Supplementary Material:
Unified Graph Structured Models for Video Understanding

Anurag Arnab Chen Sun Cordelia Schmid
Google Research
{aarnab, chensun, cordelias}@google.com

In this supplementary, we include additional qualitative results in the attached video (Sec. A1), additional analysis on AVA (Sec. A2), analysis of the computation used by our model (Sec. A3) and implementation details (Sec. A4).

A1. Additional qualitative results

Please refer to the attached supplementary video for additional qualitative results on AVA, Action Genome and UCF101-24.

The codec used is "H264 - MPEG-4 AVC" and has been tested on "VLC Media Player".

A2. AVA per-class analysis

Figure A1 presents per-class results on AVA for our SlowFast baseline, spatial and spatio-temporal graph models when using a ResNet-50 backbone. We additionally show the number of examples in the training set. Our proposed graph models improve on both head and tail classes in AVA. In fact, the largest absolute improvements are observed on tail classes such as "cut", "swim" and "sing to". However, note that the absolute accuracy of each class is still correlated with the number of training examples. The classes which all variants of our model perform the worst on, are the classes with the least training examples such as "point to (an object)", "take a photo" and "turn (eg a screwdriver)". To emphasise the long-tailed distribution of AVA, note that "point to (an object)" has the fewest training examples (just 97), whilst "watch (a person)" has the most with 168 148.

A3. Computation analysis

Table A1 shows the computation used by our models. We measure compututation in terms of floating point operations (FLOPs), and GFLOPs denotes billions of FLOPs.

Our graph model is significantly more efficient than Non-local in the last layer of the backbone as it only passes messages to actor nodes, instead of all elements in the feature map. Computation increases linearly with the number of frames processed, (and thus τ_c), as the majority of compute is performed by the ResNet backbone The computation required does not depend on τ_s.

Note that our graph model adds neglible compute compared to the ResNet backbone. In particular, the spatial model with GAT only adds 0.3% more FLOPs to the ResNet backbone, while increasing SGCls R@20 by 4.5%.

Finally, we note that compute is essentially identical for Action Genome and AVA models, as only the final "readout" layer changes.

A4. Additional Implementation details

Person detections For our spatio-temporal action detection experiments, we use external person detections as our actor region proposals like [1, 10]. For our experiments on AVA, we use the person detections publicly released by [1, 10] during both training and testing. More specifically, this is a Faster-RCNN [5] model, pretrained on Microsoft COCO [4] and then finetuned on the training set of AVA [2], using Detectron [11].

For our experiments on UCF101-24 [6], we finetune a Faster-RCNN detector pretrained on COCO on the UCF101-24 training set using Detectron.

When training on both AVA and UCF101-24 datasets, we use both ground-truth and predicted person bounding boxes. Predicted bounding boxes are assigned labels by matching them to ground-truth boxes using an IoU threshold of 0.75. Predicted bounding boxes which do not match any ground truth boxes act as negative examples for all action classes.

OpenImages object detector For our "explicit object representation" experiments on AVA, we use the publicly available Faster-RCNN detector trained on OpenImages v4[1].

Optimiser hyperparameters We train our models following the settings of the publicly released SlowFast

1. https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1

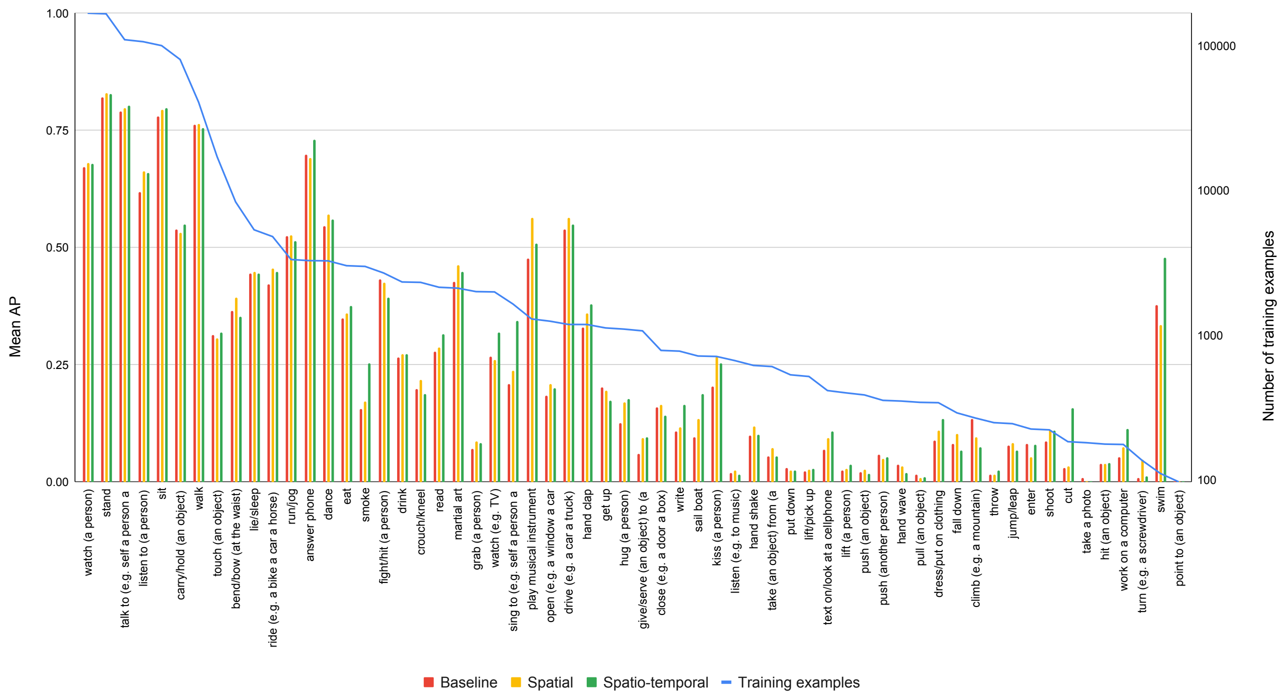

Figure A1. Mean Average Precision for each action class on AVA, for the SlowFast baseline, spatial- and spatio-temporal graph models with a ResNet-50 backbone. The number of examples in the training set are shown with the blue line on the right, vertical axis. Note that a logarithmic scale is used, as AVA has a long-tailed distribution of labels.

Table A1. Compute used by baseline and proposed models. Our graph model is significantly more efficient than Non-local in the last layer of the backbone as it only passes messages to actor nodes, instead of all elements in the feature map. Computation increases linearly with the number of frames processed, (and thus τ_c), as the majority of compute is performed by the ResNet backbone.

Model	Frames	GFLOPs	Parameters ($\times 10^6$)	SGCls R@20
SlowFast (ResNet 50)	32	74.18	34.6	48.9
Non-local in backbone	32	83.66	43.0	49.1
Spatial model, GAT, $\tau_c = 1$	32	74.41	36.5	51.1
Spatial model, Non-local, $\tau_c = 1$	32	74.47	37.0	50.4
Spatio-temporal, GAT, $\tau_c = 3$	96	223.25	36.5	53.5
Spatio-temporal, GAT, $\tau_c = 5$	160	372.08	36.5	53.8

code [1]. We train for 20 epochs using synchronous Stochastic Gradient Descent (SGD) and a momentum of 0.9 on 8 Nvidia V100 (16 GB) GPUs. The learning rate was set to 0.1, and reduced by a factor of 10 after 10 and 15 epochs respectively. We employed a linear-warmup schedule for the learning rate, increasing linearly from 1.25×10^{-4} to 0.1 in the first 5 epochs. We also used a weight decay of 10^{-7}. An epoch is defined as all the keyframes in the dataset.

Training loss functions For spatio-temporal action recognition on AVA and UCF101-24, we use the binary cross-entropy loss function. The loss function for a single example is:

$$L(x, y) = -\frac{1}{C} \sum_{i}^{C} y_i \log\left(\sigma(x_i)\right) + (1 - y_i) \log\left(1 - \sigma(x_i)\right), \tag{A1}$$

where C is the number of classes, σ is the sigmoid activation function, $x \in \mathbb{R}^C$ denotes the logits predicted by the network and $y_i \in \{0, 1\}$ denotes the binary ground truth for the i^{th} class.

For scene graph classification, we use two loss functions: One for predicting the object class of each bounding box proposal, and another for predicting the relationship label

Table A2. Ablation study of the number of heads when using GAT as the message passing function. Mean AP reported on AVA using a ResNet-50 backbone.

Number of heads	1	2	3	4	5
Mean AP	25.4	25.8	25.9	25.9	25.9

between each of the $\frac{N(N-1)}{2}$ pairs of object proposals. For the former, we use the softmax cross-entropy, as each proposal can only be assigned a single object class. And for the latter, we use the sigmoid cross-entropy, as there can be multiple relationship labels between any pair of objects. Concretely, the loss is

$$L(x, w, y, z) = \lambda L_{\text{object}}(x, y) + L_{\text{rel}}(w, z) \tag{A2}$$

$$L_{\text{object}}(x, y) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{C} y_{ij} \log\left(\text{Softmax}(x)_{ij}\right) \tag{A3}$$

$$\begin{aligned} L_{\text{rel}}(w, z) = -\frac{2}{N(N-1)R}\sum_{i}^{N}\sum_{j}^{i-1}\sum_{r}^{R} z_{ijr} \log\left(\sigma(w_{ijr})\right) \\ + (1 - z_{ijr})\log\left(1 - \sigma(w_{ijr})\right). \end{aligned} \tag{A4}$$

Here, N is the number of object proposals in the video clip, R the number of relationship classes and C the number of object classes. y is the one-hot, ground-truth object class label, z the binary ground-truth relationship label, and x and w denote the object- and relationship-logits respectively. On Action Genome [3], there are $R = 25$ relationship classes, and $C = 35$ object classes. We set $\lambda = 0.5$ during training on Action Genome to prevent the object class loss from dominating the overall loss (the softmax cross entropy has a higher loss than the binary sigmoid cross entropy at the start of training).

Multi-headed attention For the GAT [8] and Non-Local [9] message-passing functions used in the paper (Sec 3.4), it is common to compute "multi-headed" attention [7]. In all experiments reported in the main paper, we used 4 heads, as motivated by the ablation study using GAT on the AVA dataset in Tab. A2. To combine the messages from each head, we performed an attention-weighted convex combination as done in Eq. 9 of the main paper.

Multiscale testing We use three scales when performing multiscale testing (as done in Table 3 of the main paper). Specifically, we resize input video-clips so that the shortest side is 224, 256 and 320 respectively.

References

[1] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 1, 2

[2] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018. 1

[3] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *CVPR*, 2020. 3

[4] 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 *ECCV*, 2014. 1

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[6] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. In *arXiv preprint arXiv:1212.0402*, 2012. 1

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 3

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018. 3

[9] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 3

[10] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019. 1

[11] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 1