Dense Unsupervised Learning for Video Segmentation
– Supplemental Material –

Nikita Araslanov[1] **Simone Schaub-Meyer**[1] **Stefan Roth**[1,2]

[1]Department of Computer Science, TU Darmstadt [2]hessian.AI
{nikita.araslanov, simone.schaub, stefan.roth}@visinf.tu-darmstadt.de

A Overview

This supplemental material provides further details on training, the label propagation algorithm used at inference time, as well as additional qualitative examples.

B Training details

Our feature encoder has a ResNet-18 [46] architecture consisting of four residual blocks. Similarly to CRW [17], we remove the strides in the `res3` and `res4` blocks. To produce the embeddings, we additionally pass the output from `res4` through a multilayer perceptron (MLP). The MLP contains the standard `Conv1x1-BatchNorm-ReLU` block, which preserves the feature dimensionality (512), followed by a `Conv1x1` operation, reducing the feature dimensionality to 128. We experimented with other MLP architectures as well, including replacing Batch Normalisation (BN) [47] with the Layer Normalisation [43] layers, but found the effect on the final accuracy insignificant.

In each training epoch we sample only one video set (of size T, *cf.* main text) per video in the complete dataset. In total, our training requires $150K - 300K$ iterations for convergence (depending on the training data, *cf.* Table 3), which is only a fraction of the training time required by other unsupervised methods (*e. g.*, 2M iterations in [17, 19]). The computational footprint per iteration is comparable. As in previous work, we train with an input resolution of 256×256. CRW [17] samples 20 video clips with the length of 4 yielding $B \times T = 20 \times 4 = 80$ frames per forward pass, which is equivalent to our setup of using $B = 16$ video sets $T = 5$ frames each. MAST [19] uses a smaller batch size of 24 in the first 1M iterations, but processes frames with the output stride reduced by a factor of 2, hence the forward pass alone increases both the memory and the computational overhead by at least the same factor. For validation and model selection, we use 5 video sequences selected randomly from DAVIS-2017 *valid*.

B.1 Implementation

Fig. 5 demonstrates the pseudo code of the learning algorithm. We observe that our algorithm shares the simplicity of other unsupervised learning algorithms [8, 44, 50], and, in contrast to CRW [17], learns temporally consistent embeddings without iterative structures. All operations support efficient implementation in popular libraries, such as PyTorch. Note that the space-time loss term does not propagate the gradient to the regularising branch. This allows for memory-efficient implementations by means of always maintaining the regularising branch in "evaluation mode". We achieve this by simply using the main branch to also process the reference frame (one frame per video), for which the gradient is required for the cross-view consistency, and then re-combining the output into the same-sized k and $\hat{k}$. As a result, our framework has a reduced memory footprint compared to Siamese architectures [50].

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

```
# Main training loop
# B: batch size; T video length;
for x in loader:
  # x_hat: cropped and flipped x
  # 𝒯: similarity transform
  x_hat, 𝒯 = random_transform(x)

  # feature embeddings [BTxKxHxW]
  k, k_hat = net(x), net(x_hat)

  ℒ_ST = space_time_loss(k,k_hat,𝒯,N)
  ℒ_CV = cross_view_loss(k,k_hat,T,𝒯,M)
  loss = ℒ_ST + λ ℒ_CV

  loss.backward()
  optimizer.step()

# helper function
def affinity(x,y,τ)
  z = einsum("nk,bkhw->bnhw", x, y)
  return softmax(z / τ,1)
```

(a) Main loop

```
def space_time_loss(k, k_hat, 𝒯, N):
  # sampling anchors [BNNxK]
  q = grid_sample(k, N)

  # compute affinities  [BTxBNNxHxW]
  v = affinity(q, k, τ)
  v_hat = affinity(q, k_hat, τ)

  # compute pseudo labels
  u_hat = (v_hat * block_ones).argmax(1)
  loss = cross_entropy(T(v), u_hat)
  loss[::T].zero_()
  return loss.mean()

def cross_view_loss(k, k_hat, T, 𝒯, M):
  # subsampling features [BMMxK]
  r = grid_sample(T(k[::T]), M)
  r_hat = grid_sample(k_hat[::T], M)
  # affinity [BMMxBMM]
  h = softmax(mm(r, r_hat.t()) / τ, 1)
  return -log(diag(h)).mean()
```

(b) Loss functions

Figure 5: Pseudo code implementation of our method. `block_ones` is a $BT \times BN^2$ block-diagonal matrix containing B blocks of all-ones matrices of size $T \times N^2$. `diag` selects diagonal elements from a matrix. `mm` denotes matrix multiplication.

B.2 Data augmentation

We experimented with rotations and sheering initially, but found their benefits insignificant at the cost of increased implementation complexity. The main disadvantage of these transformations is the need to carefully handle the boundaries: both rotation and sheering require image padding, which needs to be removed post-hoc; otherwise, fully convolutional training would result in a degenerate solution.

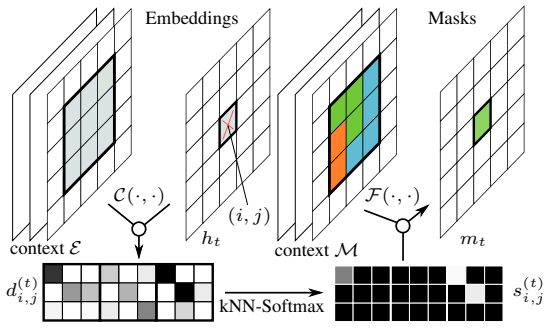
We also experimented with two versions of incorporating appearance-base augmentation (*e. g.*, colour jittering). On one hand, using frame-level augmentation (*i. e.* different augmentation per frame), we observed a decrease in VOS accuracy. We hypothesise that artificial augmentation techniques, such as sudden changes in contrast, saturation, or scale, poorly reflect natural transformations occurring in real-world video sequences. Using different augmentation per frame may also challenge a meaningful association between the anchors, extracted from one frame, and the features from the other frames, especially at the beginning of training, on which we rely for generating the pseudo labels in self-training. On the other hand, using video-level augmentation (*i. e.* the same augmentation for every frame, but different across video clips), we did not observe a significant change in accuracy. This is expected, since the framework would be additionally required to cope with distinguishing visually perturbed video clips (following our implementation of Assumption 2), which does not provide useful information for VOS.

C Inference

Fig. 6 illustrates the label propagation algorithm we use in our experiments. To predict mask m_t for the current timestep t, we make use of *context* embeddings and masks, accumulated from the previous frames. Following [17], we use the output from the penultimate residual block `res4` to obtain the embedding for frame t, denoted here as $h_t \in \mathbb{R}^{h,w,K}$, where h, w are the spatial dimensions and K is the dimension of the feature embedding. An embedding context $\mathcal{E}_t = \{h_0, h_{t-N_T+1}, ..., h_{t-1}\}$, $|\mathcal{E}_t| = N_T$, maintains embedding h_0 of the first frame,[1] which has ground-truth annotations, and the embeddings of $N_T - 1$ preceding frames. Similarly, we define the mask context as $\mathcal{M}_t = \{m_0^*, m_{t-N_T+1}, ..., m_{t-1}\}$, where m_0^* is the provided ground-truth annotation for the first frame, and $m_{t>0}$ are the masks propagated by the algorithm detailed next.

We first compute the cosine similarity of embedding $h_t(i, j)$ *w. r. t.* all embeddings in context $\mathcal{E}$, restricted to a spatio-temporal neighbourhood of size $N_T \times P \times P$, spatially centred on location

[1] In the case of YouTube-VOS, there may be multiple reference frames corresponding to objects appearing mid-sequence. Here, we handle the case of the first frame specifying all objects at once as an illustrative example.



(a) Label propagation

```
# initialise ℰ and ℳ
# from the first frame
for t,frame in enumerate(frames):
  h_t = net(frame)
  # local spatial correlation
  d_t = 𝒞(ℰ,h_t)
  # softmax b/w N_K nearest neighbours
  s_t = knn_softmax(d_t,N_K,τ)
  # mask prediction for timestep t
  m_t = ℱ(ℳ,s_t)
  # updating context
  ℰ,ℳ = add(ℰ,h_t),add(ℳ,m_t)
```

(b) Pseudo code of label propagation

Figure 6: **Label propagation.** We use the spatial correlation operator $\mathcal{C}(\cdot,\cdot)$ to implement a local attention operation by computing cosine similarities between the embedding at location (i,j) at timestep t and the embeddings in context $\mathcal{E}$ considering only the spatial neighbourhood $P \times P$ of the (i,j). Subsequently, we use local guided filtering operator $\mathcal{F}(\cdot,\cdot)$ to propagate the masks from context $\mathcal{M}$ using the computed local attention $s_{i,j}^{t}$. Example *(a)* illustrates this process for $P=3$ and context size $N_T=3$. Pseudo code *(b)* provides a general outline of the label propagation algorithm. We refer to the text for more details.

(i,j), which we denote as $\mathcal{N}_P(i,j)$. Three coordinates (t^*,l,n) for the temporal (first) and the spatial (second and third) dimensions specify the neighbour locations in $\mathcal{N}_P(i,j)$. Next, we create a new nearest-neighbour set $\mathcal{N}_P^{(t)}(i,j)$ by selecting N_K locations from $\mathcal{N}_P(i,j)$ with the highest cosine similarity, and compute local attention in a single operation, denoted in Fig. 6 as `kNN-Softmax`, as follows:

$$s_{i,j}^{(t)}(t^*,l,n) = \begin{cases} \frac{\exp\left(-d_{i,j}^{(t)}(t^*,l,n)/\tau\right)}{\sum_{(t',l',n')\in\mathcal{N}_P^{(t)}(i,j)}\exp\left(-d_{i,j}^{(t)}(t',l',n')/\tau\right)}, & \text{if } (t^*,l,n)\in\mathcal{N}_P^{(t)}(i,j) \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $d_{i,j}^{(t)}(t^*,l,n)$ is the cosine similarity between embeddings $h_t(i,j)$ and $h_{t^*}(l,n)$ from $\mathcal{E}$; and τ is the temperature hyperparameter set to 0.05, as in training (*cf.* Sec. 4). We compute the mask m_t as a weighted sum of the mask predictions at locations $\mathcal{N}_P(i,j)$ as

$$m_t = \sum_{(t^*,l,n)\in\mathcal{N}_P(i,j)} s_{i,j}^{(t)}(t^*,l,n)\, m_{t^*}(l,n), \tag{2}$$

where $m_{t^*}(l,n)$ comes from the mask context $\mathcal{M}$. Finally, we add m_t and h_t to the mask and embedding contexts, $\mathcal{E}$ and $\mathcal{M}$, respectively, and remove the oldest entries (with exception of the first reference frames) to maintain their constant size of N_T. We repeat this process for the remaining frames in the video clip. Fig. 6b outlines this algorithm using pseudo code. Note that we initialise $\mathcal{M}$ and $\mathcal{E}$ by simply replicating the masks and the embeddings of the first reference frame.

For consistency we use the label propagation implementation provided by Jabri et al. [17]. The first operator $\mathcal{C}(\cdot,\cdot)$ is a local spatial correlation operation commonly used in correlation layers of optical flow networks [49]. The second operator $\mathcal{F}(\cdot,\cdot)$ implementing Eq. (2) is a variant of local guided filtering [45], also used in pixel-adaptive convolutional networks [48]. We set $P=25$ and $N_K=10$ in our experiments with DAVIS-2017 [35] and YouTube-VOS [42]. We use a context size of $N_T=20$ for DAVIS-2017. For YouTube-VOS, where instances may appear in intermediate frames, we use a separate context for each object ID. We use the original resolution of both DAVIS-2017 and YouTube-VOS benchmarks, which is downscaled by a factor of 8 in our embedding by the feature extractor. We upsample the final object masks to the original resolution with bilinear interpolation.

D Qualitative examples

We include videos demonstrating the qualitative results in the supplementary material. The segmentation results produced by our method show clear improvements over CRW [17], despite using only a fractional amount of time and data for training. Our model used for producing DAVIS-2017

examples was trained for 150K iterations on only 4.5K videos with a total duration of 5 hours, while the CRW [17] model was trained for 2M iterations on Kinetics-400 containing 300K videos spanning 833 hours. On the YouTube-VOS benchmark, we show the results from our model trained for 200K iterations on TrackingNet, which is still about 10 times smaller than Kinetics-400. Both methods tend to produce decent segmentation quality despite self-occlusions and complex transformations in the videos. By contrast, MAST tends to produce masks of poorer quality, especially when there is ambiguity in the appearance of the foreground object and the background. Nevertheless, both our approach and CRW may struggle in videos with large spatial displacements of the tracked object. We hypothesise that this may be in part due to the spatial bias introduced by the memory context (discussed in Appendix C), which considers only a spatially local neighbourhood for mask propagation. We also observe that our approach clearly surpasses a surprisingly strong baseline model with random weight initialisation.

E License note

The parts of the code we use from Jabri et al. [17] (the label propagation algorithm) are released under a MIT license. The datasets YouTube-VOS, OxUvA, TrackingNet, and Kinetics-400 are licensed under the Creative Commons Attribution 4.0 International License, while DAVIS-2017 is provided under the Creative Commons Attribution-NonCommercial 4.0 International License.

References

[43] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450 [stat.ML]*, 2016.

[44] X. Chen and K. He. Exploring simple siamese representation learning. In *CVPR*, pages 15750–15758, 2021.

[45] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE T. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013.

[46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[47] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[48] H. Su, V. Jampani, D. Sun, O. Gallo, E. G. Learned-Miller, and J. Kautz. Pixel-adaptive convolutional neural networks. In *CVPR*, pages 11166–11175, 2019.

[49] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018.

[50] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In M. Meila and T. Zhang, editors, *ICML*, volume 139, pages 12310–12320, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss the most important limitations in Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss this briefly in Sec. 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We release our code and pre-trained models in a public repository: `https://github.com/visinf/dense-ulearn-vos`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide the training details in Sec. 4 and elaborate on further details in Appendix B and Appendix C (*e. g.*, the label propagation algorithm).
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the standard deviation of the $\mathcal{J}\&\mathcal{F}_m$ score across 5 training runs in the beginning of Sec. 4.3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We provide these details at the beginning of Sec. 4 and Table 3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We specify the license of the datasets and the code we use in Appendix E.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include a video demonstrating the qualitative results produced by our method in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use publicly available datasets in full compliance with their terms of use.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We ensure that the qualitative examples displayed in our work do not contain personally identifiable artefacts.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]