Research Internship Notes

Chang

her email

me

my email

This is more of a log. Will probably show results of experiments , screenshots .. of the stuff that i would be working on like a journal

-Adi

1. Checkpoints

Thurs Oct 24 09:30 PM CEST 2024

- [✓] push code to github -> https://github.com/adishourya/MedM/
  - submodules [✓] / standalone
    - dataset generation
    - finetuning
    - 4m
- [✓] 448 paligemma finetune (not good)
  - [] LLava-med model which was trained on PMC-MED 15M but license does not allow commercial use (does not follow our purpose of the report)
    - But we can use it just to see if our dataset is feasible . and maybe we can list it in the paper as a benchmark , and make a column of license in the table
  - [✓] collected compute metrics to run
- PMC Dataset :

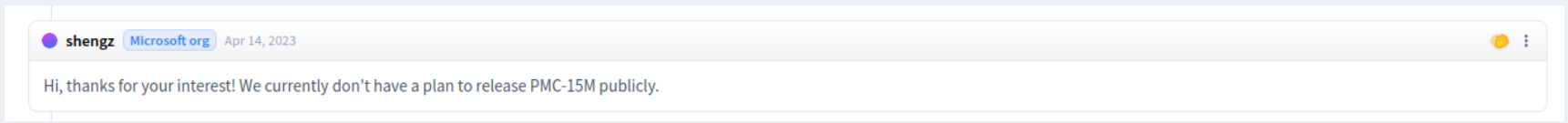


Figure 1: Like Conceptual Caption (CC12M google) dataset for medical images

- [✓] 4m Paper Summary (with snippets of code)

-Adi

2. Purpose of our article

This will keep changing, updated on :Mon Sep 30 10:24:29 AM CEST 2024

1. Make human readable report at radiologist clinics..
2. present current new and safer ways to get State of the art results for cheap.
   - Clinics/Hospitals instead of spending once should have yearly budget for local finetuning
   - as there are a lot of development on quantization and making the model smaller (budget balance with vol) while improving on the context window
   - present scaling law to calculate cost depending on the need
3. we will also present effective guarding techniques (phase after pre-training.. this is difficult..)
   - so that the report or Q/A does not give out horrible answers even if its right .. as it might be better if it came from a human
4. present a multi modality model for the ever developing need for adding new modalities. by developing a small **single** encoder-decoder model as opposed to many adapter models
   - see if we can improve masking techniques

-Adi

3. How others do it

- There are lots of ways to do it so i only focused on some of the common ones

As part of the Llama 3 development process we also develop multimodal extensions to the models, enabling image recognition, video recognition, and speech understanding capabilities. These models are still under active development and not yet ready for release. In addition to our language modeling results, the paper presents results of our initial experiments with those multimodal models.

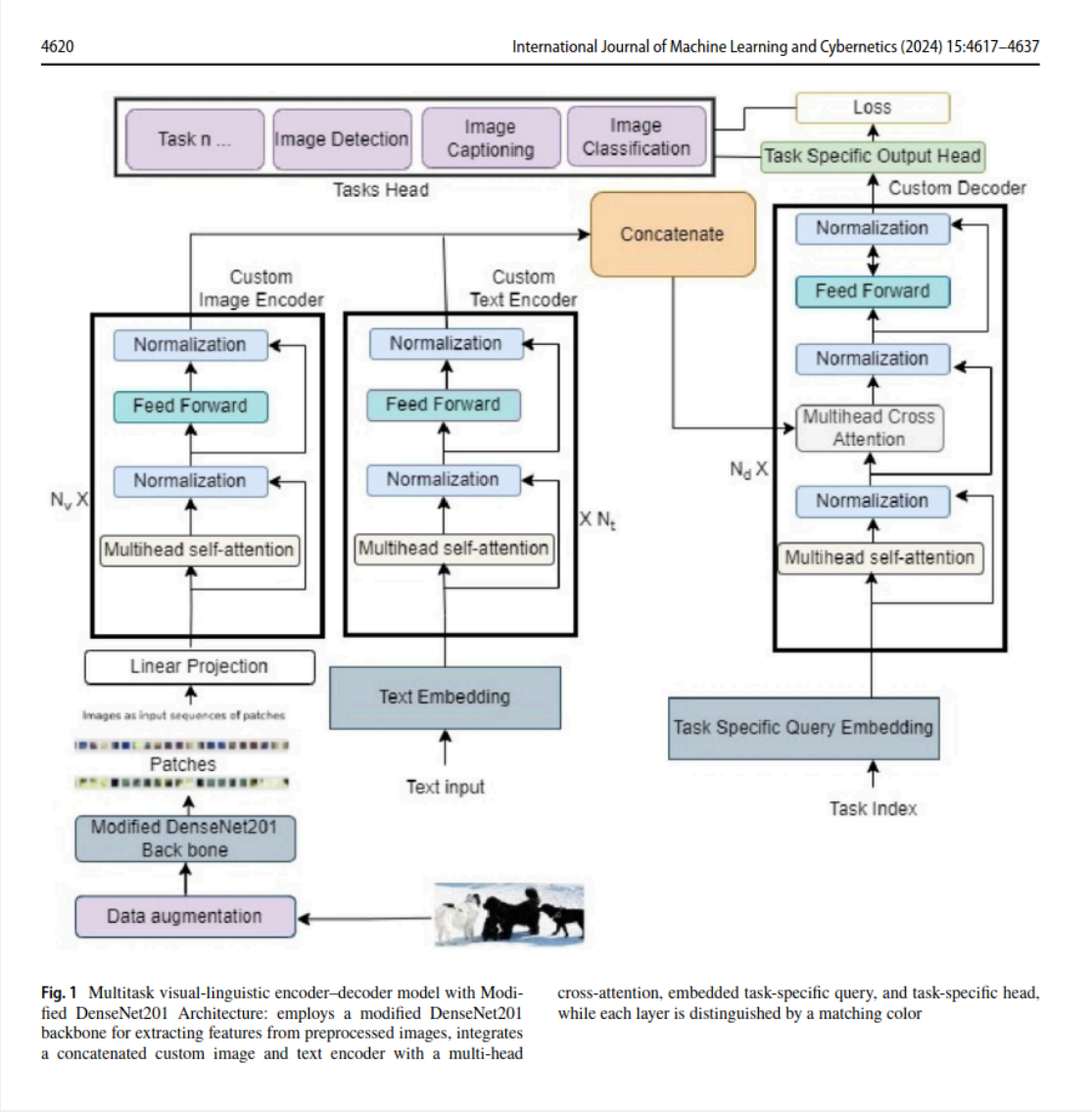


Fig. 1 Multitask visual-linguistic encoder–decoder model with Modified DenseNet201 Architecture: employs a modified DenseNet201 backbone for extracting features from preprocessed images, integrates a concatenated custom image and text encoder with a multi-head cross-attention, embedded task-specific query, and task-specific head, while each layer is distinguished by a matching color

Figure 2: Unified deep learning model for multitask representation and transfer learning: image classification, object detection, and image captioning

---

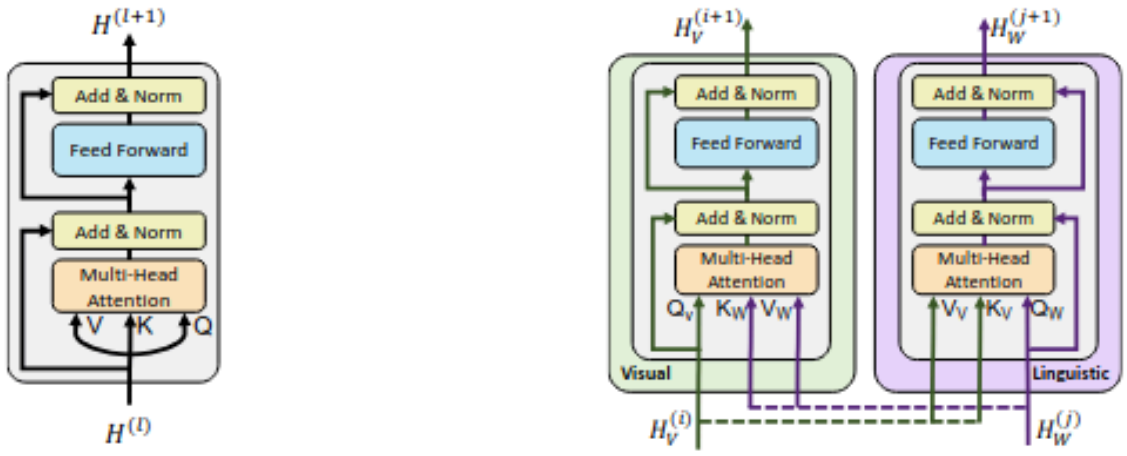
1. There is 1 encoder per modality . And no interaction between encoder blocks (self attention throughout)
2. The key value pairs come after the concatenation of the output of the encoder blocks
3. for adding new modalities we would have to add a new encoder ;then the previous encoders could be frozen . and only the new encoder and decoder would be trained
4. So there are as many embedding space as modalities ; and there is a task specific mlp head • classification task : `num_classes` , captioning task : `token max_generation`

Table 5 Image captioning training on Flickr8k and Flickr30k

Dataset	Loss	BLEU-4	Rouge1	CIDEr
Flickr8k	0.55	41.64	0.84	74.5
Flickr30k	0.52	33.42	0.82	68.9

Figure 3: Evaluation Metrics used for seqtoseq task

3.1. VILBERT Encoders only (Bidirectional Transformer)



(a) Standard encoder transformer block (b) Our co-attention transformer layer

Figure 2: We introduce a novel co-attention mechanism based on the transformer architecture. By exchanging key-value pairs in multi-headed attention, this structure enables vision-attended language features to be incorporated into visual representations (and vice versa).

Figure 4: VILBERT (Multiple)Encoder only

```
1 v is for visual
2 w is for words → linguistic
```

1. co-attention : (more like relative encoder decoder blocks) . The key value pair for linguistic encoder (here relatively decoder) comes from visual encoder (so purple block is the decoder for the green encoder block) and vice versa for the visual block
2. co-attention is very similar to bert in being bidirectional [This is a quick comparison b/n biderctional and left-right transformers]

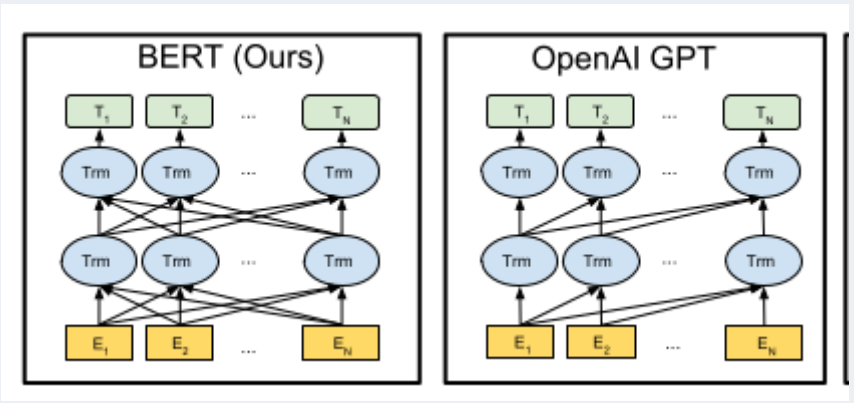


Figure 5: [Bert]Bert : Bidirectional ; GPT : Left to Right Transformer

- Straight line shows self attention and angled lines pointing inwards show the direction of passing kv pairs (cross attention in that node)
  - [BERT] all nodes have cross attention
  - [GPT] so the encoders to the right will have cross attention

Vision-and-Language. While we address many vision-and-language tasks in Sec. 3.2, we do miss some families of tasks including visually grounded dialog [4, 45], embodied tasks like question answering [7] and instruction following [8], and text generation tasks like image and video captioning [5]. These tasks may also benefit from a self-supervised approach similar to what we have presented. There are open questions on how to incorporate long sequences of images and text found in dialog, embodied tasks, and video processing. Further, it is unclear how to effectively decode output text from our bidirectional model as existing greedy decoders like beam-search do not apply.

Figure 6: [VILBERT] Section 5

4. 4m [Massivelly Multimodal Masked Modelling - Single Unified Transformer (left to right)]

• No Result table with evaluation metrics for captioning tasks in their paper …

1. need for multimodality in Medical AI :
   - > Unlike NLP, where language modeling on raw text has led to multitask capabilities, training on only RGB images with a single objective has not exhibited the same behavior for vision. Therefore, it is deemed important to incorporate multiple modalities and tasks in training. It has been indeed suggested by psychophysical studies that multimodality is one key driver behind the development of biological intelligence
     >
     > – Introduction
2. To train a single encoder decoder transformer it will have to be different than other multimodal transfomers as we can only afford 1 representation space and 1 (task free) mlp head
   - > Tokenizing modalities: We abstract away modality-specific intricacies by mapping all modalities into sequences or sets of discrete tokens, whether they are images, text, sparse data, or neural network feature maps. This allows every possible mapping between modalities to be seen as predicting one sequence or set of tokens from another.\ Tokenizing all modalities into a unified representation space allows us to train a single Transformer encoder-decoder to map between different modalities through (parallel or serialized autoregressive) token prediction
     >
     > – Section 2
   - we are allowed to use specialized tokenizer but the objective needs to learn a single embedding space like VILBERT did.
   - > This tokenization approach enhances compatibility, scalability, and sharing by removing the need for task-specific encoders and heads, allowing the Transformer to be compatible with all modalities and maintain full parameter-sharing.
     >
     > – Introduction
   - so we dont need task specific heads at the end like we did before
3. So the inputs and outputs would also have to be **masked** and prepared slightly diffently
   - > Also, although 4M operates on a large set of modalities, it can train in a highly efficient manner through the use of input and target masking. This involves randomly selecting a small subset of tokens from all modalities as inputs to the model, while another small subset of the remaining tokens is treated as targets.
     >
     > – Introduction

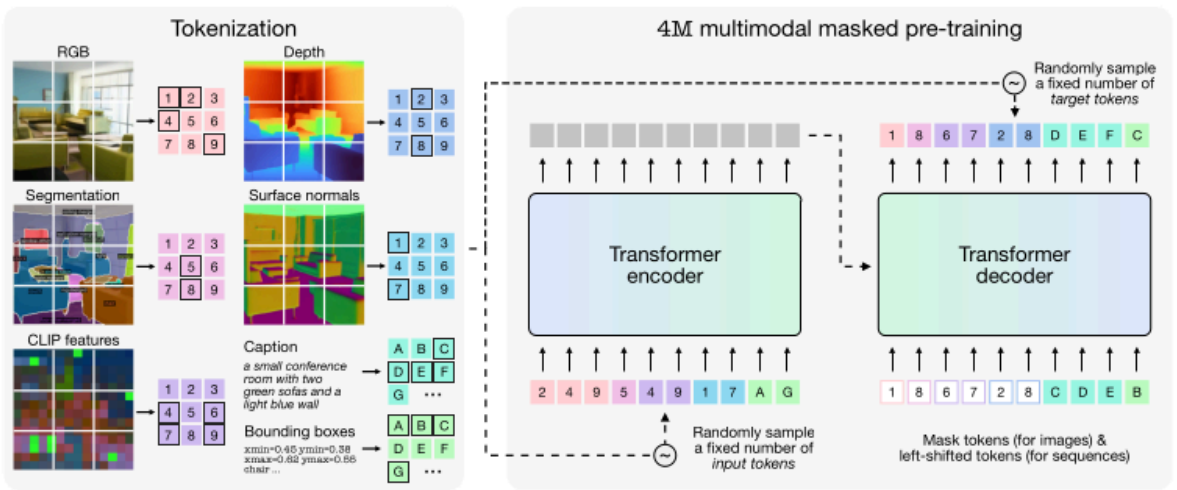


Figure 2: **Method overview.** (Left): 4M is a framework for training multimodal and multitask models that operate on tokenized versions of multiple image-like modalities (such as RGB, depth, etc.) and sequence modalities (such as captions and bounding boxes). (Right): The 4M pre-training objective consists of training a Transformer encoder-decoder to predict a randomly selected subset of tokens, which is sampled from all modalities, based on another random subset of tokens.

Figure 7: [4m]Different dataset preparation for Single Encoder Decoder System

1. Summary of what we see in the image:
   - all modalities gets indexed by modality specific definition ; integers for iamge based, characters for sequence based modalities **indexed not tokenized!**
   - sample of indices are drawn to generate an example ; input and target sequences (in black borders)
     - some of the sampled indexes are used as the input to the encoder
     - the remaining are used to make the target sequence (output of the decoder)
     - left to right transformers have generative capabilities.
     - for image based inputs : almost like autoencoder (noise fed in input)
     - dense modalities are masked from sequence inputs . so no self attention across modalities in decoder block. But they are free to tend to all key value pairs from the encoder (i.e all modalities)
     - for sequence based : right shifted output
       - here 3 from caption modality as input ; so it should be 3 from caption in the output (C D E) -> (D E F)
       - similarly for bounding box ; only 1 right shifted output (B) -> (C)
       - so there is a budget that controls number of tokens based on modalities

1. joint representational space from indices to vector , and backbone like modified densenet for image features

- > Multimodal encoder. The encoder is a standard Transformer encoder but features modality-specific learnable input embedding layers to map token indices to vectors. To each token of a specific modality, we add a learnable modality embedding and either 1D (for sequences) or 2D (for dense modalities) sine-cosine positional embeddings. To facilitate transfer learning, the encoder is additionally designed to accept RGB pixels using a learnable patch-wise linear projection, enabling it to double as a Vision Transformer backbone.
>
> – section2.2

2. job of decoder

- > Multimodal decoder. The decoder handles tokens from both dense image-like and sequence-like modalities, with each type requiring a different approach. However, two aspects are common to all tokens: First, they can all freely attend to any encoder tokens in the cross-attention layers, ensuring full access to the encoded information. Second, we employ attention masks to separate decoder tokens of different modalities. This ensures that the decoder produces consistent outputs for each specific modality, irrespective of what other outputs are being generated simultaneously. For dense image-like modalities, the decoder input consists of mask tokens along with modality and positional information. The decoder's role is to predict this masked content. For sequence-like modalities, the input to the decoder comprises modality, positional, and content information. The decoder is tasked to predict the next token in the sequence. To ensure that each token is only influenced by preceding tokens (and not by any future tokens), we apply a causal mask to the self-attention, as is standard in autoregressive models. Since all target tasks consist of discrete tokens, we can use the cross-entropy loss for all of them, which we found removes the need for task-specific loss balancing and improves training stability.
>
> – section2.2

3. These are some of the class documenations that makes it more clear and give better details

• Noise fed into image based inputs

```py
def image_mask(self, tensor: torch.Tensor, num_tokens: int, input_budget: int, target_budget: int):
    """Applies input and target masking to an image tensor

    Args:
        tensor: Image tensor
        num_tokens: Number of tokens in the tensor
        input_budget: Token budget for the input
        target_budget: Token budget for the target

    Returns:
        Dictionary containing the masked image tensor, the input mask, the target mask, and the decoder attention mask
    """
    noise = torch.rand(num_tokens)
    ids_shuffle = torch.argsort(noise, dim=0)
    ...

    return {"tensor": tensor, "input_mask": input_mask, "target_mask": target_mask, "decoder_attention_mask": decoder_attention_mask}

```

• using all modalities to generate dataset

```py
def build_fm_pretraining_dataset(
        data_path, all_domains, modality_info, modality_transforms,
        image_augmenter, text_tokenizer,
        input_tokens_range, target_tokens_range,
        sampling_weights=None):
    """Builds the FourM pre-training dataset based on the given arguments.
    This function should mainly used for smaller datasets (e.g. validation sets),
    while large training sets should be loaded with build_wds_fm_pretraining_dataloader in webdataset format.

    Returns:
        FourM pre-training dataset as a PyTorch Dataset.
    """
```

• And then they create the mixture dataset

```py
class MixtureDataset(IterableDataset):
    def __init__(self, data_iters, weights, modality_info):
        self.orig_data_iters = data_iters
        self.data_iters = [iter(data_iter) for data_iter in data_iters]  # Create initial iterators
        self.sampling_probs = np.array(weights) / sum(weights)
        self.modality_info = modality_info
```

• From `fourm/data/unified_dataset.py`

1. This needs massive masking operation to create the dataset (800 lines of code)

- > Decoupling the number of input and target tokens from the number of modalities prevents the computational cost from rapidly escalating with increasing modalities, allowing for a scalable training objective.
>
> – Introduction

2. Sampling operations

- ```py
  def sample_cosine(min_val: float = 0, max_val: float =1) -> float:
      """Sample a value from a cosine distribution between min_val and max_val"""

  def sample_uniform(min_val: float = 0, max_val: float =1) -> float:
      """Sample a value from a uniform distribution between min_val and max_val"""

  class UnifiedMasking(object):
      def __init__(self,
                   modality_info: Dict,
                   text_tokenizer: Optional[Tokenizer],
  ```
  ```

```
                input_tokens_range: Union[int, Tuple[int, int]],
                target_tokens_range: Optional[Union[int, Tuple[int, int]]],
                max_tries: int = 100,
                sampling_weights: Optional[List[float]] = None,):
        """Performs masking on a dict of modalities (both image based and sequence based modalities)

        Args:
            modality_info: Dict with the modalities and their corresponding information
            text_tokenizer: Tokenizer to use for text modalities
            input_tokens_range: Range of number of tokens to mask in the input
            target_tokens_range: Range of number of tokens to mask in the target
            max_tries: Maximum number of tries to find a valid token budgets
            sampling_weights: Sampling weights for the mixture of Dirichlet distributions
        """
```

- > Pseudo labeled multimodal training dataset. Training 4M models requires a large-scale and aligned multimodal/multitask dataset that contains all the above modalities/tasks and is sufficiently diverse.
  >
  > – Section 2

1. but none of the sampling operations are fused and torch.compile does not fuse across these sampling operations . People ussually write (`triton`) or `cuda` code with `torch.load_inline` to make it more efficient.
   - we could do that and discuss whether it improves the cost of sampling.

- a lot of papers write some pseudo code just to improve understanding but 4m paper doesnt . This is an example from siglip in `numpy`

```py
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]
- minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t
- learned temperature parameter
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss
= (loss_i + loss_t)/2
# !!Figure 3. Numpy-like pseudocode for the core of an implementation of clip
```

-Adi

<|END of Notes|>

-Adi