Sparse Fusion for Multimodal Transformers

Yi Ding*† Alex Rich*† Mason Wang‡ Noah Stier†
Matthew Turk§ Pradeep Sen† Tobias Höllerer†

† University of California, Santa Barbara
‡ Saratoga High School
§ Toyota Technological Institute at Chicago

yding@ucsb.edu, anrich@ucsb.edu, mason.wang0025@gmail.com, noahstier@ucsb.edu,
mturk@ttic.edu, psen@ece.ucsb.edu, holl@ucsb.edu

Abstract

Multimodal classification is a core task in human-centric machine learning. We observe that information is highly complementary across modalities, thus unimodal information can be drastically sparsified prior to multimodal fusion without loss of accuracy. To this end, we present Sparse Fusion Transformers (SFT), a novel multimodal fusion method for transformers that performs comparably to existing state-of-the-art methods while having greatly reduced memory footprint and computation cost. Key to our idea is a sparse-pooling block that reduces unimodal token sets prior to cross-modality modeling. Evaluations are conducted on multiple multimodal benchmark datasets for a wide range of classification tasks. State-of-the-art performance is obtained on multiple benchmarks under similar experiment conditions, while reporting up to six-fold reduction in computational cost and memory requirements. Extensive ablation studies showcase our benefits of combining sparsification and multimodal learning over naive approaches. This paves the way for enabling multimodal learning on low-resource devices.

1. Introduction

We experience and interact with the world through our five senses: sight, sound, taste, touch, and smell. The human brain is incredibly good at processing all of this information, paying attention only to the few things that matter. Imbuing a computer with the ability to process multimodal data effectively is highly desirable because it would enable a vast array of multi-sensory applications. However, processing multiple data streams increases computational cost, and it is therefore a high priority to develop efficient algorithms in this domain. Additionally, many of these applications, such as the detection of instances of domestic abuse, or detection of prolonged emotional and psychological struggles, are particularly well-suited for mobile or low-resource devices. In these resource-constrained settings, the computation cost and memory footprint become critical factors that must be considered for practical use.

Current multimodal algorithms involve some level of modality-independent feature processing followed by a fusion process which then jointly models the dependencies and cross-dependencies between the modalities. In particular, deep-learning transformer models have been used in this way to achieve state-of-the-art performance on numerous tasks [15, 19]. However training and processing such data remains prohibitively expensive in many cases, in terms of time, computational resources, and energy consumption. For example, a single layer of a vision transformer [9] requires approximately 1.35 billion floating-point operations (GFlops) for a 224×224 image for a single forward pass. If we represent a sequence of 30 frames in a similar manner for video data, this explodes to 88.24 GFlops. Although recent advancements have been made to sparsify transformers, these efforts have primarily approached the problem from a unimodal perspective [1, 3, 17, 24, 28].

Motivated by these concerns, we propose a sparse fusion method for multimodal transformers called Sparse Fusion Transformers (SFTs) that drastically reduces training time and memory consumption while maintaining the quality of existing fusion methods. Our approach is based on the hypothesis that the large amount of complementary information across different modalities allows us to sparsify unimodal information prior to multimodal fusion without the loss of accuracy. In particular, approaching a problem from a multimodal perspective enables us to sparsify the unimodal information far more aggressively. With our sparse-fusion method, we achieve faster performance with

*Equal contribution

less memory use while attending to features that are most important.

Our proposed fusion process is agnostic to input modality and makes a full multimodal classification network robust to sparsification of input representations. It is composed of three parts: a block-sparse within-modality attention to learn strong local representations, a pooling method for extracting them, and dense self-attention for cross-modal feature fusion. Furthermore, we propose to use a customized mixup to apply spatio-temporal regularization to the learned representations in a modality agnostic manner. Fusing features in this way demonstrates comparable or better performance than existing methods while requiring significantly less computation and memory. In summary, our contributions are:

- We propose a novel fusion method that maintains or exceeds the performance of previous fusion methods while demonstrating up to a six-fold reduction in computation and memory requirements.
- We demonstrate that multimodal algorithms can tolerate far more token reduction than unimodal algorithms due to complementary cross-modal information. We show that by accounting for multimodal information during sparsification, more information can be removed without loss of performance.
- We perform extensive ablation studies on fusion components using real-world datasets to determine the efficacy of each model component. We further experiment with multiple pooling methods to demonstrate model robustness under different pooling requirements.

2. Related Work

The problem of modality fusion has been explored in numerous problem spaces for a long time [2]. The primary challenge is to find an effective way to combine representations of data from disparate modalities into a single representation for more accurate modeling. While the first methods for multimodal fusion were proposed to address signal inadequacies in individual modalities [32], we are now at a time when the resolution in each modality is much higher, making some computation costly and intractable. Therefore, we wish to purposely trade off some of the signal bandwidth to improve performance.

Many methods have been proposed to tackle the task of fusion. A way to categorize all these techniques is by the time of fusion occurrence. Early fusion typically refers to combining base level representations or even input values, while late fusion primarily refers to its application near the output. Early deep-learning methods typically make use of linear layers and cross products to combine modalities [10, 29, 33]. More rudimentary forms of fusion simply involve adding the logits of individual modality predictions together. As transformer-based architectures have become very popular recently, some recent techniques have also explored their use in multimodal settings. Originally proposed in [26] for neural machine translation (NMT) tasks, they have demonstrated superior performance on multiple benchmark problems such as image classification [9], action recognition [15] and 3D reconstruction [5, 23]. The basic functionality is to apply layers of self-attention, on sequential representations. To classify a discrete output, transformers typically rely on the use of a special token (`CLS`) that is prepended to the sequence for classification.

The most natural form of transformer fusion is simply to concatenate the sequence of tokens and rely on self-attention to learn their inter-dependencies. Works such as [12, 25] that do this learn better cross-modal representations and have shown benefits relative to naive fusion methods. Very recently, multimodal bottleneck transformers [15] have demonstrated a way for early fusion to occur without the use of costly cross-modal operations. However, the process of fusing multimodal information with some form of concatenation and dense attention remains costly due to the $O(N^2)$ complexity of transformers for input sequences of length N. It is this cost we seek to address with our sparsification approach.

Recent efforts have focused on reducing computational complexity for transformers and large-scale deep learning [20, 21, 24, 37]. An effective method for this is to exploit the representation of features within a small sliding window of tokens [30] on a long sequence. However, these methods require significant engineering efforts and are hard to train [35]. Other works approach the problem via sparsification of the attention mechanism, such as random or local attention [13, 18, 31]. Sparsification methods have also been applied successfully for some computer vision tasks [17].

Training optimizations for transformers have also been explored. Regularization techniques such as dropout [22], weight decay [14], and mixup [36] have all been applied. While weight decay and dropout can be applied in a modality-agnostic manner directly onto the weights, the use of mixup has primarily been used to tackle problems in the vision domain, as its application is easily interpretable and offers large benefits to the algorithms [4, 16]. Although some recent efforts have been made to enable the application of mixup on domains in a modality agnostic manner [27], its application in a fundamentally multimodal domain remains underexplored. Its use in the mixing of fused features across modalities spatially and across time demonstrates large benefits for our application.

3. Method

In this section, we describe our proposed Sparse Fusion Transformers (SFT). See Fig. 1 for a visualiza-

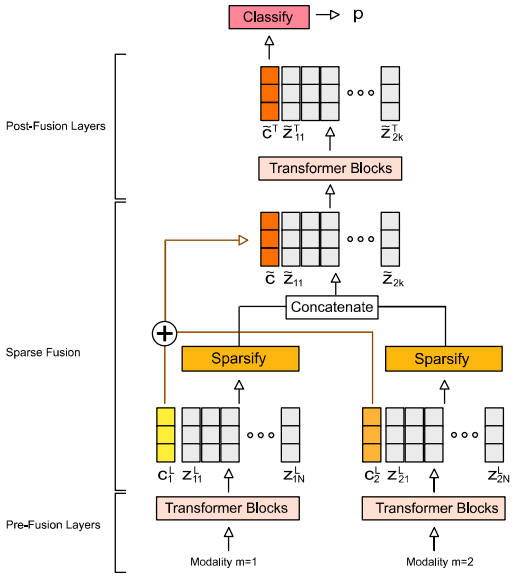


Figure 1. Visualization of our fusion method with two modalities. Following existing work, a special CLS token is appended to each unimodal token set prior to unimodal transformers. After unimodal transformers, the CLS token ($\mathbf{c}_1^L$ and $\mathbf{c}_2^L$) from each modality is summed. A pooled block-sparse attention is applied to local regions of each modality. The CLS token and pooled representations are then combined, and dense self-attention is applied to model global and cross-modal dependencies.

tion of our algorithm. As input, our method takes token sets from M different modalities, $\mathbf{Z}_1, \ldots, \mathbf{Z}_M$, with each modality consisting of N tokens of dimension D, $\mathbf{Z}_i = [\mathbf{z}_{i1}, \ldots, \mathbf{z}_{iN}] \in \mathbb{R}^{N \times D}$. Note the number of tokens N can vary from modality to modality but for simplicity of notation, we keep it fixed in our description. Additionally, if the token dimension D varies from modality to modality, we apply a per-token projection to keep the token dimension constant across all modalities. Following existing work, we prepend a special CLS token $\mathbf{c}$ with learnable parameters to each token set for each modality for the purpose of classification: $\hat{\mathbf{Z}}_i = [\mathbf{c}_i | \mathbf{Z}_i] = [\mathbf{c}_i, \mathbf{z}_{i1}, \ldots, \mathbf{z}_{iN}] \in \mathbb{R}^{(N+1) \times D}$. The goal of our method is classification, i.e., we want to learn a function $f_\theta : \mathbb{R}^{M \times (N+1) \times D} \rightarrow \mathbb{R}^C$:

$$f_\theta(\hat{\mathbf{Z}}_1, \ldots, \hat{\mathbf{Z}}_M) = \mathbf{p}, \tag{1}$$

such that $\mathbf{p}$ is the probability distribution over C classes.

Our method consists of three main parts. First, we model relationships between tokens within modalities using a standard transformer that is applied unimodally (Sec. 3.1). Second, we aggregate information within local regions of each sequence using block-sparse attention and then apply local subsequence pooling to sparsify the token set for each modality (Sec. 3.2). Third, we concatenate the sparsified features from each modality and run dense self-attention to predict a final class (Sec. 3.3). During training, we apply a novel multimodal variation of manifold mixup [27] for regularization of intermediate latent representations (Sec. 3.4).

3.1. Unimodal Modeling

In this stage, we apply a separate transformer to the token set from each modality. Following Vaswani *et al.* [26], we use a standard L-layer transformer encoder to model relationships between tokens in each modality. Each layer of the encoder consists of layer normalization (LN), Multi-head Self-Attention (MSA), and a Multi-Layer Perceptron (MLP). Given token set $\hat{\mathbf{Z}}^l$ after l transformer layers, the output of layer $l+1$ is:

$$\mathbf{Y}^l = \text{MSA}(\text{LN}(\hat{\mathbf{Z}}^l)) + \hat{\mathbf{Z}}^l \tag{2}$$

$$\hat{\mathbf{Z}}^{l+1} = \text{MLP}(\text{LN}(\mathbf{Y}^l)) + \mathbf{Y}^l \tag{3}$$

We apply a separate L-layer transformer per modality to get token sets $\hat{\mathbf{Z}}_1^L, \ldots, \hat{\mathbf{Z}}_M^L$.

3.2. Sparse Multimodal Fusion

In this stage, we apply local pooling blocks to each token set $\mathbf{Z}_i^L$ to extract k descriptive tokens per modality $\tilde{\mathbf{Z}}_i = [\tilde{\mathbf{z}}_{i1}, \ldots, \tilde{\mathbf{z}}_{ik}] \in \mathbb{R}^{k \times D}$, as represented by the "Sparsify" blocks in Fig. 1. As shown in our experiments in Sec. 5.3, information is quite redundant within and across each modality, and we hypothesize simple sub-sequence pooling to be a cheap and effective method for capturing important information while removing redundancies. Prior to pooling, we first apply a single bi-directional strided sparse attention layer [8] to enforce aggregation of dense local context and sparse global context to every token in the sequence to each modality. We then apply non-overlapping per-channel pooling blocks of stride s for each token set:

$$\tilde{\mathbf{z}}_{ij} = \text{pool}\left(\mathbf{z}_{i(js+1)}^L, \ldots, \mathbf{z}_{i(js+s)}^L\right). \tag{4}$$

A natural choice for pooling is either per-channel max pool or average pool. We explored several options in ablation studies and found our method to be robust to the choice of pooling (see Table 4). However, for our main experiments we use average pooling.

We additionally form a multimodal classification token $\tilde{\mathbf{c}}$ by summing the unimodal classification tokens:

$$\tilde{\mathbf{c}} = \sum_{i=1}^{M} \mathbf{c}_i^L \tag{5}$$

The final, fused token set $\mathbf{F}$ is formed using this classification token and the union of the unimodal pooled token sets $\tilde{\mathbf{Z}}_1, \ldots, \tilde{\mathbf{Z}}_M$:

$$\mathbf{F} = [\tilde{\mathbf{c}}, \tilde{\mathbf{z}}_{11}, \ldots, \tilde{\mathbf{z}}_{Mk}] \tag{6}$$

3.3. Dense Cross-modal Modeling and Prediction

To model cross-modal relationships, we apply a dense, T-layer transformer on the token set $\mathbf{F}$. Note the tokens of $\mathbf{F}$ are aggregated from all modalities. We adopt the same architecture used in the unimodal modeling task, denoting the token set after t transformer layers as $\mathbf{F}^t$, with the final output denoted $\mathbf{F}^T = [\tilde{\mathbf{c}}^T, \tilde{\mathbf{z}}_{11}^T, \ldots, \tilde{\mathbf{z}}_{Mk}^T]$. Finally, a small MLP followed by softmax is applied to $\tilde{\mathbf{c}}^T$ to produce a C-way class prediction $\mathbf{p}$.

3.4. Multimodal Manifold Mixup

We apply a novel variation of manifold mixup [27] for improved generalization. In the originally proposed mixup [36], given two random training inputs $\mathbf{x}_i$ and $\mathbf{x}_j$, their corresponding ground-truth labels $\mathbf{y}_i$, $\mathbf{y}_j$, and an interpolation weight $\lambda \in [0, 1]$, a classifier is trained using the following virtual training examples:

$$\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1 - \lambda)\mathbf{x}_j \tag{7}$$

$$\tilde{\mathbf{y}} = \lambda \mathbf{y}_i + (1 - \lambda)\mathbf{y}_j \tag{8}$$

Generally, the interpolation term λ is sampled from a Beta distribution $\mathrm{Beta}(\alpha, \alpha)$, where α is a hyperparameter. Manifold mixup extends this by also selecting a random layer l in an L layer network f and interpolating the latent representations $\mathbf{v}_i^l$, $\mathbf{v}_j^l$ of that layer instead of the input example:

$$\tilde{\mathbf{v}}^l = \lambda \mathbf{v}_i^l + (1 - \lambda)\mathbf{v}_j^l \tag{9}$$

Layers $l+1, \ldots, L$ of f are then applied to $\tilde{\mathbf{v}}^l$ and the output is supervised using Eq. 8. Manifold mixup has been shown to be more effective for regularization than input mixup.

We extend manifold mixup to the multimodal case for use with our model. Given our $(L + T)$-layer network, with the first L layers involving separate, unimodal transformers and the last T layers involving a single, multimodal transformer, we sample a single layer $l \in [1, L + T]$ for manifold mixup. If $l > L$, we use standard manifold mixup using Eqs. 8 and 9. If $l \leq L$, we sample a different interpolation term for each of the M modalities, $\lambda_1, \ldots, \lambda_M \sim \mathrm{Beta}(\alpha, \alpha)$. Given latent representation $\mathbf{v}_{mi}^l, \mathbf{v}_{mj}^l$ of layer l for modality m, the new latent representation is given as:

$$\tilde{\mathbf{v}}_m^l = \lambda_m \mathbf{v}_{mi}^l + (1 - \lambda_m)\mathbf{v}_{mj}^l \tag{10}$$

This is applied to every latent representation of layer l for every modality $1, \ldots, M$. After running the remaining $L + T - l$ layers, the output of the network is supervised using:

$$\tilde{\mathbf{y}} = \overline{\lambda_*}\mathbf{y}_i + (1 - \overline{\lambda_*})\mathbf{y}_j \tag{11}$$

where $\overline{\lambda_*}$ is the average of the M sampled λ values.

4. Experimental Setup

We now describe the datasets used for training and evaluation (Sec. 4.1), dataset pre-processing (Sec. 4.2), baseline network architectures used for comparison (Sec. 4.3), and training hyper-parameters we used (Sec. 4.4).

4.1. Datasets

We perform extensive experiments on two benchmark multimodal datasets: VGG-Sound [7] and CMU-MOSEI [34] The datasets tackle popular and broadly applicable tasks in multimodal machine learning for audio-visual classification and multimodal sentiment classification. The modalities evaluated include video, audio, and text data. Additionally, these datasets have differences in modality characteristics such as cross-modality alignment and information content.

4.1.1 VGG-Sound

VGG-Sound [7] consists of over 200,000 YouTube videos and their associated audio streams, each annotated with one of over 310 class labels. The audio spans a large range of challenging acoustic environments and noise characteristics of real applications. All videos are captured "in the wild." There are clear audio-visual correspondences, i.e., the sound source is visually evident. Each segment is 10 seconds long. To aid in evaluation, we select two subsets of data from VGG-Sound containing 10 classes and 100 classes each. We call these VGGS10 and VGGS100, respectively. We select VGGS10 by choosing pairs of easily confused classes, such as "baby babbling" and "baby laughing". We then build VGGS100 using these ten classes and additionally include 90 randomly chosen classes. The total training and testing set sizes for VGGS10 are 6,051 and 459. For VGGS100, the training set size is 66,180 and the test set size is 4,549. A validation set is extracted by taking 20 percent of the training set.

4.1.2 CMU-MOSEI

The CMU Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) [34] dataset is one of the largest multimodal sentiment analysis and emotion recognition datasets to date. The dataset contains more than 23,500 sentence utterance videos from more than 1000 online YouTube speakers. The dataset is gender-balanced. All utterances are randomly chosen from various topics and monologue videos. The task is to predict a 7-class sentiment score of a particular multimodal video sample. Each sample contains audio, video, and text modalities. This dataset is frequently used to explore the unaligned nature of multimodal sequences between text and video.

4.2. Pre-processing

Each modality is pre-processed with a feature extraction pipeline in order to generate the input token sequence. For the MOSEI dataset, we use the pre-processed data provided by the authors. The pre-processing pipeline that was used assumes that each video depicts a "talking head": a single human talking, whose face is visible and whose voice is clearly audible. This assumption is valid for the MOSEI dataset, and the pre-processing pipeline therefore extracts visual features such as facial landmark positions and audio features such as estimated vocal parameters. We refer the reader to Zadeh *et al*. [34] for the full details. To pre-process VGGSound, we employ a feature extraction pipeline that can be applied to videos more generally, without assuming human faces or voices are present.

For the VGGS10 and VGGS100 datasets, we extract *visual features* using I3D [6], a spatio-temporal video feature extraction model that was pre-trained on the Kinetics human action recognition dataset [6]. This is a two-stream model, which processes optical flow and raw RGB independently as two separate modalities. We also extract TV-L[1] optical flow from the VGGSound videos. For *Audio* pre-processing we follow Nagrani *et al*. [15]: we resample all audio at 16Hz and convert to mono, then compute log mel spectrograms with 128 frequency bins, using a Hamming window with size 25ms and stride 10ms.

4.3. Baseline Network Architectures

We compare against the following transformer-based fusion methods:

Self-Attention Fusion (Concat): A baseline method of fusion is to concatenate the individual modality representations prior to input to any network and rely exclusively on dense self-attention. This is a form of early fusion.

Late Fusion (LF): This method works by applying transformer blocks on individual modalities only. The final prediction is obtained via a summation of logits derived from individual class tokens. This helps us compare the benefit of modeling cross-modal interactions.

Multimodal Transformer (MulT): [25] MulT is a hybrid early-late attention-based fusion method using a unique cross-modal attention mechanism. The data is first fused via an attention mechanism by using one modality each for key, query, and value. Transformer blocks are then stacked on top. At the very end, the features are concatenated and a prediction is obtained after an FC layer.

Bottleneck Fusion (MBT): [15] This is a form of fusion in which special tokens called bottleneck tokens are introduced. These tokens are shared among all modalities, and transformers alternate operating on each modality independently. The final `CLS` token is summed from each modality and used for prediction. We additionally evaluate MBT using manifold mixup (MBT+MM) as the original paper used input mixup, and our inputs are features.

4.4. Implementation details

Our model is implemented in PyTorch. For all experiments on the smaller datasets VGGS10 and MOSEI we use a learning rate of 10^{-4}. For the larger dataset VGGS100 we use a learning rate of 10^{-3}. Learning rate is decayed by factor of 10 every 10 epochs based on minimum validation loss. We use a batch size of 24 for all experiments. For all datasets, we report results based on averaging performance training from 5 different seeds for generalization purposes and to minimize tuning effects. We use a standard 12-layer network and 5 attention heads for all evaluations. We project embeddings from each modality to 40 to minimize the effects of over-parameterization. For experiments involving latent mixup, we used a strength of $\alpha = 0.3$. We use an initial warm-up of 5 epochs in which no mixup is applied. For all other experiments we applied dropout $p = 0.2$ for regularization. For baselines, we follow descriptions in original papers and publicly available code for comparison. All experiments were conducted on consumer-grade graphics cards. We make our code and preprocessed data publicly available.

4.5. Metrics

We report results using commonly used metrics. **Top1** represents the accuracy of the most likely class. **mAP** represents the mean of per-class average precision scores. We also report the computational cost in Giga floating-point operations (GFlops) which is estimated similar to previous methods [17] (we provide the equations used for estimating this in Appendix A.2 included in the supplementary materials). Many experiments examine the effect of a reduction factor, which refers to reducing the number of tokens in the sequence dimension for transformer architectures. We report most results as a mean and standard deviation of experiments run with five different seeds.

5. Results

We first report our results against state of the art (Sec. 5.1) showcasing our performance on multiple datasets from different domains. We then perform a series of ablation studies to explore the effects of sparsification (Sec. 5.2), and the benefits of addressing within-modality redundancies during fusion (Sec. 5.3). We also study the effect of pooling choice (Sec. 5.4) and the effect of our proposed multimodal manifold mixup (Sec. 5.5).

5.1. Comparison against state of the art

We present our summary benchmark performance on real-world datasets VGGS10, VGGS100, and MOSEI in Tables 1 and 2. For each dataset, our model keeps a sub-

	VGGS10		VGGS100		MOSEI	
	Top1	**mAP**	**Top1**	**mAP**	**Top1**	**mAP**
Concat	67.62 ± 1.3	**71.46** ± .63	51.72 ± .26	51.64 ± .13	48.47 ± .23	32.40 ± .83
LF	67.10 ± .79	70.46 ± .79	52.00 ± .73	46.92 ± .28	49.10 ± .33	31.75 ± .78
MulT	65.49 ± .40	69.73 ± 1.1	51.35 ± .43	49.25 ± .43	49.36 ± .34	31.92 ± .79
MBT	66.84 ± .61	70.98 ± .78	51.67 ± .66	51.29 ± .37	49.12 ± .27	32.15 ± .47
MBT+MM	66.80 ± 1.8	70.56 ± .61	**55.97** ± .42	**57.29** ± .37	48.77 ± .37	32.03 ± 1.2
Ours	**67.71** ± 1.3	71.06 ± .81	55.61 ± .61	57.18 ± .39	**49.67** ± .23	**33.66** ± .85

Table 1. Accuracy comparison for each dataset and model. For all benchmarks we report the mean and standard deviation performance over 5 seeds to minimize tuning effects. Bold indicates best, underline second best. We are either best or close to best in all metrics.

	VGGS10/VGGS100				MOSEI			
	Mem (GB)	**Eval (ms)**	**Train (ms)**	**GFlops**	**Mem (GB)**	**Eval (ms)**	**Train (ms)**	**GFlops**
Concat	1.52 (3.16×)	3.59 (2.46×)	10.93 (2.56×)	1.68 (6.72×)	1.04 (11.95×)	2.71 (2.39×)	8.02 (2.01×)	1.16 (11.60×)
LF	1.35 (2.80×)	3.54 (2.42×)	12.32 (2.88×)	1.51 (6.04×)	0.49 (5.66×)	2.00 (1.77×)	7.66 (1.92×)	0.59 (5.90×)
MulT	1.18 (2.45×)	3.53 (2.41×)	16.73 (3.91×)	2.64 (10.56×)	0.62 (7.10×)	2.84 (2.50×)	11.49 (2.88×)	1.03 (10.30×)
MBT	1.35 (2.82×)	3.59 (2.45×)	12.02 (2.81×)	1.52 (6.08×)	0.50 (5.72×)	2.14 (1.89×)	7.42 (1.86×)	0.59 (5.90×)
Ours	**0.48**	**1.46**	**4.27**	**0.25**	**0.09**	**1.13**	**3.99**	**0.10**

Table 2. Computational cost comparison for each dataset and model. For all metrics we obtain results with a single RTX 3090. Metrics are normalized by the batch size. Our method has the lowest cost. GFlops is estimated based on number of transformer blocks and token operations and represents a theoretical cost for a single forward pass through the network. We present the equations used for calculations in the Appendix A.2 of the supplementary material.

set of tokens from each modality during pruning. For VGGSound data after pooling we have 12 tokens of RGB and flow information and 20 tokens spectrogram data. For MOSEI, we keep 10 tokens of visual and audio information and 25 tokens of text information. These numbers were chosen according to experiments described in Sec. 5.3.

We maintain the performance of existing fusion methods and exceed them in some situations while significantly reducing the amount of computation required. For MOSEI we report more than a five-fold reduction in computational cost while achieving the best performance in terms of both Top1 accuracy and mAP. For VGGS10 and VGGS100, we observe approximately a six-fold reduction in computational cost. Our method also exceeds the performance of multiple fusion methods on the VGGS100 dataset.

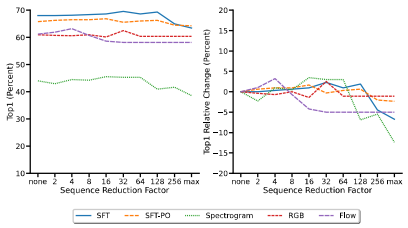
5.2. Effect of Sparsification

In this section, we explore the effect of how naively applying pooling can affect multimodal models. In particular, we are interested in how pooling affects fused versus modality-independent features. We answer this question by comparing the performance of late fusion, concatenation fusion, and our fusion method. For concatenation fusion, we concatenate all the input tokens prior to input into the model. From here, we apply a single transformer block as if the number of modalities is $M = 1$. We then apply max pool with a kernel and stride of 64. Afterwards, we apply eleven more transformer layers to obtain the result. For late fusion and our method, we also apply pooling on the representations after the first layer. However, the pooling is conducted on unimodal representations. In late fusion, transformer layers are applied independently for each modality and the final result is obtained via a summation of logits obtained from the `CLS` token. In experiments described in Sec. 5.3, we observe a drop in performance for our method with strides larger than 32 for some datasets and 128 for others, thus we assume a stride of 64 will provide meaningful comparisons between fusion methods.

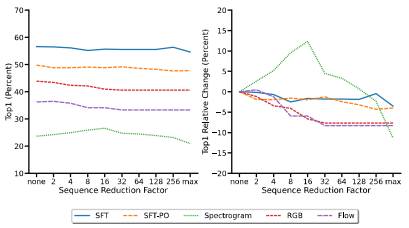
		Token Reduction Factor		
		None	**64×**	**Diff.**
Concat	*Top1*	51.72 ± .26	46.29 ± .73	−5.45
	mAP	51.64 ± .13	47.49 ± .63	−4.49
LF	*Top1*	52.00 ± .73	49.76 ± .71	−2.24
	mAP	46.92 ± .28	45.96 ± .62	−0.96
Ours	*Top1*	**55.57** ± .23	**55.98** ± .28	+0.41
	mAP	**56.54** ± .49	**56.91** ± .60	+0.37

Table 3. Comparison of our method for sparsification versus application of only pooling in baseline methods on VGGS100. `Diff` column shows difference between no reduction of tokens and taking 1/64ths of the tokens, where the minimum is one token per modality. Our method is more robust than naive methods of pooling. Pooling has a large effect when training with fused features (Concat) which we solve using our method. Difference for the same reduction factors between Top1 and mAP shows that late fusion (LF) tends to fit some samples better than others and suggests the advantages of an early-fusion method.

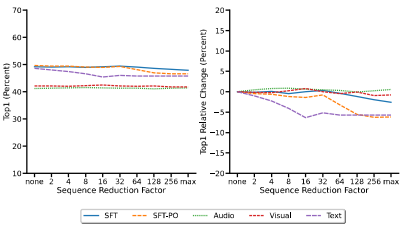
The results shown in Table 3 demonstrates that our method for sparsification is more robust than naive methods. We see that in both naive methods of pooling, the reduction in the sequence dimension causes a significant drop in performance. Our method does not see any reduction, instead experiencing a small boost in performance. Furthermore,



(a) Top 1 absolute score and relative change from no pooling for VGGS10. Multimodal performance degredation occurs after a 64-fold reduction in sequence length. Compared to flow at 4, and spectrogram at 64. We outperform all all methods at all reduction levels. SFT exceeds the pooling only variant (SFT-PO).



(b) Top 1 metrics for VGGS100. SFT degradation occurs at 256-fold reduction compared to 64 for SFT-PO and 2 for Flow and RGB. Audio representations might benefit from better feature extraction, however there is dramatic loss of performance with very few tokens, while we remain tolerant.



(c) Top 1 metrics for MOSEI. SFT degrades minimally until max while SFT-PO degrades at 32. Text modality degrades immediately. Information appears highly redundant in Audio-Visual modalities.

Figure 2. Comparison of reduction factor effect on performance difference against no reduction for unimodal and multimodal models. Reduction in total length of fused features reported in the x-axis. In cases where the reduction factor is greater than sequence length of a particular modality, a single token along the sequence dimension is passed through. Sequence lengths for VGGS10 and VGGS100 are 38 for RGB and Flow, 1200 for Spectrogram. For MOSEI, Audio and Visual is 500 while text is 50. Top1 absolute score and relative change from using no pruning is reported. For all experiments we used a batch size of 24. Multimodal models will tolerate more pruning over unimodal models by making up for the lost information through fusion. Notably, SFT exceeds performance of SFT without sparse attention or mixup (SFT-PO) in all cases and tolerates more reduction. Pooling offers some benefits for feature extraction in some cases for longer sequences.

we see that concatenation fusion tends to have a higher mAP metric, whereas late fusion has a higher Top1. Overall, our method is robust, and pooling has no detrimental effect even when removing over 98% of tokens.

5.3. Within-Modality Information Redundancy

We provide experiments to analyze why it is advantageous to address the within-modality redundancy problem during fusion. In particular, we wish to show that pooling when accounting for multimodal information is more robust than pooling without this information. We set up the experiment so that a max-pooling layer is applied after the first layer of transformers to simulate modality-independent feature sparsification for each method. We then compare pruning by an equal factor for each modality to observe the effect on overall performance, referred to as "sequence reduction factor." We set the minimum allowed sequence length to one to avoid removing all tokens. We compare against unimodal transformers for each modality. We also evaluate two versions of our method: SFT which is our full pipeline, and SFT-PO which removes the strided sparse attention layer and multimodal manifold mixup and includes only the strided pooling.

In the first column of Fig. 2, we present Top1 accuracy as a function of sequence reduction factor. In the second column, we present the relative change in Top1 accuracy when compared with no sequence reduction. Lower indicates a performance degradation from sequence reduction. Multimodal models exceed unimodal performance in all reduction factors. We generally see a performance decrease for each unimodal model as the reduction factor increases. However, some modalities do not decrease due to two likely reasons: 1) from redundancies in information and 2) that all useful information was extracted after just a single layer of transformers. We also see that some modalities experience an increase in performance as we reduce the number of tokens, signifying better feature extraction for those. However, in general, the performance of unimodal models with less redundant information all decrease, while our model (SFT) is more robust. In particular, SFT is better than using just pooling (SFT-PO) as is evident from it maintaining higher performance with greater reduction factors.

We see that up to a factor of 50 for evaluations conducted on MOSEI, there is very minimal drop in performance in the multimodal model. However, the performance of the text-only transformer drops observably larger than our multimodal model. The performance of the RGB and Audio transformers remains the same throughout the experiment. This signifies two things: that the information for label present in the text classifier is less redundant than in RGB and Audio features for this dataset, and that application of sparse fusion can compensate for the loss of information necessary for classification by exploiting the other modalities. The effect of unimodal models experiencing a decrease in performance is also evident for the optical flow modality on the VGGS10 dataset at $8\times$ reduction, and at $64\times$ reduction for spectrogram data. On VGGS100, we see the same, where both the RGB and flow modalities experience decreases in performance with a pruning factor of just $2\times$ while our model's performance remains relatively flat. Furthermore, our multimodal model with one token per-modality after pruning still achieves better performance

Pooling Method	VGGS10		VGGS100		MOSEI	
	Top1	*mAP*	*Top1*	*mAP*	*Top1*	*mAP*
Max	67.0 ± 1.1	70.7 ± 0.7	55.7 ± 0.4	57.3 ± 0.4	49.4 ± 0.2	33.2 ± 0.3
Average	67.7 ± 1.2	71.1 ± 0.7	55.6 ± 0.5	57.2 ± 0.3	49.7 ± 0.2	33.7 ± 0.9
Attn Average	67.5 ± 1.0	71.1 ± 0.7	55.4 ± 0.3	56.9 ± 0.4	49.4 ± 0.2	33.7 ± 0.8

Table 4. Comparison of pooling method on VGGS10, VGGS100, and MOSEI datasets. Based on Top 1 accuracy and mean average precision metrics, we find our method robust to pooling type.

mixup?	**Top1**	**mAP**
✓	$55.61 \pm .61$	$57.18 \pm .39$
×	$51.30 \pm .80$	$51.80 \pm .44$

Table 5. Comparison of model performance on VGGS100 when trained with and without our multimodal manifold mixup.

than a unimodal model which uses all tokens.

These observations signify that certain modalities contain information that is more redundant than others and that even if we filter out more than what a model with redundant information is able to predict, the multimodal model is able to make up for that. The same is not true for unimodal models, which cannot filter out unnecessary information as well, and is not robust to this reduction. Even under extreme circumstances where information is reduced to the length of a single token, performance of the multimodal degrades but still remains the overall top performer.

5.4. Effect of Pooling

In this section, we explore the effects of using various pooling choices in the network. See Table 4 for results on the VGGS10, VGGS100, and MOSEI datasets. We use max pooling, average pooling, and attention-weighted average pooling, denoted "Max," "Average," and "Attn Average" respectively. For attention-weighted averaging, we weight using a simple, attention-based per-token significance metric proposed by Goyal *et al.* [11]. Given the attention weights $\mathbf{W}^h \in \mathbb{R}^{N \times N}$ calculated from layer L head $h \in \{1, \ldots, H\}$ of the pre-fusion network, the significance (sig) for token i is:

$$\text{sig}(i) = \sum_{h=1}^{H} \sum_{n=1}^{N} W_{in}^{h} \tag{12}$$

Interestingly, all metrics are within 1 percentage point of each other across the three pooling types. This indicates our model is quite robust to the choice of pooling type. Average pooling appears better, but this is well within the std. dev.

5.5. Effect of Multimodal Manifold Mixup

See Table 5 for results from SFT trained on VGGS100 with and without the use of our multimodal manifold mixup during training. Without mixup, we observe over a 4% reductin in Top1 and over a 5% reduction in mAP. This drop in performance is quite significant, indicating the effectiveness of training with our proposed multimodal manifold mixup.

6. Limitations

We provide an effective method for quickly ingesting and classifying large quantities of multimodal sequential data with high levels of accuracy. However, we do not provide evaluations on how this fusion method might behave as part of a generative network and we leave this for future work. Secondly, our methods operate on extracted features such as I3D and spectrogram data. While we follow popular and common settings for feature extraction, improved unimodal modeling might be able to condense the representations and reduce within-modality redundancy. This would lead to slightly reduced complexity benefits. However, the large differences between our results and unimodal approaches as well as maintaining performance under extreme sparsification support our conclusions.

7. Conclusion

We present an effective technique that offers more than a five-fold reduction in computational cost while maintaining the performance of state-of-the-art fusion techniques. Different fusion methods exhibit improved performance under varying conditions when all input conditions are equal. However, when optimizing for speed, there are drastic improvements that can be made to feature selection during cross-modal modeling that can improve performance.

Broader Impacts: We propose sparse fusion for multimodal transformers as a method to reduce computational costs. This translates to energy savings and is beneficial for numerous applications including on mobile devices. Namely, it has the potential to train and fine-tune a network for use to a specific user without needing to offload the training to a server. This preserves the privacy of the user while providing benefits of performance and energy savings. Furthermore, we hope to spur democratization of learning on large datasets by enabling rapid development and evaluation on consumer-level hardware. However, we hope that by enabling this technology on mobile devices it is not applied to tasks such as unlawful surveillance.

References

[1] Sara Atito, Muhammad Awais, and Josef Kittler. SiT: Self-supervised vision transformer. *CoRR*, abs/2104.03602, 2021. 1

[2] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018. 2

[3] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers, 2021. 1

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019. 2

[5] Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. TransformerFusion: Monocular RGB scene reconstruction using transformers. *Proc. Neural Information Processing Systems (NeurIPS)*, 2021. 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 5

[7] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. VGGSound: A large-scale audio-visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 721–725. IEEE, 2020. 4

[8] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*, 2019. 3

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 1, 2

[10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016. 2

[11] Saurabh Goyal, Anamitra R. Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. *arXiv preprint arXiv:2001.08950*, 2020. 8

[12] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021. 2

[13] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 2

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 2

[15] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *arXiv preprint arXiv:2107.00135*, 2021. 1, 2, 5

[16] Kento Nishi, Yi Ding, Alex Rich, and Tobias Hollerer. Augmentation strategies for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8022–8031, 2021. 2

[17] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 377–386, 2021. 1, 2, 5, 11

[18] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019. 2

[19] Wasifur Rahman, Md Kamrul Hasan, Sangwu Lee, AmirAli Bagher Zadeh, Chengfeng Mao, Louis-Philippe Morency, and Ehsan Hoque. Integrating multimodal information in large pretrained transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2359–2369, Online, July 2020. Association for Computational Linguistics. 1

[20] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020. 2

[21] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training. *arXiv preprint arXiv:2101.06840*, 2021. 2

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 2

[23] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. VoRTX: Volumetric 3D reconstruction with transformers for voxelwise view selection and fusion. In *3DV*, 2021. 2

[24] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. 1, 2

[25] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2019, page 6558. NIH Public Access, 2019. 2, 5

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 2, 3

[27] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019. 2, 3, 4

[28] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[29] Yansen Wang, Ying Shen, Zhun Liu, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. Words can shift: Dynamically adjusting word representations using nonverbal behaviors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7216–7223, 2019. 2

[30] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. *arXiv preprint arXiv:1908.08167*, 2019. 2

[31] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. BP-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*, 2019. 2

[32] B.P. Yuhas, M.H. Goldstein, and T.J. Sejnowski. Integration of acoustic and visual speech signals using neural networks. *IEEE Communications Magazine*, 27(11):65–71, 1989. 2

[33] Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Memory fusion network for multi-view sequential learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[34] Amir Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246, 2018. 4, 5

[35] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for longer sequences. In *NeurIPS*, 2020. 2

[36] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 4

[37] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *arXiv preprint arXiv:2010.13369*, 2020. 2

A. Appendix

A.1. Label Distributions

We summarize the classes we used in VGGS10 and provide the label distributions in VGGS10 and VGGS100. VGGS10 is a manually curated dataset built by selecting pairs of difficult to separate classes from the full VGGSound dataset as well as for differences between video and audio modalities. We chose the following ten classes: airplane, baby babbling, baby crying, baby laughter, cat meowing, cat purring, people marching, people running, playing bass guitar, playing electric guitar. The final training set distribution for VS10 in Fig. 3, and the final VGGS100 dataset distributions are show in Fig. 4.

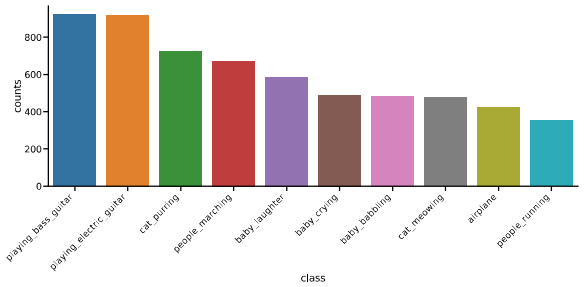


Figure 3. Label distribution of VGGS10 dataset

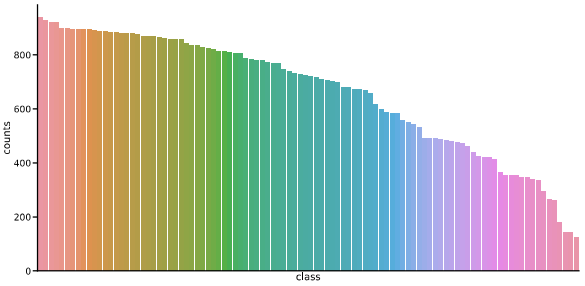


Figure 4. Label distribution of VGGS100 dataset

A.2. Flop computation

We present all flop estimates in the paper using the following equations. We primarily follow the flop estimation from [17] with some minor changes due to layer differences. Each transformer layer consists of a multi-head attention and multilayer perceptro block. A multi-head attention (MHA) block has cost of:

$$\begin{aligned}\phi_{MHA} &= \phi_{qkv} + \phi_A + \phi_O + \phi_{proj} \\ &= 3nd^2 + n^2d + n^2d + nd^2 \\ &= 4nd^2 + 2n^2d \end{aligned} \tag{13}$$

where n, d represent the length and embedding dimension, ϕ_{qkv} is the cost of projecting to the query, key, and values. ϕ_A is the cost of the attention map, ϕ_O is the cost of the self attention, and ϕ_{proj} is the cost of projection for self-attention outputs.

A MLP block includes two linear layers as well as a normalization layer for a cost of:

$$\begin{aligned}\phi_{MLP} &= \phi_{proj1} + \phi_{norm} + \phi_a + \phi_{proj2} \\ &= nd^2 + 3nd + nd + nd^2 \\ &= 2nd^2 + 4nd \end{aligned} \tag{14}$$

where ϕ_{proj1} and ϕ_{proj2} are cost of projecting into and out of latent space for transformer block, ϕ_{norm} represents cost of applying layer normalization, and ϕ_a represents the cost of an activation function.