OAGCNN for Video Object Segmentation

Adrià Caelles[1], Carles Ventura[1], Andreu Girbau[1], Laura Leal-Taixè[2], Xavier Giró[1]
[1]Universitat Politècnica de Catalunya , [2]Technische Universität München
adria.caelles@estudiantat.upc.edu

Abstract

Video object segmentation has increased in popularity since the release of Davis2016 in which a single object had to be segmented. With the release of Davis2017 and YouTube-VOS the task moved to multiple object segmentation, increasing in difficulty. In this work we focus on this scenario, presenting a novel graph convolutional neural network that has the sense of each object of the video sequence in each node, working entirely in the feature space domain. The nodes are initialized using an encoder that takes as input features from the image together with each object's mask. After graph message passing, we use a decoder on each final node state to segment the object that node is referring to.

1. Introduction

Video object segmentation aims to detect and segment objects from the scene. One of the first datasets, Davis2016, a single object was given for the entire sequence. In recent datasets, Davis2017 and YoutubeVOS one or more objects are introduced, making the task more challenging. First proposed methods tried to give emphasis on the appearance of the object [2], finetuning the model at the start of the sequence to quickly learn what it had to segment, without taking into account temporal consistency of the video. Other methods focus on this aspect leveraging RNN [8], but it has been shown that these architectures are unstable to train and have been recently outperformed by Transformers [7]. Recent work has adapted the idea of Attention [6] showing outstanding results compared to previous methods, thus, exhibiting the power of these types of architectures. Graph Neural Networks have been recently proposed for VOS as well, but we will explain the presented architectures in more detail in section 2.1.

Inspired from the work of [9] together with [8] we propose a combined architecture that uses Graph Convolutional Neural Networks but giving the nodes the sense of each object in the video, instead of working in the whole image. Our presented graph is constructed in the spatial domain, and is proposed to give to each node the power to describe its object as best as possible. When nodes exchange messages, gathering information about all the nodes gives the graph the capacity that all the nodes agree to segment its particular object. Our proposed method also uses other modules in conjunction to work. More specifically, we have divided our architecture into four main components. First we extract features from the image using a CNN. Then we introduce a Mask Encoder that receives as input the extracted features and the mask from each object in the previous frame and with this initializes the nodes of the graph. The idea behind this module is to learn to align the mask from the previous frame with the new features from the current one. Apart from this, the graph also receives the hidden state of the corresponding node at the end of the message aggregation of the last frame. At this point we can perform K-message passing steps so each node is able to focus on its particular object using the information of other objects in the frame. When finished, we use another simple CNN on the state of each node to obtain the corresponding predicted mask of each object. Recent work from Liu et al. [4] presents a similar architecture as the one we are using, but we give the graph the power to work exclusively in the feature space domain instead of working with binary masks to be aggregated.

Our contributions can be summarized as follow:

- Our GCNN gives to the nodes the sense of each object in the feature space domain, instead of working in the frame domain or working with masks.

2. Related Work

2.1. Graph Neural Networks

Based on deep neural networks and graph theory, GNNs are powerful for collectively aggregating information from data represented in graph domains. They have recently been used with high success in VOS. As far as we know, the first adaptation of graphs on VOS was made by Wang et. al. [9]. The graph was constructed from the feature representation of consecutive frames and was able to extract relations between these feature tensors via attention mechanism. After

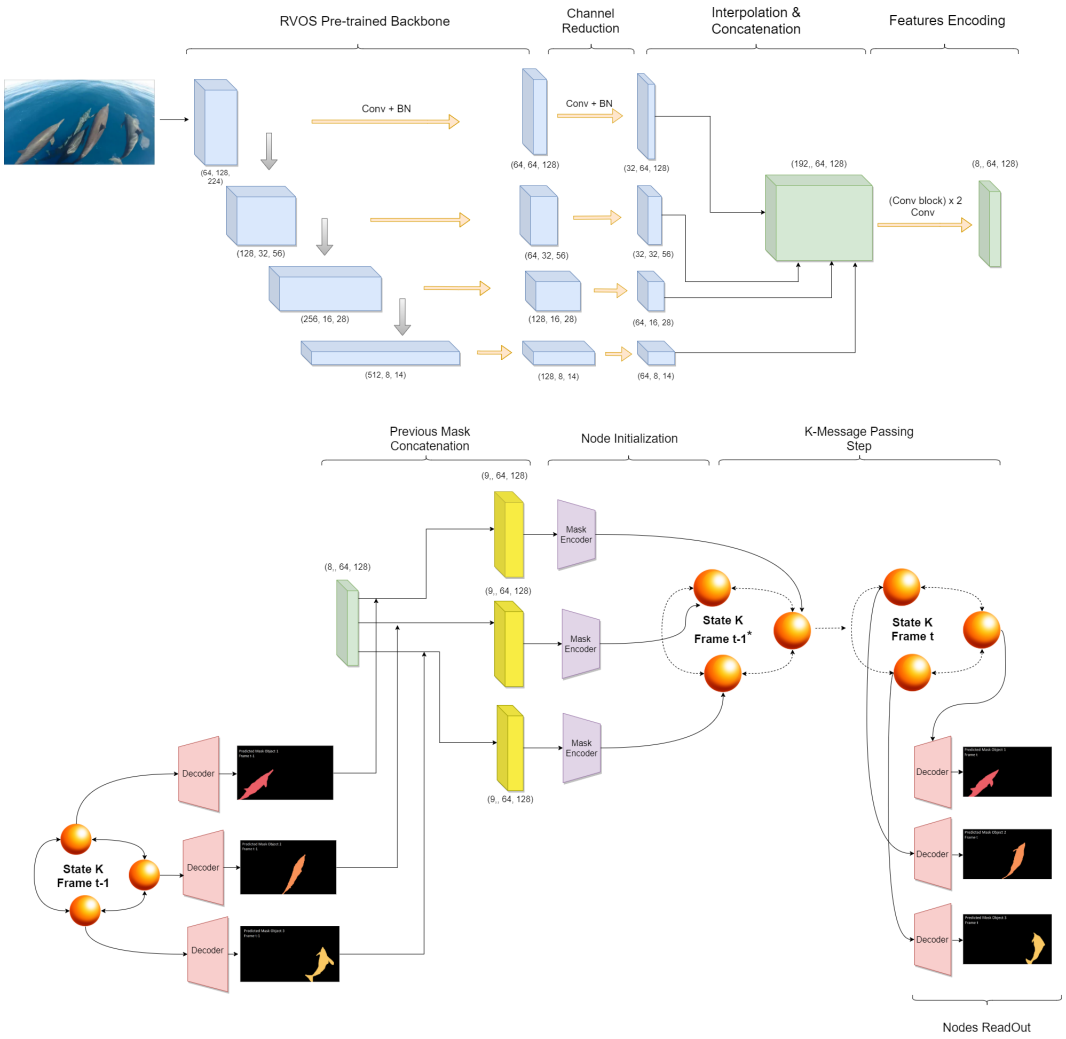

Figure 1: Our proposed architecture for video object segmentation for a single frame at time step t. Note that we use the predicted masks from previous frame t-1. **Figure illustrates state propagation, which means that final states from the previous frame are used as initial states to be updated by the first k iteration.*

K-message passing steps, a decoder module was applied in each final node (frame) in order to get the final masks from that set of frames. Similar idea was presented by Lu et. al. [5], which leverages a memory cell in conjunction with the graph in order to keep in mind all the frames that have been already seen in the video. With the K-message passing step,they learn how to segment the actual query image, based on the past segmented images. They also learn how to include the current image features in the memory cell. We can see this method as a fusion from the work of Oh et. al. [6] and Lu et. al. [5]. Finally, the most recent approach has been presented by Liu et. al. [4]. In this case, the graph aims to learn how to combine all the pieces of masks we have from a set of consecutive images to segment. The set of masks from each frame are computed combining mask propagation using Optical Flow from the previous frame together with an instance segmentation network that produces masks for each frame. The graph is used in order to refine and aggregate all the pieces of masks from all the nodes, thus, leveraging temporal coherence. Note that in this case graph nodes have the sense of object, different from both previous works that nodes represented frames. This work is the one that is more similar to the one we present. The major difference is that we aim to work on the feature domain in the graph, thus trying to give more freedom to learn what is best for the task.

Feat. ext.	Mask Enc.		Graph Neural Netowork									ReadOut	Performance
Out. dim.	Concat	Conv & Pool	K	Neigh. Agg.			State Agg.		Node Update		Temporal state passing	Skip conn.	Overall J&F
				Mean	Max	None	Sum	Conv	ReLU	GRU			
112x224x8	✓	✗	4	✓	✗	✗	✓	✗	✓	✗	✗	✗	**0.3035**
112x224x8	✓	✗	4	✗	✓	✗	✗	✓	✗	✓	✗	✗	**0.3847**
112x224x8	✓	✗	4	✗	✓	✗	✗	✓	✗	✓	✓	✗	**0.4031**
112x224x8	✗	✓	3	✗	✓	✗	✗	✓	✗	✓	✓	✗	**0.4338**
112x224x8	✗	✓	3	✗	✗	✓	✗	✓	✗	✓	✓	✗	**0.4401**

Table 1: Model performance comparison from different architectures approaches presented in 3. Overall J and F is the arithmetic mean from J_seen, J_unseen, F_seen, J_unseen from the YouTubeVOS validation split using the official evaluator server

3. Model

Our model has four major components: Feature Extractor, Mask Encoder, Graph Neural Network and finally, Decoder. At the moment we are focused on One-shot Video Object Segmentation, thus, the mask from the first appearance on the sequence of each object is given. The input of the model consists on the RGB image $I \in \mathbb{R}^{WxHx3}$ from the current frame and GT masks $S_{GT} \in \{0,1\}^N$, where N represents the number of objects.

3.1. Feature extractor

The aim of the feature extractor module is to receive as input the RGB image I from the current frame of the video sequence and obtain a feature tensor $F_t \in \mathbb{R}^{hxwxc}$ where $h, w \ll H, W$ and $c \gg C$. The first component of the Feature Extractor is a ResNet-101 [3] used to extract a set of features $f_t = \{f'_{t,1}, f'_{t,2}, ..., f'_{t,k}\}$ at different scales following the work from [8]. Then we use 1x1 convolutions to reduce the number of channels on each of the features and we resize them to concatenate all in a single tensor $F'_t = [f_{t,1}, f_{t,2}, ..., f_{t,k}]$. Finally we use a CNN composed by two sets of convolution, ReLU & Dropout layers together with a final 1x1 conv layer to obtain the final feature tensor of the image F_t

3.2. Mask Encoder

The mask encoder is the one responsible for initializing the nodes of the graph based on each object we have. It receives as input the output from the Feature Extractor F_t and a binary mask $S \in \{0,1\}^1$ for each of the N objects active on the video sequence (explained in detail in 3.2.1).We will use the ground truth mask when an object appears for the first time and otherwise we will use the predicted mask from the previous frame. We have studied two different architectures for this module: The first simply generates the h_t^0 that represents each object by simply concatenating the mask from that object with F_t (eq. 1). The second aggregates to the first one a convolution layer and a max pooling operation in order to obtain a new more compressed representation in the feature space domain of that object (eq. 2).

$$h_t^0 = [F_t, S] \in \mathbb{R}^{hxwx(c+1)xN} \quad (1)$$

$$h_t^0 = F_{GMP}(W_{encoder} * [F_t, S]) \in \mathbb{R}^{\frac{w}{2}x\frac{h}{2}x(c\times 2)xN} \quad (2)$$

where F_{GMP} indicates Global Max Pooling and ($W_{encoder}$ has a 3x3 kernel.

3.2.1 Active Object Tracker

As we are dealing with one-shot VOS we need to exactly know when an object enters in the sequence for the first time, as we will use the GT mask at that point. We have introduced an Active Object Tracker to our architecture that aims to look in each frame the objects we need to segment and look for new ground truth objects that may start appearing in the middle of the sequence. Note that during training it can only update new objects to segment and it can not deactivate objects that at some point do not appear any more, as it would not be a realistic scenario.

3.3. Convolutional Graph Neural Network

GNNs are a really powerful architecture for aggregating information from data represented in the graph domain. Thus, a critical aspect to make them work is how your data is characterized, this is, how you build the nodes and the edges. Our graph characterizes in each node the representation in the feature space of each object from a single frame of the sequence, extracted from the Mask Encoder module (3.2). Our graph is fully connected, which means that all the nodes are connected with each other. Note that our graph representation is in the spatial domain, as the K-message passing steps only occur with nodes that represent the objects of the same frame.The temporal consistency is given by the predicted mask propagation and the node state propagation.

The h_t^0 computed for each of the N objects of the frame represents the initial state we will have in each node. We will perform K-message passing steps now in the graph in order to update the representation of each node. Each of

this K-message passing step is composed by the following four different stages: Neighbour Aggregation, Intra Update, State Aggregation and Node Update. We will also present different architectures we have tested in each one. The results using different combinations are presented in Table 1

3.3.1 Neighbour Aggregation

The first step is to define how we aggregate each node neighbours in a single state. We have define two different methods. The first simply averages all the neighbour states (eq. 3). The second uses element-wise MAX operator over all the neighbours $j \in \{V_j\}$ of node i in order to take what is more descriptive of each one (eq. 4). Both are then feed to a convolutional layer with weight matrix W_{agg} to obtain the final feature representation.

$$m'^{\,k}_{\,i} = W_{agg} * (\frac{1}{\|V_j\| - 1} \sum_{j \in \{V_j \neq i\}} h_j^{k-1}) \tag{3}$$

$$m'^{\,k}_{\,i} = W_{agg} * \max_{j \in \{V_j \neq i\}} h_j^{k-1} \tag{4}$$

3.3.2 Intra update

We simply use a convolutional layer to compute a new representation of the node being update in order to learn what we want to transform from it (eq. 5).

$$h_i^{'\,k} = W_{self} * h_i^{k-1} \tag{5}$$

3.3.3 State aggregation

Now we need to combine what we have from the neighbours nodes 3.3.1 and itself new representation 3.3.2. We have studied two different approaches: simply summing both (eq. 6) or concatenating and applying a convolutional layer (eq. 7).

$$m_i^k = h_i^{'\,k} + m_i^{'\,k} \tag{6}$$

$$m_i^k = W_{state} * [h_i^{'\,k},\ m_i^{'\,k}] \tag{7}$$

3.3.4 Node update

With the computed m_i^k we can update each node new state. We have experimented along two different approaches in this step as well: using a non linear ReLU activation on the message (eq. 8) or leveraging a ConvGRU cell [1] in order to preserve the appearance of each object inducted by the Mask Encoder (eq. 9). With this second approach we also bring the possibility to use when $K = 0$ the final state from that object on the previous frame $h_i^K, t-1$ as initial state to be updated by the ConvGRU.

$$h_i^k = ReLU(m_i) \tag{8}$$

$$h_i^k = \begin{cases} U_{ConvGRU}(h_{i,t-1}^K, m_i) & k = 0 \ \& \text{ temp} \\ U_{ConvGRU}(h_{i,t}^{k-1}, m_i) & otherwise \end{cases} \tag{9}$$

3.4. Read out

Once the K-message passing steps are finished, we have in each node the final state $h_i^K \in \mathbb{R}^{hxwxcx1}$ that consists on a rich representation in the feature space domain of each particular object. To obtain the final mask, we apply Readout R(·) function to each final state and obtain the predicted $\hat{S} = \{0, 1\} \in \mathbb{R}^{HxWx1}$ mask. The architecture consists on a 3x3 convolutional layer, a non linearity and a final 1x1 convolution. The output tensor is then up-sampled to the input dimensions WxH and we finally apply the sigmoid activation function to obtain the final mask output probabilities of that object in the image (eq. 10). We have also studied a variation that concatenates h_i^K with the output from the Feature Extractor F_t 3.1 to incorporate semantic information of overall image features in order to produce better masks (eq. 11).

$$\hat{S}_i = \sigma(Up(W_{class} * ReLU(W_{dec} * h_i^K))) \tag{10}$$

$$\hat{S}_i = \sigma(Up(W_{class} * ReLU(W_{dec} * [h_i^K, F_t]))) \tag{11}$$

where Up(·) denotes the upsampling operator.

4. Experiments

4.1. Dataset

We have used YouTube-VOS dataset [11] for all the experiments conducted. YouTube-VOS consists of 3,471 videos in the training set and 474 videos in the validation set, being the largest video object segmentation benchmark. We have split the training set in 2 subsets, train and validation, with 80% and 20% of the data correspondingly. The original validation split is left as test-set.

4.2. Model configurations

We present in Table 1 the performance obtained using different combinations of the above presented variants of each module. They are all tested using the validation split from YouTube-VOS and the official evaluator server. We can extract several conclusion of what works best and which components are not contributing much, presented on the next section Next Steps 4.4. In the table below we show comparison results with other methods.

4.3. Training details

The original RGB frames and annotations have been resized to 256x448 as it is the resolution in which annotations

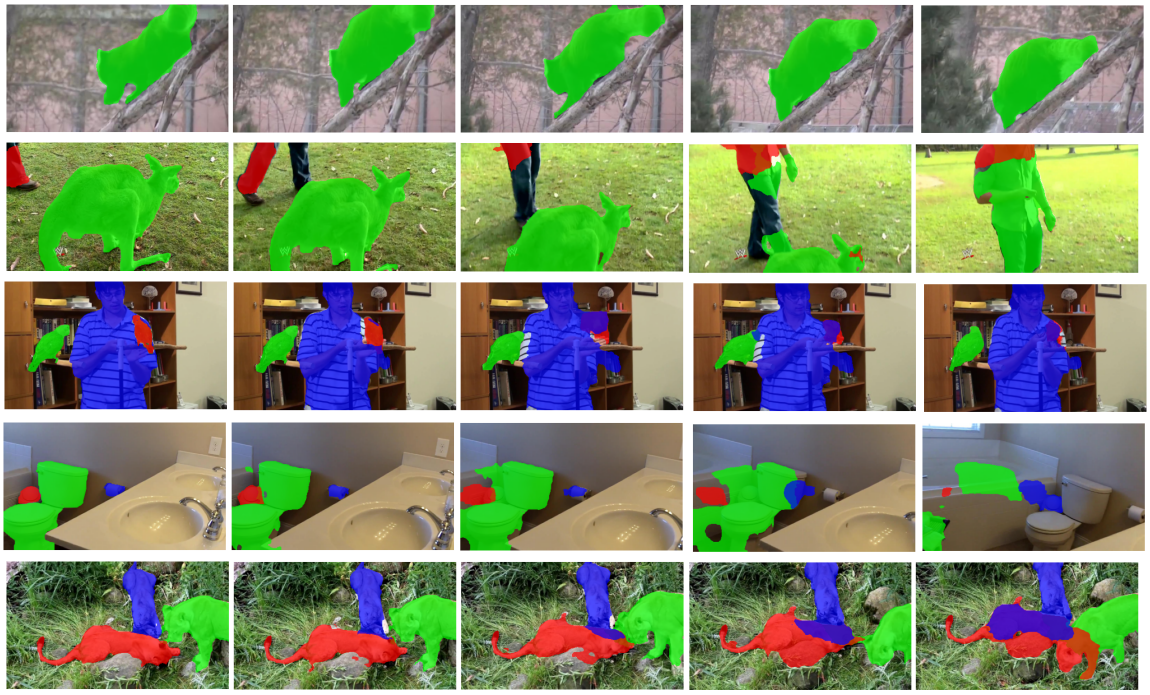

Figure 2: Qualitative results for one-shot video object segmentation on YouTube-VOS. First row shows an example with a single object, second row with two objects and the rest are for three objects.

Method	OL	Overall J&F
Ours	✗	44.0
RVOS[8]	✗	56.8
OSVOS[2]	✓	58.8
S2S[10]	✗	64.4
STM[6]	✗	79.4
EGMNN[5]	✗	80.2

Table 2: Performance comparison of SOTA architectures for VOS, all evaluated on YouTube-VOS validation split. [5] is currently the best model to our knowledge. Note that this comparison is not fair, as models are usually trained with extra training data from Davis or saliency datasets.

are given. Each training minibatch is composed with 4 clips of 5 consecutive frames, same as [8]. We have separated the training process in two different parts: First, we propagate the GT mask of the active objects in the sequence from the last frame to the Mask Encoder for 7 epochs. Then we use the predicted mask (instead of the GT) of each object which is the real scenario as we do not have GT mask when testing. The idea is to make training more stable, as we guarantee no noise propagation between frames when the model does worse. We use Adam optimizer to train our network and the initial learning rate is set $10^{-}3$. We train a total of 14 epochs including both mentioned phases. We use the trained ResNet-101 weights from [8] as initial weights for the Feature Extractor backbone and freeze them the whole training process. We use SoftIoU as our loss function.

4.4. Future Work and Conclusions

The actual performance of the model is not competitive, as almost any well-known VOS algorithm works better (see Table 2 for a comparison with popular architectures). The presented model though is not finished, and in this section we would like to introduce some ideas on how to improve parts of the model architecture.

4.4.1 Visual results

We show in Figure 2 some examples of how the models performs. The most noticeable problem is related with the motion capacity of the network, especially when we have more than one object and they share the same spatial position in the image at any time. This is, if we have an object that during the first 5 frames is still and then it moves an its spatial position is occupied by another object, we will just mix up masks from the object that enters the region and the one that was there and now is leaving. In other words, once a region is activated by one object, we will always predict that object in the region regardless it moves and that region is occupied by another one. On the other hand, we are able to handle quite successfully sequences with just one object de-

spite having motion. With more than one object, our model works when motion is really low and the objects are easy to differentiate.

4.4.2 Mask Propagation & Encoding

The idea to introduce a separated module to encode the mask concatenated with the obtained image features works well, as we can see an improve of 0.3 on the J&F (third vs fourth experiment in Table 1). The problem with it though is that its job is different depending on if we are working with the first frame of the sequence or not. In the first scenario, the mask of the object will be completely aligned with the features from the image, thus it will be relatively easy to encode the object pointed by the mask. If this is not the case, the mask will be the one from the previous frame, thus I will not be aligned with the features from the current frame. This makes the same architecture does two different jobs with 20% and 80% of the time correspondingly as we are working with clips of length five. In order to fix this, we would need a sort of optical flow estimation to properly align image features with each object mask from the previous frame (Liu. et. al. [4] uses FlowNet2.0 for this same purpose). We have thought in different approaches that could help fix this issue:

- Use optical flow as additional input to the Mask Encoder, that would be filled with zeros at $t == 0$ so it would be able to distinguish between different time steps.
- Use correlation between image features of consecutive frames: This would be the fully feature approach that would integrate better on our architecture. The idea is that by getting the correlation in the feature space domain we could have a sort of movement estimation in the feature domain that could be incorporated to the model. Note that this is typically used for attention mechanism for VOS.

4.4.3 Spatial Graph

Our presented graph architecture has a long way to go to be as powerful as it could be. In this preliminary work, we just present the simplest possible approach, connecting all the nodes with each other in the spatial domain, with no message passing in the time domain. Experiments show that this needs to be improved, as the current neighbor aggregation strategy (which is strongly related with how the graph is build) is currently making the performance of the model worse (fourth vs fifth row in Table 1). Visual results show also that it looks like objects gets mixed-up. In order to improve performance, we could add the following changes in the graph:

- Restrict nodes connectivity: A typical presented approach to connect nodes in the spatial domain is to define a distance measure between regions in the image and establish a particular threshold. Then nodes are connected in the graph if the distance between the regions of the image they describe are below the threshold. This allows to only connect nodes that refer to regions that are close in the image. This could help the graph learning to differentiate two objects that are really close to each other without receiving information from other nodes that are at the other end of the image.
- Edge weighting strategy: Recent GNN work also typically use edge weighting strategy in order to assign to each neighbour node an importance measure when being aggregated. This weight is usually obtained using any sort of attention mechanism, computing the similarity of each node with its neighbors. Note that this is complementary with the above point.
- Add temporal message propagation: The true power of the graph may be obtained when adding the temporal message passing. This could bring the possibility to allow the graph learning for instance the intrinsic flow estimation of objects. Note that this also would makes us change the overall structure of the model, as right now we operate each frame individually which is not compatible with this (several masks from different frames are predicted on the same time).
- Add extra node representing background: An interesting idea to help improve the performance could be adding an extra node that represent the background. Once initialized in the first frame, it could help the model learn the representation of the background across the video in order to differentiate it from the objects. This would require to add an extra loss function to measure background loss, thus optimizing the overall segmentation power of the model.

References

[1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations, 2016.

[2] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation, 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2016.

[4] Daizong Liu, Shuangjie Xu, Xiao-Yang Liu, Zichuan Xu, Wei Wei, and Pan Zhou. Spatiotemporal graph neural network based mask reconstruction for video object segmentation, 2020.

[5] Xiankai Lu, Wenguan Wang, Martin Danelljan, Tianfei Zhou, Jianbing Shen, and Luc Van Gool. Video object segmentation with episodic graph memory networks, 2020.

[6] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks, 2019.

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

[8] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro i Nieto. Rvos: End-to-end recurrent network for video object segmentation, 2019.

[9] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J Crandall, and Ling Shao. Zero-shot video object segmentation via attentive graph neural networks, 2019.

[10] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation, 2018.

[11] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark, 2018.