Data Learns 247 Team | Apr 2020

Deep Learning 201

Basic Concept and Implementation

Labs247
Technology in motion

Sigit Prasetyo

VP Knowledge Management Labs247

idBigData Goal Keeper, Komunitas Big Data Indonesia

sigit.prasetyo@idbigdata.com

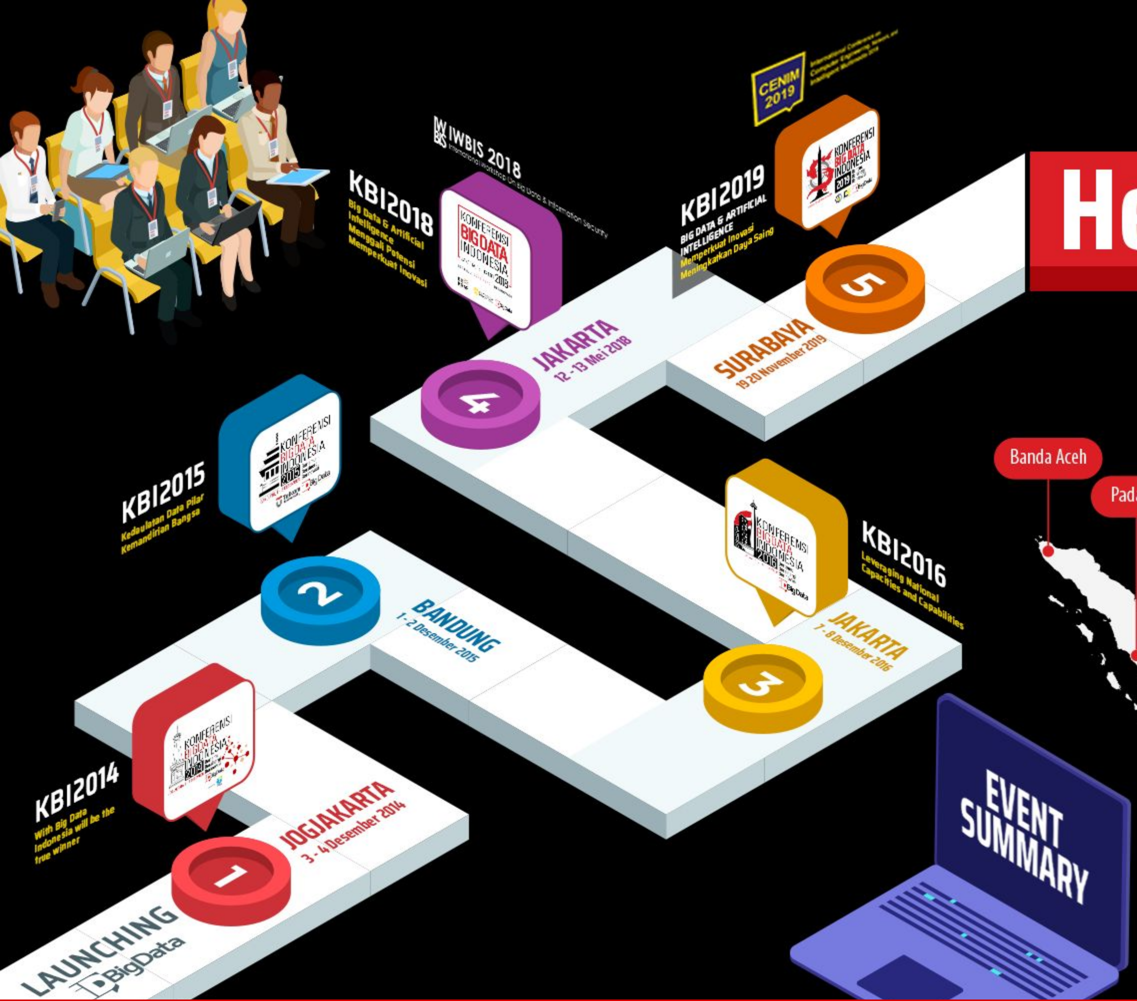
@sigitpras303

linkedin.com/in/sigitprasetyo303

flikr.com/photografer-kw3

github.com/project303

Here We Are in 2019



24 MeetUp

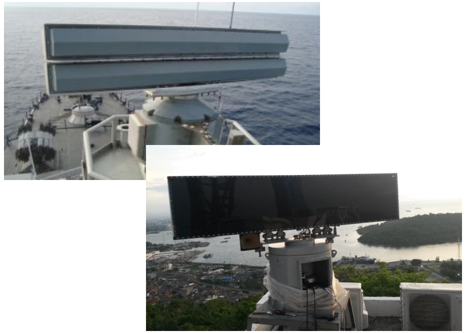
15 City

22 University

PT DUA EMPAT TUJUH

SOLUSI247 24 hours 7 days integrated ICT solution · Labs247 Technology in motion · insights247 · DATALEARNS247

An Indonesia ICT company focusing on large scale data processing and big data solution. We commits to deliver a total solution to customer with the cutting edge technology and fast information delivery



LARGE SCALE DATA PROCESSING

- Established July 24, 2000
- More than 270 employee
- 80% are engineers
- Jakarta, Jogjakarta, Surabaya

- **Client** : Telkomsel, XL Axiata, Indosat, Axis, BTel, BCA, KKP, Ristekdikti, Kemhan, BPN and more ..
- **Research Partner** : UI, UGM, ITS, BMKG, KKP, BPPT, Ristekdikti

01 Information Technology
Big data tools, data management platform, big data application

02 High Technology
Coastal Radar, Surveillance Radar, Weather Radar

03 Hybrid Solution
Surveillance framework with multi sensor fusion, Vessel Traffic Services Devices

04 Services
Custom application development, manage service and manage operation

What You Will Learn

- Definition of Deep Learning
- Use case of Deep Learning
- Refreshing basic concept of Neural Network
- Convolution Neural Network Concept
- CNN implementation for image recognition
- RNN/LSTM Concept
- GAN Concept
- Transfer Learning

How This Course is Delivered

- Combining the basic theory by focusing on implementation using Python, Tensorflow and Keras.
- The basic explanation of the theory and algorithm is not explained using in depth mathematical and statistical approach, to make it easier for participants who do not have statistics or mathematics background

CHAPTER 01

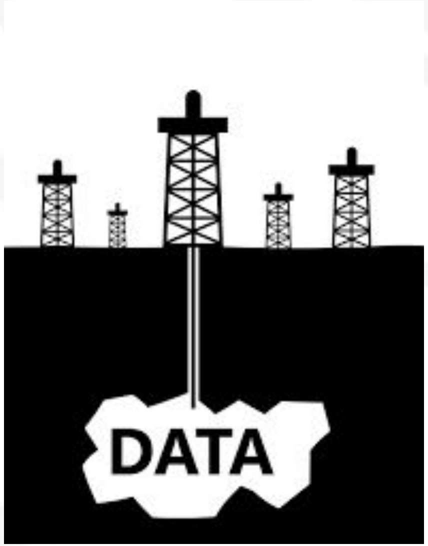
Introduction

DATA the new OIL

We need to **find** it, **extract** it, **refine** it, **distribute** it, and **use** it to drive **Economic Prosperity**

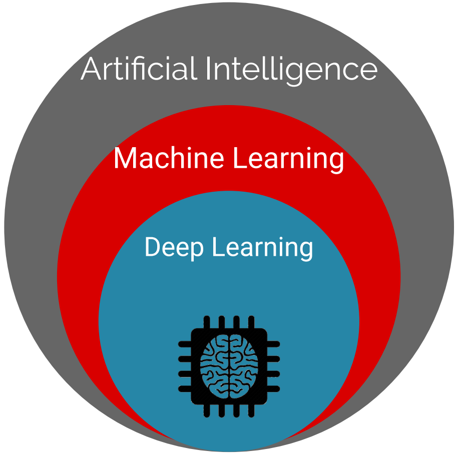
We don't have better algorithms,
we just have more data

- Peter Norvig - Director of Research at Google



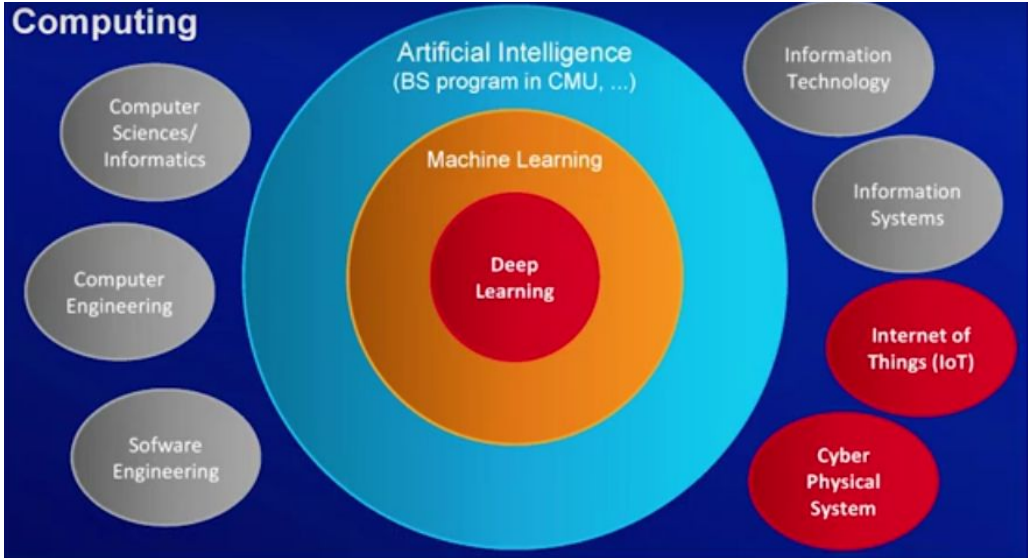
What is Deep Learning ?

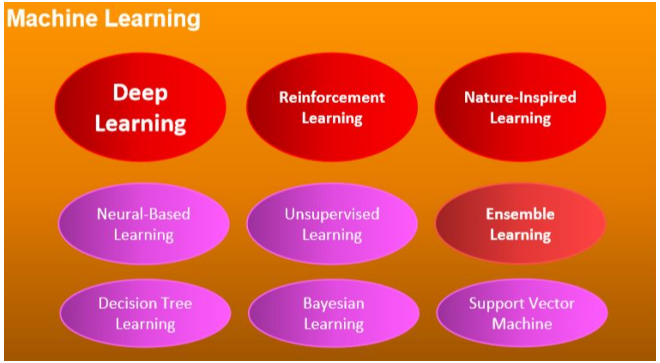
- Machine learning algorithms based on learning <u>**multiple levels**</u> (i.e deep) **of representation/ abstraction** - Yoshua Bengio
- Learning algorithms derive meaning out of data by using a hierarchy of **multiple layers** of units (neurons)
- Each neuron/node computes a weighted sum of its inputs and the weighted sum is passed through a nonlinear function, each layer transforms input data in more and more abstract representations
- Learning = find optimal weights from data

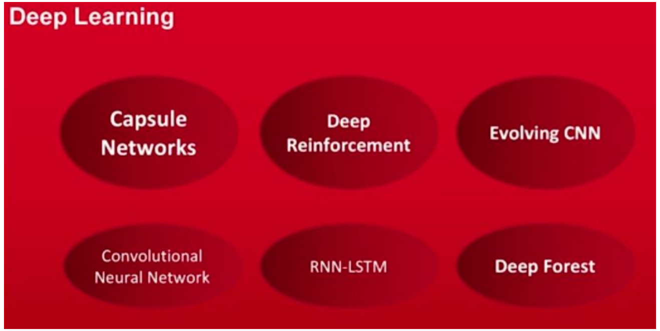


Labs247 Technology in motion

Deep Learning is a HOT Topic







Machine Learning for Big Data
Dr. Suyanto - Fakultas Informatika Telkom University

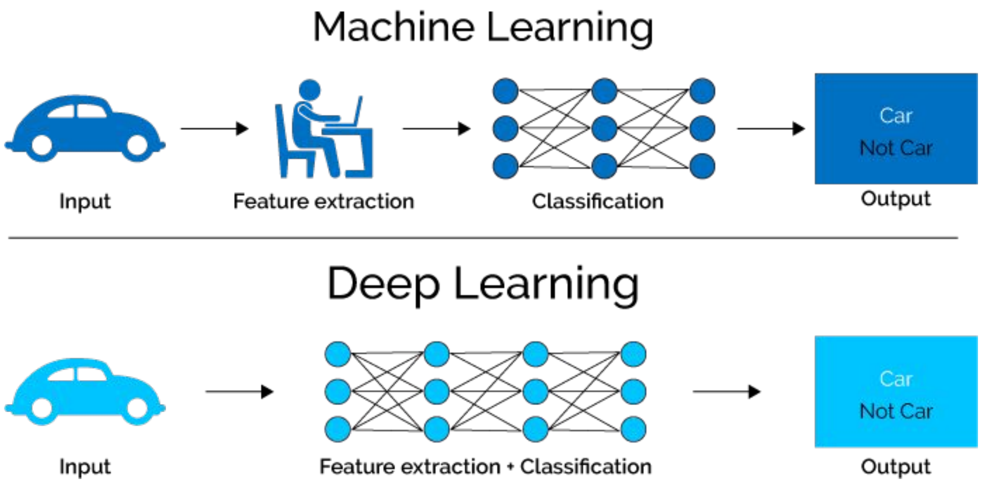
MeetUp idBigData #22 - Bandung, 27 February 2019
Source : https://youtu.be/eG04XqkB4pk

Labs247 Technology in motion

Why Deep Learning

- Manually designed features are often over-specified, incomplete and difficult to design and validate. Learned Features are easy to adapt, fast to learn
- Deep learning provides a very flexible, (almost?) universal, learnable framework for representing world, visual and linguistic information.
- Insufficiently deep architectures can be exponentially inefficient
- Distributed representations are necessary to achieve non-local generalization

Deep Learning vs Classical Machine Learning



- In classical machine learning, most of the features used require identification of domain experts
- Deep networks scale much better with more data than classical ML algorithms
- Deep learning techniques can be adapted to different domains and applications far more easily than classical ML

Why Now?

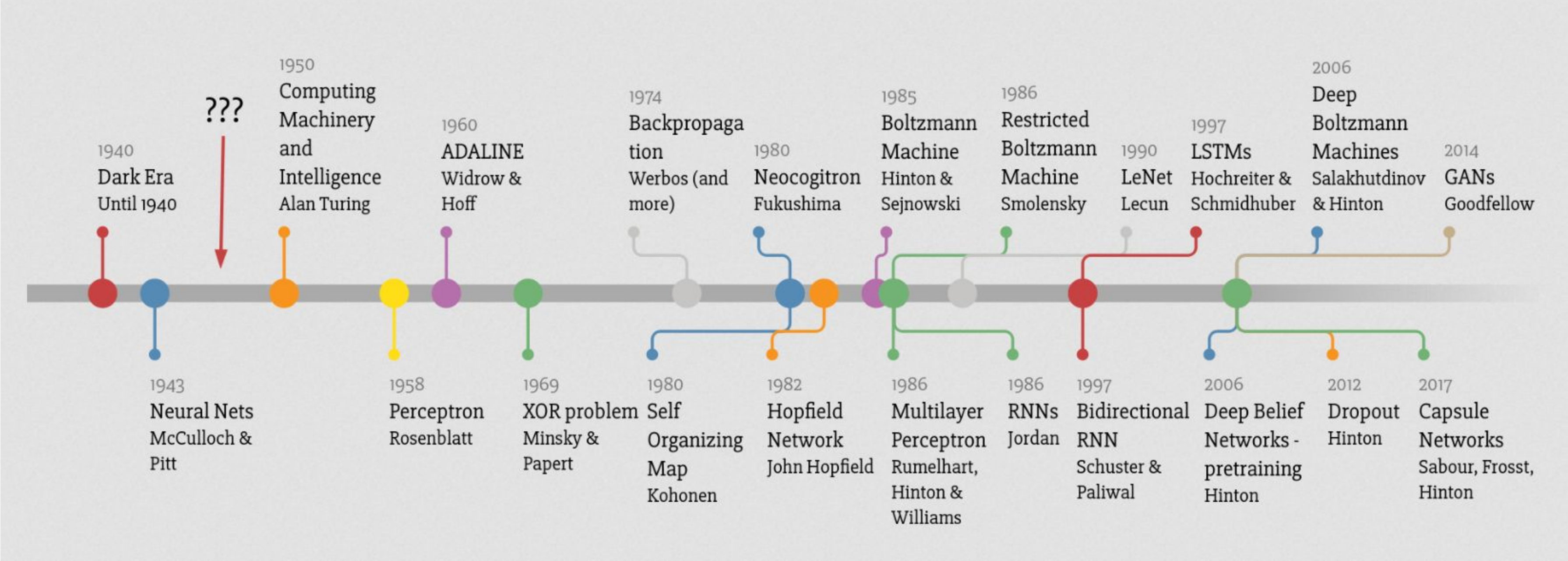
- Exponential data growth (and the ability to Process Structured & Unstructured data)
- Faster & open distributed systems (Hadoop, Spark, TensorFlow, …)
- Faster machines and multicore CPU/GPUs
- New and better models, algorithms, ideas:
  - Better, more flexible learning of intermediate representations
  - Effective end-to-end joint system learning
  - Effective learning methods for using contexts and transferring between tasks

"The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms." - Andrew Ng

When To Use Deep Learning ?

- If the data size is large
- Need to have high end infrastructure to train in reasonable time
- Complex feature introspection
- Complex problems such as image classification, natural language processing, and speech recognition

Deep Learning Timeline



Source: Favio vazquez

Use Case - Self-Driving Cars

- Uber first announced its intentions to amass a fleet of automatic cars in February 2015
- The cars themselves were packed with around 20 cameras, seven lasers, a GPS, radar and lidar, a technology that measures the distance reached by outgoing lasers so cars can “see” and interpret the action around them.
- In March 2018, an Uber self-driving car in Tempe, Arizona struck a pedestrian who was walking outside of a crosswalk at night.
- Waymo—formerly the Google self-driving car project- paid driverless taxi service could launch in December 2018

Labs247 Technology in motion

Use Case - Machine Translation

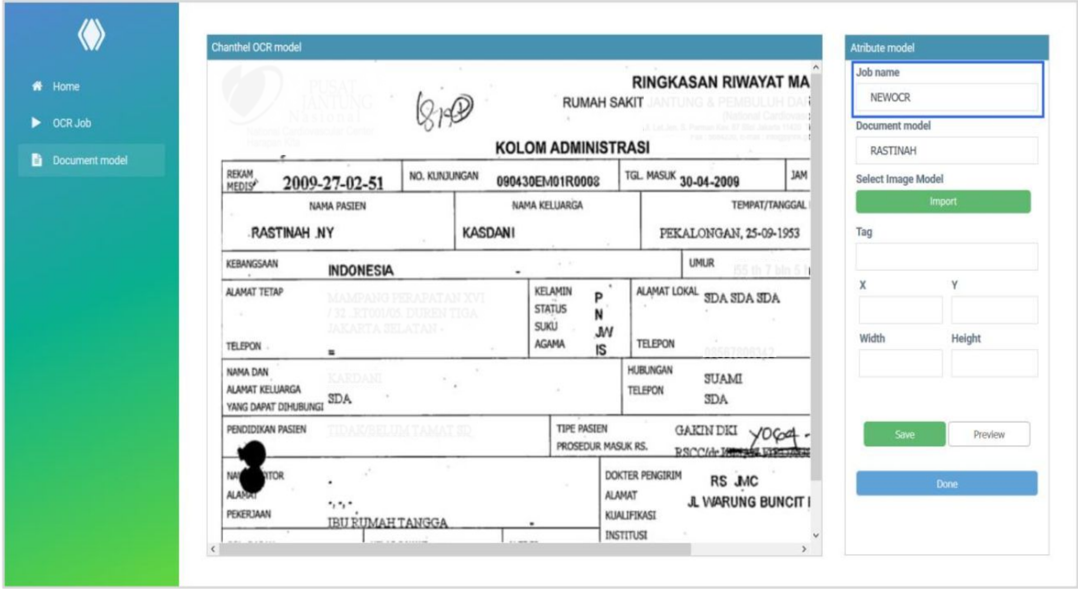
- DeepL Translator is a translation service launched in August 2017 by DeepL GmbH
- Promising to deliver 3x better results compared to Microsoft Translator and Google Translate

Use Case - Colorization Images

- Paper : ColorUNet: A Convolutional Classification Approach to Colorization
- A final project of Computer Vision courses at Stanford University
- Using Convolution Neural Network method classification to colorize grayscale images.

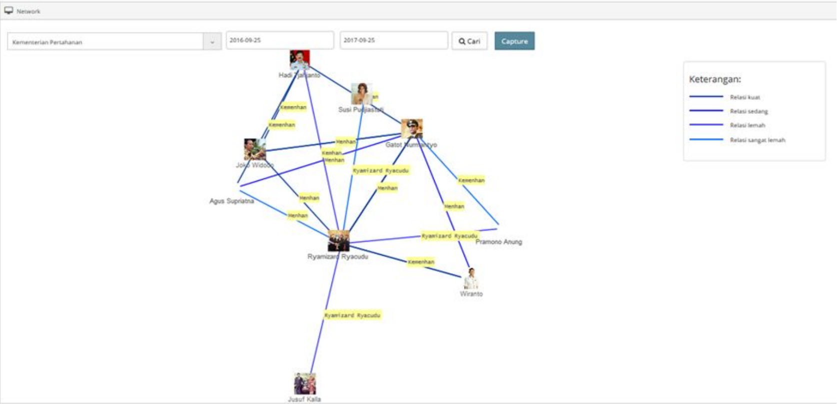
Chanthel247 - Document Management System

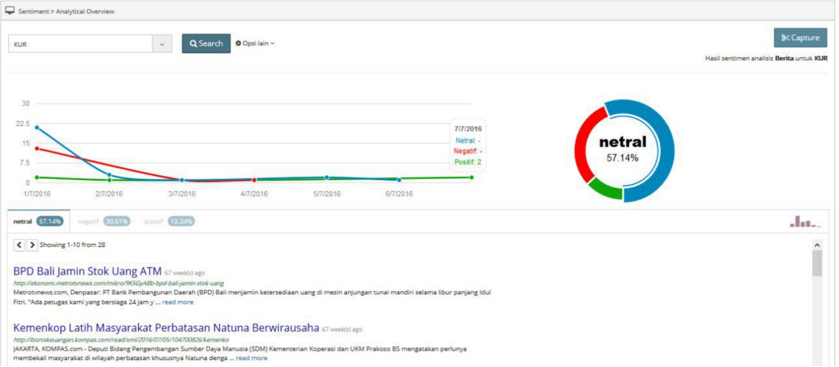
- Chanthel247 - Document Management System
  - Read a Form to get the content

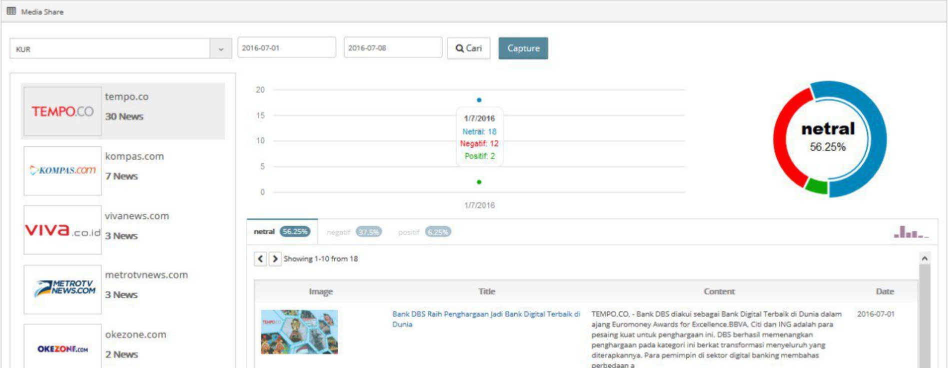


Labs247 Technology in motion

Medan247 - Media Analytics







Medis247 - Big Data Healthcare Platform

Easy to maintenance
Easy to integrate
Easy to retrain
Extendable and plugins
More collaboration


Accurate for pediatric
Excellent UX for doctor

Electronic Medical Records

Voice Interface

Doctor Notes

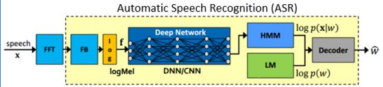
Patient Data

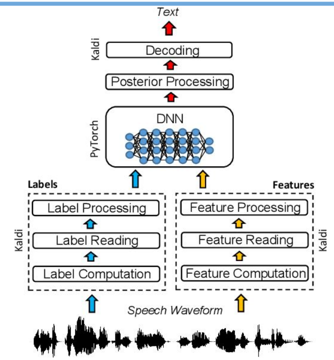
Image/Files

Documents

Medis247

ASR Interface





BPPT

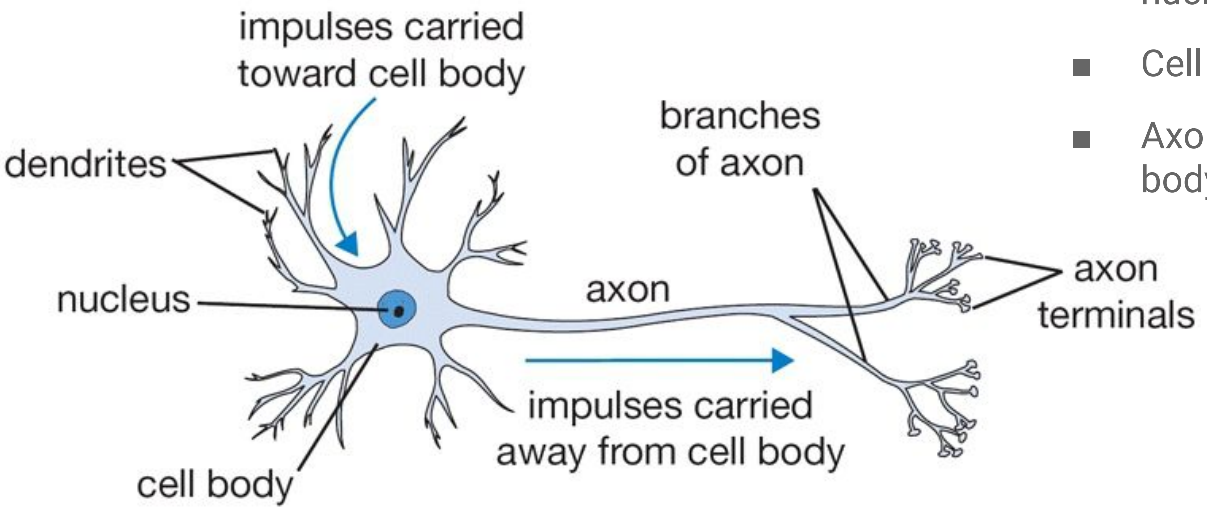
Labs247 Technology in motion

PUSAT JANTUNG Nasional
National Cardiovascular Center Harapan Kita

CHAPTER 02

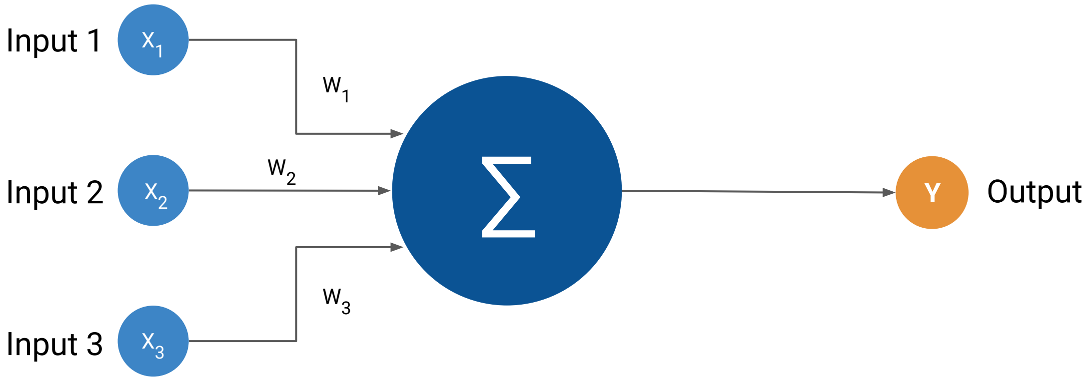
Artificial Neural Network

Anatomy of A Neuron



- Dendrite receives signal from other nucleus
- Cell Body sums all the inputs
- Axon pass message away from cell body to other neuron

How It Work - Perceptron



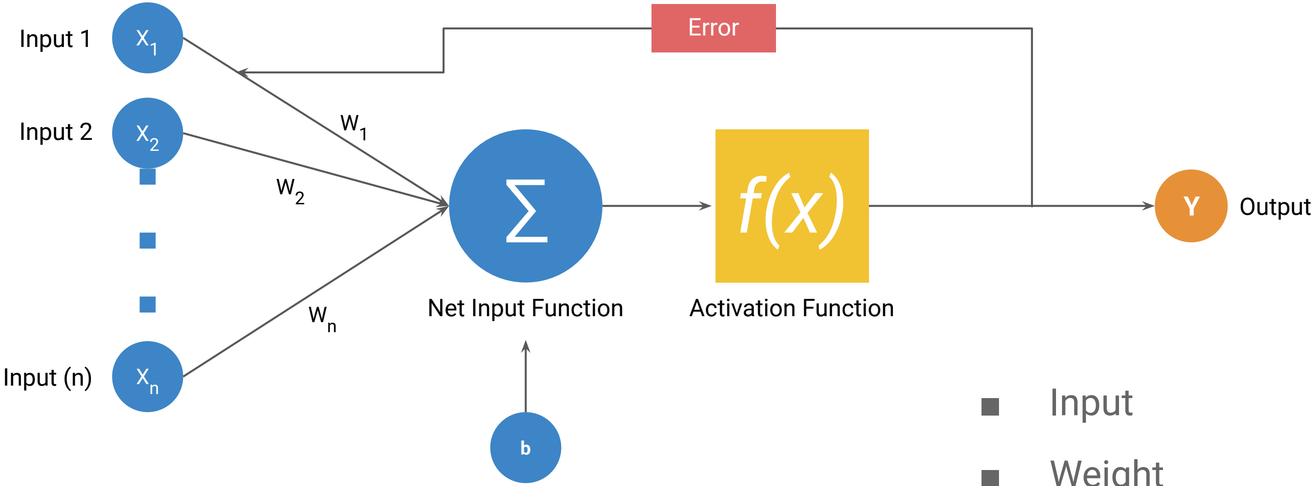
$$y = (x_1w_1) + (x_2w_2) + (x_3w_3)$$

$$= \sum_i x_i \; w_i$$

$$= [w_1 \;\; w_2 \;\; w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

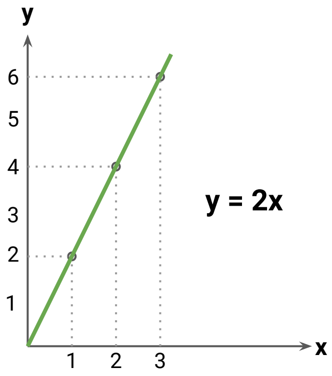
- Perceptron is the basic part of a neural network, which represents a single neuron
- A neuron is a computational unit that calculates a piece of information based on weighted input parameters
- Inputs accepted by the neuron are separately weighted.
- Weights are real numbers expressing the importance of the respective inputs to the output and can be adjusted during learning phase

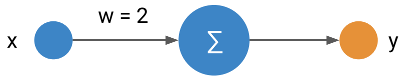
Component Of Artificial Neural Networks

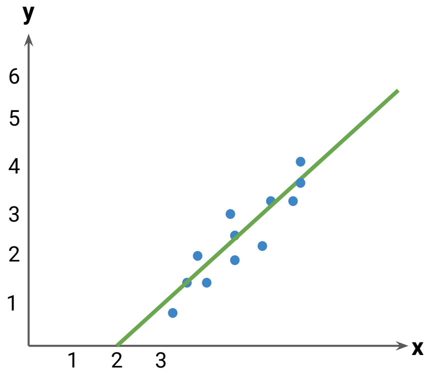


- Input
- Weight
- Bias
- Activation Function
- Output

How It Work - Bias







- A bias value allows you to shift the activation function to the left or right, which may be critical for successful learning.
- Changes in weight change the steepness of the curve, while bias shifts the entire curve so that it is more suitable.
- Bias only affects the output value, it does not interact with the actual input data
- Bias value can be adjusted during learning phase

$$y = \sum(x_i w_i) + b$$

Activation Functions

- Decides whether a neuron should be activated or not by calculating weighted sum and adding bias with it.
- Introduce non-linearity into the output of a neuron → making it capable to learn and perform more complex tasks
- If we do not use activation function, the output signal produced is only a linear function

Linear function

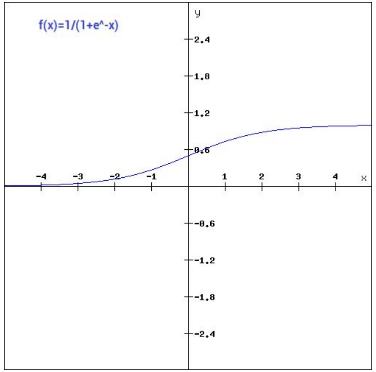
Non-linear function

Activation Functions

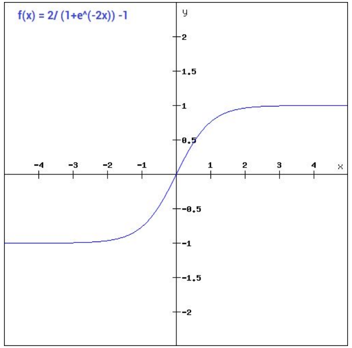
$$y = f(\sum_i x_i w_i + b)$$

- Commonly used activation functions:
  - Sigmoid function : $A = \frac{1}{1+e^{-x}}$
  - Tanh function : $tanh(x) = \frac{2}{1+e^{-2x}} - 1$
  - ReLU (Rectified Linear Unit) : A(x) = max(0,x)
  - Softmax function : converts an array of values into an array of probabilities (0 - 1)

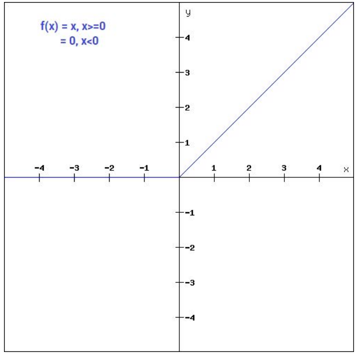
$$softmax(n) = \frac{\exp n_i}{\sum \exp n_i}$$



Sigmoid function

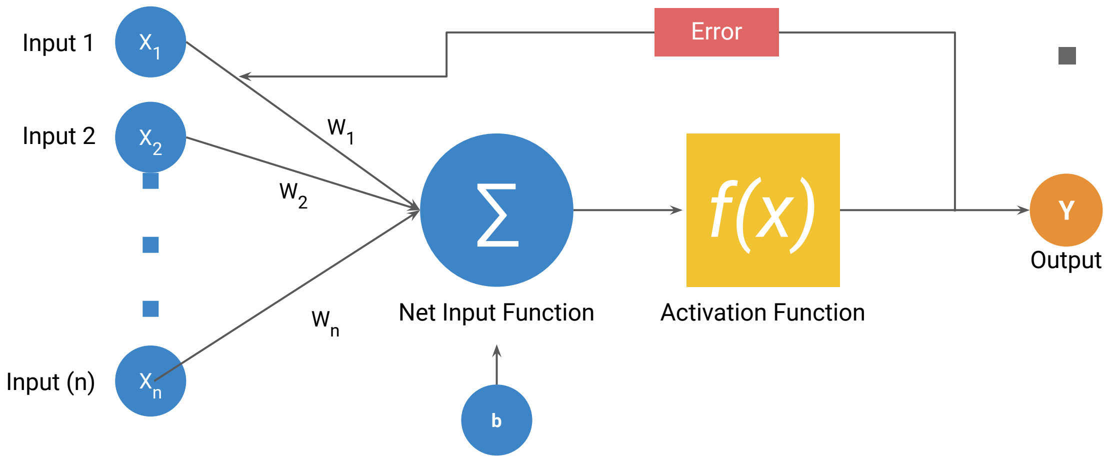


tanh function



ReLu function

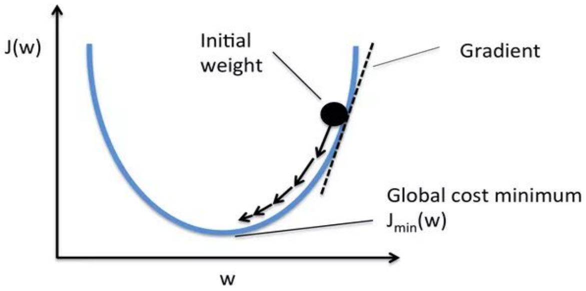
Learning Rule - 2 Steps of Learning



- **Forward propagation**
  - weights and bias are initialised randomly
  - calculate forward (from input layer to output layer) to get the output
  - compare it with the real value to get the error → using **lost function or cost function**

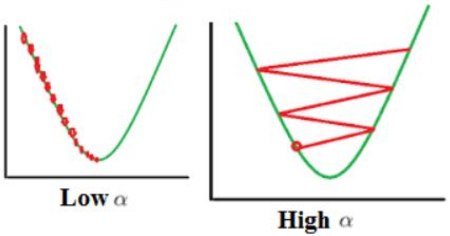
- **Backward propagation**
  propagate backwards and do:
  - finding the derivative of the **lost function** with respect to each weight
  - update the weight by subtracting this value from the weight value.

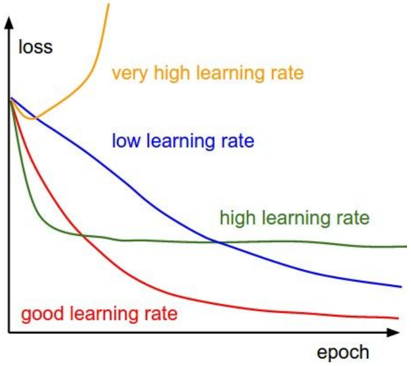
Loss Function and Gradient Descent



- Loss function : a method of evaluating how well your algorithm models your dataset.
- Learning = minimizing loss function
- How = change the weights* by calculating the **gradient** (i.e. the partial derivative of loss function with respect to weights)
- Since we want the value to be minimum → **gradient descent**. This is the simplest and most popular optimization method for deep neural network
- Other methods : Newton's method, Conjugate Gradient, Quasi-Newton, Levenberg-Marquardt algorithm.

Learning Rate



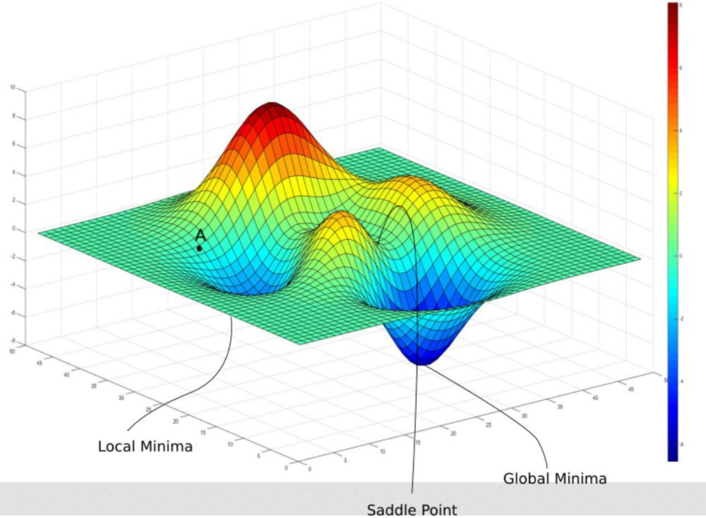


the effects of different learning rates

- The gradient told us the **direction** to change the weight.
- **How much** we must change is determined by another *hyperparameter* called **learning rate** or $\boldsymbol{\alpha}$
- Picking the value of $\boldsymbol{\alpha}$ is crucial
    - too small → training process too slow
    - too big → diverging away from the minimum / overshoot
- We can pick $\boldsymbol{\alpha}$ manually or using optimization technique that utilize adaptive learning rate (e.g. Adagrad, RMSProp, Adam)
- For example:

$$w' = w - \boldsymbol{\alpha} (\partial E/\partial w) \quad \rightarrow \boldsymbol{\alpha} \text{ is learning rate}$$

Gradient Descent Implementation

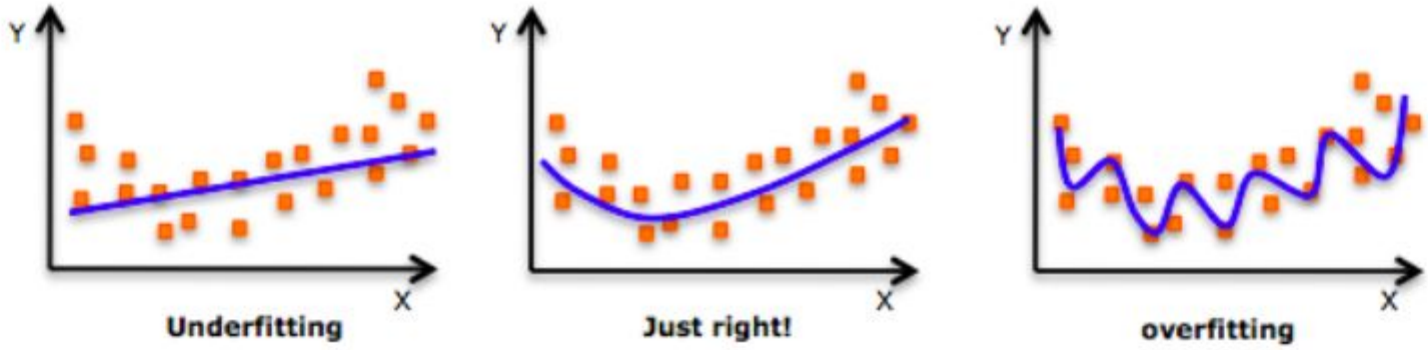


Challenges in Gradient Descent

When we should update the weight? → there are 3 types of gradient descent implementation

1. **Batch Gradient Descent** : process all training dataset in a single batch, calculate the error and update the weight accordingly.
2. **Stochastic Gradient Descent** : randomized the data, and update the weights for each training instance.
3. **Minibatch Gradient Descent** : splits the training dataset into small batches, calculate error and update weights for each small batches.

Overfitting and Regularization



- The complicated nature of deep neural network makes it prone to overfitting.
- A model is overfit if it performs well on training data but not generalize well on new unseen data
- Regularization is any modification we make to the learning algorithm that is intended to reduce the generalization error, but not its training error (Ian Goodfellow)
- The purpose is to control model complexity and reduce overfitting

Most Common Regularization Technique

- Weight penalty L2 & L1

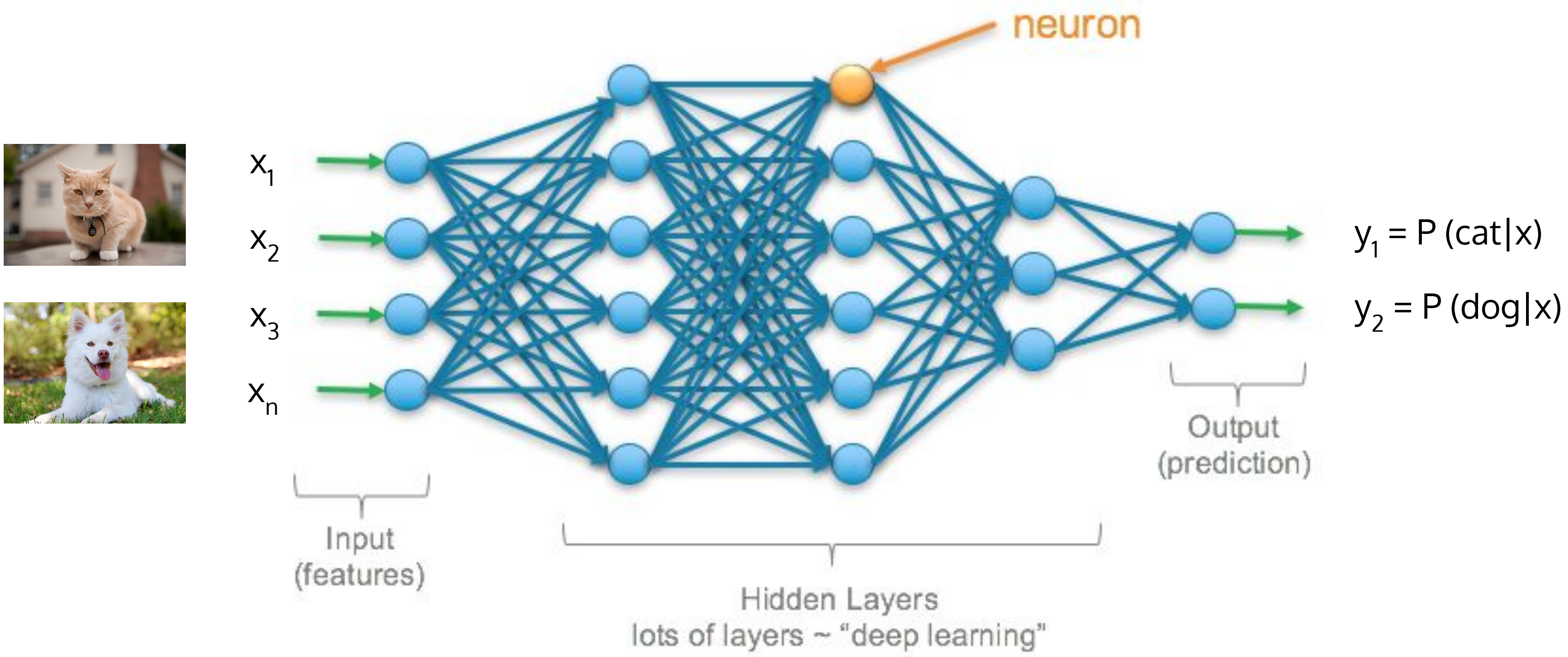
  Loss = Loss + Regularization

  $L2 \longrightarrow Loss = Loss + \boldsymbol{\alpha} \sum w^2$

  $L1 \longrightarrow Loss = Loss + \boldsymbol{\alpha} \sum w$

- Dropout
  - At each training iteration a dropout layer randomly removes some nodes in the network along with all of their incoming and outgoing connections.
- Early stopping
  - Interrupting the training procedure once model’s performance on a validation set gets worse
- Dataset augmentation
  - Use bigger dataset by using synthetic data

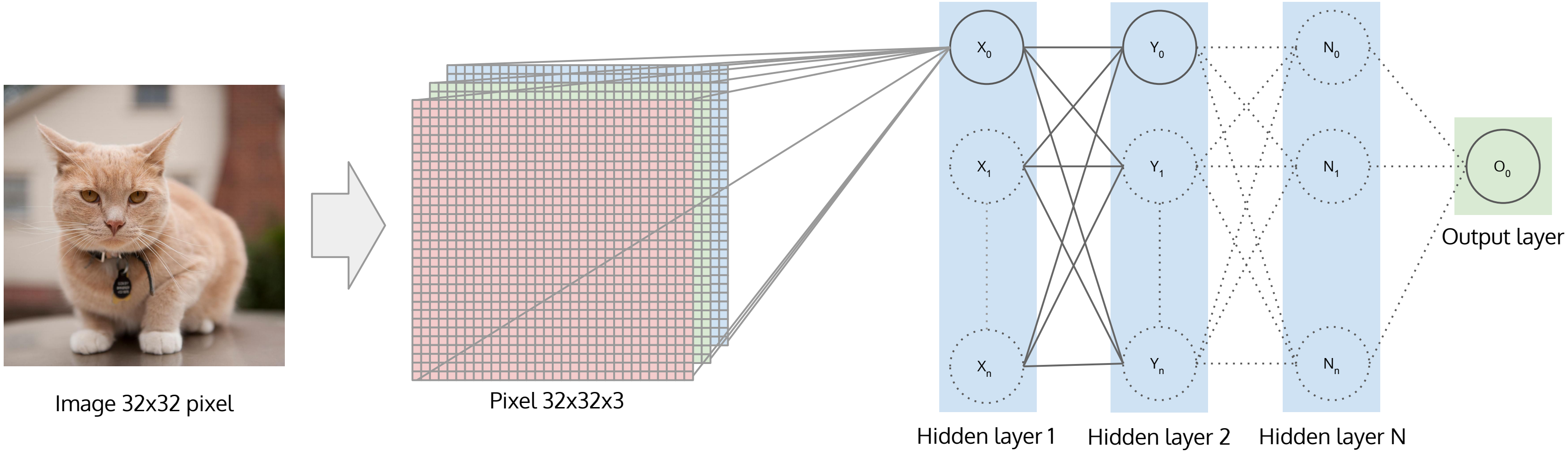
Multilayer Perceptron



CHAPTER 03

Convolution Neural Network

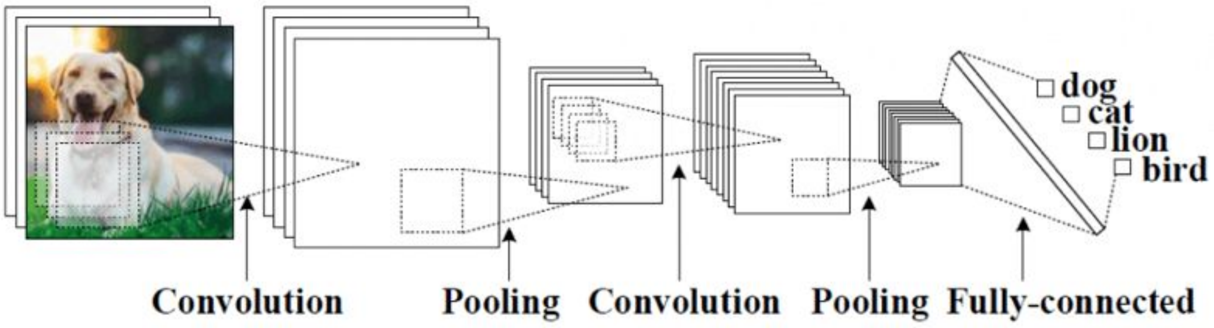
Image Processing in ANN



- Regular Neural Nets don't scale well to full images
  E.g : CIFAR-10 images size 32x32x3 (32x32, 3 color channels - RGB) → 32*32*3 = **3072 weights** in a single neuron in a first hidden layer
  For 200x200x3 image → 200*200*3 = **120,000 weights**

- Parameters would add up quickly → quickly lead to overfitting.
- In order to reduce the number of parameters, it is used by using the convolution method

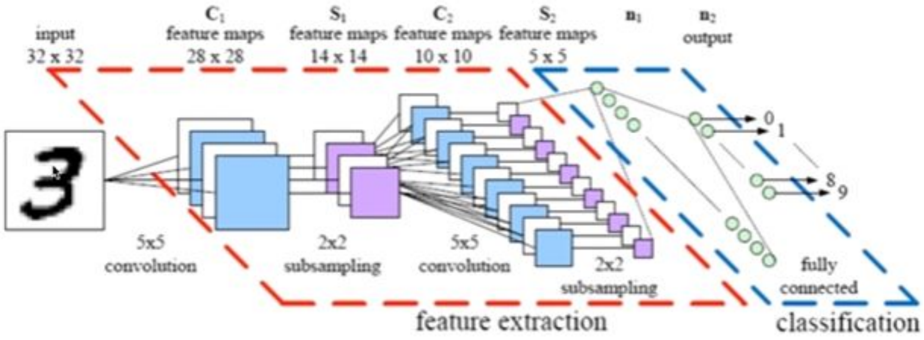
Convolutional Neural Network Overview



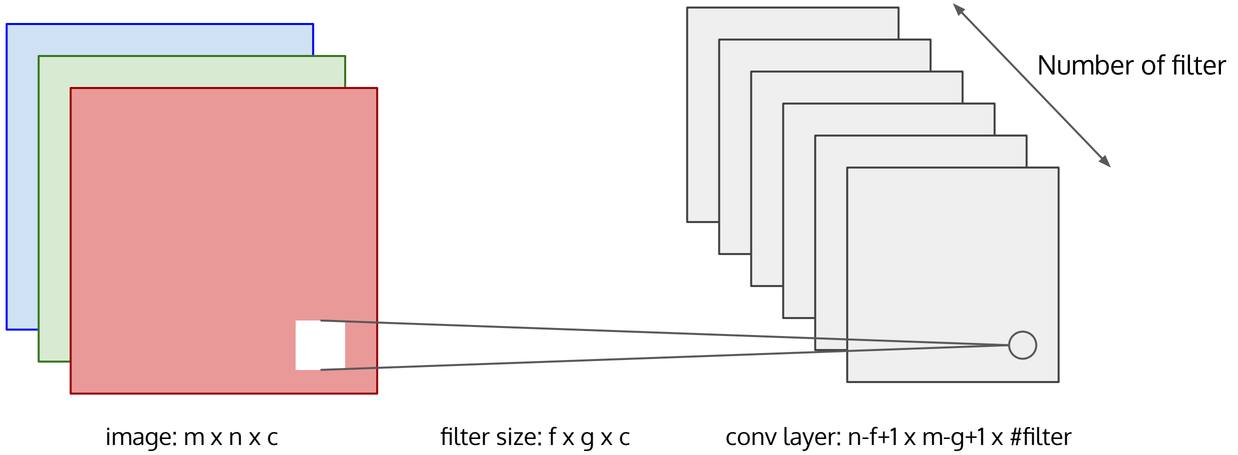
- A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network.
- in 1998, Convolutional Neural Networks were introduced in a paper by Bengio, Le Cun, Bottou and Haffner.
- Their first Convolutional Neural Network was called LeNet-5 and was able to classify digits from hand-written numbers.

CNN Layers

- The common types of layers in CNN architecture : Convolution - Pooling - Fully Connected
  - Convolution layer is the core that does most of the computation
  - Pooling layer performs dimensionality reduction and controls overfitting. Commonly puts between the convolution layers. The Convolution and Pooling layer acts as feature extraction part
  - Fully Connected layer generally ends the CNN. It acts as classification part

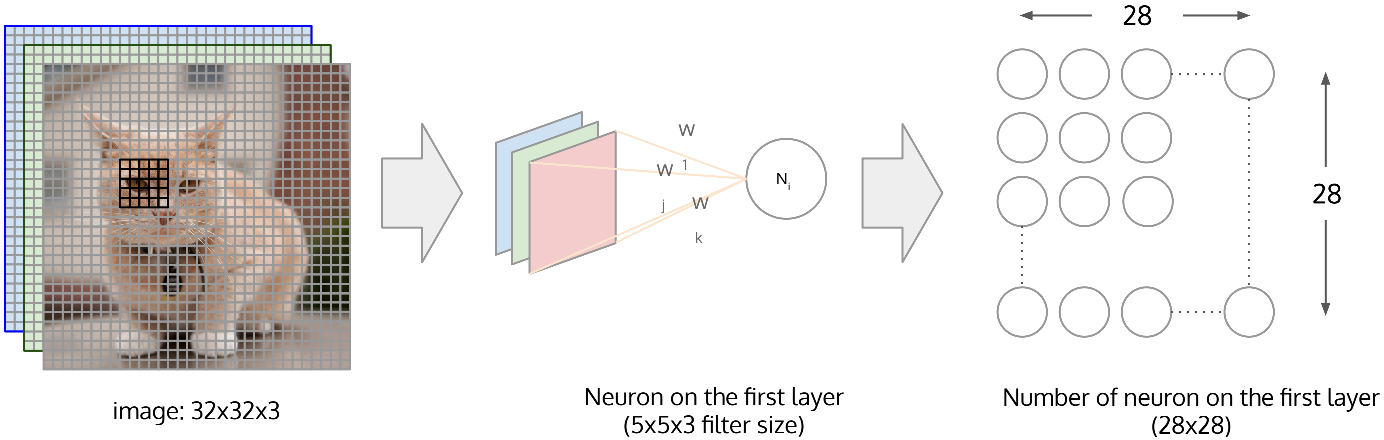


Convolution Layer Hyperparameter



- There are 3 hyperparameters to set in the conv layer: number of filter, stride, and padding
  - Filter size
  - Number of filter
  - Stride
  - Padding

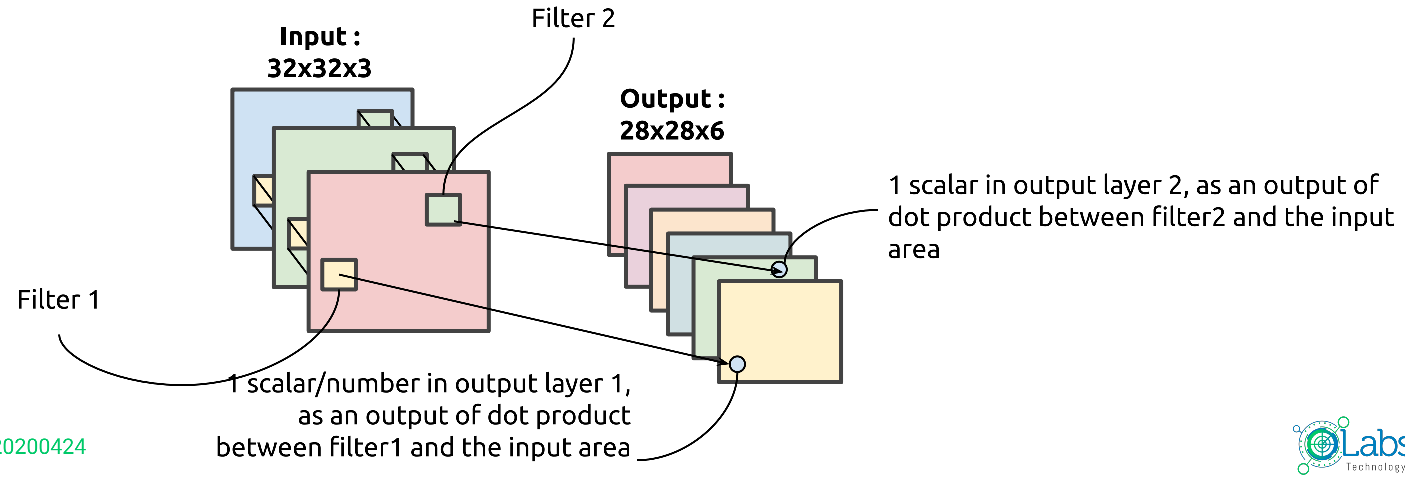
Convolution Layer



- The first layer of CNN is always convolution layer
- Convolution layer consist of a set of learnable **filters** (also called **kernel**, or weight)
- Filter is a small array of number, and covers all the input depth.
    - For the CIFAR-10 data, we may have a 5x5x3 filter (5x5 matrix for 3 color channels)
    - Each neuron will have weights to a [5x5x3] region in the input volume → 5*5*3 = 75 weights → this is what we will learn
- The filter will convolve around the image, performing dot product operation between the filter and the part of the image it convolves with

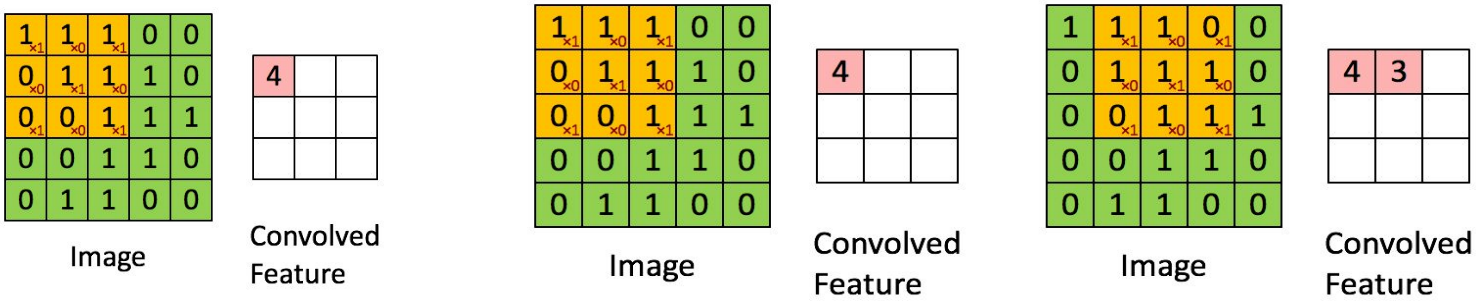
Number of Filters

- The number of filters called **depth column** (or **fibre**) → *note that it's different from **input depth***.
- We may have multiple filter for a conv layer, each layer learns different features (e.g. straight edges, blobs of color, curves, etc.)
- The 3rd dimension of output layer of a conv layer = the number of filters
- If we use 6 filters of 5x5x3 for our example, the output will be 28x28x6 (6 layers of 28x28)



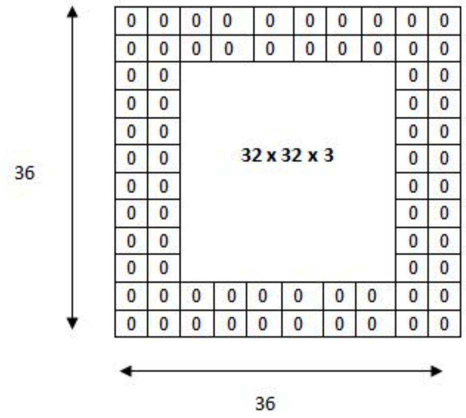
Stride

- Stride = how much we shift the filter at a time. The bigger the stride, the smaller the output.
- Formula to calculate output width and height = (W−F)/S+1, where W = input size (width or height), F = filter size (width or height), and S = stride
- Stride is normally set in a way so that the output volume is an integer and not a fraction.
- For stride = 3, the output will be (32-5)/3+1 = 10x10
- Example for 7x7 input with 3x3 filter :



Padding

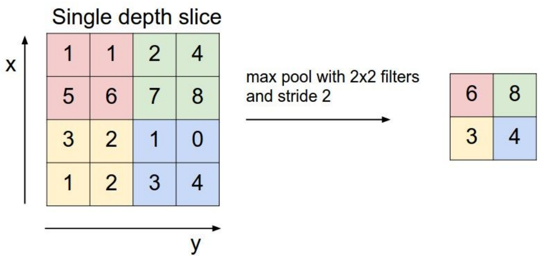
- Used to preserve the size of the output.
- The formula : (W − F + 2P) / S + 1, where P = padding
- If we want to preserve the output spatial size to 32x32 with our 5x5 filter, we can use padding=2 and stride=1, so output = (32 - 5 + 4) + 1 = 32



The input volume is 32 x 32 x 3. If we imagine two borders of zeros around the volume, this gives us a 36 x 36 x 3 volume. Then, when we apply our conv layer with our three 5 x 5 x 3 filters and a stride of 1, then we will also get a 32 x 32 x3 output volume.

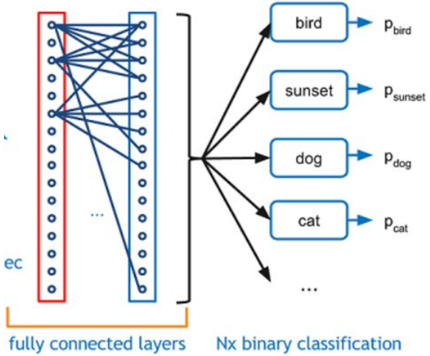
Pooling Layer

- Also called downsampling layer.
- Reduce the spatial size to reduce the amount of parameters and computation, and to control overfitting.
- It operates independently on every layer/ depth slice of the input → output depth = input depth
- The most common is : 2x2 with stride = 2. Other common setting is F=3x3 ,S=2 (overlapping pooling).
- Most common operation = max pooling. Other less common operation = average pooling



Fully Connected Layer

- Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks.
- Usually put at the end of the CNN architecture to perform the classification.
- FF layer outputs an N dimensional vector where N is the number of classes that the program has to choose from.
- Many newer CNN architectures doesn’t use Pooling and/or Fully Connected Layer



LAB 01

Train Your First CNN

Lab Description

What you will learn:

1. Coding by using basic tensorflow and keras framework
2. Creating simple Convolution Neural Network model
3. Training and testing CNN model

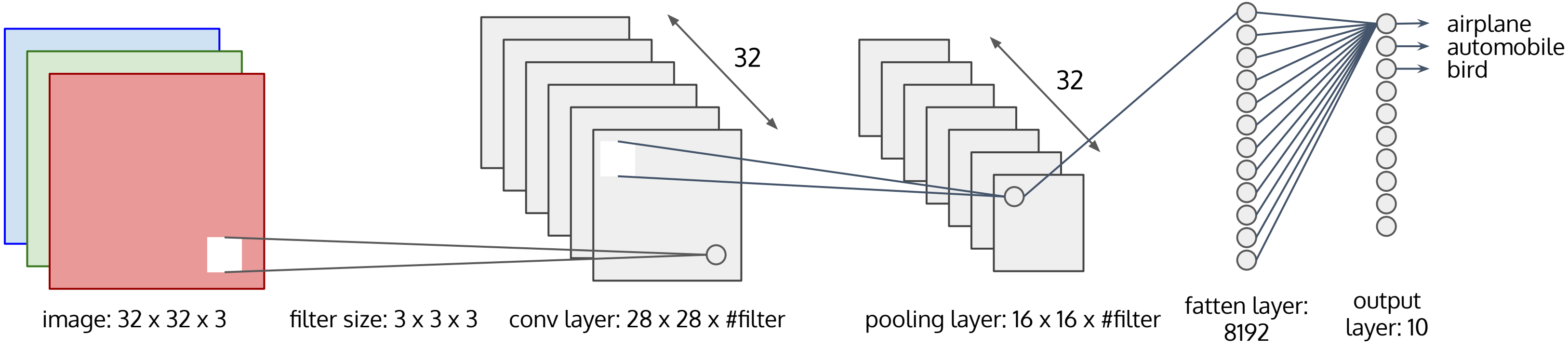
Requirement :

1. Anaconda
2. Python 3
   - numpy
   - pandas
   - Matplotlib
   - scipy
   - jupyter
   - tensorflow
   - keras

CIFAR-10 Dataset

- **The dataset** : CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes, commonly used to train machine learning and computer vision algorithms.
- **The problem** : image classification
- **Input format** : 32x32x3 arrays of numbers (32x32 RGB image)
- **Output** : the probability of the image being a certain class (e.g. 0.80 for cat, 0.15 for dog, 0.05 for bird, etc)

Neural Network Architecture



STEP 01 - Execute Notebook

- Open notebook : Simple CNN Model - Cifar10.ipynb
- Follow the instructor to run notebook
- Instructor will explain step by step process

Performance Tuning

Performance Improvements :

To achieve the best performances we may:

1. **Fine Tune Hyper-Parameters :** Hyper-parameters are the variables which are set before training and determine the network structure & how the network is trained. (eg : learning rate, batch size, number of epochs). Fine tuning can be done by : *Manual Search, Grid Search, Random Search…*

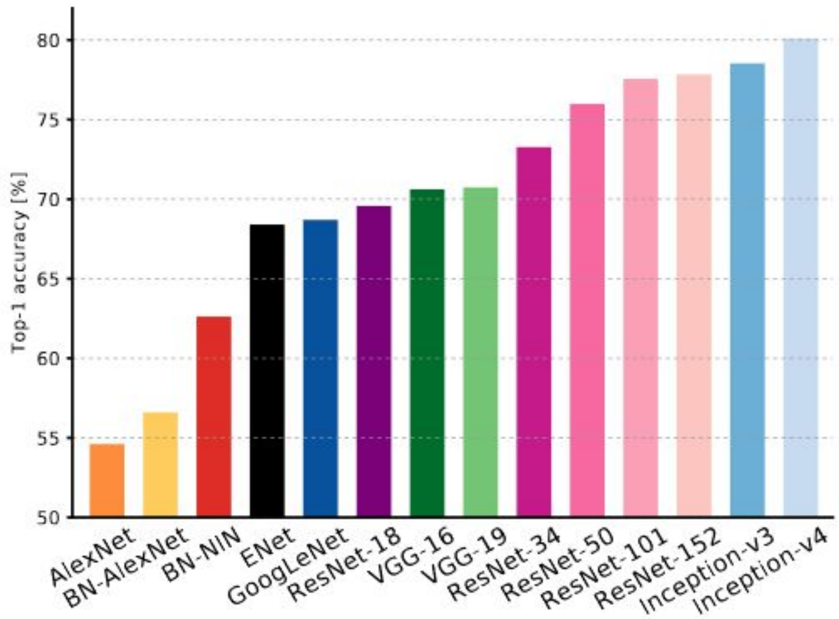
2. **Improve Pre-Processing :** Better pre-processing of input data can be done as per the need of your dataset like removing some special symbols, numbers, stopwords and so on …

3. **Use Dropout Layer :** Dropout is regularization technique to avoid overfitting (increase the validation accuracy) thus increasing the generalizing power.

CHAPTER 04

CNN State Of Art

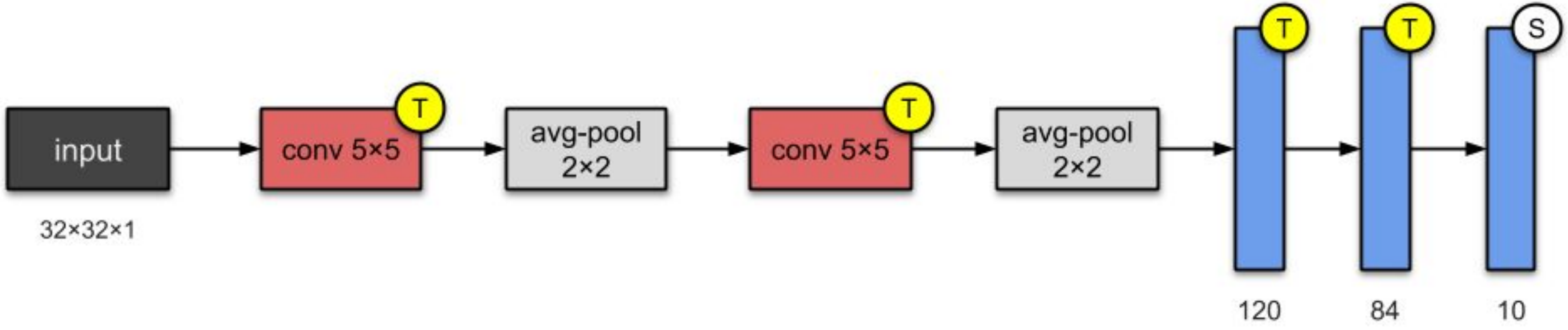
CNN - State of The Art Architectures



- There are many of models shared by community
- Chart above is benchmark various model by using imagenet dataset
- Source: An Analysis of Deep Neural Network Models for Practical Applications, Alfredo Canziani, et all

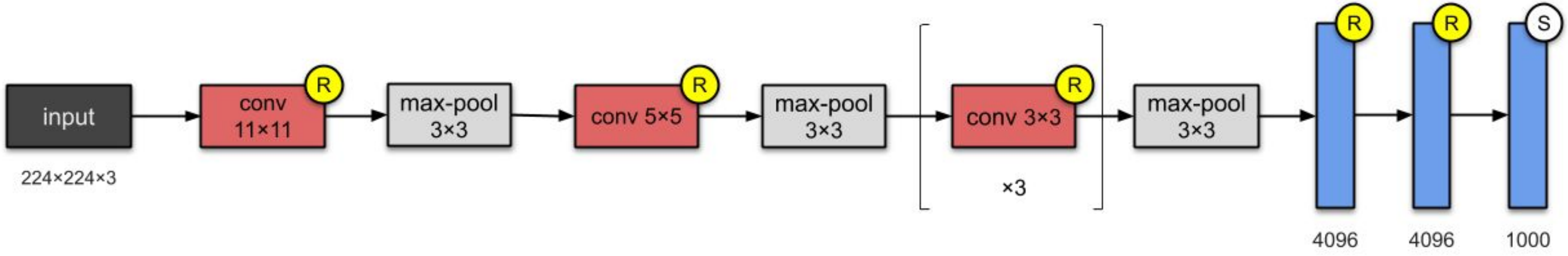
Labs247 Technology in motion

LeNet 5 (1998)



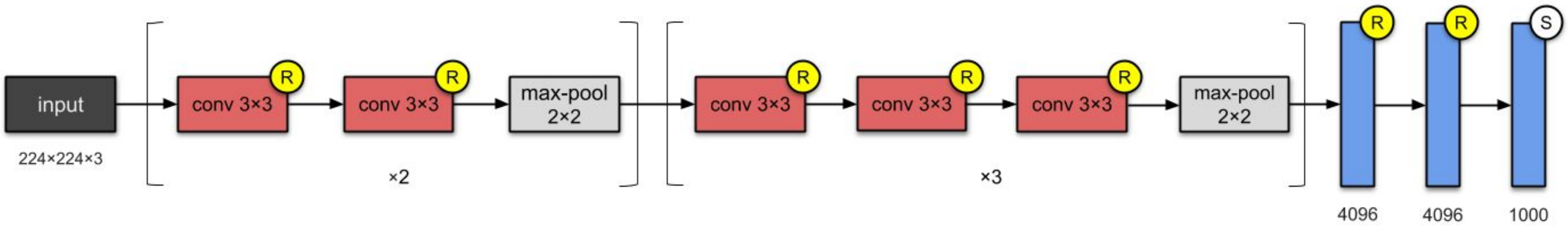
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998), *Gradient-Based Learning Applied to Document Recognition*
- Classifying single number handwriting, 32x32x1 pixels
- LeNet-5 has 2 convolutional and 3 fully connected layers. It has trainable weights and a sub-sampling layer (now known as the pooling layer). LeNet5 has about 60,000 parameters

AlexNet (2012)



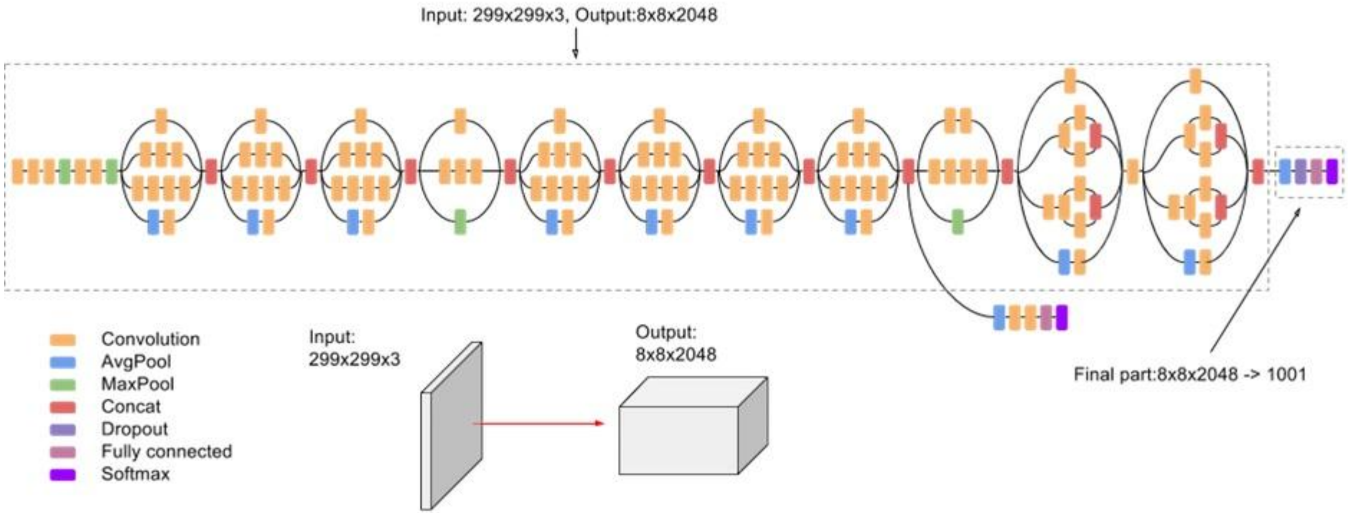
- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. University of Toronto, Canada. (2010), *ImageNet Classification with Deep Convolutional Neural Networks*
- Introduce Rectified Linear Units (ReLUs) as activation functions
- 60M parameters, AlexNet has 8 layers — 5 convolutional and 3 fully-connected. AlexNet just stacked a few more layers onto LeNet-5
- AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs

VGG-16 -(2014)



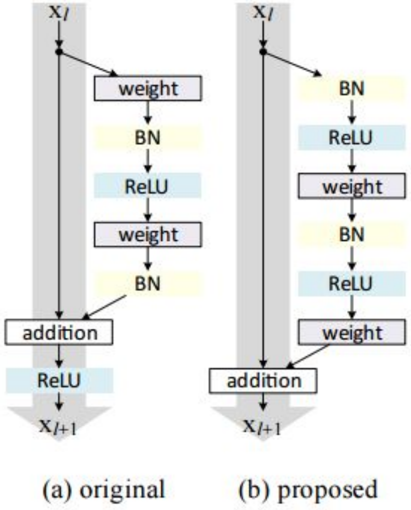
- Karen Simonyan, Andrew Zisserman. University of Oxford, UK (2016), *Very Deep Convolutional Networks for Large-Scale Image Recognition*
- VGG-16 has 13 convolutional and 3 fully-connected layers. It used ReLUs as activation functions, just like in AlexNet. VGG-16 had 138 million parameters.

Inception (2014)



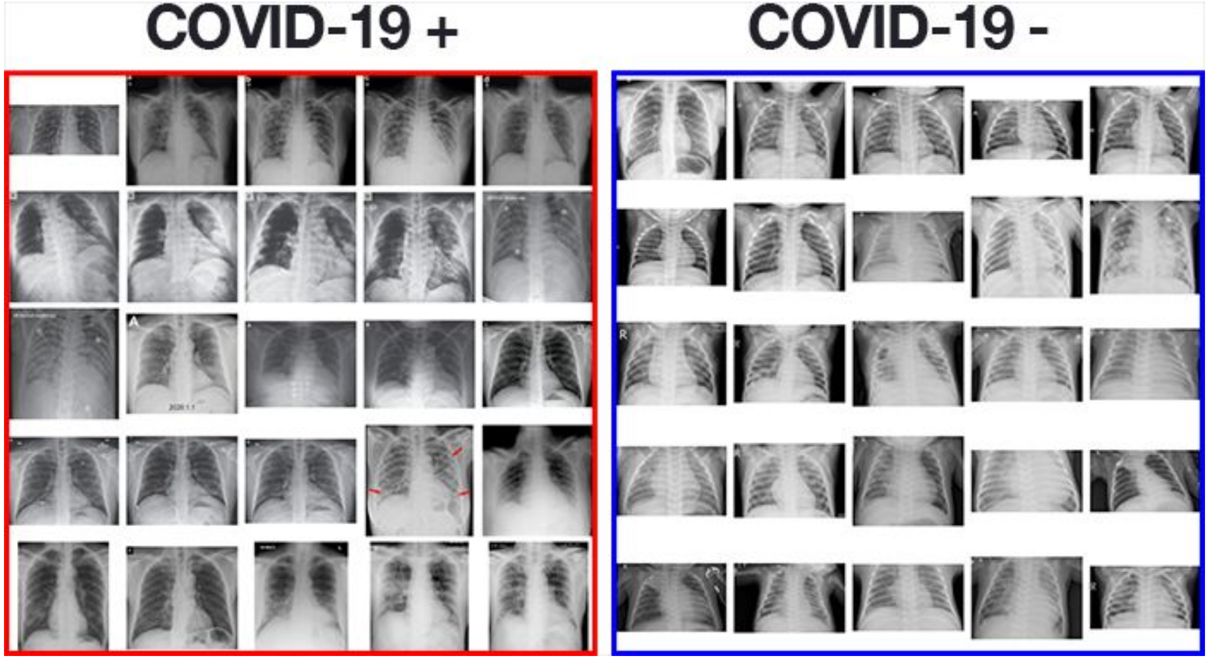
- Szegedy et al. (2014), *Going Deeper with Convolutions*: and Inception V3 (2015) *Rethinking the Inception Architecture for Computer Vision*
- The goal of the inception module is to act as a "multi-level feature extractor" by computing *1×1*, *3×3*, and *5×5* convolutions within the *same* module of the network — the output of these filters are then stacked along the channel dimension before being fed into the next layer in the network.

ResNet-50 (2015)



- He et al. (2015) , *Deep Residual Learning for Image Recognition* and (2016) *Identity Mappings in Deep Residual Networks*
- Relies on micro-architecture modules (also called “network-in-network architectures”). are then followed by a softmax classifier (above).
- Much deeper than VGG16 and VGG19, the model size is substantially smaller due to the usage of global average pooling rather than fully-connected layers — this reduces the model size down to 102MB for ResNet50 (50 layers and 26M parameters)

Challenge - Covid19 Chest X-ray



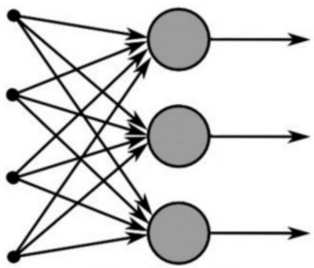
Dataset: https://github.com/ieee8023/covid-chestxray-dataset

CHAPTER 05

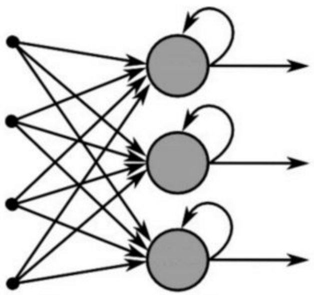
RNN and LSTM

What is RNN?

- A type of Neural Network where the ***output from previous step*** are **fed** as ***input to the current step*.**
- Regular or feed-forward Neural Network assumes the input (and output) to be :
  - Flows in one direction
  - Independent of each other
- In RNN information cycles through a loop. When it makes a decision, it considers **the current input** and **what it has learned** from the previous inputs
- RNN **remembers** the output of the previous steps



Feed-Forward Neural Network



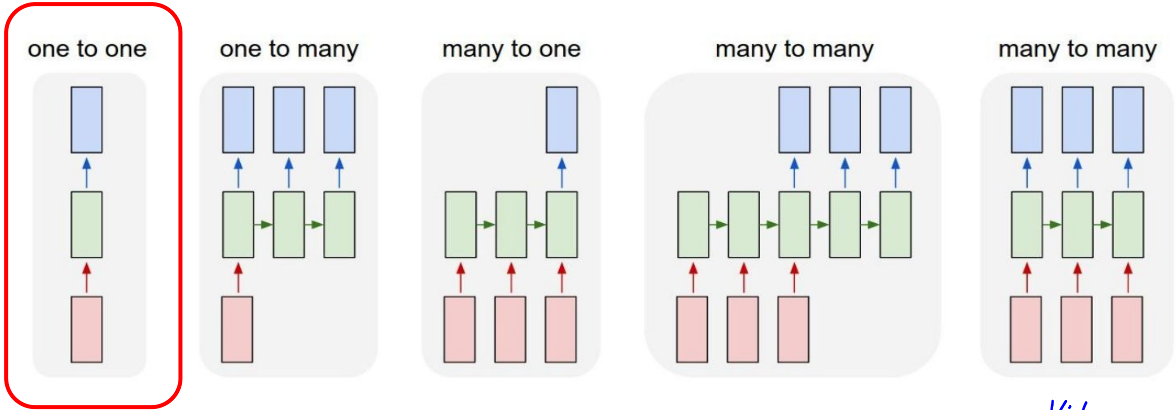
Recurrent Neural Network

Why RNN?

- To solve problems with sequential data

  Such as : time series data, speech, text, financial data, audio, video, weather, etc.

- RNN offers flexibility in terms of input and output size



vanilla neural network

Image captioning

Sentiment analysis

Machine translation

Video classification on frame level

Image from : http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

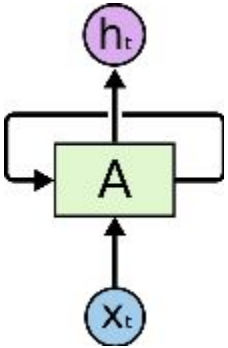
Why RNN?

- Consider the following problems :
    - Saya kembali ke rumah → I am going back … (*home* or *house*?)
    - Joe entered the room. Jane entered the room. Jane says hi to ...
- We need to remember the context (the previous words) to be able choose the right word.
- Regular NN can’t do this since it has no ‘memory’ and it considers the input as independent element

Labs247 Technology in motion

How RNN Works

- RNN has loops.
- It loops through the input sequence and process it one element at a time → the process is called **step**
- RNN takes input vector ***x*** and produce output vector ***y***
- But other than **x**, it also takes ***internal vector h*** as its input, which gets updated every time it process an input
- The transformation (i.e the function and the weight vector) is fixed for every step, just the input and the hidden state that are different

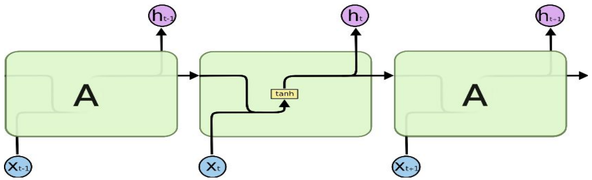


The Problem of Vanishing Gradient

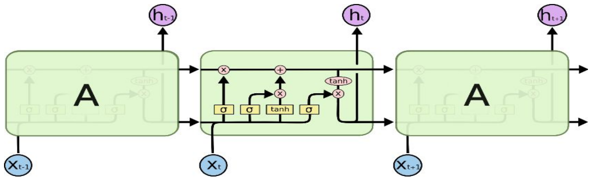
- RNN can’t ‘remember’ too far behind due to vanishing gradient problem.
- LSTM is used to solve this problem

Long Short Term Memory (LSTM)

- LSTM is a variant of RNN which capable of learning long term dependencies
- LSTM were introduced by [Hochreiter & Schmidhuber (1997)]
- The basic process is the same as RNN. The difference is in the process inside the cell
  - RNN has a single function (***tanh***)
  - LSTM has 4 function (***tanh, and 3 sigmoid functions)***, which often called :
    - **f**: Forget gate, Whether to erase cell
    - **i**: Input gate, whether to write to cell
    - **g**: How much to write to cell
    - **o**: Output gate, How much to reveal cell



vanilla RNN



LSTM

CHAPTER 06

Generative Adversarial Networks

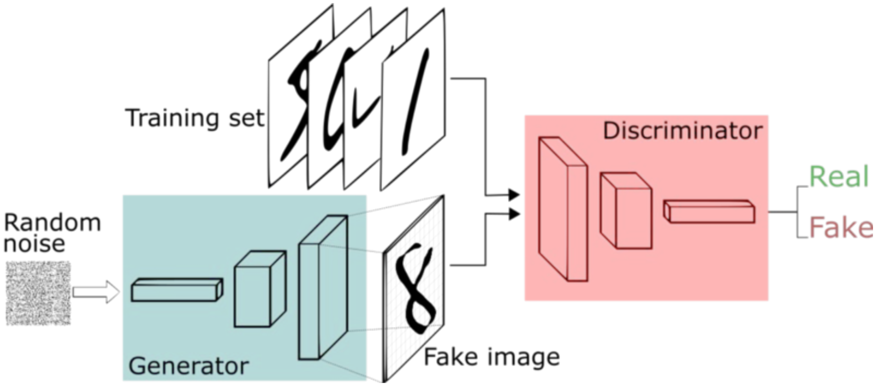
Overview

- Generative Adversarial Networks - Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio - 2014
- The main focus for GAN (Generative Adversarial Networks) is to generate data from scratch, mostly images but other domains including music have been done.
- Generative Adversarial Networks belong to the set of generative models. It means that they are able to produce / to generate new content.

"The most interesting idea in the last 10 years in Machine Learning" - Yann LeCun

How it works?

- The generator tries to produce data that come from some probability distribution.
- The discriminator acts like a judge. It gets to decide if the input comes from the generator or from the true training set.
- The generator trying to maximize the probability of making the discriminator mistakes its inputs as real.
- And the discriminator guiding the generator to produce more realistic images.



How it works?

- Generative Adversarial Networks - Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio - 2014
- The main focus for GAN (Generative Adversarial Networks) is to generate data from scratch, mostly images but other domains including music have been done.
- Afs

NVIDIA's Hyperrealistic Face Generator

Paper :
Progressive Growing Of GANS For Improved Quality, Stability, And Variation - NVIDIA

https://arxiv.org/pdf/1710.10196.pdf

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

Create Anime characters

Figure 7: Generated samples

Paper :
Towards the Automatic Anime Characters Creation with Generative Adversarial Networks

https://arxiv.org/pdf/1708.05509.pdf

Image to Image Translation

Phillip Isola, et al. in their 2016 paper titled “Image-to-Image Translation with Conditional Adversarial Networks” demonstrate GANs, specifically their pix2pix approach for many image-to-image translation tasks.

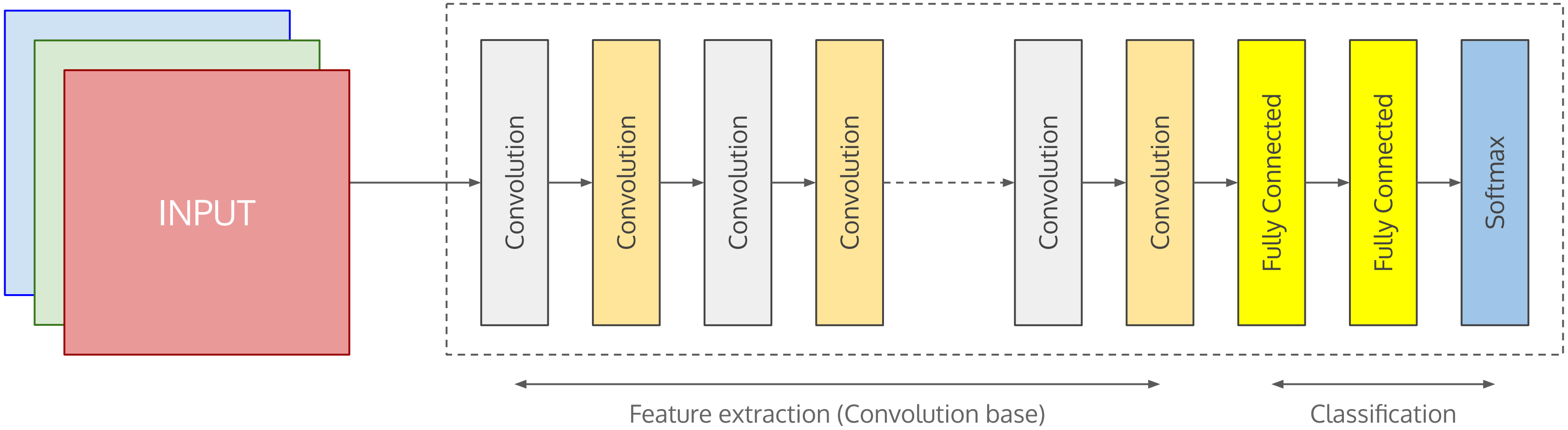
CHAPTER 04

Transfer Learning

Transfer Learning

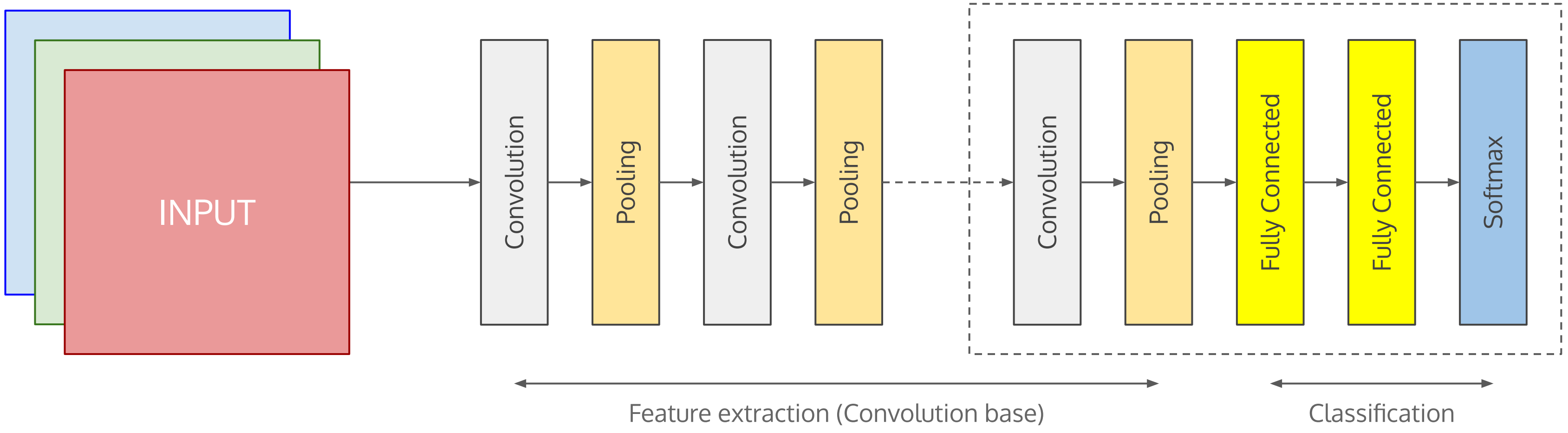
- A machine learning technique where a model trained on one task is re-purposed on a second related task → use pre trained models
- Why :
  - Small / insufficient training dataset → avoid overfitting
  - Cut-off training time
- There are many open source models trained on a very large dataset : ImageNet, COCO, KITTI, etc.
- We can download some open source implementation of the neural network, not only the code, but also the model and weights that have already trained.

Train The Entire Model



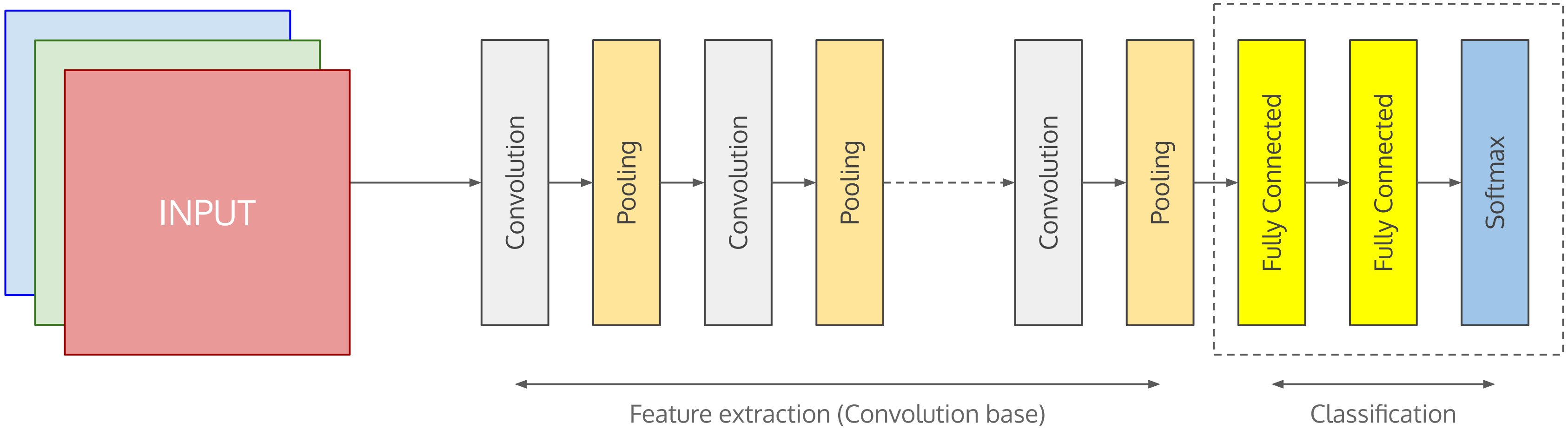
- Large dataset
- Customize softmax as needed
- Re-train all network by using trained weight as initial value

Train Partial Layer



- Large dataset
- Customize softmax as needed
- Retraining some convolution layar and classification layer
- Freeze other layer

Train Top Layer



- Small dataset
- Customize softmax as needed
- Freeze all feature layer
- Retrain classification

LAB 02

Transfer Learning

Lab Description

What you will learn:

1. Using VGG19 model
2. Retrain VGG19 model using Cifar10 dataset

Requirement :

1. Anaconda
2. Python 3
    - numpy
    - pandas
    - Matplotlib
    - scipy
    - jupyter
    - tensorflow
    - keras

STEP 01 - Execute Notebook

- Open notebook : Transfer Learning - Cifar10.ipynb in Jupyter notebook
- Follow the instructor to run notebook
- Instructor will explain step by step process

Pretrained Model - Computer Vision

- Object Detection:

  Mask R-CNN, YOLOv2, MobileNet, Ripe/unripe tomato classification, Car classification

- Facial Recognition and Regeneration

  VGG-Face Model, 3D Face Reconstruction from a Single Image

- Segmentation

  Deeplabv3, Robot Surgery Segmentation

- Image Captioning

  github.com/boluoyu/ImageCaption

Labs247 Technology in motion

Pretrained Model - NLP

- Multi-Purpose NLP Models

  ULMFiT, Transformer, Google's BERT, Transformer-XL, OpenAI's GPT-2

- Word Embeddings

  ELMo, Flair

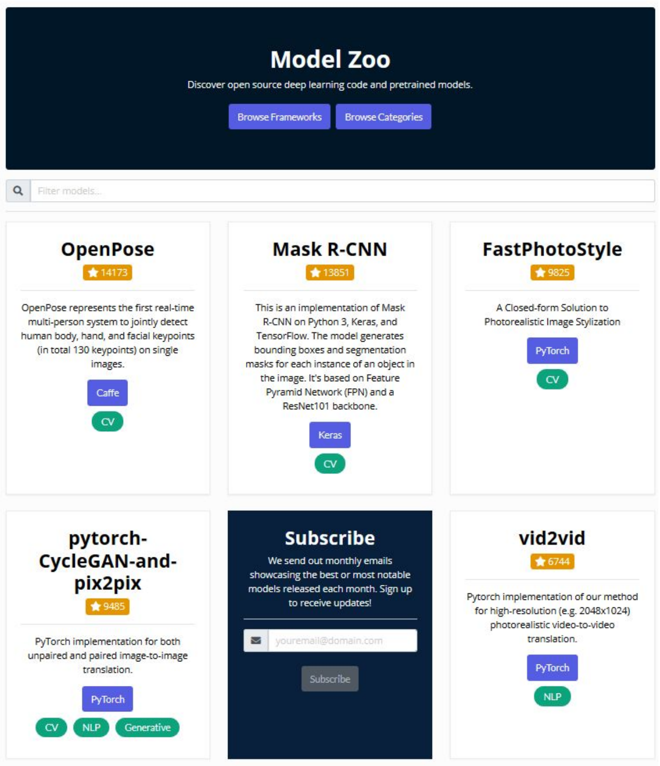

THE VERGE

OpenAI has published the text-generating AI it said was too dangerous to share

The lab says it's seen 'no strong evidence of misuse so far'

By James Vincent | Nov 7, 2019, 7:24am EST

ModelZoo.co



Model Zoo curates and provides a platform for deep learning researchers to easily find pre-trained models for a variety of platforms and uses. We regularly update the site, and provide filtering functionality for users to find models that they need, either for educational purposes, transfer learning, or other uses.


The True
WINNER
Local Appliance
Local Content, Local Support
+
Local Consulting
Local Certification, Local Engineer
=
National Benefit
Prosperity