README

Pose prediction as a multi task approach

Anmol Prasad M.S. C.S. EPFL Masters Project

This is the code for the project Pose prediction as a Multi-Task approach. It contains the files for creation, training and testing of the models as well the notebook with pre-processing required for the data.

Preprocessing

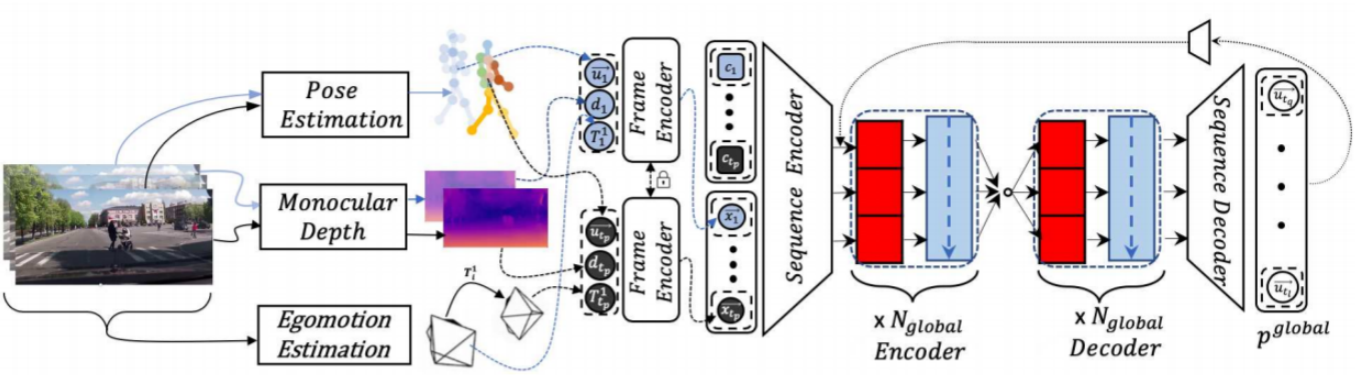
The JAAD images are first masked using the bounding box to contain 1 pedestrian in 1 frame as mentioned in the original Disentanglement paper. Then openpose/openpifpaf is used to generate the keypoints data. This data is then recombined into length 16 sequences in which all the images have the detected keypoints. Preprocess.py contains the code for keypoint filtering and selection from the raw data for both openpose and openpifpaf.
Additionally, depth and egomotion information is also extracted from the original images using the SC-sfmlearner model (https://github.com/JiawangBian/SC-SfMLearner-Release). Code is provided in preprocess_depth.py and preprocess_egomotion.py.
The keypoint data and depth is stored in csv files.

Metrics

We use ADE, FDE as our metric which measures the average displacement between keypoints and the average displacement between the final frames of the sequence.
The Utils.py contains the data loader code for different tasks as well as the metrics (ADE/FDE/VIM/VAM). It also contains the code for visualizing the keypoint data for each of SoMoF benchmark, openpose and openpifpaf keypoints.
All raw and preprocessed data is available at the following drive link ()

Models

I tried to implement the disentanglement model as described in the paper (Disentangling Human Dynamics for Pedestrian Locomotion Forecasting with Noisy Supervision(Mangalam et Al)



This work aims to disentangle motion into global and local streams, as they hypothesized that using separate modules will allow for a better prediction of motion, since the overall motion of the body is quite different from the movements of the individual joints.(Eg. The motion of the elbow while walking is oscillating as opposed to the overall single direction motion of the person). They first extract the pose, monocular depth and egomotion information from the sequences using pre-trained networks. The pose information is then passed to a denoising autoencoder trained on complete poses. This is used to recover missing and erroneous joints by replacing missing joints and joints with confidence scores lower than a threshold(0.25) by the predicted clean joints. The clean pose is then split into global and local streams, which is the disentanglement. The disentangled streams are then processed independently by the sequence prediction heads.
The disengagement is specifically done as follows:

- Global Stream: Neck Joint Sequence $(u_j^{\tilde{i}}, v_j^{\tilde{i}})_{j=t-t_p+1}^{t}$
- Local Stream = Pose - Neck Joint Sequence

$$\vec{w}_t^{\,i} = (u_t^i, v_t^i) - (u_t^{\tilde{i}}, v_t^{\tilde{i}})$$
$$\forall i \in \{1, \ldots, d\}, i \neq \tilde{i}, t \in \{t - t_p + 1, \ldots, t\}$$

The neck joint sequence is used as the global representation since it is the most commonly appearing joint in the dataset and encodes the global trajectory well. Along with the depth and egomotion estimates, it is passed to the global stream prediction head to predict the residual while the local stream is encoded into a latent dimension using a spatial encoder before passing to the local prediction head. The predictions are then merged to get the final pose estimate.
The main idea of the work is the disentanglement of the pose, which they show to give much better results experimentally than previous SOTA models.

The pose estimation, monocular depth and egomotion estimates are extracted using the preprocessing steps as described above. The implementation of other submodules are described below:

Denoising autoencoder

The idea of the denoising step is to ‘clean’ the pose before further processing to fix inconsistencies in predictions of joints and to recover joints where there is occlusion or confidence scores are low. The denoising autoencoder model aims to encode the pose using an overcomplete autoencoder which can be used to denoise the pose. The training is done on complete poses with a dropout of 0.5. I tried 2 different models: single pose at a time and sequence at a time. The single pose model is able to recreate the pose but not very accurately and results are not comparable to the original paper. The sequence at a time model did not work very well.

Spatial encoder

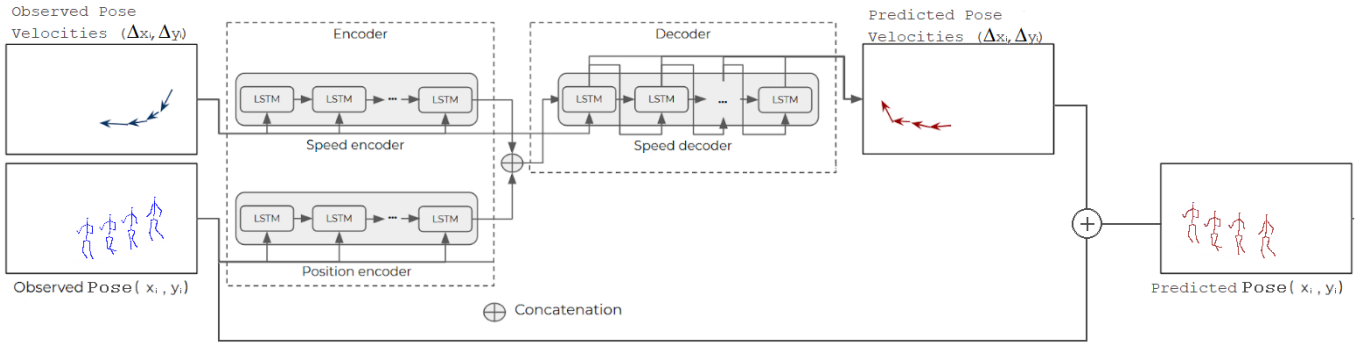
The authors also add a spatial encoding before the sequence prediction step to reduce the latent dimension of the pose or to change it to a sparse representation, as this may help in learning the representation better as observed from previous research. For learning this spatial encoder, I created 2 models in which the pose is encoded before passing to the LSTM encoder decoder. The output of the LSTM decoder is then recovered to the original pose shape using a spatial decoder. Additionally, in the second model there is a second head which tries to recreate the input pose as well with shared weights between the spatial decoders which aims to make the spatial encodings

better. In practice, these models did not improve the results and the models without any encoding performed faster and better.

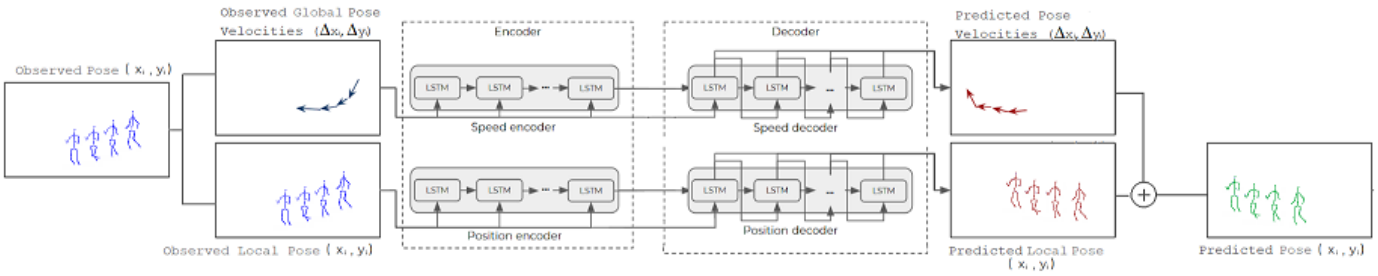
Sequence Prediction Network

There are models described below:

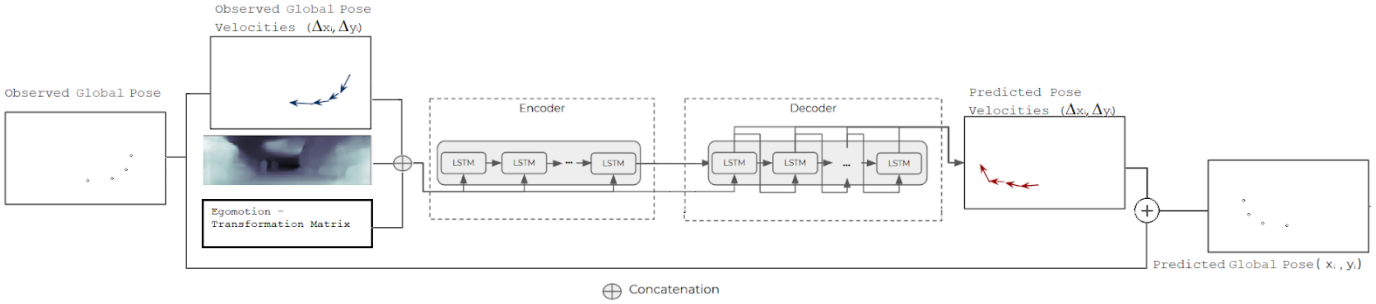
1. **MLP**
   The MLP is a fully connected model which only predicts the global trajectory. It gives decent results. The idea of this model was to establish a baseline.
2. **LSTM_Vel**
   The basic LSTM model uses velocities of the keypoints as inputs and predicts the future velocities using an LSTM encoder decoder. These are then added back to the original pose to predict the future poses. The size of the input poses vary according to the pose prediction module (openpose - 25 keypoints, openpifpaf - 17 keypoints)



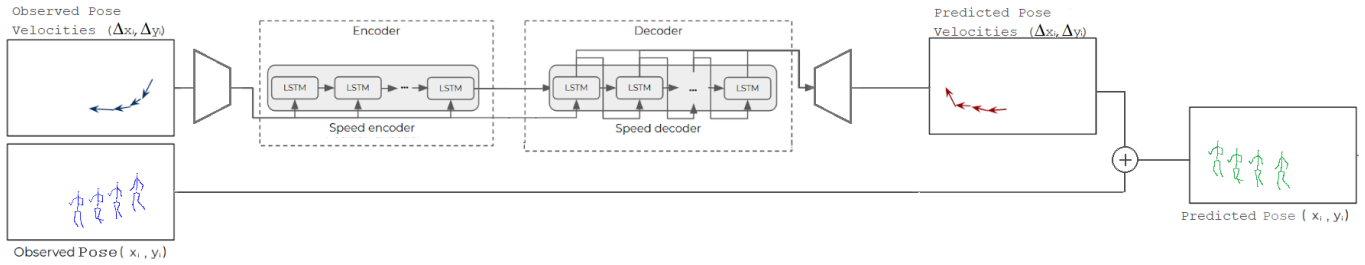
3. **PV_LSTM_DE:**
   The disentanglement model splits the pose into 2 streams. These streams are predicted independently and merged after prediction to get the final output.



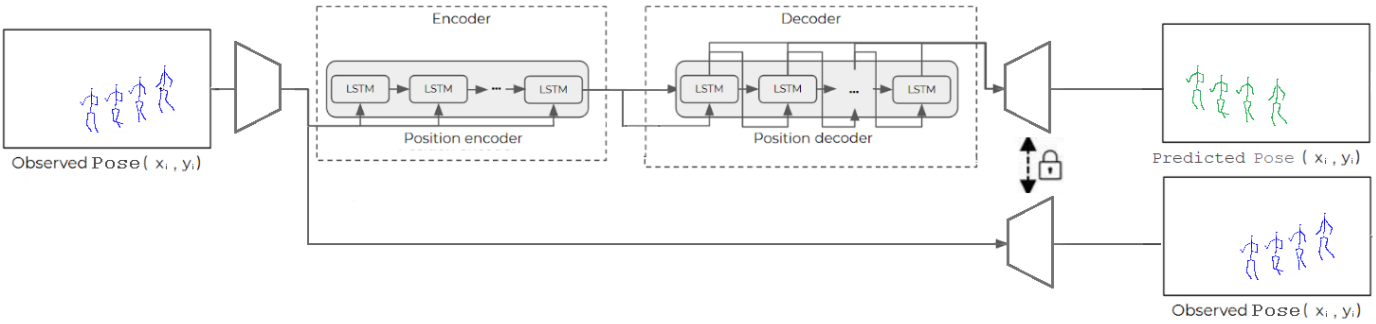
4. **glob_LSTM_DE**
   This model aims to improve the global predictions using depth and egomotion estimates along with the velocities. It is supposed to replace the global branch of the disentanglement model. The difference in terms of model was the parameter sizes (depth and transformation matrix was concatenated to the keypoint to change the input dimension to 15 instead of 2) and an additional fully connected layer to convert the predicted 15 dimension pose back to 2 dimensions(x,y). The depth and egomotion estimates are from the preprocessing steps. Results of this model were not much better than using just the velocities, but the model did not overfit compared to the model without depth/egomotion.



5. **LSTM_spat:** This model uses the spatial encoder to encode the input velocities/poses as described in the Spatial Encoder section to learn latent representations of the pose.



6. **LSTM_spat_rec:** This model adds another head to the LSTM_spat which uses the observed pose to train the spatial encoder/decoder



7. **LSTM_3dpw:** This model is the same as the LSTM_vel model with changed dimensions to predict 3D poses.
8. **LSTM_posetrack:** This model is the same as the LSTM_vel model with changed dimensions to predict poses with 28 keypoints and an additional head for the mask prediction.

Results

Pose Prediction

The model results are provided below. Training was done for 200 epochs using the Adam optimizer with a learning rate of 0.01(reduced on plateau). Results for the global trajectory prediction with and without the depth and egomotion information are provided as well.

Results on OpenPifPaf keypoints

- LSTM_vel with L2 Loss: ade_train: 8.3249 | ade_val: 17.7732 | fde_train: 10.5632 | fde_val: 32.8130
- LSTM_vel with L1 Loss: ade_train: 4.9769 | ade_val: 11.8759 | fde_train: 7.4425 | fde_val: 22.9868
- LSTM_vel with Teacher forcing and L1 loss: ade_train: 3.808 | ade_val: 12.709 | fde_train: 5.539 | fde_val: 24.336
- Spatial encoder (LSTM_spat_rec): ade_val: 40.84 | fde_val: 77.04
- Disentanglement (PV_LSTM_DE): ade_train: 7.17 | ade_val: 14.50 | fde_train: 11.39 | fde_val: 27.68

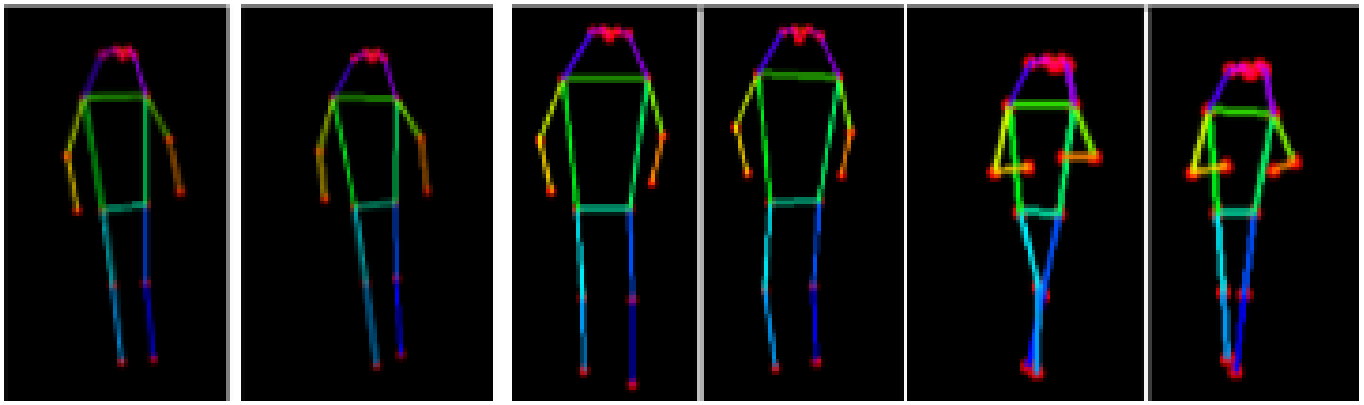


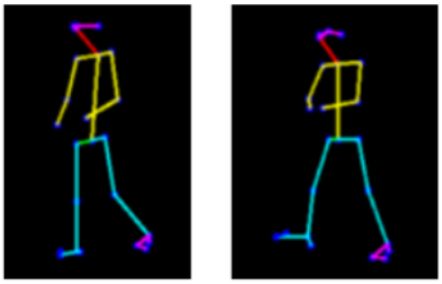
Fig. Expected and predicted outputs on openpifpaf keypoints

Results on Openpose keypoints

- No Disentanglement (LSTM_vel): ade_train: 35.23 | ade_val: 48.18 | fde_train: 43.67 | fde_val: 54.43

- Disentanglement (PV_LSTM_DE): ade_train: 26.95 | ade_val: 28.88 | fde_train: 40.68 | fde_val: 47.30

Disentanglement



Expected Output Predicted Output

Results on SoMoF (LSTM Posetrack and LSTM_3dpw): ade_train: 63.68 | ade_val: 87.47 | fde_train: 107.29 | fde_val: 148.93

The model performs well in sequences where there is less occlusion and lower movement. In cases where there is more movement, eg. in sport images, the results are not good as shown in the images below.

Fig. Results of keypoint detection at times t=t+1(top row) and t=t+5(bottom row) on posetrack

Fig. Results on images where there is restricted movement (Good predictions)

Fig. Results on images where there is more movement (Bad predictions)

Posetrack VIM	Method	Score	80 ms	160 ms	320 ms	400 ms	560 ms
	Velocity LSTM	1.00000	10.60706	19.39534	37.07473	45.70856	58.72602
Posetrack VAM	Method	Score	80 ms	160 ms	320 ms	400 ms	560 ms
	Velocity LSTM	1.00000	25.32510	41.33756	69.26933	80.76324	96.31478
3DPW VIM	Method	Score	100 ms	240 ms	500 ms	640 ms	900 ms
	Velocity LSTM	1.00000	21.62887	39.56685	75.64954	96.76794	142.21865

Global Trajectory Prediction

- MLP: ade_val_g: 16.41 | fde_val_g: 29.21
- LSTM_global: ade_val_g: 11.90 | fde_val_g: 16.73
- glob_LSTM_DE: ade_val_g: 10.83 | fde_val_g: 14.93

More results in slides
https://docs.google.com/presentation/d/1kuFpq2B-QMXNHKYZJ9Del7wSiyStp9n0l-TRwqiool8/edit?usp=sharing