Neuro Fuzzy Techniques(CSL304)
Assignment(1-3)

Topic: Object Detection
Data-set: CIFAR-10
GROUP-16

Group Members:
1. BT19CSE028 ESHAN KUMAR JAIN
2. BT19CSE100 SHARMA OJAS RAJNISH
3. BT19CSE127 WAGH ABHISHEK SAMBHAJI

PROFESSOR INCHARGE:
1. Dr. POONAM SHARMA
2. SANTOSH SAHU SIR

TABLE OF CONTENT:

1. Introduction
2. Table of contents
3. Objective
4. Cifar-10 (Dataset)
5. Classes in the dataset
6. Layers Used Generally
7. Assignment-1 (Model used:- CNN)
8. About the CNN model (and convolutional layer)
9. Assignment-2 (Model used:- ResNet)
10. Architecture of ResNet block
11. ResNet continued
12. Assignment-3 (Model used:- VGG-16)
13. Architecture of VGG
14. Code implementation for VGG-16
15. Results and Comparison
16. Links to the Assignment

Objective:

Design and implement Deep Learning Models for Object detection

Dataset used: CIFAR10

Motivation: cifar 10 is a benchmark problem in the field of computer vision and neural networks. Repeatedly increasing the predictive ability of the model using this dataset increases the knowledge of the student and evolves the ability to form new neural networks for various less complex and diverse datasets.

Feasibility: The Utilities of the Image detection is that it forms the basis of many other downstream computer vision tasks, for example, instance segmentation, image captioning, object tracking, and more. Specific object detection applications include pedestrian detection, people
counting, face detection, text detection, pose detection, or number-plate recognition.

Cifar-10

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

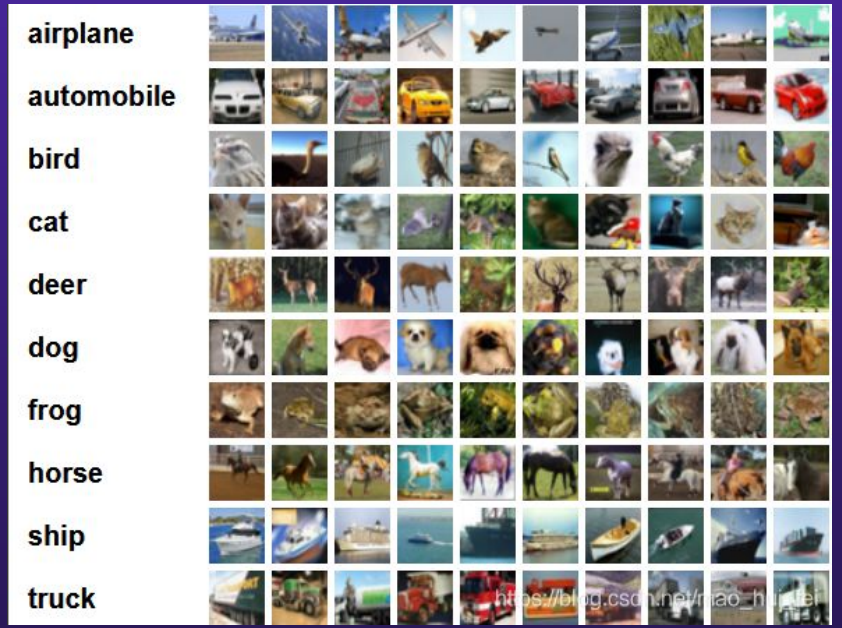
[Cifar-10 Python Version]

An image from the dataset
Dim: (32,32,3)
Class: Frog
Index: 6

CIFAR10 (cont..)

The label classes in the dataset are:

- airplane
- automobile
- bird
- cat
- deer
- dog
- frog
- horse
- ship
- truck



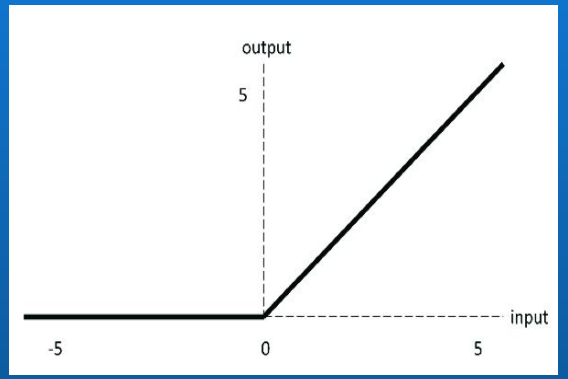
Number of images in training dataset: 50000 / 5000 image per class
Number of images in testing dataset : 10000 / 1000 image per class

Layers used generally :-

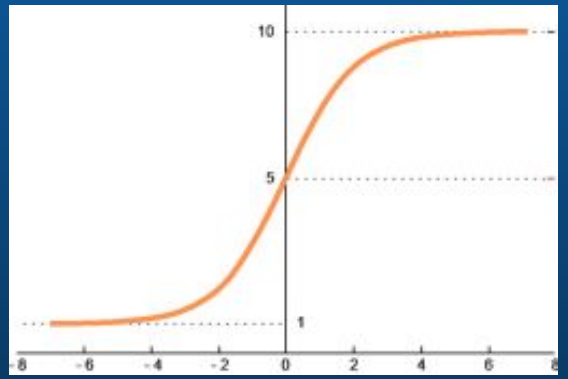
1) Convolutional Layer
2) Pooling Layer
3) Fully Connected Layer
4) Dropout Layer
5) Batch Normalization(Sometimes)
6) Activation Function
   a) Rectified Linear activation function (ReLU)

      Formula:- $y = \max(0, x)$

   b) Softmax Activation function



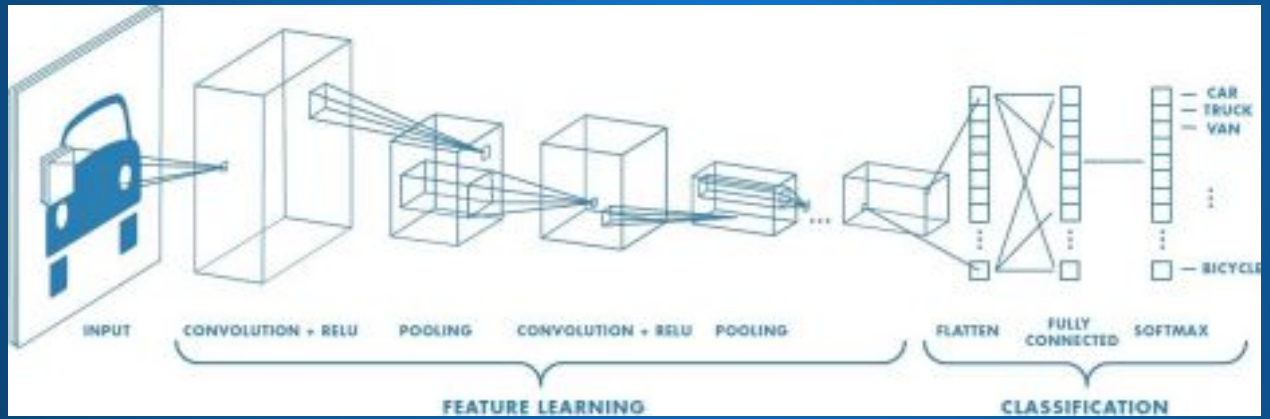
ReLU



Softmax

Assignment-1 Model : **Convolutional Neural Network (CNN)**

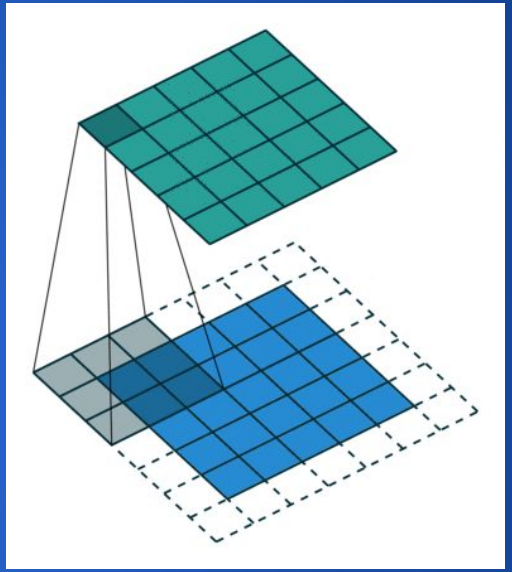
A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.



Our implementation:
input-> conv(x2) -> max_pooling -> conv(x2) -> max_pooling -> dropout -> flatten -> dense -> dense(output)

CNN algorithm has two main processes:

- **Convolution process:**
  This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but it is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.
- **A sampling process:**
  n pixels of each neighborhood through pooling steps, become a pixel, and then by scalar weighting ,adding bias and then by an activation function, produce a narrow n times feature .



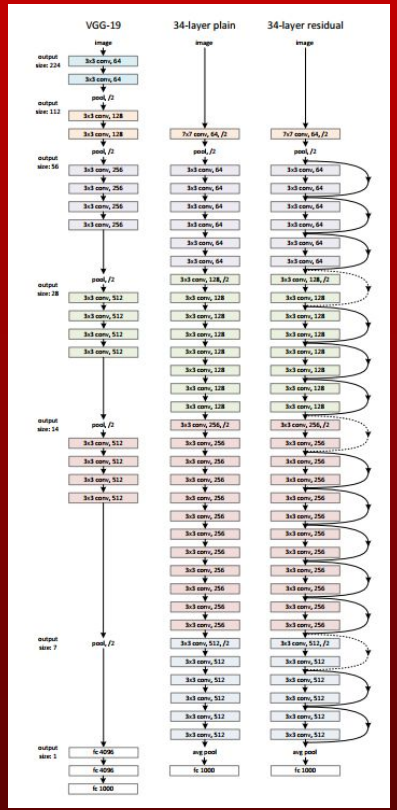
Animated representation of the convolution process

BASIC-STEPS :

- Importing the Libraries
- Importing and Loading Dataset
- Normalizing Images
- Building CNN
- Compiling the Model
- Testing the Model
- Analysing the result

Assignment-2 Model : ResNet(Residual Neural Network)

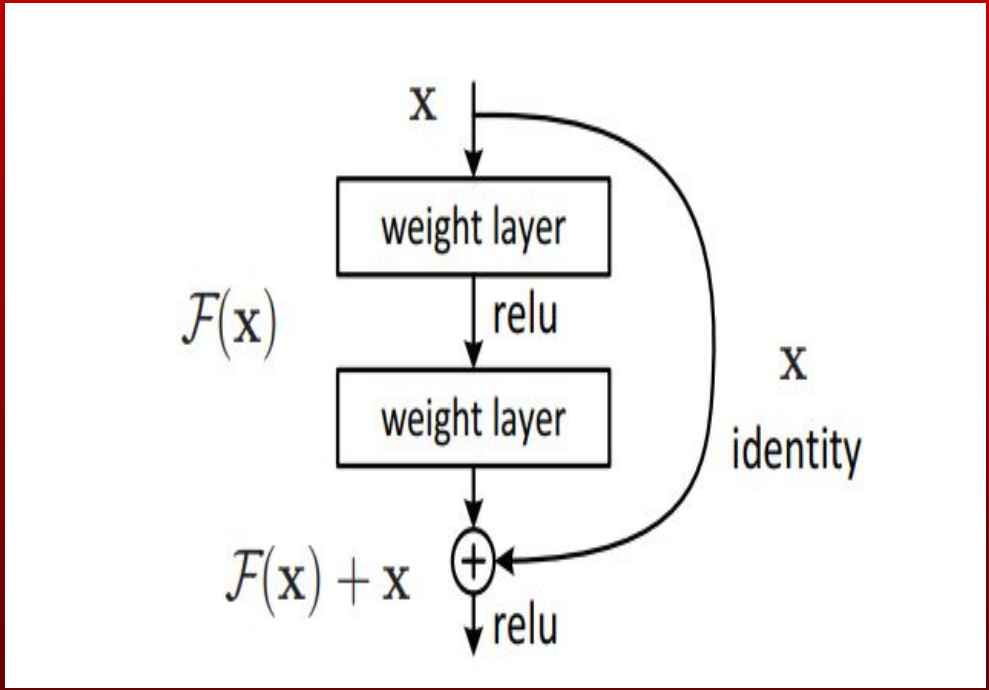
ResNet stands for Residual Network. It is an innovative neural network that was introduced in 2015. Residual Networks, or ResNets, learn residual functions with reference to the layer inputs, instead of learning unreferenced functions. Instead of hoping each few stacked layers directly fit a desired underlying mapping, residual nets let these layers fit a residual mapping. They stack residual blocks on top of each other to form a network: e.g. a ResNet-50 has fifty layers using these blocks (here we have done ResNet-20).

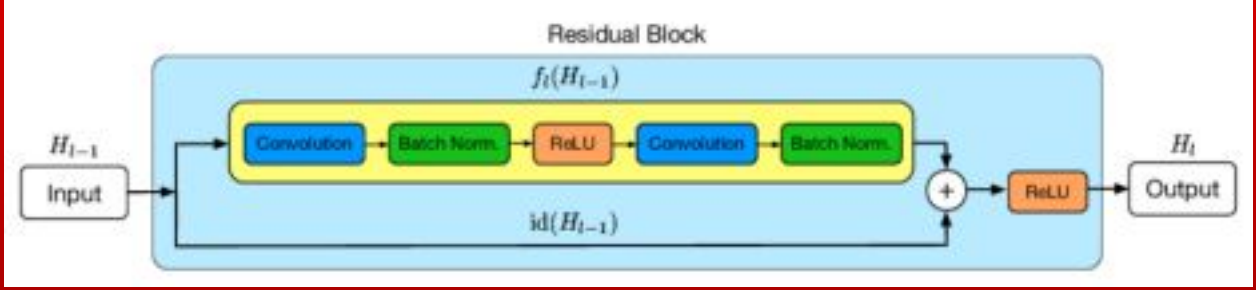


ResNet Algorithm:

ResNet Layer / Block:

- Specify the inputs
- Then, specify the weight/filter layers that are used in between
- At the end of the las last weight/filter layer pass the skipped connection of the previous input to the resnet block
- Pass the skipped layer and processed output to the ReLU activation function.





Pictorial representation of a singular resnet block

- The implementation is done mostly with the help of residual blocks
- The input is passed onto two channels within the residual block
  - One channel is further processed
  - The other is just passed to the end where earlier channel is processed
- After this both of the channels are combined and passed through ReLU activation function.
- The output from the block maybe input to other residual blocks or given into flatten then dense to then softmax to obtain the output.
- The ResNet model is fairly similar to a traditional CNN/DNN it's just the skip connections that makes it more efficient for the performance
- Skip connections enable us to further deepen the neural network.(It has been observed that there exists a certain threshold for every dataset wrt a neural implementation strategy for which after certain number of layers, the results start deteriorating so resnet proves useful in this aspect)

Assignment-3 Model: VGG-16 (Visual Geometry Groups)

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”.

Tools Used in implementation :-

1. Google Colab
2. Tensorflow
3. Keras
4. Matplotlib.pyplot
5. Numpy

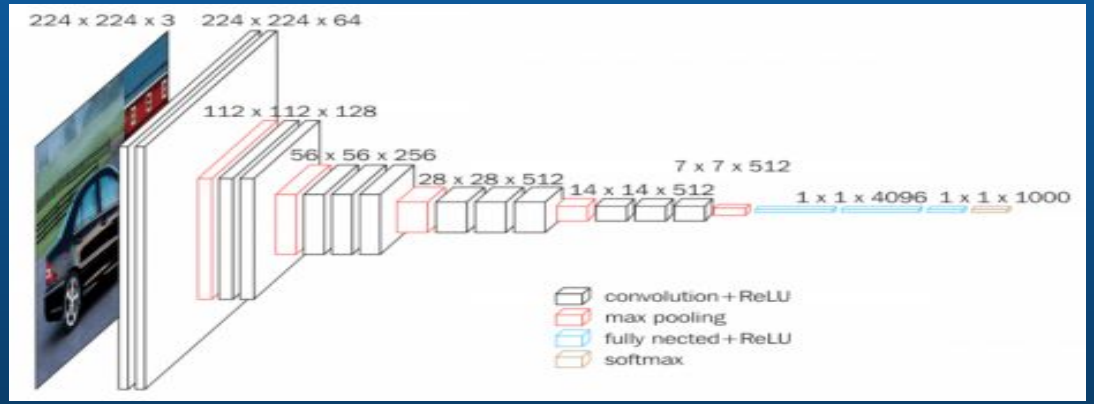


Image is taken from the official documentation

Architecture

The precise structure of the VGG-16 network:-

- The first and second convolutional layers are comprised of 64 feature kernel filters and size of the filter is 3×3. Then the resulting output is passed to max pooling layer with a stride of 2.
- The third and fourth convolutional layers are of 124 feature kernel filters and size of filter is 3×3. These two layers are followed by a max pooling layer with stride 2.
- The fifth, sixth and seventh layers are convolutional layers with kernel size 3×3. All three use 256 feature maps. These layers are followed by a max pooling layer with stride 2.
- Eighth to thirteenth are two sets of convolutional layers with kernel size 3×3.These layers are followed by max pooling layer with stride of 1.
- Fourteen and fifteen layers are fully connected hidden layers of 4096 units followed by a softmax output layer (Sixteenth layer) of 10 units.

Code Implementation For Visual Geometry Group (VGG-16)

1. Importing the Libraries
2. Importing the Dataset
3. Visualization the Dataset
4. Defining the Model
5. Model Summary
6. Scheduling the learning rate wrt epochs
7. Using ImageDataGenerator for randomising/modifying the data wrt image
8. Training the model
9. Testing and drawing the graphs for deducing the result.

Results and Comparison

Model	Training accuracy	Testing accuracy	Training loss	Testing loss
CNN (Basic)	0.9013	0.7439	0.4427	0.8363
ResNet	0.8765	0.8544	0.4927	0.5778
VGG-16	0.9643	0.9101	0.3840	0.6094

Accuracy and Loss Graphs for VGG-16

Links to the code:

- Assignment 1:
  https://github.com/OjasSharma29/Object-Detection/blob/main/Object_Detection.ipynb
- Assignment 2:
  https://github.com/OjasSharma29/Object-Detection/blob/main/G16_NFT_W21_A2.ipynb
- Assignment 3:
  https://github.com/OjasSharma29/Object-Detection/blob/main/VGG_16_CIFAR10.ipynb

Thank you !