Classification of Different Magnetic Structures from Image Data using Deep Neural Networks

by

Fahad Ibn Hasib
21141002
Nakiba Farhana Swarna
21341054

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

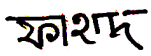
Department of Computer Science and Engineering
Brac University
September 2021

Declaration

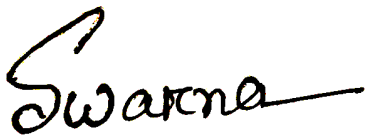
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Fahad Ibn Hasib
21141002



Nakiba Farhana Swarna
21341054

Approval

The thesis/project titled "Classification of Different Magnetic Structures from Image Data using Deep Neural Networks"submitted by

1. Fahad Ibn Hasib(21141002)
2. Nakiba Farhana Swarna(21341054)

Of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 26, 2021.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Ashraful Alam. PhD
Assistant Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam. PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We hereby declare that all the work done in preparing this document beginning from the research to implementation of the proposed methodology is our own. We accept full responsibility for our actions. All the datasets and other external resources that we used have been acknowledged. We have refrained ourselves from using any unethical means in obtaining any of the resources used. We intend to act with morality, fidelity and integrity. We hereby testify that this paper has not been previously submitted in whole or in part by anyone or anytime to any other universities institutions to complete a degree. None of our actions, participation cause any harm to any person, animal or environment on purpose.

Abstract

We apply machine learning, specially deep neural network approaches, to train a new model that can perform an effective classification of ferromagnetic, anti-ferromagnetic, skyrmion, anti-skyrmion and spin spiral configurations via supervised learning and also observe how the pre trained models like VGG16, VGG19, ResNet, Inception behave while solving this problem, draw a pattern from it and suggest path for further improving the model. The problem relies in categorization of Magnetic Configurations amongst many from input samples of simulation data in order to retrieve classified outcome from several different magnetic configurations. The data for the input sample derives from simulations of physical properties of various magnetic configurations. First CNN is used to classify between the images. Image classifications are mostly carried out using neural networks where data is placed in a graphical structure. In addition, the SVM method is applied twice, once with PCA and once without PCA. The proposed model in this research paper can successfully classify amongst magnetic configurations in real time with data obtained from spin-polarized scanning tunneling and Lorentz transmission electron microscopy. In our approach, we used a single deep neural network architecture is classify all five types of magnetic structures. All in all, this is a holistic approach for solving the classification problem of magnetic configuration and taking a step into optimizing the model.

Keywords: Deep Neural Network; Magnetic structure; Image classification; Machine learning; Res-Net.

Dedication

We would like to pay tribute to all the researchers of the world who motivated us to work on this topic. We dedicate our undergraduate thesis to our honorable supervisor Dr. Ashraful Alam Sir. Without you we would never have progressed as far as we are now. We hope that this achievement will complete the dream that you had for us when you chose to give your students the best education you could. Thank you, sir.

Acknowledgment

Starting in the name of Allah, the beneficent, the most merciful.

Firstly, all praises and thanks to the Greatest, Almighty Allah for His blessings throughout the research work to complete our Undergraduate Thesis successfully.

Secondly, we would like to express our deep and sincere gratitude to our honorable supervisor Sir, Dr. Md. Ashraful Alam. PhD, Assistant professor, Dept. of Computer Science and Engineering, BRAC University, for giving us the opportunity to work under this research topic and providing us invaluable guidance. His kind support, sincerity, vision and motivation have deeply inspired us. He helped us whenever we needed. He is the pioneer of this long journey. It was a great privilege and honor for us to work and under his supervision. We are indebted for his dedication.

Finally, we are extremely grateful to our parents for their love prayers and sacrifices for educating and preparing us for the future. With their kind support and prayer we are now on the verge of our graduation. We are also grateful to the department head of the library, officers of the internet laboratory. Computation of this research would not have been accomplished without the simultaneous work of our team members.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convolutional neural network

DNN Deep neural network

PCA Principal component analysis

$ReLU$ Rectified Linear Unit

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Motivation

In the discipline of condensed matter physics, machine learning is utilized to conduct comprehensive and cost effective research and analysis. Frequently, such algorithms are used in fields where data classification on large scale data sets is required, and results are evaluated without human intervention. When it comes to figure out how to solve molecular structures, materials and their interfaces play a vital part in the area of physics, chemistry, and material science. However, application of machine learning in material science is still in its budding stage. Therefore, categorizing various phases and transitional stages of matter is always a key issue in the study of condensed matter physics. But then again, if we manage to successfully apply machine learning (Specially deep learning) techniques to figure out magnetic configuration structure of materials without using arduous and expensive techniques like LTEM, cost of research and speed of discovery of new characteristics of materials would be reduced heavily. This algorithm can efficiently identify the magnetic structures without any tiresome microscopy technique. Even if we couldn't reach that level, if we can reduce the possibility of characteristics through this technique, which would also be of great help towards research in material science and related fields. All in all, such an attempt would grant us a chance to be pioneers of a field of knowledge which is still in its infant stage.

1.2 Problem Statement

Nowadays, computer algorithms can gain an understanding of data and forecast the outcome using machine learning. Almost all of the time, such techniques are used in domains where data categorization must be conducted on large scale data sets and results must be evaluated in the absence of human assistance. Classifying different phases and intermediate stages of matter is always an important subject in the field of condensed matter physics. The task of identifying between several phases and configurations is extremely difficult using conventional methods. Consequently, this field requires machine learning to aid in the classification of many- body stages of classical and quantum models. In order to classify, principal task remains of distinguishing between various magnetic structures whether it can be distinction of phase or configuration of the magnetic structure. Several properties must be considered

into while identifying a magnetic structure. Hence to create a data set of various properties that fundamentally affect discovering discernment among ferromagnetic, anti-ferromagnetic, skyrmion, anti-skyrmion and spin spiral configurations.

1.3 Objective

Deep learning, when applied to data science, can offer better and more effective processing models. Its ability to learn unsupervised data continuous improvement in accuracy and outcomes.Deep Neural network is highly scalable due to its ability to process massive amounts of data and perform a lot of computations in a cost- and time-effective manner. As a result, we chose to use supervised learning to use Deep Neural Network methods to effectively classify ferromagnetic, anti-ferromagnetic, skyrmion, anti-skyrmion, and spin spiral topologies. The data for the input sample comes from simulations of physical characteristics of various magnetic configurations. Our primary objective is to propose an optimized DNN model that will perform better than pre-trained models. We aim to build this architecture in a cost-effective manner and also faster in speed. After that, we will analyze the results of our proposed model with those pre-trained models.

1.4 Literature Review

Machine learning techniques were used to discern between skyrmion, paramagnetic, ferromagnetic and spin spiral phases by Iakovlev, Sotnikov and Mazurenko (2018) [13]. They also discerned transitional states in two-dimensional materials at different temperature and in the presence of various magnetic fields.
For making a quantitative analysis, they took two distinct ways. Of the said two, first one is neural network with a single-hidden layer. For the second way, they used minimum distance approach.
Unfortunately, they couldn't show an exact difference between skyrmion and spin spiral phases using the first approach. They achieved a non-co-linear configuration coming from the Monte Carlo simulation by using a feed-forward network on magnetic skyrmion and spin spiral states. k-NN and nearest centroid classifier were used for the minimum distance method.
Big skyrmion, high temperature spin spirals, ferromagnetic, spin spiral, skyrmion and paramagnetic phases were in the training data set. Complex and mixed ferromagnetic-skyrmion and skyrmion-spiral configurations were the outcome of successful quantitative classification which is unprecedented work in this field of research. Even though k-NN achieved very low accuracy in most of the cases, it achieved 100 percent accuracy for ferromagnetic and paramagnetic phases. Many layers of convolution neural network associated with several layers of neural network were used by them. They mainly researched on feeding in a lattice model made of Heisenberg, DM interaction and Zeeman term.
Their model could successfully predict both the mechanical quantities of the input configuration such as chirality, magnetization of the structure as well as physical quantities such as temperature and magnetic field at which images were generated.

In addition to that, they train their network using three different configurations consisting of x, xy and xyz components of magnetization and yet got no differences among the accuracy of the three types of training and testing data sets.
On the image classification field Karen Simonyan and Andrew Zisserman (2015) proposed pretrained model for generic image classification [10]. Then later Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun (2015) further optimized on the approach [8]. Lastly, Christian Szegedy and Vincent Vanhoucke and Sergey Ioffe and Jonathon Shlens and Zbigniew Wojna they rethought the architecture approch in their article [11]. But all of their works is on generic image recognition problem and computer vision, not specifically tailored for our problem.

1.5 Thesis outline

Chapter 1: Introduction.

Chapter 2: Fundamentals of Magnetism

Chapter 3: Brief description of main workflow and its components.

Chapter 4: Discussion of result and findings on proposed model.

Chapter 5: Discussion of result and findings on pre-trained models.

Chapter 6: Conclusion on our research and future expectations.

Chapter 2

Fundamentals of Magnetism

2.1 Magnetism

Magnetism has facilitated the development of numerous new inventions. It contributes significantly to the advancement of technological devices such as data storage, among other things. Furthermore, the world is now producing massive amounts of data on a daily basis. We must keep up with the rapidly increasing amount of data while also managing it by storing and processing it in smaller storage devices[40]. As a result, hard drives are extremely vital in the field of computer science. We also need to understand magnetism to gain a better grasp of what happens inside hard disks. Therefore, hard drives rely on magnetism to store data in a layer of magnetic material beneath the surface of the platter of a conventional hard disk. Small magnetic domains are formed in the magnetic layer, allowing information to be stored in this substance. Magnetic domains are regions of the material that have the same direction as the magnetic moment. The generation of magnetic moments in ferromagnetic materials and magnetism are both derived from electron spin [30].

2.2 Magnetic Configurations

2.2.1 Topological and Non-topological structures

Topological structures, also known as topologically-protected structures, are structures that are related with topological invariants, meaning that they do not change when exposed to external physical circumstances like external magnetic fields, high pressure, or temperature. Antiskyrmions and skyrmions are examples of such structures. Non-topological or non-topologically-protected structures, on the other hand, are those that are susceptible to external physical changes. Ferromagnetic, antiferromagnetic, and spin spiral configurations are examples of such structures.

2.2.2 Ferromagnetic Properties

A ferromagnet is a solid substance that exhibits immediate magnetization even when it is not exposed to a magnetic field [7]. These materials have very high magnetic properties, despite the fact that all materials are magnetic to some extent. Magnetic domains are formed when atomic dipoles are consistently aligned in ferromagnetic materials that do not exceed their Curie temperature, resulting in a non-zero magnetic field. According to the Ising model, ferromagnetism is caused by an interaction between the spins of certain electrons present in the atoms of the forming crystal [1]. Ferromagnets have resulted into the discovery of compasses, electromagnets and generators that are a quintessential part of today's world [6].

2.2.3 Anti-Ferromagnetic Properties

Anti-ferromagnetism is a type of magnetism in which nearby particles align themselves in opposite ways to counterbalance the overall magnetism. When the temperature falls below a crucial level, known as the Neel temperature, this alignment occurs instantly [7].

2.2.4 Skyrmions

Skyrmions have received a lot of attention in recent years, and numerous studies have been conducted on the issue [29]. Ferromagnetic materials have been the primary material used in the construction of storage devices, however due to size constraints, no more developments can be made. As we construct magnetic hard disks, for example, nanoscale ferromagnetic particles retain all of the data, but their magnetic states degrade when they are shrunk below a certain size. This results in data loss. Magnetic skyrmions can be used to tackle this problem [1]. They are basically considered as knots in a magnetization distribution with major topological properties [14].
In chiral magnets, the Dzyaloshinskii–Moriya interaction stabilizes the skyrmion, while in thin film magnets, the dipolar interaction stabilizes the skyrmion [12] .Skyrmions are said to uphold topological protection because erasing a skyrmion would require global amendment of the system [38].They maintain great stability when utilized in the construction of skyrmion-based storage systems due to their topological protection features. [40], [14]. The spins in skyrmions are organized in such a way that the spins at the edges are in the opposite direction to spins in the centre [28].

2.2.5 Antiskyrmion

Antiskyrmions are microscopic structures having topological protection, similar to skyrmions. The topological charge of an antiskyrmion, on the other hand, is the total opposite of that of a skyrmion. Antiskyrmions also have more advantages than skyrmions because of their better stability due to dipolar interactions [22]. In chiral magnets, antiskyrmions are unstable and disappear swiftly, whereas in dipolar magnets, they are metastable and survive [12].

2.2.6 Spin-Spiral Phases

Another sort of magnetic configuration is a spin spiral, in which the magnetic moment of an atom is spun by a predetermined angle from one unit cell to the next in a given direction. The rotation axis can be described as a global z-axis when the spin-orbiting coupling is absent. After that, the magnetization direction we get is -

$$\hat{e}^{n,\mu} = \begin{pmatrix} \cos\left(\mathbf{q}\cdot(\mathbf{R}^n + \boldsymbol{\tau}^\mu) + \alpha^\mu\right)\sin\theta^\mu \\ \sin\left(\mathbf{q}\cdot(\mathbf{R}^n + \boldsymbol{\tau}^\mu) + \alpha^\mu\right)\sin\theta^\mu \\ \cos\theta^\mu \end{pmatrix}$$

where Rn is the lattice vector which basically directs from origin to unit cell n, µ is the basis vector of atom µ in unit cell, q is the spin-spiral vector, µ is the angle between magnetic moment of atom, µ and rotation axis. µ is a phase shift that is dependent on atoms [30], [4].

Chapter 3

Proposed Methodology

3.1 Design of main workflow

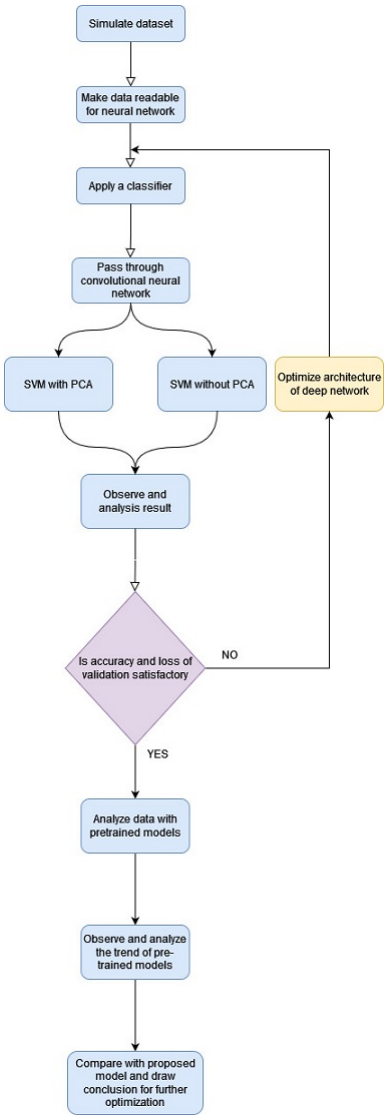


Figure 3.1: Main workflow diagram

Figure 3.1 shows the overall work plan that has been executed throughout the research period. We selected five Magnetic configurations to detect-ferromagnetic, anti-ferromagnetic, skyrmion, antiskyrmion and spin spiral. To begin with, data sets were created because they were unavailable on repositories. Data sets are preprocessed once they are generated.The data sets will next be subjected to a classifier. We saved images, loaded data and proposed our own faster, cheaper, optimised single DNN model architecture. We implemented CNN layer by layer. Until we get satisfactory performance in both accuracy and loss, we reiterated the architecture. Finally, we apply the dataset on some pre-trained models, observe their characteristics and prepare a comparison among them.

3.2 Our Approach

Since the problem we wanted to solve at the core was an image classification problem, we had to generate images. In other words, we needed to generate images of different magnetic configurations with enough small variations in each one of them so that we can detect a slightly changed or deformed phase. We needed mathematical formulations of all the magnetic configurations. We had to translate them into code and use plotting to generate our data set. After that, we prepared our dataset for DNN and implemented it layer by layer. PCA and SVM algorithm are used as classifiers and generate result that is expected. Until we get satisfactory result in both accuracy and loss we would reiterate the deep neural network model architecture.
After observing a satisfactory performance,, so that we can have a proper direction for comparison with our proposed model. Lastly, we made a comparison between the proposed models and determined some areas for further improvement and optimization.

3.3 Dataset Generation

In the subject of condensed-matter physics, machine learning is being more widely employed, and its influence is being felt in practical applications [27], [15]. Because it translates into the transferability and universality of the output model, the validity of the machine learning technique is totally reliant on the data and how it was collected.
The availability (or lack thereof) of data sets is one of the problems of employing machine learning in this sector. Because of the uniqueness of our method, we were unable to locate a data set that met our requirements. We couldn't collect data from physical observations ourselves because we didn't have access to an LTEM or equivalent microscope equipment.
The problem was addressed by simulating native data using magnetic configurations' quantum properties. This approach has been utilized in the literature previously and is far superior in many circumstances, becuase- [21],[33]

1) numerous 'exotic' magnetic phases or excitation of matter like skyrmions are

harder to generate physically in materials,
2) it is challenging to manipulate these phases at room temperature [23] and
3) we might not able to create as many variations in them physically because of this challenge.

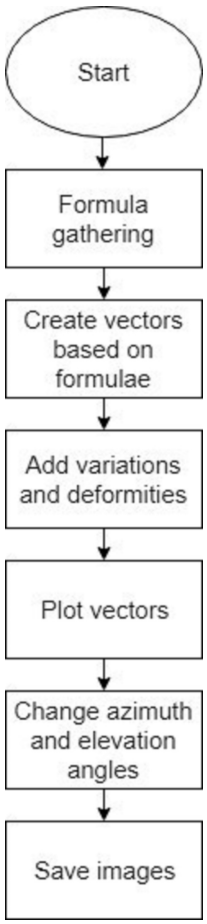


Figure 3.2: Dataset generation workflow

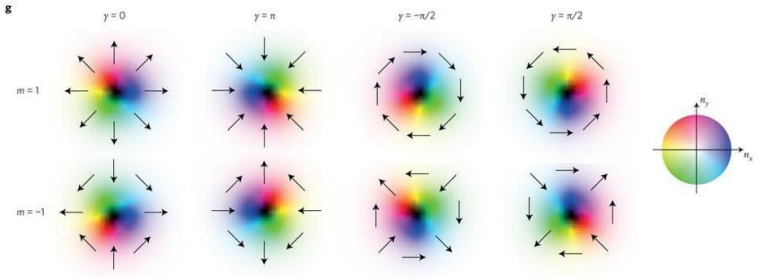


Figure 3.3: Using different gamma and m = (-1, 1) we generated different configurations

Figure 3.2 shows the method through which the data collection was compiled. To get a fundamental concept, all of the structural formulas for the magnetic configurations of interest were initially gathered. These equations were then used to construct vectors. Variations and deformities have been introduced to the equations by altering a few variables. Following that, the vectors were plotted. To give more diversity to the pictures, the azimuth and elevation angles were also varied. Finally, the photos we generated were saved.

3.4 Approach of Image Generation

All of the magnetic configurations' mathematical formulations were previously known to us. To create our data collection, we had to translate them into code and utilize charting. All variables had to be initialized initially, according to the basic code structure. Then we made a loop that iterated over all of our lattice's x, y, and z locations. Then for each position we would calculate the x, y, z and s_x, s_y and s_z values using formulations of each different magnetic configuration [16]. We may raise or reduce the number of individual simulated spins by adjusting the number of spins in each direction of the rectangle. The paragraphs that follow detail how various configurations were created.

3.4.1 Skyrmion and Anti Skyrmion

Skyrmion and anti-skyrmion are similar in the sense that only their voracity or 'm' changes mass. For every position in the lattice, we had to calculate the value of Φ in equation [7] :
$\varphi(\phi) = m\phi + \gamma$
We would get this value with m =+1, m= -1 and different γ values. What is important here is the 'm' value, when we keep the 'm' value +1 we get skyrmions and when we keep the 'm' as -1 we get anti-skyrmions as demonstrated in Fig 4.3 [30]. The helicity value was chosen as 0, π, -$\pi/2$ and $\pi/2$ for different variations. Lastly, we would give as input which would the positional argument of x, y which would be turned into a single value θ $\ni$-π<θ<π and, for some r > 0
[22]

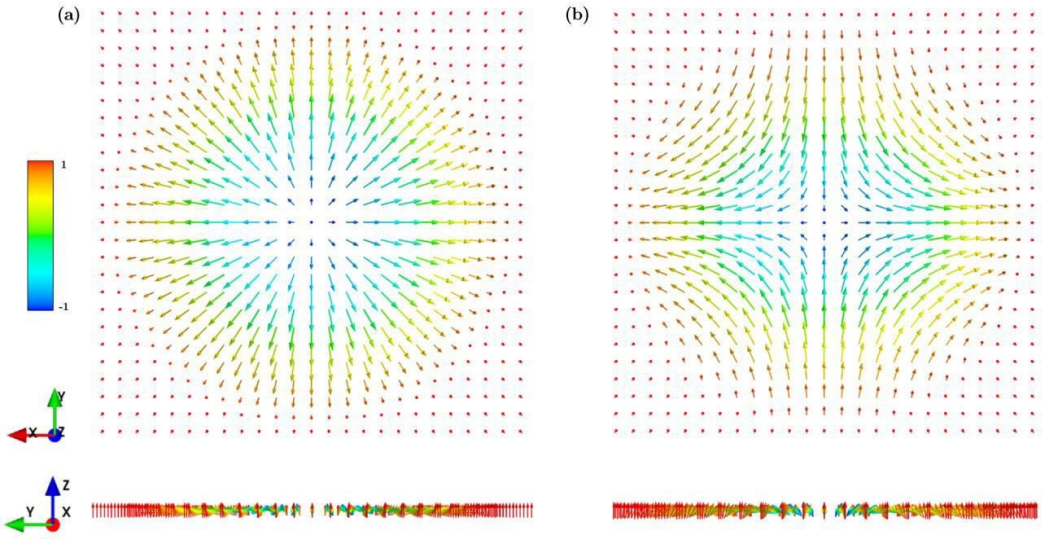


Figure 3.4: Images shows (a) skyrmion with m=+1, =0 and (b) anti skyrmion with m=-1, =0. The color bar represents the direction of the spins. At the top we have a top view and at the bottom we have a cross-sectional view of the lattice.

We had to introduce some deformities in the magnetic configurations in order to obtain a representative data set of real magnetic interactions. We did it in a straightforward manner by multiplying each vector component by a number between 0 and 1. This would generate a deformed image by giving each individual spin a different random value. This added to the dataset's robustness and universality.

3.4.2 Spin Spiral

Spin spiral is a type of magnetic configuration where the magnetization keeps rotating throughout the surface by a constant angle.[3] From equation,

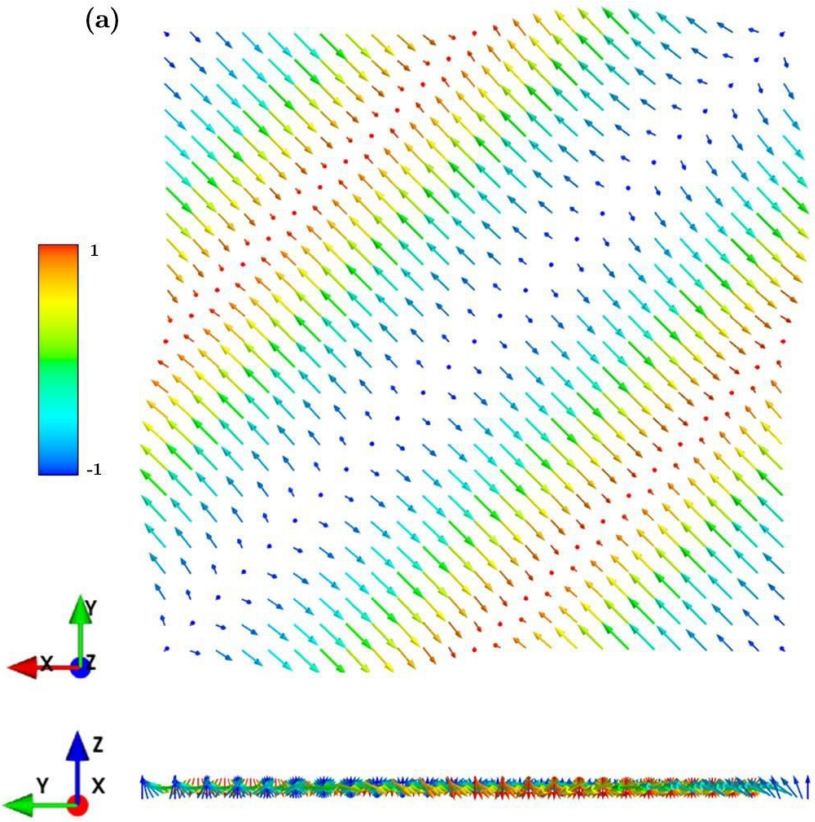


Figure 3.5: Images shows a spin-spiral. The color bar represents the direction of the spins. At the top we have a top view and at the bottom we have a cross-sectional view of the lattice

The numeric values of three components of the directional vector may be obtained. The direction of the vector with respect to the x and y axes is indicated by the θ^{μ} value. We utilized a 45° as the value of θ^{μ} in our produced pictures. The value of $q(R^n + \tau^{\mu})$ is essentially the first spin's beginning angle. Throughout the spins, we increase the value of α^{μ} to get the "swirling" form. This is the angle at which the spins rotate at all times. We utilized different numbers for both of them while creating our data collection so that we could get multiple variants. Fig 3.5 shows the end result. To produce random deformities, we multiply each vector component by a random value from (0, 1), similar to how deformities in skyrmions were formed. [7]

3.4.3 Ferromagnetic and Anti-ferromagnetic

Magnets having evenly distributed atomic dipoles are known as ferro magnets. For ferromagnetic setups, the procedure is simple. The directions of all the vectors would be the same. By altering the direction of the vector, we were able to add some variety, but the basic rule remained the same: all vectors would have the same direction. In antiferromagnetic pictures, however, the value of the s_z changes from +1 to -1 depending on the surrounding spin. [16]. Deformities in the range of were introduced in the same way as in prior setups (0,1). This change in direction was

reflected in the code, which was written in a straightforward manner. The results are in Fig 3.6 [6]

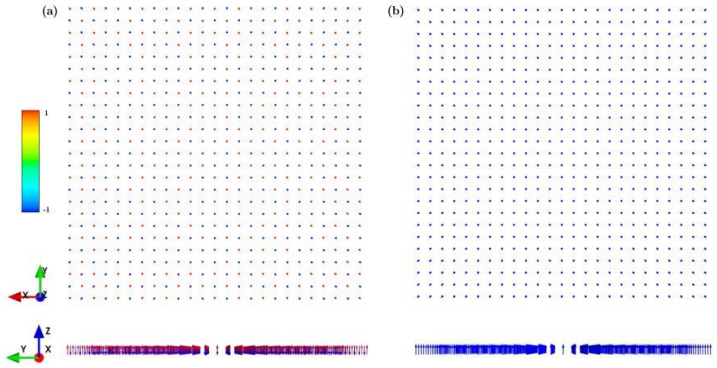


Figure 3.6: Images shows (a) Anti-ferromagnet and (b) Ferromagnet. The color bar represents the direction of the spins. At the top we have a top view and at the bottom we have a cross-sectional view of the lattice.

3.4.4 Plotting vectors

Finding a computer language and library that could plot vectors that looked like a real scan was one of the hurdles in creating the data collection. We tried a few other programming languages and discovered that python and a library named "mayavi" were the best fit for our needs. We had to plot once we got all of the data for all of the vector components. The mayavi "quiver3d" method was given with the parameters u, v, w, which were vector components in various axes, and x, y, z positional arguments, which were the locations of each individual vector in the lattice. [30] Another point worth mentioning is the hue of the various spins. This is significant because the color of the spin aids in determining the direction of each spin. We have to utilize a color by scalar technique to do this, where the spins are colored based on their 'w' vector component [30].

3.4.5 Saving Images

All that remained was for us to save the images. We used a simple loop for this. To preserve the arrangement from different angles, the loop iterated through several azimuth and elevation values. This shift in perspective, combined with the previously described deformations, resulted in a fairly robust data set. We organized all of the photos into subdirectories for each category using the "savefig" feature [16].

3.4.6 Loading data

To begin, an array for photos and another for labels are established, with images and labels put in the appropriate arrays. The images are scaled to 255 x 255 and designated as Gray-scale when they are loaded into the image array.

3.5 Prepare Data for DNN

As data are in image files, it is not readable for Deep Neural Network (DNN). To make it analyzable for DNN, it is needed to be converted into suitable array format for a particular type of neural network. For example, CNN have to be given a 4d array as input **TDS_CNN_input**. Using OpenCV library of python we could easily convert the images into desired array format with pixel values.

3.6 Applying Classifier

We opted CNN along with PCA as our classifiers, at first with SVM and subsequently without SVM. The following is a quick overview of SVM, PCA and CNN -

3.6.1 CNN

A convolutional neural network (CNN) or (ConvNet) is most commonly used in deep learning to evaluate visual pictures. Since the amazing findings were released on the object identification challenge known as the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012, the convolutional neural network (CNN), a family of artificial neural networks, has become a dominating approach in computer vision problems. [18], [9].
CNN uses numerous building blocks like as convolution layers, pooling layers, and fully connected layers to learn spatial hierarchies of data automatically and adaptively through backpropagation [26].When it comes to tasks like image identification, CNN contains at least one convolution layer, which is meant to minimize computational complexity. [38]

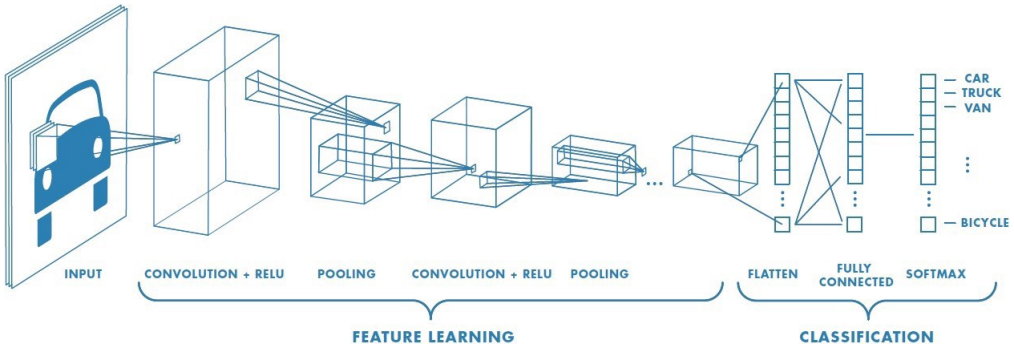


Figure 3.7: Images shows a a neural network containing multiple convolutional layers [25]

Input Layer

In the input layer, there is an image. The computer converts the image into a matrix, which is an array of pixels, depending on its size and kind. In a grayscale picture, there is only one pixel. An RGB image, on the other hand, has three color planes and three pixels as a result. Color spaces, such as HSV, CMYK, and others, are also available in a range of sizes [19]. As a result, a Convolutional layer is necessary to decrease the matrix size and generate a generic form with key properties that is

easier to process. This may also be used to build a network that isn't restricted to a particular picture kind and can be expanded to a larger data collection.

Convolutional layer

The Convolutional layer is the first layer through which the matrix passes. The feature detector is a smaller matrix or filter that travels over the input matrix starting from the top left. [19] The filter's job is to multiply its value by the pixel values in the picture. These multiplications are then added together to get a single number. If the stride is 1, the filter will keep repeating this by moving 1 unit to the right each time. A smaller matrix known as a feature map is obtained when the filter goes over all locations. The feature map obtained as a result of the dot product of an input matrix and a 3x3 filter is shown in Figure. [23]
The input matrix will be in a dimension of W1 × H1 × D1 where W1 is the width, H1 is the height and D1 is the depth of the matrix. After convolution the output produced would be $W_2H_2D_2$ where[20]:

$$W_2 = (W_1F + 2P)/S + 1 \tag{3.1}$$
$$H_2 = (W_1F + 2P)/S + 1 \tag{3.2}$$
$$D_2 = K \tag{3.3}$$

where K is the number of filters, F is the size of filter, S is the stride size and P is the amount of zero padding.
We lose a lot of characteristics with this method, but we don't lose the crucial ones; instead, we preserve the ones that will be needed in future layers. The feature maps we acquire from these methods contain essential features that are required for the convolutional neural network to detect and categorize various sorts of pictures (CNN). [20]

Non-Linearity

This layer is added after every Convolutional operation. The objective of this layer is to add an activation function that would bring non-linearity in the images. There are several activations functions such as sigmoid, tanh, ReLU,Leaky ReLU, Parametric ReLU and softmax.[17]. Without non-linearity, the network can not model the class. label efficiently.

Pooling Layer

A pooling layer is the following layer. The pooling layer performs a max-pooling operation. Pooling is used to extract the most important characteristics while removing the majority of the undesirable ones. As a consequence, you'll have a smaller feature map that's easier to work with. This is the same as randomly selecting activation based on a multinomial distribution at the time of training. Max Pool, Sum Pool, and Average Pool are three different forms of pooling. [17]. As a result, the layer accepts feature maps created by the convolution layer as input and outputs a pooled feature map.

3.6.2 SVM

The goal of the support vector machine (SVM) method is to categorize between points by finding a decision boundary in an N-dimensional space (where N is the number of features).

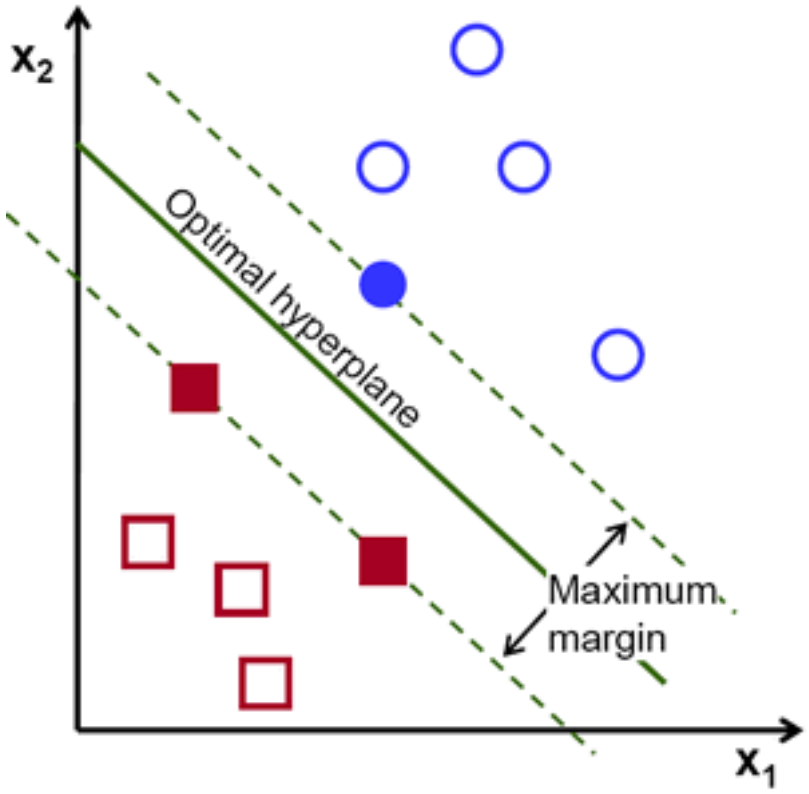


Figure 3.8: Figure showing an optimal hyperplane [20]

A hyperplane is used to establish a decision boundary between the two classes. The hyperplane is used to divide the classes. The hyperplane's dimension is determined by the number of features. If there are two input features, for example, hyperplane is a simple line. If the number of input features is 3, the hyperplane becomes a two-dimensional plane. The hyperplane becomes more complicated as the number of features grows. There are numerous potential hyperplanes, but SVM tries to identify an optimum hyperplane with the greatest margin distance between the hyperplane and data points on both sides. The optimal hyperplane is drawn in the second grid on figure 3.8.Support vectors are data points that are closer to the hyperplane and have an effect on the hyperplane's location. In the second grid on figure 3.8, the filled circle and squares are the support vectors [20].

3.6.3 PCA

When it comes to very large data sets, Principle Component Analysis (PCA) is a method that is used to reduce dimensionality [24]. This is accomplished by converting to a new collection of uncorrelated variables known as principal components (PCs), which are arranged so that the first few maintain the majority of the variance contained in all of the original variables [5]. PCA is used to display the data and speed up the machine learning algorithm. [2] It is accomplished by reducing a variable set to a smaller one that contains the most information from the large data collection. [39].

3.7 Implementation of Proposed DNN Model

From previous research in this field [30], we have seen use of neural network and a approach of one vs all to classify all the image with separate models. To take it another step further. We needed to implement a single model which is capable of classifying all the image classes. In the following sections the implementation of the model is described.

3.7.1 Architectural decisions

We started with 2 convolutional layers with each 64 nodes followed by a max pooling layer. After that, there are 2 fully connected layers which each has 128 nodes. We started with such simple configuration so that we could understand the problems and places to optimize faster and easier as it becomes increasingly complex to understand models with higher complexity. We decided on keeping the learning rate constant at $5 * 10^{-5}$ to observe how the DNN would behave. Lastly, we chose to do 30 epochs to see trend of behavior and also to not take too less epochs and hitting local minima or maxima and not take too much and cause over-fitting.

3.7.2 Revision

After multiple revisions, we decided on 3 convolutional layer, first layer having 32 nodes, second layer having 64 nodes, 3rd layer having 128 nodes. Pool size is 3x3 in each layer. Each layer has stride of 2. Each convolution layer has a maxpooling layer after them. Each maxpooling layer has pool size of 2. After all the convolution layers a flatten layer is used to make the input flattened out. All the layers use ReLU as activation function. After that there are 5 fully connected layers in (64, 64, 128, 256, 512) node configuration. All of the layers use ReLU as activation function. Lastly, there is the output layer with 5 modes as there is 5 classes of image in this problem. Softmax function is used as the activation function for this layer. Batch normalization is used after every layer to avoid over fitting. Bias is also eliminated to further improving the performance.

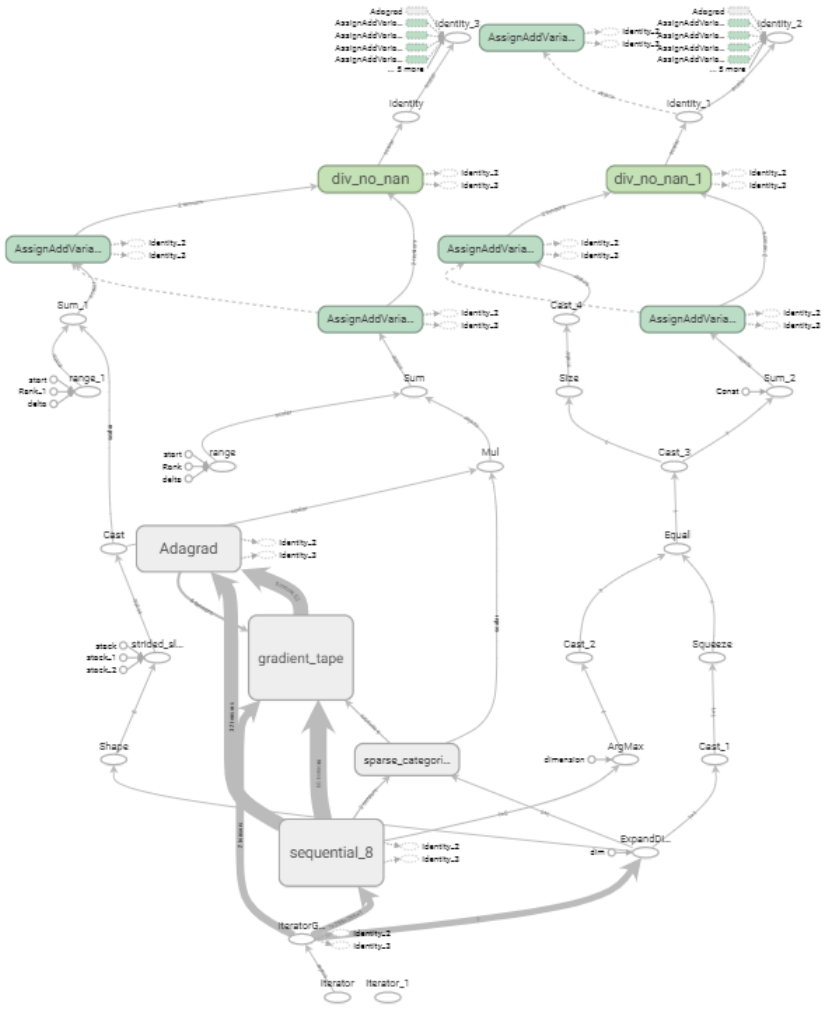


Figure 3.9: Figure showing a graph representation of the proposed final DNN model

Chapter 4

Proposed Model Result Analysis

4.1 Primary Model result

Primary result was good at initial glance. The model hit very high training accuracy very early, around 10 to 15 epoch. Then the validation accuracy score came in too low for the test data set, which is a very obvious sign for over-fitting. Thus, it called for extensive re-iteration and optimization of the model architecture.

4.2 Final Model Result

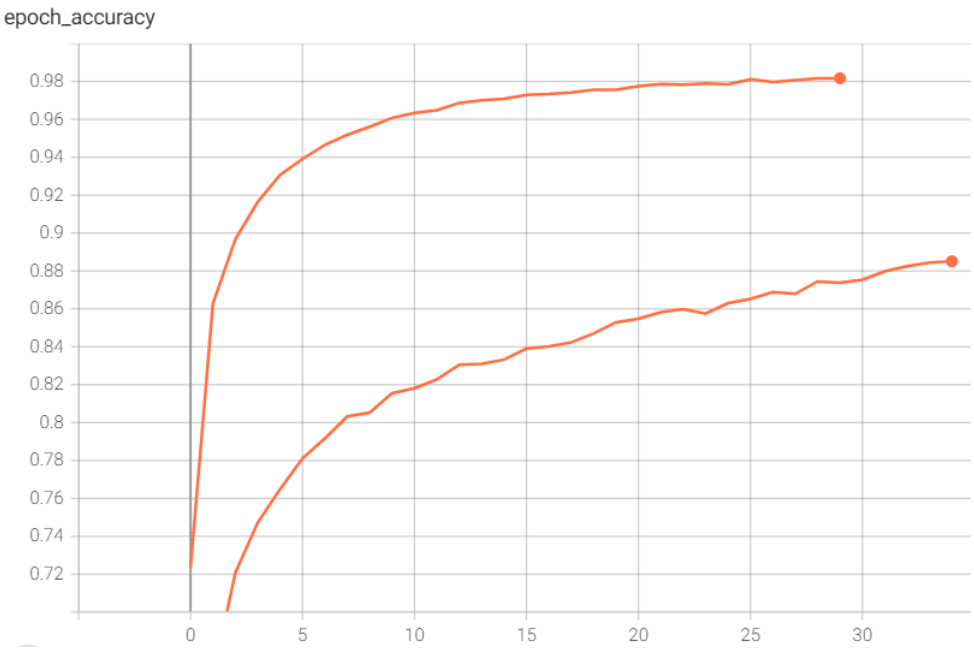


Figure 4.1: Figure showing comparative training accuracy of the first model(top line) vs the final model(bottom line)

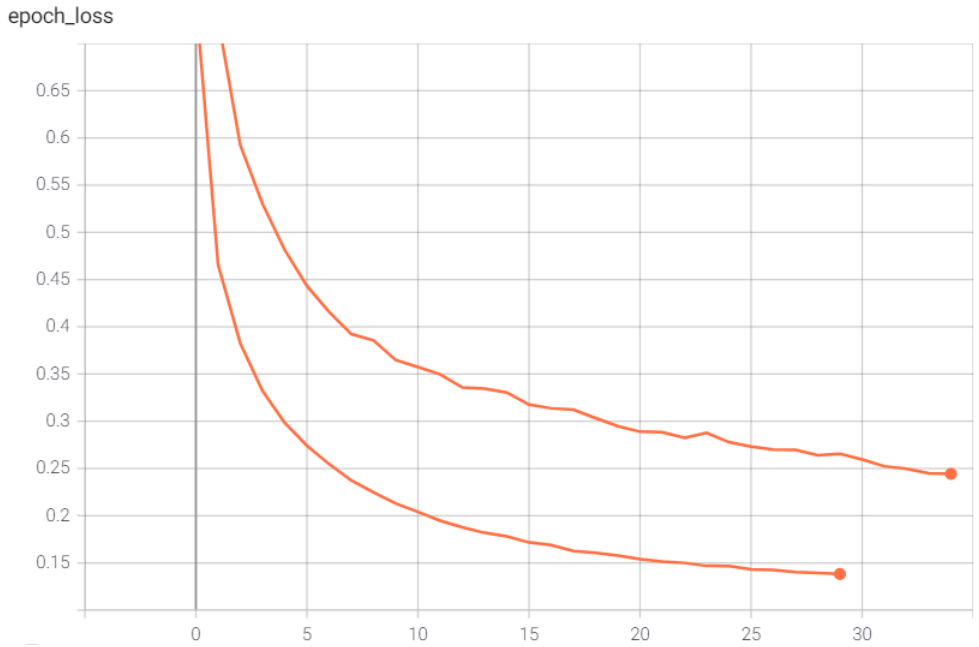


Figure 4.2: Figure showing comparative training loss of the first model(bottom line) vs the final model(top line)

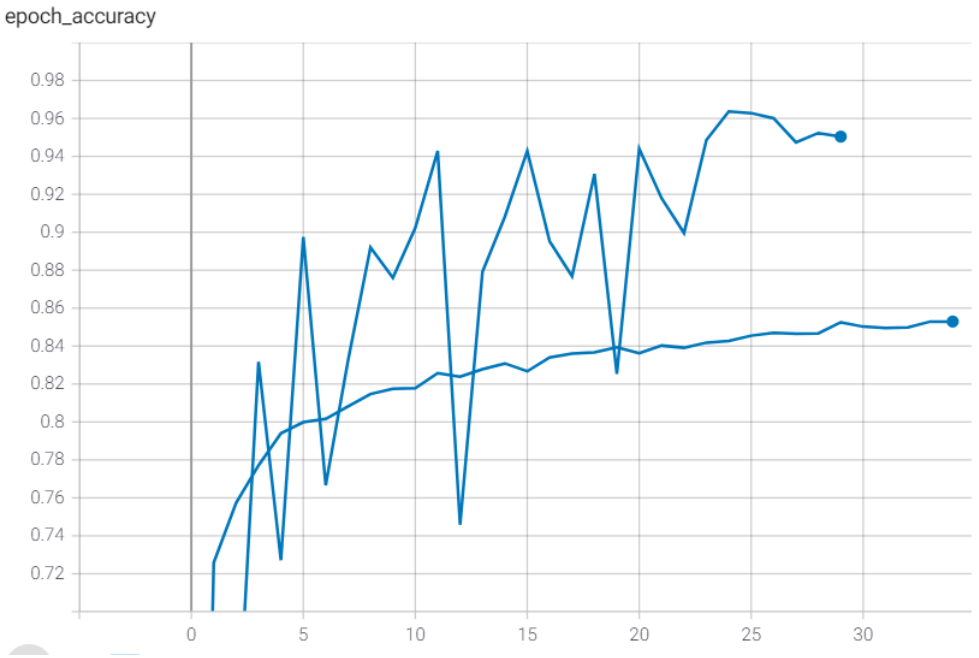


Figure 4.3: Figure showing comparative validation accuracy of the first model(top line) vs the final model(bottom line)

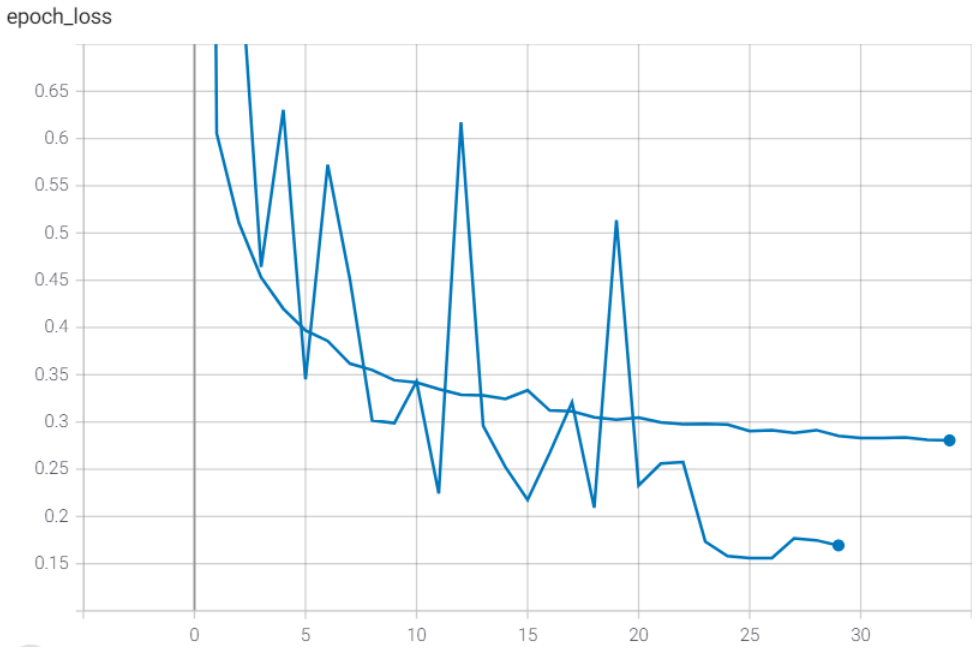


Figure 4.4: Figure showing comparative validation loss of the first model(bottom line) vs the final model(top line)

As we can see in the figures below the final proposed model at first glance doesn't perform as well as the first proposed model. But looking at validation, it becomes obvious that first model was over fitted and last model is far more optimized and accurate.
Our proposed model has 88.53% train accuracy and 24.14% train loss. And it has 85.68% validation accuracy and 27.65% validation loss. It took 14 seconds per epoch. Which makes the total time for 30 epoch just over 15 minutes. And the cost of running the model was also low memory-wise as it ran on only 8GB of VRAM of GPU and hardware-wise as the hardware used for running the model was NVIDIA GEforce 2060, an entry level GPU with capabilities of running neural network models. Which is very satisfactory for our goal.

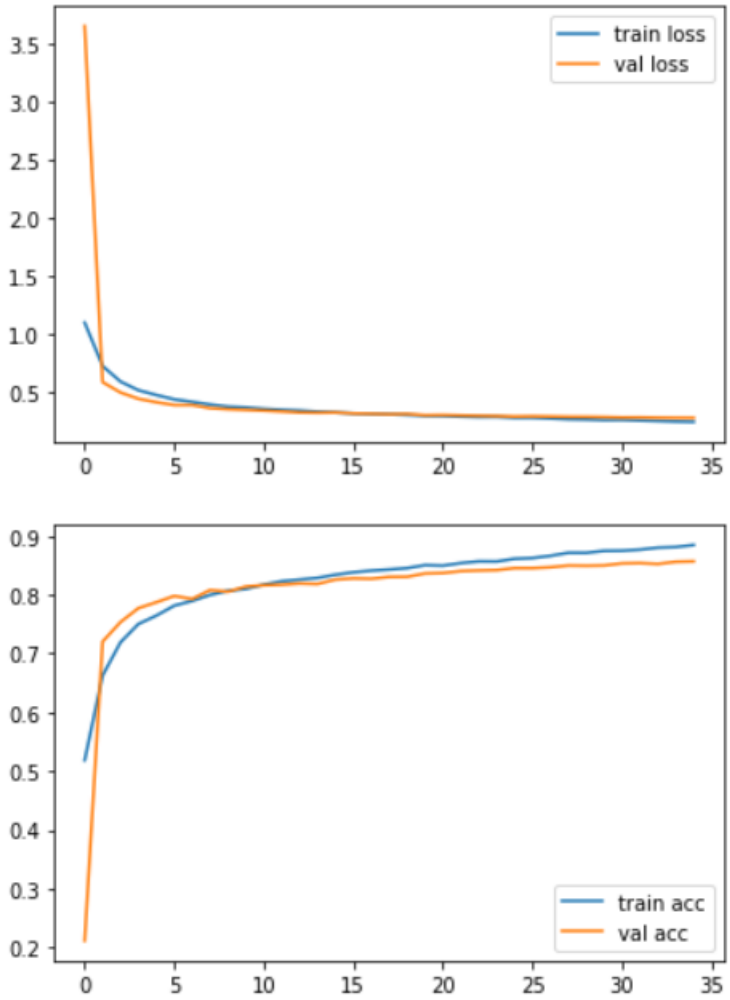


Figure 4.5: Figure showing train vs validation performance of the final model

Chapter 5

Pre-trained Model Result Analysis

5.1 Pre-Trained Model Implementation

Pre-trained models are available for downloading with pre-trained weights as they are generic and supposed to work on all types of image processing task [9]. In this paper, we have implemented some pre-trained models on our dataset. They are- VGG16, VGG19, Resnet50, Resnet101, ResNet152, Inception and Inception-ResNet. But for that reason, we need to add and edit the output layer suited for our need. As we are classifying five different types of images, we added five nodes and we used Softmax as the activation function for the output layer as it is a multi-class classification problem [18].

5.2 VGG16

VGG16 is a very deep convolutional neural network architecture, designed for classification, localisation and large scale image recognition. VGG16 also called Oxford-Net as the model was proposed by Simonyan and Zisserman (2015) and developed by Oxford University. This pretrained version of network can perform in ImageNet which is a dataset of over 14 million images and can classify them into 1000 object categories.[37] The main attributes are- convolutional layers, Max pooling layers, activation layers etc.[37] The'16'in VGG-16 refers to 16 layers which have 138 million parameters in total with 5 blocks and each block with a max pooling layer. Among them 9248 parameters are trainable. It always use same padding, convolutional kernels are of size 3x3 filter, stride 1 and Maxpool kernels are of size 2x2 filter, Stride 2. ReLU is used as an activation function in VGG-16.The default input size is 224x224. Because of its huge parameters, the architecture is comparatively slow.[37]

The formula of calculating image size of layer is: (Image size + 2 * Paddaing - Filter size) / Stride

VGG16 is a comparatively slower architecture because of its very large number of parameters which is certainly difficult to handle. Despite that, VGG16 is one of the most preferred CNN architectures in the recent past. The algorithm has achieved astounding accuracy, almost 92.7 percent in terms of image recognition. It has a

very uniform architecture and simple implementation. It allows us to execute transfer learning by importing a pre-trained model into PyTorch. After that, remove or add an extra fully connected layer at the end, depending on required instructions.[37]

5.2.1 Result of VGG16

We applied VGG16 and observed the characteristics of the model on the dataset. The result is shown by the figure below:

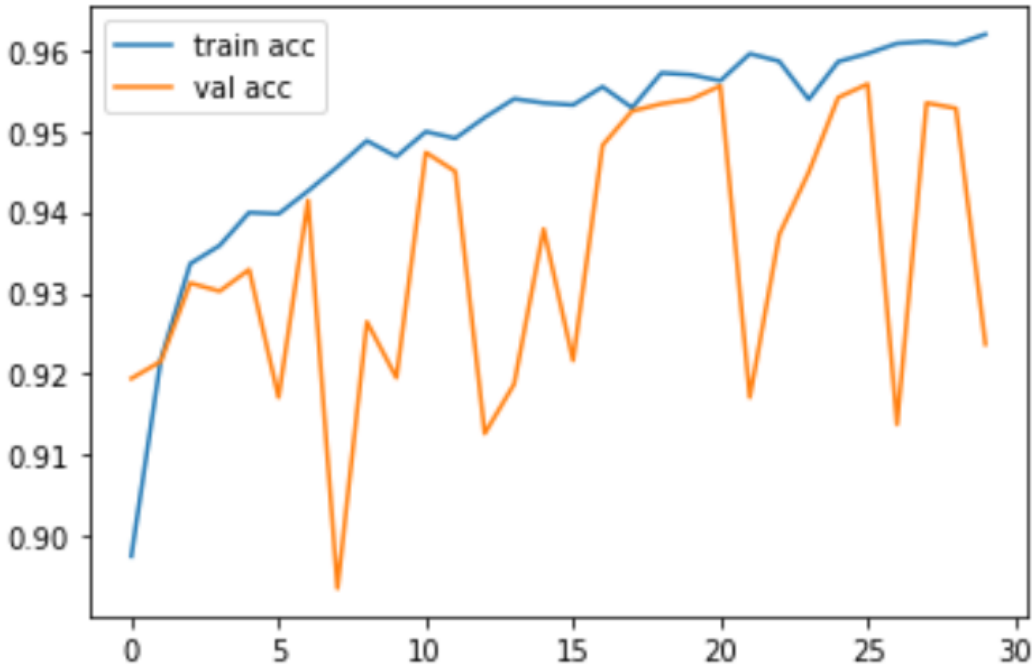


Figure 5.1: Figure showing accuracy on VGG16 model

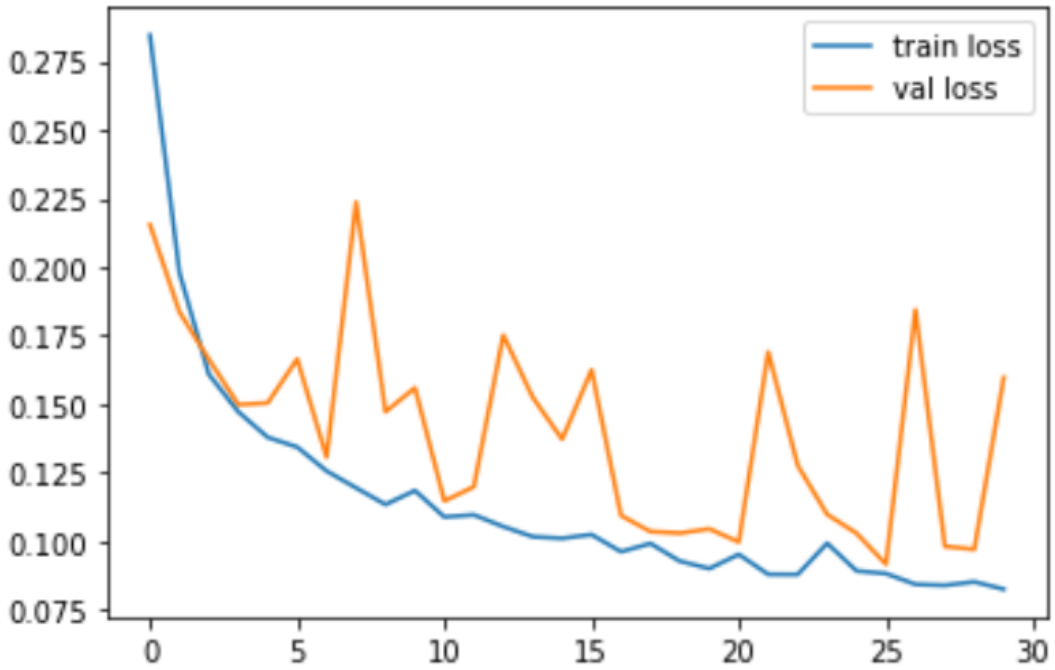


Figure 5.2: Figure showing loss on VGG16 model

5.3 VGG19

VGG19 is a deep convolutional neural network, a variant of VGG model. The concept of VGG19 is as same as VGG16 excepts the depth of the layers. VGG19 architecture consists of 19 layers including 16 Convolution layers, 3 Fully connected layers, 5 Max Pool layers and 1 SoftMax layer. VGG19 is slightly better in performance compared to VGG16, however it requires more memory. The model is trained on more than a million images from the ImageNet database. its pretrained network can identify images into 1000 different object categories. With that, the network has achieved high accuracy in rich feature representations for large scale image recognition.[36]

5.3.1 Architecture

In VGG-19, all convolution layers use(3,3) filters, and the number of filters increases by 2^n (such as: 64, 128, 256, 512). Stride length is 1 pixel and padding filter 1 pixel on each side in all Convolution layers. There are a total five sets of convolution layers. Two of them have 64 filters, next set has 2 conv layers with 128 filters, further next set has 4 conv layers with 256 filters, and next 2 sets have 4 conv layers each, with 512 filters. There are max pooling layers in between each set of conv layers. Max pooling layers use (2x2) filters, stride 2 pixels. The FLOP count of VGG19 is 19.6 billion.

This network was designed a fixed size (224 * 224) RGB image as input, indicating that the matrix was of shape (224,224,3). The only preprocessing was to subtract

the mean RGB value from each pixel, which was computed throughout the entire training set.

The output of the last pooling layer is flattened and supplied to a 4096-neuron fully connected layer. The output is supplied into a 4096-neuron fully connected layer, which in turn feeds into a 1000-neuron fully connected layer. ReLU is activated on all of these layers. At the last, a soft max layer utilizes cross entropy loss.

Only the convolution layers and the Fully Connected layers have trainable weights. The max pool layer is used to reduce the size of the input image, and soft max makes the final decision.

5.3.2 Result of VGG19

After applying VGG19, we observe the performance of the model on given dataset. The result is shown below:

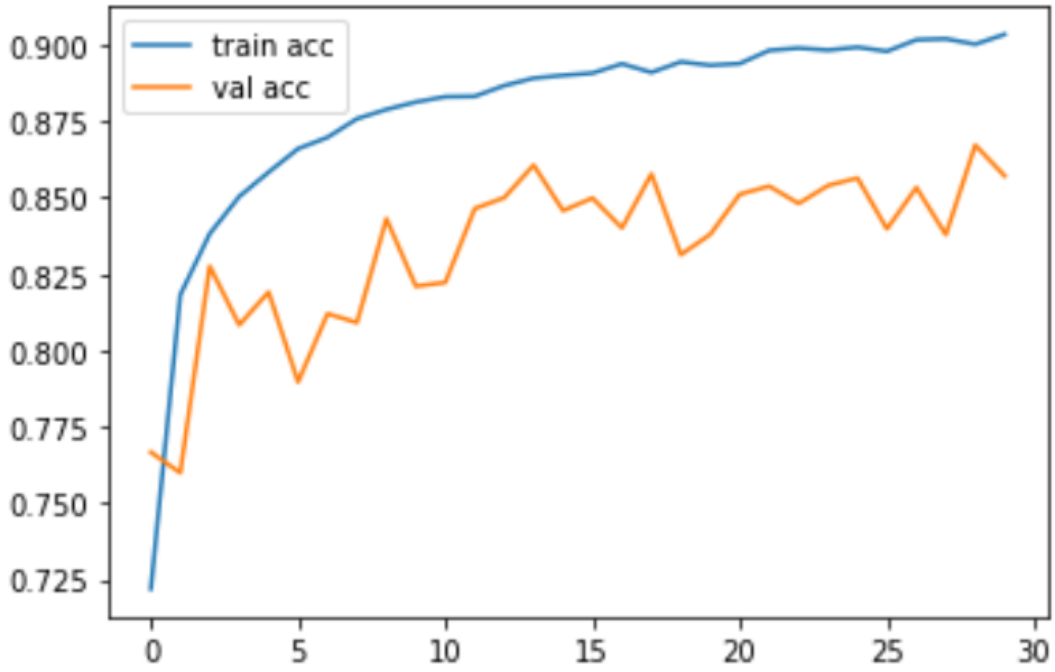


Figure 5.3: Figure showing accuracy on VGG19 model

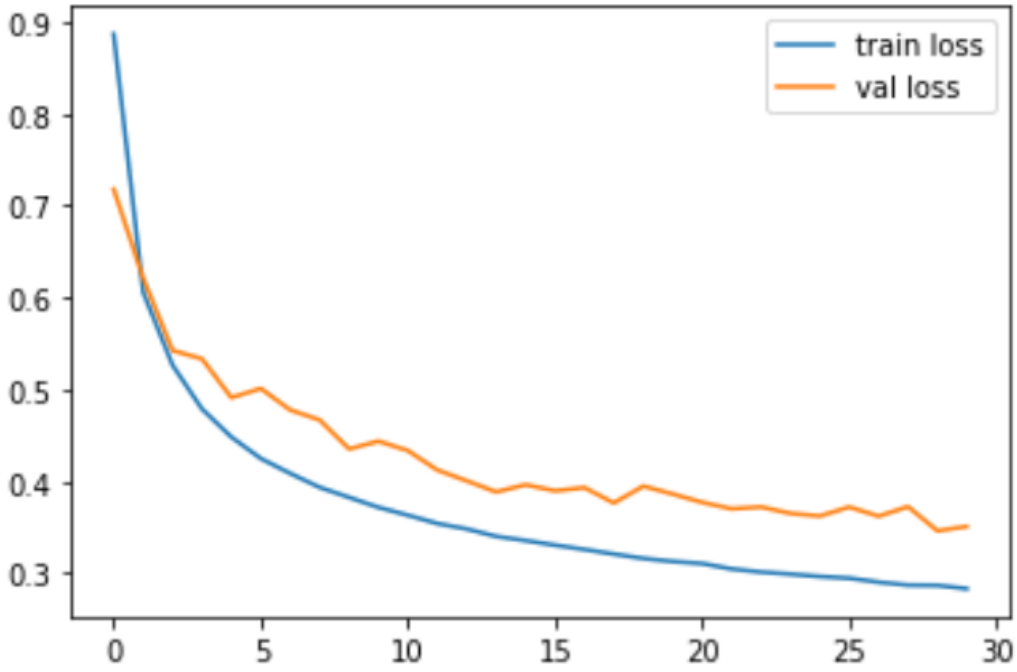


Figure 5.4: Figure showing loss on VGG19 model

Here, VGG19 is performing better than VGG16 and also more stable.

5.4 ResNet

We have applied ResNet50, ResNet101, ResNet152 on our dataset.

5.4.1 ResNet50

ResNet50 is a version of ResNet model. It is an architecture that introduced a model-framework that made it achievable to train very Deep Neural Networks containing over thousand fold layers while delivering better accuracy. The first application of it was using image recognition problem but in the published paper it is suggested that it can also be applied to other problems excluding computer vision to reach higher accuracy.[32]

Architecture

It has 48 Convolutional layers, 1 MaxPool and 1 Average Pool layer. In it, there are $3.8x10^9$ floating point operations. It takes in 224x224 size photos in the input layer. There are 5 convolution layers. First layer is a convolution layer. It has filter size of 7x7. This layer consists of 64 nodes. It has stride of 2. Second layer is a convolution layer. It consists of 4 sub layer. First sub layer is a max pooling layer of 3x3 filters and stride of 2. Next three sub layers each contains three layers. Each set of these three layers are [(1x1, 64), (3x3, 64), (1x1, 256)] in design where [(filter size, node count)]. Third layer is a convolution layer. It consists of 4 sub layer. These four sub layers each contains three layers. Each set of these three layers are [(1x1, 128), (3x3, 128), (1x1, 512)] in design. Fourth layer is a convolution layer. It consists of 6 sub layer. These four sub layers each contains three layers. Each set of these three layers are [(1x1, 256), (3x3, 256), (1x1, 1024)] in design. Fifth layer is a convolution layer. It consists of 6 sub layer. These four sub layers each contains three layers. Each set of these three layers are [(1x1, 512), (3x3, 512), (1x1, 2048)] in design. All the layers has ReLu as activation function. After convolution layers there is fully connected layer of average pool with node count of 1000. And lastly there is output layer which has same amount of node as the number image classes.

5.4.2 ResNet101

It is another variant of ResNet. The difference with ResNet50 is very minute, only in the length of 4th convolutional layer to be precise. In it, the fourth convolutional layer consist of 23 sub layers while each sub layer constructed of [(1x1, 256), (3x3, 256), (1x1, 1024)] in design. [34]

5.4.3 Resnet152

It is another variant of ResNet. The difference with ResNet50 and ResNet101 is very minute, only in the length of 4th convolutional layer to be precise. In it, the fourth convolutional layer consist of 36 sub layers while each sub layer constructed of [(1x1, 256), (3x3, 256), (1x1, 1024)] in design.

5.4.4 Result of ResNet

The result we got from running this model on our dataset is given below.

ResNet50

At first, we applied ResNet50 and observed the following characteristics:

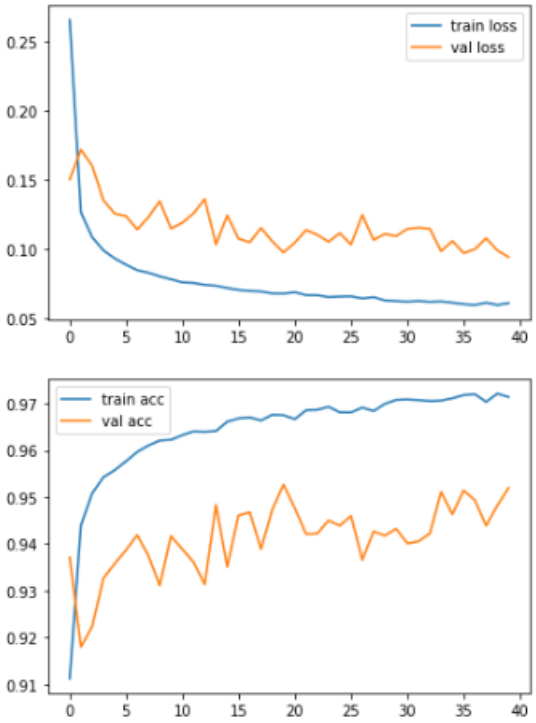


Figure 5.5: Figure showing result on ResNet50 model

ResNet101

The characteristics we found after applying ResNet101 on the dataset , is given below:

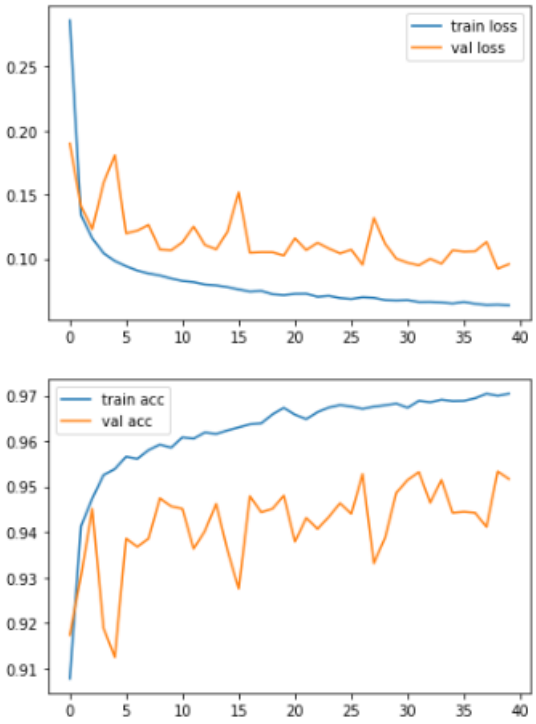


Figure 5.6: Figure showing result on ResNet101 model

ResNet152

When applied Resnet152, we found the following characteristics of our model on given dataset:

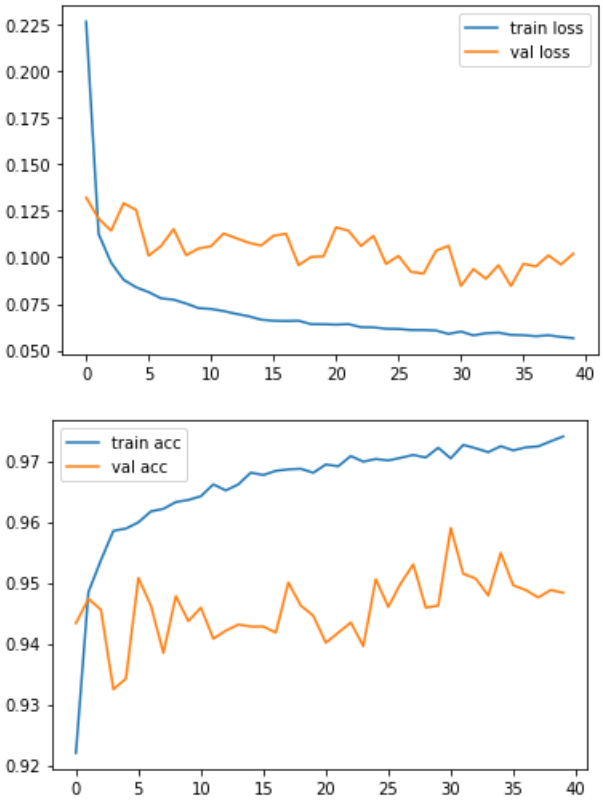


Figure 5.7: Figure showing Result on ResNet152 model

Among them the best performing model is Resnet50. Further investigation is necessary.

5.5 Inception

Inception module is an image model block that provides more efficient computation and deeper networks in a CNN. it allows us to use multiple types of filter size, instead of being restricted to a single filter size, in a single image block, which we then concatenate and pass onto the next layer[35]. The problems in VGGNet are deploying these deep architecture models is highly expensive. When the model is bigger and the training data is small, very deep networks are susceptible to overfitting. Passing gradient updates through the entire network was also difficult[35].

5.5.1 Architecture

The design of initial Inception Module is commonly known as GoogLeNet, or Inception v1. Additional variations to the inception module have been designed, reducing

issues such as the vanishing gradient problem.

Inception Net v1 is basically a 27 layers deep CNN. Inception Layer is a combination of all those layers (inception layer, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage Then, the resulting outputs are concatenated and sent to the next layer. By organizing the CNN so that all of its convolutions are performed at the same level, the network gets wider rather than deeper. Along with the above-mentioned layers, there are two major add-ons in the original inception layer:
1. 1×1 Convolutional layer before applying another layer
2. A parallel Max Pooling layer, that provides another option to the inception layer [31]

5.5.2 Result on Inception

The result we got from running this model on our dataset is given below.

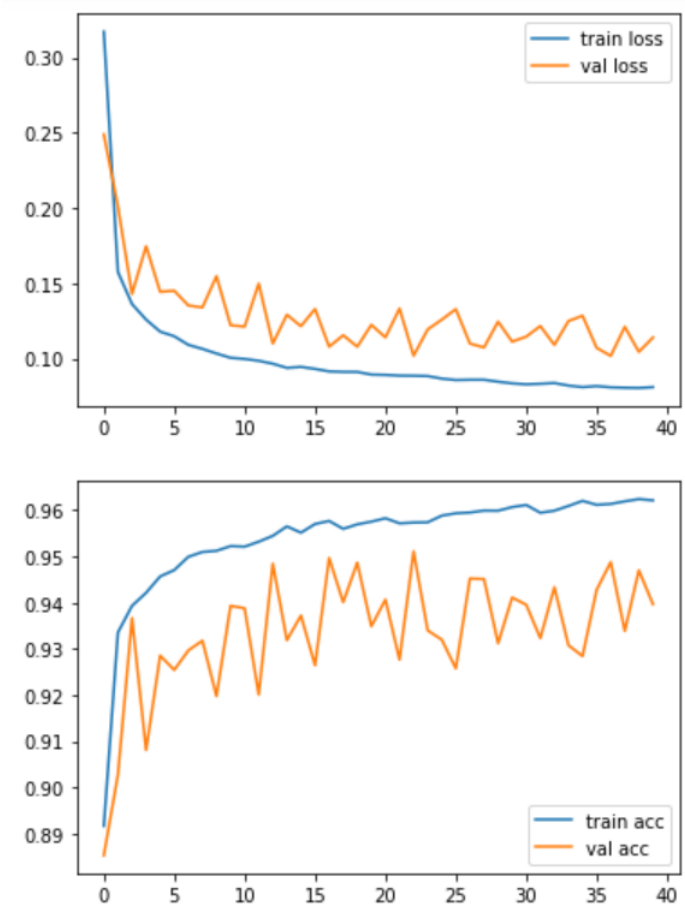


Figure 5.8: Figure showing Result on Inception model

Inception perform slightly less than ResNet50. It is observed that, for such problem, ResNet style approach is more suited.

5.6 Inception-ResNet-v2

Inception-ResNet-v2 is a 164 layers deep CNN module. The network is trained on over million images from ImageNet dataset and the module has acquired high accuracy on identifying images into 1000 classes. The module is popular for its rich feature image representations. The network has an image input size of 299-by-299.

5.6.1 Benefits

Inception Net shows relatively lower error rate. It performs better in terms of speed and accuracy compared with the VGGNet.

It drastically reduces computational cost by introducing a 1x1 convolution. To add an extra 1x1 convolution before the 3x3 and 5x5 layers and after the max pooling layers. By doing so, the number of input channels is limited and 1x1 convolutions are far cheaper than 5x5 convolutions.

1×1 convolution can help to reduce model size which can also somehow help to reduce the over fitting problem.

In the previous model, layer of the network would probably focus on the overall image to identify the different objects present there. To focus on individual parts of an image, it allows the internal layers to pick and choose which filter size will be relevant to learn the required information. So even if the size of the particular part in the image is different, the layer works accordingly to recognize that.

5.6.2 Result on Inception-ResNet

The result we got from running this model on our dataset is given below.

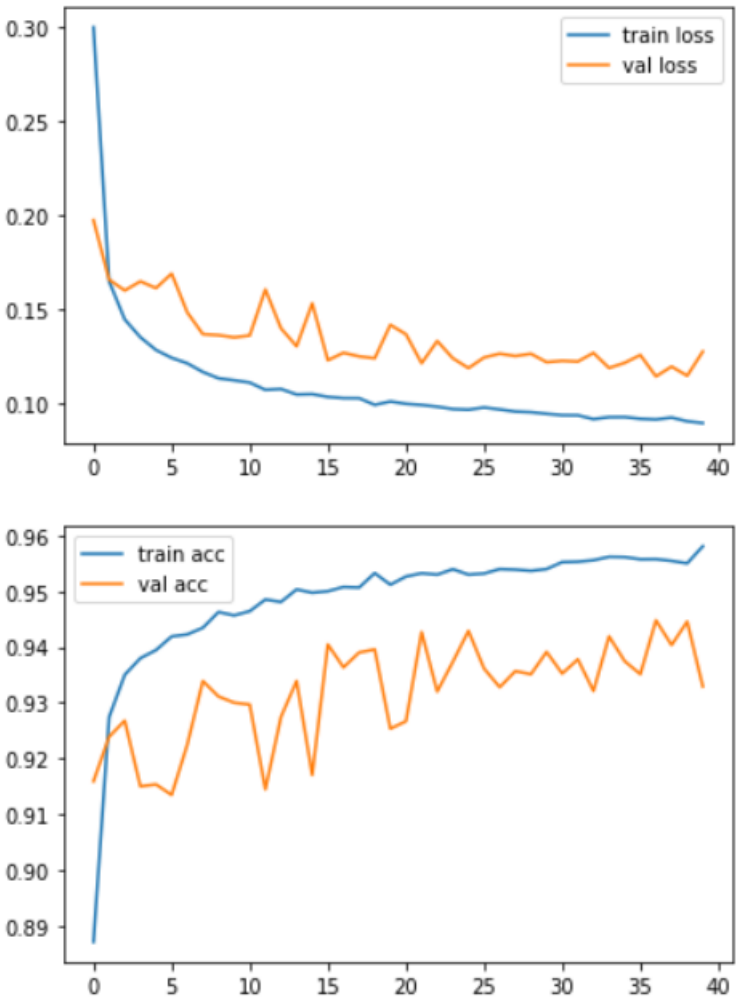


Figure 5.9: Figure showing Result on Inception-ResNet model

5.7 Comparison Between All Pre-trained Models

Here in this problem we can see that the best train accuracy is shown by ResNet152. But it does not necessarily show the best validation accuracy. The best validation accuracy is shown by ResNet50. So from here we can come to the conclusion that a deeper neural network or more layers don't always give better accuracy.
Again, in case of training loss, ResNet152 has the lowest loss. but validation loss is lowest observed in ResNet50. So, in this case we also see the same phenomenon of less layer architecture performing better.
Overall, ResNet50 architecture performs best in this problem. With its $3.8 * 10^9$ floating points operations per second and depth seems to be best suitable among all the tested architectures.
That being said, further investigation on other models are necessary to find the best suitable approach and architecture for this problem.

Model	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
VGG16	0.0825	0.9619	0.1598	0.9237
VGG19	0.2815	0.9051	0.3511	0.8571
INCEPTION	0.0802	0.9621	0.1140	0.9397
INCEPTION-RESNET-V2	0.0893	0.9586	0.1277	0.9329
RESNET50	0.0606	0.9721	0.0945	0.9520
RESNET101	0.0633	0.9706	0.0956	0.9517
RESNET152	0.0553	0.9750	0.1020	0.9483

Table 5.1: All pre trained model comparison

5.8 Comparison Between Proposed model and Pre-trained Models

When we look at the result of proposed model and pre-trained model, it is very obvious that our proposed model falls short in terms theoretical performance on accuracy and loss by a little margin. Yet it would be unjust towards the proposed model to dismiss it without evaluating it further.

Our proposed model has 88.53% train accuracy and 24.14% train loss. And it has 85.68% validation accuracy and 27.65% validation loss. It took 14 seconds per epoch. The previous research on solving this problem was an approach which used neural network rather than a deep network and used and one vs. all classification approach and used multiple models for multiple classes to classify images. In contrast to that we used a deep neural network architecture and one single class to classify multiple classes, Which is a big success as a next step in this field.

Furthermore, there are some points which our model has an edge over pre-trained models.

Firstly, Our proposed model is very lightweight in terms of nodes and deepness of layers, which in return makes it complete prediction task much further than any pre-trained model. to complete 30 Epochs, pre trained models take over 10 hours while the proposed model take less than half an hour.

Secondly, it was not feasible for us to check the behavior of the model for higher epochs (i.e. 300, 200, 100 epochs) due to memory limitation of the hardware. It would have been a very interesting information for further optimization if we could know how the model behaves at higher epochs.

Lastly, if we could simulate a larger and more diverse data set and run our proposed model on that data set, then training and optimizing the model would have been more effective. Also,the model has high possibility to achieve that goal with stronger hardware.

Chapter 6

Conclusion and Future Works

6.1 Concluding Remarks

We can claim that, we have fulfilled our main objectives. That is to accurately classify all the classes using a single model and not using a model for every other classes individually, run the model faster, cheaper and in more cost-effective manner. This approach has successfully classified the configurations with high accuracy. All the algorithms, number of epochs, learning rates, optimiser have been used after trial and error method to come up with the best accuracy levels. Thus, we managed to complete all the core objectives by implementing our proposed single optimized model.

6.2 Future Direction

Machine learning has been helping us in numerous field in our daily life for quite a while. From YouTube video recommendation to google's self driving car project, all of it has been possible due to machine learning. With the eve of deep learning, more complex task which was not possible even with machine learning has now become possible.

As it can be seen here, magnetic structure configuration detection with deep network is very feasible field to invest into. With a custom made architecture for this problem, the hardware cost would also be very low and commercial application would be feasible due to cost effectiveness. We need to further research on different and diverse data, test on real life data, reiterate certain aspect of the architecture so that it can further perform in a cost effective manner. We plan on going a step forward by including not just magnetic but all sorts of matter that is present, This could contribute to automatically identifying an unknown piece of object found. Such classification can greatly help researchers and scientists to learn more about matter.

We really hope our effort would be able to help researcher around the word further explore the topic enrich the knowledge of mankind in material science one step further.

Bibliography

[1] G. F. Newell, E. W. Montrell, I. Perlman, G. T. Seabord, D. R. Inglis, A. E. Scheidegger, and J. M. Hollander, *On the theory of the Ising model of ferromagnetism*. Published for the American Physical Society by the American Institute of Physics, 1953, vol. 25.

[2] A. Macintosh, R. Ellis, and T. Allen, *Applications and Innovations in Intelligent Systems 13*. Springer London, 2006.

[3] U. Nowak, *Classical spin models*, Dec. 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470022184.hmm205.

[4] M. Heide, G. Bihlmayer, and S. Blügel, "Dzyaloshinskii-moriya interaction accounting for the orientation of magnetic domains in ultrathin films: Fe/w(110)," *Physical Review B*, vol. 78, no. 14, 2008. DOI: 10.1103/physrevb.78.140403.

[5] I. T. Jolliffe, *Principal component analysis*. Springer Science Business Media, LLC, 2010.

[6] D. Lamberto, M. Finazzi, and F. Ciccacci, *Magnetic properties of antiferromagnetic oxide materials: surfaces, interfaces, and thin films*. Wiley, 2010.

[7] S. J. Blundell, *Magnetism in condensed matter*. Oxford University Press, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and et al., *Imagenet large scale visual recognition challenge*, Apr. 2015. [Online]. Available: https://link.springer.com/article/10.1007/s11263-015-0816-y.

[10] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, 2015. arXiv: 1512.00567 [cs.CV].

[12] W. Koshibae and N. Nagaosa, *Theory of antiskyrmions in magnets*, Jan. 2016. [Online]. Available: https://www.nature.com/articles/ncomms10542.

[13] G. Torlai and R. G. Melko, "Learning thermodynamics with boltzmann machines," *Physical Review B*, vol. 94, no. 16, p. 165 134, 2016.

[14] R. Wiesendanger, *Nanoscale magnetic skyrmions in metallic films and multilayers: A new twist for spintronics*, Jun. 2016. [Online]. Available: https://www.nature.com/articles/natrevmats201644.

[15] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, no. 6325, pp. 602–606, 2017. DOI: 10.1126/science.aag2302.

[16] J. Carrasquilla and R. G. Melko, *Machine learning phases of matter*, Feb. 2017. [Online]. Available: https://www.nature.com/articles/nphys4035.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. The MIT Press, 2017.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. DOI: 10.1145/3065386.

[19] K. Mills, M. Spanner, and I. Tamblyn, *Deep learning and the schrödinger equation*, Oct. 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.96.042113.

[20] R. Gandhi, *Support vector machine - introduction to machine learning algorithms*, Jul. 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[21] I. A. Iakovlev, O. M. Sotnikov, and V. V. Mazurenko, "Supervised learning approach for recognizing magnetic skyrmion phases," *Physical Review B*, vol. 98, no. 17, 2018. DOI: 10.1103/physrevb.98.174411.

[22] A. A. Kovalev and S. Sandhoefner, "Skyrmions and antiskyrmions in quasi-two-dimensional magnets," *Frontiers in Physics*, vol. 6, 2018. DOI: 10.3389/fphy.2018.00098.

[23] D. Maccariello, W. Legrand, N. Reyren, K. Garcia, K. Bouzehouane, S. Collin, V. Cros, and A. Fert, "Electrical detection of single magnetic skyrmions in metallic multilayers at room temperature," *Nature Nanotechnology*, vol. 13, no. 3, pp. 233–237, 2018. DOI: 10.1038/s41565-017-0044-4.

[24] T. Pyzdek and P. Keller, *The Six Sigma handbook*, 5th. McGraw-Hill Education, 2018.

[25] S. Saha, *A comprehensive guide to convolutional neural networks-the eli5 way*, Dec. 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[26] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, *Convolutional neural networks: An overview and application in radiology*, Jun. 2018. [Online]. Available: https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9.

[27] Y.-Z. You, Z. Yang, and X.-L. Qi, "Machine learning spatial geometry from entanglement features," *Physical Review B*, vol. 97, no. 4, 2018. DOI: 10.1103/physrevb.97.045153.

[28] Y. Zhou, *Magnetic skyrmions: Intriguing physics and new spintronic device concepts*, Oct. 2018. [Online]. Available: https://academic.oup.com/nsr/article/6/2/210/5123735.

[29] E. Balkind, A. Isidori, and M. Eschrig, *Magnetic skyrmion lattice by fourier transform method*, Apr. 2019. [Online]. Available: https://arxiv.org/abs/1901.02459.

[30] S. Bokul, S. S. M. A. Shukur, and S. Ahmed, *Classification of magnetic configurations using machine learning algorithms*, 2019.

[31] DeepAI, *Inception module*, May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/inception-module.

[32] J. Peng, S. Kang, Z. Ning, H. Deng, J. Shen, Y. Xu, J. Zhang, W. Zhao, X. Li, W. Gong, and et al., *Residual convolutional neural network for predicting response of transarterial chemoembolization in hepatocellular carcinoma from ct imaging*, Jul. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s00330-019-06318-1/figures/1.

[33] V. K. Singh and J. H. Han, "Application of machine learning to two-dimensional dzyaloshinskii-moriya ferromagnets," *Physical Review B*, vol. 99, no. 17, 2019. DOI: 10.1103/physrevb.99.174426.

[34] S.-H. Tsang, *Review: Resnet - winner of ilsvrc 2015 (image classification, localization, detection)*, Mar. 2019. [Online]. Available: https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8.

[35] *Inception network: Implementation of googlenet in keras*, May 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/.

[36] A. Kaushik, *Understanding the vgg19 architecture*, Feb. 2020. [Online]. Available: https://iq.opengenus.org/vgg19-architecture/.

[37] *Convolutional network for classification and detection*$_2$021, Feb. 2021. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/.

[38] [Online]. Available: http://cs231n.github.io/convolutional-networks/..

[39] Z. Jaadi, *A step-by-step explanation of principal component analysis (pca)*. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis.

[40] S. Krause and R. Wiesendanger, *Spintronics: Skyrmionics gets hot*. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2016NatMa..15..493K/abstract.