Computational Modeling for Materials Engineering

Deep Learning Application in Materials Engineering

INTRODUCTION and HANDS-ON SESSION 1

José Pablo Quesada Molina

josepablo.quesada@polimi.it

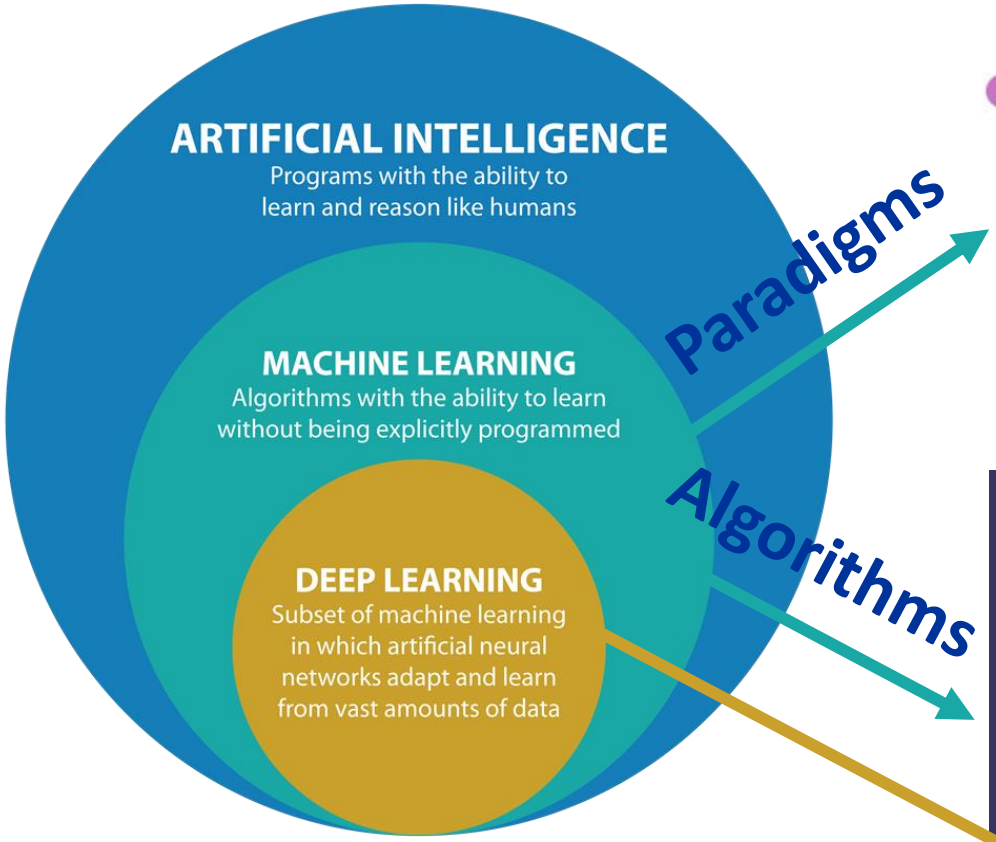
Agenda for today's class:

1. Brief overview of fundamental concepts (AI,ML,DL, ANN).
2. Application in the field of Materials Engineering: Master's Thesis Project.
3. Implementation hands-on.

Brief overview of fundamental concepts: from the general to the specifics

1

Relationship between concepts AI, ML, DL, ANN

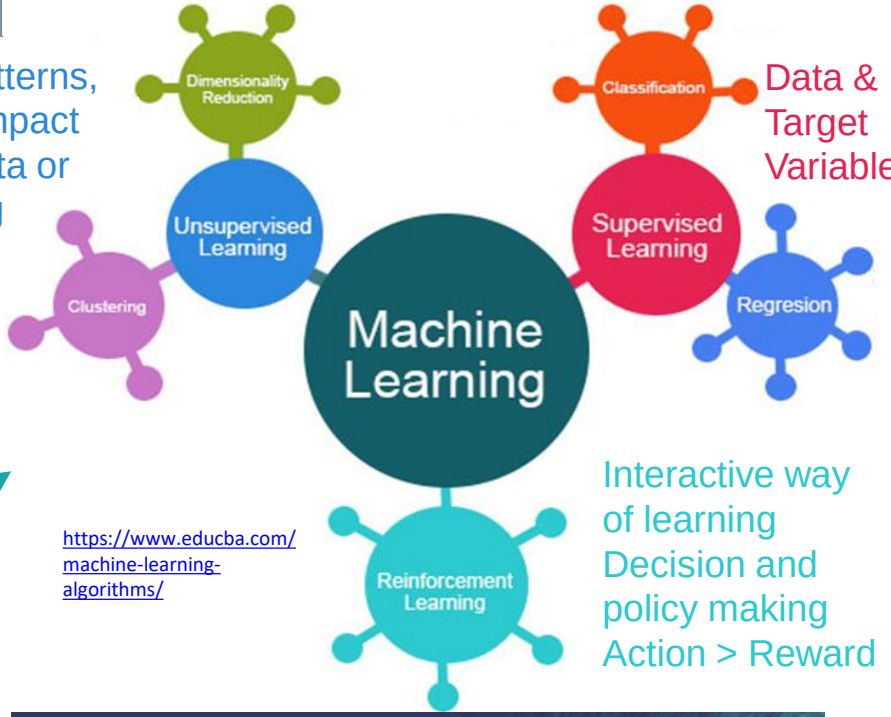


https://www.qubole.com/blog/deep-learning-the-latest-trend-in-ai-and-ml/

Paradigms

Algorithms

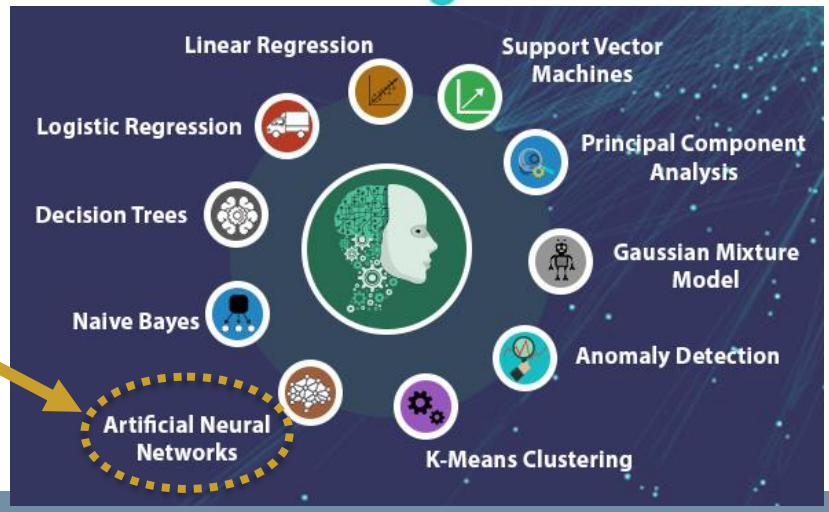
Finding structures and patterns, groups and clusters, compact representation of the data or generative modeling



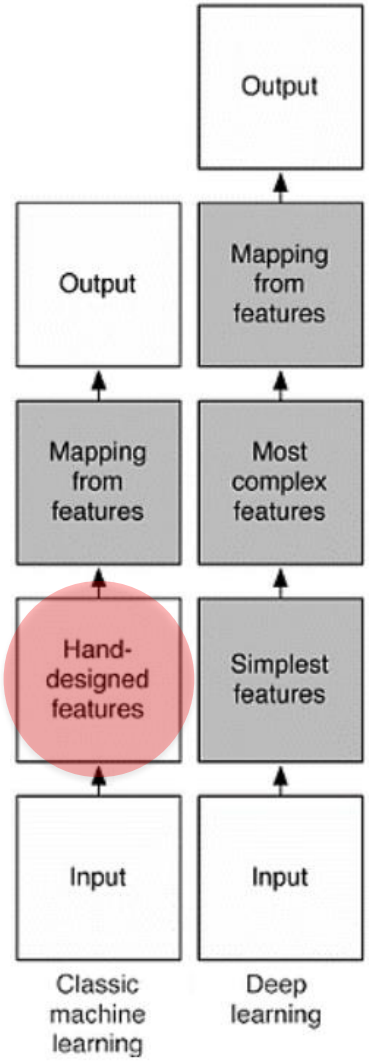
Data & Target Variable

https://www.educba.com/machine-learning-algorithms/

Interactive way of learning Decision and policy making Action > Reward

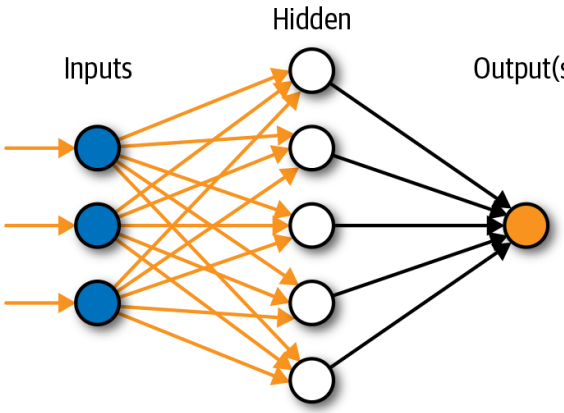


Conceptual difference between Classic ML and Deep Learning



https://towardsdatascience.com/machine-learning-vs-deep-learning-62137a1c9842

DL is a specific class of ML algorithms which use complex Deep ANNs!

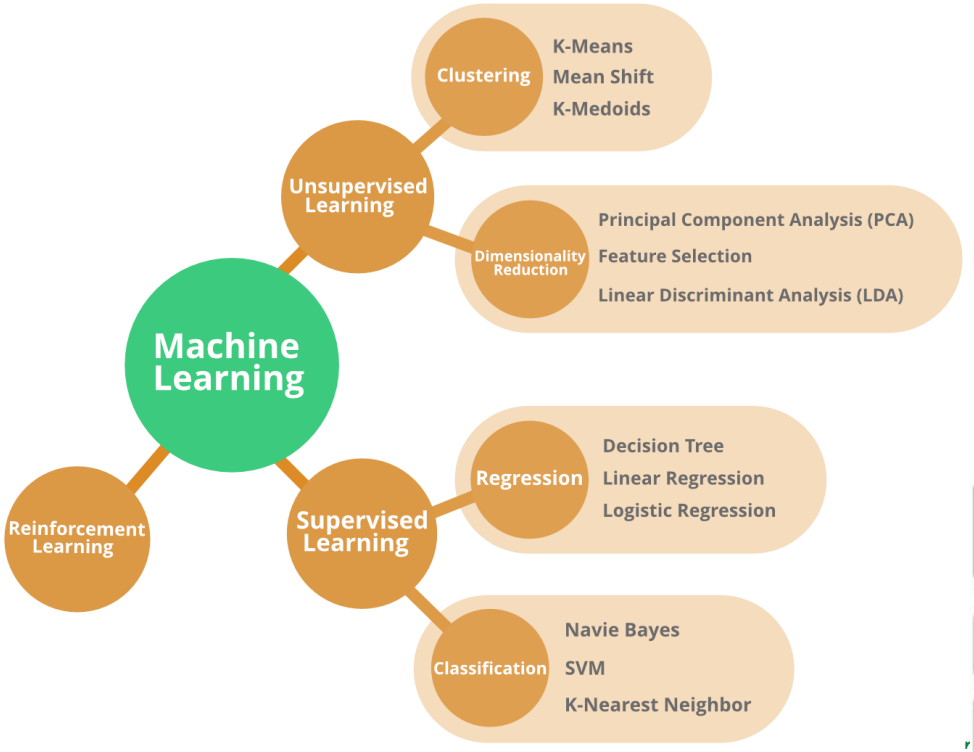


Types of Artificial Neural Networks

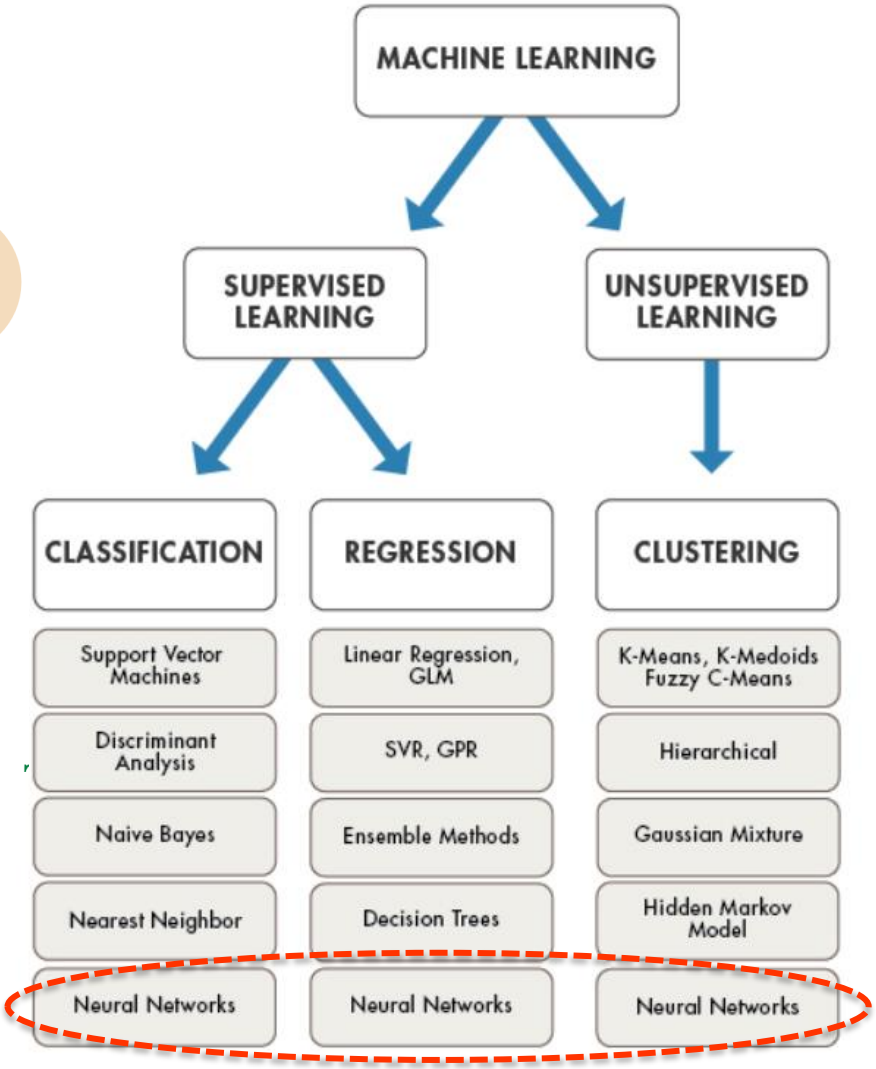
- Feedforward neural network (FNN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Generative Adversarial Neural Network (GAN)
- …

Algorithms for Supervised and Unsupervised Learning

https://www.natureknowsproducts.com/the-10-algorithms-every-machine-learning-engineer-should-know/

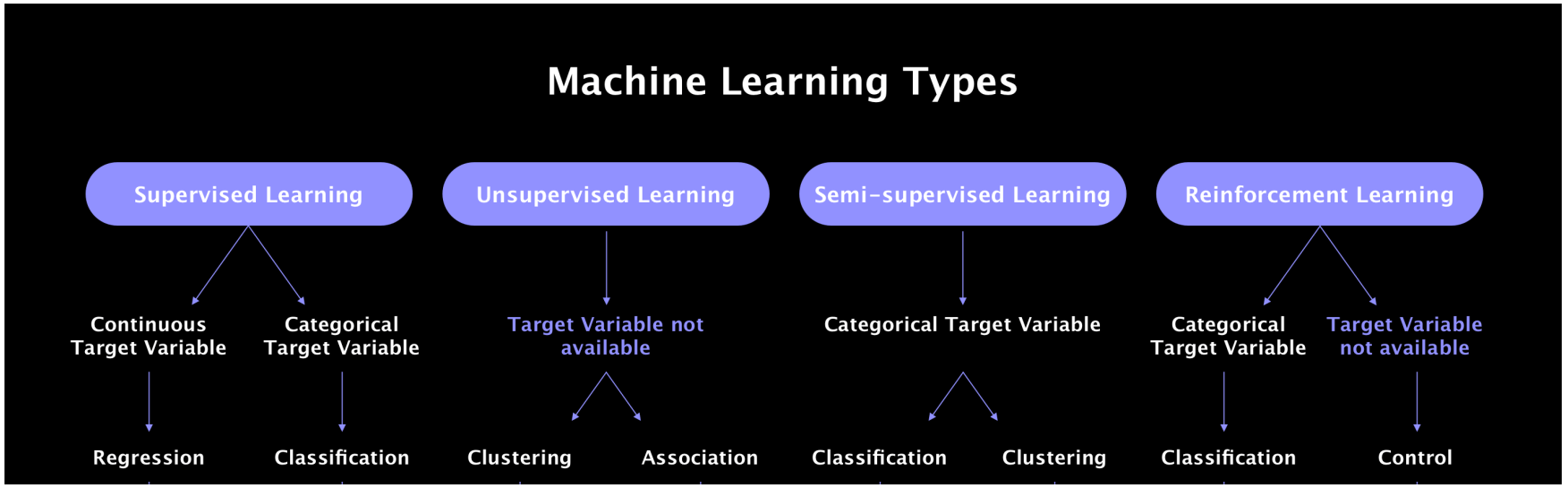


ANNs constitute the most popular class of ML algorithms, as a result of **the boost in high-performance computing** accompanied by **huge data availability**.

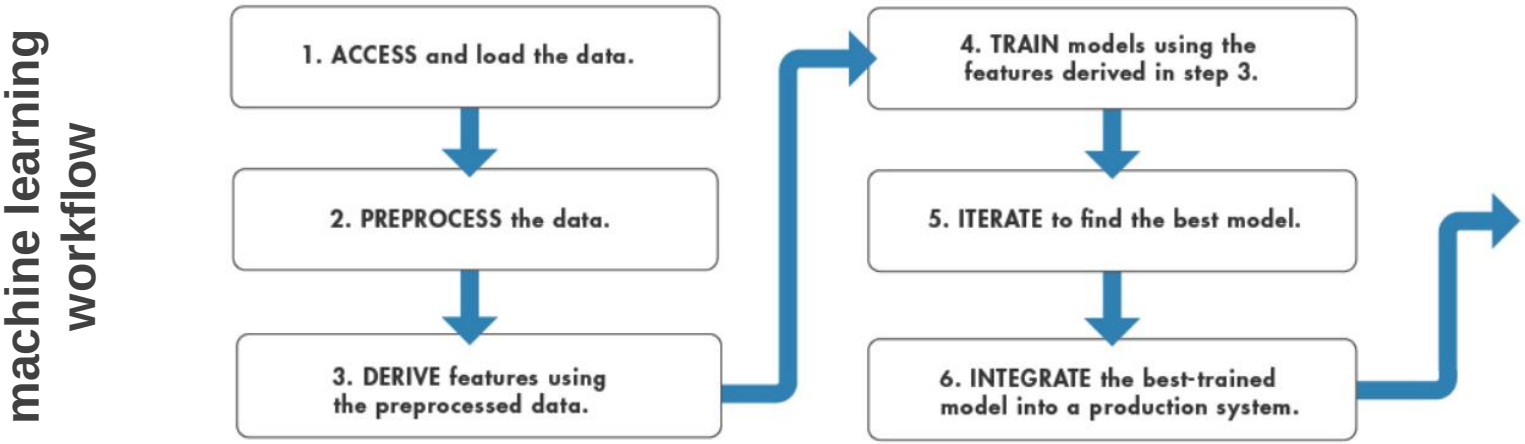


https://se.mathworks.com/help/stats/machine-learning-in-matlab.html

Target variable data types and ML Workflow

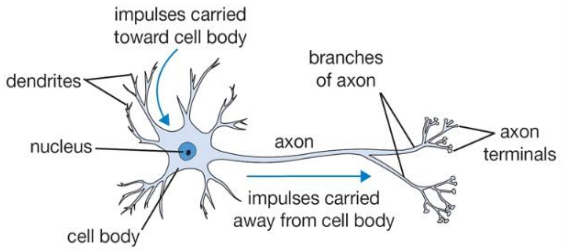


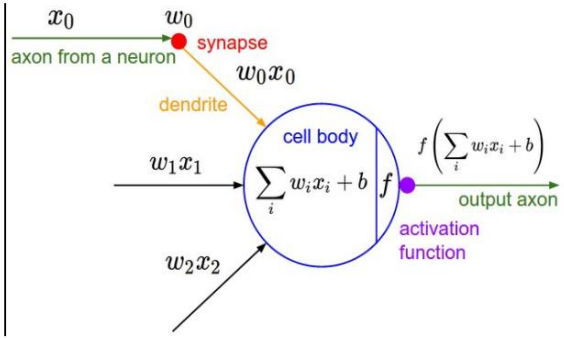
https://litslink.com/blog/an-introduction-to-machine-learning-algorithms



https://se.mathworks.com/help/stats/machine-learning-in-matlab.html

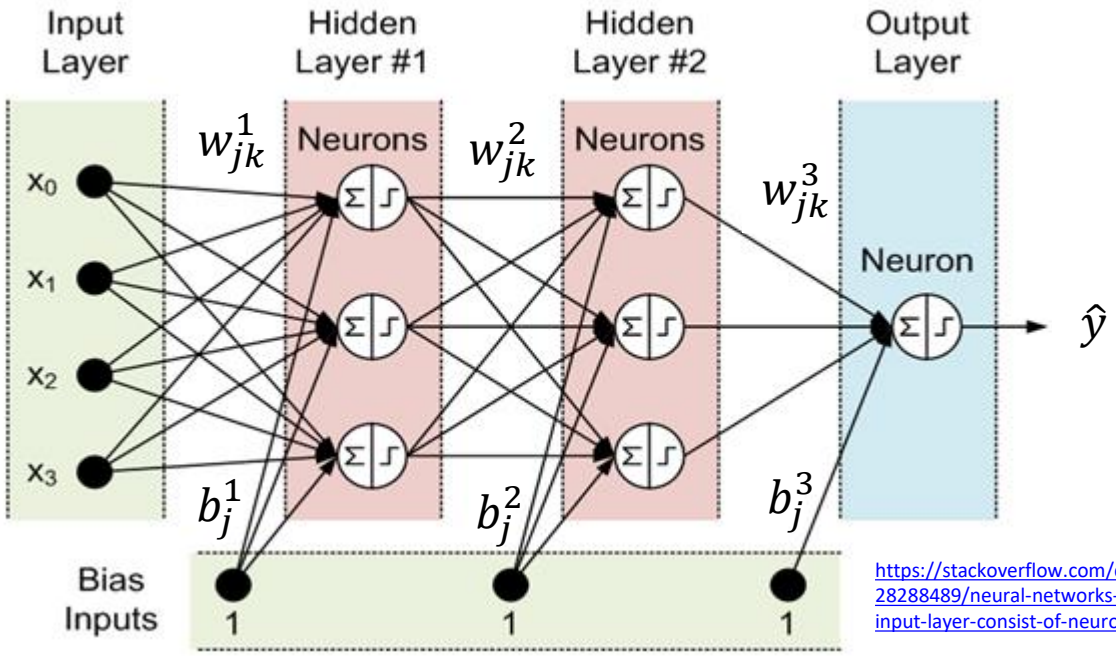
Artificial Neuron: The Perceptron





https://cs231n.github.io/neural-networks-1/

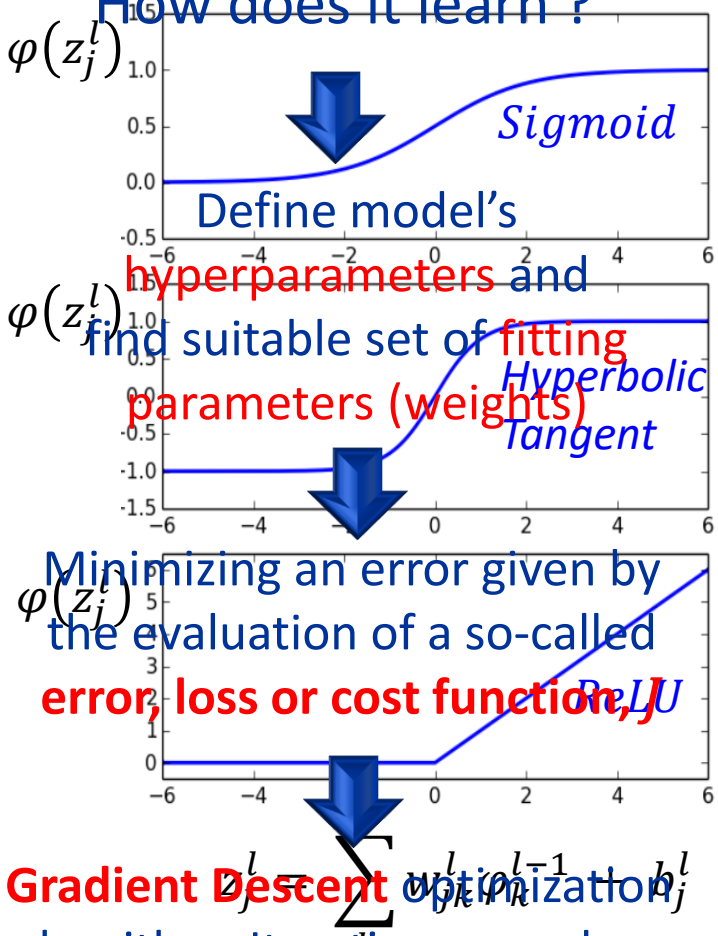
The Multilayer Perceptron (MLP)



https://stackoverflow.com/questions/28288489/neural-networks-does-the-input-layer-consist-of-neurons

How does it learn ?

Hughes C. N., (2016)



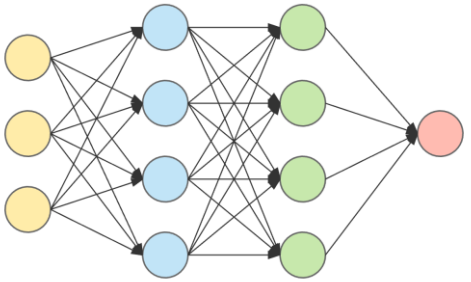
Define model's hyperparameters and find suitable set of fitting parameters (weights)

Minimizing an error given by the evaluation of a so-called **error, loss or cost function, J**

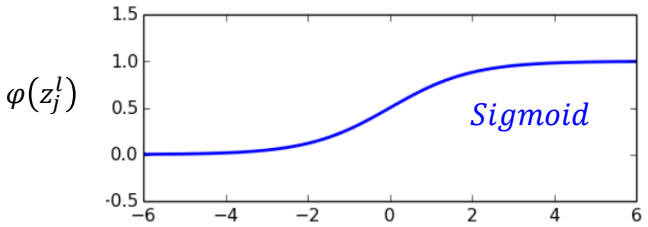
$$z_j^l = \sum_k w_{jk}^l \varphi_k^{l-1} + b_j^l$$

Gradient Descent optimization algorithm. Iterative procedure referred to as the training.

Gradient Descent (GD) Optimization Algorithm



1. **Initialize weights (internal parameters to be optimized)**.
2. **From the input $\boldsymbol{x}$**, compute the activation vector $\boldsymbol{\varphi}^1$ for the input layer l =1.
3. **Feedforward:** For each l =2, 3, …,L compute $\boldsymbol{z}^l \equiv \boldsymbol{w}^l \boldsymbol{\varphi}^{l-1} + \boldsymbol{b}^l$ and $\boldsymbol{\varphi}^l = \boldsymbol{\varphi}(\boldsymbol{z}^l)$
4. **Output error $\boldsymbol{\delta}^L$:** Compute the vector $\boldsymbol{\delta}^L = \nabla_\varphi J \odot \boldsymbol{\varphi}'(\boldsymbol{z}^L)$ where:



$$J \equiv \text{loss function e.g. } J_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

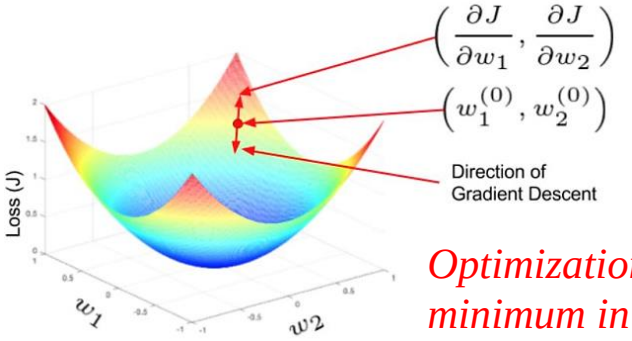
$\odot \equiv$ element wise (Hadamard) product

$\nabla_\varphi \equiv$ gradient operator with respect the activations ($\varphi_i^L = \hat{y}_i$)

5. **Backpropagate the error**: For each $l = L-1, L-2, \ldots, 1$ compute

$$\boldsymbol{\delta}^l = \left(\left(\boldsymbol{w}^{l+1}\right)^T \boldsymbol{\delta}^{l+1}\right) \odot \boldsymbol{\varphi}'(\boldsymbol{z}^l)$$

6. **Compute** the gradient of the cost function is given by $\partial J/\partial w_{jk}^l = \varphi_k^{l-1}\delta_j^l$ and $\partial J/\partial b_j^l = \delta_j^l$
7. **Update** the weights and biases for next iteration using the learning rate, α:



$$\left(w_{jk}^l\right)^{\tau+1} = \left(w_{jk}^l\right)^{\tau} - \alpha\frac{\partial J}{\partial w_{jk}^l} \quad ; \quad \left(b_j^l\right)^{\tau+1} = \left(b_j^l\right)^{\tau} - \alpha\,\partial J/\partial b_j^l$$

Optimization = find the local minimum in the loss landscape

Key! : Different adaptations to the standard parameter update equation, aiming to improve the convergence properties (**optimizer selection**).

Implementation of GD Optimization Algorithm

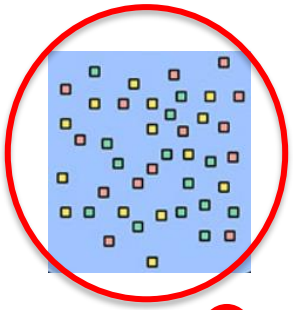
There are 3 main variations for the generic GD that arise due to practical **implementation asptects:**

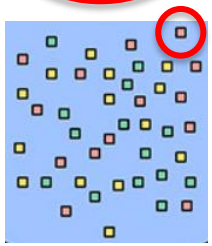
Batch-Gradient Descent: computes the **exact gradient** using the whole dataset. A single iteration which is also an epoch!

Stochastic Gradient Descent (SGD): computes a **stochastic approximation of the gradient** using each single data point, every iteration. Each single sample produces a weight update.

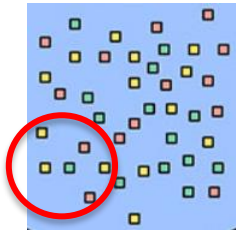
Mini-Batch Gradient Descent: consists of a combination of the previous two methods. The training set is split into small batches or mini-batches. The weights are updated over each minibatch.

Memory limitations!





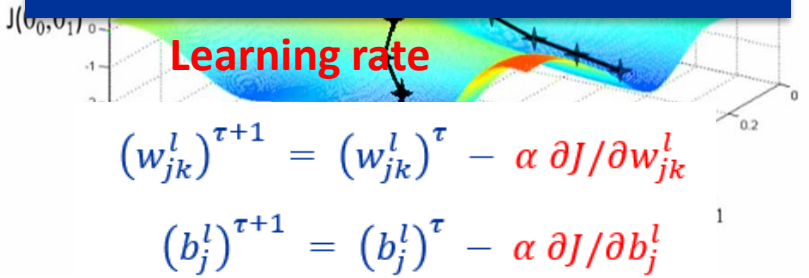
Noisy convergence! Jerk the model out of local minima!



Balance

Challenges During Optimization using GD algorithm

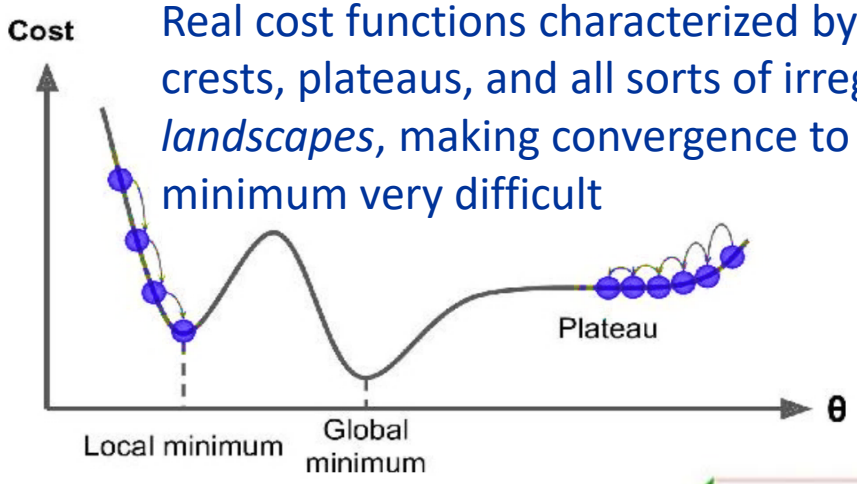
Second order optimization methods incorporate curvature information to scale the gradient dynamically during the training!



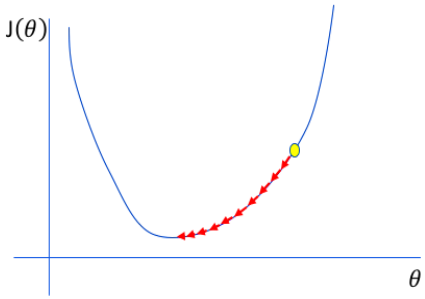
$$(w_{jk}^{l})^{\tau+1} = (w_{jk}^{l})^{\tau} - \alpha\, \partial J/\partial w_{jk}^{l}$$

$$(b_{j}^{l})^{\tau+1} = (b_{j}^{l})^{\tau} - \alpha\, \partial J/\partial b_{j}^{l}$$

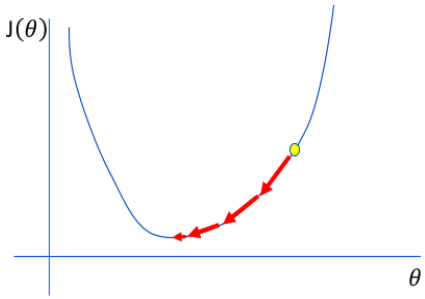
Degree of convexity of the loss surface



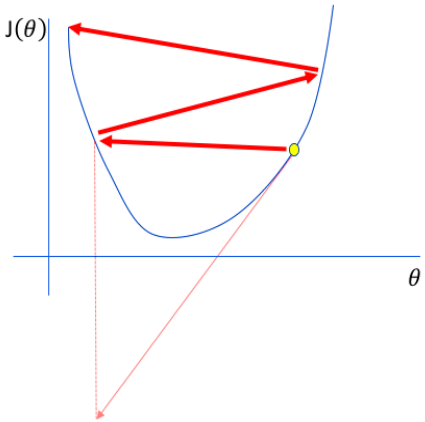
Real cost functions characterized by holes, crests, plateaus, and all sorts of irregular *landscapes*, making convergence to the minimum very difficult



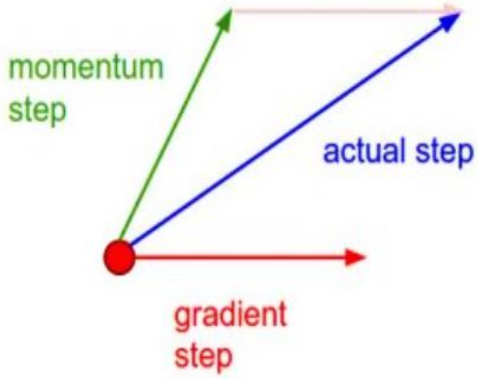
A *small* learning rate requires many updates and computing time



Adaptative learning rate optimizers (Adagrad, Adadelta, RMSprop)

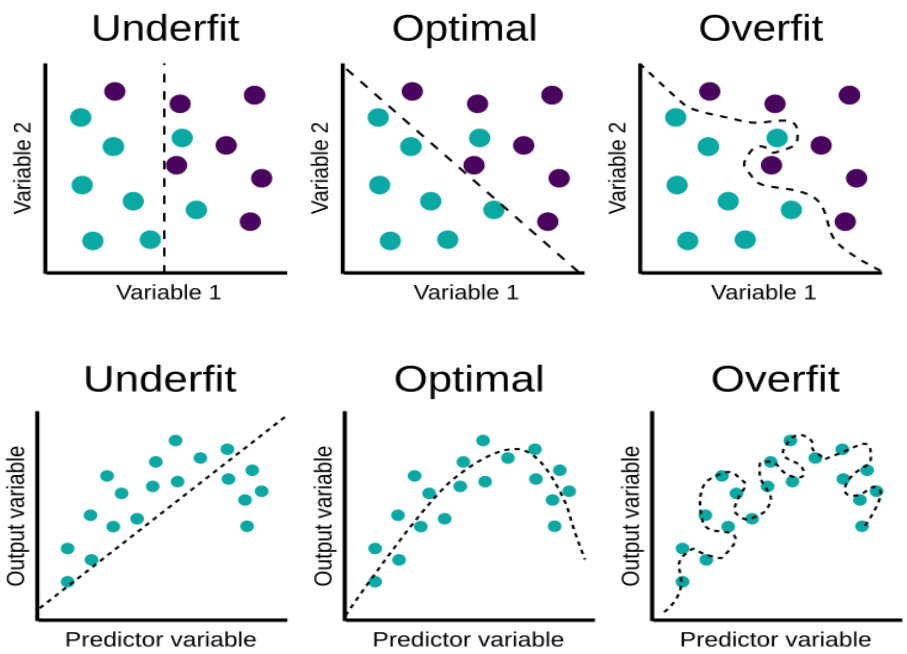


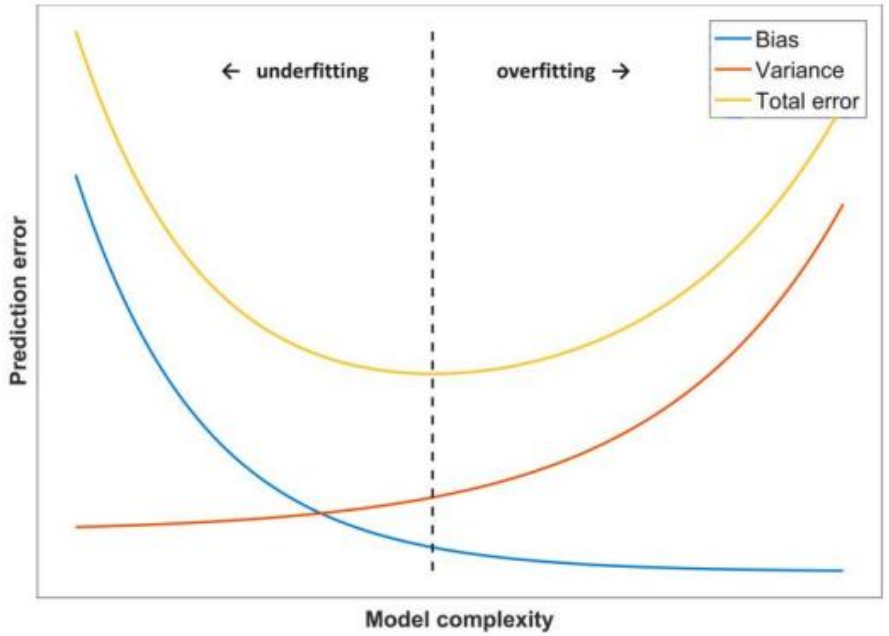
Too large values cause drastic update



Momentum update
Average gradient instead of the immediate gradient

Challenges of the Learning process



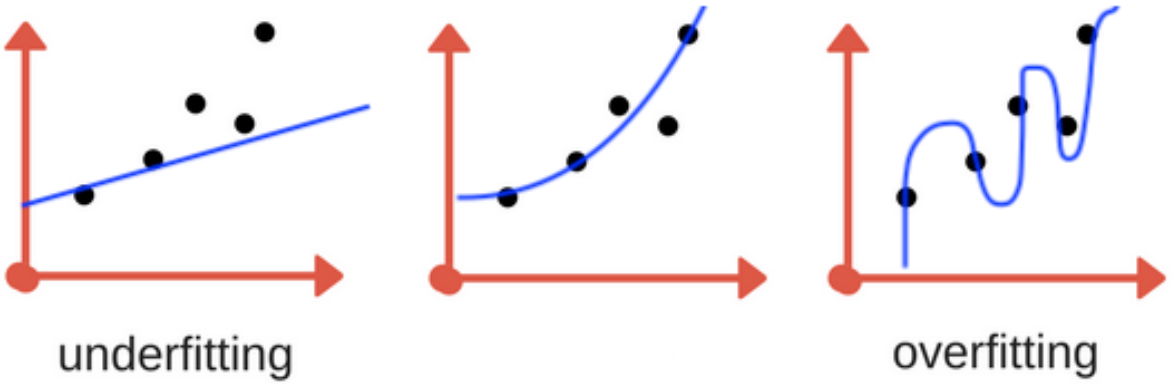


Too simple to learn the underlying structure of the data!

Model performs well on the training data, but it does not generalize well!

Complexity of the model is **proportional to the number of fitting or trainable parameters.**

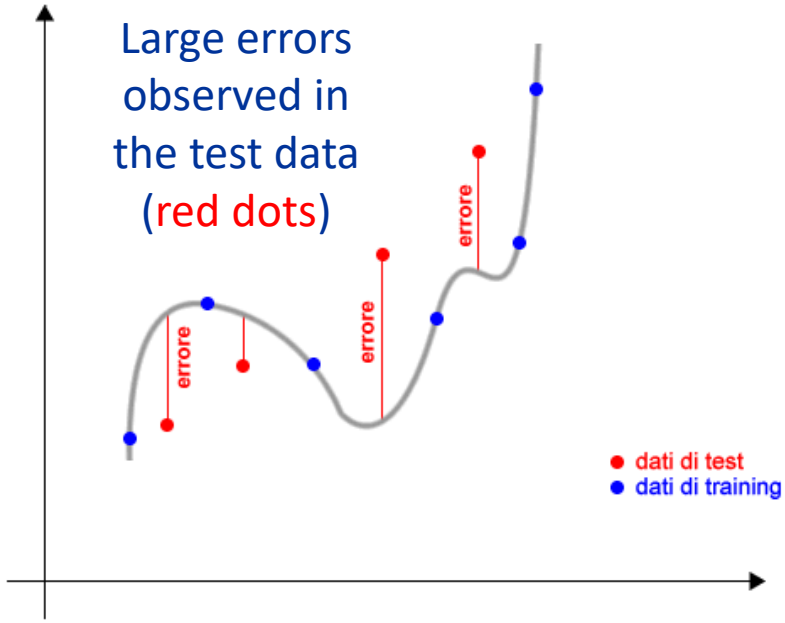
Challenges of the Learning process



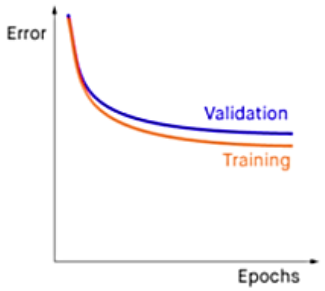
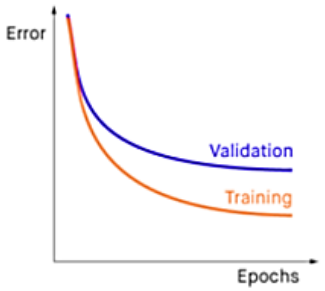
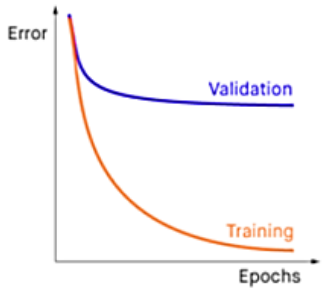
The **overfitted** model (gray line) exactly fits the training data (blue dots)

With increased model complexity the model can more accurately match the underlying relation at the risk of increasing the variance (**Overfitting**)

Large errors observed in the test data (red dots)



Assessing Underfitting and Overfitting: *loss curve evolution*

	Underfitting	Just right	Overfitting
Symptoms	→ High train error → Training error close to test error → High bias	Training error slightly lower than test error	→ Very low training error → Training error much lower than test error → High variance
Loss curve			
Possible remedies	→ Create a more complex model → Add more features → Train longer		→ Perform regularization → Get more data

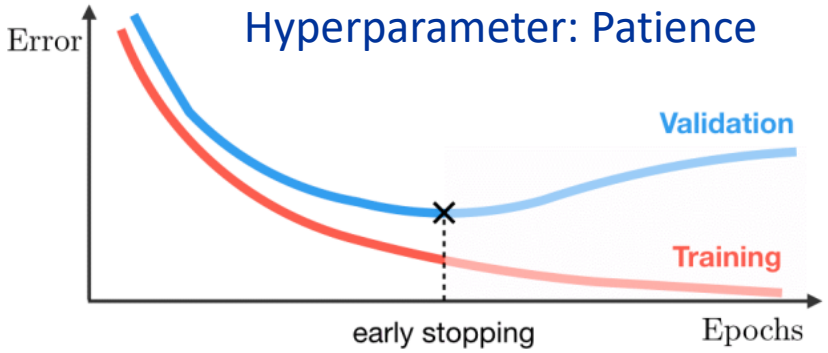
Typical regularization techniques:

- Early stopping
- L1 and L2 regularization
- Dropout
- Data augmentation (idea: artificially generated data)

(Some) Regularization techniques

Early stopping

Hyperparameter: Patience



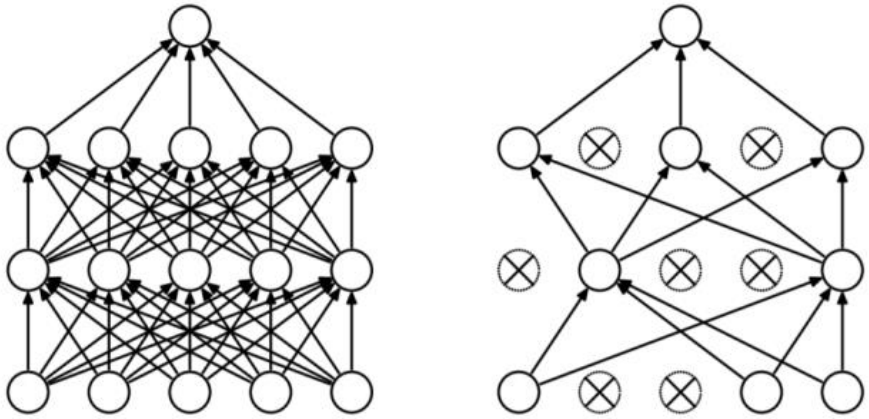
L1 and L2 regularization

Hyperparameter: λ

$$J_{L1\ regularization}(\theta) = J(\theta) + \lambda \sum_{i=1}^{n} |\theta_i|$$

$$J_{L2\ regularization}(\theta) = J(\theta) + \lambda \sum_{i=1}^{n} \theta_i^2$$

Dropout

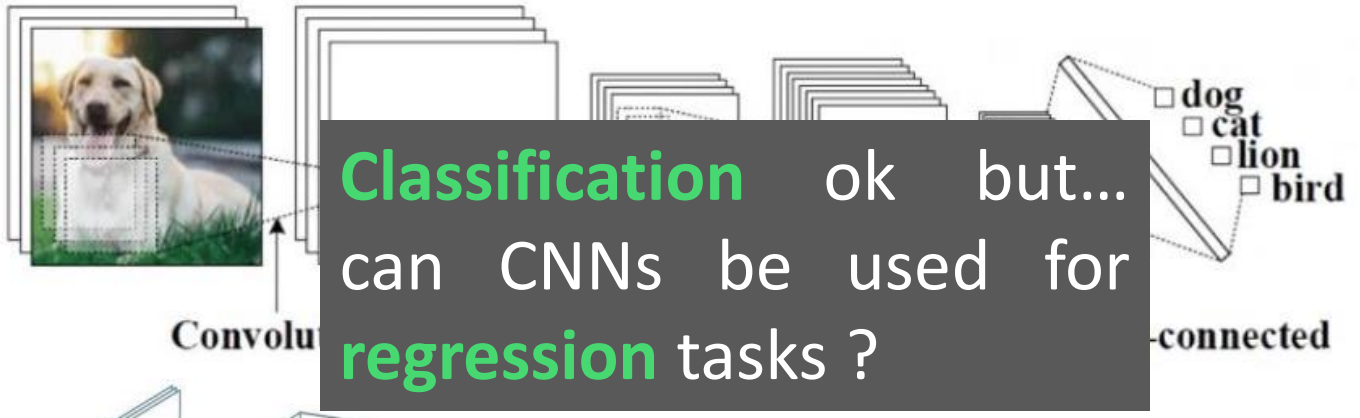


Hyperparameter: Dropout rate *p*

Data augmentation

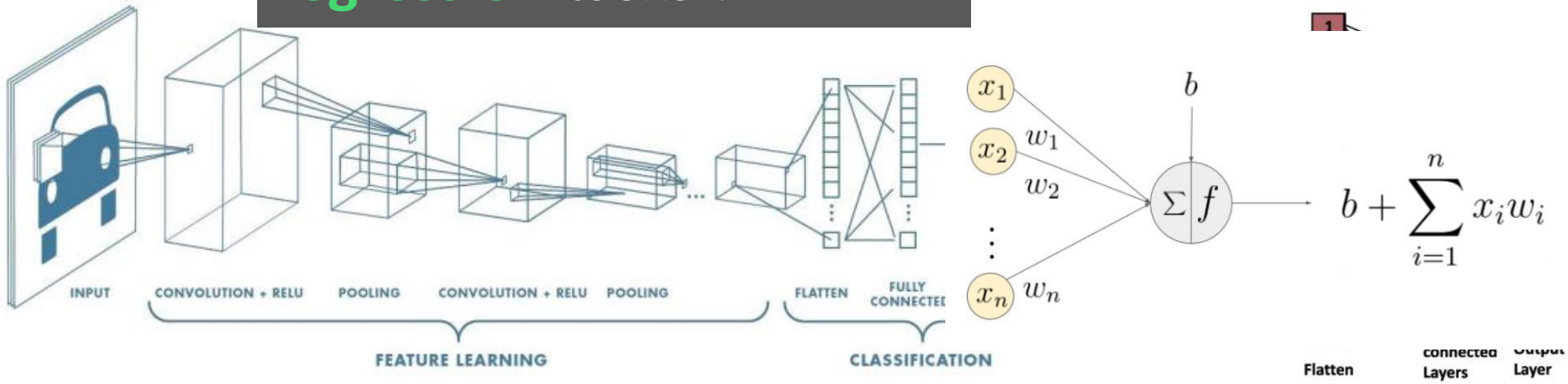
Spatially Correlated Data: Convolutional Neural Networks

- CNNs have emerged from the study of the brain's visual cortex
- Able to learn **position and scale invariant structures in the data** allow CNNs to be used more generally on other types of data that preserves a spatial relationship (speech recognition, time series analysis etc.)



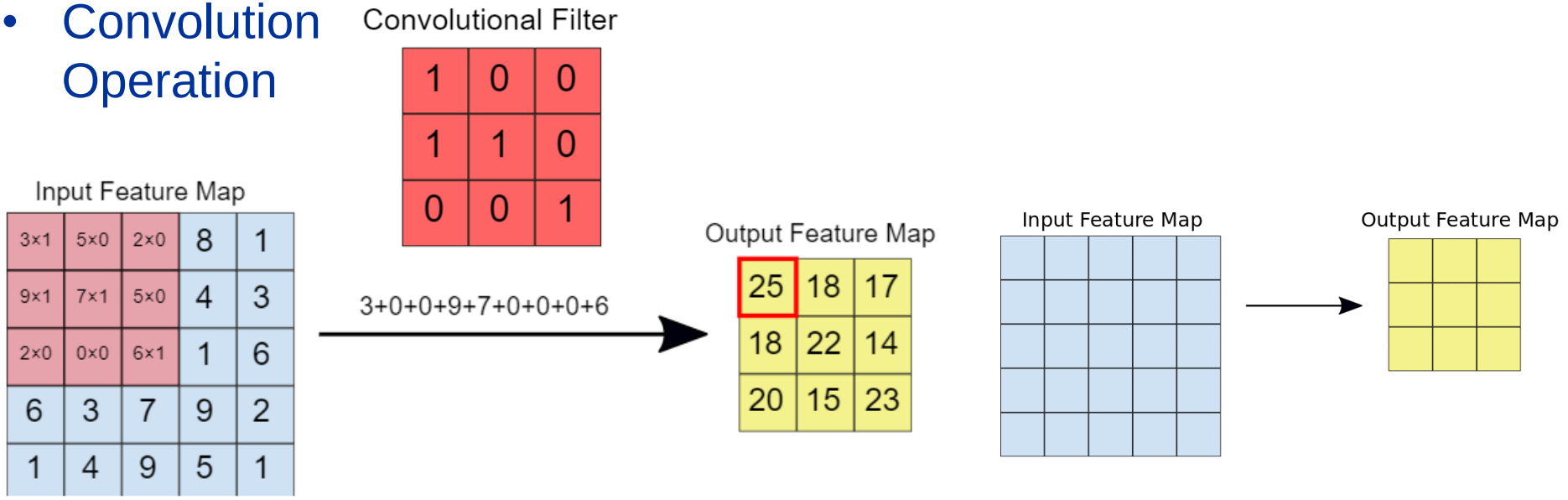
Classification ok but… can CNNs be used for **regression** tasks ?

Requires fewer fitting parameters due to the introduction of "special" type of layers



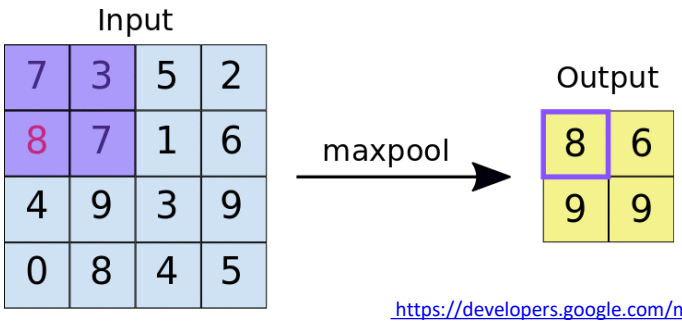
Main operations in CNNs

- Convolution Operation



https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks?hl=id

- Pooling Operation



https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks?hl=id

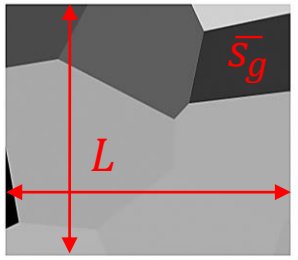
Application in the field of materials engineering: Master's Thesis Project

2

Application: Study of the effective mechanical properties in Poly-Si thin films typically found in MEMS devices

- At the length scale considered $L/\bar{s}_g \neq \infty$

  Homogenized properties yield different values from one realization to another (SVE). Ability to characterize the dispersion of these values is key to predict MEMS performance and define reliability indices.

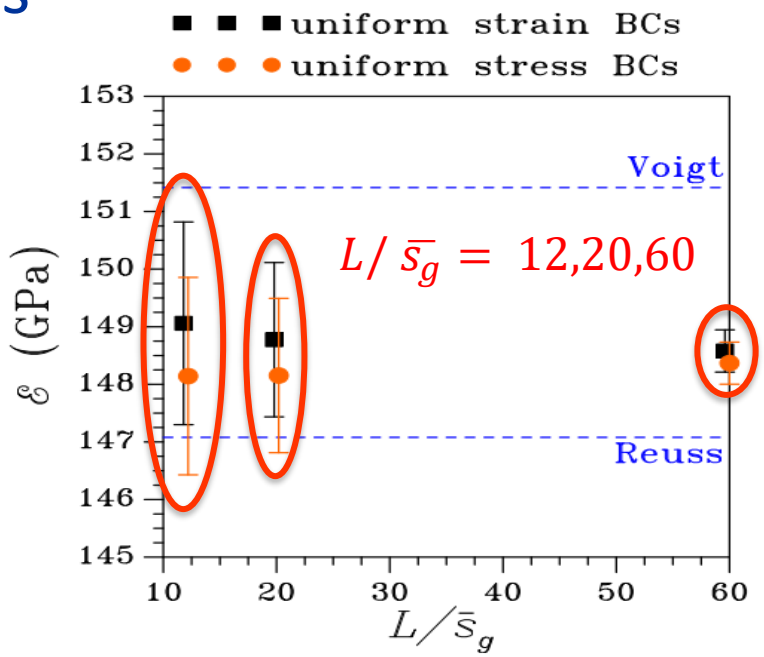


SVE: Stochastic Volume Element

- Scattering in the overall properties

  Scale dependent solution also known as size effect

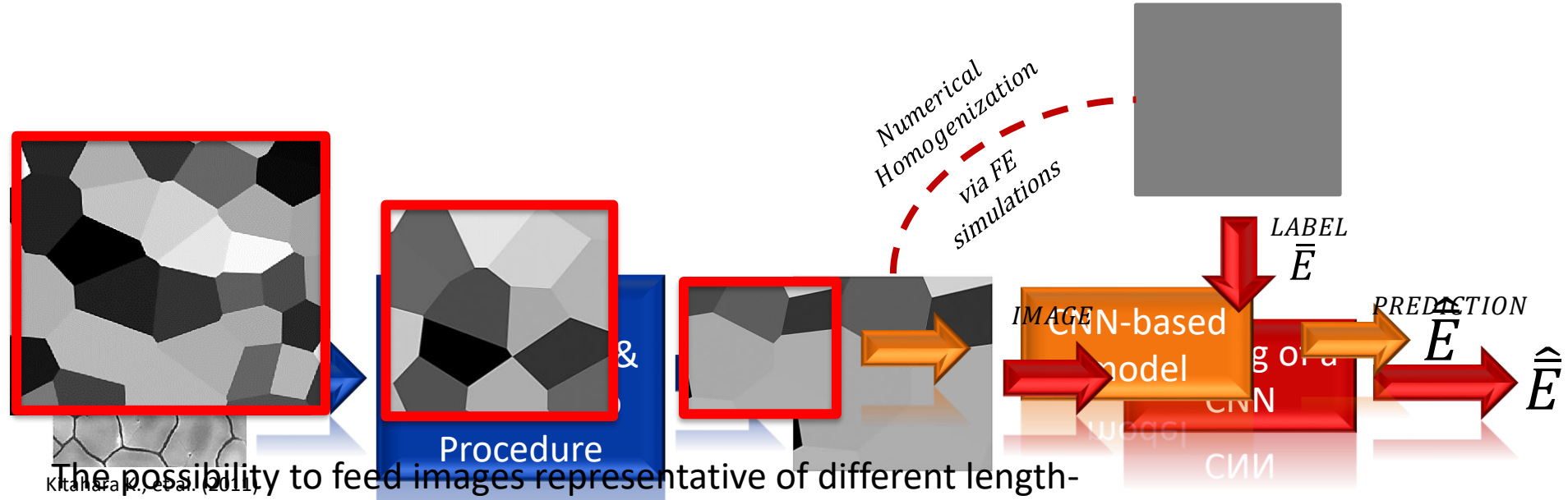
Goal: Tool for a fast multiscale statistical characterization of the scattering on effective (apparent) properties induced by the material's microstructural features (**topology of the grain boundary network** and the **lattice orientation of each grain**).



Mariani S, et al. (2011)

Application: Effective Properties in Poly-Si thin films typically used in MEMS (Master's Thesis)

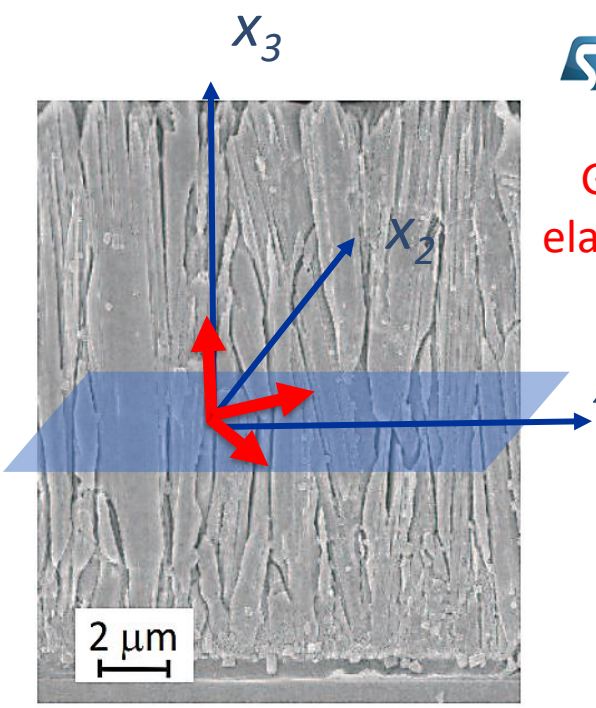
- ✓ The proposed approach represents an alternative to standard homogenization techniques.
- ✓ Limited to the characterization of 2D stochastically representative realizations of Poly-Si thin films typical of MEMS devices.
- ✓ A DL algorithm is proposed. Based on the implementation of a Convolutional Neural Network (**Image** & **Label**: Prediction)



The possibility to feed images representative of different length-scales allows for a fast multiscale exploration of size effects!

Input Data Generation : Polysilicon Films

Polysilicon Epitaxial Growth
(lateral view)

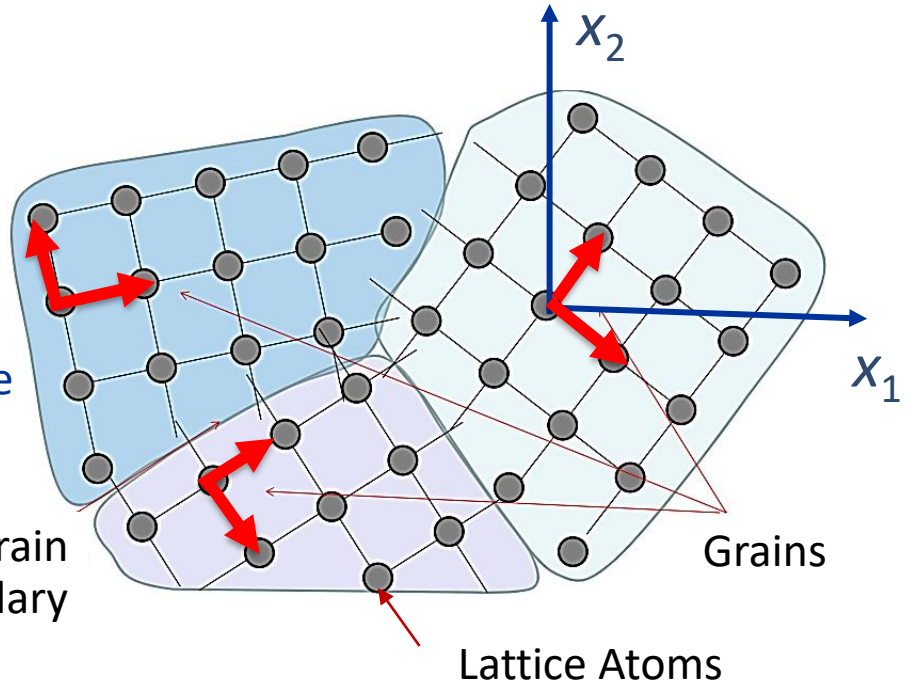


ThELMA™
STMicroelectronics

Grain axes of elastic symmetry <1 0 0>

Global reference frame (x_1 x_2 x_3)

Taking a slice
(thin film under plane stress cond.)

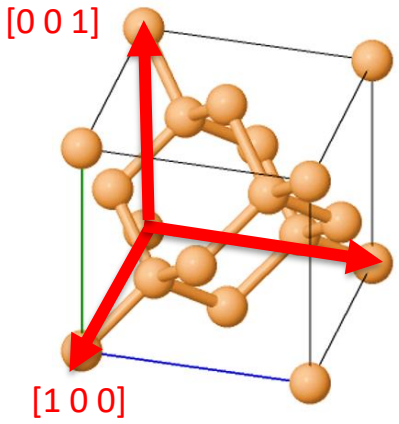


< hkℓ > denotes the set of all directions that are equivalent to [hkℓ] by symmetry

- One axis of elastic symmetry aligned with epitaxial growth direction i.e. x_3
- Random orientation of other two elastic symmetry directions in the x_1- x_2 plane.

Silicon Anisotropic Elasticity

Crystal Structure of Si



Diamond cubic lattice

Anisotropic Elasticity

http://lampx.tugraz.at/~hadley/memm/materials/silicon/silicon.php

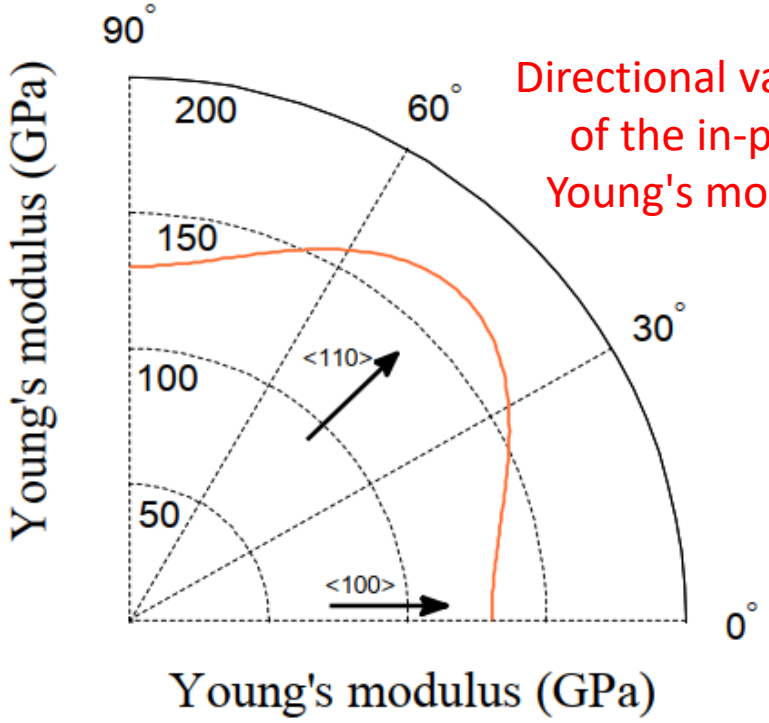
Stiffness Matrix of Silicon

$$\begin{pmatrix} 165.64 & 63.94 & 63.94 & 0 & 0 & 0 \\ 63.94 & 165.64 & 63.94 & 0 & 0 & 0 \\ 63.94 & 63.94 & 165.64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 79.51 & 0 & 0 \\ 0 & 0 & 0 & 0 & 79.51 & 0 \\ 0 & 0 & 0 & 0 & 0 & 79.51 \end{pmatrix}$$

*< 1 0 0 > aligned with (x_1 x_2 x_3)

* Reported in [GPa]

Epitaxially grown thin-films: One axis of elastic symmetry e.g. [0 0 1] aligned with epitaxial growth direction e.g. x_3

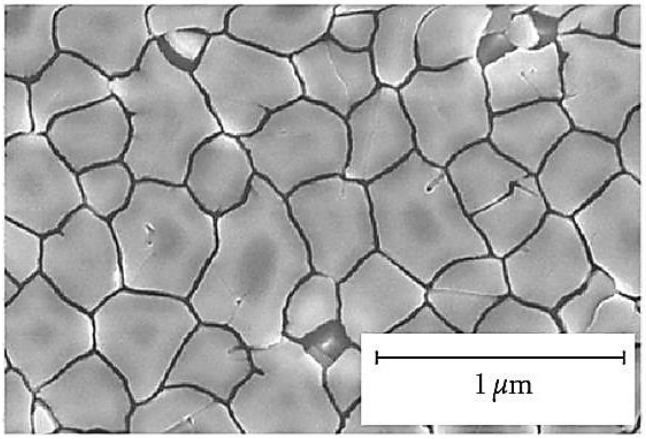


Directional variation of the in-plane Young's modulus!

Mirzazadeh R, (2017)

Real Microstructure

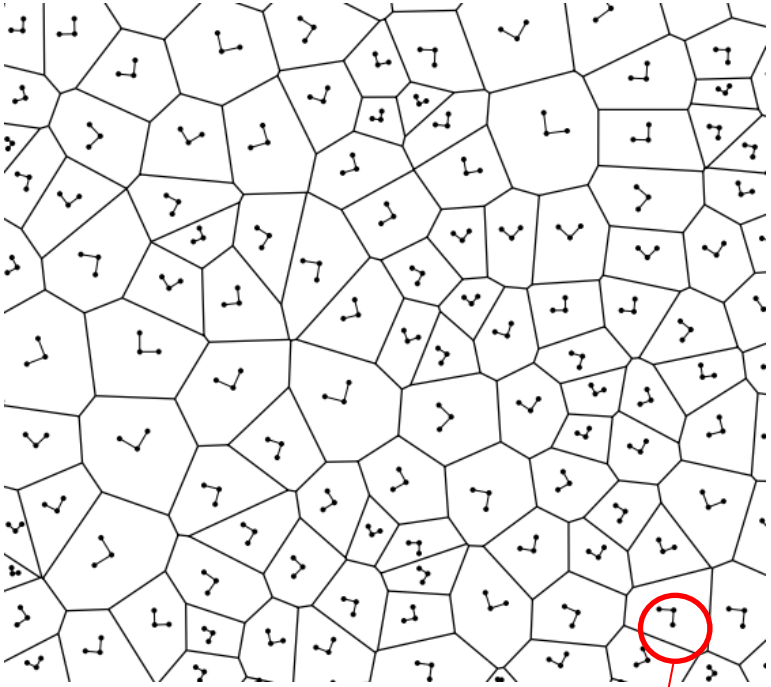
(Poly-Si Thin-Film)



Kitahara K., et al. (2011)

Digital Microstructure

(Voronoi Tessellation: Partition of the plane into n convex regions that represent the grain topology)



Mariani S, et al. (2011)

https://community.wolfram.com/groups/-/m/t/1202074

In-plane lattice orientation, θ

Monte Carlo procedure exploited to account for the stochastic effects of

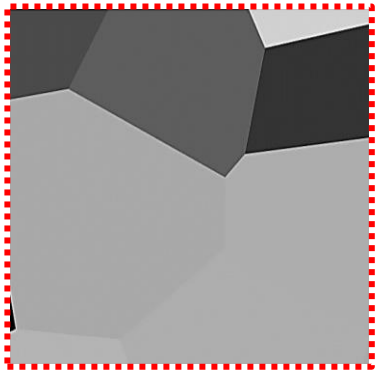
- ✓ Topology of grain boundaries (deployment sites)
- ✓ Lattice orientations (θ)

A predefined average distance between generator points defines $\bar{s}_g$

Fluctuations define actual topology

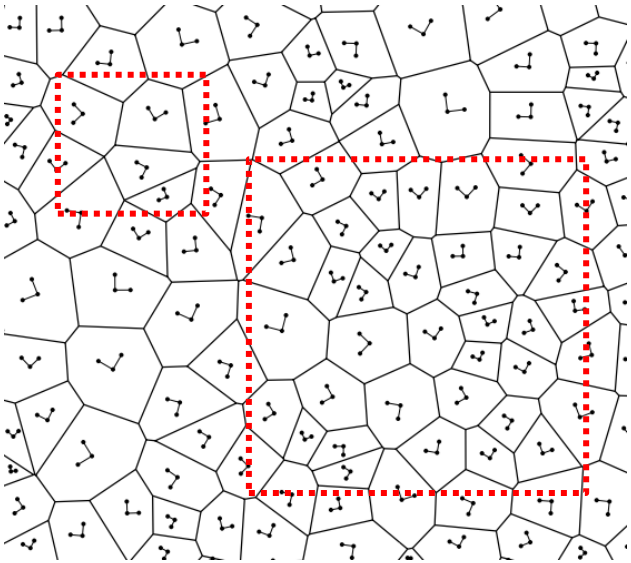
Input Data Generation: Datasets & Labels

$$\left.\begin{array}{l} \mathrm{L} = 2\mu\mathrm{m} \\ \bar{s_g} = 0.5\ \mu\mathrm{m} \end{array}\right\} \frac{L}{\bar{s_g}} = 4$$



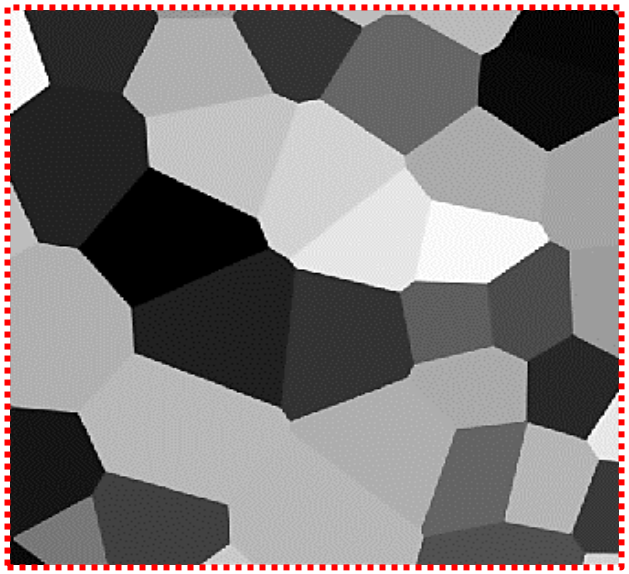
✓ Training Set

✓ Validation Set



Mariani S, et al. (2011)

$$\left.\begin{array}{l} \mathrm{L} = 5\ \mu\mathrm{m} \\ \bar{s_g} = 0.5\ \mu\mathrm{m} \end{array}\right\} \frac{L}{\bar{s_g}} = 10,6$$

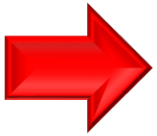


✓ Test Sets

Assessment of the generalization of the model

Pixels take values $[0, 255]$ encoding the in-plane lattice orientation

Ground-truth data $\bar{E}$?

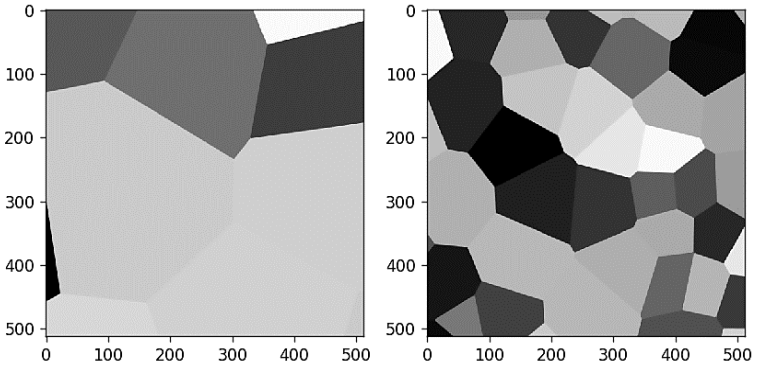


FE Simulations

Input Data Pre-Processing: Resolution and Digital Filtering

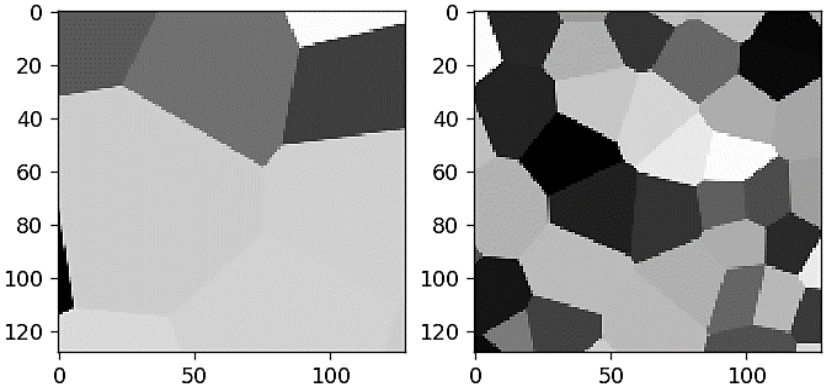
1- Resolution Adjustment

512 x 512 pixels



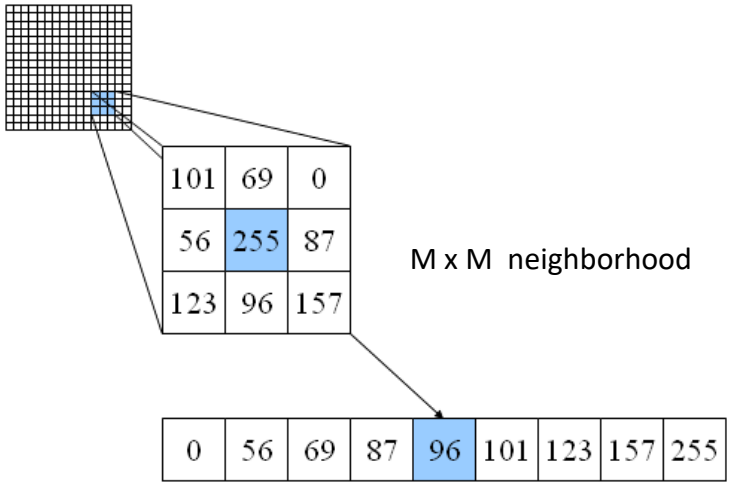
~ 72 s/epoch (~ 2.5 hours) , BS=10

128 x 128 pixels

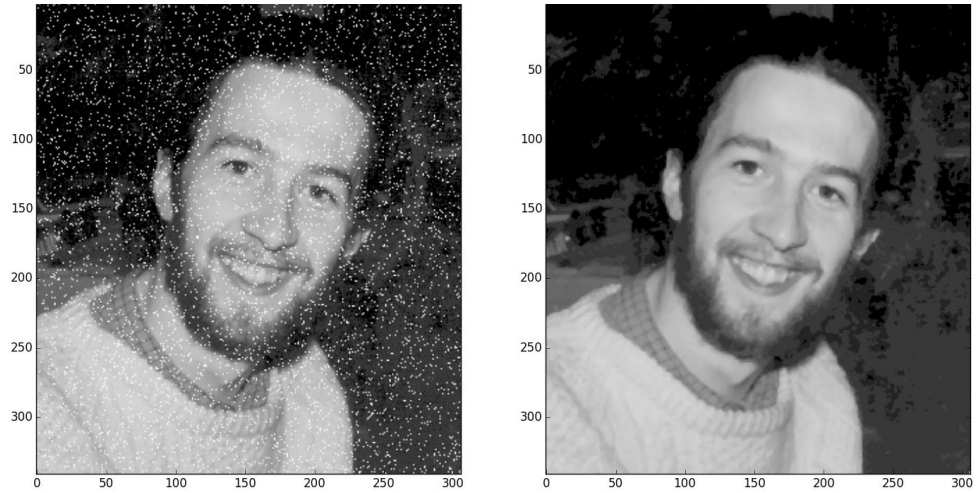


~ 4 s/epoch (~ 8 min), BS=300

2- Median Filter

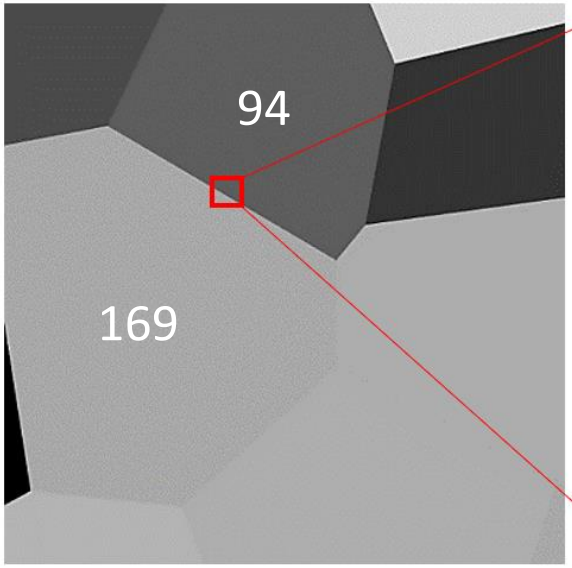


https://www.southampton.ac.uk/~msn/book/new_demo/median/.



https://stackoverflow.com/questions/31308202/what-is-the-noise-remaining-after-median-filtering

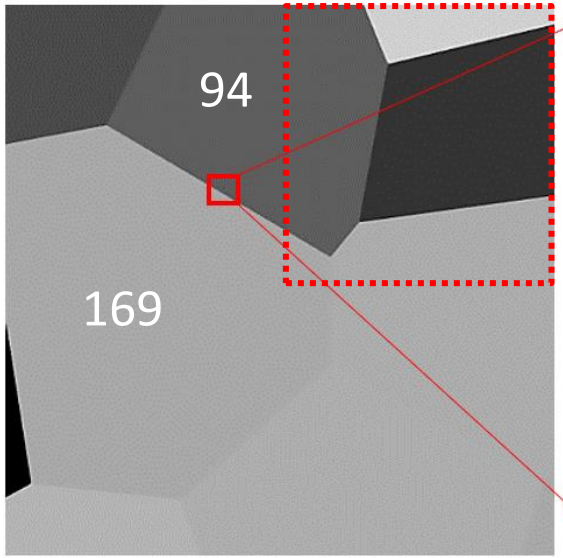
Input Data Pre-Processing: Median Filter Application



94	94	94	94	94	93
94	94	94	94	94	93
92	93	94	95	95	94
91	91	93	95	96	96
96	95	92	91	93	95
91	96	96	91	91	94
126	95	90	99	101	95
161	155	125	98	92	95
166	162	158	143	116	97
176	172	166	167	155	131
168	169	170	168	165	165
169	170	169	169	170	167
168	168	169	169	168	167
170	170	170	169	169	169
170	169	169	168	169	170
169	169	168	168	168	168
169	169	170	169	169	168
169	169	170	171	170	169

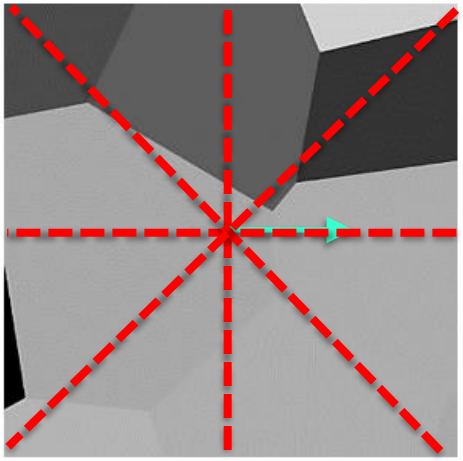
Increasing the size of the kernel was not possible due to **shape distortions**!

Median Filter
Kernel size of [3,3]
Five consecutive times

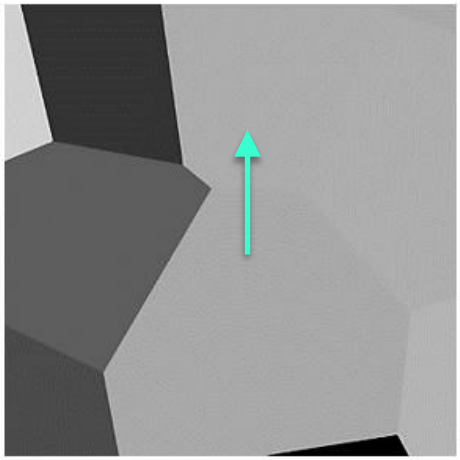


94	94	94	94	94	94
94	94	94	94	94	94
94	94	94	94	94	94
94	94	94	94	94	94
94	93	94	94	95	94
96	98	96	91	92	94
122	97	93	97	97	94
164	155	123	99	94	95
168	160	157	144	119	99
169	170	167	164	152	131
171	169	168	166	165	166
168	170	169	170	170	167
169	169	169	169	169	169
169	169	169	169	169	169
169	169	169	169	169	169
169	169	169	169	169	169
169	169	169	169	169	169
169	169	169	169	169	169

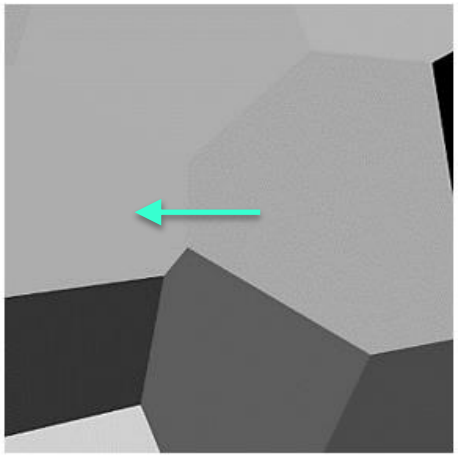
Input Data Pre-Processing: Data Augmentation



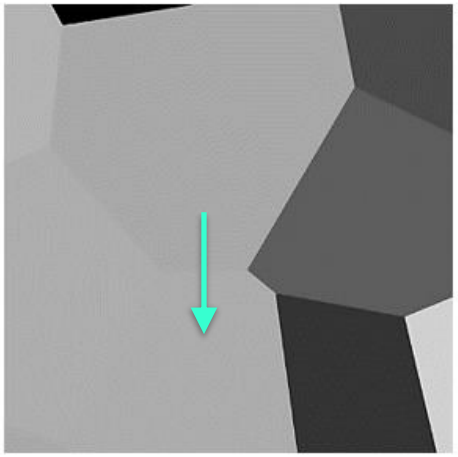
a) Original image



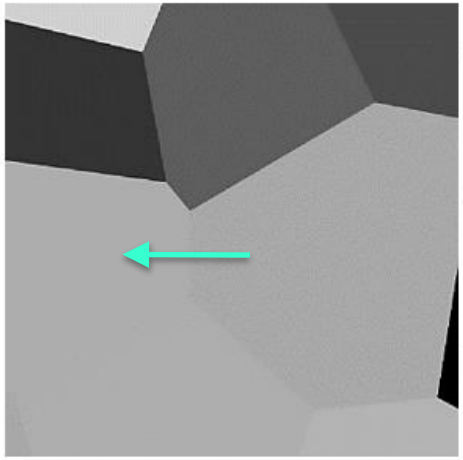
b) 90° Rotation



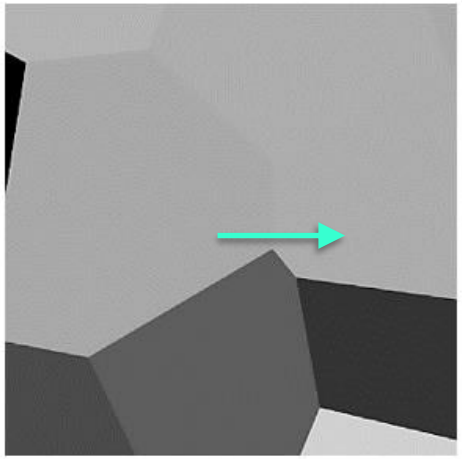
c) 180° Rotation



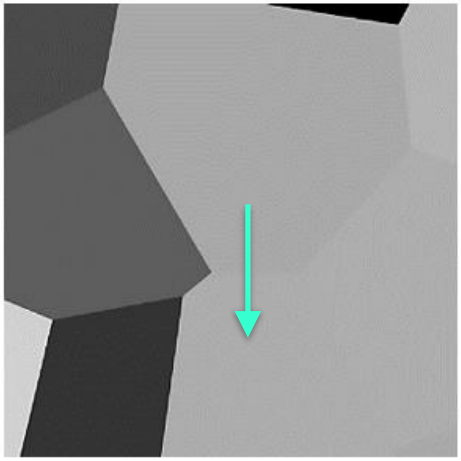
d) 270° Rotation



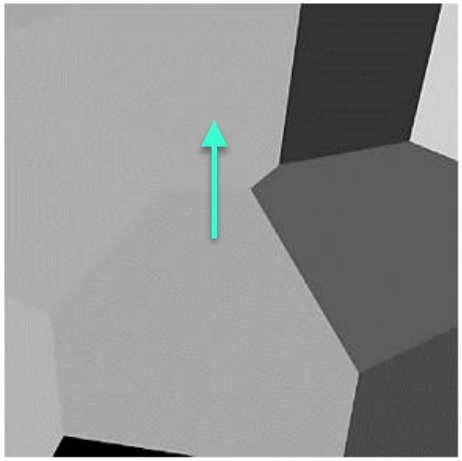
e) Vertical Flip



f) Horizontal Flip



g) First Diagonal Flip



h) Second Diagonal Flip

Model: Proposed CNN-based Architecture

Input

Image

ResNet18

$\hat{E}$

Output

Effective in-plane Elastic Modulus

Flatten

FC (100 nodes) +ReLU

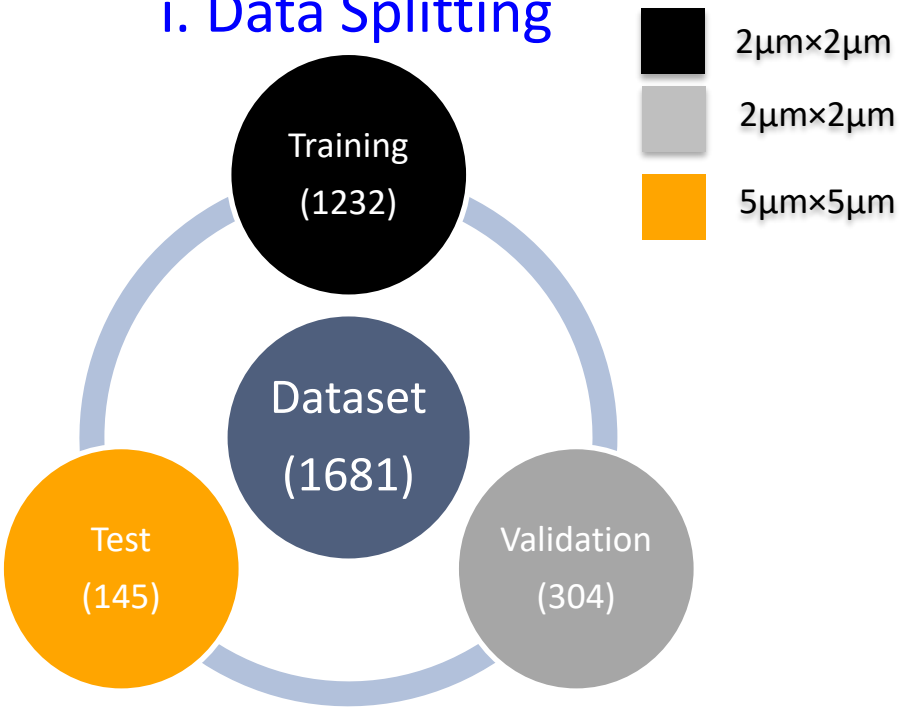
FC (Output) +Linear activation

Feature Learning Block

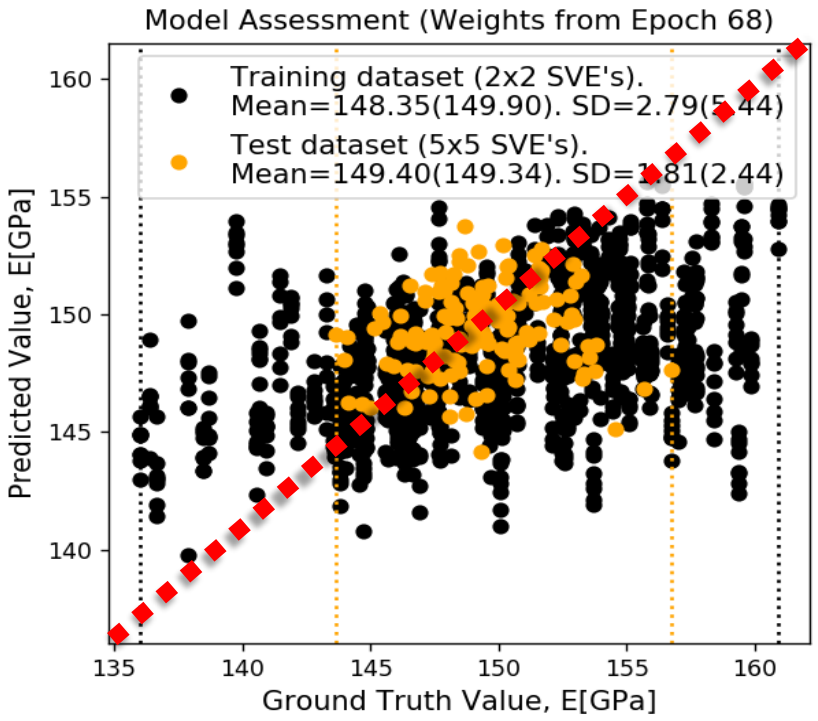
Regression Block

Results: Data Augmentation

i. Data Splitting



ii. Parity Plot



iii. Scattering around the mean

Standard Deviation Absolute Percentage Error:

48.7% for the training dataset (L/ $\overline{s_g}$ =4)

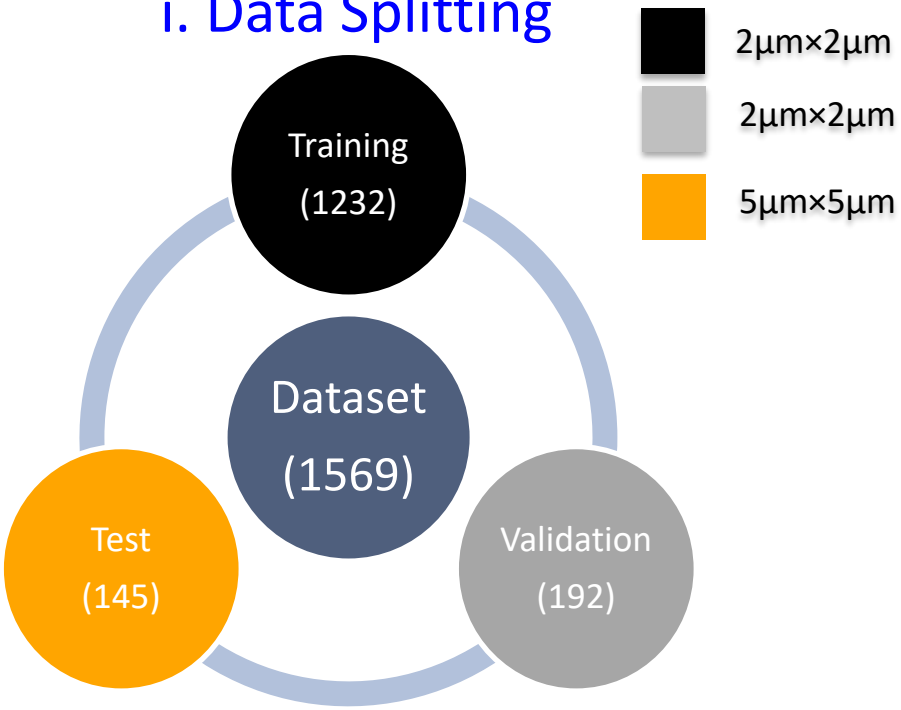
25.8% for the test dataset (L/ $\overline{s_g}$ =10)

Limited number of original images

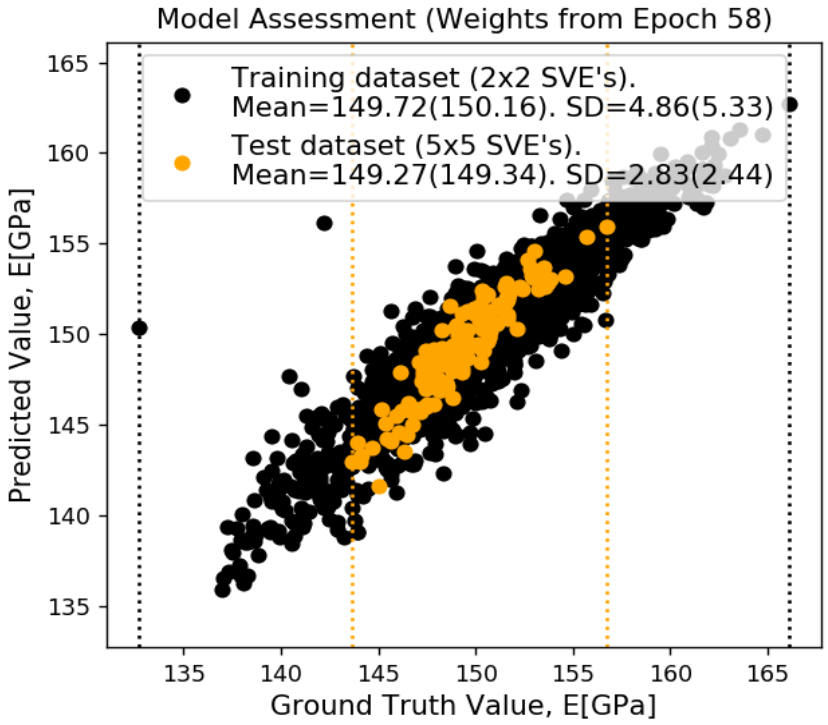
Hypothesis: Artificially augmented images have limitations

Results: Only Original Data

i. Data Splitting



ii. Parity Plot



iii. Scattering around the mean

Standard Deviation Absolute Percentage Error:

8.8% for the training dataset (L/ $\bar{s_g}$ =4)

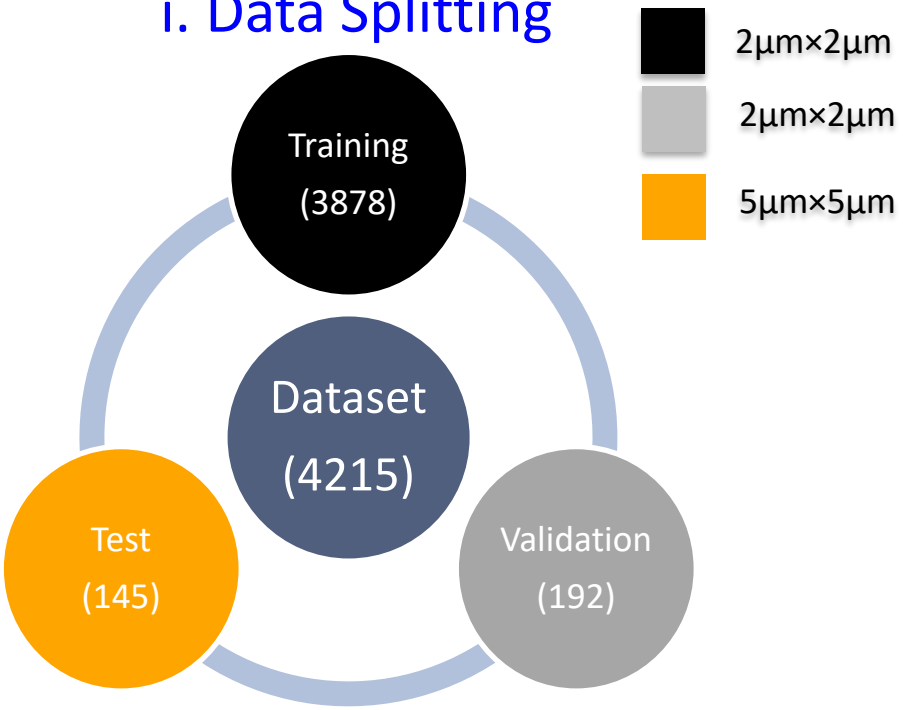
16 % for the test dataset (L/ $\bar{s_g}$ =10)

Significant improvement by adopting original images only

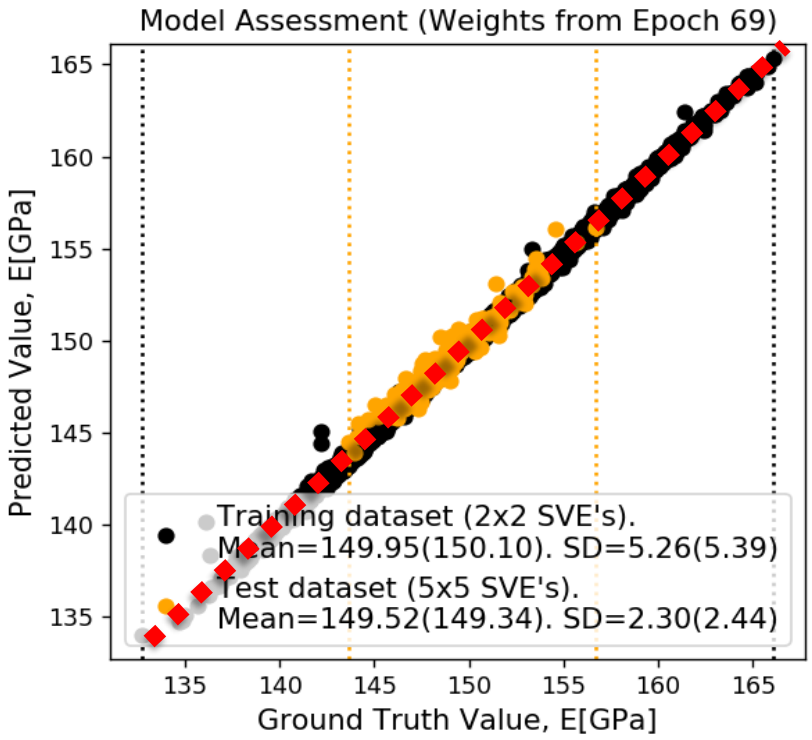
Greater variability of input information (Artificially augmented images feature same value $\bar{E}$ as that of the original image)

Results: Increasing Original Data

i. Data Splitting



ii. Parity Plot



iii. Scattering around the mean

Standard Deviation Absolute Percentage Error:

2.4% for the training dataset (L/ $\overline{s_g}$ =4)

5.7 % for the test dataset (L/ $\overline{s_g}$ =10)

Achieved remarkable generalization capabilities: map the one-to-one correspondence between microstructure-effective property

Hyperparameters of the model

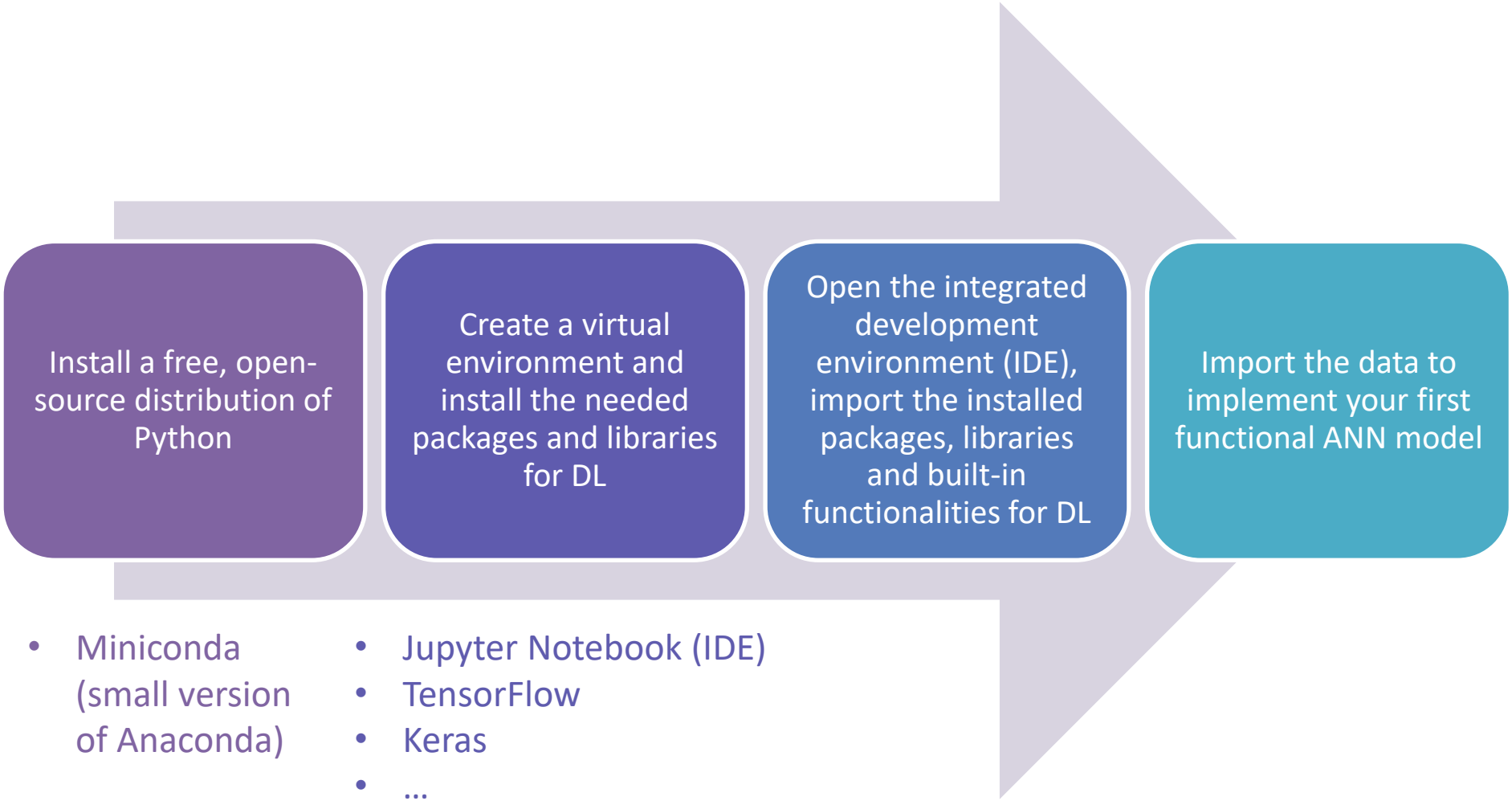
- ✓ Architecture type (MLP, CNN, …)
- ✓ Optimizer selection and related hyperparameters (Learning rate, learning rate decay, …)
- ✓ Use of regularization techniques and associated hyperparameters (patience, dropout rate)
- ✓ Number of nodes in FC layers
- ✓ Number of layers in the regression block
- ✓ Size of input datasets
- ✓ Input data preprocessing (resolution)
- ✓ Batch size
- ✓ …

It is not only finding the best performing **fitting parameters** during the training but also the right combination of **hyperparameters** !

Implementation hands-on

3

Implementation: 4 Steps

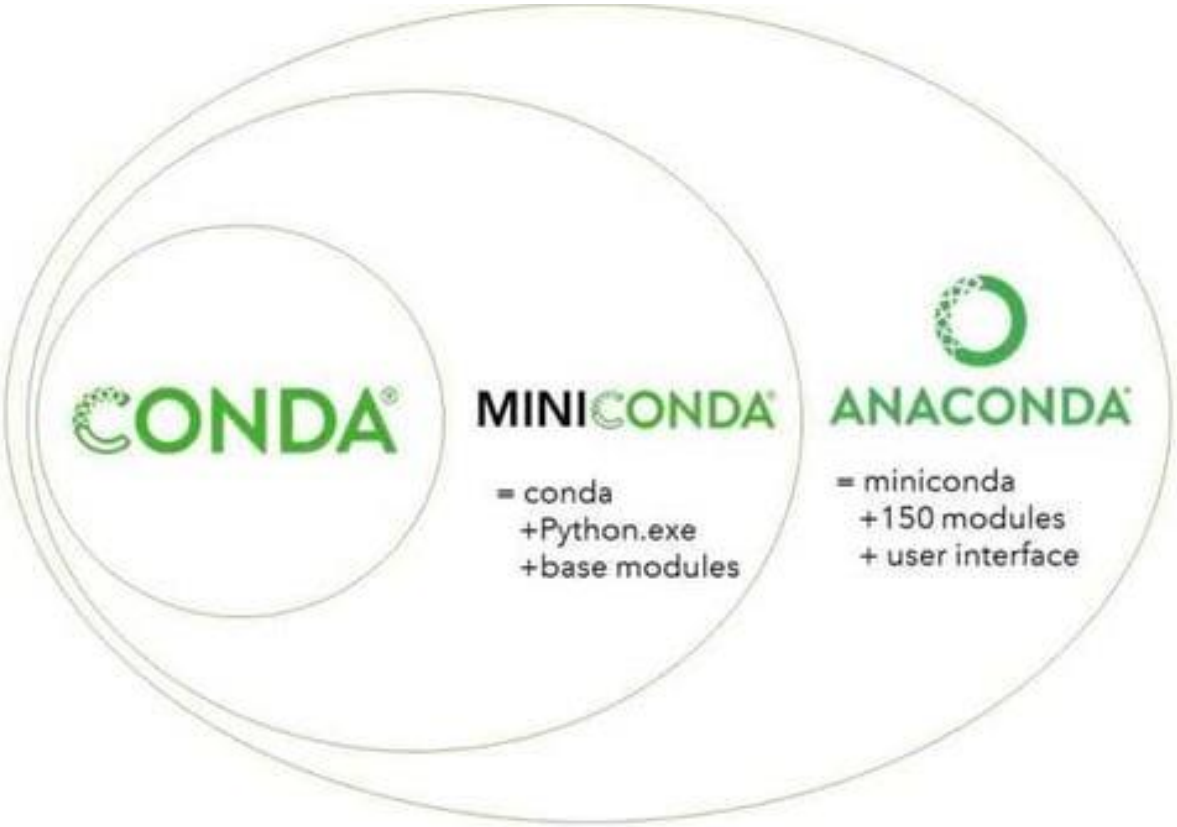


- Miniconda (small version of Anaconda)

- Jupyter Notebook (IDE)
- TensorFlow
- Keras
- …

Instructions

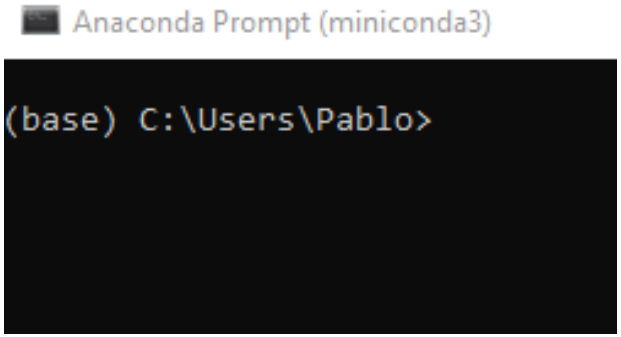
1. Install Miniconda https://docs.conda.io/en/latest/miniconda.html



Anaconda (~3 GB) https://www.anaconda.com/products/distribution

Instructions

2. Open Anaconda Prompt



3. Create a virtual environment called “MyEnv” with latest Python version running the following command:

```
conda create --name MyEnv
```

4. Activate the created virtual environment running the following command:

```
conda activate MyEnv
```

Instructions

5. Install Jupyter Notebook, open-source web application IDE that allows to develop code dynamically into cells

```
conda install jupyter notebook
```

open-source
Web app IDE

6. Install TensorFlow latest version running computations on CPU. Run:

```
conda install tensorflow
```

Open-source Python library for ML

High-level DL API

*CPU usage during running can be checked in Task Manager>Performance

Instructions

7. Install additional useful libraries


```
pip install matplotlib
pip install -U scikit-learn
pip install pandas
pip install opencv-python
```

- Additional libraries might need to be installed similarly in the virtual environment depending on the specific code application!

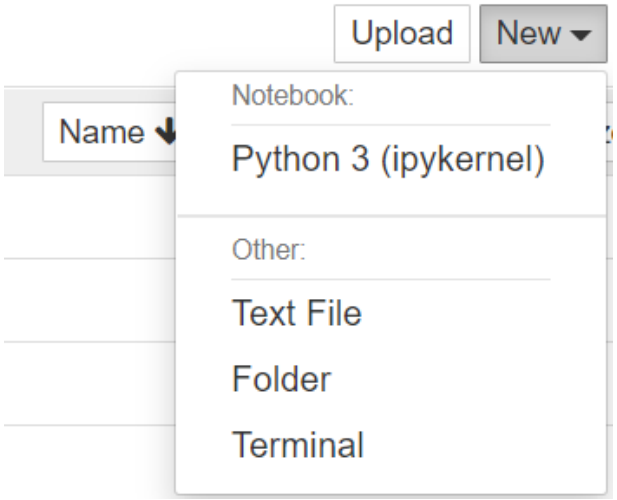
Instructions

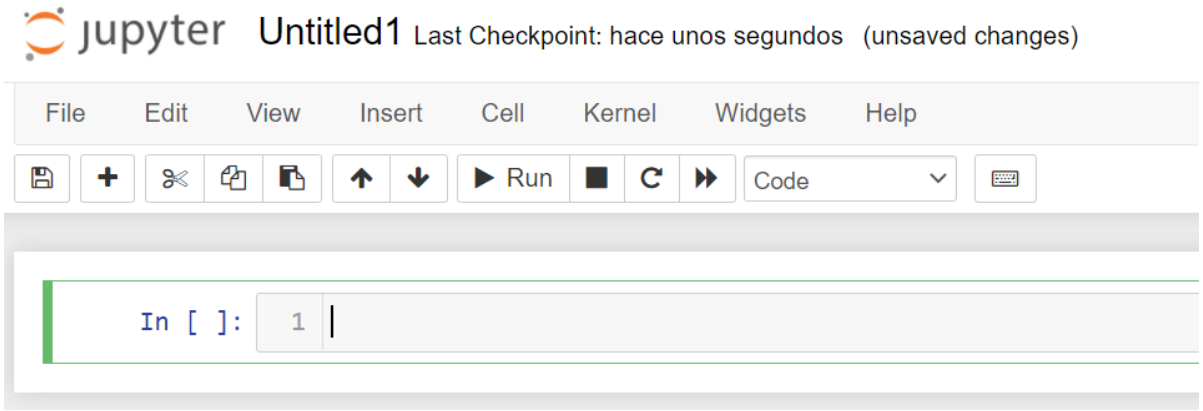
8. To check the installation process open Jupyter Notebook typing :

```
jupyter notebook
```

```
(MyEnv) C:\Users\Pablo>jupyter notebook
```

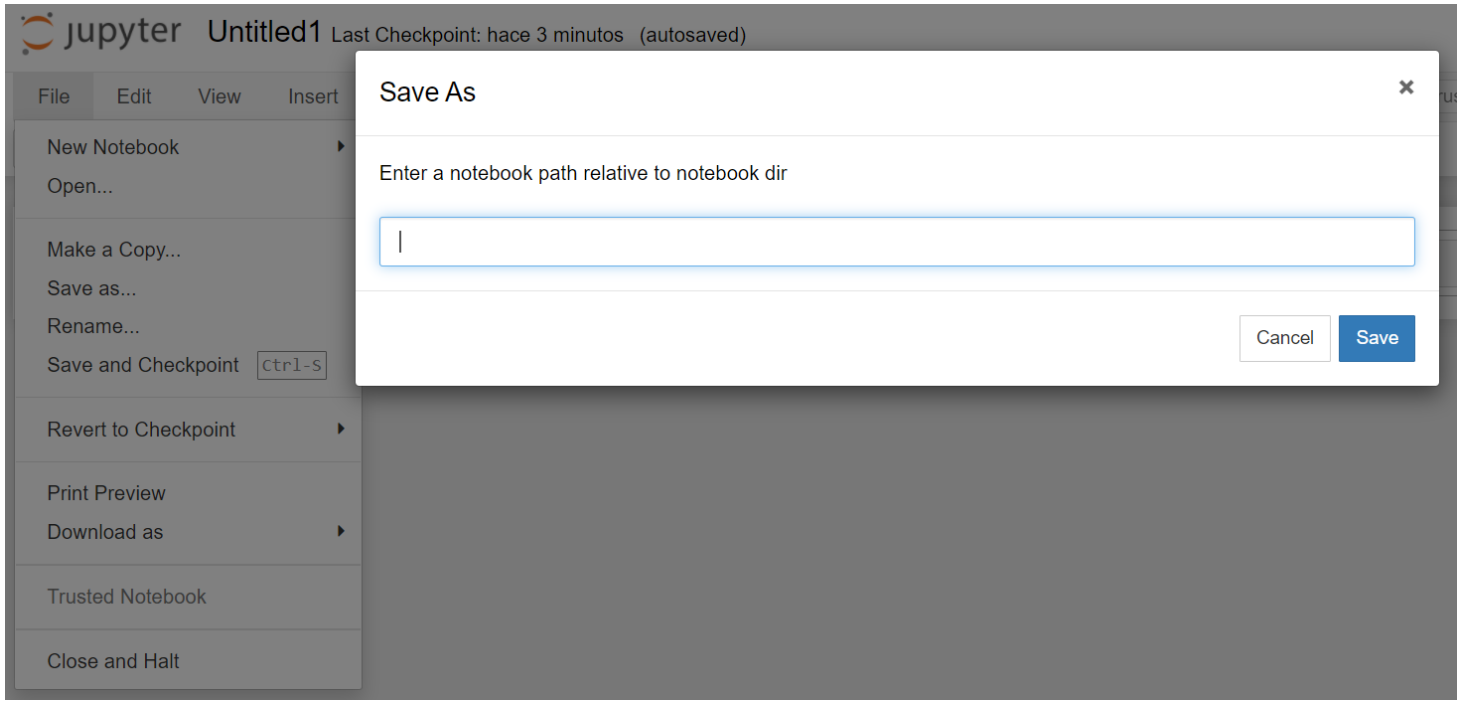
9. Create a new Jupyter Notebook clicking on **New> Python 3** :





Instructions

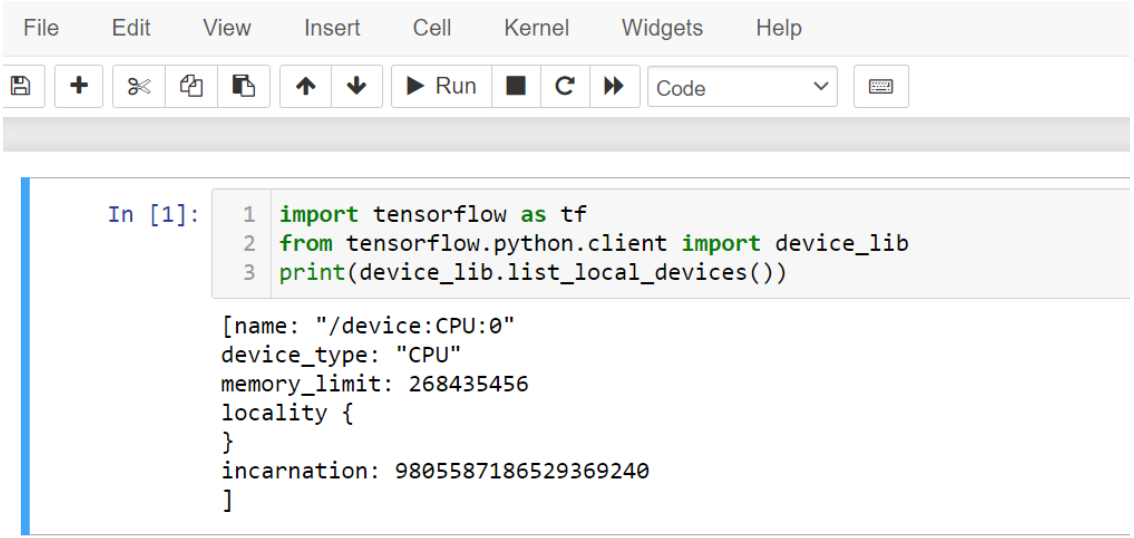
11. Save the notebook by going to **File>Save as**. The default working directory of Jupyter is **C:\Users\<username>**



Instructions

12. Copy and paste the next lines of code in the first cell and run it by clicking on the cell and then pressing **Ctrl+Enter**

```
import tensorflow as tf
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

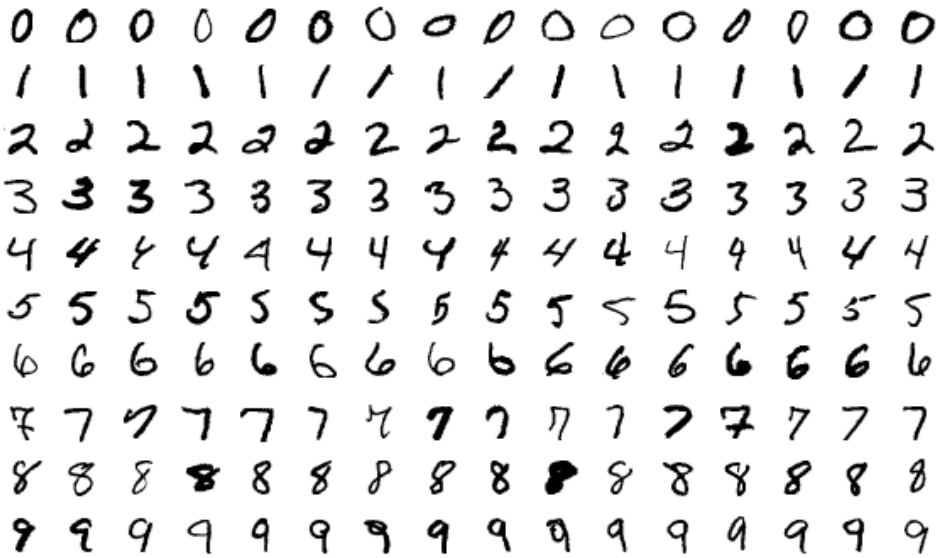


Code Example from Keras API: Classification

We will implement the next code example available at:

https://keras.io/examples/vision/mnist_convnet/

Description: A simple convnet that achieves ~99% test accuracy on MNIST dataset, a database of handwritten digits with a training set of 60000 examples, and a test set of 10000 examples.



Additional Online Resources

Keras https://keras.io/api/

You can use Google Colab (https://colab.research.google.com), a Jupyter Notebook stored in Google Drive which gives access to GPUs without any required set up in your own machine.
Watch this video https://www.youtube.com/watch?v=inN8seMm7UI

Website by Jason Brownlee PhD https://machinelearningmastery.com/

NYU

NYU Deep Learning lectures by Yann LeCun (founding father of convolutional nets)
https://www.youtube.com/c/AlfredoCanziani/featured


REVIEW article

Front. Mater., 15 May 2019
Sec. Computational Materials Science
https://doi.org/10.3389/fmats.2019.00110

This article is part of the Research Topic
Machine Learning and Data Mining in Materials Science
View all 16 Articles

A Review of the Application of Machine Learning and Data Mining Approaches in Continuum Materials Mechanics

Frederic E. Bock[1*], Roland C. Aydin[1], Christian J. Cyron[1,2], Norbert Huber[1,3], Surya R. Kalidindi[4] and Benjamin Klusemann[1,5]

Review article of several ML applications in the field of Materials Science
https://www.frontiersin.org/articles/10.3389/fmats.2019.00110/full

"Godfathers of AI" and "Godfathers of Deep Learning"

2018 Turing Award Won by 3 Pioneers in Artificial Intelligence

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

https://www.nature.com/articles/nature14539