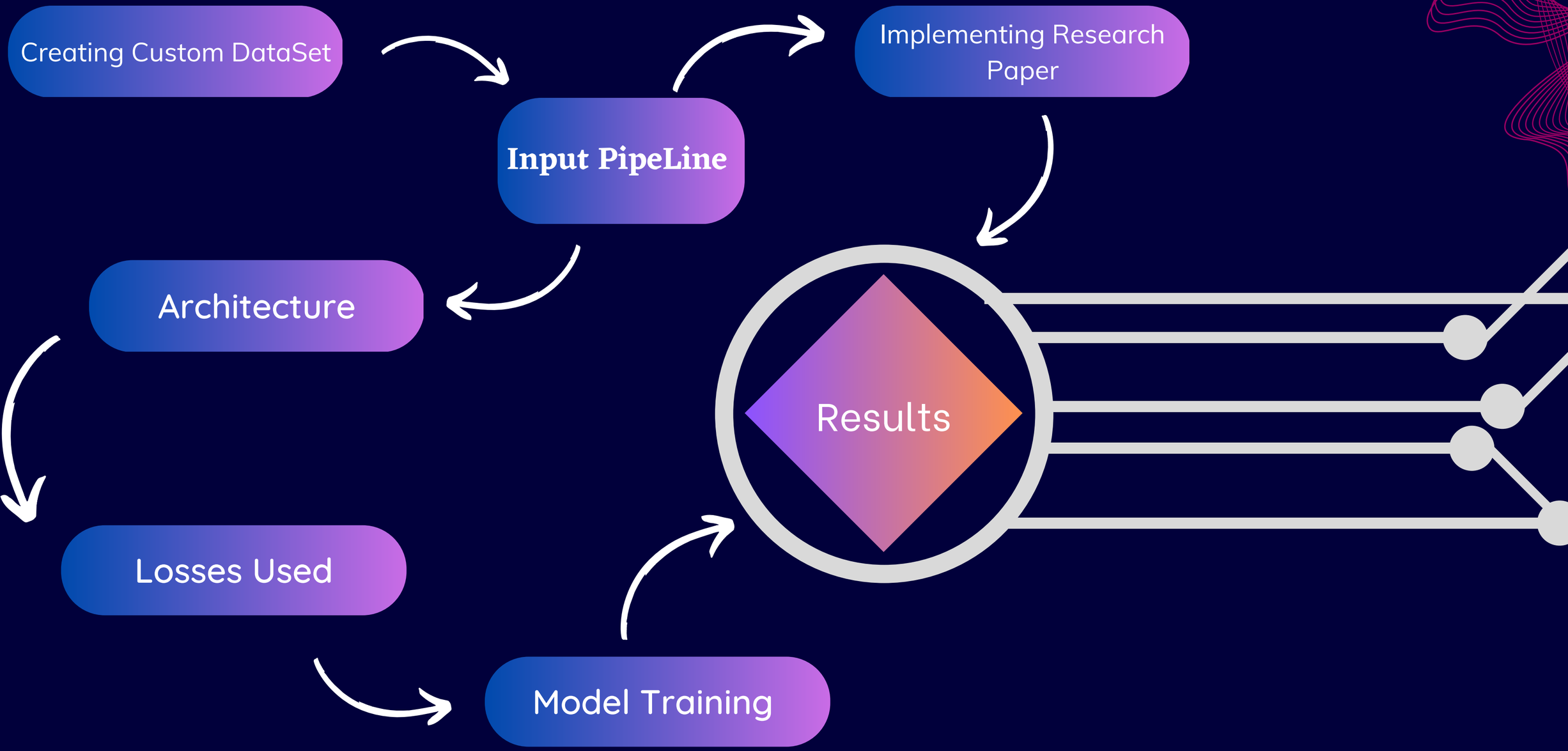
GENERATE NEW HUMAN POSES USING DEFORMABLE GANS

Using Deformable GANs

PRESENTED TO
MARS

PRESENTED BY
Adamya Vashisth
Ahmed Rocky Sakia
Himak Garg
Priyanshu maurya


Creating Custom DataSet
Input PipeLine
Implementing Research Paper
Architecture
Results
Losses Used
Model Training

Requirements -

- Thousands of images
- Each person with different poses
- Estimated pose for each image

So, We use solo dance short videos available on social media and youtube. Change the Frame to 6f/sec. And resolution to 244p.

For pose estimation, we use the pre-trained model **Movenet**.
Which gives the 17 points of human body joints.

Creating Custom Dataset

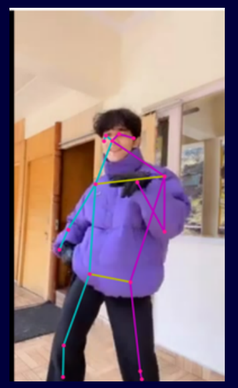
- Input Image
- Input Image pose
- Target Image {For training}
- Target image pose

Input - pipeline

image

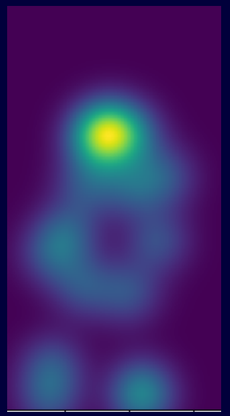
Resize to (128,72)

Movenet
Pose Estimator



Heat Map Function

Input image



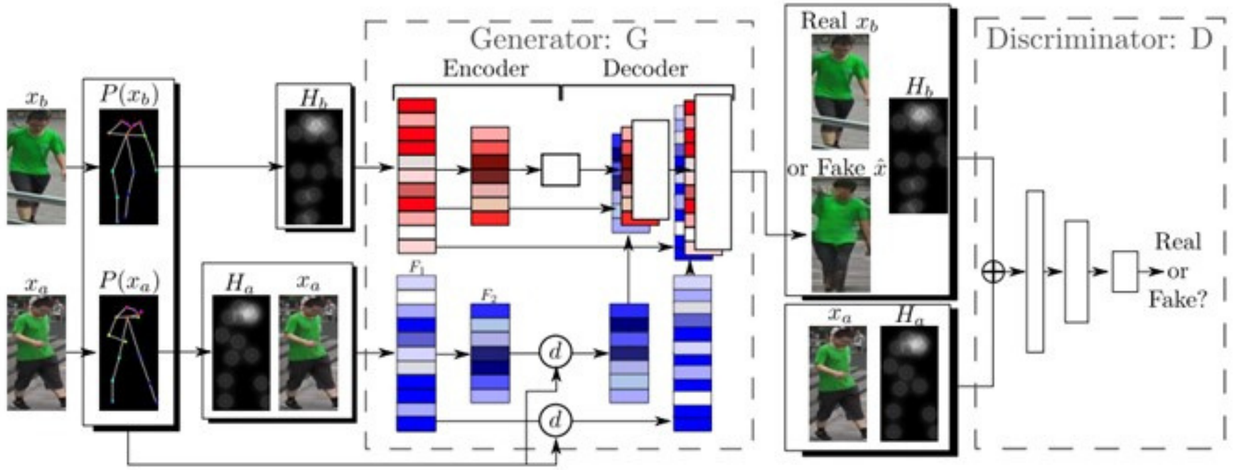
InputImage pose

Same for Target Image

Implementing Research Paper

Research Paper Implementation

Schematic of Architecture



H_j - Tensor containing information about pose

X_b – Image of target/input

$P(X_b)$ – Pose estimated from pose estimator

Red Layers – Feature maps developed directly from H_b

Blue Layers – Defomed tensors fed from Xa and Ha using which affine transformation is performed to pass texture level information

White Layers– Up convolutional layers

Research Paper Implementation

Problems Encountered

1) Setting up environment –
   a) Keras-contrib was not available through pip and had to be installed from different repo.
   b) Non availability of tensorflow=1.5.0 in windows. We were fortunate to find it in MacOS. Found a single repo for tensorflow=1.5.0 but produced a lot of errors in windows.

2) Pose estimation -
   a) The pose estimation was based on the paper "Realtime Multi-Person Pose Estimation". Computation of the coordinates is computationally expensive.

3) Training –
   a) Training is both expensive in terms of computation as well as memory. It stores immediate pose for better performance of the model.

4) Testing –
   a) Could not calculate the Inception score as it was becoming computationally expensive.

Research Paper Implementation

Training

We trained on a combination of market dataset and custom dataset for 10 epochs which showed appreciable results

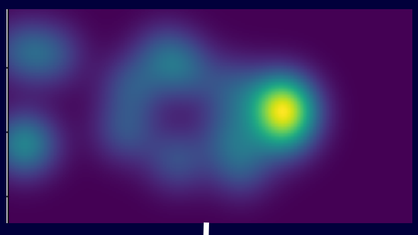
Batch size = 4

Testing

We tested on our custom dataset for based on weights of market dataset. So results were a little biased. Size : 128X64

Creating our own Deformable Gan Using Adversairal loss and NN loss

Generator



Conv2d(1,3,2)

Conv2d(3,6,2)

ConvTranspose2d(6,6,2)

ConvTranspose2d(6,9,2)

Conv2d(4,16,3)

Conv2d(16,32,3)

Conv2d(32,48,3)

Conv2d(48,64,3)

ConvTranspose2d(64,48,3)

ConvTranspose2d(48,32,3)

ConvTranspose2d(32,16,3)

ConvTranspose2d(16,4,2)

ConvTranspose2d(4,3,2)

Concate

Conv2d(12,10,3)

Conv2d(10,8,3)

Conv2d(8,6,3)

ConvTranspose2d(6,4,3)

ConvTranspose2d(4,3,3)

ConvTranspose2d(3,3,3)

Conv2d(3,64,3, bias=False)

Conv2d(64,64*2,5, bias=False)

Conv2d(64*2,1,10, bias=False)

Flatten

Linear

Linear

Linear

Discriminator

Adversarial Loss

$$L_{3D-GAN} = logD(x) + log(1 - D(G(z)))$$

Nearest Neighbor Loss

$$L_{NN}(\hat{x}, x_b) = \sum_{\mathbf{p} \in \hat{x}} min_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} ||g(\hat{x}(\mathbf{p})) - g(x_b(\mathbf{q}))||_1$$

g(x(p)) is a vectorial representation of a patch around point p in image x, obtained using convolutional filters

Loss Functions

Input image

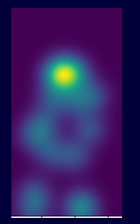
Discriminator

y'

Adv Loss

NN Loss

Generator



y

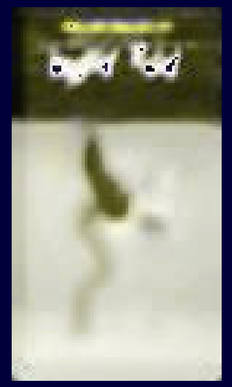
Adv Loss

Training

Target Image

Generated Image

After 2 epochs



Training of our model cost very high computation and time
Since we don't have a such capable machine.
So, we train it on our laptop for many hours and did it for up to 2 epochs.
And you can see Model is doing quite well just after 2 epochs

Result