University of Science VNUHCM

Introduction to Big Data

KERAS

Topic: ML1 - Keras

Team: KHMT1 - 02 Kinchana

Le Anh Thu	21127175
Nguyen Nhat Truyen	21127191
Nguyen Minh Dat	21127592
Huynh Duc Thien	21127693

"We are not the same. But we are, somehow, one."

—*Veronica Roth*

TABLE OF CONTENTS

01
Introduction
History, origin of Keras

02
Insights
Population

03
Demonstration
Usage and Application

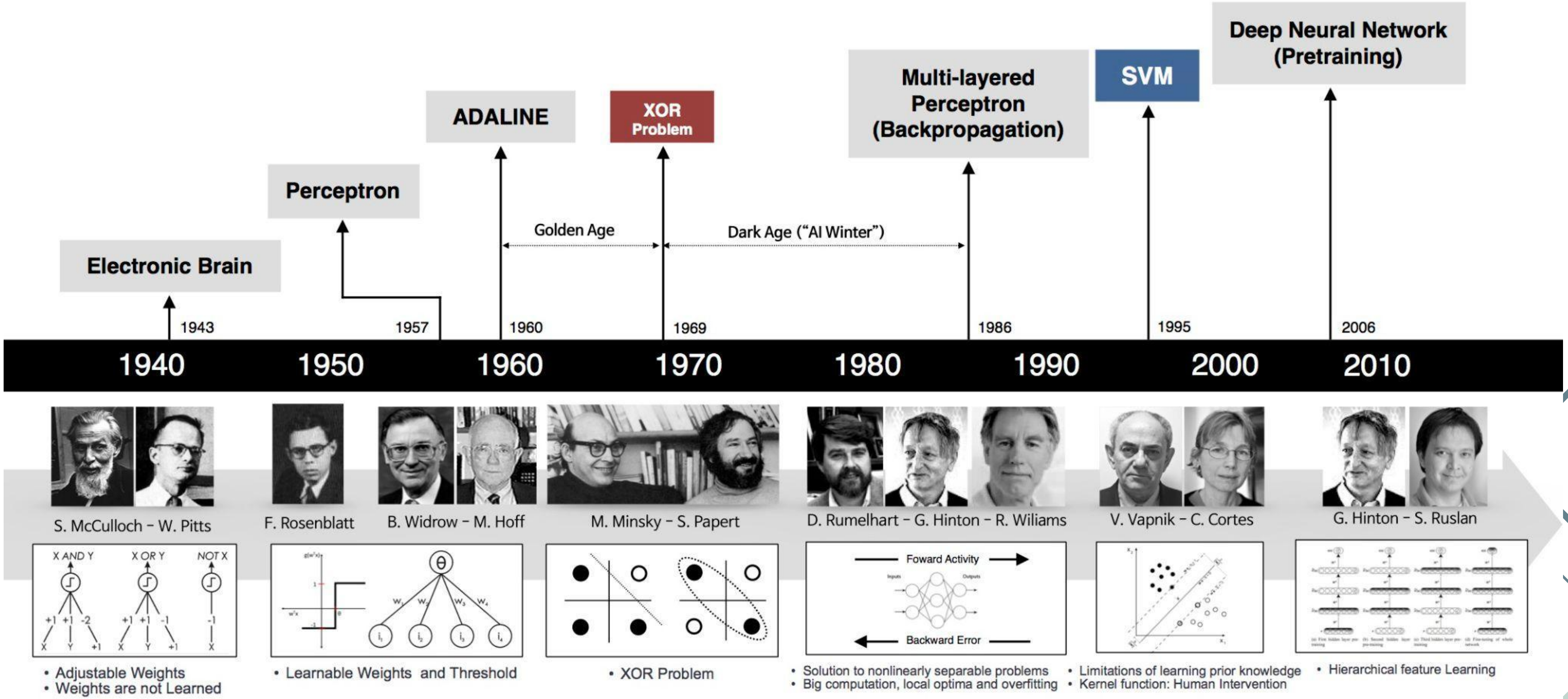
04
Conclusion
In summary

01

Introduction

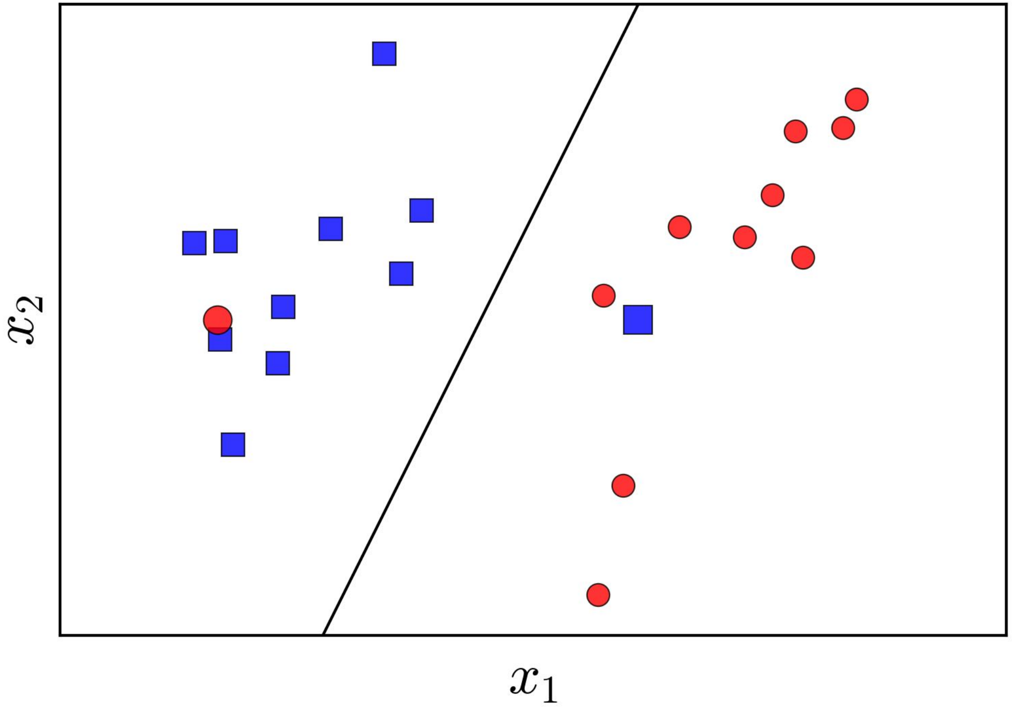
History of Deep Learning and breakthrough of new era

History of Deep Learning



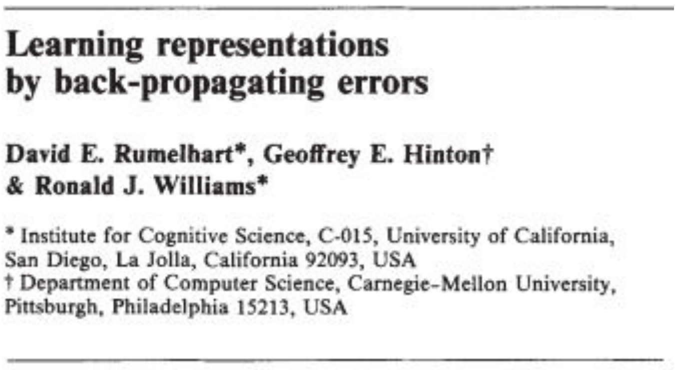
Source: Deep Learning 101 - Part 1: History and Background

1960s: Perceptron Learning Algorithm (PLA)



*Only work with linearly separable data →**The 1st AI winter***

1980s: MLP and Backpropagation



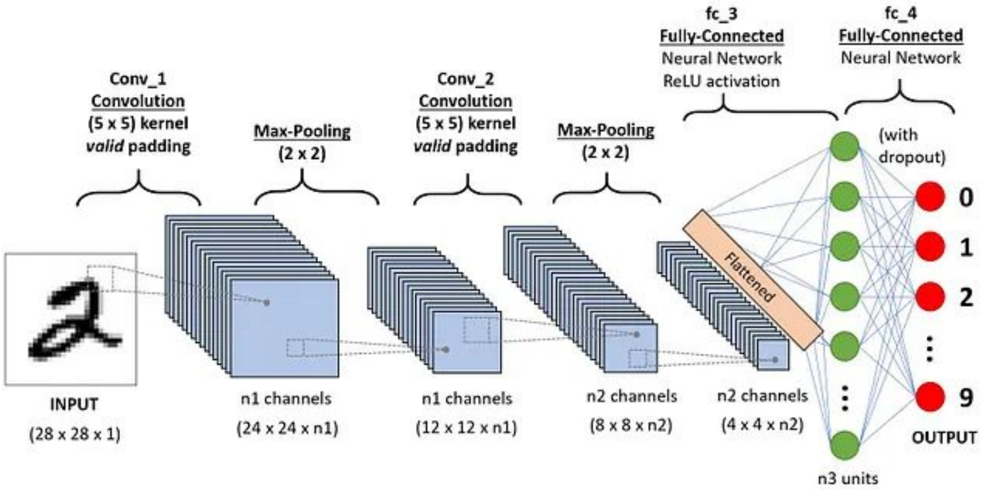
Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

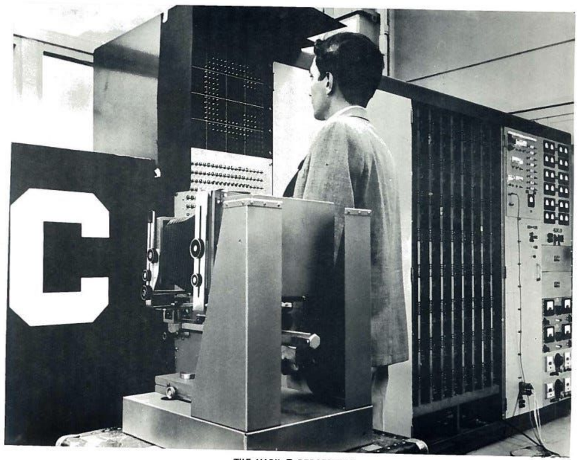
Multi-layer Perceptron (MLP)

Backpropagation



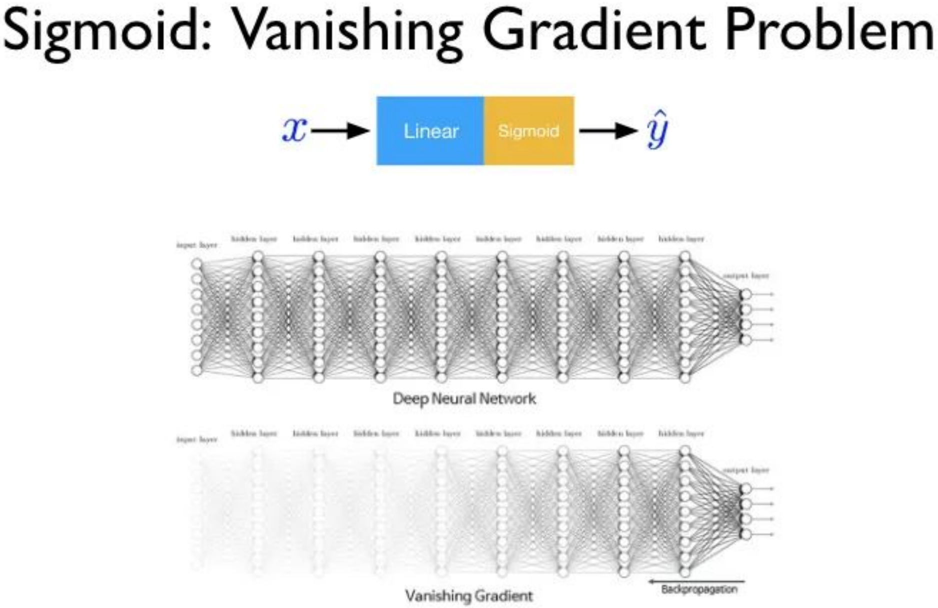
Convolutional Neural Nets

1990-2000s: The 2nd AI Winter

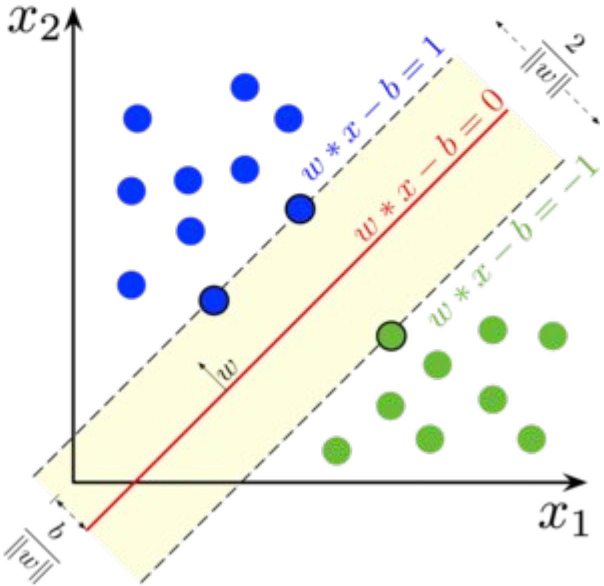


THE MARK I PERCEPTRON

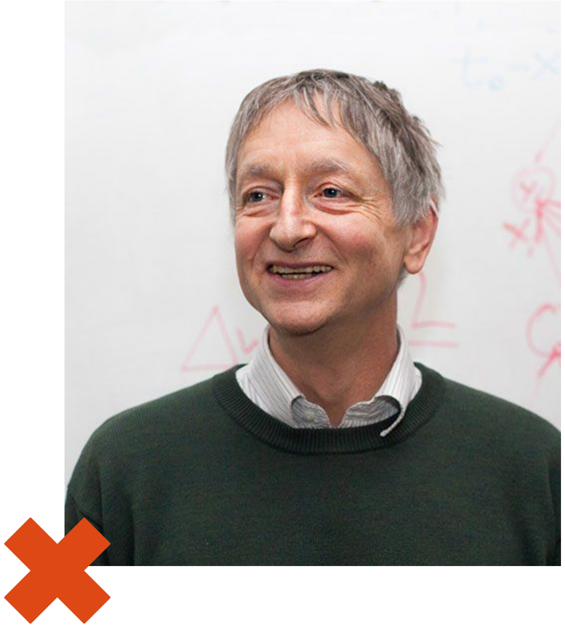
Data & Computing Limitation



1990-2000s: The 2nd AI Winter



Support Vector Machines (SVM)



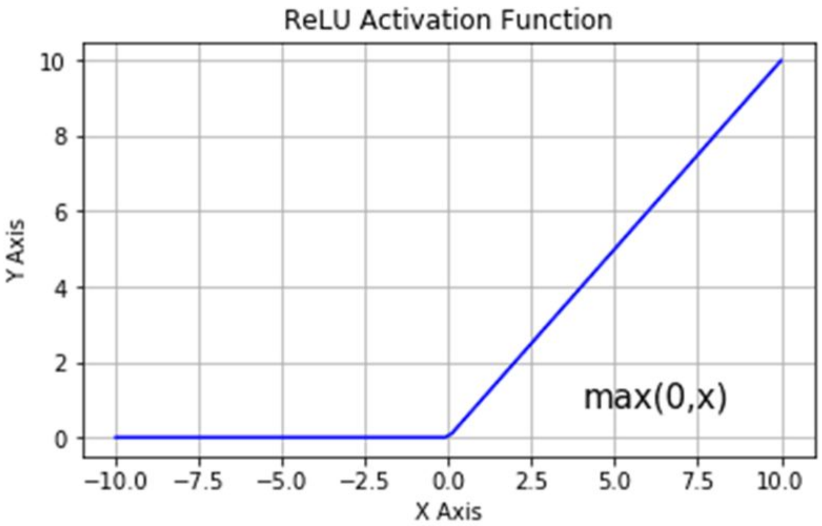
2010s: ImageNet Dataset

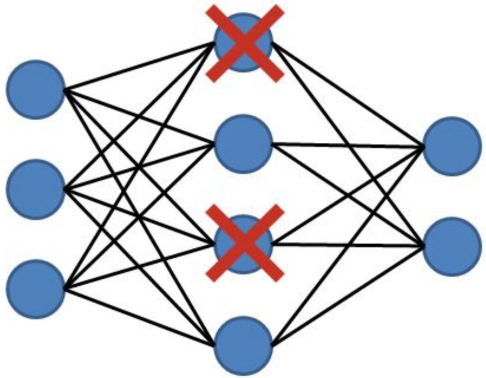
Top-5 error rate in 2010 is 28%

Top-5 error rate in 2011 is 26%

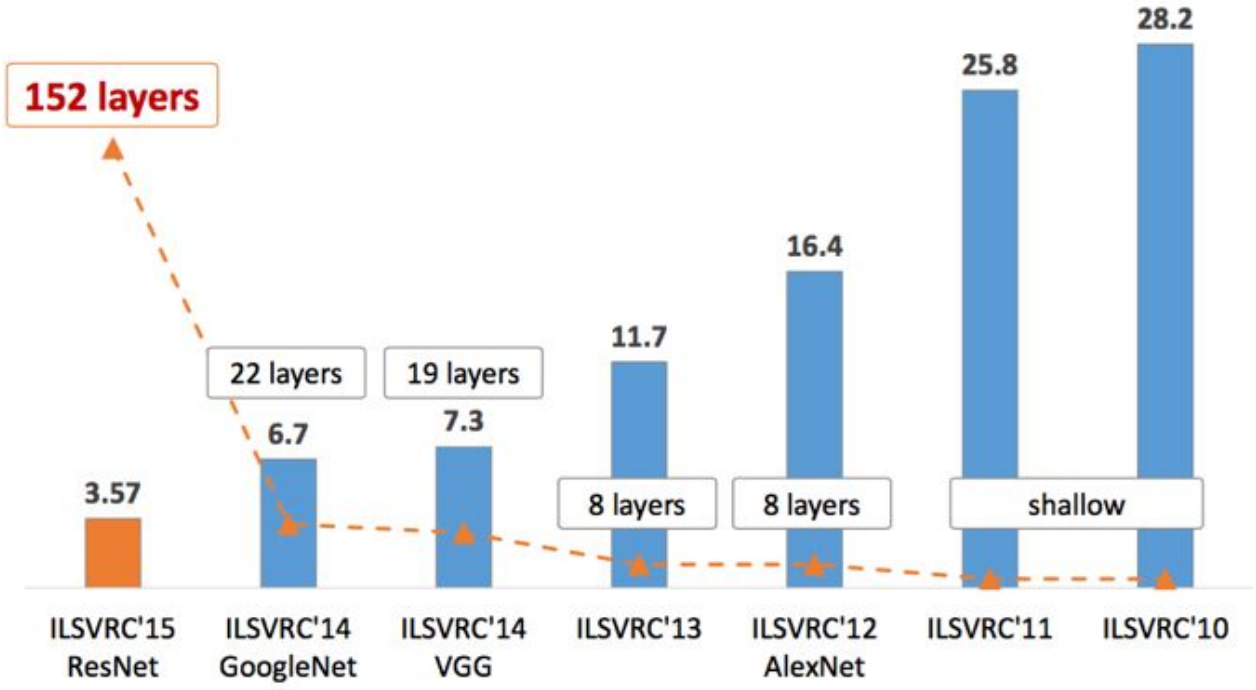
2012s: Deep Convolutional Neural Network

Top-5 error rate in 2012 is 16%





2012s-now: “Deep” in Deep Learning

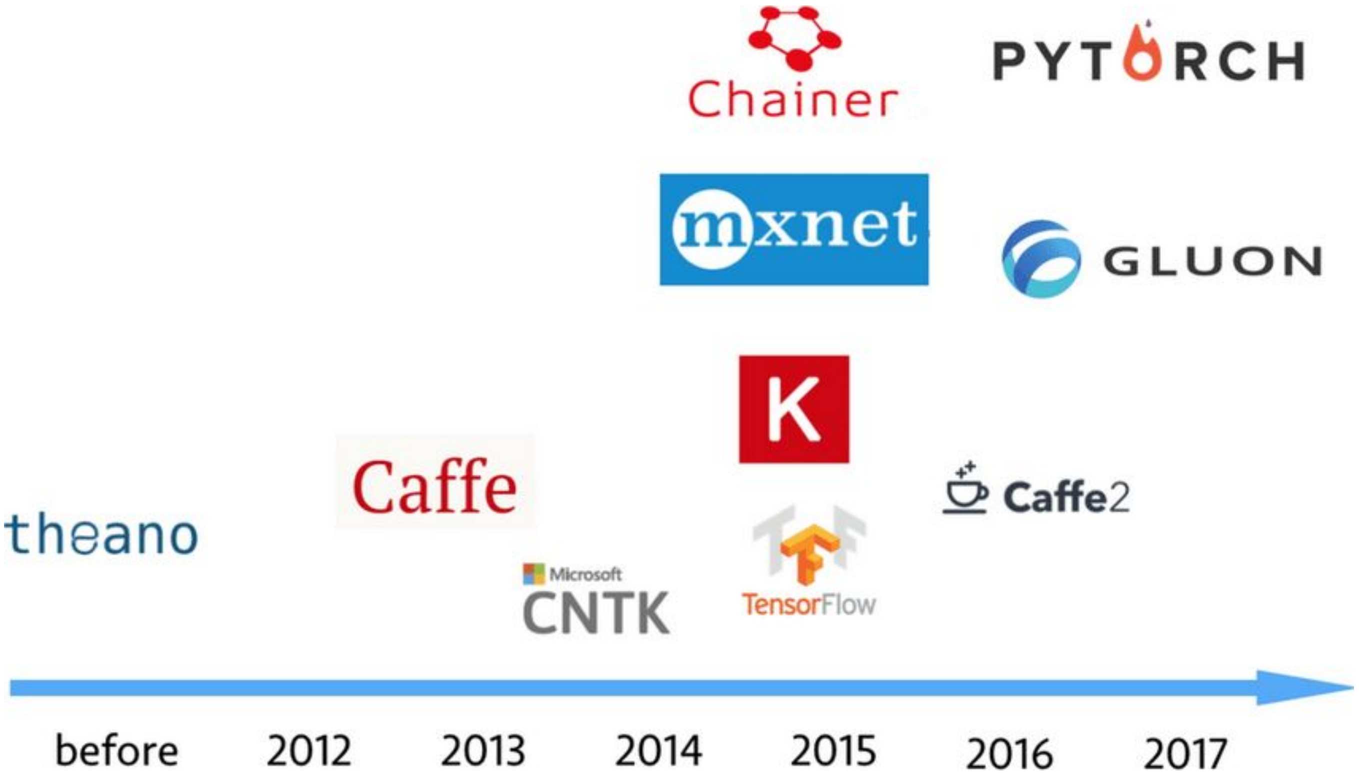


Source: CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ...

02

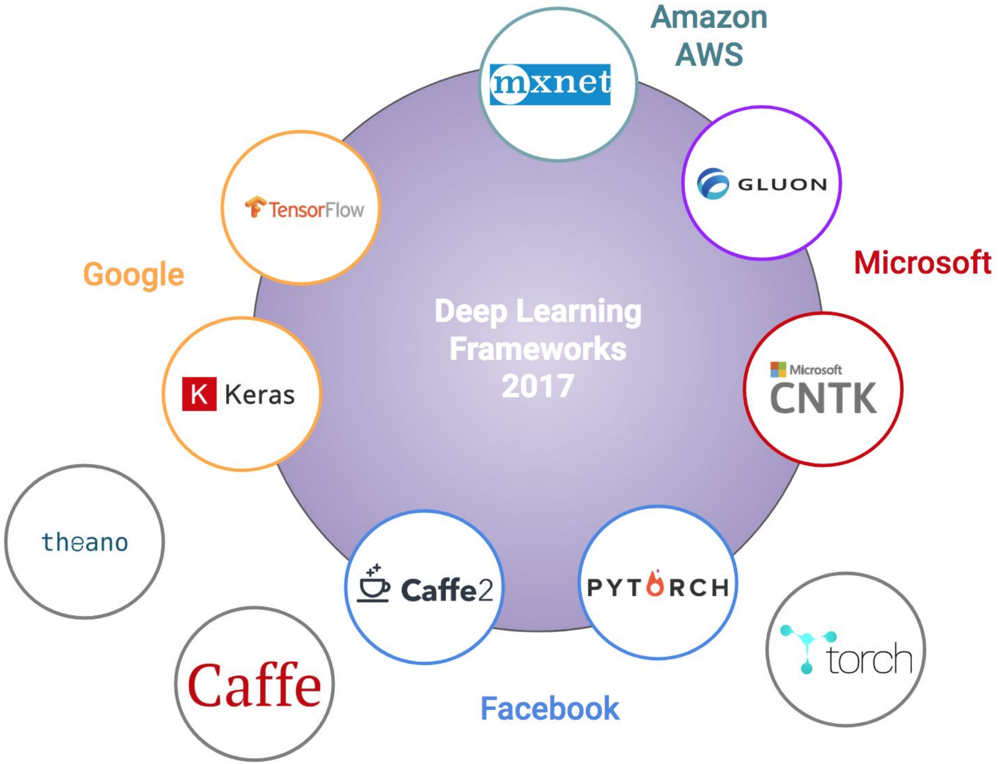
Insights

Breakthrough of DL Framework



Source: DLBench: a comprehensive experimental evaluation of deep learning frameworks

Deep Learning Frameworks



Source: Battle of the Deep Learning frameworks—Part I: 2017, even more frameworks and interfaces

A Good Framework must have

01
GPU
Computing with GPUs and distributed systems

02
Languages
C/C++, Python, Java,...

03
OS
Windows, Linux, MacOS

04
model zoo
Contains commonly trained deep learning models

05
Backpropagation
Supports automatic backpropagation calculation

06
Community
A large contributing community

Comparison of deep learning software

6 languages

Article Talk | Read Edit View history Tools

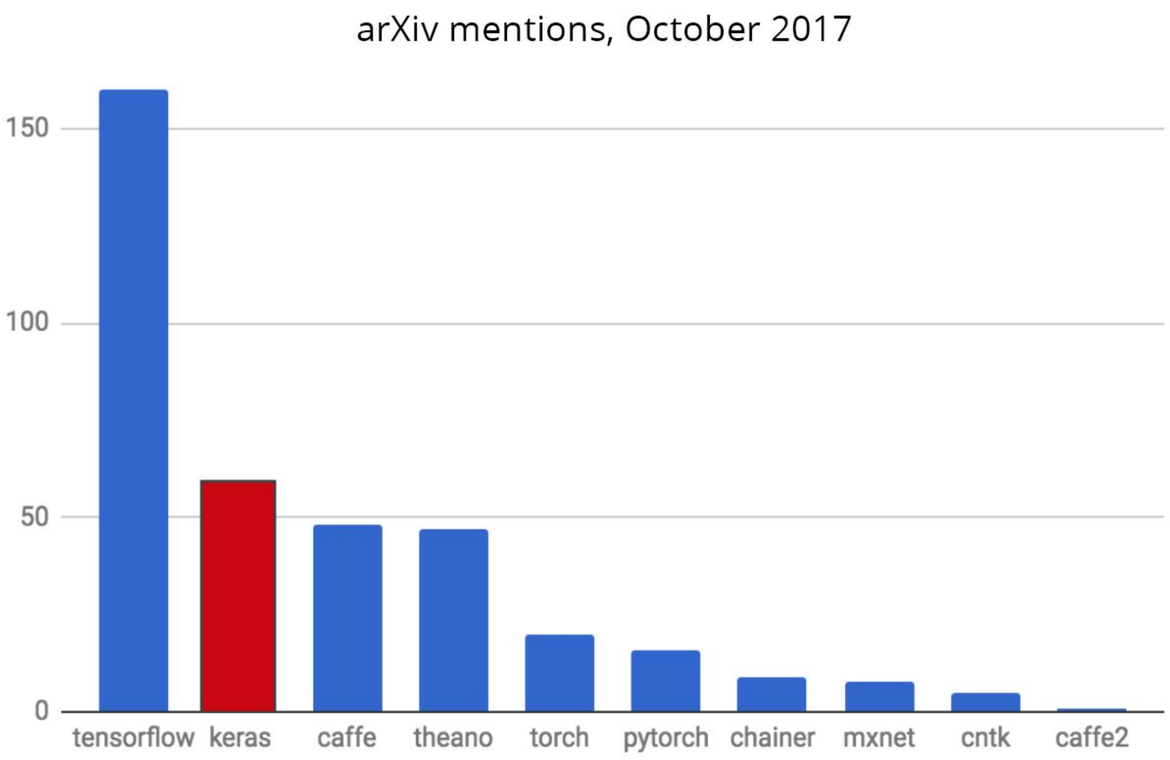
From Wikipedia, the free encyclopedia

The following table compares notable software frameworks, libraries and computer programs for deep learning.

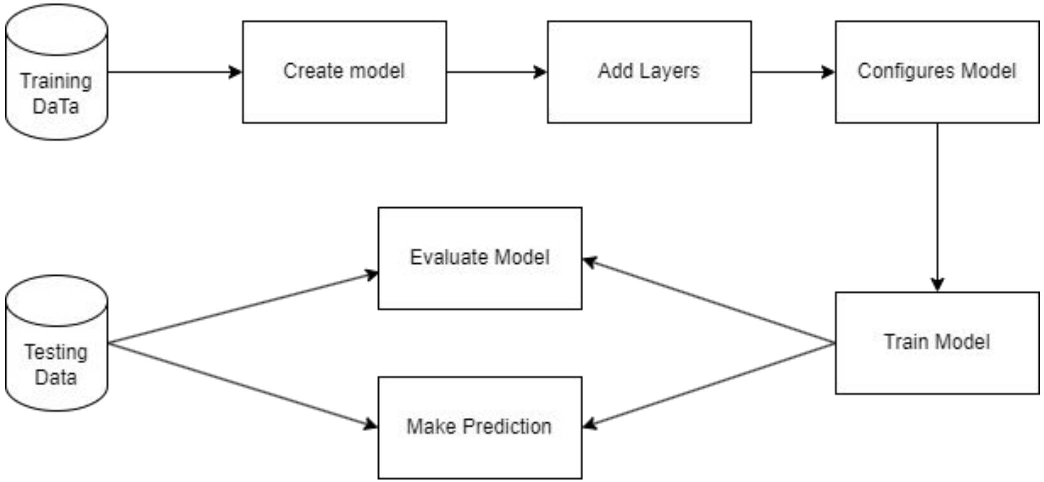
Deep-learning software by name [edit]

Software	Creator	Initial release	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	ROCm support[1]	Automatic differentiation[2]	Has pretrained models	Re
BigDL	Jason Dai (Intel)	2016	Apache 2.0	Yes	Apache Spark	Scala	Scala, Python			No	No		Yes	
Caffe	Berkeley Vision and Learning Center	2013	BSD	Yes	Linux, macOS, Windows[3]	C++	Python, MATLAB, C++	Yes	Under development[4]	Yes	No	Yes	Yes[5]	
Chainer	Preferred Networks	2015	BSD	Yes	Linux, macOS	Python	Python	No	No	Yes	No	Yes	Yes	
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	2014	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	No[5]	Yes[9][10]	No	Computational Graph	Yes[11]	
Dlib	Davis King	2002	Boost Software License	Yes	Cross-platform	C++	C++, Python	Yes	No	Yes	No	Yes	Yes	
Flux	Mike Innes	2017	MIT license	Yes	Linux, MacOS, Windows	Julia	Julia			Yes	No	Yes	Yes[13]	

Source: *Comparison of deep learning software*


arXiv mentions, October 2017
150
100
50
0
tensorflow
keras
caffe
theano
torch
pytorch
chainer
mxnet
cntk
caffe2

Keras Components



Keras Model Workflow

Keras Components

01
Sequential Model
multi-layer perceptron

02
Add Layer
add() is used to add a layer to the neural network

03
Dense Layer
Fully connected layer: Interconnected nodes, specified by parameters

04
Model Compilation
Compile the model with optimizer and loss function

05
Model Training
fit() used to train the model

06
Make Predictions
predict() is used to make predictions on new input data

07
Model Evaluation
evaluate() is used to assess the model performance

Keras Components

```
# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.1, random_state=42)

# Data preprocessing
x_train = x_train / 255.0
x_test = x_test / 255.0

# Build the model
model = Sequential()
model.add(Flatten(input_shape=(28, 28))) # Flatten the image into a vector
model.add(Dense(128, activation='relu')) # Hidden layer with 128 units and ReLU activation function
model.add(Dense(10, activation='softmax')) # Output layer with 10 units and softmax activation function

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_val, y_val))

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(x_test, y_test)
```

```
Epoch 1/5
1688/1688 [==============================] - 8s 4ms/step - loss: 0.2723 - accuracy: 0.9243 - val_loss: 19.4958 - val_accuracy: 0.9558
Epoch 2/5
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1217 - accuracy: 0.9643 - val_loss: 12.9215 - val_accuracy: 0.9697
Epoch 3/5
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0831 - accuracy: 0.9745 - val_loss: 15.4571 - val_accuracy: 0.9685
Epoch 4/5
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0625 - accuracy: 0.9810 - val_loss: 11.8054 - val_accuracy: 0.9757
Epoch 5/5
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0491 - accuracy: 0.9852 - val_loss: 13.3832 - val_accuracy: 0.9752
313/313 [==============================] - 1s 2ms/step - loss: 0.0843 - accuracy: 0.9720
Accuracy on the test set: 0.972000002861023
```

03

Demonstration

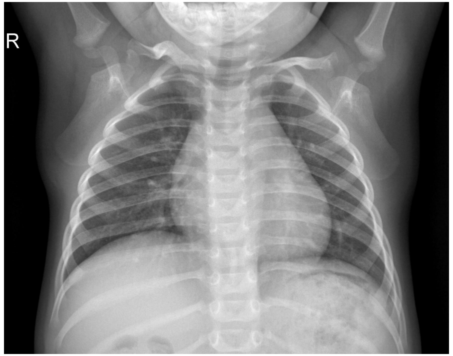
PROBLEM STATEMENT

Pneumonia

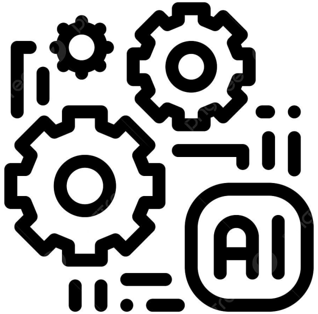
Affects our lungs
Caused by bacteria, viruses…
Leads to fluid air sacs or alveoli

Health care

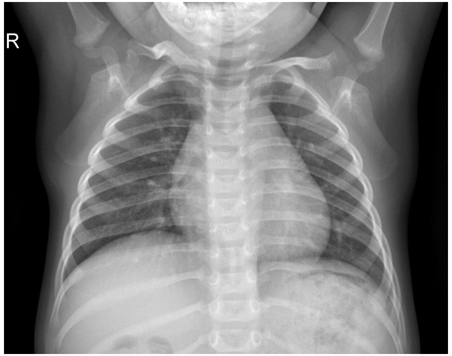
PROBLEM STATEMENT

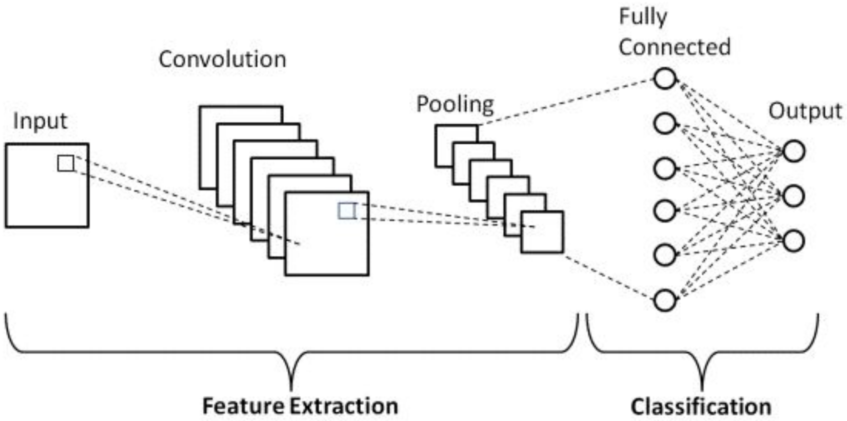


Combination



PROBLEM STATEMENT





Keras

Making Prediction

KERAS PIPELINE

01

Description

02

Explore

03

Preprocess

04

Implement

KERAS PIPELINE

01 Description

02 Explore

03 Preprocess

04 Implement

DATA DESCRIPTION

01

Structure

Organized into 3 folders
train - test - val

02

Quantity

5,863 X-Ray images
Pneumonia/Normal

03

Origin

Guangzhou Women and
Children's Medical Center

04

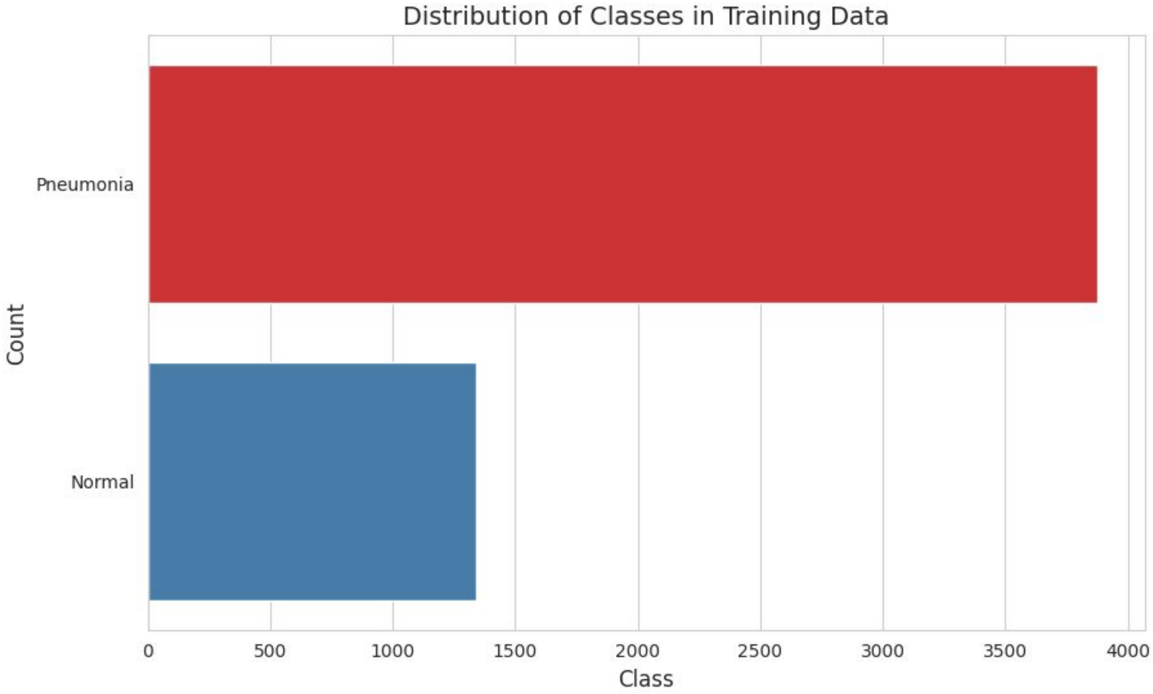
Quality

Under quality control
process

KERAS PIPELINE

01	02	03	04
Description	Explore	Preprocess	Implement

Distribution



KERAS PIPELINE

01

Description

02

Explore

03

Preprocess

04

Implement

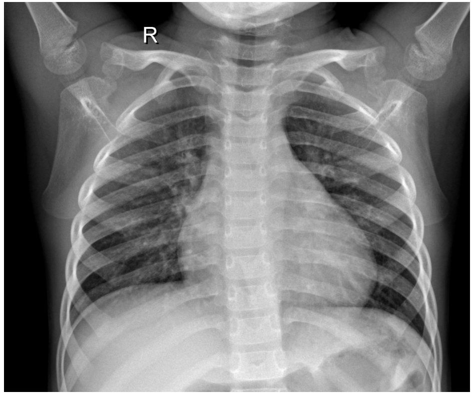
Normalization

Reshape

(N, width, length, channel)

Rescale

Scale from [0, 255] to [0, 1]



Data Augmentation

01

Size

Expand the size of the dataset

02

Variation

Resize, rotate, flip, shift, normalize,...

03

Role

Help model generalize and avoid overfitting

Keras ImageDataGenerator

01

Functionalities

- Return new variations of images at each epoch
- Do not change the original images

02

Advantages

- Loading the images in batches
- Requires lower memory usage

KERAS PIPELINE

01 Loading

02 Explore

03 Preprocess

04 Implement

MODEL

5

Conv Layers

5

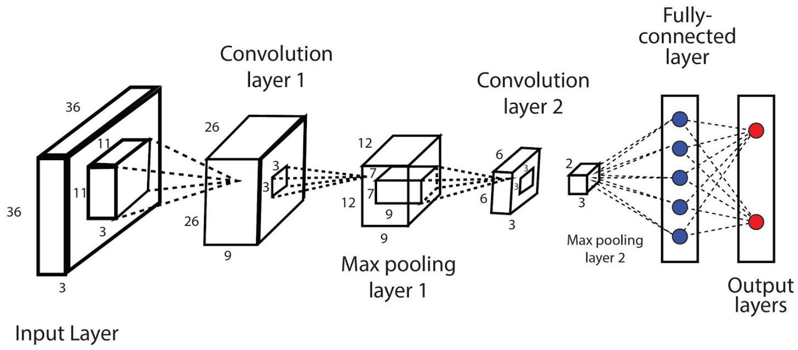
MaxPooling2D

4

Batch Normalization

2

Dropout



PYTORCH PIPELINE

01 Additional Preprocess

02 Transform

03 Sampler

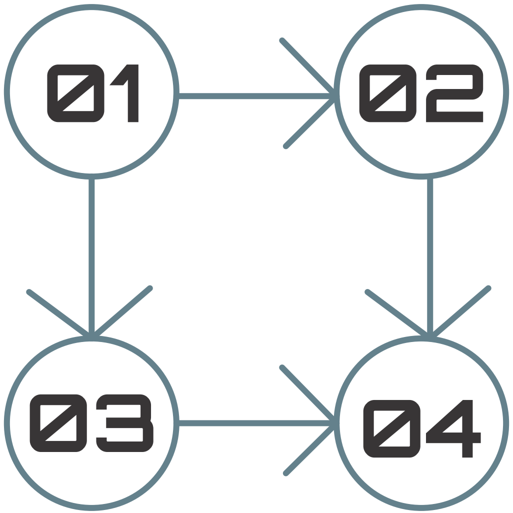
Additional Preprocessing

Nested dir

PNEUMONIA—
h1.img——
h2.img——
NORMAL—
h3.img——
h4.img——

Image dir

h1.img
h2.img
h3.img
h4.img



.csv file

“h1.img”	1
“h2.img”	1
“h3.img”	0
“h4.img”	0

Custom Dataset

PyTorch transform

Keras ImageDataGenerator

In Keras, the ImageDataGenerator can be fit to data and automatically compute the internal data stats

PyTorch transform

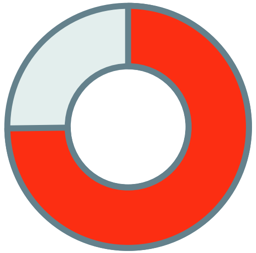
In PyTorch, we have to compute the stats **on our own** and **in advance**

Normalization

Have to compute **mean** and **std**

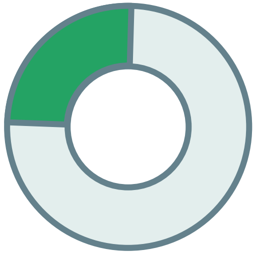
Weighted Random Sampler

75%



PNEUMONIA

25%



NORMAL

<u>Problem</u>:
Model may be tempted to predict Pneumonia
<u>Solution</u>:
Assign a higher probability of being selected to samples from the Normal class

In a nutshell

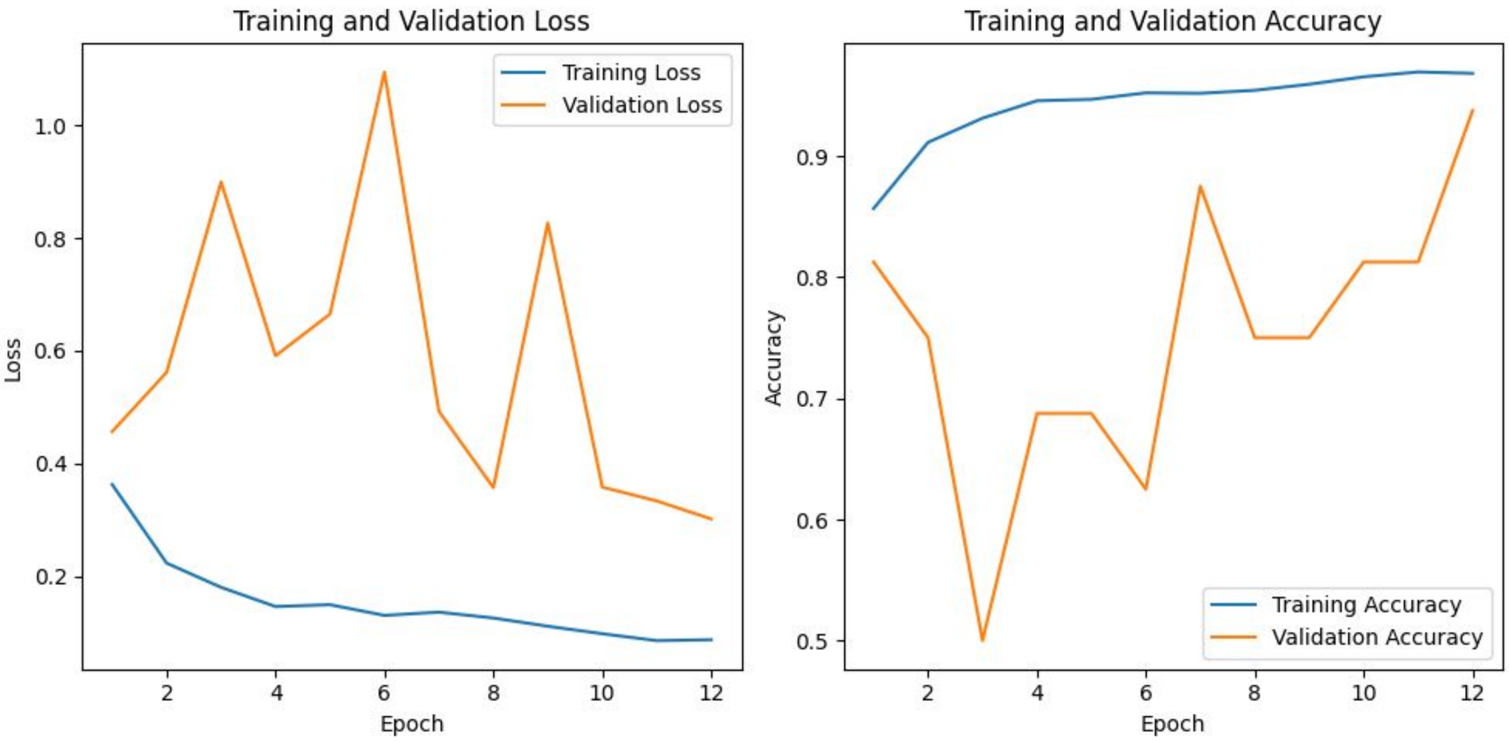
Transform + **Sampler**

Diversify dataset

Tackle imbalanced

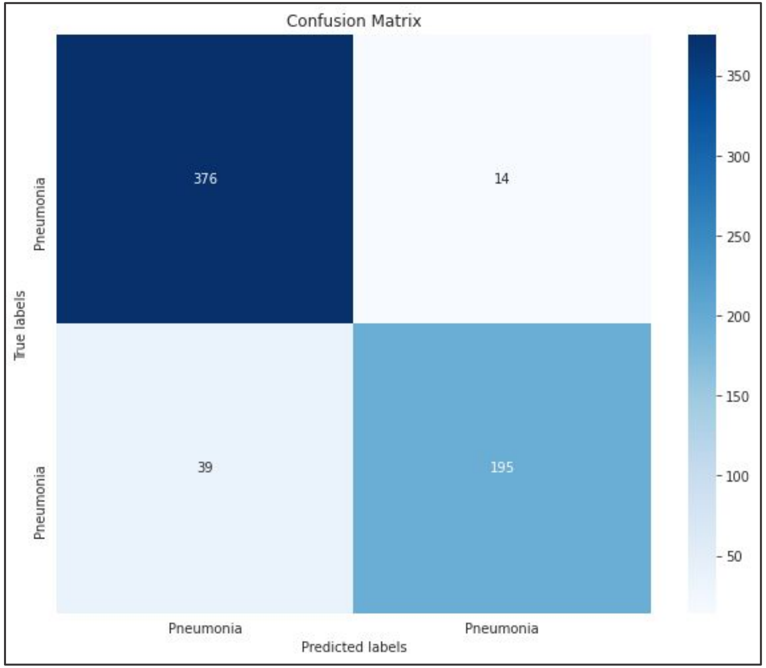
→ Model can generalize well, avoid biased and overfitting.

Evaluation


Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch
Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch

Evaluation

	precision	recall	f1-score	support
Pneumonia (Class 0)	0.91	0.96	0.93	390
Normal (Class 1)	0.92	0.84	0.88	234
accuracy			0.91	624
macro avg	0.92	0.90	0.91	624
weighted avg	0.91	0.91	0.91	624



Accuracy: 91%

CONCLUSIONS

Keras

- Easy to use
- High-level API
- Low performance

PyTorch

- More complex
- More flexibility
- High performance

Side note: Data processing can make a difference.

Thanks for listening!

Any further questions, please contacts
hdthien21@clc.fitus.edu.vn
teamKinchana