MM 2018

Deep Learning for Atomically Resolved Imaging

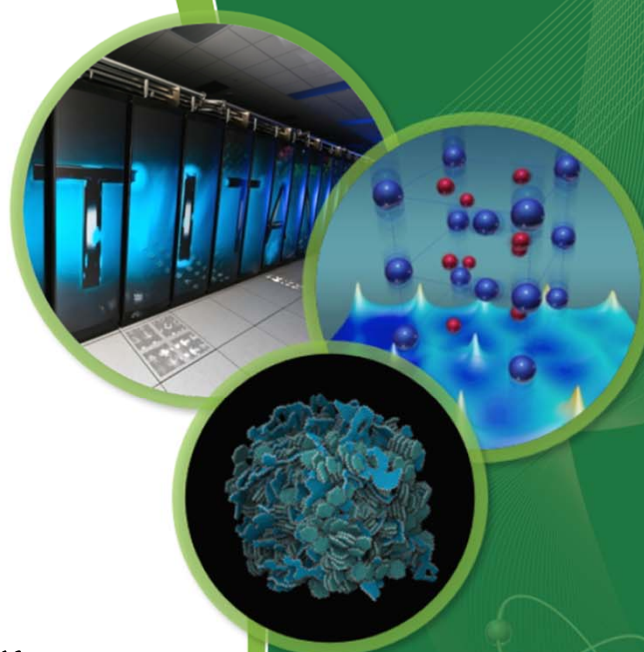
Maxim Ziatdinov

Institute for Functional Imaging of Materials

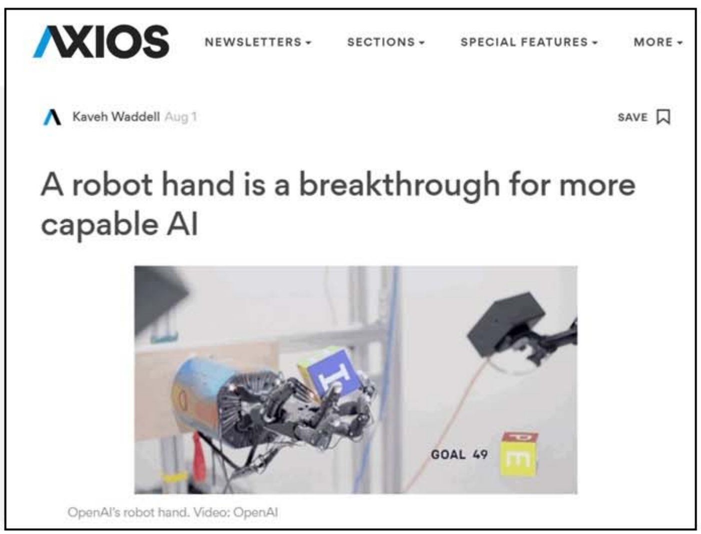
&

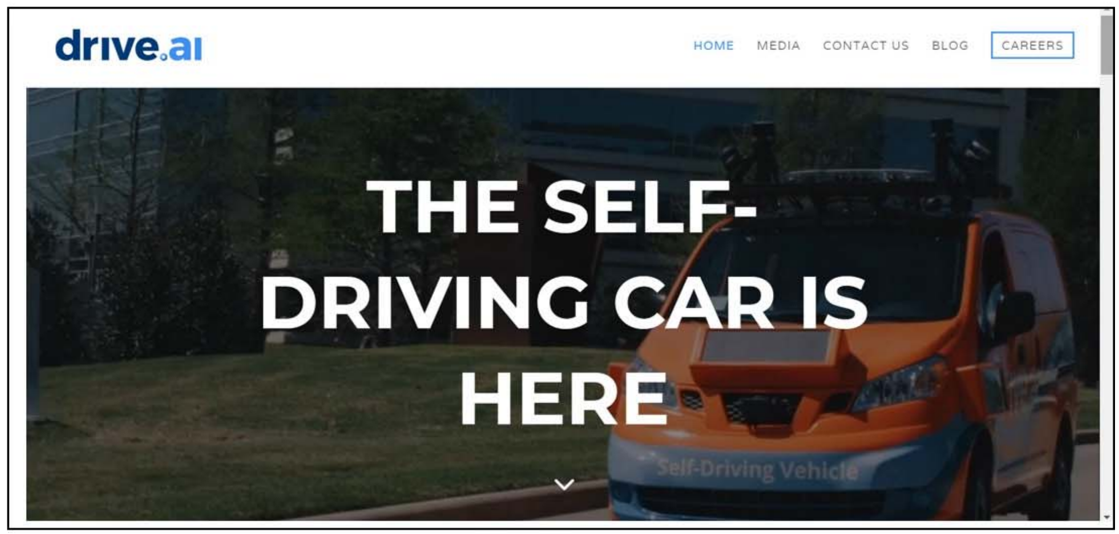
Computational Sciences and Engineering Division

August 5th 2018



Deep Learning in Modern World



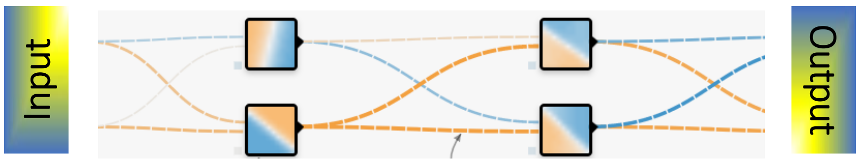


(Deep) Artificial Neural Networks

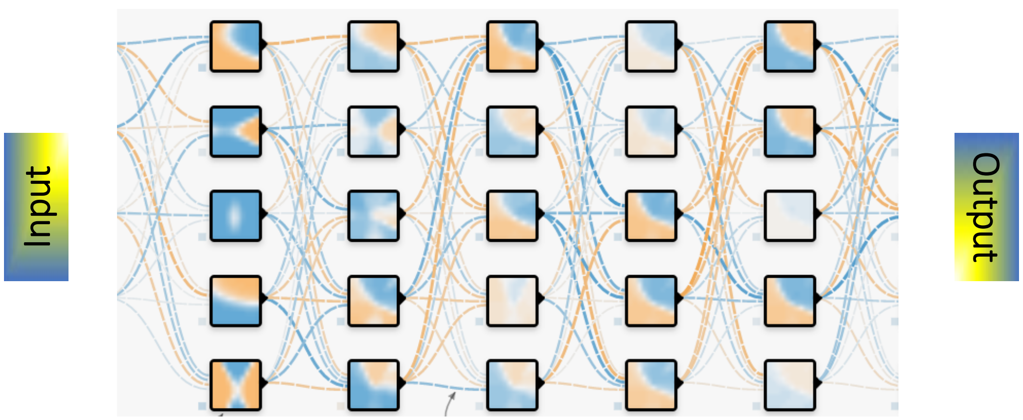
Network is trained to match input to output by adjusting weights.

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

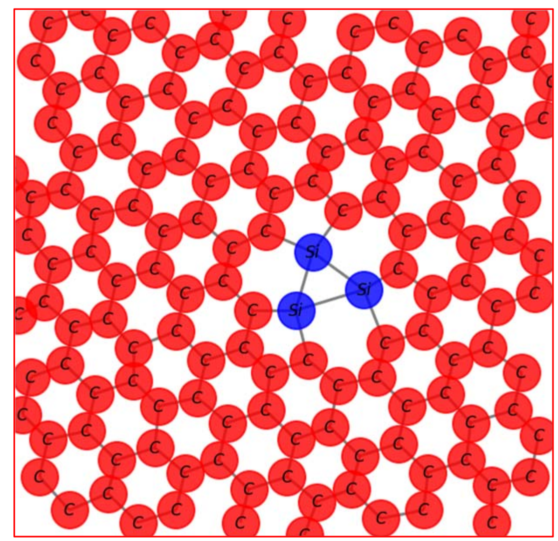


The deeper networks are able to capture more abstract, high-level features

Macro-world vs. Nano-world:

Macro-world

Nano-world



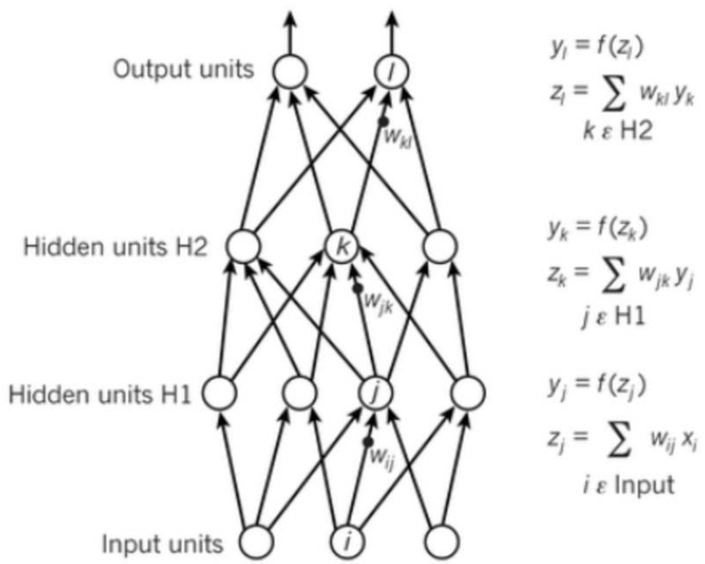
(Some) Deep Learning applications in theoretical condensed matter physics and materials sciences

Type of problem	Type of neural network	Training data/input type	Output (physics learned)
Phase transition in classical spin systems	Convolutional neural network [1,2]	Labeled 2D images corresponding to equilibrium spin configuration (e.g. from Monte Carlo simulations)	Output neuron capturing transition temperature (prediction changes from 0 to 1 at T_c)
	Variational autoencoders [3]	Same but unlabeled	Latent variable and reconstruction loss correspond to order parameter
Identification of topological and trivial phases of matter	Fully-connected neural network (with pre-input 'filter' for quantum loop tomography) [4]	Labeled quasi-2D "images" made of un-averaged values of products of loop-forming operators that are relevant for the phase of interest	Output neuron is associated with "topological response" (prediction changes from 0 to 1 at k=0.5)
Differentiating between dynamical phases of a time-dependent model (Many body localization)	Recurrent neural network [5]	Magnetization time-traces	Output neuron recovers phase boundaries
Predicting materials properties (e.g. formation energy, band gap) directly from crystal lattice (graph) structure.	Graph convolutional network [6]	Atom feature vector (group number, covalent radius, valence electrons, etc.) and bond feature vector (atom distance)	Output neuron recovers continuous or discrete properties of material
Distinguishing different topological invariants	Convolutional neural network [7]	Labeled parameterized Hamiltonians	Output neuron corresponds to a winding number

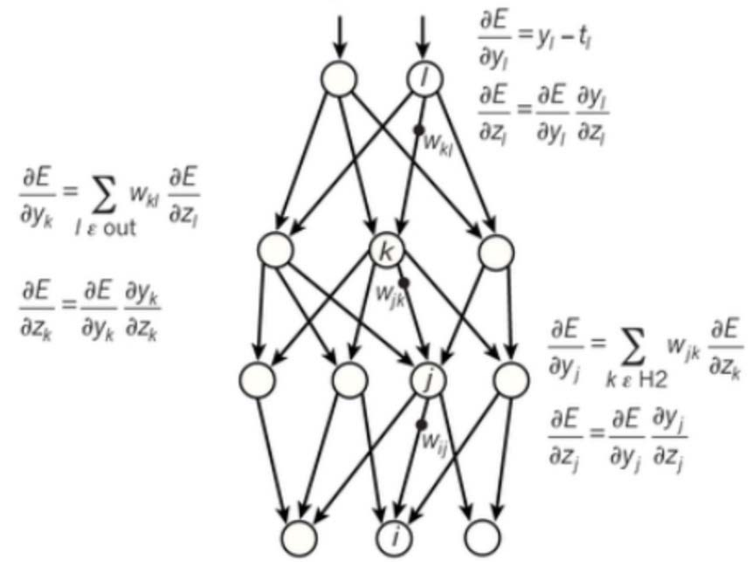
1. Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. *Nature Physics* **13**, 431 (2017).
2. Tanaka, A. & Tomiya, A. Detection of Phase Transition via Convolutional Neural Networks. *Journal of the Physical Society of Japan* **86**, 063001 (2017).
3. Wetzel, S. J. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Physical Review E* **96**, 022140 (2017).
4. Zhang, Y. & Kim, E.-A. Quantum Loop Topography for Machine Learning. *Physical Review Letters* **118**, 216401 (2017)
5. van Nieuwenburg, E., Bairey, E. & Refael, G. Learning phase transitions from dynamics. *arXiv preprint arXiv:1712.00450* (2017).
6. Xie, T. & Grossman, J. C. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Physical Review Letters* **120**, 145301, (2018).
7. Zhang, P., Shen, H. & Zhai, H. Machine Learning Topological Invariants with Neural Networks. *Physical Review Letters* **120**, 066401 (2018).

Deep feedforward networks

Forward propagation



Back propagation

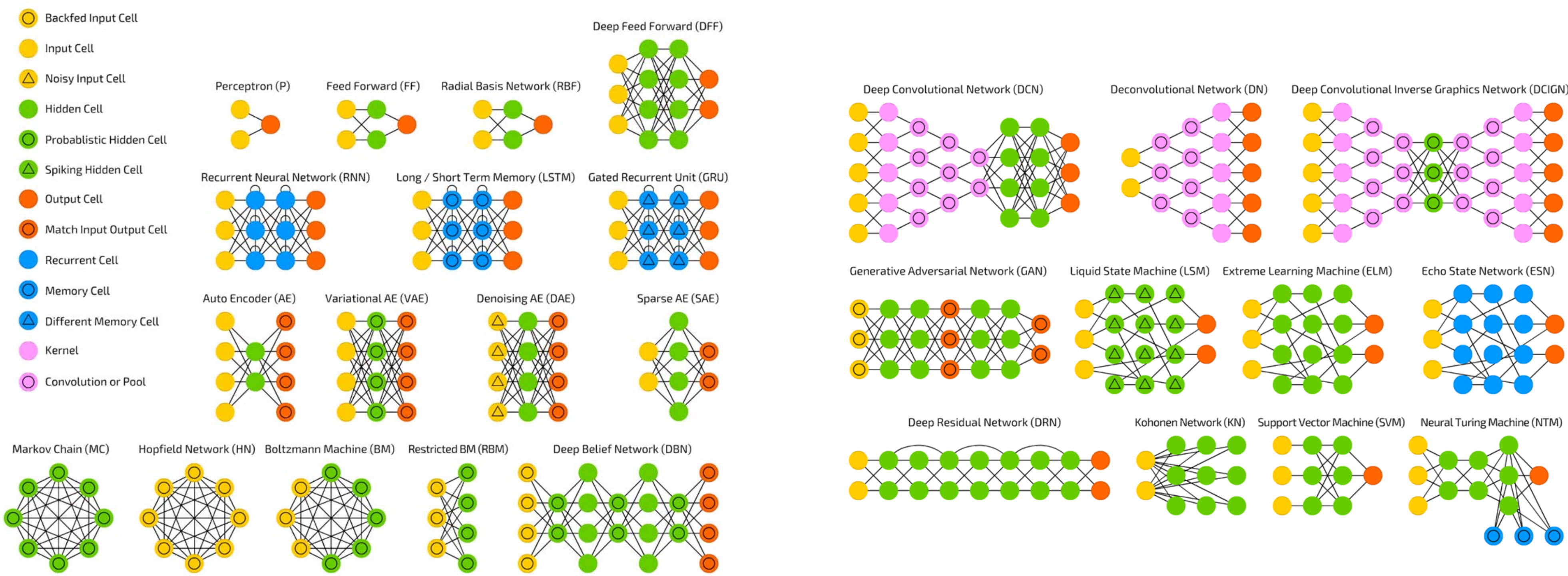


The goal of a feedforward network is to approximate some function *F**. For a classifier, *y* = *F**(*x*) maps an input *x* to a category *y*. A feedforward network defines a mapping $y = F(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the value of the parameters θ that result in the best function approximation**.** The inputs *x* provide the initial information that then propagates up to the hidden units at each layer and finally produces $\hat{y}$. During training, forward propagation can continue onward until it produces a scalar cost $J(\theta)$. The back-propagation algorithm allows the information from the cost to then flow backwards through the network, in order to compute the gradient.

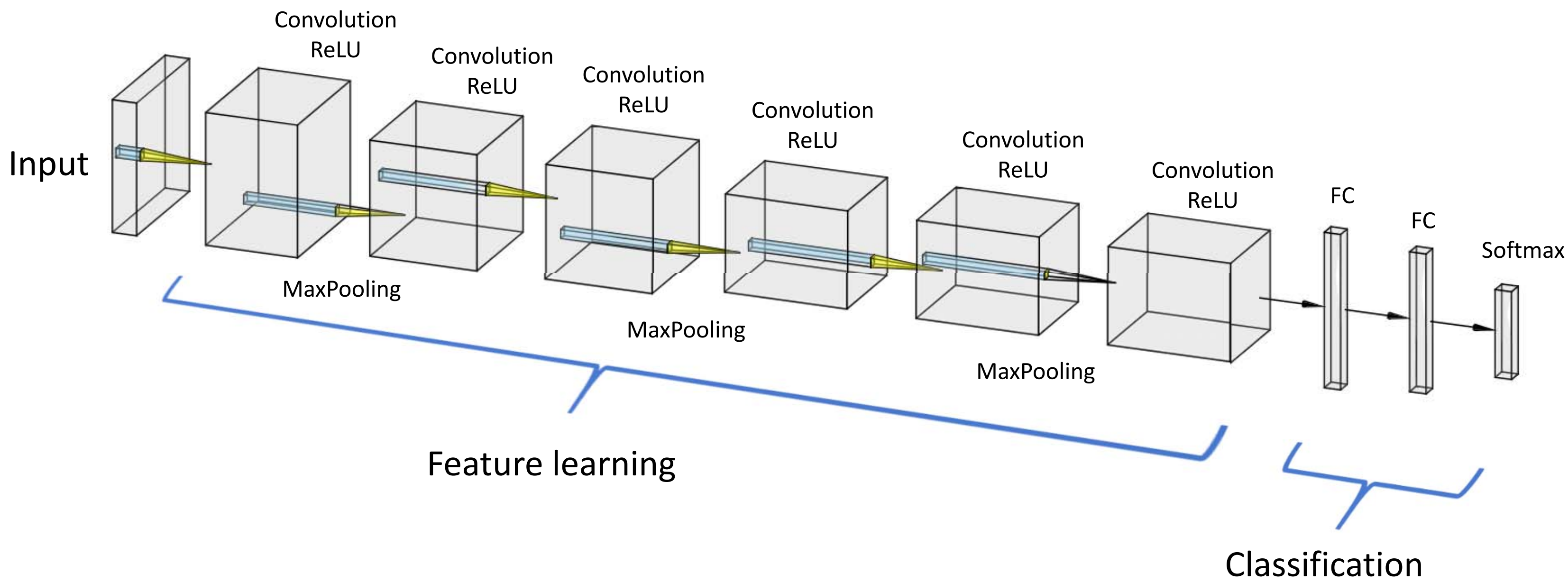
Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*; The MIT Press, 2016

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

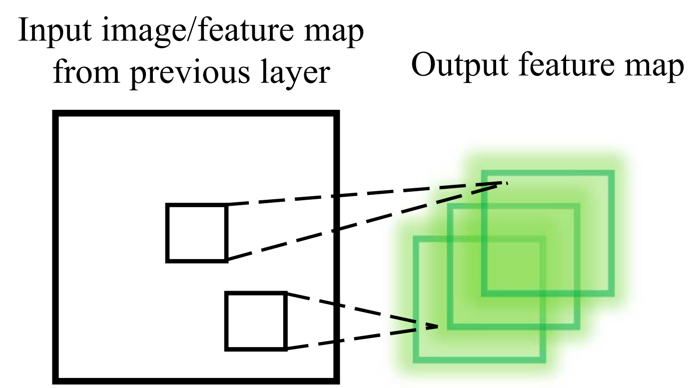
Pooling: merge similar features into one

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

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

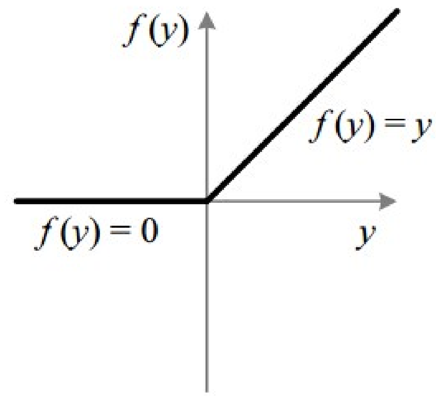


Activation Layer

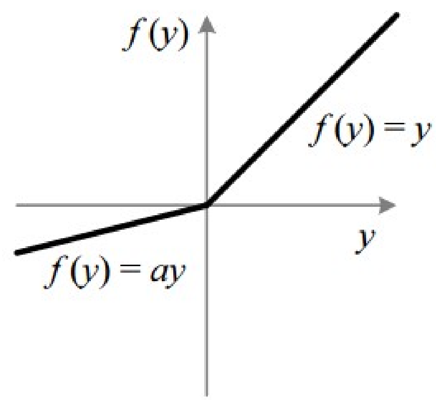
- Increases non-linearity of the network without affecting receptive fields of convolutional layers

- Type of non-linear activations: ReLU, Leaky ReLU, Tanh, ELU, softmax, etc…

ReLU

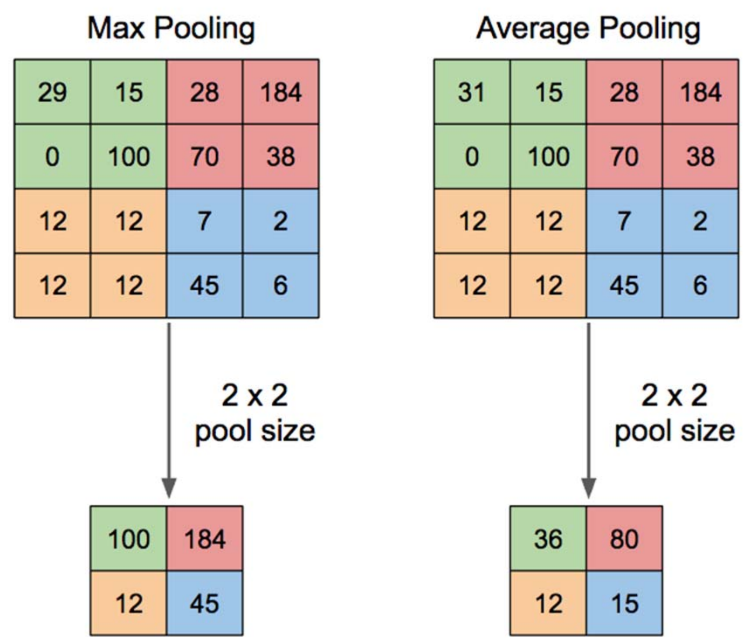


Leaky ReLU



Pooling Layer

- Accepts a volume of size $\boldsymbol{w}_1 \times \boldsymbol{h}_1 \times \boldsymbol{d}_1$
  - 2 hyperparameters:
    - $\boldsymbol{f}$: size (e.g. 2×2)
    - $\boldsymbol{s}$: stride
- Outputs a volume of size $\boldsymbol{w}_2 \times \boldsymbol{h}_2 \times \boldsymbol{d}_2$:
  - $\boldsymbol{w}_2 = (\boldsymbol{w}_1 - \boldsymbol{f})/\boldsymbol{s} + 1$
  - $\boldsymbol{h}_2 = (\boldsymbol{h}_1 - \boldsymbol{f})/\boldsymbol{s} + 1$
  - $\boldsymbol{d}_2 = \boldsymbol{d}_1$



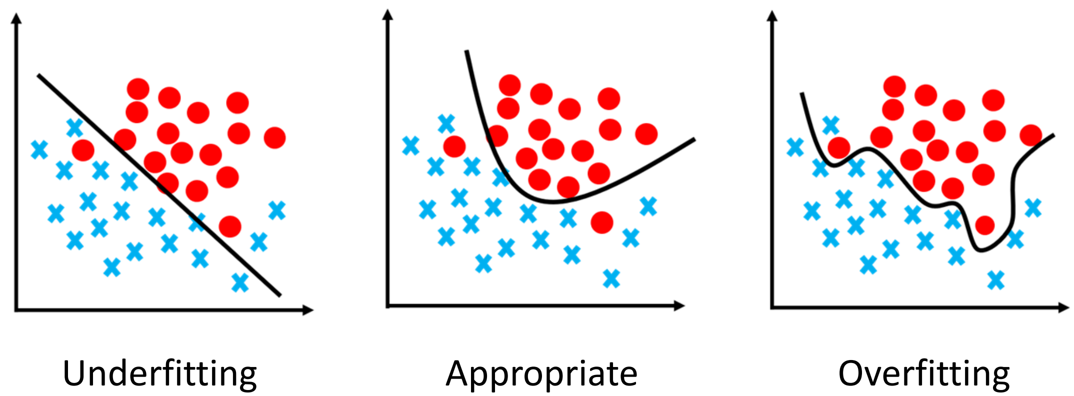
Activation Layer: Softmax

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

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

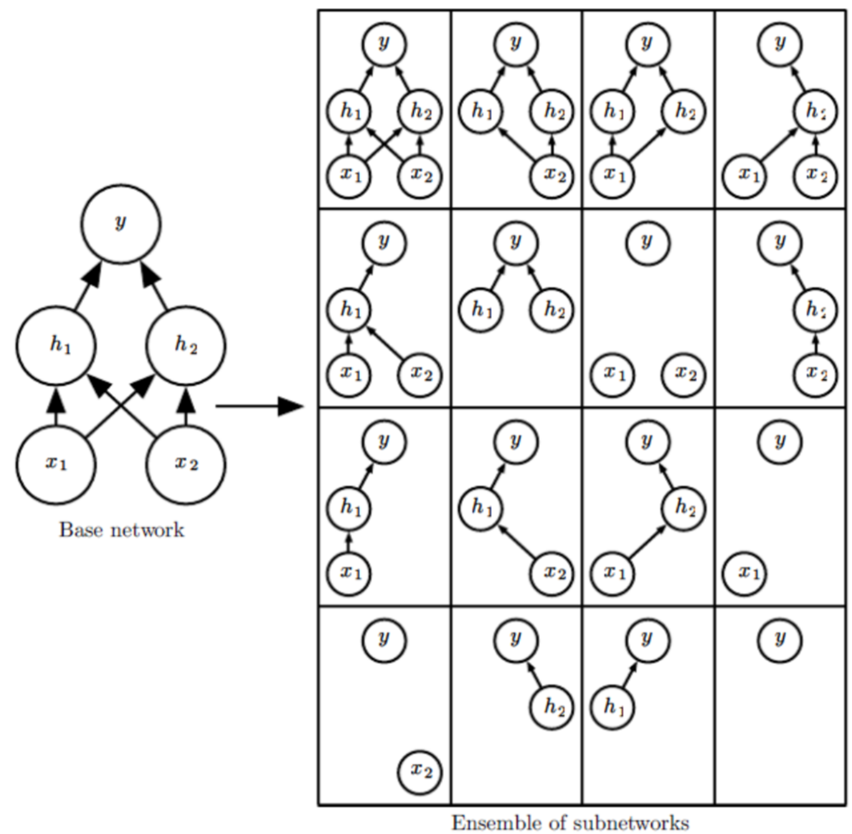
Regularization

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



Dropout layer

- 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

Importance of Training Set

VGG-19 pre-trained on ImageNet

Top 3 predictions

Very close match

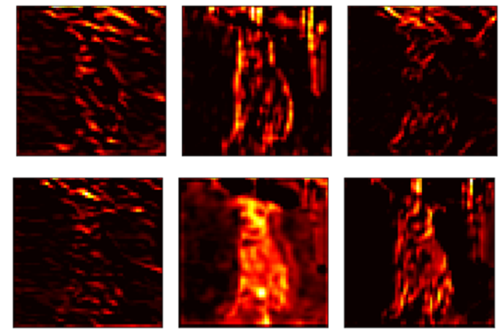
Staffordshire bullterrier 43 %

American pit bull terrier 23 %

Basenji 11 %

Meet Duffy

Some feature maps
from intermediate layers



Top 3 predictions

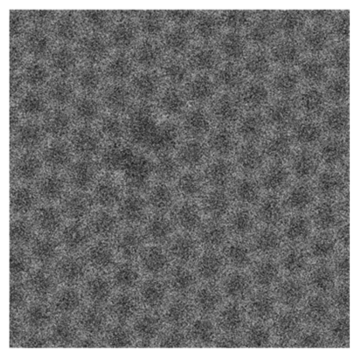
Doormat 22 %

Dishrag 5 %

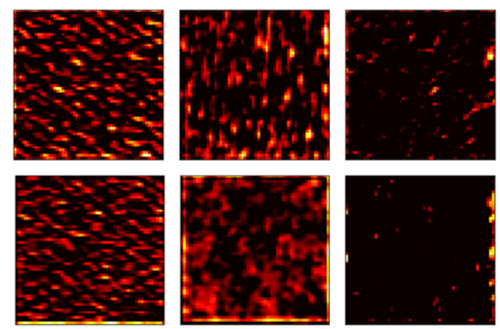
Poncho 4 %

These are not the features we are interested in!

STEM of graphene

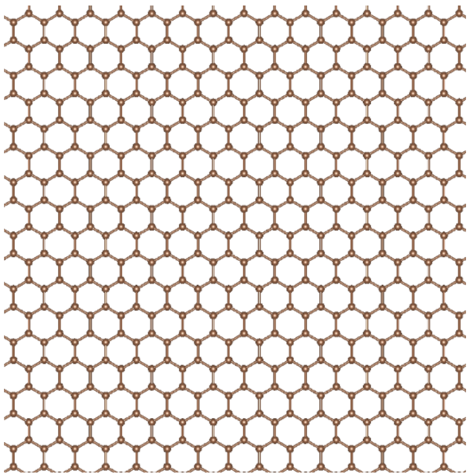


Some feature maps
from intermediate layers



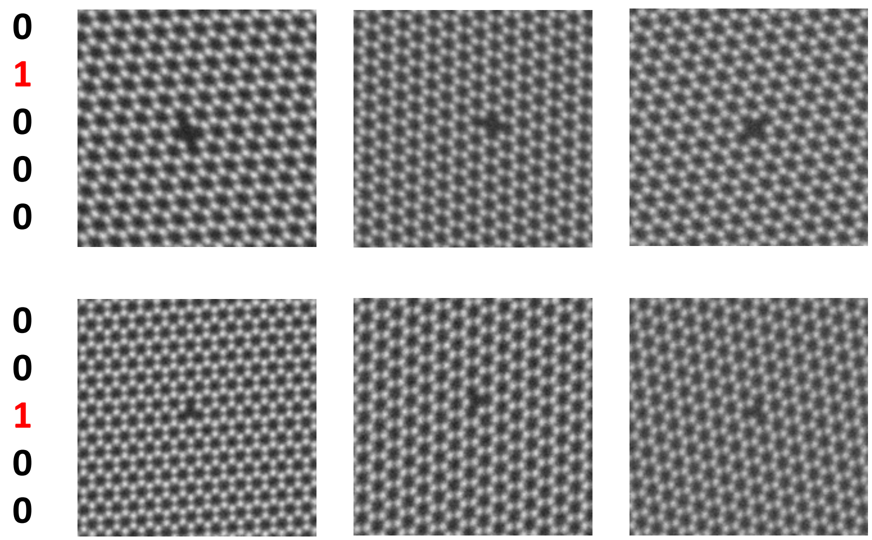
Training Set: Teaching atomic imaging concepts to neural networks

Atomic coordinates
("ground truth")



QM electron scattering
→
QM electron tunneling

Simulated (and augmented) data

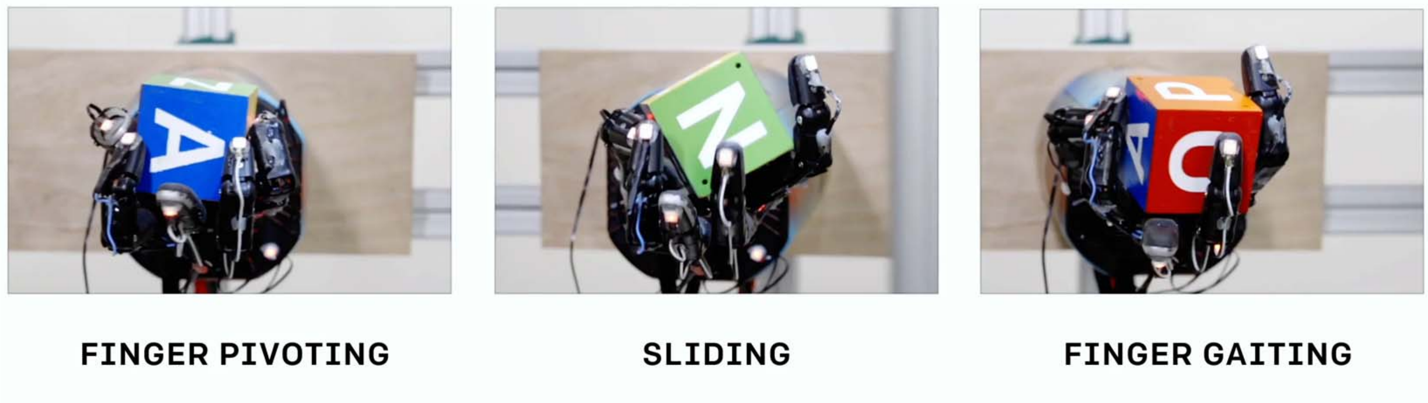


Data augmentation:

- 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**

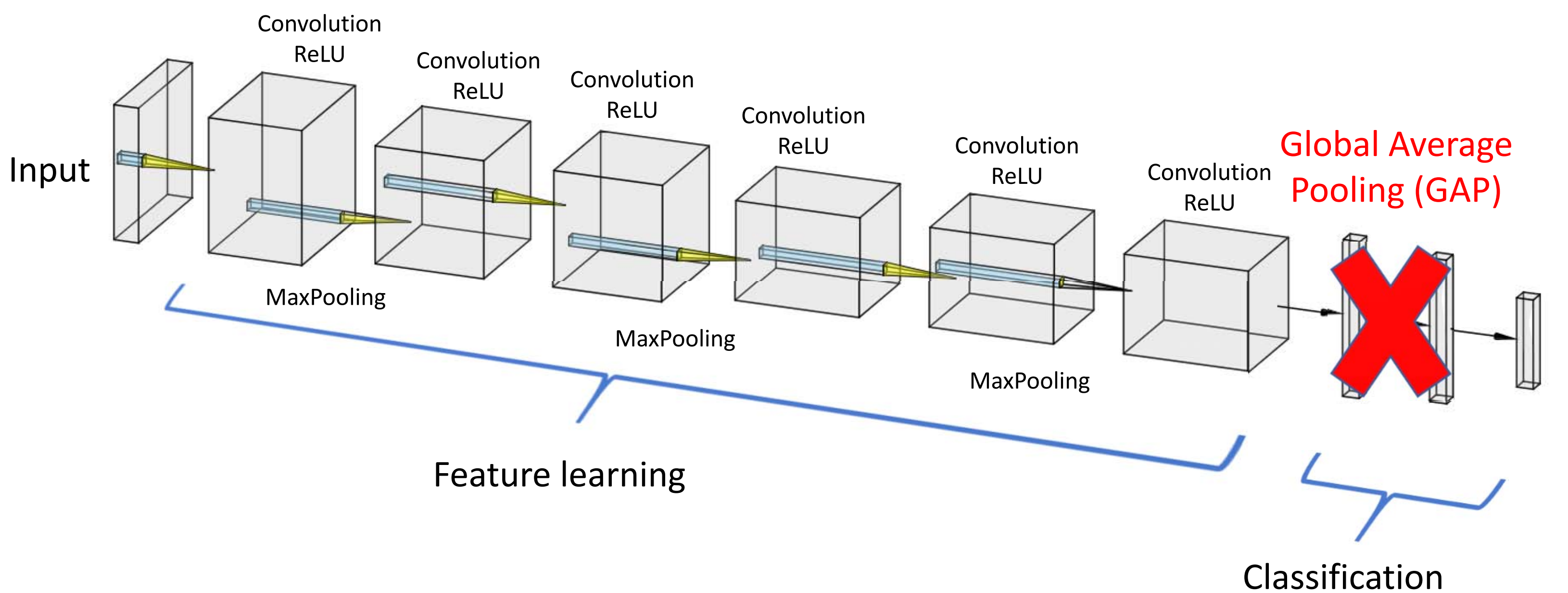
Simulations-to-physical-world knowledge transfer



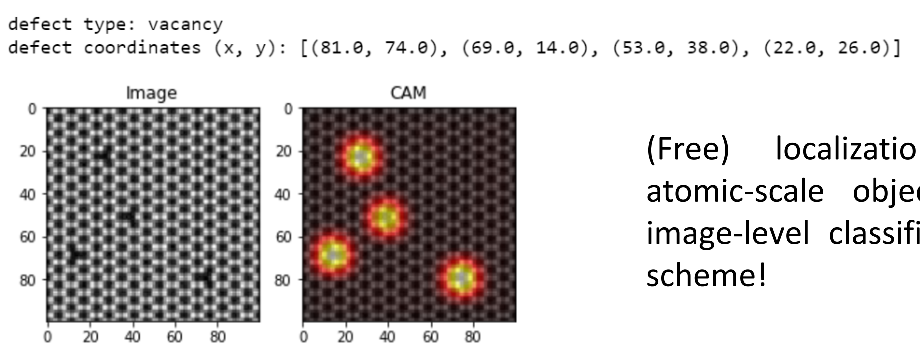
https://blog.openai.com/learning-dexterity/

"*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



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



(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

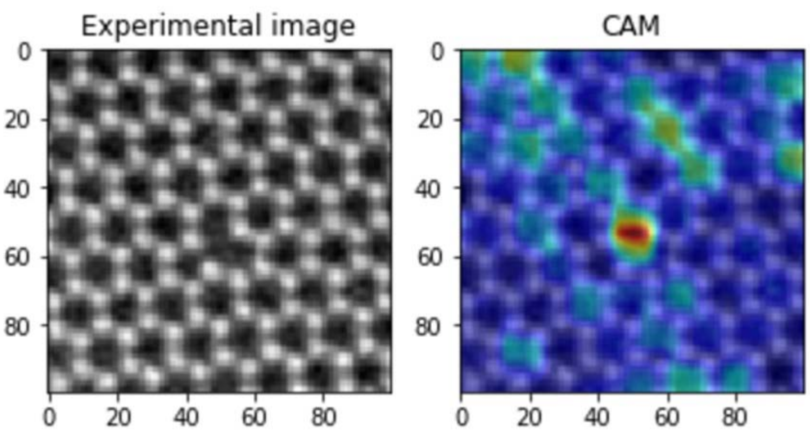
Final activation layer

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.

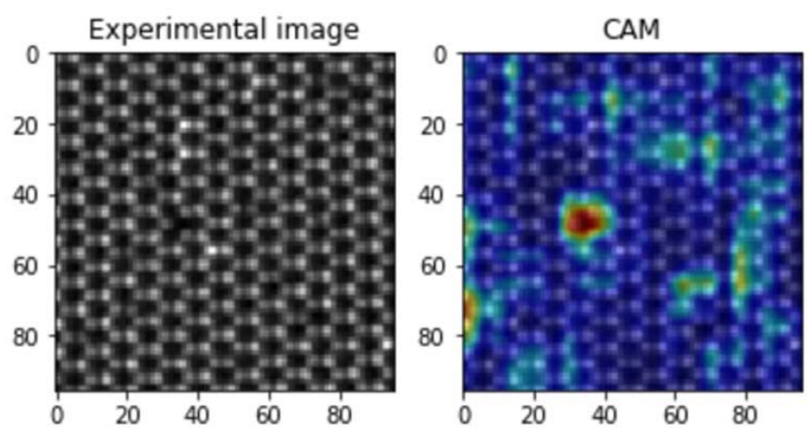
Graphene

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

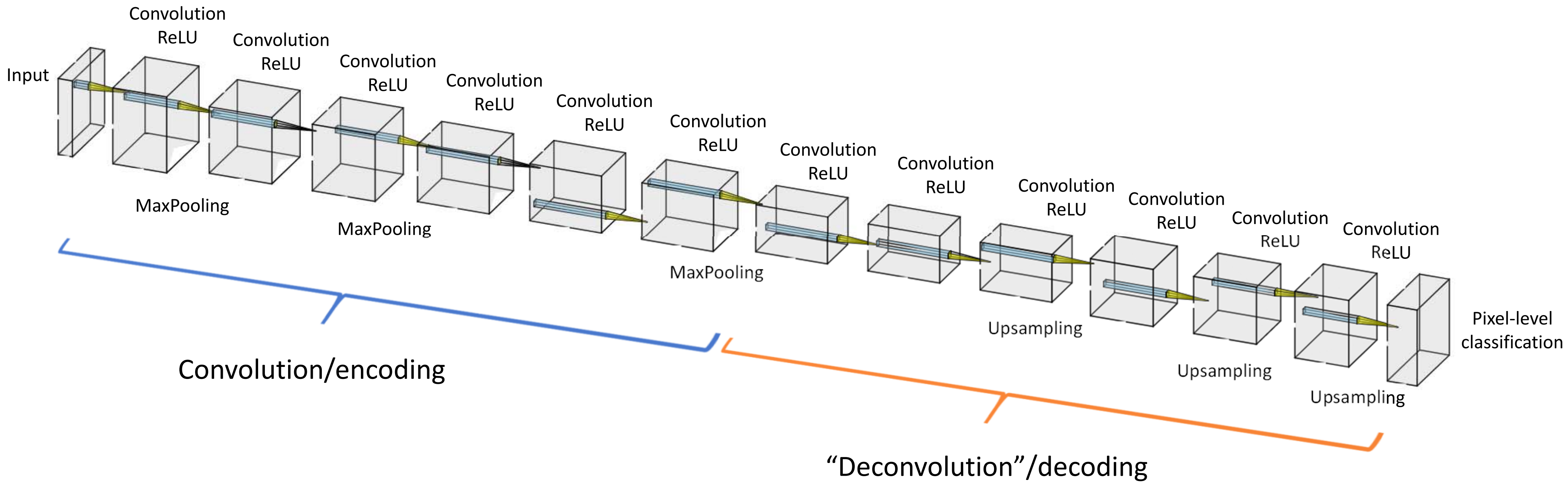
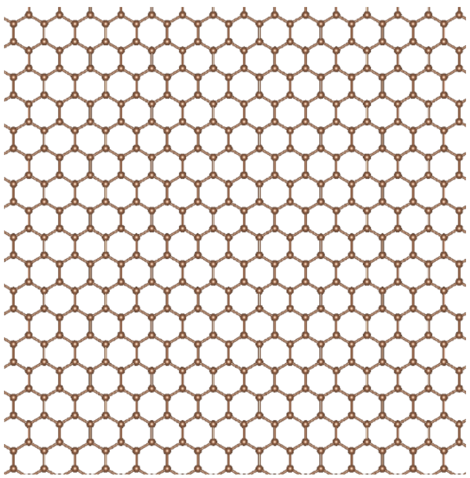


Image of CNN is generated using network drawing tools at http://alexlenail.me

Training Set: Teaching atomic imaging concepts to neural networks

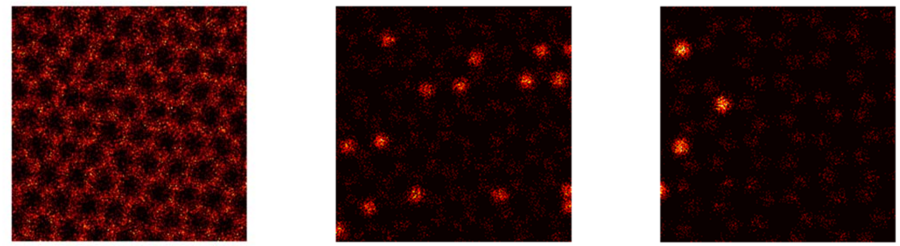
Atomic coordinates
("ground truth")



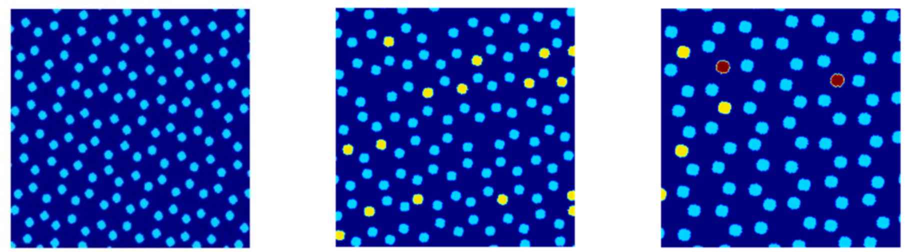
QM electron scattering

QM electron tunneling

Simulated data



Ground Truth



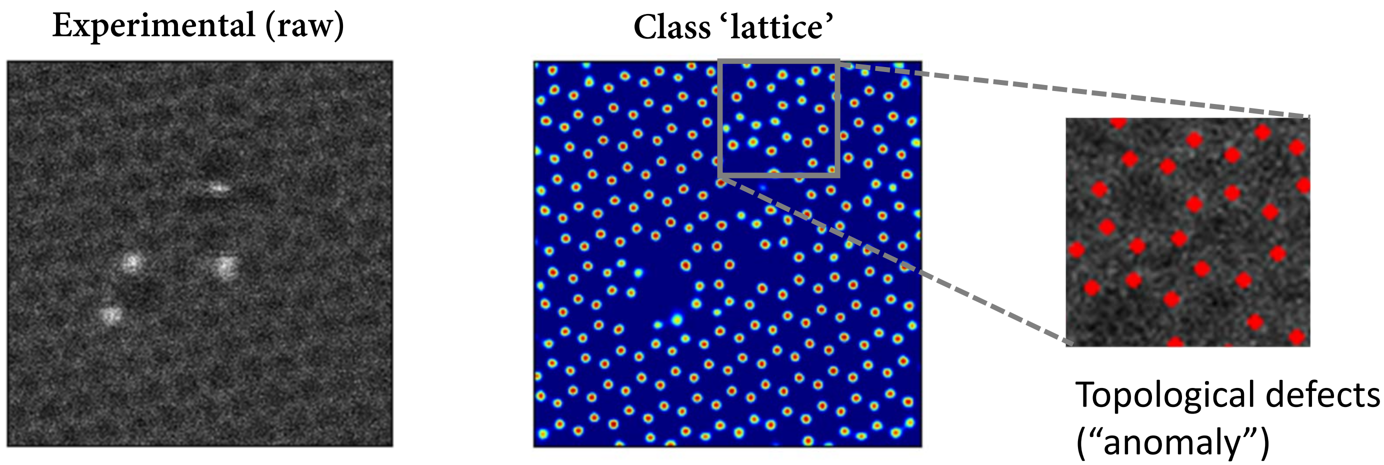
One may count vacancy as a separate class

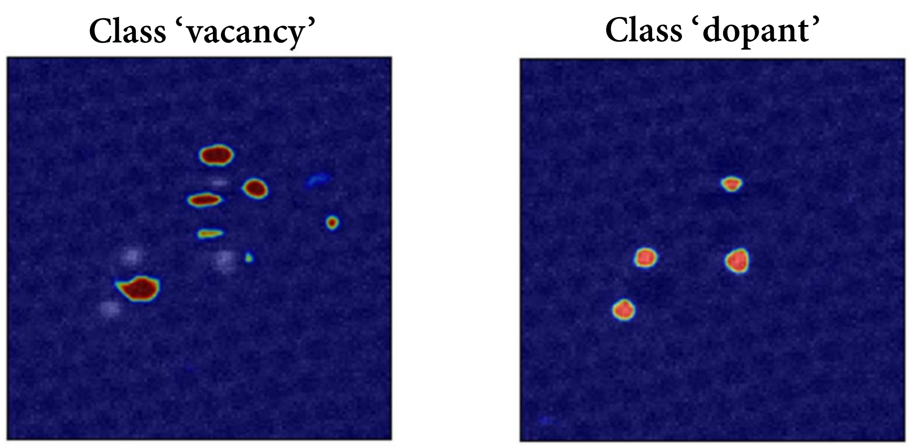
Data augmentation:

- 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





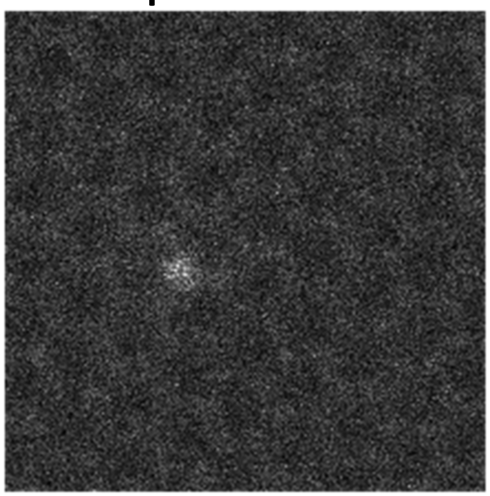
Ziatdinov *et al.*, *ACS Nano 11*, 12742 (2017)

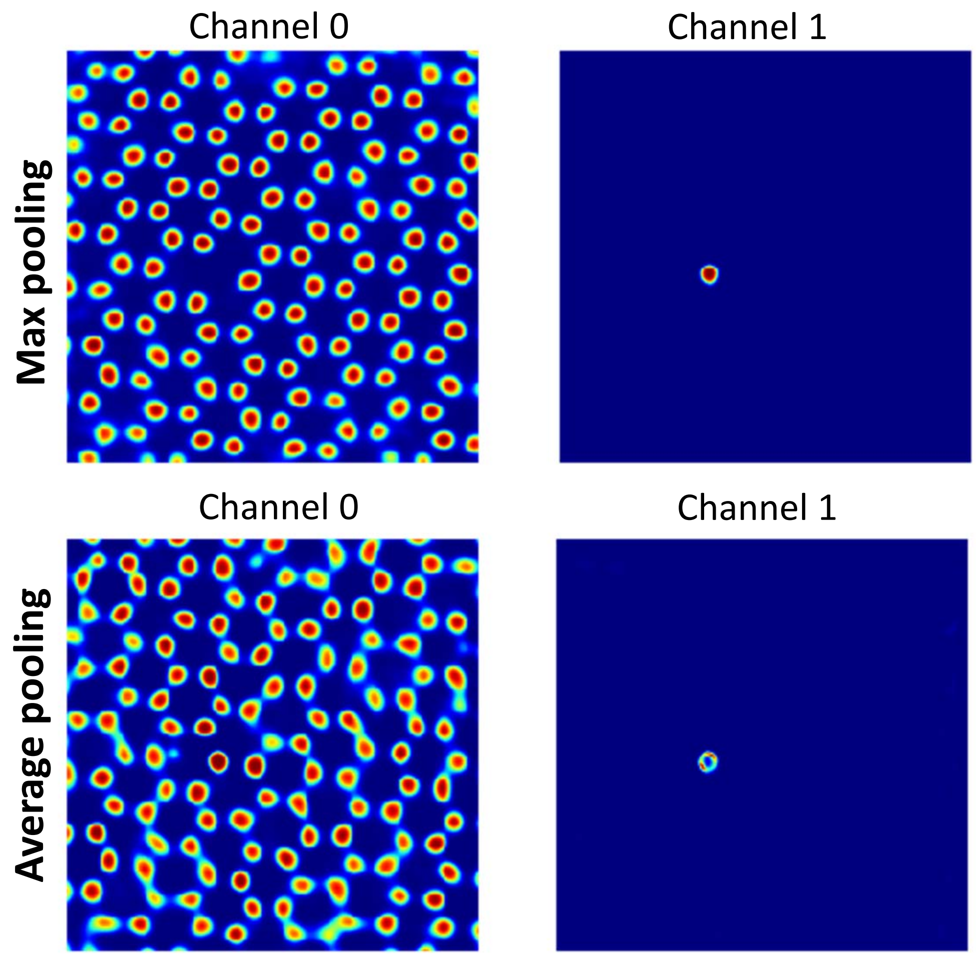
How does DL model find all the atoms?

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.

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

Experimental data

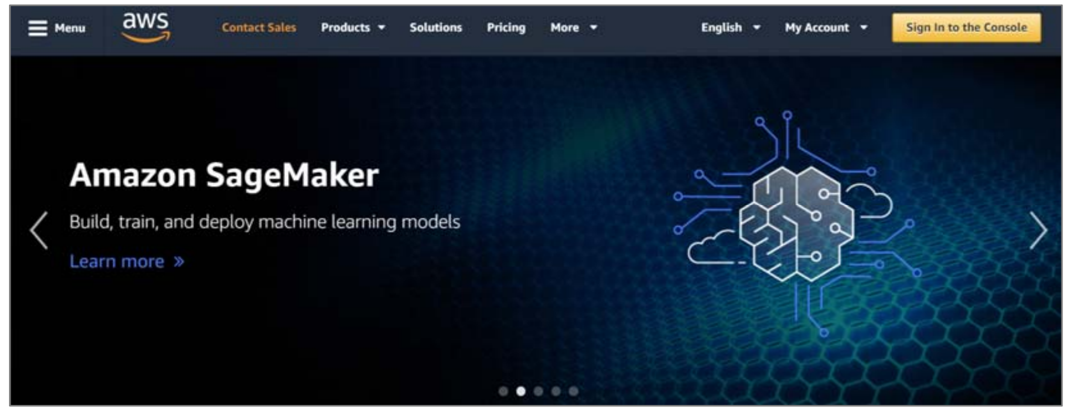




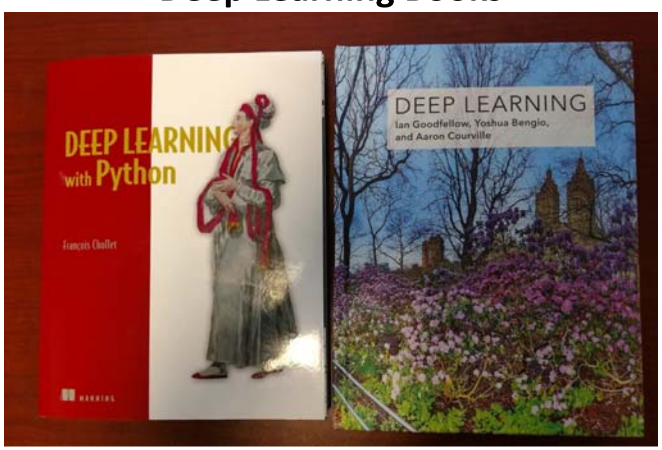
Some useful resources (personal advice)

Deep Learning libraries

Computational Recources



Deep Learning Books



Most of the researchers in the field are very active on Twitter, Facebook, Linkedin…

Please also check the Notebooks for some simple implementations of DL models for finding atoms