**HUMAN POSE ESTIMATION**

**USING DEEP LEARNING APPROACHES**

*Report submitted in fulfillment of the requirements for the*

***Exploratory Project of Second Year B.Tech***

*By:*

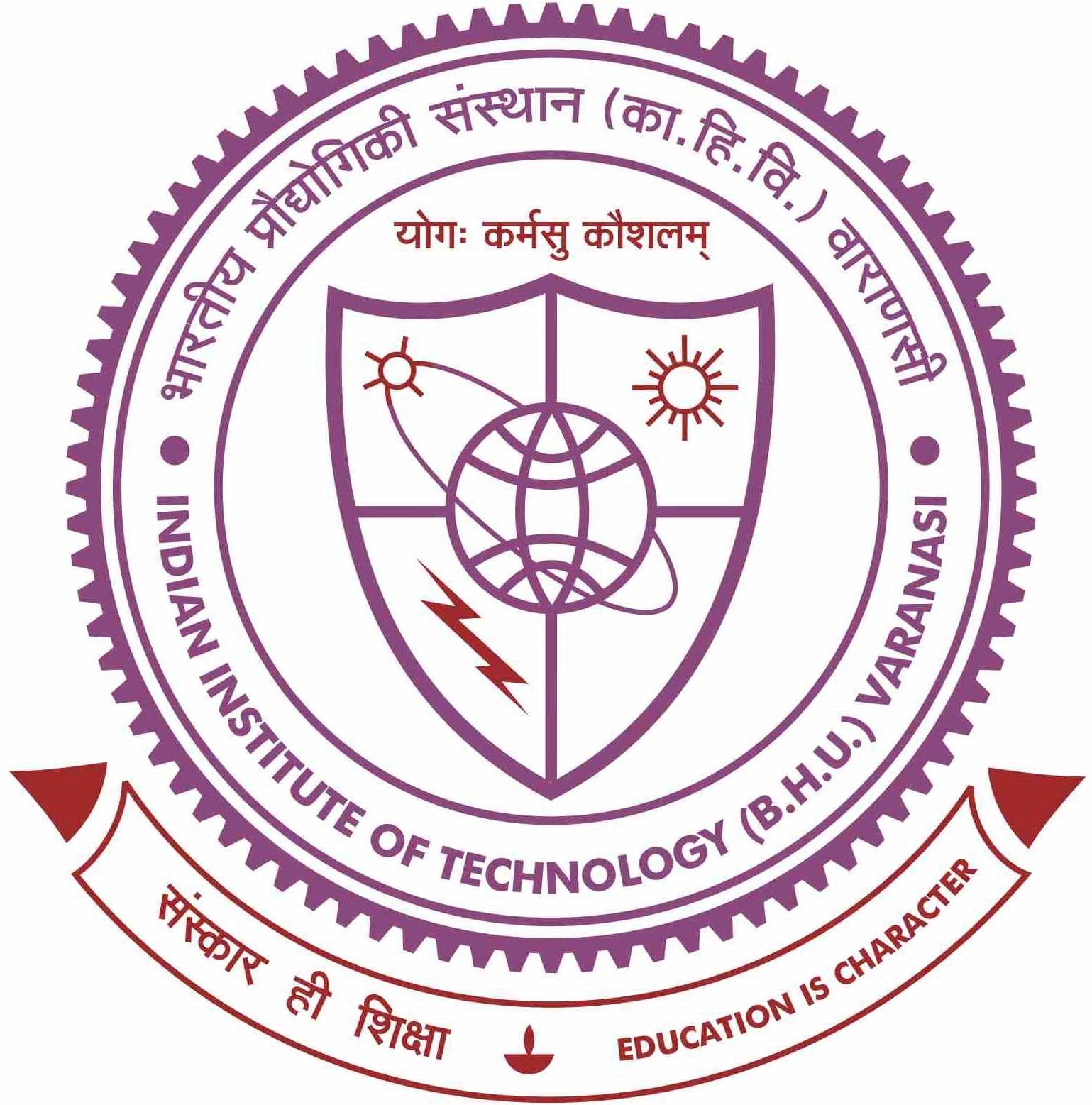
**Lakshya Gupta (17075029)**

**Janhavi Gupta (17075061)**

*Under the guidance of:*

**Dr. Rajeev Srivastava**

Professor and Head of the Department



Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY (B.H.U.) VARANASI

Varanasi, INDIA 221005

May 2019

**DEDICATED TO**

Our parents, without whom, we would not have been able to reach where we are right now.

Our teachers, who looked over us and guided us at every step with a big heart and kind words.

Our seniors, who advised whenever we got stuck at any point of time.

**DECLARATION**

We certify that

1. The work contained in this report is original and has been done by ourselves and the general supervision of our supervisor.

2. The work has not been submitted for any project.

3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi Lakshya Gupta and Janhavi Gupta

Date: B.Tech

Department of Computer Science and Engineering,

Indian Institute of Technology (BHU) Varanasi,

Varanasi, INDIA 221005.

**CERTIFICATE**

This is to certify that the work contained in this report entitled “Human Pose Estimation” being submitted by Lakshya Gupta (Roll No. 17075029) and Janhavi Gupta (Roll No. 17075061), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.

Place: IIT (BHU) Varanasi Prof. Rajeev Srivastava

Date: Department of Computer Science and Engineering,

Indian Institute of Technology (BHU) Varanasi,

Varanasi, INDIA 221005.

**ACKNOWLEDGEMENTS**

We would like to express our sincere gratitude to Dr. Rajeev Srivastava for his guidance, along Pratishtha Verma Ma’am for her support.

Place: IIT (BHU) Varanasi Lakshya Gupta and Janhavi Gupta

Date:

**ABSTRACT**

Human pose estimation holds a great significance in computer vision in today’s world. Since humans form a class of highly deformable objects, making human pose estimation a big challenge.

Recent developments in deep learning algorithms have greatly improved the performance of computer vision tasks. Deep learning has revolutionized the way we do machine learning and create intelligent systems. The era of big data and technological advancements in computational resources have allowed us to exploit the benefits of deep learning frameworks. One of the biggest gain deep learning gives us is its ability to learn from unstructured and unlabelled data.

We propose a two-step method for human pose estimation. Both steps involve the use of deep learning. The first step is human detection in still images. The second step is human pose estimation and generating an estimate skeleton for the detected human.

**CONTENT**

1. Related Work

2. Algorithms

2.1 Convolutional Neural Network

2.2 Residual Networks

2.3 YOLO Algorithm

2.4 Adam Optimisation

3. Methodology

3.1 Human Detection

3.2 Pose Estimation

4. Results and Comparison

5. Conclusion

6. Bibliography

**LIST OF TABLES**

Table 1: Modified Layers for InceptionV3

Table 2: Modified Layers for VGG19 Variant 1

Table 3: Modified Layers for VGG10 Variant 2

Table 4: PCK Scores for VGG19 Variants

Table 5: Key for Graph 3

**LIST OF FIGURES**

Graph 1: Comparison of PCKh Scores

Graph 2: PCK Scores for VGG19 Variants

Graph 3: Comparison of VGG19 Variants

**1. Related Work**

Object detection is one of the most fundamental aspects of computer vision. Over the years, a great number of algorithms have been devised to address this.

In R-CNN (Region-based CNN) and its variant algorithms (Fast R-CNN, Faster R-CNN, Mask R-CNN), the basic idea is to break the image into multiple regions of interest based on their color similarities, texture similarities, shape compatibility, etc and classify each region into classes.

These algorithms posed a few problems; the training process was too slow and the network was slow even at inference time. SSD Multibox (Single Shot Multibox Detector) architecture gave us a more efficient means of object detection. Here, object localization and classification occurred in a single forward pass of the network.

YOLO (You Only Look Once) is also one of the fast object detection algorithms we have. It breaks the input image into a grid of smaller boxes, where each box is responsible for certain bounding boxes. The bounding boxes are classified into object classes and the bounding boxes with acceptable confidence and classification scores are accepted.

Human detection is still more challenging than most of other object detection tasks because of the high deformability of human shape.

A lot of work has been done on human pose estimation as well, including deep learning too. DeepPose architecture has an AlexNet backend, and the model’s redeeming feature is refinement using cascaded regressors. A course prediction is cropped around the prediction values and is refined by later stages by working on the higher resolution region. Another human pose estimation methodology involves using a network resembling fully-connected GoogLeNet twice, once for the full resolution image and another for the half resolution image.

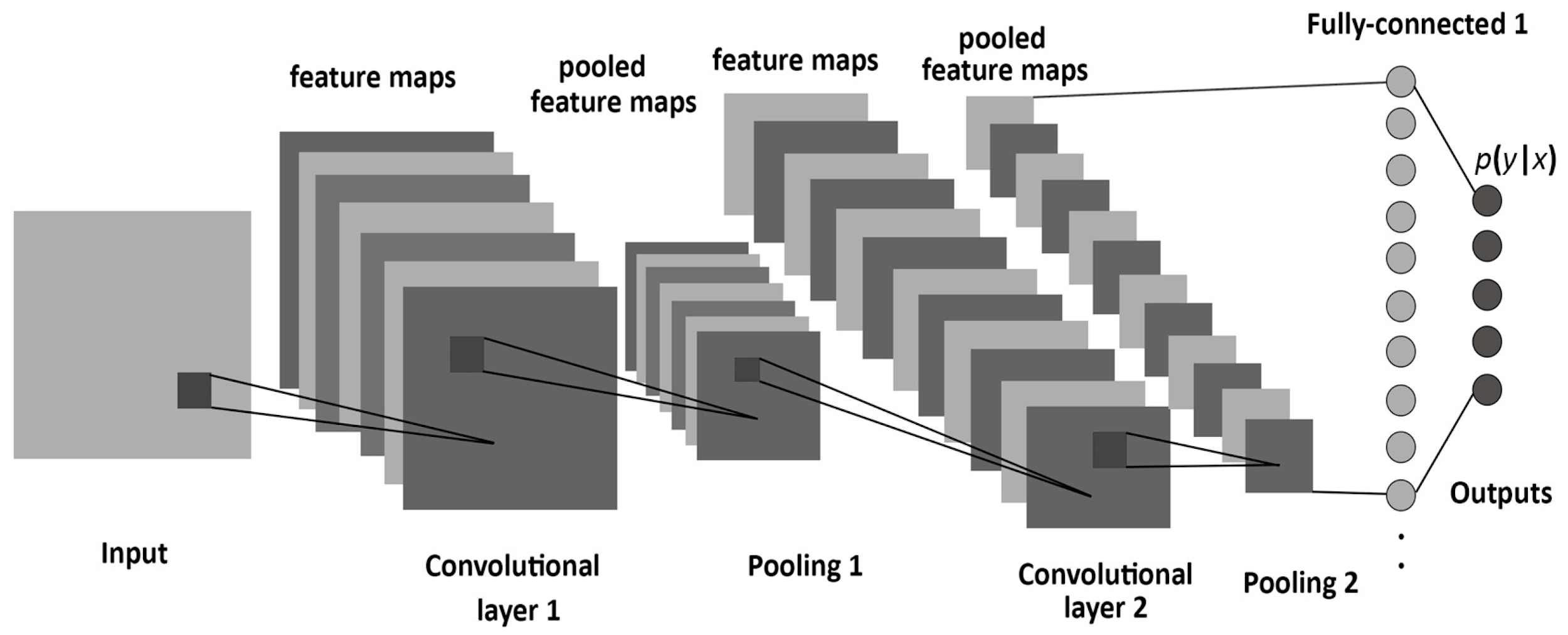
**2. Algorithms used:**

The key concepts implemented in this project are as follows:

**2.1 Convolutional Neural Network**

It is a class of deep neural networks used for analyzing visual images. A convolutional network has a structure formed of an input layer, some hidden layers, and an output layer. The hidden layers of a convolutional neural network usually include a convolutional layer, pooling layer, layers involving activation functions, normalization layer, and fully-connected layer.

* **Convolutional layer:** It is a building block of the convolutional network. Each convolutional layer processes the data only for its *receptive field*. Filters are used to detect the features out of the image at various levels.
* **Pooling layer:**  The function of pooling layer is to reduce the number of parameters by extracting the important ones and eventually to reduce the time of computation.
* **Fully-connected layer:** In a fully connected layer, each neuron in a particular layer is connected to each of the next layers. From this layer, activation is applied to classify the images.

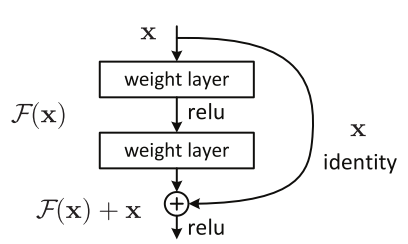


**Image 1: Convolutional Neural network**

[**https://towardsdatascience.com/how-to-teach-a-computer-to-see-with-convolutional-neural-networks**](https://towardsdatascience.com/how-to-teach-a-computer-to-see-with-convolutional-neural-networks)

**2.2 Residual Networks**

The problem with deep neural networks is that as we go deeper and deeper, the accuracy attains saturation or continues to decrement. This happens because due to the back-propagation of gradient descent in initial layers, some of the activations become infinitely small. The training error of deeper networks must not be greater than the shallow ones. This problem is resolved by introducing residual block in the network, where we skip one or two layers. Residual networks are based on the idea of ‘identity shortcut connection’.



**Image 2: Residual Block**

**https://towardsdatascience.com/an-overview-of-resnet-and-its-variants**

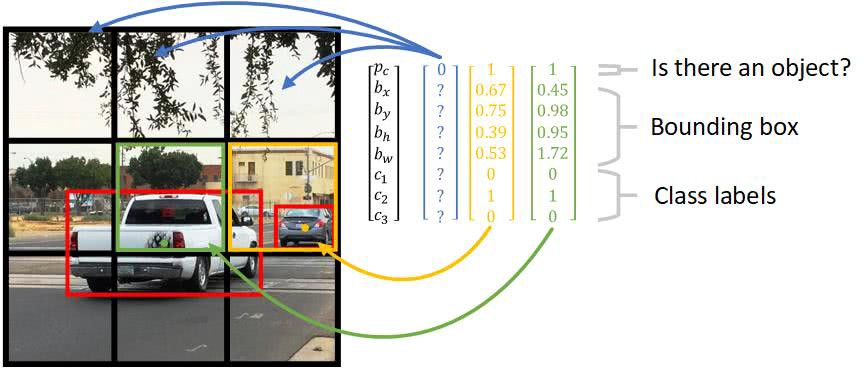
**2.3 YOLO Algorithm**

YOLO(You Only Look Once) algorithm is a modification applied to the convolutional neural networks to detect the objects in the image along with the classification. This algorithm is called ‘only look once’ because, in this, we have to forward propagate through the neural network only once to detect the objects.

The image is divided into grids and for each grid, the bounding boxes are calculated along with the associated probability scores.

The model will predict multiple bounding boxes, so non-max suppression is used to discard the ones with low confidence scores.

It is a fast algorithm as compared to other object detection algorithms.

**Image 3: YOLO**

[**https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras**](https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d)

**2.4 Adam Optimisation Algorithm**

Adam optimization is a learning rate optimization developed to train deep neural networks. It allows the gradient descent to proceed in the right direction, thus leading to faster convergence of gradient descent. It works on stochastic gradient descent, and even on mini-batch gradient descent, and combines the two algorithms of momentum and root mean square propagation(RMS prop).

In momentum, we maintain the exponentially weighted average of past gradients.

*VdW = ß1 x VdW + (1- ß1) x dW*

*Vdb = ß1 x Vdb + (1 – ß1) x db*

In RMS prop, we maintain the exponentially weighted average of squares of past gradients.

*SdW = ß2 x SdW + (1- ß2) x dW2*

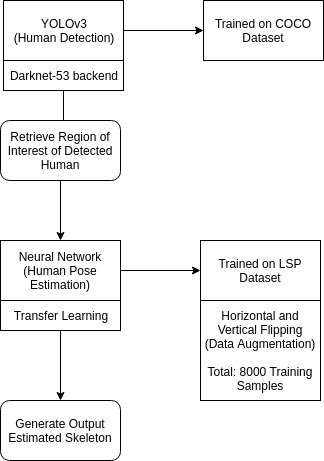
*Sdb = ß2 x Sdb + (1 – ß2) x db2*

The parameters W and b are then updated as follows:

*W = W – learning rate x (VdW/ sqrt(SdW + ε))*

*b = b – learning rate x (Vdb / sqrt(Sdb+ ε))*

**3. Methodology**The detailed proposed framework is given in this section. We divide the task into two parts as follows:



**Image 4: Project Framework**

**draw.io**

**3.1. Human Detection**

Given a still image, we have to detect humans with an acceptable confidence score. The output for this task is a set of coordinates corresponding to a detection box for each human detected in the input image.

**a) YOLOv3 Algorithm with DarkNet as backend:**

The network is trained with DarkNet-53 backend. In this version of YOLO, predictions are made thrice at three different locations in the network architecture.

The algorithm has been trained on COCO dataset.

**b) Anchors**

These are the bounding boxes used to detect the object. Our choice of anchors are: (10, 13), (26, 30), (33, 23), (50, 61), (62, 45), (99, 119), (116, 90), (156, 198), (373, 326)

**c) Leaky ReLU**

The activation function used is Leaky ReLU. We have taken *alpha* as 0.1 for the model.

**3.2. Human Pose Estimation**

After the completion of the first step, we retrieve the regions of interest where a human is detected and mark certain points on the human estimating its posture. We finally generate a skeleton for the detected human.

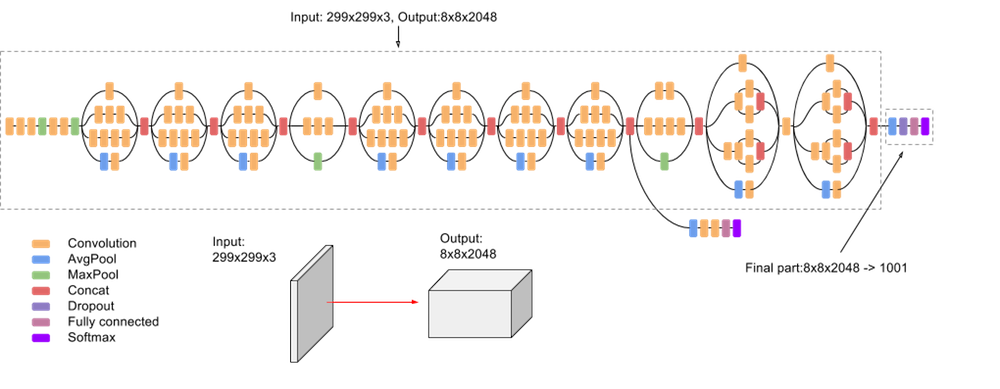
**a) Training Data**

We have used the LSP dataset which comprises of 2000 labelled images. Using horizontal and vertical flipping as data augmentation techniques, we increased our dataset size to 8000. Each image is resized to 150x150 pixels for training. For each image, we have labels for 14 joints on the body - right ankle, right knee, right hip, left hip, left knee, left ankle, right wrist, right elbow, right shoulder, left shoulder, left elbow, left wrist, neck, head top. Corresponding to each joint, we use the x and y coordinates of their location for training.

**b) Transfer Learning**

Here, the knowledge of an already trained machine learning model is applied to a different but related problem. we basically try to exploit what has been learned in one task to improve generalization in another.

**c) InceptionV3**

****

**Image 5: InceptionV3**

**https://cloud.google.com/tpu/docs/inception-v3-advanced**

We used the InceptionV3 network with initial layers pre-trained on ImageNet. The final part of the architecture was removed and modified as stated below.

This architecture is not able to predict satisfactory skeleton points for the test image. Most of the predicted joint locations are falling out of the image dimensions.

Thus, we moved on to test a new architecture.

Modified Layers:

|  |  |
| --- | --- |
| **LAYER** | **LAYER SIZE** |
| Flatten | 18432 |
| Dropout | 18432 |
| Dense | 1024 |
| Batch Normalization | 1024 |
| ReLU Activation | 1024 |
| Dropout | 1024 |
| Dense | 512 |
| Batch Normalization | 512 |
| ReLU Activation | 512 |
| Dense | 2 |

**Table 1: Modified Layers for InceptionV3**

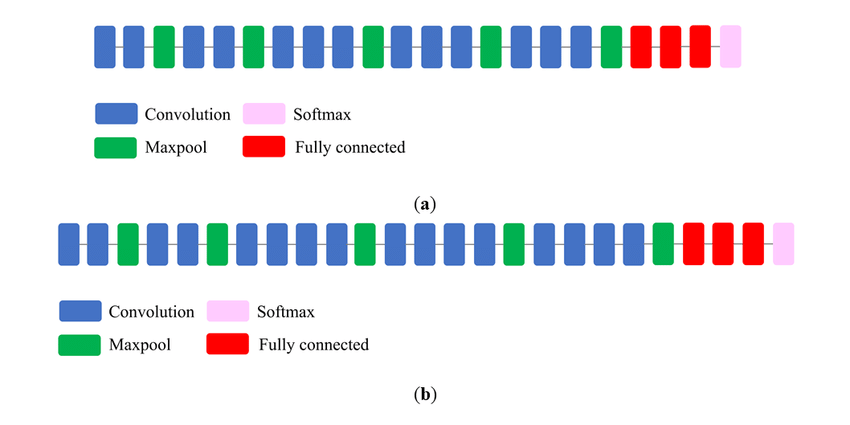
Parameters:

Total number of parameters: 41,210,146

Trainable parameters: 20,454,914

Non-trainable parameters: 20,755,232

**d) VGG19**

****

**Image 6: VGG19 https://www.researchgate.net/figure/llustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means\_fig2\_325137356**

We have used the VGG19 architecture with initial layers pre-trained on ImageNet dataset without the softmax and fully-connected layers, and modified the final part of the network as stated below:

**Variant 1 (23 Layers):**

Modified Layers:

|  |  |
| --- | --- |
| **LAYER** | **LAYER SIZE** |
| Flatten | 8192 |
| Dropout | 8192 |
| Dense | 800 |
| Batch Normalization | 800 |
| ReLU Activation | 800 |
| Dropout | 800 |
| Dense | 500 |
| Batch Normalization | 500 |
| ReLU Activation | 500 |
| Dropout | 500 |
| Dense | 200 |
| Batch Normalization | 200 |
| ReLU Activation | 200 |
| Dense | 2 |

**Table 2: Modified Layers for VGG19 Variant 1**

Parameters:

Total number of parameters: 27,085,886

Trainable parameters: 16,497,734

Non-trainable parameters: 10,588,152

**Variant 2 (21 Layers):**

Modified Layers:

|  |  |
| --- | --- |
| **LAYER** | **LAYER SIZE** |
| Flatten | 8192 |
| Dense | 1024 |
| Batch Normalization | 1024 |
| ReLU Activation | 1024 |
| Dropout | 1024 |
| Dense | 512 |
| Batch Normalization | 512 |
| ReLU Activation | 512 |
| Dense | 2 |

**Table 3: Modified Layers for VGG19 Variant 2**

Parameters:

Total number of parameters: 28,945,986

Trainable parameters: 18,357,762

Non-trainable parameters: 10,588,224

**e) Adam Optimization**

Adam optimization is used with hyperparameters:

Learning rate: 0.1

Beta1: 0.9

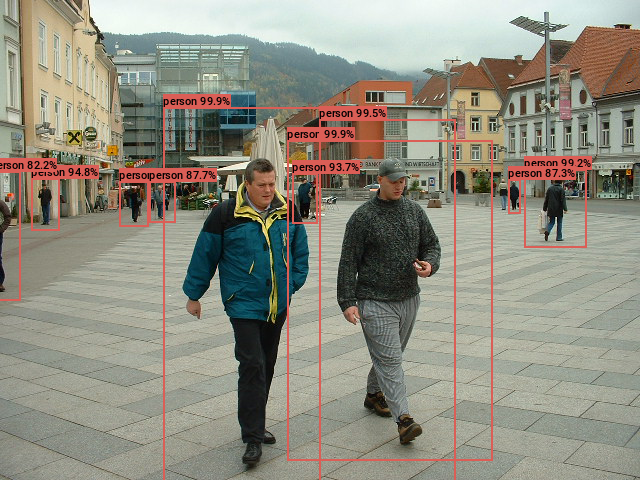
Beta2: 0.999

Epsilon: 0

**4. Results and Comparison**

We will be comparing visual results on images from INRIA dataset, and performances on LSP dataset.

**Output of YOLOv3:**



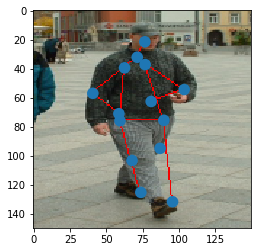
**Image 7: Human Detection**

**Pose Estimation with VGG19 Variant 1:**

****

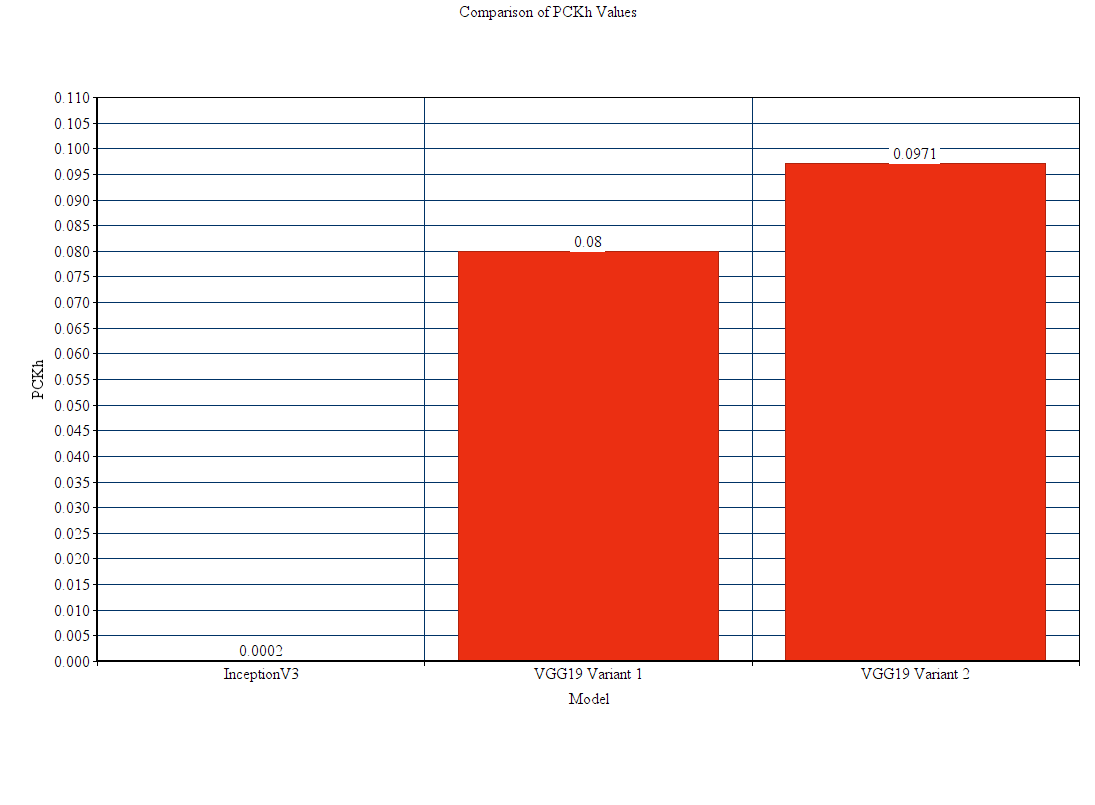
**Image 8: Pose Estimation VGG19 Variant 1**

**Pose Estimation with VGG19 Variant 2:**



**Image 9: Pose Estimation VGG19 Variant 2**

**Comparison of PCKh Scores:**

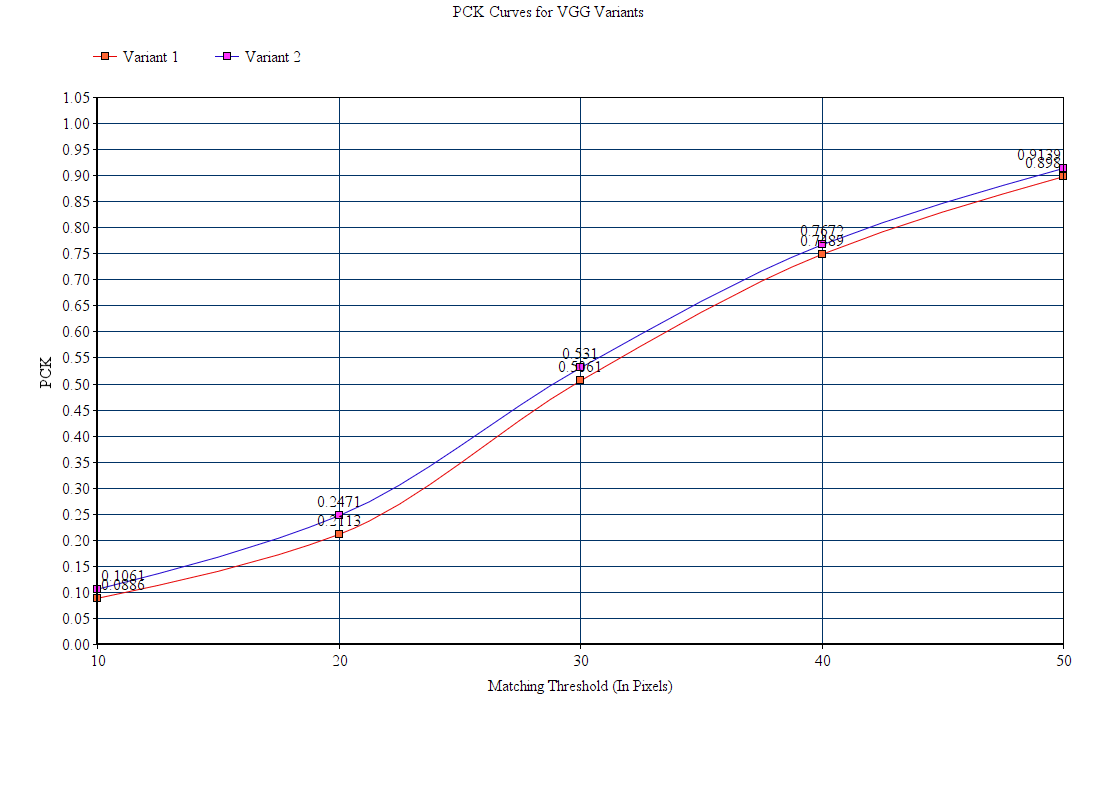
****

**Graph 1: Comparison of PCKh Scores**

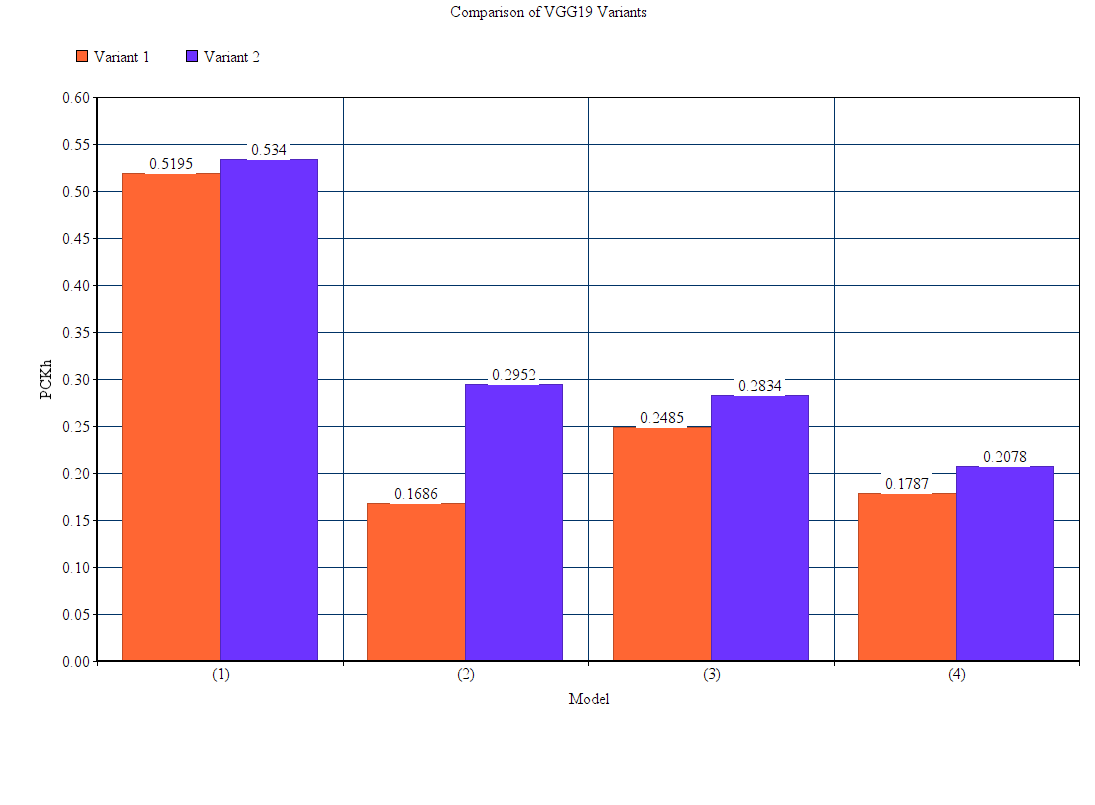
**Comparison of VGG19 Variants:**

|  |  |  |
| --- | --- | --- |
| **Matching Threshold** | **PCK Score for Variant 1** | **PCK Score for Variant 2** |
| 10 | 0.0886 | 0.1061 |
| 20 | 0.2113 | 0.2471 |
| 30 | 0.5061 | 0.5310 |
| 40 | 0.7489 | 0.7672 |
| 50 | 0.8980 | 0.9139 |

**Table 4: PCK Scores for VGG19 Variants**

****

**Graph 2: PCK Curves for VGG19 Variants**



**Graph 3: Comparison of VGG Variants**

|  |  |
| --- | --- |
| **X-Axis Label** | **Meaning** |
| (1) | Detection of head top and neck (face) |
| (2) | Detection of the head top, neck, and left and right shoulder joints |
| (3) | Detection of the head top, neck, and left and right hip joints |
| (4) | Detection of the head top, neck, shoulder and hip joints (face and torso) |

**Table 5: Key for Graph 3**

**5. Conclusion**

YOLOv3 is an efficient object detection algorithm. After detecting humans in an image, we were able to produce decent pose estimations using deep learning methods. Inception Net Version 3 gave a very poor performance when compared to VGG19 Variants. Reducing the number of layers and increasing their size resulted in improvement of performance.

**6. Bibliography**

a) Adam Optimization:

<https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

b) YOLO:

[https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm](https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0)

c) Residual Networks:

[https://towardsdatascience.com/an-overview-of-resnet-and-its-variants](https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035)

d) YOLOv3:

<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>

<https://www.kaggle.com/aruchomu/yolo-v3-object-detection-in-tensorflow>

e) Object Detection:

<https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/>

f) Dropout:

<https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>

g) Metric for Pose Estimator Performance:

<http://sergioescalera.com/wp-content/uploads/2015/08/COST-ACTION-TRAINING-SCHOOL-DATASETS-CV-TUTORIAL.pdf>

h) Transfer Learning:

<https://towardsdatascience.com/transfer-learning-946518f95666>

<https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8>

i) Deep Learning:

<https://www.deeplearning.ai/>

j) Network Architectures:

<https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/>

<https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202>

**Datasets:**

<http://cocodataset.org/#home>

<http://sam.johnson.io/research/lsp.html>