 frame** (*"many to one"*), the model **learns the transformation** of **x -> y**.

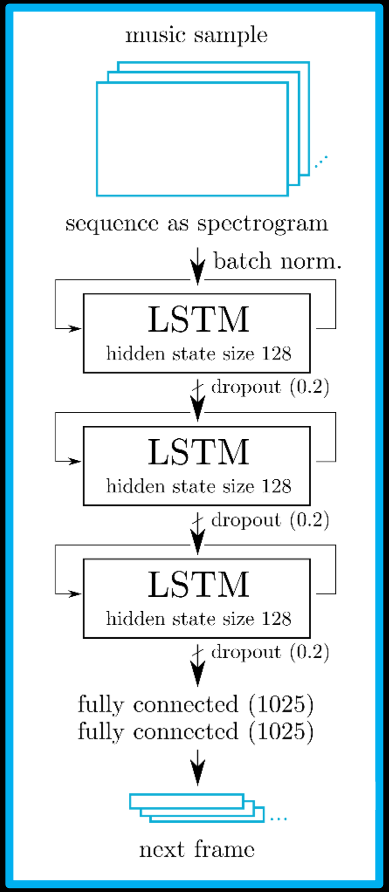
Using a trained LSTM model:

PS: In a real scenario the used model was only *slightly* more complicated >

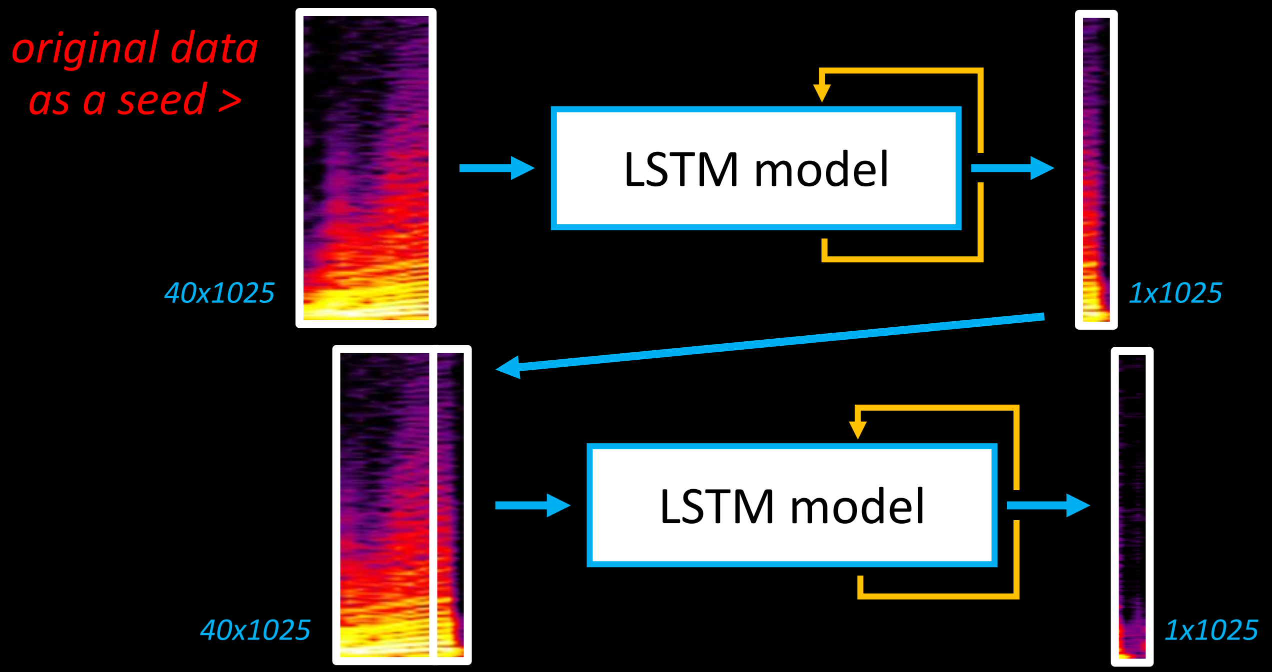


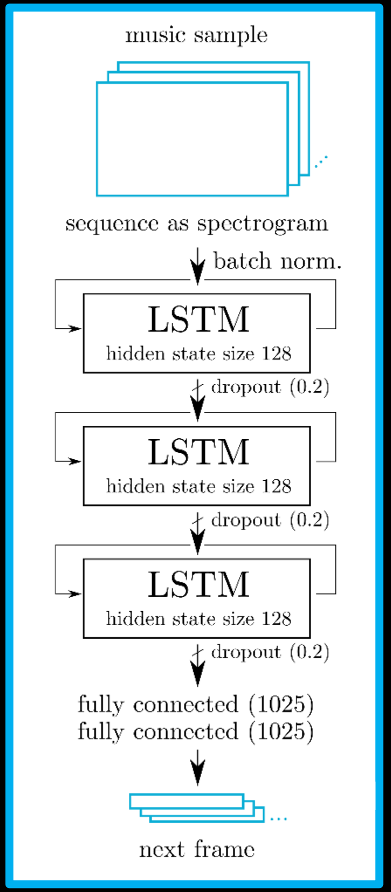
Using a trained LSTM model:



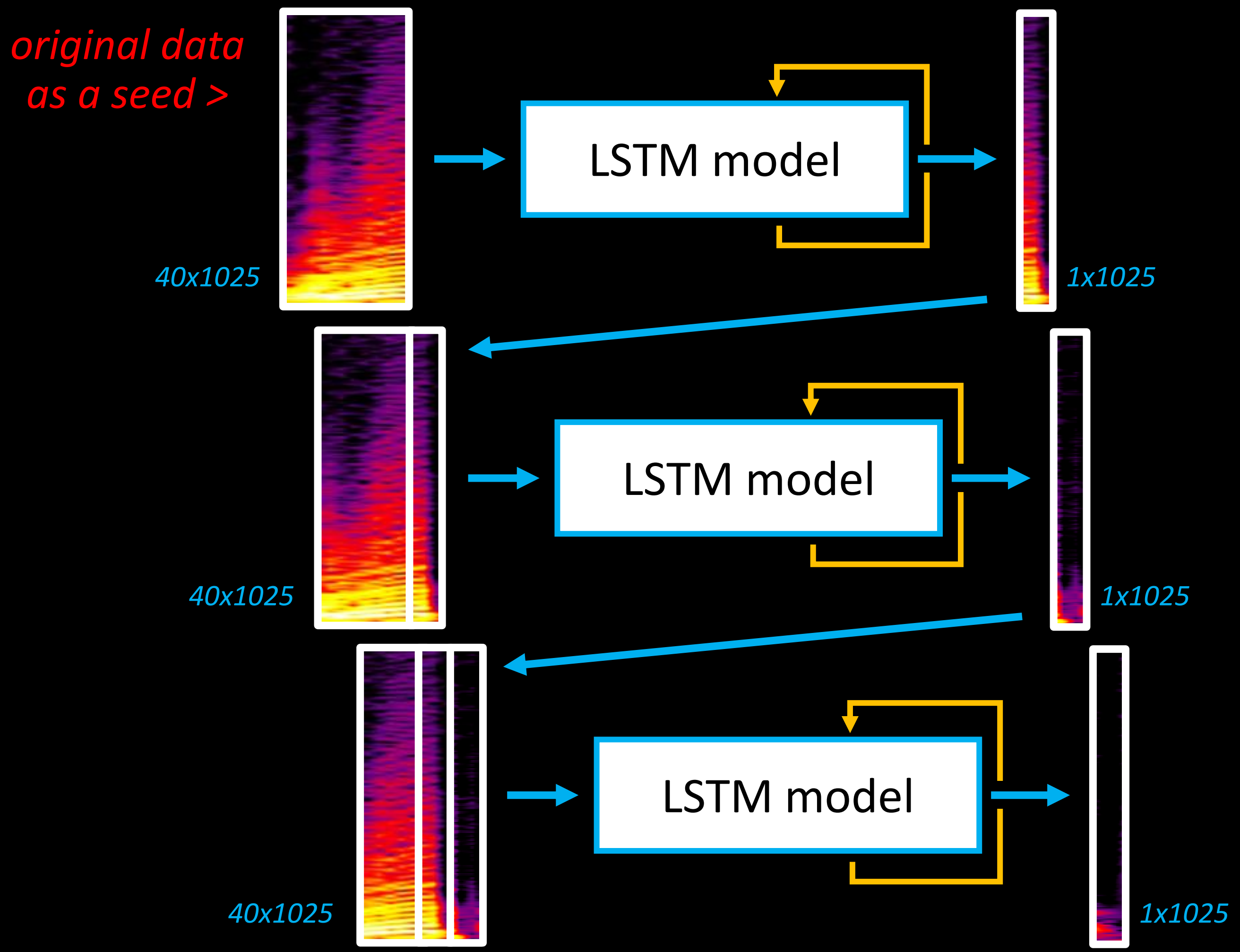


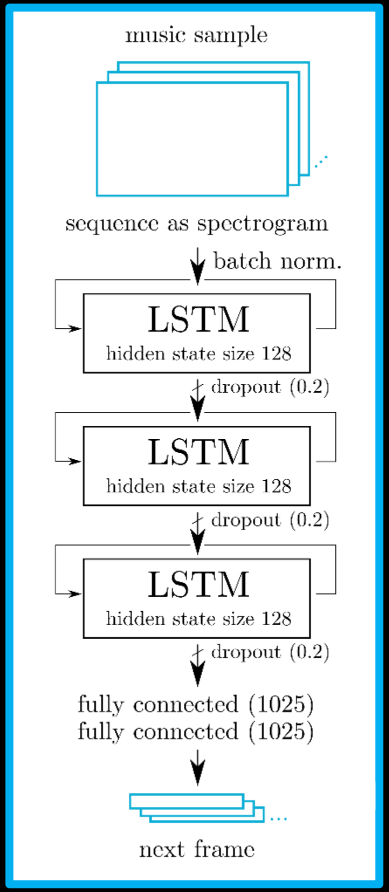
Using a trained LSTM model:



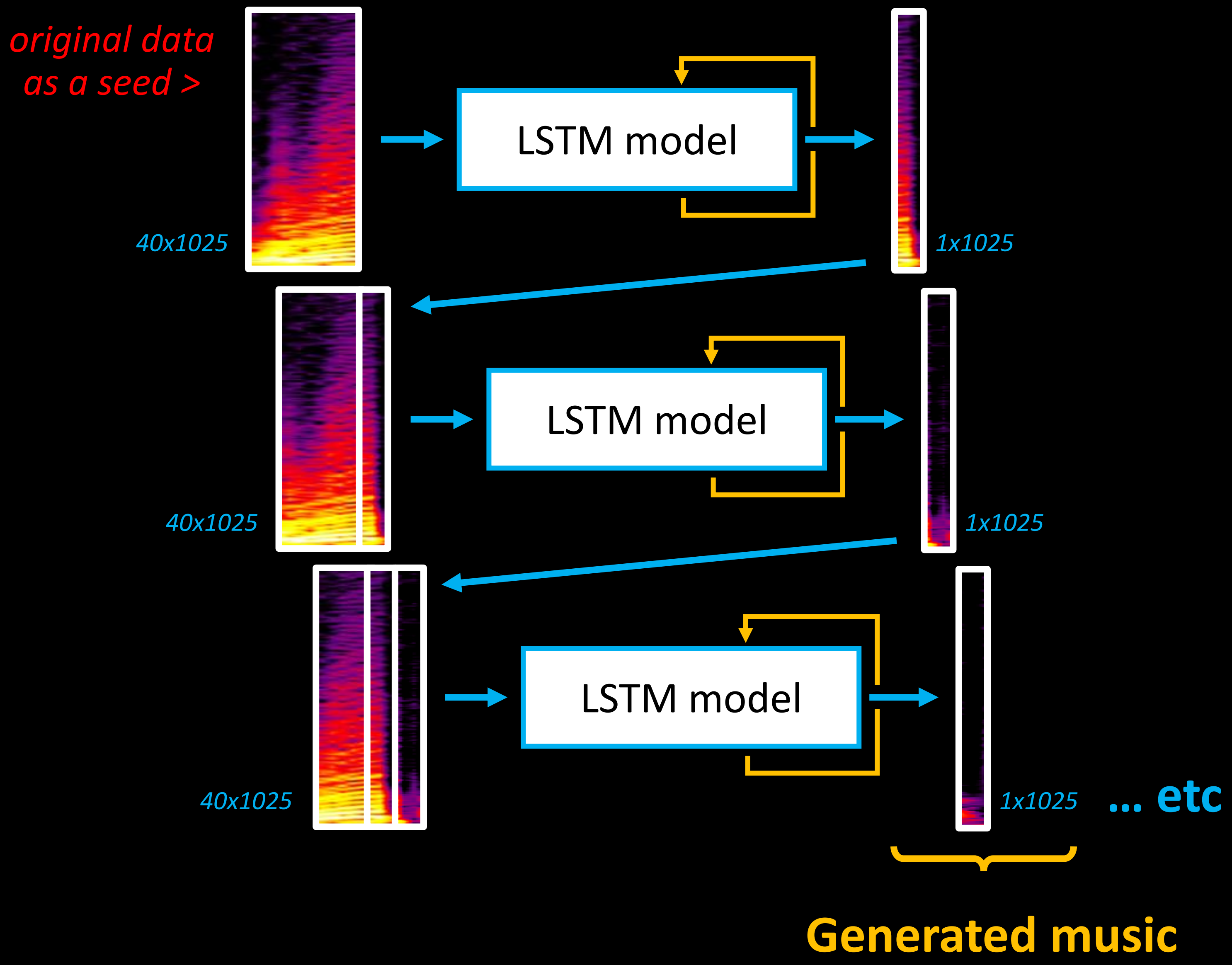


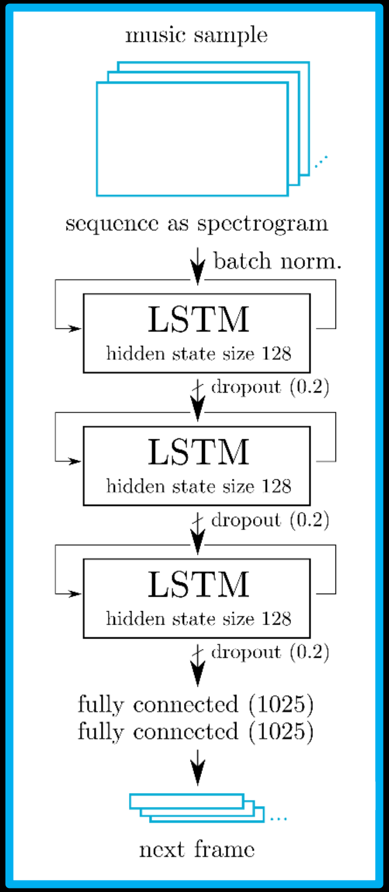
Using a trained LSTM model:





Using a trained LSTM model:





- Check the **generated music sample**: ml-jazz-meanderings-ml-generated-sounds-1

Example 2: Stock Market

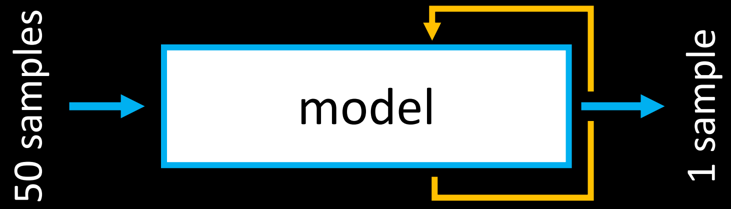
Model **x -> y**

Layer (type)	Output Shape
lstm_1 (LSTM)	(None, 50, 96)
dropout_1 (Dropout)	(None, 50, 96)
lstm_2 (LSTM)	(None, 50, 96)
dropout_2 (Dropout)	(None, 50, 96)
lstm_3 (LSTM)	(None, 50, 96)
dropout_3 (Dropout)	(None, 50, 96)
lstm_4 (LSTM)	(None, 96)
dropout_4 (Dropout)	(None, 96)
dense_1 (Dense)	(None, 1)

- Similarly prepared data from the stock market.

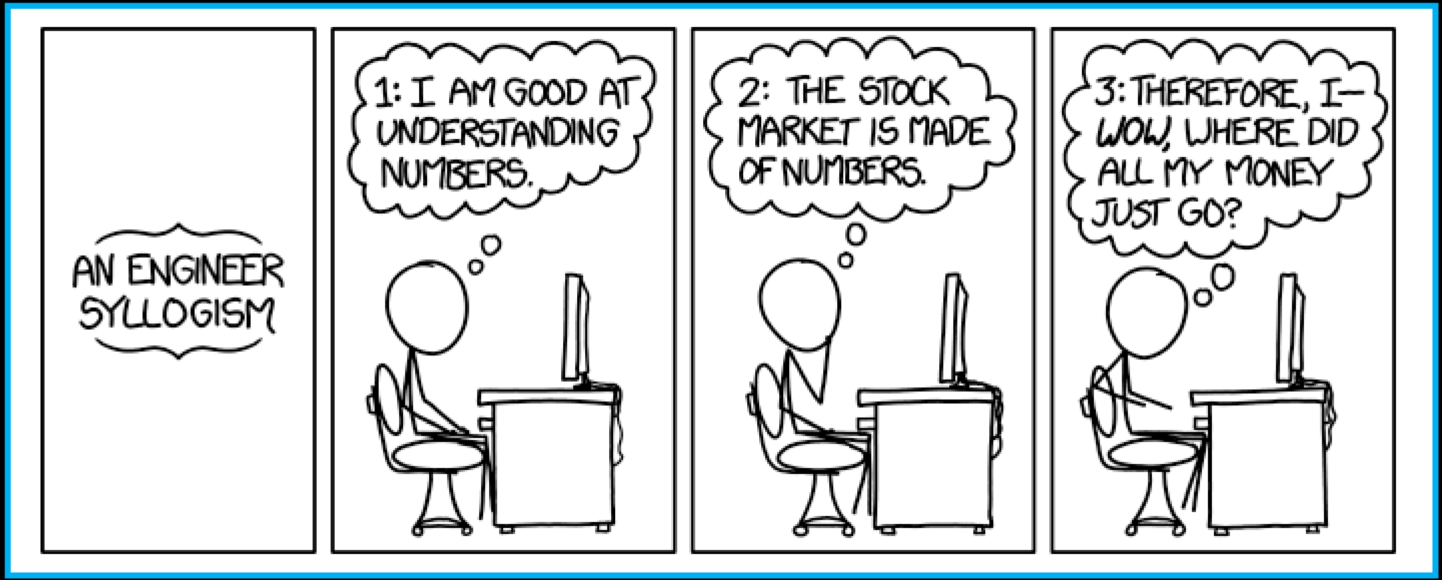
[paper]

Example 2: Stock Market



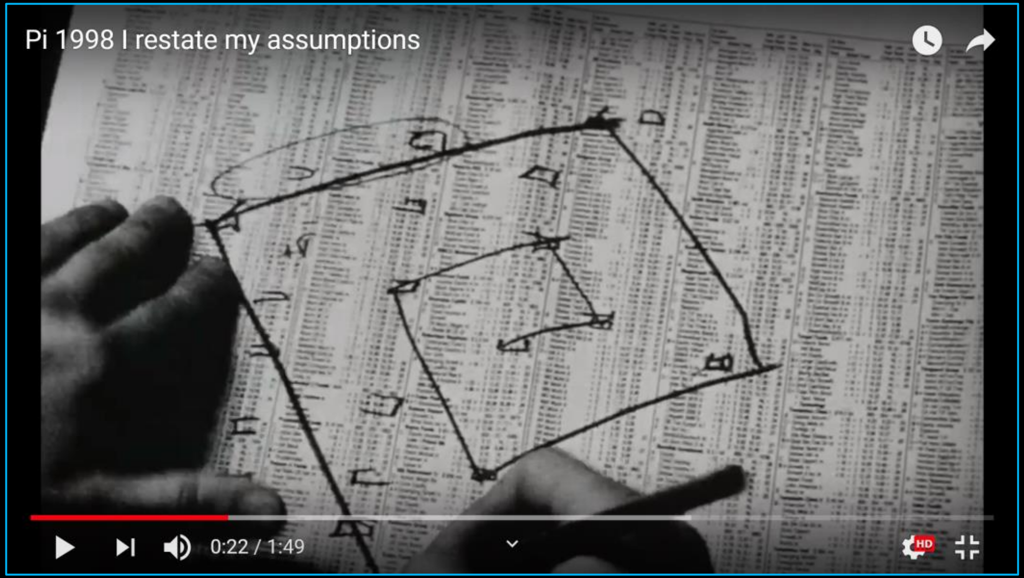
- There are similarities between the **"next value prediction" task** and a generative task (given the way how we use these models).
- Remember this when you are using them for your creative projects -> **we might want irregularities**, model breaking, training on multiple sources … etc.

[paper]



Mouseover: The less common, even worse outcome: "3: [everyone in the financial system] WOW, where did all my money just go?"

Film: **Pi** (1998) - **Darren Aronofsky**



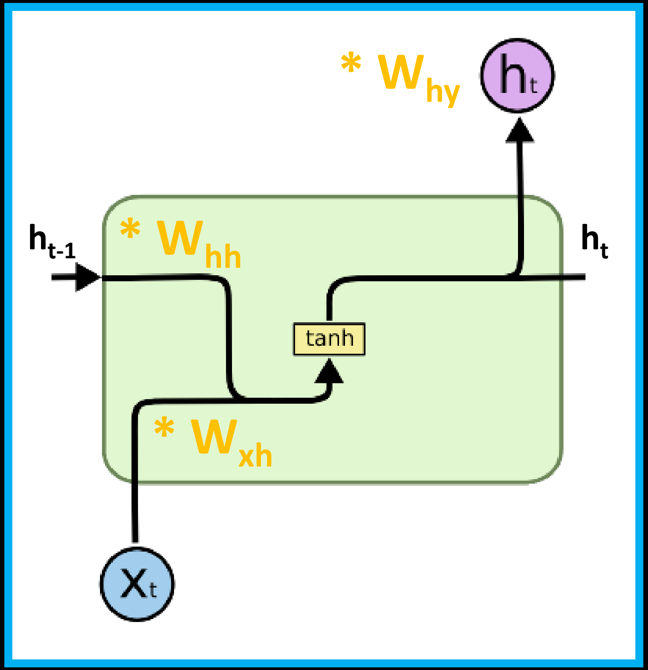
I restate my assumptions: One, Mathematics is the language of nature. Two, Everything around us can be represented and understood through numbers. Three: If you graph the numbers of any system, patterns emerge. Therefore, there are patterns everywhere in nature.

Clip: youtube.com/watch?v=AdKCLDYHXgM

***)** PS: now it’s a good time to take a pause

Vanilla RNN

- Simpler unit design:



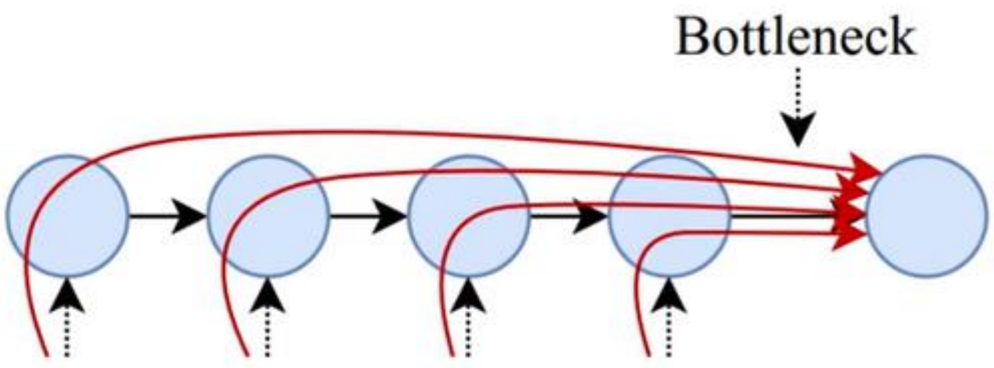
As math formulas:

$$h_t = tanh(W_{xh} * x_t + W_{hh} * h_{t-1})$$
$$y_t = W_{yh} * h_t$$

```
class RNN:
    def step(self, x):
        # update the hidden state
        self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
        # compute the output vector
        return np.dot(self.W_hy, self.h)
```

[code]

Forgetting information



(a) The bottleneck of RNN seq2seq models

$$h_t = tanh(W_{xh} * x_t + W_{hh} * h_{t-1})$$
$$y_t = W_{yh} * h_t$$

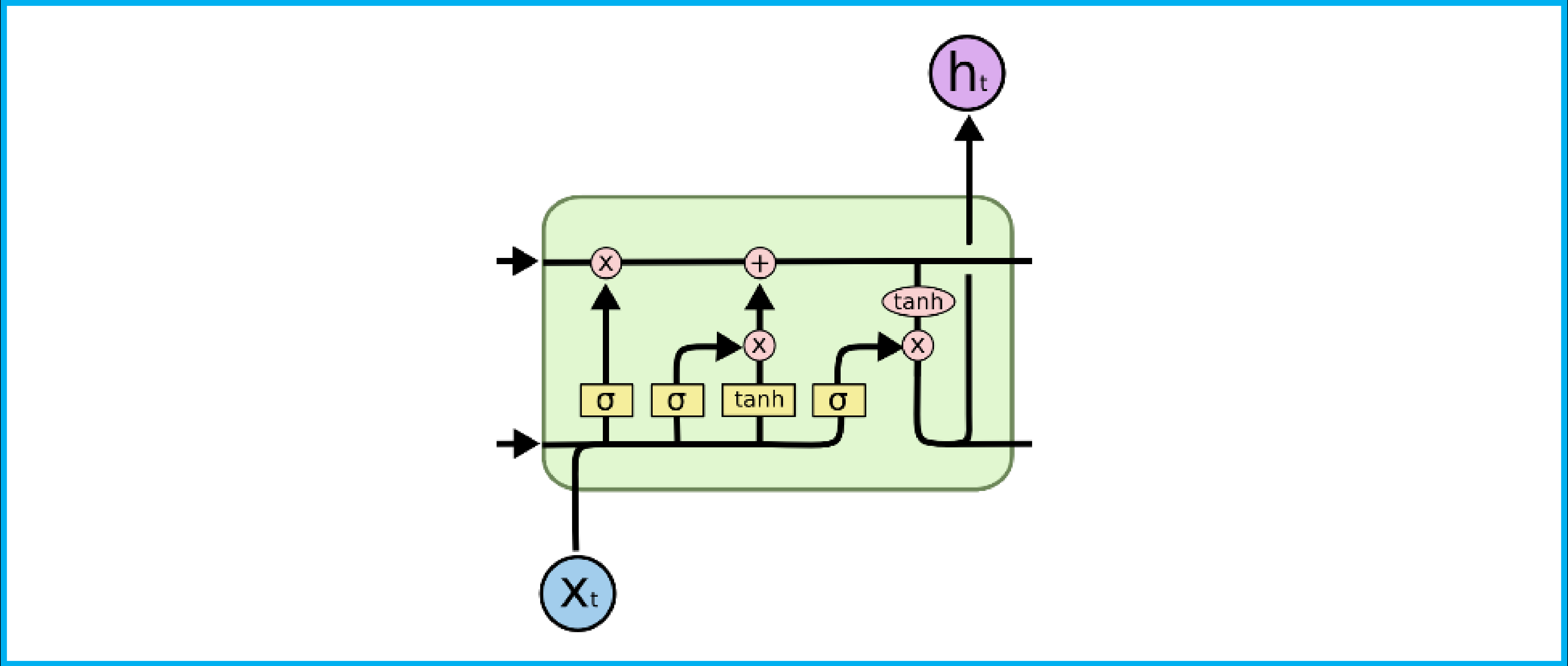
- Which has one weakness – $\mathbf{h_t}$ carries information to be stored in the memory and also for output…

Long Short Term Memory Unit (LSTM)

This will be a rough animation of data passing through LSTM. Keep in mind that we don’t really care about each detail at this point – rather we want to see why is it better at remembering long-term dependencies than the vanilla RNN …

[blog]

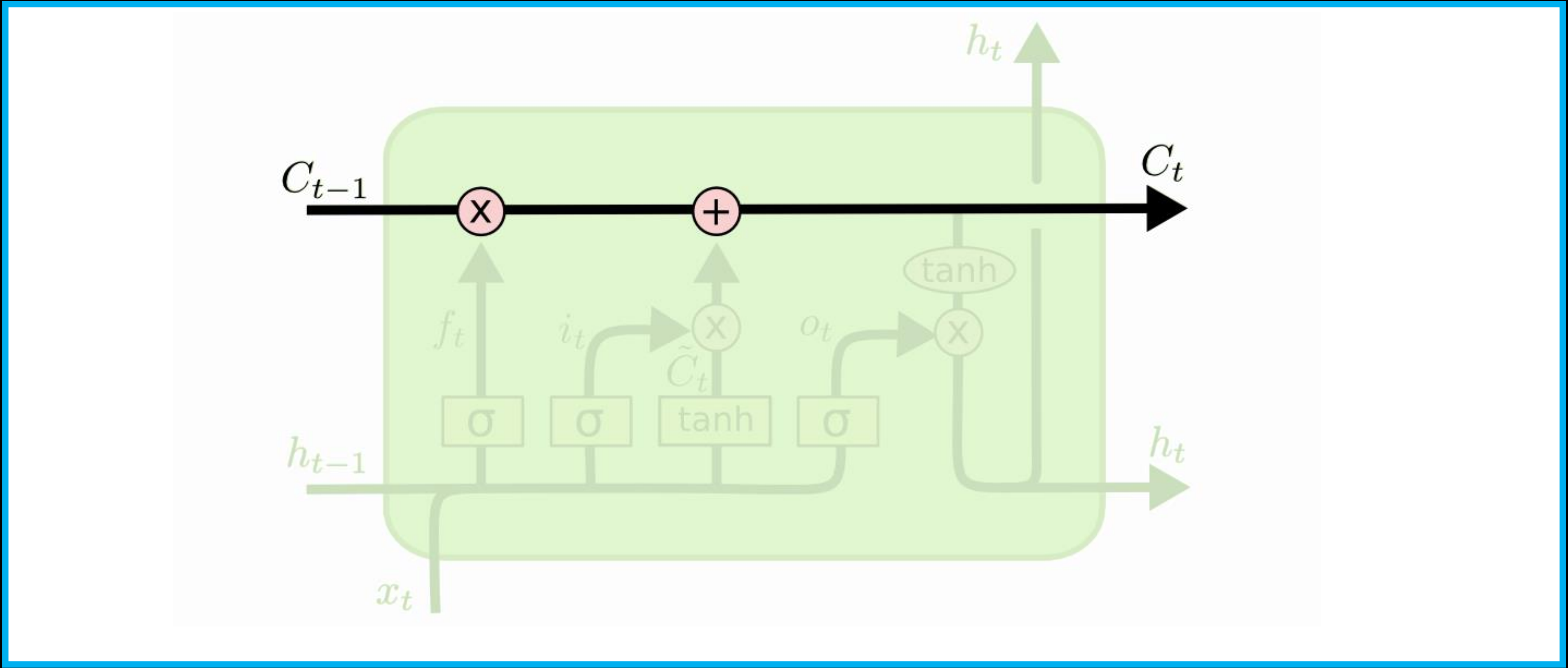
Long Short Term Memory Unit (LSTM)



PS: don’t need to know exactly, we care about the concept …

[blog]

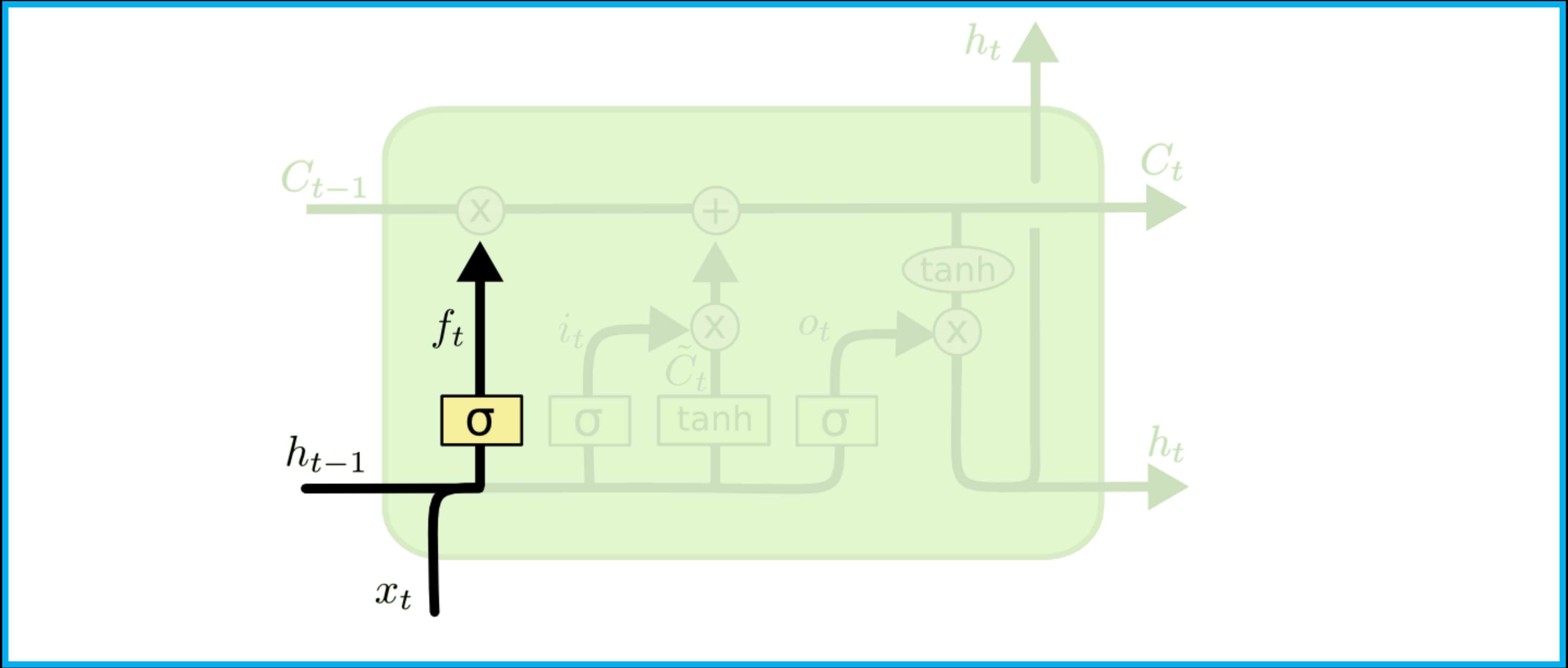
LSTM



- Main idea: independent channel to carry long-term memory

[blog]

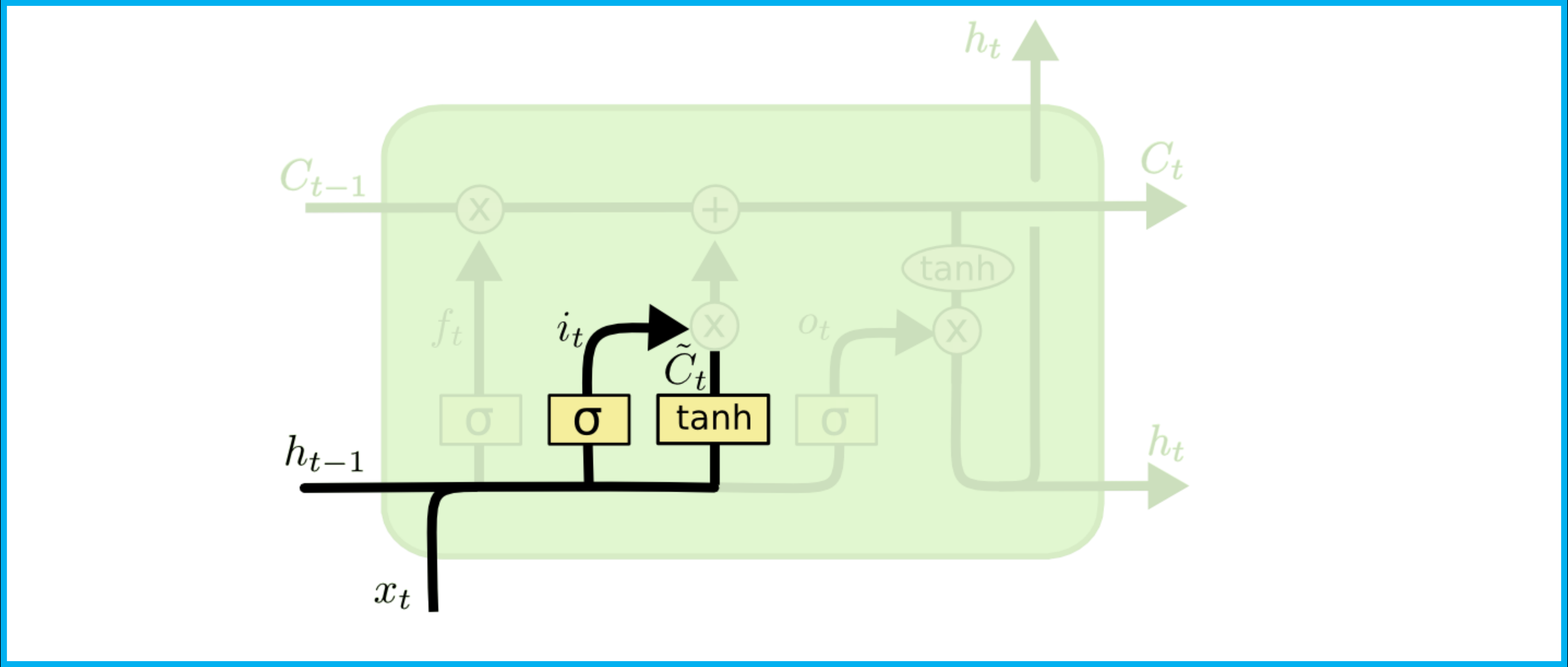
LSTM



- Forget gate influences what we delete from memory

[blog]

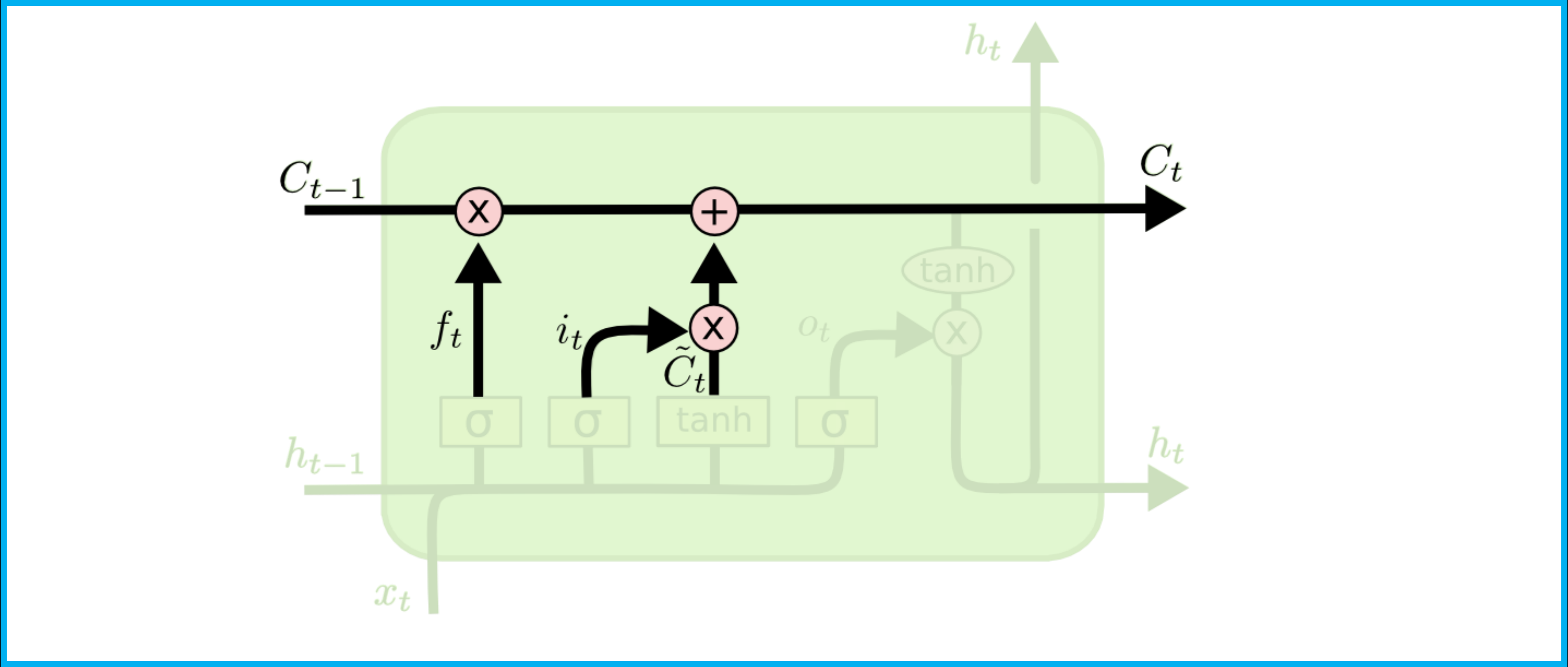
LSTM



- Input gate selects what to read from input

[blog]

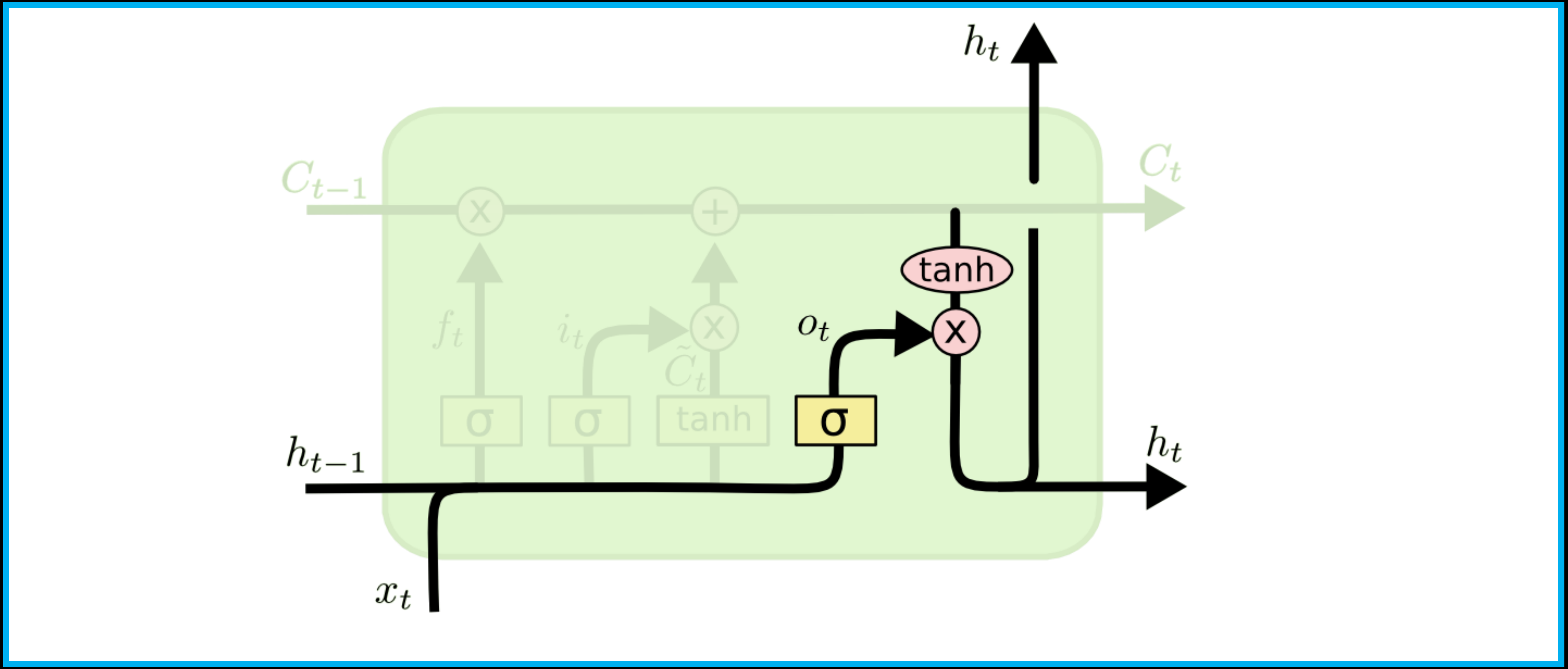
LSTM



- (Then we add it to the memory)

[blog]

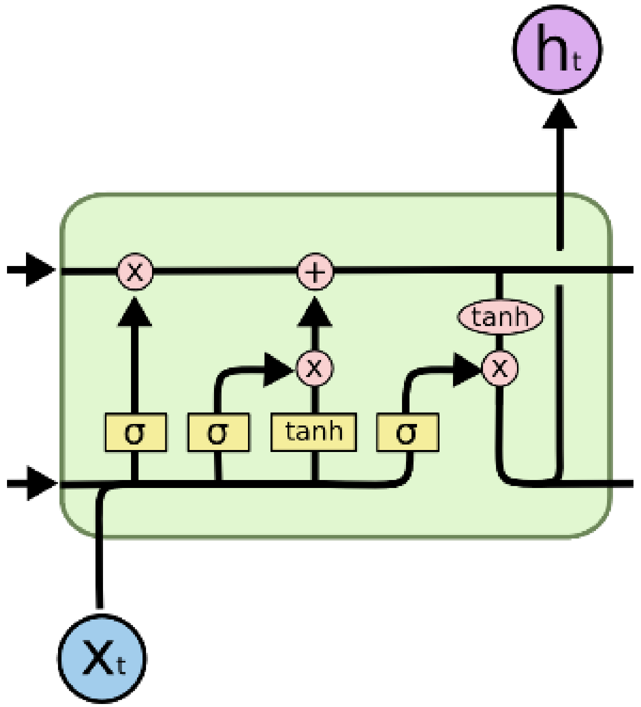
LSTM



- Output gate combines everything together

[blog]

Again, just to get the idea …

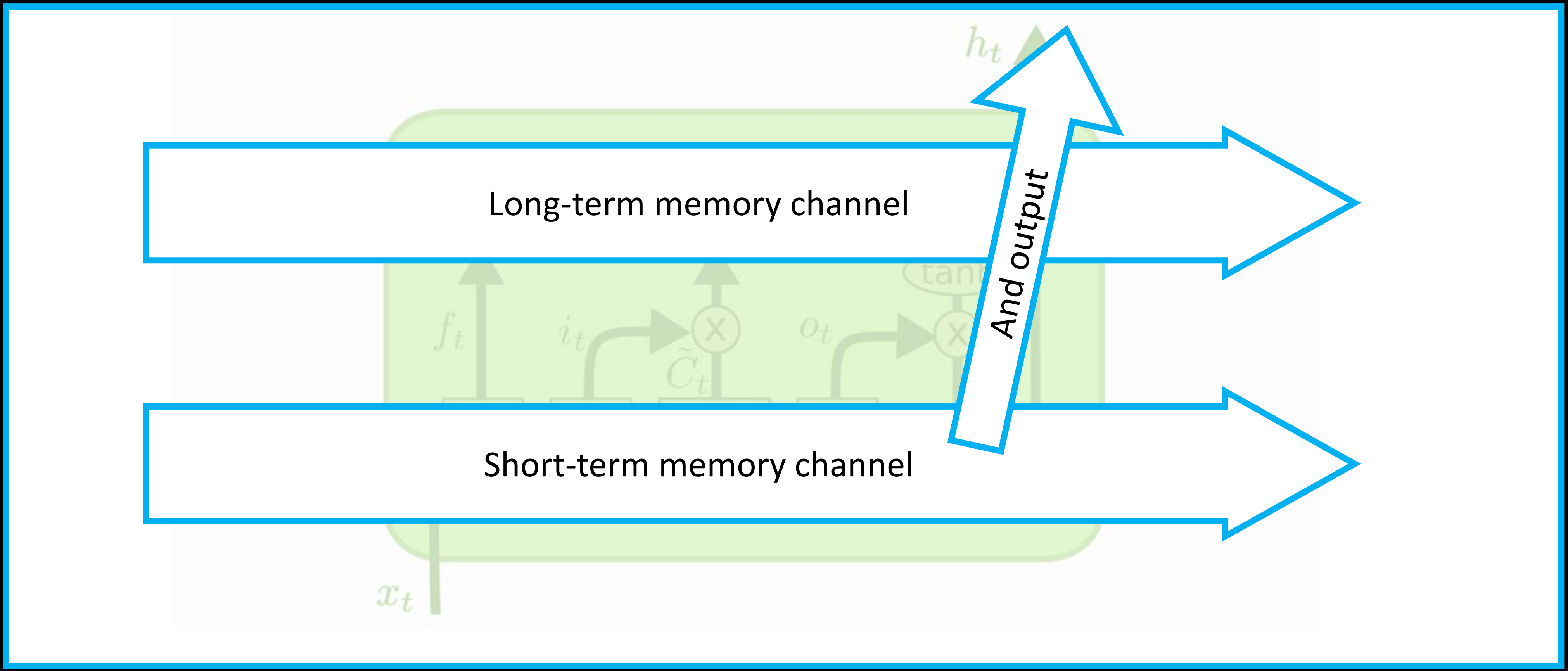


$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f)\\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i)\\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o)\\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c)\\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t\\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

- These operations are influenced by learned parameters (W,U,b)

[blog]

LSTM summarized

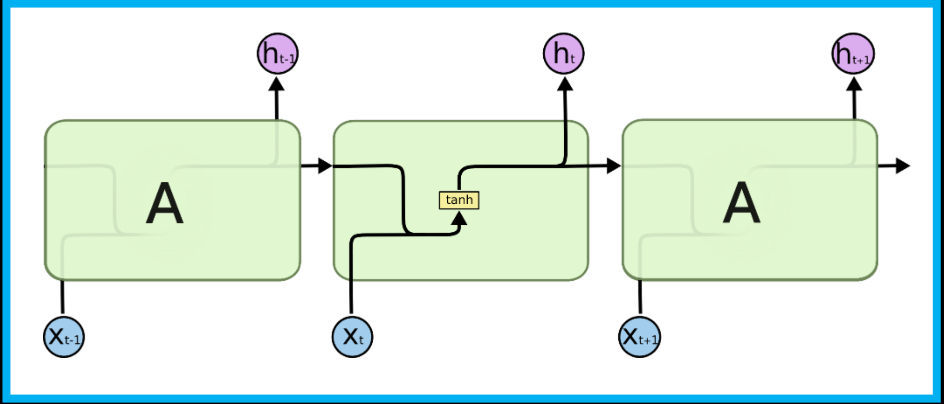


- In addition to the basic RNN we have a special channel for long-term mem.

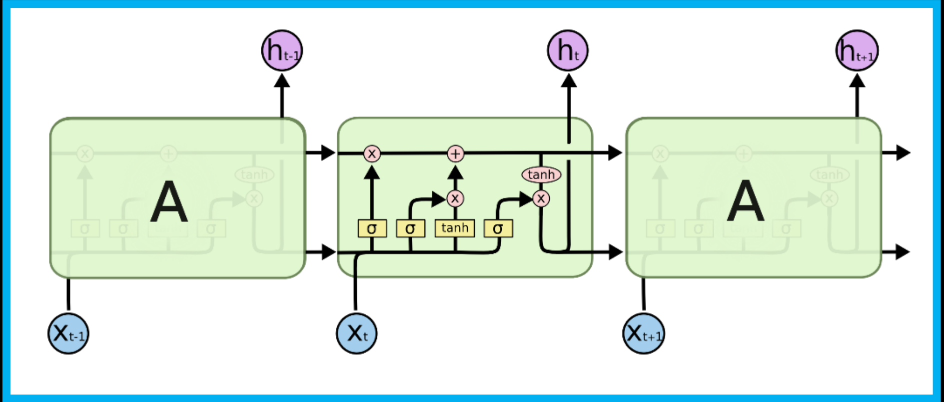
[blog]

In-depth look: RNNs and LSTMs

- **Recurrent Neural Networks** (RNN):
  - Simpler unit design



- **Long-Short Term Memory** (LSTM):
  - More complex unit design, made to **remember longer dependencies** inside the data



^ forget and input gates controlled by learned parameters (so it has more parameters)

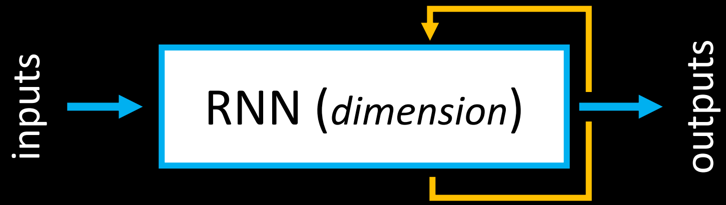
"I grew up in France… I speak fluent French."

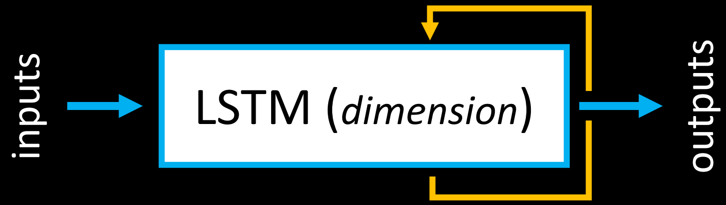
In code ?

```
from tensorflow.keras import layers

model.add(layers.RNN(128))

model.add(layers.LSTM(128))
```

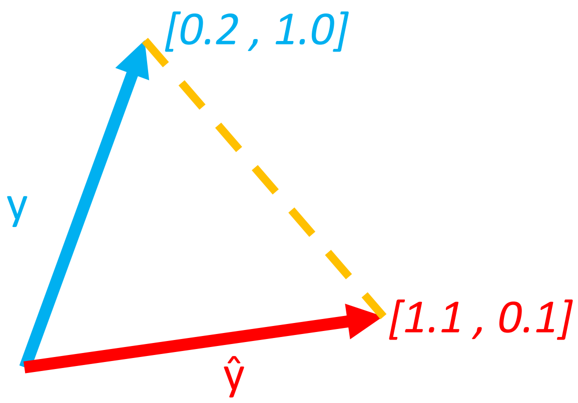




- In code we set up the **dimension** of the data flowing through these models – that’s the size of the vectors **h** and **C**

Loss functions

- We want to measure the **distance** between two vectors (typically this is between **predictions** and **labels**):



Vectors with n dimensions

Mean Squared Error (MSE):

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

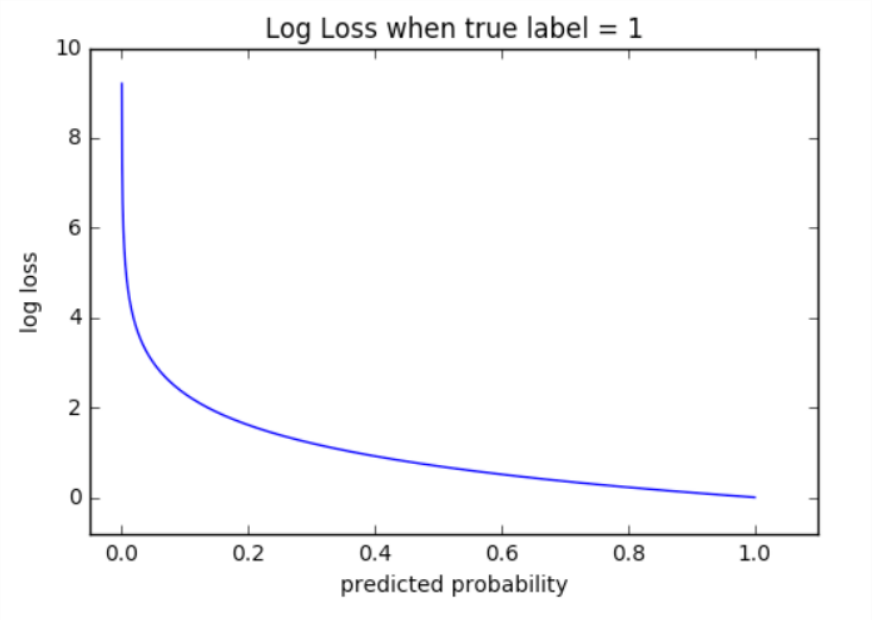
```python
def MSE(yHat, y):
        return np.sum((yHat - y)**2) / y.size
```

^ Used for regression problems

Cross-Entropy loss

Cross-Entropy

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.



$$CrossEntropyLoss = -(y\log(p) + (1-y)\log(1-p))$$

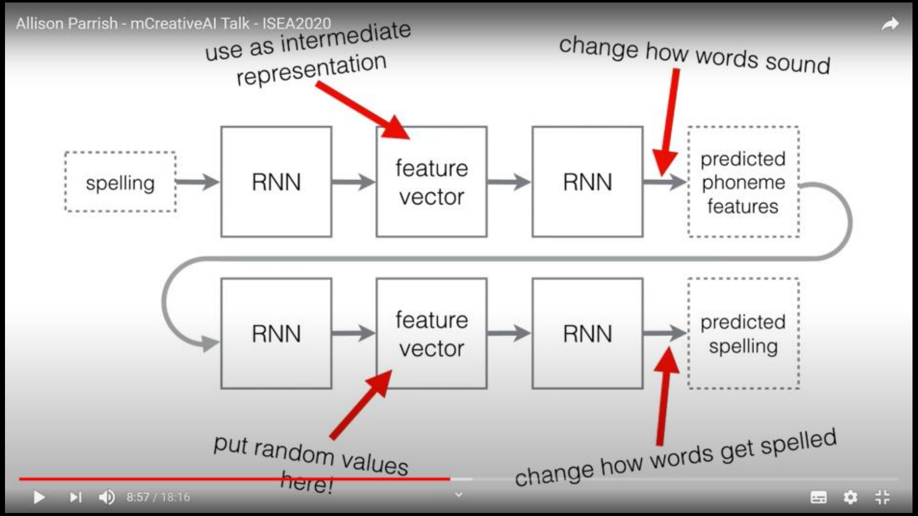
- PS: Label **y** can only be 0 or 1 … so only one of these two elements is evaluated:

```
def CrossEntropy(p, y):
 if y == 1:
   return -log(p)
 else:
   return -log(1 - p)
```

^ Used for classification problems

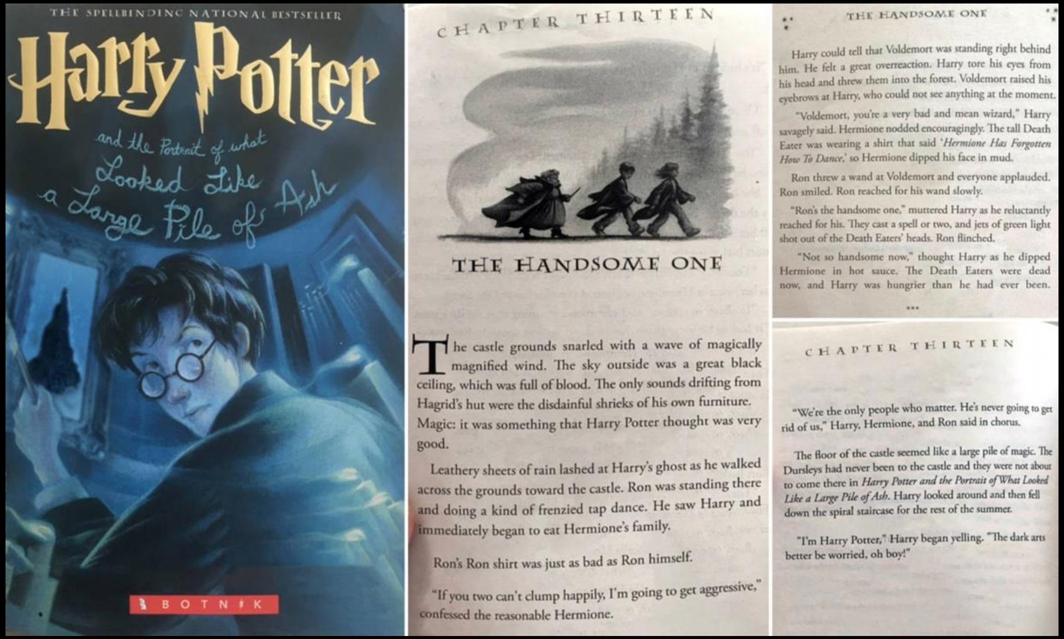
Some examples

- Generative **text**



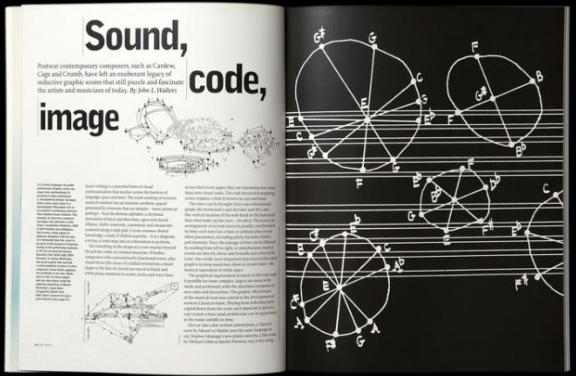
Talk by Allison Parrish at mCreativeAI, ISEA2020: www.youtube.com/watch?v=xQvbM5iRXp8

- Phoneme and graphene for NaNoGenMo
- Generating font that almost look legible



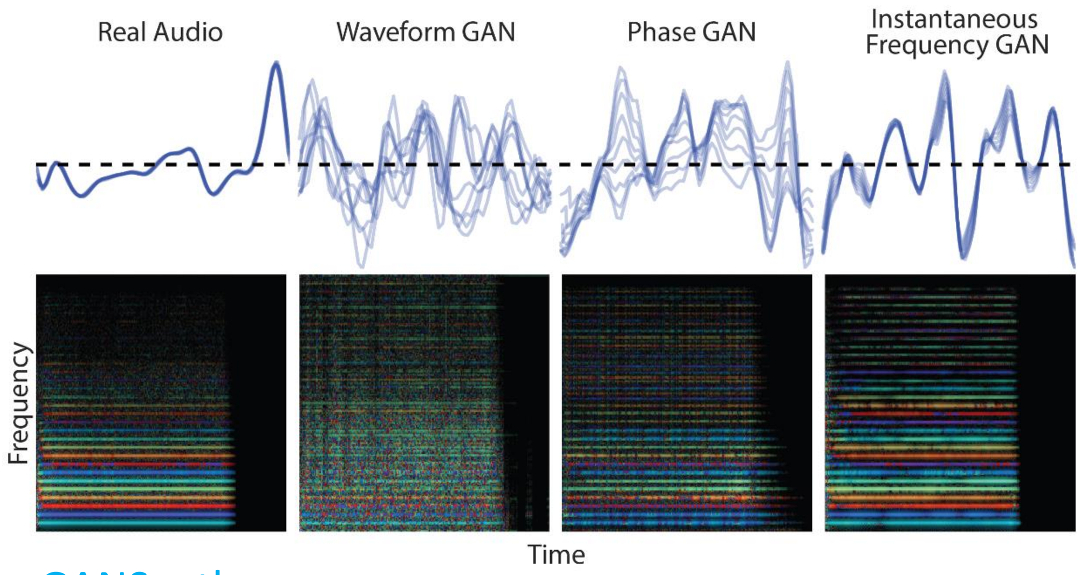
Harry Potter and the Portrait of what Looked Like a Large Pile of Ash, Botnik Studios

Some examples



~ John Cage

- Generative **music**



[GANSynth]

[notes example]

Raw data / spectrogram based vs. note based

Some examples

- Generative **images**

  - Uses **pix2pix** similarly to the works you probably already seen by Mario Klingemann or by Memo Akten

Video: youtube.com/watch?v=lr59AhOPgWQ
Blog: https://magenta.tensorflow.org/nfp_p2p

Predicting the future, one frame at a time, Damien Henry, 2017

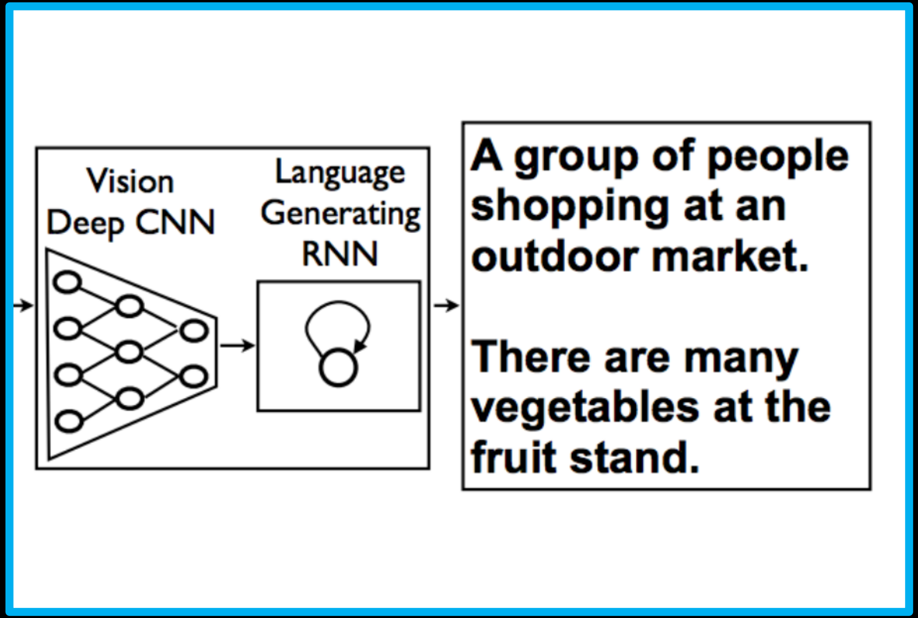
Note

youtube.com/watch?v=iDlWdOLGtWY

Note

youtube.com/watch?v=iDlWdOLGtWY

Non-traditional model designs



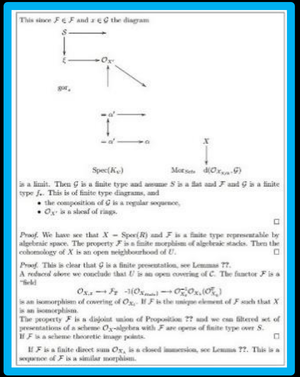
Video: youtube.com/watch?v=8BFzu9m52sc

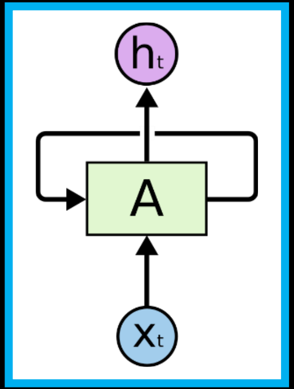
NeuralTalk, A. Karpathy, 2015

Links and additional readings:

- **Bonus readings:**
  - **Andrej Karpathy blog** “The Unreasonable Effectiveness of Recurrent Neural Networks” – rnn-effectiveness
    - Citation: *There’s something magical about Recurrent Neural Networks (RNNs). … This post is about sharing some of that magic with you. We’ll train RNNs to generate text character by character and ponder the question “how is that even possible?”*
  - **Blog** “Understanding LSTM Networks” - Understanding-LSTMs

- **Code samples:**
  - **WordRNN**: Working repository for training and using a multilayer RNN and LSTM models – word-rnn-tensorflow
  - *(**PS: Text analysis**: Gensim library examples)*





End of the lecture

***)** PS: follows material for the practical session …