10th November 2022

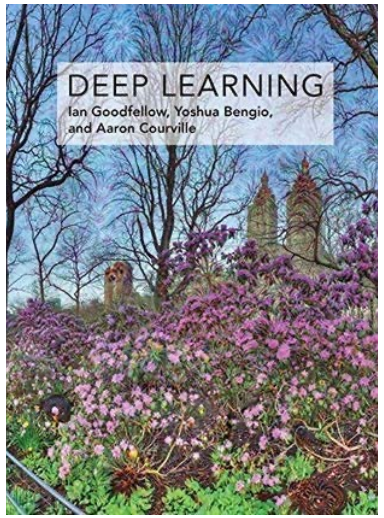
ML 4 Time-series:
Learning time-series features

crackling

Dr. Andrew P. Creagh | Dr. Anshul Thakur | Prof. David A. Clifton

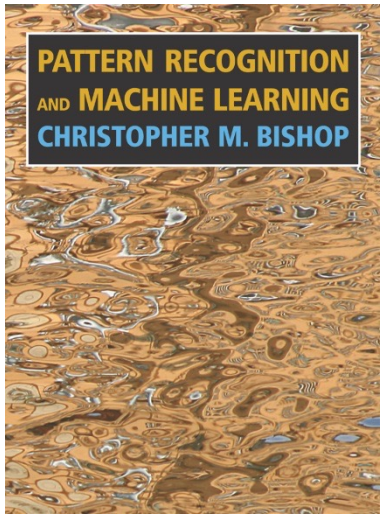
Institute of Biomedical Engineering (IBME),
Department of Engineering Science,
Old Road Campus Research Building (ORCRB),
University of Oxford

Resources



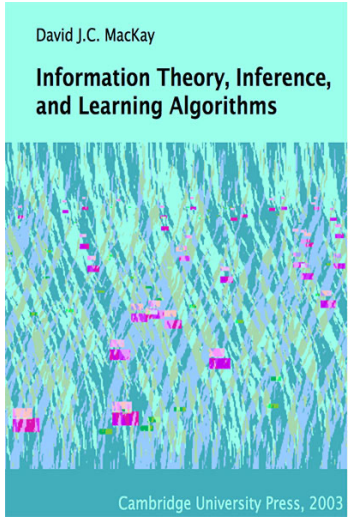
A well-written introduction to all things deep learning (DL), from leaders in the field of theoretical DL.

Very light maths



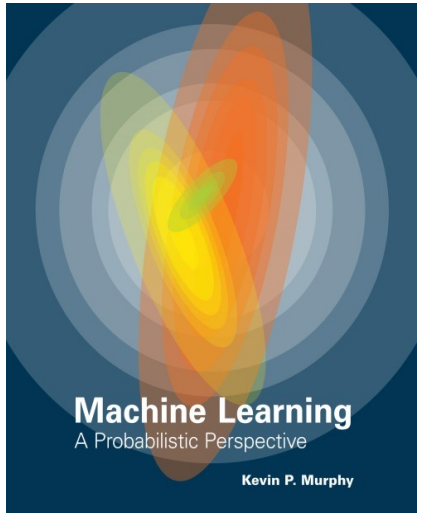
A core classic describing most non-DL algorithms. Very good for one's general understanding.

Reasonably "maths-y"



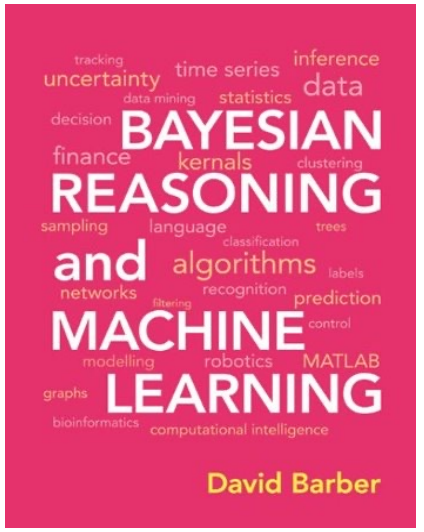
Another core classic, from one of the field's (sadly departed) foundational thinkers – good, even though it's from Cambridge

Reasonably "maths-y"



Seriously comprehensive, one of the best books for general machine learning. Excellent examples.

Really rather "maths-y"



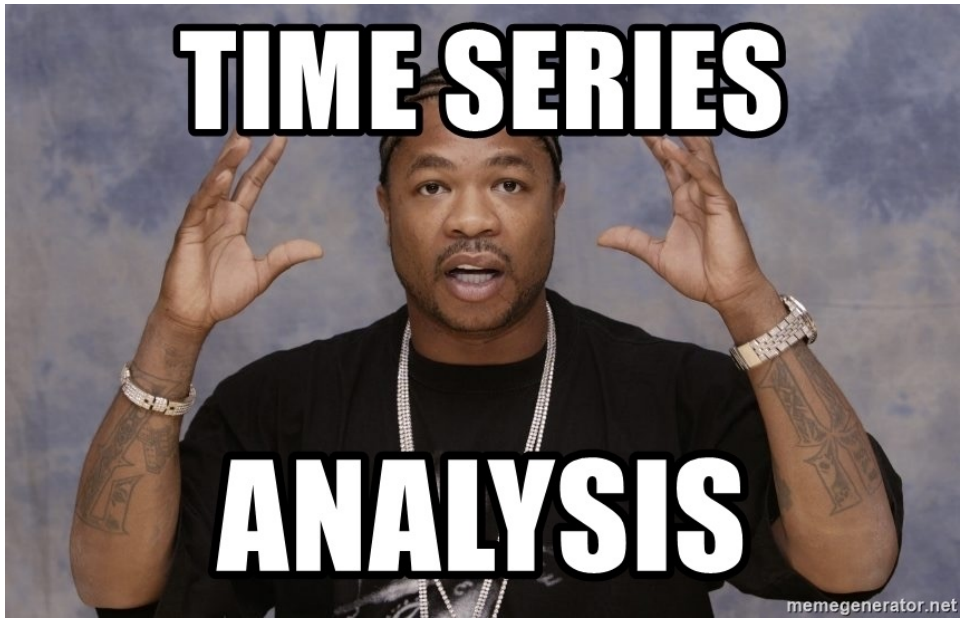
Big, bad, and Bayesian. Everything you'd like to know about Bayesian machine learning. Great for time-series analysis.

Seriously "maths-y"

mild — medium — hot — extra hot

Why time?

Clinical data is **not** static: clinical measurements are often a time-series, (e.g. heart rate) or repeated measurements taken during visits to the doctor, or a stay in hospital. Clinical events can occur at a point in time (e.g. a heart attack) or can even be causal. These events generally evolve over time (known as **Stochastic processes)**



time

Choosing your model

Models that don't consider time (i.i.d.)

- Linear / Logistic Regression
- Support Vector Machines (SVM)
- Classification and Regression Trees (CART) (e.g. Random Forest)
- eXtreme Gradient Boosting (XGBoost)
- Deep Neural Networks
- Etc.

Need to incorporate time through the features instead!

Image Credit

Choosing your model

Models that don't consider time (i.i.d.)

- Linear / Logistic Regression
- Support Vector Machines (SVM)
- Classification and Regression Trees (CART) (e.g. Random Forest)
- eXtreme Gradient Boosting (XGBoost)
- Deep Neural Networks
- Etc.

Need to incorporate time through the features instead!

Models that consider time (non i.i.d)

- Markov Chains
- Autoregressive Processes
- Hidden Markov Models (HMM)
- Recurrent Neural Networks (RNN) /
  Long-Short Term Memory (LSTM)
- Transformers,
- Etc.

Image Credit

Choosing your model

Models that don't consider time (i.i.d)

- Linear / Logistic Regression
- Support Vector Machines (SVM)
- Classification and Regression Trees (CART) (e.g. Random Forest)
- eXtreme Gradient Boosting (XGBoost)
- Deep Neural Networks
- Etc.

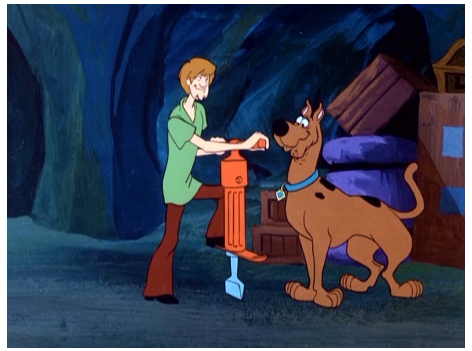
Need to incorporate time through the features instead!

Models that consider time (non i.i.d)

- Markov Chains
- Autoregressive Processes
- Hidden Markov Models (HMM)
- Recurrent Neural Networks (RNN) / Long-Short Term Memory (LSTM)
- Transformers,
- Etc.

Image Credit

Is a more complex model *necessary*?



The importance of data curation, cleaning, and pre-processing are often overlooked

Time-domain approaches for modelling data

Model driven and generative methodologies based on the raw time-series data

- Markov Chains
- Autoregressive Models
- Autocorrelation functions
- Hidden Markov Models (HMM)

Markov Chains

For a timeseries: $\mathbf{x} = \{x_t, x_{t+1}, x_{t+2}, \cdots, x_T\}$, we need a model $p(\mathbf{x})$:

$$p(\mathbf{x}) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{1:t-1}) \qquad p(x_t|x_{1:t-1}) = p(x_1) \text{ for } t = 1$$

initial transition

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)\, p(x_3|x_2) \cdots p(x_T|x_{T-1})$$

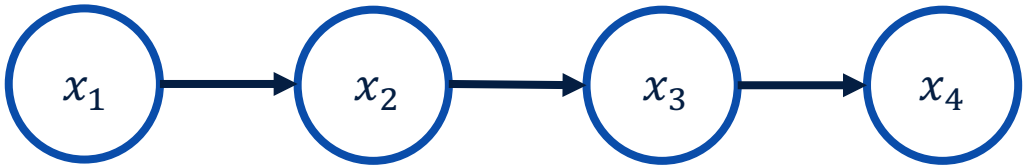
Assuming only the recent past is relevant:

$$p(x_t|x_{1:t-1}, \cdots x_{t-1}) = p(x_t|x_{1:t-L}, \cdots, x_{t-1})$$

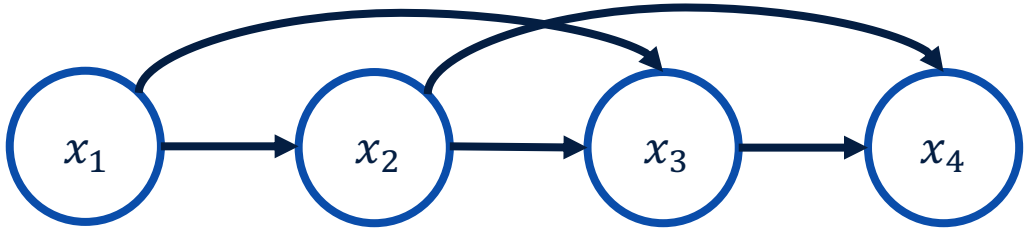
where $L \geq 1$ is the *order* of the Markov chain

Independence assumptions
We only assume that the influence of the immediate past is more relevant than the remote past and in Markov models only a limited number of previous observations are required to predict the future.

Fit by Maximum Likelihood



(a) First order Markov Chain



(b) Second order Markov Chain

Auto-Regressive Models

(are Continuous-state Markov Models)

The timeseries value x_t can modelled by a weighted sum of previous values:

$$x_t \approx \sum_{l=1}^{L} a_l x_{t-l} + c$$

a_l we can call the 'AR coefficients'

We can view this then as a form of regression, and find the AR coefficients by minimising the squared loss:

$$\sum_t \left(x_t - \sum_{l=1}^{L} a_l x_{t-l} \right)^2$$

This is a quadratic function, we can easily solve this!

Maximum likelihood estimation (MLE)

Autoregression (AR) models can also be generalised to the order l by

$$AR(l) = x_t \approx a_1\, x_{t-1} + a_2\, x_{t-2} + \cdots + a_p\, x_{t-l+c}$$

Autoregressive models (AR)

(are Continuous-state Markov Models)

The timeseries value x_t can modelled by a weighted sum of previous values:

$$x_t \approx \sum_{l=1}^{L} a_l x_{t-l} + c$$

a_l we can call the 'AR coefficients'

Autocorrelation:

We can estimate the relationship between the value at any point t, x_t, and the value at any point $(t-1)$, x_{t-1} using a first order model $AR(1)$:

$$AR(1) = x_t \approx a_1 x_{t-1} + c$$

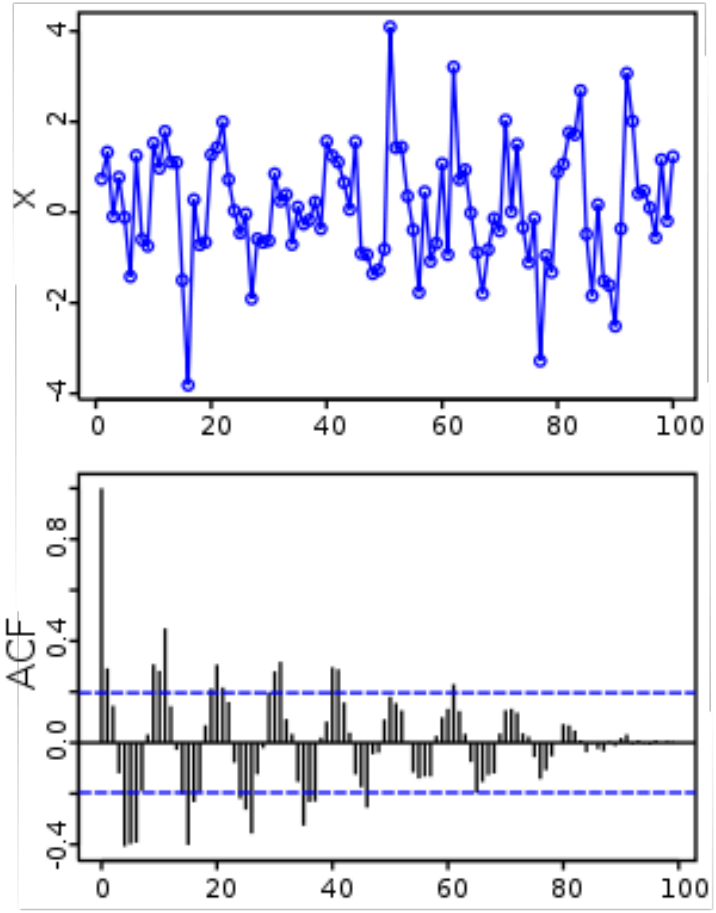
where a_1 are the weights of the model and c is the constant. The autocorrelation function measures the correlation between x_t and x_{t+k} , at various lags, k, where $k = 0, \ldots, K$. The autocorrelation for lag k is:

$$a_k = \frac{c_k}{c_0}$$

$$c_k = \frac{1}{T}\sum_{t=1}^{T-k}(x_t - \bar{\bar{\mathbf{x}}})(x_{t+k} - \bar{\bar{\mathbf{x}}})$$

$\bar{\bar{\mathbf{x}}}$: mean of $\mathbf{x}$

c_0: sample variance in $\mathbf{x}$



A plot of a series of 100 random numbers concealing a sine function. Below: The sine function revealed in a correlogram produced by autocorrelation.

Hidden Markov Models (HMM)

Markov property that only the present influences the future

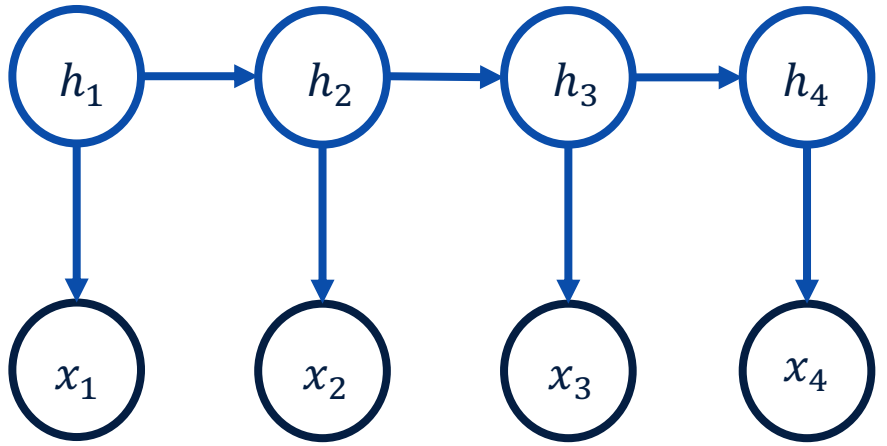
The Hidden Markov Model (HMM) defines a Markov chain on hidden (or `latent') variables $h_t = \{h_1, h_2, \cdots, h_H\}$

The observed (or `visible') variables are dependent on the hidden variables through an emission $p(x_t|h_t)$. This defines a joint distribution:

$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

emission prob

transition prob

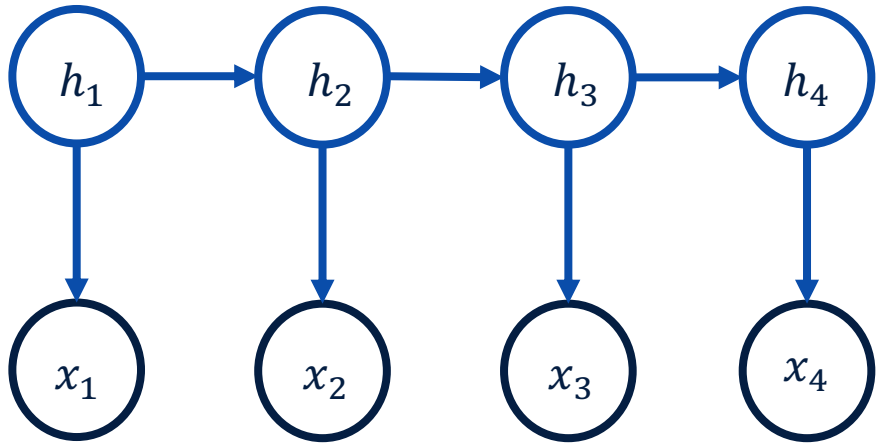


Hidden Markov Models (HMM)

Markov property that only the present influences the future

The Hidden Markov Model (HMM) defines a Markov chain on hidden (or `latent') variables $h_t = \{h_1, h_2, \cdots, h_H\}$

The observed (or `visible') variables are dependent on the hidden variables through an emission $p(x_t|h_t)$. This defines a joint distribution:



$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

emission prob

transition prob

1. The transition distribution $p(h_{t+1}|h_t)$ is defined by a $H \times H$ **transition** matrix: $\mathbf{A}_{i',i} = p(h_{t+1} = i'|h_t = i)$
2. The emission distribution, $p(x_t|h_t)$ has discrete states $x_t \in \{1, \cdots, V\}$, we can define a $V \times H$ **emission** matrix: $\mathbf{B}_{i,j} = p(x_t = i|h_t = k)$

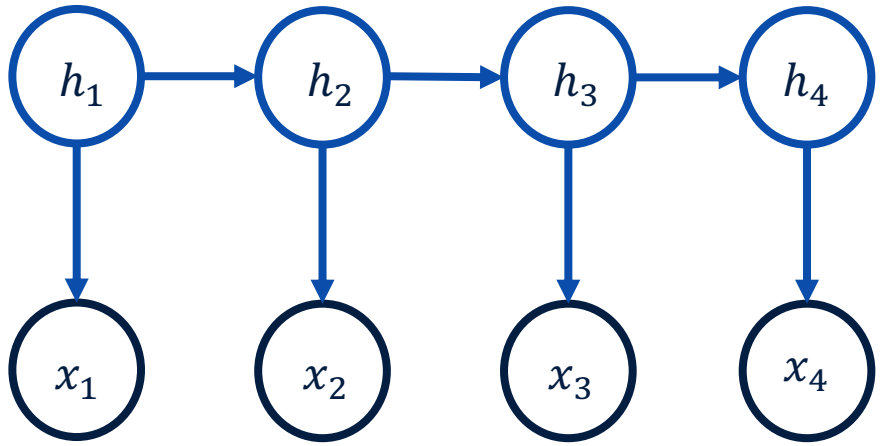
   For continuous outputs, h_t selects one of H possible output distributions $p(x_t|h_t), h_t \in \{1, \cdots, H\}$.

Hidden Markov Models (HMM)

Markov property that only the present influences the future

The Hidden Markov Model (HMM) defines a Markov chain on hidden (or `latent') variables $h_t = \{h_1, h_2, \cdots, h_H\}$

The observed (or `visible') variables are dependent on the hidden variables through an emission $p(x_t|h_t)$. This defines a joint distribution:



$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

emission prob

transition prob

1. The transition distribution $p(h_{t+1}|h_t)$ is defined by a $H \times H$ **transition** matrix: $\mathbf{A}_{i',i} = p(h_{t+1} = i'|h_t = i)$
2. The emission distribution, $p(x_t|h_t)$ has discrete states $x_t \in \{1, \cdots, V\}$, we can define a $V \times H$ **emission** matrix: $\mathbf{B}_{i,j} = p(x_t = i|h_t = k)$

   For continuous outputs, h_t selects one of H possible output distributions $p(x_t|h_t), h_t \in \{1, \cdots, H\}$.
3. Most likely Hidden path, $\arg\max_{\mathbf{h}} p(\mathbf{h}, \mathbf{x})$ is found via the ***Viterbi*** algorithm (*don't worry too much about this*)

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

Representing Time

Extracting time-domain & spatio-temporal hand-crafted features from time-series data

Fourier transform

Short-time Fourier transform

Continuous Wavelet Transform (CWT)

Discrete Wavelet Transform (DWT)

Spectral Features

Fourier Transform (recap)

The Fourier transform of a function represents a signal as an infinite weighted sum of an infinite number of sinusoids

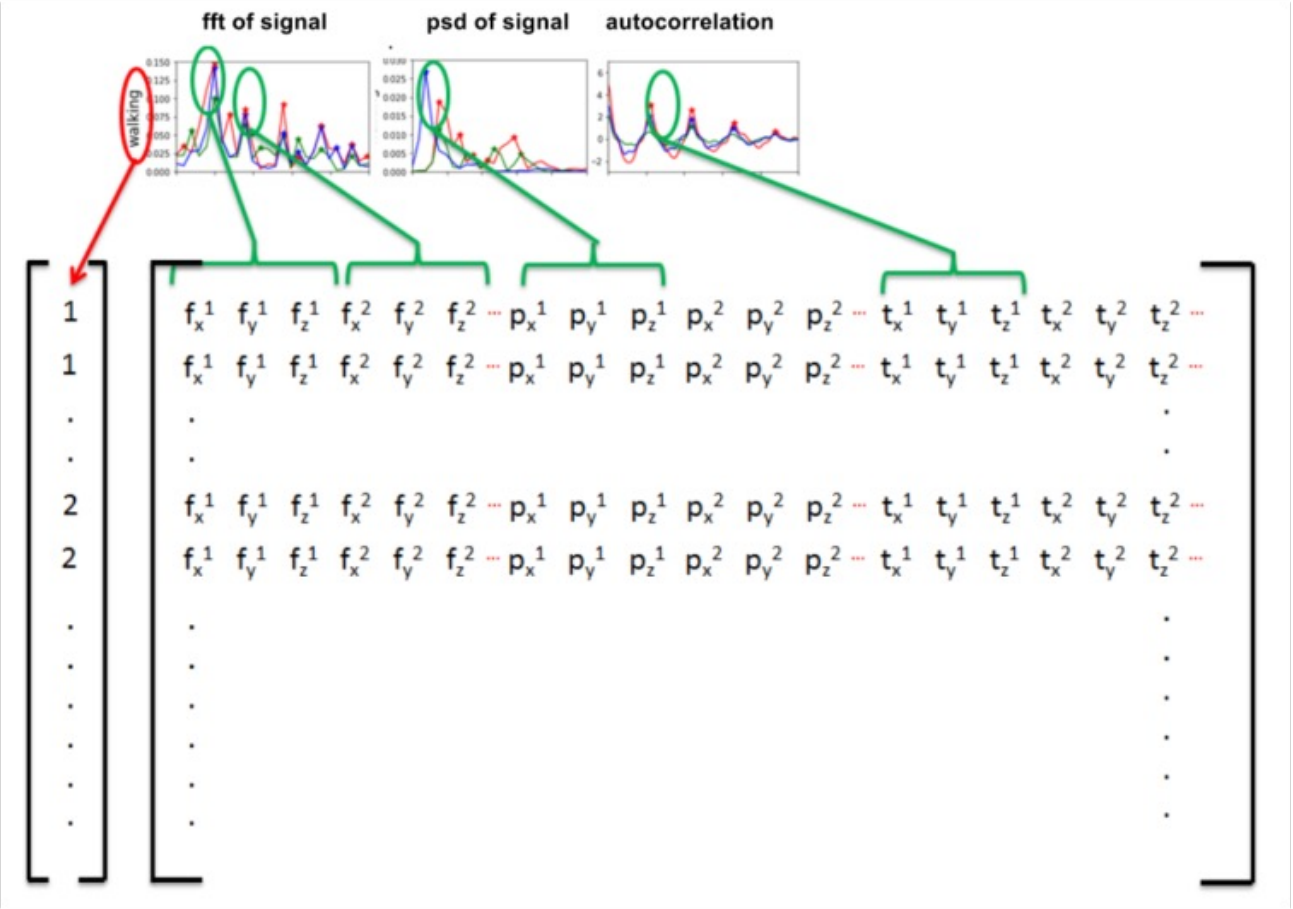
(Frequency Spectrum $\hat{f}(u)$)

$$\hat{f}(u) = \int_{-\infty}^{+\infty} f(x) e^{-i2\pi ux} dx$$

$$e^{ik} = \cos k + \sin k \; ; i = \sqrt{-1}$$

Spatial/time Domain (x) ⟶ Frequency Domain (u)

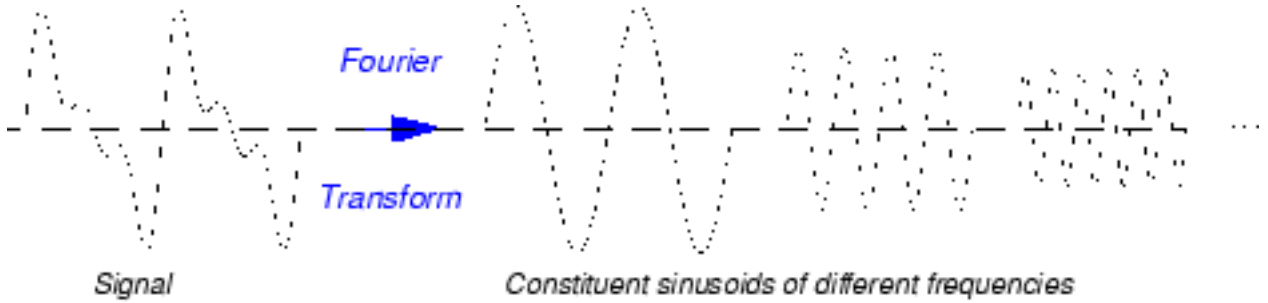
Time-domain hand-crafted features

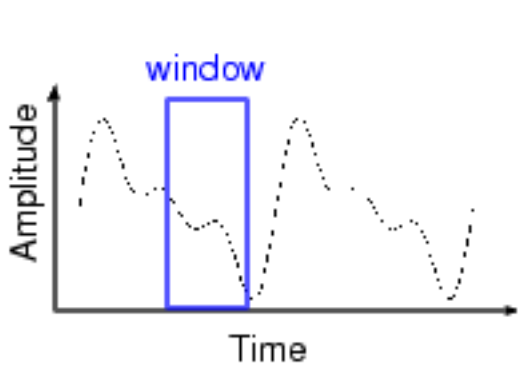


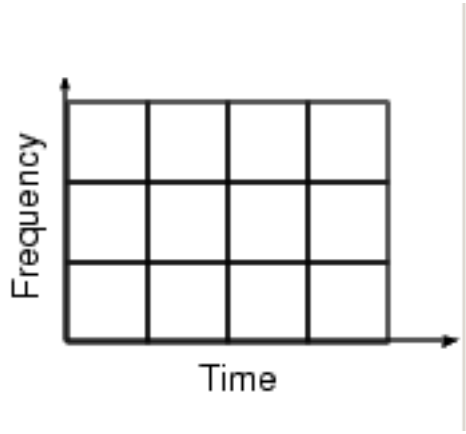
The location of the peaks in the FFT, PSD and autocorrelation can be used as features for our ML model

STFT (recap)

The short-time Fourier transform (STFT) is used to analyse how the frequency content of a nonstationary signal changes over time.



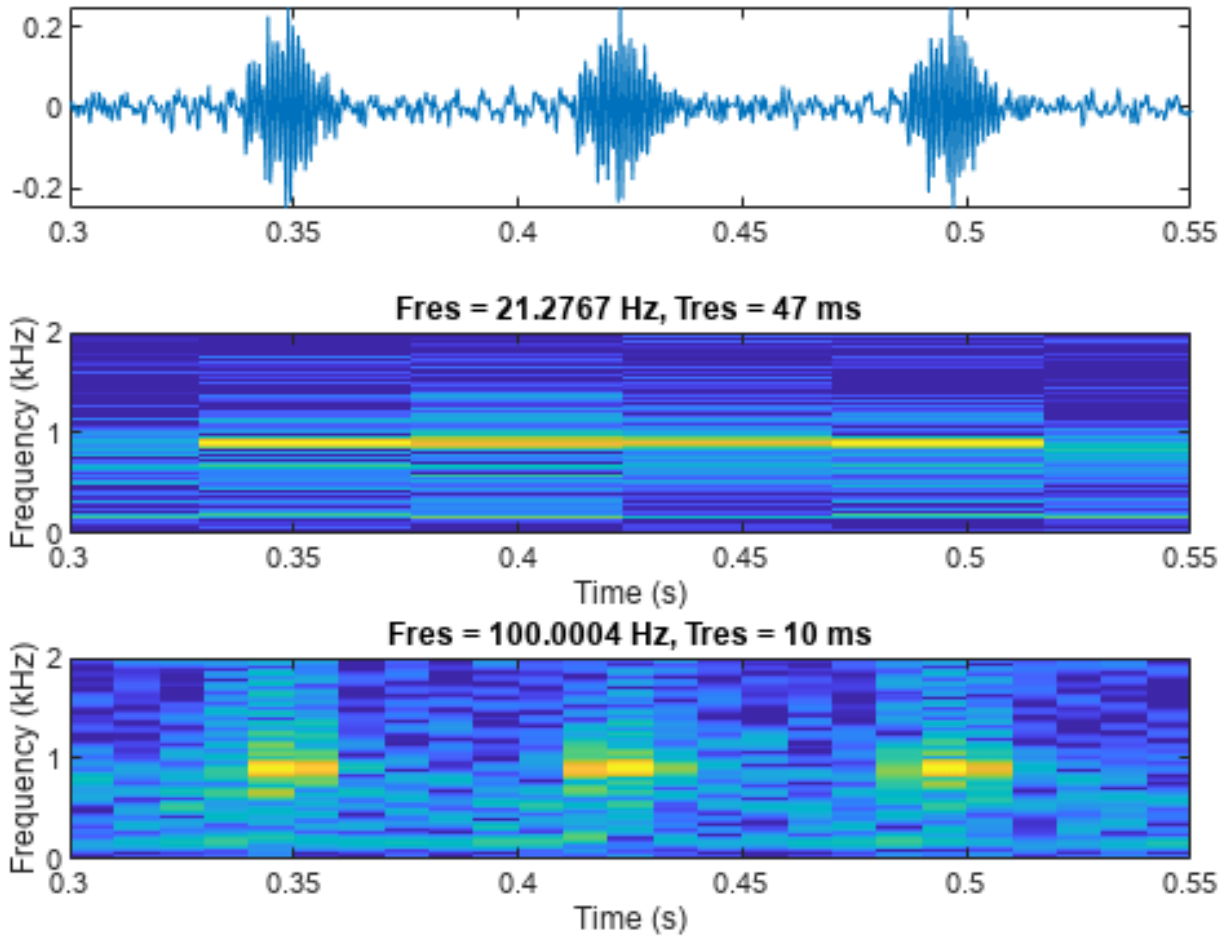




STFT (recap)

Trading Off Time and Frequency Resolution to Get the Best Representation of Your Signal

spectrogram

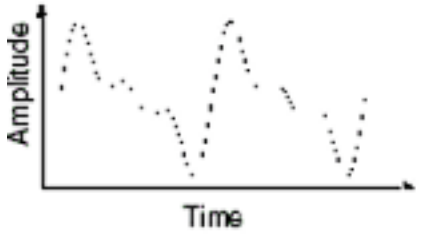


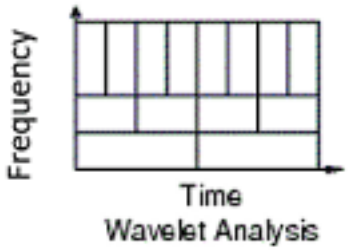
Longer segments provide better frequency resolution; shorter segments provide better time resolution

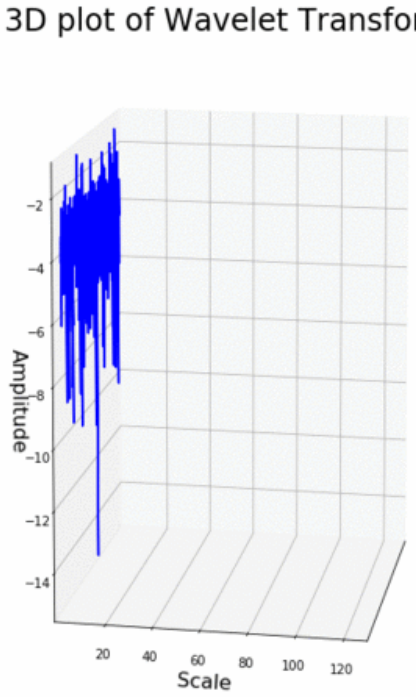
Continuous wavelet transform (CWT)(recap)

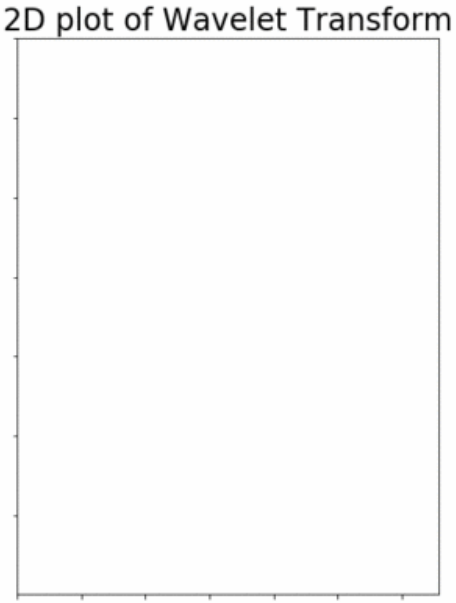
- CWT is a method used to measure the similarity between a signal and an analysing function which can provide a precise time-frequency representation of a signal.

- The CWT compares the signal to shifted and compressed (*dilation)* or stretched (*scaling)* versions of a wavelet.
- A (continuous) wavelet transform is given by the overlap integral of the signal s with the kernel (wavelet) w, where the kernel is both shifted and stretched, allowing to extract information at different scales.









source: ataspinar.com

Continuous wavelet transform (CWT)

- CWT is a method used to measure the similarity between a signal and an analysing function which can provide a precise time-frequency representation of a signal.
- The **discrete-time** version of the CWT for time signal x_n is defined as the convolution of x_n with a scaled and translated mother wavelet $\psi_0(\eta)$:

CWT

$$W_n(s) = \sum_{n'=0}^{N-1} x'_n \psi_0{}^* \left[\frac{(n'-n)\delta_t}{s}\right]$$

Mother Wavelet (e.g. Morlet)

$$\psi_0(\eta) = \frac{1}{\sqrt[4]{\pi}} e^{i\omega_0\eta} \cdot e^{\frac{-\eta^2}{2}}$$

Frequency (Hz)

$$f = \frac{f_c}{s} \quad \text{where} \quad f_c = \frac{\omega_0 + \sqrt{2+\omega_0{}^2}}{4\pi}$$

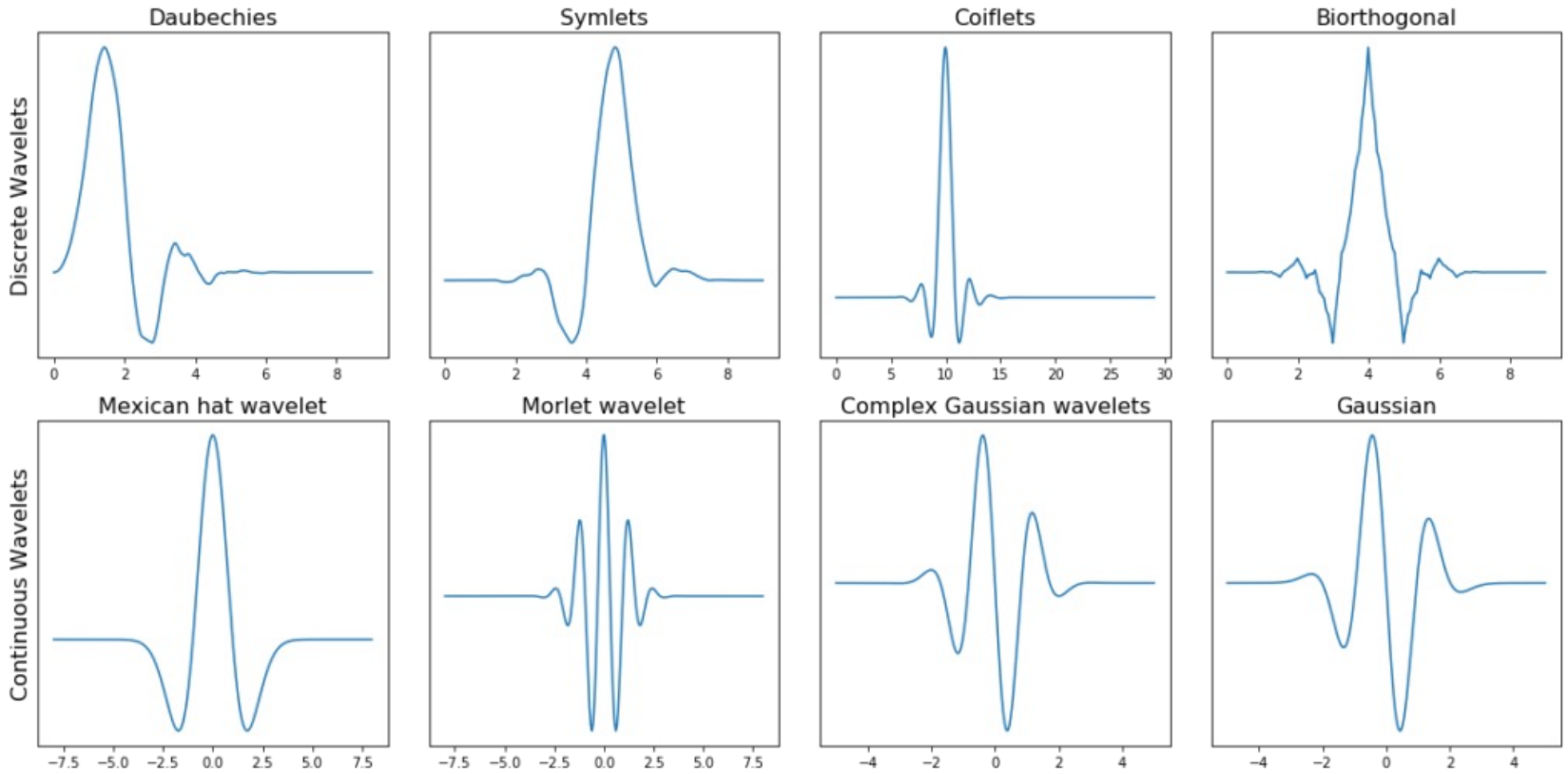
Scale dependent energy

$$E_s = \sum_{n=0}^{N-1} |W_n(s)|^2$$

- $()^*$, denotes the complex conjugate
- s, wavelet scaling factor
- n, localised time index
- η, non-dimensional time parameter
- ω_0, non-dimensional frequency
- f_c , centre frequency for specific wavelet

Types of Mother Wavelets

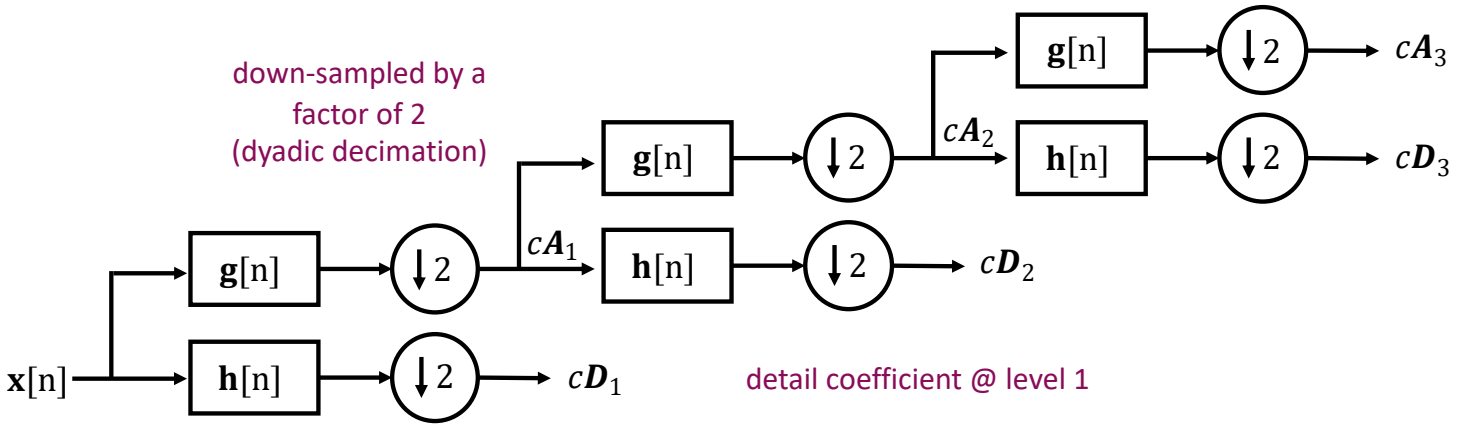
- CWT is a method used to measure the similarity between a signal and an analysing function which can provide a precise time-frequency representation of a signal.



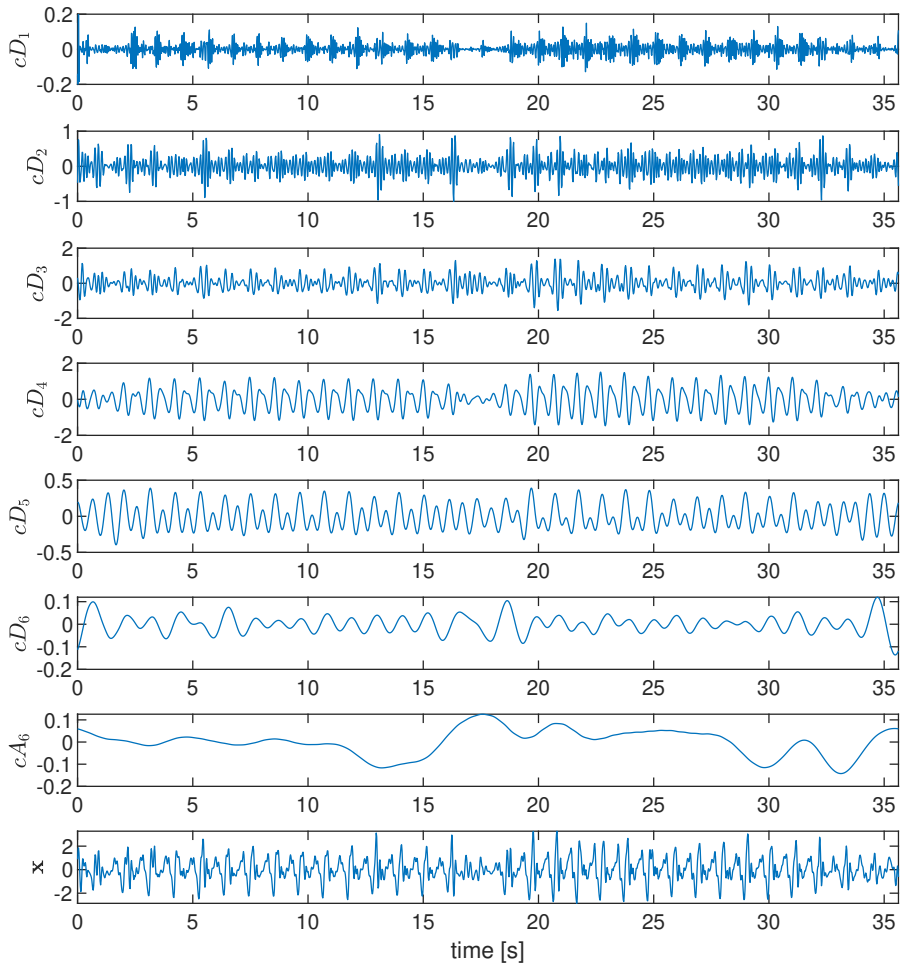
Discrete Wavelet Transform (DWT)

- The DWT provides a decomposition to analyse a signal across various frequency bands with different resolutions using coarse approximation coefficients and detail information detail coefficients.
- A DWT can obtain a multi-resolution representation of $\mathbf{x}$ through a series of cascading filter banks
- At each level L, a series of low-pass $\mathrm{g}[\cdot]$ and high-pass $\mathrm{h}[\cdot]$ filters are applied, evenly bisecting the frequency into its constituent low- and high-frequency components using a convolution between the signal and filter:

$$y[n] = \sum_{k=-L}^{L} \mathrm{x}[k]\mathrm{g}[n-k]$$



Discrete Wavelet Transform (DWT) filter bank



The result is the decomposition of $\mathbf{x}$ into multiple frequency bands, where after each level, the frequency resolution is doubled while the time resolution is halved.

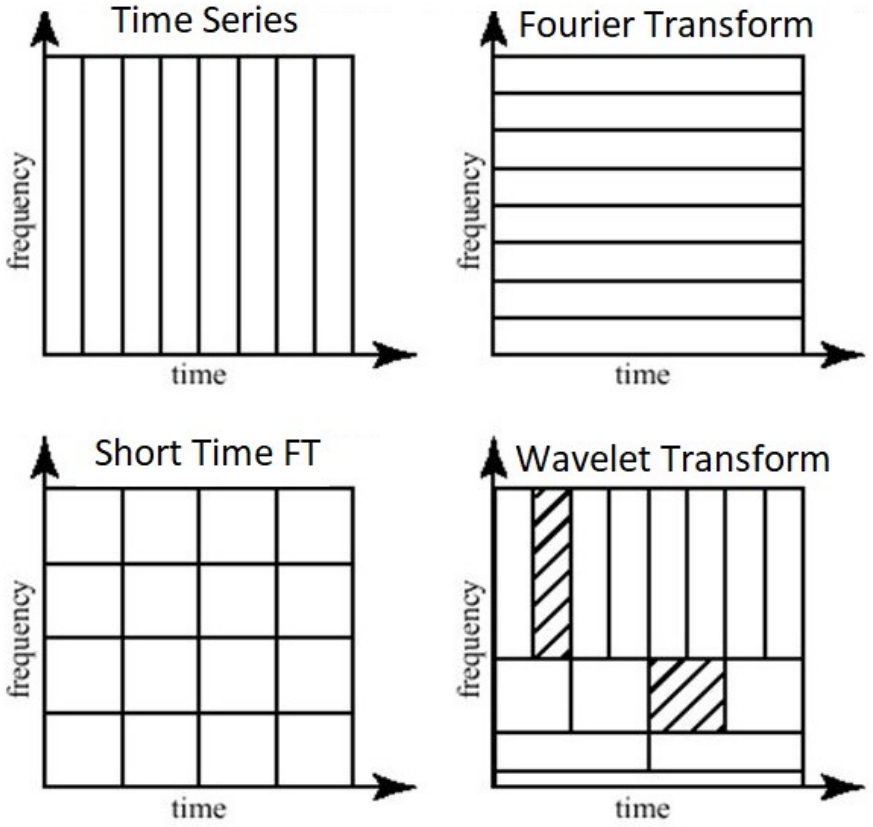
FT vs. STFT vs. CWT vs . DWT

- The major difference between the discretized version CWT and discrete wavelet transform (DWT) is how the scale parameter is discretized.
- CWT discretizes scale more finely than the discrete wavelet transform. In the CWT, you typically fix some base which is a fractional power of two, for example, $2^{j/v} \ \forall \ j = 1,2,3, \dots, N$; where v is an integer greater than 1.
- The resulting discretized wavelets for the CWT are:

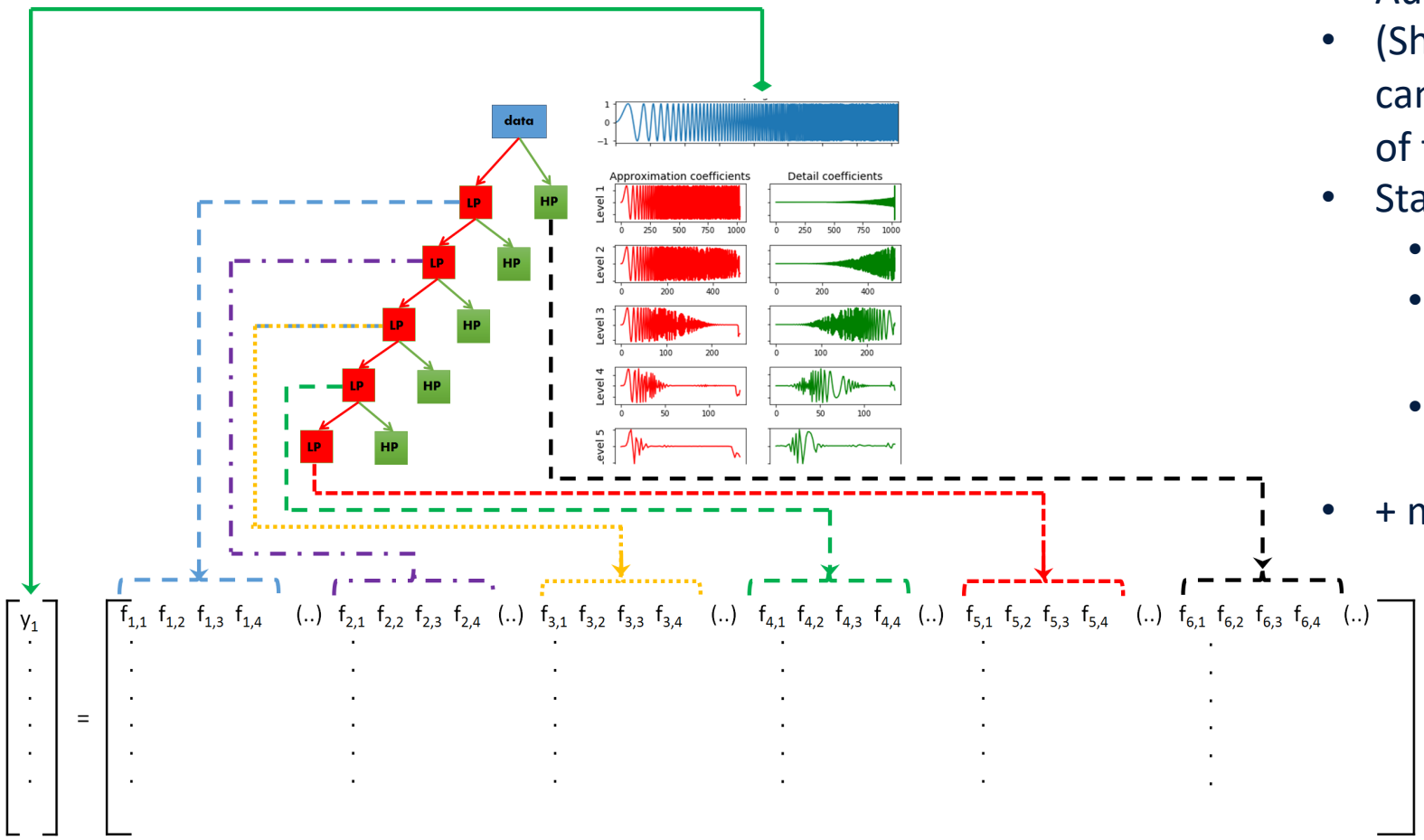
$$\frac{1}{2^{j/v}} \psi \left(\frac{n-m}{2^{j/v}} \right)$$

- DWT scale parameter is always discretized to integer powers of $2^j \ \forall \ j = 1,2,3, \dots, N$.
- In the decimated (down-sampled) discrete wavelet transform (DWT), the translation parameter is always proportional to the scale. This means that at scale, 2^j, you always translate by $2^j m$, where m is a nonnegative integer:

$$\frac{1}{2^j} \psi \left(\frac{1}{2^j} (n - 2^j m) \right)$$



DWT-based hand-crafted features

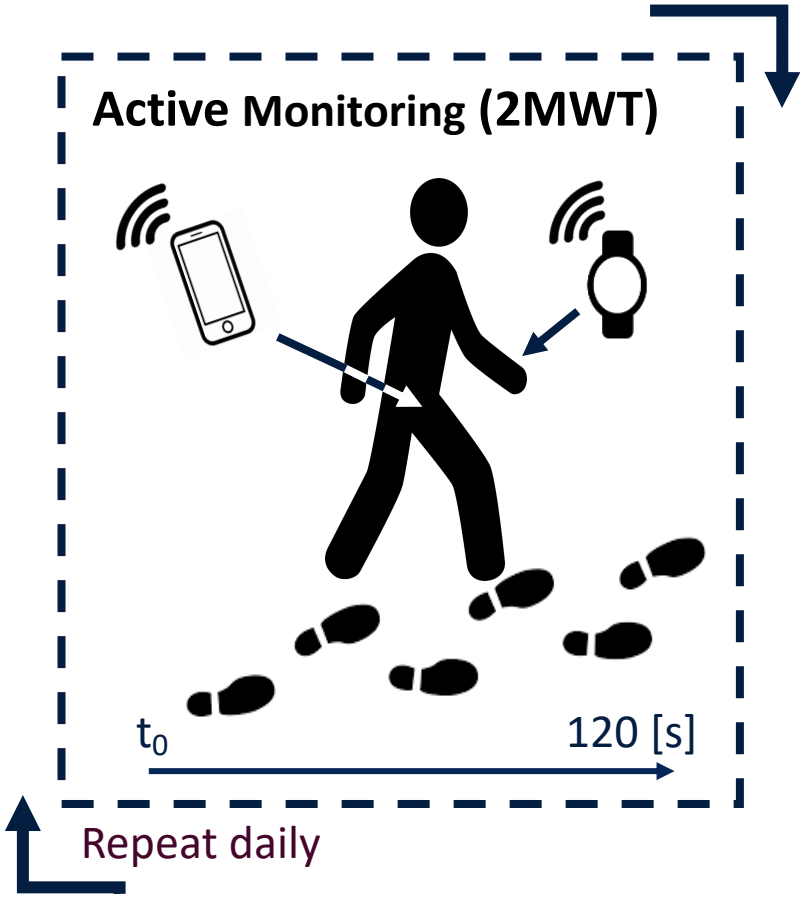


- Auto-regressive model coefficient values
- (Shannon) Entropy values; entropy values can be taken as a measure of complexity of the signal.
- Statistical features e.g.:
  - Mean, variance, percentiles, IQR, etc.
  - Zero crossing rate, i.e. the number of times a signal crosses y = 0
  - Mean crossing rate, i.e. the number of times a signal crosses y = mean(y)
- + more!

By applying the DWT on a signal we can deconstruct it its frequency sub-bands. And out of each sub-band we can generate features which can be used as input for a ML model.

Remote characterisation of ambulatory function using wearables

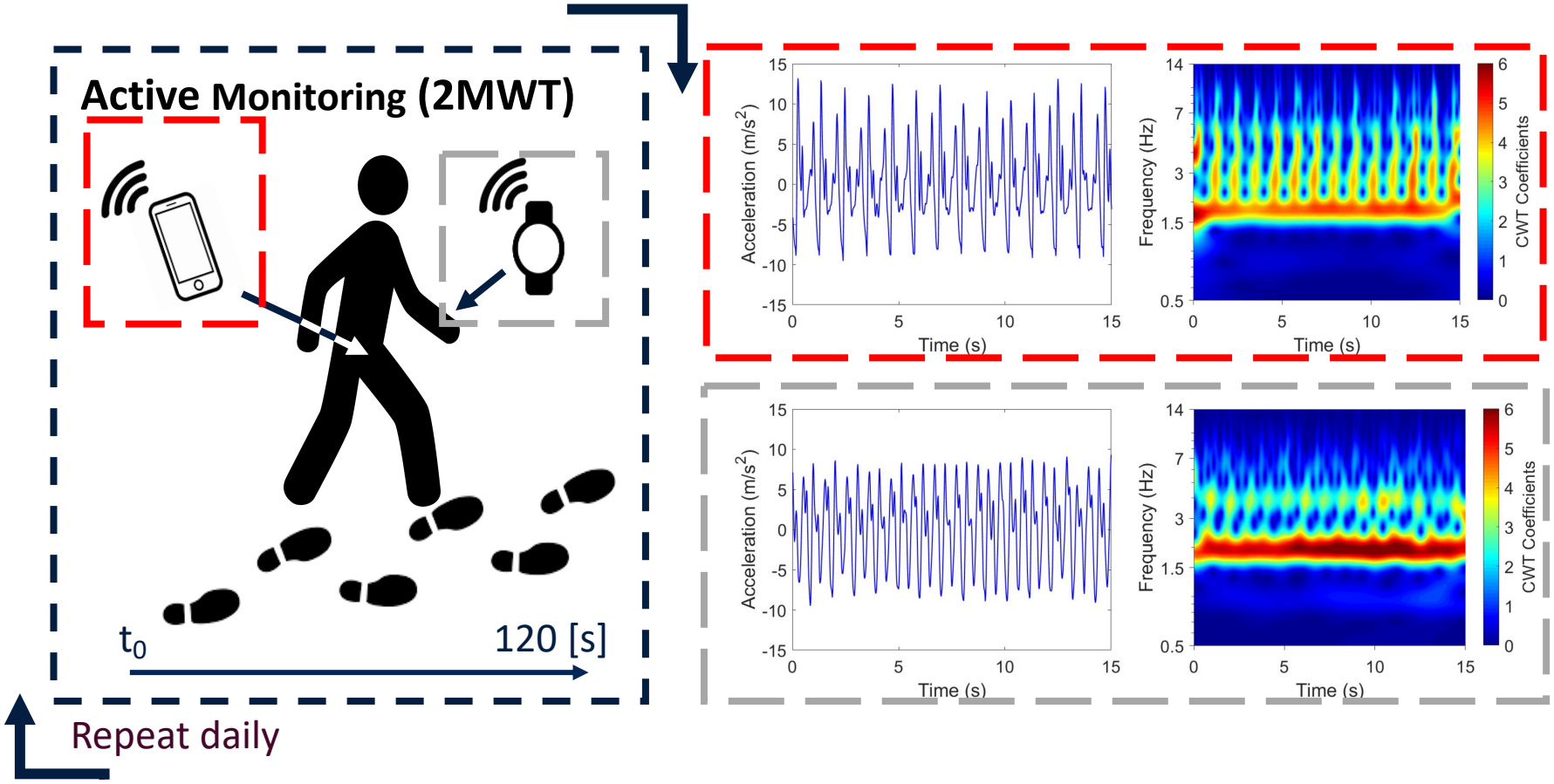
/apcreagh/MS-GAIT_feature_extraction



Creagh, A. P., *et al.* (2020). Smartphone-and smartwatch-based remote characterisation of ambulation in multiple sclerosis during the two-minute walk test. *IEEE Journal of Biomedical and Health Informatics*, *25*(3), 838-849. doi: 10.1109/JBHI.2020.2998187

Remote characterisation of ambulatory function using wearables

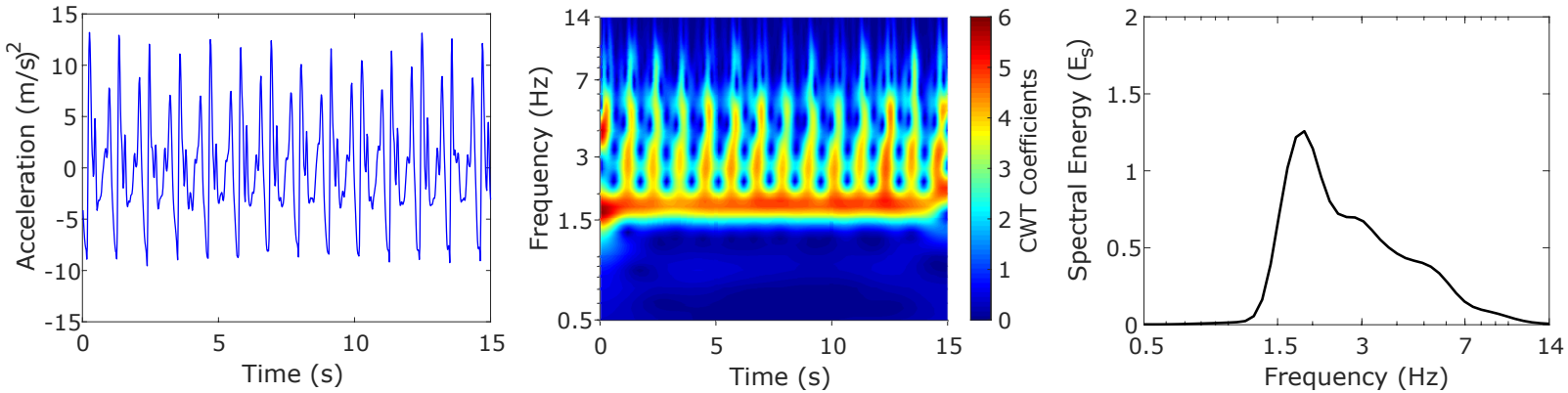
/apcreagh/MS-GAIT_feature_extraction



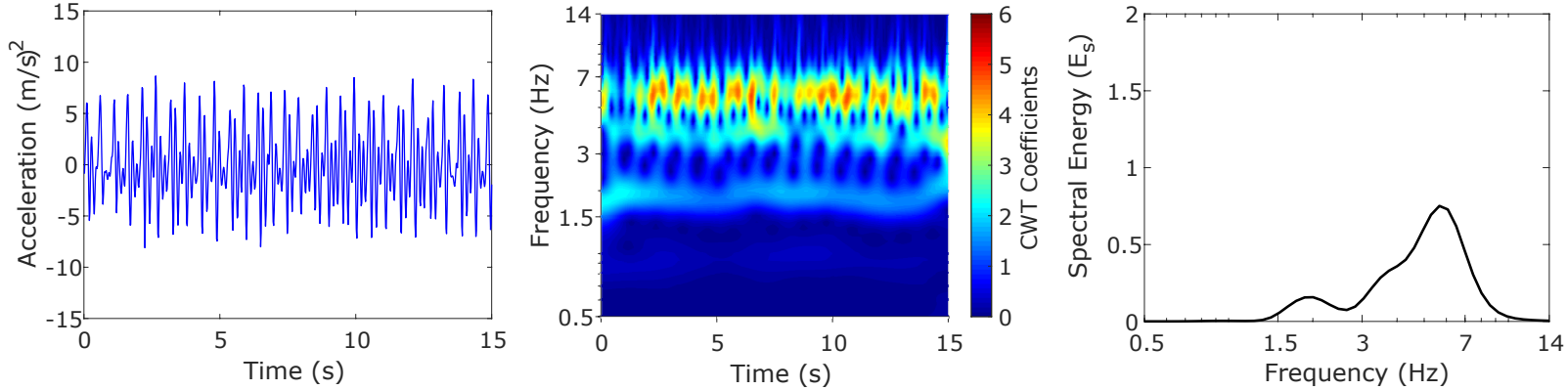
Creagh, A. P., *et al.* (2020). Smartphone-and smartwatch-based remote characterisation of ambulation in multiple sclerosis during the two-minute walk test. *IEEE Journal of Biomedical and Health Informatics*, *25*(3), 838-849. doi: 10.1109/JBHI.2020.2998187

CWT for gait analysis

Healthy Control



Multiple Sclerosis



Creagh, A. P., et al. (2020). Smartphone-and smartwatch-based remote characterisation of ambulation in multiple sclerosis during the two-minute walk test. *IEEE JBHI, 25*(3), 838-849.

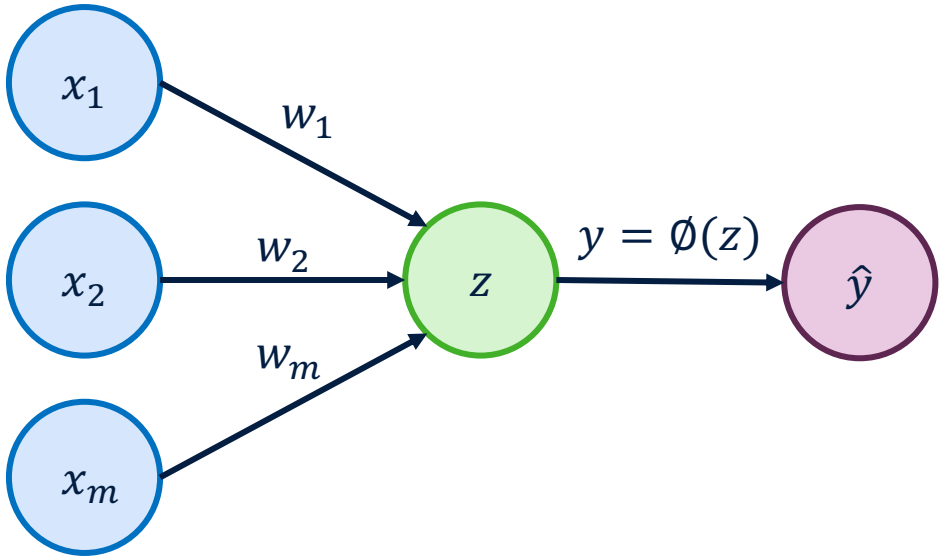
Representation Learning

Extracting unsupervised spatio-temporal features

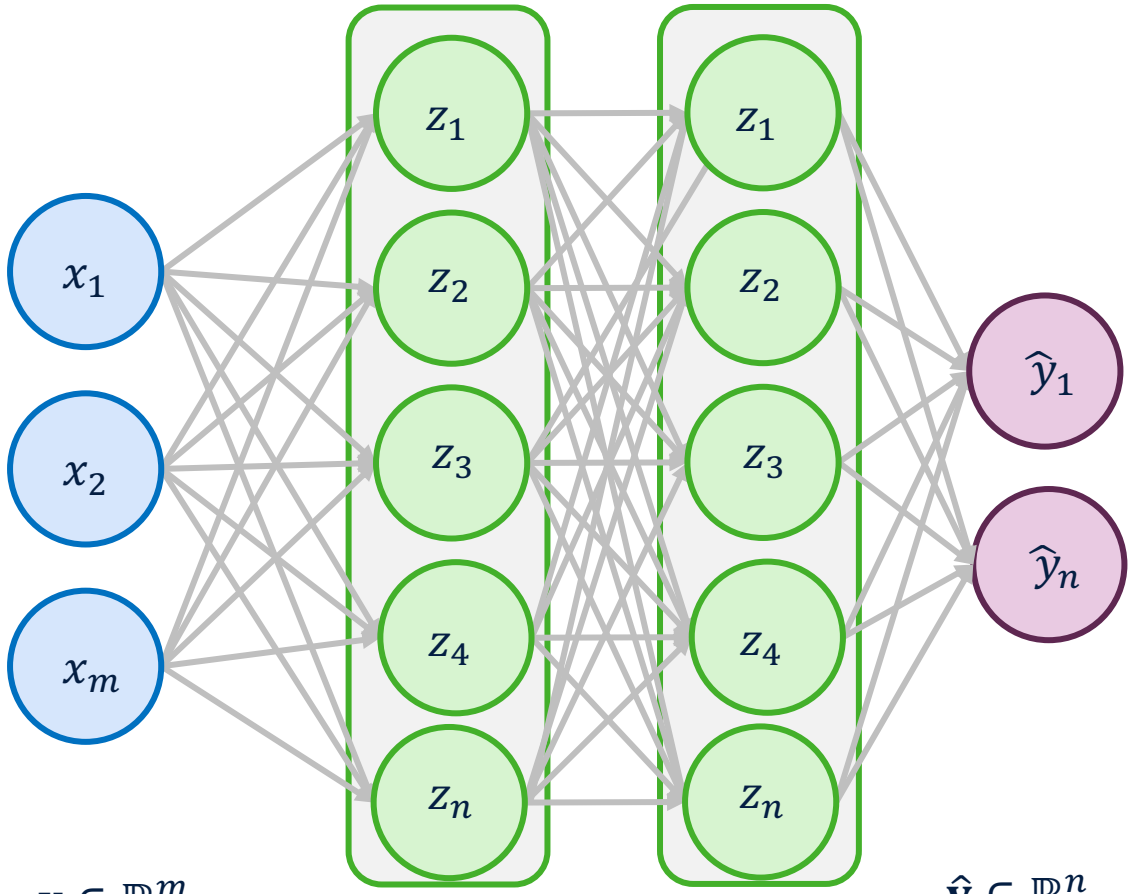
- Representation Learning
- Convolutional Neural Networks (CNN)
- Spectrograms as CNN inputs
- Raw time-series as CNN inputs
- Picking your kernels and network architectures for your task
- Incorporating temporality into your network architecture

Advancing towards deep networks

the perceptron

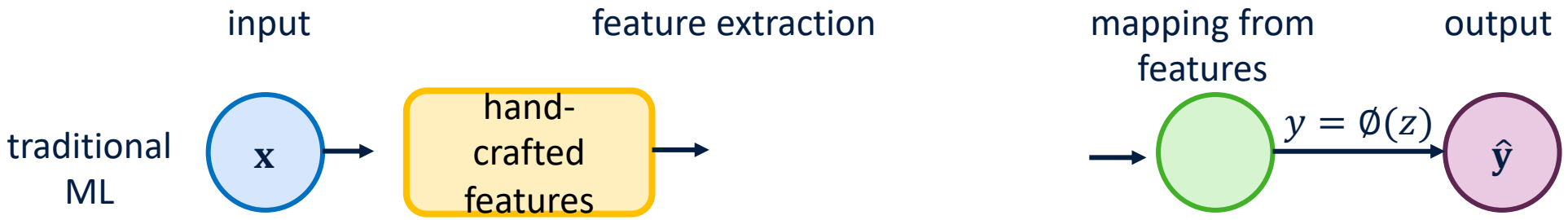


MLP, feed forward networks



Deep learning typically describes multi-layer perceptron (MLP) blocks that are stacked in cascading architectural arrangements, consisting of a number of (often non-linear) functions successively layered together, which map an input $\mathbf{x}$ to some output $\mathbf{y}$

Representation Learning



$\mathbf{x} \in \mathbb{R}^m$ $\quad z = \mathbf{w}^\intercal \mathbf{x}$ $\quad \hat{\mathbf{y}} \in \mathbb{R}^n$

Image credit: adapted from https://www.deeplearningbook.org/contents/intro.html

$\emptyset(\cdot)$: some function (often non-linear), e.g. ReLU

Representation Learning

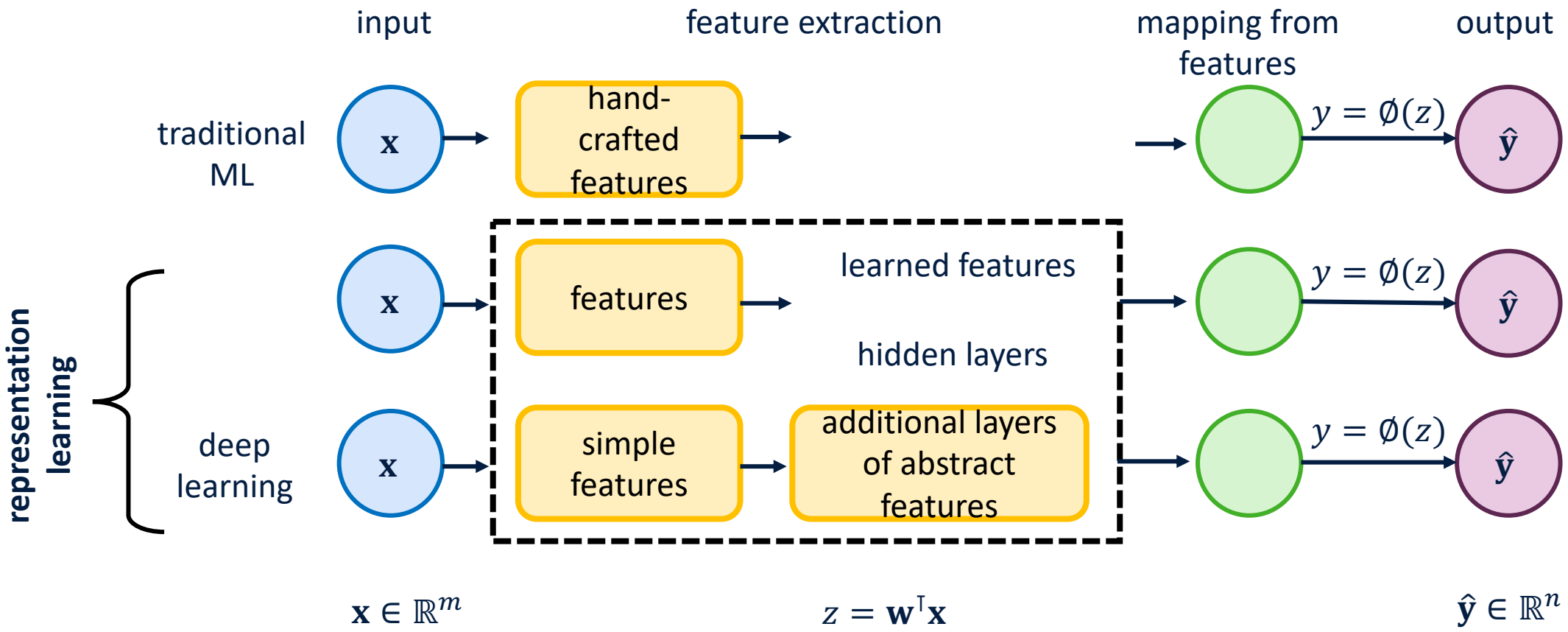


Image credit: adapted from https://www.deeplearningbook.org/contents/intro.html

$\emptyset(\cdot)$: some function (often non-linear), e.g. ReLU

Convolutional Neural Networks

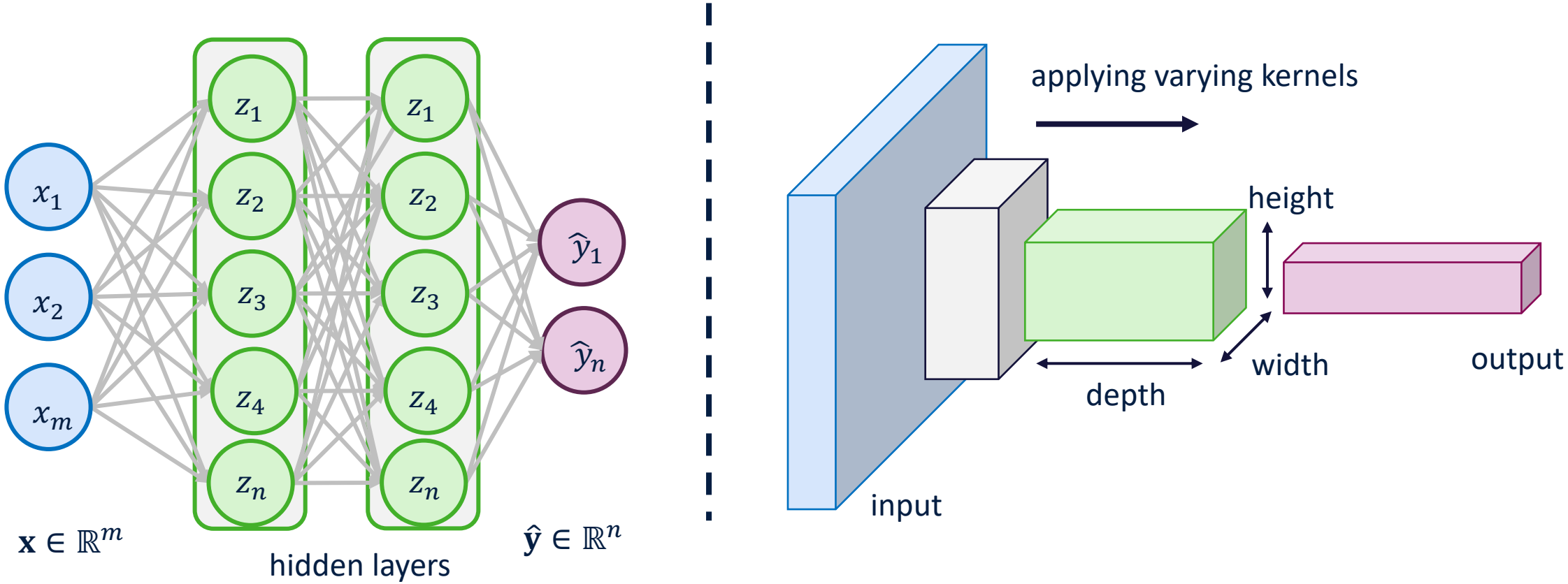
Convolutional Neural Networks (CNNs) are a class of ANN algorithm that follow a grid like topology and are especially robust displaying *translation invariance*. Translational invariance refers to a representation that is approximately invariant to small translations of the input. For instance, when presented with inputs of elements that are in a different order or have been translated, a model output will not vary.

The convolutional operation

$$s(t) = (x * w)(t) = \sum_{a=-L}^{L} x(a)w(t-a)$$

where x is the raw input data, such as a time-series, of length $2L$ and $w(\cdot)$ is the weighting function or kernel.
The output $s(t)$ at time index t is often considered as a representation of the features learned (a feature map).
The number of convolutional kernels applied to an input, as well as the width and height of $w(\cdot)$ are pre-defined.

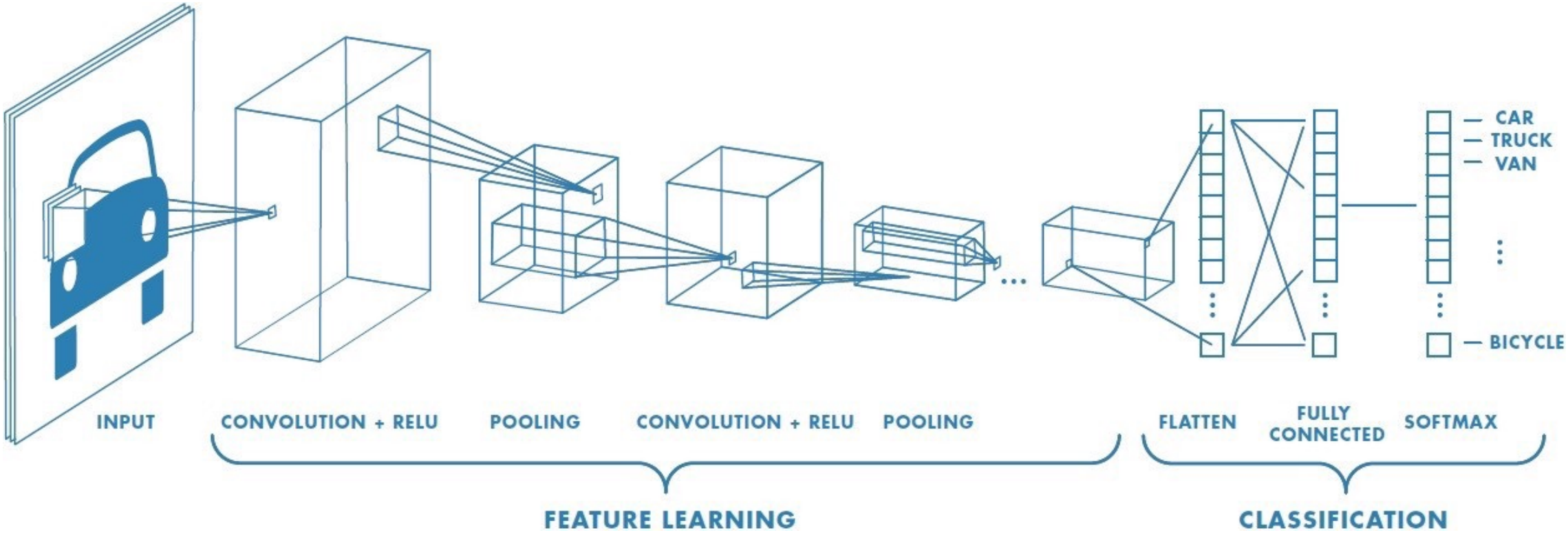
Convolutional Neural Networks can be applied to time-series!



A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)

Credit: https://cs231n.github.io/convolutional-networks/

CNNs learn features (NOT ONLY FOR IMAGES)



Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Filters as feature extractors

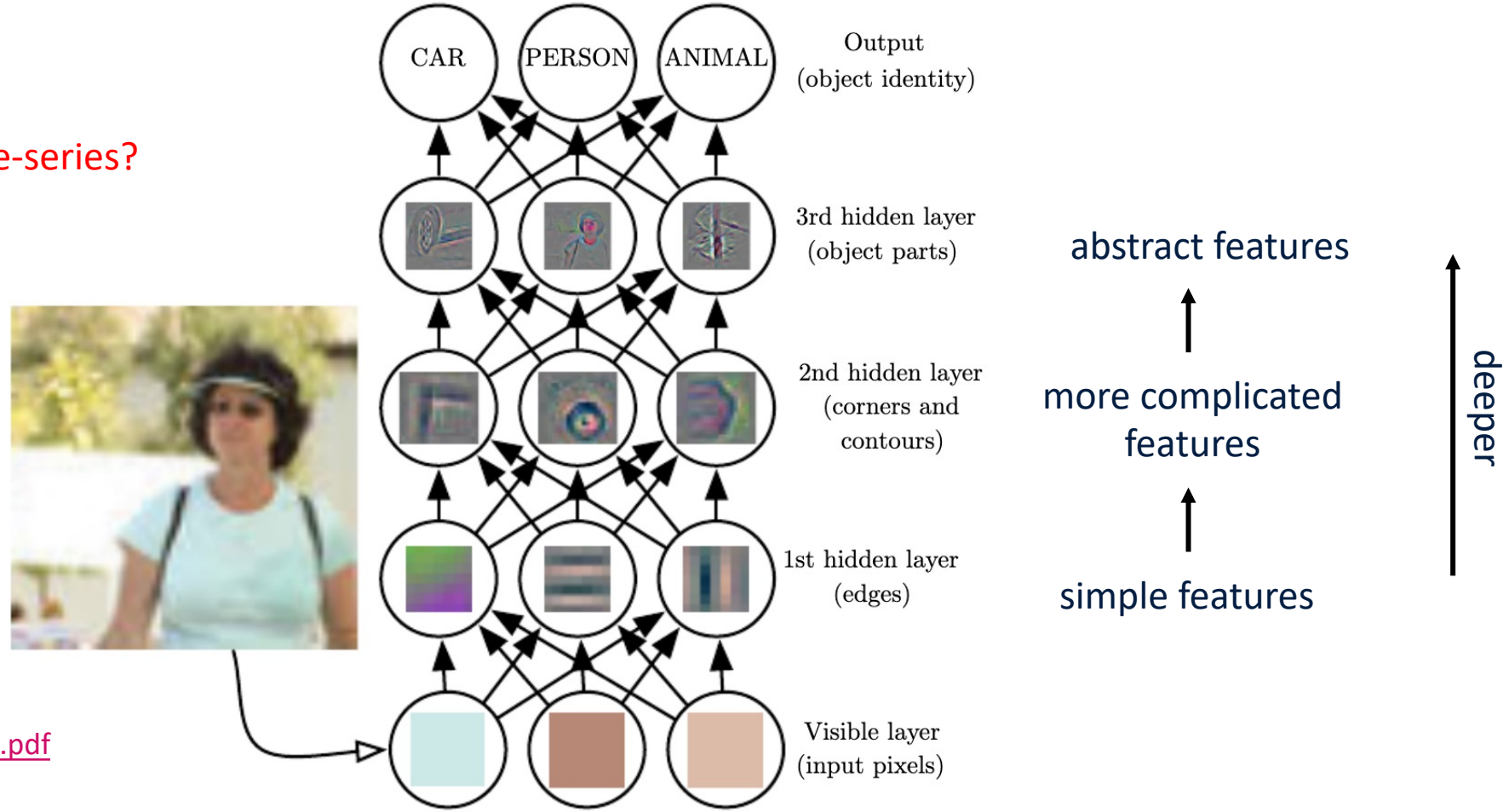
visualization over all the filters

Numerically computed images, illustrating the class appearance models, learnt by a ConvNet, trained on ILSVRC-2013. Note how different aspects of class appearance are captured in a single image.

Simonyan, Vedaldi and Zisserman (2014) https://arxiv.org/pdf/1312.6034.pdf (Oxford!)

Learning complicated representations

Q: Does this hold for time-series?



Zeiler & Fergus (2014)
https://arxiv.org/pdf/1311.2901.pdf

Deep learning breaks a complicated mapping into a series of nested simple mappings, each described by a different layer of the model. The input is presented at the visible layer, then a series of hidden layers extracts increasingly abstract features from the image. These layers are called “hidden” because their values are not given in the data; instead the model must determine which concepts are useful for explaining the relationships in the observed data. Adapted from https://www.deeplearningbook.org/contents/intro.html

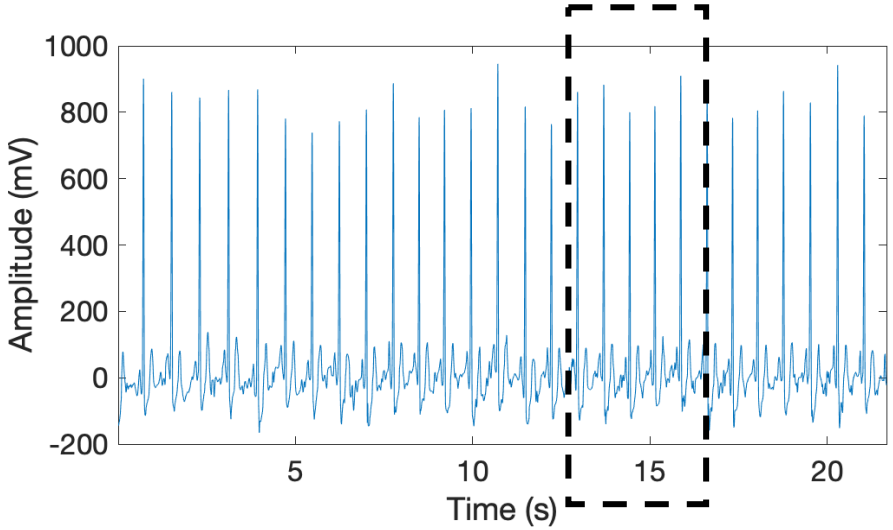
There's a meme for that

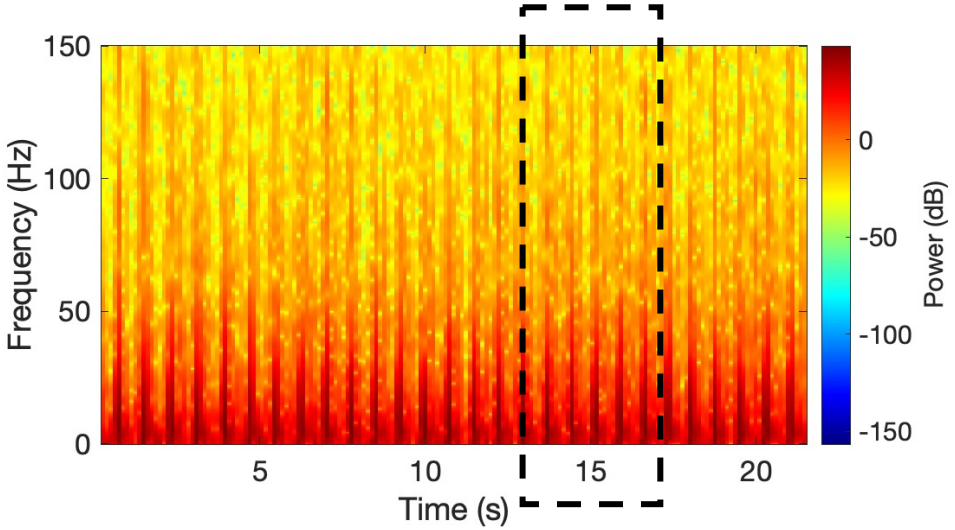
When you ask your NN
why it classified
your cat as a dog:

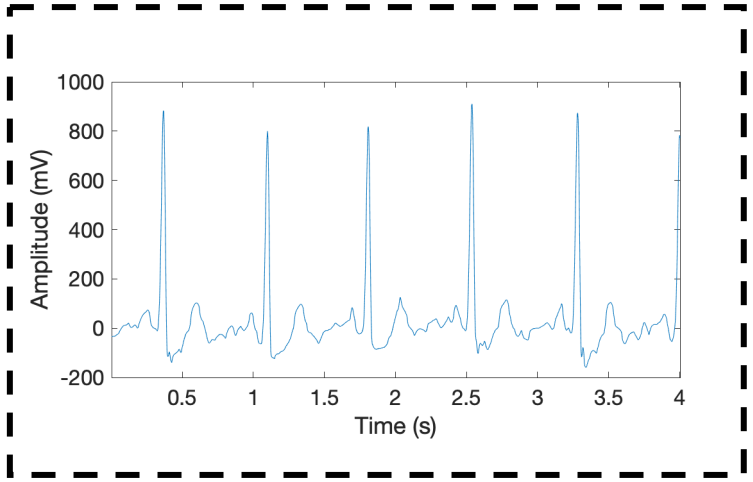
*I think this looks like a polar bear?

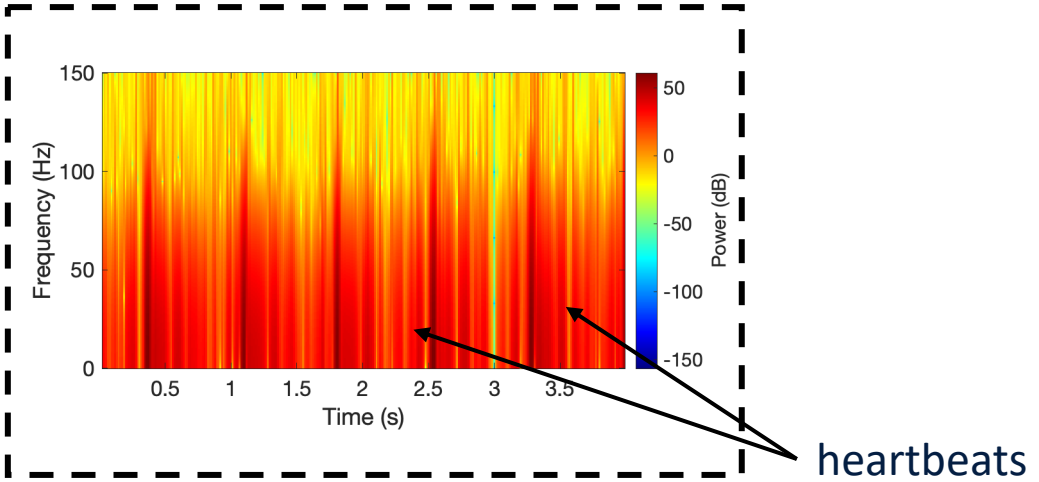
https://me.me/i/when-you-ask-your-nn-why-it-classified-your-cat-412f6f91a74e41d09b241909f432ac8f

Representing a time-series as images





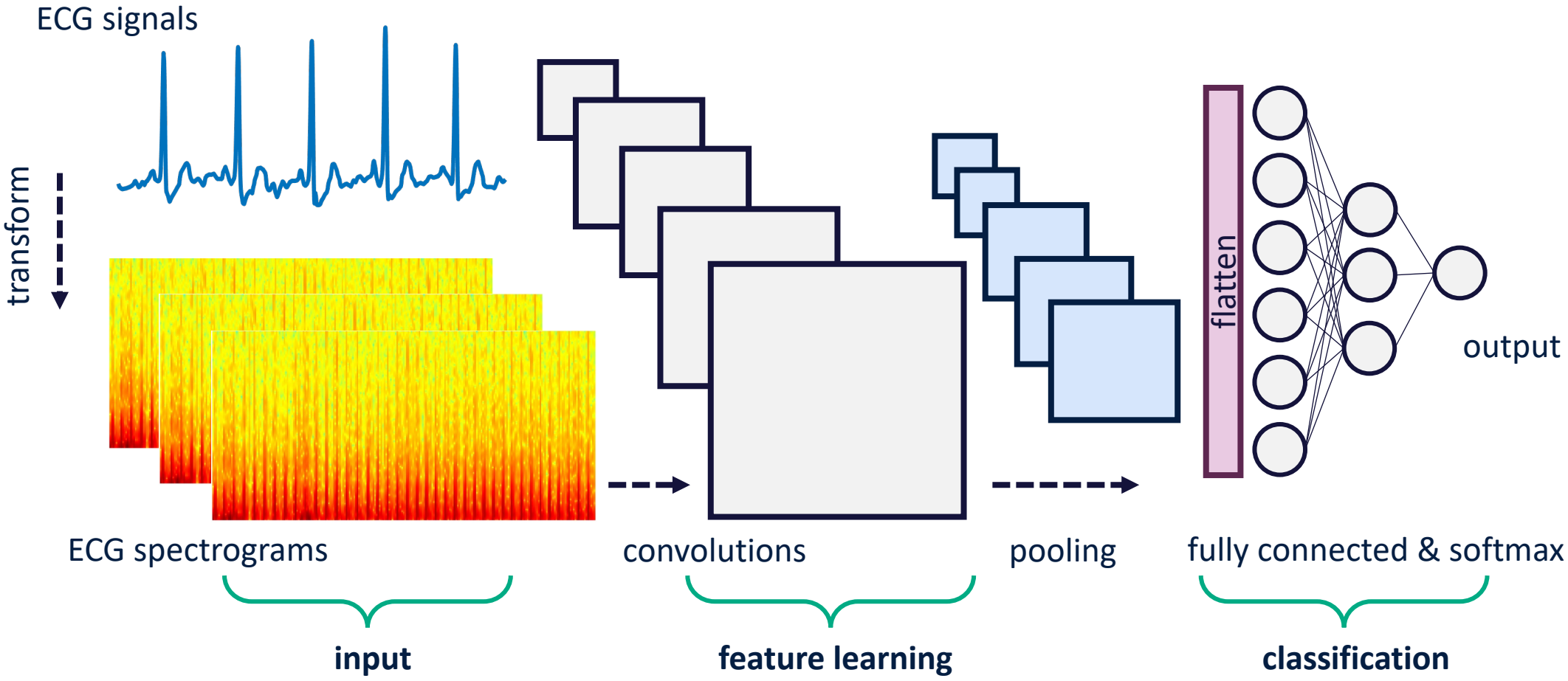




Raw ECG time series

Spectrogram of ECG

Spectrograms as 2-D CNN inputs

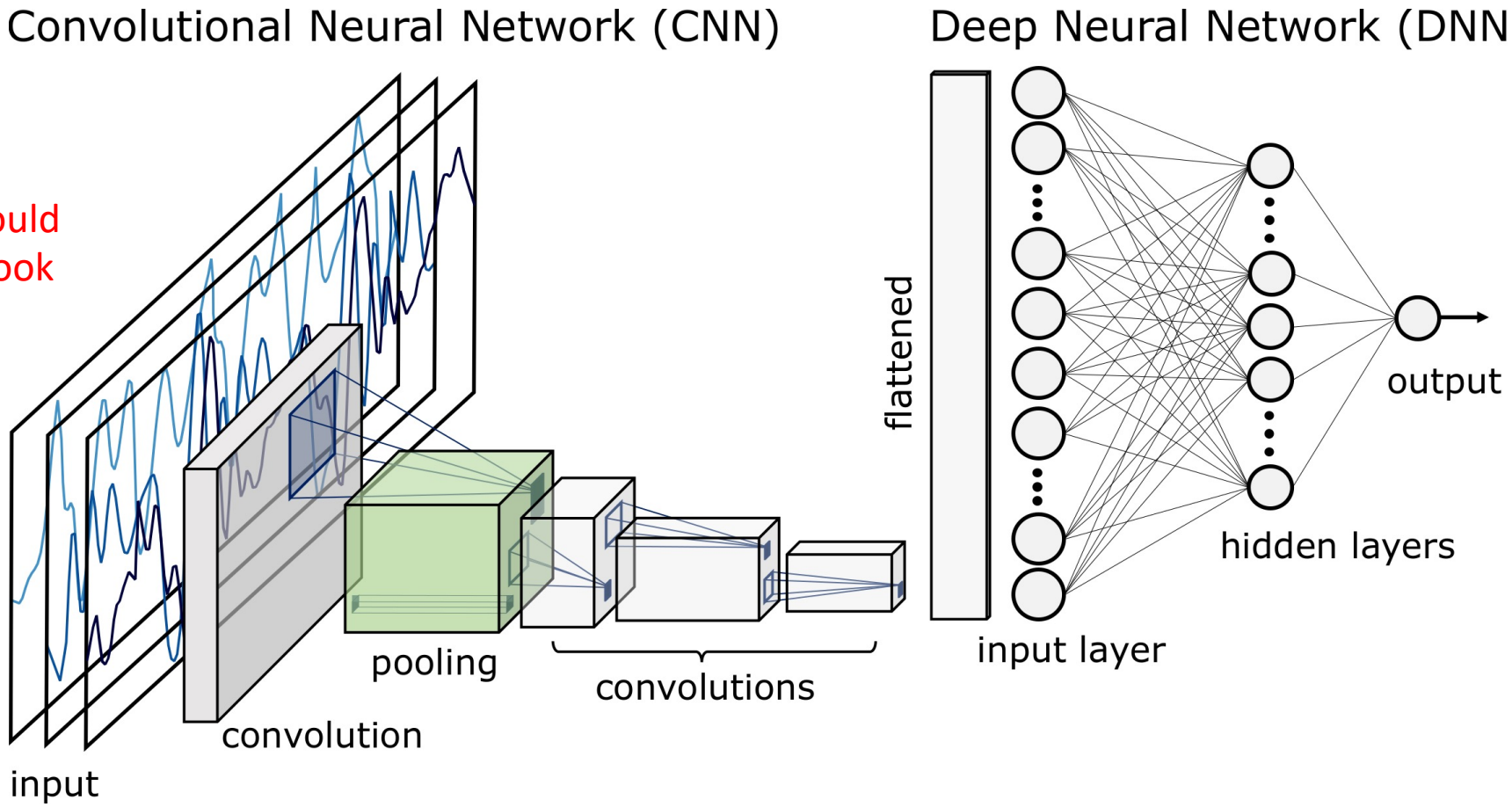


Typical CNN architecture for ECG spectrograms classification

See Andreotti, F. and Carr, O., et al, (2017). https://physionet.org/files/challenge-2017/1.0.0/papers/360-239.pdf

Raw time-series as 1-D CNN input

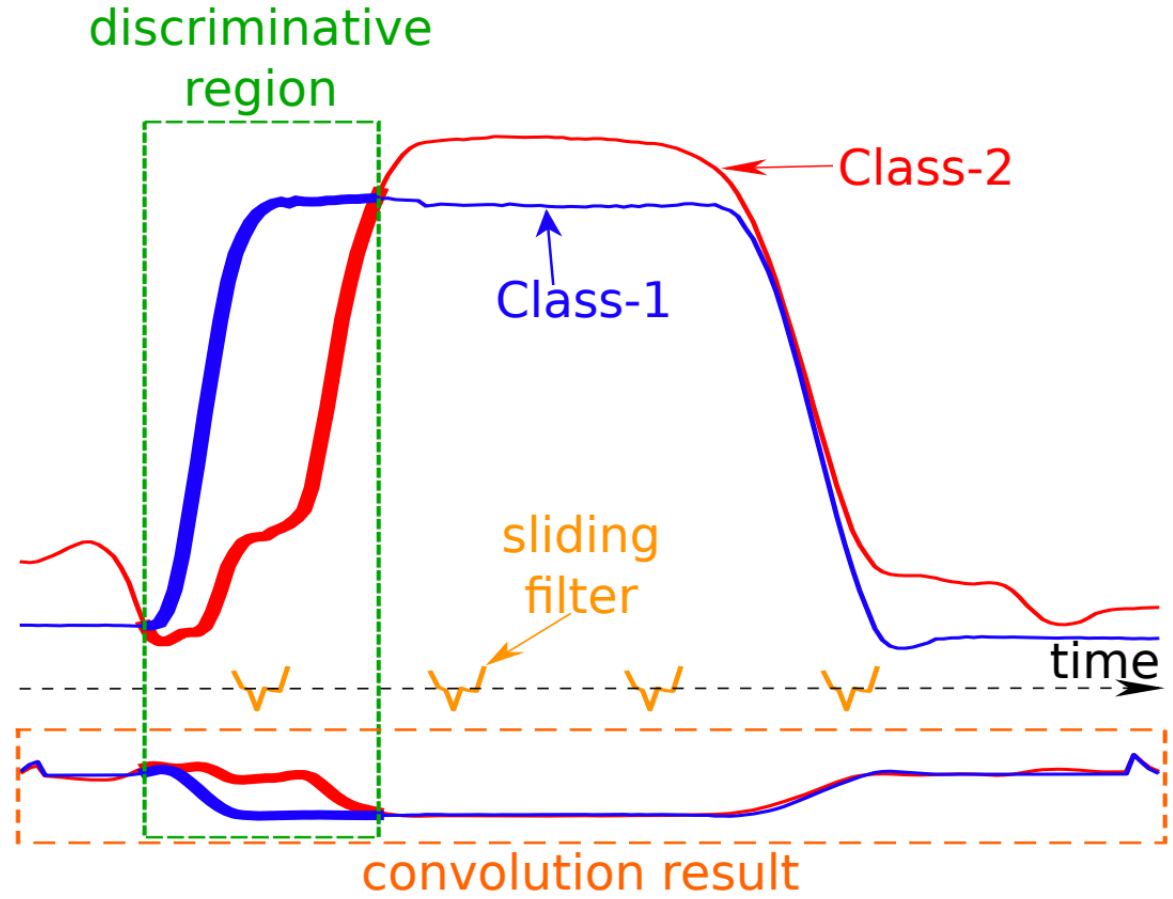
Q: What would this input look like?



Typical CNN architecture for time-series input classification

Creagh A.P., et al. (2020), *"The Development of Digital Biomarkers for Multiple Sclerosis from Remote Smartphone- and Smartwatch-Based Assessments".* D.Phil. Thesis

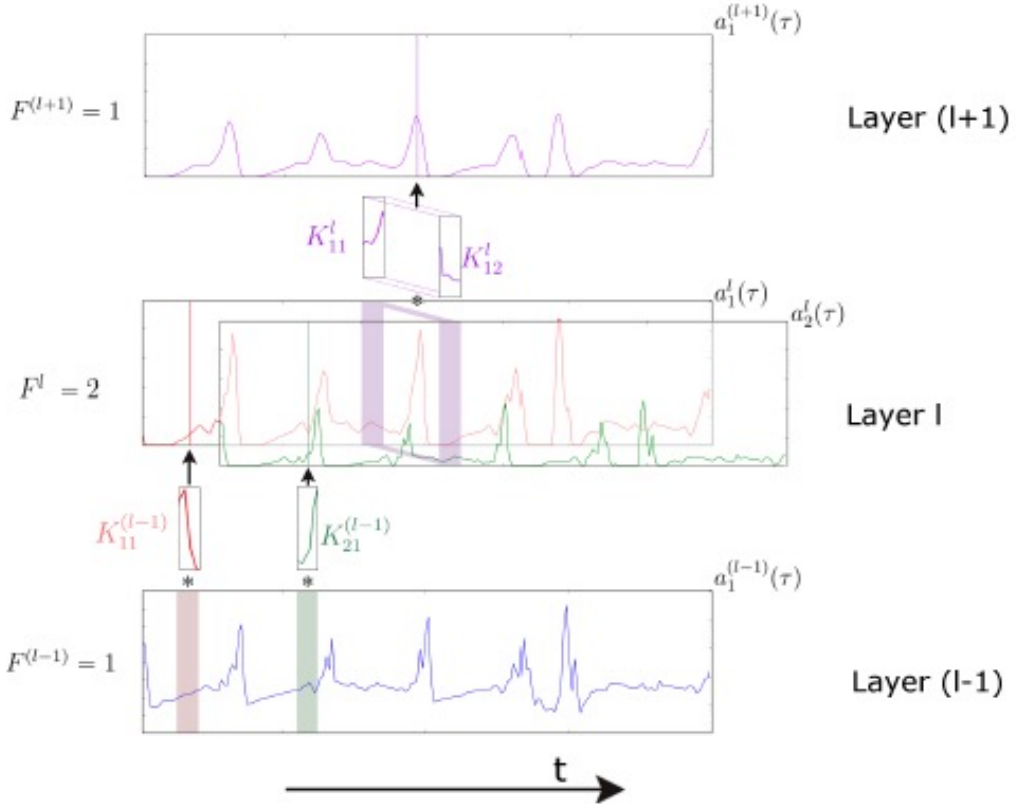
Applying CNN kernels to a time-series



The result of a applying a learned discriminative convolution on the GunPoint dataset

Fawaz et al (2019): https://arxiv.org/pdf/1809.04356.pdf

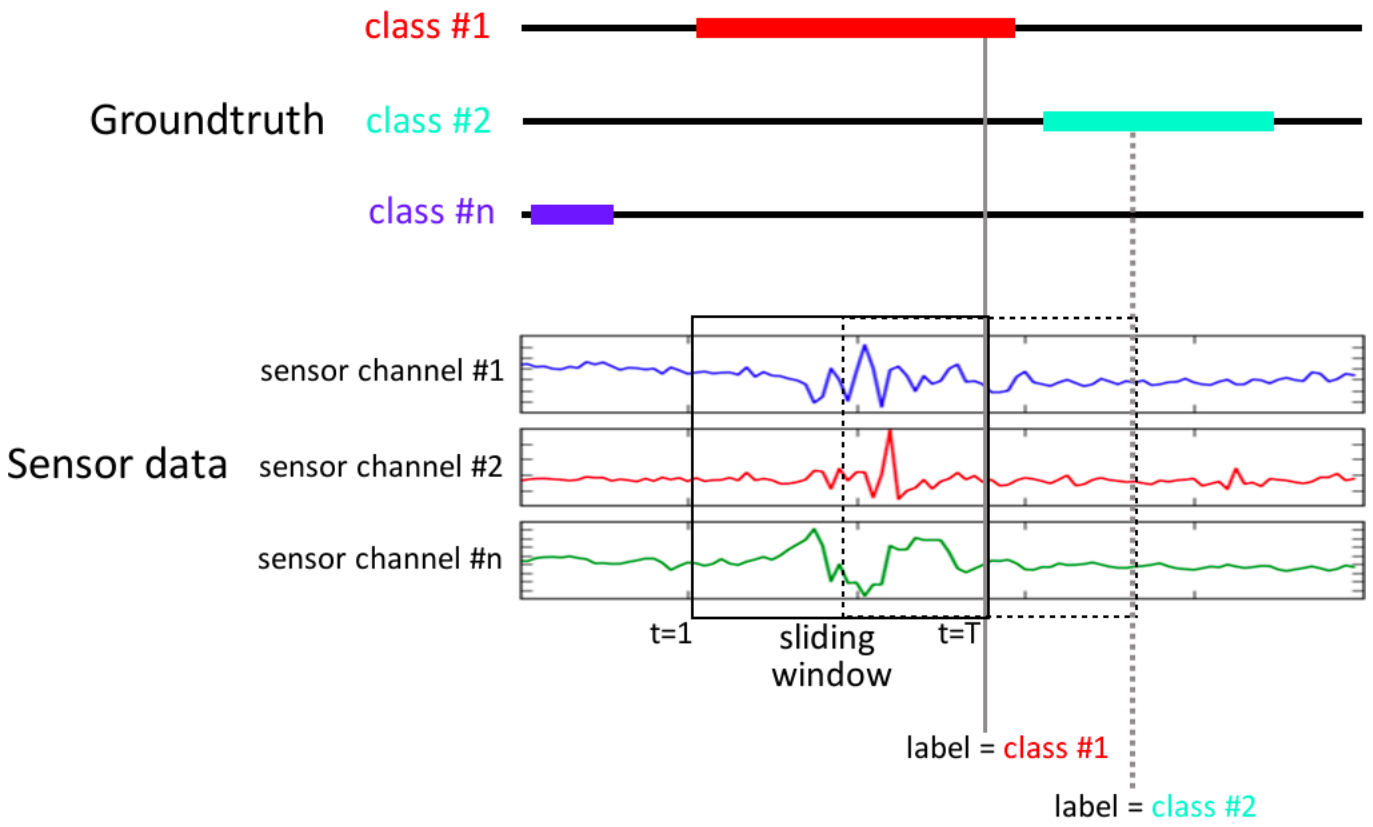
Applying CNN kernels to a time-series



Representation of a temporal convolution over a single sensor channel in a three-layer convolutional neural network (CNN). Layer $(l-1)$ defines the sensor data at the input. The next layer (l) is composed of two feature maps $\left(a_1^{(l)}(\tau) \text{ and } a_2^{(l)}(\tau)\right)$ extracted by two different kernels $\left(K_{11}^{(l-1)} \text{and } K_{21}^{(l-1)}\right)$. The deepest layer (layer $(l+1)$) is composed by a single feature map, resulting from temporal convolution in layer l of a two-dimensional kernel $K_1^{(l)}$.

Ordóñez, F.J. and Roggen, D., 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1), p.115.

Windowing (revisited)



Sequence labelling after segmenting the data with a sliding window. The sensor signals are segmented by a window. The activity class within each sequence is considered to be the ground truth label annotated at the sample T of that window.

Ordóñez, F.J. and Roggen, D., 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1), p.115.

Choosing your kernel

We can modify our kernel shapes and sizes to suit our task and signal

(FYI: kernel size can affect performance for time-series)
https://arxiv.org/pdf/2002.10061v1.pdf

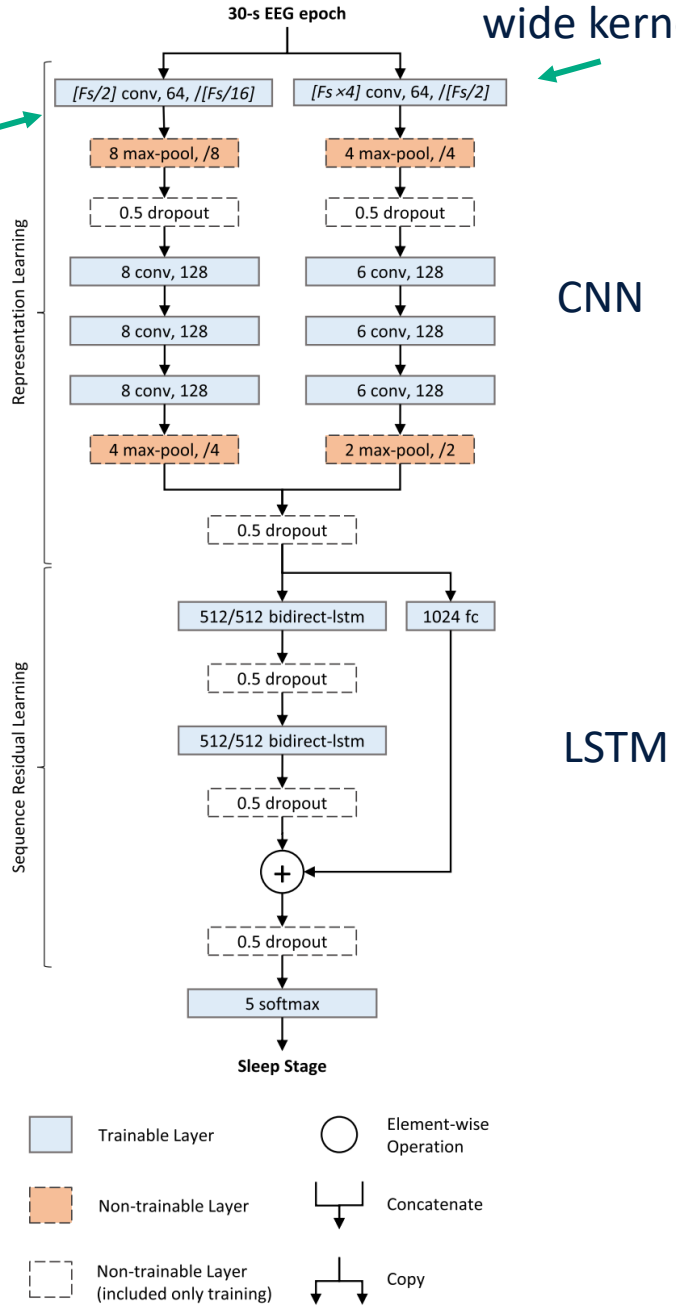
$K_{(wide)} = {}^{fs}/_{2}$ $K_{(narrow)} = {}^{fs}/_{16}$ fs: sampling frequency (Hz)

Choosing your kernel

We can modify our kernel shapes and sizes to suit our task and signal

(FYI: kernel size can affect performance for time-series)
https://arxiv.org/pdf/2002.10061v1.pdf

$K_{(wide)} = {}^{fs}/_{2}$ $K_{(narrow)} = {}^{fs}/_{16}$ fs: sampling frequency (Hz)



A. Supratak et al., "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, no. 11, pp. 1998–2008, Nov. 2017

Prince, J., Andreotti, F., & De Vos, M. (2018). Multi-source ensemble learning for the remote prediction of Parkinson's disease in the presence of source-wise missing data. *IEEE Transactions on Biomedical Engineering*, *66*(5), 1402-1411.

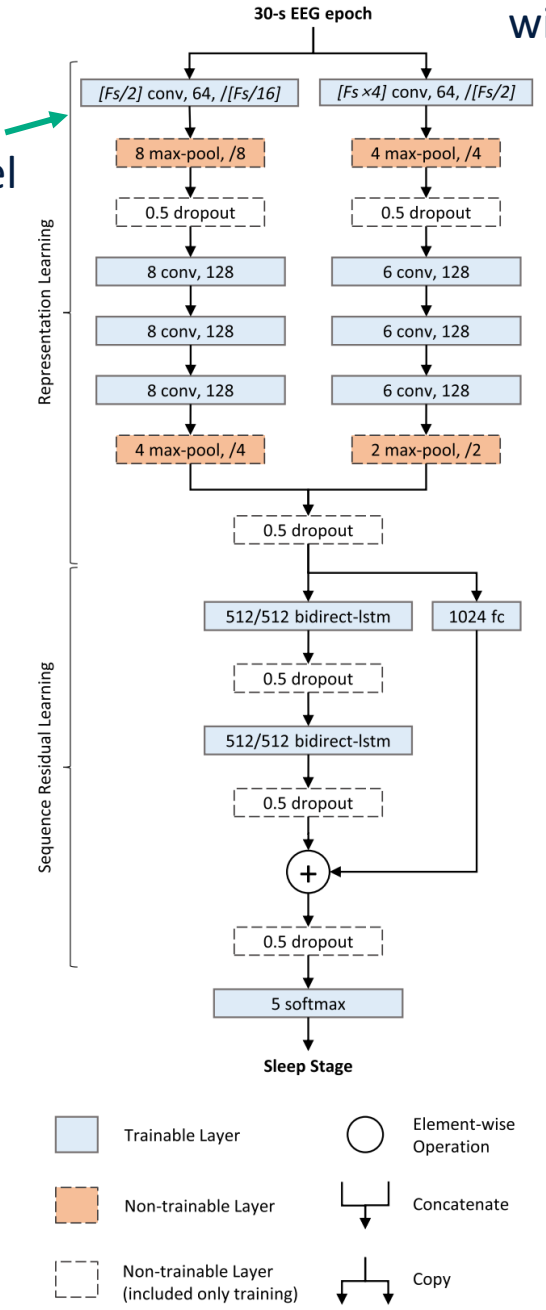
Choosing your kernel

We can modify our kernel shapes and sizes to suit our task and signal

(FYI: kernel size can affect performance for time-series)
https://arxiv.org/pdf/2002.10061v1.pdf

$K_{(wide)} = {}^{fs}/_{2}$ $K_{(narrow)} = {}^{fs}/_{16}$ fs: sampling frequency (Hz)

- *"When using convolutional filters of a large width, the frequency components of the data are better captured. Conversely, using filters of a small width better capture temporal aspects of the signal."*
- Alternative CNN architectures use small receptive fields but require many more layers and convolutional operations in order to capture the frequency components of the data.
- CNNs were trained to learn filters to extract time-invariant features from raw single channel EEG, while the bidirectional-LSTMs were trained to encode temporal information such as sleep stage transition rules into the model.



A. Supratak et al., "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, no. 11, pp. 1998–2008, Nov. 2017

Prince, J., Andreotti, F., & De Vos, M. (2018). Multi-source ensemble learning for the remote prediction of Parkinson's disease in the presence of source-wise missing data. *IEEE Transactions on Biomedical Engineering*, *66*(5), 1402-1411.

What framework to use?*

For some good comparison information:
https://www.projectpro.io/article/pytorch-vs-tensorflow-2021-a-head-to-head-comparison/416
https://medium.com/analytics-vidhya/ml03-9de2f0dbd62d

*we're using PyTorch in this workshop

CNN code example

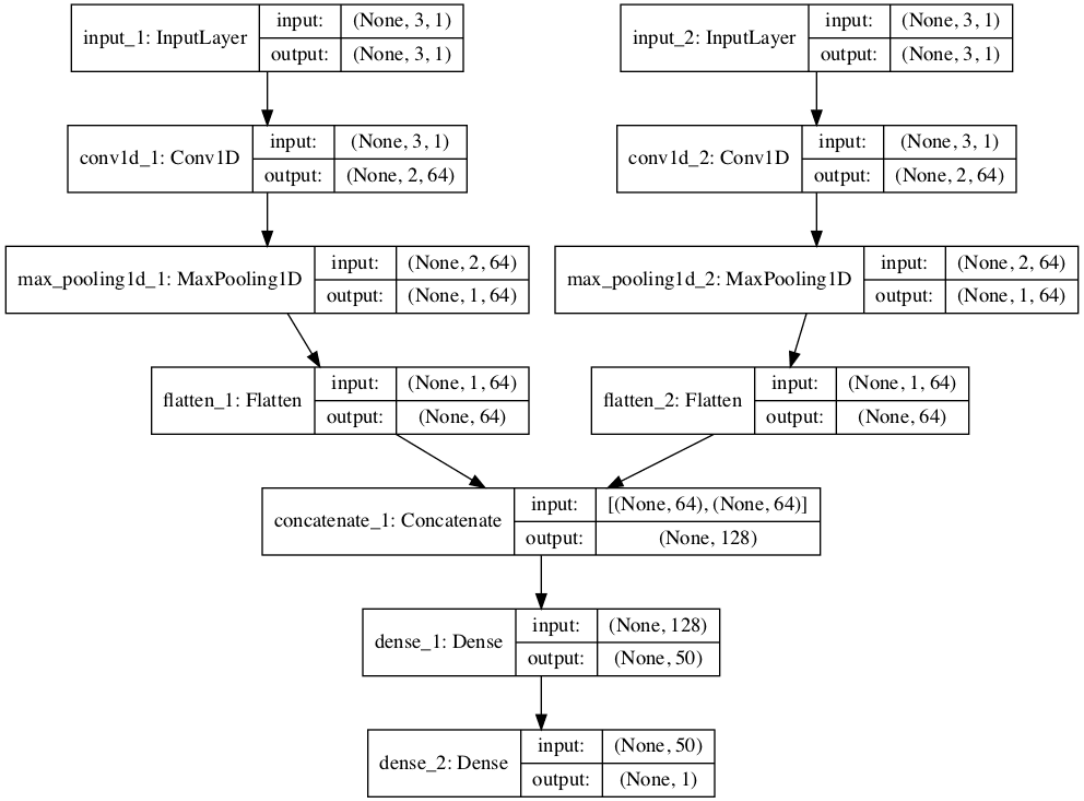
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.backends.cudnn as cudnn

class ConvBNReLU(nn.Module):
      ''' Convolution -> batch normalization -> ReLU'''
      def __init__(self, in_channels, out_channels, kernel_size=3, stride=1, padding=1, bias=True):
      #in_channels: Number of input channels / features
      #out_channels: Number of output channels / features
      #kernel_size: Size of the convolving kernel
      super(ConvBNReLU, self).__init__()
            self.main = nn.Sequential(nn.Conv1d(in_channels, out_channels, kernel_size, stride, padding, bias=bias),
            nn.BatchNorm1d(out_channels),nn.ReLU(True))
      def forward(self, x):
            return self.main(x)

class CNN(nn.Module):
      '''Pyramid CNN like VGG-Net'''
      def __init__(self, output_size=6, in_channels=3, num_filters_init=8):
            super(CNN_v3, self).__init__()
            self.cnn = nn.Sequential(
                        ConvBNReLU(in_channels, num_filters_init,kernel_size=8, stride=4, padding=2, bias=False),
                        ConvBNReLU(num_filters_init, num_filters_init*2,kernel_size=4, stride=4, padding=2, bias=False),
                        ConvBNReLU(num_filters_init*2, num_filters_init*4,kernel_size=4, stride=2, padding=2, bias=False),
                        ConvBNReLU(num_filters_init*4, num_filters_init*8,kernel_size=4, stride=2, padding=1, bias=False),
                        ConvBNReLU(num_filters_init*8, num_filters_init*16,kernel_size=2, stride=2, padding=1, bias=False),
                        ConvBNReLU(num_filters_init*16, num_filters_init*32,kernel_size=2, stride=2, padding=1, bias=False),
                        ConvBNReLU(num_filters_init*32, num_filters_init*64,kernel_size=2, stride=1, padding=0, bias=False))
            self.fc = nn.Linear(num_filters_init*64*output_size, output_size, bias=True)
      def forward(self, x):
            x=self.cnn(x).view(x.shape[0],-1)
            out=self.fc(x)
            return out
```

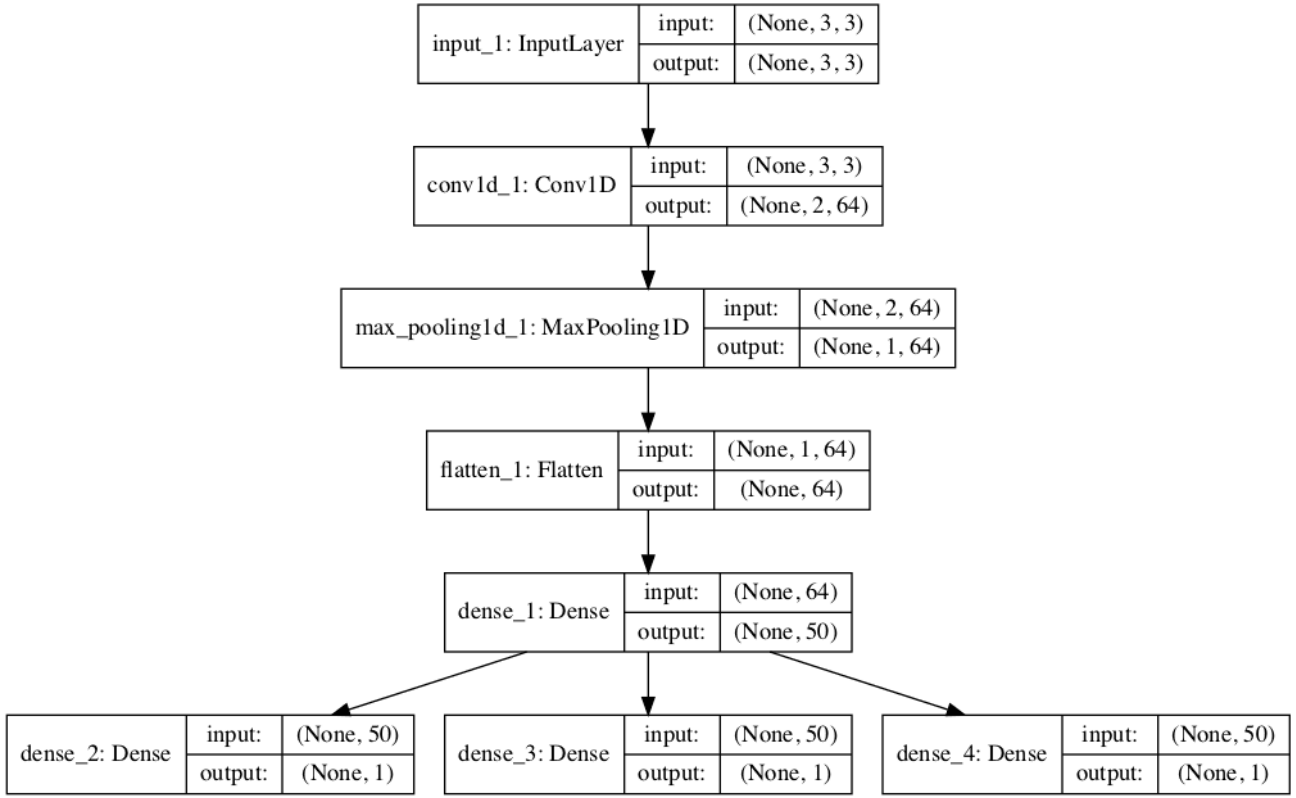
CNN architectures for time-series analysis

A separate CNN for each time series channel / feature



single output

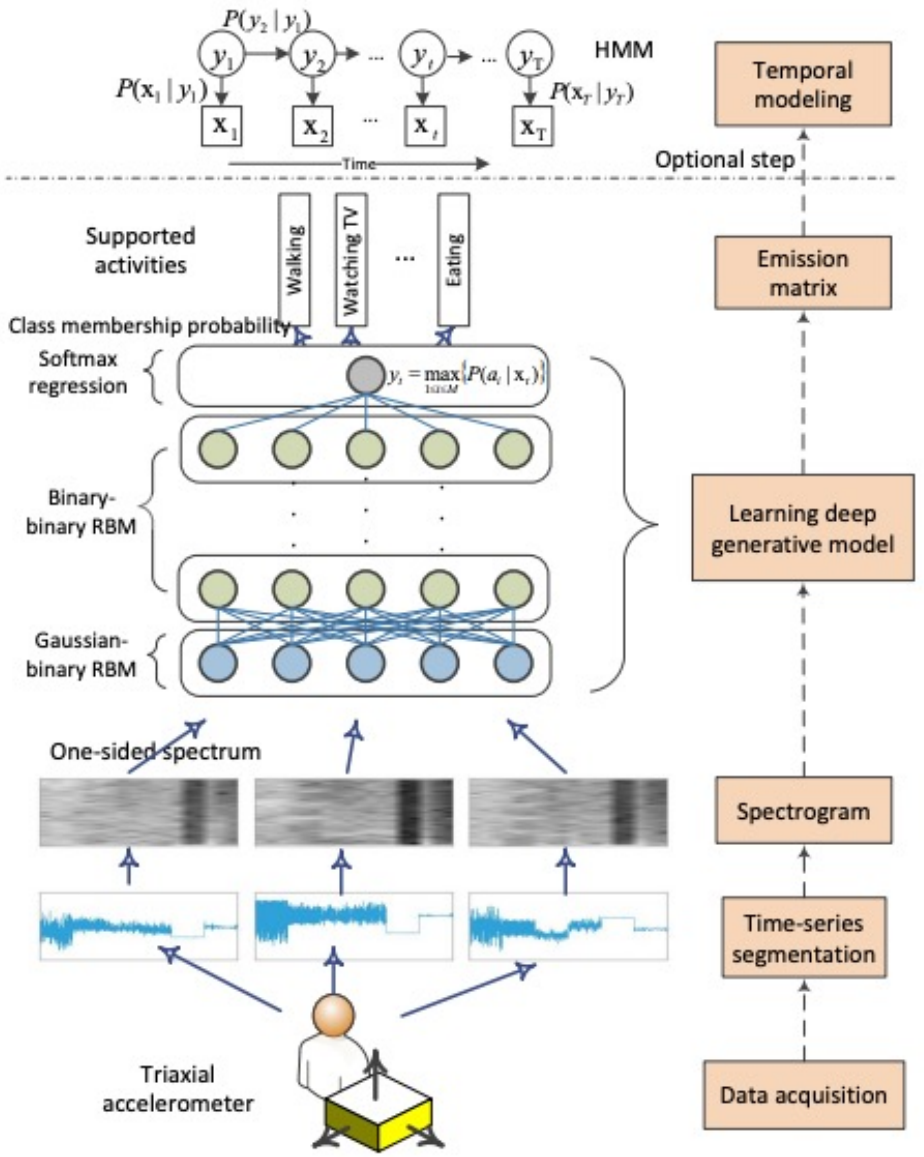
A single CNN combining each time series channel / feature



multiple outputs: e.g. an output for each time series channel

Incorporating temporal models to your CNN

activity recognition



Alsheikh, M. A., Selim, A., Niyato, D., Doyle, L., Lin, S., & Tan, H. P. (2016). Deep activity recognition models with triaxial accelerometers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Willetts, M., Hollowell, S., Aslett, L., Holmes, C., & Doherty, A. (2018). Statistical machine learning of sleep and physical activity phenotypes from sensor data in 96,220 UK Biobank participants. *Scientific reports*, *8*(1), 1-10.

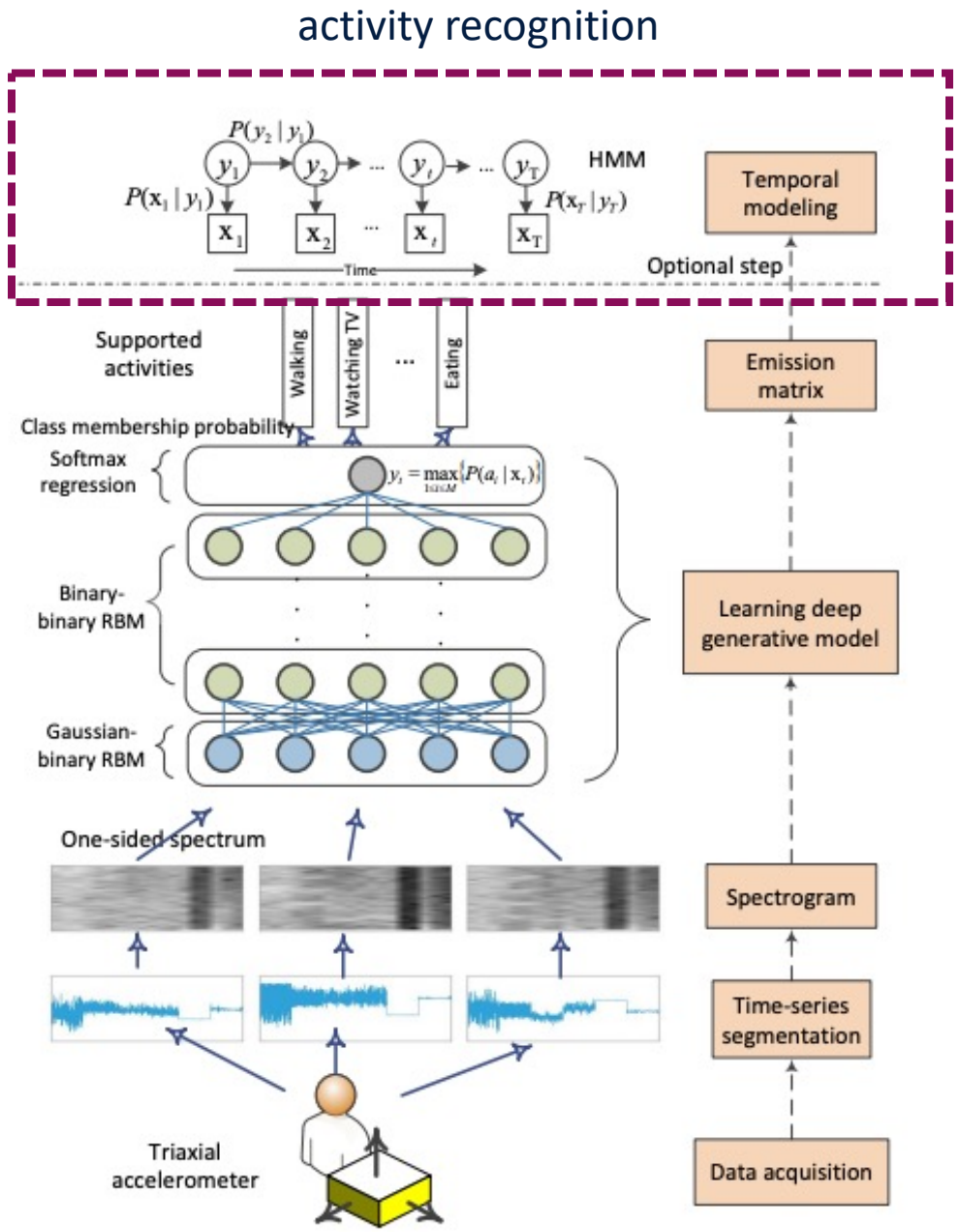
https://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/download/12627/12337

Incorporating temporal models to your CNN

Recap:
Hidden Markov Models (HMM)

$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

- The observed (or `visible') variables are dependent on the hidden variables, such that the present state influences the future state.
- The HMM adds temporal dependency in a sequence, for example, preventing predictions of *running* between two instances of *sleeping* in the middle of the night, because that is highly improbable.



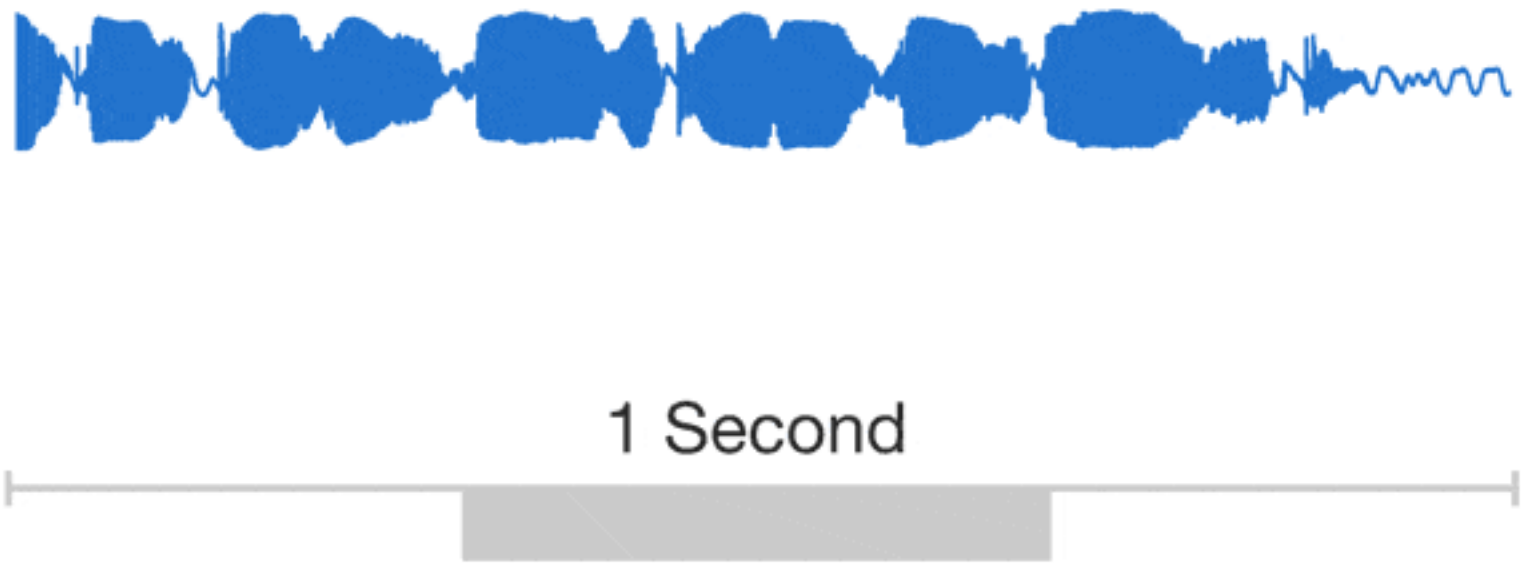
Alsheikh, M. A., Selim, A., Niyato, D., Doyle, L., Lin, S., & Tan, H. P. (2016). Deep activity recognition models with triaxial accelerometers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Willetts, M., Hollowell, S., Aslett, L., Holmes, C., & Doherty, A. (2018). Statistical machine learning of sleep and physical activity phenotypes from sensor data in 96,220 UK Biobank participants. *Scientific reports*, *8*(1), 1-10.

https://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/download/12627/12337

Case Study: Temporal CNN (TCNN)

Time-series signals, such as speech, contain important structures at many time-scales. Building a completely autoregressive model, in which the prediction for every one of those samples is influenced by all previous ones on a raw time-series is very challenging.



A second of generated speech

https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

van den Oord (2016) https://arxiv.org/pdf/1609.03499v2.pdf

Case Study: Temporal CNN (TCNN)

Time-series signals, such as speech, contain important structures at many time-scales. Building a completely autoregressive model, in which the prediction for every one of those samples is influenced by all previous ones on a raw time-series is very challenging.

Recap: Autoregressive Models

The joint probability of a waveform $\mathbf{x} = \{x_1, x_2, \cdots, x_T\}$ is factorised as a product of conditional probabilities as follows:

$$p(\mathbf{x}) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1}, \cdots, x_{t-L}) \qquad x_i = \emptyset \text{ for } i \leq 0$$

with

$$p(x_t | x_{t-1}, \cdots x_{t-L}) = \mathrm{N}\left(x_t \middle| \sum_{l=1}^{L} a_l x_{t-l}, \sigma^2 \right)$$

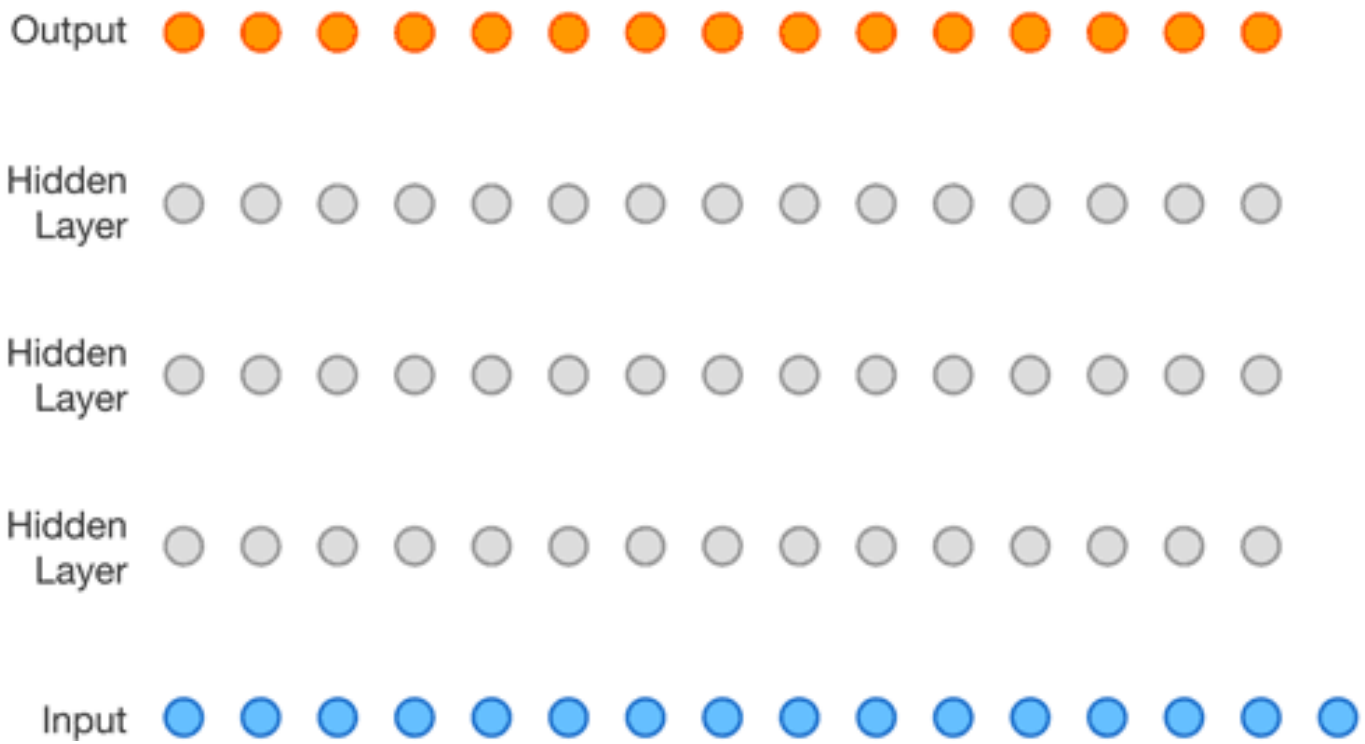
Each audio sample x_t is therefore conditioned on the samples at all previous timesteps

https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

van den Oord (2016) https://arxiv.org/pdf/1609.03499v2.pdf

*model cannot see future datapoints

Dilated Causal Convolutions



causality* → capture temporal information

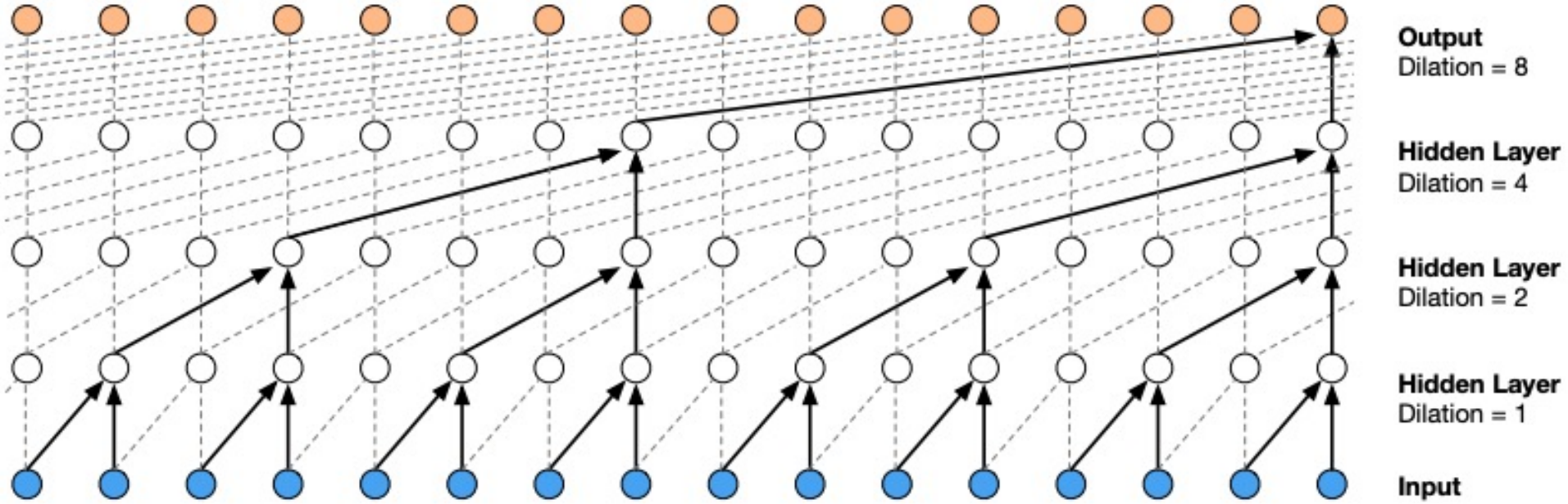
A Dilated Causal Convolution is a causal convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. A dilated causal convolution effectively allows the network to have very large receptive fields with just a few layers.

https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

van den Oord (2016) https://arxiv.org/pdf/1609.03499v2.pdf

*model cannot see future datapoints

Dilated Causal Convolutions



causality* → capture temporal information

A Dilated Causal Convolution is a causal convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. A dilated causal convolution effectively allows the network to have very large receptive fields with just a few layers.

https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

van den Oord (2016) https://arxiv.org/pdf/1609.03499v2.pdf

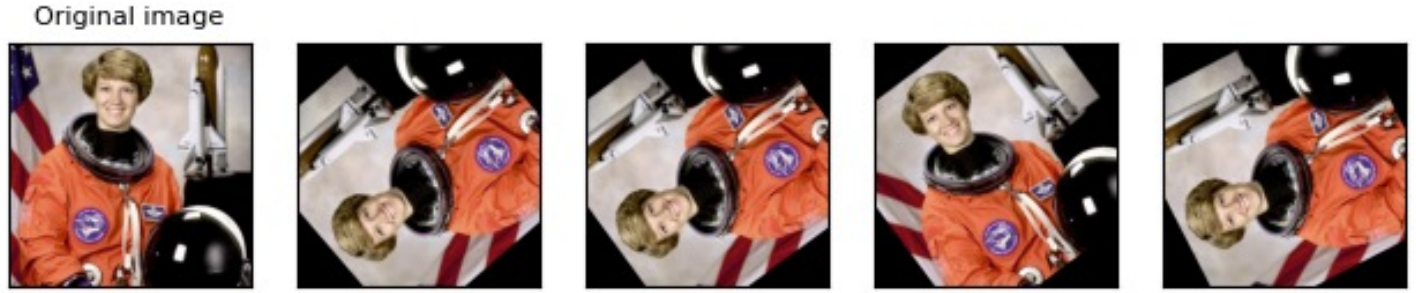
Developing robust models

Some tricks and tips for getting the most out of your deep network

- Data augmentation for time-series
- Transfer learning
- Self-supervised learning

*augment the data during training

Data Augmentation: Images



rotation

cropping

Gaussian blur

Credit: https://pytorch.org/vision/stable/auto_examples/plot_transforms.html#sphx-glr-auto-examples-plot-transforms-py

*augment the data during training

Data Augmentation: Time-Series

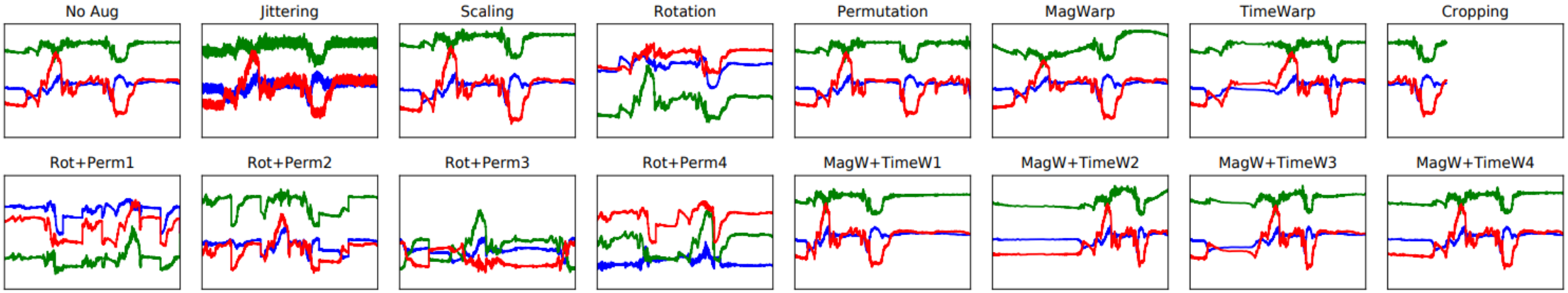


Figure 2: Various data augmentations that are used in the experiments: jittering, scaling, rotating, permutating, magnitude-warping, time-warping methods. Combinations of various data augmentations can also be applied.

T. T. Um et al., "Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks," in Proceedings of the 19th ACM International Conference on Multimodal Interaction, ser. ICMI 2017. New York, NY, USA: ACM, 2017, pp. 216–220. https://arxiv.org/pdf/1706.00527.pdf

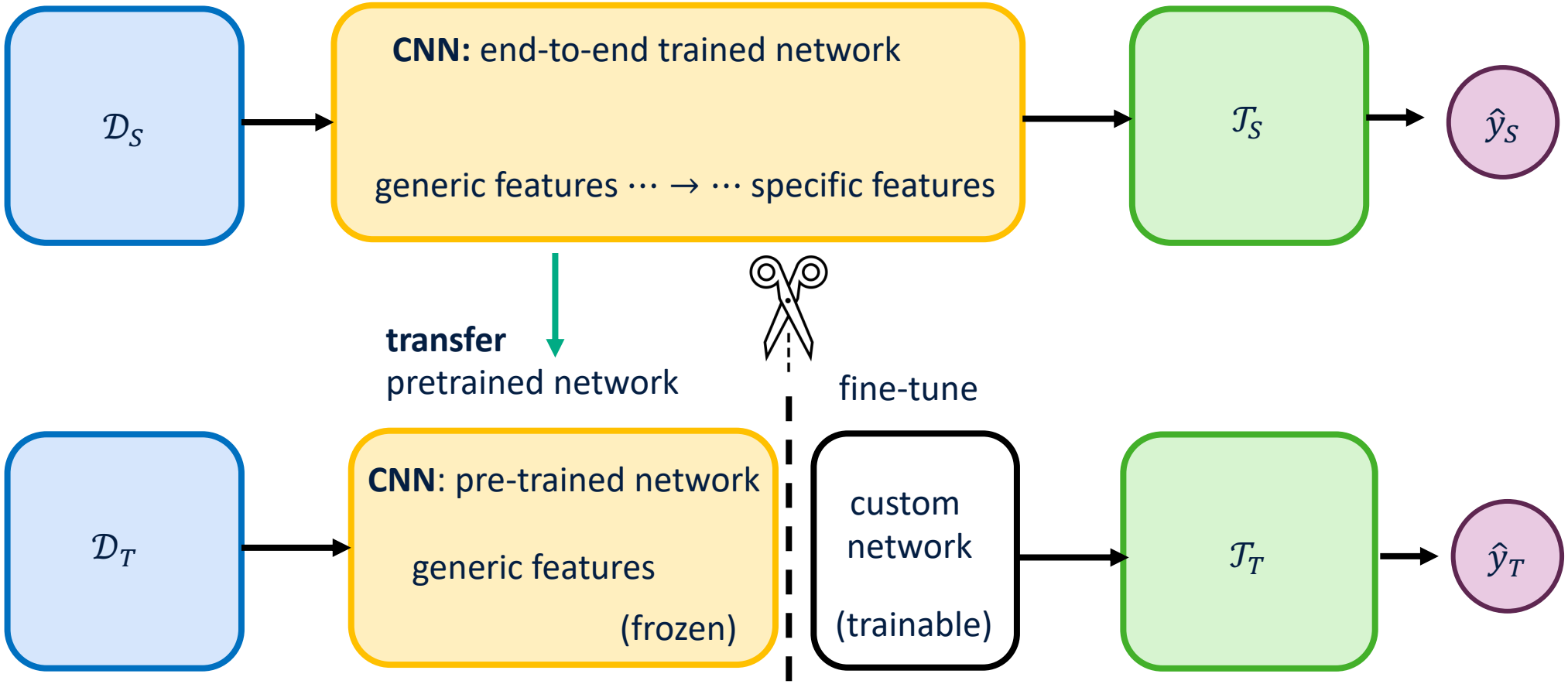
Code: https://github.com/terryum/Data-Augmentation-For-Wearable-Sensor-Data/

Transfer learning

*We often use big pre-trained networks like ResNet or Inception

Source domain: large, often generic dataset, e.g. UK Biobank

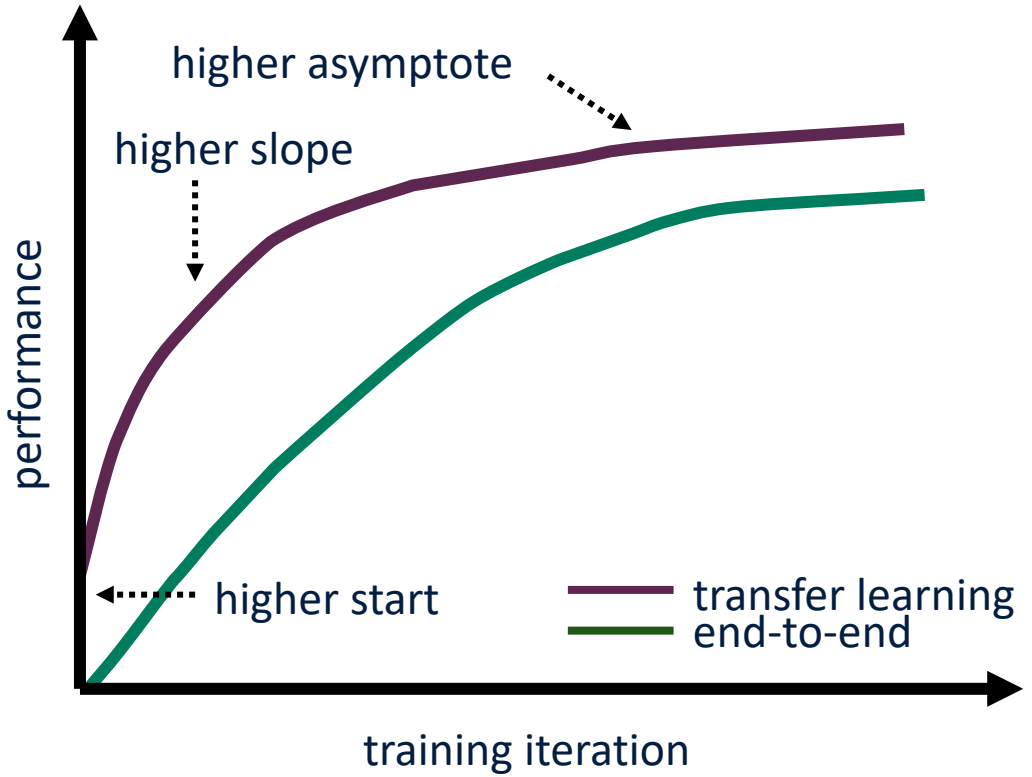
Source task: often a generic task or related task



Target domain: specific, often smaller dataset, e.g. your clinical trial

Target task: often a different but related task, e.g. disease classification

Transfer learning

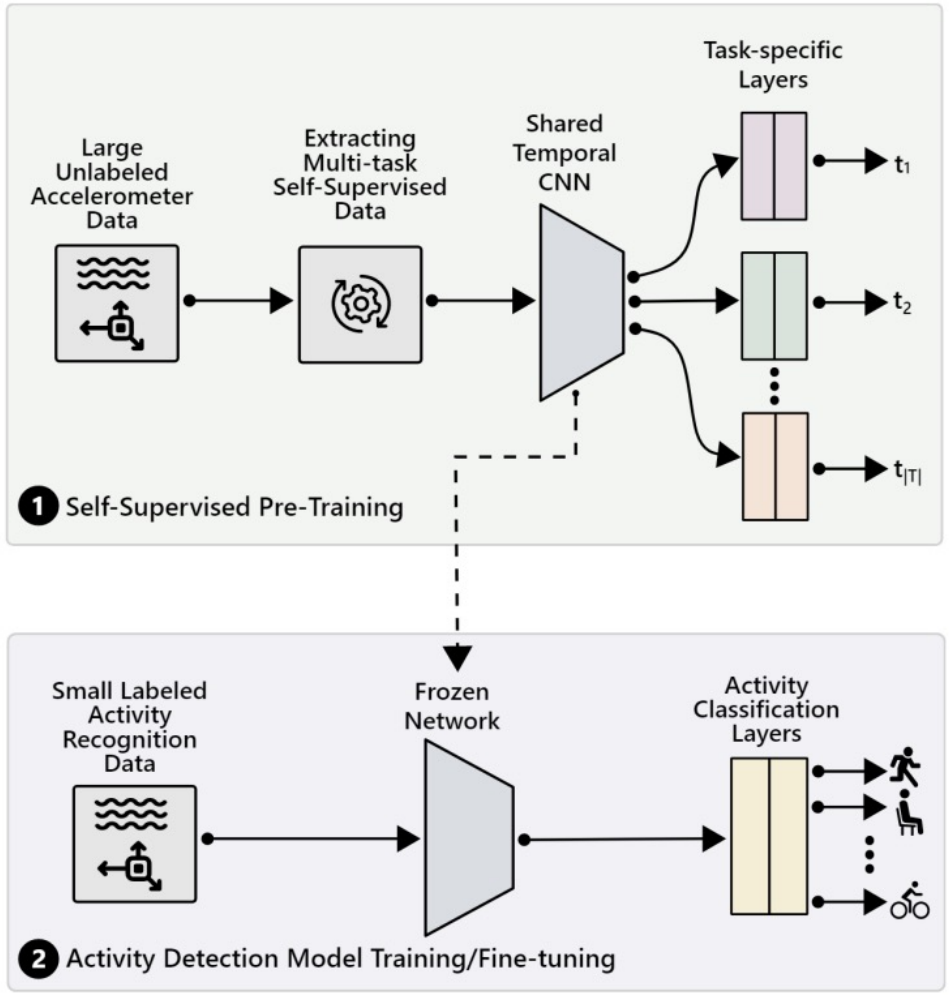


Transfer learning can improve model performance by first learning generalized features on larger and more diverse data, and then fine-tuning towards your target (specific task). It also helps mitigate against overfitting which can be problematic when training a model end-to-end on your smaller dataset.

TL;DR

Transfer learning be like

Self-Supervised Learning*; the next frontier



*What happens if your large dataset is unlabeled?

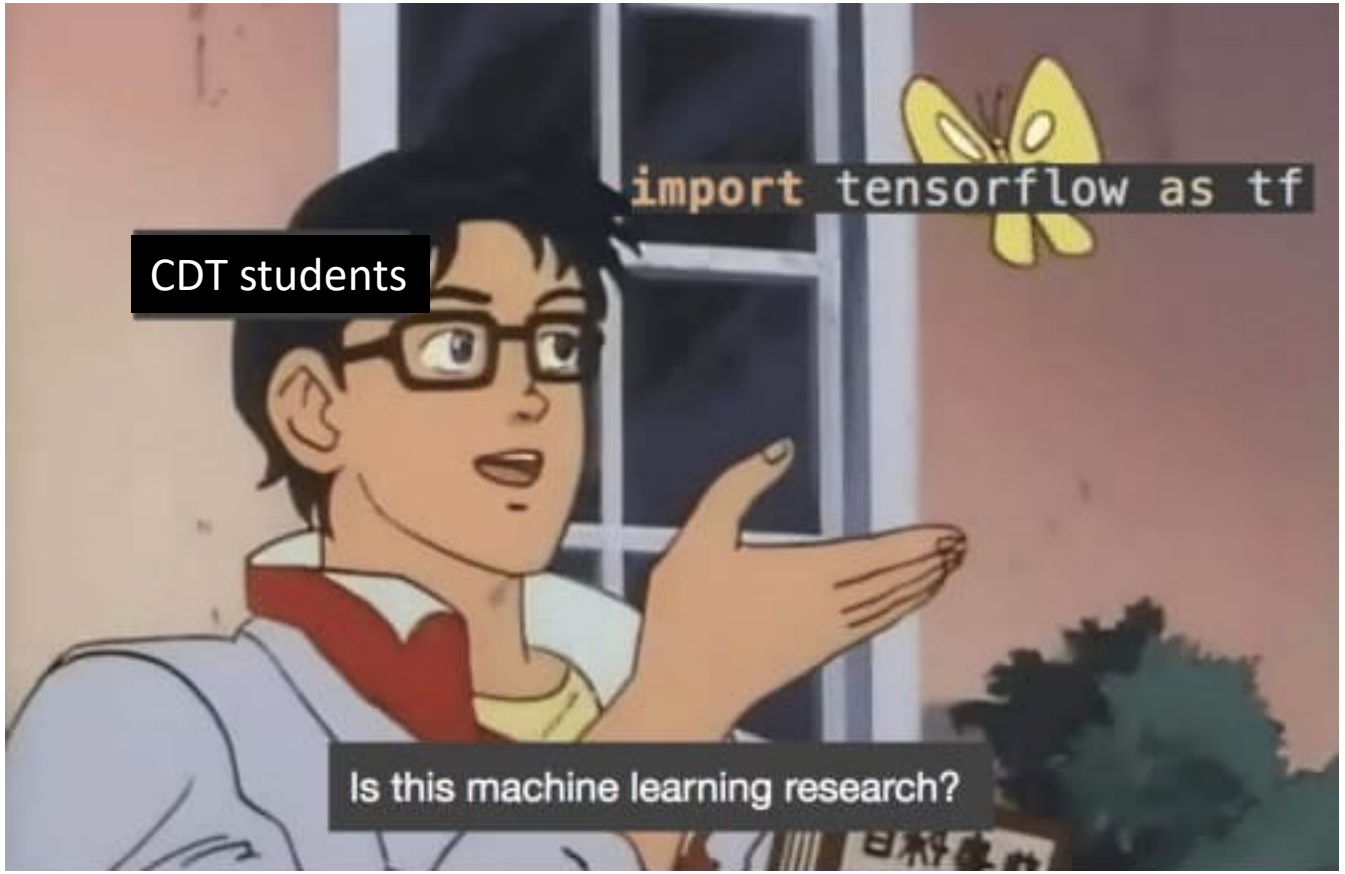
Saeed, A., Ozcelebi, T., and Lukkien, J. Multi-task self-supervised learning for human activity detection. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 3(2):1–30, 2019. https://dl.acm.org/doi/pdf/10.1145/3328932

ML 4 Time-series: Convolutional Neural Networks

What have we learned?

- Model driven and time-domain approaches for time-series
- Hand-crafted feature vs. representation learning
- CNNs can learn various abstractions of features in between layers
- We can use the raw signal or its time-frequency representation as input to our model
- Different kernels can learn different features
- Modify your network architectures for your chosen task
- Data augmentation helps you from overfitting
- Transfer learning can be a powerful tool when your dataset is small

The End.



Yes, it is.

https://medium.com/nybles/understanding-machine-learning-through-memes-4580b67527bf
https://knowyourmeme.com/memes/is-this-a-pigeon

The End.

Uh oh

https://medium.com/nybles/understanding-machine-learning-through-memes-4580b67527bf
https://knowyourmeme.com/memes/is-this-a-pigeon