Supplementary Material for Temporal Transformer Networks: Joint Learning of Invariant and Discriminative Time Warping

Suhas Lohit Qiao Wang Pavan Turaga
Geometric Media Lab, Arizona State University
{slohit, qiao.wang, pturaga}@asu.edu

1. Additional experimental results and details

1.1. Additional synthetic dataset Demonstrating rate-invariance property of TTN:

In this case, we construct a dataset such that rate variations in the signals are the major nuisance parameter. In this scenario, intuitively, minimizing classification error should lead to the following: different signals belonging to the same class, but differing (approximately) only by a γ should come closer to each other after passing through a trained TTN module. In class 1, we have signals which are a a mixture of two Gaussian functions with random warping applied and additive Gaussian noise added to them. Signals in class 2 are similar except that they are a single Gaussian function with the same mean and variance. As before, we generated 8000 training sequences and 2000 test sequences evenly balanced between classes 1 and 2. These are shown before (Column 1) and after random warping (Column 2) in Figure 1. The TTN, classifier and the training and testing protocol are the same as in dataset (1) above. From column 3 in Figure 1, it is clear the TTN leads to reduction in intra-class rate variations. Table 1 shows the classification accuracies obtained with and without the TTN module (averaged over 10 runs). When no warping is present in the input data, both variants yield perfect accuracy. When warping is introduced in the dataset, the performance of the vanilla model (i.e., without TTN) drops significantly. With the addition of the TTN module, most of the lost performance can be recovered.

	Vanilla	TTN
Unwarped	100.00 ± 0.00 %	**100.00** $\pm$ **0.00** %
Warped	96.31 ± 0.021 %	**99.03** $\pm$ **0.15** %

Table 1. Recognition results (%) for synthetic dataset 2. Addition of TTN clearly outperforms the baseline.

1.2. ICL First-Person Hand Action Dataset [1]

Training protocol details omitted in the main paper in Section 5.2: Both the classifier and the TTN are trained with momentum optimizer with momentum set to 0.9. A batch size of 16 is used and the networks are trained for 50000 iterations. We use an initial learning rate of 10^{-3} for the classifier and and 10^{-4} for the TTN. The learning rate is reduced to one-tenth after 35000 and 45000 iterations.

1.3. NTU RGB-D Dataset

TCN architecture and training details: We use the Temporal Convolution Network (TCN) described in [2]. The network consists of 10 convolutional layers with batch normalization and ReLU non-linearity. The network can be divided into 3 blocks of conv layers. After each block, the first conv layer of the next block is of stride 2 such that the inputs for successive block are of length half of that of the previous block. Residual connections are employed in the network as well. All the convolutional filters are of size 8. The convolutional layers produce 64,128 and 256 feature maps for blocks 1, 2 and 3 respectively. We use momentum optimizer with momentum 0.9 and initial learning rate of 5×10^{-3}. The network is trained for 2×10^5 iterations with a batch size of 72, and the learning rate was reduced one-tenth the value after 10^5 and 1.5×10^5 iterations. We found this to be these values of the hyperparameters to be the optimal setting for the baseline classifier network. While training, the TTN parameters are updated at one-tenth the learning rate of the TCN.

Ablation study for the TCN: We study the effect of the number of layers in the classifier network on the performance. The TCN architecture consists of 3 blocks of conv layers. We remove 1 block at a time and compare the cross-subject classification rate. The results are shown in Table 2. We see that the addition of the TTN produces better results in all cases.

Ablation study for the TTN module: We study the effect of the number and type (convolutional or fully-connected) of layers in the TTN on the cross-subject classification performance. The results are shown in Table 3. We see that the architecture of the TTN indeed has a significant effect on the performance. However, irrespective of the TTN architecture, addition of the TTN produces better performance in all cases.