"Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks"

Frederick Spreafichi

University of Trieste

June 3, 2025

Outline

Problem Description

Zero-shot video object segmentation (ZVOS) is the task of automatically identifying and segmenting the main foreground objects in a video without any prior object-category labels or manual masks at test time, making it a fully unsupervised and more challenging problem compared to semi-supervised methods that rely on an initial ground-truth mask.

Problem Description

Formally, let

$$I = \left\{ I_i \in \mathbb{R}^{H \times W \times 3} \;\middle|\; i = 1, \ldots, N \right\}$$

be a sequence of N color frames (height H, width W). The goal is to predict a **Segmentation Mask**

$$S = \left\{ S_i \in \{0,1\}^{H \times W} \;\middle|\; i = 1, \ldots, N \right\}$$

where

$$S_i(x,y) = \begin{cases} 1 & \text{if pixel } (x,y) \text{ belongs to a primary object (foreground)} \\ 0 & \text{if pixel } (x,y) \text{ is background} \end{cases}$$

Graph Attention Networks (GATs)

- **GATs**
  - Weights neighbor-node contributions instead of uniform aggregation
  - **Assign higher importance to more relevant neighbors** during message passing
  - **Dynamically adjust neighbor's influence** for more expressive node representations
- **AGNN Extension**
  - **Adapts attention to pixel-wise relations** within and between video frames
  - **Captures higher-order dependencies** for zero-shot video object segmentation

AGNN for Zero-Shot Video Object Segmentation

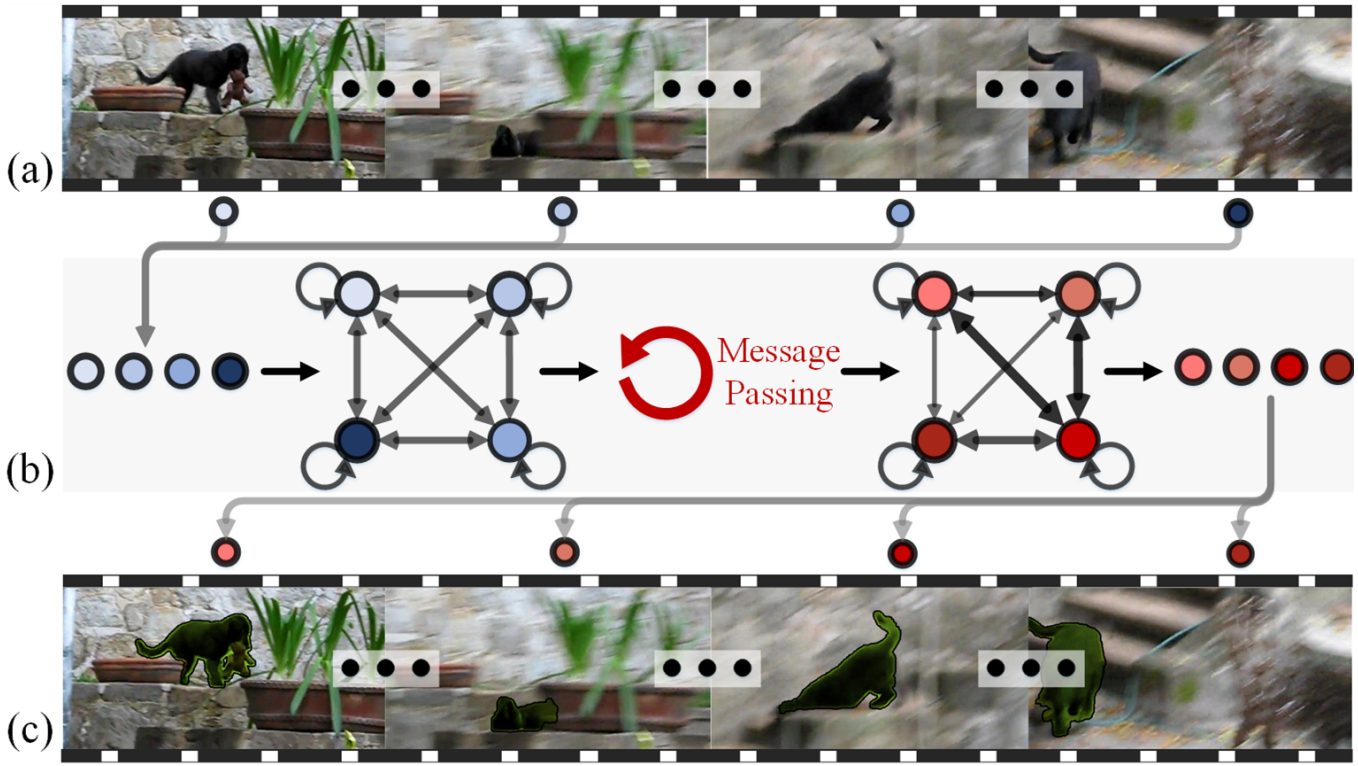

Figure: AGNN for ZVOS

Network Architecture

1. FCN-Based Node Embedding
2. Intra-Attention Based Loop-Edge Embedding
3. Inter-Attention Based Line-Edge Embedding
4. Gated Message Aggregation
5. ConvGRU Based Node-State Update
6. Read-Out Function

FCN-Based Node Embedding - DeepLabV3

- Use **DeepLabV3** (a Fully Convolutional Network for semantic segmentation) to extract initial node embeddings:

$$h_i^0 = v_i = F_{\text{DeepLab}}(I_i) \in \mathbb{R}^{H \times W \times C}$$

- **Embeddings preserve spatial structure and high-level semantic features**, essential for pixel-wise segmentation
- **Atrous (dilated) convolutions enlarge the receptive field** without losing resolution
- **Atrous Spatial Pyramid Pooling** (ASPP) applies parallel dilated convolutions at multiple rates to **capture multi-scale context**
- Outcome: detailed multi-scale feature maps enabling precise spatial analysis

DeepLabV3

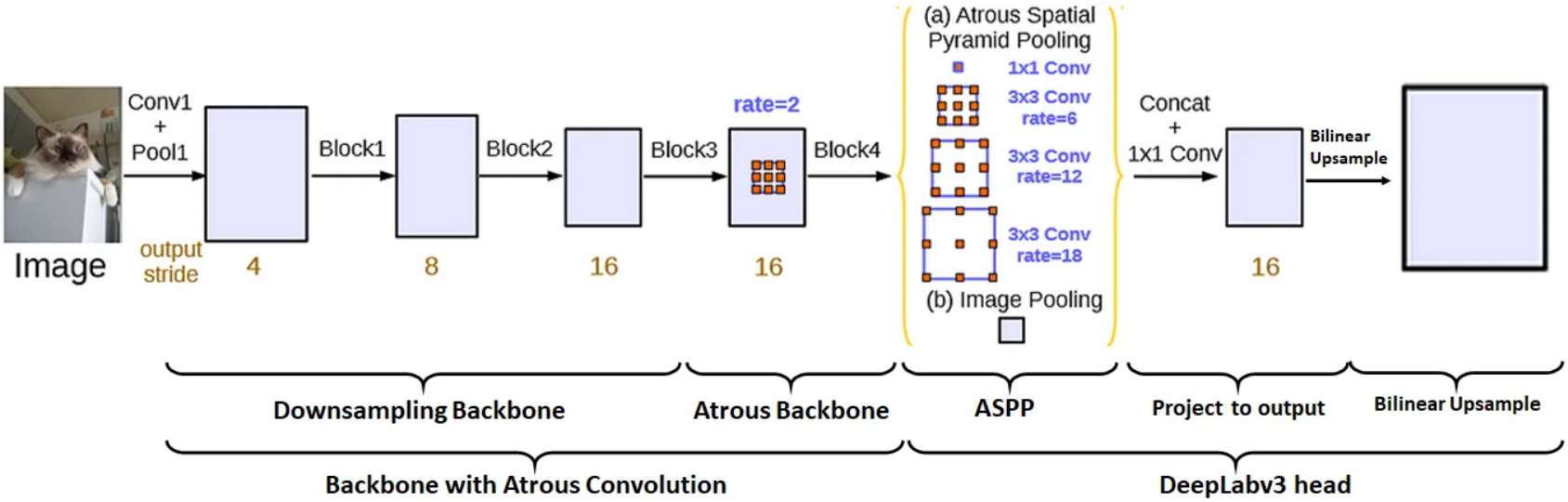
Image
Conv1
+
Pool1
output
stride
4
Block1
8
Block2
16
Block3
rate=2
16
Block4
(a) Atrous Spatial
Pyramid Pooling
1x1 Conv
3x3 Conv
rate=6
3x3 Conv
rate=12
3x3 Conv
rate=18
(b) Image Pooling
Concat
+
1x1 Conv
16
Bilinear
Upsample
Downsampling Backbone
Atrous Backbone
ASPP
Project to output
Bilinear Upsample
Backbone with Atrous Convolution
DeepLabv3 head


I_i
w×h×c
$F_{DeepLab}$
W×H×C
v_i

(a)

Intra-Attention Based Loop-Edge Embedding

- A **loop-edge $e_{i,i}$ connects node v_i to itself** in the AGNN graph
- Its **embedding $e^k_{i,i}$** is computed by:

$$e^k_{i,i} \;=\; \alpha \cdot \mathrm{SoftMax}\big((W_f * h^k_i)\,(W_h * h^k_i)^T\big)\,(W_\ell * h^k_i) \;+\; h^k_i$$

- **Intra-attention models long-range dependencies within the same frame**
- Complements convolution by letting distant but semantically related regions influence each other
- α is a learnable scalar; W_f, W_h, W_ℓ are convolution kernels; "$*$" denotes convolution

Inter-Attention Line-Edge Embedding

- A **line-edge $e_{i,j}$ connects node v_i to node v_j**
- Inter-attention computes:

$$\begin{cases} e^k_{i,j} &= h^k_i \, W_c \, (h^k_j)^T \\ e^k_{j,i} &= \left(e^k_{i,j}\right)^T \end{cases}$$

- $W_c \in \mathbb{R}^{C \times C}$ learnable weight matrix; $h^k_i \in \mathbb{R}^{(W\,H) \times C}$ flattened feature tensors
- **The embedding $e^k_{i,j}$ measures how strongly v_i influences v_j during message passing**

Attention Mechanisms

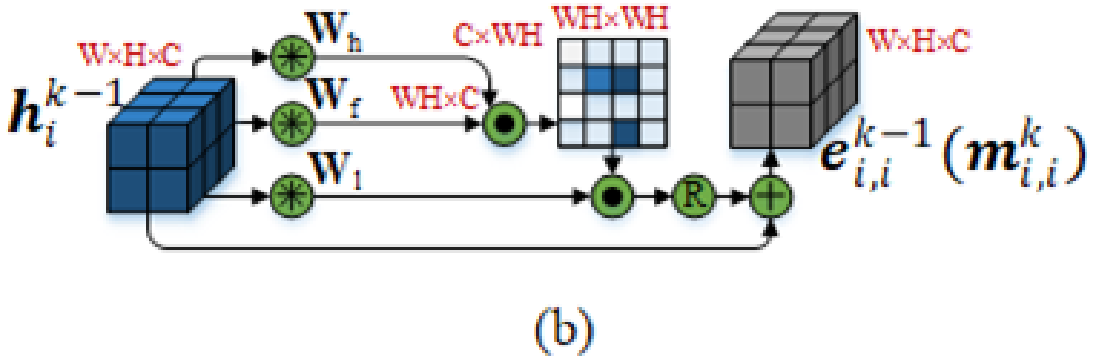

Figure: intra-attention

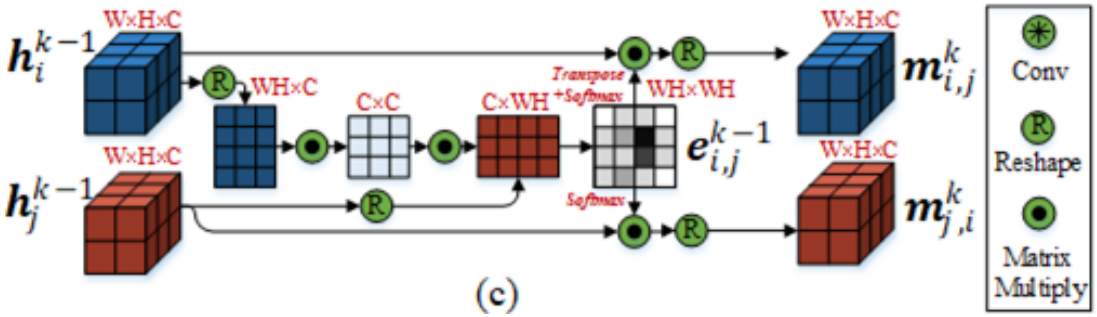

Figure: inter-attention

Gated Message Aggregation

- **Self-Loop Message**

$$m_{i,i}^{k} = e_{i,i}^{k-1}$$

- **Neighbor Message**

$$m_{j,i}^{k} = M\left(h_{j}^{k-1}, e_{j,i}^{k-1}\right) = \mathrm{SoftMax}\left(e_{j,i}^{k-1}\right) h_{j}^{k-1}$$

  - $\mathrm{SoftMax}(e_{j,i}^{k-1})$ normalizes each row of the edge embedding, turning it into weights over spatial positions
  - After computing, reshape $m_{j,i}^{k}$ back into a $W \times H \times C$ tensor

- Aggregating messages lets **each node gather information from itself and neighbors**
- Attention ensures **nodes focus only on the most useful parts of a neighbor's features**
- Crucial when some frames have occlusions or irrelevant details

Gated Message Aggregation

- **Learnable Gating** After forming each message $m_{j,i}^k \in \mathbb{R}^{W \times H \times C}$, we compute a confidence score per channel:

$$g_{j,i}^k = G(m_{j,i}^k) = \sigma\Big(\mathrm{GAP}\big(W_g * m_{j,i}^k + b_g\big)\Big) \in [0,1]^C$$

  - $\mathrm{GAP}(\cdot)$ applies global-average pooling over spatial dimensions, **summarizing each channel into one scalar**
  - A sigmoid $\sigma(\cdot)$ then maps these summaries into $[0,1]$, producing **one gating coefficient per channel**
  - Channels with higher $g_{j,i}^k(c)$ are considered more reliable and will pass through

Gated Message Aggregation

- **Gated Aggregation** Combine all self-loop and neighbor messages, but modulate each by its gate:

$$m_i^k = \sum_{v_j \in \mathcal{V}} g_{j,i}^k \odot m_{j,i}^k$$

where $\odot$ denotes channel-wise (Hadamard) multiplication:
$\big(g \odot m\big)(x, y, c) = g(c)\, m(x, y, c)$

- **Why include gating?**
  - **Not all neighbors are equally informative**—some may be noisy (occluded, out-of-focus)
  - The gate learns to down-weight unreliable channels, **improving robustness**
  - Aggregating only the "trusted" features leads to **cleaner node updates** and better performance on noisy video frames

ConvGRU-Based Node-State Update

- At iteration k, given previous state h_i^{k-1} and aggregated message m_i^k:

$$h_i^k \;=\; U_{\mathrm{ConvGRU}}\left(h_i^{k-1},\; m_i^k\right) \;\in\; \mathbb{R}^{W\times H\times C}$$

- **GRU gates decide how much past vs. new information to use**
- Convolutional layers replace dense multiplications, **preserving spatial structure**
- **Gates:**
  - **Update gate:** Controls incorporation of new message
  - **Reset gate:** Controls forgetting of previous state when computing candidate

ConvGRU Details

- **Convolutional Gating:**
  - Reset, update, and candidate computations use convolutional filters (3×3 or similar).
  - Same filters slide over all spatial positions—weights shared across $W \times H$.
- **Benefits:**
  - *Spatial consistency:* Neighboring pixels are updated uniformly.
  - *Parameter efficiency:* Far fewer learnable parameters than fully connected GRU.
  - *Iterative refinement:* Node embeddings gradually integrate new messages while retaining memory.
  - ConvGRU ensures pixel-level relationships remain coherent across iterations.

Read-Out Function

- After K message-passing iterations, use final state h_i^K (and initial h_i^0) to find the **predicted segmentation mask**:

$$\hat{S}_i \;=\; \mathrm{R}_{\mathrm{FCN}}\big([\,h_i^K,\,h_i^0\,]\big) \;\in\; [0,1]^{W\times H}$$

- **Architecture:**
  - Two 3×3 convolutional layers (with non-linear activation) to refine features
  - One 1×1 convolutional layer followed by sigmoid to produce a normalized pixel-wise mask
- **Key Point:** Implemented as a **lightweight FCN to preserve spatial resolution** and ensure fine details in the segmentation

AGNN Network

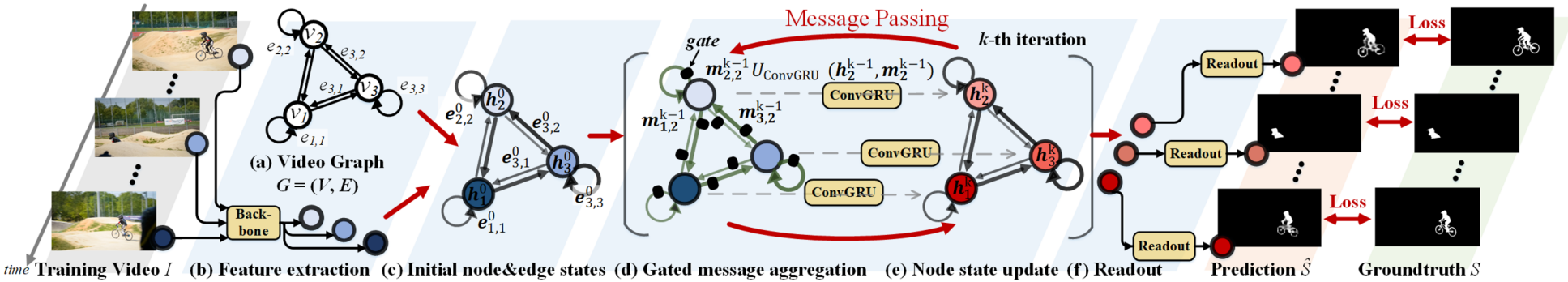

Figure: AGNN during training

Loss Function

- **Pixel-level weighted binary cross-entropy**

$$L(S, \hat{S}) = -\sum_{x=1}^{W \times H} \Big[(1-\eta)\, S_x \log\big(\hat{S}_x\big) + \eta\,(1-S_x) \log\big(1-\hat{S}_x\big) \Big].$$

- Where:
  - $S_x \in \{0, 1\}$: ground-truth label at pixel x
  - $\hat{S}_x \in [0, 1]$: predicted probability at pixel x
  - η: ratio of foreground to background pixels in S
- **Encourages the model to focus on correctly segmenting objects rather than defaulting to background**
- Weighted loss helps **AGNN learn robust object-level features**

Environment Details

- **GPU:**
  - NVIDIA GeForce RTX 3060
  - 12 GB GDDR6 VRAM
  - Driver 572.61
- **Python:** 3.10
- **PyTorch:** 11.13
- **CUDA:** 11.6

Dataset and Metrics

- **Dataset:** *DAVIS17* — A benchmark for video object segmentation, featuring high-resolution video sequences with pixel-level annotations for multiple objects across frames
- **Region Similarity J (IoU):** Average Intersection-over-Union between each predicted mask and ground truth, computed over all frames
- **Boundary Accuracy F:** Contour F-measure between predicted and GT boundaries, using a 2-pixel tolerance to judge boundary alignment
- **Temporal Stability T:** Defined as $1 - \frac{|M_t \oplus M_{t+1}|}{|G_t \cup G_{t+1}|}$, where $\oplus$ is the symmetric difference between consecutive predicted masks M_t, M_{t+1}, and normalization is by the union of their ground truth masks G_t, G_{t+1}

Training Setup

- **Sampling Strategy:**
  - Split each training video (length N frames) into N' segments
  - Randomly select one frame from each segment for training
- **Implementation Details:**
  - Due to computational limits, select 2 training videos per batch
  - Set $N' = 3$ (one frame from each of 3 segments)
  - Number of message-passing iterations $K = 3$
  - Trained for 100 epochs total

Training

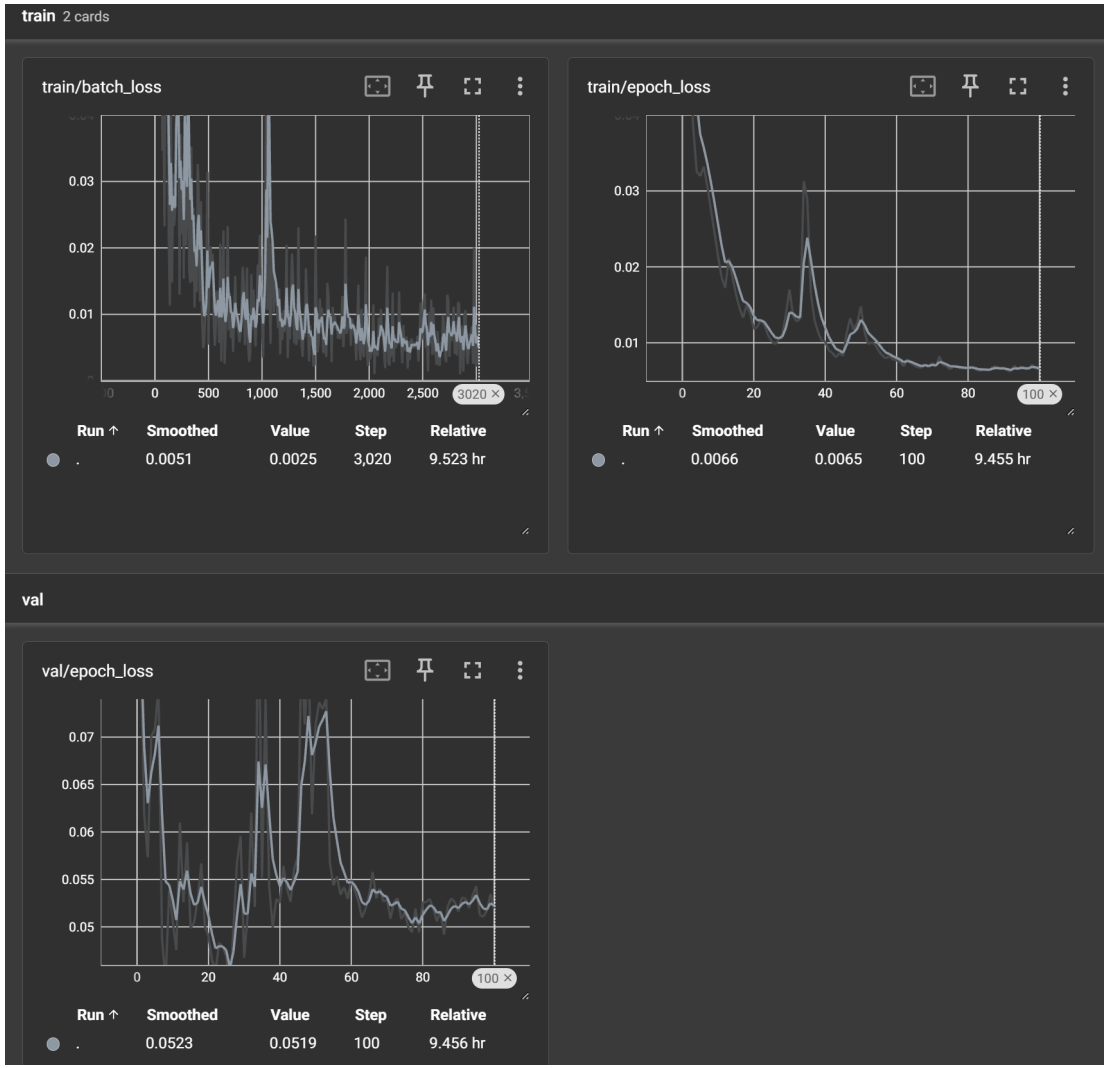

Figure: Training and validation loss

Training

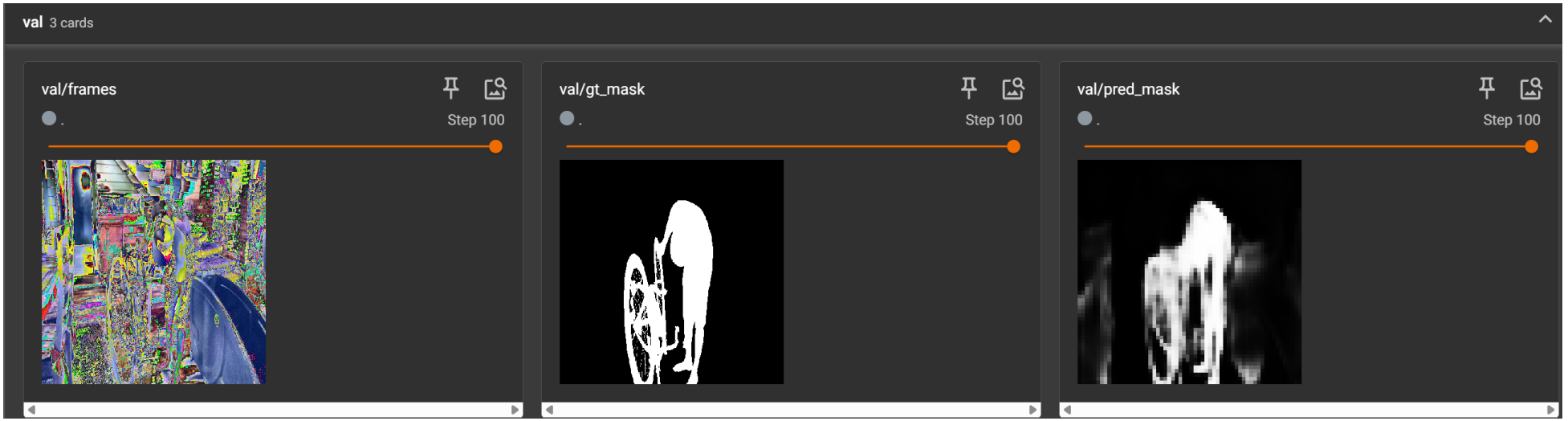

Figure: Validation

Testing Setup

- Select an input video I of N frames at 473×473 resolution
- Split I into $T = \frac{N}{N'}$ subsets: $\{I_1, \ldots, I_T\}$
- Each subset I_t contains N' frames:

$$\{ I_t,\ I_{t+T},\ \ldots,\ I_{N-T+t} \}$$

- Feed each subset into AGNN to obtain segmentation maps for its frames
- **Apply Conditional Random Field (CRF) as post-processing to refine boundaries**

Examples

Figure: Frame from Balckswan video: $\bar{J} = 0.886$; $\bar{F} = 0.701$

Figure: Frame from Breakdance $\bar{J} = 0.317$; $\bar{F} = 0.190$

Results Summary

	$\bar{J}$	$\bar{F}$	$\bar{T}$
Overall Mean	**0.635**	**0.406**	**0.723**

Video	$\bar{J}$	$\bar{F}$	$\bar{T}$
blackswan	0.886	0.701	0.906
lab-coat	0.919	0.637	0.884
cows	0.827	0.507	0.945
car-shadow	0.770	0.455	0.875
breakdance	0.317	0.190	0.120
india	0.179	0.200	0.338

Interpreting the Results

- **High-Scoring Videos (e.g., blackswan, lab-coat, cows):**
  - Large, high-contrast foreground occupies most of the frame
  - Minimal occlusion and smooth, predictable motion
  - Attention-based GNN/CRF quickly "locks on" to the object
- **Low-Scoring Videos (e.g., breakdance, india):**
  - Rapid, erratic motion and cluttered backgrounds (breakdance) or very small, low-contrast objects (india)
  - Fixed receptive field struggles to extract stable foreground signals under these conditions
- **Boundary vs. Region Scores:**
  - Even when $\bar{J}$ is in the 0.7–0.8 range, $\bar{F}$ can be only 0.27–0.36
  - High IoU (region) but mask contours often lie several pixels away from ground-truth edges, lowering the boundary F-measure

Peper's Results: DAVIS17 & YouTube-Objects

Table: DAVIS17 Test-Dev

Method	$\bar{J}$	$\bar{F}$	$\overline{J\&F}$
RVOS [63]	39.0	42.8	43.7
AGNN	**58.9**	**65.7**	**61.1**

Table: YouTube-Objects Per-Category (mean F scores)

Method	Airplane (6)	Bird (6)	Boat (15)	Car (7)	Cat (16)	Cow (20)	Dog (27)	Horse (14)	Motorbike (10)	Train (5)	Avg.
FST [43]	70.9	70.6	42.5	65.2	52.1	44.5	65.3	53.5	44.2	29.6	53.8
COSEG [60]	69.3	76.0	53.5	70.4	66.8	49.0	47.5	55.7	39.5	53.4	58.1
ARP [27]	73.6	56.1	57.8	33.9	30.5	41.8	36.8	44.3	48.9	39.2	46.2
LVO [58]	86.2	81.0	68.5	69.3	58.8	68.5	61.7	53.9	60.8	66.3	67.5
PDB [55]	78.0	80.0	58.9	76.5	63.0	64.1	70.1	67.6	58.3	35.2	65.4
FSEG [21]	81.7	63.8	72.3	74.9	68.4	68.0	69.4	60.4	62.7	62.2	68.4
SFL [7]	65.6	65.4	59.9	64.0	58.9	51.1	54.1	64.8	52.6	34.0	57.0
AGS [69]	87.7	76.7	72.2	78.6	69.2	64.6	73.3	64.4	62.1	48.2	69.7
AGNN	81.1	75.9	70.7	78.1	67.9	69.7	77.4	67.3	68.3	47.8	**70.8**

Conclusions

- **State-of-the-art performance:** AGNN outperforms AGS on DAVIS16 ($\bar{J} = 80.7$ vs. 79.7, $\bar{F} = 79.1$ vs. 77.4) and RVOS on DAVIS17 ($\bar{J} = 58.9$ vs. 39.0, $\bar{F} = 65.7$ vs. 42.8)
- **Robust feature learning:** Attention-based graph embeddings capture long-range spatial dependencies; ConvGRU preserves spatial structure and temporal consistency across iterations
- **Noise resilience:** Gated message aggregation filters out unreliable or occluded neighbors, improving robustness on cluttered or rapidly moving sequences
- **Consistent across categories:** Competitive per-category F on YouTube-Objects (Avg. 70.8), handling both simple backgrounds and complex motions without finetuning
- **Efficient design:** Lightweight, single-model architecture requiring no sequence-specific tuning and maintaining good temporal stability ($\bar{T} \approx 0.72$)