Specialist Diploma in Artificial Intelligence

AI Applications with Deep Learning

Recognizing Activities from Pose Estimations

Sub topics

- Human Pose Estimation
- Recurrent Neural Networks for Time-Series Classification
- What Deep Learning Algorithms Can We Use?
- Human Activity Recognition
- Dataset and the Importance of Data Normalization

Recognizing Activities from Pose Estimations

HUMAN POSE ESTIMATION

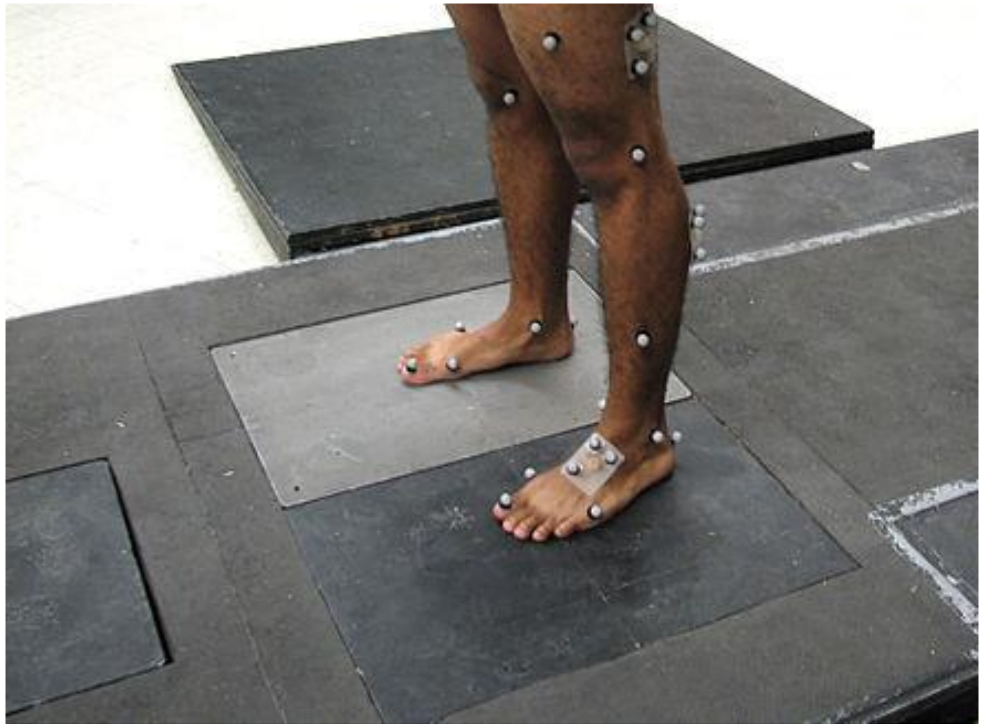
What is Human Pose Estimation

Based on Google’s Posenet:

Pose estimation is the task of using an ML model to estimate the pose of a person from an image or a video by **estimating the spatial locations of key body joints** (keypoints).

Extracting Human Pose

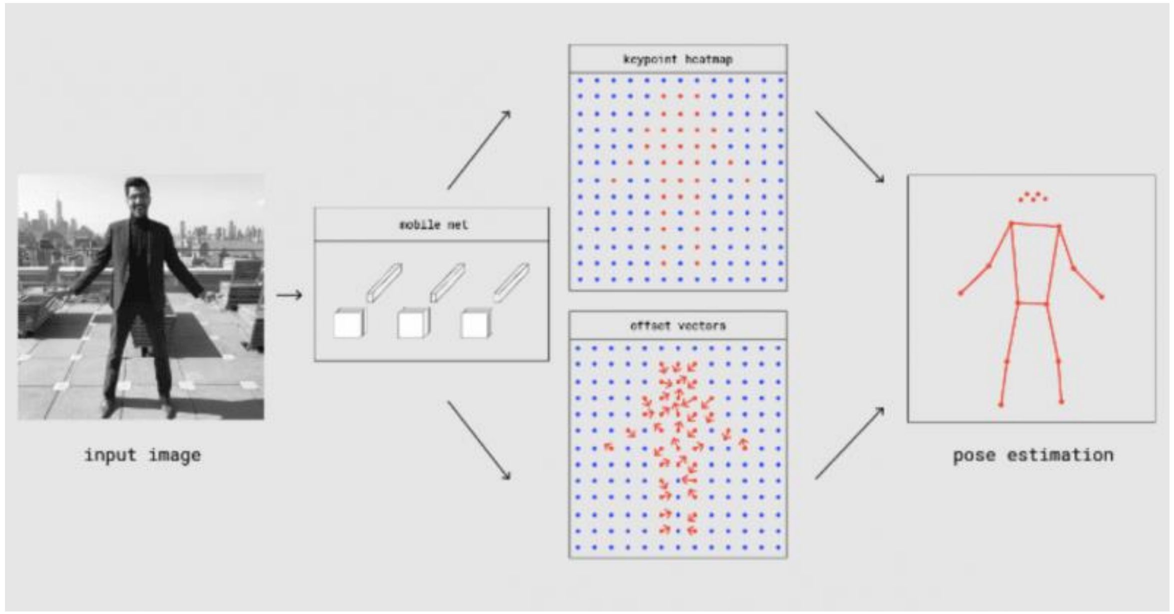
- Human Pose be extracted using:
  - Motion Capture sensors
  - Widely used in movie / gaming industry
  - Accurate (but expensive)



https://en.wikipedia.org/wiki/Motion_capture

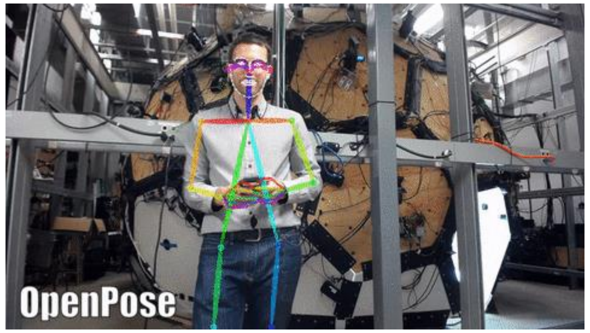
Extracting Human Pose

- Human Pose be extracted using:
  - Deep Learning technologies



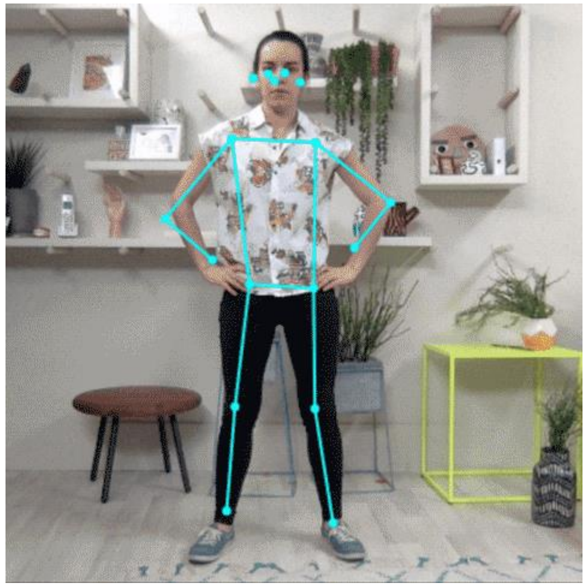
Deep Learning-Based Human Pose Estimation Technology Today

- CMU OpenPose
  https://github.com/CMU-Perceptual-Computing-Lab/openpose

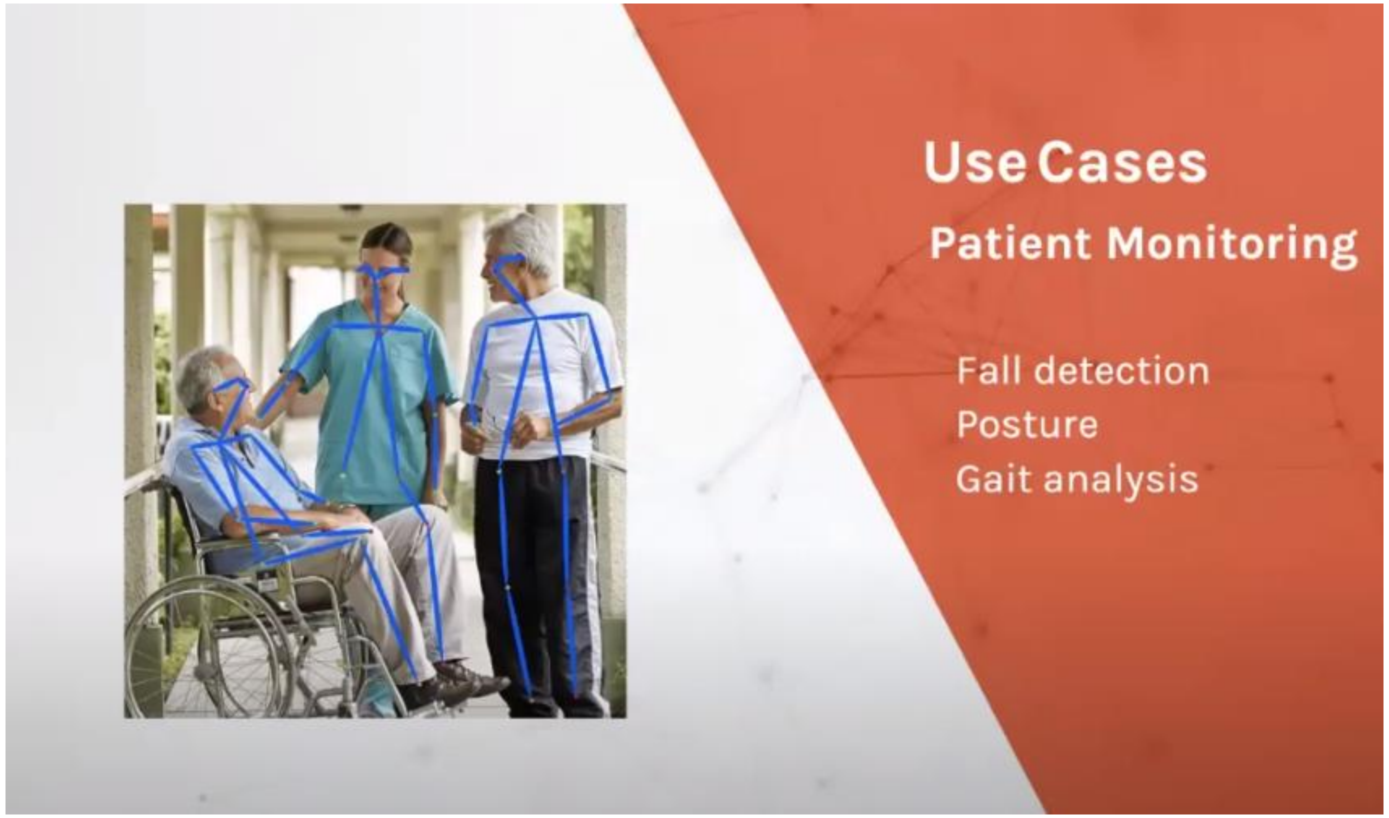


- Google's Tensorflow.js PoseNet
  https://www.tensorflow.org/lite/models/pose_estimation/overview

  (these models rely on CNN and have been pre-trained)



Use Cases with Human Pose Estimation


Use Cases

Patient Monitoring

Fall detection

Posture

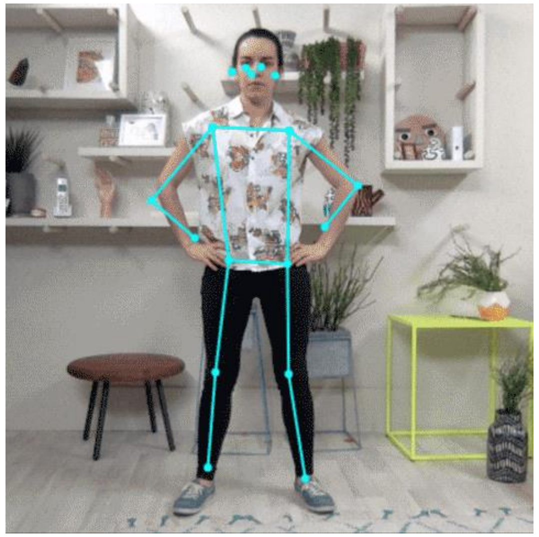
Gait analysis

https://youtu.be/rh8Vzp0REP0?t=960

Data Captured from Human Pose Estimation Model

- A typical human pose estimation library is able to detect the following:

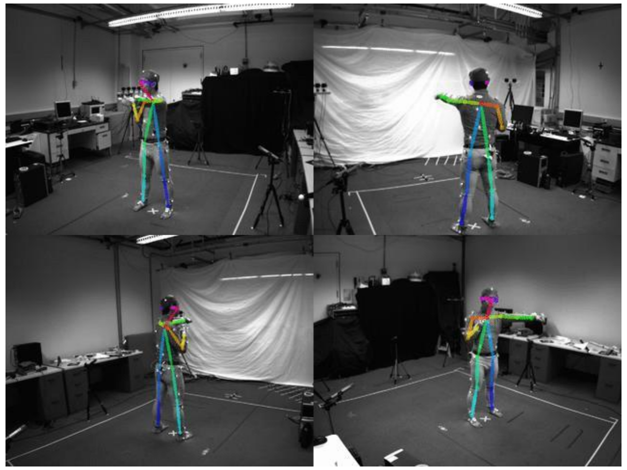
  - 0: Nose X, Y
  - 1: Neck X, Y
  - 2: Right Shoulder X, Y
  - 3: Right Elbow X, Y
  - 4: Right Wrist X, Y
  - 5: Left Shoulder X, Y
  - 6: Left Elbow X, Y
  - 7: Left Wrist X, Y
  - 8: Right Hip X, Y
  - 9: Right Knee X, Y
  - 10: Right Ankle X, Y
  - 11: Left Hip X, Y
  - 12: Left Knee X, Y
  - 13: Left Ankle X, Y
  - 14: Right Eye X, Y
  - 15: Left Eye X, Y
  - 16: Right Ear X, Y
  - 17: Left Ear X, Y

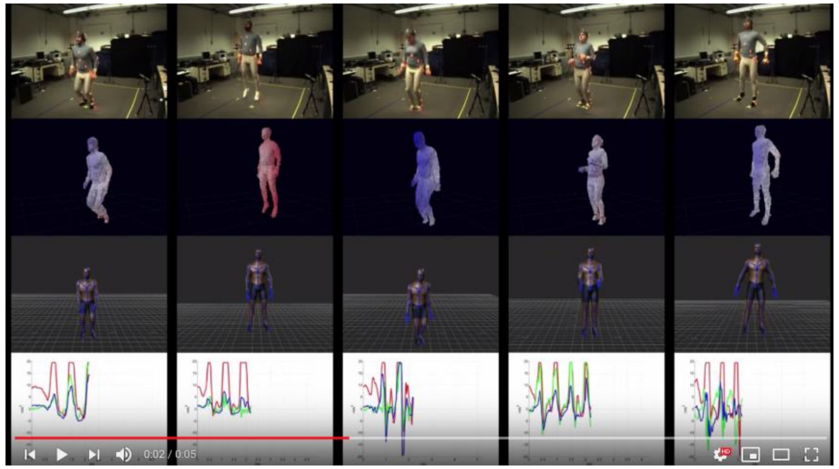


- Some advanced algorithms extract facial landmarks and finger joints.

Human Activity is a Time Series of Data

- When a person does something, the body’s position changes over time.
- So the key points are considered time-series of data





https://www.youtube.com/watch?v=Erohxy3etlw&list=PLWlWeDxs3DyEMehE20ROwshVFtbyril_j

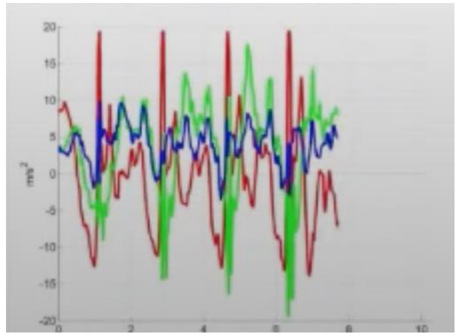
Understanding the Task and the Type of Data

- In this exercise we want to be able to classify from a video clip, whether a person is perform one of the following actions:
  - Waving, Standing, Jumping, Squatting, Punching, etc…

- Before we can decide on the type of Deep Learning architecture to classify our data, we have to understand our data

Each Time Series is a Sequence

- If we plot the joint positions of a person over time, they will look like:

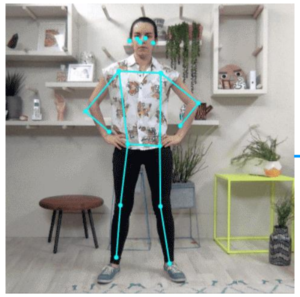


- We first have to know that the data is a time series
  (Time is an important aspect for some of the actions)
- To classify time series data, we should use:
  - Recurrent Neural Networks (or a variant such as GRUs or LSTMs)

Recognizing Activities from Pose Estimations

RECURRENT NEURAL NETWORKS FOR TIME-SERIES CLASSIFICATION

Classifying Single Frame vs Multiple Frames



1 image captured
from camera

Human
Pose
Estimstion

307.589, 162.976,
319.364, 205.944,
293.267, 204.68,
285.434, 250.281,
277.616, 286.729,
349.276, 208.539,
357.095, 255.523,
349.362, 297.17,
297.183, 290.662,
307.599,351.938,
308.905,401.454,
329.758,291.958,
333.701,357.132,
337.546,411.953,
304.994,160.291,
315.459,160.269,
329.752,161.651

X,Y coordinates of joints
of 1 image frame

Dense Layers

SITTING?
STANDING?
SQUATTING?

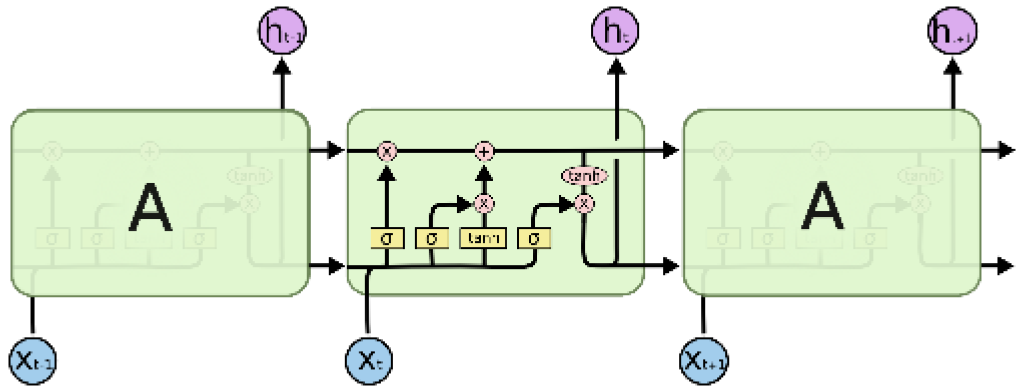
We can use a basic Fully-Connected (or Dense) Layer Neural Network to classify 1 frame of joint coordinates.

But what if we want to recognize the activity that must be recognized as a sequence?

(such as running / jumping / walking)

Long Short Term Memory (LSTM)

- Recall:
  - A LSTM is a special type of recurrent neural network has a memory cell used to store / forget memory to improve accuracy on long sequences.

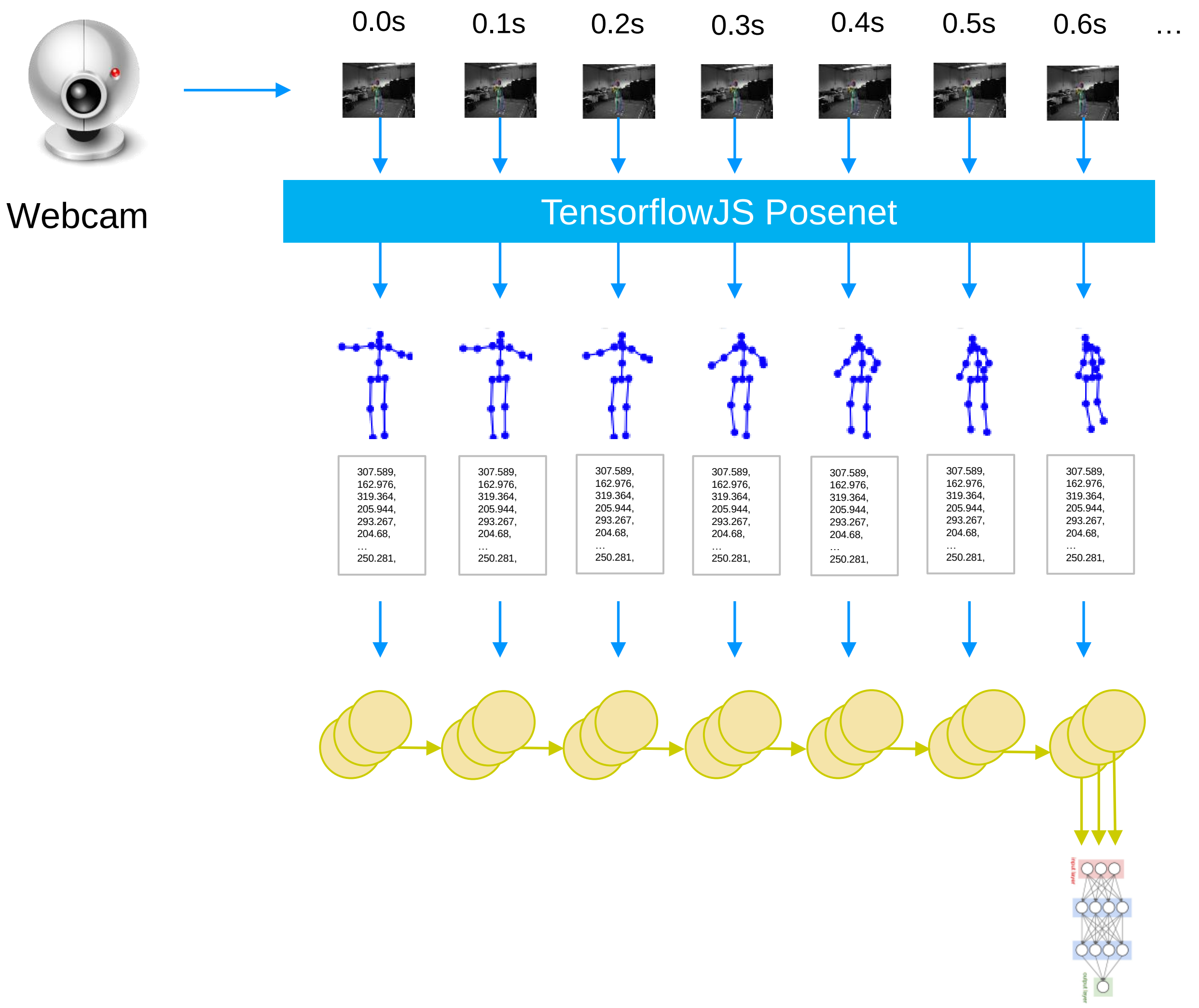


Long Short Term Memory (LSTM)

- Commonly used for classifying time-series:
  - Retains reasonable memory for long (but not too long) sequences
  - Sliding window approach

- In our practical, we will use LSTM to "retain" memory about important parts of the sequence.

Possible Neural Network Architecture



Raw Video Frames
Each frame is 1 image

Single Human Pose Estimation with CNN

Time-series of joints' coordinates
In our practical, we feed a total of 32 time-steps, each step contains 18 pairs of (x,y) coordinates into our LSTM Network

LSTM Layer (s)

Dense Layer(s)

Let’s Discuss

- From a video sequence, using human pose estimation, you can a time-series of coordinates for joints.

- Is there anything that you must consider before sending the coordinates to your Deep Neural Network for training / prediction?

Recognizing Activities from Pose Estimations

DATASET AND THE IMPORTANCE OF DATA NORMALIZATION

The Dataset

- The dataset that we will use:
  - https://github.com/stuarteiffert/RNN-for-Human-Activity-Recognition-using-2D-Pose-Input

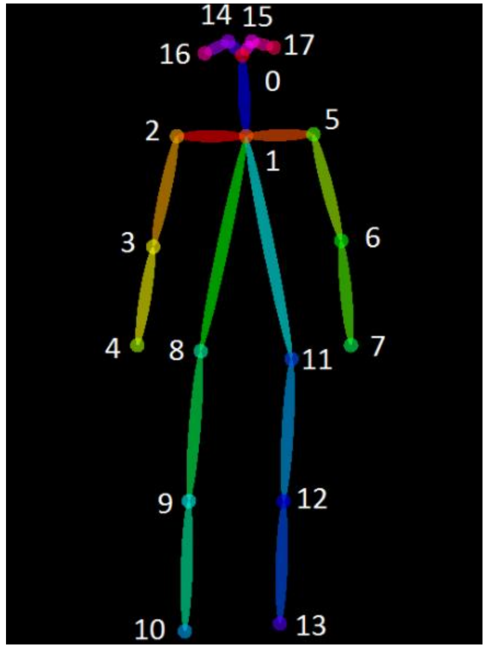
- Important notes in the readme:
  - 18 joint X and Y position keypoints and accuracies per frame
  - keeping only X and Y positions of each frame
  - very little preprocessing has been done to the dataset

The Dataset

- Files:
  - **X_test.txt** : testing dataset x inputs
    (36 keypoints per line, 32 lines per datapoint)
  - **X_train.txt** : training dataset x inputs
    (36 keypoints per line, 32 lines per datapoint)
  - **Y_test.txt** : testing class labels
  - **Y_train.txt** : training class labels

The Dataset

- The 36 key points corresponds to the 18-skeletal point **on-screen coordinates**:
  - 0: Nose X, Y
  - 1: Neck X, Y
  - 2: Right Shoulder X, Y
  - 3: Right Elbow X, Y
  - 4: Right Wrist X, Y
  - 5: Left Shoulder X, Y
  - 6: Left Elbow X, Y
  - 7: Left Wrist X, Y
  - 8: Right Hip X, Y
  - 9: Right Knee X, Y
  - 10: Right Ankle X, Y
  - 11: Left Hip X, Y
  - 12: Left Knee X, Y
  - 13: Left Ankle X, Y
  - 14: Right Eye X, Y
  - 15: Left Eye X, Y
  - 16: Right Ear X, Y
  - 17: Left Ear X, Y



https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md

The Problem

- The problem:
  - Captures only on-screen x and y positions of the skeleton keypoints in each frame
  - “The data has **not** been normalised with regards to subject position in the frame, motion across frame (if any), size of the subject, speed of action etc. It is essentially the raw 2D position of each joint viewed from a stationary camera.”

The Problem

- As a result:
  - If the end user uses cameras with different resolutions from the camera used to capture the training data, the coordinates will look very different
  - The distance the person stands from the camera affect how “closely” the joint coordinates will be clustered together
  - The position of the person on the view will also affect the actual coordinates of his joints in the camera.

Data Normalization

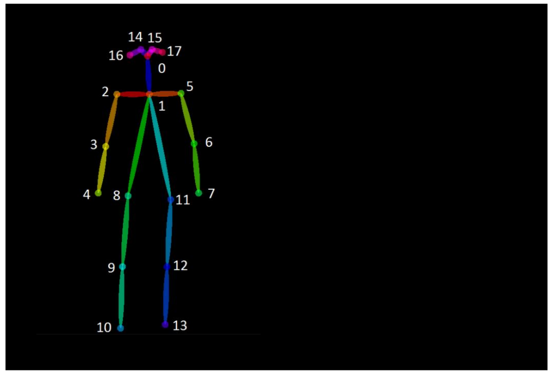
- Normalization is a technique:
  - Applied as part of data preparation for machine learning.

- Goal:
  - Change values of numeric columns in the dataset to a common scale
  - Must not distort differences in the ranges of values or lose information

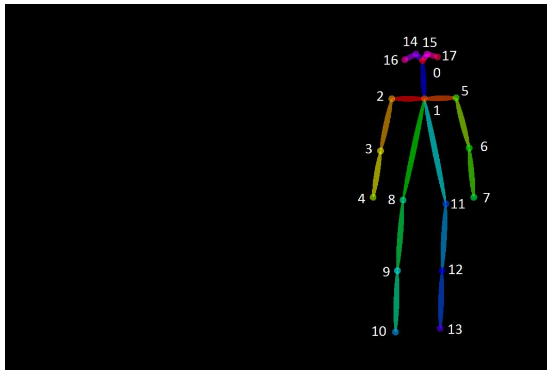
  - NOTE: Normalization is also required for some algorithms to model the data correctly.

Data Normalization

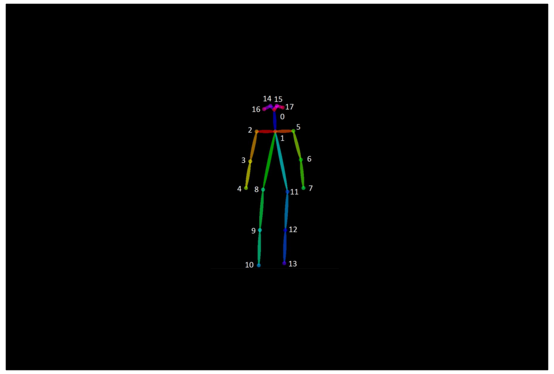
- We have to account for the following:
  - Person at different positions on the screen (key points will move in a group)
  - Person nearer / further from the camera (key points are closer)



Person on left of camera



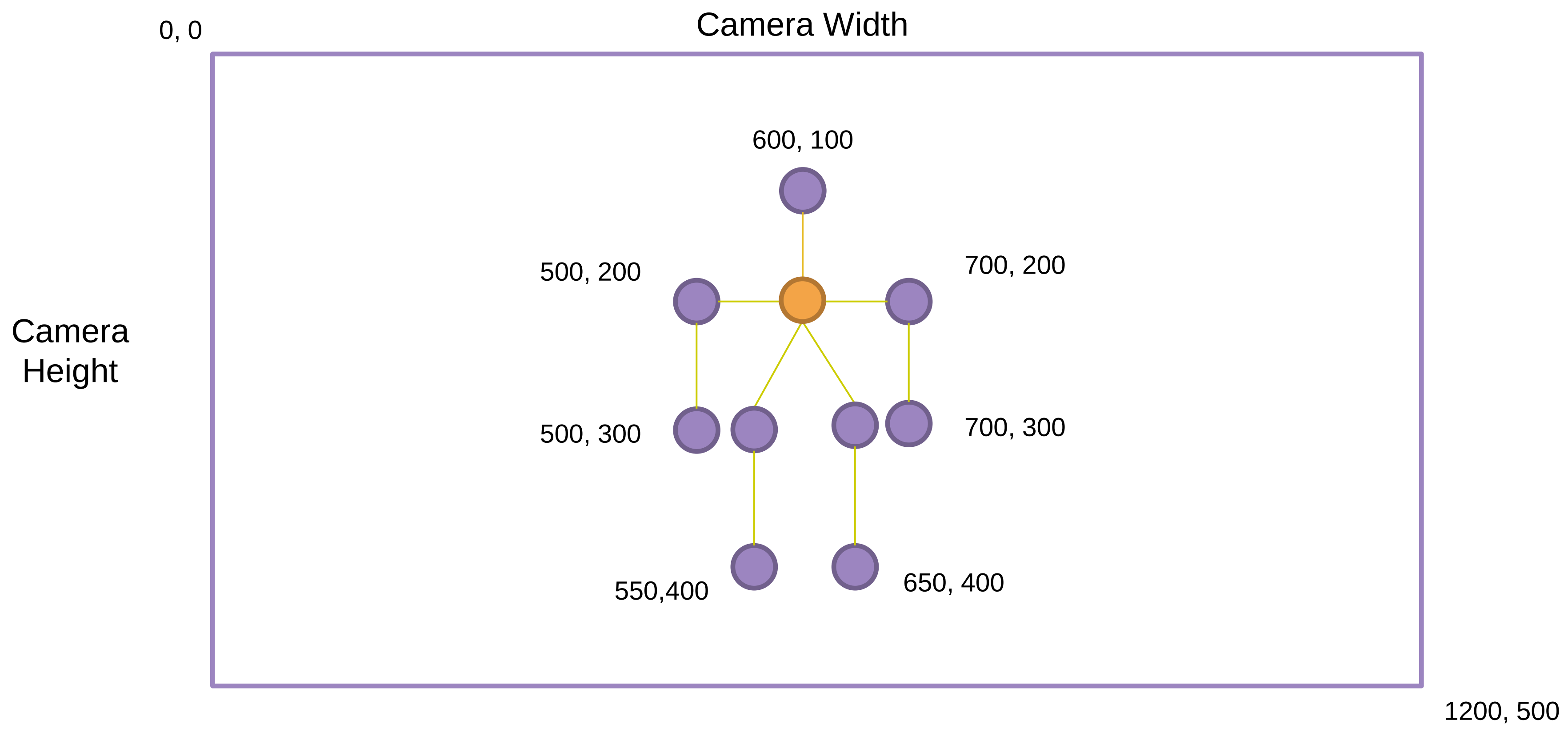
Person on right of camera



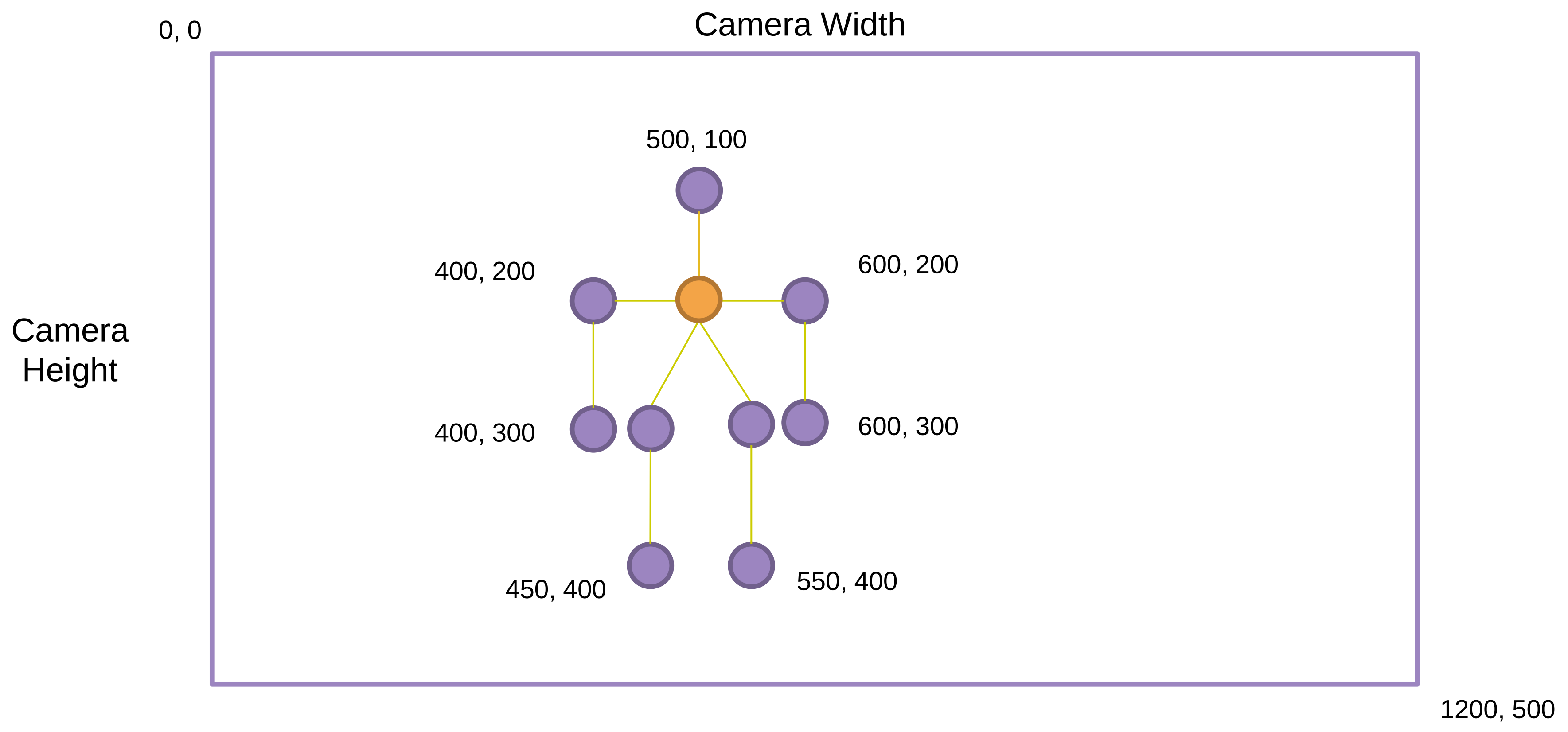
Person far from camera

Same poses, but the x, y coordinates of each key point is **different**

Data Normalization

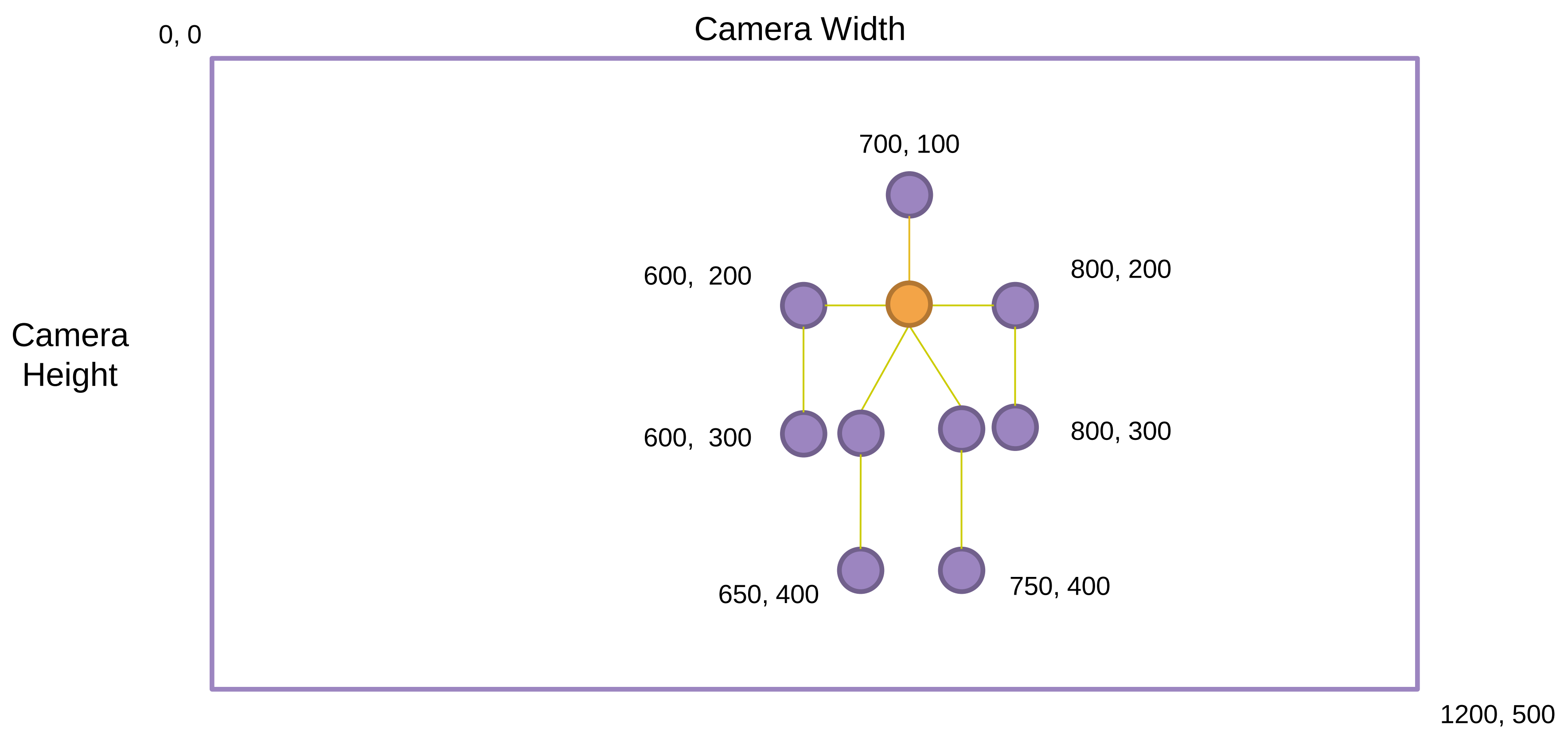


Data Normalization



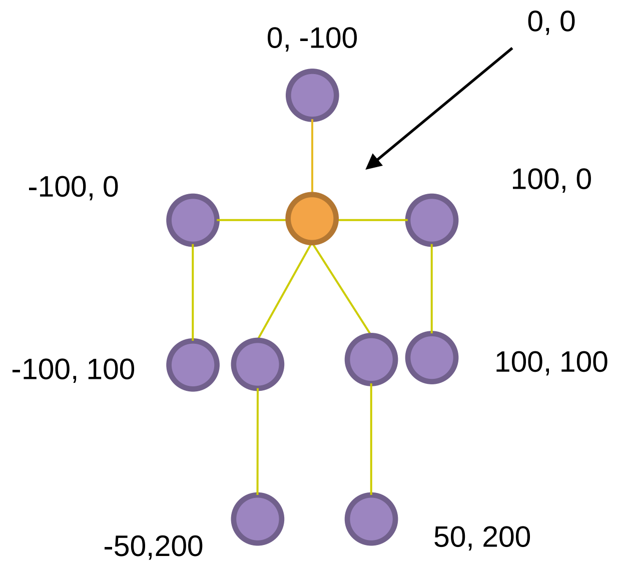
Shift left from screen center by 100 pixels

Data Normalization



Shift right from screen center by 100 pixels

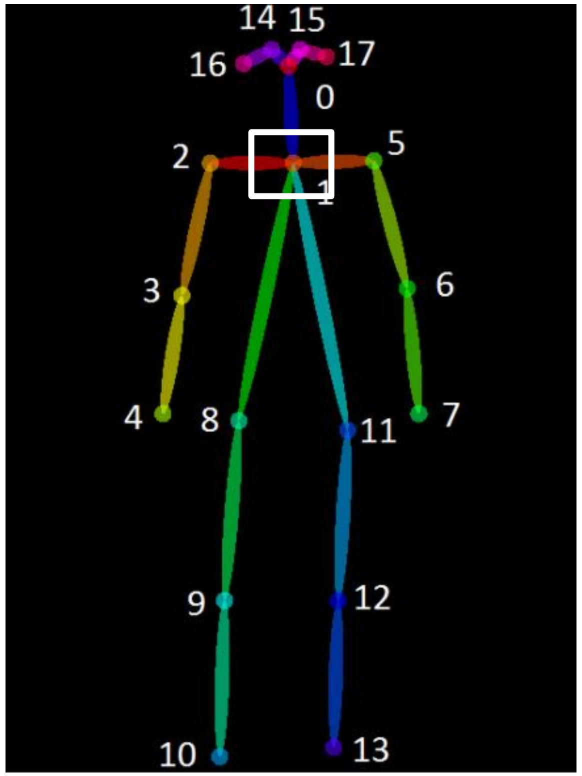
Data Normalization



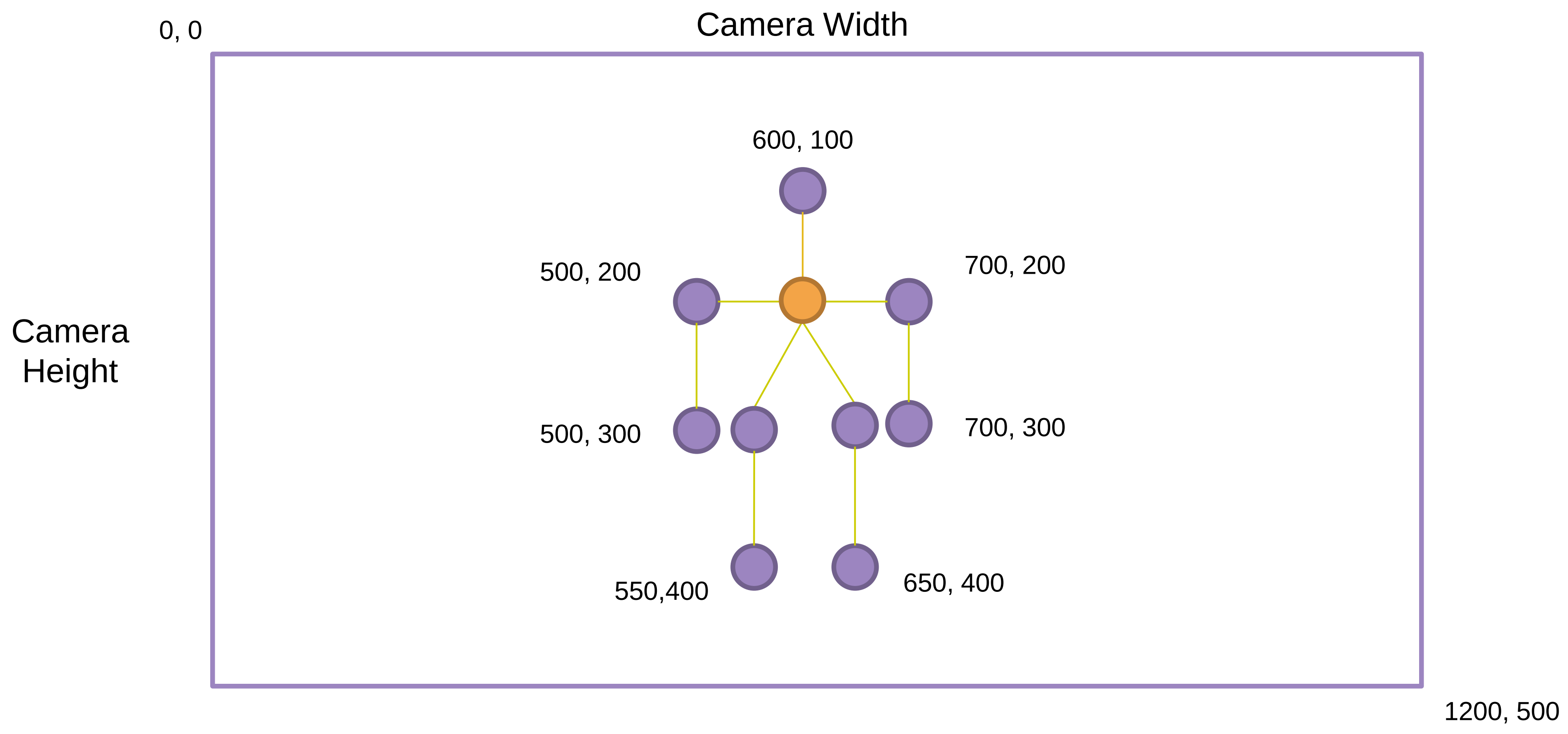
Normalized by taking the neck position as 0,0 and shifting the other points accordingly

Data Normalization

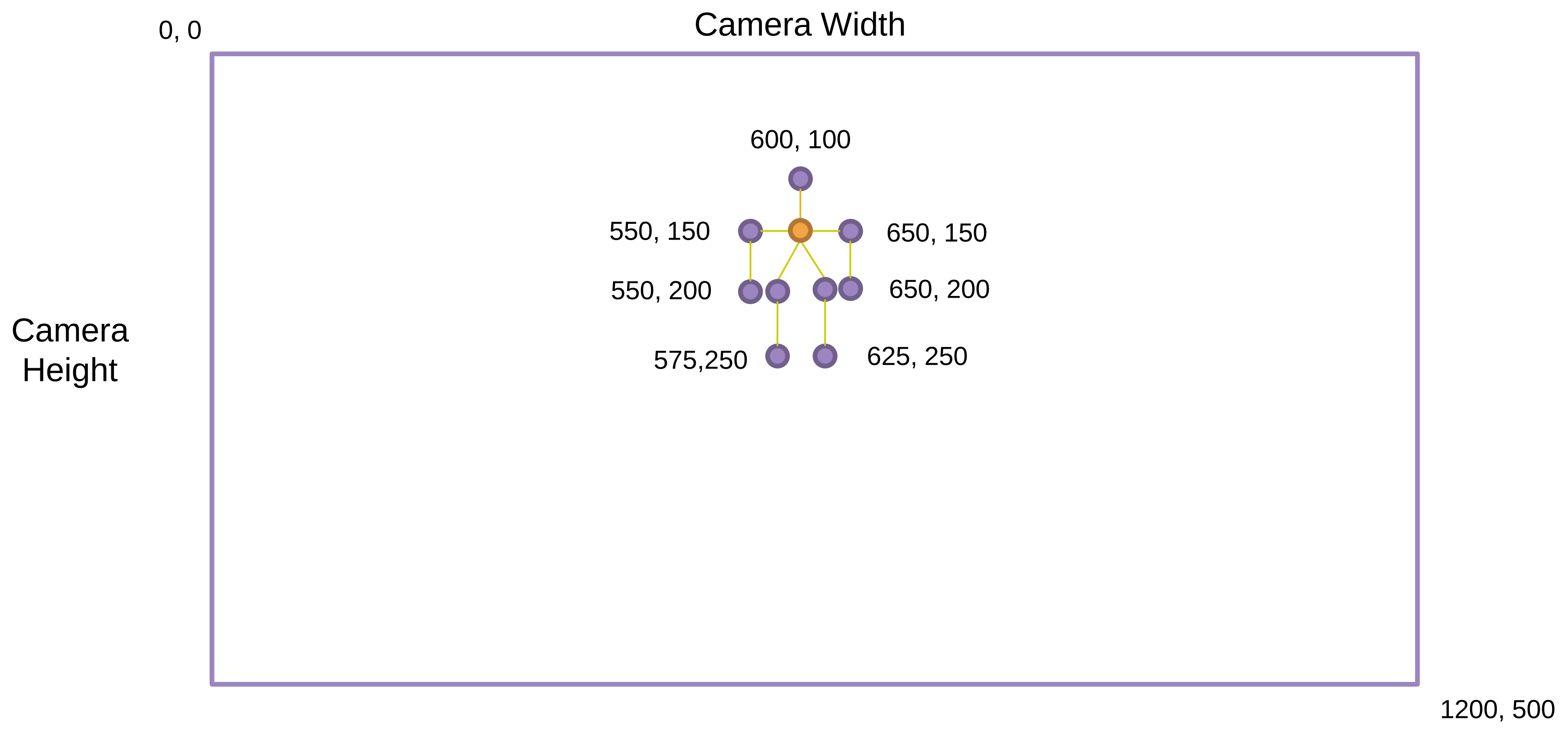
- To make it easier for our neural network to learn, we can pre-process and normalize our dataset.
- We use some reference points:
  - Neck point (point 1) as the centre of the entire skeleton
  - If the neck point can't be detected, we find the mid-point of the 2 shoulder points (point 2, 5)



Data Normalization

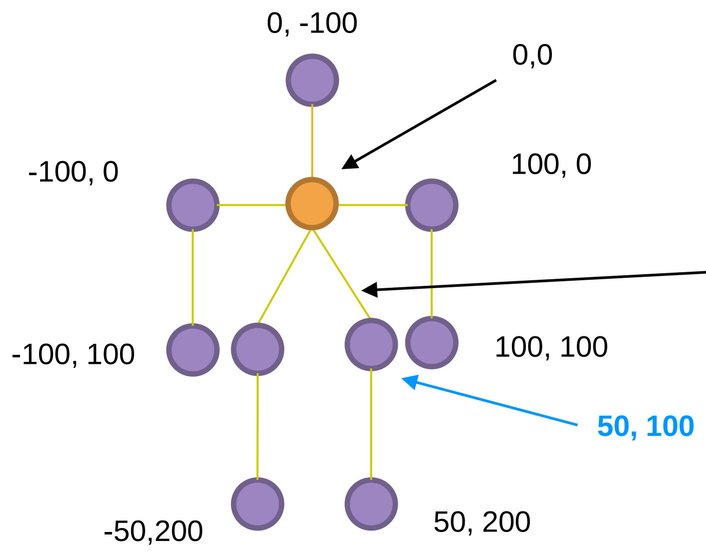


Data Normalization



Person is standing further away:
Size of the person frame is halved in this case

Data Normalization

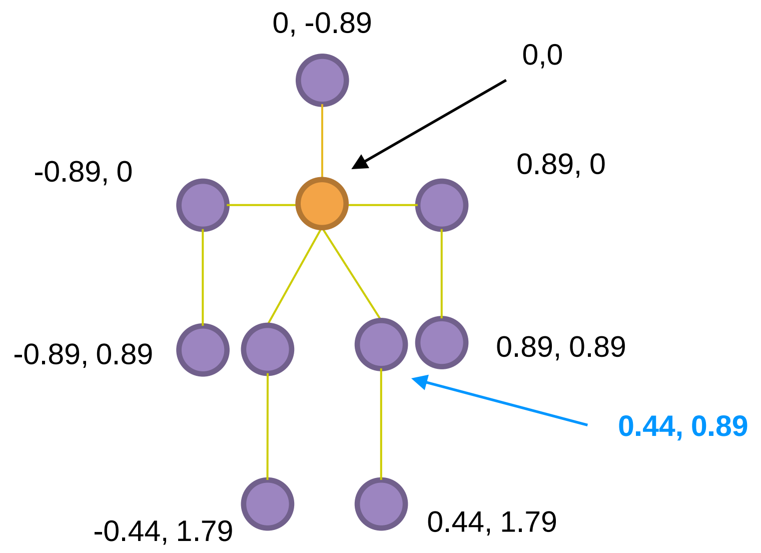


Torso length
= sqrt(50*50 + 100*100)
= 111.8

Here, we use the torso length as a reference for scaling. In theory you can use any part of the body (shoulder, arm, leg) as a reference length.

Normalized by taking the neck position as 0,0 and scale all other points accordingly based on the torso length

Data Normalization

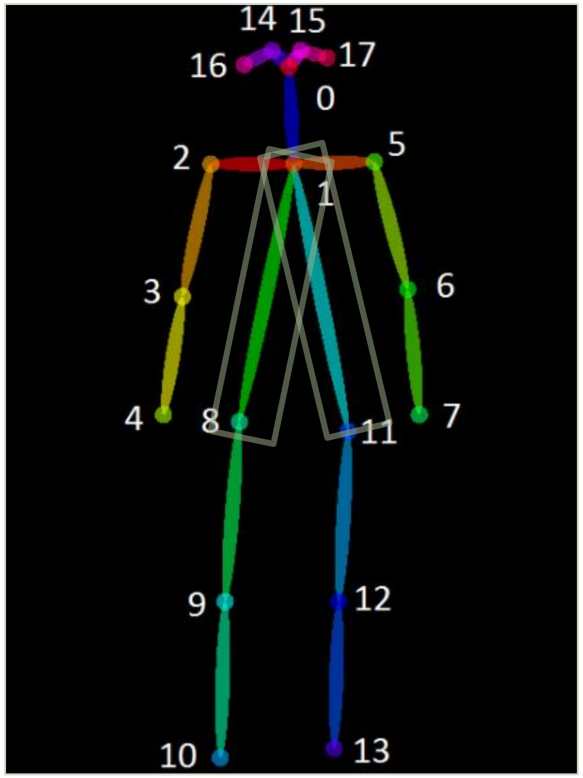


After scaling. Now, regardless of the scale, the final processed joint coordinates are the same for the same exact pose

Data Normalization

- We use some reference sizes also:
  - Torso height =
    - length(point 1 to point 8) **OR**
    - length(point 1 to point 11)

- Then do the following to all points in 1 skeleton frame:
  - P_i .x = (P_i.x – Neck.x) / torso_height
  - P_i .y = (P_i.y – Neck.y) / torso_height



Data Normalization

- Should we always normalize all skeleton frames from OpenPose?
  - It depends entirely on your task.

- For example:
  - Our task required us to determine the current action of the person, regardless where he/she is standing -> must normalize

  - if your task was to determine position of the person in the store then this task requires you to retain the exact screen coordinates of the key points -> do not normalize

Data Normalization

- What if?
  - We have an overhead camera in a MRT station, if we want to determine where a person is standing and whether his behaviour is suspicious, should we normalize the keypoints?

Recognizing Activities from Pose Estimations

OTHER TIME SERIES CLASSIFICATIONS

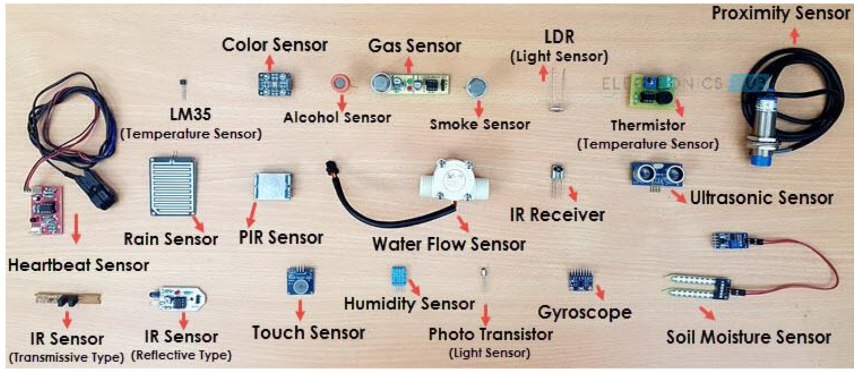
Each Time Series is a Sequence

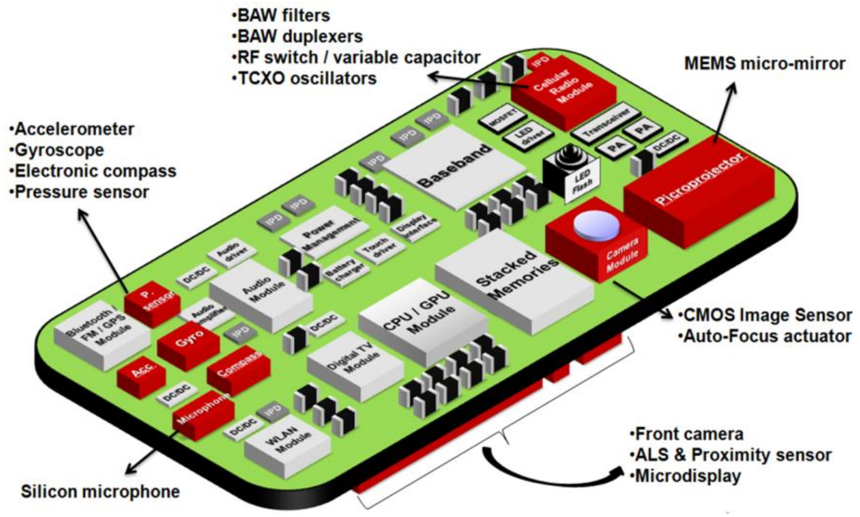
- The most well-researched Deep Learning architecture for recognizing sequences:
  - Recurrent Neural Networks (including GRUs and LSTMs)

- Technically, RNNs can be applied to any time series:
  - Data from sensors / readings
  - Data from financial investments / instruments

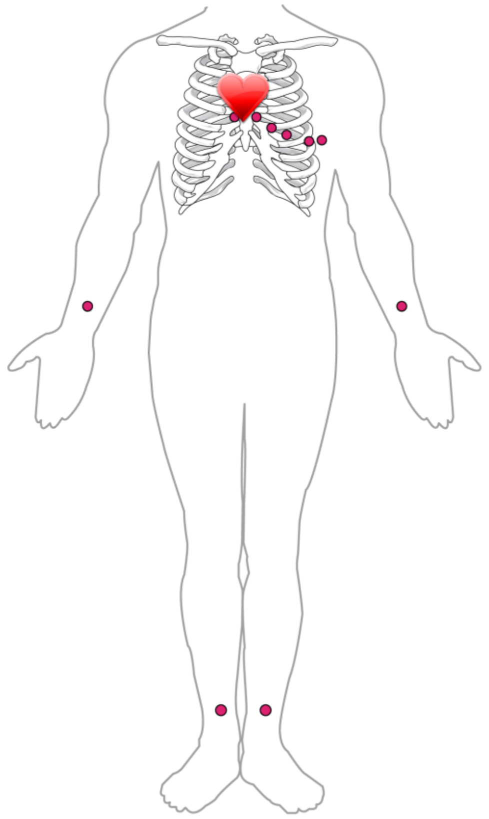
Each Time Series is a Sequence

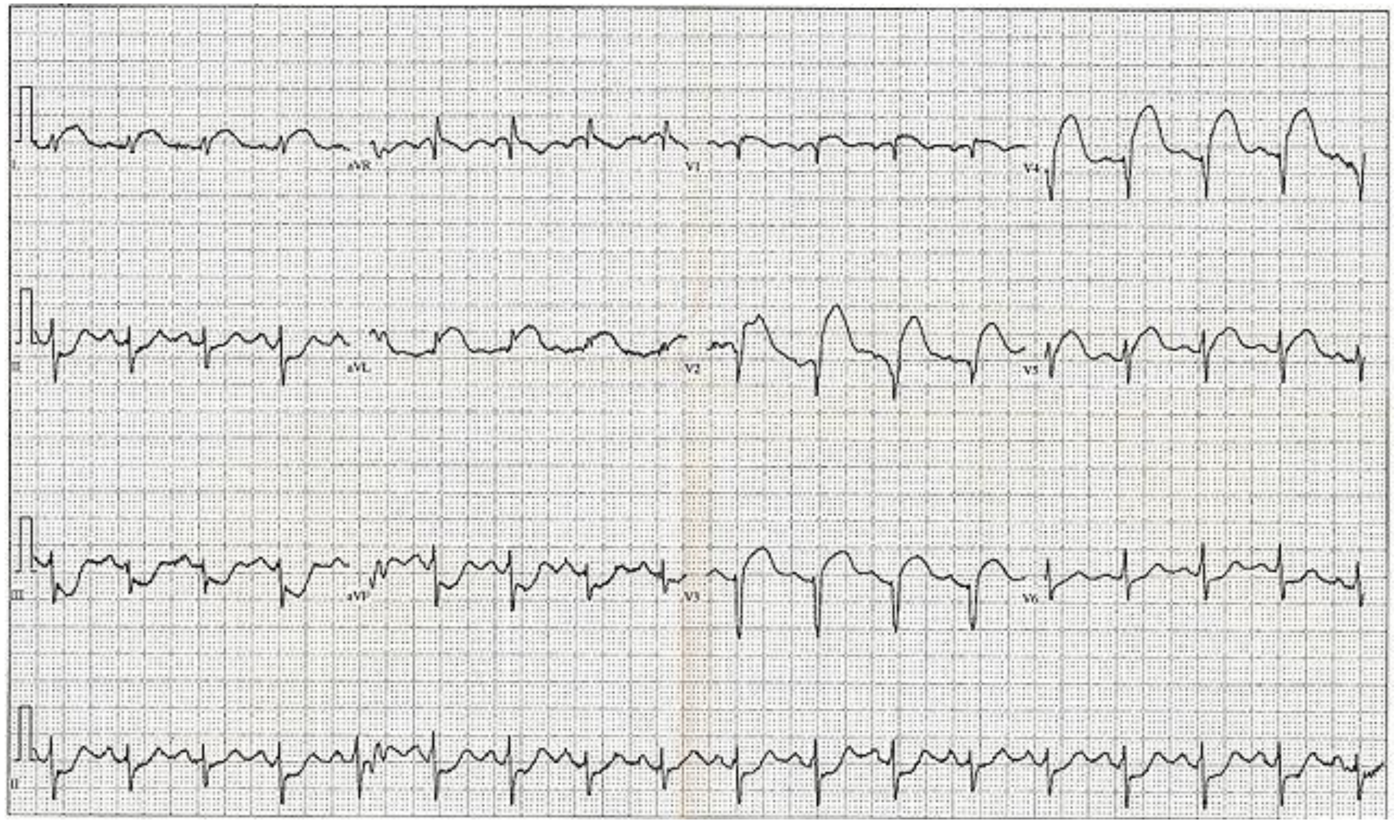
- Data that considered time-series:
  - Illumination level
  - Colour
  - Heat / Temperature / Humidity
  - Rainfall
  - Pollution Levels
  - Noise Level
  - Oxygen Level
  - Vibration
  - Speed
  - Orientation of device
    (with reference to gravity)
  - Acceleration
  - Magnetic facing
  - *and many others…*



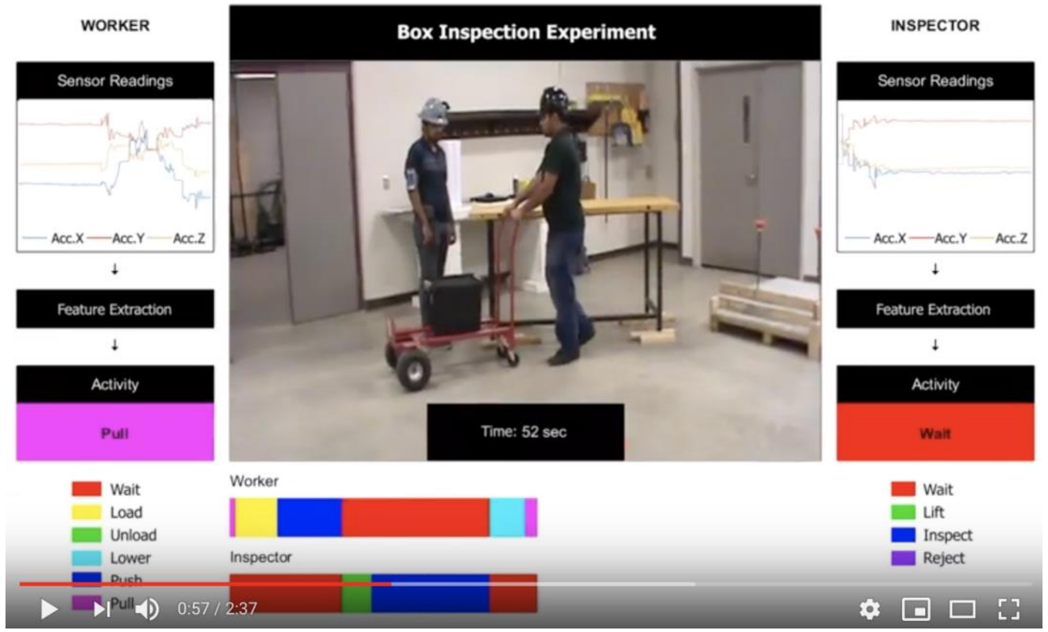


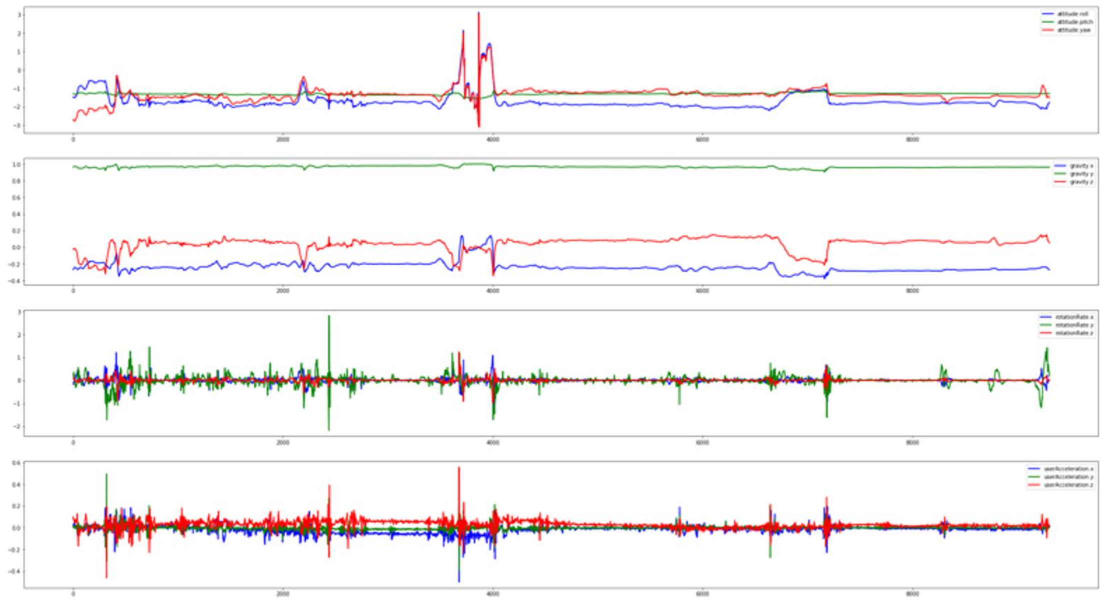
ECG Leads Provide Electrical Signals Over Time





Gyroscope / Accelerometer





https://www.youtube.com/watch?v=9tTC5-KrpE0

More Use Cases

- Energy consumption forecast important to energy suppliers => helps to plan for capacity, minimize risk, scheduling maintenance to minimize impact, and maximize use of power plants (reduce overproduction/ underproduction)

- LSTMs can be used to help forecast the consumption as a time-series and performs better than traditional methods

https://arxiv.org/pdf/1909.08182.pdf

More Use Cases

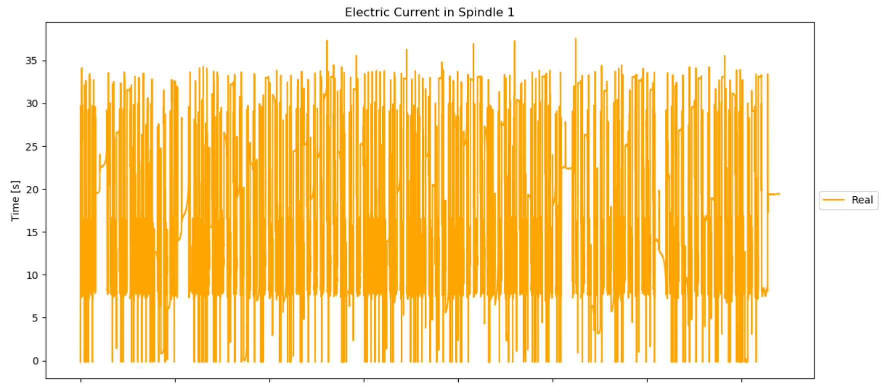
- Every airplane has a QAR (Quick Access Recorder) that records up to 2000 different parameters

- An LSTM can be used to predict hard landings and warn pilots in advance to take corrective action.

https://www.researchgate.net/publication/329410708_Aircraft_Hard_Landing_Prediction_Using_LSTM_Neural_Network

More Use Cases

- Condition-Based Monitoring using machine's electrical current + temperature to inform abnormalities to operators so the maintenances can be scheduled before the machine goes down.



https://towardsdatascience.com/lstm-neural-network-for-industry-4-0-condition-based-monitoring-afee73c7752c

Other Applications

- Robot Control
- Speech Recognition
- Rhythm Learning
- Music Composition
- Handwriting Recognition
- Human Activity Recognition
- Sign Language Translation
- Gene Sequence Classification
- Business Process Prediction
- Medical / Healthcare Pathways Prediction
- Semantic Parsing
- Airport Passenger Management
- Short-term Traffic Forecasting

Other References

- Approaches for Sequence Classification on Financial Time Series Data (talks about SVM, LSTM and CNN)
  - https://www.youtube.com/watch?v=flMCYqIn3eg

Summary

- Human Pose Estimation
- Recurrent Neural Networks for Time-Series Classification
- What Deep Learning Algorithms Can We Use?
- Human Activity Recognition
- Dataset and the Importance of Data Normalization