Deep Forward Networks - Optimization

Thursday 08h00-09h00

Géraldine Conti, Matthew Vowels, Mykhailo Vladymyrov

Bern Winter School on Machine Learning 2024, Mürren

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

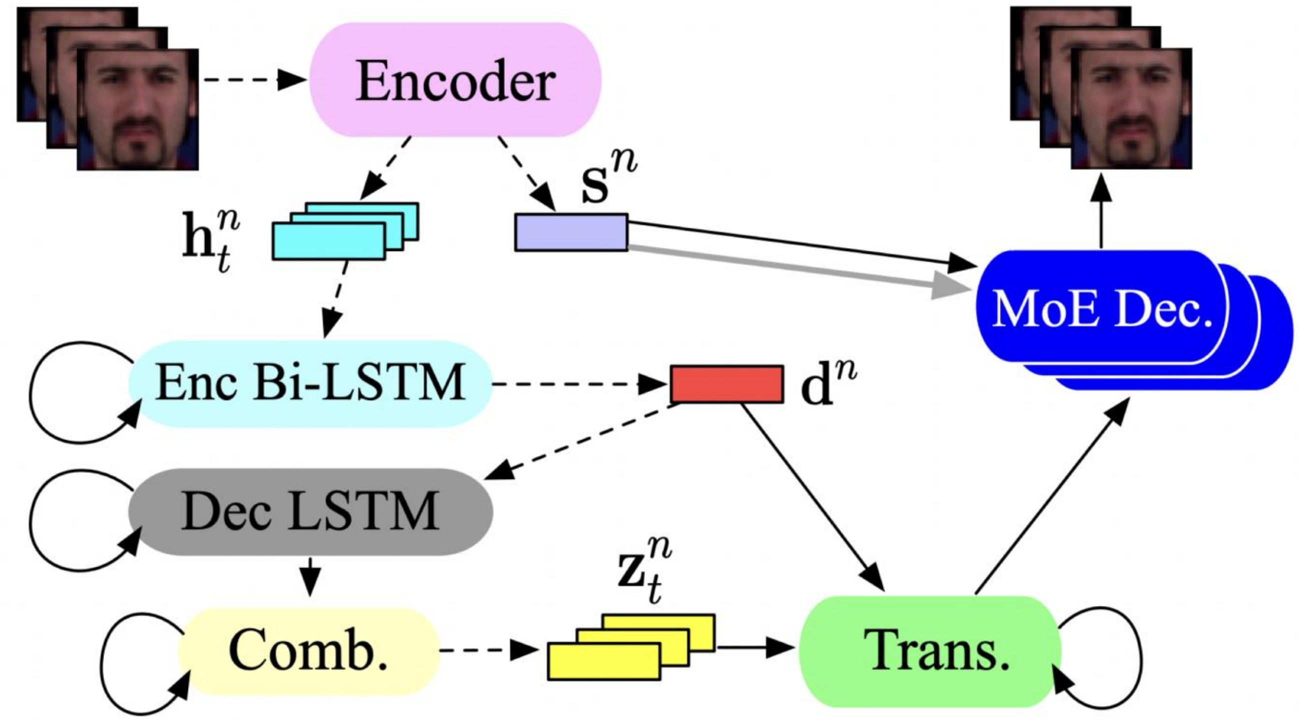
Performance Measure P

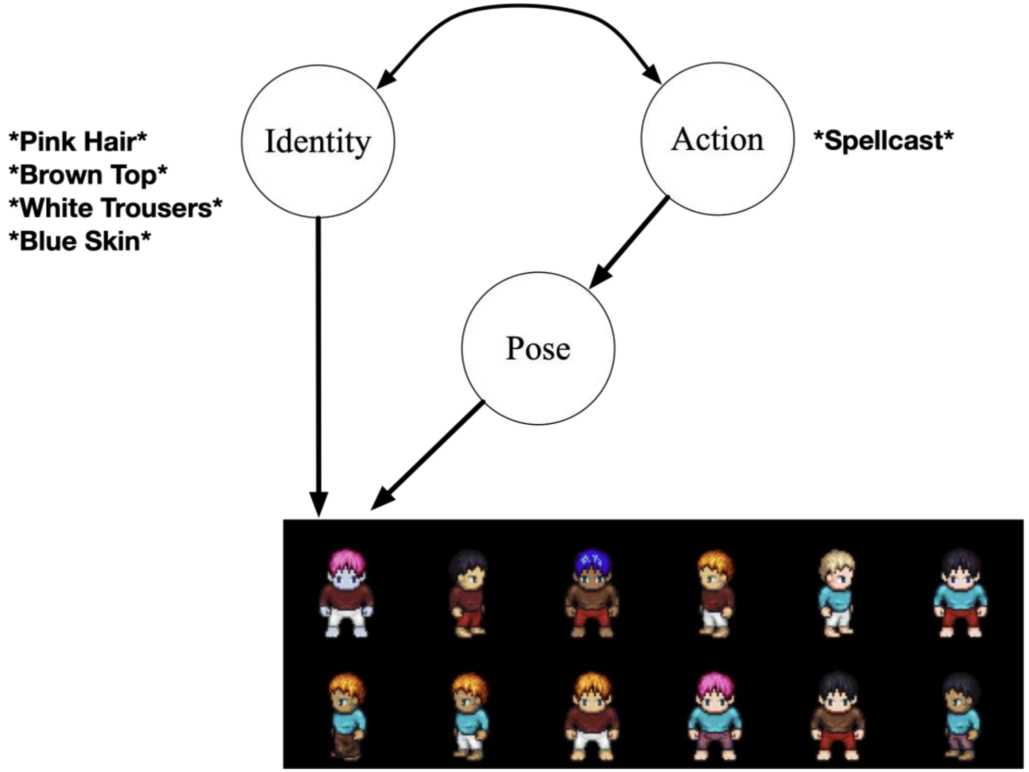
Estimate the ML algorithm performance on task T using the validation set

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)

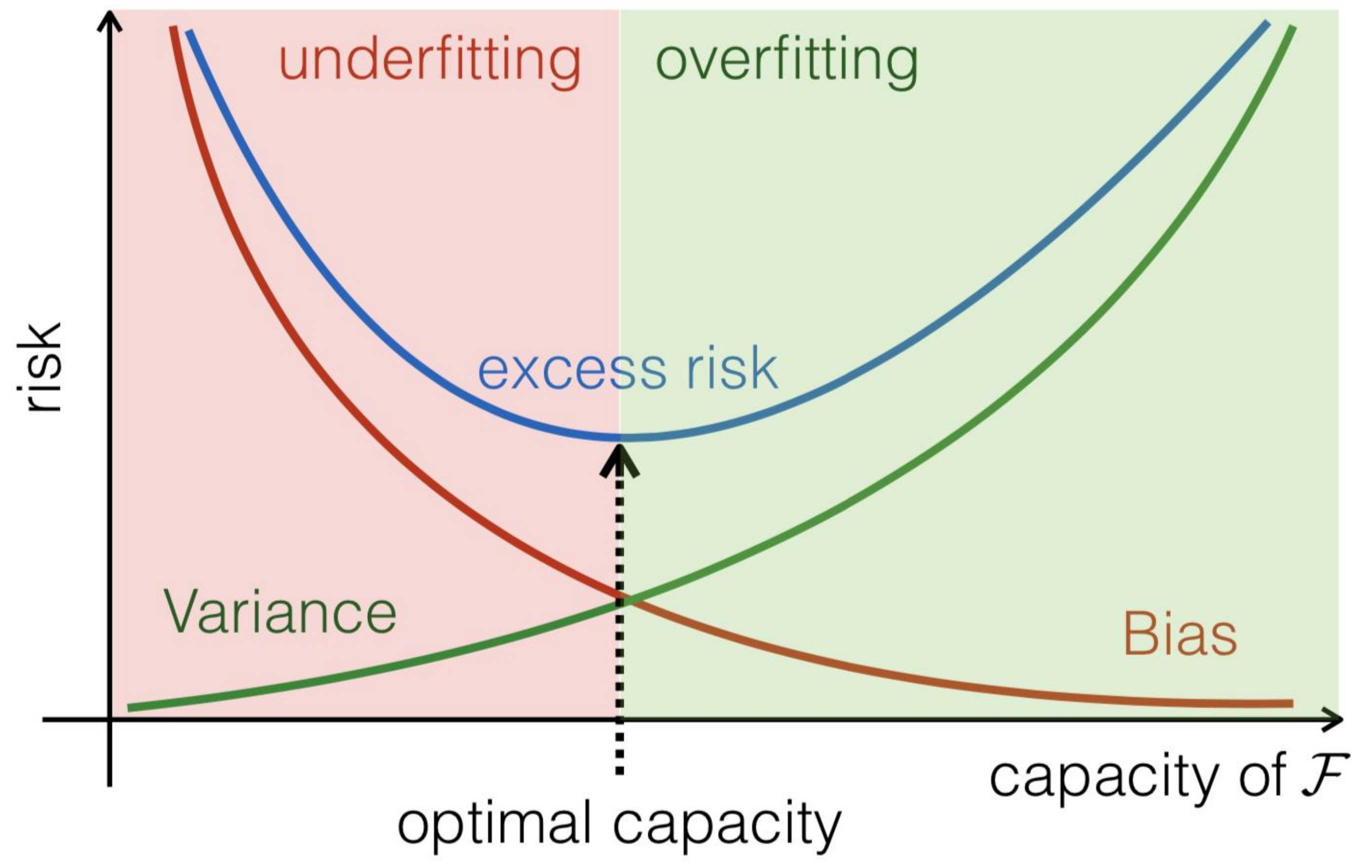
Inference

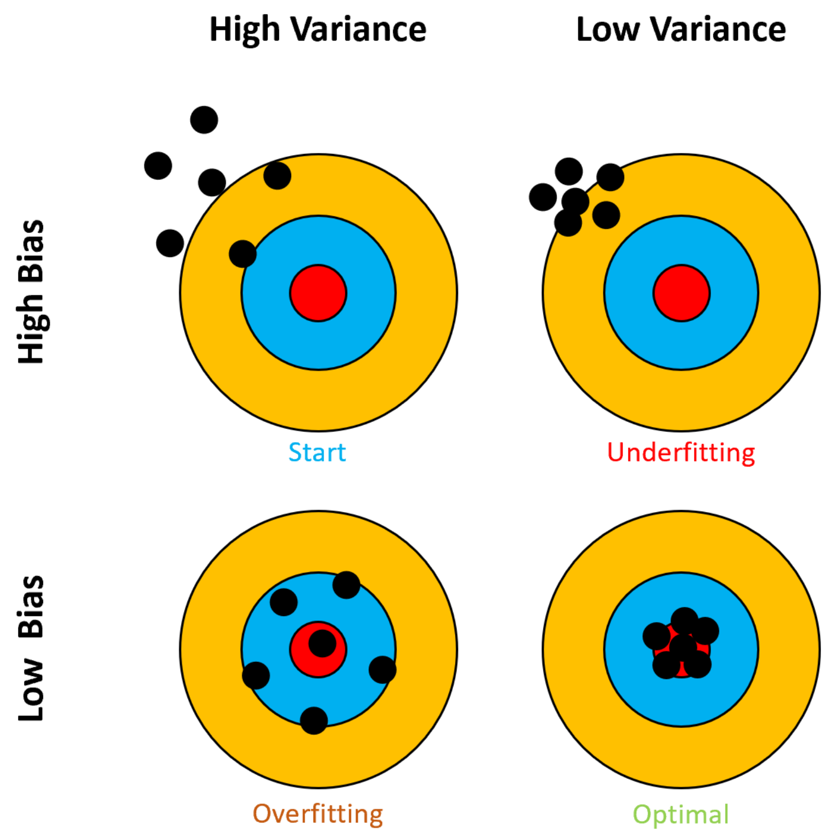




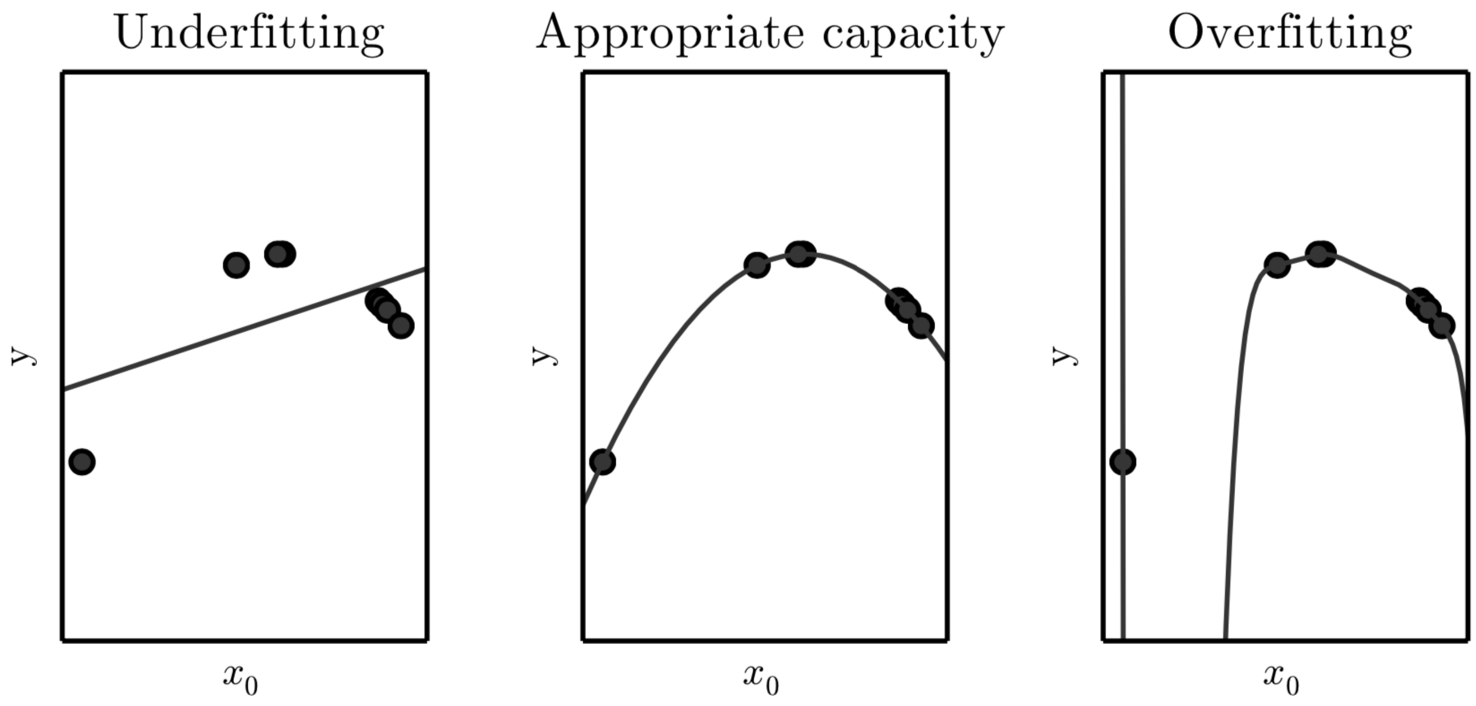
Vowels, M.J., Camgoz, N.C. and Bowden, R., 2021. VDSM: Unsupervised Video Disentanglement with State-Space Modeling and Deep Mixtures of Experts. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8176-8186).

Bias and Variance - Overfitting and Underfitting



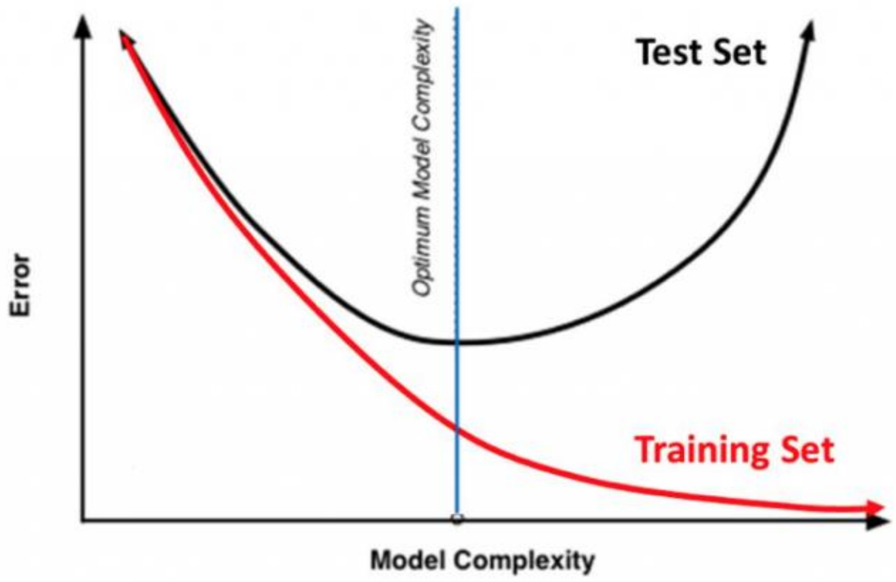


Overfitting and Underfitting



High bias

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

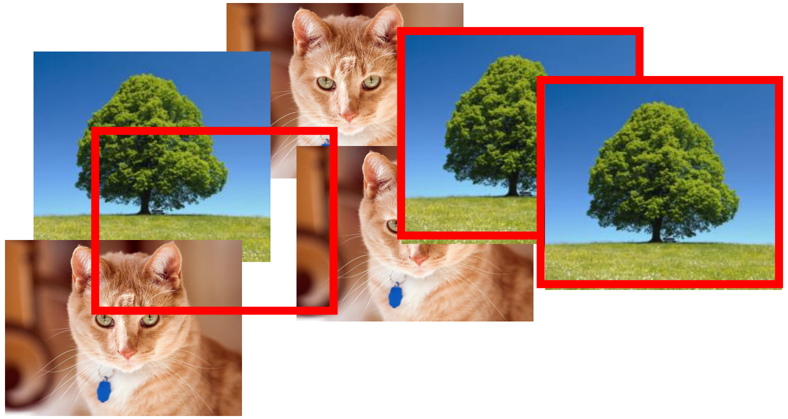
Algorithm	Classification error (%)
A	**3%**
B	**5%**

➢ *Which one is best ?*

Evaluation metrics

Ground Truth

Predictions



- **Precision** (p)

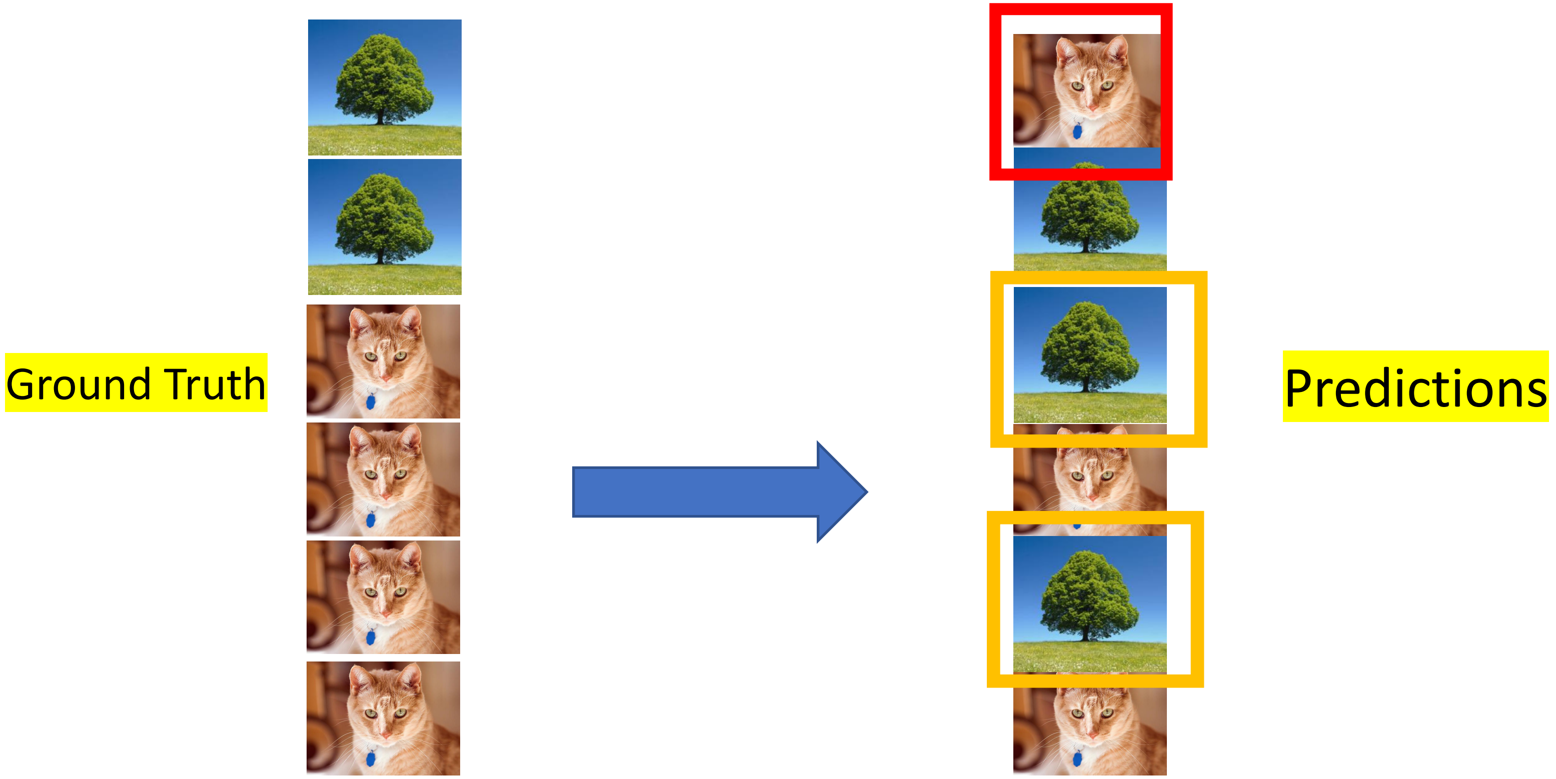
$$\text{Precision (\%)} = \frac{\textit{True positive}}{\textit{Number of predicted positive}}\ x\ 100 = \frac{\textit{True positive}}{(\textit{True positive} + \textit{False positive})}\ x\ 100$$

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

- **Recall** (r)

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

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



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

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

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

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

See also Sensitivity (same as recall) and Specificity

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

		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
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 $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	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 $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2 PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Source: wiki (Precision and Recall)

F1-score

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

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

	Precision	Recall	F_1Score
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 ?)

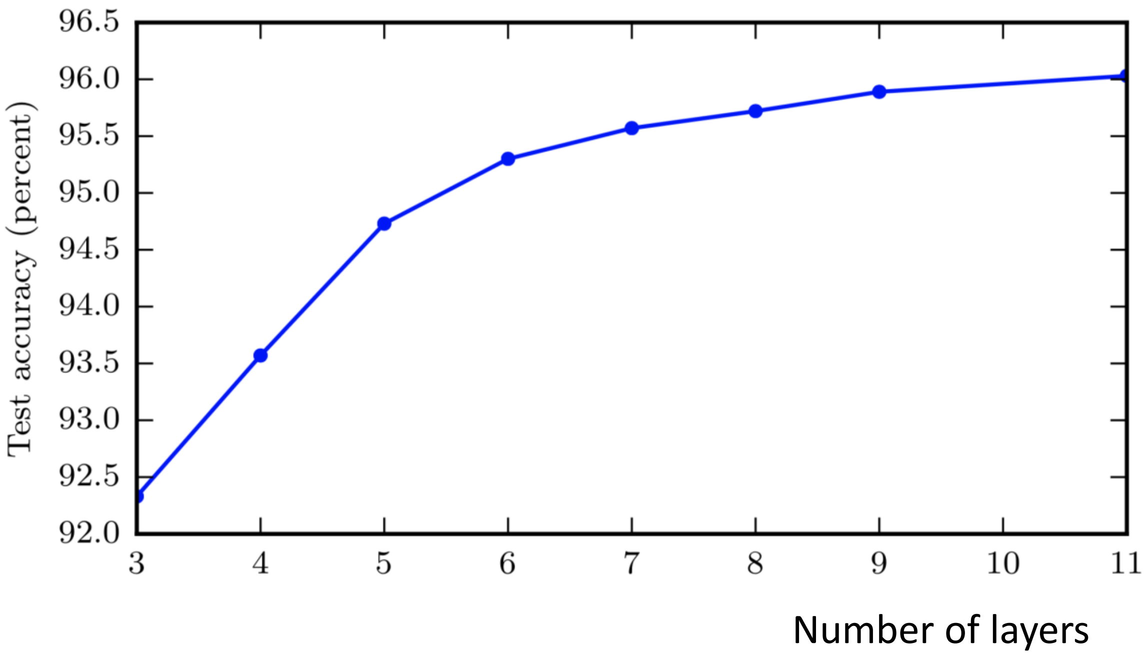
- Clean up incorrectly labeled data
  - Apply same process to your dev and test sets !

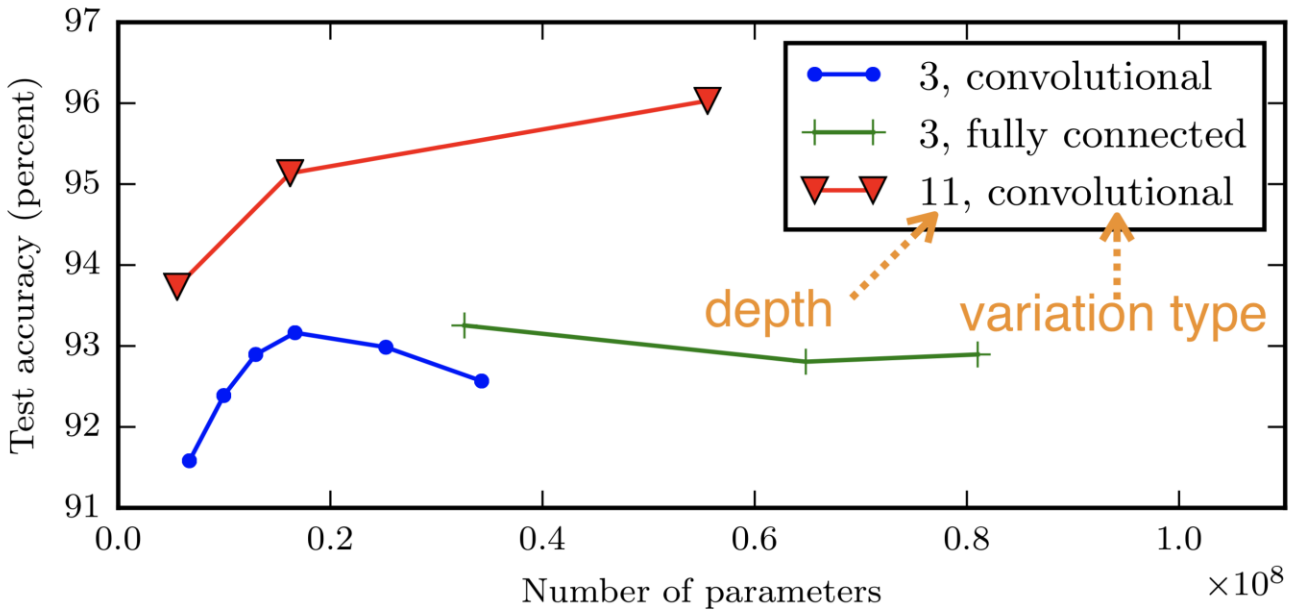
Bias Reduction techniques

- Hyperparameter tuning
- Model tuning
- Optimization algorithm

Hyperparameters : number of hidden layers/units

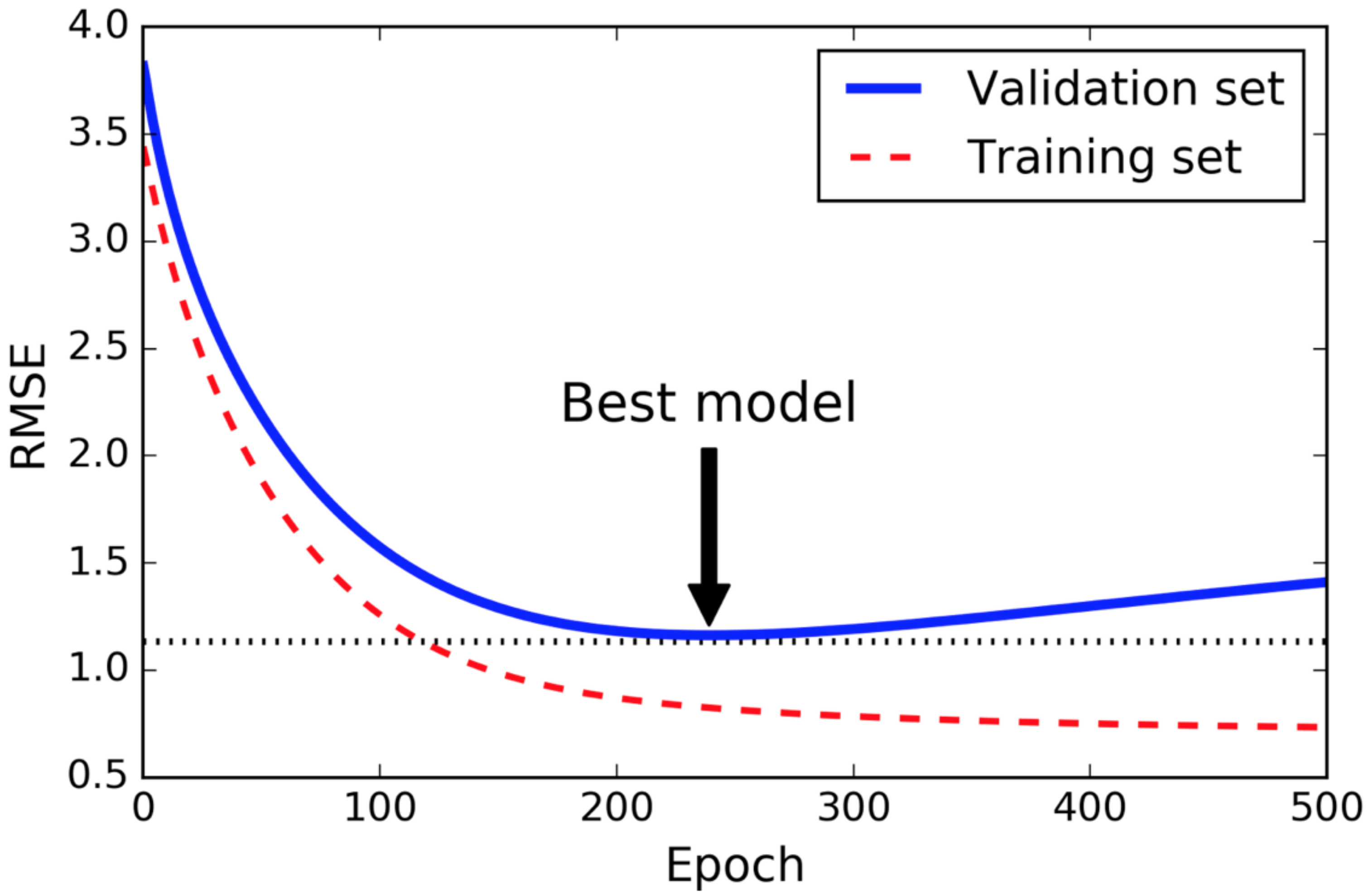
- To go deeper helps generalization (but depends on application)
- *better to have many simple layers than few highly complex ones*





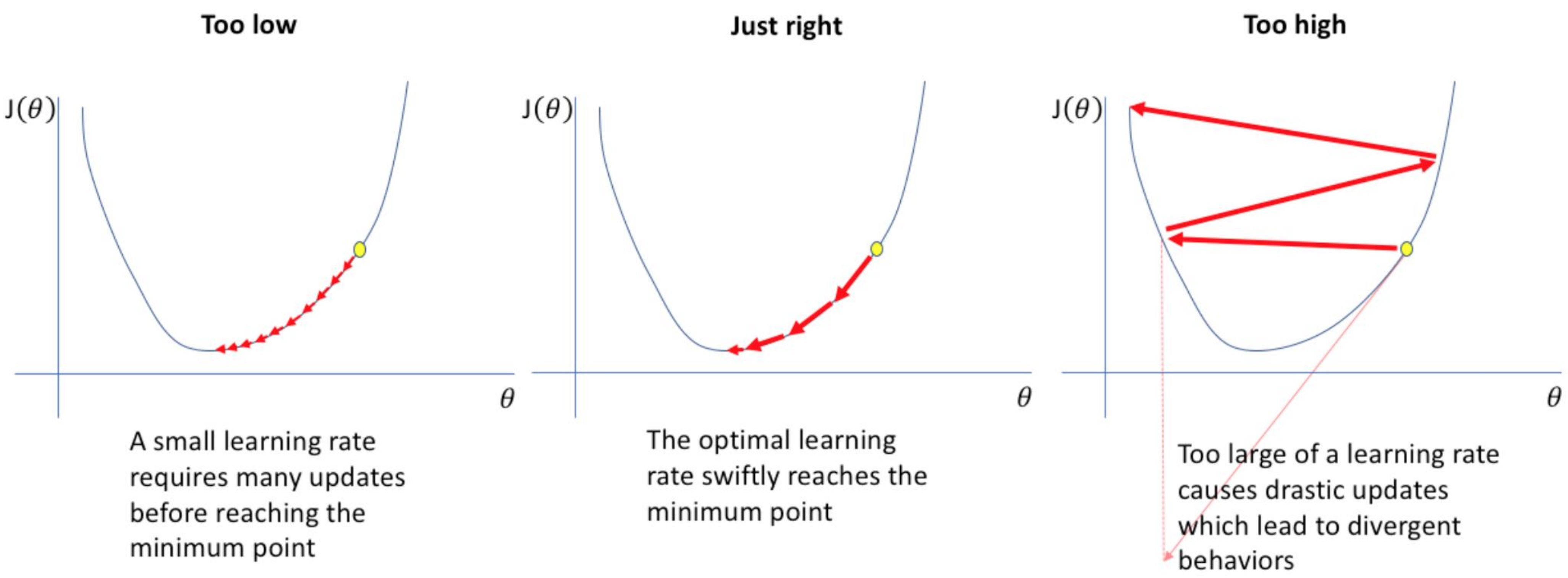
Hyperparameters : epochs

- Train *longer*



Hyperparameters : learning rate α

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

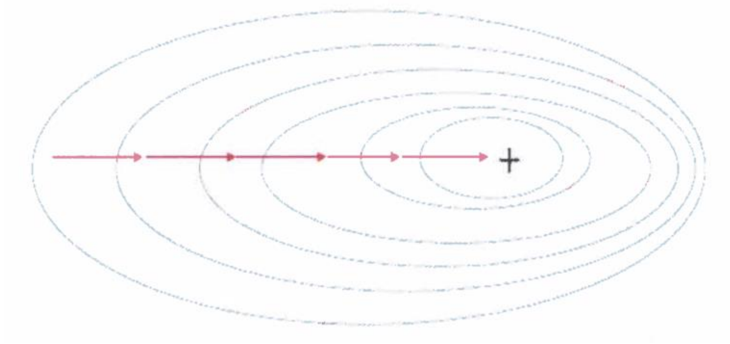


- We can also design a scheduler for the learning rate

Hyperparameters : batch size

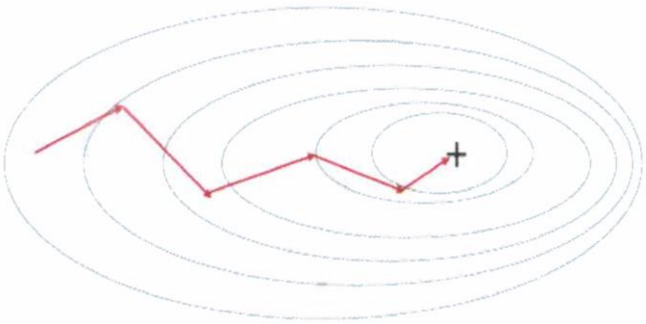
- At each iteration :

Gradient descent (GD)



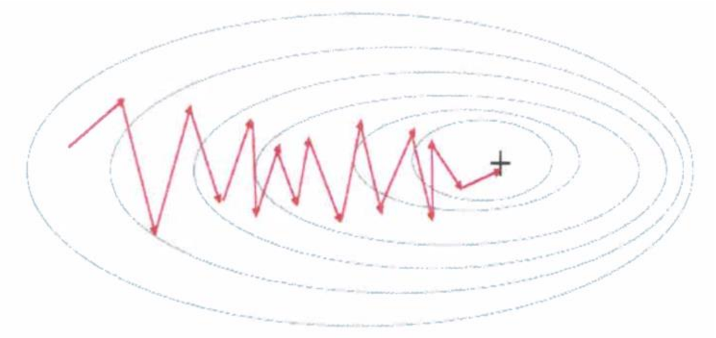
the *whole* training set

Mini-batch gradient descent



a *batch* of samples

Stochastic gradient descent (SGD)

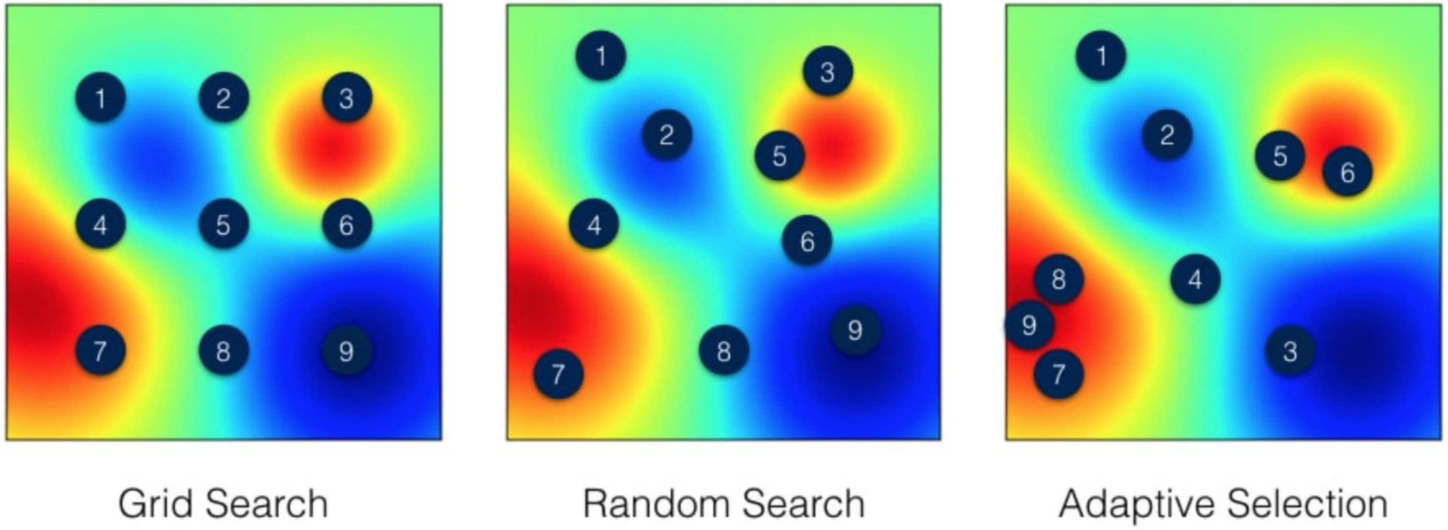


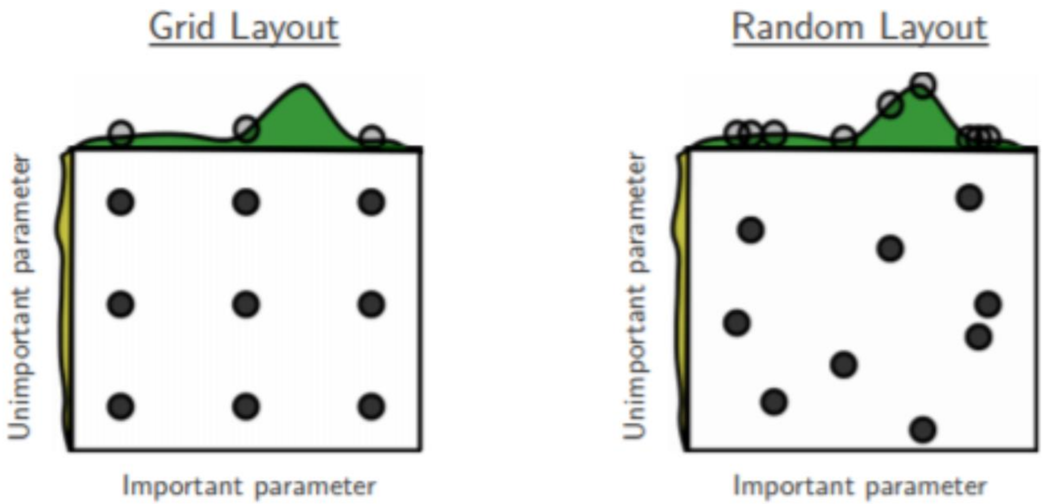
1 sample

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

Hyperparameters : Global Search

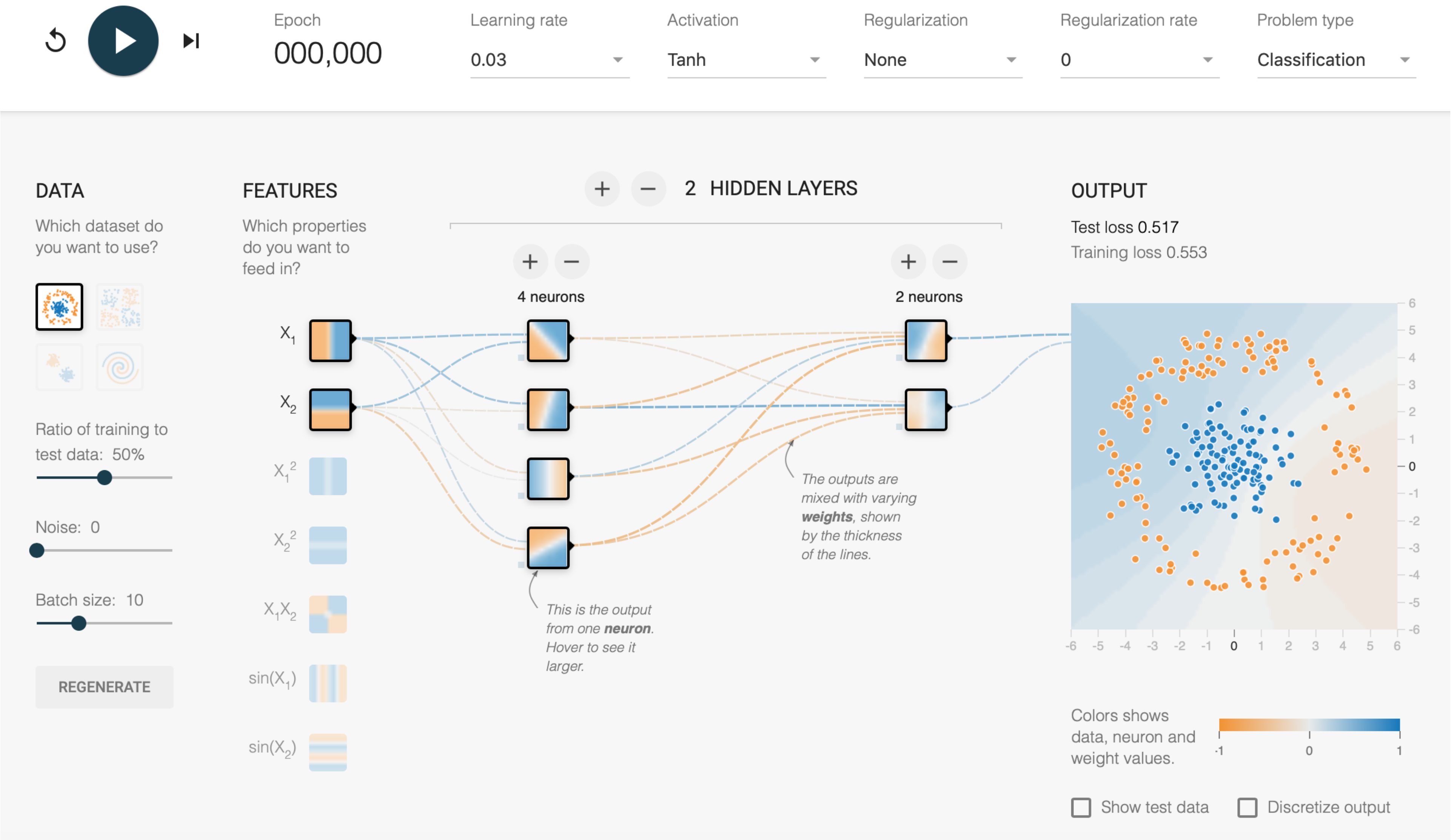
- List :
  - α (0.0001 – 1)
  - number of hidden layers
  - number of hidden units
  - learning rate decay
  - mini-batch size
  - …
- Advice is to use **random values**





Hyperparameters : Global Search

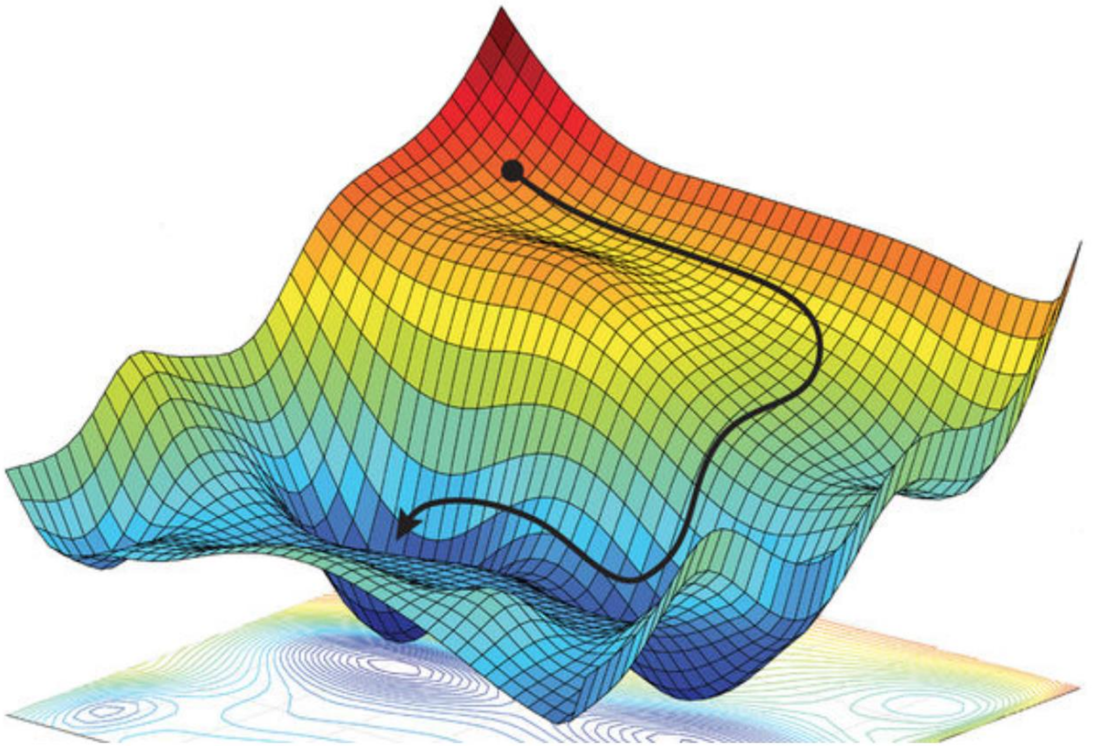
Demo



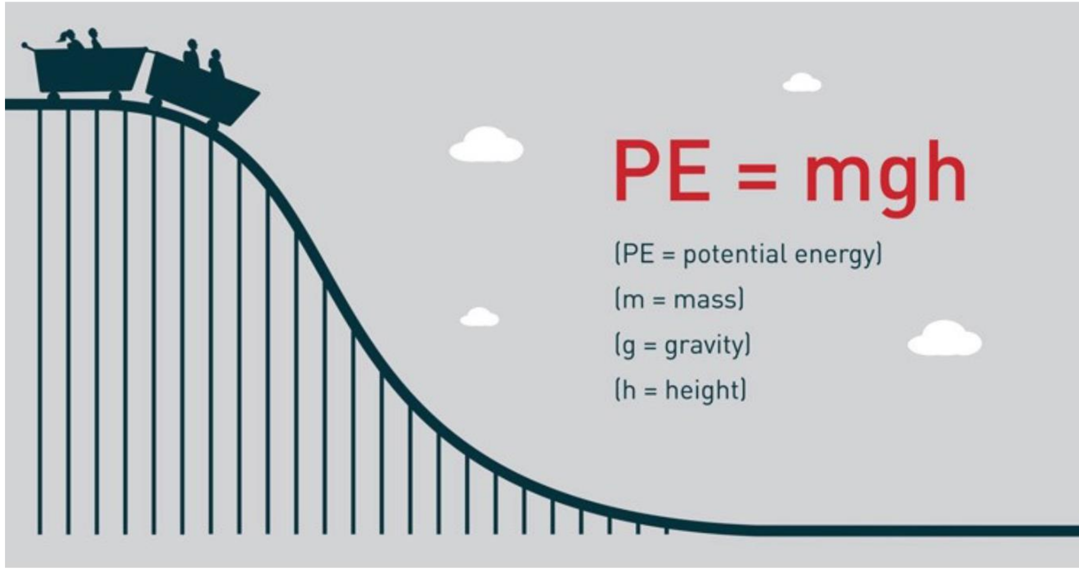
Model : Weight initialization

- 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)

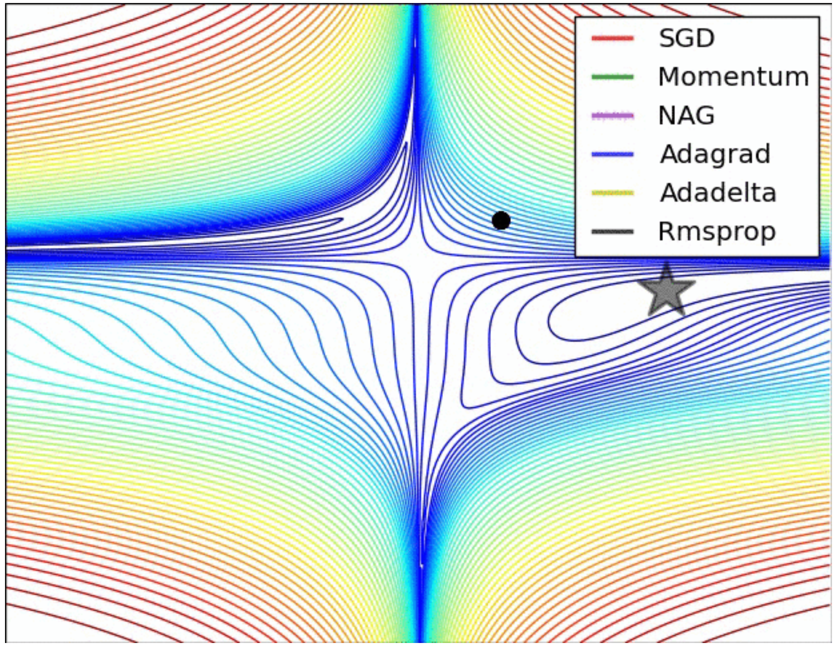
Optimization algorithm

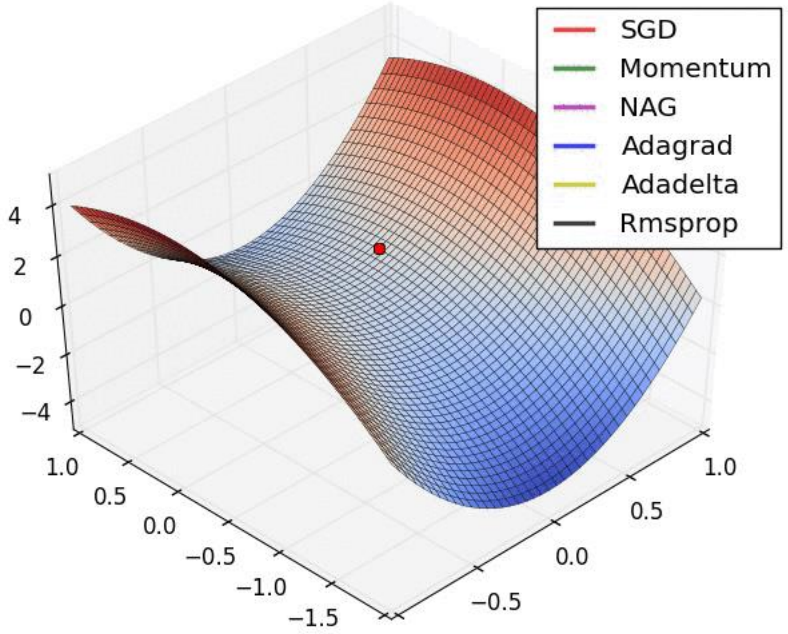


- No consensus on what algorithm performs best
- Most popular choices :
  - SGD (mini-batch gradient descent)
  - SGD + Momentum
  - RMSProp
  - RMSProp + Momentum
  - Adam
- Strategy : *pick one and get familiar* with the tuning

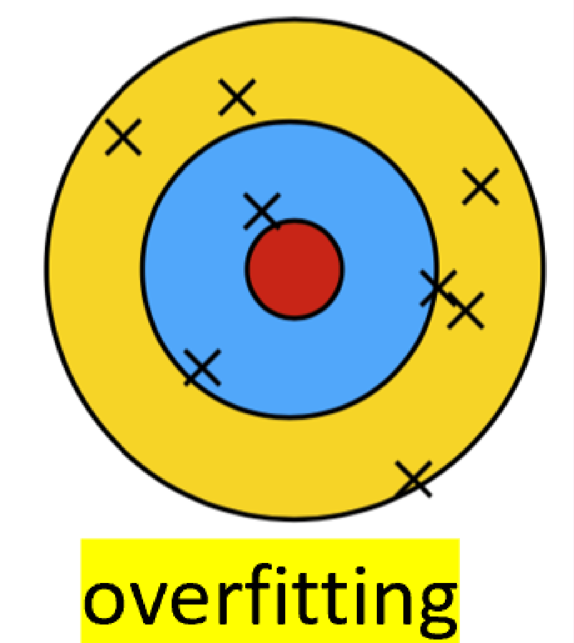


Gradient Descent Variations





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)

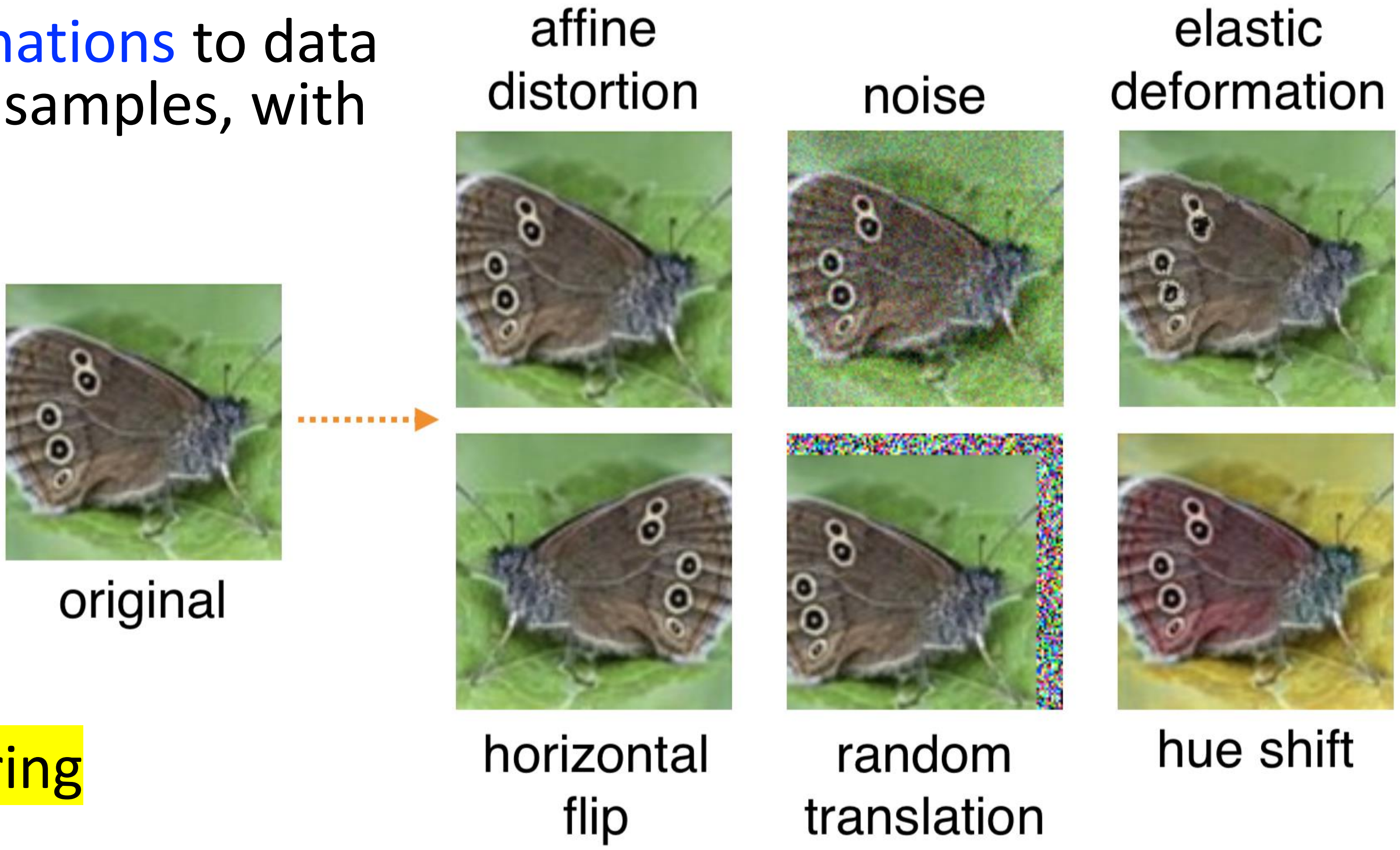
- Use cases : if few data or if model has more than 50 layers (CNN)

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

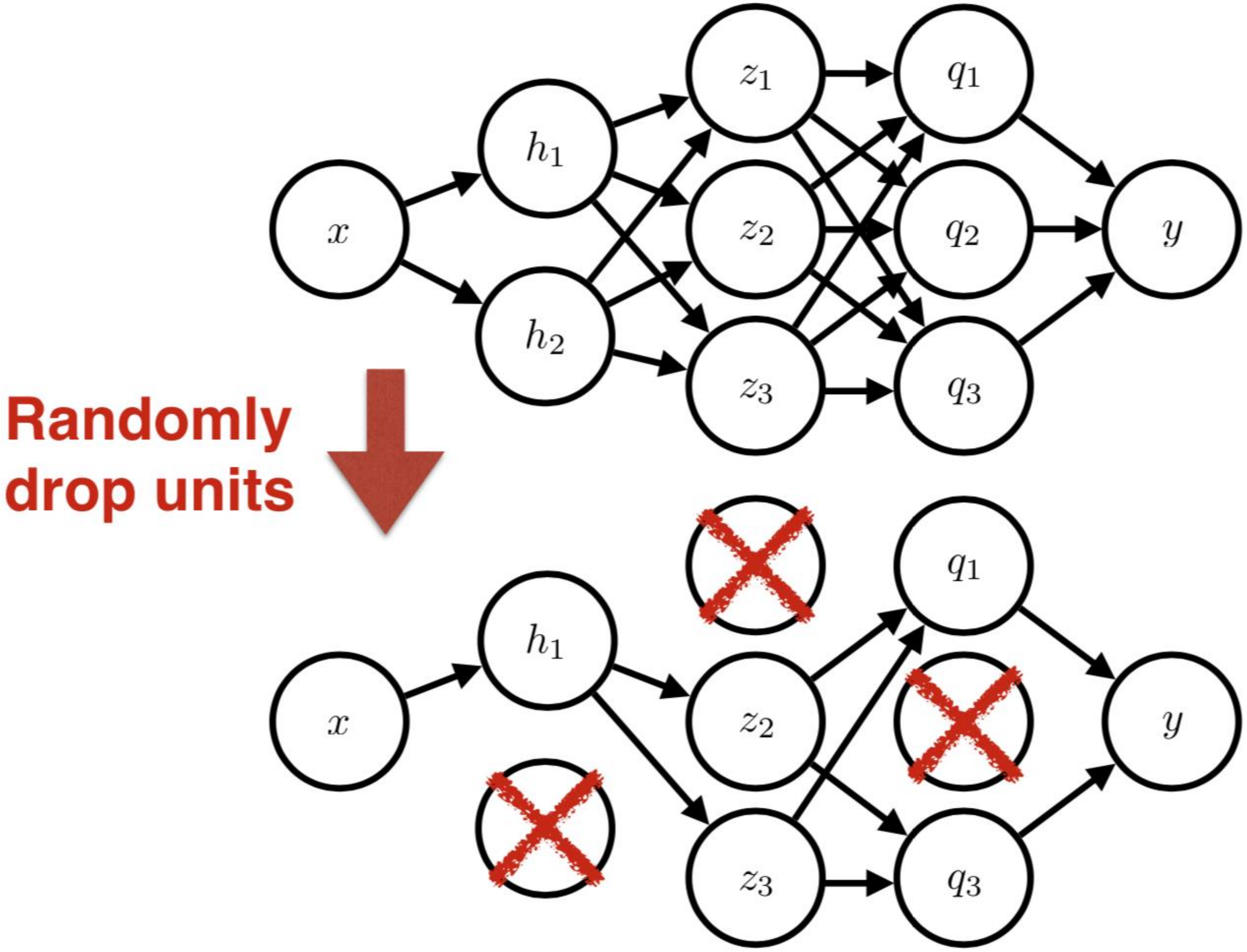
Regularization (*Dataset*) : Augmentation

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



- Process also called jittering

Regularization (*Model*) : Dropout

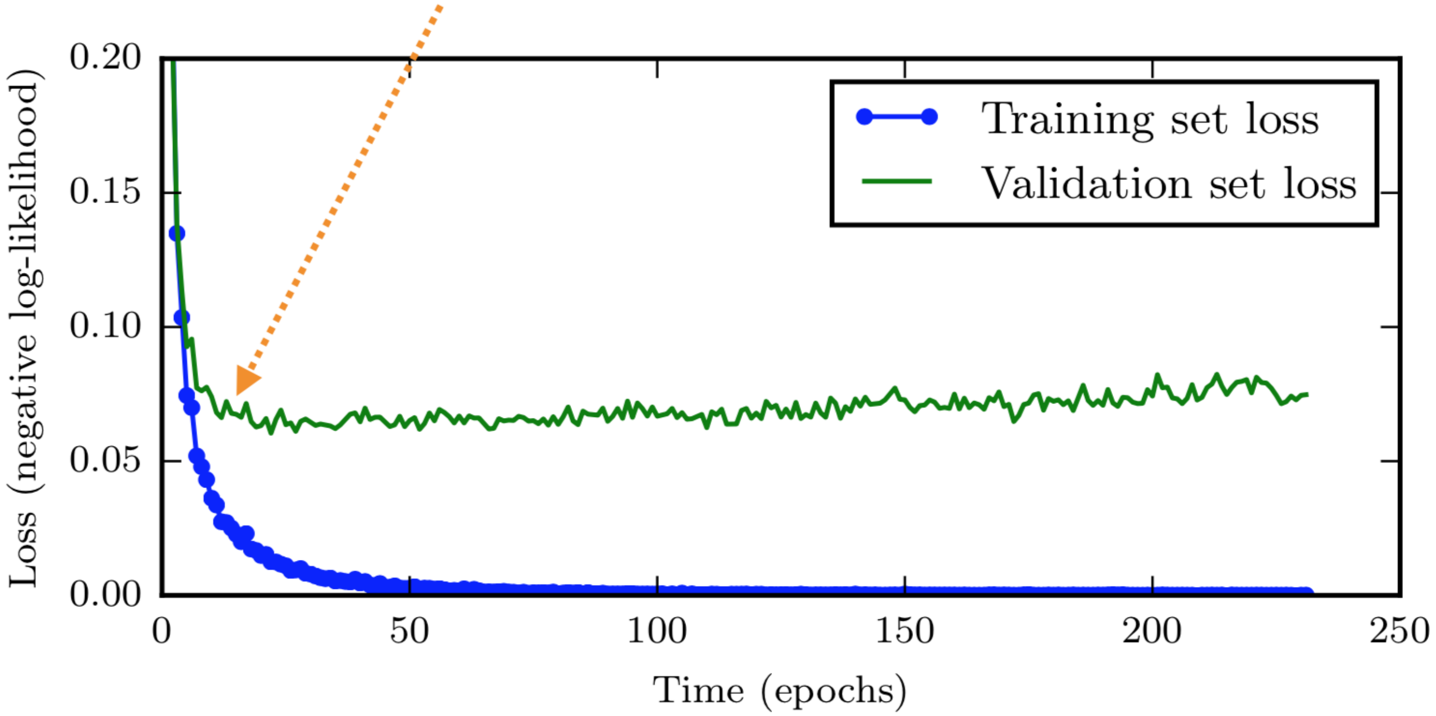


- Apply it both in forward and backward propagations
- *BUT* use it only in the *training phase !*

Regularization (*Training*) : Early stopping

- Limit the number of iterations

terminate while validation set
performance is better

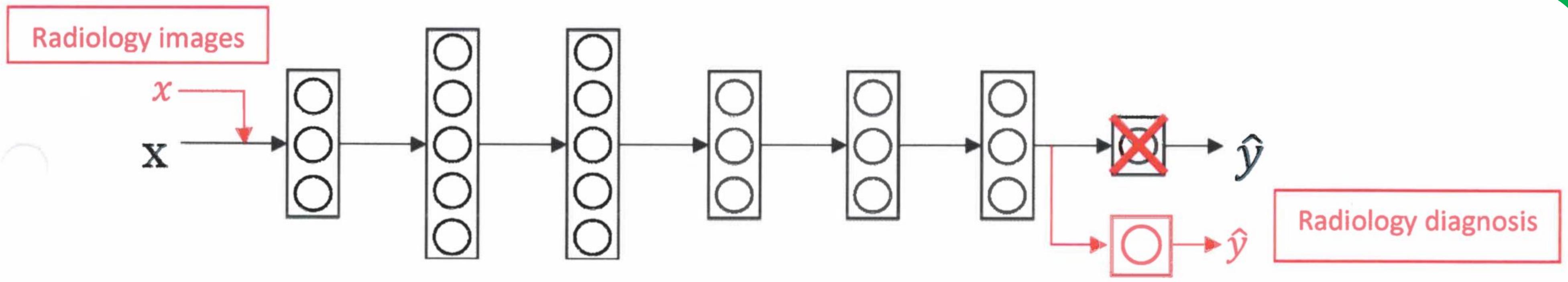


Stop the training when dev set error starts increasing again

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

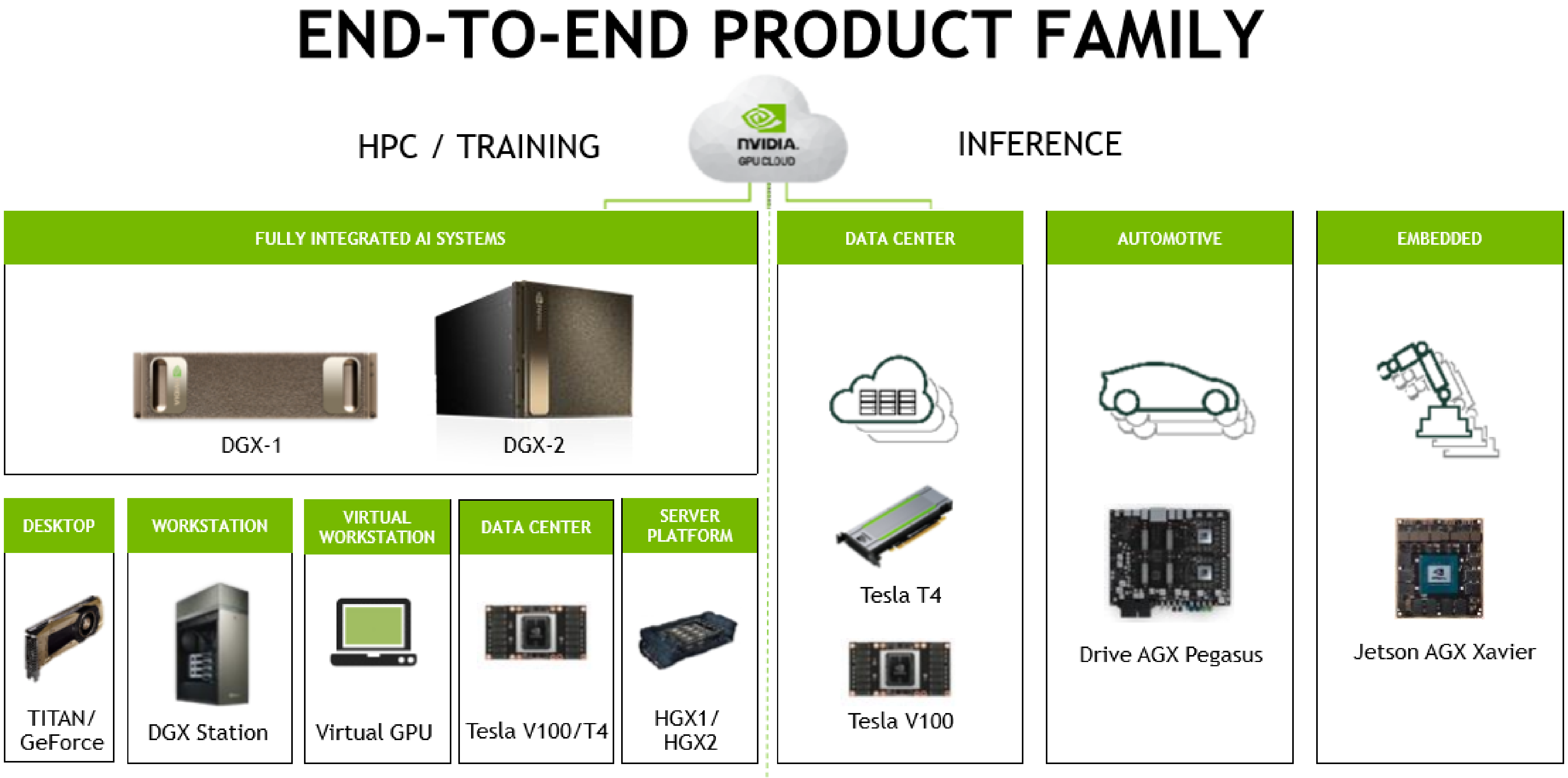


Pre-training : training on cat images
Fine-tuning : update the weights for radiology

2. Time

How to improve time consumption when critical to get results

Material Acceleration (GPUs)



Tutorial / Practical