AML Final Report: Image Captioning

Group 25

Introduction

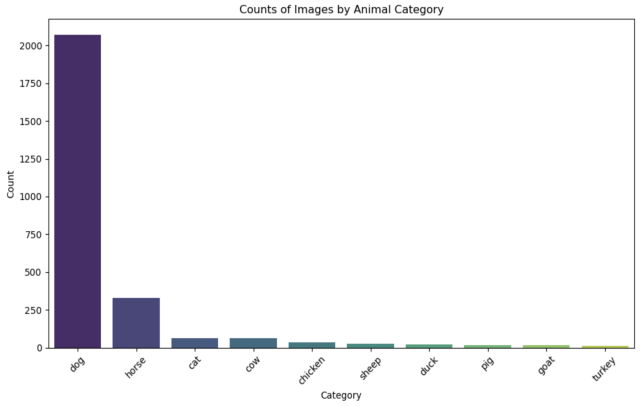
Our project aims to caption images containing animals. We utilize deep-learning techniques from computer vision and natural language processing to achieve this. Our dataset consists of pictures of animals from the famous Flickr30k dataset. We demonstrate that our network can learn to integrate encodings of both image and text data and generate captions informed by image embeddings.

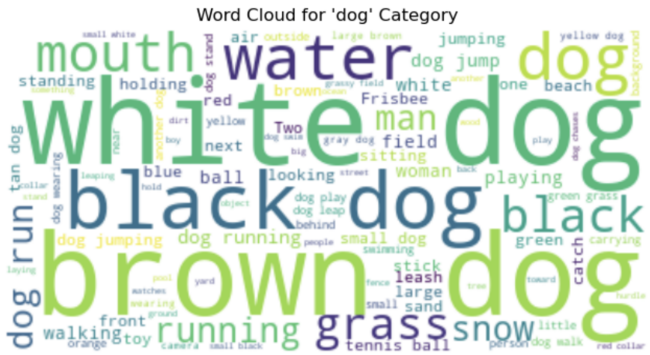
Summary of Dataset

We work with the Flickr dataset that has 31783 images in total, each with five captions. The dataset consists of annotated images, of various categories. However, for our use case, we will be focusing on the categories of Animals, particularly - ***Dog, Cat, Cow, Horse***. After filtering out images relevant to these four categories, we only had 9403 captions and 2494 images.

Exploratory Data Analysis

From our Deliverable 2, we obtained important information on the distribution of animal categories and the common words in each category. Our main visualizations are included below, which is a bar chart of the number of samples per category, and the word cloud for our majority 'dog' category.





Data Pre-Processing

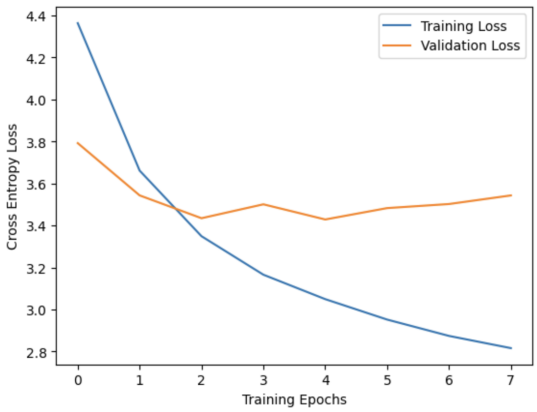
- We cleaned our textual data by making all characters lowercase, and removing all numbers and punctuations.
- Since the dataset is imbalanced, we use Stratified splitting for the train-validation-test-split.
- We downloaded the VGG16 architecture together with its pre-trained weights from ImageNet. This allowed us to focus training solely on the caption generation in the recurrent layers. The keras library in TensorFlow also includes a function for pre-processing images for the VGG16 network. This module was used to resize and normalize images.
- Each image was then passed to the feature_extractor which opens the image, passes it to the VGG16 preprocessor, and finally obtains embeddings on the processed image from VGG16.

Our Model

- This will consist of a CNN (specifically the VGG16 pre-trained model) and an LSTM architecture that we will attach to the penultimate layer of the CNN. We are building and training this LSTM from scratch, and hence will require pre-processing steps to deal with the captions.
- VGG16 is composed of 16 layers that have weights. This includes 13 convolutional layers, followed by three fully connected layers. The convolutional layers use small (3x3) filters, which allow the network to capture fine-grained patterns in the image. The ReLU activation function is used by the model.
- Given the success of VGG16 in image recognition, its weights from training on the ImageNet dataset were imported. This is crucially important since all the animals in our dataset are included in the 1000 categories of ImageNet and so the pre-trained model was able to identify these animals without need for further training.
- A basic captioning network was designed using LSTM layers. After encoding the captions using our tokenizer, we passed the encoded captions to an embedding layer to transform them into the text embedding space. Once the encoded captions were in the embedded space, they were passed to the LSTM layers, so learning of captions could happen. LSTM layers are known for capturing long-term dependencies and as such, are well-suited for caption generation where such dependencies are essential.
- Given the small size of our dataset, and the massive number of trained parameters (over 6 million), overfitting was a serious challenge that we were concerned about. To avoid overfitting, we included a validation set during the training process and enabled early stopping with a patience counter to halt training if validation loss fails to improve over 3 epochs.
- Furthermore, to mitigate the effects of overfitting, we added dropout layers in nearly every part of the network. A dropout layer was added to the image embeddings obtained from VGG16 with a dropout rate of 25%. A dropout layer with a rate of 25% was similarly applied to the caption embeddings. Finally, and most crucially, a dropout rate of 35% was also applied inside the LSTM units to avoid overfitting to the training captions.

Training Loss vs Validation Loss Plot

The model was set to train for 50 epochs, with a batch size of 64. The optimizer used for learning rate updates was Adam. Since early stopping was enabled, training stopped at only 8 epochs. Due to the callbacks, the optimal model weights were saved. These weights can be found as model_weights.h5 on the shared GitHub repository.



Test Metric

Since our model generates a new caption for every image it encounters, we cannot use vanilla metrics such as accuracy, f1 score, etc. Instead, we will be using the BLEU (Bilingual Evaluation Understudy) score to evaluate the quality of our generated caption on the test dataset. We chose this metric because it is language-independent and easy to understand (calculated by converting the predicted and actual caption to its n-gram form and computing the sum of fractions of grams that coincide). It lies between 0 and 1, with a higher score the better. Since the calculation of the BLEU score is computationally expensive, performing it on the entire dataset would be nearly impossible without relying on a powerful GPU. Since these resources weren't available, we provided a few samples from the test data, together with generated captions, actual captions, and the BLEU score.

Example 1: (dog)

```
Generated Caption: dog running through the water with a stick in its mouth is running through the water with a stick
Actual Captions:
   a spotted black and white dog splashes in the water
   a black and white dog is running though water whilst bearing its teeth
   a black and white dog is running and splashing in water
Average Bleu Score:  0.28953074057302325

Highest Bleu Score:  0.38403413539641235
```

Example 2: (horse)

```
Generated Caption: horse jumping rider in a rodeo rodeo rodeo a rodeo a rodeo is riding a horse in a rodeo
Actual Caption:  ['a bucking brown horse next to a falling cowboy']

Bleu Score:  0.16840394156133062
```

Future Work

The CNN+RNN architecture for image captioning is heavily reliant on the size and diversity of the data. As such, with a dataset as small as ours, a high accuracy would be nearly impossible. The model could be improved significantly if images are filtered from other datasets, such as COCO, and combined together. However, this would create a large dataset that would require high memory not available for free on platforms like Google Colab. The LSTM architecture could be improved further to learn more details about dependencies in the captions. However, that would again require more computational power not available on free tier versions of most platforms like Google Colab.