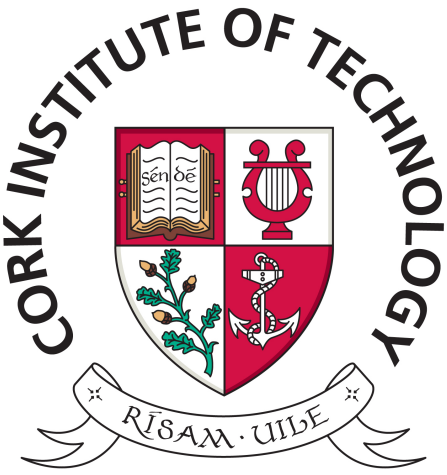

Recognition of Yoga postures from Images using Deep learning

by

Sriranjani Sridharan

This thesis has been submitted in partial fulfillment for the degree of Master of Science in Artificial Intelligence

in the

Faculty of Engineering and Science

Department of Computer Science

May 2020

Declaration of Authorship

I, Sriranjani Sridharan , declare that this thesis titled, 'Recognition of Yoga postures from Images using Deep learning' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an masters degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Sriranjani Sridharan

In recent times, Yoga has gained a lot of popularity across the globe due to its physical and mental health benefits. The objective here, is to build a classification model that would recognize various Yoga postures or asanas from a small set of images using a combination of OpenPose [1] and deep learning. This model can be extended and implemented as a tool to help beginners practice yoga with minimal investment and in the comfort of their homes by just using their cameras and receiving feedback on their forms. A lot of different methods are still being researched with respect to yoga pose recognition and development of self-training systems. The proposed work is based on the high level approach followed by the students of Stanford University in using OpenPose with a conventional deep learning model for classification of yoga postures from images [2]. Considering the small size of the data being used, the impact of using transfer learning was studied in addition to building a conventional deep learning model from scratch for the classification problem. The results from both the conventional and transfer learning approach seemed to converge closely around the same performance level with the latter having a slight edge in outperforming the former. The overall system achieved an accuracy of 82% in classifying 10 different yoga postures from images.

Acknowledgements

I am very grateful to all my mentors and supervisors for patiently supporting and guiding me through my entire work.

Especially, I would like to thank Dr Diarmuid Grimes for his insightful inputs and feedback with the dissertation development. I would also like to extend my appreciation to Dr Ted Scully for his guidance in understanding the field of Deep Learning.

Finally, I would like to thank my husband Badri Narayanan, and my entire family who have been a great support throughout this masters.

Contents

List of Figures

List of Tables

Abbreviations

NN	**N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
IDE	**I**ntegrated **D**evelopment **E**nvironment
API	**A**pplication **P**rogramming **I**nterface

Chapter 1

Introduction

Sports and physical fitness exercises has always been an attraction to people around the world. With advancements in computer vision and multimedia related techniques, the thirst for automatic or semi-automatic fitness training systems has increased as its one of the most flexible option for people involved in a rapidly moving lifestyle. In this study, we focus on a computer vision based approach that would help in building yoga self training systems for people to learn and practice yoga.

1.1 Motivation

Yoga is a form of practice, that originated 5000 years ago as part of ancient Indian philosophy. It combines the art of breathing, meditation and physical training based on different body postures or asanas that enhance human well-being. Yoga exercises boost physical health and help to cleanse the body, mind, and soul [3]. Self-training systems for practising yoga can spread the benefits behind this ancient science and gain popularity while ensuring that it is practiced correctly. Our current work is considered as a start to developing a complete self training system that can be used as an application or a tool by beginners to practice yoga at the comfort of their homes.

1.2 Executive Summary

Human posture recognition is an interesting and challenging Computer vision problem, that has been studied by researchers for many years. The application of human activity recognition/pose estimation techniques has shown a great potential in the field of multimedia, surveillance, dance choreography, physical training, and Virtual reality.

However, the exploration in the field of yoga has just started to gain its pace. In the proposed work, a classification system was modelled to learn and automatically recognize yoga postures from images. The proposed system is capable of classifying up to 10 yoga postures, including: (1) Bridge (Setu Bandha Sarvangasana), (2) Downward-Facing Dog (Adho Mukha Shvanasana), (3) Child's Pose (Balasana), (4) Mountain (Tadasana), (5) Plank (Kumbhakasana), (6) Seated Forward Bend (Paschimottanasana), (7) Tree (Vrksasana), (8) Triangle (Utthita Trikonasana), (9) Warrior1 (Virabhadrasana I) and (10) Warrior2 (Virabhadrasana II). Although the names of these postures originated from Sanskrit language, we will use the associated English posture names to refer them throughout this study. For further details on each yoga posture please refer [4].

The input to the system is an image or a frame that shows a person doing yoga. This image is run through an open source tool for pose estimation called OpenPose. OpenPose is a system developed by Carnegie Mellon University to jointly detect human body and hand key-points on single images. Openpose adopts an algorithm that recognizes and highlights human body parts and generates a frame with skeletal marks based on the human posture. The primary purpose of identifying body key-points is to remove any unwanted background noise from images. The key-points generated is then passed to the training model which is a convolutional neural network(CNN) that will predict an output identifying the class of the posture. The high-level model structure was implemented based on the work done by the students of Stanford University [2]. The training model was studied based on two approaches: a Convolutional Neural Network(CNN) built from scratch and with pre-trained models using the concept of transfer learning.

1.3 Contribution

There has been a lot of work related to rocognition of yoga postures from images. A yoga tutor project was proposed by Patil et al [5] that uses SURF algorithm to detect the difference in postures between an expert and a user practising yoga. A yoga self training system introduced by Chen et al. [6] assists in rectifying yoga postures for 12 different asanas. It involves manual feature extraction and building an individual model for each asana which increases the computational costs. Although a number of studies exist, none of the works that the author can find combines the pose estimation technique OpenPose with pre-trained models (Transfer learning) for yoga pose recognition. Moreover, the evaluation of the model built, is implemented with a very small set of images for yoga posture recognition, which is a major challenge in the underlying field. A principal contribution of this study determines if Transfer learning outperforms the traditional methods for the classification problem given the small amount of data.

1.4 Structure of This Document

The remaining sections in the document adds more detail to the topic in discussion and is structured as follows: Chapter 2 presents existing solutions as well as background information on the trends and techniques in the underlying field and outlines the state of the art approaches utilized in similar image classification tasks; Chapter 3 presents the proposed work towards classification of yoga postures that includes the implementation framework, the tools used, the model architecture and the approaches for training and tuning the models; Chapter 4 details the study of experiments with special emphasis on the evaluation and result observations ; Chapter 5 describes the challenges, conclusions and findings from the study along with a discussion of the future work arising from this study.

Chapter 2

Background

In this chapter, a brief background of the areas applicable to the current study and previous work done in the related field is described. It also includes an overview of the techniques and concepts involved in the related research area that has been used in the current study.

2.1 Computer vision

As humans, perceiving the three-dimensional structures that exist around us, is easy. We are able to construct a vision of particular objects as a sequence of images based on our previous knowledge. The human brain links all the abstract knowledge to construct and process these images in order to perceive the world around us. To impart such knowledge into machines is not as easy as it is with humans. However, we are trying to do the inverse in computer vision, where we describe what we see through a set of images and reconstruct its properties such as the colour distributions, its shape and so on [7]. Computer vision as a field, collects the methods for analyzing, obtaining, processing and understanding the images and other high dimensional data that we provide. The aim of this discipline is to learn from these data and produce numerical information that is understood by the computer in the form of decisions [8] and the fundamental concept behind it is to match the abilities of human vision by perceiving and understanding the data electronically.

The input data can be of any form such as coloured images, videos, sequence of images from multiple cameras, medical scan images etc. Computer vision seeks to apply its collection of theories and methods for the construction of decision models or systems. Computer vision has been used to address key tasks in many sub-fields such as object

detection, face recognition, action and activity recognition, scene understanding, image restoration and human pose estimations [9]. Our study will look into the aspect of action and activity recognition in relation to yoga postures.

2.2 Deep Learning

Over the past few years, deep learning methods have been outperforming previous state of the art machine learning techniques in various fields, with the most prominent cases being in computer vision [9]. Deep learning allows us to build computational models consisting of multiple hidden layers of mathematical functions to process and learn from the given data. It works by mimicking the human brain with multiple levels of abstraction in order to capture and understand the intricate structures of the data being represented. The foundation of deep learning is rooted from the classical Neural Networks(NN) architecture. Neural networks are built using single or multiple hidden layers of neurons, where each neuron represents a mathematical function and is modelled to function like a biological neuron. A deep learning network contains at least 3have hidden layers and hence is referred to as "Deep" meaning more than 2 layers. Figure2.1 represents the architecture of a simple neural network.

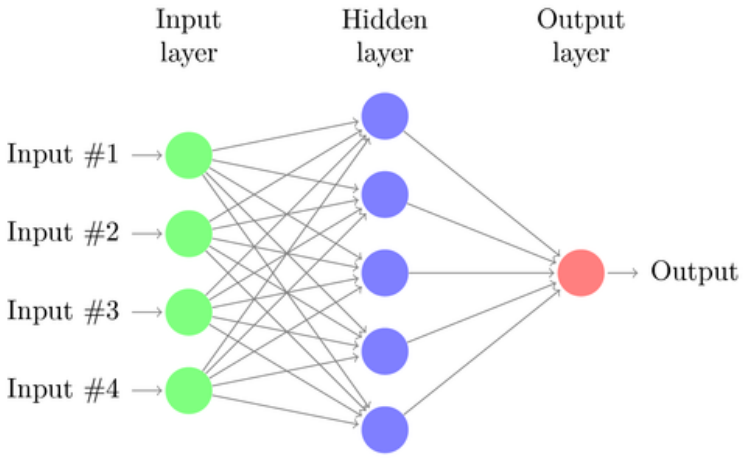

FIGURE 2.1: A simple neural network[10]

With this simple network architecture as a baseline, numerous variants of neural network architectures have been implemented and studied over the years such as Convolutional neural networks(CNNs), Deep Belief Networks, Recurrent Neural networks, Stacked Autoencoders and Deep Boltzmann Machines. Among the the several variants, CNNs have had the greatest impact in the field of Deep Learning and has proven to be a great asset to computer vision applications due to its unique capability of learning features from a given data and being invariant to transformations. A review of some of the

significant deep learning applications with respect to computer vision problems is studied in [9].

2.3 Convolutional Neural Networks

As mentioned previously, CNNs have demonstrated remarkable performance in many areas such as image recognition, video understanding, time series predictions and much more. CNNs have proven to be a powerful feature extraction mechanism when it comes to classification or segmentation of data [11]. Figure2.2 shows a high-level architecture of a CNN which is built using kernels, image filters or neurons that are processed based on learnable parameters such as weights and biases. The structure of a CNN contains convolutional layers, pooling layers and fully connected layers.

1. **Convolutional layers**

   It comprises the core building blocks of CNNs and does most of the heavy computational processes. The primary purpose of this layer is to extract features from the input data such as an image that is represented in the form of pixels as shown in the figure 2.2. The stacked set of squares within the convolution layers is referred as filters. A particular filter is used to understand the image features by learning from small sub-squares of pixel combinations in the input image. Such multiple filters are applied to the input image resulting in a 2-dimensional output array from the convolutional layers are called feature maps or activation maps [12]. These are fed as input to the following layer.

2. **Pooling layers**

   They work on reducing the spatial dimensions of the input volume being fed into the next set of convolutional layers or fully connected layers. This operation is also referred to as downsampling or subsampling as this size reduction leads to a loss of information [9]. However, it retains the most important information. This operation is performed to achieve better generalization, robust to distortion, faster convergence and works against overfitting.

3. **Fully connected layers**

   Following the set of convolutional and pooling layers, the high-level decision making in the network is performed by the fully connected layers. This layer resembles the network design shown in Figure2.2, although the number of hidden layers and the neurons might vary as required by the application in the given problem. Every neuron in one layer is fully connected to every other neuron in the following

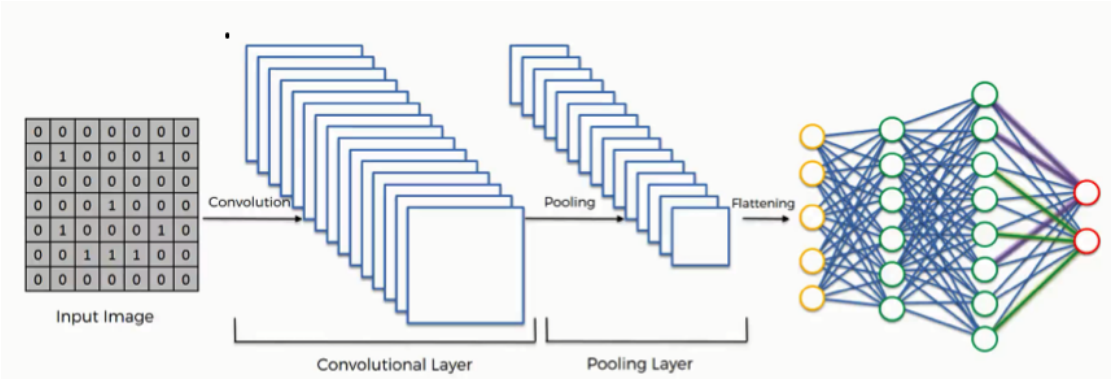

FIGURE 2.2: Convolutional neural network design which is an extension or variant of a simple neural network[13]

layer.The goal of this layer is to employ the features received from the previous layers, for reasoning the image classification problem. Each neuron in these layers is governed by a mathematical function also referred to as the activation function and it controls the output from the neuron. There are many activation functions like Sigmoid, Tanh, ReLU, Leaky ReLU and much more. However, ReLu has been the most widely used and the well performing activation function for image classification tasks [14].

The resulting output from the final layer in a multi-class classification problem is usually a set of probabilities of the image being classified into that particular class/category. This is achieved by employing Softmax as the activation function in the final layer of the network. All the output class probabilities for a particular image sum up to 1. If it is a binary classification problem, the output is a 0 or 1. Since the study involves multiple classes or categories of yoga postures, the Softmax activation function was used throughout the work.

With this hierarchical structure, the CNNs are trained iteratively by assigning weights to the features being learnt from the data and updating it using standard back-propagation algorithms for a required number of iterations [15]. There are numerous training parameters for a CNN that can be adjusted based on the problem in hand, that will help the low-level features to evolve over the layers and the iterations to more complex features.

2.4 Transfer Learning

Although traditional CNNs have attracted a lot of attention since its inception, in recent years there has been a lot of challenges with varying data formats, availability of data for training models and so on. This has lead to the creation of many variants of the CNN models. Apart from the CNNs built from scratch, one variant that is gaining a lot

of attention are per-trained CNNs [16] otherwise referred to as transfer learning models. Especially, when it comes to dealing with very small datasets transfer learning models have been very successful with image classification. Currently, there has been a lot of interest in using pre-trained networks , to accomplish a variety of classification tasks in any domain. Especially, for tasks that lack a comprehensive amount of labelled data to train a deep neural network [17].

Building and training a CNN model from scratch in many cases requires huge amounts of data, high computational resources and hours or even days of training time. Collecting huge amounts of domain specific data is an expensive and challenging process with respect to real world applications. To overcome this challenge, many researchers studied the impact of applying previously learnt knowledge to assist in learning new knowledge through similarity and connection. This led to the concept of transfer learning that refers to a process where a CNN model that was trained on one problem is used in some way or another related problem.

The only difference between using a pre-trained CNN and a conventional CNN architecture discussed in the previous sub-section, is the former represents a very complex model that includes repeated and additional layers built by training on a huge set of data. This training involves the fine-tuning of numerous parameters and hence it takes a very long time to build such models. However, the fully built model serves as an excellent framework to be incorporated with any other related problem and reduces the computation time and the effort of building a model scratch. Many groups of researchers and academic groups have leveraged such pre-trained networks for varying image classification tasks and have reported a general success in doing so [18–20].

There are two approaches that are mainly employed when using transfer learning [21] as shown below. Figure 2.3 visualizes both the transfer learning approaches for a classification task of images.

- Transfer Learning through Fine tuning
- Transfer Learning through Feature extraction

In the **Fine tuning** approach, the weights in some layers of the CNN are preserved or left unchanged, while the remaining layers are tuned to adjust to the new task or dataset. Usually the initial layers are the ones that are left unchanged as the features obtained from these layers are more generic and high-level abstractions that will be applicable to other tasks. Since the final layers provide more information on specific in-detail features, these layers benefit from the fine tuning approach and will be adjusted to suit the targeted dataset.

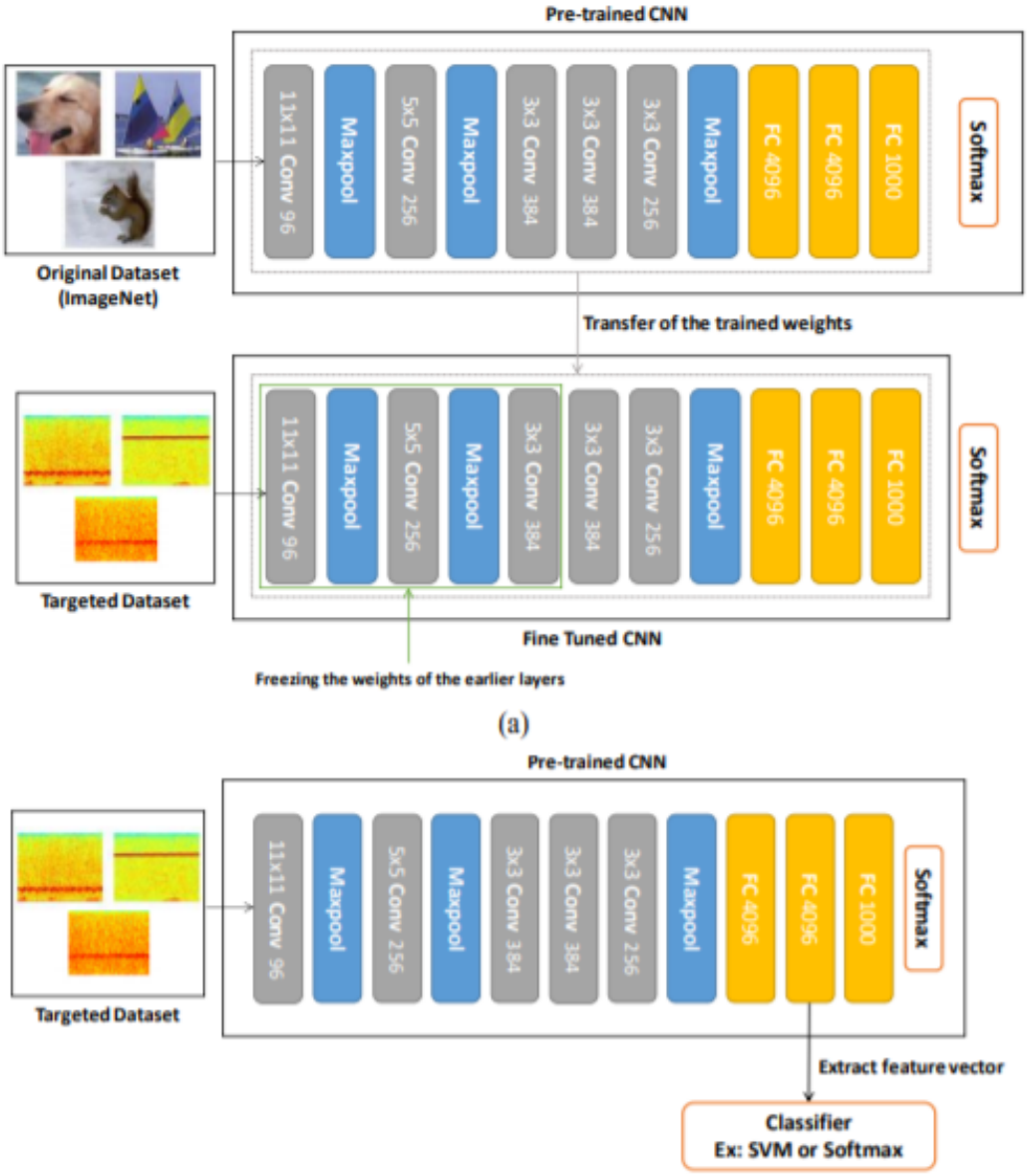

FIGURE 2.3: Transfer learning approaches[21]

With respect to the **Feature Extraction** approach, the model weights in all the layers are left unchanged. The idea is to use these unmodified weights to access the features learned and generated from the new data at the end of all or some of the pre-trained layers, and use these encoded feature rich data to train any standard or softmax classifier. This doesn't include any fine tuning, adjusting the convolutional layers or weights from the pre-trained models.

2.5 Handling Overfitting

The goal of a good neural network model is to generalize well from the training data to any new data in the same problem domain. However, if the details and the noise that is learnt from the training data is too specific, then it negatively impacts on new data and is termed as Overfitting. In other words, overfitting is a phenomenon, where a network learns a particular function with very high variance so that it can perfectly model the data. Overfitting can lead to poor model performance, especially when the data to train is scarce. An overview of the techniques that seek to reduce overfitting is given below.

2.5.1 Dropout

Dropout is a technique when applied to layer, randomly turns some of the neurons in that layer to be inactive based on a threshold parameter. The parameter specifies the probability to determine which nodes remain inactive. Dropout reduces the dependency on the inputs being received by the next hidden layer. By applying the technique of randomly turning off a neuron, the over-reliance on a particular input is reduced which results in learning from only a random set of input, improving generalization.

2.5.2 L1/L2 Regularization

In neural networks, there is a relationship between the complexity of the model and the magnitude of the weights between the layers. In other words, while we execute many iterations through a deep network, some of weights tend to become quite large where it starts to overfit the data. To overcome this, the Regularization technique is used to reduce the magnitude of the weights that is being updated using back propogation algorithms. L1 or Lasso Regression and L2 or Ridge Regression are the two types of Regularization techniques. The only difference between L1 and L2 regression is that the former uses just the sum of the weights during the minimization process whereas the latter uses sum of the squared weights. From Andrew Ng's paper [22], L1 is efficient when the training data contains a lot of irrelevant features, otherwise L2 performs comparatively better..

2.5.3 Data Augmentation

While working with images and Deep learning models, the amount of data to be used to train the model plays a very important role. The more data we have the better the performance of the model[[23], [24]]. However, it might not be possible to access

huge amounts of data in every domain. Very small amounts of data increase the risk of overfitting the data. This results in reducing the generalization capabilities of a model. Data Augmentation is a technique that overcomes this problem and also helps to overcome the translational in-variance that arise due to different image scales, lighting, occlusion, rotation and more. In most of the image recognition taks, data augmentation is used as one of the pre-processing steps to prevent the model from overfitting.

Data augmentation helps to generate affine transformations of an image that increases its diversity without creating the need to collect huge amounts of data. From works done by [25] and [26], it clearly shows that data augmentation works towards improving the performance of deep learning models particularly when approaching a problem with a small dataset.

2.6 Image Recognition

With respect to image recognition, deep learning models have been used for a variety of tasks, majorly contributing to the healthcare industry to diagnose an illness or any other problem and has achieved in producing successful models. In [27],an automatic brain tumour segmentation model was built based on deep convolutional neural networks which achieved a very good performance in terms of accuracy and computational speed out performing the state-of-the-art methods. Similarly, George et al. [28] proposes a system to diagnose breast cancer based on classification of cytological images and has reported accuracy rates ranging from 76% to 94% on a considerably small dataset. Following the numerous studies in the healthcare industry, other industries related to automobiles, entertainment, security and retail have also had many researchers building successful deep learning models for image recognition tasks.

2.7 Activity recognition

In recent years, human activity recognition has also been a very popular area of study as it has gained a lot of importance with real word applications like surveillance, criminal investigation, health activities and so on. [29] proposes a model built using pre-trained CNNs to classify on a dataset consisting of sports actions, such as jogging, running, diving, kicking etc. and has reported accuracy rates ranging from 91% to 98%. Another work [30] done by A Tang et al. proposes a hand posture recognition system that was built using CNNs that can be used for sign language recognition. A yoga tutor project was proposed by Patil et al [5] that uses SURF algorithm to detect the difference in

postures between an expert and a user practising yoga. Although, this system aims to practice yoga correctly, an approximate comparison of postures by only using contour information will not suffice. The proposed thesis study will be positioned in the area of human activity recognition for benefiting from health activities with respect to image classification.

2.7.1 Pose Estimation

Activity recognition tasks can be difficult if the images contain a lot of background data in addition to what is required for learning process. In many studies, to extract the required pattern of data or to extract the features related to the human activity, the person performing the activity is monitored through wearable sensors [31–33]. However, it is not possible to obtain sensor based data for every activity. Moreover, the advancements in Computer vision and Deep Learning has made it more flexible to extract such patterns from raw data.

In [34], a system was proposed to extract features from the relational graph of upper body poses of an Indian dance form called Bharatanatyam. The system uses skeletonization techniques combined with approaches for matching the shape of human figure to get a good model of the figure that describes its shape in the image. Based on the obtained results, an attributed relational graph is then generated from which feature vectors representing joints and endpoints are extracted and used to discriminate between shapes for classifying Bharatanatyam poses. However, this work was experimented with images having no background clutter and fails to perform as expected if multiple persons are detected. Another drawback is that it only studies the recognition of upper body parts like head, torso and arms failing to detect the lower parts of the body.

A pose estimation approach applied to detect the entire human body was proposed by [35], and involves matching template images to targets based on local image features and a novel convexification scheme that shrinks the regions to match to a small target area. This approach incurs high computation costs if the templates for matching become more complex and depends on the availability of representative training images that contains similar appearances. In 2018, a comparative study focusing on different 2D pose estimation techniques was published, done by Mohan et al. [36] that shows the advantages and drawbacks in methods such as skeletonization, silhouette analysis-based pose recognition, Shape context matching, segmentation and other feature recognition tools. However, these algorithms are highly dependent on the dataset and the environment in which the images were captured.

2.8 Current State of the Art

2.8.1 Yoga Posture Recognition

In order to build a good image classification model to be used in the current study, certain state of the art data transformation techniques and classification approaches were studied and implemented based on the following literature survey. In [37], a self-training system was developed using Adaboost algorithm and was trained by an expert yoga trainer resulting in an accuracy of 94.78%. However, the images used for training the model was captured using depth sensor-based camera which might not be available to many users. A yoga self training system introduced by Chen et al. [6] assists in rectifying yoga postures for 12 different asanas when practised using a Kinect depth camera. The approach involves manual feature extraction and building an individual model for each asana which increases the computational costs.

A lot of recent works since the advent of [38] has combined deep neural network approaches with pose estimation methods to detect human activities. In [39], an approach for yoga recognition from realtime videos has produced excellent results in terms of accuracy, by using OpenPose to extract the pose information from the data. This feature data is then fed into a hybrid deep learning model of Convolutional Neural networks(CNN) and long short-term memory(LSTM) for classification. The proposed approach by [40], combines OpenPose and CNN to build a performance evaluation system that classifies the correctness of a Yoga Pose by calculating the angle differences of the body between a learner and an instructors pose. The effectiveness of this approach was confirmed by its application to the real world scenario. The undertaken thesis study is highly focused on classifying yoga images by using OpenPose for feature extraction and CNN to train the prediction model.

2.8.2 Pose Estimation

Identifying individual body parts or the shape of the human body can provide more specific knowledge in the learning process during model training. By doing this, we feed only the data that adds value to the learning process and ignore all the background noise in the input images. In order to highlight and extract such important features to the classification model a variety of different techniques have been studied over the years.

All the conventional approaches focusing on skeletonization was replaced by methods based on Deep learning since the advent of Toshev's DeepPose [38]. DeepPose uses neural network based models to directly regress on the the joint coordinates and predict

the location of human body parts. However, it suffers from localization problems [39]. An approach to overcome this was suggested by Cao et al. [1] which focuses on 2D multi person pose estimation in images and videos using Part Affinity Fields. This is a major revolution in the pose estimation field and it is a publicly available library that can be easily implemented on diverse computer operating systems. It can also be effortlessly combined with various deep learning frameworks [41]. This approach is used in the current study to efficiently and accurately identify the body parts of the person doing yoga, thus ignoring any background noise in the images.

2.8.3 Data Classification

Conventional CNNs and Transfer learning models are still being used in a lot of recent studies for image classification. However, Transfer learning is gaining a little more attention as it reduces the effort in building a CNN from scratch. With respect to the classification of Yoga postures, most of work done uses Conventional CNNs. Hence an implementation of a conventional model will be a good place to start with in this study. A more interesting analysis will be to analyze the impact of using pre-trained models which has promised very good results with image classification using very little data.

There are numerous pre-trained model architectures that has been developed by many researchers and academic groups that were trained to perform image recognition tasks. Some of these models that are readily available with Keras(a front-end tool to build deep learning models), is given below.

- Xception
- VGG 16
- VGG 19
- ResNet, ResNetV2
- InceptionV3
- InceptionResNetV2
- MobileNet
- DenseNet
- NASNet

All the above models were trained on the same dataset as part of the ImageNet Large Scale Visual Recognition Challenge. The goal of this challenge was to create a model that correctly classifies images into 1000 different object categories. The dataset consisted of close to 1 Million training images and 100K test images. These images includes any kind of data like cats, dogs, vehicles, household objects, activities and much more [42]. The results reported from the ImageNet challenge serves as a benchmark and is considered state of the art for any novel computer vision classification algorithm.

There are many studies in different domains that use pre-trained models for image classification tasks. In [43], the author shows the implementation of the MobileNet model for the image classification of garbage into categories like metallic, cardboard, wet waste, dry waste and so on. The model has reported an accuracy of 87%, and the author has also pointed out the popularity with other pre-trained models such as VGG16 and VGG19. Another work demonstrated in [44] reports a 95% accuracy on malaria cell-image classification with ResNet-50. In [45], a comparison of image classification using VGG16, VGG19, InceptionV3 and ResNet models is outlined, with Inception outperforming the remaining models.

In the proposed thesis, some of the popular and accessible pre-trained models were used to analyze the performance on the yoga posture classification process. In addition to implementing transfer learning models, a conventional CNN was also implemented to observe its behavior in handling the problem as they are still popular in the current field of work.

Chapter 3

Design and Implementation

This chapter provides an overview of the problem being researched, along with the design and implementation of the solution to approach this problem.

3.1 Problem Definition

The purpose of this thesis is to build a deep learning model to recognize different yoga postures or asanas performed by different people. More specifically, this study focuses on the classification of 10 different yoga postures from images. A lot of research is still being performed in this particular field as physical fitness through yoga has gained a lot of popularity. However, it is very important to learn and reproduce yoga postures accurately. While learning and practising yoga, if the correct body alignments are not followed, then the individual might face serious consequences in the form of body ache or breakdown of nervous system. Hence learning and performing a particular yoga posture accurately is vital. This study focuses on recognizing yoga postures accurately as it will help in building self training systems for people to learn yoga in the comfort of their homes and in the most inexpensive way.

3.2 Proposed Approach

The overall image recognition process as shown in Figure 3.1 and can be divided into 3 main stages; Pose Estimation, Data augmentation and Classification using CNNs. As outlined and discussed in Chapter 2, estimating the human figure from the image is very important in this task as it helps to improve the classification process by removing unwanted background noise in images. The data augmentation step will help us to

enlarge our dataset as it is very difficult to collect a huge amount of diverse data as required by the training process. The variant images generated using the augmentation can be a flipped image, cropped image, rotated image or a zoomed image of the original version. The pose estimation and data augmentation are very important pre-processing steps that will add a lot of value to the final process, which is the classification of image. In the classification stage, we will be training the pre-processed images using Convolutional Neural Networks(CNNs). As discussed in 2.3, there are two types of CNN architectures that can be implemented, the conventional CNN or the transfer learning models. We will be using both the approaches in this study considering that our dataset is very small. This will help us evaluate as to which model type handles the given data accurately.

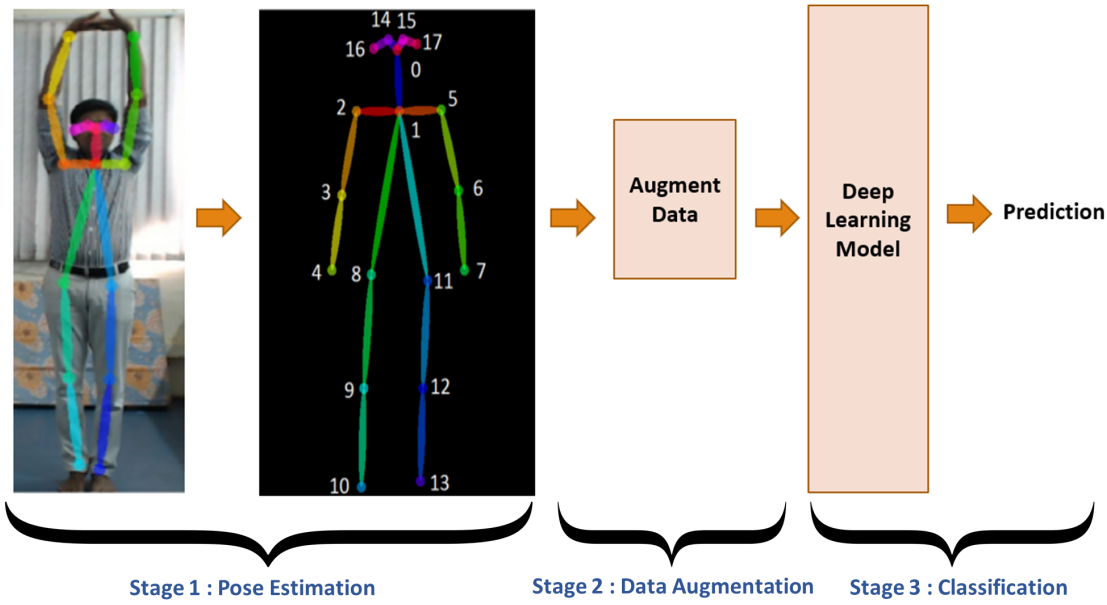

FIGURE 3.1: Proposed Overall Architecture

3.3 Implementation Framework

3.3.1 Hardware Requirements

Graphics Processing Unit(GPU) is the single and most important requirement when building any Deep learning model. Training a deep network requires high computational power to perform complex matrix calculations that gradually build the model. GPUs provide the necessary computational power and thereby reduce the training time to a great extent when compared to the standard CPU executions. Accessibility to such GPUs are provided by cloud services such as Google Cloud Platform, Google Collab,

FloydHub and many more. Out of all the services, Google Collab is most easily accessible and the most popular platform to work with deep learning frameworks.

3.3.2 Software Requirements

The Development Environment consists of an IDE, much like any other Software Development. For our project, the IDE used is Google Collab Notebook as we will be utilizing its accessible GPU enabled services. Although there are many programming languages to implement a deep learning solution such Python, R, C and C++, we will be using Python for the current project as it is one of the most popular languages and has got the support of a variety of tools and libraries to build deep learning models.

3.3.3 Deep Learning Framework Requirements

3.3.3.1 Back end

TensorFlow is the most used open source backend library in Deep Learning. The complex lower level calculations behind every layer in the deep learning network is carried out by TensorFlow. Similar to Python, TensorFlow has a variety of support networks with respect to building Deep learning models. Although there are other frameworks such as PyTorch and Apache Spark, the seamless integration of TensorFlow with Python and the front end tool Keras, makes it the most logical choice [46]. The TensorFlow version that will be used is TensorFlow 2.2.0.

3.3.3.2 Front end

Keras is a front end API that was built on top of TensorFlow to make the development task easier as the latter is extremely verbose in its implementation. Keras offers a lot of tailored in-built functions that supports in interfacing the backend network with the real world data.

3.4 Solution Architecture

3.4.1 Image Pre-processing

The pre-processing of raw data of any form, plays a crucial role in deep learning algorithms. Feeding feature rich data to the image classification models is compulsory for

achieving better performance when it comes to human activity recognition [47]. Raw data when applied to any classification model does not achieve a good performance and this has been analyzed by KK Pal and KS Sudeep [48] using the images from the CIFAR-10 dataset. Especially, human activity recognition from images can be very difficult when the image contains a lot of background noise in addition to the actual human activities that needs to detected. In order to build a good image classification model to be used in our current study, certain data transformation techniques were implemented as given below.

3.4.1.1 Pose Estimation using OpenPose

The OpenPose framework that is used for the pose extraction in itself has a multi convolutional neural network(CNN) based architecture that focuses on identifying the body parts of the person doing yoga. It uses part affinity scores to jointly learn and associate the identified body parts. This results in a RGB skeleton image, mapping the different joints to different colors based on the limbs it represents.

The images are fed into the multi convolutional neural network architecture of OpenPose [1] that is depicted in Figure 3.2. The network is a "two-branch multi-stage" CNN where the two branches are indicated by the beige(top) and blue(bottom) colors in the figure and the multi stage refers to the concept of using the two branches stacked over one another in each stage of the network. The top branch predicts the confidence maps of body part locations such as shoulders, eyes, limbs etc. The bottom branch predicts the part affinities, which is the association between the parts detected. The initial set of confidence maps and part affinities are produced in the first stage. In subsequent stages, the predictions from the previous stage branches are concatenated with the original image features to produce better refined predictions. After the final stage, the confidence maps and affinity fields are processed using greedy inference to generate the key-points of the person in the image. OpenPose will be able to generate a frame with upto 25 key points, however in this study it was used to estimate only up to 18 key-points as the remaining key-points add redundancy in generating extra points in the feet and hands.

After passing through the overall pipeline of the OpenPose architecture, the final resulting image is a refined version of the original image with indication of the person's body parts against a black background as shown in Figure 3.3.

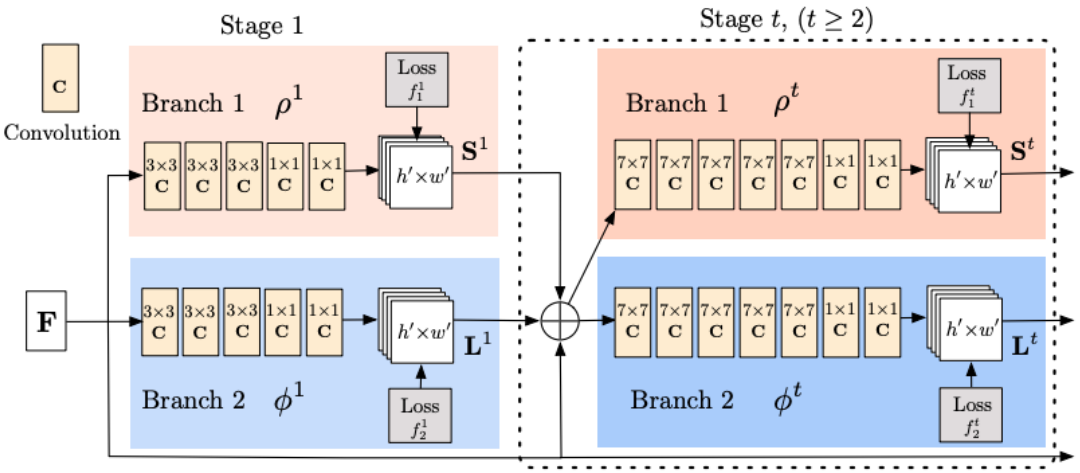

FIGURE 3.2: OpenPose framework architecture [1]

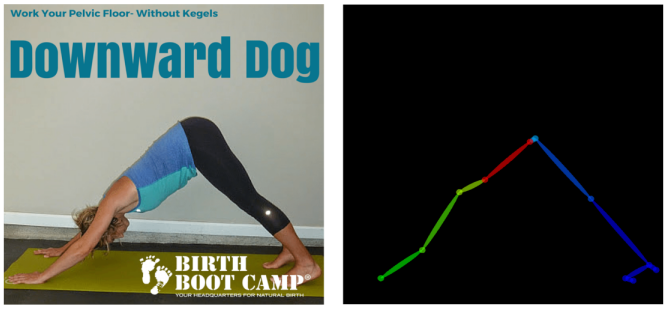

FIGURE 3.3: Original Image and estimated posture from OpenPose

3.4.1.2 Image Augmentation

As discussed in Section 2.5.3, data augmentation is necessary to handle the challenges of overfitting in any from of neural networks. Since we are dealing with a small set of images for the training process, the augmentation process is a necessary pre-processing step before feeding the images into the classification model. It also helps to boost the performance of the model to a certain extent.

The skeletal key-point images obtained from OpenPose serve as input to the augmentation process. We will be using the Keras library data augmentation functions to flip, re-scale and zoom an image and create an augmented set of images. The images in Figure 3.4 and Figure 3.5 shows the original image and the augmented versions for the corresponding OpenPose output image. This technique will increase the size of the training set and improve the model performance to generalize between the postures. The generated transformed versions of the original image are automatically mapped to the class to which the original images are assigned. This process is applied only to the data that will be used for the training process and not to the one used for evaluating the performance of the model as we want to maintain the authenticity in predicting real world

data. The increased set of training images are then passed through the classification process to learn from the transformed set of data.

FIGURE 3.4: Original Image of the triangle pose

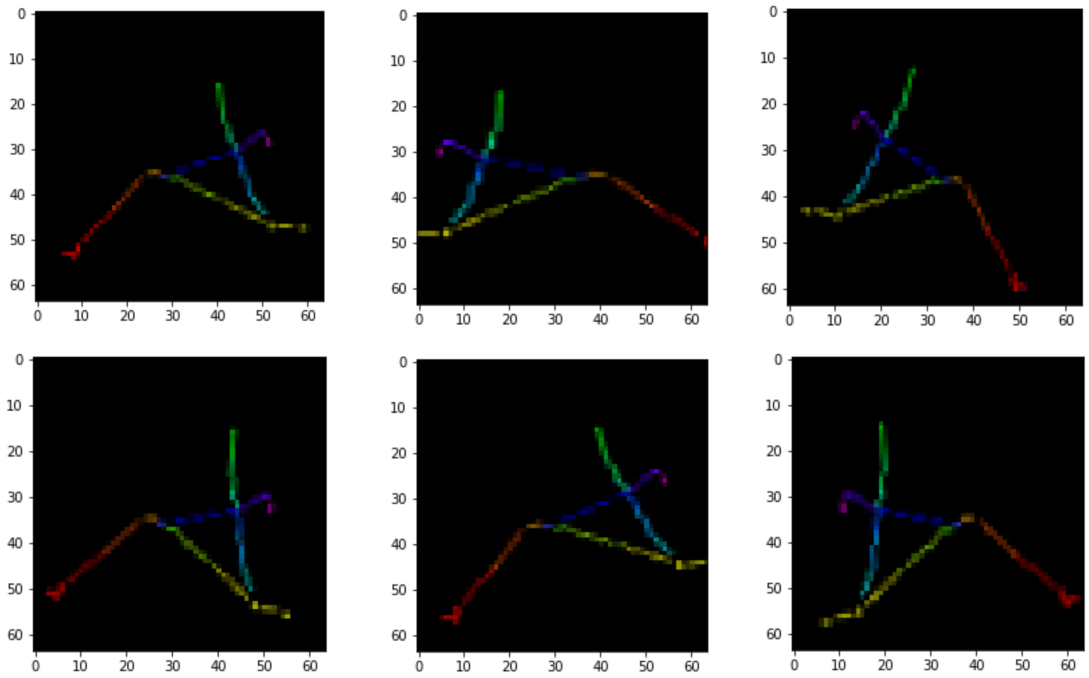

FIGURE 3.5: Augmented versions of the corresponding OpenPose image

3.4.2 Image Classification

The classification task is the final stage of the overall architecture, where the actual learning process starts. The pre-processed images are fed into the classification model that predicts the class or the yoga posture of a particular image. Both conventional

CNNs and transfer learning CNNs have proven to be performing very well with Image classification tasks. However, everything comes to the factor on how these models perform with the current data and with the combination of OpenPose. Hence we will be implementing a conventional CNN model and transfer learning models to carefully choose the one that performs better than the other.

3.4.2.1 Conventional CNN

The building blocks of a conventional CNN was discussed as part of the literature survey. The CNN used in our study will contain a few hidden layers and a final SoftMax layer. The SoftMax layer will be made up of 10 neurons matching the 10 different yoga postures to be identified.

Building a CNN from scratch has been made easy with the help of Keras as a lot of supporting functions are now available that can be used to assemble a model for image classification. The main challenge is to train the model with the correct set of parameters that would learn at its best from the given data. Often, deep learning models tend to overfit during the learning process. Data Augmentation is already included as part of the overall implementation as it has been proven to be a powerful technique to handle overfitting and to leverage the performance with small datasets. Other techniques such as Dropout and Regularization will be experimented and incorporated as required by the models. With respect to deciding between L1 and L2 Regularization, we used the L2 Regularization technique throughout the study as we filter out irrelevant features as part of pre-processing [22]. A few other parameters that will be studied to tune the model includes the number of fully connected layers, number of Epochs or iterations and the learning rate. The resulting output from the final SoftMax layer of the CNN is a set of probabilities of the image belonging to each of the 10 classes or yoga postures. The class with maximum probability is assigned as the predicted class.

3.4.2.2 Pre-Trained CNNs

A number of pre-trained models based on image classification have been created, that aids in training much complex feature intense models with lesser training times and improved performance. In our current study, we will be analyzing the performance from models such as MobileNet, VGG16, VGG19, Xception and ResNet50 which has been outlined and reviewed as some of the popular models with respect to image classification. A graphic of one of the pre-trained model architecture (VGG16) is shown in Figure 3.6. The figure shows the complex design involved in a pre-trained CNN, where multiple convolutional and pooling layers are stacked together followed by fully connected layers.

The difference between all the pre-trained networks lies in the volume of these stacked layers. In other words the volume refers to the number of convolutional and pooling layers in the network.

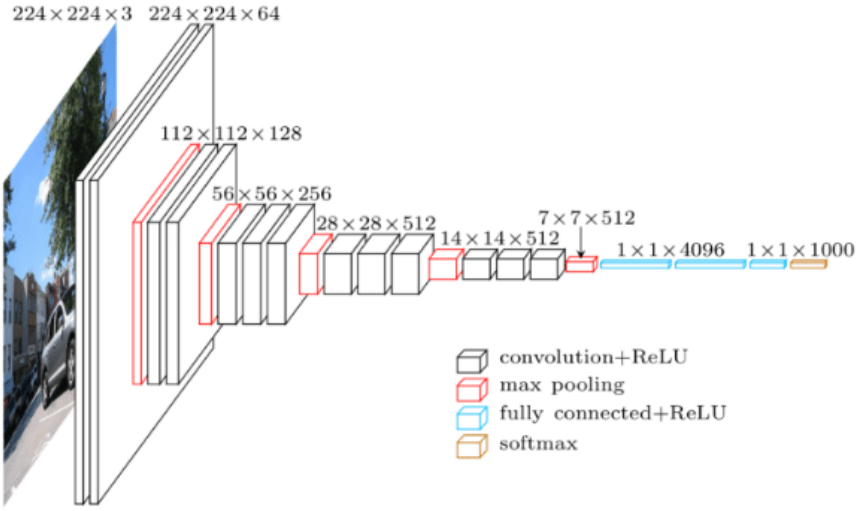

FIGURE 3.6: Architecture of the VGG16 pre-trained model [49]

In this study we will be implementing the **Feature extraction and the Fine tuning approach** associated with Transfer Learning that was discussed in Section 2.3. Adapting a pre-trained model to our current work, involves working with all or some of the convolutional and poling layers. The fully connected layer and the final SoftMax layer will be linked to it based on the current data and the number of prediction classes available.

For the **Feature extraction** process, we will be using all the convolutional and pooling layers to extract the features with the previously loaded weights that was used to train the ImageNet dataset. It means that the pretrained network is used only to extract the abstract information from the images such as edges, curves etc based on its previous trained knowledge. This will result in feature rich data that describes the input images fed into the model. In this case, it describes the augmented images generated from OpenPose. These features will then be passed to the fully connected layer and the SoftMax layers that was designed to suit the dataset used in this study and will be trained by updating the weights associated with these layers. The output is a set of probabilities of the image belonging to each of the 10 classes or yoga postures. The class with maximum probability is assigned as the predicted class.

For the **Fine Tuning** process, a similar approach as given above is followed with a small variation in the usage of the pre-trained layers. For fine tuning, we will partially use some of the lower convolutional and pooling layers as part of the training process to incrementally learn from our data. This means that unlike the previous approach, where

all the layers use pre-trained model weights learned from ImageNet, in this approach the weights of some of the bottom layers will be updated along with the fully connected and SoftMax layers.

Chapter 4

Evaluation and Experimental Results

4.1 Experimental Setup

4.1.1 Dataset

The dataset used for training and evaluating the deep learning model contains a total of 712 images that captured people doing 10 different yoga postures. The different yoga postures or classes used for classification were bridge, childs, downwarddog, mountain, plank, seatedforward, tree, trianglepose, warrior1 and warrior2. The data was originally collected by a Georgia Tech Researcher Anastesia Marchenkova, who initiated an approach for classifying yoga postures using Deep Learning [50]. The images were not captured with restricting backgrounds and hence will be cluttered in factors such as attire, background colour, gender, age etc. However, all the images in our dataset are single-person images.

4.1.2 Data Cleansing

The images in the dataset were all segregated class-wise, however there was an imbalance or inconsistency in the images between a few classes. The class mountain and downward-dog contained a few images that were flipped and hence were redundant to the original image. Such images were removed to maintain a consistent format of data across all classes and to overcome the imbalance in the data. These flipping operations will be ultimately introduced in the data augmentation phase of the proposed architecture.

4.1.3 OpenPose Implementation

The first step in studying the proposed approach is pre-processing or transforming the set of input images in the dataset for the classification process. To do this, the OpenPose Framework was initially setup using the steps provided in OpenPose Setup, and all the 712 images in the dataset were processed against it. The Output from this implementation is a set of images (712 images) with the extracted human figure against a black background along with a set of key-point coordinates as a JSON file for each image. We will be using only the retrieved image data for further processing. However, during the implementation it was found that not all images were valid from the OpenPose output. Some images were not recognized accurately by the OpenPose library and hence resulted in empty image frames. Such images will also be removed from the original dataset as it does not add any value to the training model.

4.1.4 Classification Setup

Any classification process with deep learning involves a train and a test set that will be used for learning and evaluating the model respectively. For the proposed architecture, the OpenPose output data was evaluated over an 80%-20% train and test split with 5-Fold cross validation. A different set of images was used as the test set for every training fold. This enables us to study the model and its behaviour to varying images and its characteristics without any bias. In this study, the test split will act both as the validation and test data, as there are limited number of images to train.

With Keras, the Data Augmentation and the classification is a coupled process that will be implemented together sequentially in batches. The augmentation methods from Keras include zooming the image, rotating the image, flipping the image, changing the shear angle of the image and normalising the image. All the available selection methods were used to induce as much as diversity as possible in the learning process. It is important to note that these augmentation methods were only applied to the training set and not to the test set as we want to maintain the authenticity in predicting real world data. However, normalizing is done to both the train and test data.

4.2 Conduct of Experiments

Some of the parameters or hyper-parameters that were used to study the training process for our proposed model, include the following [51].

- Network Architecture - This determines if we use the conventional CNN model or the transfer learning model
- No.of Epochs - The Epochs refer to the number of iterations for which the network will update itself and learn from the data.
- Number of hidden layers in the network - This refers to the number of fully connected layers before the final SoftMax layer
- Learning rate - The learning rate is the step size in each iteration while moving towards minimizing the loss function. These parameters will be studied and tuned as required to boost the performance of the model
- Batch size - The training data is split into small batches during the learning process in order to make models to be computationally efficient and to escape from local minimums [52]
- Regularization parameters - The L2 regularization technique was used to handle overfitting. The model with and without regularization was studied
- Dropout Dropout layers are also used to handle overfitting. The model with and without Dropout was studied

The batch size parameter was kept constant throughout the experiments in order to ensure that the data was evenly split between all batches during training and to make full use of the test data [53]. The number of Convolutional layers used were referred from '[2].All the remaining parameters were tuned to boost the performance of the model at its fullest. The training and validation accuracy and loss values were recorded from the experiments that were conducted.

With respect to studying the Conventional CNN, a set of experiments was executed based on the fine tuned parameter settings without Dropout or Regularization. The same set of experiments was repeated with L2 Regularization and with Dropouts individually to study its impact and to choose the best performing technique for the model.

For studying the transfer learning model, we will initially evaluate a few selected pre-trained models with the data to see which model best learns from the given data. the model architectures included in the study are MobileNet, VGG16, VGG19, ResNet50 and Xception. The best performing model was then tuned based on some of the above listed hyper-parameters as an attempt to improve the model performance. Both the transfer learning approaches of feature extraction and fine tuning was used to choose the best performing model. The related code can be referred in Experiments and Code.

4.3 Results and Observations

4.3.1 Evaluation criteria

A similar strategy as demonstrated in [39, 54, 55] was followed in this study. The performance of a model is based on how accurately the trained model recognizes a yoga posture from a new set of data. In other words, the accuracy of the model on the test set determines a model's performance and is determined as the ratio of correctly classified images over the total number of images. The higher the accuracy, the better the model is, in terms of performance. In addition to the accuracy, we also consider the test loss which determines how well the model has learnt to perform this image recognition task. The loss here refers to the output from the SoftMax loss function which is used to determine the final probabilities of the image belonging to each of the 10 classes.

Keras makes it easy by recording the loss and the accuracy after every Epoch in a particular model for both the training and the test or validation data. The model performance can be determined based on the reported values from the final Epoch. The recorded results for an experiment is the average accuracy and average loss for both the training and the test set across the 5-folds.

4.3.2 Results from Conventional CNN

The initial experiments were conducted with a Conventional CNN or CNN built from scratch. The results for the experiments are shown in Tables 4.1, 4.2 and 4.3.

The Table 4.1 shows the results where the model performance was studied without involving any Dropout or regularization technique. Table 4.2 shows the results for the same set of experiments with L2 Regularization and the Table 4.3 shows the results for similar experiments with Dropout with certain changes to the epochs for the last four experiments based on the observed results from the first four experiments. A Dropout rate of 0.5 was found to be the best value for the given dataset.

From Table 4.1 results, it can be observed that the model is significantly overfitting as the training accuracy is very high but the test accuracy is quite low. A similar pattern is observed with the training and test loss. This large gap between the training and test values indicates that the model is trying to learn too well on the specifics of training data and hence loses its generalization capability resulting in a lower accuracy with new data from the test set. To overcome this trend, the regularization and Dropout strategies were studied.

Without Dropout and Regularization							
Exp. No	**Epochs**	**Learning Rate**	**No. of hidden layers**	**Accuracy**		**Loss**	
				Training	**Test**	**Training**	**Test**
1	**50**	0.001	1	0.90	0.77	0.27	0.93
2	**70**	0.001	1	0.91	0.78	0.24	1.02
3	**90**	0.001	1	0.95	0.80	0.13	1.0
4	**120**	0.001	1	0.95	0.80	0.13	1.12
5	90	0.001	**2**	0.94	0.81	0.16	0.99
6	90	0.001	**3**	0.94	0.81	0.16	1.04
7	90	**0.0001**	2	0.83	0.75	0.48	0.82
8	90	**0.01**	2	0.11	0.12	2.30	2.30

TABLE 4.1: Results from Conventional CNN without Dropout and Regularization

With L2 Regularization							
Exp. No	**Epochs**	**Learning Rate**	**No. of hidden layers**	**Accuracy**		**Loss**	
				Training	**Test**	**Training**	**Test**
9	**50**	0.001	1	0.87	0.75	0.62	1.1
10	**70**	0.001	1	0.9	0.79	0.54	1.06
11	**90**	0.001	1	0.91	0.79	0.50	1.07
12	**120**	0.001	1	0.91	0.78	0.46	1.06
13	90	0.001	**2**	0.90	0.79	0.51	1.03
14	90	0.001	**3**	0.90	0.79	0.59	1.04
15	90	**0.0001**	2	0.87	0.74	1.58	2.02
16	90	**0.01**	2	0.38	0.35	1.74	1.9

TABLE 4.2: Results from Conventional CNN with Regularization

With Dropout							
Exp. No	**Epochs**	**Learning Rate**	**No. of hidden layers**	**Accuracy**		**Loss**	
				Training	**Test**	**Training**	**Test**
17	**50**	0.001	1	0.85	0.8	0.41	0.74
18	**70**	0.001	1	0.89	0.81	0.29	0.78
19	**90**	0.001	1	0.94	0.81	0.19	0.88
20	**120**	0.001	1	0.93	0.81	0.20	0.93
21	50	0.001	**2**	**0.81**	**0.81**	**.52**	**0.68**
22	50	0.001	**3**	0.82	0.8	0.53	0.74
23	50	**0.0001**	2	0.53	0.67	1.3	1.07
24	50	**0.01**	2	0.11	0.11	2.29	2.30

TABLE 4.3: Results from Conventional CNN with Dropout

The results from Table 4.2 with regularization, shows that the gap between the training and the test values mildly reduces, however this has not improved the test set performance in any way. With Dropout applied, the results from Table 4.3 shows an overall

improvement over the test accuracy and test loss. More specifically, the experiment no(s). 17, 18, 21 and 22 has shown an improvement in comparison to the previous set of results in terms of both the accuracy and loss with lesser overfitting patterns. The experiment no. 21 shows the best performing model with a lesser test loss and a higher test accuracy when compared to all the models.

4.3.3 Results from Transfer Learning based CNN

The first set of experiments was done to choose the pre-trained CNN model that performs comparatively better with the given set of images. The pre-trained models that were used for the selection process include MobileNet, VGG16, VGG19, ResNet50 and Xception. These models were implemented with feature extraction and fine tuning, which are the two approaches associated with transfer learning and the best model was chosen based on the best accuracy. All the experiments were executed with similar parameter configurations (Epochs = 90, Initial learning rate = 0.0001, No. of hidden layers = 1, With Dropout). Figure 4.1 shows the test accuracy of all the models with both the transfer learning approaches.

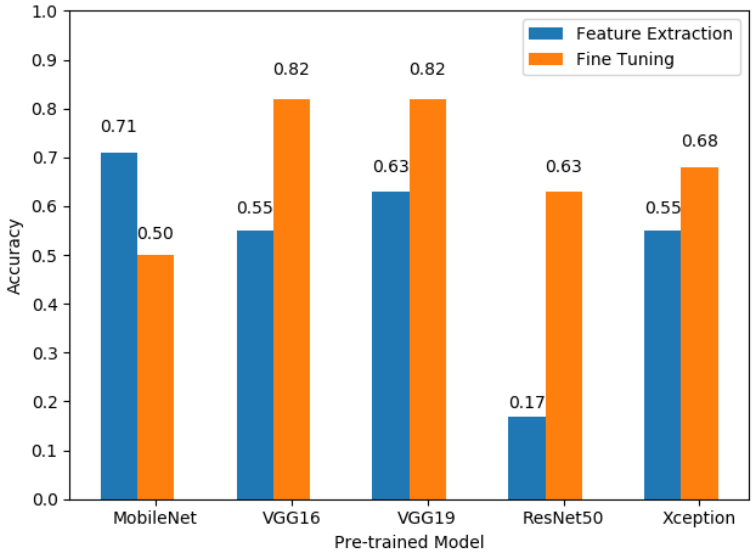

FIGURE 4.1: Transfer learning model performances with Feature extraction and Fine tuning approach

There are two important observations from Figure 4.1. Firstly, it depicts that the fine tuning approach is better than the feature extraction approach, as its related accuracy for all the models is higher except for MobileNet. Secondly, VGG16 and VGG19 has outperformed all the model architectures and has reported a very similar accuracy of 82%. We break the tie with respect to the model that has the lowest validation loss.

The reported loss for VGG16 and VGG19 are 0.73 and 0.86 respectively, which decides that VGG16 is the model to work with for the remaining part of this study.

VGG16 with Dropout							
Exp. No	**Epochs**	**Learning Rate**	**No. of hidden layers**	**Accuracy**		**Loss**	
				Training	**Test**	**Training**	**Test**
25	**50**	0.001	1	0.87	0.77	0.39	0.77
26	**70**	0.001	1	0.91	0.8	0.3	0.71
27	**90**	0.001	1	0.94	0.82	0.19	0.73
28	**120**	0.001	1	0.94	0.81	0.18	0.76
29	90	0.001	**2**	0.86	0.81	0.41	0.76
30	90	0.001	**3**	0.77	0.8	0.67	0.71
31	90	**0.0001**	1	0.92	0.82	0.23	0.65
32	90	**0.00001**	1	**0.90**	**0.82**	**.29**	**.60**

TABLE 4.4: Results from pre-trained VGG16 with Dropout

The VGG16 model was tuned with the hyper-parameters involved in this study and the results are shown in Table 4.4. A similar set of experiments as done with the Conventional CNNs was studied with the VGG16 model. The dropout was included throughout, as its impact in reducing overfitting was observed from the previous results. When applying L2 regularization to this model, it does not improve the performance in any way to what is already shown in the table. The experiment no.32 reported the highest possible accuracy of 82% and the lowest loss of 0.60, and was determined as the best performing model in this study.

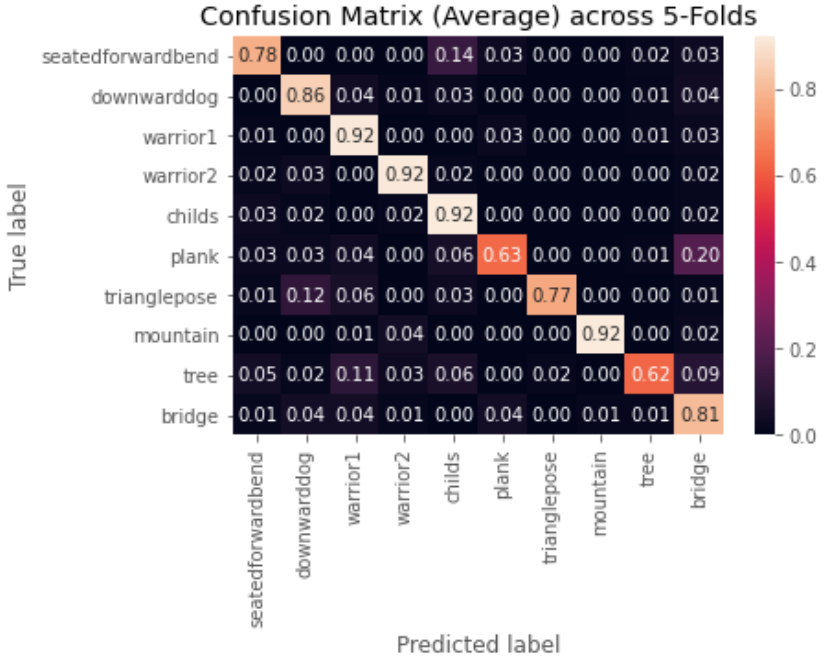

FIGURE 4.2: Confusion matrix for the best performing model

The confusion matrix for this model is shown in Figure 4.2. The higher density across the diagonal indicate that most of the yoga postures were predicted correctly. However, the plank and tree postures show a comparatively higher number of mis-classifications leading to the accuracy of 82%. The Mountain pose, Warrior1, Warrior2 and Child pose show the highest class-wise accuracy. The best performing model accuracy and loss across 5 folds is shown in Figure 4.3. The model for Fold1 and Fold3 slightly overfit the data, however the overall performance has been good considering the small size of training data.

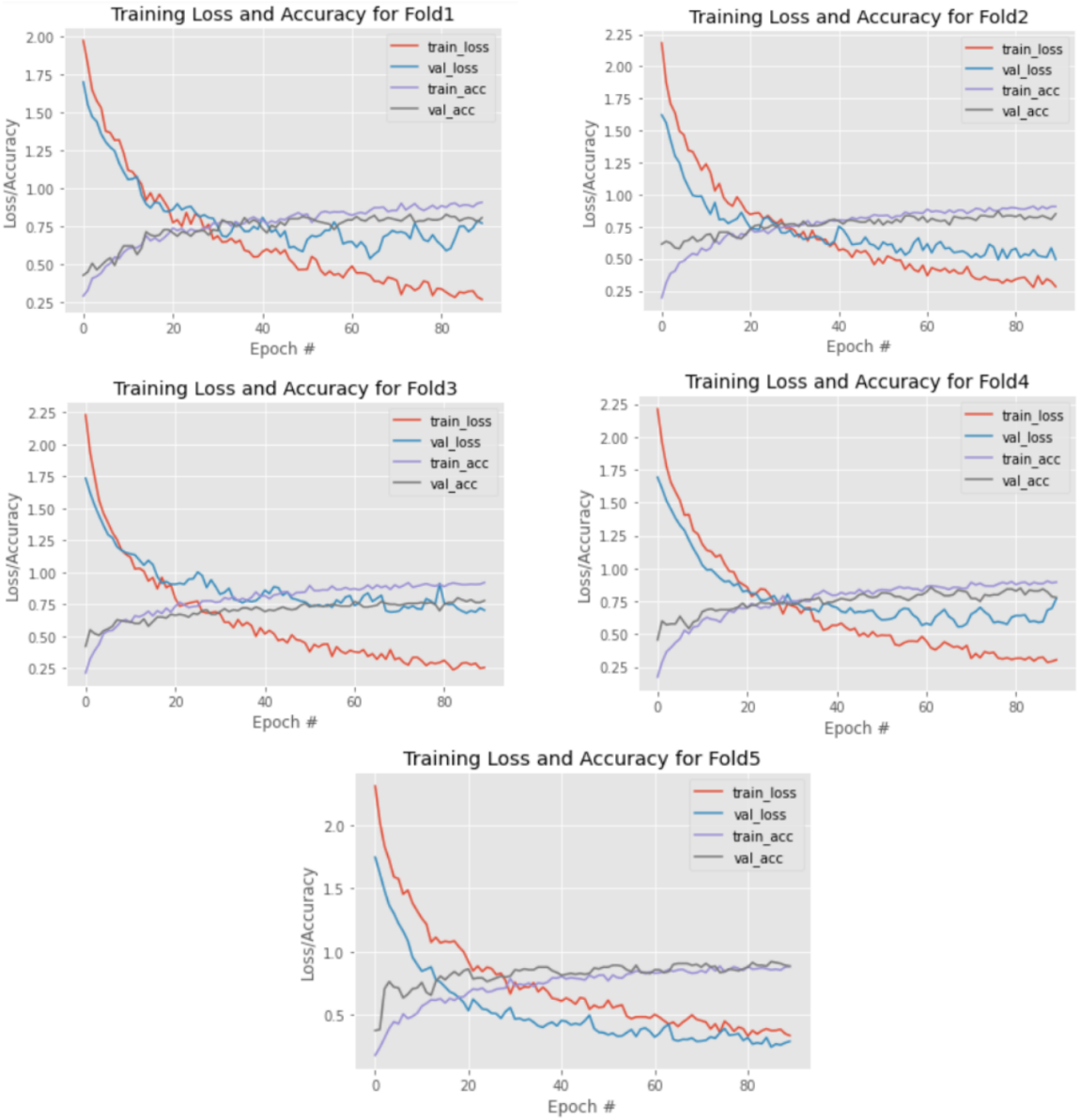

FIGURE 4.3: Test Accuracy and Loss of the best model across 5-folds

Chapter 5

Conclusions and Future Work

5.1 Discussion

When comparing the performances from the conventional CNN and transfer learning models, it can be observed that both the approaches almost converged to the same point in terms of the accuracy. Even a simple CNN has proven to be a powerful feature extraction mechanism in image based classification. However, the transfer learning approach has added a very small performance boost to the final model. Although a couple of Folds still slightly overfit, the overall performance has been good. Also, a lot of techniques such as Data Augmentation and Dropout has majorly contributed to the performance of the model.

In [37], a self training system that was developed by Trejo et al. reported an accuracy of 94%, but the images used in the training process were captured using specialized devices which is not a very feasible model that everybody can use. Many studies that concentrated on recognizing yoga postures from videos [39] [6] have reported excellent performance but the significant success behind these approaches was the manual data collection procedure which resulted in a large amount of data for the training process and therefore an improved prediction process. When matching the results from this study to the available state of the art evaluations, it is to be noted that there is still room for improvement.

One of the significant reasons behind a decreased accuracy is that OpenPose was not much capable of capturing the activity posture precisely for all the images. There were a few images that resulted in an empty black background without any skeletal key-points since OpenPose could not detect them. This was a challenge because, a lot of body parts are hidden or overlapping in some of the postures and it becomes difficult to locate the

joints in these images. However, if there are better suitable pose estimation techniques, that can ignore the background noise and can precisely estimate the human activity, then we will be able to generate a better model with a higher accuracy.

Another challenge in this study was the availability of data to train and test the model. Not a lot of publicly available datasets exist for yoga postures. In spite of the fact that a few sources are starting to come up with related images, the size of the dataset is always small. Although data augmentation was used to enlarge the training dataset to overcome this challenge, a larger set of distinct images would certainly make a difference in model generalization and performance.

5.2 Conclusion

In this report, an image based classification system was proposed to recognize 10 different yoga postures from a set of 732 images. OpenPose was used as a pre-processing tool to extract the required pose recognition key-points that provided feature rich data for the classification process. Two different approaches for classification involving CNN models built from scratch and Transfer learning models were studied, to determine the best performing yoga recognition model. A varied set of network parameters were experimented to assist the model in learning from the given data and make accurate predictions. Techniques such as data augmentation and Dropout has certainly helped in building the model considering the small size of the data used for the learning process.

The transfer learning model using VGG16 reported the best results achieving an accuracy of 82% and showing an ability to perform well without having much distraction to the background noise in the images. However, the overall system is constrained by the quality of pose estimation using OpenPose as it fails to detect feature key-points in cases of hidden or overlapping body parts. The end-to-end deep learning based framework can be implemented against any set of new images that fall under the category of the 10 yoga postures. For any new posture that needs to be included, the model can be easily retrained by adding an additional neuron to the final network layer (SoftMax layer) of the model. By taking advantage of the ongoing developments of more flexible and dynamic deep learning based architectures, the overall system can be easily reshaped to make it more efficient and adaptable to any kind of activity recognition tasks.

5.3 Future Work

Some of the possibilities for future research on this project are enumerated below.

1. With respect to Transfer learning, a lot of studies have reported excellent results with the InceptionV3 model. However, there were a few challenges with the computational power when implementing the Inception model for this project. It would be fascinating to see if the model can be improved by experimenting it in an appropriate environment.

2. In regards to the pre-processing feature extraction phase using OpenPose, it would be interesting to study an alternative approach such as Dense Pose and PersonLab [56] and observe the impact in the overall system performance.

3. Another possible area of investigation would be to try a different set of yoga postures and see how the model can learn the different arrangement of features from the images and its prediction rates.

4. The final area of investigation would be to use Ensemble neural network models for the classification process. Variability of any form is very useful when training deep learning networks as it will enable the model to generalize and learn better. In our study, the variability was induced using the 5-Fold cross validation with the data. In Ensemble networks, the variability is induced with the behaviour of multiple network architectures on the same data.

Bibliography

[1] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7291–7299.

[2] A. Lai, B. Reddy, and B. v. Vlijmen, "Yog.ai: Deep learning for yoga," http://cs230.stanford.edu/projects_winter_2019/reports/15813480.pdf, 2019.

[3] H.-T. Chen, Y.-Z. He, C.-C. Hsu, C.-L. Chou, S.-Y. Lee, and B.-S. P. Lin, "Yoga posture recognition for self-training," in *International Conference on Multimedia Modeling*. Springer, 2014, pp. 496–505.

[4] "Yoga journal)," http://www.yogajournal.com/.

[5] S. Patil, A. Pawar, A. Peshave, A. N. Ansari, and A. Navada, "Yoga tutor visualization and analysis using surf algorithm," in *2011 IEEE Control and System Graduate Research Colloquium*. IEEE, 2011, pp. 43–46.

[6] H.-T. Chen, Y.-Z. He, and C.-C. Hsu, "Computer-assisted yoga training system," *Multimedia Tools and Applications*, vol. 77, no. 18, pp. 23 969–23 991, 2018.

[7] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[8] T. Morris, *Computer vision and image processing*. Palgrave Macmillan, 2004.

[9] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[10] K. M. Fauske, http://www.texample.net/tikz/examples/neural-network/, 2006.

[11] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert systems with applications*, vol. 59, pp. 235–244, 2016.

[12] M. Coşkun, A. Uçar, Ö. Yildirim, and Y. Demir, "Face recognition based on convolutional neural network," in *2017 International Conference on Modern Electrical and Energy Systems (MEES)*. IEEE, 2017, pp. 376–379.

[13] "Github forum," https://github.com/skvark/opencv-python/issues/171, 2019.

[14] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[15] A. Van Ooyen and B. Nienhuis, "Improving the convergence of the back-propagation algorithm," *Neural networks*, vol. 5, no. 3, pp. 465–471, 1992.

[16] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[17] B. Kieffer, M. Babaie, S. Kalra, and H. R. Tizhoosh, "Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks," in *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE, 2017, pp. 1–6.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2015.

[19] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[20] Y. Bar, I. Diamant, L. Wolf, and H. Greenspan, "Deep learning with non-medical training used for chest pathology identification," in *Medical Imaging 2015: Computer-Aided Diagnosis*, vol. 9414. International Society for Optics and Photonics, 2015, p. 94140V.

[21] E. Al Hadhrami, M. Al Mufti, B. Taha, and N. Werghi, "Transfer learning with convolutional neural networks for moving target classification with micro-doppler radar spectrograms," in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE, 2018, pp. 148–154.

[22] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[23] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[24] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[25] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[26] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 2018, pp. 117–122.

[27] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle, "Brain tumor segmentation with deep neural networks," *Medical image analysis*, vol. 35, pp. 18–31, 2017.

[28] Y. M. George, H. H. Zayed, M. I. Roushdy, and B. M. Elbagoury, "Remote computer-aided breast cancer detection and diagnosis system based on cytological images," *IEEE Systems Journal*, vol. 8, no. 3, pp. 949–964, 2013.

[29] A. B. Sargano, X. Wang, P. Angelov, and Z. Habib, "Human action recognition using transfer learning with deep representations," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 463–469.

[30] A. Tang, K. Lu, Y. Wang, J. Huang, and H. Li, "A real-time hand posture recognition system using deep neural networks," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 2, pp. 1–23, 2015.

[31] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, "Physical human activity recognition using wearable sensors," *Sensors*, vol. 15, no. 12, pp. 31 314–31 338, 2015.

[32] M. Ermes, J. Pärkkä, J. Mäntyjärvi, and I. Korhonen, "Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions," *IEEE transactions on information technology in biomedicine*, vol. 12, no. 1, pp. 20–26, 2008.

[33] Y. Liu, L. Nie, L. Liu, and D. S. Rosenblum, "From action to activity: sensor-based activity recognition," *Neurocomputing*, vol. 181, pp. 108–115, 2016.

[34] A. Sugathan *et al.*, "Attributed relational graph based feature extraction of body poses in indian classical dance bharathanatyam," *Int. J. Eng. Res. Appl*, vol. 4, no. 5, pp. 11–17, 2014.

[35] H. Jiang, Z.-N. Li, and M. S. Drew, "Recognizing posture in pictures with successive convexification and linear programming," *IEEE MultiMedia*, vol. 14, no. 2, pp. 26–37, 2007.

[36] S. P. Mohan, A. Sugathan, and S. Sekharan, "A comparison on different 2d human pose estimation method," 2018.

[37] E. W. Trejo and P. Yuan, "Recognition of yoga poses through an interactive system with kinect device," in *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*. IEEE, 2018, pp. 1–5.

[38] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.

[39] S. K. Yadav, A. Singh, A. Gupta, and J. L. Raheja, "Real-time yoga recognition using deep learning," *Neural Computing and Applications*, pp. 1–13, 2019.

[40] M. C. Thar, K. Z. N. Winn, and N. Funabiki, "A proposal of yoga pose assessment method using pose detection for self-learning," in *2019 International Conference on Advanced Information Technologies (ICAIT)*. IEEE, 2019, pp. 137–142.

[41] S. Qiao, Y. Wang, and J. Li, "Real-time human gesture grading based on openpose," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, 2017, pp. 1–6.

[42] A. Rosebrock, https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras//, 2017.

[43] S. L. Rabano, M. K. Cabatuan, E. Sybingco, E. P. Dadios, and E. J. Calilung, "Common garbage classification using mobilenet," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. IEEE, 2018, pp. 1–4.

[44] A. S. B. Reddy and D. S. Juliet, "Transfer learning with resnet-50 for malaria cell-image classification," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019, pp. 0945–0949.

[45] J. R. Rajayogi, G. Manjunath, and G. Shobha, "Indian food image classification with transfer learning," in *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, vol. 4, 2019, pp. 1–4.

[46] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE, 2016, pp. 99–104.

[47] X. Zheng, M. Wang, and J. Ordieres-Meré, "Comparison of data preprocessing approaches for applying deep learning to human activity recognition in the context of industry 4.0," *Sensors*, vol. 18, no. 7, p. 2146, 2018.

[48] K. K. Pal and K. Sudeep, "Preprocessing for image classification by convolutional neural networks," in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2016, pp. 1778–1781.

[49] T. Sugata and C. Yang, "Leaf app: Leaf recognition with deep convolutional neural networks," *IOP Conference Series: Materials Science and Engineering*, vol. 273, p. 012004, 11 2017.

[50] A. Marchenkova, "Convolutional neural networks for classifying yoga poses," https://www.amarchenkova.com/2018/12/04/data-set-convolutional-neural-network-yoga-pose/, 2018.

[51] F. SHAIKH, "Tutorial: Optimizing neural networks using keras (with image recognition case study)," https://www.analyticsvidhya.com/blog/2016/10/tutorial-optimizing-neural-networks-using-keras-with-image-recognition-case-study/, 2016.

[52] D. Kapil, "Stochastic vs batch gradient descent)," https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1, 2019.

[53] J. Brownlee, "Difference between a batch and an epoch in a neural network)," https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/, 2019.

[54] Q. Xu, G. Huang, M. Yu, and Y. Guo, "Fall prediction based on key points of human bones," *Physica A: Statistical Mechanics and its Applications*, vol. 540, p. 123205, 2020.

[55] M. A. Takalkar and M. Xu, "Image based facial micro-expression recognition using deep learning on small datasets," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2017, pp. 1–7.

[56] D. Mwiti, "Difference between a batch and an epoch in a neural network)," https://heartbeat.fritz.ai/a-2019-guide-to-human-pose-estimation-c10b79b64b73, 2019.