DEEP LEARNING FOR PHM

PHM Conference 2017

Emilien Dupont

WHAT IS DEEP LEARNING?

ONE SENTENCE DEFINITION

Set of methods to learn very flexible differentiable functions that seem to work well for most real life data

Best way to think of it as a computational graph

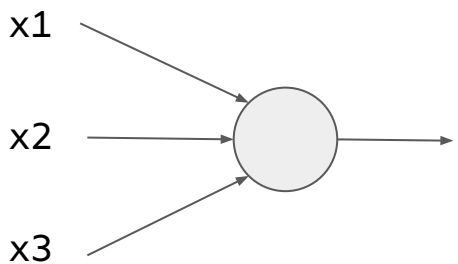
THE WORLD'S SIMPLEST COMPUTATIONAL GRAPH

Input

Music: Yes/No

Friends: Yes/No

Weather: Good/bad



Output

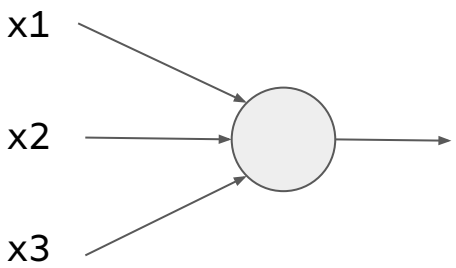
Go out: Yes/No

HOW IT WORKS

Music: No

Friends: Yes

Weather: Good



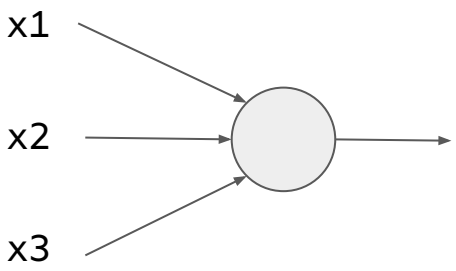
Go out: Yes

HOW IT WORKS

Music: 0

Friends: 1

Weather: 1



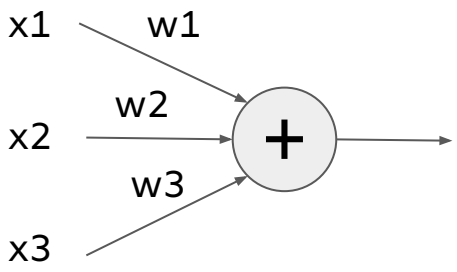
Go out: 1

THE ADDITION NODE

Music: 0

Friends: 1

Weather: 1



= x1 * w1 + x2 * w2 + x3 * w3

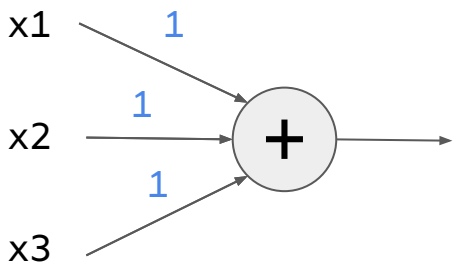
= 0 * w1 + 1 * w2 + 1 * w3

THE ADDITION NODE

Music: 0

Friends: 1

Weather: 1

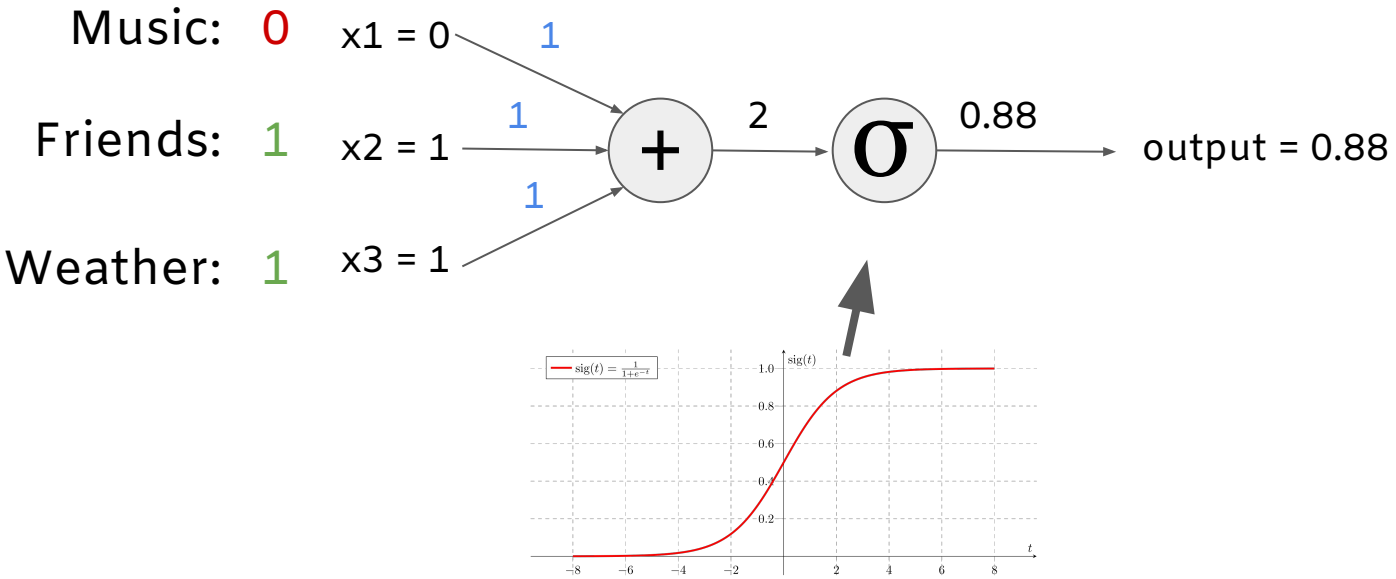


= x1 * w1 + x2 * w2 + x3 * w3

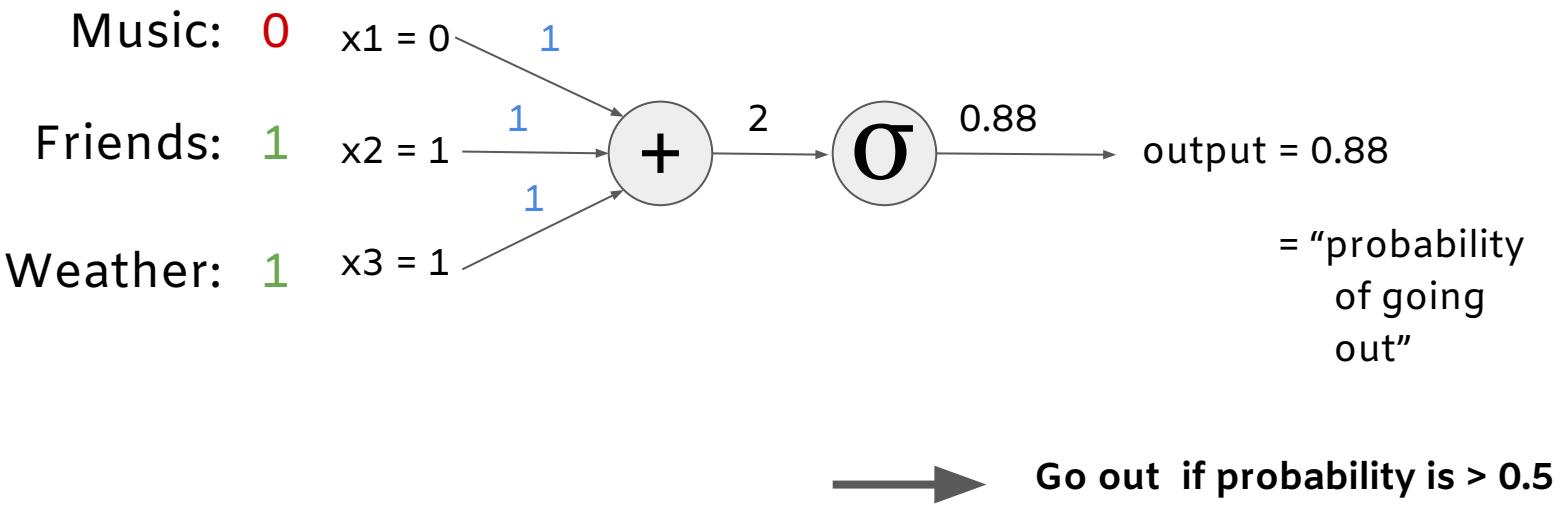
= 0 * 1 + 1 * 1 + 1 * 1

= 2

TURN IT INTO A DECISION



TURN IT INTO A DECISION



Go out if probability is > 0.5

HOW TO TRAIN

We now have a neural net that can make simple decisions

How can we train the neural net, i.e. **adjust w1, w2, w3** so that the decisions it makes are “correct”?

HOW TO TRAIN

Show it examples:

Music, No Friends, Bad Weather -> Don’t go

No Music, Friends, Good Weather -> Go

etc...

Tell it **when it’s doing bad** and **when it’s doing well**

DEFINE A LOSS

Define a loss or cost, i.e. a way to tell the neural network how bad it is doing

Loss = (output - decision)^2

For example

Loss = (0.88 - 1)^2 = 0.014

MINIMIZE LOSS

Choose w1, w2, w3 so the loss is minimized on the examples

How to adjust w1, w2, w3?

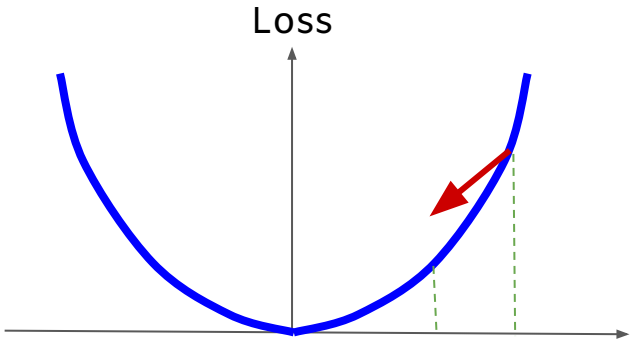
WEIGHT UPDATES

Calculate **derivative of loss with respect to weights**

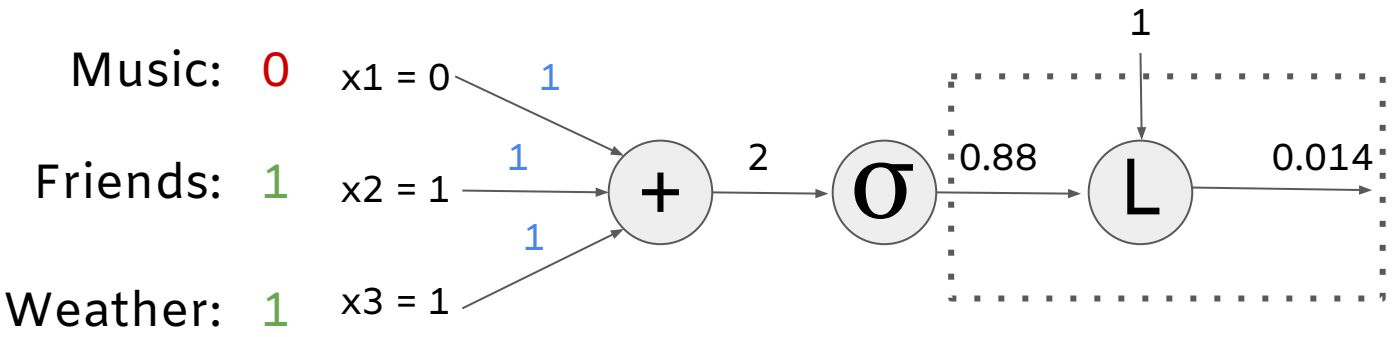
$$\frac{dL}{dw1}$$

Update weights to minimize loss

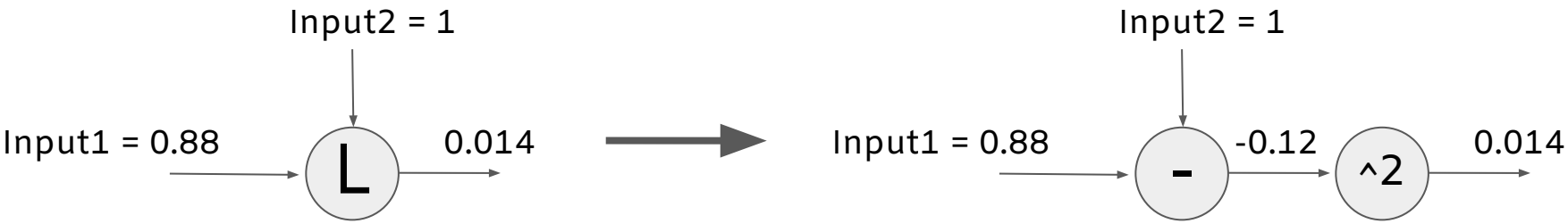
$$w1 \leftarrow w1 - a \frac{dL}{dw1}$$



CALCULATING DERIVATIVE



CALCULATING DERIVATIVE



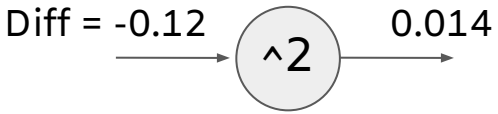
Loss = (Input1 - Input2)^2

We can also write this as

Diff = Input1 - Input2

Loss = Diff^2

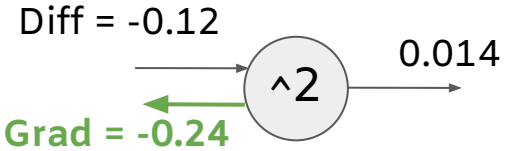
CALCULATING DERIVATIVE



Loss = Diff^2

It's easy to calculate the gradient of the loss with respect to its input, it is simply:

Derivative=2*Diff



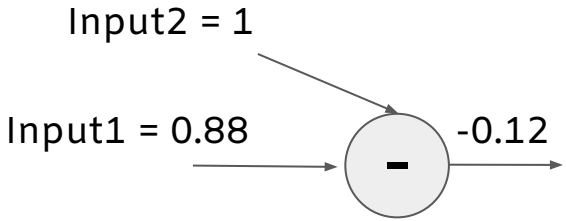
A NODE IN THE GRAPH

The squaring node:

Given an input, we can compute the output and the gradient of the loss with respect to the input

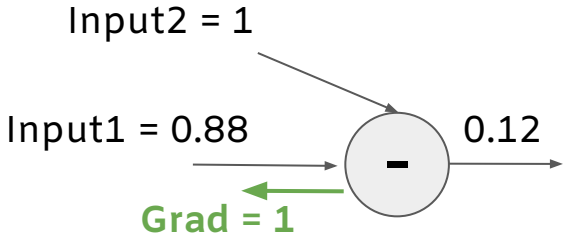
Super useful!

THE SUBTRACTION NODE



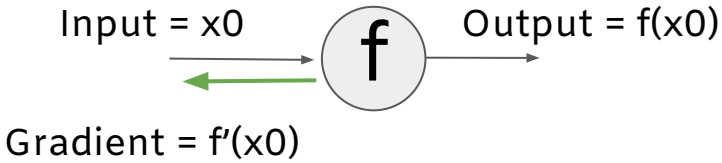
Output = Input1 - Input2

Derivative with Input1 = 1



GENERAL NODE

Clearly it is easy to create nodes like this for simple differentiable functions



Give node **input**: get **output** and **gradient with respect to its input** back

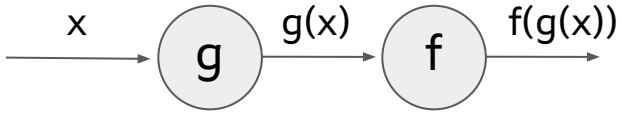
MANY NODES

How to combine nodes?

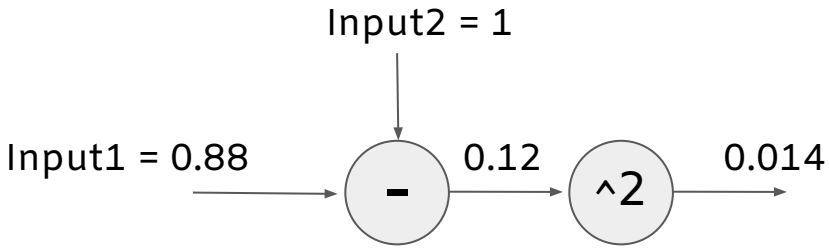
Chain rule!

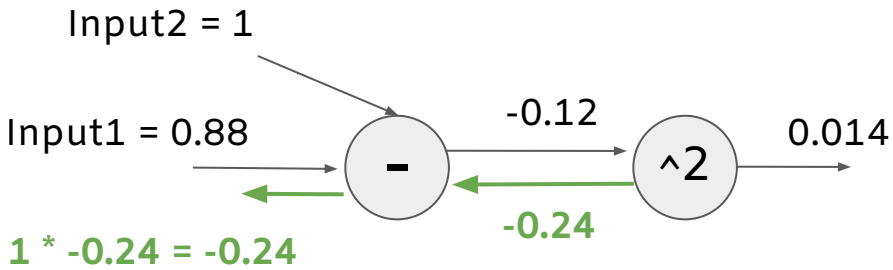
$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

MULTIPLY GRADIENTS

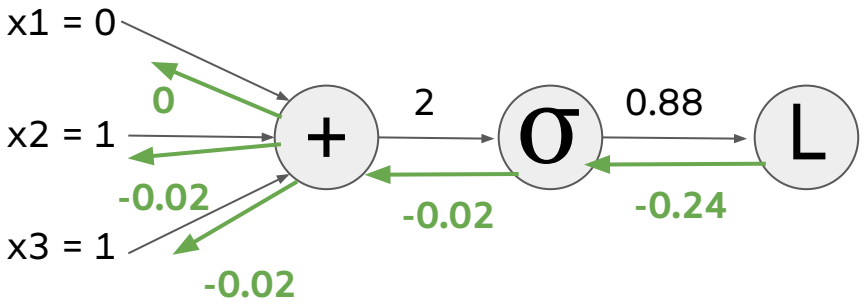


EXAMPLE

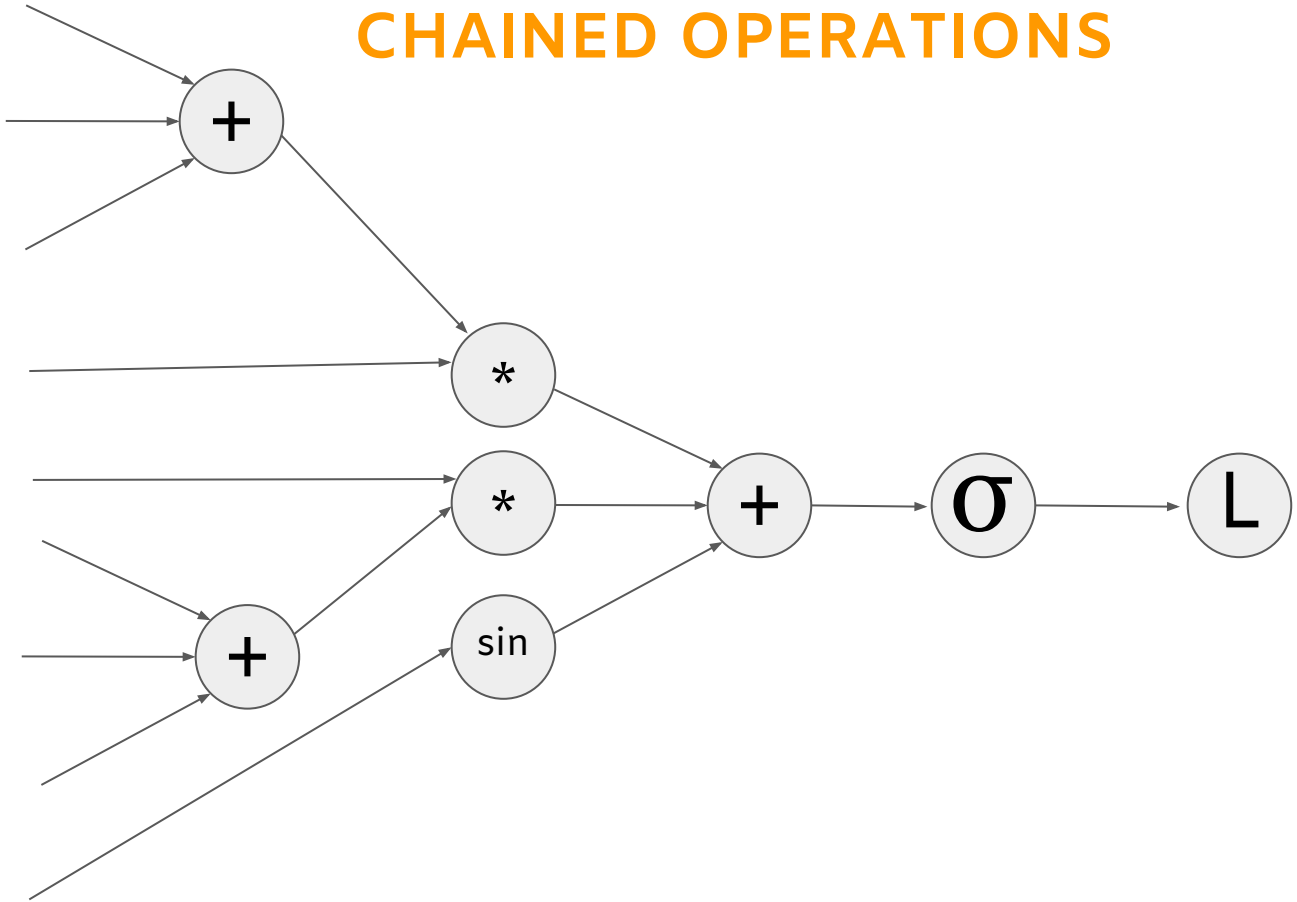

Input2 = 1
Input1 = 0.88
-
0.12
^2
0.014


Input2 = 1
Input1 = 0.88
-
-0.12
^2
0.014
-0.24
1 * -0.24 = -0.24

CHAINED OPERATIONS



CHAINED OPERATIONS



NOTE

We never had to write the gradient down

Method to calculate gradient at any point *without knowing formula for gradient*

Crucial when scaling to large networks

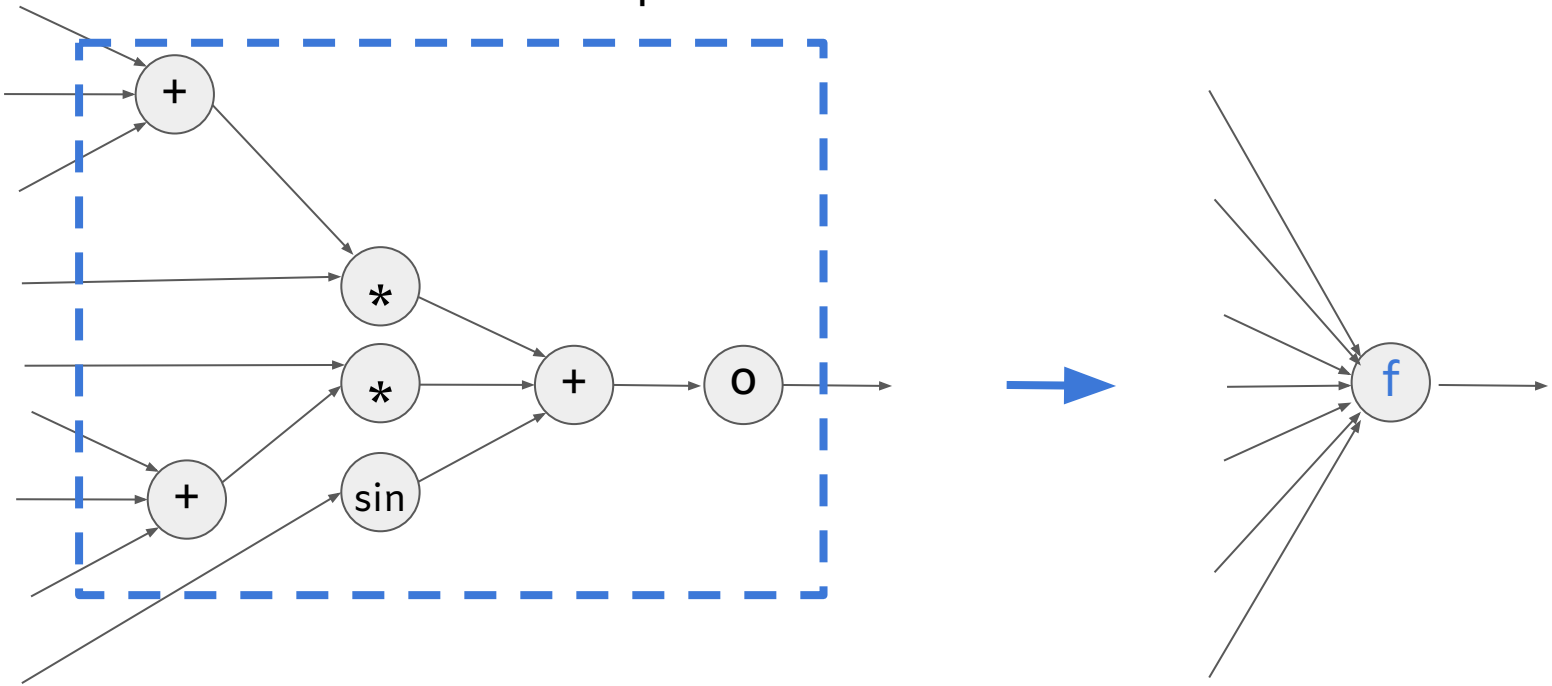
Modern Deep Learning architectures would requires 10’s or 100’s of pages just to write down the gradient

DEEP LEARNING METHOD

1. Create computational graph
2. Show examples
3. Calculate gradients
4. Update weights using gradients
5. Iterate until happy

MODULARITY AND ABSTRACTION

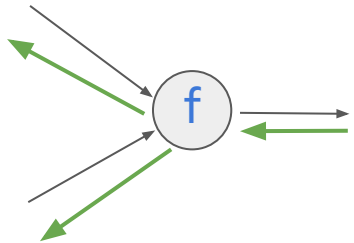
You can always summarize one big block of operations as one node



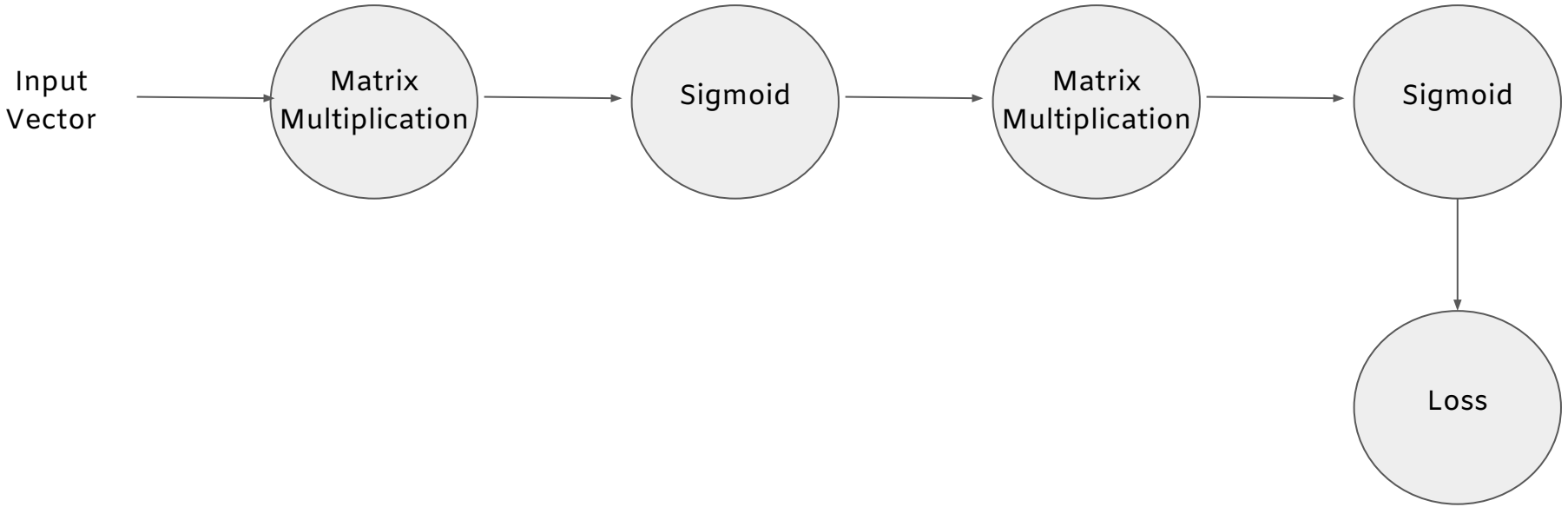
MODULARITY AND ABSTRACTION

A node can be **anything** as long as it's differentiable (you can input value and get output and gradient)

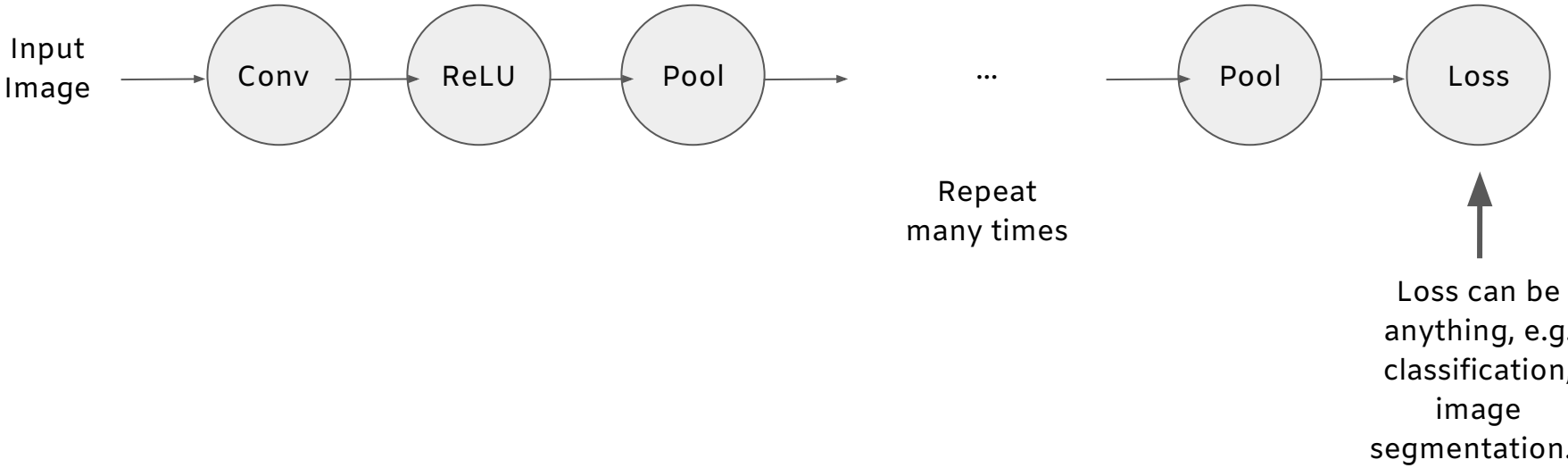
It can be addition, a wavelet transform, exp(sin(log(x^2))), Gaussian kernel smoothing, anything that is differentiable!



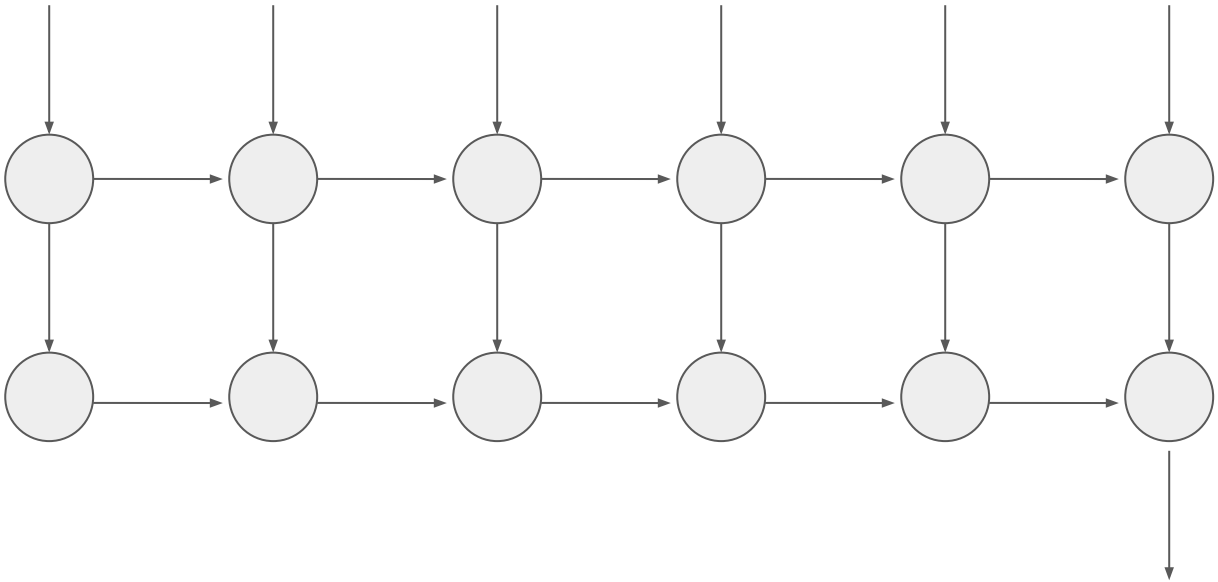
CLASSIC FEEDFORWARD NETWORK



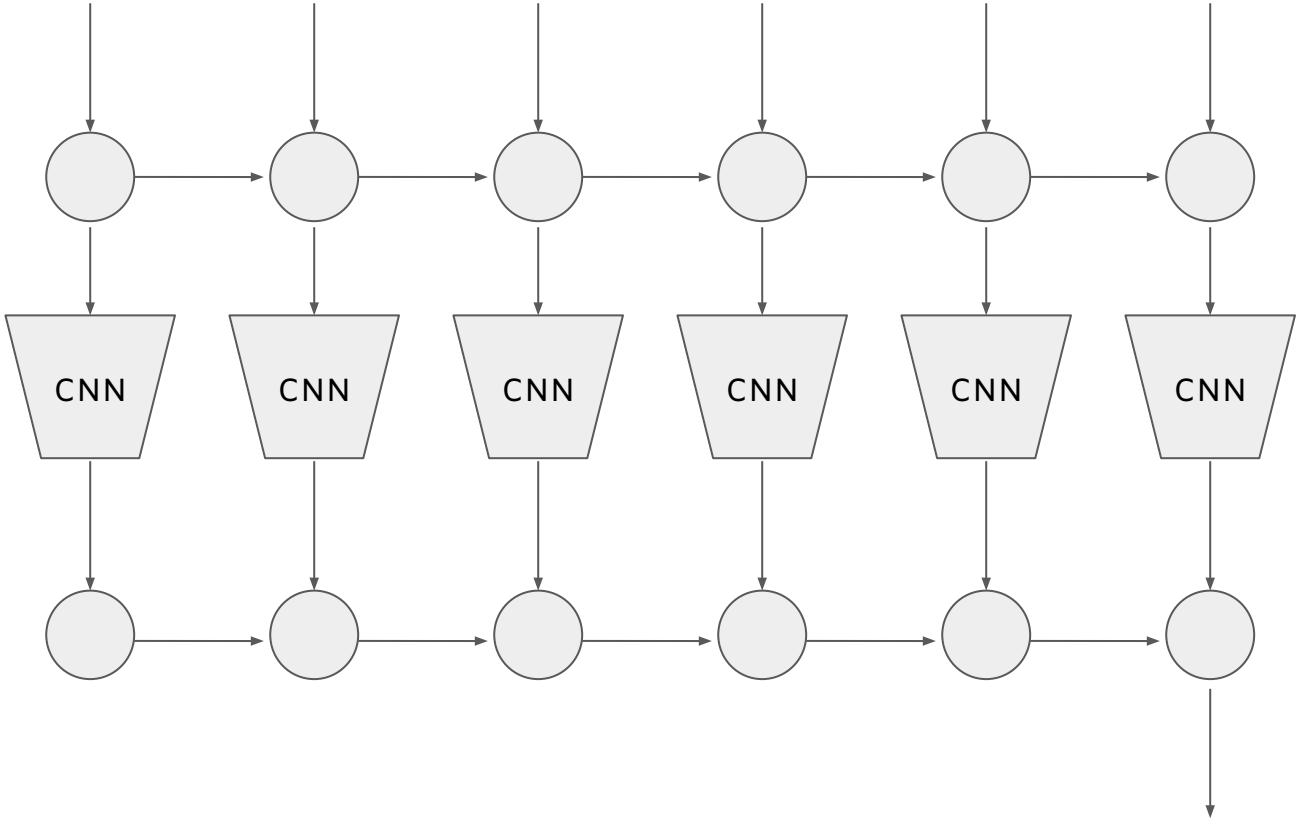
CNN


Input Image
Conv
ReLU
Pool
...
Pool
Loss
Repeat many times
Loss can be anything, e.g. classification, image segmentation...

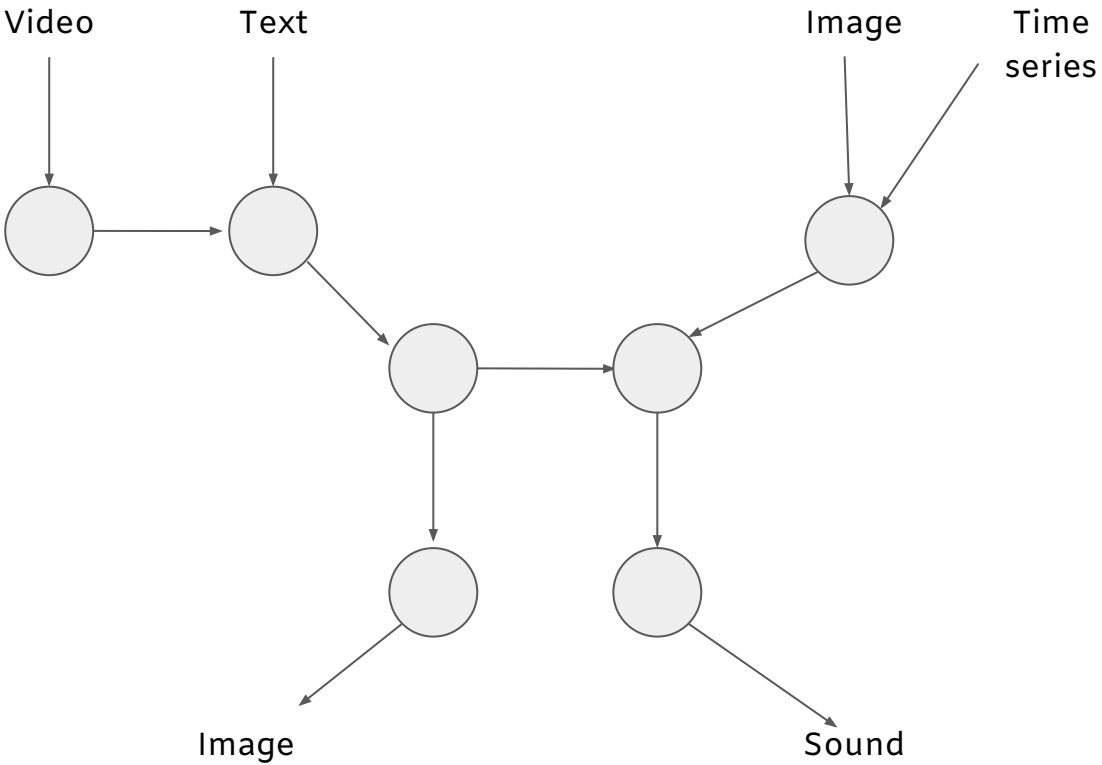
RNN



VIDEO RNN



RANDOM ARCHITECTURE



TAKEAWAYS

Very powerful framework to optimize arbitrary differentiable functions

Can define any layer you want, any function you want, any loss you want etc...

Never try to write down the actual derivative, not helpful, very tedious and error prone

WHAT IS NEW IN DEEP LEARNING?

WHAT'S NEW?

Neural networks already existed in the 80's and 90's so why are people getting so excited?

WHAT'S NEW?

More data

More compute (GPUs)

Better algorithms

There is a critical mass of data you need to make deep learning really good and that was only reached recently

WHAT'S NEW?

Computational Graph Interpretation

Frameworks to run computational graphs
(Tensorflow, PyTorch, CNTK, Theano...)

Modularity

Word2vec

Adam

BatchNorm

WHAT'S NEW?

Dropout

Residual Blocks

Optimization initializations

GANs, VAEs

And much much much more

All these things have become standard and deep learning is pretty bad otherwise

HOW TO START DOING DEEP LEARNING?

INSTALL A FRAMEWORK

Open your terminal and type:

```
pip install keras
```

SIMPLE HOW TO

Open your code editor and create a python script

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
```

DEEP LEARNING APPLIED TO PHM

WHAT TYPE OF PROBLEMS/DATA?

Time Series

Images

Sensor Data

Sounds

Anything else you can think of! (Text?)

WHEN CAN DEEP LEARNING BE APPLIED TO PHM?

This is not an easy question to answer…

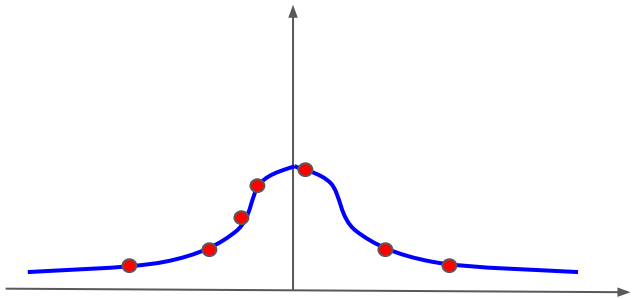
Two factors to keep in mind:

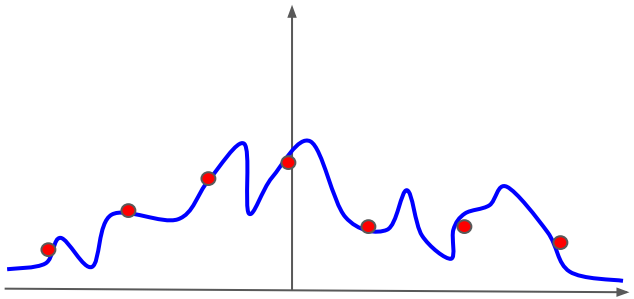
1. Need enough data
2. Data should not be too noisy

HOW MUCH DATA?

No single answer but approximately:

Enough samples to represent the distribution of the data





HOW MUCH DATA?

With **traditional machine learning** you can train with **less data** because you restrict/confine the problem more:

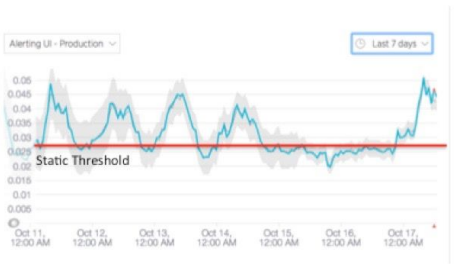
Feature selection, feature engineering, less powerful model...

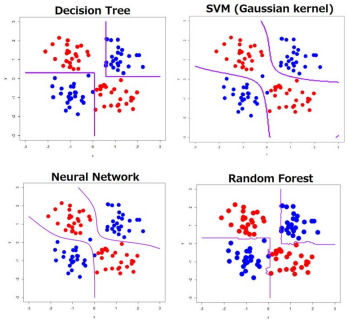
However, if enough data is available for the machine to learn its own features, it will often do better than humans

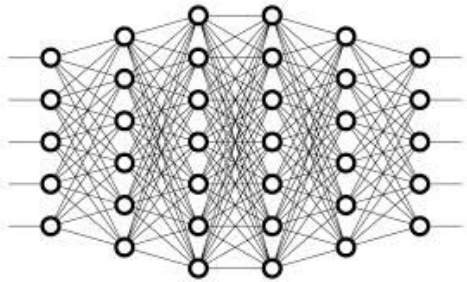
WHEN CAN DEEP LEARNING BE APPLIED TO PHM?

Always have a baseline (e.g. simple heuristic)

Always try a traditional machine learning approach first (I've made this mistake many times...)



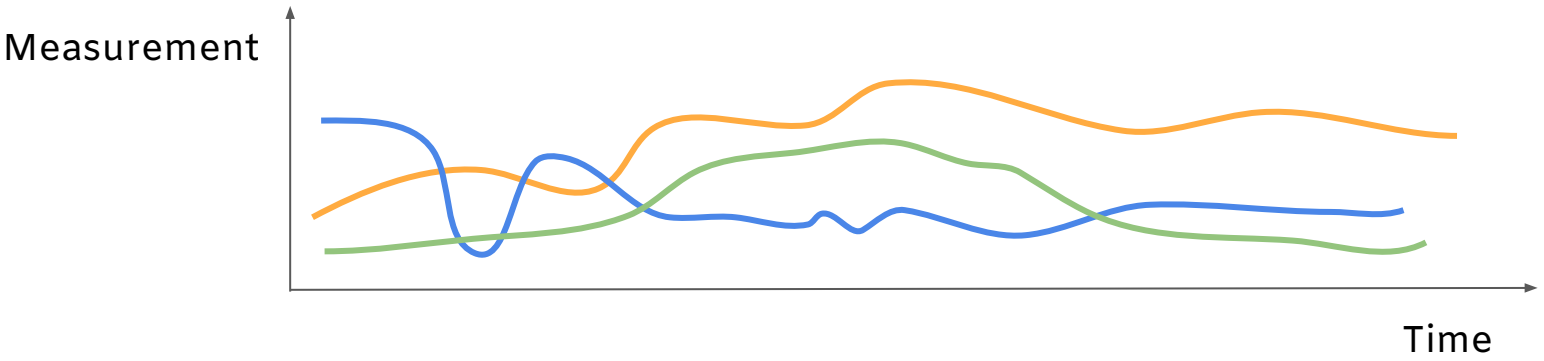




APPLICATIONS: TIME SERIES

Example: Consider an asset which has *n* sensors recording data at some frequency.

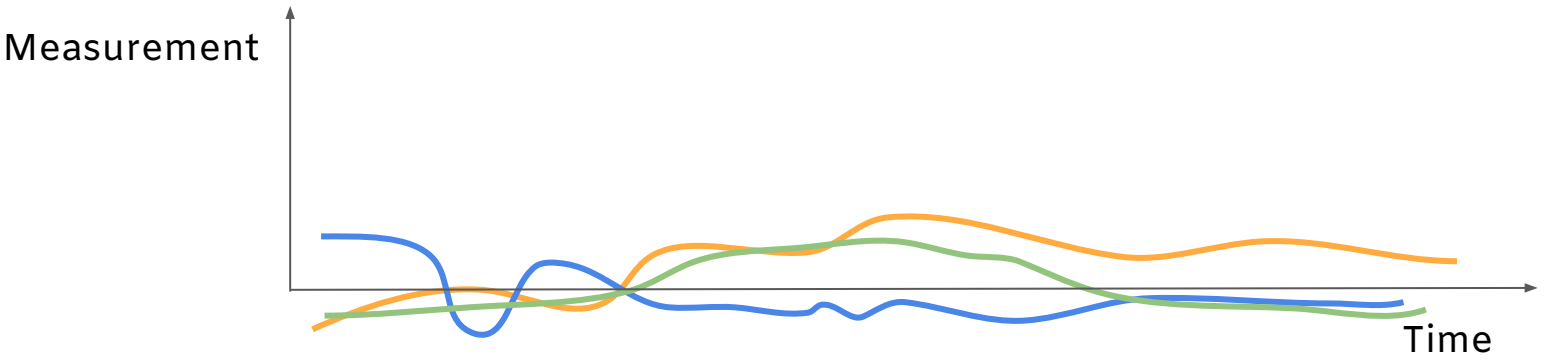
We want to use deep learning to set alarms or predict anomalies



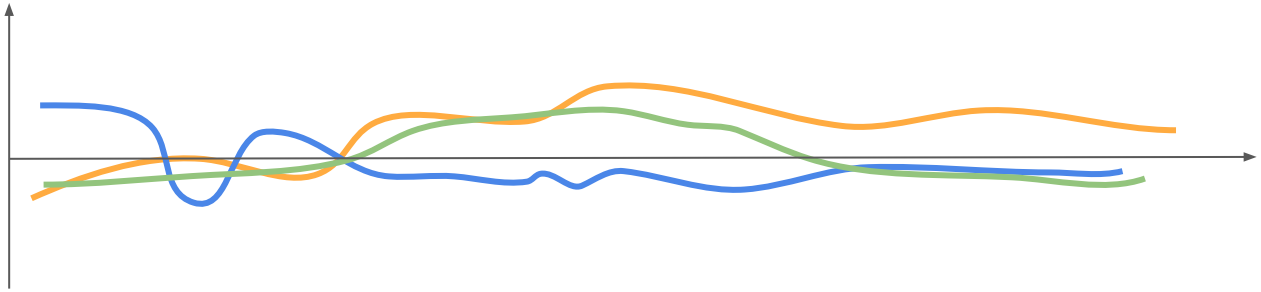
NORMALIZE YOUR DATA

For every sensor, subtract mean and divide by variance

Required for convergence of neural net

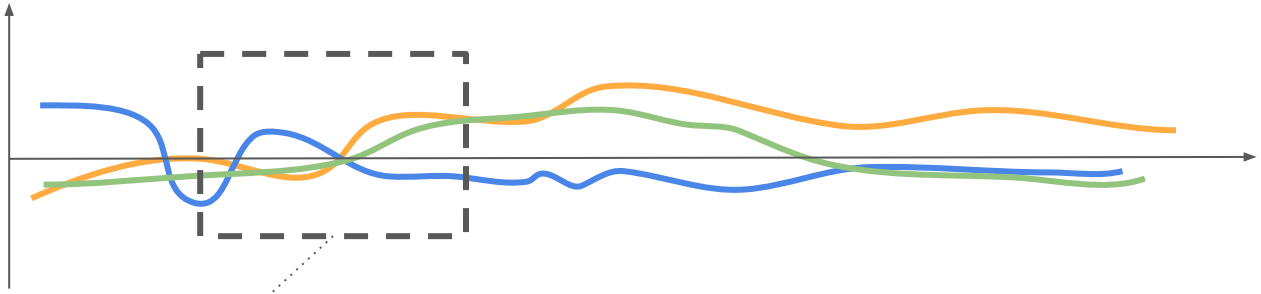


ADD LABELS

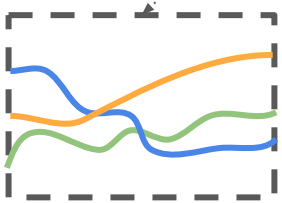


Labels 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0

DIVIDE DATA INTO WINDOWS



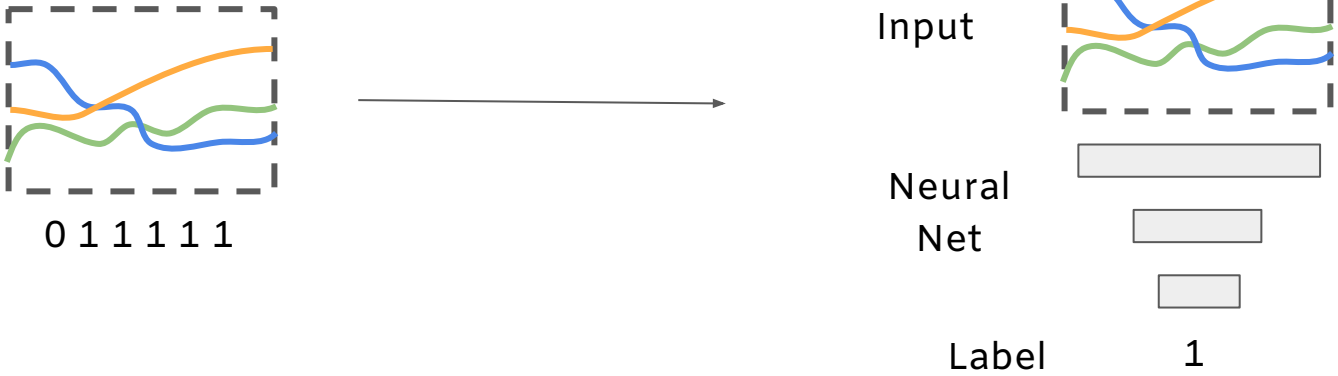
Labels 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0



0 1 1 1 1 1

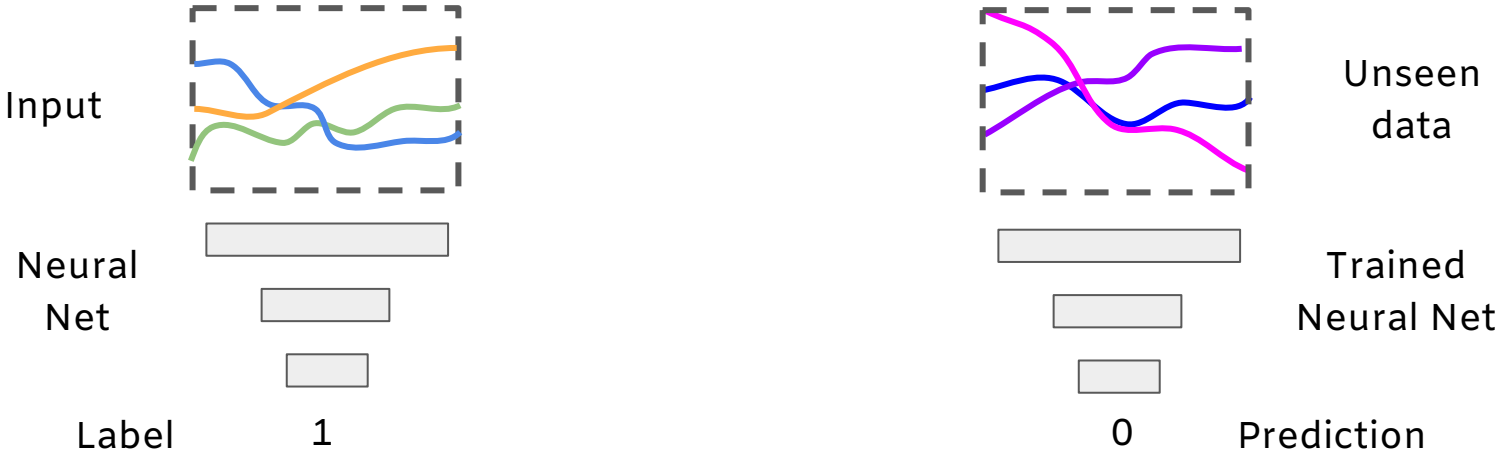
MODEL SETUP

Build model that takes as input sensor data
and outputs label at last timestep



MODEL SETUP

Adjust weights of neural net on labelled examples, test on unseen data



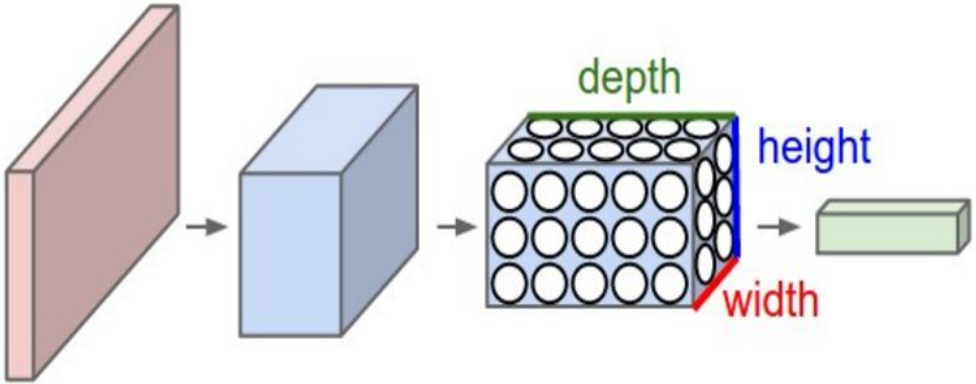
WHICH MODEL TO USE?

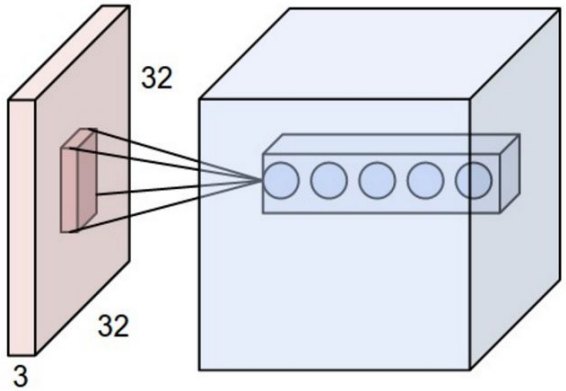
RNN is most obvious choice for time series

However, RNNs are much harder to train, have a hard time dealing with long term time dependencies (at least in the current state), generally less stable

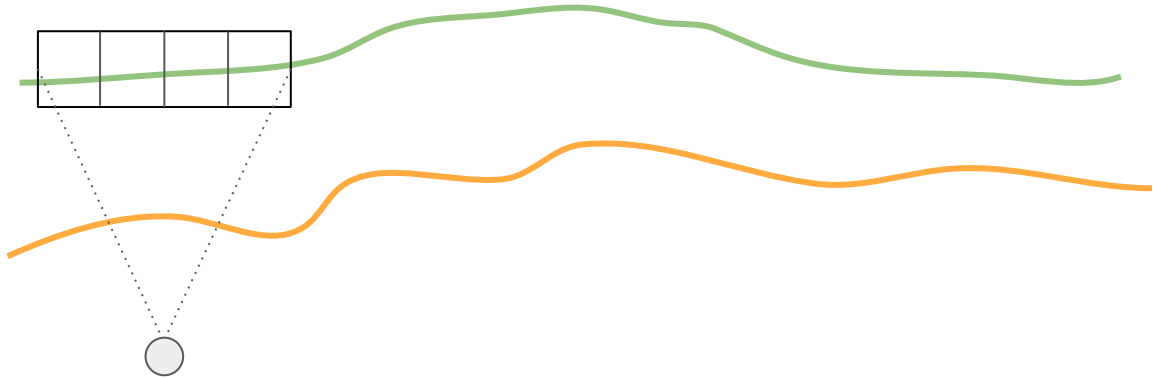
In my experience 1D CNNs have worked well

CONVOLUTIONAL NEURAL NETWORKS

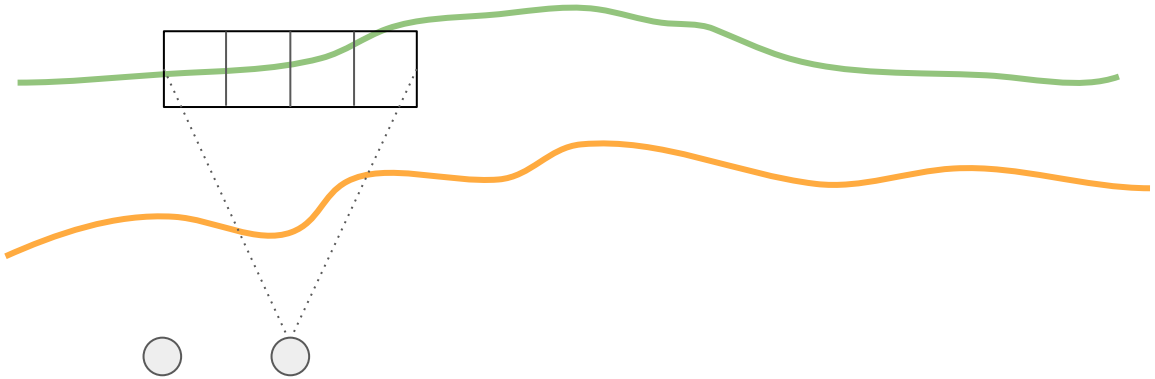




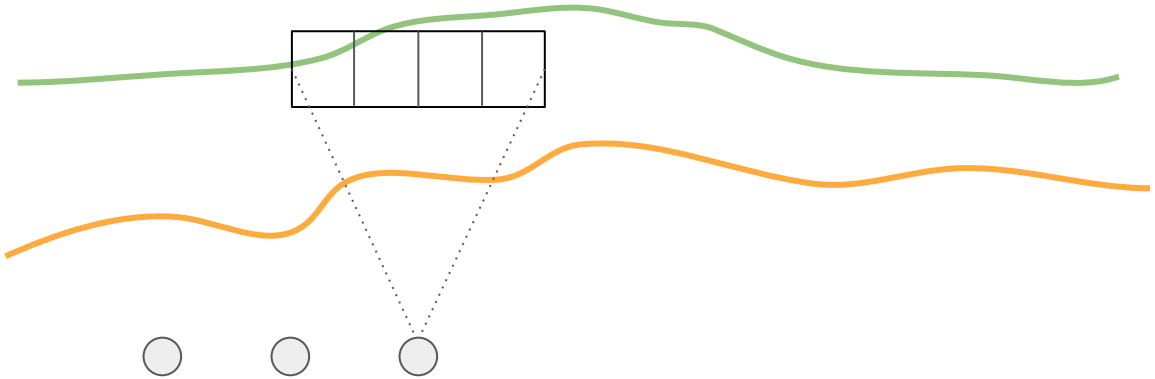
CONVOLUTIONAL NEURAL NETS FOR TIME SERIES



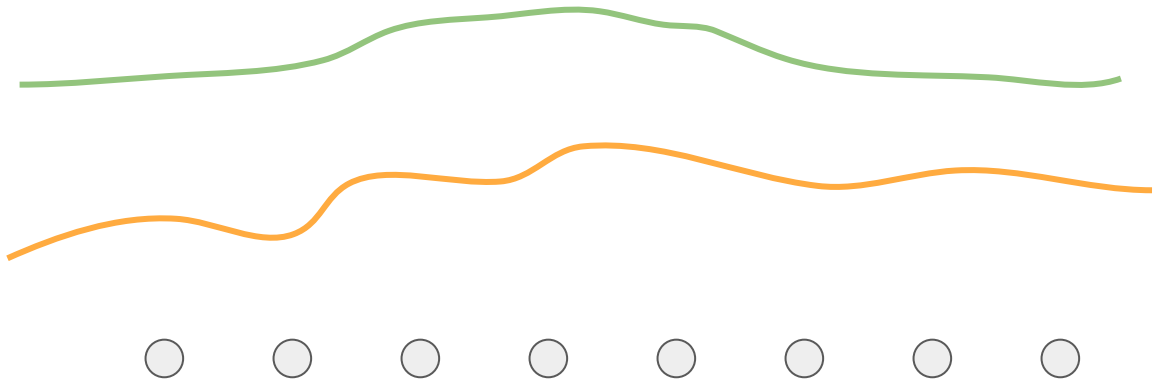
CONVOLUTIONAL NEURAL NETS FOR TIME SERIES



CONVOLUTIONAL NEURAL NETS FOR TIME SERIES

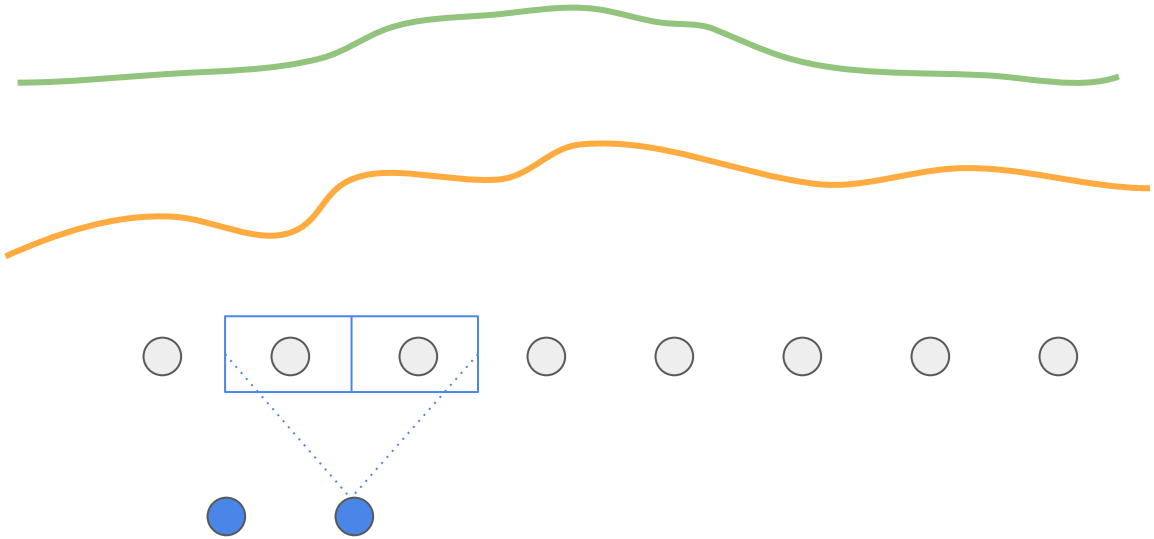


CONVOLUTIONAL NEURAL NETS FOR TIME SERIES



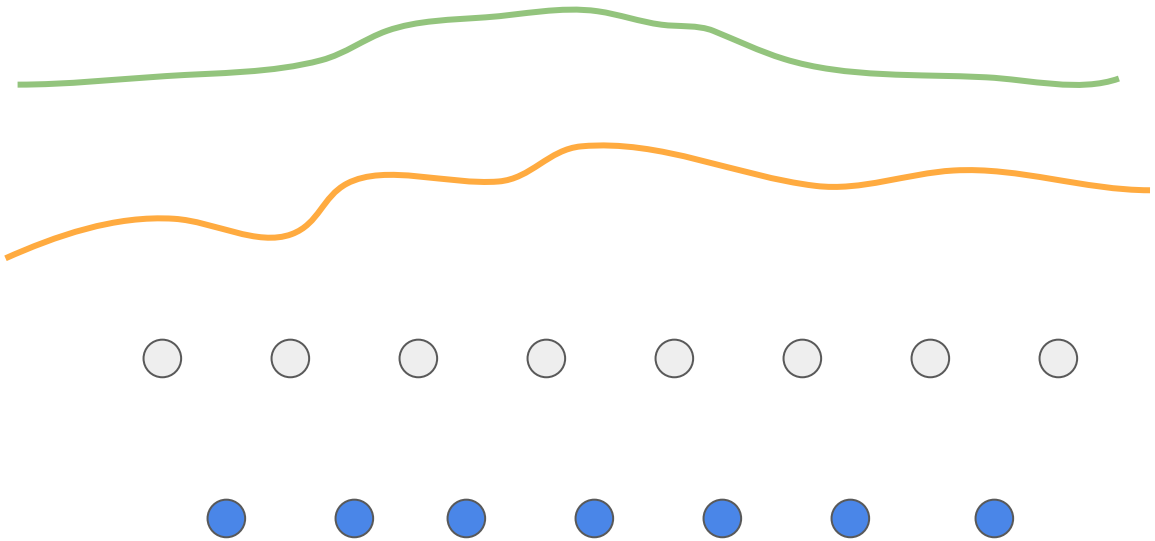
CONVOLUTIONAL NEURAL NETS FOR TIME SERIES

CONVOLUTIONAL NEURAL NETS FOR TIME SERIES



Note: CNNs can capture time dependencies longer than its filter length

CONVOLUTIONAL NEURAL NETS FOR TIME SERIES



Note: CNNs can capture time dependencies longer than its filter length

SUMMARY

Can capture time dependencies and correlation between sensors to make predictions

Normalize data

Use windows of data

Always have a baseline model

Can set up deep learning model with about 15 lines of code

UNSUPERVISED DEEP LEARNING FOR PHM

WHAT IS UNSUPERVISED DEEP LEARNING?

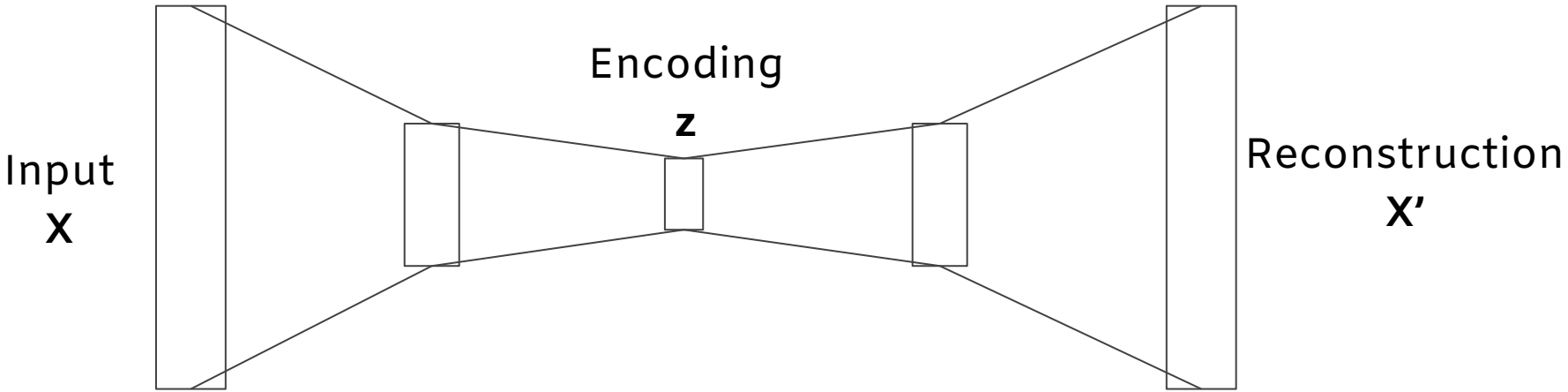
Broadly two categories:

Autoencoders

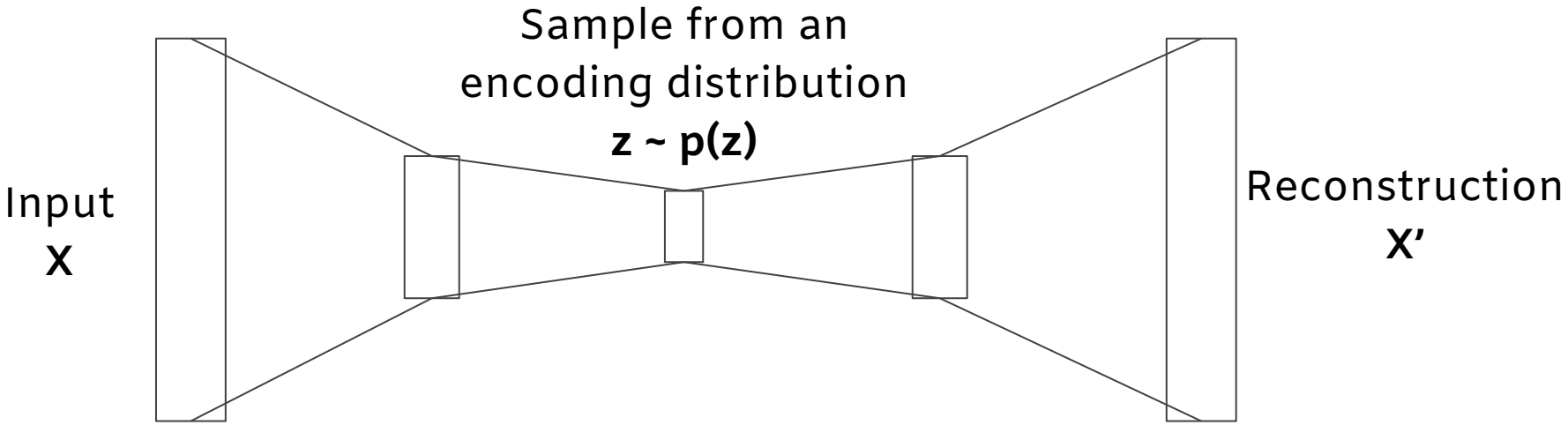
- Non Probabilistic
- Probabilistic: VAE

Generative Adversarial Networks

WHAT ARE AUTOENCODERS?

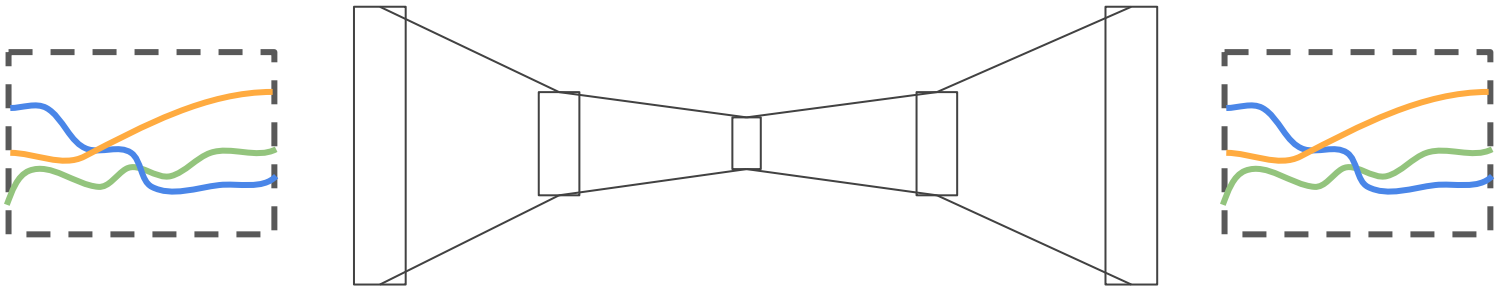


WHAT ARE VARIATIONAL AUTOENCODERS?



AUTOENCODERS FOR ANOMALY DETECTION

Use autoencoders to learn an unsupervised model of normal and abnormal behaviour



AUTOENCODERS FOR ANOMALY DETECTION

Supposedly after seeing a lot of data, the autoencoder will learn an encoding that **efficiently** encodes **normal** data

Passing previously unseen data through the autoencoder, we would expect **normal** data to be **efficiently** encoded and **abnormal** data to have **poor** encodings

AUTOENCODERS FOR ANOMALY DETECTION

How to measure quality of encodings?

Reconstruction loss!

Normal: Low reconstruction loss

Abnormal: High reconstruction loss

In VAE, as a bonus, reconstruction loss has a probabilistic interpretation

FUTURE IDEAS FOR DEEP LEARNING IN PHM

WHAT TO EXPECT?

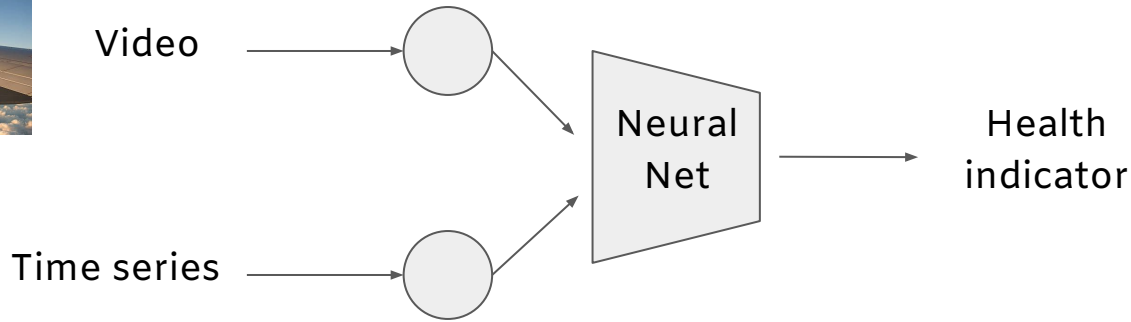
Models that can use more varied data

More data (hopefully!)

Solving entirely different types of problems (e.g. learning control with reinforcement learning)

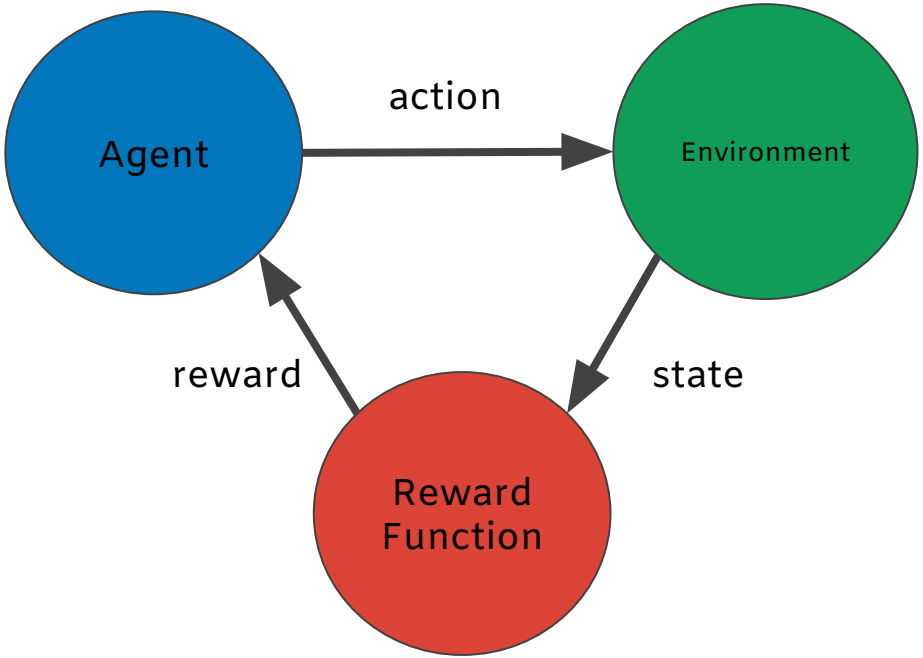
INFERENCE FROM MIXED DATA

For example a camera filming some equipment while sensors are making some measurements in form of time series



REINFORCEMENT LEARNING FOR PHM

REINFORCEMENT LEARNING FOR PHM



REINFORCEMENT LEARNING FOR PHM

Imagine a system where we have an accurate physical simulation of an asset

Let AI agent learn optimal actions to take in order to minimize wear while solving the task at hand

THANK YOU!

Photo: Nvidia