DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY,
SHARDA UNIVERSITY, GREATER NOIDA

Human Posture Estimation Using Different Deep Learning Techniques

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Technology in Computer Science and Engineering

by

SATNAM SINGH (2019001788)
VISWAJEET KUMAR (2019003633)
NALIN KASHYAP (2019005417)

Supervised by:
MS. KANIKA SINGLA, ASSISTANT PROFESSOR (CSE)

May, 2023

CERTIFICATE

This is to certify that the report entitled "Human Posture Estimation Using ResNet" submitted by "Satnam Singh (2019001788), Viswajeet Kumar (2019003633) and Nalin Kashyap (2019005417)" to Sharda University,

towards the fulfillment of requirements of the degree of "Bachelor of Technology" is record of bonafide final year Project work carried out by them in the "Department of Computer Science & Engineering, Sharda School of Engineering and Technology, Sharda University".

The results/findings contained in this Project have not been submitted in part or full to any other University/Institute forward of any other Degree/Diploma.

Signature of Supervisor

Name: Ms. Kanika Singla

Designation: Assistant Professor

Signature of Head of Department

Name: Prof. (Dr.) Nitin Rakesh

(Office seal)

Place:

Date:

Signature of External Examiner

Date:

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. We consider our self very lucky and honored to have so many wonderful people lead us through in completion of this project.

First and foremost, we would like to thank Dr. Nitin Rakesh, HOD, CSE who gave us an opportunity to undertake this project.

My grateful thanks to Ms. Kanika Singla for her guidance in my project work. Ms. Kanika Singla, who in spite of being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where we would have been without his help.

CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Name and signature of Students:

SATNAM SINGH (2019001788)

VISWAJEET KUMAR (2019003633)

NALIN KASHYAP (2019005417)

ABSTRACT

This research paper presents a method for determining human posture by utilizing multiple neural network techniques. Various architectures, like CNNs and ResNet. The performance of each network is evaluated using a dataset of human posture images the results are compared. The findings of this study indicate that using multiple neural network techniques leads to increased accuracy in human posture estimation, and it could have potential applications in areas such as human-computer interaction, rehabilitation, and health monitoring.

LIST OF FIGURES

Fig 1.	Dataset	25
Fig 2.	Architecture of CNN	27
Fig 3.	Residual Learning Block	30
Fig 4.	Basic ResNet Architecture	30
Fig 5(a).	CNN implementation with 16 Neurons with 4 Hidden Layers (Loss vs Epoch)	35
Fig 5(b).	CNN implementation with 16 Neurons with 4 Hidden Layers (Accuracy vs Epoch)	35
Fig 6(a).	CNN implementation with 32 Neurons with 4 Hidden Layers (Loss vs Epoch)	36
Fig 6(b).	CNN implementation with 32 Neurons with 4 Hidden Layers (Accuracy vs Epoch)	36
Fig 7(a).	CNN implementation with 64 Neurons with 4 Hidden Layers (Loss vs Epoch)	37
Fig 7(b).	CNN implementation with 64 Neurons with 4 Hidden Layers (Accuracy vs Epoch)	37
Fig 8(a).	CNN implementation with 128 Neurons with 4 Hidden Layers (Loss vs Epoch)	38
Fig 8(b).	CNN implementation with 128 Neurons with 4 Hidden Layers (Accuracy vs Epoch)	38
Fig 9(a).	16 in First, 32 in second, 64 in third and 256 neurons in fourth Hidden Layer (Loss vs Epoch)	39
Fig 9(b).	16 in First, 32 in second, 64 in third and 256 neurons in fourth Hidden Layer (Accuracy vs Epoch)	39
Fig 10(a).	ResNet implementation with 20 Layers (Accuracy vs Epoch)	40
Fig 10(b).	ResNet implementation with 20 Layers (Loss vs Epoch)	40
Fig 11(a).	ResNet 18 implementation (Accuracy vs Epoch)	41
Fig 11(b).	ResNet 18 implementation (Loss vs Epoch)	42
Fig 12(a).	ResNet 34 implementation (Accuracy vs Epoch)	43
Fig 12(b).	ResNet 34 implementation (Loss vs Epoch)	43

LIST OF TABLES

Table 1	Literature Survey	6-22
Table 2	CNN Comparison summary Table	40

CONTENTS

TITLE .... **i**
CERTIFICATE .... **ii**
ACKNOWLEDGEMENT .... **iii**
ABSTRACT .... **iv**
LISTOF FIGURES .... **v**
LISTOF TABLES .... **vi**
CHAPTER1: INTRODUCTION **1**
1.1 Problem Statement 1
1.2 Project Overview 1
1.3 Expected Outcome 3
1.4 Hardware & Software Specifications 3
1.5 Other Non-Functional Requirements 3
1.6 Report Outline 3
CHAPTER2: LITERATURESURVEY **4**
2.1 Existing Work 4
2.2 Literature Survey 6
2.3 Proposed Work 22
2.4 Feasibility Study 23
CHAPTER3: SYSTEM DESIGN& ANALYSIS **24**
3.1 Project Perspective 24
3.2 Performance Requirements 24
3.3 System Features 24
3.4 Methodology 24
CHAPTER4: RESULTS ANALYSIS **34**
4.1 CNN implementation with 16 Neurons with 4 Hidden Layers 35
4.2 CNN implementation with 32 Neurons with 4 Hidden Layers 36
4.3 CNN implementation with 64 Neurons with 4 Hidden Layers 37
4.4 CNN implementation with 128 Neurons with 4 Hidden Layers 38
4.5 16 in First, 32 in second, 64 in third and 256 neurons in fourth Hidden Layer 39
4.6 ResNet implementation with 20 Layers 40
4.7 ResNet 18 implementation 41
4.8 ResNet 34 implementation 43
CHAPTER5: CONCLUSION **45**
5.1 Conclusion 45
5.2 Future Scope 45
CHAPTER6: REFERENCES **47**
ANNEXURE 1 .... **49**
CODE SNIPPETS **50**

CHAPTER 1

INTRODUCTION

Human posture estimation is a crucial task in various fields such as human-computer interaction, rehabilitation, and health monitoring. Accurate posture estimation can provide valuable information about an individual's movements and activities, which can be used for various applications such as motion tracking, gesture recognition, and fall detection. However, estimating posture from 2D images is a challenging task due to variations in lighting conditions, background, and viewpoint. Recognition of human posture is a difficult task.

1.1 Problem Statement

Image recognition of human posture is a difficult task. The field of posture recognition receives a lot of attention in the field of human sensing **[1],** in the area as remote control of vision impaired, needs advanced computer vision techniques able to localise people in their environment, recognise their posture and behavior **[2].**

- To classify people and detect moving items.
- Surveillance, indoor and outdoor monitoring where a person with suspicious postures tries to do malicious activity.
- In healthcare and other fields such as in rehabilitation training of people with posture defects.
- Accidental recovery in the scenarios where a person had an accident near a road, and he/she is laying on the road this can cause an emergency response to the nearest hospital.

1.2 Project Overview

Human posture estimation is a crucial task in various fields such as human-computer interaction, rehabilitation, and health monitoring. Accurate posture estimation can provide valuable information about an individual's movements and activities, which can be used for various applications such as motion tracking, gesture recognition, and fall detection **[1].** A difficult job is posture estimation from 2D photos owing to differences in illumination, backdrop, and viewpoint. Traditionally, human posture estimation has been tackled using various methods such as template matching, edge detection, and model-based approaches. These methods have limitations, such as being sensitive to changes in lighting conditions and background and are not robust enough to handle variations in body shape and posture. Deep learning methods have significantly improved posture estimation in recent years **[2].** Convolutional neural networks (CNNs), in particular, have demonstrated tremendous promise in this area. Due to its capacity to learn hierarchical representations of pictures, CNNs are especially well suited for image processing applications **[3].** They are able to pick out certain features in pictures, such as edges, corners, and textures, and use them to classify images. ResNet is one of the most effective CNN designs. ResNet, an abbreviation for Residual Network, takes advantage of residual connections to enables the network to learn more intricate operations. In traditional CNNs, the deeper layers of the neural network can struggle to learn useful features due to the vanishing gradients problem. By enabling the gradients to propagate through the layers without any intermediate nodes, residual connections enable the network to get around this issue **[4].** This enables ResNet to achieve exceptional performance on image classification tasks.

In this paper, a multi-modal approach for estimating human posture from 2D images using neural networks is proposed. Our approach explores the use of several neural network architectures, including CNNs, recurrent neural networks (RNNs) and long short-term memory (LSTM) networks **[5].** This work evaluates the performance of each network on a dataset of human posture images and compare the results. Our experiments demonstrate the effectiveness of using a combination of neural network techniques for human posture estimation from 2D images, achieving higher accuracy than using a single network alone. This research will discuss the importance of human posture estimation and how it's been tackled in the past. This work will also delve into the details of CNNs and ResNet and how they can be used for posture estimation. This work will present our proposed method for posture estimation using multiple neural network architectures, and evaluate its performance using a dataset of human posture images. This work will also compare our results with other existing methods and discuss the potential applications of our proposed approach.

In summary, this research aims to improve the accuracy of human posture estimation from 2D images by using a combination of neural network techniques, specifically multiple architectures like CNN, RNN and LSTM. Through studies on a collection of photographs of human posture, this work will show the efficacy of our suggested strategy and contrast the outcomes with those of other approaches already in use **[6].**

1.3 Expected Outcome

This work aims to implement Comparative Study for Human Posture Estimation using ResNet and CNN with varying neurons and layers to find which pair gives the best result. also visualizing each of the pair (tweaking of layers with varying neurons in Convulsions Neural Network) with help of graphs. At the end of comparison ResNet will perform highest accuracy.

1.4 Hardware and Software Specifications

- CPU: Intel® Core i5 @1.60 Ghz
- GPU: 8GB
- RAM: 16GB
- Display: Standard monitor
- OS: Windows Operating System
- Language: Python
- IDE: Open-source programming tools like Jupyter Notebook, Google Colab.

1.5 Other Non-Functional Requirements

- Collage Lab Computer
- IEEE Posture Dataset

1.6 Report Outline

In Chapter 1 the problem statement is presented, the project overview, expected outcome and hardware and software requirements. Related works, a literature review, a proposed system, and a feasibility study make up Chapter 2. Chapter 3 is mainly about the methodology used along the project. Chapter 4 will be about the result of the differs algorithms used toward the project and the output of the project implementation will also be detailed. Chapter 5 is about the conclusion and what is the future prospective.

CHAPTER 2

LITERATURE SURVEY

2.1Existing Work

In this study, the author presented YOLOv5-HR-TCM, a quick, end-to-end model for predicting 3D human posture. We offer a model that takes into account each estimate stage, such as person recognition, 3D human pose estimation and 2D human pose estimation. It is based on the lifting approach from 2D to 3D for determining 3D human positions **[7].** Perceptual multimode inputs are extracted from RGB images and combined by a multimode stream system suitable for different input modalities. Construction of 3D skeletal joint LSTM (long-term short-term memory). 3D ResNet (RGB) 3D ResNet 92.1 (color separation of body parts) 94.6 LSTM (3D skeleton linking) 95.4 The combined result is 96 **[8].** In order to recognize and differentiate between human right and left hands, this research presents a parallel network based on body position estimation and hand detection. Using hand movements, the network is used for human-robot interaction (HRI). The results from the two channels are then combined. Using knowledge of human anatomy, the fusion module may alter hand recognition findings and distinguish between the left and right hands. This HRI system can be used with our technique **[9].** In another study, the author created a non-contact video-based method to track body movements and postures automatically while a person is sleeping. The effectiveness of seven renowned pre-trained networks, including Google Net, ResNet50, ResNet-101, ResNet-152, AlexNet, VGG-16, VGG-19 has been studied. ResNet152 achieved the greatest accuracy of 95.1%, outperforming all other pre-trained networks, including a 4-layer CNN network. **[10].** Using dynamic, instance-aware convolutions, FCPose is a fully convolutional approach for estimating the posture of many people. Mask R-CNN (ResNet-101) performs better (64.8% APkp vs. 64.3% APkp) and is faster (9.26 FPS vs 41.67 FPS) Additionally, compared to other state-of-the-art techniques, FCPose offers a better speed/accuracy trade-off. brand-new CNN network. The results of our studies demonstrate that FCPose is a simple yet effective approach for estimating multi-person poses **[11].** To improve decision-making in posture recognition, this study used two transfer learning algorithms and two deep learning algorithms. MLP.y. and CNN. With a validation accuracy of 91.2%, AlextNet + HPO outscored the other four models, while VGG16 +

HPO came in second with 90.2% **[12].** Utilizing real-world photos and integrating object detection, segmentation, and classification techniques, we suggest a hybrid model for identifying human position. Inception-ResNet-v2 with a revision is then utilized for posture recognition after Mask R-CNN is used to detect and segment subjects. **[13].** This paper suggested a unique hybrid strategy combining machine learning classifiers using deep learning classifiers (Naive Bayes, Decision tree, KNN, SVM, KNN, random forest, linear discrete analysis and quadratic discretization) (LSTM, two-way LSTM), (lattices) 1D, 2D and 3D convolutional neurons, LSTM and KNN). The suggested strategy takes use of deep learning (DL) and machine learning (ML) prediction to improve the effectiveness of current approaches. Results from experiments revealed an accuracy of more than 98 percent **[14].** The main problem of pose estimation is to visualize certain heat maps created by the ground truth, the ResNet-50 baseline model, and the ResNet-50 Plus L-PGCN model, in addition to capturing ordered links between significant human body locations **[15].** DeneSVM achieves a test accuracy of 95 percent and a score of 94.72 percent for the 30th epoch. followed by DeneNet-121 with an accuracy of 92%, ResNet-50V2 with an accuracy of 93%, and DenseNet-121 with an accuracy of 93% **[16].** Given that the detection implementations in both models (see appendix) are identical, the gains can only be attributed to enhanced networks. The most striking outcome is an increase of 6.0 percent in COCO's standard measure, which is a 28 percent relative improvement over the challenging COCO dataset. **[17].** The dual learning tasks 3D to 2D pose projection and 2D to 3D pose transformation were carried out using CNN’s, spatiotemporal modelling, self-guided learning, and geometric deep learning. It is noteworthy that the model improves performance by over 12% when compared to the conference version (63.67 mm against 73.10 mm). **[18].** Dense spatial regression using a mixture model, Spatial regression models, often with a linear additive specification, in which the correlation between areal units is exogenously given using a weight matrix that replicates the spatial structure and the spatial interaction pattern, results concludes that a higher speed-accuracy trade-off is provided by a mixture model and can increase the purpose for multiscale evaluation. Also, it leads to faster convergence **[19].** Hybrid fuzzy logic and ML approach used for the classification of human postures while resting in bed with the greatest possible data efficiency. The proposed method produced data-efficient posture categorization with a 97.1 percent accuracy rate **[20].** System based on a 3D-RCNN, Regions Proposal Networks (RPNs) and container proposals network (CPN) utilized to recognize and categorize the many classes of a person's posture, including sit, lie, and stand, for one-person posture identification, CNN and CPN shared, with reduction gave 78.2% accuracy **[21].**

2.2 Literature Survey

The Literature Survey is shown with the help of table (Table 1.)

S. No	**Name of Paper**	**Year**	**Authors**	**Objectives**	**Algorithm**	**Outcome**
1.	Human Posture Detection Using Image Augmentation and Hyperparameter-Optimized Transfer Learning Algorithms **[12]**	2022	Roseline Oluwaseun, Ogundokun, Rytis Maskeliunas, Robertas Damasevicius	A fresh decision-support system for the hyperparameter optimization of the multilayer perceptron (MLP), CNN, VGG16, and AlexNet models to achieve the best classification outcomes	Multilayer perceptron, VGG16, CNN, and Alex Net (MLP)	Two deep learning algorithms and two transfer learning algorithms were applied to CNN and MLP.y, two of the four models used in this work for decision assistance in posture detection. With a validation accuracy of 91.2 percent, AlextNet + HPO surpassed the other four models, while VGG16 +

						HPO came in second with 90.2 percent.
2.	Knowledge-Guided Deep Fractal Neural Networks for Human Pose Estimation **[13]**	2018	Guanghan Ning, Zhi Zhang, Zhiquan He	Important components of interconnected human postures. We suggest investigating the optimal representation and input of external information into deep neural networks in order to regulate the training process using learnt projections that impose the suitable prior. Without using any explicit graphical modelling, we build a fractal network specifically to regress photos of	Inception-ResNet Modules, Fractal Networks, Knowledge-Guided Learning.	In this study, we put forward the idea of encoding and injecting external human knowledge into deep neural networks to direct their training using learnt projections for more accurate human posture prediction.

				human posture into heatmaps using the stacked hourglass architecture and inception-ResNet module.		
3.	Human Pose Estimation via Improved ResNet 50 **[14]**	2017	Xiao Xiao, Wanggen Wan	It will offer a technique. The prediction of 2D human posture estimation is formulated as a regression issue towards body joints using top-down methodologies.	ResNet-50	We Present our knowledge, the first application of ResNet-50 to human posture estimation. On several difficult academic datasets, we are able to produce results that are state-of-the-art or better as a result.

4.	Structure-aware human pose estimation with graph convolutional network **[15]**	2020	Yanrui Bina , Zhao-Min Chenb , Xiu-Shen Wei c, Xinya Chena , Changxin Gaoa , Nong Sang	In accordance with the natural compositional model of a human body, the model constructs a directed network between body important points. A 3-D tensor made up of several feature maps serves as the representation for each node (key point).	CNN, ResNet	Pose estimation's main problem is capturing the structured relationships between significant human body regions. Additionally, it shows a number of heat maps that were produced using the ResNet-50 baseline model, ResNet-50 + L-PGCN model, and actual data.
5.	A Novel Deep Transfer Learning Approach Based on Depth-Wise Separable CNN for Human	2022	Roseline Oluwaseun Ogundokun , Rytis Maskeliunas , Sanjay Misra, Robertas	The model usually improves accuracy noticeably as the number of epochs increases without experiencing any	DenseNet-121, ResNet-50V2, DeneSVM	ResNet-50V2 obtains an accuracy of 93 percent, followed by DeneSVM with a test accuracy score of 94.72 percent for the 30th epoch and 95

	Posture Detection **[16]**		Damasevicius	performance problems or overfitting. The technique is effective in classification and detection exhibitions as well, with just a small number of parameters and reasonable computational costs needed.		percent. DenseNet-121 earns an accuracy of 92 percent. DenseNet-121 has a 92 percent accuracy rate.
6.	Deep Residual Learning for Image Recognition **[17]**	2015	Jin Sun, Xiangyu Zhang, Shaoqing Ten, and Kaimaing He	The Objective of this paper is Object Detection on PASCAL and MS COCO with different algorithms.	Residual Networks, ResNet-50	Given that the detection implementations in both models are identical, the improvements can only be attributed to enhanced networks. The most remarkable result is a 6.0% rise in COCO's standard measure, a

						28% relative improvement over the challenging COCO dataset.
7.	Transfer Learning for Clinical Sleep Pose Detection Using a Single 2D IR Camera **[10]**	2021	Shirin Enshaeifar, Adrian Hilton, and Sara Mahvash Mohammadi	This article describes a technique for motion and posture-based, non-contact sleep monitoring. Using supervised machine learning methods and a transfer learning approach, four predefined postures and the empty bed state during 8–10-hour overnight sleep episodes were successfully measured. The method was evaluated in	CNN, Transfer Learning, PSG	A non-contact video-based system was created in this study to automatically track body positions and movement as you sleep. To enhance the learning effectiveness of pre-trained deep networks, transfer learning was introduced. ResNet152 outperformed all other pre-trained networks and a 4-layer de novo CNN network in terms of accuracy, coming in at 95.1%. Future research

				contrast to postures that were computed using clinical polysomnography measurement equipment and poses that were manually scored during sleep.		directions include examining the use of this strategy across the lifespan and creating an intelligent non-contact monitoring system for the home environment.
8.	Pose ResNet: 3DHuman Pose Estimation Based on Self-Supervision **[22]**	2023	Dong Liang, Wenxia Bao, Zhongyu Ma, Xianjun Yang, and Tao Niu	A self-supervised 3D posture estimation model dubbed posture ResNet can extract features from 2D photos without the need for 3D ground truth labelling. It uses a deconvolution network, synthetic occlusion, transfer learning,	ResNet50, CNN	This study suggests a technique that combines synthetic occlusion, transfer learning, ResNet50, CBAM, and WASP to create 3D labels utilizing epipolar geometry for self-supervised learning. On the Human3.6M dataset, the final MPJPE

				convolutional block attention, waterfall arouse spatial pooling, convolutional block attention, and a self-supervised training method. The findings reveal that the mean per joint position error (MPJPE) is 74.6 mm even without the usage of 3D ground truth labels.		was shrunk to 74.6 mm.
9.	FCPose: Fully Convolutional Multi-Person Pose Estimation with Dynamic Instance-Aware	2021	Weian Mao , Zhi Tian , Xinlong Wang1 , Chunhua Shen, The University of Adelaide , Australia	FCPose is a dynamic, instance-aware, fully convolutional framework for multi-person pose estimation. Regardless of the amount of	R-CNN, ResNet-101	FCPose is a unique key point detection framework that instructs the model to focus on instances by using dynamic key point heads rather than

	Convolutions **[28]**			people in the image, it does away with ROIs and grouping post-processing and has essentially consistent inference times. It is easy to use but effective, according to experiment findings.		ROIs. Numerous tests show that it provides a straightforward, quick, and efficient architecture, and on the COCO dataset, it can run at 42 frames per second with 64.8% APkp on a single 1080Ti GPU.
10.	Token Pose: Learning Keypoint tokens for Human Pose Estimation **[29]**	2021	Yanjie Li, Zhicheng Wang, Shoukui Zhang, Sen Yang, and Wankou Yang	In this study, a unique method called TokenPose for estimating human stance is proposed. To understand constraint connections and appearance cues from images, it embeds each key	CNN, COCO	TokenPose is a brand-new token-based presentation for estimating human poses that create visual tokens from the image's patches and embeds key point entities into token embeddings. By interacting with oneself, it may capture constraint and appearance

				point as a token. Numerous tests reveal that although being more lightweight, the small and big TokenPose models are equally as effective as their cutting-edge CNN-based equivalents. There is open access to the code.		cues, and hybrid designs outperform CNN-based techniques in terms of performance.
11.	Dual-Hand Detection for Human–Robot Interaction by a Parallel Network Based on Hand Detection and Body Pose Estimation **[9]**	2019	Jinguo Liu, Zhaojie Ju, and Wing Gao	In order to recognise and differentiate between a human's right and left hands, the author of this paper introduces a parallel network based on hand detection and body position estimation.	ResNet, Deep Neural Network	The left and right hands of astronauts could be recognized and located with accuracy using a parallel deep neural network that contained hand and human body features. Its objectives were to separate the structural

				To enhance hand identification and distinguish between the right and left hands, it is necessary to make use of knowledge about hand features and human body structure.		characteristics of the hands and the body, locate the left and right hands, and combine the outputs of the two subnetworks. According to research, the RI-SSD network may increase the accuracy of hand recognition and guarantee real-time performance.
12.	P-CNN: Pose-based CNN Features for Action Recognition **[4]**		Ivan Laptev, Guilhem Cheron, and Cordelia Schmid	We suggest the Pose-based Convolutional NeuralNetwork descriptor (P-CNN), which collect's motion and appearances data along tracks of human body components , for action detection.	P-CNN, ResNet	Compared to other posture-based features like HLPF, pose-based convolutional neural network features (P-CNN) are more resistant to mistakes in human pose estimation. They complement dense trajectory

						characteristics and perform better than HLPF in recognizing fine-grained actions. The recent advances in pose estimation indicate a bright future for stance-based action detection techniques.
13.	Adapting Mobile Nets for mobile based upper body pose estimation **[17]**	2018	Bappaditya Debnat, Mary O'Brien, Motonori ,Yamaguchi	With a new split stream design and the transfer of learned features from MobileNets that have already been pre-trained on ImageNet, a lightweight and effective CNN architecture for mobile and embedded vision applications	CNN, MobileNets	In order to lessen over-fitting, MobileNets are modified for heatmap regression, and a unique split architecture is created. The updated MobileNets outperform the baseline across PCK thresholds and achieve performances that are nearly state-of-the-art. The development

				, MobileNets is adaptable to human posture estimation.		of embedded and mobile vision applications will benefit from this.
14.	3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. **[23]**	2016	Antoni B. Chan and Sijin Li	This paper suggests using a deep convolutional neural network to infer 3D human posture from monocular pictures. It is learned using two different approaches: a pre-training strategy that uses a network trained for body part recognition to initialize the posture regressor, and a multi-task framework that simultaneously trains pose	SVM, CNN,	In order to predict a 3D person position from monocular pictures, this study combined a deep convolutional neural network with two distinct approaches—a multi-task framework and regression task pre-training with detection tasks. Results over baseline methods indicated a significant improvement. The capacity of the network to produce structural dependencies will be

				regression and body part detectors. It performs noticeably better than the conventional methods.		investigated in further studies.
15.	Deep Learning-Based Human Posture Recogniti on **[6]**	2022	Ayre-Storie, Zhang, L.	A hybrid model is proposed that combines object detection, segmentation, and classification algorithms to recognise three human postures: leaping, sitting, and standing. and standing—from real-world photos in order to create efficient spatial-temporal representations.	Inception-ResNet-v2 and Mask R-CNN	The two-stage method generated excellent results, with mAP scores of 86% for leaping, 95% for sitting, and 94% for standing for the acquired real-world data set.

16.	A Hybrid Posture Detection Framework: Integrating Machine Learning and Deep Neural Networks **[27]**	2021	The authors are Liaqat, S., Dashtipour, K., Arshad, K., Assaleh, & Ramzan.	On the basis of DL approaches, a novel hybrid strategy is created to identify posture prediction. The hybrid approach trains the meta-learning with a variety of predictions using deep learning and machine learning. The results of the experiments show that the proposed hybrid approach performs better than both DL and ML algorithms.	KNN, LSTM, CNN, SVM	The experimental results on a widely used benchmark dataset are displayed; the accuracy of the results was over 98 percent.
17.	3D Human	2020	Liang Lin,	The 2D to 3D pose	3D human pose	External 2D human pose

	Pose Machines with Self-Supervised Learning **[18]**		Pengxu Wei, Chenhan Jiang, Keze Wang, Chen Qian, and	transformation and 3D to 2D pose projection are two dual learning tasks that are included in the suggested technique.	machine,CNNs ,Geometric deep learning.	data may be used without the requirement for additional 3D annotations thanks to the suggested self-supervised correction technique, which may close the gap between 3D and 2D human poses. The effectiveness and superiority of our suggested technique have been thoroughly tested on two 3D human pose datasets that are available to the public.
18.	Human Posture Recognition Using a Hybrid of Fuzzy Logic and Machine	2020	Ji, H., Liu, X., Ma, O., & Ren, W.	In order to characterise human postures during sleeping and maximise data	SVM and fuzzy logic hybrid	The proposed method produced data-efficient posture categorization with a 97.1

	Learning Approaches **[20]**			economy, the study uses a combined fuzzy logic and machine learning technique.		percent accuracy rate.

2.3 Proposed Work

The system proposed in this study is mainly composed of three essential parts that are data acquisition, data preprocessing and the last being model definition.

• Dataset gathered from http://shorturl.at/estE0

• Applied multiple CNN and ResNet Architectures.

• A study of the effects of number of layers and number of neurons on accuracy.

2.4 Feasibility Study

It is challenging to identify human posture from an image. In the domain of remote control for vision-impaired persons, powerful computer vision algorithms are required that can localize people in their surroundings and recognize their posture and behavior. This subject of posture recognition is receiving a lot of interest.

Human posture estimation presents challenges, including:

- Data collection: Collecting a representative dataset of human postures can be challenging, as it may require a diverse population and a significant amount of time and resources.
- Labeling: Labeling the data with the correct posture can be time-consuming and subjective, which can introduce errors and affect the accuracy of the algorithm.
- Algorithm development: Developing an accurate and efficient algorithm for posture estimation can be complex, and it may require expertise in computer vision, machine learning, and signal processing.

- Computational resources: Implementing and testing the posture estimation algorithm may require significant computational resources, such as high-performance computing clusters, GPUs, or specialized hardware.
- Variability in posture: Human postures can vary significantly, making it difficult to develop a posture estimation algorithm that can detect and classify all possible variations. The algorithm should be robust enough to handle different body shapes, clothing, and movements.
- Real-world limitations: The feasibility of posture estimation in real-world scenarios may be limited by factors such as environmental noise, occlusions, and lighting conditions, which may affect the accuracy and robustness of the algorithm.

To address these problems and challenges, posture estimation studies should carefully consider the research questions, methods, and limitations and involve multidisciplinary teams with expertise in computer science, healthcare, and ethics. The algorithm should be developed and evaluated using appropriate performance metrics and validated using independent datasets. Finally, appropriate measures should be taken to protect individuals' privacy rights and ensure that the data collected is secure and anonymized.

CHAPTER 3

SYSTEM DESIGN AND ANALYSIS

3.1 Project Perspective

In this project focuses to identify human posture estimation of four postures which are Sitting, Standing, Bending and Lying with the help of different machine learning (ML) & deep learning (DL) architectures.

3.2 Performance Requirements

The system proposed in this study is mainly composed of three essential parts that are data acquisition, data preprocessing and the last being model definition.

- A minimum 8 GB of GPU is recommended for optimal performance.
- A minimum 16 GB of RAM.
- Windows/ Linux Operating System.
- A CPU with Advanced Vector Extensions (Intel/ AMD 64-bit).
- IDE required Google Colab/ Jupyter Notebook
- Programming Language Python.
- All dependencies like Tensorflow, numpy, pandas, cv2 etc.

3.3 System Features

- Estimating Human Postures.
- Varying ResNet models to achieve the state of art.

3.4 Methodology

Deep learning algorithms, such as deep CNN, have demonstrated outstanding performances in classification methods in computer vision-based algorithms. CNN provides data-driven learning, hierarchical images, substantially representative features directly from massive amounts of data, but also automatically produces medium/high level constructions derived from raw photos. ResNet outperformed classic ML algorithms, particularly when it came to object categorization. ResNet can provide higher accuracy and reduced error rates by enabling deeper designs, better gradient flow, and faster convergence. Its skip connections allow the flow of gradients to skip over some network layers, allowing for the construction of considerably deeper structures. For some applications that demand great accuracy and the capacity to handle complex characteristics and interactions between them, ResNet is a superior option because of these advantages.

3.4.1 Dataset

The data set contains 4800 images which are categorized in four postures which are Sitting, Standing, Bending and Lying. For every posture mentioned contains 1200 images. The images have a resolution of 512 * 512 pixels. (Fig.1)

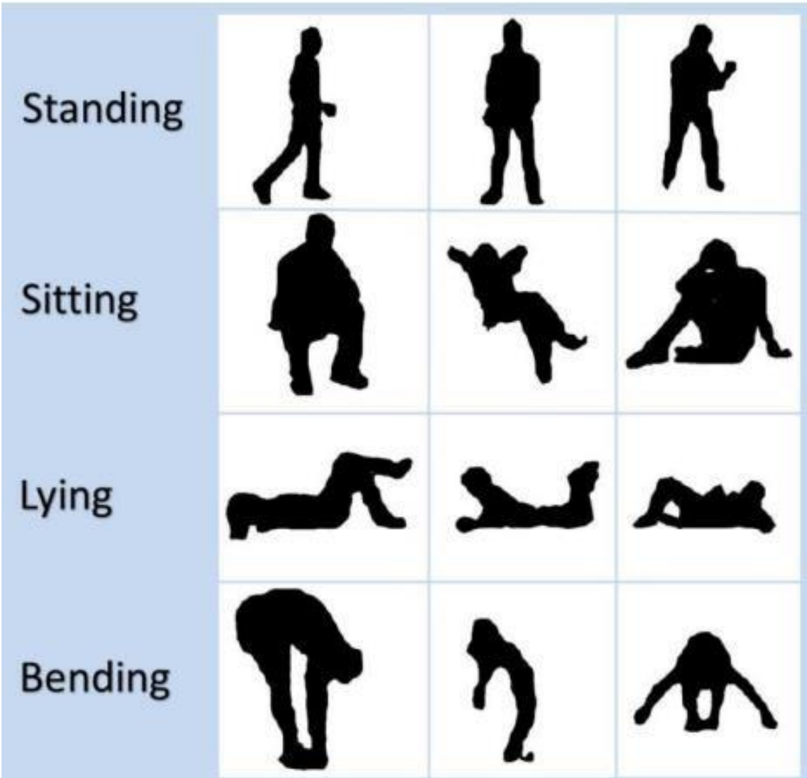


Fig.1. Dataset

3.4.2 Algorithms

- **Convolutional Neural Network (CNN)**

Like any other model of the kind of neural network, CNNs are composed of neurons that are connected in layers and may thus learn hierarchical representations. Weights and biases are used to link neurons between layers. The input layer is the first layer. In between there are hidden layers that rearrange the feature space of input given do that it can fit the output.

They have three main types of layers which are:

1.Convolutional layer

2.Pooling layer

3.Fully-connected (FC) layer

1. Convolutional Layer

- The convolutional layer, which houses the majority of the computation, is the essential part of a CNN. Among other things, it requires input data, a filter, and a feature map.
- We have a kernel or filter that moves through the receptive fields of the image to detect features, also known as feature detectors. This technique is described as convolution.

2.Pooling Layer

Reduces the dimensionality, which reduces the number of input parameters. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but this filter lacks weights. Instead, the kernel uses an aggregation function to fill the output array with values from the receptive field.

3. Fully Connected Layer

The pixel values of the input image are not directly linked to the output layer when two layers are partially linked. On the other hand, in the completely linked layer, every node in the output layer is directly connected to a node in the layer above it.

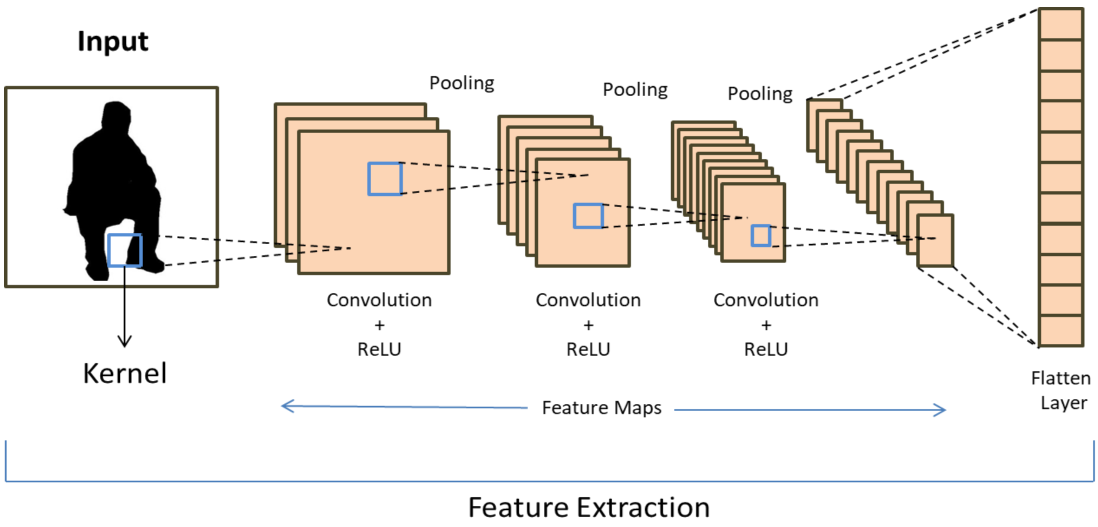


Fig 2. The architecture of a convolutional neural network

To exploit patterns, CNNs incorporates one convolutional layer at least as a hidden layer (in the context of this review predominantly spatial patterns). Other non-convolutional layers may also be included. Convolutional layers are made up of several optimizable filters (Fig.2) that change the input or previous hidden layers, which we pass through our image and transform it based on the filter settings. The formula below is used to calculate subsequent feature map values, where I stand for the input image and k for our kernel. b and d stand for the row and column indices of the result matrix, respectively.

$$G[b,d] = (i * k)[b,d] = \sum_{p=0}^{p} \sum_{q=0}^{q} k[p,q] f[b-p, d-q] \qquad \textbf{Eq. (1)}$$

3. Forward propagation

It is split into two halves. The initial step is to compute the intermediate value I, which we get by convolution of the previous layer's input data with W tensor (including filters) then bias b is added. The next step is to apply a nonlinear activation function to the intermediate value produced (g denoted activation). In addition, the picture below shows a little representation explaining the size of the tensors utilized in the equation.

$$I^{[i]} = W^{[i]}.A^{[i-1]} + b^{[i]} \qquad A^{[i]} = g^{[i]}\left(I^{[i]}\right) \qquad \textbf{Eq. (2)}$$

3. Convolutional Layer Backpropagation

As with densely linked neural networks, we generate derivatives and then utilise them to change the values of our parameters using gradient descent. We want to examine how changing the settings influences the producing features map and, as a result, the result.

$$\boldsymbol{dO}^{[i]} = \frac{\partial L}{\partial \boldsymbol{O}^{[i]}}\ \boldsymbol{dI}^{[i]} = \frac{\partial L}{\partial \boldsymbol{I}^{[i]}}\ \boldsymbol{dW}^{[i]} = \frac{\partial L}{\partial \boldsymbol{W}^{[i]}}\ \boldsymbol{db}^{[i]} = \frac{\partial L}{\partial \boldsymbol{b}^{[i]}} \qquad \textbf{Eq. (3)}$$

Computing dW[i] and db[i] - derivatives related with current layer parameters - also the value of dO[i -1] which is passed on to the previous layer, the input dO[i] is given. Of course, tensors W and dW, b and db, and O and dO all have same dimensions. 1st step is to calculate the intermediate value of dI[i] by taking the derivative of activation function and applying it on the input tensor. Outcome of this operation will be used later, according to the chain rule.

$$\boldsymbol{dI}^{[i]} = \boldsymbol{dO}^{[i]} * g'\left(\boldsymbol{Z}^{[i]}\right) \qquad \textbf{Eq. (4)}$$

To handle the backward propagation of convolution a matrix operation known as complete convolution, which is seen below is used. This procedure is defined by the formula defined, where W represents the filter and dI[b,d] represents scalar belonging to a partial derivative produced from the preceding layer.

$$\boldsymbol{dO} \mathrel{+}= \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} \ \boldsymbol{W}.\boldsymbol{dI}[b,d] \qquad \textbf{Eq. (5)}$$

Based on the features that were retrieved from the preceding layers and their various filters, this layer conducts the classification operation. FC layers often utilize a softmax activation function to categorize inputs appropriately, producing a probability ranging from 0 to 1. Convolutional and pooling layers typically use ReLu functions.

This layer performs the classification process using the features that were received from the preceding levels and their associated filters. To properly classify inputs, FC layers frequently use a softmax activation function, which generates a probability ranging from 0 to 1. ReLu functions are commonly employed in convolutional and pooling layers.

- **Residual Networks (ResNet)**

ResNet, which is short for "Residual Network," is a deep neural network architecture that was developed in 2015 by researchers at Microsoft. ResNet is designed to overcome the degradation problem that occurs when training deep neural networks. This problem occurs because as the network gets deeper, it becomes increasingly difficult to train due to the vanishing gradient problem. ResNet solves this problem by employing residual blocks, which enable the network to learn residual functions, or the difference between each block's input and output. ResNet solves this problem by employing residual blocks, which enable the network to learn residual functions, or the difference between each block's input and output. This enables the network to learn more efficiently and with fewer parameters. Among the many computer vision tasks ResNet has mastered are image classification, object identification, and semantic segmentation **[22].**

1. ResNet Working

Convolution and aggregation process layers are included in the ResNet neural network design. The network is trained using back propagation with stochastic gradient descent (SGD) optimization. ResNet's fundamental idea is the usage of residual connections, which enables the network to learn the remaining functions. The network can skip one or more tiers thanks to these remaining links, allowing the input signal to pass directly to the output of the residual block (Fig.2). This approach makes it easier to train very deep neural networks.

Let's take a closer look at how the residual connections work mathematically. Suppose we have a traditional convolutional neural network with L layers, denoted as H(x), where x is the input to the network. The output of the Lth layer is denoted as: -

$$H(x) = f(x, W) \qquad \textbf{Eq. (6)}$$

Where W represents the weights of the network. Now, let's add a residual connection between the (L-1)th and Lth layers. The output of the (L-1)th layer is denoted as: -

$$F(x) = H(x) - x \qquad \textbf{Eq. (7)}$$

We can then write the output of the Lth layer as: -

$$H(x) = f(F(x), W) + x \qquad \textbf{Eq. (8)}$$

This means that the output of the Lth layer is the sum of the residual function F(x) and the input x. By adding these residual connections, we are able to propagate the original input signal through the network and ensure that the network can learn residual functions.

ResNet employs bottleneck blocks in practice, which are composed of a 1x1 convolutional layer, a 3x3 convolutional layer, and a final 1x1 convolutional layer. The final output is created by combining the output from the last convolutional layer with the residual connection.

The formula for the output of a ResNet bottleneck block can be written as follows:

$$H(x) = f(F(x), \{W\_1, W\_2, W\_3\}), W\_4) + x \qquad \textbf{Eq. (9)}$$

Where F(x, {W_1, W_2, W_3}) represents the residual function, and W_1, W_2, W_3, and W_4 represent the weights of the four convolutional layers in the bottleneck block.

In summary, ResNet uses residual connections to bypass one or more layers (Fig.3), allowing the input signal to pass directly to the output of the residual block. By doing so, ResNet makes it possible to train extremely deep neural networks, which has produced cutting-edge results on a variety of computer vision problems.

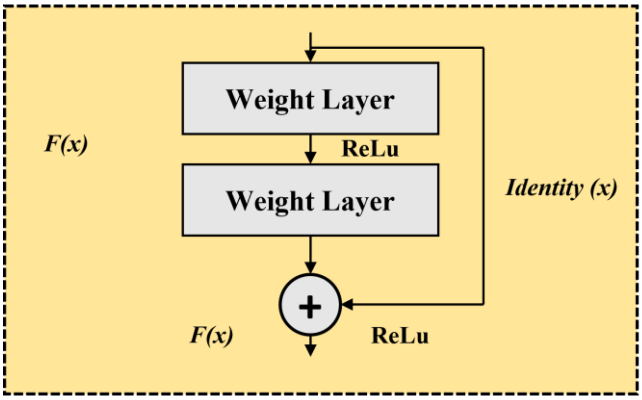


Fig.3. Residual Learning Block

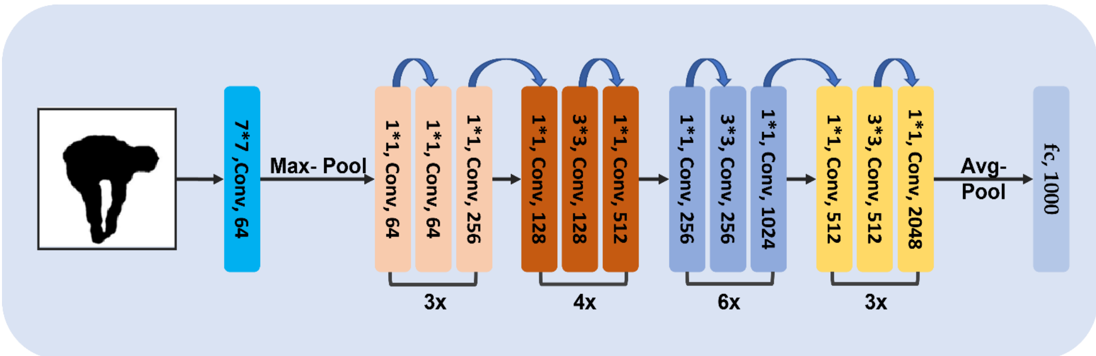


Fig.4. Basic ResNet Architecture

3.4.3 Algorithms Implementation

Convolutional Neural Network is referred to as CNN. For tasks like image classification, image segmentation and object detection, it is a kind of deep learning neural network. Two of the multiple layers present in a CNN are a convolutional layer that performs feature extraction by applying a series of filters to the input image, and a pooling layer that down samples the feature map. The final classification output is produced by a completely connected layer that receives the output from the previous layer. Residual Network, on the other hand, is referred to as ResNet. This particular CNN architecture was unveiled in 2015 and has won numerous computer vision competitions. Utilizing residual connections, also known as skip connections, allows the network to learn residual functions rather than the underlying mapping directly, which is the fundamental idea behind ResNet. This makes it possible to train models with many more layers and lessens the vanishing gradient issue that affects very deep networks. By making network optimization simpler, the residual links also contribute to greater accuracy. The data must first be cleaned, normalized, and divided into validation, training and test sets in order to be ready for CNN implementation. The architecture of the CNN, which includes the number of layers, filter sizes, and activation functions, must next be created. The loss function, optimizer, and metrics are then configured before the model is built. The model must then be trained using the prepared data after being assembled, with the weights and biases being adjusted using backpropagation and gradient descent. After the model has been trained, it must be assessed against the validation set to ascertain its accuracy and, if necessary, modify the hyperparameters. The model is then put to the test on the test set to determine its final CNN accuracy.

ResNet's architecture consists of a batch normalization layer, a ReLU activation function, and many convolutional layers **[23].** The residual block, which has two convolutional layers with the same number of filters and a skip connection to omit the convolutional layers, is the fundamental component of the ResNet architecture. The gradient may travel through the network without being diluted because to this connection, which makes it easier to train very deep networks. A single convolutional layer and a max pooling layer are found in the initial block of the ResNet architecture, which is made up of many blocks of residual units. The number of filters doubles with each block, with the number of residual units increasing with each block. A global average pooling layer and a SoftMax layer are added after the last block to create the output for the final classification. The architecture also uses a method known as bottleneck blocks, which lowers the network's computational complexity by first applying 1x1 convolutional layers to flatten the input's dimensionality before utilising 3x3 convolutional layers. This keeps accuracy high while enabling the network to learn more effectively. For a range of image classification tasks, the ResNet design has consistently shown the model’s performance, and its success has prompted the creation of comparable structures for additional deep learning applications.

Code implements and trains a ResNet model for image classification using the Keras deep learning library. The ResNet model is defined in two functions: resnet_layer and resnet_v1. resnet_layer builds a stack of a a batch normalization layer, 2D convolutional layer and an activation layer (in that order). resnet_v1 defines a ResNet version 1 model architecture that uses the resnet_layer function to build a series of residual blocks. The two convolutional layers of each block are followed by a skip connection with batch normalisation and ReLU activations, This increases the second convolutional layer's output by adding the input. When creating the ResNet model using the compile method of the Keras Model class, the loss function, optimizer, and metric were all set to categorical cross-entropy, Adam, and precision, respectively. The model is then trained using the fit_generator method, which generates batches of training data on the fly using a ImageDataGenerator object **[24].** The training progress is stored in a history object. Finally, the trained model is evaluated on a separate test set using the evaluate_generator method, and the test accuracy is printed to the console.

Conv2D code block defines a function resnet_layer that builds a stack of layers comprising of a Batch Normalization, 2D Convolution and Activation function. The Conv2D layer takes several arguments such as the number of filters, kernel size, stride, padding, kernel initializer, and kernel regularizer. The input tensor x is first set to the inputs argument of the function. If the conv_first argument is True, the Convolution-Batch Normalization-Activation layers are applied in that order. Otherwise, the Batch Normalization-Activation-Convolution layers are applied in that order. In both cases, the Conv2D layer is followed by Activation and Batch Normalization layers, if specified. Finally, the resulting tensor x is returned. This function is used in building the ResNet model, which consists of a stack of residual blocks that include the resnet_layer function.

The function resnet34 takes two arguments as input: input_shape, which defines the shape of the input image, and num_classes, which defines the number of classes that the model will classify the input into. The function starts by creating an input layer using the Input function from the Keras API. Then, the input image is padded with zeros using the ZeroPadding2D function with a padding size of (3,3) to make the input size compatible with the model. The function then defines the ResNet34 architecture in several stages. Batch normalisation and ReLU activation are performed after a convolutional layer with 64 filters, a kernel size of 7x7, and a stride of 2 in the first stage **[25].** Following that, the output is max-pooled with a pool size of (3,3) and a stride of 2. The next stages (2-5) are composed of repeated ResNet34 layers, each consisting of multiple convolutional layers with varying numbers of filters, followed by batch normalization and ReLU activation. The ResNet34 layer function used in these stages has an optional parameter for the number of filters and the stride size. The last layer is a global average pooling layer, which reduces the spatial dimensions of the output of the previous layer to a single dimension. Finally, a dense layer with num_classes neurons and softmax activation is added to produce the final output. The model is then instantiated using the Model function from the Keras API, which takes the input and output layers as arguments, and returns the model. The function returns this instantiated model. The performance shown is the training and validation accuracy and loss of a neural network model over 10 epochs. In each epoch, the model was trained on 72 batches of data, where each batch contains a number of images specified by the batch size. The reported training loss and accuracy values are the average of the losses and accuracies computed over all the batches in the epoch. Similarly, the reported validation loss and accuracy values are the average of the losses and accuracies computed over all the validation batches in the epoch. The performance metrics show that the model improved significantly from the first epoch to the last. Training accuracy went from 0.62 to 0.96, while the training loss went from 2.40 to 0.25. The validation loss and accuracy both increased, indicating that the model may not be overfit to the training set of data. The final validation accuracy of 0.84 is quite respectable and shows that the model can accurately categorize the photos in the validation set.

CHAPTER 4

RESULTS ANALYSIS

During implementation multiple layers are implemented. Variations are done with the number of layers and neurons present in every layer. A type of layer is "dense" A typical layer type that works in most circumstances is dense. In a dense layer, every node from the layer before is connected to every node from the layer in front of it, model capacity increases as the number of nodes in each layer increases. The layer's activation function is referred to as "activation." Models can consider relationships that are not linear thanks to an activation function. Rectified Linear Activation, often known as ReLU, will be the activation function we employ. It has been demonstrated that, despite having two linear parts, it performs well in neural networks. An input shape is required for the top layer. pixels make up the input layer. The output layer produced is the final one. Only four of its nodes are for our prediction layer. Classification is carried out in this layer. The model needs to be assembled next. The two parameters required to create the model are loss and optimizer. The learning rate is controlled by the optimizer. In this situation, Adam is the optimizer. Adam is a useful optimizer to use in many circumstances. The Adam Optimizer changes the learning rate throughout training. The learning rate determines how soon the best weights for the model are computed. A lower learning rate may provide more accurate weights up to a point, but the computation of the weights will take longer. This work will include the practice on machine learning models. Utilization of the 'fit()' function to train. The data is randomly split between testing and training groups during validation split. The validation loss, which represents the mean squared error of the model applied on the set of validation, will be visible to us during training, how frequently the model iterates over the data is determined by the number of epochs.

4.1 CNN implementation with 16 Neurons with 4 Hidden Layers

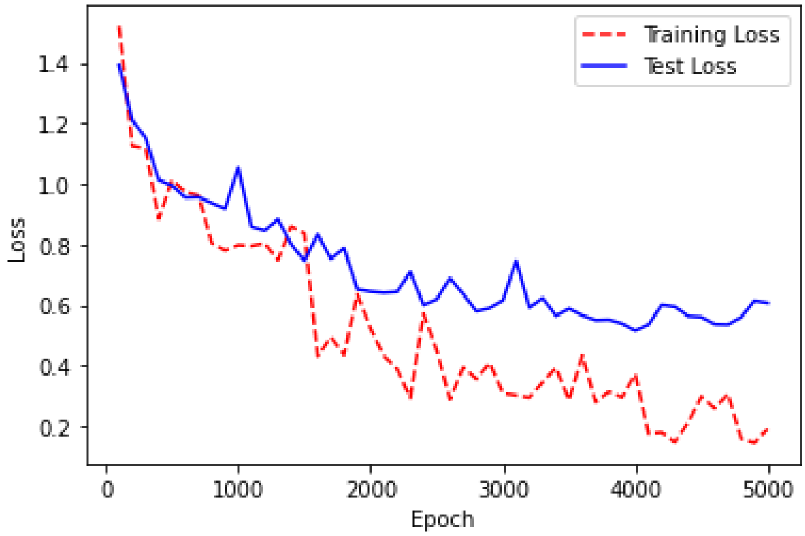


Fig. 5(a) CNN implementation with 16 Neurons with 4 Hidden Layers

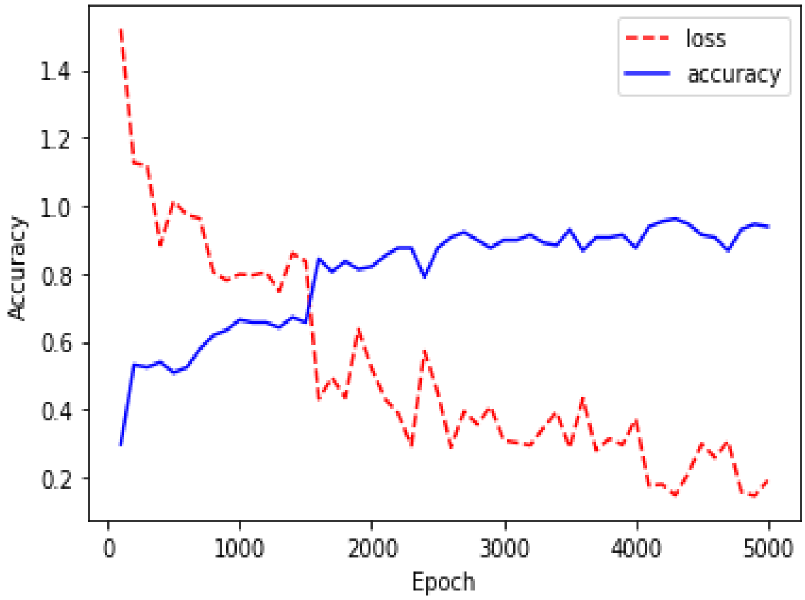


Fig. 5(b) CNN implementation with 16 Neurons with 4 Hidden Layers

We start with implementing 4 CNN layers each having 16 neurons this implementation results in accuracy of 93.75%. while training and loss of 19.04% during the same. During validation we see loss of 60.69% and accuracy of 81.88%.

4.2 CNN implementation with 32 Neurons with 4 Hidden Layers

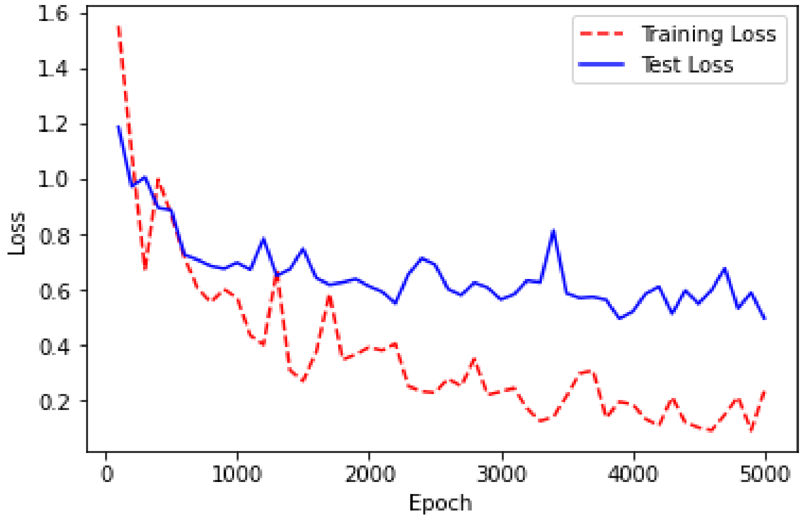


Fig. 6(a) 32 CNN implementation with 32 Neurons with 4 Hidden Layers

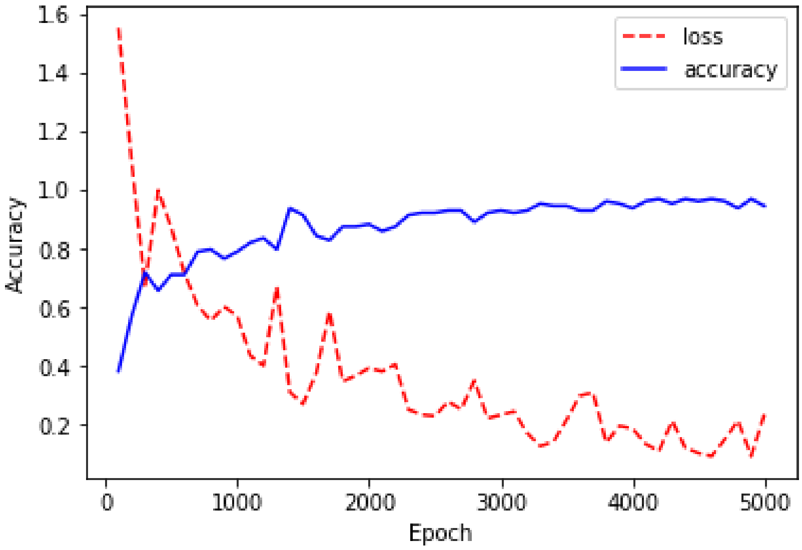


Fig. 6(b) 32 CNN implementation with 32 Neurons with 4 Hidden Layers

On implementation of 4 CNN layers each having 32 neurons this implementation results in accuracy of 94.75%. while training and loss of 23.46% during the same. During validation we see loss of 49.67% and accuracy of 83.85%.

4.3 CNN implementation with 64 Neurons with 4 Hidden Layers

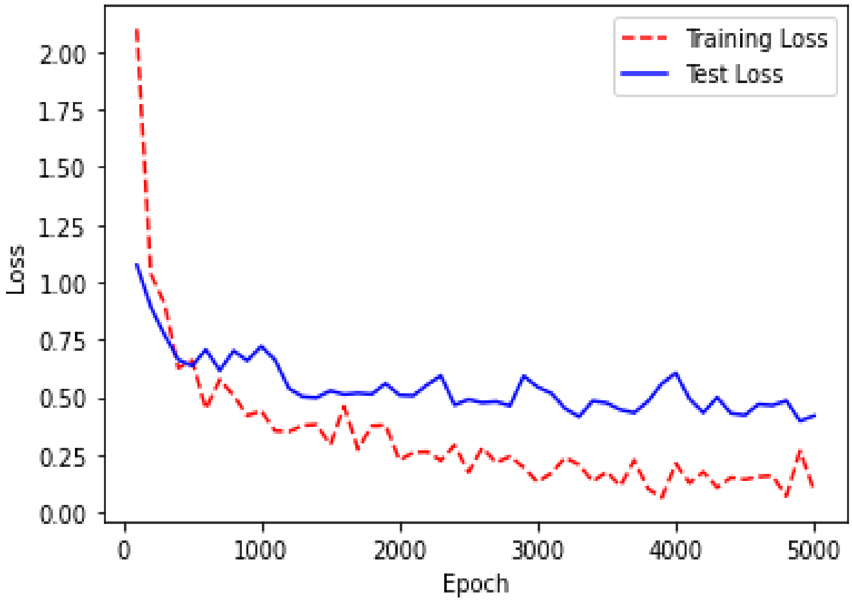


Fig. 7(a) CNN implementation with 64 Neurons with 4 Hidden Layers

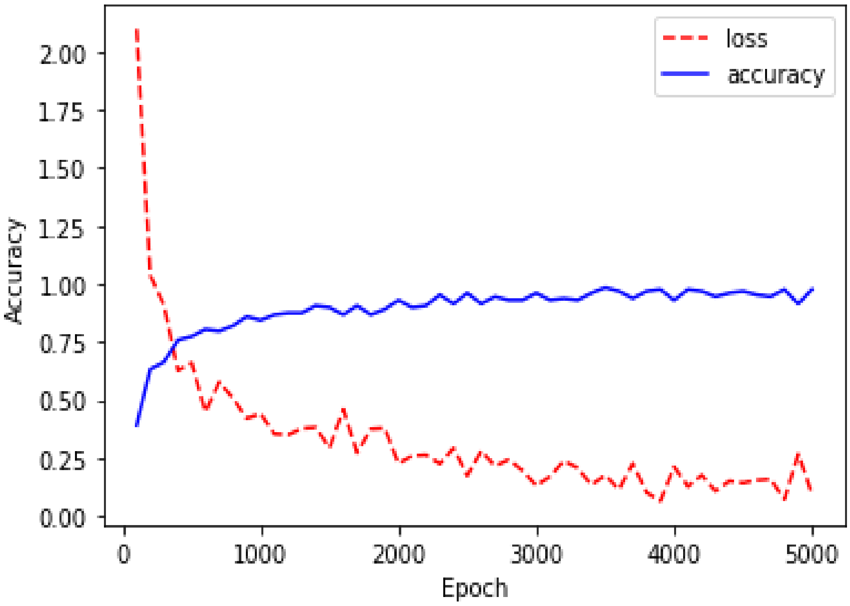


Fig. 7(b) CNN implementation with 64 Neurons with 4 Hidden Layers

On implementation of 4 CNN layers each having 64 neurons, training accuracy is 97.66% and there is loss of 9.62% during the same. During validation we see loss of 41.93% and accuracy of 85.94%.

4.4 CNN implementation with 128 Neurons with 4 Hidden Layers

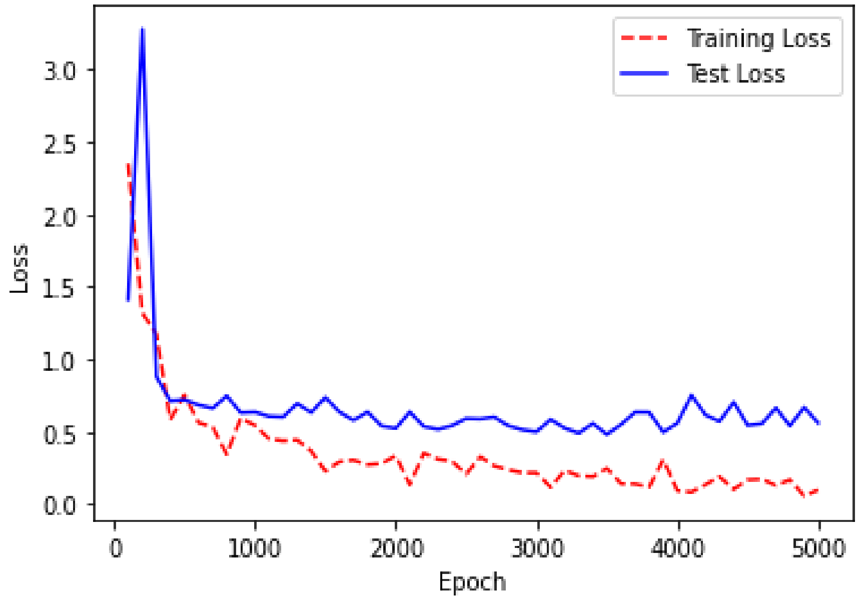


Fig. 8(a) CNN implementation with 128 Neurons with 4 Hidden Layers

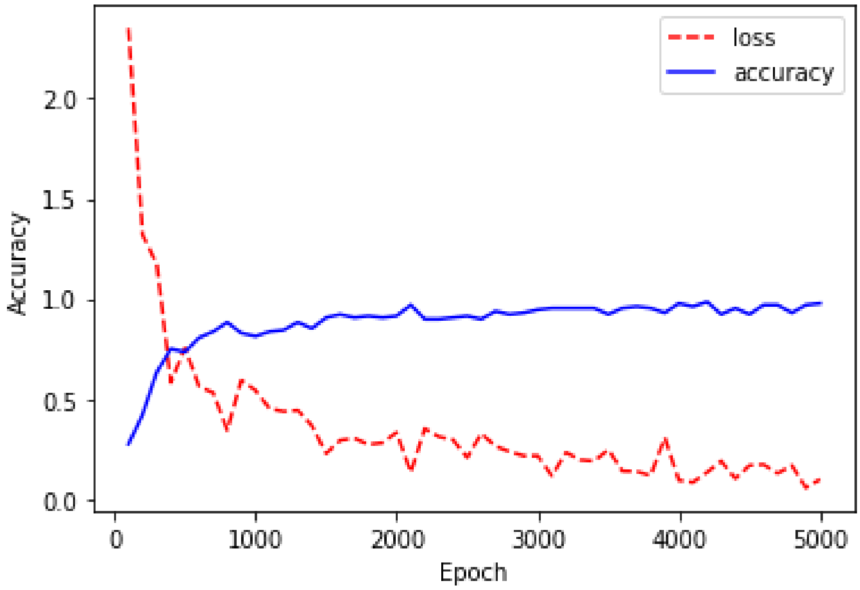


Fig.8(b) CNN implementation with 128 Neurons with 4 Hidden Layers

On implementation of CNN 4 hidden layers each having 64 neurons, training accuracy is 97.66% and there is loss of 9.62% during the same. During validation we see loss of 41.93% and accuracy of 85.94%.

4.5 First hidden layer with 16 neurons, Second hidden layer with 32 neurons, Third hidden layer with 128 neurons and Fourth hidden layer with 256 neurons.

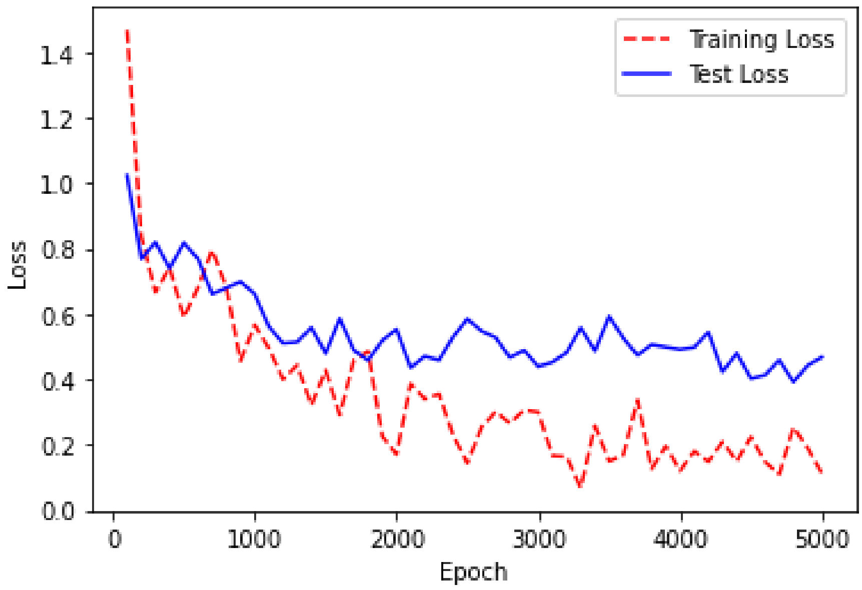


Fig. 9(a) 16 in First, 32 in second, 64 in third and 256 neurons in fourth Hidden Layer

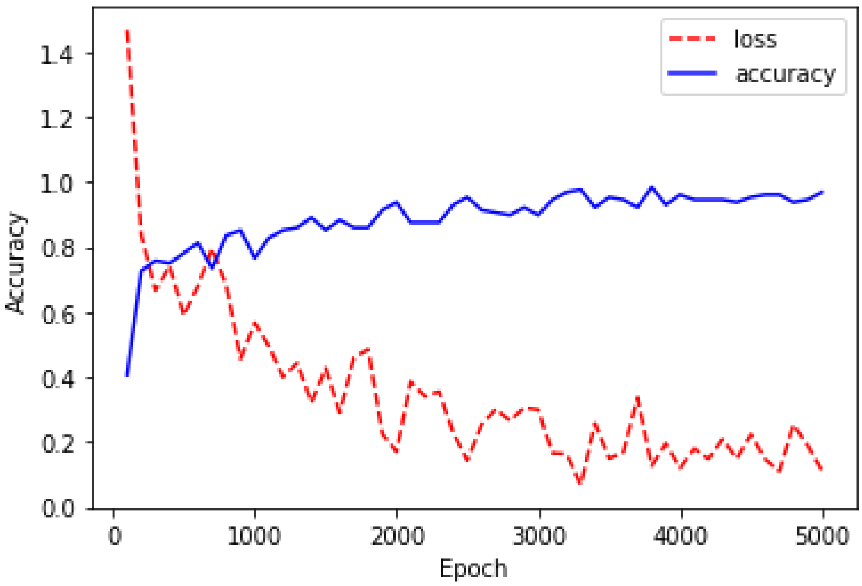


Fig.9(b) 16 in First, 32 in second, 64 in third and 256 neurons in fourth Hidden Layer

We then implemented 5 layers with having 16,32,64,128 and 256 nodes respectively training accuracy is 96.09% and there is loss of 10.90%. During validation we see loss of 45.81% and accuracy of 87.40%.

Layers & Neurons	Training loss	Training accuracy	Val loss	Val accuracy
4 & 16	19.04%	93%	60.69%	81.88%
4 & 32	23.46%	94%	49.67%	83.85%
4 & 64	9.62%	97.66%	40.93%	85.94%
4 & 128	9.62%	97%	41.93%	85.94%

Table 2. CNN Comparison summary Table

4.6 ResNet implementation with 20 Layers

In following graph is generated with Resnet is implemented with 20 Layers

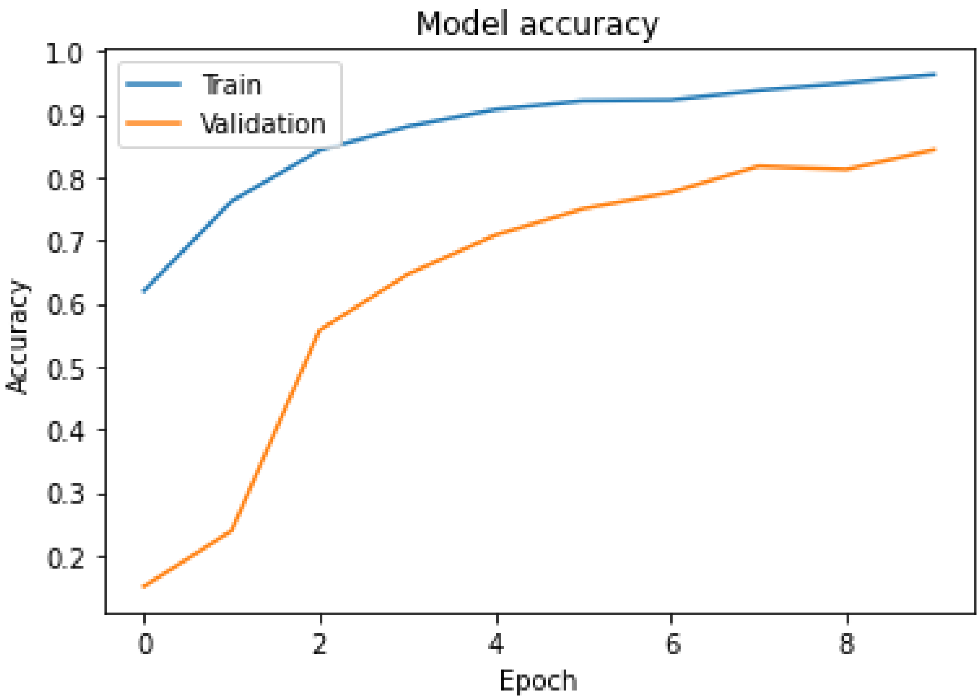


Fig.10(a) ResNet implementation with 20 Layers

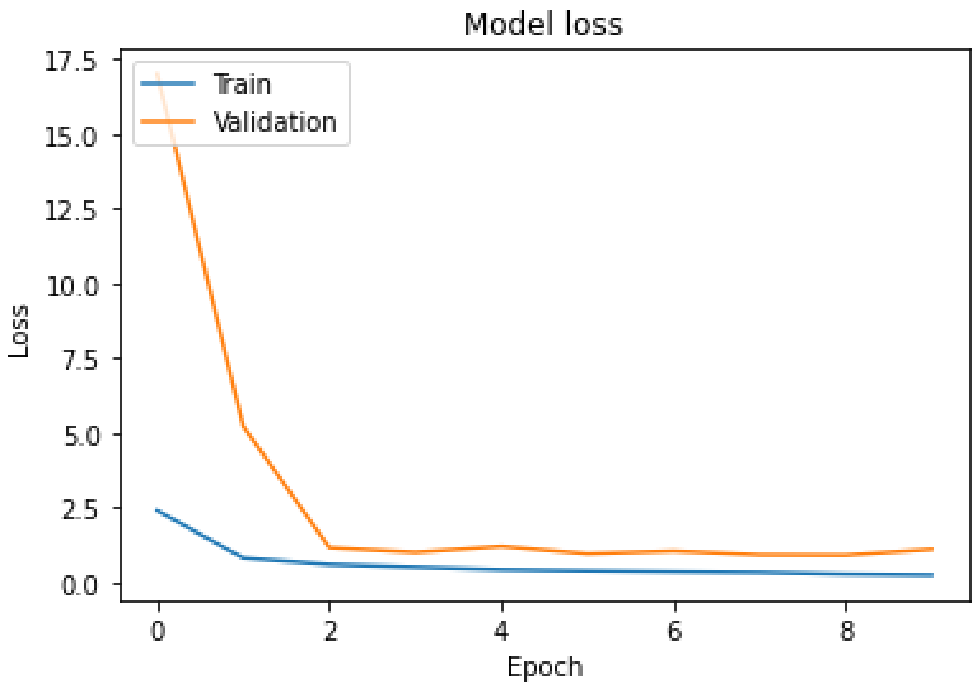


Fig.10(b) ResNet implementation with 20 Layers

The performance metrics (Fig 4(a) & Fig 4(b)) for ResNet with 20 layers show that the model improved significantly from the first epoch to the last. Training accuracy went from 0.62 to 0.96, while the training loss went from 2.40 to 0.25. The validation loss and accuracy both increased, indicating that the model may not be overfit to the training set of data. The model can accurately categories photos in the validation set, as shown by the final validation accuracy of 0.84, which is pretty good.

4.7 ResNet 18 implementation

In the following graph is generated with Resnet 18 is implemented.

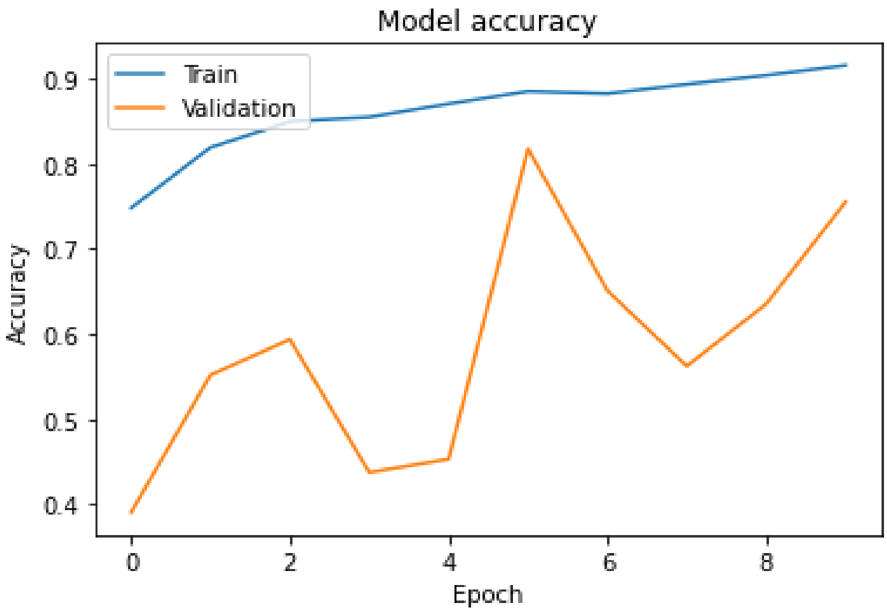


Fig.11(a) ResNet 18 implementation

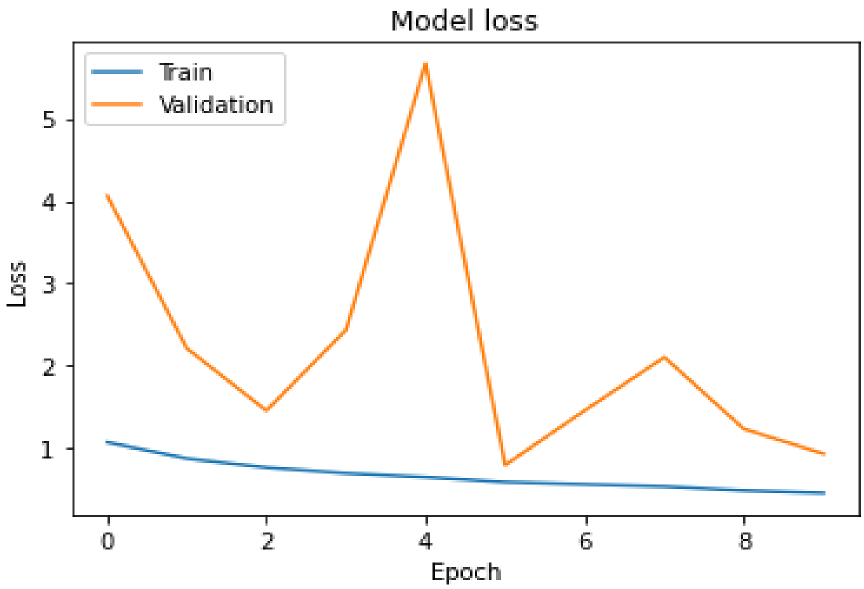


Fig.11(b) ResNet 18 implementation

For resnet18 (Fig 5 (a) & Fig 5(b)), In the first epoch, the loss was 1.0718 and the accuracy was 0.7483 on the training set. The validation loss was 4.0654 and the validation accuracy was 0.3906. This means that the network did not perform very well on the validation set and was likely overfitting to the training data. In subsequent epochs, the training loss and accuracy improved, which indicates that the network was learning from the data. However, the validation loss and accuracy were more erratic, with some epochs showing improvement and others showing degradation. Overall, the network seems to have performed reasonably well, with a final validation accuracy of 0.7552, which indicates that it was able to generalize to new data. However, there is still room for improvement, as the validation loss was not consistently decreasing throughout the training process.

4.8 ResNet 34 implementation

In following graph is generated with Resnet 34 is implemented.

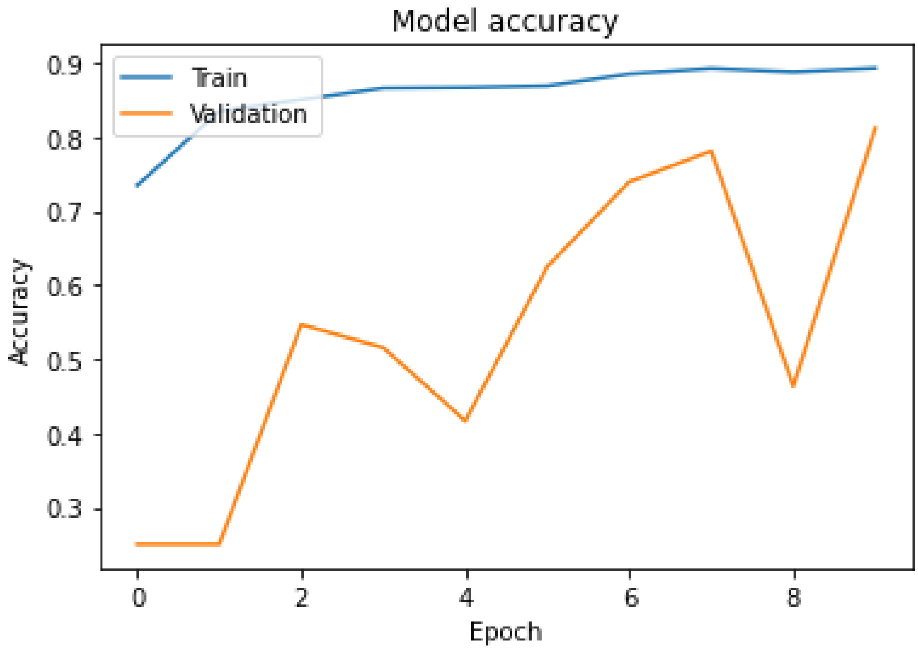


Fig.12(a) ResNet 34 implementation

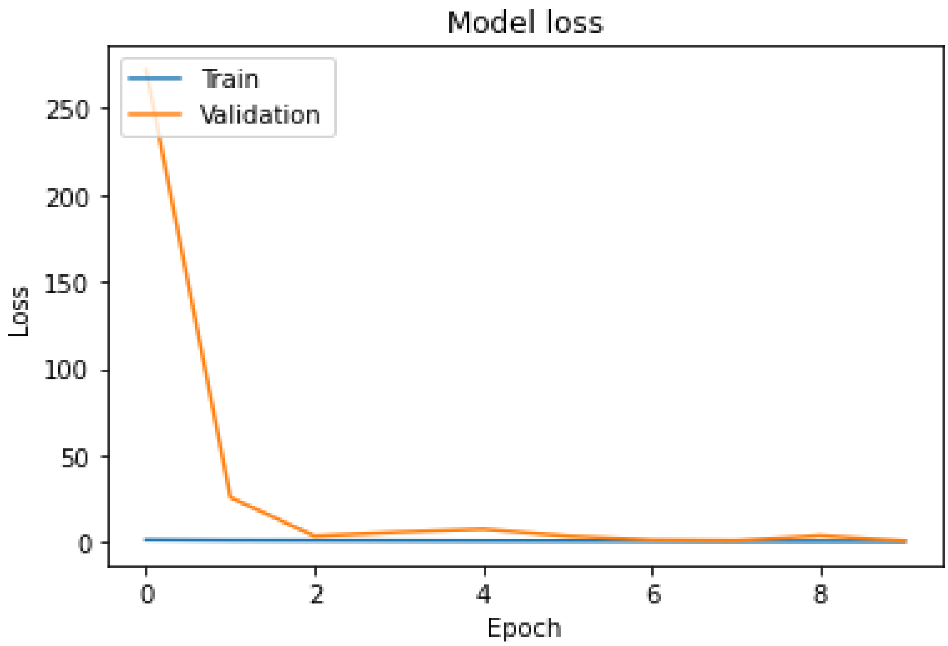


Fig.12(b) ResNet 34 implementation

In implementation of resnet34 (Fig 6(a) & Fig 6(b)), the training and validation loss values drop with time, as can be seen, demonstrating the model's improvement in prediction accuracy. The training and validation accuracy numbers are increasing at the same time, which suggests that the model is improving its ability to correctly identify the input data. It is significant to notice that there are differences between the validation loss and accuracy values and the training loss and accuracy values. We can determine how effectively the model generalizes to fresh, untested data by looking at the validation measures. In this instance, it appears that the model's validation loss and accuracy values change over the course of training, which may be a sign that the model has been overfitted to the training set. Overfitting, when a model becomes very complex and starts to remember training data rather than learning generalizable patterns, is a prevalent problem in deep learning. With a validation accuracy of 81.25% in the most recent epoch, the model appears to perform well overall.

CHAPTER 5

CONCLUSIONS

5.1 Conclusion

The work constitutes of CNN and ResNet architecture, which is used for Human Posture Estimation. constitutes Convolutional Neural Network architecture, which is used for Human Posture Estimation, this work uses altering of CNN for the purpose mentioned, which gives accuracy up-to 98%. Finally, we found a great illustration of accuracy using graphs with defined accuracy and losses while adjusting neurons and layers with varying parameters. This work also uses variation of ResNet's models for the purpose mentioned, which gives training accuracy up-to 96.31% and validation accuracy of 86%. ResNet has demonstrated to be a successful method for estimating human posture. ResNet has demonstrated promise in reliably detecting human postures from photos or video data due to its capacity to handle complicated features and interactions between them. By enabling the flow of gradients to skip some network layers, ResNet's skip connections enable the development of significantly deeper architectures, which can result in higher accuracy and reduced error rates. ResNet is a potent tool for estimating human posture, and additional research in this field is anticipated to produce even more promising outcomes. However, difficulties like postural variability and computational resources still exist. Finally, we found a great illustration of accuracy using graphs with defined accuracy and losses while adjusting neurons and layers with varying parameters.

5.2 Future Scope

The future scope for human posture estimation using ResNet:

1. Multi-scale ResNet: The use of multi-scale ResNet architectures to capture features at different scales can improve the accuracy of human posture estimation.
2. Real-time human posture estimation: Future research could explore ways to optimize the ResNet architecture for real-time inference, such as by reducing the number of layers or using lightweight architectures.

3. Robustness to occlusions and variations: Improving the robustness of ResNet-based models to occlusions and variations is important for human posture estimation in real-world scenarios.

CHAPTER 6

REFERENCES

[1] Hua, G., Li, L., & Liu, S. (2020). Multipath affinage stacked—hourglass networks for human pose estimation. Frontiers of Computer Science, 14, 1-12.

[2] Ferrari, V., Marin-Jimenez, M., & Zisserman, A. (2008, June). Progressive search space reduction for human pose estimation. In 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8). IEEE.

[3] Sengupta, A., Jin, F., Zhang, R., & Cao, S. (2020). mm-Pose: Real-time human skeletal posture estimation using mmWave radars and CNNs. *IEEE Sensors Journal*, *20*(17), 10032-10044.

[4] Chéron, G., Laptev, I., & Schmid, C. (2015). P-cnn: Pose-based cnn features for action recognition. In Proceedings of the IEEE international conference on computer vision (pp. 3218-3226).

[5] Meng, Z., Zhang, M., & Wang, H. (2020). CNN with pose segmentation for suspicious object detection in MMW security images. *Sensors*, *20*(17), 4974.

[6] Ayre-Storie, A., & Zhang, L. (2021, December). Deep Learning-Based Human Posture Recognition. In 2021 International Conference on Machine Learning and Cybernetics (ICMLC) (pp. 1-6). IEEE.

[7] Nguyen, H. C., Nguyen, T. H., Scherer, R., & Le, V. H. (2022). Unified end-to-end YOLOv5-HR-TCM framework for automatic 2D/3D human pose estimation for real-time applications. Sensors, 22(14), 5419.

[8] Nguyen, N. H., Phan, T. D. T., Lee, G. S., Kim, S. H., & Yang, H. J. (2020). Gesture recognition based on 3D human pose estimation and body part segmentation for RGB data input. *Applied Sciences*, *10*(18), 6188.

[9] Gao, Q., Liu, J., Ju, Z., & Zhang, X. (2019). Dual-hand detection for human–robot interaction by a parallel network based on hand detection and body pose estimation. *IEEE Transactions on Industrial Electronics*, *66*(12), 9663-9672.

[10] Mohammadi, S. M., Enshaeifar, S., Hilton, A., Dijk, D. J., & Wells, K. (2020). Transfer learning for clinical sleep pose detection using a single 2D IR camera. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *29*, 290-299.

[11] Liu, J., Luo, H., & Liu, H. (2022). Deep learning-based data analytics for safety in construction. *Automation in Construction*, *140*, 104302.

[12] Ogundokun, R. O., Maskeliūnas, R., & Damaševičius, R. (2022). Human posture detection using image augmentation and hyperparameter-optimized transfer learning algorithms. Applied Sciences, 12(19), 10156.

[13] Ning, G., Zhang, Z., & He, Z. (2017). Knowledge-guided deep fractal neural networks for human pose estimation. *IEEE Transactions on Multimedia*, *20*(5), 1246-1259.

[14] Xiao, X., & Wan, W. (2017). Human pose estimation via improved ResNet-50.

[15] Bin, Y., Chen, Z. M., Wei, X. S., Chen, X., Gao, C., & Sang, N. (2020). Structure-aware human pose estimation with graph convolutional networks. *Pattern Recognition*, *106*, 107410.

[16] Ogundokun, R. O., Maskeliūnas, R., Misra, S., & Damasevicius, R. (2022). A Novel Deep Transfer Learning Approach Based on Depth-Wise Separable CNN for Human Posture Detection. *Information*, *13*(11), 520.

[17] Debnath, B., O'Brien, M., Yamaguchi, M., & Behera, A. (2018, November). Adapting MobileNets for mobile based upper body pose estimation. In 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) (pp. 1-6). IEEE.

[18] Wang, K., Lin, L., Jiang, C., Qian, C., & Wei, P. (2020). 3D Human Pose Machines with Self-Supervised Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(5), 1069–1082. https://doi.org/10.1109/TPAMI.2019.2892452

[19] Varamesh, A., Tuytelaars ESAT-PSI, T., & Leuven, K. (n.d.). Mixture Dense Regression for Object Detection and Human Pose Estimation. https://github.com/alivaramesh/MixtureDenseRegression

[20] Ren, W., Ma, O., Ji, H., & Liu, X. (2020). Human Posture Recognition Using a Hybrid of Fuzzy Logic and Machine Learning Approaches. IEEE Access, 8, 135628–135639. https://doi.org/10.1109/ACCESS.2020.3011697

[21] Matuska, S., Paralic, M., & Hudec, R. (2020). A Smart System for Sitting Posture Detection Based on Force Sensors and Mobile Application. Mobile Information Systems, 2020. https://doi.org/10.1155/2020/6625797

[22] Bao, W., Ma, Z., Liang, D., Yang, X., & Niu, T. (2023). Pose ResNet: 3D Human Pose Estimation Based on Self-Supervision. *Sensors*, *23*(6), 3057.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[24] Li, S., & Chan, A. B. (2015). 3d human pose estimation from monocular images with deep convolutional neural network. In Computer Vision--ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II 12 (pp. 332-347). Springer International Publishing.

[25] Valdez-Rodríguez, J. E., Calvo, H., Felipe-Riverón, E., & Moreno-Armendáriz, M. A. (2022). Improving depth estimation by embedding semantic segmentation: a hybrid CNN model. *Sensors*, *22*(4), 1669.

[26] Ji, P., Wang, X., Ma, F., Feng, J., & Li, C. (2022). A 3D Hand Attitude Estimation Method for Fixed Hand Posture Based on Dual-View RGB Images. *Sensors*, *22*(21), 8410.

[27] Liaqat, S., Dashtipour, K., Arshad, K., Assaleh, K., & Ramzan, N. (2021). A hybrid posture detection framework: Integrating machine learning and deep neural networks. IEEE Sensors Journal, 21(7), 9515-9522.

[28] Mao, W., Tian, Z., Wang, X., & Shen, C. (2021). Fcpose: Fully convolutional multi-person pose estimation with dynamic instance-aware convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9034-9043).

[29] Li, Y., Zhang, S., Wang, Z., Yang, S., Yang, W., Xia, S. T., & Zhou, E. (2021). Tokenpose: Learning keypoint tokens for human pose estimation. In Proceedings of the IEEE/CVF International conference on computer vision (pp. 11313-11322).

ANNEXURE 1

The presentation for the research paper entitled “Using Convolutional Neural Network for Human Posture Estimation: A study of the effects of number of layers and number of neurons on accuracy” has been **accepted** in the IEEE International Conference on Disruptive Technologies (ICDT-2023) and the presentation for the same is scheduled on 11th May.

Paper Title:

Using Convolutional Neural Network for Human Posture Estimation: A study of the effects of number of layers and number of neurons on accuracy.

Abstract:

Human Posture estimation is a field which gathers huge researchers interest due to its variations in different machine learning (ML) & deep learning (DL) architectures to estimate human postures This work includes tweaking of layers with varying neurons in Convolutional Neural Network Architecture to test which pair of neurons and layers gives the best accuracy, also visualizing each of the pair with help of graphs. The results of this work provide great results with high accuracy.

Authors: Nalin Kashyap, Satnam Singh, Viswajeet Kumar, Kanika Singla

ICDT 2023 Authors guidelines for the presentation (Virtual/Physical) Mode External

ICDT2023 <icdt2023-0@easychair.org> Fri, Apr 28, 2:44 PM (4 days ago)
to me

Dear Satnam Singh,

Greeting from ICDT 2023 team from GL Bajaj Institute of Technology and Management, Greater Noida. Thank you for the part of the ICDT 2023 conference.

The Presentation guidelines are as follows :

Oral presentation guidelines (Physical)
• The allocated time for Oral Presentations is 20 minutes including 15 minutes for presentation and 5 minutes for questions.
• PowerPoint or similar presentations should be brought to the conference on a USB and be PC compatible.
• Your presentation should be loaded onto the lectern computer and tested prior to your session (during registration time, coffee breaks, or lunchtime) so that they run smoothly without any delays. Student volunteers will be available in each session room to assist with loading your presentations onto the lectern computer.
• All session rooms will have projector screens and a computer with Microsoft PowerPoint and Adobe PDF reader.
• Please arrive at the session room at least 10 minutes before the start of the session and introduce yourself to the Session Chair and student assistant.

CODING SNIPPETS

Convolutional Neural Network

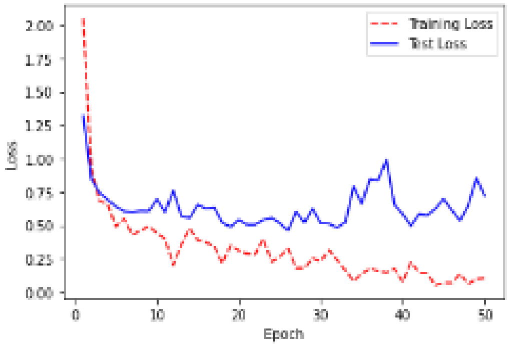
In [5]:
```
model = tf.keras.models.Sequential([
tf.keras.layers.Conv2D(128,(3,3),activation ="relu", input_shape =(512,512,3)),
tf.keras.layers.MaxPool2D(2,2),
tf.keras.layers.Conv2D(128,(3,3),activation ="relu"),
tf.keras.layers.MaxPool2D(2,2),
tf.keras.layers.Conv2D(128,(3,3),activation ="relu"),
tf.keras.layers.MaxPool2D(2,2),
# tf.keras.layers.Conv2D(128,(3,3),activation ="relu"),
# tf.keras.layers.MaxPool2D(2,2),
# tf.keras.layers.Conv2D(256,(3,3),activation ="relu"),
# tf.keras.layers.MaxPool2D(2,2),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation = "relu"),
tf.keras.layers.Dense(4,activation ="softmax")])
```

In [6]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 510, 510, 128)     3584

 max_pooling2d (MaxPooling2D  (None, 255, 255, 128)    0
 )

 conv2d_1 (Conv2D)           (None, 253, 253, 128)     147584

 max_pooling2d_1 (MaxPooling  (None, 126, 126, 128)    0
 2D)

 conv2d_2 (Conv2D)           (None, 124, 124, 128)     147584

 max_pooling2d_2 (MaxPooling  (None, 62, 62, 128)      0
 2D)

 flatten (Flatten)           (None, 492032)            0

 dense (Dense)               (None, 128)               62980224

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 63,279,492
Trainable params: 63,279,492
Non-trainable params: 0
_________________________________________________________________
```

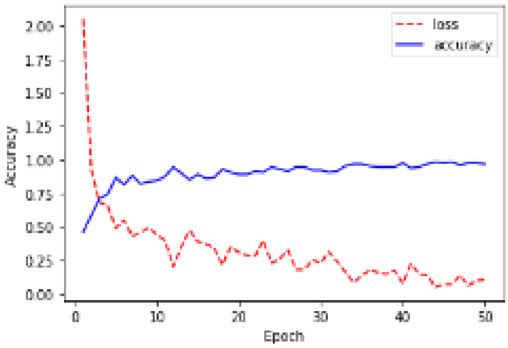
In [10]:

```
import matplotlib.pyplot as plt
# Get training and test loss histories
training_loss = model_fit.history['loss']
test_loss = model_fit.history['val_loss']

# Create count of the number of epochs
epoch_count = range(1, len(training_loss) + 1)

# Visualize loss history
plt.plot(epoch_count, training_loss, 'r--')
plt.plot(epoch_count, test_loss, 'b-')
plt.legend(['Training Loss', 'Test Loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show();
```



In [11]:

```
loss = model_fit.history['loss']
accuracy = model_fit.history['accuracy']

# Create count of the number of epochs
epoch_count = range(1, len(accuracy) + 1)

# Visualize loss history
plt.plot(epoch_count, loss, 'r--')
plt.plot(epoch_count, accuracy, 'b-')
plt.legend(['loss', 'accuracy'])
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.show();
```



Residual Neural Network (ResNet)

```python
def resnet_v1(input_shape, depth, num_classes=10):
    """ResNet Version 1 Model builder [a]
    Stacks of 2 x (3 x 3) Conv2D-BN-ReLU
    Last ReLU is after the shortcut connection.
    At the beginning of each stage, the feature map size is halved (downsampled)
    by a convolutional layer with strides=2, while the number of filters is
    doubled. Within each stage, the layers have the same number filters and the
    same number of filters.
    Features maps sizes:
    stage 0: 32x32, 16
    stage 1: 16x16, 32
    stage 2:  8x8,  64
    The Number of parameters
    """
    if (depth - 2) % 6 != 0:
        raise ValueError('depth should be 6n+2 (eg 20, 32, 44 in [a])')
    # Start model definition.
    num_filters = 16
    num_res_blocks = int((depth - 2) / 6)

    inputs = Input(shape=input_shape)
    x = resnet_layer(inputs=inputs)
    # Instantiate the stack of residual units
        # Instantiate the stack of residual units
    for stack in range(3):
        for res_block in range(num_res_blocks):
            strides = 1
            if stack > 0 and res_block == 0:  # first layer but not first stack
                strides = 2  # downsample
            y = resnet_layer(inputs=x,
                             num_filters=num_filters,
                             strides=strides)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters,
                             activation=None)
            if stack > 0 and res_block == 0:  # first layer but not first stack
                # linear projection residual shortcut connection to match
                # changed dims
                x = resnet_layer(inputs=x,
                                 num_filters=num_filters,
                                 kernel_size=1,
                                 strides=strides,
                                 activation=None,
                                 batch_normalization=False)
            x = keras.layers.add([x, y])
            x = Activation('relu')(x)
        num_filters *= 2

    # Add classifier on top.
    # v1 does not use BN after last shortcut connection-ReLU
    x = AveragePooling2D(pool_size=(8,8))(x)
    y = Flatten()(x)
    outputs = Dense(num_classes,
                    activation='softmax',
                    kernel_initializer='he_normal')(y)

    # Instantiate model.
    model = Model(inputs=inputs, outputs=outputs)
    return model

# Instantiate the ResNet model
model = resnet_v1(input_shape=(512, 512, 1), depth=20, num_classes=4)


# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Fit the model using the generator
history = model.fit_generator(train_generator,
                              validation_data=validation_generator,
                              steps_per_epoch=train_generator.samples // 32,
                              validation_steps=validation_generator.samples // 32,
                              epochs=10)
```

In [6]:

```
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```