Supplementary Material

1. Details of the Flash–MNIST Dataset and Experiments

In this section, we provide more details of the dataset generation, local feature extraction, and implementation of our attention cluster training on Flash–MNIST.

1.1. Dataset Generation

We generate the 102,400 training samples and 10,240 test samples that constitute our proposed Flash–MNIST dataset as follows:

1. **Generating frames with noisy background**: We randomly generate 25 different 28×28 noise frames for each video sample. First, since the each pixel of MNIST images is represented by an integer value in [0,255], we compute the distribution over such pixel intensity values in the MNIST training data, i.e., we count the frequency of each such integer in the training set. Then, for each pixel in each frame to be generated for our Flash–MNIST data, we randomly sample an integer value in $[0, 255]$ in accordance with the computed probability distribution.

2. **Randomly sampling digits**: We randomly sample a category from the set of 1024 possible categories in Flash–MNIST. Then, for each digit belonging to the sampled category, we randomly sample one or two corresponding images from MNIST, either from the training or the testing set, depending on target video. Thus, we may sample $0 - 20$ MNIST digit images per target video.

3. **Inserting digits into random frames**: We randomly pick the frame into which each sampled image shall be inserted. Then, we overlay the digit images on the random background by keeping the maximum value for each pixel.

1.2. Local Feature Extraction

To extract local features, we apply the following steps:

1. **Collecting samples for pretraining**: For each sample in the MNIST training split, we randomly generate 5 noise background images and place a digit on them. We also generate 30,000 noisy backgrounds without any digit. The goal is to classify each sample to one of 11 categories, digits in $\{0, \ldots, 9\}$ or just a noisy background.

2. **Training the network for local feature extraction**: We apply CNNs for frame classification. The CNNs consist of 2 successive convolutional layers with 5×5 kernels, 10/20 filters followed by *relu* activations and max-pooling with stride 2, one fully connected layer with 50 hidden units followed *relu* activations, and finally a fully connected layer with 11 hidden units followed by *softmax*. We use Adam optimization [3] with a learning rate of 0.001 to update the parameters and train for one epoch on the collected samples.

3. **Extracting local features for frames**: We apply the pretrained CNNs except for the last fully connected layer to extract local features with a dimensionality of 50 for frames in Flash–MNIST.

1.3. Attention Cluster Training Details

After extracting local features for each sample, we apply our attention clusters approach to obtain the global feature representation and then use a fully connected layer with 1024 hidden units followed by a *softmax* layer for classification. In order to reduce overfitting, we apply dropout with probability 0.5 before the final fully connected layer. We rely on the Adam algorithm to update parameters with a learning rate of 0.001 and at most train for 100 epochs.

2. Details of Multimodal Integration Methods

In this section, we introduce the network structure for the comparison of multimodal integration methods described in Section 5.6. The Average and Flatten methods serve as basic baselines. TS-LSTM [4], Temporal-Inception [4], and Bi-directional LSTMs provide baselines with temporal modeling.

- **Average**: The Average baseline is the most straightforward method. This method generates global features by concatenating the global averages of each modality. The global feature is used for classification through a fully connected layer.
- **Flatten**: The Flatten method is also a straightforward one. This method first flattens the local features of each modality. This means that we can obtain an LM length output for a local feature set of dimensionality $L \times M$. Then, the outputs of three modalities are concatenated for classification.
- **TS-LSTM and Temporal-Inception**: We simply extend the two-modality integration methods described in the original work [4] to a three-modality version. We set the number of segments to 5 for TS-LSTM.
- **Bi-directional LSTMs**: For local features of the RGB/flow/audio signals, we apply a batch normalization layer [2] followed by bi-directional LSTMs [1] with 1024/512/512 hidden units, respectively. The averages of the output hidden states of each bi-directional LSTM are then concatenated for classification.

References

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2

[2] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[4] C.-Y. Ma, M.-H. Chen, Z. Kira, and G. AlRegib. TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *arXiv preprint arXiv:1703.10667*, 2017. 2