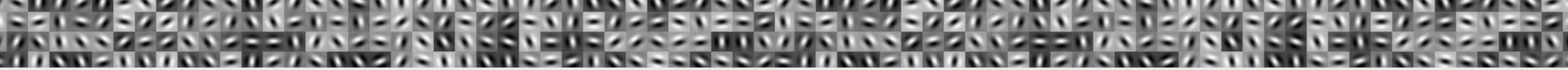


Deep Learning

Men-Andrin Meier, Noah Luna, Lion Krischer & Team

Munich Earth Skience School, February 2019

Basically, it's very simple …

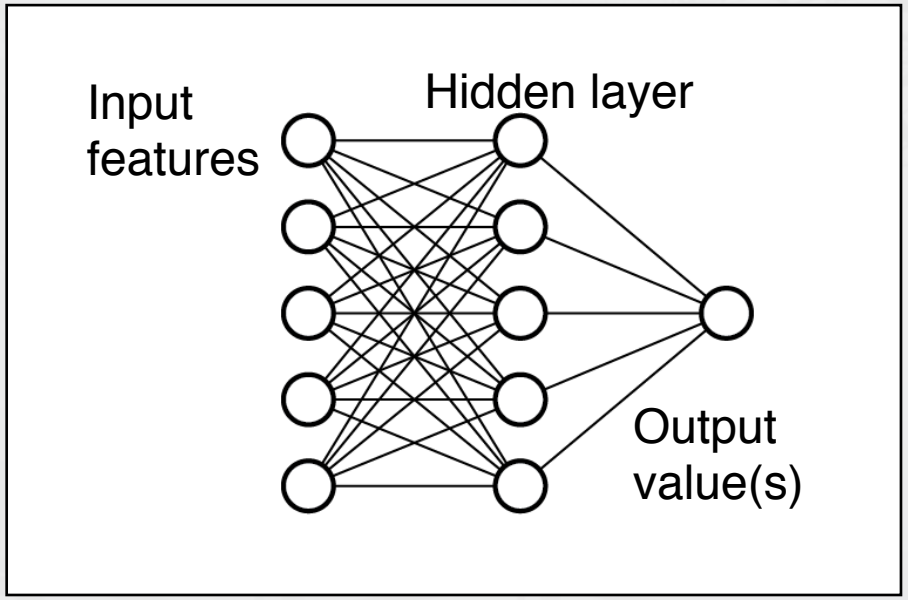
Machine Learning is …

- a class of powerful algorithms that can *"learn information directly from data without relying on a predetermined equation as a model"*

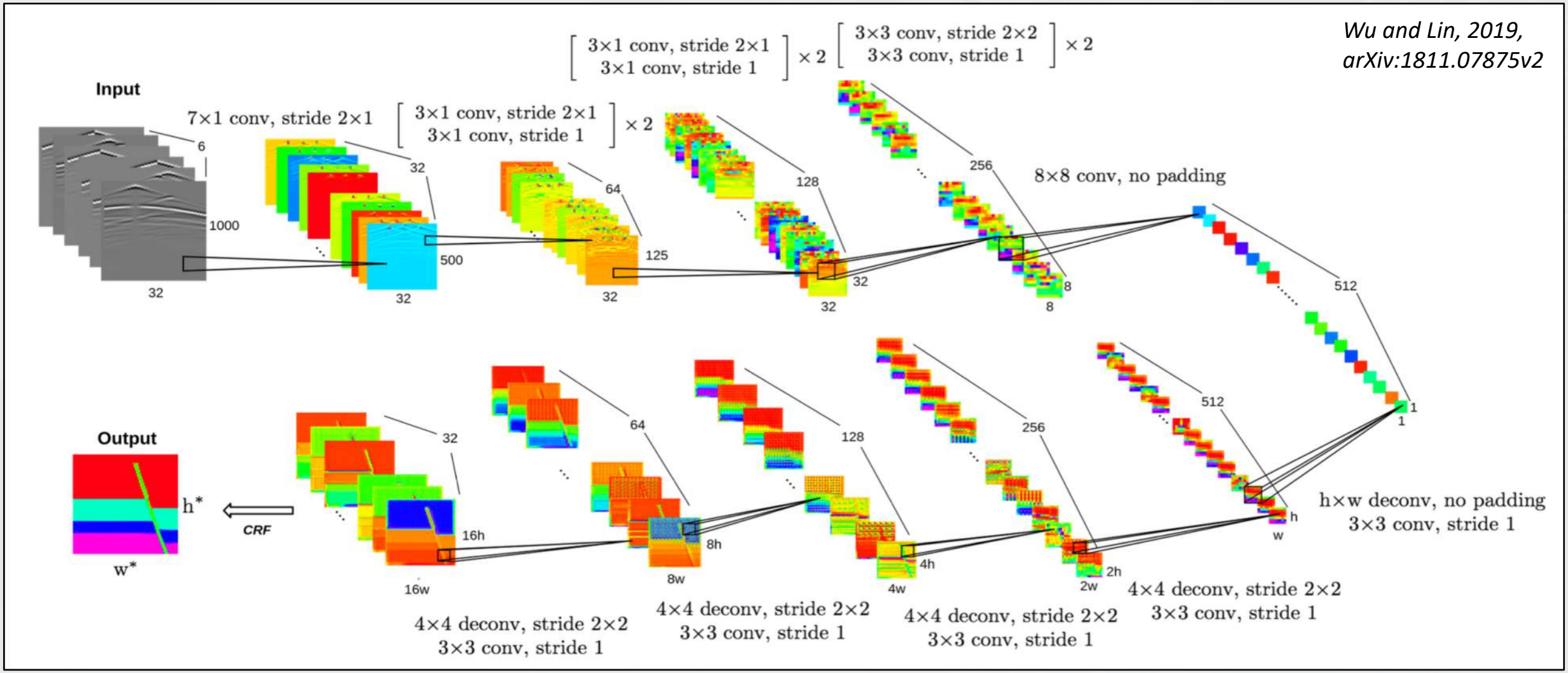
Deep Learning …

- uses deeper models, with more layers
- works on raw input data directly, rather than on features

From this ...

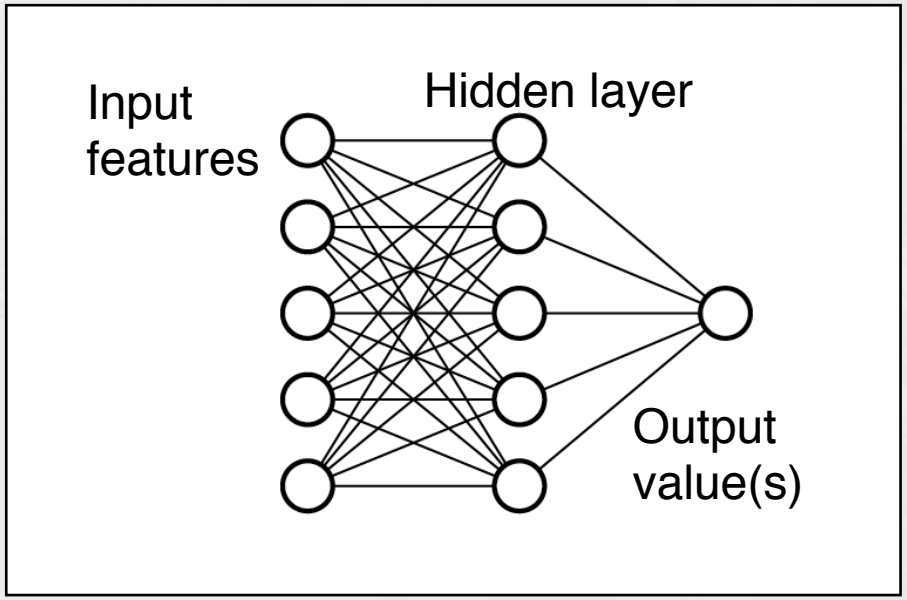


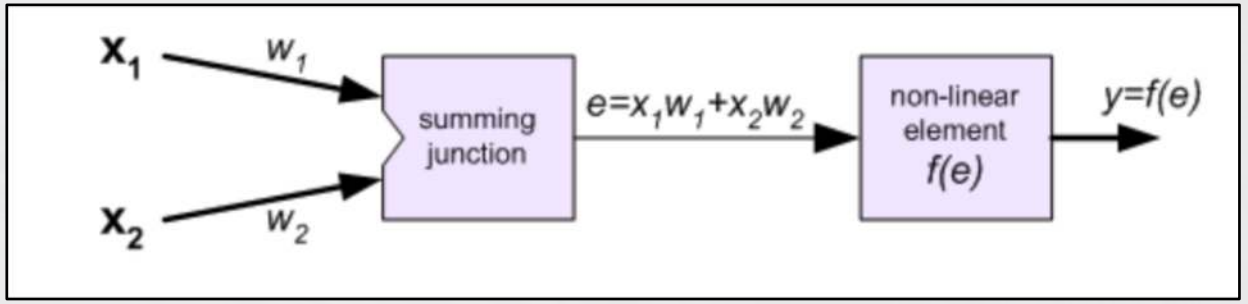
... to this:


Wu and Lin, 2019,
arXiv:1811.07875v2
Input
7×1 conv, stride 2×1
[3×1 conv, stride 2×1 / 3×1 conv, stride 1] × 2
[3×3 conv, stride 2×2 / 3×3 conv, stride 1] × 2
8×8 conv, no padding
6
1000
32
32
500
32
64
125
32
128
32
256
8
8
512
1
1
Output
h*
w*
CRF
32
16h
16w
64
8h
8w
128
4h
4w
256
2h
2w
512
h
w
4×4 deconv, stride 2×2
3×3 conv, stride 1
4×4 deconv, stride 2×2
3×3 conv, stride 1
4×4 deconv, stride 2×2
3×3 conv, stride 1
4×4 deconv, stride 2×2
3×3 conv, stride 1
h×w deconv, no padding
3×3 conv, stride 1

Artificial Neural Networks (recap)

- Network of simple operations = non-linear mapping function
- Training: Optimize weights to obtain maximum performance across large data set
- Back-propagation of error function; stochastic gradient descent
- Generic systems capable of discovering complex non-linear relationships between input features and output values





Universal Approximation Theorem

If we had enough data and computation power, we could brute-force approximate *any* mapping function with a three layer FCNN

Often not feasible in practice, and there are much more efficient/elegant solutions

The "art" of Deep Learning is to design complex, but trainable models, e.g. with

- Convolutional Neural Networks
- Recurrent Neural Networks
- Generative Networks
- …

Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function. Let I_m denote the m-dimensional unit hypercube $[0, 1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that we may define:

$$F(x) = \sum_{i=1}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

as an approximate realization of the function f; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

Overview

- Morning: Convolutional Neural Networks
  - How ConvNets work
  - Building and training a convnet
  - Applications of ConvNets in seismology
- Stochastic Gradient Descent in Practice (Skiing!)
- Afternoon I: Recurrent Neural Networks
- Afternoon II: Generative Models

How to teach a computer to recognise cats?

House cat

Yoga cat

Actually, visual recognition is difficult …

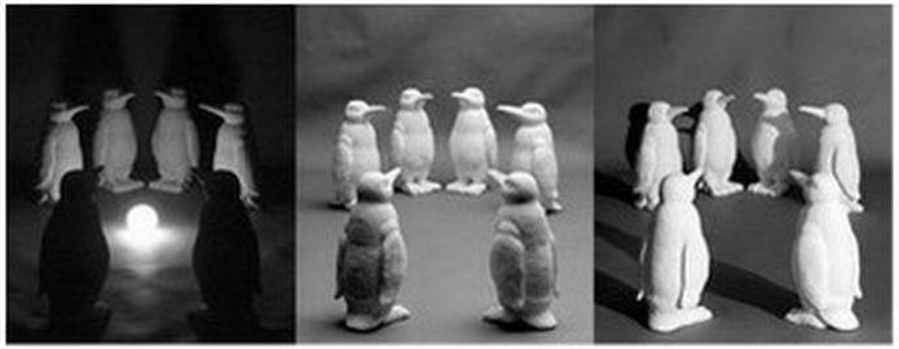
Viewpoint variation

Scale variation

Deformation

Occlusion

Illumination conditions



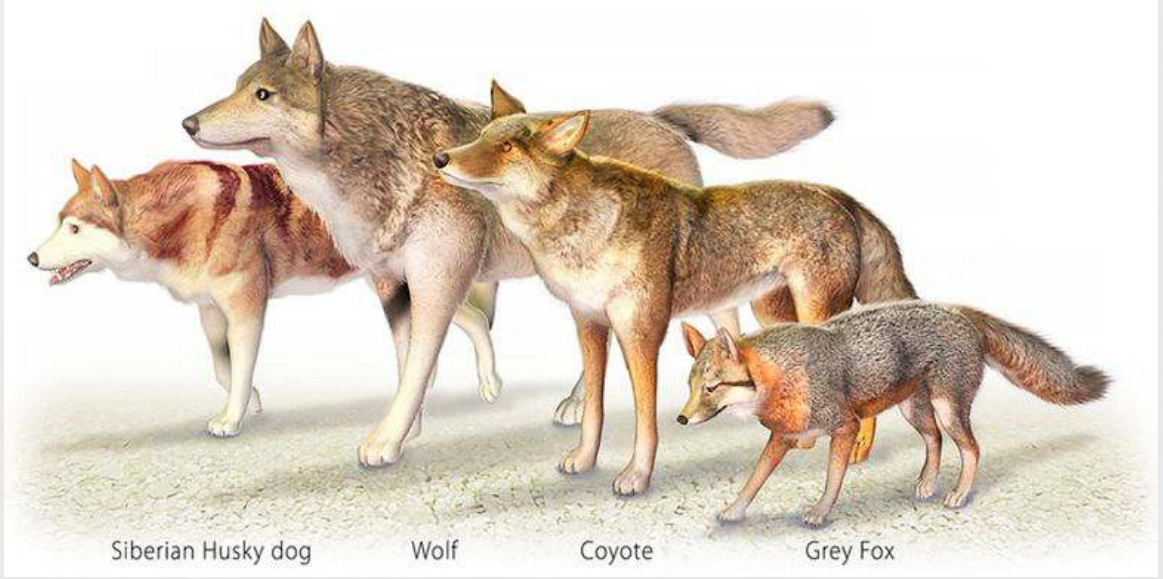
Background clutter

Intra-class variation

Shallow models struggle with „selectivity-invariance dilemma“

„ … problems such as image and speech recognition require the [model] to be insensitive to irrelevant variations of the input, such as variations in position, orientation or illumination of an object, or variations in the pitch or accent of speech, while being very sensitive to particular minute variations (for example, the difference between a white wolf and a breed of wolf-like white dog)„

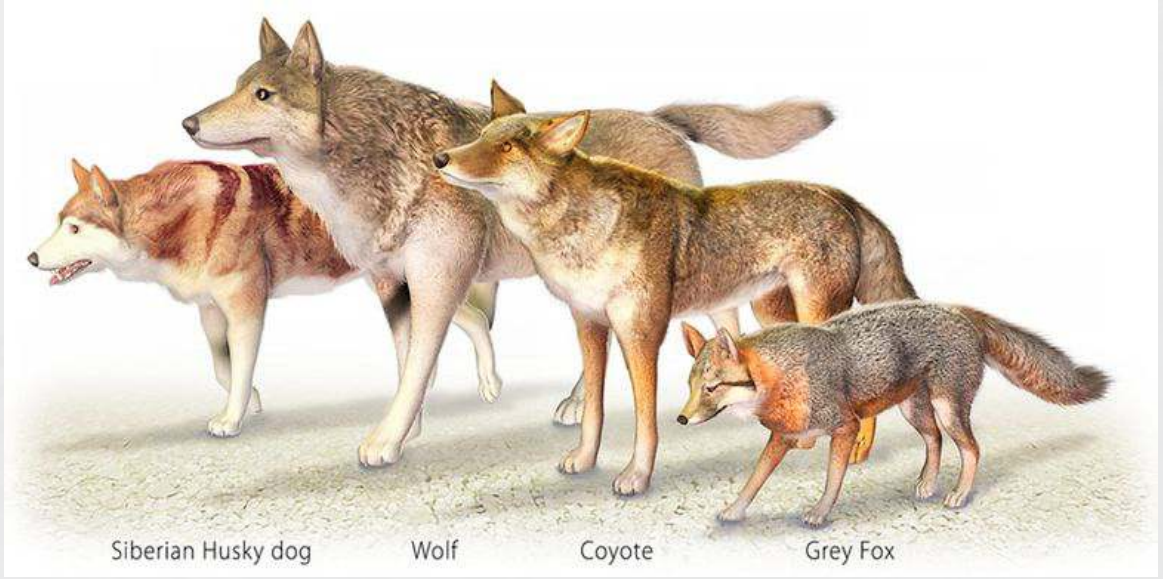
LeCun et al., 2015, Nature, “Deep Learning”



Deep models can handle it

„A deep-learning architecture is a multilayer stack of simple modules [...]. Each module in the stack transforms its input to increase both the selectivity and the invariance of the representation. With multiple non-linear layers, say a depth of 5 to 20, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details – distinguishing [dogs and wolves] – and insensitive to large irrelevant variations such as the background, pose, lighting and surrounding objects."

LeCun et al., 2015, Nature, "Deep Learning"



AlexNet: Breakthrough for Visual Recognition

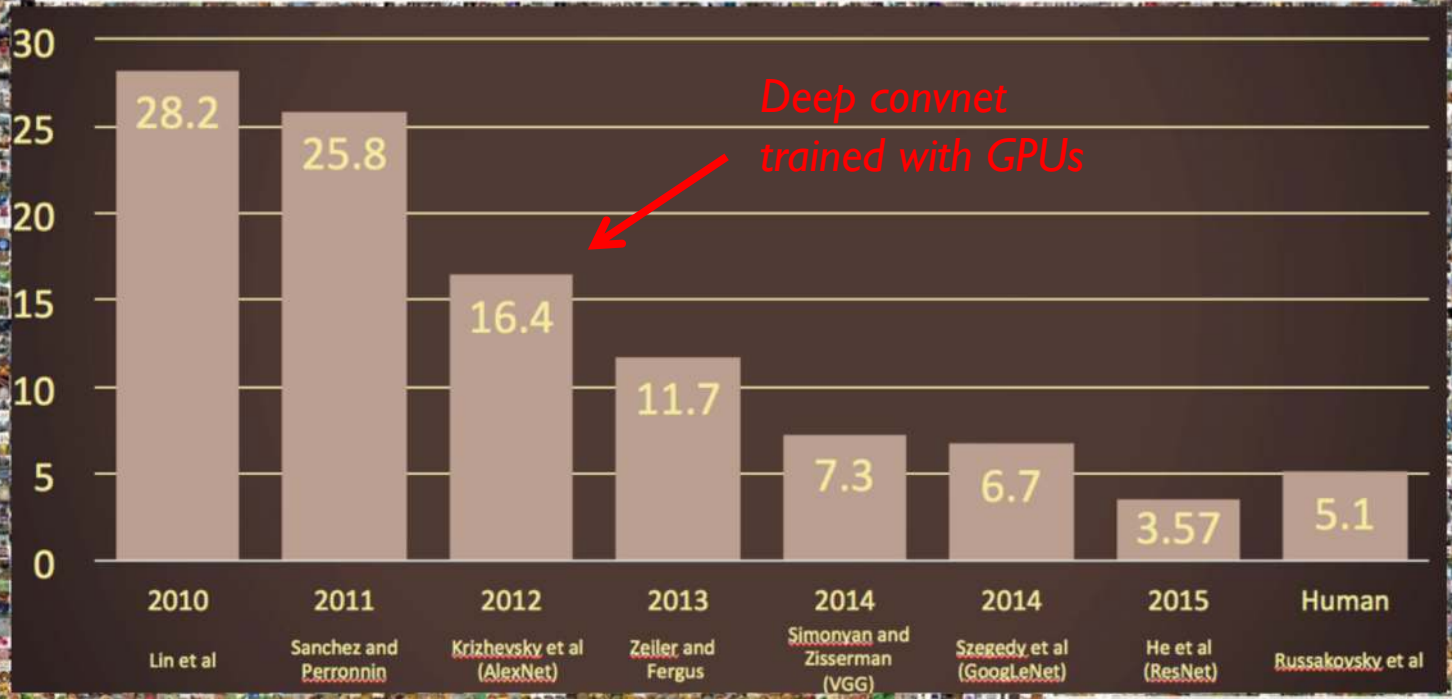
Slide from Stanford's cs231n class

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

1,000 object classes

1,431,167 images



Russakovsky et al. arXiv, 2014

Slide from Stanford's cs231n class

Convolutional Neural Networks

Overview

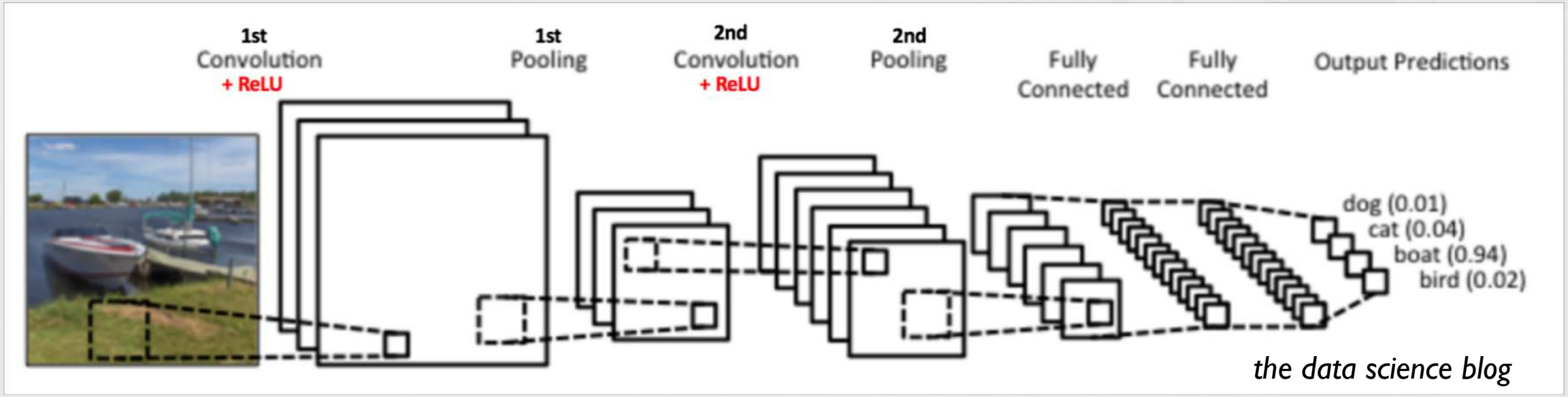
Principles & Properties

Optimisation / Training

Applications

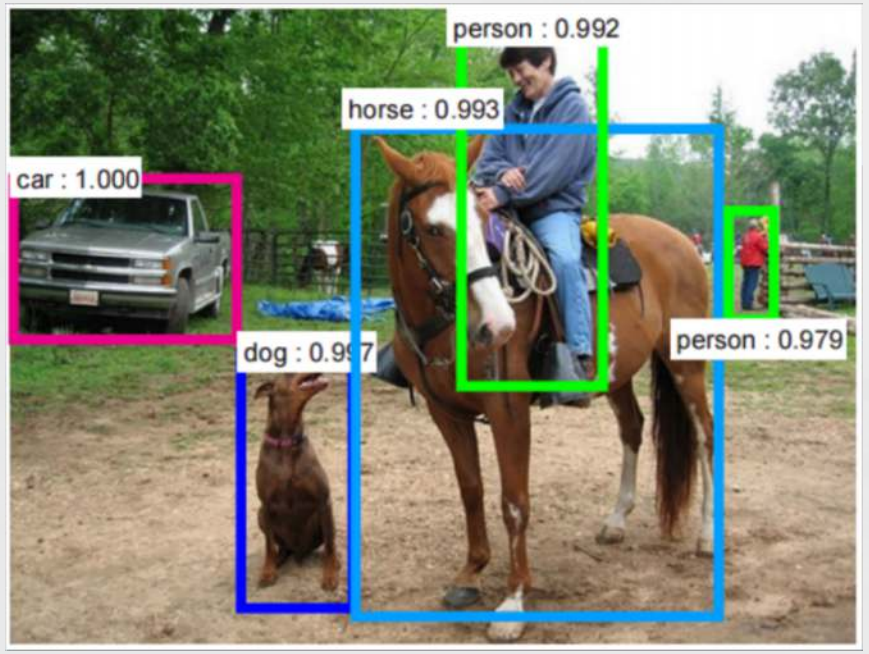
Convolutional Neural Networks

- Combine fully connected NN with convolutional layers
- Convolutional layers contain numerous parallel and sequential digital filters
- Filters transform raw input data (e.g. bitmap image or waveform) into feature vector
- Filter coefficients themselves are learned during training → network learns to find relevant features
- Between convolutions, output is down-sampled → scale-invariant object detection



Convolutional Neural Networks

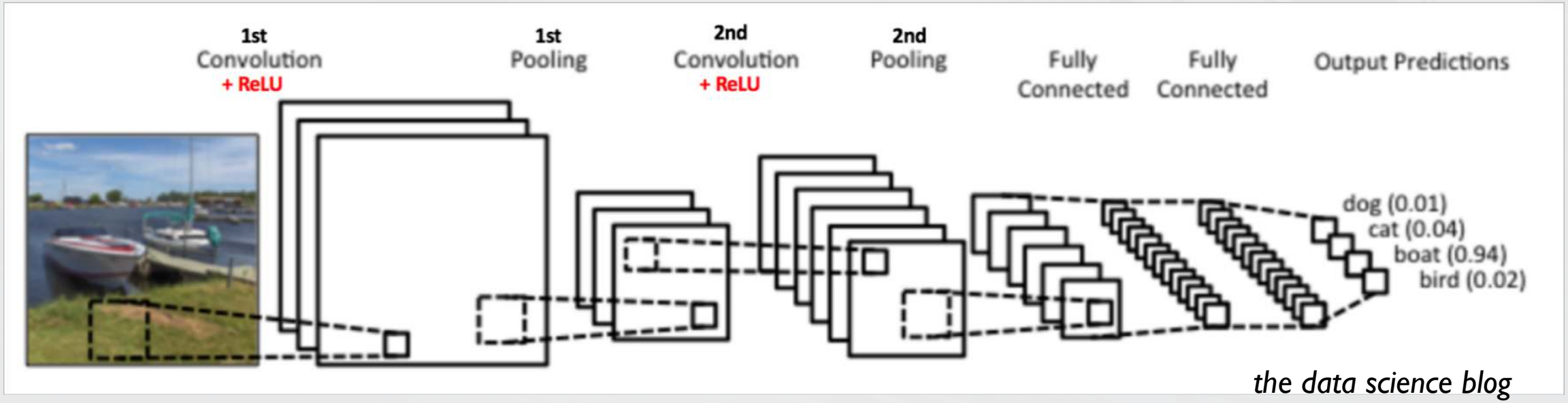
- Combine fully connected NN with convolutional layers
- Convolutional layers contain numerous parallel and sequential digital filters
- Filters transform raw input data (e.g. bitmap image or waveform) into feature vector
- Filter coefficients themselves are learned during training → network learns to find relevant features
- Between convolutions, output is down-sampled → scale-invariant object detection



the data science blog

Convolutional Neural Networks

- Excellent at invariant pattern recognition (e.g. translation, reflection, distortion)
- Powerful for regression and classification tasks
- Major limitation is large amounts of necessary labeled data samples



Convolutional Neural Networks

Overview

Principles & Properties

Optimisation / Training

Applications

Convolutional Neural Networks

Overview

Principles & Properties

- Convolution operation
- Automated feature extraction through learnable filters
- Efficiency through weight sharing
- Hierarchical data representation

Optimisation / Training

Applications

Convolution Operation

$$y_n = x_n * h_n = \sum_{k=0}^{n} x_k h_{n-k},$$

Input signal $x_0, x_1, x_2, x_3, x_4, x_5$

Filter $h_0, h_1, h_2.$

$$\begin{aligned}
y_0 &= x_0 h_0 \\
y_1 &= x_0 h_1 + x_1 h_0 \\
y_2 &= x_0 h_2 + x_1 h_1 + x_2 h_0 \\
y_3 &= x_1 h_2 + x_2 h_1 + x_3 h_0 \\
y_4 &= x_2 h_2 + x_3 h_1 + x_4 h_0 \\
y_5 &= x_3 h_2 + x_4 h_1 + x_5 h_0 \\
y_6 &= x_4 h_2 + x_5 h_1 \\
y_7 &= x_5 h_2
\end{aligned}$$

- Convolution is a linear, differentiable operation

$$\mathbf{y} = \mathbf{H}\mathbf{x}.$$

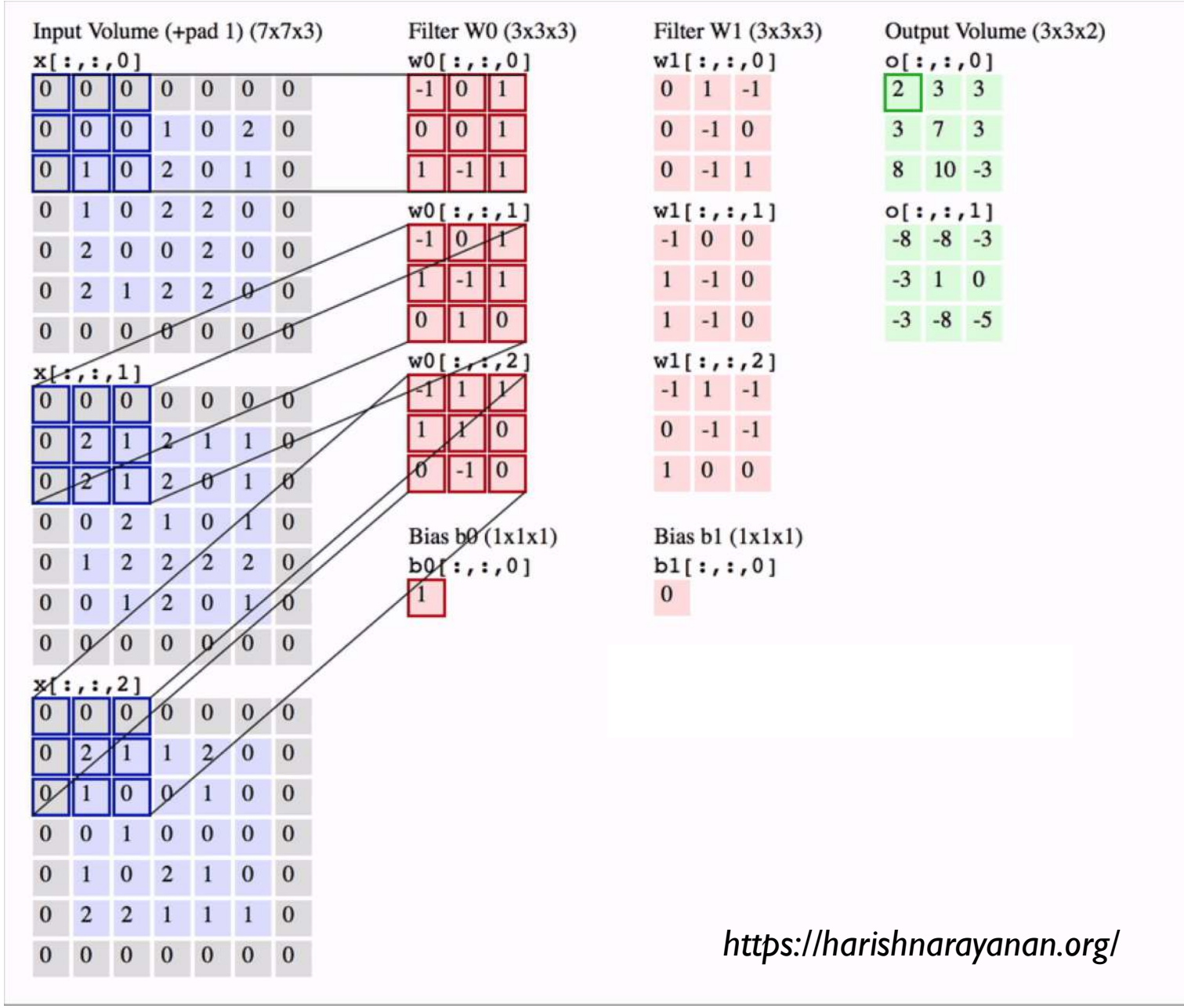
$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} h_0 & 0 & 0 & 0 & 0 & 0 \\ h_1 & h_0 & 0 & 0 & 0 & 0 \\ h_2 & h_1 & h_0 & 0 & 0 & 0 \\ 0 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & h_2 & h_1 & h_0 & 0 \\ 0 & 0 & 0 & h_2 & h_1 & h_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

Convolution Operation

$$y_n = x_n * h_n = \sum_{k=0}^{n} x_k h_{n-k},$$

Input signal $x_0, x_1, x_2, x_3, x_4, x_5$

Filter $h_0, h_1, h_2.$



https://harishnarayanan.org/

Convolution Operation

$$y_n = x_n * h_n = \sum_{k=0}^{n} x_k h_{n-k},$$

Input signal $x_0, x_1, x_2, x_3, x_4, x_5$

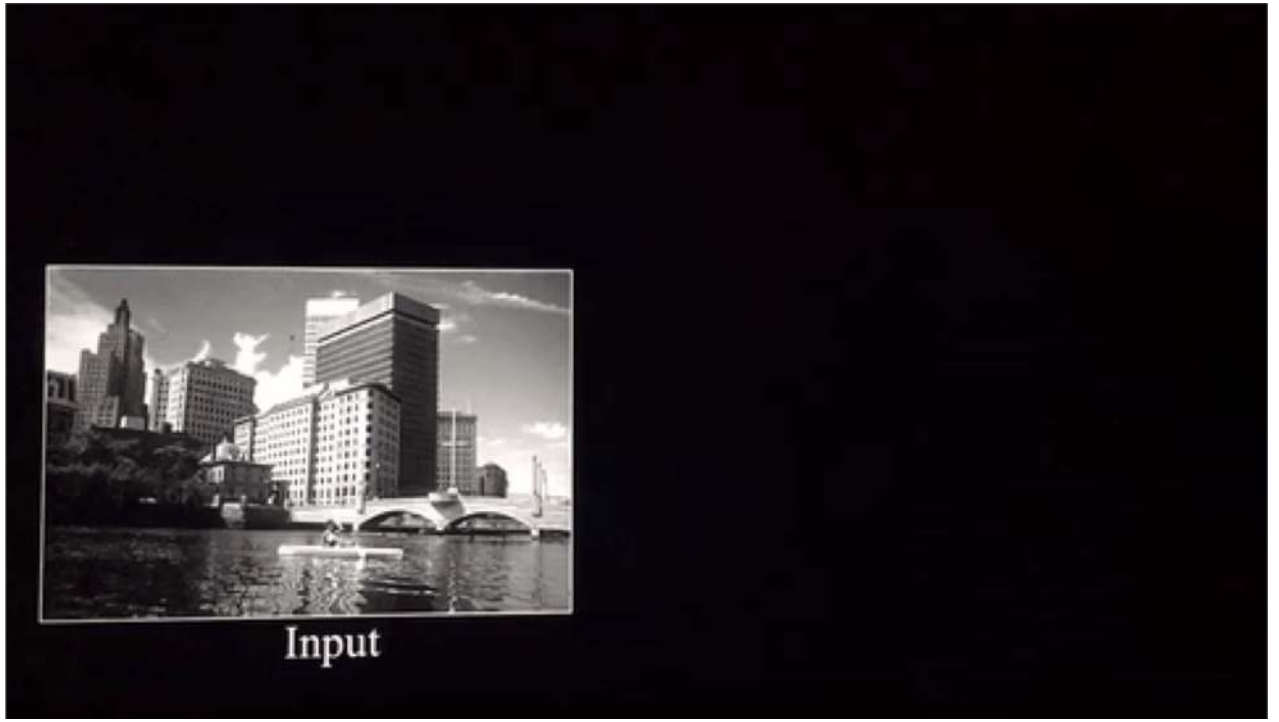
Filter $h_0, h_1, h_2.$

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Convolution Operation

$$y_n = x_n * h_n = \sum_{k=0}^{n} x_k h_{n-k},$$

Input signal	$x_0, x_1, x_2, x_3, x_4, x_5$
Filter	$h_0, h_1, h_2.$



the data science blog

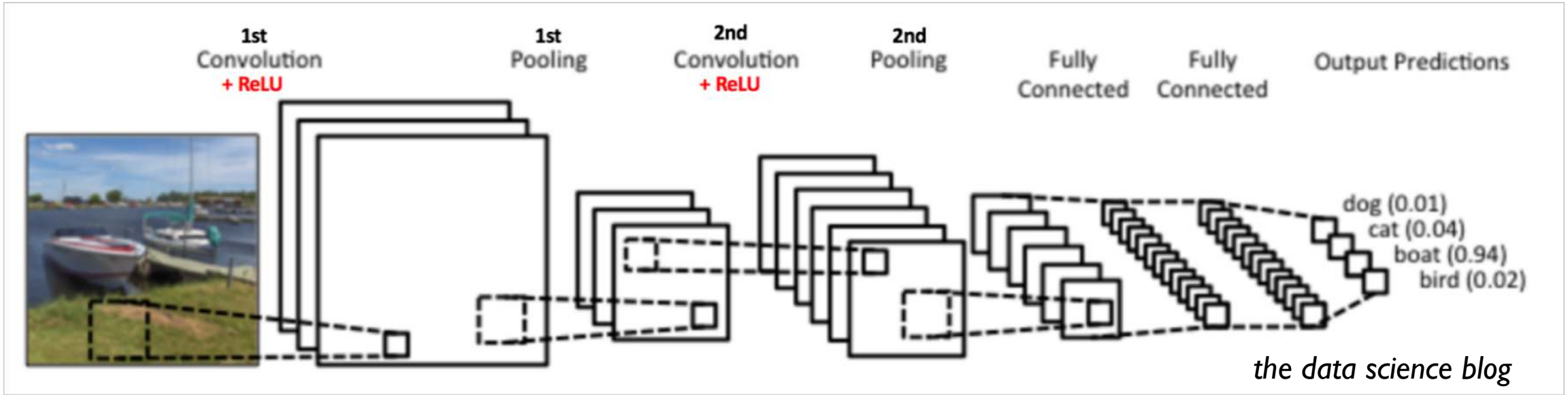
Filters of ConvNets are learned during training

- Filters are characterized by their scalar coefficients
- These coefficients are optimized during training
- The ConvNet *learns what features are relevant* for the target task
- *Automated feature extraction system* (wanted since 1950!)

$$y_n = x_n * h_n = \sum_{k=0}^{n} x_k h_{n-k},$$

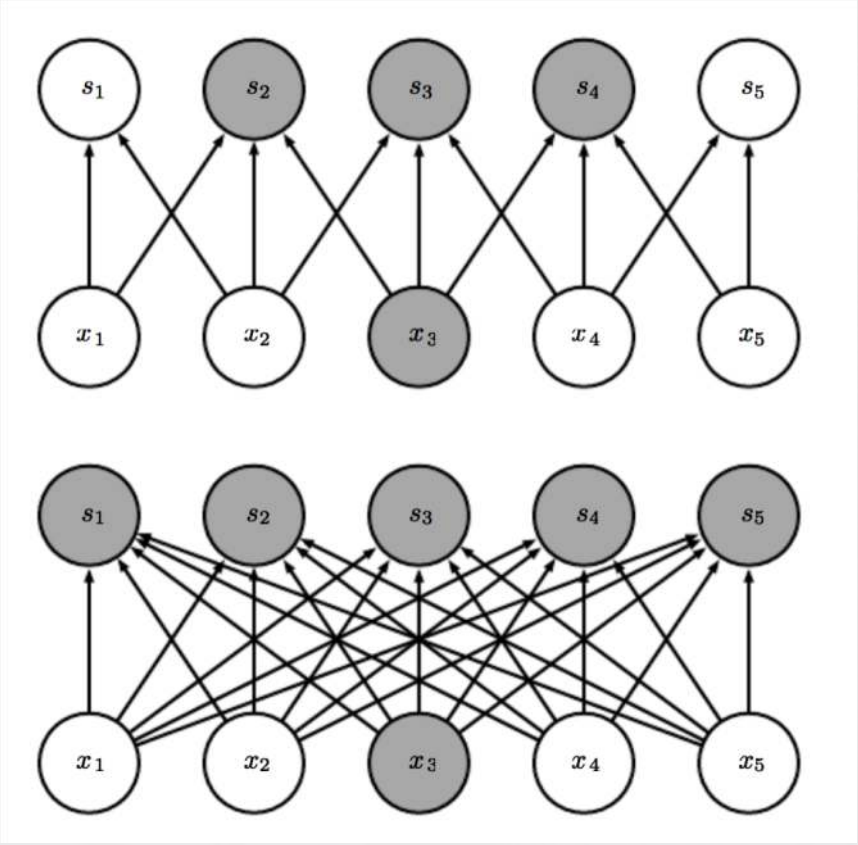
Input signal $x_0, x_1, x_2, x_3, x_4, x_5$

Filter $h_0, h_1, h_2.$



Local connectivity & weight sharing

- Convolutional neurons are only locally connected
- The same filter is used for all positions in an image
- This dramatically reduces the number of free parameters

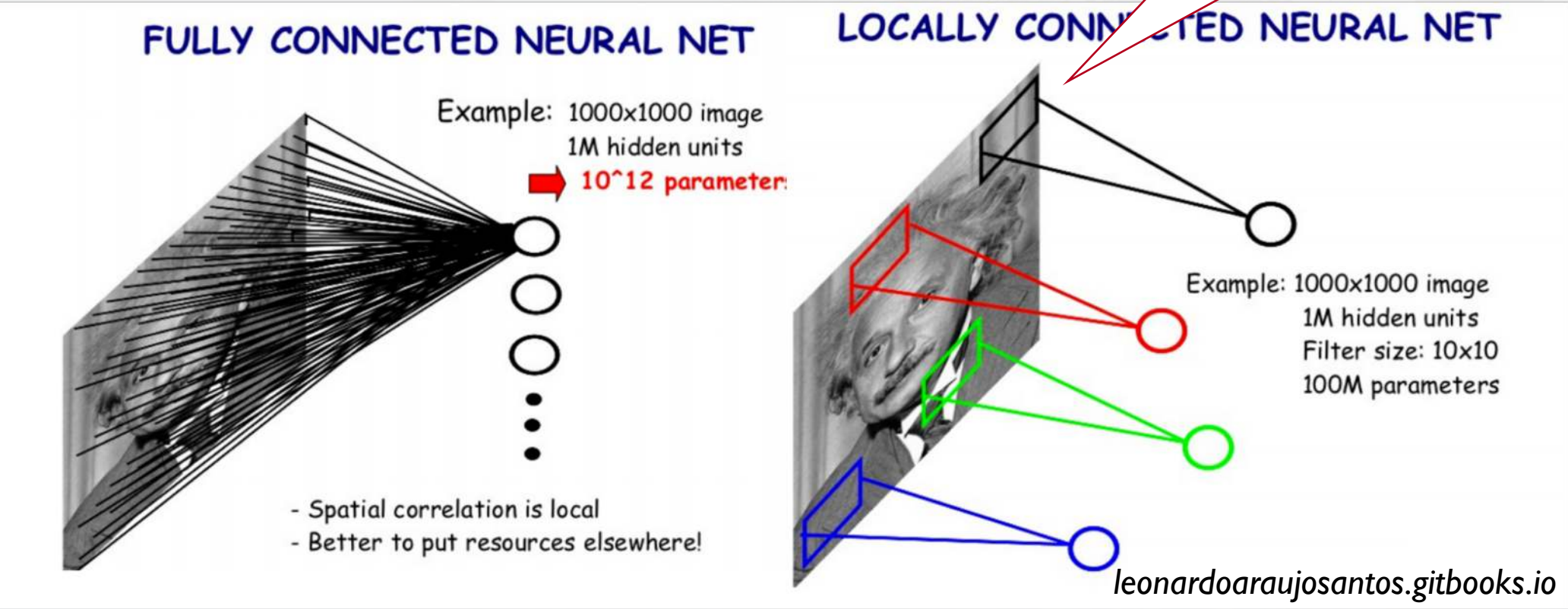


The Deep Learning Book

Local connectivity & weight sharing

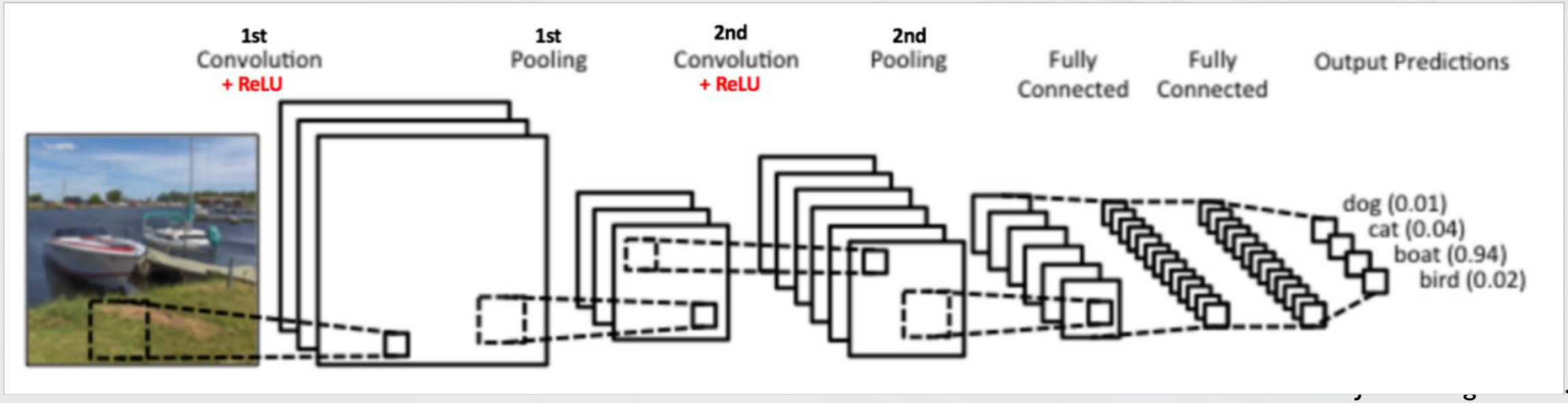
- Convolutional neurons are only locally connected
- The same filter is used for all positions in an image
- This dramatically reduces the number of free parameters

For each neuron: Instead of taking a dot-product across the entire image, only take a dot-product with part of image spanned by filter.



Natural Hierarchical Representation

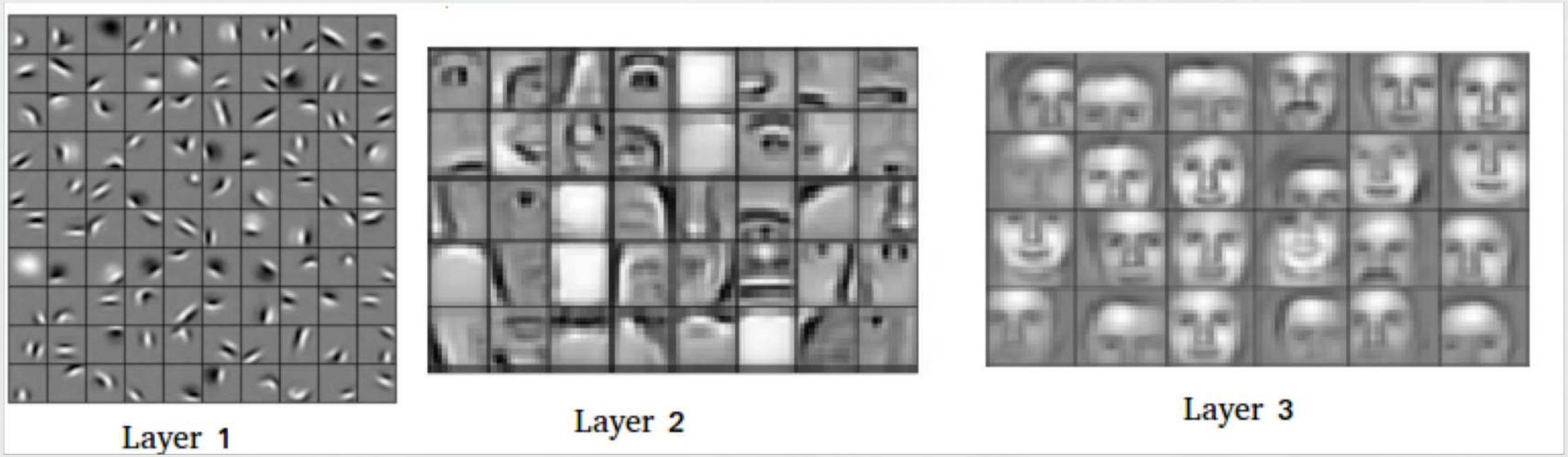
- The filters are applied both in parallel (channels), and sequentially (layers)
- The trained filters turn out to be hierarchically organised



.io

Natural Hierarchical Representation

- The filters are applied both in parallel (channels), and sequentially (layers)
- The trained filters turn out to be hierarchically organised



leonardoaraujosantos.gitbooks.io

Visalisation of information flow through ConvNet

By Adam Harley

http://scs.ryerson.ca/~aharley/vis/conv/flat.html

Visalisation of information flow through ConvNet

By Adam Harley

http://scs.ryerson.ca/~aharley/vis/conv/flat.html

Can you find ways to fool it?

Convolutional Neural Networks

Overview

Principles & Properties

- Convolution operation
- Automated feature extraction through learnable filters
- Efficiency through weight sharing
- Hierarchical data representation

Optimisation / Training

Applications

Convolutional Neural Networks

Overview

Principles & Properties

Optimisation / Training

- Iterative Optimization with SGD
- Batch Normalization
- Activation
- Pooling
- Backpropagation
- Vanishing Gradient

Applications

A tiny keras example

```
model3 = Sequential()

model3.add(Conv1D(4, 8, padding='same', input_shape=(400, 3)))
model3.add(BatchNormalization())
model3.add(Activation('relu'))
model3.add(MaxPooling1D(pool_size=2))

model3.add(Flatten())  # write the output into a single 1D vector

model3.add(Dense(8))
model3.add(BatchNormalization())
model3.add(Activation('relu'))

model3.add(Dense(2))
model3.add(Activation('softmax'))
```

Iterative optimization with SGD

- We update parameters iteratively with stochastic gradient descent (SGD):

1. Compute loss function
2. Compute gradient with respect to model parameters
3. Update parameters

- Instead of processing entire data set before each parameter update, only process one “mini batch” at a time
- Mini batch can be as small as a few dozen records or images
- Facilitates “batch normalisation”

Batch Normalization

- Before activation operation: subtract sample mean and divide by sample standard deviation
- Keeps activations in Gaussian range → empirical standardisation
- BatchNorm is a form of regularisation:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

> *"Because different examples are randomly chosen for inclusion in the minibatch at each step, the standard deviation randomly fluctuates. Batch norm also subtracts a random value (the mean of the minibatch) from each hidden unit at each step. Both of these sources of noise mean that every layer has to learn to be robust to a lot of variation in its input."*
>
> *Ian Goodfellow on stackExchange*

Other forms of regularisation

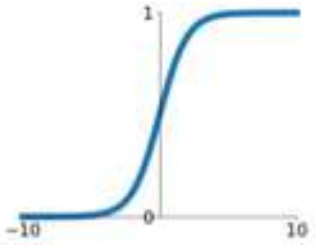
- L1 / L2 / Tikhonov Regularisation
- Dropout
  - Set a fraction of weights to zero randomly
  - Forces network to find multiple paths to same solution
- Scaled Exponential Linear Unit

Activation

- Map output values into desired range, e.g. [0, 1] or [-1, 1]

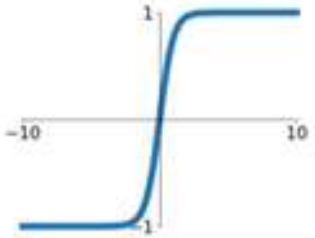
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



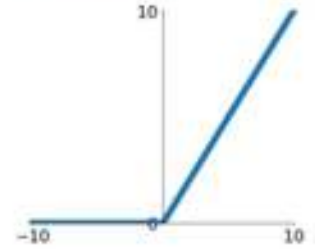
tanh

$$\tanh(x)$$



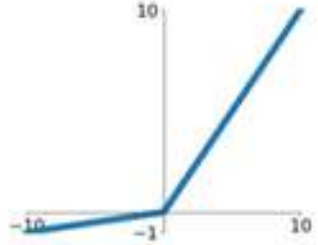
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

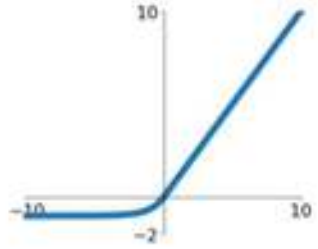


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

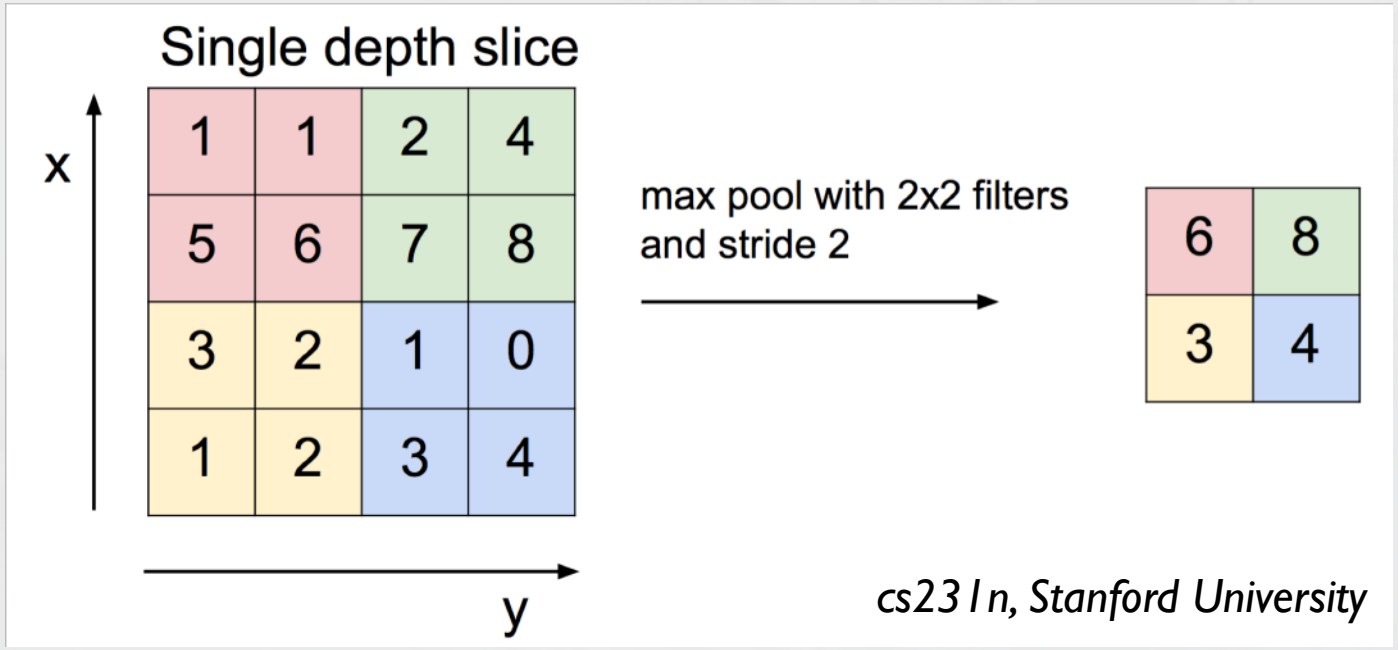
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Pooling

- Low-pass filter input data; represent data at different scale
- Enables scale-invariant object detection
- Makes representation invariant to small translations of input

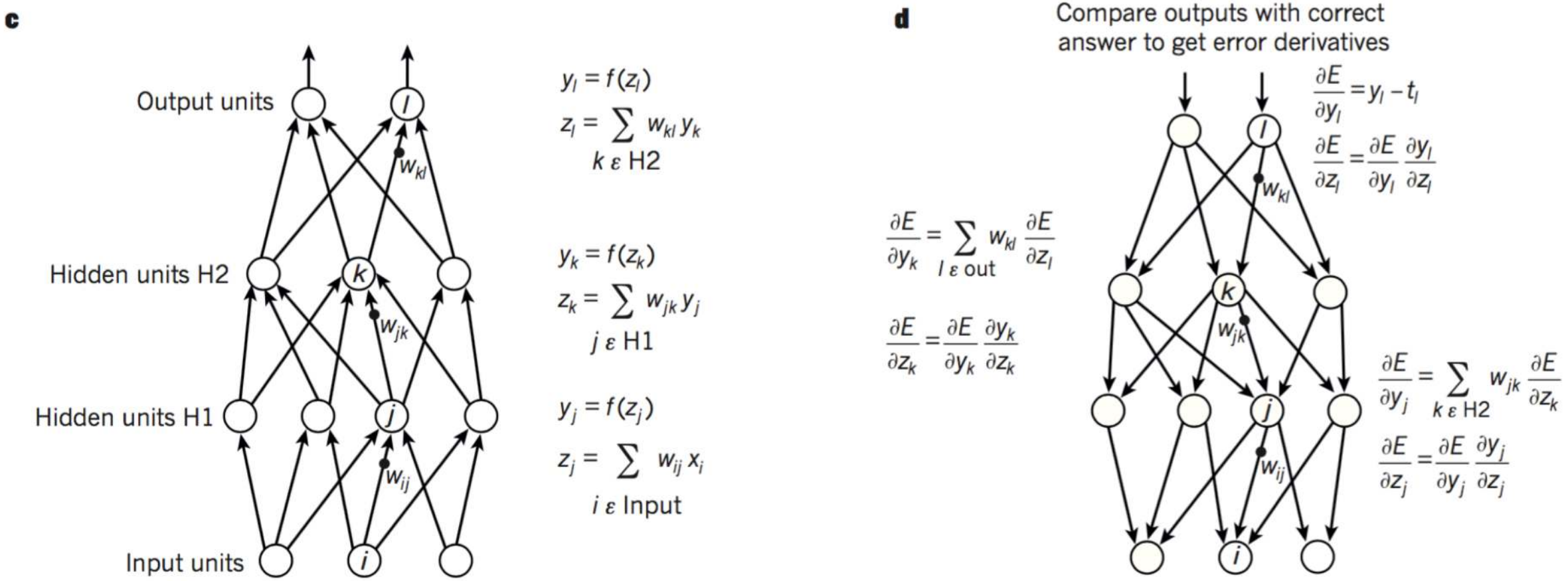


A tiny keras example

```
model3 = Sequential()

model3.add(Conv1D(4, 8, padding='same', input_shape=(400, 3)))
model3.add(BatchNormalization())
model3.add(Activation('relu'))
model3.add(MaxPooling1D(pool_size=2))

model3.add(Flatten())  # write the output into a single 1D vector

model3.add(Dense(8))
model3.add(BatchNormalization())
model3.add(Activation('relu'))

model3.add(Dense(2))
model3.add(Activation('softmax'))
```

Optimization of deep network with back-propagation

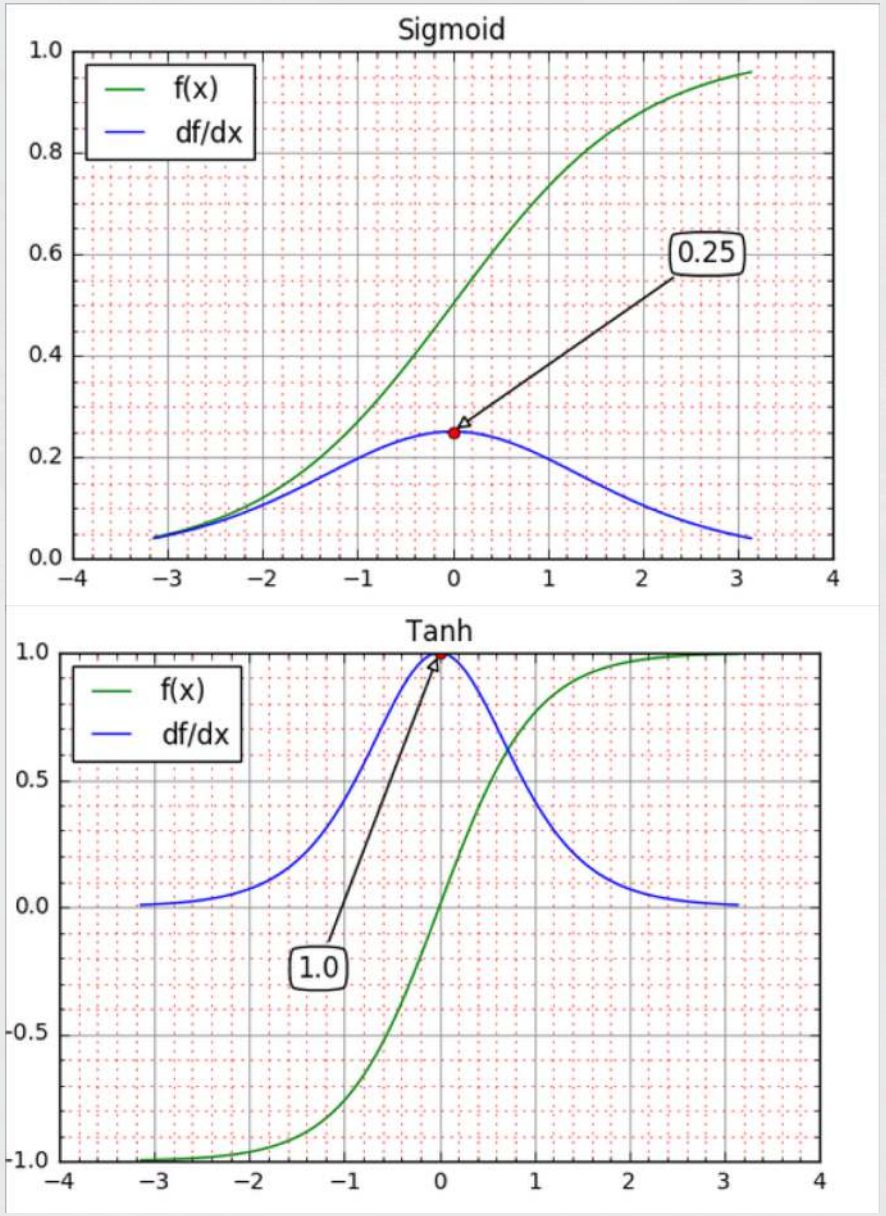
- "nothing more than a practical application of the chain rule"



LeCun et al., 2015, Nature

Vanishing Gradient

- In back-propagation through deep models we multiply partial derivatives of loss function from all layers
- If derivatives are small, total gradient can approach zero, or „vanishes“
- Gradient decreases exponentially with number of layers
- Even strong parameter modifications in early layers do not have strong effect on network output
- Can slow down or even break training
- For many activation functions, large regions of input space map into very small range
- ReLU map x to max(0,x), i.e. don‘t squash output



Exploding Gradient

Exploding Gradient

- Strong gradients lead to extreme parameter updates
- Can make training unstable
- Lots of tricks and hacks to avoid it
  - Shrink network
  - Clip gradients
  - Weight regularisation
  - ...

Convolutional Neural Networks

Overview

Principles & Properties

Optimisation / Training

- Iterative Optimization with SGD
- Batch Normalization
- Activation
- Pooling
- Backpropagation
- Vanishing Gradient

Applications

Convolutional Neural Networks

Overview

Principles & Properties

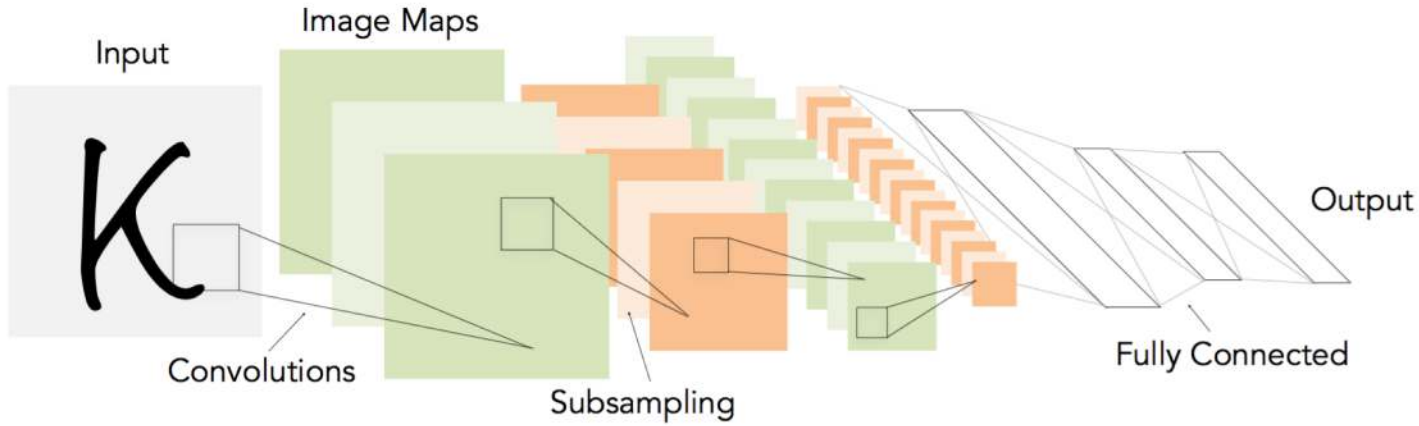
Optimisation / Training

Applications

- A few state of the art ConvNets
- Practical
- ConvNets for seismology

1998

LeCun et al.



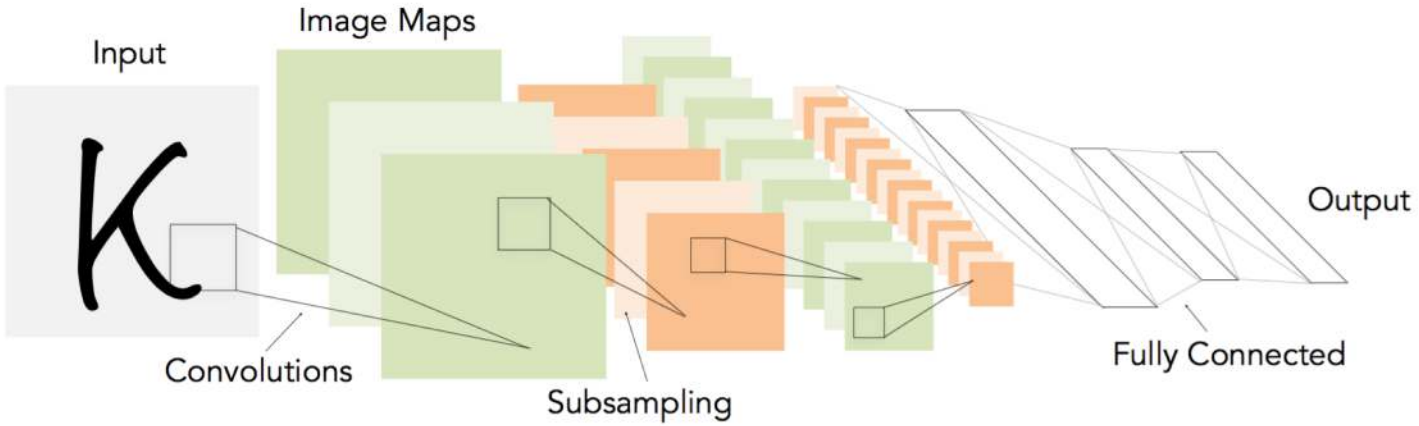
# of transistors

10^6

# of pixels used in training

10^7 NIST

1998
LeCun et al.



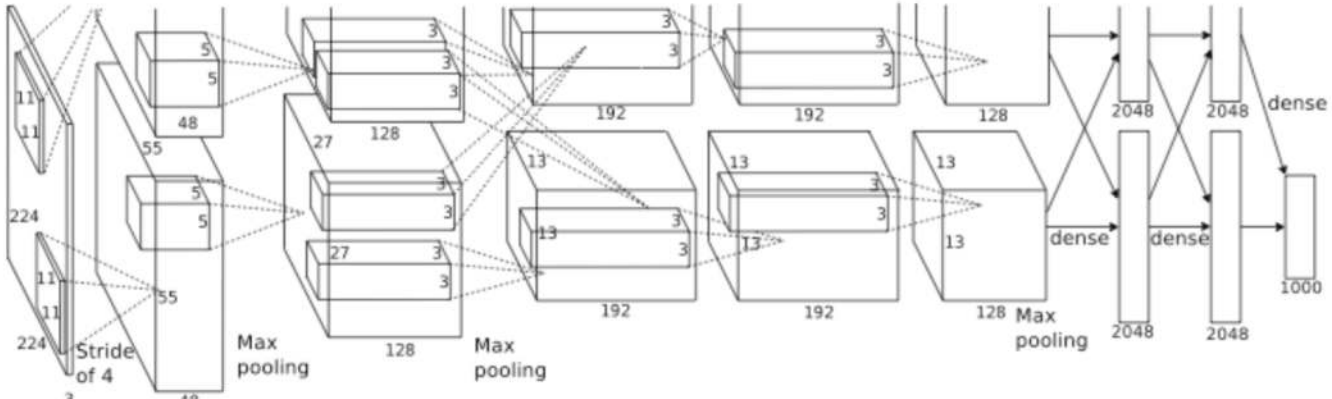
# of transistors

10^6

# of pixels used in training

10^7 NIST

2012
Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

Dense descriptor grid: HOG, LBP

Coding: local coordinate, super-vector

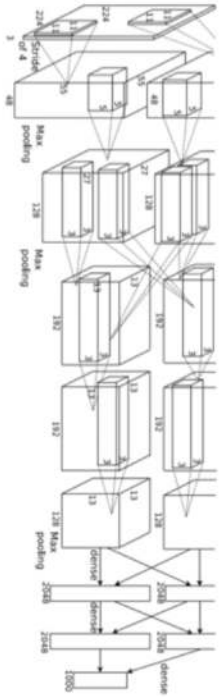
Pooling, SPM

Linear SVM

[Lin CVPR 2011]

Year 2012

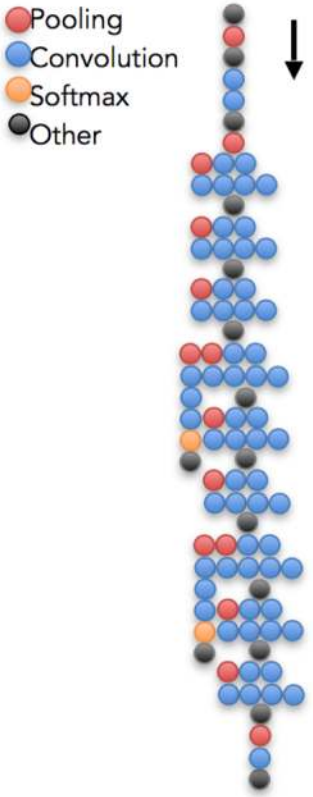
SuperVision



[Krizhevsky NIPS 2012]

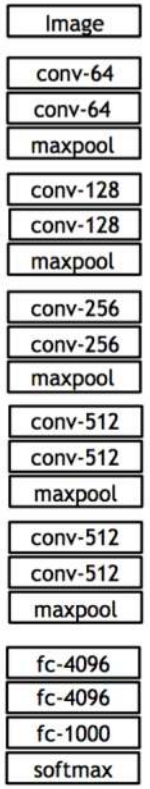
Year 2014

GoogLeNet



[Szegedy arxiv 2014]

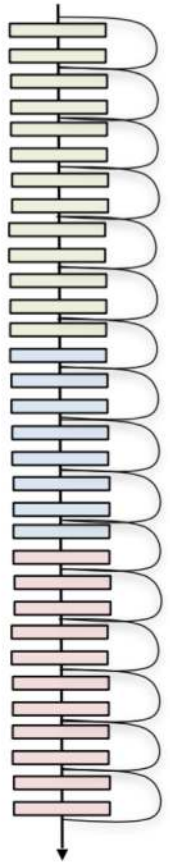
VGG



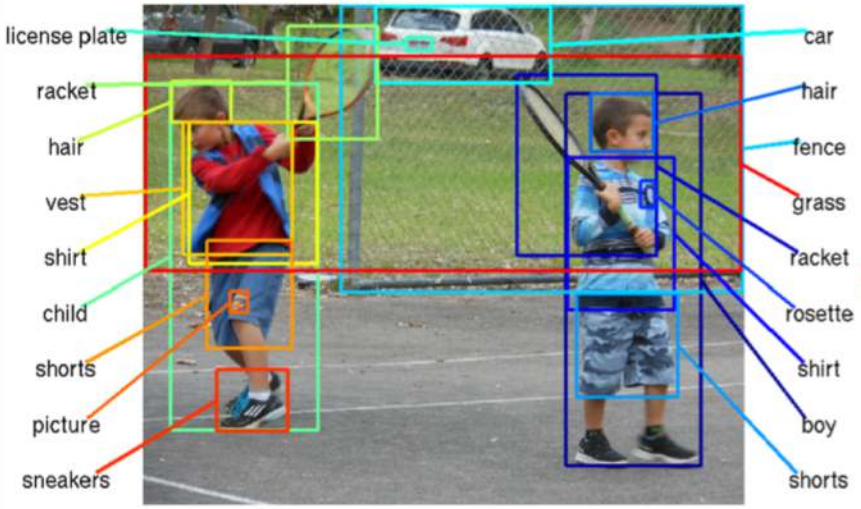
[Simonyan arxiv 2014]

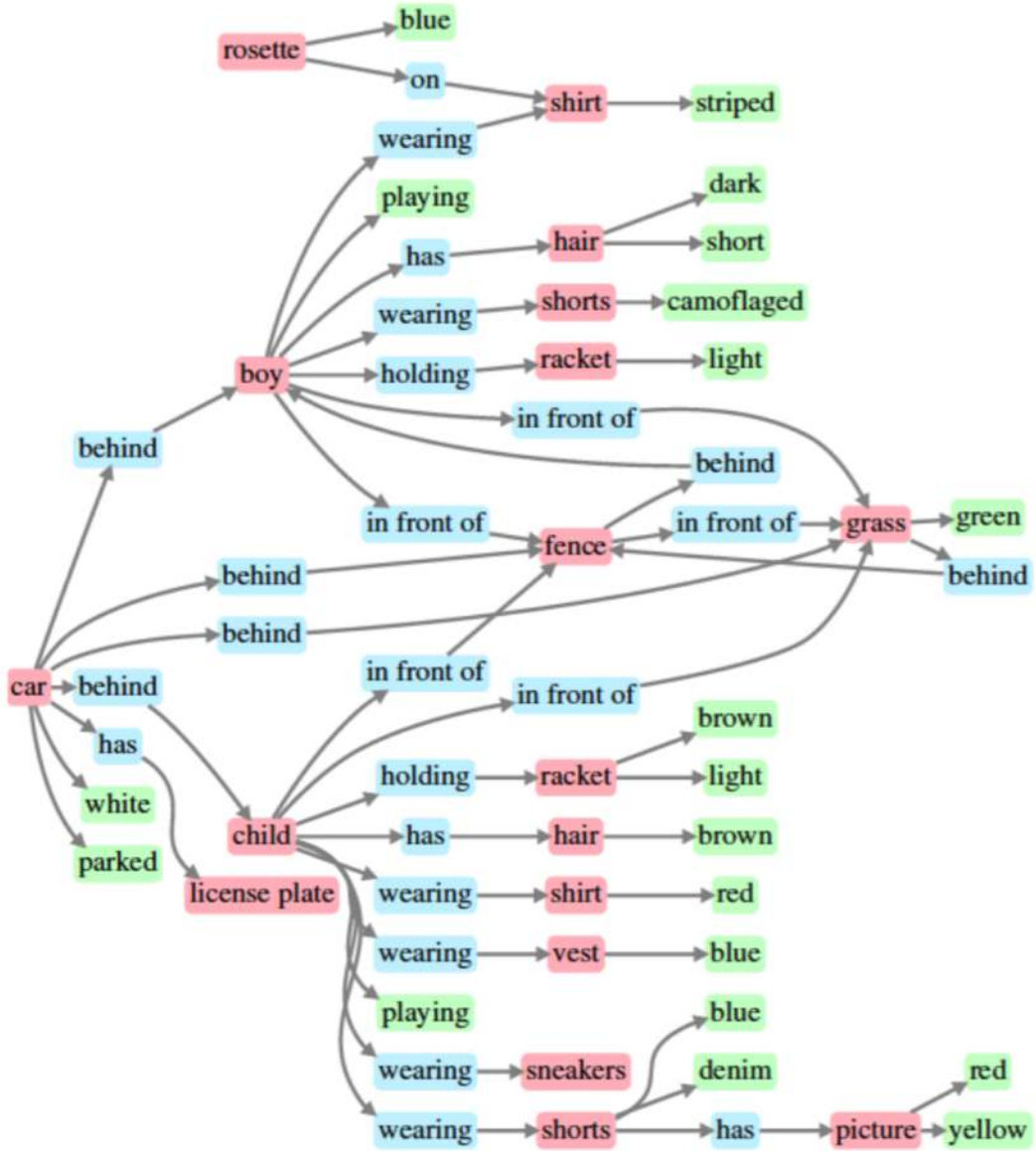
Year 2015

MSRA



[He ICCV 2015]





Johnson *et al.*, "Image Retrieval using Scene Graphs", CVPR 2015

Slide from Stanford's cs231n class

PT = 500ms

Some kind of game or fight. Two groups of two men? The man on the left is throwing something. Outdoors seemed like because i have an impression of grass and maybe lines on the grass? That would be why I think perhaps a game, rough game though, more like rugby than football because they pairs weren't in pads and helmets, though I did get the impression of similar clothing. maybe some trees? in the background. (Subject: SM)

Fei-Fei, Iyer, Koch, Perona, JoV, 2007

Slide from Stanford's cs231n class

No errors

A white teddy bear sitting in the grass

Minor errors

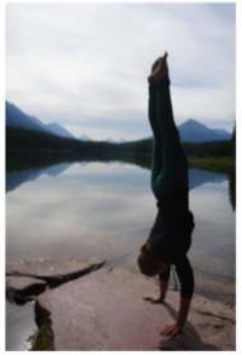
A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

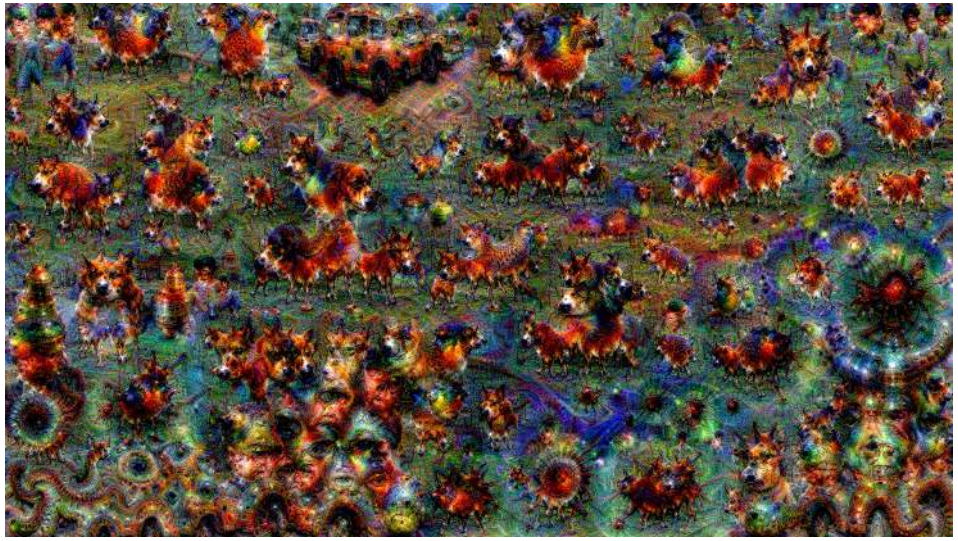


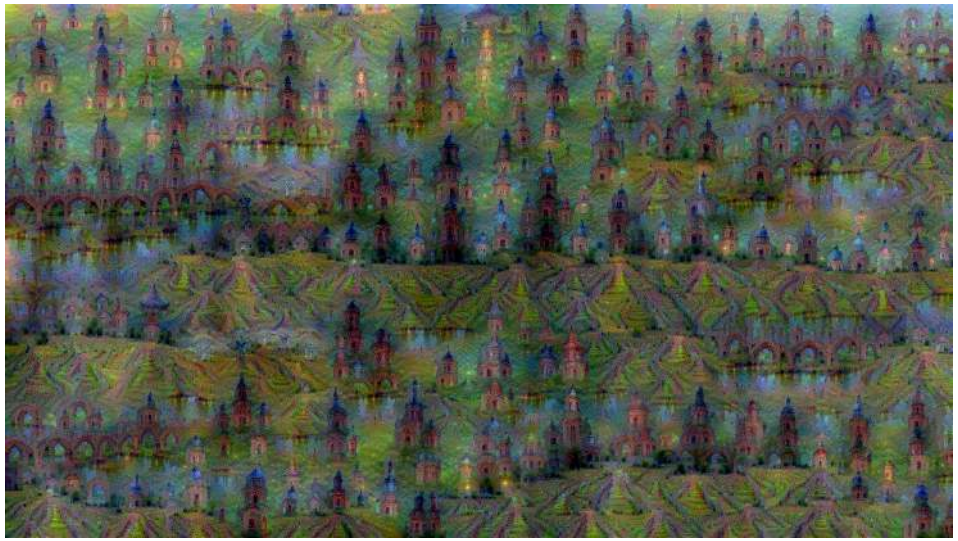
A woman standing on a beach holding a surfboard

Image Captioning

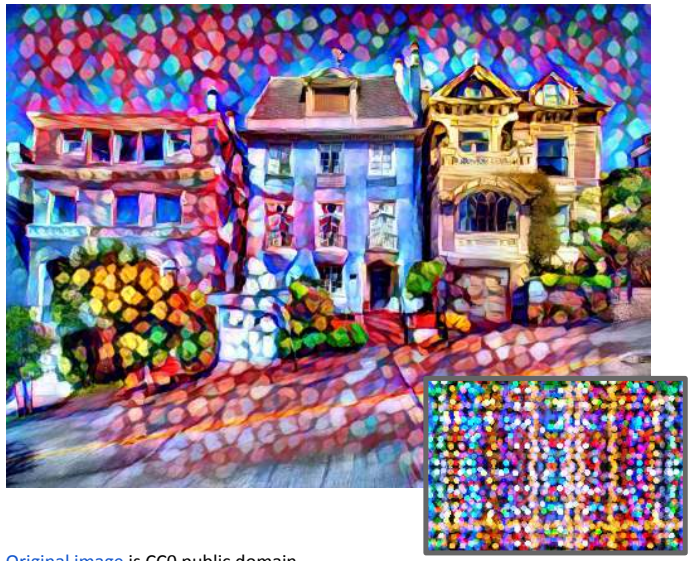
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

Captions generated by Justin Johnson using Neuraltalk2





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

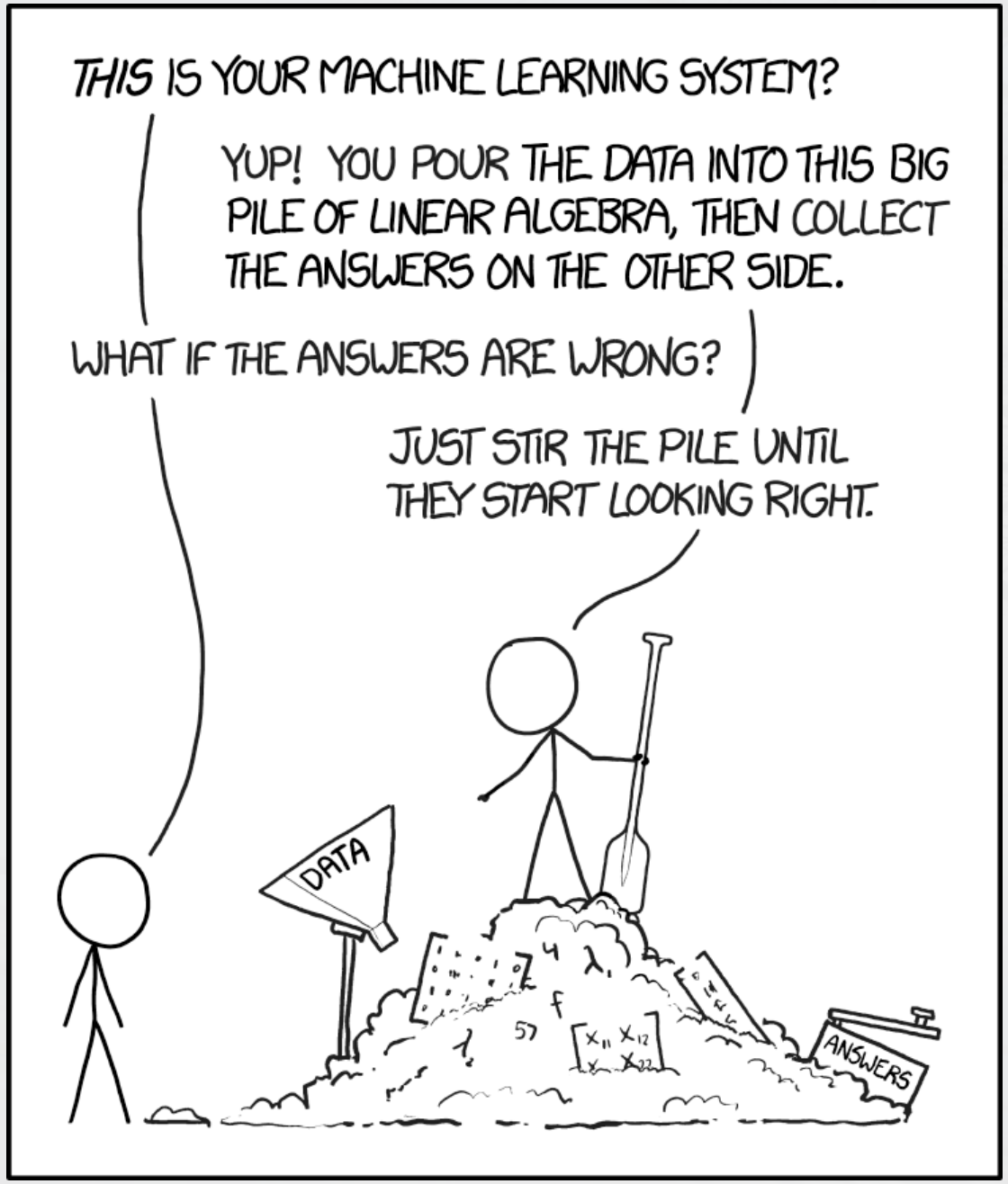


Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017; reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Slide from Stanford's cs231n class

ConvNet Summary


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

xkcd.org

ConvNet Summary

- DL models …
  - … are highly flexible models with very large numbers of free parameters
  - … require large amounts of data and computational resources to train
  - … with enough data, perform much better than simpler models
- Convolution preserves spatial or temporal structure in data
- ConvNets have rather intuitive explanations
- *"models that will learn how to best represent your data (feature selection) automatically and based on the dataset given to it."*
- *„if your problem gets more complex you just make your model 'deeper' and get more data (a lot) to train to your new problem and the model will then learn what is the best features for your particular task."*

finally, practical …

Convolutional Neural Networks

Overview

Principles & Properties

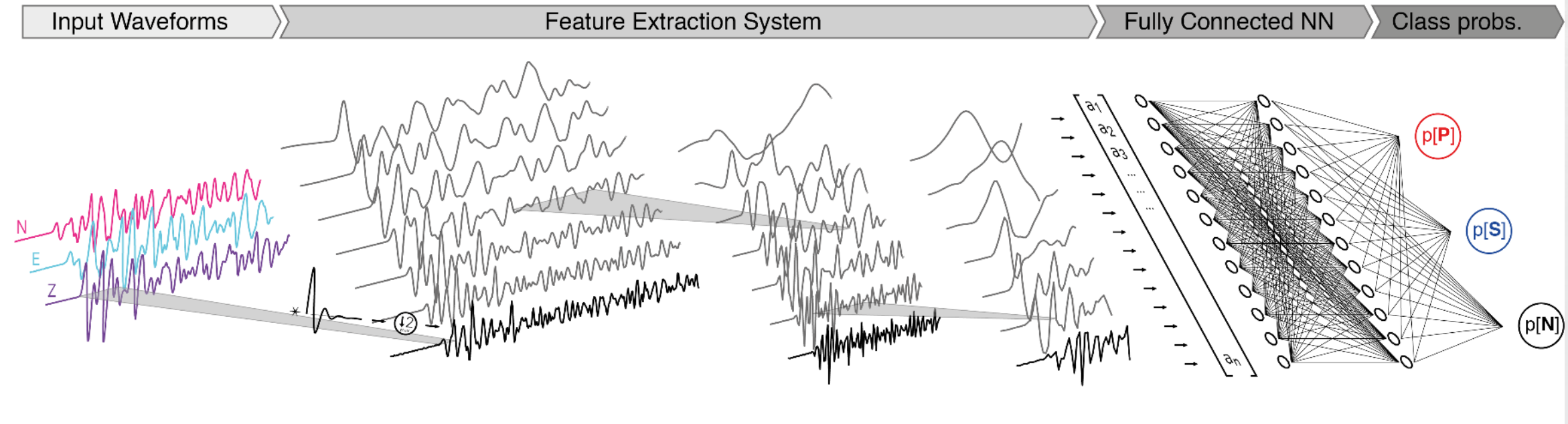
Optimisation / Training

Applications

- A few state of the art ConvNets
- Practical
- **ConvNets for seismology**

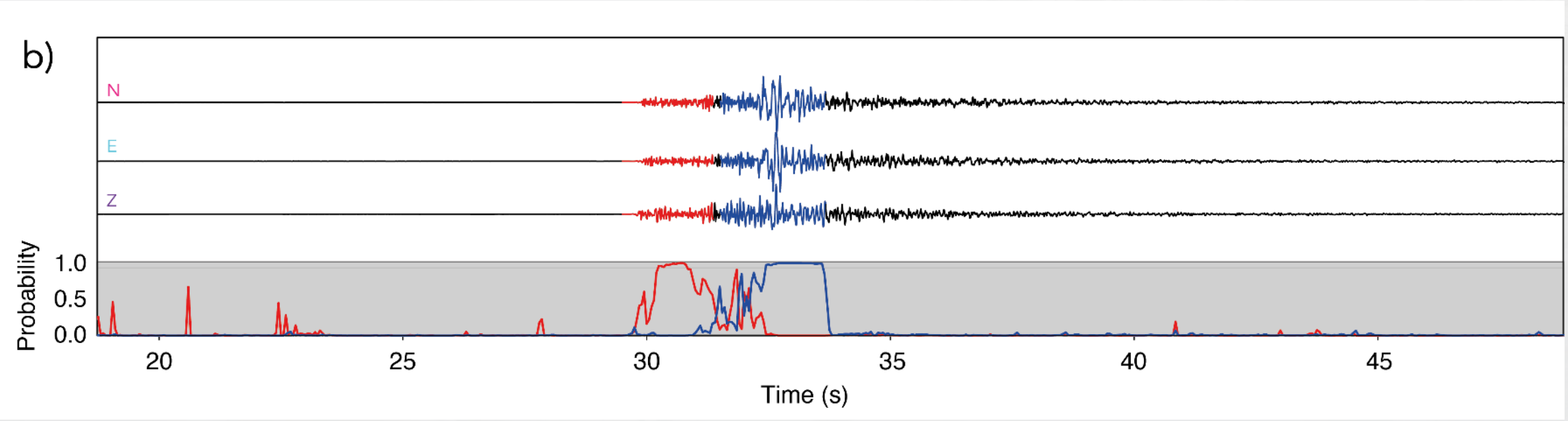
ConvNets in Seismology

- 3 color (RGB) bitmap image in computer vision <> 3-component seismogram
- Goal: distinguish between direct P-, S- and noise phases
- In the training process we *teach* the ConvNet *what seismic phases generally look like*



Generalized Phase Detection

For details, see Ross, Z. E., Meier, M.-A., Hauksson, E., and T. H. Heaton (2018). Generalized Seismic Phase Detection with Deep Learning, *BSSA*



Ross et al., 2018, BSSA

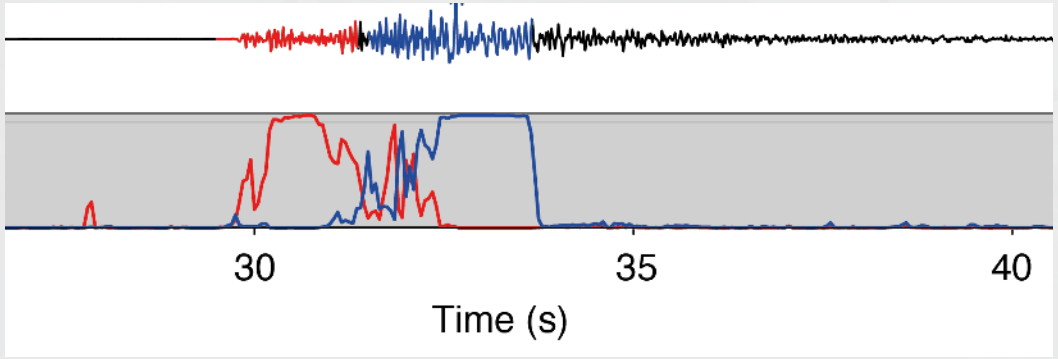
Generalized Phase Detection

- Performance on ~1.125 Mio. validation records
  - Precision = TP / (TP + FP)
  - Recall = TP / (TP + FN)

Of all the detections you've made, how many were correct?

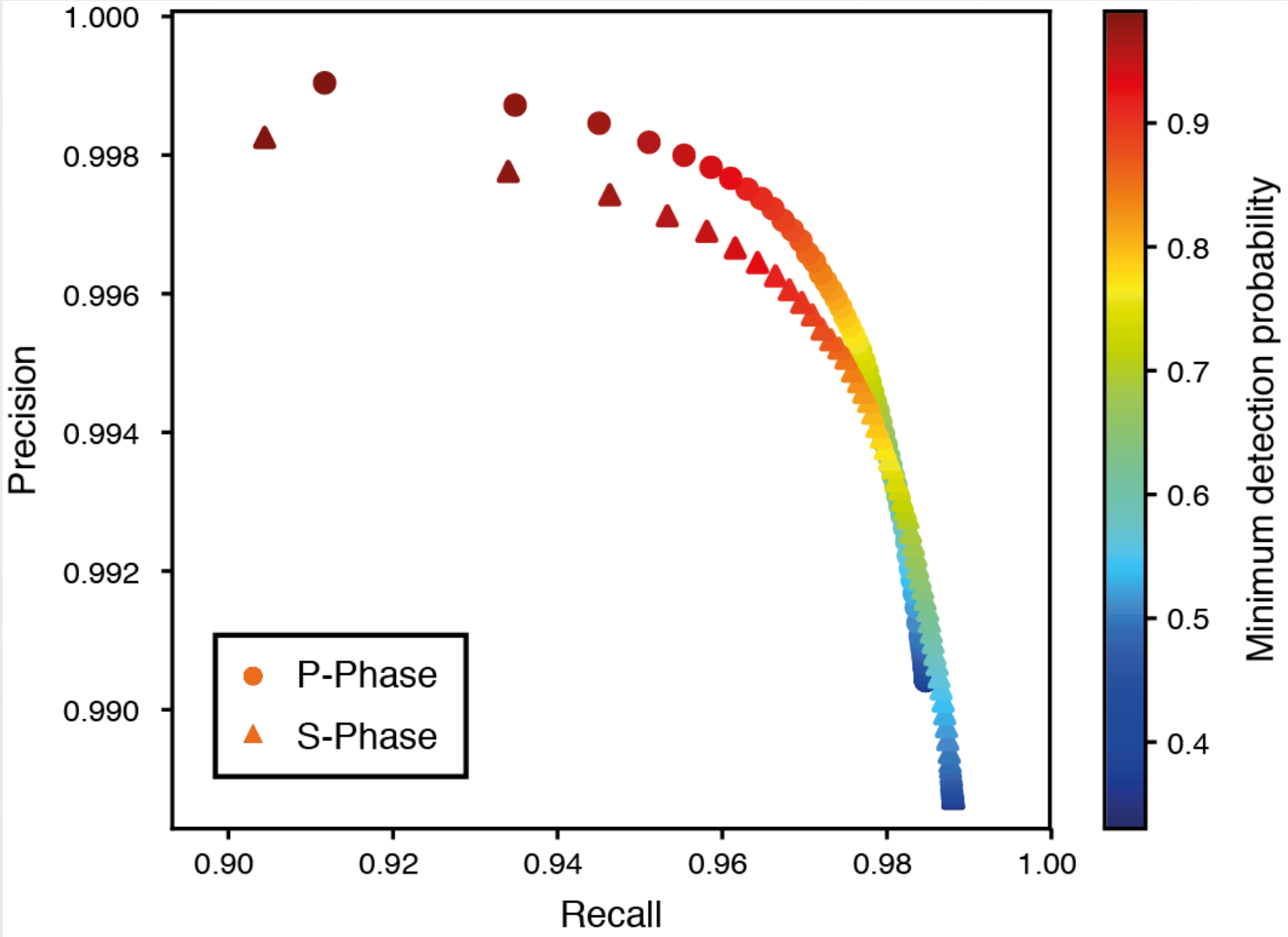
Of all the phases you should have detected, how many did you detect?

M \ H	P	S	N
P	TP_{PP}	ε_{SP}	ε_{NP}
S	ε_{PS}	TP_{SS}	ε_{NS}
N	ε_{PN}	ε_{SN}	TP_{NN}



Ross et al., 2018, BSSA

Generalized Phase Detection

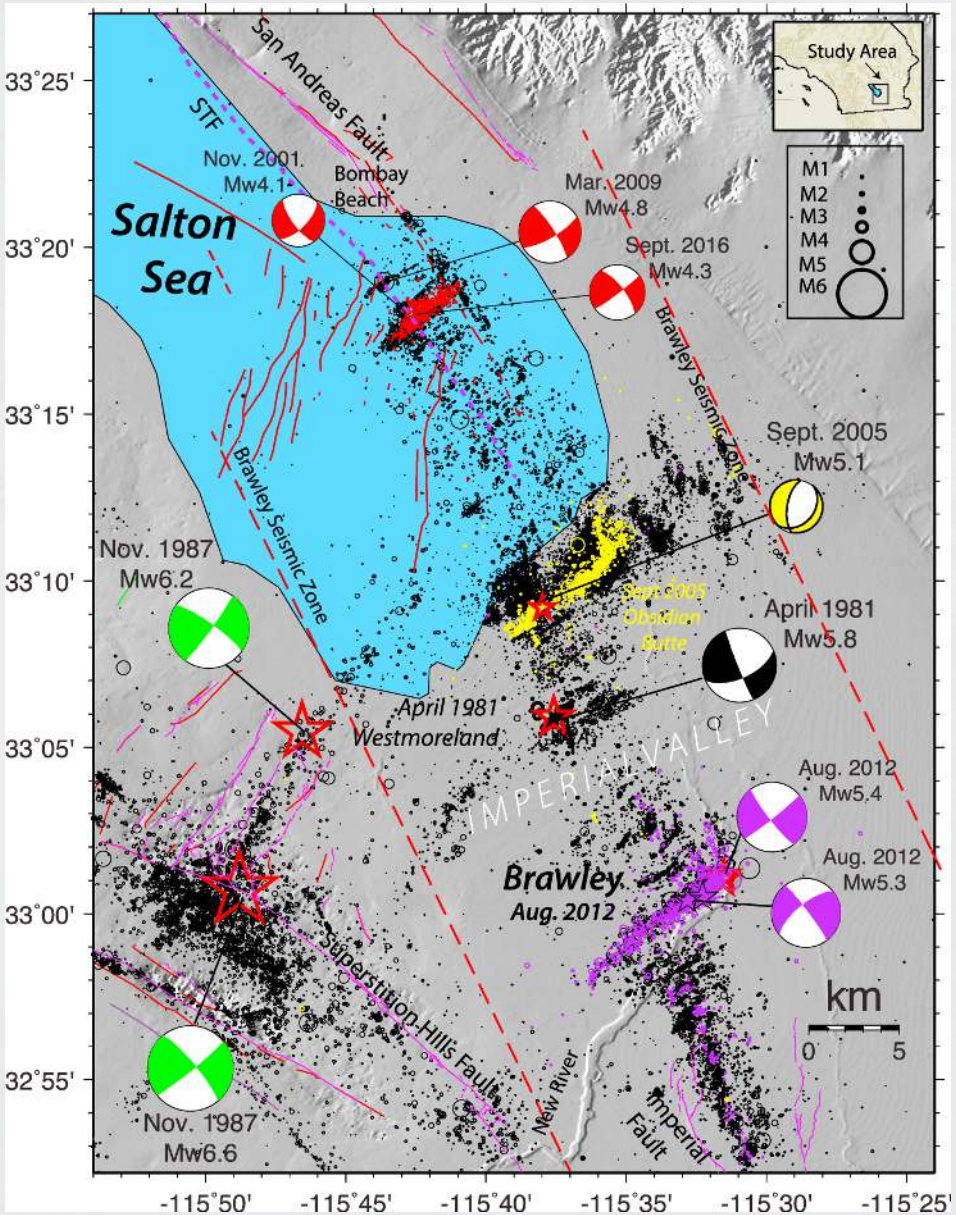


M \ H	P	S	N
P	TP_{PP}	ε_{SP}	ε_{NP}
S	ε_{PS}	TP_{SS}	ε_{NS}
N	ε_{PN}	ε_{SN}	TP_{NN}

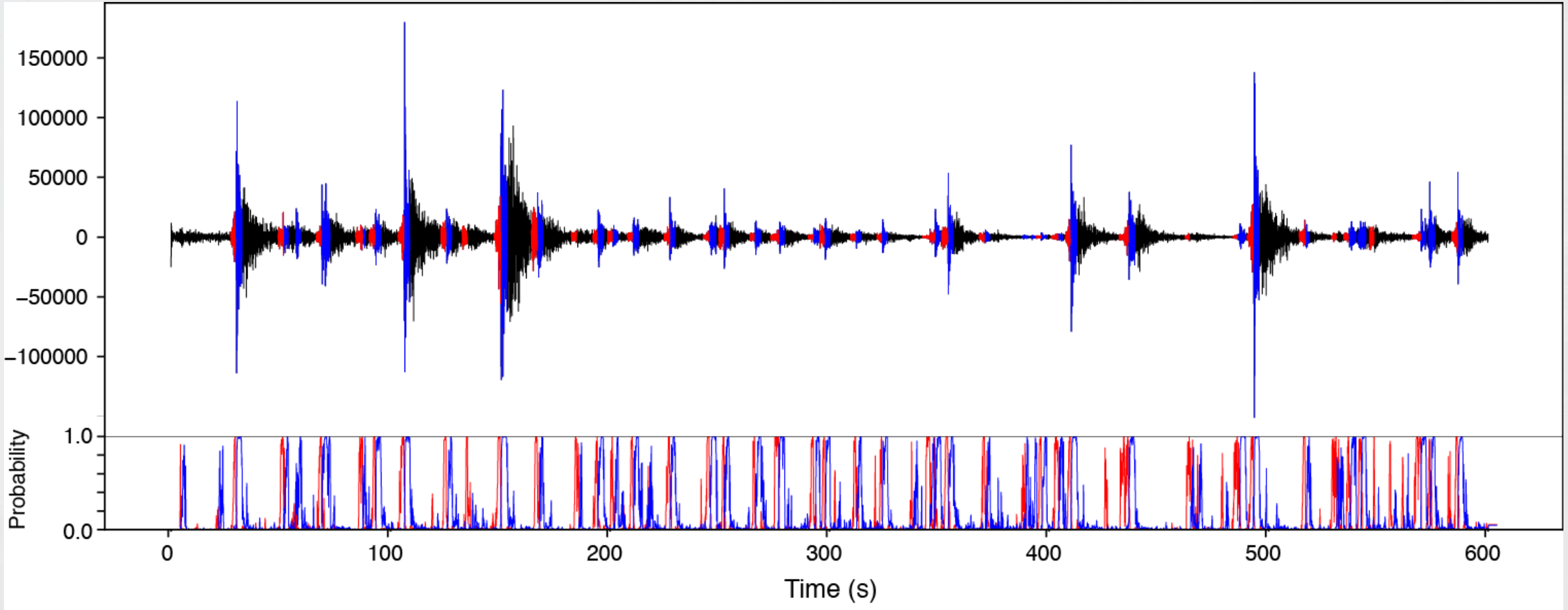
Ross et al., 2018, BSSA

Application to continuous data

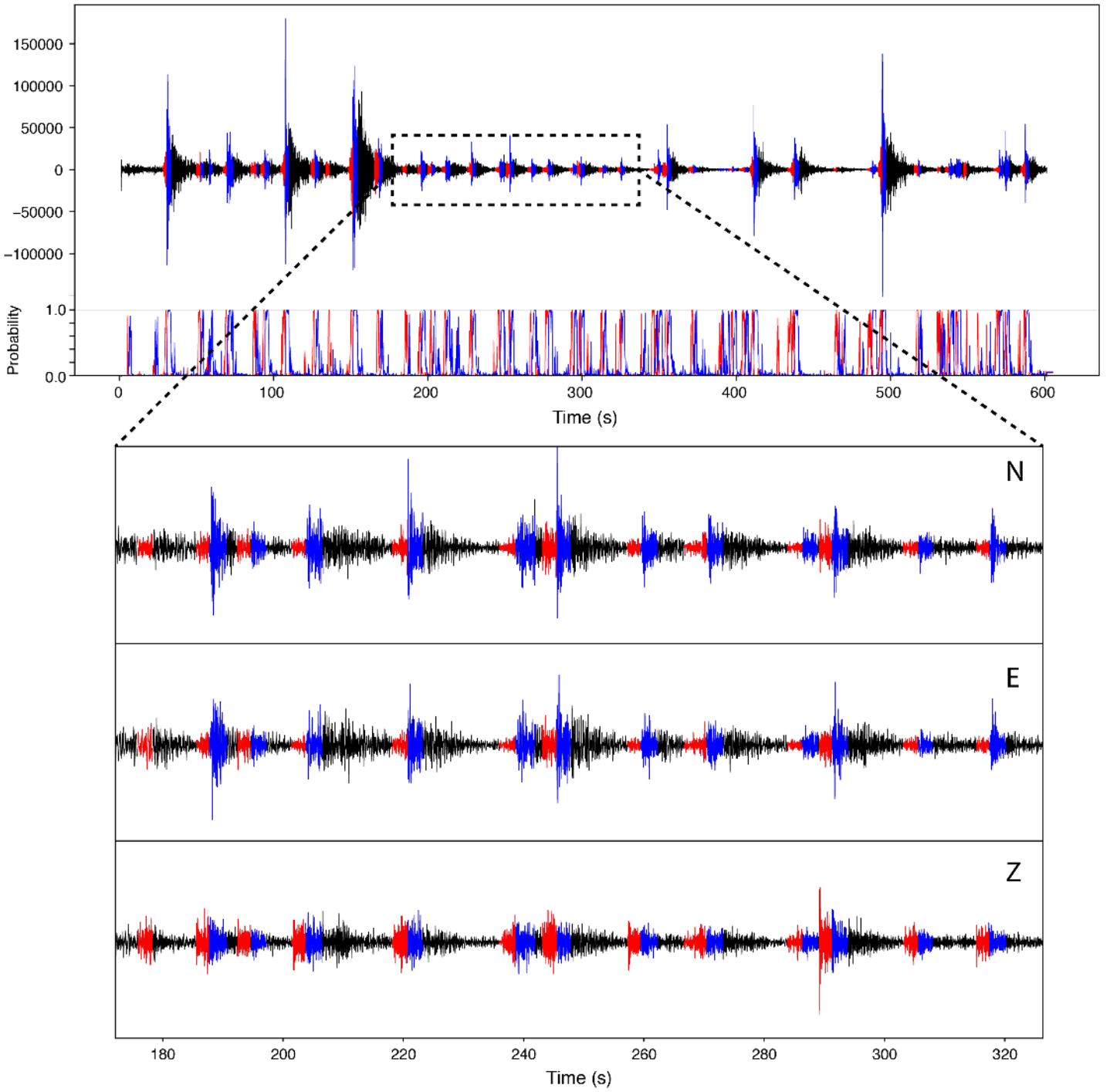
2016 Bombay Beach, California swarm



Hauksson et al., 2017, GRL

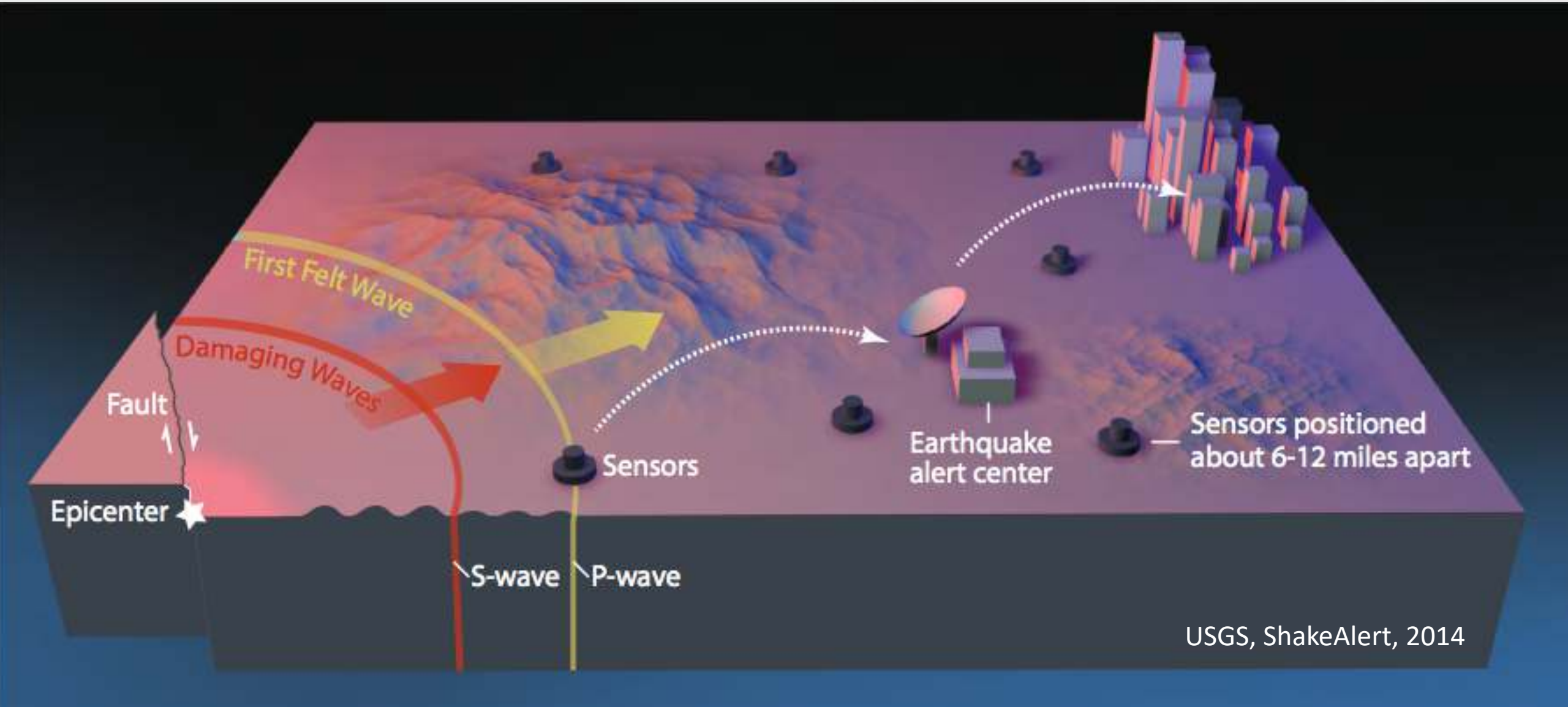


Ross et al., 2018, BSSA

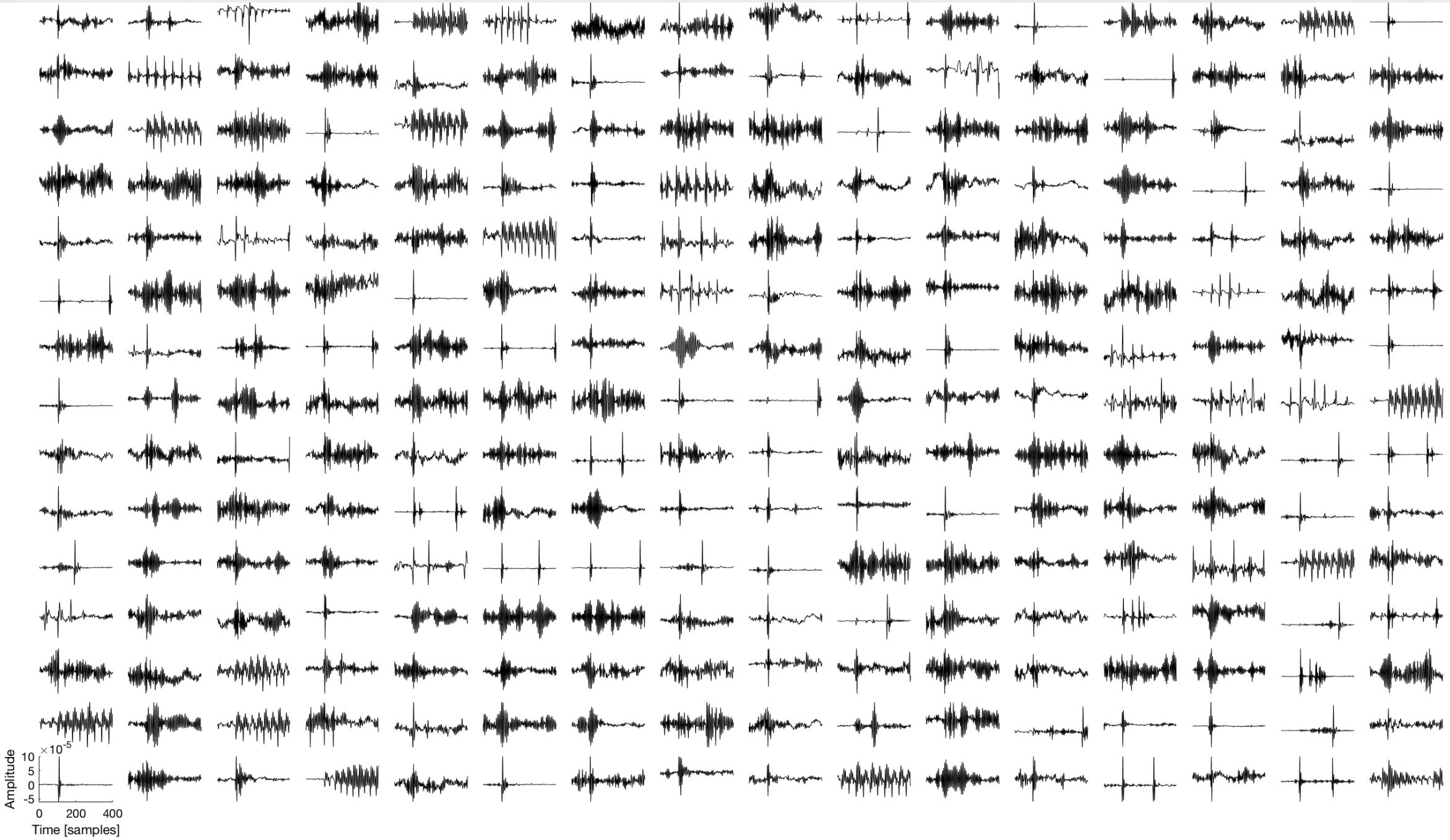


Ross et al., 2018, BSSA

Application to impulsive noise data



USGS, ShakeAlert, 2014


×10^-5
10
5
0
-5
Amplitude
0
200
400
Time [samples]

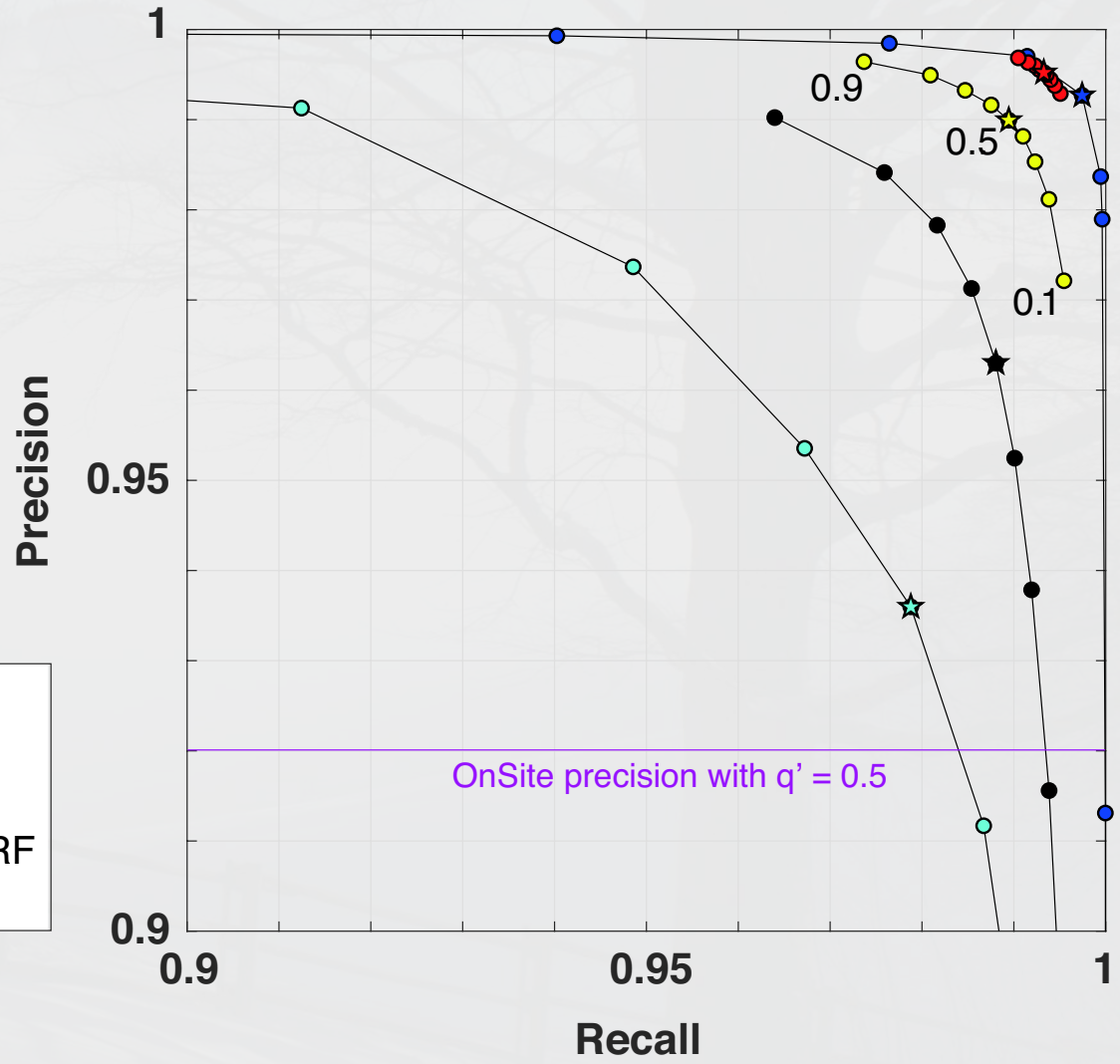
Aplication to impulsive noise data

- "Noise" data set
  - 946k impulsive noise signals
  - STA/LTA triggers from ShakeAlert log-files
  - Discard ~600k records to balance number of records
- "Quake" data set,
  - 374k signals from 8,432 earthquakes
  - From Japan & SoCal
  - M4-9

Meier et al., 2019, JGR

Comparison with simpler classifiers

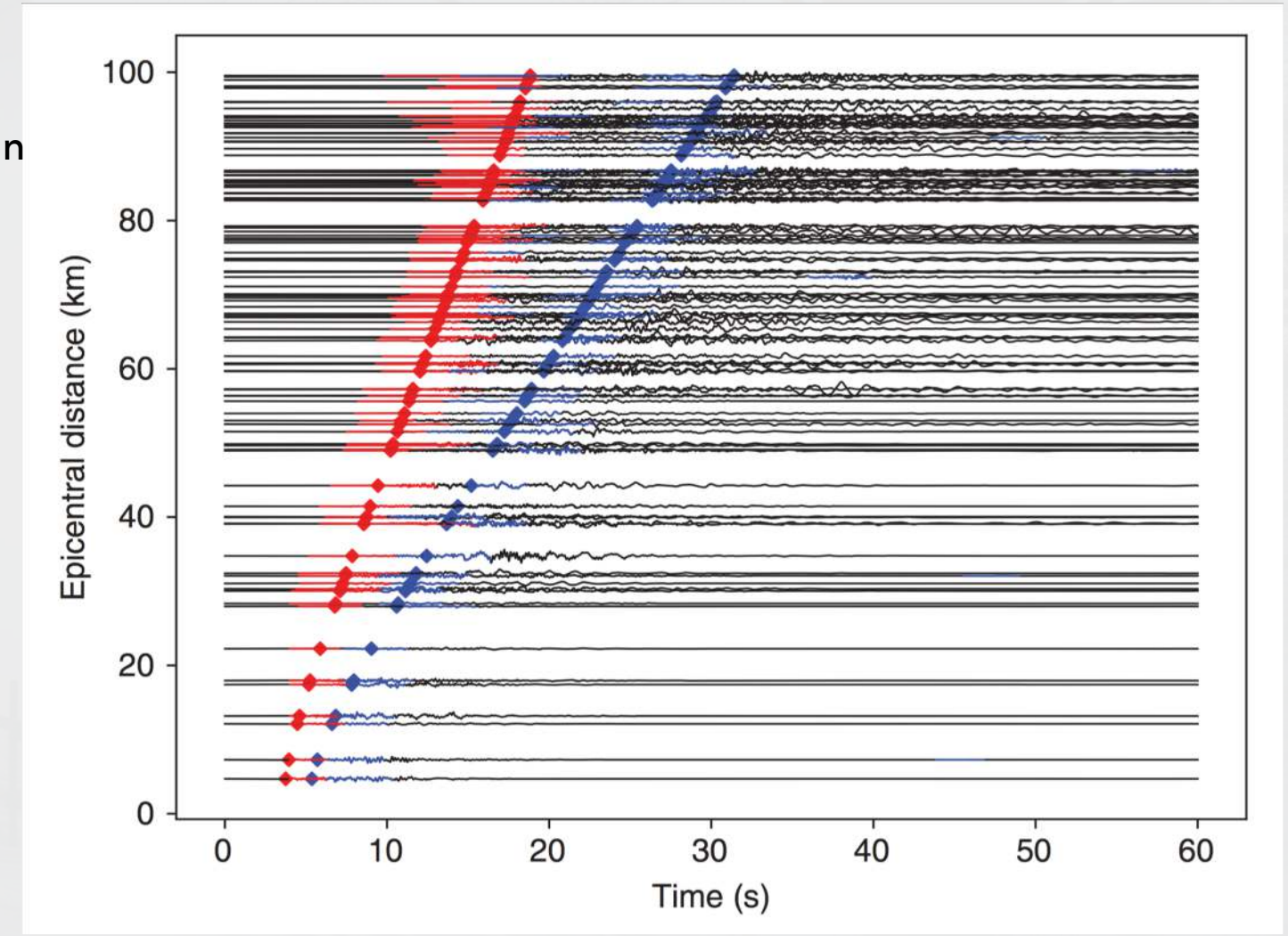
- Deep models perform significantly better than simpler feature-based models



Meier et al., 2019, JGR

How portable are GPD models?

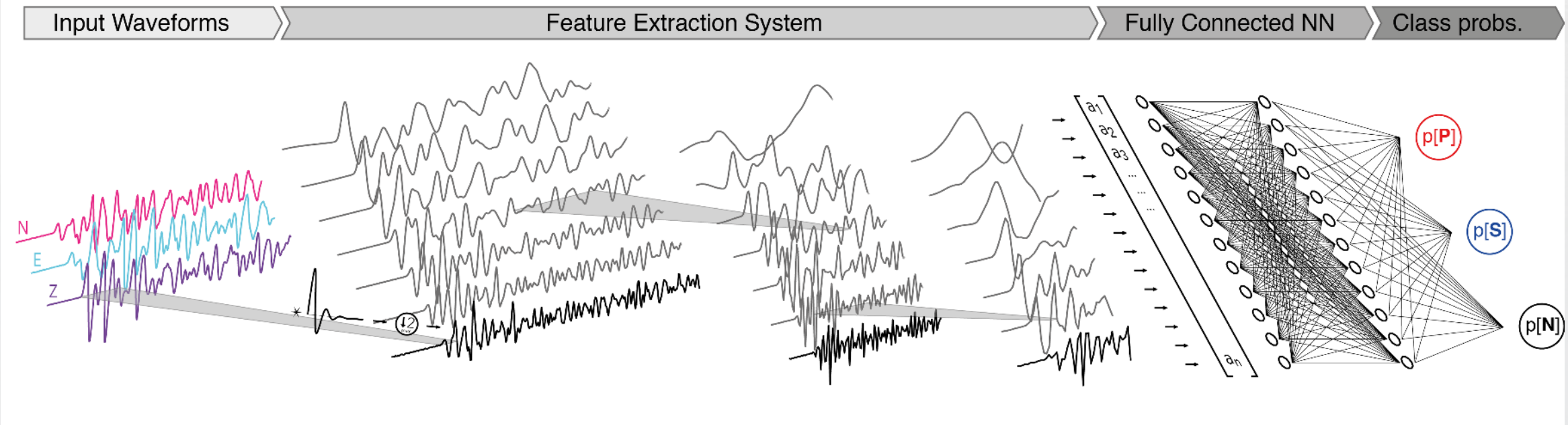
- Trained on SoCal data, mostly M<5
- Applied to 2016 M7.0 Kumamoto, Japan



Ross et al., 2018, BSSA

Comments on Generalized Phase Detection

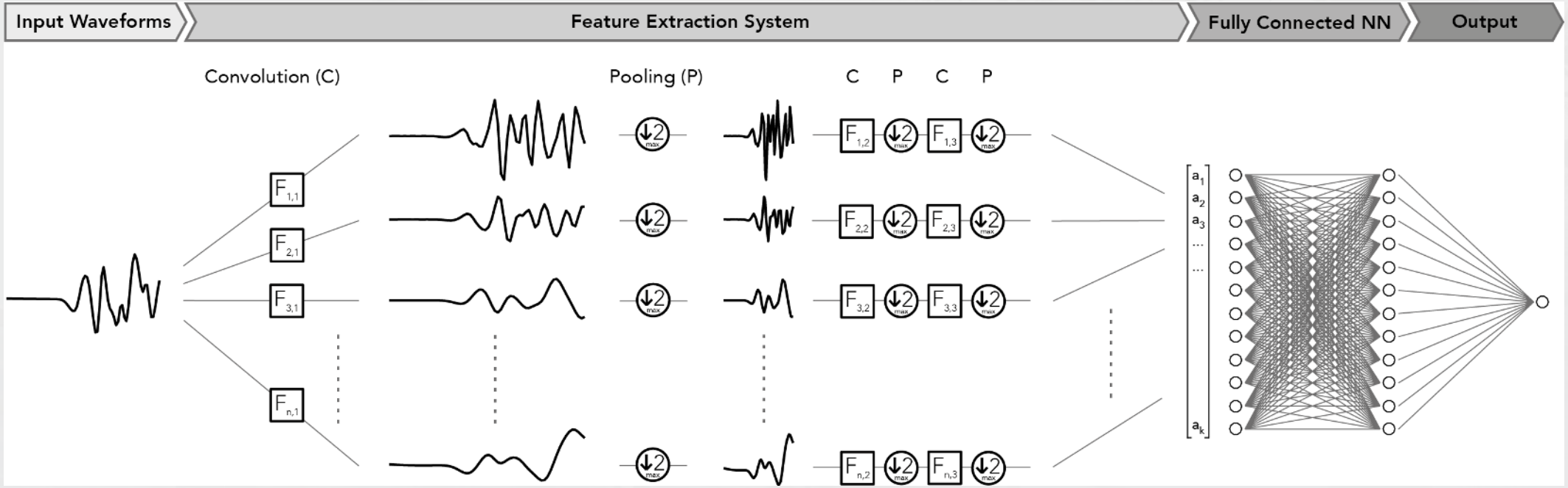
- GPD model has learned from large data set *what seismic waves look like*
- This is in contrast to e.g. STA/LTA detectors, which trigger on any impulsive signal



Other DL applications: Arrival Time Picking

Use similar ConvNet as a regressor

The only code change needed: omit activation of last layer

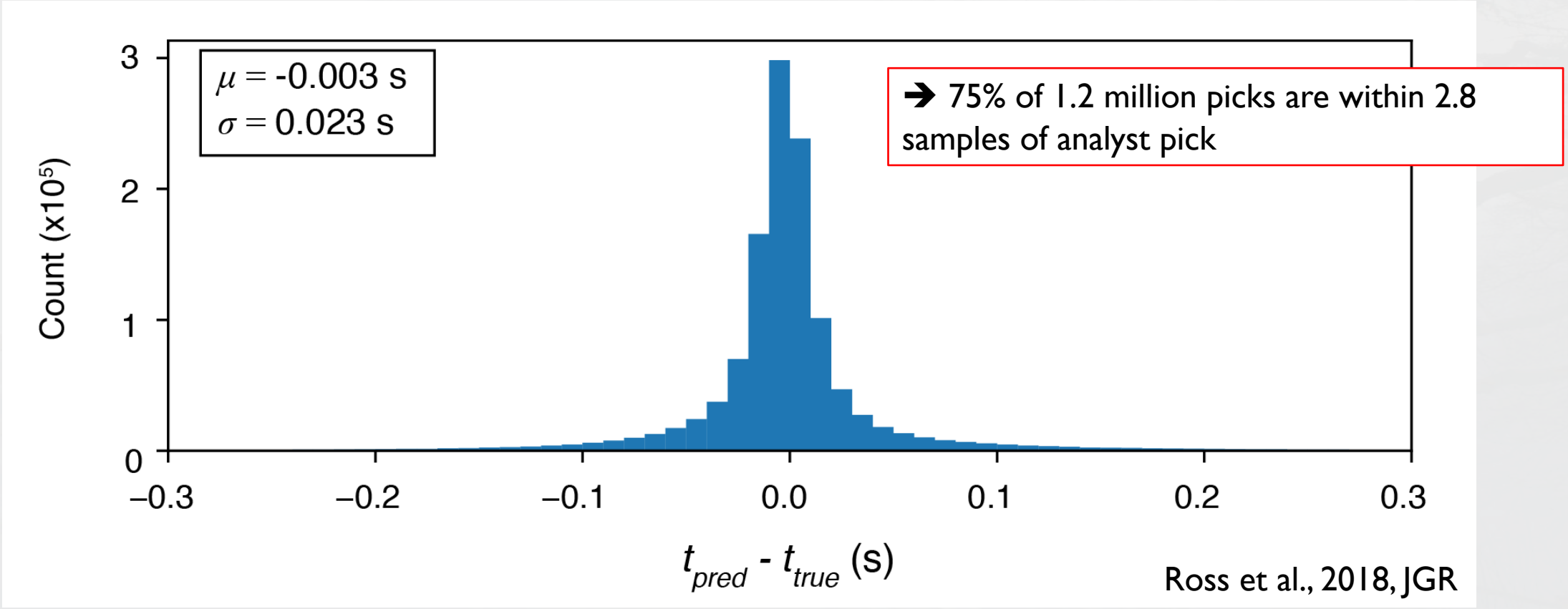


Ross et al., 2018, JGR

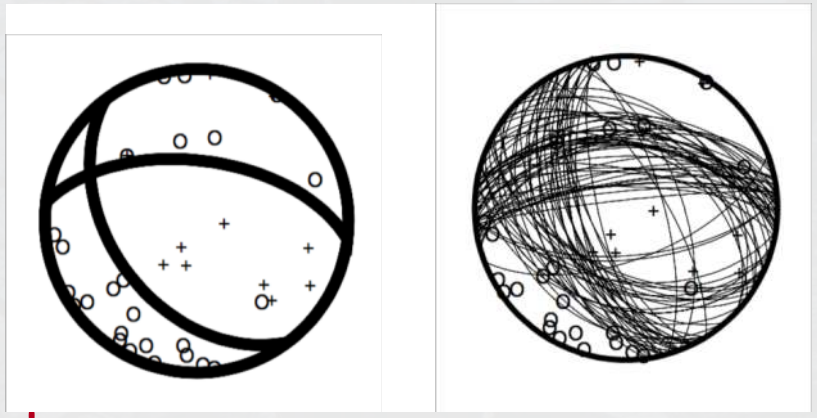
Other DL applications: Arrival Time Picking

Use similar ConvNet as a regressor

The only code change needed: omit activation of last layer

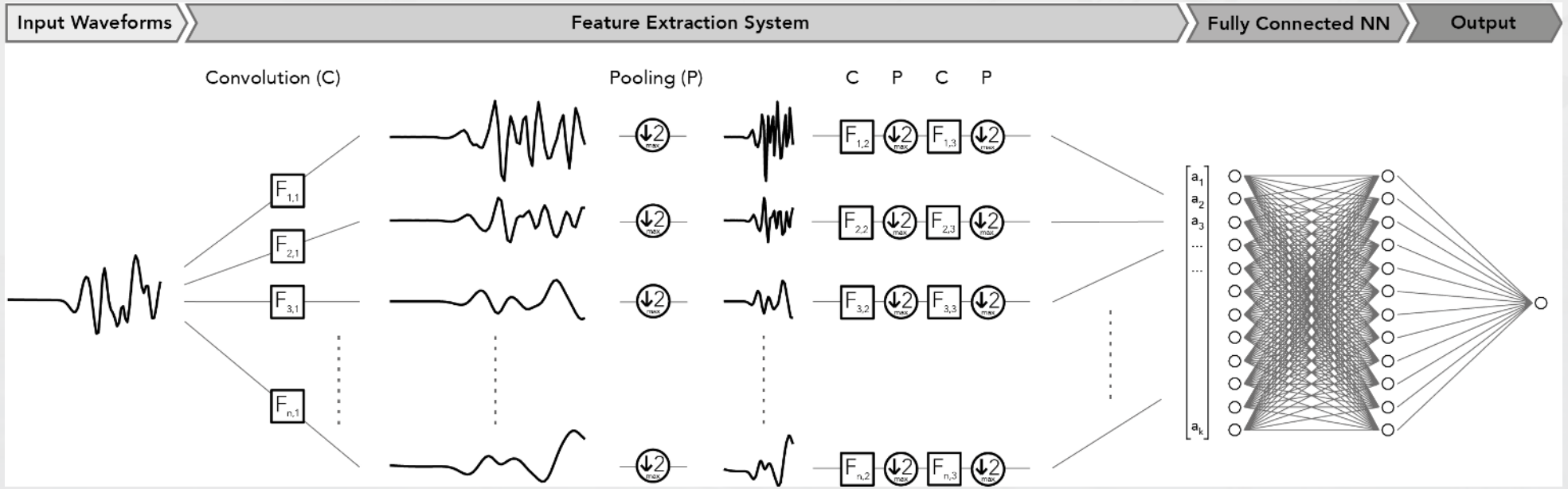


First motion polarities



- 95% overall classification accuracy
- Convnet assigns polarities where human analysts said 'unknown'

💣 We can check if these polarities are consistent with radiation pattern!

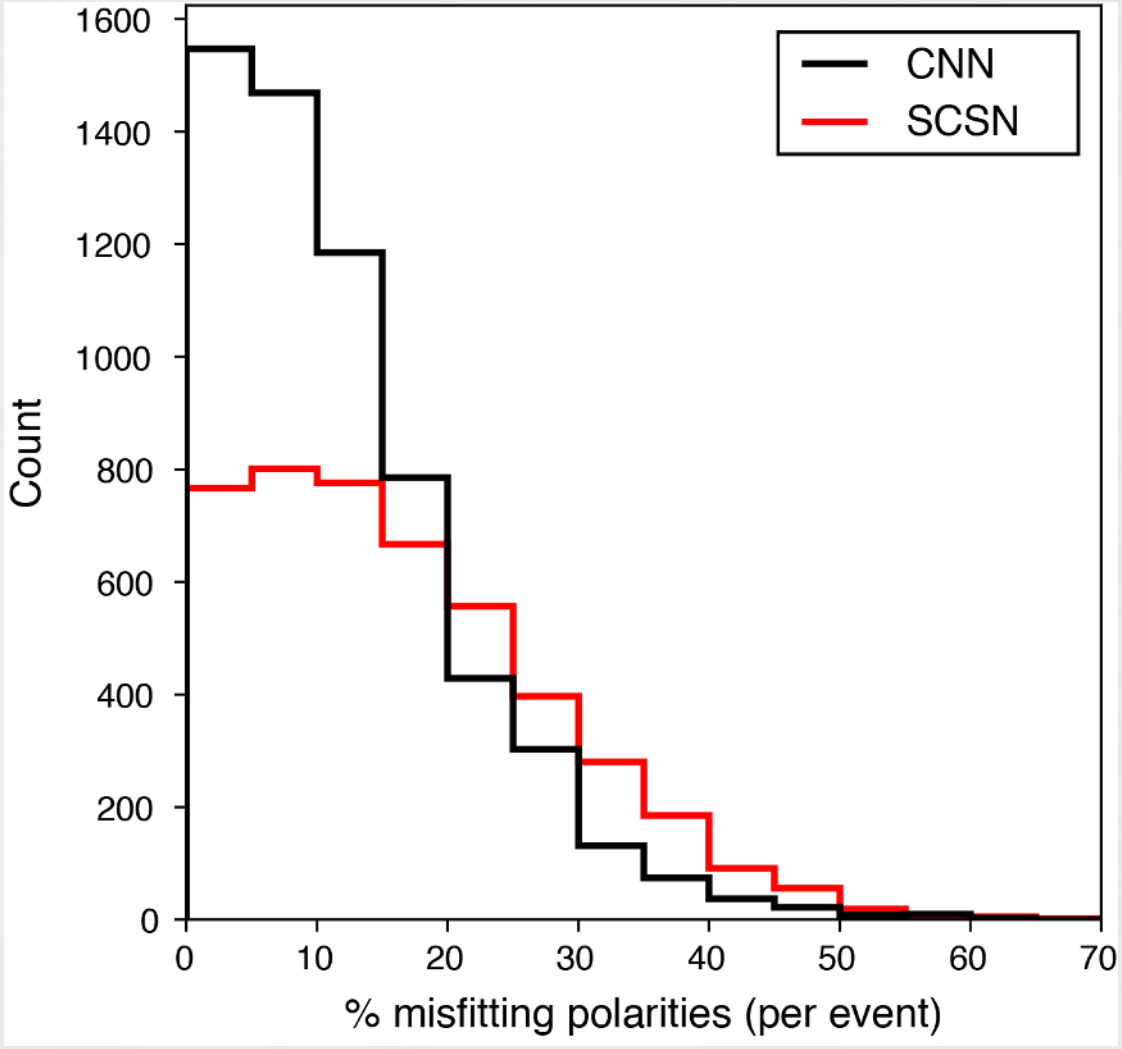


Ross et al. (2018), JGR

First motion polarities

Automated dataset has more polarities, and, mostly consistent ones

→ More and better FM
→ ConvNet outperforms human experts



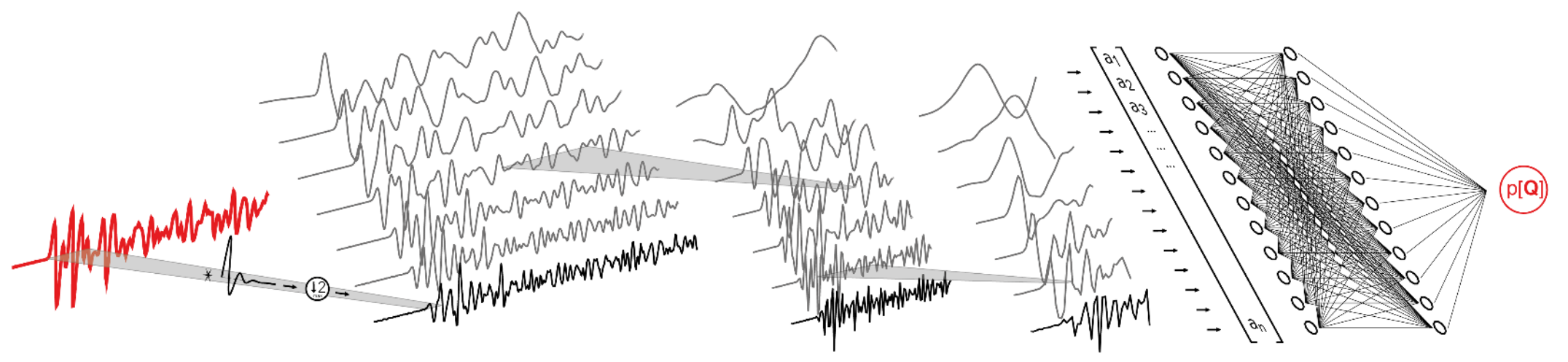
Ross et al. (2018), JGR

Discussion

- Supervised ML algorithms can be trained to perform various monitoring tasks
  - phase detection and classification
  - arrival times and first motion polarities
  - phase associations
  - signal denoising
  - hypocenter estimation
  - waveform modeling
  - …
- When we have enough data, or when we can train on synthetics, ML algorithms can approach or even exceed performance of human experts
- More powerful than standard methods, because …
  - More complex, more parameters, inherently non-linear
  - Flexible and high-dimensional decision boundaries
  - Optimized over very large data sets
  - Rigorous separation of training and test data
  - …

Why do the DL models work so well?

- **DL allows us to mimic human expert behavior** much better than conventional algorithms
- Before DL, automating tasks required much stronger simplifications
- DL algorithms, like us, learn from "experience"; on vastly more data
- We are essentially **building the *Virtual Seismologist*** *TH*

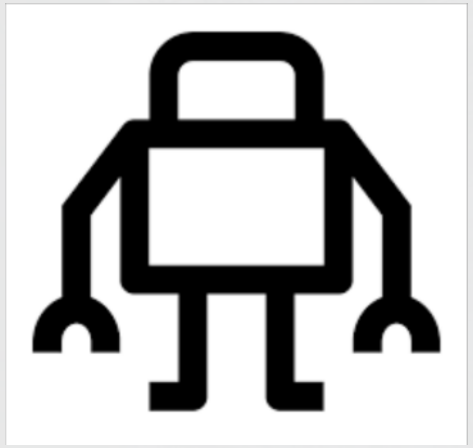


Conclusions

- When we have enough data, we can train powerful supervised deep learning models for a wide variety of tasks
- This afternoon we'll be looking into different classes of supervised deep learning models that go beyond classification and regression
  - Recurrent Neural Networks
  - Generative Models

The bigger picture

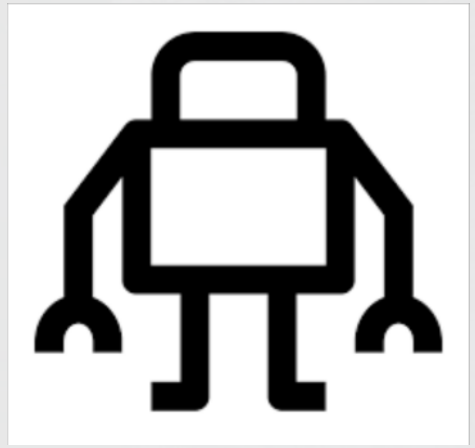
Supervised deep learning models are great at explicit and „shallow“ tasks, such as object classification. They do not, however, have a deep understanding of data content, e.g. the meaning of a scene in an image.



The bigger picture

Supervised deep learning models are great at explicit and „shallow“ tasks, such as object classification. They do not, however, have a deep understanding of data content, e.g. the meaning of a scene in an image.

Recurrent NN can write entire books with perfect grammar. But the text content is complete nonsense.



PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.

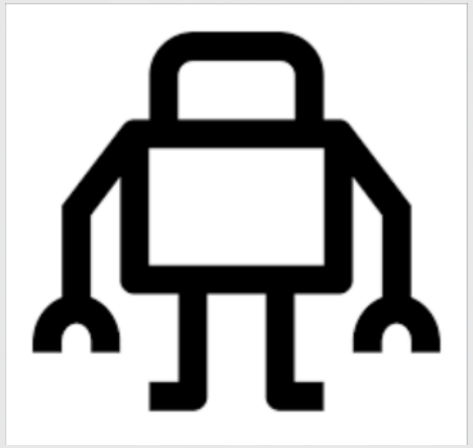
The bigger picture

Supervised deep learning models are great at explicit and „shallow“ tasks, such as object classification. They do not, however, have a deep understanding of data content, e.g. the meaning of a scene in an image.

Recurrent NN can write entire books with perfect grammar. But the text content is complete nonsense.

Most supervised models are explicitly trained to perform a specific task. This has little to do with human-like intelligence.

It is mostly unsupervised methods - which today perform much less well – that aim for human-like intelligence.



thank you.

References

- Ross, Z. E., Meier, M.-A., Hauksson, E., and T. H. Heaton (2018). Generalized Seismic Phase Detection with Deep Learning, *BSSA*
- Ross, Z. E., Meier, M.-A., and E. Hauksson (2018). P-wave arrival picking and first-motion polarity determination with deep learning, *JGR*
- Ross, Z. E., Yue, Y. Meier, M.-A., et al. (2019). A Deep Learning Approach to Seismic Phase Association, JGR
- Meier, M.-A., Ross, Z. , Ramachandran, A., Nair, S., et al. (2019). Reliable Signal/Noise Discrimination with Machine Learning, JGR
- Li, Z., Meier, M. - A., Hauksson, E., Zhan, Z. and Andrews, J., (2018). Machine Learning Seismic Wave Discrimination: Application to Earthquake Early Warning. *GRL*.
- Hauksson, E., Meier, M.-A., Ross, Z. E., and L. M. Jones (2017). Evolution of seismicity near the southernmost terminus of the San Andreas Fault: Implications of recent earthquake clusters for earthquake risk in southern California. *GRL*