Automatic Image Captioning for visually impaired

Machine Learning Engineering Nanodegree Capstone Project
By Muhammed Khaled

DEFINTION

PROJECT OVERVIEW

image captioning is the task of generating text deceptions for input image which can help in many areas of life such as search engines can generate captions for images they have in their database and when a user inputs a search query trying to find an image of a cat sitting on a couch for example the engine can then search for an image that has that description in the database and return this image to user this could improve search experience for users.

But in our project my motivation was to find a solution that could help people with vision loss have images described for them.

Problem Statement

There is a lot of good solutions in that describes images in English language but there is not many that do so in Arabic so my aim in this project is to build a machine learning model that could be used to help Arabic speaking people who would need this project.

To achieve this goal there are simple steps to follow:

I. Download a data-set that could be used to train a machine learning model with Arabic captions I choose the Flicker8k data-set (more descriptions will come later)
II. Preprocess the text captions to be suitable for deep learning
III. Build a deep learning model and train it on the data
IV. Use the model to generate captions for images and save the resulting captions in mp3 format to be played and listened to

METRICS

The initial plan was to train and evaluate with bleu-score but since bleu-score nature of calculation is not suitable to be passed as a target metric to tensorflow I used Accuracy to train and evaluate and evaluate the accuracy of the predictions with bleu and that makes since since when we use bleu we want to see how accurate is our generated captions to expert references.

ACCURACY

I. $\textbf{accuracy} = \frac{\textbf{number of correct prediction}}{\textbf{total number of predictions}}$

II. Categorical accuracy Class in tf.keras was used as it's an implementation more suitable for the model I used as my model returns predictions as logits when computing the the argmax of logits it give us probabilities of classes and could be used just like normal accuracy in classifiers (more on model architecture later)

BLEU

I. BLEU (BiLingual Evaluation Understudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text (in our case Predicted Captions) to a set of high quality reference translations(in our case label captions in our data-set) predictions on the test Set can be evaluated by using Bleu-Score:

II. It was used to evaluate the test-set predictions

$$\begin{aligned} Geometric\ Average\ Precision\ (N) &= exp(\sum_{n=1}^{N} w_n\ log\ p_n) \\ &= \prod_{n=1}^{N} {p_n}^{w_n} \\ &= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}} \end{aligned}$$

$$Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}$$

$$Bleu\ (N) = Brevity\ Penalty \cdot Geometric\ Average\ Precision\ Scores\ (N)$$

ANALYSIS

Data Exploration

For the project I choose the flicker8k data-set for image captioning, Original data file of the flicker contains

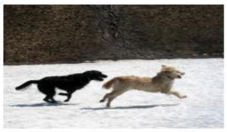
1. **Flicker8k**: contains 8091 jpg images images are placed in a single folder
2. Along with the images folder we have 4 .txt files
    a) The Flickr8k.arabic.full.txt where each line in this file is
        I. <image-name> <#number(1-4)> <caption>
        II. The number represents expert judgment of how accurate is the caption
    b) Next we have 3 files
        I. Flickr_8k.trainImages contains 6000 images with their captions to be used for training
        II. Flickr_8k.devImages contains 1000 images with their captions to be used for validation
        III. Flickr_8k.testImages contains 1000 images with their captions to be used for testing

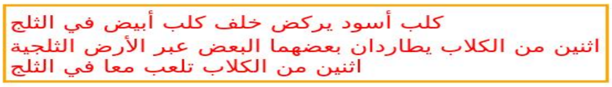
For each image we have 3 corresponding captions that totals out to 18000 data points

Original flicker8 data-set with English captions can be found on Kaggle here

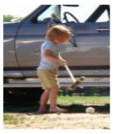
For the Purpose of our project thanks to this Paper we can have the same data-set with the same images but the captions are in Arabic

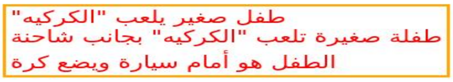
Data sample



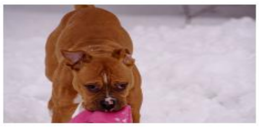


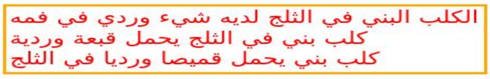
كلب أسود يركض خلف كلب أبيض في الثلج
اثنين من الكلاب يطاردان بعضهما البعض عبر الأرض الثلجية
اثنين من الكلاب تلعب معا في الثلج





طفل صغير يلعب "الكركيه"
طفلة صغيرة تلعب "الكركيه" بجانب شاحنة
الطفل هو أمام سيارة ويضع كرة

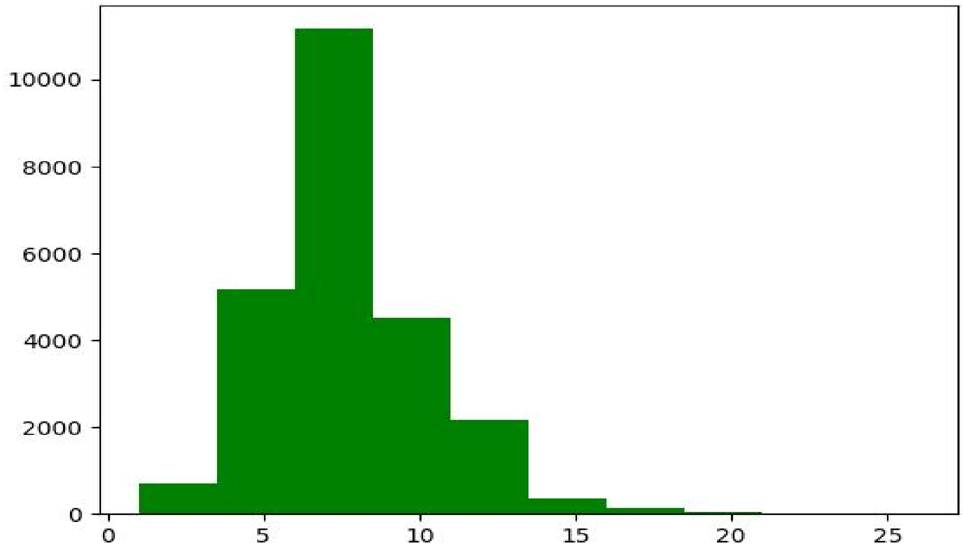




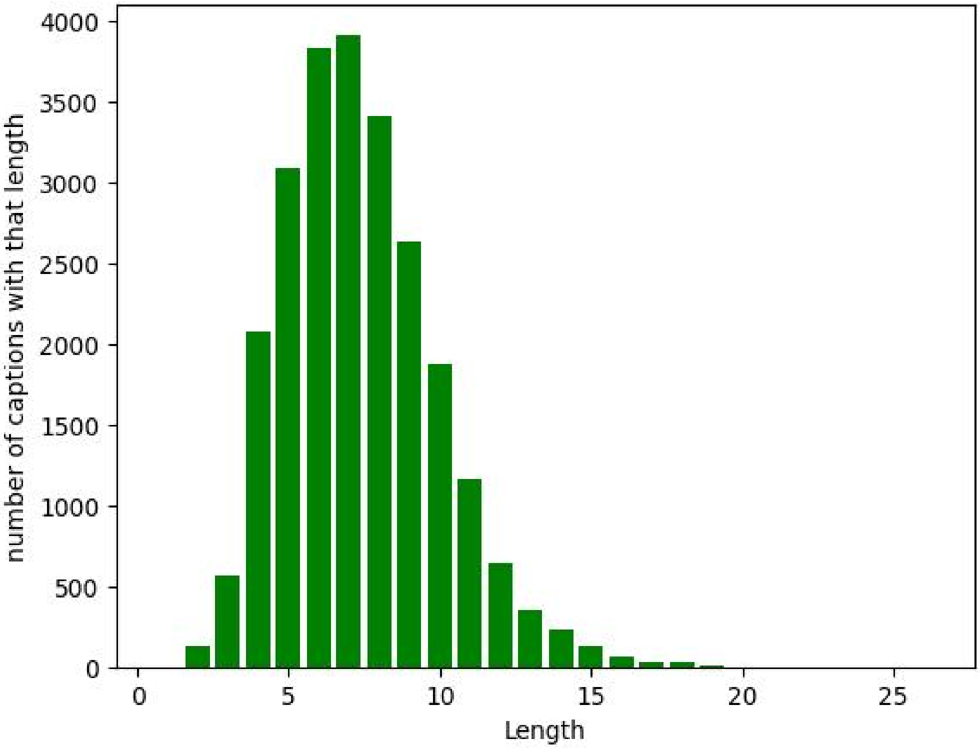
الكلب البني في الثلج لديه شيء وردي في فمه
كلب بني في الثلج يحمل قبعة وردية
كلب بني يحمل قميصا ورديا في الثلج

EXPLORATORY VISUALIZTION

Exploring the captions data revealed that sentences have varying lengths the following histogram shows that most sentences lengths distributed between values 7-15



Calculating the number of occurrences of each length



We could see that few captions have length >15 and largest length will be 20

ALGORITHMS AND TECHNIQUES

The task is split in two parts extracting features from images data and predicting captions or sequences and thus two types of models is used for each task this type of modeling is know as Encoder-Decoder Architecture.

Encoder

i. A VGG16 model was used using the power of transfer learning we could take a pre-trained powerful model capable of extracting features and classifying them and use it for our needs

ii. The VGG16 model consists of many convolution and pooling layers followed by fully connected networks and in the end a classifying Layer with 1000 units since it was trained on image-net to classify different object in images

iii. To use it for our case we are not interested in final classification layer so I excluded the classification layer and add my own to output a feature representation of shape (1,4096) that could be used by Decoder

Decoder

I. Decoder makes use of LSTM units (long-short-term-memory) cells and a stack of dense layers and Embedding layer

II. The LSTM units process sequences of data and keeps information in their memory cells, embedding layer computes representation of words in higher dimension and Dense layers in the end outputs prediction of the next word in the sequence

[1]

Benchmark Model

We will take the encoder-decoder model built in this paper which is the paper we obtained the data from as benchmark for our work in project as this model is trained on the same data we will train our model on.

The authors model managed to get a Bleu-Score of 0.35 so this what i will base the model performance on.

[1] **data citations**: M. Hodosh, P. Young and J. Hockenmaier (2013) "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artifical Intellegence Research, Volume 47, pages 853-899

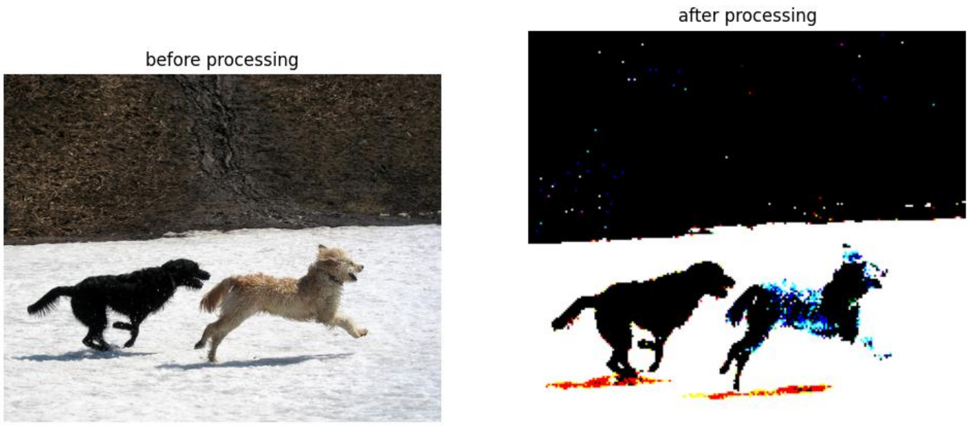
METHODOLOGY

DATA PREPROCESSING

The data processing is divided in two parts

Processing images

I. using tensorflow io we load the images from the disk and resize the images to shape [224,224]
II. Apply the vgg16 preprocessing function to the images which prepares the data with focusing and extracting relevant parts of the image
III. Example of preprocessing of image



Processing Captions

To prepare the captions to be used with Decoder model the following steps has been made

I. Getting the number unique words in captions to build a vocabulary of words
II. Filtering the vocabulary with words that occur more than 5 times in captions to yield in the end a vocabulary of 4089 words

III. Cleaning the Arabic text by striping harakat reducing tashkeel normalization of letters such as the different forms of this letter (ا أ إ) to single shape and lemmatization of words

IV. Converting words to indices using to turn a caption into sequences of ids

V. Padding sequences to max length so all captions have the same input length to model

VI. Add [بدايه] [نهايه] to start and end of sequence to denote start and end

IMPLEMENTATION

the process to implement the model went through the following steps

- ✓ Loading the image files from disk and preprocessing them as explained above.
- ✓ Since encoder is just pre-trained model and we don't do any training on the images, images are passed to the encoder to extract the features vectors for each image and then saved into image_features dictionary and dumped to disk to save time.
- ✓ Image features dictionary contains key image name and value feature vector extracted with encoder vgg16 .
- ✓ Captions are preprocessed as explained earlier and for each tokenized padded caption the output is (1,20) long sequence.
- ✓ Creating a tensorflow dataset with images and captions.
- ✓ To pass the data to the decoder we need to give it start of the sequence which is start word and the next word in the sequence as label and the decoder job is given the first word is to predict the next in sequence, in order to achieve this we shift the input sequence to the right and get the output sequence.
- ✓ As a result of the shift each (1,20) long sequence will output (19,20) output multiplied by 3 that's a lot of data to fit in memory so we use generator function and fit with tensorflow from generator.
- ✓ Build the decoder architecture in tensorflow and start training.

REFINMENT

The usage of pretrained Word Embedding and setting training = False increased model performance with fewer epochs than training my own embedding layer

In the beginning I used mobilenetv3 model as the encoder to for simplicity which achieved lower accuracy 40% than the current accuracy then I moved to VGG16

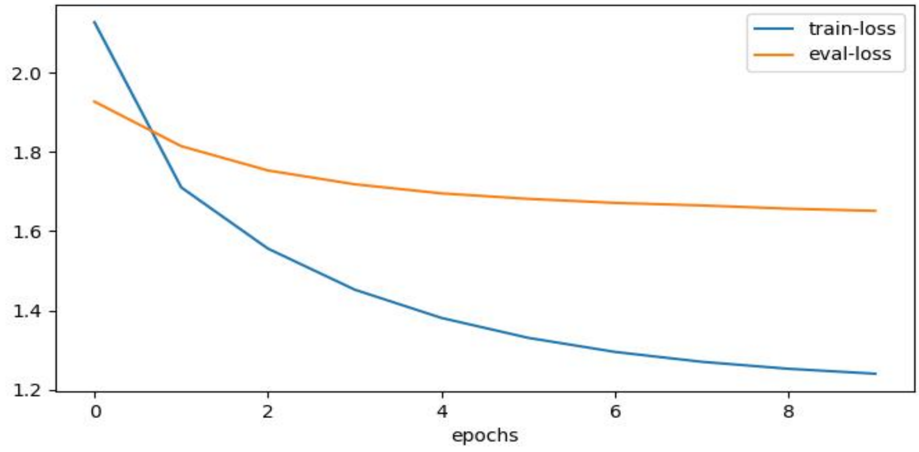
When first using VGG16 I used only the convolution layers of the model and added my own stack of layer which included dense and flatten, pooling layers to output feature vectors for images the results were better than the first benchmark of mobilenetv3 model then I tried using trained dense layers in the VGG16 model instead of adding my own stack only excluding the final Classification Softmax layer and the results were better without having to add but one dense layer

The addition of dropout layers to regularize the model and reduce overfitting and learning rate-scheduler, scheduler helps the model start fast and take large steps to decrease the loss and as it’s going through the epochs it decreases the learning rate to avoid model divergence

RESULTS

MODEL EVALUATION AND VALIDATION

As stated mentioned earlier that we have a dev-set and test-set each contains 1000 images each image has 3 captions the same preprocessing discussed earlier was applied to the dev and test set and they were used to evaluate the model during training a custom callback was used to output predictions captions for a test image and bleu-scores to monitor how the model is doing during each epoch as it’s training
The following figure shows loss plots through the epochs



We can see from the plots that the eval loss din’t improve much after epoch 5 so it’s better to just train for 5 epochs or get more data.

VALIDATION

Hyperparameters of different layers in the model was choosed based on what gave the best results on the validation scores and they are as follows

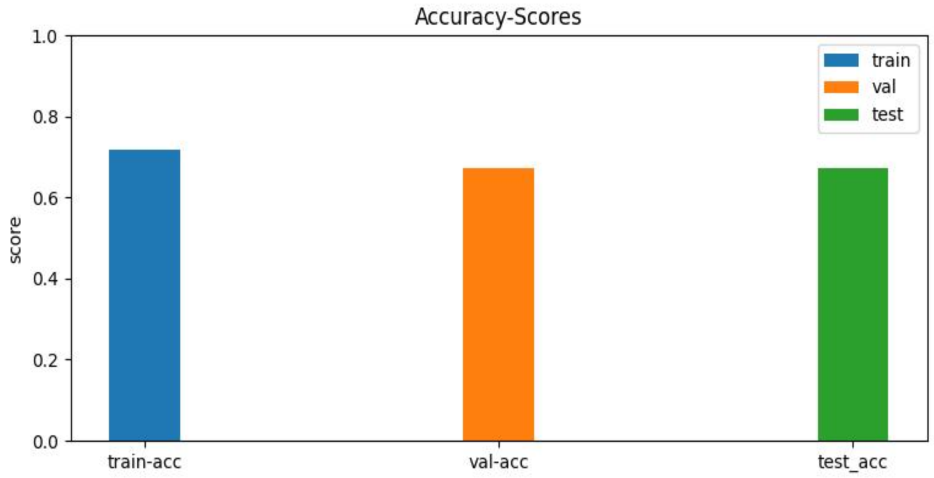
I. Encoder was selected as is as explained in refinement stage using VGG16 pre-trained excluding final softmax and connecting the input feature vector to one Dense Layer with units = 256 with activation = 'tanh'.

II. A dropout layer was added with rate = 0.3

III. Decoder hyperparameters of different layers are as follows
   - ✓ Embedding Layer number of output dimension = 256 and mask_zero=True
   - ✓ LSTM number of units = 256
   - ✓ Dropout with rate = 0.3
   - ✓ 2 Dense layers in the output after the LSTM the first Dense Layer has units =256 and activation = 'relu'
   - ✓ The final prediction layer with units = 4089 and activation = 'softmax'

EVALUATION

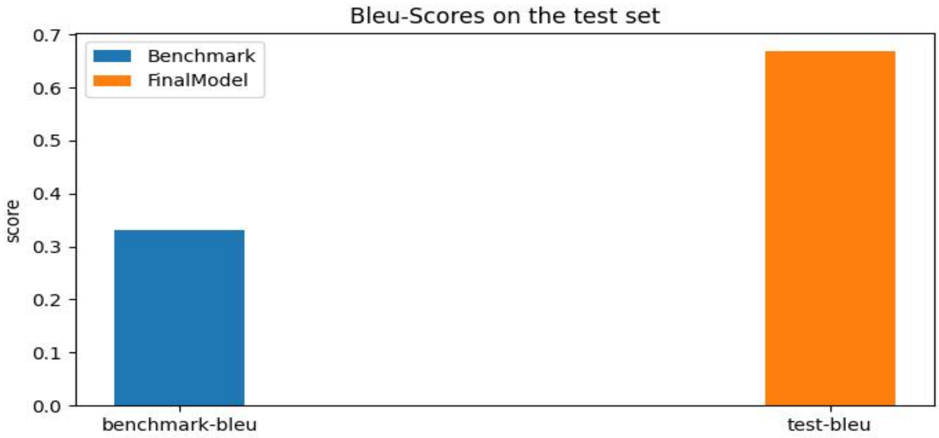
Predictions were made and captions were generated on the test set using bleu-score to evaluate the how close is the predicted captions to the actual captions, the model score 0.66 bleu score and as for the accuracy scores the following figure shows scores on train val test



We could see that model accuracy is acceptable for only 10 epochs of training

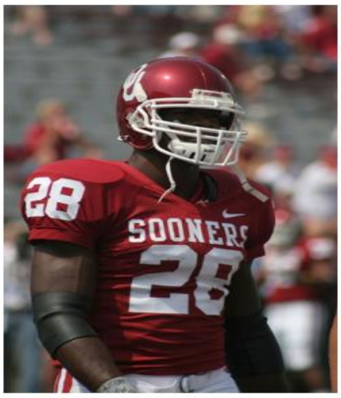
JUSTIFICATION

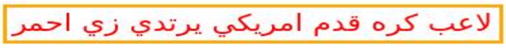
Comparing the model results to the benchmark was done by comparing their bleu-score accuracy on the test set since the test set is the only data that the model never saw before so doing well on the test set will prove model robustness the following figure show comparison between final model and benchmark model



The model managed to get far better score than the benchmark model with the help of the use of LSTM units dropouts and pre-trained Word Embeddings which improved model accuracy drastically.

Example of model output predictions





لاعب كره قدم امريكي يرتدي زي احمر

Predicted Caption: American football player wearing a red suit

REFERENCES

https://arxiv.org/abs/1502.03044
https://www.tensorflow.org/tutorials/text/image_captioning
https://github.com/bakrianoo/aravec
https://www.researchgate.net/publication/340044948_Resources_and_End-to-End_Neural_Network_Models_for_Arabic_Image_Captioning