Neural Networks via FluCoMa In Max

Music Hackspace

Ted Moore

tedmooremusic.com

ted@tedmooremusic.com

FluCoMa: Fluid Corpus Manipulation

- Toolbox for Programmatic Data Mining of Sound Banks
- Integrating Machine Listening and Machine Learning in…
- Max, SuperCollider & Pure Data
- Learning Resources (learn.flucoma.org)
- Discourse Community (discourse.flucoma.org)

Ted Moore, (James Bradbury, Owen Green, Jacob Hart, Gerard Roma, & Pierre Alexandre Tremblay)

University of Huddersfield | UK

Plan for Today

- **Training a Neural Network (refresher)**
  - classification
- **Validating a Trained Model**
- **MLP Parameters Overview**
- **Adjusting parameters during training**
  - regression with control data
  - regression with real-time analysis
- ***tapin* and *tapout***
  - wavetable autoencoder (advanced)

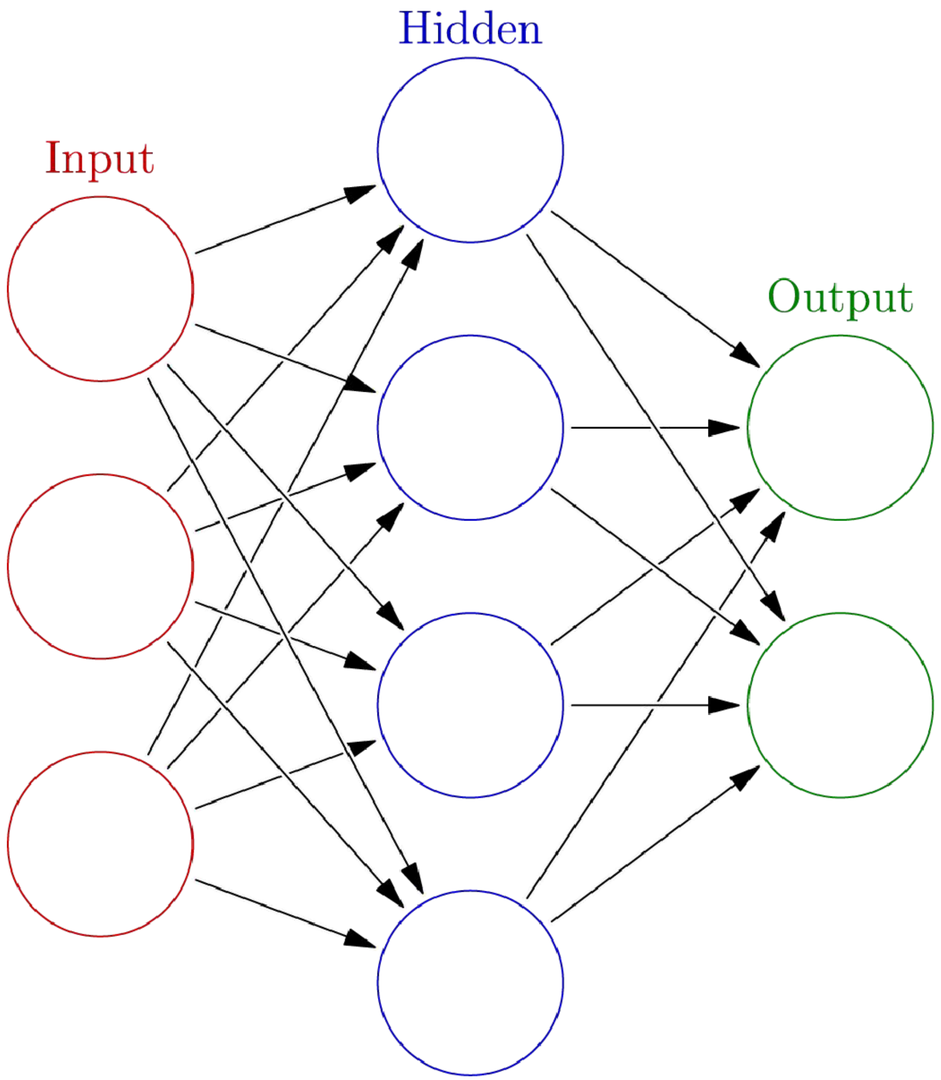
Classification

feed

using a neural network for real-time audio classification

training a machine to hear the way I hear

Neural Network (Multi-Layer Perceptron)



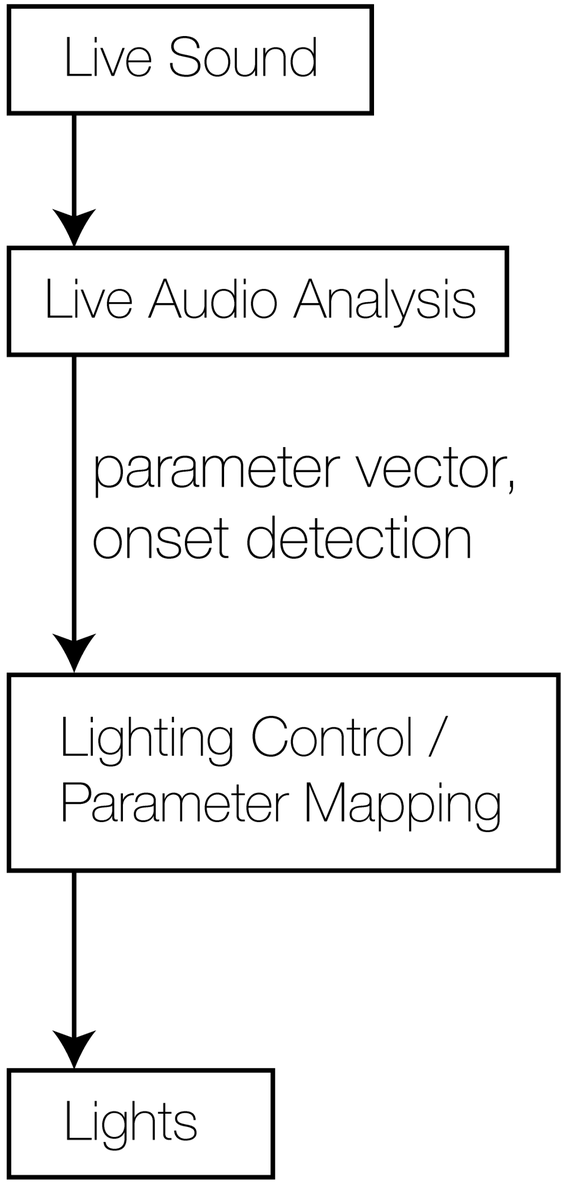
<u>Classification:</u>

a neural network predicts

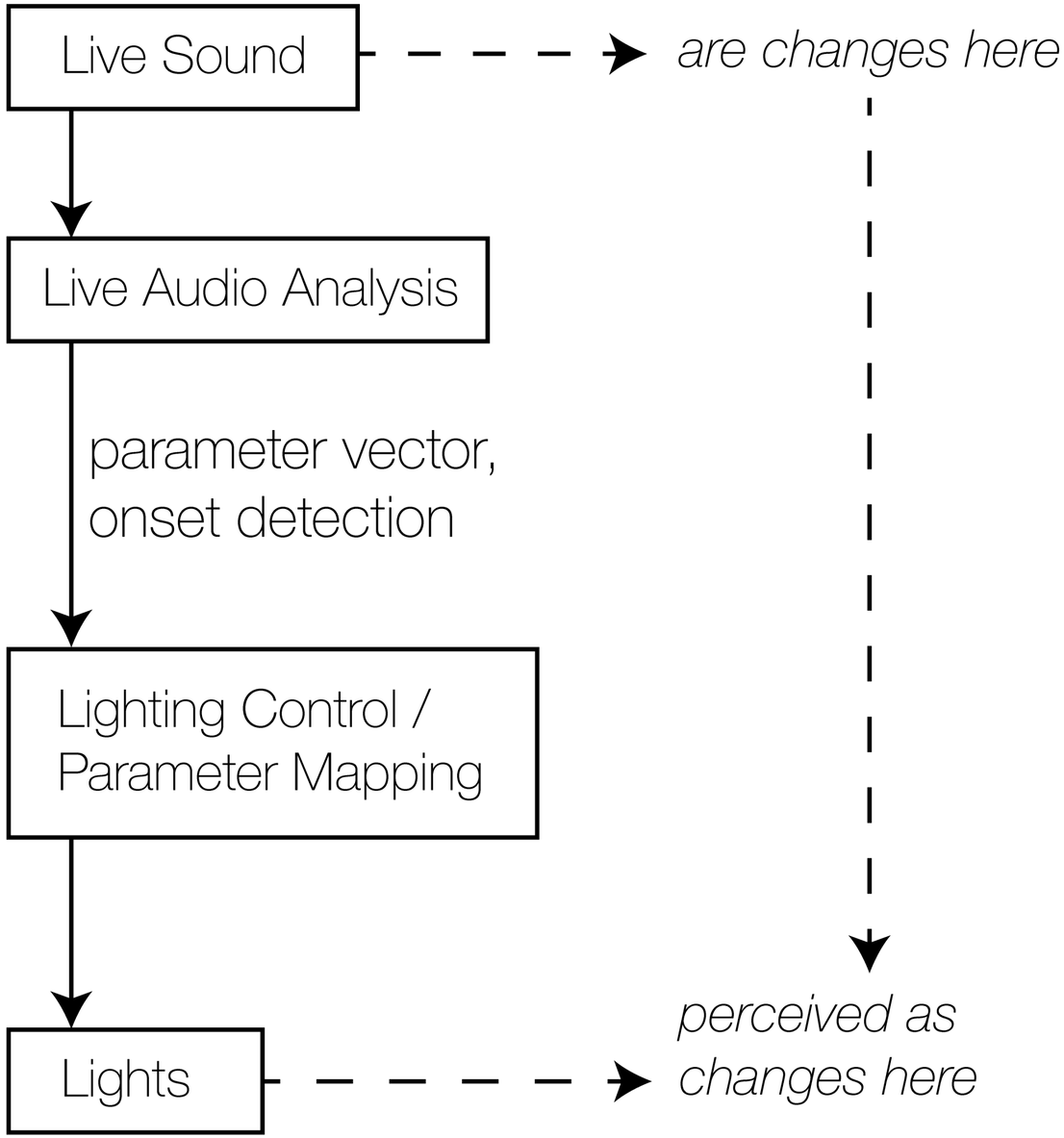
which category (or “class”)

an input belongs to

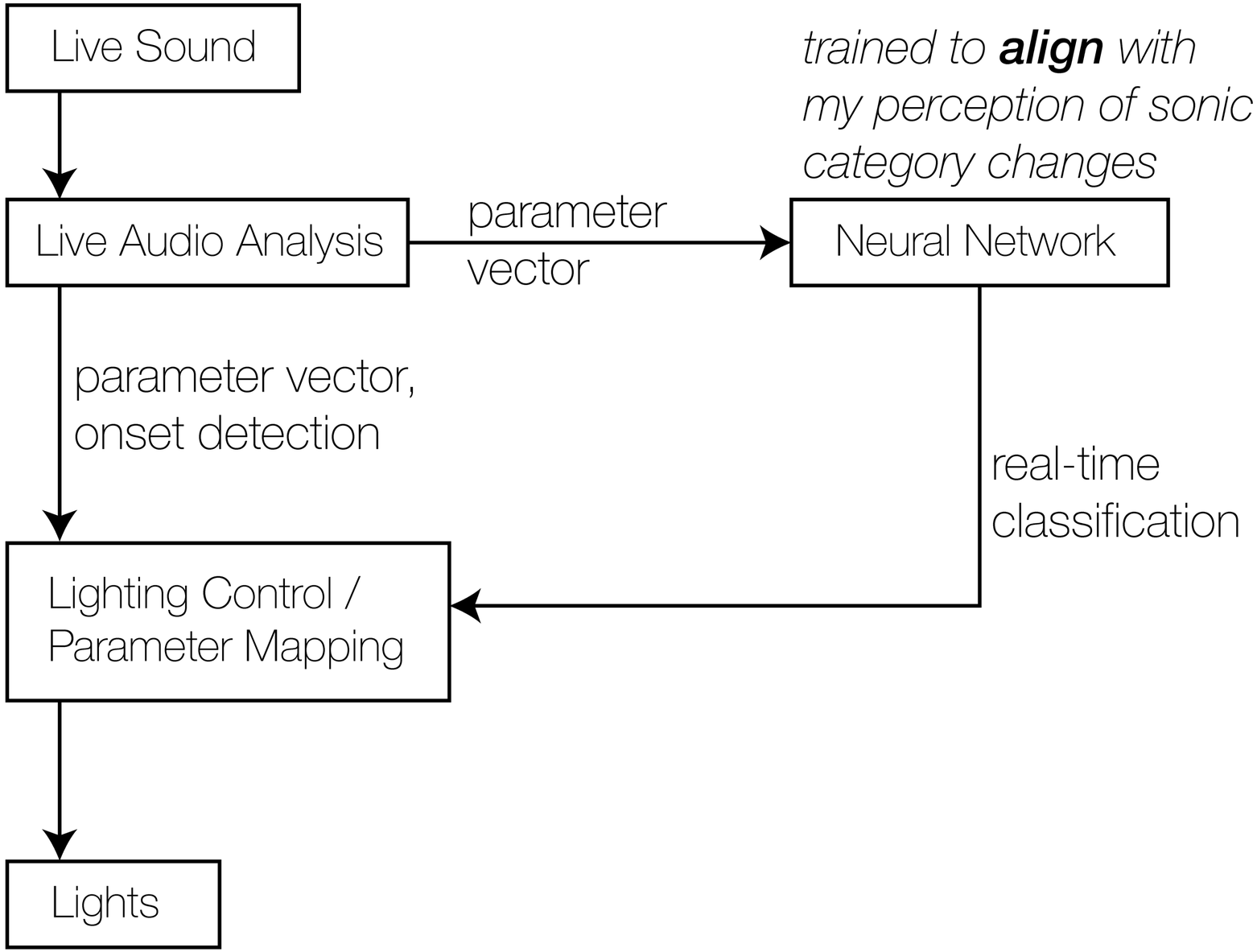
Machine Listening System

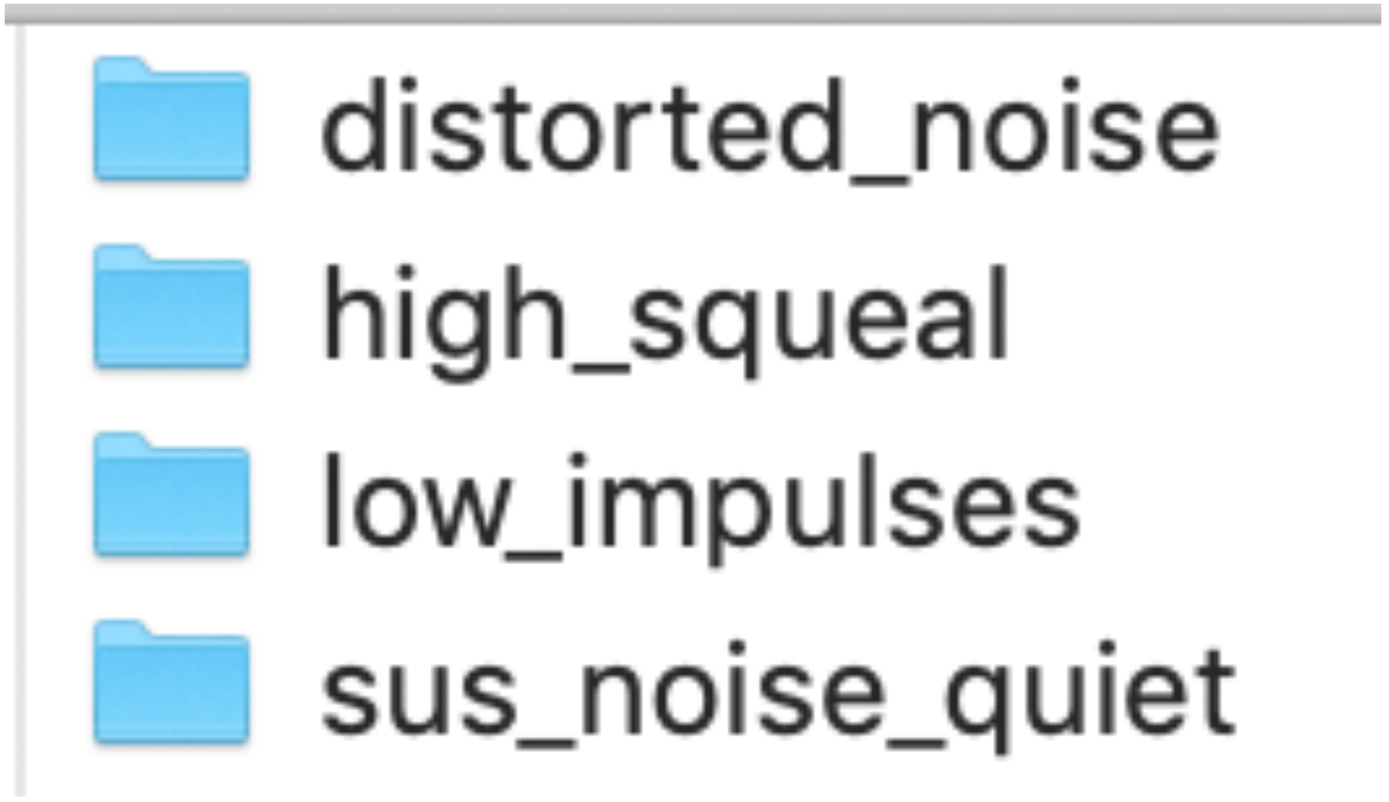


Machine Listening System



Machine Learning System




distorted_noise
high_squeal
low_impulses
sus_noise_quiet

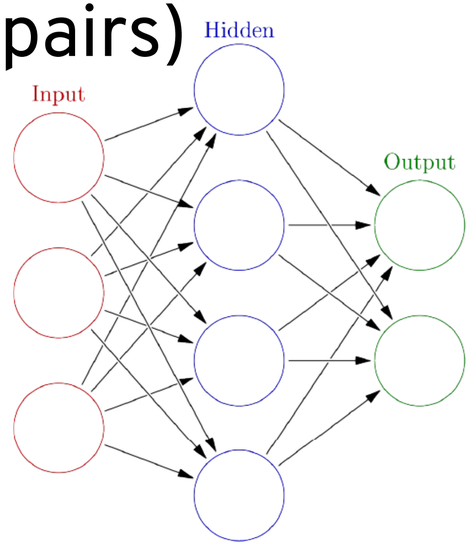
flashing lights warning

Neural Network *Training a Classifier*

trombone examples	oboe examples

Supervised Learning

(uses input-output example pairs)



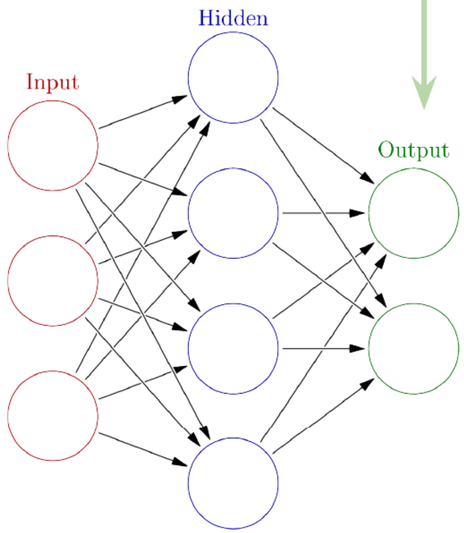
Neural Network *Training a Classifier*

trombone examples

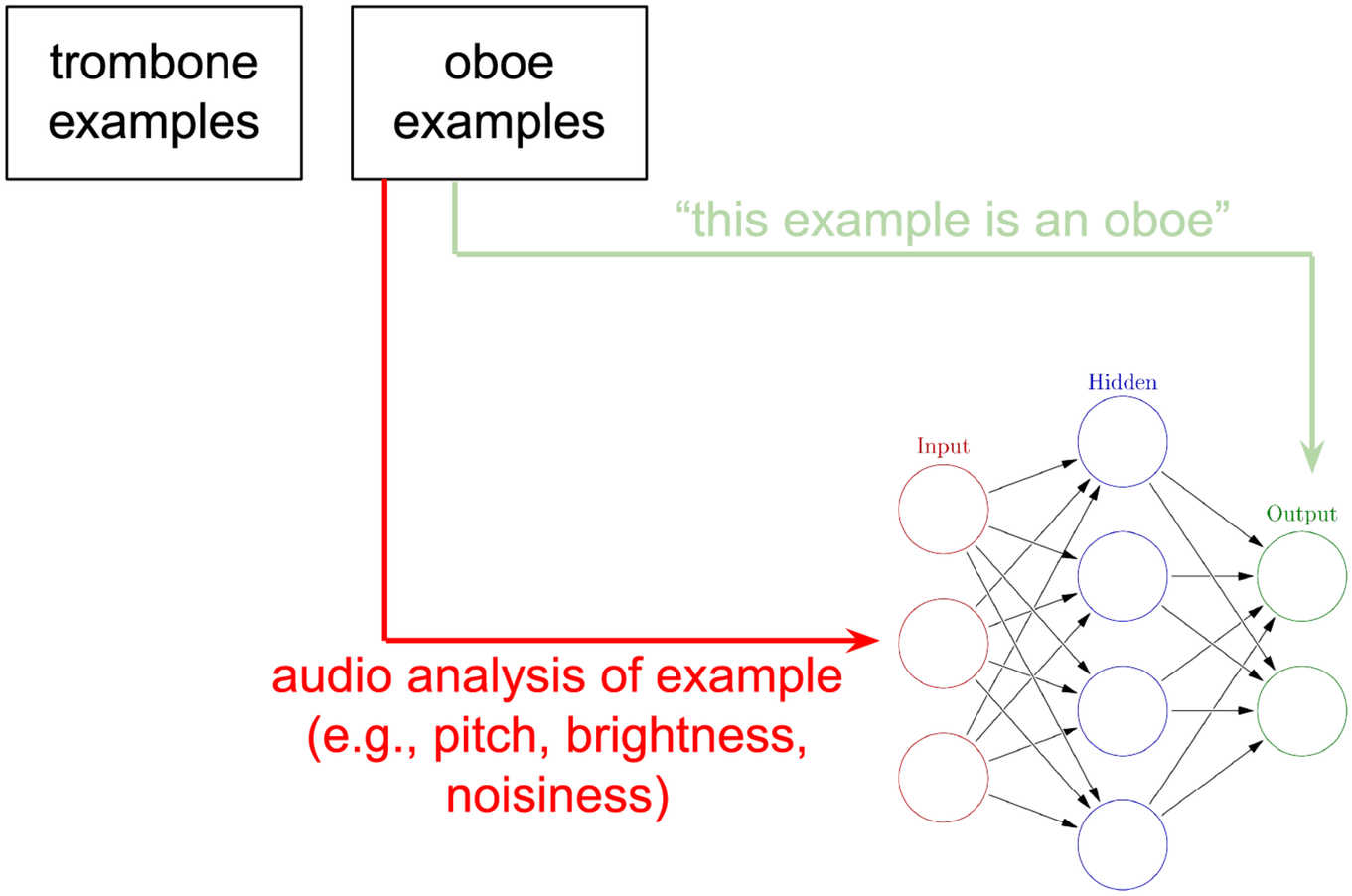
oboe examples

"this example is a trombone"

audio analysis of example (e.g., pitch, brightness, noisiness)



Neural Network *Training a Classifier*



feed-forward

vs.

back-propagation

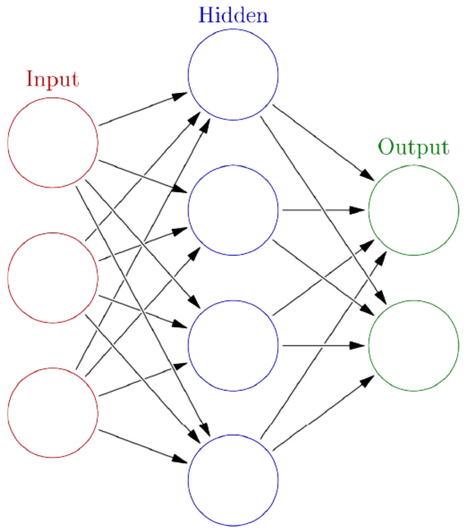
Neural Network *Predicting a Classification*

trombone examples

oboe examples

new example it has never seen before

audio analysis of example (e.g., pitch, brightness, noisiness)



“this new example is most like the oboe examples you showed me before”

(or trombone…)

fluid.mlpclassifier~

oboe vs. trombone

fluid.mlpclassifier~

change color mapping parameters

fluid.mlpclassifier~

validation

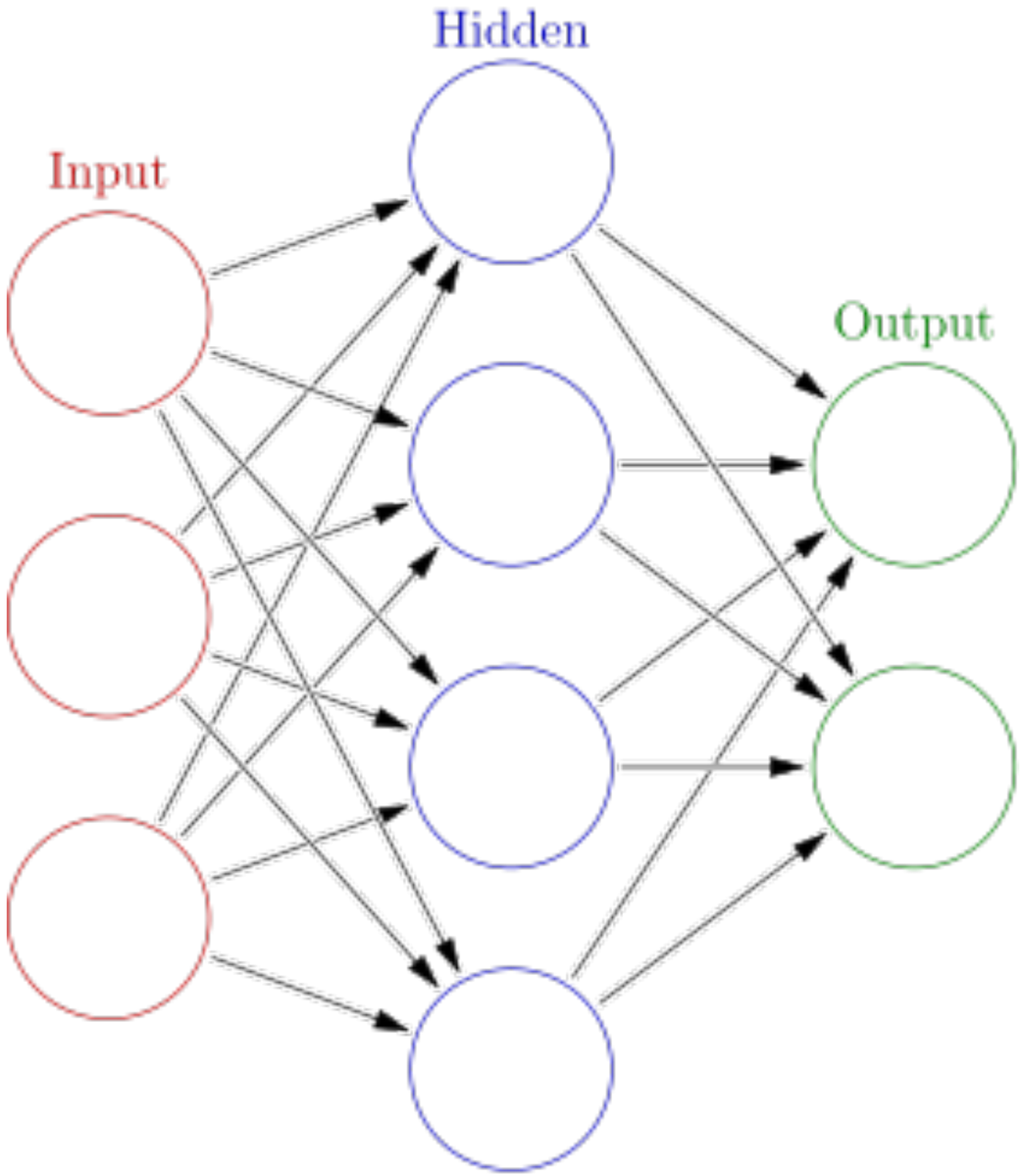
- **overfitting**
  - excellent on training data, poor on anything else
- **underfitting**
  - poor on everything…keep training!

MLP Parameters

@hiddenlayers (list)

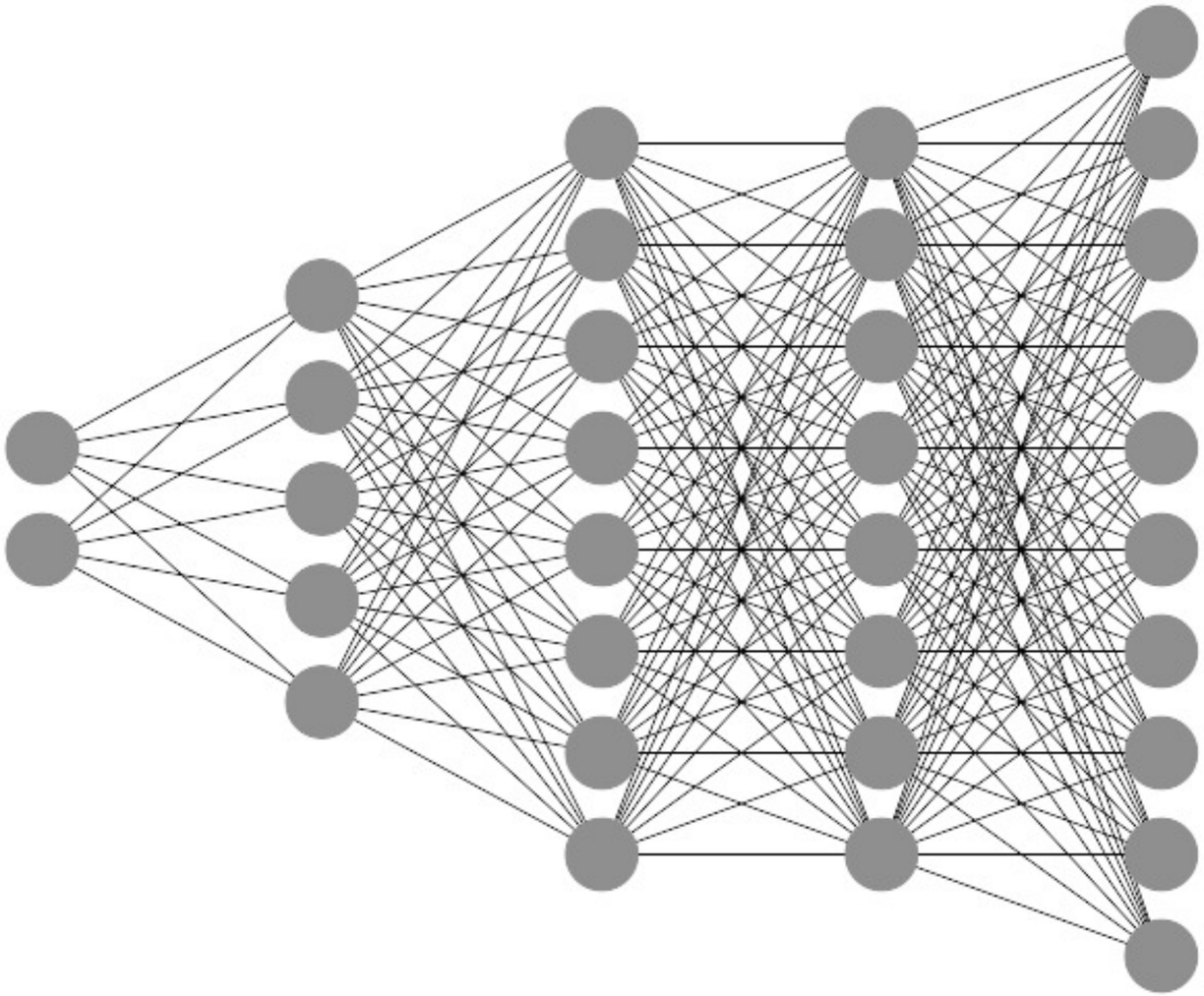
n elements = n layers

Integers = n neurons

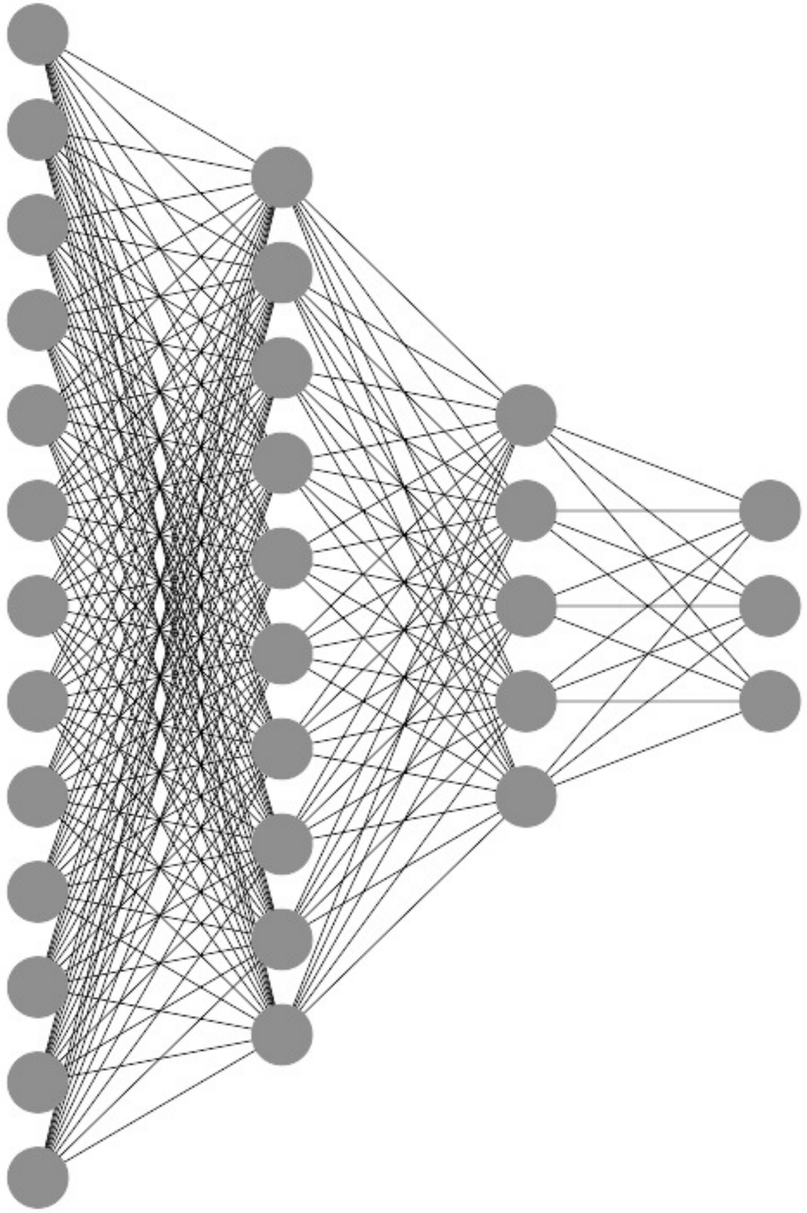


(@hiddenlayers 4)

will reset network



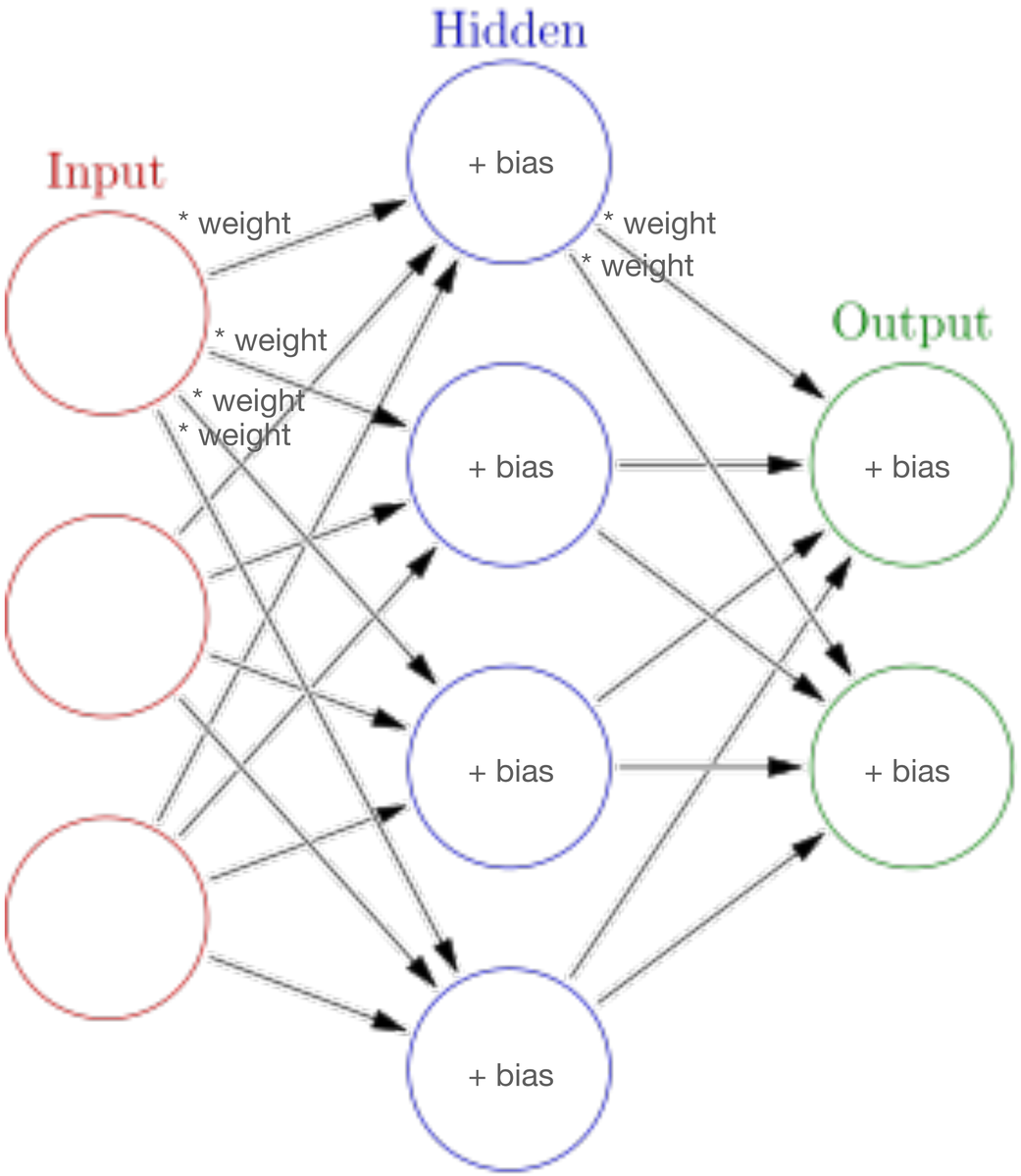
(@hiddenlayers 5 8 8)



(@hiddenlayers 10 5)

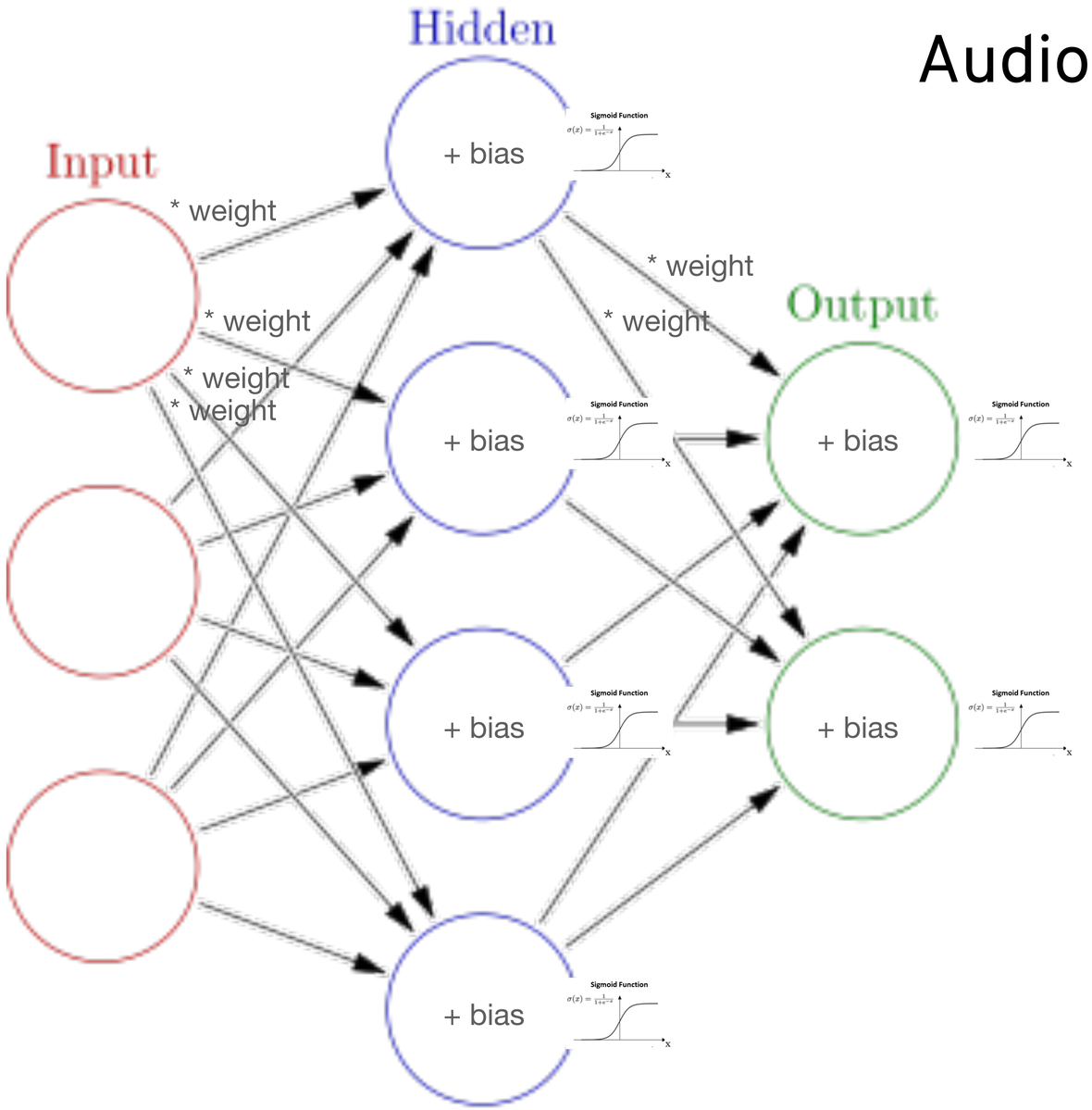
Internal Parameters

weights & biases



@activation

@outputactivation



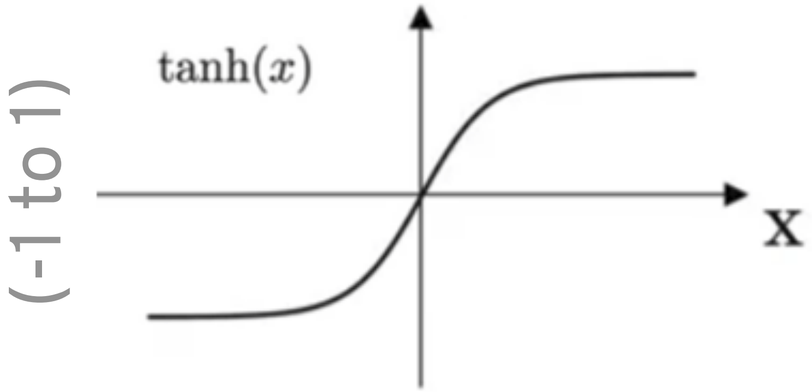
Audio Mixer Metaphor

will reset network

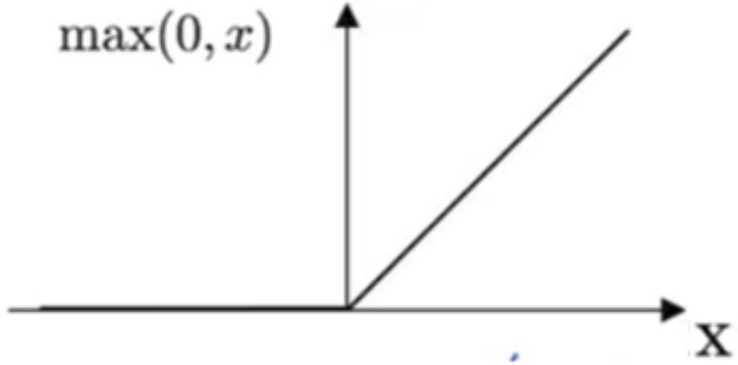
@activation

@outputactivation

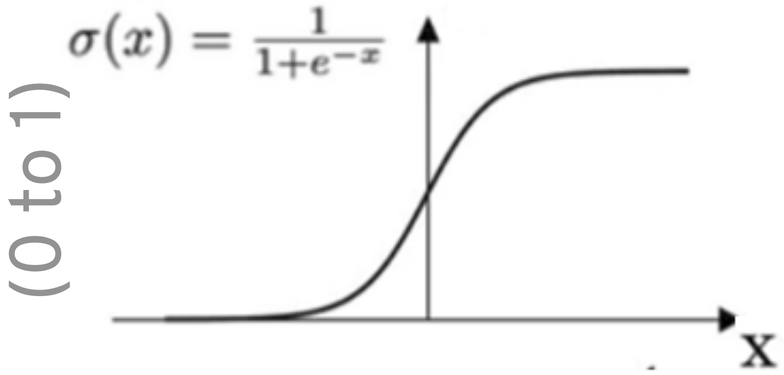
(3) **Hyper Tangent Function**



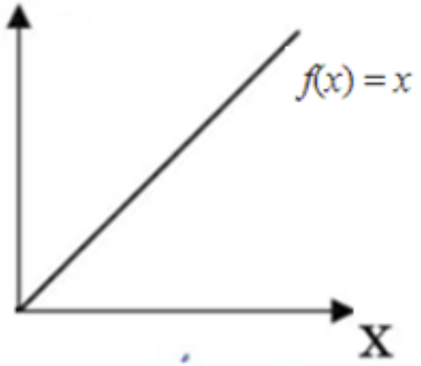
(2) **ReLU Function**



(1) **Sigmoid Function**



(0) **Identity Function**

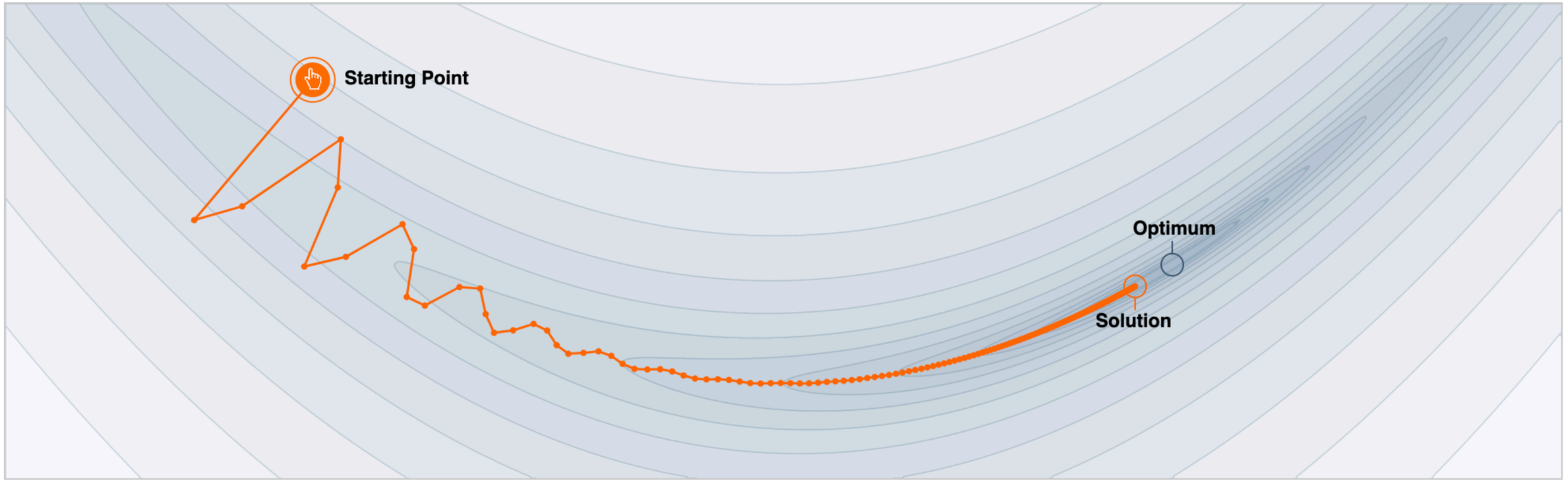


(input = output)

will reset network

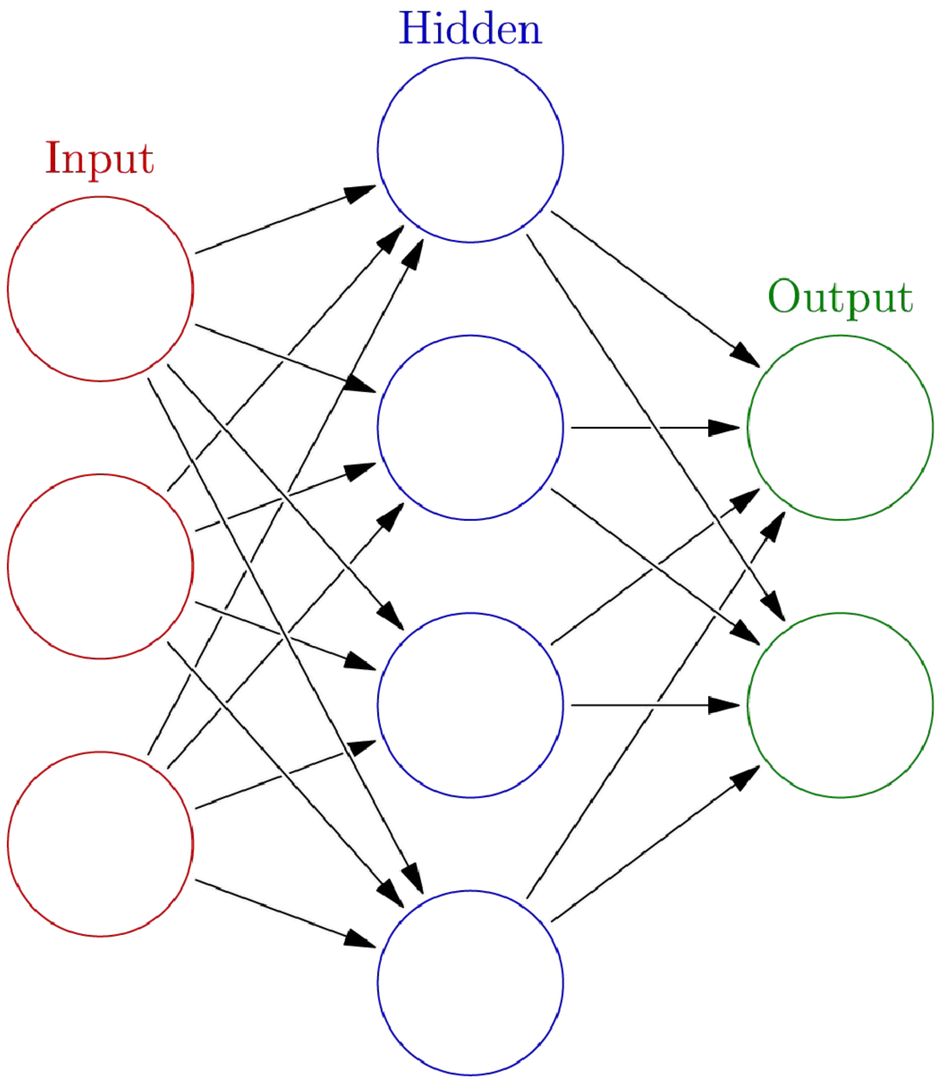
@learnrate	a scalar for indicating how much the neural network should adjust its internal parameters during training (during back-propagation)—starting around 0.1 and decreasing over time is usually a good start
@maxiter	number of times to use all the examples on each “fit” message—also called an “epoch”
@batchsize	number of input examples feed-foward before checking their output pairs and adjusting
@validation	percentage (as a decimal) of training data to hold back (not train on) then test with—when there’s *lots* of data, may be helpful to prevent overfitting

@momentum apply smoothing adjustments



Regression with Control Data

Neural Network



<u>Regression:</u>

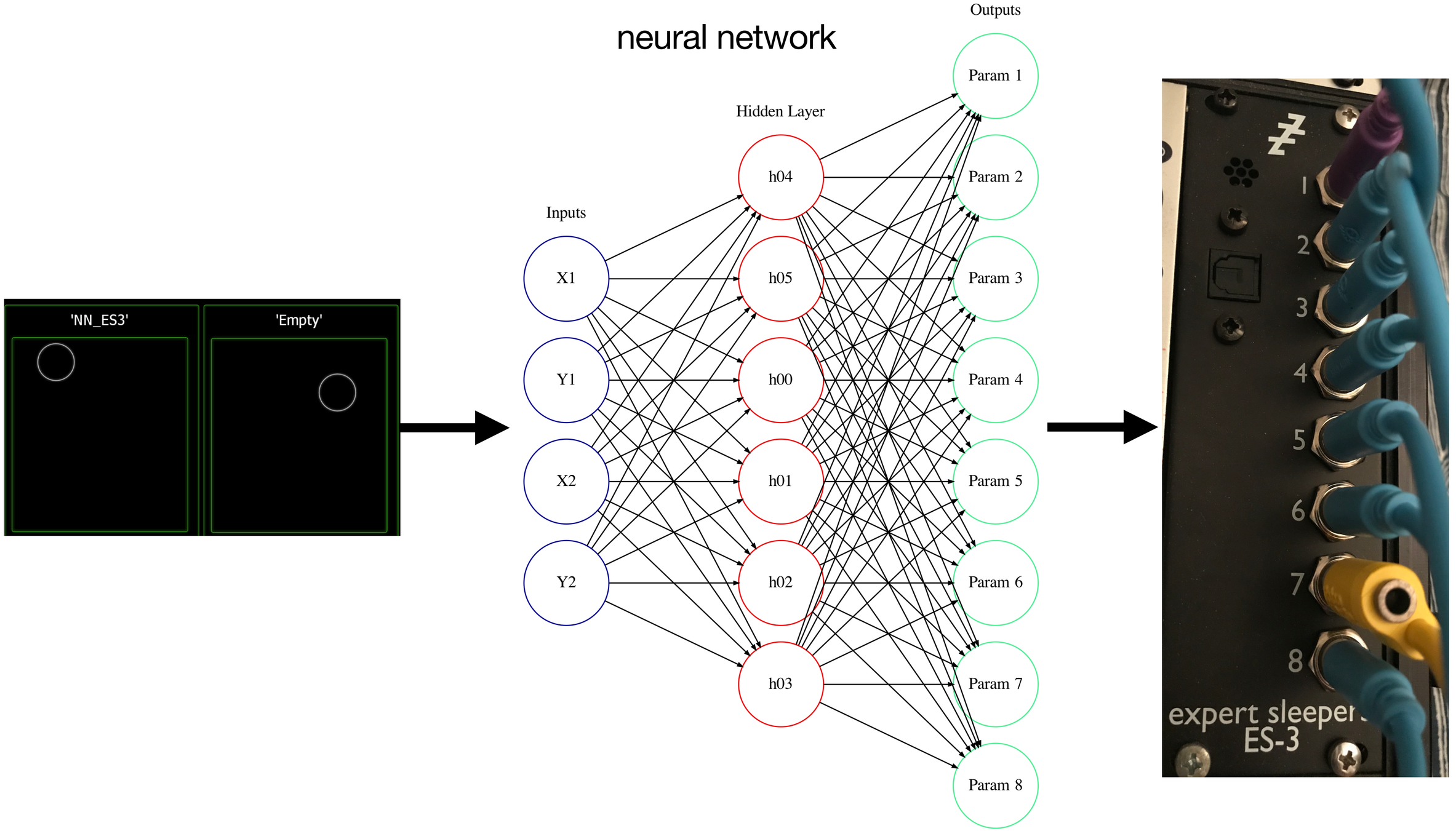
a neural network predicts

output values (to anything)

from

input values (from anything)

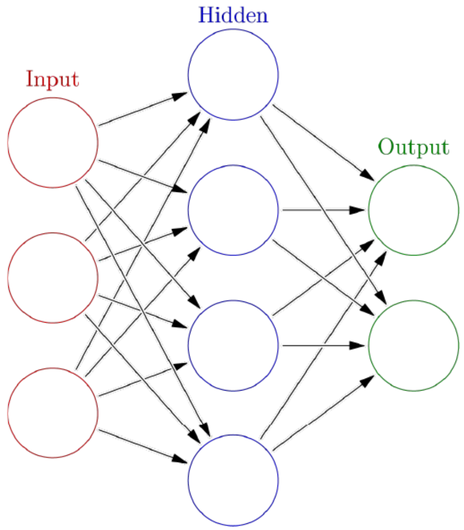

KORG
nanoKONTROL2
TRACK
MARKER
CYCLE
SET
S
M
R
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8
Keith McMillen
QuNeo
0
1
2
3
SPEARPlayer
PShiftDel
Random Delay
SpecSmear
PitchShift
Benjolin
AmpMod


neural network
'NN_ES3'
'Empty'
Inputs
X1
Y1
X2
Y2
Hidden Layer
h04
h05
h00
h01
h02
h03
Outputs
Param 1
Param 2
Param 3
Param 4
Param 5
Param 6
Param 7
Param 8
1
2
3
4
5
6
7
8
expert sleepers
ES-3


Slingerland

Neural Network *Training a Regressor*

identifier	input examples		output examples
“a”:	[0.1, 0.5, 1]	→	[0.5, 0.7]
“b”:	[0, 1, 1]	→	[0.1, 0.1]
“c”:	[1, 0, 1]	→	[0.5, 0.6]
“d”:	[0.2, 0, 1]	→	[0.3, 0.9]
“e”:	[0, 0, 0.5]	→	[0.4, 0]
“f”:	[0.8, 0.2, 1]	→	[0.2, 0.7]
“g”:	[0.9, 0.3, 0]	→	[0.5, 0.6]
“h”:	[0.4, 1, 0.7]	→	[0.6, 0.1]
“i”:	[0.2, 0.3, 1]	→	[0.5, 0.3]
“j”:	[0.1, 1, 0.1]	→	[0.5, 0.7]
“k”:	[0, 1, 1]	→	[0.1, 0.8]
“l”:	[0, 1, 1]	→	[0.5, 0.1]
“m”:	[0, 0, 1]	→	[0.4, 0.2]
“n”:	[0, 0, 1]	→	[0.1, 0.8]



Neural Network *Training a Regressor*

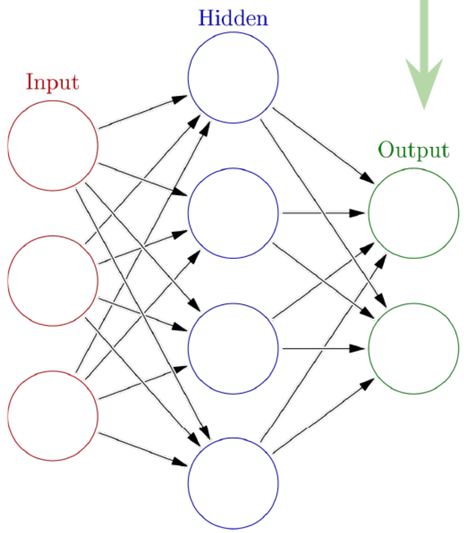
identifier

input examples

output examples

"please return these output values"

"when I input these values…"



Makes a guess on every example and determines how it was wrong. Improves a bit every time.

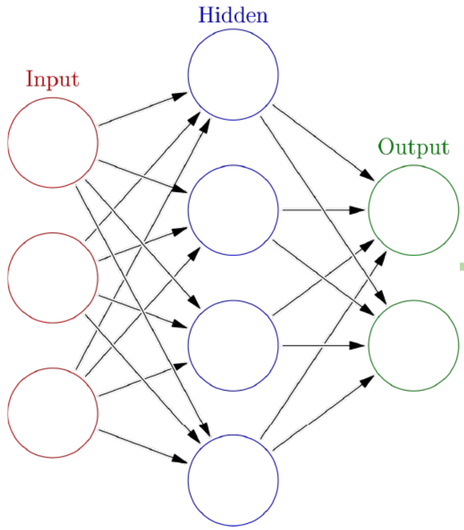
Neural Network *Predicting with Regression*

input examples

output examples

“here are some values you’ve maybe never seen before”

[0.6, 0.2, 0.4]



“here’s an output that would ‘fit’ in your training data”

[0.9, 0.8]

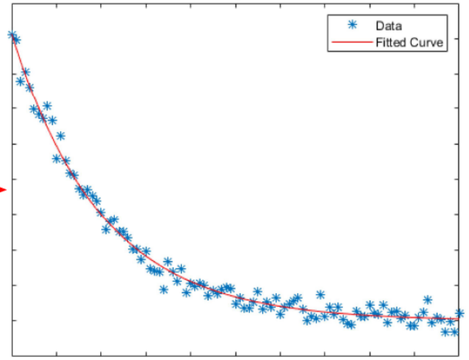
Neural Network *Predicting with Regression*

input examples

output examples

"here are some values you've maybe never seen before"

[0.6, 0.2, 0.4]



"here's an output that would 'fit' in your training data"

[0.9, 0.8]

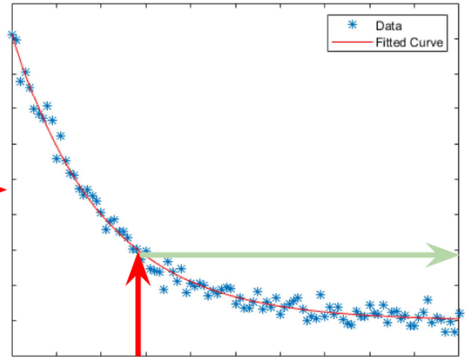
Neural Network *Predicting with Regression*

input examples

output examples

"here are some values you've maybe never seen before"

[0.6, 0.2, 0.4]

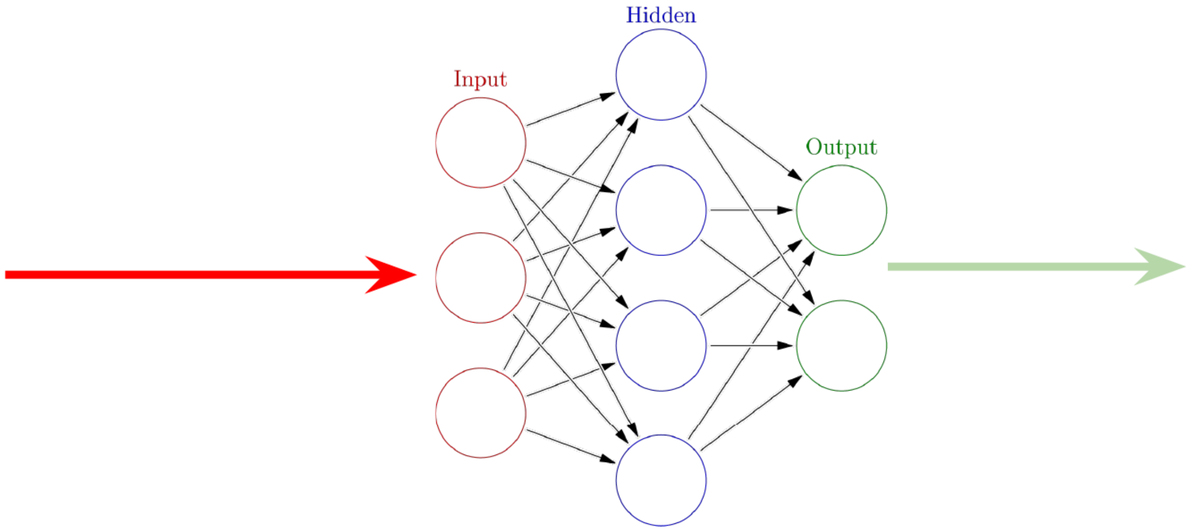


"here's an output that would 'fit' in your training data"

[0.9, 0.8]

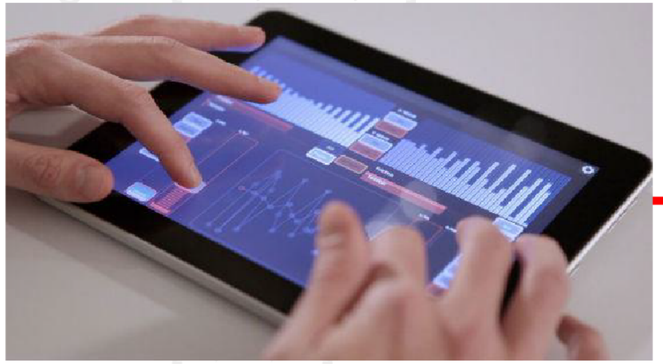
Neural Network *Predicting with Regression*

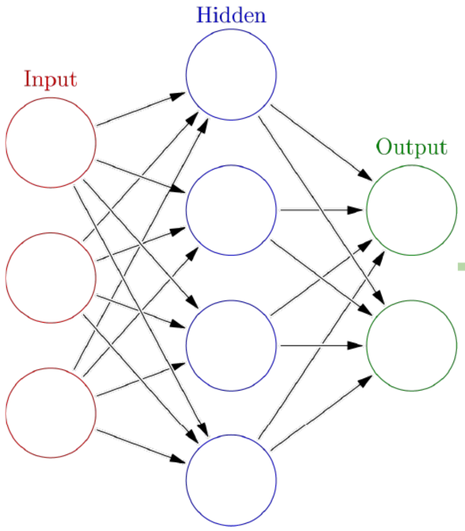
"a": [0.1, 0.5, 1]
"b": [0, 1, 1]
"c": [1, 0, 1]
"d": [0.2, 0, 1]
"e": [0, 0, 0.5]
"f": [0.8, 0.2, 1]
"g": [0.9, 0.3, 0]
"h": [0.4, 1, 0.7]
"i": [0.2, 0.3, 1]
"j": [0.1, 1, 0.1]
"k": [0, 1, 1]
"l": [0, 1, 1]
"m": [0, 0, 1]
"n": [0.4, 1, 0.7]
"o": [0.2, 0.3, 1]
"p": [0.1, 1, 0.1]
"q": [0, 1, 1]
"r": [0, 1, 1]

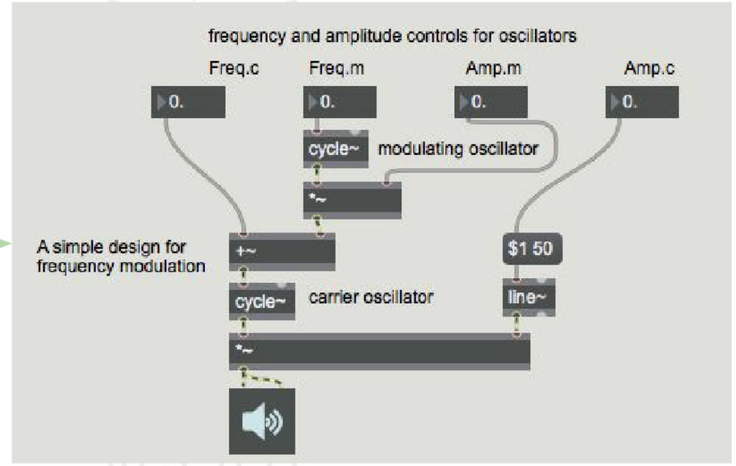


[0.5, 0.7]
[0.1, 0.1]
[0.5, 0.6]
[0.3, 0.9]
[0.4, 0]
[0.2, 0.7]
[0.5, 0.6]
[0.6, 0.1]
[0.5, 0.3]
[0.5, 0.7]
[0.1, 0.8]
[0.5, 0.1]
[0.4, 0.2]
[0.1, 0.8]
[0.5, 0.6]
[0.3, 0.9]
[0.4, 0.3]
[0.2, 0.7]

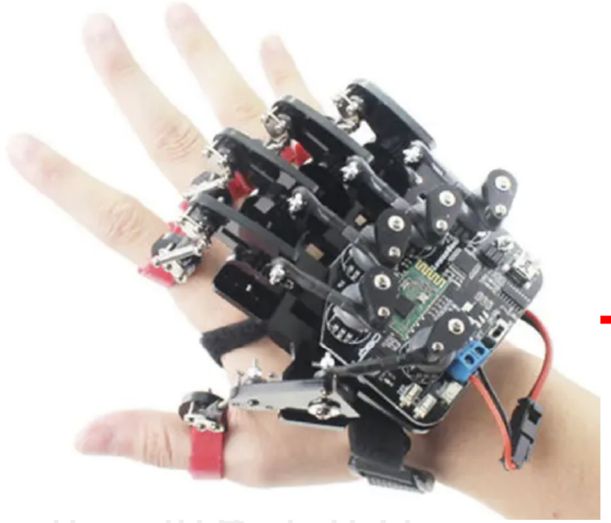
Neural Network *Predicting with Regression*

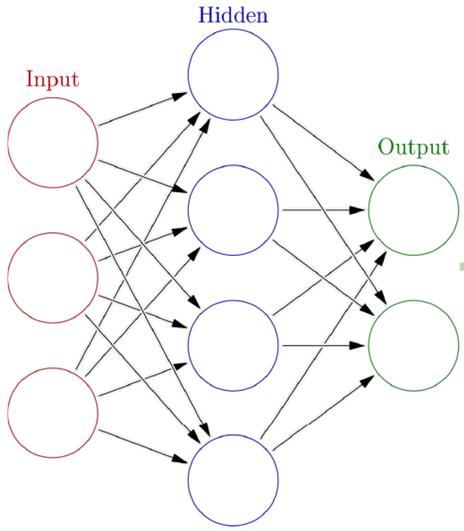


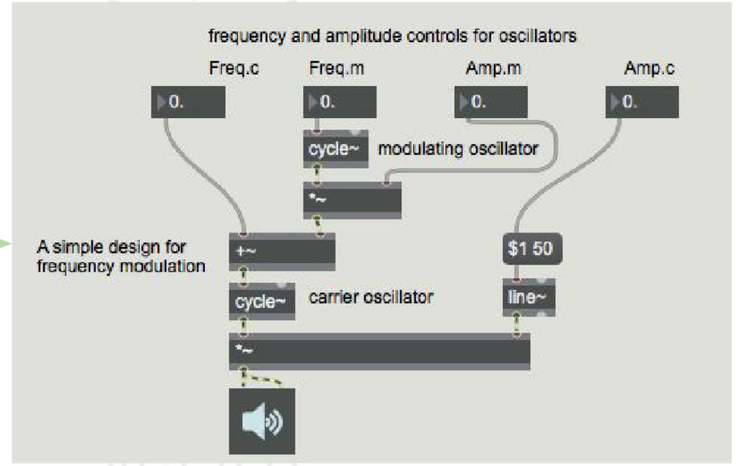




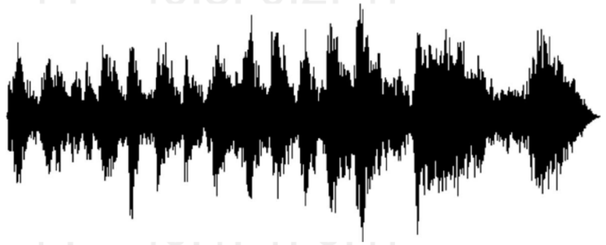
Neural Network *Predicting with Regression*



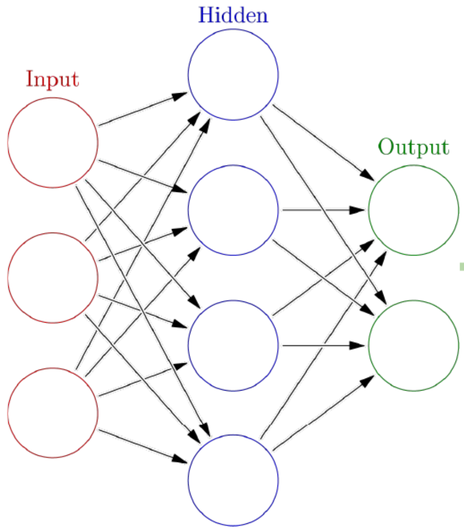


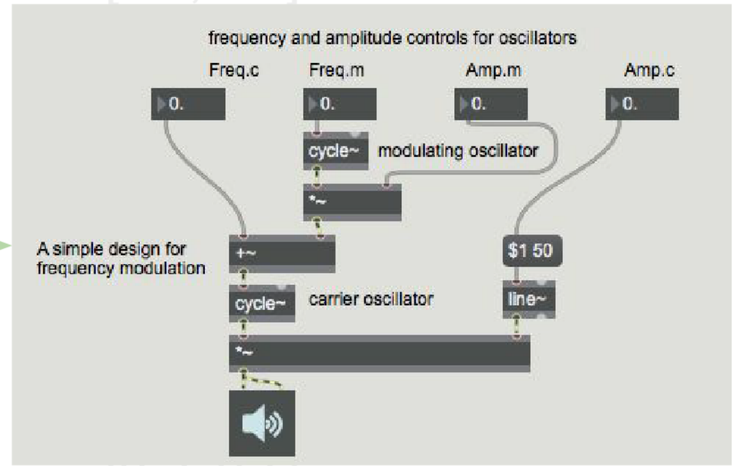


Neural Network *Predicting with Regression*

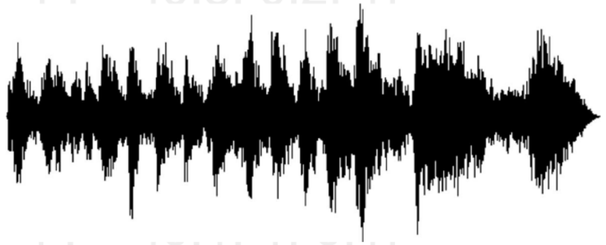


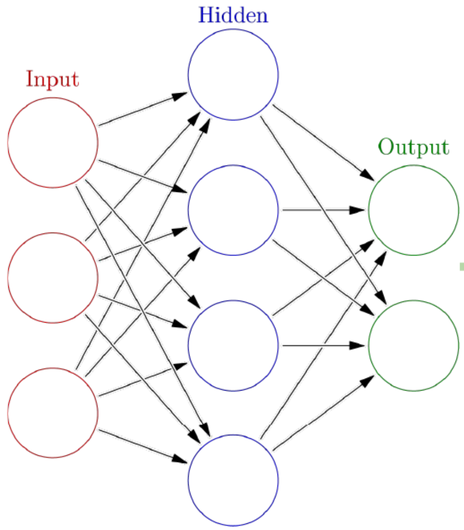
e.g.:
pitch
loudness
spectral centroid
spectral flatness





Neural Network *Predicting with Regression*

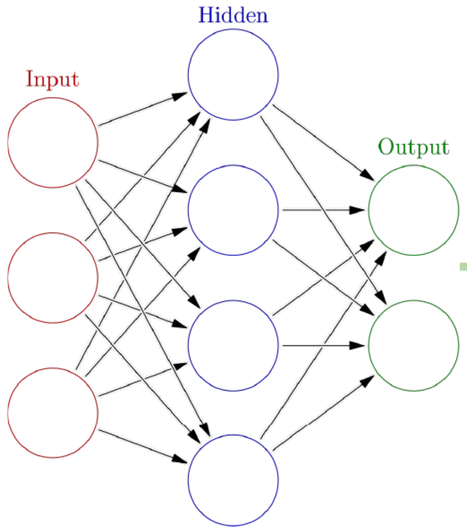




Neural Network *Predicting with Regression*

“a”: [0.1, 0.5, 1]
“b”: [0, 1, 1]
“c”: [1, 0, 1]
“d”: [0.2, 0, 1]
“e”: [0, 0, 0.5]
“f”: [0.8, 0.2, 1]
“g”: [0.9, 0.3, 0]
“h”: [0.4, 1, 0.7]
“i”: [0.2, 0.3, 1]
“j”: [0.1, 1, 0.1]
“k”: [0, 1, 1]
“l”: [0, 1, 1]
“m”: [0, 0, 1]
“n”: [0.4, 1, 0.7]
“o”: [0.2, 0.3, 1]
“p”: [0.1, 1, 0.1]
“q”: [0, 1, 1]
“r”: [0, 1, 1]

?



?

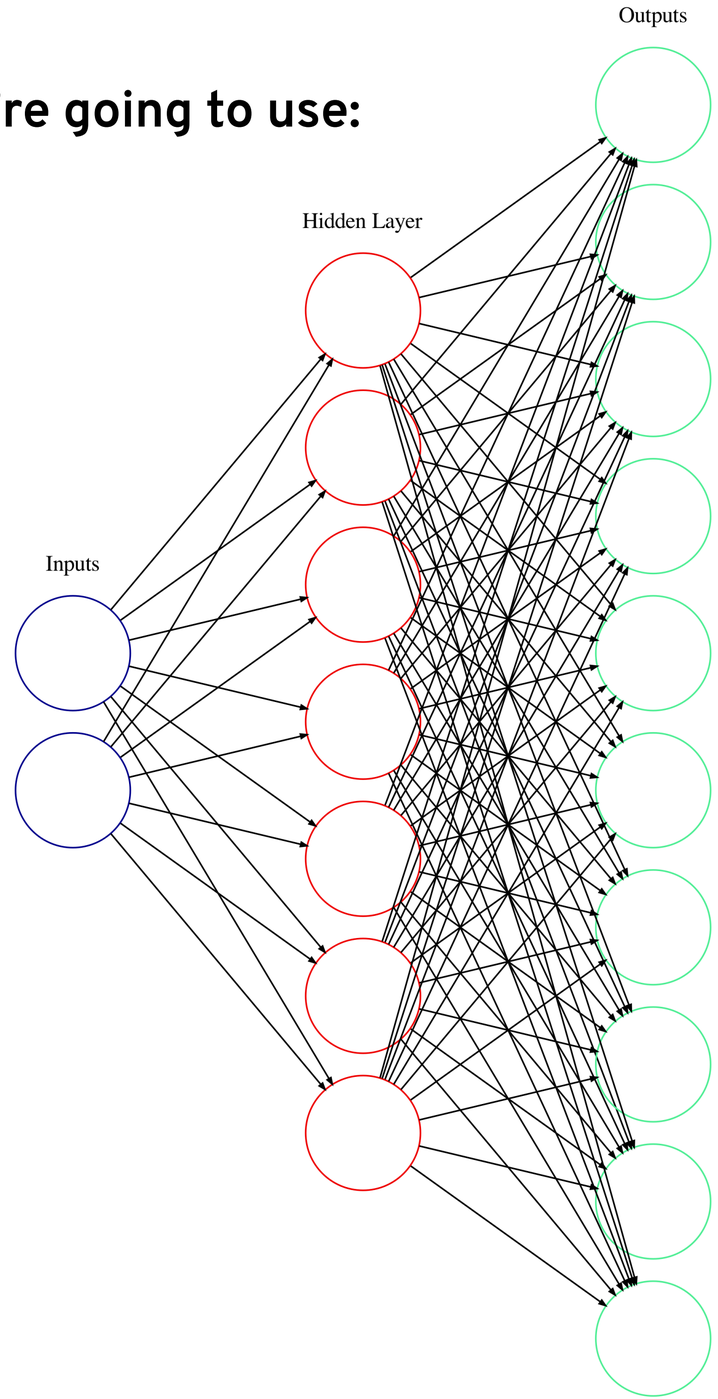
[0.5, 0.7]
[0.1, 0.1]
[0.5, 0.6]
[0.3, 0.9]
[0.4, 0]
[0.2, 0.7]
[0.5, 0.6]
[0.5, 0.7]
[0.1, 0.1]
[0.5, 0.6]
[0.2, 0.7]
[0.6, 0.1]
[0.5, 0.3]
[0.5, 0.7]
[0.1, 0.8]
[0.5, 0.1]
[0.4, 0.2]
[0.1, 0.8]

Structure of the neural network we're going to use:

2 inputs

1 hidden layer of 7 nodes

10 outputs

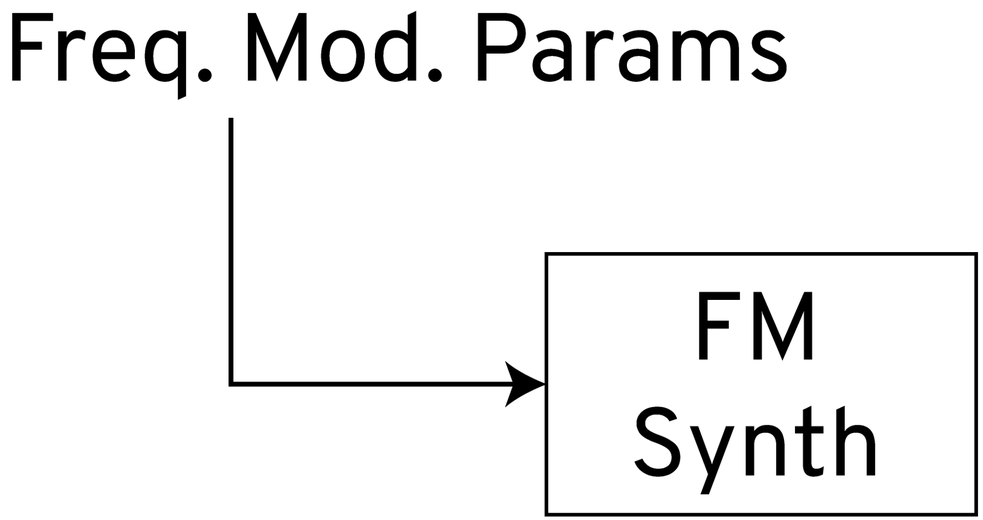


Regression with Audio Analyses

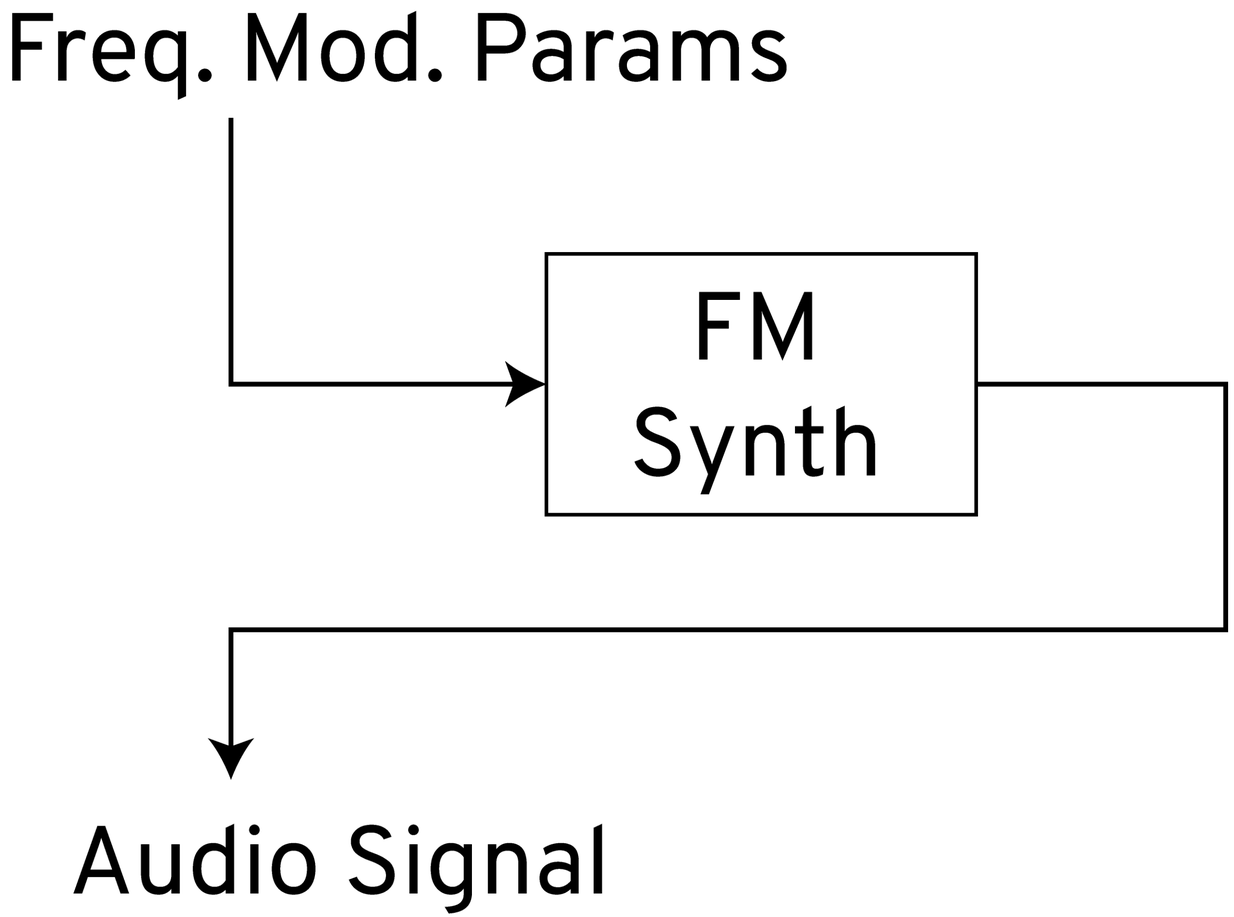
Making the Datasets

Freq. Mod. Params

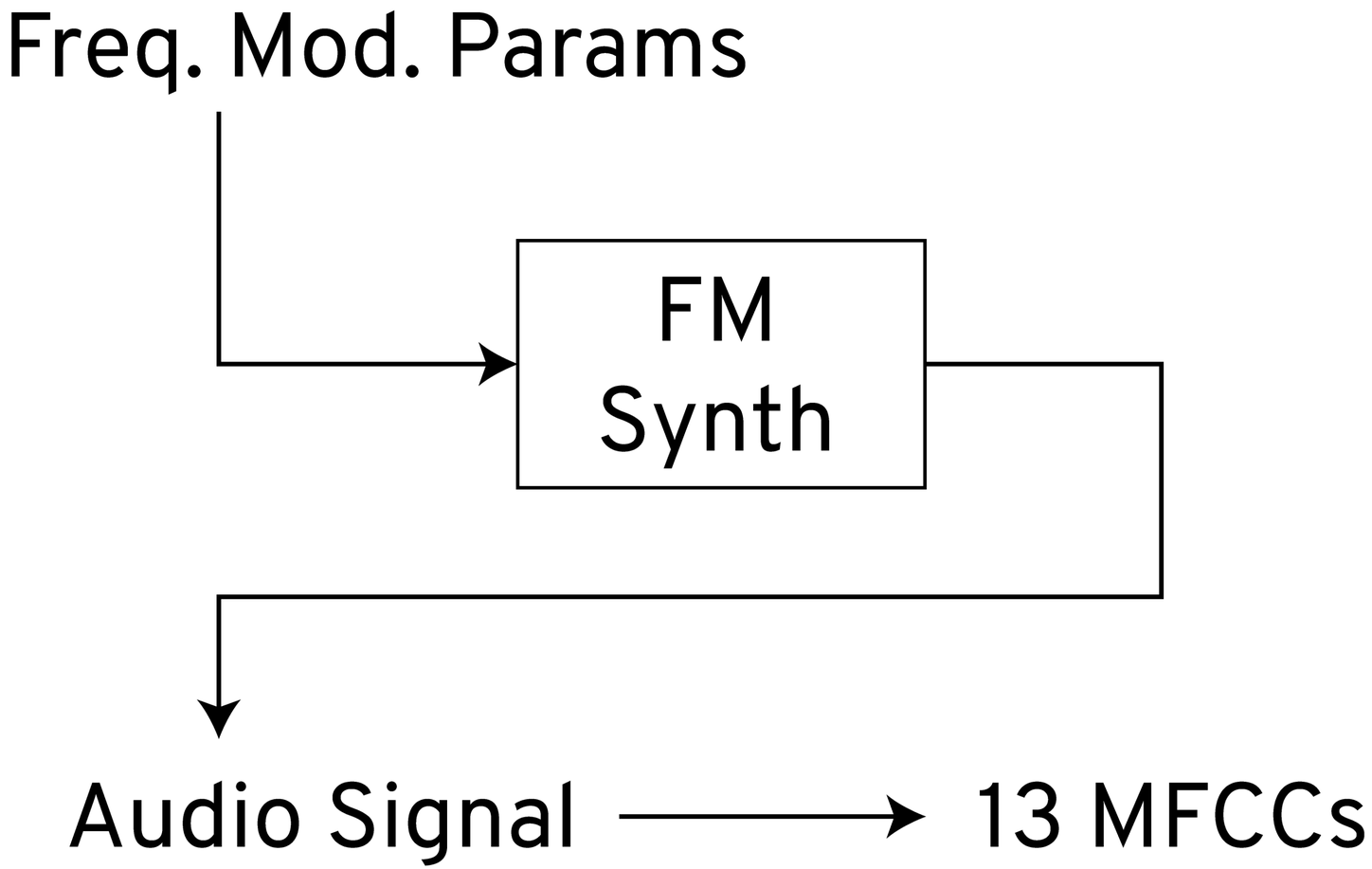
Making the Datasets



Making the Datasets



Making the Datasets



Making the Datasets

Freq. Mod. Params

13 MFCCs

Making the Datasets

Freq. Mod. Params

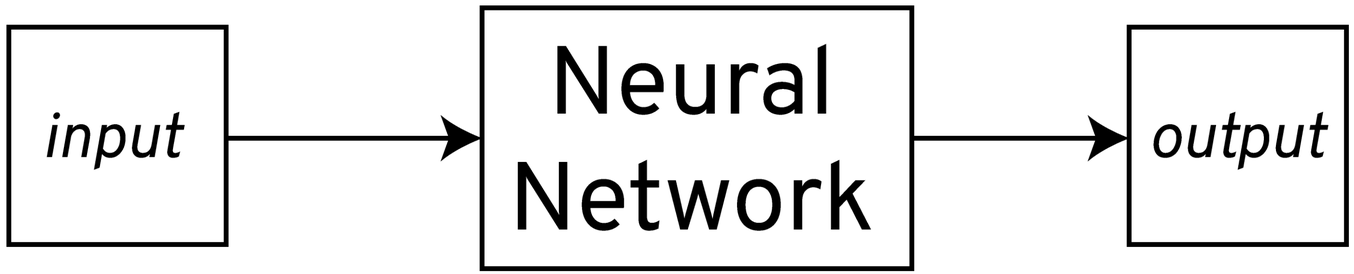
```
DataSet 64:
rows: 100 cols: 3
point-0     2411.1    206.48    3.9116
point-1     69.214    65.334     3.862
point-2      214.7    67.041    6.1554
        ...
point-97     2114.2    82.574    7.8388
point-98     307.08    307.08    15.905
point-99     3797.7    60.559    15.826
```

```
DataSet 63:
rows: 100 cols: 13
point-0     183.98   -204.99    -97.85       ...    17.116    6.7544   -2.6597
point-1     238.64    92.367    67.025       ...   -2.1895   0.26965    3.3148
point-2     279.43     96.95     44.27       ...    9.4531    2.0989   -1.2892
        ...
point-97     162.65   -185.08   -142.91       ...   -19.588    9.1036    14.504
point-98     171.71   -210.44    64.047       ...    0.6732    -15.17   -8.3311
point-99      4.409   -235.47    69.209       ...   -12.299   -6.0117    28.718
```

13 MFCCs

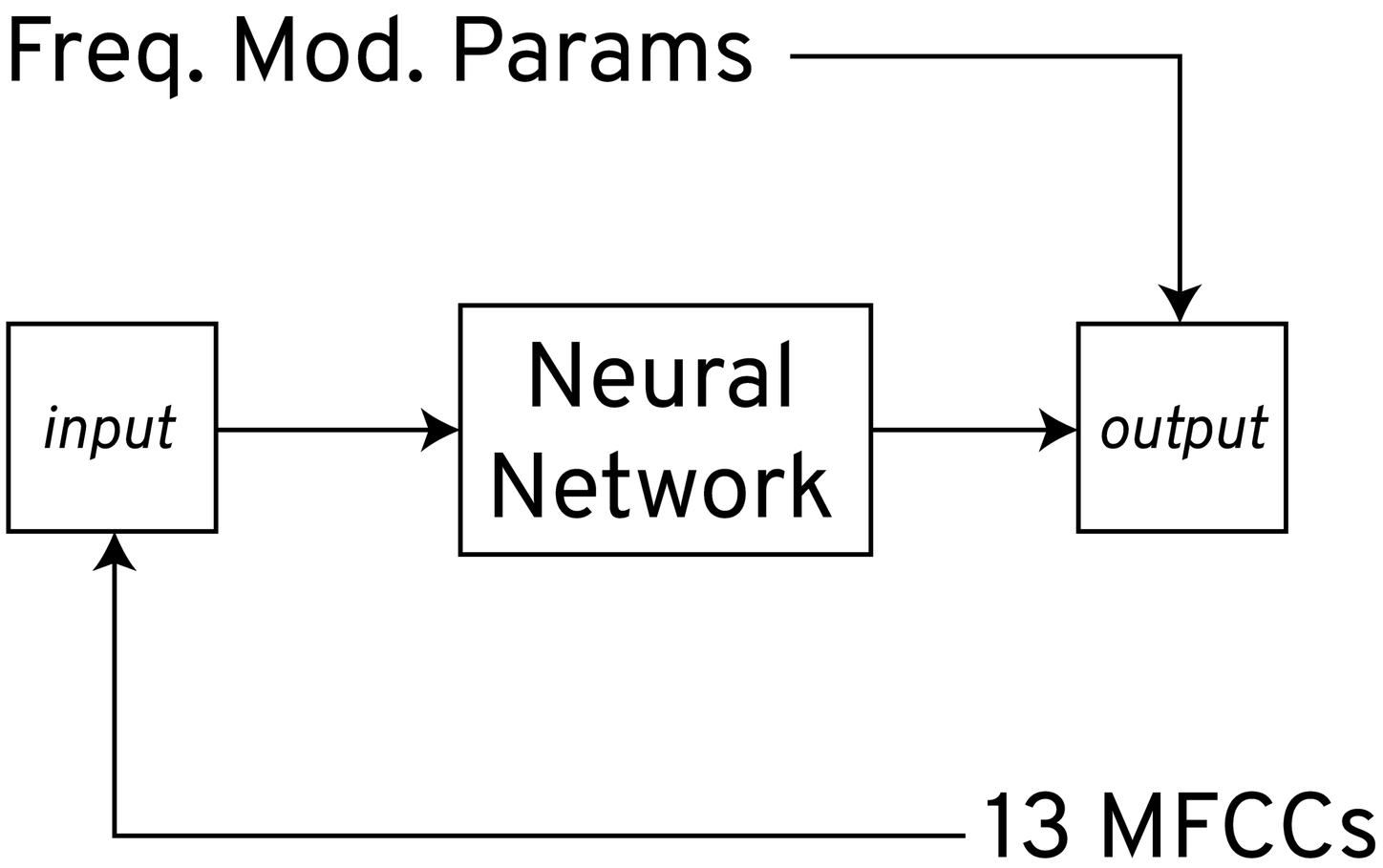
Training

Freq. Mod. Params

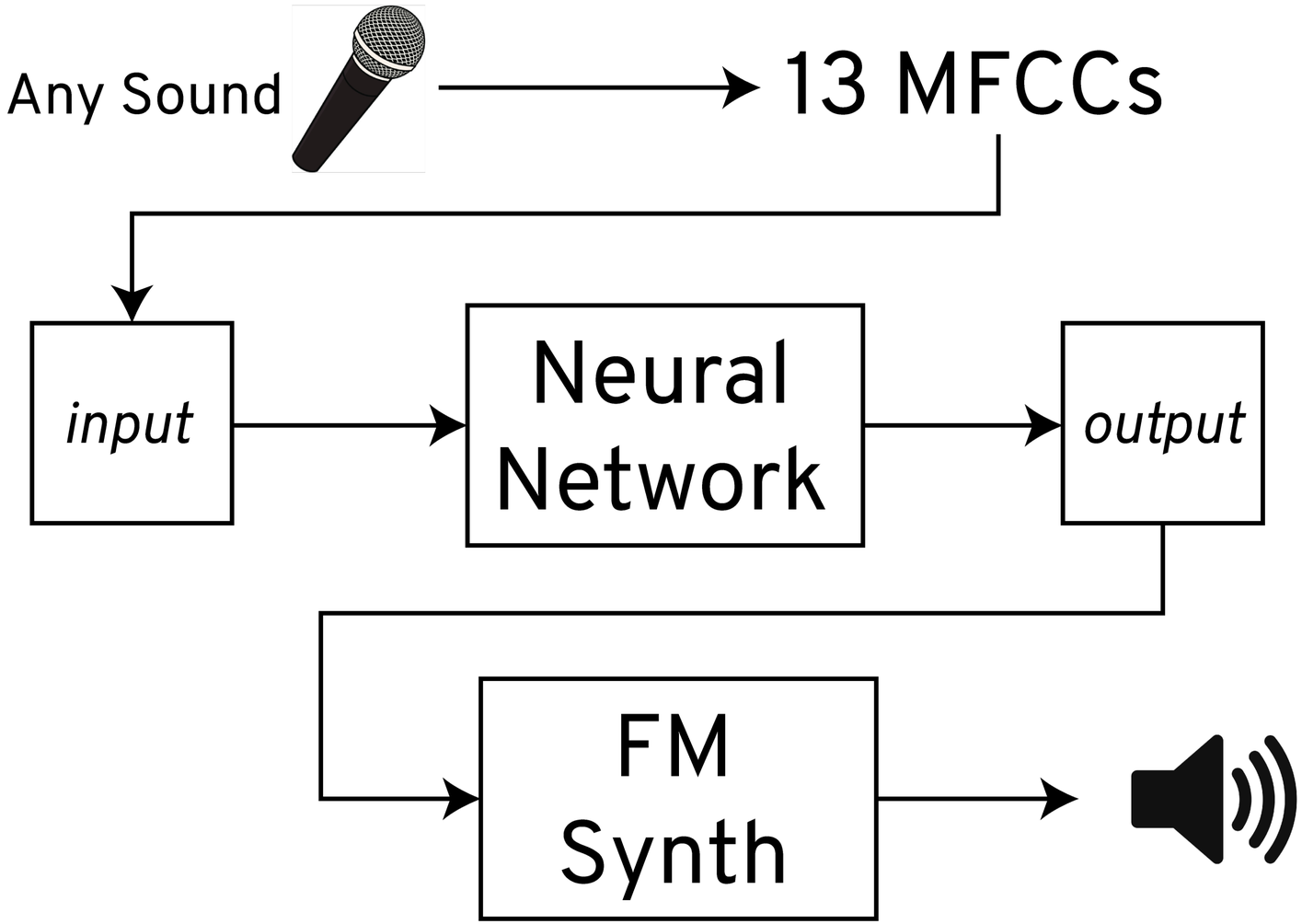


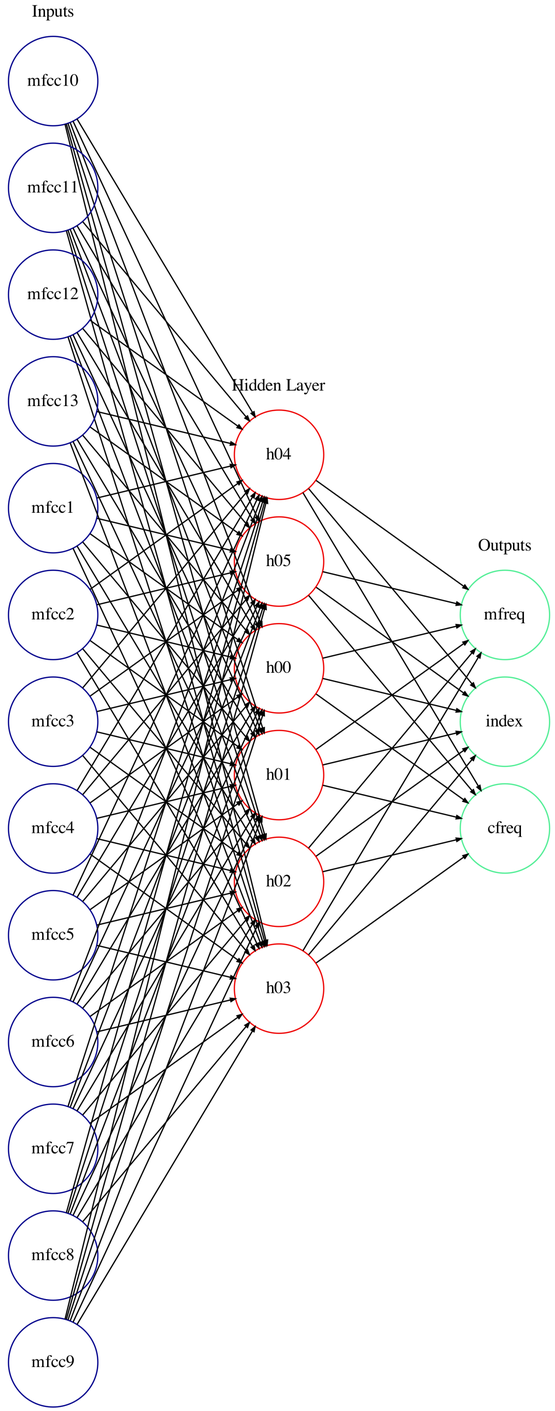
13 MFCCs

Training



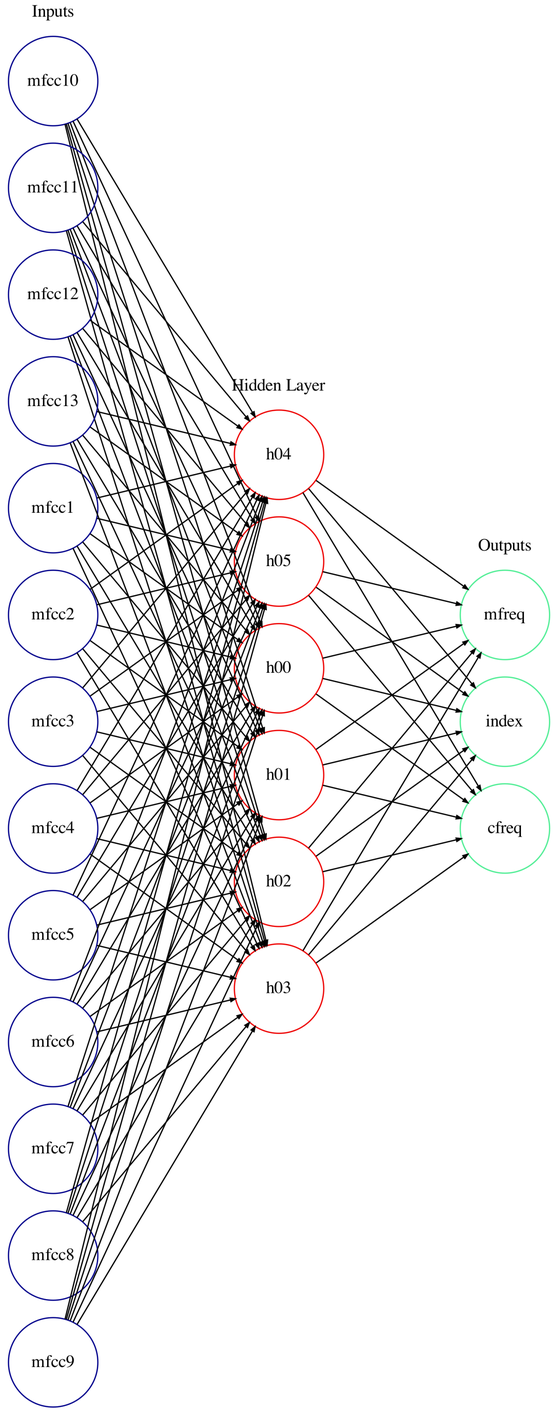
Performing



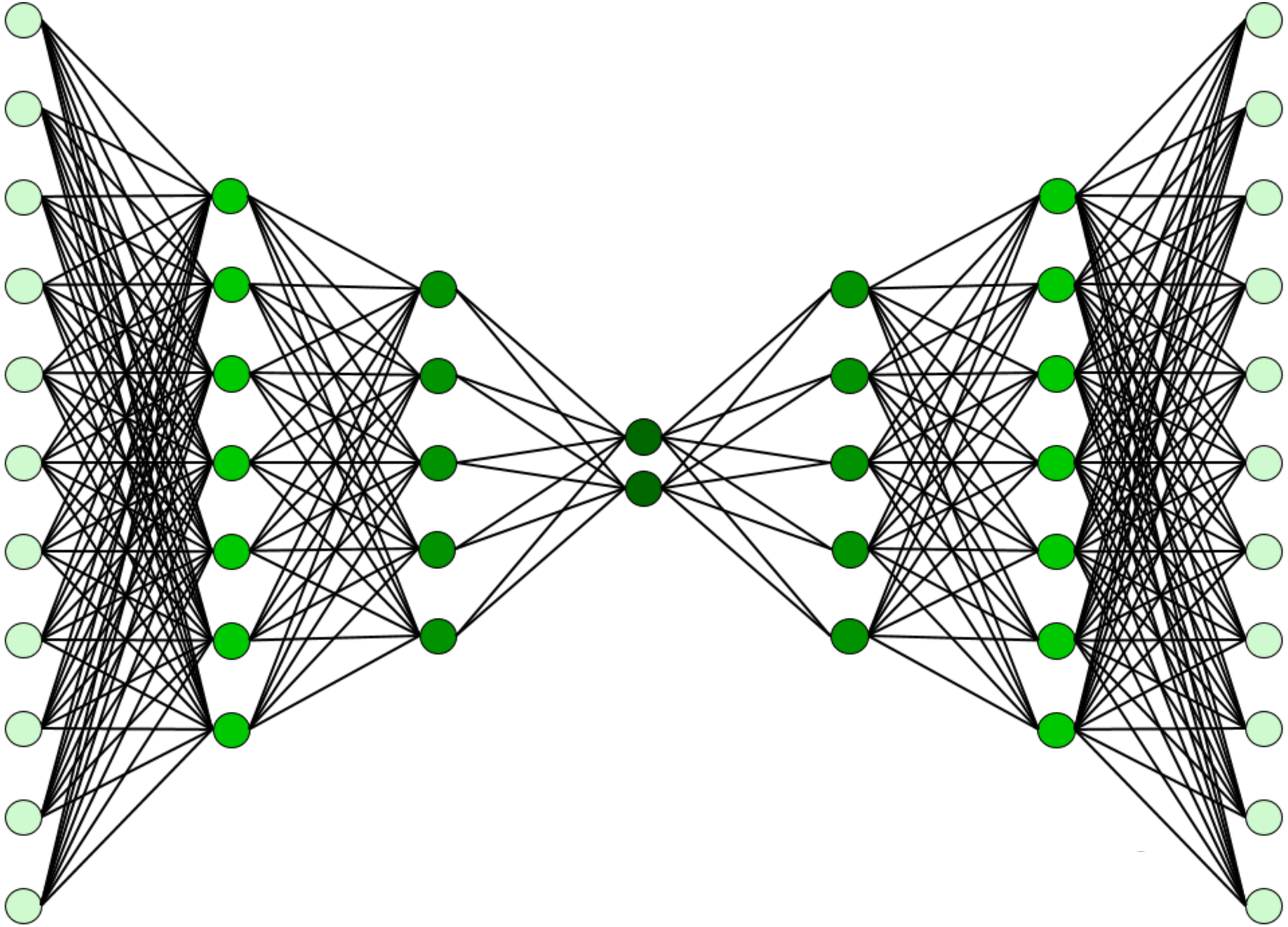

Inputs
mfcc10
mfcc11
mfcc12
mfcc13
mfcc1
mfcc2
mfcc3
mfcc4
mfcc5
mfcc6
mfcc7
mfcc8
mfcc9
Hidden Layer
h04
h05
h00
h01
h02
h03
Outputs
mfreq
index
cfreq

Wavetable Autoencoder

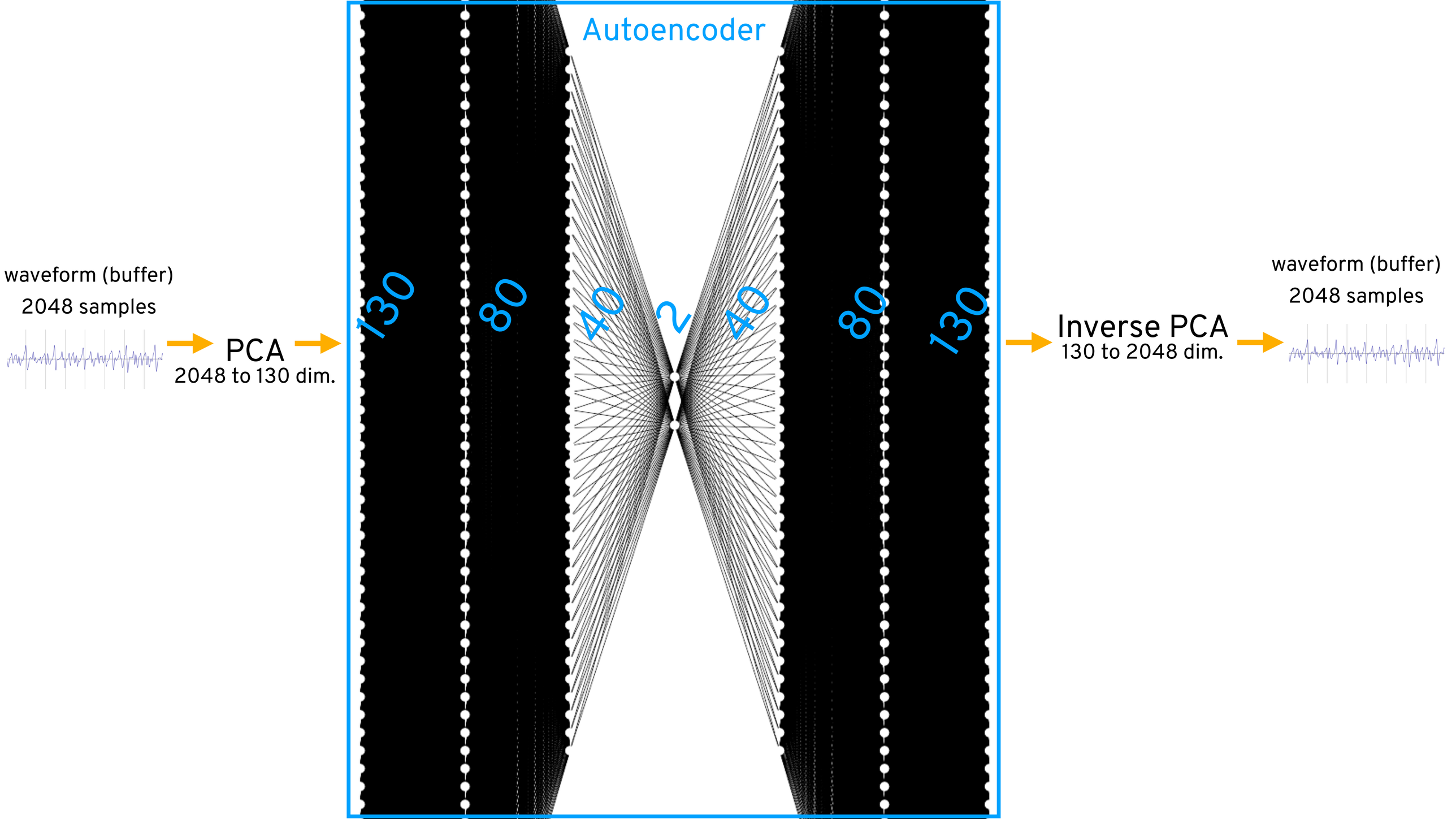
MFCCs —> FM Params



Autoencoder



https://encodebox.medium.com/auto-encoder-in-biology-9264da118b83


waveform (buffer)
2048 samples
PCA
2048 to 130 dim.
Autoencoder
130
80
40
2
40
80
130
Inverse PCA
130 to 2048 dim.
waveform (buffer)
2048 samples

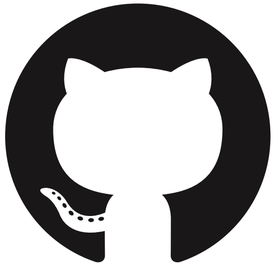
flucoma.org
learn.flucoma.org

download FluCoMa package
toolkit reference
learn from tutorials
hear & see works made with FluCoMa

discourse.flucoma.org

share works in progress
ask questions
get ideas

https://github.com/flucoma



bug reports
pull requests

https://www.youtube.com/c/
fluidcorpusmanipulation

tutorials
performances


Tell me the truth...I'm...I'm ready to hear it.
YOU DON'T NEED MACHINE LEARNING FOR THAT
40
1,406
10.3K
Tweet your reply