**Image Captioning**

CMPE 258 Deep Learning

Fall 2021

Group 5

Vishalsinh Gadhvi 015231450

Leonid Grekhov 012152608

Himayu Mahnot 015242357

Adam Romayor 015447731

Vishwa Shah 015226523

# 

[**Introduction**](#_3yjtscz4gba9) **3**

[**Literature Review**](#_k4tn1zkuttz2) **3**

[**Data Exploration**](#_cxqd4rg141he) **4**

[**Data Preprocessing**](#_y95h5zh51upd) **6**

[Image Processing](#_islcooprn9rk) 6

[Caption Processing](#_6zgnk4dn1b32) 6

[**Problem Formulation**](#_scchfpoxw1yg) **6**

[**Model Selection**](#_1g8d7ji8xtgv) **7**

[The Image Input](#_otutzj2nq506) 7

[The Caption Input](#_wwyhhyadsuql) 7

[NLP Model](#_n2n4l2izbrvr) 7

[Attention Model](#_pjc1mhdrq0jg) 8

[The Model Output](#_ifx1e92sgozu) 9

[Training](#_rb4aht1rgfm9) 9

[Caption Generation](#_ido77fglib40) 9

[**Evaluation and Result Analysis**](#_pwex86t8x7gs) **10**

[**Visualization**](#_mehjnlgby1xp) **11**

[Vgg16 and Bidirectional LSTM](#_efwrgp3sg514) 11

[InceptionV3 and Bidirectional LSTM](#_kw4s1cqccdg) 12

[Resnet50 and LSTM](#_oz0te7z75c5d) 13

[InceptionResNetV2 and LSTM](#_imgrwkf9ahua) 14

[DNN201 and Bidirectional LSTM](#_6bp6kmg6p11f) 15

[InceptionV3 and Attention Model](#_2quez2kpcm1q) 16

[**Conclusion**](#_no01ii9odepi) **16**

[**Future Work and Improvement**](#_j8mt7omp0bki) **17**

[**References**](#_l7i32rty7po1) **17**

# 

# Introduction

For our project, we participated in the VizWiz Image Captioning Challenge 2021. Image captioning is the task of describing the contents of an image in a few words. The goal of this project was to generate a caption for a given image. The purpose of this challenge is to help remove any accessibility barriers that are faced by blind people. Blind people heavily rely on image captioning services to understand the world around them and the aim is to use technology and data science to assist them by automatically generating a caption. We were provided with 23,431 training images and 117,115 captions.

# Literature Review

Image caption generator follows the encoder-decoder architecture. The reason encoder-decoder architecture best suits the image captioning is that the image is translating to the words. And as per the previous research, translation operation is well performed using the encoder-decoder architecture.

Automatically creating captions for images is a process that is important to scene interpretation, one of computer vision's core aims. Not only can caption generation models resolve the computer vision issues associated with recognizing which items are in an image, but they must also be capable of capturing and expressing their connections in plain language. As a result, caption creation has long been regarded as a challenging task. It is a significant issue for machine learning algorithms, as it entails replicating the exceptional human capacity for compressing massive amounts of meaningful visual information into descriptive language [1].

The attention Model is the evolutionary idea for natural language processing. In Show, attend, and tell paper significantly improved the image captioning model's performance by combining the convolutional network for decoding the image and recurrent neural network to generate caption. But here, before feeding the hidden state to the next layer, an attention model is used to create attention on the feature vector, which is useful to generate the next word. To evaluate the model performance, three standard datasets were used: Flickr 8k, Flickr30k, and the MS COCO dataset.

Here two attention-based image caption generator frameworks were created.

1] Hard stochastic attention mechanism

2] Soft deterministic attention mechanism trainable by standard back-propagation methods.

Model details :

**Encoder**: Convolution layers

The model generates the feature vector from the input image one at a time and generates a caption of length C. Each word is represented as the encoded vector of size equals the dataset vocabulary size.

In order to make the feature vector and input word vector the same size, the 2D feature vector is transformed into the 1D vector by using the fully connected layer.

**Decoder**: Long Term Short Term Memory Network

LSTM language model is used to generate the next word as each timestep is based on the image context vector and previously generated word.

**Stochastic Hard Attention**

Generally, in the hard attention, instead of calculating the sum of the encoder hidden state here, it would be the CNN output feature vector. Only the selection of the hidden states are used to calculate the attention score.

**Stochastic Soft Attention**

In contrast to Hard attention, all CNN's output feature vectors are weights sum to generate the attention score.

**Regularization Method**

The VGG model trained on the ImageNet dataset without fine-tuning was used here. The output dimension of the encoder is 196 \* 512. So here, dropout is used on the encoder side. Another regularization strategy was the early stopping based on the BLEU score.

The attention model used for the caption generator gave the best state-of-the-art performance based on the BLEU and METEOR metrics.

# Data Exploration

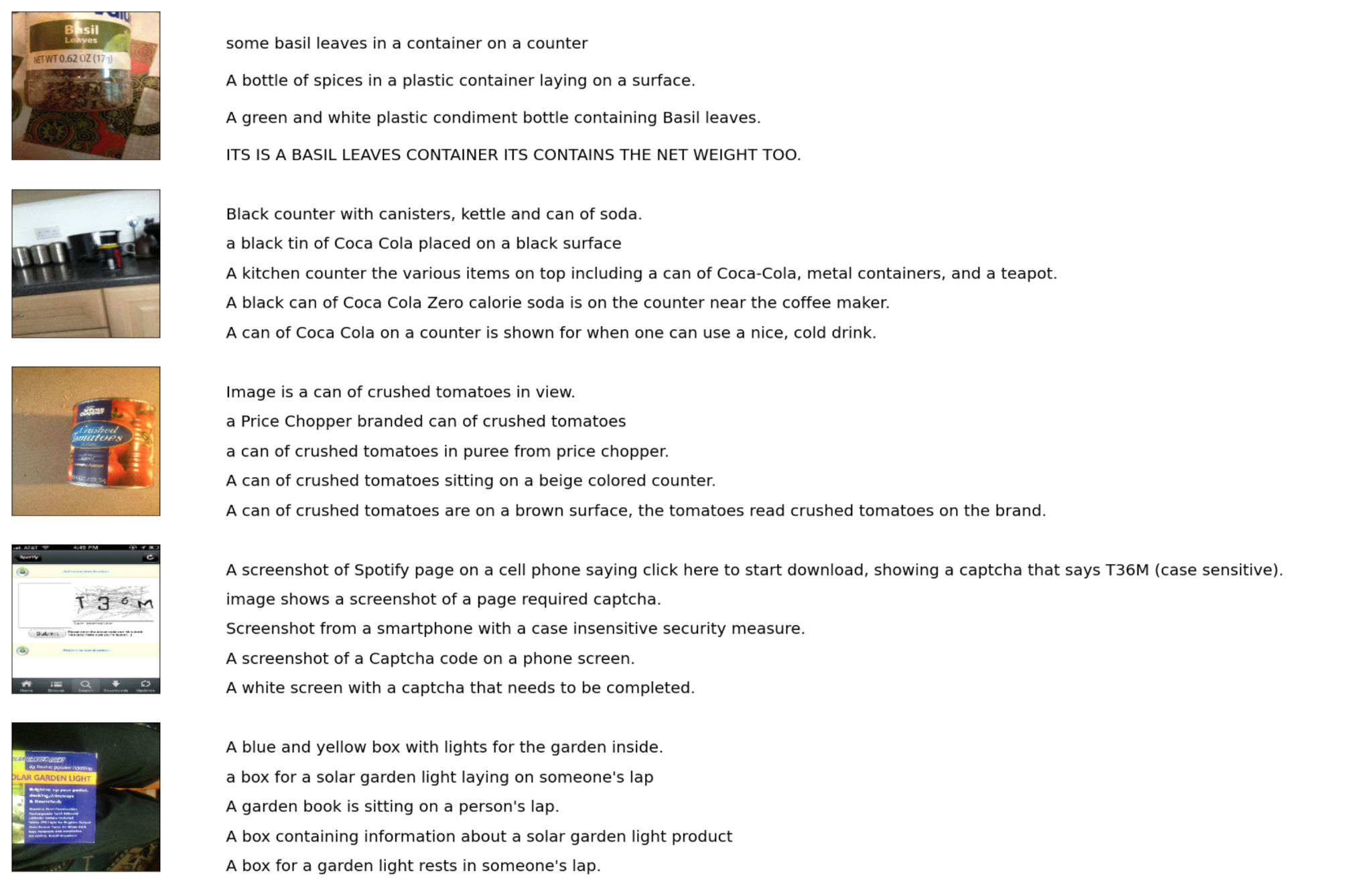
The data is from the VizWiz Caption dataset. The original dataset had 23,431 images with 117,155 captions for those images. This also had 7,750 validation images with 38,750 captions and a final test set with 8,000 images and 40,000 captions.

# 

**Figure 1.** The format for the caption dataset.

The captions came in json format and mapped the caption to the correct image using the image\_id. In figure 1 there are also three flags for each capiton. The first, is\_precanned, indicates if the caption was pre-prepared text, or if it was authentic. The pre-canned text on these images was: “Quality issues are too severe to recognize visual content.” The second, is\_reject, indicates if the caption was rejected or accepted. A caption would be rejected if it was considered spam, or if it was not relevant to the photo. The third, text\_detected, indicates if there is text in the photo.

In this dataset, there were up to five captions for each photo.



**Figure 2.** A photo and the corresponding captions.

# Data Preprocessing

There are two steps to preprocessing the data. The first step is to preprocess the images and the second step is to process the captions using various techniques.

## Image Processing

* We used transfer learning to pass an image through a CNN encoder
* We used different models to extract the feature vectors: VGG16, InceptionV3, Resnet50, InceptionResNetV2, DNN201
* Since we did not need to classify the images, and only needed the feature vectors, we removed the last dense layer which is used to make classifications
* This output (feature vectors) would be used as an input to our NLP model which will be used to generate the captions.

## Caption Processing

* First, we remove all the captions that are rejected or precanned. This helps our model avoid learning noise and being biased to such data.
* Then we convert all the text to lowercase and remove all the punctuations from each input caption.
* Following that, we append “startseq” and “endseq” to each caption
* The output would look like this:
  + Original: “ITS IS A BASIL LEAVES CONTAINER ITS CONTAINS THE NET WEIGHT TOO.”
  + Preprocessed: “startseq its is a basil leaves container its contains the net weight too endseq”
* After each caption is preprocessed, we create a vocabulary based on training captions
* Then we create a dictionary to map words to an index, and a separate dictionary to map an index to a word.
* Finally, we create an embedding matrix where each column in the matrix represents the GloVe Embedding of that word

# Problem Formulation

This problem required two separate models to generate captions. The first model takes the feature vectors of the preprocessed images, and goes through a feed-forward dense neural network. The next model processes the captions using an NLP model. The input to the models will be the feature vector of an image and a zero-padded sequence of word embeddings. The final output of the model are the probabilities of the next word in the sequence.

# Model Selection

There are two separate parts of the model that were experimented with. The first model tested was the encoder to preprocess the images. The CNN models that generated the feature vector for each image. Transfer learning was applied for each CNN model, and the weights were frozen. For each of these models, the final classification layer was removed. Removing the classification layer allows the feature vectors to be passed to the main model. The CNNs tested were VGG16, InceptionV3, Resnet50, InceptionResnetV2, and DNN201.

The next part of the model that was experimented with is the Natural Language Processing model. Three different architectures were attempted. The first model was a basic LSTM layer with 256 nodes. The next model tested was a Bidirectional LSTM model. The final NLP model test was an attention model.

## The Image Input

The first input to the model is the feature vector of a single image. This portion of the model then passes through a Dropout layer and two Dense layers. The dropout layer has a rate of 0.5 to help reduce overfitting. The first Dense layer has 512 nodes and the second has 256 nodes. Both dense layers have a ReLu activation function.

## The Caption Input

The input to this part of the model is a zero-padded sequence of word embeddings. This sequence is the same length as the maximum caption length. The next layer is an Embedding layer with input dimension set to the vocab size and the output dimensions set to 200. The output dimension is 200, because the embedding matrix has 200 element vectors. Mask\_zero is set to true, because the inputs are zero-padded sequences. This embedding layer has the weights set to the embedding matrix described in the Data Preprocessing section. Then, these weights are frozen to prevent further training. The embedding layer is followed by a dropout layer with a rate of 0.5 to reduce overfitting. After the dropout layer the NLP model is added.

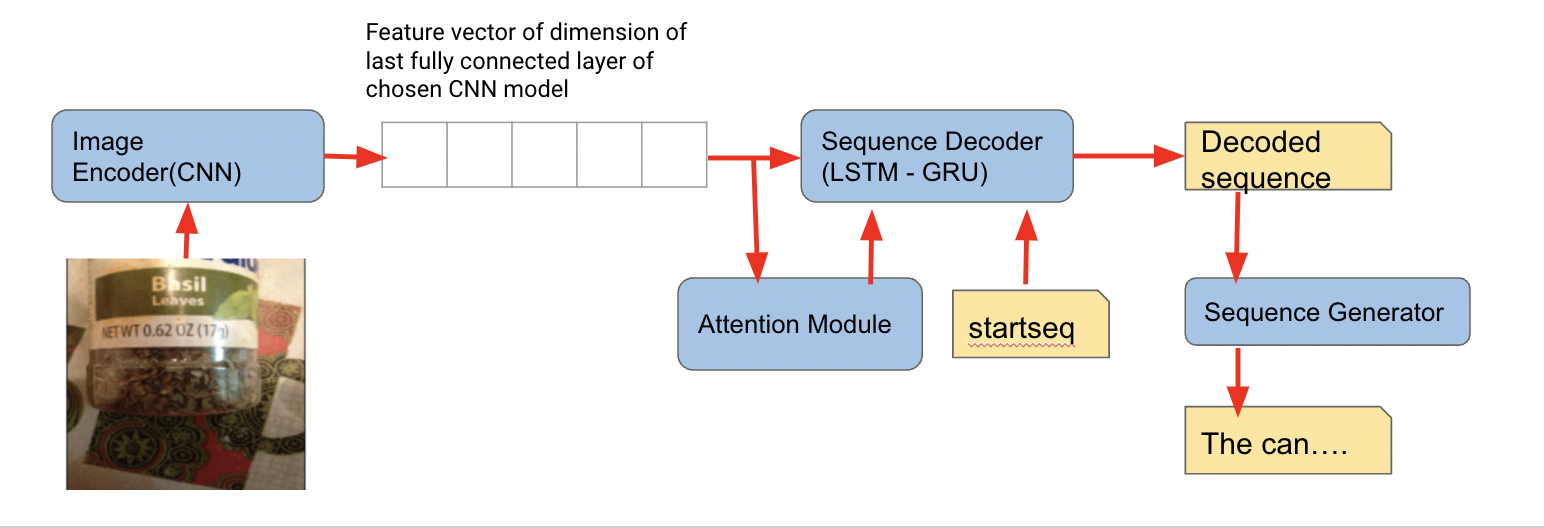
## NLP Model

The simple LSTM model has a single layer with 256 nodes. This output dimension is the same as the output shape of the model with the image input.

The bidirectional LSTM model has two Bidirectional layers with 128 nodes each. The first layer has return\_sequences set to true, so these sequences can be passed to the second layer. The output shape of this model is also 256.

## Attention Model

One innovation has been made in the image captioning architecture is to apply attention which helps the model focus on the most relevant portion of the image while generating word in each timestep. The motivation got from the show, attend and tell paper.



**Figure 3. General Architecture of the image caption generator**

Here the raw image has been passed to CNN model and output is the feature vector. Now that feature vector is passed to the attention module.Output of the attention module is the context vector having the modified pixel value and the attention weights. So the first word of the sequence which is always the startseq and the context vector is fed to the sequence decoder and this loop runs till the “endseq” word is found. In the sentence generator, it takes the decoder sequence of the word and produces the probability of each word from the vocabulary for each position in the predicted sequence.

Input and output dimension of the each layer in the model:

**Step 1** : Encoding image

Input : (batch\_size, 299, 299)   
Output : (batch\_size, 8,8, 2048)

**Step 2**: Tokenize Captions

Input : Sequence of english words

Output:

Word to index and index to word dictionary

Size of dictionary = Vocabulary size

**Step 3**: Pad all the captions with max length of the caption in dataset = 81

**Step 4:** CNN encoder

Input: (batch\_size, 64, 2048)

Output: (batch\_size, 64, embedding\_dim)

**Step 5:** Language Model with attention

Input: Feature vector with attention weights + word vector of <startseq>

Output: (batch\_size \* max\_length, vocab\_size)

Probability distribution of each word in the vocabulary.

**Optimizer** : Adam   
**Metrics** : Categorical cross entropy loss

## The Model Output

Both the image input and the caption input models are combined using an add layer. This goes through an additional Dense layer with 256 nodes and a ReLu activation function. The final layer is a Dense layer with an output dimension of the vocab size. This layer has a softmax activation function.

## Training

Once the model is created, a data generator is used to pass the correct inputs to the model. A caption is broken up into multiple sequences. The input to the model is a single feature vector, and a zero-padded sequence of the caption. The output of the model is the probabilities of the entire vocabulary. The word with the highest probability should be the next word in the sequence.

The model compiled with the Adam optimizer and the categorical\_crossentropy loss function. This loss function is chosen, because this is a classification problem where each class is a word in the vocabulary. The different models were trained for up to 100 epochs. Each model was trained on 1,000 images and up to 5 captions per photo.

## Caption Generation

A caption is generated by passing the same feature vector of the photo to the model with the updated sequence of words. This process is started with a zero-padded sequence containing only the ‘startseq’ token. The model will output probabilities for the next word in the sequence. The word with the highest probability will be appended to the sequence and passed to the model again, with the same feature vector. This process ends when the ‘endseq’ token is seen, or the max length of the caption has been generated.

# Evaluation and Result Analysis

The Bleu score was used to evaluate each mode for its effectiveness. The bleu score calculates how accurate a model is by checking how many words in a caption are correct on a scale from 0 being not accurate to 1 being 100% accurate. In the example of bleu-1 we evaluate how often 1 word is accurate in a caption. On the other end of the spectrum in bleu-4 we evaluate out of 4 words how accurate is the caption. As it is difficult to capture the entire caption, most models were able to classify at least 1 word per caption fairly often but tended to do worse with more words in the caption. One of the ways the bleu score could be improved is with more images used in the training of the models as only 1000 images were used for the first 5 models.

| **Table 1.** Bleu 1-4 scores for all the models tested. | | | | |
| --- | --- | --- | --- | --- |
| **CNN Model Name** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **Bidirectional LSTM** | | | | |
| **VGG16** | 0.45 | 0.20 | 0.06 | 0.02 |
| **InceptionV3** | 0.41 | 0.19 | 0.33 | 0.44 |
| **Resnet50** | 0.50 | 0.23 | 0.38 | 0.48 |
| **InceptionResNetV2** | 0.38 | 0.39 | 0.47 | 0.53 |
| **DNN201** | 0.02 | 0.13 | 0.26 | 0.36 |
| **Attention Model** | | | | |
| **InceptionV3** | 0.13 | 0.34 | 0.48 | 0.56 |

# Visualization

## Vgg16 and Bidirectional LSTM

****

**Figure 4.** A box of chicken sitting on a wooden surface.



**Figure 5.** A hand holding a food can with a red label in it.

Figures 4 and 5 show two captions that were very close to the ground truth. Figure 3 shows the model correctly identifies a box of food on a wooden surface. However it predicts chicken instead of beef. In Figure 4, the model accurately identifies the hand holding a can of food, and even indicates that the can has a red label.

## InceptionV3 and Bidirectional LSTM



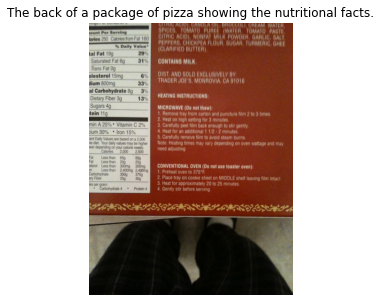
**Figure 6.** A person is holding a box of hersheys pills.



**Figure 7.** A can of great value corn sits on a counter.

Figure 6 shows how the classifier can be incorrect in the classification. The only correct part of the caption is the detection of a box in the image. In figure 7 the classifier only partially classified the image detecting a can but missing all other key words.

## Resnet50 and LSTM



**Figure 8.** The back side of a box of dog food with the instructions showing on the back.



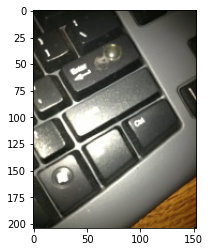
**Figure 9.** A person is holding a can of beer in the hand.

Figure 8 shows how the classifier can over classify an image. The image is described in great detail identifying the back of a box, instructions. Figure 9 compares the results of the classifier to another model and we observe that the classifier in the case recognizes more information such as the person holding a can and that it is in their hand. The classifier still misclassified the can as a can of beer.

## InceptionResNetV2 and LSTM

## 

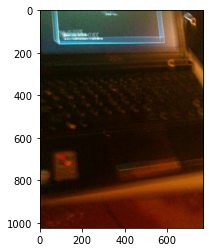
**Figure 10.** A white bottle of some sort of brand product.



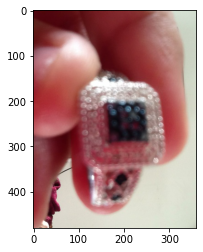
**Figure 11.** A small electronic keyboard for a small screen or surface.

Figure 10 and Figure 11 show the sample captions generated by the model. In Figure 10, it was able to predict the caption which was very close to the ground truth. However, in Figure 11 although the caption generated was descriptive of the context described in the picture, it was not able to generalize well as the model described the keyboard being small but in reality it was a full-sized keyboard.

## DNN201 and Bidirectional LSTM



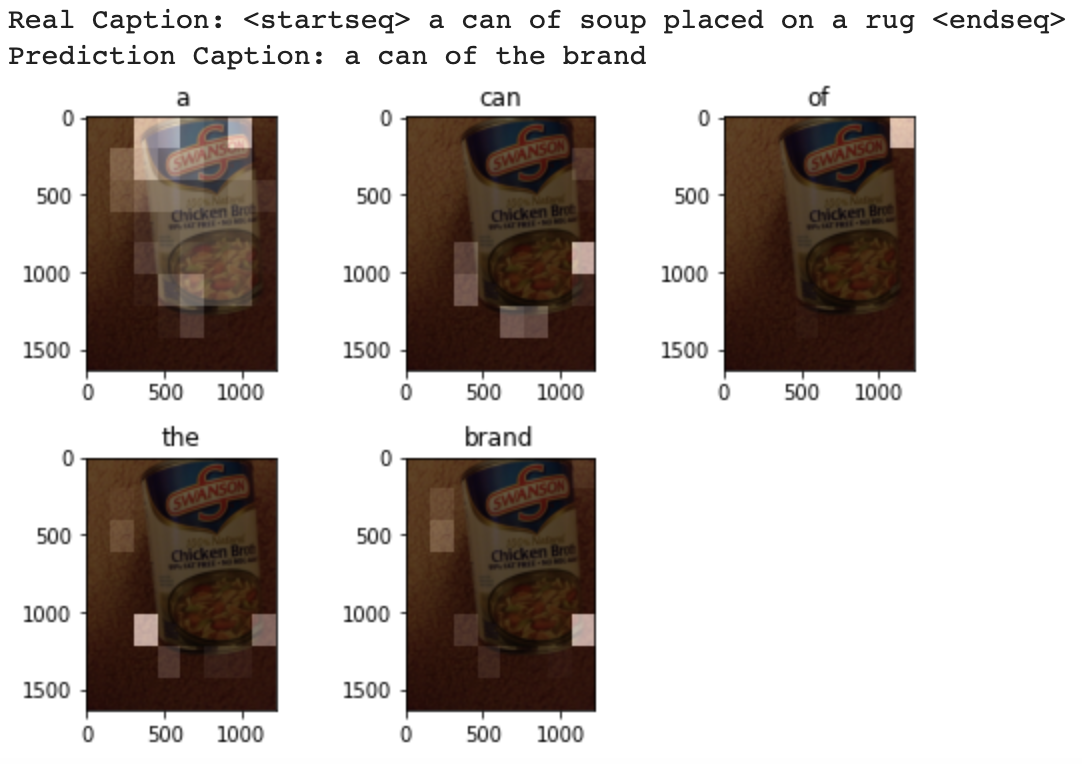
**Figure 12.** The keyboard and bottom screen of a laptop.



**Figure 13.** Someone is holding a ring in his hand.

Figures 12 and 13 show two captions that were very close to the ground truth. Figure 12 shows the model correctly identifies the laptop. However the caption sentence formation seems to be inappropriate. In Figure 13, the model accurately identifies the hand holding a ring.

## InceptionV3 and Attention Model



**Figure 14.** Attention plots and the predicted words.

In Figure 14, the article and preposition are coming frequently in the generated caption so observation is that we need to remove the article from the training dataset to avoid the model giving attention to the non useful word in describing an image. Another observation is that, in the second sub image of Figure 14 where the model predicted the “can '' which can be assumed by the edge of the object. So here we can determine that the model is well trained on the generating word based on the edge of the object.

# Conclusion

Although we tested many models one of the biggest conclusions we came to is that more images and more training time is required to improve our bleu scores and improve image captioning. Most models were trained on 1000 images over 30 epochs which took around 3 to 4 hours to train on google colab. When selecting our best model our results were subjective as some models did better at predicting at least 1 word in the caption more often while other models were more consistent at predicting more than 4 words per caption. One of the challenges we faced was that some of the models tended to overgeneralize the images. In some examples it is clear that the model would detect objects in the image but not accurately describe other existing objects or fail to classify them. Our best model was the InceptionV3 with attention as it was the most consistent. This was well depicted in the bleu score as the score rose with more detected words. DNN201 and Bidirectional LSTM model was one of the most descriptive and accurate models as well.

# Future Work and Improvement

As stated in the conclusion most improvements could be achieved with more training data and training time. Due to limitations such as poor internet for uploading more images to increase training size and lacking computational power improvement on current models is difficult. Other possible improvements include implementing a vision transformer. With more time this could be achieved but it is unknown if the computational power of google colab would be sufficient for training the new model.

# References

[1] K. Xu, J. Lei Ba , R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel , and Y. Bengio, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention,” *In International conference on machine learning*, pp. 2048–2057, 2016.