Capstone Project on:

Image Classification Using Neural Networks

Sileshi Hirpa

Data Science Immersive (DSIR-111) at General Assembly

Feb 07/2022

Content

- Problem statement
- Introduction
- Data preprocessing
- Building Artificial and Convolutional Neural Network
- Regularizations parameters
- Model Selection and Prediction
- Conclusions/Next Step

Problem Statement

As human use their eyes to perceive and react to their surrounding, computers use computer vision (CV) algorithms to detect images in order to do the tasks they are instructed to do.

We are using CV in our daily lives with all our smart devices like security cameras, smart phones, smart door locks, and so on that uses face recognition to authenticate users. The applications of CV algorithms are also expanding in autonomous driving, detecting COVID-19, Breast and Skin Cancer detections and so on.

This project builds a model that accurately classifies images based on unique features of training dataset.

Goal: identifying unique or distinguishing features of images.

Measurement metrics: accuracy

1. Introduction

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

Table: Dataset class labels

According to IBM, Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.

- CV helps physical world interact with the digital world

The basic algorithms that CV apply include:

- Input data:
- Preprocessing
- Feature extraction
- Modeling

Dataset

This capstone project uses a CIFAR-10 (Canadian Institute for Advanced Research) dataset which is an established computer-vision dataset used for object recognition.

It consists of 60,000 32x32 color images containing one of 10 object classes, with 6,000 images per class. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The dataset is reset as:

- Training: 30,000 images
- Validation: 20,000 images
- testing: 10,000 images

2. Data preprocessing

Sample picture from my Jupyter Notebook

- ➔ Converting to floats,
- ➔ Normalization,
- ➔ Resizing,
- ➔ Color transformation (depends on what you're looking for), ⚠️
  - ◆ care should be given to the type of image we're working on since traffic image detection, and skin cancer detection outputs are affected by image color
- ➔ One Hot encoding,
- ➔ and so on.

Not that much EDA for this image data 🙂

3. How do computers “See” images?

- Images are just a 2D arrays of numbers of pixels (for grayscale pictures)
- A pixel is the smallest item of information in an image
  - the more pixels we use, the more closer a digital image to the original image will be.

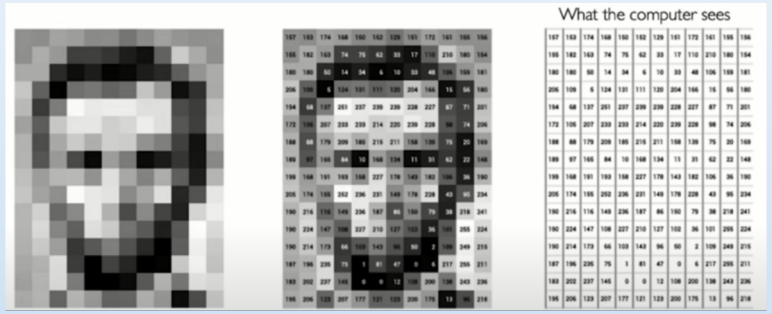


Fig. Source Computer vision

Vs

What human see

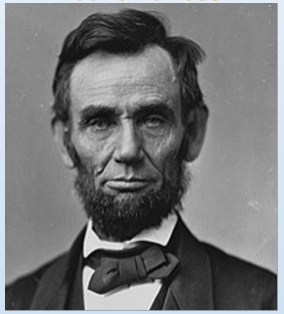
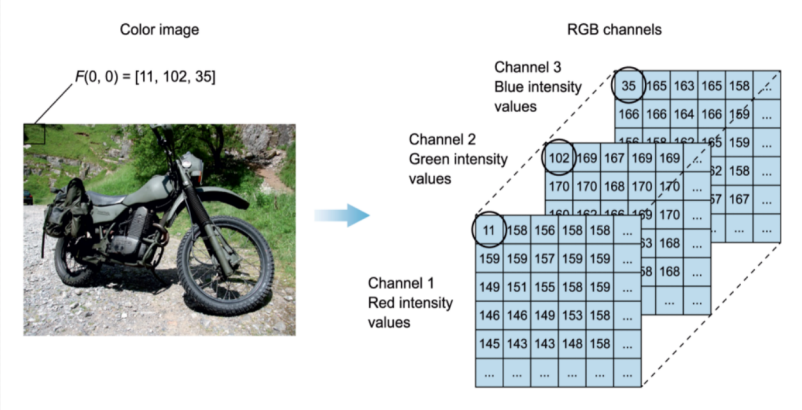
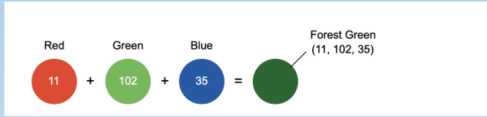


Fig. source Wikipedia

- Standard images (RGB) are also be viewed by computers as 3D array of numbers.





RGB color combination

Source for these two figs (Mohammed 22)

4. Building Artificial and Convolutional NNs

4.1. Artificial Neural Nets(ANN)

- ANN is imitation of how information is processed in human brain.
- A stack of thousands/millions of single neurons (perceptrons) form ANNs.

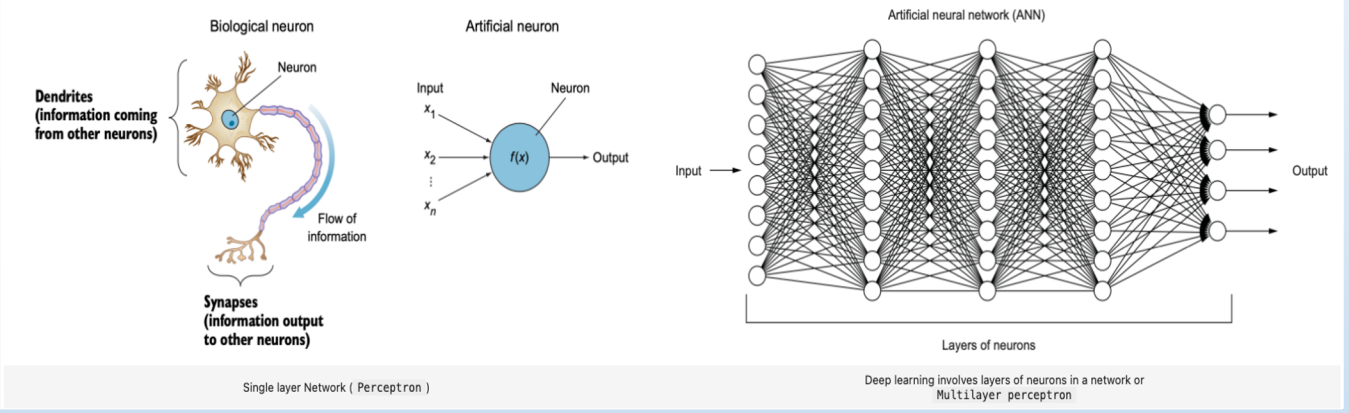


Fig source (Mohammed 9)

4.1.1. Perceptron

- When we zoom in the ANN of the figure, we'll see a single neuron known as perceptron.
- We need to understand how the perceptron works to better figure out how the whole ANN works.
  - ➔ A Perceptron is an Artificial Neuron
  - ➔ It is the simplest possible NNs
  - ➔ Frank Rosenblatt (1928-1971) was an American psychologist who discovered perceptron (artificial neuron) based on his study on human brain cells (neurons) in 1957.
  - ➔ His algorithm was:
    - ◆ set a threshold value
    - ◆ multiply all the inputs
    - ◆ sum all the results
    - ◆ activate the output (source).

Perceptron …cont'd

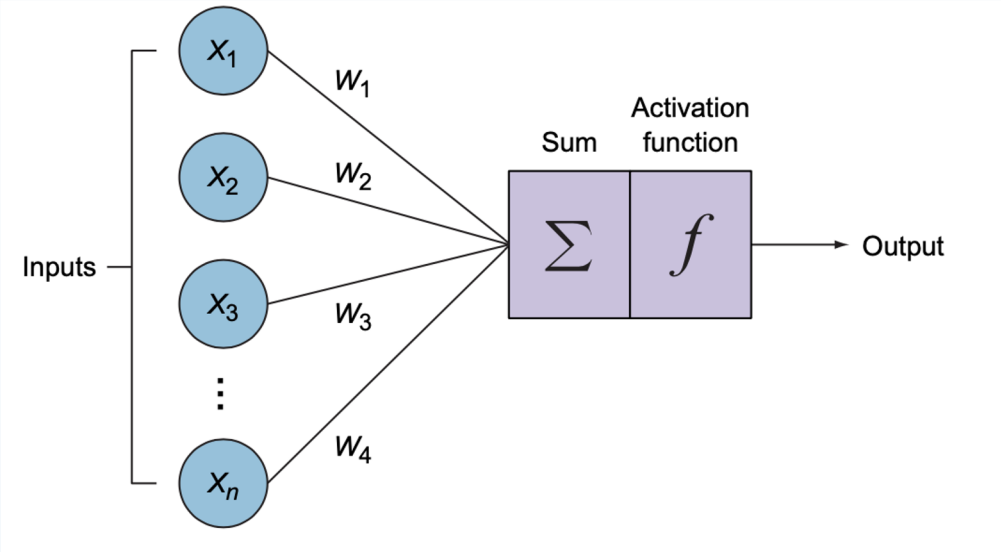


Fig. source (Mohammed 40)

As we can see, a perceptron consists of 4 parts:

1. Input values or One input layer
2. Weights and Bias
3. Net sum
4. Activation Function

A.B= $A^T B = \sum_{i=1}^{n} a_i b_i$

The output$(\hat{y}) = \sigma(b + \sum x_i * w_i)$; $where$

x_i and w_i are inputs and their corresponding weights.

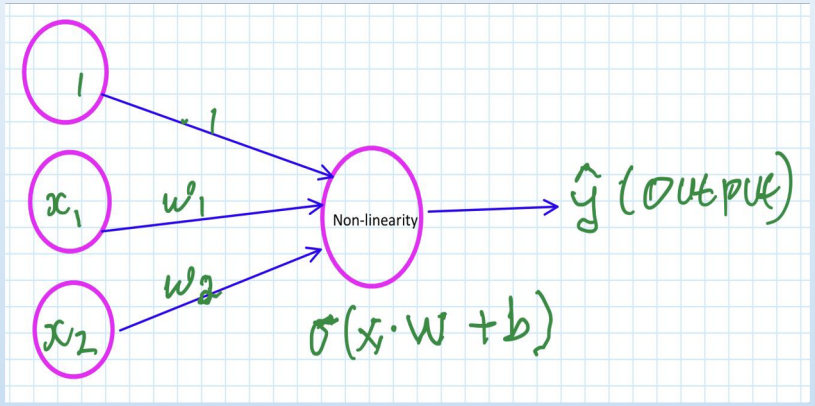
The activation function, σ, we'll be using is the sigmoid function:

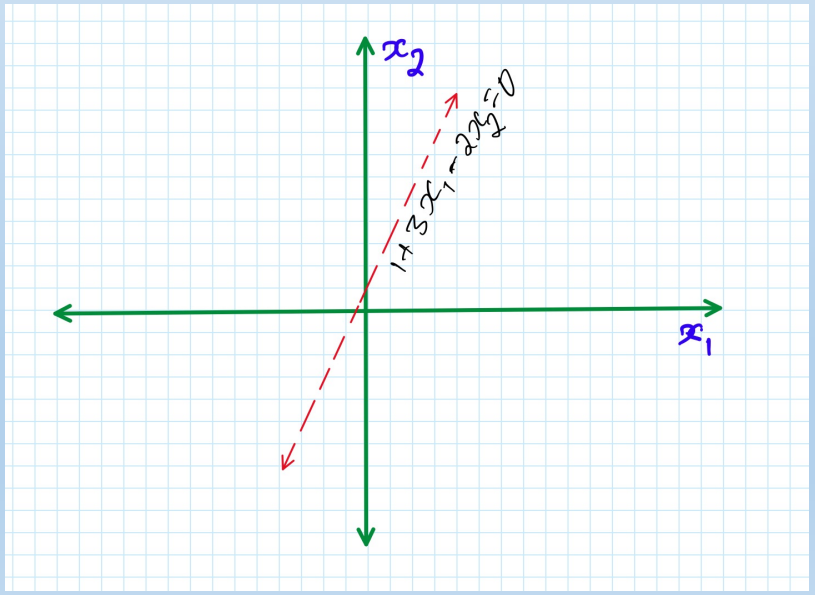
$$\hat{y} = \sigma(z)$$

$$= \frac{1}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-(b+\sum x_i * w_i)}}$$

Perceptron …cont'd





Figs. by author

Let's see an example with only two inputs and weights as shown:

$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, and $W = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$ and let $b = 1$

Now,

$$\hat{y} = \boldsymbol{\sigma}(b + X^T W)$$

$$= \sigma\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$= \sigma \underbrace{(1 + 3x_1 - 2x_2)}_{\text{Equation of a hyperplane in 2D}}$$

Perceptron …cont'd

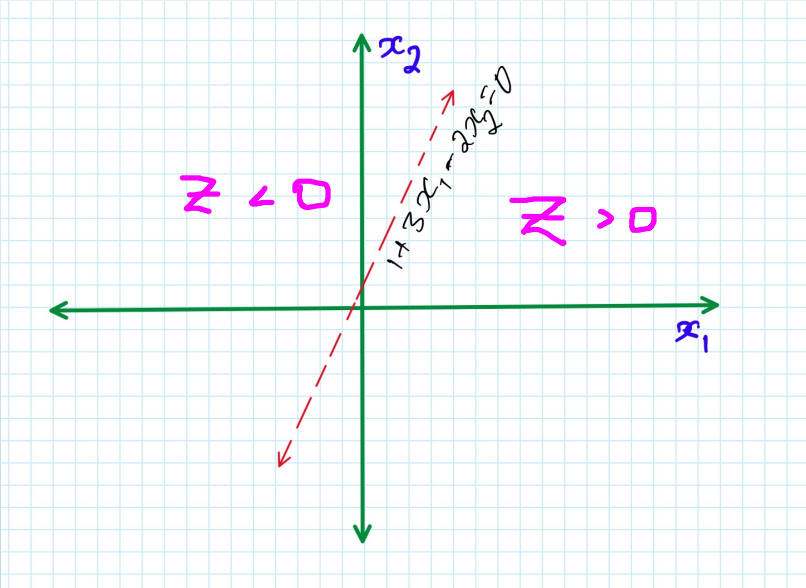


Fig. by author

Now, if we've inputs:

$X = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$, and $W = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$\hat{y} = \sigma(z)$
$= \sigma(1+3x_1 - 2x_2)$
$= \sigma(1+3*-1 -2*2)$
$= \sigma(-6)$

- $X = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ is classified as a point on the left side of the linear classifier from the left graph.

$\hat{y} = \sigma(z)$

$= \frac{1}{1+e^{-z}}$

$= \frac{1}{1+e^{6}}$

≈ 0.0025

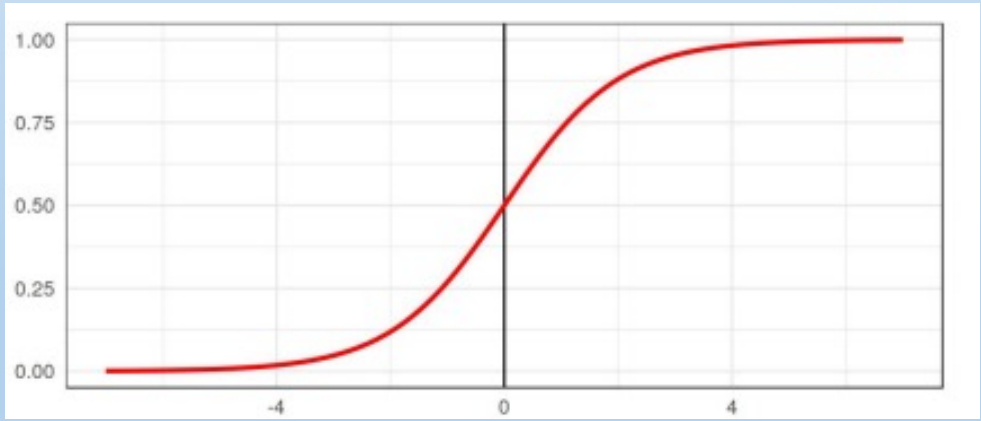


Fig. from DSIR-111 slide note

Perceptron …cont'd

- Since artificial neurons are designed to imitate how human brain works, we can stack perceptrons to form a multilayer perceptron (MLP) and hence we build the ANNs.

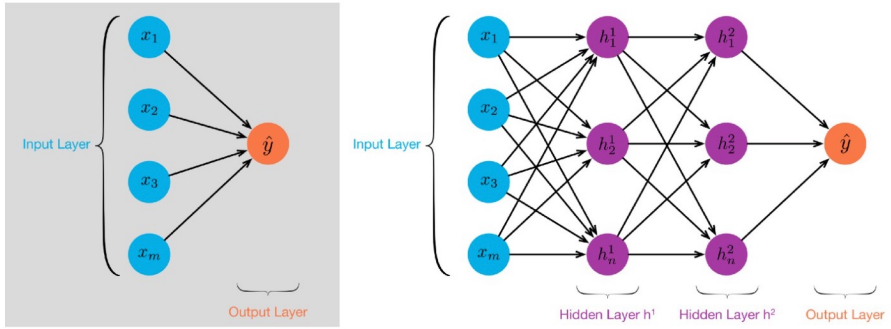
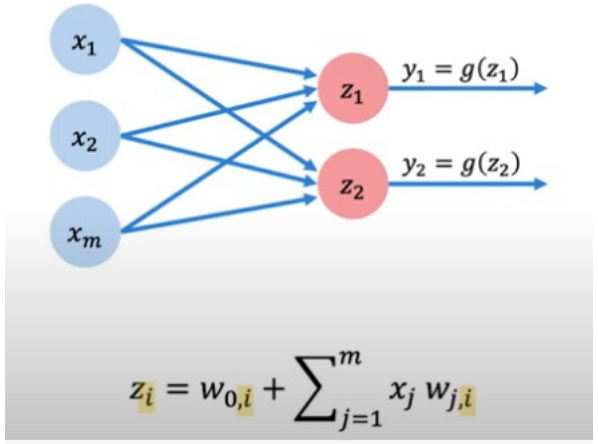
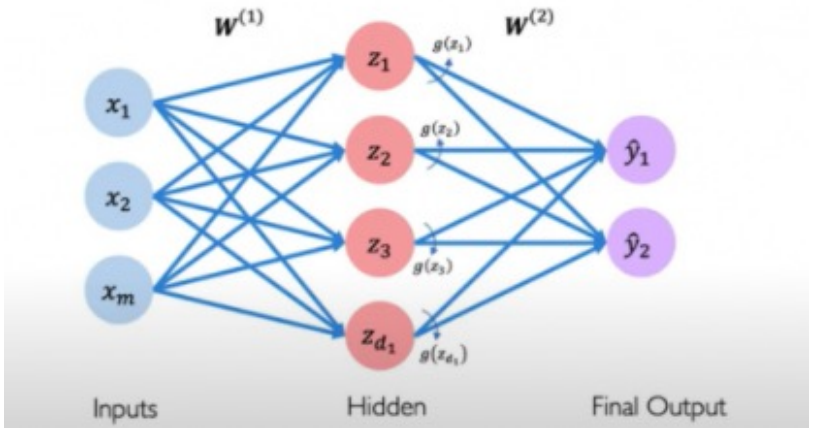


Fig. source

- Now if we want to define a multi-output NN, we can simply add another perceptron to this above picture so instead of having one perceptron now we have two perceptrons and so on.





Figs. source

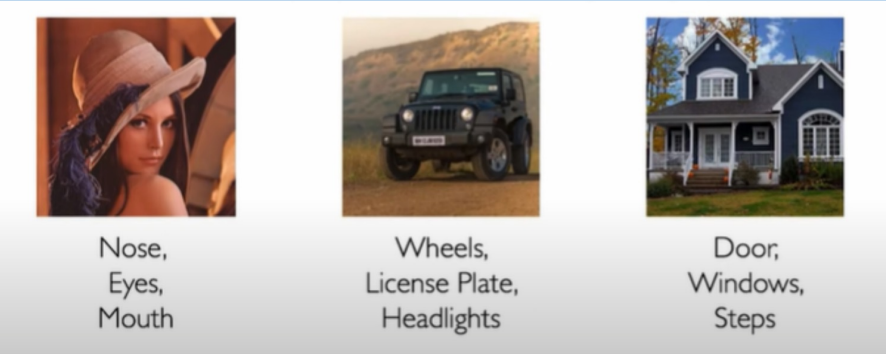
4.2 Convolutional Neural Networks (CNN)

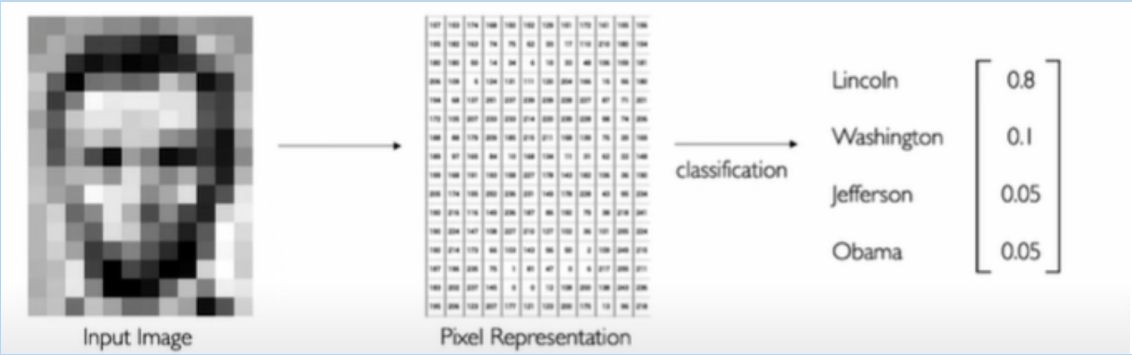
Although we were able to build ANNs by stacking millions of perceptrons, manually doing so is ineffective in extract image features. This is when convolutions come into play.

According to Wikipedia, mathematical convolution is defined as:

$$(f * g)(t) := \int_0^t f(\tau)g(t-\tau)d\tau \quad for \quad f, g: [0, \infty) \to \mathbb{R}.$$

- Convolution helps us extract features
- Features make pictures unique.





Figs. source

Convolution …cont'd

Convolution operation has:

- Input images
- filter/kernel
- feature map

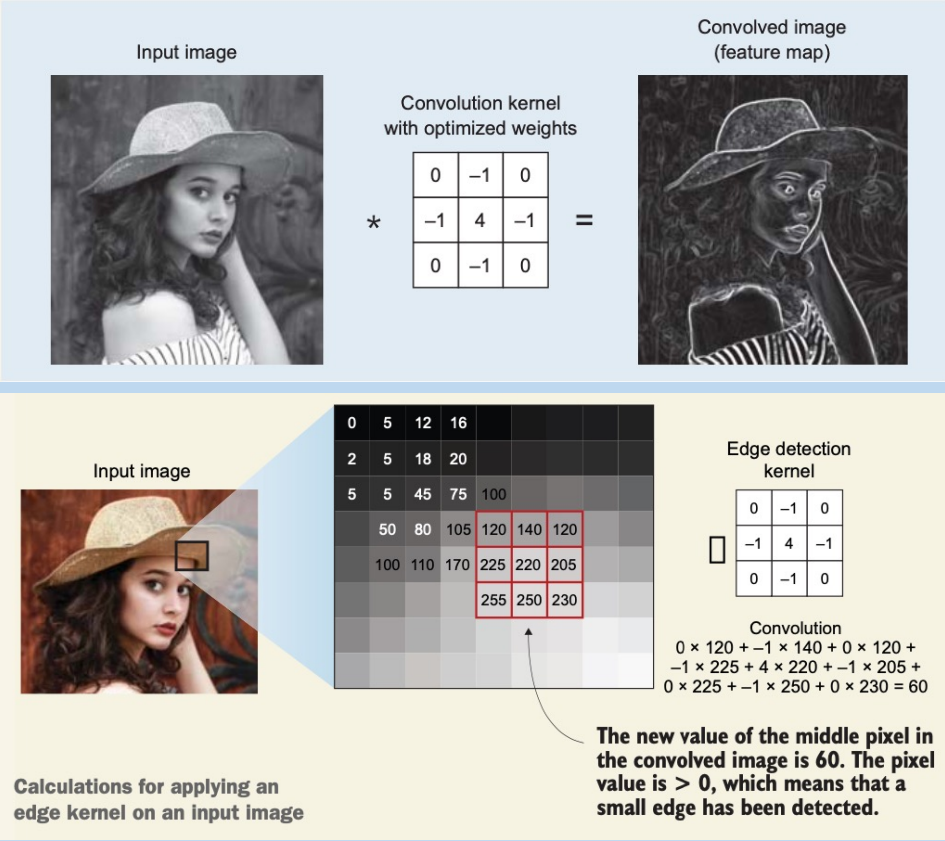


Fig. source (Mohammed 109)

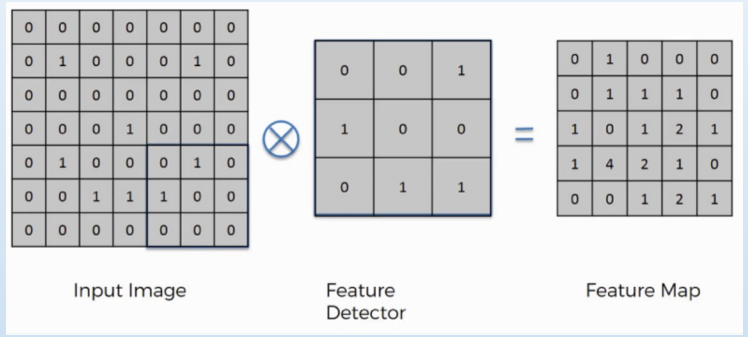


Fig. source

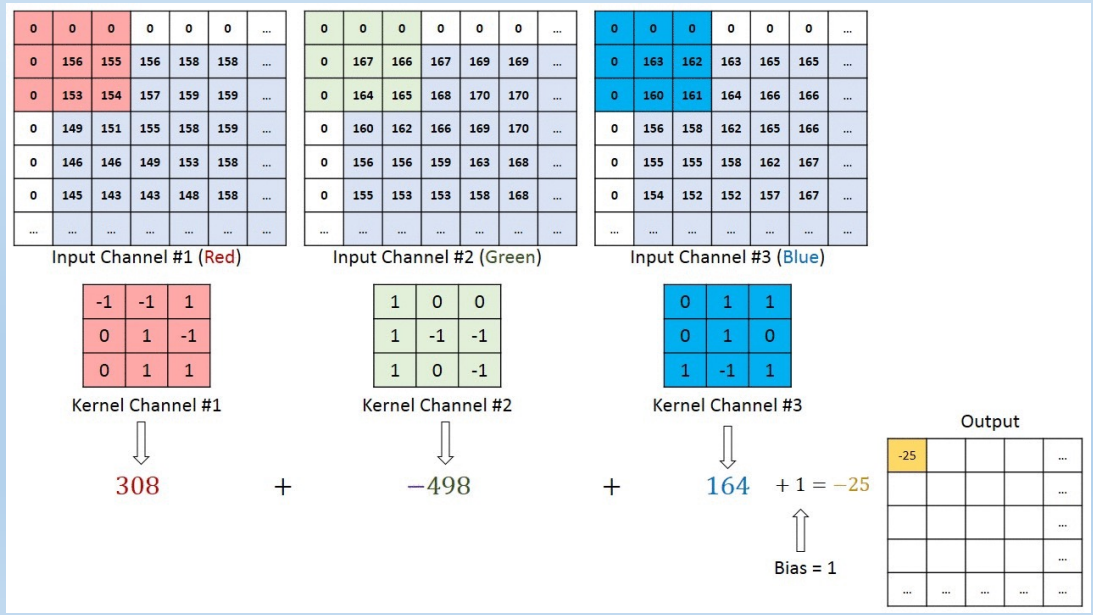
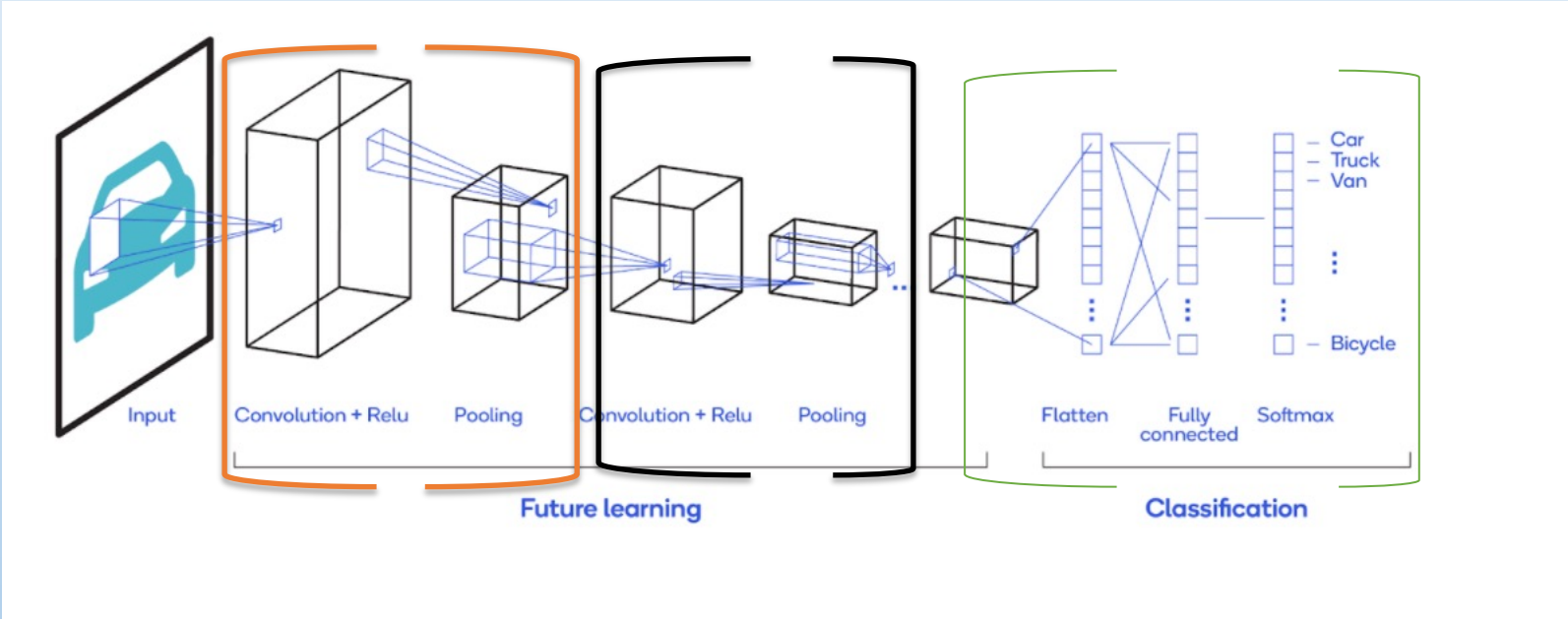


Fig. source

Convolution …cont'd

It's through try-and-error during the training phase that NNs determine important feature detectors and use them to develop feature maps, which we call convolutional layers.

So, putting it all together:



Applying:

Conv + ReLU + Pooling + Conv + ReLU + Pooling + …+ FC

operations as many times as we like, we will get the desired model to make our final prediction on unseen data.

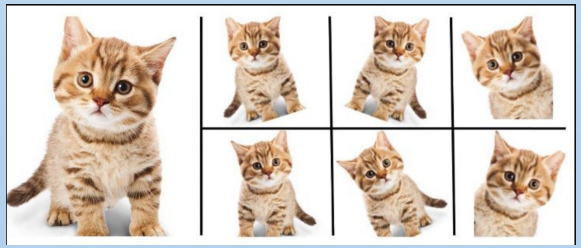
5. Regularizations

Even with so much dense layers, we are prone to overfitting:

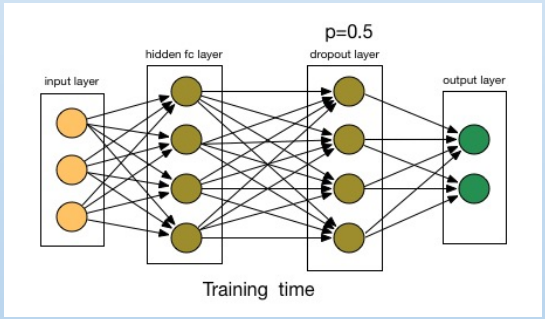
Train_accur: 91.29%

test_accur: 67.68%

Some of the the regularizations used include:

- Dropout
- Early stopping
- Batch normalization
- L_1 and L_2 regularization
- 

Augmentation fig. source



Dropout fig. source: DSIR-111 slide note

6. Model Selection and Prediction

Trying out all the necessary regularizations that I'd think of, my best model has:

Train_accur: 86.04%

Test_accur: 86.96%

- Out of the 10,000 testing datasets 1,034 of them were misclassified.

7. Conclusions/Recommendations

- Understanding how artificial neurons work is fundamental to know the whole NN algorithms
- feature extraction is at the heart of NNs
- Neural networks are prone to overfitting since they try to learn too much or too many details in the training data along with the noise with too many parameters.
- Regularizations of parameters help minimize the overfitting
- NNs may make dangerous mistakes unless trained properly.

8. Next Step

- **GUI for image prediction(or Streamlit)**
- **Video Classification.**

Videos are a collection of images frames.

I will also use CNNs to extract features from frames of videos.

Then build a model that will be able to classify videos into their respective categories.

Steps:

- ✓ exploring the dataset:
  - using training dataset to train the model
  - using validation dataset to evaluate the trained model
- ✓ extracting frames from the training and validation datasets
- ✓ preprocessing these frames and model

References:

1. Action Recognition and Video Classification using Keras and Tensorflow on Medium.
2. Introduction to Video Classification on Medium.
3. Step-by-Step Deep Learning Tutorial to Build your own Video Classification Model on Analytics Vidhya.

Questions and Discussion

Thank You

Good Luck everyone!