Image Processing using Keras API

Daniyal Kautsar

Google Developer Student Club UGM
February 7, 2023

All of these materials are available at:

https://github.com/mdaniyalk/image-processing-gdsc-ugm or bit.ly/gdsc-ugm-impro

Prerequisite:

Little bit knowledge of python, math, & linear algebra

Main Topics

- Deep Learning & Neural Network
- Convolutional Neural Network and Computer Vision
- Hands-on and Practice Lab: Image Classification
- Exploration: Advanced Image Processing & Computer Vision
- Q&A Session

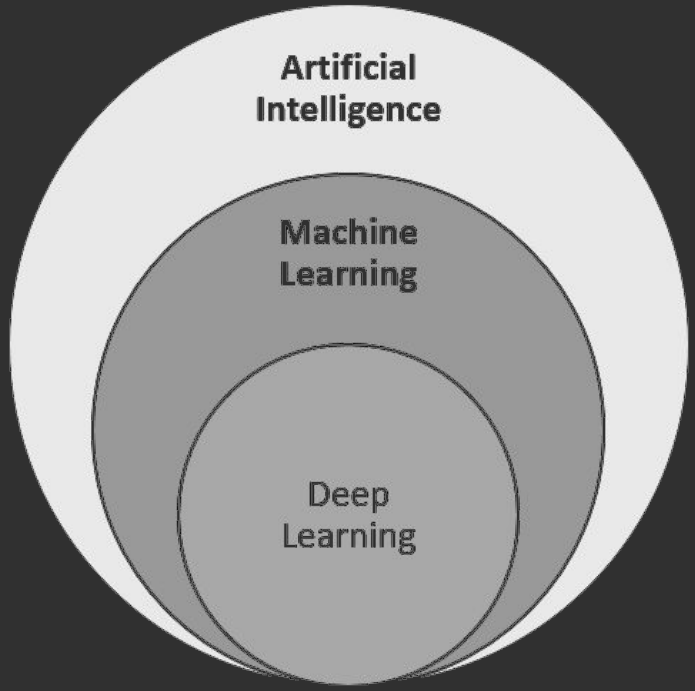
Deep Learning

Deep Learning

Artificial Intelligence, any technique that enable computer to mimics human behavior.

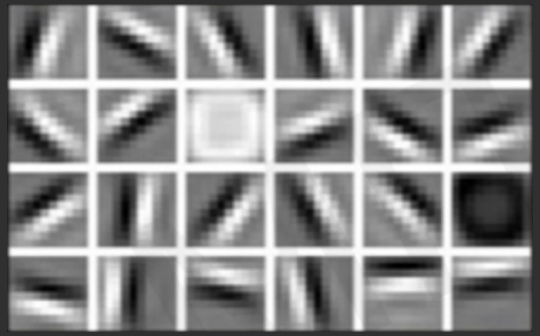
Machine Learning, ability to learn without explicitly being programmed.

Deep Learning, extract pattern from data using neural networks



Why deep learning?

Low level features



Lines & Edges

Mid level features

Eyes, Nose, & Ears

High level features

Facial Structures

Can we extract and hard coded a program to understand those features?

Introduction to Tensorflow Keras API

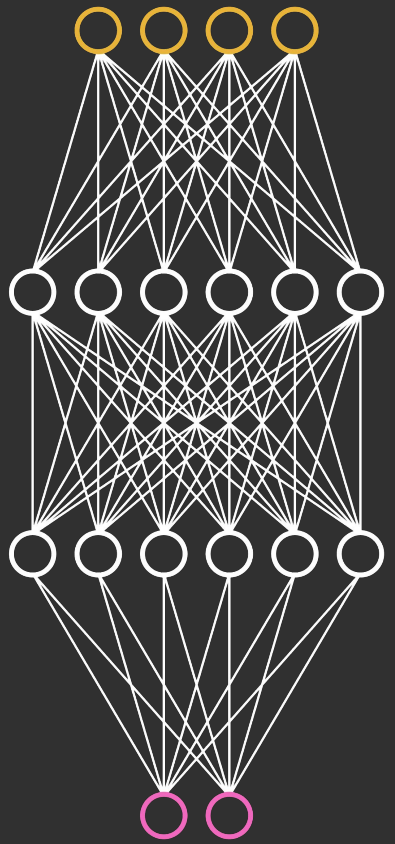
- Tensorflow is an open-source ML framework that develop by Google.
- It can be used in almost any devices, including microcontroller.
- Available in C++, **Python**, Java, Javascript, etc.
- Tensorflow offer two Model API. Functional and **Sequential API**.
- More on tensorflow at tensorflow.org

Neural Network

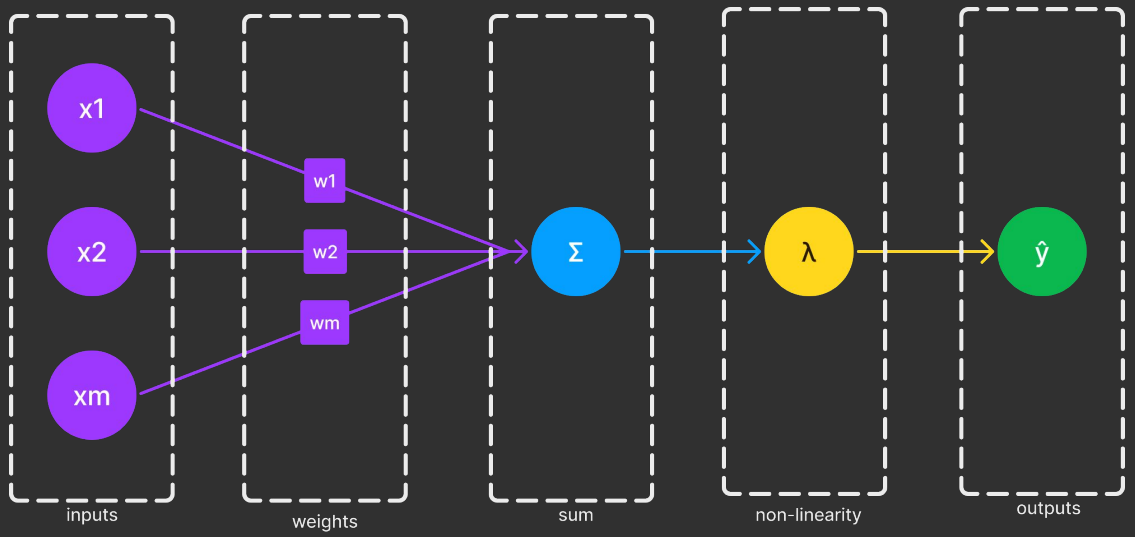
- Type of algorithm that inspired by human brain
- Consist of interconnected neurons
- Think of each neuron is a linear regression itself wrapped with non-linear function

```
outputs = activation(dot(input, weight) + bias)
```

- Use forward & back propagation to update it’s weight and minimize loss on every iteration (epochs)



Neural Networks: Neuron



Output

Linear combination of inputs

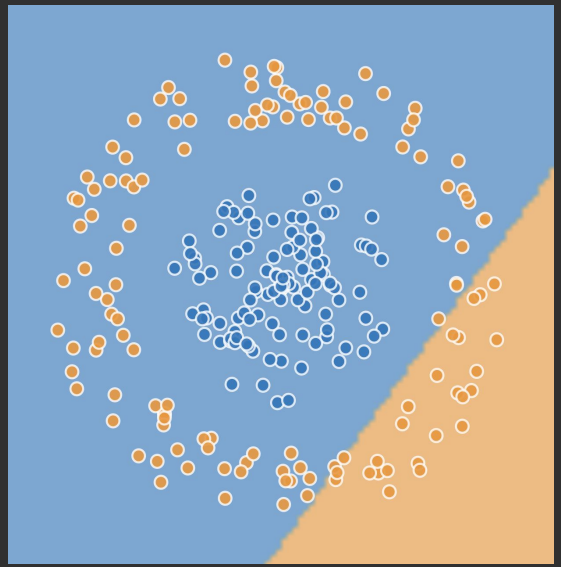
$$\hat{y} = g\left(\sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

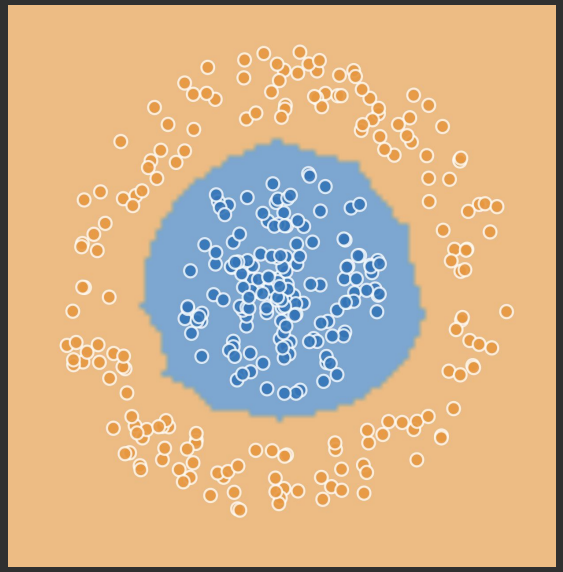
$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

Neural Networks: Activation Function



Linear activation can't approximate complex functions



Non-linearities allow us to approximate complex functions

We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

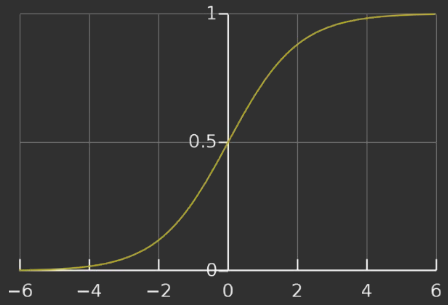
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

This is just a line in 2D!

Neural Networks: Common Activation Function

Sigmoid Function

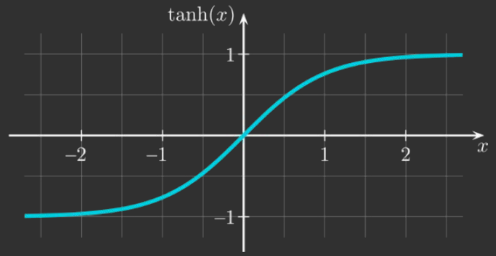


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.keras.activations.sigmoid(out)`

Hyperbolic Tangent

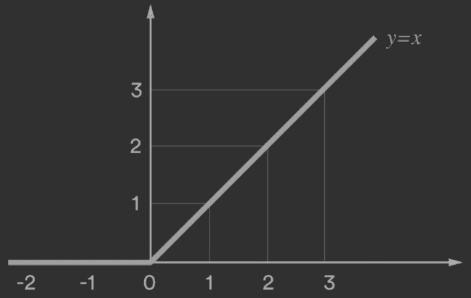


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.keras.activations.tanh(out)`

Rectified Linear Unit (ReLU)



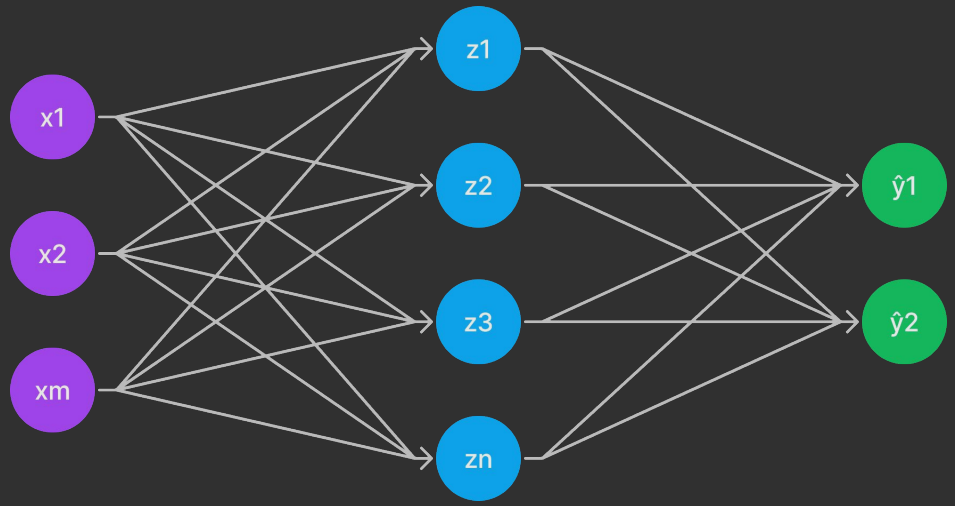
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

`tf.keras.activations.relu(out)`

NOTE: Non-linear activation functions

Neural Networks: Single layer NN

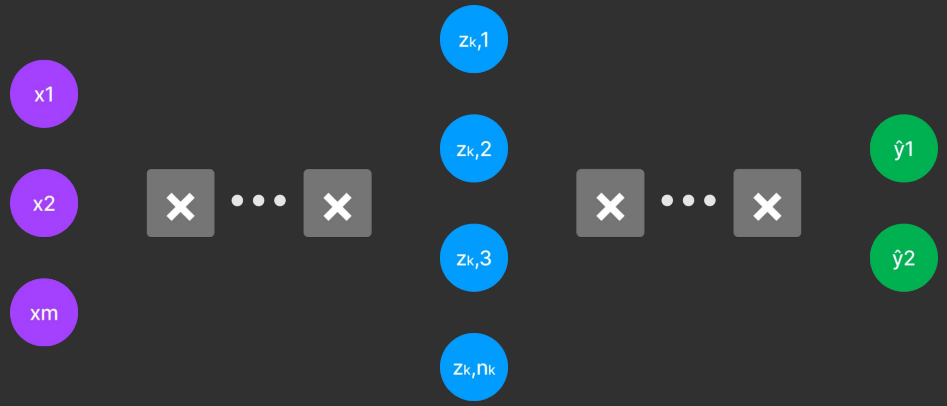


$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

```
import tensorflow as tf

model = tf.keras.Sequential([
    tf.keras.layers.Dense(n),
    tf.keras.layers.Dense(2)
])
```

Neural Networks: Deep NN



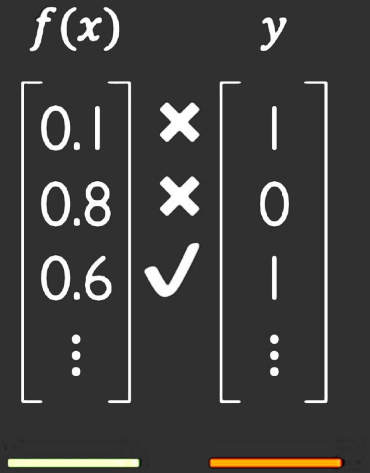
$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

```
import tensorflow as tf

model = tf.keras.Sequential([
    tf.keras.layers.Dense(n1),
    tf.keras.layers.Dense(n2),
    ...,
    tf.keras.layers.Dense(2)
])
```

Neural Networks: Loss Function

Loss measure the cost of incurred from incorrect predictions



$$\mathcal{L}\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\right)$$

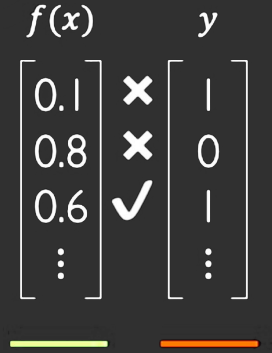
Neural Networks: Cross Entropy Loss

Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events.

Can be used for models that outputs a probability between 0 and 1

Binary for two classes

Multi-class for more than two classes



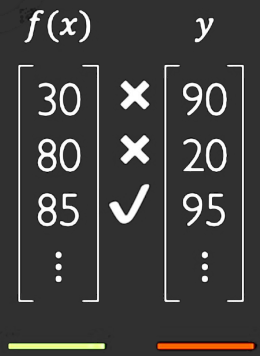
$$J(W) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log \underbrace{\left(f(x^{(i)}; W)\right)}_{\text{Predicted}} + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log \left(1 - \underbrace{f(x^{(i)}; W)}_{\text{Predicted}}\right)$$

```
loss = tf.keras.losses.BinaryCrossentropy()
```

Neural Networks: Mean Squared Error (MSE) Loss

MSE is a measure of average squared distances of predicted and actual value.

Can be used with regression models that output continuous value.



$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}; W\right)\right)^2$$

Actual Predicted

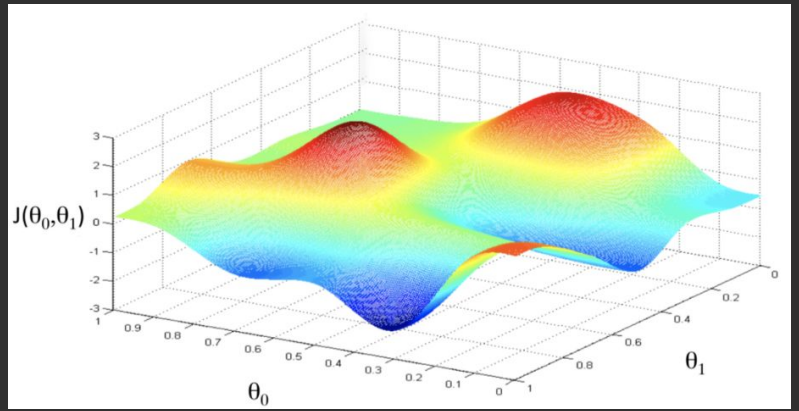
```
loss = tf.keras.losses.MeanSquaredError()
```

Neural Networks: Loss Optimization

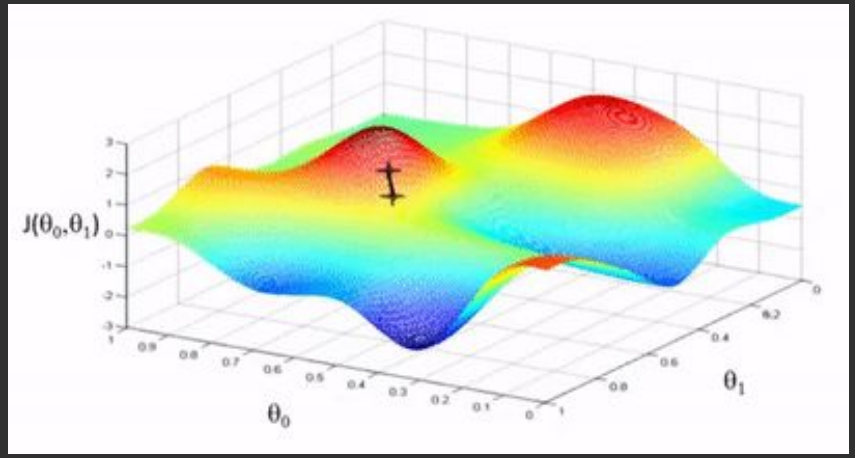
We want to find the network weights that achieve the lowest loss

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} J(\boldsymbol{W})$$



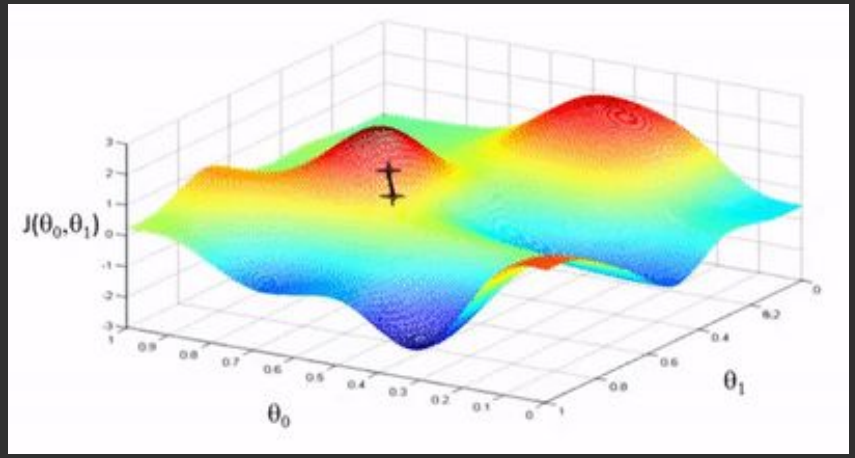
Neural Networks: Gradient Descent



Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Return weights

Neural Networks: Gradient Descent



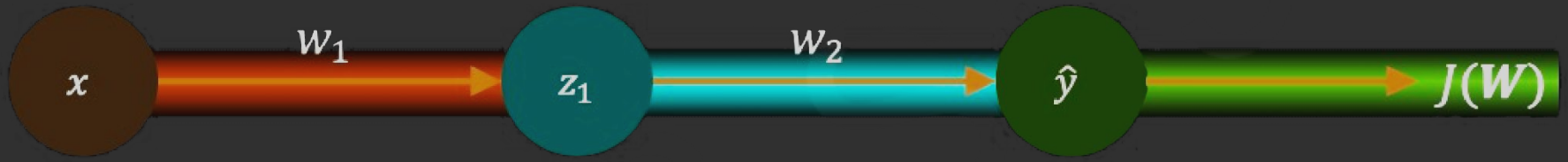

```
import tensorflow as tf

weight = tf.Variable([tf.random.normal()])

while True:
    with tf.GradientTape() as grad:
        loss = compute_loss(weight)
        gradient = grad.gradient(loss,
weight)

    weight = weight - lr *  gradient
```

Neural Networks: Backpropagation



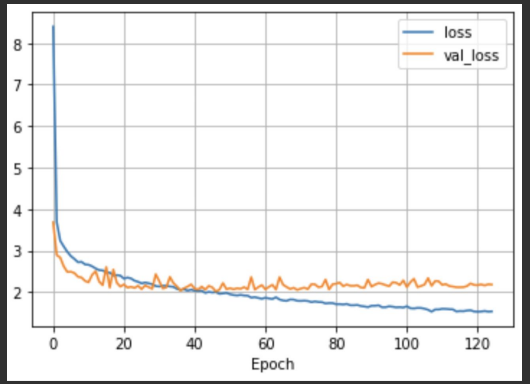
$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

Repeat for every weight in the network using gradient from later layers

Neural Networks: Gradient Descent Algorithms

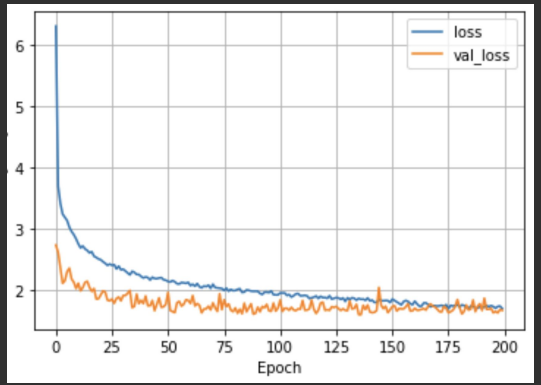
- SGD (Stochastic Gradient Descent) `tf.keras.optimizer.SGD`
- Adam `tf.keras.optimizer.Adam`
- Adadelta `tf.keras.optimizer.Adadelta`
- Adagrad `tf.keras.optimizer.Adagrad`
- RMSProp `tf.keras.optimizer.RMSProp`

Neural Networks: Learning Curve & Overfitting Problem

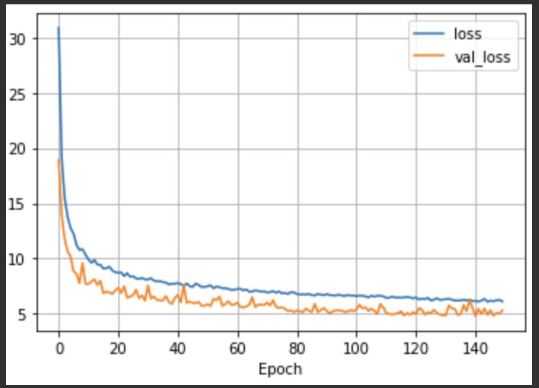


Overfitted model

Our model can’t generalize well



Fitted model



Underfitted model

Our model can’t fully understand the data

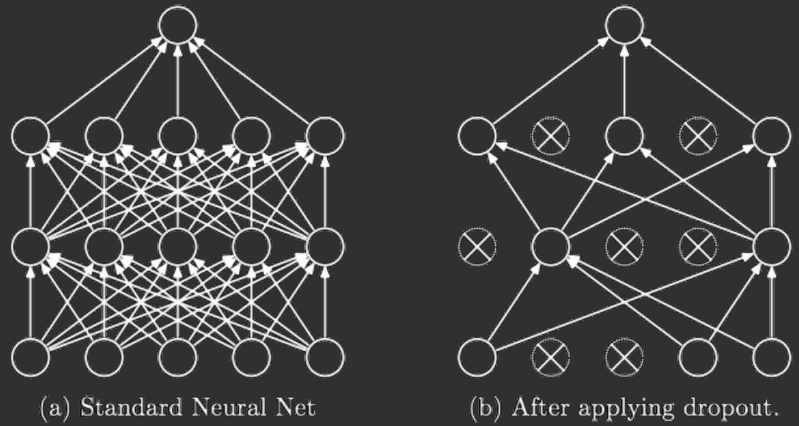
Neural Networks: Regularization

Technique to prevent overfitting

1. Droupout
2. Early stopping
3. BatchNormalization

Dropout

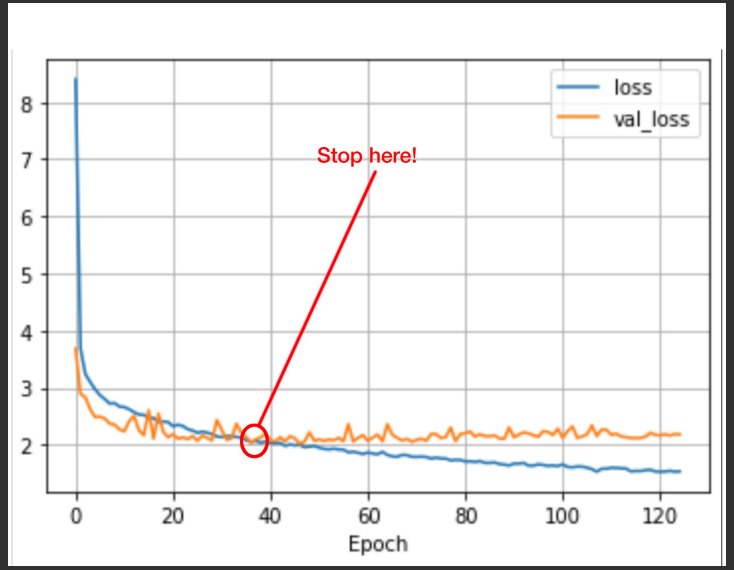
Dropout refers to dropping out the nodes (input and hidden layer) in a neural network with probability of *p.*



(a) Standard Neural Net

(b) After applying dropout.

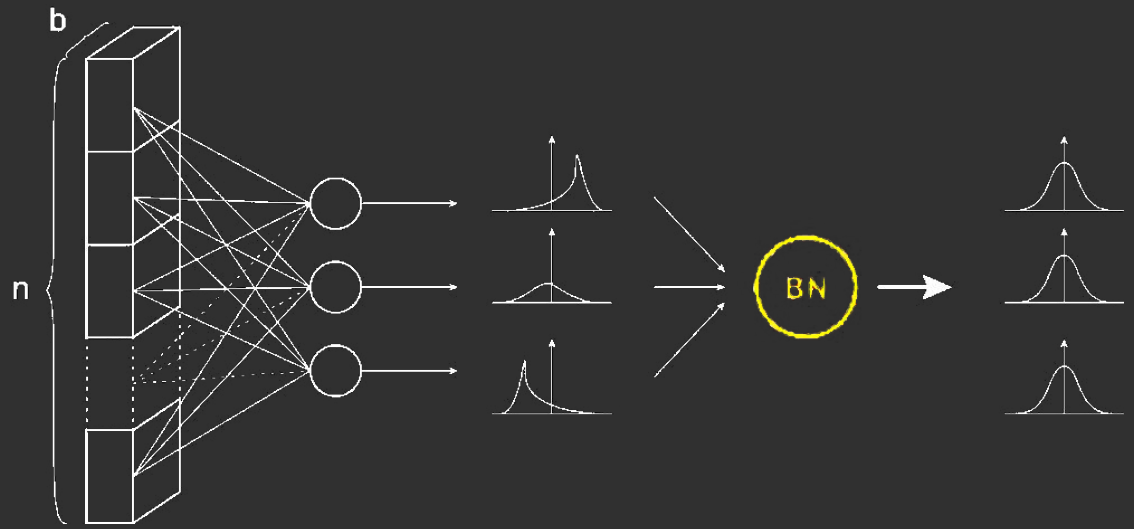
Early Stopping



Stop the training before the model overfit

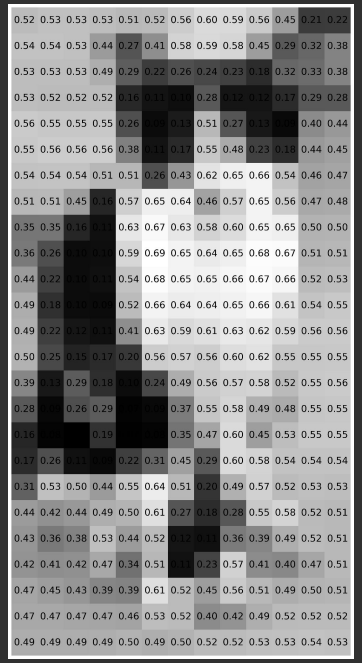
BatchNormalization

Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.



Convolutional Neural Network and Computer Vision

How computers see images?

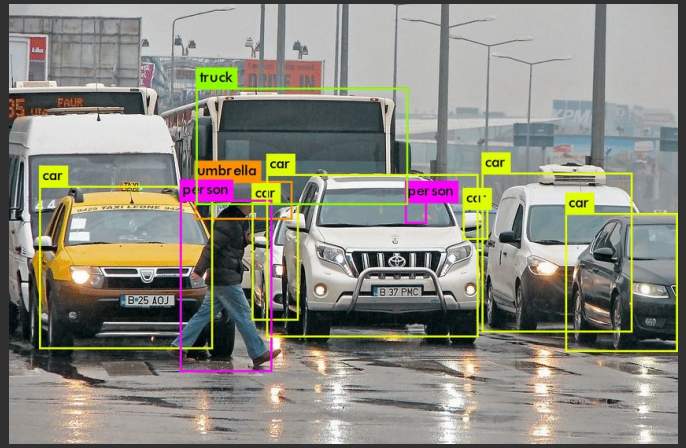


```
[[0.52 0.53 0.53 0.53 0.51 0.52 0.56 0.6  0.59 0.56 0.45 0.21 0.22]
 [0.54 0.54 0.53 0.44 0.27 0.41 0.58 0.59 0.58 0.45 0.29 0.32 0.38]
 [0.53 0.53 0.53 0.49 0.29 0.22 0.26 0.24 0.23 0.18 0.32 0.33 0.38]
 [0.53 0.52 0.52 0.52 0.16 0.11 0.1  0.28 0.12 0.12 0.17 0.29 0.28]
 [0.56 0.55 0.55 0.55 0.26 0.09 0.13 0.51 0.27 0.13 0.09 0.4  0.44]
 [0.55 0.56 0.56 0.56 0.38 0.11 0.17 0.55 0.48 0.23 0.18 0.44 0.45]
 [0.54 0.54 0.54 0.51 0.51 0.26 0.43 0.62 0.65 0.66 0.54 0.46 0.47]
 [0.51 0.51 0.45 0.16 0.57 0.65 0.64 0.46 0.57 0.65 0.56 0.47 0.48]
 [0.35 0.35 0.16 0.11 0.63 0.67 0.63 0.58 0.6  0.65 0.65 0.5  0.5 ]
 [0.36 0.26 0.1  0.1  0.59 0.69 0.65 0.64 0.65 0.68 0.67 0.51 0.51]
 [0.44 0.22 0.1  0.11 0.54 0.68 0.65 0.65 0.66 0.67 0.66 0.52 0.53]
 [0.49 0.18 0.1  0.09 0.52 0.66 0.64 0.64 0.65 0.66 0.61 0.54 0.55]
 [0.49 0.22 0.12 0.11 0.41 0.63 0.59 0.61 0.63 0.62 0.59 0.56 0.56]
 [0.5  0.25 0.15 0.17 0.2  0.56 0.57 0.56 0.6  0.62 0.55 0.55 0.55]
 [0.39 0.13 0.29 0.18 0.1  0.24 0.49 0.56 0.57 0.58 0.52 0.55 0.56]
 [0.28 0.09 0.26 0.29 0.07 0.09 0.37 0.55 0.58 0.49 0.48 0.55 0.55]
 [0.16 0.08 0.07 0.19 0.07 0.08 0.35 0.47 0.6  0.45 0.53 0.55 0.55]
 [0.17 0.26 0.11 0.09 0.22 0.31 0.45 0.29 0.6  0.58 0.54 0.54 0.54]
 [0.31 0.53 0.5  0.44 0.55 0.64 0.51 0.2  0.49 0.57 0.52 0.53 0.53]
 [0.44 0.42 0.44 0.49 0.5  0.61 0.27 0.18 0.28 0.55 0.58 0.52 0.51]
 [0.43 0.36 0.38 0.53 0.44 0.52 0.12 0.11 0.36 0.39 0.49 0.52 0.51]
 [0.42 0.41 0.42 0.47 0.34 0.51 0.11 0.23 0.57 0.41 0.4  0.47 0.51]
 [0.47 0.45 0.43 0.39 0.39 0.61 0.52 0.45 0.56 0.51 0.49 0.5  0.51]
 [0.47 0.47 0.47 0.47 0.46 0.53 0.52 0.4  0.42 0.49 0.52 0.52 0.52]
 [0.49 0.49 0.49 0.49 0.5  0.49 0.5  0.52 0.52 0.53 0.53 0.54 0.53]]
```

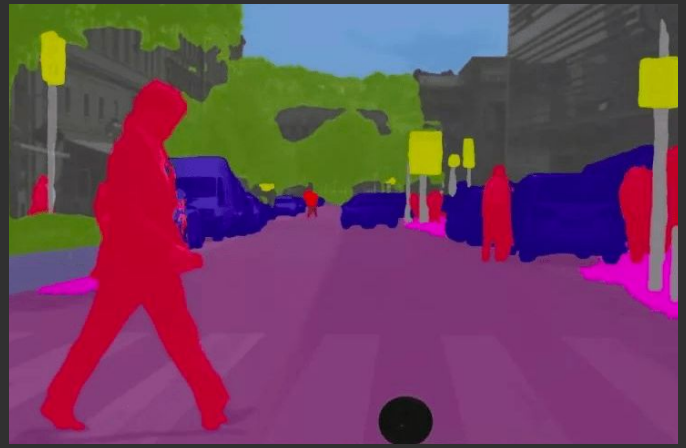
Image is just array of brightness value from each color channel!

Example Use Case of Computer Vision

Regression: output variable takes continuous value. E.g. bounding box.



Object Detection



Object Segmentation

Example Use Case of Computer Vision

Classification: output variable takes class label.

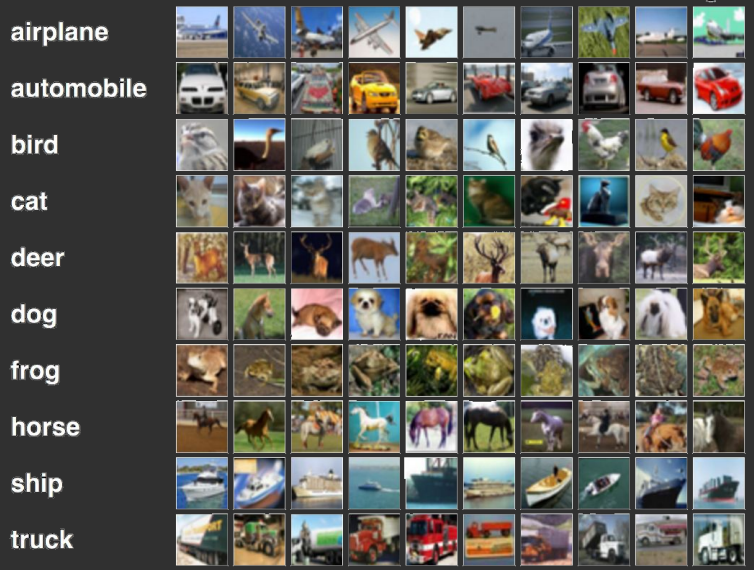
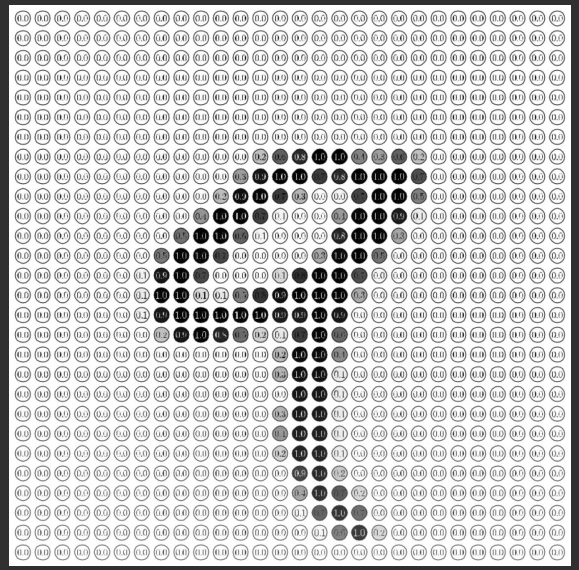
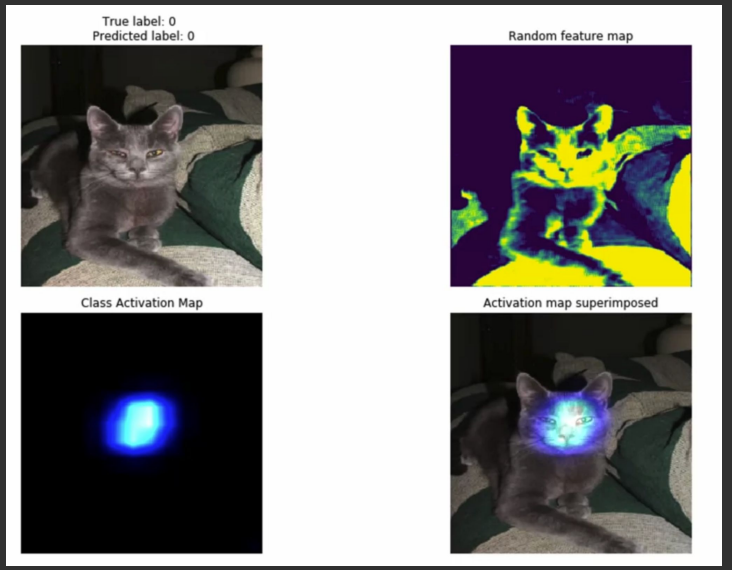


Image classification

Image Classification: What computers see



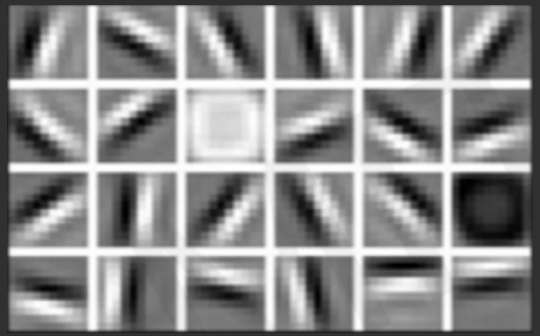
Array representation of image



Class activation map

Image Classification: What computers do

Low level features



Lines & Edges

Mid level features

Eyes, Nose, & Ears

High level features

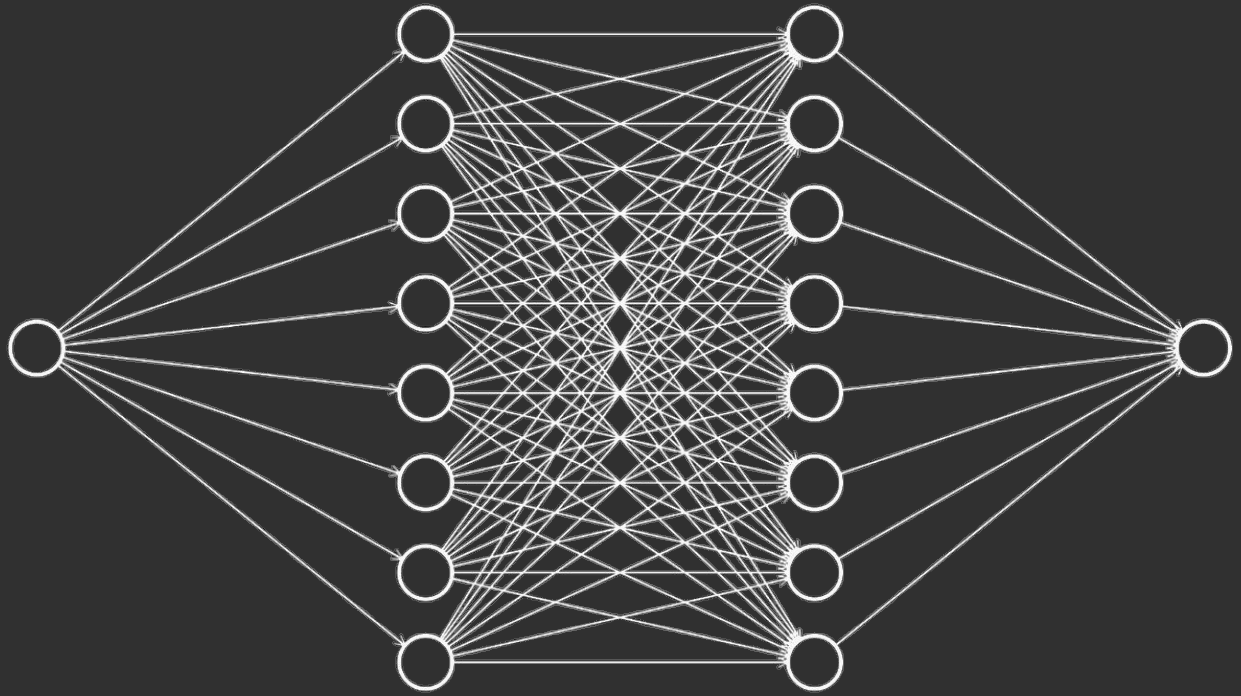
Facial Structures

Learning feature representations directly from data!

So, how can we make computer learning visual representation of images?

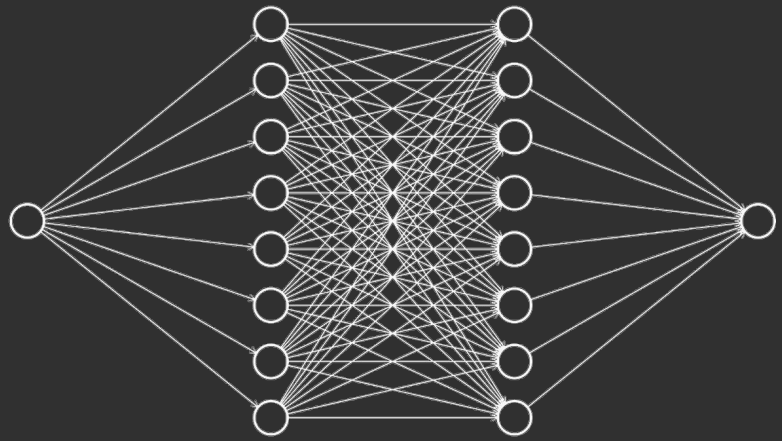
Fully Connected Neural Network

Fully Connected Neural Network



Fully Connected Neural Network: Disadvantages

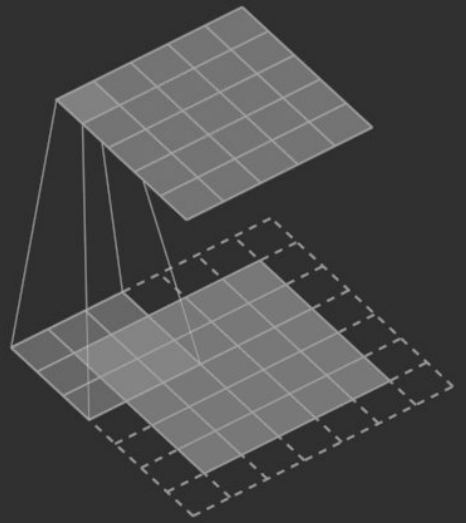
- No spatial information!
- 2D image input will be flatten out to 1D
- Lots of trainable parameters!
  (imagine a 100x100 px RGB image = 30k+ trainable parameter on neuron in first layer!)



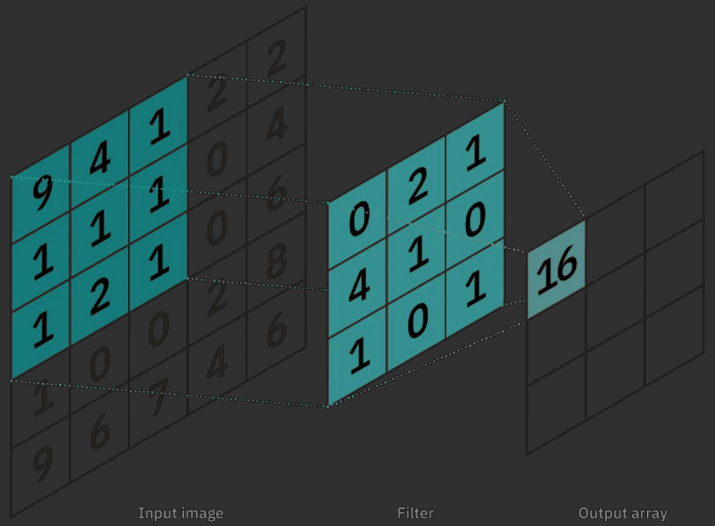
Convolutional Neural Network (CNN)

Convolution Layer

a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.



Convolution Layer



It’s a dot product operation of given input image and it’s filters.

Convolution: Feature Map

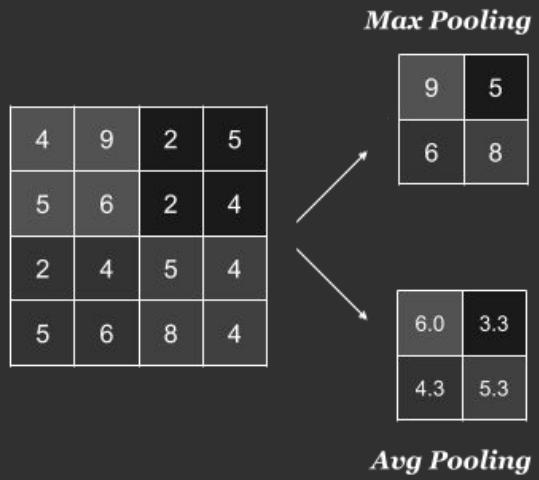
Original

Sharpen

Edge Detect

Pooling Layer

A common down-sampling technique in CNN




```
tf.keras.layers.MaxPool2D(pool_size=(2,2), stride=2)
```

```
tf.keras.layers.AveragePooling2D(pool_size=(2,2), stride=2)
```

CNN for Classification

1. Feature Extractor:

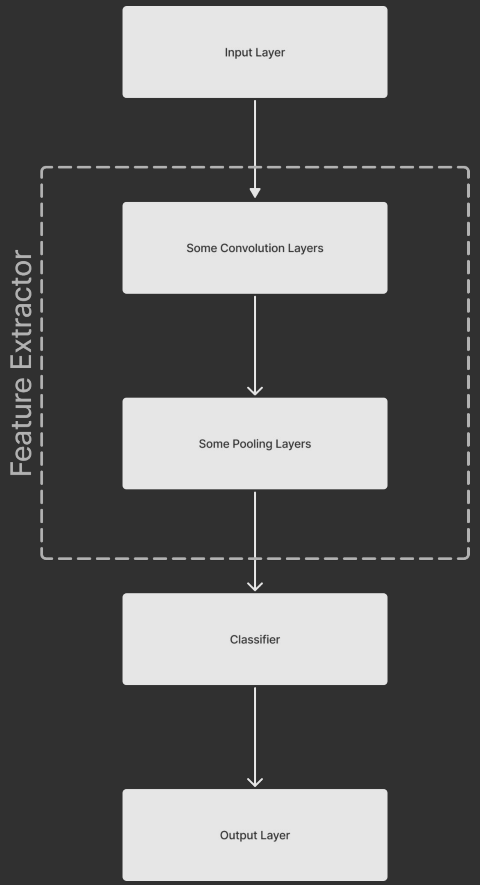
   Conv. Layer and Pooling Layer

2. Classifier:

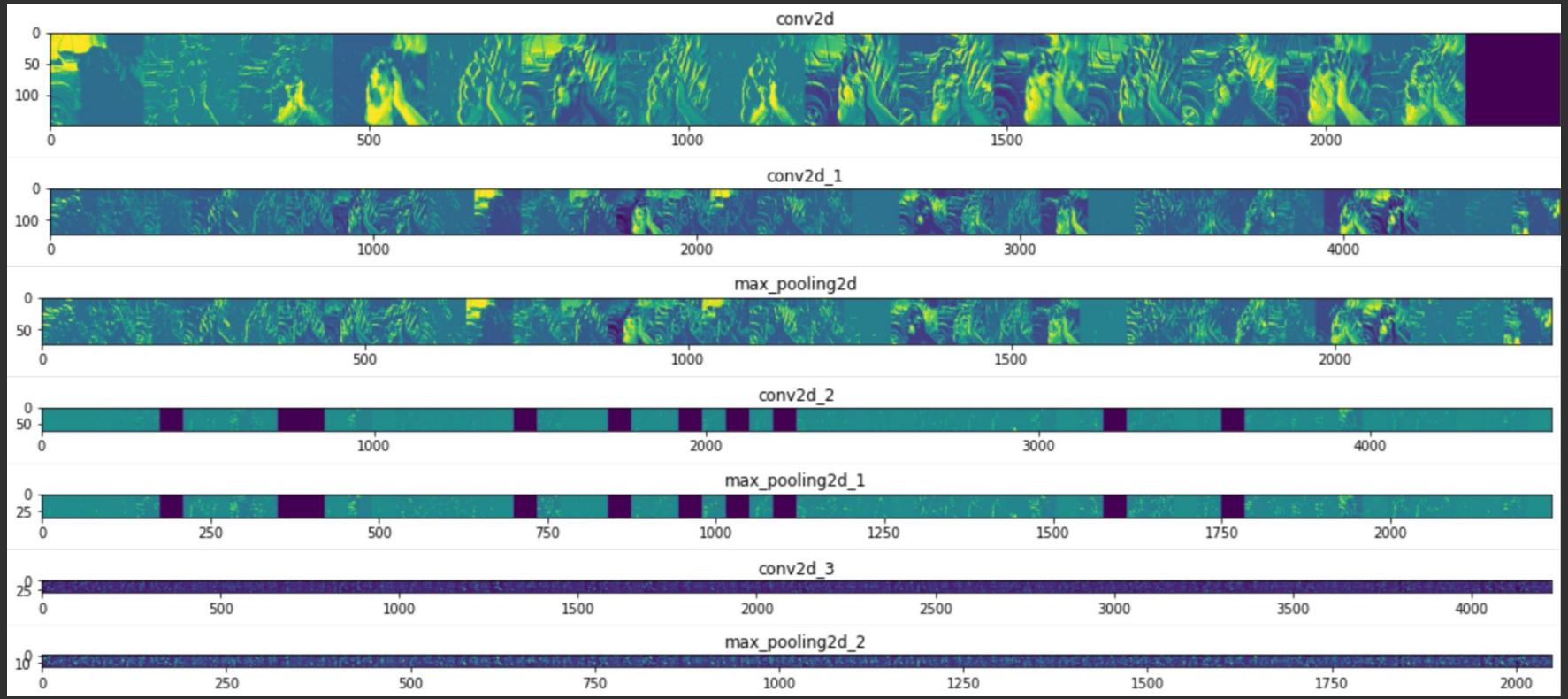
   Dense Layer with n neuron

3. Additional Layer:

   Dropout or Batch Normalization (to prevent overfitting)



CNN: Feature Extractor



CNN for Classification in Tensorflow

```
import tensorflow as tf
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(filters, kernel_size, padding, activation='relu',
input_shape=(image_shape)),
    ...,
    tf.keras.layers.MaxPooling2D(pool_size, strides),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(num_class, activation='softmax')
])
```

Hands-on: Cat vs Dog Image Classification

Colab : https://bit.ly/gdscugm_impro_handson

Practice Lab: Chest Cancer Detection

Colab : https://bit.ly/gdscugm-impro-practice

Exploration: Advanced Image Processing & Computer Vision

Crowd Counting (Object Detection)
Combined CSRNet & mobilenetv3_ssdlite

Human Pose, Face Landmarks, and Hand Tracking
Mediapipe API

Exploration: Advanced Image Processing & Computer Vision

Style Transfer (GANs)
Fast Neural Style Transfer

Image to Video (GANs)
NeRF: Neural Radiance Fields

Exploration: Advanced Image Processing & Computer Vision

You can explore more on this area from this topic:

- Advances Computer Vision using Tensorflow Functional API
- Generative Deep Learning
- Transfer Learning Techniques
- etc.

Q&A Session

Thank You

For further questions, contact me at daniyal.kausar@gmail.com