Image Captioning Using LSTMs and Attention

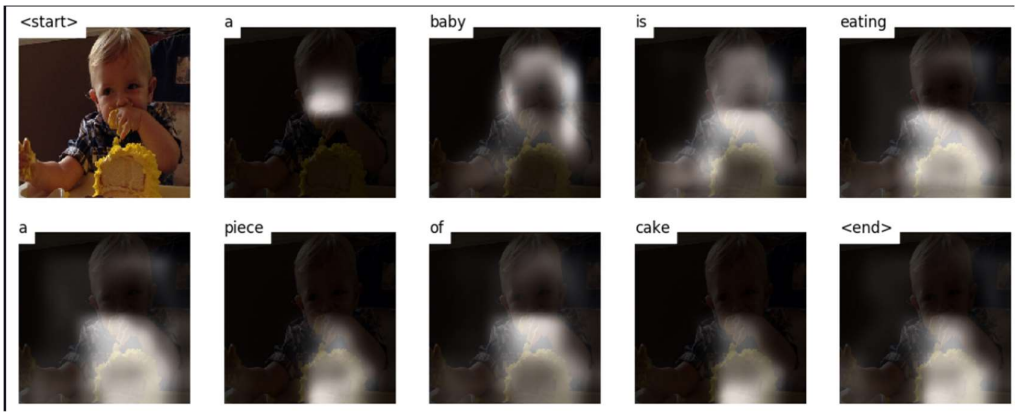
Neel Subodh Mehta
Electrical and Computer Engineering
University of Illinois at Chicago
Chicago, USA

Mohammed Tanveer
Electrical and Computer Engineering
University of Illinois at Chicago
Chicago, USA

***Abstract*—Using Image Attention coupled with LSTMs to create an image captioning model.**

I. Description and Key Ideas

This template, This model learns *where* to look. As you generate a caption, word by word, you can see the model's gaze shifting across the image. This is possible because of its *Attention* mechanism, which allows it to focus on the part of the image most relevant to the word it is going to utter next.



Important Concepts:

Encoder-Decoder architecture.

Typically, a model that generates sequences will use an Encoder to encode the input into a fixed form and a Decoder to decode it, word by word, into a sequence.

Attention.

The use of Attention networks is widespread in deep learning, and with good reason. This is a way for a model to choose only those parts of the encoding that it thinks is relevant to the task at hand. The same mechanism you see employed here can be used in any model where the Encoder's output has multiple points in space or time. In image captioning, you consider some pixels more important than others. In sequence-to-sequence tasks like machine translation, you consider some words more important than others.

Transfer Learning.

This is when you borrow from an existing model by using parts of it in a new model. This is almost always better than training a new model from scratch (i.e., knowing nothing). As you will see, you can always fine-tune this second-hand knowledge to the specific task at hand. Using pretrained word embeddings is a dumb but valid example. For our image captioning problem, we will use a pretrained Encoder, and then fine-tune it as needed.

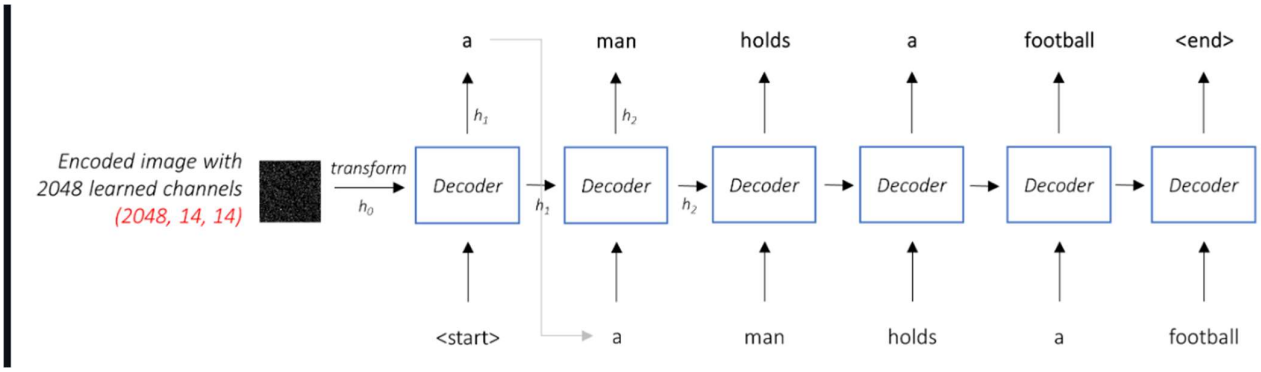
Encoder

The Encoder encodes the input image with 3 color channels into a smaller image with "learned" channels. This smaller encoded image is a summary representation of all that's useful in the original image. Since we want to encode images, we use Convolutional Neural Networks (CNNs).We don't need to train an encoder from scratch. Why? Because there are already CNNs trained to represent images. For years, people have been building models that are extraordinarily good at classifying an image into one of a thousand categories. It stands to reason that these models capture the essence of an image very well. I have chosen to use the 101 layered Residual Network trained on the ImageNet classification task, already available in PyTorch. As stated earlier, this is an example of Transfer Learning. You have the option of fine-tuning it to improve performance. These models progressively create smaller and smaller representations of the original image, and each subsequent representation is more "learned", with a greater number of channels. The final encoding produced by our ResNet-101 encoder has a size of 14x14 with 2048 channels, i.e., a

2048, 14, 14 size tensor. I encourage you to experiment with other pre-trained architectures. The paper uses a VGGnet, also pretrained on ImageNet, but without fine-tuning. Either way, modifications are necessary. Since the last layer or two of these models are linear layers coupled with softmax activation for classification, we strip them away.

Decoder

The Decoder's job is to look at the encoded image and generate a caption word by word. Since it's generating a sequence, it would need to be a Recurrent Neural Network (RNN). We will use an LSTM. In a typical setting without Attention, you could simply average the encoded image across all pixels. You could then feed this, with or without a linear transformation, into the Decoder as its first hidden state and generate the caption. Each predicted word is used to generate the next word. In a setting with Attention, we want the Decoder to be able to look at different parts of the image at different points in the sequence. For example, while generating the word football in a man holds a football, the Decoder would know to focus on – you guessed it – the football!



Attention

The Attention network computes these weights. Intuitively, how would you estimate the importance of a certain part of an image? You would need to be aware of the sequence you have generated so far, so you can look at the image and decide what needs describing next. For example, after you mention a man, it is logical to declare that he is holding a football. We will use soft Attention, where the weights of the pixels add up to 1. If there are P pixels in our encoded image, then at each timestep t –

$$\sum_{p}^{P} \alpha_{p,t} = 1$$

You could interpret this entire process as computing the probability that a pixel is the place to look to generate the next word.

In summary,

Once the Encoder generates the encoded image, we transform the encoding to create the initial hidden state h (and cell state C) for the LSTM Decoder.

At each decode step,

the encoded image and the previous hidden state is used to generate weights for each pixel in the Attention network.

the previously generated word and the weighted average of the encoding are fed to the LSTM Decoder to generate the next word.

II. New knowledge gained

Starting off, we had a fair idea of what an LSTM was, and what a CNN was. But this allowed us to learn how to combine a image model with a language model. We refreshed our knowledge on neural network modelling but we learnt about so many new concepts like transfer learning, and image attentions.

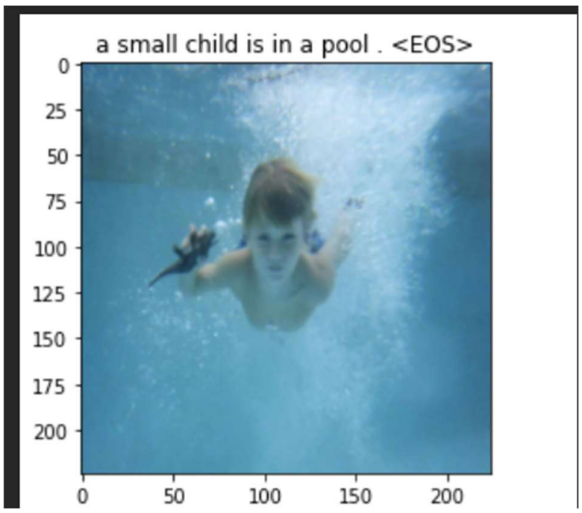
III. Challenging part(s) of the project

The most challenging part of the project was understanding how language and image models work. These are relatively heavy deep learning topics and recent too. Understanding all the mathematical algebra and calculus behind these models was an arduous task.

IV. WORK DISTRIBUTION AMONG TEAM MEMBERS

When it came to work distribution, both of us were equally responsible for the coding, shooting ideas off each other on how to tune the hyperparameters. It was very difficult to learn all the required tasks on our own, so we distributed the learning part and explained each other the implementation of what we learned.

SIMULATIONS



The network correctly classified most images. Extremely good considering the training data used was only 8K images

REFERENCES

[1] Show, attend and tell: neural image caption generation with visual attention- Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37

[2] Attention Is All You Need- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin