Team: Kerim Kochekov
Group: B18-DS02
Github link: https://github.com/KerimKochekov/PMLDL-Project-Image-Text-Matching

Project name: Image-Text Matching
Paper inference: Order-Embeddings of Images and Language [5]
Datasets: Flickr8k[1], Flickr30k[2]
Used frameworks in Python: Pytorch, Keras

Goal: My main goal for implementing this project is to match images and texts somehow in the same embedding space. Since they are different particles that can not be compared easily, so I wanted to apply somehow Cross-Modal retrieval[6] idea to project both of the media into the same spaces. We can easily see that the naive idea of turning the image into vector space with CNN and turning text into another vector space with RNN can not be compared. That is why I tried to develop a loss function that can compare bimodal architecture and penalize each Encoder to encode images and text vectors in the same space.

My main contribution: Own designed Triplet loss in Cross-Modal for training two different models (Image and Caption) in the same modal space.

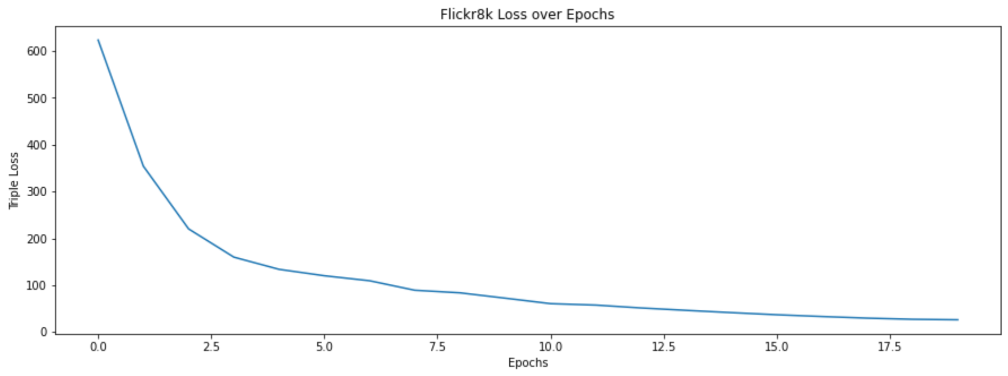
Analyzed pre-trained models for image feature extraction:

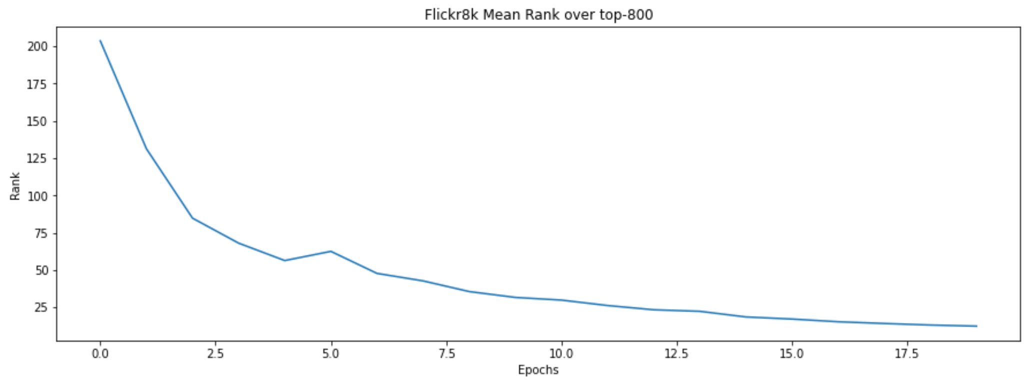
- The 16-layer VGG network from Simonyan and Zisserman, because the model is beautiful, powerful, and available with Caffe. [3]
- The 22-layer VGG16 model in Keras with pre-trained weights on NVIDIA Tesla K80 vgg16_weights_tf_dim_ordering_tf_kernels.h5 [4]

Architecture:

- **Image Encoder:** Feature extraction with VGG16 to get a 4096 sized embedding vector to represent images. Then encoder applies a linear transformation to the incoming data: y = xA^T + b, with the dimension (4096, 1024).
- **Caption Encoder:** Feature extraction with one-hot encoding on generated vocabulary word indexes. Then encoder applies a linear transformation to the incoming data: y = xA^T + b, with the dimension (vocabulary_size, 300) and applied GRU with the dimension (300, 1024) to represent the image and caption in the same space with the same size.

Performance:

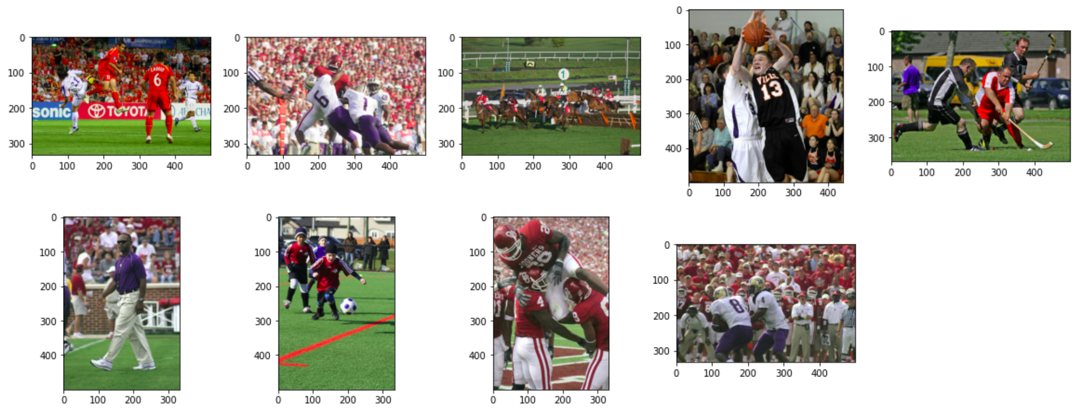


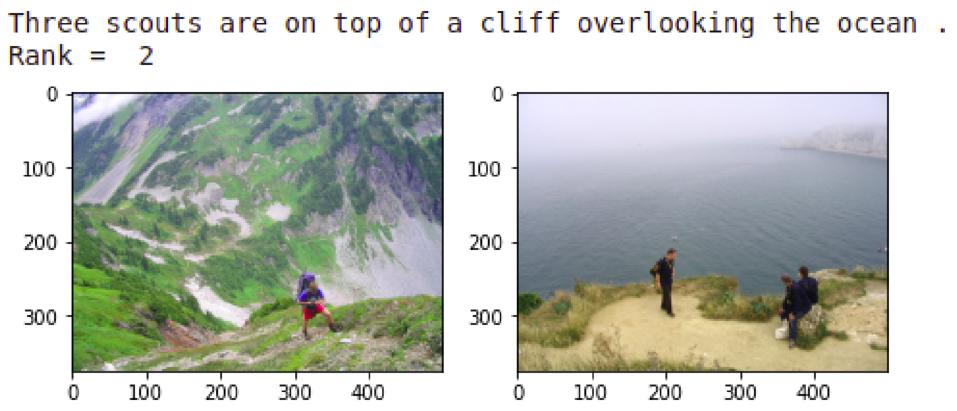


Test:

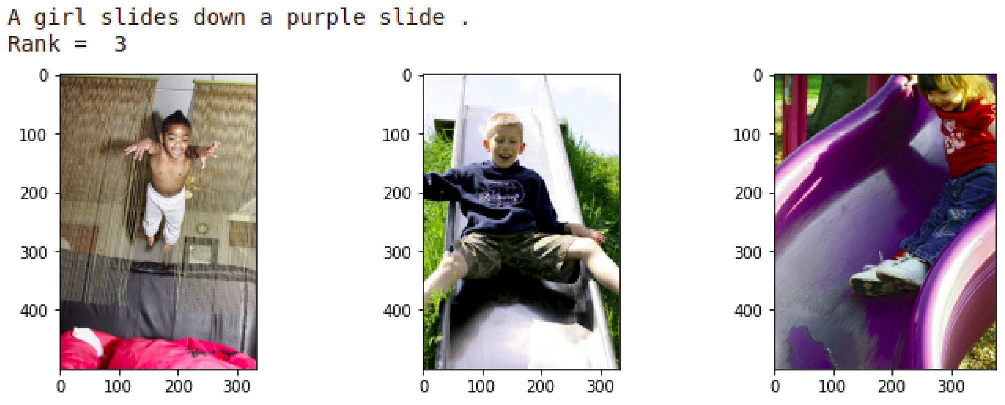
Search image with caption, it can be also applied inversely (search caption with image)

```
 The football players wear purple pants , gold helmets and white jerseys .
 Rank =  9
```





```
Three scouts are on top of a cliff overlooking the ocean .
Rank =  2
```



```
A girl slides down a purple slide .
Rank =  3
```

Hyperparameters:
Optimizer: Adam
margin_loss = 0.05
image_feature_dim = 4096
encoder_dim = 1024
batch_size = 256
word_embedding_dim = 300
learning_rate = 0.001

How to run:

```
bar@foo$:~/python3 prep.py
bar@foo$:~/python3 train.py
bar@foo$:~/python3 eval.py
```

Details:

- Preprocessing(done by prep.py): The first step is to read and normalize read features from all the images. These features will be used for all further processing.
- Training(done by train.py): Loads the image-encoder and the sentence-encoder and trains the model using Order Embedding loss. No negative mining is being done in this code. The hyperparameters have been set according to the Research Paper.
- Testing(done by eval.py): Finds the mean retrieval rank against the test dataset.

Check utils.py for utility function definitions and model.py for encoder architecture.

References:

1. https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_Dataset.zip
2. https://www.kaggle.com/hsankesara/flickr-image-dataset/download
3. http://www.robots.ox.ac.uk/~vgg/research/very_deep/
4. https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5
5. https://arxiv.org/pdf/1511.06361.pdf
6. https://ieeexplore.ieee.org/document/8953669