CV PROJECT

- Team ID: 15

- Team Members:
  - Sai Jashwanth (20171178), Sai Soorya Rao (20171052), Raviteja (20171067)
  - TA Mentor : Pranay Gupta

- Project ID, TITLE : 19, **View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition**

ABSTRACT

- One of the famous problems in Computer Vision is to determine the pose/action of a human based figure given the data.

- The data for this problem can be of two types RGB-D or 3D Skeleton data.Many Papers were written taking RGB-D data into consideration.This paper discusses tackling the problem with 3D skeleton data.

- Skeletons have merits of being robust to appearances, surrounding distractions and variation of viewpoints.

ABSTRACT

- Key challenges in action Recognition: Large diversity of viewpoints of the captured human action data.

- To tackle this challenge, this paper introduces a novel view adaptation scheme which automatically determines the virtual observation viewpoints over the course of an action in a learning based data driven manner.

- They designed two view adaptive neural networks, i.e., VA-RNN and VA-CNN. For each network, a novel view adaptation module learns and determines the most suitable observation viewpoints, and transforms the skeletons to those viewpoints for each sample.

ABSTRACT

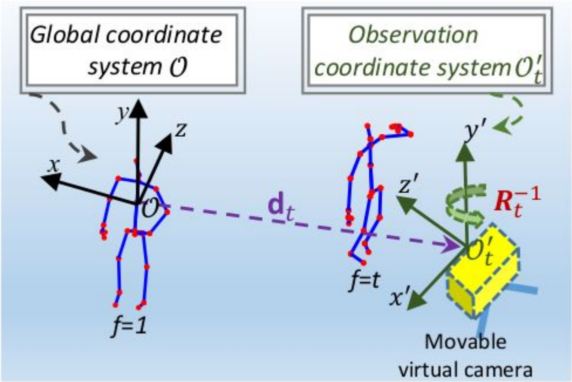
- This enables the classification module (main classification network) to “see” the skeleton representation under new viewpoint for efficient recognition.

- With the objective of maximizing the recognition performance, the view adaptation subnetwork and the main classification net, which comprise the entire network is trained from end to end.

- The continuity of an action is maintained besides effectively regulating the skeletons to more consistent view points.

VIEW ADAPTIVE MODELLING AND PROBLEM FORMULATION

- The VA module automatically determines the virtual observation viewpoints and outputs a set of transform parameters $\mathcal{T}_t$ for each time/frame t (or $\mathcal{T}$ for a sequence).



3D Skeletons correspond to Global coordinate system with origin translated to body center

O(t) is the observed coordinate system (output of VA subnet)

PROBLEM FORMULATION

Given a skeleton sequence S under the global coordinate system O, the jth skeleton joint on the t'th frame is denoted as $\mathbf{v}_{t,j} = [x_{t,j}, y_{t,j}, z_{t,j}]^{\mathrm{T}}$, where $t \in (1, \cdots, T)$, T denotes the total number of frames a sequence, $j \in (1, \cdots, J)$, J denotes the total number of skeleton joints in a frame. The set of joints in the t'th frame is denoted as $V_t = \{\mathbf{v}_{t,1}, \cdots, \mathbf{v}_{t,J}\}$.

$$\mathbf{v}'_{t,j} = [x'_{t,j}, y'_{t,j}, z'_{t,j}]^{\mathrm{T}} = \mathbf{R}_t(\mathbf{v}_{t,j} - \mathbf{d}_t). \quad \mathbf{R}_t = \mathbf{R}^x_{t,\alpha}\mathbf{R}^y_{t,\beta}\mathbf{R}^z_{t,\gamma},$$

$$\mathcal{T}_t = \{\alpha_t, \beta_t, \gamma_t, \mathbf{d}_t\}$$

The skeleton representation, $V'_t = \{\mathbf{v}'_{t,1}, \cdots, \mathbf{v}'_{t,J}\}$ under new observation coordinate.

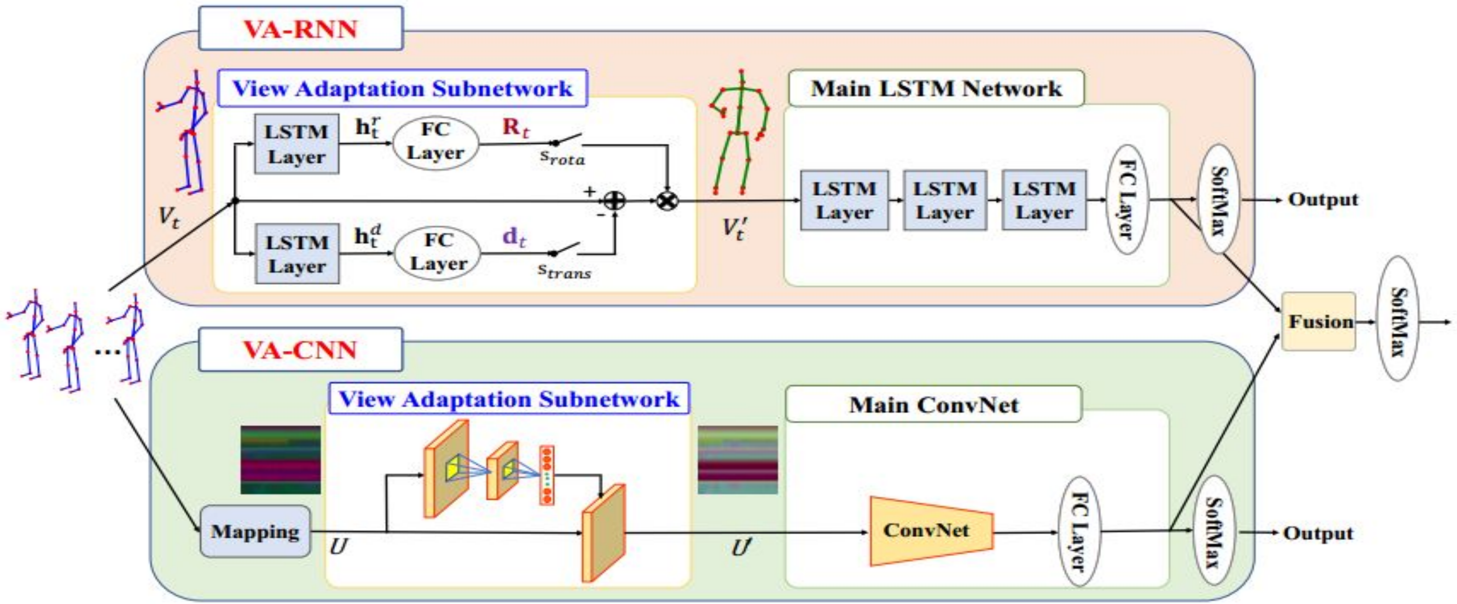
PROBLEM FORMULATION

- All the skeleton joints in the t-th frame have the same transformation parameters. View points can be different for different frames.

- Key problem becomes how to determine the viewpoints(Transformation parameters) of the movable virtual camera.

$$\mathbf{R}^{x}_{t,\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_t) & \sin(\alpha_t) \\ 0 & -\sin(\alpha_t) & \cos(\alpha_t) \end{bmatrix} \quad \mathbf{R}^{y}_{t,\beta} = \begin{bmatrix} \cos(\beta_t) & \sin(\beta_t) & 0 \\ -\sin(\beta_t) & \cos(\beta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}^{z}_{t,\gamma} = \begin{bmatrix} \cos(\gamma_t) & 0 & -\sin(\gamma_t) \\ 0 & 1 & 0 \\ \sin(\gamma_t) & 0 & \cos(\gamma_t) \end{bmatrix}$$

WORKFLOW


VA-RNN
View Adaptation Subnetwork
LSTM Layer
h_t^r
FC Layer
R_t
s_{rota}
V_t
LSTM Layer
h_t^d
FC Layer
d_t
s_{trans}
V_t'
Main LSTM Network
LSTM Layer
LSTM Layer
LSTM Layer
FC Layer
SoftMax
Output
Fusion
SoftMax
VA-CNN
View Adaptation Subnetwork
Mapping
U
U'
Main ConvNet
ConvNet
FC Layer
SoftMax
Output

VIEW ADAPTIVE NEURAL NETWORKS

- ➢ The paper suggested two view adaptive networks : VA-RNN and VA-CNN.By using these two architectures, we combine the scores from each of the networks to estimate the final parameters.

- ➢ Note that any one architecture would have also suffice to get decent results.Experimental results suggested that using only VA-RNN achieved upto 75% accuracy and using VA_CNN they achieved upto 85% accuracy.

- ➢ We, in our project implemented VA-CNN for the view adaptation sub module.

Mapping Skeletons to Image:

- In this sub network, we map sequence of skeletal data into a image map to facilitate the spatial-temporal dynamics.We refer to this as skeleton map from now.

- In this skeleton map, each row is a frame and each cell contains a single skeleton joint, the RGB of each cell is the (X,Y,Z) coordinate of the skeleton joint.Then we convert each pixel according to the following formula to get into range (0-255).

$$\mathbf{u_{t,j}} = floor(255 \times \frac{\mathbf{v_{t,j}} - \mathbf{c_{min}}}{c_{max} - c_{min}}),$$

Mapping Skeletons to Image:

$V_{t,j}$ = j^{th} joint of the skeleton in the t^{th} frame.

C_{min} = min value of all the values in the skeleton map.

C_{max} = max value of all the values in the skeleton map.

Floor is the Greatest Integer Function.

View adaptation subnetwork-Outline:

On observing from new observation point which is obtained from the sub network, We transform $v_{t,j}$ to $v'_{t,j}$ by using the following formula.

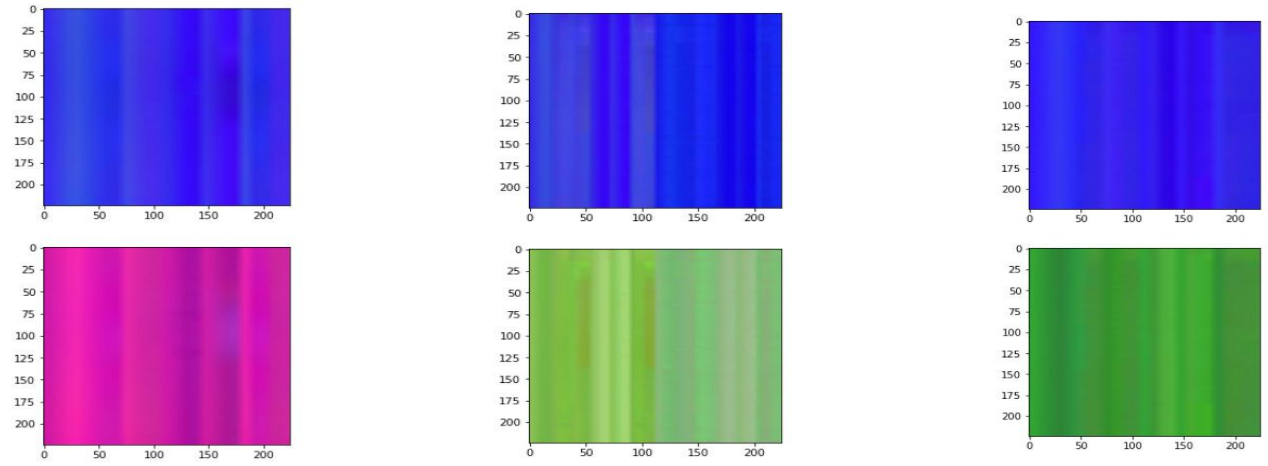
$$\mathbf{v}'_{t,j} = [x'_{t,j}, y'_{t,j}, z'_{t,j}]^{\mathrm{T}} = \mathbf{R}_t(\mathbf{v}_{t,j} - \mathbf{d}_t).$$

Using these two equations,Each pixel in the updated skeleton map is calculated as:

$$\begin{aligned}\mathbf{u}'_{\mathbf{t,j}} &= 255 \times \frac{\mathbf{v}'_{t,j} - \mathbf{c}_{\mathbf{min}}}{c_{max} - c_{min}} \\ &= \mathbf{R}_{t,j}\mathbf{u}_{t,j} + 255 \times \frac{\mathbf{R}_{t,j}(\mathbf{c}_{min} - \mathbf{d}_{t,j}) - \mathbf{c}_{min}}{c_{max} - c_{min}}.\end{aligned}$$

View adaptation subnetwork-Outline:

Input Skeleton map(above) & Transformed skeleton map(below)



View adaptation subnetwork-Working:

- The CNN-based view adaptation network is designed to learn and determine the observation viewpoint of each skeleton sequence and performs the transform on the skeleton map.

- We build the view adaptation subnetwork by stacking some convolutional layers and a fully connected layer to regress the transformation parameters, i.e., α, β, γ for $R_{t,j}$ and d_t .
    - More details of architecture is discussed at ending

View adaptation subnetwork-Working:

- In theory, it is ideal to regress these 6 parameters for each frame.i.e, estimating 6xT parameters for T-width skeleton image.

$$\mathcal{T}_t = \{\alpha_t, \beta_t, \gamma_t, \mathbf{d}_t\},$$

- But in practice, it is found that 6x1 parameters i.e, 6 parameters for the given T sequence of frames is found to give superior performance. This most probably happened because there are fewer parameters to learn.
- Later this transformed skeleton image map is fed to the Main classification ConvNet.Here we used pretrained 'resnet50'.

Data loading / Pre Processing from NTU Dataset

- ➢ There are 1 or more subjects in each frame and each subject has 25 joints according to NTU Dataset.
- ➢ We consider a maximum of 2 subjects.
- ➢ Skeleton seq is mapped to image map to facilitate the process. Image map has 3 channels.
- ➢ Columns in image map correspond to different frames. Rows correspond to different joints.
- ➢ This is employed in collate_fn of DataLoader.

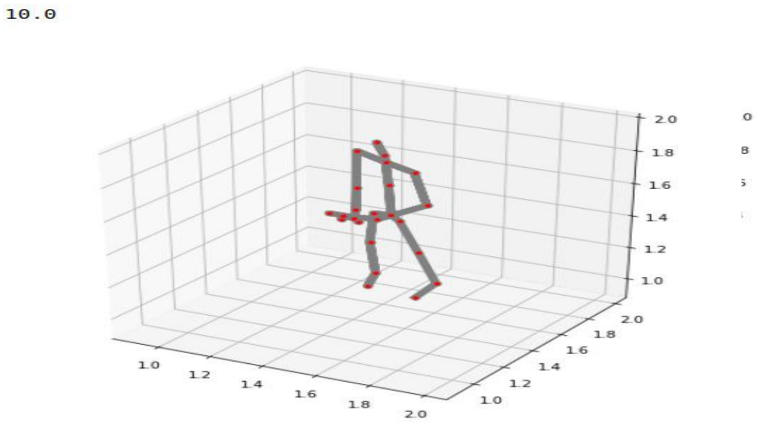
PREPARATION:

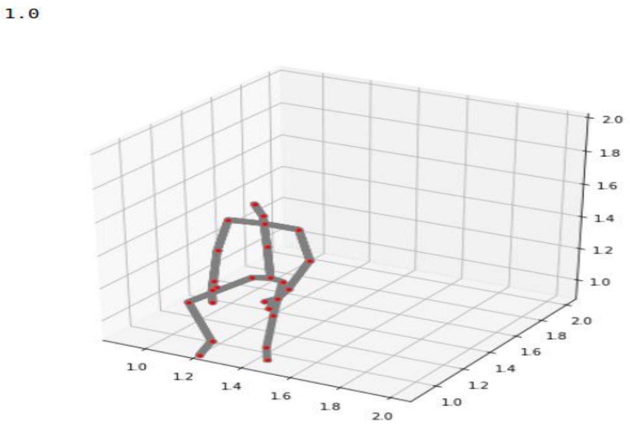
- ➢ From .skeleton files of dataset, we retrieved the data in the form of an np array.
- ➢ Then converted it into (no. of videos * no. of frames * 50* 3). 50 corresponds to 2 subjects in the frame where each subject has 25 joints.
- ➢ Each frame in the data is considered as a row in the input and each column in the cell is one of the skeleton joint.
- ➢ Thus we get a (no.of.frames)x50x3 image and min-max values of this imagemap.
- ➢ We then resize it to (224x224x3) which acts as input to our network.

NTU - DATA LOADING

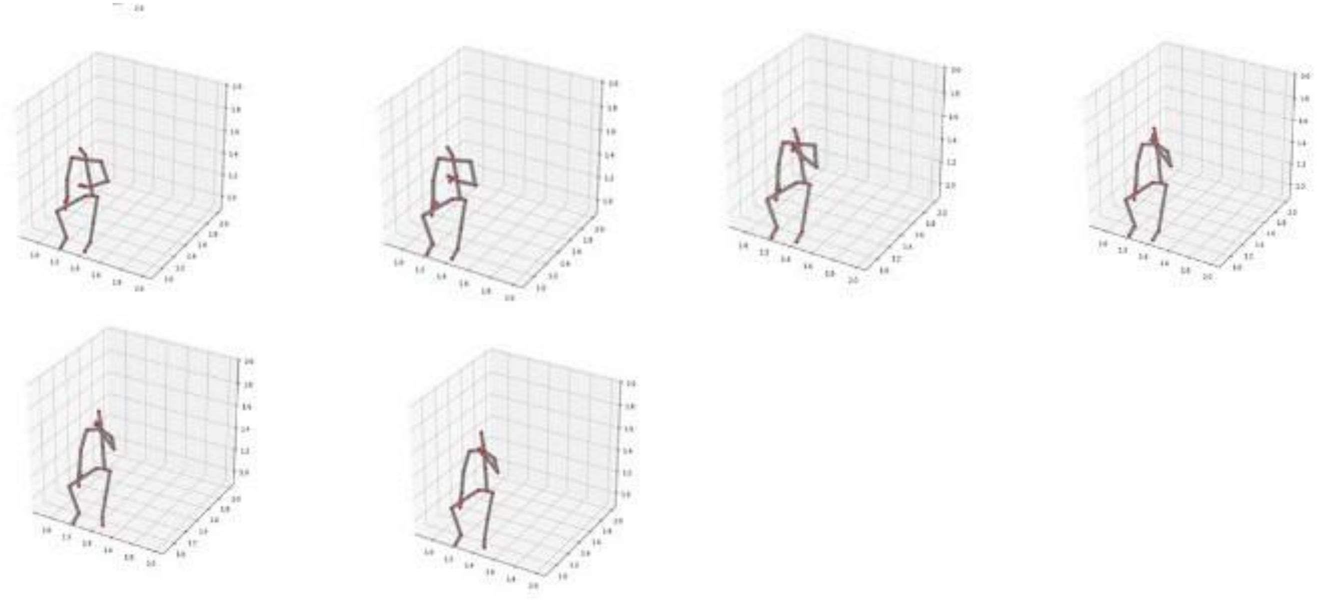
➢ Thus we can view the image map as spatially diverse across columns and temporally across rows.

VISUALIZATION:



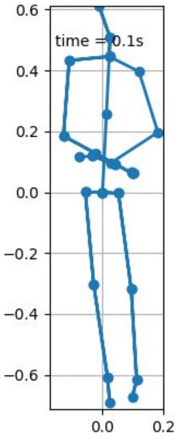


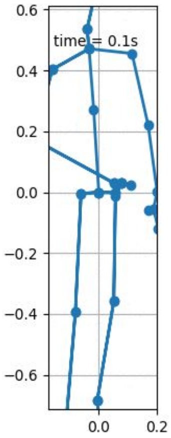
VISUALIZATION

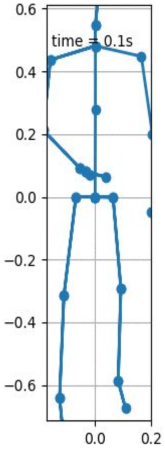


NTU-Data Visualization

GIF Data Drinking created on our dataset using 30 frames







ARCHITECTURE:

- We build this subnet by stacking 2 conv. Layers and an FC layer. Activation layer is ReLu.
- Each FC layer is followed by Batch Normalization Layer.
- Initial weights of FC are set to zero.
- Main Classification ConvNet is pretrained Resnet50 model.

WORKING:

- Skeleton map is obtained by processing and converting the raw 3D skeleton data to rgb format.
- Once we obtain the 6 parameters from the VA-CNN, we call transform function which takes (skeleton map, c_{min} , c_{max}) as parameters and outputs the transformed skeleton map.

IMPLEMENTATION

➢ This transformed skeleton map is fed to main classification network and is trained end-to-end.

PERFORMANCE:

```
Validation: Epoch-6   Accuracy 81.1471

Epoch-7    50 batches    loss 0.1290
Epoch-7   100 batches    loss 0.4327
Epoch-7   150 batches    loss 0.1484
Epoch-7   200 batches    loss 0.4848
Epoch-7   250 batches    loss 0.3584
Epoch-7   300 batches    loss 0.2030
Epoch-7   350 batches    loss 0.1597
Epoch-7   400 batches    loss 0.2123
Epoch-7   450 batches    loss 0.4386
Epoch-7   500 batches    loss 0.1202
Epoch-7   550 batches    loss 0.1780
Epoch-7   600 batches    loss 0.0983
Epoch-7   650 batches    loss 0.1361
Epoch-7   700 batches    loss 0.3368
Epoch-7   750 batches    loss 0.1412
Epoch-7   800 batches    loss 0.0886
Epoch-7   850 batches    loss 0.1109
Epoch-7   900 batches    loss 0.2378
Epoch-7   950 batches    loss 0.1656
Epoch-7   1000 batches   loss 0.1808
Epoch-7   1050 batches   loss 0.1127
Epoch-7   1100 batches   loss 0.6882
Epoch-7   1150 batches   loss 0.4002

Validation: Epoch-7   Accuracy 80.1496
```

```
Validation: Epoch-3   Accuracy 75.4613

Epoch-4    50 batches    loss 0.2950
Epoch-4   100 batches    loss 0.2842
Epoch-4   150 batches    loss 0.2306
Epoch-4   200 batches    loss 0.2707
Epoch-4   250 batches    loss 0.3937
Epoch-4   300 batches    loss 0.3272
Epoch-4   350 batches    loss 0.3434
Epoch-4   400 batches    loss 0.3366
Epoch-4   450 batches    loss 0.7987
Epoch-4   500 batches    loss 0.3925
Epoch-4   550 batches    loss 0.2808
Epoch-4   600 batches    loss 0.3678
Epoch-4   650 batches    loss 0.3192
Epoch-4   700 batches    loss 0.2848
Epoch-4   750 batches    loss 0.3639
Epoch-4   800 batches    loss 0.7104
Epoch-4   850 batches    loss 0.5250
Epoch-4   900 batches    loss 0.5371
Epoch-4   950 batches    loss 0.1982
Epoch-4   1000 batches   loss 0.3675
Epoch-4   1050 batches   loss 0.3456
Epoch-4   1100 batches   loss 0.4630
Epoch-4   1150 batches   loss 0.3713

Validation: Epoch-4   Accuracy 72.3691
```

```
Validation: Epoch-1   Accuracy 59.5511

Epoch-2    50 batches    loss 0.8401
Epoch-2   100 batches    loss 0.6783
Epoch-2   150 batches    loss 0.9141
Epoch-2   200 batches    loss 0.8566
Epoch-2   250 batches    loss 0.4557
Epoch-2   300 batches    loss 0.5108
Epoch-2   350 batches    loss 0.4814
Epoch-2   400 batches    loss 1.1379
Epoch-2   450 batches    loss 0.9929
Epoch-2   500 batches    loss 0.5302
Epoch-2   550 batches    loss 0.5805
Epoch-2   600 batches    loss 0.4376
Epoch-2   650 batches    loss 0.5457
Epoch-2   700 batches    loss 0.6542
Epoch-2   750 batches    loss 0.6318
Epoch-2   800 batches    loss 0.4342
Epoch-2   850 batches    loss 0.5445
Epoch-2   900 batches    loss 0.6484
Epoch-2   950 batches    loss 0.5312
Epoch-2   1000 batches   loss 0.6563
Epoch-2   1050 batches   loss 0.3708
Epoch-2   1100 batches   loss 0.4782
Epoch-2   1150 batches   loss 1.2062

Validation: Epoch-2   Accuracy 78.0050
```