Multimodal Image Retrieval System using Deep Semantic Common Embedding Space

Submitted by

Amit Bahir: (2017UGCS044)
Purusharth Verma: (2017UGCS066)
Harshal Desai: (2017UGCS086)

Under the Supervision of

Dr. Vinay Kumar

Department of Computer Science and Engineering

National Institute of Technology Jamshedpur

Abstract

Information retrieval, recovering information from a database of documents, has been a task prevalent in the field of computer science for a long time now. Several approaches have been made in designing systems for image retrieval involving annotating and storing metadata along with the images. Carrying out search of relevant documents has had several approaches involving string matching but these approaches do not capture semantic visual information associated with the images.

We have developed an end-to-end image retrieval system which creates a common embedding space between images and textual modality and thus is not only able to perform text to image search but also image to image search. We will show how the system can be used to make queries which could be either single keywords or sentences with actual meaning and get all the top-k semantically relevant images to the particular text query. Also, it is able to capture visual information in a given image query and return the top-k relevant images. Using this approach we achieve average accuracies of 74.46, 89.72 and 93.18 for exact image matches in top 1, 5, and 10 results respectively and, accuracies of 99.86, 98.26 and 96.26 for top 1, 5 and 10 results using a similarity threshold of 0.5 on the Flickr 8k dataset [1].

Table of Contents

List of Figures

List of Tables

1 Introduction

The traditional approaches in image retrieval involving searches based on literal string matching are inefficient due to the following reasons: first reason being this data needs a lot of human intervention and proper annotation with correct textual metadata. The image data itself being so large in size, the requirement of human intervention in the annotation task would limit how many tags could be added to an image thus limiting the actual semantic correlation between the tags and the image. The second reason being these search algorithms do literal string matching which limits search in many ways. The search query must be a single token or if the query is a sentence, it must be broken into tokens and the correlation between tokens may not be considered. For. example: For a textual query “A girl holding a flower”, the literal string matching algorithm would tokenize the sentence, preprocess it by removing the stopwords like a and the tokens would be searched for on the basis of their frequency in the image tags or metadata. The issue with this would be the images having ‘girl’ and ‘flower’ with high frequency would be ranked higher but the image where a girl is actually holding a flower may not be of relevance to the algorithm. Another reason this would be inefficient would be where the query says “fruits” but the images with tags even just “fruit” would not be considered or images where actual ‘fruits’ “mangoes” and “oranges” are actually present. A solution to that would be to lemmatize the text to perform search on the root but text where the lemma is the same as the token such as “runner” will have the same root word would not be a result for query “run” but is still relevant.

To counter these issues, a model which is able to have a clear understanding of the language of the query must be used. Hence, we develop a multimodal image search system by creating a common semantic embedding space which projects the visual features of the images in the same space as the language model and thus is able to capture the semantic aspects of the images relevant to the textual query. We project the images to a high dimensional space and thus when a query is made, all nearest neighbours based on the angular distance would be retrieved. Another advantage this system gives is the

search query is not just restricted to textual modality but now extended to images too as the search is now between representative vectors of the documents and not the actual document.

2 Background and Motivation

Retrieval of images based on their visual attributes is a difficult task. The metadata based approaches are limited by the annotations given to the images and thus are unable to capture the semantic aspects of the images thus making retrieval difficult. These approaches deal with only the textual annotations which leads to an incomplete model which does not leverage the visual aspects of the images involved and thus limiting the search to only a single modality i.e. text. Hence, an approach which considers all the modalities involved must be looked for.

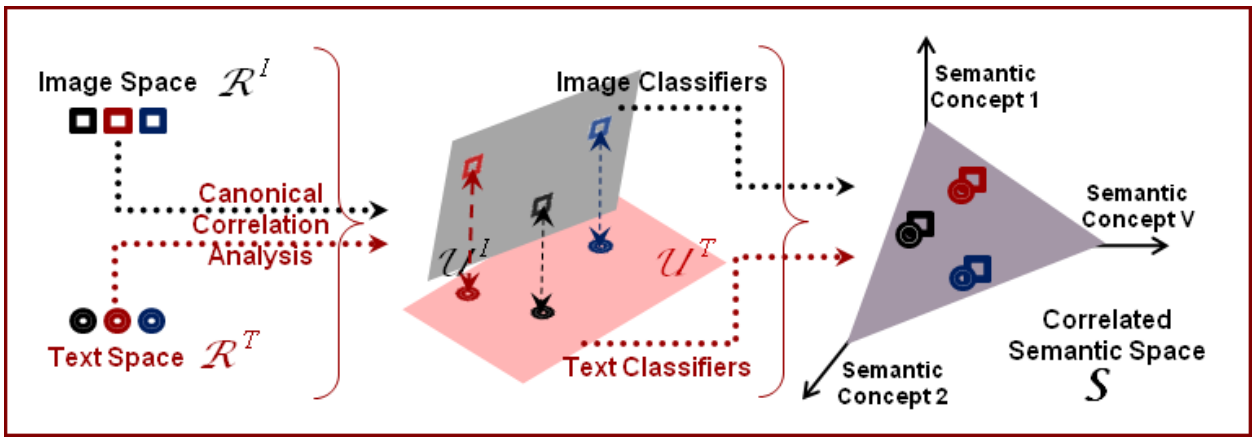


Figure 2.1: Canonical Correlation Analysis

Several attempts have been made in considering and thus relating both textual and visual modalities to perform image retrieval. These approaches separately train models into different spaces and then combine them using pairwise correlation. Such approaches involve PCA[2] and CCA[3] which are essentially dimensionality reduction and mapping algorithms. In Figure 1. showing Canonical Correlation Analysis, it can be seen that individual text and image classifiers thus create separate spaces

for both modalities and then they are correlated using specific correlation techniques such as canonical correlations which map a sample from one modality to another based on a fixed function. Although these approaches seem useful, they are attempting to project one modality onto another which brings in a lot of limitations such as procedures that maximize correlation between canonical variate pairs do not necessarily lead to solutions that make logical sense. Also, pairings of such representations are independent of each other and only linear relations are possible. Thus, the semantic aspect of the mapping is not achieved as there may be more than just linear relations between the images and the related texts involved.

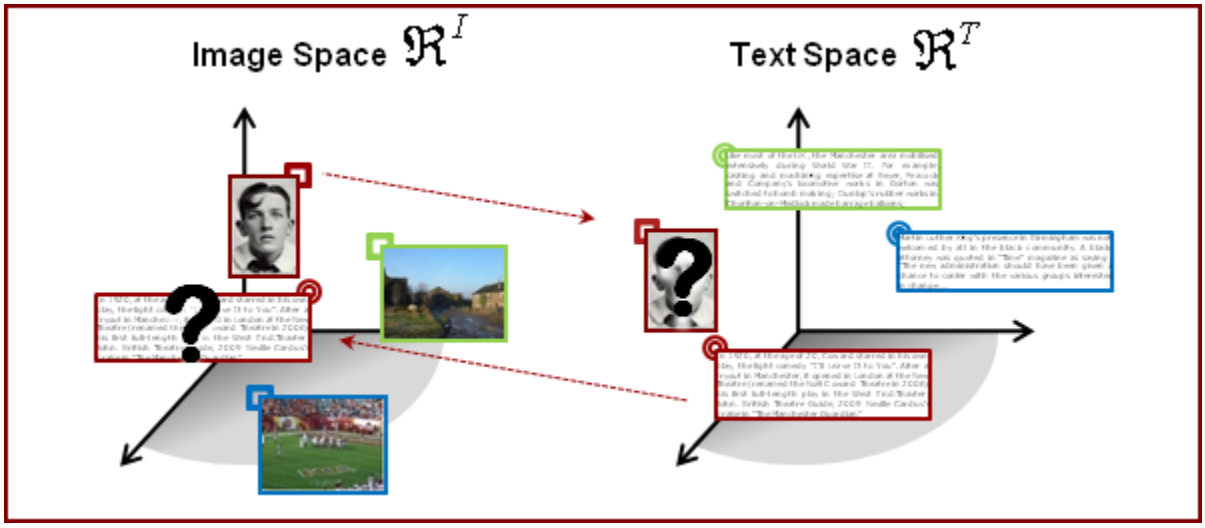


Figure 2.2: Image and Text Spaces

In order to overcome this, creation of a common embedding space which consists of not only just text or not just text but contains both modalities simultaneously must be created. This common embedding space would be able to capture the semantics of both modalities thus making the space semantically more appropriate.

3 Methodology

It is very well known that language models are extremely good at capturing semantic relations in the textual modality. Recent developments in transformer[4] architectures such as BERT[5] which are able to capture bidirectional information in a text could be leveraged to make a robust language model to deal with it's semantic aspects. We consider the Flickr 8k[1] dataset which contains 8091 images and corresponding each image we have 5 textual captions which best describe the image. We use this dataset to create our retrieval system. We take the captions and pass them through the sentence encoders of RoBERTa[6] which is a successor of BERT[5] and calculate mean semantic embedding corresponding to each image. Sentence encoders are a modification of the pretrained RoBERTa[6] network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the effort for finding the most similar pair from 65 hours with RoBERTa[6] to about 5 seconds with SBERT[7], while maintaining the accuracy from the original model.

Now, we have an image and 1024 dimensional embeddings textual representation of the image. Thus, these images can now be placed into the semantically rich textual space space by predicting the corresponding embeddings using a Convolutional Neural Network. This common embedding space would thus be able to represent both textual and image modality into the same space along with being very semantically rich. The images can now be retrieved not only by using textual descriptions but also images as a query image may be passed into the CNN and the predicted embedding can be used to find the most similar images based on it's visual semantic features.

3.1 Data Preprocessing

The Flickr 8k dataset[1] considered for the task consists of 8091 images and corresponding textual captions for the image. They are preprocessed as follows:

3.1.1 Converting captions to target embeddings:

Here, each caption is converted into a 1024-dimensional embedding by using the RoBERTa sentence encoder model[7]. RoBERTa: Robustly Optimised BERT Training Approach[6] is a transformer[4] based model which was formed by retraining BERT on various datasets and achieved State of the Art performance on various tasks. It is a transformer based language model which is able to understand the semantics of the text corpus with very high accuracy and has since then been used for various language based tasks involving fine tuning it for the relevant text corpus and using the encoder for tasks such as text classification, text summarization, topic modelling, etc.

The RoBERa sentence encoder has been developed like a siamese network where two sentences are passed through the model and cosine similarity between them is measured. The model has given State of the Art results in Semantic Sentence Similarity and Natural Language Inference tasks on many benchmark datasets. Thus, the sentence encoder model does the job of projecting sentences into a higher dimension of semantic embedding space with high efficiency, Thus, all the 5 textual captions corresponding to an image were passed through RoBERTa Natural Language Inference large model and mean 1024-dimensional embeddings were obtained for each of the images and set them as targets for the images to predict.

3.1.2 Image Augmentations

The images were squished to size the required size and passed through an image augmentation pipeline consisting of various transformations as follows:

Figure 3.1: Original Image

- Random Crop with reflection padding mode:

Figure 3.2: Random Crop

- Random Horizontal Flip with probability of 0.5:

Figure 3.3: Random Horizontal Flip

- Symmetric Warp with probability=0.75 and magnitude=(-0.2, 0.2):

Figure3. 4: Symmetric Warp

- Rotate with probability and degrees=(-10, 10):

Figure 3.5: Rotate

- Zoom with probability=0.75, scale=(1.0, 1.1):

Figure 3.6: Zoom

- Brightness with probability=0.75, change=(0.4, 0.6):

Figure 3.7: Brightness

- Contrast with probability=0.75, scale=(0.8, 1.25):

Figure 3.8: Contrast

3.2 Architecture

After creation of the augmentation pipeline and target embeddings. The task was straightforward, input was an image and the output was a 1024-dimensional embedding with continuous values. and thus was a regression task to be achieved using Convolutional Neural Network structure.

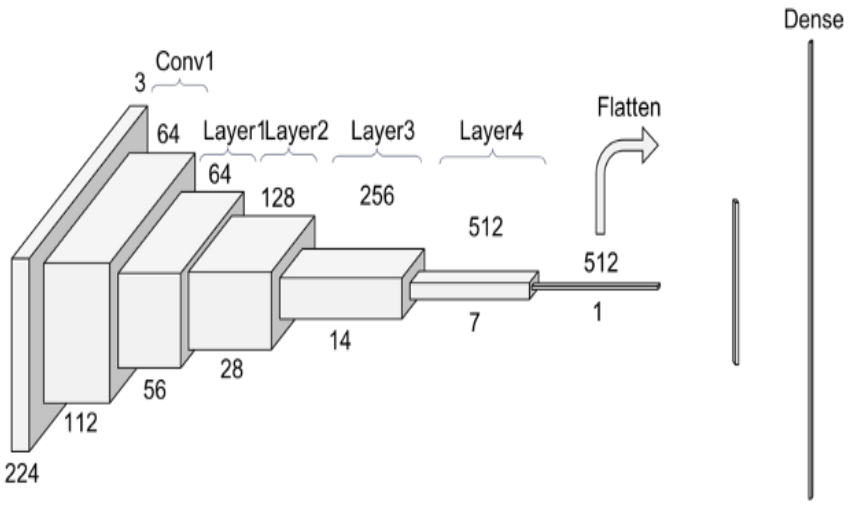


Figure 3.9: CNN Architecture

The ResNet34[9] architecture was used as a base and a few more layers were added at the end of the model and the prediction layer was a dense layer of 1024 linear units. The advantage of using Residual Networks is they allow the training of deep neural networks without moving too far from the original image which is achieved using residual blocks.

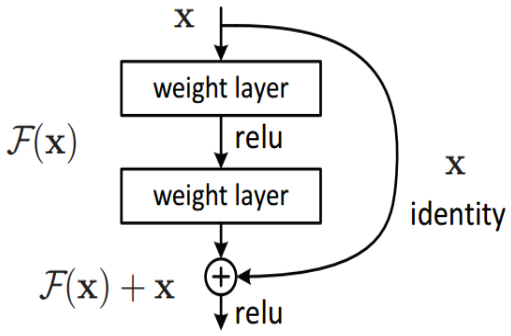


Figure 3.10: Residual Block

Residual blocks make use of skip connections which add the original input back to the output feature map obtained by passing the input through one or more convolutional layers. Let us consider a neural network block, whose input is *x* and we would like to learn the true distribution *H(x)*. Let us denote the difference (or the residual) between this as

$$R(x) = \text{Output} - \text{Input} = H(x) - x$$

Rearranging it, we get,

$$H(x) = R(x) + x$$

Our residual block is overall trying to learn the true output, *H(x)* and if you look closely at the image above, you will realize that since we have an identity connection coming due to *x*, the layers are actually trying to learn the residual, *R(x)*. So to summarize, the layers in a traditional network are learning the true output (*H(x)*)whereas the layers in a residual network are learning the residual (*R(x)*). Hence, the name: *Residual Block*.

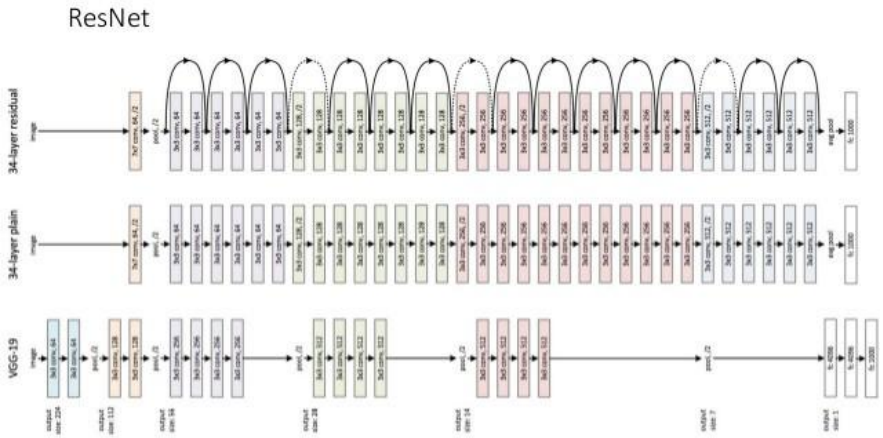


Figure 3.11: ResNet34 architecture comparison

There are multiple versions of the ResNet architectures available. We particularly make use of the ResNet 34 architecture for our purpose.

3.3 Training

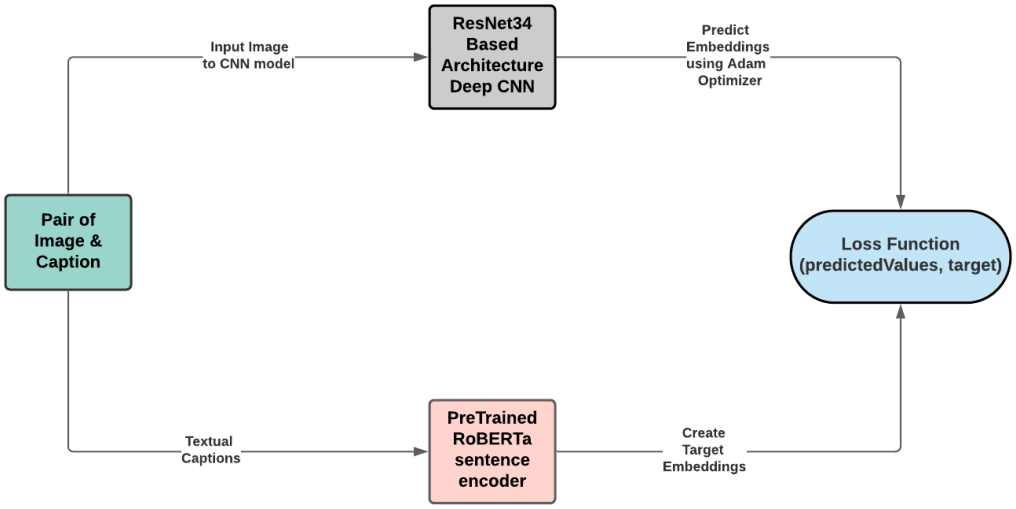


Figure 3.12: Training Process

The training was done in multiple stages and using two different loss functions namely mean squared error and cosine loss and Adam[10] Optimizer with momentum. We use the pre-trained weights of the ResNet34 architecture from the ImageNet dataset as the base architecture, remove the last classification layer with 1000 units and add in a few more layers and construct the head to predict the target embeddings.

```
(1): Sequential(
  (0): AdaptiveConcatPool2d(
    (ap): AdaptiveAvgPool2d(output_size=1)
    (mp): AdaptiveMaxPool2d(output_size=1)
  )
  (1): Flatten()
  (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): Dropout(p=0.25, inplace=False)
  (4): Linear(in_features=1024, out_features=512, bias=True)
  (5): ReLU(inplace=True)
  (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): Dropout(p=0.5, inplace=False)
  (8): Linear(in_features=512, out_features=1024, bias=True)
)
```

Figure 3.13: CNN Head

We initialize the head with random weights and freeze all the other layers except the head and fine tune it with a learning rate found by analyzing the learning rate vs. loss over the training set[12]. One Cycle Policy[11] with Adam optimizer along with varying momentums was used for training the network at various stages. One cycle policy is a training method put forward to achieve fast convergence[13] of a loss function by varying the learning rate over a cycle so as to start from a minimum value, increase it to the maximum learning rate value that we pass in and then gradually decrease it to the minimum again.

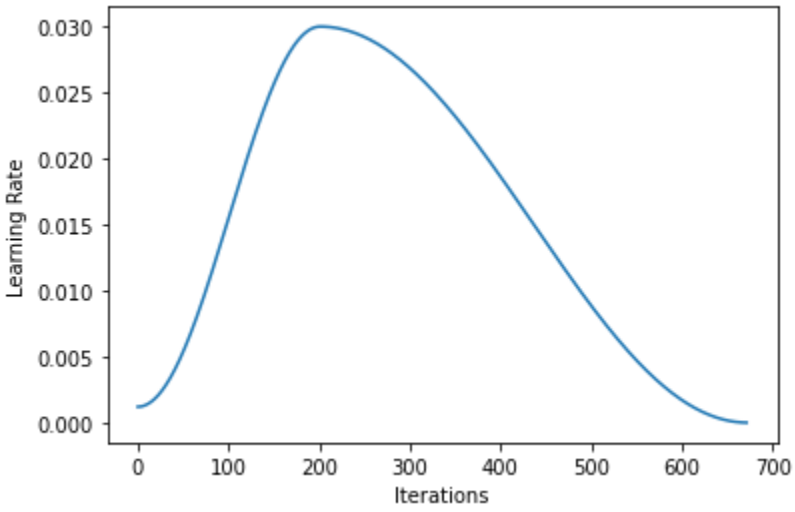


Figure 3.14: One Cycle Policy: Iterations vs. Learning Rate

3.4 Creating the Search System

After the model was trained, the predictions of all the images in the database were recorded and a search index. The system was designed as follows: for a given query embedding and value k, the system returns a list of indices of the top k nearest neighbours in a non-decreasing order of their angular distances to the query embedding. As the embedding space is too large for an accurate search to make the system efficient we make use of a non-metric space based paradigm to search for approximate

nearest neighbours[14]. The algorithm used has shown state of the art results on various retrieval tasks and is currently a part of Amazon ElasticSearch.

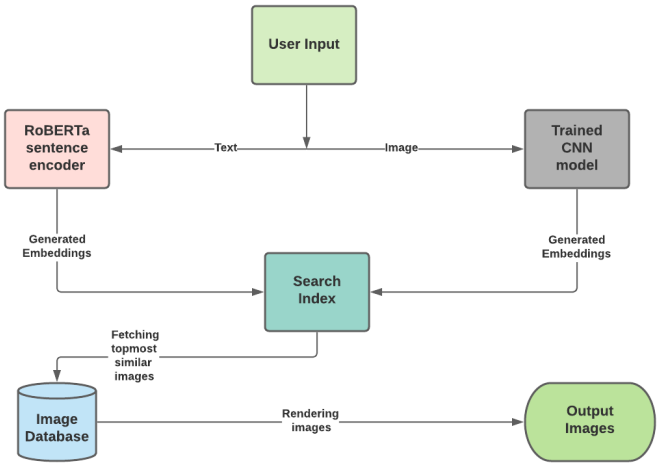


Figure 3.15: Complete Retrieval System

If a textual query is given as an input, a 1024-dimensional embedding is obtained by passing it through the RoBERTa model and the search for top k nearest images is done, and if an image is given an input, it is passed through the CNN model and the 1024-dimensional prediction searched for using the same process as in textual query and top k queries are returned. The model was also deployed onto a ReactJS based frontend system which interacted with the model at the backend using a web API.

4 Experimental Setup

Various experiments were performed during the training process using mean squared error as the loss function.

Image Size	Stage	Learning Rates	# Epochs
(224, 224, 3)	1 (Head)	3e-2	24
(224, 224, 3)	2 (Complete Network)	slice(5e-6, 3e-5)	12

(256, 256, 3)	1 (Head)	4e-4	12
(256, 256, 3)	2 (Complete Network)	slice(8e-6, 1e-4)	12

Table 4.1: Training Hyperparameters

4.1 Progressive Resizing

As the images we are dealing with have varying sizes, the images were first loaded in size of (224, 224, 3) and the network trained on these image sizes was used to fine tune the network to use larger image sizes till there wasn't a significant performance change.

4.1.1 Image Size 224

Initially, all images were squished to size (224, 224, 3) and trained using the pretrained weights from ImageNet on the ResNet34 architecture and by adding a custom head as described in figure 3.13. The network was then trained in two stages. In the first stage, all layers except the head were frozen and the randomly initialized head was fine tuned using the learning rate found by analyzing the mean squared error as a loss function vs. the learning.

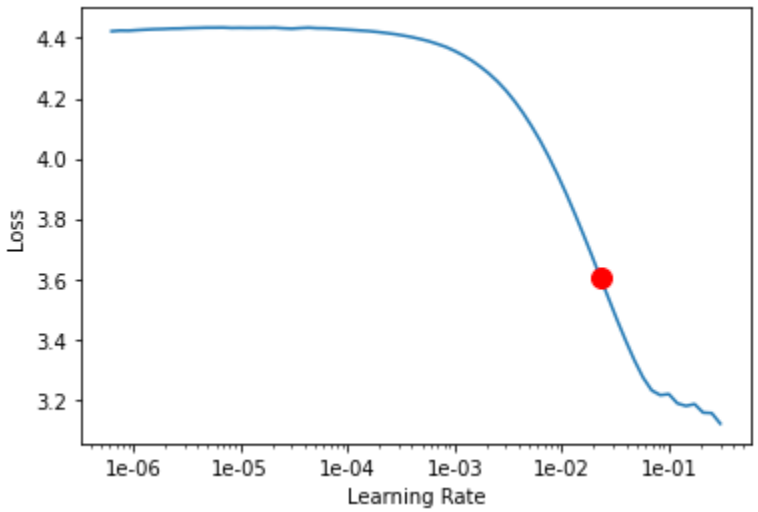


Figure 4.1: LR Finder Stage 1 : learning rate vs. loss

As we can see, 3e-2 is a good learning rate as the loss slope is steep and the loss is still decreasing at that rate. Hence, we train the head for 24 epochs using the one cycle policy.

Next, in stage 2, all the layers were unfrozen and again the learning rate finder is used to analyze the learning rate vs. loss. This time, the graph obtained was as follows:

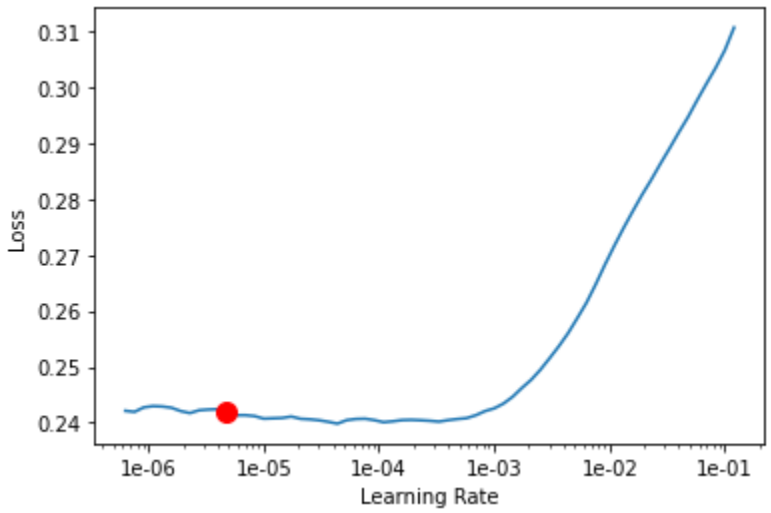


Figure 4.2: LR Finder Stage 2: learning rate vs. loss

Here, we pass in a learning rate slice of (5e-6, 3e-5). The reason behind using this is to leverage the benefits of discriminative learning rates. The idea behind using discriminative learning rates is to usually set lower learning rates at early layers and higher learning rates at later layers. So, for convolutional neural networks, we know that the layers at the beginning are good at learning general features whereas the layers towards the end of the architecture learn specific features. Using this idea, when we unfreeze all the layers and are ready to update the weights, we can set the learning rate to be different for different layers. Specifically, we want the first layer to have the lowest learning rate and final layer to have the highest. The layers in between thus have learning rates equally distributed ranging between the lowest and the highest learning rate.

4.1.2 Image Size 256

The images were now resized to (256, 256, 3) and again passed through the same augmentation pipeline. The pretrained weights from the network which has become very good in capturing information from image size 224 were used and again the same two stages were followed for training.

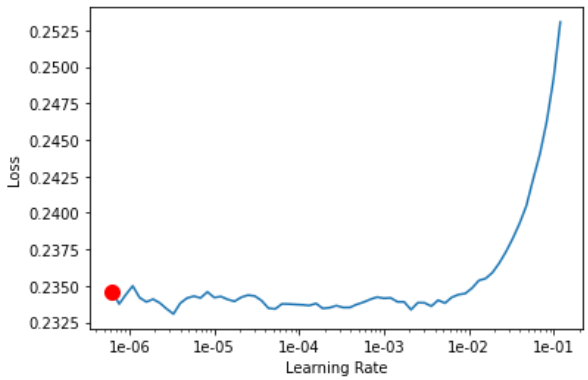
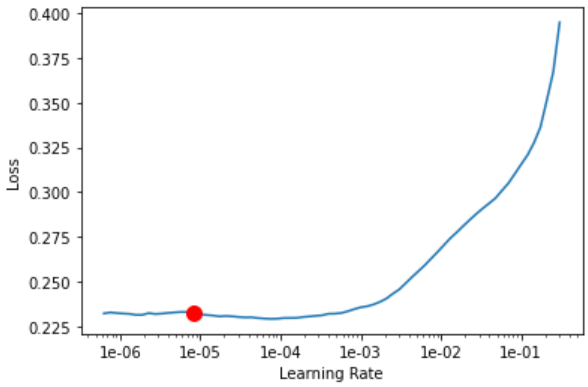
Stage 1	Stage 2
	

Table 4.2: LR finders size 256

The head was fine tuned at stage 1 with a learning rate of 4e-4 and then the whole network was trained with a learning rate slice of (8e-6, 1e-4). The image size was increased further to 299 and 512 but as the performance on the validation set didn't show any improvements. The 256 image size network was finalized.

5 Results and discussion

5.1 Training

The model was trained using mean squared error with Adam optimizer and validated using a validation set of 0.1 percent from the whole dataset. The training progress can be seen from the training plots of various stages.

- **Training Progress:**

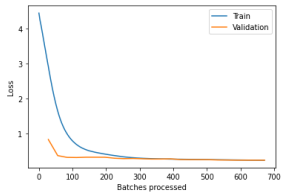
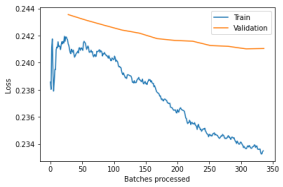
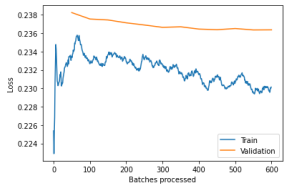
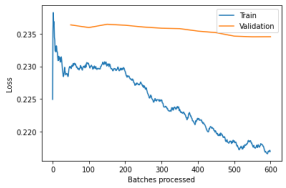
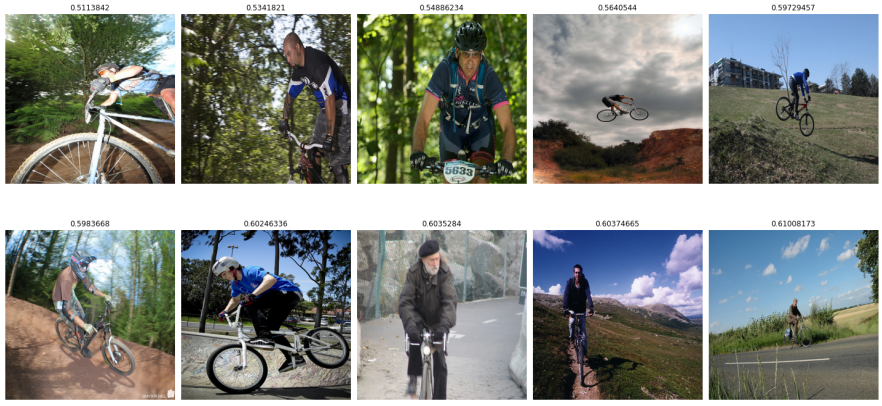
Image Size 224		Image Size 256	
Stage 1	**Stage 2**	**Stage 1**	**Stage 2**
			

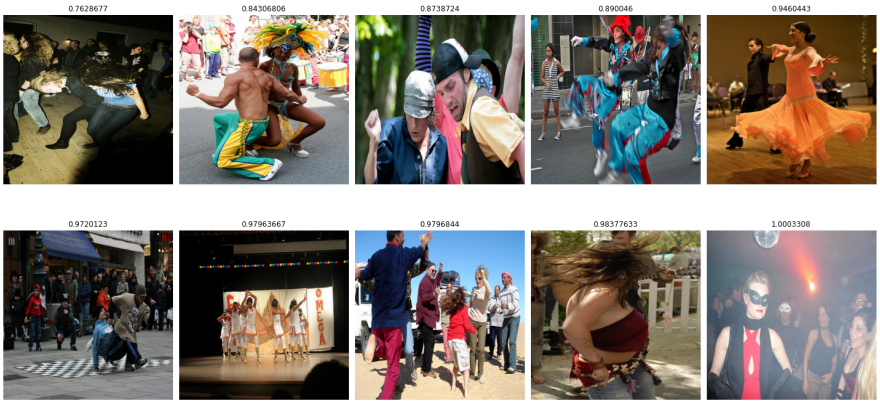
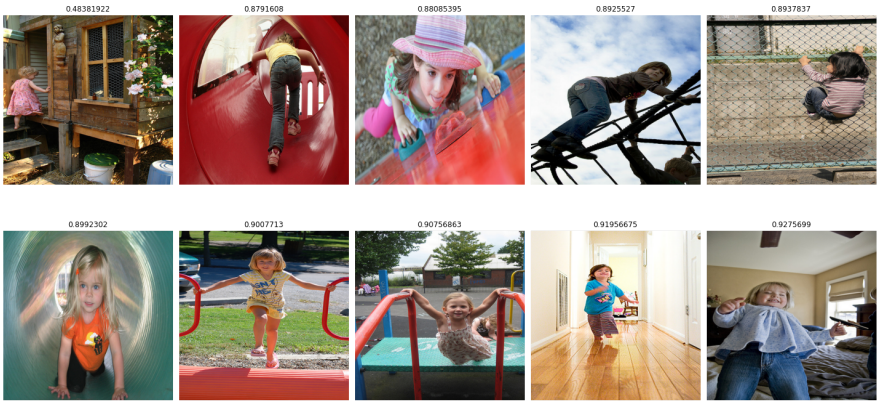
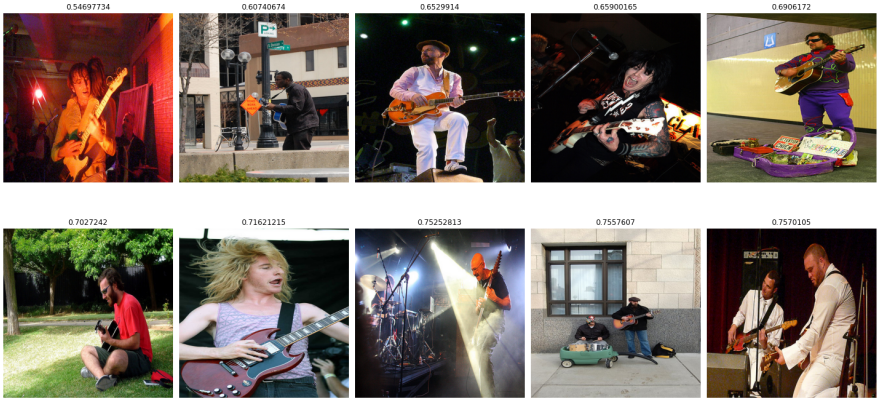
Table 5.1: Training Progress

As is evident from the plots, both the training and validation losses improved from 2.961700 and 0.834557 to 0.216961 and 0.234539 respectively which is a significant leap from where the model started. Also, the difference between the training and validation losses is not much which shows that the model did not overfit the training set.

5.2 Retrieval Results

- **Text to Image Search:**

Query	Top 10 Results
“ A man riding a bike”	

“People dancing”	
“A little girl climbing into a wooden playhouse .”	
“A man playing a guitar”	

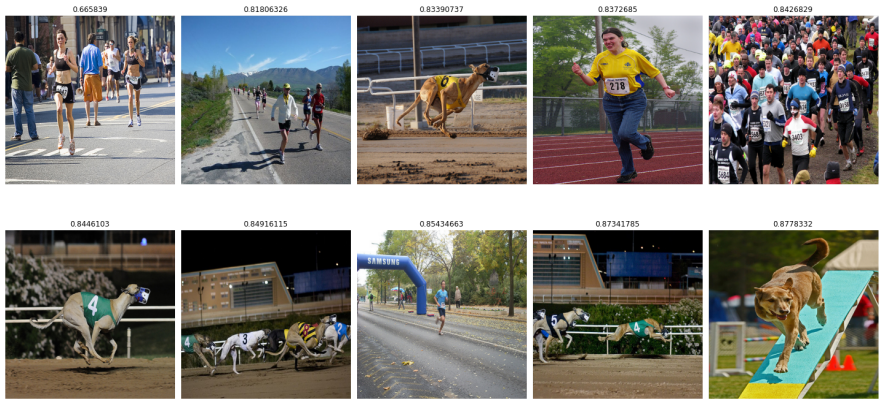
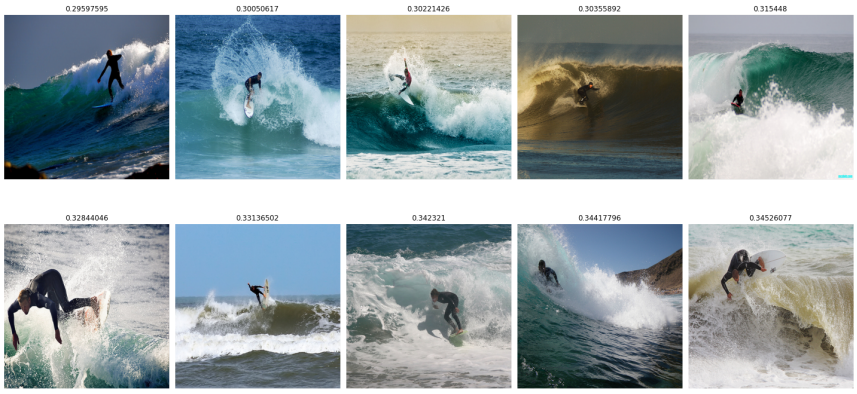
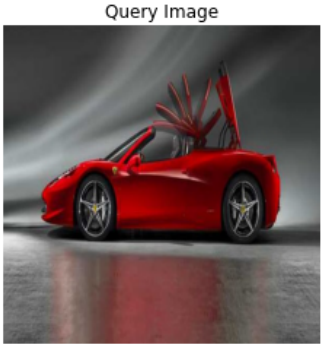
“A person running in a marathon.”	
“fireworks”	

Table 5.2: Textual Query Results

As we can see from the above table, the model is able to capture semantically rich results matching with the textual description. Not only it is able to correctly depict what the caption means but if specific query result does not match fully, it returns its closest match as seen in the example of **“A person running in a marathon”** where the most images exactly match the description but the rest images have animals in them but the thing to notice is that the model is able to capture the process of **“running”** successfully.

- **Image to Image Search:**

Query	Top 10 Results
Query Image 	
Query Image 	
Query Image 	

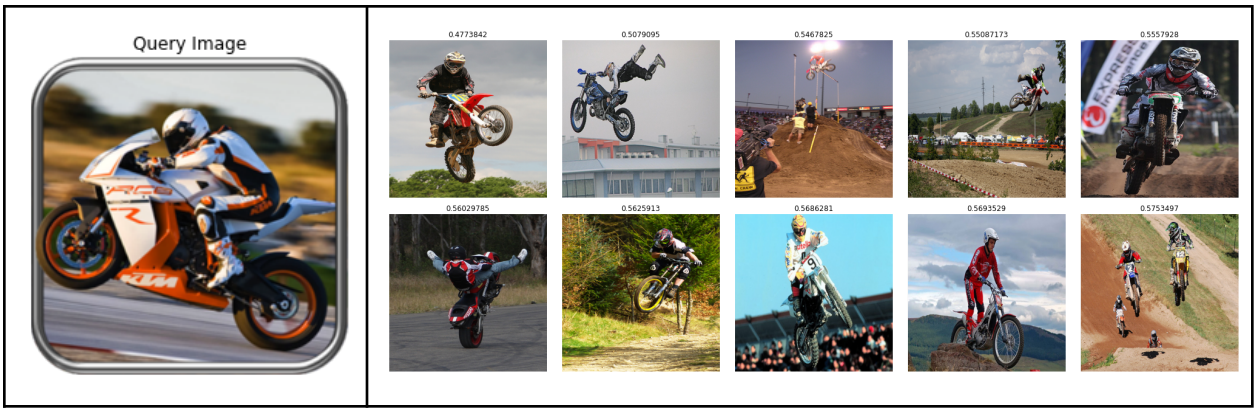


Table 5.3: Image Query Results

Even for image to image search, the model is able to capture the semantics in the query image and thus provide extremely relevant results. In the example of having players **playing cricket,** the system is not able to capture players playing cricket but is able to understand the environment which is a **playground** and related results where **people are playing some kind of sport if not exactly cricket on a playground**.

5.3 Accuracies

- **Top k accuracies**

  The top 1, 5, and 10 accuracies were calculated for each of the 5 captions as queries and if the image corresponding to the query was present in the top k results.

Caption #	Top 1	Top 5	Top 10
1	72.77	88.38	92.56
2	75.63	91.09	94.24
3	76.33	90.55	93.73
4	74.77	89.79	93.30
5	72.81	88.80	92.07
Mean	**74.46**	**89.72**	**93.18**

Table 5.4: Top k accuracies for exact image match

- **Top k accuracies based on cosine similarity**

The top k results which have cosine similarity with the query text greater than the threshold are considered relevant to calculate this metric.

Similarity Threshold	Top 1	Top 5	Top 10
0.7	**93.31**	**61.72**	**48.81**
0.5	**99.86**	**98.36**	**96.26**

Table 5.5: Averaged top k accuracies for semantically relevant images

6 Conclusions and recommendations

As we saw, the model is correctly giving relevant results for single keyword based textual queries such as "**fireworks**" . Also, caption based queries such as "**People dancing**" and "**A person playing a guitar.**", etc. This shows that the model understands the semantics of the language and is correctly able to fetch the relevant images. The image to image search also shows excellent results in the sense that for the surfing image query, the model is actually able to capture the visual semantics like involvement of water and man actually on a surfboard in the process of **surfing**. Also, we can see that for the query of **red-ferrari**, the top images are actually **red** colored cars. This shows that the model is able to capture the color and environment of the object involved too.

This search system shows some extraordinary results and thus, we have shown that the approach of multimodal common embedding space for image retrieval tasks works successfully. A few more approaches may be tried and experimented such as using a pre-trained object detection model such as YOLO or segmentation model in place of simple CNN. But these models would take huge processing time as object detection models tend to look for bounding boxes using approaches such as sliding windows which are time consuming. Self supervised learning based approaches such as contrastive loss based learning may also be explored.

7 References

1. Hodosh, Micah, Peter Young, and Julia Hockenmaier. "Framing image description as a ranking task: Data, models and evaluation metrics." Journal of Artificial Intelligence Research 47 (2013): 853-899.
2. Jolliffe IT (2002) Principal component analysis, 2nd edn. Springer, New York
3. Jia Chen, Gang Wang, Yanning Shen, & Georgios B. Giannakis. (2018). Canonical Correlation Analysis of Datasets with a Common Source Graph.
4. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, & Illia Polosukhin. (2017). Attention Is All You Need.
5. Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
6. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, & Veselin Stoyanov. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.
7. Nils Reimers, & Iryna Gurevych. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.
8. Nils Reimers, & Iryna Gurevych. (2020). Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation.
9. Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2015). Deep Residual Learning for Image Recognition.
10. Diederik P. Kingma, & Jimmy Ba. (2017). Adam: A Method for Stochastic Optimization.
11. Leslie N. Smith. (2017). Cyclical Learning Rates for Training Neural Networks.
12. Leslie N. Smith. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay.
13. Leslie N. Smith, & Nicholay Topin. (2018). Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates.
14. Yu. A. Malkov, & D. A. Yashunin. (2018). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs.
15. Leonid Boytsov and David Novak and Yury Malkov and Eric Nyberg (2016). Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016* (pp. 1099–1108). ACM.