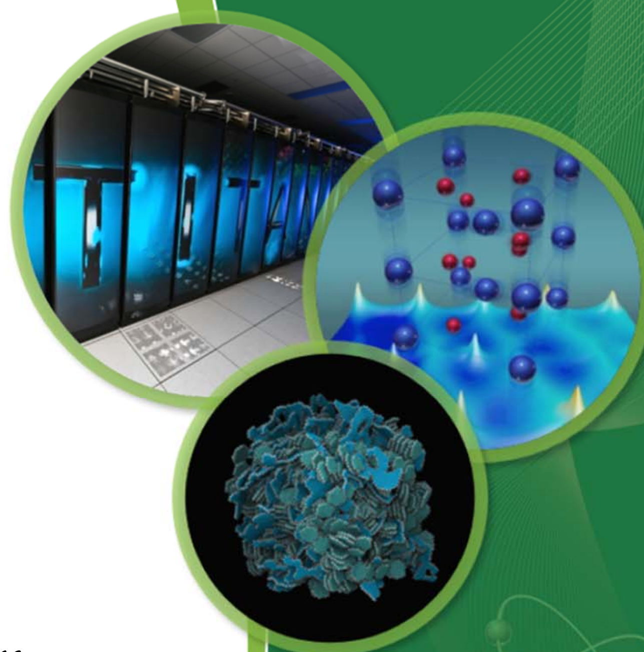
CNMS USER MEETING 2018

Deep Learning for Nanoscale Imaging



Maxim Ziatdinov

Institute for Functional Imaging of Materials
&
Computational Sciences and Engineering Division

August 15th 2018

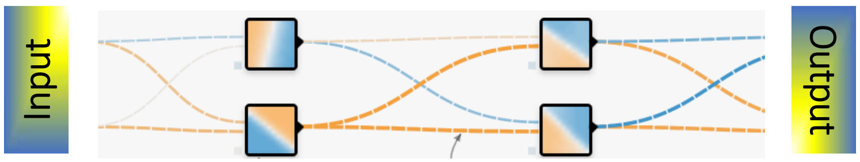
OAK RIDGE
National Laboratory

(Deep) Artificial Neural Networks

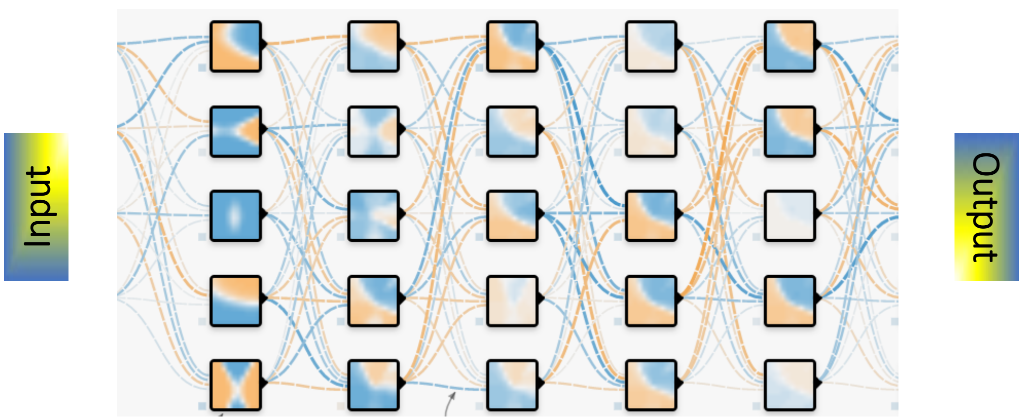
Network is trained to match input to output by adjusting weights.

It can then generalize to data samples that were not part of the training set.

Shallow Neural Network
Small number of processing units/layers



Deep Neural Network (aka deep learning)
Large number of processing units/layers

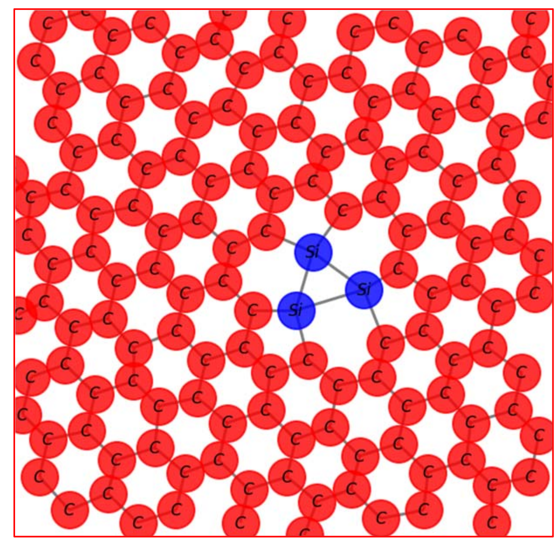


The deeper networks are able to capture more abstract, high-level features

Macro-world vs. Nano-world:

Macro-world

Nano-world

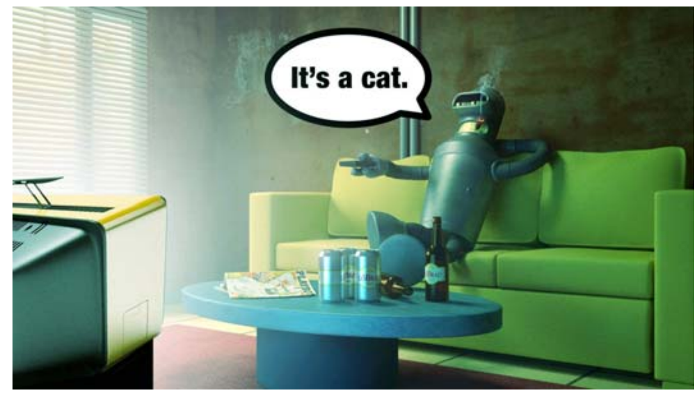


What has deep learning achieved so far

- ✓ Human-level image classification
- ✓ Near-human-level speech recognition
- ✓ Near-human-level autonomous driving
- ✓ Ability to answer natural language questions
- ✓ Improved search results on the web
- ✓ Digital assistants (Alexa, Google Now)
- ✓ Superhuman(!) Go playing

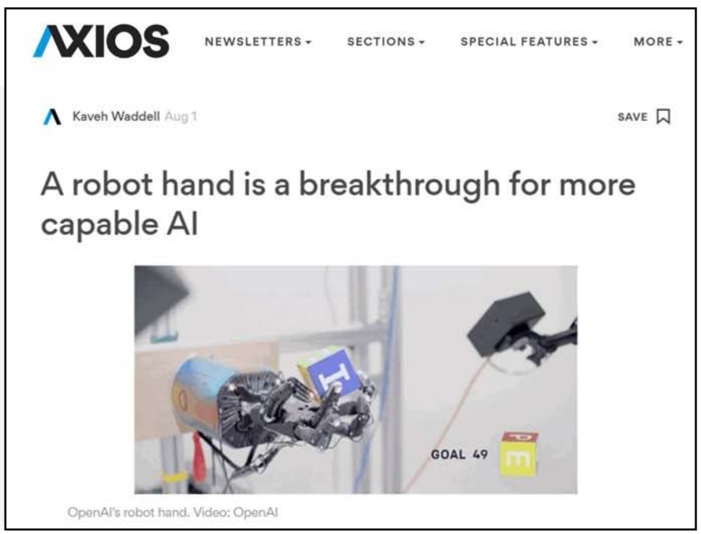
…

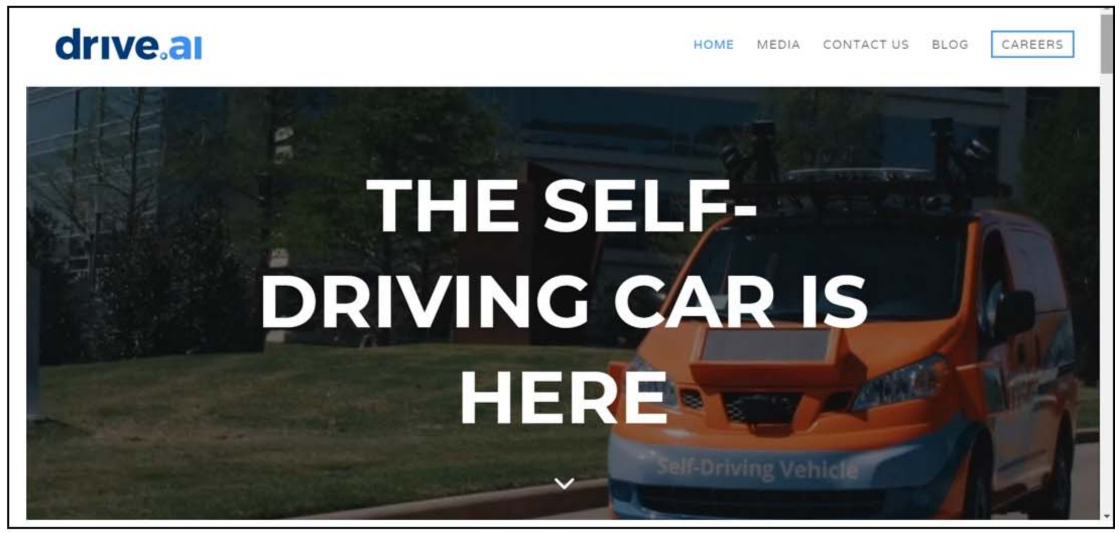
The ultimate goal of artificial intelligence? ☺



Deep learning completely **automates** the most crucial step in machine learning – **feature engineering**

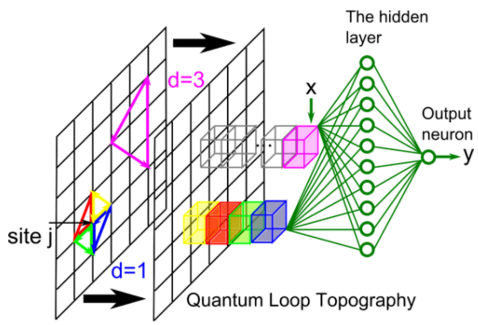
It also has generated lots of headlines…





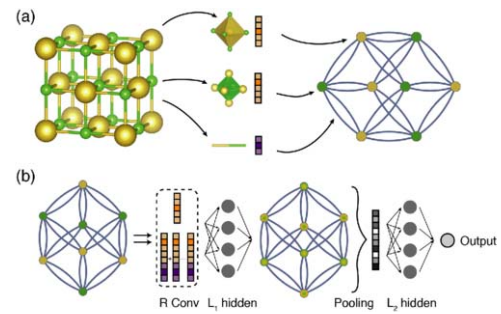
Deep learning in theoretical physics

Quantum loop topography



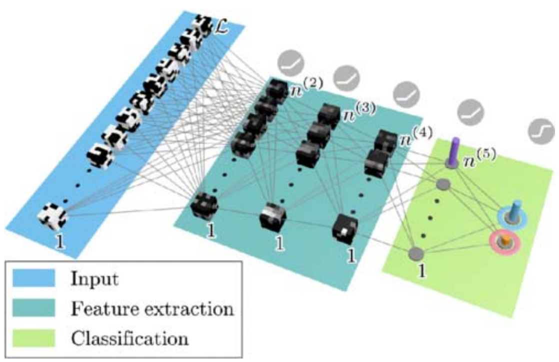
Phys. Rev. Lett. **118**, 216401 (2017)

Crystal graph convolutional networks



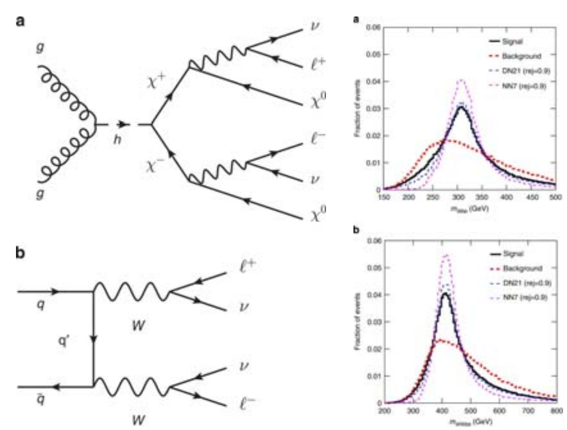
Phys. Rev. Lett. **120**, 145301 (2018)

Deep learning phases of strongly correlated fermions



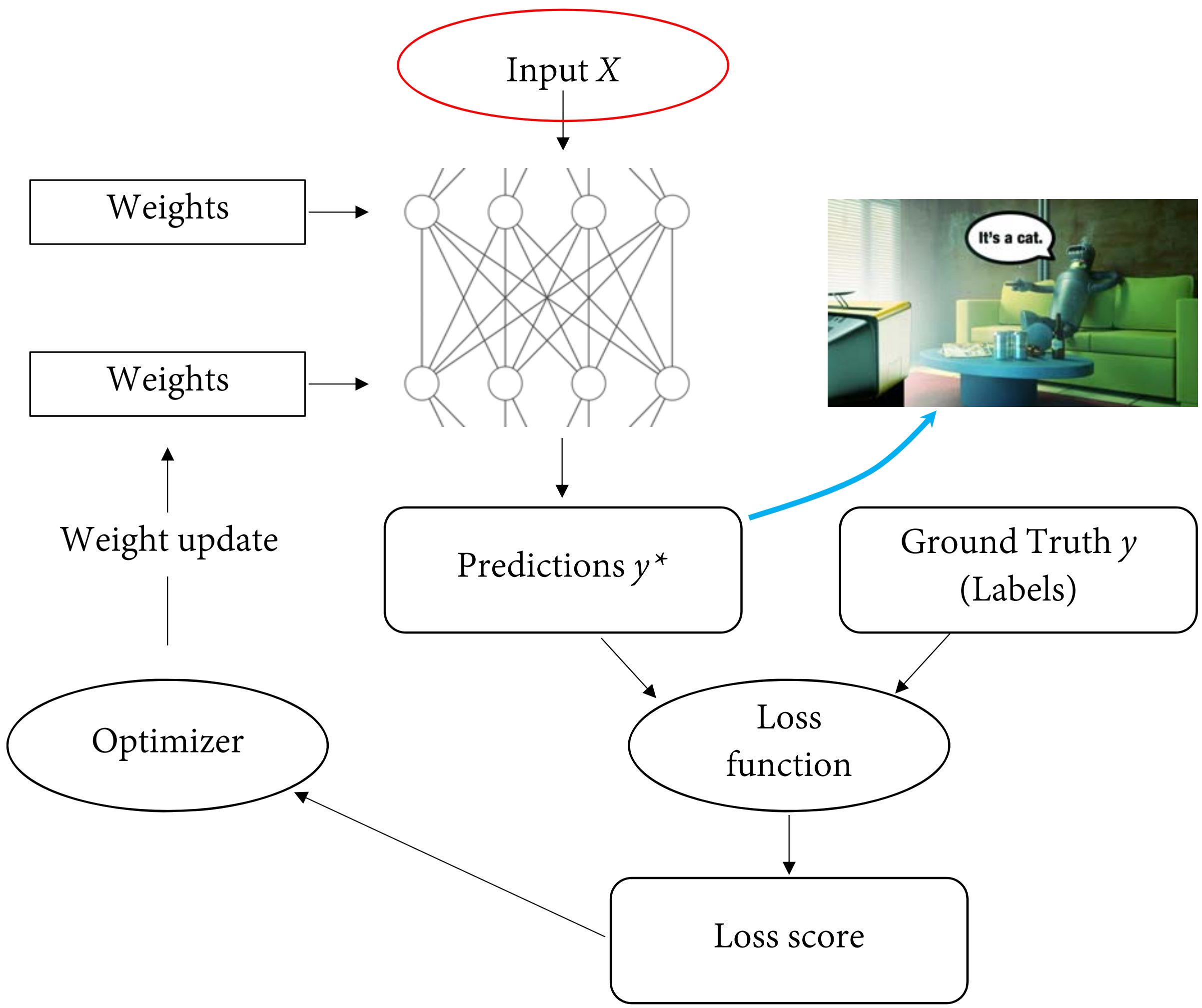
Phys. Rev. X **7**, 031038 (2017)

Deep learning applications to LHC data



Nat. Commun. **5**, 4308 (2014)

Artificial Neural Network in a Nutshell



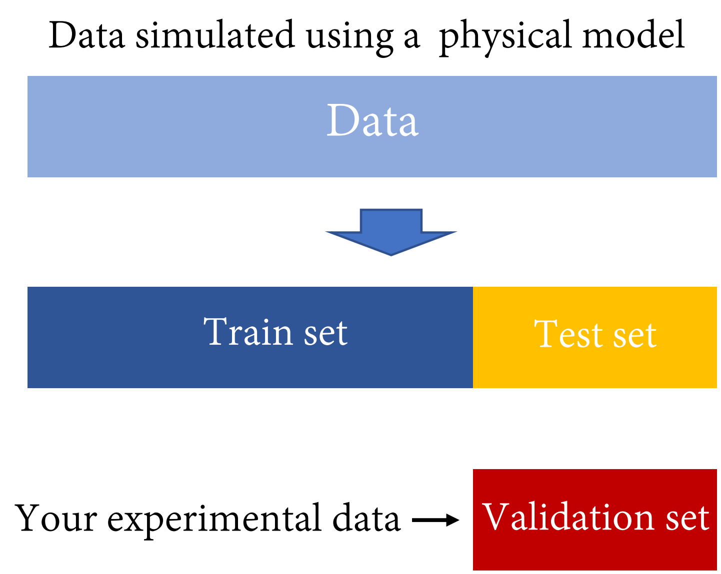
3 key principles

- The network is parameterized by its weights
- A loss function measures the quality of the network's output
- The loss score is used as a feedback to adjust the weights

Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*; The MIT Press, 2016; Chollet, F.: *Deep Learning with Python*; Mannings Publication Co., 2018

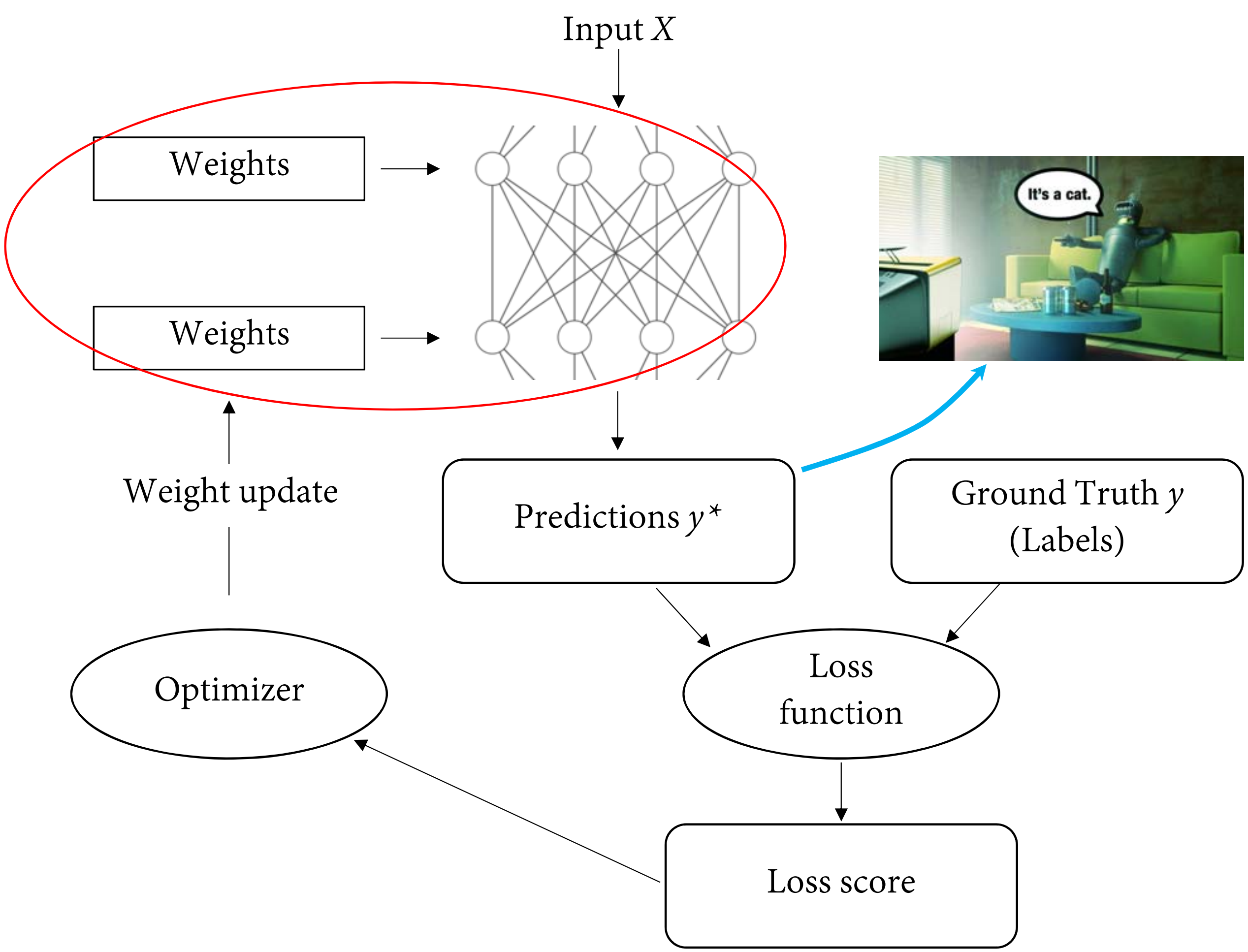
Input Data

- Vector data:
  2D tensor: samples, features.
- Timeseries data:
  3D tensor: samples, timesteps, features
- Image data:
  4D tensor: samples, width, height, channels
- Video data:
  5D tensor: samples, frame, width, height, channels



Notice that deep learning models do not process the entire dataset at once. Instead, the data is broken into small batches (typical size between several dozens and several hundreds of samples per batch)

Artificial Neural Network in a Nutshell



3 key principles

- The network is parameterized by its weights
- A loss function measures the quality of the network's output
- The loss score is used as a feedback to adjust the weights

“Layers” of Neural Network

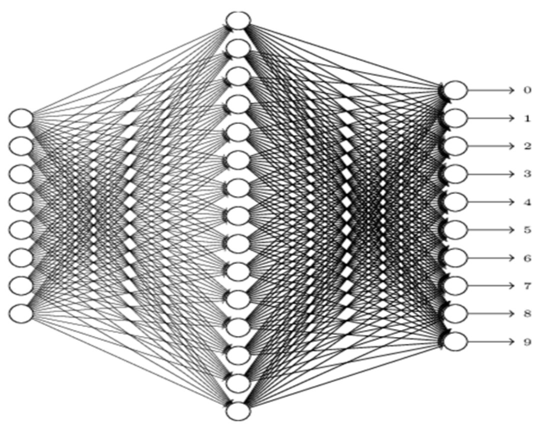
Each neural layer transforms its input data as follows:

Output = relu(dot(W, input) + b)

W (weights) and b (bias) are attributes of the layer.

They contain information learned by the network from exposure to training data.

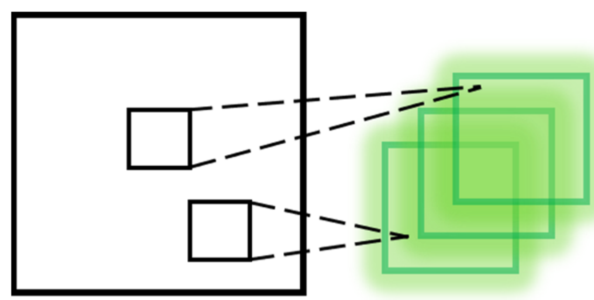
“Standard” fully connected layers



Convolutional Layers

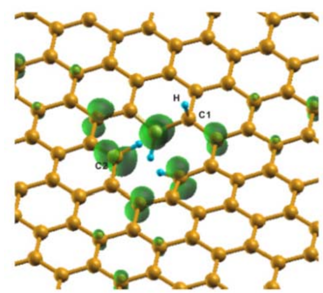
Input image

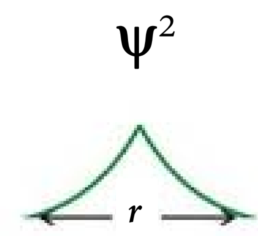
Output feature map



Studying physics locally

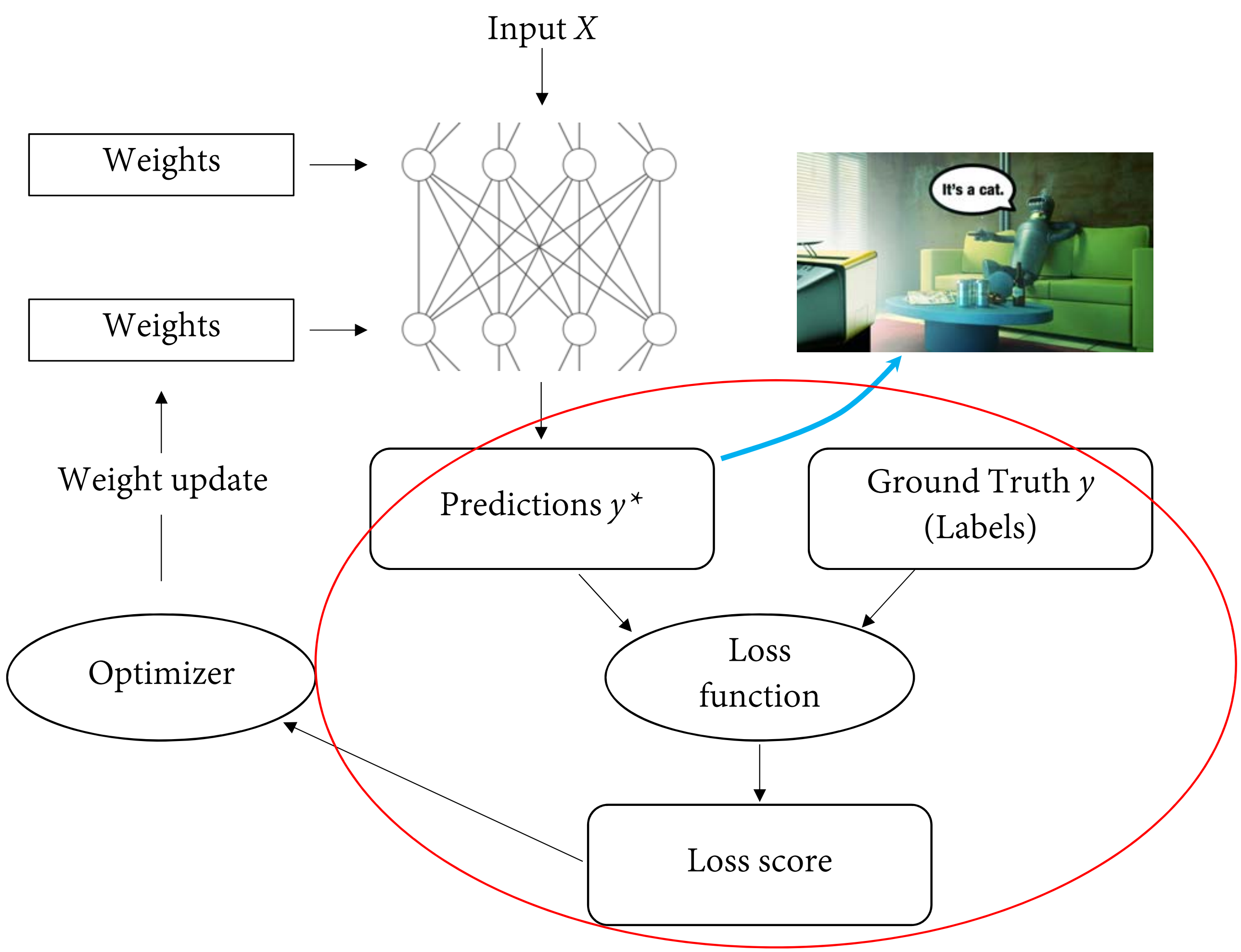
The effect of defects on materials structure & electronic properties is usually localized.





- Standard NN is not suitable for image analysis (e.g. image of 30 px*30 px would require 0.5 million parameters and 900 inputs)
- Convolutional NN uses proximity/locality to restrict number of connections

Artificial Neural Network in a Nutshell



3 key principles

- The network is parameterized by its weights
- A loss function measures the quality of the network's output
- The loss score is used as a feedback to adjust the weights

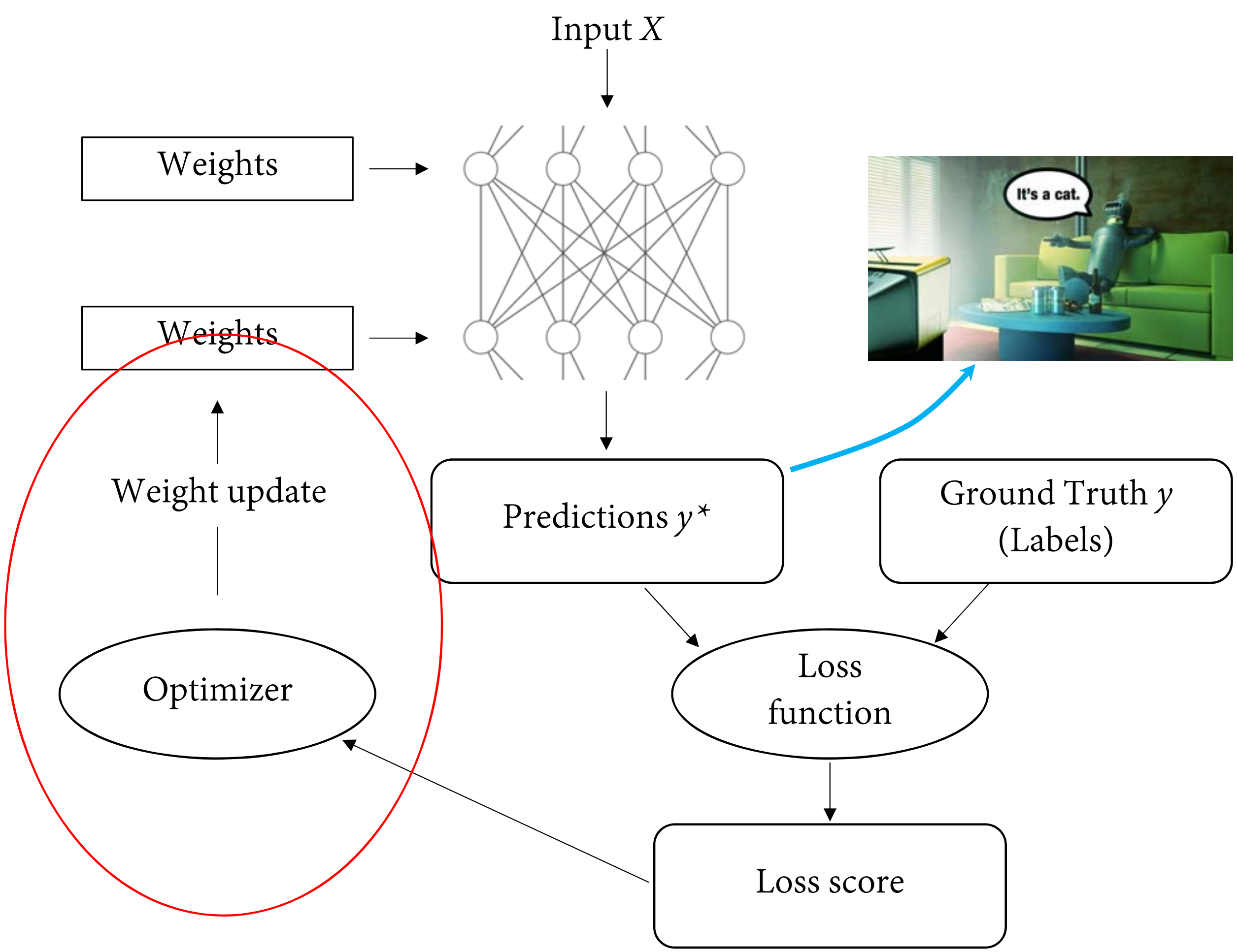
Loss Functions

Loss function is the quantity that will be minimized during training (measure of success)

- Neural network with multiple outputs have multiple loss functions
- For gradient decent process, all losses are combined via averaging into a single scalar quantity

- Different loss functions for different problems:
  - 2-class classification problem: binary cross-entropy
  - Multi-class classification problem: categorical cross-entropy
  - Regression problem: mean-squared problem
  - Sequence-learning problem: connectionist temporal classification
  - …
  - Physical-problem-based loss functions？

Artificial Neural Network in a Nutshell



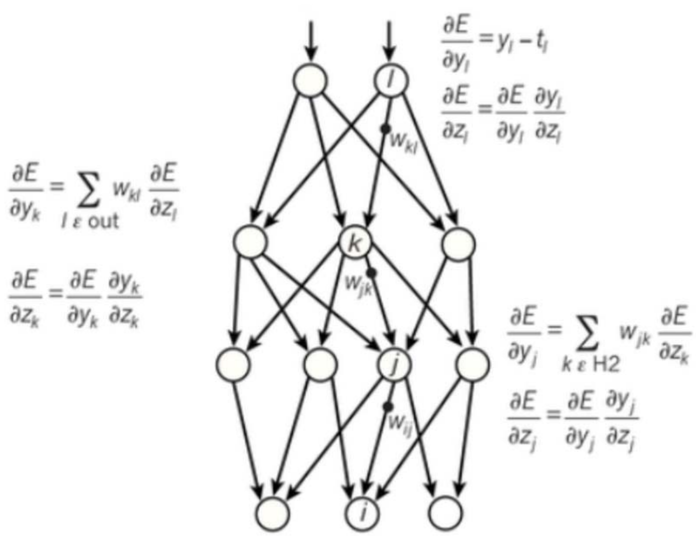
3 key principles

- The network is parameterized by its weights
- A loss function measures the quality of the network's output
- The loss score is used as a feedback to adjust the weights

Optimizing neural network parameters (weights)

Forward propagation

Back propagation



Stochastic gradient decent

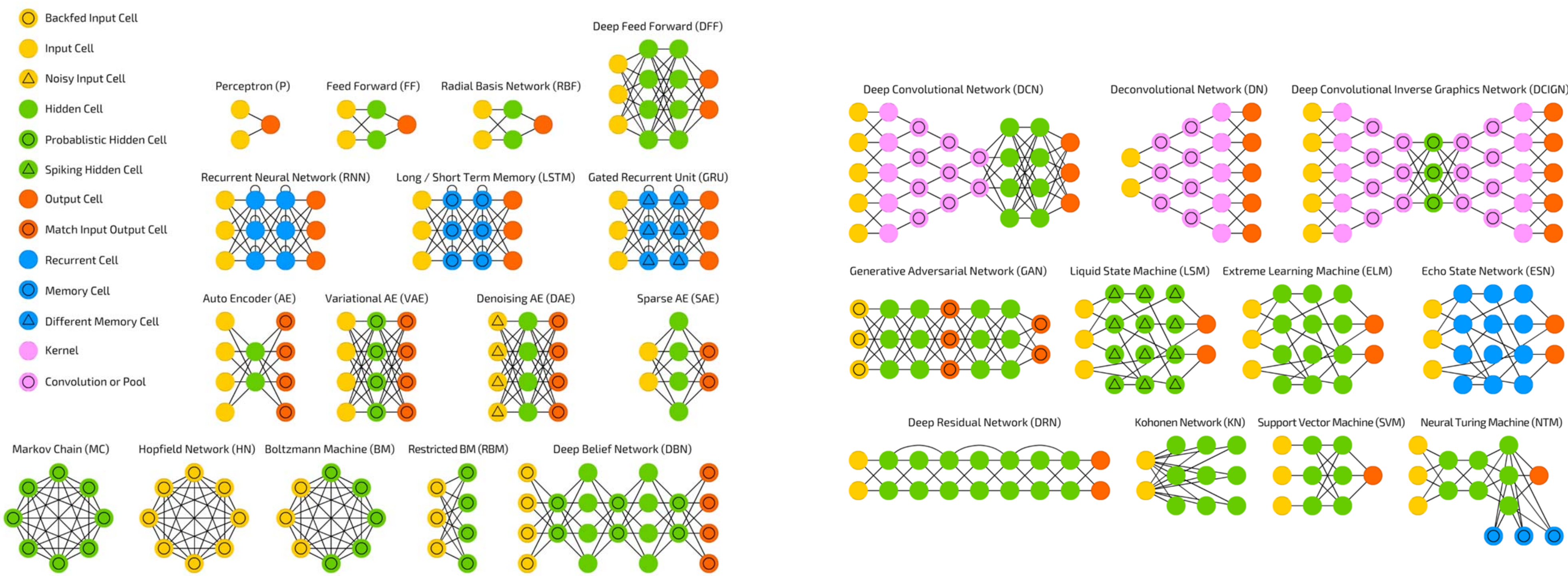
- Draw a batch of training samples x and targets (labels) y
- Run the network to obtain predictions y*
- Compute minibatch between y and y* (loss function)
- Compute the gradient of the loss with regard to the network's parameters (a backward pass)
- Move the parameters a little in the opposite direction from the gradient to reduce the loss on the batch a bit.

Backpropagation algorithm is the result of applying the chain rule to the computation of the gradient values of a neural network. It works backwards from the top layers to the bottom layers applying the chain rule to compute the contribution that each parameter had in the loss values.

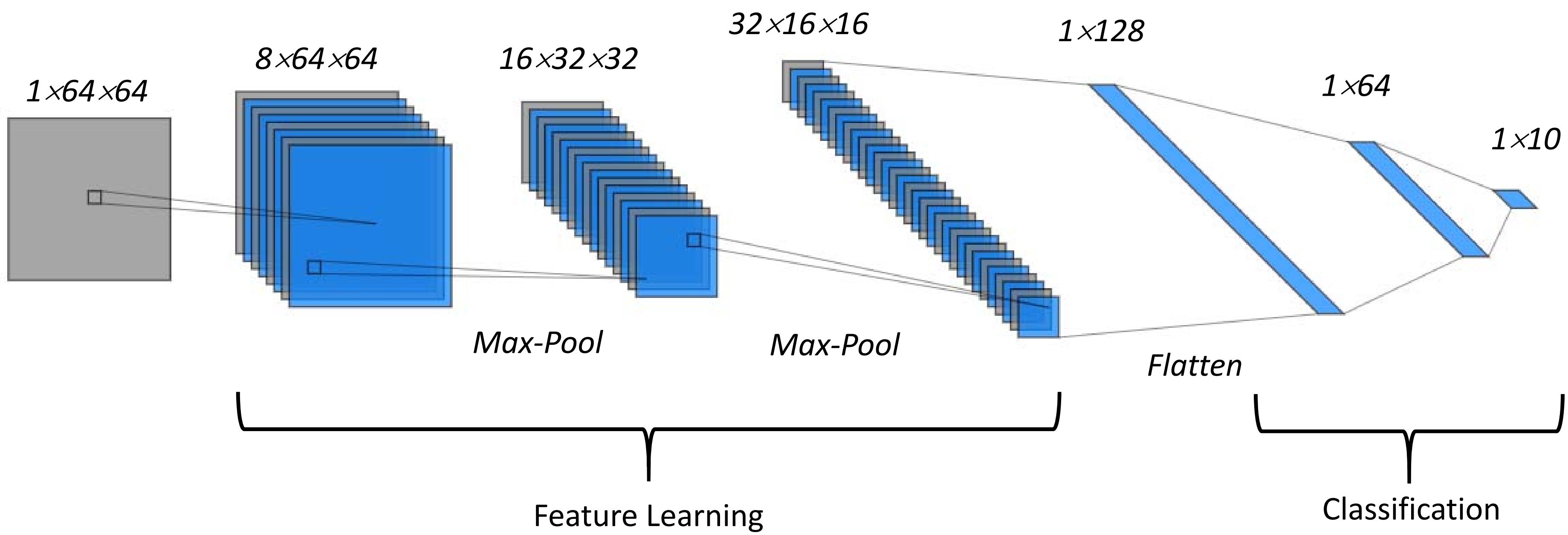
Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*; The MIT Press, 2016; Chollet, F.: *Deep Learning with Python*; Mannings Publication Co., 2018

A mostly complete chart of

Neural Networks



Convolutional neural network for image processing



Convolutional neural networks (CNN) represent one of the key examples of a successful application of neuroscientific principles to the field of machine leaning.

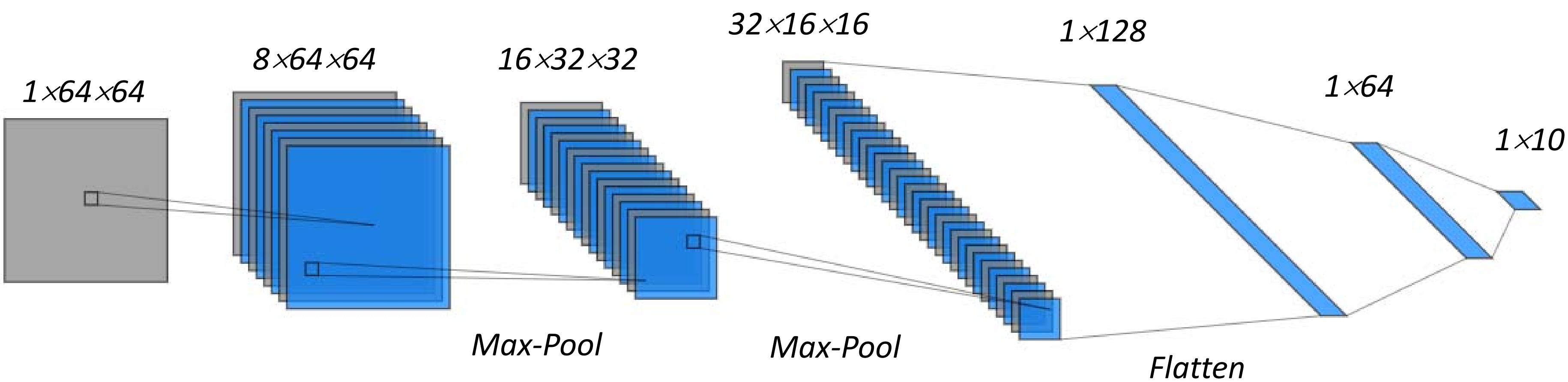
Local connections: local values are correlated

Shared weights: local statistics is invariant to location

Pooling: merge similar features into one

Image of CNN is generated using network drawing tools at http://alexlenail.me/NN-SVG

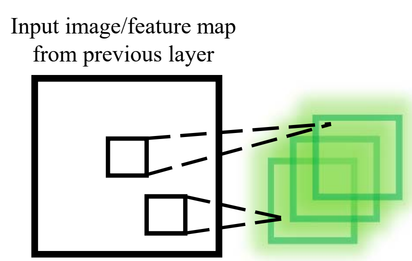
Convolutional neural network for image processing



Convolutional Layer

- Accepts a volume of size $\boldsymbol{w}_1 \times \boldsymbol{h}_1 \times \boldsymbol{d}_1$
  - 4 hyperparameters:
    - $\boldsymbol{k}$: number of filters
    - $\boldsymbol{f}$: size of filters (e.g. 3×3, 5×5, etc.)
    - $\boldsymbol{s}$: stride
    - $\boldsymbol{p}$: amount of zero padding
- Outputs a volume of size $\boldsymbol{w}_2 \times \boldsymbol{h}_2 \times \boldsymbol{d}_2$:
  - $\boldsymbol{w}_2 = (\boldsymbol{w}_1 - \boldsymbol{f} + 2\boldsymbol{p})/\boldsymbol{s} + 1$
  - $\boldsymbol{h}_2 = (\boldsymbol{h}_1 - \boldsymbol{f} + 2\boldsymbol{p})/\boldsymbol{s} + 1$
  - $\boldsymbol{d}_2 = \boldsymbol{k}$

Extracting local info ↔ physics



Activation Layer

- Increases non-linearity of the network without affecting receptive fields of convolutional layers
- Type of non-linear activations: ReLU, Leaky ReLU, Tanh, ELU, softmax, etc…

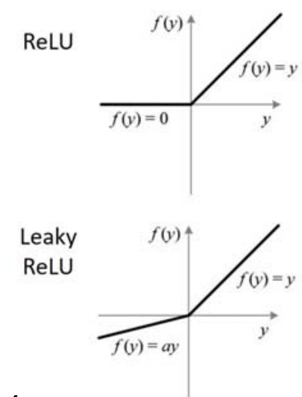
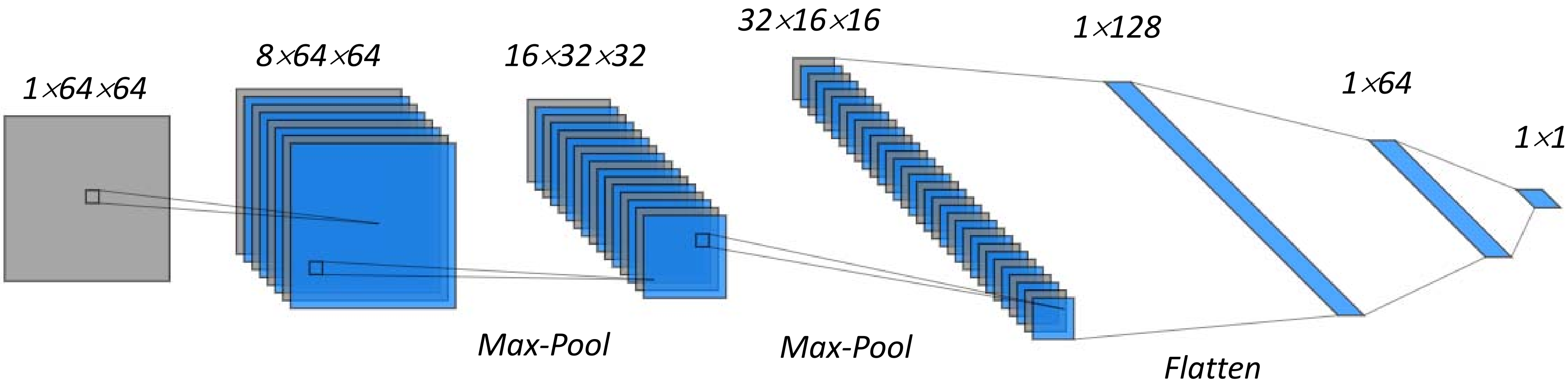


Image of CNN is generated using network drawing tools at http://alexlenail.me/NN-SVG

Convolutional neural network for image processing



Pooling operation

- Accepts a volume of size $w_1 \times h_1 \times d_1$
  - 2 hyperparameters:
    - f: size (e.g. 2×2)
    - s: stride
- Outputs a volume of size $w_2 \times h_2 \times d_2$:
  - $w_2 = (w_1 - f)/s + 1$
  - $h_2 = (h_1 - f)/s + 1$
  - $d_2 = d_1$

Max Pooling

29	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

2 x 2
pool size

100	184
12	45

Average Pooling

31	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

2 x 2
pool size

36	80
12	15

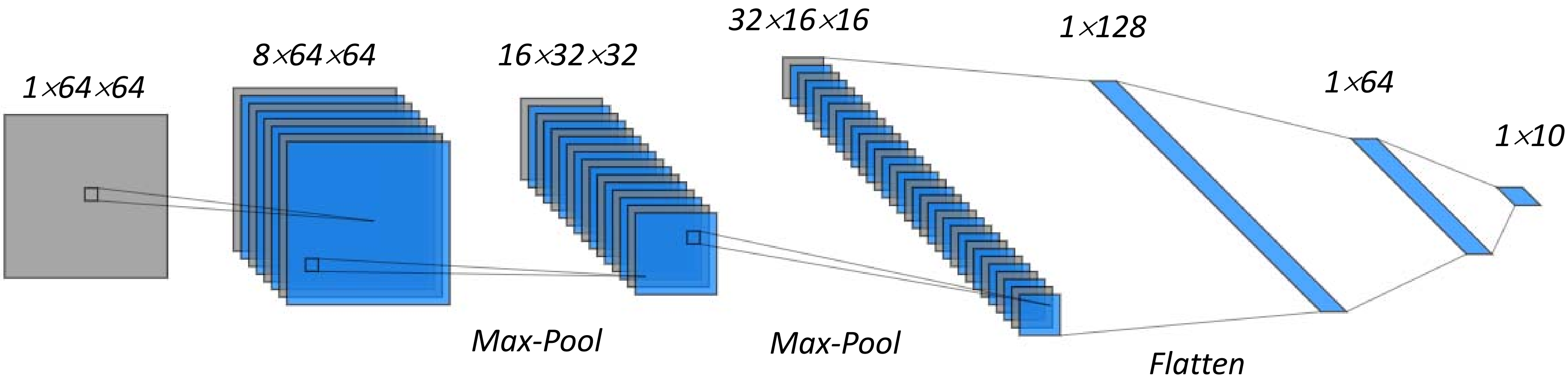
Softmax

- Softmax activation layer is a special kind of layer used at the end of network to produce a discrete probability distribution vector

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

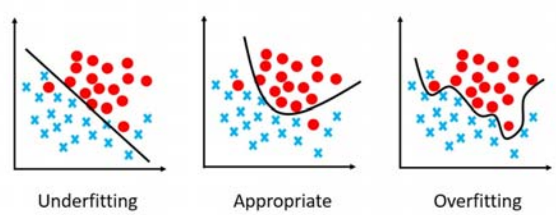
Image of CNN is generated using network drawing tools at http://alexlenail.me/NN-SVG

Convolutional neural network for image processing



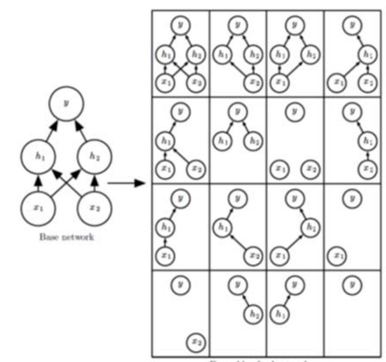
Regularization

- Used to prevent overfitting
- Types of regularization:
  - L1/L2
  - Dropout
  - Batch normalization
  - Max norm constraint



Dropout layer as regularization

- Prevents overfitting by reducing correlations between neurons
- During training, randomly ignores activations by probability ***p***
- During testing, uses all activations but scaled by ***p***
- Can be in principle used for uncertainty quantification



Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*; The MIT Press, 2016; Chollet, F.: *Deep Learning with Python*; Mannings Publication Co., 2018

Importance of Training Set

VGG-19 pre-trained on ImageNet

Top 3 predictions

Very close match

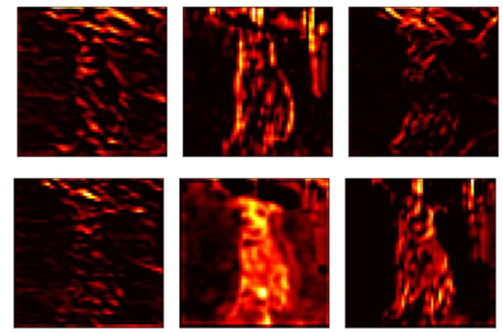
Staffordshire bullterrier 43 %

American pit bull terrier 23 %

Basenji 11 %

Meet Duffy

Some feature maps
from intermediate layers



Top 3 predictions

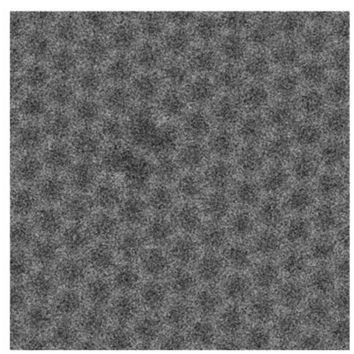
Doormat 22 %

Dishrag 5 %

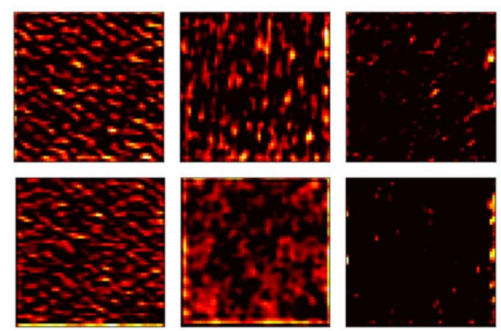
Poncho 4 %

These are not the features we are interested in!

STEM of atomic defect

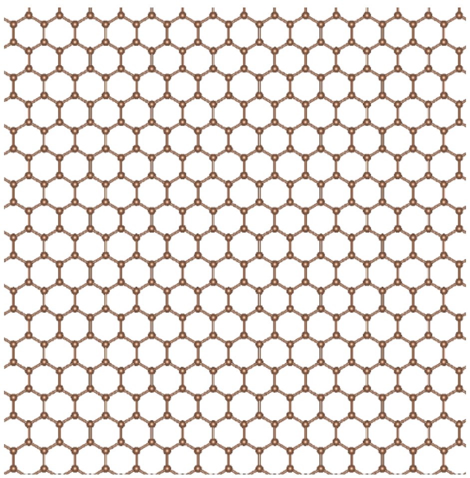


Some feature maps
from intermediate layers



Training Set: Teaching atomic imaging concepts to neural networks

Atomic coordinates
("ground truth")



QM electron scattering

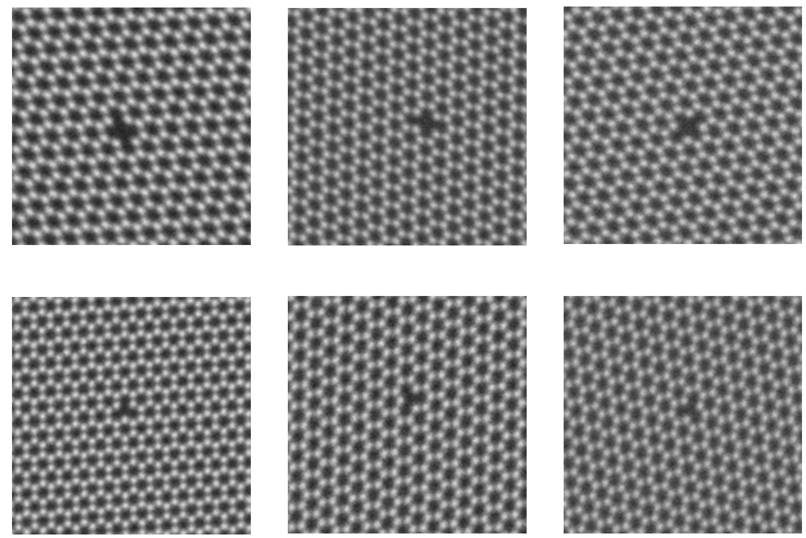
→

QM electron tunneling

Simulated (and augmented) data

0
1
0
0
0

0
0
1
0
0

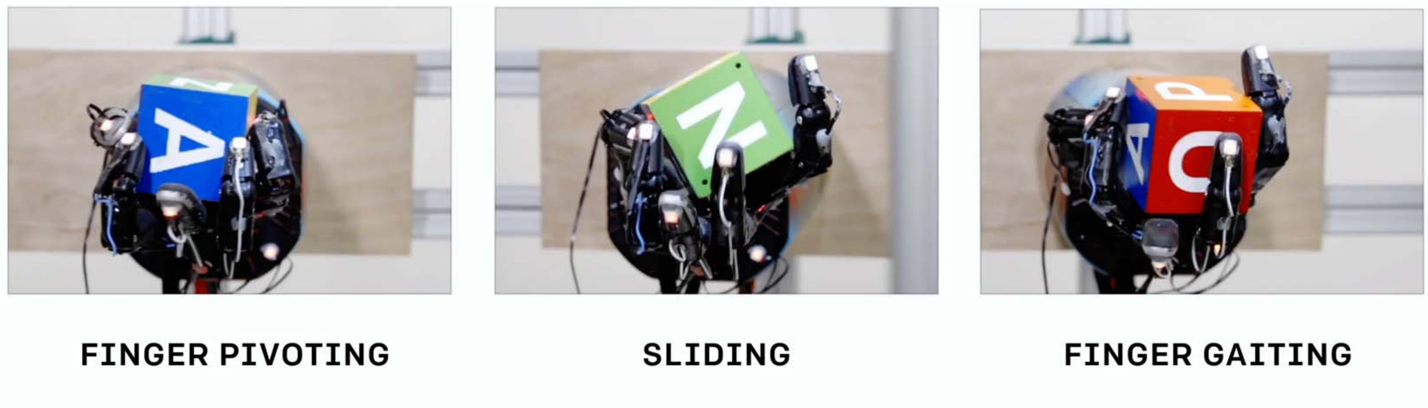


Data augmentation:

- random shifts
- rotations
- zoom-in/out
- local atom displacements
- shear transformations
- adding noise, blurring

← **to account for variations in imaging conditions during real data acquisition**

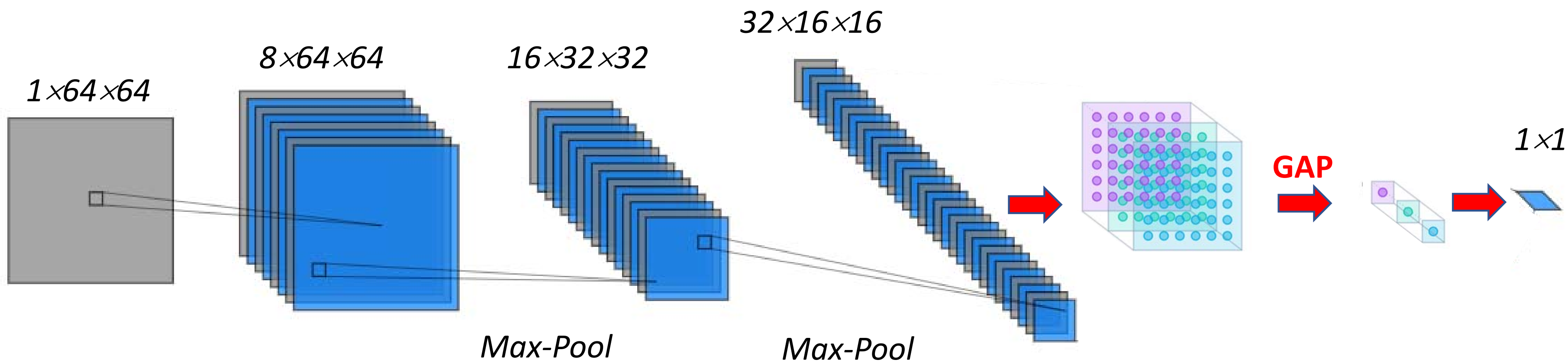
Simulations-to-physical-world knowledge transfer



https://blog.openai.com/learning-dexterity/

“Dactyl learns to solve the object reorientation task entirely in simulation without any human input. After this training phase, the learned policy works on the real robot without any fine-tuning.”

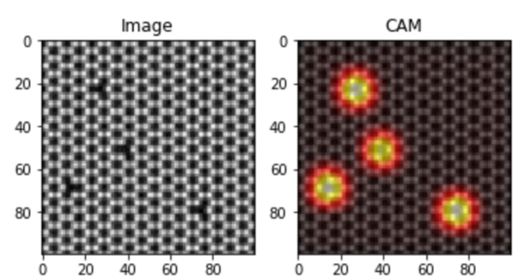
CNN-based classification of atomic structures



GAP representation from alexisbcook.github.io

Example: CNN + dense layers *vs.* CNN+GAP

```
defect type: vacancy
defect coordinates (x, y): [(81.0, 74.0), (69.0, 14.0), (53.0, 38.0), (22.0, 26.0)]
```



(Free) localization of atomic-scale objects in image-level classification scheme!

Class activation maps (CAM)

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

Softmax weights

Final activation layer

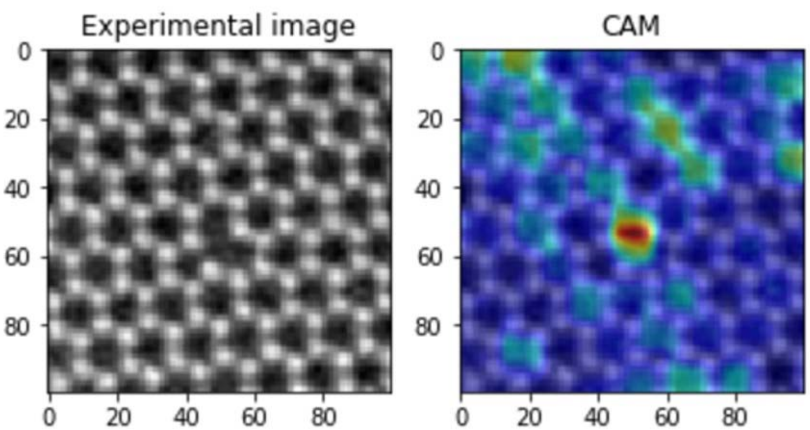
Examples (notebooks)

Localization of lattice defects via class activation maps technique
Real (experimental) data

Good generalization ability: We use our network to look for certain types of defects beyond the simple synthetic system and even material (!) on which it was trained.

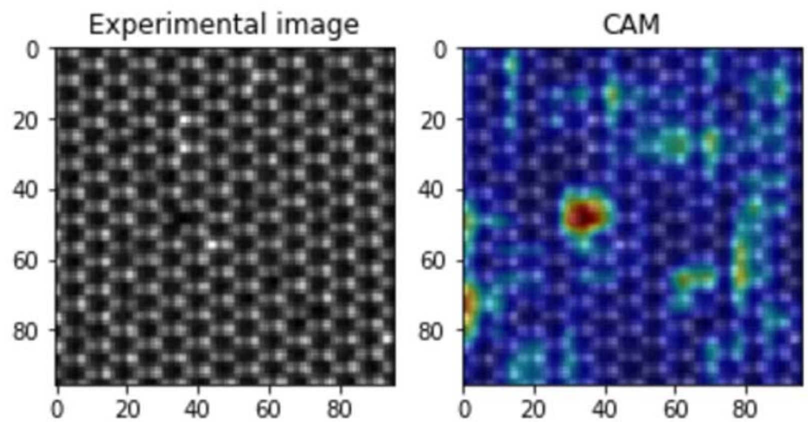
Graphene

Defect type: Vacancy
Defect coordinates: (x, y) = (50, 54)



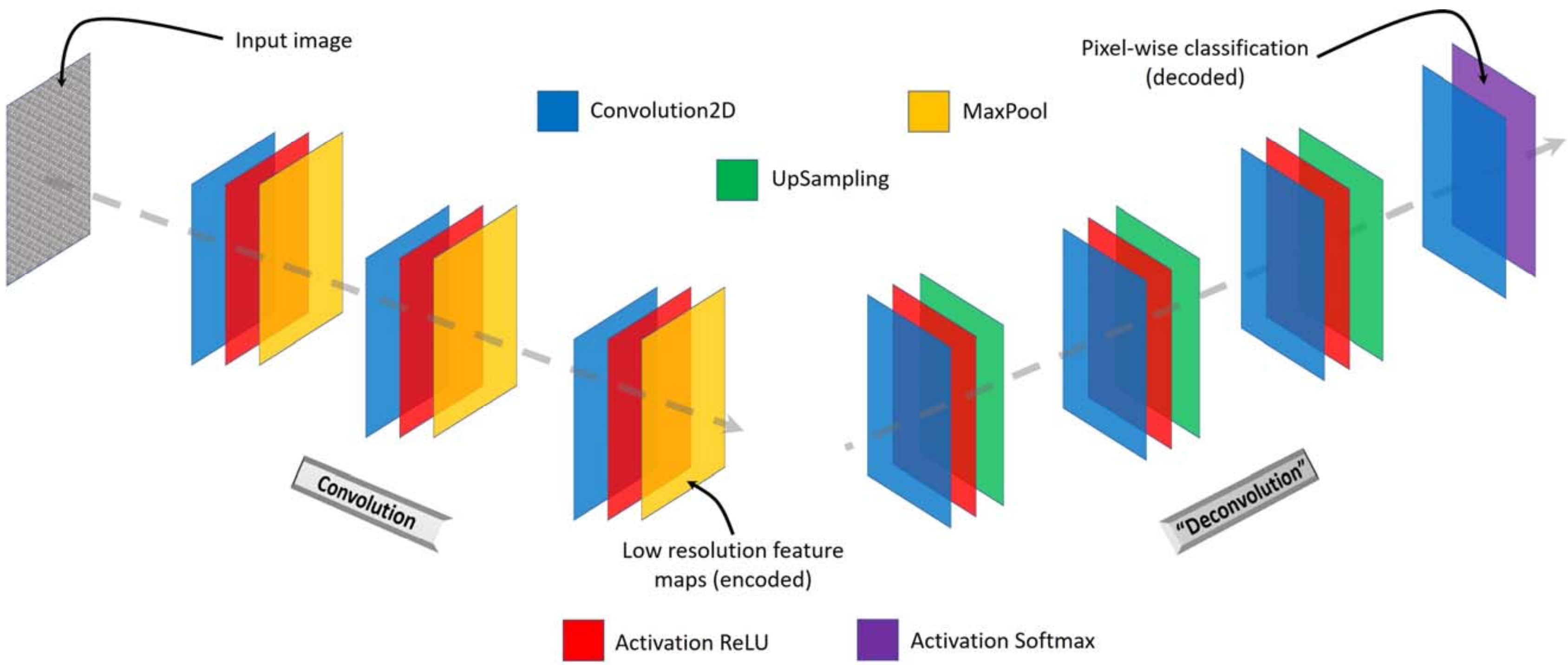
$Mo_{1-x}W_xSe_2$

Defect type: Vacancy
Defect coordinates: (x, y) = (34, 48)



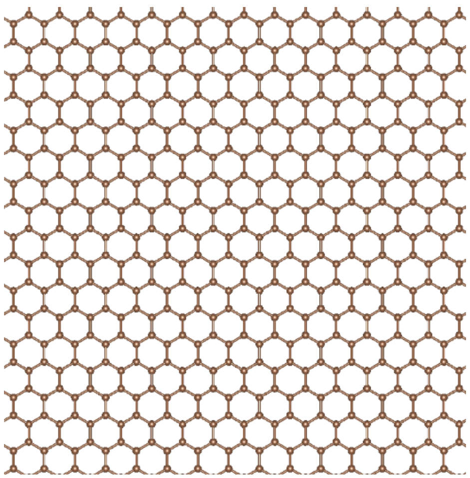
Fully convolutional neural network for pixel-wise classification

- It is crucial to be able to learn precise location of atomic species and/or defects
- The output of the DL model must provide the probability of each pixel belonging to certain atom and/or defect and its resolution/size should match that of the input image



Training Set: Teaching atomic imaging concepts to neural networks

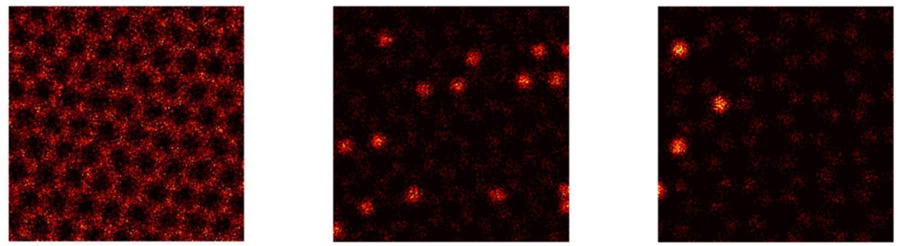
Atomic coordinates
(“ground truth”)



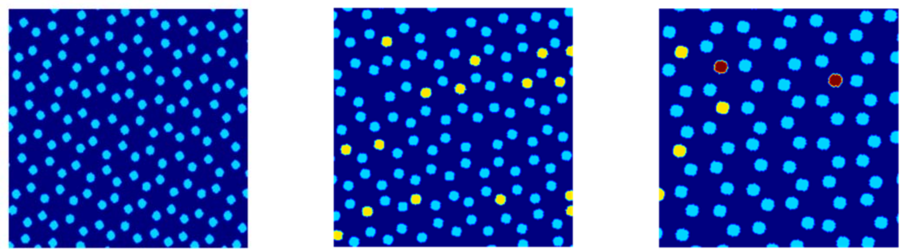
QM electron scattering

QM electron tunneling

Simulated data



Ground Truth



One may count vacancy as a separate class

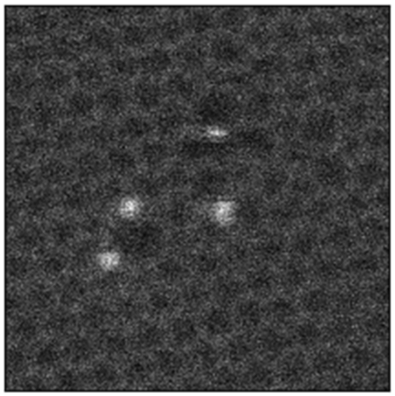
Data augmentation:

- random shifts
- rotations
- zoom-in/out
- local atom displacements
- shear transformations
- adding noise, blurring

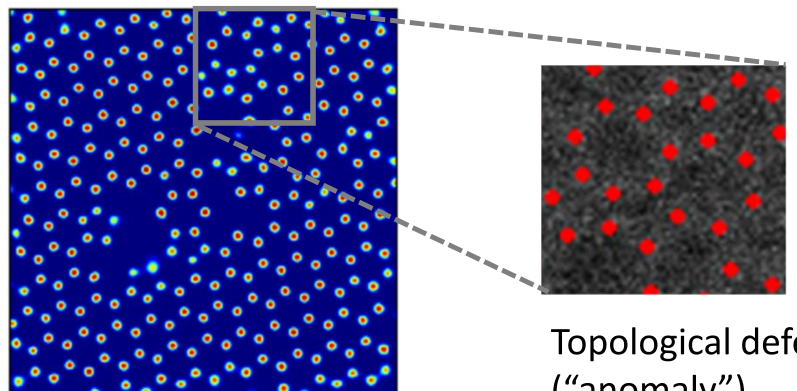
← **to account for variations in imaging conditions during real data acquisition**

Finding lattice atoms and defects in graphene via deep learning

Experimental (raw)

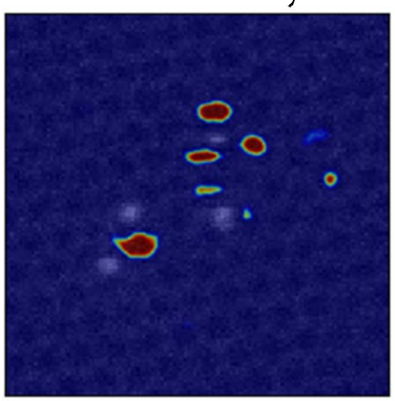


Class ‘lattice’

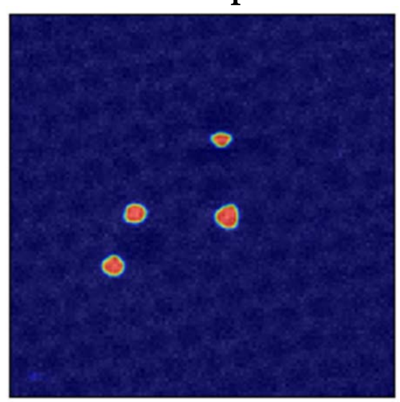


Topological defects
(“anomaly”)

Class ‘vacancy’



Class ‘dopant’

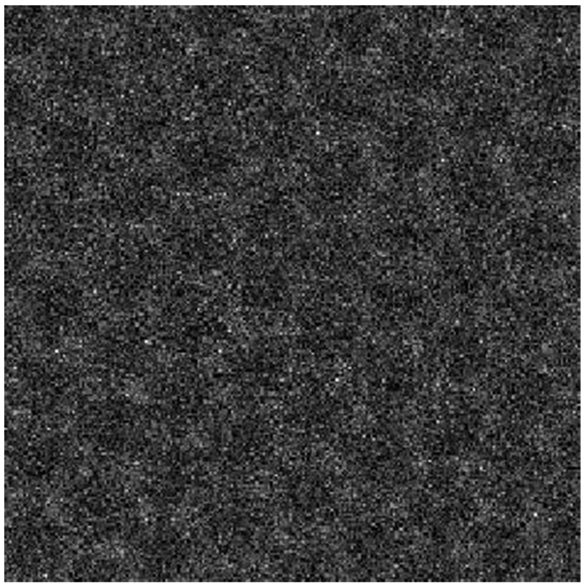


How does DL model find all the atoms?

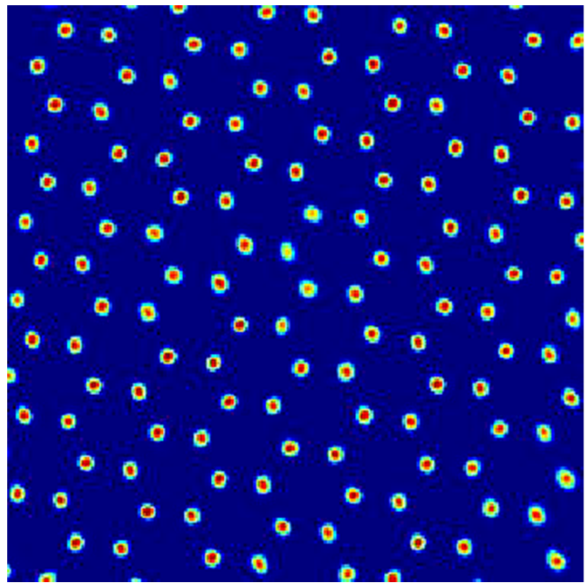
When tasked with finding atomic positions the deep neural network considers what is known as deep, high-level features in the image such as the shape of an individual hexagon, as well as shapes of its neighbors.

How convolutional neural network sees the atomic world

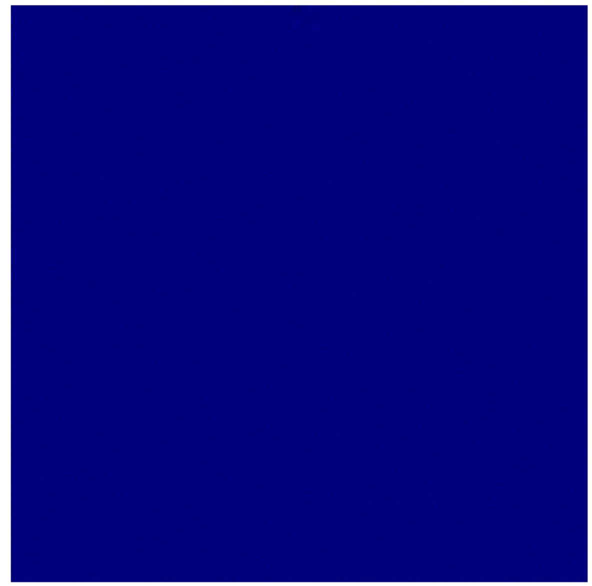
Experimental data



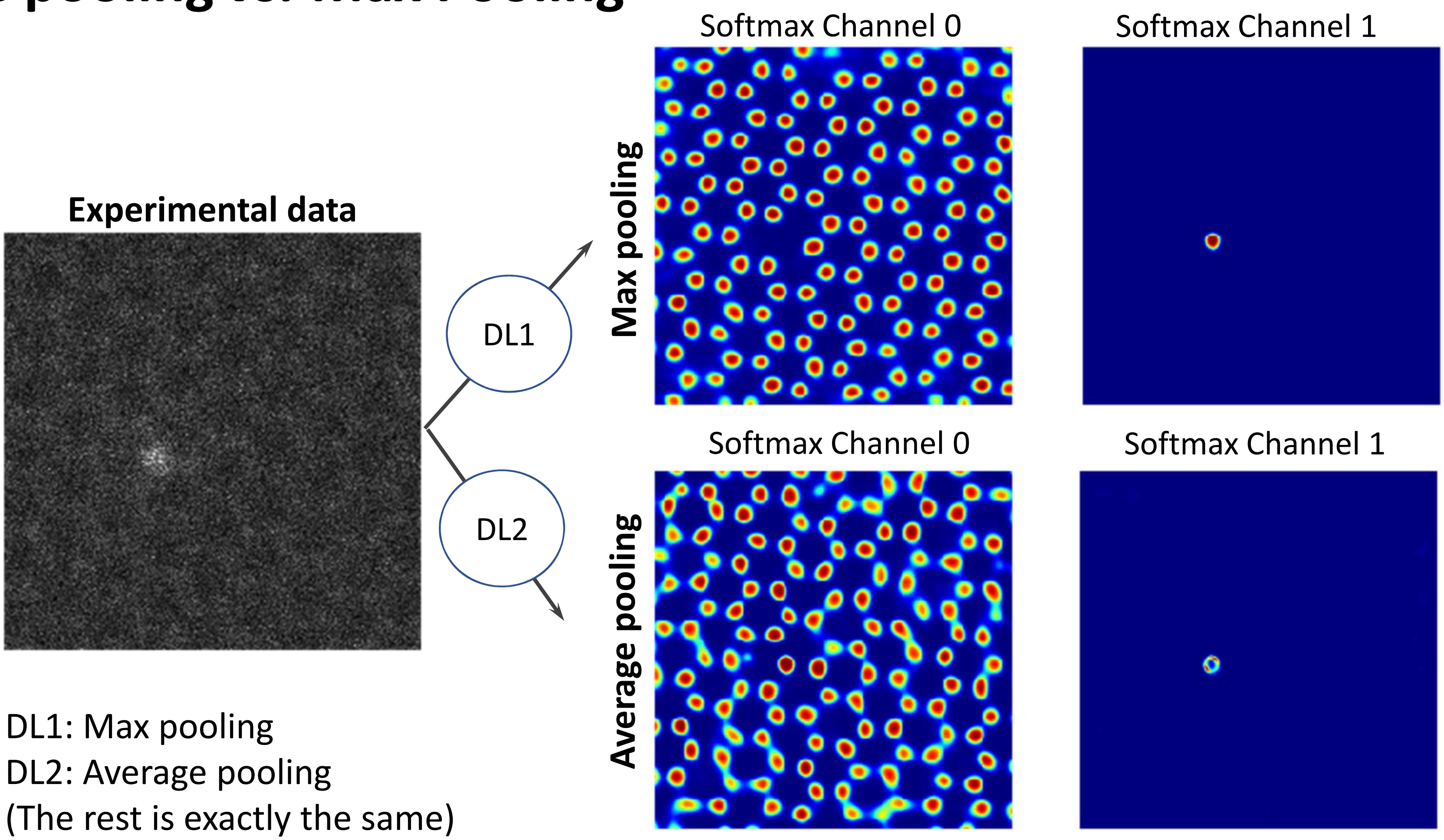
Softmax layer (channel = ‘C’)



Softmax layer (channel = ‘Si’)



Importance of model architecture:
Average pooling *vs.* Max Pooling



DL1: Max pooling
DL2: Average pooling
(The rest is exactly the same)

Analyzing displacement domains from STEM image of 3D ferroelectric material

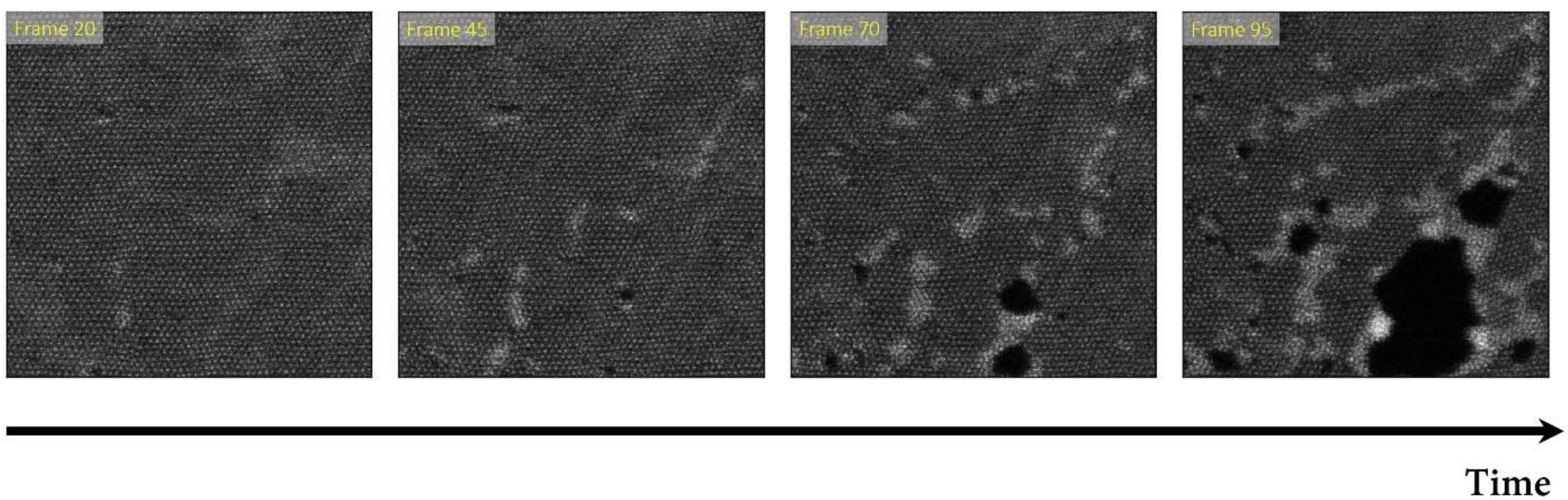
UNPUBLISHED DATA

We have successfully trained CNN models that work with patch by patch and sliding window approaches to image analysis. This allowed us to analyze high resolution images. CNN models were trained using Multislice simulations *and* simulations when atoms are modelled simply as 2D gaussians. Interestingly, both models showed very similar results.

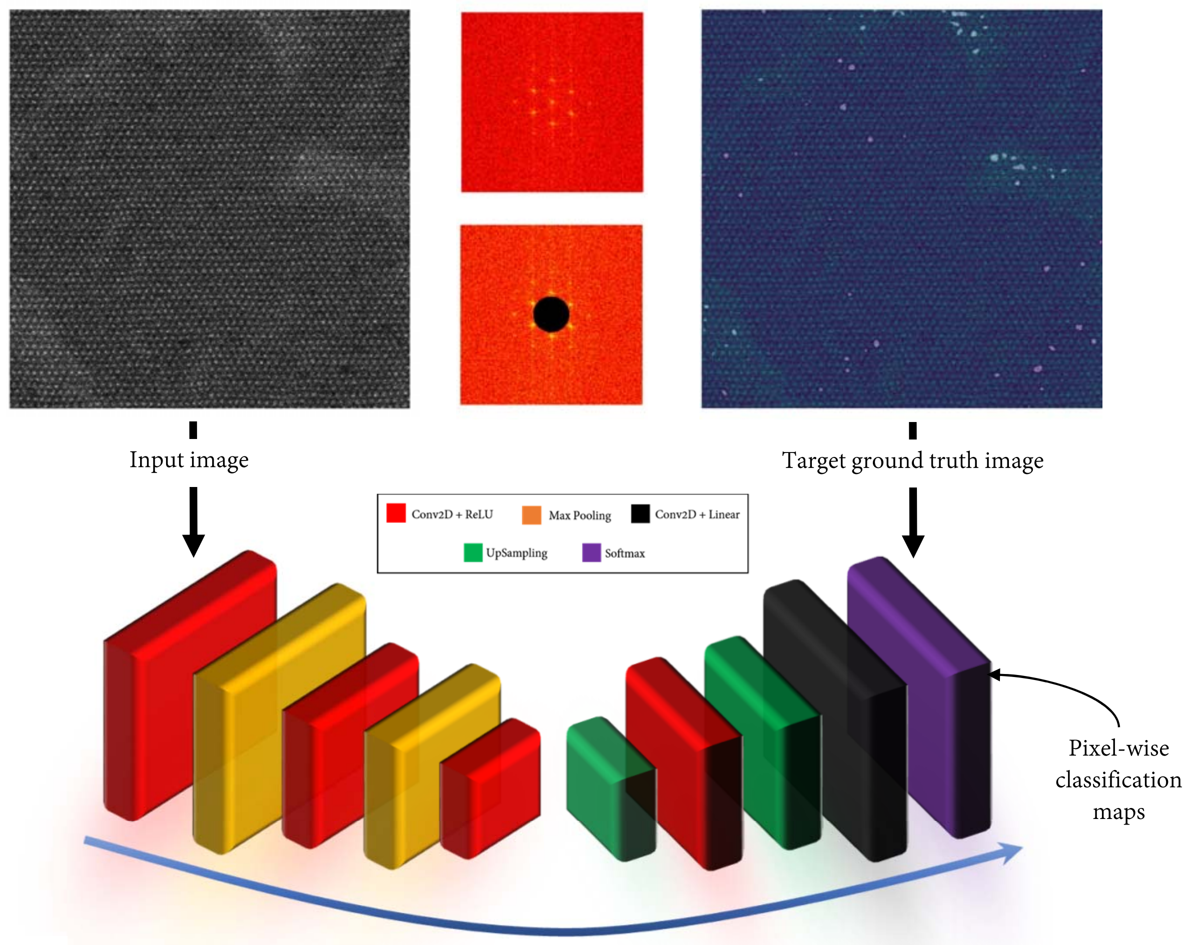
Examples (notebooks)

Tracking defects in STEM movies

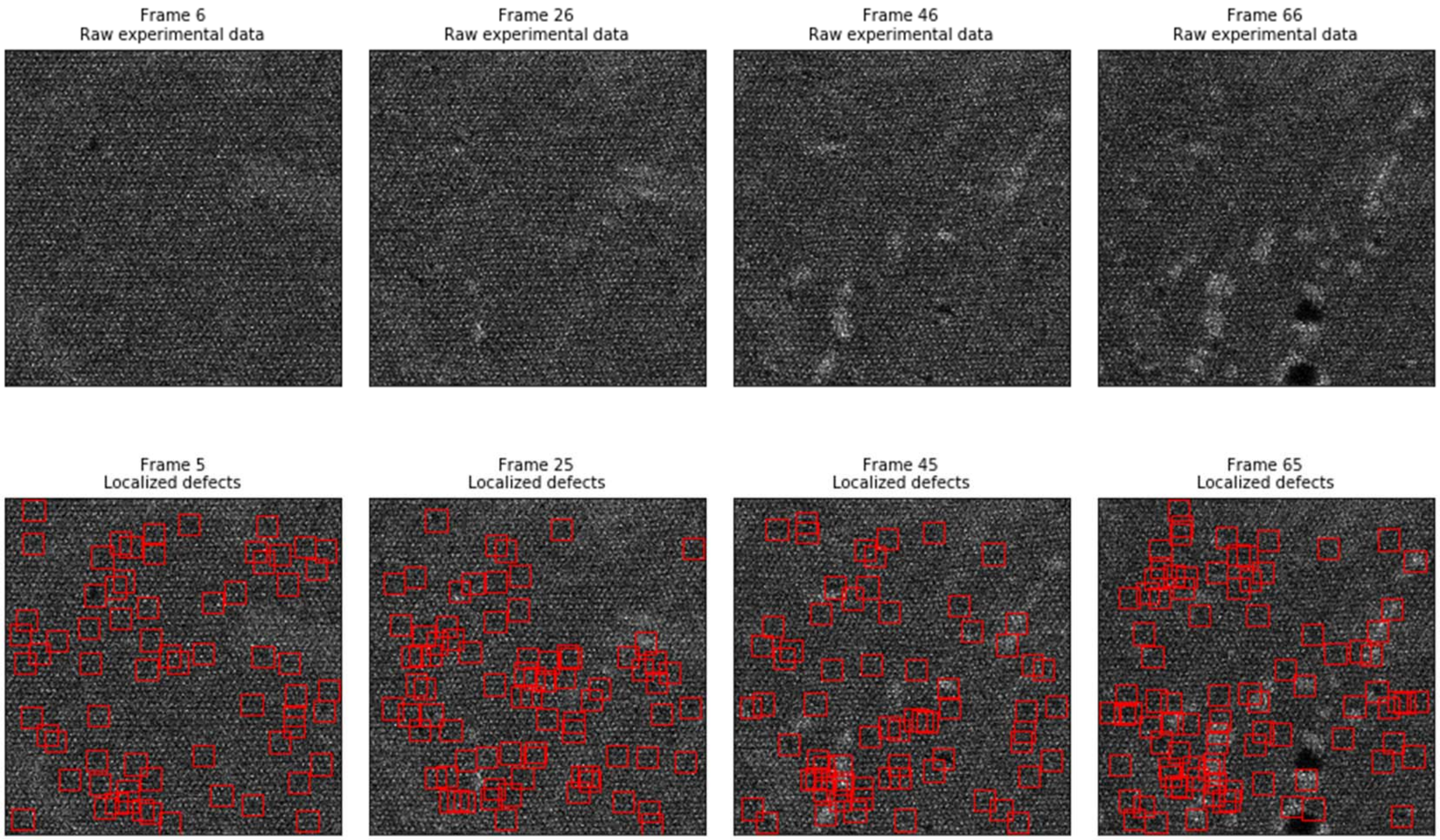
- Can we teach AI a concept of defect without priors?
- Can we track defect evolution in time?
- What can we learn?



Tracking defects in STEM movies

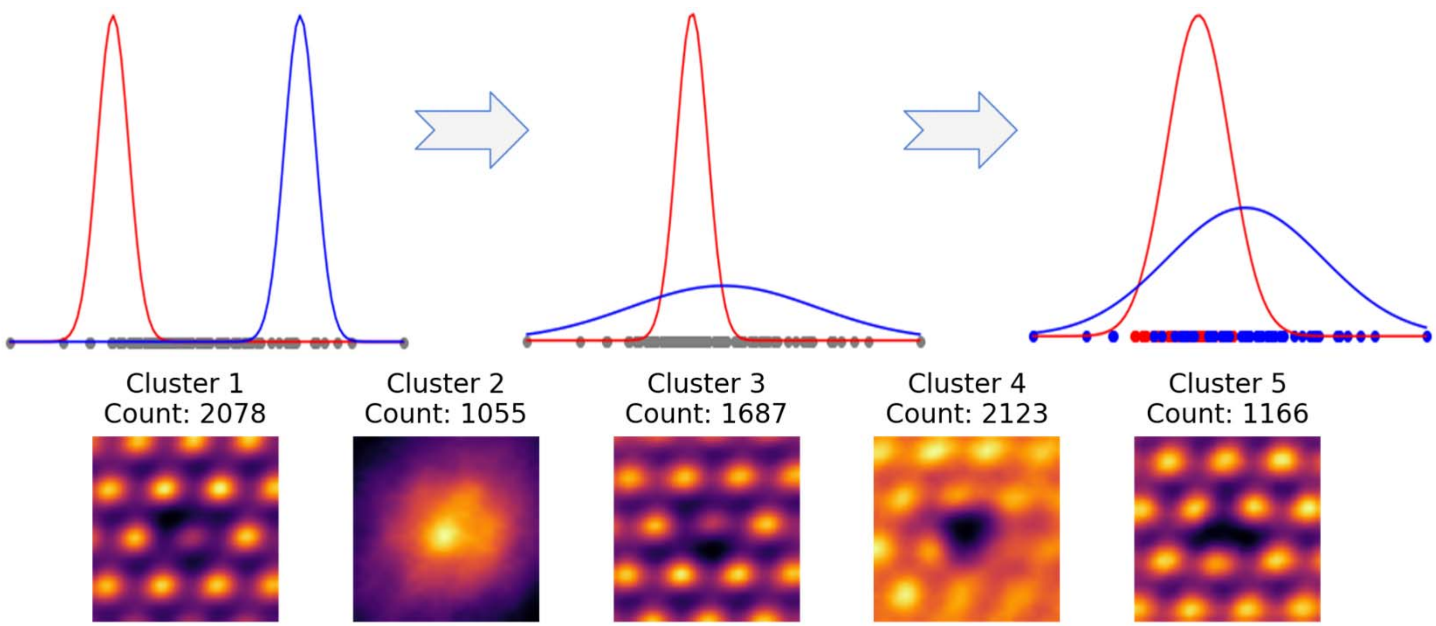


Deep FCNN for identifying defects

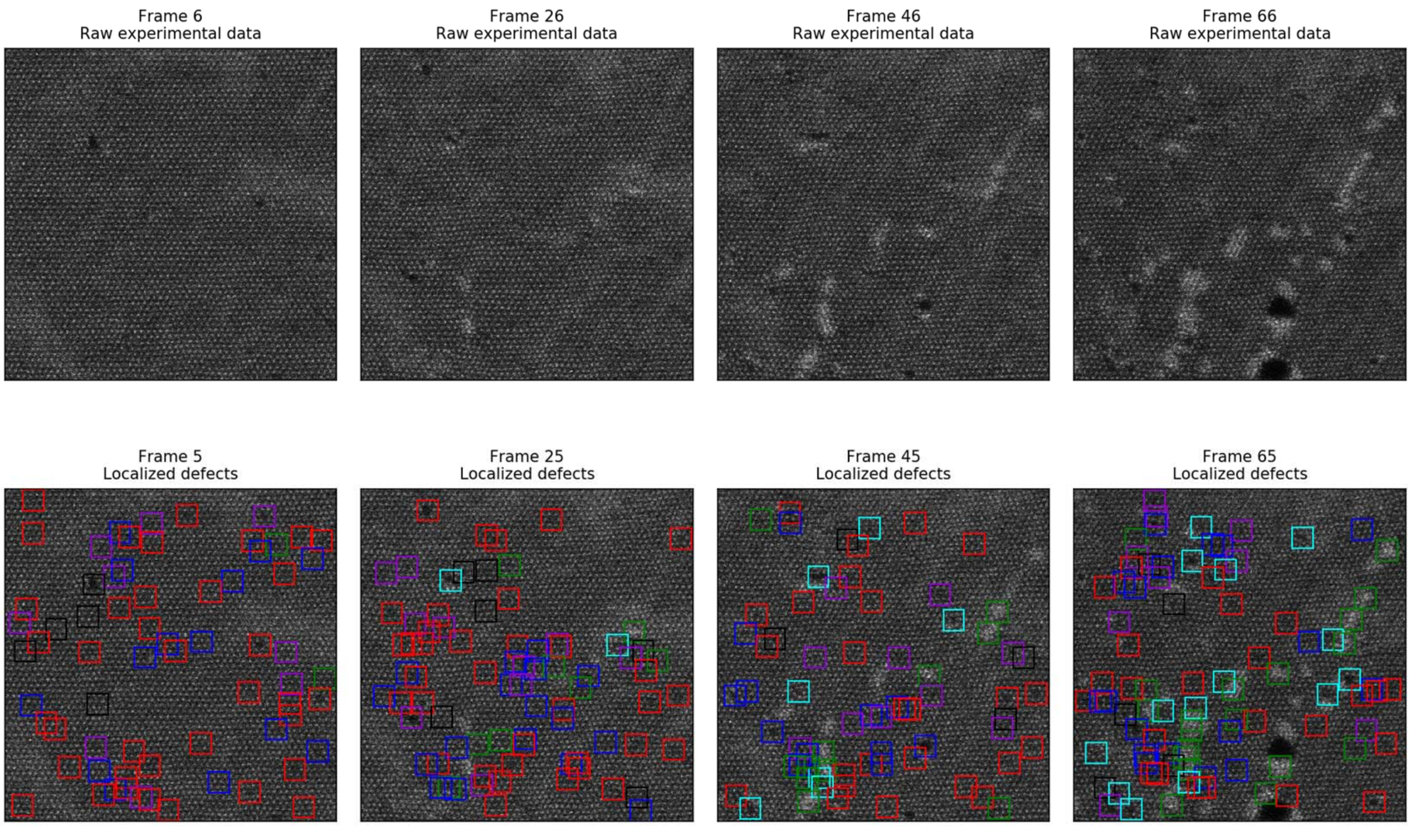


Gaussian Mixture Model for unsupervised classification of the defects

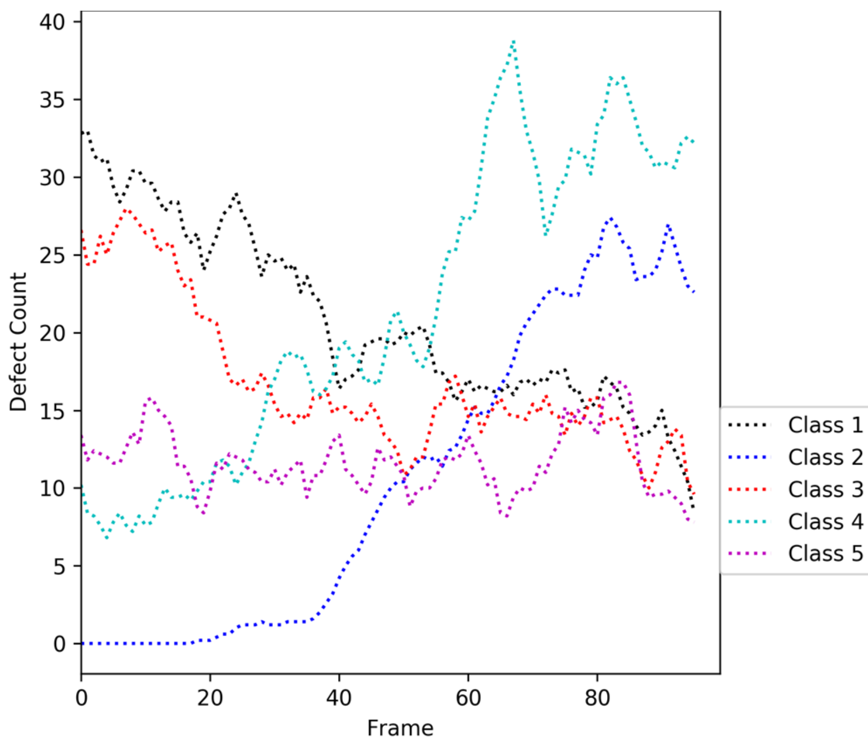
- GMM iterates from initial guess assuming Normal distribution
- GMM produces probabilities of belonging to a class
- GMM also allows to construct average class representations

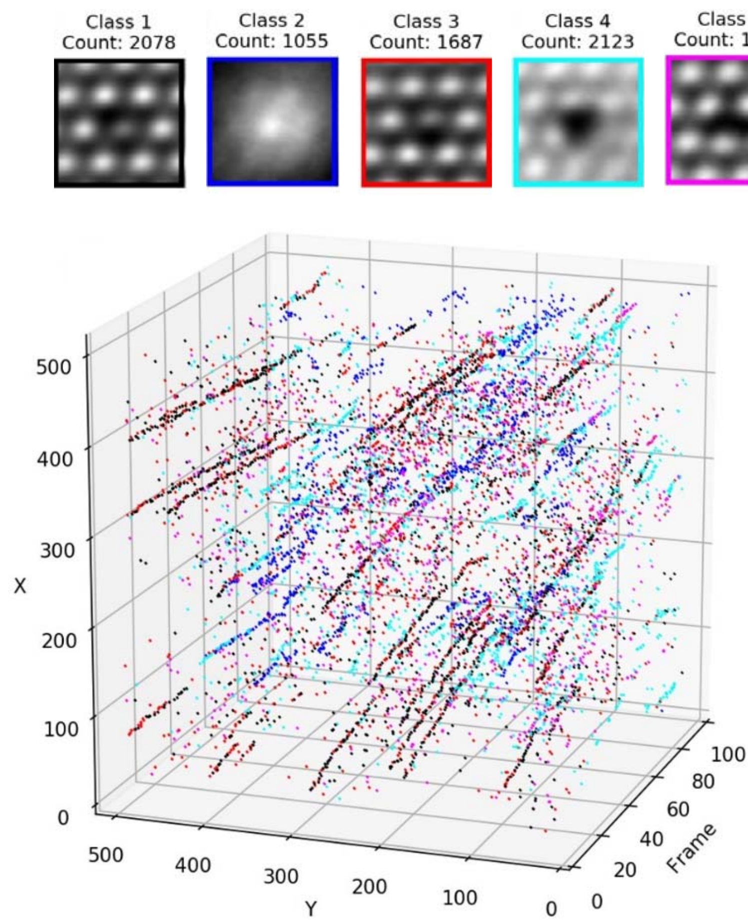


Deep FCNN + GMM for classification



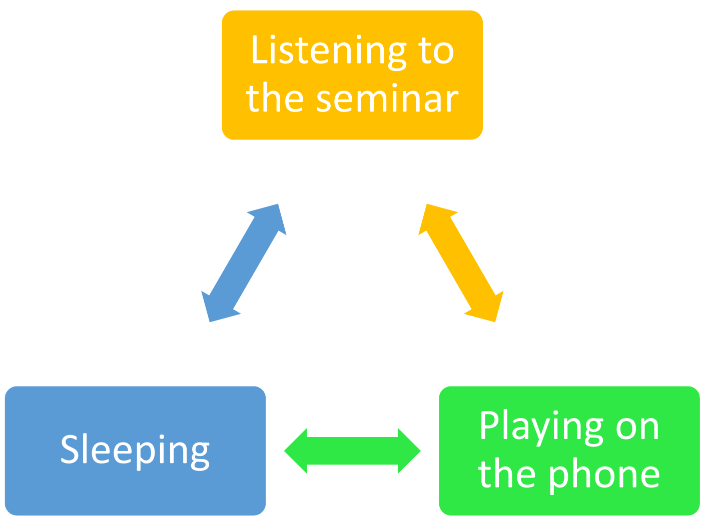
Evolution of defects in time



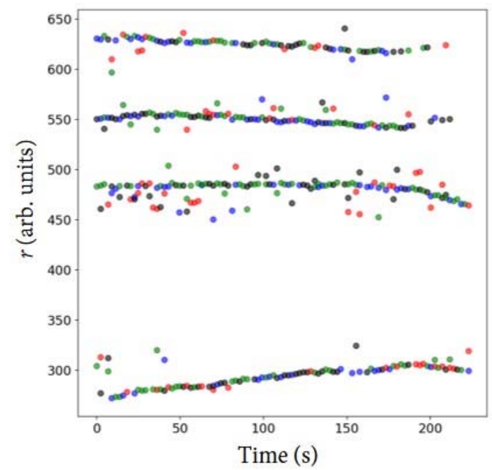


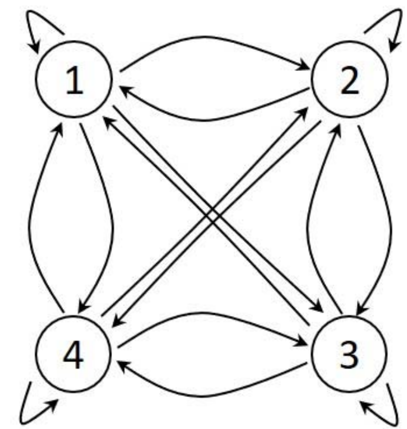
Markov Process

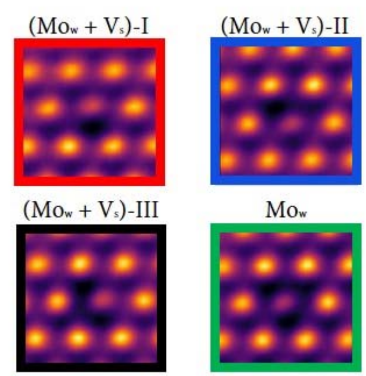
- Assuming Markov property:
  - future state depends only on the present state and not on preceding states
  - State space is completely defined
- We can derive transition probabilities from observations

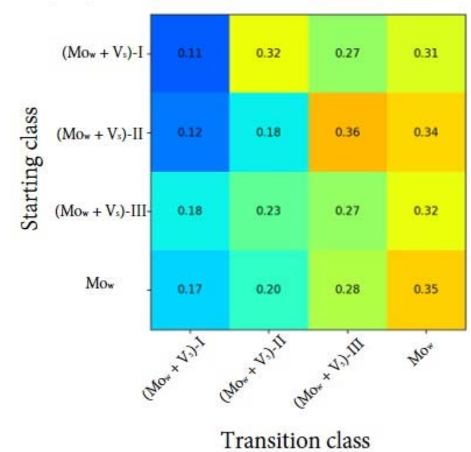


Evolution of defects as a Markov Process









Examples (notebooks)

Some useful resources

Books:
Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*; The MIT Press, 2016.
Chollet, F.: *Deep Learning with Python*; Mannings Publication Co., 2018.

Deep Learning Libraries (open source):
https://keras.io/
https://www.tensorflow.org/
https://pytorch.org/

Cloud GPUs
https://aws.amazon.com/ (tested – works nicely)

Also, note that most of the researchers in the field are quite active on social networks (Twitter, Facebook, Linkedin...)