Contactless Dynamic Gesture Based Interaction Systems

Mithesh Ramachandran

B Tech Data Science

SVKM'S NMIMS Deemed to be UNIVERSITY
MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Agenda

Part 1	A New methodology for efficient 2-D Hand Pose Estimation
Part 2	A New methodology for Dynamic Gesture Recognition
Part 3	Reproducible research and contributing to the community

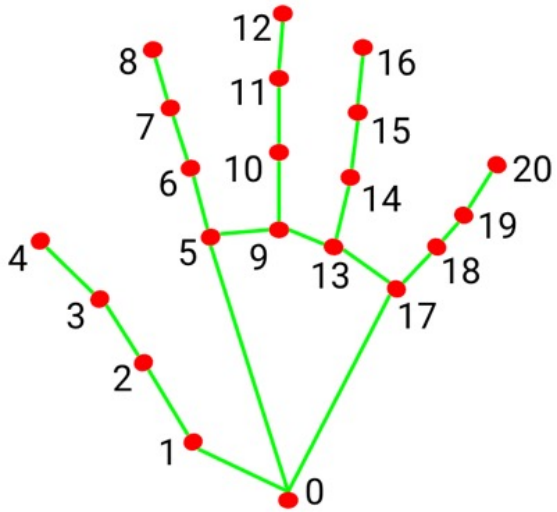
Part 1 **A New methodology for efficient 2-D Hand Pose Estimation**

Part 2 A New methodology for Dynamic Gesture Recognition

Part 3 Reproducible research and contributing to the community

What is a hand-pose?

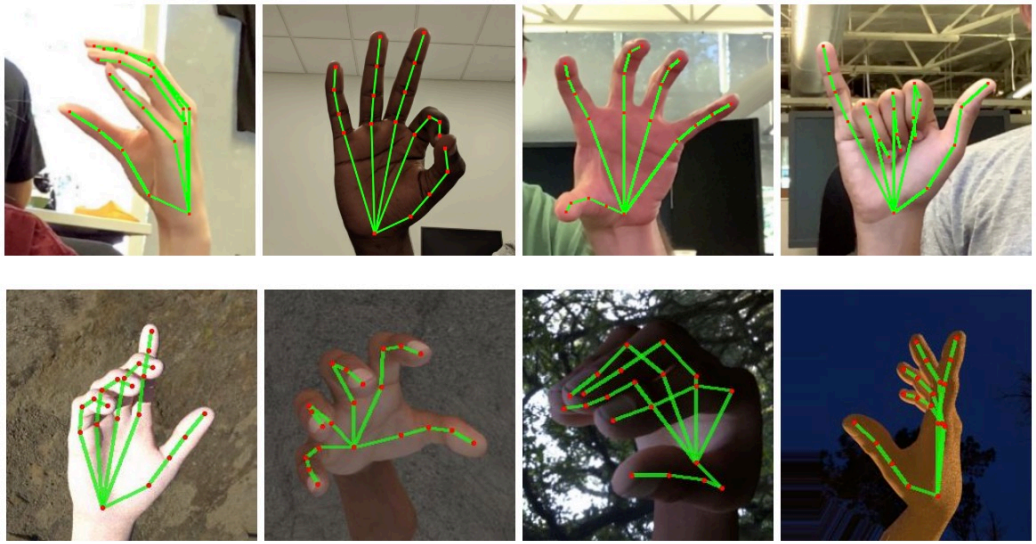
Estimating hand landmarks and joints with an image of the hand in different orientations



What is a hand-pose?

Estimating hand landmarks and joints with an image of the hand in different orientations

Used in Gesture Recognition, VR and MR, 3-D Reconstruction.



Motivation

Traditional methods of Hand Pose

Use a 3-D "Depth" image or a stereo image

Bulky equipment

Very expensive

Intel Realsense

Traditional methods of Hand Pose

Use a 3-D "Depth" image or a stereo image

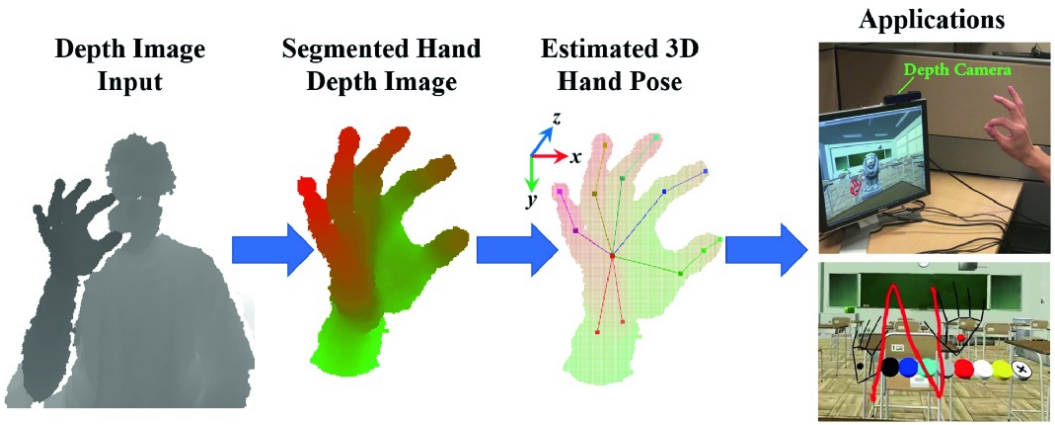
Bulky equipment

Very expensive

More channels = Higher image dimensions

Datasets are larger in size but fewer in samples

High processing power is required



Traditional methods of Hand Pose	2-D Hand Pose Estimation
Use a 3-D "Depth" image or a stereo image	Use RGB images
Bulky equipment	RGB cameras are small
Very expensive	Less expensive
More channels = Higher image dimensions	Fewer channels = Lesser image dimensions
Datasets are larger in size but fewer in samples	Datasets are larger in samples but smaller in size
High processing power is required	Less processing power is required

2-D Hand Pose Estimation

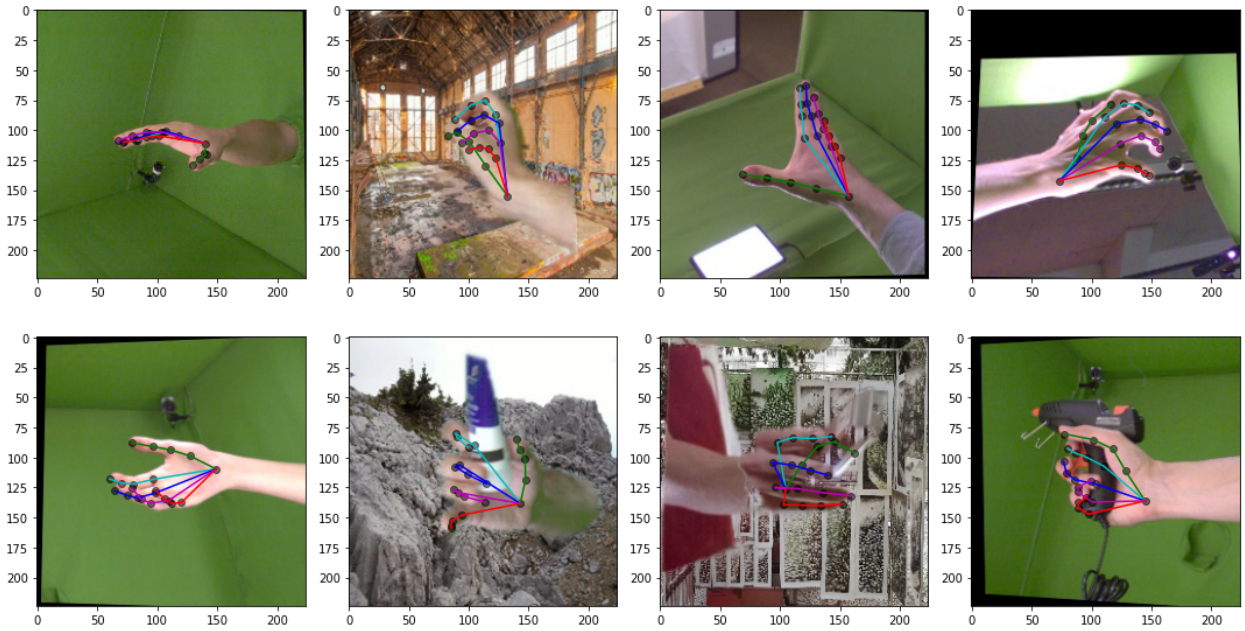
FreiHAND Dataset

32560 RGB images

Augmented background adds to 130240 images

5.3 GB in size

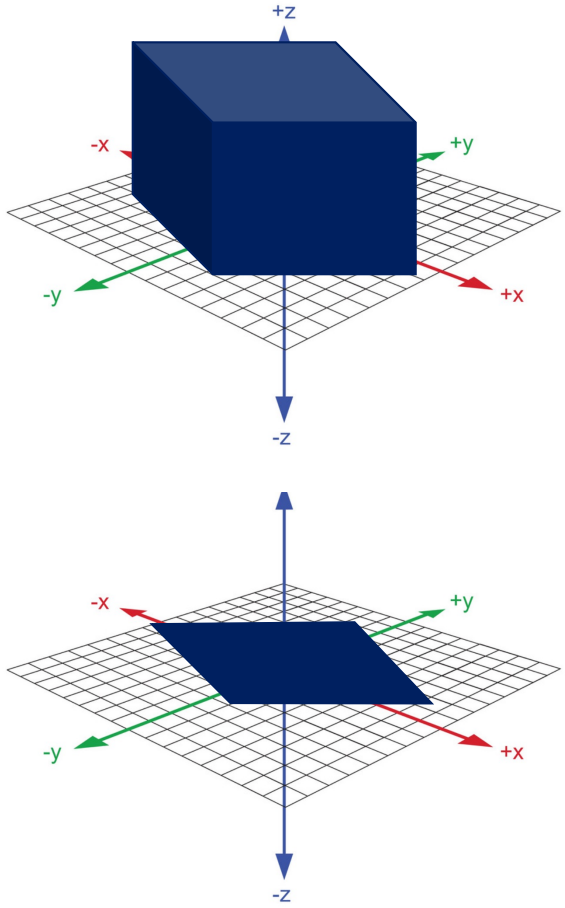
Each image is 224x224x3



Methodology

Step 1: Project the 3D Coordinates onto the 2D Plane.

$$\begin{matrix} g * f & s * f & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} X \\ Y \\ Z \end{matrix} = \begin{matrix} f * \frac{X}{Z} + x_0 \\ f * \frac{Y}{Z} + y_0 \\ 1 \end{matrix}$$



Methodology

Step 1: Project the 3D Coordinates onto the 2D Plane.

Step 2: Generate Heatmaps

One point becomes a region of points

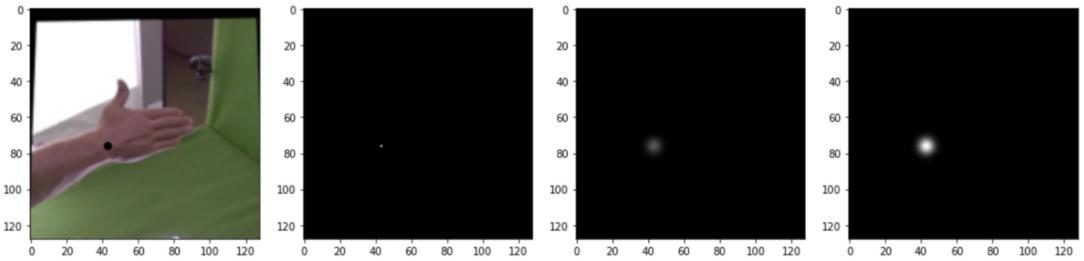
This helps in regressing the points better

Optimum kernel size

Methodology

Step 1: Project the 3D Coordinates onto the 2D Plane.

Step 2: Generate Heatmaps



Hand Image

Point Coordinates

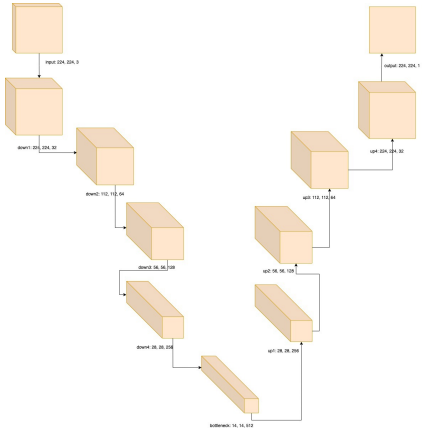
Increase in Gaussian Blur Kernel Size

Methodology

Step 1: Segmenting the hand from the image

Step 2: Model outputs the 21 heatmaps for 21 hand key points

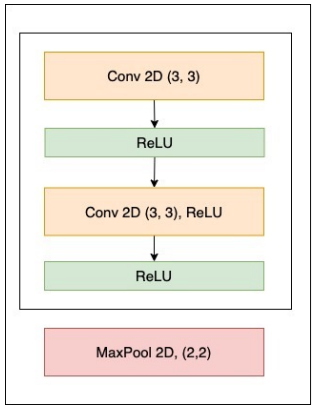
Stacked Convolutional Neural Network



U-Net

Methodology

Step 3: Build Model



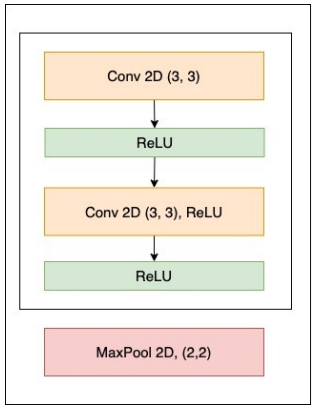
Traditional U-Net Block

Top-down approach to find the perfect depth

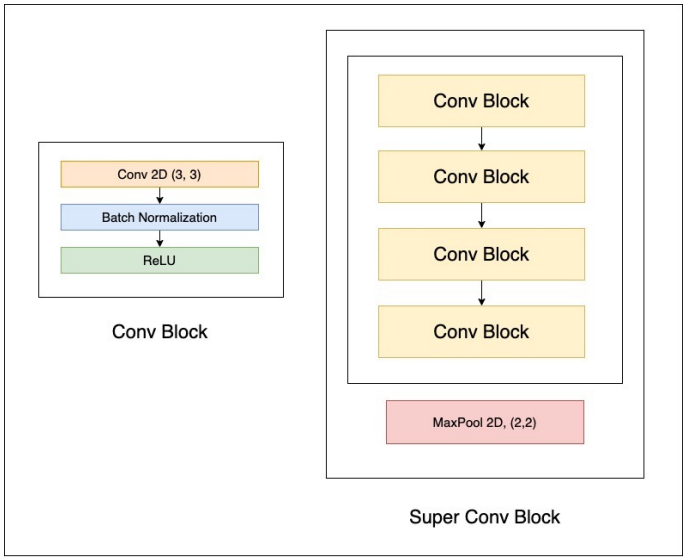
Traditional U-Net blocks involve use of 2 convolution blocks

Methodology

Step 3: Build Model



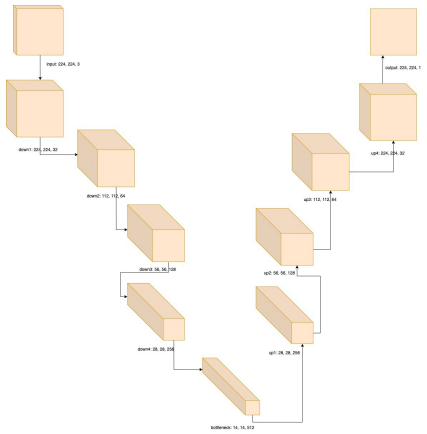
Traditional U-Net Block



Our Modified U-Net Block

Methodology

Step 3: Build Model



Traditional U-Net

4 Down sampling

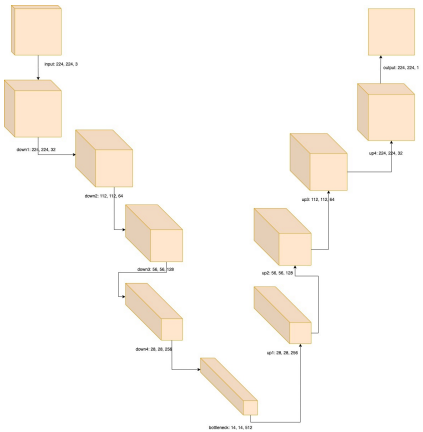
Bottleneck

4 Up sampling

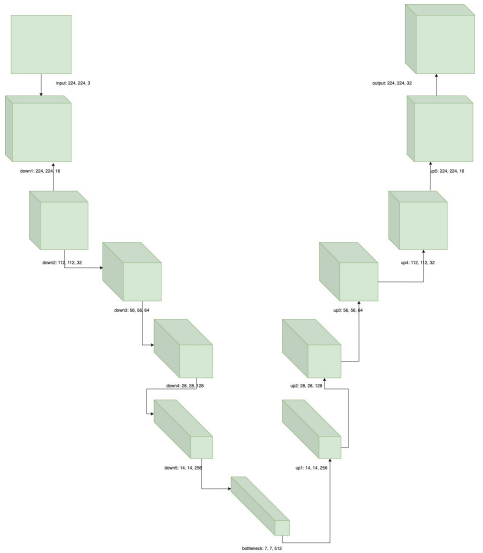
32 Filters in the first down sampling block

Methodology

Step 3: Build Model



Traditional U-Net



Our Modified U-Net

Methodology

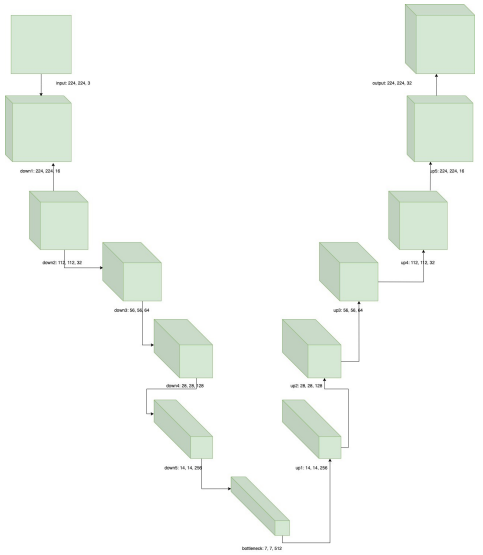
Step 3: Build Model

5 Down sampling

Bottleneck

5 Up sampling

16 Filters in the first down sampling block



Our Modified U-Net

Down sampling:

$$K = \sum P_0 + (7 * P_1) + (4 * P_2)$$

$$P_0 = P_1 * 2$$
$$P_1 := P_1 * 2$$
$$P_2 := P_2 * 2$$

Equation 2: Down sampling parameters
P_0 = First convolution layer parameters
P_1 = Subsequent convolution layers parameters
P_2 = Number of filters in that layer

Up sampling:

$$K = \sum (6 * P_1) + (4 * P_2)$$

$$P_1 := P_1 * \frac{3}{4}$$
$$P_2 := P_2 * 2$$

Equation 3: Down sampling parameters
P_1 = First convolution layers parameters
P_2 = Number of filters in that layer

Block No.	Layer Type	Input Shape 4 * Conv Block	Output Shape 4 * Conv Block	Max Pooling 2D / Up sampling Output Shape	Parameters
1	Down sampling Block 1	224, 224, 3	224, 224, 16	112, 112, 16	7472
2	Down Sampling Block 2	112, 112, 16	112, 112, 32	56, 56, 32	32512
3	Down Sampling Block 3	56, 56, 32	56, 56, 64	28, 28, 64	129536
4	Down Sampling Block 4	28, 28, 64	28, 28, 128	14, 14, 128	517120
5	Down Sampling Block 5	14, 14, 128	14, 14, 256	7, 7, 256	2066432
6	Bottleneck Layer	7, 7, 256	7, 7, 512	14, 14, 512	8261632
7	Up Sampling Block 1	14, 14, 512	14, 14, 256	28, 28, 256	3540992
8	Up Sampling Block 2	28, 28, 256	28, 28, 128	56, 56, 128	885760
9	Up Sampling Block 3	56, 56, 128	56, 56, 64	112, 112, 64	221696
10	Up Sampling Block 4	112, 112, 64	112, 112, 32	224, 224, 32	55552
11	Up Sampling Block 5	224, 224, 32	224, 224, 16	224, 224, 21	13952
-	-	-	-	Final Output	**15,735,680**

Results

Loss Metrics: MSE, IoU

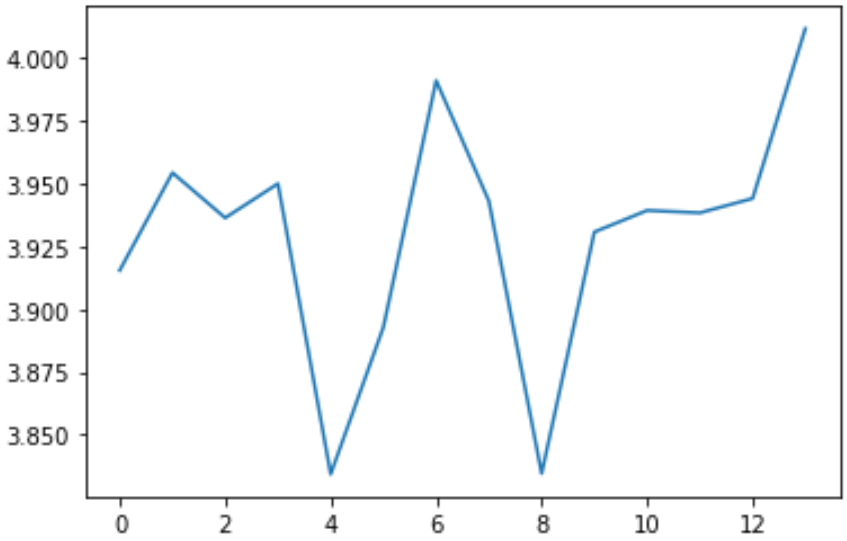
$$IoU\ Loss = 1 - \frac{\sum y_i * t_i}{\sum y_i * y_i + \sum y_i * t_i - \sum y_i * t_i}$$

Equation 4: IoU Loss [13]; $y_i = predicted\ pixel\ value$;
$t_i = target\ pixel\ value$; i = *pixel*

Optimizers: Adam, AdaBoost, AdaDelta

Model Name	Modifications	Mean Pixel % Error	Pixel Error (224, 224, 3) image	Pixel Error (!28, 128, 3) image
U-Net	MSE Loss, Adam	6	15	8
U-Net (3 conv layers)	+ Batch Normalization, Adam	5.8	14	8
U-Net (3 conv layers)	IoU Loss + Batch Normalization, Adam	4.5	10	6
Super U-Net (4 conv layers)	MSE Loss + Batch Normalization, Adam	4.9	10	6
Super U-Net (4 conv layers)	IoU Loss + Batch Normalization, Adam	4.5	10	6
Super U-Net (4 conv layers)	IoU Loss + Batch Normalization, Adam	4.3	10	6
Super U-Net (4 conv layers)	IoU Loss + Batch Normalization, AdaBoost	4.4	10	6
Super U-Net (4 conv layers)	**IoU Loss + Batch Normalization, AdaDelta**	**3.9**	**9**	**5**

Results



Error % per 1000 image batch in the eval set

The highest error of the final model is lower than the lowest error of all other models!

Final error % per pixel: 3.9

Results

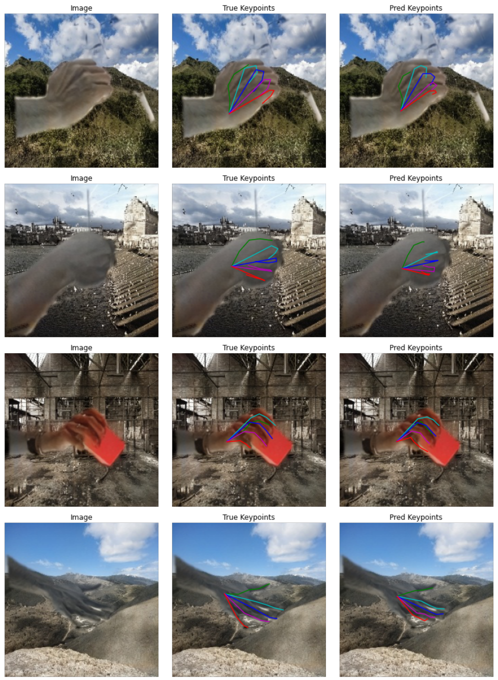


Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints

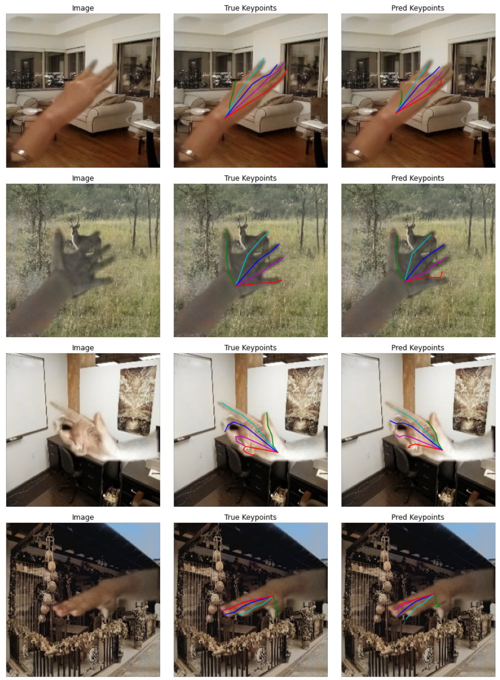


Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints

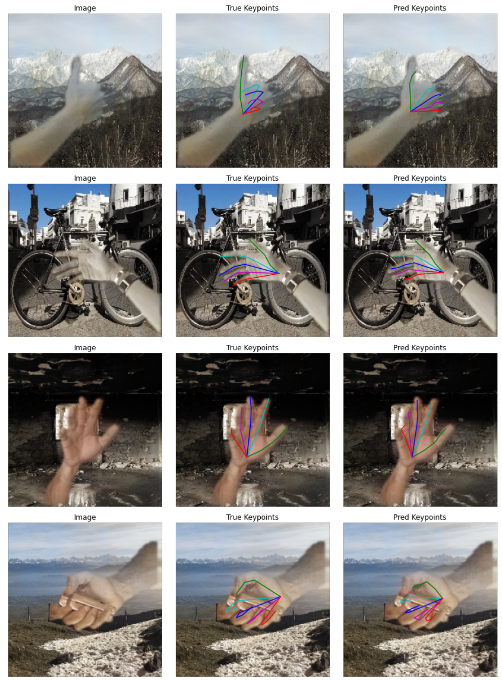


Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints
Image
True Keypoints
Pred Keypoints

Next Step

Improve landmark error

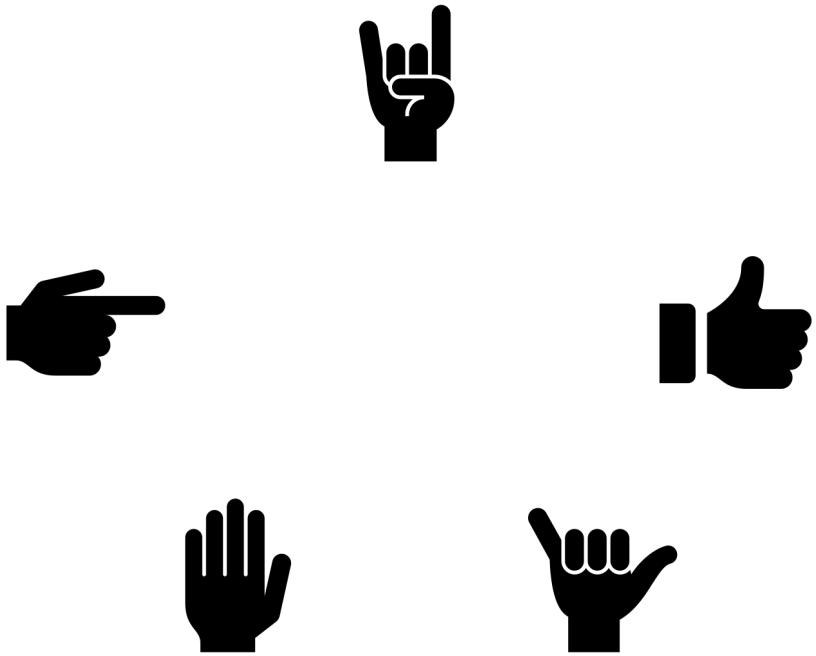
Improve frame rate, ensuring 24-30 FPS.

Part 1 A New methodology for efficient 2-D Hand Pose Estimation

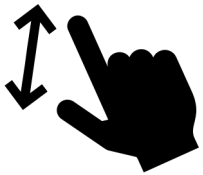
Part 2 A New methodology for Dynamic Gesture Recognition

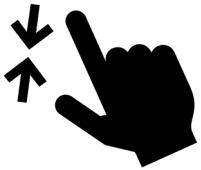
Part 3 Reproducible research and contributing to the community

Gesture Recognition



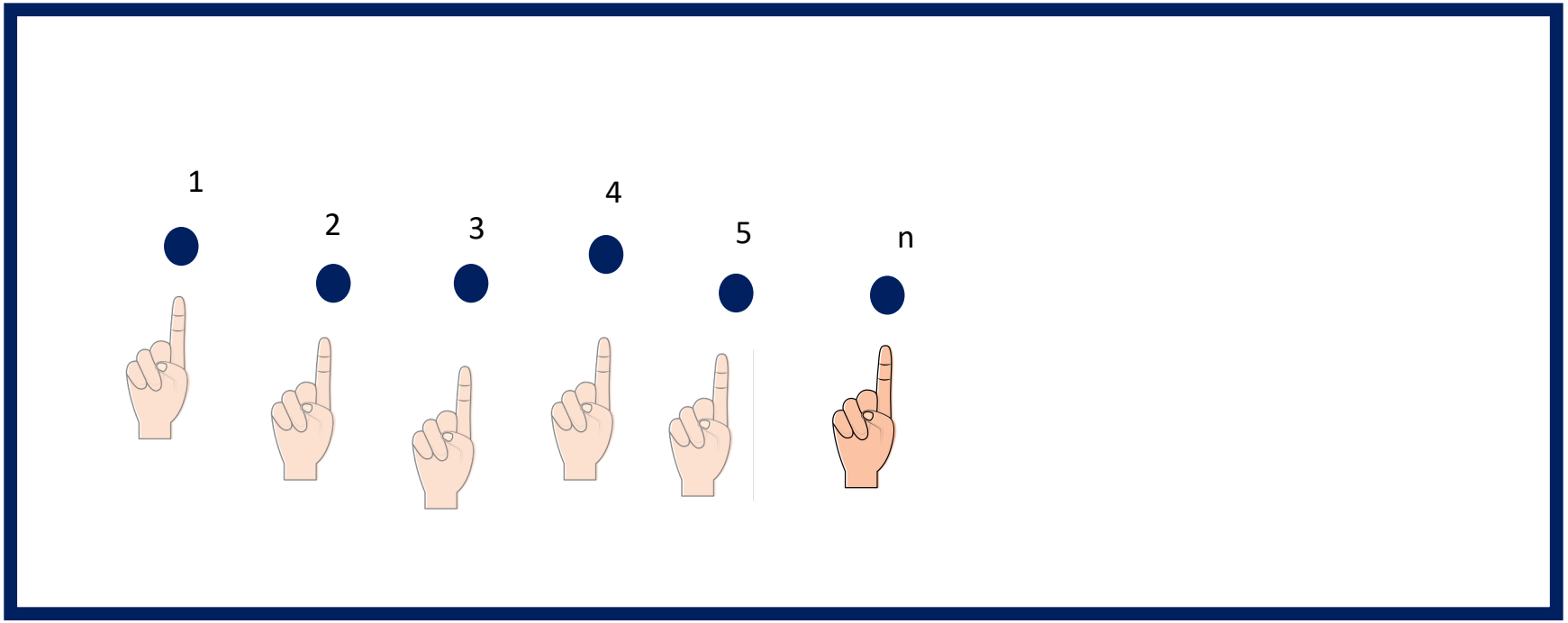
Dynamic Gesture Recognition





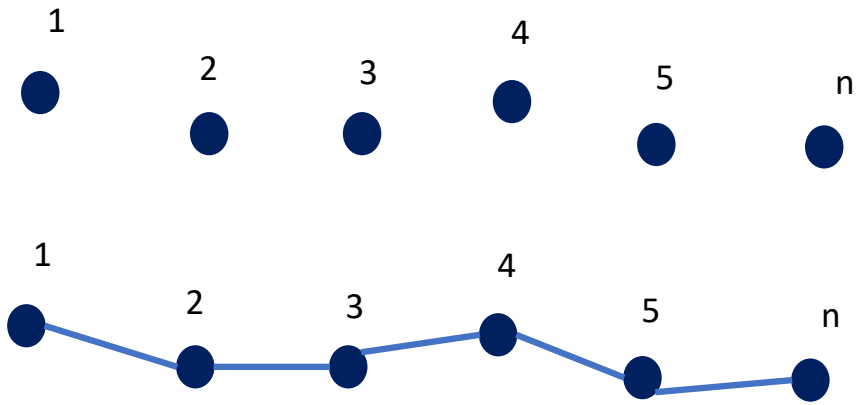
Dataset

Custom recorded dataset using Mediapipe



Dataset

Custom recorded dataset using Mediapipe



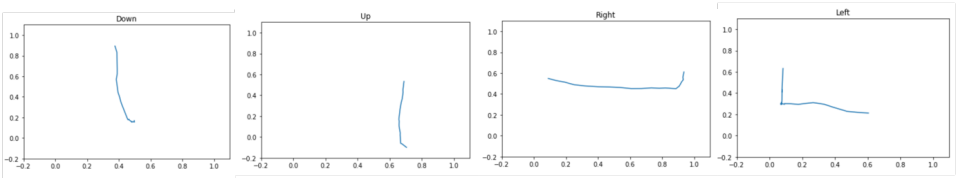
Dataset

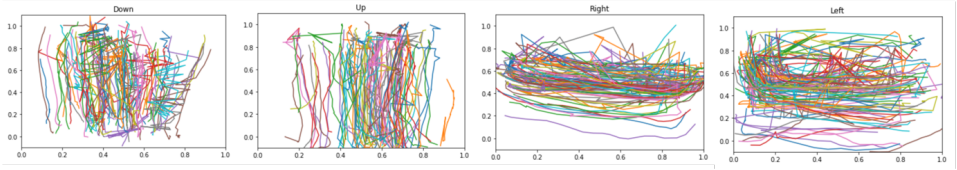
Custom recorded dataset using Mediapipe

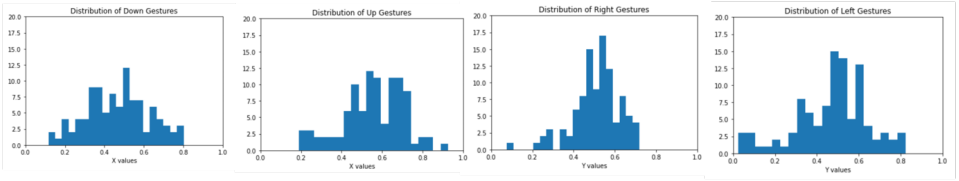
Variable feature lengths [11, 16, 21]

Multi axes

Equally distributed across the frame

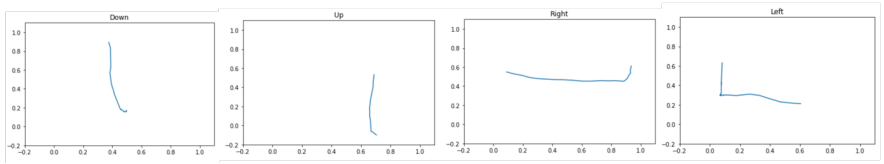


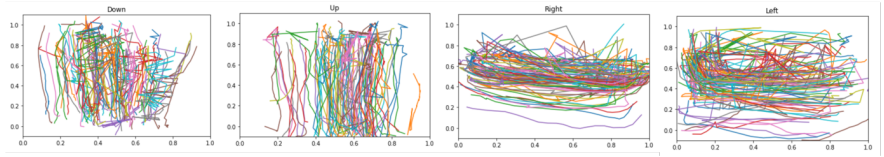


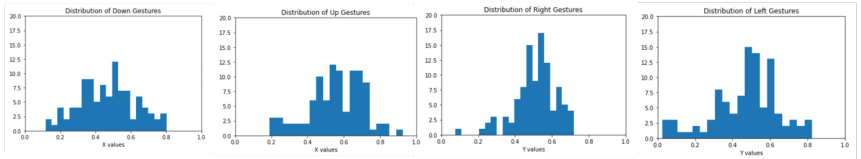


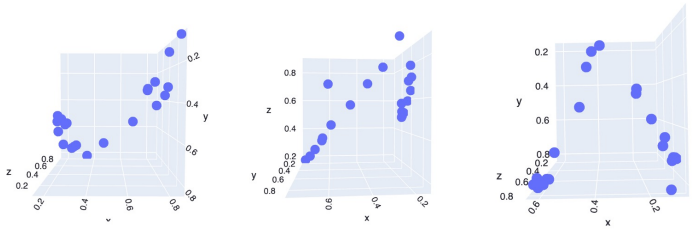
Dataset

Single Point Gestures





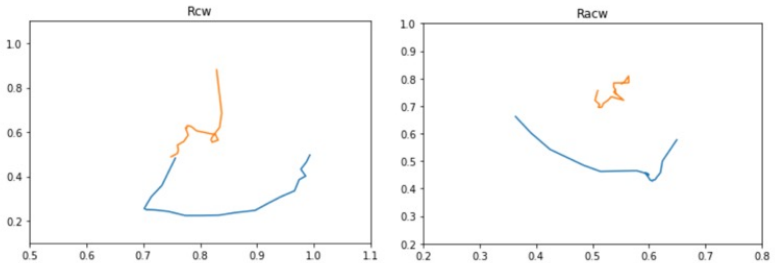


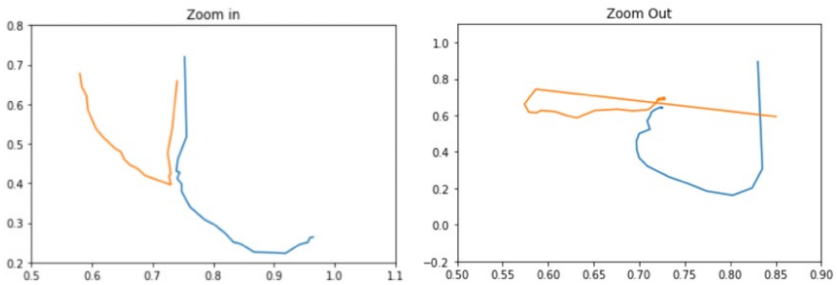


Tick Star Triangle

Dataset

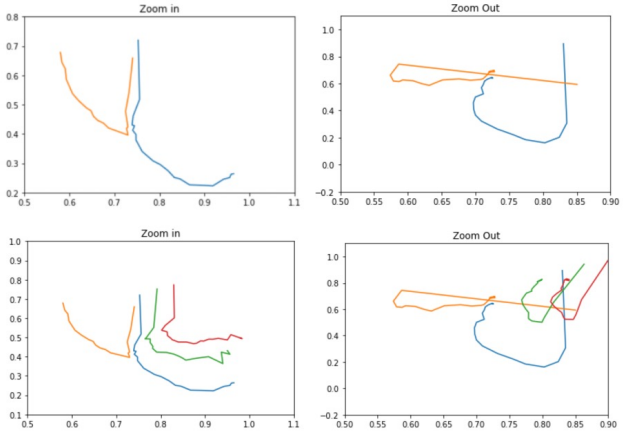
Two Point Gestures





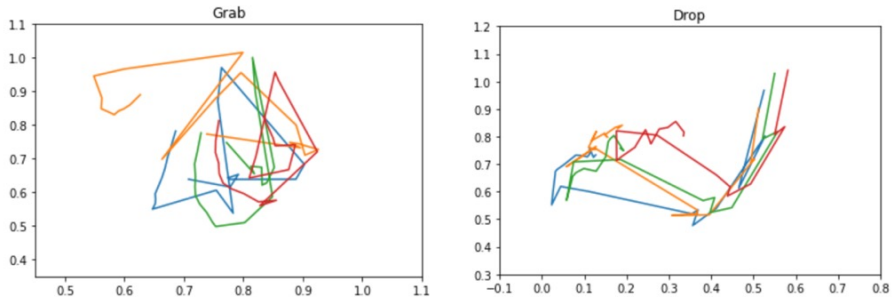
Dataset

Multi Point representation of two-point Gestures

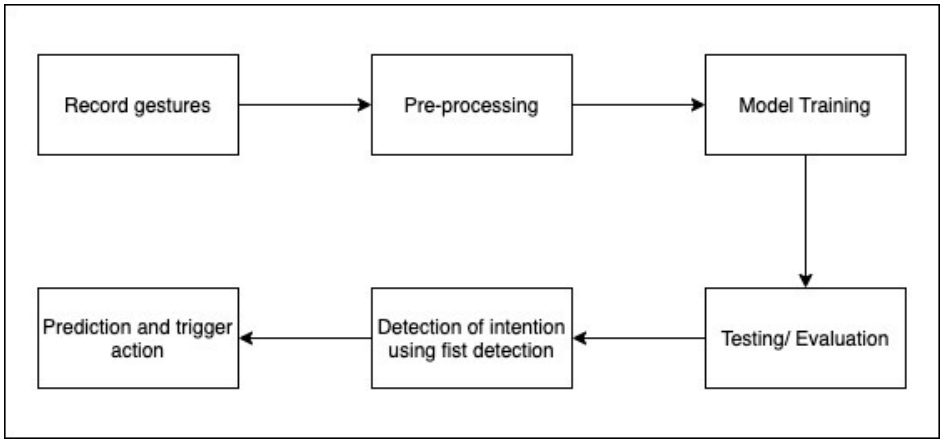


Dataset

Multi Point Gestures

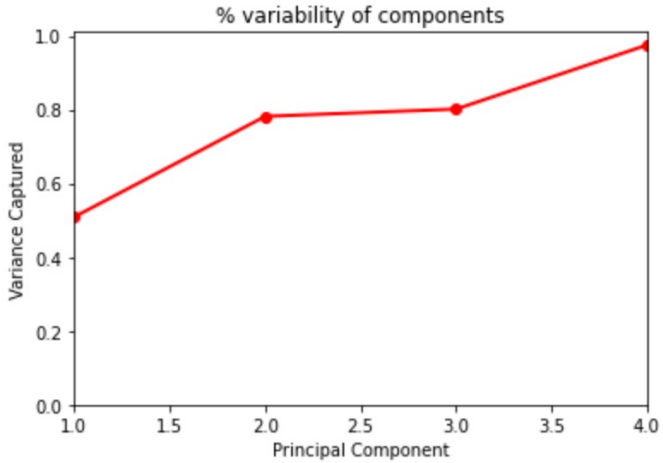


Methodology



Methodology

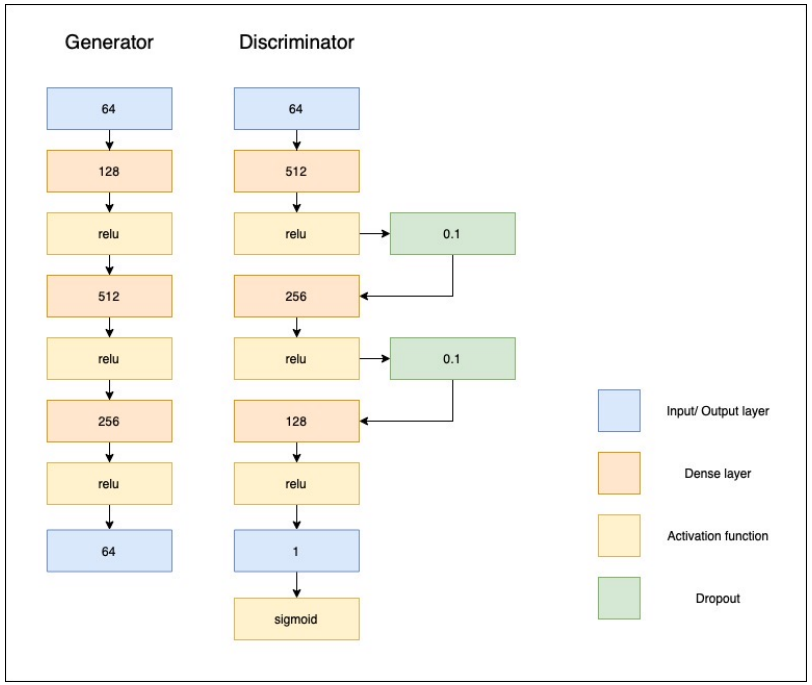
PCA

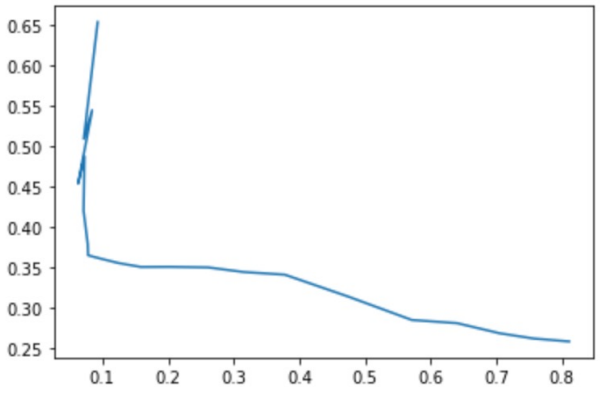


Initial dimensions were 63 features. in the final setup, 4 components were used which captured 97.49% variability. This helped reduce dimensions by 93.65%.

Methodology

GAN



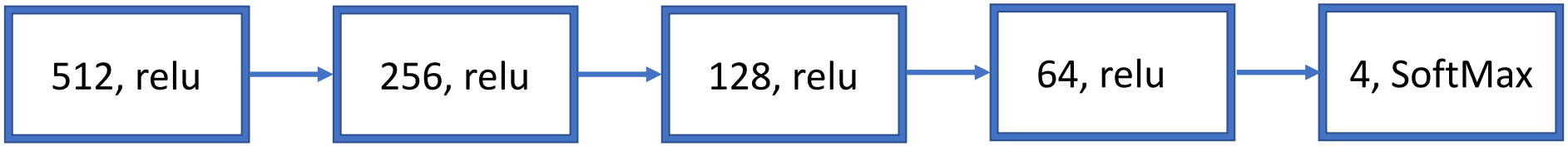


Generated Left gesture

Methodology

Training

Input: 1, f * 21*3, 3 for each axes, f = number of point or fingers required in that gesture



Output also trained for category and combined classes

Results

Gesture	Gesture Category	Category Accuracy	Multi Point Accuracy
Left	Single Point	97.5%	88.8%
Right	Single Point		
Up	Single Point		
Down	Single Point		
Tick	Single Point		
Star	Single Point		
Triangle	Single Point		
Zoom In	Two Point	91.2%	
Zoom Out	Two Point		
Rotate Clockwise	Two Point		
Rotate Anti-Clockwise	Two Point		
Grab	Multi Point	90.1%	
Drop	Multi Point	90.1%	

Part 1 A New methodology for efficient 2-D Hand Pose Estimation

Part 2 A New methodology for Dynamic Gesture Recognition

Part 3 Reproducible research and contributing to the community

Problems in Reproducing Research

Developers struggle with code

Unclear methodology

Use of multiple apps and platforms that cause delays

AirInteract

Record Gestures

Pre-process them

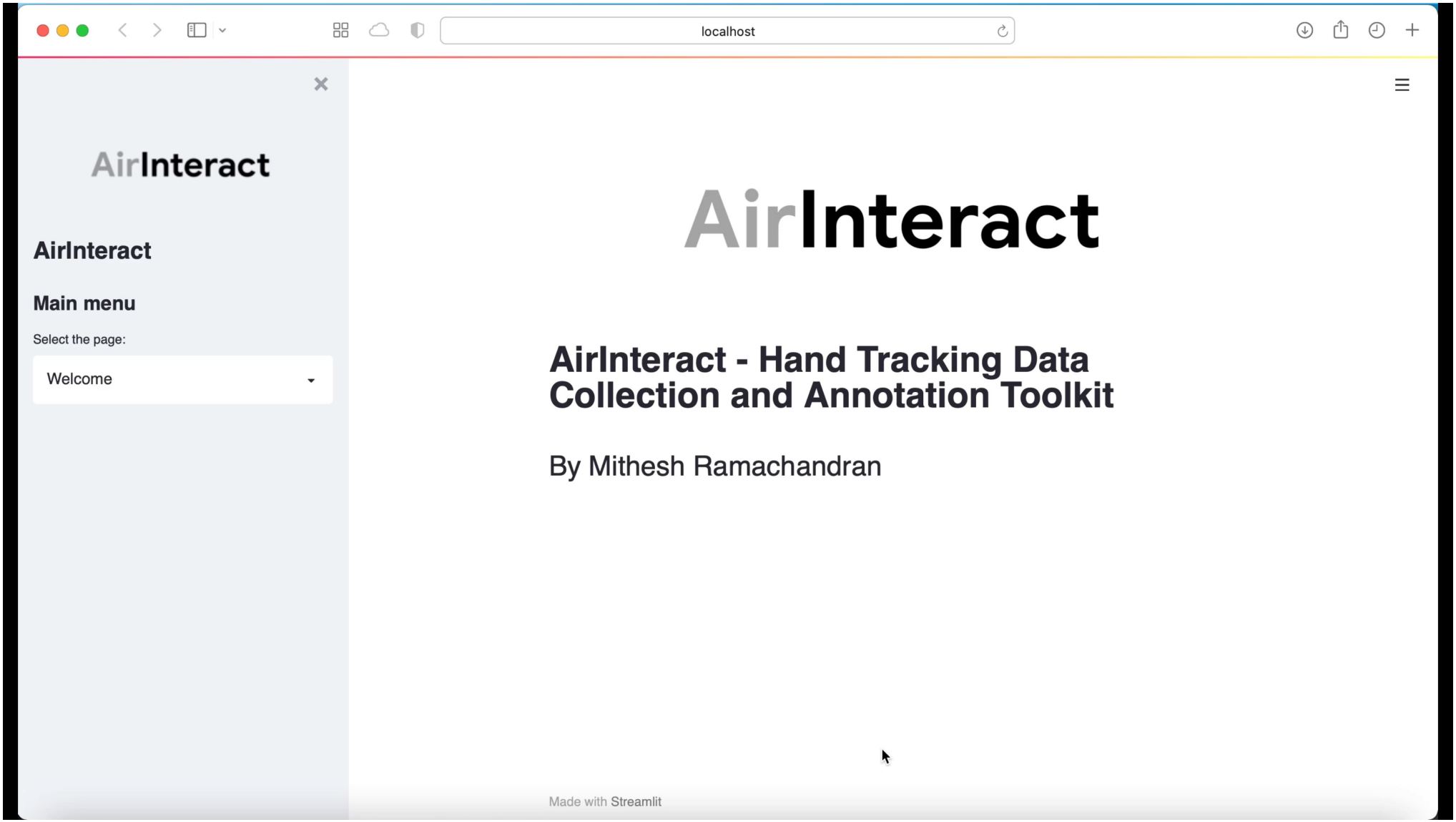
Visualize them

Train models

Evaluate and Test

Without a single line of code

Demo


localhost
AirInteract
AirInteract
Main menu
Select the page:
Welcome
AirInteract
AirInteract - Hand Tracking Data Collection and Annotation Toolkit
By Mithesh Ramachandran
Made with Streamlit