ACKNOWLEDGMENT

First of all, I wish to express our gratitude to the Almighty for giving me the strength to perform our responsibilities and complete the report.

The capstone project program is very helpful to bridge the gap between theoretical knowledge and real-life experience as part of the Bachelor of Science (BSc) program. This report has been designed to have a practical experience through theoretical understanding.

I also acknowledge our profound sense of gratitude to all the teachers who have been instrumental in providing us with the technical knowledge and moral support to complete the project with full understanding.

It is imperative to show our appreciation for our honorable faculty member **Dr. Mahdy Rahman Chowdhury** for his undivided attention and help to achieve this milestone. Also, our gratefulness is divine to the ***North South University***, ***ECE*** department for providing us with a course such as **CSE499** in which I could really work on this project and materialize it the way I have dreamt of.

We thank our course instructor **Aimon Rahman and Deponker Sarker Depto** who helped us in our project.

ABSTRACT

Cancer Classification using Convolution Neural Network (CNN) is a deep learning based classification process by which we will train machine using cell images and classify the cancer from it. VGG16 model is best for this classification and we use thismodel which help machine to learn from the images in order to provide better accuracy. In this research we have tried ɒ implement VGG16, VGG19, Inception V3, ResNet50, ResNet101, ResNet152, Densenet121, model on different types of datasets and tried to achieve better accuracy. We have used different techniques like data preprocessing, data augmentation, and feature normalization to achieve better result. Data preprocessing is used for proper feature selection and feature extraction. Selecting important portion and removingirrelevant portion of input features is the main aim of data preprocessing. Data Augmentation is used for all the original images were transformed and augmented every epoch and then used for training to avoid overfitting. This allowed the model to be more robust and accurate, as it was trained on different variations of the same image. This method was preferred over random oversampling, which was one way of dealing with class imbalances. Random oversampling consisted of re-sampling less frequent samples to adjust their amount in comparison with predominant samples. However, the distribution of classes would change significantly, where the smaller classes would be much less variable, and the larger classes would have richer variations. Thus, we have tried to find out how our model works on publicly available three datasetsFold1, Fold2 and Fold2 and achieve better result in our project. We tried to do classification these datasets and tried to overcome the obstacles we found during this process.

Table of Content

CHAPTER 1: Overview

In this chapter I am going to discuss about the introductory part, cancer, cancer classification and why cancer classification is needed for the betterment ofthe society. I am going to discuss my project aim, objectives and motivation for doing this project.

1.1 INTRODUCTION

Deep learning is a sub field of machine learning. DL provides excellent opportunitiesin image processing, image classification, segmentation and detection. There are many Convolutional Neural Network (CNN) which can be used for biomedical sector. A deep learning-based approach provides better result forclassification of multi-class cancer cells, so we have used many different models in this purpose.There are many researches going on in this sector to classify different cancer cells but which architecture is more affective for classification is not proven yet. It has both encoder and decoder part in the architecture. The encoder part consists of severalconvolutional and max pooling layer. This part mainly identifies the context of image by labeling each pixel. The decoder part is a fullyconnected convolutional network. It does not learn any pixel of context of the image, rather it localizes the important portion of the image. It just classifies theimportant pixels and highlight it to do classification.
In the field of medical science cancer examination is an important task for the physicians. Because of its high social impact, many approaches had done in order to help diagnosis. In the past two decades many computer-based algorithms had been proposed for specific cancer types but many of them failed to achieve the state of art. It happened because of the limitation of classification process. The handcrafted classification process is expensive as well as time consuming. But the developmentof Convolutional Neural Network (CNN) provides a brilliant way of feature learning. A CNN is a class of deep neural networks that use convolution in place of general matrix multiplication in at least one of their layers. It excels in analyzing visual imagery as they are fully connected (FC) feed forward neural networks that reduce the number of parameters very efficiently without losing out on the quality of models.

1.2 CNN

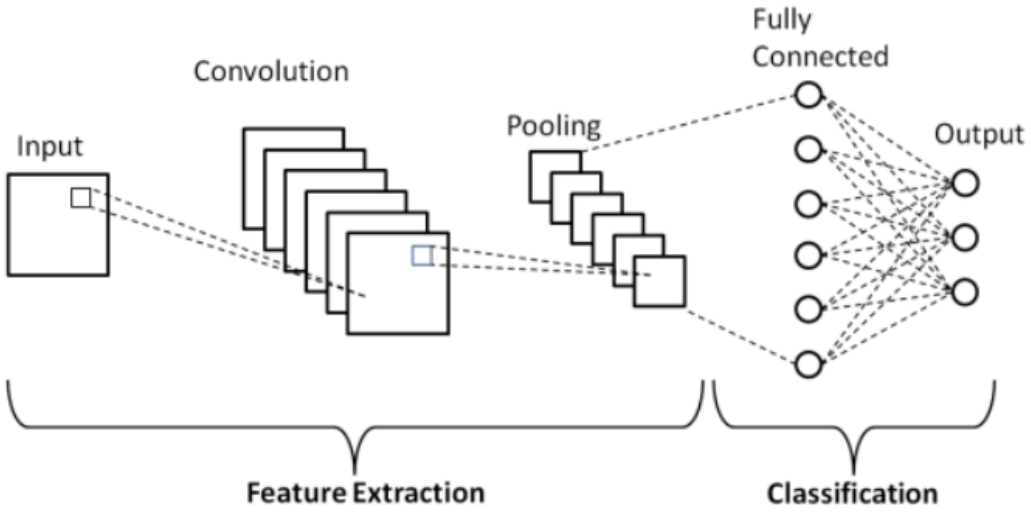
Remarkable progress has been made in image recognition, primarily due to the availability of large-scale annotated datasets and deep convolutional neural networks (CNNs). CNNs enable learning data-driven, highly representative, hierarchical image features from sufficient training data. A CNN is a class of deep neural networks that comprise of FC feed forward neural networks and utilize convolution in place of general matrix multiplication in at least one of their layers. This allows the network to reduce the number of parameters very efficiently without losing out on the quality of models, thriving in analyzing visual imagery. The network keeps on learning new higher dimensionality and more complex features with every layer. It comprises of several kinds of layers:

1) Convolutional Layer: Applies a filter that scans the whole image a couple of pixels at a time, producing a feature map for class probabilities predictions of each feature.

2) Pooling Layer: A layer that scales down the information produced from the convolution layer yet still maintains the most essential information. Different types of pooling include max pooling and average pooling.

3) FC Input Layer: Flattens previous layer's outputs into one vector for use in the next FC layers.

4) FC Output Layer Determines the image class by generating the final probabilities.



Schematic diagram of a basic convolutional neural network (CNN) architecture [26].

1.3 CANCER CLASSIFICATION

Cancer is a disease in which some of the body's cells grow uncontrollably and spread to other parts of the body. Cancer can start almost anywhere in the human body, which is made up of trillions of cells. Normally, human cells grow and multiply (through a process called cell division) to form new cells as the body needs them. When cells grow old or become damaged, they die, and new cells take their place.

Sometimes this orderly process breaks down, and abnormal or damaged cells grow and multiply when they shouldn't. These cells may form tumors, which are lumps of tissue. Tumors can be cancerous or not cancerous (benign).

Cancerous tumors spread into, or invade, nearby tissues and can travel to distant places in the body to form new tumors (a process called metastasis). Cancerous tumors may also be called malignant tumors. Many cancers form solid tumors, but cancers of the blood, such as leukemia, generally do not.

Benign tumors do not spread into, or invade, nearby tissues. When removed, benign tumors usually don't grow back, whereas cancerous tumors sometimes do. Benign tumors can sometimes be quite large, however. Some can cause serious symptoms or be life threatening, such as benign tumors in the brain.

Cancer classification is a multiclass image classification task, with 19 different classes. By primary site of origin, cancers may be of specific types like breast cancer, lung cancer, prostate cancer, liver cancer renal cell carcinoma (kidney cancer), oral cancer, brain cancer etc.

As cancer can be specified by the origin of cancer cells, so identifying the position of the cancer

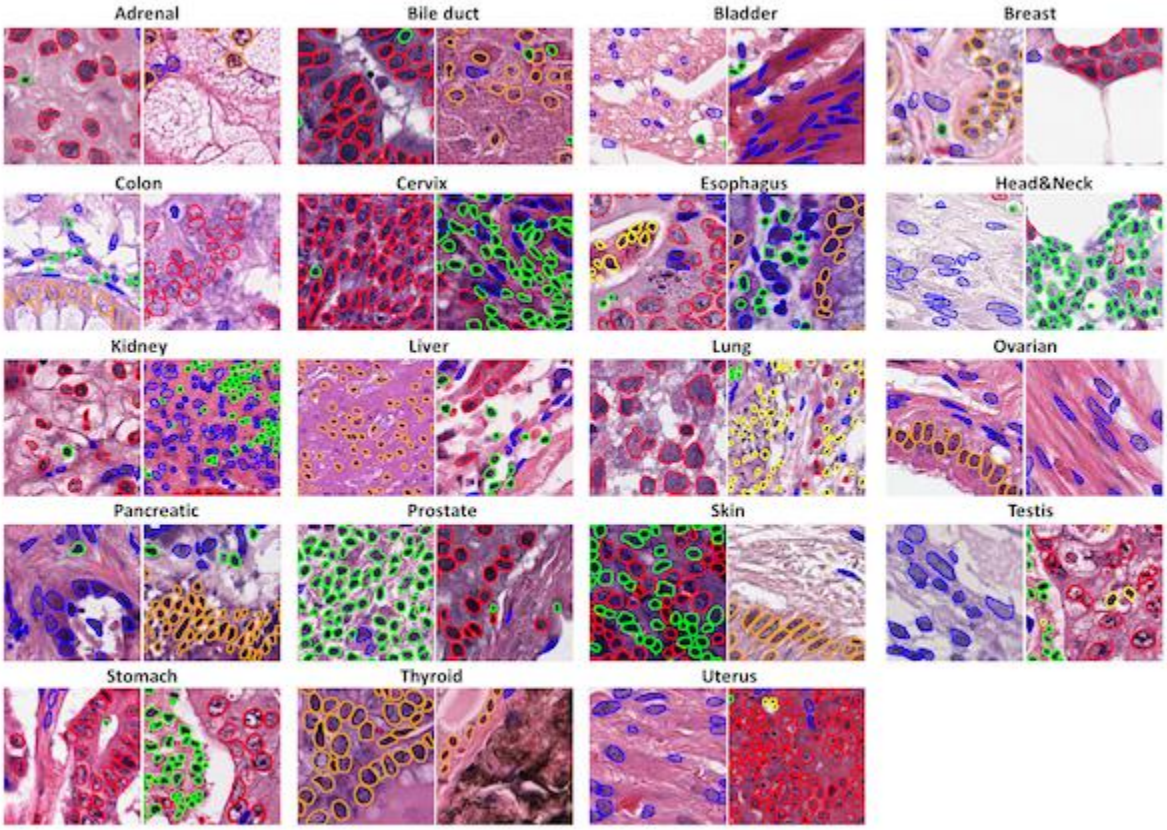
cell is important. From the image of the cell it's going to be identified, which type of cancer it is. From different types of cell images, it is going to be classified, that which type of cancer it is.

1.4 Motivation

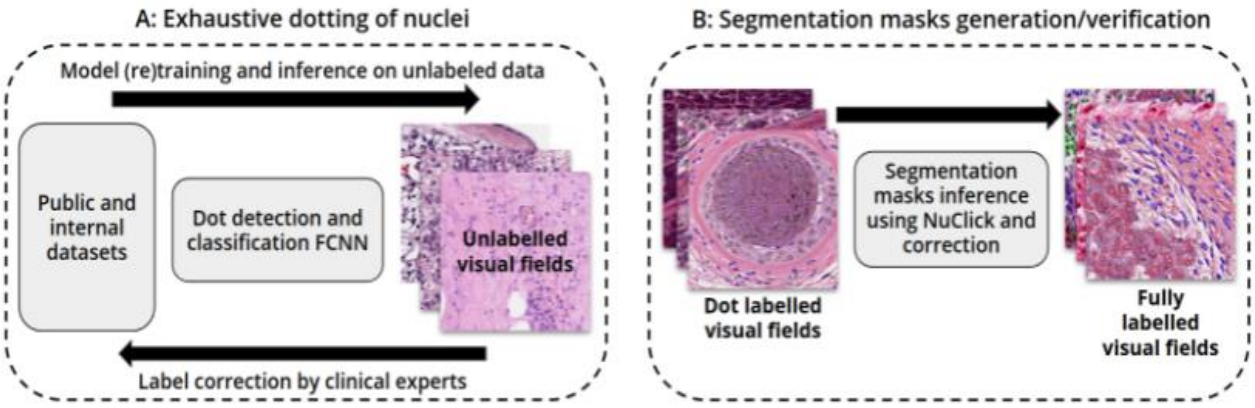
We have chosen the project considering huge social impact. Cancer is common in developed countries. People around the world are suffering from different types of Cancer like Lung, Pancreases, Breast, Skin etc. These diseases are severing and can cause permanent organ failure, even death as well if not treated. In the last two decades, a lot of algorithms for specific cancer classification have been proposed. Most of them concentrate on the cancer cell image classification, because proper characterization of the cancer cells playssignificant role in different cancer diagnostics. So, we want to do this projectto help the biomedical sector. Physicians can use the classified images of cancer cells andtreat their patients at early stage. The better result of our project brings better outcome to the society. Multi-class cancer classification using CNN will bringgreat change in biomedical sector. High accuracy and performance of this project can replace the traditional way of taking fundus image of cancer cells. This project can show the way of cancer cell image classification better than a skillful physician. Our aim of achieving higher accuracy rate and proper classification of cancer cell will help us to fulfill our goal. So, our main motivation behindthis project lies in helping physicians as well as patients so that we can help peopleof every stage of the society. The need to help people who are suffering from cancer related diseases motivated us to choose this project. We hope to finda better solution for both patients and physicians by achieving better result in this project.

CHAPTER 2: Datasets

Dataset is an important part of deep learning procedure. In machine learning datasets are collection of numbers and values, which is needed to train the model.These numbers and values are essential for the algorithms of the model. Algorithmsof the models use this collective data and train the model using it, so that the modelcan predict the result on another data which is used in test period. But, in deep learning we usually use pictures or images to train the model. In our project we the image of 3 different folder of dataset Fold1, Fold2, Fold3 to see the outcome.



The dataset we are using is PanNuke dataset. It is a semi automatically generated nuclei instance segmentation and classification dataset with exhaustive nuclei labels across 19 different tissue types. The dataset consists of 481 visual fields, of which 312 are randomly sampled from more than 20K whole slide images at different magnifications, from multiple data sources. In total the dataset contains 205,343 labeled nuclei, each with an instance segmentation mask.



Models trained on pannuke can aid in whole slide image tissue type segmentation, and generalise to new tissues. PanNuke demonstrates one of the first succesfully semi-automatically generated datasets.

In short the dataset is:

- Pannuke dataset
- Data in 3 folds
- In each fold, there are images, types, masks
- For classification, we only need images and types.
- Here, we have 19 types of cancer
- Almost 7 thousand data in total of 3 folds

CHAPTER 3: Literature Review

We searched the Google Scholar, IEEE Xplore and Web of Science databases for systematic reviews and original research articles published in English. Only papers that reported sufficient scientific proceedings are included in this review. there are many works on specific cancer classification like breast cancer, skin cancer, brain but there is no paper on combine cancer classification.

Image Processing techniques are extensively being used for the purpose. However due to limitations of Image Processing where certain parameters have to be sent manually, this approach for automated detection turns out practically to be inefficient. The need is to move towards an efficient and robust automated skin cancer classification system, which can provide highly accurate and speedy predictions. Thus it would be apt to train the machine to identify the cancer areas without using complex algorithms, which in turn increases the accuracy and efficiency of the system/algorithms.

3.1 Skin Cancer Image Classification

Here are some of the projects' literature review: used the MobileNet model from the HAM10000 dataset, and received an overall accuracy of 83.1%. Achieved a 92.9 percent accuracy with fast.ai and using ResNet34. This is 2.9% lower than the ISIC 2018 challenge obtained with a considerably more complex approach, employing a machine learning algorithm on top 18 different neural networks. With another project the accuracy was found to be 90.5% with ResNet model and 78% with VGG16 model, while with other models the accuracies were way lower around 65%. A modified VGGNet was also utilized, where the classification of melanoma versus nevi or lentigines was addressed using dermatoscopic images. The authors compared the classification accuracy of

i. a CNN trained from scratch,
ii. a pre-trained CNN with transfer learning and frozen layers,
iii. and a pre-trained CNN with transfer learning and finetuning of the weighting parameters.

All three configurations were tested with 379 images from the ISBI 2016 Challenge dataset, and the last-mentioned configuration achieved the highest accuracy of 81.33%. Mobile Net v2: 81.9%, Inception v3: 84.3%, ResNet50 v2: 81.3%, ResNet 152V2: 83.9% The top accuracy achieved on dilated versions of VGG16, VGG19, MobileNet, and InceptionV3 is 87.42%, 85.02%, 88.22%, and 89.81% respectively. 16 Dilated Inception V3 achieved the highest classification accuracy. Mobile Net also has high classification accuracy while having the lightest computational complexities.

3.2 Breast Cancer Classification

Now we are going to discuss about literature review of breast cancer image classification. It's about CNN classifier base on salient features of mammography and bit-plane to improve recognition accuracy of breast cancer image. The classifier received overall accuracy of 64% - 92%. Some paper also extracted 25 feature subgraphs from the original image. Different feature subgraphs can provide the different characteristics of the original image features. Combining two salient features to improve recognition accuracy of breast cancer image. [29]. At the same time, some paper has fully tested the integration features of different bit-plane images. [29]

In a proposed breast cancer histopathology image classification by assembling multiple compact Convolutional Neural Networks (CNNs). [30]. First, a hybrid CNN architecture is designed, which contains a global model branch and a local model branch. By local voting and two-branch information merging, a hybrid model obtains stronger representation ability. Second, by embedding the proposed

Squeeze-Excitation-Pruning (SEP) block into the hybrid model, the channel importance can be learned and the redundant channels are thus removed. The proposed channel pruning scheme can decrease the risk of overfitting and produce higher accuracy with the same model size. At last, with different data partition and composition, we build multiple models and assemble them together to further enhance the model generalization ability. [30]

3.3 Brain Tumor Image Classification

Brain tumor diagnosis and classification still rely on histopathological analysis of biopsy specimens today. The current method is invasive, time-consuming and prone to manual errors. [31]. These disadvantages show how essential it is to perform a fully automated method for multi-classification of brain tumors based on deep learning. This paper aims to make multi-classification of brain tumors for the early diagnosis purposes using convolutional neural network (CNN). Three different CNN models are proposed for three different classification tasks. [31]. Brain tumor detection is achieved with 99.33% accuracy using the first CNN model. The second CNN model can classify the brain tumor into five brain tumor types as normal, glioma, meningioma, pituitary and metastatic with an accuracy of 92.66%. The third CNN model can classify the brain tumors into three grades as Grade II, Grade III and Grade IV with an accuracy of 98.14%. All the important hyper-parameters of CNN models are automatically designated using the grid search optimization algorithm. To the best of author's knowledge, this is the first study for multi-classification of brain tumor MRI images using CNN whose almost all hyper-parameters are tuned by the grid search optimizer. [31]. The proposed CNN models are compared with other popular state-of-the-art CNN models such as AlexNet, Inceptionv3, ResNet-50, VGG-16 and GoogleNet. [31]. Satisfactory classification results are obtained using large and publicly available clinical datasets. The proposed CNN models can be employed to assist physicians and radiologists in validating their initial screening for brain tumor multi-classification purposes. [31]

3.4 Lung Cancer Image Classification

Here's some another information about lung cancer prediction using deep learning. All models based on CNN-derived features were able to perform binary classification of tumor histology (ADC vs SCC). The 4096-D feature vector seemed to correlate with marginally better predictive performance with most machine learning classifiers. The kNN model had the highest performance (AUC = 0.71, p = 0.017). [32]. This was on par with or better than the CNN (AUC = 0.71, p = 0.018). Other classifiers also showed significant predictive power, with an AUC of 0.68 (p = 0.042) for SVC with linear kernel (c = 0.1), AUC of 0.64 (p = 0.107) for non-linear SVC classifier. RF had the lowest predictive performance in all instances (AUC = 0.57, p = 0.423), although this improved to an AUC of 0.61 (p = 0.197) with the 512-D feature vector. All models had higher specificity than sensitivity, while accuracy was again highest with the kNN model. The VGG-16 based model achieved significant predictive performance differentiating between ADC and SCC on a held-out test set of 51 patients with AUC of 0.71 (p = 0.018). [32]. Similar fine-tuning and model evaluation was performed with another widely adopted ImageNet architecture, the ResNet50 network architecture49. There was no significant difference in its discriminative output and results from this analysis are included in the supplement. Discriminative performance of deep learning based radiomics models as represented by area under the ROC curve (AUC) scores. One model is tuned with a dataset containing adenocarcinoma (ADC) and squamous cell carcinoma (SCC) only, displayed an AUC of 0.71 for the 51 patient ADC vs SCC test set, and the other model was tuned with a dataset containing all histology types had AUC of 0.58 on a heterogenous test set of 83 patients (ADC vs SCC vs Other). Also shown are AUC scores for models

combining deep learning derived feature maps with machine learning classifiers. When used on a 4096-D feature vector represented by the first fully connected layer in Model A with dimensionality reduction, the kNN model had an AUC of 0.71, Linear SVM model had AUC of 0.68, SVM model had AUC of 0.64, and RF had AUC of 0.57. When used on a 512-D feature vector, the kNN model had AUC of 0.64, Linear SVM model had AUC of 0.62, SVM model had AUC of 0.63, and RF had AUC of 0.61. As a comparison, univariate logistic regression models using clinical parameters yielded AUC of 0.64 ($p = 0.118$) with smoking status, AUC of 0.55 ($p = 0.544$) with age, and sex was the strongest predictor of histology in our cohort, with an AUC of 0.69 ($p = 0.039$).

Of note, these findings are consistent with what has been described in the literature, with female and non-smoker predominance in lung adenocarcinoma of young patients. [32]. First model also demonstrated predictive value with the independent validation dataset (Lung3), achieving AUC of 0.60 ($p = 0.251$). This dataset contained a sample of 49 patients, of which 30 (61%) had SCC and 19 (39%) had ADC, which is a different skew from the BLCS fine-tuning and test sets. The median age and survival for the Lung3 group was 67.9 years and 3.34 years, respectively. [32]. Their best performing model was able to detect adenocarcinoma with higher specificity than sensitivity, suggesting greater potential in computer assisted diagnosis, and limited value as a screening tool. Furthermore, there was deterministic signal using this model to predict histology on an independent and different data set, again demonstrating the robustness of the model. The ability to non-invasively predict tumor histology has the potential to boost pathologist accuracy and productivity providing significant cost and time saving benefits.

CHAPTER 5: System Requirements

5.1 Software Requirements

5.1.1 Python: Python is a dynamically semantic, interpreted, object-oriented high- level programming language [4]. Its high-level built-in data structures, together with dynamic typing and dynamic binding, making it ideal for Rapid Application Development and as a scripting or glue language for connecting existing components [4]. We use Python version **3.7.9**

5.1.2 TensorFlow: TensorFlow Federated (TFF) is an open-source framework fordecentralized data machine learning and other computations [5]. TFF was createdto promote open study and experimentation with Federated Learning (FL), a machine learning approach in which a single global model is taught across numerous clients who maintain their training data locally [5]. FL has been used totrain prediction models for mobile keyboards without having to submit sensitive typing data to servers, for example [5].We use TensorFlow version **2.1.0**

5.1.3 GitHub: Over 65 million developers collaborate on GitHub to create the futureof software. Manage your Git repositories and contribute to the open-source community. GitHub keeps track of the numerous modifications made to each iteration of your source code projects in a variety of programming languages [6]. As a result, the term "Git" refers to a version control system, which is a tool that allows programmers to keep track of their code's constant revisions [6].

5.1.4 Jupyter Notebook: The Jupyter Notebook is an open-source web tool for creating and sharing documents with live code, equations, visualizations, and text [7]. Jupyter Notebooks are a fork of the IPython project, which used to have its ownIPython Notebook project [7]. Jupyter gets its name from the three main programming languages it supports: Julia, Python, and R [7]. The IPython kernel comes pre-installed in Jupyter, allowing you to develop Python programs [7]. Not only for studying and teaching a programming language like Python, but also for sharing your data, the Jupyter Notebook is quite handy [7]

5.1.5 Anaconda: Anaconda is an open-source distribution of the Python and R programming languages for scientific computations that is publicly available. Anaconda Navigator is a desktop graphical user interface (GUI) that comes with theAnaconda® distribution and allows you to run programs and manage conda packages, environments, and channels without having to use command-line commands [8]. Navigator can use Anaconda.org or a local Anaconda Repository to find packages [8]. This is the perfect area for us to conduct any machine learning or deep learning project. It contains a number of programs that will support us in the development of our machine learning and deep learning projects. These software's feature a stunning graphical user interface, which makes our job a lot easier. We use it to run our Python script as well.

5.2 Hardware Requirements

5.2.1 **RAM**: A minimum of 16 GB of RAM is necessary, but I recommend utilizing 32GB if possible because training any algorithm requires a lot of heavy lifting. Multitasking can be difficult if your memory is less than 16 GB. I have used 8 GB ofRAM. It takes far too long to train the model for this.

5.2.2 **CPU**: Intel Corei7 7th Generation processors are recommended since they aremore powerful and give high performance.

5.2.3 **GPU**: This is the most crucial factor because Deep Learning, a sub-field of Machine Learning, relies on neural networks to function and is computationally expensive. Working with images or videos necessitates a large number of matrix calculations. GPUs make it possible to process these matrices in parallel. Without aGPU, the operation could take days or months to complete. Your Best Laptop for Machine Learning, on the other hand, can complete the same task in hours. My GPU was GTX 1650. We alsouse the GPU of Google Colab.

5.2.4 **TPU**: Google Search, Google Street View, Google Photos, and Google Translate all use Google's accelerated neural network, commonly known as TPU. It is one of the most sophisticated deep learning training platforms available. TPU provides a 15-30x performance gain over current CPUs and GPUs, as well as a 30- 80x performance-per-watt ratio. The TPU is a 700MHz ASIC with a 12.5GB/s effective bandwidth that fits into a SATA hard drive slot and is connected to its hostthrough a PCIe Gen3X16 connection.

5.2.5 **Storage**: Because the datasets are growing in size by the day, a minimum of 1TB HDD is necessary. If you have an SSD, a minimum of 256 GB is recommended.If you have limited capacity, however, Cloud Storage Options are an option. You can even acquire machines with powerful GPUs there.

5.2.6 **Operating System:** I use Windows 10 as my operating system. Because I havebeen using it for a long time and am familiar with it. However, while Linux is the most popular operating system, Windows and MacOS can both run Virtual Linux environments and you can work on both systems as well.

CHAPTER 6: Library Details

6.1 Keras: Keras is an opensource package that works well on both the CPU and theGPU. Deep learning, specifically neural networks, makes advantage of it. Thepopular ML library works with neural network construction blocks like:

- Activation functions,
- Layers,
- Objectives, and
- Optimizers.

Keras supports convolutional and recurrent neural networks in addition to regularneural networks. The ML library also includes a slew of tools for manipulating pictures and text graphics.

Highlights

- Can run on top of:
  - Microsoft Cognitive Toolkit (CNTK)
  - PlaidML
  - R
  - TensorFlow and
  - Theano.
- Allows for rapid deep neural network research.
- It provides a high-level, understandable set of abstractions tomake deep learning model creation easier.
- Superb community support.
- Support available in TensorFlow's core library.

6.2 Matplotlib: Matplotlib is a machine learning package that uses 2D plotting to create publication ready figures, pictures, and plots in a variety of formats. The Matplotlib library allows you to create rich, high quality graphs with just a few linesof code.

- Bar charts,
- Error charts,
- Histograms,
- Scatter plots, etc.

Although Matplotlib is intuitive, users who are used to the MATLAB interface will

find it much more so, especially when utilizing the pyplot module. The ml library provides an object-oriented API for embedding graphs and plots in programsutilizing common GUI toolkits like as GTK+, Qt, and wxPython.

Highlights

- Ample documentation.
- Excellent community support.
- Functionality extension using many toolkits, including: o Cartopy
  - Excel tools
  - GTK tools and
  - Qt interface
- The greater degree of customization.
- SciPy uses Matplotlib.

6.3 NumPy: The acronym NumPy stands for Numerical Python. It's evident from the name that this is a computation-oriented library. Developers can save a lot of time in scientific computations that need a lot of matrix operations by using the Python-based library.

NumPy arrays, a unique type of array used by the NumPy library, execute massive matrix-based calculations in milliseconds. This is made feasible by the C programming language's implementation of NumPy arrays.

NumPy has become one of the most popular libraries/packages for machine learning, particularly natural language processing, as a result of theaforementioned.

Highlights

- There is a comprehensive set of high-complexity mathematical functionsavailable for processing massive multi-dimensional arrays and matrices.
- Excellent for Fourier transformations, linear algebra, andrandom numbers.

- Excellent community support.
- TensorFlow uses this to manipulate tensors on the backend.
- Out-of-the-box integration tools for C, C++, and Fortran code.

6.4 Pandas: When it comes to working with massive amounts of tabular data, pandas are the go-to machine learning library. Pandas is what Microsoft Excel is toWindows for Python. The ML library reduces the time and effort required for large,complex calculations to just a few lines of code.

In addition, pandas come with a huge list of pre-existing instructions that will save ML developers time by eliminating the need to write code for various mathematicalprocesses. Apart from data manipulation, the Pandas library also assists with data transformation and visualization. The panda's library makes use of two different kinds of data structures:

- Series (1-dimensional), and
- DataFrame (2-dimensional).

Developers may manage a wide range of data requirements and scenarios from engineering, finance, science, statistics, and other fields by using this duo in-line.

Pandas allow data scientists to focus less on creating boilerplate code and more onthe real problem-solving connected with the tabular data they're working with.

Highlights

- Able to handle any type of statistical or observational datasetwith ease:
  - Data in an arbitrary matrix with homogeneous orheterogeneous data.
  - Data in both ordered and unordered time series.
  - Tabular data with heterogeneous data columns
- Outstanding community support.
  - The library of choice for tackling real-world data analysis in Python, with exceptional efficiency in processing unequal time-series data.
- Supports data structures that are expressive, quick, and versatile, and can operate with both labeled and relational data.

6.5 Scikit-Learn: You name it, and Scikit-Learn has anything, whether it's decision trees, linear regression, logistical regression, or SVMs. It's one of the most widely used machine learning libraries for creating algorithms. Scikit-Learn can also:

- Preprocess data and
- Vectorize text using BOW, hashing vectorization, TF-IDF, andother methods.

Scikit-Learn, which is written in C and Python, has a growing and devoted community of programmers, machine learning hobbyists, and IT professionals all around the world. It is based on NumPy and SciPy, two of the most widely used machine learning frameworks for scientific computation.

The Scikit-learn library's only flaw is that it doesn't provide adequate support for distributed computing in large-scale production environments. Hopefully, it will beresolved in future releases of the famous machine learning library.

Highlights

- It has a wide range of supervised and unsupervised learning techniquesand can be used for data analysis and data mining.
- Support from the mushrooming community.
- In addition to model management and preprocessing, the MLlibrary can handle functions related to:
    - Classification,
    - Clustering,
    - Reduced dimensionality, and
    - Regression.

6.6 TensorFlow: TensorFlow is one of the most powerful deep learning packages available. The ML library, created by Google, is a quick-start alternative for product-based businesses since it provides outstanding model prototyping, production, andeverything in between.

The Tensorboard, a web-based visualization tool included with the TensorFlow framework, allows developers to see model parameters, gradients, and performance. TensorFlow Lite and TensorFlow Serving are two frameworks provided by the DL library for quickly deploying machine learning models.

Despite its many advantages, the machine learning library has been chastised for its poor graph implementation. This is due to the library's requirement that the graph is compiled first. Hopefully, with subsequent rollouts, we will see improvements in the aforementioned.

Highlights

- Google's support.
- Provides C++ and Python APIs that are extremely stable. It canalso expose APIs for other programming languages that are backward compatible. (However, these can beunstable.)
- There is a lot of documentation.
- Has a versatile architecture that lets it run on a variety of CPUs,GPUs, and TPUs (Tensor Processing Units).
- Supports a large range of toolkits for constructing ML modelsat various levels of abstraction; more than just a library, it's apopular computational framework for developing strong Machine learning models.
- A large, dependable community.

6.7 Pre-defined classes and functions

We use Keras modules in our projects that they contain pre-defined classes,functions, and variables. In this section, we'll learn about the Keras modules.

Keras.models: The real neural network model is represented by the Keras model. Keras offers two ways to build neural network models [9]. Sequential API's basic concept is to arrange Keras layers in a sequential manner [9]. The Functional API has a more versatile and powerful Functional API for the same purpose, but it's notyet available [9].

The Sequential() API can be used to generate an ANN model, as seen below [9].

```
from keras.models import Sequentialmodel =
Sequential()
```

A functional API is a way to build more complicated models [9]. You can define several inputs or outputs that share layers in a functional model [9]. To access the model's input and output, we first build an instance for it and connect it to the layers of its layers [9].

keras.optimizers: Right optimizers are Classes or methods used to change the attributes of your machine/deep learning model such as weights and learning ratein order to reduce the losses [10]. Optimizers help to get results faster [10]. There are many optimizers' algorithms we have in PyTorch and TensorFlow library [10].

TensorFlow primarily supports nine optimizer classes, including Adadelta, FTRL, NAdam, Adadelta, and many others [10].

- Adadelta: Adadelta is an optimizer that uses the Adadelta algorithm [10].
- Adagrad: an optimizer that uses the Adagrad algorithm [10].
- Adam: an optimizer that uses the Adam algorithm [10].
- Adamax: an optimizer that uses the Adamax algorithm [10].
- Ftrl: Ftrl is an optimizer that uses the FTRL algorithm [10].
- Nadam: The NAdam algorithm is implemented by this optimizer [10].
- Optimizer class: Keras optimizers' base class [10].
- RMSprop: This is an optimizer that uses the RMSprop algorithm [10].
- SGD: Optimizer for gradient descent (with momentum) [10].

keras.layers: Keras layers are the most basic component of keras models [11]. Layers are made with a variety of layer_ functions, and they're usually put togetherby stacking calls to them with the pipe operator [11].

There are several different levels to choose from, including:

- Core Layers
- Convolutional Layers
- Pooling Layers
- Activation Layers
- Dropout Layers
- Locally-connected Layers
- Recurrent Layers
- Embedding Layers
- Normalization Layers
- Noise Layers
- Merge Layers
- Layer Wrappers

keras.utils: Keras includes the NumPy utility library, which includes routines for manipulating NumPy arrays [12]. A NumPy array (or) a vector with integers that represent distinct categories can be transformed into a NumPy array (or) a matrix with binary values and columns equal to the number of categories in the data using the method to categorical() [12].This function returns a binary vector matrix [12].

This function returns a binary values matrix (either'1' or '0') [12]. It has the same number of rows as the input vector and the same number of columns as the number of classes [12].

keras.callbacks: A call back is a set of functions that can be used at specific points during the training process [13]. During training, you can use call-backs to acquire a view of the model's internal states and statistics [13]. When you wish to automate some tasks after each training/epoch, you define and use a call back [13]. This allows you to have more control over the training process [13]. Stopping training when you attain a specified accuracy/loss score, saving your model as a checkpoint after each successful session, altering learning rates over time, and more are all examples of this [13].

keras.applications: The Keras applications module is used to deliver deep neural network models that have already been trained [14]. Keras models are used for feature extraction, prediction, and fine tuning [14]. Model Architecture and modelWeights are the two aspects of a trained model [14]. Because model weights are a huge file, we must download and extract them from the ImageNet database [14].

Sklearn.model_selection: This approach divides arrays or matrices into random train and test subsets [15]. When running a (supervised) machine learning experiment, it's typical to set aside a portion of the data as a test set [15]. This allows us to assess how effectively the model will generalize to new data sets not observed during the training phase [15]. In other words, this technique attempts toavoid the overfitting issue, in which the algorithm begins to memorize the labels from the training set but is unable to predict anything for the unknown test data. [15]. However, if we only do a train/test split, the findings may be influenced by a random selection for the pair of train/test sets [15].

matplotlib.pyplot:

Pyplot is a Python matplotlib API that essentially turns matplotlib into a viable open source alternative to MATLAB. Matplotlib is a data visualization package that generates plots, graphs, and charts [16]. Pyplot adds two important featuresto matplotlib [16]:

A MATLAB-style interface makes it easier for individuals who are already familiar with MATLAB to learn Python [16]. Pyplot is stateful, which means it remembers the state of an object when you plot it for the first time [16]. Until plt.close() is encountered in the code, this is required for use in the same loop or session state[16]. When establishing multiple plots in a row, state is also significant [16].

The pyplot API is made up of a hierarchy of Python code objects that comprises anumber of functions, the most important of which is matplotlib.pyplot [16]. This stack can be divided into three layers, each of which is interdependent [16]:

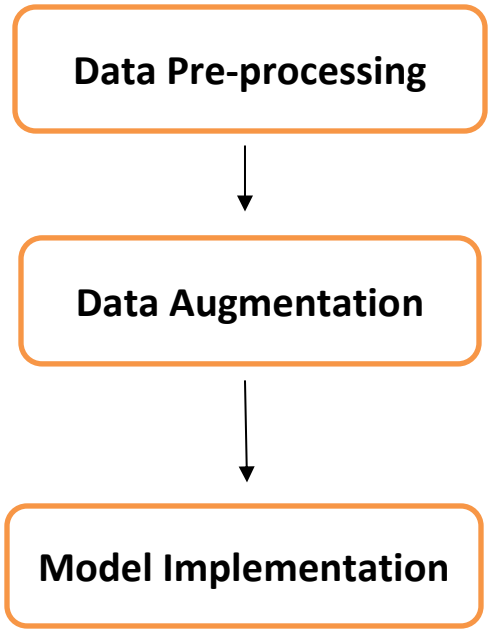
- Scripting layer – this layer is used to define a figure that has one or moreplots with axes [16].
- Artist Layer – used to alter plot features such adding labels, drawing lines, and so on [16].
- Backend Layer – used to style a graphic for display in a target applicationlike a Jupyter Notebook [16].

CHAPTER 7: Methodology

This chapter gives an overview of the different parts of the work chronologically. It mainly discusses the theories, techniques, and step by step workflow of the work.

7.1 Workflow

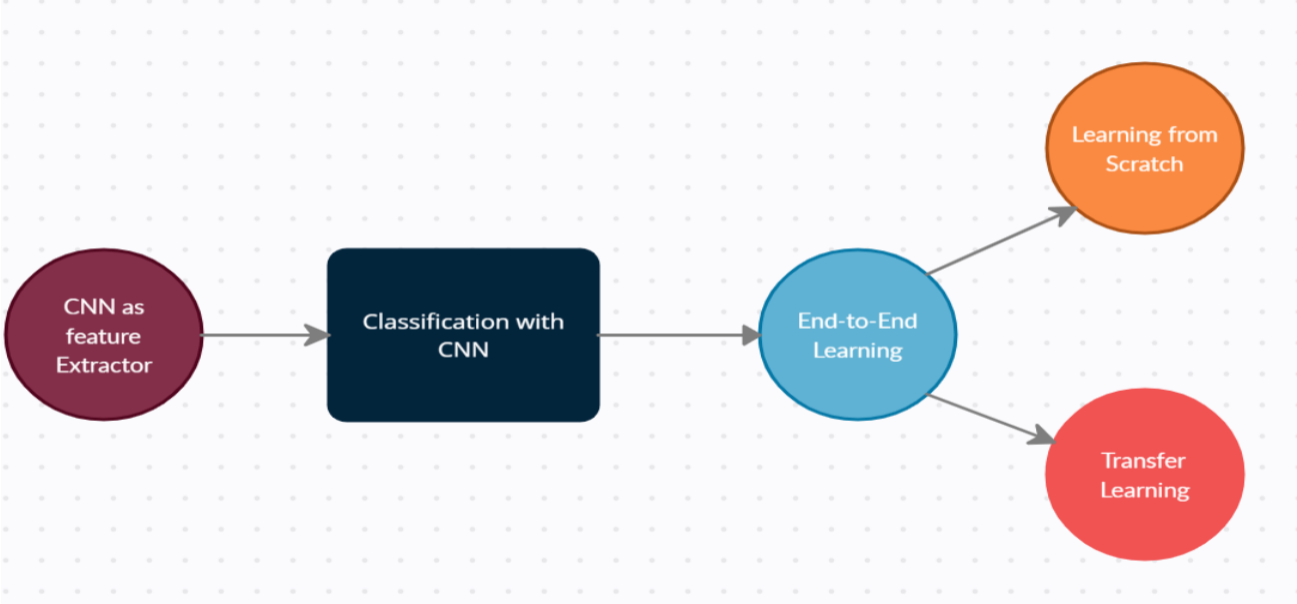
A complete workflow diagram of the proposed method is shown here.



The above diagram shows the complete picture for better understanding of the whole system.

7.2 Implemented Model:

We used several famous models, so that we could use Transfer Learning.



7.2.1 VGG16

VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.
Now let's get to the detail implementation of VGG16. I will**:**

- Show the overview of VGG16

Overview

The architecture depicted below is VGG16.

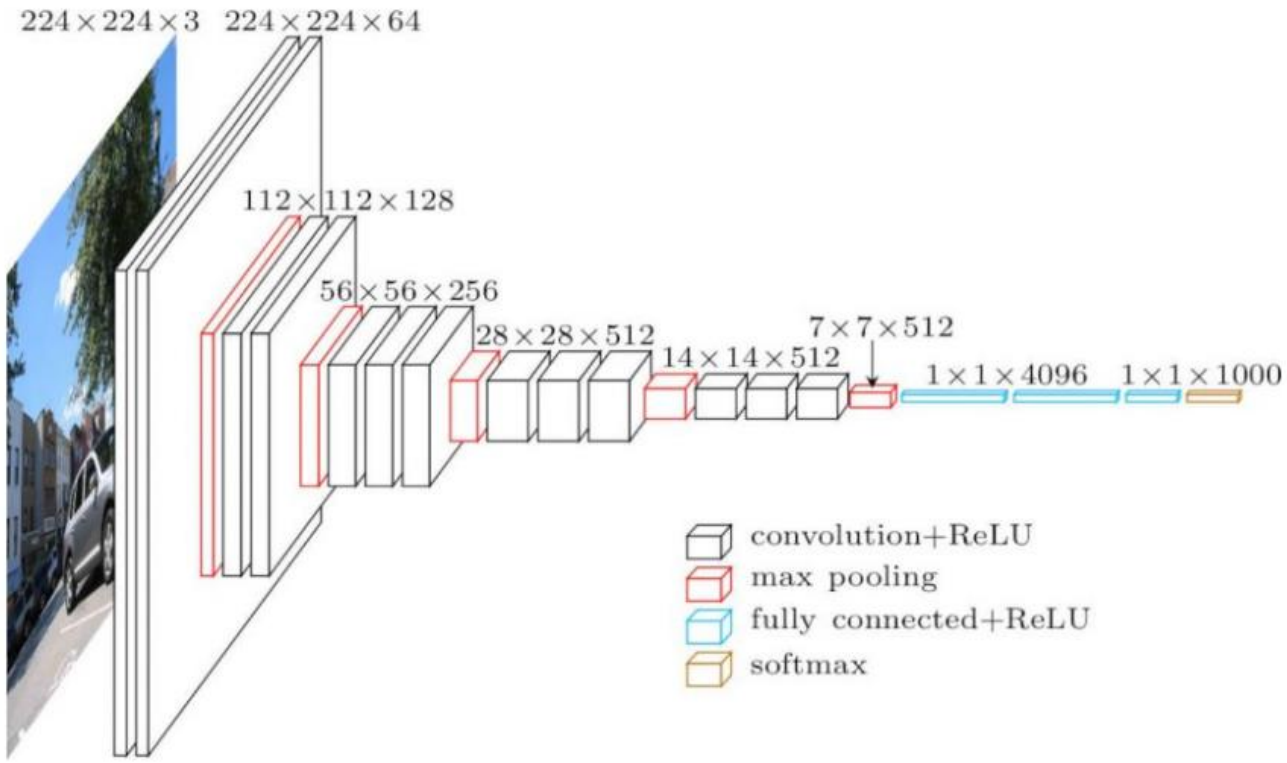


Figure 1. VGG16 Architecture

The input to cov1 layer is of fixed size 224 x 224 RGB image. 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. [1]

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. [1]

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

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.

7.2.2 VGG19

VGG-19 is a convolutional neural network that is 19 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.[2]

Now let's get to the detail implementation of VGG19. I will**:**

- Show the overview of VGG19

Overview

- A fixed size of (224 * 224) RGB image was given as input to this network which means that the matrix was of shape (224,224,3). [2]
- The only preprocessing that was done is that they subtracted the mean RGB value from each pixel, computed over the whole training set. [2]
- Used kernels of (3 * 3) size with a stride size of 1 pixel, this enabled them to cover the whole notion of the image. [2]
- spatial padding was used to preserve the spatial resolution of the image.
- max pooling was performed over a 2 * 2 pixel windows with sride 2. [2]
- this was followed by Rectified linear unit(ReLu) to introduce non-linearity to make the model classify better and to improve computational time as the previous models used tanh or sigmoid functions this proved much better than those. [2]
- implemented three fully connected layers from which first two were of size 4096 and after that a layer with 1000 channels for 1000-way *ILSVRC* classification and the final layer is a softmax function. [2]

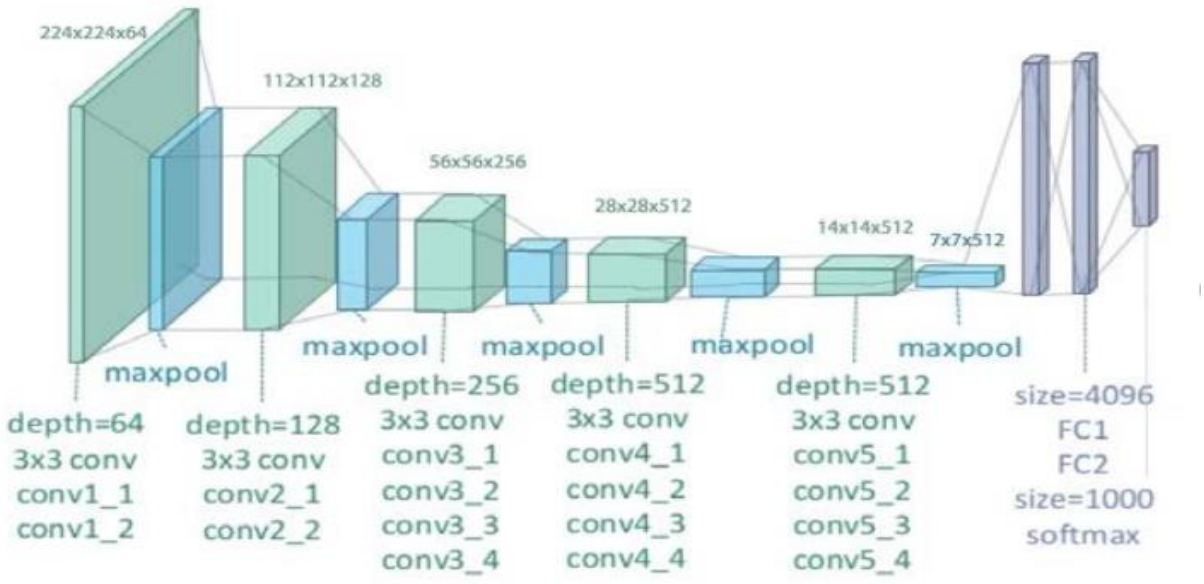


Illustration of the network architecture of VGG-19 model: conv means convolution, FC means fully connected

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The column E in the table is for VGG19 (other columns are for other variants of VGG models)

7.2.3 Inception V3

Inception v3 is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for Googlenet. It is the third edition of Google's Inception Convolutional Neural Network, originally introduced during the ImageNet Recognition Challenge. The design of Inceptionv3 was intended to allow deeper networks while also keeping the number of parameters from growing too large: it has "under 25 million parameters", compared against 60 million for AlexNet. [3]

Now let’s get to the detail implementation of InceptionV3. I will**:**

- Show the overview of InceptionV3

<u>Overview</u>

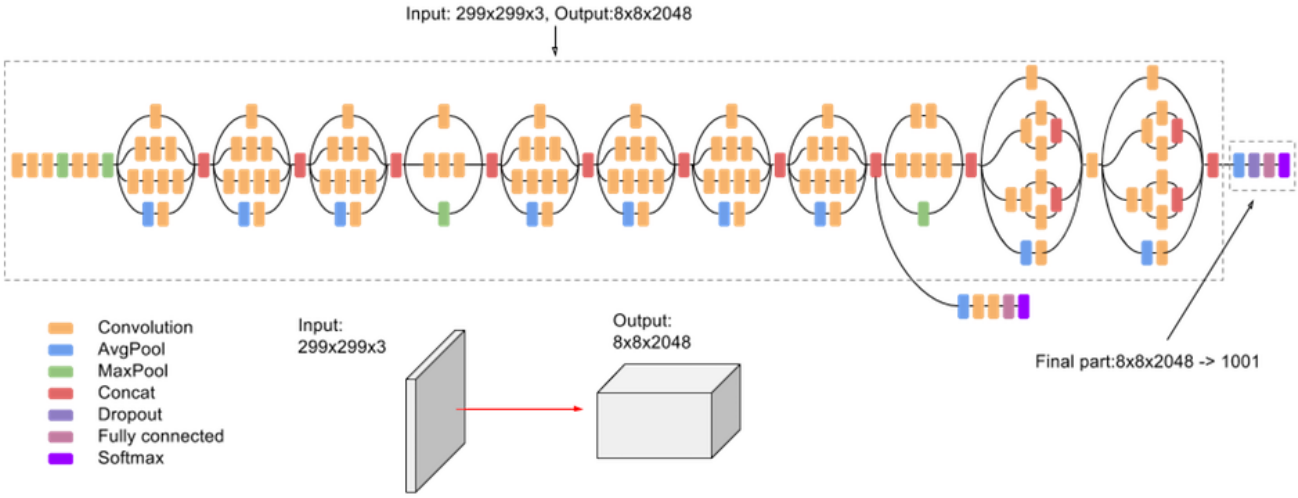
The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:

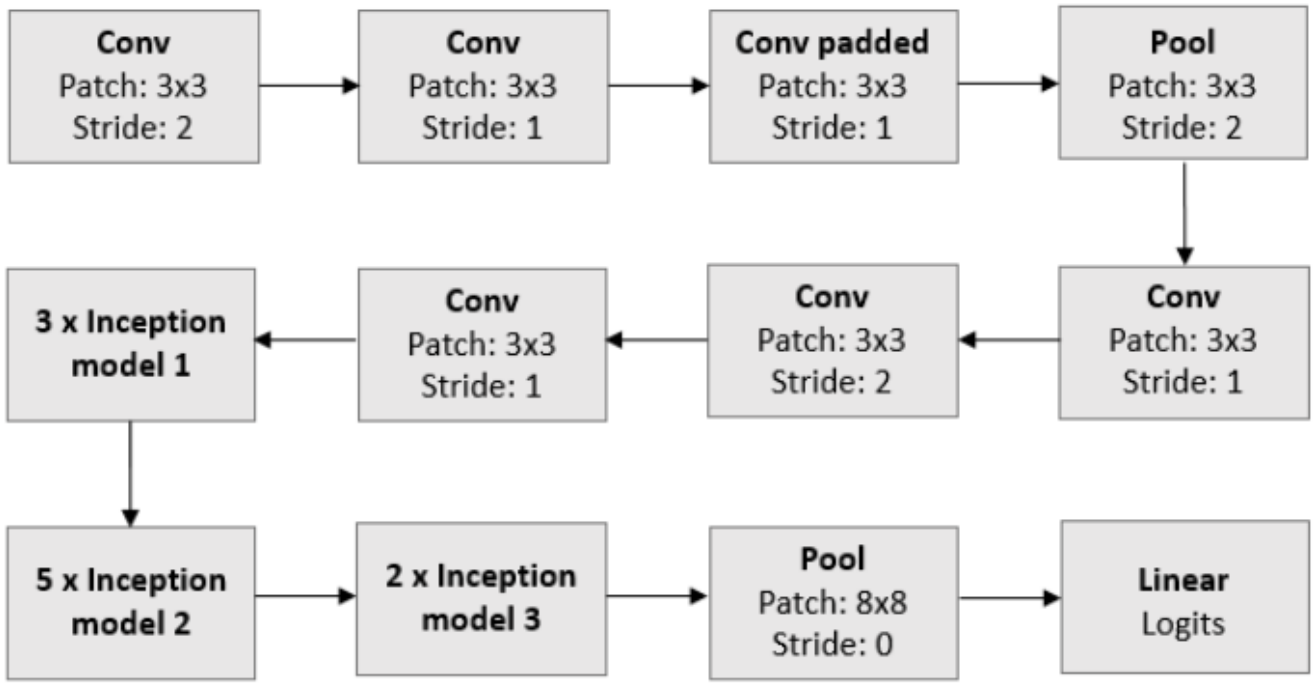
1. **Factorized Convolutions:** this helps to reduce the computational efficiency as it reduces the number of

parameters involved in a network. It also keeps a check on the network efficiency.

2. **Smaller convolutions:** replacing bigger convolutions with smaller convolutions definitely leads to faster training. Say a 5 × 5 filter has 25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3*3 + 3*3) parameters instead. [4]

3. **Asymmetric convolutions:** A 3 × 3 convolution could be replaced by a 1 × 3 convolution followed by a 3 × 1 convolution. If a 3 × 3 convolution is replaced by a 2 × 2 convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed.





The basic architecture of Inception-v3.

4. **Auxiliary classifier:** an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In GoogLeNet auxiliary classifiers were used for a deeper network, whereas in Inception v3 an auxiliary classifier acts as a regularizer.
5. **Grid size reduction:** Grid size reduction is usually done by pooling operations.

7.2.4 ResNet

ResNet network uses a 34-layer plain network architecture inspired by VGG-19 in which then the shortcut connection is added. These shortcut connections then convert the architecture into the residual network. [21]

Now let's get to the detail implementation of ResNet50, ResNet101, ResNet152. I will**:**

- Show the overview of ResNet50, ResNet101, ResNet152

Overview

Each ResNet block is either two layers deep (used in small networks like ResNet 18, 34) or 3 layers deep (ResNet 50, 101, 152). [22]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

50-layer ResNet: Each 2-layer block is replaced in the 34-layer net with this 3-layer bottleneck block, resulting in a 50-layer ResNet (see above table). They use option 2 for increasing dimensions. This model has 3.8 billion FLOPs. [22]

101-layer and 152-layer ResNets: they construct 101-layer and 152-layer ResNets by using more 3-layer blocks (above table). Even after the depth is increased, the 152-layer ResNet (11.3 billion FLOPs) has lower complexity than VGG-16/19 nets (15.3/19.6 billion FLOPs). [22]

7.2.5 DenseNet121

A DenseNet is a type of convolutional neural network that utilises dense connections between layers, through Dense Blocks, where we connect all layers (with matching feature-map sizes) directly with each other. [23]

Now let's get to the detail implementation of ResNet50, ResNet101, ResNet152. I will**:**

- Show the overview of ResNet50, ResNet101, ResNet152

Overview

A summarization of the various architectures implemented for the ImageNet database have been provided in the table above. Stride is the number of pixels shifts over the input matrix. A stride of 'n' (default value being 1), indicates that the filters are moved 'n' pixels at a time. [24]

Using the DenseNet-121 architecture to understand the table, we can see that every dense block has varying number of layers (repetitions) featuring two convolutions each; a 1x1 sized kernel as the bottleneck layer and 3x3 kernel to perform the convolution operation. [24]

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Also, each transition layer has a 1x1 convolutional layer and a 2x2 average pooling layer with a stride of 2. Thus, the layers present are as follows:

1. Basic convolution layer with 64 filters of size 7X7 and a stride of 2
2. Basic pooling layer with 3x3 max pooling and a stride of 2
3. Dense Block 1 with 2 convolutions repeated 6 times
4. Transition layer 1 (1 Conv + 1 AvgPool)
5. Dense Block 2 with 2 convolutions repeated 12 times
6. Transition layer 2 (1 Conv + 1 AvgPool)
7. Dense Block 3 with 2 convolutions repeated 24 times
8. Transition layer 3 (1 Conv + 1 AvgPool)
9. Dense Block 4 with 2 convolutions repeated 16 times
10. Global Average Pooling layer- accepts all the feature maps of the network to perform classification
11. Output layer

Therefore, DenseNet-121 has the following layers:

- 1 7x7 Convolution
- 58 3x3 Convolution
- 61 1x1 Convolution
- 4 AvgPool
- 1 Fully Connected Layer

In short, DenseNet-121 has 120 Convolutions and 4 AvgPool.

All layers i.e. those within the same dense block and transition layers, spread their weights over multiple inputs which allows deeper layers to use features extracted early on.

Since transition layers outputs many redundant features, the layers in the second and third dense block

assign the least weights to the output of the transition layers.
Also, even though the weights of the entire dense block are used by the final layers, there still may be more high level features generated deeper into the model as there seemed to be a higher concentration towards final feature maps in experiments.

7.3 Data Preprocessing

Preprocessing data is an important task and is very important for accuracy. This is because, data is mostly noisy. It sometimes has missing values and also false values. So, feature selection and feature extractions are two major methods of Preprocessing which directly impact the accuracy of the model. Feature selection is a method of selecting some features out of the data and discarding the irrelevant ones. Feature extraction is the method which convert M attributes of the data to N attributes. The original number of attributes may be greater than the new number of attributes. They may be less or equal to as well. In our **Drive** and **Stare** datasets each image can be made up of thousands of pixels. In that case, the algorithm we train with it may turn out to be inefficient. So, changing pixels can be an important part of data preprocessing. The feature extraction techniques in images are mostly dependent on the data. In our dataset, our data was not level. So, we are used '**One-Hot-Encoding'** to level our data.

Pixel changing and Stacking

As we know, in our given dataset, all the data were divided into 3 Folds and the size of each image were 256x256. At first, we stacked all the Folds together and turn 3 Folds into 1 Fold. So that, we could have more data during Training to fit the models better to get a higher accuracy. After stacking (v-stacking the images and concatenating the labels), we got a total of 7901 images in one-Fold together. Then we change the pixel size of each image from 256x256 to 224x224 as most of the models we implemented require the input size to be 224x224.

One-Hot-Encoding

One-Hot-Encoding is a technique for converting categorical variables into a format that can be fed into machine learning algorithms to improve prediction accuracy. A technique for representing categorical variables as binary vectors is also known as One Hot Encoding. The categorical variables must be converted to integer values for this to work. The binary 0 and 1 will be represented using the integer values. This is a difficult situation. This is implemented in the SciKit-Sklearn (One Hot Encoder) library used in Python.
When using categorical data in linear and SVM models, this is critical. Because many Machine Learning models can't work with categorical data directly, this is part of the data pre-processing stage. The data must first be converted to binary vectors. The encoder can find the unique values per feature in a two-feature dataset and convert the data to a binary one-hot encoding.

In this strategy, each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column.

Training-Validation-Testing Split:

After one hot encoding, we needed to split the whole dataset into 3 sets: Training, Validation, Testing. We splits into 3 sets in these ratio – Training: Validation : Testing = 0.68 : 0.17: 0.15.

7.4 Data Augmentation

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data [18]. It acts as a regularize and helps reduce overfitting when training a machine learning model. Data augmentation is very useful if you would like to augment your data or increase the amount of training or validation data. Now if you have to 2000 images let's say and you would like to get 5,000 or 10,000 of those then this can be very useful. But if you only have 5 or 10 images don't expect to get 2,000 or 20,000 images sort of data augmentation a still be able to get decent results out of your deep learning. Because, Deep learning will be highly biased towards these 5 or 10 images that you're augmenting.
In augmentation part, we did rotation, zoom, centering, standard normalization, changing brightness/height/weight, horizontal/vertical flipping, rescaling etc.

7.5 Optimizer

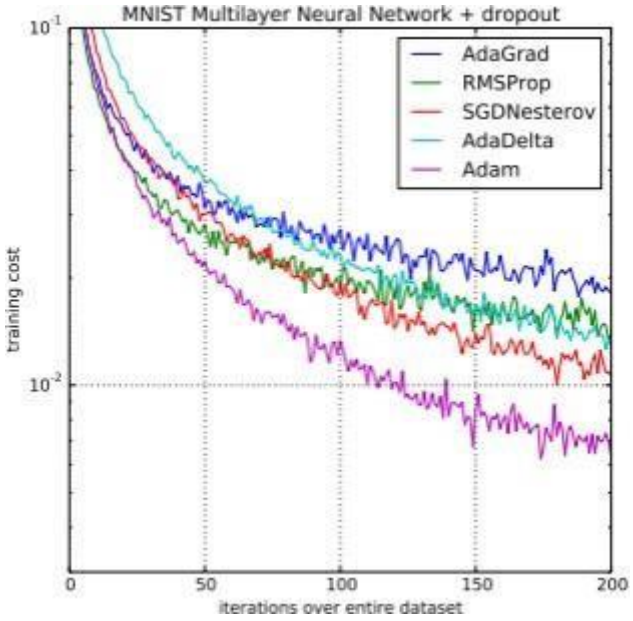
Optimizers are algorithms or methods used to change the attributes of your neuralnetwork such as weights and learning rate in order to reduce the losses. Optimizershelp to get results faster. There are lots of Optimizer in image processing. But we used Adam Optimizer.
Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data

How Does Adam Work?

Adam is not the same as traditional stochastic gradient descent. For all weight updates, stochastic gradient descent maintains a single learning rate (called alpha),which does not change during training. Each network weight (parameter) has its own learning rate, which is adjusted separately as learning progresses.
"The method computes individual adaptive learning rates for different parametersfrom estimates of first and second moments of the gradients [20]."
The authors describe Adam as combining the advantages of two other extensionsof stochastic gradient descent [20]. Specifically:

- Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameterlearning rate that improves performance on problems with sparse gradients (e.g., natural language and computer vision problems) [20].
- Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g., how quickly it is changing).This means the algorithm does well on online and non-stationary problems (e.g., noisy) [20].

Using large models and datasets, we demonstrate Adam can efficiently solvepractical deep learning problems.



Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron

Taken from Adam: A Method for Stochastic Optimization, 2015

7.6 Loss Function

One of the most important components of Neural Networks is the loss function. Loss is nothing more than a Neural Net prediction error. Loss Function is the method for calculating the loss.

The error between the algorithm's output and a given target value is quantified bya loss function, cost function, or error function.

The loss function binary cross entropy tells you how inaccurate your model's predictions are. It's used to make yes-or-no decisions. Consider multi-label classification issues. Where an example can belong to multiple classes at once, themodel tries to determine whether the example belongs to that class or not for eachclass.

Cross-entropy is the default loss function to use for binary classification problems.

It is intended for use with binary classification where the target values are in the set {0, 1}.

Under the maximum likelihood inference framework, it is the preferred loss function mathematically. It is the loss function that should be evaluated first, and only changed if there is a compelling reason to do so.

For predicting class 1, cross-entropy will compute a score that summarizes the average difference between the actual and predicted probability distributions. Thescore is minimized and a perfect cross-entropy value is 0.

Cross-entropy can be specified as the loss function in Keras by specifying

‘binary_crossentropy‘ when compiling the model.

```
model.compile(loss='binary_crossentropy', optimizer=opt,metrics=['accuracy'])
```

The function requires that the output layer is configured with a single node and a

‘sigmoid ‘activation in order to predict the probability for class 1.

```
model.add(Dense(1, activation='sigmoid')
```

7.7 Activation Function

The output of a neural network is determined by activation functions, which are mathematical equations. The function is attached to each neuron in the network and determines whether it should be activated that means fired or not based on whether each neurons input is relevant for the model's prediction. Activation functions also help normalize the output of each neuron to a range between 1 and

0 or between minus 1 and 1. Additionally, because activation functions are calculated across thousands or even millions of neurons for each data sample, theymust be computationally efficient. Modern neural networks use a technique calledback propagation to train the model which places an increased computational strain on the activation function and its derivative function. In a neural network numeric data points called inputs are fed into the neurons in the inputs layer. Eachneuron has a weigh and multiplying the input number with the weight gives the output of the neuron which is transferred to the next layer.

The activation function is a mathematical gate that connects the current neuron's input to its output, which goes to the next layer. It can be as simple as a step function that turns the neuron output on and off depending on a ruler threshold. Or it can be a transformation that map's the input signals into output signals that are needed for the neural network to function. Increasingly neural networks use nonlinear activation functions which can help the network learn complex data compute and learn almost any function representing a question and provide accurate predictions.

Softmax

The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1.[27]

The softmax function is sometimes called the softargmax function, or multi-class logistic regression. This is because the softmax is a generalization of logistic regression that can be used for multi-class classification, and its formula is very similar to the sigmoid function which is used for logistic regression. The softmax function can be used in a classifier only when the classes are mutually exclusive. [27]

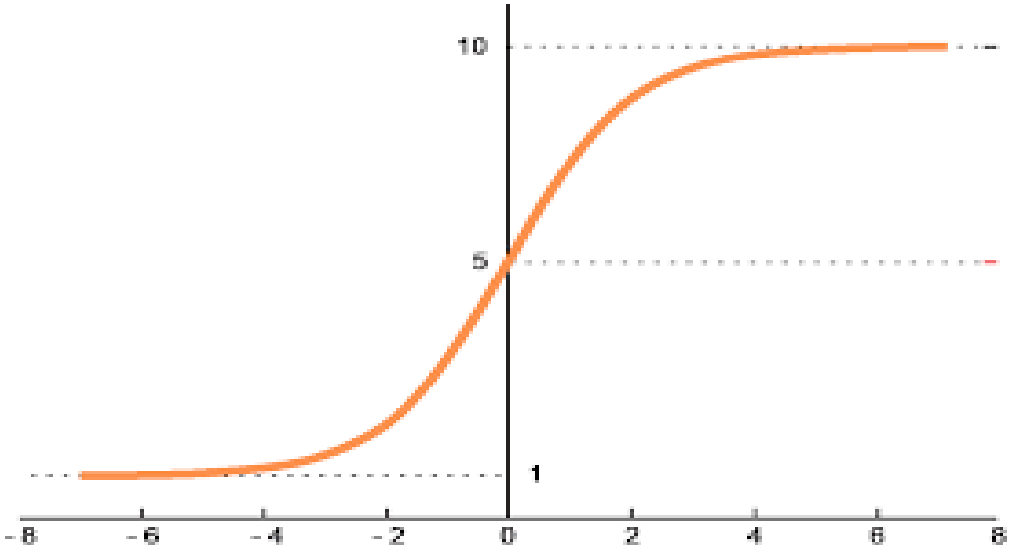
Many multi-layer neural networks end in a penultimate layer which outputs real-valued scores that are not conveniently scaled and which may be difficult to work with. Here the softmax is very useful because it converts the scores to a normalized probability distribution, which can be displayed to a user or used as input to other systems. For this reason, it is usual to append a softmax function as the final layer of the neural network. [27]

Softmax Formula

The softmax formula is as follows:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Mathematical definition of the softmax function



where all the zi values are the elements of the input vector and can take any real value. The term on the bottom of the formula is the normalization term which ensures that all the output values of the function will sum to 1, thus constituting a valid probability distribution. [27]

7.8 Hyper-Parameters

In machine learning, a hyperparameter is a parameter whose value is used to control the learning process and whose value is given by the Programmer where the values of other parameters are derived via training. In Deep Learning algorithms, there are so many hyper-parameters whose values need to be adjusted like knobs which makes it really annoying.

Choosing hyper-parameter values is formally equivalent to the question of model selection. We define a hyper-parameter for a learning algorithm A as a variable to be set prior to the actual application of A to the data, one that is not directly selected by the learning algorithm itself. It is basically an outside control knob. The hyper-parameters can be fixed by hand or tuned by an algorithm, but their value has to be selected. The value of some hyperparameters can be selected based on the performance of A on its training data, but most cannot. For any hyper-parameter that has an impact on the effective capacity of a learner, it makes more sense to select its value based on out-of-sample data (outside the training set), e.g., a validation set performance, online error, or cross-validation error.[26]

Training Hyperparameters	Spatial Feature Learning Hyperparameters
Learning rate, ϵ	Patch size, m
Momentum, α	Convolutional layers, g
Learning rate decay, ϵ_d	Fully connected layers, t
Early stopping patience,	Number of filters, k
Maximum number of epochs, e_n	Filter size, h
Weight decay, λ	
Dropout rate, d_r	

7.8.1 Hyper-Parameters of the Approximate Optimization

- **The initial learning rate:**

  This is often the single most important hyperparameter and one should always make sure that it has been tuned (up to approximately a factor of 2). Typical values for a neural network with standardized inputs (or inputs mapped to the (0,1) interval) are less than 1 and greater than 10^−6 but these should not be taken as strict ranges and greatly depend on the parametrization of the model.[26]
  We started the initial learning rate with 0.001 and then started tuning. And we saw that, as the learning rate decreases, the model performs better. So, the final learning rate, we decided on was 0.0001.

- **Learning rate scheduler:**

  The choice of strategy for decreasing or adapting the learning rate schedule (with hyperparameters such as the time constant τ in the Equation below). The default value of $\tau \rightarrow 1$ means that the learning rate is constant over training iterations. In many cases the benefit of choosing this default value is small. An example of O(1/t) learning rate schedule, used in Bergstra and Bengio (2012) is

$$\epsilon_t = \frac{\epsilon_0 \tau}{\max(t, \tau)}$$

  which keeps the learning rate constant for the first τ steps and then decreases it in $O(1/t^{\alpha})$, with traditional recommendations suggesting α = 1.[26]
  An adaptive and heuristic way of automatically setting τ above is to keep ϵ_t constant until the training criterion stops decreasing significantly from epoch to epoch. An alternative to a fixed schedule with a couple of (global) free hyper-parameters like in the above formula is the use of an adaptive learning rate heuristic at regular intervals during training, using a fixed small subset of the

training set (what matters is only the number of examples used, not what fraction of the whole training set it represents), continue training with N different choices of learning rate (all in parallel), and keep the value that gave the best results until the next re-estimation of the optimal learning rate.[26]

In our case, we used to decrease the learning rate after 8 epochs if the validation accuracy was not increasing, with a factor of 0.6 until we reach the minimum learning rate of 5×10^–5.

- **mini-batch size (B):**

The Mini batch size is typically chosen between 1 and a few hundred.[26]
B = 32 is a good default value that we used in our case.
B with values above 10 takes advantage of the speed-up of matrix-matrix products over matrix-vector products. The impact of B is mostly computational, i.e., larger B yields faster computation (with appropriate implementations) but requires visiting more examples in order to reach the same error, since there are fewer updates per epoch.[26]

- **Number of training iterations**

The number of iterations can be optimized almost for free using the principle of early stopping: by keeping track of the out-of-sample error (as for example estimated on a validation set) as training progresses (every N updates), one can decide how long to train for any given set of all the other hyper-parameters. Early stopping is an inexpensive way to avoid strong overfitting. Practically, one needs to continue training beyond the selected number of training iterations Ť (which should be the point of lowest validation error in the training run) in order to ascertain that
validation error is unlikely to go lower than at the selected point. A heuristic introduced in the '*Deep Learning Tutorials*' is based on the idea of patience, which is a minimum number of training examples to see after the candidate selected point Ť before deciding to stop training. As training proceeds and the new candidate selected points Ť (new minima of the validation error) are observed, the patience parameter is increased, either multiplicatively or additively on top of the last Ť found. Hence, if we find a new minimum at t, we save the current best model, update Ť←t and increase our patience up to t+constant or t× constant. [26]
In our project, we tried varieties of number of iterations starting from 10 to 15 to 20 and so on. Finally, we used early-stopping with a total number of iteration (epoch) of 200 (in the VGG models). But it only went as far as 142 epochs because of early stopping and with the patience of 15.

7.8.2 Hyper-Parameters of the Model and Training Criterion

- **Number of hidden units:** n_h

Each layer in a multi-layer neural network typically has a size that we are free to set and that controls capacity. Larger than optimal values typically do not hurt generalization performance much, but of course, they require proportionally more computation in $O(n_h^2)$, if scaling all the layers at the same time in a fully connected architecture. [26]
With the hidden layers, Our model Summary was (for VGG16 model):

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
```

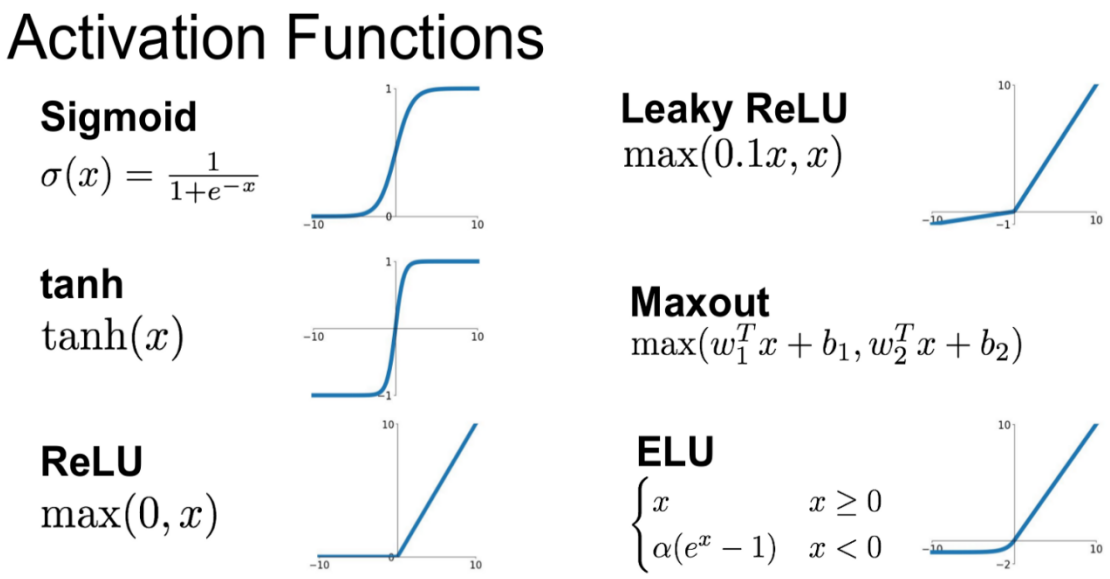
```
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              51382272
_________________________________________________________________
dense_2 (Dense)              (None, 1072)              2196528
_________________________________________________________________
dropout_1 (Dropout)          (None, 1072)              0
_________________________________________________________________
dense_3 (Dense)              (None, 19)                20387
=================================================================
Total params: 68,313,875
Trainable params: 53,599,187
Non-trainable params: 14,714,688
_________________________________________________________________
```

- **Weight decay:**

  Weight decay is a regularization coefficient λ. A way to reduce overfitting is to add a regularization term to the training criterion, which limits the capacity of the learner. The parameters of machine learning models can be regularized by pushing them towards a prior value, which is typically 0. L2 regularization adds a term $\lambda \sum_i \theta_i^2$ to the training criterion, while L1 regularization adds a term $\lambda \sum_i |\theta_i|$. [26]
  We tried using a weight decay of 1×10^–6 in our project but unfortunately, it failed to provide better results. so, we had to eliminate it.

- **Neuron non-linearity:**

  The typical neuron output is s(a) = s(w′x + b), where x is the vector of inputs into the neuron, w is the vector of weights, and b is the offset or bias parameter, while s is a scalar nonlinear function. [26]
  For this, the neural network can successfully approximate functions that do not follow linearity or it can successfully predict the class of a function that is divided by a decision boundary that is not linear. It is very hard to find any physical world phenomenon which follows linearity straightforwardly. We need a nonlinear function that can approximate the non-linear phenomenon. [28]
  There are many non-linearity functions (also known as *'Activation Functions'*) that have been introduced:



In our project, we used ReLU as the activation function which is very computationally efficient.

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

But it has a problem which is that the gradient of the function becomes zero when inputs approach zero or are negative. Then the network cannot perform backpropagation and so cannot learn.

- **Random seeds:**

There are often several sources of randomness in the training of neural networks and deep learners (such as for random
initialization, sampling examples, or sampling corruption noise in denoising autoencoders).
Some random seeds could therefore yield better results than others. Typically, the choice of random seed only has a slight effect on the result and can mostly be ignored in general or for most of the hyper-parameter search process. [26]

7.9 Model fitting

```
model.fit(orig_images, masks, epochs = 15, batch_size= 16, shuffle = True,validation_split = 0.1)
```

The essence of machine learning is model fitting. The outcomes produced by your model will not be accurate enough to be useful for practical decision-making if it does not fit your data correctly. Hyperparameters in a properly fitted model capture the complex relationships between known variables and the target variable, allowing the model to find relevant insights and make accurate predictions.
Fitting is an automated process that ensures your machine learning models have the individual parameters that are best suited to solving your specific real-world business problem with high accuracy.

CHAPTER 7: Problems

In this project while working with our dataset we faced Overfitting problem. In this chapter I will discuss it.

7.1 Overfitting

Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the model does extremely well against the training data. but unfortunately cannot perform accurately later against unseen data.
It is recognized by:

- Low error rates and a high variance are good indicators of overfitting.
- K-fold cross-validation is one of the most popular techniques to assess accuracy of the model.

7.2 How we solved it

As our dataset in each fold is very small (2.5 thousands), OVERFITTING would be a very common problem here.
To solve this problem, we needed Data Regularization.
We used:

- More neurons in the fully connected Layer
- DropOut Layer (Regularization)
- Data Augmentation
- Increase the number of data

CHAPTER 8: Results

In this section, we'll looking at the results.

After data augmentation, we got:

Fold no	Training Accuracy	Validation Accuracy
Fold 1	76%	72%
Fold 2	76%	76%
Fold 3	68%	67%

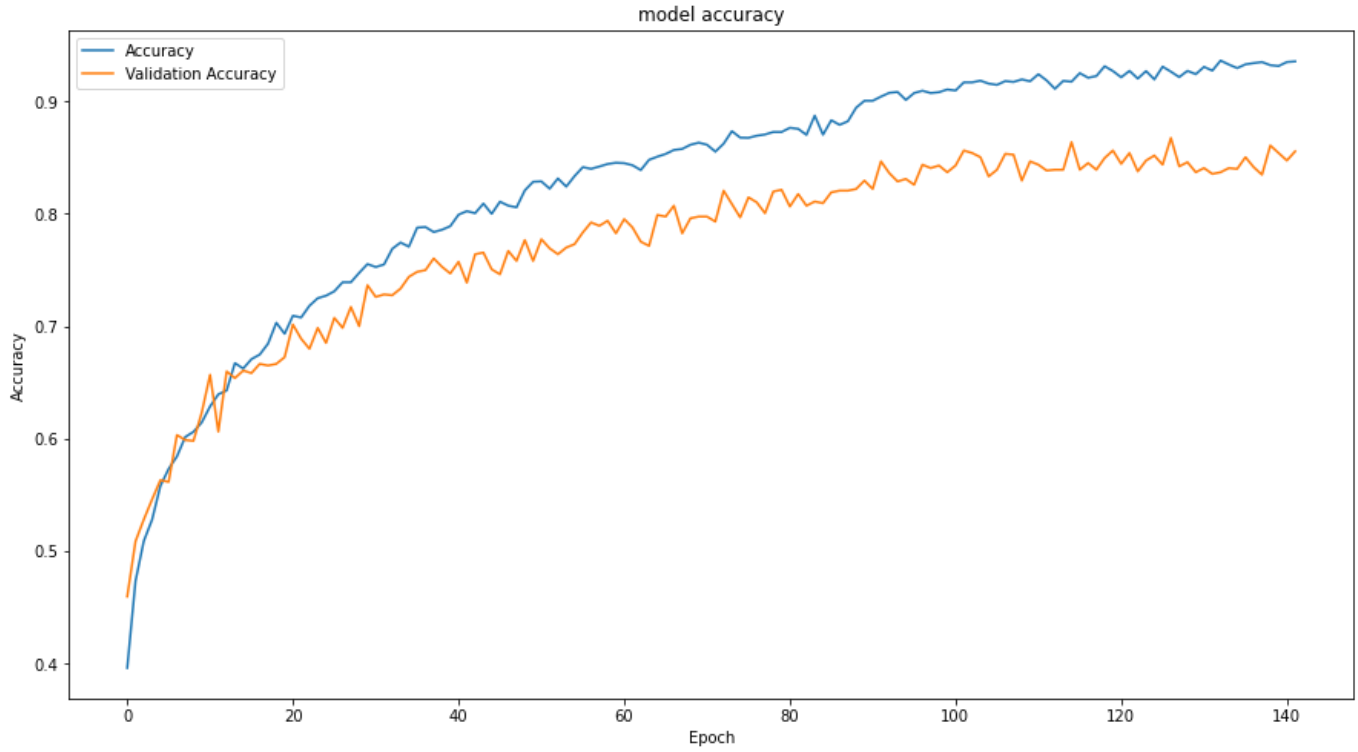
To get more much better accuracy, we needed more data so that the models could train better. So, we merged three folds of data into one and then again train them using different models.

After using different models, we got:

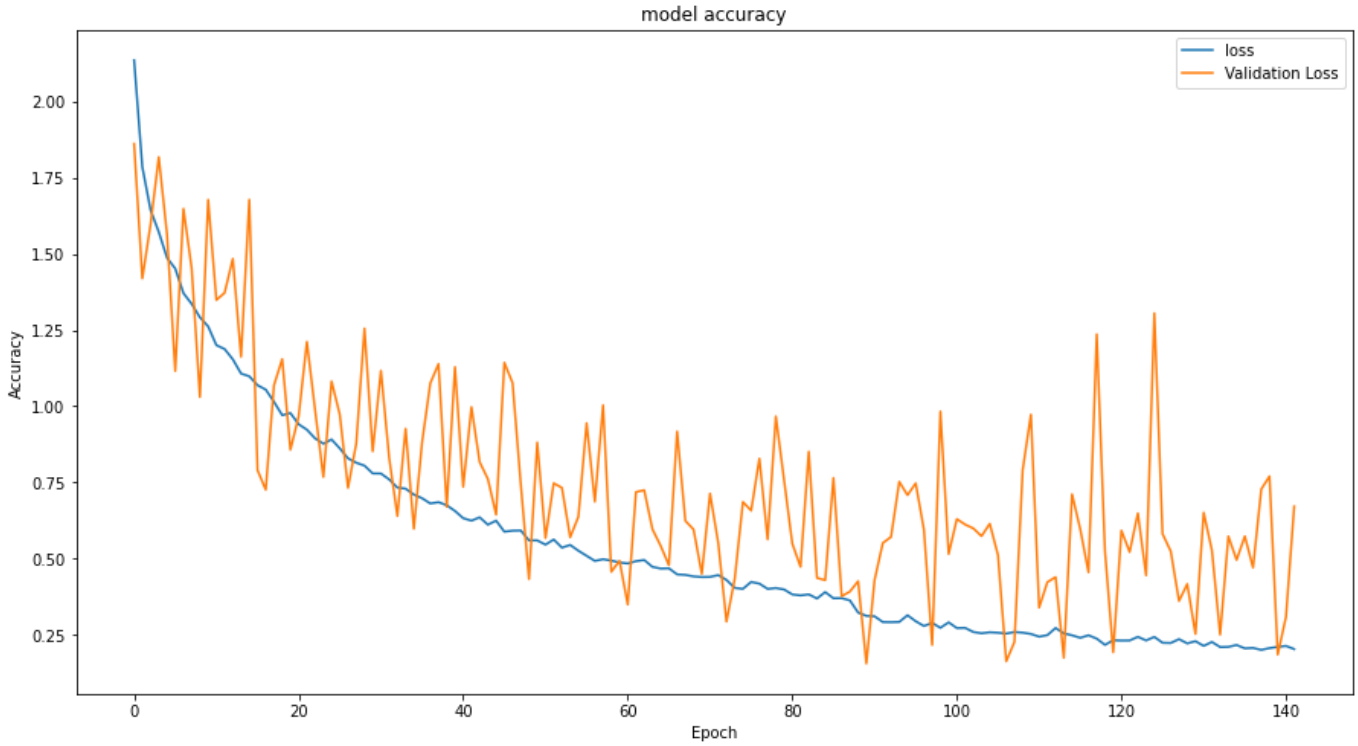
Model Name	Training Accuracy	Validation Accuracy	No of Epoch Needed
InceptionV3	57.4%	27.5%	17
ResNet50	77.4%	17.05%	16
ResNet101	79.6%	19.3%	17
ResNet152	83.5%	18.8%	25
DenseNet121	73.2%	40.1%	18
VGG16	93.4%	86.8%	142
VGG19	90.2%	83.6%	125

As we can see, VGG16 model gave us the best accuracy, so saved this model as our best model and tried to find out the testing accuracy on data that the models has never seen before. After testing, we got almost 70% testing accuracy from VGG16 model.

Here is the Learning Curve : Accuracy



Here is the Learning Curve: Loss



CHAPTER 9: Discussion

Our project was on general cancer classification. Till now, we saw lots of papers on different types of specific cancer classification like breast, lung, brain, skin. But our project is a start of Combination of 19 different types of cancer in one dataset. As Our dataset was divided into 3 folds, we trained 3 folds of data separately at first. At the beginning, we got really low validation accuracy which indicated overfitting. Overfitting means our data was high in variance. So, we needed to find the sweet spot between high variance and high bias which we did through data augmentation. After the data augmentation, we got 72-76% validation accuracy in 3 different folds of data which is still a pretty low accuracy. In order to increase the accuracy, we merged 3 folds of data into one fold which will provide more data during training which can help us to get a better accuracy. After the merge, we got 86% validation accuracy with over 93% training accuracy and 70% test accuracy. Even after that, the accuracy is below our expectations. So, this confirms that more data can provide better accuracy. Merging three folds into one fold, instantly increased accuracy a lot. We also saw, adam optimizer worked better than SGD on our dataset. In the accuracy curve, the training accuracy curve was static but the validation accuracy curve was quite chaotic because of some noise in it.

As it was a generalized cancer project, we have to go through different types of individual cancer classification for research. They were skin, breast, brain, and lung. In our project the dataset we got was in numpy file, we resized the file, and the classification was done on that file. But previously in most of the classification problems, the classification was done on images from directory in jpg/png/mpeg/jepg etc format.

1. As mentioned before, in skin cancer classification, using Mobile Net, they achieved an accuracy of 81.9%. Other models they used were the same as ours. They also mentioned about the lightest computational complexities of Mobile Net. So in future we have a plan to implement this specific model on our dataset, hoping to find a better accuracy.
2. Now if we look into the breast cancer classification, they also used images for classification. But usage of Squeeze-Excitation-Pruning (SEP) block into the hybrid model couldn't help to get better accuracy. But on the other hand we got satisfactory accuracy using pre trained semi customized model.
3. In brain tumor image classification usage of GoogleNet is quite satisfactory with the accuracy of 92.66%. All the important hyper-parameters of CNN models are automatically designated using the grid search optimization algorithm. It is clear that satisfactory classification results are obtained using large and publicly available clinical datasets. Also it is inspiring us to implement another new model in our multiclass cancer classification.

In our results, we only concentrated on accuracy to evaluate our results. But in case of classification, accuracy is not everything. There are some other criteria on which models are evaluated like precision, recall, F1 score, AUC curve, confusion matrix etc. Precision and recall are two extremely important evaluation metrics. Precision means the percentage of the results which are relevant. On the other hand, recall refers to the percentage of total relevant results correctly classified by the algorithm. We could either give a higher priority to maximizing precision, or recall but not both. Because, increasing one, decreases the other. There is one balanced spot between precision and recall which is F1 score which is a harmonic mean of precision and recall. Confusion matrices demonstrate visual display of how correctly the model has classified the classes. With this we can see how many samples are classified classes correctly and what and how many classes are misclassified as which classes. As the size of the dataset was quite large, it was really difficult to train the dataset on personal level computer systems. We wanted to take 128 or 64 as batch size but had to take 16 as batch size because of hardware limitations.
For further research on our project, the semantic segmentation part can be implemented. Semantic

segmentation is the process of classifying each pixel belonging to a particular class/label. It doesn't different across different instances of the same object. With this, it can be pointed out the exact place in the image where cancer happened. Our results can also be implemented for practice purposes. For example, if an app is developed which takes in suspected cancer images for general people as users, they can get the initial idea if they have cancer or not. If the app gives positive results for cancer, then they can consult professional doctors for further medical treatment. But if the result comes out negative, then they didn't have to spend money and time only to find negative results.

For further implementation, we have few more plans-

1. When We have lots of data, then neural network generalizes well. It's better to have more data. When we get more data, we will try to increase accuracy.
2. We can also move our work to segmentation.
3. We will also try to implement our project in android application and web application.
4. We will do AB testing for better performance. It will help the algorithm to fetch real data and experience.

   A/B Testing is a widely used concept in most industries nowadays, and data scientists are at the forefront of implementing it. A/B testing is a basic randomized control experiment. It is a way to compare the two versions of a variable to find out which performs better in a controlled environment.

CHAPTER 10: Conclusion

In this project we wanted to classify multi-class cancer cells using convolution neural network. We have achieved better score using VGG16 model on given dataset. Mainly in this project we observed how a deep learning model works, the ins and outs of the multi-class classification as well as architecture of VGG16, VGG19, InceptionV3, ResNet50, ResNet101, ResNet152, and DenseNet121. This knowledge helps us to do further work in this sector. We could achieve training accuracy of 93.4, and validation accuracy of 86.8 using VGG16. After using VGG19 the training accuracy was of 89, and validation accuracy was of 83. We also applied 3 different versions of ResNet model, 2 different versions of DenseNet model, and Inception V3. For each of the models we got Training Accuracy about 85% - 88%. Validation Accuracy for 3 different versions of ResNet model is about 19% - 20%. Validation Accuracy for DenseNet121 model is about 40% - 42%

And Validation Accuracy for Inception V3 model is 36%.

Now, our future plan for this project.

1. Collecting more data.
2. Second we can also generalize the training data for better performance.
3. We can also move our work to segmentation. And, finally
4. A/B testing

We have equally worked on this project, and tried our best to get the outcome of the task and achieved a good performance. We hope to achieve better result on remaining work in future.

References

[1] "VGG16 – Convolutional Network for Classification and Detection" [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/

[2] "Understanding the VGG19 Architecture" [Online]. Available: https://iq.opengenus.org/vgg19-architecture/

[3] "Inceptionv3" [Online]. Available: https://en.wikipedia.org/wiki/Inceptionv3

[4] https://www.python.org/doc/essays/blurb/. [Accessed 16 July 2021].

[5] "tensorflow," [Online]. Available: https://www.tensorflow.org/federated. [Accessed 16 July 2021].

[6] E. Novoseltseva, "apiumhub," 30 November 2017. [Online]. Available: https://apiumhub.com/tech-blog-barcelona/using-github/. [Accessed 16 July 2021].

[7] M. Driscoll, "realpython," [Online]. Available: https://realpython.com/jupyter-notebook-introduction/#reader-comments. [Accessed 16 July 2021].

[8] "anaconda," [Online]. Available: https://docs.anaconda.com/anaconda/navigator/index.html.[Accessed 16 July 2021].

[9] "tutorialspoint," [Online]. Available: https://www.tutorialspoint.com/keras/keras_models.htm.[Accessed 16 July 2021].

[10] M. MAITHANI, "analyticsindiamag," 18 January 2021. [Online]. Available: https://analyticsindiamag.com/guide-to-tensorflow-keras-optimizers/. [Accessed 16 July 2021].

[11] J. A. F. C. Daniel Falbel, "keras.rstudio," [Online]. Available: https://keras.rstudio.com/articles/about_keras_layers.html.

[12] adnanirshad158, "geeksforgeeks," 23 June 2021. [Online]. Available: https://www.geeksforgeeks.org/python-keras-keras-utils-to_categorical/. [Accessed 16 July 2021].

[13] A. Duong, "kdnuggets," August 2019. [Online]. Available: https://www.kdnuggets.com/2019/08/keras-callbacks-explained-three-minutes.html.

[14] "tutorialspoint," [Online]. Available: https://www.tutorialspoint.com/keras/keras_applications.htm. [Accessed 16 July 2021].

[15] "http://ethen8181.github.io," [Online]. Available: http://ethen8181.github.io/machine-learning/model_selection/model_selection.html. [Accessed 16 July 2021].

[16] "activestate," [Online]. Available: https://www.activestate.com/resources/quick-reads/what-is-pyplot-in-matplotlib/. [Accessed 16 July 2021].

[17] J. Zhang, "towardsdatascience," U-net, 18 Oct 2019. [Online]. Available: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5. [Accessed 17 July2021].

[18] JSTOR, "Wikipedia," Data Augmentation, 01 September 2020. [Online]. Available: https://en.wikipedia.org/wiki/Data_augmentation. [Accessed 17 July 2021].

[19] S. Sinha, "Geeksforgeeks," OpenCV, 29 April 2019. [Online]. Available: https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/. [Accessed 17 July2021].

[20] J. Brownlee, "Machine Learning Mastery," Deep Learning Performance, 3 July 2017. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed 17 July 2021].

[21] Hussain Mujtaba. (2020, September). Introduction to Resnet or Residual Network.

[22] "ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks " [Online]. Available: https://neurohive.io/en/popular-networks/resnet/

[23] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. (*Jan 28, 2018).* Densely Connected Convolutional Networks

[24] "Architecture of DenseNet-121" [Online]. Available: https://iq.opengenus.org/architecture-of-densenet121/

[25] James McDermott. "Convolutional Neural Networks — Image Classification w. Keras"

[26] Yoshua Bengio. (2012, September 16). "Practical Recommendations for Gradient-Based Training of Deep Architectures"

[27] Thomas Wood. "What is the Softmax Function?"

[28] "Understanding Non-Linear Activation Functions in Neural Network" [Online]. Available: https://medium.com/ml-cheat-sheet/understanding-non-linear-activation-functions-in-neural-networks-152f5e101eeb#:~:text=What%20does%20non%2Dlinearity%20mean,boundary%20which%20is%20not%20linear.

[29] Zeduo Yuan; Guoming Chen; Qiang Chen; Wanyi Li; Shun Long. (2020). "Breast Cancer Image Classification Based on CNN Classifier"

[30] Chuang Zhu, Fangzhou Song, Ying Wang, Huihui Dong, Yao Guo & Jun Liu. (2019). "Breast cancer histopathology image classification through assembling multiple compact CNNs"

[31] Emrah Irmak. (April 22, 2021). "Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework"

[32] Tafadzwa L. Chaunzwa, Ahmed Hosny, Yiwen Xu, Andrea Shafer, Nancy Diao, Michael Lanuti, David C. Christiani, Raymond H. Mak & Hugo J. W. L. Aerts. (March 09, 2021). "Deep learning classification of lung cancer histology using CT images"