# Image Captioning Using Deep Learning

**Arnav Arnav**
Indiana University Bloomington
School of Informatics and Computing
aarnav@iu.edu

**Hankyu Jang**
Indiana University Bloomington
School of Informatics and Computing
hankjang@iu.edu

**Pulkit Maloo**
Indiana University Bloomington
School of Informatics and Computing
maloop@iu.edu

## Abstract

Scene understanding has always been an essential task in computer vision, and image description or image captioning is one of the major areas of AI research since it aims to mimic the human ability to compress an enormous amount of visual information in a few sentences. The task aims to provide short but detailed descriptions of the image in a few sentences and requires the use of techniques from computer vision and natural language processing. Recent developments in deep learning and the availability of image caption datasets such as Flikr and COCO have enabled significant research in the area. Our experiment on training the model using Flickr8k data using Convolutional Neural Network (CNN) as an encoder and Recurrent Neural Network (RNN) as decoder shows that our model is capable of generating some reasonable captions.

**Keywords :** Image Captioning, Convolutional Neural Network, Recurrent Neural Network

## Introduction

Humans can describe a scene in an image with no difficulty, but this task has been difficult for computers (Karpathy and Fei-Fei 2015). Scene Description can be instrumental, such as helping visually impaired people better understand images on the internet (Vinyals et al. 2015).

Scene Understanding has been an active area of research in computer vision since the task is broad and requires models that can perform various tasks as one. Mainly, scene understanding requires detecting and recognizing known objects, localizing the objects and learning the spatial relationships between the objects. The model should also be robust to changes in illumination changes, and be able to handle occlusions (Aarthi and Chitrakala 2017).

The task of image captioning or scene description is more challenging since generating meaningful descriptions requires identifying essential objects and learning language dependencies and correctly using the recognized objects in a sentence based on the language.

Through this project, we aim to understand the approaches used in image captioning. We choose one approach and experiment with it and show the results that were obtained.

## Background

### Convolutional Neural Network

A convolutional neural network is an architecture of a neural network that has been used for image classification and object detection with great success. The most common architecture consists of three main operations that are repeated several times. Firstly, a convolution is applied to the image, and the weights of these convolution kernels or filters are not predetermined but are learned from the data. Next, the result of these convolutions is compressed into smaller matrices with the help of pooling. Pooling is the process of aggregating results from regions in a manner similar to convolution. Pooling can be done in various ways such as max-pooling that retains the maximum element in the pooling region, min pooling or average pooling.

The size of the convolution filter, the pooling region, the number of filters to use and the number of convolution and dense layers are hyperparameters and need to be set by trying what works best for a problem. The number of dropout layers, that randomly allow a node to propagate its result further, the probability of dropout and their position is also important. These are the parameters that need to be set properly for best results. (contributors 2018a)

There are various architectures of Convolutional Neural networks that have been used for different tasks, and there is active research in the area. Three models were used for the purpose of this project - VGG16, VGG19, and ResNet50.

The VGG architecture was first introduced in (Simonyan and Zisserman 2014) for image classification on the ImageNet data set. The architecture is known for its simplicity as it uses only a $3 \times 3$ convolutional layers stacked on top of each other. The VGG19 model has more of these layers stacked compared to the VGG16. The volume is reduced in each step with the help of max pooling. These convolutional layers are followed by two fully connected layers that flatten the output and a softmax layer that predicts the probability of each of the objects the model was trained for. (Rosebrock 2017).

The Residual Networks (ResNet) introduced first in (He et al. 2016) consists of stacked Residual blocks. The residual block comprises a skip connection which makes it easier to learn the identity function and hence stacking these residual blocks helps us go more in-depth and avoid the diminishing gradient problem (Rosebrock 2017).

### Recurrent Neural Network

One of the significant limitations of using traditional feed forward neural networks is that they are trained with a set of input and output vectors. The order of these vectors does not affect the predictions made after training, and they produce fixed size outputs. There is no way of learning sequences and learning context or dependencies in various vectors in the sequence when using traditional neural networks. This is where Recurrent Neural Networks come to help (Karpathy 2015).

The input to the hidden layer in an RNN (with a single hidden layer) is the input vector, along with the output of the hidden layer for the previous time step. The RNNs are trained to learn to predict the next word given the current example. In other words, the output from the first training example is the following training example. This is done for multiple times in each iteration, which represents the length of the sequence that the RNN can learn and predict later. The Network is trained using back propagation through time, which adjusts the weights between the hidden layer for a given time step and the next time step. Once trained for various iterations, the RNN can learn to model the sequence (contributors 2018c).

There are problems with RNNs, when learning long sequences, in situations such as the language translation of large documents, where it may be necessary to remember only the context over a small time period. For this purpose, Long Short Term Memory (LSTM) networks are used, where each cell has three gates - input, forget and output - and can learn when to forget the previous context, along with other parameters. The LSTM is trained such that each LSTM cell updates its weights at each time step and the all the weights are updated after each iteration. This helps the network learn long sequences and decide which parts of the sequences are related with some context (Trask 2015) (contributors 2018b). Gated Recurrent Units (GRUs), another type of Recurrent Neural Network, have also become quite popular to solve the same problem. In a GRU unit, there is only an update gate and a reset gate; however, there exist variations of GRU such as a minimal gated unit in which there exists only one gate (forget gate). GRUs train faster compared to LSTMs while giving comparable results.

RNNs, LSTMs, and GRUs have been used to learn long sequences of text and music and to generate new documents given some starting words or phrases (Karpathy 2015).

## Related Work

One of the influential papers by Andrej Karpathy et al. in image captioning divides the task into two steps: mapping sentence snippets to visual regions in the image and then using these correspondences to generate new descriptions (Karpathy and Fei-Fei 2015).

The authors use a Region Convolutional Neural Network (RCNN) to represent images as a set of $h$ dimensional vectors each representing an object in the image, detected based on 200 ImageNet classes. The authors represent sentences with the help of a Bidirectional Recurrent Neural Network (BRNN) in the same $h$ dimensional space. Each sentence is a set of $h$ dimensional vectors, representing snippets or words. The use of the BRNN enriches this representation as it learns knowledge about the context of each word in a sentence. The authors find that with such a representation, the final representation of words aligns strongly with the representation of visual regions related to the same concept. They define an alignment score on this representation of words and visual regions and align various words to the same region generating text snippets, with the help of a Markov Random Field. With the help of these correspondences between image regions and text snippets, the authors train another model that generates text descriptions for new unseen images (Karpathy and Fei-Fei 2015).

The authors train an RNN that takes text snippets and visual regions as inputs and tries to predict the next word in the text based on the words it has seen so far. The image region information is passed to the network as the initial hidden state at the initial time step, and the network learns to predict the log probability of the next most likely word using a softmax classifier. The authors use unique START and END tokens that represent the beginning and end of the sentence, which allows the network to make variable length predictions. The RNN has 512 nodes in the hidden layer (Karpathy and Fei-Fei 2015).

The network for learning correspondences between visual regions and text words was trained using stochastic gradient descent in batches of 100 image-sentence pairs. The authors used dropouts on every layer except the recurrent layers and clipped the element-wise gradients at 5 to prevent gradient explosion. The RNN to generate descriptions for unseen images was trained using RMSprop which dynamically adjusts the learning rate (Karpathy and Fei-Fei 2015).

Kelvin Xu et.al (Xu et al. 2015) use the concept of attention to better describe images. The authors propose models that focus on which area of the image, and what objects in the image are being given attention and evaluate these models on different image captioning datasets. The idea behind the approach is that much like the human visual system, some parts of the image may be ignored for the task of image description, and only the salient foreground features are considered. The authors use a CNN to learn important features of the image and an LSTM (Long short-term memory network) to generate description text based on a context vector.

Jyoti Aneja et al. in (Aneja, Deshpande, and Schwing 2017) use a convolutional approach to generate description text instead of a simple RNN, and show that their model works at par with RNN and LSTM based approaches.

Andrew Shin et al. (Shin, Ushiku, and Harada 2016) use a second neural network, finely tuned on text-based sentiment analysis to generate image descriptions which capture the sentiments in the image. The authors use multi-label learning to learn sentiments associated with each of the images, then use these sentiments, along with the input from the CNN itself as inputs to an LSTM to generate sentences which include the sentiment. The LSTM is restricted so that each description contains at least one term from the senti-

Figure 1: Flickr8k - Sample image and captions

ment vocabulary.

Alexander Mathews et al. (Mathews, Xie, and He 2016) emphasize how only a few image descriptions in most datasets contain words describing sentiments, and most descriptions are factual. The authors propose a model that consists of two CNN + RNN models each with a specific task. While one model learns to describe factual content in the image, the other learns to describe the sentiment associated, thus providing a framework that learns to generate sentiment based descriptions even with lesser image sentiment data.

Quanzeng You et.al in (You, Jin, and Luo 2018) propose approaches to inject sentiment into the descriptions generated by image captioning methods.

Tsung Yi Lin et.al in (Lin et al. 2014) describe the Microsoft Common Objects in Context dataset, that is widely used for benchmarking image captioning models.

## Approach

### Dataset

The FLickr8k data set is a collection of 8000 images with five captions each, collected in one place, and available to be used for the benchmarking of image captioning and image querying approaches (Rashtchian et al. 2010). The authors show that better results can be achieved when multiple captions are used with each image, to train the model.

Figure 1 is a sample image file in Flickr8K dataset. The image is paired with following five human-generated training captions:

- A black dog running in the surf .
- A black lab with tags frolicks in the water .
- A dog splashes in the water
- The black dog runs through the water .
- This is a black dog splashing in the water .

### Data Preprocessing

We divide the training data (8000) and the captions into three different data sets - the training set (6000), the validation set (1000) and the test set (1000). For each of the captions in the three data sets, we create a set of training input and target captions by shifting the training input caption by one word to get the training target caption.

**Image Preprocessing** To generate image features we use pretrained weights of CNNs trained on ImageNet image classification dataset (VGG16, VGG19, and ResNet50) and remove the final dense layers from the model. We preprocess images and generate image features using the by performing a forward pass on the image on using these weights and save these features to a file.

**Caption Preprocessing** To preprocess the image captions in the training data, we first identify all the words that are there in the data set. We then generate a histogram of the distribution of these words and drop the words that occur less than five times. We end up with the vocabulary of size 2531 words, including $< bos >, < eos >, < unk >$ and $< pad >$ tags which specify the beginning of a sentence, end of a sentence, unknown word and padding respectively. We generate one hot encoding captions for each of these words and use the index of the class (index of 1) for training the network.

Our model takes an image and then generates a caption as shown in figures 2 and 3.

### Model

The model that was used for the project consists of two different input streams, one for the image features, and the other for the preprocessed input captions. The image features are passed through a fully connected (dense) layer to get a representation in a different dimension. The input captions are passed through an embedding layer. These two input streams are then merges and passed as inputs to an LSTM layer. The image is passed as the initial state to the LSTM while the caption embeddings are passed as the input to the LSTM. The architecture is shown in figure 2

### Training

The model was trained first on Indiana University's Big Red II. We faced memory problems using different batch sizes and hence moved to an Amazon Web Services (AWS) instance. The *p2.xlarge* instance was used, which includes one NVIDIA Tesla K80 GPU, with 4 Virtual CPUs and 61 GB of RAM. Training the model takes about one hour on the AWS instance with a batch size of 32 (Amazon Web Services 2018).

To train the model, for each image and each of the input captions that were generated during preprocessing, we pass the image features through the dense layer, and the preprocessed input captions to the embedding layer. We then use the image as the initial state to the LSTM, along with the caption which is passed as the input to the LSTM. The model outputs a predicted caption and the error with respect to the actual caption is back propagated using RMSprop optimization, as shown in figure 3.

**Word Embedding** Word Embeddings provide a vector representation of words that can capture something about the context of the word. There are many pretrained word embedding available; however, our model learns the word embedding as part of the model itself.

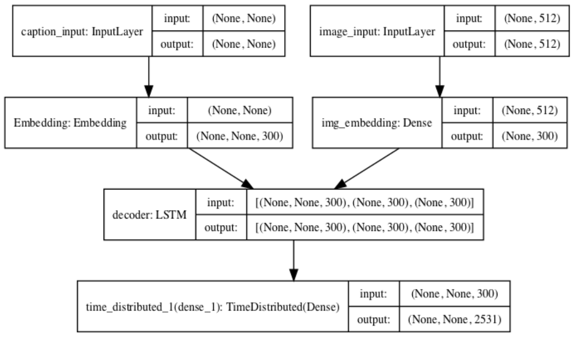

caption_input: InputLayer
input: (None, None)
output: (None, None)
image_input: InputLayer
input: (None, 512)
output: (None, 512)
Embedding: Embedding
input: (None, None)
output: (None, None, 300)
img_embedding: Dense
input: (None, 512)
output: (None, 300)
decoder: LSTM
input: [(None, None, 300), (None, 300), (None, 300)]
output: [(None, None, 300), (None, 300), (None, 300)]
time_distributed_1(dense_1): TimeDistributed(Dense)
input: (None, None, 300)
output: (None, None, 2531)

Figure 2: Our Model Architecture

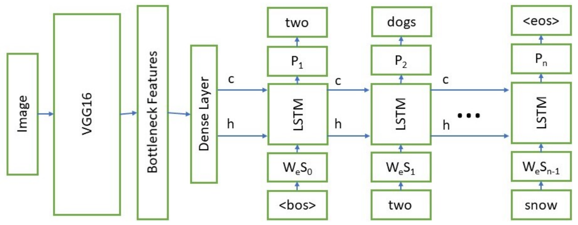


Figure 3: Training the model using VGG16 image features

**Loss Function** We used categorical cross-entropy loss function in the model. Cross-entropy in information theory defines the minimum number of bits required to identify an event drawn from a set two event distributions when the coding scheme used is generated from an estimated probability distribution instead of the true distribution (Wikipedia contributors 2018b). We want to minimize the loss to minimize the difference between the distribution of the predicted sentences and the actual captions of the image given in the training data.

**Evaluation Metric** The Bilingual Evaluation Understudy score (BLEU score) was used as the metric to evaluate the generated captions generated by the model. The BLEU score is a metric for evaluation of machine-generated translations, but can also be used to evaluate machine generated sentences in various natural language processing tasks. It is a common metric for evaluating image captions (Jason Brownlee 2017).

The BLEU score is designed to give a score between 0 to 1 (often scaled to a range of 0-100) on a corpus level and often does not produce good results on individual sentences. The BLUE score can be calculated for various n-grams and represents for each sentence the relative number of matching n-grams in the reference sentences. The scores of all the sentences are combined using a geometric mean with a penalty applied to short sentences to prevent very short sentences that are not suitable translations, from having high scores (Wikipedia contributors 2018a).

For the purpose of this project we calculate BLEU scores for unigrams u to 4-grams (BLEU1 to BLEU4 respectively) to compare results of different models that were used. We use the images in the test set, and generate captions using our model. We use a list of 5 target captions as the reference sentences and compute the blue score for all the images in the data.

| Metric | VGG16 | VGG19 | ResNet50 |
|---|---|---|---|
| BLEU1 | 51.26 | 52.64 | 51.60 |
| BLEU2 | 21.41 | 21.95 | 22.71 |
| BLEU3 | 8.32 | 8.24 | 8.99 |
| BLEU4 | 3.31 | 3.26 | 3.94 |

Table 1: Greedy (embedding size: 300, LSTM size: 300, learning rate: 0.0001, dropout: 0.2, batch size: 32, epochs: 10)

| Metric | VGG16 | VGG19 | ResNet50 |
|---|---|---|---|
| BLEU1 | 54.52 | 55.02 | 56.78 |
| BLEU2 | 24.04 | 24.21 | 25.87 |
| BLEU3 | 10.05 | 10.08 | 10.93 |
| BLEU4 | 3.97 | 3.91 | 4.46 |

Table 2: Beam Search (embedding size: 300, LSTM size: 300, learning rate: 0.000051, dropout: 0.2, batch size: 32, epochs: 33)

**Optimization** The RMSprop optimization was used to minimize the loss and train the model. The problem with training deep networks for complicated tasks is that the gradient of these arbitrarily complicated functions can either explode or vanish as the errors are back propagated. RMSprop optimization uses a moving average of the squared gradients to normalize the gradient. This in effect adaptively changes the step size depending on the gradient value medium-RMSprop. RMSprop was developed for batch training of neural networks and it has been observed that RMSprop works well for LSTM networks.

## Inference

To perform inference, we first obtain image embedding by passing the image through the CNN model and then the dense layer. Then to generate captions using the model, we first feed the LSTM cell with $< bos >$ as the first input and image embedding as its initial states. The LSTM produces a word and its hidden states, and we keep feeding this word and hidden states again to the LSTM cell until it outputs $< eos >$ or reaches the max sentence length as shown in figure 4. If the LSTM produces $< unk >$ or $< pad >$

| Metric | VGG16 | VGG19 | ResNet50 |
|---|---|---|---|
| BLEU1 | 55.20 | 55.60 | 57.49 |
| BLEU2 | 24.77 | 24.73 | 26.60 |
| BLEU3 | 10.75 | 10.56 | 11.82 |
| BLEU4 | 4.33 | 4.50 | 5.01 |

Table 3: Beam Search (embedding size: 512, LSTM size: 512, learning rate: 0.000051, dropout: 0.2, batch size: 32, epochs: 33)

token, we discard that sentence.

We want to maximize the joint probability of the sentence produced given an image. We took the log of the probabilities to avoid underflow problem.

$$log\ p(S|I) = \sum_{t=1}^{N} log\ p(S_t|I, S_0, ..., S_{t-1})$$

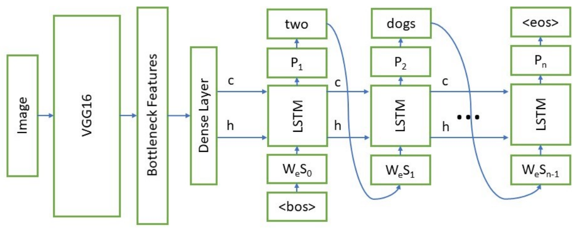


Figure 4: Inference: Generating captions once the LSTM is trained

**Beam Search** Instead of sampling in a greedy approach described above, the better way is to do Beam Search where we keep $k$ best sentences produced so far up to time $t$ to generate sentences of size $t + 1$. In this way, we get $k$ best sentences at the end. Beam Search significantly improved our BLEU scores as shown in table 2. We used various beam sizes for our experiments. Using Beam Search favors sentences with shorter lengths, and so we implemented length normalization to get the sentences with maximum average log probability. We also raised the length normalizing term to power alpha (0.7) which is a generally used hack that produced better sentences.

## Experiments

We trained the model 3 times for each of the CNNs models that we used. First, we trained the model using the learning rate as 0.0001 for 10 epochs and used greedy approach to generate captions. The BLEU scores for this set of hyperparameters are shown in table 1.

Next we decreased the learning rate to 0.000051, and trained the model to 33 epochs and used beam search to generate sentences. The BLEU scores for this set or hyperparameters are given in table 2. We can see that there is a significant improvement in the BLEU scores

Finally we increased the size of the embedding layer and the dense layer from 300 to 512, increased the LSTM size from 300 to 512 and trained the model again for 33 epochs. The BLEU scores for these hyperparameters are given in table 3. We can see that increasing the LSTM size and the size of the embedding layer lead to even better results, even though it took significantly longer time to train the models with these hyperparameters.

We cleaned up our code and created two files that use argument parser to specify parameters for training and the model, and generating captions using trained weights. We also created a simple web application in python with the help of flask that allows users to upload an image and uses the the trained weights to generate image captions. The application is included in the repository along with the code.

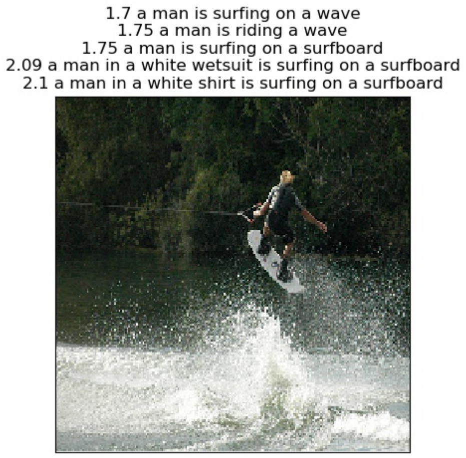
The link to the github repository with all the code can be reached by clicking HERE

## Results

The table 3 shows the best results and we see that the BLEU score is highest with the model that uses ResNet to generate image features. The results shown here are for the ResNet50 model using a LSTM size of 512 and a embedding layer and a dense layer size of 512.

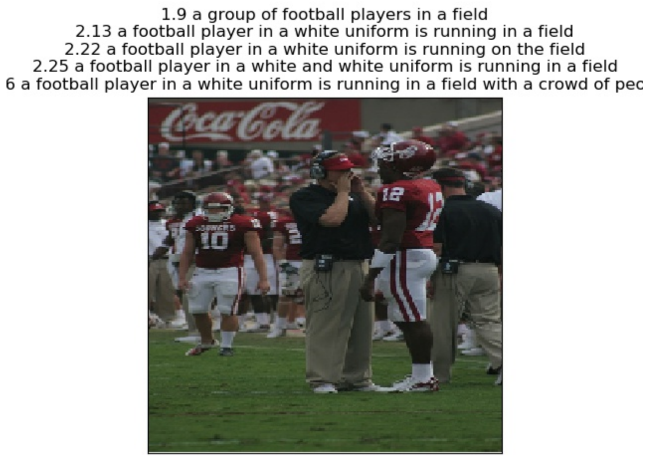
The images in the results show 5 sentences, generated using a beam size of 5 along with the of the average log probability of the sequence of words. The first sentence corresponds to the best caption according to the beam search and the last sentence corresponds to the worst.

**Caption that make sense** We see from the images in figure 5 and figure 6 that the image captions make sense and describe the scene pretty well. More such images are shown in the appendix.



1.7 a man is surfing on a wave
1.75 a man is riding a wave
1.75 a man is surfing on a surfboard
2.09 a man in a white wetsuit is surfing on a surfboard
2.1 a man in a white shirt is surfing on a surfboard

Figure 5: Surfer: Results using ResNet50, LSTM size 512



1.9 a group of football players in a field
2.13 a football player in a white uniform is running in a field
2.22 a football player in a white uniform is running on the field
2.25 a football player in a white and white uniform is running in a field
6 a football player in a white uniform is running in a field with a crowd of pec

Figure 6: Football: Results using ResNet50, LSTM size 512

**Caption with more errors** We can see from the images in figure 7 and figure 8 that the captions describe the scene well but better descriptions can be expected and the model gets confused sometimes. More such images can be found in the appendix.

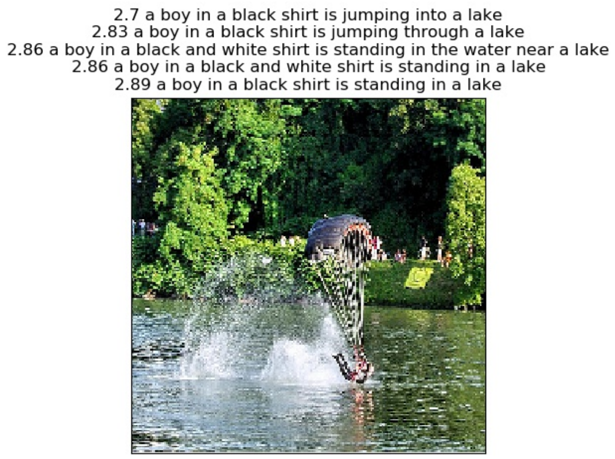

2.7 a boy in a black shirt is jumping into a lake
2.83 a boy in a black shirt is jumping through a lake
2.86 a boy in a black and white shirt is standing in the water near a lake
2.86 a boy in a black and white shirt is standing in a lake
2.89 a boy in a black shirt is standing in a lake

Figure 7: lake: Results using ResNet50, LSTM size 512

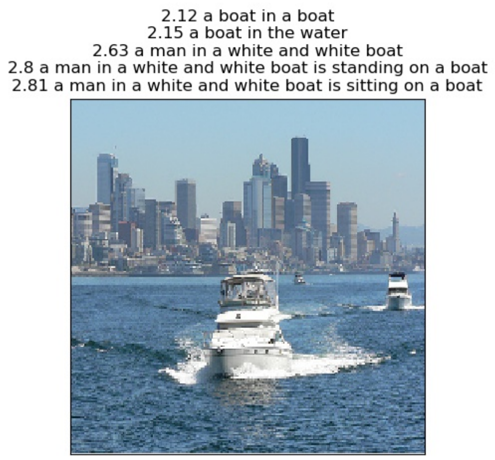

2.12 a boat in a boat
2.15 a boat in the water
2.63 a man in a white and white boat
2.8 a man in a white and white boat is standing on a boat
2.81 a man in a white and white boat is sitting on a boat

Figure 8: boat: Results using ResNet50, LSTM size 512

**Captions that make no sense** We can see from the images in figure 9 and figure 10 that the captions are grammatically correct sentences but fail to explain what is happening in the image. More such examples can be found in the appendix.

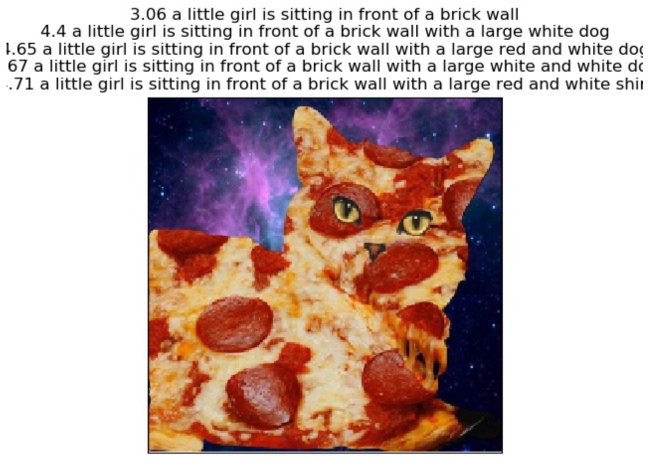

3.06 a little girl is sitting in front of a brick wall
4.4 a little girl is sitting in front of a brick wall with a large white dog
.65 a little girl is sitting in front of a brick wall with a large red and white dog
67 a little girl is sitting in front of a brick wall with a large white and white dc
.71 a little girl is sitting in front of a brick wall with a large red and white shi

Figure 9: Cat pizza: Results using ResNet50, LSTM size 512

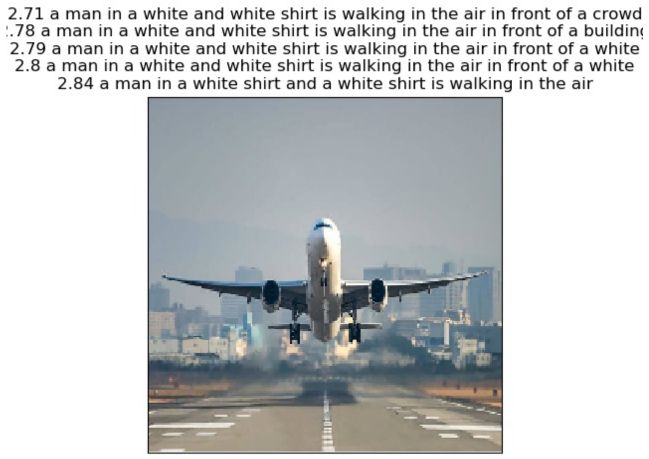

2.71 a man in a white and white shirt is walking in the air in front of a crowd
.78 a man in a white and white shirt is walking in the air in front of a building
2.79 a man in a white and white shirt is walking in the air in front of a white
2.8 a man in a white and white shirt is walking in the air in front of a white
2.84 a man in a white shirt and a white shirt is walking in the air

Figure 10: Aeroplane: Results using ResNet50, LSTM size 512

## Conclusion

Through this project, we learned about the deep learning techniques used for image captioning problem. We experimented with three distinct pre-trained CNN models and compared the results of different models using BLEU scores. We understood that although the weights of pre-trained CNN models were trained on ImageNet, it captured more than enough information about the image to generate captions.

We learned that the result of generated captions is influenced by the training dataset. The Flickr8k dataset contains many outdoor images of humans and dogs and our model gives better results on outdoor images with people and dogs and confuses many different things in other images to dogs and people.

We implemented beam search and found that the BLEU scores for sentences generated using beam search are significantly better.

## Future Work

In this project we implemented a simple show-and tell model, and experimented with different parameters to see the results. Future exploration can be done to compare the current results to those obtained with using different CNN models. Further comparisons can be made to approaches that include visual and temporal attention.

## Acknowledgements

We thank Professor David Crandall for giving us the opportunity to work on the project and for providing the necessary resources. We thank the Associate Instructors of the class for guiding us throughout the project.

## References

[Aarthi and Chitrakala 2017] Aarthi, S., and Chitrakala, S. 2017. Scene understandinga survey. In *Computer, Communication and Signal Processing (ICCCSP), 2017 International Conference on*, 1–4. IEEE.

[Amazon Web Services 2018] Amazon Web Services. 2018. Amazon ec2 p2 instances.

[Aneja, Deshpande, and Schwing 2017] Aneja, J.; Deshpande, A.; and Schwing, A. 2017. Convolutional image captioning. *arXiv preprint arXiv:1711.09151*.

[contributors 2018a] contributors, W. 2018a. Convolutional neural network — wikipedia, the free encyclopedia. [Online; accessed 30-March-2018].

[contributors 2018b] contributors, W. 2018b. Long short-term memory — wikipedia, the free encyclopedia. [Online; accessed 30-March-2018].

[contributors 2018c] contributors, W. 2018c. Recurrent neural network — wikipedia, the free encyclopedia. [Online; accessed 30-March-2018].

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

[Jason Brownlee 2017] Jason Brownlee. 2017. A gentle introduction to calculating the bleu score for text in python.

[Karpathy and Fei-Fei 2015] Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3128–3137.

[Karpathy 2015] Karpathy, A. 2015. The unreasonable effectiveness of recurrent neural networks. Andrej Karpathy's Blog.

[Lin et al. 2014] Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

[Mathews, Xie, and He 2016] Mathews, A. P.; Xie, L.; and He, X. 2016. Senticap: Generating image descriptions with sentiments. In *AAAI*, 3574–3580.

[Rashtchian et al. 2010] Rashtchian, C.; Young, P.; Hodosh, M.; and Hockenmaier, J. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 139–147. Association for Computational Linguistics.

[Rosebrock 2017] Rosebrock, A. 2017. Imagenet: Vggnet, resnet, inception, and xception with keras. pyimagesearch website.

[Shin, Ushiku, and Harada 2016] Shin, A.; Ushiku, Y.; and Harada, T. 2016. Image captioning with sentiment terms via weakly-supervised sentiment dataset. In *BMVC*.

[Simonyan and Zisserman 2014] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Trask 2015] Trask, A. 2015. Anyone can learn to code an lstm-rnn in python (part 1: Rnn). iamtrask github.io blog.

[Vinyals et al. 2015] Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 3156–3164. IEEE.

[Wikipedia contributors 2018a] Wikipedia contributors. 2018a. Bleu — Wikipedia, the free encyclopedia. [Online; accessed 30-April-2018].

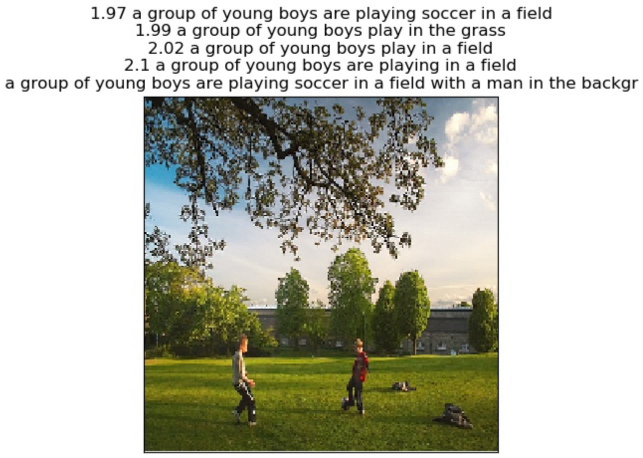
[Wikipedia contributors 2018b] Wikipedia contributors. 2018b. Cross entropy — Wikipedia, the free encyclopedia. [Online; accessed 30-April-2018].

[Xu et al. 2015] Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, 2048–2057.

[You, Jin, and Luo 2018] You, Q.; Jin, H.; and Luo, J. 2018. Image captioning at will: A versatile scheme for effectively injecting sentiments into image descriptions. *arXiv preprint arXiv:1801.10121*.

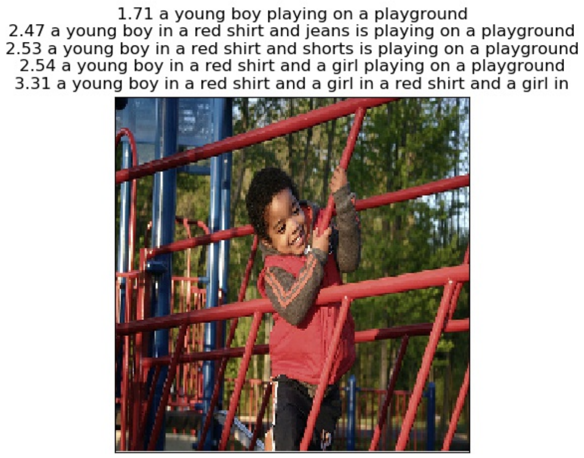
## Appendix : Caption that make sense

The model performs well on images in in figures 11 to figure 20. The captions describe the scenes reasonably well and we can see that the model can generalize well to unseen images (images from the test set).



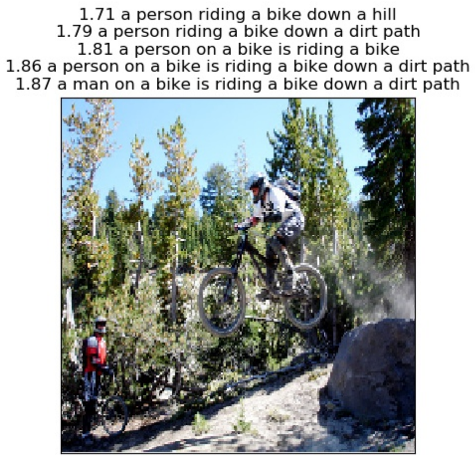
1.97 a group of young boys are playing soccer in a field
1.99 a group of young boys play in the grass
2.02 a group of young boys play in a field
2.1 a group of young boys are playing in a field
a group of young boys are playing soccer in a field with a man in the backgr

Figure 11: Boys soccer: Results using ResNet50, LSTM size 512



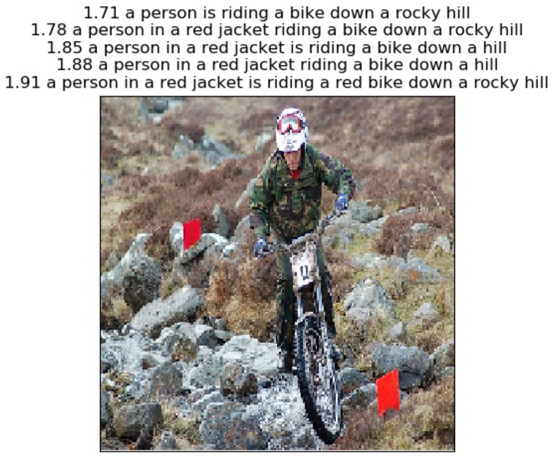
1.71 a young boy playing on a playground
2.47 a young boy in a red shirt and jeans is playing on a playground
2.53 a young boy in a red shirt and shorts is playing on a playground
2.54 a young boy in a red shirt and a girl playing on a playground
3.31 a young boy in a red shirt and a girl in a red shirt and a girl in

Figure 12: Young boy: Results using ResNet50, LSTM size 512



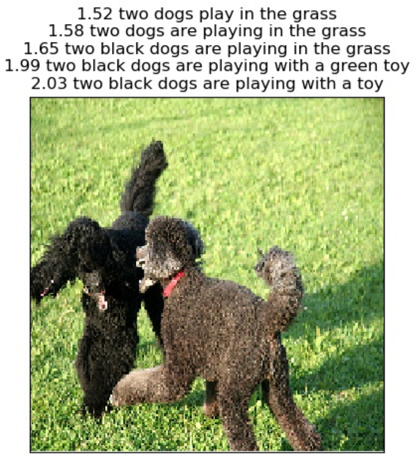
1.71 a person riding a bike down a hill
1.79 a person riding a bike down a dirt path
1.81 a person on a bike is riding a bike
1.86 a person on a bike is riding a bike down a dirt path
1.87 a man on a bike is riding a bike down a dirt path

Figure 13: Person bike: Results using ResNet50, LSTM size 512



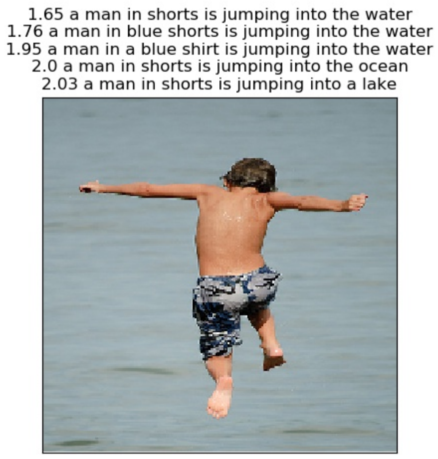
1.71 a person is riding a bike down a rocky hill
1.78 a person in a red jacket riding a bike down a rocky hill
1.85 a person in a red jacket is riding a bike down a hill
1.88 a person in a red jacket riding a bike down a hill
1.91 a person in a red jacket is riding a red bike down a rocky hill

Figure 14: Person bike2: Results using ResNet50, LSTM size 512



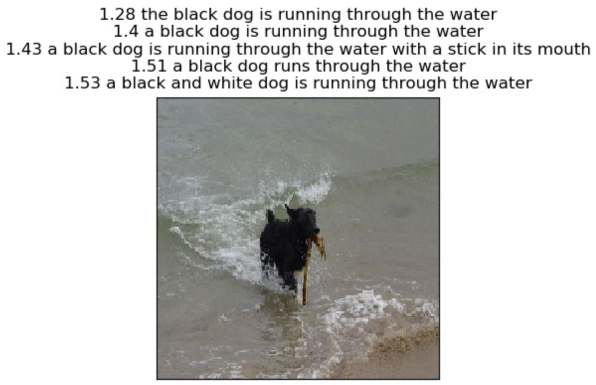
1.52 two dogs play in the grass
1.58 two dogs are playing in the grass
1.65 two black dogs are playing in the grass
1.99 two black dogs are playing with a green toy
2.03 two black dogs are playing with a toy

Figure 15: Two dogs: Results using ResNet50, LSTM size 512



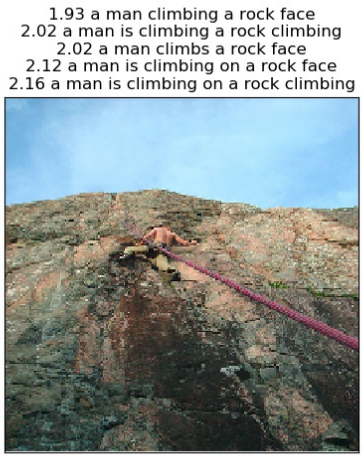
1.65 a man in shorts is jumping into the water
1.76 a man in blue shorts is jumping into the water
1.95 a man in a blue shirt is jumping into the water
2.0 a man in shorts is jumping into the ocean
2.03 a man in shorts is jumping into a lake

Figure 16: Boy jumping: Results using ResNet50, LSTM size 512



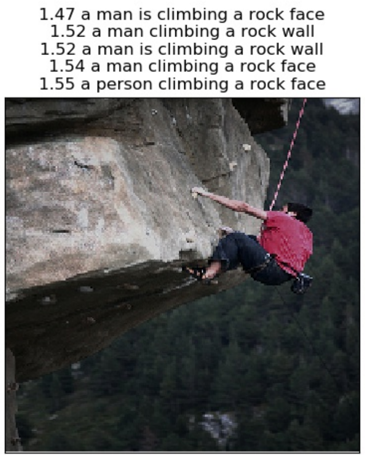
1.28 the black dog is running through the water
1.4 a black dog is running through the water
1.43 a black dog is running through the water with a stick in its mouth
1.51 a black dog runs through the water
1.53 a black and white dog is running through the water

Figure 17: Dog water: Results using ResNet50, LSTM size 512



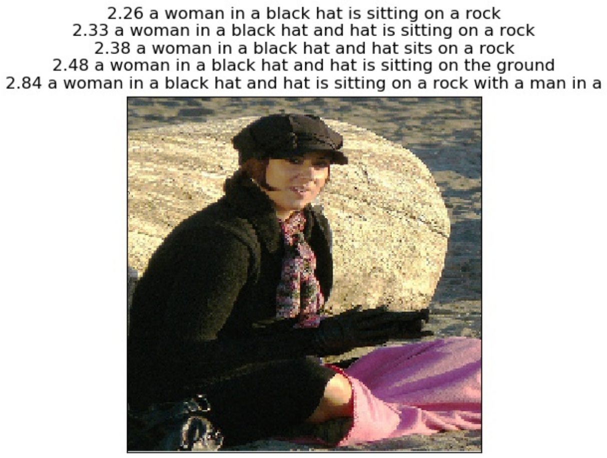
1.93 a man climbing a rock face
2.02 a man is climbing a rock climbing
2.02 a man climbs a rock face
2.12 a man is climbing on a rock face
2.16 a man is climbing on a rock climbing

Figure 18: Man climbing: Results using ResNet50, LSTM size 512



1.47 a man is climbing a rock face
1.52 a man climbing a rock wall
1.52 a man is climbing a rock wall
1.54 a man climbing a rock face
1.55 a person climbing a rock face

Figure 19: Man climbing2: Results using ResNet50, LSTM size 512



2.26 a woman in a black hat is sitting on a rock
2.33 a woman in a black hat and hat is sitting on a rock
2.38 a woman in a black hat and hat sits on a rock
2.48 a woman in a black hat and hat is sitting on the ground
2.84 a woman in a black hat and hat is sitting on a rock with a man in a

Figure 20: Woman hat: Results using ResNet50, LSTM size 512

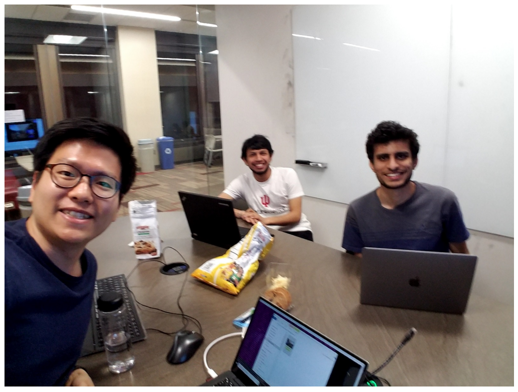


Figure 21: “a man and woman are sitting in a room with a man in a white shirt”: Results using VGG16, LSTM size 300

## Appendix : Caption with more errors

We can see from images in figure 22 to figure 27 that the model gets confused and produces results that are not the best. It can be argued why some of these captions may be meaningful but we avoid doing that here.

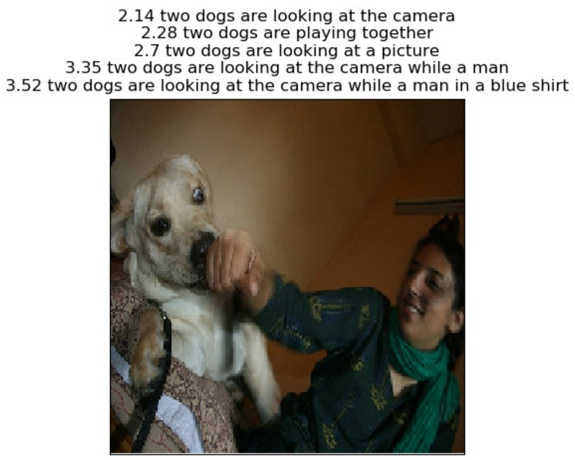

2.14 two dogs are looking at the camera
2.28 two dogs are playing together
2.7 two dogs are looking at a picture
3.35 two dogs are looking at the camera while a man
3.52 two dogs are looking at the camera while a man in a blue shirt

Figure 22: Girl dog: Results using ResNet50, LSTM size 512

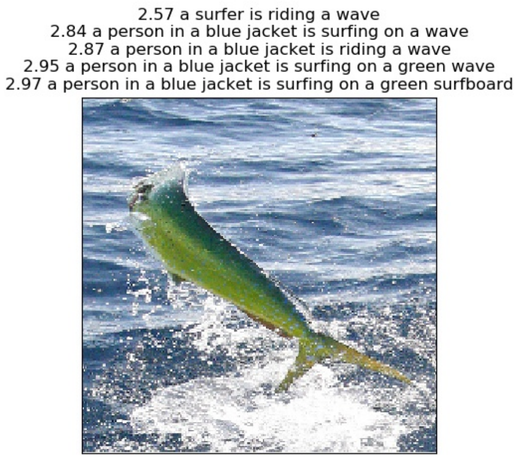

2.57 a surfer is riding a wave
2.84 a person in a blue jacket is surfing on a wave
2.87 a person in a blue jacket is riding a wave
2.95 a person in a blue jacket is surfing on a green wave
2.97 a person in a blue jacket is surfing on a green surfboard

Figure 23: Fish: Results using ResNet50, LSTM size 512

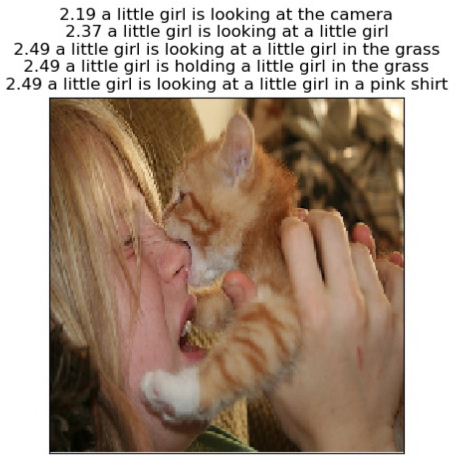

2.19 a little girl is looking at the camera
2.37 a little girl is looking at a little girl
2.49 a little girl is looking at a little girl in the grass
2.49 a little girl is holding a little girl in the grass
2.49 a little girl is looking at a little girl in a pink shirt

Figure 24: Girl cat: Results using ResNet50, LSTM size 512

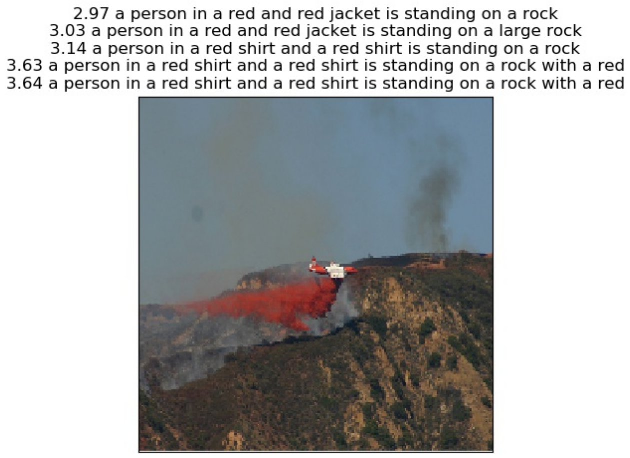

2.97 a person in a red and red jacket is standing on a rock
3.03 a person in a red and red jacket is standing on a large rock
3.14 a person in a red shirt and a red shirt is standing on a rock
3.63 a person in a red shirt and a red shirt is standing on a rock with a red
3.64 a person in a red shirt and a red shirt is standing on a rock with a red

Figure 25: Airplane mountain: Results using ResNet50, LSTM size 512

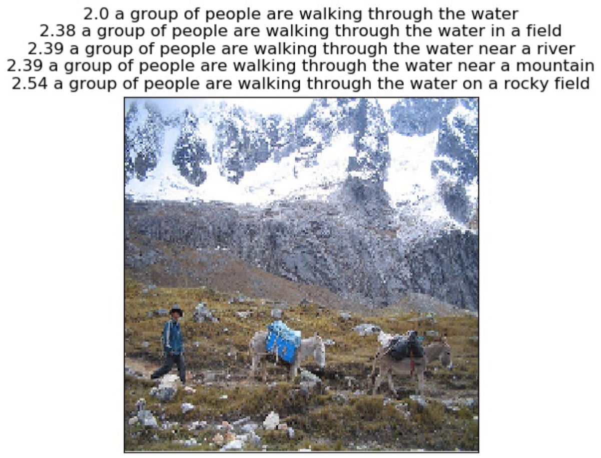

2.0 a group of people are walking through the water
2.38 a group of people are walking through the water in a field
2.39 a group of people are walking through the water near a river
2.39 a group of people are walking through the water near a mountain
2.54 a group of people are walking through the water on a rocky field

Figure 26: Horses mountain: Results using ResNet50, LSTM size 512

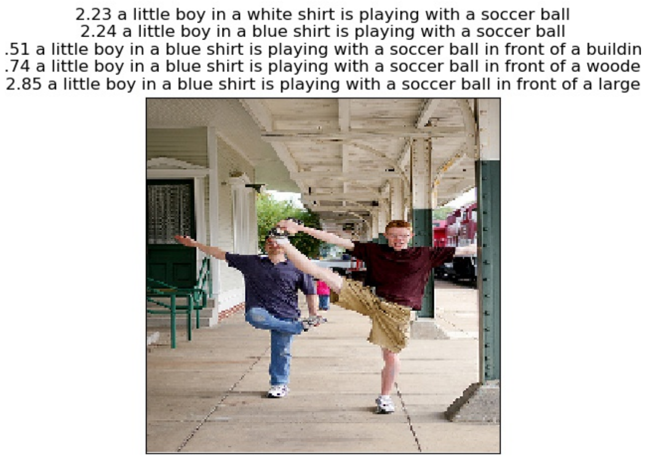

2.23 a little boy in a white shirt is playing with a soccer ball
2.24 a little boy in a blue shirt is playing with a soccer ball
.51 a little boy in a blue shirt is playing with a soccer ball in front of a buildin
.74 a little boy in a blue shirt is playing with a soccer ball in front of a woode
2.85 a little boy in a blue shirt is playing with a soccer ball in front of a large

Figure 27: Two men: Results using ResNet50, LSTM size 512