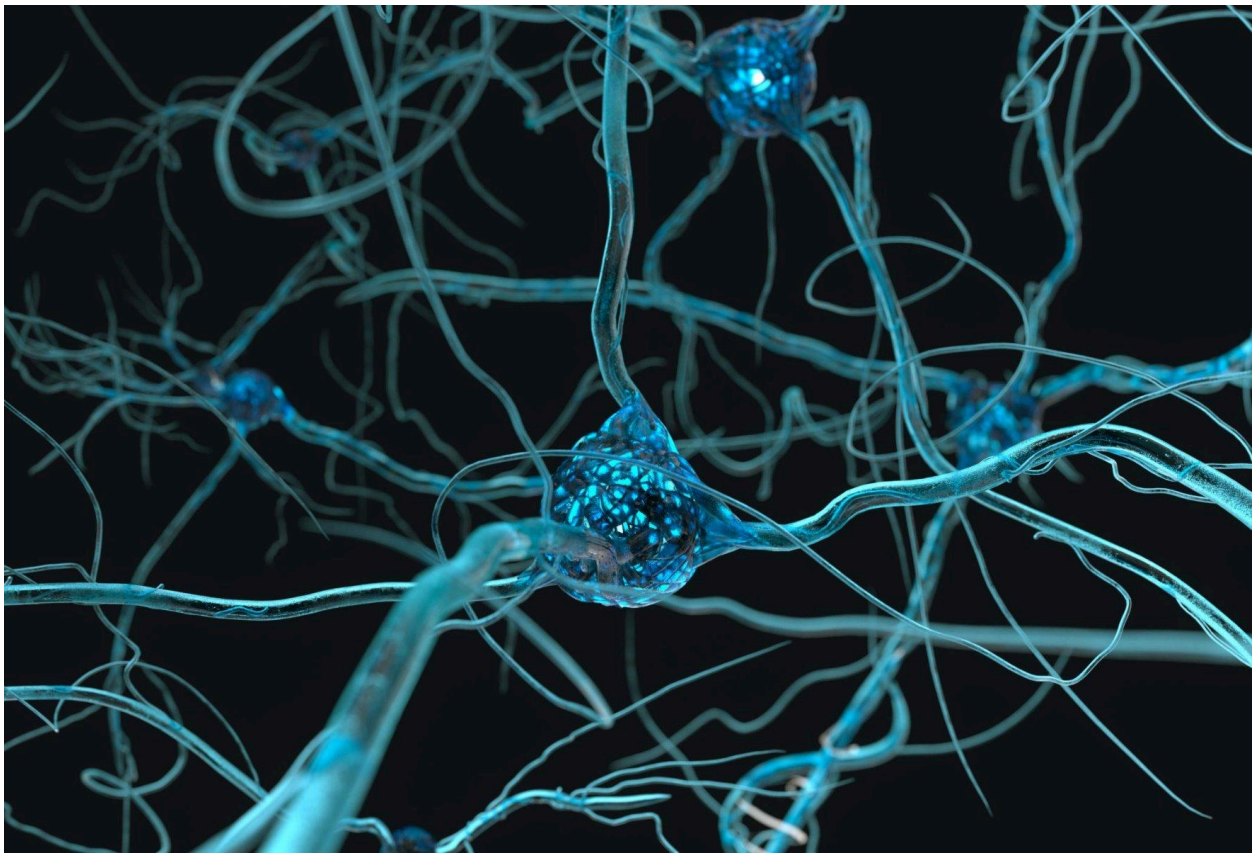


# IMAGE TEXT MATCHING USING ATTENTION FRAMEWORKS

—

SAKETH GAJAWADA - IMT2020531

AISHWARYA V KOUSHIK - MT2023506

## ABSTRACT

In this work, we propose attention frameworks for the task of image-text matching, where the goal is to align textual descriptions with their corresponding visual representations. Our approach uses attention mechanisms consisting of multiple layers that iteratively focus on relevant parts of the text and image through complementary attention flows in both directions. A bottom-up visual attention module based on Faster R-CNN detects important image regions, which along with the words are mapped to a joint embedding space to enable direct cross-modal similarity scoring. On standard image-text retrieval benchmarks, our attention frameworks achieve state-of-the-art performance, while offering interpretability by explicitly inferring the alignments between words and visual regions. Visualizations of the learned attention further provide insights into the model's cross-modal reasoning.

## INTRODUCTION

The problem addressed in this work is text-image matching, which involves aligning textual descriptions with their corresponding images. This is an important task that requires grounding natural language expressions to visual representations for true multimodal understanding.

Existing approaches for image-text matching typically involve encoding the text into fixed-length vector representations using tools like word embeddings, and encoding the images into fixed-length representations using convolutional neural networks (CNNs) or other image encoding techniques. However, many prior methods treated the text and image representations equally when measuring their similarity, ignoring the differential importance of different words or image regions.

The authors propose using attention frameworks, a class of neural network architectures inspired by human vision, that allow models to dynamically focus on the most relevant parts of the input text and images. Specifically, they introduce stacked cross attention models that consist of multiple layers of cross-attention mechanisms. In each layer, the model computes attention weights indicating the importance of different parts of one modality (text or image) with respect to the other modality. By stacking multiple such cross-attention layers, the model can capture increasingly complex relationships and correspondences between the text and images. The attended features from both modalities are then aligned using the computed attention weights and fused together into

a joint representation. This joint representation, encoding the alignments between words and image regions, is finally fed to a classifier or regressor for the image-text retrieval task.

## MOTIVATION

- **Importance of grounding language to vision:** To truly understand multimodal data consisting of natural language and visual information, it is crucial to ground the language expressions to their corresponding visual representations. This enables tighter integration and reasoning across the two modalities.

- **Interpretability of multimodal models:** Many existing image-text matching models act as opaque black boxes, without providing insights into why they make certain predictions. By explicitly inferring the alignments between individual words and visual regions, the proposed attention-based approach can make these models more transparent and interpretable.

- **Limitations of previous approaches:** Prior work either ignored the varying importance of different words and image regions, or could only capture one semantic alignment between the two modalities at a time through multi-step attention mechanisms. This limited the models' ability to handle the diverse and varying correspondences present across different image-text pairs.

- **Varying alignments across data:** The number and nature of semantic correspondences between words and visual concepts can vary significantly across different image-text examples. A flexible model architecture is needed to handle this data variation.

- **Leveraging attention for alignments:** The remarkable success of attention mechanisms in both the vision and natural language processing domains motivates the use of attentional reasoning to infer the latent alignments between textual and visual representations.

## LITERATURE REVIEW

The authors position their work in the context of several prior approaches for image-text matching:

1. **Image-Text Matching with Bottom-Up Attention:** Early work by Karpathy and Fei-Fei (2015) detected image regions using R-CNN and aggregated region-word similarity scores, but without employing attention mechanisms.

   Niu et al. (2017) mapped noun phrases and objects to a shared embedding space on top of whole image/sentence embeddings.

2. **Whole Image/Sentence Embedding Methods:** Initial efforts like those of Kiros et al. (2014) and Faghri et al. (2017) mapped whole images and sentences to a common embedding space, without considering explicit region-word alignments. The proposed region-word alignment approach is positioned as being more interpretable compared to these embedding-based models.
3. **Conventional Attention-based Methods:** Nam et al. (2017) proposed a dual attention network capturing the interplay between vision and language through multiple steps. However, these models focused on one semantic alignment at a time with a predefined number of steps, limiting their flexibility.

## TECHNICAL APPROACH - CODE FRAGMENTS

- To start with, all the necessary libraries have been installed onto the system
- CapsCollate
  - This class is a collate function used in PyTorch's DataLoader to apply padding to captions within batches.
  - It takes a batch of data as input and pads the captions to ensure they have the same length.
  - The **pad_sequence** function from PyTorch is utilized for padding.

- Flickr Dataset
  - This class represents the dataset, loading images and their corresponding captions.
  - It initializes with the root directory of the images, the file containing captions, and optional transformations.
  - During initialization, it builds the vocabulary based on the captions.

- The **__getitem__** method loads an image and its corresponding caption, applies transformations, numericalizes the caption, and returns them as tensors.

- Vocab and Spacy
  - This class manages the vocabulary for the captions.
  - It initializes with pre-reserved tokens like **<PAD>**, **<SOS>**, **<EOS>**, and **<UNK>** (unknown).
  - Methods are provided to tokenize text, build the vocabulary based on a frequency threshold, and convert text to numericalized tokens.
  - It imports **spacy** for natural language processing tasks and defines **spacy_eng** to load the English language model.

- Modified ResNet-50 architecture
  - **Initialization**:
    - The **EncoderCNN** class inherits from **nn.Module**.
    - Inside the **__init__** method, the ResNet-50 model is loaded with pre-trained weights (**pretrained=True**).
    - It freezes the parameters of the ResNet model by setting **requires_grad** to **False** for all parameters.
    - The last two layers (average pooling and fully connected layers) are removed from the ResNet architecture, leaving the convolutional layers.
    - The modified ResNet model is stored as **self.resnet**.
  - **Forward Pass**:
    - The **forward** method takes a batch of images (**images**) as input.
    - The input images are passed through the ResNet model (**self.resnet**), resulting in a tensor of features.
    - The shape of the features tensor is **(batch_size, 2048, 7, 7)**, where **2048** represents the number of channels and **7x7** represents the spatial dimensions.
    - The tensor is permuted to change its dimensions to **(batch_size, 7, 7, 2048)**.
    - Finally, the tensor is flattened along the spatial dimensions (**7x7**) to produce a feature tensor of shape **(batch_size, 49, 2048)**, where **49** is obtained by flattening **7x7**.

- Data Loader
    - A PyTorch **DataLoader** is created to handle batching and loading of data.
    - It is initialized with the dataset, batch size, number of workers, and a collate function (**CapsCollate**) to apply padding to captions within batches.
    - The **collate_fn** argument specifies how batches are collated, and it uses the **CapsCollate** function.
    - Additionally, the vocabulary size is obtained from the dataset, and the device (CPU or GPU) is determined based on availability.

- Batch Processing and Loss Function
    - For each batch of images and captions, the images and captions are transferred to the specified device (CPU or GPU).
    - Gradients are zeroed using **optimizer.zero_grad()** to clear any previous gradients.
    - The model is then fed forward with the images and captions to obtain the model outputs and attentions.
    - The batch loss is calculated using the CrossEntropyLoss between the model outputs and the target captions.
    - Backpropagation is performed using **loss.backward()** to compute gradients.
    - The optimizer updates the model parameters using **optimizer.step()**.
    - The loss function is defined as **nn.CrossEntropyLoss**.
    - The **ignore_index** parameter is set to the index of the **<PAD>** token in the vocabulary. This ensures that the padding tokens are not considered when computing the loss.

- Logging and Evaluation
    - If the index of the current batch plus one modulo the **print_every** value equals zero, the current loss is printed.
    - Additionally, the model generates a caption for a sample image and displays it.
    - The model is set to evaluation mode during caption generation using **model.eval()** and reverted to training mode afterward using **model.train()**.

- Obtaining Attentions
    - "Obtaining attentions" refers to the process of computing attention weights or scores over the spatial features of the image.
    - During the decoding phase of the model, when generating a caption for an image, the attention mechanism dynamically focuses on different parts of

the image depending on the current decoding state (hidden state) and the previously generated words in the caption. This mechanism allows the model to attend to relevant regions of the image while generating each word of the caption.

- The attention weights indicate how much each spatial feature (or region) of the image contributes to generating the next word in the caption. These weights are typically visualized as heatmaps, showing where the model is focusing its attention within the image at each decoding step.

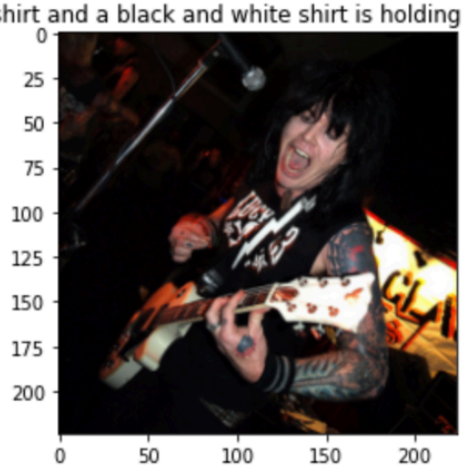
- Attention Visualization
  - **plot_attention(img, result, attention_plot)**:
    - This function takes three arguments: the input image (**img**), the generated caption (**result**), and the attention weights (**attention_plot**).
    - A figure is created for visualization, and the attention heatmaps are plotted alongside the input image and corresponding caption.
    - Each heatmap represents the attention weights for a specific word in the caption, showing where the model focuses its attention within the image.
    - The function displays the figure containing the input image, caption, and attention heatmaps.

# RESULTS AND DISCUSSION

## EPOCHS VS CROSS ENTROPY LOSS

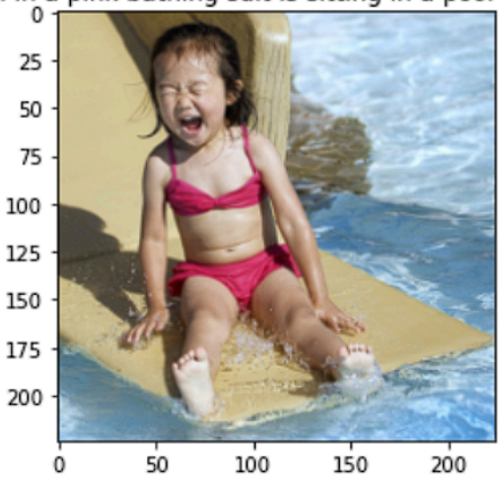
```
Epoch: 15 loss: 2.26783
```

a man in a red shirt and a black and white shirt is holding a guitar on a stage .




```
Epoch: 25 loss: 2.03723
```

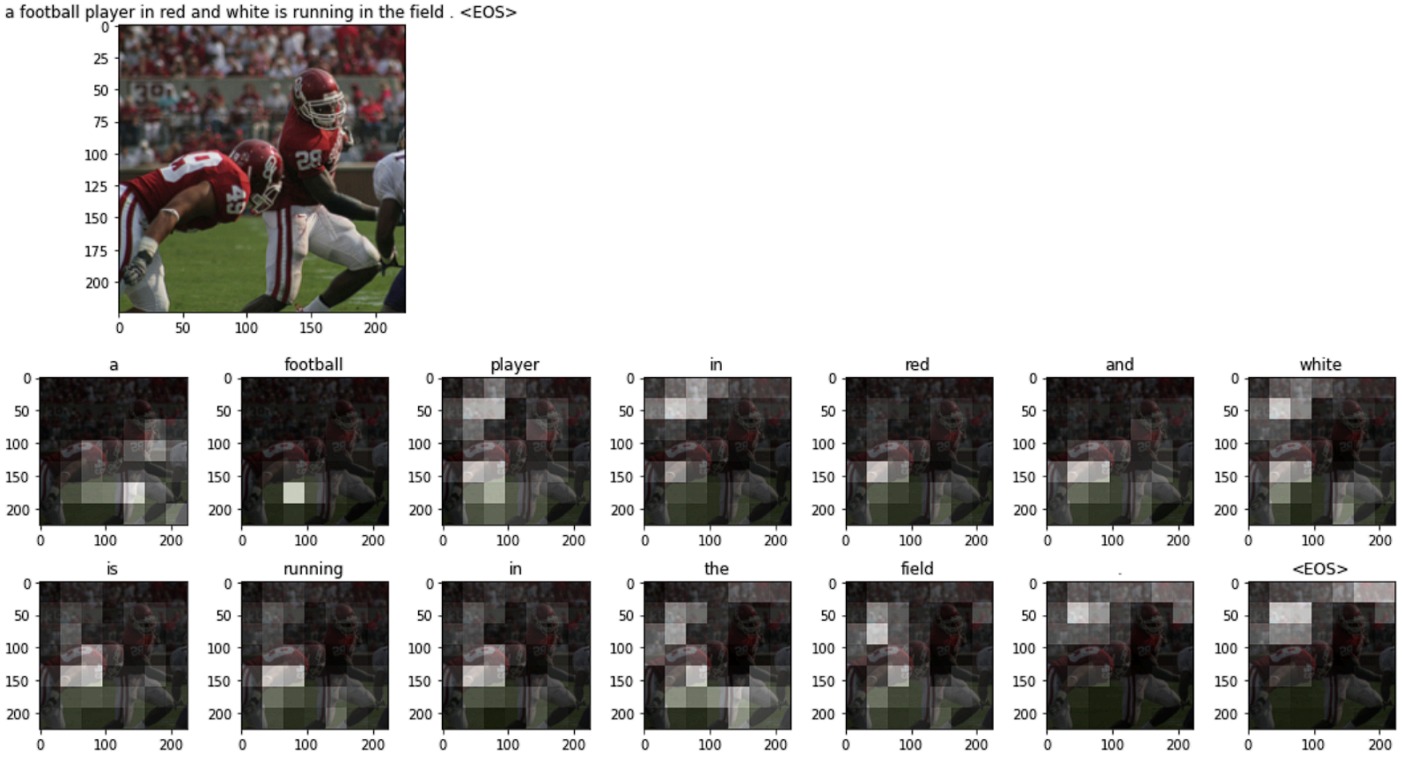
a girl in a pink bathing suit is sitting in a pool . <EOS>



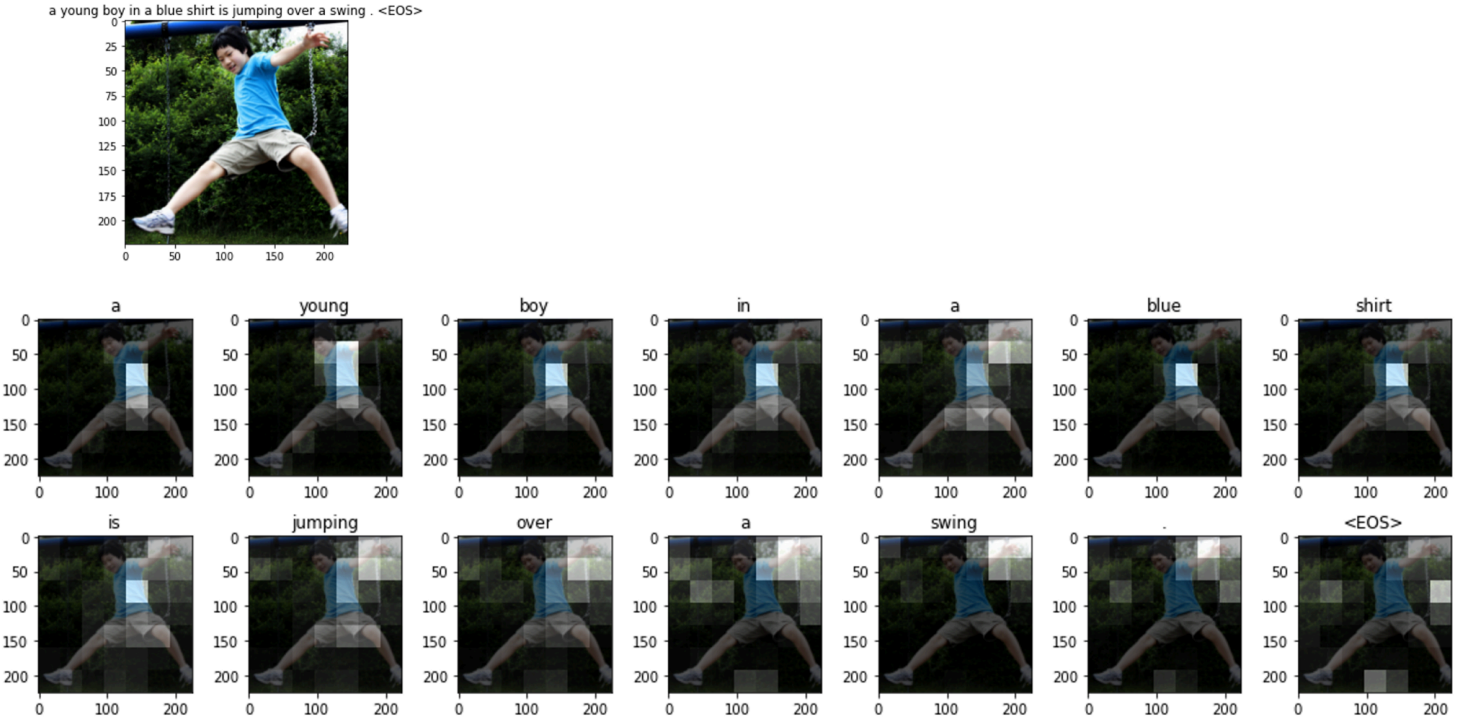

Each EPOCH takes about 4-5 minutes. As seen in the above figures, increasing the number of EPOCHS reduces the cross entropy loss.

# ATTENTION HEATMAPS

a football player in red and white is running in the field . <EOS>




a young boy in a blue shirt is jumping over a swing . <EOS>



Each heat map depicts how the image-text model maps every word in a  sentence to a particular region of the image.

## CONCLUSION AND FUTURE SCOPE

- The paper uses the proposed attention frameworks and achieves state-of the-art results for image retrieval given text query and vice versa.
- A few potential areas where we can expect significant advancement -
    - Exploring better ways to represent and attend to image regions beyond just object detection ie, incorporating relationships between objects
    - Models that accommodate visual question-answering, captioning etc where fine-grained alignments between vision and language could be beneficial
    - Investigating more powerful architectures like transformers in place of CNNS/RNNs
    - Scalability for larger image-caption datasets