

Deep Forward Networks - Optimization

Thursday
08h00-09h00

Machine Learning Definition

« A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . »

Tom M. Mitchell (1997)



how well the algorithm
performs on the “walking” task

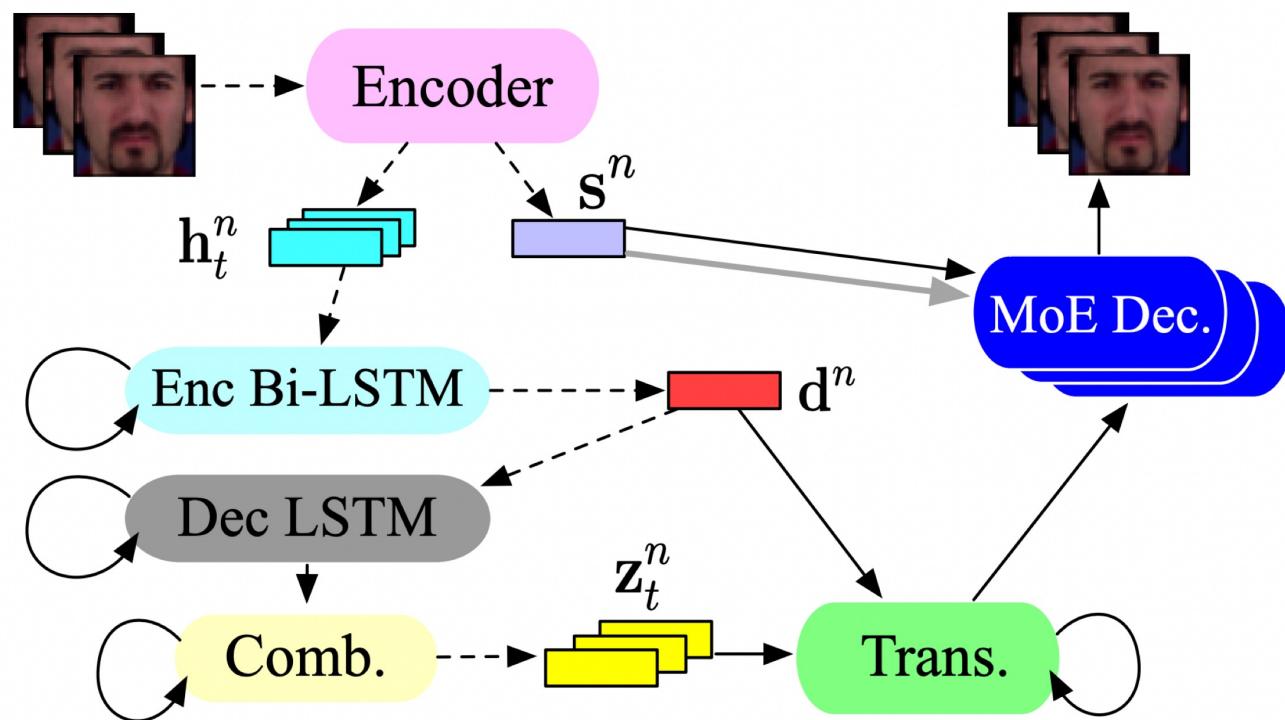
Performance Measure P

Estimate the ML algorithm performance on task T using the validation set

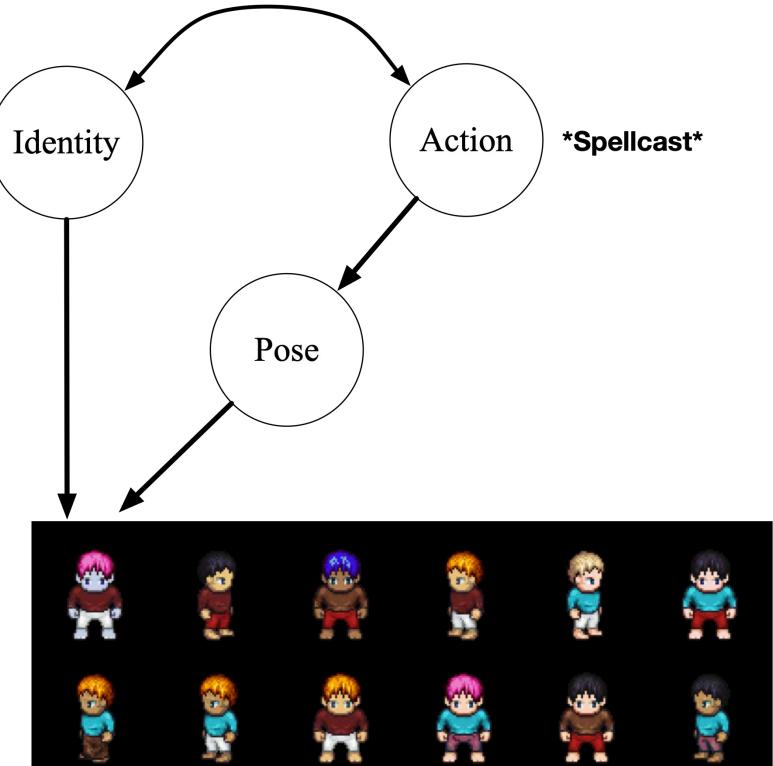
Introduction

- To evaluate a ML algorithm, we need a way to measure **how well it performs on the task**
- It is measured **on a separate set** (test set) from what we use to build the function f (training set)
- **Examples :**
 - Classification accuracy (portion of correct answers)
 - Error rate (portion of incorrect answers)
 - Regression accuracy (e.g. least squares errors)

Inference

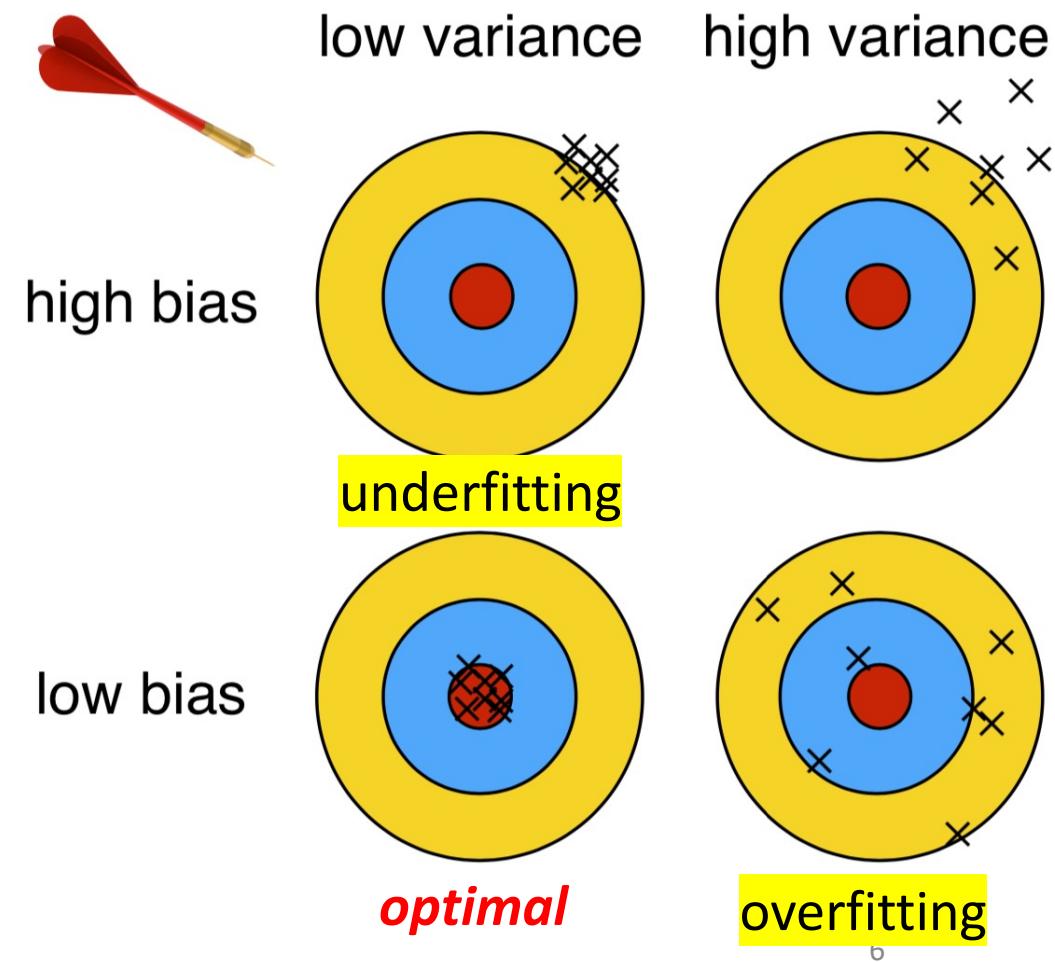
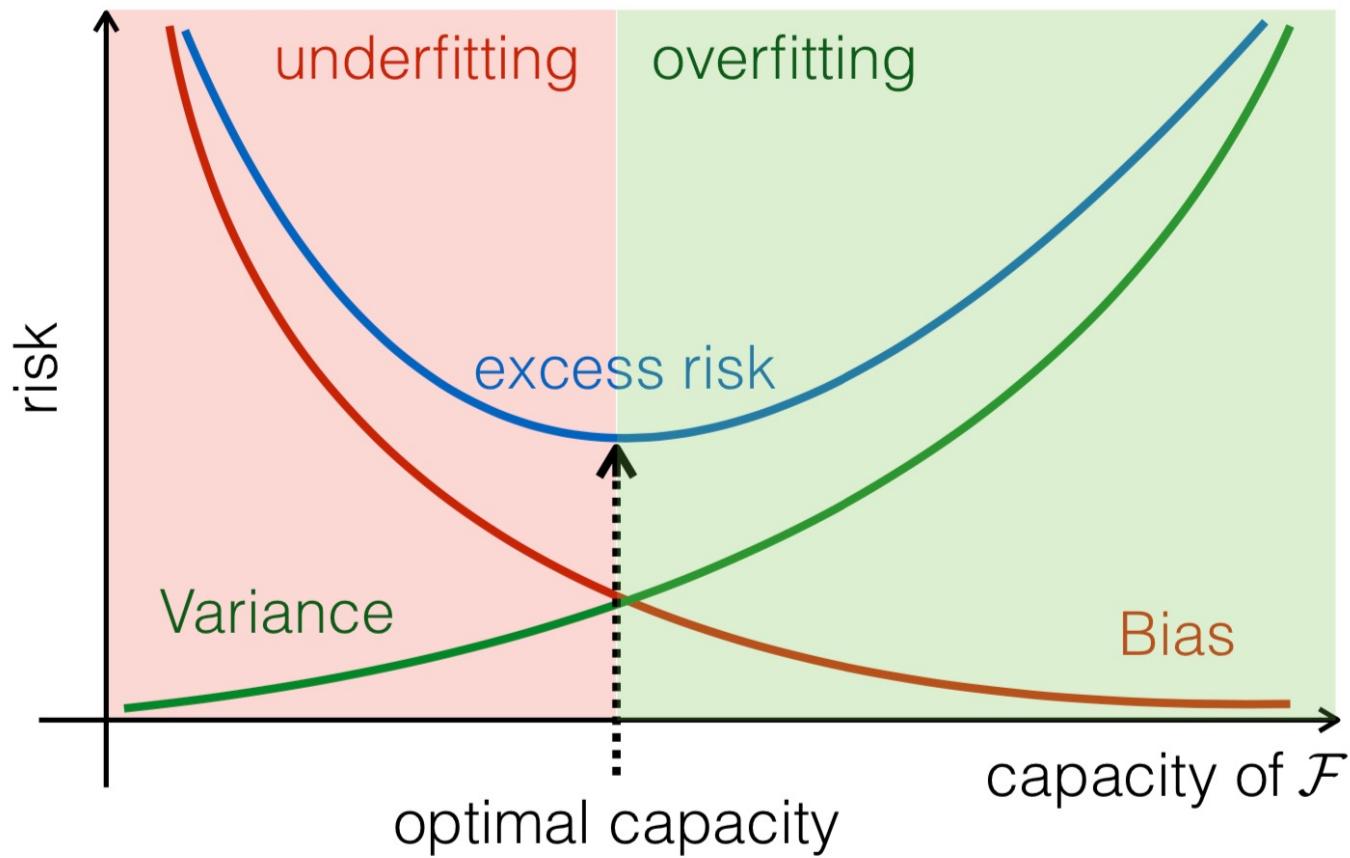


Pink Hair
Brown Top
White Trousers
Blue Skin



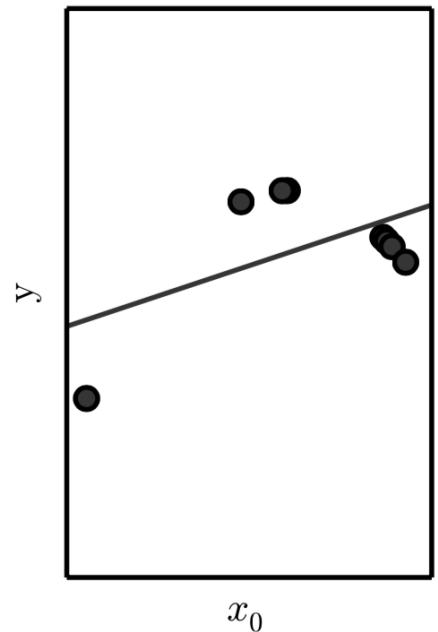
Vowels, M.J., Camgoz, N.C. and Bowden, R., 2021. VDSM: Unsupervised Video Disentanglement with State-Space Modeling and Deep Mixtures of Experts. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8176-8186).

Bias and Variance - Overfitting and Underfitting



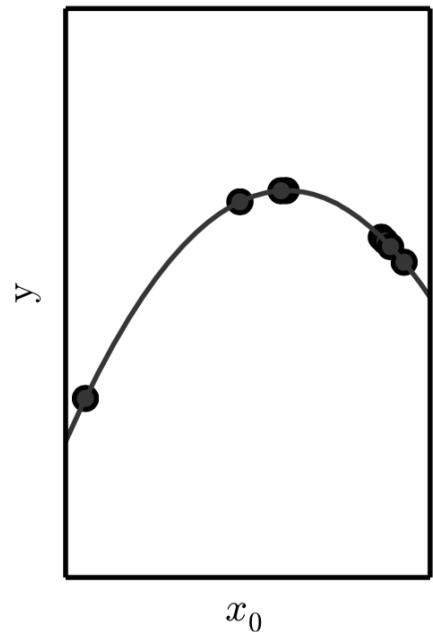
Overfitting and Underfitting

Underfitting

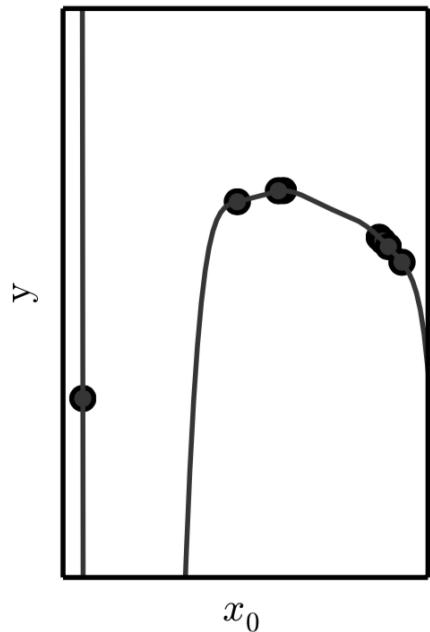


High bias

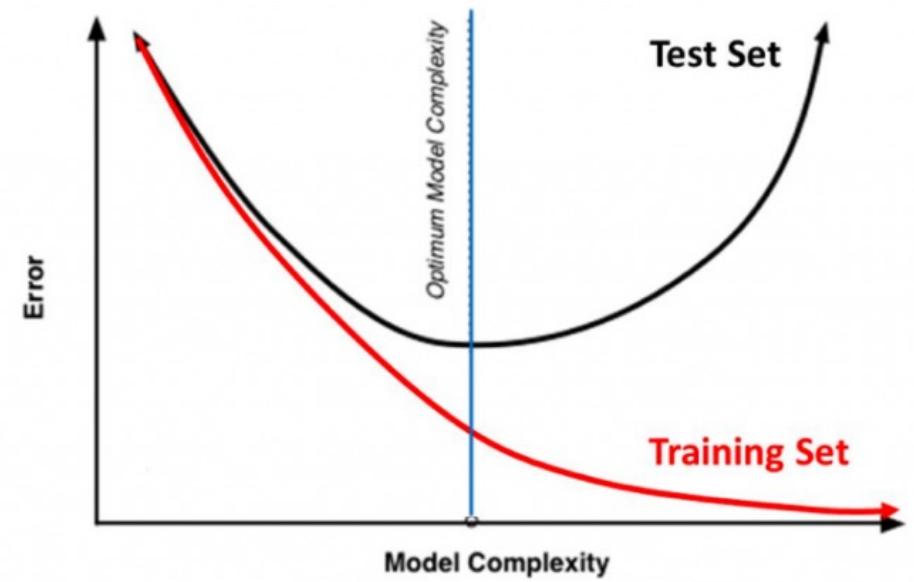
Appropriate capacity



Overfitting



High variance



Optimization

In terms of performance and time



1. Performance

- Bias reduction techniques
- Variance reduction techniques



Case

- You want to find cats in images
- Classification error (portion of wrong answers) used as an evaluation metric



Algorithm	Classification error (%)
A	3%
B	5%

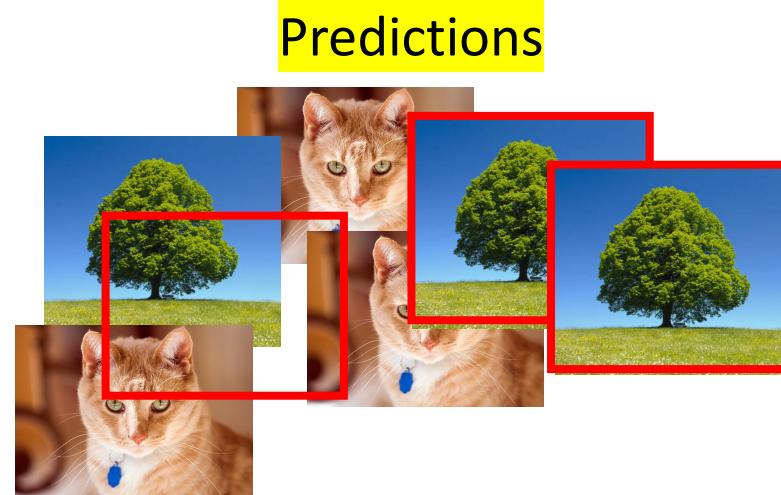
➤ *Which one is best ?*

but also check at which class / job the classifier performs well

Evaluation metrics



Ground Truth



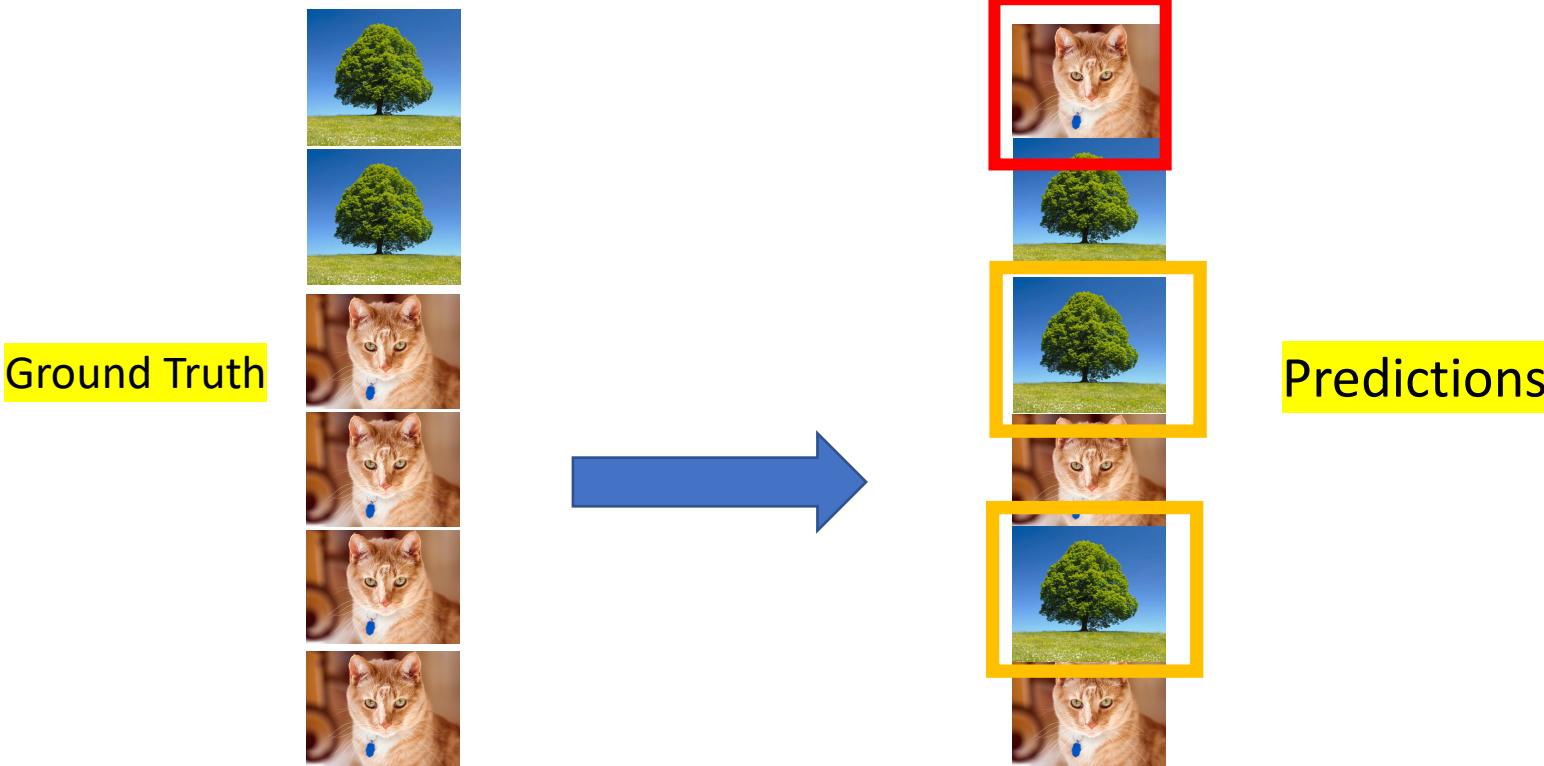
Predictions

- **Precision (p)** $\text{Precision (\%)} = \frac{\text{True positive}}{\text{Number of predicted positive}} \times 100 = \frac{\text{True positive}}{(\text{True positive} + \text{False positive})} \times 100$

$$\frac{2}{2+1} \times 100 = 66\%$$

- **Recall (r)** $\text{Recall (\%)} = \frac{\text{True positive}}{\text{Number of predicted actually positive}} \times 100 = \frac{\text{True positive}}{(\text{True positive} + \text{False negative})} \times 100$

$$\frac{2}{2+2} \times 100 = 50\%$$



$$\text{Precision (\%)} = \frac{\text{True positive}}{\text{Number of predicted positive}} \times 100 = \frac{\text{True positive}}{(\text{True positive} + \text{False positive})} \times 100$$

$$\frac{2}{2+1} \times 100 = 66\%$$

$$\text{Recall (\%)} = \frac{\text{True positive}}{\text{Number of predicted actually positive}} \times 100 = \frac{\text{True positive}}{(\text{True positive} + \text{False negative})} \times 100$$

$$\frac{2}{2+2} \times 100 = 50\%$$

See also Sensitivity (same as recall) and Specificity

Sources: [6][7][8][9][10][11][12][13][14] [view](#) · [talk](#) · [edit](#)

		Predicted condition		
		Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= \text{TPR} + \text{TNR} - 1$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{\text{TP}}{\text{P}} = 1 - \text{FNR}$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{\text{FP}}{\text{N}} = 1 - \text{TNR}$
Prevalence $= \frac{\text{P}}{\text{P} + \text{N}}$	Positive predictive value (PPV), precision $= \frac{\text{TP}}{\text{PP}} = 1 - \text{FDR}$	False omission rate (FOR) $= \frac{\text{FN}}{\text{PN}} = 1 - \text{NPV}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Negative likelihood ratio (LR-) $= \frac{\text{FNR}}{\text{TNR}}$
Accuracy (ACC) $= \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$	False discovery rate (FDR) $= \frac{\text{FP}}{\text{PP}} = 1 - \text{PPV}$	Negative predictive value (NPV) $= \frac{\text{TN}}{\text{PN}} = 1 - \text{FOR}$	Markedness (MK), deltaP (Δp) $= \text{PPV} + \text{NPV} - 1$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR-}}$
Balanced accuracy (BA) $= \frac{\text{TPR} + \text{TNR}}{2}$	F_1 score $= \frac{2\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$	Fowlkes–Mallows index (FM) $= \sqrt{\text{PPV} \times \text{TPR}}$	Matthews correlation coefficient (MCC) $= \sqrt{\text{TPR} \times \text{TNR} \times \text{PPV} \times \text{NPV}} - \sqrt{\text{FNR} \times \text{FPR} \times \text{FOR} \times \text{FDR}}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}$

Source: wiki (Precision and Recall)

F1-score

- F1-score is a harmonic mean combining p and r

$$\text{F1-Score} = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

	<u>Precision</u>	<u>Recall</u>	<u>F1 Score</u>
Algo 1 →	0.5	0.4	0.444 ✓
Algo 2 →	0.7	0.1	0.175
Algo 3 →	0.02	1.0	0.0392

- See also balanced accuracy (average recall)

First of all, understand your data !

- Carry out manual **error analysis**
 - Look at *mislabeled development set* examples (*do not look at test set*)
 - For example : check by hand 500 pictures (incorrect labels ? Foggy pictures ? Other causes ?)
- Clean up **incorrectly labeled** data
 - Apply same process to your dev and test sets !



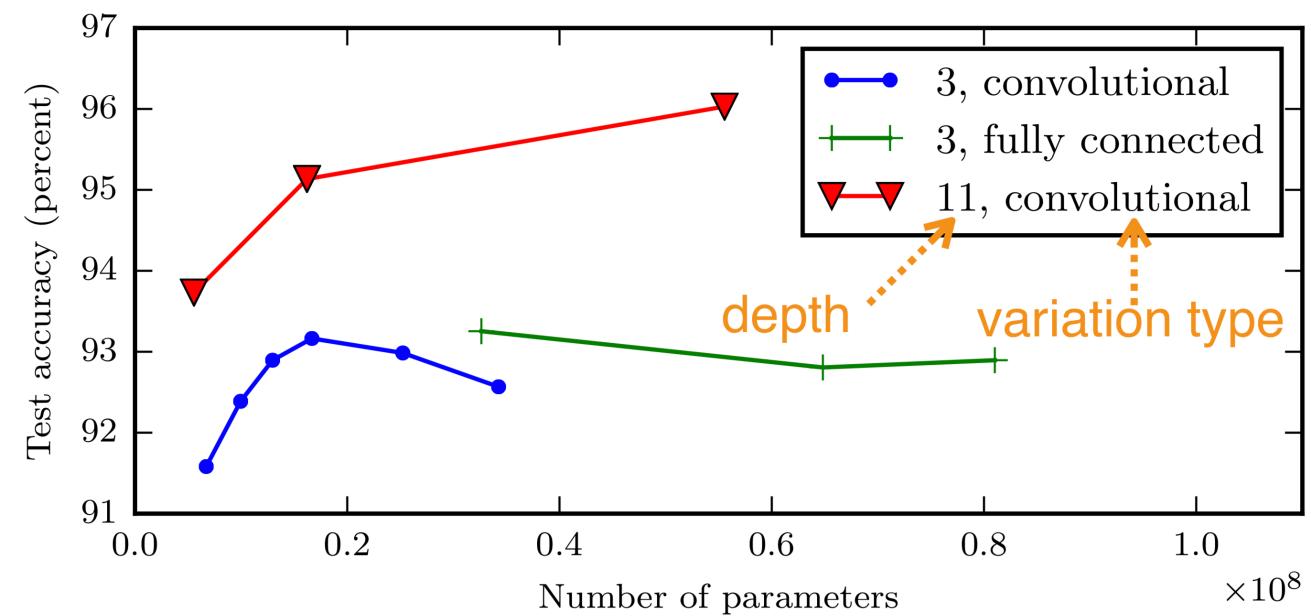
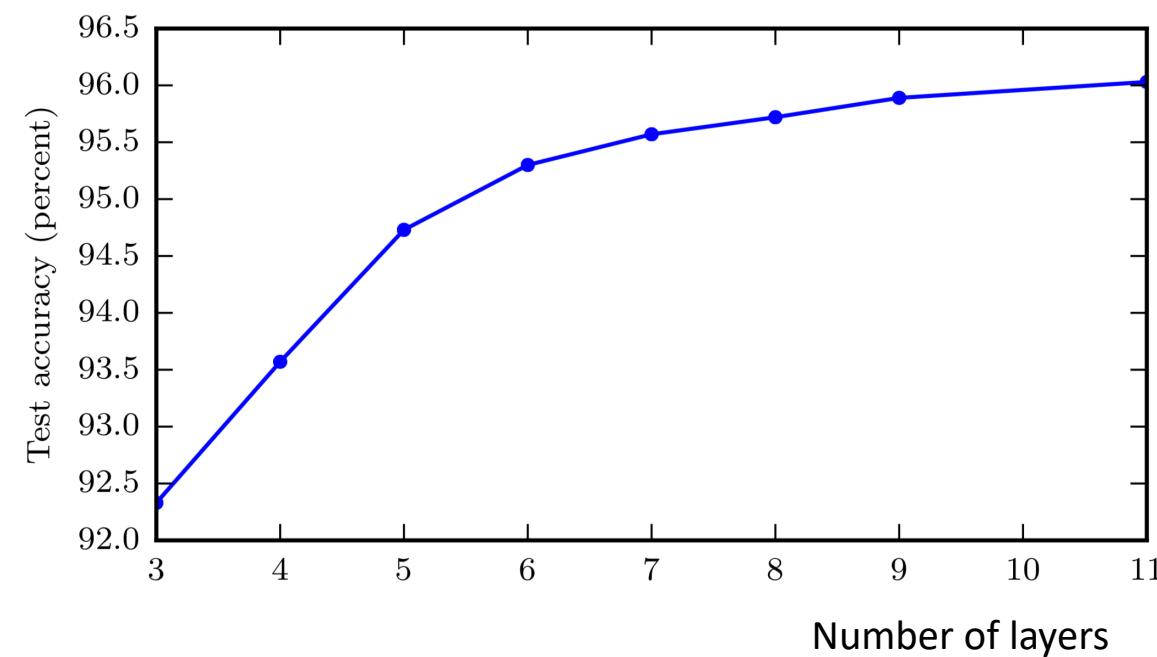


Bias Reduction techniques

- Hyperparameter tuning
- Model tuning
- Optimization algorithm

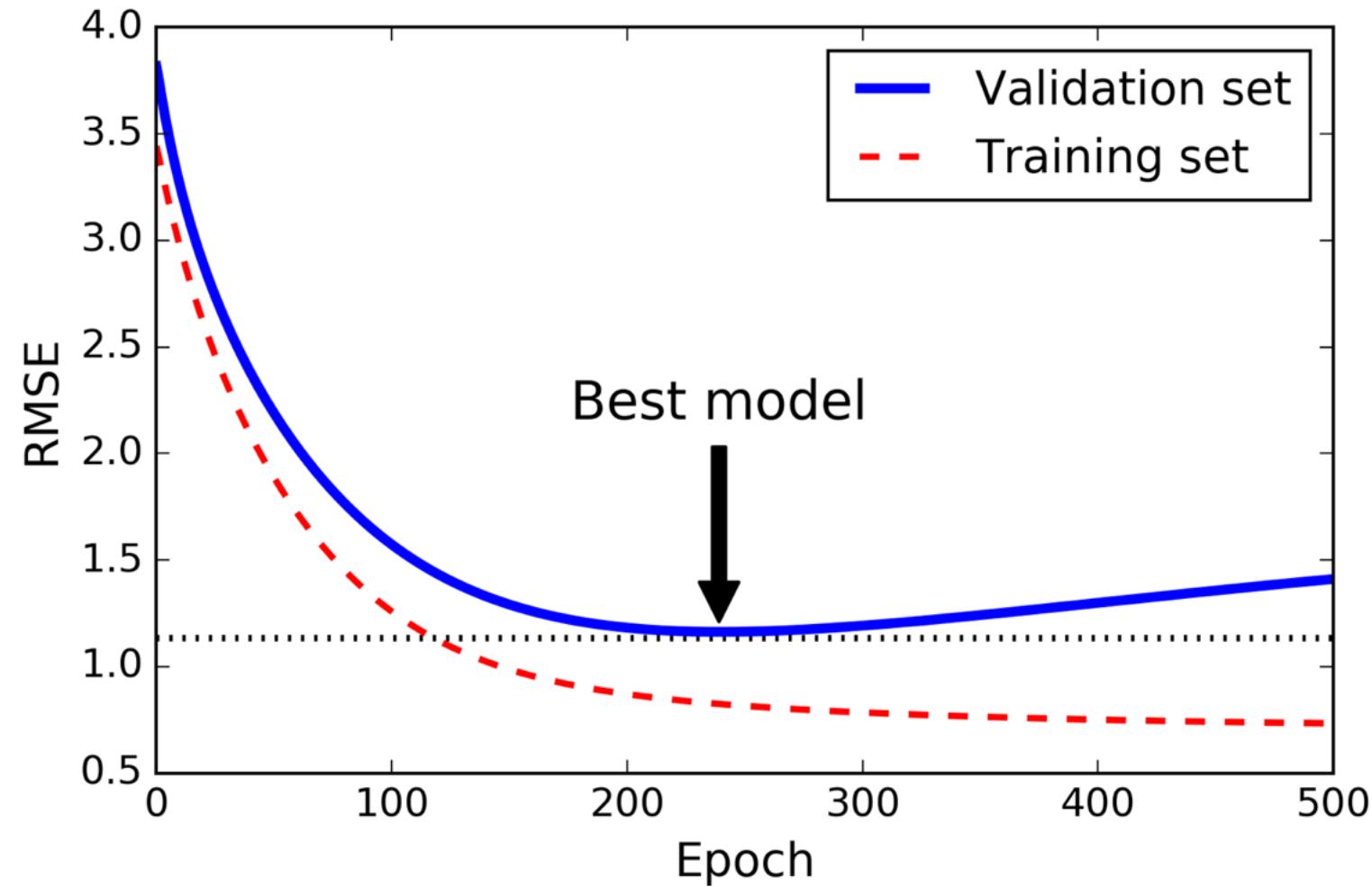
Hyperparameters : number of hidden layers/units

- To go **deeper** helps generalization (but depends on application)
- *better to have many simple layers than few highly complex ones*



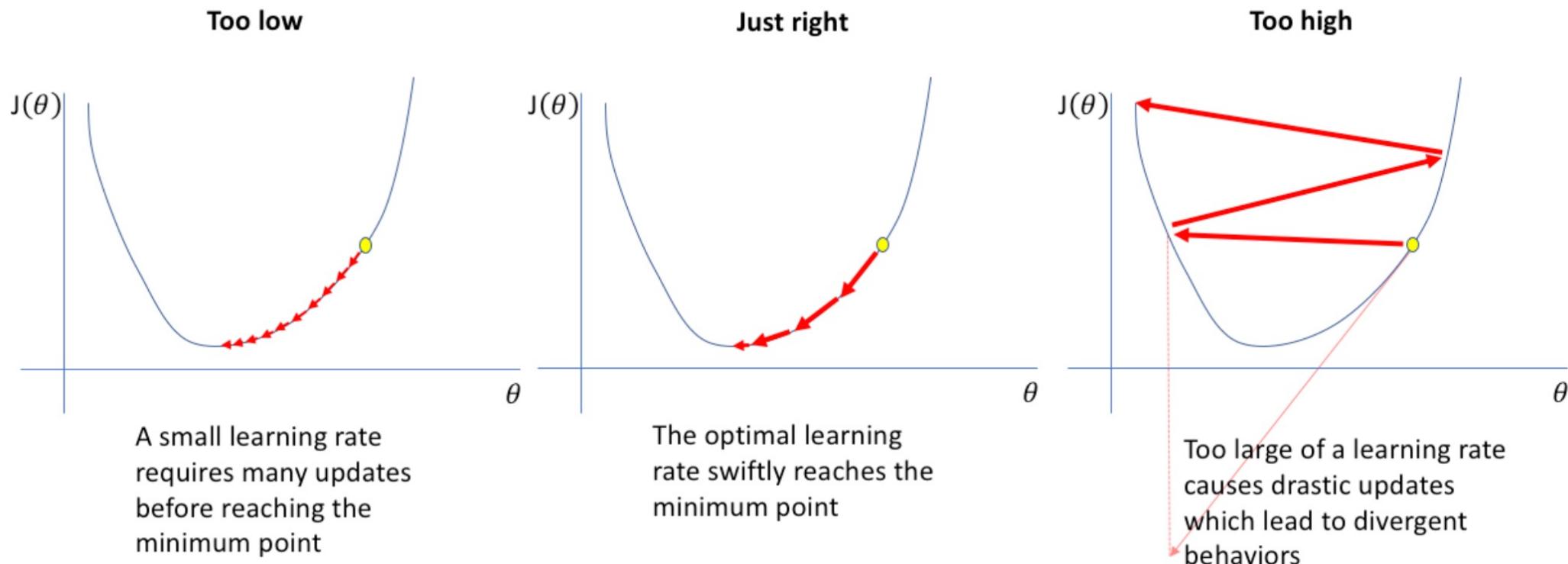
Hyperparameters : epochs

- Train *longer*



Hyperparameters : learning rate α

- Has a significant impact on the model performance, while being **one of the most difficult parameters** to set

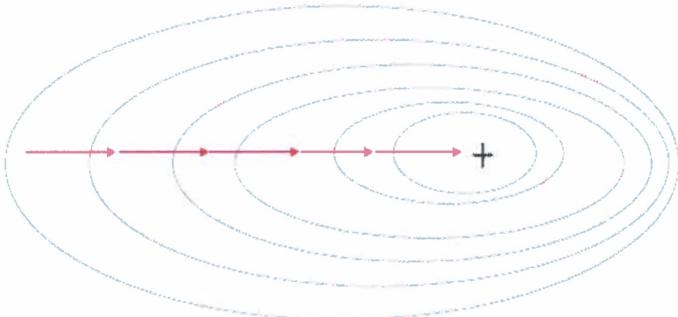


- We can also design a scheduler for the learning rate

Hyperparameters : batch size

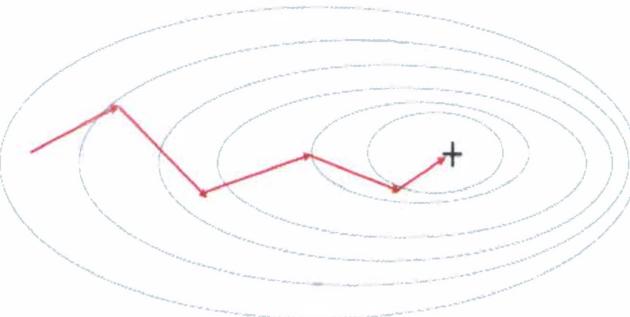
- At each iteration :

Gradient descent (GD)



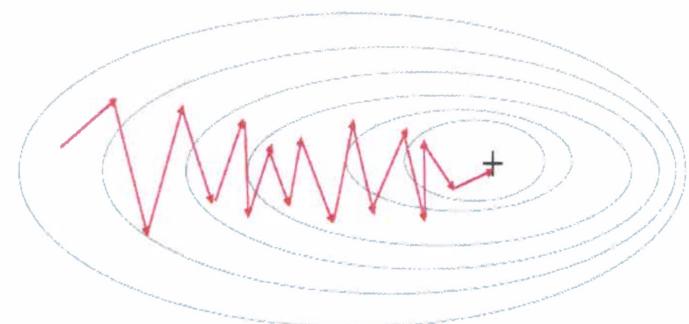
the *whole* training set

Mini-batch gradient
descent



a *batch* of samples

Stochastic gradient
descent (SGD)

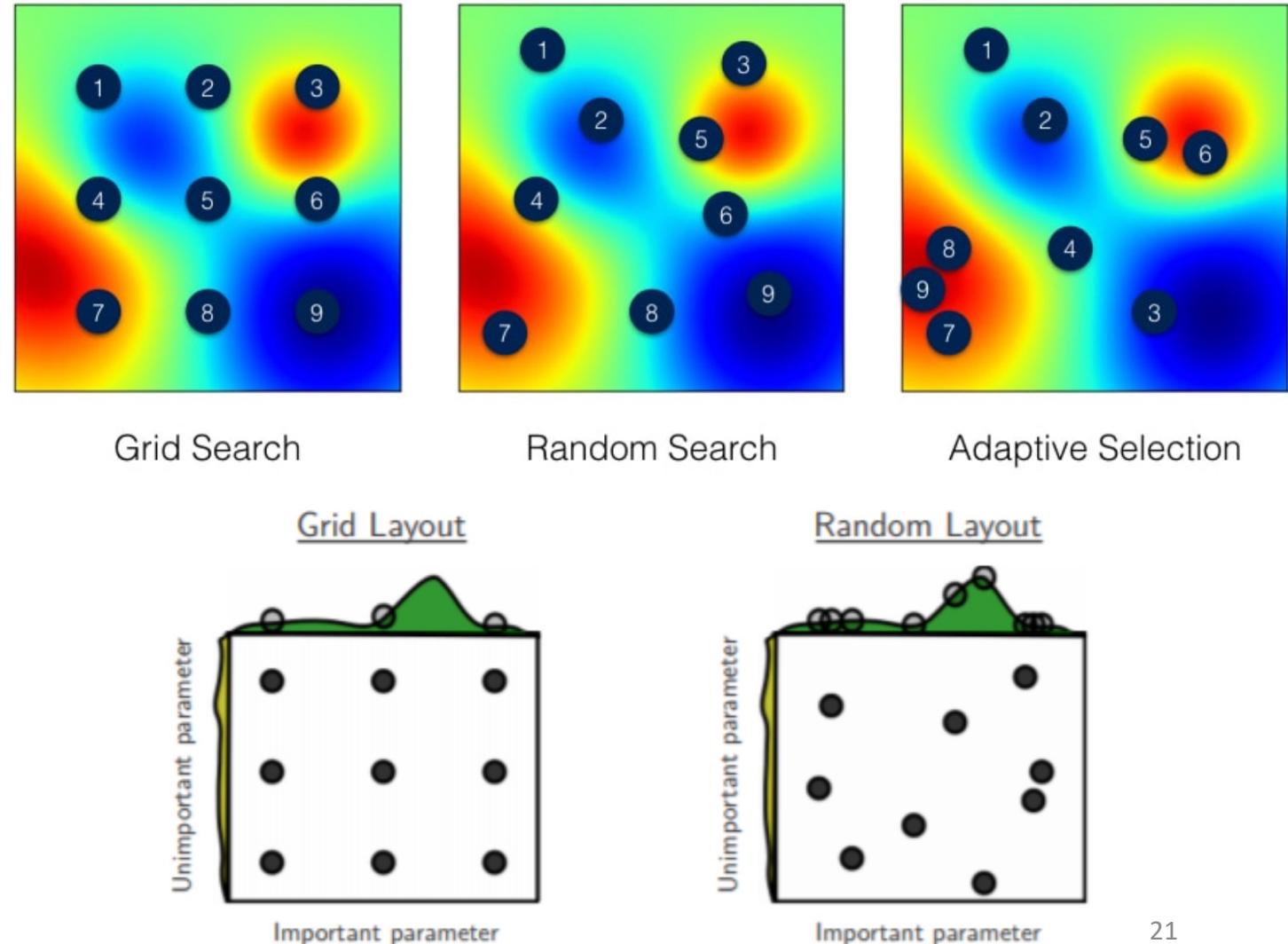


1 sample

- Batch size choice *typically 32,64,128,256,512*

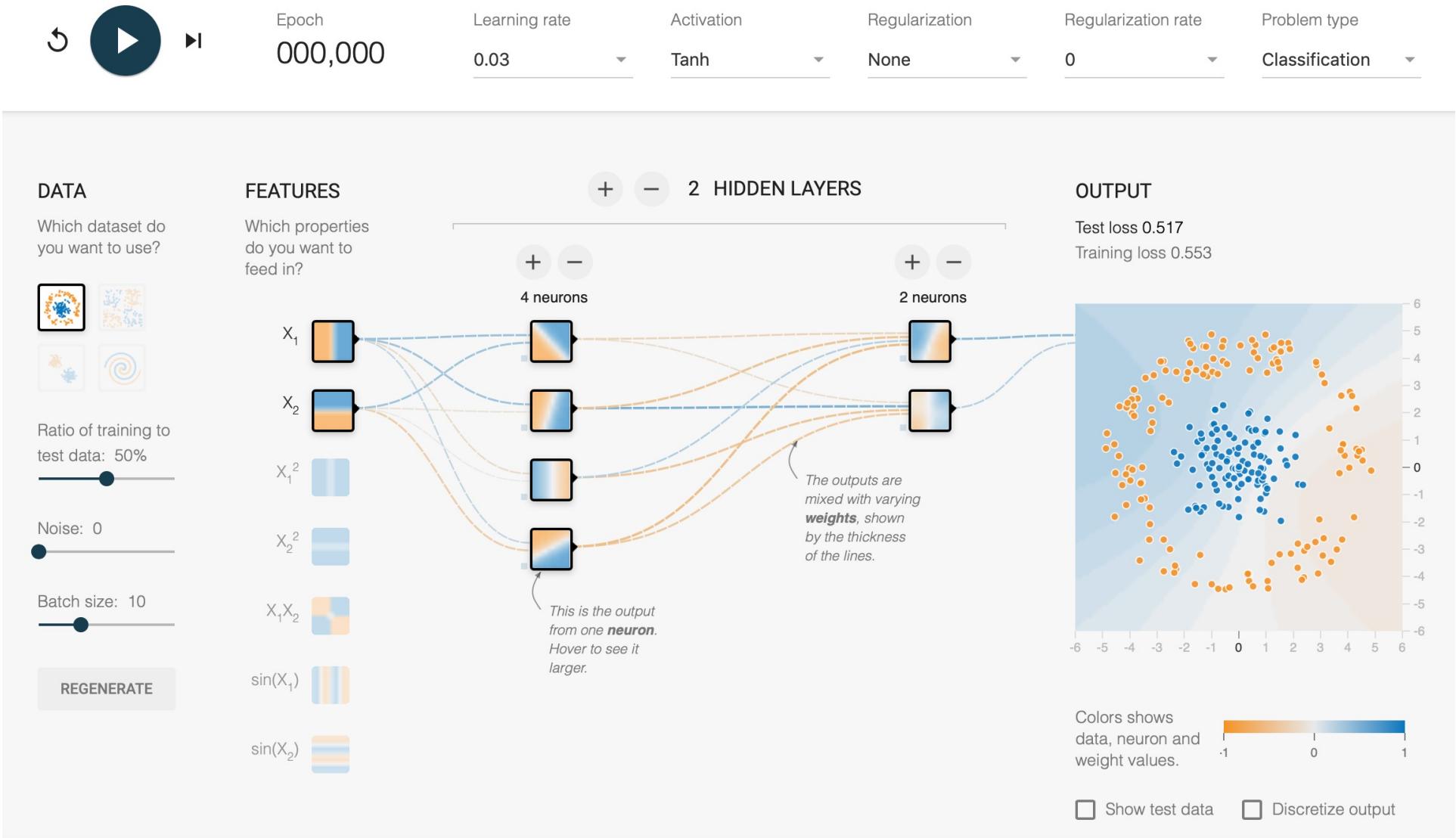
Hyperparameters : Global Search

- List :
 - α (**0.0001 – 1**)
 - number of hidden layers
 - number of hidden units
 - learning rate decay
 - mini-batch size
 - ...
- Advice is to use **random values**



Hyperparameters : Global Search

Demo



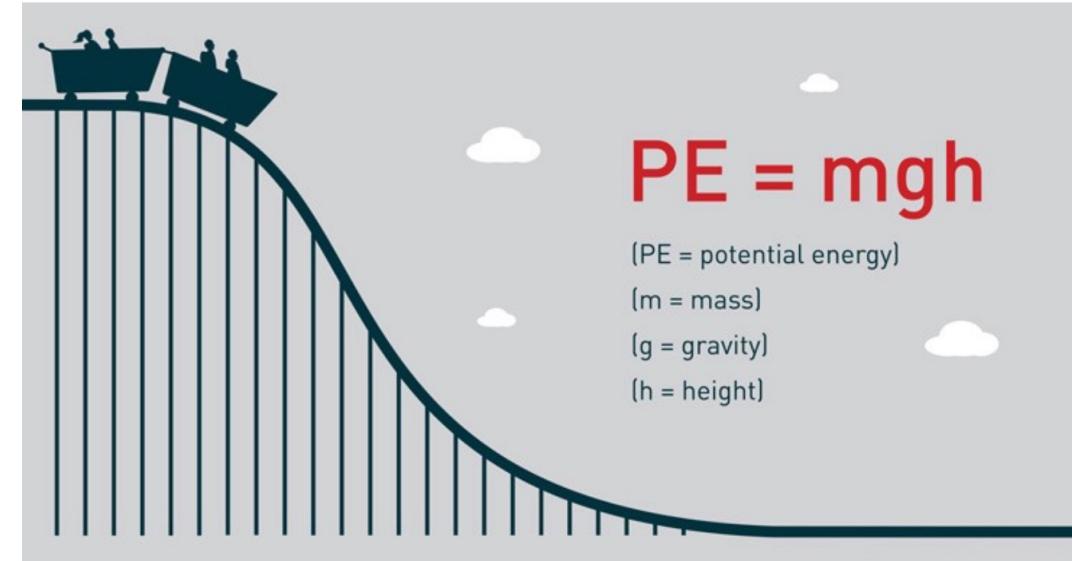
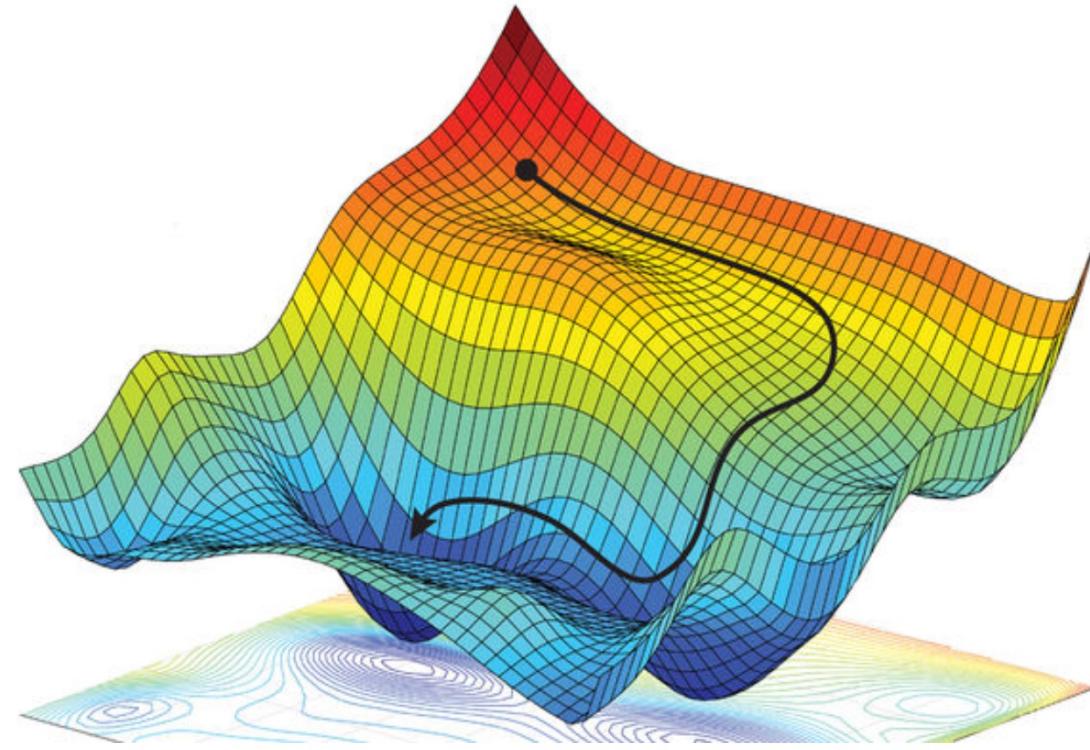
Model : Weight initialization

- The initial parameters need to **break the symmetry** between different units
- Use ***random weights*** from a Gaussian or Uniform distribution. Alternatively, use ***Xavier weights***
- Another strategy is to initialize weights by **transferring weights** learnt via an unsupervised learning method (method also called **fine-tuning**)

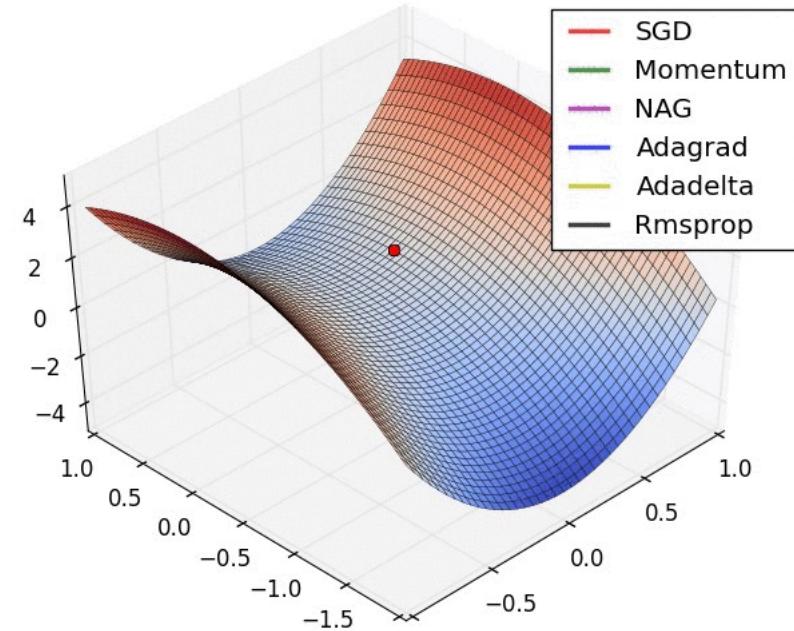
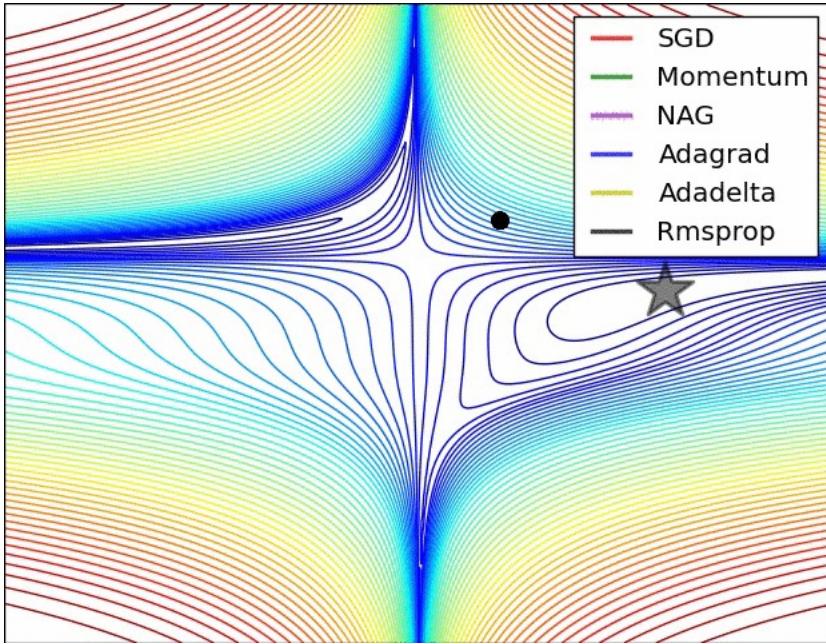


Optimization algorithm

- No consensus on what algorithm performs best
- Most popular choices :
 - SGD (mini-batch gradient descent)
 - SGD + Momentum
 - RMSProp
 - RMSProp + Momentum
 - Adam popular choice
- Strategy : *pick one and get familiar* with the tuning



Gradient Descent Variations



<https://ruder.io/optimizing-gradient-descent/>



Variance Reduction techniques

- Bigger training set
- Regularization

Regularization

- Different **strategies** :
 - **Dataset** (division, augmentation,...)
 - **Model** (dropout, L2-, ...)
 - **Training** (early stopping)
- **Use cases** : if few data or if model has more than 50 layers (CNN)

Regularization (*Dataset*) : Division

- Divide the data into a **training**, **validation** and **test** sets
 - **Training set** to define the optimal predictor
 - **Validation set** to choose the capacity
 - **Test set** to evaluate the performance

as much training data as possible, but we need enough data to test it on



Regularization (*Dataset*) : Augmentation

- Apply **realistic transformations** to data to create new synthetic samples, with same label



original



affine
distortion



noise



elastic
deformation

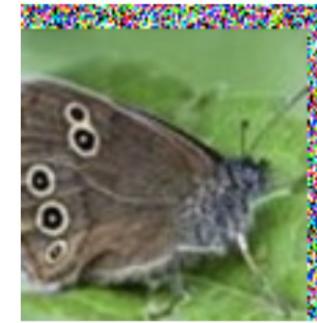


- Process also called **jittering**

horizontal
flip



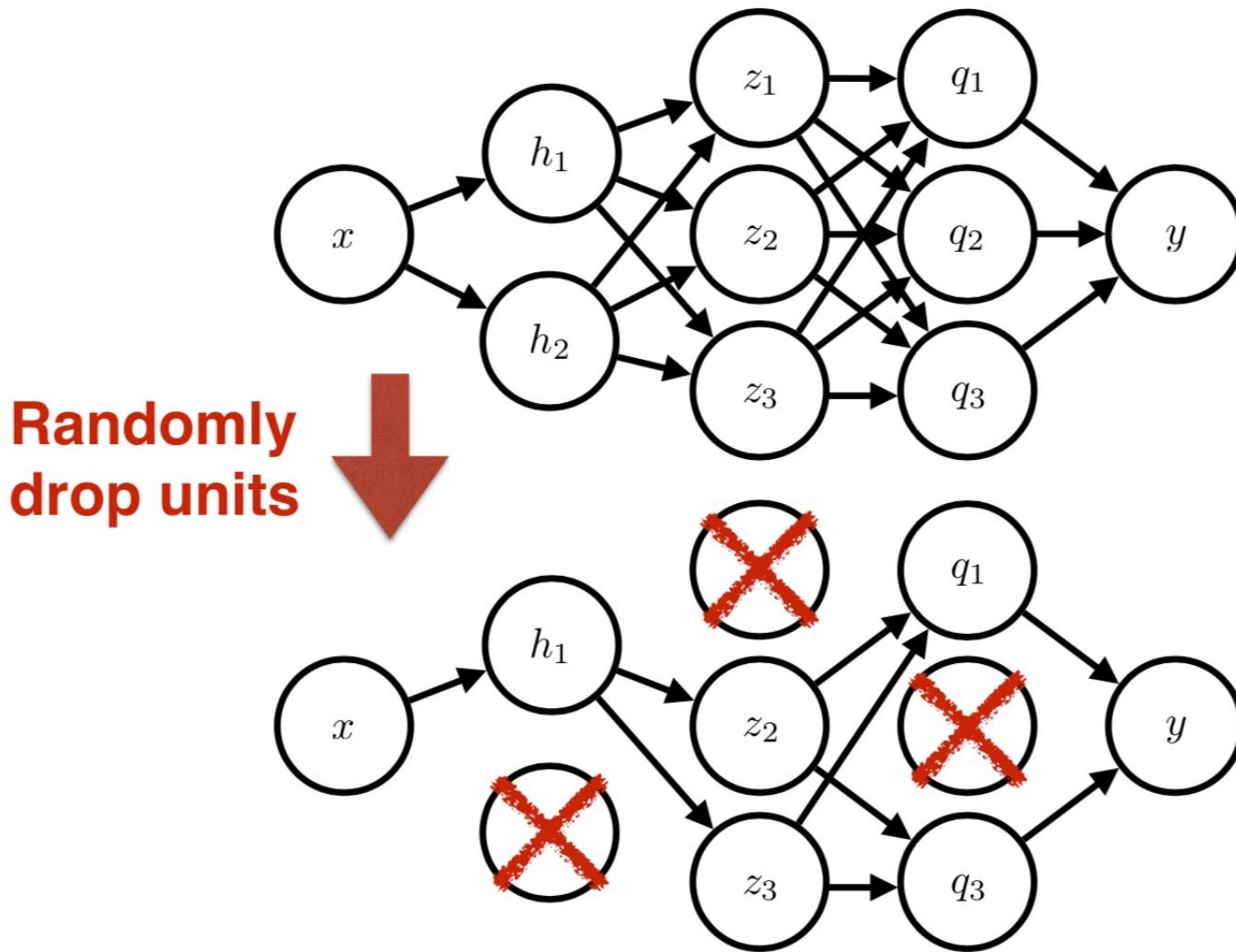
random
translation



hue shift



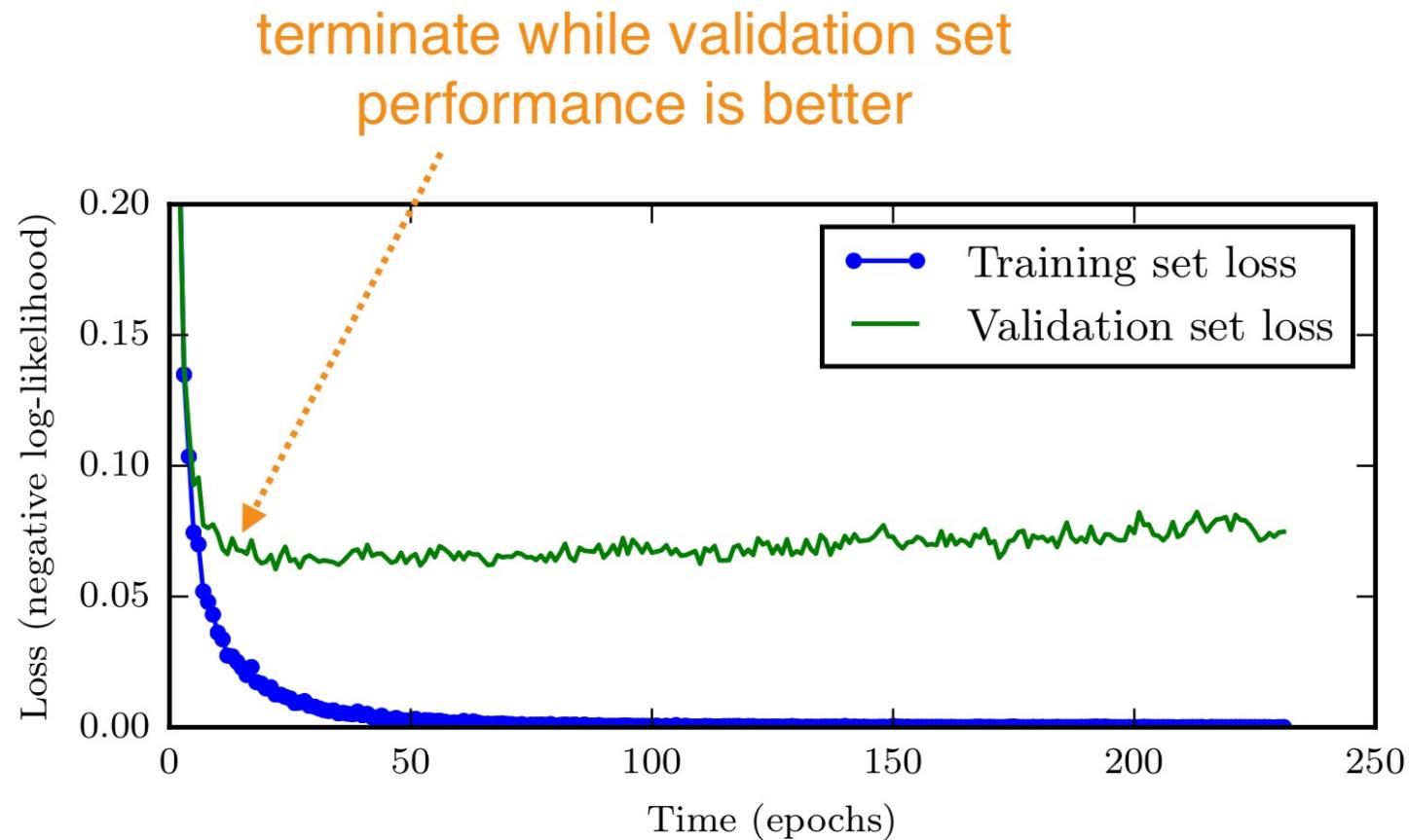
Regularization (*Model*) : Dropout



- Apply it **both** in forward and backward propagations
- **BUT** use it only in the *training phase* !

Regularization (*Training*) : Early stopping

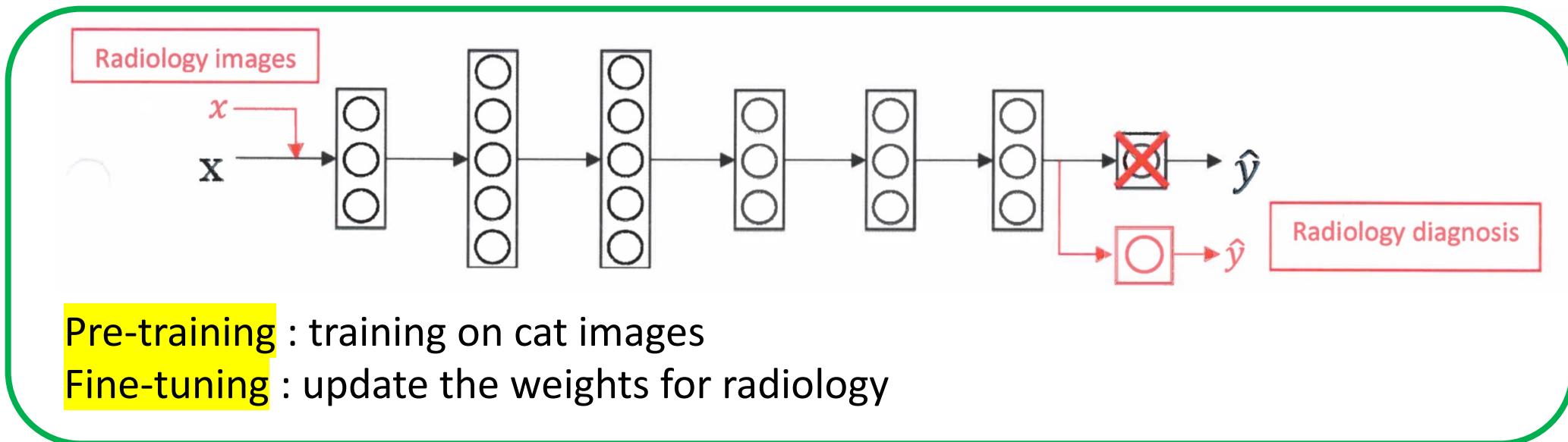
- Limit the number of iterations



Stop the training when dev set error starts increasing again

Transfer Learning

- Use weights that have been previously trained for another task
- Use cases :
 - Tasks A and B have the same input X
 - A lot more data for Task A than Task B
 - Low level features from Task A could be helpful for Task B



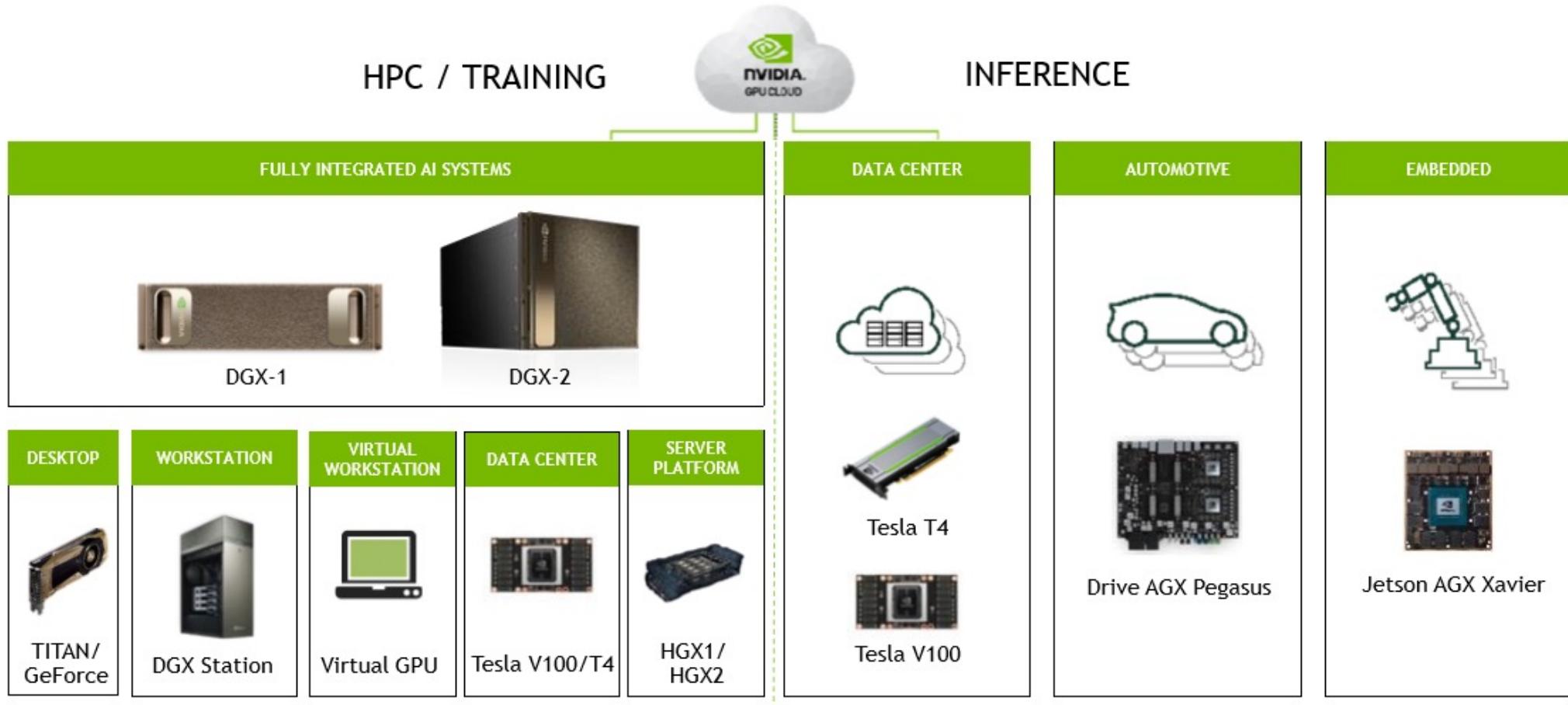
2. Time

How to improve time consumption when critical to get results



Material Acceleration (GPUs)

END-TO-END PRODUCT FAMILY



Tutorial / Practical

