OmniNet: A unified architecture for multi-modal multi-task learning

Anonymous authors
Paper under double-blind review

Abstract

Transformer is a popularly used neural network architecture, especially for language understanding. We introduce an extended and unified architecture that can be used for tasks involving a variety of modalities like image, text, videos, etc. We propose a spatio-temporal cache mechanism that enables learning spatial dimension of the input in addition to the hidden states corresponding to the temporal input sequence. The proposed architecture further enables a single model to support tasks with multiple input modalities as well as asynchronous multi-task learning, thus we refer to it as *OmniNet*. For example, a single instance of *OmniNet* can concurrently learn to perform the tasks of part-of-speech tagging, image captioning, visual question answering and video activity recognition. We demonstrate that training these four tasks together results in about three times compressed model while retaining the performance in comparison to training them individually. We also show that using this neural network pre-trained on some modalities assists in learning unseen tasks such as video captioning and video question answering. This illustrates the generalization capacity of the self-attention mechanism on the spatio-temporal cache present in *OmniNet*.

1 Introduction

Transformer (Vaswani et al., 2017) is currently one of the best performing models for any sequence transduction tasks, especially those involving natural language. It is originally designed for a single task at a time. In fact, most of the generic deep learning architectures (Wu et al., 2016; Szegedy et al., 2016; Battenberg et al., 2017) that have been designed and developed are able to learn, albeit very well, a single task and handle one task specific input domain like image, text or audio. Furthermore with these models, we often rely on the generalization capability of the trained network to guarantee performance on unseen examples. Transfer learning (Shu et al., 2018; Hu et al., 2018) is another popular paradigm used to adapt the model to learn a related task with similar input domain. The success of neural networks across these challenges is known to be due to their ability in learning effective representations of the data. For example, the self-attention mechanism in Transformers can capture the global temporal dependence in sequential data very well. Naturally, the question arises whether we can extend these architectures, like the Transformer, to be able to learn shared representations from multiple input domains and to be able attend on these representations to perform a multitude of tasks concurrently.

The research into multi-task models that learn to solve varied tasks across a multitude of input domains is not new. Work done in Ngiam et al. (2011) demonstrates an architecture capable of learning a shared representation across audio and video modalities. Similarly in Collobert & Weston (2008) a convolutional architecture has been designed to support a variety of NLP tasks. However, most of these architectures are designed to learn specific set of tasks with known input domains. To the best of our knowledge, there does not exist a single unified architecture that works out of the box for any combination of multi-modal inputs.

To address this gap, we extend Transformer towards a unified architecture, namely *OmniNet*, which enables a single model to support tasks with multiple input modalities and asynchronous multi-task learning. We consider that most real-life data like image, text, speech, video, etc. is a direct conjunction of spatial and temporal components. Therefore, we employ a spatio-temporal cache mechanism to learn a shared representation of the input data across the spatial (space) and temporal (time) dimension. Using a generalized *encode()* function, *OmniNet* can process and store spatio-temporal representation for each of the input domains and then *decode()* predictions across a multitude of tasks. In our experiments, we train a single instance of the *OmniNet* to solve a number of tasks spanning multiple multi-domains such as part-of-speech tagging, image captioning, visual question answering and video activity recognition. To make our work reproducible, open to scrutiny and further development, we will open source a demonstration of our system implemented using Pytorch (Paszke et al., 2017).

2 Related Work

Multi-task learning has been extensively studied in the literature, with applications to a wide set of problems ranging from natural language processing (NLP) (Collobert & Weston, 2008; Johnson et al., 2017; Dong et al., 2015) to speech recognition (Seltzer & Droppo, 2013; Krishna et al., 2018) to vision (Zhang et al., 2014; Chen et al., 2018; Anderson et al., 2018). It has also found its use in a combination of diverse tasks like image captioning and text translation and parsing (Luong et al., 2016; Zhao et al., 2018). However, most of these architectures assume the set of tasks to be known in advance. Similarly, multi-modal learning has been essential for solving a broad range of interesting problems such as Visual Question Answering (Kim et al., 2016; 2018) and Video Question Answering (Lei et al., 2018). Again, the state-of-the-art models are highly specific to the objective in hand and not easily adaptable to different tasks or domains. (Kaiser et al., 2017) proposed MultiModel architecture for learning multiple tasks but lacks support for multi-modal tasks with more than one input domains such as visual question answering.

3 Proposed Model

We propose a unified architecture, namely *OmniNet*, to enable learning multi-modal tasks with multiple input domains and support generic multi-tasking for any set of tasks. The *OmniNet* architecture consists of multiple sub-networks, called peripheral networks, connected to a common central neural network called the Central Neural Processor (CNP) (Figure 1). Each peripheral network is used to encode the domain specific input into feature representations. In this work, we describe image, text and video peripherals (Section 3.1). One can add more, say speech peripheral, depending on the task. The output representation of a peripheral network is always a spatio-temporal tensor $x \in \mathbb{R}^{t \times s \times d_{model}}$, where t & s are the temporal and spatial dimensions of the input respectively, and d_{model} is the model dimension input to the CNP.

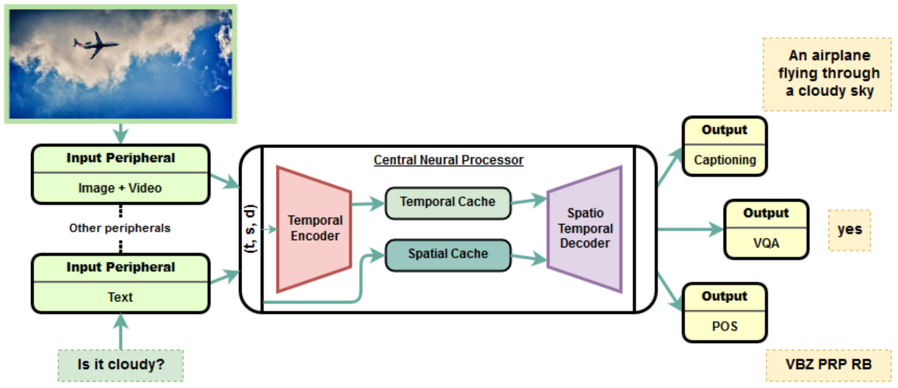

Figure 1: *OmniNet* performing image captioning, visual question answering and POS tagging at once

The spatio-temporal representations generated by the peripheral networks corresponding to each input domain are then processed by the CNP. The CNP uses fully attention based encoder-decoder (Bahdanau et al., 2015; Sutskever et al., 2014; Cho et al., 2014) model for sequence transduction similar to the Transformer architecture (Vaswani et al., 2017), which is the state-of-the-art for multiple language modeling tasks (Section 3.2). During the encoding stage, the CNP implements a generic *encode(x, D)* function to first process and store the spatio-temporal representations of the input, where $x \in \mathbb{R}^{t \times s \times d_{model}}$ is the spatio-temporal tensor produced by the peripheral networks and $D \in \mathbb{Z} : 0 \leq D < D_{len}$ is the domain id and D_{len} is the max number of domains supported by the CNP. The *encode()* function is called multiple times, once for each multi-modal input from respective peripheral. During the decoding stage, a *decode($y_{shifted}$, τ)* function is used to decode predictions as softmax probabilities, where $y_{shifted} \in \mathbb{Z}^{N-1}$ are the target outputs shifted one time-step to the right, N is the length of the output sequence; $\tau \in \mathbb{Z} : 0 \leq \tau < \tau_{len}$ is task id and τ_{len} is the total number of supported tasks. The decoding step is similar to Vaswani et al. (2017), modified to incorporate a two-step attention mechanism over spatial and temporal cache.

3.1 Peripheral networks

First, we elaborate on how we support multiple input domains using peripheral networks. A peripheral network can use a pre-trained model from existing literature to ultimately encode a given domain input to a standardized feature representation $x \in \mathbb{R}^{t \times s \times d_{model}}$, where t & s are the temporal and spatial dimensions of the input respectively, and d_{model} is the model dimension input to the Central Neural Processor. Here we detail text and vision peripherals and one can add more peripherals or alter the peripheral design depending on the task.

Vision peripheral: This peripheral uses a convolutional neural network to encode image and video inputs in the tasks. For an image of dimension $h \times w \times n_c$, this peripheral down-samples in to $h' \times w' \times n'_c$, where h, w, n_c are the height, width and number of input channels respectively. For a video, each frame is input to the peripheral to produce $F \times h' \times w' \times n'_c$, where F is the total number of frames in the video. The encoding vectors are then projected to dimension d_{model} using a fully connected layer. The output is then reshaped into a spatio-temporal tensor of $x \in \mathbb{R}^{t \times h'w' \times d_{model}}$, where $t = 1$ for an image and $t = F$ for a video. In our experiments, we use the pre-trained *ResNet-152* model, a variant of ResNet (He et al., 2016) consisting of 152 convolutional layers. We remove the final fully connected and avg-pooling layers to generate spatial feature representations for a given image/video.

Language peripheral: The Language peripheral uses byte-pair encoding (Sennrich et al., 2016) to generate subwords for a given input sentence. The subwords are passed to an embedding layer to generate subword embeddings of dimension d_{emb} and projected to dimension d_{model} using a fully connected layer. The output is then reshaped into a spatio-temporal tensor $x \in \mathbb{R}^{t \times 1 \times d_{model}}$, where t equal to number of subwords in the input sentence. As we do not have any spatial dimension in textual data, the spatial dimension of x from a Language peripheral is always 1. In our experiments, We used pre-trained subword embeddings with $d_{emb} = 300$ and $vocab\_size = 25000$ from Heinzerling & Strube (2018), which includes pre-trained subword embeddings of over 275 languages, to initialize the weights of the embedding matrix.

3.2 Central Neural Processor (CNP)

To process the spatio-temporal information in the input data, the CNP implements a spatial cache C_s, temporal cache C_t and a link array L. The spatial and temporal cache and the link array are a list of elements, initialized as empty before the encoding process. During the encoding stage, an *encode()* routine takes as input, the tensor x generated from the peripheral and corresponding domain/peripheral id D. This function processes the spatial and temporal information in the input x and stores them into the spatial cache C_s and the temporal cache C_t , respectively and stores their dimensions t & s in the link array. For a given task, this *encode()* routine is called K times, where K is the number of inputs in the task. Note that these inputs can belong to same or different domains.

Encode (x, D): For a given input $x \in \mathbb{R}^{t \times s \times d_{model}}$ and domain identifier D, the *encode()* routine is described in Algorithm 1. Since inputs can come from multiple peripherals, the algorithm first concatenates the input with the domain embedding to ensure a domain-aware encoding of the input (Steps 2 to 3). Steps 4 to 7 process the spatial information in x by unrolling the time dimension and adding these unrolled vectors into the spatial cache. Steps 8 to 10 process the temporal information in x by averaging the spatial dimension of x and then passing the averaged tensor to a self-attention based *TemporalEncoder*. This *TemporalEncoder* is similar to the encoder used in Vaswani et al. (2017) as shown in Figure 2 is used to calculate temporal embeddings of the input sequence. The output from the *TemporalEncoder* is appended to the temporal cache.

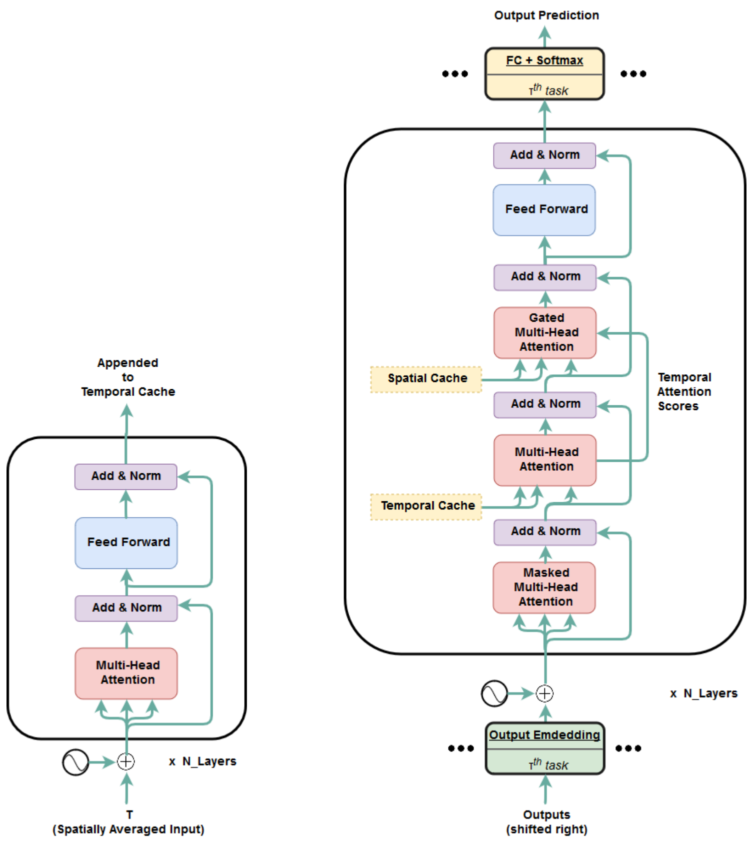

Figure 2: **left:** *TemporalEncoder* architecture; **right:** *OmniNet decode()* architecture.

Algorithm 1 *encode()*: Encodes spatial and temporal representations into spatial and temporal cache

```
Require: $x \in \mathbb{R}^{t \times s \times d_{model}}$, $D$, $C_s$, $L$, $C_t$
 1: $L \leftarrow L \cup (t \rightarrow s)$
 2: $D_{emb} \leftarrow EmbedLayer(D)$
 3: $x \leftarrow FC(Concat(x, D_{emb}), d_{model})$
 4: if $s > 1$ then
 5:     $S \leftarrow Reshape(x, (ts, d_{model}))$ {where, output $S = [S_1, \dots, S_{ts}]$ s.t. $S_i \in \mathbb{R}^{d_{model}}$ is a spatial feature vector.}
 6:     $C_s \leftarrow C_s \cup [S_1, \dots, S_{ts}]$ {Append spatial representations to spatial cache}
 7: end if
 8: $T \leftarrow (\sum_{i=1}^{s} x[:, i, :])/s$
 9: $T \leftarrow TemporalEncoder(T)$ {where, output $T = [T_1, \dots, T_t]$ s.t. $T_j \in \mathbb{R}^{d_{model}}$ is the encoding of temporal dimension in $x$.}
10: $C_t \leftarrow C_t \cup [T_1, \dots, T_t]$ {Append temporal representations to temporal cache}
```

The above encoding routine keeps appending spatio-temporal information to C_t & C_s for each input $x_k \in \mathbb{R}^{t_k \times s_k \times d_{model}}$. Note the superscript k to denote correspondence to k-th input of the task, where $k \in 1, \dots, K$. After K calls, we have the temporal cache $C_t = [T_1, \dots, T_R]$, where $R = \sum_{r=1}^{K} t_r$; the spatial cache $C_s = [S_1, \dots, S_P]$, where $P = \{\sum_p t_p * s_p : p \in 1, \dots, K \wedge s_p > 1\}$ and the link array $L = [(t_1 \rightarrow s_1), \dots, (t_K \rightarrow s_K)]$. Note that C_s can also be empty in case the *encode()* is only called with inputs with $s_k = 1 \forall k$. Next, we use the *decode()* routine to generate predictions as softmax probabilities.

Decode $(y_{shifted}, \tau)$: The architecture of the *decode()* function is shown in Figure 2. The *decode()* takes as argument the output labels $y_{shifted}$ shifted one time step to the right, a task id τ and generates predictions by attending from the spatial and temporal cache. The *decode()* function is structured similar to the decoder used in the *Transformer* architecture Vaswani et al. (2017) and jointly attends on the vectors stored in the temporal and spatial cache. Similar to Vaswani et al. (2017), the decoding first starts by attending over the output embeddings using masked multi-head scaled dot product attention. The attention layer for the temporal cache uses scaled dot-product attention with multiple heads as specified in Vaswani et al. (2017). Attention layer for the spatial cache, uses gated multi-head attention to attend over the elements of the spatial cache. For inputs with both time and space dimension (e.g. video), we want the spatial attention layer to attend more on frames which have relatively high attention scores in the temporal cache attention layer. Therefore, the attention score output from the temporal cache multi-head attention layer $A \in \mathbb{R}^{n_h \times N \times R}$, is used to calculate the tensor $G \in \mathbb{R}^{n_h \times N \times P}$ used for gating the attention score output in the spatial attention layer, where n_h is the number of heads in multi-head attention as described in Vaswani et al. (2017). The tensor G is calculated using A & L as detailed in Algorithm 2. Given[1] Q: the matrix of queries, K: keys of dimension d_k and V: values of dimension d_v, the scaled dot-product attention for the spatial layer is modified as:

$$\text{Attention}(Q, K, V, G) = \left(\text{Softmax}\left(\frac{QK^T}{\sqrt{d_v}}\right) \odot G\right)V \tag{1}$$

In order to use the same CNP for multiple tasks with varying output vocabularies, we use multiple output embedding layers $OutputEmbedLayer_1, \ldots, OutputEmbedLayer_{\tau_{len}}$, to generate the output embeddings for each task. At the final layer, we use multiple $(FC + Softmax)_1, \ldots, (FC + Softmax)_{\tau_{len}}$ classification layers for each task. We also calculate a task embedding vector using τ and always start decoding using the task embedding vector.

Algorithm 2 Calculate G using output scores from temporal attention and link array

```
Require: $L$, $A$
 1: $idx \leftarrow 0$
 2: for each t, s in L do
 3:     $G \leftarrow []$
 4:     if $s > 0$ then
 5:         $A' \leftarrow A[:, :, idx : idx + t]$
 6:         $A' \leftarrow Expand(A', (n_h, N, t, s))$ {where $Expand(tensor, dimension)$ expands tensor
            according to a given dimension}
 7:         $A' \leftarrow Reshape(A', (n_h, N, ts))$
 8:         $G \leftarrow G \cup A'$ {Append the respective temporal attention scores to $G$}
 9:     end if
10:     $idx \leftarrow idx + t$
11: end for
12: $G \leftarrow Stack(G)$ {Stack the list of tensors to construct tensor $G$ of dimension $(n_h, N, P)$}
```

3.3 Multi-task learning

In order to train a single a model simultaneously on mutiple tasks we used the HogWild training approach as described in Recht et al. (2011). Similar to the approach described in Mnih et al. (2016), the main process holds a global copy of the model. We create separate worker processes for each task, where each process maintains a local copy of a model. At each training iteration, each process starts by synchronizing its local model with the global copy. This is done through forward and backward propagation on its local copy and then copying the locally computed gradients to the global model asynchronously. Each process then calls the global model optimizer asynchronously to update the weights of the global model. Instead of storing the model in CPU as in Mnih et al. (2016) we always store the local copies across multiple GPUs.

4 Tasks and Setup

To evaluate the effectiveness of our proposed framework for tasks spanning diverse modalities, we choose a set covering all possible spatio-temporal data archetypes: Image Captioning, Part-of-

[1]For brevity, we reuse the notations of Vaswani et al. (2017) in this description

Speech (POS) tagging, Visual Question Answering (VQA) and Video-activity Recognition. Each of these tasks explores a unique potential spatio-temporal configuration of the input containing different values of t and s, where t and s are the temporal and spatial dimensions of the input respectively. This enables us to perform a comprehensive study of the multi-modal and multi-input capabilities of the system. We further elaborate on the properties of these tasks below.

For training, we always use cross-entropy loss with Adam optimizer (Kingma & Ba, 2015) and schedule the learning rate using Noam scheduler (Shazeer & Stern, 2018) similar to (Vaswani et al., 2017) [2]. In the vision peripheral, we freeze the layers of pretrained ResNet model. Remaining parts of the architecture (peripherals and CNP) are kept trainable. In this section, we provide details on the datasets used and the model setup for each of these tasks.

Part-of-speech (POS)Tagging: To illustrate the task with only temporal modality ($\boldsymbol{t > 1}$ & $\boldsymbol{s = 1}$, where t and s are the temporal and spatial dimensions of the input respectively), we consider POS tagging problem. Given an input sequence of words, the model should produce a sequence of POS tags corresponding to each word. We use Penn Tree-bank[3] (Marcus et al., 1994) which contains gold annotations on English WSJ articles by experts. During the encoding stage, each input sentence is processed by the language peripheral to generate a spatio-temporal tensor $x \in \mathbb{R}^{t \times 1 \times d_{model}}$, where t is the sequence length of subwords in the input. The CNP $encode()$ function is then used to encode x into the temporal cache. Note that spatial cache is empty for text inputs as $s = 1$. Therefore, the decoding stage is same as that of Transformers to predict the sequence of POS tags.

Image Captioning: This task represents the ones with inputs containing only spatial modality ($\boldsymbol{t = 1}$ & $\boldsymbol{s > 1}$). The captioning model is required to predict text caption for a given image. We use the MSCOCO 2014 dataset (Lin et al., 2014) for training and present results on the COCO validation set. During the encoding stage, the input image is resized to 224×224 and processed by the vision peripheral containing pre-trained ResNet-152 to produce image embeddings $x \in \mathbb{R}^{1 \times 49 \times d_{model}}$. x is then input to the *encode()* function which populates corresponding spatial and temporal cache. The decoding stage uses *decode()* function with output vocabulary size 25000 to generate the captions.

Visual Question Answering: For the task with inputs from multiple domain, such that each contains either spatial or temporal modality (either $\boldsymbol{t > 1}$ & $\boldsymbol{s = 1}$, or $\boldsymbol{t = 1}$ & $\boldsymbol{s > 1}$ for each input), we choose the task of visual question answering. Given a question over an image as inputs, the model is supposed to predict the correct answer label. We use the recently introduced VQA v2.0 dataset (Goyal et al., 2017) for this purpose and perform evaluation on the VQA test-dev set. All the images are resized to dimension 224×224 before training. The encoding stage of this task utilizes two peripherals: the vision peripheral is used to generate a tensor $x_1 \in \mathbb{R}^{1 \times 49 \times d_{model}}$ for the input image. The language peripheral is used to encode the questions into $x_2 \in \mathbb{R}^{t \times 1 \times d_{model}}$, where t is equal to the length of the subwords in the question. The *encode()* function is the called two times, first with x_1 and second with x_2 as input. Finally, the $decode()$ with output vocabulary size 3500 is to generate the answers as softmax probabilities in a single decoding step.

Video Activity Recognition: For tasks which contain both spatial and temporal modality in a single input ($\boldsymbol{t > 1}$ & $\boldsymbol{s > 1}$), we consider the action recognition task on videos. For this purpose, we use the HMDB dataset (Kuehne et al., 2011). The dataset consists of over 5000 short length clips of real life actions with 51 classes. We present our results on train-test split 1. We use 16 frames per video and resize each of them to 224×224. During the encoding stage, each frame of the video is passed through the vision peripheral to cumulatively generate a video encoding $x \in \mathbb{R}^{16 \times 49 \times d_{model}}$ which is then used as input to the *encode()* function. Finally, the $decode()$ with output vocabulary size 51 is to predict the action as softmax probabilities in a single decoding step.

5 Results and Discussion

We present the evaluation on (a) Tasks of various modalities illustrated in Section 4 (b) Multi-tasking setup for these tasks (Table 1) (c) Reuse of the multi-task model for an unseen task (Figure 3). In addition, we also provide some ablation studies on the architecture (Table 2)

[2]The hyperparameter values used for n_h, d_{model}, N_Layers, d_k, d_v are same as that specified in Transformer base model (Vaswani et al., 2017).

[3]https://catalog.ldc.upenn.edu/LDC99T42; We use splits 0-18 as training, 19-21 as development and 22-24 as test sets

Performance of proposed architecture on individual tasks: We choose a set of four tasks with diverse input modalities and combinations as described in previous Section 4. We train the *OmniNet* model independently across each of the above tasks. Each of the tasks demonstrates unique capabilities of this generic architecture. More specifically, in Table 1 we compare our results with the following state-of-the-art[4]:- POS tagging: Spoustová et al. (2009); image captioning & VQA: Anderson et al. (2018) and HMDB: Simonyan & Zisserman (2014). It is important to note that we do not perform any hyper-parameter optimization. We believe that, with more computational power, fine tuning the hyperparameters towards these tasks should result in comparable or even improved performance to the state-of-the-art. These results can indeed be used as a baseline for any future work which aims at using a single architecture across various possible spatio-temporal archetypes. It is interesting to note that the model is extensible to a new domain without any modification to the CNP as long as one can add a specific peripheral to convert domain inputs into spatio-temporal tensors. This aspect of the architecture makes it applicable to several popular multi-modal tasks.

	POS	**Captioning**		**Visual Question Answering**				**HMDB**	
	Acc.	BLEU-4	Meteor	Overall	Y/N	Num.	Other	Acc.	**#PARAMS**
SOTA	97.44	36.2	27.0	63.2	80.3	42.8	55.8	59.4	-
IND	95.61	28.9	25.2	55.31	74.09	35.17	46.35	55.29	$450m$
MULT-3	95.82	28.8	25.2	56.79	76.75	35.82	47.16	-	$149.03m$
MULT-4	95.44	27.4	24.5	55.76	75.49	35.64	46.08	54.44	$149.07m$

Table 1: Performance of *OmniNet* on diverse set of tasks. IND: Model trained individually for each of the given tasks; MULT-3: Multi-task model trained on POS, Captioning & VQA; MULT-4: Multi-task model trained across all the four tasks.

Effect of training a diverse set of tasks together: We trained two multi-task models: (1) MULT-3 (POS+VQA+Captioning) and (2) MULT-4 (POS+VQA+Captioning+HMDB), using hogwild approach. While the MULT-3 model attains similar and sometimes better performance, the final MULT-4 model attains slightly reduced performance, when compared to the independent task scores. We believe this is due to the skewness in the size of the HMDB dataset containing only 5000 training samples. However as a tradeoff, adding the HMDB task shows interesting zero-shot results demonstrated below. Using a multi-task model also results in three times reduction in the total number of parameters. That is, when a separate model is used for each task, we have a total of over 450×10^6 parameters. Whereas during multi-tasking since a single model is shared, we have a total of over 149×10^6 parameters, while achieving similar performance. Interestingly, the model is able to attend on spatio-temporal components of the inputs from different tasks and concurrently generate predictions across them, thus demonstrating the generalization capability of our architecture.

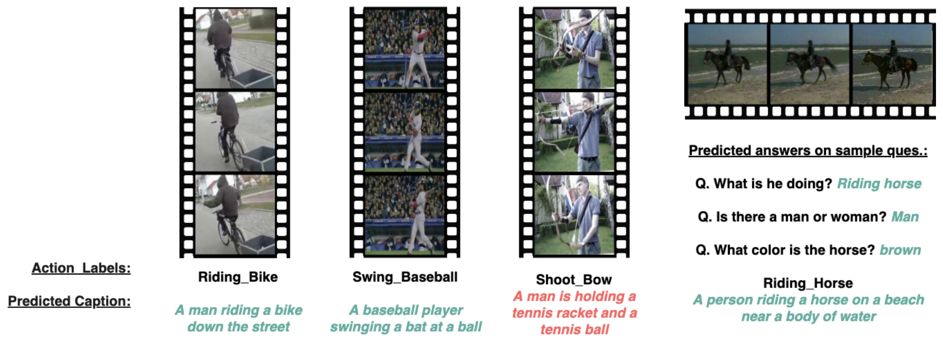
Predicted answers on sample ques.:
Q. What is he doing? Riding horse
Q. Is there a man or woman? Man
Q. What color is the horse? brown
Action_Labels:
Predicted Caption:
Riding_Bike
A man riding a bike down the street
Swing_Baseball
A baseball player swinging a bat at a ball
Shoot_Bow
A man is holding a tennis racket and a tennis ball
Riding_Horse
A person riding a horse on a beach near a body of water

Figure 3: Results of zero-shot video captioning and video question-answering.

Towards zero-shot learning: reuse of pre-trained network for unseen task: Sharing representations across multiple tasks provides the benefit to transfer of useful knowledge across multiple domains. Since, image and video are processed by the same vision peripheral, we conducted an experiment to see whether our model pre-trained on all the four tasks (MULT-4) can perform video

[4]Since most of these tasks are popular challenges, we compare with state-of-the-art which are generically applicable for the respective task instead of the challenge dataset.

captioning and video question-answering without any explicit training on these tasks i.e. zero-shot learning. The results of the evaluation on randomly picked instances from the HMDB test split 1 are shown in Figure 3. Interestingly, the model performs quite well on related actions that were present in the COCO and VQA training set; such as captions related to horse riding and baseball; or questions related to concepts present in VQA. Without training on any video captioning & video QA instance, the model could use the trained information from image captioning, image QA (VQA) and video action recognition (HMDB) apply them on videos to generate meaningful predictions, hence demonstrating the capability of the model to transfer knowledge across related multi-modal tasks. However, on concepts that are not present in the trained datasets, the model either describes the environment in the video or replaces with alternate known concepts. This case study, although not comprehensive, shows the capability of the model to learn shared representations and ability to transfer knowledge across domains. We believe that adding more tasks and domains will lead to more interesting zero-shot learning results in future across a wide range of problems.

Impact of individual architectural components: In order to support different input domains, our architecture introduces spatial cache and link array components to the original *Transformer* architecture (which only consists of mechanisms to handle temporal data). We conducted an ablation study on each of these components to verify their importance across various tasks as shown in Table 2. The ablation was conducted on the independent (IND) as well the multi-tasking model (MULT-4). The second row ablates the link array from our architecture i.e. removing the multiplication of G in Equation 1. The link array was designed to assist in tasks with inputs such as video, containing both spatial as well as temporal modality in a single input. The total number of spatial components becomes very large as number of frames in the video increases, thereby making it difficult to attend on various spatial regions throughout the video. Using link array the spatial attention layer can attend more on specific important frames in the video. Therefore, removal of link array leads to a huge reduction in performance in HMDB compared to other tasks as they do not have both spatio-temporal modalities for any single input. Removal of spatial cache, on the other hand, has significant effect on performance across all tasks containing spatial modality. Since, image captioning contains primarily spatial modality and hence the BLEU drops significantly after ablation. As other tasks utilize the temporal cache for prediction, in the multi-task setting the captioning task learns to utilize the spatial average of the image stored in the temporal cache for prediction and hence retains some performance after ablation of the spatial cache. On the other hand when trained independently on captioning, the network learns to utilize the information in the spatial cache only, and hence the score drops to zero after ablation. VQA leverages both, spatial information from image and temporal information from question, retains some performance from the use of temporal cache. Note that, POS tagging task is not affected by ablation of any of the components since it only has temporal modality in the input.

	POS (Acc.)		**Captioning** (BLEU-4)		**VQA** (Overall)		**HMDB** (Acc.)	
	IND	MULT-4	IND	MULT-4	IND	MULT-4	IND	MULT-4
OmniNet	95.61	95.44	28.9	27.4	55.31	55.76	55.29	54.44
w/o link array	95.61	95.44	28.9	27.4	54.05	55.30	45.94	46.79
w/o spatial cache	95.61	95.44	0	11.9	39.86	44.24	10.91	11.50

Table 2: Ablation study on the effect of proposed architectural components.

6 Conclusions and Future Work

We present a unified neural network architecture *OmniNet* capable of learning tasks with multiple inputs of varying modalities. The architecture can be further adopted for multi-task learning across any set of tasks containing spatio-temporal data. Sharing one model across multiple tasks also results in a significant reduction in the total number of parameters. We further demonstrate that this shared model can learn robust representations from various spatio-temporal inputs which are reusable for unseen tasks. We believe that this proposed architecture has wide applicability to any task with spatio-temporal inputs. To extend its usability, we would like to introduce new peripherals supporting more domains such as speech. We are also keen on exploring other aspects to the data beyond temporal and spatial dimensions such as graphs and relational data. Further, it would be interesting to investigate scheduling mechanisms for optimizing the multi-tasking framework.

REFERENCES

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6077–6086, June 2018. doi: 10.1109/CVPR.2018.00636.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, David Seetapun, Anuroop Sriram, and Zhenyao Zhu. Exploring neural transducers for end-to-end speech recognition. *CoRR*, abs/1707.07413, 2017. URL http://arxiv.org/abs/1707.07413.

Yaran Chen, Dongbin Zhao, Le Lv, and Qichao Zhang. Multi-task learning for dangerous object detection in autonomous driving. *Information Sciences*, 432:559 – 571, 2018. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2017.08.035. URL http://www.sciencedirect.com/science/article/pii/S0020025517308848.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL http://doi.acm.org/10.1145/1390156.1390177.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *ACL*, 2015.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Benjamin Heinzerling and Michael Strube. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

Lanqing Hu, Meina Kan, Shiguang Shan, and Xilin Chen. Duplex generative adversarial network for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017. doi: 10.1162/tacl_a_00065. URL https://www.aclweb.org/anthology/Q17-1024.

Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017. URL `http://arxiv.org/abs/1706.05137`.

Jin-Hwa Kim, Sang-Woo Lee, Donghyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Multimodal residual learning for visual qa. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 361–369. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6446-multimodal-residual-learning-for-visual-qa.pdf`.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 1564–1574. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7429-bilinear-attention-networks.pdf`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Kalpesh Krishna, Shubham Toshniwal, and Karen Livescu. Hierarchical multitask learning for ctc-based speech recognition. *CoRR*, abs/1807.06234, 2018. URL `http://arxiv.org/abs/1807.06234`.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. TVQA: localized, compositional video question answering. *CoRR*, abs/1809.01696, 2018. URL `http://arxiv.org/abs/1809.01696`.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. In *International Conference on Learning Representations*, 2016.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pp. 114–119, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3. doi: 10.3115/1075812.1075835. URL `https://doi.org/10.3115/1075812.1075835`.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1928–1937. JMLR.org, 2016. URL `http://dl.acm.org/citation.cfm?id=3045390.3045594`.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 689–696, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL `http://dl.acm.org/citation.cfm?id=3104482.3104569`.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 24*, pp. 693–701. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf.

M. L. Seltzer and J. Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6965–6969, May 2013. doi: 10.1109/ICASSP.2013.6639012.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235, 2018. URL http://arxiv.org/abs/1804.04235.

Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *CoRR*, abs/1802.08735, 2018.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 568–576. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 763–771, Athens, Greece, March 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E09-1087.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL http://arxiv.org/abs/1602.07261.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *In ECCV. 94108*, 2014.

Wei Zhao, Benyou Wang, Jianbo Ye, Min Yang, Zhou Zhao, Ruotian Luo, and Yu Qiao. A multi-task learning approach for image captioning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 1205–1211. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/168. URL `https://doi.org/10.24963/ijcai.2018/168`.