Where to fuse

Master thesis

Lukas Petersson

January 13, 2024

University Supervisor: Andreas Jakobsson, Lund University
Company Supervisors: Zhe Dong and Enrique Alfonseca, Google Zürich

Abstract

This thesis investigates fusion techniques in multimodal transformer models, focusing on enhancing the capabilities of large language models in understanding not just text, but also other modalities like images, audio, and sensor data. The study compares late fusion (concatenating modality tokens after separate encoding) and early fusion (concatenating before encoding) techniques, examining their respective advantages and disadvantages. It examines a mid-fusion approach, aiming to combine the strengths of both methods. The effectiveness of this approach is evaluated in terms of accuracy and computational impact on the Visual Question Answering (VQA) task. Using a pretrained T5 model, the research incorporates image tokens (calculaed by Vision Transformer, ViT) into intermediate activations of the model. The findings indicate that standard early fusion techniques underperform with larger decoders, while late fusion with a smaller decoder yields the best results on the VQA task. This conclusion also extends to pooled modality tokens. Additionally, the thesis includes a comprehensive literature study, identifying benchmark datasets for video understanding in multimodal learning and highlighting datasets that demand a robust understanding of all involved modalities. This research contributes to the field by exploring and validating a novel fusion technique in multimodal learning, offering insights into its practical applications and limitations.

Contents

1 Introduction

1.1 Motivation

1.1.1 Where to Fuse

Natural Language Processing (NLP) is an interdisciplinary field that merges computer science, linguistics, and artificial intelligence to enable machines to understand, interpret, and generate human language. At its core, NLP seeks to bridge the gap between human communication and computer understanding. In this thesis I say that a model "understand" when it is able to accomplish tasks which is only achievable by a human if the human understand. A Language Model, a central concept in NLP [1], is a machine learning model that is trained to predict the likelihood of a sequence of words. It can generate text, complete sentences, or even predict the next word in a sequence based on the context provided by the preceding words.

Machine Learning (ML) plays a pivotal role in building these models in NLP. ML refers to the process where computers use statistical techniques to learn from data, improving their performance on a specific task without being explicitly programmed for that task [2]. Among various ML approaches, deep learning, a subset of ML, has shown remarkable success in NLP. Deep learning involves the use of neural networks with multiple layers, each layer learning different aspects of the data and contributing to the model's overall understanding and decision-making process [2].

A Layer in a neural network context refers to a collection of neurons, basic units of computation in the brain, working together within the network [2]. Each layer can transform the input data in a different way, thanks to the neurons' ability to learn from data. NLP has recently experienced significant advancements as it was discovered that its capabilities grow steadily as you increase the model sizes, meaning more and bigger layers. These larger models can capture a broader range of language nuances and complexities, leading to more accurate and coherent outputs.

The public became widely aware of these capabilities when OpenAI released ChatGPT [3]. This event marked a turning point, showcasing the practical applications and potential of advanced NLP models. Following this, researchers have started to investigate how language models can use other modes of input when generating text. A modality in this context refers to the type of data or way in which information is represented, such as text, images, or sound. One noteable outcome of this research is Google's Gemini model [4].

Multimodal Language Models, therefore, are models that can process and integrate information from multiple different modalities. These models are designed to understand and generate content that goes beyond mere text, potentially leading to richer and more contextually relevant outputs. However, a critical challenge in multimodal learning is modality fusion, the process of combining different types of data into a unified representation that can be processed by the model.

The two most common modality fusion techniques are early fusion and late fusion. Early fusion involves combining the different modalities at an initial stage, before feeding them into the model [5]. This approach allows the model to learn from the integrated data from the start but can be computationally expensive and less flexible. Late fusion, on the other hand, involves processing each modality separately with different models and combining their outputs at a later stage [5]. This method is more flexible and less computationally demanding but may fail to capture the interdependencies between modalities effectively.

Mid fusion potentially offers a middle ground, fusing modalities at an intermediate layer of the model. It can be a compromise between the pros and cons of early and late fusion, leveraging the benefits of both. Mid fusion is not a single method but a spectrum of approaches, ranging from fusing at an early layer (close to layer 0) to a later layer (close to the last layer). Mid fusion is more complex to implement and has therefore not been studied as much in practice.

The main problem statement of this work is to investigates which layer is best to fuse at in a mid fusion setting. Finding the optimal fusion point could lead to models that are both better and computationally efficient. Such efficiency is particularly important for making advanced NLP models accessible on mobile devices, where computing resources are limited. Additionally, better-performing models can lead to more accurate and reliable applications of NLP in various domains, ranging from automated customer service to enhanced language translation services. The potential benefits of this research are vast, promising to further the reach and effectiveness of NLP in my increasingly digital world. More efficent algorithms is also a benefit for the enviroment.

1.1.2 Evaluation

Evaluating language models is a complex and nuanced task. Unlike other machine learning tasks where you can often compare model predictions directly with ground truths, language presents unique challenges. For example, in image classification, the model's output can be directly compared with the actual label of the image to determine accuracy [2]. However, when it comes to text, the inherent flexibility and variability of language come into play. There are multiple ways to express the same idea or answer the same question, making direct comparison less straightforward.

This complexity is further amplified in the context of evaluating multimodal models. In these models, it is essential to ensure that the evaluation process comprehensively tests the model's understanding of all individual modalities involved. An effective evaluation should not allow the model to achieve high scores by only excelling in one modality while neglecting others [24]. The evaluation must be holistic, ensuring that the model demonstrates an integrated understanding of all the modalities it is designed to process.

To address these challenges, there have been attempts to quantify the similarity between sentences [18]. This approach can be used to evaluate model outputs by comparing how similar the generated text is to a target sentence. Such similarity measures, however, must be sophisticated enough to capture the nuances of language, including semantics, syntax, and context.

This work also includes a comparison between different video understanding benchmarks and their suitability for evaluating multimodal capabilities. Video understanding benchmarks are particularly relevant for multimodal models as they often involve the integration of visual and auditory information with text. By analyzing and comparing these benchmarks, this research aims to identify which are most effective at assessing a model's ability to process and integrate information across different modalities.

Understanding the strengths and limitations of various evaluation benchmarks is crucial for several reasons. First, it helps in the development of more robust and capable multimodal models by providing clear and comprehensive criteria for assessment. Second, it advances the field by setting standards for what constitutes a successful integration of multiple modalities in language models. Finally, by identifying the most effective benchmarks, this research contributes to the broader goal of improving the accuracy and reliability of multimodal language models, making them more useful and applicable in real-world scenarios where multiple forms of data are often present.

1.1.3 Contributions

This work show how a multimodal large language model perform on a visual question answering task [16]. This can be used to build intuition for how other models act in similar circumstances, which can be useful for projects with smaller compute budgets. It remains to be determined to what extent these result carry over to other models. Furthermore I identify three video understanding datasets which show promise as good benchmarks for multimodal understanding.

1.2 Background

1.2.1 History of NLP

Language models (LMs) have become a cornerstone of modern natural language processing (NLP) [1]. In its essence, they are next token predictors but can be used in applications ranging from simple text classification to complex dialogue systems. Tokens are a group of characters that often appear together. For example, the word "is" is a single token, but "training" will be split into two tokens: "train" and "ing" [1]. .

The early development of language models was significantly influenced by statistical methods, with n-gram models playing a pivotal role. An n-gram is a sequence of 'n' items (words or characters) from a given sample of text or speech. The core principle of n-gram models is based on the Markov assumption, which posits that the probability of a word is dependent only on a limited number (n-1) of preceding words [1].

This concept is more clearly demonstrated in the bigram model (where n=2), which approximates the probability of a word (w_n) given all preceding words ($P(w_n|w_1, w_2, ..., w_{n-1})$) by using only the conditional probability based on the immediately preceding word ($P(w_n|w_{n-1})$). In other words, the bigram model simplifies the probability estimation by focusing only on the most recent word in the sequence.

Similarly, this approximation can be extended to trigrams (n=3) and larger n-grams, where the focus is on the last $n-1$ words. The general form of this approximation for an n-gram model can be mathematically expressed as:

$$P(w_n|w_1, w_2, ..., w_{n-1}) \approx P(w_n|w_{n-N+1}, ..., w_{n-1})$$

In this formula, 'N' represents the size of the n-gram (e.g., N=2 for bigrams, N=3 for trigrams). This approximation allows for the computation of the probability of a complete sequence of words ($P(w_1, w_2, ..., w_n)$) by considering the conditional probability of each word given its immediate $N-1$ word context [1].

Despite their effectiveness in capturing local context, n-gram models suffer from the curse of dimensionality and have difficulties handling long-range dependencies due to their Markovian assumption (i.e., the future is independent of the past given the present). Also, n-grams require a large corpus (collection of text) to accurately model the probabilities of word sequences, and they struggle with the sparsity of linguistic data, often encountering unseen word combinations during training.

To overcome the limitations of traditional n-gram models, which could only capture limited context and were prone to data sparsity issues, researchers shifted focus towards distributed representations of words. This approach led to the development of models that represent words as vectors in a continuous vector space. A significant milestone in this evolution was the introduction of Word2Vec by Mikolov et al. [6]. Word2Vec models, especially its two architectures - Skip-gram and Continuous Bag-of-Words (CBOW), learn to represent words as dense vectors (i.e., embeddings) in a way that words with similar meanings have similar representations. The optimization of these models aims to create word embeddings that capture both semantic (meaning-based) and syntactic (grammar-based) properties of words.

These learned embeddings are then employed in various Natural Language Processing (NLP) tasks, like text classification, sentiment analysis, and machine translation. They have shown to provide superior results compared to traditional sparse representations, such as one-hot encoding, mainly because they efficiently capture the nuanced relationships between words in a dense, low-dimensional space.

Building on the success of word embeddings, a new class of models started to gain popularity in the field of natural language processing. These models are based on the concept of a neural network, which is a computational framework inspired by the structure and functioning of the human brain. Neural networks consist of interconnected units or nodes that work together to process and analyze data, learning to perform tasks by considering examples. Among these models, the Recurrent Neural Network (RNN) and its variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have become particularly prominent. An RNN is a type of neural network specifically designed to handle sequential data. Unlike traditional neural networks, which process inputs independently, RNNs have loops in them, allowing information to persist and flow based on the sequence of the data. This architecture makes RNNs ideal for tasks like language modeling, where understanding the sequence and context of words is crucial [7]. Unlike n-gram models, RNNs can, in theory, capture long-range dependencies:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$

$$y_t = W_{hy}h_t + b_y \tag{2}$$

Here, x_t is the input at time step t, h_t is the hidden state, y_t is the output, and W and b are the learnable parameters of the model. f is a non-linear activation function, often tanh or a sigmoid function [2]. RNNs process input sequences one element at a time, maintaining a 'memory' of previous inputs through their internal states.

Despite their theoretical advantages, RNNs and LSTMs are notoriously difficult to train effectively due to issues like vanishing and exploding gradients. These challenges arise from the multiplicative gradient that can exponentially decrease or increase through time, particularly with long sequences, making it difficult for the model to learn long-range dependencies. This limitation led to the exploration and eventual adoption of alternative architectures, like the transformer, for processing sequential data in language tasks.

Prior to transformers, attention mechanisms were integrated into neural network architectures as a solution to the shortcomings of RNNs and LSTMs in capturing long-range dependencies. Attention

mechanisms allow models to focus on different parts of the input sequence when predicting each word in the output sequence, essentially providing a shortcut for information flow across distant positions in the sequence [1]. Here, focus mean to lay more importance on the contributions of certain inputs when calculating the output.

Bahdanau et al.'s introduction of the attention mechanism in neural machine translation was a seminal development [8]. The attention mechanism is a process that allows a neural network to focus selectively on certain parts of an input sequence when producing an output, similar to how human attention works when I focus on specific aspects of what I see or hear.

This method enables the model to dynamically select and focus on different parts of the input sequence, improving its ability to capture long-distance dependencies, a crucial requirement in many NLP tasks, especially in machine translation. The success of attention mechanisms in enhancing sequence modeling capabilities paved the way for the development of transformer models, which entirely rely on attention mechanisms, dispensing with recurrence and convolutions.

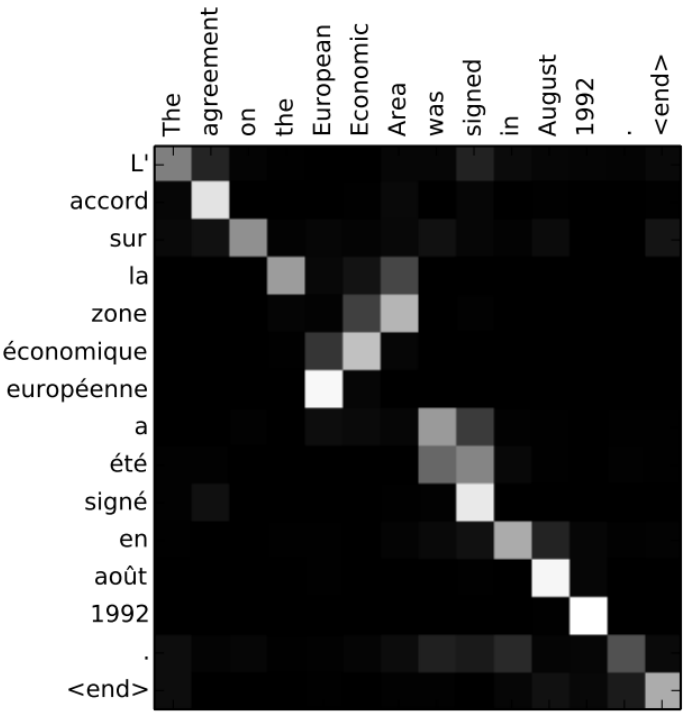

Figure 1: Example of an attention map from a transformer model during English-French translation, with grayscale intensity reflecting the model's focus on word pairings. Attention in transformers is computed through a high-level process that assigns importance to different parts of the input when predicting each word in the output, using a mechanism that emphasizes relevant information and diminishes the less important [8].

The journey from the earliest statistical models to the neural network-based architectures, with an emphasis on capturing the sequential nature and contextual nuances of language, has been driven by the need to model human language more naturally and effectively. The progression through various models highlights not only technical advancements but also a deeper understanding of language representation and processing. This path of development leads us to the introduction of transformer models, which I will explore in the next section.

1.2.2 Transformers

Transformers, introduced by Vaswani et al. in their 2017 paper "Attention Is All You Need" [9], represent a paradigm shift in how sequence data is processed in machine learning models, particularly for NLP tasks. Their unique architecture allows them to handle long-range dependencies with greater

efficiency than previous RNN and LSTM models. The key innovation in transformers is the self-attention mechanism, which enables the model to weigh the significance of different parts of the input data independently of their sequential position. It is a version of the previously discussed attention mechanism where the attention of each word in a sentence is calculates for all other words in the same sentence,

The original transformer model comprises two primary components: the encoder and the decoder. The encoder generates a vector representation of the model's inner state, and the decoder translates this to a probability distribution over all words Each consists of a stack of identical layers **??**, with each layer containing two sub-layers for the encoder and three for the decoder. The sub-layers include multi-head self-attention mechanisms and position-wise fully connected feed-forward networks.

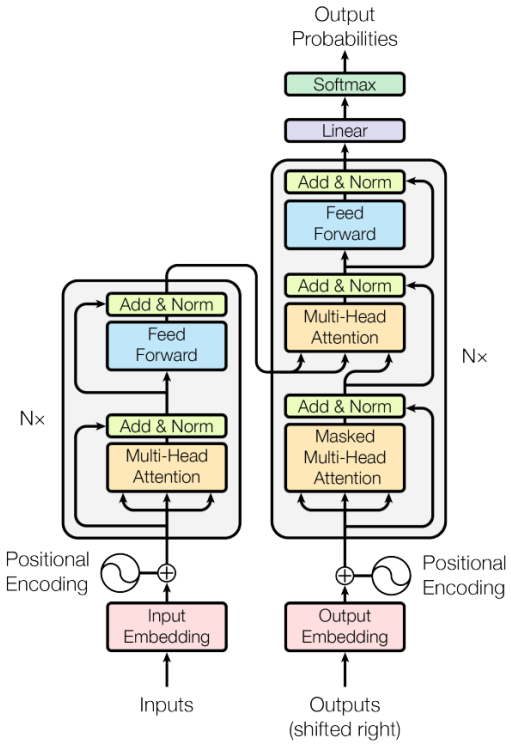

Figure 2: Transformer architecture[9]

The self-attention mechanism in neural networks, particularly in sequence processing models, is designed to dynamically weigh the significance of different elements in a sequence for each element of that sequence. This mechanism is implemented through the computation of three sets of vectors for each element in the input sequence: queries (Q), keys (K), and values (V). These vectors are derived from the input data using learned linear transformations [9].

1. Queries: These are vectors that represent the current element in the sequence that I are trying to understand or focus on. The query essentially asks which elements in the sequence are most relevant.

2. Keys: These vectors correspond to all elements in the sequence, including the current element. They act like identifiers for each element. The compatibility or relevance of each sequence element with the query is determined by comparing the query with all the keys.

3. Values: These are also vectors corresponding to all elements in the sequence. After determining the relevance of each element (through the query-key comparison), the values provide the actual content that I want to focus on. They carry the information from the input data that should be used in the output representation of the current element.

The attention function is mathematically represented as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

In this equation, QK^T represents the dot product between the query and key vectors, which

measures their compatibility. The scaling factor $\sqrt{d_k}$, where d_k is the dimension of the keys, is used to normalize the dot products, preventing excessively large values that can lead to unstable gradients in the training process. The softmax function is then applied to these normalized dot products to create a distribution of weights that sum up to one. These weights are used to compute a weighted sum of the values, resulting in the final output of the attention mechanism for the current element. This process is repeated for each element in the sequence, allowing the model to consider the entire sequence when constructing the representation of each element.

The concept of "Multi-Head Attention" is a significant enhancement to the self-attention mechanism [9]. It allows the model to process information in multiple representation subspaces at different positions simultaneously. This is achieved by creating multiple "heads" in the attention mechanism, each with its own set of linear projections for queries, keys, and values.

In more detail, for each head (totaling h heads), the queries (Q), keys (K), and values (V) are transformed using distinct, learned linear projections. This means that for each head i, there are three separate weight matrices: W_i^Q for queries, W_i^K for keys, and W_i^V for values. These matrices are parameters of the model, learned during the training process. They enable each head to focus on different aspects or parts of the input sequence, as they project the input vectors into different subspaces.

Once the queries, keys, and values are projected, the attention function is applied to each set of projections in parallel. This results in d_v-dimensional output values from each head. These output values from all the heads are then concatenated together. This concatenated vector captures a diverse range of features or patterns from the input, as each head may focus on different parts of the input sequence.

Finally, this concatenated vector is transformed again using another learned linear transformation, represented by the weight matrix W^O. This final transformation is crucial for combining the information from all the heads into a unified output that can be used in subsequent layers of the model.

The mathematical representation of Multi-Head Attention is as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

In these equations, the W_i^Q, W_i^K, and W_i^V are the individual weight matrices for each head that transform the queries, keys, and values respectively. W^O is the weight matrix that combines the outputs of all the heads. These matrices are essential for the model to capture and integrate a wide range of information from different parts of the input sequence, enhancing its ability to understand complex patterns and relationships [9].

In addition to the attention mechanism, each layer of the transformer model incorporates a fully connected feed-forward network (FFN). This FFN is applied independently but identically at each position. The FFN consists of two linear transformations with an activation function in between. Specifically, the activation function used here is the Rectified Linear Unit (ReLU). The ReLU function is defined as $\max(0, z)$, where z is the input to the function. It essentially sets all negative values in z to zero, allowing only non-negative values to pass through [9].

The feed-forward network can be represented by the following equation:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{6}$$

In this equation: - W_1 and W_2 are the weight matrices for the two linear transformations. They scale and transform the input data. - b_1 and b_2 are the bias vectors for the two linear transformations. They are added to the output of the weight matrix multiplication to shift the result.

The combination of linear transformations and the ReLU activation function allows the network to learn complex patterns in the data.

Transformers process data without an inherent understanding of the sequence or order of that data. This is a limitation when dealing with sequential data like text, where the position of each word is crucial for understanding. To overcome this, transformers add 'positional encodings' to their input embeddings at the beginning of both the encoder and decoder sections [9]. These positional encodings

are designed to have the same dimensionality (d_{model}) as the embeddings, which allows them to be directly added together [9].

The positional encoding for a particular position in the sequence (pos) and dimension (i) is calculated using sine and cosine functions. For even dimensions (represented by $2i$), the sine function is used:

$$\text{PE}_{(\text{pos},2i)} = \sin\left(\text{pos}/10000^{2i/d_{\text{model}}}\right) \tag{7}$$

For odd dimensions (represented by $2i+1$), the cosine function is used:

$$\text{PE}_{(\text{pos},2i+1)} = \cos\left(\text{pos}/10000^{2i/d_{\text{model}}}\right) \tag{8}$$

In these equations, pos refers to the position in the sequence (like the position of a word in a sentence), and i is a particular dimension within the embedding space. This approach allows the model to factor in the order of the data, enhancing its ability to understand and generate sequential information like natural language. Using sine and cosine in the calculations makes the encoding positional in nature. This allows the model to more easily understand the relative position in the sequence of each input vector.

Transformers offer several key advantages over previous sequence modeling approaches:

1. Parallelization: Unlike RNNs and LSTMs, which process data sequentially, transformers allow for much more parallelization, leading to significant speedups in training and inference.

2. Handling Long-range Dependencies: The self-attention mechanism enables the model to consider the entire sequence of inputs simultaneously, making it more effective at capturing long-range dependencies in the data.

3. Flexibility and Generalization: Transformers have demonstrated remarkable flexibility, being effective not only in NLP tasks such as machine translation, question answering, and text summarization, but also in areas like image recognition and generative tasks in computer vision [9].

1.2.3 Multimodality in Machine Learning

In the context of machine learning, the term "modality" refers to the type or form of data input or output by a model. Common modalities include text, images and audio, each characterized by distinct data structures and features [4].

Multimodal learning involves the integration and processing of data from multiple modalities. It aims to build models that can understand and relate information across these different forms, capitalizing on the strengths and compensating for the weaknesses of each individual modality [14].

Transformers are able to perform cross-modal attention, wherein the self-attention mechanism is adapted to attend not just within the same modality, but across different modalities. This works because the representation of each modality is a numeric vector, allowing them to be treated in the same way. For example, in a text-and-image model, the attention mechanism can enable the model to focus on specific words in the text while simultaneously attending to relevant regions in the image.

Unlike early multimodal systems that stitched together separately trained unimodal models, modern multimodal transformers are often trained end-to-end. This unified training allows for more seamless integration of information across modalities and better optimization towards the task at hand.

1.2.4 Modality Fusion

Modality fusion is the process of combining multiple modalities to produce a unified representation [5]. Fusion methods can be broadly categorized into three types: early, mid, and late fusion. Each method has its specific characteristics, advantages, and challenges [14].

Early fusion involves combining features from different modalities at an early stage before feeding them into the learning model. This means raw data from each modality is transformed into a feature vector, and these vectors are concatenated or otherwise combined before any significant processing occurs.

Mathematically, if $X_1, X_2, \ldots, X_n$ represent feature sets from n different modalities, early fusion may involve an operation like concatenation:

$$X_{\text{early fused}} = [X_1; X_2; \ldots; X_n] \tag{9}$$

Pros: 1. Simplicity: Early fusion is conceptually simple and straightforward to implement. 2. Feature Interactions: It allows for the interaction of features from different modalities at the earliest stages, which can be beneficial when these interactions are important for the task.

Cons: 1. Feature Dominance: One modality may dominate the feature space, overshadowing important features from other modalities. For example by having a much greater magnitude. 2. Computation: The transformer model's time complexity is quadratic with respect to the input length. Combining inputs early results in more layers having to deal with longer input lengths.

Late fusion is at the other side of the spectrum. It occurs at the final stage of the process, where each modality is processed independently, and the results are combined only at the decision stage.

For late fusion, separate models are trained independently for each modality. The outputs $D_1, D_2, \ldots, D_n$ (like class scores or probability distributions) are then combined for example with concatenation:

$$D_{\text{late fused}} = [D_1; D_2; \ldots; D_n] \tag{10}$$

Pros: 1. Modality Independence: Each modality can be processed using the most suitable model and training dataset.

Cons: 1. Limited Interaction: Fails to capture complex interactions between modalities. 2. Decision Alignment: Requires alignment of decisions, which can be challenging if the outputs are not directly comparable.

Mid fusion happens at an intermediate level of processing and can be seen as a compromise between early and late fusion. Here, each modality is processed separately to an extent, and their representations are fused before the final decision-making layers of the model.

In mid fusion, separate feature extractors first process each modality. The extracted features $R_1, R_2, \ldots, R_n$ are then combined, often using methods like concatenation:

$$R_{\text{mid fused}} = [R_1; R_2; \ldots; R_n] \tag{11}$$

Pros: 1. Balance: Offers a balance between preserving modality-specific features and enabling interactions among modalities. 2. Flexibility: Allows different modalities to be processed using their optimal architectures.

Cons: 1. Complexity: More complex than early fusion, requiring careful design to effectively combine intermediate representations.

Due to the added complexity, mid fusion has been studied less in practise.

1.3 Related works

1.3.1 T5: Text-To-Text Transfer Transformer

The Text-To-Text Transfer Transformer (T5) model, as introduced by Raffel et al. [10], stands as a paradigmatic example of transformer-based language models. It marks a transformative development in natural language processing (NLP), integrating diverse NLP tasks within a unified text-to-text framework.

Central to T5's design philosophy is the concept of formulating every NLP challenge as a text-to-text problem. This paradigm entails interpreting both inputs and outputs exclusively as textual strings. This approach diverges notably from traditional NLP models, which typically employ distinct architectures and frameworks tailored to specific tasks such as text classification, summarization, or translation. By consolidating these varied tasks into a cohesive model structure, T5 streamlines the application of transfer learning in NLP.

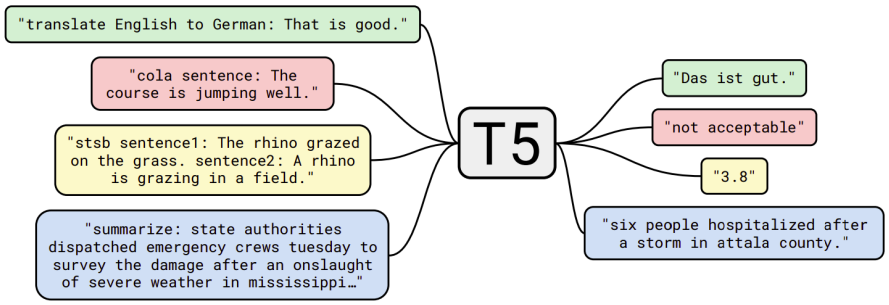

Figure 3: Diagram of T5. Every task is cast as a sequence to sequence task [10]

The T5 architecture, inspired by the seminal transformer model of Vaswani et al., incorporates an encoder and decoder, each composed of multiple transformer blocks. T5 distinguishes itself in its pre-training and fine-tuning methodologies [10].

Pretraining of T5 is done through supervised learning where the input is a sentence with some words blanked out and labels beeing these words. This give T5 a base understanding of language. Fine-tuning involves adapting this pre-trained model to specific tasks, reinterpreting all tasks as text sequence problems. For example, a classification task, typically seen as label prediction, is reframed to predict textual outputs like "positive" or "negative."

T5 has exhibited exceptional efficacy across a multitude of NLP tasks. Its success can be attributed to several factors:

- Unified Framework: By transforming all tasks into a text-to-text format, T5 facilitates the application of transfer learning across diverse tasks, enhancing performance and simplifying the modeling of various NLP challenges.

- Comprehensive Pre-training: The span corruption technique in pre-training equips T5 with an in-depth, contextual understanding of language, advantageous across different tasks.

- Scalability: With training available in sizes ranging from small to extra-large, T5 offers versatility to meet various computational and application needs.

1.3.2 Vision Transformer

The Vision Transformer (ViT) [11] marks a significant advancement in the application of transformer models, traditionally used in NLP, to the domain of computer vision. Introduced by Dosovitskiy et al., ViT demonstrates that pure transformer models, when pre-trained on sufficiently large datasets, can achieve state-of-the-art results in image classification tasks, challenging the dominance of Convolutional Neural Networks (CNNs) in this field.

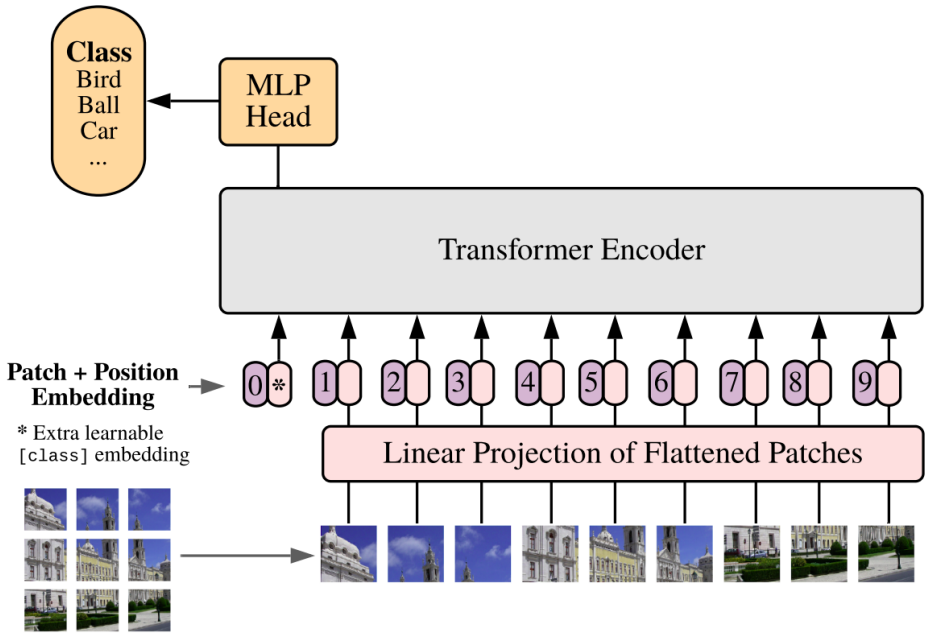

Figure 4: ViT architecture [11]

Transformers, designed initially for sequential data like text, need to be adapted for processing images, which are inherently two-dimensional. ViT accomplishes this by treating an image as a sequence of fixed-size patches, analogous to how words (or tokens) are treated in NLP tasks.

The key idea in ViT is to split an input image into a sequence of smaller image patches as can be seen in 4. Each patch is then flattened and linearly embedded (similar to word embeddings in NLP). This process can be represented as:

$$x_p = [x_p^1 E, x_p^2 E, \ldots, x_p^N E] + E_{\text{pos}}, \tag{12}$$

where x_p^i denotes the i-th image patch, E is the embedding matrix for the patches, N is the total number of patches, and E_{pos} represents the positional embeddings added to maintain spatial information. The sequence of embedded patches and class token is passed through a standard transformer encoder, as originally proposed by Vaswani et al. The transformer encoder consists of alternating layers of multi-head self-attention and MLP blocks, along with layer normalization:

$$\text{Transformer Encoder} = \text{LN}(\text{MHSA}(\text{LN}(\text{MLP}(\cdot)))) \tag{13}$$

where LN, MHSA, and MLP stand for Layer Normalization, Multi-Head Self-Attention, and Multi-Layer Perceptron (a single layer in a standard neural network), respectively.

1.3.3 Pooling Mechanisms in Deep Learning

Pooling is an essential technique in deep learning, commonly used in a variety of architectures, including Vision Transformers (ViT) [11]. It reduces the model dimensionality by aggregating inputs. Initially a key feature of convolutional neural networks (CNNs) for image processing, pooling methods have been adapted for use in models handling sequences of tokens, such as transformers. These operations are crucial for reducing the size of the data, extracting significant features, and helping models to become less sensitive to certain changes in the data. [12].

Mean pooling is a specific type of pooling operation. It calculates the arithmetic average of a group of values. In the case of transformers like ViT, which are advanced models used in multimodal Natural Language Processing (NLP) involving text and images, mean pooling is used on a sequence of token embeddings (or patch embeddings for ViT). This process results in a single, comprehensive representation vector.

The equation for mean pooling is represented as follows:

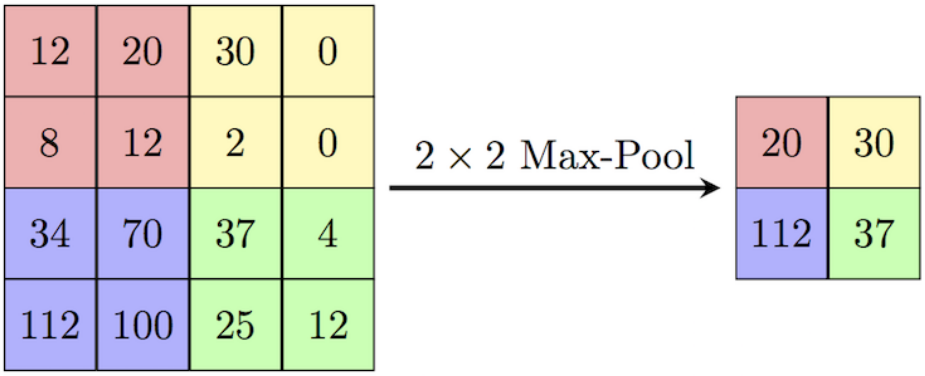

Figure 5: Max pooling example [25]

$$v_{\text{mean}} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{14}$$

In this equation, v_{mean} is the result of the mean pooling operation. The term x_i represents the embedding of the i-th token or patch, where an embedding is a form of representation that captures the essence of the token or patch in a numerical form. N stands for the total number of tokens or patches. Mean pooling is an efficient method that captures the average feature of the entire sequence, but it might lose some details about individual differences within the sequence.

Max pooling, another concept borrowed from the CNN domain, selects the maximum value from a set of values. When applied to the sequence of embeddings in transformers, max pooling focuses on capturing the most salient features:

$$v_{\text{max}} = \max_{i=1}^{N} x_i \tag{15}$$

Max pooling is particularly useful for capturing the most dominant feature in a set, but, like mean pooling, may miss out on other informative aspects of the data. An example of max pooling can be seen in Figure 5.

Attention-based pooling is more sophisticated pooling technique. It weighs different parts of the input not uniformly (as in mean pooling) or solely based on the maximum value (as in max pooling), but based on a learned attention mechanism. In essence, attention-based pooling allows the model to learn which parts of the sequence are more important for a specific task:

$$v_{\text{attention}} = \sum_{i=1}^{N} \alpha_i x_i \tag{16}$$

Here, α_i are weights learned through an attention mechanism, which determine how much focus should be given to each token or patch x_i in the sequence. This form of pooling is dynamically adaptive, enabling the model to focus on different parts of the input for different instances or tasks [12]. An exmple can be found in Figure 6.

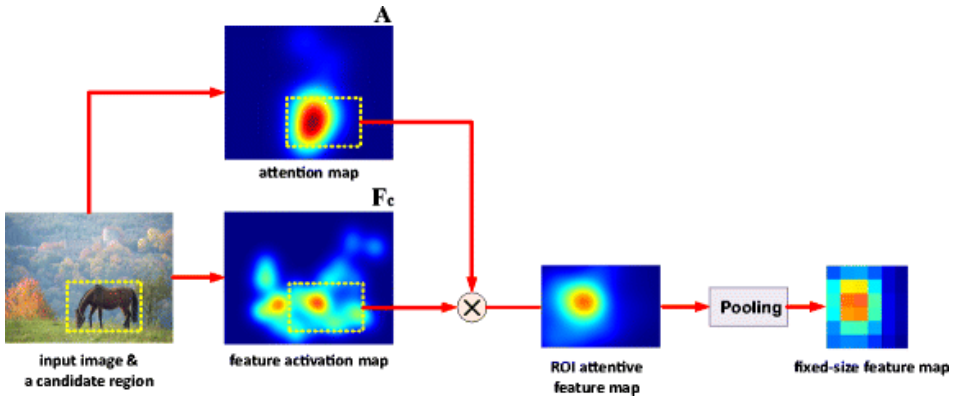

Figure 6: Attention pooling visualized [13]

While discussed here in the context of ViT, these pooling techniques are modality-agnostic and can be applied to any sequence of tokens - whether in NLP (e.g., word or subword tokens), audio processing (e.g., spectral or temporal features), or other sequence-based applications. The choice of pooling method often depends on the specific requirements of the task and the nature of the input data:

- Mean Pooling: Useful for when an overall average representation is needed, such as in document classification or average image style representation. - Max Pooling: Ideal for tasks requiring the capture of peak or dominant features, such as detecting the most significant sound in an audio clip or identifying critical features in an image. - Attention-Based Pooling: Best suited for contexts where the relevance of different parts of the input can vary significantly, such as in machine translation, contextual image captioning, or complex scene understanding in computer vision.

1.3.4 Evaluating Multimodal Capabilities

In addition to text-based benchmarks, the exploration of multimodal models demands datasets that combine various forms of data, such as text and images or videos.

1. MS COCO (Common Objects in Context): Primarily used for image recognition, segmentation, and captioning tasks, MS COCO is renowned for its large-scale dataset containing diverse everyday scenes with common objects in context. It includes over 300,000 images and 2,500,000 image captions, making it a comprehensive resource for training and evaluating image-captioning algorithms [15].

2. VQA (Visual Question Answering): This dataset challenges models to answer open-ended questions about images. It contains images paired with question-answer pairs, where the questions range from simple identification ("What color is the cat?") to more complex queries requiring reasoning about the scene [16].

Figure 7: VQA example [16]

In the realm of both text and multimodal tasks, various metrics have been developed to evaluate different aspects of model performance:

1. BLEU (Bilingual Evaluation Understudy): Commonly used for assessing machine translation quality, BLEU measures how many words and phrases in the model's output match a reference translation. BLEU scores range from 0 to 1 (or sometimes expressed as 0 to 100), where higher scores indicate better matches with the reference [18].

2. CIDEr (Consensus-based Image Description Evaluation): Specifically designed for image captioning, CIDEr evaluates how well the captions of a generated image align with human-written captions. It considers the consensus among a set of reference sentences and computes a score based on the commonality of n-grams between the generated and reference captions [19].

These metrics and datasets are integral to advancing the field of multimodal machine learning, offering benchmarks to assess the progress and guide the development of more sophisticated, contextually aware models.

One primary challenge in evaluating multimodal models is ensuring that the model genuinely understands and integrates all modalities involved, rather than over-relying on one. For instance, a model trained on a video dataset might perform well in classification tasks by mainly leveraging visual cues, even if its understanding of the accompanying audio is poor. This scenario makes it difficult to assess the model's true multimodal comprehension.

Understanding the complex interactions between different modalities and designing tasks that accurately capture this interplay is challenging. A model might learn superficial correlations between modalities rather than a deeper, semantic understanding. For example, in image-text tasks, a model could correlate certain objects in images with text descriptions without truly understanding the context or the nuances of either modality.

Subjectivity in evaluation, especially for generative tasks like image captioning or storytelling from images/videos, adds another layer of complexity. Metrics like BLEU or CIDEr might not fully capture the richness or relevance of the generated descriptions relative to the visual content. Human judgment often remains a crucial but subjective and challenging-to-scale component in evaluating these models.

To mitigate these challenges, there is a need for more comprehensive, rigorously designed tasks and benchmarks that can probe deeper into a model's ability to understand and integrate multiple modalities. This includes tasks that require reasoning, inferring, and making judgments based on combined modalities, rather than just identifying or matching content across them.

Developing balanced datasets that equally represent and challenge the model on all involved modalities is crucial. This approach ensures that a model's proficiency in one modality doesn't overshadow its weaknesses in another.

1.3.5 T5X and Flaxformer

T5X [20] is an advanced training and research framework specifically designed for the T5 model (Text-to-Text Transfer Transformer) and other transformer-based models. Its purpose is similar to that of Tensorflow, but with a focus on transformer models. Developed as a collaborative project between Google Research and DeepMind, it builds on the architectural principles of T5, which itself was a milestone in the field of natural language processing (NLP). T5X's development was driven by the need for a more modular, flexible, and scalable approach to training large transformer models.

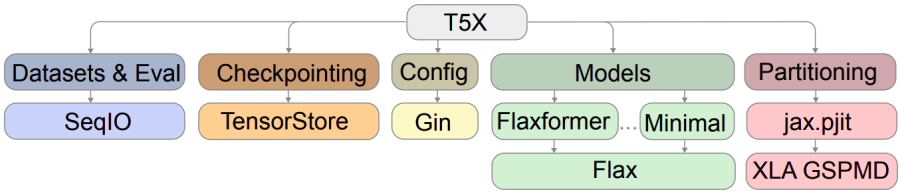

Figure 8: Overview of T5X [20]

At its core, T5X leverages the power of JAX, a numerical computing library. JAX's functional programming model offers both efficiency and ease of use, which are crucial for handling the computational demands of large-scale transformer models [20]. The T5X framework capitalizes on these advantages by providing a structured yet flexible pipeline for model training, evaluation, and inference.

One of the key design principles of T5X is its modular architecture. This modularity facilitates the customization and extension of various components such as input pipelines, model architectures, and optimization strategies. Researchers and practitioners can easily experiment with different configurations and tailor the framework to their specific needs, making T5X a versatile tool in both academic and industrial settings. An overview of T5X can be seen in Figure **??**.

FlaxFormer, a high-level library used by T5X to define models in python code. It offers an additional layer of abstraction for working with transformer models. Designed to simplify the implementation and experimentation with various transformer architectures [20].

The primary goal of FlaxFormer is to reduce the complexity inherent in designing and training transformer models. It achieves this by providing a suite of pre-defined components such as attention mechanisms, feed-forward networks, and encoder-decoder structures that are common in transformer models. These components are highly customizable, allowing users to modify and combine them in novel ways to create unique model architectures.

FlaxFormer also emphasizes interoperability with other tools and libraries in the JAX ecosystem. This interoperability ensures that researchers can integrate advanced optimization techniques, hardware accelerators (like TPUs and GPUs), and other JAX-based utilities seamlessly with their transformer models.

With its modular design, the model implementations in T5X can be flexible. Layers and modules can be written directly with Flax or using a higher-level library such as FlaxFormer. The relationship between T5X and FlaxFormer is symbiotic: while T5X provides the overarching framework and infrastructure for training transformer models, FlaxFormer offers the building blocks and abstractions for model design and implementation.

This integration allows for a streamlined workflow where researchers can design complex transformer architectures with FlaxFormer and then deploy them efficiently for training and evaluation using T5X's robust infrastructure. This combination harnesses the strengths of both frameworks, making it simpler to develop, fine-tune, and scale transformer models for a wide range of NLP tasks.

Both T5X and FlaxFormer are open-source projects, which underscores their role in the broader AI and ML community. Being open-source encourages a collaborative approach to development and innovation. It allows researchers and developers from around the world to contribute to their codebases, propose enhancements, and share their adaptations of these frameworks for different tasks and challenges in NLP and beyond. The open-source nature also ensures transparency, fostering trust and wider adoption in both academic research and industry applications.

2 Method

In this study, I developed a multimodal learning model based on the T5 architecture. My objective was to create a framework capable of integrating and understanding multiple data modalities, including text, images, and audio. While the model was implemented to allow a variable number of modalities, the experiments were run with only text and image. Building a flexible model allow for future experiments with more modalities, but time was limited in this work.

2.1 Selection of Evaluation Dataset

To ensure an effective evaluation of my model's multimodal capabilities, I established specific criteria for selecting the dataset. These criteria were:

1. Challenge for Multimodal Understanding: The dataset must present a significant challenge that necessitates a deep understanding of various modalities.

2. Prevalence in Prior Research: Preference was given to datasets commonly used in previous studies to enable effective benchmarking of my results.

3. Compatibility with YouTube Video Links: Considering my available tools for processing YouTube video links internally at Google, datasets with a focus on such links were prioritized.

4. Open-Source Licensing: In line with my open-source commitment, I looked for datasets licensed under MIT or similar open-source licenses.

My approach to identifying a suitable dataset involved a systematic review of existing literature and multimodal datasets. I rigorously assessed each potential dataset against the above criteria, paying special attention to the diversity and complexity of the modalities it encompassed.

2.2 Architectural Design of the Model

I chose the Text-To-Text Transfer Transformer (T5) model as my base. This because the model has already been pretrained to get basic language knowledge, and its weights are available as open source [10] to allow for reproducibility. I modified this architecture to accommodate additional modalities, integrating them at specific layers in the encoder by concatenating to the activations between each layer. An overview can be seen in Figure 9

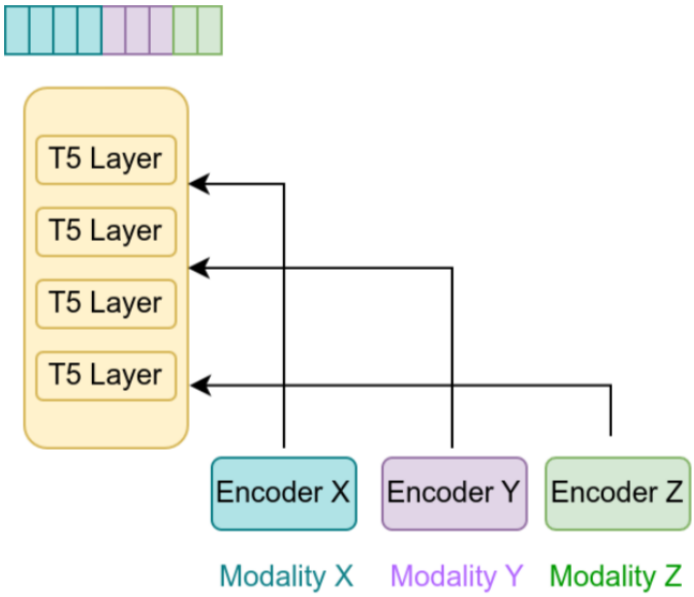

Figure 9: Overview of the Where to Fuse encoder.

The architecture allows for the concatenation of different modalities at predetermined layers. This is defined in a configuration file, enabling easy adjustments. For instance,

```
fusion_spec = { 0:  ["text_title", "image"], 7:  ["audio"], 8:  ["text_description"]}
```

specifies the modalities to be fused at each layer.

To handle varying modalities efficiently, I implemented different pooling techniques - mean, max, and attention-based. This allows for dimension reduction of each modality, adjusting the token sequence length as needed. For instance, with `pool_type="attention"` and `pooled_modalities={"image": 16, "audio": 4}`, the model can dynamically adjust the sequence lengths for image and audio modalities. The integration of pooling in the Where to Fuse encoder can be visually be seen in 10.

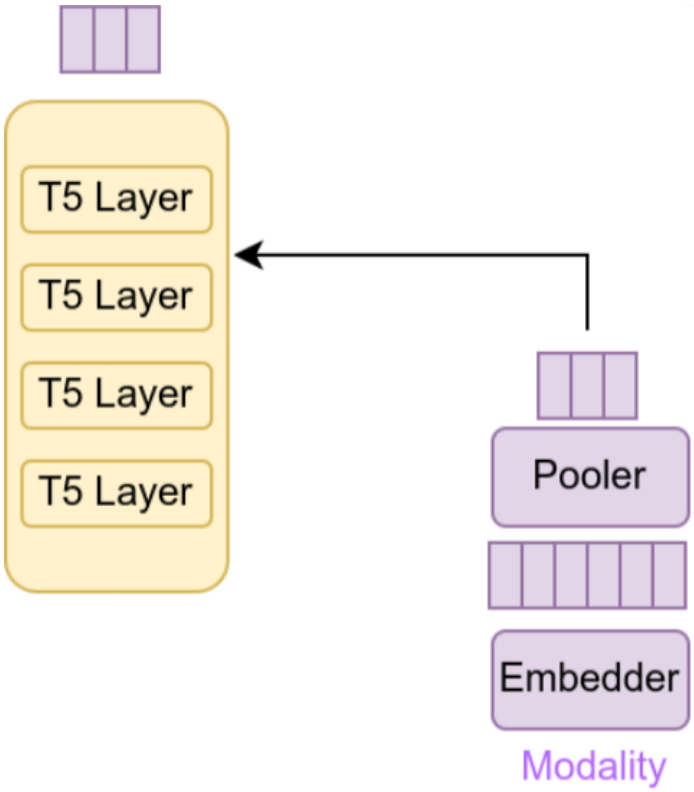

Figure 10: Where to Fuse encoder with pooling.

The most common modalities in multimodal learning are image and text. To ease the use when using these modalities I implemented support for inputing images as raw pixel values, and text as a string (rather than vectors). The text tokenizer used for this is the same as the one used in T5 [10]. To support images as raw pixels, a Vision Transformer (ViT) model was implemented. This module encodes pixel values of 224x224 images into 14x14=196, 768-dimensional tokens. The encoded images are then processed like other modality tokens. To enhance efficiency, I developed a caching mechanism for the ViT calculations. This is done by running inference with ViT separately and storing them in a database. Running the full pipeline could then optionally be done by reading image embeddings from the database rather than calculating them on the fly.

2.3 Experimental Setup and Execution

The initial plan was to use the benchmarks found in the "Evaluation dataset" subsection of the "Result" section of this paper, but I opted not to due to time constraints. Instead, I focused on datasets that were quicker to implement. I chose the VQA datasets for my experiments and reported accuracy [16].

The experiments were conducted using T5X and monitored using Tensorboard. This setup allowed for efficient training and real-time performance monitoring. The question was fused in layer 0 of the Where to Fuse encoder (Figure 9) and the layer to fuse the image embeddings was varied across

experiments (Figure 11). I also varied the size of the decoder by changing the number of transformer blocks.

A given configuration was then finetuned on a random slice (80%) of the VQA dataset. The remaining 20% was used for evaluation.

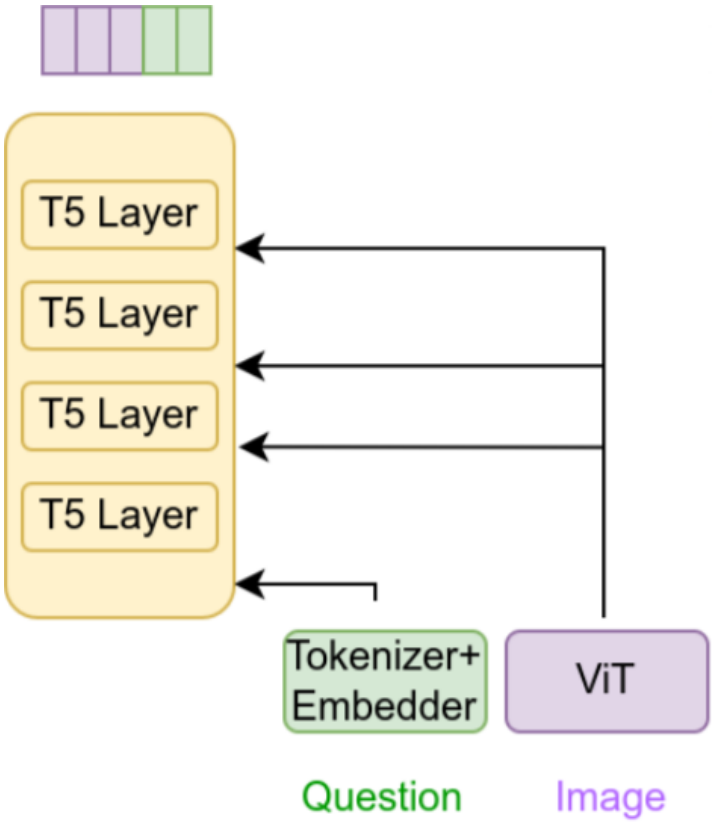

Figure 11: Where to Fuse encoder in the setting of the experiments.

3 Results

3.1 Evaluation dataset

This section provides a evaluation of a collection of video datasets, each scrutinized for its potential to test and enhance the multimodal capabilities of my language model. I systematically analyzed datasets based on several criteria: the complexity and necessity of multimodal understanding (good performance should not be attainable without understanding of all modalities), the extent of usage in prior research, the availability of YouTube video links, and the compliance with open-source licenses.

The datasets were first categorized based on their source, with a preference for those extensively comprising YouTube links due to my existing tools designed for processing such content. Licensing was the next crucial filter, with datasets under permissive licenses like MIT being highly favored to maintain the open-source nature of my project. The multimodality importance of each dataset was also a determining factor; datasets that inherently required the integration of multiple input types (e.g., audio, visual, textual) were given precedence.

Following a preliminary overview, I identified a subset of datasets that not only met my initial criteria but were also backed by published papers featuring ablation studies where models are evaluated without the access to one of the modalities. These studies are crucial as they illustrate the degradation in performance when one or more modalities are excluded, thereby validating the necessity of a multimodal approach. The following subsections showcases ablation studies of the datasets identified as the best suited according to my criteria.

3.1.1 VGGSound

The VGGSound [26] dataset, with its rich collection of audio-visual content, provided a formidable testbed for my multimodal model. The ablation study results (Figure 12) clearly demonstrated a marked improvement when both audio and visual cues were utilized in tandem. The models that only used one modality lagged by a significant margin in comparison to the multimodal ones, underscoring the synergistic potential of multimodal learning.

Result with only 1 modality

VGGSound (Top-1 Accuracy, %)	Audio	Video	Fusion
Chen *et al.* [21]	48.8	-	-
AudioSlowFast [29]	50.1	-	-
MBT [24]	52.3*	51.2*	64.1
Our Cross-Modal Attention Model	-	-	62.9±0.2
Our Modal-Independent Model	56.5±0.1*	49.7±0.2*	65.7±0.2
Our UAVM Model	**56.5±0.1**†	**49.9±0.2**†	**65.8±0.1**

Much better together!

Figure 12: Ablation study results from the VGGSound dataset, evidencing the comparative performance of unimodal versus multimodal models. Sources for these experiments can be found in [24].

3.1.2 AudioSet

Similarly, the AudioSet [27] dataset (Figure 13) offered insights into the superiority of multimodal learning. Ablation results pointed to a consistent pattern where the fusion of audio and visual data outperformed the unimodal approaches.

Result with only 1 modality

Full AudioSet (mAP)	Audio	Video	Fusion
GBlend [30]	32.4*	18.8*	41.8
Attn Audio-Visual [31]	38.4*	25.7*	46.2
Perceiver [14]	38.4*	25.8*	44.2
MBT [24] (w/ 500k training samples)	44.3*	32.3*	52.1
Our Cross-Modal Attention Model	-	-	**50.4±0.1**
Our Modal-Independent Model	45.5±0.0*	26.8±0.1*	48.1±0.1
Our UAVM Model	**45.6±0.0**†	**27.4±0.1**†	48.0±0.0

Better together

Figure 13: Results from the Full AudioSet dataset ablation study, highlighting the disparities in model performance between unimodal and multimodal configurations multimodal models. Sources for these experiments can be found in [24].

3.1.3 VALOR

The VALOR [23] dataset emerged as another top pick, with its comprehensive benchmarking of multimodal learning strategies. As illustrated in Figure 14, the multimodal approach significantly outstripped the unimodal one. The performance metrics clearly favored models that could seamlessly integrate and interpret both visual and auditory inputs.

Method	#Example	Mod	VALOR-32K
Group-A: pretrain with <10M examples			
ClipBert [71]	5.6M	V	-
Frozen [13]	6.1M	V	32.9/60.4/71.2
BridgeFormer [72]	5.5M	V	-
MILES [73]	5.5M	V	-
OA-Trans [74]	5.5M	V	-
Nagrani et al. [75]	1.03M	V+A	-
LF-VILA [76]	8.5M	V	-
VALOR$_B^-$ (Ours)	5.5M	V	43.3/70.3/80.0
VALOR$_B$(Ours)	6.5M	V+A	**67.9/89.7/94.4**

Figure 14: Performance metrics on the VALOR dataset, showcasing the enhanced efficacy of multimodal models over their unimodal counterparts. The reported numbers are from a retrieval task when 1, 5 or 10 picks are used to calculate recall. Blank entries are experiments not carried out by the paper. multimodal models. Sources for these experiments can be found in [23].

In conclusion, the datasets chosen for this study have demonstrated through rigorous ablation studies the indispensability of multimodal inputs for optimal performance. My systematic approach to dataset selection and analysis has laid a strong foundation for further advancements in multimodal language model development.

3.2 Where to fuse

Here I analyse the optimal fusion layer in my transformer-based model, applied to Visual Question Answering (VQA) tasks. My goal is to find the optimal layer for integrating image features to maximize model performance on a separate VQA evaluation set. To this end, I conduct a series of experiments where the text tokens are consistently fused at the initial layer, while image fusion is tested at various decoder layers. Because of the time limitation during this work, the experiments were only run once. All data and pretrained weights are open source, the results can be reproduced by implementing the model explained in the method section.

The extended set of experiments, each replicated five times with decoder layers ranging from 1, 2, 4, 8, to 12, is designed to investigate if the results carry over to varying decoder sizes. This approach allows for a more detailed examination of the trend in data, facilitating the identification of any patterns associated with the fusion layer's depth and the model's accuracy on the VQA tasks.

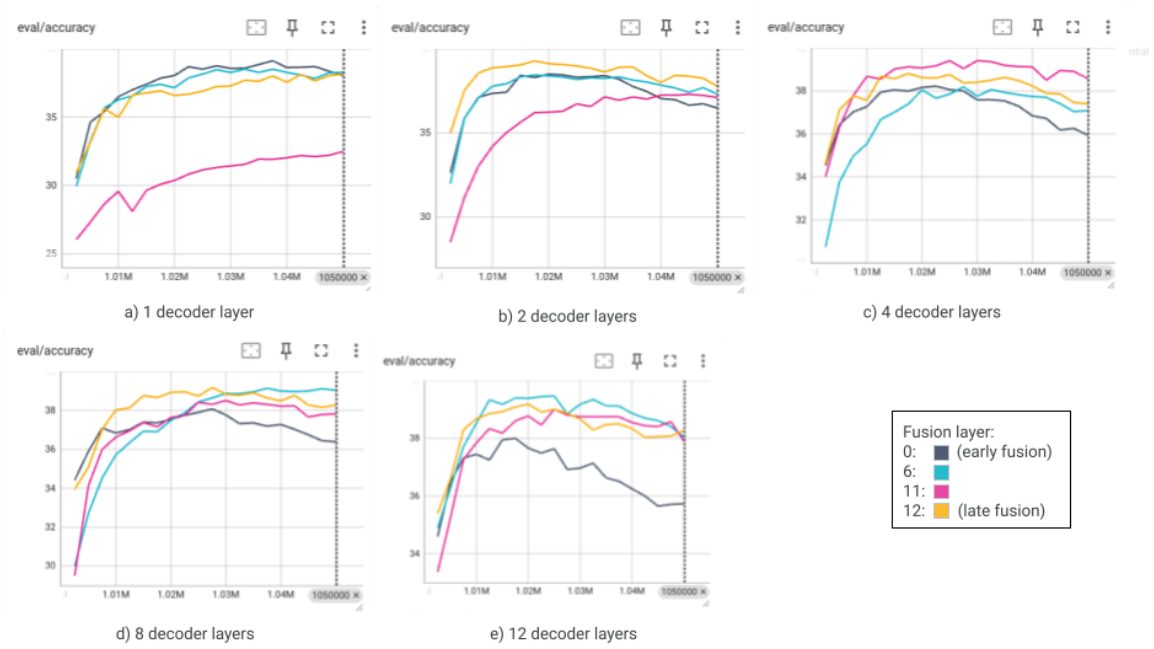

Figure 15: Evaluation accuracy (% of questions correclty answered by the model) of fine-tuning on VQA data across different fusion points. Experiments were conducted with varying numbers of transformer decoder layers (1, 2, 4, 8, 12).

Determining the most effective fusion point from my data requires a nuanced approach. To better discern trends, I present a comparative analysis in Figure 16, where I refine my evaluation by aggregating the outcomes of each experimental run into a single metric. I calculate this metric by considering either the highest accuracy achieved at any point during training (maximum accuracy) or the accuracy at the final step of training (last step accuracy). Standard practice typically emphasizes the latter; however, without hyperparameter optimization tailored to the specific length of training, the maximum accuracy metric may more accurately reflect the model's potential. In addition, I also run experiments for a greater number of different fusion points (0, 1, 3, 5, 7, 9, 11, 12), ensuring a more robust and insightful analysis of the fusion strategy's impact on VQA performance.

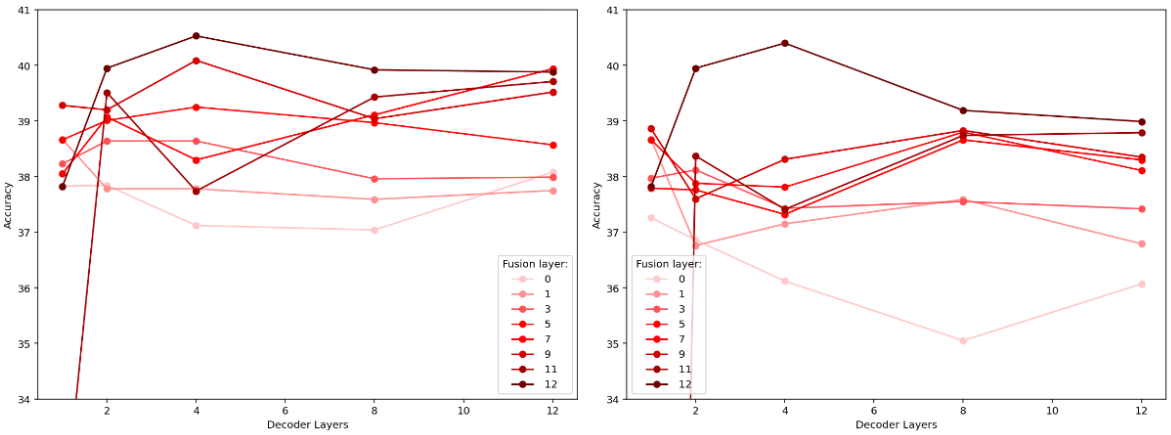

Figure 16: Comparative performance analysis using aggregated scores from multiple runs, utilizing either maximum (a) accuracy achieved during training or accuracy at the final training step (b). Accuracy is the % of questions correclty answered by the model

3.2.1 Late fusion generally benefit from increased decoder size

My analysis of Figure 16 reveals several key insights into the relationship between the timing of fusion and the number of decoder layers. Firstly, I observe a trend where models with a greater number of decoder layers tend to exhibit improved performance when the fusion occurs later in the network. This pattern is particularly pronounced in Figure 16 (b).

3.2.2 Late fusion with moderate decoder size is the best

Secondly, the experimental setup that yielded the highest performance featured late fusion paired with a model comprising 4 decoder layers. This is again in contrast to a prior hypothesis. Normally, machine learning models benefits from bigger models with more trainable parameters. Here a balanced number of decoder layers is optimal, not the maximum amount. This is not only seen by the fact that the maximum point is not with 12 decoder layers, but also that there is not ovbious trend in Figure 16 where more decoder layers results in better performance.

3.2.3 Early fusion get worse with increased decoder size

Furthermore, early fusion appears to be less effective as the number of decoder layers increases. One possible explanation of this decline in performance could be that the fusion of modalities upsets the models prior text understanding which it gained during pretraining. In this context, having more layers with only the text modality might be beneficial.

3.2.4 Early fusion better for small decoder sizes

In configurations with fewer decoder layers, early fusion tends to yield better results, although it looks like the correlation is not particularly strong. A speculation is that there is not enough trainable parameters in the decoder to gain image understanding capabilities during pretraining if the decoder is small. Because of this, the image needs to be fused earlier in the encoder to benefit from its learnable parameters.

3.2.5 Early fusion is slower

Finally, Figure 17 provides an interesting computational perspective, showing that later fusion results in reduced computation time. Given the quadratic time complexity of transformers in relation to input length, this finding aligns with expectations. Late fusion effectively shortens the input sequence for the majority of the transformer's layers, thus reducing the overall computational burden.

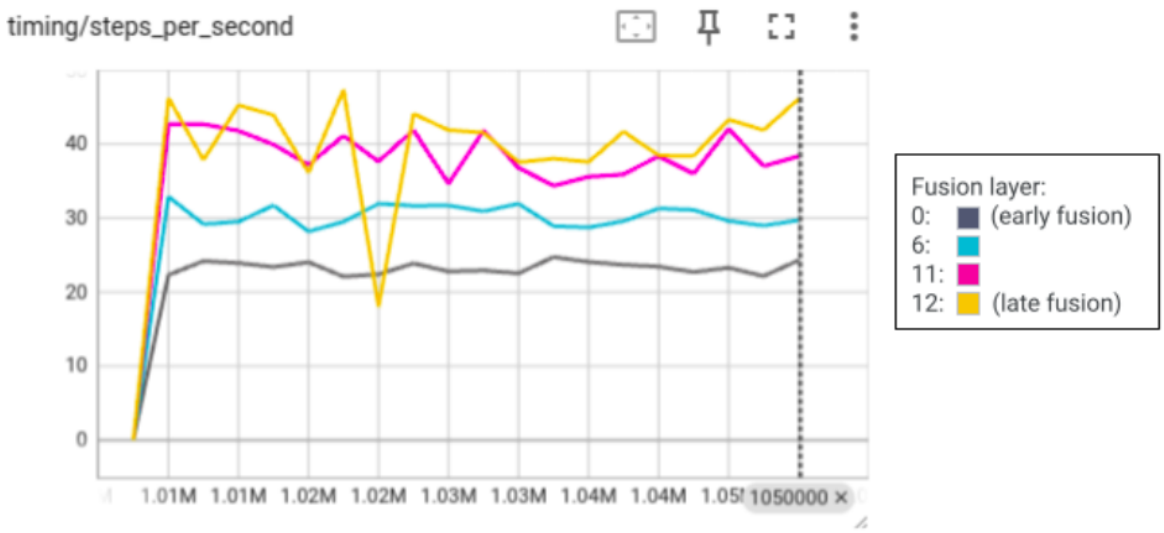

Figure 17: Training steps per second for various fusion strategies in a VQA task, indicating higher computational efficiency (more steps per second) with later fusion points (0, 6, 11, 12), consistent with the transformer model's reduced time complexity at shorter sequence lengths as more layers have more inputs for earlier fusion. The X axis represents finetuning steps.

3.3 Pooling

Building upon my previous exploration of the optimal layer for token fusion in a transformer-based model for Visual Question Answering tasks, I now extend my investigation to include the effects of token pooling on the fusion process. Pooling strategies are commonly employed in machine learning to reduce dimensionality and to abstract features, potentially leading to more efficient processing. Pooling is a very common strategy in practice. To make this work maximally useful for other researchers, I investigate if my findings still apply when pooling is used. The original model configuration utilizes 196 image tokens (ViT generates one token per patch), reflecting a fine-grained representation of the visual input. In this section, I experiment with two pooling techniques—mean pooling and attention pooling—to condense these image tokens into more compact representations of 14 and 1 token, respectively.

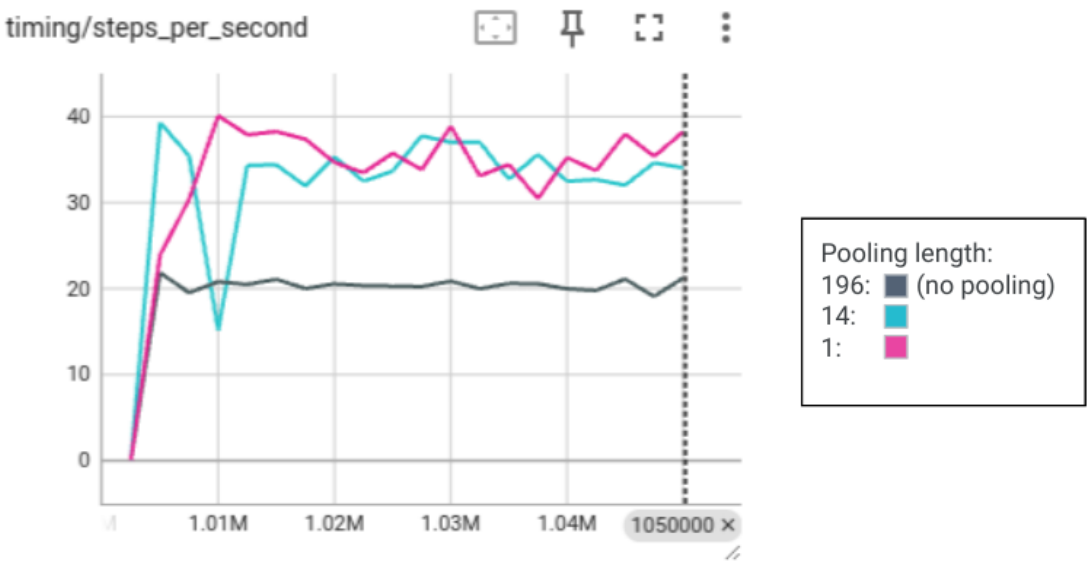

Figure 18: A graph depicting the steps per second for different pooling lengths when pooled with mean pooling in a early fusion model. The X axis represents finetuning steps. Note that the text tokens are not pooled.

3.3.1 Pooling reduces computation

The analysis presented in Figure 18 illustrates that the application of pooling techniques to the image tokens significantly reduces computational demands. As I transition from no pooling (196 tokens) through intermediate (14 tokens) to extensive pooling (1 token), there is a clear and consistent decrease in the computational resources required, as evidenced by the increased number of steps processed per second. This inverse relationship underscores the efficiency gains achieved through pooling, supporting the conclusion that by abstracting and compressing the image token representation, the model becomes more computationally efficient.

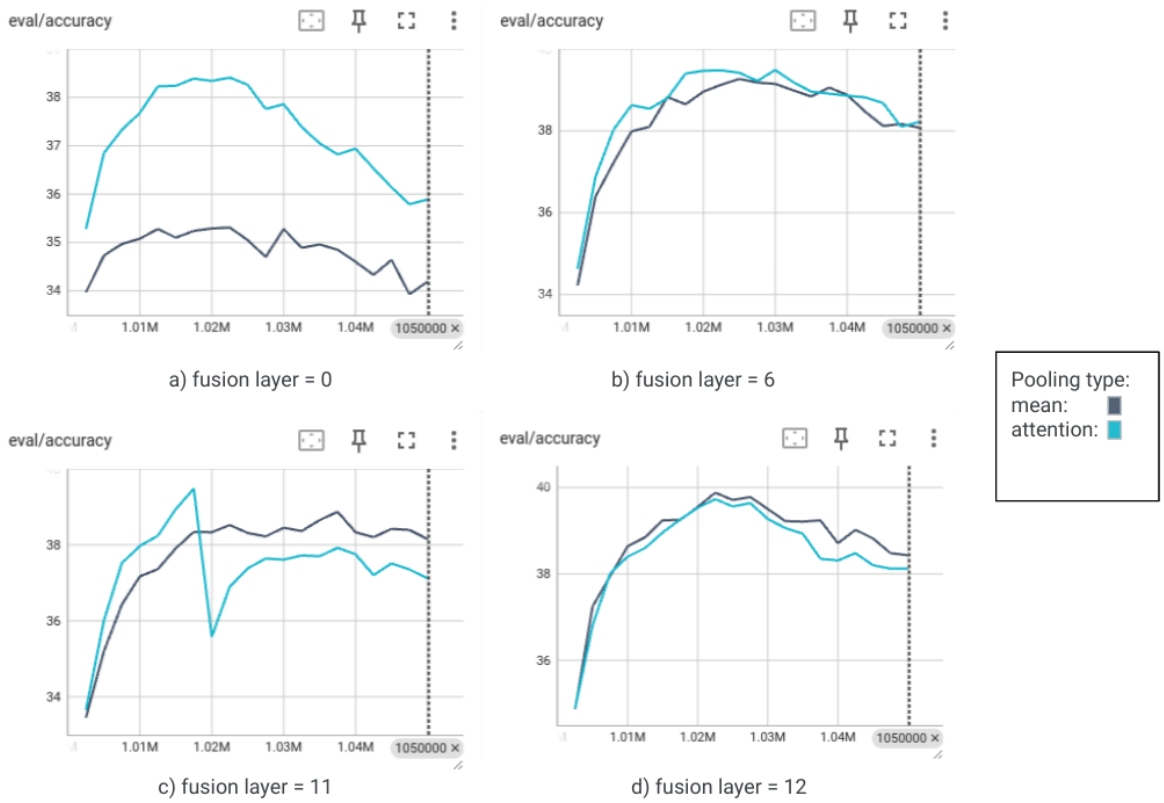

Figure 19: Comparative evaluation of mean and attention pooling methods across different fusion points with a decoder of 12 layers and pooling length of 14 tokens. % of questions correclty answered by the model is reported. (c) show unexpected behaviour around 20000 steps, this behaviour was repeated in 4 of 4 reruns. The X axis represents finetuning steps.

3.3.2 No one size fit all solution

Within the context of a 12-layer decoder configuration and a pooling length of 14 tokens, the comparative analysis of mean and attention pooling techniques reveals insights into their respective efficacies at different fusion points. As depicted in Figure 19, attention pooling demonstrates superior performance for early and mid fusion, while mean pooling is achieves higher accuracy for late fusion. This is especially apparent for early fusion in Figure 19 (a).

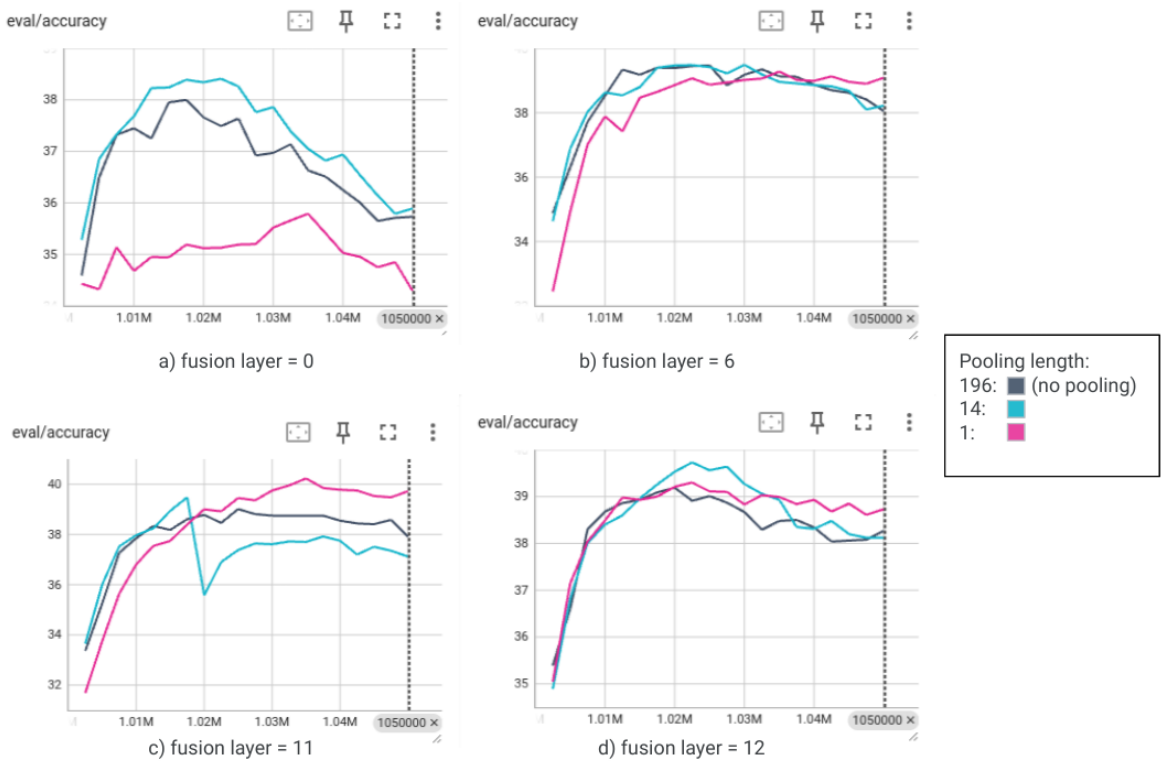

Figure 20: Performance comparison (% of questions correclty answered by the model) of different pooling lengths (196, 14, and 1 token) at various fusion points (0, 6, 11, 12) using attention pooling.The X axis represents finetuning steps.

3.3.3 Moderate pooling is optimal for attention pooling

Figure 20 presents a view of how varying pooling lengths affect model accuracy across different fusion points when employing attention pooling. Contrary to what one might expect, the models without pooling (196 tokens) do not yield the highest accuracy. Instead, it is the model with an intermediate pooling length of 14 tokens that consistently outperforms the others. One possible explanation for this is that the trainable parameters in the attention pooling mechanism contributes to the increased performance, this speculation is based on the trend in NLP where bigger models generally perform better [28].

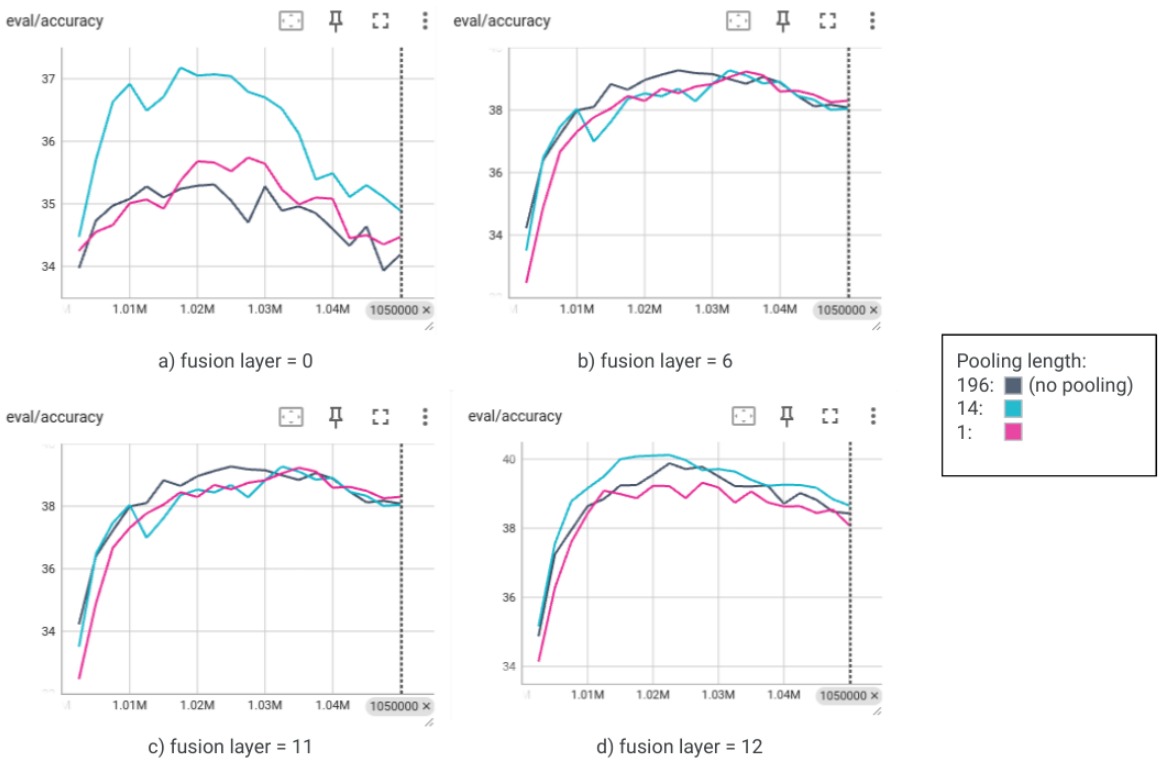

Figure 21: Evaluation of model accuracy (% of questions correclty answered by the model) using mean pooling across different pooling lengths (196, 14, and 1 token) and fusion points (0, 6, 11, 12). The X axis represents finetuning steps.

3.3.4 No pooling is best for mean pooling

The data visualized in Figure 21 reinforces the hypothesis that the added parameters of attention mechanisms can be beneficial. Unlike attention pooling, mean pooling does not introduce additional parameters to the model. Here, I observe that models with no pooling (196 tokens) exhibit relatively stronger performance when compared to the results seen with attention pooling in Figure 20. It could be speculated that the advantage of attention pooling may partially stem from the additional parameters that allow the model to learn more nuanced representations of the visual input.

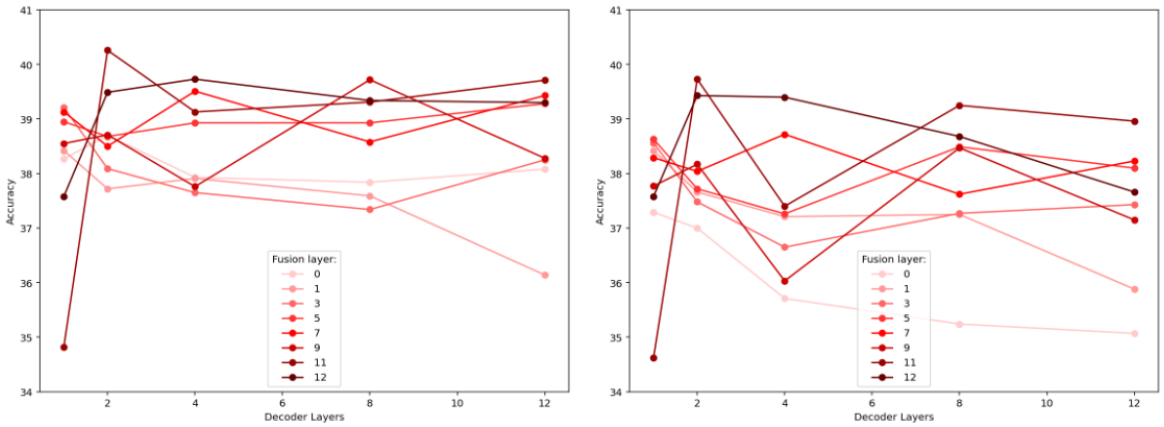

Figure 22: Comparative performance (% of questions correclty answered by the model) analysis using aggregated scores from multiple runs with attention pooling to 14 image tokens, utilizing either maximum (a) accuracy achieved during training or accuracy at the final training step (b).

3.3.5 Main findings from non-pooled set carry over

My non-pooled experiments' key outcomes are illustrated in Figure 16, while Figure 22 presents similar data for experiments employing attention pooling with 14 image tokens. This comparison reveals that the core findings from the non-pooled experiments are applicable even when pooling is implemented. In both scenarios, models with larger decoders benefit more from later fusion. There's no apparent advantage to having larger decoders, and the performance diminishes when early fusion is applied to an increasing number of decoder layers. However, the optimal setup slightly varies in the pooled context, favoring 2 decoder layers with fusion at layer 11, as opposed to 4 decoder layers with fusion at layer 12 in the non-pooled case.

4 Conclusion

This study presents a examination of multimodal fusion techniques within transformer models, which could help the enhancement of large language models in processing and understanding text and image. Through rigorous evaluation, I have delineated the effectiveness of various fusion strategies—early, late, and mid-fusion—within the context of Visual Question Answering (VQA) tasks.

I observed that standard early fusion techniques often underperform, particularly with larger decoder configurations. Conversely, late fusion, especially when paired with models featuring fewer decoder layers, emerged as the most effective approach in my VQA experiments. This configuration consistently outperformed others, striking an optimal balance between model complexity and the effective utilization of fused multimodal features.

In addition, my exploration into token pooling techniques discovered that while both pooling methods enhance computational efficiency, their impact on model performance varies depending on the fusion point within the network. Attention pooling excels in early fusion scenarios. Mean pooling, however, appears more advantageous at later fusion stages, suggesting that the choice of pooling strategy should be informed by the intended fusion layer. Furthermore, I show that my findings from non-pooled experiments transfer over to the pooled setting.

My systematic approach to dataset selection and analysis, particularly the focus on comprehensive benchmark datasets for video understanding has contributed valuable insights into the field of multimodal learning. I identify VggSound, Audioset and Valor as the most useful dataset to evaluate multimodal understanding in the video context.

References

[1] Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing. (Draft of January 7, 2023). [Copyright © 2023. All rights reserved.]

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org

[3] OpenAI. (n.d.). ChatGPT. Retrieved January 13, 2024, from

[4] Pichai, S., & Hassabis, D. (2023, December 6). Introducing Gemini: Our largest and most capable AI model. Google Blog. Retrieved January 13, 2024, from

[5] Tziafas, G., & Kasaei, H. (2022). Early or Late Fusion Matters: Efficient RGB-D Fusion in Vision Transformers for 3D Object Recognition. 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 9558-9565.

[6] Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. International Conference on Learning Representations.

[7] Sak, H., Senior, A.W., & Beaufays, F. (2014). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition.

[8] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate.

[9] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. Neural Information Processing Systems.

[10] Raffel, C., Shazeer, N.M., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P.J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. J. Mach. Learn. Res., 21, 140:1-140:67.

[11] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

[12] Chen, F., Datta, G., Kundu, S., & Beerel, P.A. (2022). Self-Attentive Pooling for Efficient Deep Learning. 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 3963-3972.

[13] Jiang, W., Zhao, Z., & Su, F. (2018). Weakly supervised detection with decoupled attention-based deep representation. Multimedia Tools and Applications, 77, 3261-3277.

[14] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. (2011). Multimodal Deep Learning. International Conference on Machine Learning.

[15] Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C.L. (2014). Microsoft COCO: Common Objects in Context. European Conference on Computer Vision.

[16] Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C.L., Parikh, D., & Batra, D. (2015). VQA: Visual Question Answering. International Journal of Computer Vision, 123, 4 - 31.

[17] Xiao, J., Shang, X., Yao, A., & Chua, T. (2021). NExT-QA: Next Phase of Question-Answering to Explaining Temporal Actions. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 9772-9781.

[18] Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. Annual Meeting of the Association for Computational Linguistics.

[19] Vedantam, R., Zitnick, C.L., & Parikh, D. (2014). CIDEr: Consensus-based image description evaluation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4566-4575.

[20] Roberts, A., Chung, H.W., Levskaya, A., Mishra, G., Bradbury, J., Andor, D., Narang, S., Lester, B., Gaffney, C., Mohiuddin, A., Hawthorne, C., Lewkowycz, A., Salcianu, A., Zee, M.V., Austin, J., Goodman, S., Baldini Soares, L., Hu, H., Tsvyashchenko, S., Chowdhery, A., Bastings, J., Bulian, J., García, X., Ni, J., Chen, A., Kenealy, K., Clark, J., Lee, S., Garrette, D.H., Lee-Thorp, J., Raffel, C., Shazeer, N.M., Ritter, M., Bosma, M., Passos, A., Maitin-Shepard, J.B., Fiedel, N., Omernick, M., Saeta, B., Sepassi, R., Spiridonov, A., Newlan, J., & Gesmundo, A. (2022). Scaling Up Models and Data with t5x and seqio.

[21] Nagrani, A., Yang, S., Arnab, A., Jansen, A., Schmid, C., & Sun, C. (2021). Attention bottlenecks for multimodal fusion. Advances in Neural Information Processing Systems, 34, 14200-14213.

[22] Li, J., Selvaraju, R.R., Gotmare, A., Joty, S.R., Xiong, C., & Hoi, S.C. (2021). Align before Fuse: Vision and Language Representation Learning with Momentum Distillation. Neural Information Processing Systems.

[23] Chen, S., He, X., Guo, L., Zhu, X., Wang, W., Tang, J., & Liu, J. (2023). VALOR: Vision-Audio-Language Omni-Perception Pretraining Model and Dataset.

[24] Gong, Y., Liu, A.H., Rouditchenko, A., & Glass, J. (2022). UAVM: Towards Unifying Audio and Visual Models. IEEE Signal Processing Letters, 29, 2437-2441.

[25] FirelordPhoenix. (2018, February 26). Computer Science Wiki. Retrieved from: https://computersciencewiki.org/index.php/File:MaxpoolSample2.png#filelinks.

[26] Chen, H., Xie, W., Vedaldi, A., & Zisserman, A. (2020). Vggsound: A Large-Scale Audio-Visual Dataset. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 721-725.

[27] Gemmeke, J.F., Ellis, D.P., Freedman, D., Jansen, A., Lawrence, W., Moore, R.C., Plakal, M., & Ritter, M. (2017). Audio Set: An ontology and human-labeled dataset for audio events. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 776-780.

[28] Kaplan, J., McCandlish, S., Henighan, T.J., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling Laws for Neural Language Models.