Major Project Report

on

"Detecting Lightning Strikes in High-Speed Videos"

Submitted in partial fulfillment of the requirement for the award of the degree Of

Bachelor of Technology
In

Computer Science and Engineering

Submitted by

Rishabh Thukral
Roll No: 1444268
Semester : 8th
Batch : 2014-2018

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH GROUP OF COLLEGES- COLLEGE OF ENGINEERING, LANDRAN (MOHALI)
May 2018

Major Project Report

on

“Detecting Lightning Strikes in High-Speed Videos”

Submitted by

Rishabh Thukral
Roll No: 1444268
Semester : 8th
Batch : 2014-2018

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH GROUP OF COLLEGES – COLLEGE OF ENGINEERING, LANDRAN (MOHALI)
May 2018

College Of Engineering
Landran, Mohali

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in the Major Project entitled "**Detecting lightning strikes in high-speed videos"** in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering affiliated to **Punjab Technical University, Jalandhar** and submitted to the Department of Computer Science and Engineering of College Of Engineering, Landran, is an authentic record of my own work carried out during a period from **29th January, 2018** to **31st July, 2018**. The matter represented in this report has not been submitted by me for award of any other degree of this or any other institute/university, under the guidance of **Dr. Suren Chilingaryan and Dr Andreas Kopmann** along with the distance guidance of **Dr. Manish Mahajan**.

Date : - **Name: Rishabh Thukral**

Roll no: 1444268

This is to certify that the above statement made by the candidate is correct to the best of our knowledge and belief.

Date: **I/c Major Project** **Head - CSE**

ACKNOWLEDGEMENT

This internship report and my internship period at Institute of data Processing and Electronics (IPE), KIT, Germany, is an accumulation of many people's endeavor.

I am highly thankful to **"IPE-KIT"** for giving me this opportunity to showcase my abilities and contribute and make a difference in the ongoing research. I am very grateful to my (Supervisor) **Dr. Suren Chilingaryan,** (Department Head) **Dr. Andreas Kopmann,** and the team of physicists in Armenia for their patience, encouragement, and many fruitful discussions from an early stage of this project.

I would like to express my deep regards to **Dr. Manish Mahajan,** Head of Department of Computer Science and Engineering, Chandigarh Group Of Colleges- College Of Engineering, Landran and all the faculty in charge.

I wish to express my appreciation to my family for always believing in me, never failing to provide all the support, and for coping with the pressure that naturally comes with such endeavor.

Rishabh Thukral

About the University

The **Karlsruhe Institute of Technology** (KIT) (German: **Karlsruher Institut für Technologie**) is a public research university and one of the largest research and educational institutions in Germany. KIT was created in 2009 when the University of Karlsruhe (Universität Karlsruhe), founded in 1825 as a public research university and also known as the "Fridericiana", merged with the Karlsruhe Research Center (Forschungszentrum Karlsruhe), which had originally been established in 1956 as a national nuclear research center (Kernforschungszentrum Karlsruhe, or KfK) .

KIT is one of the leading universities for engineering and the natural sciences in Europe, ranking sixth overall in citation impact. **KIT** is a member of the **TU9 German Institutes of Technology** e.V. As part of the German Universities Excellence Initiative KIT was awarded an excellence status in 2006. In the 2011 performance ranking of scientific papers, Karlsruhe ranked first in Germany and among the top ten universities in Europe in engineering and natural sciences. **Ranked 26th worldwide in computer science** in the internationally recognized "Times Higher Education" ranking, KIT is among the leading universities worldwide in computer science.

As of 2018, **six Nobel laureates** are affiliated with KIT. The Karlsruhe Institute of Technology is well known for many inventors and entrepreneurs who studied or taught there, including **Heinrich Hertz, Karl Friedrich Benz and the founders of SAP SE**.

The **Campus Nord** (Campus North), the former **Forschungszentrum**, was founded in 1956 as Kernforschungszentrum Karlsruhe (KfK) (Karlsruhe Nuclear Research Centre). Initial activities focused on Forschungsreaktor 2 (FR2), the first nuclear reactor built by Germany. With the decline of nuclear energy activities in Germany, Kernforschungszentrum Karlsruhe

directed its work increasingly towards alternative areas of basic and applied sciences. This change is reflected in the change of name from Kernforschungszentrum Karlsruhe to Forschungszentrum Karlsruhe with the subheading Technik und Umwelt (technology and environment) added in 1995. This subheading was replaced by in der Helmholtz-Gemeinschaft (in the Helmholtz Association of German Research Centers) in 2002.

Campus Nord is the site of the main **German national nuclear engineering research centre and the Institute for Transuranium Elements**. Also at the site is a nanotechnology research centre and the neutrino experiment KATRIN.

Campus Nord also hosts a 200 metre tall guyed mast for meteorological measurements.

Research at KIT

In 1979, the **Interfakultatives Institut für Anwendungen der Informatik** (Interfaculty Institute for Informatics Applications)[16] was founded. It brings together research in physics, mathematics, and engineering based on computer science. Its mathematical pendant is the Institut für Wissenschaftliches Rechnen und Mathematische Modellbildung (Institute for Scientific Calculations and Mathematical Modelling).[17] Its aim is to enhance the exchange between mathematics and engineering in the fields of scientific calculations.

The **Interfakultatives Institut für Entrepreneurship** (Interfaculty Institute for Entrepreneurship)[18] was established with SAP funding. Its teaching professors were entrepreneurs on their own. Before being shut down in 2010, a former professor of this faculty was Götz Werner, founder of dm-drogerie markt.

In 2001, the **Centre for Functional Nanostructures (CFN)** was established. It merges the fields within material sciences, biology, chemistry, engineering, and physics which are related to nanotechnology. CFN is one of the three **Exzellenzzentren** (English: Excellence Instituitions) of the University of Karlsruhe. Another interdisciplinary institution is the **Centre for Disaster Management and Risk Reduction Technology (CEDIM).**

The **Karlsruhe School of Optics and Photonics (KSOP)** was established in 2006 as a publicly funded project by the Deutsche Forschungsgemeinschaft under the German Universities Excellence Initiative. KSOP was the first graduate school at the University of Karlsruhe and covers the fields of photonic materials and devices, advanced spectroscopy, biomedical photonics, optical systems and solar energy. It is supported by several of the university's institutes and professors. It is also a partner in the EUROPHOTONICS consortium, which provides scholarship for master's and PhD degrees under the European Commission's prestigious Erasmus Mundus cooperation and mobility program.

KIT operates several TCCON stations as part of an international collaborative effort to measure greenhouse gases globally. One station is near the campus.

TABLE OF CONTENTS

Chapters **Page no.**

CHAPTER - 1

INTRODUCTION TO PROJECT

1.1 Overview

Detecting Lightning Strikes from high speed videos (Problem Statement)

The propagation behavior of lightning channels and the reason for formation of different types of lightning has been difficult problems in lightning scientific research field for a long time (Tan et al., 2014). However, many important questions of the lightning initiation are not answered till now. Among them are: How can lightning start when the electric fields in thunderclouds are well below the classical break- down field Ek that is required for electron multiplication and ionization growth? And when the height of the high electric field zone is typically smaller than a kilometer (Dubinova et al., 2015)?

To answer these and other questions on Mt. Aragats networks of electric field sensors, particle detectors and optical cameras were developed. The disturbances of near-surface electric field and coherently changing particle fluxes observed on the Earth surface provide information on the developing Lower positive charged region (LPCR) having a dominant role in the lightning initiation (Nag & Rakov, 2009). Optical cameras operating on the frequencies of 30 Hz confirm the type of lightning and the data from Automotive weather station reveals the weather conditions supporting lightning initiation. But first of all to answer all these unanswered questions, it's really important to find an efficient way to detect the lightning strikes in the data itself, and to separate all the relevant data from the irrelevant junk, so that physicists can get to doing their job

efficiently and this project is an attempt in that detection.

Initially, we just had a sample video file of the feed from the camera under consideration and using this as the training data we had to devise a model that could successfully tell if there is any lightning present in the current frame or not.

For doing this, we first extract all the frames from the given video and generate a dataset out of it on which a model could be trained. If we look at it from a wider angle, and try to break the problem into pieces, it can just be broken down to a simple image classification problem, that whether in the given fame, there is any possibility of it being a lightning image or not.

LDCLI - (Lightning Detection Command Line Interface)

LDCLI is the command line interface build on top of the trained model based on Deep Convolutional Neural Networks, capable enough to solve our classification problem with an accuracy of 79.3%.

Solution:

- The LDCLI interface allows you to train a new model on top of pre-trained Deep CNN architectures like VGGNet and Resnet etc, and also lets you to input hyper parameters as script parameters.
- Using LDCLI user can use the trained model saved as a pytorch check to predict the results for any image file. User also gets the opportunity to predict results for file in bulk as well.

Benefits:

- Reduction (progressive) in manual errors by more than 70%.
- A new approach towards a conventional problem using AI.
- Massive Time saving by like a 1000 times. (telling by experience)

1.2 Deep Learning?

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

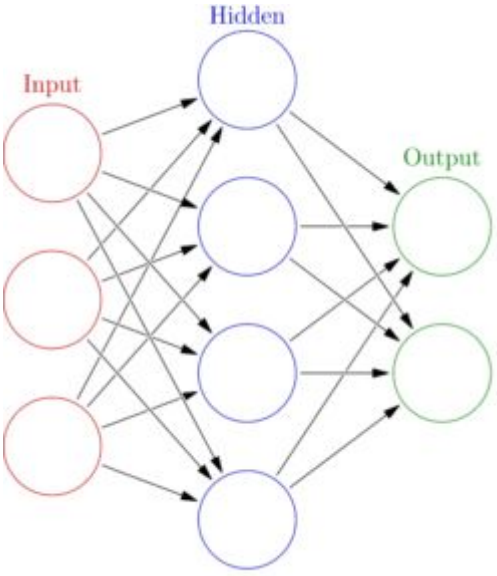
If you are are not in the field of deep learning or you had some experience with neural networks some time ago, you may be confused. The leaders and experts in the field have ideas of what deep learning is and these specific and nuanced perspectives shed a lot of light on what deep learning is all about.

Andrew Ng from Coursera and Chief Scientist at Baidu Research formally founded Google Brain that eventually resulted in the productization of deep learning technologies across a large number of Google services.

Using brain simulations, hope to:

- Make learning algorithms much better and easier to use.
- Make revolutionary advances in machine learning and AI.

I believe this is our best shot at progress towards real AI



Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features

These methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.

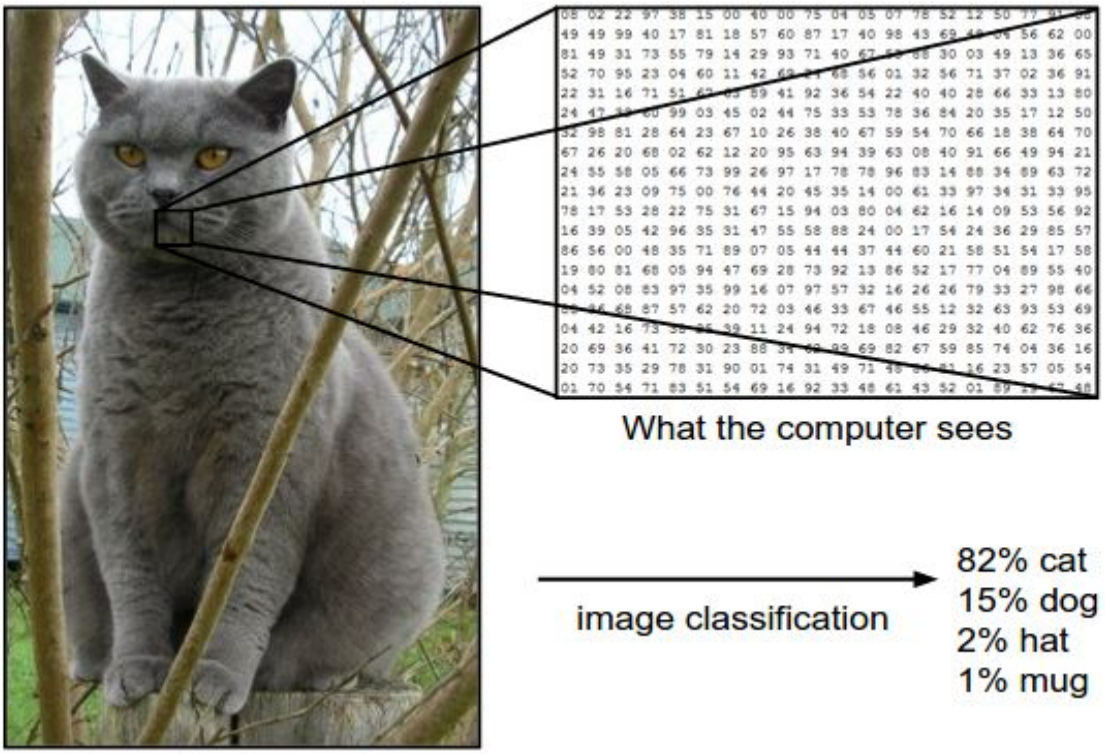
The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning.

The quintessential example of a deep learning model is the feedforward deep network or multilayer perceptron (MLP).

1.2.1. Deep Learning for Image Classification

Image Classification problem is the task of assigning an input image one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Moreover, many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.

For example, in the image below an image classification model takes a single image and assigns probabilities to 4 labels, {cat, dog, hat, mug}. As shown in the image, keep in mind that to a computer an image is represented as one large 3-dimensional array of numbers. In

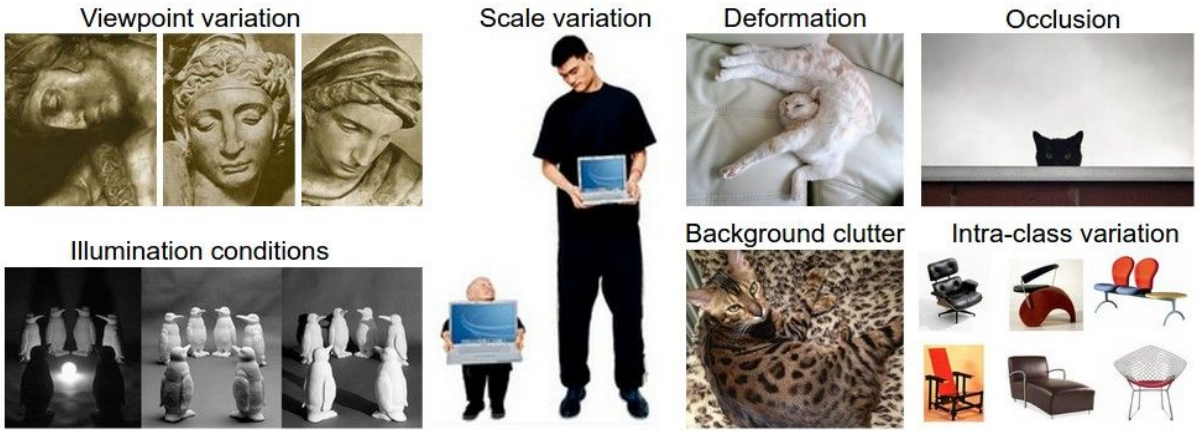


this example, the cat image is 248 pixels wide, 400 pixels tall, and has three color channels Red, Green, Blue (or RGB for short). Therefore, the image consists of 248 x 400 x 3 numbers, or a total of 297,600 numbers. Each number is an integer that ranges from 0 (black) to 255 (white). Our task is to turn this quarter of a million numbers into a single label, such as "cat".

The task in Image Classification is to predict a single label (or a distribution over labels as shown here to indicate our confidence) for a given image. Images are 3-dimensional arrays of integers from 0 to 255, of size Width x Height x 3. The 3 represents the three color channels Red, Green, Blue.

1.2.2. Challenges to overcome

Since this task of recognizing a visual concept (e.g. cat) is relatively trivial for a human to perform, it is worth considering the challenges involved from the perspective of a Computer Vision algorithm. As we present (an non exhaustive) list of challenges below, keep in mind the raw representation of images as a 3-D array of brightness values:



The scope of automation is the area of your Application Under Test which will be automated. Following points help determine scope:

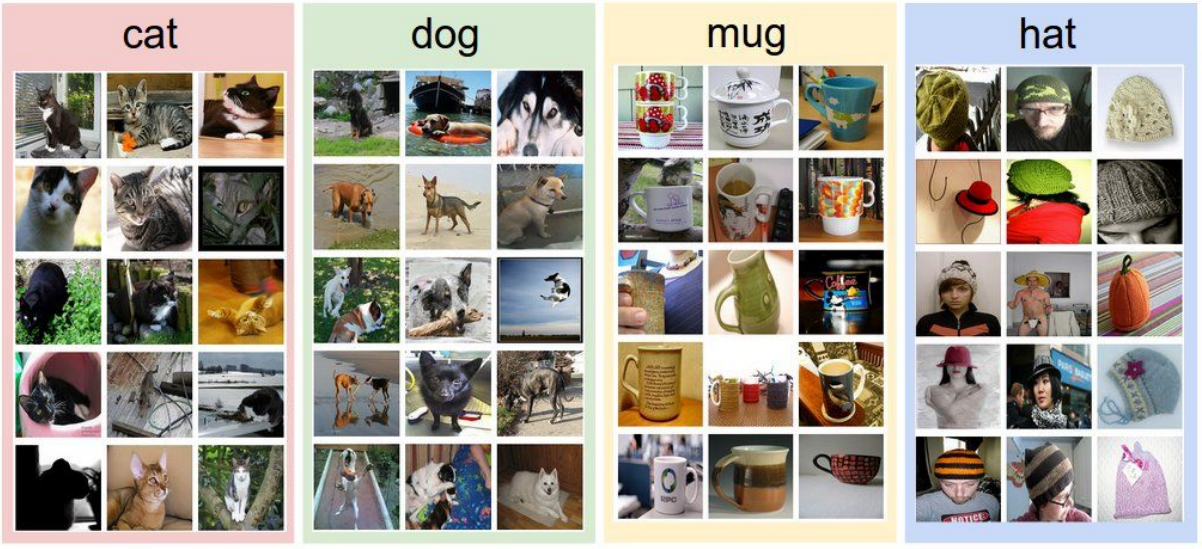
- **Viewpoint variation.** A single instance of an object can be oriented in many ways with respect to the camera.
- **Scale variation.** Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).

- **Deformation.** Many objects of interest are not rigid bodies and can be deformed in extreme ways.
- **Occlusion.** The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.
- **Illumination conditions.** The effects of illumination are drastic on the pixel level.
- **Background clutter.** The objects of interest may blend into their environment, making them hard to identify.
- **Intra-class variation.** The classes of interest can often be relatively broad, such as chair. There are many different types of these objects, each with their own appearance.

A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations.

1.2.3. The Data Driven Approach

How might we go about writing an algorithm that can classify images into distinct categories? Unlike writing an algorithm for, for example, sorting a list of numbers, it is not obvious how one might write an algorithm for identifying cats in images. Therefore, instead



of trying to specify what every one of the categories of interest look like directly in code, the approach that we will take is not unlike one you would take with a child: we're going to

provide the computer with many examples of each class and then develop learning algorithms that look at these examples and learn about the visual appearance of each class. This approach is referred to as a data-driven approach, since it relies on first accumulating a training dataset of labeled images. Here is an example of what such a dataset might look like: this phase, you create Automation strategy & plan, which contains following details.

An example training set for four visual categories. In practice we may have thousands of categories and hundreds of thousands of images for each category.

1.2.4. Image Classification Pipeline

We've seen that the task in Image Classification is to take an array of pixels that represents a single image and assign a label to it. Our complete pipeline can be formalized as follows:

- **Input:** Our input consists of a set of N images, each labeled with one of K different classes. We refer to this data as the training set.
- **Learning:** Our task is to use the training set to learn what every one of the classes looks like. We refer to this step as training a classifier, or learning a model.
- **Evaluation:** In the end, we evaluate the quality of the classifier by asking it to predict labels for a new set of images that it has never seen before. We will then compare the true labels of these images to the ones predicted by the classifier. Intuitively, we're hoping that a lot of the predictions match up with the true answers (which we call the ground truth).

1.3 Why Deep Learning now?

The question is, why use deep learning over any other conventional machine learning or computer vision technique. The problem could have been solved with algorithms like PCA(Principal Component Analysis), but there was no use using it after the results achieved from the Deep Neural Networks. It is an established fact now, that a neural network out performs any other conventional method if you give the model enough data and enough compute.

1.4 Feasibility Analysis

1.4.1 Feasibility Study

An Important outcome of the preliminary investigation is the determination that the system requested is feasible. There are 3 aspects in the feasibility study.

- Technical Feasibility
- Economical Feasibility
- Operational Feasibility

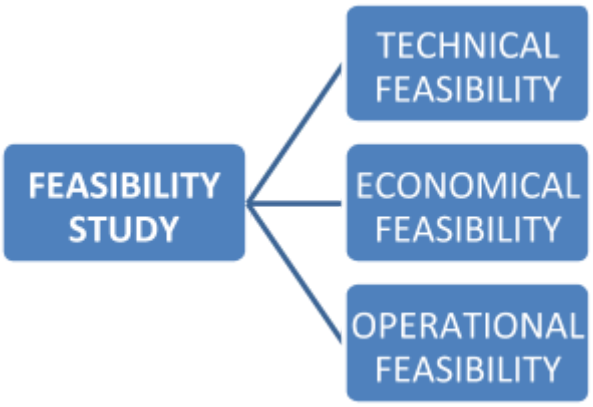


Fig. 1.2 Feasibility study

1.4.2 Technical Feasibility

Technical feasibility study is the complete study of the project in terms of input, processes, output, fields, programs and procedures. It is a very effective tool for long term planning and troubleshooting. The technical feasibility study should most essentially support the financial information of an organization.

We do a complete field study to find out the problems coming in way and give you the best solutions to overcome those problems. We consider many things while doing a technical feasibility study including materials, labor, transportation, physical location and technology. Under the material segment we consider the parts needed to produce a product, supplies and other materials that will be needed in producing the product. Our experts have the skills to identify the significant parts of the system that have the major bearing on its consistency and

performance, and evaluating those parts in a better way. So, if you're really looking for some experts to carry out your technical feasibility study, contact us. We'll give you the right kind of services and you won't get a chance to complain**.**

In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not
- Whether the required resources are available –
- Manpower-programmers, testers & debuggers

1.4.3. Economical Feasibility

Economic feasibility analysis is the most commonly used method for determining the efficiency of a new project. It is also known as cost analysis. It helps in identifying profit against investment expected from a project. Cost and time are the most essential factors involved in this field of study.

We do certain assumptions on the basis of which we give you solid plan of investment. These include

- Economic feasibility cash flow.
- Estimated total project cost.
- Estimated total earnings.
- Risk factors.
- Cost benefits.

For any system if the expected benefits equal or exceed the expected costs, the system can be judged to be economically feasible.

In economic feasibility, cost benefit analysis is done in which expected costs and benefits are evaluated. Economic analysis is used for evaluating the effectiveness of the proposed system.

In economic feasibility, the most important is cost-benefit analysis. As the name suggests, it is an analysis of the costs to be incurred in the system and benefits derivable out of the system.

1.4.4. Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

To ensure success, desired operational outcomes must be imparted during design and development.

These include such design-dependent parameters as reliability, maintainability, supportability, usability, product ability, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realized.

A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters.

A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design.

Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

CHAPTER -2

SYSTEM REQUIREMENT SPECIFICATION

System Analysis and Design refers to the process of examining a situation with the intent of improving it through better procedures and methods. System Analysis is a process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvement to the system. In brief we can say Analysis specifies what system should do. System Analysis is a management technique, which helps us in designing a new system or improving an existing system.

The basic procedure that will be used by the software is to first input the information about all the courses being conducted by the Organization. Then all the information about any new students who joins the college is stored. Once this is done all the other information that needs to be stored about the student namely the marks, attendance details and any other information about the student is also stored.

For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. The output of this process can directly be used into implementation in programming languages.

Software design is the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.

2.1. Software Design Levels

Software design yields three levels of results:

- **Architectural Design -** The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.

- **High-level Design-** The high-level design breaks the 'single entity-multiple component' concept of architectural design into less-abstracted view of subsystems and modules and depicts their interaction with each other. High-level design focuses on how the system along with all of its components can be implemented in forms of

modules. It recognizes modular structure of each sub-system and their relation and interaction among each other.

- **Detailed Design-** Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules.

2.2. Design Verification

The output of software design process is design documentation, pseudo codes, detailed logic diagrams, process diagrams, and detailed description of all functional or non-functional requirements.

The next phase, which is the implementation of software, depends on all outputs mentioned above.

It is then becomes necessary to verify the output before proceeding to the next phase. The early any mistake is detected, the better it is or it might not be detected until testing of the product. If the outputs of design phase are in formal notation form, then their associated tools for verification should be used otherwise a thorough design review can be used for verification and validation.By structured verification approach, reviewers can detect defects that might be caused by overlooking some conditions. A good design review is important for good software design, accuracy and quality.

2.3. Requirement Analysis

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management.

Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded

by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mold user expectations to fit the requirements.

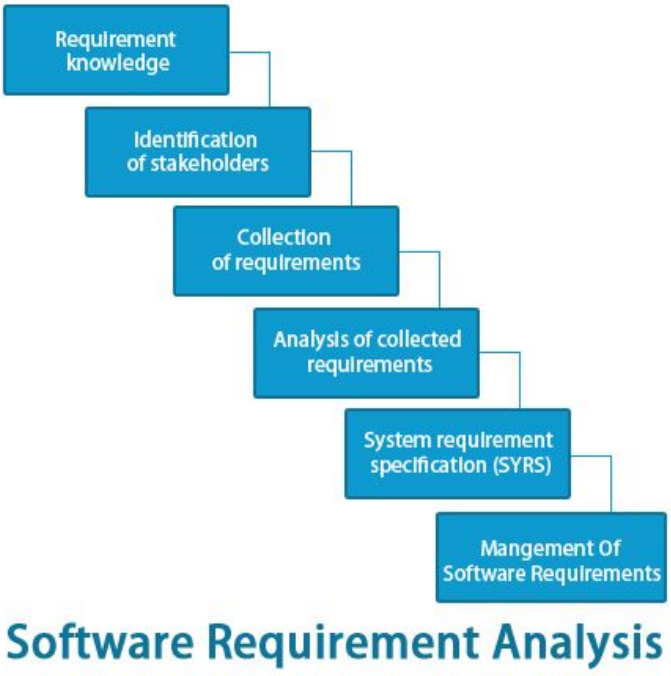


Fig. 2.1 Software Requirement Analysis

In software and system engineering, requirement analysis includes task that governs the condition or requirement to meet for a new product. It includes taking account of conflicting requirements of other stakeholders.

It includes various things:

- Regular communication with the software users to know about their expectations
- Resolution of complaints made by the user or group of users
- Avoid feature creep
- Keep up-to-date all the documentations from starting to present of project development

Necessity of Requirement Analysis

According to statistics major reason of failure of software is that it does not meet with the requirement of the user. It is possible that over years the demands of the client increases and there can come requirement of updating the software.

Requirement analysis involves the task that determines the needs of the software, which mainly includes complaints and needs of various clients/stakeholders. It is a major key in the lifecycle of software and is the starting step of the project.

2.4. Hardware and Software Requirements

The following hardware and software were available with "**ContentTrak"** are:

Specification:

- **Operating System:** Ubuntu Linux and Opensuse.
- **Coding Environment:** Jupyter Notebook
- **Web Browser:** Any browser.
- **Other Tools :** PyTorch
- **Technology:** Python 3.0+ with dependencies stated in requirements.txt installed
- **Compute Required:** Titans and Teslas of Kepler Generation

2.5. Software Requirement Specification

A **software requirements specification** (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system we should have clear understanding of Software system. To achieve this we need to continuous communication with customers to gather all requirements.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios. Using the Software requirements specification (SRS) document on QA

lead, managers creates test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

A software requirements specification (SRS) is a description of a software system to be developed, its defined after business requirements specification (CONOPS) also called stakeholder requirements specification (StRS) other document related is the system requirements specification (SyRS). The software requirements specification (SRS) lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven project, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

Qualities of SRS:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable

CHAPTER - 3

The Science Behind LDCLI

3.1. Neural Networks

It is possible to introduce neural networks without appealing to brain analogies. In linear classification we compute scores for different visual categories given the image using the formula s=Wx, where W was a matrix and x was an input column vector containing all pixel data of the image. In the case of CIFAR-10, x is a [3072 x 1] column vector, and W is a [10 x 3072] matrix, so that the output scores.

An example neural network would instead compute s=W2max(0,W1x). Here, W1 could be, for example, a [100 x 3072] matrix transforming the image into a 100-dimensional intermediate vector. The function max(0,−) is a non-linearity that is applied elementwise. There are several choices we could make for the non-linearity (which we'll study below), but this one is a common choice and simply thresholds all activations that are below zero to zero. Finally, the matrix W2 would then be of size [10 x 100], so that we again get 10 numbers out that we interpret as the class scores. Notice that the nonlinearity is critical computationally - if we left it out, the two matrices could be collapsed to a single matrix, and therefore the predicted class scores would again be a linear function of the input. The nonlinearity is where we get the wiggle. The parameters W2,W1 are learned with stochastic gradient descent, and their gradients are derived with chain rule (and computed with backpropagation).

A three-layer neural network could analogously look like s=W3max(0,W2max(0,W1x)), where all of W3,W2,W1 are parameters to be learned. The sizes of the intermediate hidden vectors are hyperparameters of the network and we'll see how we can set them later. Let's now look into how we can interpret these computations from the neuron/network perspective.

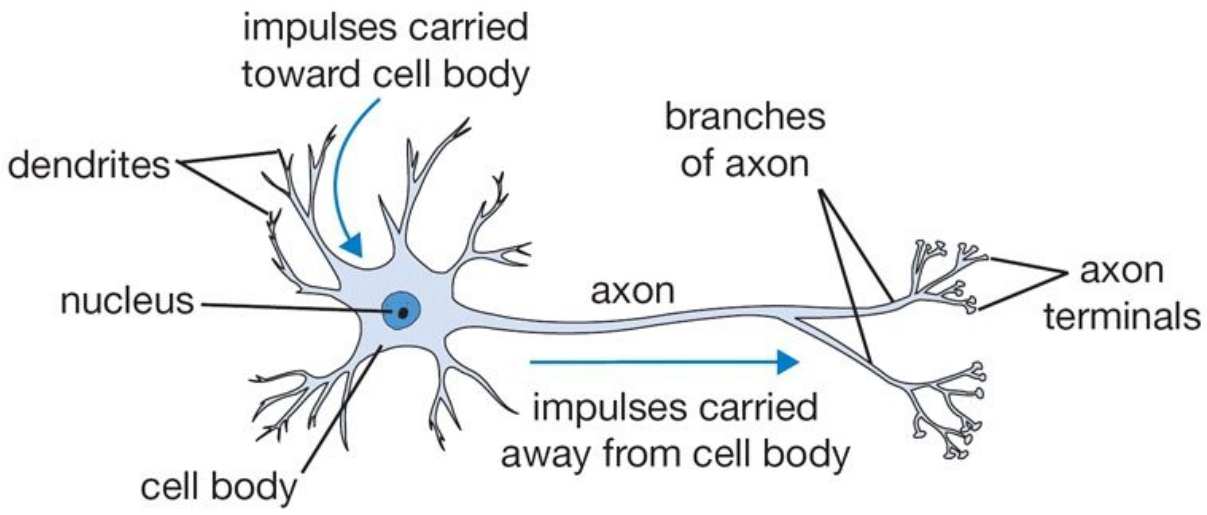
3.1.1 Modeling One Neuron

The area of Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems, but has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks. Nonetheless, we begin our discussion with

a very brief and high-level description of the biological system that a large portion of this area has been inspired by.

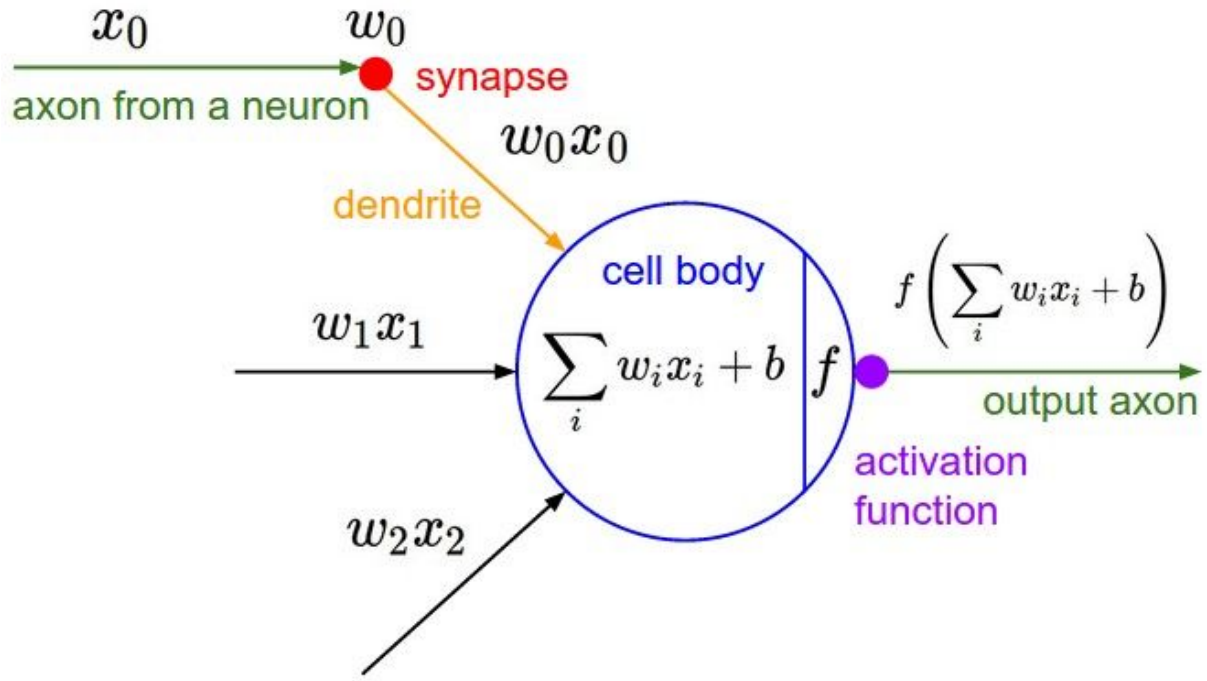
3.1.2 Biological motivation and connection

The basic computational unit of the brain is a **neuron**. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^14 - 10^15 **synapses**. The diagram below shows a cartoon drawing of a biological neuron (left)



and a common mathematical model (right). Each neuron receives input signals from its **dendrites** and produces output signals along its (single) **axon**. The axon eventually branches out and connects via synapses to dendrites of other neurons. In the computational model of a neuron, the signals that travel along the axons (e.g. **x0**) interact multiplicatively (e.g. **w0x0**) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. **w0**). The idea is that the synaptic strengths (the weights w) are learnable and control the strength of influence (and its direction: excitatory (positive weight) or inhibitory (negative weight)) of one neuron on another. In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of the firing communicates information. Based on this rate code interpretation, we model the firing rate of the neuron with an **activation function f**, which represents the frequency of the spikes along the axon.

Historically, a common choice of activation function is the **sigmoid function σ**, since it takes a real-valued input (the signal strength after the sum) and squashes it to range between 0 and 1.



3.1.3 Single neuron as a linear classifier

The mathematical form of the model Neuron's forward computation might look familiar to you. As we saw with linear classifiers, a neuron has the capacity to "like" (activation near one) or "dislike" (activation near zero) certain linear regions of its input space. Hence, with an appropriate loss function on the neuron's output, we can turn a single neuron into a linear classifier:

Binary Softmax classifier. For example, we can interpret **σ(∑iwixi+b)** to be the probability of one of the classes **P(yi=1∣xi;w).** The probability of the other class would be **P(yi=0∣xi;w)=1−P(yi=1∣xi;w)**, since they must sum to one. With this interpretation, we can formulate the cross-entropy loss as we have seen in the Linear Classification section, and optimizing it would lead to a binary Softmax classifier (also known as logistic regression). Since the sigmoid function is restricted to be between 0-1, the predictions of this classifier are based on whether the output of the neuron is greater than 0.5.

Binary SVM classifier. Alternatively, we could attach a max-margin hinge loss to the output of the neuron and train it to become a binary Support Vector Machine.

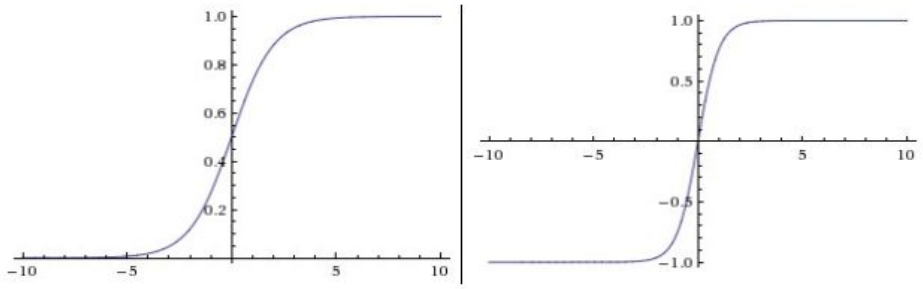
Regularization interpretation. The regularization loss in both SVM/Softmax cases could in this biological view be interpreted as gradual forgetting, since it would have the effect of driving all synaptic weights w towards zero after every parameter update.

A single neuron can be used to implement a binary classifier (e.g. binary Softmax or binary SVM classifiers)

3.1.4 Commonly Used Activation Functions

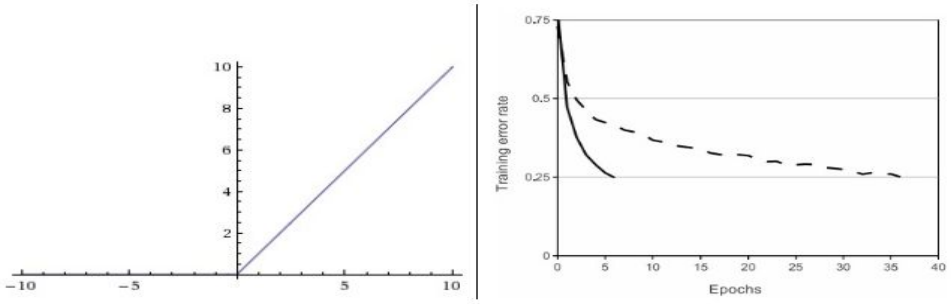
Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. There are several activation functions we encounter in practice:

- Sigmoid Units



Left: Sigmoid non-linearity squashes real numbers to range between [0,1] **Right:** The tanh non-linearity squashes real numbers to range between [-1,1].

- Tanh

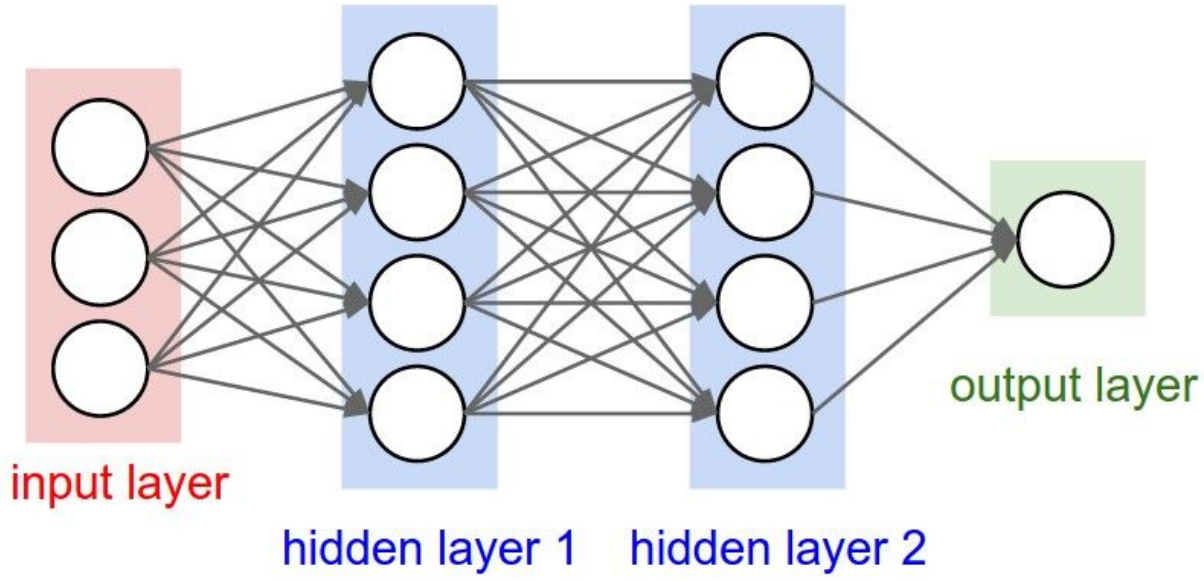


Left: Rectified Linear Unit (ReLU) activation function, which is zero when x < 0 and then linear with slope 1 when x > 0. **Right:** A plot from Krizhevsky et al. (pdf) paper indicating the 6x improvement in convergence with the ReLU unit compared to the tanh unit.

- ReLU
- LeakyReLU
- Maxout
- TLDR

3.1.5 Neural Network Architectures

Neural Networks as neurons in graphs. Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. Cycles are not allowed since that would imply an infinite loop in the forward pass of a network. Instead of an amorphous blobs of connected neurons, Neural Network models are often organized into distinct layers of neurons. For regular neural networks, the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections. Below are two example Neural Network topologies that use a stack of fully-connected layers:



3.2 Deep Convolutional Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

So what does change? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

A ConvNet is made up of Layers. Every Layer has a simple API: It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.

3.2.1 Layers used to build Convnets

As we described above, a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture.

Example Architecture: Overview. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. In more detail:

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small

region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.

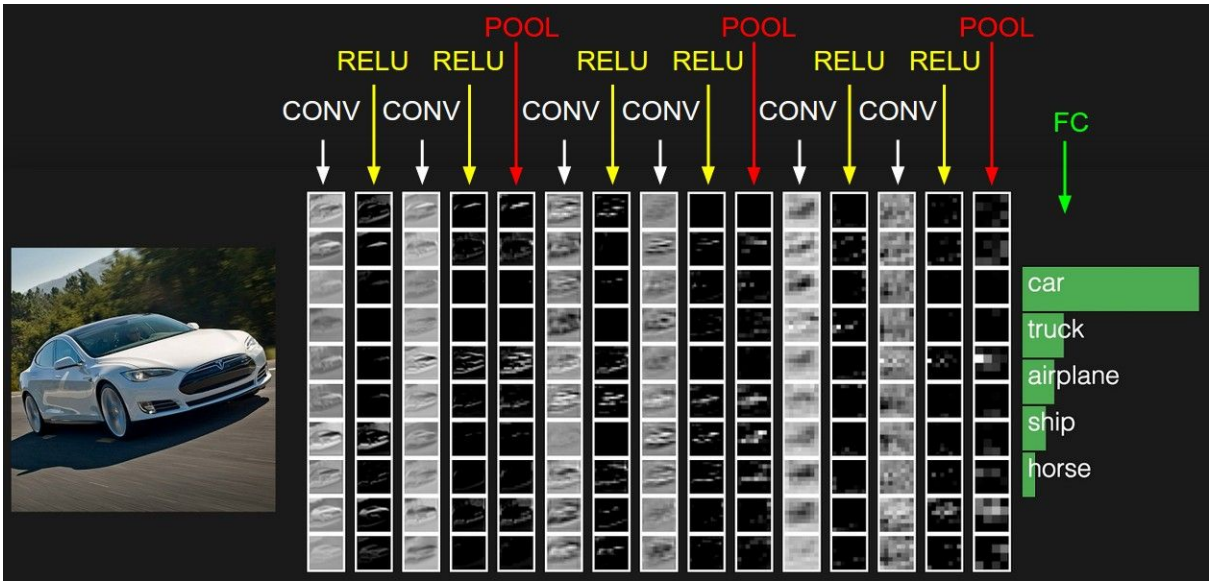
- RELU layer will apply an elementwise activation function, such as the $\max(0,x)$max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image.

In summary:

- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don't)

- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn't)



3.3 Transfer Learning

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest. The three major Transfer Learning scenarios look as follows:

- **ConvNet as fixed feature extractor**. Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features **CNN codes**. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case). Once you extract the

4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

- **Fine-tuning the ConvNet.** The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the ConvNet, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. In case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.
- **Pretrained models.** Since modern ConvNets take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final ConvNet checkpoints for the benefit of others who can use the networks for fine-tuning. For example, the Caffe library has a Model Zoo where people share their network weights.

3.3.1 When and how to fine Tune a dataset?

How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset (small or big), and its similarity to the original dataset (e.g. ImageNet-like in terms of the content of images and the classes, or very different, such as microscope images). Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, here are some common rules of thumb for navigating the 4 major scenarios:

1. *New dataset is small and similar to original dataset*. Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns. Since the data is similar to the original data, we expect higher-level features in the ConvNet

to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the CNN codes.

2. *New dataset is large and similar to the original dataset*. Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.
3. *New dataset is small but very different from the original dataset*. Since the data is small, it is likely best to only train a linear classifier. Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features. Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.
4. *New dataset is large and very different from the original dataset*. Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch. However, in practice it is very often still beneficial to initialize with weights from a pretrained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

Practical advice. There are a few additional things to keep in mind when performing Transfer Learning:

- *Constraints from pretrained models*. Note that if you wish to use a pretrained network, you may be slightly constrained in terms of the architecture you can use for your new dataset. For example, you can't arbitrarily take out Conv layers from the pretrained network. However, some changes are straight-forward: Due to parameter sharing, you can easily run a pretrained network on images of different spatial size. This is clearly evident in the case of Conv/Pool layers because their forward function is independent of the input volume spatial size (as long as the strides "fit"). In case of FC layers, this still holds true because FC layers can be converted to a Convolutional Layer: For example, in an AlexNet, the final pooling volume before the first FC layer is of size [6x6x512]. Therefore, the FC layer looking at this volume is equivalent to having a Convolutional Layer that has receptive field size 6x6, and is applied with padding of 0.

- *Learning rates*. It's common to use a smaller learning rate for ConvNet weights that are being fine-tuned, in comparison to the (randomly-initialized) weights for the new linear classifier that computes the class scores of your new dataset. This is because we expect that the ConvNet weights are relatively good, so we don't wish to distort them too quickly and too much (especially while the new Linear Classifier above them is being trained from random initialization).

3.4 Model Proposed

To solved out image classification problem, we propose placing a Feed Forward classifier network on top of a pre trained network, trained on the Image Net dataset like the VGG16 model.

```
class FFClassifier(nn.Module):

    def __init__(self, in_features, hidden_features, out_features, int_features = 2048, drop_prob=0.5):
        super().__init__()

        self.fc1 = nn.Linear(in_features, hidden_features)
        self.fc2 = nn.Linear(hidden_features, hidden_features)
        self.fc3 = nn.Linear(hidden_features, int_features)
        self.fc4 = nn.Linear(int_features, out_features)

        self.drop = nn.Dropout(p=drop_prob)

    def forward(self, x):
        x = self.drop(F.relu(self.fc1(x)))
        x = self.drop(F.relu(self.fc2(x)))
        x = self.drop(F.relu(self.fc3(x)))
        x = self.fc4(x)

        x = F.log_softmax(x, dim=1)
        return x
```

The above image is an implementation of a Feedforward classifier network using Pytorch in Python. The following is the layer by layer of our deep model with the Layer type, output shape and number of parameters

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 224, 224]           1792
              ReLU-2         [-1, 64, 224, 224]              0
            Conv2d-3         [-1, 64, 224, 224]          36928
              ReLU-4         [-1, 64, 224, 224]              0
         MaxPool2d-5         [-1, 64, 112, 112]              0
            Conv2d-6        [-1, 128, 112, 112]          73856
              ReLU-7        [-1, 128, 112, 112]              0
            Conv2d-8        [-1, 128, 112, 112]         147584
              ReLU-9        [-1, 128, 112, 112]              0
        MaxPool2d-10          [-1, 128, 56, 56]              0
           Conv2d-11          [-1, 256, 56, 56]         295168
             ReLU-12          [-1, 256, 56, 56]              0
           Conv2d-13          [-1, 256, 56, 56]         590080
             ReLU-14          [-1, 256, 56, 56]              0
           Conv2d-15          [-1, 256, 56, 56]         590080
             ReLU-16          [-1, 256, 56, 56]              0
        MaxPool2d-17          [-1, 256, 28, 28]              0
           Conv2d-18          [-1, 512, 28, 28]        1180160

           Conv2d-18          [-1, 512, 28, 28]        1180160
             ReLU-19          [-1, 512, 28, 28]              0
           Conv2d-20          [-1, 512, 28, 28]        2359808
             ReLU-21          [-1, 512, 28, 28]              0
           Conv2d-22          [-1, 512, 28, 28]        2359808
             ReLU-23          [-1, 512, 28, 28]              0
        MaxPool2d-24          [-1, 512, 14, 14]              0
           Conv2d-25          [-1, 512, 14, 14]        2359808
             ReLU-26          [-1, 512, 14, 14]              0
           Conv2d-27          [-1, 512, 14, 14]        2359808
             ReLU-28          [-1, 512, 14, 14]              0
           Conv2d-29          [-1, 512, 14, 14]        2359808
             ReLU-30          [-1, 512, 14, 14]              0
        MaxPool2d-31            [-1, 512, 7, 7]              0
           Linear-32                 [-1, 4096]      102764544
          Dropout-33                 [-1, 4096]              0
           Linear-34                 [-1, 4096]       16781312
          Dropout-35                 [-1, 4096]              0
           Linear-36                 [-1, 2048]        8390656
          Dropout-37                 [-1, 2048]              0
           Linear-38                    [-1, 2]           4098
     FFClassifier-39                    [-1, 2]              0
================================================================
Total params: 142655298
Trainable params: 127940610
Non-trainable params: 14714688
----------------------------------------------------------------
```

CHAPTER - 4

Implementation

For implementing the planned model, many technologies and concepts were used. These include Python, Flask, SQLite, SQLAlchemy, Alembic. There were many reasons behind choosing these specific technologies. For development, Git was used as a version control system.

4.1 Python

Python is a widely used high-level programming language for general-purpose programming. An interpreted language, Python has a design philosophy, which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax, which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.

4.1.1 Features and philosophy

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming. Many other paradigms are supported via extensions.

Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The following points summarize the core philosophy of Python:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. Python can also be embedded in existing applications that need a programmable interface.

4.1.2. Rationale

The features and philosophy match the wide range of applications that are provided by LDCLI. The high extensibility of python has been proven useful for development. The primary audience for LDCLI is physicists, scientists and researchers. The scientific communities are in strong of favor of Python. In addition to that, the predecessor to nova was written in Python as well, so certain ideas and implementations could be taken verbatim.

4.2 Flask

Flask is a Python web framework built with a small core and easy-to-extend philosophy. It is often called a micro framework. “Micro” does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The “micro” in micro framework means Flask aims to keep the core simple but extensible. Flask won’t make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to the user, so that Flask can be everything the user needs and nothing the user does not.

By default, Flask does not include a database abstraction layer, form validation, or anything else where different libraries already exist that can handle that. Instead, Flask supports extensions to add such functionality to your application as if it was implemented in Flask

itself. Numerous extensions provide database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be “micro”, but it’s ready for production use on a variety of needs.

4.2.1 Advantages of Flask

- It's easy to set up
- It's supported by an active community
- It's well documented
- It's very simple and minimalistic, and doesn't include anything the user won't use
- It's flexible enough that you can add extensions if you need more functionality

4.2.2 Rationale

Flask being a micro framework, allows for high extensibility, which again allows a range of applications to be developed within the framework itself, ultimately reducing the complexity of the code.

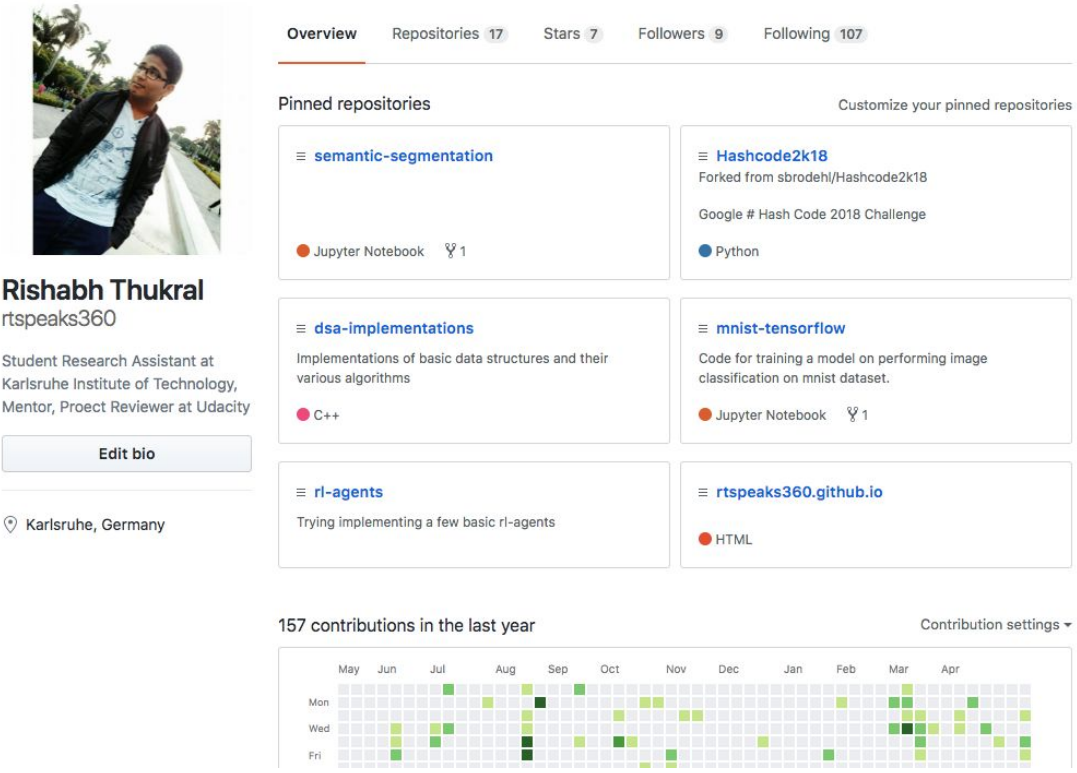
4.3 Github

Using Git for the development was necessary since there are multiple people working on this project and the course of development is 2 years. The product requires the merging of work done by different people at different points of time. Version control was necessary and as git is the most widely used VCS, it would not be much more of a hassle to use git.

Github is a commonly used service that uses git for open source collaboration. Github hosts more than 58 million projects and has more than 21 million active users.

Git (/ɡɪt/[7]) is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development,[8] but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed,[9] data integrity,[10]

and support for distributed, non-linear workflows.[11] Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.[12] Its current maintainer since 2005 is Junio Hamano. As with most other distributed version control systems, and unlike most client–server systems, every Git



directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.[13] Git is free and open source software distributed under the terms of the GNU General Public License version 2.

4.4 PyTorch

PyTorch Original author(s) Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan Initial release October 2016; 1 year ago Stable release 0.4.0 / 24 April 2018; 23 days ago Repository github.com/pytorch/pytorch Written in Python, C++, CUDA Operating system Linux, macOS, Windows Type Library for machine learning and deep learning Website pytorch.org PyTorch is an open source machine learning library for Python, based on Torch,used for applications such as natural language processing. It is primarily developed by

Facebook's artificial-intelligence research group,and Uber's "Pyro" software for probabilistic

programming is built on it.

CHAPTER - 5

Demonstration

5.1 Data Preprocessing and cleanup

To even remotely begin training the model, we first need to have a proper clean data set, for that we first need to make the dataset.

Once you have the data set, it would look something like.

- Data
    - Train
        - 0
            - ***(Images)
        - 1
            - ***(Images)
    - Test
        - 0
            - ***(Images)
        - 1
            - ***(Images)
    - Valid
        - 0
            - ***(Images)
        - 1
            - ***(Images)

5.2 Training the model

Here we train the parameters of the Feed forward classifier network keeping the parameters of the pre trained network frozen.

5.2.1 Hyperparameters used

The network parameters used for the training of the model are as follows:

- Number of Epochs: 15
- Batch Size: 32
- Learning Rat: 0.0001
- Activation Function: ReLU
- Fraction of Data: The division of data into train, test and validation set, was done with a ratio of 101:34:23
- Loss Function: NLLLoss (Negative Log Likelihood Loss)

5.2.2 Training the model(episodes)

The following images show how the model was trained for 15 epochs, its training accuracies, validation accuracies and loss function values.

```
Epoch 1/15
Step : 3
Training loss 0.711
Valid loss: 0.683 Accuracy: 0.573 Val time: 0.848 s/batch
Step : 6
Training loss 0.704
Valid loss: 0.604 Accuracy: 0.646 Val time: 0.881 s/batch
Valid loss: 0.788 Accuracy: 0.354 Val time: 0.800 s/batch
Epoch 2/15
Step : 3
Training loss 0.683
Valid loss: 0.611 Accuracy: 0.698 Val time: 0.869 s/batch
Step : 6
Training loss 0.673
Valid loss: 0.722 Accuracy: 0.438 Val time: 0.936 s/batch
Valid loss: 0.828 Accuracy: 0.438 Val time: 0.890 s/batch
Epoch 3/15
Step : 3
Training loss 0.640
Valid loss: 0.548 Accuracy: 0.802 Val time: 0.854 s/batch
Step : 6
Training loss 0.623
Valid loss: 0.522 Accuracy: 0.688 Val time: 0.812 s/batch
Valid loss: 0.557 Accuracy: 0.615 Val time: 0.775 s/batch
Epoch 4/15
Step : 3
Training loss 0.511
Valid loss: 0.552 Accuracy: 0.781 Val time: 0.792 s/batch
Step : 6
Training loss 0.553
Valid loss: 0.439 Accuracy: 0.854 Val time: 0.759 s/batch
Valid loss: 0.598 Accuracy: 0.802 Val time: 0.811 s/batch
```

```
Epoch 5/15
Step : 3
Training loss 0.480
Valid loss: 0.494 Accuracy: 0.698 Val time: 0.765 s/batch
Step : 6
Training loss 0.506
Valid loss: 0.384 Accuracy: 0.875 Val time: 0.780 s/batch
Valid loss: 0.425 Accuracy: 0.865 Val time: 0.866 s/batch
Epoch 6/15
Step : 3
Training loss 0.476
Valid loss: 0.445 Accuracy: 0.781 Val time: 0.821 s/batch
Step : 6
Training loss 0.495
Valid loss: 0.337 Accuracy: 0.854 Val time: 0.850 s/batch
Valid loss: 0.440 Accuracy: 0.698 Val time: 0.779 s/batch
Epoch 7/15
Step : 3
Training loss 0.457
Valid loss: 0.423 Accuracy: 0.823 Val time: 0.772 s/batch
Step : 6
Training loss 0.488
Valid loss: 0.330 Accuracy: 0.885 Val time: 0.772 s/batch
Valid loss: 0.383 Accuracy: 0.917 Val time: 0.905 s/batch
Epoch 8/15
Step : 3
Training loss 0.429
Valid loss: 0.331 Accuracy: 0.823 Val time: 0.840 s/batch
Step : 6
Training loss 0.408
Valid loss: 0.366 Accuracy: 0.750 Val time: 0.870 s/batch
Valid loss: 0.370 Accuracy: 0.771 Val time: 0.847 s/batch
```

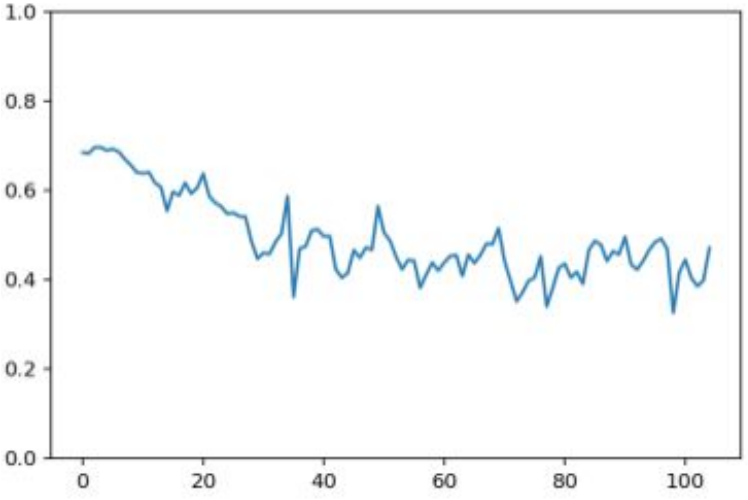
```
Epoch 9/15
Step : 3
Training loss 0.520
Valid loss: 0.262 Accuracy: 0.906 Val time: 0.789 s/batch
Step : 6
Training loss 0.500
Valid loss: 0.272 Accuracy: 0.865 Val time: 0.917 s/batch
Valid loss: 0.339 Accuracy: 0.781 Val time: 0.891 s/batch
Epoch 10/15
Step : 3
Training loss 0.386
Valid loss: 0.330 Accuracy: 0.823 Val time: 0.934 s/batch
Step : 6
Training loss 0.456
Valid loss: 0.285 Accuracy: 0.906 Val time: 0.919 s/batch
Valid loss: 0.286 Accuracy: 0.906 Val time: 0.890 s/batch
Epoch 11/15
Step : 3
Training loss 0.488
Valid loss: 0.397 Accuracy: 0.750 Val time: 0.971 s/batch
Step : 6
Training loss 0.411
Valid loss: 0.292 Accuracy: 0.917 Val time: 0.919 s/batch
Valid loss: 0.257 Accuracy: 0.927 Val time: 0.971 s/batch
Epoch 12/15
Step : 3
Training loss 0.448
Valid loss: 0.275 Accuracy: 0.833 Val time: 0.896 s/batch
Step : 6
Training loss 0.428
Valid loss: 0.489 Accuracy: 0.740 Val time: 0.955 s/batch
Valid loss: 0.479 Accuracy: 0.677 Val time: 0.908 s/batch
```

```
Epoch 13/15
Step : 3
Training loss 0.385
Valid loss: 0.264 Accuracy: 0.938 Val time: 0.943 s/batch
Step : 6
Training loss 0.384
Valid loss: 0.268 Accuracy: 0.917 Val time: 0.915 s/batch
Valid loss: 0.264 Accuracy: 0.938 Val time: 0.950 s/batch
Epoch 14/15
Step : 3
Training loss 0.378
Valid loss: 0.414 Accuracy: 0.750 Val time: 0.903 s/batch
Step : 6
Training loss 0.426
Valid loss: 0.366 Accuracy: 0.854 Val time: 0.938 s/batch
Valid loss: 0.322 Accuracy: 0.927 Val time: 0.961 s/batch
Epoch 15/15
Step : 3
Training loss 0.390
Valid loss: 0.255 Accuracy: 0.927 Val time: 0.895 s/batch
Step : 6
Training loss 0.415
Valid loss: 0.239 Accuracy: 0.875 Val time: 0.935 s/batch
Valid loss: 0.379 Accuracy: 0.781 Val time: 0.944 s/batch
```

5.3 Testing results on the test set

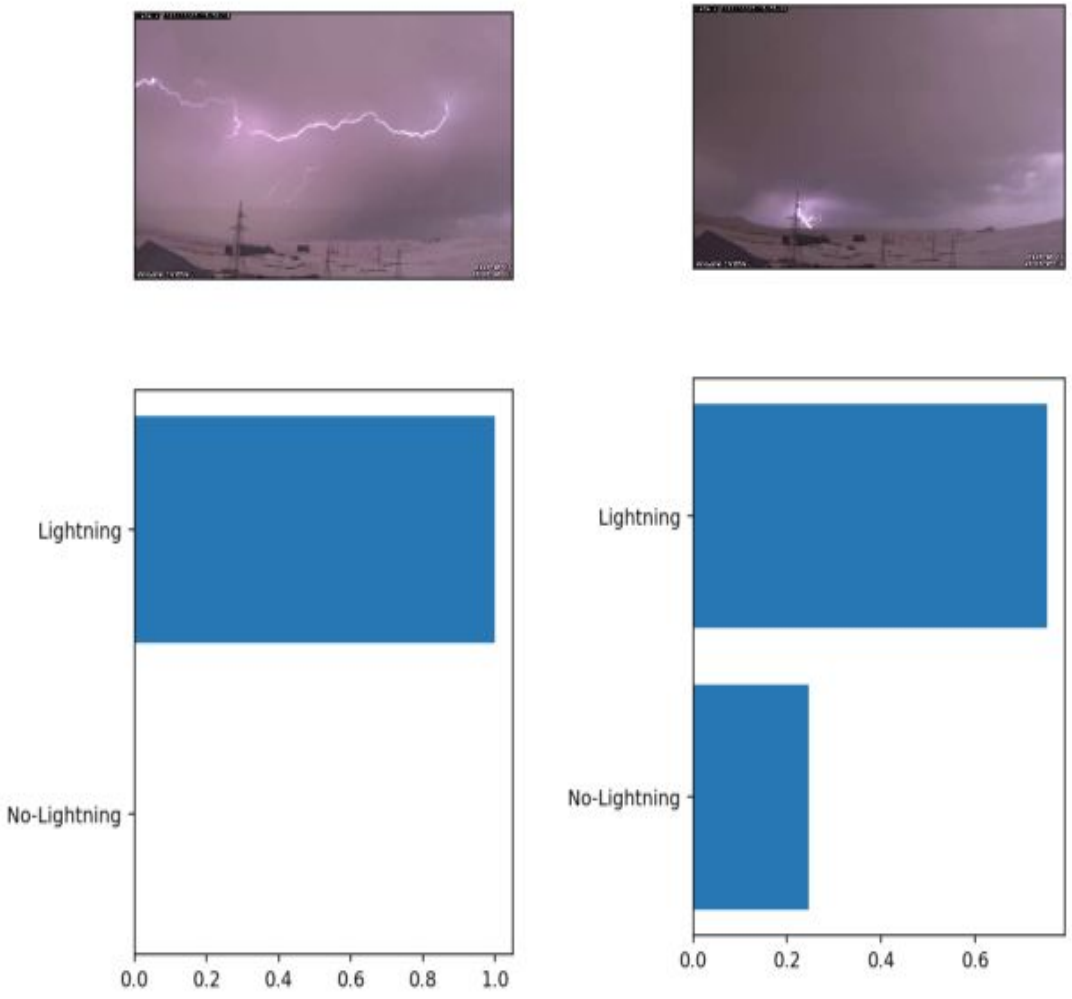
```
model.cuda()
model.eval()
validation(model, dataloaders['test'], criterion, cuda = True)

plt.plot(losses)
plt.ylim([0,1])
plt.show()
```

```
Valid loss: 0.133 Accuracy: 0.793 Val time: 0.674 s/batch
```

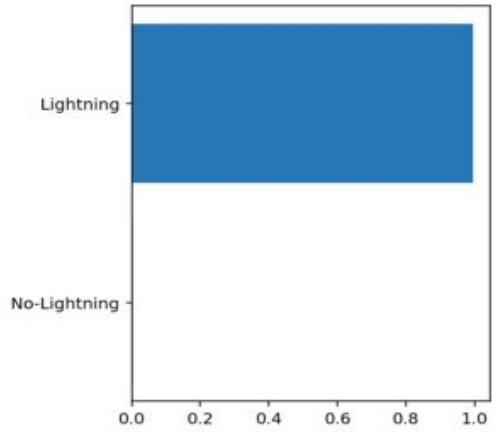


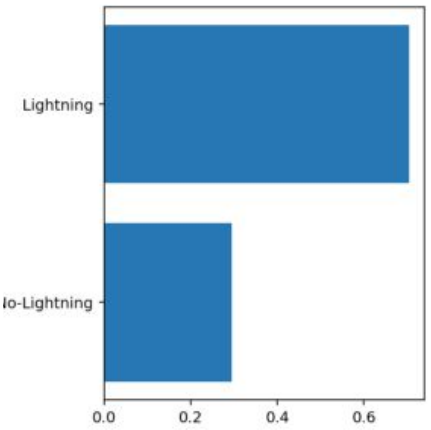
5.4 Results on Various Test images

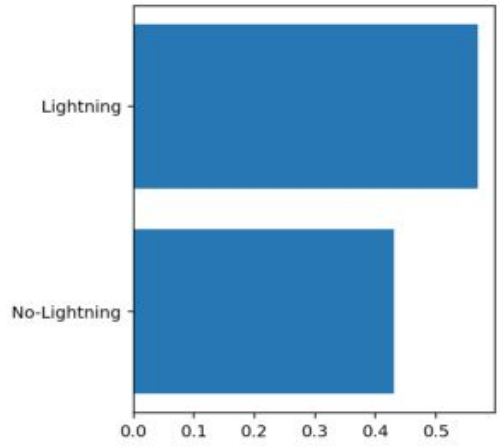
Following are the various images from both categories that we tried the model on with their corresponding result that we got. On the test data, our trained model predicts the probability

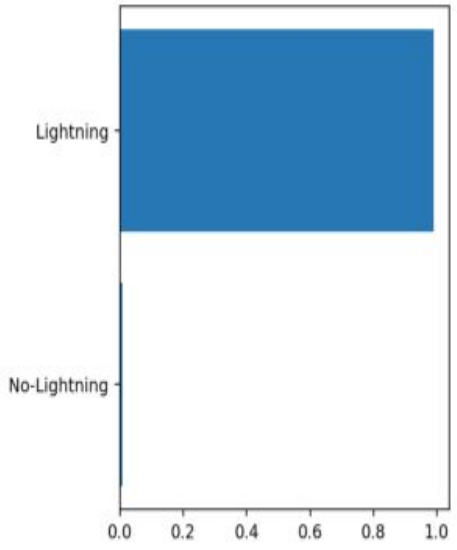


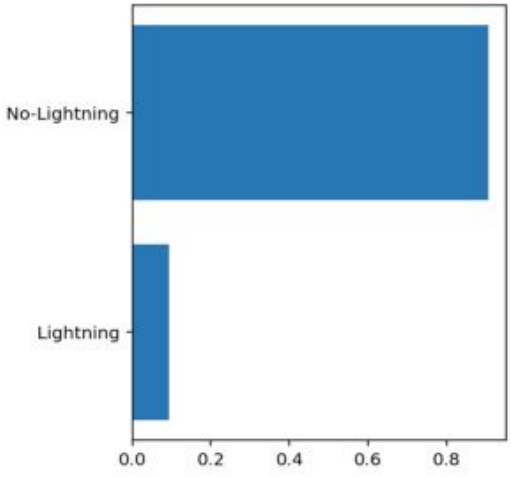
with an accuracy of 79.3%, and that is a good score if we see in terms of accuracy for Conventional Computer vision and Image classification models. Now this devised / trained model can be deployed into production and will predict the results for any number of images with a linear time complexity.

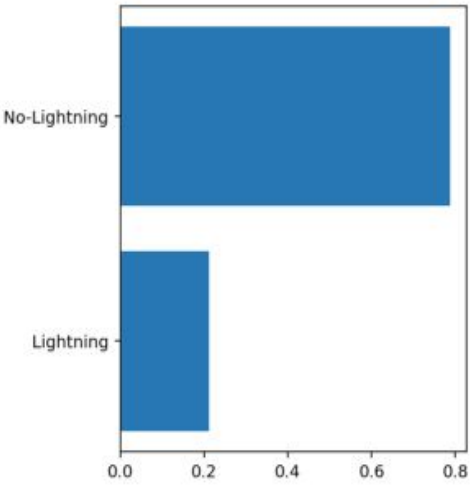

Lightning
No-Lightning
0.0
0.2
0.4
0.6
0.8
1.0

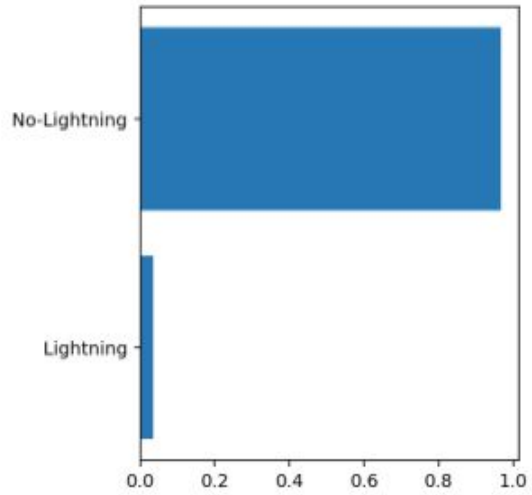

Lightning
lo-Lightning
0.0
0.2
0.4
0.6

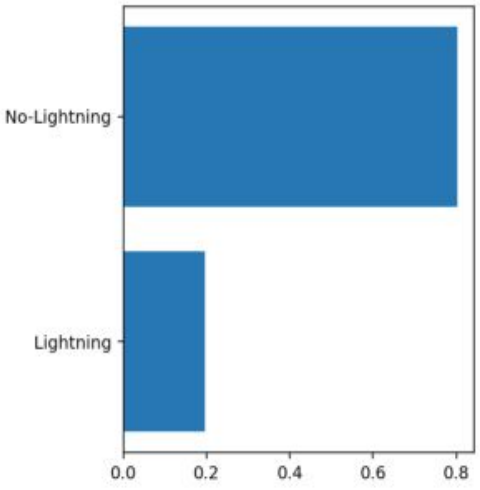

Lightning
No-Lightning
0.0
0.1
0.2
0.3
0.4
0.5


Lightning
No-Lightning
0.0
0.2
0.4
0.6
0.8
1.0


No-Lightning
Lightning
0.0
0.2
0.4
0.6
0.8


No-Lightning
Lightning
0.0
0.2
0.4
0.6
0.8


No-Lightning
Lightning
0.0
0.2
0.4
0.6
0.8
1.0


No-Lightning
Lightning
0.0
0.2
0.4
0.6
0.8

CHAPTER - 6

Conclusion

In the end after carrying out all the experiments, and looking at the results that we have achieved, we can say that Deep Neural Networks can be used to solve the task of computer vision using Convolutional Neural Network, and that too with the great level of accuracy.

As an end product we have a PyTorch model trained on top of VGG net, that can predict results about an image being lightning or not at an accuracy of 79.3%.

CHAPTER - 7

Bibliography

1. Course Notes for CS231n Stanford University
2. Course Notes and content of FAST.AI
3. Research blog of Google brain team
4. https://machinelearningmastery.com/what-is-deep-learning/
5. Udacity AI programing with python Nanodegree Content
6. PyTorch Documentation.
7. Stack overflow
8. Kaggle
9. VGG for providing the pretrained model that we could work on top of
10. IPE KIT for providing the resources to provide the opportunity and resources to get the job done