Bilkent University
CS 464 - Introduction to Machine Learning
Spring 2021
Term Project Progress Report
Group 14

Project Title: AutoCapNet: Deep Learning Based Approaches for Automatic Image Captioning

Group Members:

Melisa Taşpınar	21803668
Yiğit Gürses	21702746
Emin Adem Buran	21703279
Musa Ege Ünalan	21803617
Mustafa Göktan Güdükbay	21801740

Introduction

In this project, we intend to perform automatic image captioning using deep learning-based methods. To be more specific, our goal is to describe given images with sentence-based captions using several deep learning methods and at the end compare these methods based on their performance. As of now, we use InceptionV3 [1] (pretrained on the ImageNet dataset [2]) for the encoding stage and custom GRU [3] and LSTM [4] based networks for the decoder stage.

For our project, we will be using the *Flickr8k* [5] dataset as we stated in our Proposal. *Flickr8k* is a benchmark dataset designed to be used for the task of sentence-based image description. The dataset contains 8,091 images and each image has five different captions describing the contents of each image.

Background Information

Automatic image captioning can be defined as the task where a deep learning model is trained to automatically designate captions to digital images. The assigned captions are typically strings of words that describe the contents of the image. There exist a multitude of real life applications where image captioning proves useful such as aiding blind people to beware of their environments, annotating pictures and interpreting the contents of the images shared on social media. [6]

To perform automatic image captioning, the first task to fulfill would be to find a large dataset, then clean and preprocess it. Afterwards, the process of automatic image captioning is often divided into two smaller processes: encoding images and decoding the acquired features to form strings of words.

For the first stage, i.e. encoding the images, it is possible to use a CNN (convolutional neural network), as it will be able to extract features from the input images. For instance, if we take a CNN that performs classification and get rid of its last layer, it will be outputting the relatively high-level features it would use for classification. These outputted features can then be flattened to acquire a vector of features, which can in turn be transformed and resized so that it has the same length as the word embedding. [7]

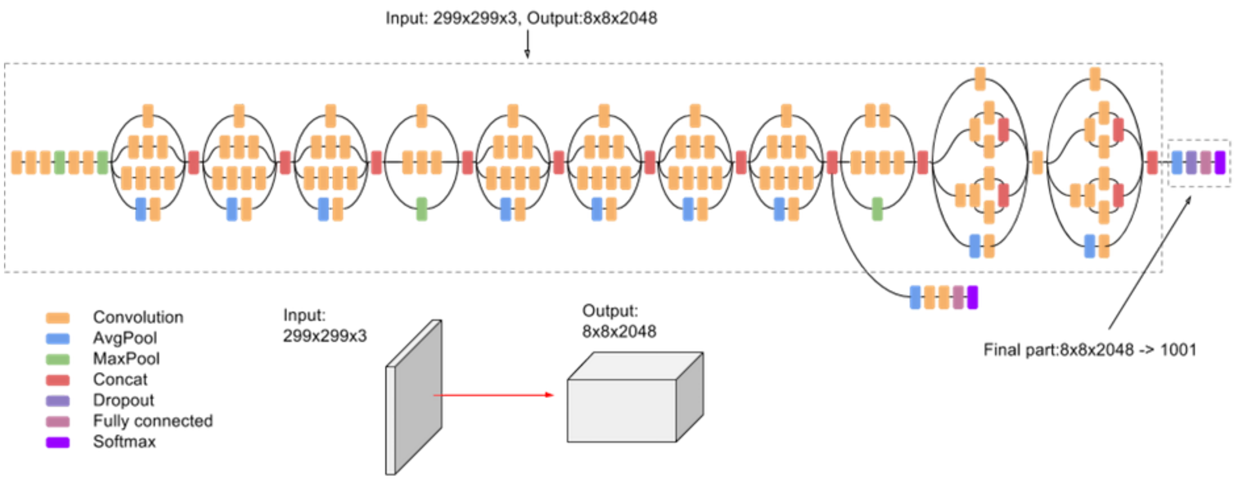

Figure 1: Inception V3 [8]

Here, let us give more information on the Inception V3 architecture. This architecture is built to obtain a computationally efficient convolutional neural network. It uses factorized convolutions with larger filter size, smaller convolutions, asymmetric convolutionsm auxiliary classifiers and grid size reduction. By this way, the network contains less number of parameters and less computations which allows for faster training of the model. [9, 10]

The second stage is to decode the features acquired in the first stage and to use these to compose sequences of words that fittingly describe the given images. For this task, an RNN (recurrent neural network) can be used. RNN's, unlike CNN's, accommodate loops that allow them to maintain information, or in other words, to have some kind of memory. However, this memory tends to be limited, and as a result RNN's suffer from so-called "long-term dependencies." [11] This is where a particular type of RNN's called LSTM's (long short term memory) comes in. LSTM's are structured with the purpose of tackling the long-term dependency problem. These networks have so-called memory cells in their architectures that can preserve information for an extended amount of time. Thus, they can be used for the decoding task during image captioning. [7] Using such a network, we can decode the features we extracted in the first stage to acquire sequences of words. These sequences will be the final product, each will be the caption outputted for a given input image.

An LSTM has four layers: the input gate, forget gate, cell state and output gate. The forget gate passes the previous hidden state and current input features through a sigmoid layer to decide on which information to discard. The input gate passes the previous hidden state and the current input through a sigmoid function to decide on the importance of the inputs to the cell. An intermediate candidate cell state is produced by passing the previous hidden state and the current input through a tanh function. The current cell state is then updated by using the output of the forget gate, previous cell state, output of the input gate and the candidate cell. The output gate decides on the next hidden state. The output gate passes previous hidden state and current input through a sigmoid function. The current cell state that was calculated is passed through the tanh function. Lastly, an element wise multiplication is performed with the result of the output gate and the result of the tanh function to obtain the output of the hidden layer [3, 11].

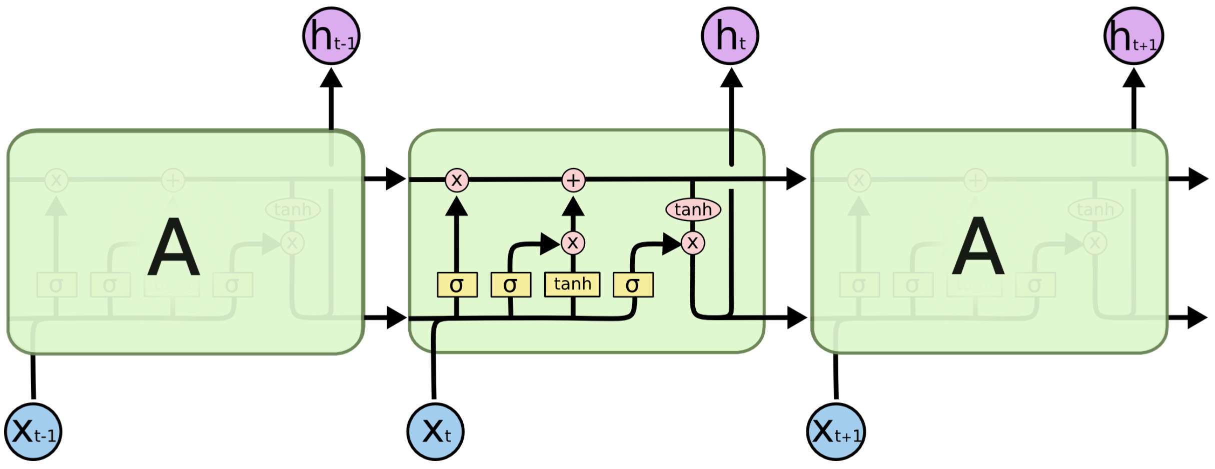

Figure 2: The repeating module in a LSTM [11]

Equations for the LSTM are as follows [3, 11]

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

where

- f_t is the forget gate vector.
- i_t is the input gate vector.
- $\tilde{C}_t$ is the intermediate cell state vector
- C_t is the current cell state vector
-
- o_t is the output gate vector
- h_t is the vector for the output of the hidden layer

Another type of RNN's are GRU's (Gated Recurrent Unit). GRU's are similar to LSTM's but they are simpler, and they also deal with the vanishing gradient problem. GRU's have a reset and an update gate. The reset gate decides if the previous cell's output has an effect and the update gate decides on if the current input has an effect on the next state. The update state and reset states are calculated by passing the input feature and the previous hidden state through a sigmoid function each with different weights. An intermediate memory is produced by passing the input feature and the output of the reset gate through the hyperbolic tangent function. Lastly, the output is calculated by adding two components: the element-wise multiplication of the output of the update gate and the previous hidden state, and the element-wise multiplication of the output of the update gate subtracted from 1 and the intermediate memory. [3, 11, 12] There are different types of the GRU's: Fully Gated Unit, Type 1, Type 2, Type 3. These types differ from each other by using different formulas for the update and reset gates. For example, Fully Gated Unit depends on the input features, previous state and bias term. [13]

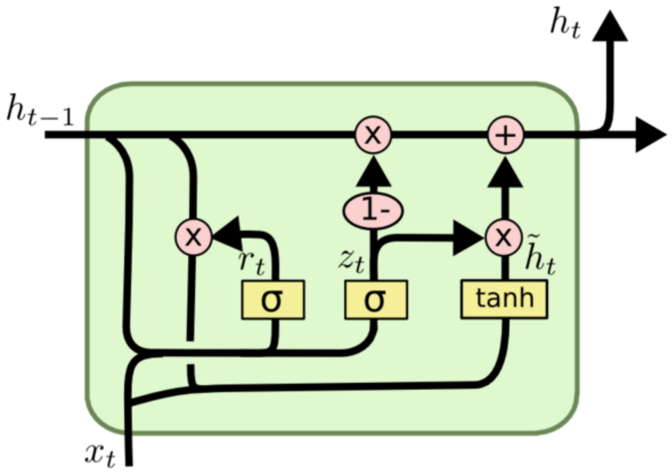

Figure 3: GRU structure [11]

Equations for the GRU are as follows [12]

$$z_t = \sigma(W_r x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_z h_{t-1} + b_z)$$
$$\hat{h}_t = tanh(W_h x_t + r_t * U_h * h_{(t-1)} + b_z)$$
$$h_t = z_t * h_{(t-1)} + (1 - z_t) * \hat{h}_t$$

where

- z_t is the update gate vector.
- r_t is the reset gate vector.
- $\hat{h}_t$ is the candidate activation vector.
- h_t is the output vector for the hidden layer.

To compare RNN, LSTM and GRU, one could say that each of these architectures has its use in different cases. RNN's are faster and require less computation while they have the long term memory problem. LSTM's and GRU's overcome this problem by using a memory unit to store previous activation values. [14]

In our project, we will be using BLEU (Bilingual Evaluation Understudy) score to evaluate the captions our model generates by treating them like translations. BLEU is calculated using brevity penalty and a precision score calculated from the n-grams of the generated captions and the captions from the dataset. The calculations are as follows. [15]

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \leq r \end{cases}$$

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n log p_n)$$

where

- BP is the Brevity penalty
- r is the count of words in a reference translation (one of the original captions from the database R)
- c is the count of words in a candidate translation (a caption generated by our model)
- N is the number of n-grams, we usually use unigram, bigram, 3-gram, 4-gram
- w_n is the weight for each modified precision, by default N is 4, w_n is 1/4=0.25
- p_n is the modified precision

What Has Been Done So Far

As promised in the proposal, we built an InceptionV3-LSTM architecture and trained it. To be more specific, we first downloaded and preprocessed the *Flickr8k* dataset [5]. Here, we tokenized the captions using symbols such as <SOS> (start of sentence), <EOS> (end of sentence), <PAD> (padding) and <UNK> (unknown). We then encoded the images to obtain features using Inception-V3 [1], which was pre-trained on the ImageNet dataset [2]. Finally, we decoded these features to get outputs of captions using an LSTM layer and MLP networks. All in all, we prepared a decent model until the progress demo as promised.

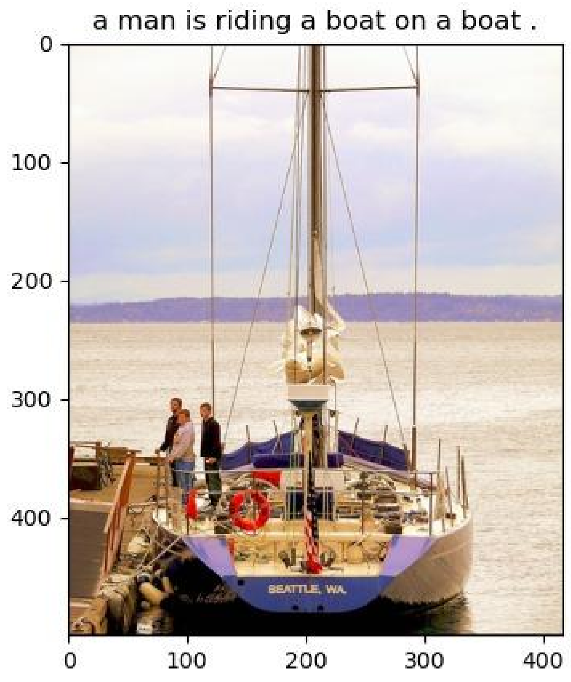

Figure 4: Example captioning output

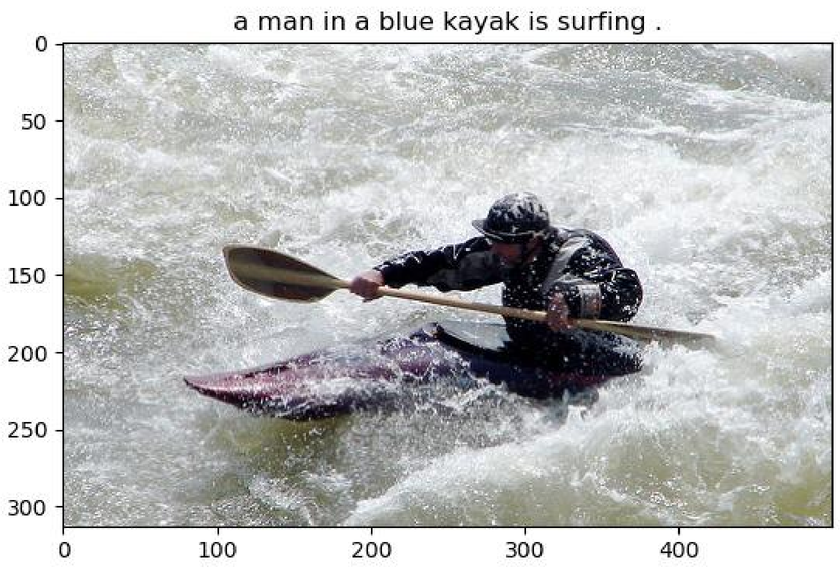

Figure 5: Example captioning output

In addition to that, we used a tutorial on image captioning with Tensorflow to compare our results [16]. This tutorial also used Inception-V3 in the encoding part. However, it used GRU in the decoder part. This GRU was combined with Bahdanau attention which aims to improve the our model in machine translation by aligning the decoder with the relevant input sentences [17]. As a result, we could caption some images but the second model was open to improvements.

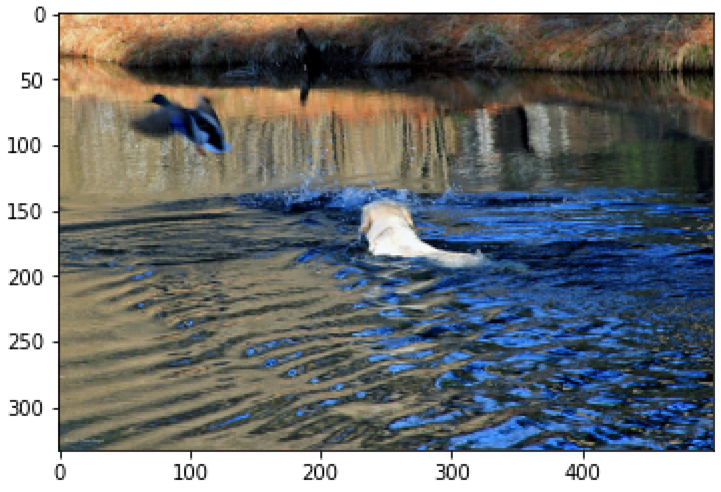

Figure 6: Example captioning input, output is given below.

In the above figure,

- ➔ the real caption: "`a dog chasing a duck in a pond`"
- ➔ the predicted caption: "`a dog swims in the water`"

What Remains to Be Done

After this point, we will be creating more models using different approaches and methods, so that in our final report we can perform a comprehensive comparison between our models based on their performance. The new approaches we are thinking of trying are

- ➔ using other various pre-trained CNN's for the encoding stage
- ➔ creating a custom CNN of our own for the encoding stage
- ➔ trying to optimize our models and playing with hyperparameters
- ➔ if there is remaining time, create a model using transformers instead of RNN
- ➔ trying to improve our model (the one that we created using the tutorial) by trying different RNN architectures combined with different attention mechanisms such as Luong attention mechanism which takes all the encoder's hidden states as input to derive context vector, and possibly different CNN architectures [18].

Division of Work Among Teammates

- ➢ Melisa Taşpınar: Helped write every section of the report, except for the parts about GRU's and the tutorial-model. Code-wise, worked on data preprocessing and on the model with the InceptionV3-LSTM architecture using PyTorch.
- ➢ Yiğit Gürses: Worked on the code for data preparation and the PyTorch InceptionV3-LSTM model. Helped with the report's "what has been done so far" section.
- ➢ Mustafa Göktan Güdükbay: Worked on the code that used Tensorflow InceptionV3-GRU model. Contributed to the report's "background information" section.
- ➢ Musa Ege Ünalan: Worked on the InceptionV3-GRU model code, mainly on adapting it to our dataset. Also helped with the "background information" section.
- ➢ Emin Adem Buran: Worked on the InceptionV3-GRU model constructed on the basis of Tensorflow tutorial. In addition to this, contributed to the report's "what has been done so far" and "what remains to be done" parts**.**

References

[1] "Inception V3 model, with weights pre-trained on ImageNet.," *TensorFlow*. [Online]. Available: https://tensorflow.rstudio.com/reference/keras/application_inception_v3/ . [Accessed: 03-Apr-2021].

[2] "IMAGENET," IMAGENET, [Online]. Available: http://www.image-net.org/index. [Accessed: 03-Apr-2021].

[3] M. Phi, "Illustrated Guide to LSTM's and GRU's: A step by step explanation," Medium, 28-Jun-2020. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 [Accessed: 03-Apr-2021]

[4] W. Liu, S. Chen, L. Guo, X. Zhu and J. Liu, "CPTR: Full Transformer Network for Image Captioning," 2021. [Online]. Available: arXiv: 2101.10804v3 [cs.CV] https://arxiv.org/pdf/2101.10804v3.pdf [Accessed: 03-Apr-2021]

[5] adityajn105, "Flickr 8k Dataset," *Kaggle*, 27-Apr-2020. [Online]. Available: https://www.kaggle.com/adityajn105/flickr8k. [Accessed: 03-Apr-2021].

[6] A. Arya, "Auto Image Captioning," *Medium*, 31-Dec-2020. [Online]. Available: https://medium.com/ai-techsystems/auto-image-captioning-8efcfa517402. [Accessed: 03-Apr-2021].

[7] D. Garg, “Automatic Image Captioning with PyTorch,” *Medium,* 20-Aug-2020. Available: https://medium.com/@deepeshrishu09/automatic-image-captioning-with-pytorch-cf576c98d319. [Accessed: 03-Apr-2021].

[8] “Advanced Guide to Inception v3 on Cloud TPU,“ *Google*. [Online]. Available: https://cloud.google.com/tpu/docs/inception-v3-advanced. [Accessed: 04-Apr-2021]

[9] V. Kurama, “A Review of Popular Deep Learning Architectures: ResNet, InceptionV3, and SqueezeNet,” *PaperspaceBlog,* 5-Jun-2020. [Online]. Available:https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/ [Accessed: 04-Apr-2021]

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *arXiv.org*, 11-Dec-2015. [Online]. Available: https://arxiv.org/abs/1512.00567v3. [Accessed: 04-Apr-2021].

[11 A1] colah, “Understanding LSTM Networks,” *colah’s blog,* 27-Aug-2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 03-Apr-2021].

[12 A4] S. Russell-Puleri, “Gated Recurrent Units explained using Matrices: Part 1,” *Medium*, 01-Dec-2020. [Online]. Available: https://towardsdatascience.com/gate-recurrent-units-explained-using-matrices-part-1-3c781469fc18. [Accessed: 04-Apr-2021].

[13] “Gated recurrent unit,” *Wikipedia*, 29-Dec-2020. [Online]. Available: https://en.wikipedia.org/wiki/Gated_recurrent_unit. [Accessed: 05-Apr-2021].

[14] H. Pedamallu, “RNN vs GRU vs LSTM,” *Medium*, 30-Nov-2020. [Online]. Available: https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573. [Accessed: 04-Apr-2021].

[15] R. Khandelwal, “BLEU - Bilingual Evaluation Understudy”, *Medium*, 25-Jan-2020. [Online]. Available: https://towardsdatascience.com/bleu-bilingual-evaluation-understudy-2b4eab9bcfd1 [Accessed: 04-Apr-2021]

[16] “Image captioning with visual attention : TensorFlow Core,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/tutorials/text/image_captioning. [Accessed: 05-Apr-2021].

[17] G. Loye, “Attention mechanism,” *FoydHub*, 17-Jan-2020. [Online]. Available: https://blog.floydhub.com/attention-mechanism/. [Accessed: 05-Apr-2021].

[18] P. Ganesh, “Image Captioning using Luong Attention and SentencePiece Tokenizer,” *Medium*, 15-Dec-2020. [Online]. Available: https://preetham-ganesh.medium.com/image-captioning-using-luong-attention-and-sentencepiece-tokenizer-2658b0e6f887. [Accessed: 05-Apr-2021].