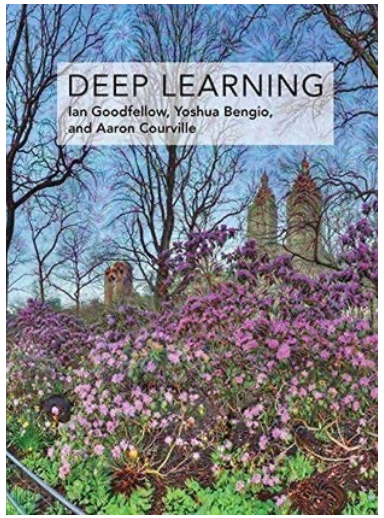
10th November 2021

ML 4 Time-series: Convolutional Neural Networks

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

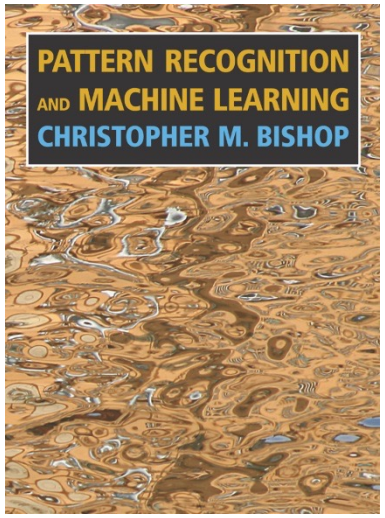
Institute of Biomedical Engineering (IBME),
Department of Engineering Science,
Old Road Campus Research Building (ORCRB),
University of Oxford

Resources



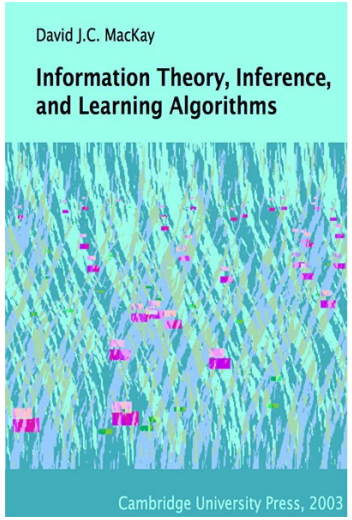
A well-written introduction to all things deep learning (DL), from leaders in the field of theoretical DL.

Very light maths



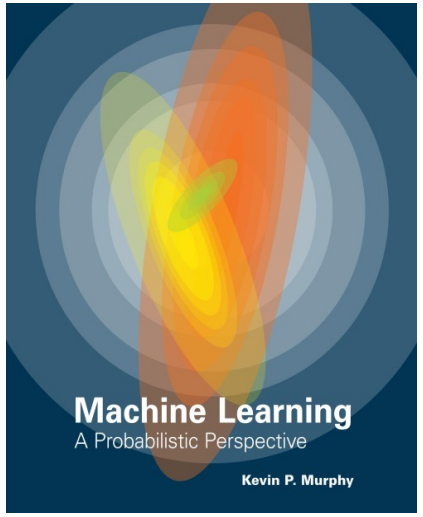
A core classic describing most non-DL algorithms. Very good for one's general understanding.

Reasonably "maths-y"



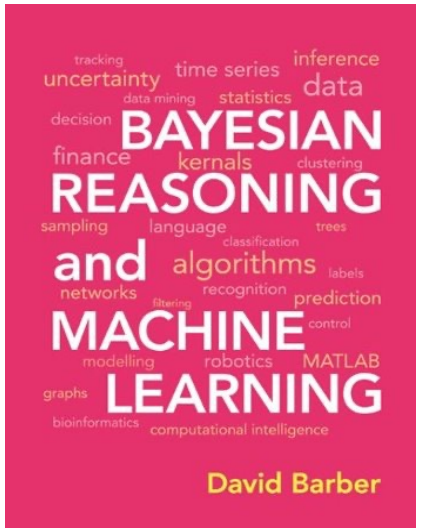
Another core classic, from one of the field's (sadly departed) foundational thinkers – good, even though it's from Cambridge

Reasonably "maths-y"



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

Really rather "maths-y"



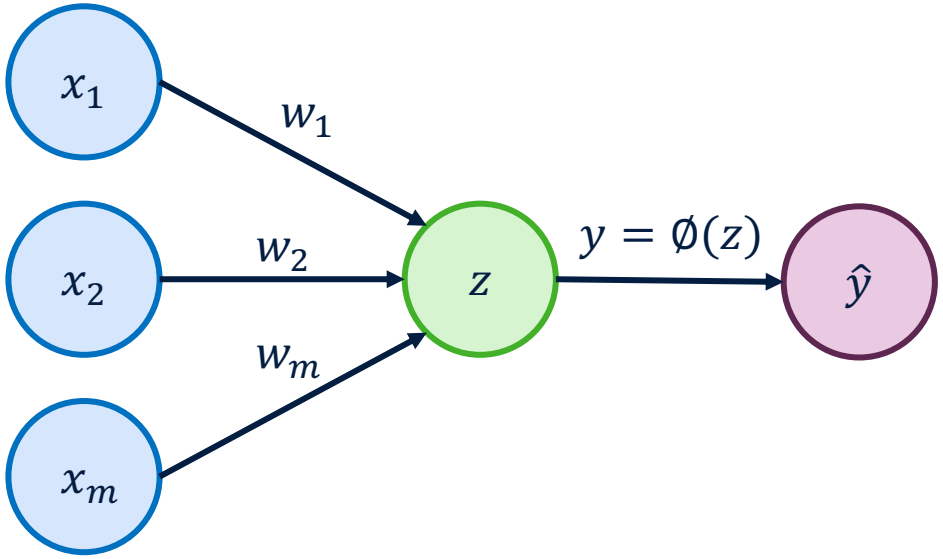
Big, bad, and Bayesian. Everything you'd like to know about Bayesian machine learning. Great for time-series analysis.

Seriously "maths-y"

mild — medium — hot — extra hot

The Perceptron

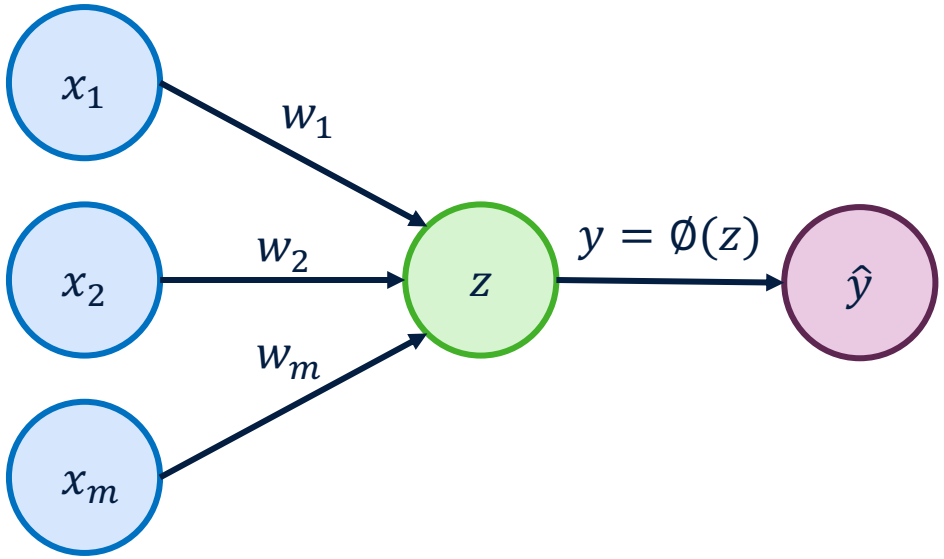
The perceptron is the building block to modern day feed-forward deep networks. A perceptron layer takes a weighted linear combination (sum) of the inputs $\mathbf{x} \in \mathbb{R}^m$ and transform this to an output through some (often non-linear) positive and monotonically increasing activation function $\emptyset(\cdot)$ to yield a prediction or output $\hat{y}$



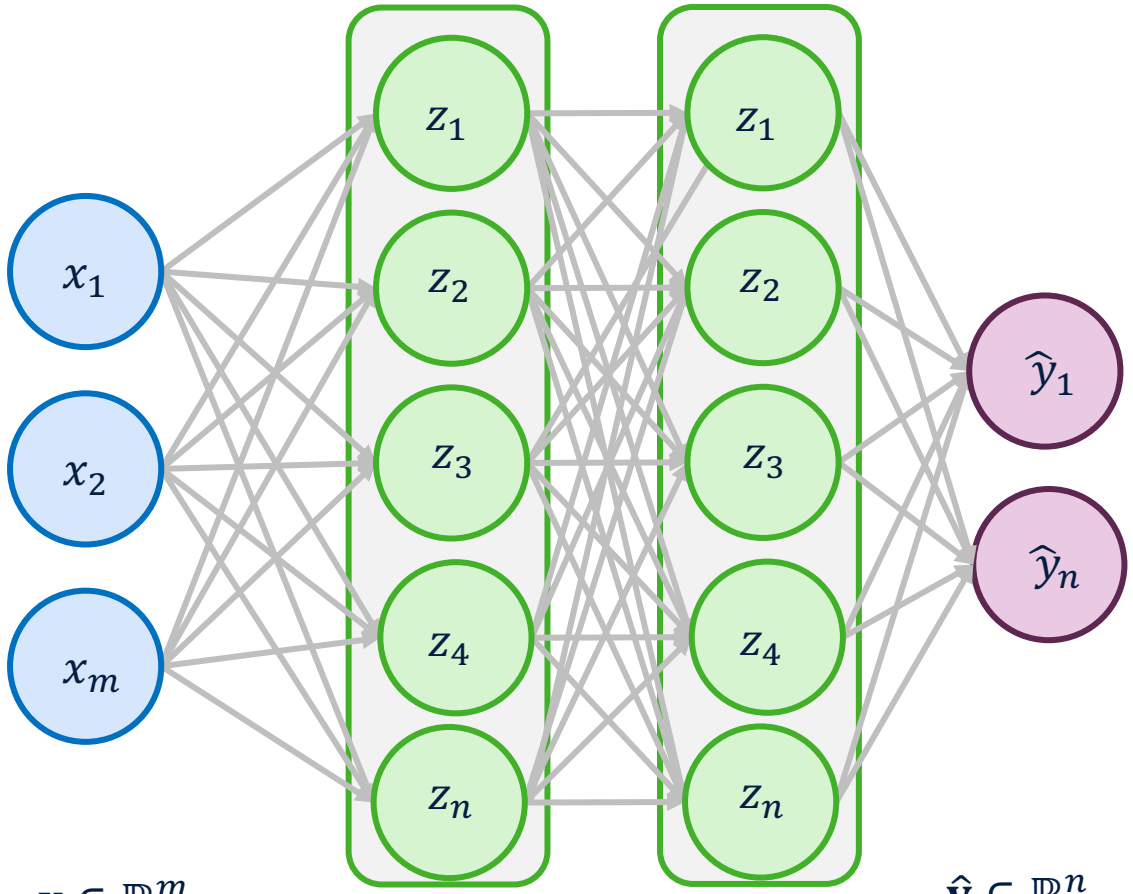
$\mathbf{x} \in \mathbb{R}^m$ $\quad z = \mathbf{w}^\intercal \mathbf{x}$ $\quad \hat{\boldsymbol{y}} = \emptyset(\mathbf{w}^\intercal \mathbf{x})$

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

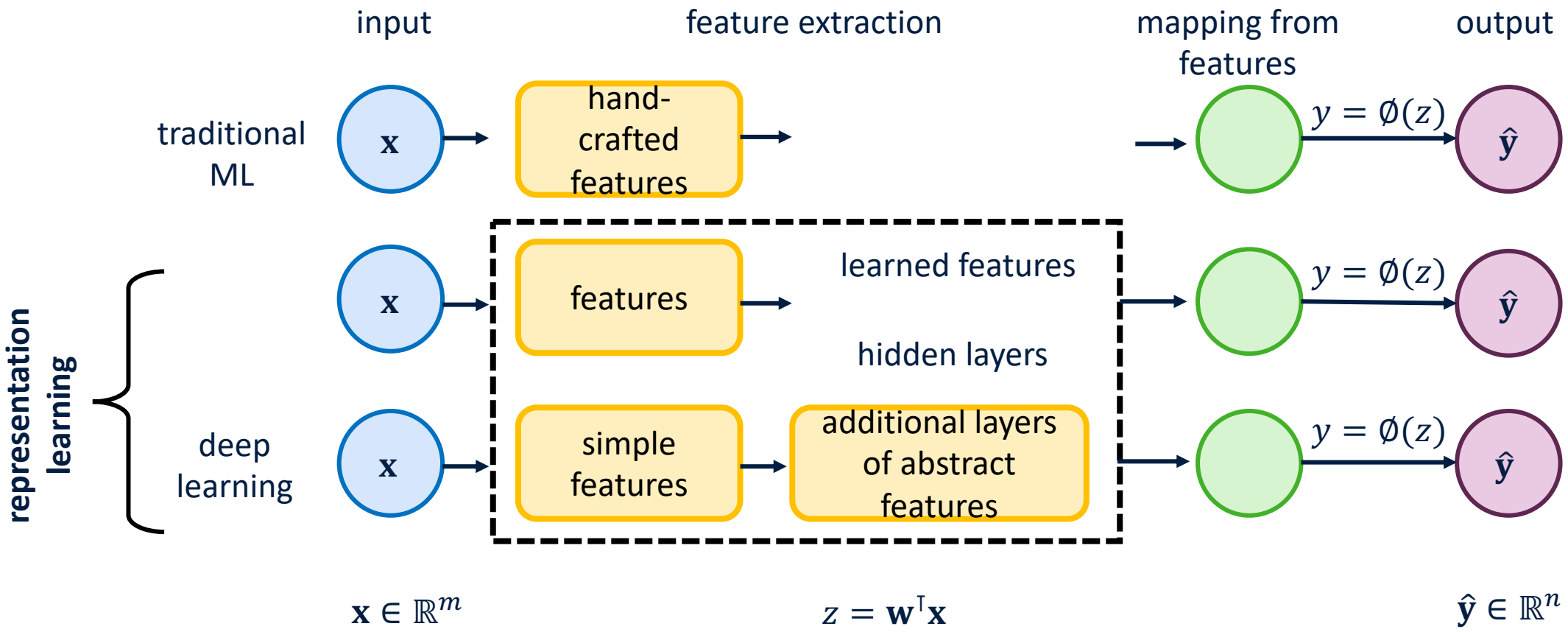


Image credit: adapted from https://www.deeplearningbook.org/contents/intro.html

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

Convolutional Neural Networks

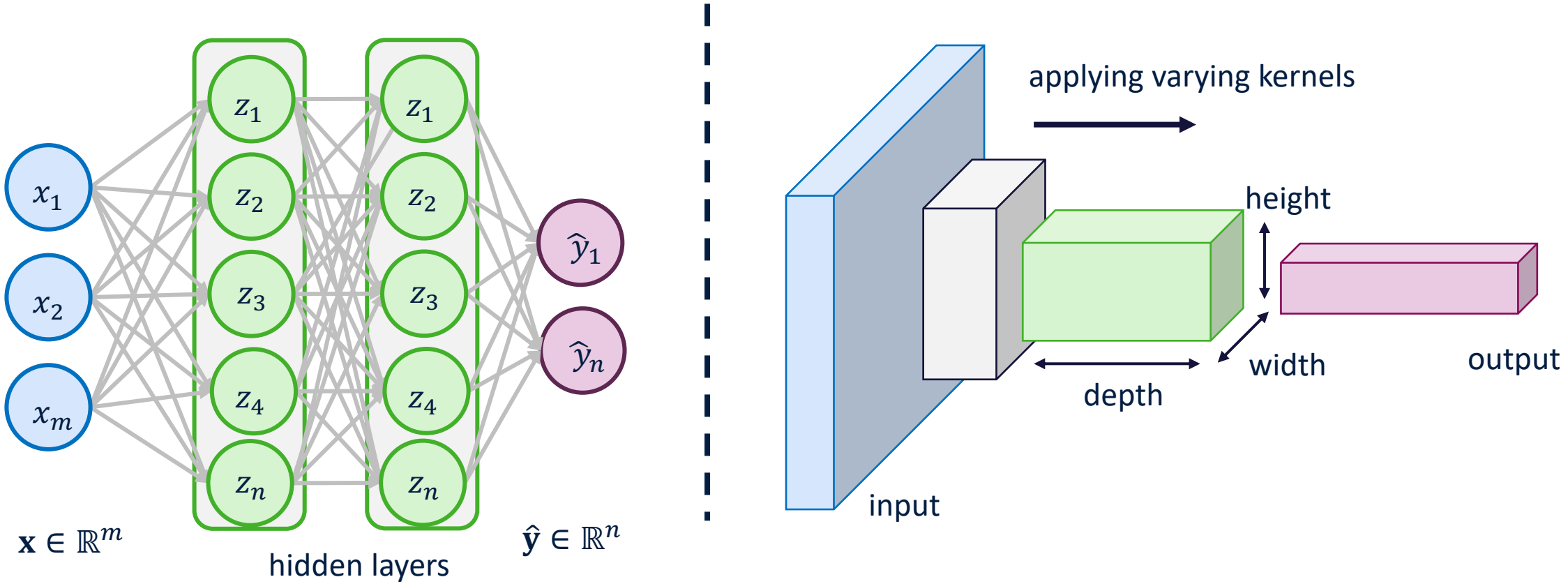
Convolutional Neural Networks (CNNs) are a class of ANN algorithm that follow a grid like topology and are especially robust displaying *translation invariance*. Translational invariance refers to a representation that is approximately invariant to small translations of the input. For instance, when 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.

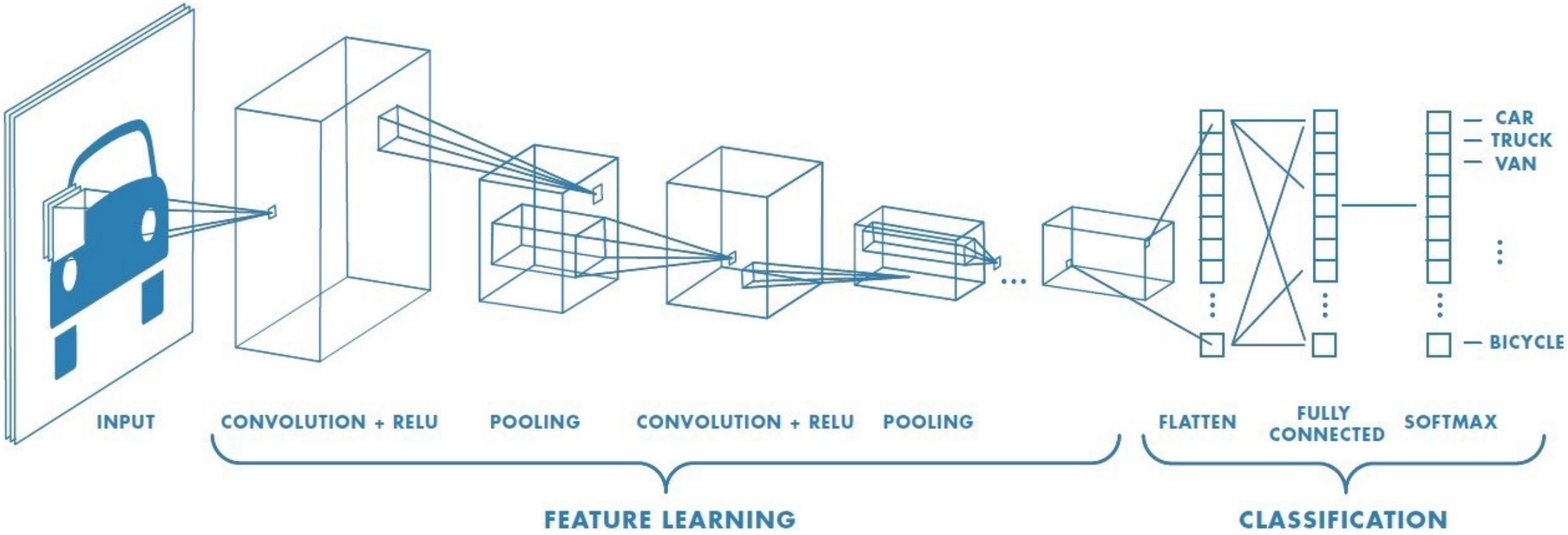
Convolutional Neural Networks



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)

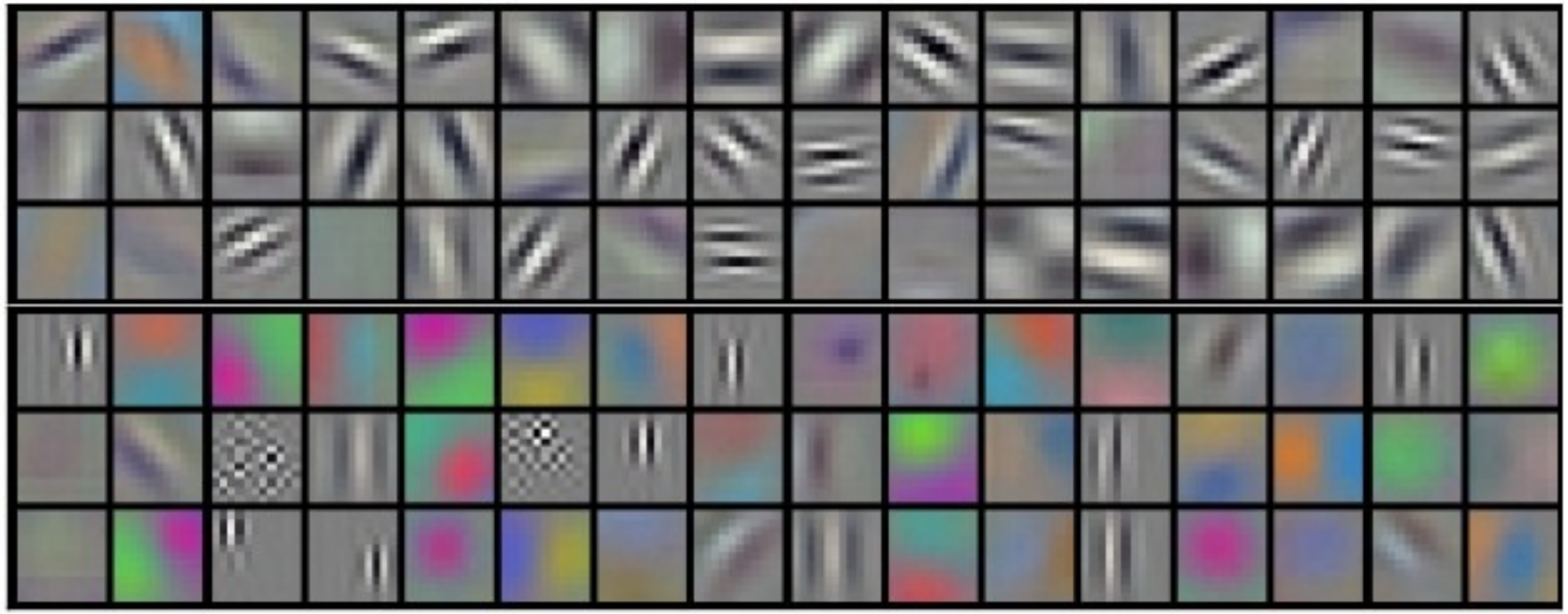
Credit: https://cs231n.github.io/convolutional-networks/

CNNs learn features



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

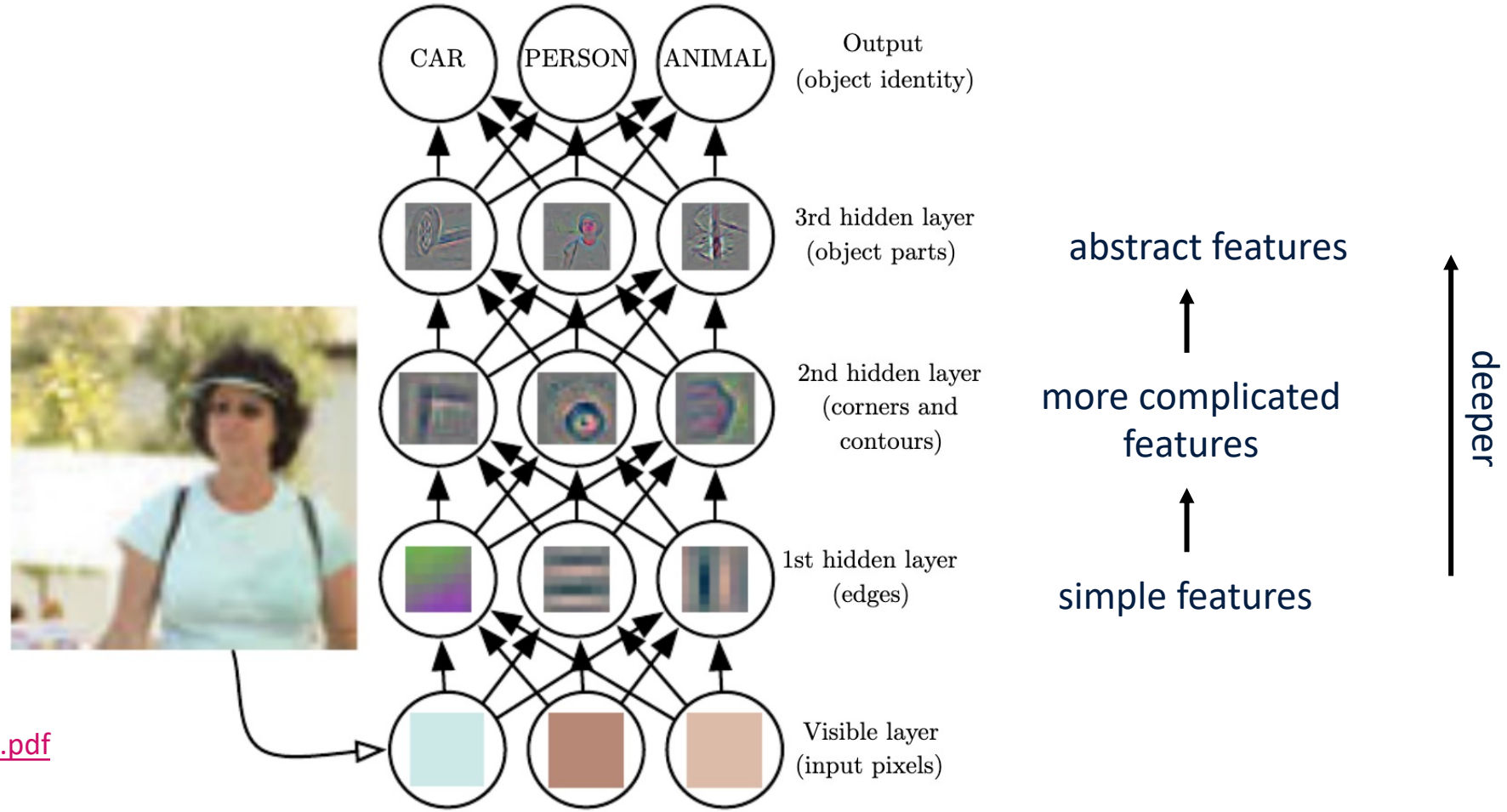
Filters as feature extractors



Example filters learned by Krizhevsky et al. (2012). Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice. Given the **parameter sharing assumption,** if detecting a horizontal edge is important at some location in the image, it should intuitively be useful at some other location as well due to the translationally-invariant structure of images. There is therefore no need to relearn to detect a horizontal edge at every one of the 55*55 distinct locations in the Conv layer output volume; it is learned by a filter.

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

Learning complicated representations

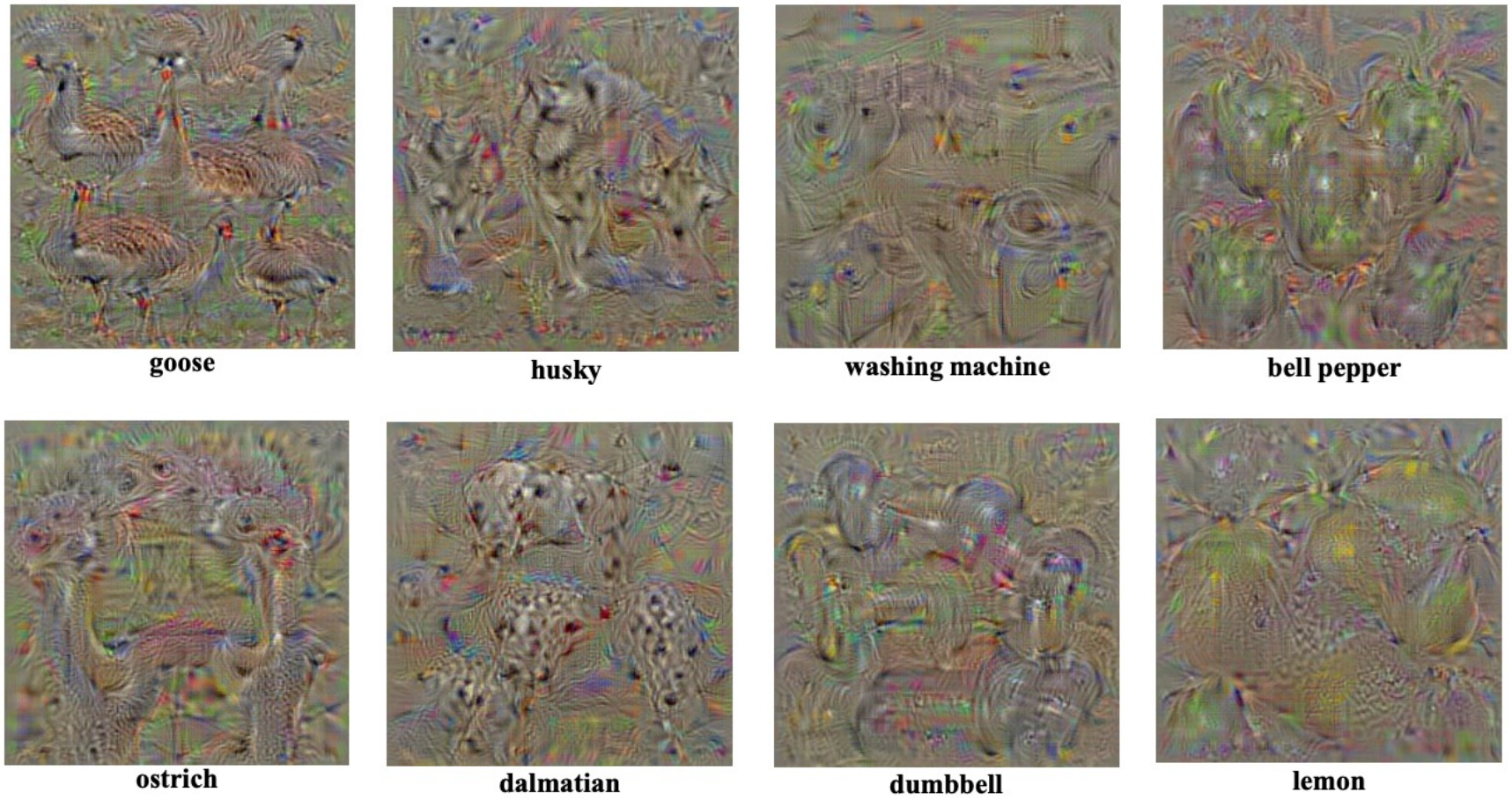


Zeiler & Fergus (2014)
https://arxiv.org/pdf/1311.2901.pdf

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

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!)

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

The race to the bottom*

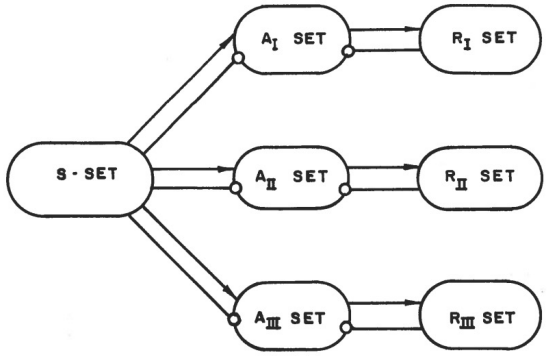
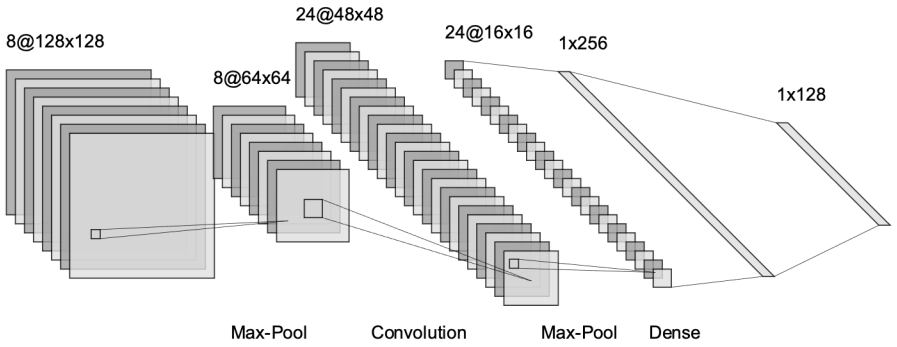


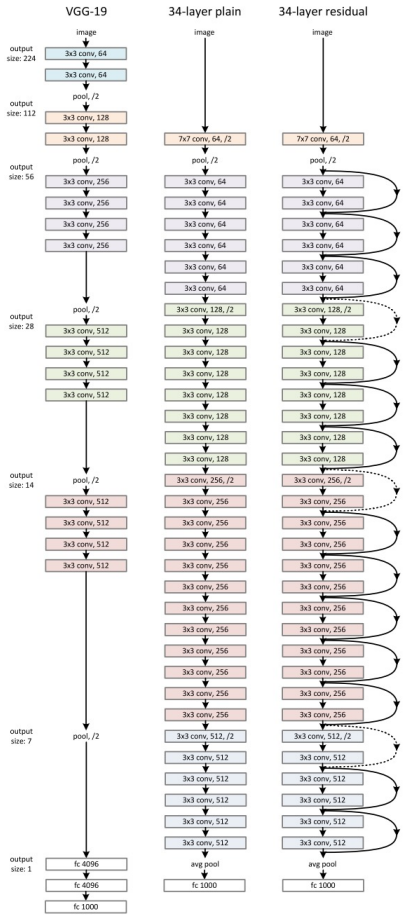
FIGURE 2

ORGANIZATION OF A PERCEPTRON WITH THREE INDEPENDENT OUTPUT-SETS

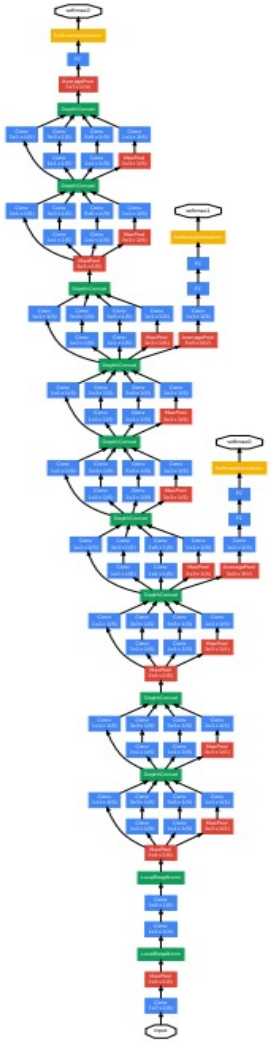
Perceptron (1957)



LeCun Net (1998)



ResNet (2015)



InceptionV3 (2015)

*probably ended circa 2015-2018

The race to the bottom*

Inception: Christopher Nolan film with Leonardo DiCaprio (2010)

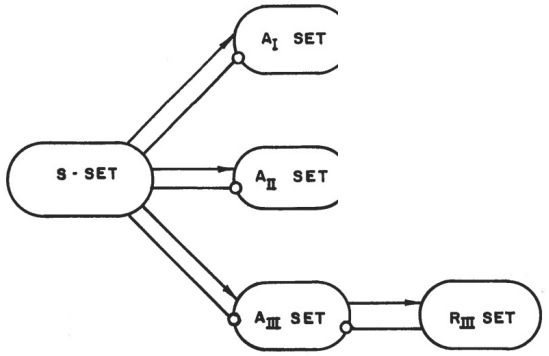
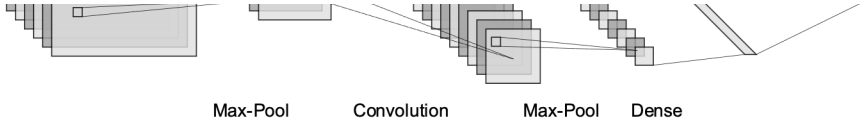
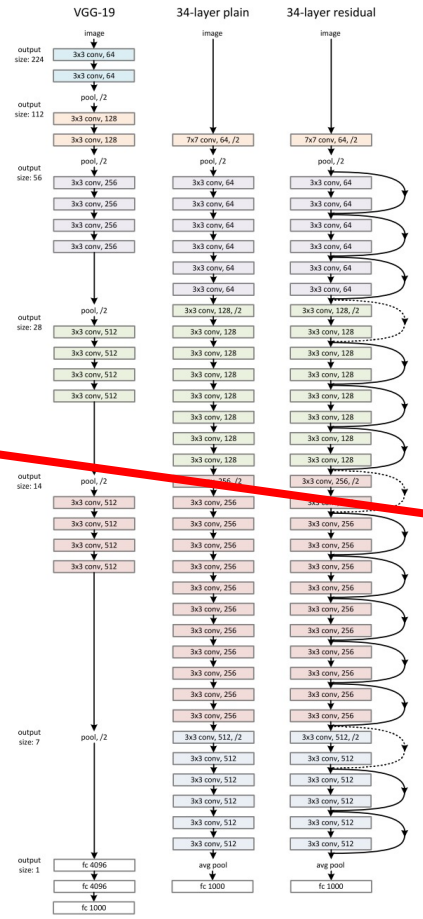
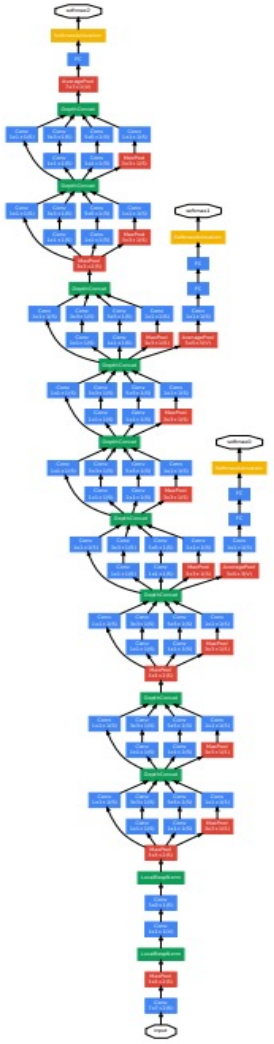


FIGURE 2

ORGANIZATION OF A PERCEPTRON WITH THREE INDEPENDENT OUTPUT-SETS







Perceptron (1957)

LeCun Net (1998)

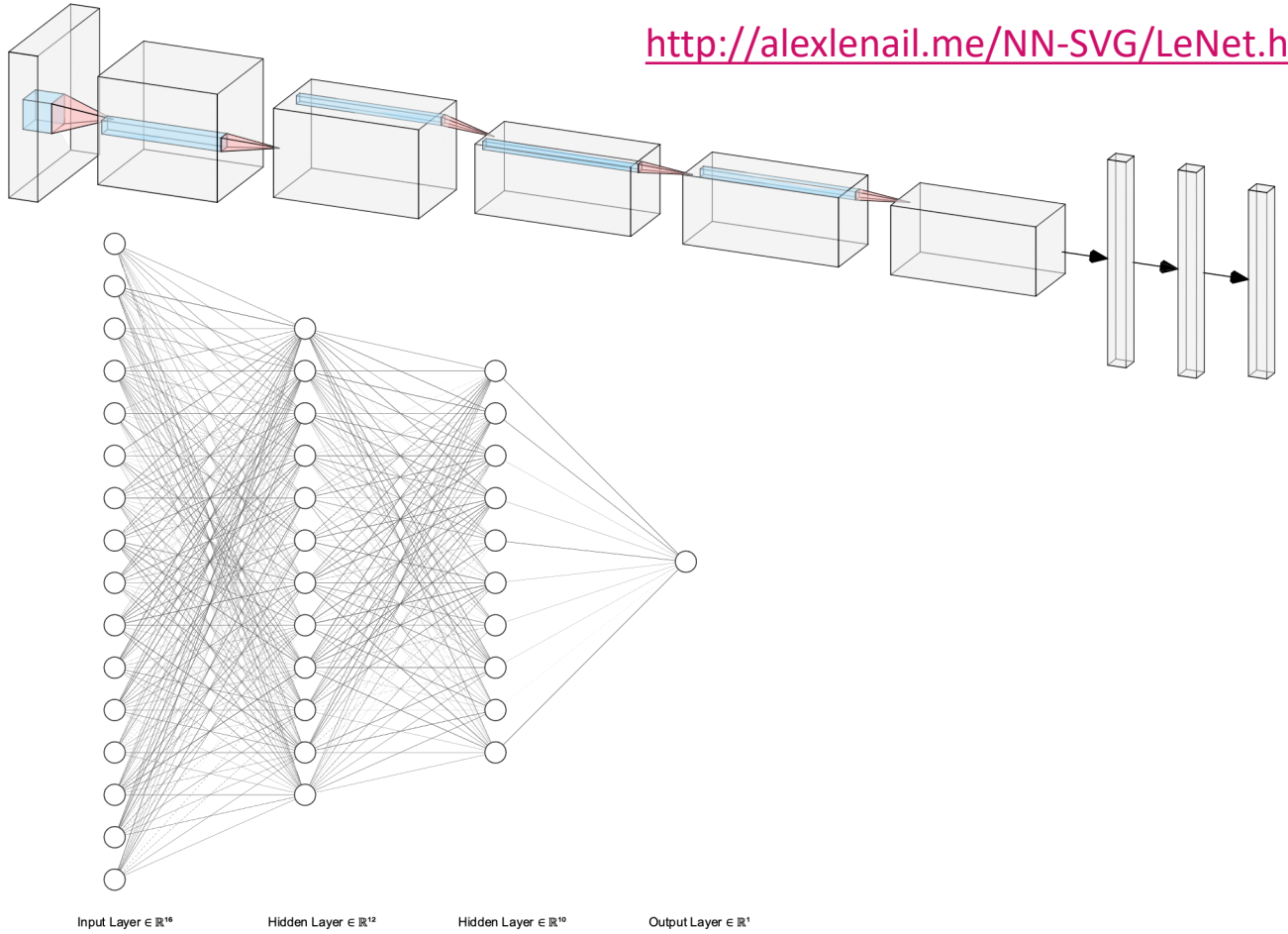
ResNet (2015)

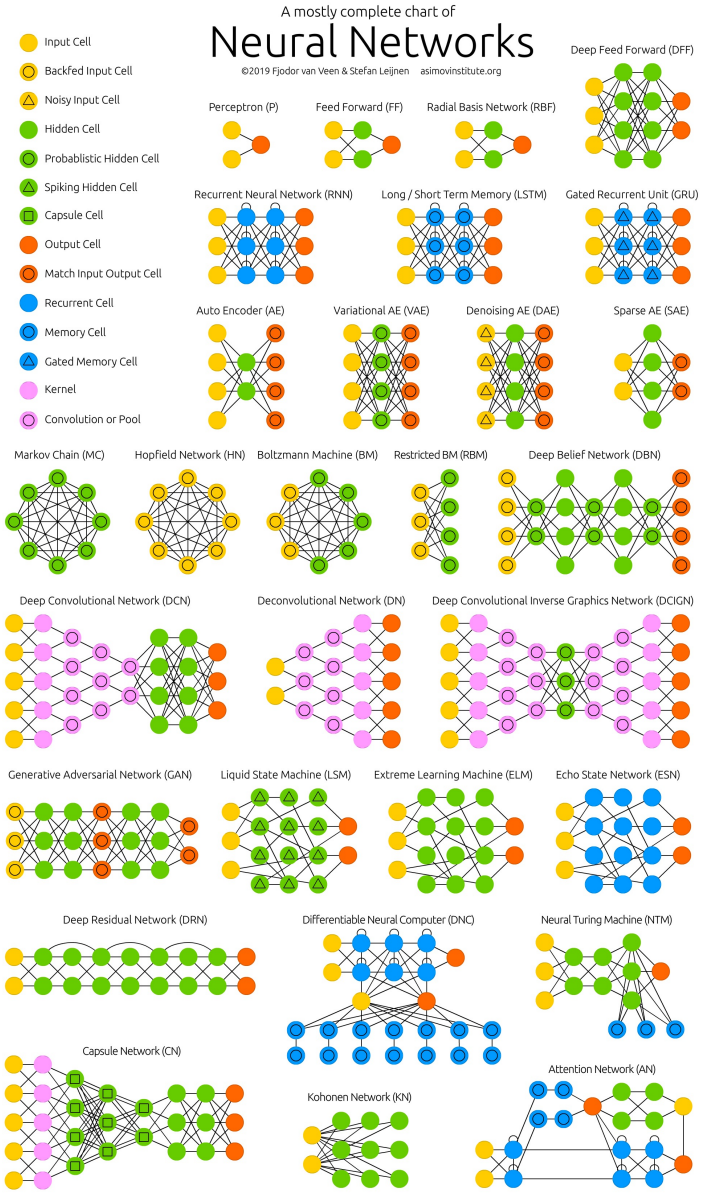
InceptionV3 (2015)

*probably ended circa 2015-2018

Some nice resources: plotting networks

http://alexlenail.me/NN-SVG/LeNet.html





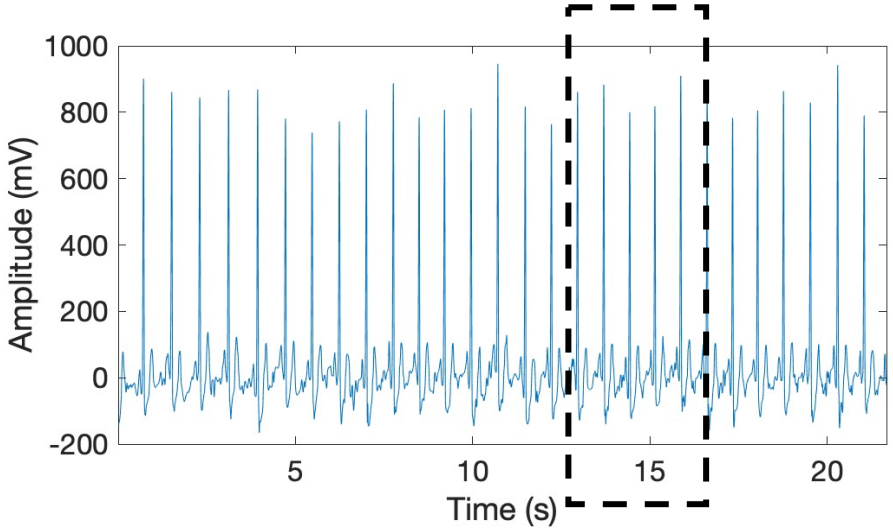
https://www.asimovinstitute.org/neural-network-zoo/

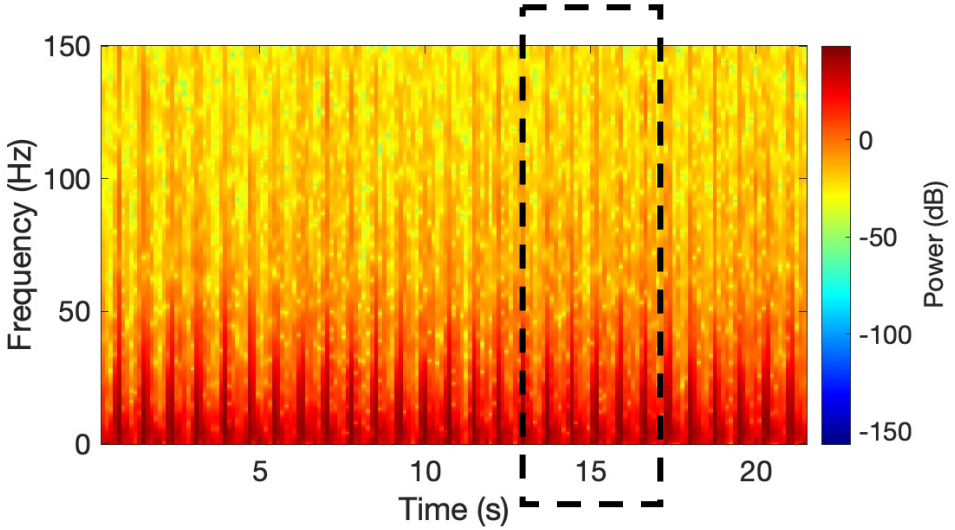
CNNs 4 time-series analysis

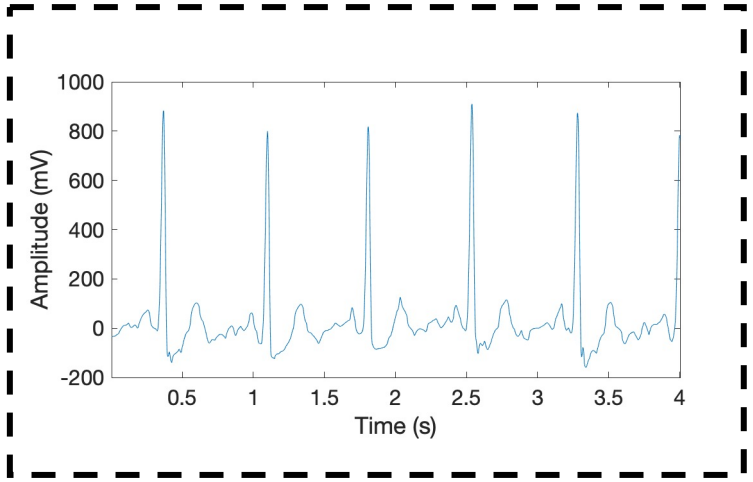
How do I use CNN with a time-series?

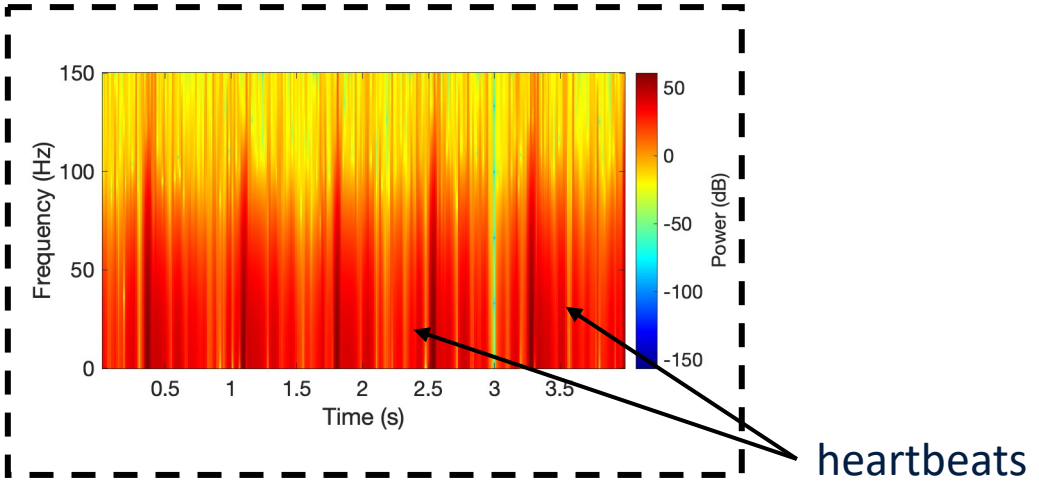
- Spectrograms as CNN inputs
- Raw time-series as CNN inputs
- Picking your kernels and network architectures for your task
- Incorporating temporality into your network architecture

Representing a time-series as images





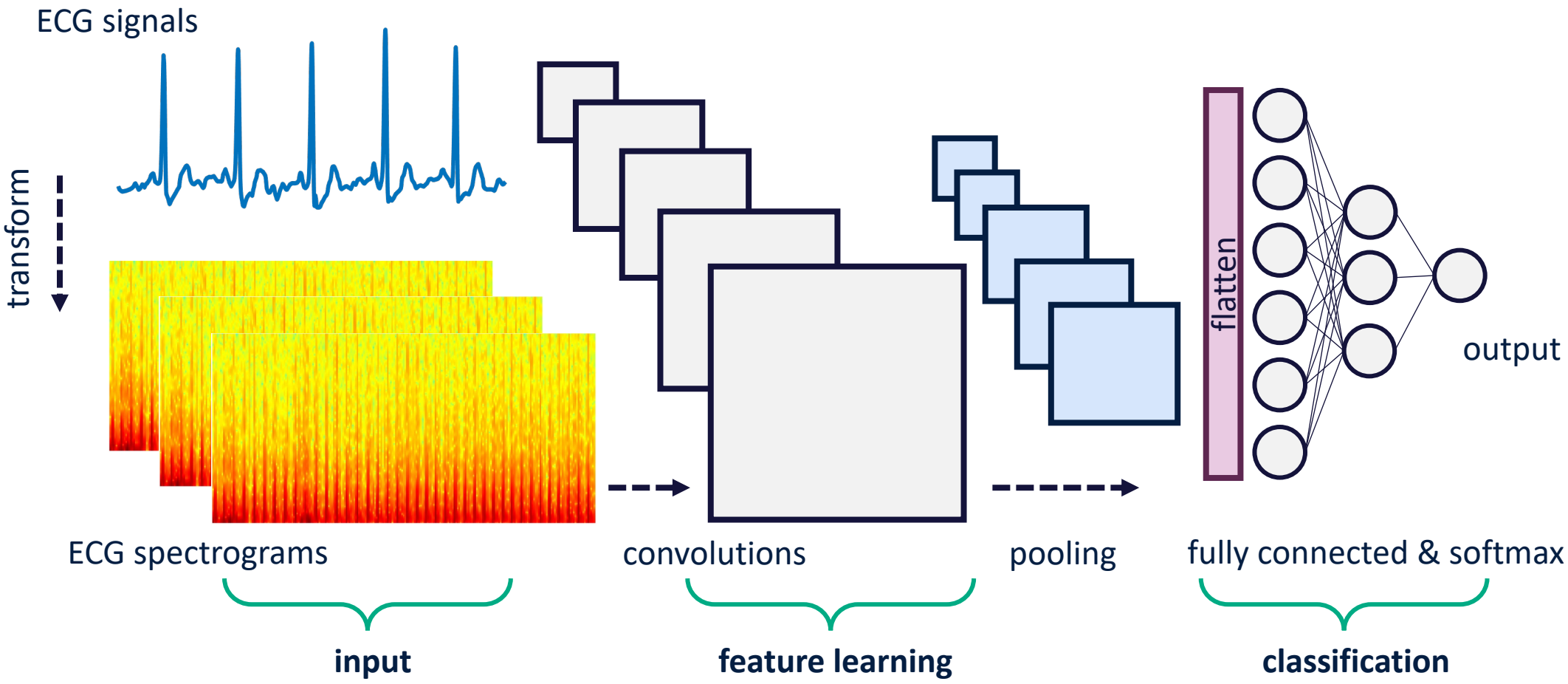




Raw ECG time series

Spectrogram of ECG

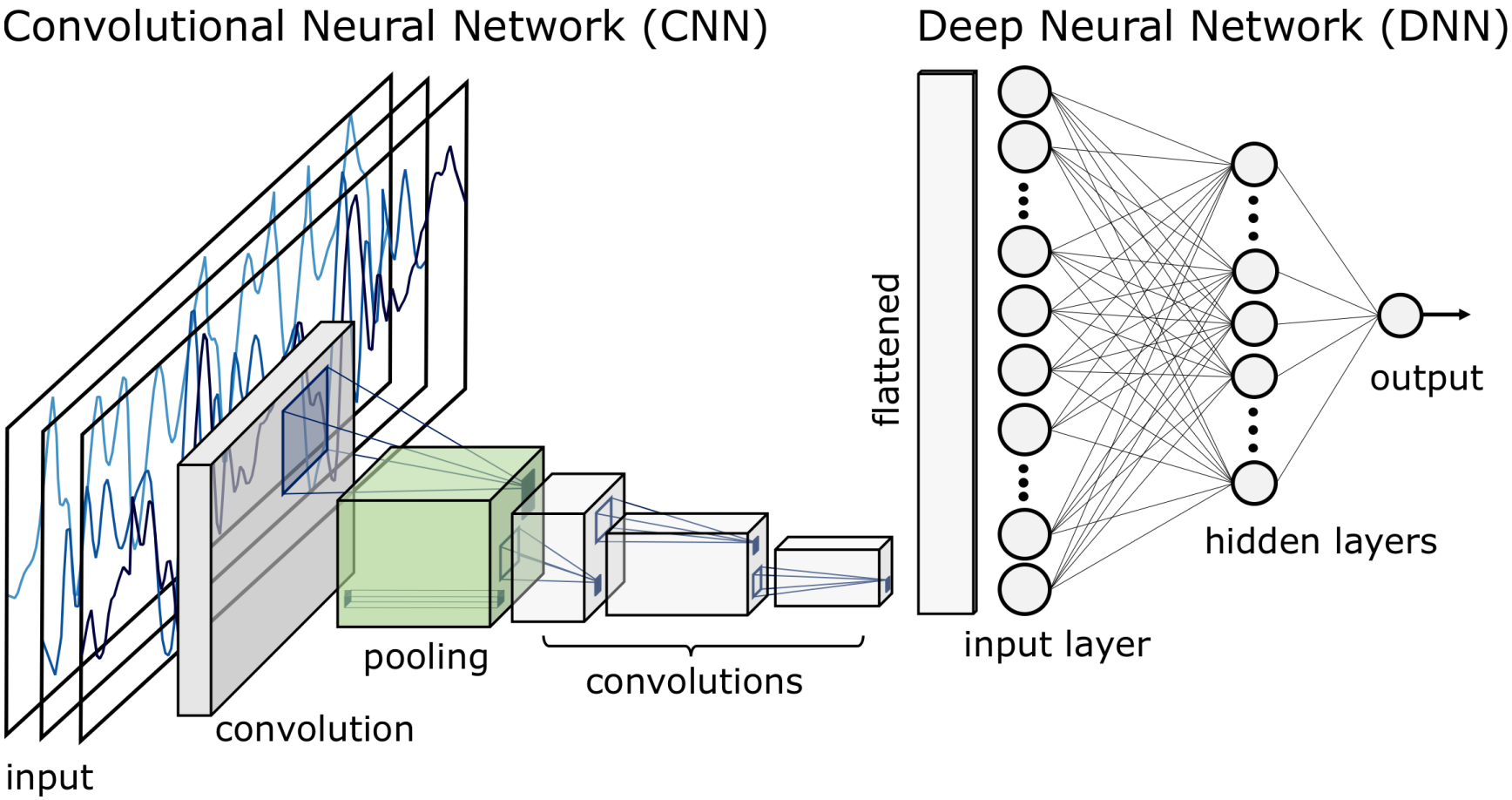
Spectrograms as CNN inputs



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

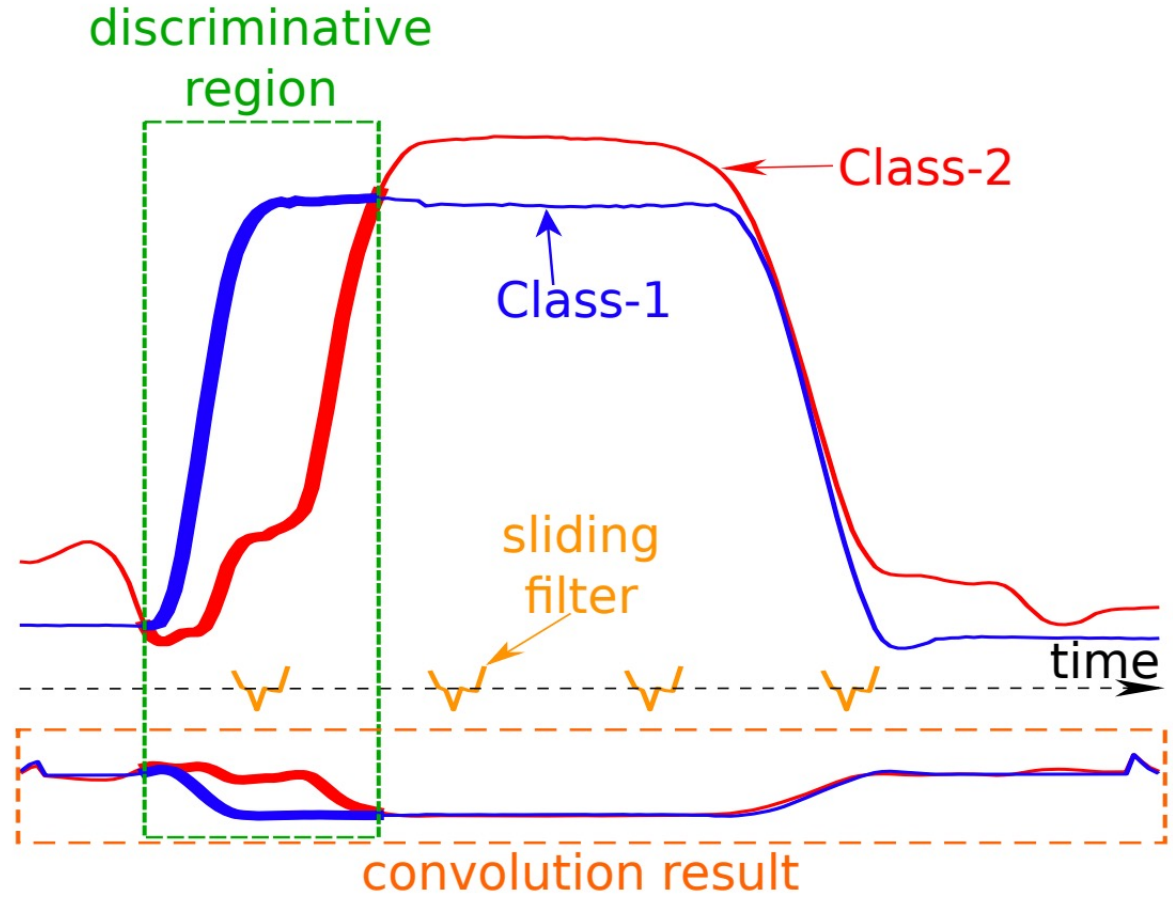
Raw time-series as CNN input



Typical CNN architecture for time-series input classification

Creagh A.P., et al. (2021), *"The Development of Digital Biomarkers for Multiple Sclerosis from Remote Smartphone- and Smartwatch-Based Assessments".* D.Phil. Thesis

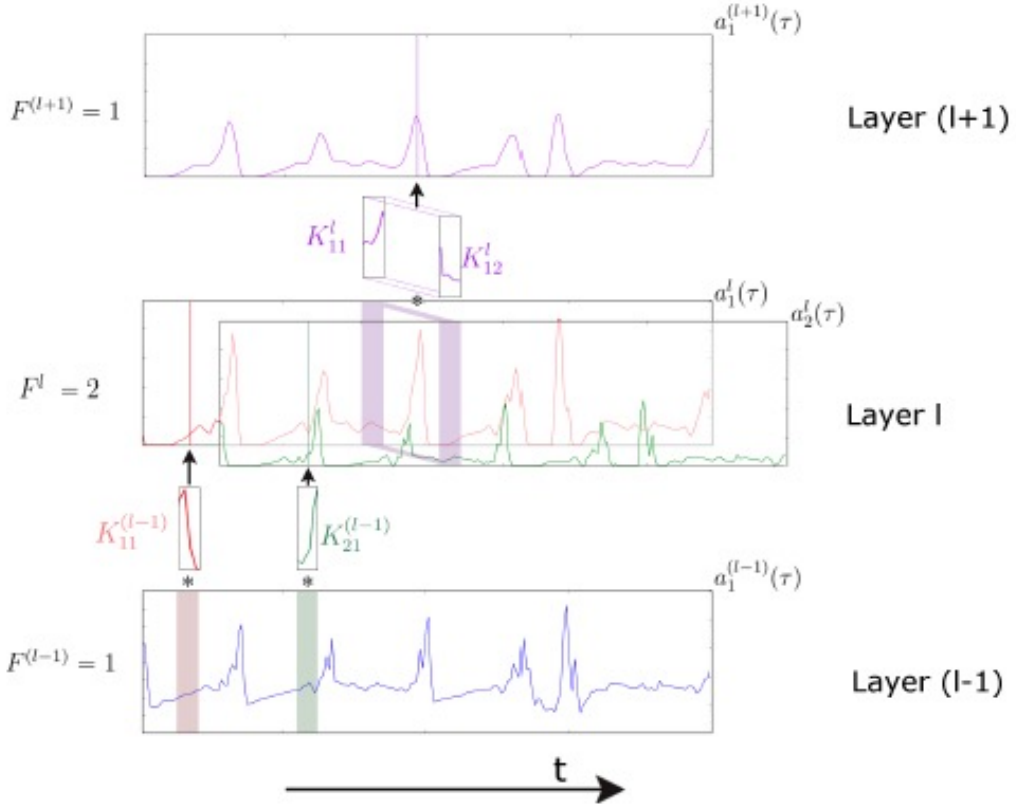
Applying CNN kernels to a time-series



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

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

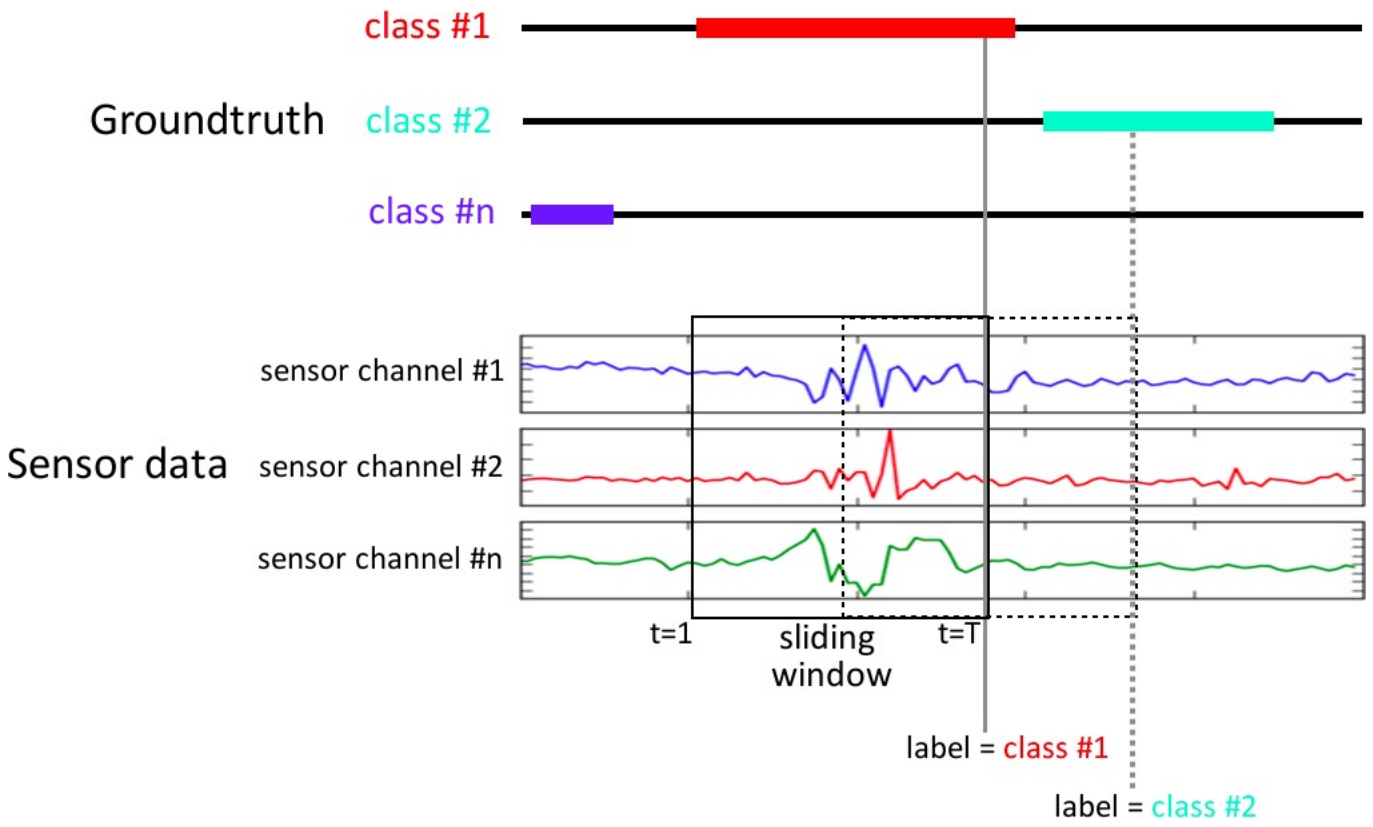
Applying CNN kernels to a time-series



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

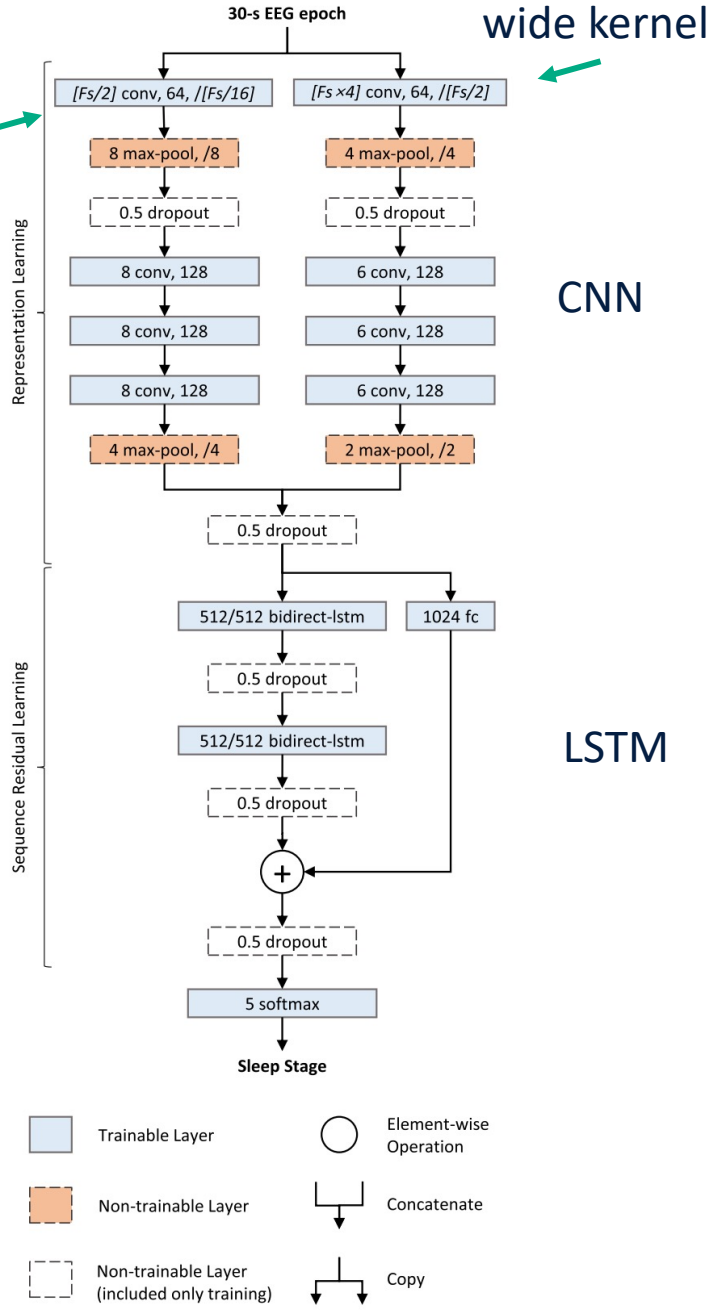
$K_{(wide)} = {}^{fs}/_{2}$ $K_{(narrow)} = {}^{fs}/_{16}$ fs: sampling frequency (Hz)

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.

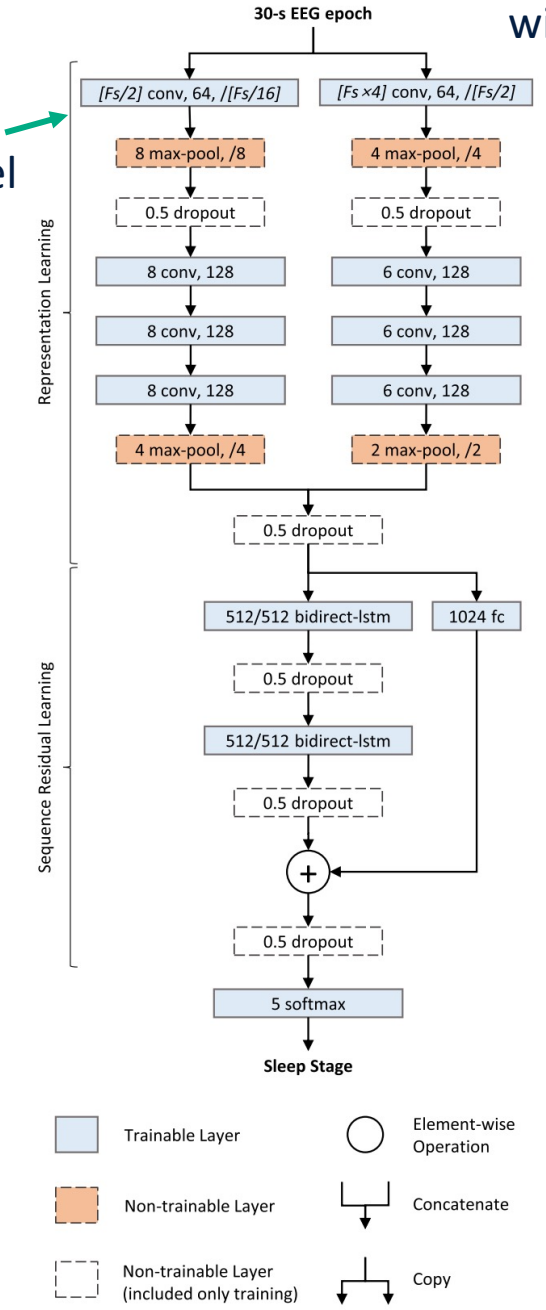
Choosing your kernel

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

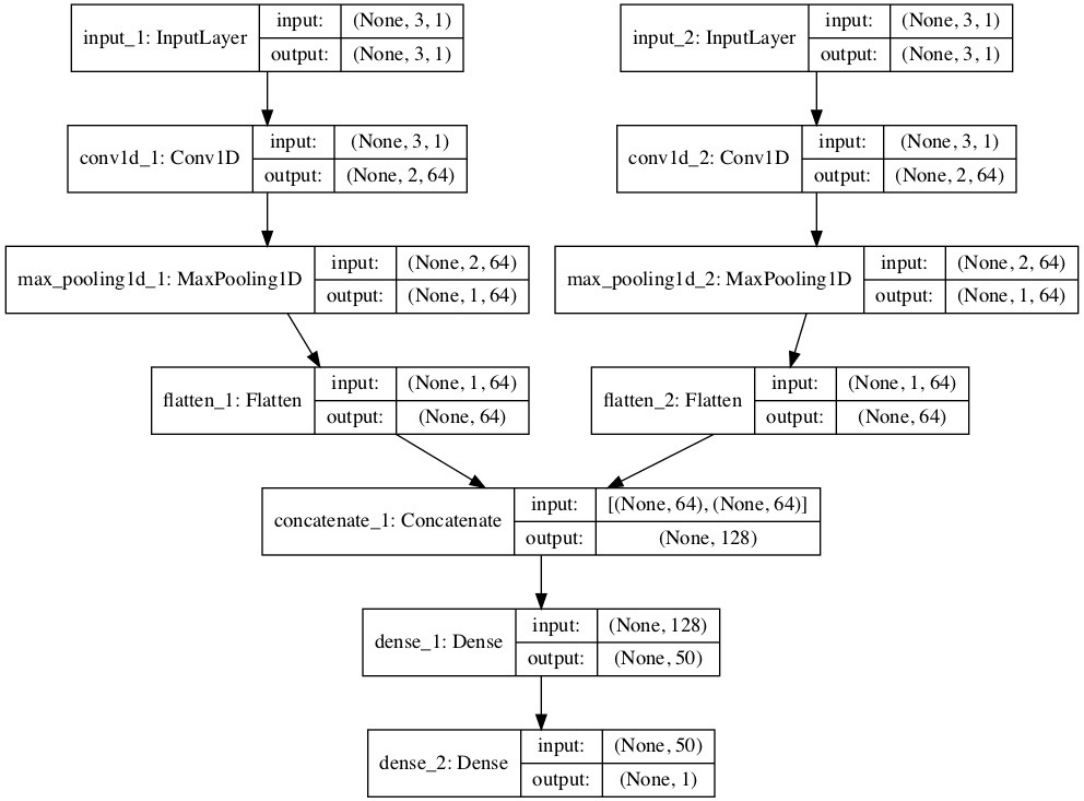
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.backends.cudnn as cudnn

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
```

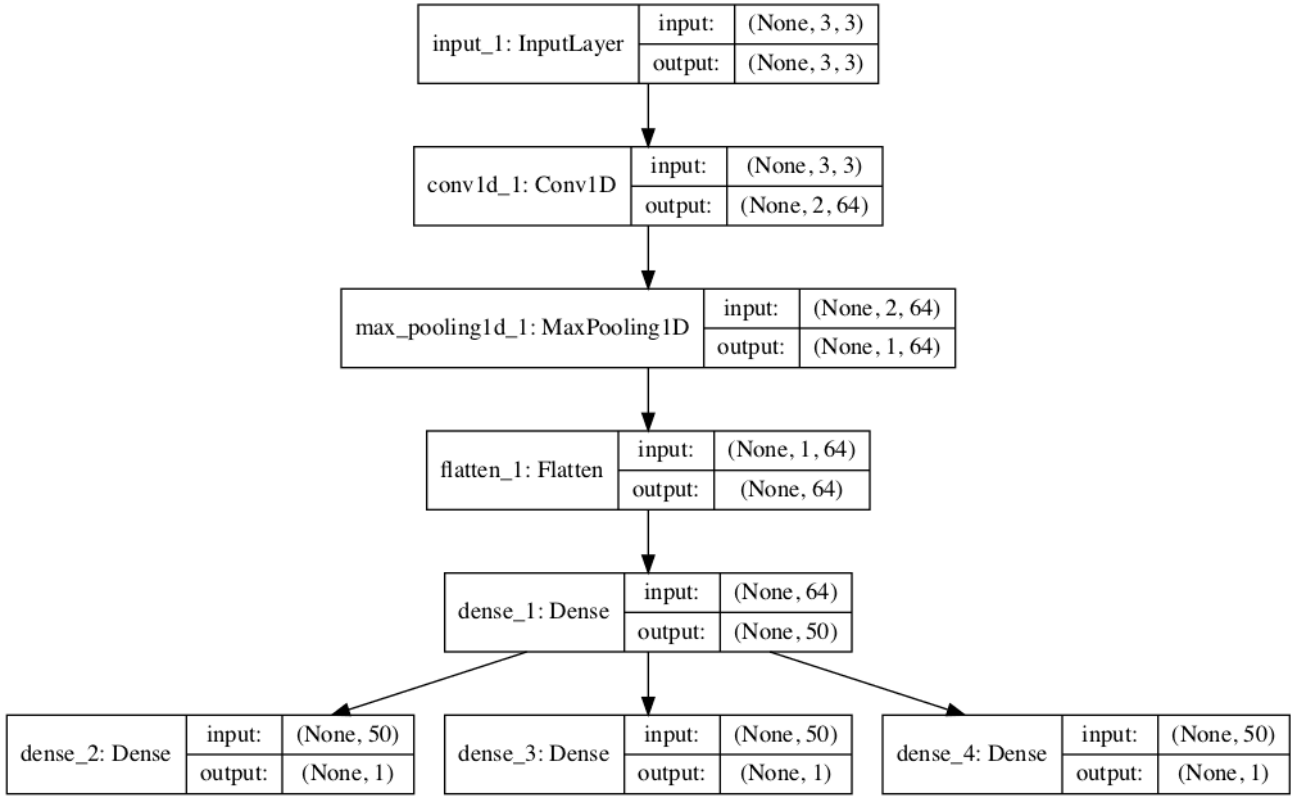
CNN architectures for time-series analysis

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

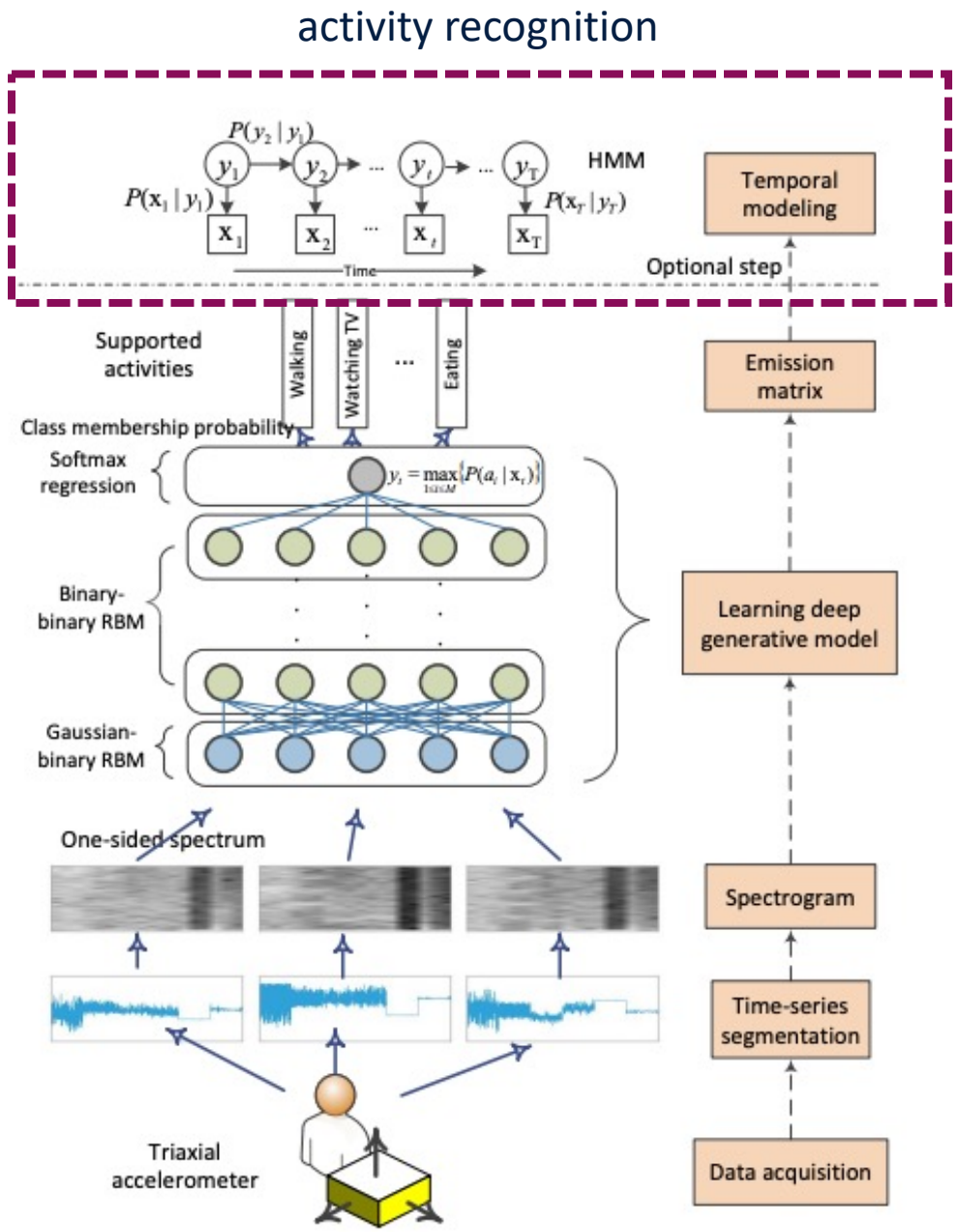
Source: Brownlee, Jason. Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery, 2018.

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.



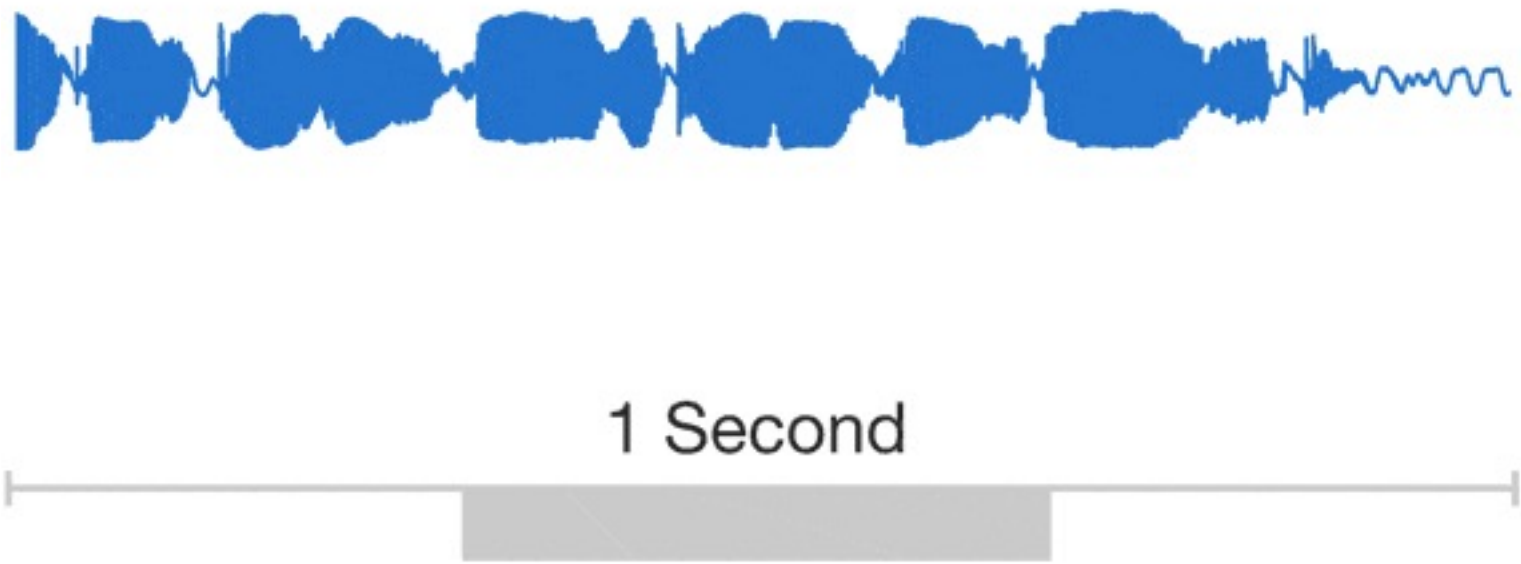
Alsheikh, M. A., Selim, A., Niyato, D., Doyle, L., Lin, S., & Tan, H. P. (2016). Deep activity recognition models with triaxial accelerometers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

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

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

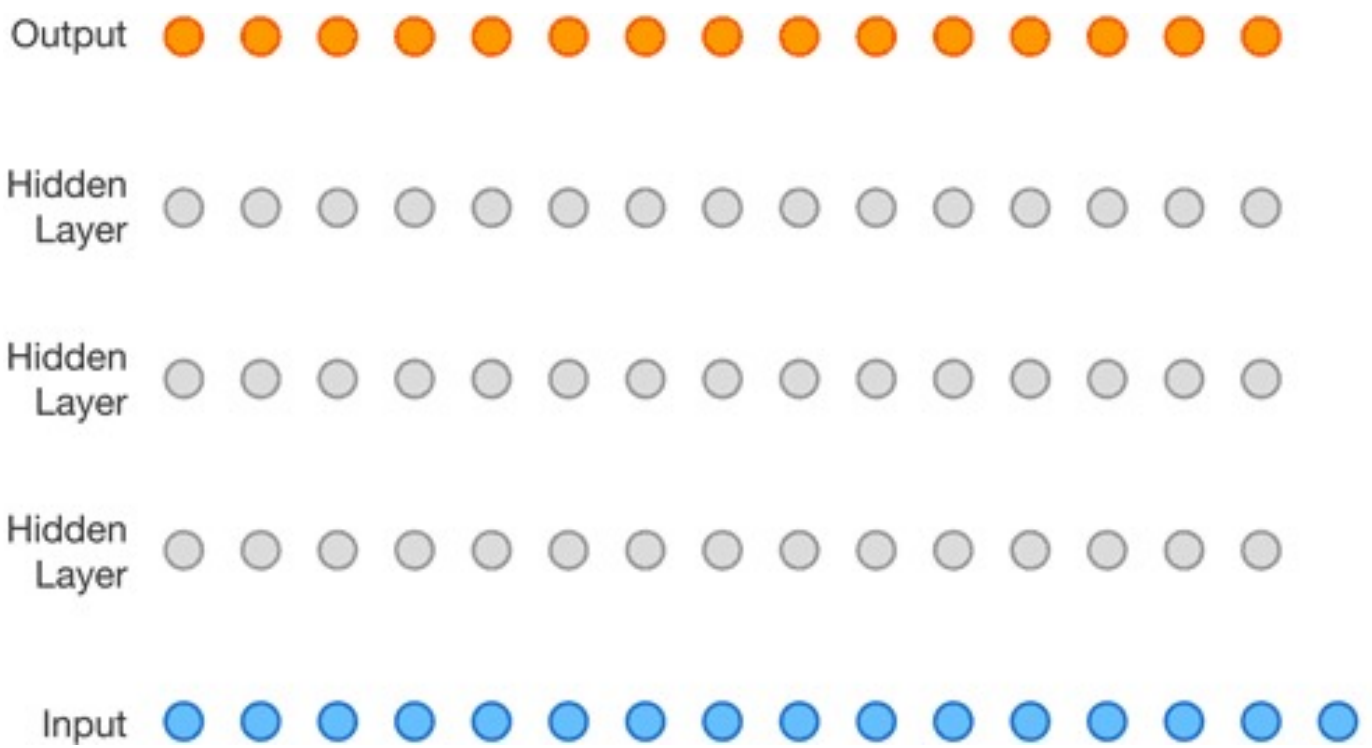
Each audio sample x_t is therefore conditioned on the samples at all previous timesteps

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

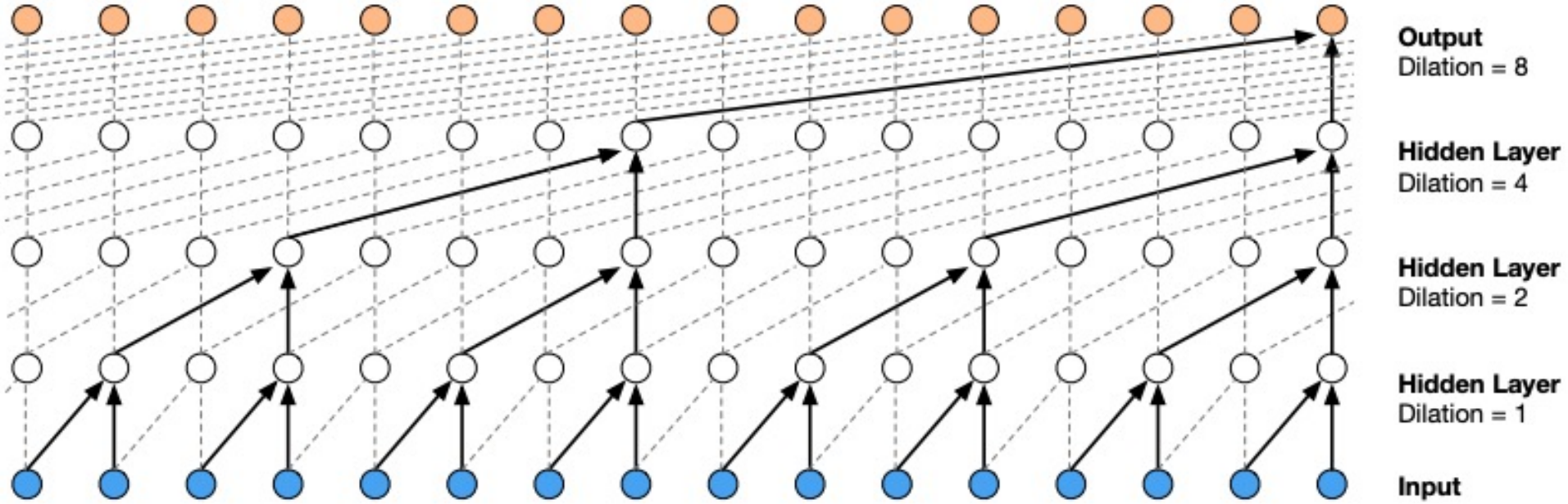
A Dilated Causal Convolution is a causal convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. A dilated causal convolution effectively allows the network to have very large receptive fields with just a few layers.

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

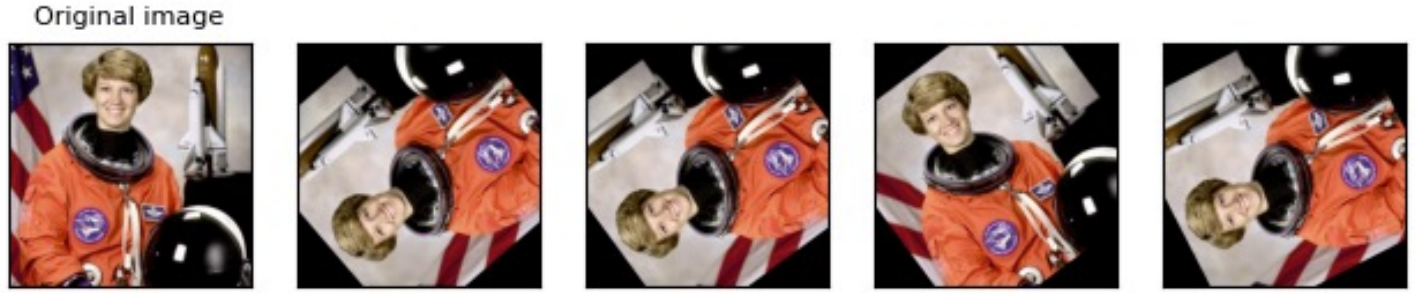
Developing robust models

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

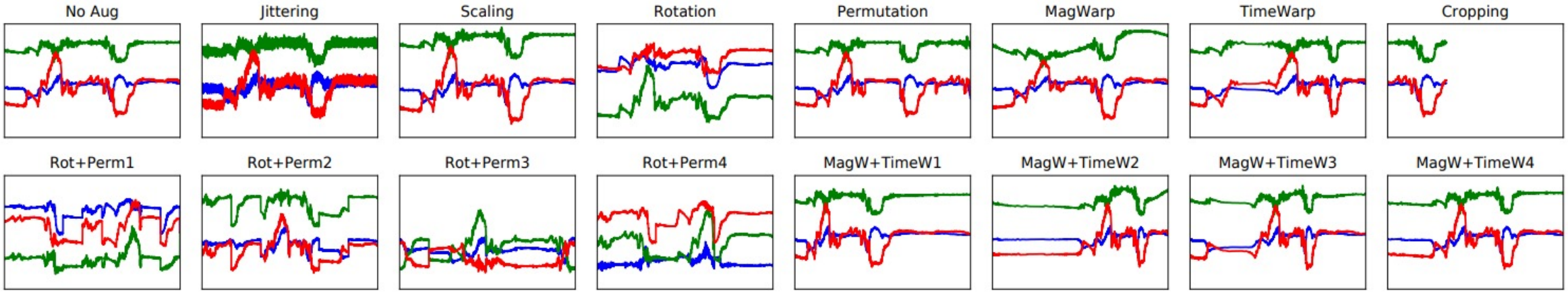


Figure 2: Various data augmentations that are used in the experiments: jittering, scaling, rotating, permutating, magnitude-warping, time-warping methods. Combinations of various data augmentations can also be applied.

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

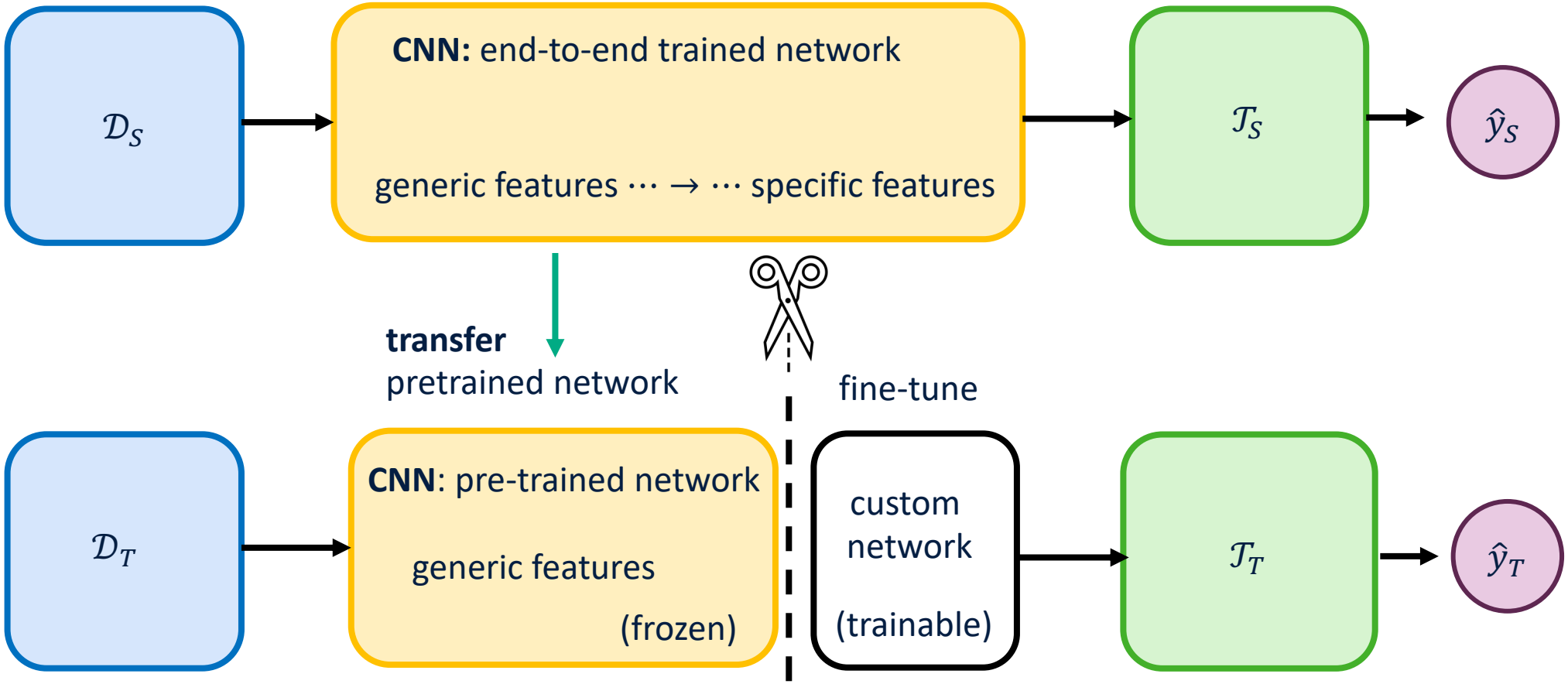
Code: https://github.com/terryum/Data-Augmentation-For-Wearable-Sensor-Data/

Transfer learning

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

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

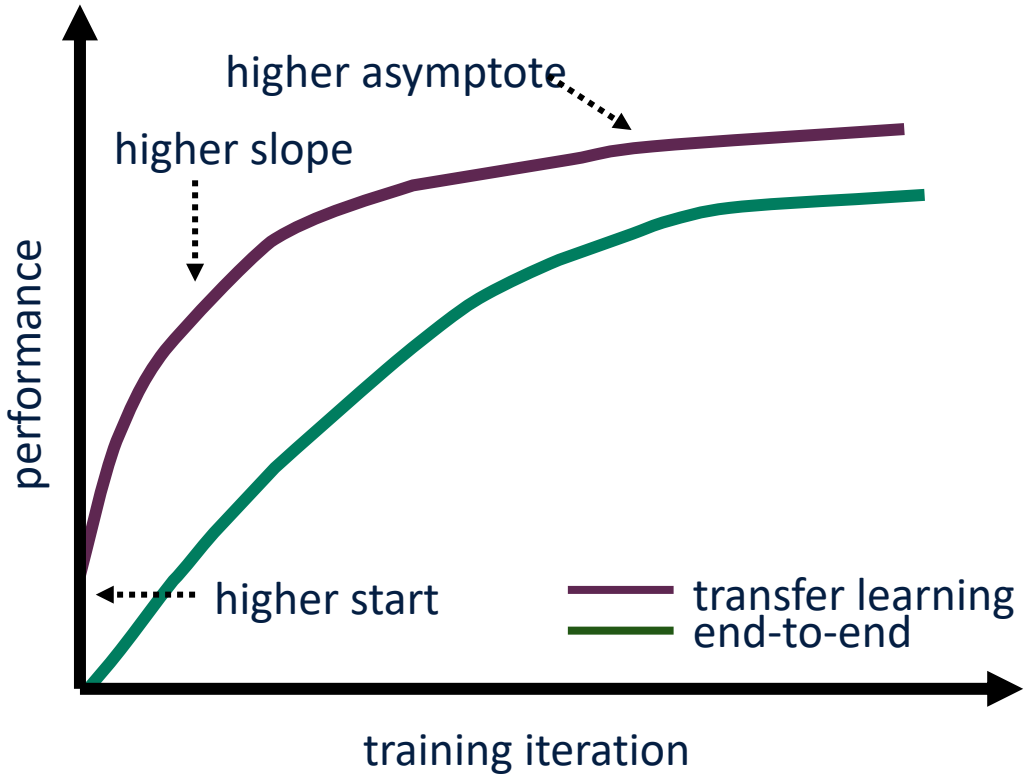
Source task: often a generic task or related task



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

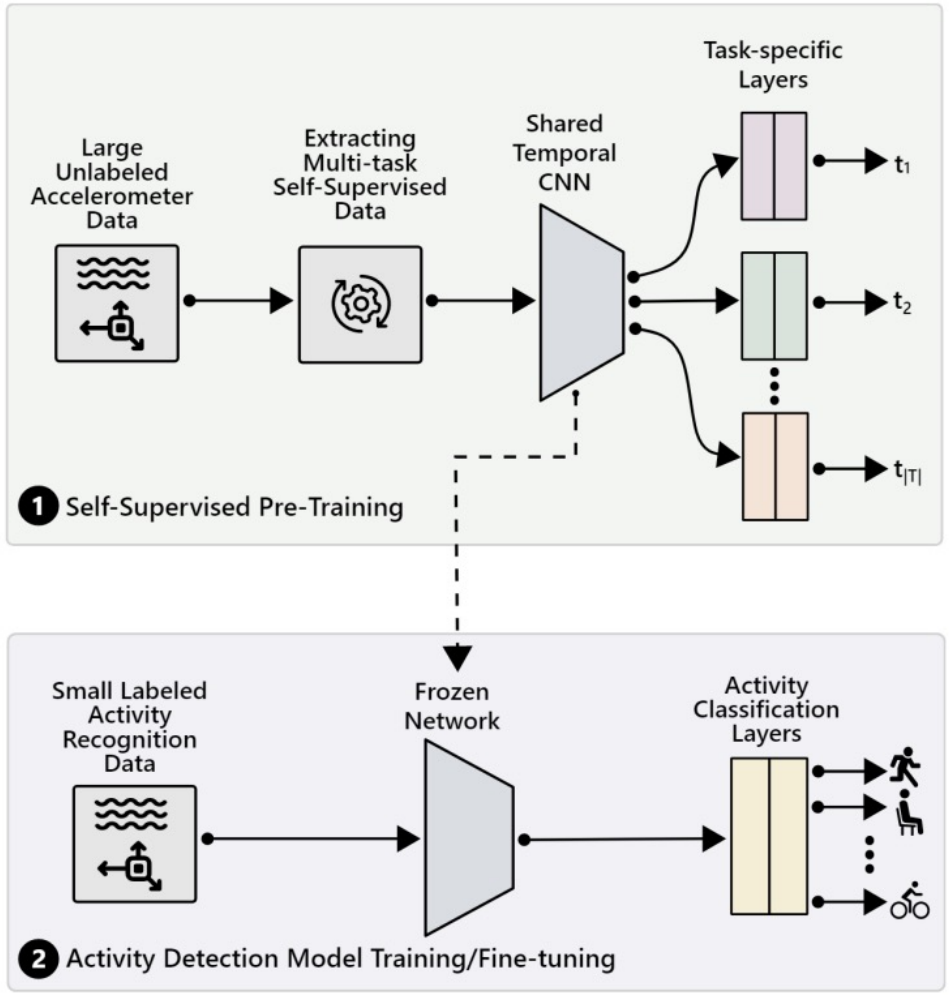


Transfer learning can improve model performance by first learning generalized features on larger and more diverse data, and then fine-tuning towards your target (specific task). It also helps mitigate against overfitting which can be problematic when training a model end-to-end on your smaller dataset.

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

Case Study 1

Human Activity Recognition from Wearable Technologies

Using smartphones and smartwatch accelerometer data to predict daily activities, such as running, walking, cycling, sleeping, etc.

Generating insights from digital technologies

Touch Screen

IMU

Microphone

GPS

Camera

ECG

Step Count

Logic

PPG

Thermometer

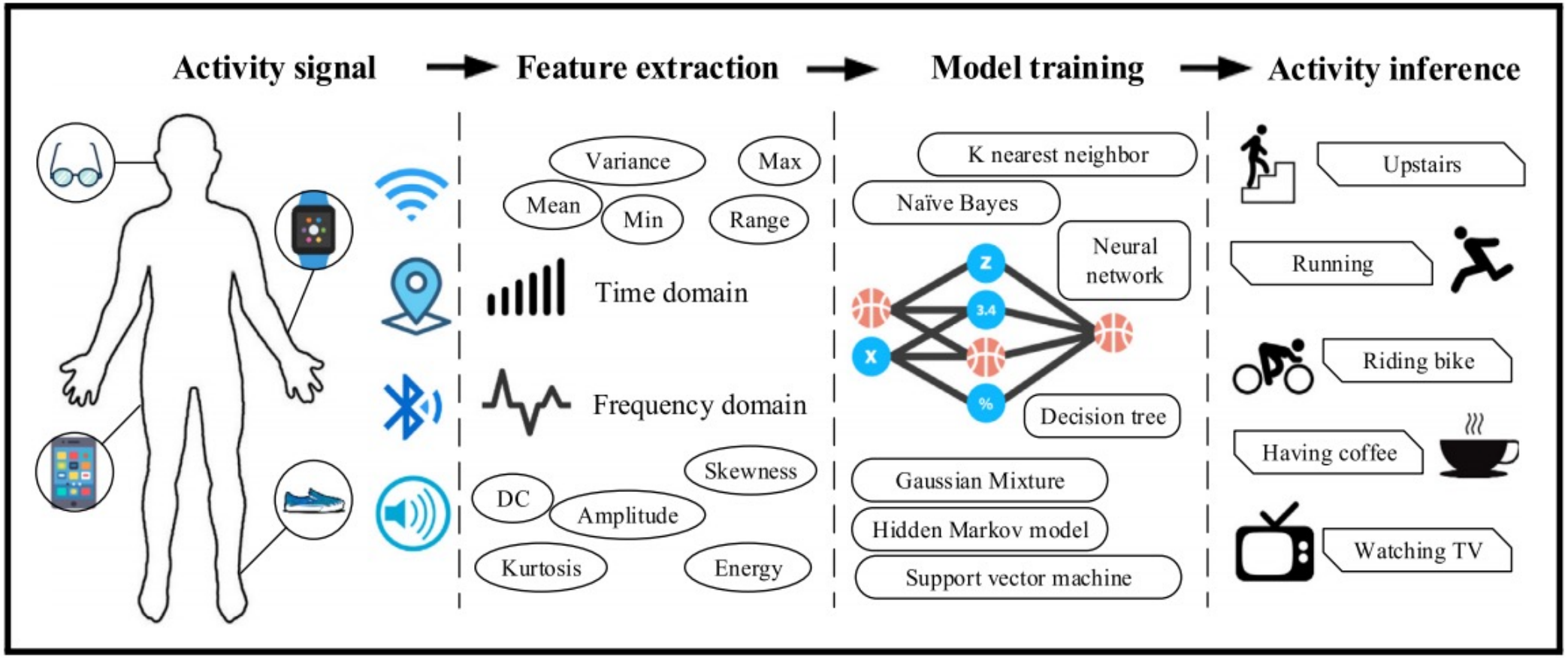
EKG

Social Network

Altimeter

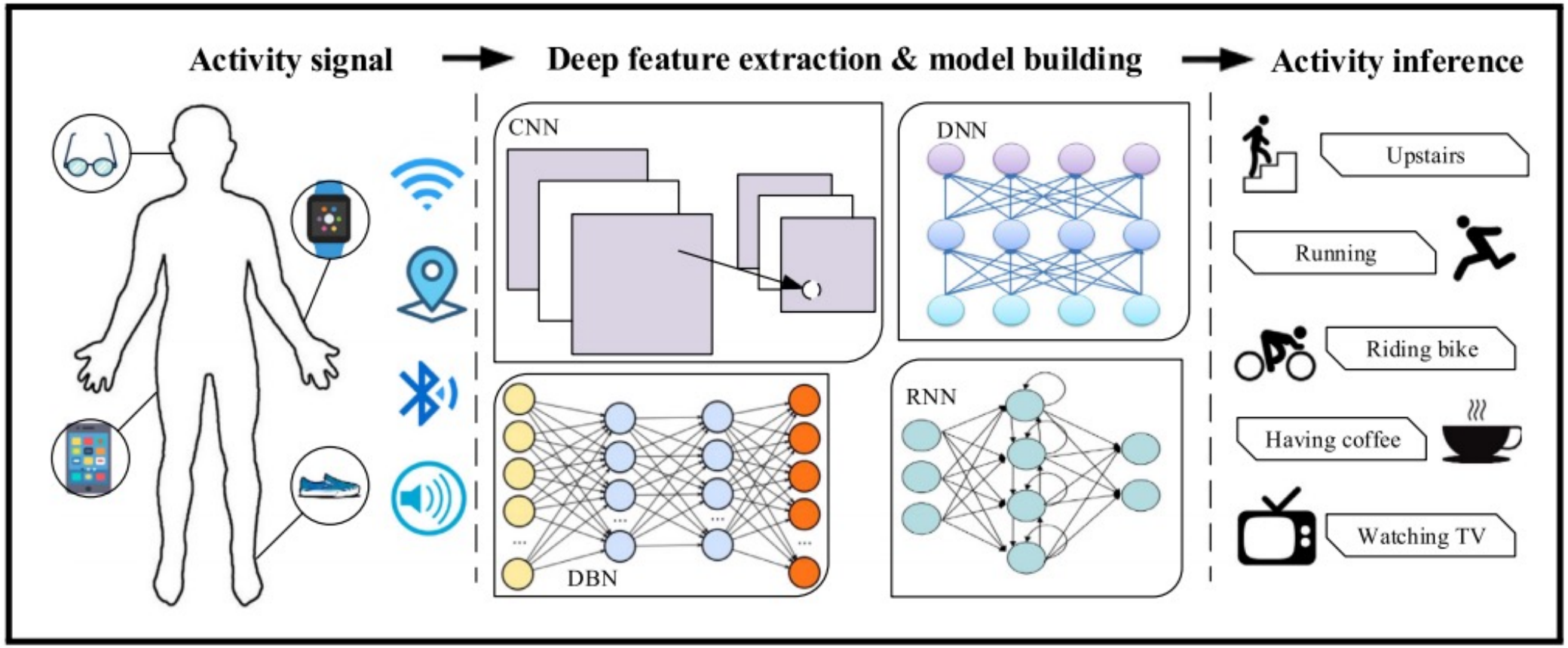
Barometer

Activity Recognition



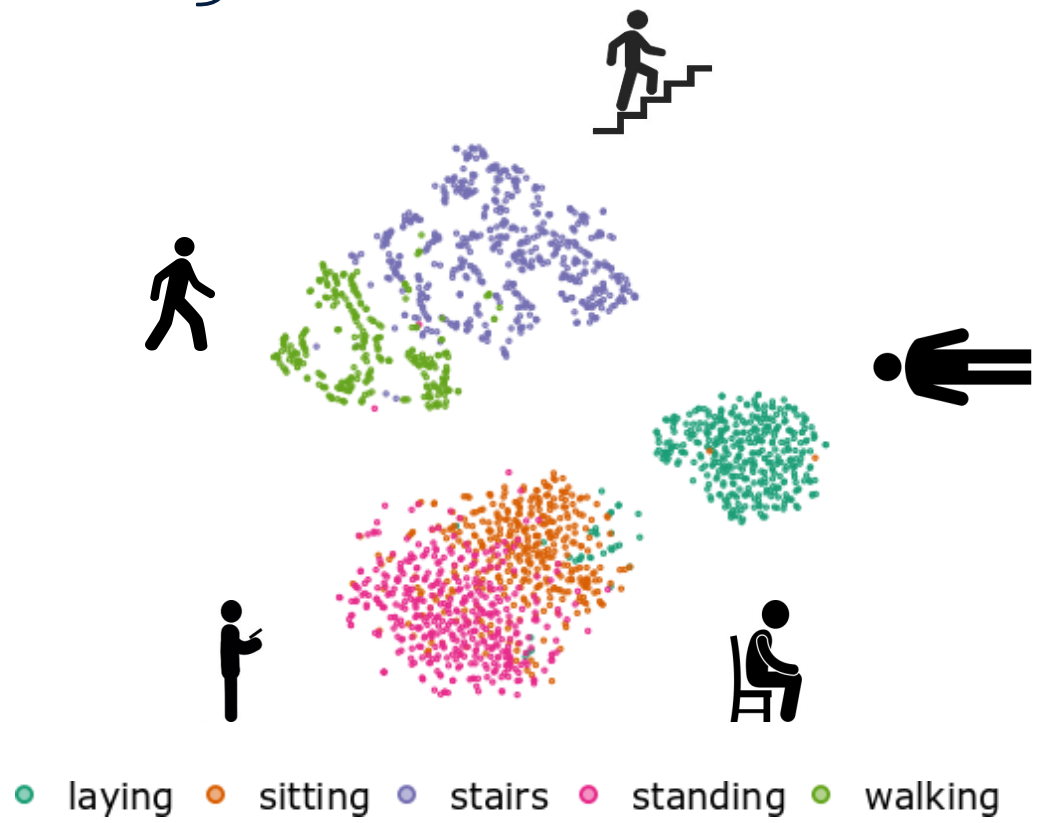
Wang J., *et al.* (2019) *"Deep learning for sensor based activity recognition: A survey,"* Pattern Recognition Letters, vol. 119, pp. 3–11, 2019.

Activity Recognition



Wang J., *et al.* (2019) "*Deep learning for sensor based activity recognition: A survey*," Pattern Recognition Letters, vol. 119, pp. 3–11, 2019.

Deep networks are really good at activity recognition



Actual \ Predicted	laying	sitting	stairs	standing	walking
laying	93.4% (1784)	2.0% (39)	0.05% (1)	4.6% (87)	0% (0)
sitting	10.2% (177)	68.1% (1183)	0.1% (2)	21.6% (376)	0% (0)
stairs	0% (0)	0.04% (1)	98.6% (2745)	0.1% (3)	1.3% (35)
standing	0.05% (1)	7.6% (145)	0.1% (2)	92.3% (1774)	0% (0)
walking	0% (0)	0.2% (4)	4.6% (76)	0% (0)	95.2% (1578)

Cheng, W.-Y., Lipsmeier F., **Creagh, A.P**, *et. al*. (2018). *"Large-Scale Continuous Mobility Monitoring of Parkinson's Disease Patients Using Smartphones",* International Conference on Wireless Mobile Communication and Healthcare.

Creagh, A. P (2020), '*The Development of Digital Biomarkers for Multiple Sclerosis from Remote Smartphone- and Smartwatch-Based Assessments*', DPhil thesis.

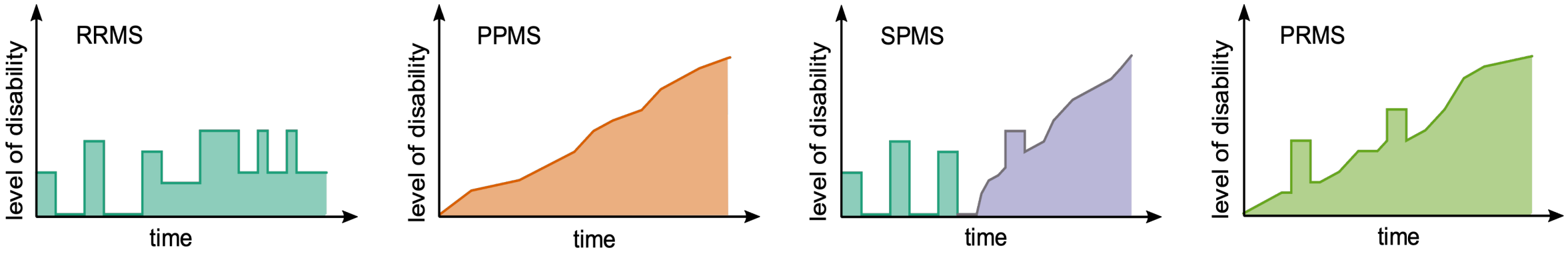
Case Study 2

Digital Biomarkers from Wearable Technologies

Using smartphones and smartwatch accelerometer data to predict daily disease severity of people with multiple sclerosis (PwMS)

Why is it so difficult to track multiple sclerosis disease progression?

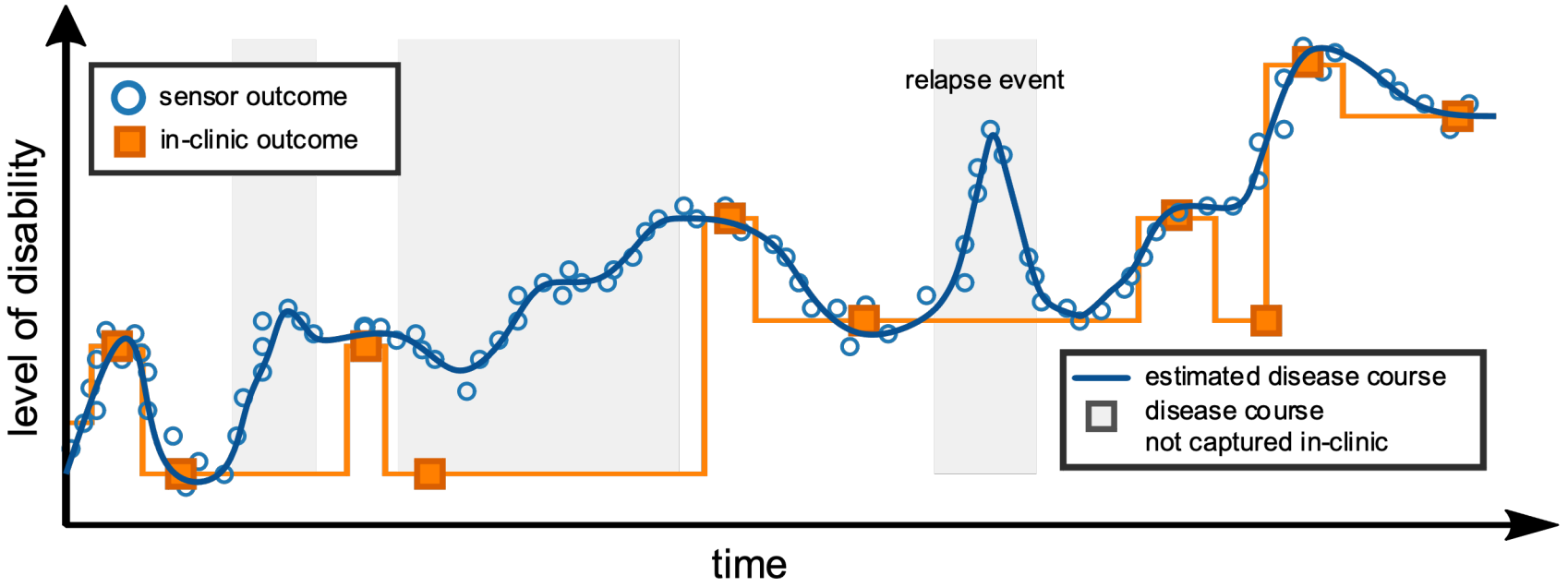
Relapsing-remitting MS (RRMS);
Primary-progressive MS (PPMS);
Secondary-progressive MS (SPMS);
Progressive-relapsing MS (PRMS).

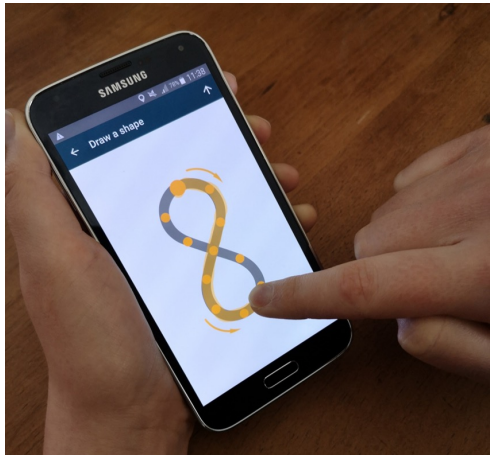


Disease progression is variable for different multiple sclerosis (MS) patients, and over time

Fred D Lublin and Stephen C Reingold. (1996) "Defining the clinical course of multiple sclerosis: results of an international survey". In: *Neurology* 46.4, pp. 907–911.

Can smartphones remotely monitor patients at-home?

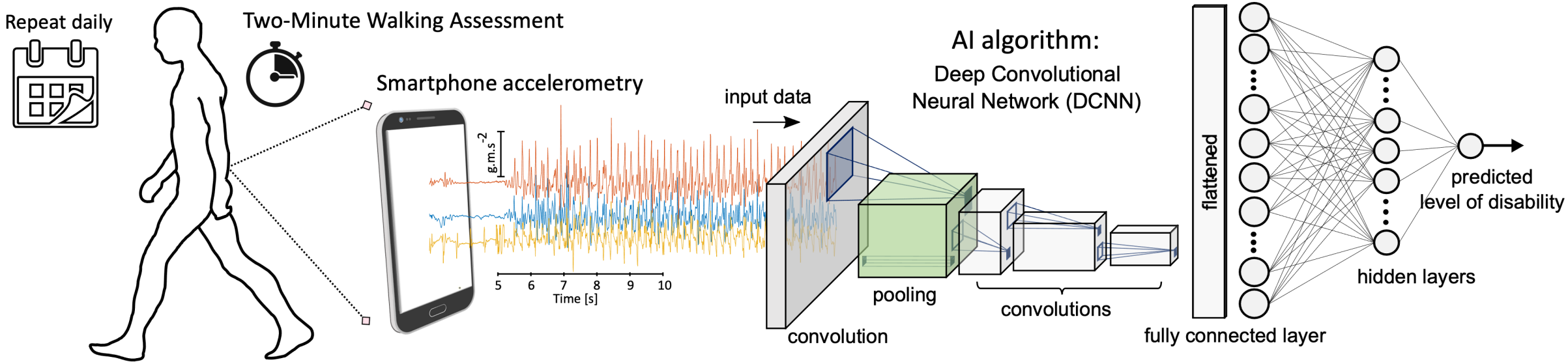




Smartphones and smartwatches can track patient symptoms between clinical visits

Creagh A. P. (2021), "*The Development of Digital Biomarkers for Multiple Sclerosis from Remote Smartphone- and Smartwatch-Based Assessments*". D.Phil. Thesis.

Could AI algorithms learn digital patterns of disease: *digital biomarkers*?

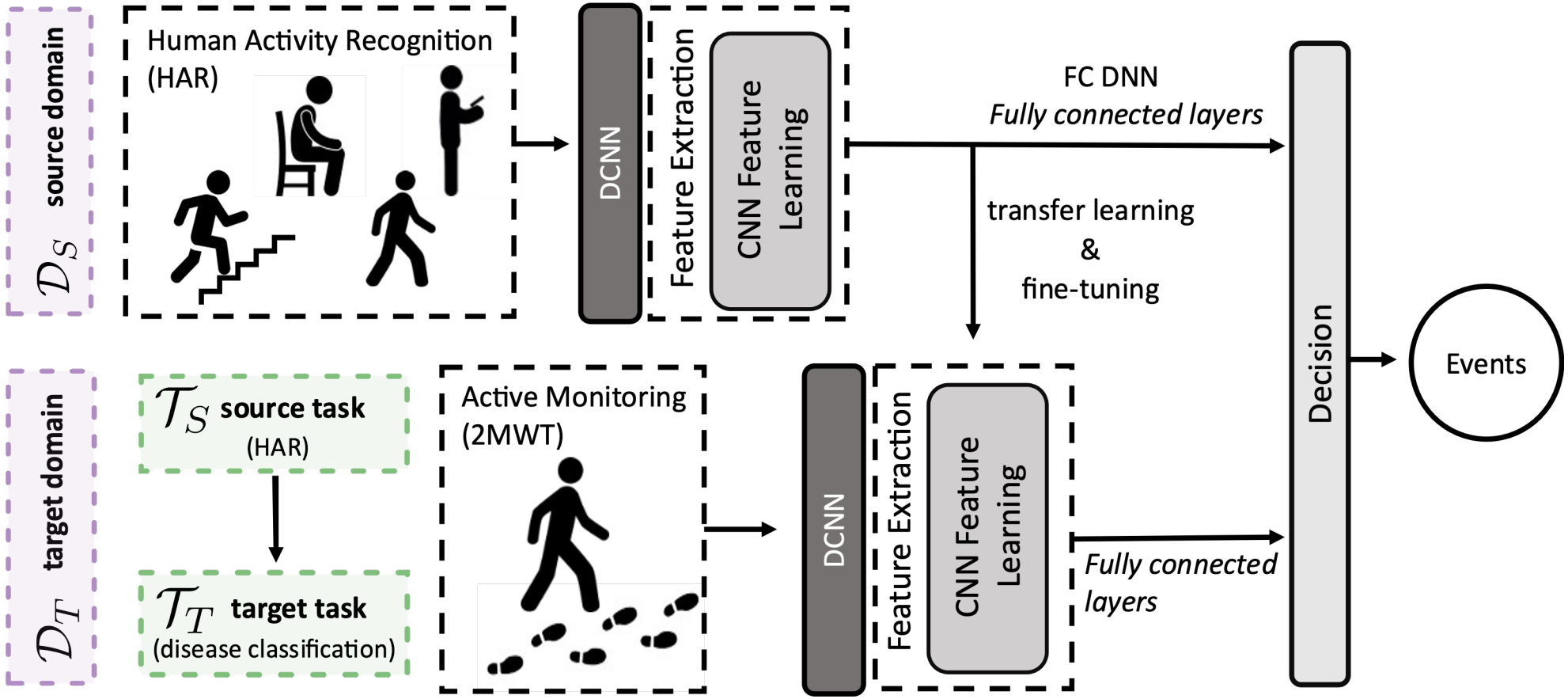


Artificial intelligence (AI) algorithms can transform smartphone measurements to predict MS patient severity

Creagh, A.P., Lipsmeier, F., Lindemann, M. *et al.* Interpretable deep learning for the remote characterisation of ambulation in multiple sclerosis using smartphones. Nature, *Sci Rep* **11,** 14301 (2021). https://doi.org/10.1038/s41598-021-92776-x

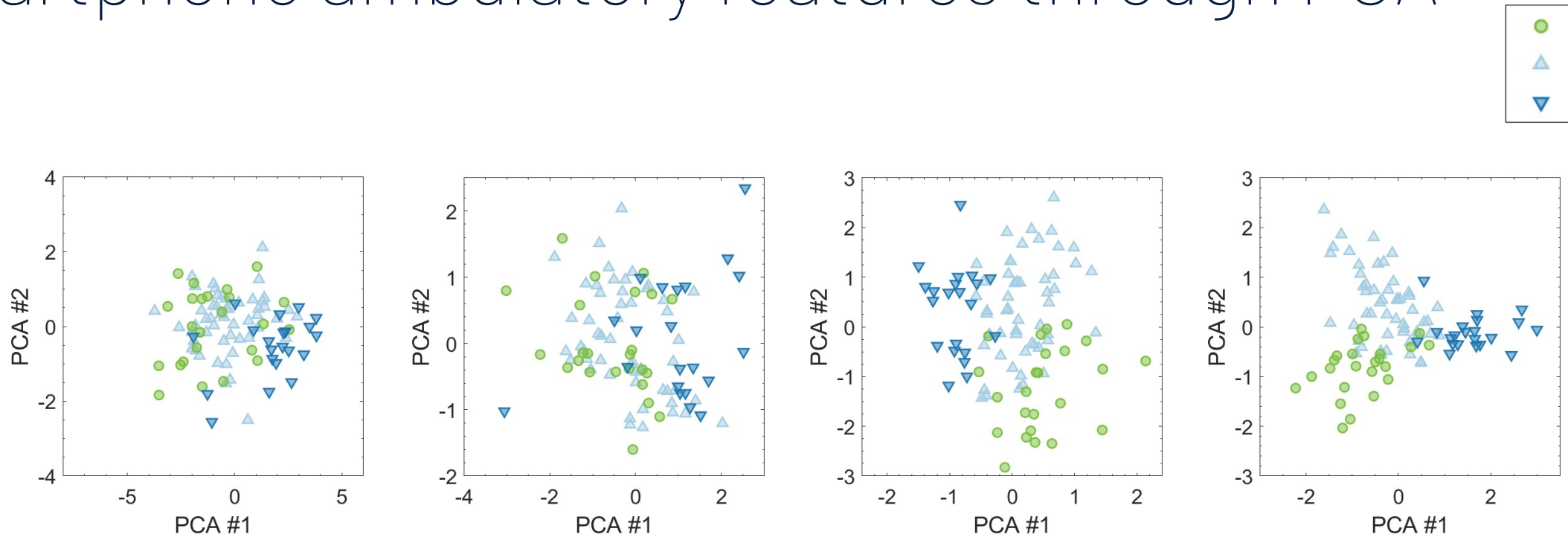
Transfer Learning

DCNN; *Deep Convolutional Neural Network*



Creagh, A.P., Lipsmeier, F., Lindemann, M. *et al.* Interpretable deep learning for the remote characterisation of ambulation in multiple sclerosis using smartphones. Nature, *Sci Rep* **11,** 14301 (2021). https://doi.org/10.1038/s41598-021-92776-x

Visualisation of hand-crafted versus learned smartphone ambulatory features through PCA



(a) Original features (hand-crafted)

(b) DCNN features (end-to-end)

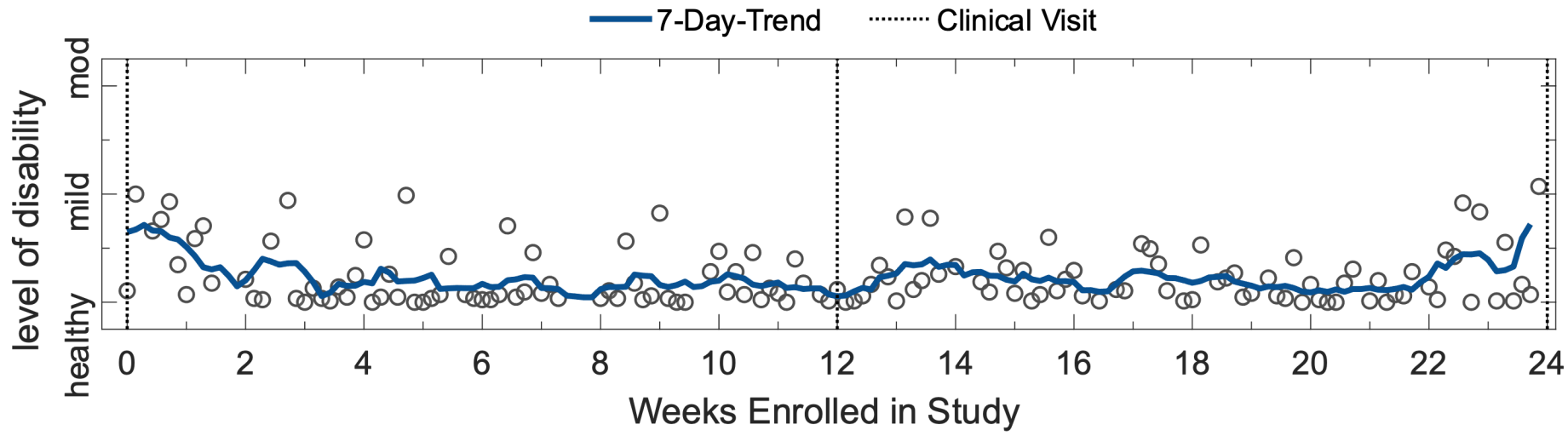
(c) DCNN features (after transfer learning)

(d) Final classification (after transfer learning)

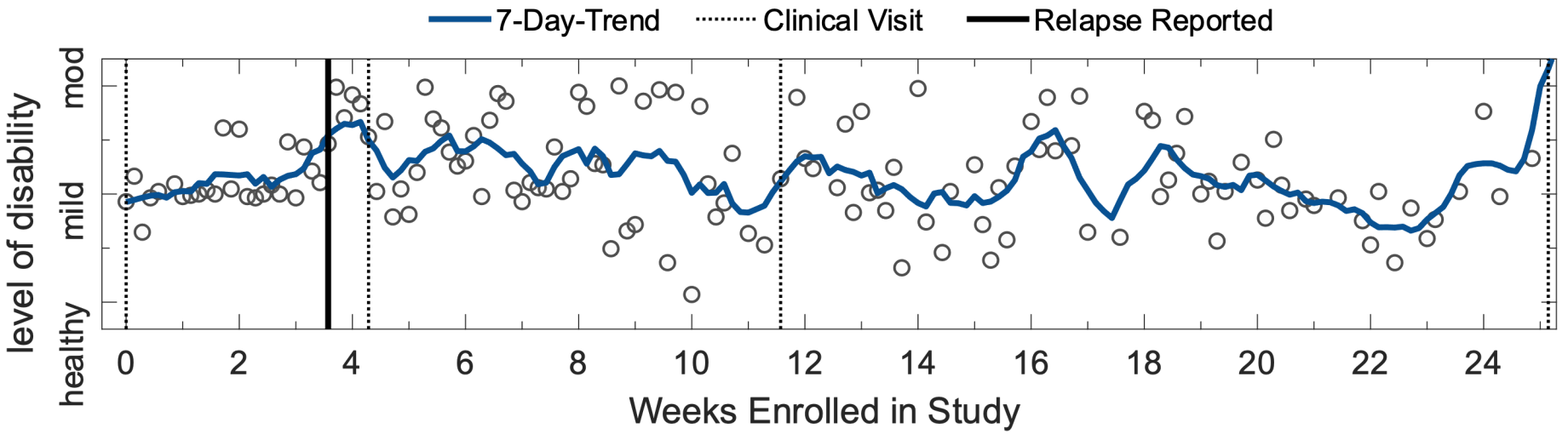
Figure - Example of Principal Component Analysis (PCA)

Creagh, A. P (2020), '*The Development of Digital Biomarkers for Multiple Sclerosis from Remote Smartphone- and Smartwatch-Based Assessments*', DPhil thesis.

Smartphone measurements accurately predict MS symptom severity over 6 months



Healthy study participant



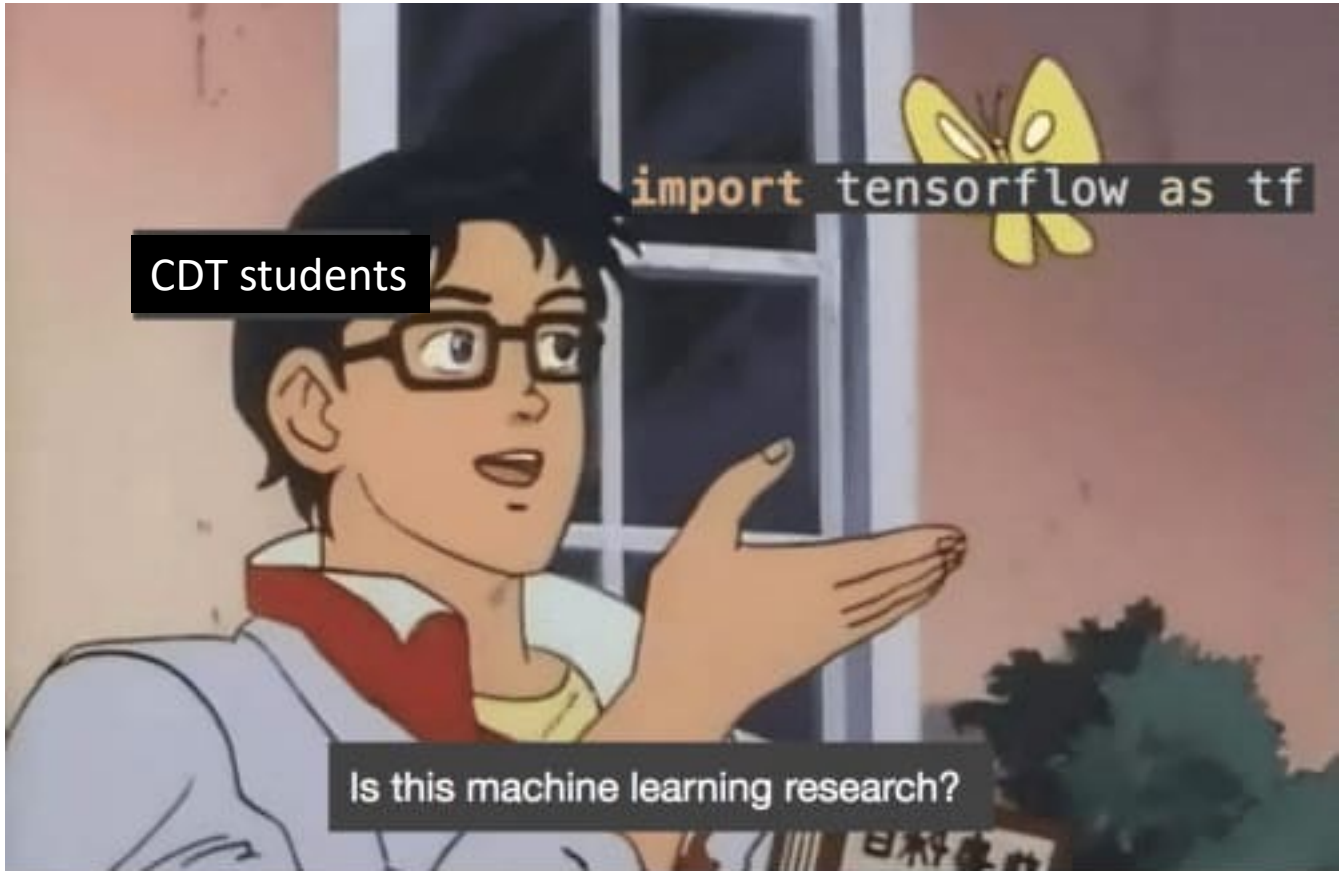
Relapsing mildly disabled MS study participant

ML 4 Time-series: Convolutional Neural Networks

What have we learned?

- 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