Attention and Fusion of Deep Representations for Computer Vision

by

Brendan Duke

A Thesis
Presented to
The University of Guelph

In Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science
in
Engineering

Guelph, Ontario, Canada
© Brendan Duke, September, 2020

ABSTRACT

ATTENTION AND FUSION OF DEEP REPRESENTATIONS FOR COMPUTER VISION

Brendan Duke
University of Guelph, 2020

Advisor:
Graham W. Taylor

In this thesis by articles we make contributions related to attention and fusion at the intersection of the deep learning and computer vision fields.

In our first article, we investigate the design of neural network operators that fuse features extracted from different modalities to make predictions on a single task. We propose a generalized class of multimodal fusion operators for visual question answering (VQA). We identify generalizations of existing multimodal fusion operators, and show that specific non-trivial instantiations of our operator exhibit superior performance in terms of open-ended accuracy on the VQA task.

In our second article, we introduce Transformers to video object segmentation (VOS). We propose a scalable, end-to-end method for VOS called "Sparse Spatiotemporal Transformers" (SST) to address runtime, scalability and temporal dependency issues of prior work. We show that our method achieves competitive results on YouTube-VOS 2019 with increased scalability compared with the state of the art.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Graham Taylor, and the rest of the MLRG lab for their mentorship and providing an environment where I could learn how to do research. I came to the University of Guelph without a research background. Dr. Taylor and the MLRG members were patient and allowed me to gradually find my own process for doing research, while providing helpful advice and discussions that continue through today. I am thankful that I had the opportunity to work with this group of bright, kind, motivated people and I hope our collaborations will continue into the future.

I would also like to thank our academic collaborators from the Deep Vision project, who are fantastic role models for how to be good scientists. In particular I would like to thank Dr. Christian Wolf, who provided thoughtful insight and ideas about our work as well as the machine learning and computer vision fields in general.

Finally, I would like to thank Shannon Despond for her constant loving support throughout my studies.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF SYMBOLS

a	Vector (including scalars)
A	Matrix or scalar constant
$\mathcal{A}$	Tensor with more than two dimensions

Chapter 1

BACKGROUND

Our work concerns the investigation of complex computer vision problems using the method of extracting deep hierarchical representations known as deep learning. In our articles we consider the relation of vision and natural language, and apply deep learning to a challenging pixelwise tracking problem. Underpinning the methods in our articles are a set of deep learning architectures, each of which constitutes its own active topic of research. After briefly describing deep learning as a whole, we introduce the particular subcomponents underlying our models, in order to provide additional background context before presenting our articles.

1.1 Deep Learning

Deep learning is a subfield of machine learning that uses a hierarchy of neural network layers for representation learning (Lecun et al., 2015). Representation learning, then, means to extract from raw data meaningful information in the form of vectors or tensors of real-valued numbers, called "features". The defining quality of deep learning is its learning of a deep hierarchy of representations using backpropagation (Rumelhart et al., 1988), which is the term used by the deep learning community for reverse-mode automatic differentation (Griewank and Walther, 2008). Thanks to breakthroughs in computation, data and algorithms, in recent years deep learning methods have made progress in domains that had previously proven challenging for traditional machine learning approaches. Today, deep learning is a top-performing approach for solving complex machine-learning problems including image and speech recognition, language translation and natural language understanding topics such as question answering (Lecun et al., 2015).

We introduce feedforward neural networks (FNNs) as a simple yet concrete example of a deep learning model. FNNs can be expressed as a sequence of hidden layers

$$h_i = f_i({W_i}^\top h_{i-1} + b_i), \tag{1.1}$$

where h_i is a hidden unit, W_i is the weight matrix of layer i of the neural network, b_i is a bias vector and f_i is a non-linearity or link function, such as the sigmoid function $f(x) = 1/(1 + e^{-x})$ or ReLU $f(x) = \max(x, 0)$. We may consider h_0 as the input feature vector to the FNN, and h_L as the output prediction for an FNN with L layers. As with other deep models, the representation capacity of FNNs can be tuned by increasing the depth of the network, i.e., the total number of layers of the form in Equation 1.1, or by increasing the width of each of the layers i, i.e., the dimension of the hidden representation h_i.

In our articles we make use of more complex neural network architectures than FNNs, which may include FNNs as a subcomponent. In particular we focus on convolutional neural networks, recurrent neural networks, attention networks, and fusion operators, all of which we describe below.

1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural network designed to model data that is arranged in a grid, as the pixels in an image are arranged. While each neuron in a layer of an FNN is connected to all neurons in its previous layer via a weight matrix, neurons in a layer of a CNN are only locally connected to neurons in the previous layer.

We explain the meaning of "locally connected". We assume that a given layer has a 3D tensor $\mathcal{V}$ as input, where in addition to the single dimension of the hidden unit vector h of FNNs, there are two extra spatial dimensions (e.g. x and y dimensions of an image). A CNN connects the input tensor $\mathcal{V}$ to an output tensor $\mathcal{Z}$ through a 4D tensor $\mathcal{K}$ called a "kernel", which contains the CNN's learned parameters.

The kernel has one dimension matching the "channels" (the third, i.e. non-spatial, dimension of the input tensor $\mathcal{V}$), one dimension that becomes the channels of the output tensor $\mathcal{Z}$, and two spatial dimensions that are normally much smaller than the input feature tensor $\mathcal{Z}$'s spatial dimensions. E.g. for an input image of dimensions 224×224 pixels and 3 colour channels, a CNN kernel's spatial dimensions would normally be 3×3. The CNN kernel's weights are thus shared spatially, and each element of the output tensor is contributed to only by a 3×3 spatial region of the input tensor, via a sum over dot products on the vectorized slice of the kernel and the vectorized slice of the spatial region.

Formally, each element $\mathcal{Z}_{i,j,k}$ of the output tensor corresponds to the sum given in Equation 1.2, where i is the index of the output channel and j and k are the spatial indices of the output element.[1] The index l runs over the input channels, while m and n are restricted by the spatial dimensions of the kernel, e.g. in our 3×3 kernel example we have $m, n \in \{-1, 0, 1\}$.

[1] Note that while CNNs are called "convolutional", Equation 1.2 technically describes cross-correlation. True convolution in the sense of signal processing would be identical to cross-correlation, except for a 180° rotation of the spatial slice of the kernel weights (Lyons, 1996). This technicality is inconsequential in machine learning, since our weight initialization techniques generally do not vary under rotation, and learned weights would adapt to accomodate the rotation during training.

$$\mathcal{Z}_{i,j,k} = \sum_{l,m,n} \mathcal{V}_{l,j+m,k+n} \mathcal{K}_{i,l,m,n} \tag{1.2}$$

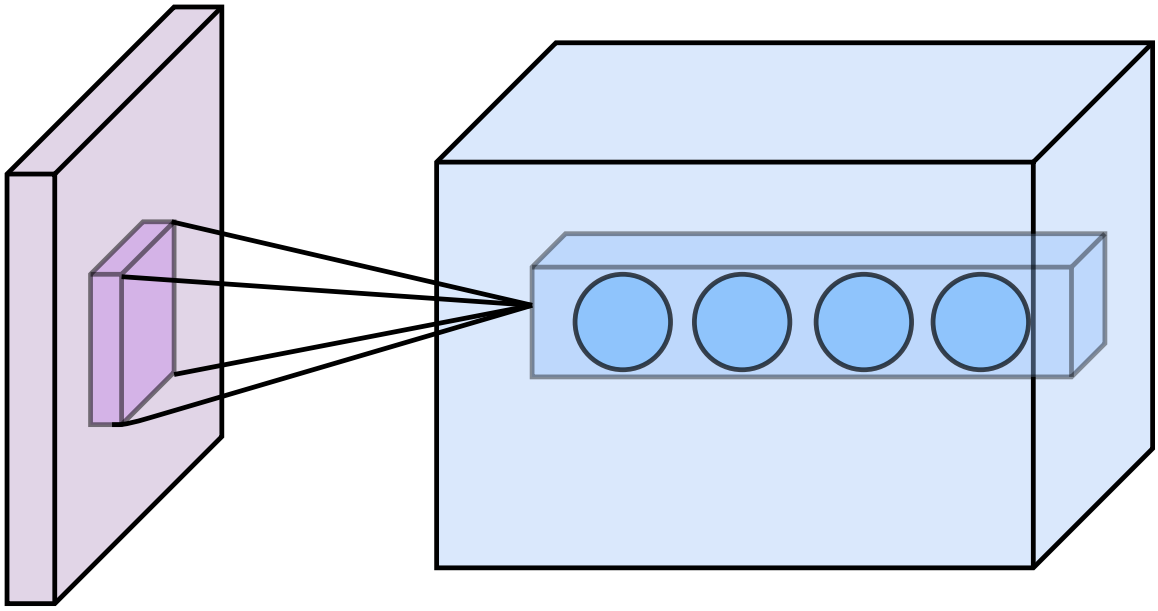

Figure 1.1
Two subsequent layers of a convolutional neural network.

Figure 1.1 depicts the interaction between two subsequent layers of a CNN. The kernel $\mathcal{K}$ (inner box on the left) is multiplied with a volume of the input $\mathcal{V}$ (outer box on the left) to produce a single unit consisting of a column of values at a single spatial location in the output $\mathcal{Z}$.

CNNs generalize well in comparison to FNNs on visual data, since the number of parameters are dramatically reduced compared with an equivalent FNN by sharing parameters in the spatial dimensions (LeCun, 1989). This is an example of improving generalization of neural networks by using prior knowledge (i.e. in this case, the spatial equivariance of image data) when constructing a model for a given type of data.

In the case of the visual question-answering task, we make use of a particular CNN architecture called a ResNet (He et al., 2016) that has 152 layers. This CNN architecture has been pre-trained on a large dataset of image data to extract useful generic feature representations from images. The fusion operator takes this image feature representation as input, along with a feature representation of the question, in order to predict an answer to the posed question conditioned on the image.

1.3 Recurrent Neural Networks

Like CNNs, Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986) are a type of neural network model that has been designed using prior knowledge of the type of data distribution relevant to the given task. In particular, RNNs are designed to work on sequential data of arbitrary length, such as time-series data including stock prices over time, or language data, where the inputs are the word or character tokens making up sentences and paragraphs.

Instead of sharing parameters spatially as in CNNs, RNNs share parameters over time. RNNs compute on variable-length sequence recursively, by computing a "hidden state" $h^{(t)}$ activation at each timestep of an input sequence using the input from the current timestep and the hidden state of the previous timestep. Formally,

$$h^{(t)} = \phi\big(h^{(t-1)}, x^{(t)}\big), \tag{1.3}$$

where ϕ represents the function defined by a particular recurrent unit.

In its basic form, the vanilla RNN, ϕ may be as simple as the tanh or sigmoid nonlinearity of an affine combination of $h^{(t-1)}$ and $x^{(t)}$. In this vanilla RNN case, the pre-activation $a^{(t)}$ at each timestep t is constructed from both the hidden state from the previous timestep $h^{(t-1)}$ as well as the input $x^{(t)}$ of the current timestep, as given by Equation 1.4,

$$a^{(t)} = Wh^{(t-1)} + Ux^{(t)} + b, \tag{1.4}$$

where W, U and b are the shared parameters of the RNN. The hidden state $h^{(t)}$ is related to the pre-activation $a^{(t)}$ of a given timestep t by the link function f, i.e. $h^{(t)} = f(a^{(t)})$, where f is usually tanh.

1.4 Skip-Thought Vectors

Skip-thought vectors (Kiros et al., 2015) is a method of encoding a feature representation from a sentence, in such a way that the feature representation can be re-used as a generic, continuous sentence representation in a variety of tasks. In skip-thought vectors, an "encoder" RNN is trained to extract a feature representation with enough context from the current sentence in order to predict words in the sentences immediately preceding and following the current sentence.

As their basic neural network component, skip-thought vectors use a type of RNN called a Gated Recurrent Unit (GRU) (Cho et al., 2014). Vanilla RNNs typically propagate information forward through time through an alternating sequence of affine transformations and saturating

nonlinearities. In backpropagation through the resulting vanilla RNN computation graph, gradients must sequentially traverse as many saturating nonlinearities as there are timesteps, which causes the vanishing gradient problem. Cho et al. designed GRUs to improve over the vanilla RNN activation function in order to alleviate vanishing gradients in RNNs (Cho et al., 2014) using a similar mechanism to skip-connections, which improve gradient flow in ResNets (He et al., 2016). Just as skip-connections compute each layer's activation as the weighted sum of its input and a function of that input, GRUs do the same for each timestep's activation. GRUs allow gradients to backpropagate without traversing saturating nonlinearities, by computing their hidden states $h^{(t)}$ as the linear interpolation between a candidate hidden state $\widetilde{h}^{(t)}$ and the previous hidden state $h^{(t-1)}$.

In order to compute the linear interpolation between previous and candidate hidden states, GRUs include an update gate $z^{(t)}$, defined as

$$z^{(t)} = \sigma\big(U_z h^{(t-1)} + W_z x^{(t)}\big), \tag{1.5}$$

where $\sigma(\cdot)$ is the sigmoid function. The hidden state, then, is computed as

$$h^{(t)} = \big(1 - z^{(t)}\big) \odot h^{(t-1)} + z^{(t)} \odot \widetilde{h}^{(t)}. \tag{1.6}$$

GRUs compute candidate hidden state $\widetilde{h}^{(t)}$ similarly to the vanilla RNN's hidden state given by Equation 1.4, except for the inclusion of a reset gate $r^{(t)}$. The reset gate varies between zero and one. When the reset gate is near one, the candidate hidden state acts as the vanilla RNN hidden state defined in Equation 1.4. When the reset gate is near zero, the previous hidden state is ignored and the candidate hidden state is computed solely from the current input. Formally,

$$\widetilde{h}^{(t)} = f\big(W h^{(t-1)} + r^{(t)} \odot \big(U x^{(t)}\big) + b\big) \tag{1.7}$$

where f is a nonlinearity, usually tanh. Another set of parameters U_r and W_r compute $r^{(t)}$ just as U_z and W_z compute $z^{(t)}$ in Equation 1.5, i.e., $r^{(t)} = \sigma\big(U_r h^{(t-1)} + W_r x^{(t)}\big)$.

Skip-thought vectors use an RNN, in our case a GRU, as an encoder, which encodes the sequence of words in a sentence into a feature representation corresponding to that sentence. In our first article, we use skip-thought vectors to extract a feature representation from the question, which is coupled with the feature representation extracted from the image as a dual input to a multi-modal fusion operator. The multi-modal fusion operator takes the two feature-representation inputs and produces an output prediction as to the answer to the given question, conditioned on the image. For the purposes of our first article, our multi-modal fusion operator predicts a categorical variable, which represents a multiple-choice style answer.

1.5 Attention

Attention in deep learning is an operator motivated by the idea that making predictions for a given task may require weighing a subset of inputs more heavily than others, or even focusing on a single subset of inputs exclusive of all others. In vision, attention is motivated by the human fovea, which is responsible for our sharp central vision used for demanding vision tasks such as driving and reading. In natural language, we structure our sentences differently depending on which cues in our environment capture our focus (Myachykov and Posner, 2005).

The deep learning community drew inspiration from the importance of attention in human perception, with early work using Boltzmann machines to combine foveal glimpses (Larochelle and Hinton, 2010), and using foveated images for recognition and tracking (Denil et al., 2012). Graves used differentiable attention in the form of weights of a mixture of Gaussians over shifts for handwriting synthesis (Graves, 2013). Mnih et al. used non-differentiable attention trained with reinforcement learning to perform classification with reduced computation by observing only predicted regions of an image (Mnih et al., 2014). Bahdanau et al. introduced attention models for neural machine translation, by predicting a set of weights $(\alpha_1, \ldots, \alpha_t)$ over the sequence of hidden state vectors $(h_1, \ldots, h_t)$ over t timesteps output by an encoder-decoder architecture (Cho et al., 2014).

In the following we describe mathematically the attention operator relevant to this thesis. We first introduce attention on sequences, i.e., 1D attention, before describing our generalization to the spatiotemporal domain in our second article.

Scaled Dot-Product Attention

Suppose we are given two matrices Q and K in $\mathbb{R}^{t_1 \times c}$ and $\mathbb{R}^{t_2 \times c}$, respectively, defined as $Q \equiv (q_1, \ldots, q_{t_1})$ and $K \equiv (k_1, \ldots, k_{t_2})$. Q and K represent input sequences of c-vectors, for example sequences of word embeddings. We wish to compute an output sequence $Y \equiv (y_1, \ldots, y_{t_1})$ that is a linear combination of a third sequence $V \equiv (v_1, \ldots, v_{t_2})$ whose weights in the linear combination depend on the affinity between pairs of vectors in Q and K.

In their work introducing Transformers for NLP tasks, Vaswani et al. (2017) interpreted attention in the framework of data retrieval, and we follow their perspective here. An attention layer retrieves vector elements of the value V based on the similarity of a query vector element from Q with the key vector from K that corresponds to the value. From our perspective, attention is a method for using query Q to perform a lookup in the values V, indexed by the key K.

In particular, output vectors $y_t \in \mathbb{R}^c$ are a linear combination of all value vectors v_i, with

the weighting of v_i in the linear combination dependent on the similarity of query vector q_t with key vector k_i. At this point in our definition of attention we are faced with a decision about how we should quantify the similarity between query vector q_t and key vector k_i. Vaswani et al. refer to the similarity between query and key as attention's "compatibility function". One possible compatibility function would be the "additive alignment model" used by Bahdanau et al. (2015), which computes compatibility by concatenating query and key and using this concatenated vector as input to a feedforward neural network. Another possibility is to use a dot-product compatibility function, which computes the compatibility between query and key as their dot product.

The dot-product compatibility function has similar theoretical complexity to additive compatibility, but has the advantage of being able to compute all pairwise query-key compatibilities in a single matrix multiplication $QK^\top$. Matrix multiplication is highly optimized on modern compute hardware, and therefore dot-product attention runs faster than additive attention out of the box. For this reason, we use dot product attention in our work.

Our overall attention function is therefore

$$\texttt{Attention}\big(Q, K, V\big) = \texttt{softmax}\left(\frac{QK^\top}{\sqrt{c}}\right)V. \tag{1.8}$$

In Equation 1.8 we scale by the square root of the dimension of query and key vectors $\sqrt{c}$ in the argument to the softmax. The intuition behind this scaled dot product attention, as given by Vaswani et al., is that if scalar elements of query and key vector were drawn from independent zero-mean, unit-variance distributions, then the magnitude of the query-key dot product grows like the square root of their dimension. Therefore for query-key combinations whose dimensions are large, without scaling it is likely that the softmax would saturate early in training, causing vanishing gradients.

Multi-Head Attention

Building on scaled dot-product attention, multi-head attention is another technique to improve the capacity while reducing the computation of attention. Multi-head attention breaks the attention computation up into multiple attention computations determined by the "number of attention heads" hyperparameter n_h. We illustrate multi-head attention in Figure 1.2.

A single attention operator as given by Equation 1.8 would have a theoretical complexity proportional to the query and key dimension c. In multi-head attention, we project query, key and value vectors by learned weights $W_i^Q \in \mathbb{R}^{c\times c/n_h}$, $W_i^K \in \mathbb{R}^{c\times c/n_h}$, and $W_i^V \in \mathbb{R}^{c\times c/n_h}$, respectively, where i ranges from 1 to n_h. We then perform the n_h attention operations on the projected query, key and value vectors. Hence, multi-head attention on the projected vectors has

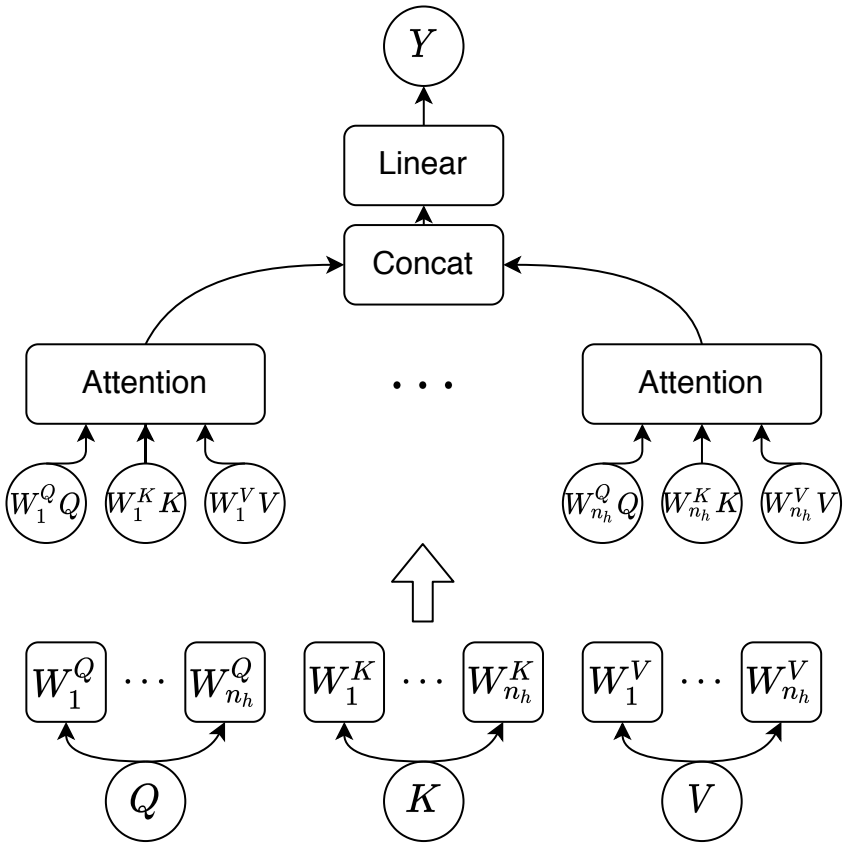

Figure 1.2
Multi-head attention. Adapted from Vaswani et al. (2017).

the same theoretical complexity as a single attention operator on the original query, key and value vectors. Intuitively, multi-head attention has the advantage that each ith head can specialize in learning to attend to particular features corresponding to the affine subspace spanned by columns of the ith projection weight matrix.

We concatenate outputs of the multi-head attention layers and feed the resulting c-vector into an output linear layer to produce the output Y. This final concatenation and projection step aggregates information from the multiple attention heads, for example, to combine information gathered by different attention heads "looking at" different regions of an image.

Positional Encoding

Positional information such as the absolute or relative position of tokens in a sequence, or the absolute or relative spatiotemporal position of pixels in a video tensor, provide an important cue for

making predictions on input data that contain this positional structure. However, up until this point in our description of attention, our attention operators are unable to incorporate positional information, since query and key vectors interact pairwise with each other. While recurrent networks have a positional inductive bias, both convolutional and attention networks are translation equivariant and require injection of position information to perform position-dependent tasks such as sequence-to-sequence learning. Following Gehring et al.'s use of positional encodings for convolutional sequence-to-sequence networks (Gehring et al., 2017), in Transformers Vaswani et al. (2017) introduced positional encodings to attention.

Positional encoding means adding to each key vector another vector with encoded positional information. Therefore with positional encoding Equation 1.8 becomes

$$\texttt{Attention}_E\big(Q, K, V\big) = \texttt{softmax}\left(\frac{Q(K+E)^\top}{\sqrt{c}}\right)V, \tag{1.9}$$

where positional encoding matrix $E \in \mathbb{R}^{t_2 \times c}$ contains a unique positional encoding c-vector for each of the t_2 elements in our queried sequence.

We consider positional encodings constructed of learned embeddings, and sinusoidal positional encodings. In the case of learned embeddings, the positional encoding matrix E is initialized at the beginning of training and updated along with the rest of the model parameters using backpropagation. Sinusoidal positional encodings, on the other hand, are not learned and instead are fixed to a vector of values from a set of sinusoids of varying period evaluated at the timestep, or sequence element, t. In particular,

$$\begin{aligned} E_{t,2i} &= \sin\left(\frac{t}{\tau^{2i/c}}\right) \\ E_{t,2i+1} &= \cos\left(\frac{t}{\tau^{2i/c}}\right) \end{aligned} \tag{1.10}$$

where τ is a fixed constant, for example Vaswani et al. fixed τ to 10 000.

1.6 Fusion

We refer to a topic specific to multimodal applications as fusion. Following Lahat et al.'s definition (Lahat et al., 2015), multimodal applications involve observing a single phenomenon using data collected from multiple distinct sensors. We refer to the raw stream of data collected from each sensor as a modality. For example, we could treat a video as a multimodal application by making use of RGB video frames, extracted optical flow and audio as input. Autonomous driv-

ing also makes use of multimodal fusion to combine information collected from multiple sensors, including LiDAR, radar, and camera sensors (Feng et al., 2019).

Multimodal fusion refers to the technique by which information extracted from these multiple raw data streams is combined. Using fusion holds the promise of producing superior predictions by leveraging the complementary information contained in separate data streams. As an example proposed by Kazakos et al. (2019), consider an egocentric action recognition system that is given as input a video of a person at a kitchen sink. When the sink tap is occluded, the audio signal provides a discriminative cue for the actions "turn tap on" and "turn tap off". On the other hand, video input would be more useful for predicting "wiping hands", which has no discriminative auditory signal. Hence, in this case multiple sensors provide the multimodal action recognition system with complementary inputs leading to superior predictions.

Traditional multimodal fusion researcher categorize multimodal fusion techniques into early and late fusion (Ramachandram and Taylor, 2017). Early fusion involves combining modalities at the data level, which proves challenging particularly for data drawn from sensors that may have different sampling rates or resolutions. Late fusion became popular in the early 2000s with the rise of ensemble classifiers (Kuncheva, 2004). Compared with early fusion, late fusion proves easier to implement, while removing the need to retrain models pretrained on each modality, since late fusion combines modalities at the prediction level.

Deep learning introduced a third category of multimodal fusion, "intermediate fusion", made possible due to deep learning models' hierarchical representations (Ramachandram and Taylor, 2017). The hierarchical feature representation provided by the many layers of deep models offers a flexible method for performing fusion, since representations from different modalities can be fused at different levels of the representation hierarchy. In deep multimodal models, separate modality-specific encoders, which could be for example CNNs or RNNs, first extract representations from each modality independently. A "fusion operator" then combines the modality-specific representations into a shared representation.

Fusion operators themselves have varying designs. A naïve approach would be to concatenate the representations from different modalities and feed their concatenation into a feedforward neural network to make a prediction. However, this simple concatenation may lead to overfitting, and features extracted from different modalities may be distributed differently so as to make taking their affine combination an unfavourable fusion strategy. Neverova et al. (2016) highlighted this overfitting phenomenon and proposed a simple fix via careful weight initialization, along with a novel regularization strategy to prevent co-adaptation of modalities. In orthogonal work, Kim et al. (2017) addressed both overfitting and modality distribution differences by performing multimodal fusion by elementwise multiplication, or Hadamard product.

Hadamard product fusion reduces the model parameter count in comparison with concatenation and forces modalities to interact to form predictions.

In general, the subject of designing fusion operators that form effective priors for combining modalities for a given task is a topic of active research. Fusion operators are an important component, for example, in the visual question answering (VQA) (Fukui et al., 2016) and image hashtag prediction (Durand, 2020) domains. Standard fusion operators include elementwise sum and product, and concatenation. Examples of research into more complex fusion operators include MCB (Fukui et al., 2016), MUTAN (Ben-younes et al., 2017), GLU (Dauphin et al., 2017) and TIRG (Vo et al., 2019). Designing a search space over multimodal fusion operators for VQA is the topic of our first article.

Chapter 2

PROLOGUE TO FIRST ARTICLE

2.1 Article Details

Duke, B. and Taylor, G. W. (2018). Generalized hadamard-product fusion operators for visual question answering. In *2018 15th Conference on Computer and Robot Vision (CRV)*.

Personal Contributions

The idea for conducting architecture search in the design space of fusion operators came principally from Graham Taylor, precipitated by our prior work on fusing pose, optical flow, depth and RGB modalities for action recognition. I devised the particular search space design for visual question answer fusion operators, motivated by adding nonlinearity to MUTAN (Ben-younes et al., 2017), and conducted all experiments. I wrote the manuscript with the assistance of Graham Taylor.

2.2 Context

Visual question answering (VQA) is a topic intended to test the ability of machine learning models to comprehend a question about a presented image. Early work on VQA extracted features separately from question and image using pretrained feature extractors: pre-trained CNNs and RNNs were used to extract features from image and question, respectively. These early works used simple fusion operators such as elementwise sum and product, or concatenation, until Fukui et al. showed that more complex fusion operators can outperform simple fusion (Fukui et al., 2016). Fukui et al.'s work spawned further work exploring fusion operator design for VQA (Kim et al., 2017; Ben-younes et al., 2017), and all of this model design was done manually.

In parallel, Zoph et al. showed that neural architecture search (NAS), an automatic method for searching for neural network models, is capable of producing architectures even superior to highly tuned image classification and language models (Zoph and Le, 2017). By creating a design space of architectures to search over automatically, NAS is capable of discovering complex architectures that would be difficult to propose manually.

We were inspired by Zoph et al.'s work, and the manual design of increasingly complex fusion operators for VQA. Our core idea was that NAS is capable of finding superior fusion operators than can be designed manually. Particularly, since multimodal fusion operators operate in

high-dimensional feature spaces it is difficult to manually design effective priors for these models, and automatic model search methods are especially important.

2.3 Contributions

In this article we proposed the idea for conducting architecture search in the design space of fusion operators. Furthermore, we proposed a specific example of a fusion operator search space for the task of visual question answering. We showed that specific operators drawn from our search space are superior to a baseline model, MUTAN, and offered intuition into the advantages of our method.

2.4 Recent Developments

We were recognized with a "Best Computer Vision Paper" award at the 2018 Computer and Robot Vision Conference for our work. Perez-Rua et al. published a paper investigating multimodal fusion architecture search, the core motivation for our paper, at CVPR 2019 (Perez-Rua et al., 2019).

Chapter 3

GENERALIZED HADAMARD-PRODUCT FUSION OPERATORS FOR VISUAL QUESTION ANSWERING

3.1 Introduction

Multimodal applications offer a challenge to model selection in machine learning, as the interactions between different data modalities (e.g., between video and audio, or between images and questions) may require a complicated prior in order to accurately capture regularities necessary for downstream tasks.

The particular multimodal application that we consider in this work is that of visual question-answering (VQA), i.e., of producing a natural language response to the combined input consisting of an image and a natural language question pertaining specifically to that image. In the case of VQA, the complexity of the task exists in both extracting useful feature representations from the question and the image, as well as the "fusion" of these feature representations by combining them in order to predict the answer to the question. In this article, we focus on the problem of combining feature representations in the VQA task through models based on a generic "fusion operator" definition.

We illustrate the complexity of model selection for the VQA task by designing a class of multimodal fusion operators, each of which combines raw question and visual data streams to predict answers to the given questions based on an image. We evaluate specific instances of high performing fusion operators belonging to the same design class.

We evaluate and discuss three multimodal fusion architectural components that emerge as improving performance as part of the investigation into the general class of fusion operators:

1. The use of a gating mechanism, wherein individual features extracted by the fusion operator are turned on or off by a multiplicative interaction.

2. The introduction of distinct nonlinearities between parallel components in the fusion operator, which we hypothesize adds a performance boost due to an ensembling effect.

3. The additional introduction of learned nonlinearity in the form of a neural network inside the fusion operator, which takes features from the bilinear interaction of a pair of question and visual feature vectors as input.

3.2 Related Work

Model Selection

We propose that the task of model selection applied to multimodal problem domains can be improved by using automated architecture search techniques. Reinforcement learning has been used to conduct automated search for neural network architectures (Zoph and Le, 2017), gradient descent optimizers (Bello et al., 2017) and activation functions (Ramachandran et al., 2017). Other options for architecture search include evolutionary optimization (Liu et al., 2018), Bayesian optimization (Malkomes et al., 2016), and gradient descent-based methods.

We contribute to multimodal model selection by describing a generalized class of fusion operators that can be used as a design space for many of the search techniques described above. We enable future research to bypass the difficult problem of model selection for multimodal applications by using automated model design techniques that use the search space we describe as a basis.

Fusion Operators

We focus on multimodal fusion of two modalities, applied to the task of visual question-answering. We use modality to refer to a raw data stream of information, as would be presented to an observer from a sensor. In the case of the VQA application, the first modality is the sentence corresponding to the asked question. The second modality is the image about which the question is being asked.

Current research in multimodal fusion for VQA focuses on performance improvements by approximating a bilinear product between a feature vector q extracted from the question, and a feature vector v extracted from the image (Fukui et al., 2016; Kim et al., 2017; Ben-younes et al., 2017). The feature vectors q and v can be produced by pre-trained feature extraction methods specific to encoding information from the sentence and image modalities into vector representations.

In the experimental design of this work, as well as in the work of Kim et al. (2017) and Ben-younes et al. (2017), a pre-trained Residual Network model (He et al., 2016) is used as a feature extractor for the image data stream, and a pre-trained Skip-Thought Vectors (Kiros et al., 2015) model is used to extract features from the sentence data stream.

In general, we define a fusion operator for the VQA task as a function $\mathcal{F}_\theta$, parametrized by θ, of the question feature vector q and the visual feature vector v. The fusion operator $\mathcal{F}_\theta$ computes

a vector output, which is consumed downstream by a function g, e.g. a linear layer followed by a softmax layer, in order to model a probability distribution over the answer y conditional on q and v:

$$p(y \mid q\,, v\,; \theta) = g(\mathcal{F}_\theta(q\,, v))\,. \tag{3.1}$$

The methods of Fukui et al. (2016), Kim et al. (2017) and Ben-younes et al. (2017) all propose that in the multimodal application of VQA, the outer product of the feature vector q extracted from the question, with the feature vector v extracted from the image, produces a more expressive feature representation than straightforward concatenation, element-wise product, or element-wise sum. These previous works design fusion operators based on approximations to the outer product $q \otimes v$.

The Multimodal Compact Bilinear Pooling (MCB) method of Fukui et al. (2016) uses the Compact Bilinear Pooling method of Gao et al. (2016) to approximate a bilinear interaction between q and v. In the case of MCB, the fusion operator $\mathcal{F}_\theta$ consists of projecting q and v using Count Sketch (Charikar et al., 2002), followed by convolution of q and v, which is done efficiently using an element-wise multiplication in the frequency domain.

The MCB method makes use of the fact that the Count Sketch of the outer product $\Psi(q \otimes v\,, h\,, s)$, where h and s are uniform random variables as described in Fukui et al. (2016), is equal in expectation to the convolution $\Psi(q\,, h\,, s) * \Psi(v\,, h\,, s)$ of the question and visual feature representations, i.e., $E[\Psi(q \otimes v\,, h\,, s)] = E[\Psi(q\,, h\,, s) * \Psi(v\,, h\,, s)]$. However, since this expectation is intractable to compute, Kim et al. (2017) proposes a fusion operator based on the Hadamard (element-wise) product.

The fusion operator in Kim et al. (2017), the Multimodal Low-rank Bilinear Attention Networks (MLB), uses the Hadamard product coupled with projections of q and v with learned weight matrices W_q and W_v, in order to make a low-rank approximation to the general outer product $q \otimes v$. The projected vectors are multiplied by a third weight matrix W_z, such that the fusion operator in the case of MLB is $\mathcal{F}_\theta(q\,, v) = {W_z}^\top({W_q}^\top q \odot {W_v}^\top v)$, where $\odot$ denotes the Hadamard product. Combined, the weight matrices W_q, W_v and W_z approximate the general weight tensor $\mathcal{T}$ corresponding to the bilinear interaction between q and v in the special case where $\mathcal{T}$ is of low rank. The idea of using a constrained-rank weight tensor in the outer product $q \otimes v$ is the same idea used in the MUTAN fusion operator of Ben-younes et al. (2017), which generalizes MLB by allowing the bilinear interaction tensor between q and v to be of rank R.

MUTAN is a fusion operator that is motivated in Ben-younes et al. (2017) by the Tucker decomposition (Kolda and Bader, 2009), which we discuss in Section 3.2. In Section 3.3, we

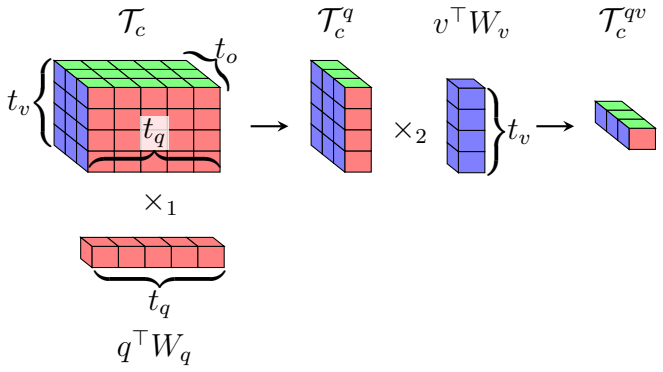

Figure 3.1
A geometric representation of the first two n-mode products of the right hand side Tucker decomposition of Equation 3.3, where $A = q^\top W_q$ and $B = v^\top W_v$, i.e. $\mathcal{T}_c \times_1 q^\top W_q \times_2 v^\top W_v$. From left to right, the vector $q^\top W_q$ is first combined with the core tensor $\mathcal{T}_c$ by taking the inner product along the dimension of size t_q between $q^\top W_q$ and each of the $t_v \times t_o$ fibers composing $\mathcal{T}_c$, producing the matrix $\mathcal{T}_c^q$. Then, $v^\top W_v$ is combined with $\mathcal{T}_c^q$ by taking the inner product between $v^\top W_v$ and the t_o columns of $\mathcal{T}_c^q$, producing the vector $\mathcal{T}_c^{qv}$.

derive the form of bilinear interaction used by the MUTAN fusion operator from the definition of the Tucker decomposition, since we base our fusion operator models on this derivation as well. The MUTAN fusion operator generalizes the MLB fusion operator to an interaction tensor of rank R, and therefore is,

$$\mathcal{F}_\theta(q\,,v) = \sum_{r=1}^{R} W_z^{r\top}(W_q^{r\top} q \odot W_v^{r\top} v)\,. \tag{3.2}$$

Tucker Decomposition

The Tucker decomposition is a form of higher-order principal component analysis (Kolda and Bader, 2009), which decomposes a tensor into a series of three n-mode tensor products between a single core tensor $\mathcal{T}_c$ and three matrices:

$$\mathcal{T} \approx \mathcal{T}_c \times_1 A \times_2 B \times_3 C \tag{3.3}$$

In Equation 3.3, the n-mode tensor product denoted by $\times_n$ is the multiplication of a tensor by a matrix, or by a vector, as in the case of MUTAN and our own method.

In general, the n-mode tensor product of a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $W \in \mathbb{R}^{J \times I_n}$ yields a new tensor $\mathcal{T} \times_n W$ with size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$.

In our special case, where we have an n-mode product between $\mathcal{T}$ and a vector z, the size of $\mathcal{T} \times_n z$ becomes $I_1 \times \cdots \times I_{n-1} \times 1 \times I_{n+1} \times \cdots \times I_N$. The mode-$n$ "fiber", referring to the generalization of rows and columns of matrices to higher-order tensors as defined in Kolda and Bader (2009), $\mathcal{T}_{i_1 \cdots i_{n-1} : i_{n+1} \cdots i_N}$ of $\mathcal{T}$ along I_n becomes a single element resulting from the dot product of $\mathcal{T}_{i_1 \cdots i_{n-1} : i_{n+1} \cdots i_N}$ with z.

Formally, the element-wise definition of the n-mode product between tensor $\mathcal{T}$ and vector z is:

$$\left(\mathcal{T} \times_n z\right)_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} \mathcal{T}_{i_1 \cdots i_N} z_{i_n} \,, \tag{3.4}$$

where in Equation 3.4, the dimensionality of $\mathcal{T} \times_n z$ is reduced to $N - 1$ by summing over the nth dimension. The n-mode product is depicted in Figure 3.1.

In Section 3.3, we use the definition of the Tucker decomposition from Equation 3.3, along with the definition of the n-mode product for the special case of multiplication between a tensor and a vector, in order to derive the MUTAN fusion operator of Equation 3.2, and to further generalize the derived fusion operator to include Feature Gating and Nonlinearity Ensembling.

3.3 Methods

In this section, we present the methods used to derive a generalization of bilinear fusion operators for the combination of two feature vectors in multimodal applications. We sub-divide our methods into the data and model sub-categories of the machine learning process. We first discuss the data used in our experiments, followed by discussion of the fusion operator model and how we generalized the model into a class of fusion operators over which we instantiated and evaluated a range of specific instances.

Data

The VQA 1.0 dataset (Antol et al., 2015) is a dataset of "free-form and open-ended Visual Question Answering (VQA)". The VQA task is to give an open-ended natural language reponse to an input consisting of a natural language question and an image. In VQA 1.0, there are 23 234 unique one-word answers for real images, and therefore the model's outputs can be represented as a multiple choice over these answers. In practice, we limit the number of choices to the top 2000 most common answers in the training dataset.

The VQA 1.0 real images dataset consists of 204K images from the MSCOCO dataset (Lin et al., 2014b), 614K questions, and 6M answers (ten answers per question). The dataset is separated into a $2/1/2$ training/validation/test split by the VQA 1.0 authors.

It is known that due to the data collection methodology of the VQA 1.0 dataset, there exists an imbalance in the dataset that allows questions to be answered with high accuracy without taking the image into account; this is remedied with VQA 2.0 (Goyal et al., 2017), which balances the answers to each question, so that the best scores on the VQA 2.0 dataset are not possible using language priors alone.

The balancing of VQA 2.0 is due to the addition of complementary images, wherein for a given image and question pair (I, Q) with answer A, a complementary image I' is found that is similar in appearance, but has a different answer A' to the question. The new example (I', Q, A') is then added to the dataset. VQA 2.0 includes 195K, 93K, and 191K complementary images in the training, validation, and test sets, respectively. In total, the VQA 2.0 dataset has 443K training, 214K validation, and 453K test (image, question) pairs.

The increased difficulty of the VQA 2.0 dataset with regards to emphasizing the importance of the combination of visual and question features creates a larger gap between relatively strong and weak fusion operators, i.e., the performance gap between strong and weak fusion operators increases when moving from VQA 1.0 to VQA 2.0, even when the absolute performance decreases for both models under consideration (Goyal et al., 2017). Therefore, we evaluate our proposed fusion operators on VQA 2.0.

Model

Following the work of Kim et al. (2017) and Ben-younes et al. (2017), we represent the sentence and visual modalities with extracted feature vectors. We use a pre-trained Skip-Thought Vectors model (Kiros et al., 2015) to extract a "question vector" feature representation q from a question. As well, we use a ResNet (He et al., 2016) pre-trained on ImageNet (Russakovsky et al., 2015) to extract a "visual vector" feature representation v from an image.

For the purpose of this article, the Skip-Thought Vectors model and ResNet can each be thought of as "black boxes" used to extract useful feature representations from sentences and images, respectively. By using this black-box abstraction, the fusion operators we develop can automatically benefit from improved feature extraction models without altering the algorithm presented here.

The Skip-Thought Vectors model extracts a d_q vector from the question, and the ResNet model extracts a $d_v \times S \times S$ tensor from the image, where S is the pre-pooling spatial dimension

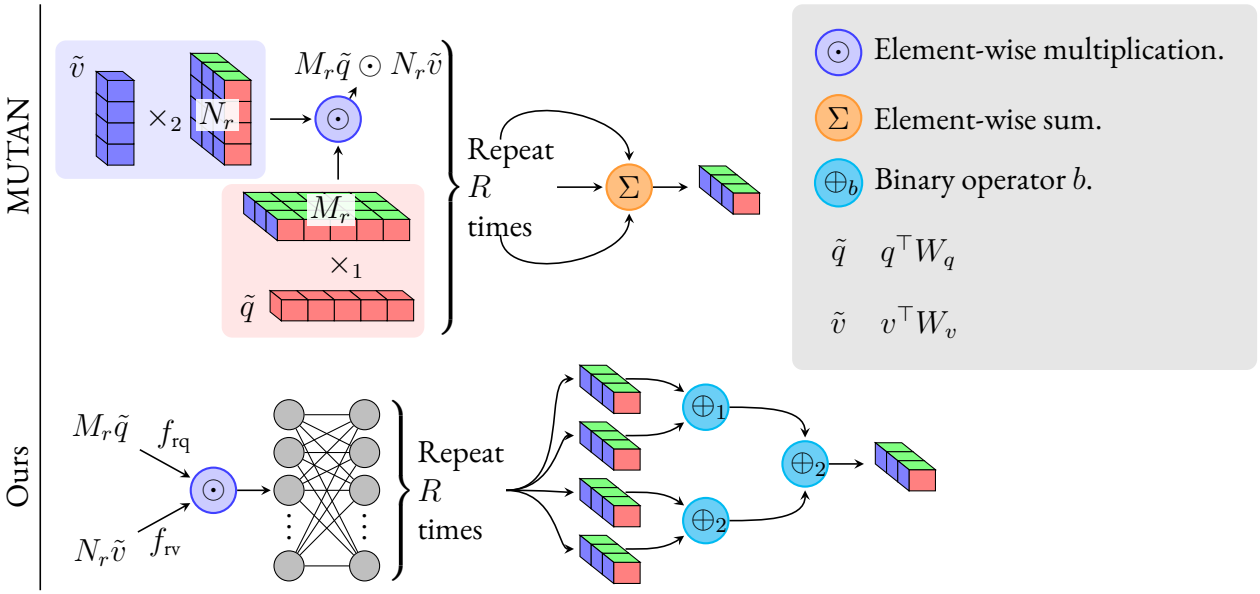

Figure 3.2
A comparison between the MUTAN fusion operator of Ben-younes et al. (2017) (top) and our generalized fusion operator (bottom). Both MUTAN and our fusion operator take the features $M_r\tilde{q}$ and $N_r\tilde{v}$ as input. MUTAN approximates the Tucker decomposition shown in Figure 3.1 by constraining the bilinear interaction between the vectors $\tilde{q}$ and $\tilde{v}$ to be of rank R. Note that while $\mathcal{T}_c$ in Figure 3.1 is a learned tensor whose parameters implicitly describe the bilinear interaction between $\tilde{q}$ and $\tilde{v}$, here the parameters of M_r and N_r are separate and the bilinear interaction is due to the Hadamard product. Our fusion operator generalizes MUTAN first by allowing different nonlinearities f_{rq} and f_{rv} to act on the features $M_r\tilde{q}$ and $N_r\tilde{v}$. The result is then composed with an additional learned nonlinearity in the form of a neural network, producing a set of R output features, as in MUTAN after the Hadamard product step $M_r\tilde{q} \odot N_r\tilde{v}$. In the case of MUTAN, the R output feature vectors are combined by element-wise summation, whereas in our fusion operator the R feature vectors are combined by applying a tree of binary operators $\oplus_b$, as described by Algorithm 1.

of the feature maps produced by the ResNet.

The MUTAN fusion operator of Ben-younes et al. (2017) and MLB fusion operator of Kim et al. (2017) combine the image and sentence feature vectors by approximating a bilinear interaction between the vectors. The bilinear interaction is learned, and the output of that interaction is input into a linear predictive layer, which is in turn fed into a softmax layer that outputs probabilities over the top 2000 most common answers in the training data.

We generalize the bilinear interaction of MUTAN and MLB such that both MLB and MUTAN, as well as element-wise multiplication and element-wise addition, can all be represented in

a common form.

We derive our fusion operator as a generalization of the MUTAN fusion operator by first deriving MUTAN from the definition of the Tucker decomposition (Kolda and Bader, 2009), then discussing the generalization of this expression. We begin from the definition of the Tucker decomposition given in Equation 3.3.

In the case of MUTAN, A in Equation 3.3 corresponds to the vector $q^\top W_q \in \mathbb{R}^{1\times t_q}$, B is $v^\top W_v \in \mathbb{R}^{1\times t_v}$, and C is $W_o \in \mathbb{R}^{|A|\times t_o}$, where $|A|$ is the dimensionality of the output vector (i.e., the number of answer classes), and t_q, t_v and t_o are the respective dimensionalities of the vector spaces that the question features, image features, and fused question-image features are projected onto before applying a linear prediction layer W_o.

Therefore, MUTAN is a special case of the Tucker decomposition in Equation 3.3, as given by Equation 3.5, where y is the prediction output of the fusion operator:

$$y = \mathcal{T}_c \times_1 q^\top W_q \times_2 v^\top W_v \times_3 W_o \tag{3.5}$$

In the following, we derive from the Equation 3.5 form of MUTAN an expression for MUTAN that can be generalized to a family of fusion operators, over which a variety of specific instantiations can be chosen and evaluated. Throughout the derivation, $\tilde{q}$ is shorthand for $q^\top W_q$, and $\tilde{v}$ represents $v^\top W_v$.

Referring to $\mathcal{T}_c \times_1 q^\top W_q$ as $\mathcal{T}_c^q$, each element of the matrix $\mathcal{T}_c^q$ in terms of elements of $\tilde{q}$ and $\mathcal{T}_c$ is $\mathcal{T}_c^q[j,k] = \sum_{i=1}^{t_q} \mathcal{T}_c[i,j,k] \cdot \tilde{q}[i]$, which follows from the definition of the n-mode tensor product.

Similarly, we expand the elements of the vector $\mathcal{T}_c^{qv} = \mathcal{T}_c^q \tilde{v}$ in terms of elements of $\tilde{q}$ and $\tilde{v}$, and slices of $\mathcal{T}_c$.

$$\begin{aligned}
\mathcal{T}_c^{qv}[k] &= \sum_{j=1}^{t_v} \mathcal{T}_c^q[j,k] \cdot \tilde{v}[j] \\
&= \sum_{j=1}^{t_v} \left(\sum_{i=1}^{t_q} \mathcal{T}_c[i,j,k] \cdot \tilde{q}[i] \right) \cdot \tilde{v}[j] \\
&= \tilde{q}^\top \mathcal{T}_c[:,:,k] \tilde{v}
\end{aligned} \tag{3.6}$$

The constraint enforced by Ben-younes et al. (2017) in MUTAN is that each slice of the core tensor, i.e., $\mathcal{T}_c[:,:,k]$, must be of rank R, and therefore a sum of matrices each given by an outer product of vectors. Therefore,

$$\mathcal{T}_c[:,:,k] = \sum_{r=1}^{R} m_r^k \otimes n_r^k = \sum_{r=1}^{R} m_r^k {n_r^k}^\top , \tag{3.7}$$

where $m_r^k \in \mathbb{R}^{t_q \times 1}$ and $n_r^k \in \mathbb{R}^{1 \times t_v}$. Substituting Equation 3.7 into the last line of Equation 3.6 gives:

$$\begin{aligned} \mathcal{T}_c^{qv}[k] &= \tilde{q}^\top \left(\sum_{r=1}^{R} m_r^k {n_r^k}^\top \right) \tilde{v} \\ &= \sum_{r=1}^{R} \left(\tilde{q}^\top m_r^k \right) \left({n_r^k}^\top \tilde{v} \right) , \end{aligned} \tag{3.8}$$

where each term in the sum is a scalar. Equation 3.8 states that $\mathcal{T}_c^{qv}$ is the sum over $r \in \{1, \ldots, R\}$ of matrix-vector element-wise products $M_r \tilde{q} \odot N_r \tilde{v}$, i.e.,

$$\mathcal{T}_c^{qv} = \sum_{r=1}^{R} M_r \tilde{q} \odot N_r \tilde{v} \, . \tag{3.9}$$

In Equation 3.9, $M_r \in \mathbb{R}^{t_o \times t_q}$ and $N_r \in \mathbb{R}^{t_o \times t_v}$ are learned matrices, whose rows are the vectors m_r^k and n_r^k, respectively.

We have presented an equivalent form of the MUTAN fusion operator in Equation 3.9, of which the MLB fusion operator of Kim et al. (2017) is a special case where $R = 1$. In this article, we propose a further generalized extension to Equation 3.9, by allowing the fusion operator to contain the following transformations:

- A unique pair of unary activation functions (f_{rq}, f_{rv}) that can wrap the individual factors before the Hadamard product. We refer to the combination of unique pairs of nonlinearities as Nonlinearity Ensembling.
- A sequence of L neural network layers ϕ_l composing a feedforward neural network module Φ that can take the features produced by the bilinear interaction $M_r \tilde{q} \odot N_r \tilde{v}$ as input, thereby introducing additional nonlinearity into the fusion operator.
- Skip connections (He et al., 2016; Srivastava et al., 2015) may exist between the inputs to the fusion operator, the outputs of any given neural network layer ϕ_j in the fusion operator, and the inputs to any other neural network layer ϕ_k, or the final output of the fusion operator.

- The sum over R terms in Equation 3.9 is generalized to arbitrary binary relations $\oplus_b(\cdot\,,\cdot)$. Each binary relation $\oplus_b$ is associated with a set $\{\mathcal{T}_r^{qv}\}_b$ corresponding to an element of a partition of the outputs $\mathcal{T}_r^{qv}$ from the R branches of the fusion operator, before those branches are joined. $\mathcal{T}_r^{qv}$ is defined according to Equation 3.10:

$$\mathcal{T}_r^{qv} = \Phi_r(f_{rq}(M_r\tilde{q}) \odot f_{rv}(N_r\tilde{v}))\,. \tag{3.10}$$

  There is also an ordering over the binary operations such that $\oplus_b$ form a sequence $(\oplus_b)_{b=1}^{B}$, where B represents the total number of distinct binary operators and is equal to the number of subsets in the partition $\{\mathcal{T}_r^{qv}\}_b$.

  The sequence $(\oplus_b)_{b=1}^{B}$ is applied recursively to the partitions $\mathbb{B}_b \coloneqq \{\mathcal{T}_r^{qv}\}_b$ according to Algorithm 1.

  The generalization over binary operators allows for binary operations besides addition to be applied to the $\mathcal{T}_r^{qv}$ output from each branch of the fusion operator, and also allows definition of precedence rules so that binary operators can be applied in a defined order. The fusion operator in Equation 3.9 is the special case of the generalized fusion operator where $B = 1$, $\oplus_1 = +$, and $\mathbb{B}_1$ is the entire set of branch outputs $\{\mathcal{T}_r^{qv}\}$.

Algorithm 1 Recursively applies the binary operator sequence $(\oplus_b)_{b=1}^{B}$ to the sequence $(\mathbb{B}_b)_{b=1}^{B}$ of elements of a partition of the set of outputs from each of the R branches of the generalized fusion operator. The IDENTITY$(\oplus)$ function returns the identity for the binary operator $\oplus$.

function APPLYBINOPSEQUENCE($(\mathbb{B}_b)_{b=1}^{B}$, $(\oplus_b)_{b=1}^{B}$)
  $v \leftarrow$ IDENTITY$(\oplus_1)$
  for all $b \in \{1, \ldots, B\}$ **do**
    $v_b \leftarrow$ IDENTITY$(\oplus_b)$
    for all $\mathcal{T}_r^{qv} \in \mathbb{B}_b$ **do**
      $v_b \leftarrow v_b \oplus_b \mathcal{T}_r^{qv}$
    end for
    $v \leftarrow v \oplus_b v_b$
  end for
  return v
end function

Our generalized fusion operator is compared alongside MUTAN in Figure 3.2.

3.4 Experiments

In this section, we instantiate specific models based on the generalized fusion operator described by Algorithm 1. We individually test the specific extensions for a generalized fusion operator, as described in Section 3.3. We then demonstrate the degree to which the extensions' performance gains are complementary.

In all of the following experiments, the Adam optimizer (Kingma and Ba, 2015) is used to train each model with a learning rate of 10^{-4}. The models are trained for 100 epochs. For the validation set, models with the best validation accuracy are selected. For the test-dev set, the model parameters used to generate test-dev predictions correspond to the early-stopping epoch determined from the validation set.

Following the hyperparameter settings of Ben-younes et al. (2017), the number of branches used in the fusion operator is $R = 5$. The dimensionality of the question vector q is the default for uni-skip Skip-Thought Vectors $d_q = 2400$, while the visual vector v is the $d_v = 2048$ dimensional output of a ResNet. Question and visual features are projected into a $t_q = t_v = 310$ dimensional vector space before being reprojected into a $t_o = 510$ dimensional vector space where q and v are combined by Hadamard product. The batch size used in our experiments is 128.

Nonlinearity Ensembling (NE)

To test the contribution of using a variety of activation functions per branch of the fusion operator, we first conducted a grid search over possible combinations where each f_{rq} and f_{rv} was drawn uniformly from the following set of candidate nonlinearities: identity function, leaky ReLU (Maas et al., 2013), SeLU (Klambauer et al., 2017), sigmoid, and tanh. We subsampled the VQA 1.0 dataset and ran a grid search with each run lasting only 5 epochs in order to quickly observe many combinations of nonlinear activation function pairs.

From the grid search, we observed the pattern that the stronger nonlinearity pair combinations used SeLU as the visual vector activation function, while using a variety of different activation functions on the question vector. Therefore, to test Nonlinearity Ensembling, we used branches where the visual vectors always used the SeLU nonlinearity, and the branches used one of each activation function on the question vector.

We propose that the diversity in the activation function on the question vector decreases the correlation between the branches, hence improving the ensembling effect of summing the branches' predictions together.

Post-fusion neural networks with skip connections

In order to allow for a nonlinear function to act on the bilinear features extracted by the Hadamard product between $M_r\tilde{q}$ and $N_r\tilde{v}$, we propose adding a neural network Φ_r to the fusion operator. We embed a "tiny" neural network in the fusion operator analogous to how Network in Network (Lin et al., 2014a) embeds a tiny neural network in a convolution.

In our implementation, Φ_r consists of a sequence of blocks of three feedforward layers where the activation function for each layer matches that of f_{rq}. There is a skip connection from the input, and from every third layer, to the output.

Feature Gating (FG) and Polarity Swap (PS)

Algorithm 1 introduces a method of generalizing the sum operation over fusion operator branch outputs $\mathcal{T}_r^{qv}$ to an ordered set of binary operations. We test two particular instantiations of Algorithm 1 where $B = 2$, $\oplus_1 = +$, $\oplus_2 = \odot$ and $|\mathbb{B}_2| = 1$, i.e. $R - 1$ of the branch outputs $\mathcal{T}_r^{qv}$ are summed and then element-wise multiplied by the remaining branch output.

Assume that the branch output that gets multiplied with the sum of the other branch outputs is $\mathcal{T}_R^{qv}$. The output of $\mathcal{T}_R^{qv}$ is first squashed by a nonlinearity f before being multiplied into the sum over $\mathbb{B}_1$.

In the case of our first experiment, the squashing nonlinearity is the logistic function f_{sigmoid}, which independently squashes the output $f_{\text{sigmoid}}(\mathcal{T}_R^{qv})$ into the range $(0, 1)$. The effect of multiplying by this squashed output is therefore to turn each feature on or off, so we refer to the first experiment as Feature Gating.

The squashing nonlinearity used in the second experiment is tanh f_{tanh}, which has the effect of squashing $f_{\text{tanh}}(\mathcal{T}_R^{qv})$ to the range $(-1, 1)$. We refer to the second experiment as Polarity Swap, since negative features can be conditionally swapped to positive and vice-versa.

Figure 3.3 demonstrates an example instantiated fusion operator that makes use of the Feature Gating idea, and implements the Feature Gating experiment described above. In Figure 3.3, the number of branches is set to $R = 3$ for clarity, while in the experiments of Table 3.1, the models have $R = 5$ branches.

3.5 Results

In Table 3.1, we compare performance improvements between different instantiations of the generalized fusion operator described in Equation 3.10 and Algorithm 1, which are discussed in Section 5.4. We use the VQA 1.0 validation set to compare the effect on the performance of fusion

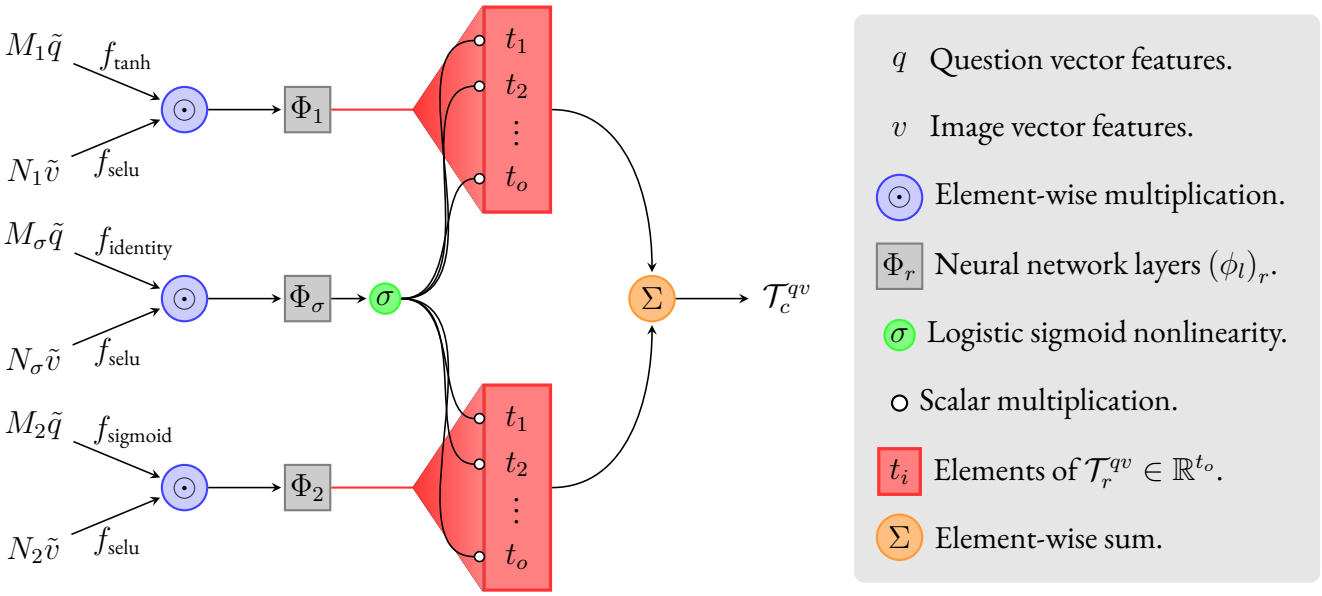

Figure 3.3
An example network using the Feature Gating neural network component, where each node represents a computation and the arrows represent the forward flow of information. The question and image feature vectors q and v are shared inputs to all branches. The Φ_r nodes represent post-fusion feedforward neural networks with skip connections. The logistic sigmoid node σ squashes output features $\mathcal{T}_\sigma^{qv}$ from Φ_σ to a vector of values in $(0, 1)$. The output from σ is element-wise multiplied with all other $\mathcal{T}_r^{qv}$ features from each branch, effectively turning on or off each feature channel. The resultant gated $\mathcal{T}_r^{qv}$ features are summed to become $\mathcal{T}_c^{qv}$, features that are input into a predictive layer to score the most common answers to questions from the VQA task.

operators of adding Nonlinearity Ensembling, as well as Feature Gating and Polarity Swap, independently. We chose to first investigate these static components of the fusion operator design before fixing them while investigating possible post-fusion neural networks. Since the neural network is a learned component of the architecture, it has to adapt to the static components during training, and hence the optimal hyperparameters of the neural network may vary depending on the choice of static components in the model.

The Nonlinearity Ensembling component improves the fusion operator's performance on the VQA 1.0 validation set, and this improvement stacks with the performance improvement achieved from Feature Gating. Furthermore, the performance of Feature Gating is better compared to Polarity Swap, when each is combined with Nonlinearity Ensembling.

We find that the improvements from the Feature Gating and Polarity Swap model elements

Table 3.1
An ablation study on Nonlinearity Ensembling, Feature Gating, and Polarity Swap.

Model	VQA 1.0 val
MUTAN (Ben-younes et al., 2017)	61.54
Nonlinearity Ensembling (NE)	61.66
Feature Gating (FG)	61.72
NE + Polarity Swap (PS)	61.77
NE + FG	**61.86**

do not stack. We combine the ideas by using one branch each to multiply by a vector of tanh and sigmoid outputs. The combined model performs slightly worse than the Polarity Swap model by itself, which in turn performs worse than the Feature Gating model.

Therefore, the best performing combination amongst Nonlinearity Ensembling, Feature Gating and Polarity Swap occurs from using the Nonlinearity Ensembling and Feature Gating together.

Building on the best model based on the ablation study of Table 3.1, we use the more challenging VQA 2.0 validation set to evaluate different post-fusion neural network architectures. We find that a post-fusion neural network with six layers and 128 hidden units per layer outperforms the NE + FG model by a margin of 0.53 percentage points, improving the VQA 2.0 validation open-ended accuracy (as defined in Goyal et al. (2017)) from 60.57% to 61.1%. We find that with the low number of 128 hidden units, dropout is detrimental to the accuracy, and the best model does not use dropout.

In Table 3.2, we evaluate our best model on the VQA 2.0 test-dev and test-std sets, and compare to previous state of the art models upon which our work is based, as well as to the best models of Teney et al. (2017), the winners of the 2017 VQA challenge. We find that our best model achieves an absolute percentage point improvement of 1.1% over the strong baseline of Ben-younes et al. (2017). We note that the improvement in open-ended accuracy of our model on the test-dev set is significantly larger in magnitude when compared to the improvement of Ben-younes et al. (2017) over Fukui et al. (2016). We attribute our relatively large improvement in accuracy to the introduction of nonlinearities in the fusion operator. The nonlinearities both allow the fusion operator to ensemble nonlinear functions of the input features (via Nonlinearity Ensembling), and to model nonlinear relationships between the bilinear features extracted by the Hadamard product (via the post-fusion neural networks).

When comparing our model's performance to that of the models of Teney et al. (2017) in Table 3.2, we note that the best performing models of Teney et al. (2017) gain a significant

Table 3.2
A comparison with the state of the art of our best single model on the VQA 2.0 test-dev and test-std sets.

	VQA 2.0 test-dev				VQA 2.0 test-std			
Model	All	Y/N	Number	Other	All	Y/N	Number	Other
MCB[1]	61.96	78.41	38.81	53.23	62.27	78.82	38.28	53.36
MUTAN[2]	63.13	80.7	39.4	53.55	—	—	—	—
ResNet[3]	62.07	79.20	39.46	52.62	62.27	79.32	39.77	52.59
Bottom-up[4]	65.32	81.82	44.21	56.05	65.67	82.20	43.90	56.26
Ours (single)	64.22	81.19	40.95	55.05	64.64	81.62	41.19	55.22

performance boost from using superior image features for the VQA task. In particular, Teney et al. (2017) use bottom-up attention (Anderson et al., 2017), which makes use of a Faster R-CNN (Ren et al., 2015) pipeline, to obtain features from object proposal regions of an image. Bottom-up attention features improve the models of Teney et al. (2017) by $\approx$ 3% absolute percentage points on average, and since our contribution focuses solely on improving the fusion operator, our model should gain similar improvements from using bottom-up attention features. The multimodal fusion operator used in Teney et al. (2017) is a Hadamard product, as in MLB.

3.6 Conclusion and Future Work

We presented a generalization of the MUTAN operator aimed at fusing multi-modal representations. We demonstrated the expressibility of the operator, showing that it could ensemble a variety of different nonlinearities, implement neural networks post-ensembling, and generalize MUTAN's sum-based reduction to hierarchical fusion with arbitrary binary operators. The few configurations we tested demonstrated significant gains relative to MUTAN and MCB on the VQA 2.0 task. However, the most promise lies not in choosing a configuration by hand, but leveraging this generalization as a design space for architecture search by reinforcement learning or other methods. This is the focus of future work.

[1] Fukui et al. (2016) as reported in Goyal et al. (2017)

[2] Ben-younes et al. (2017) as trained and evaluated by us

[3] ResNet features 7×7 (single) (Teney et al., 2017)

[4] Bottom-up attention image features, adaptive K (single) (Teney et al., 2017)

Chapter 4

PROLOGUE TO SECOND ARTICLE

4.1 Article Details

Duke, B., Ahmed, A., Wolf, C., Aarabi, P., and Taylor, G. (2020). Scalable video object segmentation with sparse spatiotemporal transformers. unpublished.

Personal Contributions

Abdalla Ahmed and I worked together writing a PyTorch codebase for video object segmentation (VOS). I devised the idea of using Transformers for VOS due to the natural analogy between attention and correspondence, and the idea that VOS can be reduced to finding correspondences in feature space. I adapted our sparse attention operators to the spatiotemporal domain and implemented the operators in CUDA. I conceived of, implemented and ran all experiments. I wrote the manuscript, with the assistance of Graham Taylor.

4.2 Context

VOS challenges computer vision researchers with the task of tracking an object pixelwise through a video sequence. At the time of writing, most VOS methods are recurrent, and feed the predictions or feature representations extracted from the previous frame as input to the current frame. Meanwhile, Wang et al. showed that correspondence can be learned through the cycle-consistency of time and that those correspondences can be used for VOS (Wang et al., 2019). The similarity between attention with the normalized cross-correlation used as a subcomponent by Wang et al. to find correspondences motivated me to investigate using end-to-end attention architectures for VOS.

4.3 Contributions

Our work introduces Transformers to the VOS domain, and we propose a method of incorporating reference object identity knowledge into a spatiotemporal Transformer architecture. We provide CUDA implementations of our sparse spatiotemporal attention operators in order to enable attention computation on tensor dimensions that would be otherwise unfeasible to compute on using existing deep learning primitives.

Chapter 5

SCALABLE VIDEO OBJECT SEGMENTATION WITH SPARSE SPATIOTEMPORAL TRANSFORMERS

5.1 Introduction

Video object segmentation (VOS) involves simultaneous tracking and segmentation of one or more objects throughout a video clip. Our work addresses semi-supervised VOS, which conditions on one single-frame reference annotation for each object track in a sequence. VOS is a challenging task in which algorithms must overcome object appearance changes, occlusion and disocclusion, as well as distinguish similar objects in motion over time.

Research solving these VOS challenges to create a highly performant VOS system is important in any downstream tracking applications where pixelwise tracking information is useful. Tasks that can use VOS downstream include player tracking in sports analytics, person tracking in security footage, and car and road obstacle tracking in self-driving vehicle applications. VOS methods are also relevant in interactive annotation of video data, where annotator time can be used more efficiently by employing automatic video object segmentation in an iterative annotate-predict-refine loop. Furthermore, our work uses VOS as a proxy task to investigate scalable algorithms for extracting spatiotemporal representations from video in general, and these algorithms can be re-used for yet other video prediction tasks.

Previous methods that attempt to solve VOS can be divided into three major categories: online finetuning, mask refinement and temporal feature propagation.

Online finetuning, mask refinement and temporal feature propagation methods each have inherent drawbacks. Online finetuning methods (Caelles et al., 2017) cannot adapt to changes in object appearance throughout a sequence. Mask refinement and temporal feature propagation methods (Ventura et al., 2019) currently in the literature are recurrent. Due to their sequential nature, recurrent methods for VOS exhibit compounding error over time, and are not parallelizable across a single example.

We propose a novel method for semi-supervised VOS that overcomes the drawbacks of online finetuning and sequential methods by processing videos in a single feedforward pass of an efficient attention-based network in which, at each layer, each spatiotemporal feature vector simultaneously interacts with all other feature vectors in the video. No online finetuning is required for our method, although our method could still benefit from online finetuning at the cost of the aforementioned runtime penalty. Furthermore, since our model is feedforward, we avoid the compounding error issue inherent in recurrent methods. Finally, SST is fully paral-

lelizable across a single example and can therefore take advantage of the scalability of current and future compute architectures.

Our method is motivated by work on Transformer architectures in language translation, as well as by orthogonal work on correspondence matching in computer vision.

Transformer architectures (Vaswani et al., 2017) have shown that end-to-end attention networks are a dominant current approach to a multitude of text natural language processing (NLP) tasks (Devlin et al., 2019; Dai et al., 2019), as well as speech recognition tasks (Lüscher et al., 2019; Kim et al., 2019). Since VOS is a timeseries prediction task traditionally approached with recurrent methods (Ventura et al., 2019), and recurrent methods were succeeded by Transformers in NLP, we hypothesized that we could achieve similar success with Transformers in VOS as has been achieved in NLP.

Besides Transformers' success in NLP, we were also motivated to use attention for VOS by the relation of attention to cross-correlation used for computing optical flow (Dosovitskiy et al., 2015) and tracking (Li et al., 2018), in that both cross-correlation and attention use pairwise dot products of embeddings to find matching patches of feature maps. Methods such as FlowNet (Dosovitskiy et al., 2015) that use cross-correlation to compute optical flow are relevant to semi-supervised VOS because a perfect optical flow computation could produce a video object segmentation by warping the reference mask throughout a video (subject to error due to disocclusions). However, computing perfect optical flow is difficult, particularly for the challenging in-the-wild sequences in VOS datasets.

Our method is related to FlowNet in that we also use cross-correlation (attention) to compute a matching of pixels in feature space. Instead of warping an object's reference mask, we use attention to warp object-discriminative features. Since our method uses flow in feature space, we avoid the difficulties in explicitly computing optical flow in pixel space. Ultimately our method uses flow in feature space to build a spatiotemporal object feature representation used to predict the given object's segmentation mask over the entire video. Compared with warping masks with explicit flow, we also achieve efficiency gains by instead warping compressed object features.

Applying spatiotemporal attention operators to VOS raises two challenges: computational complexity and distinguishing foreground objects. Naïve spatiotemporal attention is square in the dimensionality of the video feature tensor, i.e., $O((THW)^2C)$. We resolve this computational complexity issue with sparse attention operators, of which we compare two promising candidates. Furthermore, we resolve the "distinguishing foreground objects" problem, i.e., how to inform the model of which object to segment, by introducing an Expand-Mask-Project operator.

Our work achieves an overall score of 68.1 on the official YouTube-VOS validation set, com-

paring favourably with prior work. Furthermore, our method is robust to subsampling of video sequences, so that we are able to predict future frame segmentations in constant time by sampling and processing a fixed number of intermediate frames.

Contributions

For the first time, we apply video attention on video object segmentation, by using sparse attention operator variants to apply Transformer models on high-resolution videos. We also contribute empirical evaluation of Transformer models applied to video, as Girdhar et al. (2019), a work on video action recognition, is the only prior work using Transformers on video that we are aware of.

We extend sparse attention variants to video so that they can be used for VOS. Specifically we extend to 3D, with two spatial axes and one temporal axis, Criss-Cross Attention (Huang et al., 2019) from 2D semantic segmentation, and Sparse Attention (Child et al., 2019) from 1D language translation. Our sparse video attention operators are not VOS specific, and could be applied to other dense video prediction tasks.

We will provide a public implementation available upon our article's acceptance.

5.2 Related Work

Our work is related to previous work in video object segmentation (VOS) and feature matching. We are motivated by ideas from the feature matching literature and generalize explicit correspondence by extracting representations of transitive correspondence between features by following connectivity patterns in 3D activation maps.

Video Object Segmentation

In the VOS literature, **online finetuning** approaches (Bao et al., 2018; Caelles et al., 2017; Maninis et al., 2018; Voigtlaender and Leibe, 2017; Hu et al., 2018a; Khoreva et al., 2017; Li and Loy, 2018) do one-shot, or semi-supervised, VOS by finetuning a semantic segmentation network on an initial frame. A number of methods: (Cheng et al., 2018; Caelles et al., 2017; Maninis et al., 2018; Voigtlaender and Leibe, 2017) performed VOS on independent frames using finetuning alone without explicitly modelling temporal coherence. Maninis et al. (2018) built on the original usage of this approach in Caelles et al. (2017) by adding instance-level semantic information, while Voigtlaender and Leibe (2017) added adaptive training during the sequence. **Offline** methods must use temporal information to produce future segmentations from past frames, as done

using optical flow in Jang and Kim (2017) and Tsai et al. (2016). Our method is in principle able to take advantage of online finetuning to improve performance, and also performs competitively using offline training alone.

Research into temporal coherence in VOS falls into two categories: those that refine masks, and those that propagate temporal features.

Mask refinement approaches (Khoreva et al., 2017; Oh et al., 2018; Khoreva et al., 2017; Hu et al., 2017; Jang and Kim, 2017) refine a previous mask using feedforward models. Khoreva et al. (2017) implemented mask refinement by a recurrent connection on the concatenated frame $t-1$ output masks and frame t RGB inputs where the recurrent connection is a VGG16. Oh et al. (2018) extended recurrent mask refinement by concatenating the feature map from the first frame. Yang et al. (2018) used a spatial prior on the target location, with channel-wise attention mechanism and meta-learning to adapt the network to the object given in the first frame. Bao et al. (2018) propagated masks by approximate inference in an MRF, with temporal dependencies based on optical flow, and spatial dependencies using a CNN. Optical flow has also been used to add motion information via jointly training optical flow and VOS (Cheng et al., 2017), while Dutt Jain et al. (2017) and Hu et al. (2018a) also added motion information via optical flow.

Temporal feature propagation approaches (Tokmakov et al., 2017; Xu et al., 2018a; Hu et al., 2017; Salvador et al., 2017) improve upon mask refinement methods by increasing the expressive power of the mask feature representation. At the time of writing, all temporal feature propagation methods have used RNNs to encode and propagate spatiotemporal representations through time. Jampani et al. (2017) also used spatiotemporal features to forward-propagate information.

Our approach falls under the temporal feature propagation category. We use sparse attention operators to propagate features temporally in a single feedforward operation.

Luiten et al. (2018) split the task of semi-supervised object segmentation into two tasks. First, Luiten et al. generated multiple mask proposals for each frame and then selected and merged the masks to generate accurate and temporally consistent masks for the video sequences. In contrast to Luiten et al. (2018), our approach is conceptually simple, based on a single network architecture composed of attention operators, and we perform inference in a single end-to-end forward pass.

Voigtlaender et al. proposed FEELVOS (Voigtlaender et al., 2019) as a simple and fast method for solving the VOS problem. Unlike most other VOS methods, Voigtlaender et al. trained FEELVOS end-to-end using a pixel-wise embedding, along with global and local matching mechanisms using the reference and previous frames to generate the masks for the video sequence. Our work shares similarities with Voigtlaender et al. (2019)'s in that both methods are end-to-

end trainable and conceptually simple. Our method has the added advantages of simultaneously extracting features from multiple frames using attention, and using positional encodings to learn spatiotemporal position-aware representations.

Johnander et al. (2019) used a Gaussian Mixture Model representation of foreground and background appearance to propagate appearance information forward through a video sequence. Appearance models are complementary to our method and we show that combining appearance models with SST performs better than either method individually.

5.3 Method

We first present a high level view of our method's architecture, before describing implementation details of our "grid" and "strided" sparse attention modules for VOS.

Architecture

We draw parallels with the canonical text-based Transformer architecture (Vaswani et al., 2017) in order to motivate our architecture. Like NLP Transformers, our architecture consists of an encoder and decoder. Unlike the encoder of an NLP Transformer, which takes as input embeddings extracted from a text sequence, our encoder's input consists of the frames of the video to segment, and the reference masks of objects to segment. Like NLP Transformer decoders, our decoder conditions on the output of our encoder. Unlike NLP Transformer decoders, which take output sequence embeddings as input, our decoder takes as input an object representation initialized to the encoder reference features repeated over the length of the video.

Our encoder architecture, depicted in Figure 5.1, takes a sequence of T RGB video frames as input, and produces two streams of output: a generic video representation stream, and an object-discriminative representation stream. We subsample the input video tensor $\mathcal{X} \in \mathbb{R}^{3\times T\times H\times W}$ to a fixed size using Temporal Segment Network (TSN) sampling (Wang et al., 2016) during training. At test time we replace subsampling with an identity function. A 2D ResNet (He et al., 2016) encodes input frames independently, before splitting them into separate streams representing the video and individual objects-to-segment.

Our encoder architecture's generic video representation stream (top right in Figure 5.1) processes the framewise 2D ResNet output features with a sequence of self-attention layers, adding temporal context to the video representation.

The object-discriminative feature stream (bottom right in Figure 5.1) must provide an initialization that informs the decoder of which objects to segment, achieved by our Expand-Mask-Project operator shown in Figure 5.2.

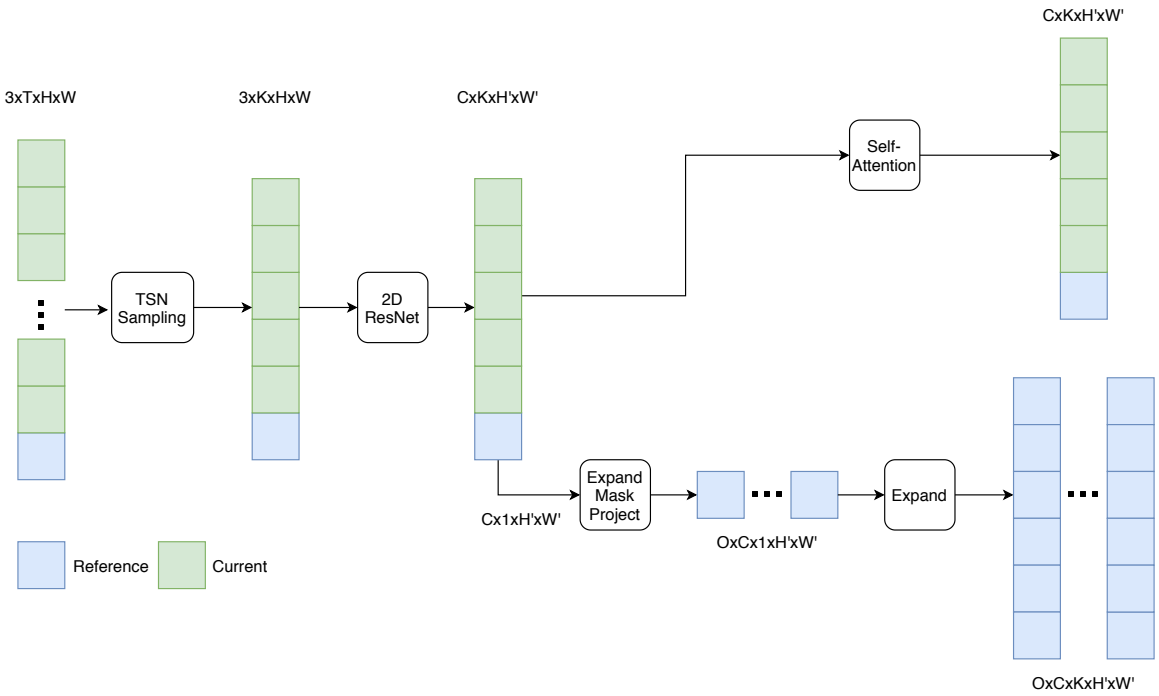

Figure 5.1
SST encoder architecture. The input to the SST encoder is a video $\in \mathbb{R}^{3\times T\times H\times W}$, where T is the total number of video frames. From left to right, the video is first subsampled (using TSN sampling (Wang et al., 2016) during training) to a fixed number of frames K, which are processed independently by a 2D ResNet (He et al., 2016) to produce a feature tensor $\in \mathbb{R}^{C\times K\times H'\times W'}$. The encoder then splits into two representation streams. The top stream encodes generic video features, and incorporates temporal context using a self-attention layer. The bottom stream encodes object-discriminative features by applying the Expand-Mask-Project operator to the reference features, and repeating the resultant feature tensor K times. Both streams are input to the SST decoder, described in Figure 5.4. Blue indicates reference features and green indicates non-reference video features (best viewed in colour).

The Expand-Mask-Project layer (Figure 5.2) relays to the decoder information about whether each pixel in the feature map belongs to an object by projecting object foreground and background to different affine subspaces described by the parameters of affine transformations $T_{foreground}$ and $T_{background}$.

To separately project object foreground and background features we use each object's downsampled mask and inverse mask (i.e., $1 - M$ for binary object mask M) to extract object foreground and background features. Foreground and background affine transformations $T_{foreground}$ and $T_{background}$ then project their respective extracted features to foreground and background affine subspaces. An elementwise addition operation then combines the projected object foreground and background features, producing a single $O \times C \times H' \times W'$ reference feature map, where O is the number of objects and C the number of channels, that contains information about which pixels of the feature map belong to which object.

Finally, the SST encoder's object-discriminative stream repeats the projected object foreground and background features by the number of sampled frames K times along a new axis to produce an $O \times C \times K \times H' \times W'$ output to feed into the SST decoder.

We briefly justify our choice to repeat the projected reference features across the video. The SST decoder gradually updates the object-discriminative features to produce the video object segmentation. When the SST encoder passes its object-discriminative features to the decoder, the encoder has not yet incorporated temporal context. Therefore, repeating the projected reference features across the video is the encoder's best initialization for object location throughout the video based only on the prior knowledge given in the reference frame.

The SST encoder produces two outputs passed to the SST decoder: a generic video feature representation $\mathcal{T} \in \mathbb{R}^{C \times K \times H' \times W'}$, and an initial object-discriminative video feature representation $\mathcal{R} \in \mathbb{R}^{O \times C \times K \times H' \times W'}$.

The SST decoder, shown in Figure 5.4, is a sequence of cross-attention layers that match object-discriminative and generic video features to warp the video object representation prior to the object's features throughout the video. The final layer of the SST decoder is a scoring convolution and sigmoid (at training time) or argmax (at test time) layer that produces the video object segmentation probability or segmentation masks $\mathcal{Y} \in \mathbb{R}^{K \times H \times W}$ from the final object-discriminative representation. In the case of multiple objects we have probability scoremaps in $\mathbb{R}^{O \times 2 \times K \times H \times W}$, i.e., foreground-background probabilities for each object for each pixel in the video, which we want to reduce to a tensor in $\mathbb{R}^{K \times H \times W}$ of object integer labels. We use the "naïve" inference protocol (Oh et al., 2018) and take, for each pixel, the argmax over all object probabilities including the background probability, computed as the maximum background probability over all objects.

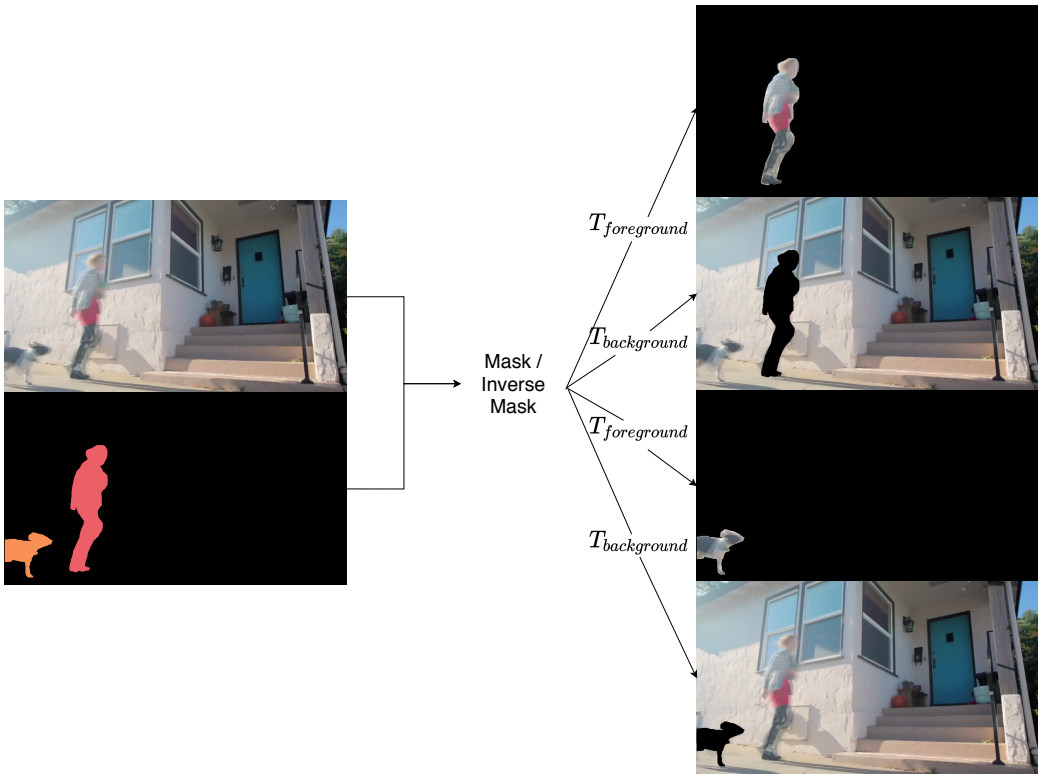

Figure 5.2
Expand-Mask-Project operator. We use an RGB image to visualize feature maps for clarity. The actual Expand-Mask-Project operator uses downsampled object masks and reference features processed by a 2D ResNet (He et al., 2016) of stride 8. The Expand-Mask-Project operator has three steps: **Expand**, **Mask**, and **Project**. In the **Expand** step, we repeat the $C \times H' \times W'$ reference features along a new object axis to get an $O \times C \times H' \times W'$ tensor, where O is the number of objects to segment. We then use each object o's foreground mask M_o and corresponding background mask $1 - M_o$ in the **Mask** step to create two copies of the original feature map for each object: one with only that object (and the background zero'ed out), and the other with only the background (i.e., everything besides the object, and the object zero'ed out). Finally, in the **Project** step, we use affine transformations $T_{foreground}$ and $T_{background}$ to separately project each object's foreground and background to the affine subspaces designating foreground and background features, respectively.

As we elaborate on in Section 5.3, each cross-attention layer takes query $\mathcal{Q}$, key $\mathcal{K}$, and value $\mathcal{V}$ tensors as input and returns a weighted sum of $\mathcal{V}$, where the weight of each value at an index in $\mathcal{V}$ increases with the dot product of the query and key at the same index. In the SST decoder, the role of query is played by the video representation, while the object-discriminative representation plays the role of both key and value, such that the function signature of each attention layer is $\texttt{Attention}\big(\mathcal{Q} = \mathcal{T}, \mathcal{K} = \mathcal{R}, \mathcal{V} = \mathcal{R}\big)$. Note that in object-discriminative features $\mathcal{R}$ foreground and background have been projected to separate affine subspaces, and we can use the projection weights in attention to perform a lookup in either of those subspaces to match against our video features.

From the perspective of attention as mapping a query and key-value pair, at each step we use feature vectors from our evolving object-discriminative representation to query the video key-value pair. We update our previous estimate of the video object-discriminative features with the resulting linear combination of video features from this lookup. Intuitively, we use our prior knowledge about reference features to perform a search in the video feature tensor to find features similar to our object. In subsequent steps we reuse the results of previous queries to perform a cascade of new searches for the object using our updated object feature estimate.

Sparse Attention

Attention is a dense operator that allows each element of a tensor to interact with all other elements at each attention layer. Attention is desirable in VOS because it can capture long-range dependencies without recurrence, and because attention can be viewed intuitively as a cross-correlation operator that uses CNN features for correspondence (Long et al., 2014), similar to prior work that used matching layers for optical flow (Dosovitskiy et al., 2015).

Formally, we follow Vaswani et al. (2017) in defining attention as

$$\texttt{Attention}\big(\mathcal{Q}, \mathcal{K}, \mathcal{V}\big) = \texttt{softmax}\big(\mathcal{Q}\mathcal{K}^\top\big)\mathcal{V}, \tag{5.1}$$

where query $\mathcal{Q}$, key $\mathcal{K}$, and value $\mathcal{V}$ are all matrices in $\mathbb{R}^{S\times C}$ for flattened spatial dimensions $S = THW$. As we alluded to in Section 5.3, we use object discriminative features $\mathcal{R}$ as query, and generic video features $\mathcal{T}$ as both key and value. Intuitively we use object-discriminative features initialized by the reference features to do a lookup in the video features in order to update our video object features estimate.

We adapted for VOS characteristic components of the Transformer architecture as described by Vaswani et al. (2017), including multi-head attention and positional encodings. We compare the performance of different positional encoding schemes applied to VOS in Section 5.4. We did

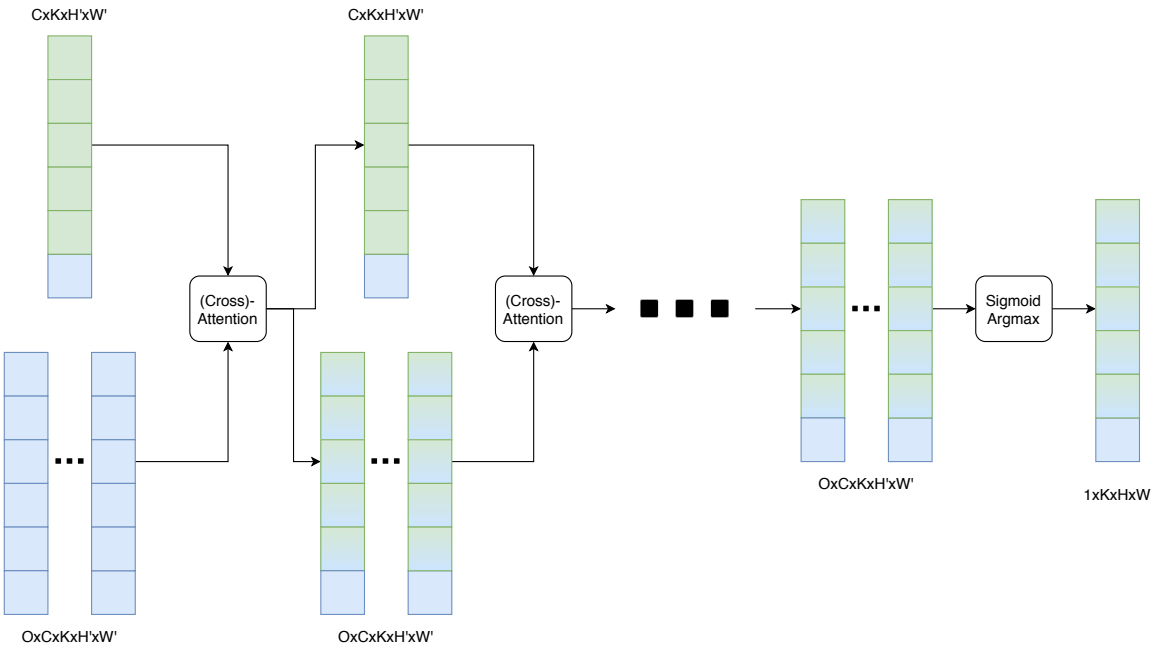

Figure 5.3
SST decoder architecture. Decoder inputs are a generic video representation tensor $\mathcal{T} \in \mathbb{R}^{C \times K \times H' \times W'}$ (top left, blue and green), along with object-discriminative reference features $\mathcal{R} \in \mathbb{R}^{O \times C \times K \times H' \times W'}$ (bottom left, blue), where O, C, K and $H' \times W'$ denote the number of objects, feature channels, subsampled video frames, and video spatial dimensions, respectively. The SST decoder is a series of cross-attention layers that warp reference features $\mathcal{R}$ to follow objects' movements in the video by matching with video features $\mathcal{T}$. We use colour gradients to show that object representations after the decoder input are a mixture of object-specific (blue) and generic video (green) features, therefore this figure is best viewed in colour.

not normalize the softmax argument in Equation 5.1 by the inverse square root of channels, as we found this scaling factor hurt performance. The difference in impact of scaling factor between our VOS attention and Vaswani et al.'s NLP attention could be due to our attention operator having a comparatively low number of channels.

A computational barrier prevents naïvely using Equation 5.1 to perform our desired lookup of video features $\mathcal{T}$ using object-discriminative features $\mathcal{R}$. The attention operation given in Equation 5.1 is $O((THW)^2C)$, which poses a problem for video object segmentation since for dense prediction tasks such as segmentation, model performance tends to improve with greater input resolution (Zhao et al., 2018). As an illustration of the infeasibility of using naïve attention for VOS, consider that a single layer of attention on a 16 frame video with a 64×64 feature map with 32 channels would cost more than 137 billion FLOPs, far more than the most computation-

ally expensive CNNs in the literature at the time of writing (Tan and Le, 2019).

We used two different sparse attention methods to make our attention operator computationally tractable at our desired framerate and resolution.

Grid Attention

We adapted our first sparse attention operator from Huang et al., who also noted the computational complexity issue when applying attention for semantic segmentation (Huang et al., 2019), although in VOS the issue is exasperated further by the addition of the time dimension. We refer to this operator as grid attention since we have generalized it from two to three dimensions, where the moniker criss-cross attention is no longer fitting.

At each layer of grid attention, each pixel of the video feature activation aggregates information from other pixels along its X, Y, and T axes independently. Each pixel interacts once with every other pixel in the video feature activation tensor that shares at least two of its X, Y, and T coordinates. Formally, for query $\mathcal{Q}$, key $\mathcal{K}$, and value $\mathcal{V}$ tensors all in $\mathbb{R}^{C\times T\times H\times W}$, we define our `GridAttention` operator as

$$\texttt{GridAttention}\big(\mathcal{Q},\mathcal{K},\mathcal{V}\big)=\Phi\big(\Omega\big(\mathcal{Q},\mathcal{K}\big),\mathcal{V}\big), \tag{5.2}$$

where Φ and Ω are the 3D generalizations of the **aggregation** and **affinity** operations defined by Huang et al. (2019).

Concretely, we define affinity operator Ω as

$$\Omega\big(\mathcal{Q},\mathcal{K}\big)_p=\texttt{softmax}\big(\mathcal{Q}_p\Psi\big(\mathcal{K},p\big)^\top\big) \tag{5.3}$$

where $\Psi:\mathbb{R}^{S\times C}\times\mathbb{R}^3\to\mathbb{R}^{(T+H+W-2)\times C}$ returns all pixels along the horizontal, vertical, or temporal axes incident to location $p\equiv(x,y,t)$. $\Omega(\mathcal{Q},\mathcal{K})$ is then a matrix in $\mathbb{R}^{S\times(T+H+W-2)}$ where each row contains, for a pixel in the video feature tensor, weights of a weighted average over all pixels along its temporal, vertical, and horizontal axes.

Aggregation operator Φ computes for each pixel p in the video feature tensor a weighted average over pixels along the temporal, vertical, and horizontal axes incident to p. Concretely, we define $\Phi:\mathbb{R}^{S\times(T+H+W-2)}\times\mathbb{R}^{S\times C}\to\mathbb{R}^{S\times C}$ as

$$\Phi\big(A,\mathcal{V}\big)_p={A_p}^\top\Psi\big(\mathcal{V},p\big). \tag{5.4}$$

Note that we implemented our grid attention operator in place, so there is no overhead from indexing tensors by p.

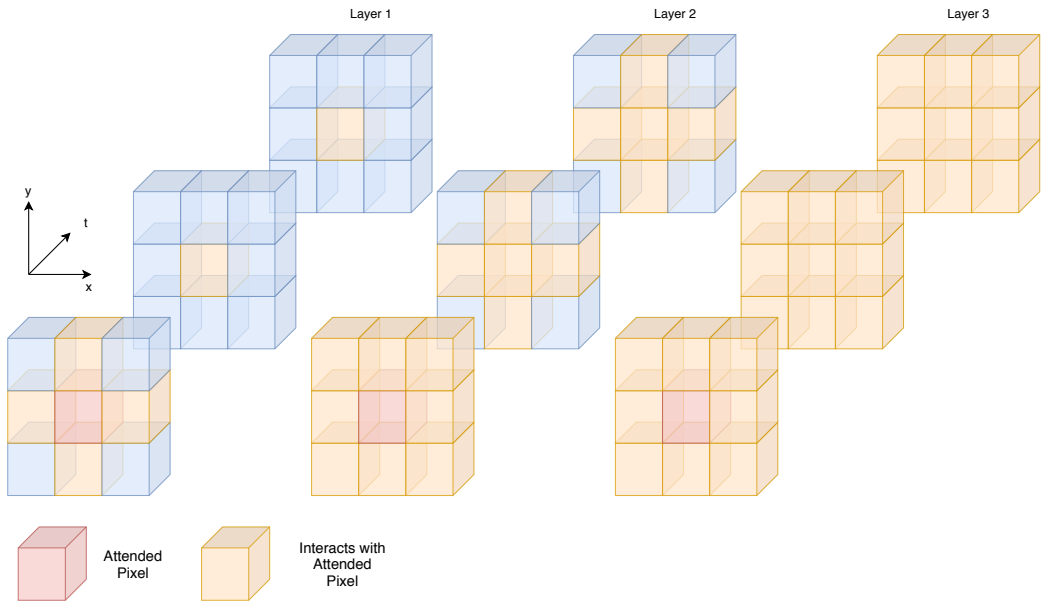

Figure 5.4
Interaction propagation from a given pixel via grid attention.

In Figure 5.4 we illustrate how grid attention propagates interactions from a single attended pixel to all other pixels in three sequential layers. The first grid attention layer propagates information to other pixels in the same frame vertically and horizontally, and to pixels at the same spatial location in all other frames. The second layer propagates interactions to the entire current frame, and vertical and horizontal axes in other frames. Finally, the third layer propagates information to all pixels in the video feature tensor.

To demonstrate mathematically information propagation in grid attention we consider, for sake of clarity, a function $\tilde{\Omega}$ that is Ω without the softmax in Equation 5.3. Further assume that our query is our video feature tensor, and our key and value are both our object feature tensor, i.e., $\mathcal{Q} = \mathcal{T}$, $\mathcal{K} = \mathcal{R}$, and $\mathcal{V} = \mathcal{R}$. The first layer outputs, for each pixel $p \equiv (x, y, t)$,

$$\begin{aligned}\Phi\big(\tilde{\Omega}(\mathcal{T},\mathcal{R}),\mathcal{R}\big)_{xyt} = &\sum_{i=1}^{W}\big(\mathcal{T}_{xyt}^{\top}\mathcal{R}_{iyt}\big)\mathcal{R}_{iyt} + \\ &\sum_{\substack{j=1\\ j\neq y}}^{H}\big(\mathcal{T}_{xyt}^{\top}\mathcal{R}_{xjt}\big)\mathcal{R}_{xjt} + \\ &\sum_{\substack{k=1\\ k\neq t}}^{T}\big(\mathcal{T}_{xyt}^{\top}\mathcal{R}_{xyk}\big)\mathcal{R}_{xyk}\end{aligned} \tag{5.5}$$

We can show that composing one application of Φ and $\tilde{\Omega}$ with two applications of self-attention on $\mathcal{T}$, which we denote for brevity as $\Phi^{3}_{\tilde{\Omega}}$, produces

$$\begin{aligned}\Phi^{3}_{\tilde{\Omega}}(\mathcal{T},\mathcal{R})_{xyt} = \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{T} &\big(\mathcal{T}_{xyt}^{\top}\mathcal{T}_{iyt}\big) \\ &\big(\mathcal{T}_{iyt}^{\top}\mathcal{T}_{ijt}\big) \\ &\big(\mathcal{T}_{ijt}^{\top}\mathcal{R}_{ijk}\big)\mathcal{R}_{ijk} + \cdots,\end{aligned} \tag{5.6}$$

where $\cdots$ represents other similar third order terms. We show in Equation 5.6 that grid attention propagates information along "routes" through the video feature tensor: for a pixel at position (x, y, t) to interact with another pixel at an arbitrary position (i, j, k), interactions must propagate along a "route" through the video feature tensor of pairs of similar pixels. Just as we might give travel directions through a city grid such as "first walk ten blocks North, then walk three blocks East", grid attention interactions must propagate a fixed number of pixels in the X, Y and T directions, in some order, before connecting the interaction source pixel with its target pixel.

Consider what happens if we replace the value $\mathcal{R}_{ijk}$ returned by the inner cross-attention in Equation 5.6 with a foreground mask value m_{ijk}. We see that the output routes reference mask values m_{ijk} over paths of feature vectors in the video tensor $\mathcal{T}$ that transitively correspond to reference features $\mathcal{R}_{ijk}$.

By replacing dense attention with grid attention we reduced the computational complexity of video attention from $O(C(THW)^2)$ to $O(C(T + H + W)THW)$, achieving our goal of making attention tractable for video.

Strided Attention

We also investigated strided attention as an alternative sparse attention method in addition to grid attention. Drawing inspiration from Child et al. (2019), we propagate information by following paths of locations through sparse connectivity patterns in our video feature tensor. We define a connectivity pattern $I = \{I_{p_0}, \ldots, I_{p_S}\}$ where I_p is a set of coordinates (i, j, k) that index a 3D tensor. Each layer of strided attention then returns, for each pixel p,

$$\texttt{StridedAttn}\big(\mathcal{Q}, \mathcal{K}, \mathcal{V}\big)_p = \texttt{softmax}\big(\mathcal{Q}_p \mathcal{K}_{I_p}^\top\big)\mathcal{K}_{I_p}. \tag{5.7}$$

We define I_p as a generalization of Child et al.'s strided attention from 1D to 3D. Our strided attention uses two different connectivity patterns I_p^1 and I_p^2 corresponding to separate, sequential heads of multihead attention. The first connectivity pattern I_p^1 routes to all pixels in a cube of side-length h from p, i.e., $I_p^1 = (p + (l_x, l_y, l_z) \ : \ l_x, l_y, l_z < h)$. The subsequent connectivity pattern I_p^1 routes to all pixels in the video tensor that can reach p by taking steps of size h along each axis, i.e., $I_p^2 = (p + (l_x, l_y, l_z) \ : \ l_x, l_y, l_z \mod h = 0)$. We choose $h \approx \sqrt{H}$ to reduce the computational complexity by a square root to $O(C(THW)^{3/2})$ from $O(C(THW)^2)$.

The relative efficiency of grid and sparse attention depends on the size of T, since we assume that H and W are comparably large and both larger than T. During training when we subsample video sequences, sparse attention costs about 1.3 to 1.4 times as many operations as grid attention, given our training configuration where $H, W \in \{64, 128\}$, and $T \approx 8$.

Appearance Model

Johnander et al. showed that appearance models are effective for VOS, particularly in generalizing to classes not seen in the training set (Johnander et al., 2019). Hence, following Johnander et al., we define object appearance models as

$$\mu_o = \frac{\sum_p \alpha_{p,o} x_p}{\sum_p \alpha_{p,o}} \tag{5.8a}$$

$$\Sigma_o = \frac{\sum_p \alpha_{p,o} \text{diag}\{(x_p - \mu_o)^2 + r_k\}}{\sum_p \alpha_{p,o}} \tag{5.8b}$$

where o indexes objects in the video, p indexes pixels in the video, $\alpha_{p,o} \in \{0, 1\}$ indicates whether a given pixel p in the reference frame belongs to object o, and r_k are model parameters.

Since $\alpha_{p,o}$ takes discrete values of unity inside an object, and zero outside, in our current definition we still form object appearance models (μ_o, Σ_o) independently of other objects in a video.

We hypothesize that future work on a learned component that incorporates explicit knowledge about visual context, including both other objects and background, into an object's appearance model would improve the ability to extract object representations that are discriminative.

The log probability of a pixel's feature vector being generated by a given object's appearance model forms an additional feature, which is concatenated to the input to the self-attention block in the SST encoder.

5.4 Experiments and Results

We present benchmark experiment results against state-of-the-art (SOTA) methods on the DAVIS 2017 (Pont-Tuset et al., 2017) and YouTube-VOS (Xu et al., 2018b) datasets. We further analyze the effect of different sparse attention operators and positional encodings through ablation studies using YouTube-VOS.

YVOS

Table 5.1
Comparison with SOTA methods on YouTube-VOS (Xu et al., 2018b). We compute region similarity $\mathcal{J}$ over seen and unseen categories, then average those scores with contour accuracy $\mathcal{F}$ seen and unseen to get overall score $\mathcal{G}$. We compute region similarity and contour accuracy as in Perazzi et al. (2016). We distinguish methods by those that use online finetuning (O-Ft), and those that do not.

Method	O-Ft	$\mathcal{G}$ overall (%)	$\mathcal{J}$ seen (%)	$\mathcal{J}$ unseen (%)
S2S (Xu et al., 2018a)	✓	64.4	71.0	55.5
OSVOS (Caelles et al., 2017)	✓	58.8	59.8	54.2
OnAVOS (Voigtlaender and Leibe, 2017)	✓	55.2	60.1	46.6
MSK (Khoreva et al., 2017)	✓	53.1	59.9	45.0
OSMN (Yang et al., 2018)	✗	51.2	60.0	40.6
S2S (Xu et al., 2018a)	✗	57.6	66.7	48.2
RGMP (Oh et al., 2018)	✗	53.8	59.5	45.2
A-GAME (Johnander et al., 2019)	✗	66.0	66.9	**61.2**
SST (Grid)	✗	66.5	67.8	60.2
SST (Strided)	✗	**68.1**	**69.9**	60.8

YouTube-VOS (Xu et al., 2018b) is a large scale VOS dataset comprised of 4453 YouTube video clips spanning 94 object categories. YouTube-VOS includes an official validation set of 474

videos with heldout labels, which can be evaluated only through an evaluation server. The YouTube-VOS validation set contains 26 object categories that are unique to the validation set, used to test the generalization capability of VOS models to object classes unseen in the training set. The convention is to compute region similarity $\mathcal{J}$ and contour accuracy $\mathcal{F}$ as defined by Perazzi et al. (2016). As a single metric for comparing results, it is also customary to compute overall score $\mathcal{G}$ as the average of four values comprising region similarity and contour accuracy scores for seen and unseen classes.

In Table 5.1 we present our model's results on YouTube-VOS alongside previous SOTA results. Our model (SST) performs favourably against all previous methods in overall score $\mathcal{G}$, even methods that use online finetuning (denoted by O-Ft).

Note that our unique method performs competitively against recurrent methods that have undergone multiple research and development cycles where one method builds on the foundation of another, for example Johnander et al. (2019) extends Oh et al. (2018), which in turn extends Khoreva et al. (2017).

DAVIS

Table 5.2
Comparison with SOTA methods on DAVIS2017 (Pont-Tuset et al., 2017).

Method	O-Ft	$\mathcal{J}\&\mathcal{F}$ mean (%)	$\mathcal{J}$ (%)	$\mathcal{F}$ (%)
CINM (Bao et al., 2018)	✓	70.6	67.2	74.0
OSVOS-S (Maninis et al., 2018)	✓	68.0	64.7	71.3
OnAVOS (Voigtlaender and Leibe, 2017)	✓	65.4	61.6	69.1
OSVOS (Caelles et al., 2017)	✓	60.3	56.6	63.9
DyeNet (Li and Loy, 2018)	✗	69.1	67.3	71.0
RGMP (Oh et al., 2018)	✗	66.7	64.8	68.6
VM (Hu et al., 2018b)	✗	-	56.5	-
FAVOS (Cheng et al., 2018)	✗	58.2	54.6	61.8
OSMN (Yang et al., 2018)	✗	54.8	52.5	57.1
A-GAME (Johnander et al., 2019)	✗	70.0	67.2	72.7
SST (Strided)	✗	53.2	50.5	55.9

DAVIS2017 is the latest dataset in the DAVIS initiative to promote VOS research. DAVIS2017 comprises 150 sequences, which include 376 separately annotated objects (Pont-Tuset et al., 2017).

We additionally evaluate our method on DAVIS2017 (Pont-Tuset et al., 2017), and compare our results with SOTA in Table 5.2. We report our DAVIS results following the traditionally used region similarity $\mathcal{J}$ and contour accuracy $\mathcal{F}$ metrics as well as their mean $\mathcal{J}\&\mathcal{F}$. According to our DAVIS2017 evaluation, SST performs relatively poorly compared to prior work, since SST achieves a mean $\mathcal{J}\&\mathcal{F}$ score of 53.2, whereas previous SOTA scored a $\mathcal{J}\&\mathcal{F}$ of 70.6. We believe that our method's performance on the DAVIS2017 validation set is reflective of a weakness of our current formulation. The DAVIS2017 validation set is made up of only 30 sequences, and since the DAVIS2017 evaluation is done per object, sequences with many objects are weighed far heavier in the evaluation. For example, a single sequence with a group of people dancing accounts for more than 10% of the final score.

We hypothesize that our method performs poorly on exactly the sequences that DAVIS2017 weighs heavily in evaluation, where there are many objects of the same class that need to be tracked in a video. We observe that in these cases our method performs well in the initial frames, and then our method's performance deteriorates rapidly. SST's most common failure mode is confusing multiple objects of the same class in medium and long temporal sequence lengths.

Long et al. showed that ConvNets learn correspondence, and furthermore ConvNet features can align intraclass instances (Long et al., 2014). By using attention for VOS we leverage this correspondence property of ConvNet features in order to do pixelwise tracking. However, to adapt attention to VOS we must extract representations that distinguish one object from another, whereas Long et al. showed that intraclass ConvNet features match so well that they are effective for intraclass alignment. We believe that the inductive bias of recurrent methods gives a temporal cue to the VOS model to favour matching features that are spatially nearby, since recurrent methods always propagate information from the previous frame. Since our method currently lacks this inductive bias, our model is unable to separate objects of the same class beyond the initial frames close to the reference frame. In future work we propose to inject this same "spatial locality" inductive bias into SST by updating our object-discriminative and reference features gradually with the predictions from earlier layers, instead of simply initializing object-discriminative and reference features with the reference frame.

We evaluate only on DAVIS2017 and not DAVIS2016 (Perazzi et al., 2016) because DAVIS2017 is strictly a more challenging superset of DAVIS2016. Furthermore DAVIS2016 contains only single object annotations and therefore we could make only limited evaluation of SST's ability to handle multi-object context using DAVIS2016. In general we consider YouTube-VOS the currently most superior benchmark for comparing VOS algorithms due to its large size, variety, evaluation on a heldout set using an evaluation server, as well as its unseen classes, which are useful for evaluating generalization.

Ablation Studies

We were motivated to use attention-based models for VOS due to spatiotemporal attention's ability to incorporate temporal context without incurring a decay in performance due to accumulating errors as is inherent to reccurent and optical flow methods, and also noticed by Yang et al. (2019). In Figure 5.5 we show a qualitative example on the YouTube-VOS validation set of SST handling foreground occlusion, where a motorcycle entirely occludes a person for one frame then the person becomes disoccluded in the following frame.

Figure 5.5
A qualitative example from YouTube-VOS validation of SST handling occlusion.

We used YouTube-VOS to perform ablation studies on interesting components of our method, including sparse attention operators and positional encodings. We first describe and compare different design choices for our positional encodings.

For spatial dimensions we followed Parmar et al. (2019) in using learned relative positional embeddings as the encoding, following Parmar et al.'s findings that relative are superior to absolute positional embeddings for spatial attention. Since we contribute an extension of attention-based models from 2D to 3D, we focused on experimental investigation of different temporal positional encodings.

As we described in Equation 5.7, in sparse video attention we propagate information through the outer product of query tensor $\mathcal{Q}_p$ with key tensor $\mathcal{K}_{I_p}$ indexed by connectivity pattern I_p. Adding relative positional embeddings, Equation 5.7 becomes

$$\texttt{StridedAttn}\big(\mathcal{Q}, \mathcal{K}, \mathcal{V}\big)_p = \texttt{softmax}\big(\mathcal{Q}_p\big(\mathcal{K}_{I_p}^\top + \mathcal{E}_{\tilde{I}_p}^\top\big)\big)\mathcal{K}_{I_p} \tag{5.9}$$

where $\tilde{I}_p = (f(q,p) : q \in I_p)$ is the sequence of relative position indices for points q in sequence I_p relative to pixel p, and $f : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ is a function mapping pairs of positions to scalar indices. For grid attention pixel q would differ along at most one of the three spatiotemporal axes X, Y, T, so the relative positional embedding for pair (p, q) would correspond to either $\Delta_x \equiv q_x - p_x$, $\Delta_y \equiv q_y - p_y$, or $\Delta_t \equiv q_t - p_t$. For strided attention, the relative positional

embedding would be the concatenation of three separate embeddings corresponding to Δ_x, Δ_y, and Δ_t.

In addition to relative positional embeddings, we also investigated sinusoidal positional encodings for the temporal dimension, as used by Vaswani et al. (2017) in Transformers for language translation. We hypothesized that sinusoidal positional encodings would be superior to absolute positional embeddings because of the data imbalance of absolute temporal positions in VOS datasets, which are skewed towards lower frame numbers. Sinusoidal positional encodings can be interpolated or extrapolated to generalize to underrepresented absolute frame numbers, whereas absolute positional embeddings have no such generalization mechanism. Furthermore, we hypothesized that sinusoidal positional encodings would be superior to relative positional embeddings because the absolute position information encoded in the sinusoidal representation encodes information about a given pixel's temporal distance from the reference frame, whereas relative positional embeddings encode no information related to distance from the reference frame.

Table 5.3
Comparison of positional encoding schemes for the "grid" sparse attention variant of video attention.

Positional Encoding	Grid $\mathcal{G}$
None	61.1
K Embedding	61.8
T Embedding	60.5
T Sinusoidal	**62.6**

We present our results comparing different positional encodings in Table 5.3 using the overall score $\mathcal{G}$ on the YouTube-VOS validation set for both grid and strided sparse attention variants. The positional encoding labeled "None" is our baseline attention with no positional information, while all remaining positional encodings use relative positional embeddings for spatial dimensions X, Y. "K Embedding" and "T embedding" refer to using relative and absolute temporal positional embeddings, respectively, and "T sinusoidal" refers to using a sinusoidal temporal positional encoding. Relative temporal positional embeddings outperformed the baseline, while absolute temporal positional embeddings underperformed the baseline, possibly due to the data imbalance of absolute temporal positions. Sinusoidal temporal positional encodings performed best, which is in line with our hypothesis that information about distance-from-reference is important in positional encodings for VOS.

At training time we used TSN sampling (Wang et al., 2016) to subsample sequences to a fixed number of frames since backpropagation constrained our memory usage. However at test

Table 5.4
Comparison of sparse and dense evaluation on YouTube-VOS (Xu et al., 2018b) for both strided and grid sparse attention variants.

Method	Evaluation	$\mathcal{G}$
SST (Grid)	Sparse	65.8
SST (Grid)	Dense	66.5
SST (Strided)	Sparse	66.7
SST (Strided)	Dense	**68.1**

time we do not have the same memory constraint, so we have the ability to simultaneously predict on an entire sequence of frames at once. In Table 5.4 we compare the TSN and "all frames" test time sampling methods, which we refer to as sparse and dense evaluation, respectively. We show that simultaneously attending to all frames densely produces superior predictions for both grid and strided sparse attention variants. We also maintain reasonable accuracy when using TSN sampling at test time to subsample sequence, showing an advantage of our method over recurrent methods that would have to process every preceding frame in order to predict on a given timestep.

Discussion

Our work represents the first algorithm for VOS purely based on end-to-end attention. Future work in attention-based models for VOS could be analogous to Transformer models' progression on language translation tasks, where researchers successfully applied Transformers to increasingly long sequences. For example, Dai et al. combined recurrence with attention to translate arbitrary-length sequences (Dai et al., 2019), and Kitaev et al. introduced locality-sensitive hashing instead of dot-product attention, to reduce computational complexity from squared to $O(N \log N)$ while using reversible networks to model arbitrary-length sequences with constant memory usage (Kitaev et al., 2020). In order to evaluate VOS on long sequences the VOS community would have to overcome a dataset creation challenge, since the current benchmark dataset YouTube-VOS contains sequences with at most 36 labeled frames, sampled at 6 frames per second. We propose that future work could use interactive and semi-automatic annotation methods, based on the existing high-quality VOS models, to create datasets with longer and therefore more challenging sequences.

5.5 Conclusions

We presented Sparse Spatiotemporal Transformers (SST), which constitutes the first application of an entirely attention-based model for video object segmentation (VOS). We showed that SST is capable of state-of-the-art results on the benchmark VOS dataset YouTube-VOS, attaining an overall score of $\mathcal{G} = 68.1$, while having superior runtime scalability compared with previous state of the art methods. We provide code to reproduce all experiments in our work, including sparse video-attention operator implementations, so that the community can build on the promising idea of using attention-based models for video, in the VOS domain and beyond.

Chapter 6

Discussion

In this thesis we investigated representation learning focused on developing improved attention and fusion operators for computer vision. We introduced the idea of learning fusion operators via neural architecture search, and proposed a search space for fusion operator search for VQA. We demonstrated the potential of our search space by manually finding configurations of our search space that outperform the baseline fusion model, MUTAN (Ben-younes et al., 2017), based on Tucker decomposition. Our improved fusion operator used a gating mechanism to conditionally enable or disable features, ensembled activation functions, and added additional learned nonlinearity via a neural network output function. We showed that our method achieved an absolute percentage point increase of 1.1% in top-1 accuracy over the baseline MUTAN model. Furthermore, our proposed fusion operator search space holds promise for use with neural architecture search to discover new, even superior fusion operators.

Although not a focus of our main article, our multimodal fusion operator for VQA is used not only to compute classification logits, but also to compute spatial attention over features extracted from the input image. Therefore our performance improvements for VQA may be attributed to both superior predictive capacity and improved attention computation. In our second article we investigated attention in greater detail, by exploring design choices for constructing a video object segmentation (VOS) architecture entirely from attention building blocks. We introduced the attention-based Transformer architecture (Vaswani et al., 2017) to VOS, while exploring sparse video-attention variants, which is at time of writing an underexplored area. In particular, we wrote custom CUDA implementations for 3D "grid attention" and "strided attention" operators and provide these implementations to the community for use on VOS and other dense video prediction tasks.

Both of our articles, then, could be viewed as exploring attention variants applied to different subfields of computer vision. We expect that attention's importance in computer vision, alongside the broader machine learning community, will continue to grow in the coming years. Early deep learning applied to computer vision has been built on convolutional neural networks (CNNs), which were designed with an effective translation-equivariant prior for processing images. As computer vision research moves beyond the 2D image domain to more complex domains such as 3D and video, we believe that CNNs will face computational challenges, coupled with decreased effectiveness of the convolutional layer as a prior in these domains.

We believe that in domains where CNNs are limited, either due to computation or modeling capacity, models based on attention have potential. Attention-based models have recently

been shown by Cordonnier et al. (2020) to be general enough even to contain convolutional layers as a subset, under certain conditions. Furthermore, attention models hold the promise of being a fitting component for computing on a variety of different inputs, including variable length sequences, as proven by the success of Transformers for language processing and now video object segmentation.

One area of computer vision where attention methods could be further developed is in 3D vision scenarios where data can be represented as a graph. Such a scenario exists in 3D mesh representations, such as the low-level representation of 3D faces. In recent work, Ranjan et al. (2018) have shown that graph neural networks (GNNs) operating as an autoencoder on 3D face meshes outperform traditional linear methods. Furthermore, Knyazev et al. (2019) showed that attention, if initialized correctly, can improve the performance of GNNs while making GNNs generalize well to noisy test inputs. Based on Knyazev et al's findings, pursuing research into attention operators for computation on 3D meshes holds potential for improving the interpretability, performance and generalization of computer vision tasks that use 3D representations.

Additionally, attention-based architectures have the ability to conditionally compute via top-k pooling, as proposed by Gao and Ji (2019). The conditional pooling of attention layers contrasts with the fixed, general pooling of convolutional layers, and provides a method for attention-based models to find efficiencies and thereby achieve faster model runtimes in comparison to CNNs. Simultaneously, in order to realize these efficiencies on real devices, the community must spend research and engineering effort designing accelerated software and hardware implementations for attention networks, as has already been done for CNNs. We look forward to the community's future progress in developing ever superior attention-based models, and applying attention to underexplored areas of computer vision and machine learning.

References

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2017). Bottom-up and top-down attention for image captioning and vqa. *arXiv preprint arXiv:1707.07998*.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). VQA: Visual Question Answering. In *Int. Conf. on Computer Vision (ICCV)*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bao, L., Wu, B., and Liu, W. (2018). CNN in MRF: video object segmentation via inference in A cnn-based higher-order spatio-temporal MRF. In *CVPR*.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. (2017). Neural optimizer search with reinforcement learning. In *Int. Conf. on Machine Learning (ICML)*.

Ben-younes, H., Cadene, R., Cord, M., and Thome, N. (2017). MUTAN: Multimodal tucker fusion for visual question answering. In *IEEE Int. Conf. on Computer Vision (ICCV)*.

Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2017). One-shot video object segmentation. In *CVPR*.

Charikar, M., Chen, K., and Farach-Colton, M. (2002). Finding frequent items in data streams. In *Proc. 29th Int. Colloquium on Automata, Languages and Programming*, pages 693–703.

Cheng, J., Tsai, Y.-H., Hung, W.-C., Wang, S., and Yang, M.-H. (2018). Fast and accurate online video object segmentation via tracking parts. In *CVPR*.

Cheng, J., Tsai, Y.-H., Wang, S., and Yang, M.-H. (2017). Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. In *arXiv:1904.10509*.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Cordonnier, J.-B., Loukas, A., and Jaggi, M. (2020). On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*.

Denil, M., Bazzani, L., Larochelle, H., and de Freitas, N. (2012). Learning where to attend with deep architectures for image tracking. *Neural Computation*, 24:2151–2184.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

Duke, B., Ahmed, A., Wolf, C., Aarabi, P., and Taylor, G. (2020). Scalable video object segmentation with sparse spatiotemporal transformers. unpublished.

Duke, B. and Taylor, G. W. (2018). Generalized hadamard-product fusion operators for visual question answering. In *2018 15th Conference on Computer and Robot Vision (CRV)*.

Durand, T. (2020). Learning user representations for open vocabulary image hashtag prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dutt Jain, S., Xiong, B., and Grauman, K. (2017). Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *CVPR*.

Feng, D., Haase-Schuetz, C., Rosenbaum, L., Hertlein, H., Duffhauss, F., Gläser, C., Wiesbeck, W., and Dietmayer, K. C. J. (2019). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. In *arXiv:1308.0850*.

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M. (2016). Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proc. 2016 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 457–468.

Gao, H. and Ji, S. (2019). Graph u-nets. In *Proceedings of the 36th International Conference on Machine Learning*.

Gao, Y., Beijbom, O., Zhang, N., and Darrell, T. (2016). Compact bilinear pooling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 317–326.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*.

Girdhar, R., Carreira, J. a., Doersch, C., and Zisserman, A. (2019). Video action transformer network. In *CVPR*.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Graves, A. (2013). Generating sequences with recurrent neural networks. In *arXiv:1308.0850*.

Griewank, A. and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, USA, second edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hu, P., Wang, G., Kong, X., Kuen, J., and Tan, Y.-P. (2018a). Motion-guided cascaded refinement network for video object segmentation. In *CVPR*.

Hu, Y.-T., Huang, J.-B., and Schwing, A. (2017). Maskrnn: Instance level video object segmentation. In *Advances in Neural Information Processing Systems 30*.

Hu, Y.-T., Huang, J.-B., and Schwing, A. G. (2018b). Videomatch: Matching based video object segmentation. In *ECCV*.

Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., and Liu, W. (2019). Ccnet: Criss-cross attention for semantic segmentation.

Jampani, V., Gadde, R., and Gehler, P. V. (2017). Video propagation networks. In *CVPR*.

Jang, W.-D. and Kim, C.-S. (2017). Online video object segmentation via convolutional trident network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johnander, J., Danelljan, M., Brissman, E., Khan, F. S., and Felsberg, M. (2019). A generative appearance model for end-to-end video object segmentation. In *CVPR*.

Kazakos, E., Nagrani, A., Zisserman, A., and Damen, D. (2019). Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Khoreva, A., Perazzi, F., Benenson, R., Schiele, B., and Sorkine-Hornung, A. (2017). Learning video object segmentation from static images. In *CVPR*.

Kim, C., Shin, M., Garg, A., and Gowda, D. (2019). Improved Vocal Tract Length Perturbation for a State-of-the-Art End-to-End Speech Recognition System. In *Proc. Interspeech 2019*.

Kim, J.-H., On, K. W., Lim, W., Kim, J., Ha, J.-W., and Zhang, B.-T. (2017). Hadamard Product for Low-rank Bilinear Pooling. In *The 5th Int. Conf. on Learning Representations (ICLR)*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd Int. Conf. on Learning Representations (ICLR)*.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. In *Advances in Neural Information Processing Systems 28 (NIPS)*.

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 972–981.

Knyazev, B., Taylor, G., and Amer, M. (2019). Understanding attention and generalization in graph neural networks. In *Neural Information Processing Systems (NeurIPS)*. Early version appeared at the International Conference on Learning Representations (ICLR) Workshop on Representation Learning on Graphs and Manifolds.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. In *SIAM review*, pages 455–500.

Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, USA.

Lahat, D., Adalı, T., and Jutten, C. (2015). Multimodal data fusion: An overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103:1449–1477.

Larochelle, H. and Hinton, G. E. (2010). Learning to combine foveal glimpses with a third-order boltzmann machine. In *NIPS*.

LeCun, Y. (1989). Generalization and network design strategies. In Pfeifer, R., Schreter, Z., Fogelman, F., and Steels, L., editors, *Connectionism in Perspective*, Zurich, Switzerland. Elsevier. an extended version was published as a technical report of the University of Toronto.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature Cell Biology*, 521(7553):436–444.

Li, B., Yan, J., Wu, W., Zhu, Z., and Hu, X. (2018). High performance visual tracking with siamese region proposal network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, X. and Loy, C. C. (2018). Video object segmentation with joint re-identification and attention-aware mask propagation. In *The European Conference on Computer Vision (ECCV)*.

Lin, M., Chen, Q., and Yan, S. (2014a). Network in network. *Int. Conf. on Learning Representations (ICLR)*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014b). Microsoft coco: Common objects in context. In *European Conf. on Computer Vision (ECCV)*, pages 740–755.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2018). Hierarchical representations for efficient architecture search. *Int. Conf. on Learning Representations (ICLR)*.

Long, J. L., Zhang, N., and Darrell, T. (2014). Do convnets learn correspondence? In *Advances in Neural Information Processing Systems 27*.

Luiten, J., Voigtlaender, P., and Leibe, B. (2018). Premvos: Proposal-generation, refinement and merging for video object segmentation. In *Asian Conference on Computer Vision*.

Lyons, R. G. (1996). *Understanding Digital Signal Processing*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition.

Lüscher, C., Beck, E., Irie, K., Kitza, M., Michel, W., Zeyer, A., Schlüter, R., and Ney, H. (2019). RWTH ASR Systems for LibriSpeech: Hybrid vs Attention. In *Proc. Interspeech 2019*.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Int. Conf. on Machine Learning (ICML)*.

Malkomes, G., Schaff, C., and Garnett, R. (2016). Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 2900–2908.

Maninis, K.-K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Van Gool, L. (2018). Video object segmentation without temporal information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Mnih, V., Heess, N., Graves, A., and kavukcuoglu, k. (2014). Recurrent models of visual attention. In *Advances in Neural Information Processing Systems 27*.

Myachykov, A. and Posner, M. (2005). *Attention in Language*, pages 324–329.

Neverova, N., Wolf, C., Taylor, G. W., and Nebout, F. (2016). Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Oh, S. W., Lee, J.-Y., Sunkavalli, K., and Kim, S. J. (2018). Fast video object segmentation by reference-guided mask propagation. In *CVPR*.

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems 32*.

Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*.

Perez-Rua, J.-M., Vielzeuf, V., Pateux, S., Baccouche, M., and Jurie, F. (2019). Mfas: Multimodal fusion architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., and Van Gool, L. (2017). The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*.

Ramachandram, D. and Taylor, G. W. (2017). Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108.

Ramachandran, P., Zoph, B., and Le, Q. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Ranjan, A., Bolkart, T., Sanyal, S., and Black, M. J. (2018). Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. In *Int. Journal of Computer Vision*, pages 211–252.

Salvador, A., Bellver, M., Baradad, M., Marques, F., Torres, J., and Giro-i Nieto, X. (2017). Recurrent neural networks for semantic instance segmentation. In *arXiv:1712.00617*.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2377–2385.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*.

Teney, D., Anderson, P., He, X., and Hengel, A. v. d. (2017). Tips and tricks for visual question answering: Learnings from the 2017 challenge. *arXiv preprint arXiv:1708.02711*.

Tokmakov, P., Alahari, K., and Schmid, C. (2017). Learning video object segmentation with visual memory. In *ICCV*.

Tsai, Y.-H., Yang, M.-H., and Black, M. J. (2016). Video segmentation via object flow. In *CVPR*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Ventura, C., Bellver, M., Girbau, A., Salvador, A., Marques, F., and Giro-i Nieto, X. (2019). Rvos: End-to-end recurrent network for video object segmentation. In *CVPR*.

Vo, N., Jiang, L., Sun, C., Murphy, K., Li, L.-J., Fei-Fei, L., and Hays, J. (2019). Composing text and image for image retrieval-an empirical odyssey. In *CVPR*.

Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., and Chen, L.-C. (2019). Feelvos: Fast end-to-end embedding learning for video object segmentation. In *CVPR*.

Voigtlaender, P. and Leibe, B. (2017). Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Val Gool, L. (2016). Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*.

Wang, X., Jabri, A., and Efros, A. A. (2019). Learning correspondence from the cycle-consistency of time. In *CVPR*.

Xu, N., Yang, L., Fan, Y., Yang, J., Yue, D., Liang, Y., Price, B., Cohen, S., and Huang, T. (2018a). Youtube-vos: Sequence-to-sequence video object segmentation. In *ECCV*.

Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., and Huang, T. S. (2018b). Youtube-vos: A large-scale video object segmentation benchmark. In *ECCV*.

Yang, L., Wang, Y., Xiong, X., Yang, J., and Katsaggelos, A. K. (2018). Efficient video object segmentation via network modulation. In *CVPR*.

Yang, Z., Wang, Q., Bertinetto, L., Hu, W., Bai, S., and Torr, P. H. S. (2019). Anchor diffusion for unsupervised video object segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Zhao, H., Qi, X., Shen, X., Shi, J., and Jia, J. (2018). Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *5th Int. Conf. on Learning Representations (ICLR)*.