Image Captioning with Visual Attention

Aleena Chanda, Gayara Fernando, Ved Piyush

12/15/2021

Abstract

Image captioning is the process of generating descriptive text for images. This report describes how the image captioning model can be trained using standard deep neural network architectures. We also implement the concept of visual attention in the model's training. The model can itself decide the salient parts to focus on while generating the caption. We used the MS COCO (Common Objects in Context) [1] dataset for the model training and validation.

Introduction

Recently there have been many advancements in the architectures related to Convolutional Neural Networks (CNNs) [2] and Recurrent Neural Networks (RNNs) [3]. Image captioning is one such field that utilizes the power of CNNs and RNNs to deal with both images and text required to train a robust caption generating model. The task of image captioning is something that human brains can do with minimal effort, but it has been traditionally rigid for deep learning models to mimic this human behavior.

The concept of visual attention (figure 1) [4] helps the model automatically find relevant areas in the image when it generates a specific word in the predicted caption. The benefit of this is that instead of compressing the entire image into a low-level representation and using that representation for each generated word, the model can determine a dynamic representation. To extract the features from the images, we use a pre-trained Inception V3 [5] model and extract the feature maps from the last convolutional block. These feature maps have a shallow height and width dimension. Although it alleviates the memory and computational time issues it introduces the challenge of extracting meaningful interpretation as the features at the last convolutional block are not very interpretable. The visual attention mechanism can map these feature maps to the spatial dimensions of the original image, thereby allowing us to see where the model looks when it generates predictions.

Image captioning has many different uses but can be primarily used for image indexing (figure 2) and scene description (figure 3). The generated text in the case of image indexing provides relevant information for downstream searching algorithms. For example, we can retrieve the pictures of monuments once the image captioning model spits out the monument names in the captions. Similarly, for scene understanding, the generated text can explain the content of the scene and can be very useful for visually impaired people.

Dataset

The MS COCO dataset has everyday objects (vehicles, humans, animals, etc.) in different day-to-day situations. Each image has five human annotations (figure 4). We use 6400 images to train the model and 1600 images to monitor the validation loss.

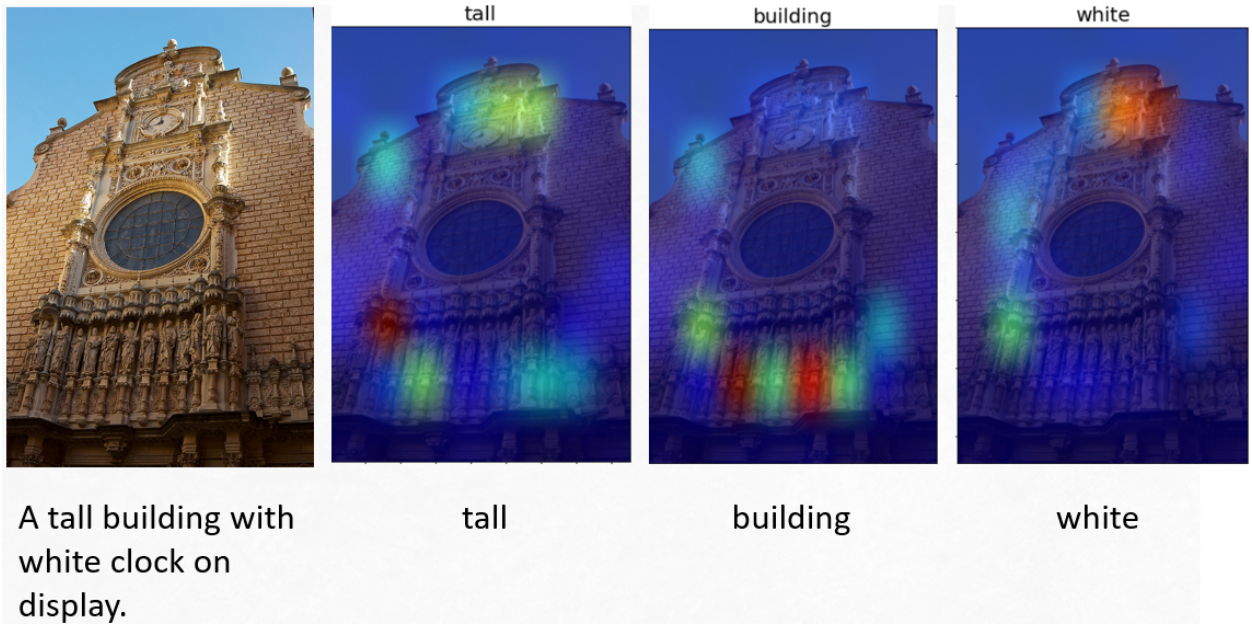


Figure 1: Example for visual attention - The red areas in the image are most predictive for the corresponding word and the blue areas are the least predictive.

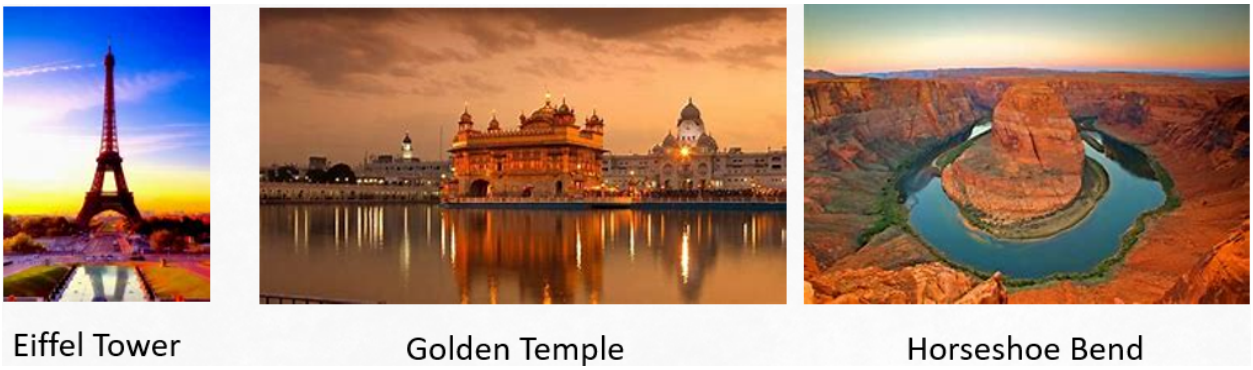


Figure 2: Example for image indexing

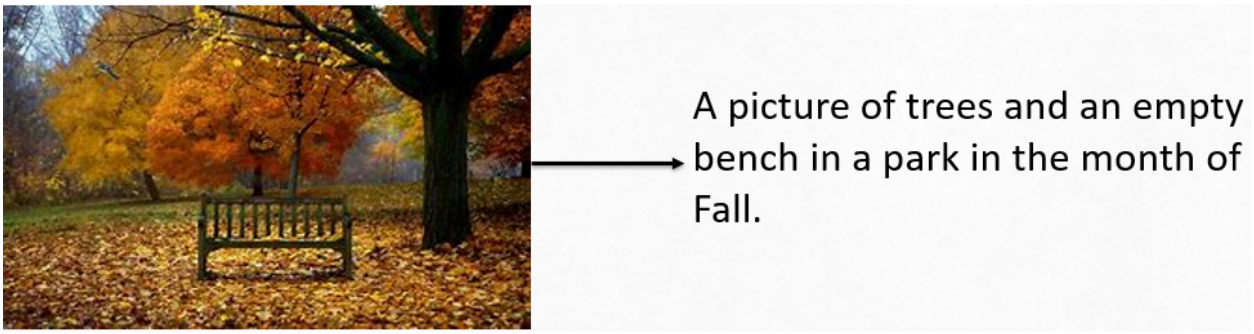


Figure 3: Example for scene description

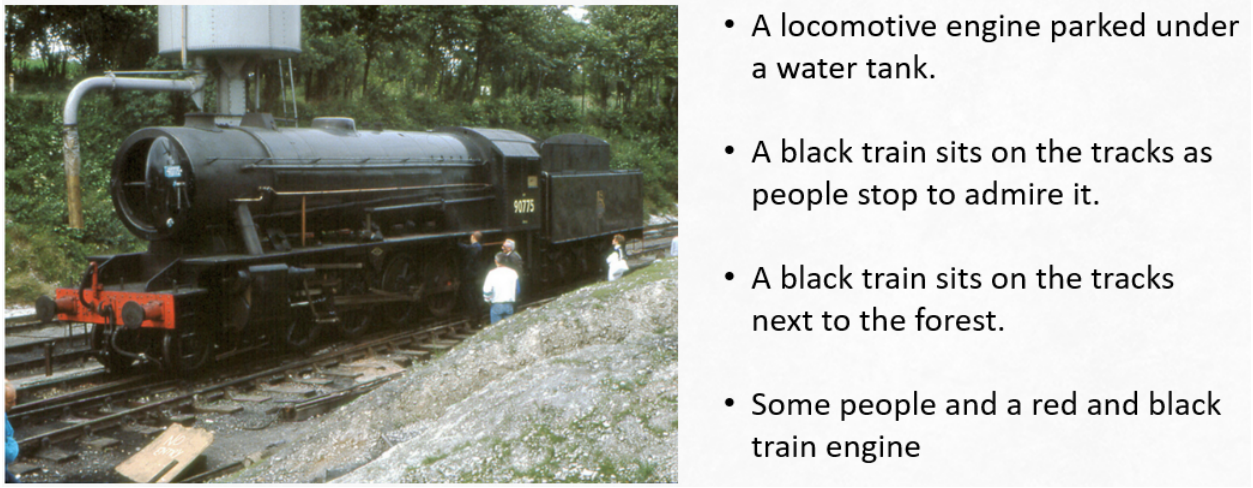


Figure 4: Example annotations for an image

Model Details

Feature Preprocessing

Images

The feature maps for the images are extracted from the Inception V3 model. We extract the feature maps from the last convolutional block, resulting in 512 feature maps of size 7 x 7. We retrieve these feature maps for all the images and then store them on the disk. These feature maps would be used later to train the image captioning model. Although the feature maps from the last convolutional block are not interpretable to illustrate the concept of feature extraction from pretrained models, we can visualize the feature maps from one of the first convolutional blocks (figure 5). We flatten the three-dimensional tensors to a two-dimension tensor, i.e., 7 x 7 x 512 gets converted to 49 x 512 (figure 6).

Text Annotations

To preprocess the text, we first convert each unique word to an integer and then keep the top 6000 most frequently occurring words. We remove punctuations and convert all words to lowercase. We also add a and token to the start and end of the caption, respectively, so that the model learns how to start and end a caption for an image. For training convenience, we pad all captions to be off the same length as the longest caption. We allow the image captioning model to automatically learn a d dimensional vectored representation for these top 6000 words. This is a supervised version of learning word embeddings [6].

Model Architecture

The model architecture is the encoder-decoder architecture, where the encoder learns to encode the feature extracted from the images into a lower dimension space. The decoder learns to decode this encoded representation by mapping it to the ground truth correct word. The architecture is best summarized in figure 5.

We would now explain the model's working, as shown in figure 7. The model's training composes a loop wherein each iteration of the loop uses the current word and the image tensor to predict the next word. Then in the next iteration, the next word in sequence and the image tensor is used to predict the subsequent word and so on. At each iteration of this loop the model also learns to focus its gaze on the salient parts of the

Figure 5: Feature extraction from Inception V3

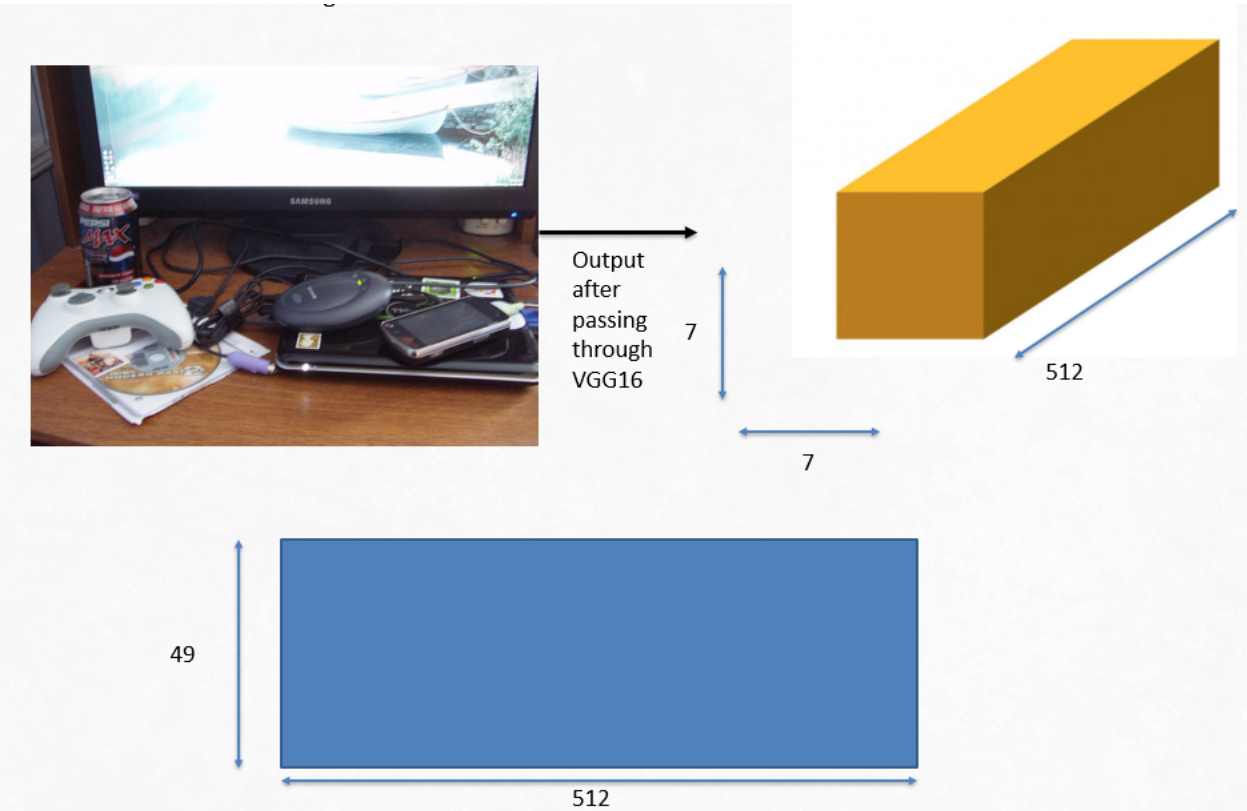


Figure 6: Conversion of a 3-D tensor to 2-D

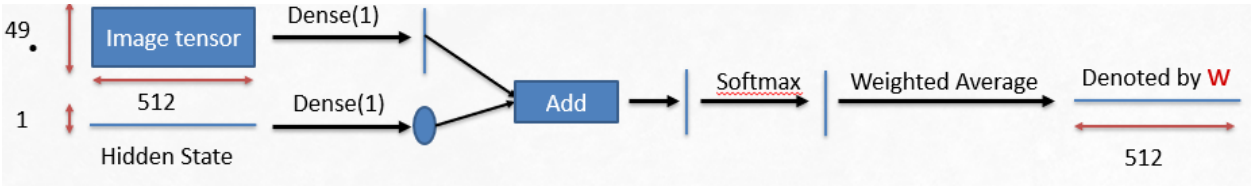


Figure 7: Encoder Decoder Architecture

image which can then be visualized as heatmaps over the original image. To retain memory in the caption generating process, there is a hidden state in the model, which is initialized to a vector of zero at the first iteration of the training loop. Both the image tensor and the hidden state transform using a fully connected layer. The fully connected layer collapses across the width dimension of the tensor to convert the 49 x 512 dimensional tensor to a 49 x 1 dimensional vector; similar transformations for the hidden state convert the 512 x 1 dimensional vector into a single scalar value. We add this scalar value to the 49 x 1 dimensional vector via element-wise addition. We then pass the resultant vector obtained through a softmax operation that will convert all elements into a probability distribution. We can interpret each element in the softmax vector as the relative importance of the corresponding row in the 49 x 512-dimensional tensor. For example, the first element tells us how important the first row of the 49 x 512-dimensional tensor is. We then use this softmax vector to take a weighted average of the 49 x 512 dimensional tensor and collapse it to a 1 x 512 dimensional vector (`w`). We then extract the embedding of the current word and concatenate it with the weighted average vector `w`, and denote the concatenated vector as `c`. We then pass this concatenated vector through a sequential neural network type model such as a Recurrent Neural Network and get the processed output from it and denote it as `o`. This output vector `o` can then be treated as features to a multinomial logistic regression. We are trying to map the output vector to the corpus of the vocabulary words we have retained in the training data. This means passing the output vector through some fully connected layers and finally getting a probability distribution over the vocabulary space. At the culmination of the iteration, we set the hidden state vector as the output `o` vector, which comes from the Recurrent Neural Network, and the seed word to be the ground truth target from the current step. We repeat the same process for all words in the training ground truth caption. The training steps are summarized below:

- Instantiate the hidden state by a vector of zeros; denote this as `h`.
- The starting seed word is `<START>`.
- For each word in the ground truth caption:
  - Get the 49 x 512 dimensional image feature.
  - Pass the hidden state and the 49 x 512 dimensional image tensor through fully connected dense layer to obtain 49 x 1 and 1 x 1 vector.
  - Add the 1 x 1 vector element-wise to the 49 x 1 vector to obtain a 49 x 1 dimensional vector.
  - Pass the previous vector through a softmax operation to obtain a 49 x 1 dimensional probability distribution vector.
  - Use the softmax vector from the previous step to take a weighted average across the rows to convert the 49 x 512 dimensional tensor to a 49 x 1 dimensional vector, denote this vector as `w`.
  - Get the embedding for the seed word and concatenate it with the vector w, denote this vector as `c`.
  - Pass the vector `c` through a recurrent neural network to obtain the output `o` $o = W_h c + Bh + u$ (W, B, and u are the weights for the RNN)
  - Pass the output `o` through another fully connected dense layer to obtain o_1, $o_1 = W_1 o + b$ (W_1 and b are the weights for the fully connected dense layer).
  - Pass o_1 through the softmax operation to get a probability distribution across the vocabulary of words.
  - Set the hidden state `h` = `o` and the seed word as the target word from this iteration. This concept of using the target from the current iteration as an input into the next iteration is known as teacher forcing.

– Repeat this process for all words in the ground truth caption

Model Training and Evaluation

We predict the correct word at each iteration in the loop in a multiclass classification problem. The loss function suitable for this supervised problem is the categorical cross-entropy loss function. The model is an end to end trainable, which means that all the weights and biases of the model are learned jointly from the data. The optimizer used is the Adam optimizer which is an improvement over the Stochastic Gradient Descent, and it obviates the need to hand tune the learning parameter. We also use a validation set to be able to track the validation loss (figure 8) along with the training loss to detect any overt signs of overfitting. The model evaluation loop is the same as the training loop except that we do not use teacher forcing since we do not know the ground truth labels while predicting. Instead, after getting the expected probability distribution across the vocabulary of words, we sample a new word using the multinomial distribution and the predicted probability distribution.

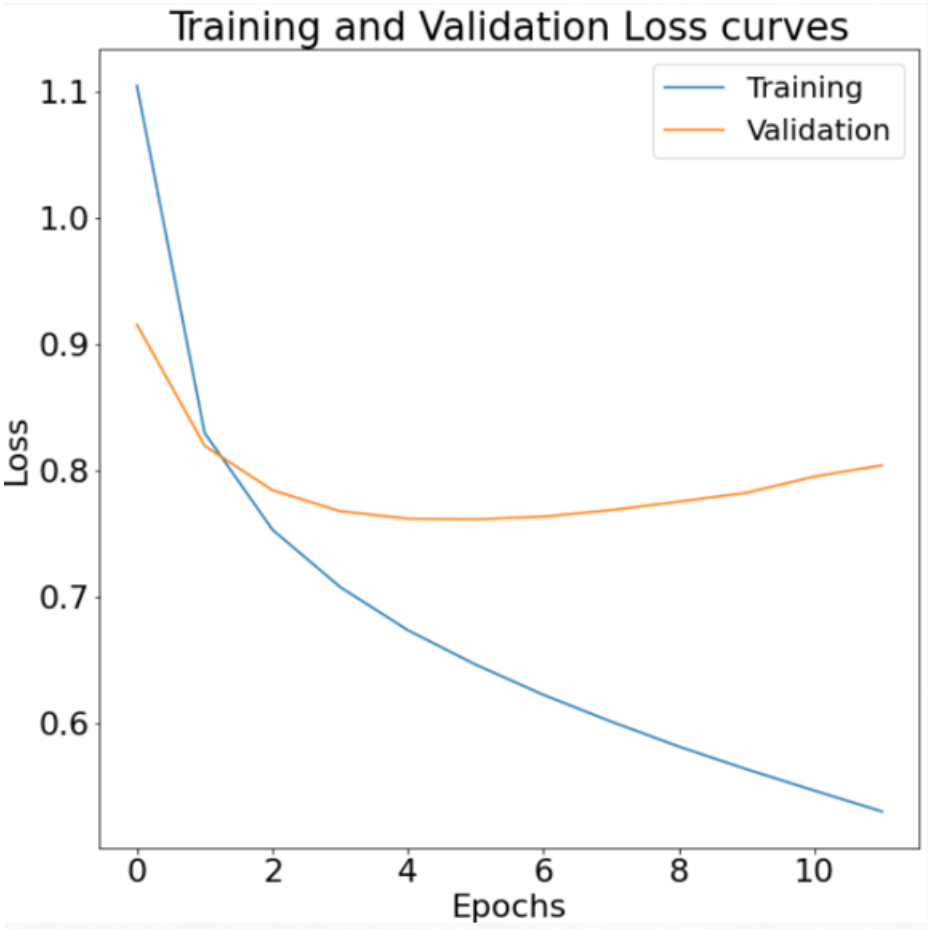


Figure 8: Training vs Validation Learning Curves

Visual Attention - How to get the heatmap for where the model looks?

A fascinating artifact of the image captioning model is to generate heatmaps for the image for each predicted word in the caption. This can help us troubleshoot any obvious biases the model has and then decide on a strategy to mitigate the bias.

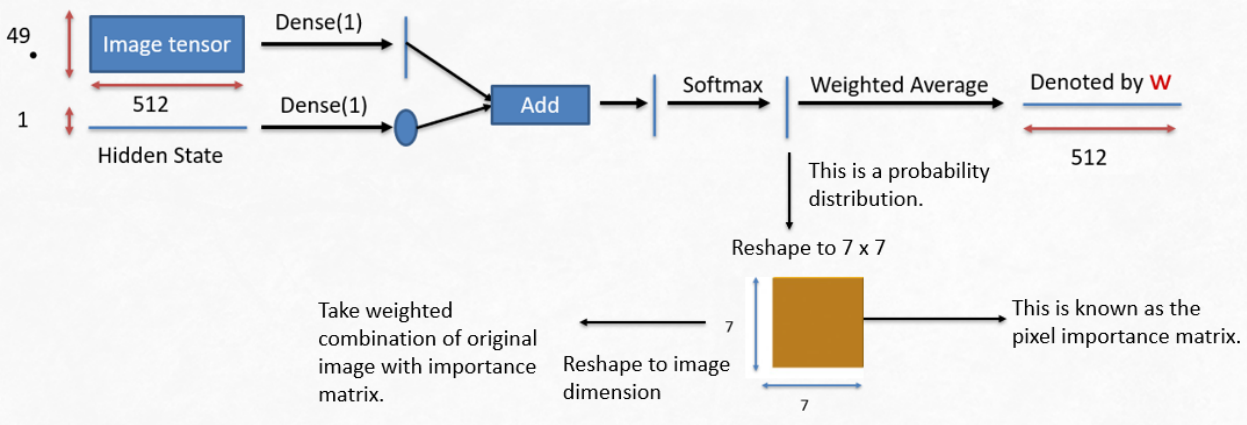


Figure 9: How to get the heatmaps for the image?

In the image in figure 9 we can see that the softmax vector can be reshaped into a 7 x 7 two-dimensional vector. This 7 x 7 two-dimensional tensor can be thought of as the pixel importance matrix. We first normalize each element in this 7 x 7 two-dimensional tensor by scaling it using the maximum value so that it comes in the range of 0 to 1, the same range that we use for the pixels in the image. The next step is to reshape this 7 x 7 normalized pixel importance matrix to the original dimensions of the image. We can then get the heatmaps superimposed on top of the images by taking a simple weighted average of the original image with the pixel importance matrix. Another example for the visual attention is shown in figure 10.

The softmax operation would not zero out any element of the 49 x 1 dimension vector, which is then reshaped into the 7 x 7 pixel importance matrix. This corresponds to the soft attention, giving a nice smooth transition in the heatmaps. However, instead of the softmax operation, we could use a sigmoid operation independently on the 49 x 1 dimension vector, which would zero out some elements and give sharper focused (hard) attention plots. Some additional theoretical details of visual attention and soft and hard attention are given in the appendix (we apologize for the bad formatting for the appendix).

Additional caption examples

We also show some additional caption results in figure 11.

Why is this project a computational project?

We believe this project is a computational project for many reasons, which we will explain in this section. First, the variety of data that we have in training in terms of images and natural language makes it imperative to brainstorm sound feature extraction procedures. We leveraged pre-trained CNNS for feature extraction from the images and then used RNNs to process the natural language features (word embeddings). These models can be intractable even with CPU parallelization. We learned how to train these complex neural network models by leveraging NVIDIA CUDA-enabled GPUs that expedite the training by orders of magnitude. Also, the model's training for image captioning involves two new aspects of training with a loop

Figure 10: Visual Attention Example

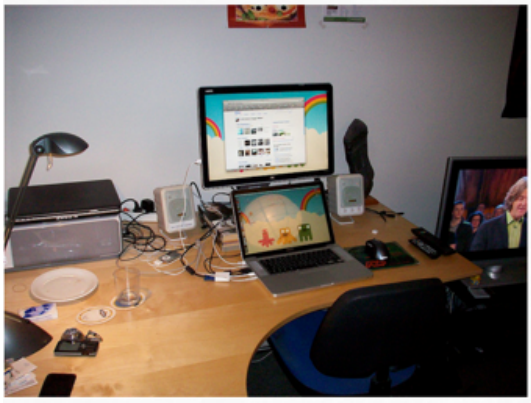
and teacher forcing. Training the model (or using backpropagation to change the model weights) in a loop allows us to use the target words from the previous position as inputs to predict the next position. This is an entirely new concept that we haven't used or been exposed to so far in our graduate school coursework or research. Also, the design of the complex model architecture requires quite a lot of thought in terms of which features to merge/concatenate with which other features, how to use the learned weights and reshape and then be able to generate the visual attention heatmaps was also quite challenging for us and encouraged us to leverage some basic computer vision skills. Not to mention the many hyper-parameters that such a model involves, for example, what dimension to encode the extracted features from the images in, what embedding size to use, what optimization algorithms, etc. In summary we feel this project challenged us and we had to a lot of trial and error and also read, experiment, and implement some high level TensorFlow code which instilled in us the confidence and abilities to tackle more complex architectures and think of other novel projects such as the current image captioning project.

Future Work

- We want to bake the feature extraction of images from the pretrained network in an end-to-end fashion. The feature extraction is currently not baked in, and the model is not end-to-end. This stops us from fine-tuning the weights of the pre-trained network (via Transfer Learning) and optimizing the weights to our domain dataset.
- Similar to the first point, we could also use GLOVE embeddings for the words instead of learning them from scratch. This will allow us to start our model training from cognizance rather than a completely random point in the parameter space. We could also leverage transfer learning to fine-tune the GLOVE embedding vectors to our domain dataset.
- As was suggested by Professor Ghosh, we would also want to check the behavior of the visual attention heatmaps to various transformations such as rotation and addition of noise. In particular, we would like to see whether the heatmaps stay more or less stable or the introduction of translation and noise confuses it, thereby giving some evidence on further probing that needs to be done to understand where the model is actually looking fully.

a snowboarder is standing in plaid jacket ready to slopes <end>

several cars are sitting next to a road <end>



a desktop and other monitors and two laptops and shelves and a desk next to it <end>

several different types of pastries on a table topped with tulips and ready plate <end>

Figure 11: Additional caption examples

- It would be nice to embed this model prediction pipeline in an application or, better yet, Raspberry Pi so we could generate image captions on real-world images and see how the model fares in the wild.
- Looking at the learning curves in figure 8, there is clear evidence of overfitting. Therefore, we also want to introduce some regularization via the dropout layers, L1/L2 penalty, etc.

References

[1] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015, February 21). Microsoft Coco: Common Objects in Context. arXiv.org. Retrieved December 16, 2021, from https://arxiv.org/abs/1405.0312

[2] ImageNet classification with deep convolutional ... - neurips. (n.d.). Retrieved December 16, 2021, from https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[3] Sherstinsky, A. (2021, January 31). Fundamentals of Recurrent Neural Network (RNN) and long short-term memory (LSTM) network. arXiv.org. Retrieved December 16, 2021, from https://arxiv.org/abs/1808.03314

[4] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2016, April 19). Show, attend and tell: Neural image caption generation with visual attention. arXiv.org. Retrieved December 16, 2021, from https://arxiv.org/abs/1502.03044v3

[5] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015, December 11). Rethinking the inception architecture for computer vision. arXiv.org. Retrieved December 16, 2021, from https://arxiv.org/abs/1512.00567

[6] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, September 7). Efficient estimation of word representations in vector space. arXiv.org. Retrieved December 16, 2021, from https://arxiv.org/abs/1301.3781

Image Captioning with Visual Attention

Appendix

•**Common Framework:** We consider the probability of determining the appropriate captions for an input image. This normally consists of two steps:

The image has to be encoded in an internal vector representation h using CNN(Convolutional Neural Network).

h is decoded into word vectors signifying captions using an RNN.

Mathematically we can write this as:

h_t = f(x,h_{t-1}); x being the image input.

next word = g(h_t)

However, the problem with this method is that when generating the single word of the caption the LSTM looks at the whole image representation h every time. This is not very efficient as when the model is trying to generate the next word of the caption, this word is usually describing only a part of the image. ([2]) The LSTM mechanism has been illustrated in Figure 1. To solve this problem we create different non-overlapping sub regions h_i which represents the internal representation to generate the i^{th} word. When a decoder decides on a caption for every word; it only looks at specific regions of the image leading to a more accurate description. [5]. We will see how the decoder decides on which regions to consider.

•**Attention Model:** The key difference between an LSTM model and the one with attention is that an attention model pays attention to particular areas or objects rather than treating the whole image equally. Let us look at Figure 2. . At the beginning of caption creation we start with an empty context. Our first attention area starts with a man who walks towards us. So we update the context vector with the word "A". The attention area is not changed and the next word is predicted to be "man". So the new context vector is updated to be "A man", The attention area is now shifted where the model looks at what the man is holding in his hand. Thus an attention mechanism predicts the whole caption by continuously exploring and shifting the attention locations. The final caption predicted by this mechanism comes out to be "A man holding a couple plastic containers is walking down in an intersection towards me". An attention unit considers

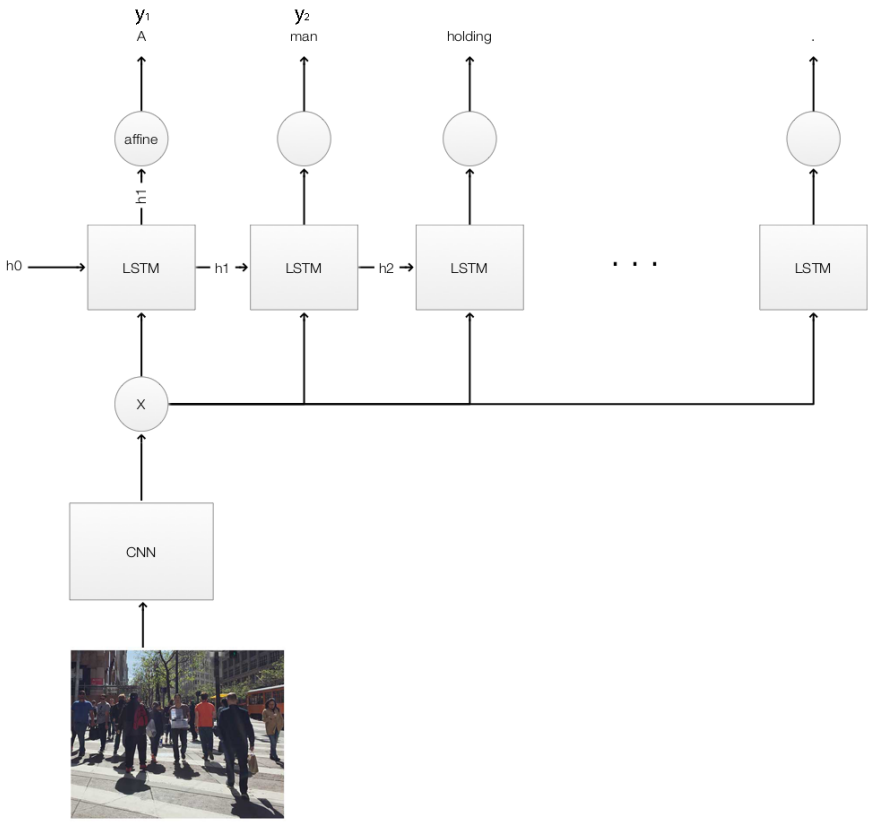


Figure 1: Caption Generation with LSTM Mechanism

Figure 2: Attention Mechanism

all sub regions and contexts as its inputs and outputs the weighted Arithmetic Mean of these regions. In attention model we replace x in the LSTM model with an attention module attention. This can be mathematically represented by: h_t = f(attention(x,h_{t-1}),h_{t-1}) ([6]). Figure 3 helps in a detailed understanding of the model.

•**Soft Attention:** The Soft Attention Model, instead of using the image x as an input to the LSTM, inputs weighted image features accounted for attention.This can be clearly seen in Figure 4. Soft attention discredits irrelevant areas by multiplying the corresponding feature maps with a low weight. Accordingly, high attention areas (brighter in the picture) keep the original value while low attention areas get closer to 0 (become dark in the visualization). With the context of "A man holding a couple plastic", the attention module creates a new feature map with all areas darkened except the plastic container area.([1]) The mechanism of Soft Attention Model has been described in details. The attention model starts with computing a score s_i to measure how much attention should be given to the image input x_i. A tanh activation function is applied and the s_i's are passed through a softmax function which outputs them as probabilities α_i. Finally the inner products of the probabilities α_i and the sub regions x are calculated to get the final output Z of the relevant regions of the entire image. ([2]). The whole process has been mathematically represented as follows:

$s_i = \tanh(W_c\text{C}+W_x\text{X}) = \tanh(W_c h_{t-1}+W_x\text{X})$.

$\alpha_i = \text{softmax}(s_1, s_2, \cdots, s_i, \cdots)$

$\text{Z} = \sum_i \alpha_i x_i$

•**Hard Attention:** Hard attention model focuses on the most relevant parts of the image [4]. The model uses α_i as a sample rate to pick one x_i as the input to the LSTM. The score s_i follows a Bernoulli distribution

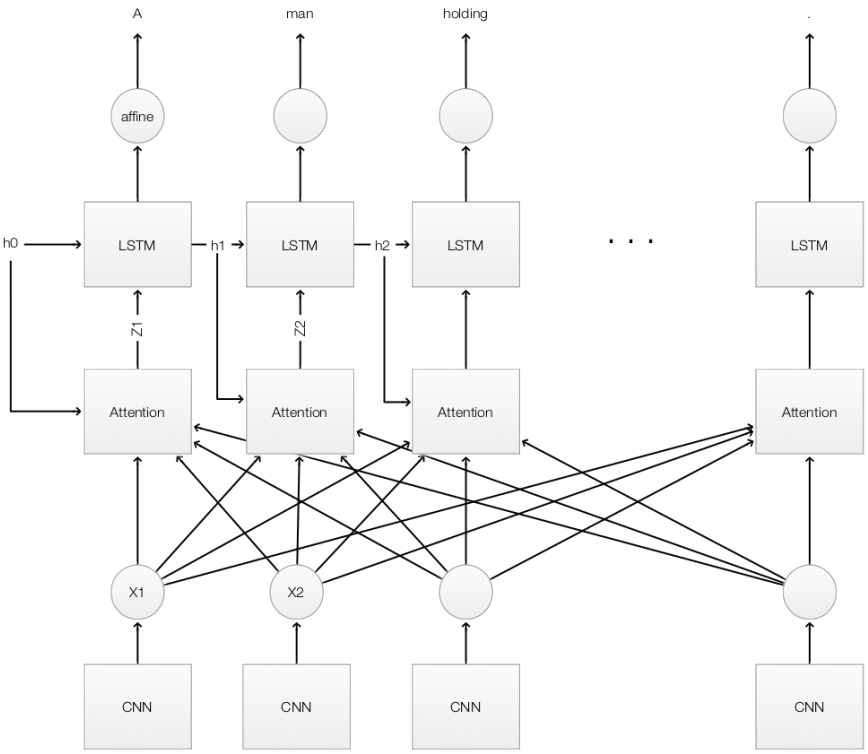


Figure 3: Complete Flow of the LSTM Model Using Attention

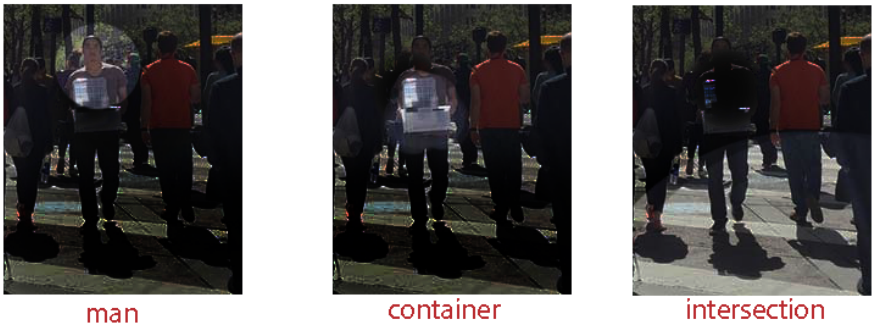


Figure 4: Soft Attention

characterized by the parameter α_i where s_i is an indicator variable which is set to 1 if the i^{th} location is the one to extract the visual features. The final output Z is given by the inner product of α_i and x_i.[3] This can be mathematically represented as follows:

$P(s_i{=}1) = \alpha_i$

$Z = \sum_i \alpha_i x_i$

References

[1] https://jhui.github.io/2017/03/15/Soft-and-hard-attention/

[2] https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

[3] @inproceedingsxu2015show, title=Show, attend and tell: Neural image caption generation with visual attention, author=Xu, Kelvin and Ba, Jimmy and Kiros, Ryan and Cho, Kyunghyun and Courville, Aaron and Salakhudinov, Ruslan and Zemel, Rich and Bengio, Yoshua, booktitle=International conference on machine learning, pages=2048–2057, year=2015, organization=PMLR

[4] https://towardsdatascience.com/attention-in-neural-networks-e66920838742

[5] https://www.youtube.com/watch?v=W2rWgXJBZhUt=321s

[6] https://theaisummer.com/attention/