

Deep Neural Networks

Bùi Tiến Lên

2022



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents



1. History
2. Architecture
3. Learning Problem
4. Workflow

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Notation

symbol	meaning	operator	meaning
$a, b, c, N \dots$	scalar number	w^T	transpose
$w, v, x, y \dots$	column vector	$X Y$	matrix
$X, Y \dots$	matrix	X^{-1}	multiplication
\mathbb{R}	set of real numbers	$X \odot Y$	inverse
\mathbb{Z}	set of integer numbers		an element-wise
\mathbb{N}	set of natural numbers		matrix-vector
\mathbb{R}^D	set of vectors		multiplication
$\mathcal{X}, \mathcal{Y}, \dots$	set		
\mathcal{A}	algorithm		



History

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Ups & Downs of Neural Networks

NNs and AI have experienced several hype cycles, followed by disappointment, criticism, and funding cuts:

- **1950s - 70s:** Golden years of AI (funded by DARPA): solve algebra, play chess & checkers, reasoning, semantic nets “within ten years a digital computer will be the world’s chess champion”
- **1969:** shown that XOR problem cannot be solved by Perceptron (led to the invention of multi-layer networks later on)
- **Mid 1970s:** Chain reaction that begins with pessimism in the AI community, followed by pessimism in the press, followed by a severe cutback in funding, followed by the “end” of serious research (“AI winter”)
- **1980s:** Governments (starting in Japan) and industry provide AI with billions of dollars. Boom of “expert systems”.



Ups & Downs of Neural Networks (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

- **1986:** Backpropagation had been invented in the 1970s, but only 1986 it became popular through a famous paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams. It showed that also complex functions became solvable through NNs by using multiple layers.
- **Late 1980s:** Investors - despite actual progress in research - became disillusioned and withdrew funding again.

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Why no Deep Learning in the 1980s?

Neural Networks could not become “deep” yet - because:

- Computers were slow. So the neural networks were tiny and could not achieve (the expected) high performance on real problems.
- Datasets were small. There were no large datasets that had enough information to constrain the numerous parameters of (hypothetical) large neural networks.
- Nobody knew how to train deep nets. Today, object recognition networks have > 25 successive layers of convolutions. In the past, everyone was very sure that such deep nets cannot be trained. Therefore, networks were shallow and did not achieve good results.



History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Ups & Downs of Neural Networks

- **1991:** Hornik proved 1 hidden layer network can model any continuous function (universal approximation theorem)
- **1991-1992:** Vanishing Gradient, problem in multi-layer networks where training in front layers is slow due to backpropagation diminishing the gradient updates through the layers. Identified by Hochreiter & Schmidhuber who also proposed solutions.
- **1990s - mid 2000s:** Due to lack of computational power, interest in NNs decreased again and other Machine Learning models, such as Bayesian models, Decision Trees and Support Vector Machines became popular.



History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Resurrection of Deep Learning

- **2000s:** Hinton, Bengio and LeCun (“The fathers of the age of deep learning”) join forces in a project. They overcome some problems that caused deep networks not to learn anything at all
 - **2006:** Breakthrough with Layer-wise pre-training by unsupervised learning (using RBMs)
 - **2010s - present:** Important new contributions:
 - Simpler initialization (without pre-training)
 - Dropout
 - Simpler activations: Rectifier Units (ReLUs)
 - Batch Normalization
- not a re-invention of NNs but paved the way for very deep NNs



Architecture

- Activation functions
- Why Deep?



Deep Learning vs. Classical Machine Learning

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

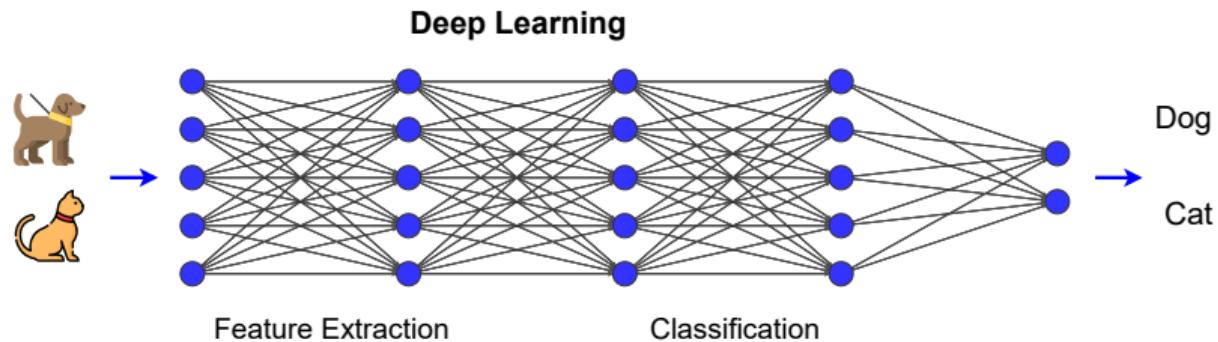
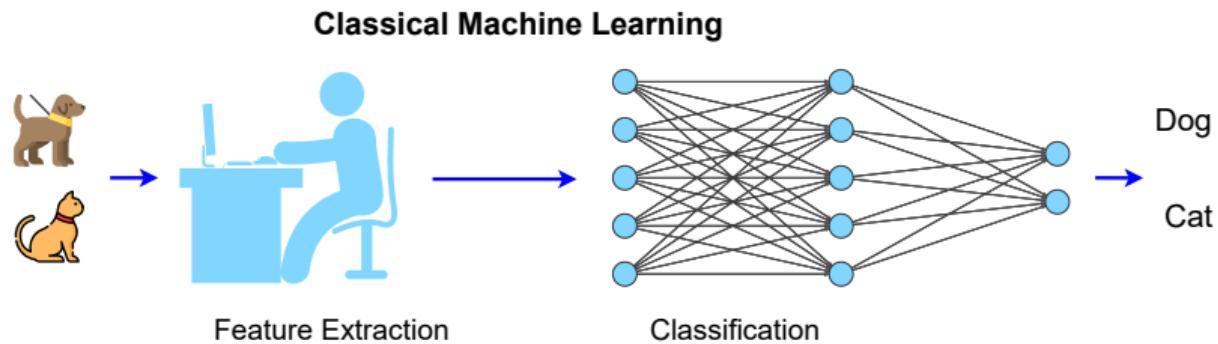
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning





Deep neural network (DNN)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

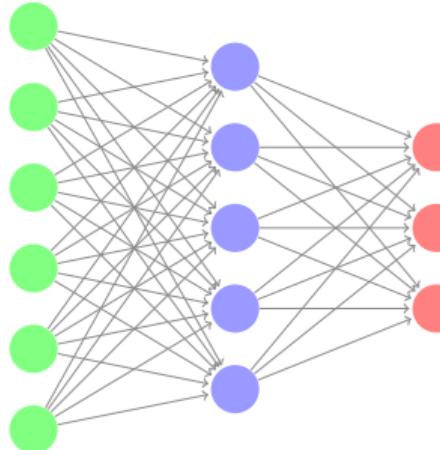
Choose Classifier

Train Classifier

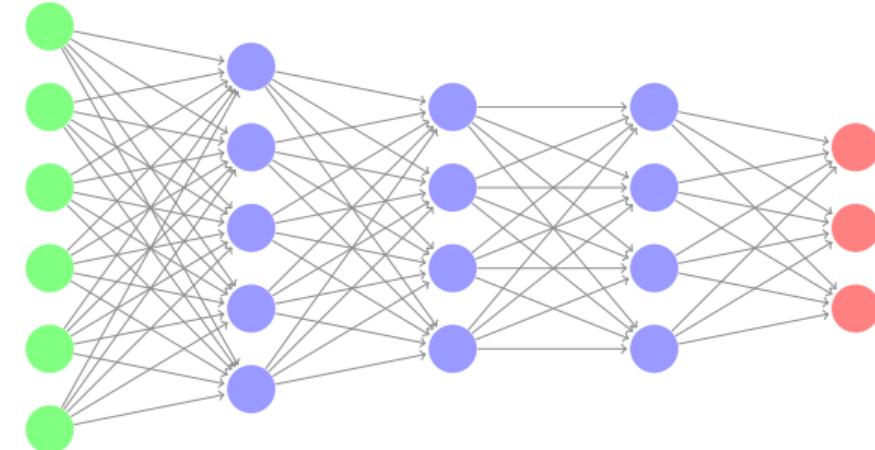
Explainable Machine
Learning

- Deep neural networks (DNNs) are basically ANNs with multiple hidden layers between the input and output layers.

Shallow Neural Network



Deep Neural Network





Activation functions

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

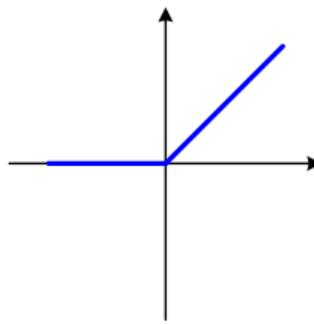
Prepare Data

Choose Metric

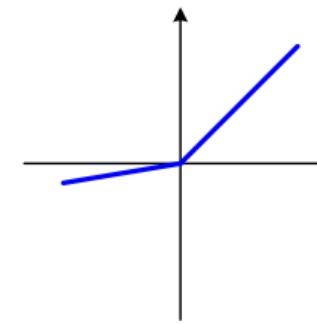
Choose Classifier

Train Classifier

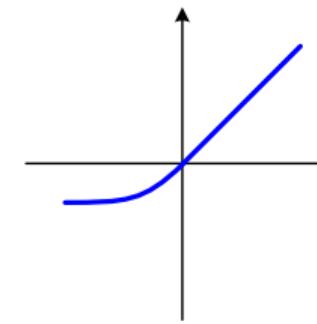
Explainable Machine
Learning



ReLU



Leaky ReLU



ELU



Activation functions (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

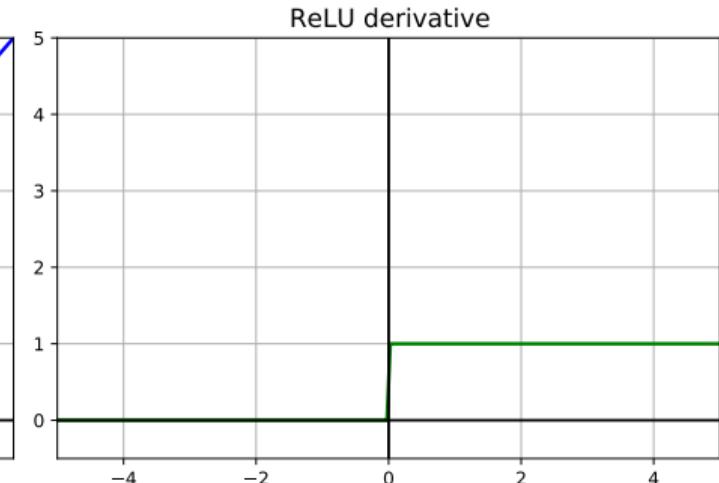
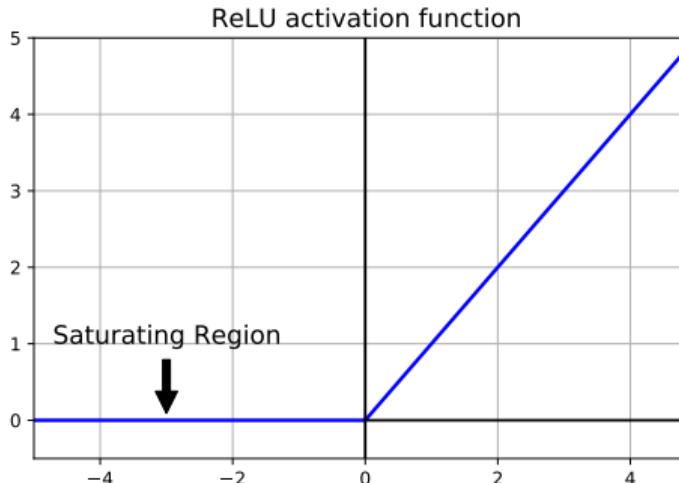
Choose Classifier

Train Classifier

Explainable Machine
Learning

- ReLU function

$$a = \text{ReLU}(x) = \max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (1)$$





Activation functions (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

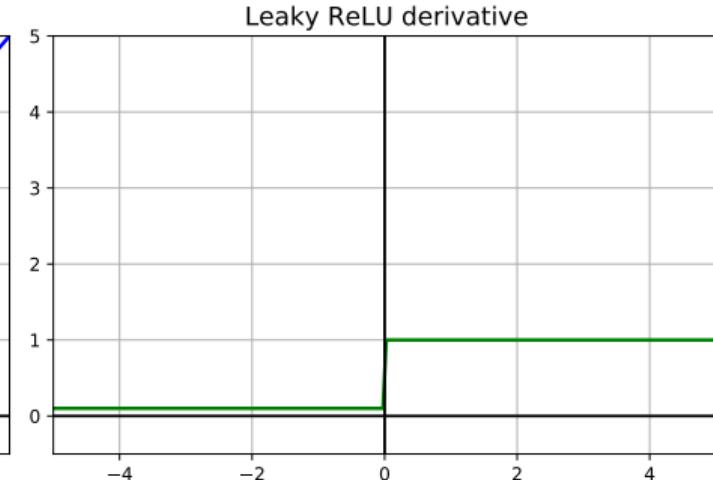
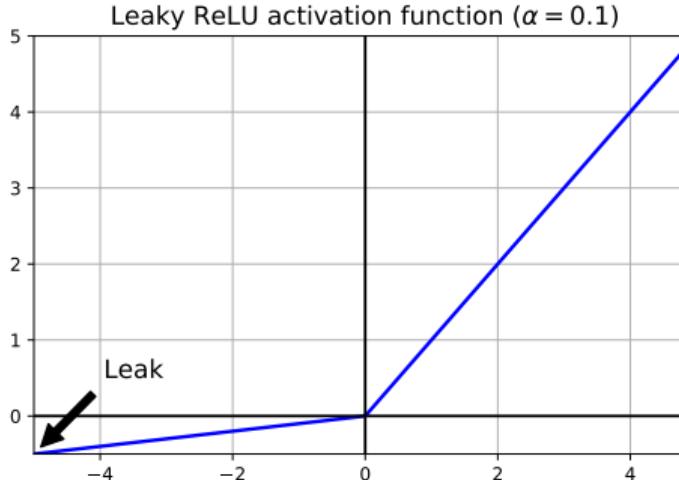
Choose Classifier

Train Classifier

Explainable Machine
Learning

- Leaky ReLU function

$$a = \text{LeakyReLU}(x) = \max(\alpha x, x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases} \quad (2)$$





Activation functions (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

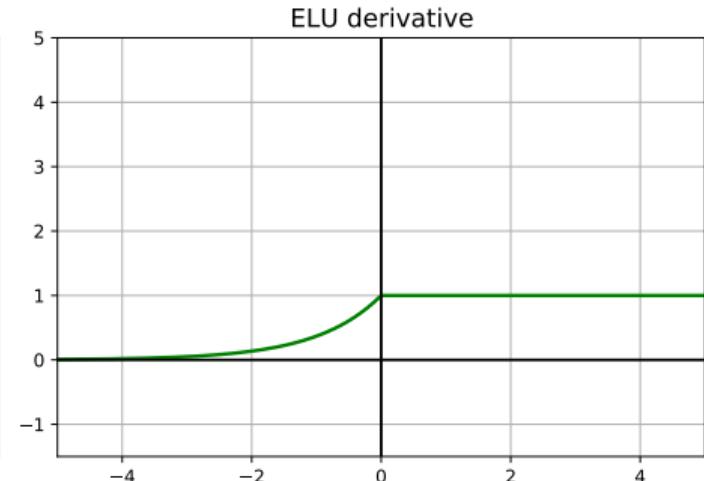
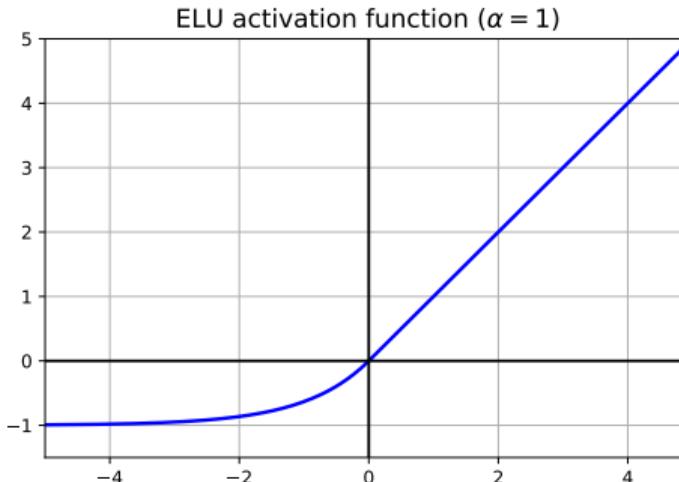
Choose Classifier

Train Classifier

Explainable Machine
Learning

- ELU function

$$a = \text{ELU}(x) = \begin{cases} x & x \geq 0 \\ \alpha(\exp(x) - 1) & x < 0 \end{cases} \quad (3)$$





Maxout Gate

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

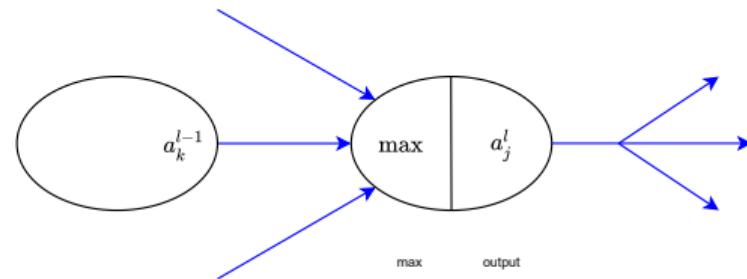
Choose Classifier

Train Classifier

Explainable Machine
Learning

- The maxout unit takes a group of input activations and output the maximum value

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_k) = \max_i(x_i) \quad (4)$$



[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Why Deep?

Experiment

- Deep networks are empirically better in solving many real-world problems
- Deep networks is providing breakthrough results in speech recognition and image classification

Network	Error	Layers
AlexNet	16.0%	8
ZFNet	11.2%	8
VGGNet	7.3%	19
GoogLeNet	6.7%	22
MS ResNet	3.6%	152!





History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Why Deep? (cont.)

Analogy

Logic circuits

- Logic circuits consists of **gates**
- **A two layers** of logic gates can represent any Boolean function.
- Using multiple layers of logic gates to build some functions are much **simpler**

→ less gates needed

Neural network

- Neural network consists of **neurons**
- **A hidden layer** network can represent any continuous function.
- Using multiple layers of neurons to represent some functions are much **simpler**

→ less parameters



Why Deep? (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

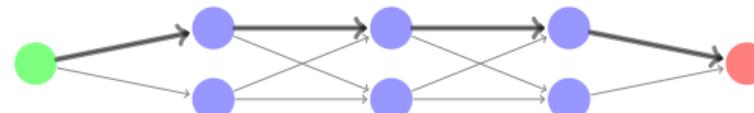
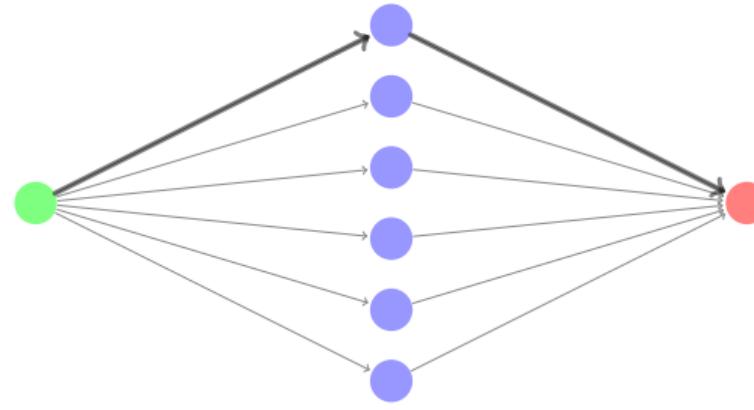
Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Another reason





Learning with DNNs

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

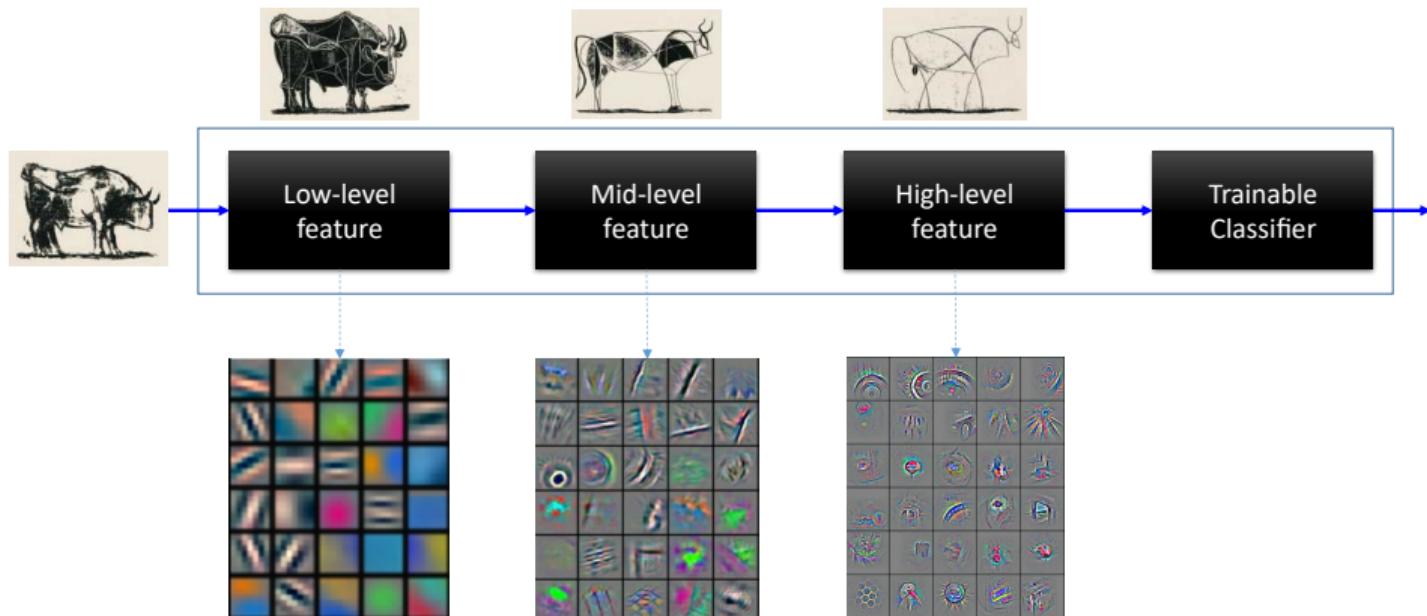
Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

- **Feature hierarchies** (feature engineering) are learnt such that features from higher levels formed by the composition of lower level features.





Learning with DNNs (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

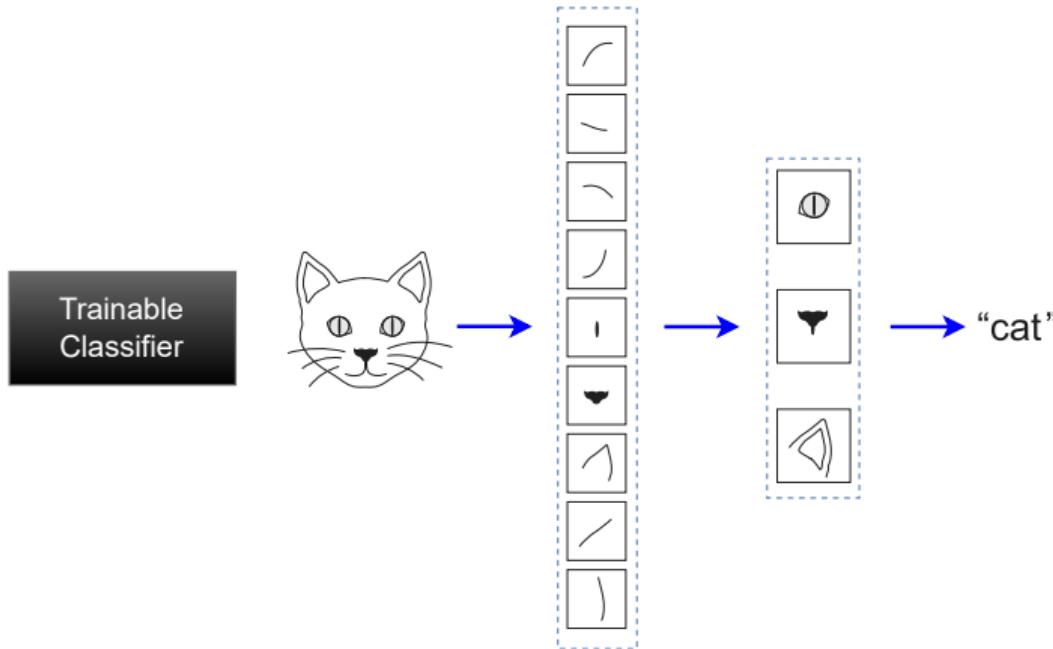
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning





Learning with DNNs (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

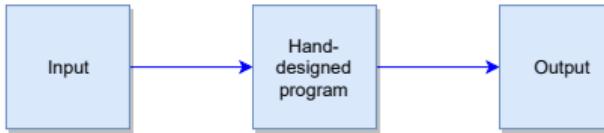
Choose Metric

Choose Classifier

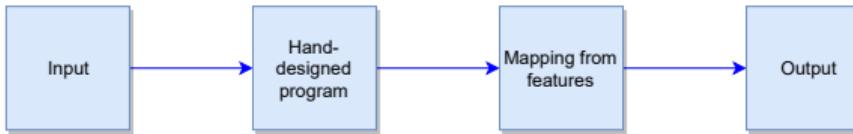
Train Classifier

Explainable Machine
Learning

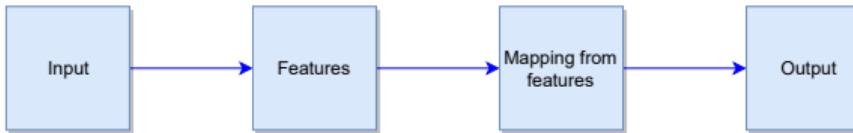
Rule-based systems



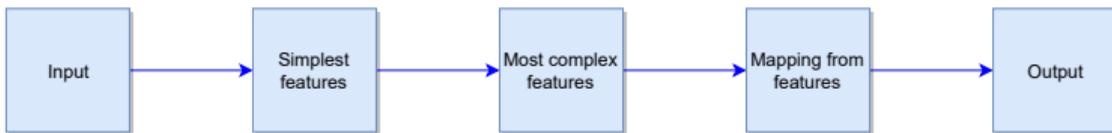
Classic machine learning



Representation learning



Deep learning





Learning Multiple Levels of Abstraction

- The big payoff of deep learning is to allow learning higher levels of abstraction
- Higher-level abstractions disentangle the factors of variation, which allows much easier generalization and transfer





Types of Deep neural networks

- Convolutional neural network (CNN)
- Recurrent Neural Networks (RNN)
- Autoencoders
- Generative Adversarial Nets (GAN)
- Region Based CNN (R-CNN)
- YOLO
- RNN ConvNet
- Attention Models
- Transformer Models

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning



Learning Problem

- Vanishing gradient problem
- Nonsaturating Activation Functions
- Dropout
- Batch Normalization



Why are deep neural networks hard to train?

History

Architecture

Activation functions

Why Deep?

Learning Problem

Vanishing gradient problem

Nonsaturating Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

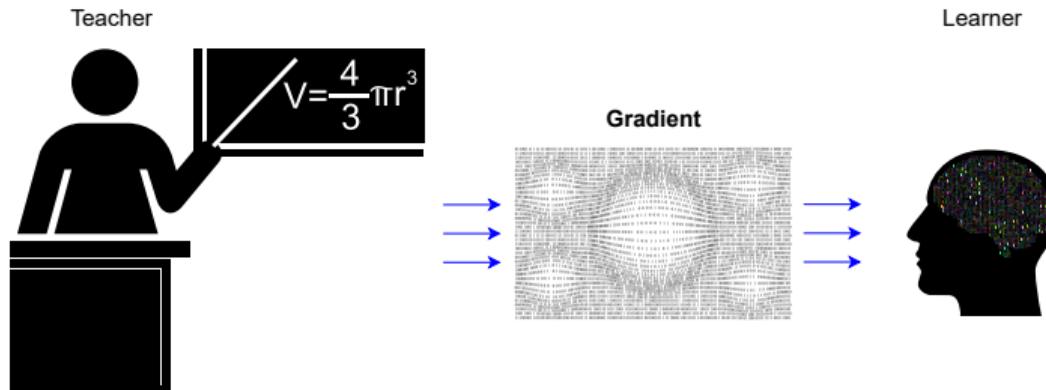
Choose Classifier

Train Classifier

Explainable Machine Learning

Learning

- There's an **intrinsic instability** associated to learning by **gradient descent** in many-layer (deep) neural networks
- This instability tends to result in either the early or the later layers getting stuck during training
 - The **vanishing gradient problem** and
 - The opposite of this problem is called the **exploding gradient problem**





The vanishing gradient problem

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

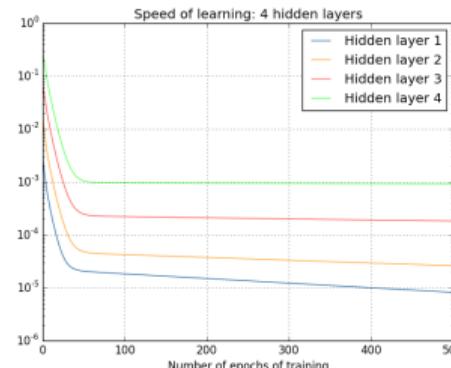
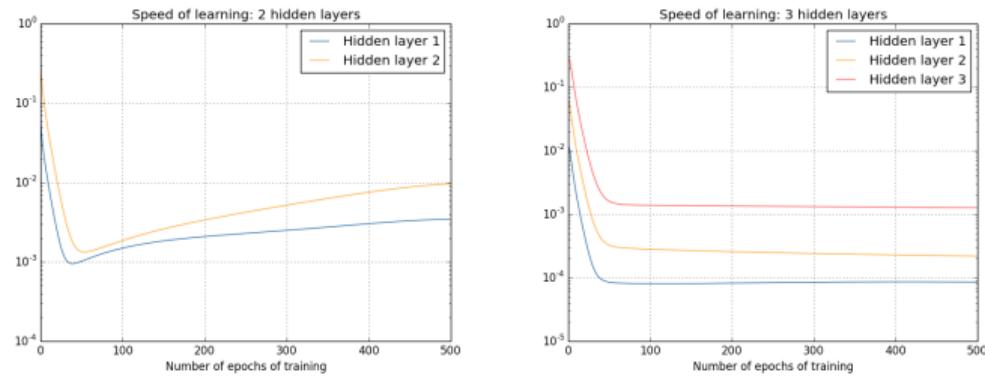
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning





What's causing the vanishing gradient problem?

History

Architecture

Activation functions

Why Deep?

Learning Problem

Vanishing gradient problem

Nonsaturating Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

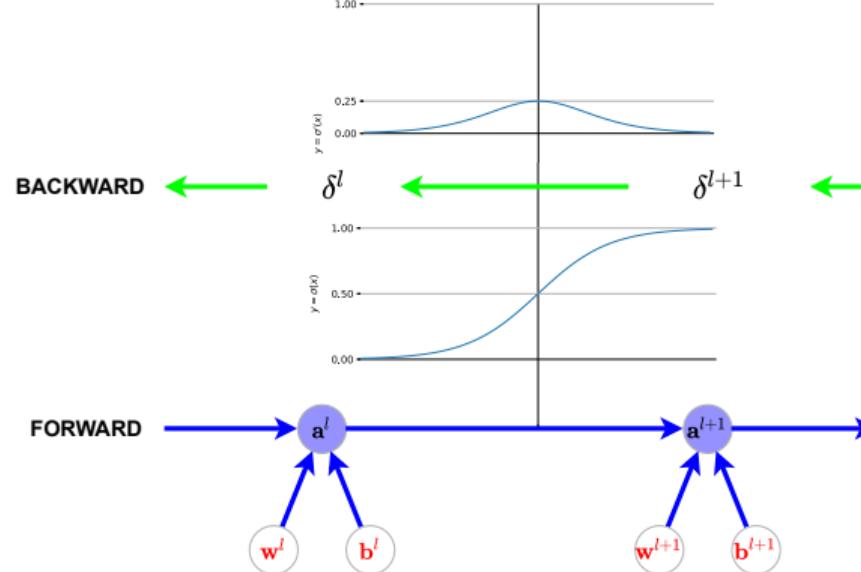
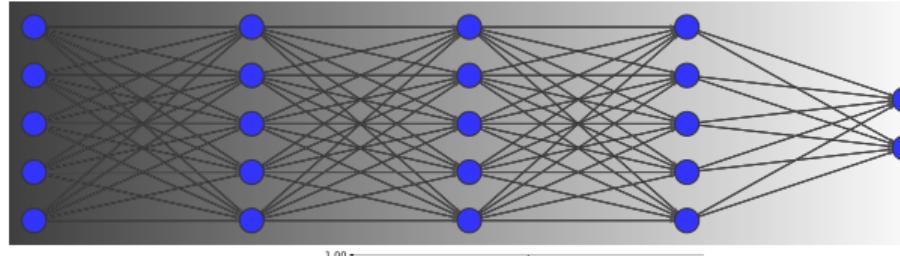
Choose Metric

Choose Classifier

Train Classifier

Explainable Machine Learning

Explaining



[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

ReLU

Rectified linear units (ReLU) allow for faster and effective training of DNNs on large and complex datasets.

- **Biological plausibility**

- One-sided, compared to the anti-symmetry of tanh

- **Sparse activation**

- E.g., in a randomly initialized network, only about 50% of hidden units are activated (having a non-zero output)

- **Better gradient propagation**

- Fewer vanishing gradient problems compared to sigmoid activation functions that saturate in both directions

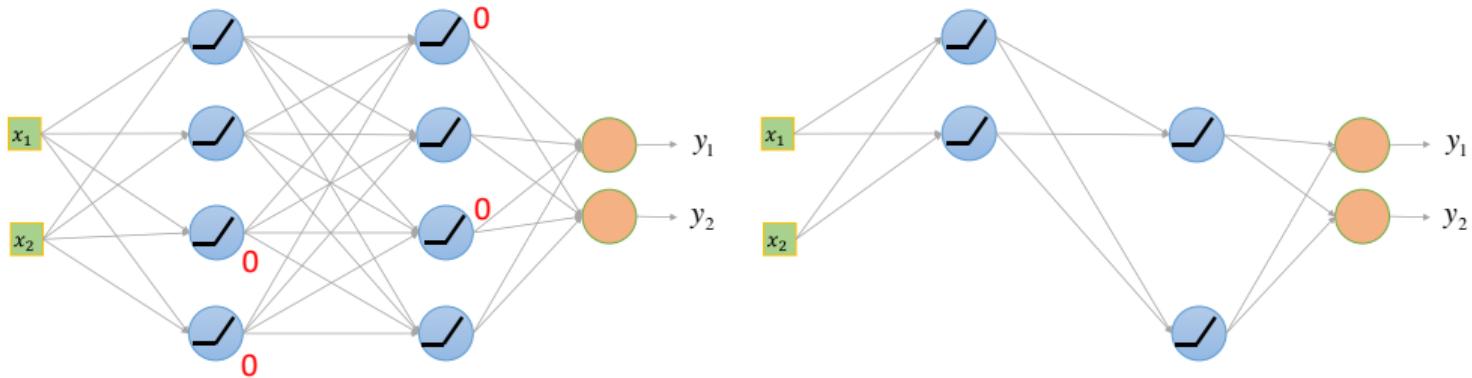
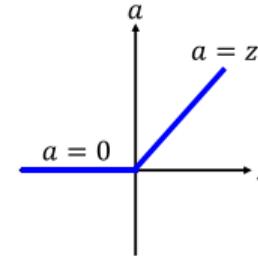
- **Efficient computation:** Only comparison

- **Scale-invariant**

$$\forall x, \max(0, ax) = a \max(0, x)$$



How does a ReLU work?





ReLU: Potential issues

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

- Non-differentiable at zero
 - However, it is differentiable anywhere else, including points arbitrarily close to (but not equal to) zero.
- Non-zero centered
- Unbounded
- Dying ReLU problem



Dropout

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

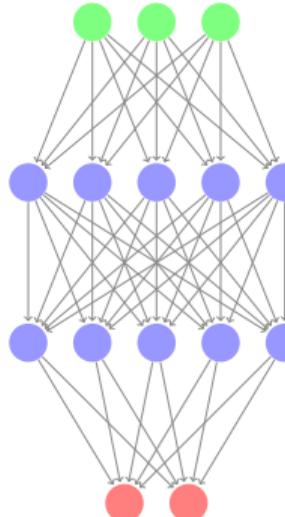
Prepare Data

Choose Metric

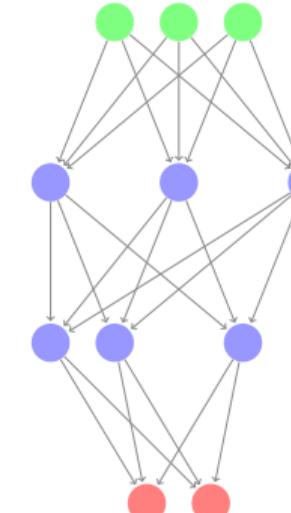
Choose Classifier

Train Classifier

Explainable Machine
Learning



Standard neural network



After applying dropout



Training Procedure

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

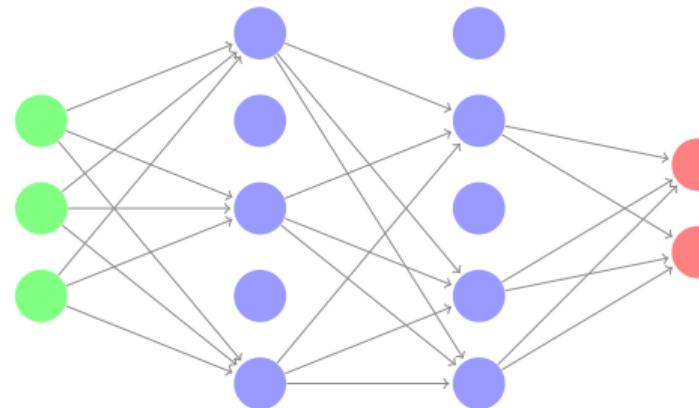
Train Classifier

Explainable Machine
Learning

For each mini-batch

- Start by randomly (and temporarily) deleting $p\%$ the hidden neurons in the network, while leaving the input and output neurons untouched
- We forward-propagate the input through the modified network, and then back-propagate the result to update the appropriate weights and biases

Each neuron has $p\%$ to dropout





Testing Procedure

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

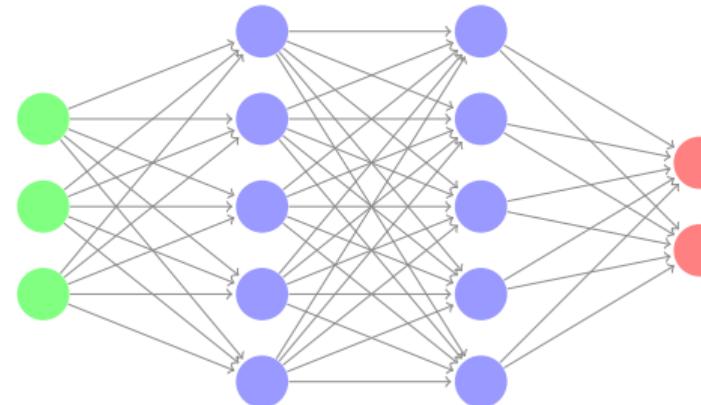
Train Classifier

Explainable Machine
Learning

Using original neural network, no dropout

- If the dropout rate at training is $p\%$, all the weights times $(1 - p)\%$; e.g., assume that the dropout rate is $p = 40\%$, if a weight $w_{train} = 1$ by training, set $w_{test} = 0.6$ for testing

$$w_{test} \leftarrow w_{train} \times (1 - p)$$





Why it works

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

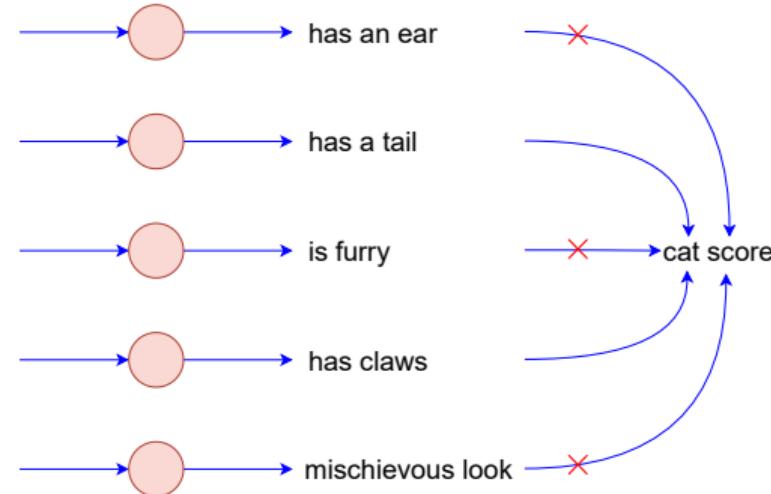
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning





Why it works (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

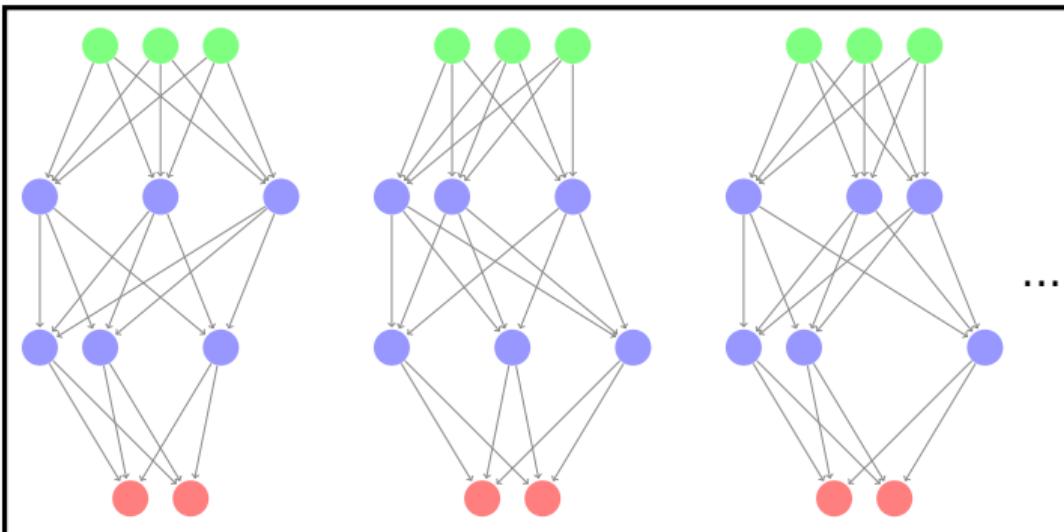
Choose Classifier

Train Classifier

Explainable Machine
Learning

- It can also be seen as training a large ensemble of models that share parameters

Ensemble of networks



Strong network

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)

Batch Normalization

- **Batch normalization** reparametrizes the model to make artificial neural networks faster and more stable
- **Key idea:** normalize, shift and rescale data
- **Learning**

Input: Values of x over a **training set** mini-batch, each of size m : $\mathcal{B} = \{x^{(1..m)}\}$

Parameters to be learned: γ, β

Output: $\{y^{(i)} = \text{BN}_{\gamma, \beta}(x^{(i)})\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x^{(i)} \quad (\text{mini-batch mean})$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu_{\mathcal{B}})^2 \quad (\text{mini-batch variance})$$

$$\hat{x}^{(i)} \leftarrow \frac{x^{(i)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad (\text{normalize})$$

$$y^{(i)} \leftarrow \gamma \hat{x}^{(i)} + \beta \quad (\text{scale and shift})$$

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Batch Normalization (cont.)

- Inference

Input: x

Output: y

$$E(x) \leftarrow E_{\mathcal{B}}[\mu_{\mathcal{B}}]$$

$$\text{Var}(x) \leftarrow \frac{m}{m-1} E_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

$$y = \frac{\gamma}{\sqrt{\text{Var}(x) + \epsilon}} x + \left(\beta - \frac{\gamma E(x)}{\sqrt{\text{Var}(x) + \epsilon}} \right)$$



Batch Normalization (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

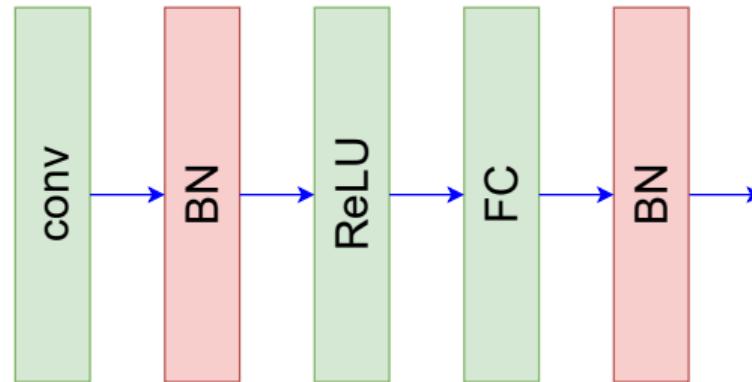
Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

- **Common configuration:** insert BN layers right after Conv or FC layers, before ReLU layers





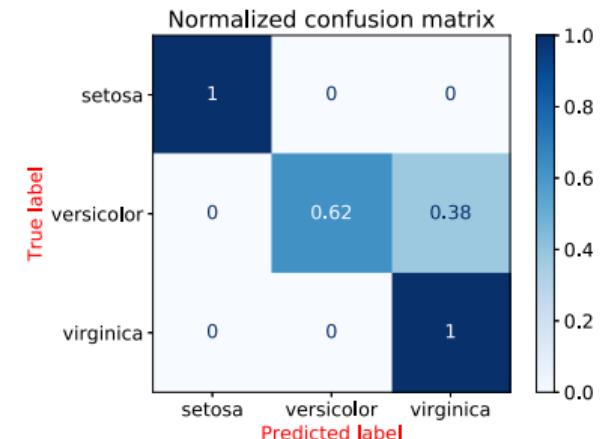
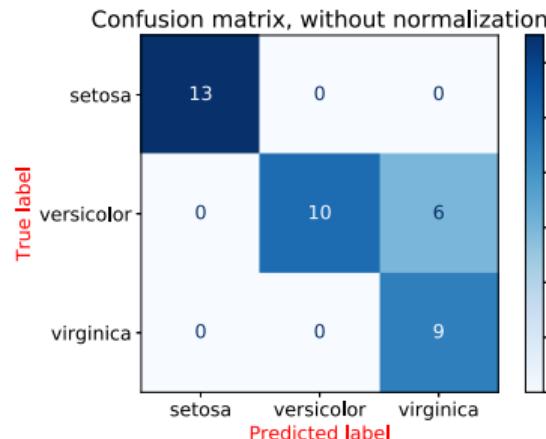
Workflow

- Metrics
- Workflow
- Prepare Data
- Choose Metric
- Choose Classifier
- Train Classifier
- Explainable Machine Learning

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Confusion matrix

- Each **column** of the matrix represents the instances in a **predicted class** while each **row** represents the instances in an **actual class**





Two-class confusion matrix

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Consider two-class problem with two classes \oplus and \ominus

- Number of **true positives** TP
- Number of **true negatives** TN
- Number of **false positives** FP
- Number of **false negatives** FN
- The number of **actual positives** $P = TP + FN$
- The number of **actual negatives** $N = TN + FP$

	Predicted \oplus	Predicted \ominus
Actual \oplus	TP	FN
Actual \ominus	FP	TN

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Example

- Let's assume we are building a binary classification to classify **cat/non-cat**(⊕/⊖) images. And let's assume our test set has 1100 images (1000 non-cat images, and 100 cat images), with the below confusion matrix.

	Predicted cat	Predicted non-cat
Actual cat	90	10
Actual non-cat	60	940



Classification related metrics

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Concept 1

Accuracy (acc) is the number of correct predictions divided by the total number of predictions.

$$acc = \frac{TP + TN}{P + N} \quad (5)$$

Error rate (err) is defined as

$$err = 1 - acc \quad (6)$$

$$acc = \frac{90 + 940}{1000 + 100} = 93.6\% \text{ and } err = 1 - acc = 6.4\%$$



Classification related metrics (cont.)

- Accuracy is considered as a weighted average
- There are many cases in which classification accuracy is not a good indicator of your model performance. One of these scenarios is when your class distribution is imbalanced

Concept 2

Precision (pre) is defined as

$$pre = \frac{TP}{TP + FP} \quad (7)$$

$$pre_{\text{cat}} = \frac{90}{90 + 60} = 60\% \text{ and } pre_{\text{non-cat}} = \frac{940}{940 + 10} = 98.9\%$$



Classification related metrics (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Concept 3

Sensitivity (*recall, hit rate, or true positive rate tpr*) is defined as the fraction of samples from class \oplus which are correctly predicted by the model

$$tpr = \frac{TP}{P}. \quad (8)$$

Specificity (*selectivity or true negative rate tnr*) is defined as the fraction of samples from class \ominus which are correctly predicted by the model

$$tnr = \frac{TN}{N} \quad (9)$$

$$tpr_{\text{cat}} = \frac{90}{100} = 90\% \text{ and } tpr_{\text{non-cat}} = \frac{940}{1000} = 94\%$$



Classification related metrics (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

- Depending on application, you may want to give higher priority to *sensitivity* or *precision*.
- But there are many applications in which both *sensitivity* and *precision* are important.

Concept 4

F1-score (F_1) is defined as the **harmonic mean** of *precision* and *sensitivity*

$$F_1 = \frac{2 \times \text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}} \quad (10)$$

$$F_{1\text{cat}} = \frac{2 \times 0.6 \times 0.9}{0.6 + 0.9} = 72\%$$



Classification related metrics (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Concept 5

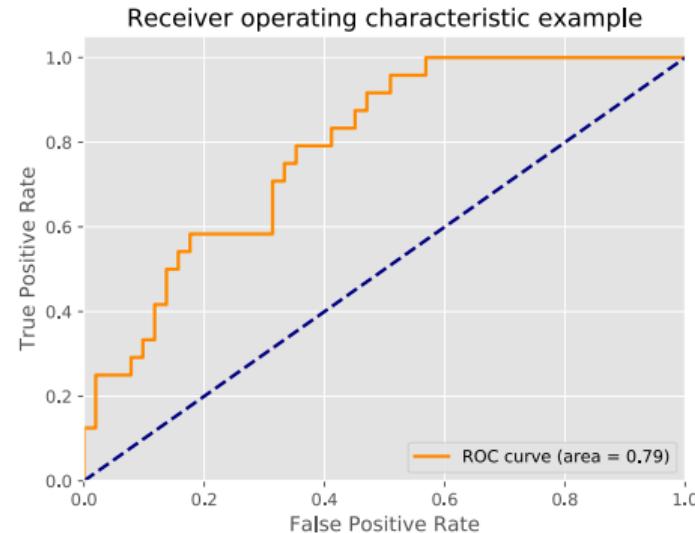
Fall-out or false positive rate (fpr) is defined as

$$fpr = \frac{FP}{N} \quad (11)$$

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

ROC Curve

- The receiver operating characteristic (*ROC*) curve is plot which shows the performance of a binary classifier as function of its cut-off threshold.
- It essentially shows the *true positive rate (tpr)* against the *false positive rate (fpr)* for various threshold values.
- The area under the curve (*AUC*) is an aggregated measure of performance.



[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Regression related metrics

Concept 6

“Mean squared error” (MSE) is the average squared error between the predicted and actual values.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Workflow

Important Question: How well a system will do on new, unseen data?

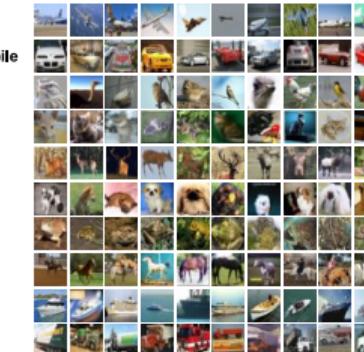
The workflow includes

1. Selecting features and collecting data.
2. Choosing a performance metric.
3. Choosing a classifier and optimization algorithm.
4. Evaluating the performance of the model.
5. Tuning the algorithm.

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Data

- Collecting data
- Artificially augmenting the data
- Splitting the data into three independent sets: **training**, **validation**, and **test**
 - We *train* on *the training set* and *evaluate* our model on *the validation set*. Once our model is ready for prime time, we *test* it one final time on *the test set*.
 - An *essential* rule of the training and testing process is that *we never learn from the test data.*



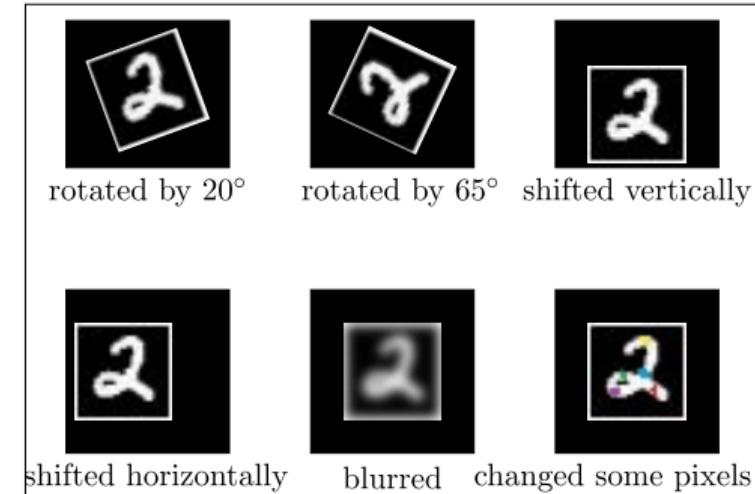
[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Data augmentation

- Geometric: flipping, rotation, shearing, multiple crops
- Photometric: color transformations
- Other: add noise, compression artifacts, lens distortions, etc.



label = 2



label = 2

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Data preprocessing

- Zero centering

- Subtract *mean image* – all input images need to have the same resolution
- Subtract *per-channel means* – images don't need to have the same resolution

- Optional: rescaling – divide each value by (per-pixel or per-channel) standard deviation

- Be sure to apply the same transformation at training and test time!
 - Save training set statistics and apply to test data

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Performance metric

Identify Needs

- High accuracy or low accuracy?
 - Surgery robot: high accuracy
 - Celebrity look-a-like app: low accuracy

Choose Metrics

- Accuracy? (% of examples correct)
- Coverage? (% of examples processed)
- Precision? (% of detections that are right)
- Recall? (% of objects detected)
- Amount of error? (For regression problems)

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

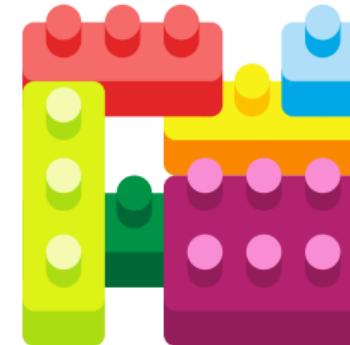
Classifier

Deep or Not?

- Lots of noise, little structure → not deep
 - Logistic regression, SVM, boosted tree are all good
- Little noise, complex structure → deep

What Architecture?

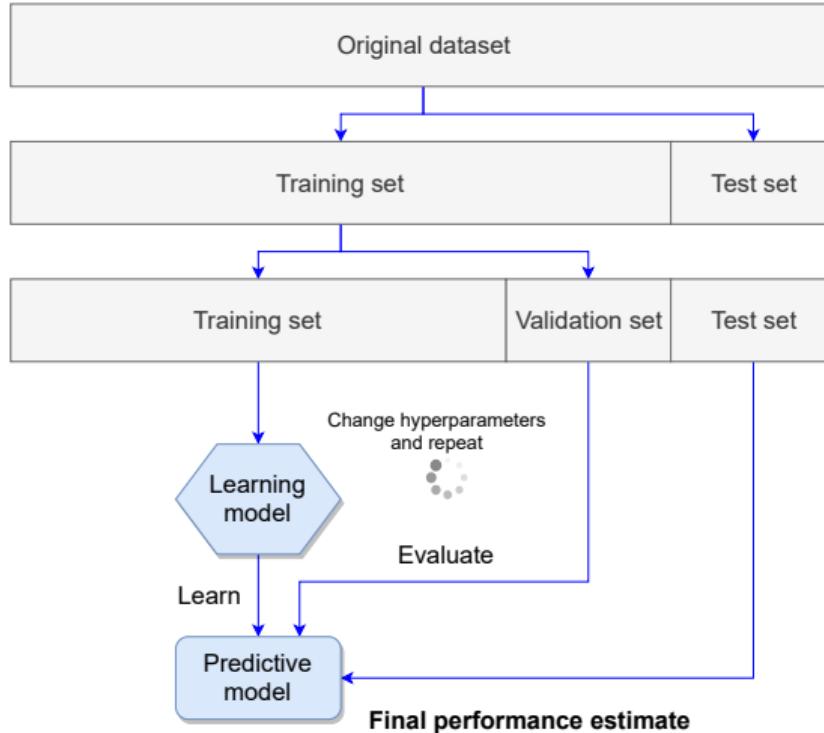
- No structure → fully connected
- Spatial structure → convolutional
- Sequential structure → recurrent
- Mixed structure → **our design**



[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Hold-out validation

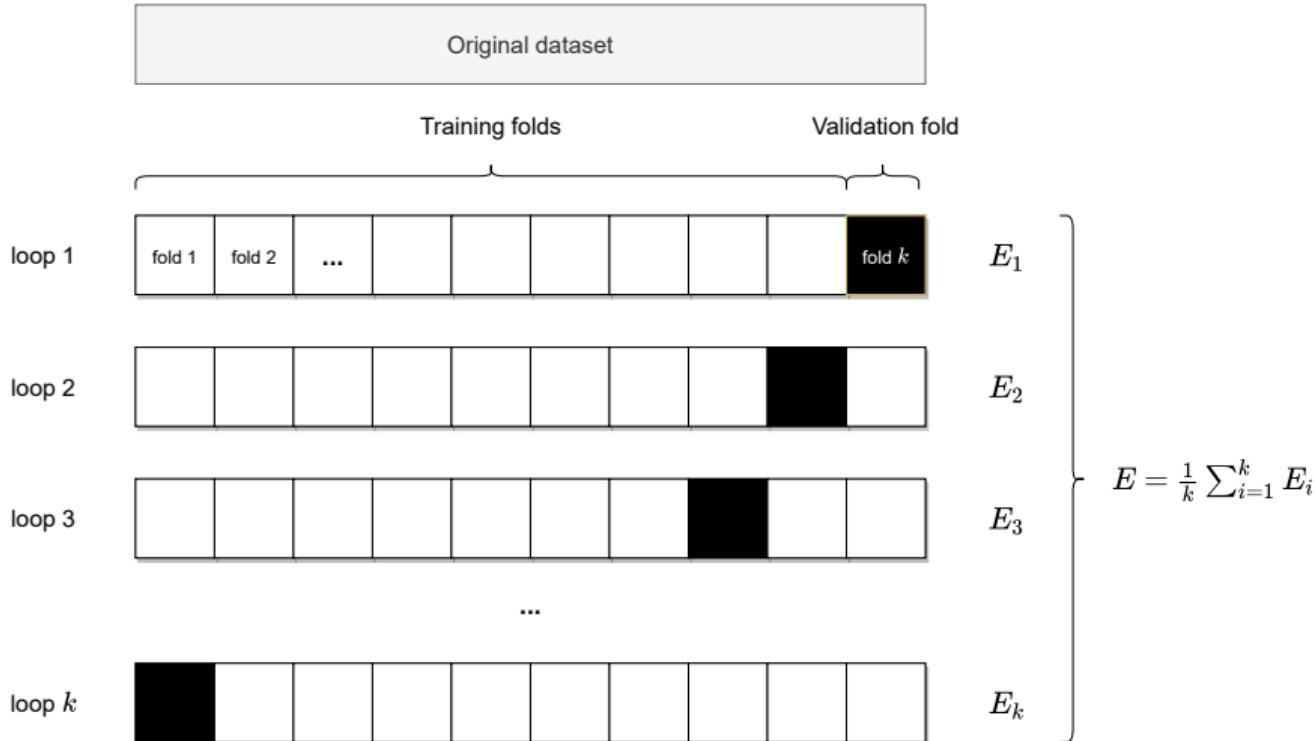
- If we have much data



[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

K-fold validation

- If we don't have much data





History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

Sanity test

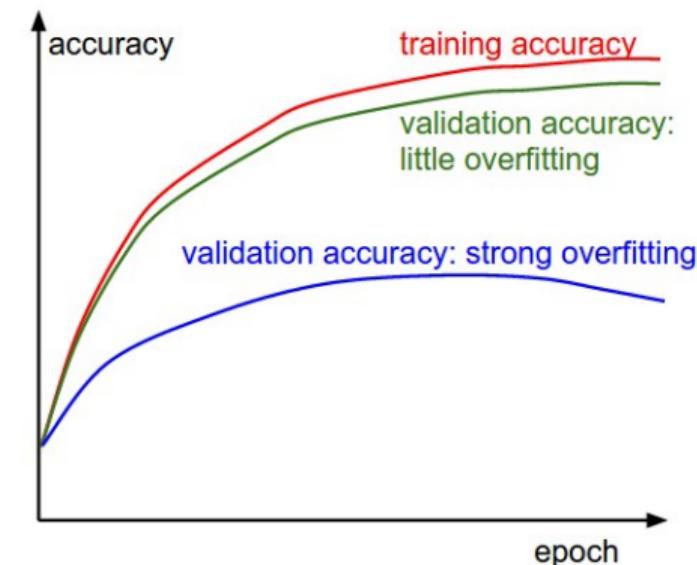
- Make sure that the model can overfit very small portion of the training data.
- Take ≈ 20 samples, turn off regularization and make sure that model can get a loss of 0 (accuracy 100%).
 - If model can't overfit there is a problem: or there is something broken or we have to scale up your network.



Monitor learning curves

The gap between the training and validation accuracy indicates the amount of overfitting

- Gap between train-validation is too big: overfitting, increase regularization
- Gap between train-validation too small: increase model capacity





Monitor learning curves (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

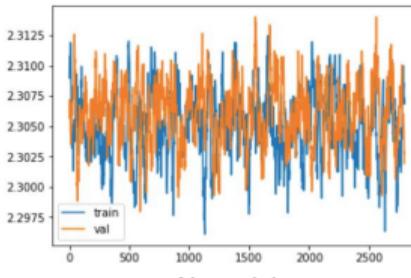
Prepare Data

Choose Metric

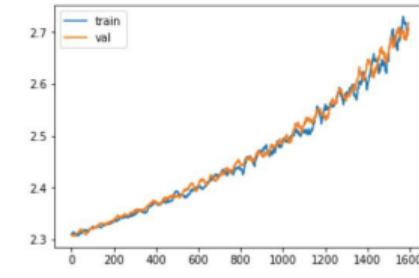
Choose Classifier

Train Classifier

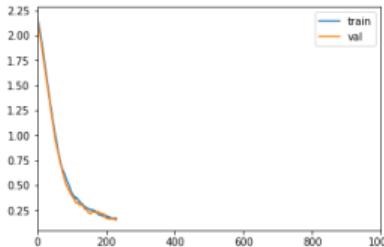
Explainable Machine
Learning



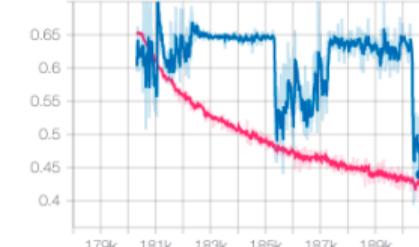
Not training
Bug in update calculation?



Error increasing
Bug in update calculation?



Get NaNs in the loss after a number of iterations:
Numerical instability



Weird cyclical patterns in loss:
Data not shuffled

Shuffling off
Shuffling on



Monitor learning curves (cont.)

History

Architecture

Activation functions

Why Deep?

Learning Problem

Vanishing gradient problem

Nonsaturating Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

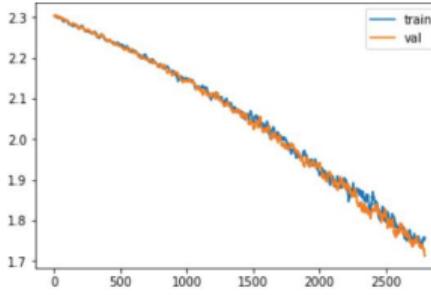
Prepare Data

Choose Metric

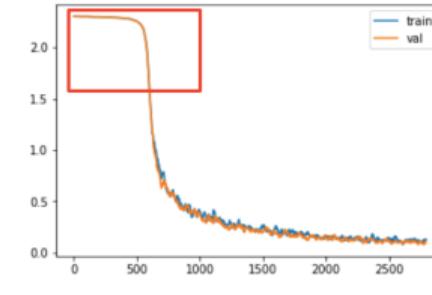
Choose Classifier

Train Classifier

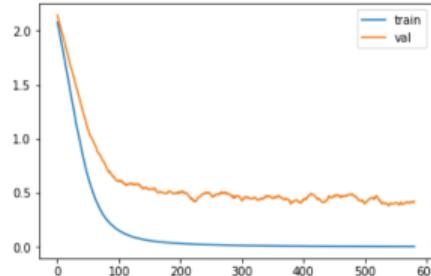
Explainable Machine Learning



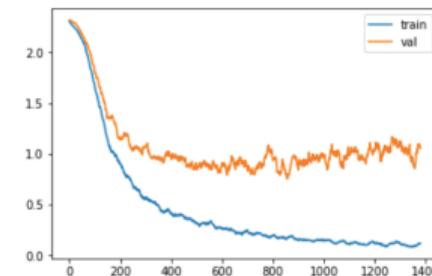
Not converged yet
Keep training, possibly increase learning rate



Slow start
Bad initialization?



Possible overfitting

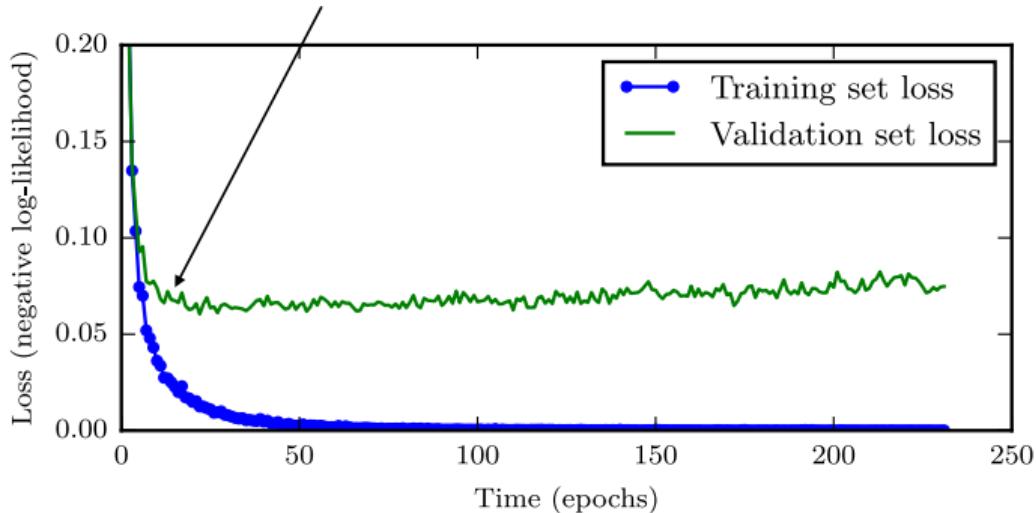


Definite overfitting



When to stop training?

- Monitor validation error to decide when to stop
 - “Patience” hyperparameter: number of epochs without improvement before stopping
 - **Early stopping:** terminate while validation set performance is better (a kind of regularization)



[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Other techniques

- Track the ratio of *weight updates* and *weight magnitudes*.

$$r = \frac{\text{weight_updates}}{\text{weight_magnitudes}} \quad (12)$$

- If this is too high to decrease learning rate
- If it is too low to increase learning rate
- If dropout is not working for model, we should probably be using a bigger network.

[History](#)[Architecture](#)[Activation functions](#)[Why Deep?](#)[Learning Problem](#)[Vanishing gradient problem](#)[Nonsaturating Activation Functions](#)[Dropout](#)[Batch Normalization](#)[Workflow](#)[Metrics](#)[Workflow](#)[Prepare Data](#)[Choose Metric](#)[Choose Classifier](#)[Train Classifier](#)[Explainable Machine Learning](#)

Using a pretrained model

Concept 7

A **pretrained network** is a saved network that was previously trained on a large dataset

- A common and highly effective approach to deep learning on small image datasets is to use a pretrained network.
- There are two ways to use a pretrained network: *feature extraction* and *fine-tuning*.



Using a pretrained model (cont.)

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

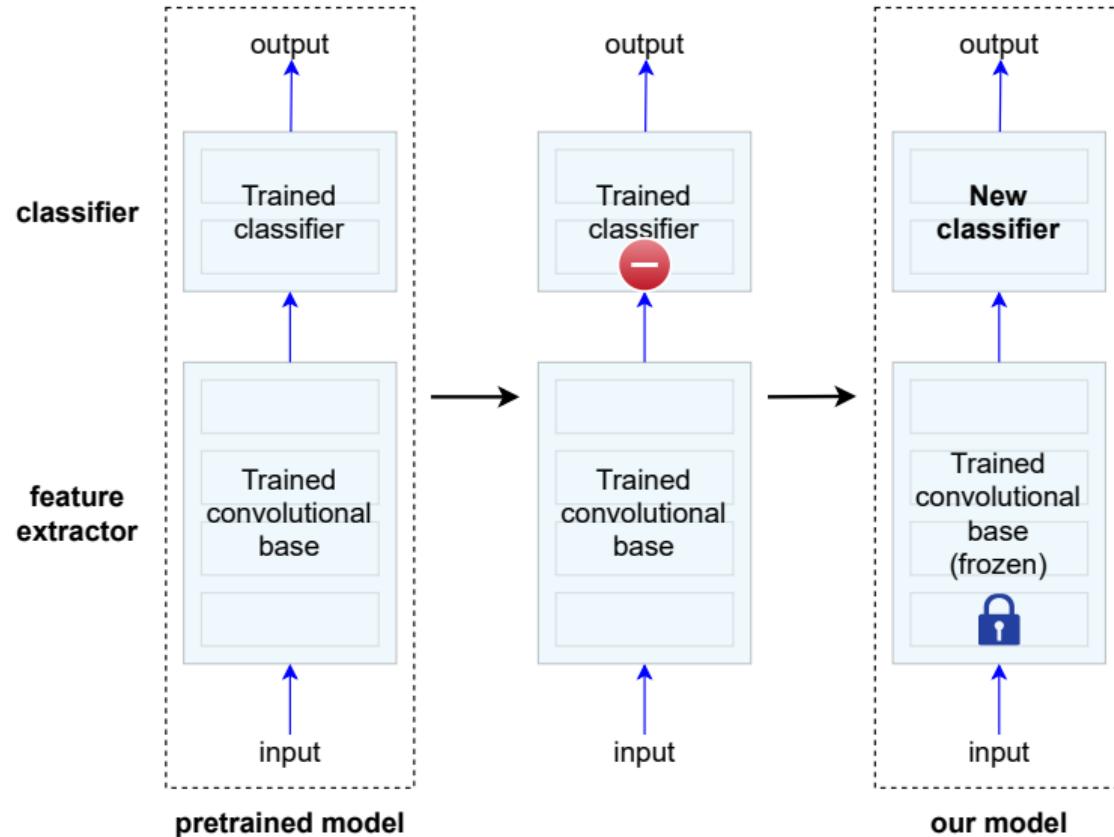
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning





Essential Skills to ML scientists/engineers

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

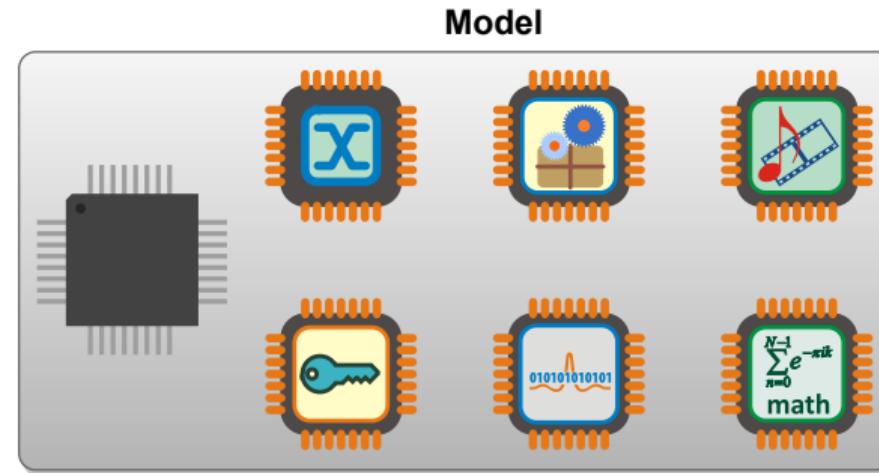
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning



- **Design** a model
- **Run + Debug** the model
- **Revise** the model



The Explainable Artificial Intelligence

History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning

- **Explainable** could mean **interpretable**
- **Abstraction:** how much of an explanation do we need?
- The **Explainable AI (XAI)** program aims to create a suite of machine learning techniques that:
 - Produce more explainable models, while maintaining a high level of learning performance (prediction accuracy)
 - Enable human users to understand, appropriately trust, and effectively manage the emerging generation of artificially intelligent partners.



The Explainable Artificial Intelligence (cont.)

History

Architecture

Activation functions

Why Deep?

Learning Problem

Vanishing gradient problem

Nonsaturating Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

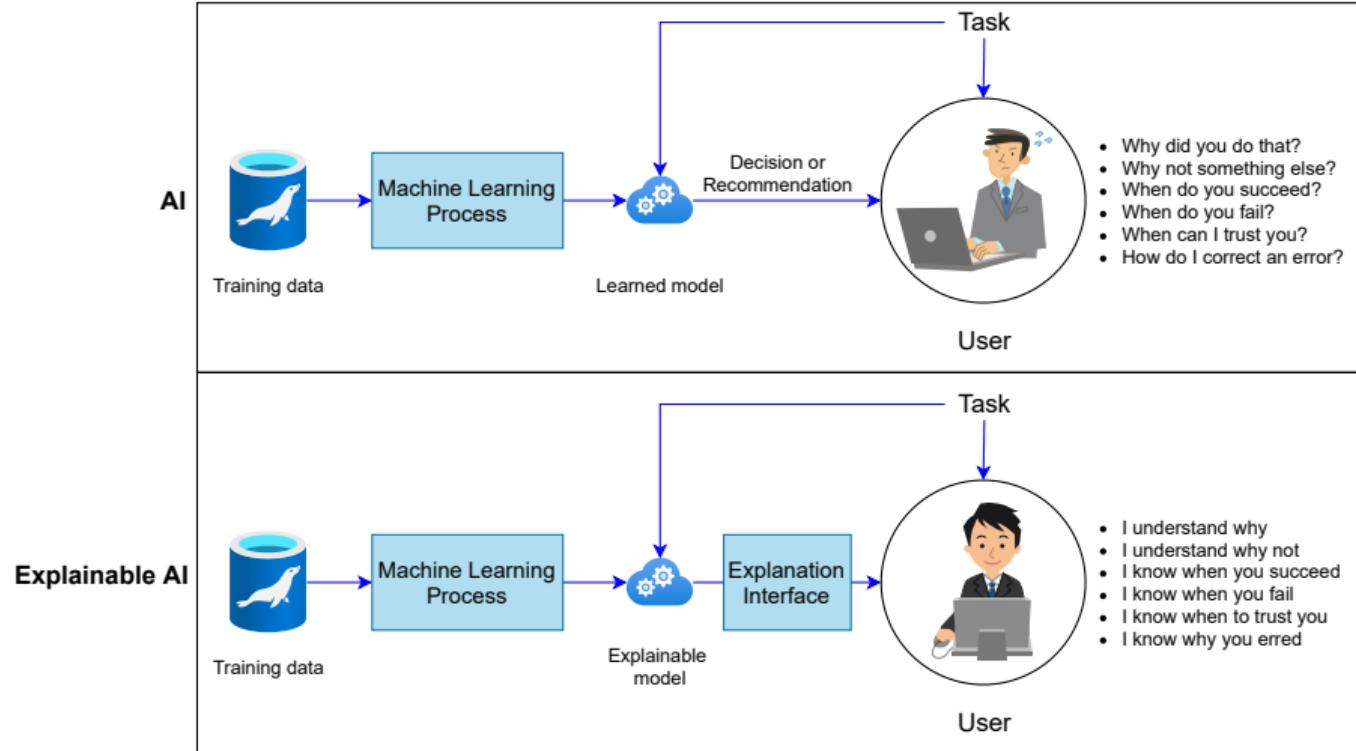
Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine Learning

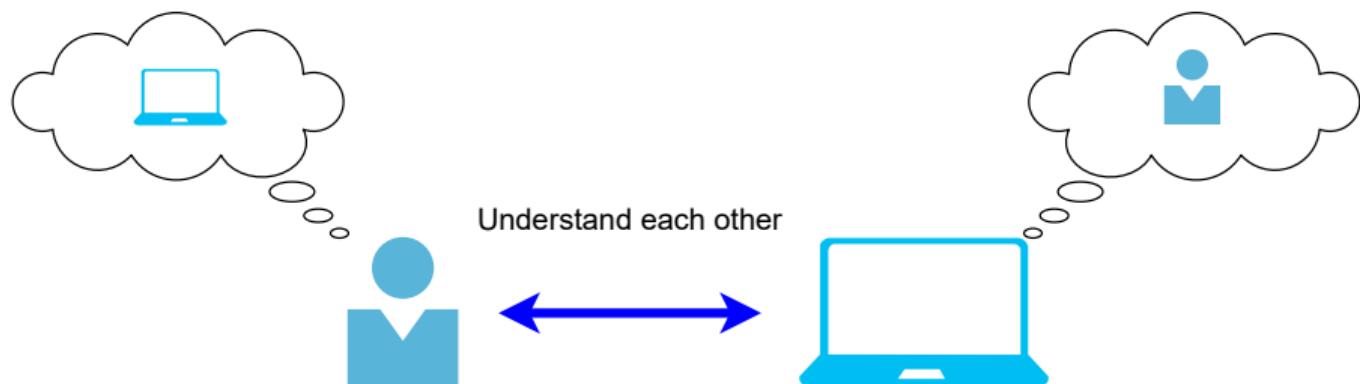




Human vs. Model

The challenge is people don't understand the system and the system doesn't understand the people.

- We need models for the system to use to understand and explain things to the human.
- We also need models in the human's head about what the system does.



History

Architecture

Activation functions

Why Deep?

Learning
Problem

Vanishing gradient
problem

Nonsaturating
Activation Functions

Dropout

Batch Normalization

Workflow

Metrics

Workflow

Prepare Data

Choose Metric

Choose Classifier

Train Classifier

Explainable Machine
Learning



References

- Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.
- Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.
- Russell, S. and Norvig, P. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.