REAL TIME YOGA POSE DETECTION

Sanskar Ghyar (200873)
Happy Pachori (200399)

Supervised by Dr. Tushar Sandhan

Department of Electrical Engineering
IIT Kanpur
Jan 2023 - Apr 2023

Introduction

Nowadays, yoga has gained worldwide attention because of increasing levels of stress in the modern way of life, and there are many ways or resources to learn yoga. In the modern world we suggest having a Tutor around while doing yoga. It does not always happen to have a Tutor or join yoga classes these days. So an AI-based program helps in the estimation and identification of yoga poses.
Activity recognition is the process of continuously monitoring a person’s activity and movement. Human posture recognition can be utilized to assemble a self-guidance practice framework that permits individuals to accurately rehearse yoga postures without getting help from anyone else. With the use of deep learning algorithms, we propose an approach for the efficient detection and recognition of various yoga poses.

Timeline

STAGE I

Basic idea of the problem statement
Dataset collection

STAGE II

Strong baseline of the problem statement
Trained different models

STAGE III

Own contribution :

Add angle feature vector
Made keypoints feature vector robust
Realtime pose classifier

Warrior II

Warrior I

Tree

Standing big toe

Downward facing dog

Plank

Cobra

Cat & cow

Child

Dataset

Three different types of dataset were used from different sources:-

- yoga_poses (Tensorflow) : 1500 images of 5 poses
- Kaggle dataset : 1500 images of 5 poses
- yoga_82 (Google) : 2100 images for 9 poses

The finally chosen dataset consists of 9 different poses chosen from the yoga_82 dataset.

Methodology

Extract key points :-

To extract key points of body skeletal for the pose, 2 models were used:-

1. MoveNet Thunder
2. Mediaipipe

Split dataset for training and testing

Preprocess data and create labels and features

Train classification models :-

We used 3 different types of model:-

1. Artificial Neural Networks (ANN)
2. Recurrent Neural Networks (RNN)
3. k-Nearest Neighbour (KNN)

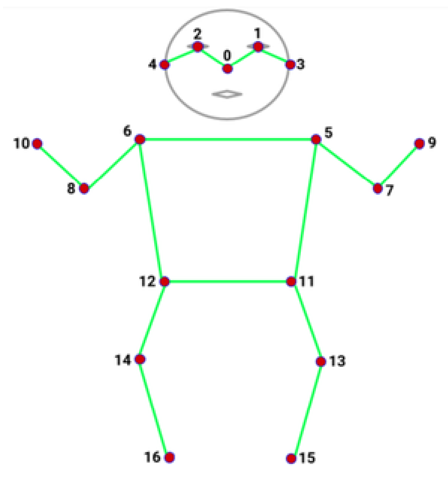
After this we saved the weights of best model, made predictions based on our best model and evaluate using confusion matrix and accuracy.

Finally we test our model in real time.

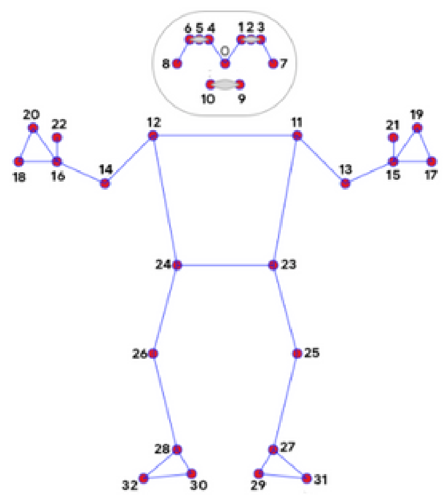
Key points extraction

In the context of yoga, key point extraction can be used to track the positions of key joints in the body during a yoga practice. Two different models were used to extract key points from the image of yoga pose: MoveNet Thunder and Mediapipe.

Movenet Thunder - It is a deep learning-based pose estimation model with very high accuracy which tracks 17 keypoints of body which consist x & y coordinates and corresponding confidence score for each keypoint



Mediapipe - It is similar model to Movenet but with a very low computational cost. It track 33 keypoints which consist x, y & z coordinates and corresponding confidence score for each keypoint



Dataset: yoga_poses

MoveNet Thunder

train_accuracy = 100%
val_accuracy = 99.02%
time required to extract keypoints = 70min

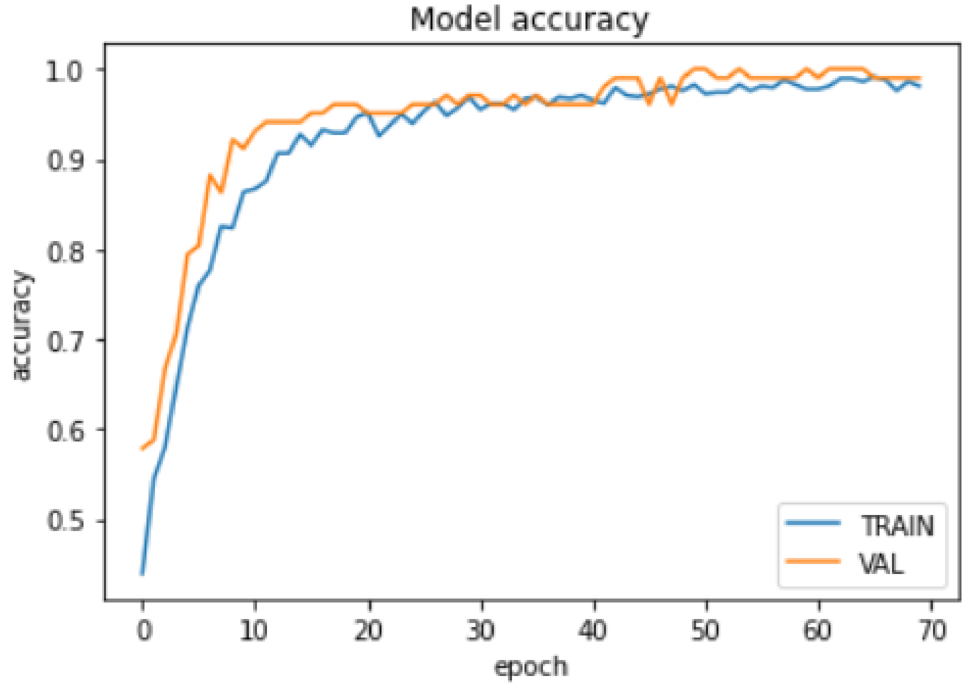
Mediapipe

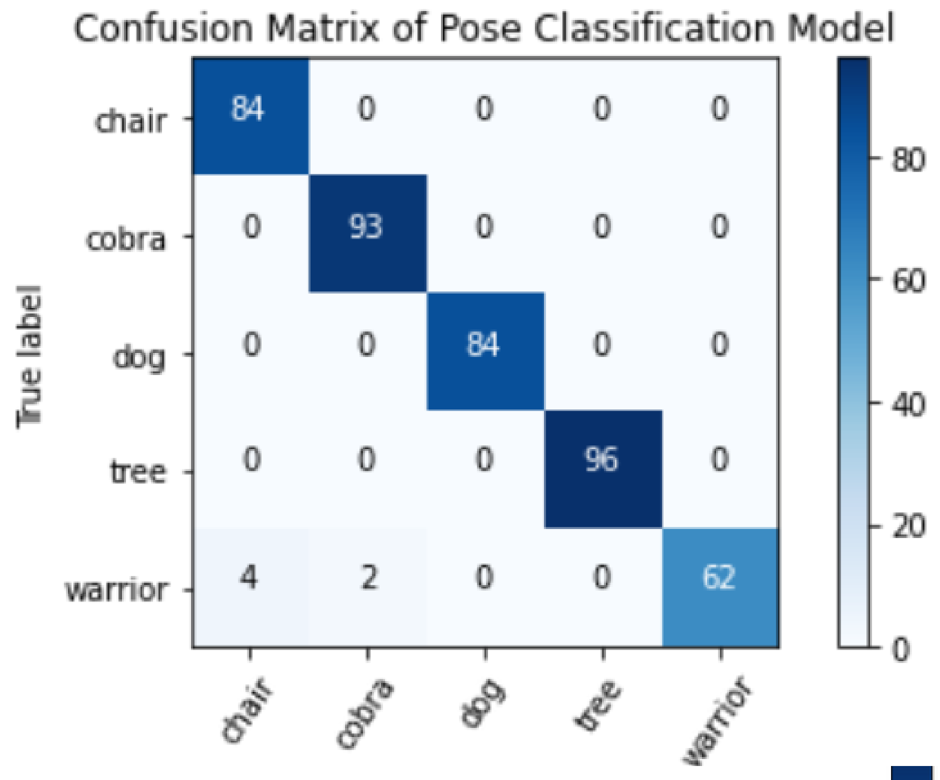
train_accuracy = 100%
val_accuracy = 97.27%
time required to extract keypoints = 6.5min

Since the time required for Mediapipe model is approximately 1/10th of MoveNet and accuracy is not very differentiable, hence Mediapipe will be used for further key points extraction and also our major aim is to detect key points faster with a reasonable accuracy.

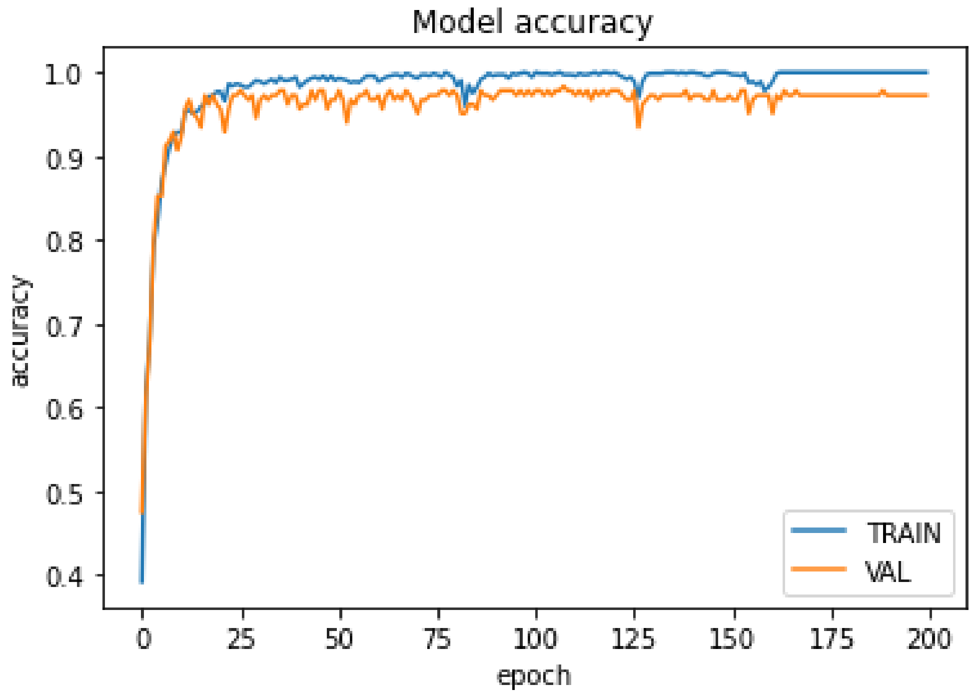
Results from which we concluded our choice for best model to detect key points

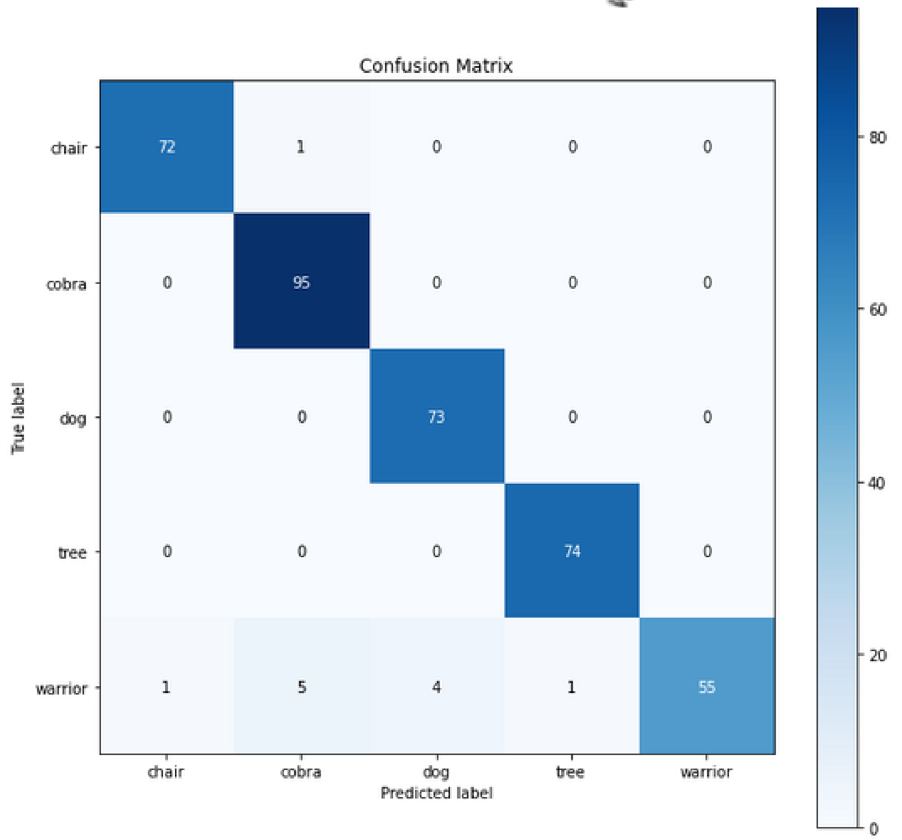
MoveNet Thunder





Mediapipe





Keypoints of Mediapipe:

0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Preprocess Data and create labels and features

Dataset was cleaned for the images having more than one person and doodle images After extraction of various key points from the images and adding label according to various poses we compile our results into a single .csv file. Then dataset was split in ratio of 4:1 for training and testing respectively for each individual classes so that dataset is equally split for each label.

Types of feature vectors -

1. 33 key points (x, y & score of each key points)
2. Angles of main joints
3. Reduced key points
4. Angle of main joints + key points
5. Relative and normalised keypoints

Feature vector

Reduced keypoints(23):-
Nose is the only important keypoint in face, hence other keypoints were dropped.
8 angles of main joints:-
Left & right elbow angle, left & right shoulder angle, left & right hip angle and left & right knee angle
Angle of joints = $\tan^{-1}\frac{y3-y2}{x3-x2}-\tan^{-1}\frac{y2-y1}{x2-x1}$

Relative & Normalised feature vector:-
Pose centre of body was considered as midpoint of hip:-
pose_centre = 0.5*(left_hip+right_hip)
Then relative position was taken for each keypoints:-
relative_keypoints = keypoints - pose_centre
normalised_relative_keypoints = relative_keypoints / distance from pose_centre

Both 8 angle of main joints and relative & Normalized feature vector are independent of the distance of body from camera.

Classification models

We train different types of classification model on 33 key points feature vector (which gives the maximum information about the pose) and select best model and weights to train our features based on validation accuracy.

k-Nearest Neighbour (KNN)

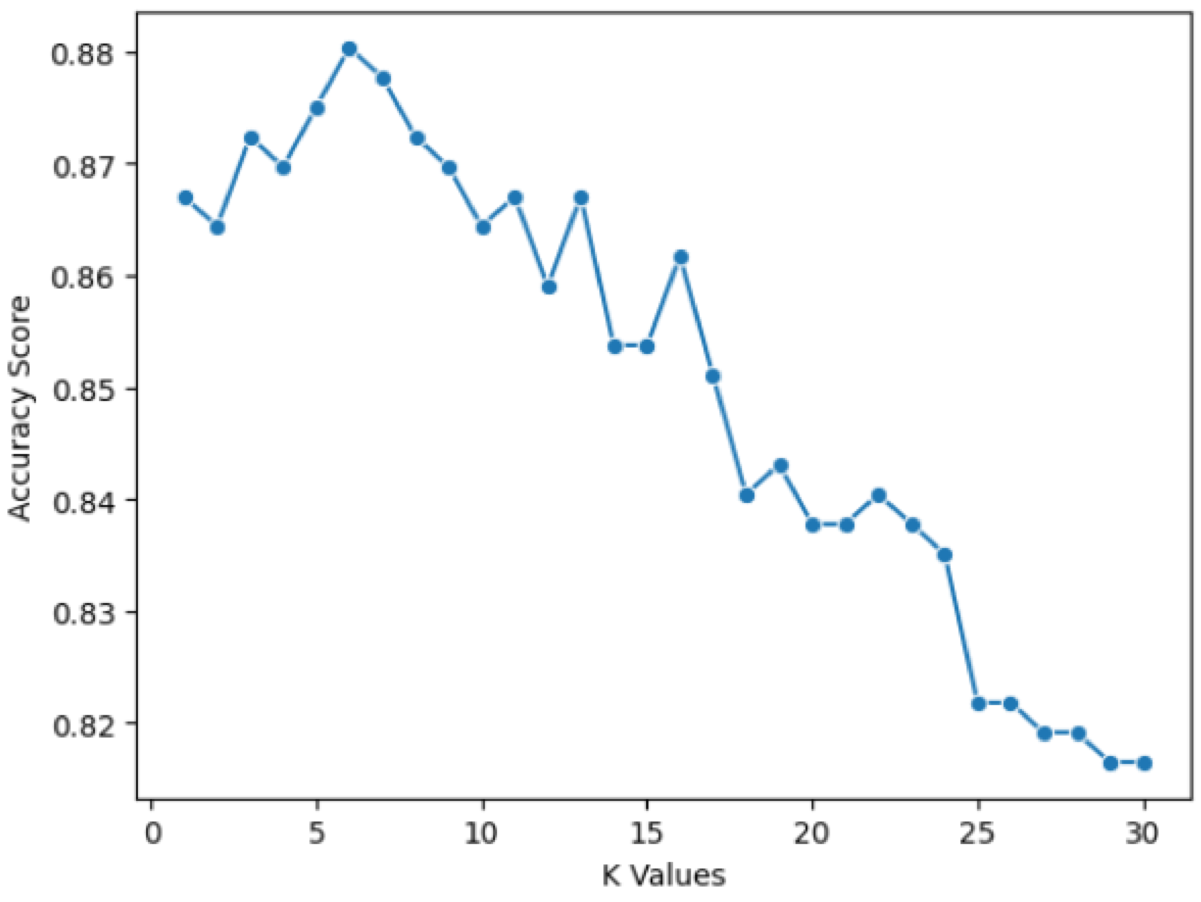
We are going to do multi-class classification using K Nearest Neighbours. KNN is a super simple algorithm, which assumes that similar things are in close proximity of each other. So if a datapoint is near to another datapoint, it assumes that they both belong to similar classes.

KNN model was trained for k_neighbours from 1 to 30, and at k_neighbours = 6, accuracy is maximum. As these are tied for the best score, it is advisable to use a smaller value for k_neighbours. This is because when using higher values of k, the model will use more data points that are further away from the original.

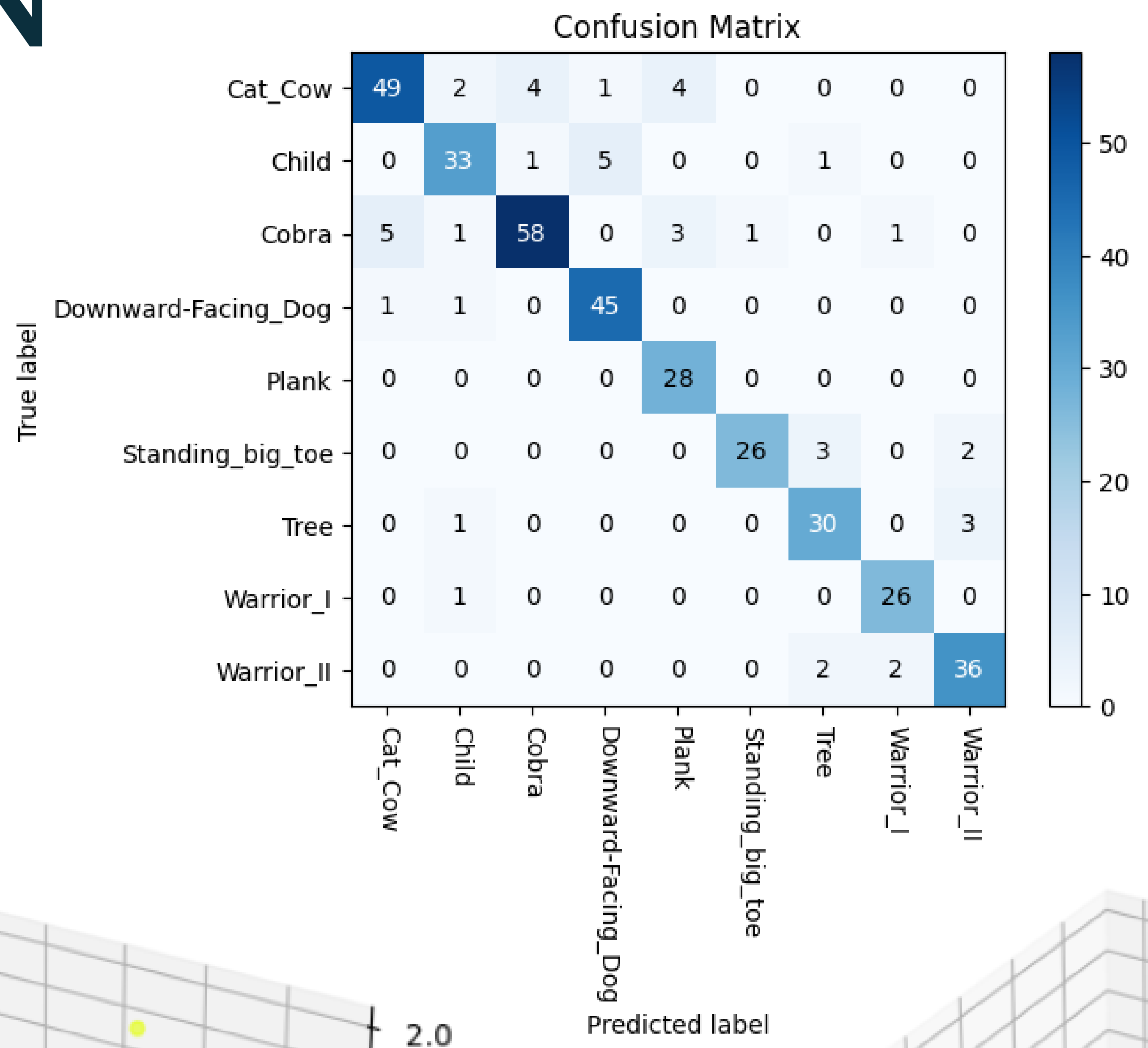
train_accuracy = 90.41%

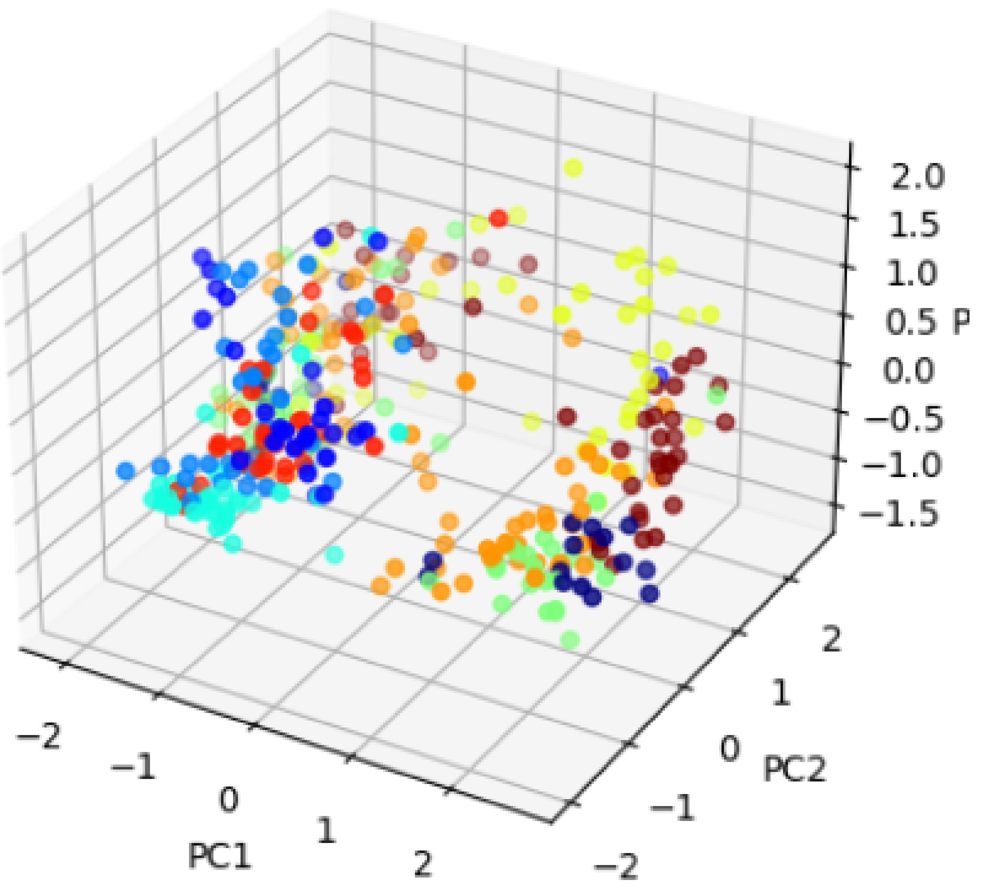
test_accuracy = 88.03%

KNN

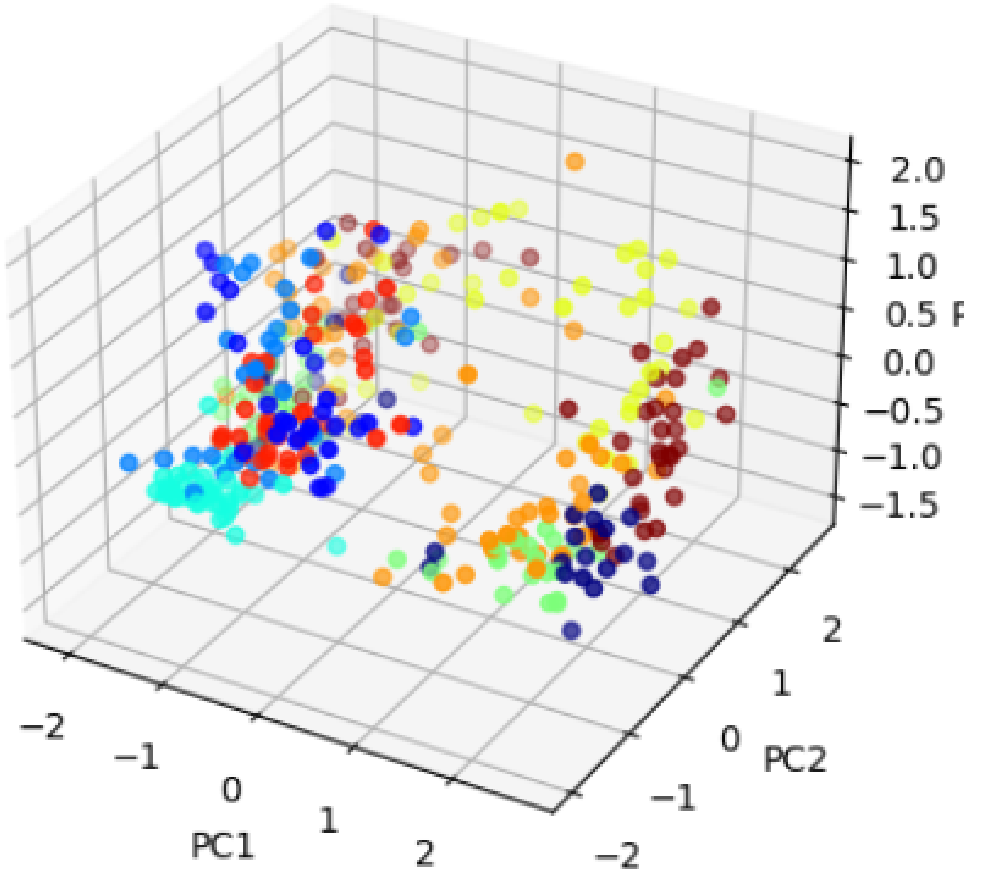


Accuracy score of KNN for different number of nearest neighbourt





Original clusters



Predicted clusters

ANN

Input layer for ANN model in flatten array of size 99 (x,y and score each for 33 keypoints)
The activation function used in this layer is ReLU, which is a piece-wise linear function that gives an output equal to 0 when the input provided is less than 0, else it gives as output the given input [31,32].
The activation function of ReLU:
ReLU(x) = max(0, x) ; where x^R
The final layer used is a Dense layer that uses Softmax as the activation function, which assigns probabilities of different poses based on the current given input. The mathematical equation of Softmax activation function is presented in Equation.

$$\sigma(\underline{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

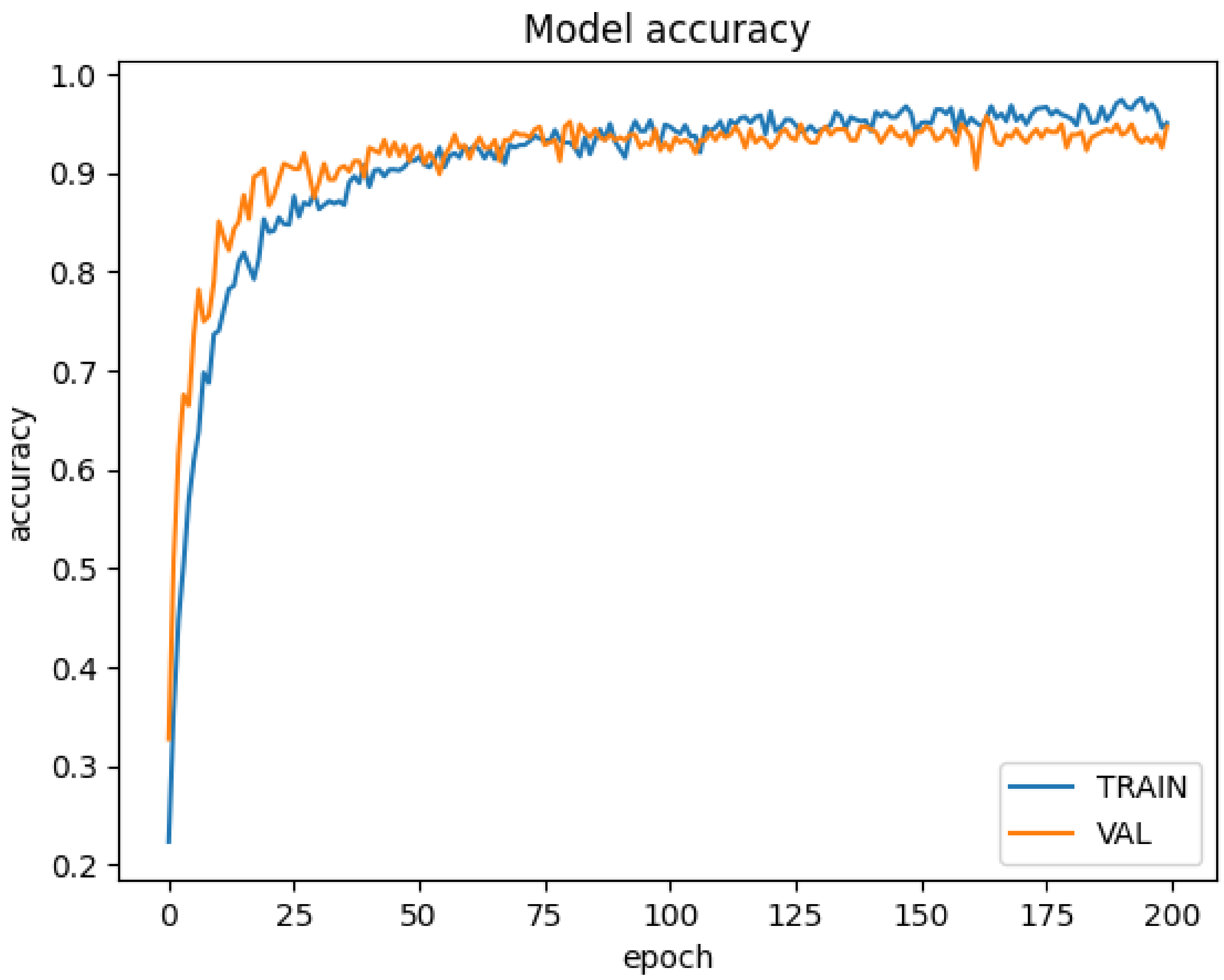
The Adam optimizer is employed; this optimizer helps the model to fast converge by the addition of momentum term and scaling term.

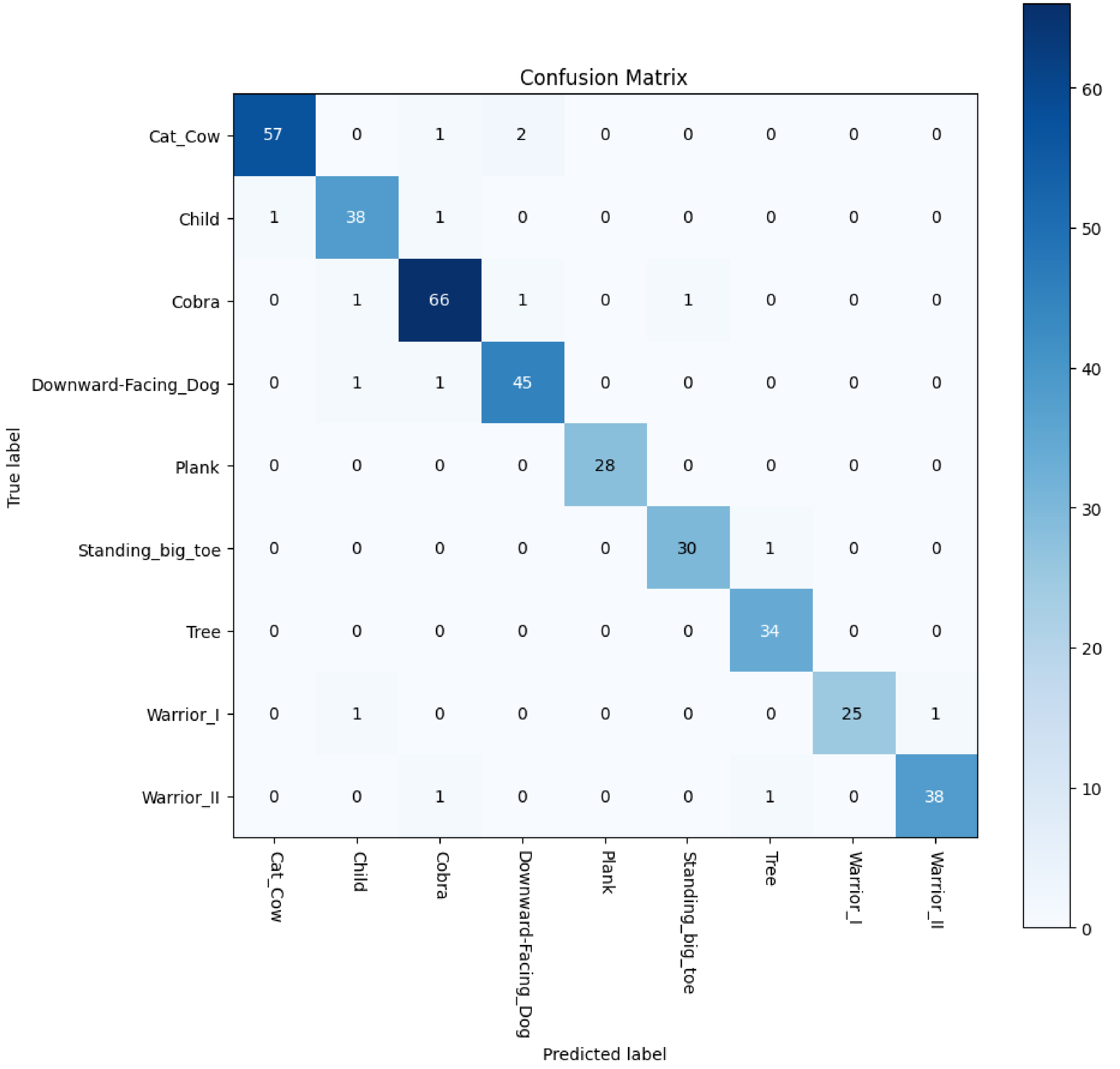
The loss function used is categorical cross-entropy which is very popular for multiclass classification tasks. Equation depicts the mathematical equation used in categorical cross-entropy loss function.

$$E_{CC} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}(p_{ic}log(y_{ic}))$$

Layer (type)	Output Shape	Params
dense (Dense)	(None, 128)	12800
dense_1 (Dense)	(None, 256)	33024
dense_2 (Dense)	(None, 64)	16448
dense_3 (Dense)	(None, 9)	585
Total params = 62857	Trainable params = 62857	Non trainable params = 0

ANN





train_accuracy = 98.9%
val_accuracy = 95.7%

RNN

Input layer for RNN model is array of size (1,99) (x, y and score each for 33 keypoints)
Long Short-Term Memory (LSTM) unit is the portion that does the remembering.
The activation function used in this layer is ReLU, which is a piece-wise linear function that gives an output equal to 0 when the input provided is less than 0, else it gives as output the given input [31,32].
The activation function of ReLU:
ReLU(x) = max(0, x) ; where x^R
The final layer used is a Dense layer that uses Softmax as the activation function, which assigns probabilities of different poses based on the current given input.
The mathematical equation of Softmax activation function is presented in Equation.

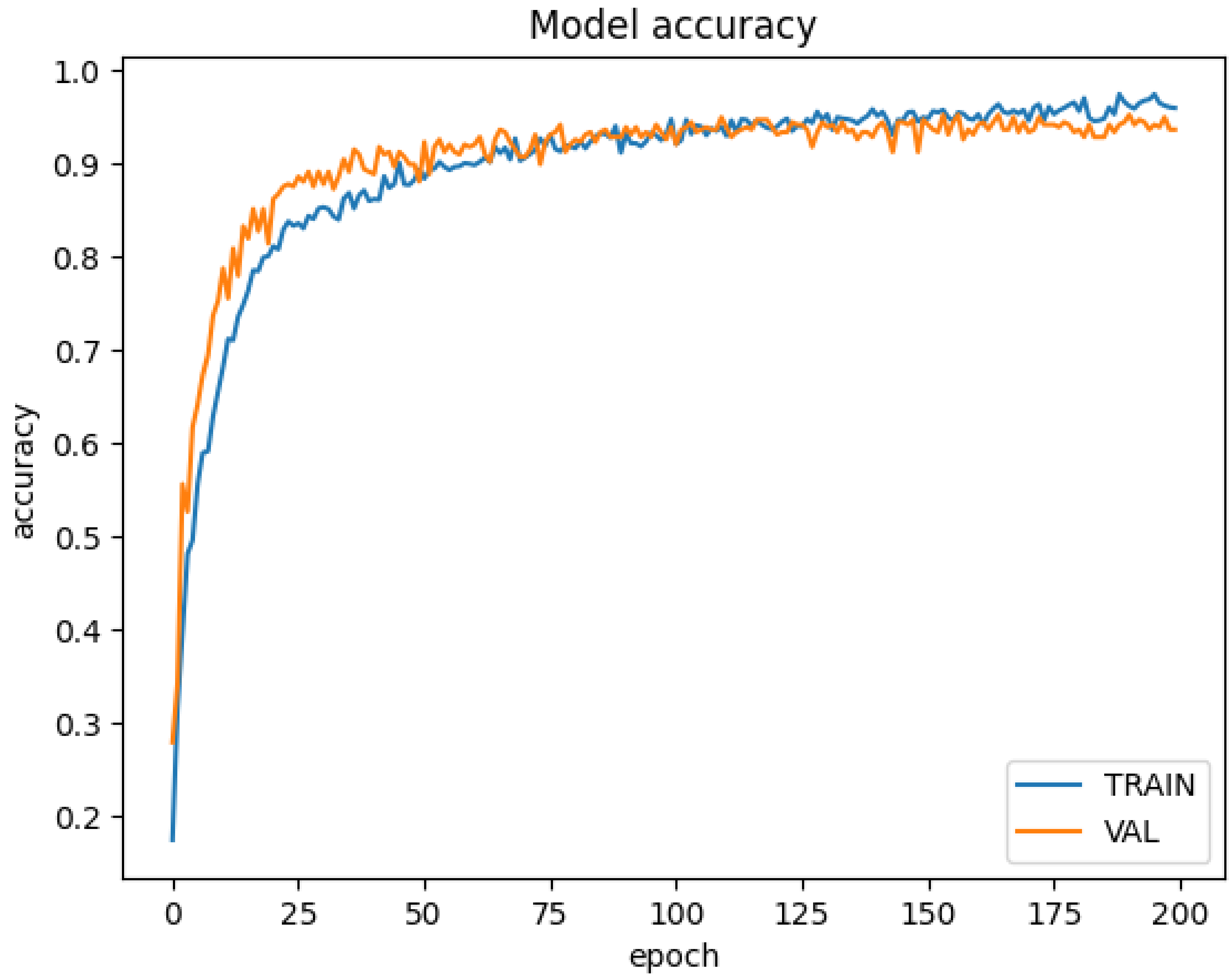
$$\sigma(\underline{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

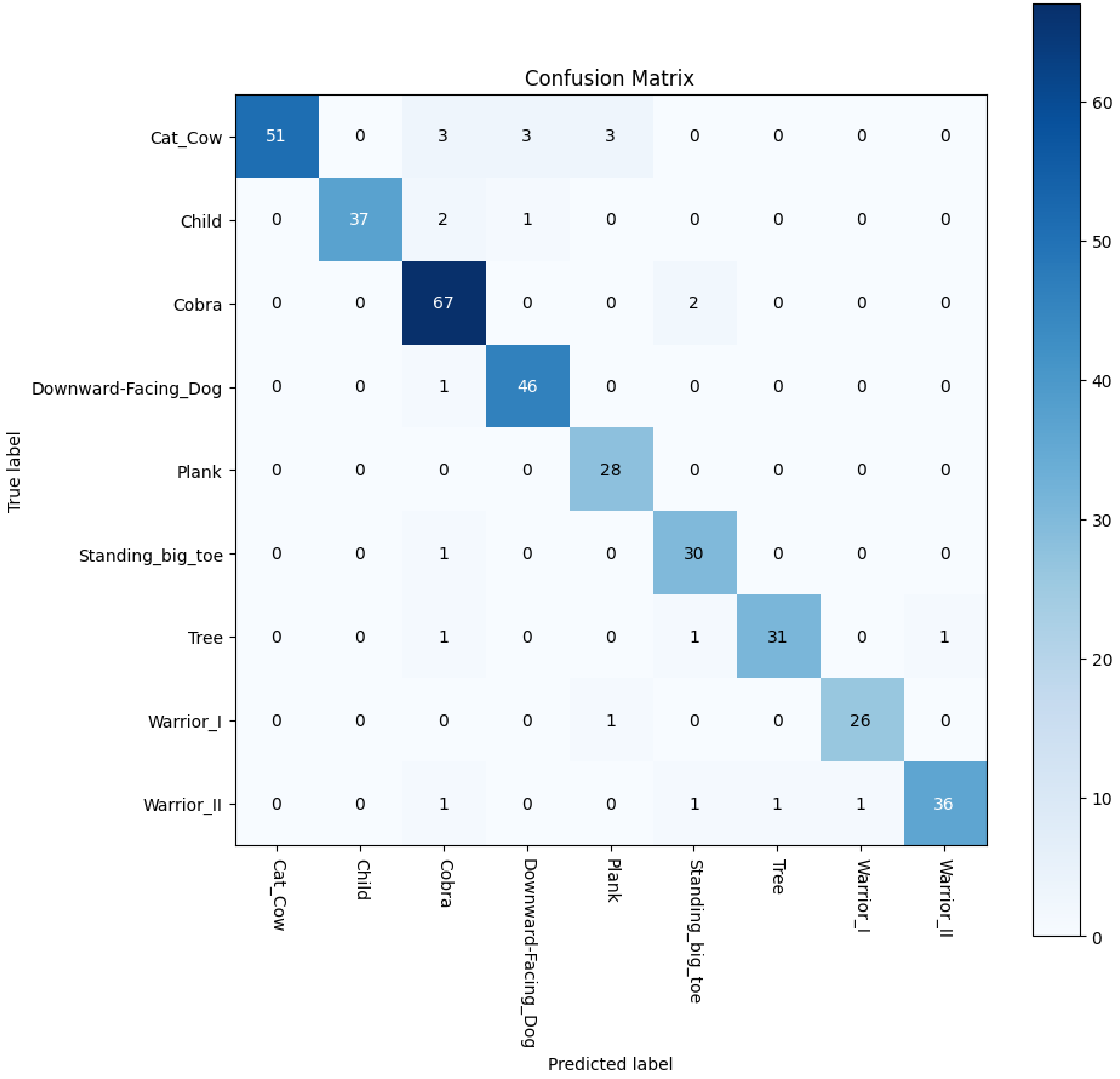
The Adam optimizer is employed; this optimizer helps the model to fast converge by the addition of momentum term and scaling term.
The loss function used is categorical cross-entropy which is very popular for multiclass classification tasks. Equation depicts the mathematical equation used in categorical cross-entropy loss function.

$$E_{CC} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}(p_{ic}log(y_{ic}))$$

Layer (type)	Output Shape	Params
lstm (LSTM)	(None, 1, 128)	116736
lstm_1 (LSTM)	(None, 256)	394240
dense (Dense)	(None, 64)	16448
dense_1 (Dense)	(None, 9)	585
Total params = 528009	Trainable params = 528009	Non trainable params = 0

RNN





train_accuracy = 98.53%
val_accuracy = 93.62%

Best Model

We used three different types of model for our problem: KNN, ANN and RNN.
Out of these three model ANN performs best in terms of accuracy with which it classifies the different classes.

KNN model has some limitation and performs best according to the limitations with which it can classify.

Our ANN and RNN model are exactly the same having one different layer in RNN which is LSTM layer, now adding this layer actually decreases our accuracy because LSTM layer is used to learns long-term dependencies between time steps in time series and sequence data. Here our poses used are independent of other poses. No pose is in the sequence to reach the other pose all poses are individual poses. So using LSTM layer or RNN model is going to be of no benefit and it will increase the complexity of the model hence reducing the accuracy.

So the model we are going to use for best results from different feature vector is ANN.

Results obtained on training ANN on different feature vectors

Feature vectors used	test_accuracy	f1_score	precision	recall
33 keypoints	95.74	95.75	95.90	95.74
Angles	92.28	92.38	92.74	92.28
Reduced (23) keypoints	93.88	93.84	94.06	93.88
Angles + 33 keypoints	92.02	92.13	92.49	92.02
33 Normalised keypoints	94.14	94.14	94.2	94.14

Conclusion based on results obtained

Conclusion about feature vectors

33 Keypoints feature vector and normalized & Relative feature vector almost give the same result. This means that midepipe model is extracting keypoints from the given dataset quiet effectively and normalizing, finding relative postion of keypoints will not provide much of a difference in the results.

Angle of major joints feature vector shows less accuracy because if the midepipe model is not able to extract information about let's say one point correctly then the definition of the pose according to angle will change abruptly and hence classify our poses incorrectly.

Angle +Keypoints feature vector confuses the model about the defination of pose according to angle which we discussed above and according to keypoints because of this reason the accuracy decreases.

Reduced keypoint feature vector is showing less accuracy because reducing keypoints is simply reducing the information about a certain pose.

Final Conclusion and Future Development

Many studies on posture estimation in humans have been conducted in recent years. Estimating human posture requires localising and constructing human body parts based on a known human body structure, which is different from other computer vision tasks. By including posture assessment in the process, exercises can be made safer and more effective. We contend that yoga can be made more generally accepted while also ensuring that it is practised correctly through a self-learning approach. Deep learning approaches are interesting because of all the excellent research being done in this field. On the basis of Mediapipe data, all nine yoga poses were classified using ANN and LSTM models, in this ANN is seen to be a very successful method. It is also possible to use SVMs, but SVMs do not work with large datasets. This is less efficient than the models we used in this project. Therefore, this project will help people to perform yoga posture accurately and effectively by tracking and estimating the pose they are doing.

THANKS

FOR YOUR ATTENTION