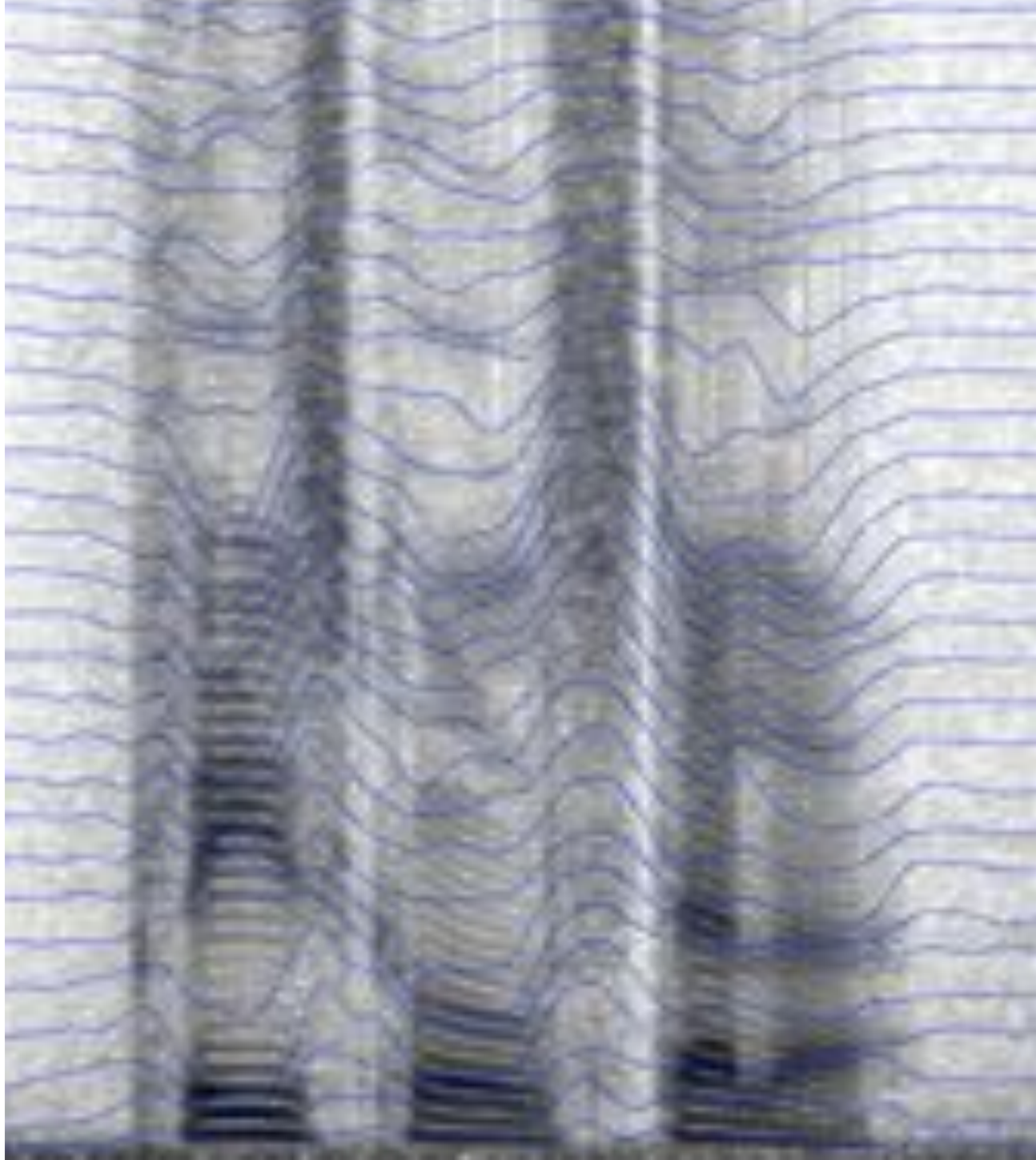
Speech Synthesis

Simon King
University of Edinburgh



Speech synthesis using Neural Networks

- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network

What you should already know

- Text processing in the front end
  - what the available linguistic features are
  - how they can be **flattened** on to the phonetic sequence
  - how **categorical** linguistic features can be treated as **binary**
- HMM-based speech synthesis
  - how questions in a **regression tree** use those binary features
  - typical **speech parameters** used by vocoders

"**one-hot**" encoding

also known as

`1-of-K` or `1-of-N`

Speech synthesis using Neural Networks

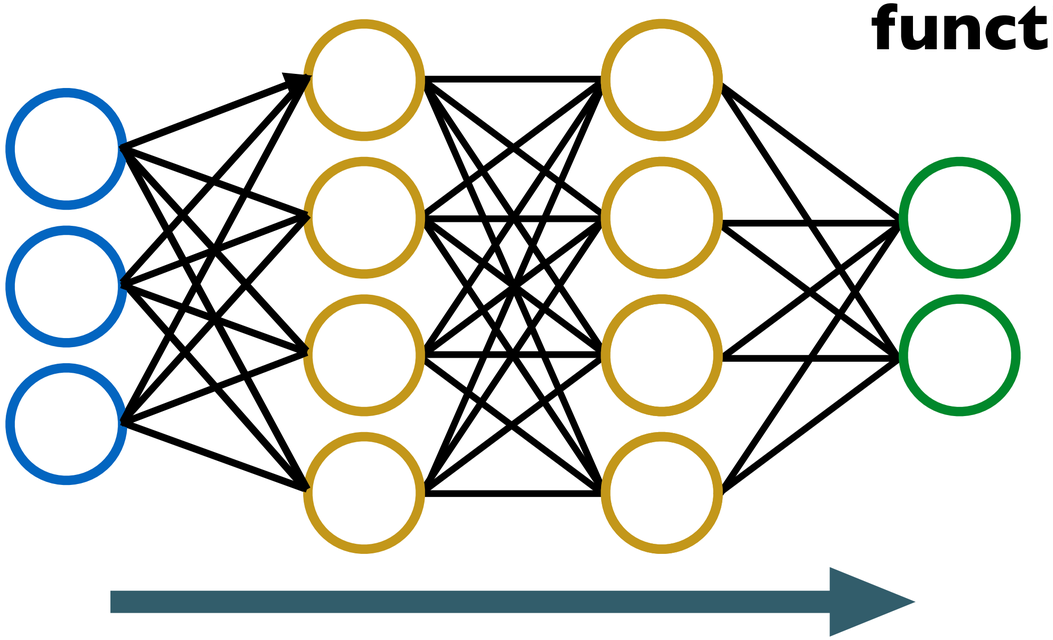
- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network

A simple "feed forward" neural network

directed connections, each with a **weight**

a hidden **layer**

units (or "neurons"), each with an **activation function**



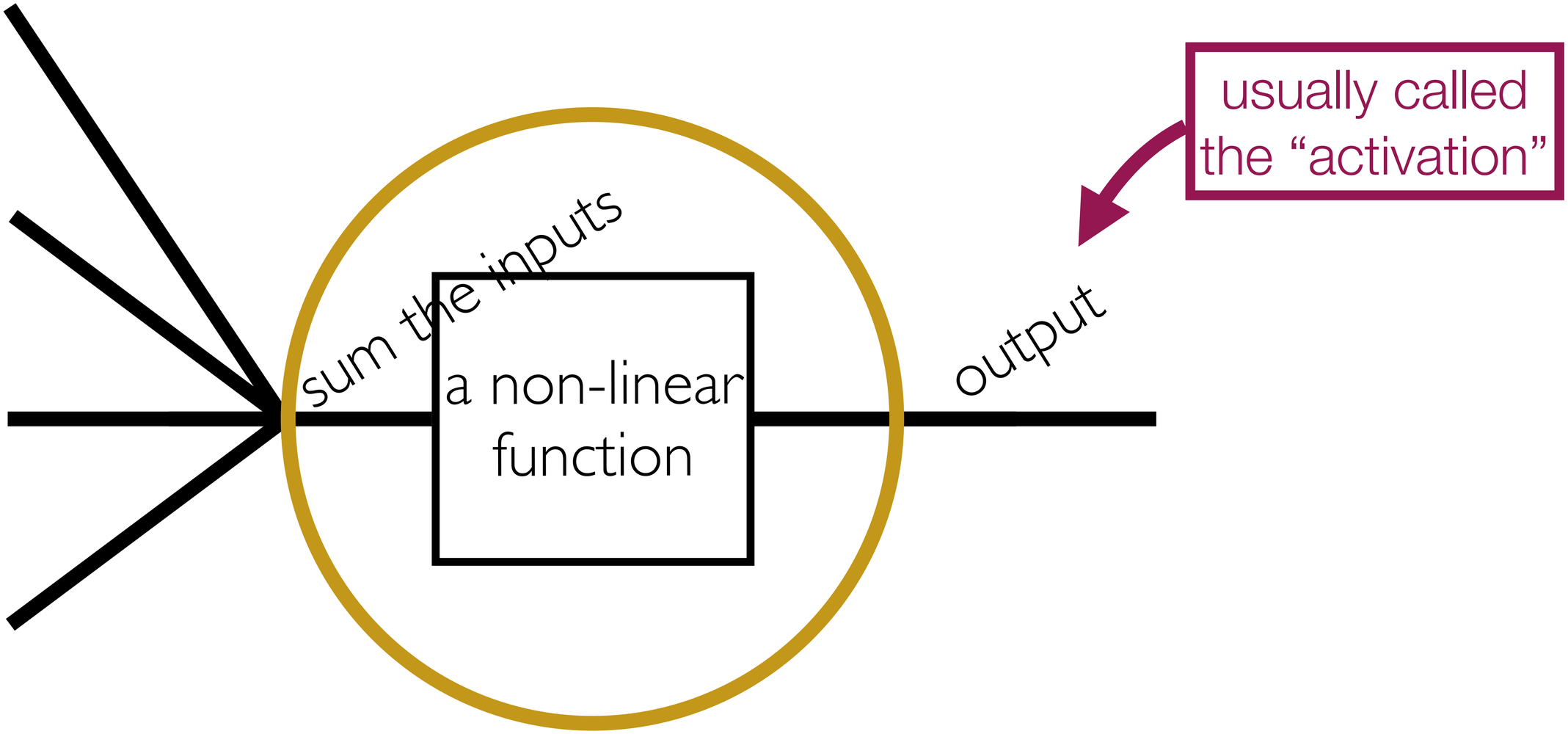
input **layer**

output **layer**

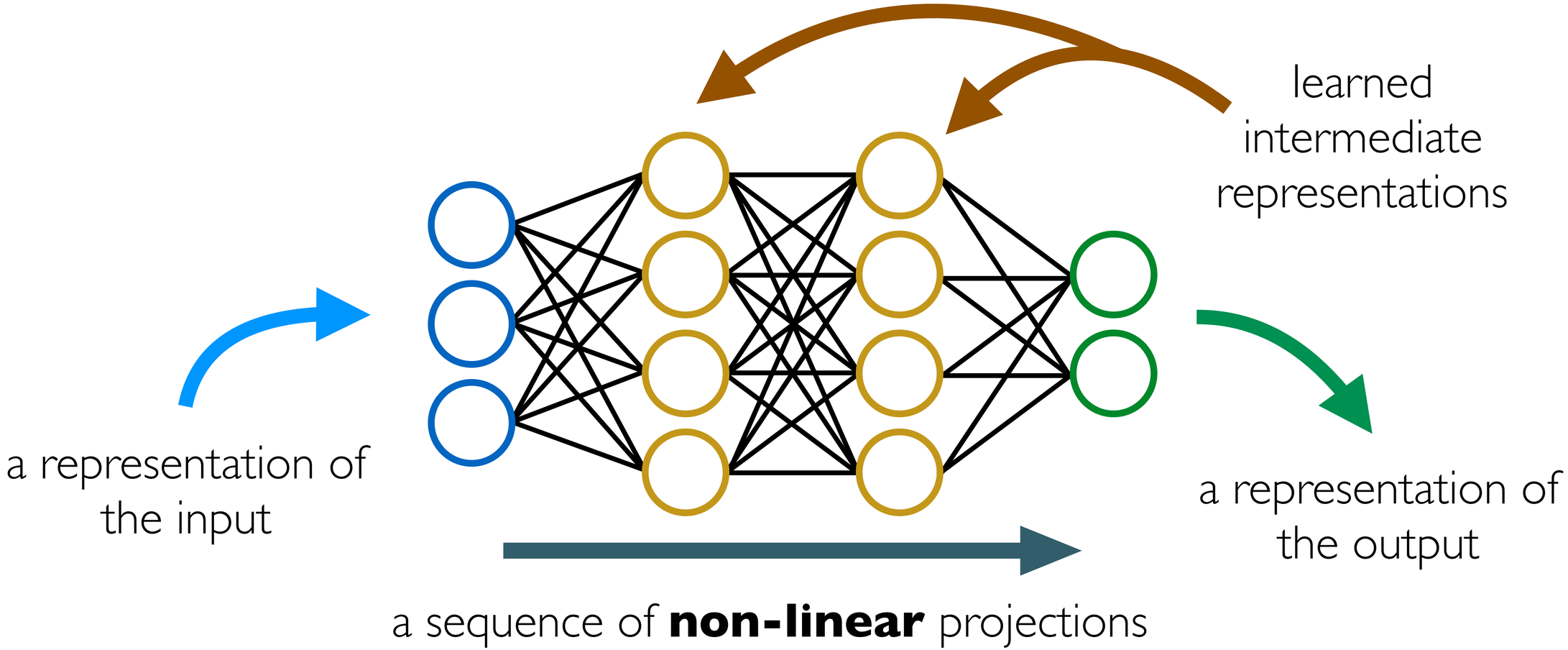
a weight **matrix**

information flows in this direction

What is a unit, and what does it do?



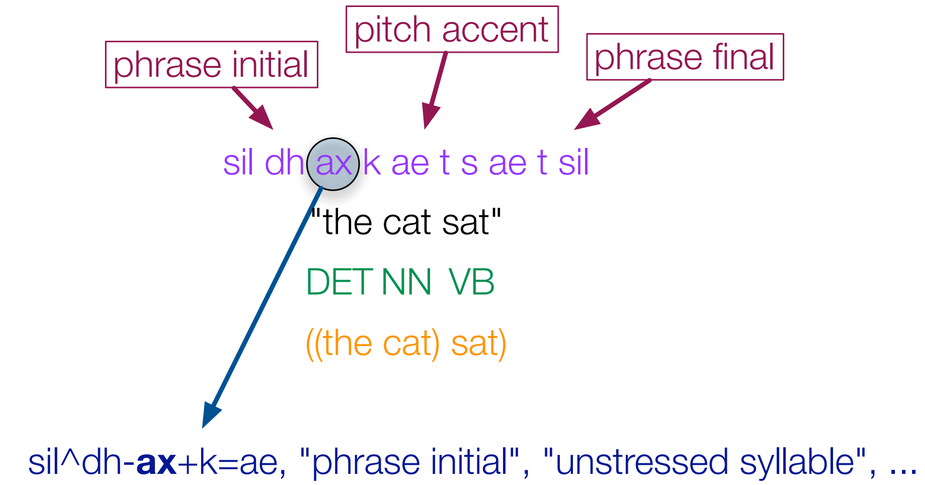
What are all those layers for?



Speech synthesis using Neural Networks

- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network

Synthesis with a neural network: flatten the linguistic structure



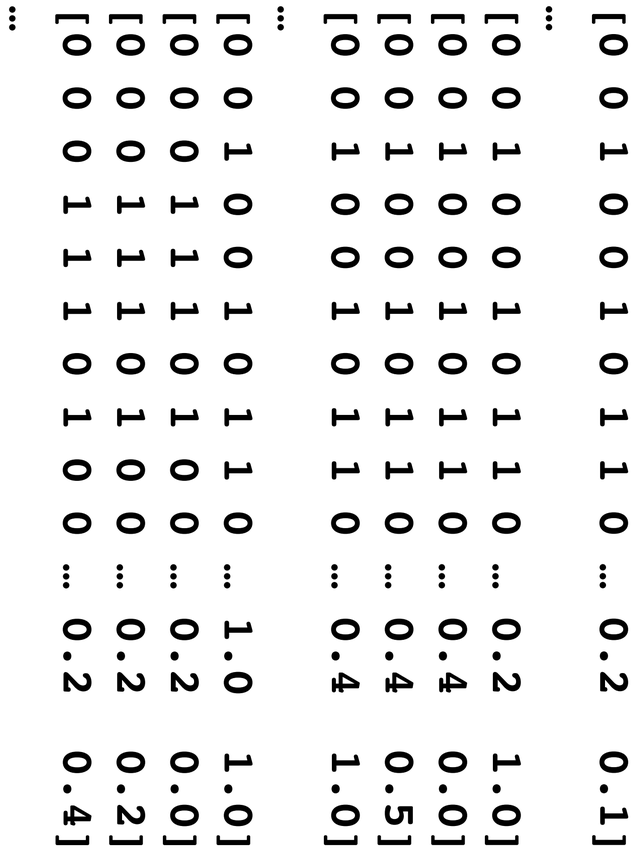

```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

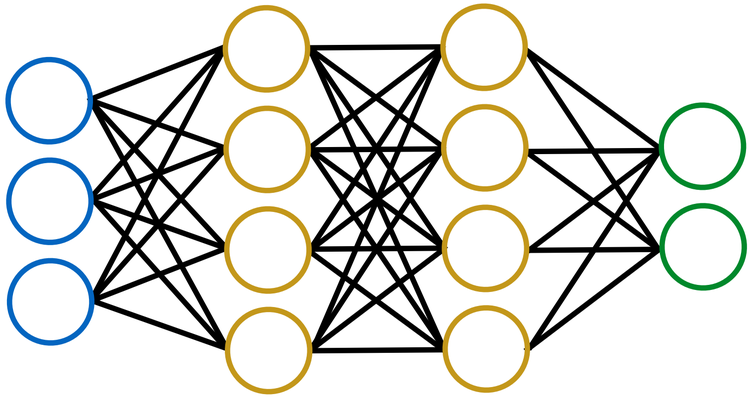
Synthesis with a neural network: encode the context-dependent-phones

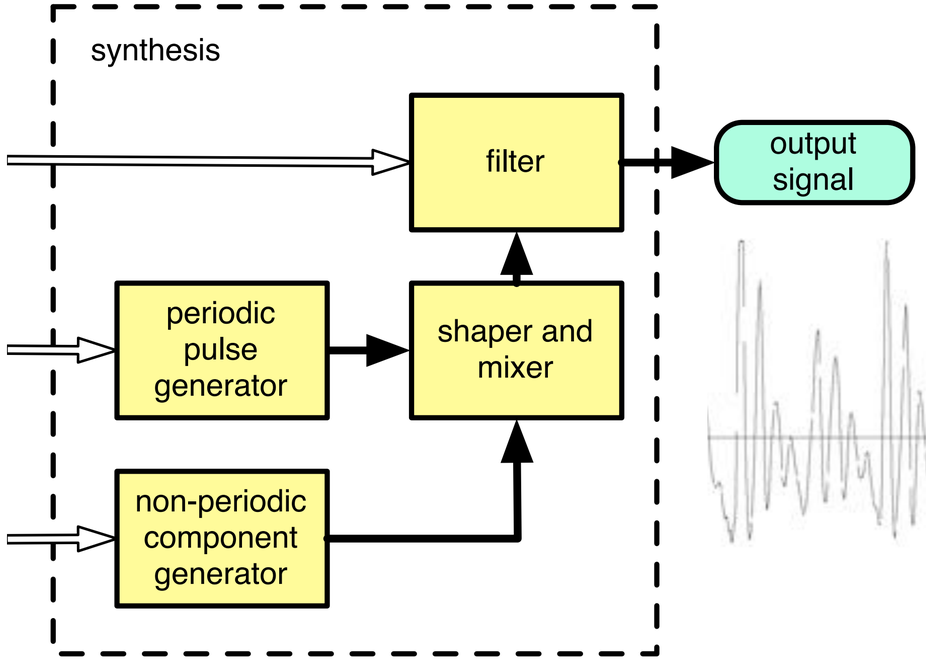

```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]
…
```

Synthesis with a neural network: generate



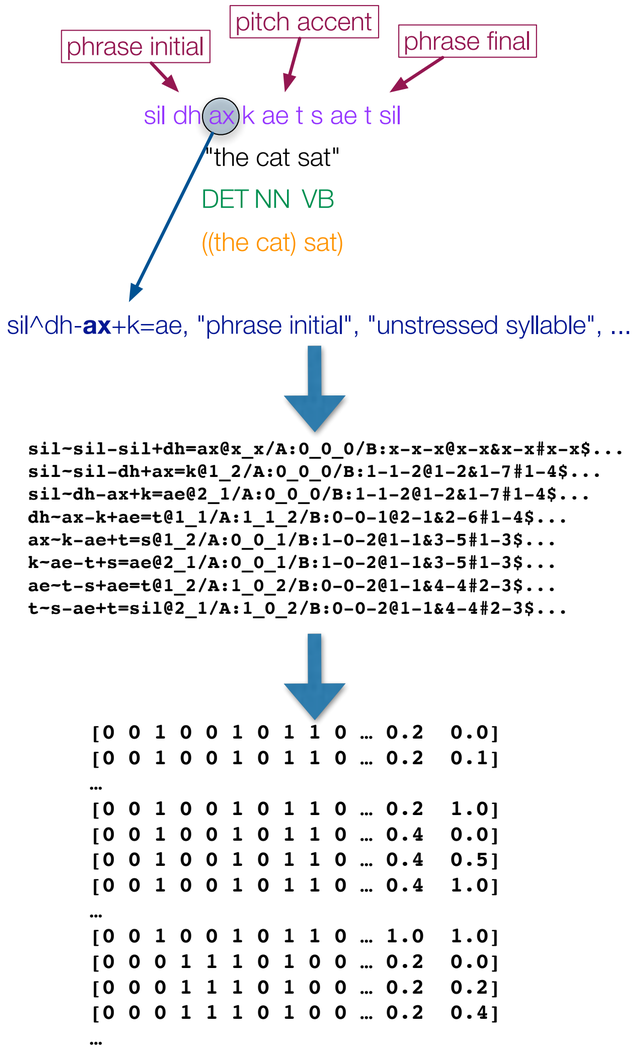




How to construct the sequence of input features

```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]
…
```

Preparing the input



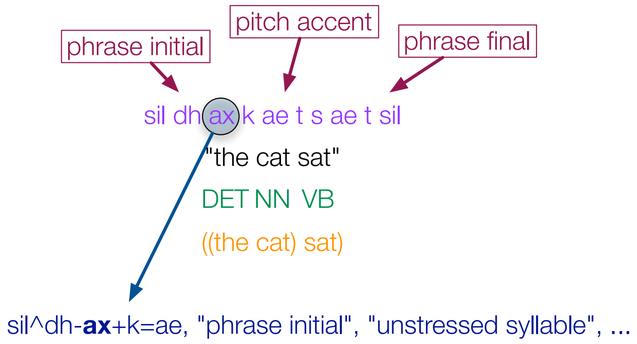
- Run the front end
  - obtain linguistic specification

linguistic timescale

time is now at a fixed framerate

Preparing the input: flatten linguistic specification

linguistic timescale: phones



Preparing the input:
a sequence of context-dependent phones

linguistic timescale: phones

“Please call . . .”

```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-O+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```

quinphone

positional features
(e.g., position of phone in syllable)

POS features

This is the sequence of model names that we would use in HMM-based speech synthesis

Preparing the input:
predict durations at the subphone level

linguistic timescale: subphones

"Please call . . ."

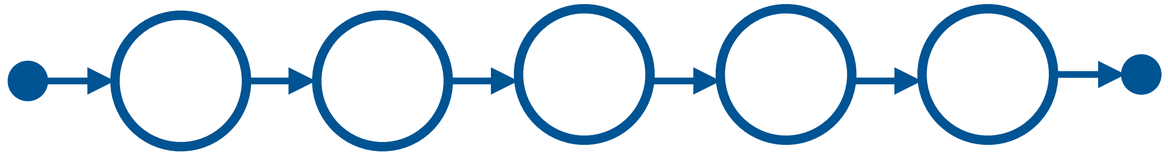
```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-O+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```

linguistic timescale: subphones

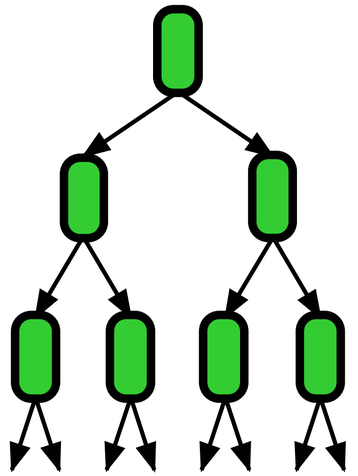
What is the "subphone" ?

- All early DNN systems employ HMMs as a sub-phonetic "clock"
  - duration is then modelled at the **state** (i.e, subphone) level

```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
```



duration (in frames)	2	1	3	1	3



regression tree duration model

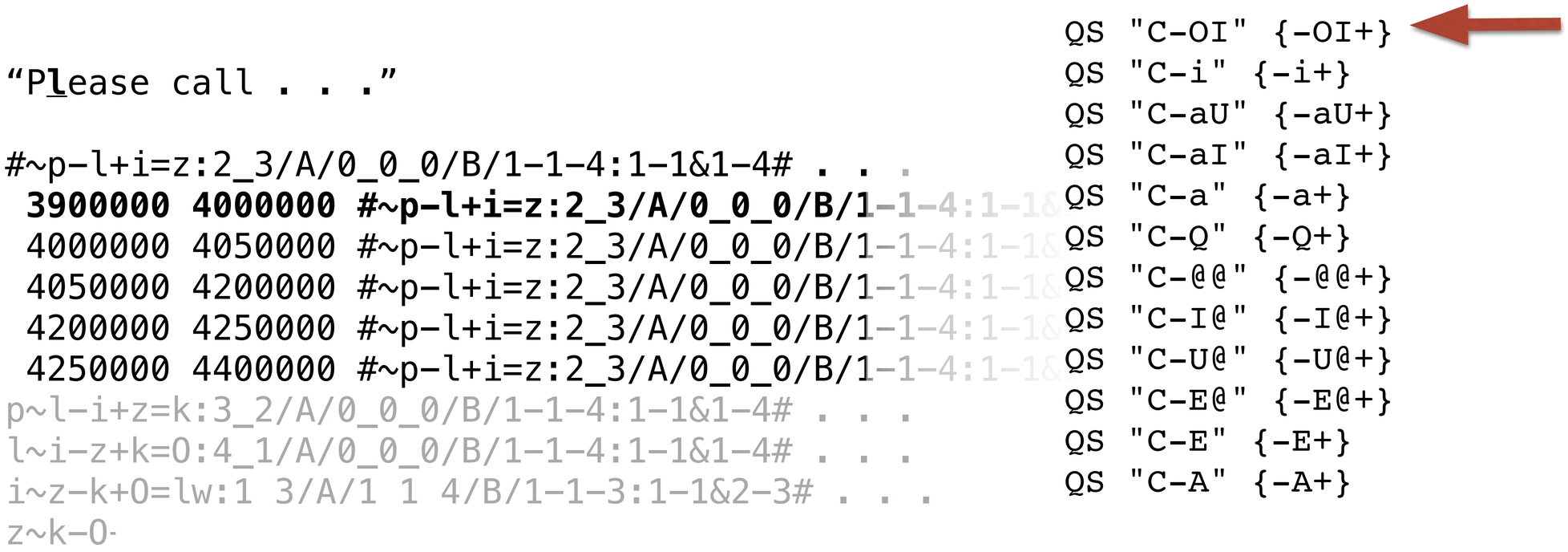
Preparing the input:
predict durations at the subphone level

linguistic timescale: subphones

"Please call . . ."

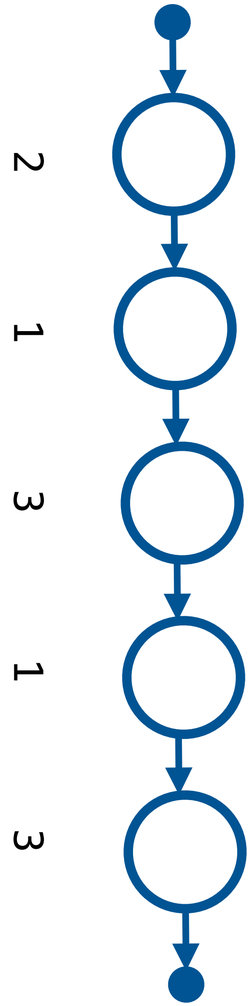
```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                  #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-O+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```

Preparing the input: convert each state of each context-dependent phone to a vector of binary features



Position-within-phone and *position-within-state* features

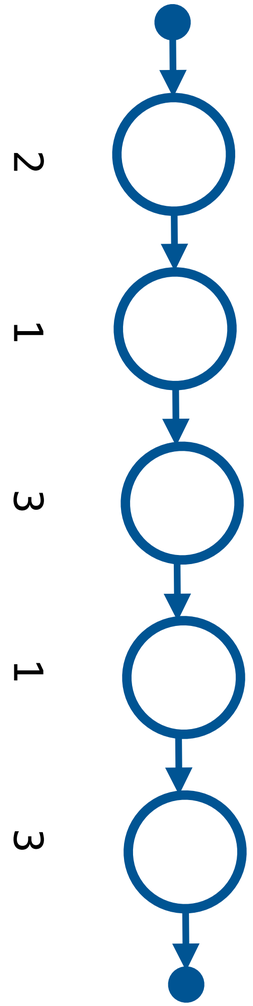
time is now at a fixed framerate



00000100000100010000 . . .

time is now at a fixed framerate

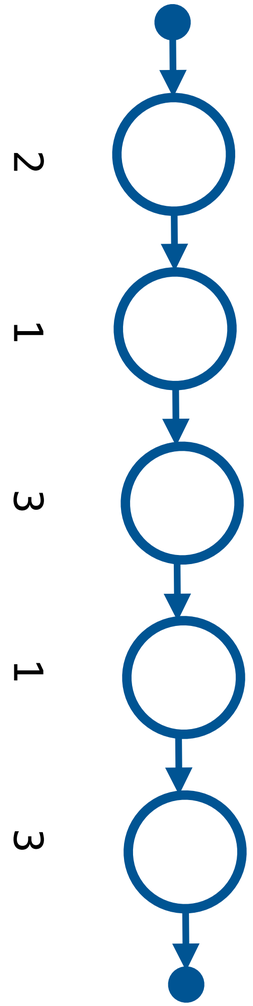
Position-within-phone = state counter




```
0000010000010001000000 . . . 2
0000010000010001000000 . . . 3
0000010000010001000000 . . . 4
0000010000010001000000 . . . 5
0000010000010001000000 . . . 6
```

Position-within-state feature

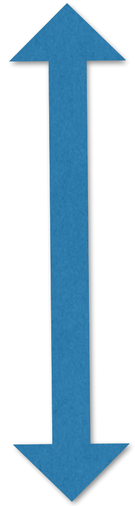
time is now at a fixed framerate




```
000001000001000100000 . . . 2 0.50
000001000001000100000 . . . 2 1.00

000001000001000100000 . . . 2 0.50
000001000001000100000 . . . 2 1.00
000001000001000100000 . . . 3 1.00
000001000001000100000 . . . 4 0.33
000001000001000100000 . . . 4 0.66
000001000001000100000 . . . 4 1.00
000001000001000100000 . . . 5 1.00
000001000001000100000 . . . 6 0.33
000001000001000100000 . . . 6 0.66
000001000001000100000 . . . 6 1.00

000001000001000100000 . . . 6 0.33
000001000001000100000 . . . 6 0.66
000001000001000100000 . . . 6 1.00
```



[**l**] in the context
#~p-**l**+i=z:2_3/. . .

with a duration of
10 frames (50ms)

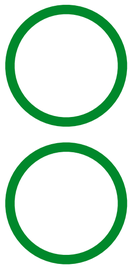
Speech synthesis using Neural Networks

- what is a Neural Network?
- doing Text-to-Speech with a Neural Network
- training a Neural Network

Preparing the inputs and outputs for training

- Inputs
  - linguistic features
  - plus positional features ('counters')
  - re-write as vectors
    - [0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 ….. 0.2 0.1]

- Outputs
  - same speech features (vocoder parameters) used in HMM synthesis



- Form input/output pairs, one pair per frame (e.g., every 5 msec)
  - how to get the alignment?

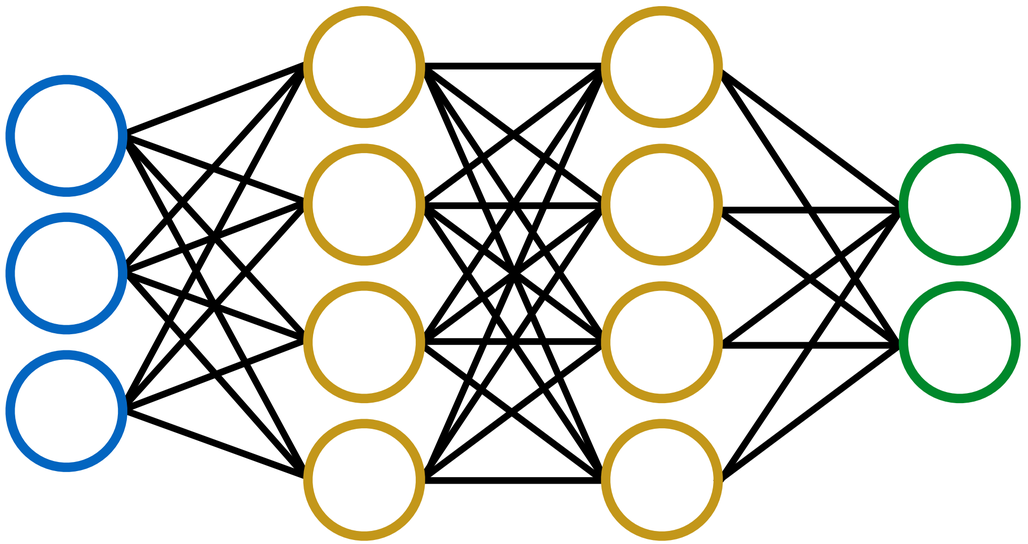
Training a neural network: pairs of input/output vectors

```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]        [0.12 2.33 2.01 0.32 6.33 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]        [0.43 2.11 1.99 0.39 4.83 … ]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]        [1.11 2.01 1.87 0.36 2.14 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]        [1.52 1.82 1.89 0.34 1.04 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]        [1.79 1.74 2.21 0.33 0.65 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]        [1.65 1.58 2.68 0.31 0.73 … ]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]        [1.55 1.03 3.44 0.30 1.07 … ]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]        [1.92 0.99 3.89 0.29 1.45 … ]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]        [2.38 1.13 4.02 0.28 1.98 … ]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]        [2.65 1.98 3.94 0.29 2.16 … ]
…
```

Training a neural network: back-propagation

input

[1 0 0 1]



output

[ ]

[ ]

target

[0.12 2.33]

Orientation

- Simple neural networks
  - feed-forward architecture

a straightforward replacement for the regression tree

- Constructing the input features
  - converting categorical features to binary
  - mapping linguistic timescale to fixed frame rate using the duration model

Early work borrowed a duration model from an HMM system. Later work uses a better duration model

What next?

- Neural networks for speech synthesis
  - from this point onwards, we must rely on **reading the literature**

- Hybrid synthesis
  - in the next module, we'll use our Neural Network as the basis for an **ASF target cost function**