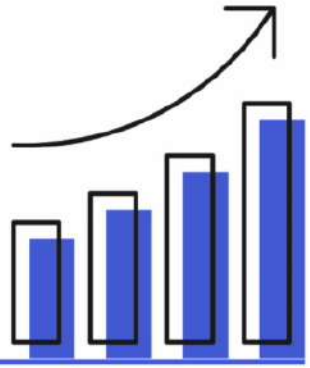
Data Science Project

Brain Tumor detection using Convolutional Neural Networks



Learning Outcomes:

In this project we will:

- Learn and understand about image classification using features
- Learn how to clean and process images for such tasks
- Learn about Convolutional Neural Networks
- Deploy our model as a Flask webapp

Part I (i): Image Classification

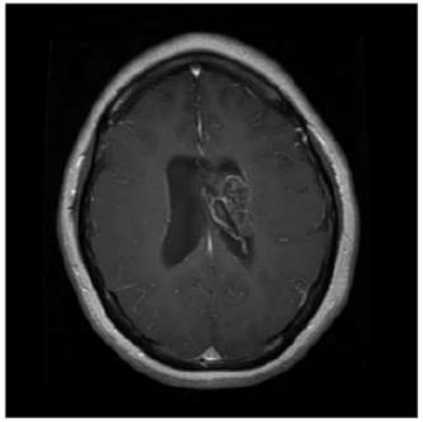
- From birth, us humans have learnt to visually identify things around us.
- We take notice of “features” in something we are looking at to compare and identify.
- For example, let us look at a simple example; of a car and a plane.

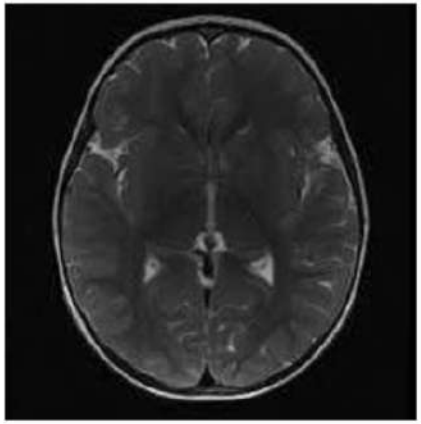
Part I (ii): Image Classification

Well, we clearly understand the difference between these 2 images, it is obvious! But how do we go about understanding which is a car and which a plane?

- Firstly, the plane has wings. The car doesn't.
- The car has 4 big wheels, the plane has multiple smaller wheels which are sometimes not even visible.
- Even though they share having windows, doors, etc. the shape and structure are completely different.

These obvious "features" are helping us identify one from another right?
Now let us look at these images. If we are asked which one shows a disease, other than to a trained doctor, it is not that obvious is it?





Part I (iii): Image Classification

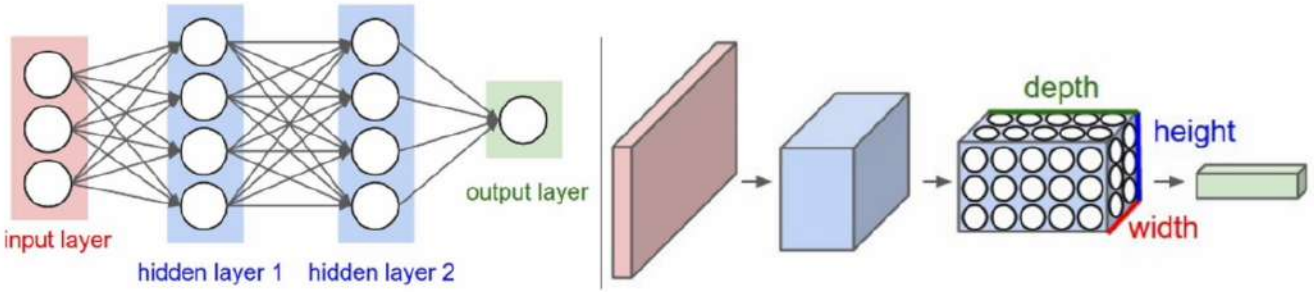
- Thus, machine aided image classification has seen a rise for some time now.
- When trained on a lot of data, it can help classifying pictures as and when needed.
- Be it healthcare, industry tools, etc. it has helped massively.
- A machine also has the same way of learning; understand "features" from known data and then apply its knowledge on new data.
- Although a machine's features are very much different from our features. We shall take a deeper look into this when we look at Convolutional Neural Networks.

Part I (iv): The Dataset

- The 2 scans of the brain we saw are a from the dataset we will be using in this project.
- It has 4600 scans of healthy and unhealthy (containing a tumor) brain, contained in two folders. Our model will have to classify each scan.
- The dataset can be found and downloaded from: https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset

Part II (i): Convolutional Neural Networks

- Convolutional Neural Networks are very similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.
- CNNs make the forward function more efficient to implement and vastly reduce the number of parameters in the network.



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Source: Deep Learning for Computer Vision, Stanford University

Part II (ii): Convolutional Neural Networks

Let us look at the Layers that make up the CNN:

•INPUT will hold the raw pixel values of the image, having the shape of (W,H,D). D is the depth denoting whether RGB (3), Grayscale (1), etc.

•CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This layer uses "kernels" to get features from the image.

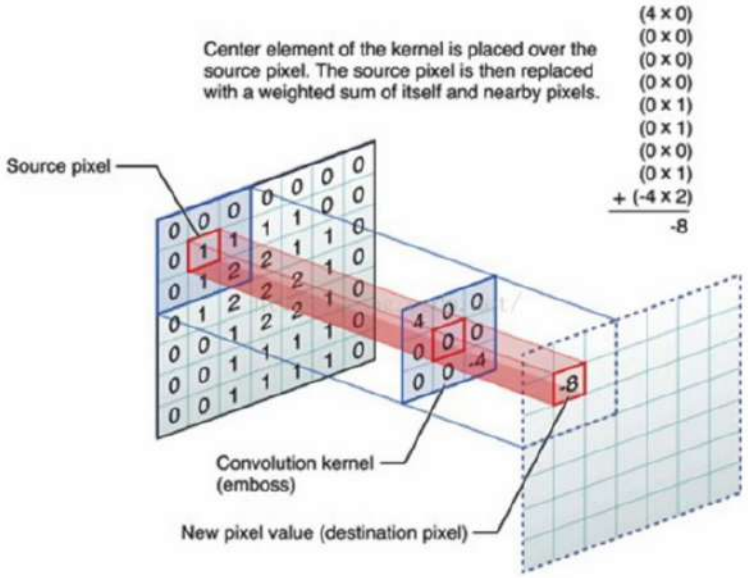
•RELU layer will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero.

•POOLING layer will perform a down-sampling operation along the spatial dimensions (W,H).

•Fully Connected layer will compute the class scores as in a regular neural network. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Source: Deep Learning for Computer Vision, Stanford University

Part II (iii): Convolutional Neural Networks



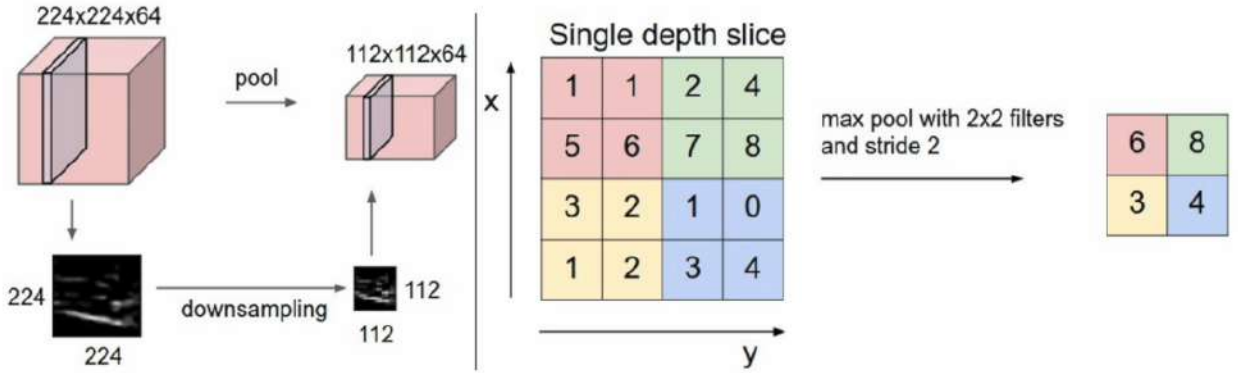
There are various types of kernels. We can specify the number of kernels to be used by tuning the hyperparameter "depth"

Source: Deep Learning for Computer Vision, Stanford University

Part II (iii): Convolutional Neural Networks

Pooling Layer:
Pooling layer down-samples the volume spatially, independently in each depth slice of the input volume.

- **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved.
- **Right:** The most common down-sampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).



Source: Deep Learning for Computer Vision, Stanford University

Part III: Implementing CNN and webapp

- We will be implementing our CNN model using TensorFlow and the high-level API Keras.
- We can access the layers mentioned earlier from TensorFlow and set the hyperparameters ourselves.
- We will be adding 2 layers of Convolutional Layers along with 2 Max Pooling layers.
- Finally we will have a Dense (fully-connected) layer with 512 neurons.
- We will also have an input layer defining the size of the image being fed into the network and an output layer having one neuron, since it is a binary classification problem.
- Lastly, we will save our model and make a Flask app for this project.