# A solution of object recognition for car datasets using computer vision algorithms and applications

**Thesis Submitted to the Faculty of Engineering & Technology of Jadavpur University in partial fulfillment of the requirement for the Degree of Master of Engineering in Software Engineering.**


*SUBMITTED BY*

**Surya Pratap Kahar**

CLASS ROLL NUMBER: **001711002009**

EXAMINATION ROLL NUMBER: **M4SWE19013**

REGISTRATION NUMBER: **140968 of 2017-2018**

UNDER THE SUPERVISION OF

**Dr. SAIYED UMER**

ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
**ALIAH UNIVERSITY**
&

UNDER THE GUIDENCE OF

**Dr. BIBHAS CHANDRA DHARA**

PROFESSOR
DEPARTMENT OF INFORMATION TECHNOLOGY
**JADAVPUR UNIVERSITY**


**MAY 2019**

# CERTIFICATE OF SUBMISSION

I hereby recommended the thesis, entitled ***"A solution of object recognition for car datasets using computer vision algorithms and applications",*** prepared under my supervision by Surya Pratap Kahar be accepted in partial fulfillment of the requirements for the degree of Master of Software Engineering from the Department of Information Technology under Jadavpur University.

---

**Signature of Supervisor**
**Dr. Saiyed Umer**

**Assistant Professor**
**Department of Computer Science Engineering**
**Aliah University**

---

**Signature of Guide**
**Dr. Bibhas Chandra Dhara**

**Professor**
**Dept. of Information Technology**
**Jadavpur University**

Countersigned by:

---

**Head of the Department**
**Dr. Bhaskar Sardar**

**Dept. of Information Technology**
**Jadavpur University**

---

**Dean**

**Faculty of Engineering & Technology**
**Jadavpur University**

**DEPARTMENT OF INFORMATION TECHNOLOGY**
**JADAVPUR UNIVERSITY**

## CERTIFICATE OF APPROVAL

The thesis at instance is hereby approved as a creditable study of an engineering subject carried out and presented in a manner of satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

**Signature of Supervisor**
**Dr. Saiyed Umer**

**Assistant Professor**
**Department of Computer Science Engineering**
**Aliah University**

**Signature of Examiner**

**Signature of Guide**
**Dr. Bibhas Chandra Dhara**

**Professor**

**Dept. of Information Technology**
**Jadavpur University**

# Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by me, as part of my **Master of Engineering** in **Software Engineering** during academic session 2017-2019.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

_____

Name: **Surya Pratap Kahar**

Class Roll No.: **001711002009**

Exam Roll No.: **M4SWE19013**

Registration No.: **140968 of 2017-2018**

**M.E. in Software Engineering**

**Jadavpur University**

Thesis Title: ***A solution of object recognition for car datasets using computer vision algorithms and applications.***

# Acknowledgement

I would like to express my sincere gratitude to Jadavpur University for providing a conductive environment which helped me to pursue my study in the field of Master of Software Engineering. I am much obliged to be a student of School of Information Technology Department.

I would like to thank distinguished Dr. Bibhas Chandra Dhara for his valuable guidance and technical support in the accomplishment of my thesis. I am indebted to him for giving his precious time and motivation throughout the project.

Also, I would like to express my gratitude to my supervisor Dr. Saiyed Umer for giving me great intellectual freedom to pursue my topic of interest, for his immense patient and understanding and for guiding me each and every step of the process with knowledge and wisdom.

In addition I would like to thank all my seniors and friends for their resourceful advice during these days. Finally, I would also like to thank all teaching and non-teaching staffs who contributed, without any discrimination, to make this environment an excellent place for academic purposes.

Our journey begins by the path shown by our parents. I will be in debt to my parents who sacrificed and dedicated all these times to keep me in the path of honesty and hard work, and also shown faith in me.

(**Surya Pratap Kahar**)
Class roll no.: **001711002009**
Exam roll no.: **M4SWE19013**
Registration no.: **140968 of 2017-2019**
**M.E. in Software Engineering**
**Jadavpur University**
**Kolkata**

# ABSTRACT

Image classification is one of the very important problems in computer vision. They possess huge scope in healthcare, gaming, automobile, merged reality, security, social media platforms, visual search engines etcetera. This problem has been found to be solved with very good accuracy using deep learning techniques. Here, we have initially applied non-deep learning techniques (LBP, SIFT, HOG) and then applied deep-learning techniques (VGG16, ResNet50, InceptionV3). The deep-learning algorithms are trained on ImageNet dataset and then fine-tuned using training images of our car dataset. The results for deep-learning are fairly and expectedly high. The question which arises is, which algorithm solves this image classification problem in the best possible way? We have tried to answer this question by giving a comparison of the results obtained using different algorithms which have been applied to the same car dataset.

# Contents

# Chapter-1
# Introduction

## 1.1 Computer vision

We humans use our eyes along with brain to see and visually sense the world around us. Computer vision is the science that aims to provide the same capabilities of visual sense to machines and computers as we humans possess. Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding [3].

Computer vision can be defined in many ways. Some of the formal textbook definitions are:

- The construction of explicit, meaningful descriptions of physical objects from images.
- Computing properties of the 3D world from one or more digital images.
- To make useful decisions about real physical objects and scenes based on sensed images.

After knowing computer vision, we should now understand why do we need to care about studying computer vision. The best way to understand the importance of something is by knowing it's applications and its impact on human life. There are numerous applications of computer vision. Some of these are as follows:

- Biometrics: Fingerprint, iris and face matching remains some common methods in biometric identification.

- Forensics: Computer vision in combination with deep learning can be used to easily identify suspects and potential breakers, thieves, terrorists etc.

- Face recognition: Snapchat and Facebook use face-detection algorithms to apply filters and recognize us in pictures.

- Image retrieval: Google Images uses content-based queries to search relevant images. The

algorithms analyze the content in the query image and return results based on best-matched content.

- Gaming and controls: A great commercial product in gaming that uses stereo vision is Microsoft Kinect.
- Surveillance: Surveillance cameras are ubiquitous at public locations and are used to detect suspicious behaviors.
- Smart cars: Vision remains the main source of information to detect traffic signs and lights and other visual features.
- Gesture analysis: Making gestures in front of camera can make machines do many jobs for us like from unlocking mobile phones to switching ON or OFF the home A.C. and many more applications.
- Augmented reality: Computer vision quickly analyzes the terrain, surroundings and ambience using a camera and can reproduce virtual figures fitting the terrain serving a virtual purpose.

## 1.2 Image classification

**Image classification** is a technique which involves labeling an image based on the content of the image. A fixed set of labels will be present out of which our model will have to predict the label that is most suitable for a given query image. This is very different from how we humans see and classify images as the machine will see it in a stream of numbers and so it makes this task of predicting label challenging [1]. Images are nothing but matrix of numbers. The computer sees the images from a totally different perspective than we human does. Humans are trained from the very beginning of their life in order to classify different objects and living beings. Humans continuously get their brains trained by experiencing and learning. So, humans can easily classify the type of objects or living beings into different categories which they have already seen. The same concept has been applied for machines but instead of having a brain and training it, we have a model (analogous to human brain) which we train by feeding millions of images as well as their labels and by doing so, we are actually training the model. The model in turn makes

decision surfaces or boundaries or applies some sort of classification mechanism which allows it to accurately tell the class of an image if it is fed to it after training the model. If a totally new type of image comes then it will be extremely difficult, not only for the machines but also for the humans, to classify the new image.

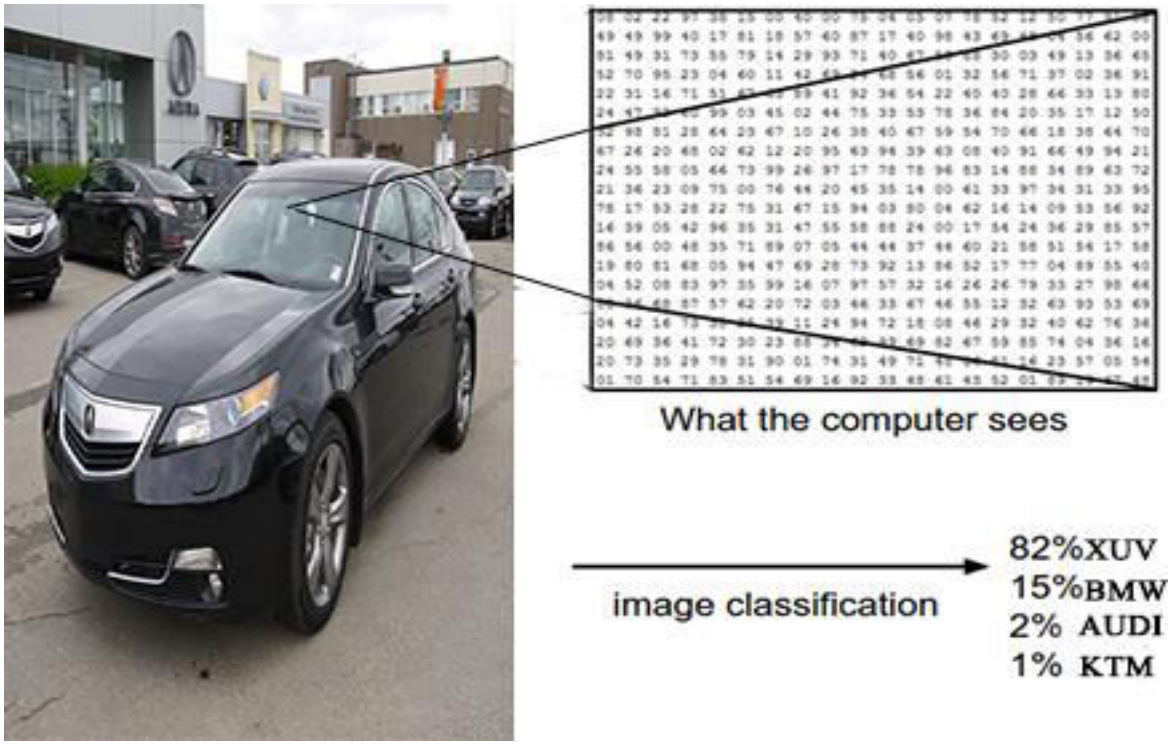


Fig 1.1: Image classification

Humans learn from errors only if some source of correct information is available. Likewise, machines also learn from errors as long as there is some source which says about the correct class label. This step is involved in training a model using dataset.

## 1.3 Object detection

**Object detection** is a technique that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos [2]. It basically involves recognizing various sub images and drawing a bounding box around each recognized sub image [1].

The task of object detection involves outputting bounding boxes and labels for individual objects. This differs from the classification / localization task by applying classification and localization

to many objects instead of just a single dominant object. We only have 2 classes of object classification, which means object bounding boxes and non-object bounding boxes. For example, in car detection, we have to detect all cars in a given image with their bounding boxes. Every object has its own set of special features that helps in classifying the class, for example, all circles are round. In this example, roundness of the circle is it's special feature. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

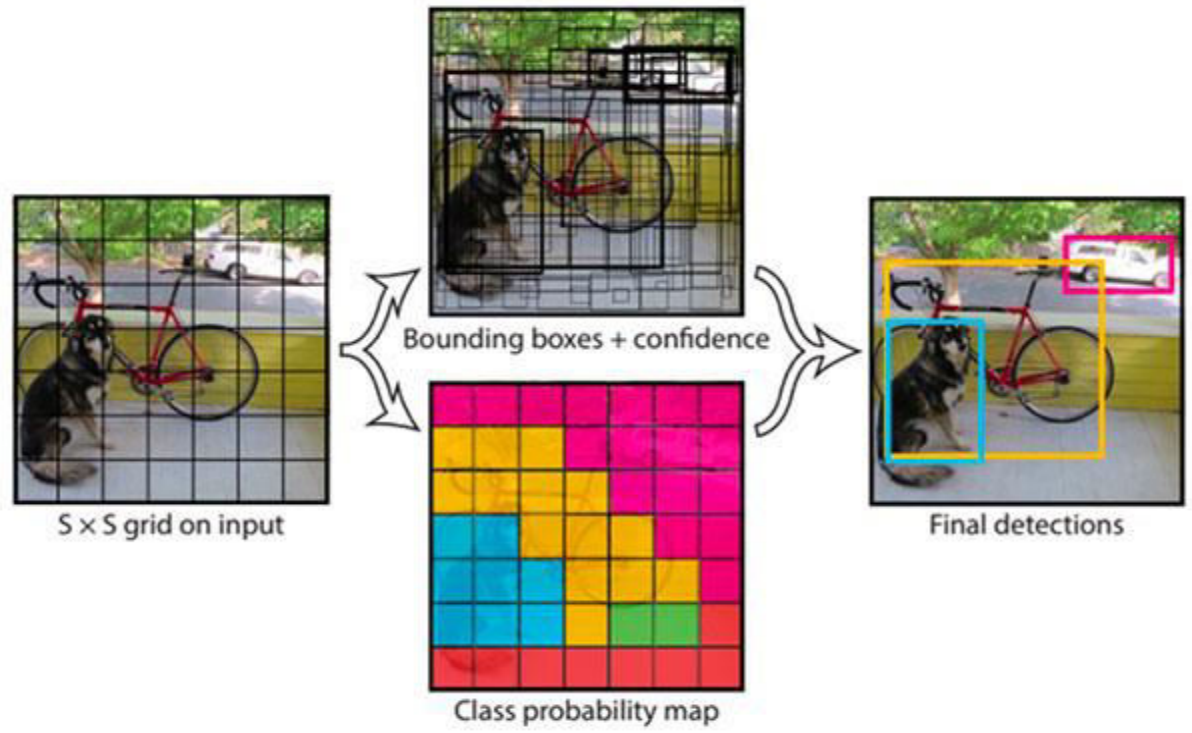


Fig 1.2: Object detection

Methods for object detection are generally categorized into either machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques that are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

Machine-learning based approaches:-

- Viola–Jones object detection framework based on Haar features
- Scale-invariant feature transform (SIFT)
- Histogram of oriented gradients (HOG) features

Deep-learning approaches:-

- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

Object detection has become an essential in this modern world and has wide range of uses in computer vision task. Some of the applications are as follows:-

- Face Detection
- Face Recognition
- Video object co-segmentation

## 1.4 Face recognition

Facial recognition is a category of biometric software that maps an individual's facial features mathematically and stores the data as a faceprint. The software uses deep learning algorithms to compare a live capture or digital image to the stored faceprint in order to verify an individual's identity.

High quality cameras in mobile devices have made facial recognition a viable and easy option for authentication as well as identification. For example, Apple’s iphone X and many other phones include Face ID technology that lets users unlock their phones with a faceprint mapped by the phone's camera. The software of the phone is designed with 3-D modeling to resist being spoofed by photos or masks, captures and compares over 30,000 variables. Even the purchases can be authenticated using Face ID with, for example, Apple Pay. The same technique can also be used for iTunes, App store and iBook store. Apple encrypts and stores faceprint data in the cloud, but authentication takes place directly on the device.

Now a simple question arises. How does a simple facial recognition application works? The facial recognition software identifies 80 nodal points on a human face. Nodal points here means end points which are used to measure variables of a person's face, such as the length or width of the nose, the depth of the eye sockets and the shape of the cheekbones. The system works by capturing data for nodal points on a digital image of an individual's face and storing the resulting data as a faceprint. The faceprint is then used as a basis for comparison with data captured from faces in an image or video.

Even though the facial recognition system only uses 80 nodal points, it can quickly and accurately identify target individuals under favorable conditions. However, if the subject's face is partially obscured or in profile rather than facing forward, or if the light is insufficient, this type of software is less reliable.

Let us now see some of the uses of facial recognition technology in order to understand the importance of facial recognition. Some of the applications are as follows:-

- They are used in Biometrics in order to record attendance of students or employees.
- Facial recognition are also used ass payments methods by companies like Amazon, MasterCard and Alibaba.
- Facial recognition is used to identify museum doppelgangers by matching a real person's faceprint with portrait's faceprint. This is applied in Google Arts & Culture application.
- We can take a picture of an individual and -- within seconds -- return the individual's name, date of birth and social security number (Person Identification). This has been developed by a research team at Carnegie Mellon University. This is a proof-of-concept iPhone application.
- Facial Recognition is used to identify criminals using a deep learning model which identifies whether the person in question has any past criminal records or not. This technique is also used to detect wanted persons.

## 1.5 ImageNet

The ImageNet project, which is a large visual database, is designed for use in visual object recognition software research. It consists of 14 million images which have all been hand-annotated by the project to indicate what objects are being pictured. At least one million of these

images, bounding boxes have also been provided. Number of categories in ImageNet is more than 20,000 with a typical category, such as “baloon” or “strawberry” containing hundreds of images. The database for the annotations is freely available directly from ImageNet. However, the actual images are not owned by ImageNet. Since 2010, the ImageNet project runs an annual contest named ILSVRC (ImageNet Large Scale Visual Recognition Challenge). In this contest, software programs compete in order to correctly classify and detect objects and scenes.

This database was presented for the first time as a poster at the 2009 Conference on Computer Vision and Pattern Recognition (CVPR) in Florida by researchers from the computer Science department at Princeton University. The primary researchers and inventors of ImageNet include Stanford University computer science professor and researcher Fei-Fei Li.

ImageNet crowdsources its annotation process. Image-level annotations indicate the presence or absence of an object class in an image, such as "there are balloons in this image" or "there are no balloons in this image". Object-level annotations provide a bounding box around the (visible part of the) indicated object. ImageNet uses a variant of the WordNet schema to categorize objects, augmented with 120 categories of dog breeds to showcase fine-grained classification. ImageNet was the world’s largest academic user of Mechanical Turk. The average worker identified 50 images per minute.

The question which now comes to our mind is why should we care about ImageNet? Why is ImageNet important? The ImageNet project is inspired by a growing sentiment in the image and vision research field – the need for more data. Ever since the birth of the digital era and the availability of web-scale data exchanges, researchers in these fields have been working hard to design more and more sophisticated algorithms to index, retrieve, organize and annotate multimedia data. But good research needs good resource. To tackle these problem in large-scale, it would be tremendously helpful to researchers if there exists a large-scale image database. This was the motivation for putting together ImageNet. The inventors of ImageNet had hoped that it will become a useful resource to our research community, as well as anyone whose research and education would benefit from using a large image database.

# Chapter-2
# Literature Survey

In 2018, Simon, Shlens and Le [4] tried to find the answer to the question, “do better ImageNet models transfer better?”. The cornerstone of computer vision is transfer learning. An implicit hypothesis which we make is that those models tend to perform better on most of the vision tasks that performs better on ImageNet. In this paper, they compared the performance of 16 classification networks on 12 image classification datasets. It was found that, on two small fine-grained image classification datasets, the benefits of pretraining using ImageNet was minimal which indicates that the learned features from ImageNet do not transfer well to fine-grained tasks. The results of their work shows that ImageNet architectures generalize well across datasets but ImageNet features seems to be less general as was hypothesized before.

In 2015, He, Zhang, Ren and Sun [5], proposed a paper which speaks about surpassing human-level performance on ImageNet classification. They showcased two aspects. In first aspect, they proposed a Parametric Rectified Linear Unit (PReLU) that generalizes the traditional rectified unit. PreLU had an advantage that it could do model fitting and with nearly zero extra computational cost and very little chance of overfitting. In second aspect, they derived a robust initialization method that basically considered the rectifier nonlinearities. This method enabled them to train very deep rectified models directly from scratch and also to investigate deeper and wider network architectures. Based on these two aspects, they had achieved a 4.94% top-5 test error on the ImageNet 2012 classification dataset. This was a very good improvement of 26% relative improvement over the ILSVRC 2014 winner. Their result was the first one to even surpass the reported human-level performance of 5.1% on this dataset.

In 2017 You, Gitman and Ginsburg [6], proposed and showed that the most natural way to speed-up the training process of very large networks is by using data-parallelism on multiple GPUs.In order to scale this stochastic gradient based methods to more number of processors, one need to increase the batch size to make full use of the computational power of each GPU. Currently, we use the state-of-the-art method of manipulating the learning rate which is

proportional to the batch size.They showed that by controlling the LR during the training process, we can efficiently use large batch size in ImageNet training. The found that the optimization difficulty leads to the accuracy loss for large-batch training. Only using the existing methods like linear scaling and warmup scheme are not enough to complicated application. They also determined that changing the network structure like adding batch normalization can help improve the accuracy, but adding only the batch normalization is not enough. They also proposed LARS(Layerwise Adaptive Rate Scaling), which uses different LRs for different layers based on the norm of their weights and the norm of the gradient. They found out LARS to be highly efficient on their experiment.

In 2018, He, Girshick and Dollar [7], proposed a paper where they reported competitive results on object detection and instance segmentation on the COCO dataset using standard models trained from random initialization. The results were seemed to be comparable and even better than the ImageNet pre-training counterparts even when using the hyper-parameters of th baseline system (Mask R-CNN) which were optimized for fine-tuning pretrained models. This was done for the sole exception of increasing the number of training iterations so the randomly initialized models may converge. They found out that even training from random initialization is surprisingly robust and their results even held when using only 10% of the training data, for wider and deeper models and for multiple tasks and metrics. Experiments performed by them showed that ImageNet pre-training speeds up convergence early in training, but it does not necessarily provide regularization or improve final target task accuracy.

In 2016, Simon, Rodner and Denzler [8], proposed a paper where they showcased a new set of pretrained models with popular state-of-the-art architectures for the caffe framework. Convolutional neural networks (CNN) which are pretrained on ImageNet are the backbone of most state-of-the-art approaches. This paper basically focused on batch-normalization-variants of AlexNet and VGG19 as well as residual networks. They were able to reproduce the ImageNet results of residual networks. All their models out-performed the previous pre-trained models.

In 2017, You, Zhang, Demmel and Keutzer [9], proposed a paper where they showed how to perform 100-epoch ImageNet training with AlexNet in just 24 minutes. Generally, finishing 90-epoch ImageNet-1k training with ResNet-50 on a NVIDIA M40 GPU takes 14 days. The number of total single precision operations is $10^{18}$ on doing this training. In 2017, the world's fastest

supercomputer could do a total of $2*10^{17}$ single precision operations per second. This meant that if we could somehow make use of it's full potential then we could do the training in just 5 seconds. However, the current bottleneck for fast Deep Neural Network (DNN) training is at the algorithmic level. If the batch size is small then we cannot make use of many processors. Therefore, for large-scale DNN training, we focus on using large-batch data-parallelism synchronous SGD without even losing accuracy in fixed number of epochs. The LARS algorithm enabled them to scale the batch size to an extremely large scale. They reported finishing the 100-epoch ImageNet training on AlexNet in just 24 minutes (which is the world record!).

In 2012, Krizhevsky, Sutskever and Hinton [10], proposed a paper where they trained a large,deep convolutional neural network to classify 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, they had achieved top-1 and top-5 error rates of 37.5% and 17% respectively, which is considered to be even better than the previous state-of-the-art. The neural network had 60 million parameters and 650000 neurons and it consisted of five convolutional layers. Some of them were followed by a max-pooling layers and three fully-connected layers with a final 1000-way softmax. In order to make their training faster, they used non-saturating neurons and a very efficient GPU implementation of convolution operation. They even employed a very recently developed regularization method called "dropout" that proved to be very effective in reducing overfitting in a fully-connected layers. The results of this paper shows that a large, deep convolutional neural network is capable of achieving extremely good results on a very highly challenging dataset using purely supervised learning.

In 2016, Hentschel, Wiradarma and Sack [11], proposed a paper on fine tuning CNNs (Convolutional Neural Networks) with scarce training data. Deep CNNs are shown to outperform previous state-of-the-art approaches for image classification. The success is said to be because of the availability of large labeled training sets such as those provided by ImageNet benchmarking initiative. CNNs have failed to learn descriptive features when training data is very small or scarce. Recent work and experiments shows that supervised pre-training on external data followed by domain-specific fine-tuning yields a very significant improvement in performance when external data and target domain show similar visual characteristics. This paper analyzes the performance of different types of feature representations for classification of paintings into art

epochs. They evaluated the impact of training set sizes on convolutional neural networks (CNNs) trained with and without external data. The obtained model was compared to linear models based on Improved Fisher Encodings. The results shown by them shows the superior performance of fine-tuned CNNs but they propose Fisher Encodings in cases or scenarios where training data is limited.

In 2015, Reyes, Juan and Camargo [12], published a paper on fine-tuning Deep Convolutional Networks for plant recognition. This paper described the participation of the ECOUAN team in the LifeCLEF 2015 challenge, where they had used a deep learning technique in which the complete system was learned even hand-engineered components. They had pre-trained a convolutional neural network using 1.8 million images and had used a fine-tuning technique in order to transfer the learned capabilities of recognition from general domains to specific domain challenges (in this case, the plant identification task). The accuracy obtained by this method of classification even outperformed the previous best method which was proposed in 2014. They even proposed a future work where they plan to evaluate deeper architectures of the convolutional neural networks (CNNs) and some domain specific adaptations so as to improve the performance of these algorithms in terms of classification to even a greater extent.

In 2015, Yanai and Kawano [13], proposed a paper on food image recognition using deep convolutional neural network (DCNNs) with pre-training and fine-tuning. In this paper, they examined the effectiveness of deep convolutional neural networks (DCNNs) for recognition of food pictures. Food recognition is considered to be a relatively harder problem than conventional image recognition tasks as it requires fine-grained visual recognition. They tackled this problem by using a combination of DCNN-related techniques such as pre-training the model with large-scale ImageNet data, fine-tuning the model and finally activation feature extraction from the pre-trained DCNN model. From their experiments, they concluded that the fine-tuned DCNN which was pre-trained using 2000 categories in the ImageNet which included 1000 food-related categories was considerably the best method which achieved a whooping78.77% as the top-1 accuracy for UECFOOD100 and 67.57% for UEC-FOOD256. These were the best accuracy till date of this paper. They also applied the food classifier which employed the best combination of the DCNN techniques to twitter photograph data. They achieved very good improvement on food data mining in terms of number of food photos as well as accuracy. They also found that DCNN

was very suitable for large-scale image data because it takes only 0.03 seconds to classify one food picture with GPU.

In 2018, Howard and Ruder [14] published their work on universal language model fine-tuning(ULMFiT) for text classification. The ULMFiT is an effective transfer learning method that we can apply to any of the tasks in NLP. Inductive transfer learning has greatly impacted the area of computer vision. The method proposed in this paper greatly outperforms the state-of-the-art on six text classification tasks. It reduced error by 1824% on most of the datasets and with only 100 labelled examples, it matches the performance of training from scratch on 100x more data. This is a huge leap in terms of improvement in performance and accuracy.

In 2004, Foody, Giles and Mathur [15], proposed a paper which showed a way towards intelligent training of supervised image classifications. In order to achieve a complete description of each class in feature space, a large training set is typically needed. For classification using SVM (support vector machine), only the training samples that are support vectors are required (which lies on part of the edge of the class distribution in feature space). The rest of the training samples have no contribution to the model training effectively. If we can somehow intelligently identify the regions where the support vectors will be present then we can sample out entire data to just include those regions of maximum probability (where support vectors are present) and then we can train using newly sampled smaller dataset which takes less time as well as reduces complexities. This is known as intelligent training using samples of training data. Applying this method on classification of agricultural crops from multispectral satellite sensor data, only 25% of the original training data was sampled for actually training SVM and the accuracy of crop classification using this method came out to be 92.5% (good accuracy). This was repeated by limiting the training on sample acquisition of only regions of specific soil type and the accuracy in this case came out to be near perfect (approx 99.99%) which is tremendously good considering we sampled the data to make our training job easier. Their paper illustrates the potential to use this method of sampling the most useful training samples to allow efficient and accurate image classification using SVM.

In 1999, Chapelle, Haffner and Vapnik [16], proposed their work on using SVM for histogram-based image classification. Traditional classification methods and approaches don't give good perfomance in image classification tasks because of the high dimensionality of feature space.

Their paper shows that support vector machines (SVMs) can work very well and give good performance in terms of accuracy on difficult image classification problems where the only features present are high dimensional histograms. They showed that it is possible to push the classification performance which are obtained on image histograms to a pretty high level having error rates as low as just 11% for the classification of 14 corel categories and 16% for a more generic set of objects. The only thing which was known during this process was that the input was some sort of color histogram or discrete density and nothing else was known about the task. They found that the extremely good performance was due to very good generalization ability of SVM's in high-dimensional spaces to the use of heavy-tailed RBF kernels.

In 2015, Ross Girshick [17] published his work at Microsoft research which was based on a fast region-based convolutional network method (Fast R-CNN) for object detection. Fast R-CNN is a technique which builds on previous work to efficiently classify the object proposals using the method of deep convolutional networks. The Fast R-CNN employs many innovations in order to increase training and testing speed while also increasing the detection accuracy. Fast R-CNN trains the very deep VGG16 network which is 9 times faster than R-CNN, is 213 times faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Fast R-CNNs are faster to train as well as test and are also very accurate. Fast R-CNN is implemented in Python and C++ (using Caffe). Fast R-CNN is a clean and fast update to R-CNN and SPPnet. The Fast R-CNNs gives state-of-the-art detection results.

In 2015, Rothe, Timofte and Gool [18], proposed a paper on deep expectation of apparent age from a single image. In their paper, they tried to estimate the apparent age in a given bunch of still face images using the technique of deep learning. They had used convolutional neural networks which made use of VGG-16 architechture[13] and are pre-trained on ImageNet for the purpose of image classification. There are only a limited number of apparent age annotated images. They explored the benefits of finetuning over crawled internet face images with available age. Crawler ran over 0.5 million images of celebrities from IMDB to Wikipedia and many other sites. This was the largest public dataset in order to do age prediction till date of writing their paper. They posed the age regression problem as a deep classification problem which was followed by a softmax classifier. The given method, Deep Expectation (DEX) of apparent age, first detects the face in the given test image and then it extracts the CNN

predictions using an ensemble of 20 networks on the cropped face. The CNNs of the given method (DEX) was finetuned on the crawled images and then on the given images having apparent age annotations. Explicit face landmarks are not used by DEX. The DEX discussed in this paper is the winner (1st place) of thhe ChaLearn LAP 2015 challenge on apparent age estimation with 115 registered teams which significantly out-performed the human reference.

In 2017, Akiba, Suzuki and Fukuda [19], proposed their work on extremely large minibatch SGD (Stochastic Gradient Descent) which included training ResNet-50 on ImageNet in 15 minutes. They showed that training ResNet-50 on ImageNet for 90 epochs can be achieved in 15 minutes with 1024 Tesla P100 GPUs. It was made possible by using a large minibatch size of 32,000. They employed several techniques such as RMSprop warm-up, batch normalization without moving averages, and a slow-start learning rate schedule. The paper also describes hardware details and system softwares used to achieve the given performance. The resultant top-1 score, after training on 90 epochs using 1024 GPUs, on the validation images was 74.94%+-0.09.

In 2017, Xia, Xu and Nan [20], proposed a paper on inception-v3 for flower classification. Flower classification is a very important subject in the field of Botany. A classifier having high accuracy will bring a lot of ease and fun to everyone's life and work. The background of the flowers are very complex due to which the similarity between the different species of flowers as well as the differences between the same species of flowers pose a difficult challenge in the recognition of flower images. The traditional method of flower classification is basically based on three features, namely: color, shape and texture. This classification requires people to select features for classification, and the accuracy is not high. Their paper made use of Inception-v3 model of TensorFlow platform. Transfer learning technology is required to retrain the flower category datasets, which can greatly improve the accuracy of flower classification. The classification accuracy of the given model are 95% on Oxford-17 flower dataset and 94% on Oxford-102 flower dataset, which is higher that other methods used for the given purpose of flower classification.

In 2016, Hassannejad, Matrella, Munari and others [21], proposed a paper on food image recognition using very deep convolutional networks which evaluated the effectiveness in classifying food images of a deep-learning approach based on the specifications of Google's image recognition architecture Inception. This architecture is a deep convolutional neural

network (DCNN) having a depth of 54 layers. They fine-tuned this architecture for classifying food images from three well known food image datasets : ETH Food-101, UEC FOOD 100, and UEC FOOD 256. They achieved top-1 accuracy of 88.28%, 81.45% and 76.17% and top-5 accuracy as 96.88%, 97,.27% and 92.58% on the given 3 datasets. The results obtained in this experiment significantly improved the best published results obtained on the same datasets and it also required less computation power because the number of parameters and the computational complexity are much smaller. The evaluation result suggests that the model benefits from new building blocks called “Inception modules”.

In 2017, Perez and Wang [22], published their work on the effectiveness of data augmentation in image classification using deep learning. In their paper, they explored and comapred many different solutions to the problem of data augmentation in image classification. Works already done before this paper has demonstrated the effectiveness of data augmentation through simple techniques like cropping, rotating and flipping input images. They artificially constrain the access to the data to a small subset of the ImageNet dataset, and then compared each data augmentation technique one by one. They experimented with GANs to generate images of different styles. They also proposed a method in order to design neural net so as to learn augmentations that best improve the classifier (they are called as neural augmentation). The successes and shortcomings of this methods are discussed on different datasets in this paper as well.

# Chapter-3

# Handcrafted Features

## 3.1 LBP

Local binary pattern (LBP) [36] is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. LBP has been used in computer vision for a wide range of applications like:-

- face recognition
- facial expression recognition
- pedestrian detection
- remote sensing and texture classification
- building powerful visual object detection systems

In the most common approach, each 3*3 window in a given image is processed in order to extract an LBP code. We need to threshold the center pixel of that window with the surrounding pixels (using either mean or median or actual center pixel, as thresholds). The steps involved in the overall process is as follows:-

1. Threshold the values in a neighborhood (in chosen image window) with the chosen threshold placing 1 where the value is greater or equal than the threshold and 0 otherwise.
2. Multiplying the resulting binary map with a predefined mask (usually incremental powers of two).
3. Sum the values to obtain an 8-bit LBP Code.

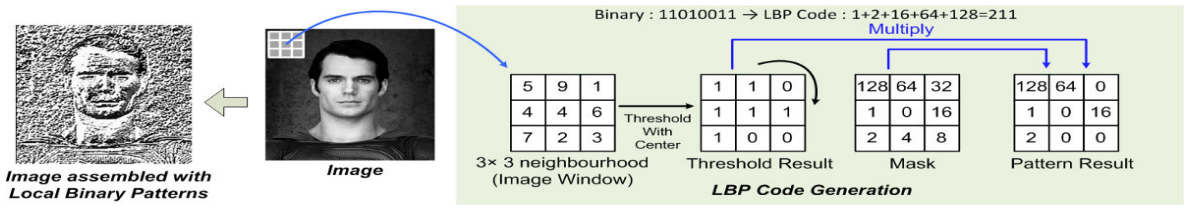


Fig 3.1: LBP code generation

Now, the next step (in order to build LBP based descriptor) requires dividing the LBP-based image into k blocks of w width and w height pixels (e.g. 2*4, 4*4). Now local image descriptors are build by getting the local histogram generated for each block of the given image. Now, all the local histograms are concatenated together to form a single global histogram. Now, the individual LBP code contains information at the pixel-level, the local histograms contains information on a regional level and the concatenated regional histograms contain a global description. The global histogram approach effectively expresses information in these three levels. The resulting histogram encodes both local and global characteristics in a compact representation which makes it more robust to object pose and illumination variations. We are actually just taking the relative values in LBP calculation therefore if image block is illuminated then values in each cell of block will increase otherwise will decrease in low light, but everyone will increase or decrease together and so the relative effect will still be zero. Therefore, LBP is not affected by illumination conditions. This is a very important feature of LBP.

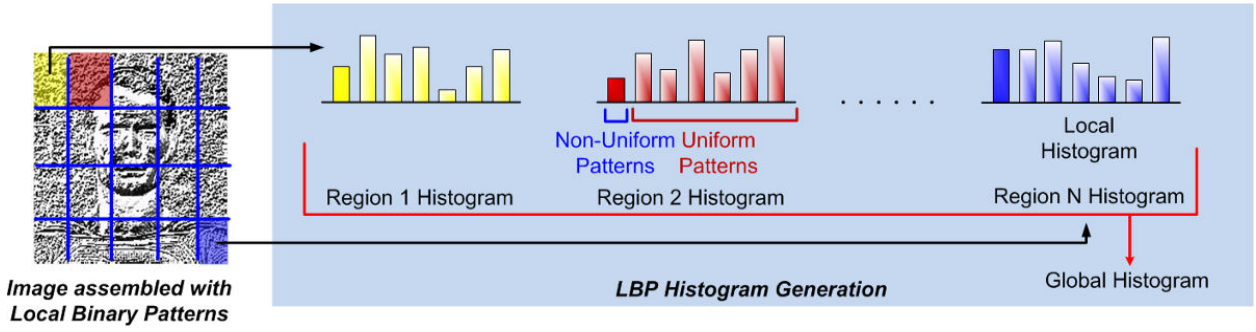


Fig 3.2: LBP: Histogram generation

Each local histogram measures the occurrence of each of the 256 possible LBP codes in the block. Below picture shows face description using LBP.

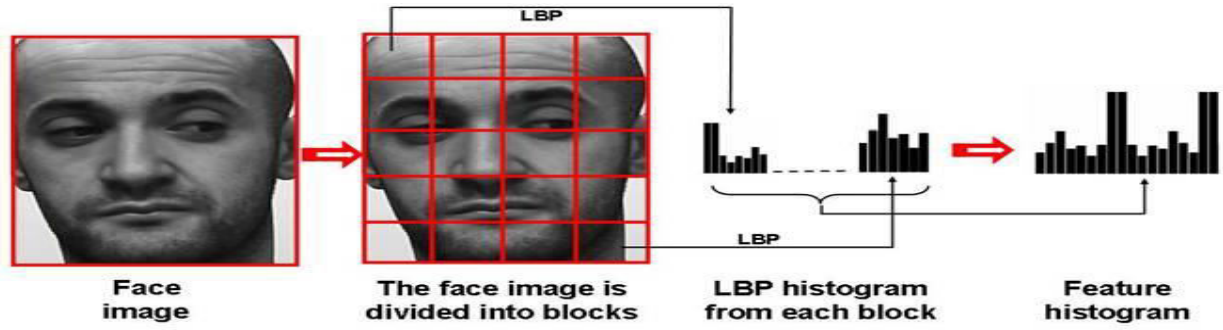


Fig 3.3: Face description with local binary patterns

## 3.2 HOG

The histogram of oriented gradients (HOG)[39] is a feature descriptor used in computer vision and image processing for the purpose of object detection. A feature descriptor [38] is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information. This technique counts the occurrences of gradient orientation in localized portions of an image.

Let us now see the steps described below in order to understand how to calculate histogram of oriented gradients (HOG).

- Preprocessing: A given image may be of any size. We analyze patches at multiple scales at many image locations. The only constraint which should be fulfilled is that the patches being considered should have a fixed aspect ratio. If aspect ratio is 1:2 then resolution can be 100*200 or 128*256 etc. The picture shown below has a resolution of 720*475. We have selected a patch of 100*200 of a runner in yellow dress (3rd from left). The patch is cropped out of the image and then resized to 64*128.

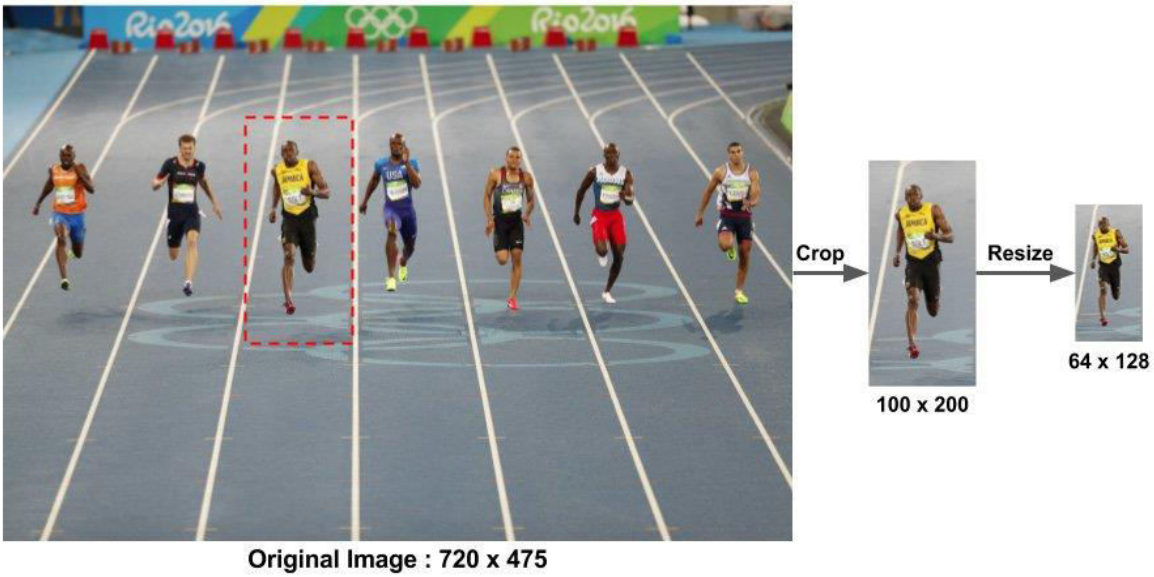


Fig 3.4: Preprocessing an image

- Calculate the gradient images: To calculate HOG descriptor, we need to first calculate the horizontal and vertical gradients. This is done by filtering the image by using the kernels

as shown below.

| -1 | 0 | 1 |
|---|---|---|

| -1 |
|---|
| 0 |
| 1 |

Fig 3.5: Kernels for HOG calculation

Magnitude and direction of gradient can be found using the following formula:-

$$g = \sqrt{g_x^2 + g_y^2} \ , \theta = \arctan \frac{g_y}{g_x}$$

The figure below shows the gradients.

Fig 3.6: Left : Absolute value of x-gradient. Center : Absolute value of y-gradient. Right : Magnitude of gradient.

We can notice that x-gradient fires on vertical lines and y-gradient fires on horizontal lines. Whenever there is a sharp intensity change then the gradient fires otherwise if region is smooth it doesn't fire. At every pixel, the gradient has a magnitude and a direction. For color images, the

gradients of the three channels are evaluated. The magnitude of gradient at a pixel is the maximum of the magnitude of gradients of the three channels, and the angle is the angle corresponding to the maximum gradient.

Calculate HOG in 8*8 cells: One of the important reasons to use a feature descriptor to describe a patch of an image is that it provides a compact representation. An 8×8 image patch contains 8x8x3 = 192 pixel values. The gradient of this patch contains 2 values (magnitude and direction) per pixel which adds up to 8x8x2 = 128 numbers. Individual gradients may have noise, but a histogram over 8×8 patch makes the representation much less sensitive to noise. Why have we taken 8*8 only and not say 32*32? It is a design choice informed by the scale of features we are looking for. HOG was used for pedestrian detection initially. 8×8 cells in a photo of a pedestrian scaled to 64×128 are big enough to capture interesting features (e.g. the face, the top of the head etc.). The histogram is essentially a vector of 9 bins corresponding to angles 0, 20, 40…160. Let us look at one 8*8 patch in the image and see how the gradients look.

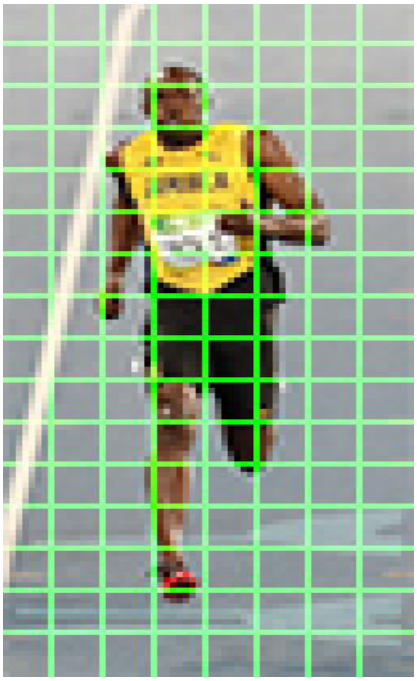

Fig 3.7 : 8*8 cells of HOG

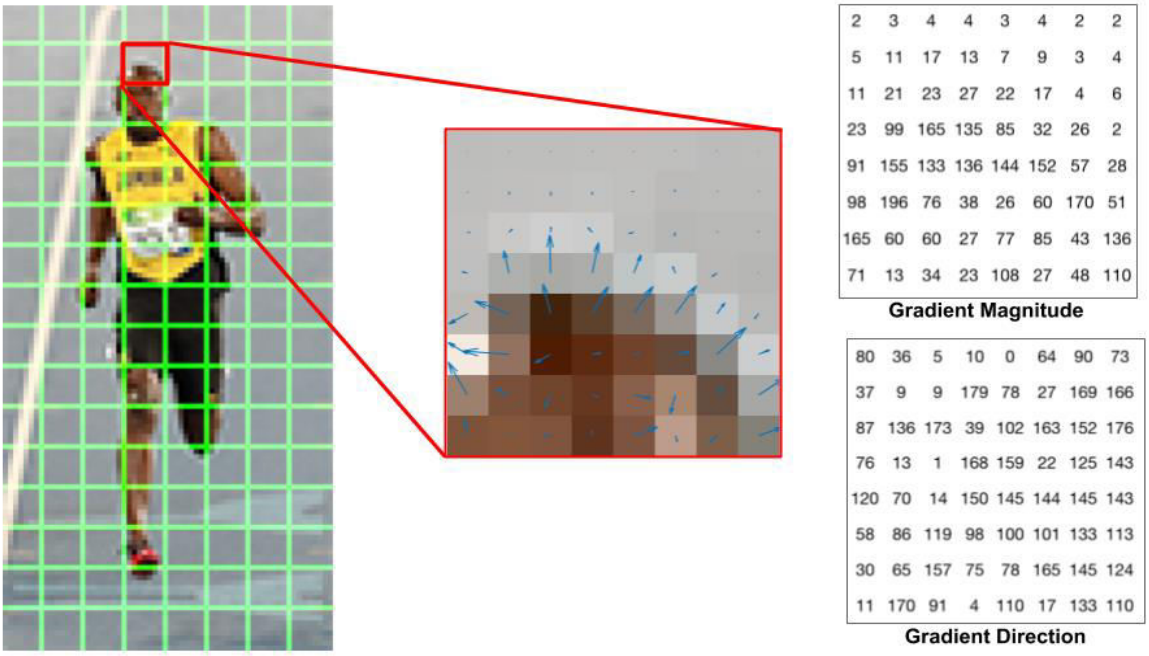


Fig 3.8: Center : The RGB patch and gradients represented using arrows. Right : The gradients in the same patch represented as numbers

The image in the middle overlaid with arrows shows the gradient. The arrow shows the direction of gradient and the arrow length shows the magnitude of gradient. We can notice that direction of arrows points to the direction of change of intensity and the magnitude shows how big the difference is. The figure to the right shows raw number for both magnitude and direction of gradients. The angles are between 0 to 180 degrees instead of 0 to 360 degrees because these are unsigned gradients and so negative and positive gradients are represented using same number. Empirically it has been shown that unsigned gradients work better than signed gradients for pedestrian detection.

Now, we will create histogram of gradients in these 8*8 cells using the 9 bins. The figure illustrates the process. We are looking at magnitude and direction of our 8*8 cell from our image. Now, we start scanning the gradient direction matrix and keep adding the corresponding magnitude values to corresponding bin. The addition to bin is weighted and so if gradient direction = 15 and gradient magnitude = 20 then we will add 15 to bin 20 and add 5 to bin 0 (weighted distribution). The bins wrap around and so 0 and 180 degrees are the same.

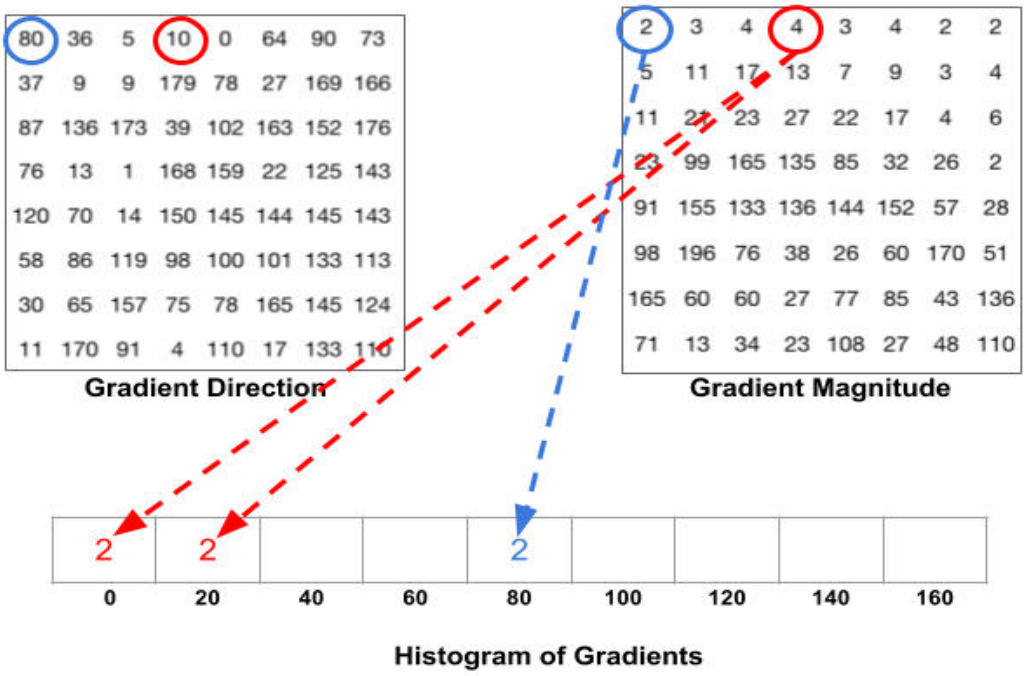


Fig 3.9: Filling 9-bins using gradient magnitude and direction

The contribution of all the pixels in the 8*8 cells is added up to create the 9-bin histogram. The histogram looks as shown below.

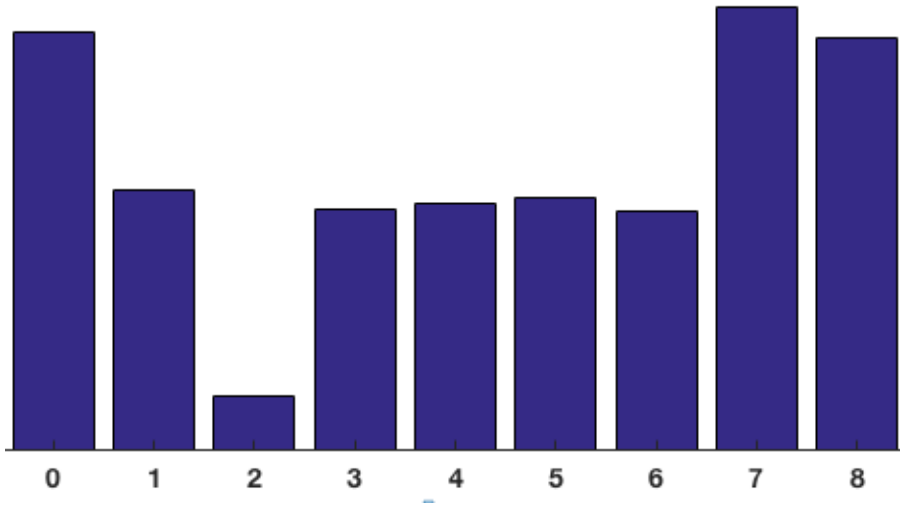


Fig 3.10: Histogram of our 8*8 patch

- 16*16 block normalization: We already created histogram based on gradient of image. The problem is that, gradients of image are sensitive to overall lighting condition. If we say decrease the lighting then the image gets darker, gradient value will reduce

proportionally and so histogram values will proportionally decrease. Ideally, we would like to make our descriptor independent of lighting condition. For this reason we normalize our histogram. We take L2 norm of a vector and divide each element of the vector by its L2 norm and the resulting vector is the normalized vector. Our normalized version will be independent of lighting condition.

- Calculate the HOG feature vector: We calculate the feature vector for the entire image patch. The smaller vectors are concatenated into one giant vector. SO, the dimensionality of vector will increase. Lets say we have each 16×16 block which is represented by a 36×1 vector. So when we concatenate them all into one gaint vector we obtain a 36×105 = 3780 dimensional vector.

## 3.3 SIFT

Matching features across different images is a common problem in computer vision. When all images are similar in nature (same scale, orientation etc), simple corner detections can work. But when we have images of different scales and rotations, we need to use the scale invariant feature transformation (SIFT)[41]. The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images[40].

Why do we even care about using SIFT? SIFT is not only scale invariant but we can change the following and still get good results:-

- Scale
- Rotation
- Illumination
- Viewpoint

Let us take an example and understand what we really want to do with SIFT. Let us take some target images (which we want SIFT to find) as shown on next page.

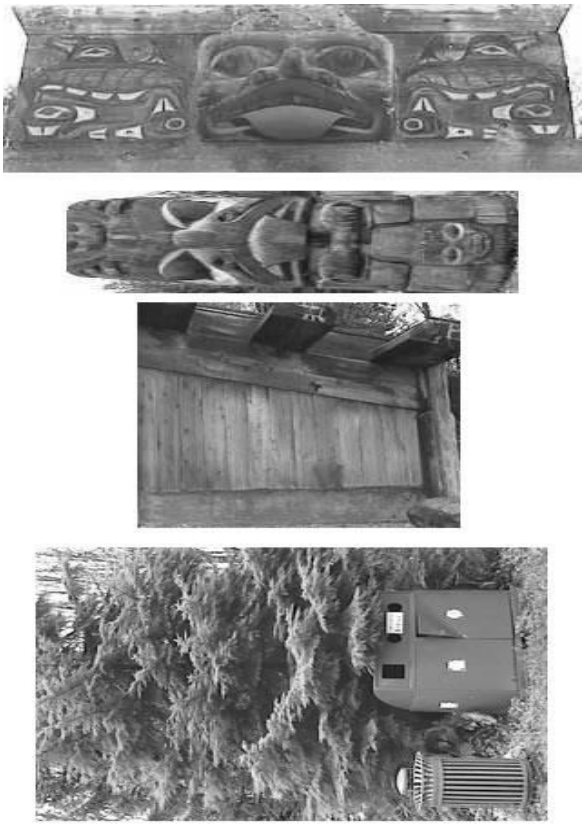

Fig 3.11: Target object Images

We want to find the target objects in a scene which is shown below.

Fig 3.12: Image on which SIFT has to be applied

The result is shown below.

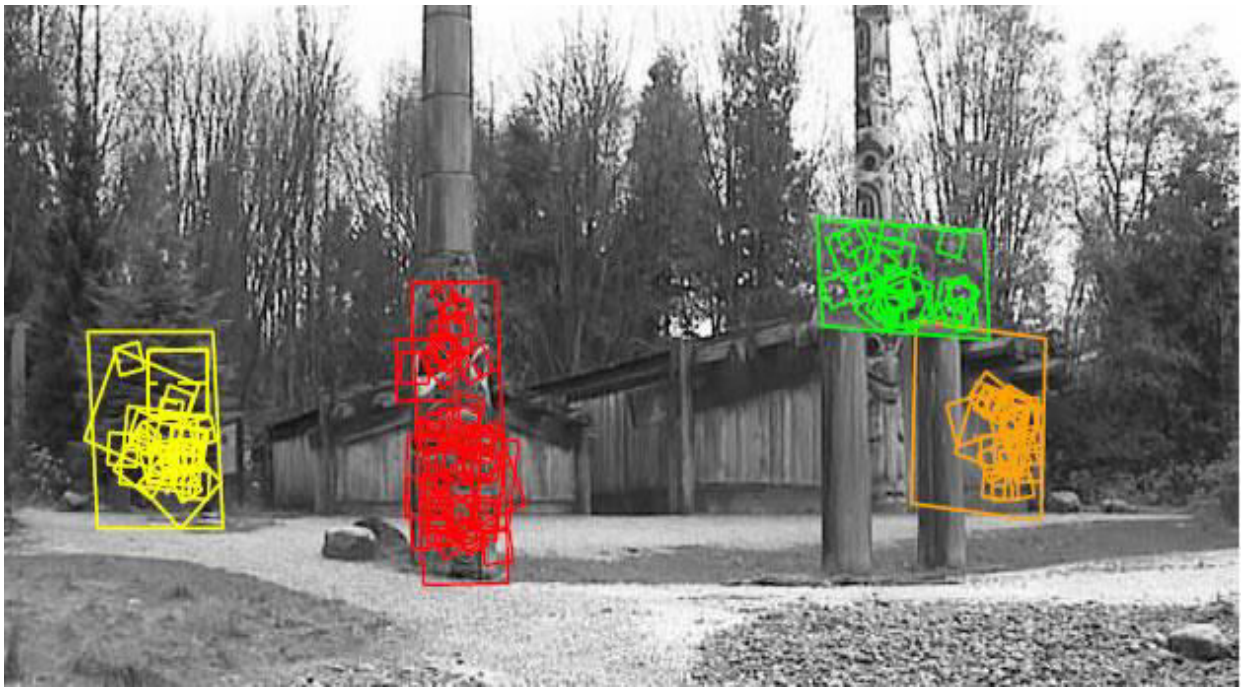

Fig 3.13: Image after applying SIFT

Having seen the work of SIFT, let us now understand the SIFT algorithm. SIFT has a lot of things going on and become quite confusing and therefore, the sift algorithm will be explained into multiple parts. Let us look at the outline of SIFT algorithm part by part.

- Constructing a scale space: This is the initial step. We create internal representations of the original image to ensure scale invariance. This can be done by generating "scale space".
- LoG approximation: The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
- Finding Keypoints: With the super fast approximation speed, we now try to find key points. These are nothing but maxima and minima in the difference of Gaussian image which we calculate in previous step.
- Get rid of bad keypoints: Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector can be used.
- Assigning an orientation to the keypoints: An orientation is calculated for each key point. Any

further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.

- Generate SIFT features: Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say we have 50,000 features. With this representation, we can easily identify the feature we a're looking for (say, a particular eye, or a sign board). This was an overview of the entire algorithm.

After we run through the algorithm, we will have SIFT features of our image. Once we have these features, we can do whatever we want using it. We can track images, detect and also identify objects. This algorithm is patented and hence only good enough for academic purposes.

Some of the applications of SIFT are:-

- Object recognition
- Robotic mapping and navigation
- Image stitching
- 3D modeling
- Gesture recognition
- Video tracking
- Individual identification of wildlife
- Match moving

# Chapter-4

# Classification Techniques

## 4.1 Support Vector Machine (SVM)

Support vector machines (SVMs), also known as support-vector networks[26], are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A support vector machine can be thought of as a discriminative classifier[27] formally defined by a separating hyperplane. In other words, given labeled data, like in supervised learning, the algorithm outputs an optimal hyperplane which categorizes new query points. In two dimentional space this hyperplane is a line dividing a plane in two parts wherein each class is present in either side.

Suppose we want to do binary classification task then how can we decide on a separating hyperplane. Take an example dataset as shown below. How can we draw a separating hyperplane?

Fig 4.1: Binary class dataset in 2D

We might think of a strategy to separate these binary class points by using a straight line as shown in figure on next page.

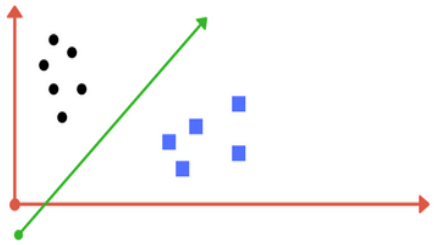

Fig 4.2: Straight line dividing dataset in two classes

This straight line does a fair job in diving the class into two parts so that if any query points comes from say test dataset then according to its position on 2D plane, it will either be classified as negative class or a positive class by the straight line. Therefore, we can say that our given straight line is working as expected. But now the question comes, we can draw many different parallel and non-parallel lines separating the classes, but how do we decide on which straight line to consider as our target line(in 2D) or plane(in 3D) or hyperplane(in nD)? This is decided by making use of support vectors as shown in the figure below.

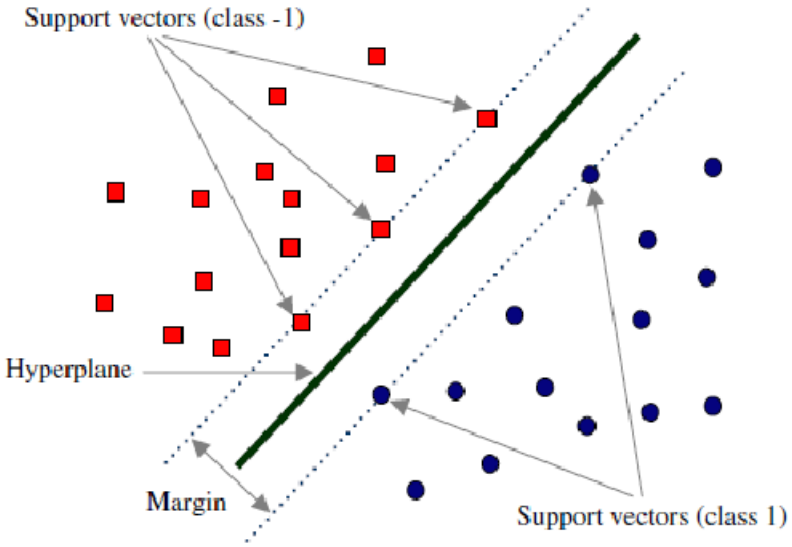


Fig 4.3: Calculating optimal plane for SVM

We connect the boundary points in negative class by a straight line and take another straight line which is parallel to first line and now connect the boundary positive points. Now we calculate the distance between these lines and then calculate the perpendicular to the distance line(i.e. the line perpendicular to both these boundary lines). The new line is exactly halfway between between

the boundary lines and this calculated line serves as the optimal hyperplane for the support vector machine(SVM).

The advantage of choosing the optimal hyperplane instead of just any straight line dividing these classes is that it reduces the chances of making error on future unseen data. This is because this optimal line is the one which is equally farthest from the positive class as well as negative class. Now if any data point comes and falls between our optimal line or plane and the negative class then that point will be classified to be negative and vice-versa. If we take just any line other than the optimal line then the given data point may also have got classified as positive point if the assumed line is closer to negative dataset and many points may incorrectly get classified as negative point if the assumed line is closer to positive dataset. Therefore, we can conclude that optimal line or plane or hyperplane does the best job of classification for support vector machine (SVM).

If we have different type of dataset (as shown in figure below) which cannot be directly classified by using a straight line or plane or hyperplane then what do we do?

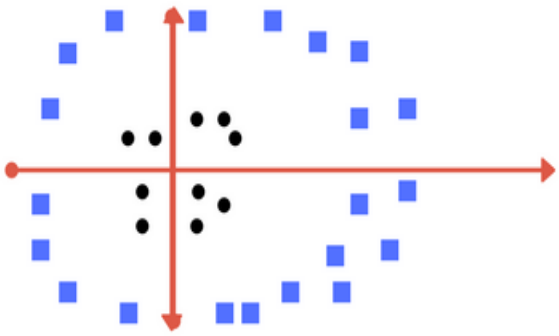

Fig 4.4 : Can we draw separating line on this dataset?

This doesn't seem to be possible to be divide using a line or plane but the truth is, we can still divide the classes here using a straight line but this cannot be directly done. What we need to do is, we need to first apply something known as feature transformation. We add one more dimension as we call it z-axis. Lets assume value of points on z plane, $w = x^2 + y^2$. In this case we can manipulate it as distance of point from z-origin. Now if we plot in z-axis, a clear separation is visible and a line can be drawn .

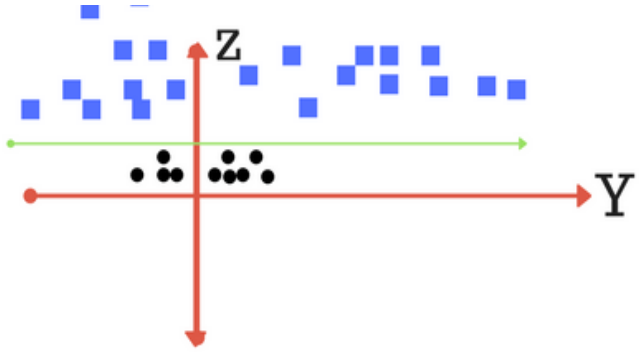


Fig 4.5: Plot of ZY axis (a separation can be made here)

When we transform back this line to original plane, it maps to circular boundary as shown in image E. These transformations are called kernels. The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. We can see that feature engineering is a great mathematical tool in order to process datasets and make them suitable for SVMs. Feature engineering is the most important aspect of a data scientist in machine learning.

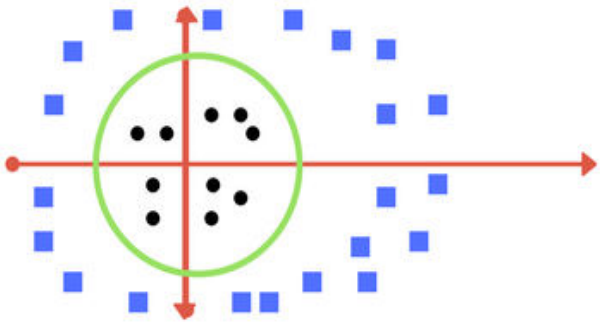

Fig 4.5: Transforming back to x-y plane, a line transforms to circle.

SVMs is very popular due to its simplicity, ease of use and wide applications. Some of the applications of SVMs are as mentioned below:-

- SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines.

- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach.
- Hand-written characters can be recognized using SVM.
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support-vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

We discussed all about linear SVMs and they are of two types as follows:-

- Hard-margin SVM
- Soft-margin SVM

In few words we can say that SVMs are simple and easy to use yet very powerful classification machine.

## 4.2 Convolutional Neural Network

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines[28]. Researchers and enthusiasts, both in the similar manner, work on numerous aspects of the field to make amazing things happen. One of such very important domain is Computer Vision. The prime agenda of this field is to enable machines to view the world as we as human beings do, perceive and feel it in a similar way and even use the knowledge for a variety of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing (NLP) and many others. The development and advancements in Computer Vision using Deep Learning has been constructed and perfected with time. This has been done primarily over one particular algorithm known as the Convolutional Neural Network.

Convolutional neural networks (CNNs), just like neural networks, are made up of tiny units called neurons (just like the ones in our brains) having learnable weights and biases. Each neuron recieves many inputs, calculates the weighted sum over them and then it passes it through an activation function and finally responds with an output. The entire convolutional network has a loss function. Let us now understand about neurons.

In order to understand neural networks in depth, we need to first look at how does neurons work. The neurons of CNN are assumed to be analogous to our human brain neurons and therefore, the entire network build out of these millions of tiny neurons forming a network structure mimics our human brain and so tries to give intelligent outputs.

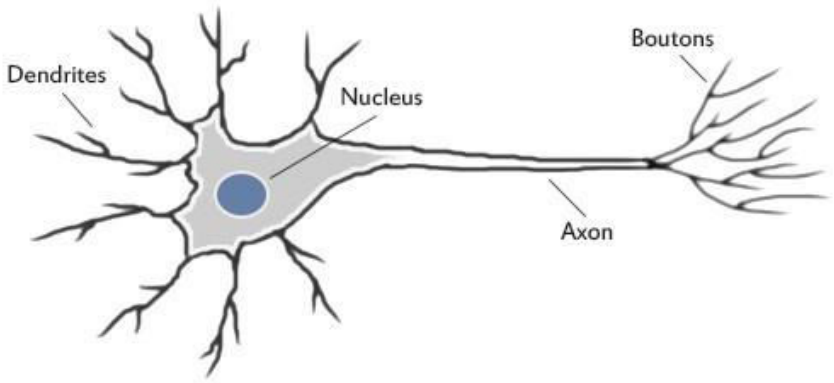


Fig 4.6: A brain neuron

Our human brain uses a very large interconnected network of neurons for processing information and to model the world around us. The neurons makes use of tiny structures called dendrites in order to collect inputs from other neurons. The neurons calculates the weighted sum of all the inputs and if it crosses a given threshold margin then that neuron fires. This fired signal is sent to other connected neurons by making use of axons (like a pipe or wire). Now, the next question which arises is, how do we model artificial neurons? The figure[31] given below shows a neuron connected with n other neurons and therefore, it receives n inputs (x1, x2, …..xN). This configuration is called a perceptron.

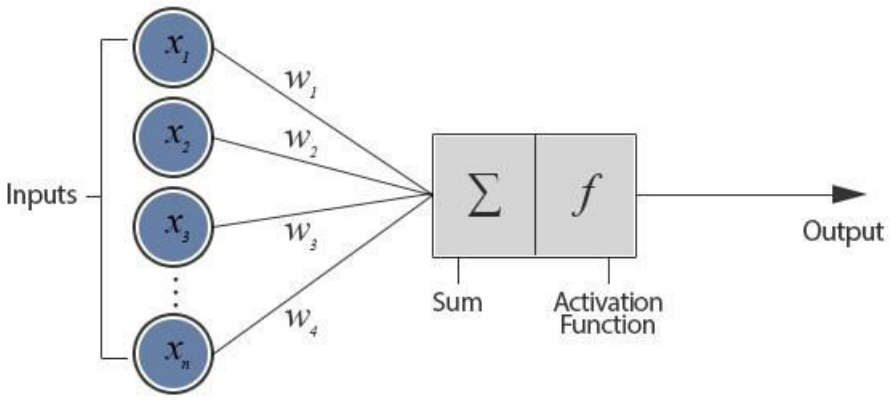


Fig 4.7: Model of an artificial neuron

The inputs(x1, x2, …..xN) as well as the weights (w1, w2,…..wN) are real numbers and can be either positive or negative. We can in the figure that the perceptron consists of weights, summation processor, an activation function and a threshold processor (also known as bias). All the given inputs are weighted individually and then they are added together and only after that they are passed into the activation function. Among the different types of activation functions available, one of the simplest is the step function. A step function outputs 1 if the input is above threshold, otherwise outputs 0. Other popular activation function is sigmoid.

Let us take an example where input1(x1) = 0.6, input2(x2) = 1.0 and weight1(w1) = 0.5 and weight2(w2) = 0.8. Let threshold be 1.0 . Let us now find out whether the given inputs to a neuron makes it fire. Weighing the inputs and adding them together gives:-

$$x1w1 + x2w2 = (0.6 \times 0.5) + (1 \times 0.8) = 1.1$$

Since the total input is higher than given threshold, therefore the given neuron will fire.

Now how do we train these perceptrons? Let us first understand how do we train our brain's perceptrons. Lets take an example and understand how do we try teaching a child in order to recognize a bus. We will definitely show her examples and will say that it is a bus and that one is not a bus. This is repeated until the child learns the concept of what a bus exactly is. Now if any new things are seen by that trained child then we would expect him to differentiate that new object whether it is a bus or not. This is the same idea as we have behind perceptron. In perceptrons, input vectors from a given training set are fed to the perceptron sequentially and the weights are continuously modified according to the given equation.

W(i) = W(i) + a*(T-A)*P(i) ; where a is learning rate, W is weight.

W: Weight

P: Input vector

T: Correct output or label.

A: Perceptron output

When we complete an entire pass of input training vectors without encountering an error then we can say that our perceptron has learnt the target concept. Now, after training, if we give any random input vector or a query point from test dataset then it should give the correct output of class label.

What is the perceptron actually doing? It is basically adding all the inputs and then dividing them into just 2 categories, one that causes it to fire and the other which doesn't. So, it can be thought of as drawing a line (equation shown below) and separating points on either side of the line. Therefore, the entire data is divided into just 2 categories.

w1x1 + w2x2 = t, where t is the threshold.

The weight and thresholds can be anything, that is just a line across 2 dimensional input space.

Along with the benefits of perceptrons, they too have some disadvantages and failure cases. The input points that can separated using a line are called linearly separable. If the points or vectors are not linearly separable then learning of perceptrons will never reach a point where it can classify all the vectors correctly.The most famous problem on which this linear separation doesn't work is the boolean XOR problem. These are also known as non-separable vectors.

We can even train out neural network in multiple layers[31]. This is the most common and popular method. In this case (as shown in diagram on next page), each input from the given input layer is fed into each node of hidden layer. There may be multiple hidden layers and any number of nodes per layer. The output of last hidden layer is fed as input to the output layer. By using our learning algorithm, we should be able to tune not only the weights between output layer and hidden layer but also between input layer and hidden layer.

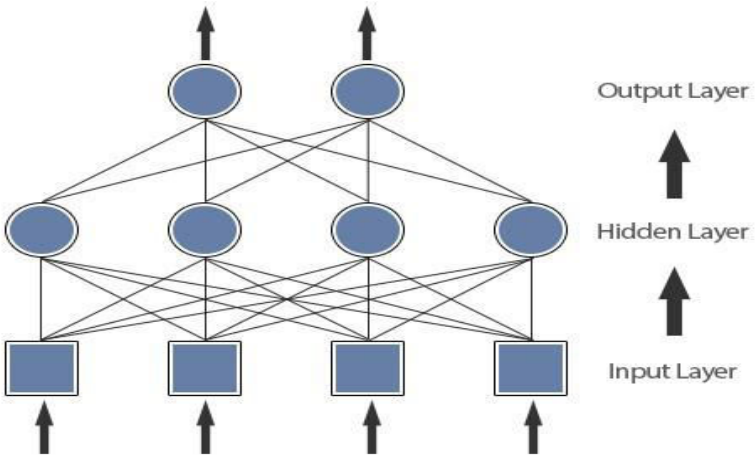


Fig 4.8: A multi-layered neural network

Convolutional Neural Networks operate over volume. Unlike the neural networks, in which the input is in the form of a vector, in this case input is multi-channeled image. The figure on next page shows an example of RGB image (which is assumed to be an input image).

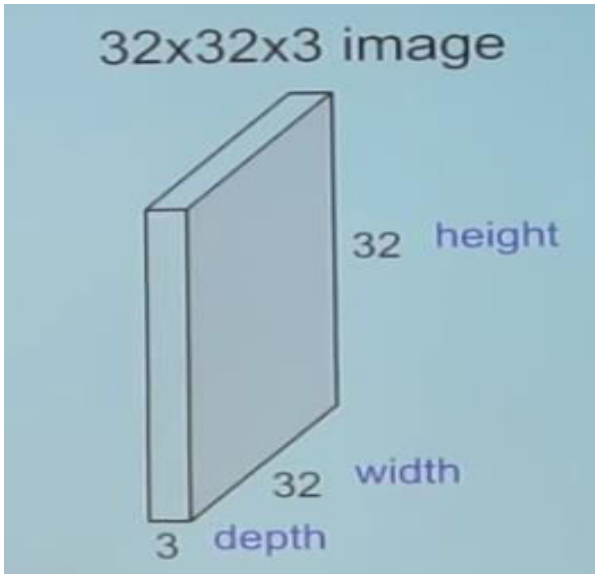


Fig 4.9: Example of a RGB image (input image)

We will now take a 5*5*3 filter and slide it over the complete image. Along the way, keep calculating the dot product between the filter and chunks of the input image.

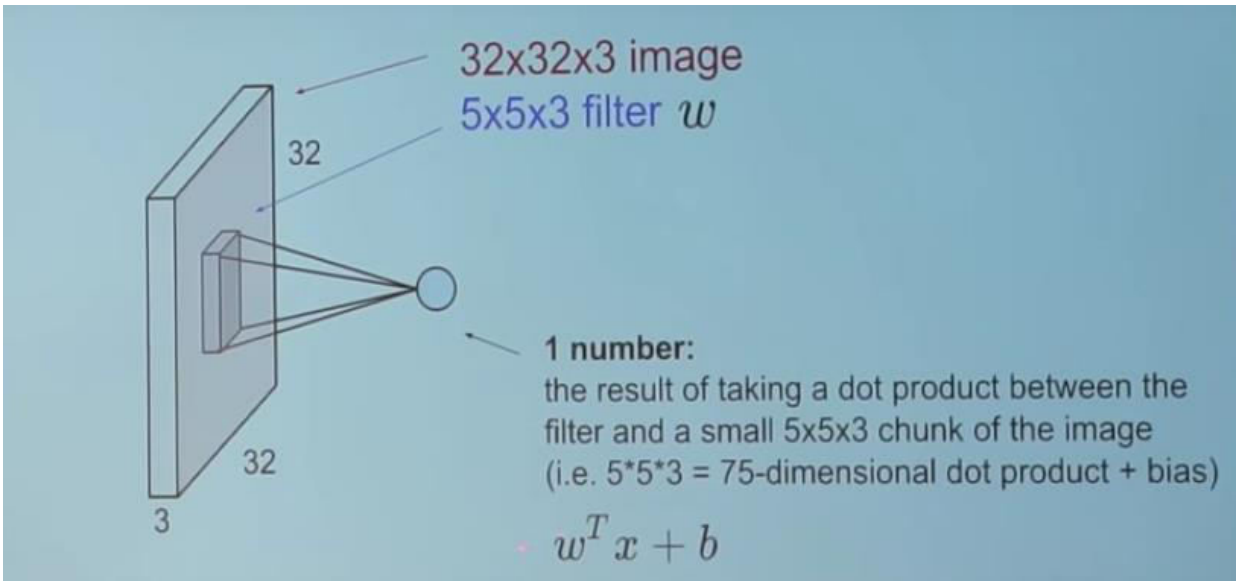


Fig 4.10: This is how it looks

For every dot product that we take, the result will always be a scalar value. Now, the figure on the next page shows the conculution of a 5*5*3 filter over 32*32*3 image. The outcome will be 28*28*1 because we will have only 28*28 unique postitions when we slide 5*5*3 filter over 32*32*3 image.

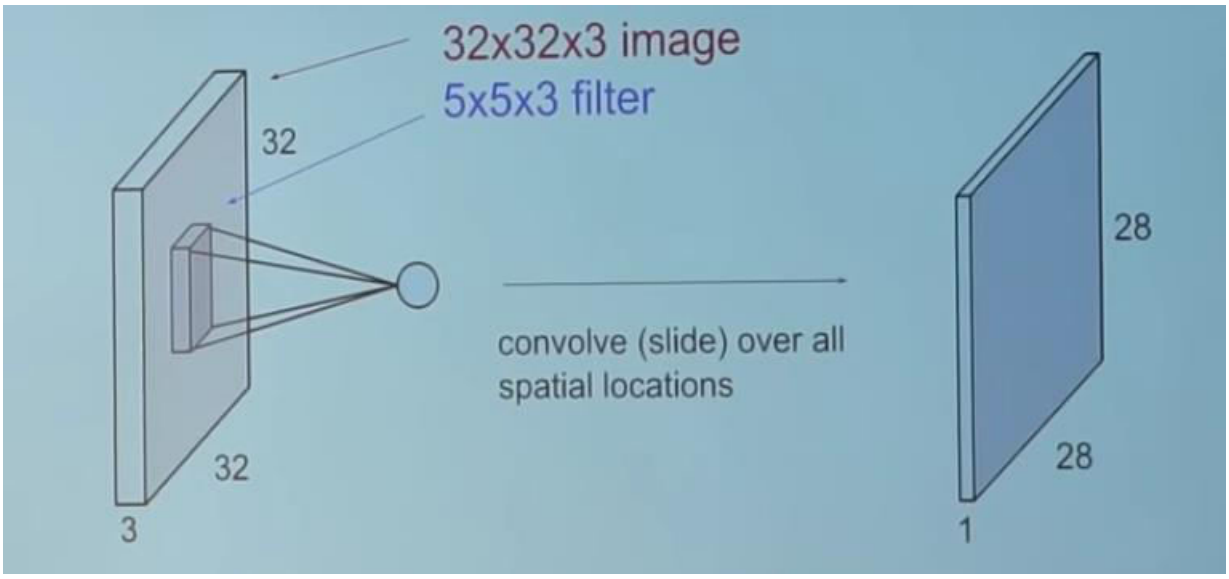


Fig 4.11: Resultant figure

Convolution layer is the basic and main building block of convolutional neural network. The convolution layer comprises of a set of many independent filters (6 in the figure below).

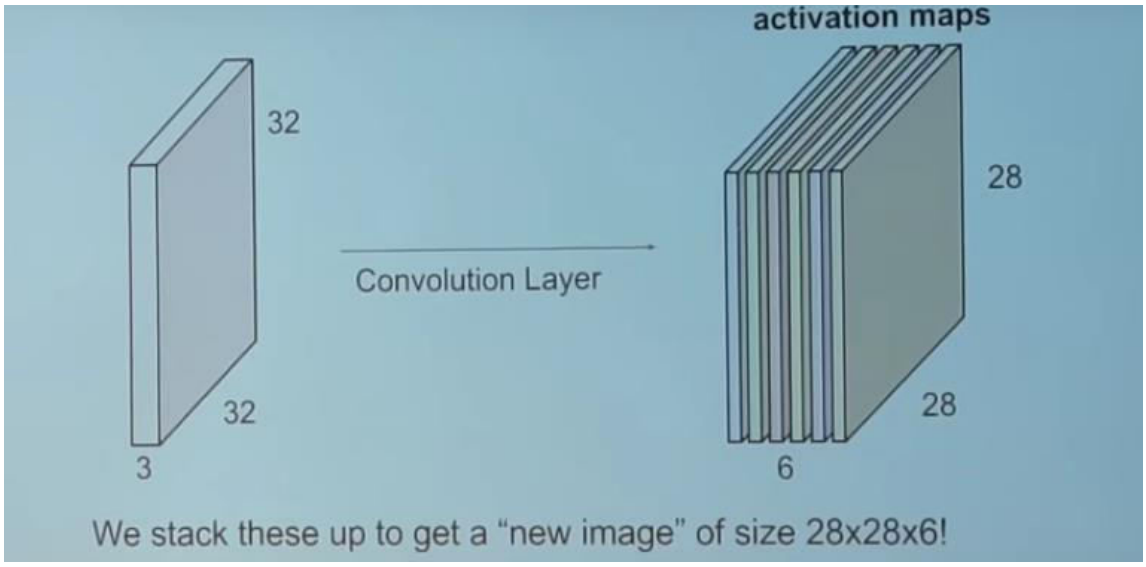


Fig 4.12: Convolution layer

Each of these filters is independently convolved with the given image and therefore, we end up with 6 feature maps of shape 28*28*1. If we have a number of convolution layers present in a sequence then what happens?

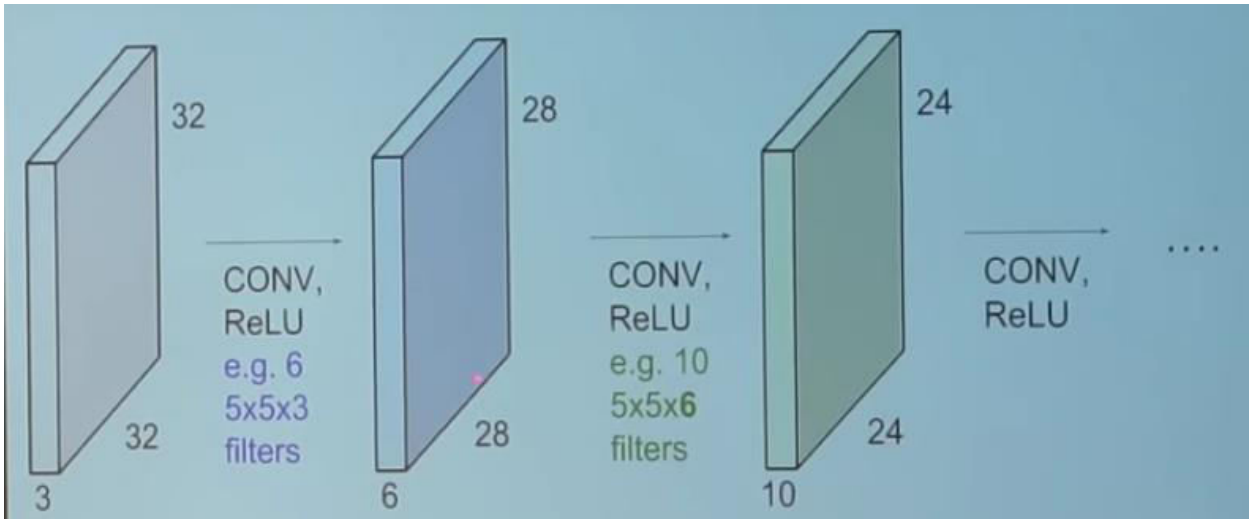


Fig 4.13: Convolution layers in sequence

These filters are randomly initialized and they become our parameters which will be learned by the network subsequently.

Let us now look at aan example of a trained network.

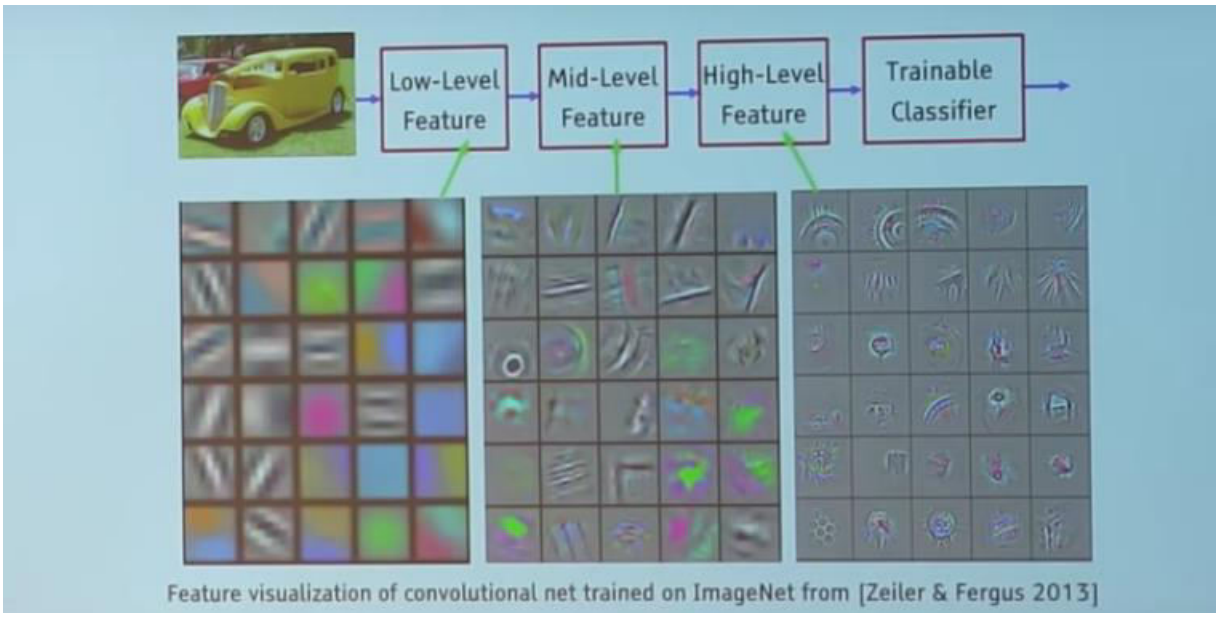


Fig 4.14: Filters in a trained network

Let us have a look at the filters in the very first layer (these filters are of 5*5*3 dimension). They have tuned themselves by using back propagation and have become blobs of colored pieces and edges. As we move deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers. So, they are taking the smaller coloured pieces or edges and making larger pieces out of them.

CNNs have a couple of concepts called parameter sharing and local connectivity. Parameter sharing is nothing but sharing of weights by all neurons in a particular feature map. Local connectivity is the concept of each neural connected only to a subset of the input image (unlike a neural network where all the neurons are fully connected). This makes the computation process more efficient by reducing the number of parameters in the entire system.

A pooling layer is another building block of a convolutional neural network (CNN).

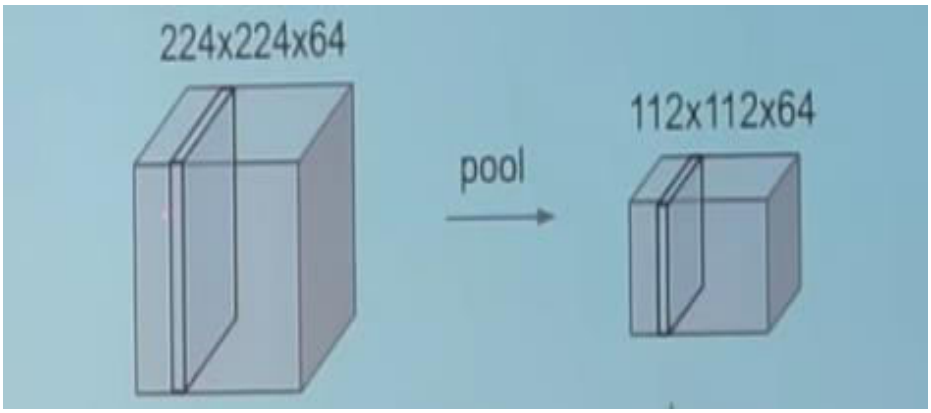


Fig 4.15: Pooling

The purpose of pooling is to progressively reduce the spatial size of the representation in order to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature layer independently. The most common approach which is used in pooling is max pooling.

In max pooling, we scan the 2*2 area (in our case as shown in figure) and take the maximum value in a cell from that area and fill it in a single cell representation. The objective behind doing this is to down-sample an input representation (it may be an image, hidden-layer output matrix etcetra), which reduces the dimensionality which in turn allows for assumptions to be made about features contained in the sub-regions binned.

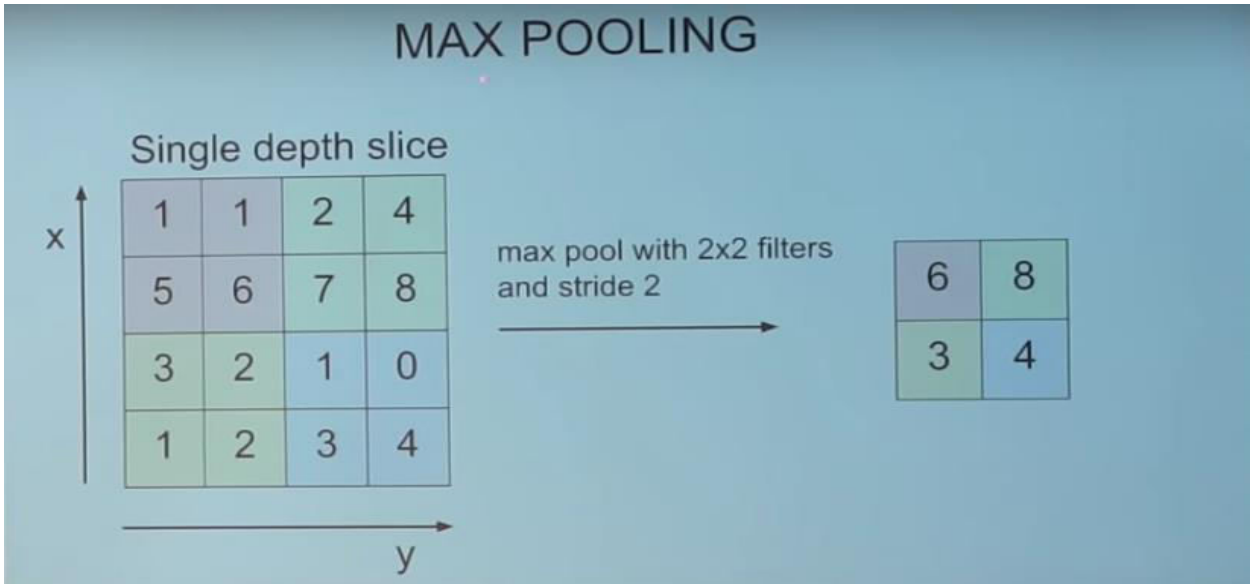


Fig 4.16: Max Pooling

The benefits of max pooling is concentrated in the following area:-

- It helps with the over-fitting problem by providing an abstracted form of representation.
- It reduces computational cost as it reduces the number of parameters to learn and it also provides a basic translation invariance to the internal representation.

Max pooling is done by applying a max filter to non-overlapping sub-regions of the initial representation.

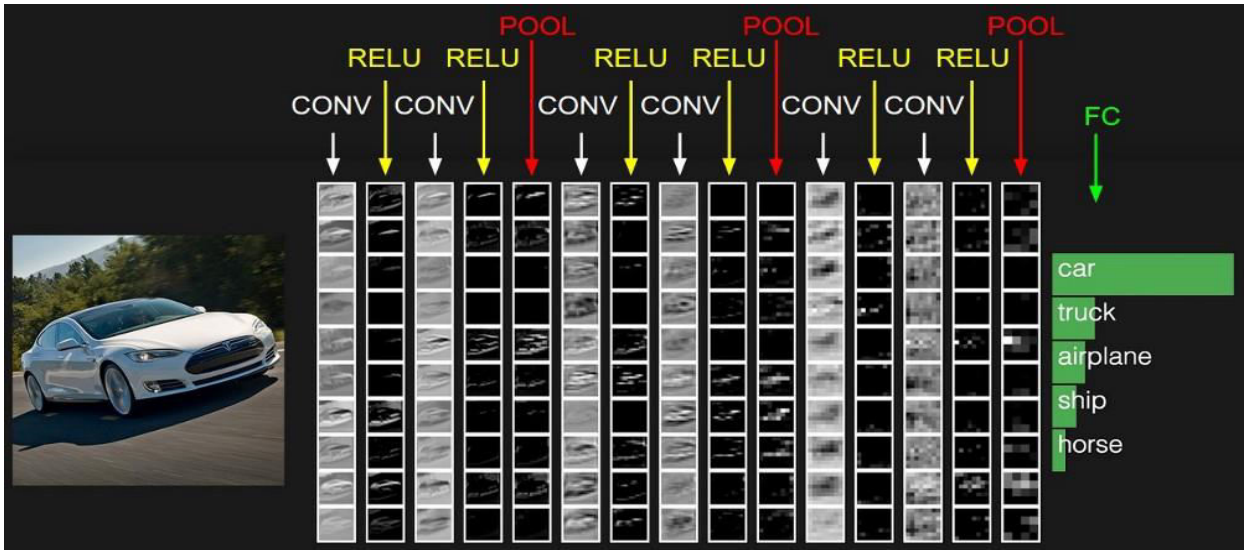


Fig 4.17: Typical architecture of CNN

Rectified Linear Unit (ReLU) is basically a non linearity which is applied to neural networks. FC means the fully connected layer of neurons at the end of CNN. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks and work in a similar way.

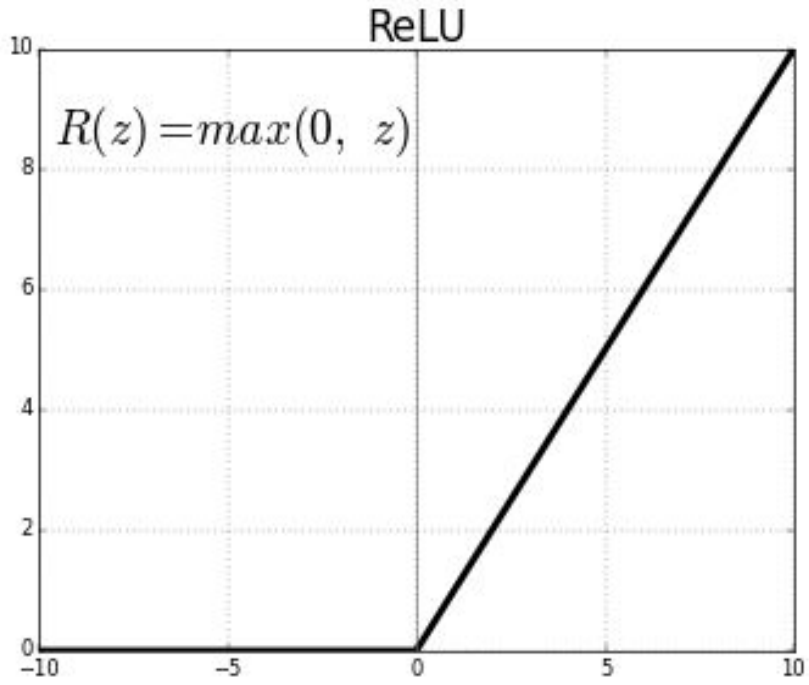


Fig 4.18: ReLU function

Our brain receives a huge amount of data out of which it separates the important data and unimportant data. In the similar way, neural networks also need some mechanism which can segregate the important and useful information from the non-useful information. This segregation is done by activation functions in machine learning. Therefore, activation functions can be thought of like a filter of data. They basically decide whether a neuron should be activated or not. Whether the information that the neuron is receiving is relevant for the given information or should it be ignored.

Some of the activation functions used in CNN[32] are as follows:-

- Linear activation function
- Nonlinear activation function
  a) Sigmoid or Logistic Activation function.
  b) Tanh or hyperbolic tangent activation function
  c) ReLU (Rectified linear unit) activation function

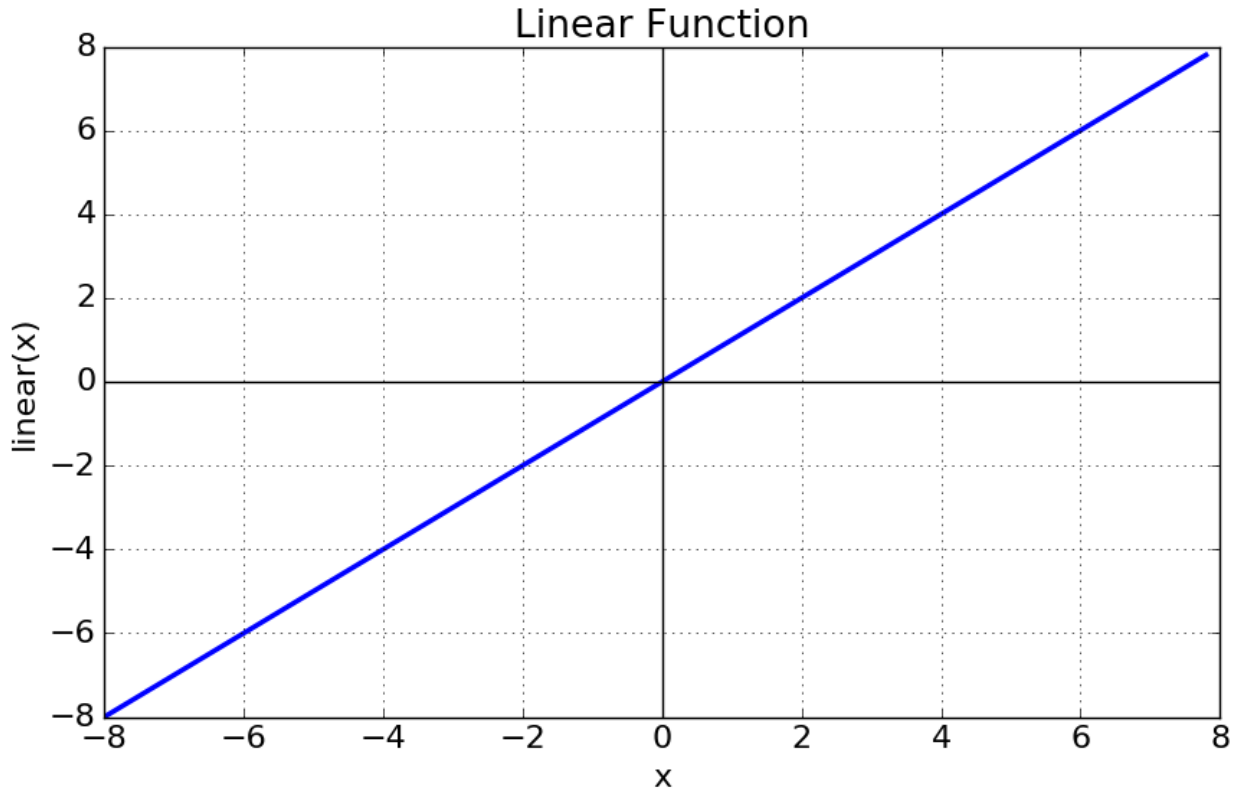


Fig 4.19: Linear activation function

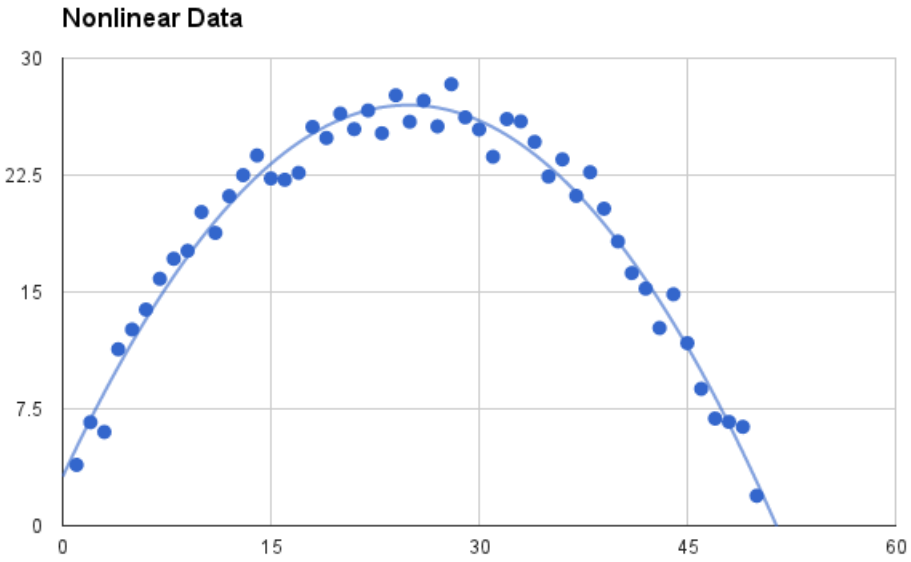


Fig 4.20: Nonlinear activation function

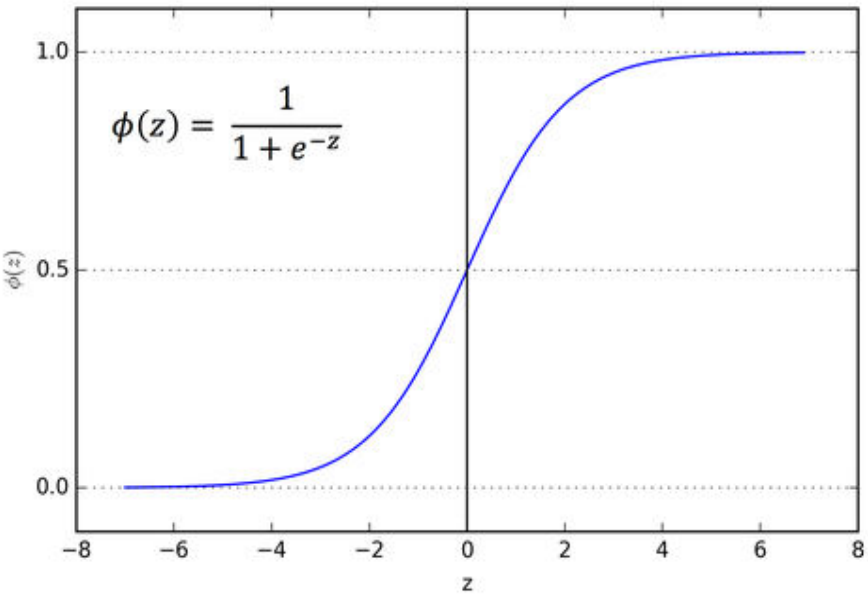


Fig 4.21: Sigmoid or Logistic Activation function.

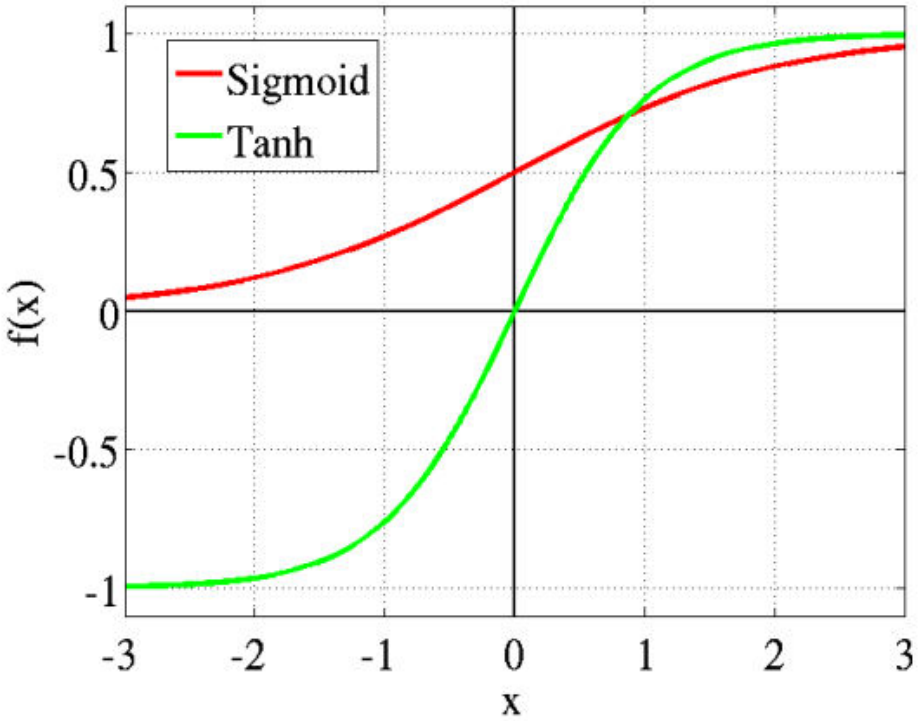


Fig 4.22: Tanh or hyperbolic tangent activation function

Advantage of CNN:-

- Very high accuracy.

Disadvantage of CNN:-

- High computational cost.
- Very slow to train (due to high computations).
- Huge amount of data is needed for training the CNN model.

Convolutional neural network (CNN) is one of the most widely used deep learning technique for machine learning purpose throughout the globe.

## 4.3 Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. Transfer learning and domain adaptation refer to the situation where what has been learned in one setting, is exploited to improve generalization in another setting. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. It is currently very popular in the field of Deep Learning because it enables us to train Deep Neural Networks with comparatively little data. This is very useful since most real-world problems typically do not have millions of labeled data points to train such complex models.

In our experiment, we have taken three types of methods :-

- VGG-16
- ResNet 50
- Inception-V3

These all are pre-trained on ImageNet dataset. We are using this model and then training it further (fine-tuning) using our own dataset in order take the benefits of deep neural network. The picture on next page shows an old classifier which was built on a dataset was again fine-tuned using some other dataset in order to form a new classifier. This new classifier is built with the intention of taking the benefits of deep neural networks as well as taking the benefit of the model

being pre-trained on old dataset of related data so as to perform better on a given specific condition.

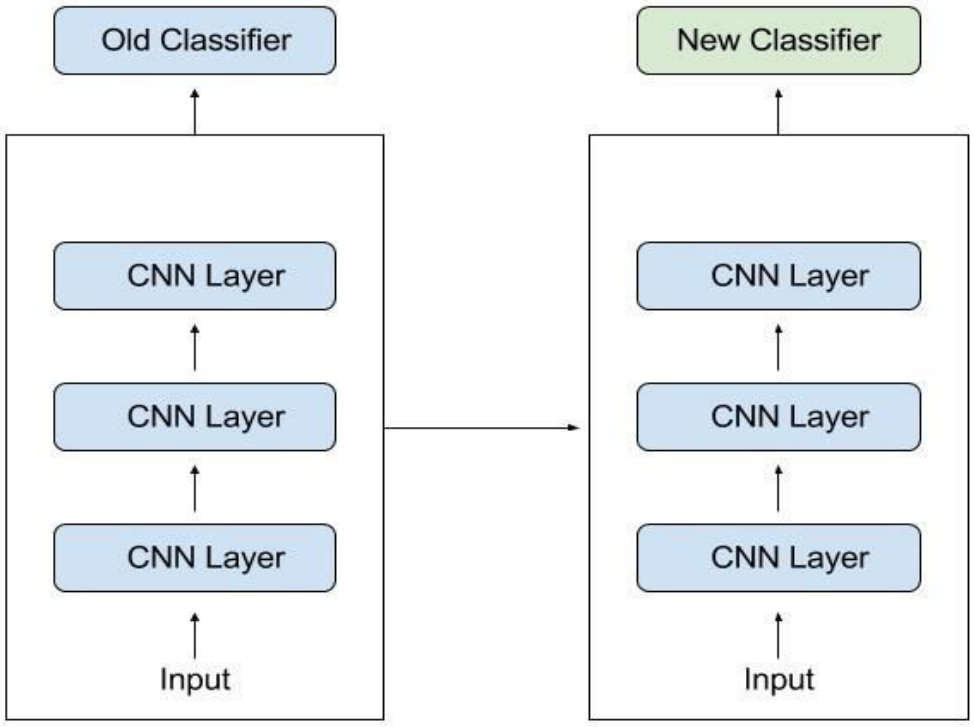


Fig 4.23: Old-classifier is trained/fine-tuned using new dataset to form new classifier

### 4.3.1 VGG-16

VGG16 [33] is a convolutional neural network (CNN) model which was proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". This model achieves 92.7% top-5 test accuracy in ImageNet (it is a dataset containing more than 14 million images belonging to 1000 classes). It was one of the famous model submitted to ILSVRC-2014. It replaces large kernel-sized filters of AlexNet and hence makes an improvement over AlexNet (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

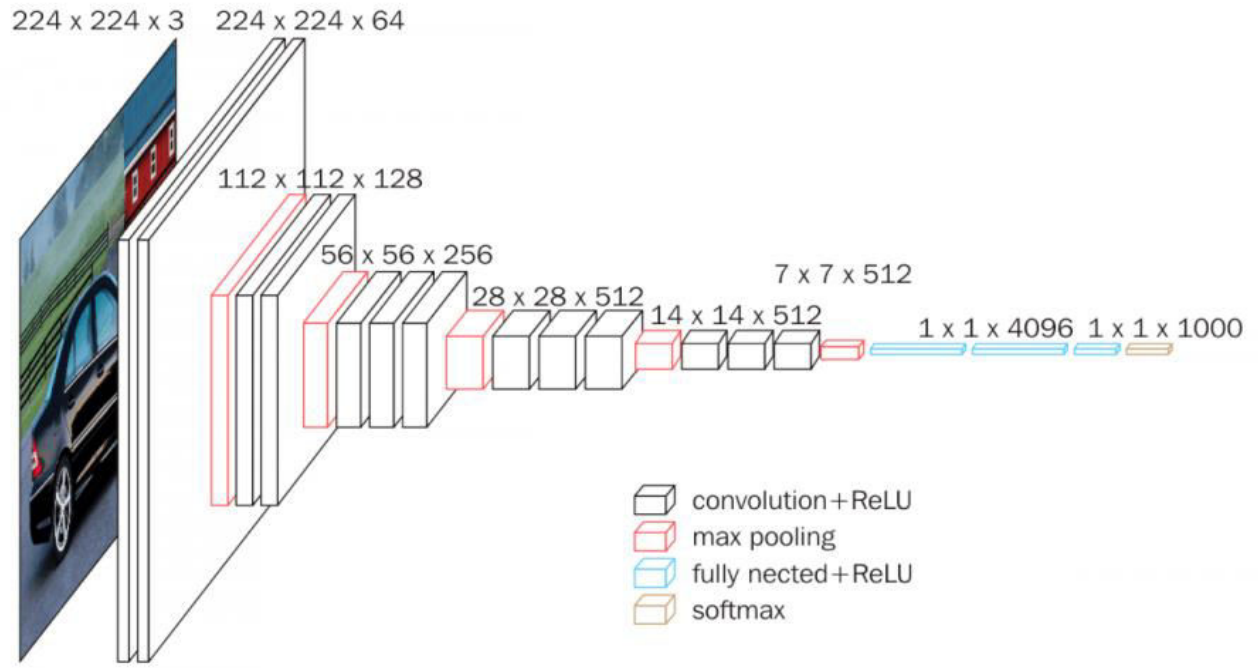


Fig 4.24: VGG-16 architecture

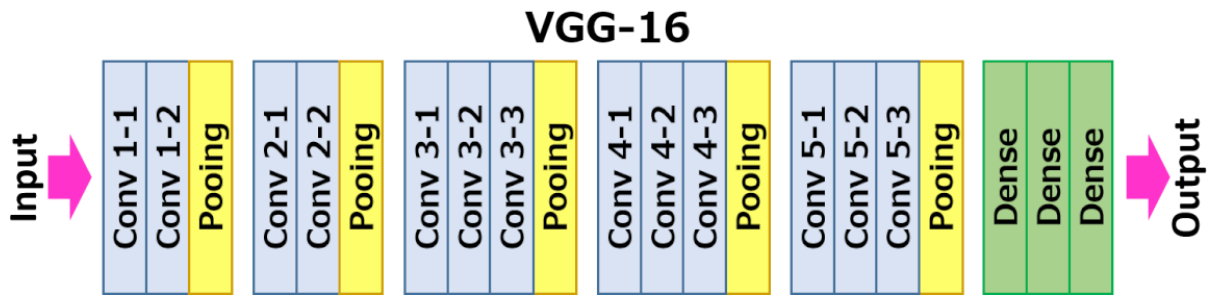


Fig 4.25: VGG-16 layered architecture

The conv1 layer gets an input of fixed size which is an RGB image of size 224 x 224. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center).In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with

stride 2.

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

There are two major drawbacks of VGGNet:-

- It is very slow to train.
- The network architecture weights themselves are quite large.

Due to the depth and number of fully-connected nodes, VGG is tiresome to deploy. VGG-16 outperforms the previous generation models by a significant margin. The VGG16 result is also competing for the classification task winner (GoogLeNet with 6.7% error) and substantially outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2% with external training data and 11.7% without it.

### 4.3.2 Resnt50

ResNet is a short name for Residual Network. As the name of the network indicates, the new terminology that this network introduces is residual learning.

What is the need for Residual Learning? Deep convolutional neural networks (DCNNs) have led to a series of breakthroughs for image classification. Many other visual recognition tasks have also greatly benefited from very deep models. So, over the years there is a trend to go more deeper, to solve more complex tasks and to also increase or improve the classification or recognition accuracy. But, as we go deeper; the training of neural network becomes difficult and also the accuracy starts saturating and then degrades also. Residual Learning tries to solve both these problems.

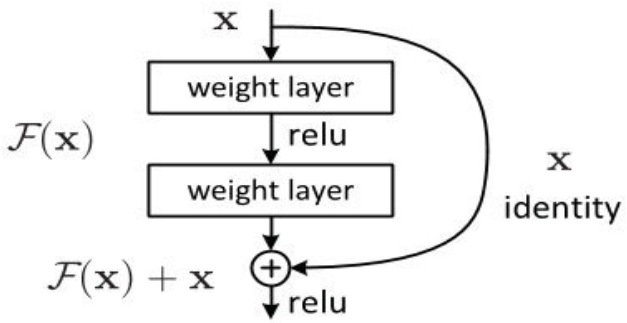


Fig 4.27: Residual learning: a building block

What is Residual Learning? In general, in a deep convolutional neural network, several layers are stacked and are trained to the task at hand. The network learns several low/mid/high level features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn some residual.

Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some $(n+x)^{th}$ layer. It has proved that training this form of networks is easier than training simple deep convolutional neural networks and also the problem of degrading accuracy is resolved. This is the fundamental concept of ResNet. ResNet50 is a 50 layer Residual Network. There are other variants like ResNet101 and ResNet152 also.

Seeing all the above mentioned points, can we say ResNet is successful? ResNet won the $1^{st}$ place in the ILSVRC 2015 classification competition with top-5 error rate of just 3.57% using an ensemble model. ResNet also won the 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation. On replacing VGG-16 layers in Faster R-CNN with ResNet-101, researchers observed a relative improvements of 28%. ResNet can have efficiently trained networks with 100 layers and 1000 layers as well.

ResNet faces a problem when deeper networks starts converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly.

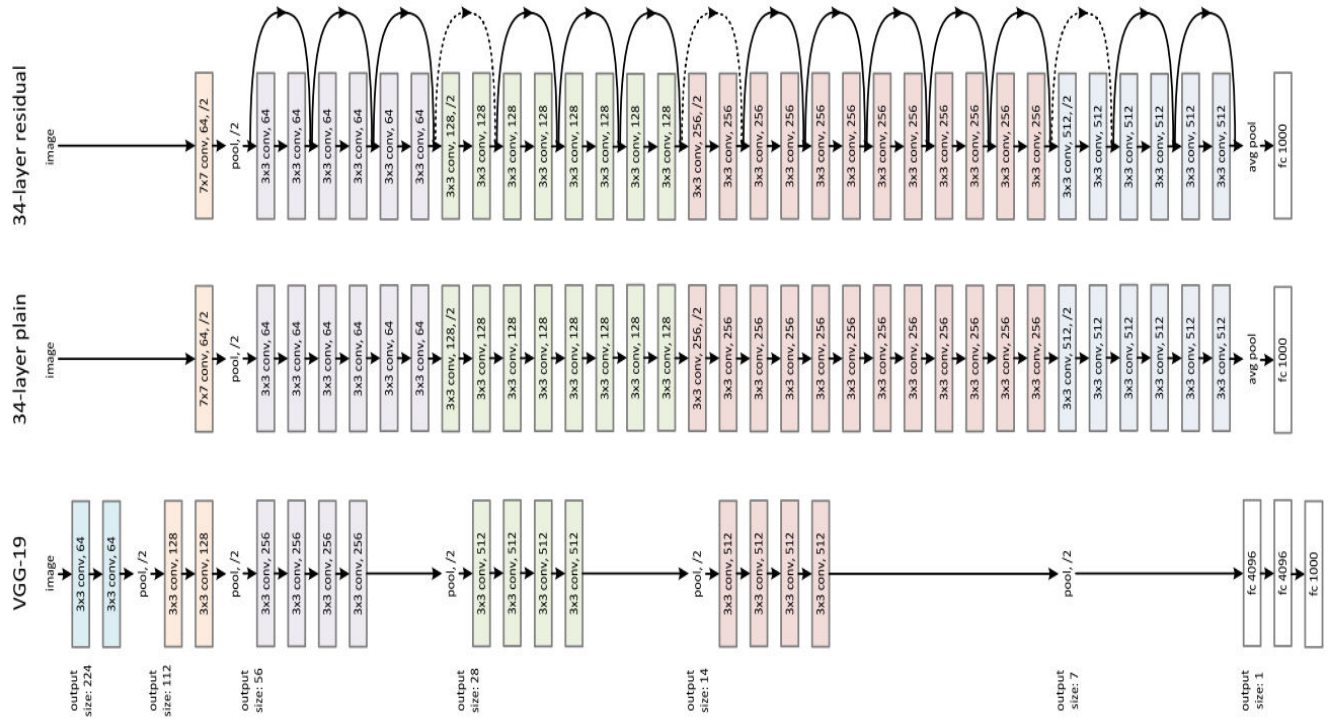


Fig 4.28: ResNet-50 architecture

### 3.3.3 Inception-v3

There are 4 versions of Inception. Inception-v3 [34] is one of them. The Inception deep convolutional architecture was introduced as GoogLeNet in the paper (Szegedy et al. 2015a), here named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Inception-v2) and later by additional factorization ideas in the third iteration which is referred to as Inception-v3.

Inception-v3 with 42 layers deep learning network exhibits lower error rate made it become the 1st runner up for image classification in ILSVRC 2015. Inception-v3 [35] is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset.

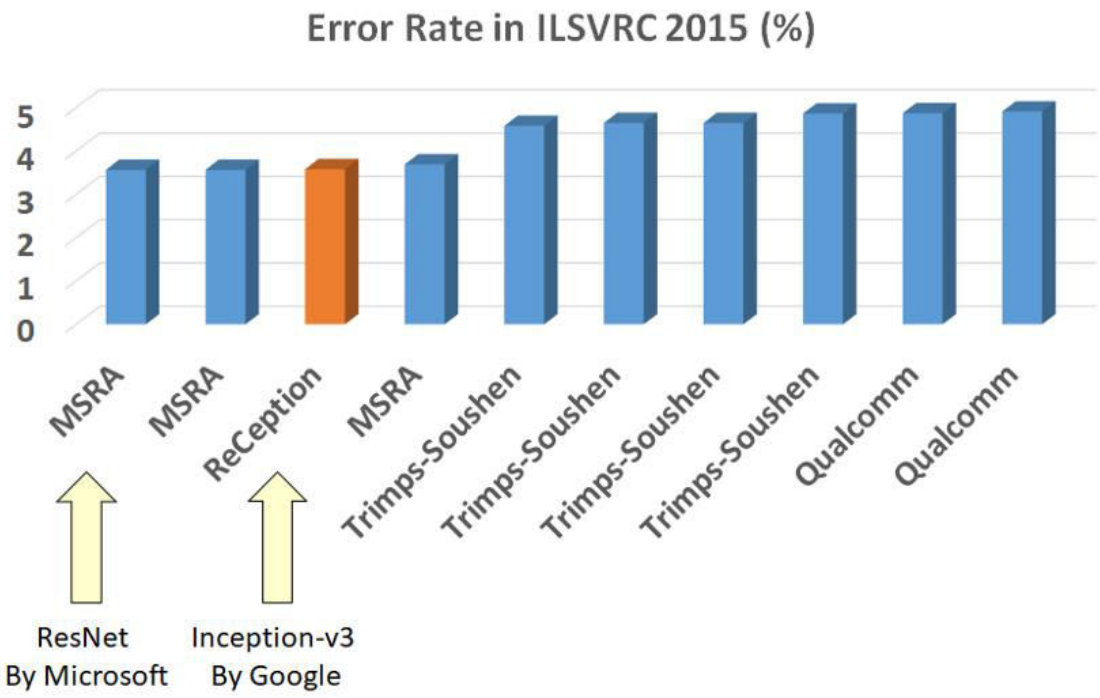


Fig 4.29: Comaprision of error rate in ILSVRC 2015 for ResNet and Inception-v3

The model is the pinnacle of many ideas developed by multiple researchers over the years. The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

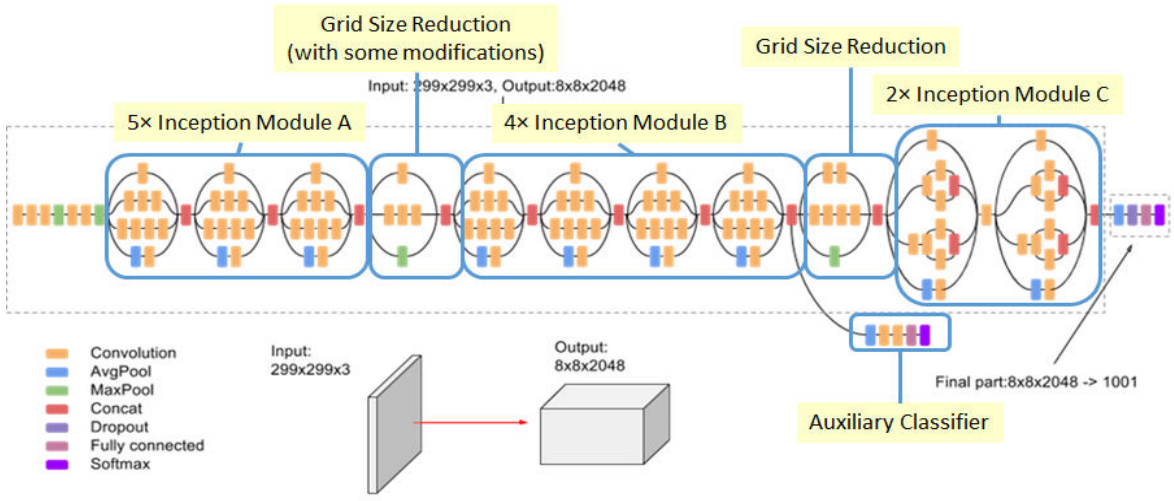


Fig 4.30: Inception-v3 architecture

# Chapter-5

# Experimental results and discussions

## 5.1 Data Overview:

**Car dataset:** This dataset [42] contains a total of 16,185 images of different companies, model and year of manufacture of car. These images are distributed into 196 classes of cars. Classes are typically at the level of "make, model, year" (example, "BMW, M3 Coupe, 2012). The entire dataset is approximately divided into a 50-50 split. The train dataset has 8,144 images while the test dataset has 8,041 images.

Table 5.1: Car dataset

| | |
|---|---|
| Total number of images | 16,185 |
| Number of classes | 196 |
| Number of train dataset images | 8,144 |
| Number of test dataset images | 8,041 |

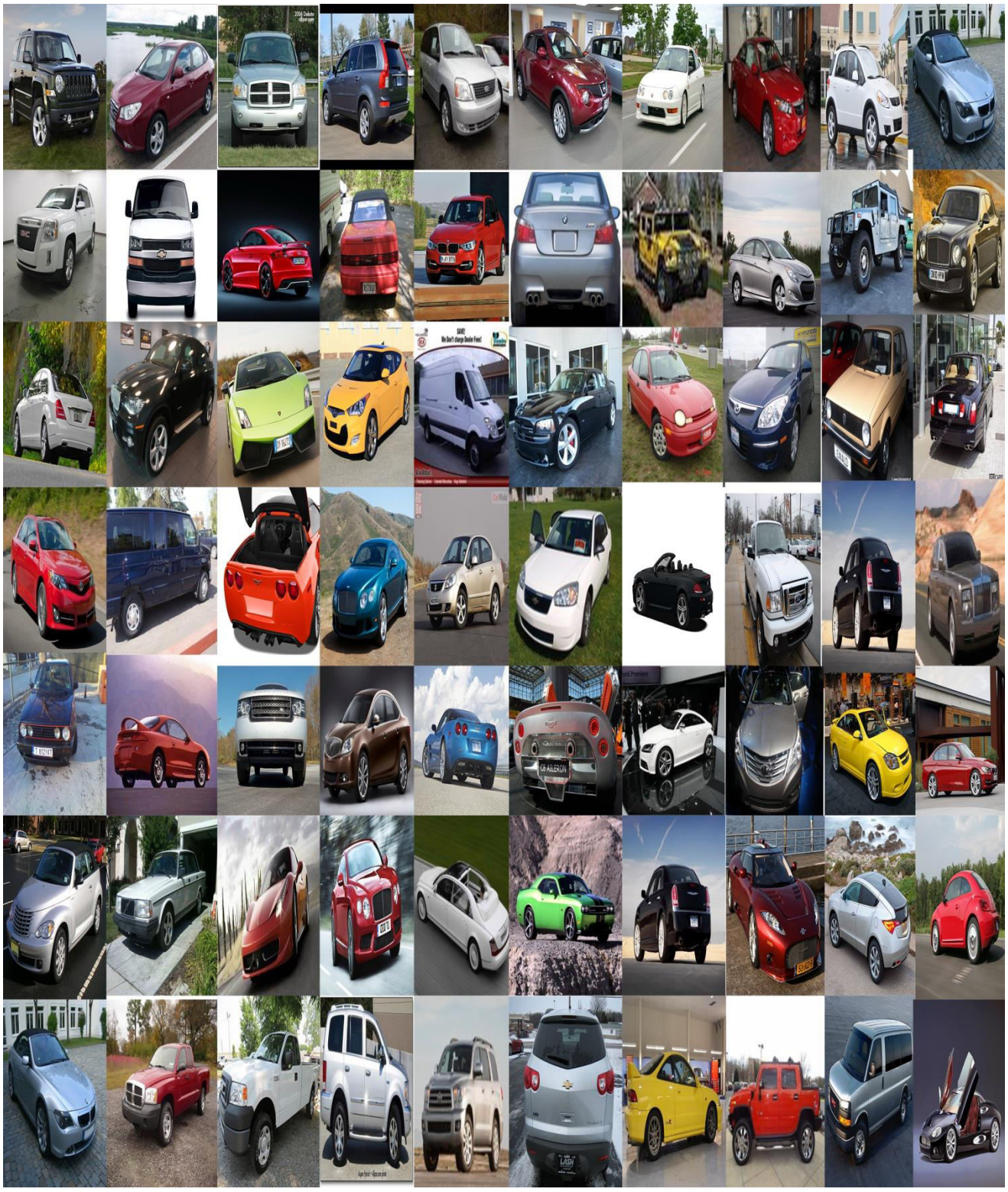

Fig 5.1: Car-196 dataset sample

**Special BMW-10 dataset:** This is a smaller dataset [42] having a total of 512 images of BMW cars only. This dataset has 10 different classes of BMW cars. The entire dataset is divided approximately into a 50-50 split where train data has a total of 254 images while test data has a total of 258 images.

Table 5.2: Special BMW-10 dataset

| | |
|---|---|
| Total number of images | 512 |
| Number of classes | 10 |
| Number of train dataset images | 254 |
| Number of test dataset images | 258 |

Fig 5.2: Special BMW-10 dataset sample

## 5.2 Problem definition

We are given a dataset of images of cars belonging to different classes. We split the dataset into train as well as test dataset dataset using 50-50 split. Now, we have trained out model on training dataset and then we have tried to predict the class lablels of the car images from testing dataset. We then calculated the prediction accuracy and hence defined the performance of our model on our given dataset. This is a typical image classification problem.

## 5.3 Proposed methods and preprocessing

The handcrafted methods which we applied are LBP, SIFT and HOG. For applying the handcrafted features on our dataset, we have first cropped the images using annotation points available and then we have converted the images to gray images. Cropping the images using annotation points helped us focus only on the area of interest in the images.

In order to run VGG16 and Resnet50 on our dataset, the images had to be resized to 224*224*3 (height, width, channels) shape. We had to reshape the images to 299*299*3 (height, width, channels) for running InceptionV3 on our dataset.

## 5.4 Experimental setup

**Hardware System Configuration:**

Processor: Intel(R) Xeon(R) CPU E5-2609 v4

Speed: CPU 1.70 GHz

Number of processors: 2

Number of cores: 8

RAM: 128 GB

Hard Disk: 500 GB (SSD)

**Software System Configuration:**

Operating System: Windows 10 Pro

System Type: 64-bit Operating System

Processor: x64-based processor

Numerical Computing Environment: MATLAB, Python

Version: MATLAB R2017a, PYTHON 3.5.6

## 5.5 Result and discussion

Table 5.3: Result for handcrafted features of car dataset

| **Feature** | **Accuracy** |
|---|---|
| SIFT | 31.00 % |
| HOG | 39.95 % |
| LBP | 32.12 % |

Table 5.4: Result for handcrafted features on special BMW-10 dataset

| **Feature** | **Accuracy** |
|---|---|
| SIFT | 45.01 % |
| HOG | 42.18 % |
| LBP | 44.92 % |

Table 5.5: Result for CNN features on CAR-196 dataset

| **Model** | **Accuracy** |
|---|---|
| VGG16 | 92.12 % |
| Resnet50 | 93.21 % |
| InceptionV3 | 93.33 % |

Table 5.6: Result for CNN features on special BMW-10 dataset

| **Model** | **Accuracy** |
|---|---|
| VGG16 | 85.12 % |
| Resnet50 | 91.20 % |
| InceptionV3 | 91.24% |

The result obtained using handcrafted features are far less accurate as compared to the CNN models. This is expected as the handcrafted features are using only the provided trained dataset

and then trying to predict the resulting class labels on the test dataset images. The training images are very low in number and so the result too will not be as good as our own expectations. This dataset is more challenging because of the fact that we are actually identifying between similar looking cars, which one belongs to what company and model. The problem would have been much simpler if were required to classify if the given image is of a car. But we have different set of problem altogether, which is challenging as well. Therefore the result is around 91 % .

If we apply CNN model directly on our dataset then since our dataset has very low number of images, we will not be harnessing the benefits of deep neural networks. In order to take the maximum advantage of the CNN algorithms, the CNN models (VGG16, Resnet50, InceptionV3) are pre-trained using the ImageNet dataset (which has a very large collection of 1000 classes of images). After this, we are training these pre-trained models with our own training dataset. By doing this, we are applying fine-tuning strategy and trying to reap the benefits of transfer learning. The pre-trained models alongwith the training dataset creates a deep neural network and hence enhances the classification accuracy to a great extent.

We have used the softmax layer as last layer in our classification task using CNN models (deepl learning). This softmax gives a probability distribution of a given image over all possible classes and then selects the predicted class as the one having the highest probability score (max voting technique). The softmax is very commonly used for multi-class classification problem.

# Chapter-7

# Conclusion and future scope

Image classification has a wide range of applications in current age and so it becomes a very important problem to solve and improve upon. We tried to solve this problem using a larger car dataset having 196 classes and a smaller special BMW-10 dataset. We first used the handcrafted features to classify our test images and later applied the CNN models on the given dataset. The results shows the power of the more recent deep learning algorithms. The deep learning algorithms clearly outperforms the classic handcrafted features by a huge margin. The deep learning algorithms makes use of deep neural networks and using the pre-trained CNN models on ImageNet dataset which was fine tuned using our car dataset, it did a really good job at our classification job even though it took high training time and a large number of computations. Therefore, we can conclude that CNN using pre-trained models are very good for image classification using small dataset (transfer learning).

Even though we have tried to classify our dataset using the state-of-the-art deep learning algorithms, still we don't have mind blowing results. Our accuracy is around 93% but CNN models are known to perform extremely well in producing accuracy very close to 99.99 %. This will be a tremendous achievement on the given classification problem using our dataset. The algorithm took a lot of time to train and a large number of computations as well.

In our future work, we can try to find the best fitting CNN classification algorithm which will greatly improve the accuracy of classification task as well as take lower number of computations in order to make this classification process faster.

# REFERENCES


[1]https://medium.com/deep-dimension/an-analysis-on-computer-vision-problems-6c68d56030c3

[2] Thomas, Alexander, Vittorio Ferrar, Bastian Leibe, Tinne Tuytelaars, Bernt Schiel, and Luc Van Gool. "Towards multi-view object class detection." In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 2, pp. 1589-1596. IEEE, 2006.

[3] http://www.bmva.org/visionoverview

[4] Kornblith, Simon, Jonathon Shlens, and Quoc V. Le. "Do better imagenet models transfer better?." *arXiv preprint arXiv:1805.08974* (2018).

[5] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In Proceedings of the IEEE international conference on computer vision, pp. 1026-1034. 2015.

[6] You, Yang, Igor Gitman, and Boris Ginsburg. "Scaling sgd batch size to 32k for imagenet training." arXiv preprint arXiv:1708.03888 6 (2017).

[7] He, Kaiming, Ross Girshick, and Piotr Dollár. "Rethinking imagenet pre-training." arXiv preprint arXiv:1811.08883 (2018).

[8] Simon, Marcel, Erik Rodner, and Joachim Denzler. "Imagenet pre-trained models with batch normalization." arXiv preprint arXiv:1612.01452 (2016).

[9] You, Yang, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. "100-epoch imagenet training with alexnet in 24 minutes." ArXiv e-prints (2017).

[10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[11] Hentschel, Christian, Timur Pratama Wiradarma, and Harald Sack. "Fine tuning CNNS with scarce training data—Adapting ImageNet to art epoch classification." In 2016 IEEE International Conference on Image Processing (ICIP), pp. 3693-3697. IEEE, 2016.

[12] Reyes, Angie K., Juan C. Caicedo, and Jorge E. Camargo. "Fine-tuning Deep Convolutional Networks for Plant Recognition." CLEF (Working Notes) 1391 (2015).

[13] Yanai, Keiji, and Yoshiyuki Kawano. "Food image recognition using deep convolutional network with pre-training and fine-tuning." In 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), pp. 1-6. IEEE, 2015.

[14] Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." arXiv preprint arXiv:1801.06146 (2018).

[15] Foody, Giles M., and Ajay Mathur. "Toward intelligent training of supervised image classifications: directing training data acquisition for SVM classification." Remote Sensing of Environment 93, no. 1-2 (2004): 107-117.

[16] Chapelle, Olivier, Patrick Haffner, and Vladimir N. Vapnik. "Support vector machines for histogram-based image classification." IEEE transactions on Neural Networks 10, no. 5 (1999): 1055-1064.

[17] Girshick, Ross. "Fast r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.

[18] Rothe, Rasmus, Radu Timofte, and Luc Van Gool. "Dex: Deep expectation of apparent age from a single image." In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 10-15. 2015.

[19] Akiba, Takuya, Shuji Suzuki, and Keisuke Fukuda. "Extremely large minibatch SGD: training resnet-50 on imagenet in 15 minutes." arXiv preprint arXiv:1711.04325 (2017).

[20] Xia, Xiaoling, Cui Xu, and Bing Nan. "Inception-v3 for flower classification." In 2017 2nd International Conference on Image, Vision and Computing (ICIVC), pp. 783-787. IEEE, 2017.

[21] Hassannejad, Hamid, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. "Food image recognition using very deep convolutional networks." In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management, pp. 41-49. ACM, 2016.

[22] Perez, Luis, and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning." arXiv preprint arXiv:1712.04621 (2017).

[23]https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b

[24] https://searchenterpriseai.techtarget.com/definition/facial-recognition

[25] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.

[26] Joachims, Thorsten. Making large-scale SVM learning practical. No. 1998, 28. Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1998.

[27]https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[28] Sharif Razavian, Ali, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 806-813. 2014.

[29]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[30] Yin, Wenpeng, Katharina Kann, Mo Yu, and Hinrich Schütze. "Comparative study of cnn and rnn for natural language processing." arXiv preprint arXiv:1702.01923 (2017).

[31]https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-part-2-1218d5dc043

[32] Jain, Anil K., Jianchang Mao, and K. M. Mohiuddin. "Artificial neural networks: A tutorial." Computer 3 (1996): 31-44.

[33] Zhang, Xiangyu, Jianhua Zou, Kaiming He, and Jian Sun. "Accelerating very deep convolutional networks for classification and detection." IEEE transactions on pattern analysis and machine intelligence 38, no. 10 (2015): 1943-1955.

[34]https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

[35] Xia, Xiaoling, Cui Xu, and Bing Nan. "Inception-v3 for flower classification." In 2017 2nd International Conference on Image, Vision and Computing (ICIVC), pp. 783-787. IEEE, 2017.

[36] Wang, Xiaoyu, Tony X. Han, and Shuicheng Yan. "An HOG-LBP human detector with partial occlusion handling." In 2009 IEEE 12th international conference on computer vision, pp. 32-39. IEEE, 2009.

[37] Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. "Face description with local binary patterns: Application to face recognition." IEEE Transactions on Pattern Analysis & Machine Intelligence 12 (2006): 2037-2041.

[38] https://www.learnopencv.com/histogram-of-oriented-gradients/

[39] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." In international Conference on computer vision & Pattern Recognition (CVPR'05), vol. 1, pp. 886-893. IEEE Computer Society, 2005.

[40] Mortensen, Eric N., Hongli Deng, and Linda Shapiro. "A SIFT descriptor with global context." In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 184-190. IEEE, 2005.

[41] http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

[42] Krause, Jonathan, Michael Stark, Jia Deng, and Li Fei-Fei. "3d object representations for fine-grained categorization." In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 554-561. 2013.