Image Captioning:
a Show and Tell implementation

Nassim Habbash (808292) *University of Milano-Bicocca*
Final Project for the Advanced Machine Learning course
n.habbash@campus.unimib.it

***Abstract*—Automatic generation of images description is a task linking natural language processing and computer vision, and has seen much activity in the past couple of years. This work illustrates how an encoder-decoder architecture works in generating natural language captions from an image. The model is trained on a smaller image captioning dataset, validated using Bilingual Evaluation Understudy, and hyperparameter tuned, showing promising results given the scope of the project.**

***Index Terms*—deep learning, natural language processing, computer vision, generative model, image captioning, description generation**

I. Introduction

Progress in the last decade in the field of machine translation and computer vision have created a surge of interest in the task of image captioning. The task consists in the generation of a caption in natural language, given an input image. Automatic caption generation can have many impactful ramifications, such as from automatic image indexing to social media image caption generation to help the visually impaired. The task goes beyond just classifying the main object in the picture, but a generated caption must be descriptive of all the objects, their properties, relations and interactions taking place in the scene. Hence, this task merges two different fields of artificial intelligence: visual understanding and language modeling.

Many approaches and variations exist in the literature regarding this task [1], but results are not steadily increasing [2], this may be due the challenging topic and lack of inspection into competing and more recent works in the field, lowering the goals of researchers.

The work presented here implemented a variant of the Neural Image Captioner (NIC) model described in [3], one of the most cited paper in the field according to Google Scholar, with 3435 citations. The model follows an encoder-decoder architecture, working in an end-to-end manner, inspired by neural machine translation. A convolutional neural network works as an encoder, extracting high-level features from an image, and feeding them to a recurrent neural network - the decoder - to generate sequence of words.

(a) Caption: dog running through snow

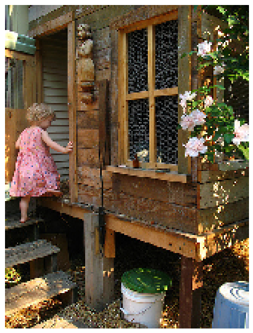

(b) Caption: girl going into wooden building

Fig. 1: Sample images and captions from the dataset

II. Dataset

The dataset used for this work is Flickr8k [4], specifically created for image description ranking tasks and image caption generation. The dataset is composed by 8091 RGB images with varying sizes and resolutions. The images were collected by different Flickr groups, and underwent a manual selection to contain a variety of scenes and situations. Each image is paired by 5 different captions, collected via crowdsourcing, describing the events depicted in the image while also exhibiting some linguistic variations. The dataset contains a grand total of 40453 captions.

The dataset comes with predefined train, validation and test splits. The splits are as follows:

- Train: 6000 images
- Val: 1000 images
- Test: 1000 images

The splits are widely used by the literature using this dataset, and as such will be used in this work as well.

A. Preprocessing

Some preprocessing has been applied on each caption of the dataset. The descriptions have been lowercased, the punctuation, the words shorter than 2 characters and words

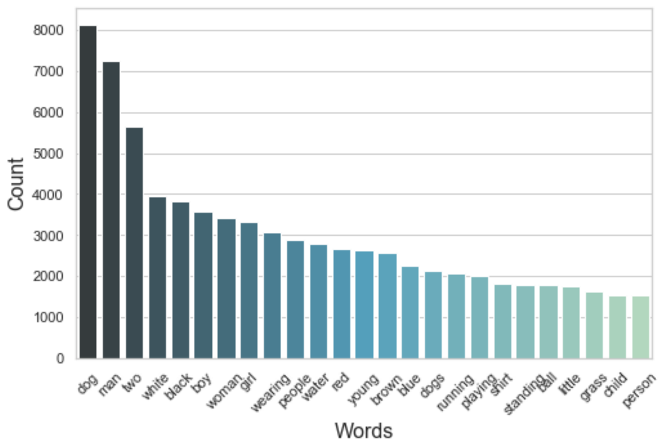


Fig. 2: Most frequent 25 words in the dataset

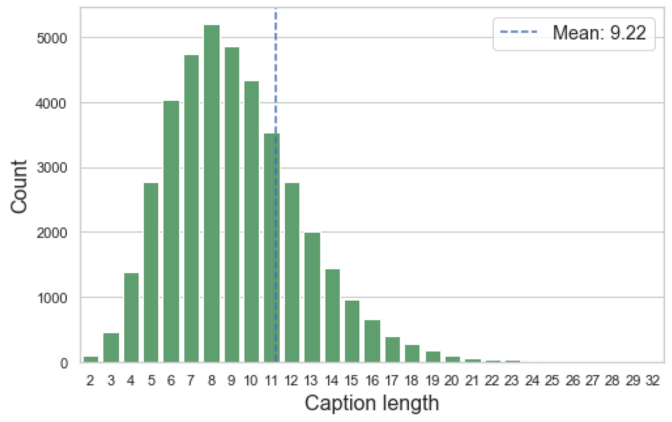


Fig. 3: Caption length distribution

containing numbers in them have been removed altogether. A full stopword removal hasn't been applied as some stopwords were deemed fundamental in the description of the action occurring in the scene.

B. *Data Augmentation*

The images are also augmented to maximize the model's generalization. The following transformations are applied:

- Random crop to 224x224 pixel
- Random horizontal flip
- Normalization by the mean and standard deviation of the ImageNet images' RGB channels

The augmentation occurs online during the model training.

C. *Exploratory Data Analysis*

Understanding the data presented can allows to discover what are the main trends in the dataset, as in what are the most common subjects and actions depicted by the photos. This is, in turn, gives a better understanding of what to expect from the final model trained on this dataset. Figure 2 contains a histogram of the most frequent words in the captions in the dataset. The most common subjects in the dataset are dogs, followed by men, boys and women. The most common verbs are "wearing", "running", "playing" and "standing". Different adjectives appear as frequently, such as colors, size or age indicators. The images seem to contain mostly one or two subjects engaging in simple activities. The captions' lengths appear distributed along a skewed normal distribution, with a mean of 11 words per caption and a mode of 8 words per caption, as shown in Figure 3.

III. Methodological Approach

The goal of the model is to generate a caption given an input image in an end-to-end fashion. This is achieved by using neural networks in an encoder-decoder architecture. The encoder's role is to extract the features of the input and encode them into a fixed-size vector state. This state is then passed into the decoder to start generating words sequences. Intuitively, given an image, the model "translates" it into its description.

A. *Vocabulary building and caption encoding*

A vocabulary V is a word map mapping words to unique indices. This is done to encode the captions to a numeric array to allow neural networks to process them. The vocabulary is built by first counting the frequencies of each word in the captions dataset - effectively extracting its language model - and then adding to the vocabulary the words which are above a certain minimum frequency count. Three special tokens are also added to the vocabulary: ⟨start⟩, ⟨end⟩ and ⟨unk⟩. The ⟨start⟩ and ⟨end⟩ tokens mark the beginning and end of the caption, while ⟨unk⟩ is used for all the words that do not appear in the vocabulary.

As such, the caption

$$\langle \text{start} \rangle \text{ dog running through snow } \langle \text{end} \rangle$$

is transformed into the variable-size vector

$$C = [0, 123, 9482, 1110, 45, 1]$$

B. *Models architecture*

Formally, for a pair of (I, C) of an image I and its caption C, the model maximizes the likelihood $p(C|I)$ as

$$log\ p(C|I) = \sum_{t=0}^{N} log\ p(C_t|I, C_0, ..., C_{t-1}) \tag{1}$$

The conditional probability $p(C_t|I, C_0, ..., C_{t-1})$ is modeled through a Recurring Neural Network, in particular, to avoid the issue of exploding and vanishing gradients, an LSTM network. To allow the LSTM to know about the contents of the image, the encoder acts as a feature embedder. Figure 4 shows the encoder model. it's composed by a pretrained CNN, namely ResNet-101, and an additional linear (FC) and batch normalization layers. These two final layers embed the feature vector in an embedded space of the same size of the words embeddings that will be fed to the LSTM. Given the image I, its feature vector output of $CNN(I)$ and the trained parameters of the linear and batch normalization layers W_{ENC}, the embedded image feature vector I_{emb} is

$$I_{emb} = W_{ENC} \cdot CNN(I) \tag{2}$$

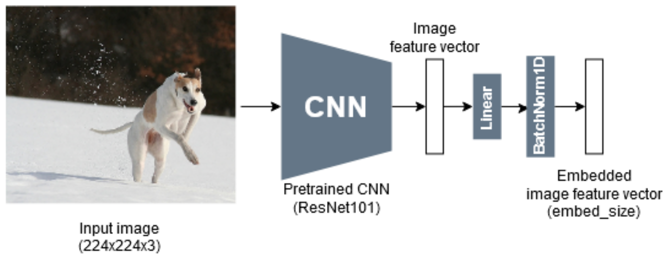


Fig. 4: Encoder model, the CNN block represents a pretrained ResNet-101, with an additional finetuned Linear (FC) and Batch Normalization layers

Following, given the LSTM state h at time $t+1$ as

$$h_{t+1} = f(h_t, x_t) \tag{3}$$

where h_t is the state at time step t, x_t is the input token at time step t and f are the LSTM nonlinearities, then, the following equations represent how the decoder goes from the embedded image feature vector I_{emb} to generating captions tokens

$$x_{-1} = I_{emb} \tag{4}$$

$$p_t = LSTM(x_{t-1}) \tag{5}$$

The model takes I_{emb} as input only once at the beginning, and outputs probabilities for every word in the vocabulary V. The loss is computed as the cross-entropy of the correct word at each time-step with respect to the parameters of the LSTM of the decoder and the linear and batch normalization layers of the encoder, such that:

$$L = -\sum_{t=1}^{N} log\ p_t(C_t) \tag{6}$$

Figures 4 and 5 show visually the structures of the encoder and decoder.

C. Inference

Inference of the caption can be done as either a greedy search or a beam search through the probabilities p over the vocabulary V at each timestep t.

Greedy sampling is done, after feeding I_{emb} to the LSTM, by sampling each word x_t at each timestep as

$$x_t = argmax(p_t) \tag{7}$$

and continuing until reaching the $\langle$end$\rangle$ token or a certain maximum length.

Beam search sampling, on other hand, keeps memorized the top k best words at each timestep, sampling for each following timestep words for each k words memorized at the previous timestep. Reached the end, the search returns the most likely sequence of words computed as sentences.

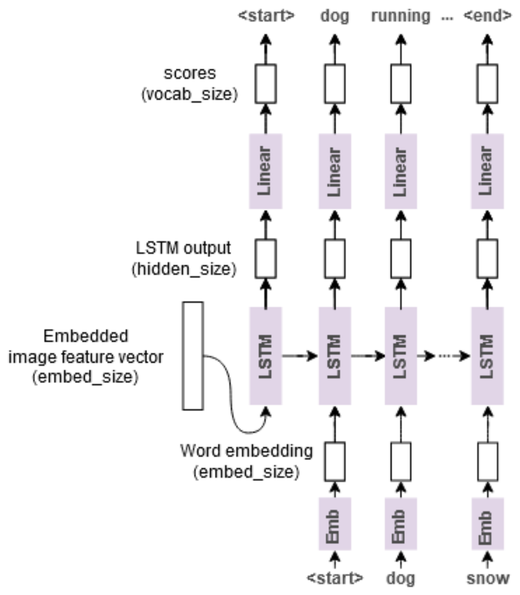


Fig. 5: Decoder model, represented "unrolled". Takes in input the embedded image's feature vector at time $t = -1$, and proceeds to generate tokens from there. Each generated token is then embedded, and passed through the cell until the caption is generated.

D. Training and validation

Training is done using the Adam optimizer on the crossentropy loss described in the previous section. The hyperparameters considered are:

- Batch size: Mini-batching size
- Learning rate: Adam's LR
- Momentum: Batch Normalization layer's momentum
- Hidden size: Units in the decoder's linear layer
- Embed size: Size of the embedding space
- # Layers: Number of layers of the LSTM

One important aspect to notice is how the mini-batching works: captions are of variable lengths, but batches have to be fixed-size tensors. One way to solve this issue is padding captions to the same size with a padding token, but necessitates packing and unpacking sequences at every step. Another solution, and the one applied in this work, is sampling a caption length, where its probability is proportional to the caption's length distribution in the dataset (see Figure 3). The batch produced for a certain iteration then contains all captions of the same length. This approach is taken by [5], and shows its computational effectiveness without degrading performance.

Training keeps track of the loss and the perplexity of the model. The perplexity of the model is defined as $perplexity = 2^{entropy} = e^{loss}$, and captures the degree of "uncertainty" a model has in predicting the captions. Validation keeps track of the same measures for the validation set, but in addition also tracks the BLEU score during training.

BLEU, namely Bilingual Bilingual Evaluation Understudy, is a metric used to compare candidate translations to a reference translation, and measure their accuracy. While mostly used for machine translation, it can also evaluate text gen-

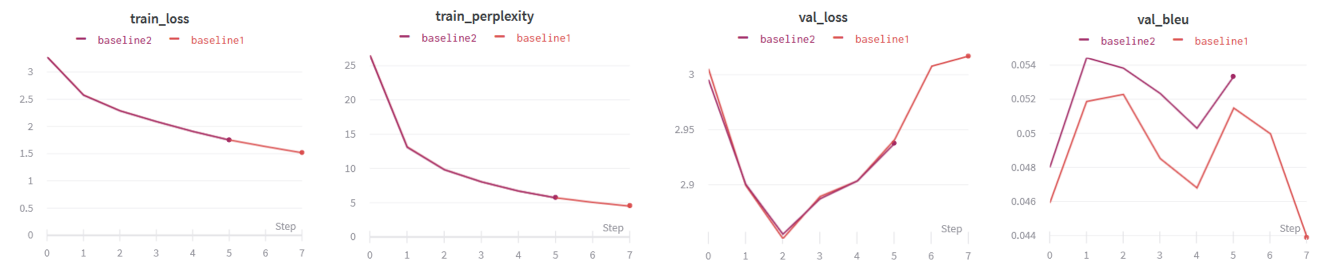


Fig. 6: Baseline (*baseline1* and *baseline2*) experiments metrics, one step corresponds to one epoch

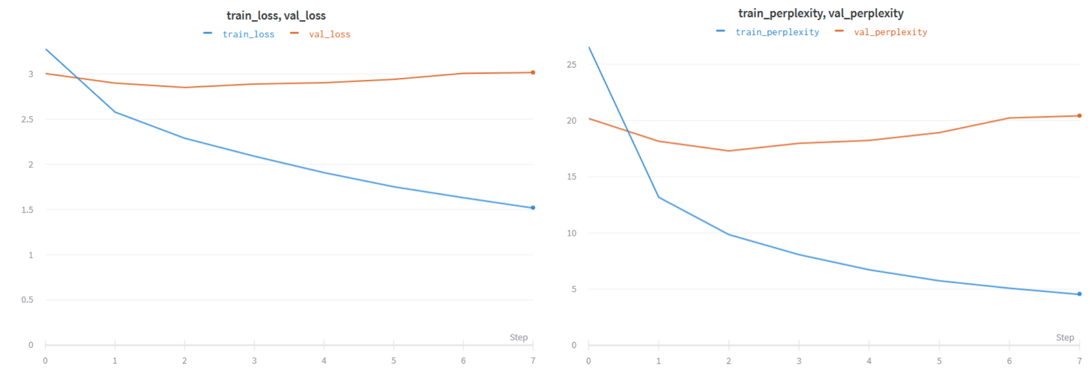


Fig. 7: Train and validation loss and perplexity change for the *baseline1* experiment

eration from a groundtruth. It works by checking how many n-grams overlap between the reference and candidate text, and is defined in $[0, 1]$, where 0 means the text don't match at all, while 1 means the text match perfectly.

An Early Stopping mechanism has been put in place using the validation BLEU score during training, allowing the process to stop if the average BLEU score hasn't improved for n epochs, $n = 3$.

IV. Results and Evaluation

A. Tools

Following are the main tools used for this work:

- PyTorch, deep learning framework in Python
- Weights and Biases, machine learning experiment tracking tool
- Ax, Adaptive Experimentation Platform for hyperparameter tuning

Model training and testing has been using GPU accelleration on a laptop's Nvidia GTX 1660ti.

B. Baseline

The baseline model has been trained on the following hyperparameters:

- Batch size: 32
- Learning rate: 0.001
- Momentum: 0.01
- Hidden size: 512
- Embed size: 512
- # Layers: 1
- # Epochs: 10

The hyperparameter have been taken both from the original paper and from the literature after a round of testing.

The baseline experiment has been executed twice to account for variance of the results. The models have been trained on the train split and validated on the validation split of the dataset. Figure 6 shows the reported metrics for each epoch. Both baseline experiments produce similar results, as the loss and perplexity gradually decrease. The validation loss, on other hand, sharply increases after the second epoch. The validation BLEU score shows a more erratic movement, making it hard to understand its behaviour. Both experiments terminated through the early stop mechanism.

Figure 7 show how the training and validation loss and perplexity change during training of one of the baseline experiments (the other was omitted as it behaved similarly). The validation loss sharp increase is accompanied by the training loss constant descent, possibly indicating overfitting of the model.

The model actual performance has been analyzed by testing BLEU scores on both Greedy Search and Beam Search inferences on the Test dataset. Both inferences BLEU distribution place in the 15th percentile, as Figure 8 shows. There does not appear to be a striking difference in terms of performance between the two, but the Beam search does perform marginally better than the Greedy search, as the peak of the distribution is slightly lower, giving in average more higher-score results than the Greedy search. Overall, the best results have been obtained with Beam search, where the model scores a mean BLEU score of 10 (against 9 with Greedy search)

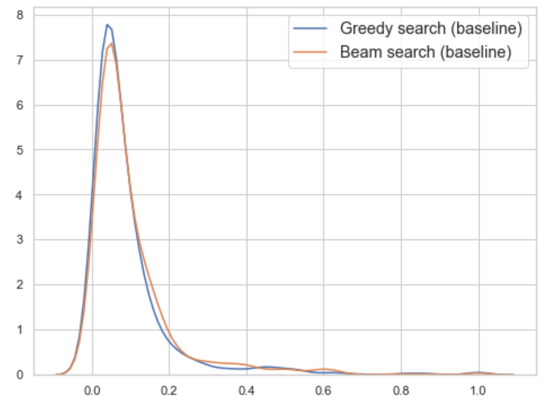


Fig. 8: Baseline BLEU score distribution on the Greedy and Beam searches

C. Hyperparameter tuning

Hyperparameter tuning has been based on the Ax framework. Ax offers different techniques for SMBO, such as Bayesian optimization or Bandit optimization. Bayesian Optimization usually works by creating a surrogate model as a Gaussian Process, and sequentially optimizing the hyperparameters according to an objective function. Ax contains an optimization strategy selector to allow the selection of an optimal strategy for optimization according to the search space. The search space of the model considered is composed of mostly discrete hyperparameters (# Layers, Embed Size, Hidden Size) compared to the continuous ones (Learning rate, Momentum). As such, the strategy selector has chosen to tune using Sobol Sequences instead of GPs, as it would allow for a faster optimization. A Sobol Sequence is a series of quasi-random numbers designed to cover the space more evenly than uniform random numbers. These sequences are used to cover the search space more or less evenly before passing to effective Bayesian Optimization. Following is the defined search space for tuning:

- Learning rate: $range \in [0.0005, 0.002]$
- Momentum: $range \in [0.005, 0.02]$
- Hidden size: $choice \in \{128, 256, 512\}$
- Embed size: $choice \in \{128, 256, 512\}$
- # Layers: $choice \in \{1, 2, 3\}$

Tuning has been run two times with a different objective function to optimize. The first run tried to optimize validation perplexity, while the second run tried to optimize the validation BLEU score. Each run has been given a budget of 5 trials.

TABLE I: Run #1 configurations with corresponding perplexity

Trial #	1	2	3	4	5
Embed Size	512	512	512	128	512
Hidden Size	128	128	512	512	256
LR	1.946e-3	1.455e-3	8.803e-4	1.359e-3	1.938e-3
Momentum	6.988e-3	5.957e-3	1.196e-2	9.105e-3	1.626e-2
# Layers	1	1	3	2	1
Perplexity	20.0	19.97	19.97	19.97	19.95

TABLE II: Run #2 configurations with corresponding BLEU score

Trial #	1	2	3	4	5
Embed Size	512	128	128	128	512
Hidden Size	128	512	256	512	128
LR	7.334e-5	5.577e-4	1.882e-3	7.79e-4	7.269e-4
Momentum	1.31e-2	1.481e-2	1.066e-2	1.183e-2	1.583e-2
# Layers	1	1	3	1	3
BLEU	0.0439	0.05	0.05	0.05	0.05

TABLE III: Best model between runs

Run #	1	2
Embed Size	512	128
Hidden Size	256	512
LR	1.938e-3	5.577e-4
Momentum	1.626e-2	1.481e-2
# Layers	1	1
Perplexity	19.95	19.39
BLEU	0.044	0.05

Both runs found a marginally better configuration at each sequential trial by minimizing perplexity or maximizing BLEU score, as shown from Tables I and II. The budget given does not seem enough to successfully explore the search space, as the increases in the objective function seen have been quite small, and it doesn't look like the tuning managed to find even a local optimum. Table III compares the best models between runs. The model from the second run achieved, again, a marginally better score in both BLEU and perplexity, and as such will be the model used henceforth.

The performances of the better model have been, at the end, compared with those of the baseline in terms of BLEU score for Greedy and Beam search on the test dataset. Figure 9 reports the score distribution for both searches for the baseline and tuned model. While all the model are contained in the same 15th percentile range, the tuned model does perform once again marginally better, obtaining a mean BLEU on the test dataset of 11.

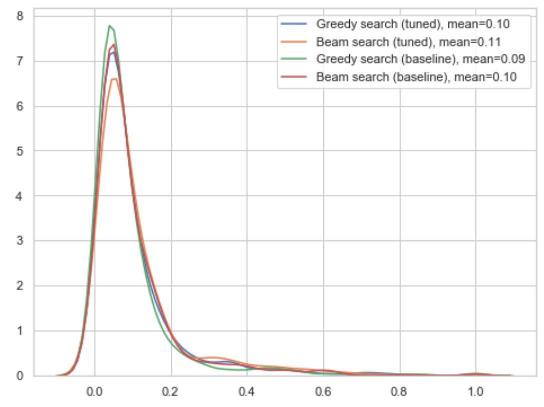
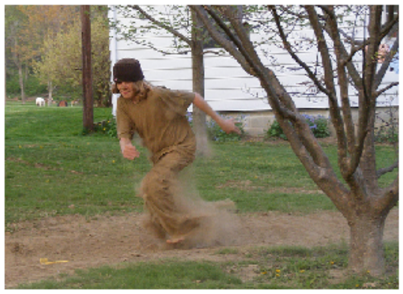


Fig. 9: Performance comparison between the baseline and tuned models

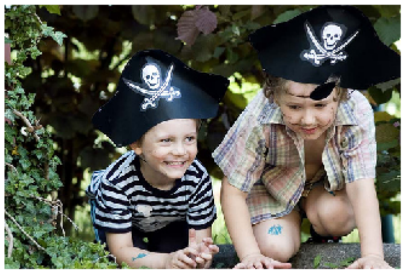
(a) Dog swimming with stick in its mouth, BLEU = 1.0

(b) Woman sings into microphone while playing guitar, BLEU = 1.0

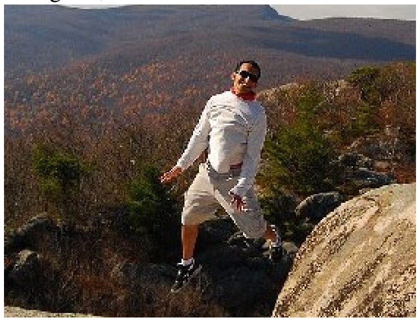
(c) Brown and white dog runs through the grass, BLEU = 0.765



(d) Man throwing stick for dog to fetch, BLEU = 0.0



(e) Man and woman pose for picture, BLEU = 0.0



(f) Man is on the ground by his arms trees, BLEU = 0.0

Fig. 10: Three of the top and three of the bottom captions generated by BLEU

V. Discussion

The goal of the work has been substantially reached. The biggest bottlenecks have been long training times ($> 10h$), hardware limitations, and setting up experiment tracking pipelines, which have somehow capped further exploration through training of the model, but the architecture has been shown to work correctly, as Figure 10 shows. The model so far manages to correctly generate captions for a couple of pictures, but it's of note how, even for pictures with BLEU score close to 0, the model seems to be capable to translate features into words. For example, Figure 10d displays a man running close to a tree, and while the generated caption does not capture the action, it did manage to successfully identify a ”man” and a ”stick”, although the latter was a tree. Figure 10d displays the same pattern to a higher degree. The picture displays kids in pirate costume playing, and while the model does not generate an appropriate caption, manages to generalize that there are effectively two people in the picture in some sort of pose. As such, it may be not entirely appropriate to score these texts as an effective 0, as the model has shown some higher abstraction ability that is not represented by the low scores. BLEU scoring, while widely used, is not the literature standard measure for text generation validation, and more often than not it's paired with different kind of other measures to ascertain its measurements. The results of the experiments show the trickiness in measuring effectively the performances of models such as this. The model's metrics show indications of overfitting the dataset, and this might be due the relative small size of the dataset Flickr8k. State-of-the-art results are often obtain with bigger datasets such as Microsoft COCO or Flickr30k, through much in-depth hyperparameter tuning. The model from the original paper [3] does reach a BLEU score of 27, when trained on MSCOCO, while the model presented here reaches a score of 11, which is less than ideal but shows the effectiveness of the architecture even when trained on subpar hardware and less data. Possible improvements of the present work include:

- Train on faster hardware
- Train on bigger datasets
- Give a higher budget for hyperparameter tuning
- Consider changes in the model's architecture (e.g. Using other pretrained CNN than ResNet, adding individual layers as dropouts)
- Develop a better experiment tracking workflow
- Adding an attention mechanism

VI. Conclusions

Automatic caption generation is a subfield in machine learning that has seen incredible and fast progress in the past couple of years thanks to the progress in both natural language processing and computer vision. Although varied, ML presents many points of contact between neighboring subfields, which allow for breakthroughs to happen at a staggering pace.

The original paper [3] this work is based on is already quite old, as the literature has moved on from simple encoder-decoder generative models, progressing with Attention-based models, where the model is able to ”focus” on certain parts of the picture while generating words, and Generative Adversarial Networks. That being said, this kind of model is still very influential, and does show the power of complex neural network architectures. This, however, comes at the cost of computational expensiveness, as shown, especially when considering AutoML pipelines added to tune the final model. In a day where it's possible to train models via Cloud for a decent price though, and models do actually get more accurate with less parameters (through techniques such as pruning, not explored in this work), computational expensiveness does not seem as big of a roadblock as it may have been years prior. The work has shown satisfying results and is deeply interesting, as it shows how it's now possible to generate text from images in an end-to-end fashion.

References

[1] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," 2018.

[2] R. Staniute and D. Šešok, "A systematic literature review on image captioning," *Applied Sciences*, vol. 9, p. 2024, 05 2019.

[3] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," 2014.

[4] P. Y. Hodosh, Micah and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *Journal of Artificial Intelligence Research 47*, 2013.

[5] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," 2015.