OmniVec: Learning robust representations with cross modal sharing

Siddharth Srivastava, Gaurav Sharma
TensorTour Inc.
{siddharth, gaurav}@tensortour.com

Abstract

Many applied machine learning methods aim to extract useful representations from data. However, a majority of such methods are modality and task specific. Building methods that can work with multiple modalities is a relatively recent research direction [25, 36, 37, 46, 62, 105]. Learning tasks together with a unified network can lead to regularization effects, as a large amounts of shared parameters are trained to perform varied tasks, and hence are more likely to extract meaningful representations from data without overfitting to one task or modality. It can also aid in utilizing available labelled data from different domains, hence potentially eliminating the cost and effort of labelling large amounts of data in a specific modality for a specific task. With the ability of sharing knowledge from multiple modalities (e.g. image, video, depth map and speech) from different domains (e.g. visual, acoustic, textual), the modality agnostic learning frameworks have been shown to provide better robustness [1, 28] to traditional unimodal networks. We contribute to that line of work, and develop a framework that can learn embeddings in a shared space from different modalities and also deliver high generalization performance. Specifically, we propose to learn embeddings from distinct modalities with modality specific encoders, and process them with a shared transformer backbone. The transformer backbone maps the input embeddings to a shared embeddings space. The network is then trained in an end-to-end manner.

Prior works towards generalized modality agnostic learning can be categorized into following three approaches. (i) Methods which directly take multiple heterogeneous modalities (image, 3D, audio) as input, without separate encoders for each modality, and directly learn representations from them [36, 37]. (ii) Methods that take representations from modality specific encoder as input and learn generalized modality specific embeddings using a common objective in the latent space [7], and, (iii) Methods which aim at sharing knowledge among different modalities by keeping either a common encoder [25] or separate encoders [1]. The first two approaches generally target modality agnostic input representation, which lend them capability to keep the network definition same for different modalities. However, such networks, in general, can be trained on one modality at a time, and hence do not facilitate cross modal knowledge sharing. On the other hand, the third approach facilitates jointly training networks on multiple modalities. Our work is closer to the third set of approaches. Specifically, similar to [7], the proposed method employs different encoder for each modality. Similar to [25] we share knowledge among modalities, and train on multiple modalities sequentially allowing embeddings to generalize across modalities. Unlike [25], we do not limit our method to a specific subset of modalities, and train on multiple modalities in a sequential manner. Further, we do not assume any correspondence between the training data i.e. paired training sets across modalities, which is different from previous works, e.g. [1], where correspondence in data among modalities is assumed.

1. Introduction

Many applied machine learning methods aim to extract useful representations from data. However, a majority of such methods are modality and task specific. Building methods that can work with multiple modalities is a relatively recent research direction [25, 36, 37, 46, 62, 105]. Learning tasks together with a unified network can lead to regularization effects, as a large amounts of shared parameters are trained to perform varied tasks, and hence are more likely to extract meaningful representations from data without overfitting to one task or modality. It can also aid in utilizing available labelled data from different domains, hence potentially eliminating the cost and effort of labelling large amounts of data in a specific modality for a specific task. With the ability of sharing knowledge from multiple modalities (e.g. image, video, depth map and speech) from different domains (e.g. visual, acoustic, textual), the modality agnostic learning frameworks have been shown to provide better robustness [1, 28] to traditional unimodal networks. We contribute to that line of work, and develop a framework that can learn embeddings in a shared space from different modalities and also deliver high generalization performance. Specifically, we propose to learn embeddings from distinct modalities with modality specific encoders, and process them with a shared transformer backbone. The transformer backbone maps the input embeddings to a shared embeddings space. The network is then trained in an end-to-end manner.

Prior works towards generalized modality agnostic learning can be categorized into following three approaches. (i) Methods which directly take multiple heterogeneous modalities (image, 3D, audio) as input, without separate encoders for each modality, and directly learn representations from them [36, 37]. (ii) Methods that take representations from modality specific encoder as input and learn generalized modality specific embeddings using a common objective in the latent space [7], and, (iii) Methods which aim at sharing knowledge among different modalities by keeping either a common encoder [25] or separate encoders [1]. The first two approaches generally target modality agnostic input representation, which lend them capability to keep the network definition same for different modalities. However, such networks, in general, can be trained on one modality at a time, and hence do not facilitate cross modal knowledge sharing. On the other hand, the third approach facilitates jointly training networks on multiple modalities. Our work is closer to the third set of approaches. Specifically, similar to [7], the proposed method employs different encoder for each modality. Similar to [25] we share knowledge among modalities, and train on multiple modalities sequentially allowing embeddings to generalize across modalities. Unlike [25], we do not limit our method to a specific subset of modalities, and train on multiple modalities in a sequential manner. Further, we do not assume any correspondence between the training data i.e. paired training sets across modalities, which is different from previous works, e.g. [1], where correspondence in data among modalities is assumed.
Our proposed framework, OmniVec, consists of the following components: (i) a modality specific encoder, (ii) a shared backbone network, and (iii) task specific heads where tasks can be any machine learning task. The framework facilitates end-to-end training. In simple terms, OmniVec works as follows. For a given task and a modality we select a modality compatible encoder and an appropriate task head. We attach the encoder and task heads to, the beginning and end of the shared backbone network respectively. Then to train on another modality, we replace the encoder while keeping the backbone same. If the task is to be changed as well, we replace the task head. To further facilitate learning of better representations and cross-modal information sharing, we train the network numerous tasks. We borrow the motivation from earlier works where it has been shown that training networks on multiple related tasks can provide better generalization [103]. Similar improvements, in generalization, have been reported for multimodal multi-task learning as well [16, 34, 59, 69]. However, we do not train in a traditional multi-task setting, where all tasks are available together and are trained for together. Instead, we train the network in a sequential manner, i.e. we train on different tasks, one after another.

Motivated by empirical observations and previous works indicating that robustness of multi-task mechanisms depends on the complexity of tasks selected for joint training [59, 71], we propose to group the tasks based on the extent of information exploited by the task across different modalities, e.g., a semantic segmentation task forces the network to embed more local information in the learned representation, as compared to a classification task [15]. In addition to grouping the tasks, we also construct training data by mixing samples from each modality for a particular task. We train the network by replacing encoder modulus of each task while keeping the backbone same. Based on earlier works indicating that self-supervised pretraining helps networks in better exploiting multiple modalities [16, 24], we pretrain the network with masked pretraining.

In summary, we make the following contributions. (i) We propose a novel method to learn embeddings from many modalities. The method has a common backbone to process the different modalities and perform different tasks. Specifically, we show that the proposed method works with RGB images and videos, depth images, point clouds, audio, speech and text data. (ii) We propose a novel training mechanism to allow learning using multiple tasks from both spatial (e.g. image, 3D point clouds, depth maps) and temporal (e.g. video, audio, speech, text) data. Owing to the common backbone of the method, and a synchronous training mechanism, the method shares knowledge between different modalities and tasks, resulting in improved performance and generalization. (iii) The proposed method allows for infusing cross domain information in the feature vectors, i.e. allowing embeddings from text data to be close to similar data in image domain. (iv) We propose an iterative training mechanism by mixing modalities and grouping tasks. Different from earlier works, we also propose to perform self supervised masked pretraining across visual as well as non visual modalities. (v) With exhaustive experiments on numerous popular benchmarks across, we show that the proposed framework achieves state-of-the-art results or performs close to the competing methods. (vi) We also study the generalization ability of the proposed framework by demonstrating the robust performance of the learned embeddings on unseen tasks. (vii) We conduct an extensive ablation study to demonstrate the impact of the design choices.

2. Related Works

In this section, we discuss similar works and various similar paradigms to our work. We begin with transformers, which are basis of our work, and then move to methods which work with multiple modalities. Among methods that work with multiple modalities, many of them work on utilizing the modalities simultaneously, while others propose networks which take the modalities as input, one at a time. Transformers. Transformers were proposed originally for Natural Language Processing tasks [78]. The main contribution of this work was to demonstrate the effectiveness of multi-head attention in representing long-range correlation between words. Owing to the popularity of transformers in NLP tasks [49], attempts were made to extend it to vision tasks. Early work in this direction [14, 53, 93] involved utilizing features from convolutional neural networks. However, with vision transformers [19], transformers obtained an ability to process raw images and achieved performance competitive to CNNs. After that, transformers have dominated nearly all the vision related tasks [41]. As transformers have demonstrated robust performance across modalities, recent methods across various modalities use them to solve various tasks [21, 49, 63, 64, 91].

Multi-modal methods. Majority of the current multimodal methods use modality specific feature encoders [2, 38, 39, 62, 88] and are hence concerned with methods of feature fusion with their proposed architectures. In general the networks for different modalities differ from each other and can not be easily used together without architectural modifications. They also need to decide on when to fuse the features from various modalities, when to fine-tune, how to pre-train etc. [90]. Such problems inhibits extending networks such as transformers to be applied as a common backbone across multiple domains such as point clouds, audio and images.

Common network for multiple modalities. Recently, many methods have been proposed which learn from multi-
ple modalities [7, 8, 25, 37]. Among the most popular, however recent, are methods that do not have separate encoders for each modality. Such methods generally transform the input raw data to a common input representation prior to generally being processed by a transformer network. Among them, the perceiver and similar methods [8, 36, 37] have tried to learn from multiple modalities together without separate encoders. Perceiver architecture works by cross-attention among a set of latent queries. Similarly, hierarchical perceiver [8] builds upon it proposes to group the input array while preserving the locality structure. On the other hand, methods such as data2vec [7] use modality specific encoders. Other methods such as Omnivore [25] have a common encoder. However, Omnivore is limited to only visual modalities (image, depth map, video). Then, methods such as VATT [1] have a common backbone for each text, image and audio. However, it processes each modality independently using a transformer. Such methods which learn from multiple modalities have been shown to provide better robustness [1, 28]. Our methods largely overlaps with the from multiple modalities have been shown to provide better visual modalities (image, depth map, video). Then, methods such as VATT [1] have a common backbone for each text, image and audio. However, it processes each modality independently using a transformer. Such methods which learn from multiple modalities have been shown to provide better robustness [1, 28]. Our methods largely overlaps with the methods largely overlaps with the motivation of such methods, however, it differs from such methods in that earlier methods operate on training for one task or one modality at a time, while we learn by training on multiple modalities and multiple tasks while using a single common backbone architecture.

Multi-task learning. We have discussed many methods that attempt at learning from multiple inputs. As discussed in the previous section, recent years have seen many methods that work with multiple modalities. PerceiverIO [36] extends Perceiver [37] and enables learning multiple tasks using the same network architecture. While PerceiverIO can also learn multiple tasks at a time using a single architecture, generally multiple networks are used [102]. Many techniques [7, 16, 25, 34, 59] learn from multiple modalities and from their raw representation and apply to multiple tasks.

Multi-modal masked pretraining. Methods such as [52, 84, 92] use masked pre-training. Masked pretraining has shown to improve the performance of deep networks networks for various modalities and tasks [1, 6, 7, 24, 31, 98] and motivated by such works we also use masked pre-training as a self supervised step leveraging large amounts of data available. However, different from earlier works, we perform masked pre-training on multiple modalities and multiple datasets on the same common backbone.

3. Approach

We now describe our framework for learning multiple tasks in multiple modalities with a common backbone network, allowing for cross modality knowledge sharing. The overview of the proposed framework is shown in Figure 1. The network comprises six building blocks, i.e. modality encoders, meta token block, projection block, transformer, vectorizer and task heads. We now explain each block in detail.

3.1. OmniVec Framework

Modality Encoder. The modality encoder takes as input, one modality at a time and extracts feature embedding for each of the modalities. In the proposed framework, the modality encoder can be a transformer, convolutional neural network or can directly use raw signals [1]. As we do not assume any specific structure for the modality encoder, the proposed framework allows incorporating any appropriate deep network as a modality encoder.

For current work, we use domain specific transformer based encoders for each of the modalities as shown in Table 1 followed by a common backbone network. It is worth noting that each of the networks in visual and auditory domain is based on Vision Transformer architecture i.e. image and depth directly use ViT, video (ViViT) differs from ViT in input tokenization that extends 2D patches to 3D (spatio-temporal mapping), audio (AST) transformers differ from ViT only in input representation i.e. uses log-mel spectrograms instead of images. Simple3D-former for point cloud uses a 2D ViT transformer as the base network with modified positional embeddings and tokenization approach. We use a standard BERT transformer for textual data. We train each of these models from scratch.

Meta Tokens. We extract meta tokens from the input modalities. This meta representation is a vector that encodes the type of modality (I), size of temporal dimension (T), height (H), width (W) in spatial dimension, number of channels (C) and length or number of tokens (L). In general, the meta tokens can also hold additional information to make the framework adapt to additional modalities. The value in each of these representation variables is conditioned on the type of modality e.g. non spatial data may have H, W only with the other non-spatial parameter set as a special token, denoting lack of such information.

Projection Layer. The projection layer inputs the intermediate representations from the modality encoder network and is conditioned on the meta tokens. It then converts the input representation to patches that are provided as input to the subsequent transformer network. We obtain n-dimensional vector for each patch by applying linear pro-
projection. Similar to ViT [19], this projection is applied with a learnable weight $W_{ip} \in \mathbb{R}^{t \times h \times w \times 3 \times 1}$ for each modality $i$. The meta tokens make the projection layer adaptable to varying number and dimensions of input patches and generate latent representations compatible with the subsequent transformer network. For instance, we represent RGB images as $I \in \mathbb{R}^{1 \times h \times w \times 3 \times 1}$ with $t=1$ frames and $c=3$ channels. Similarly, we represent video as $V \in \mathbb{R}^{t \times h \times w \times 3 \times 1}$ with $t$ frames ($t > 1$) and $c=3$ channels, depth as $D \in \mathbb{R}^{1 \times h \times w \times 4 \times 1}$ with $c=4$ channels, point cloud as $P \in \mathbb{R}^{1 \times 1 \times 1 \times 3 \times l}$ with $l$ points, audio as $A \in \mathbb{R}^{t \times h \times w \times 1}$ with spectrogram input, and text as $L \in \mathbb{R}^{1 \times 1 \times 1 \times l}$ with $l$ tokens. Each patch $x$ is processed independently and projected to an embedding $e$ followed by a LayerNorm [5].

**Transformer.** The transformer network is the common part of the framework and is in effect a ‘bottleneck’ block. While different modalities may arrive here through different encoders, they all have to pass through this transformer network. The transformer network inputs the patches generated by the projection layer and outputs features. While the OmniVec framework can use any standard transformer architecture, we use [18] as our backbone architecture. In our transformer network, the multi head attention involves standard self-attention [78], and GeLU [33] activation prior to the MLP layer.

**Vectorizer.** The vectorizer layer takes patches from the transformer network as input, and outputs embeddings for the original data point. It outputs a single embedding $e = f(X)$ for an input $X$. We name the output embeddings of the vectorizer as Omni Embeddings, as these embeddings constitute knowledge from multiple tasks and modalities due to forward pass from the transformer block where cross modality and cross task information is infused.

For our implementation, we concatenate the output patches and pass them through a linear layer to obtain a $d$-dimensional embedding. At the time of training, we use the outcome of vectorizer as input to subsequent task heads. However, using the outcome of vectorizer as input to task heads is optional as the task head may also directly take input patches from the previous transformer bottleneck. Once the model has been trained, the output from vectorizer can be used for fine-tuning and evaluation on downstream tasks.

**Task Heads** The final parts of network, the task heads are $\sum_{i \in \mathbb{N}} T_{ih}$ independent networks which learn $h$ for every $i$th modality. The task heads can generally be any computer vision, natural language processing or other modality specific task. We experiment with classification (image, video, audio, text), segmentation (image, point clouds) etc. We describe them in Section 4.

### 3.2. Training OmniVec Framework

We train the OmniVec Framework in two stages. First we perform masked pretraining. Then we fine tune the network on multiple modalities. Both these stages are described below.

**Masked Pretraining.** We pretrain the network with masked autoencoders [1, 24]. Specifically, for an input with $N$ patches, we mask $K$ patches, and feed non-masked patches and their positions to the encoder. For each modality, we use the encoder from Table 1 followed by our bottleneck transformer that outputs per patch embeddings i.e. we keep a shared bottleneck transformer encoder for each of the modalities. Similar to [1, 32], the per patch embeddings are concatenated with $K$ replicas of learnable mask tokens resulting in $N$ embeddings. We add corresponding positional embeddings to each of the $N$ embeddings, and pass to the decoder. We use the same masking strategy for modalities from visual and auditory domains. For textual data, we follow [66] and randomly permute the sentences [95] and use a small fraction $f$ of tokens as predicted tokens, followed by utilizing 8:1:1 strategy of BERT [18] for constructing mask tokens. The training objective is to minimize the reconstruction error between the input and decoder outputs. For image, video, point clouds and audio spectrogram input, we minimize $l_2$ distance between the $K$ predicted and target patches. For visual inputs, the input samples are normalized to zero mean and unit variance. For textual data, we use the permuted language modelling of XLNet [95] as
the objective. We refer this strategy of constructing mini-batches from each dataset separately. However, a segmentation or depth prediction task, requires each pixel to be predicted, and hence we refer it as a dense task. We detail each of the tasks, the datasets used to train them and their task grouping in Section 4.

As we do not assume any correspondence between data from various modalities, we propose mixing samples from all datasets for a particular task to share knowledge between various modalities. An alternative approach would be to construct mini-batches from each dataset separately. However, we found it performs poorly compared to mixing samples from modalities. We refer this strategy of constructing mini-batches as modality mixing. Specifically, for a particular task \( h \) belonging to a type of task \( t \) (simple, dense), for each modality \( i \), we extract sample \( s_{i,h,t} \) from the datasets. After task grouping and modality mixing, we train the network in an end-to-end manner iteratively for simple and dense tasks. Specifically, we train the network for \( E \) epochs, we train the network for \( v_1 \) epochs with mini-batches from simple tasks and \( v_2 \) for dense tasks. We continue training the network in iterative manner i.e. switching between simple and dense tasks for \( E \) epochs.

### 4. Experiments

**Masked pretraining.** We do masked pretraining using the modality mixing as described in Section 3. We use AudioSet (audio) [23], Something-Something v2 (SSv2) (video) [30], English Wikipedia (text), ImageNet1K (image) [17], SUN RGB-D (depth maps) [67], ModelNet40 (3D point cloud) [87] for pretraining the network. As we perform autoencoder based pre-training, we do not group the tasks, and instead uniformly sample data from each of the datasets and modalities. Further, we randomly select patches for masking. For image, video and audio, we randomly mask 90% of the patches. For point cloud, we mask 80% of the patches, and for text we mask 95% of the patches. Further, we keep \( f = 5\% \) of the tokens as predicted tokens (unlike 15% in [95]). We perform pretraining for 2000 epochs.

**Modality Encoder.** For modality specific encoders, we use the networks from Table 1. We use the same network configurations for these networks as in corresponding publications. We pretrain the model using masked pretraining as described in Section 3, followed by training on specific modalities as per task groups and modality mixing. For different tasks on a modality, we keep the modality encoder same, while changing the task heads with appropriate loss functions. We train modality encoders for \( E = 900 \) epochs with 2 consecutive epochs each for simple and dense task groups.

**Datasets for training on multiple modalities and tasks.** After masked pre-training, we fine tune the network on multiple tasks across modalities. The datasets and their corresponding task groups and modality are given in Table 7.

**Task Heads.** For classification tasks, we use standard classification head from ViT [19] while use [4] for video classification and [27] for audio classification. For image and point cloud segmentation tasks, we use the segmentation head from [61]. For text summarization, we use a 3-layered transformer.

We provide more implementation details in the supplementary material.

### 4.1. Results

**Comparison of pretrained OmniVec with similar methods.** Table 2 compares OmniVec model with masked pretraining to various similar methods. The table also indicates the modalities supported by various methods (Col.-Supp. Modalities), and that if the method supports sharing knowl-
edge between modalities (Col.-Cross-Modal sharing). Further, it also details the learning objectives by these methods. The table reports results on six benchmark datasets on seven tasks as AudioSet supports two tasks (audio only, and audio with video). These datasets are used to perform masked pretraining on the OmniVec model as described in Section 3. We discuss the comparison to state-of-the-art methods in three out of four metrics and come in second (v) Text Table 11 shows state of the art on DialogueSUM dataset for text summarization. OmniVec surpasses other methods in three out of four metrics and comes in second on the R-L metric. Despite utilizing significantly fewer datasets for text (only two) in comparison to visual tasks (ten datasets), OmniVec demonstrates strong performance. This suggests OmniVec’s capacity to bridge the modality gap [48] across distinct domains in the latent space, even when the data distribution is skewed.

4.2. Ablations

**Impact of task grouping and modality mixing.** Table 8 shows the effect of task grouping and modality mixing. We evaluate four network variations: (i) OmniVec-1 without either of task grouping and modality mixing, (ii) OmniVec-2 with just task grouping, (iii) OmniVec-3 with only modality mixing, and (iv) OmniVec-4 combining both. OmniVec-1 uses masked pretraining on single datasets. OmniVec-
2 groups tasks by modality, OmniVec-3 mixes modalities randomly, and OmniVec-4 follows the settings from Section 3. Comparatively, OmniVec-1 lags behind the others. Both OmniVec-2 and OmniVec-3 outperform OmniVec-1 by around 30% to 45%, showing their efficacy. However, OmniVec-4, which combines both approaches, performs better, emphasizing the benefits of integrating tasks and modalities.

**Influence of size of the modality encoder.** We evaluated the impact of enlarging the base modality encoder to the scale of our suggested network, using modality-specific data. This change slightly improved performance. For example, on ImageNet1K, the top-1 accuracy went from 88.5% with the base ViT [19] to 89.1% with the augmented ViT having a similar parameter count, while OmniVec achieved 92.4%. These findings suggest that even with enhancements, the augmented base modality encoder still lags significantly behind OmniVec, highlighting OmniVec’s advantage of leveraging information from multiple modalities.

**Fine-tuning with the same datasets after masked pretraining and comparison to state-of-the-art.** In Table 13, we show the results of fine-tuning the OmniVec-4 model on each of the datasets that was used for masked pretraining. As during masked pretraining, we use the standard train sets for each of these datasets for fine-tuning.

It can be observed from the results that OmniVec achieves better performance on each dataset than existing state of the art method. As we are using same backbone (OmniVec-4) for each of these datasets, it shows the robustness of the embeddings and the capacity of the network to adapt to different tasks and distribution of dataset.

### 4.3. Generalization Ability

**Generalization on unseen datasets.** We evaluate the performance of the learned embeddings on unseen datasets. Specifically, we show results on the tasks of fine grained image classification (Oxford-IIIT Pets [56]), Video Classification (UCF-101 [68], HMDB51 [43]), 3D point cloud classification (ScanObjectNN [76]), 3D point cloud segmentation (NYUv2 [65]) and text summarization (SamSum [26]). Our findings, tabulated in Table 12 [rows 1-6], demonstrates that even without fine-tuning, OmniVec surpasses most state-of-the-art methods. Further, while the pretrained OmniVec slightly underperformed on ScanObjectNN (92.1%) compared to PointGPT’s 93.4%, when fine-tuned, OmniVec achieved 96.1% accuracy, outperforming PointGPT. This shows OmniVec’s generalizability on datasets where it is exposed to analogous tasks.

**Generalization on unseen tasks - Monocular Depth Prediction on KITTI Depth Prediction Benchmark.** We fine tune the network for the task of depth prediction on KITTI Depth Prediction benchmark [75]. Our network has not seen such image to image style transfer tasks. The results on KITTI depth prediction benchmark are shown in Table 12 (row 7). We outperform the state of the method VA-DepthNet [50] i.e. 10.44 iRMSE on VA-DepthNet cf. 10.2 for OmniVec. As can be observed from Figure 2, the depth maps obtained by OmniVec are able to better capture the details near edges.

**Cross-domain generalization.** Following prior work [1], we evaluate on the task of zero-shot text-to-video retrieval. The results are reported in Table 12. On the YouCook2 dataset, our pretrained OmniVec surpasses the state of the art in zero-shot retrieval, achieving a Recall@10 of 64.2% compared to VideoCLIP’s 63.1%. On MSR-VTT, when compared with SM [99], our fine-tuned OmniVec embeddings yield a Recall@10 of 89.4% against SM’s 90.8%. With just pretraining, SM has a Recall@10 of 80%, slightly above our 78.6%. SM utilizes large-scale pretraining on internet scale data, while OmniVec uses much less data. Further, the second-best MSR-VTT method [11] achieves only

<table>
<thead>
<tr>
<th>Method</th>
<th>Task Grouping</th>
<th>Modality Mixing</th>
<th>AudioSet (A+V)</th>
<th>AudioSet (A)</th>
<th>SSV2</th>
<th>GLUE</th>
<th>ImageNet1K</th>
<th>Sun RGBD</th>
<th>ModelNet40</th>
</tr>
</thead>
<tbody>
<tr>
<td>OmniVec-1 (baseline)</td>
<td>✓</td>
<td>✓</td>
<td>37.5</td>
<td>36.3</td>
<td>62.6</td>
<td>57.5</td>
<td>70.2</td>
<td>79.8</td>
<td>66.4</td>
</tr>
<tr>
<td>OmniVec-2</td>
<td>✓</td>
<td>✗</td>
<td>42.6</td>
<td>40.1</td>
<td>73.5</td>
<td>69.5</td>
<td>79.8</td>
<td>66.4</td>
<td>75.2</td>
</tr>
<tr>
<td>OmniVec-3</td>
<td>✗</td>
<td>✓</td>
<td>39.2</td>
<td>39.4</td>
<td>70.2</td>
<td>68.8</td>
<td>77.3</td>
<td>65.5</td>
<td>72.2</td>
</tr>
<tr>
<td>OmniVec-4</td>
<td>✓</td>
<td>✓</td>
<td>48.6</td>
<td>44.7</td>
<td>80.1</td>
<td>84.3</td>
<td>88.6</td>
<td>71.4</td>
<td>83.6</td>
</tr>
</tbody>
</table>

Table 8. Impact of various training strategies on OmniVec. We report results with and without each of task grouping and modality mixing. The results are reported with masked pretraining only. We observe that individually, both task grouping and modality mixing improve the results over the baseline method. However, there combination outperforms individual performance using these mechanisms.

<table>
<thead>
<tr>
<th>Method</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
<th>B-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODS [93]</td>
<td>44.27</td>
<td>17.90</td>
<td>36.98</td>
<td>70.49</td>
</tr>
<tr>
<td>SICK [42]</td>
<td>46.4</td>
<td>20.39</td>
<td>40.83</td>
<td>71.32</td>
</tr>
<tr>
<td>OmniVec</td>
<td>46.91</td>
<td>21.22</td>
<td>40.19</td>
<td>71.91</td>
</tr>
</tbody>
</table>

Table 11. DialogueSUM text summarization ROGUE scores.

<table>
<thead>
<tr>
<th>Method</th>
<th>ModelNet40C</th>
<th>ModelNet40C</th>
<th>ModelNet40C</th>
<th>ModelNet40C</th>
<th>ModelNet40C</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet++ [60]</td>
<td>0.236</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGCN+PCM-R [100]</td>
<td>0.173</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCT + PCM-R [45]</td>
<td>0.173</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCT + PCM-R [72]</td>
<td>0.163</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OmniVec</td>
<td>0.156</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9. ModelNet40-C Error Rate.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>SDDIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointTransformer+CBL [73]</td>
<td>71.6</td>
<td></td>
</tr>
<tr>
<td>StratifiedTransformer [44]</td>
<td>72.0</td>
<td></td>
</tr>
<tr>
<td>PTv2 [86]</td>
<td>72.6</td>
<td></td>
</tr>
<tr>
<td>Swin3D [91]</td>
<td>74.5</td>
<td></td>
</tr>
<tr>
<td>OmniVec</td>
<td>75.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 10. Stanford Indoor Dataset mIoU.
Table 12. Generalization performance of OmniVec on unseen datasets (Oxford-IIIT Pets, UCF-101, HMDB51, ScanObjectNN, NYUv2 Seg, SamSum), unseen tasks (KITTI Depth Prediction) and cross-domain generalization (YouCook2, MSR-VTT). Pre. indicates network with pretraining only, FT indicates network finetuned on training set of respective datasets. See supplementary for more detailed results.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Modality</th>
<th>Task</th>
<th>Metric</th>
<th>OmniVec (Pre.)</th>
<th>OmniVec (FT.)</th>
<th>SOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF-101</td>
<td>Video</td>
<td>Action Recognition</td>
<td>3-Fold Accuracy</td>
<td>98.1</td>
<td>99.6</td>
<td>99.6 (VideoMAE V2-g [79])</td>
</tr>
<tr>
<td>HMDB51</td>
<td>Video</td>
<td>Action Recognition</td>
<td>3-Fold Accuracy</td>
<td>89.31</td>
<td>91.6</td>
<td>88.1 (VideoMAE V2-g [79])</td>
</tr>
<tr>
<td>Oxford-IIIT Pets</td>
<td>Image</td>
<td>Fine grained classification</td>
<td>Top-1 Accuracy</td>
<td>97.4</td>
<td>99.2</td>
<td>97.1 (EffNet-L2 [20])</td>
</tr>
<tr>
<td>ScanObjectNN</td>
<td>3D Point Cloud</td>
<td>Classification</td>
<td>Accuracy</td>
<td>92.1</td>
<td>96.1</td>
<td>93.4 (PointGPT [8])</td>
</tr>
<tr>
<td>NYU V2</td>
<td>RGBD</td>
<td>Semantic Segmentation</td>
<td>Mean IoU</td>
<td>58.6</td>
<td>60.8</td>
<td>56.9 (CMN [51])</td>
</tr>
<tr>
<td>SamSum</td>
<td>Text</td>
<td>Meeting Summarization</td>
<td>ROUGE(R-L)</td>
<td>51.2</td>
<td>54.6</td>
<td>50.88 (MoCa [101])</td>
</tr>
<tr>
<td>KITTI</td>
<td>RGB</td>
<td>Depth Prediction</td>
<td>iRMSE</td>
<td>—</td>
<td>10.2</td>
<td>10.4 (VA-DepthNet [50])</td>
</tr>
<tr>
<td>YouCook2</td>
<td>Video+Text</td>
<td>Zero Shot Text-to-Video Retrieval</td>
<td>Recall@10</td>
<td>64.2</td>
<td>70.8</td>
<td>63.1 (VideoCLIP [89])</td>
</tr>
<tr>
<td>MSR-VTT</td>
<td>Video+Text</td>
<td>Zero Shot Text-to-Video retrieval</td>
<td>Recall@10</td>
<td>78.6</td>
<td>89.4</td>
<td>80.0(Pre.)90.8(FT)/SM [99]</td>
</tr>
</tbody>
</table>

Table 13. Comparison with state of the art after fine tuning on respective training sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>OmniVec</th>
<th>SOTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioSet(A)</td>
<td>mAP</td>
<td>54.8</td>
<td>53.3 (MAVL [35])</td>
</tr>
<tr>
<td>AudioSet(A+V)</td>
<td>mAP</td>
<td>55.2</td>
<td>51.2 (CAV-MAE [29])</td>
</tr>
<tr>
<td>SSv2</td>
<td>Top-1 Acc</td>
<td>85.4</td>
<td>77.3 (MVD [80])</td>
</tr>
<tr>
<td>ImageNet1K</td>
<td>Top-1 Acc</td>
<td>92.4</td>
<td>91.1 (BASIC-L [12])</td>
</tr>
<tr>
<td>Sun RGBD</td>
<td>Top-1 Acc</td>
<td>74.6</td>
<td>67.2 (Omnivore [25])</td>
</tr>
<tr>
<td>ModelNet40</td>
<td>Overall Acc</td>
<td>96.6</td>
<td>95.4 (GeomGCNN [70])</td>
</tr>
</tbody>
</table>

Figure 2. Qualitative results on test set of KITTI Depth Prediction. Ground truth is not available. For an RGB input image (left), the outputs from VA-DepthNet [50](middle) and OmniVec (right) are shown. See supplementary material for more qual. results.

73.9% Recall@10 (see supplementary), which is behind our pretrained OmniVec.

5. Conclusion and Limitations

Conclusion. We proposed OmniVec, a unified data and task agnostic learning framework with a single backbone. The main idea behind OmniVec is that modalities in different domains can aid learning process. Further, we also proposed a novel training mechanism by grouping tasks and constructing mini batches by mixing inter-modality datasets. With experiments on 22 datasets spanning across image, video, point cloud, depth, audio, text; we show that the proposed framework is highly generalizable along with being extremely robust. It can also generalize well to seen tasks with different data distribution as well as can adapt to unseen tasks effectively. We also studied the cross-domain knowledge sharing by evaluating a zero shot video-text retrieval task. We achieve state of the art or close to state of the art performance on all the evaluated datasets.

Limitations. OmniVec trains on unpaired multi-modal data, but paired data, though better, is expensive to obtain. The method employs multiple encoders per modality, increasing computational demands. Future research may address these computational challenges in unified networks.

Societal Impact. Modality agnostic techniques enhance realistic data cloning, risking misinformation and identity theft. These networks, syncing various modalities and using extensive internet data, amplify privacy, security, and bias concerns.
References


[77] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inertialist species classification and detection dataset. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8769–8778, 2018. 6
Advances in neural information processing systems, 30, 2017. 2, 4


[86] Xiaoyang Wu, Yixing Lao, Aidan G Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017. 2, 4


