A Deeper Dive Into What Deep Spatiotemporal Networks Encode: Quantifying Static vs. Dynamic Information

Supplementary Material

Matthew Kowal[1,2], Mennatullah Siam[1], Md Amirul Islam[2,3]
Neil D.B. Bruce[2,5], Richard P. Wildes[1,4], Konstantinos G. Derpanis[1,2,4]
[1]York University, [2]Vector Institute for AI, [3]Ryerson University, [4]Samsung AI Centre Toronto, [5]University of Guelph
{m2kowal,msiam,wildes,kosta}@eecs.yorku.ca, mdamirul@ryerson.ca, brucen@uoguelph.ca

1. Introduction

Our supplemental material is organized in five major sections. Section 2 documents an associated supplementary video. Sections 3 and 4 provide details regarding our action recognition and video object segmentation experiments, resp. Each of these sections is partitioned into an initial subsection that presents implementation details, followed by a series of subsections providing supplemental empirical results and analyses. Finally, Section 7 documents all assets employed in our work. All references to equations refer to equations defined in the main paper.

2. Supplemental video

We include an accompanying supplemental video as part of the supplementary materials which can also be found on our project page[1]. In this video, we show examples of the static and dynamic pairs for both action recognition and video object segmentation (VOS). The video is in MP4 format and approximately three minutes long. Layouts for each sampling pair are described in detail followed by the example video samples. The codec used for the realization of the provided video is H.264 (x264).

3. Action recognition

In this section, we provide details for action recognition. We begin by presenting implementation details for all models evaluated in the main submission. Subsequently, we provide a supplementary series of experiments where we consider various frame rates as input to the SlowFast network [4], variation of the threshold, λ, in the unit-wise metric, (3), and the effect of the training dataset.

3.1. Implementation details

The main repository used for our action recognition experiments is the SlowFast [4] repository[2]. This repository contains dozens of pre-trained action recognition architectures trained on multiple datasets. The only model taken from a different repository is the TimeSformer [1], which has its own codebase[3] built upon the SlowFast repository. For all models, we use the standard configuration files provided by the repository except for the following.

The FastOnly model is implemented by us based on the SlowFast architecture found in the SlowFast repository. For a fair comparison with the SlowFast model, the FastOnly model is implemented using the same frame and sampling rate as the SlowFast-Fast branch (32 total frames sampled every two frames).

All models trained on Kinetics-400 [2] and Something-Something-v2 [5] (SSv2) are taken directly from the SlowFast repository, except for the FastOnly model. The FastOnly model is trained on Kinetics-400 for 40 epochs with SGD, a weight decay of 1e-4, a batch size of 32 and a base learning rate of 0.03 that decreases by a factor of 10 at epochs 15, 30 and 35. On SSv2, the FastOnly model is trained for 25 epochs with SGD, weight decay of 1e-4, a batch size of 32 and a base learning rate that is decreased by a factor of 10 at epochs 10 and 20.

All models trained on Diving48 [9] are trained by us. The FastOnly model is trained on Diving48 for 100 epochs with SGD, weight decay of 1e-4, a batch size of 32 and a base learning rate of 0.0375 that decreases by a factor of 10 at epochs 40, 60 and 80. The SlowOnly model is trained on Diving48 for 100 epochs with SGD, weight decay of 1e-4, a batch size of 32 and a base learning rate of 0.00375 that de-

[1] https://yorkucvil.github.io/Static-Dynamic-Interpretability/

[2] https://github.com/facebookresearch/SlowFast

[3] https://github.com/facebookresearch/TimeSformer

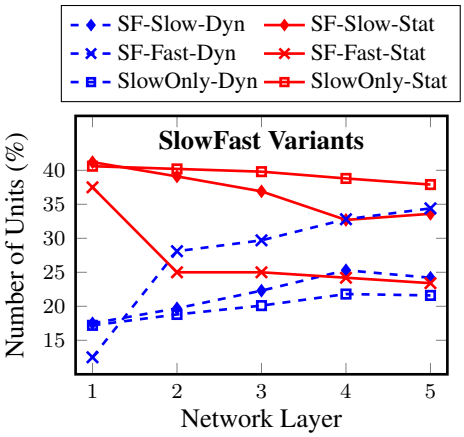

Figure 1. Static and dynamic bias analysis on SlowFast variants with alternative sampling rates trained on Kinetics-400 [2] using the layer-wise metric, (3). All models are trained with four frames sampled every 16 frames (*i.e.* 4×16). SF-Slow and SF-Fast denote the representation taken before the fusion layer from the Slow and Fast branches, respectively.

creases by a factor of 10 at epochs 40, 60 and 80. All models trained with temporal frame shuffling (see Sec. 4.2.1 of the main paper) are trained with the same hyperparameters as their unshuffled counterparts.

We use standard augmentations that are found in the SlowFast repository, which include random spatial cropping and random temporal cropping, followed by resizing to 224×224. The number of frames and sampling rate for all models is 8×8 unless otherwise specified. At validation time, a single clip was spatially and temporally center cropped. All models were trained on four NVIDIA Tesla T4s. Training times for each model and dataset vary significantly. Training the SlowFast model on the Diving48 dataset takes approximately 2.5 days which is the longest training time among all considered models.

3.2. SlowFast frame rates

Figure 1 shows the static and dynamic units estimated using the layer-wise metric, (1). The main paper examined architectures with a frame number and sampling rate of 8×8 while Fig. 1 shows SlowFast variants trained with a frame number and sampling rate of 4×16. It can be seen that the Fast branch injects dynamic information into the Slow branch via the fast-to-slow cross connections. The last layer of the SlowOnly model has 21.6% units (*i.e.* channels) encoding dynamics, while when trained jointly with the Fast branch, the SF-Slow model has 24.2% dynamic units in the final layer. This increase of 2.6% is similar to the one seen with sampling parameters of 8×8, which saw an increase of 3.3% in the last layer.

3.3. Varying thresholds

Figure 2 shows the static and dynamic unit-wise analysis, (3), with varying thresholds, λ, for various action recognition architectures. The FastOnly and SlowFast-Fast models are the only ones that produce specialized dynamic units which is consistent with the findings in the main paper. Moreover, the SlowFast-Fast branch retains a significant number of dynamic units even at the higher thresholds (*e.g.* 0.8). This pattern further shows the efficacy of using the two-stream architecture for capturing separate types of information. Note that all models produce more residual units as the threshold increases since few units have correlation coefficients in the range 0.8 to 1.

3.4. Training dataset effect

In this section, we provide additional models trained on SSv2 and Diving48. In the main paper, we showed that SSv2 produces dynamic units and Diving48 produces residual units, while Kinetics-400 mainly produces static units. To this end, we analyse SlowFast models trained on each dataset using the unit-wise metric, (3), for $\lambda = 0.5$ on the Slow and Fast branches separately; see Fig. 3. The findings from the main paper are consistent with those seen here. Diving48 is the only dataset to produce a notable number of residual units, which suggests that there are other factors than static and dynamic that are important for classifying dives in this dataset. We leave it for future work to explore what these residual units capture. SSv2, on the other hand, yields a large number of dynamic units regardless of the architecture. Note that the Fast branch contains dynamic units regardless of the dataset, again showing the efficacy of this two-stream approach for separating static and dynamic information.

4. Video object segmentation

In this section, we provide details for video object segmentation. We begin by presenting implementation details for all models evaluated in the main submission. Subsequently, we provide a series of supplementary experiments where we consider the effect of architectures on static vs. dynamic encoding. For each architecture we (a) consider the effect of the threshold, λ, in our unit-wise metric, (3), (b) examine all fusion layers and (c) present supporting experiments on a dataset that especially emphasizes the importance of motion in segmentation, as the objects of interest are camouflaged in single frames, MoCA [8]. Finally, we demonstrate the individual unit analysis on different VOS datasets with varying thresholds, λ, to provide additional confirmation of our final conclusions.

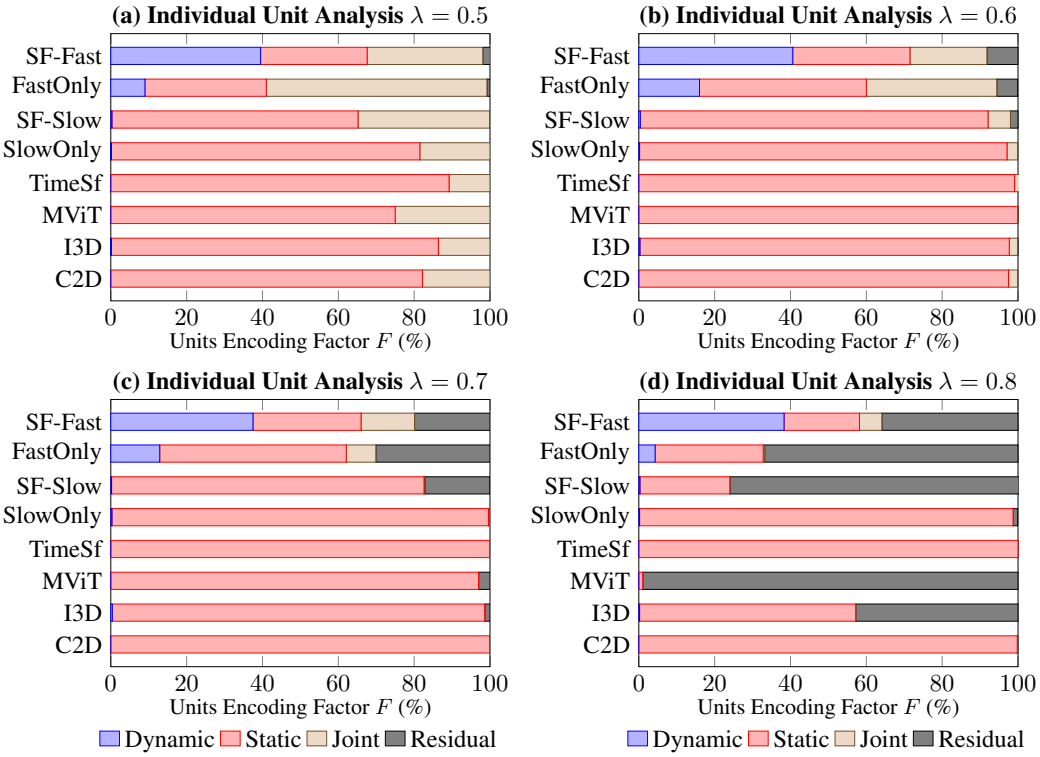

Figure 2. Estimates of the dynamic, static, joint and residual units using the unit-wise metric, (3), for the different action recognition architectures at varying values of the threshold, λ.

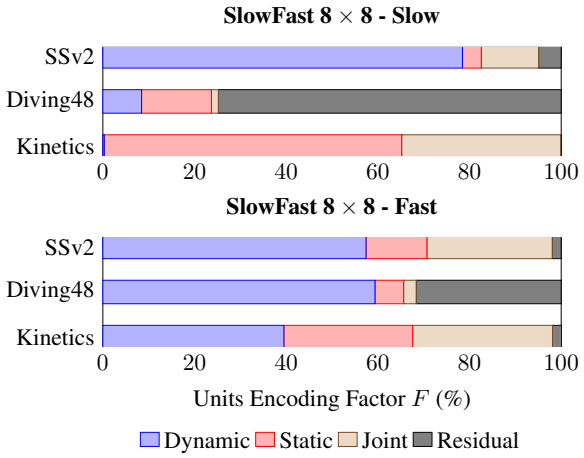

Figure 3. Analyses of static and dynamic biases of action recognition datasets using the unit-wise metric, (3), for the SlowFast architecture with the number of frames and sampling rate of 8×8. Slow and Fast represent the Slow branch and Fast branch of the SlowFast model, respectively. The estimates are based on the penultimate layer of each stream separately, before the concatenation of the representations.

4.1. Implementation details

In this subsection, we describe the implementation details for FusionSeg [7] modified version, MATNet [14] variants and the evaluation on MoCA [8]. In our modified version of FusionSeg we follow the original in using a ResNet-101 [6] backbone with five stages, the first being the early convolutional layers and the rest being four ResNet-like stages. However, unlike the original work, we apply the fusion between both motion and appearance features on the intermediate representations at stages two and five. As explained in the main submission, we make this adjustment to allow for comparison with MATNet and RTNet that perform fusion on the intermediate representations. We use 1×1 convolutional layers for the fusion, which take concatenated features and output 256 and 2048 feature channels at stages two and five, respectively. Similar to the original approach, the segmentation decoder uses atrous spatial pyramid pooling (ASPP), but we also use an encoder-decoder architecture [3]. Specifically, we concatenate the features extracted from the fusion of stage two and ASPP features to produce the final segmentation mask.

Our FusionSeg model is trained with a batch size of eight, using SGD with learning rate 0.001, along with a momentum of 0.9, and weight decay 1×10^{-4}. Additionally,

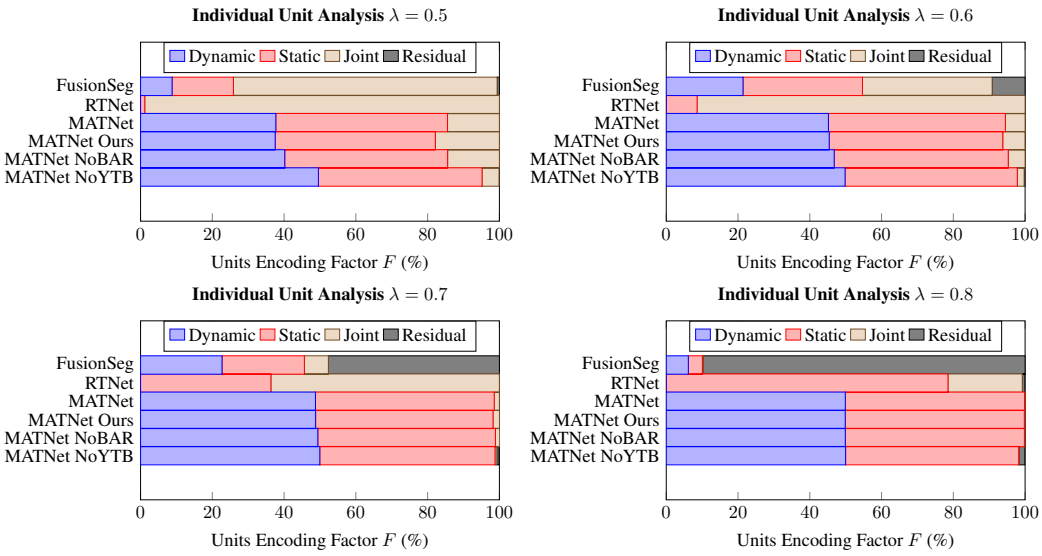

Figure 4. Estimates of the dynamic, static, joint and residual units using our metric for unit-wise analysis, (3), for the different VOS models at various settings of the threshold, λ.

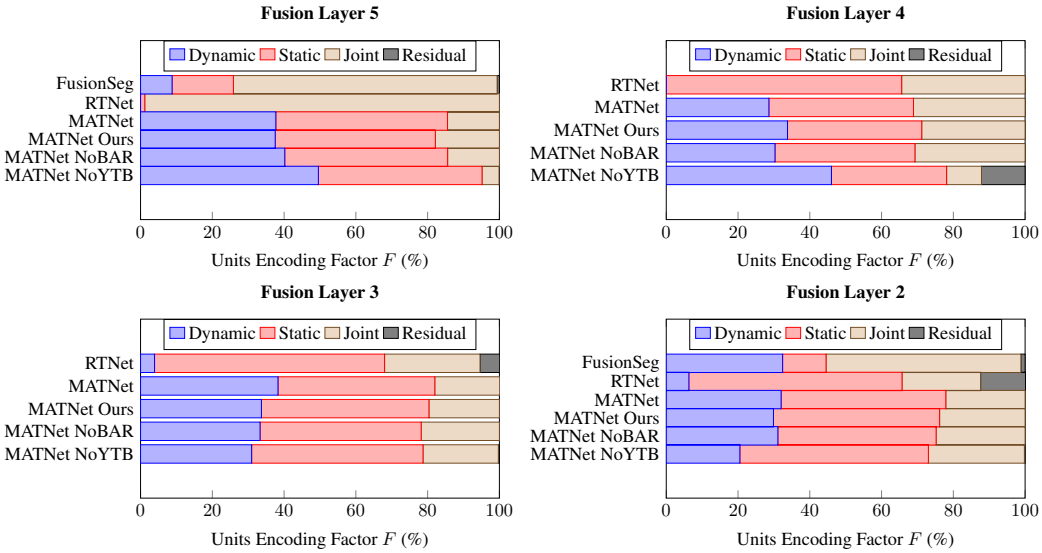

Figure 5. Estimates of the dynamic, static, joint and residual units using our metric for unit-wise analysis, (3), $\lambda = 0.5$ comparing MATNet variants and RTNet at different fusion layers. Our modified version of FusionSeg has only two fusion layers (*i.e.* fusion layer two and five) to be compared.

we use a "poly" learning rate policy using a power of 0.9. We use random scaling with scale randomly sampled between (0.5, 2.0), random cropping with size 513×513 and random horizontal flipping for data augmentation. We do not perform pretraining for the motion stream, unlike what was proposed in the original paper. We make this choice because we focus on training the joint model directly on three different datasets to assess their effect.

MATNet variants are trained on a multi-GPU (with two GPUs 1080 Ti) machine with batch size six (unlike the original MATNet that used batch two and no multi-GPU training), the rest of the training hyperparameters follow the original work [14]. We denote the original model provided by the authors without finetuning or training on our side as "MATNet", while reproduction of MATNet with training on multi-GPU and a batch size of six as "MATNet Ours". We train a MATNet variant without boundary aware refinement (BAR) modules that we call "MATNet NoBAR", where we

remove all BAR modules and the boundary loss. We also experiment with another MATNet variant that does not train on additional YouTubeVOS data [12], unlike the originally proposed model, we call this version "MATNet NoYTB". We specifically introduce MATNet Ours to provide a MATNet variant that is directly comparable to other variants that we introduce (*i.e*. MATNet NoBAR and MATNet NoYTB), as it has the same training paradigm, unlike the original MATNet [14]. We evaluate the static and dynamic units for these variants to investigate the reason behind the increased dynamic units with respect to other models considered (*i.e*. FusionSeg and RTNet). We make no modifications to RTNet and use the public version provided by the authors [10]. They provide a model with a ResNext50 backbone, which we denote as "RTNet" throughout the paper. For all architectures, we use RAFT [11] to supply the optical flow estimates used for sampling of static and dynamic pairs on the stylized DAVIS16 validation dataset.

Finally, we describe the evaluation details on the Moving Camouflaged Animals dataset (MoCA) [8]. We follow previous work by removing videos that contain no predominant target locomotion, which produces a subset of 88 videos for evaluation [13]. We evaluate using mean intersection over union and success rate with varying IoU thresholds, τ, ranging from 0.5 to 0.9. We evaluate our modified FusionSeg, RTNet provided by the original work, and our MATNet variants on MoCA. The original MATNet evaluation on MoCA is reported in previous work [13]. It is worth noting, that the original MATNet used data augmentation as horizontal flipping during the inference and averaged predictions from the original and flipped versions. To ensure fair comparison with RTNet and FusionSeg on MoCA we disable the data augmentation during inference when reporting on MoCA.

4.2. Architectural effect

Figure 4 shows the unit-wise analysis, (3), on all studied VOS architectures with various settings of the threshold, λ, for the late fusion layer (*i.e*. fusion layer five). We vary the threshold, λ, between 0.5 to 0.8. It is seen that the off-the-shelf MATNet [14] consistently contains more dynamic units than both RTNet [10] and our modified FusionSeg [7]. For RTNet, increased values of λ yield an increased number of static units at the expense of joint units, while the number of dynamic units always remain small. For FusionSeg, both dynamic and static units initially increase at the expense of joint units as λ increases; however, at the highest value of λ the majority of units become residuals. The pattern of decreased numbers of joint units with increased values of λ arises because the requirement for units to be judged as jointly encoding becomes increasingly stringent; see (3).

In further comparing the different MATNet variants (including the variant that lacks boundary aware refinement

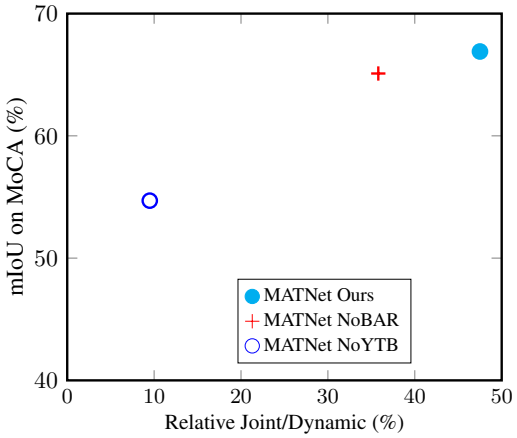

Figure 6. The relative joint/dynamic units in the final fusion layer compared with the mean intersection over union on MoCA for dynamically biased models (*i.e*. MATNet variants).

modules and the variant that lacks additional YouTubeVOS training) to FusionSeg and RTNet, it is seen that the proportion of dynamic units remains higher. This result suggests that the source behind the increase in the number of dynamic units in MATNet is the motion-to-appearance cross connections, rather than additional data or boundary refinement. It also shows for MATNet variants that with higher settings of the threshold, λ, the joint encoding units decrease and the specialized static/dynamic units increase. Figure 5 shows the comparison among different VOS architectures on all fusion layers (*i.e*. fusion layers two, three, four and five). It is seen that in fusion layers three, four and five, MATNet consistently has more dynamic units compared to FusionSeg and RTNet, which instead tend to have more jointly encoding units. The same pattern was shown in the main submission, albeit only for the final fusion layer. It is also seen that MATNet tends to balance between the specialized units of both static and dynamic factors.

Interestingly, however, in fusion layer two, FusionSeg appears on par with MATNet in terms of dynamic units, while it has fewer static units and more joint units. In comparison, RTNet tends to have the most unbalanced units of all three models, which are skewed toward the jointly encoding units in the late fusion layers (*i.e*. three, four and five). This pattern confirms that models with less ability to capture dynamics in the late fusion layers (*i.e*. FusionSeg and RTNet), generally tend to favour jointly encoding units over specialized units, and have less balance between both static and dynamic units. Thus, over all fusion layers and thresholds MATNet consistently has more balance between dynamic and static units and generally more dynamic units than other models, making models with cross connections that are not pretrained on saliency segmentation datasets one of the best to capture dynamics.

Method	mIoU	Success Rate					
		$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$	SR_{Mean}
FusionSeg [7] Modified	42.3	47.9	43.6	35.9	24.2	9.4	39.2
RTNet [10]	60.7	67.9	62.4	53.6	43.4	23.9	50.2
MATNet [14]	64.2	71.2	67.0	59.9	**49.2**	24.6	54.4
MATNet Ours	**67.3**	**75.9**	**70.8**	**61.9**	48.6	**26.0**	**56.6**
MATNet NoBAR	65.1	73.6	68.0	58.9	44.7	21.5	53.3
MATNet NoYTB	54.7	59.9	53.5	44.0	31.0	13.4	40.3

Table 1. Evaluation of VOS models on MoCA [8]. Success rates are reported using different IoU thresholds, τ. FusionSeg, RTNet and MATNet are the same versions reported in the main submission and can be compared directly. The three MATNet variants (MATNet Ours, MATNet NoBAR and MATNet NoYTB) are trained on our side, while MATNet [14] is the original without our training; see Sec. 4.1. Thus, while the three variants trained on our side are directly comparable, they cannot be compared directly with the original MATNet.

To further support the conclusion that MATNet's architecture provides the best ability to capture dynamics, we evaluate on a downstream task that requires capturing dynamics (*i.e.* segmenting moving camouflaged animals). In particular, we evaluate all models on the MoCA subset reported in previous work [13] and show results in Table 1. It is seen that the original MATNet and MATNet Ours both outperform RTNet and FusionSeg when motion is key to segmentation (MoCA).

We now further pursue the main driving factors behind MATNet's improved performance on MoCA over alternative state-of-the-art models. Figure 6 shows the mean intersection over union on MoCA with respect to the relative joint to dynamic units in the final fusion layer in different MATNet variants. For this experiment, we consider only the MATNet variants trained on our side (MATNet Ours, MATNet NoBAR and MATNet NoYTB), as they are directly comparable, unlike the original MATNet; see Sec. 4.1. The best MATNet variant on MoCA is the one trained with additional YouTubeVOS data and using the boundary aware refinement modules with auxiliary boundary losses (*i.e.* MATNet Ours). Interestingly, having more dynamic units along with maintaining a relative number of joint to dynamic units above a certain threshold improves MoCA performance. All MATNet variants generally have more dynamic units in their fusion layers than the rest of the VOS models. This suggests the driving reasons behind the state-of-the-art performance of MATNet on MoCA encompasses two main choices: (i) the inclusion of cross connections and (ii) additional training with YouTubeVOS.

4.3. Training dataset effect

In this section, we conduct additional experiments for understanding the training dataset effect on our modified version of FusionSeg [7] by augmenting the results shown in the main submission by varying λ. Figures 7 and 8 show results obtained with our unit-wise metric, (3), for both fusion layers five and two. It is seen that TAO-VOS yields more specialized dynamic units than ImageNet VID with all thresholds, λ. It is also seen that TAO-VOS in fusion layer five yields more specialized dynamic units with respect to DAVIS16 on thresholds $\lambda = \{0.5, 0.6\}$ and then starts to be on-par with DAVIS16 at higher thresholds. In contrast, in fusion layer two TAO-VOS yields more dynamic units than DAVIS16. Consistently, it is further seen that there are a higher number of residual units resulting from TAO-VOS than the other two datasets in fusion layer five for thresholds $\lambda = \{0.5, 0.6, 0.7\}$. These results indicate that there are also other factors beyond static and dynamic factors that are captured when training on TAO-VOS. We leave it for future work to explore what these residual units capture.

5. Neuron mask removal

To evaluate the effect of the estimated static and dynamic biased units on overall performance, we conduct perturbation experiments. In these experiments, we remove the top k units (*i.e.* channels) that are biased towards the static or dynamic factor during inference and evaluate the final accuracy drop. The removal is done by setting all activations to zero in the identified channels. We compare these static or dynamic biased units with respect to randomly selected channels. Figures 9 and 10 show the unit removal results for the action recognition and video object segmentation experiments, respectively. In action recognition we remove the top-k static, dynamic and random channels from the final layer in the SlowFast model trained on SSv2. We then evaluate on the SSv2 validation set and report the top-1 accuracy. As can be seen in Fig. 9, the dynamic factor maximally reduces the model's performance, which may be because (i) the SlowFast model encodes a significant amount of dynamic information in the fast branch and (ii) dynamics are important to solve the SSv2 dataset. We conduct similar experiments to video object segmentation for the four fusion layers of the MATNet model trained on DAVIS and YouTube-VOS. We evaluate on the MoCA dataset and report the mean intersection over union (mIoU). The results in

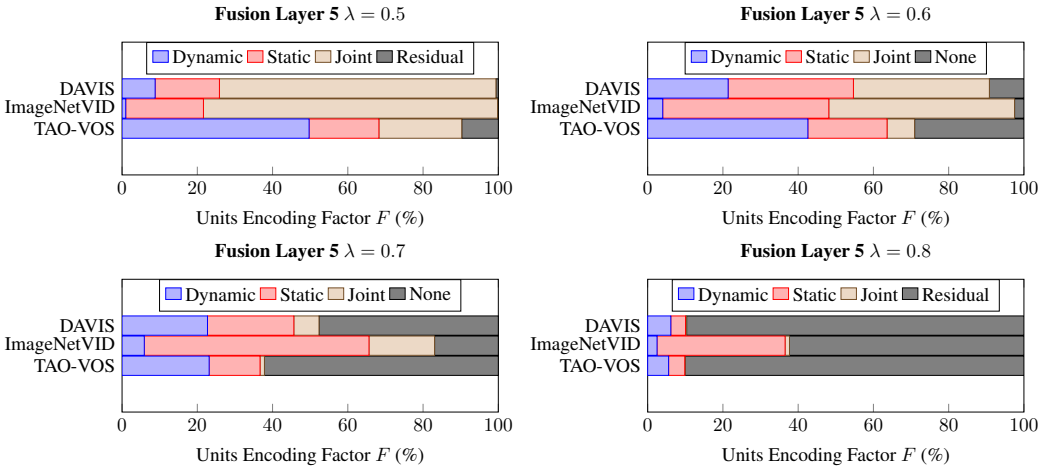

Figure 7. Estimates of the dynamic, static, joint and residual units using the unit-wise metric, (3), for the different VOS datasets at various settings of the threshold, λ, for fusion layer five.

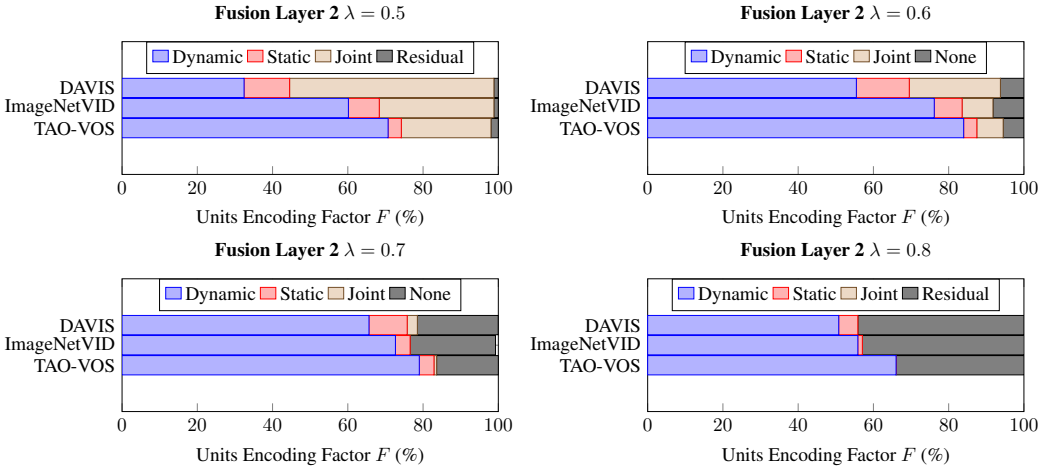

Figure 8. Estimates of the dynamic, static, joint and residual units using the unit-wise metric, (3), for the different VOS datasets at various settings of the threshold, λ, for fusion layer two.

Fig. 10 consistently demonstrate that for every fusion layer the factor with the highest impact on performance is the factor it is most biased toward, as examined earlier in the main submission (Fig. 4). In both tasks, these experiments document that masking out the top-k channels based on our proposed static/dynamic bias estimate can help us control what the model is biased toward and consequently affect its accuracy compared with randomly selected channels.

6. Computational load

We provide details of the models used in the paper in regards to their computational load. For each model, we list their FLOPs and parameter count in Table 2. We do not observe any correlation between computational load and biases of the model and leave a deeper analysis of this connection for future work.

7. Assets

Action recognition. We use provided code and trained weights from the SlowFast repository[4] and TimeSformer repository[5]. SlowFast is licensed under the Apache 2.0 license[6]. TimeSformer is licensed under the CC-NC 4.0 In-

[4] https://github.com/facebookresearch/SlowFast

[5] https://github.com/facebookresearch/TimeSformer

[6] https://github.com/facebookresearch/SlowFast/blob/main/LICENSE

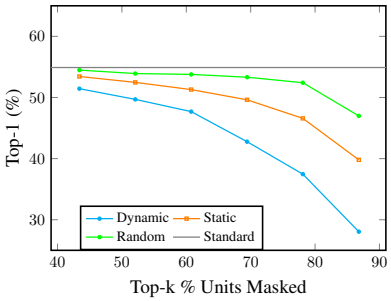

Figure 9. Top-k unit removal experiments for the static and dynamic factors with respect to random units for action recognition. The final layer of the SlowFast model trained on the SSv2 dataset is considered.

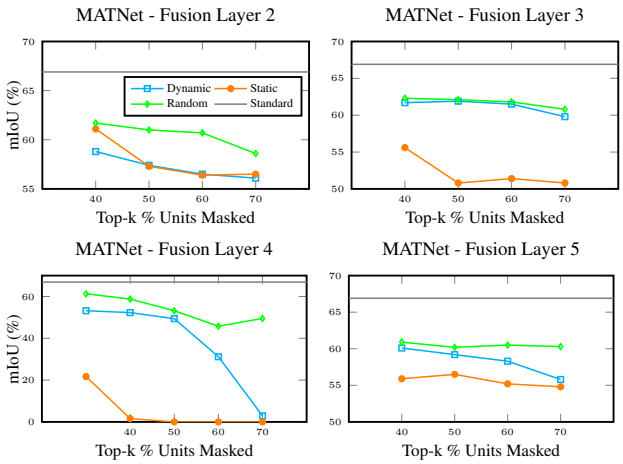

Figure 10. Top-k unit removal experiments for the static and dynamic factors with respect to random units for video object segmentation. The three fusion layers of the MATNet model trained on DAVIS and YouTube-VOS are considered.

ternational license[7] and Apache 2.0 license[8]. We use the Kinetics-400[9], SSv2[10] and Diving48[11] datasets.

Video object segmentation. We use provided code and trained weights for MATNet[12] and RTNet[13]. No accompanied licences are provided with the aforementioned code.

[7] https://github.com/facebookresearch/TimeSformer/blob/main/LICENSE

[8] https://github.com/facebookresearch/SlowFast/blob/main/LICENSE

[9] https://github.com/cvdfoundation/kinetics-dataset

[10] https://20bn.com/datasets/something-something

[11] http://www.svcl.ucsd.edu/projects/resound/dataset.html

[12] https://github.com/tfzhou/MATNet

[13] https://github.com/OliverRensu/RTNet

Model	Action Recognition	
	Parameters (M)	GFLOPs
C2D	24.3	25.6
I3D	28.0	37.3
X3D-m	3.8	6.4
SlowOnly	32.5	54.8
FastOnly	0.6	7.0
SlowFast (8x8)	34.6	66.1
MViT	36.6	70.7
TimeSformer	121.6	196.1
Model	**Video Object Segmentation**	
	Parameters (M)	GFLOPs
MATNet	142.7	156.0
RTNet	277.2	309.7
FusionSeg	113.0	112.5

Table 2. Computational loads of the action recognition and video object segmentation models studied.

Additionally, we use the DAVIS16[14], TAO-VOS[15] and ImageNet VID[16] datasets.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning*, 2021. 1

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1, 2

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 801–818, 2018. 3

[4] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6202–6211, 2019. 1

[5] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5842–5850, 2017. 1

[14] https://davischallenge.org/davis2016/code.html

[15] http://www.vision.rwth-aachen.de/page/taovos

[16] http://vision.cs.utexas.edu/projects/fusionseg/training_data.html

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[7] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. FusionSeg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2126. IEEE, 2017. 3, 5, 6

[8] Hala Lamdouar, Charig Yang, Weidi Xie, and Andrew Zisserman. Betrayed by motion: Camouflaged object discovery via motion segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2, 3, 5, 6

[9] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision*, pages 513–528, 2018. 1

[10] Sucheng Ren, Wenxi Liu, Yongtuo Liu, Haoxin Chen, Guoqiang Han, and Shengfeng He. Reciprocal transformations for unsupervised video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15455–15464, 2021. 5, 6

[11] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision*, pages 402–419. Springer, 2020. 5

[12] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. YouTube-VOS: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018. 5

[13] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *Proceedings of the International Conference on Computer Vision*, 2021. 5, 6

[14] Tianfei Zhou, Shunzhou Wang, Yi Zhou, Yazhou Yao, Jianwu Li, and Ling Shao. Motion-attentive transition for zero-shot video object segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13066–13073, 2020. 3, 4, 5, 6