Module 2 :

Deep Networks

Deep Forward Networks

Géraldine Schaller-Conti, August 2024

Discussion Session

- Review of Notebook 4 (module 1)

- Perceptrons
- Activation functions
- Image classifier
- Regression MLP
- Saving and Restoring a model

Bibliography

- Deep Learning book (Goodfellow, Bengio, Courville)
- Machine Learning @ Stanford (Prof Andrew Ng)
- Hands-On Machine Learning with Scikit-Learn & Tensorflow (Aurélien Géron)

Learning Objectives

- What are Deep Forward Networks ?
- Gradient problems and solutions
- Optimization
  - Performance
  - Time

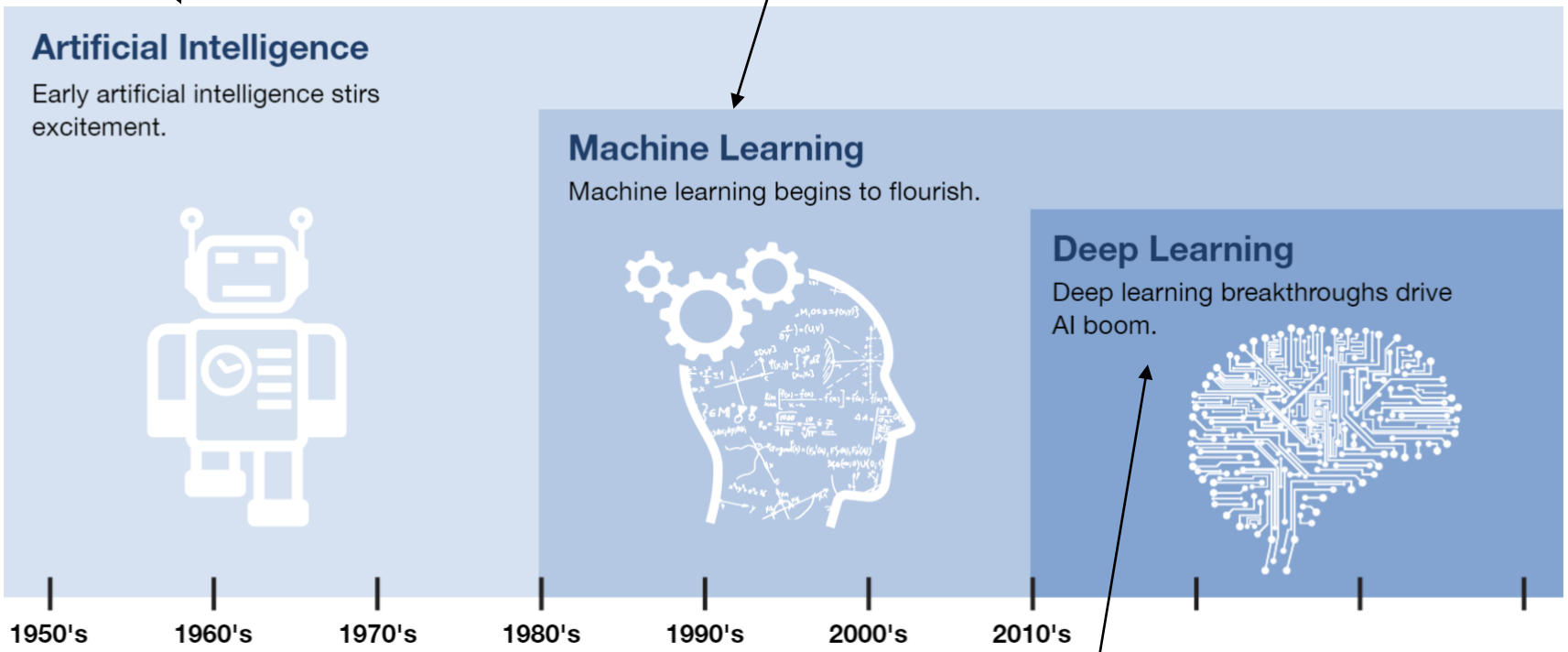
Deep Learning

- Definition
- Motivations
- Comparison with Machine Learning

Deep Learning: Apparition

Ability of a machine to act in a way that imitates intelligent human behaviour

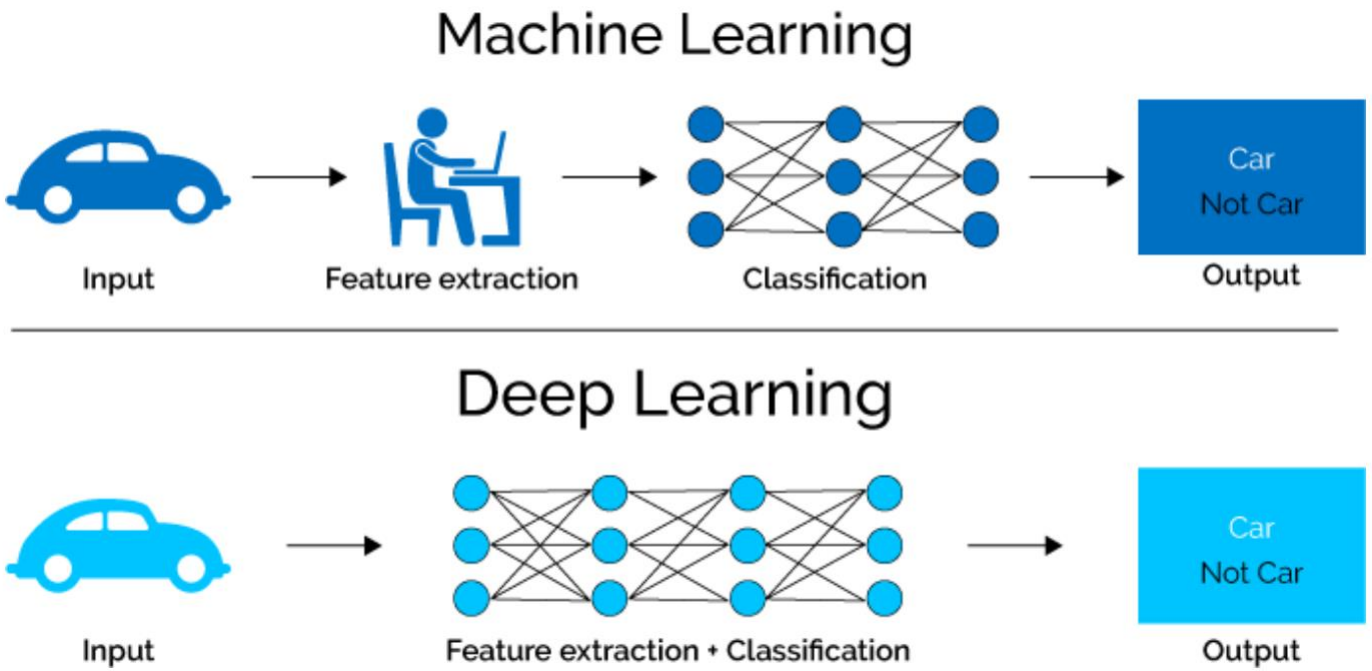
Study of the algorithms and methods that enable computers to solve specific tasks without being explicitly instructed how to solve these tasks



Subset of ML algorithms that make use of large arrays of Artificial Neural Networks

Deep Learning: Definition

- Multiple levels of features represented by multiple layers
- Each level represents abstract features that are discovered from the features represented in the previous level.
  - level of abstraction increases with each level

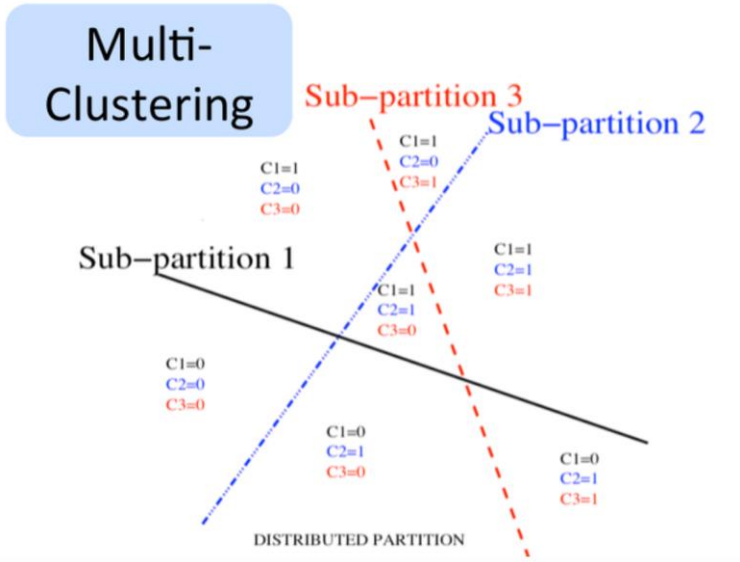


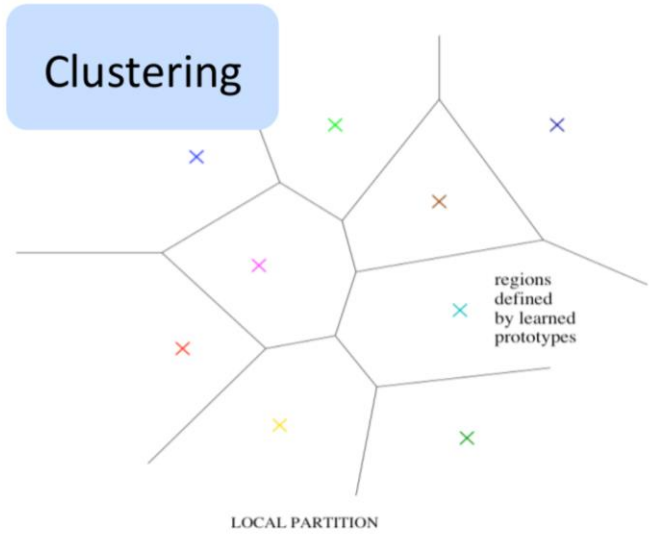
Ingredients towards AI

- 3 key ingredients for ML towards AI :
  - Lots of data
  - Very flexible models (as we get more data)
    - *we change the number of hidden units*
  - Powerful priors to defeat curse of dimensionality
    1) distributed representations
    2) deep architecture

1. Distributed Representations

- Possible to represent an exponential number of regions with a linear number of parameters
- Can learn a very complicated function with a low number of examples
- Features are individually meaningful





Algorithms using non-distributed representations :

clustering, nearest neighbours, RBF SVMs, decision trees,…

2. Deep Architecture

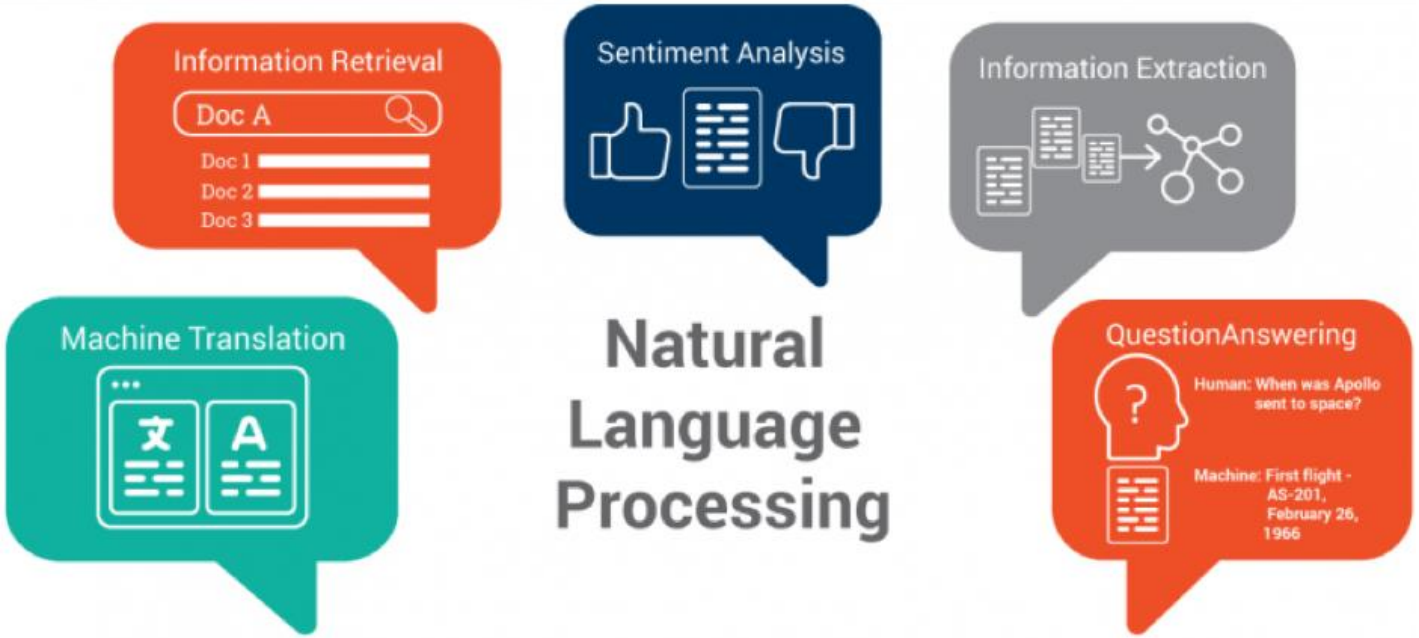
- Universal approximation property (shallow NN sufficient to represent any function with required degree of accuracy)
- Deep network allows to represent the same function with fewer hidden units.
- Number of units needed in a shallow network can be *exponentially larger* than in deep enough network

Importance of Deep Nets

- Form a basis for many commercial applications

1) CNNs are a special kind of FNN
   - Used to recognize objects from photos,…

2) They are a conceptual stepping stone to RNNs
   - Power many Natural Language Processing (NLP) applications



DL versus ML

Criteria	Machine Learning	Deep Learning
Data size		
Computer infrastructure		
Feature engineering		
Nature of problem		

Design Decisions

- Similar to linear model
  - Basics of gradient descent :
    - Optimizer
    - Cost function
    - Form of output units

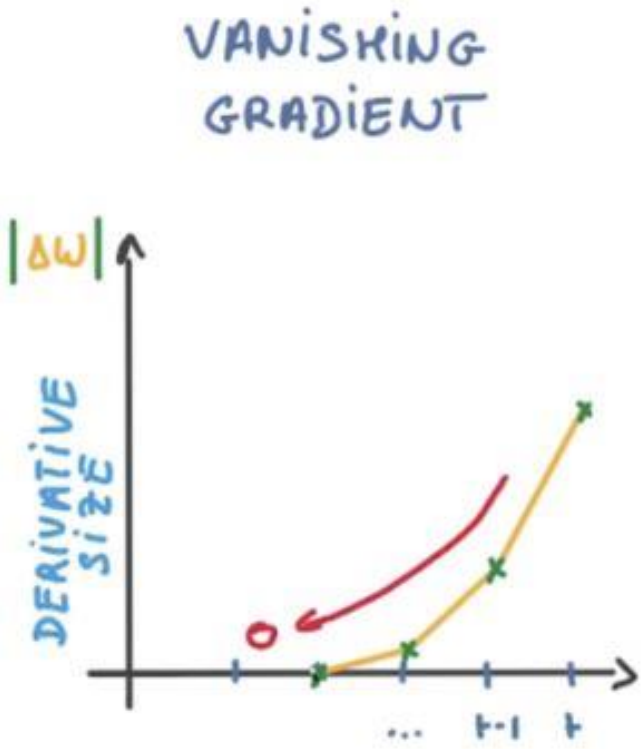
- Unique to DFNN
  - Architecture of network
    - Number of layers
    - Number of units in each layer
  - Learning requires gradients of complicated functions
    - Backpropagation

Training Challenges

- Example : 10 layers with 100s of neurons each, connected by hundreds of thousands of connections
- Vanishing/exploding gradient problem
  - Problem affecting deep neural networks
- Training would be extremely slow with such a large network
  - Transfer learning could help

Vanishing Gradient Problem

- Problem affecting deep neural networks
- During backpropagation, gradients get smaller and smaller
- Gradient Descent update leaves the lower layer connection weights (close to input) virtually unchanged
  - The lower layers are very hard to train
- training never converges to a good solution



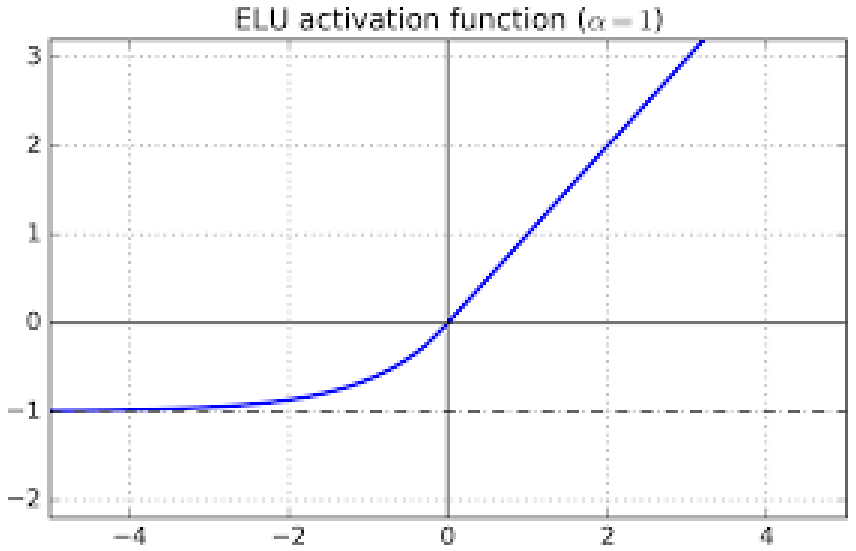
Mitigating the vanishing gradient problem (1)

- Historically, people found 2 ingredients to act upon to cure the gradient problem

  1) activation function in the hidden layers

  - logistic sigmoid activation function could be an issue

  - use instead ReLU, ELU or any variant of it



Mitigating the vanishing gradient problem (2)

2) random initialization of the weights could be an issue

- The initial weights need to break the symmetry between different units

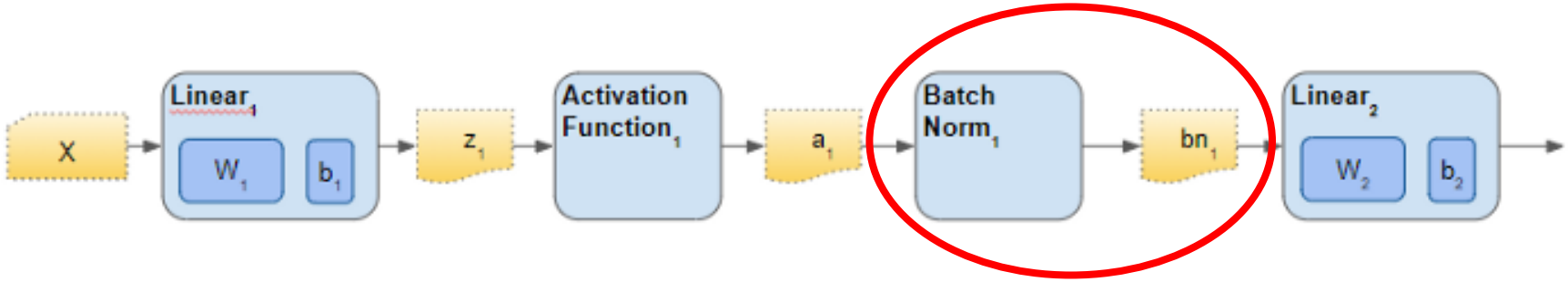
- With Tanh, Xavier weights are very good to prevent the gradient problem
  - random draws from uniform probability distribution (U)
    - weight = U[-(1/sqrt(n)), 1/sqrt(n)], n=number of inputs to the node

- With ReLU, a good option is to use He weights
  - random number with a Gaussian probability distribution (G)
    - weight = G(0.0, sqrt(2/n)

Mitigating the vanishing gradient problem (3)

- Batch normalization helps the hidden layers work together smoothly
  - It adjusts and normalizes their input values as the network learns

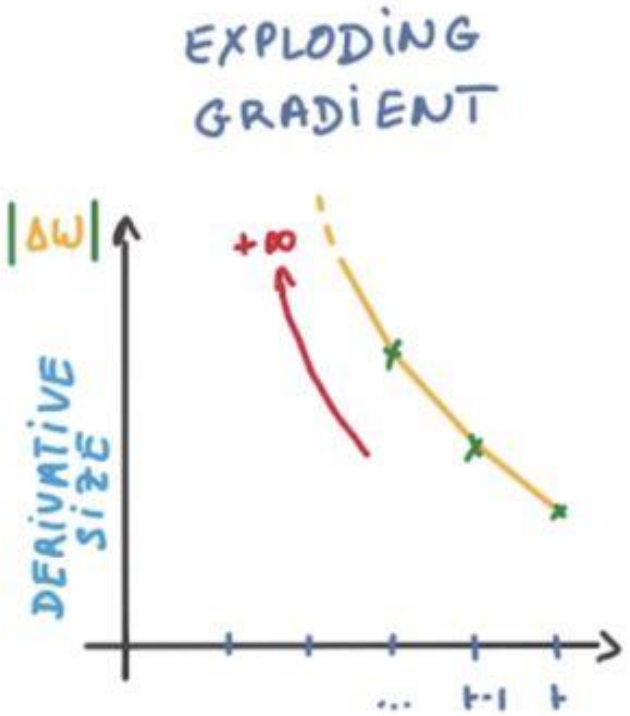
- Recipe :
  - Collect a batch of data during training [5 10 15]
  - Calculate mean and variance [mean=10, variance=16.67]
  - Normalize the values such that they have a similar size range [-0.6, 0, 0.6]
  - Scale the values back to a desired range, shift them to make sure they have the right overall distribution (scale=2, shift=1 → [0.8, 1, 2.20] the new values)
  - Trainable parameters helping the scaling and shifting



- As a consequence, gradient descent will converge better during training
  - Small signals will not get lost during backpropagation

Exploding Gradient Problem

- Problem affecting mostly RNNs
- Gradients grow bigger and bigger
- Many layers get insanely large weight updates
- The algorithm diverges

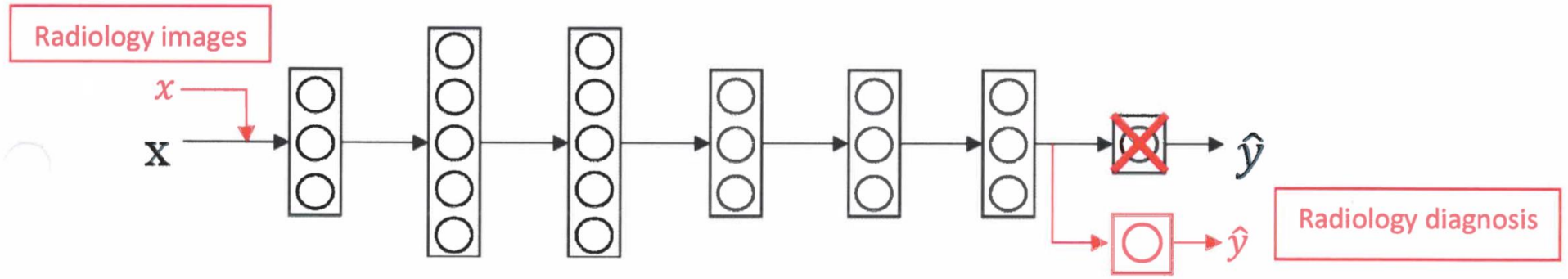


Gradient Clipping

- Tackles the exploding gradient problem
- Clip the gradient during backpropagation so that they never exceed some threshold
- Mostly used in RNNs
- Nowadays, people prefer batch normalization

Transfer Learning

- Always try to find an existing neural network that accomplishes a similar task to the one you are trying to tackle
- reuse the lower layers of this network
  - Output layer should usually be replaced
- Speeds up training and requires much fewer training data



Pre-training : training on cat images

Fine-tuning : update the weights for radiology

Summary of techniques to speed up training

- Good initialization strategy for the connection weights (Xavier, He)
- Good activation function in hidden layers (ReLU, ELU)
- Batch Normalization
- Reusing parts of a pretrained network (transfer learning)
- Faster optimizer (momentum, Adam)

Exercise

Initialization	He initialization
Activation function	ELU
Normalization	Batch normalization
Optimizer	Gradient Descent
Learning rate schedule	None

What can you try if :

- You can’t find a good learning rate ?
- Your training set is too small ?
- You need a faster model at test time ?

https://www.youtube.com/watch?v=He4t7Zekob0

Two-Minute Papers

Optimization

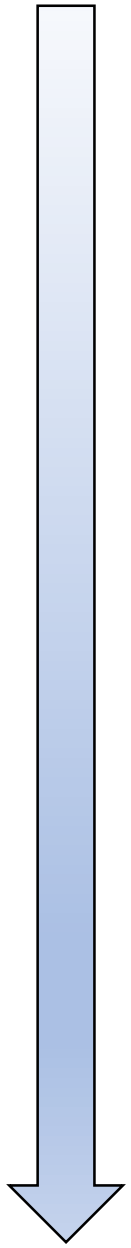
In terms of performance and time

1. Performance

Bias reduction techniques

Variance reduction techniques

List of Errors



- Bayes optimal error (theoretical limit)
- Human-level error
- Training error
- Val error
- Test error
- Real world

Avoidable bias

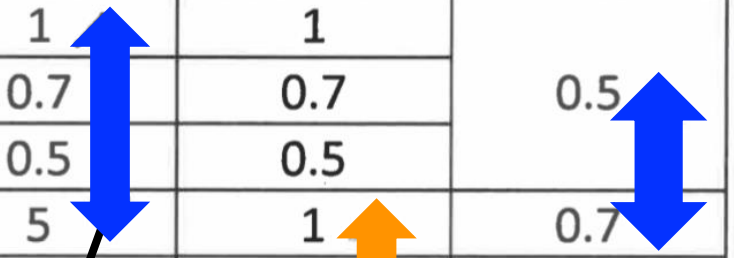
Variance

Depending on where you get the *largest discrepancy*, you will use a different technique to tackle the problem

List of Errors

Medical image classification

	Classification error (%)		
	Scenario A	Scenario B	Scenario C
Human (proxy for Bayes error)	1	1	0.5
	0.7	0.7	
	0.5	0.5	
Training error	5	1	0.7
Validation error	6	5	0.8



Bias reduction technique

Variance reduction technique

Bias Reduction techniques

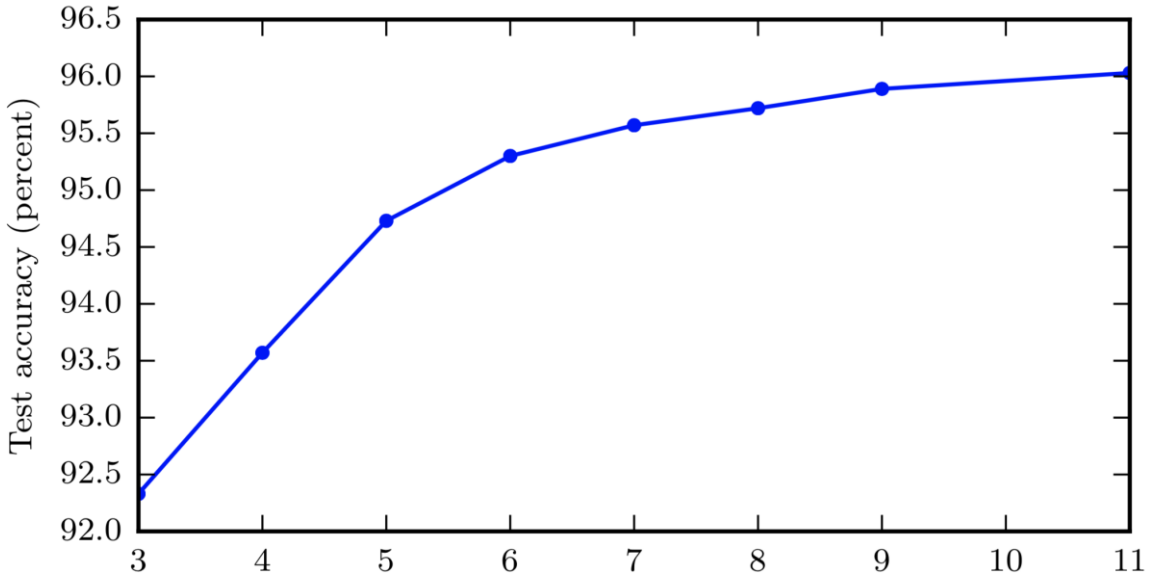
Hyperparameter tuning

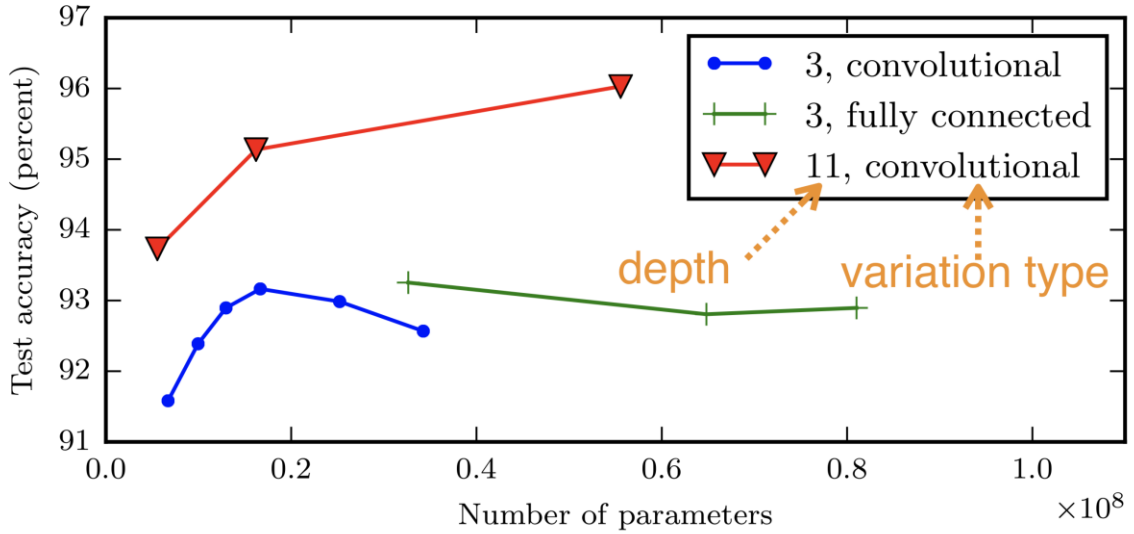
Model tuning

Optimization algorithm

Hyperparameters:
number of hidden layers

- To go deeper helps generalization





better to have many simple layers than few highly complex ones

Hyperparameters:
number of neurons in hidden layers

- Input and output layer number of neurons is determined by the type of input/output of the task
- For hidden layers, a common practice was historically to size them to form a funnel, with fewer neurons at each layer
- Now, simply use the same size for all hidden layers (homogeneous/symmetric architecture)
  - *Only one hyperparameter left*
  - Better parameter sharing, regularization as network is less sensitive to the choice of layer sizes

Hyperparameters:
activation function

A generalisation of ReLU is

$$g(z, \alpha) = \max\{0, z\} + \alpha \min\{0, z\}$$

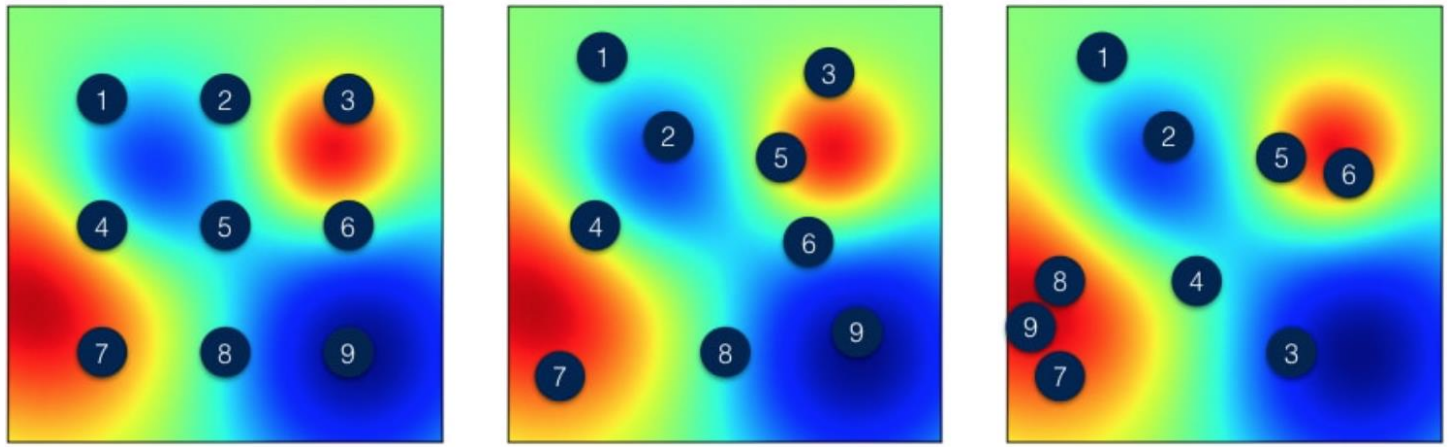
To avoid a null gradient the following are in use

1. Absolute value rectification $\alpha = -1$
2. Leaky ReLU $\alpha = 0.01$
3. Parametric ReLU α learnable
4. Maxout Units $g(z)_i = \max_{j \in S_i} z_j$

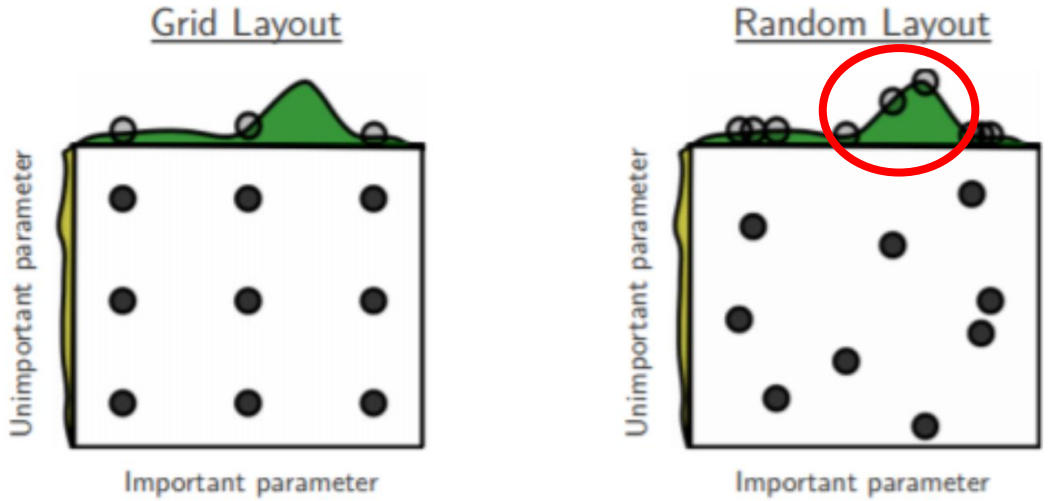
$$\cup_i S_i = [1, \ldots, m]$$

$$S_i \cap S_j = \varnothing \quad i \neq j$$

Hyperparameters: global search



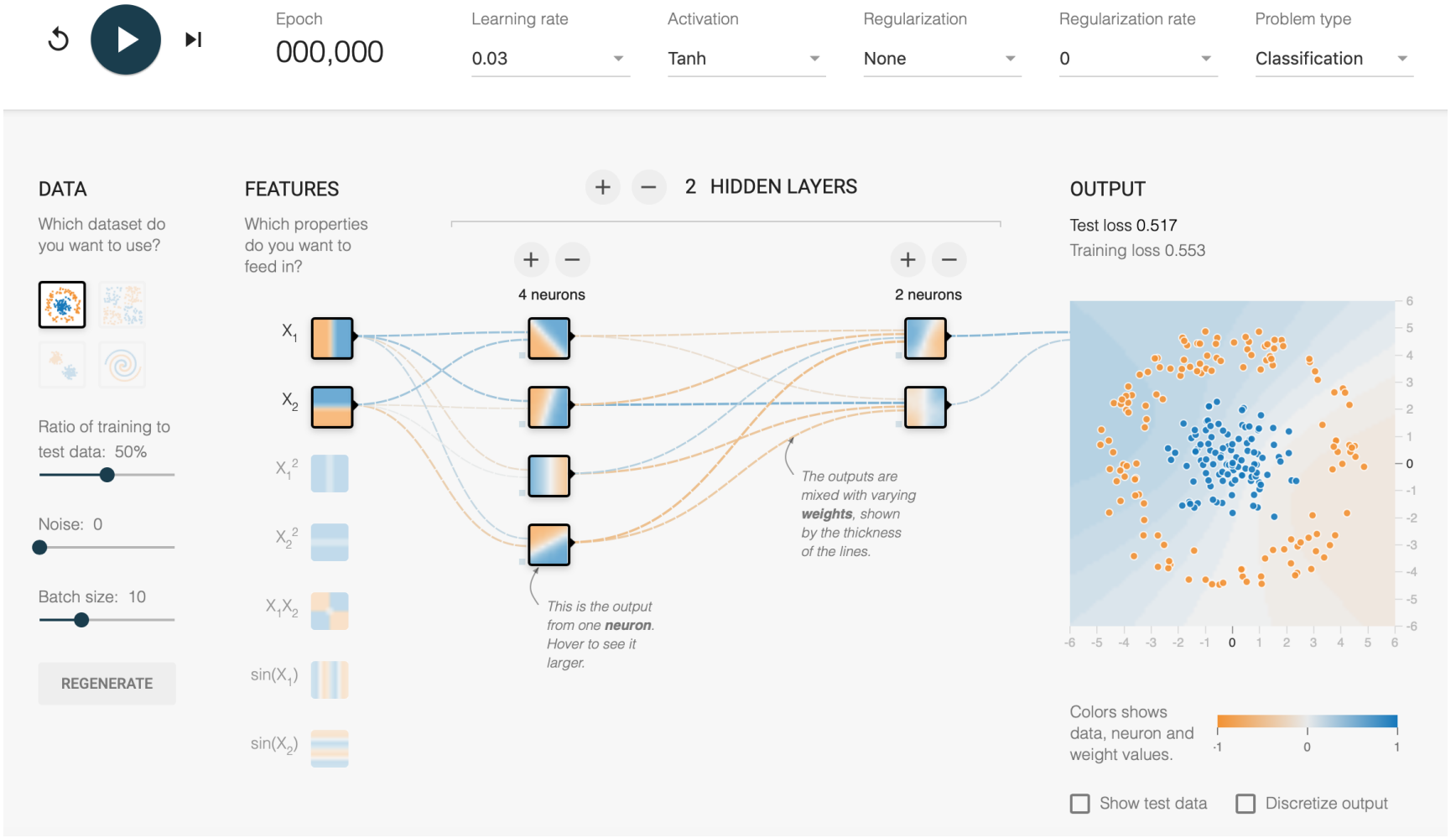
Which one is best ?



Demo

http://playground.tensorflow.org

Hyperparameters:
global search

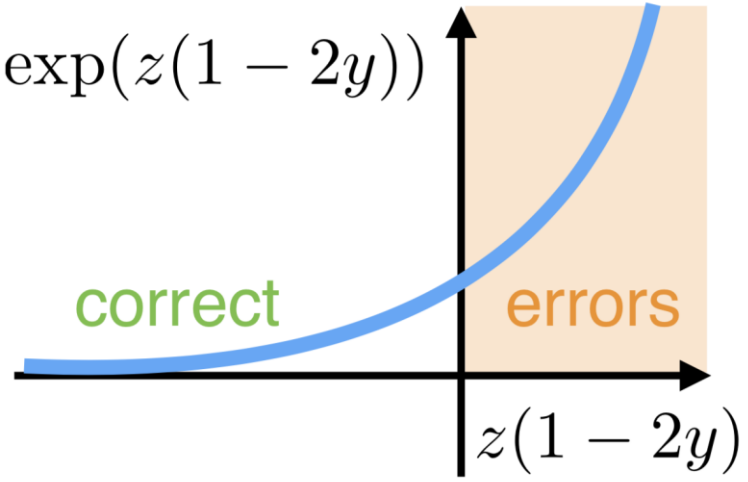


Model:
loss function

- chosen such that they have a non-flat region when the answer is incorrect

- *Exponential* or *logarithm* functions help

- Other functions :
  - $L_1(\hat{y}, y) = \sum_{i=0}^{m} \left| y^i - \hat{y}^i \right|$ (L1 loss)
  - $L_2(\hat{y}, y) = \sum_{i=0}^{m} \left(y^i - \hat{y}^i \right)^2$ (L2 loss)
  - …



Model: global architecture

DARTS: Differentiable Architecture Search

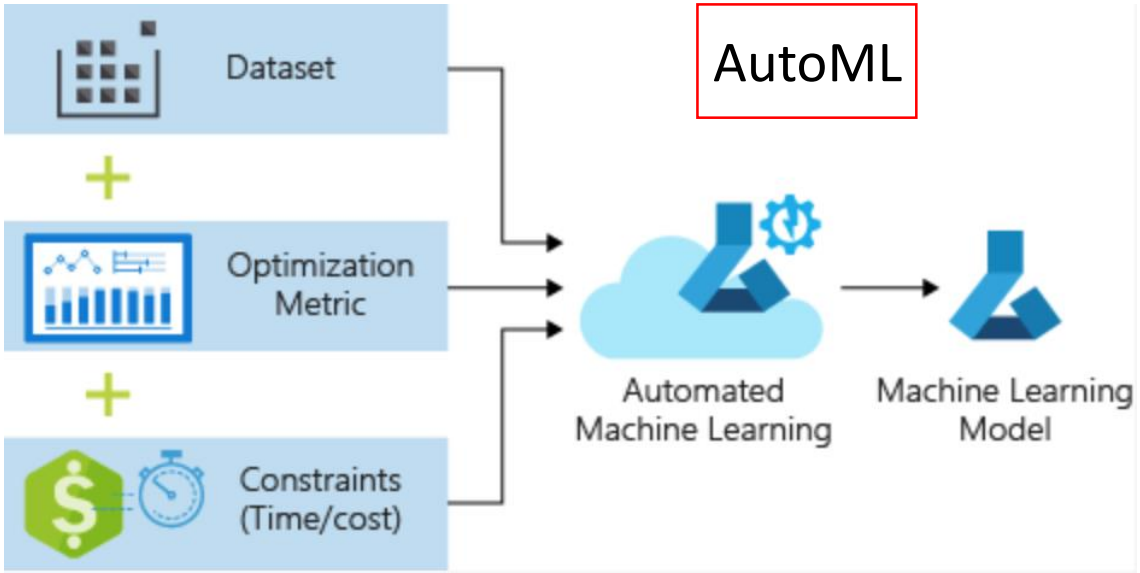
Hanxiao Liu*
CMU
hanxiaol@cs.cmu.com

Karen Simonyan
DeepMind
simonyan@google.com

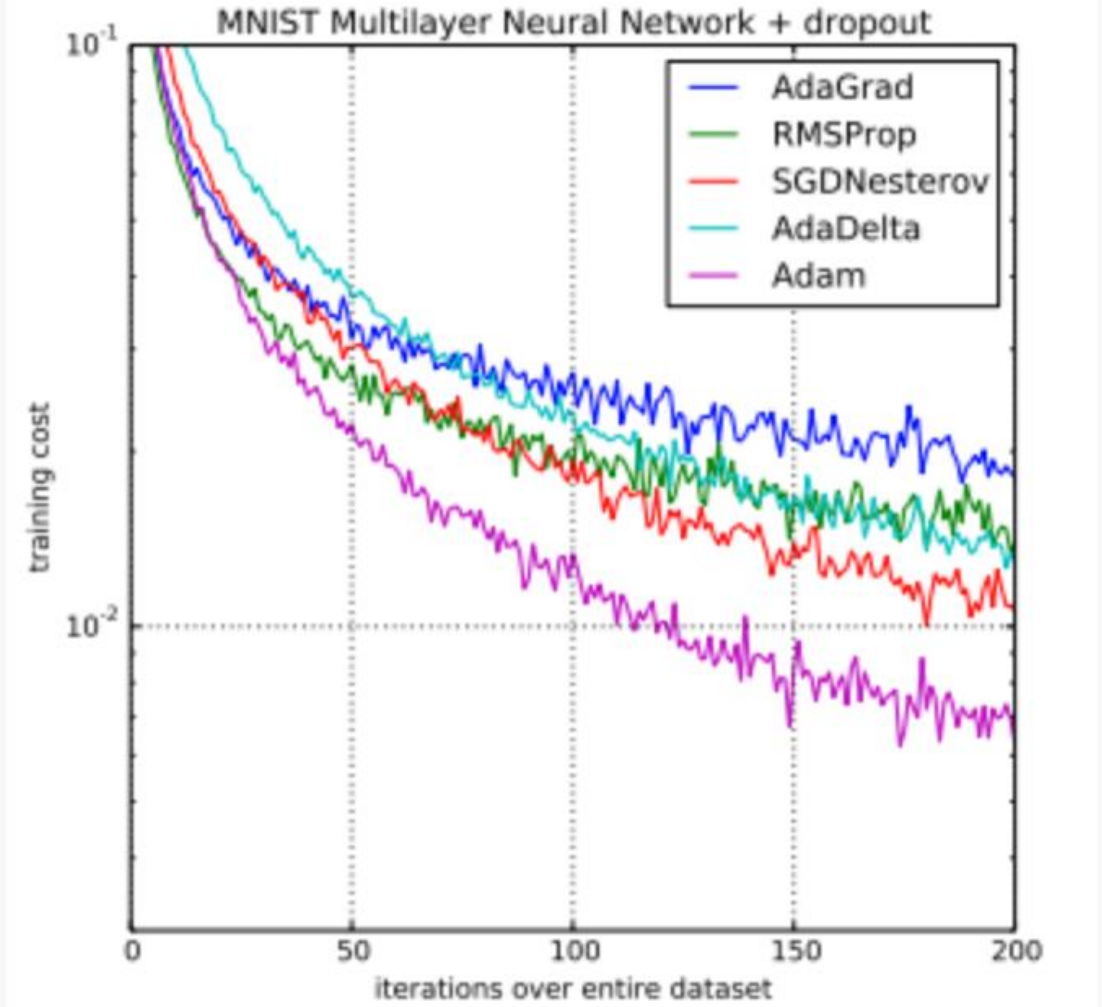
Yiming Yang
CMU
yiming@cs.cmu.edu

Abstract

This paper addresses the scalability challenge of architecture search by formulating the task in a differentiable manner. Unlike conventional approaches of applying evolution or reinforcement learning over a discrete and non-differentiable search space, our method is based on the continuous relaxation of the architecture representation, allowing efficient search of the architecture using gradient descent. Extensive experiments on CIFAR-10, ImageNet, Penn Treebank and WikiText-2 show that our algorithm excels in discovering high-performance convolutional architectures for image classification and recurrent architectures for language modeling, while being orders of magnitude faster than state-of-the-art non-differentiable techniques. Our implementation has been made publicly available to facilitate further research on efficient architecture search algorithms.



Optimization Algorithm (Reminder)



Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron

Taken from Adam: A Method for Stochastic Optimization, 2015.

https://www.youtube.com/watch?v=6JZNEb5uDu4

Two-Minute Papers

Variance Reduction techniques

Bigger training set

Regularization

Regularization

- Different strategies :

  1) Dataset (division, augmentation,…)

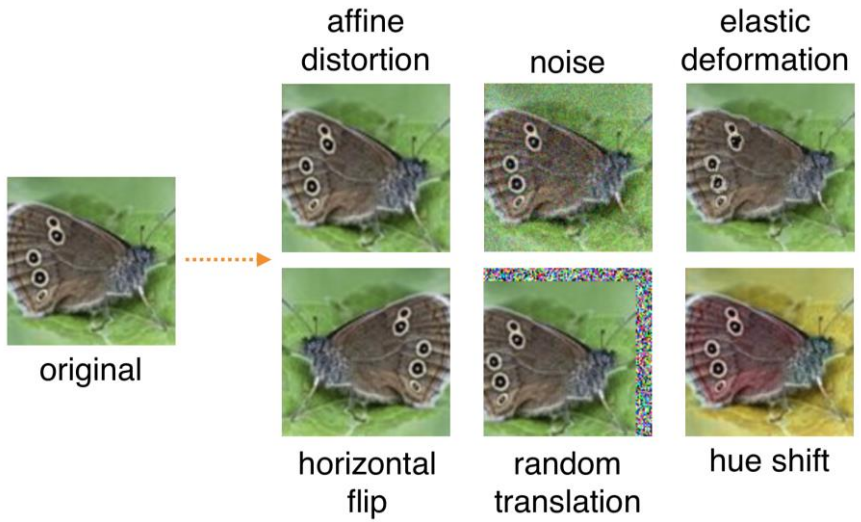
  2) Model (L2 regularization, dropout,…)

  3) Training (early stopping)

1. Regularization (dataset): Augmentation

- Apply realistic transformations to data to create new samples

Classification :



Regression :

1. **Original Data:**

Let's assume you have a dataset like this:

```makefile
Input: 2   Target: 10
Input: 5   Target: 25
Input: 8   Target: 40
```

- **Random Noise:** Add a small amount of random noise to your input data. For example:

```makefile
Input: 2.2   Target: 10
Input: 4.9   Target: 25
Input: 7.8   Target: 40
```

- **Scaling:** Multiply your input data by a small random factor:

```makefile
Input: 2.5   Target: 10
Input: 5.2   Target: 25
Input: 8.3   Target: 40
```

- **Translation:** Add a constant value to your input data:

```makefile
Input: 4   Target: 10
Input: 7   Target: 25
Input: 10  Target: 40
```

- **Combination:** Combine different augmentation techniques, like adding noise and scaling together.

Regularization (model): L2

- Modify the cost function (add soft constraint) :
  - L2-regularization cost = $\lambda * \Sigma(w^2)$

Cost function (regularized) = cost function + L2-regularization cost

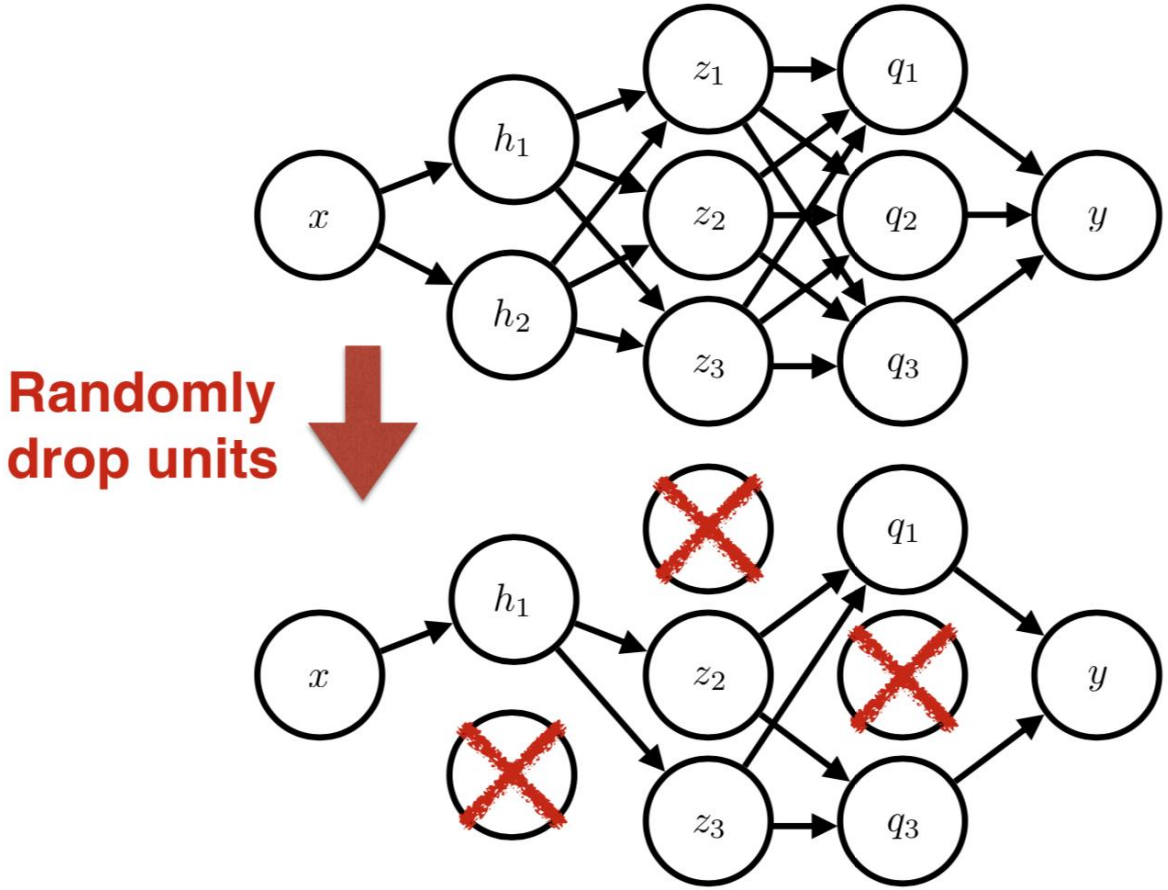
- λ regulates the complexity/capacity of the predictors. It smoothens the decision boundary

  → weights pushed to smaller values, preventing any weight from becoming too large and dominating the learning process

- Typical values : logarithmic scale between 0 and 0.1, such as *0.1, 0.001, 0.0001* (tuned using validation set)
  - If λ is too large, it can "oversmooth", resulting in a model with high bias

2. Regularization (model): Dropout

- Dropout randomly drops units during training

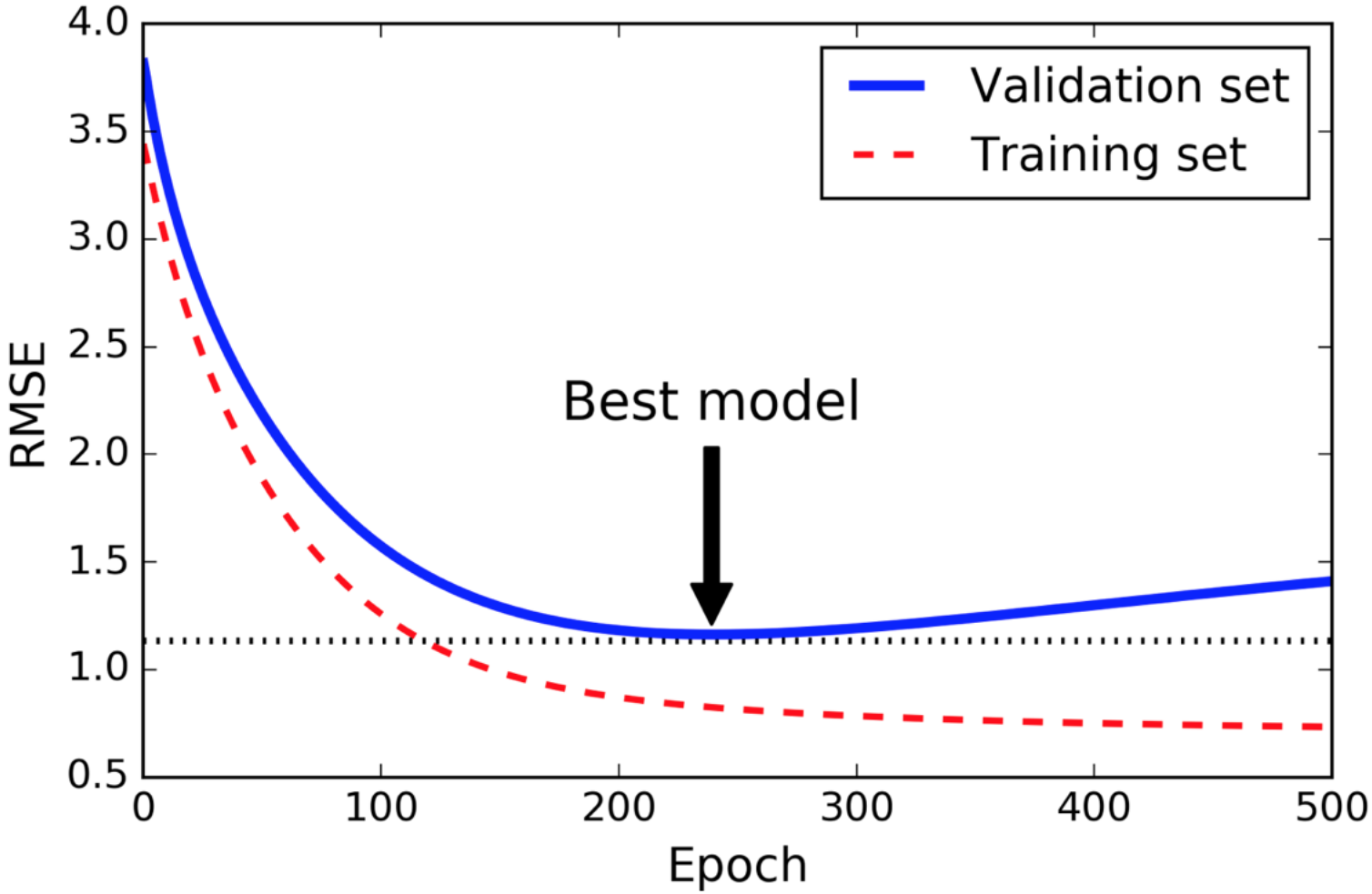


- During validation and test times, dropout is turned off

https://www.youtube.com/watch?v=LhhEv1dMpKE

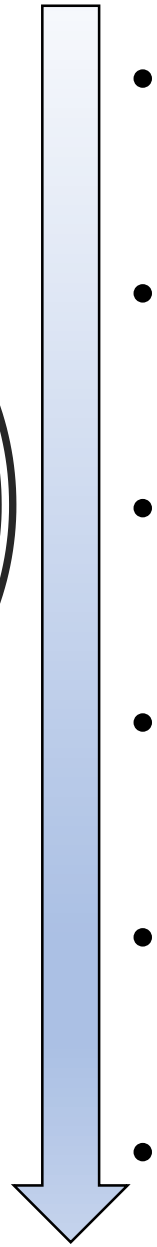
Two-Minute Papers

3. Regularization (training) : early stopping



- Stop training before the validation set error start growing

Summary of methods



- Bayes optimal error
- Human-level error
- Training error
- Val error
- Test error
- Real world

Bias reduction techniques :

- Hyperparameter tuning
- Model tuning
- Optimization algorithm
- decrease regularization

Variance reduction techniques :

- Bigger training set
- Regularization
- Hyperparam. and model tuning

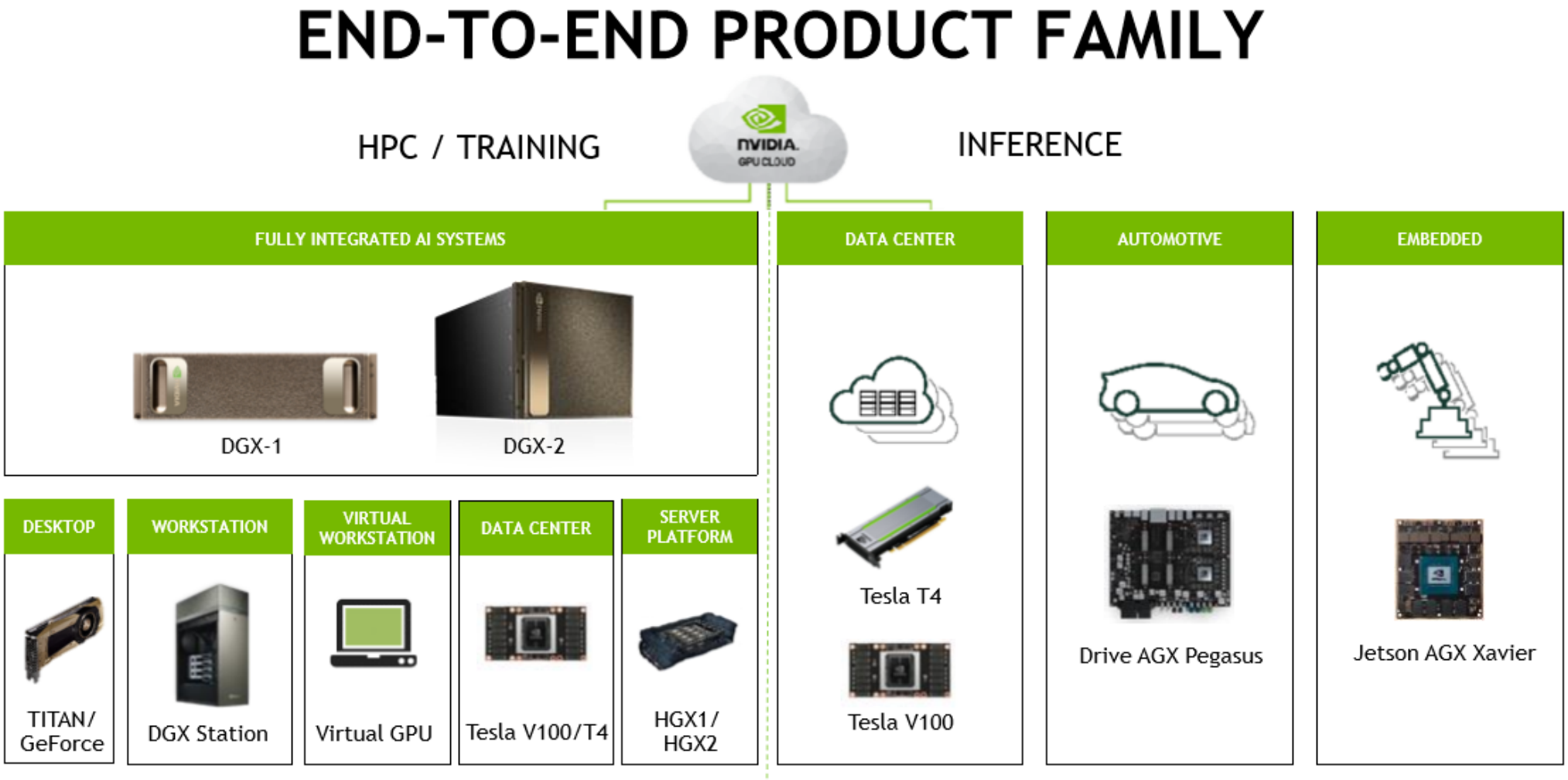
- bigger validation set

- change validation set
- change cost function

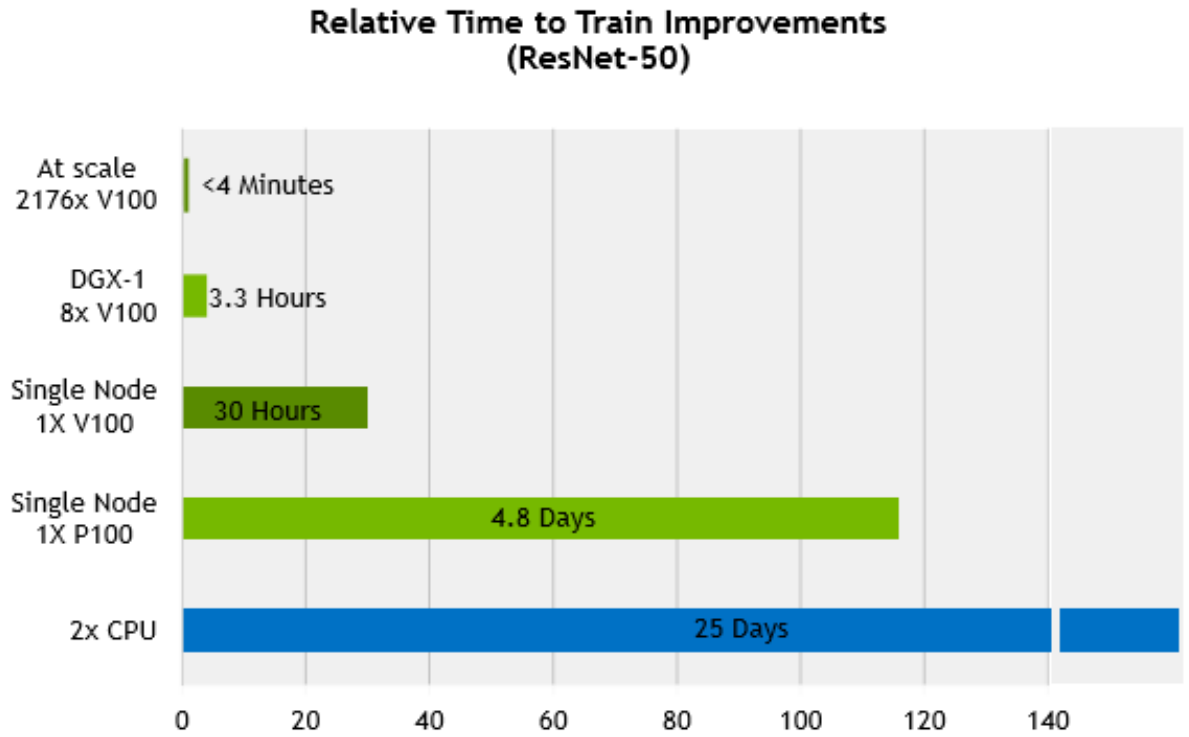
2. Time

How to improve time consumption when critical to get results

Material Acceleration (GPUs)



TESLA PLATFORM ENABLES DRAMATIC REDUCTION IN TIME TO TRAIN



ResNet-50, 90 epochs to solution | CPU Server: dual socket Intel Xeon Gold 6140
Sony 2176x V100 record on https://nnabla.org/paper/imagenet_in_224sec.pdf

Quiz

https://b.socrative.com/login/student/

Room : CONTI6128