Sequence-to-sequence models

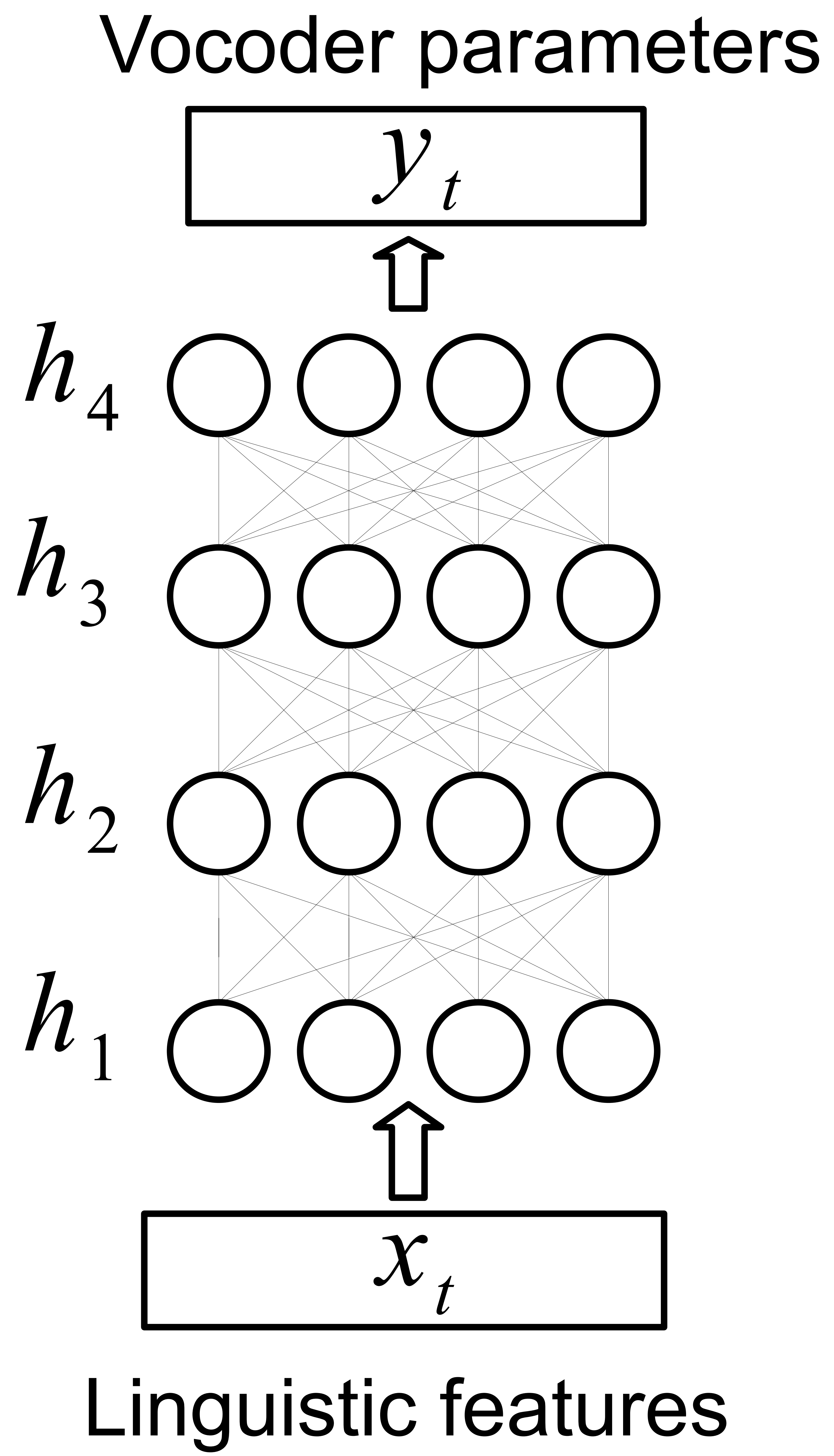
- Class slides

What you should already know

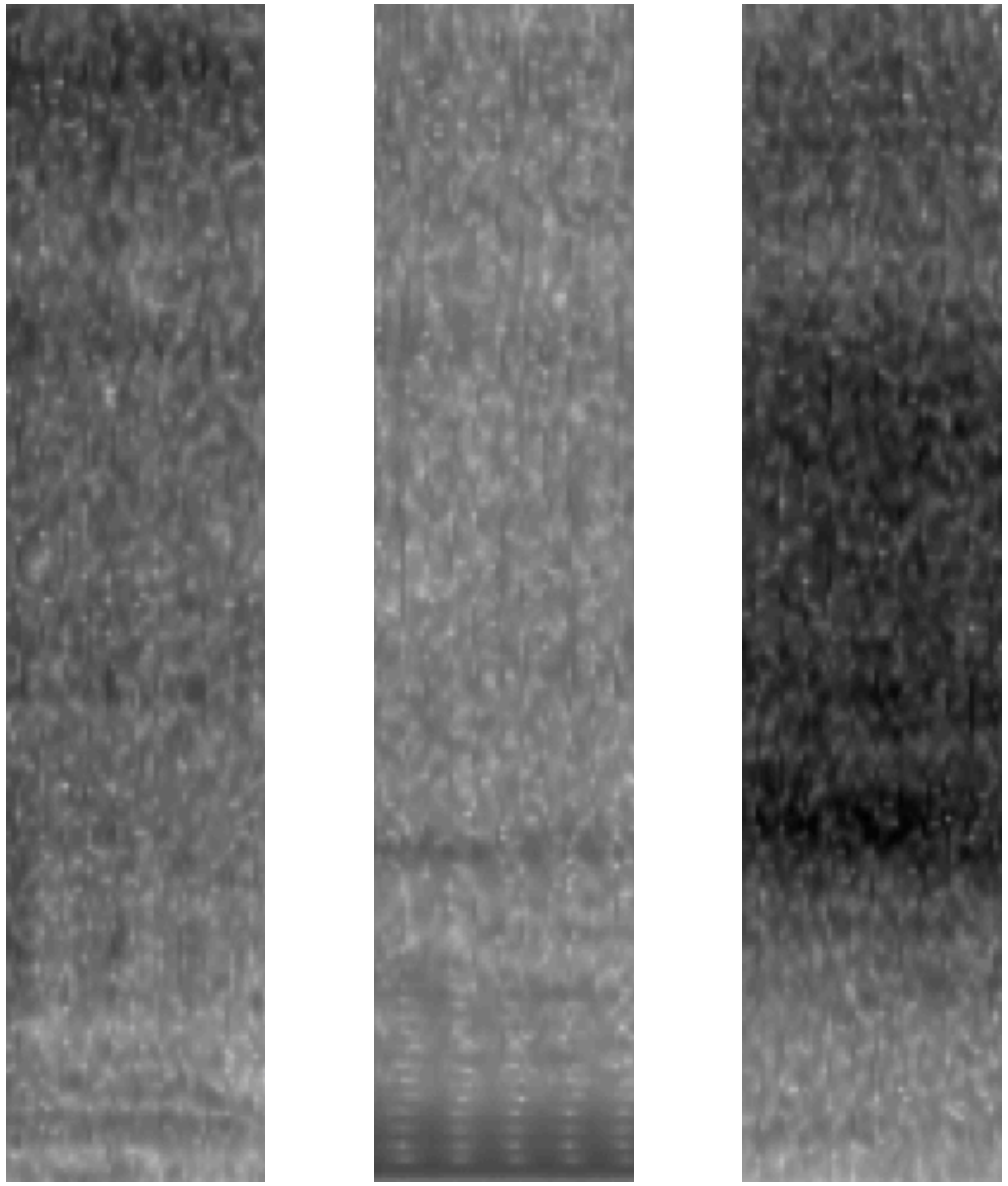
- Converting the linguistic specification into a form suitable for input to DNN
- The input is now simply a sequence of vectors
- Simple Deep Neural Network maps one input vector to one output vector

Recap

Doing TTS with a DNN

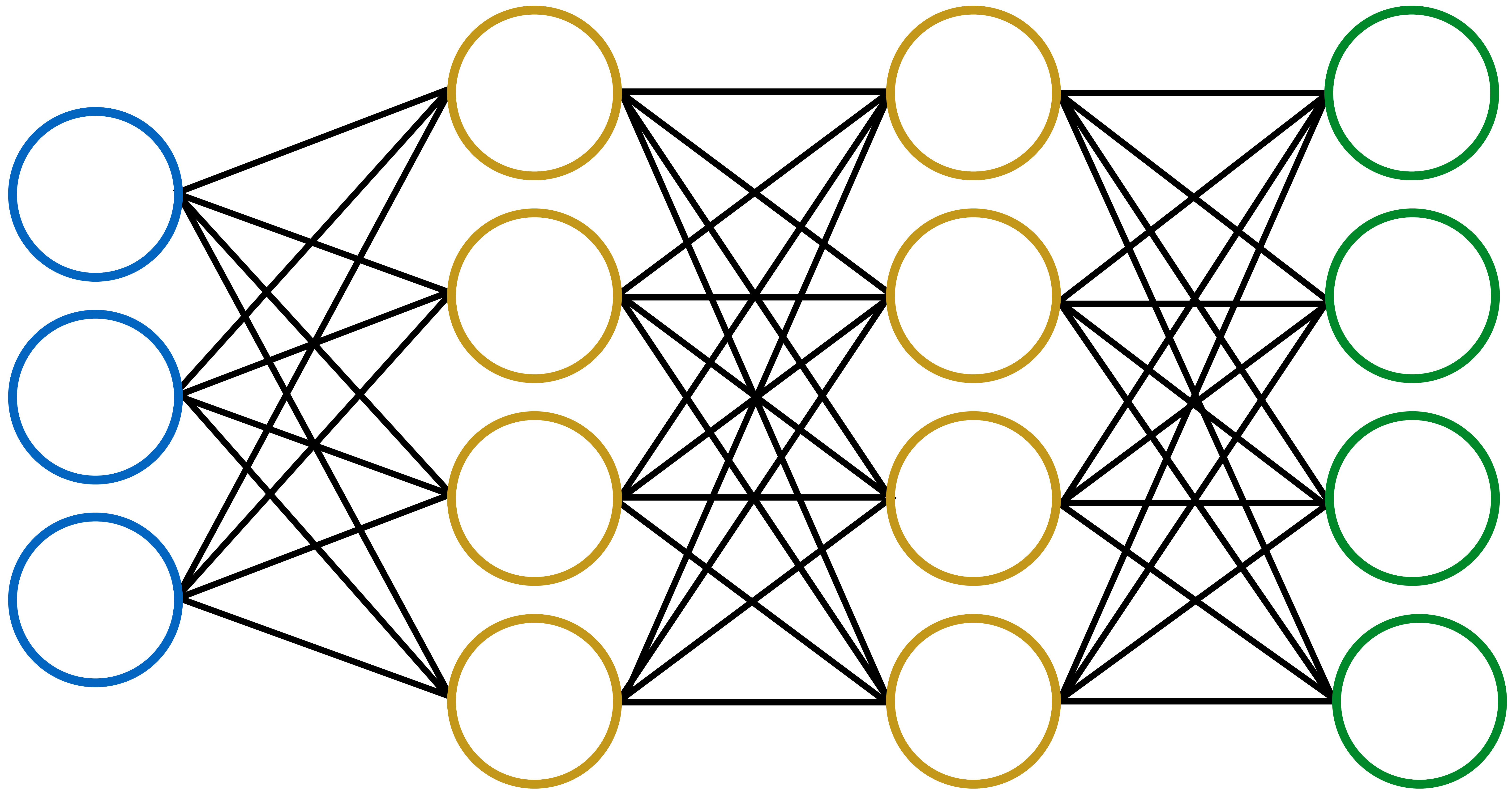


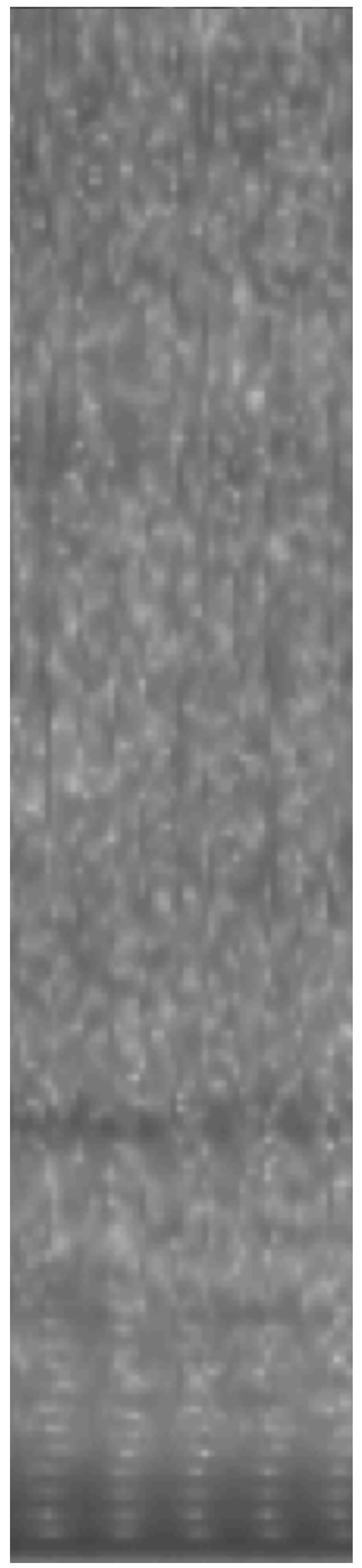
θ ð ʃ



Doing regression by performing a forward pass through the DNN

$\hat{\theta}$

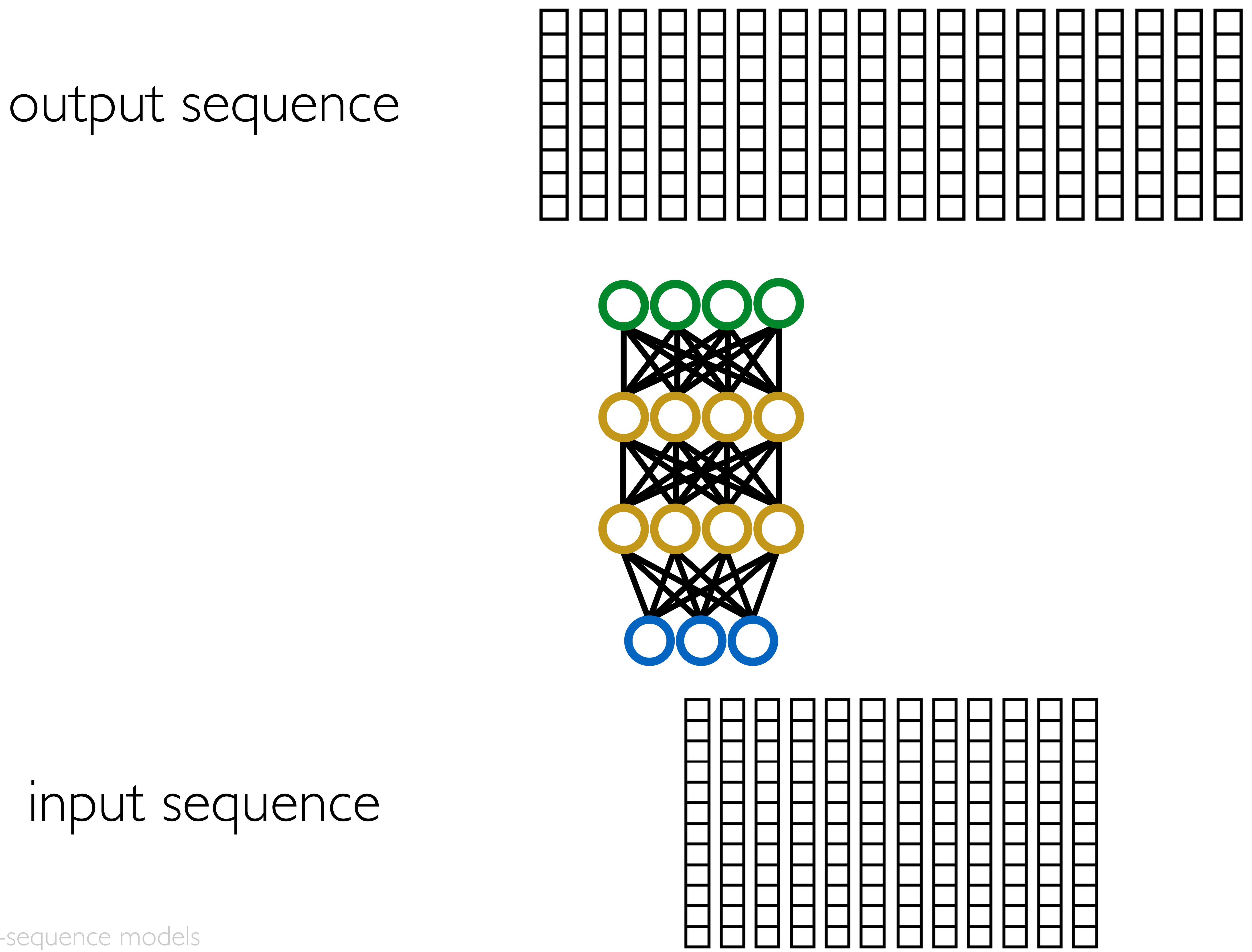




Terminology

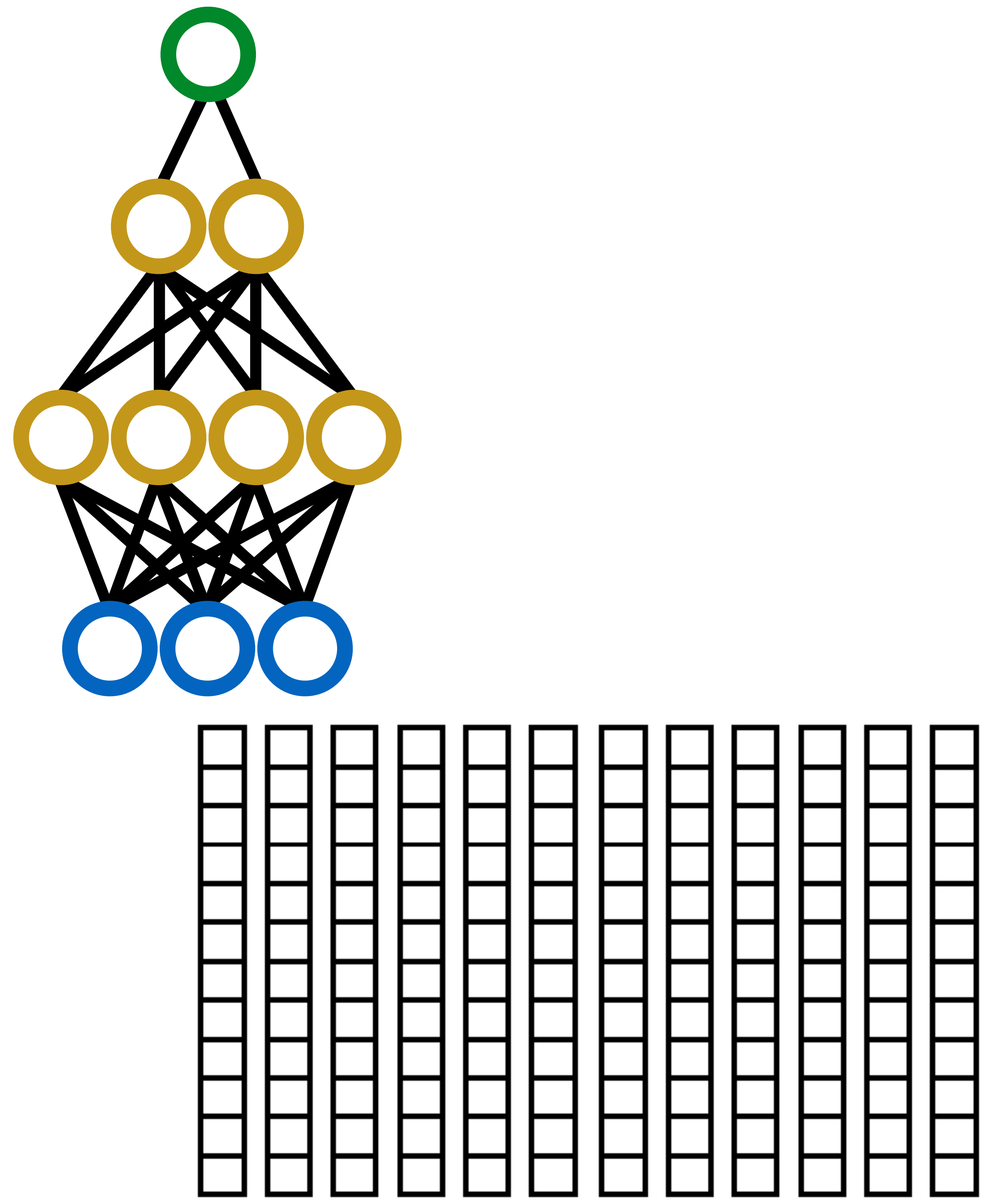
- regression
- inference
- forward pass

Sequence-to-sequence regression using a DNN - dealing with duration



Sequence-to-sequence regression using a DNN - dealing with duration

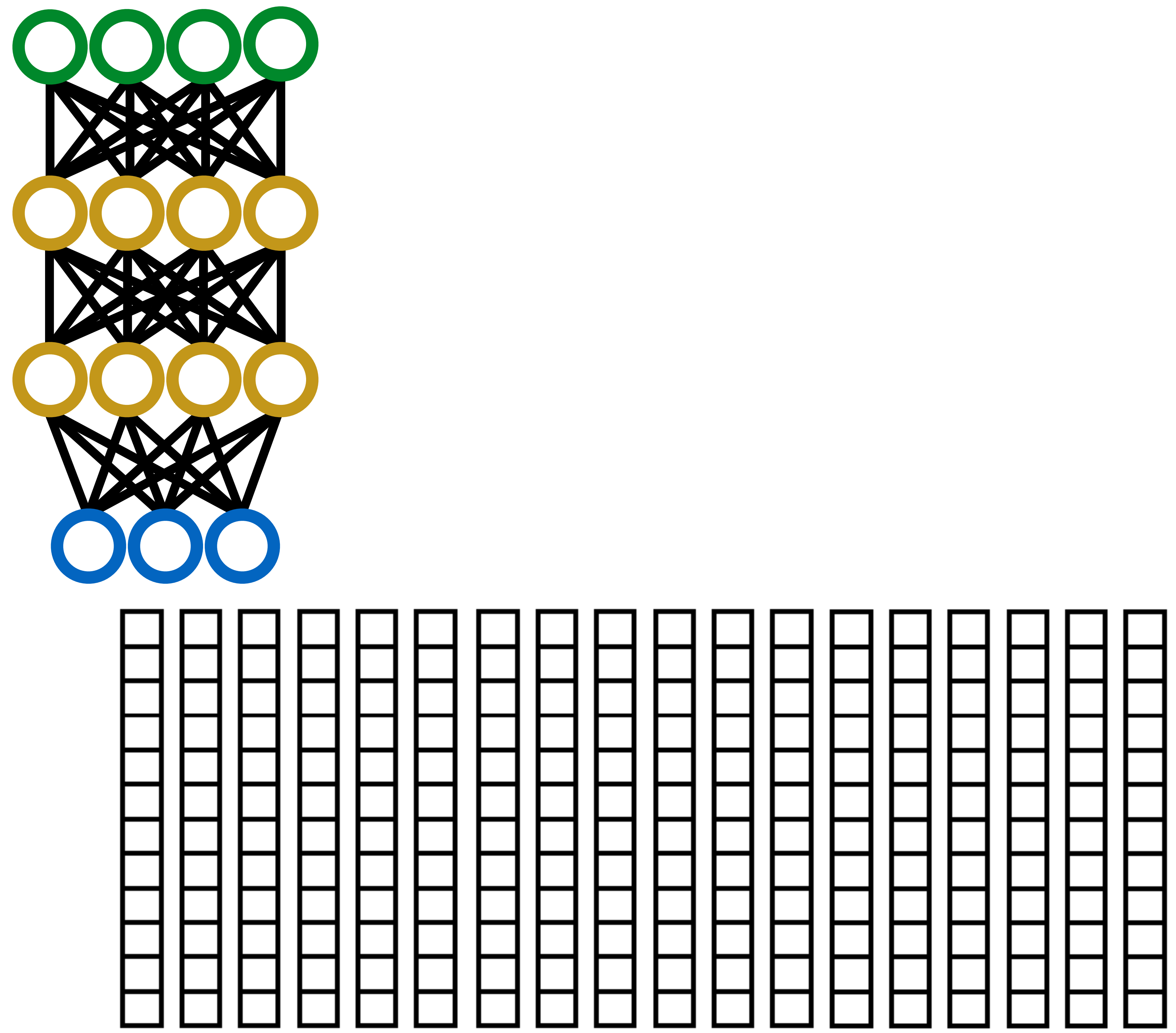
upsampled input sequence



input sequence

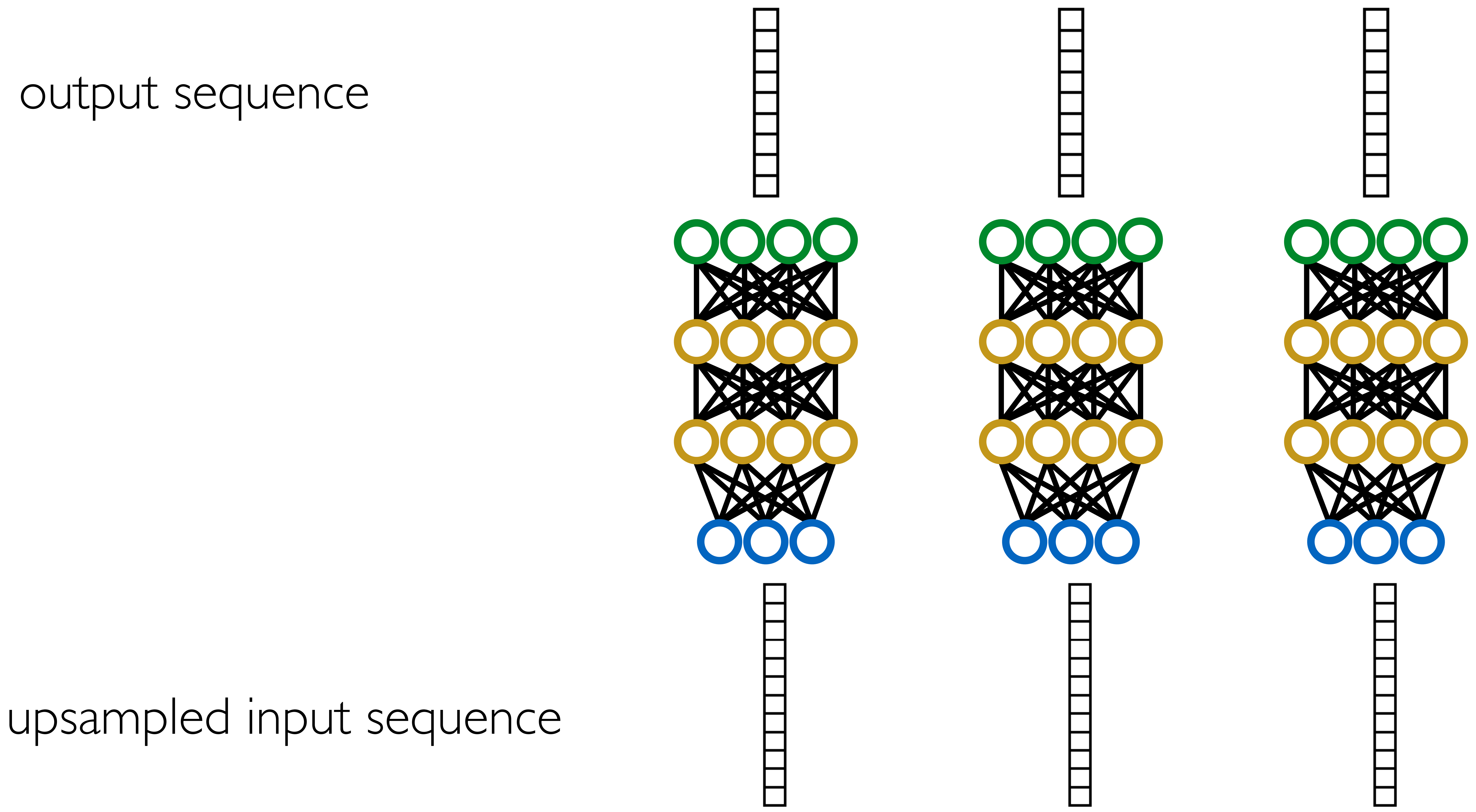
Sequence-to-sequence regression using a DNN - dealing with duration

output sequence



upsampled input sequence

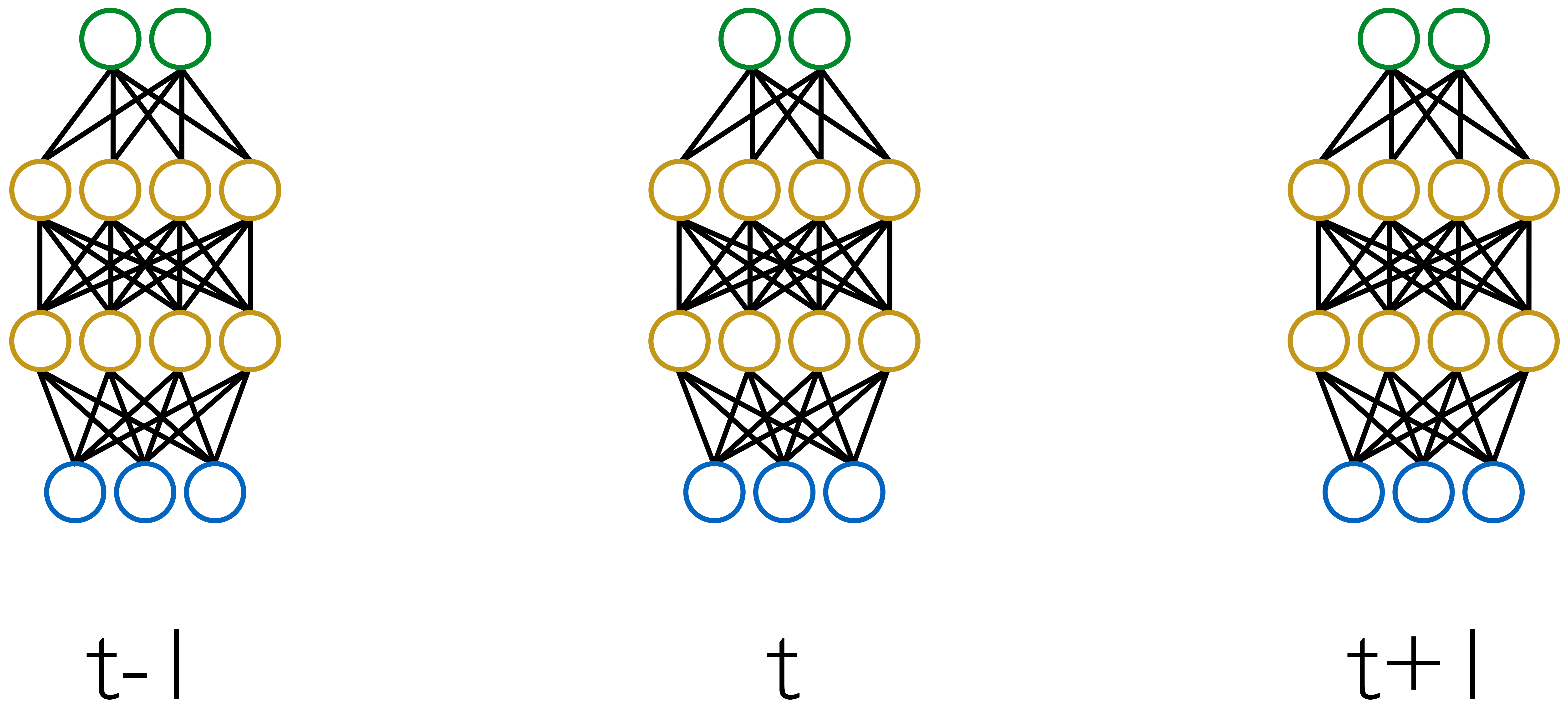
Processing the entire sequence at once = duplicate model for every time step



Terminology

- time step

Limitations of processing each time step independently



Limitations of processing each time step independently

- Input features
  - Requires assembling all necessary contextual information and placing at current input
  - Features pre-determined using knowledge-driven feature engineering (e.g., quinphones)
- Duration
  - Must be handled separately
- Sequence modelling
  - A constant regression function, time-independent, memoryless
- Output features
  - Predicted using only the input features
  - Output is conditionally-independent of previous/next outputs, given the current input

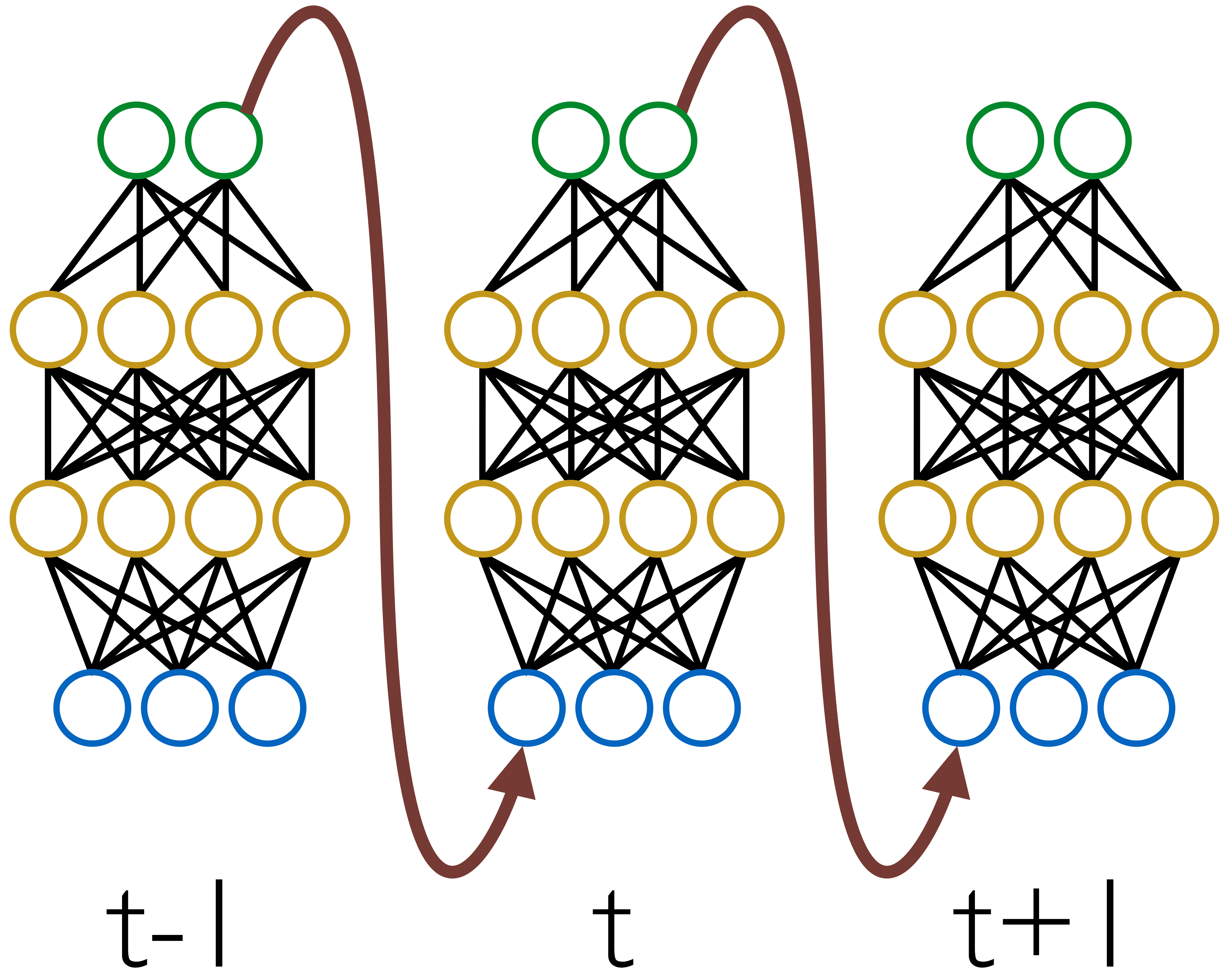
Things to improve next

- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

Recurrent (naive version)

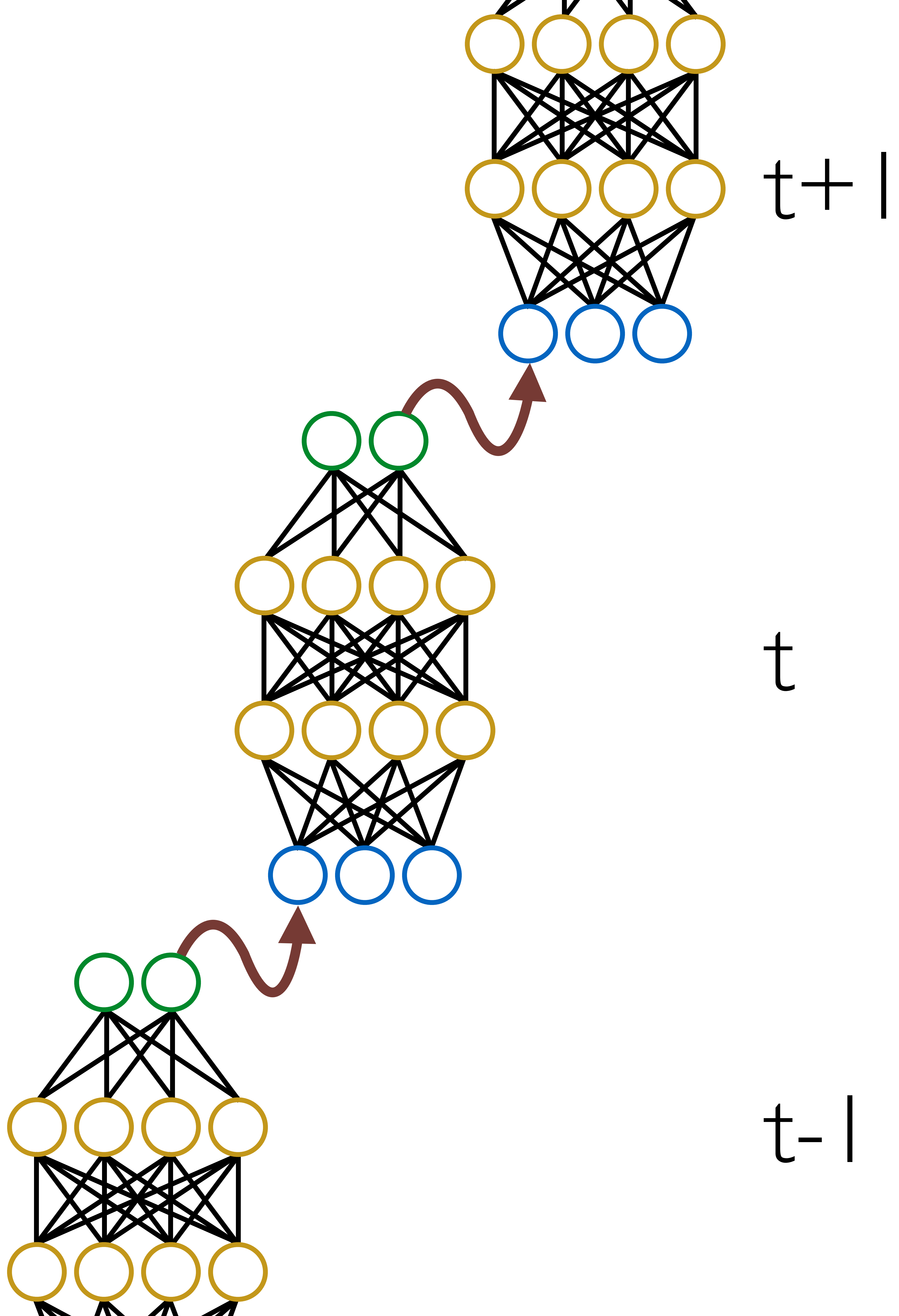
- Pass some of the outputs (or hidden layer activations) forwards in time, typically to the next time step
- A kind of **memory**
- Provides "infinite" left context

- (could also pass information backwards in time)



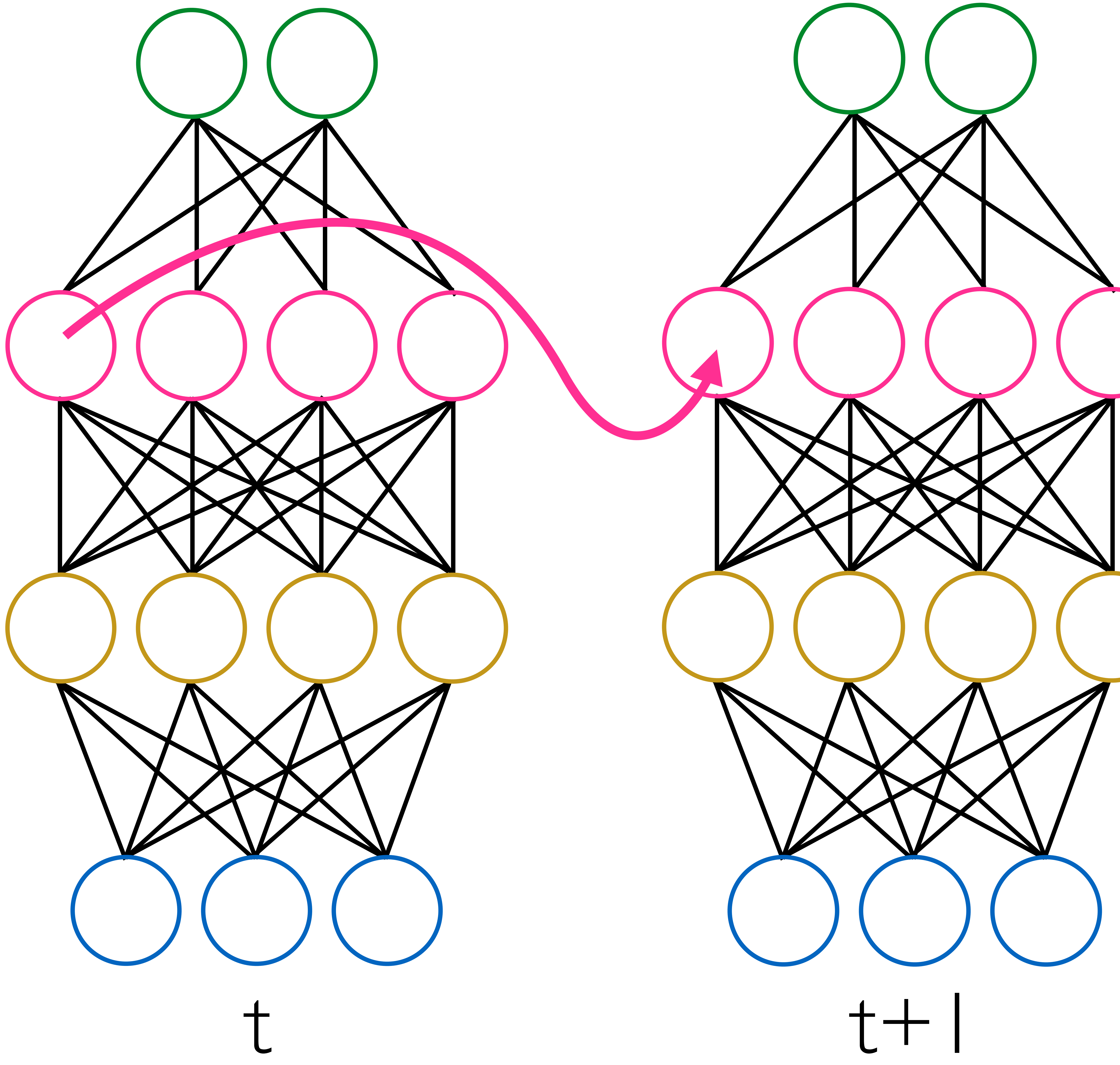
Recurrent

- Simple recurrence is equivalent to a **very deep network**
- To train this network, we have to backpropagate the derivative of the the errors (the **gradient**) through all of the layers
  - "backpropagation through time"
- Suffers from the "**vanishing gradient**" problem, for long sequences



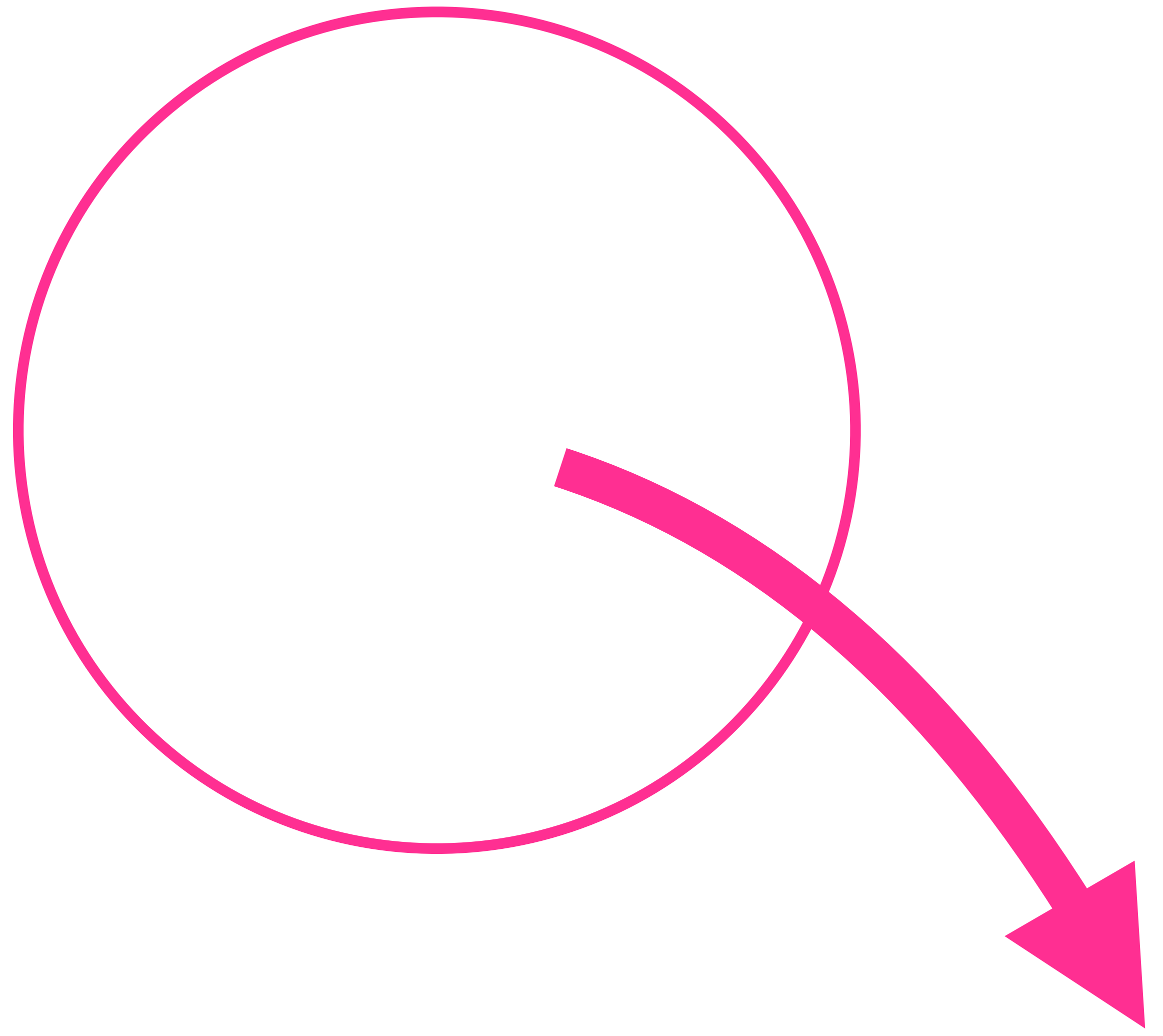
Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- <u>Conceptually</u>
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn when to **forget**



Long short-term memory
(a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- <u>Conceptually</u>
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn when to **forget**



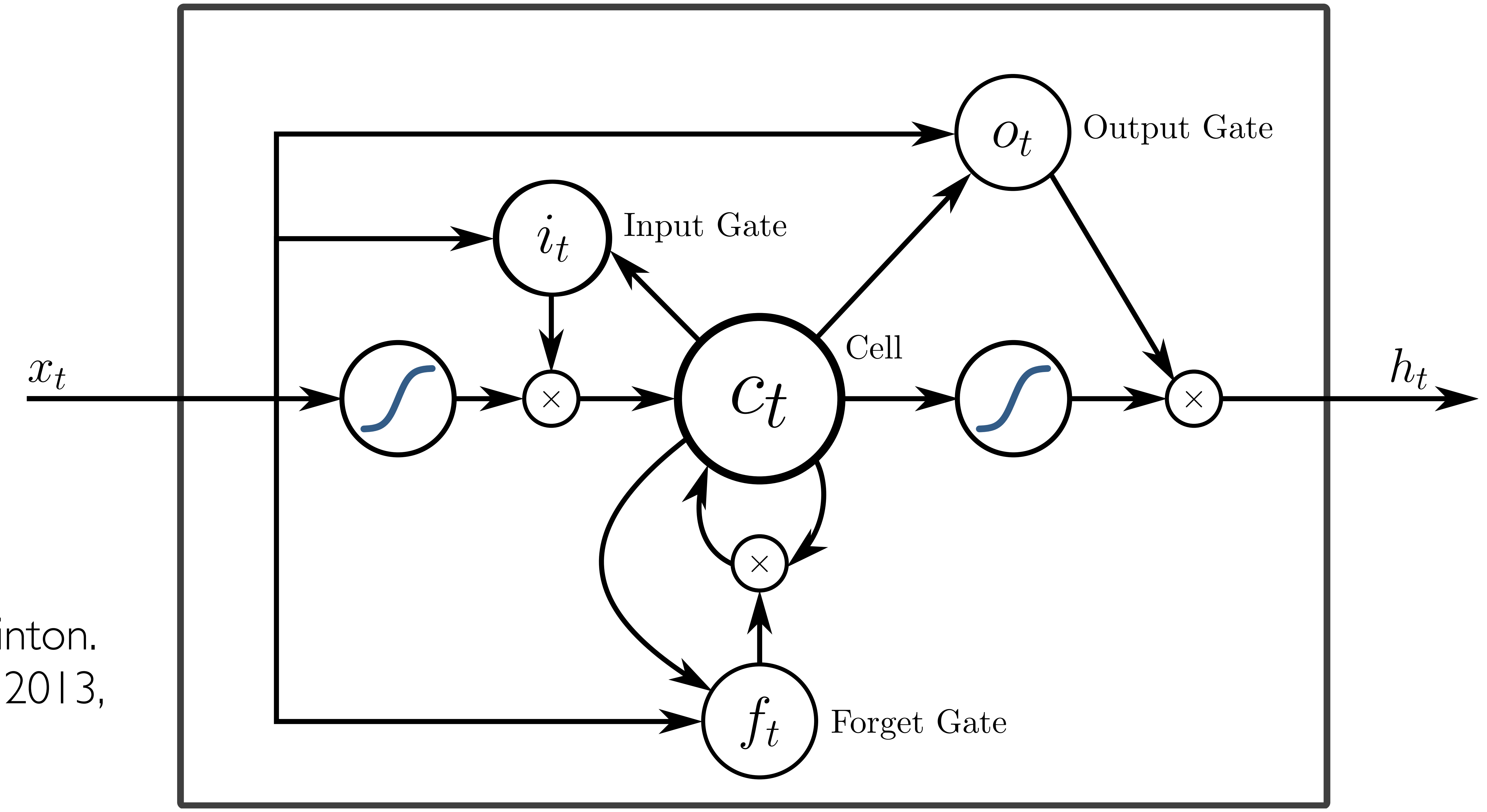
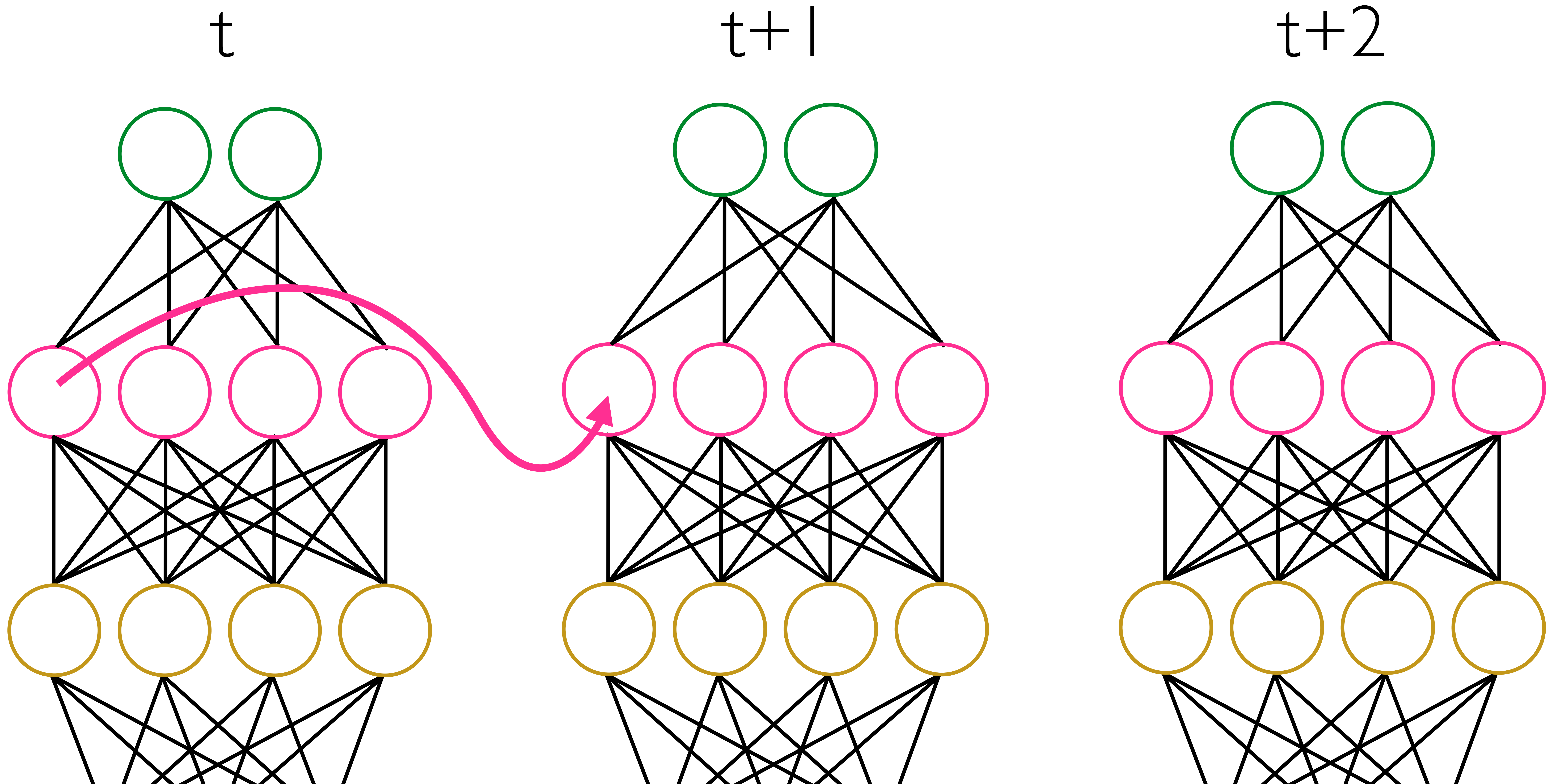
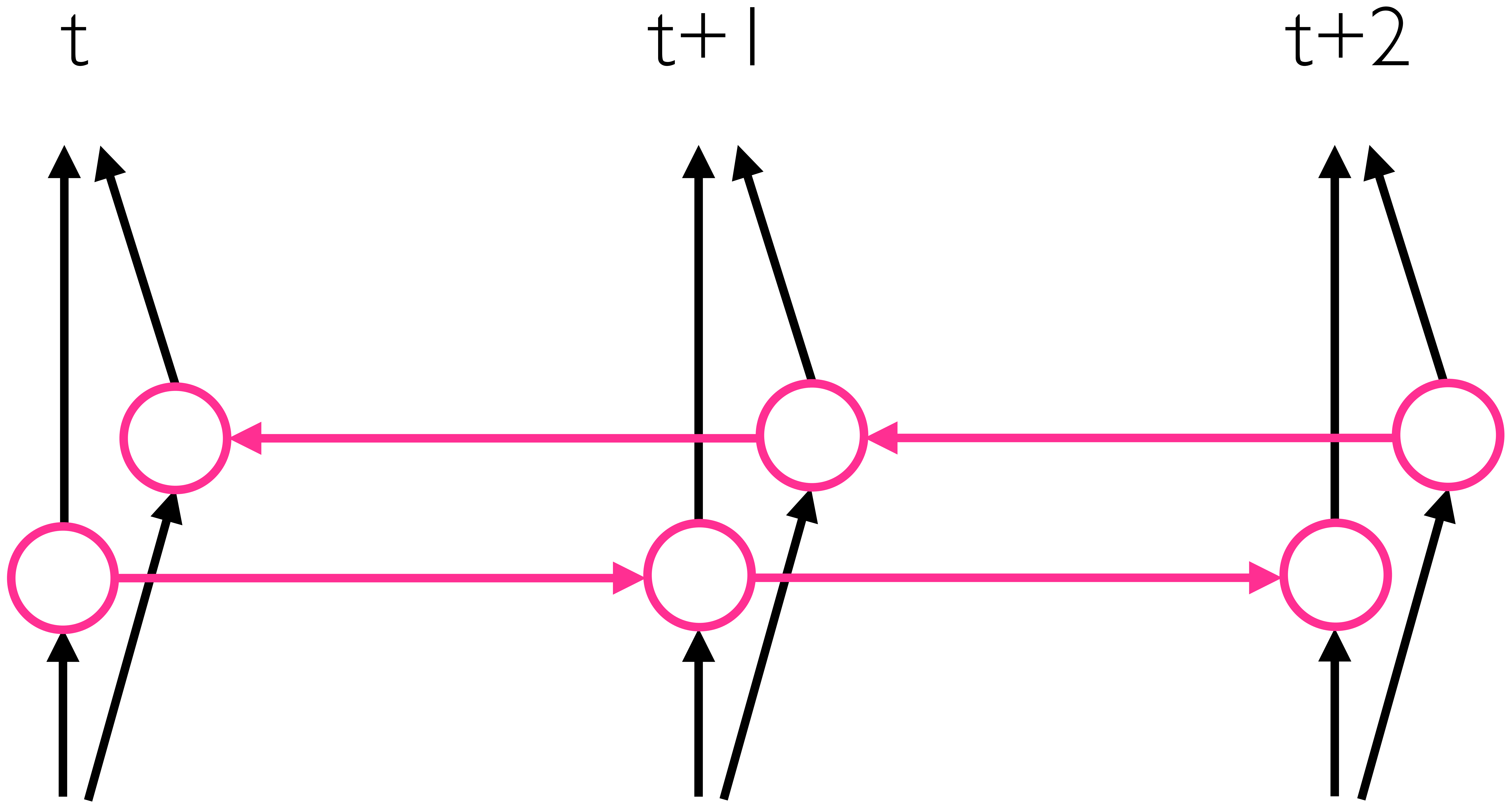


Figure from Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks" ICASSP 2013, redrawn as SVG by Eddie Antonio Santos

LSTM units & Gated Recurrent Units (GRUs)



Neural building blocks : (bidirectional) LSTM layer



Orientation

- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

- Feed-forward architecture
  - no memory
- "Simple" recurrent neural networks
  - vanishing gradient problem
- **LSTMs or GRUs** (which avoid the vanishing gradient problem)

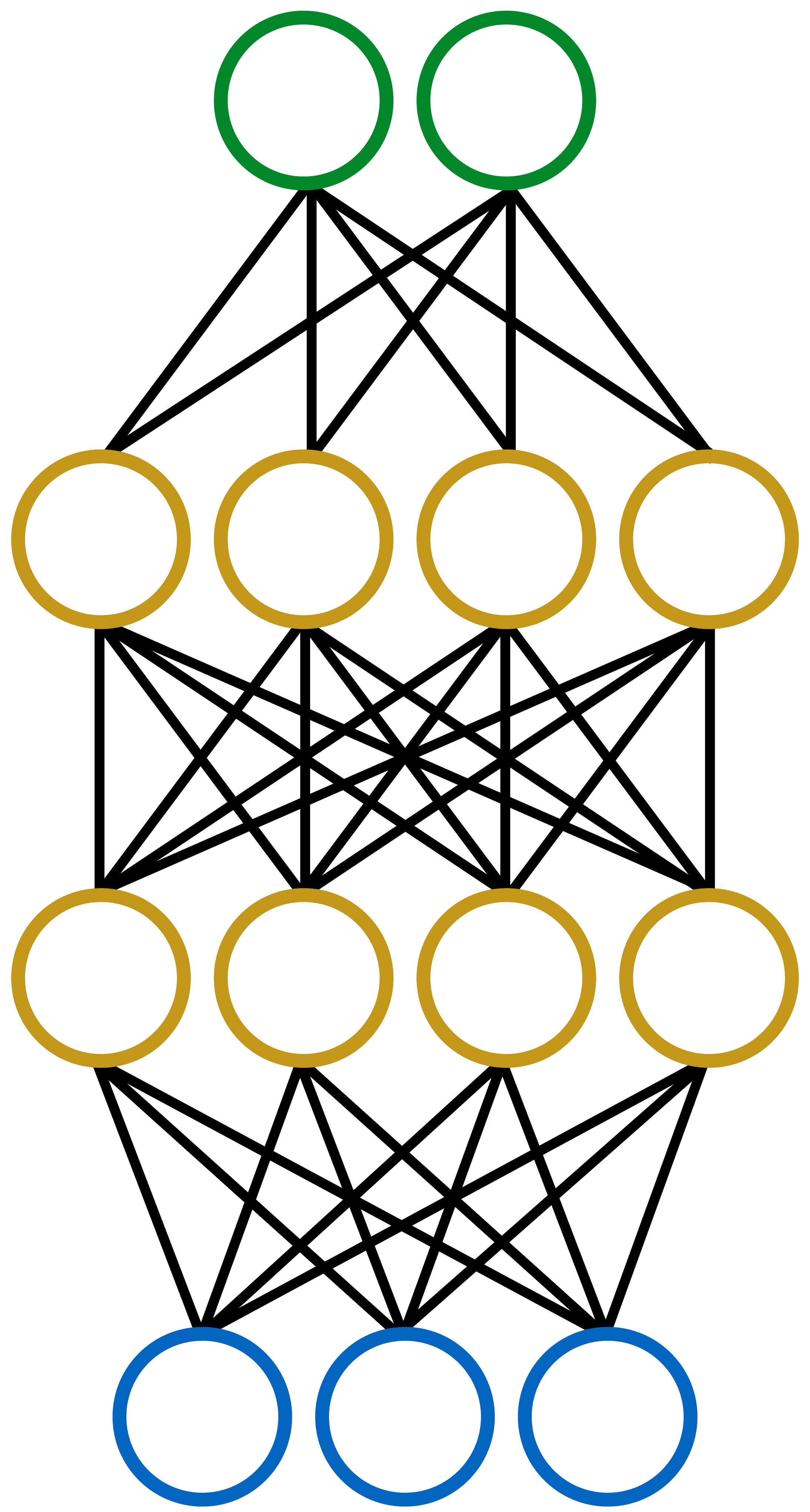
During training: alignment During inference: duration prediction

- Length of input sequence is generally **different** to length of output sequence

- For example
  - input: sequence of phones
  - output: acoustic frames (e.g., a spectrogram, to be input to a vocoder)

- <u>Conceptually</u>
  - **read** in the input sequence; **memorise** it using a **learned representation**
  - given that representation, **write** the output sequence

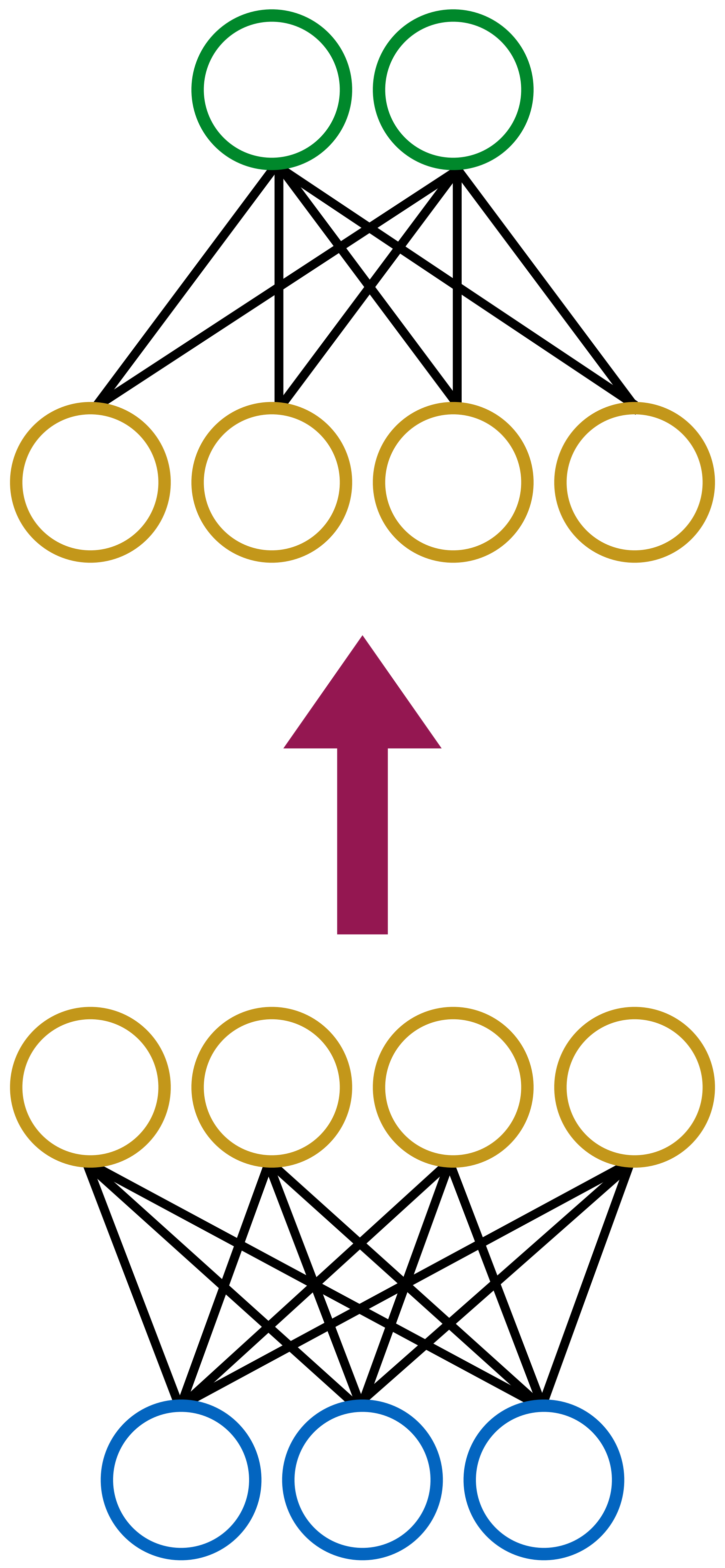
output time steps are frames (e.g., of a mel spectrogram)

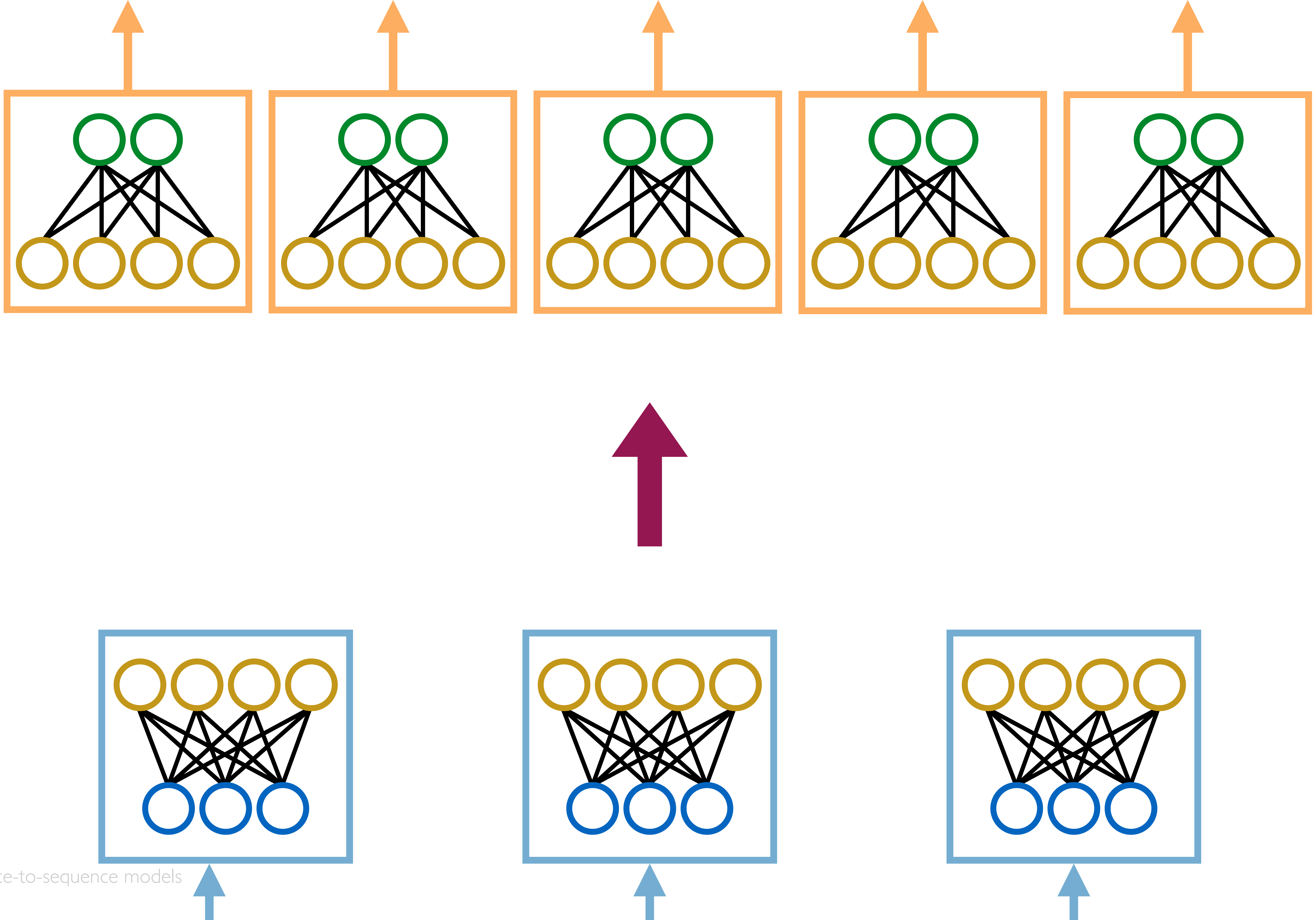


input time steps are linguistic units (e.g., phones)

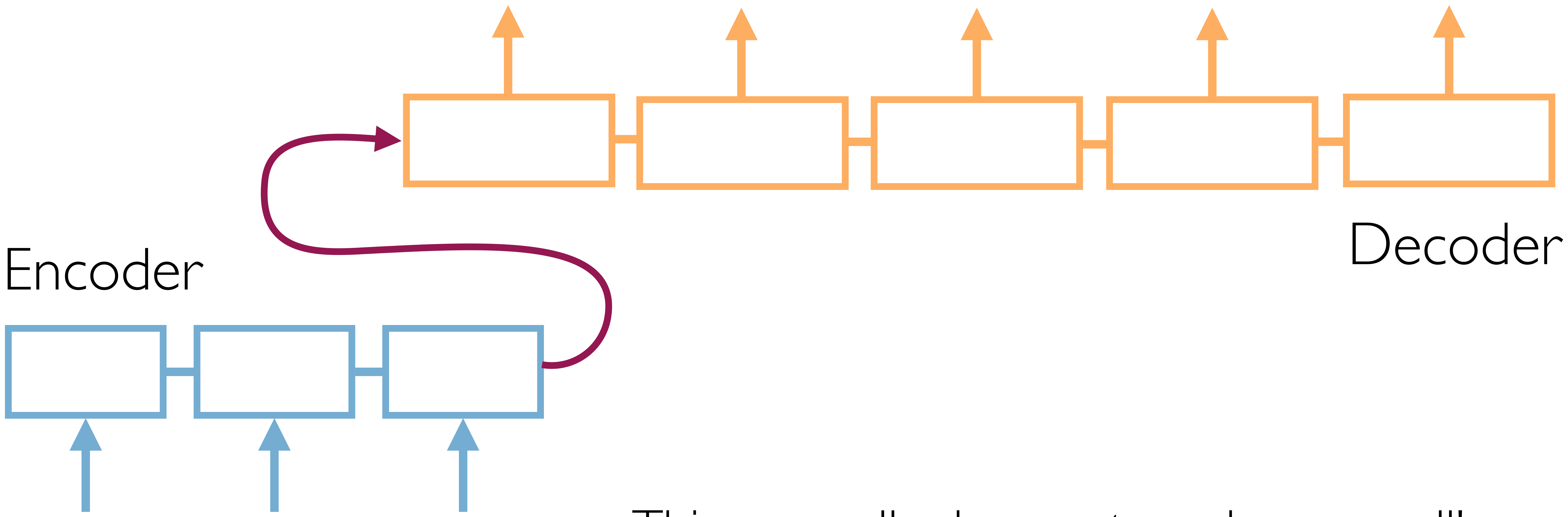
Decoder

Encoder

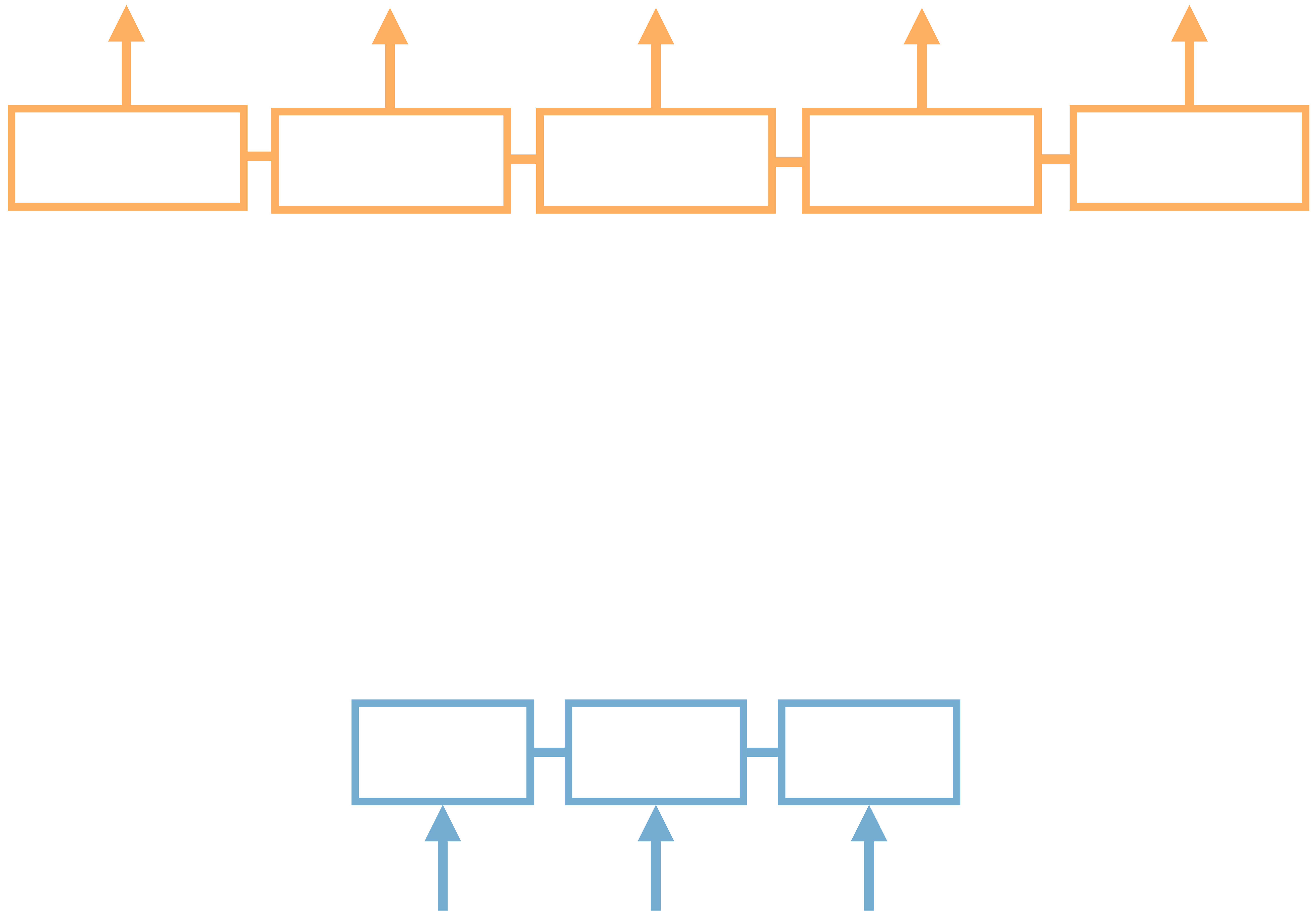


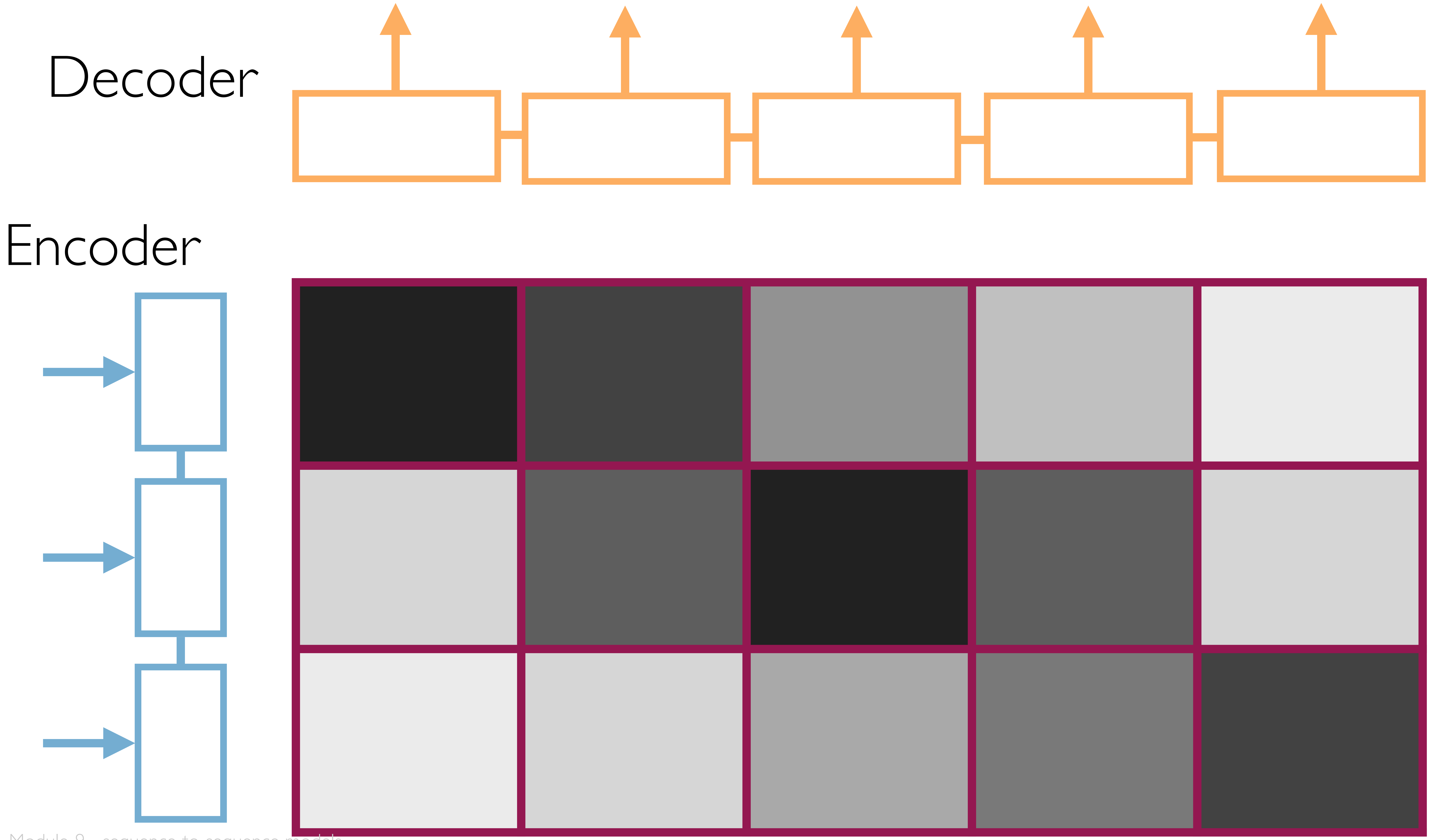


A sequence-to-sequence network using an encoder-decoder architecture

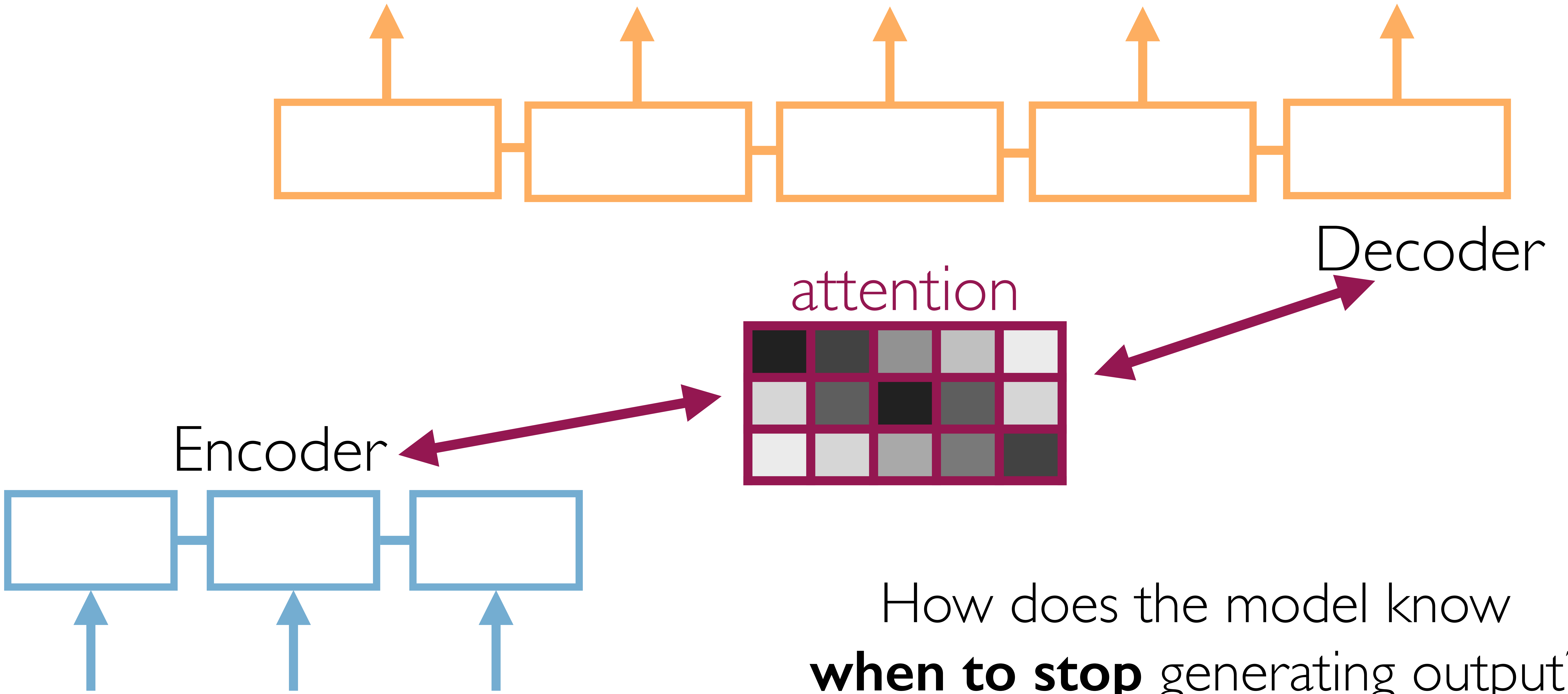


This generally does not work very well!
Why?




Decoder
Encoder

Encoder-decoder with attention



How does the model know **when to stop** generating output?

Terminology

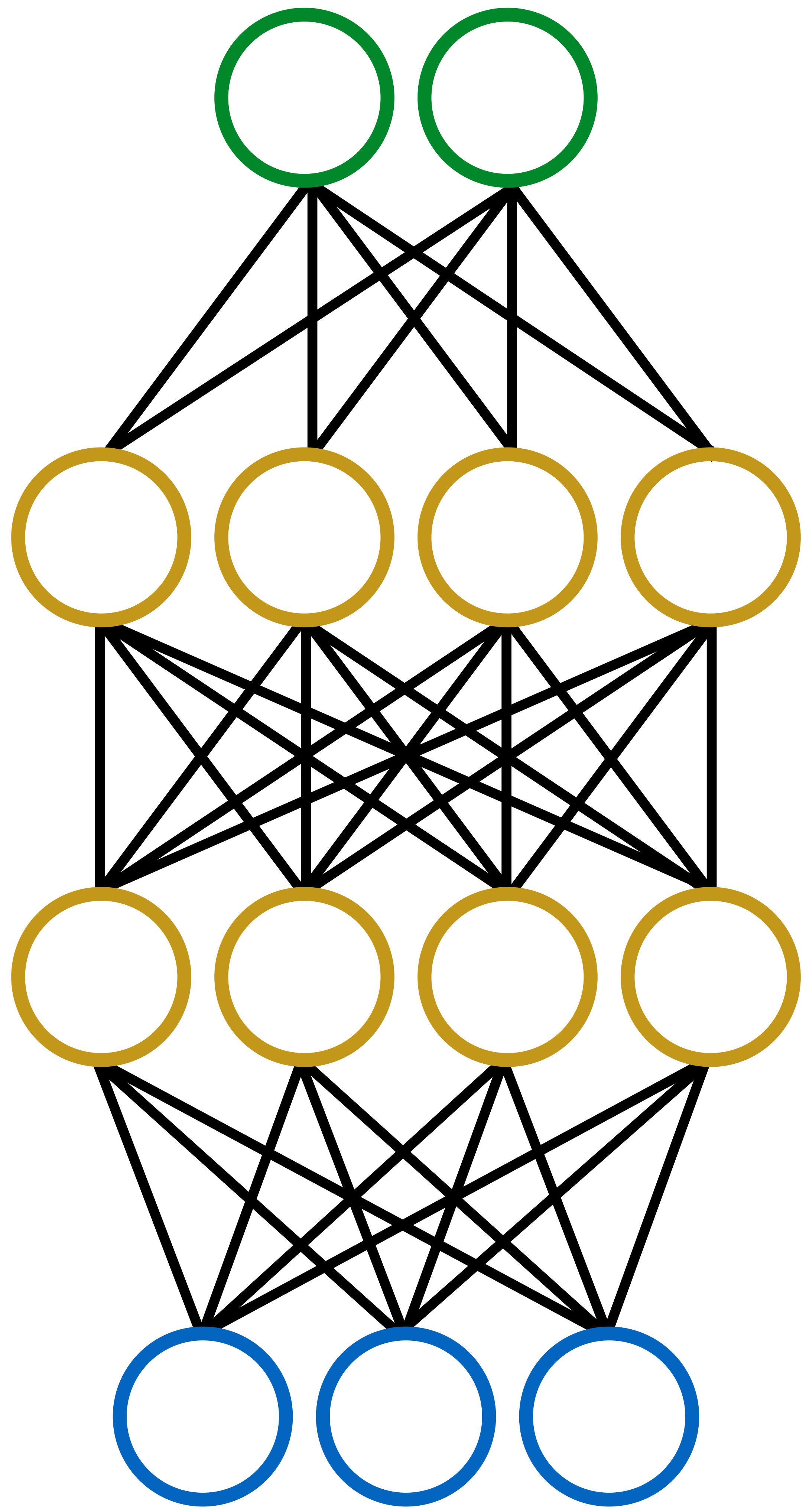
- encoder
- decoder
- attention

Orientation

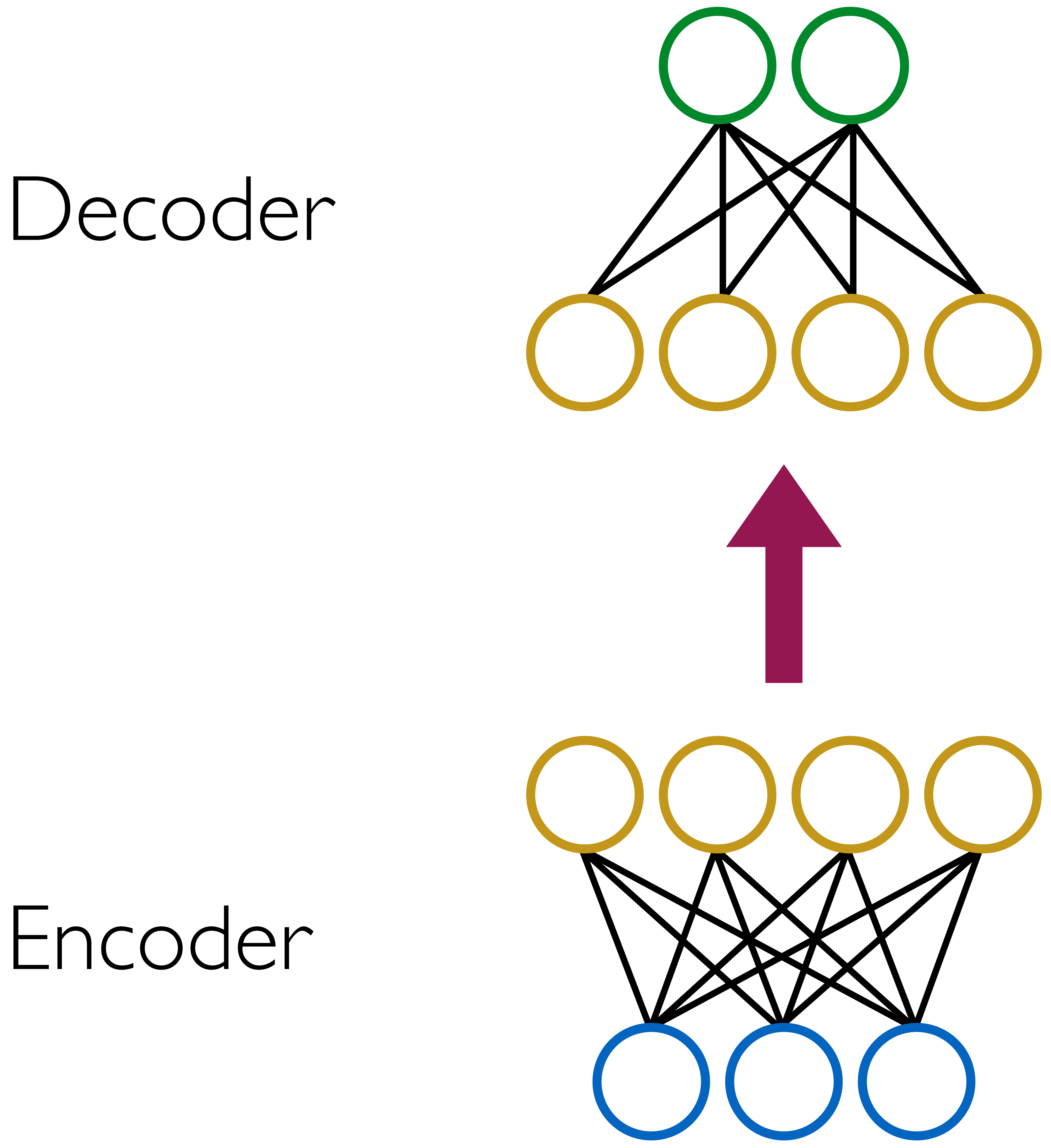
- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

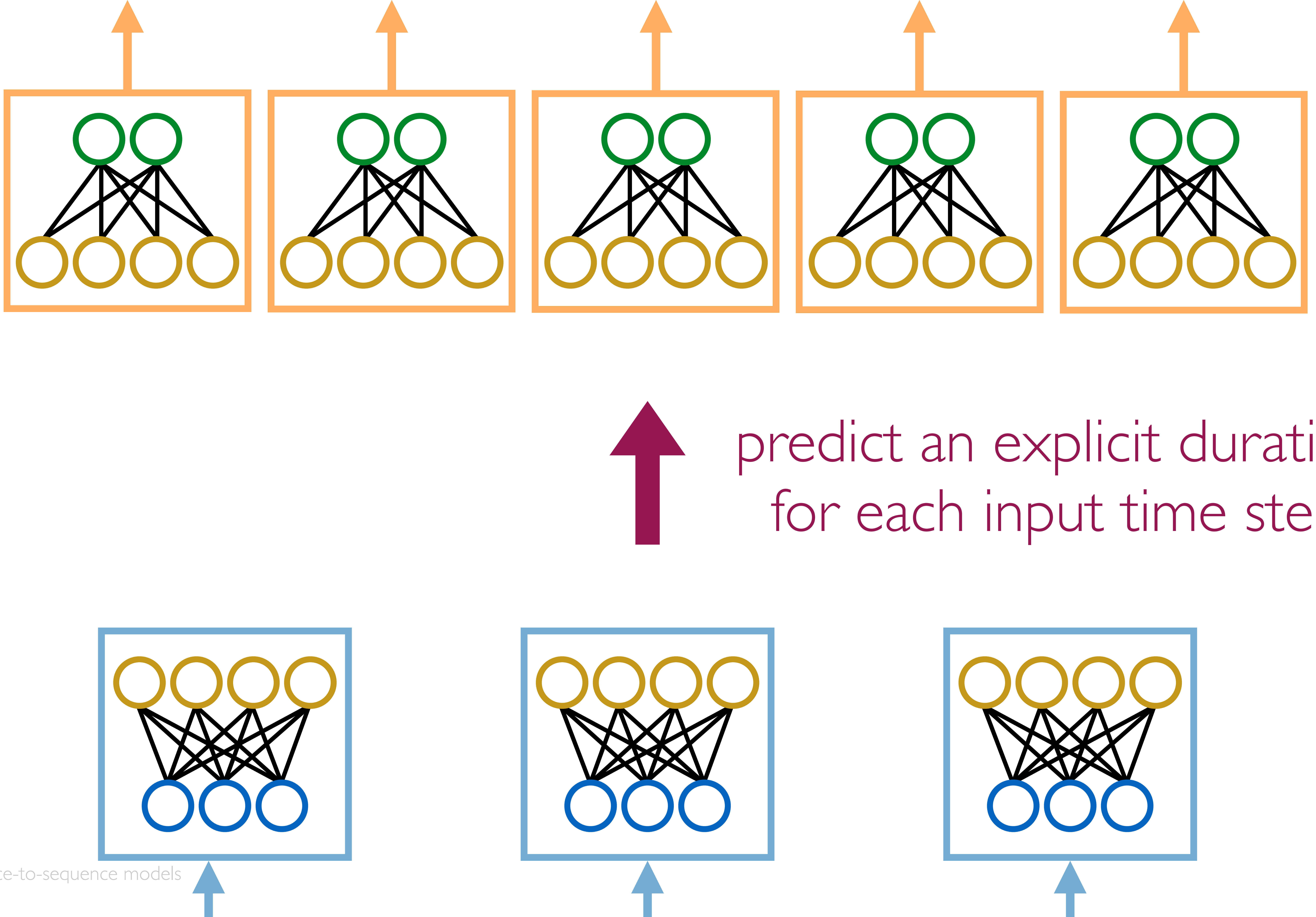
- Solution 1: attention
- Solution 2: explicit duration model

output time steps are frames (e.g., of a mel spectrogram)



input time steps are linguistic units (e.g., phones)


Decoder
Encoder

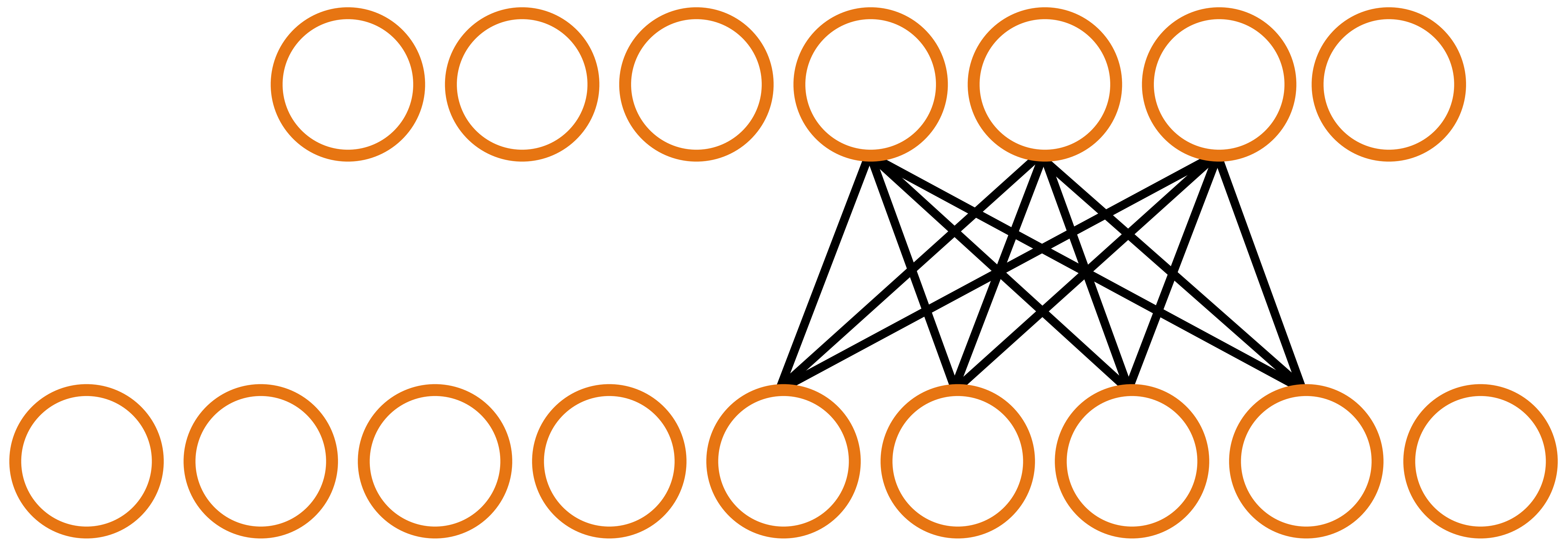

predict an explicit duration
for each input time step

Orientation

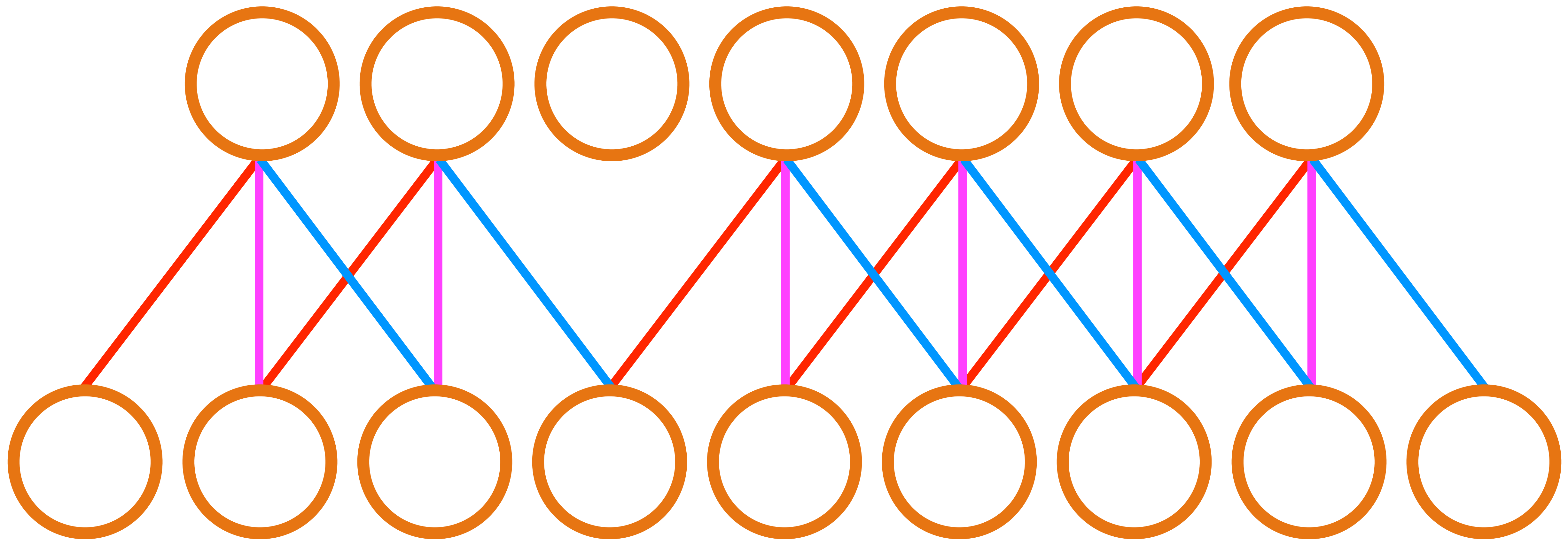
- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

- Solution 1: attention
- Solution 2: explicit duration model

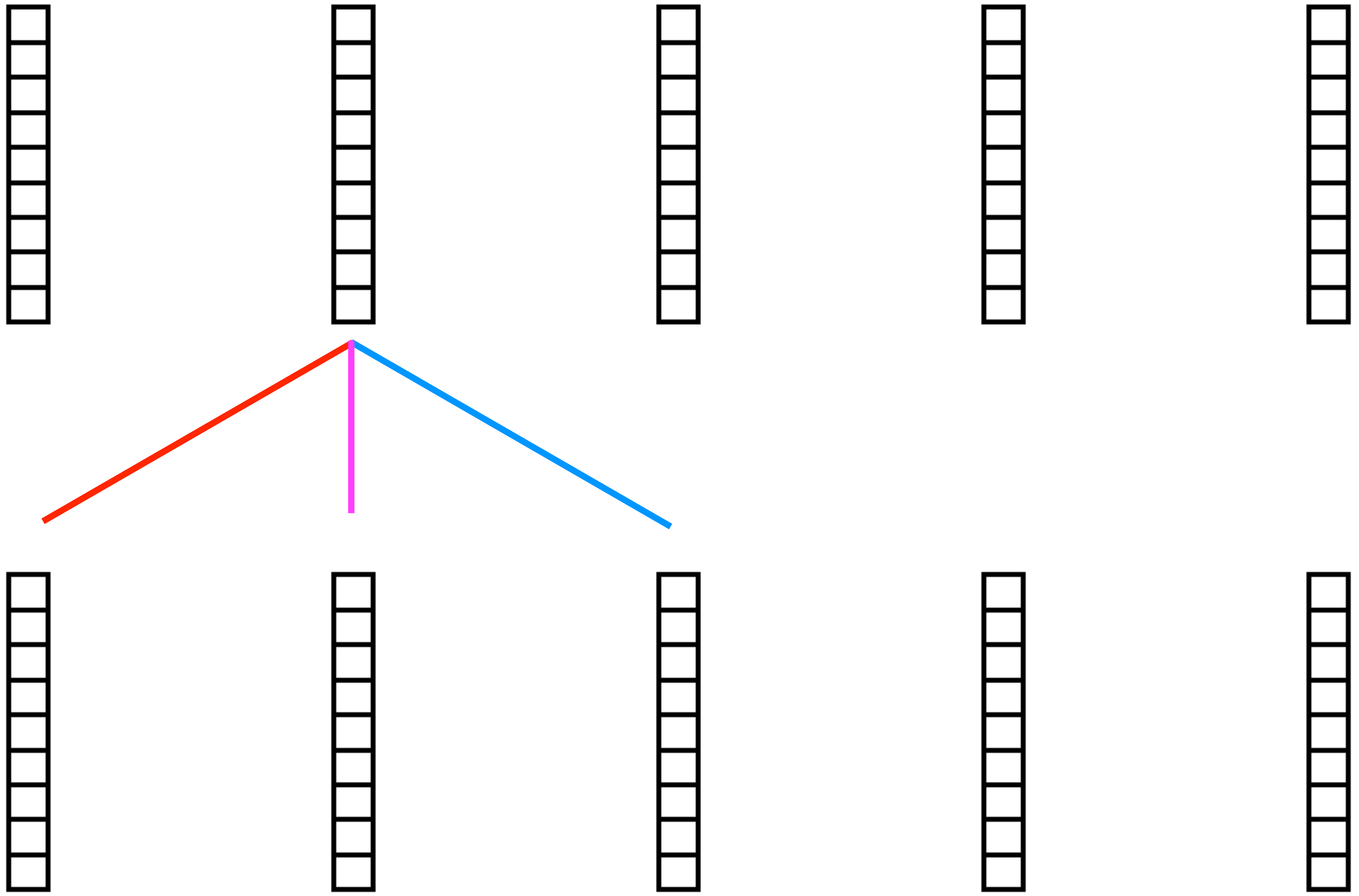
Neural building blocks : fully connected layer



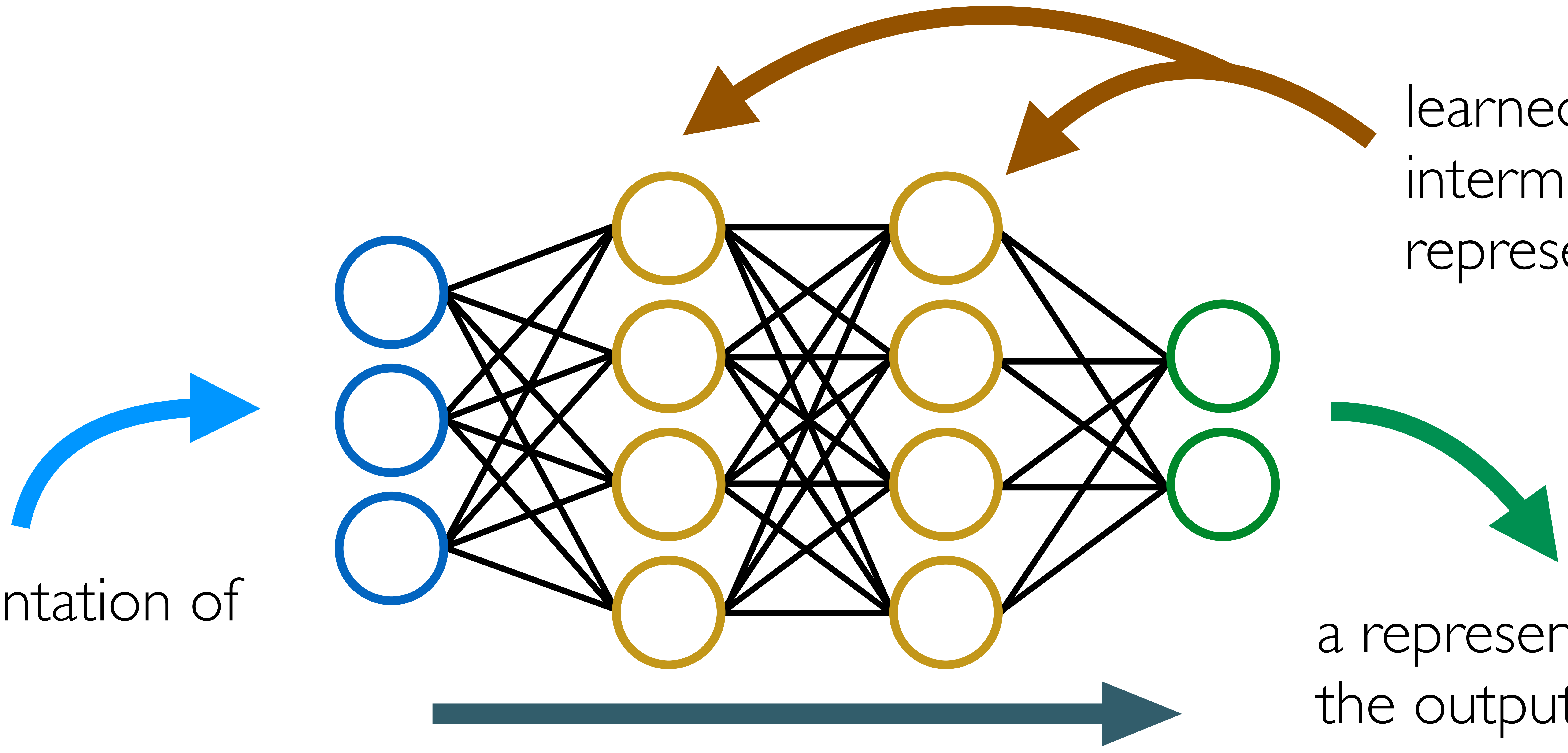
Neural building blocks : convolutional layer



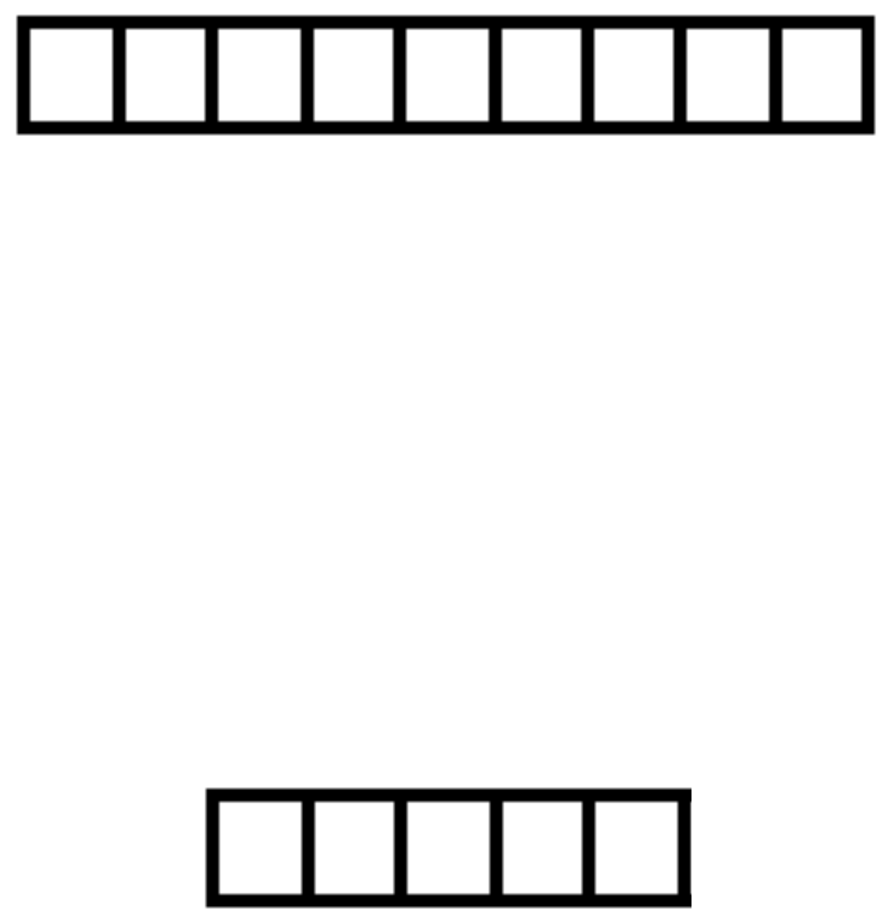
Using convolution to learn input feature engineering



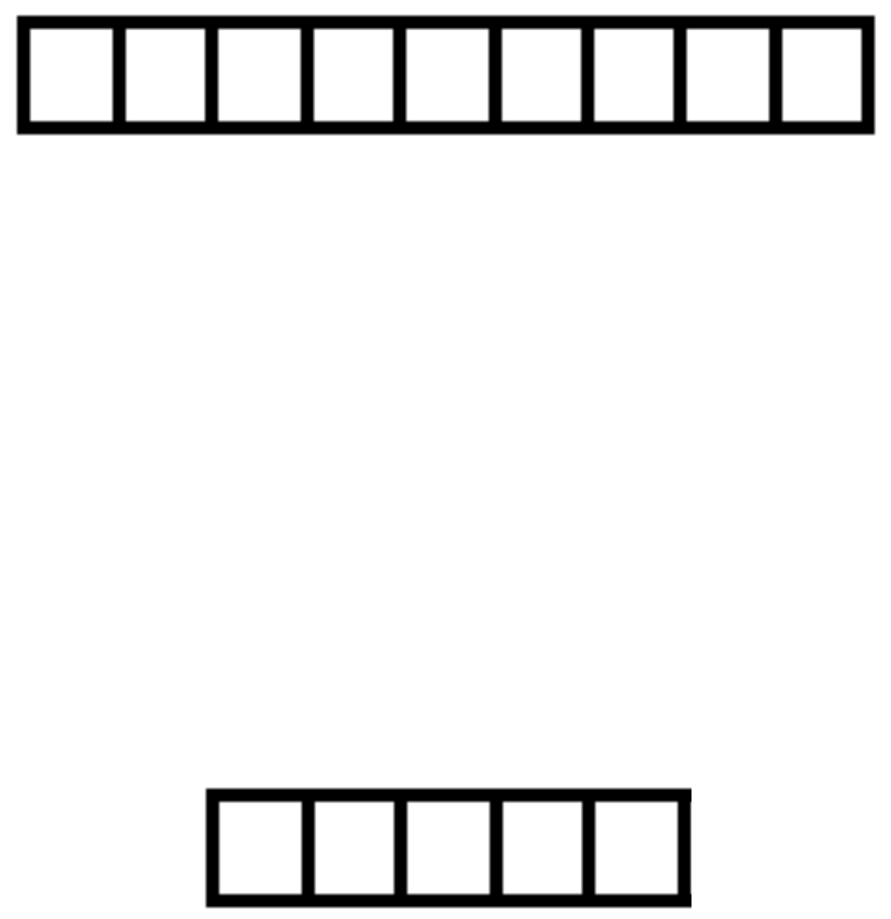
PAUSE! What are all those layers for? Learning **representations**!


learned intermediate representations
a representation of the input
a representation of the output
a sequence of **non-linear** projections

Inputting a one-hot vector into the model: **embedding**

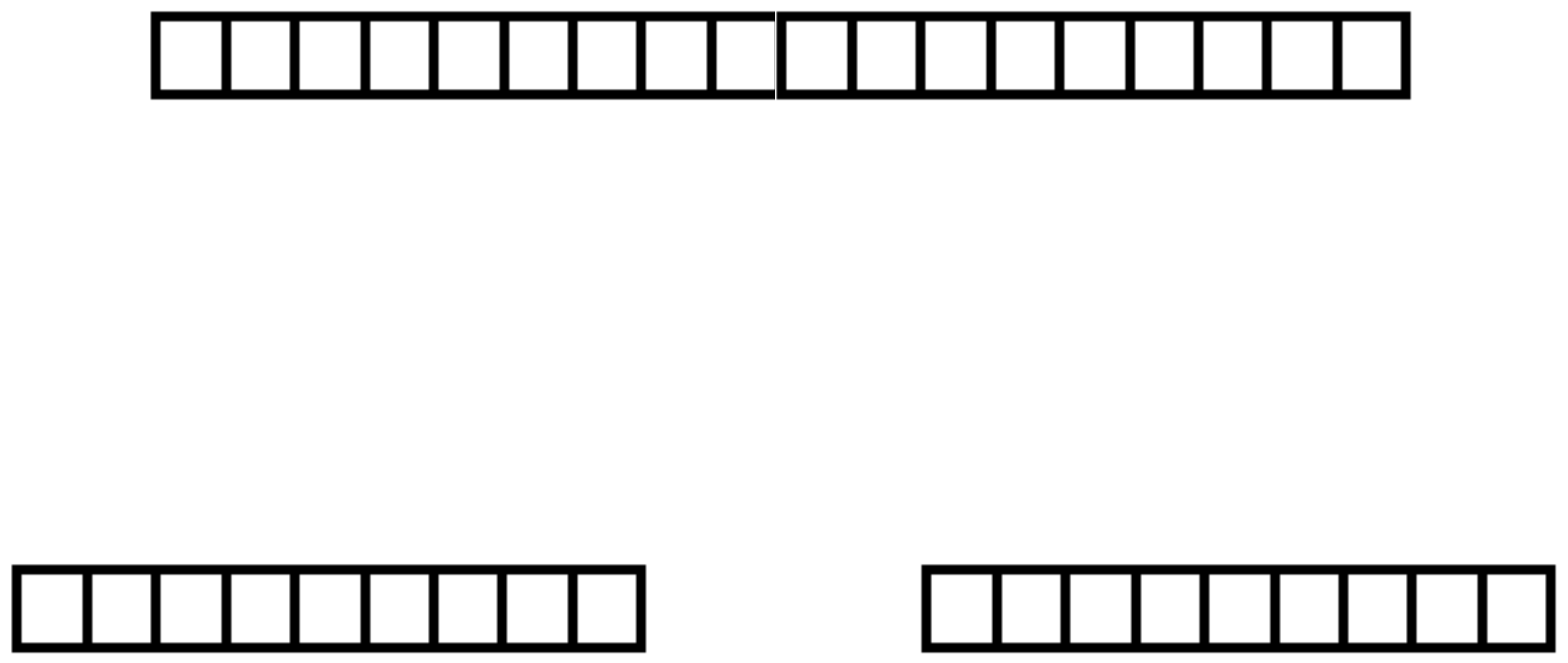


Changing the dimensionality of the representation: **projection**

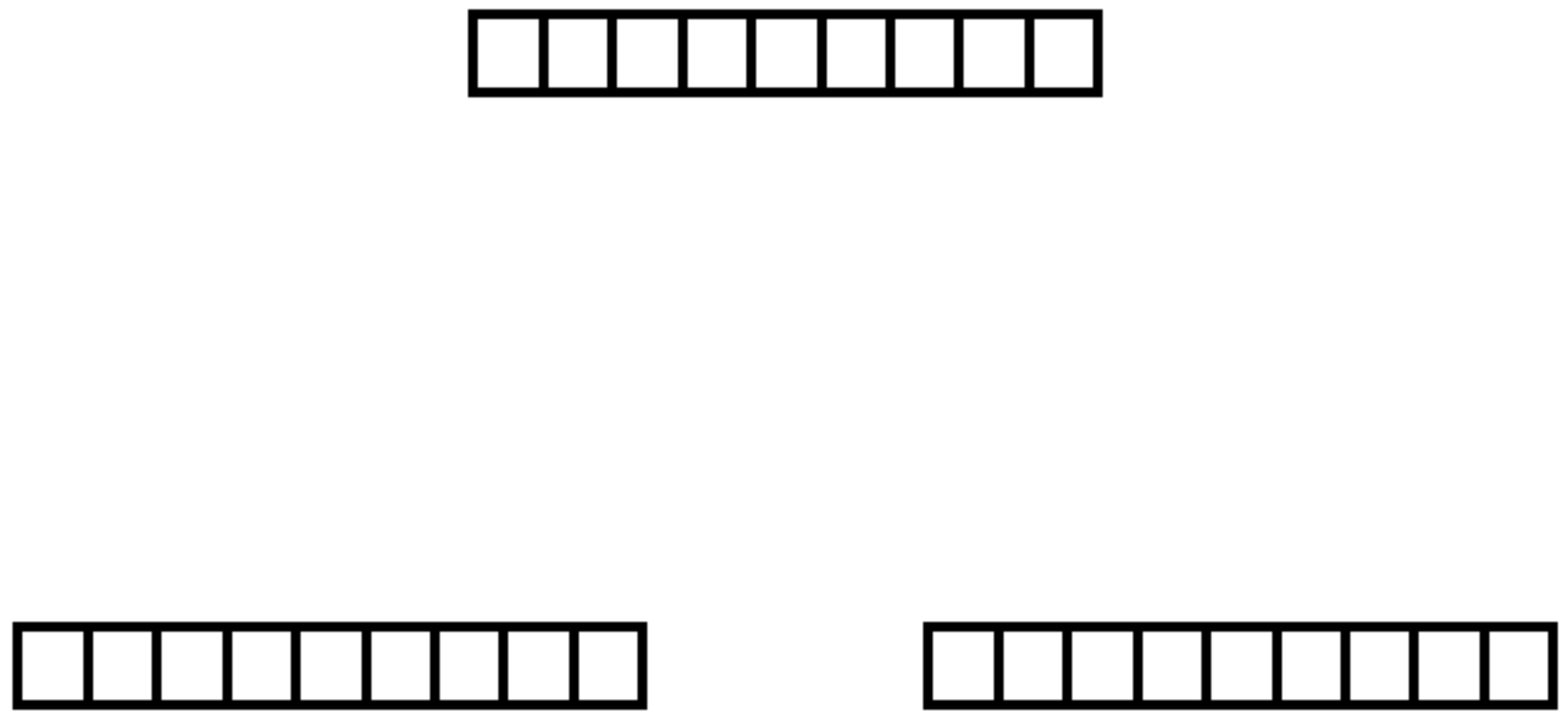


Combining representations as information flows through the model

Option 1: concatenate

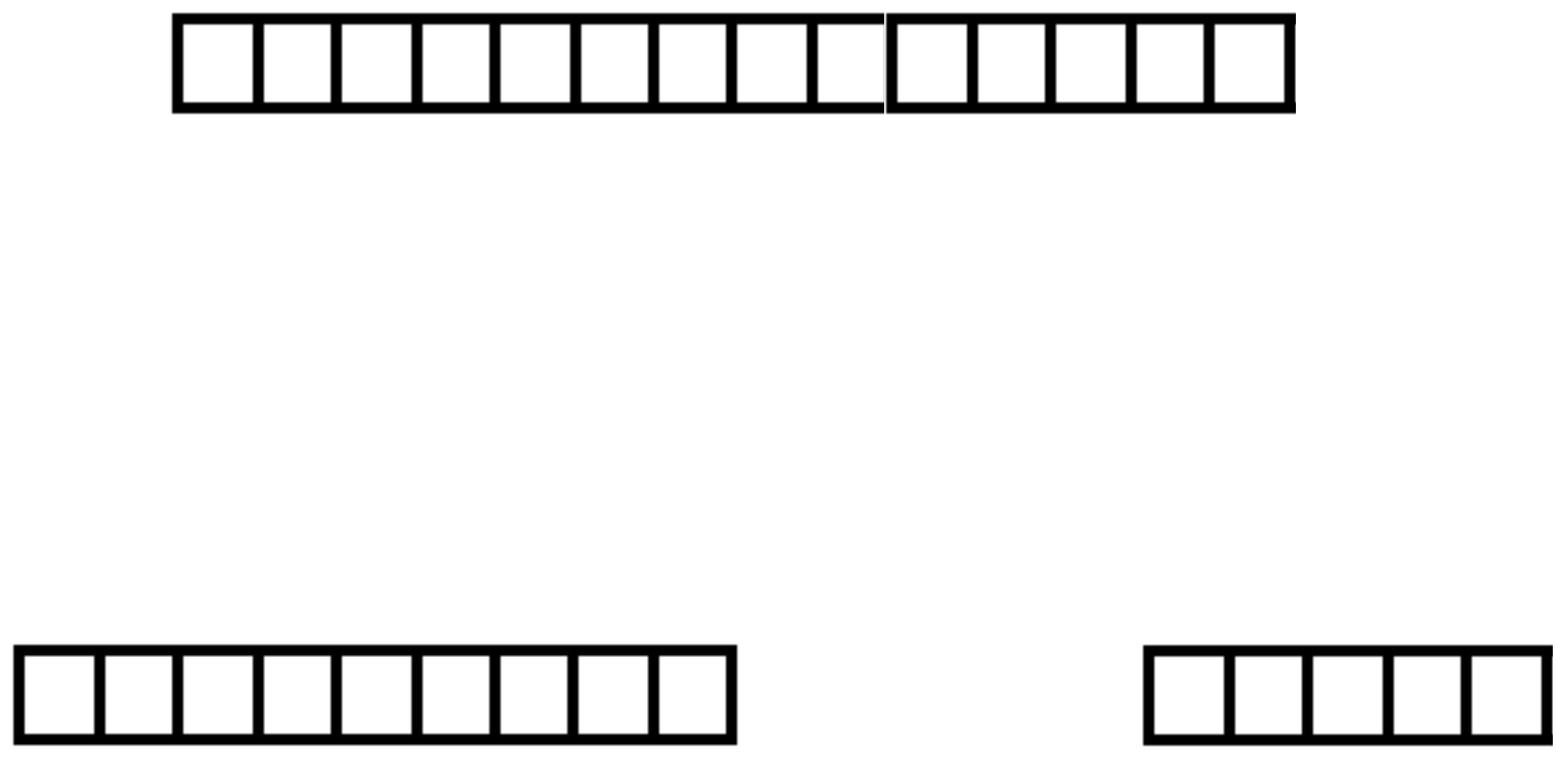


Option 2: sum

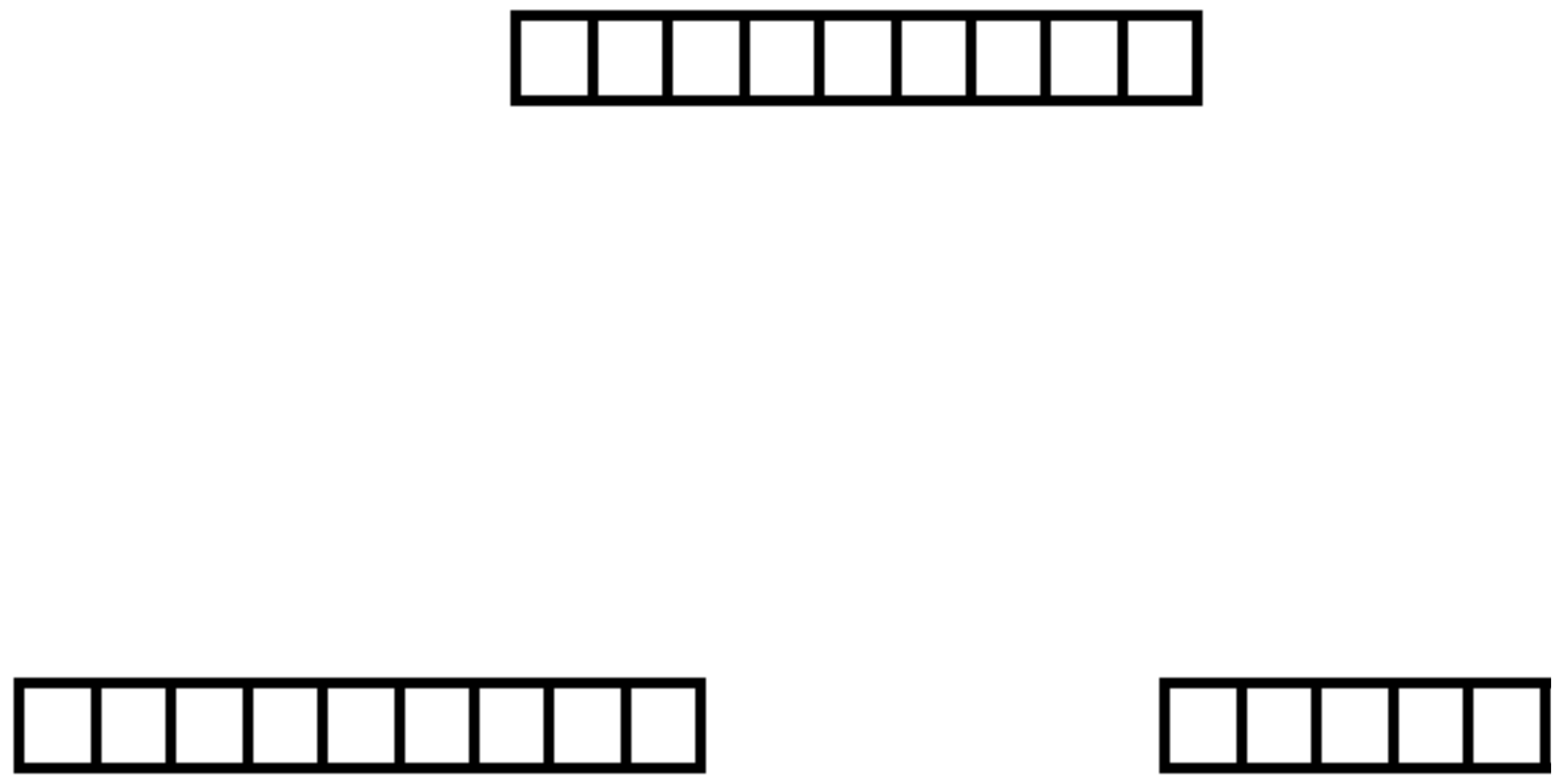


Combining representations as information flows through the model

Option 1: concatenate

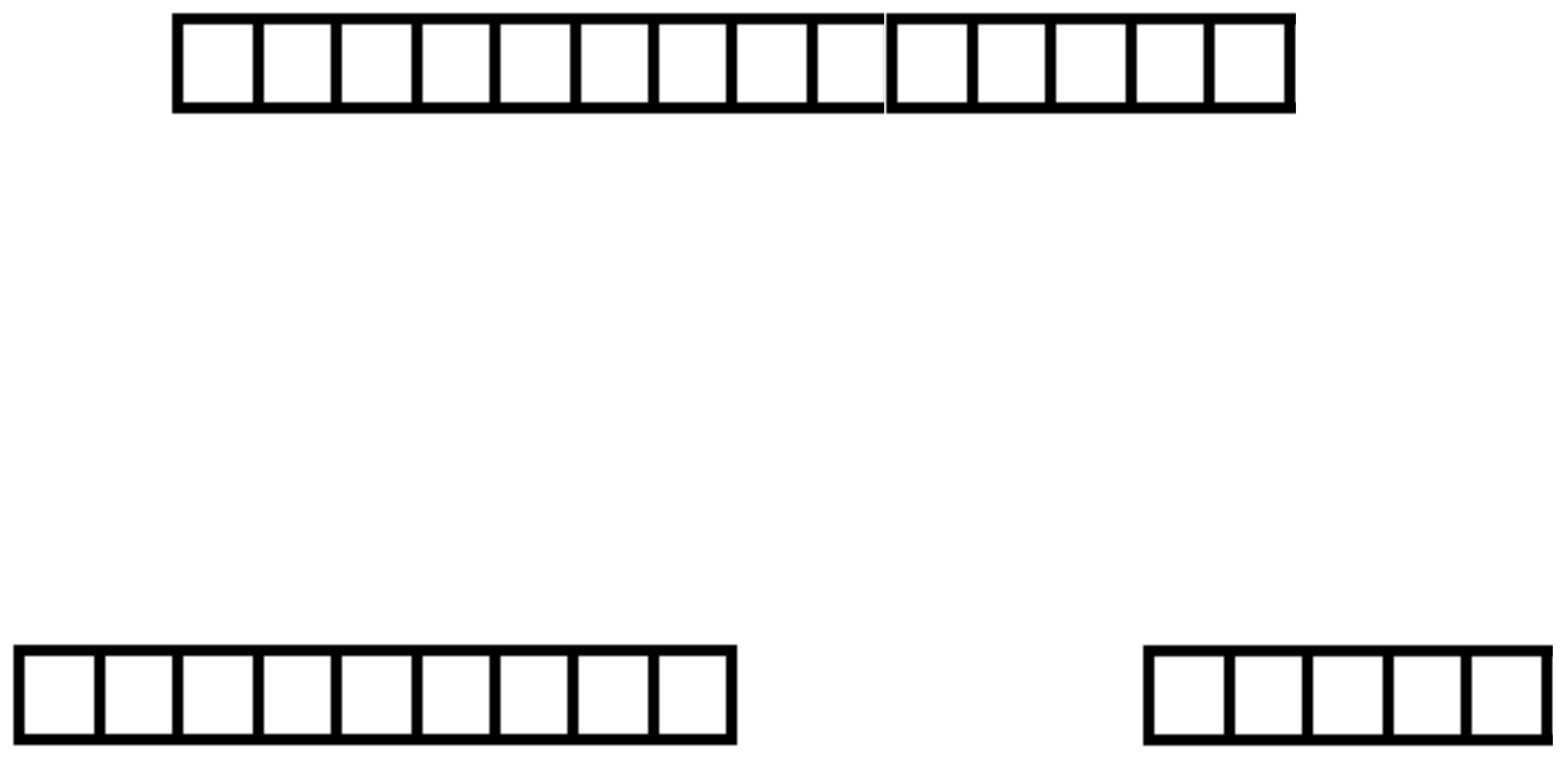


Option 2: sum

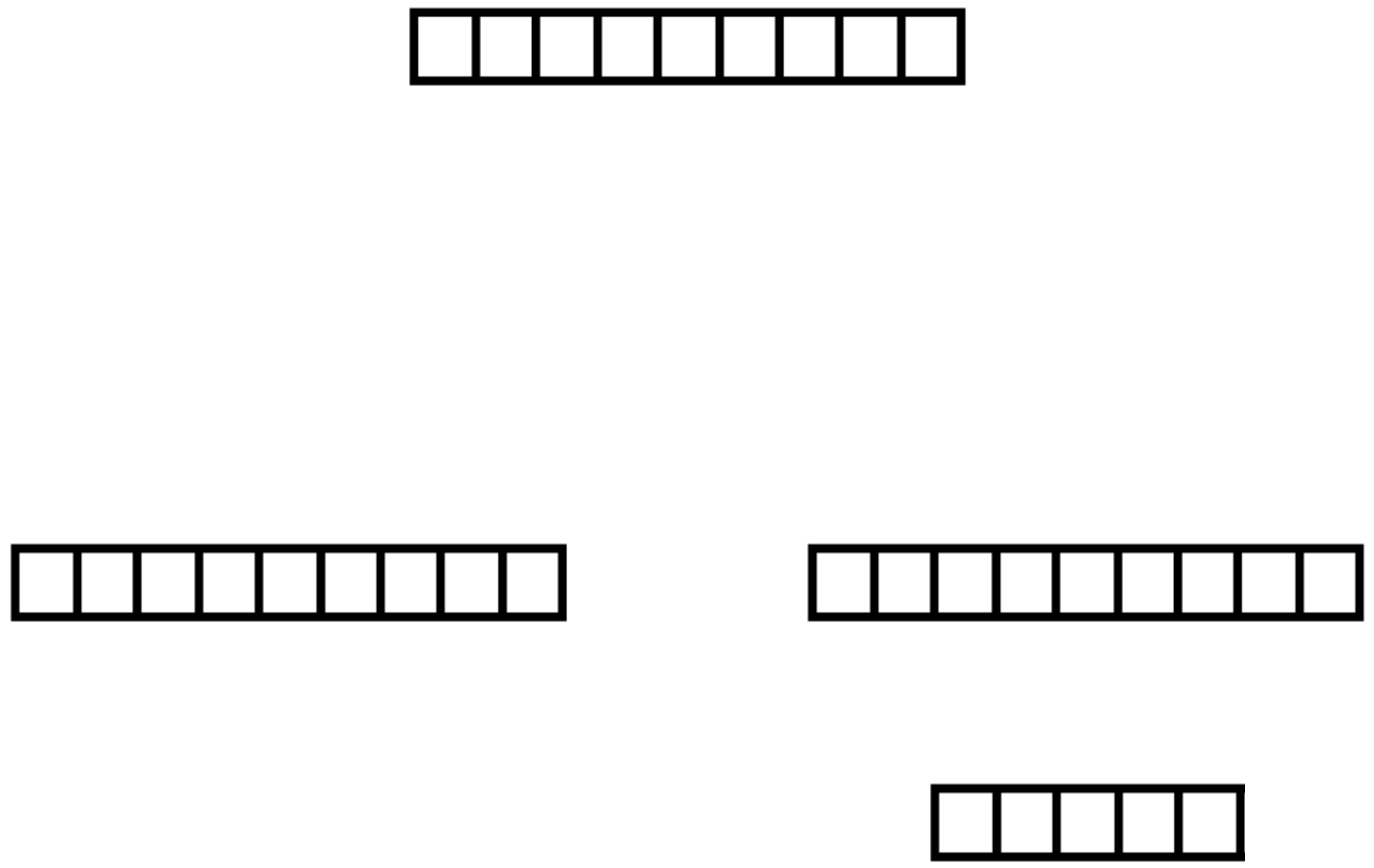


Combining representations as information flows through the model

Option 1: concatenate



Option 2: sum



Terminology

- types of layer
  - fully-connected (FC)
  - recurrent
  - LSTM, GRU, bidirectional LSTM (BiLSTM)
  - convolutional (conv, conv 1D)

- operations
  - embedding
  - projection
  - sum ($\oplus$) vs. concatenation (concat)

Orientation

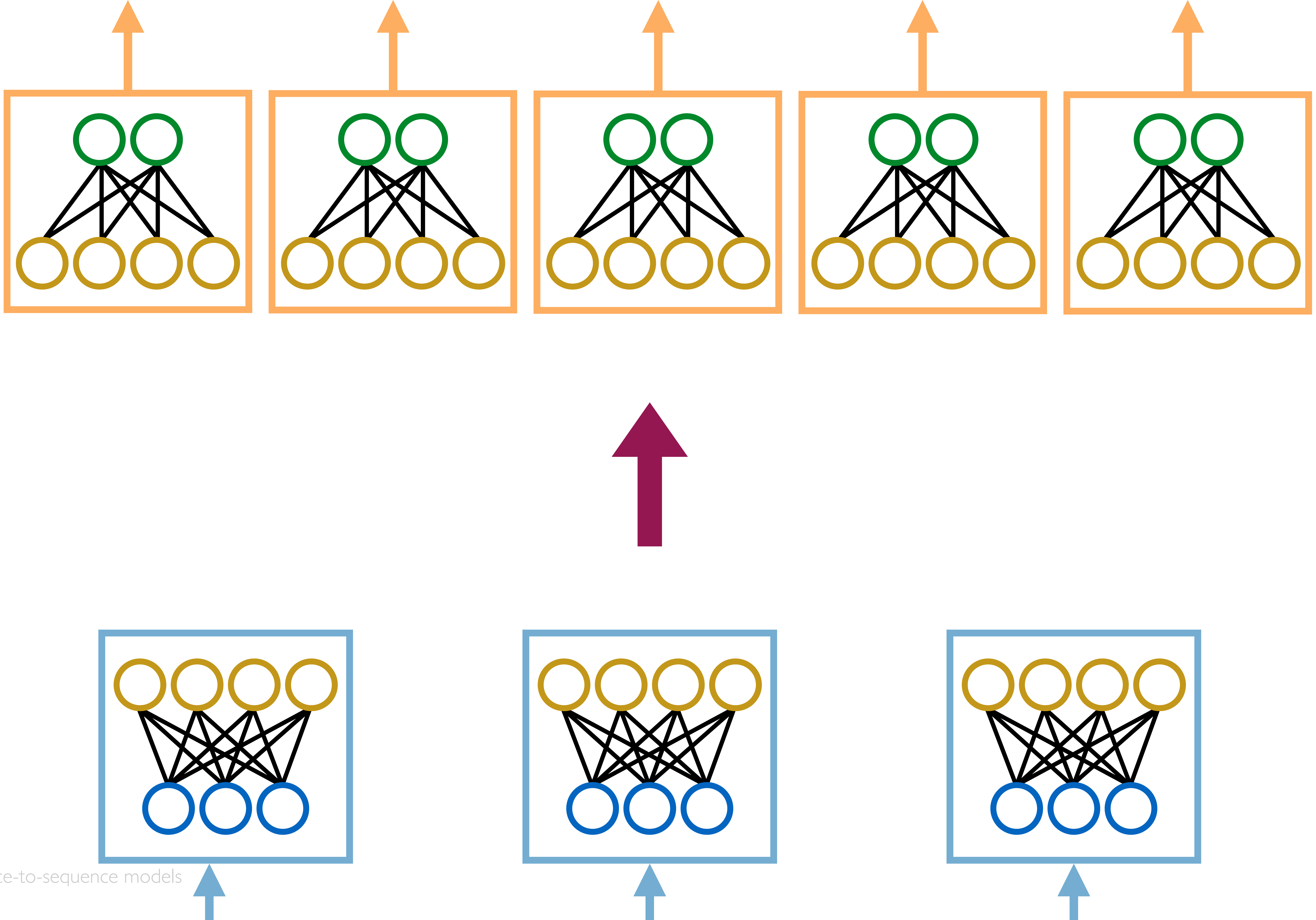
- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

Orientation

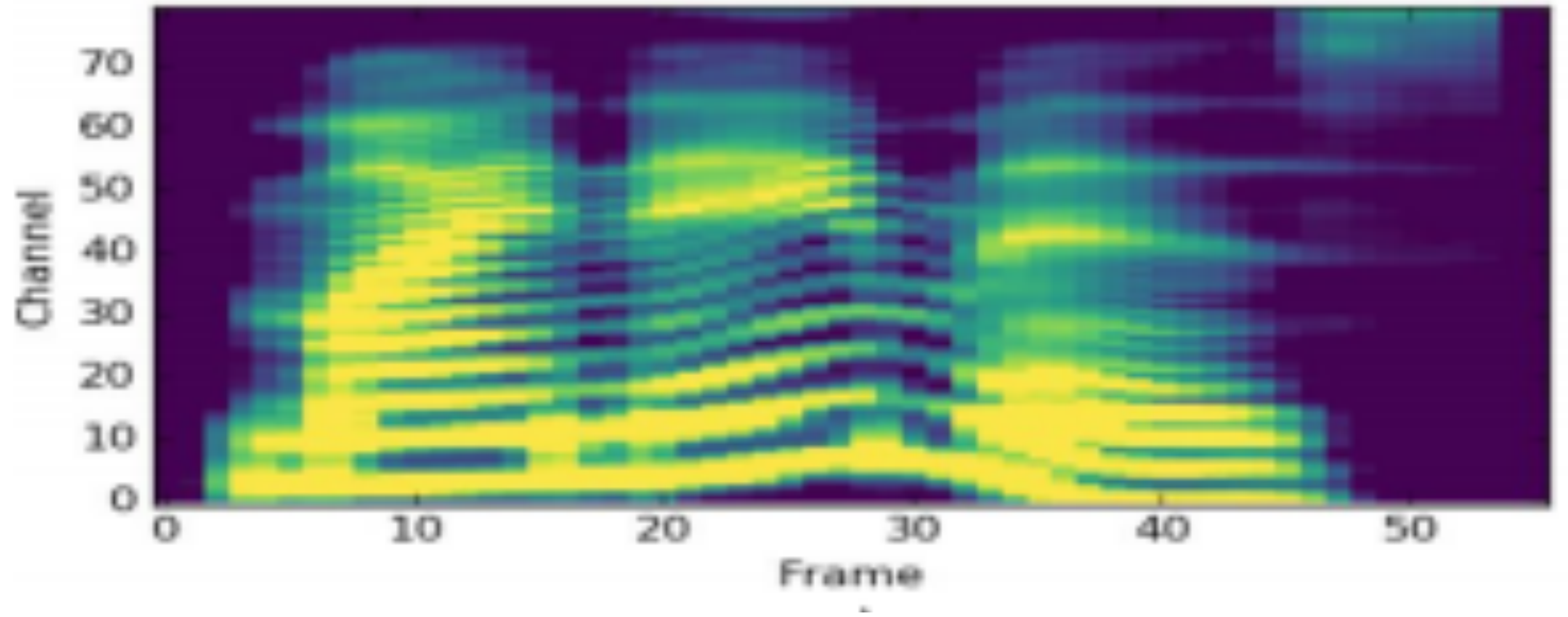
- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

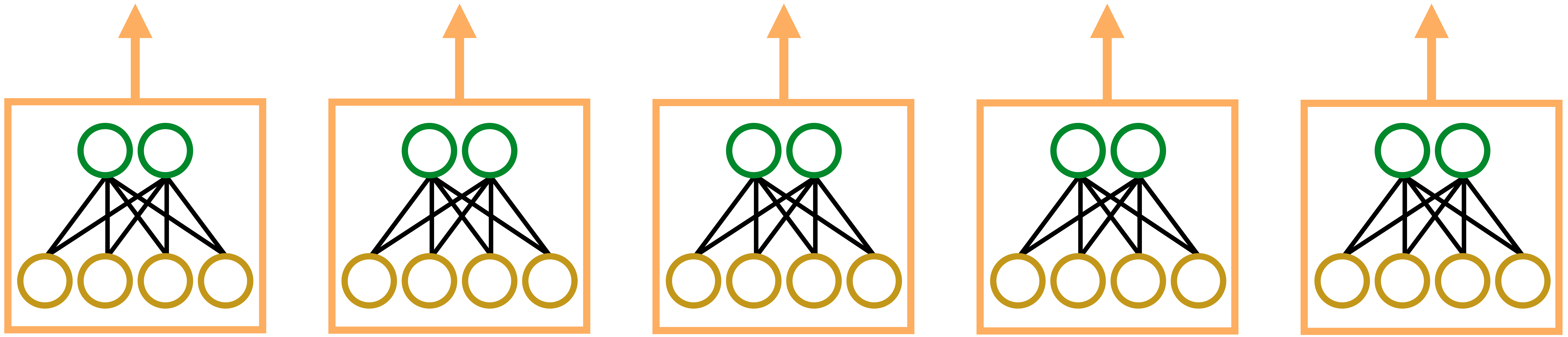
$\rightarrow$

- Convolutional layer(s)



mel spectrogram





Terminology

- autoregressive

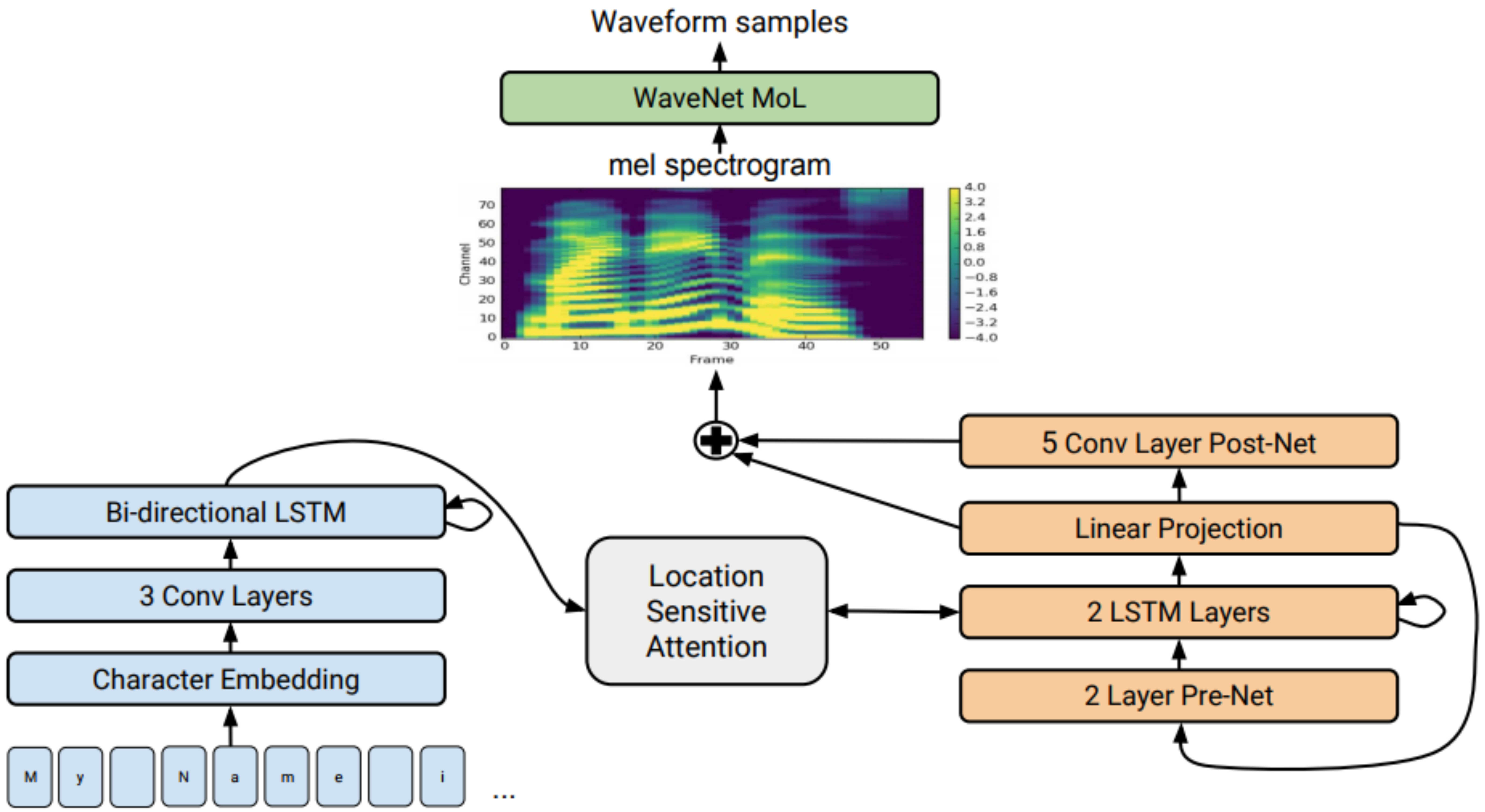
Orientation

- Input features
  - the model should **learn input feature engineering**
- Duration
  - **integrate** into the model
- Sequence modelling
  - enable the model to pass information between time steps - give it a **memory**
- Output features
  - allow output to **depend** on previous outputs

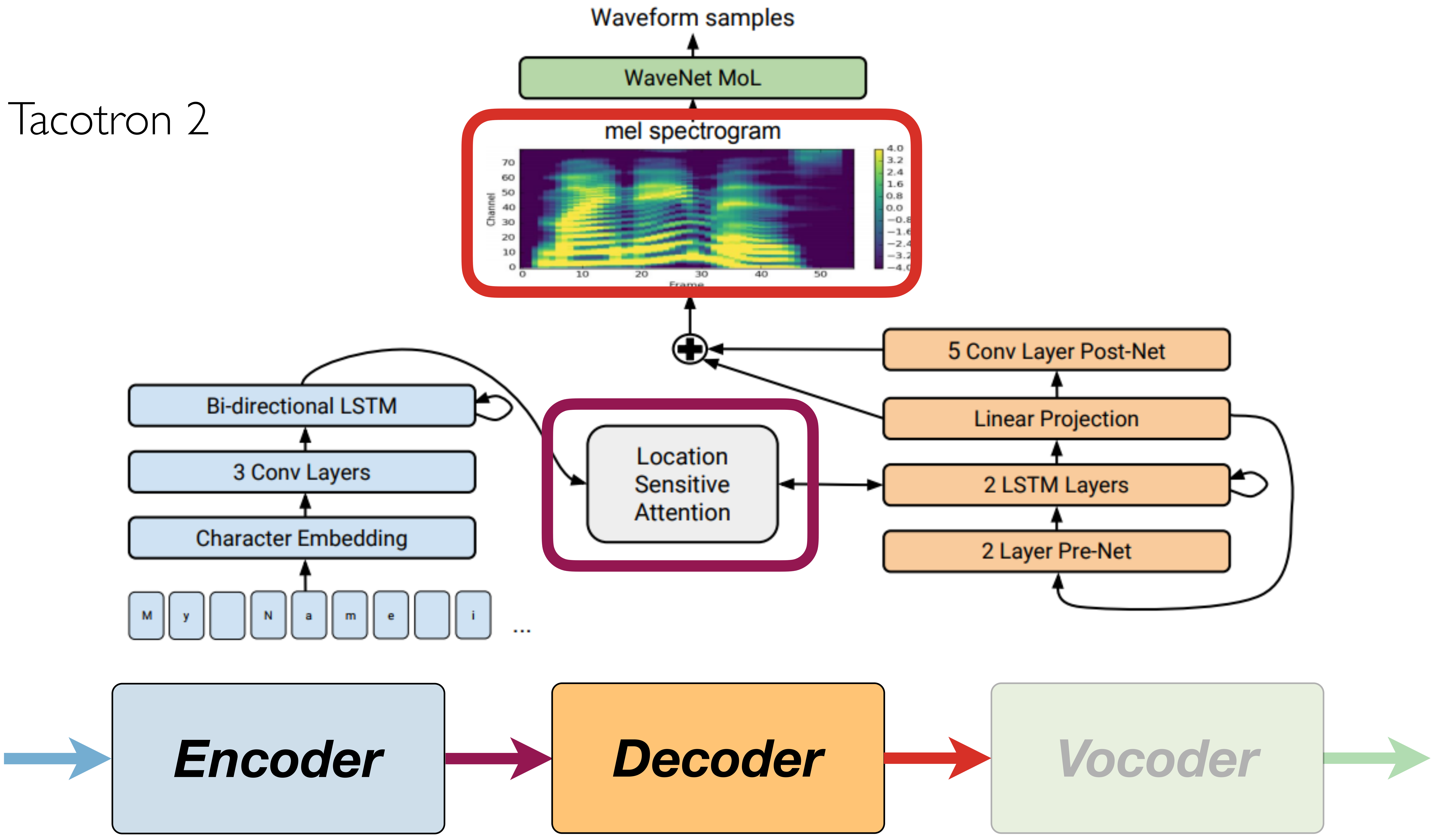
- Autoregressive model

Case study

Tacotron 2

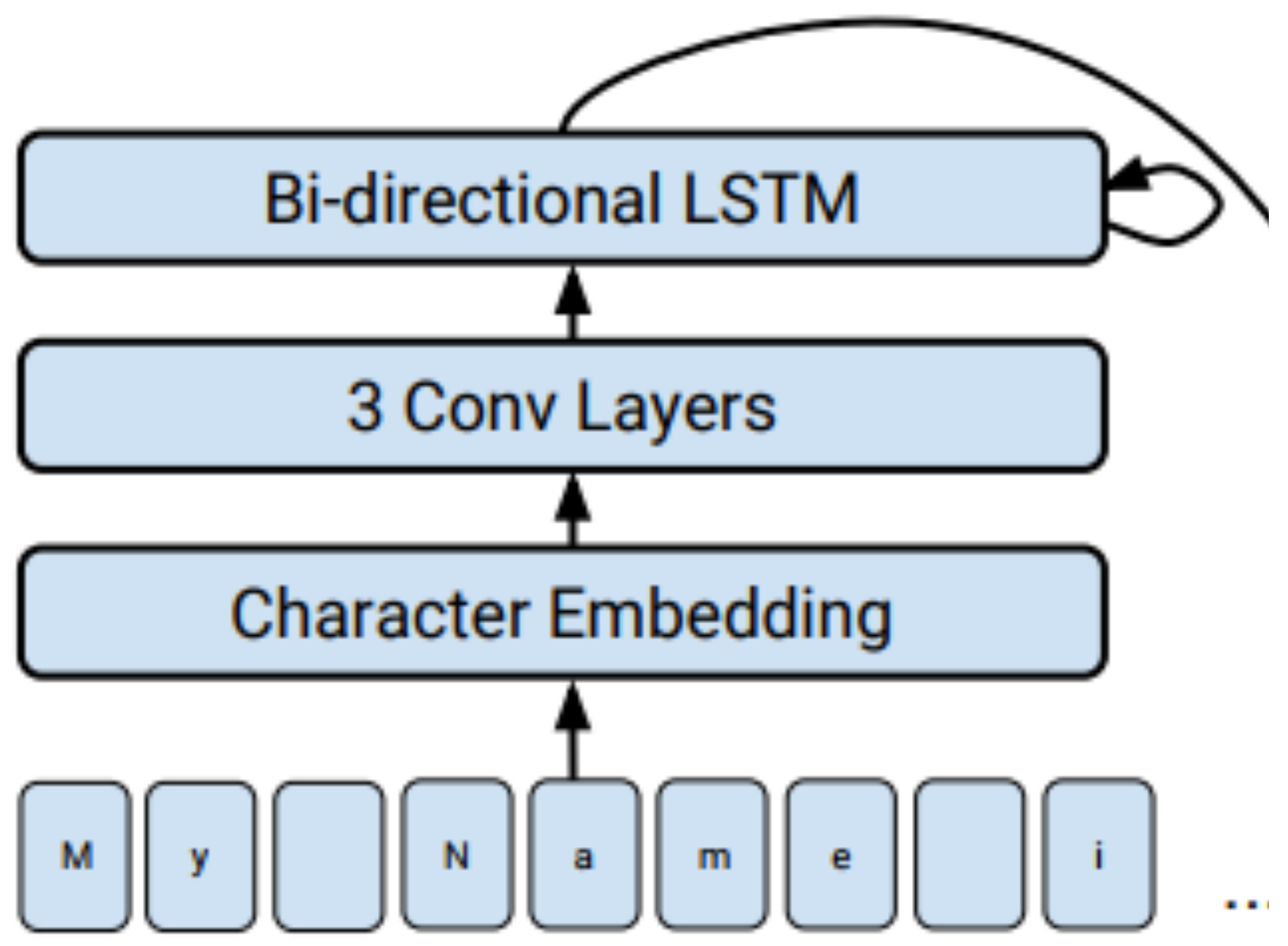


Tacotron 2

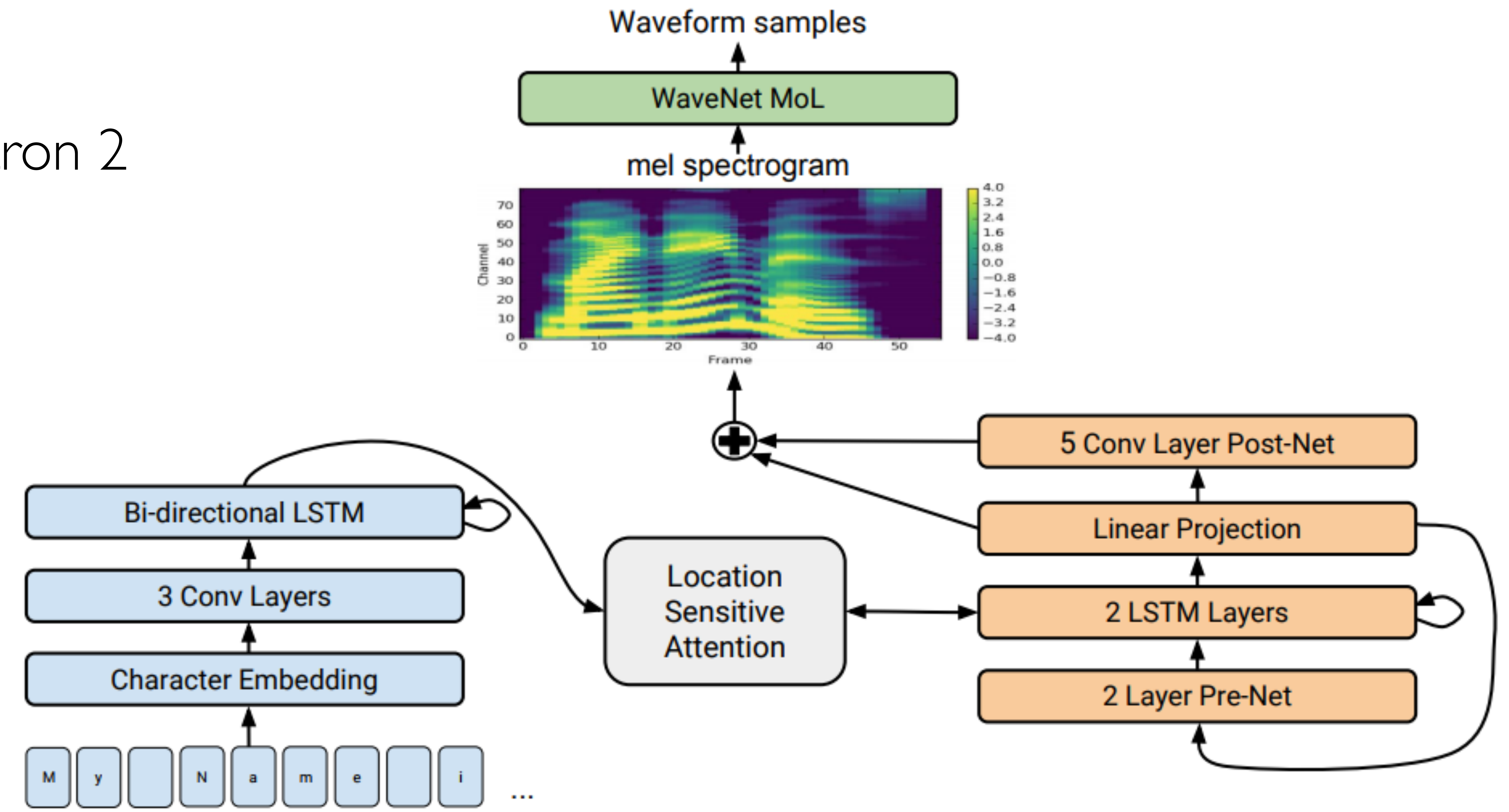


Waveform samples
WaveNet MoL
mel spectrogram
Channel
Frame
5 Conv Layer Post-Net
Linear Projection
2 LSTM Layers
2 Layer Pre-Net
Location Sensitive Attention
Bi-directional LSTM
3 Conv Layers
Character Embedding
M y N a m e i ...
Encoder
Decoder
Vocoder

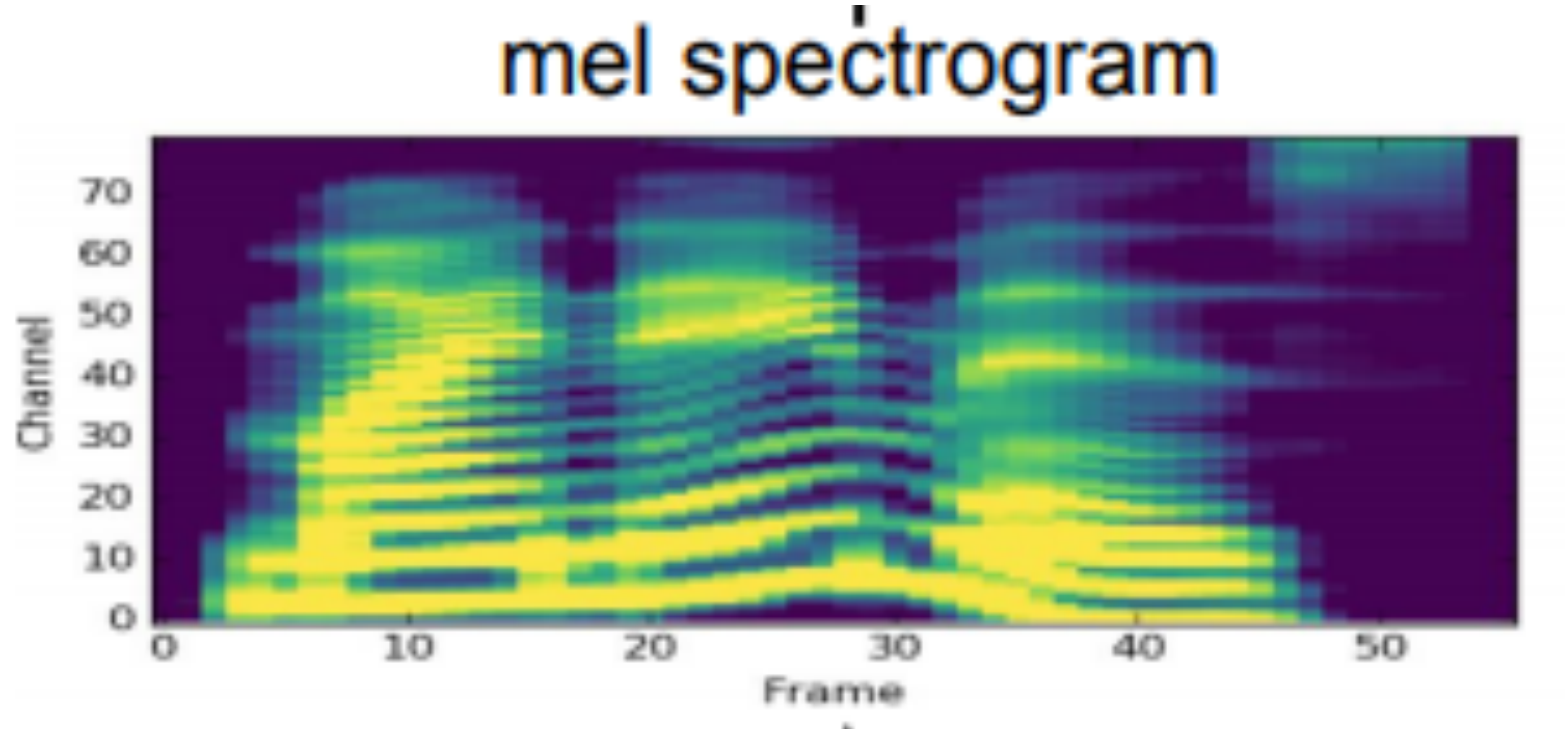
Tacotron 2

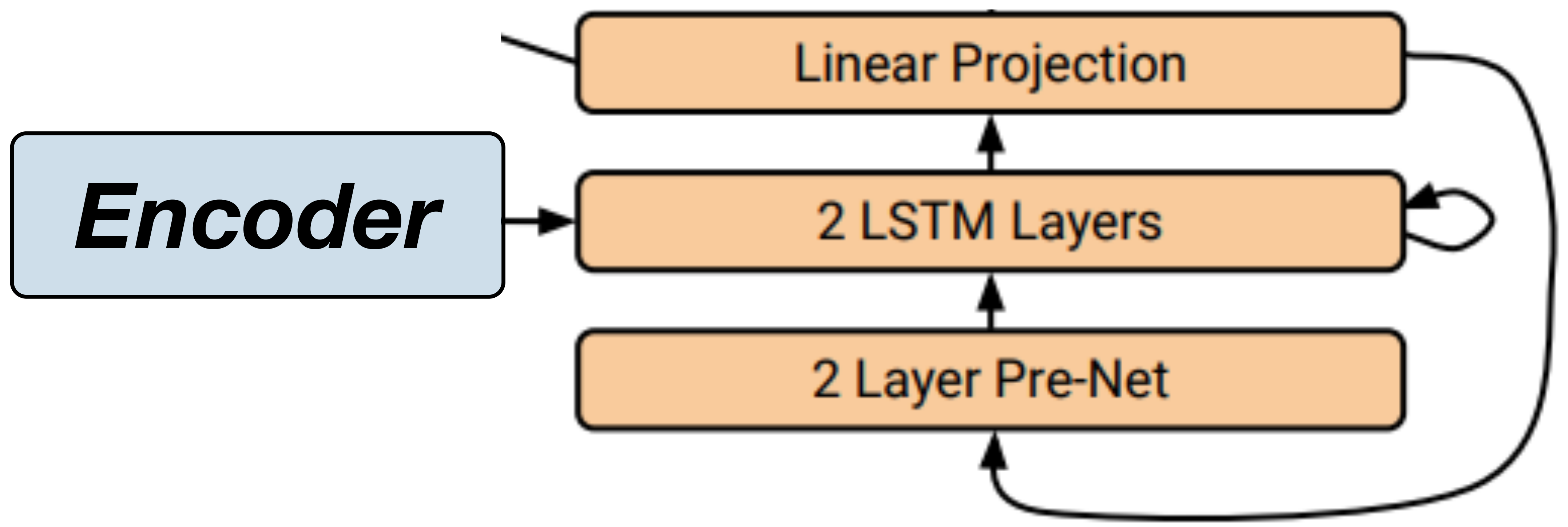

Bi-directional LSTM
3 Conv Layers
Character Embedding
M
y
N
a
m
e
i
...

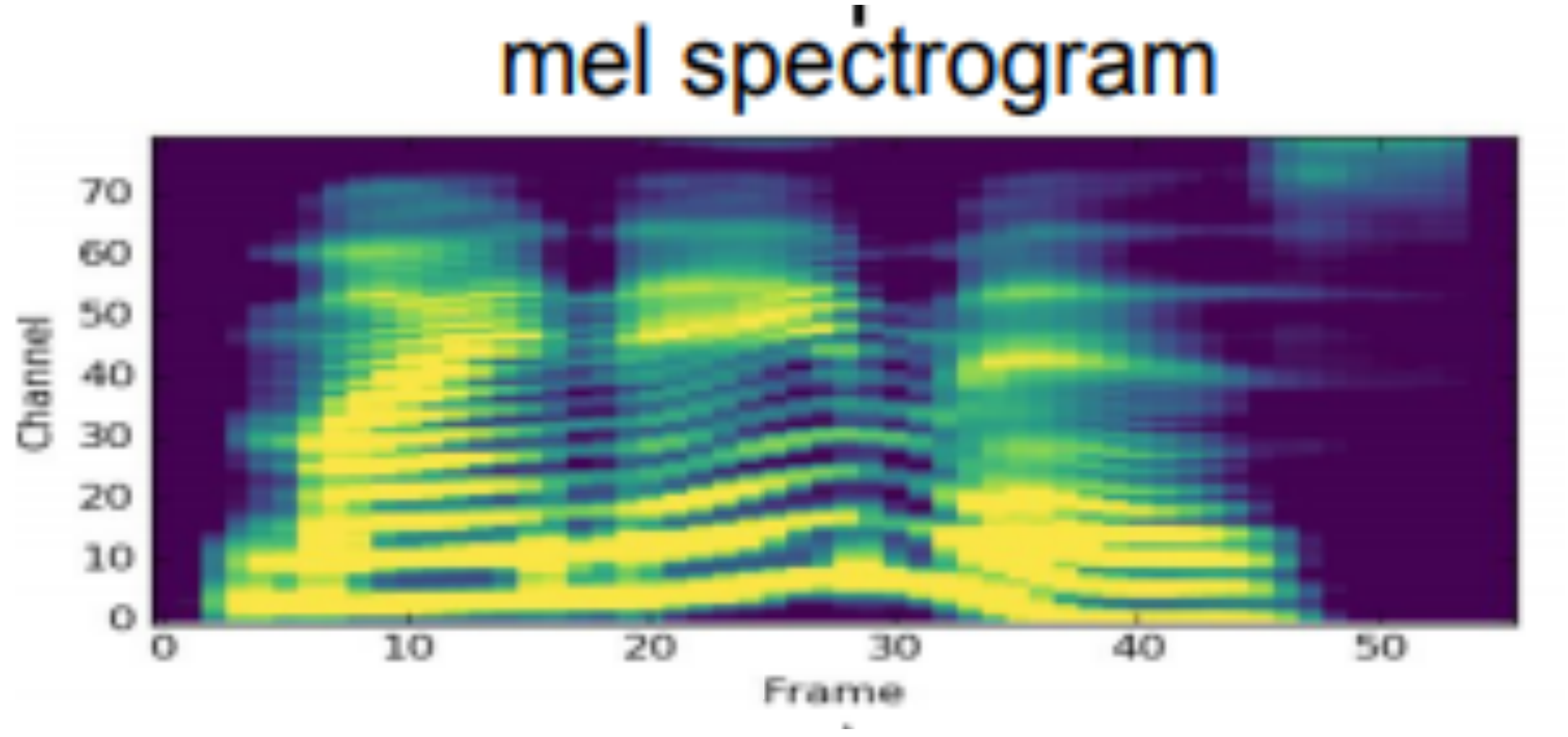
Tacotron 2

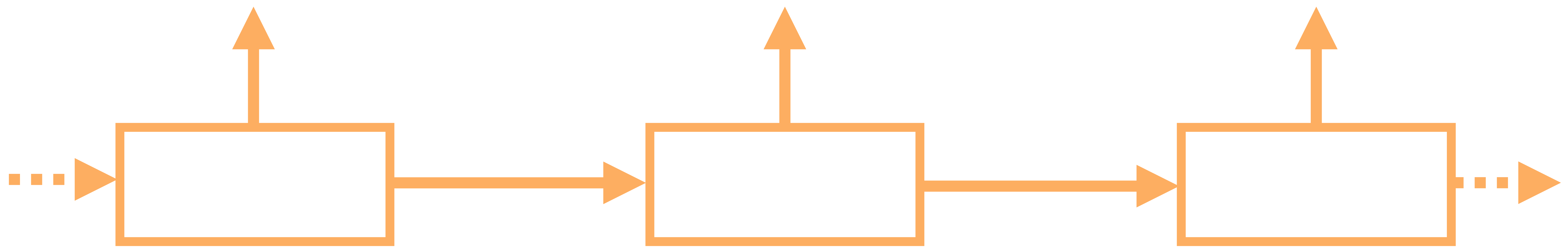

Waveform samples
WaveNet MoL
mel spectrogram
Channel
Frame
5 Conv Layer Post-Net
Linear Projection
2 LSTM Layers
2 Layer Pre-Net
Location Sensitive Attention
Bi-directional LSTM
3 Conv Layers
Character Embedding
M
y
N
a
m
e
i
...

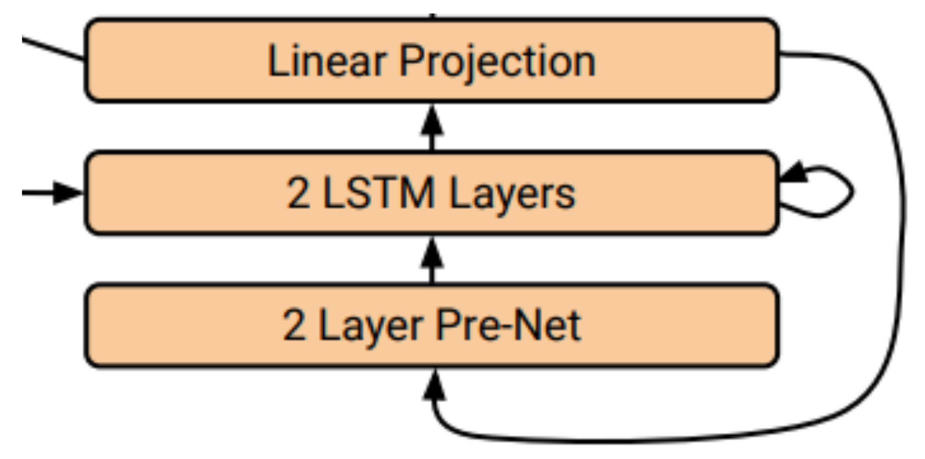
Tacotron 2


mel spectrogram
Channel
0
10
20
30
40
50
60
70
Frame
0
10
20
30
40
50

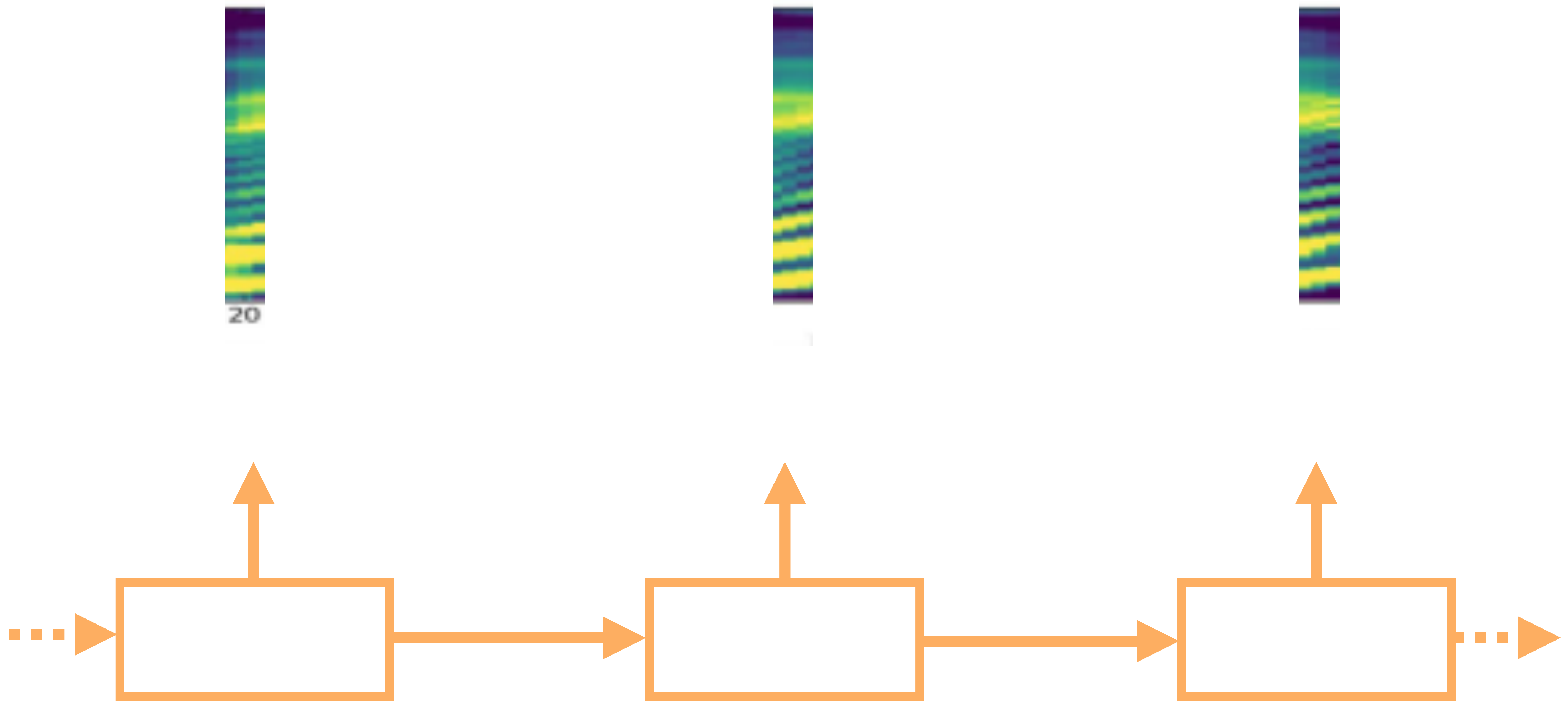

Encoder
Linear Projection
2 LSTM Layers
2 Layer Pre-Net


mel spectrogram
Channel
Frame
0
10
20
30
40
50
60
70




Linear Projection
2 LSTM Layers
2 Layer Pre-Net

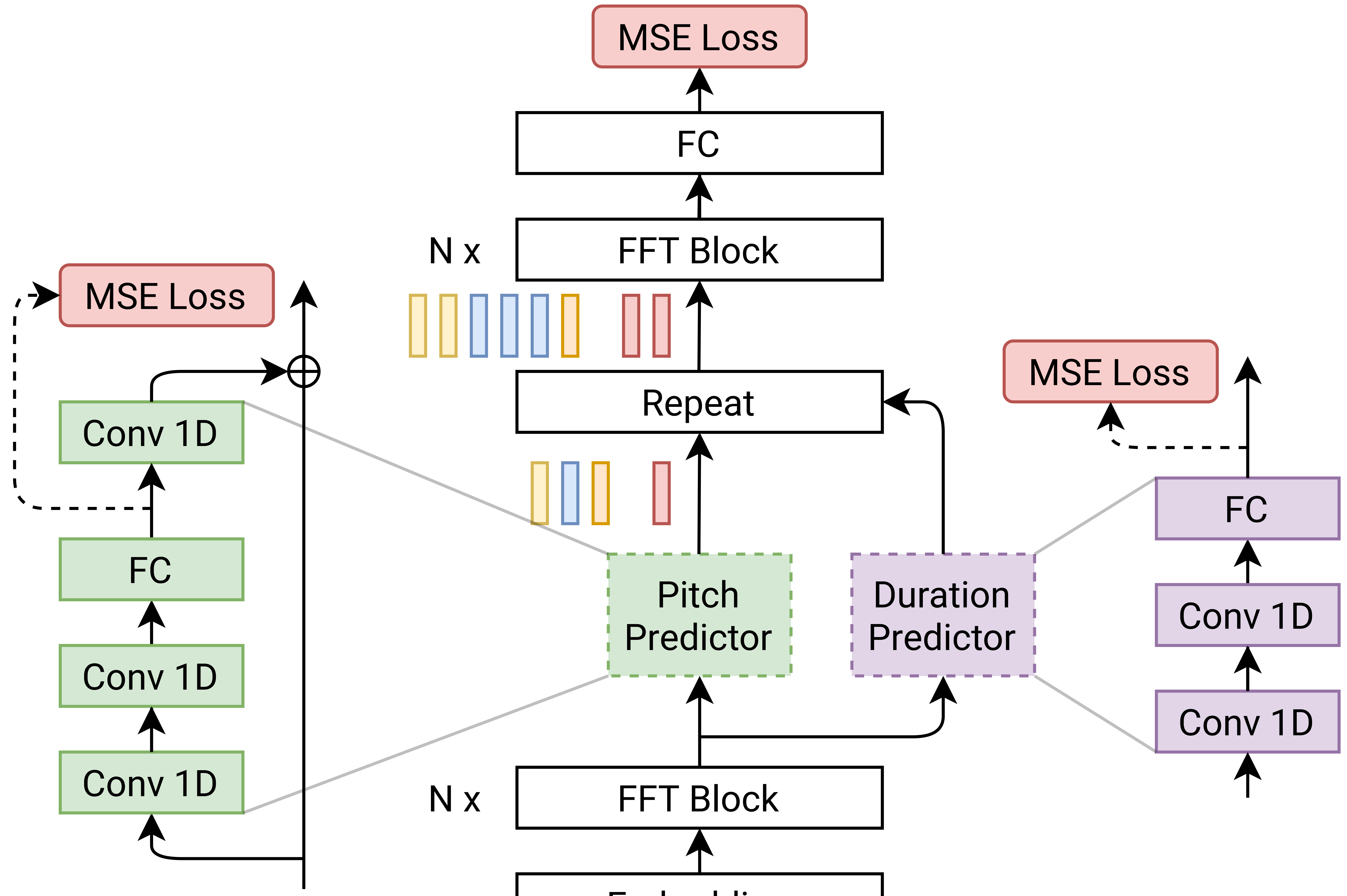
Encoder



Encoder

Case study

FastPitch



Understanding architecture diagrams

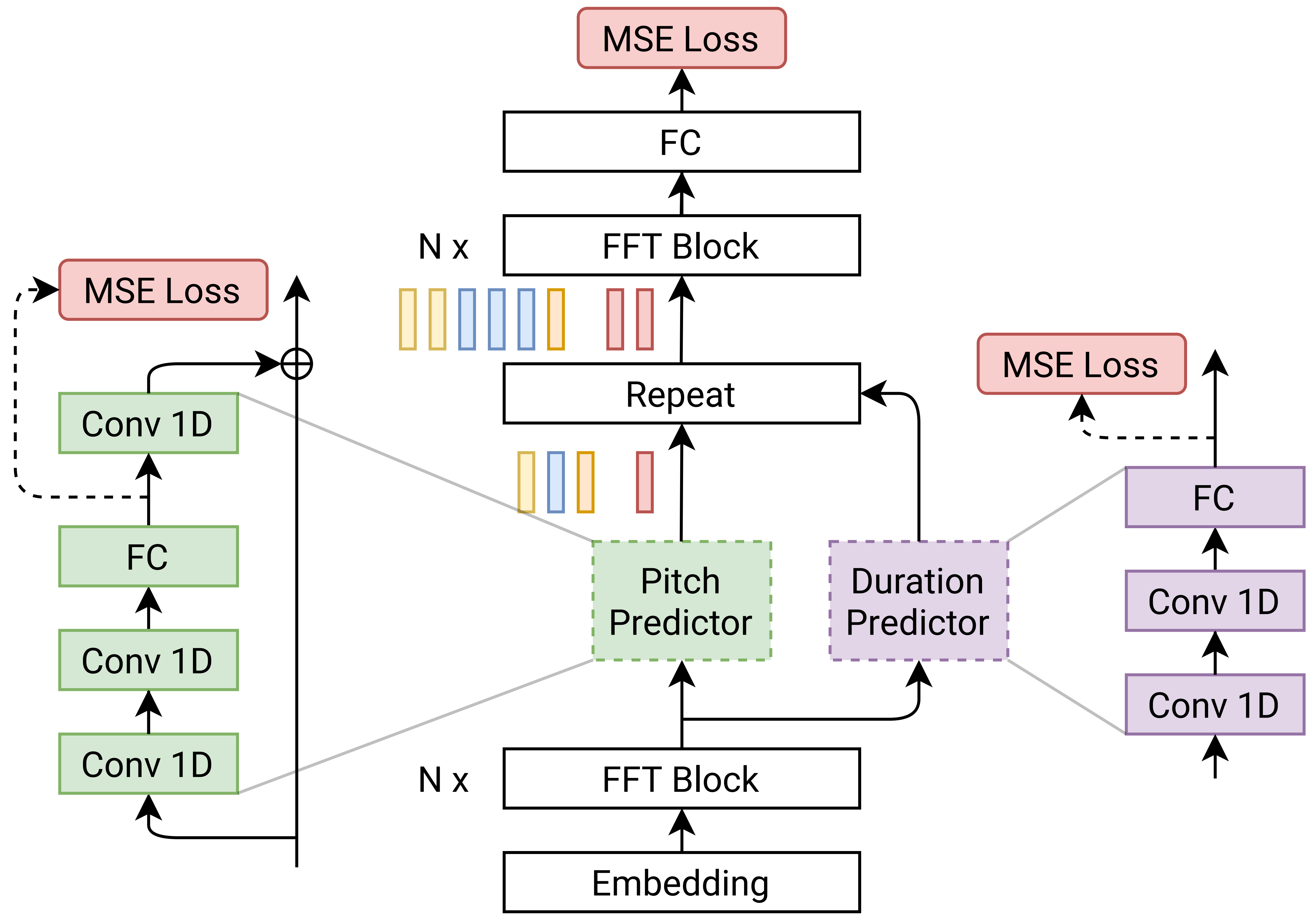
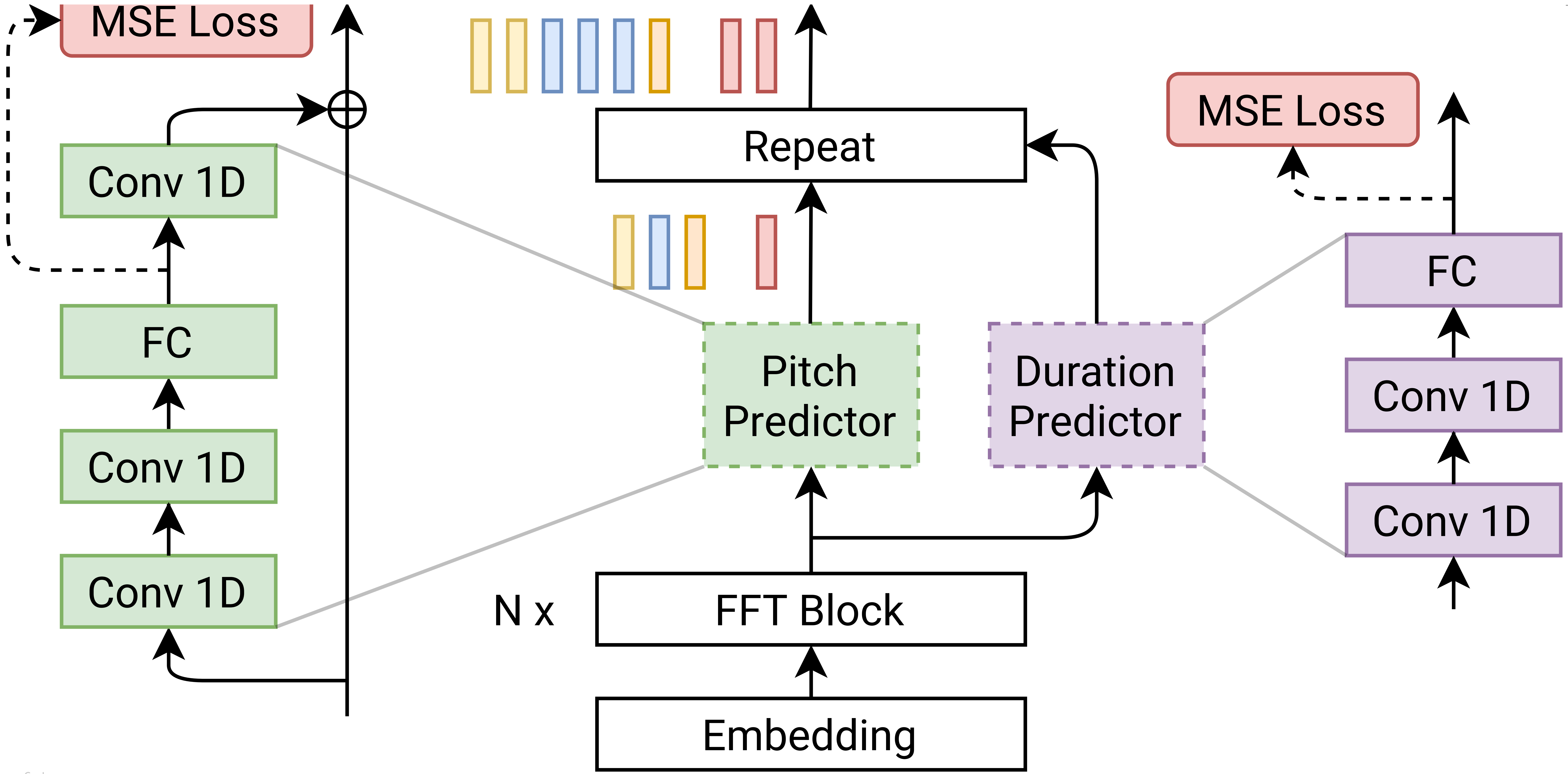


Fig. 1. **Architecture of FastPitch** follows FastSpeech [1]. A single pitch value is predicted for every temporal location.

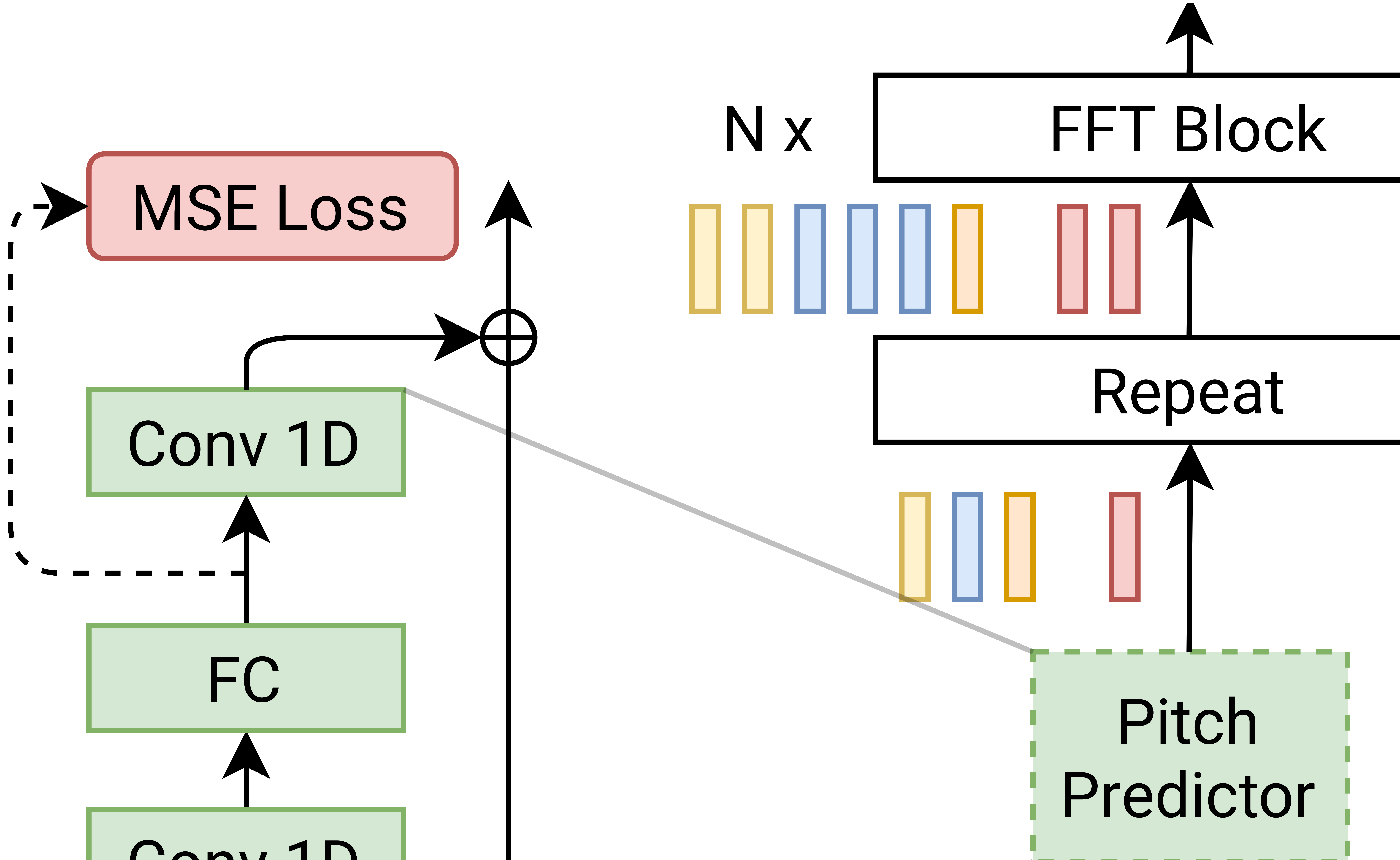
Understanding architecture diagrams : sequences



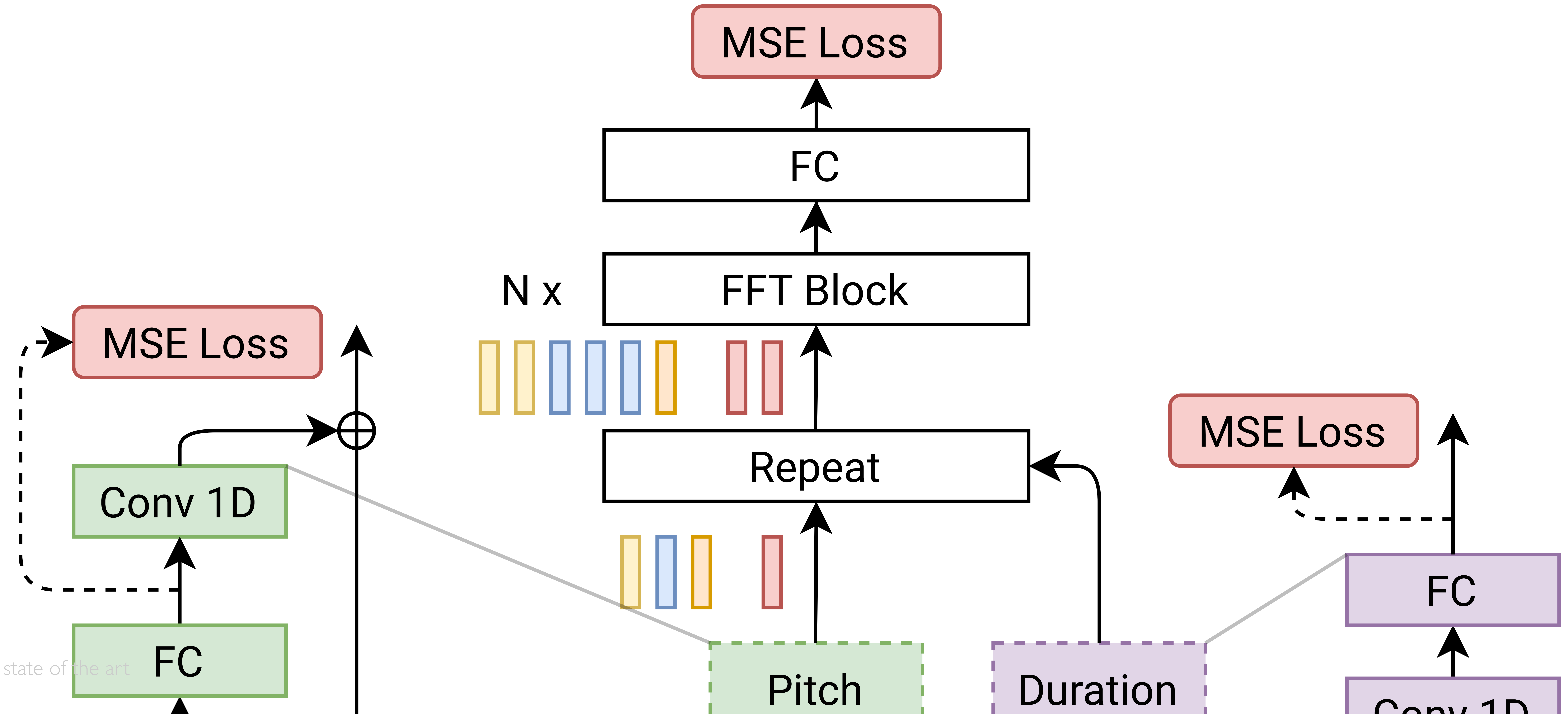
Understanding architecture diagrams : flow of information

FC
Conv 1D
Conv 1D
Pitch Predictor
Duration Predictor
N x
FFT Block
Embedding

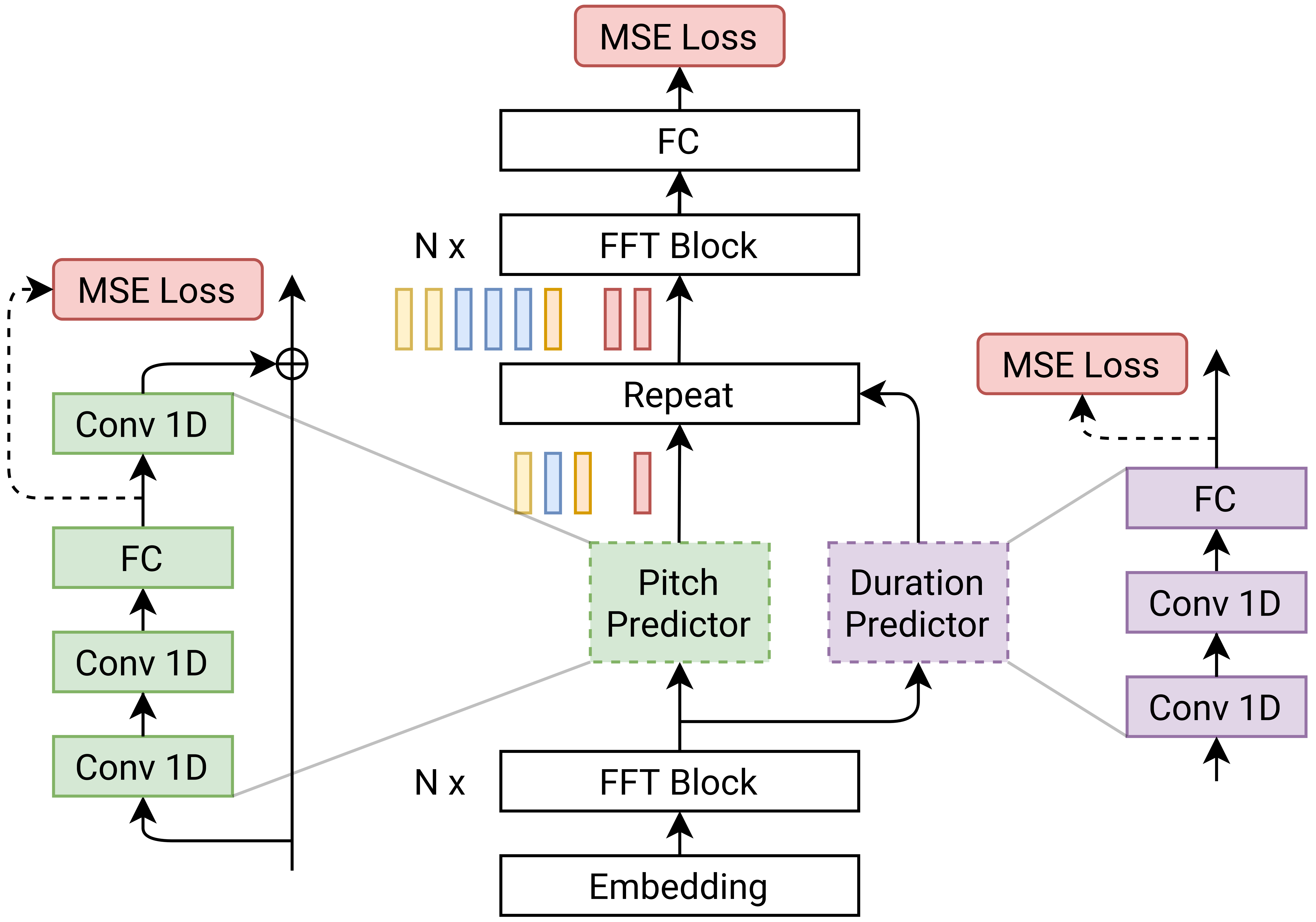
Understanding architecture diagrams : combining information



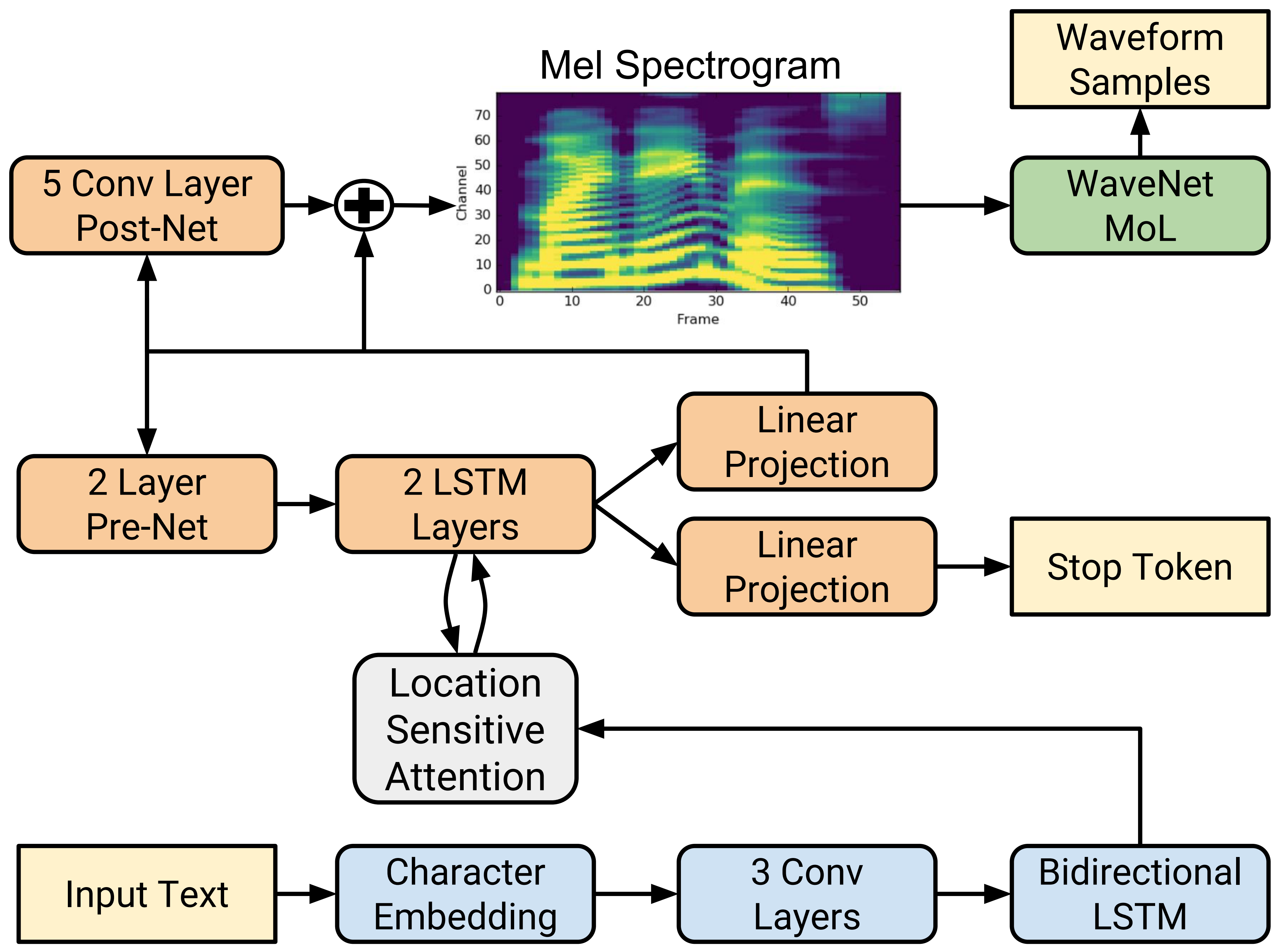
Understanding architecture diagrams : loss

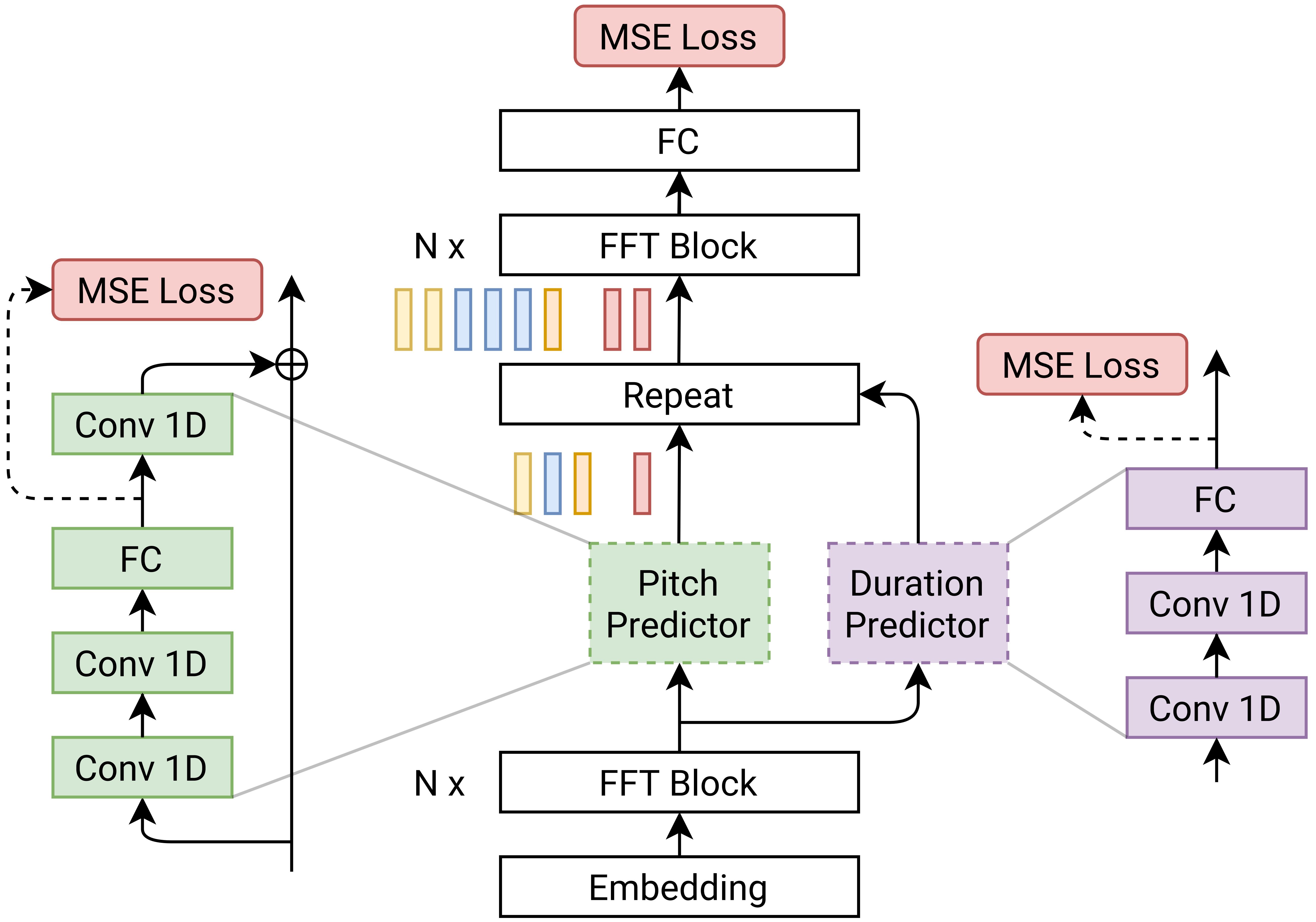


Understanding architecture diagrams : training vs. inference



Neural building blocks : layers





What next?

- Neural vocoders
- Approaches based on language models

- Plus some selection of
  - very recent models
  - tasks beyond TTS