Skeleton-Based Typing Style Learning For Person Identification
Supplementary Material

1. Implementation details

1.1. Data pre-processing

We used *YOLOv3* [4] object detector for localizing the hand in the input frame. For the joint detector, we used *Convolutional Pose Machine* [6] (CPM). This model outputs a belief map of the joints location, where each belief map denotes a specific joint. The joint's location is given by a Gaussian whose σ and peak value are set according to the model's confidence, *i.e.*, small σ with large peak value if the model is very confident in the location of the joint and large σ with small peak value otherwise. In that manner, the CPM model can predict a location for a joint, even when the joint is entirely or partially occluded in a given frame. It can predict the joint's location according to the hand's context and decrease its belief score in exchange. This kind of method can help with cases of hidden joints since *StyleNet* can utilize the joint's score as an indicator for the liability of the data related to that joint.

1.2. Models implementation details

Pre-process pipeline: We implemented our models for the pre-process using Tensorflow framework. An Input frame of size 240×320 was given to the hand localizer to output a bounding box coordinates of the hand in the given frame. We cropped the hand centered frame according to the given bounding box and resized the cropped frame to a size of 368×368 with respect to the aspect ratio. The resized frame is given to the joint detector that produces belief maps in return. The belief maps are resized back to fit the original frame size with respect to the translation of the bounding box produced by the hand localizer. Finally, $argmax$ is applied to each belief map to locate the joints coordinates. We repeat this process for the entire dataset to produce the joints locations matrix, which consists of all 21 joints locations and belief scores by frame.

StyleNet: We implemented StyleNet using PyTorch framework. We defined A which is the adjacency matrix of the hand's joints and normalized it according to eq.2 from the main paper, where $\Lambda_k^{ii} = \sum_j (A_k^{ij}) + \sigma$ and σ equal to 0.001 is used to avoid empty rows. For each video, we sample a total of 32 matrices, where each matrix refers to a certain frame and comprises the frame's 21 (x, y) joints locations and their belief score. We created the bone data by subtracting the (x, y) coordinates of each neighboring joints pair to extract the bone vectors, while we multiplied both neighboring joints belief score to produce a bone belief score. Our model is following the AGCN [5] architecture, where each layer constructed from a spatial GCN unit that processes the joints or bones intra-frame relations and a temporal unit that process the temporal inter-frame relations. The model's 8th GCN unit modified according to eq.3 from the main paper to improve the long-range dependencies of the spatial feature maps before expanding the number of feature maps channels. We also modify the 10th TCN unit according to eq.7 from the main paper to improve the long-range dependencies between the different frames. The downsampling unit is applied after the 10th TCN unit for better downsampling of the final feature maps before forwarding to the classification layer.

1.3. Training details

Pre-process: We used *YOLOv3* model pre-trained on *COCO* dataset [2]. To train the model for our task, we created a single "hand" label and used *Hands* dataset [3] that contains $\sim 13k$ training and $\sim 2.5k$ validation images, labeled with hands bounding boxes location. We used Adam optimizer with an initial learning rate of 1e-3 and ran our training with a batch size of 16 for 150 epochs. We trained *CPM* model using trained weights [1] as an initial starting point. We used 1256 random frames from our *80Typing2* dataset labeled with their joints locations. Training data consist of 1100 frames and 156 frames used for validation. Data augmentation applied during training to prevent overfitting. We used Adam optimizer with an initial learning rate of 1e-3 and a batch size of 16 for a total of 960 epochs.

StyleNet: We used a batch size of 32, where each sampled video consists of 32 sampled frames from the entire video. We used Adam optimizer with an initial learning rate of 1e-3, a momentum of 0.9, and a weight decay of 1e-5. Both stream weights initialized to 1. A dropout rate of 0.3 was applied to increase the model's generalization ability. We trained the model for 100 epochs and decreased the learning rate by a factor of 10 after 40, 70, and 90 epochs. No data augmentation needed due to the natural augmentation of the data results from the sampling of the video.

2. Ablation Study

We conducted an ablation study to examine the effectiveness of our added blocks using *60Typing10*. We performed this experiment in the same manner as described in section 5.2 from the main paper, as this scenario offers a more challenging test case in which the true value of our comprised modules can manifest.

According to the results reported in table 2, we can see that each added block improves the accuracy rate when compared with the baseline. The most significant improvement was achieved when all the blocks added together. On a broader note, applying [7], [5], or any other variant of these methods on a small deformable structure will bias toward close-ranged dependencies (due to the *Softmax* normalization constructing C_k). As the close and long-range concept is no longer applicable in our task (moving only one of the hand's joints is almost impossible), these models achieve inferior results to our model, which focuses on non-local spatial and temporal connectivity. Specifically, it constructs a new order of information. Each joint can interact with **all** relevant (by attention) joints from all time steps, helping our model extract more meaningful motion patterns in space and time.

Model	[4,2,4] Acc(%)	[3,2,5] Acc(%)	[2,2,6] Acc(%)
2sAGCN [5]	99.04	98.82	97.97
W downsample unit	99.47	99.39	98.72
W downsample + TNL	99.62	99.59	99.13
W downsample + SNL	99.76	99.71	99.28
StyleNet	**99.84**	**99.77**	**99.50**

Table 1. Test accuracy of user classification on unseen sentences on *60Typing10* when adding each module to our baseline. $[\alpha, \beta, \gamma]$ denotes the number of sentences for train, validation, and test, respectively. NL denotes non-local, SNL denotes temporal non-local unit, and SNL denotes spatial non-local unit

References

[1] Tim Ho. Implementation of convolutional pose machines tensorflow. `https://github.com/timctho/convolutional-pose-machines-tensorflow`.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[3] Arpit Mittal, Andrew Zisserman, and Philip HS Torr. Hand detection using multiple proposals. In *BMVC*, volume 40, pages 75–1. Citeseer, 2011.

[4] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[5] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12026–12035, 2019.

[6] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[7] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.