NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Design & Innovation Project (DIP)

Project Report

Design of an Automatic Review System

Project Group: E012

School of Electrical and Electronic Engineering
Academic Year 2020/21
Semester 1

Table of Contents

1. Introduction

Food heavily influences customers' physical well-being and is a critical contributor to their pleasure, worry, and stress (Rozin, P, 1999). On one hand, customers are exploring and looking for better food resources, on the other hand, all food suppliers are trying to provide competitively priced food products with high quality and better service (Wilcock, A, 2004). Therefore, food online ordering systems came out. The main purpose of an online ordering system is to offer customers an efficient way to make an order at a restaurant over the internet connection. With mobile applications and websites, customers can easily browse all available dishes in restaurants, customize orders to their requirements, and place orders. The needs of ordering food online are surging for recent years and especially for COVID_19 period (Chang, H, 2020).

Nevertheless, the information provided by the business side is not a strong incentive for people to trust the quality of food. People need product validation from other customers to evaluate the alternative of the product (Mudambi, S, 2010). According to Search Engine Land's statistics, the overwhelming majority (90 percent) of customers read 10 reviews or fewer before they feel that they could trust business and more than half (60 percent) of these customers are looking for ordering food online. Although customer review is a critical component of online ordering services, writing a review is very time-consuming and customers often skip this process. Hence, we need a tool that could simplify the process of writing reviews to encourage customers to generate reviews. In this project, we aim to develop an automatic review system to assist users in writing review by automatically generating a review based on a given image.

An automatic review system needs two parts -- food image processor and caption generation. Most previous attempts have proposed to apply image processing on food to estimate food calories and analyze people's eating habits for healthcare and also the safety level of food (Bossard, L, 2014). Also, many researchers are exploring image caption generators to help understand the scene and generate sentences over provided images (Baig, M, 2018). Thus, we proposed to utilize previous works and develop a system containing both components.

In this report, we combined food image processor and caption generator to develop an automatic review system(ARS) to help customers easily write reviews and hence help the business side get more potential to sell out food online. After we reviewed the development process of image-caption-generator methods in recent years and compared their performances and costs, we chose to use the encoder-decoder framework which is easier to implement and costs less memory (Vinyals, O, 2015). This separated parts framework allows us to generate encoded features for image by using a Convolutional Neural Network (CNN, Encoder) and then generate a caption with extracted image features from CNN by using a Recurrent Neural Network (RNN, Decoder). Both CNN and RNN excel at their respective tasks, those being object identification and classification, and natural language processing tasks like machine translation respectively. With our proposed ARS, customers only need to take a photo of food and can easily have a review-suggestion which is automatically generated by ARS.

2. Project Summary

In order to create an automatic review system over food images, we tried to train a model which could generate captions based on food images provided by input. This kind of model normally needs two parts – food image processor and caption generator. Therefore, our model is in Encoder-Decoder architecture and we used some useful mechanisms, such as attention algorithm, word to vector, transfer learning and beam search. This separated parts framework allows us to generate encoded features for a food image by using a CNN (Encoder) and then generate a caption with extracted image features from CNN by using a Recurrent Neural Network (Decoder). To improve our models, we tried changing the size of word map, the dimension of embedding layer and even using pre-trained embedding layer (Glove). Our team is divided into three sub teams and each is provided with different jobs. Team 1 applies Encoder-Decoder architecture model without attention mechanism (Model 1: InceptionV3 + LSTM) on our food80k dataset, team2 applies Encoder-Decoder architecture model with attention mechanism (Model 2: InceptionV3 + LSTM) on MSCOCO dataset and team 3 applies Encoder-Decoder architecture model with attention mechanism (Model 3: ResNet101 + LSTM) on food80k dataset. We see some good results but also some unsatisfied output. Due to the variety of food and limited training dataset and time, there are still some limitations in our model, but we already have a clear plan about how to improve it furthermore.

3. Scope

3.1 Background Knowledge

In this section, we introduce the modules and also some mechanisms applied in our project.

1) Convolutional Neural Network (CNN)

One of the methods used to achieve object detection, image recognition, face recognition and image classification is the Convolutional Neural Network or CNN. CNN is the process of taking in an input image, process it and categorizing it (Prabhu, 2018).

Unlike humans, computers view images as a set of pixels. The resolution of an image does play an important role too. Based on the resolution size of an image, the machine views it as Height x Width x Dimension.

In CNN, there are hidden layers inside the Convolution layer that train and test each input image. Such hidden layers are:

- Convolution (with ReLU) layer
- Pooling layer
- Fully connected layer

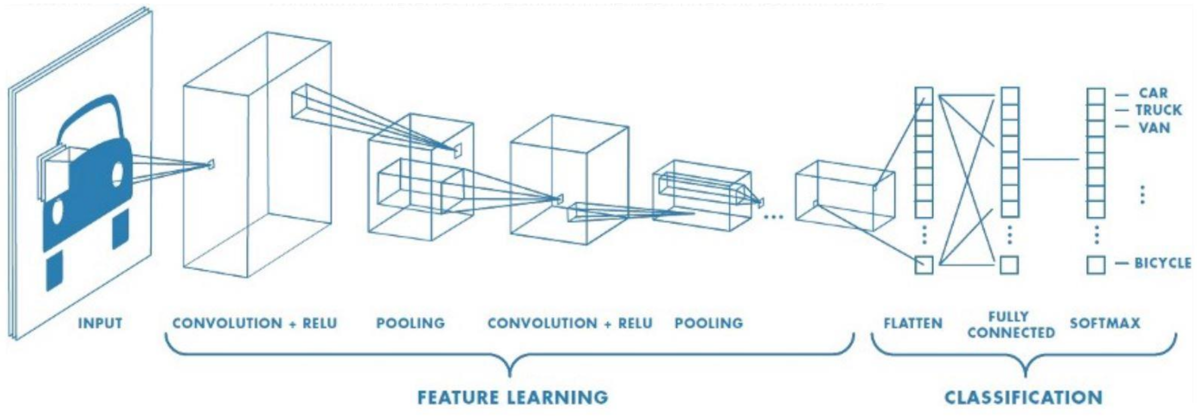


Figure 1: CNN process. Source: (Prabhu, 2018)

2) Recurrent Neural Network and Long Short-Term Memory

The Recurrent Neural Network (RNN) is a type of Neural Network. In this network, the output that is given out from the previous step, is passed on to the current step as their input.

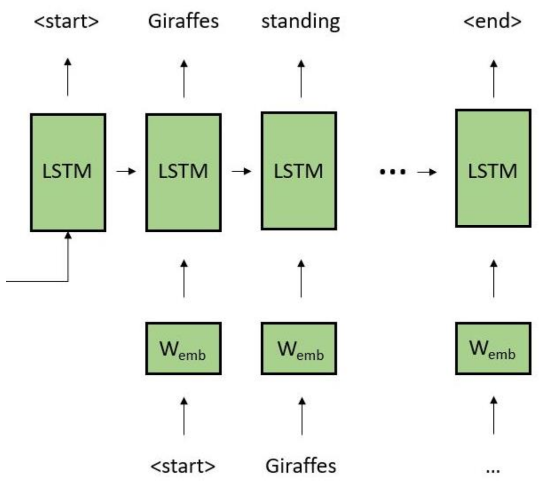


Figure 2: RNN (LTSM) process. Source: (Aladdin, 2020)

As you can see from above, the output word "Giraffes" is being passed on to the next input. In this model, we are using "Long Short-Term Memory" or LSTM for short. It is a type of RNN that is able to learn order dependence in sequence prediction problems. LSTM is the solution to the sequence prediction problems.

3) Rectified Linear Unit (ReLU)

The ReLU is an activation function that is widely used in the deep learning models today. Said function will return a "0" if the input is a negative value. For a positive input, it will return the value back.

$$f(x) = max(0,x)$$

Being a nonlinear activation function, ReLU is important because functions that the model is trying to learn is nonlinear. Without using a nonlinear activation function, the model will not be able to achieve features such as image classification or text prediction.

4) Pooling Layer

The role of the pooling layer is to maximize the capabilities of the convolution. This is achieved by "Max Pooling". Max pooling is done by taking a small sized pixel group, preferably 2x2 and applying it on the input image.

Max pooling will take the largest value amongst the selected group of pixels.

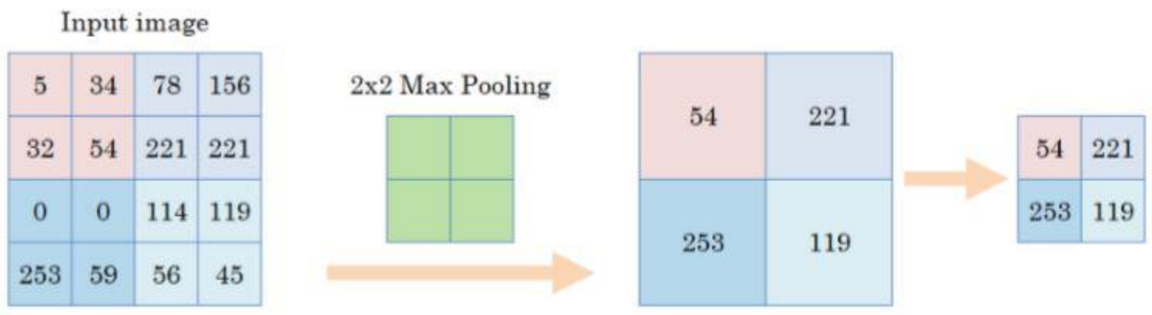


Figure 3: Max pooling process. Source: (Zafra, n.d.)

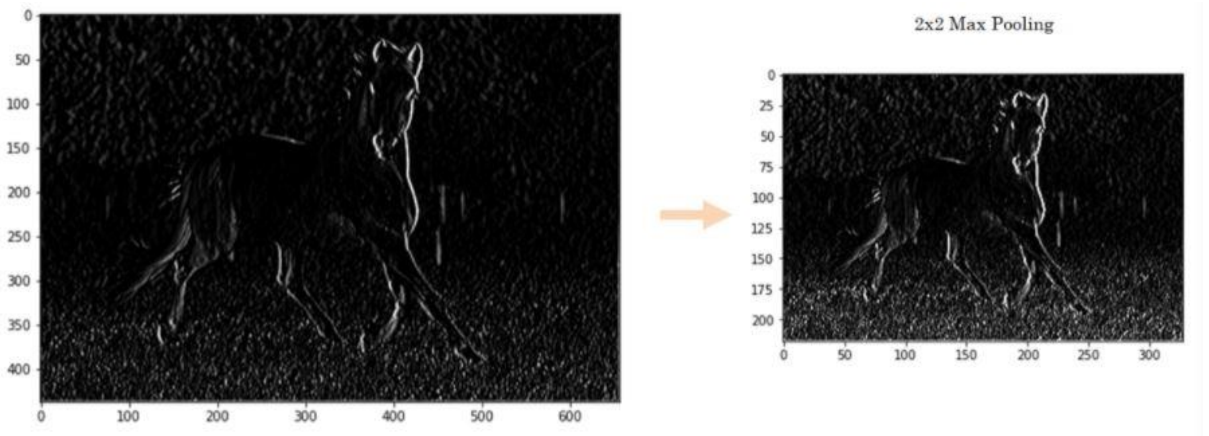


Figure 4: Result after applying Max pooling. Source: (Zafra, n.d.)

After applying the Max pool function, the overall size of the image is being reduced. The size is being reduced by taking groups of 2x2 pixels and keeping only the maximum values. Also, the vertical edges are also highlighted more prominently.

5) Dropout

In a deep neural network with large parameters, overfitting tends to occur. An overfitted model learns the training dataset too well, resulting in a good performance on the training dataset but performs poorly when given an unseen dataset. One of the main reasons that causes overfitting is co-adaptation. (Peng, 2018)

To overcome co-adaptation, dropout is used. It works by temporarily omitting randomly chosen neurons in each iteration during training. We used a dropout probability value of 0.5 which indicates the probability of a neuron, in the hidden layer, being removed. 0.5 is a commonly used value for the hidden layers as it was found to be the optimal probability for a wide range of neural networks. (Hinton, 2012)

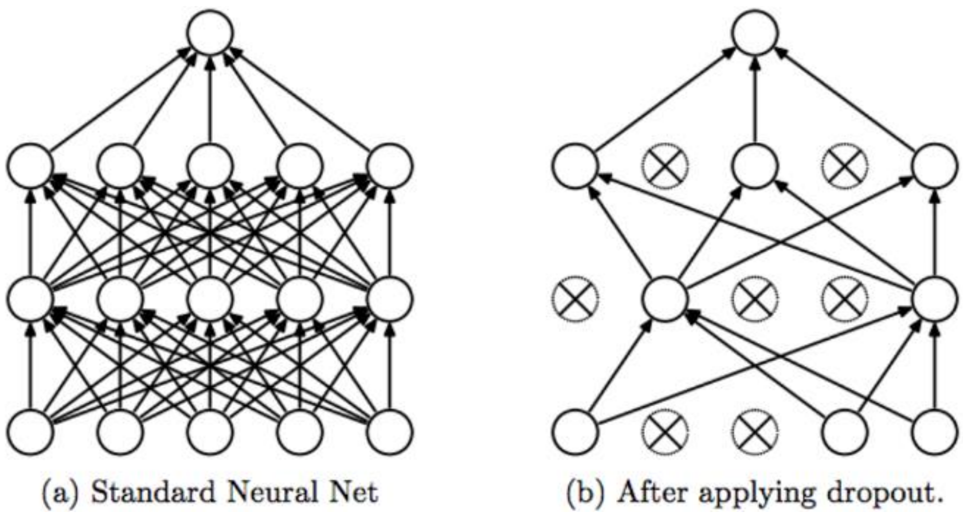


Figure 5: Dropout Diagram. Source: (Srivastava, 2014)

6) Evaluation Metrics

Bleu Score 1 to 4

Bilingual Evaluation Understudy (BLEU) is one of the scoring algorithms used to evaluate the quality of text generated or translated by a machine.

The way to calculate using the BLEU algorithm is by comparing the captions with the “n-gram” method (IBM, n.d.).

Example sentence: “Once you stop learning, you start dying”.

Unigram (1)	Bigram (2)	Trigram (3)
Once	Once you	Once you stop
you	you top	you stop learning
stop	stop learning	stop learning, you
learning	learning you	learning, you stop
you	you start	you start dying
start	start dying	
dying		

Table 1: n-gram demonstration

BLEU will compare each n-gram candidate(predicted) and reference captions and count the number of matches. If there are more matches between both captions, the better it is.

For BLEU-1, we will use the unigram.
For BLEU-2, we will use the bigram.
For BLEU-3, we will use the trigram.
For BLEU-4, we will use the n-gram.

Precision is how we determine the score for each caption for the BLEU algorithm. It is defined as:

$$\text{Precision} = \frac{\text{No. of candidate translation words occuring in any reference translation}}{\text{Total no. of words in the candidate translation}}$$

Reference caption: The cat is on the mat.
Example candidate(predicted) caption 1: the the the the the the the the.
Example candidate(predicted) caption 2: the cat is mat the on.

Precision for candidate caption 1 is 2/7 (0.28571428571)
Precision for candidate caption 2 is 7/7 (1)

METEOR

Metric for Evaluation of Translation with Explicit ORdering (METEOR) uses harmonic mean of the unigram. METEOR was made to solve the problems found in the BLEU metric (Graham et al., 2018).

In METEOR, the score is calculated by having the number of unigrams found in the predicted caption that are also in the reference caption and divide it by the number of unigrams found in the predicted caption.

ROUGE

ROUGE or known as Recall-Oriented Understudy for Gisting Evaluation, is a metric used to evaluate machine translation. Similar to BLEU, ROUGE also uses the n-gram method to calculate its precision (Graham, n.d.).

ROUGE-N: Like BLEU-4, it uses unigram, bigram, trigram and higher order of n-gram

ROUGE-L: Measure scores by using the longest matching sequence. Does not use n-gram method, as it measures by matching sentences rather than word by word.

CIDEr

Consensus-based Image Description Evaluation (CIDEr) evaluates caption based on consensus. It uses two datasets: PASCAL-50S and ABSTRACT-50S. Each of them contains 50 sentences that describes each image. The CIDEr metric uses human judgement of consensus (Vedanta et al., 2014).

7) Attention

Attention is arguably one of the most powerful concepts in the deep learning field nowadays. It is based on a common-sensical intuition that we "attend to" a certain part when processing a large amount of information. To process food image, we also applied this method to make model generate word more specific and accurate. It is a way to enhance a model to choose which part of the image to focus and use to generate sentences. Intuitively, we will decide which word is next with aware of the sequence or the order of the word in the sentence, so does our model. Our model could be smatter as it needs to consider the order of the word and hence focus on certain areas in the image. This is one example from GitHub.

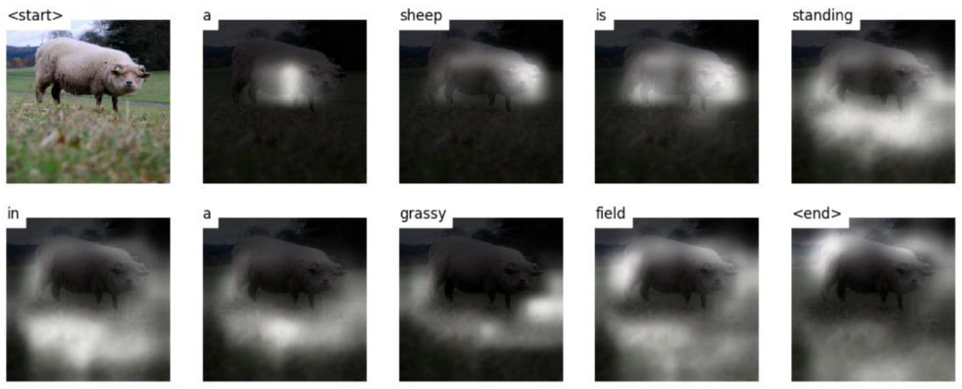


Figure 6: Attention Diagram. Source: (Sgrvined, 2020)

8) Beam Search

We use a linear layer in model to transform the output of decoder layer into a list of scores for each word in the vocabulary. Then normally, model will choose the word with the highest score and combine all of them to generate a sentence. However, I want to make our model consider the words before and after current word. Hence, we brought in Beam search. By this method, a k value is set to help model make decision. If k is equal to 3, model will firstly choose top three words for first place and then choose words in second place with highest score by combining with three words of first place. Model will continue this process until it gets top 3 sentences and finally choose the highest one from it.

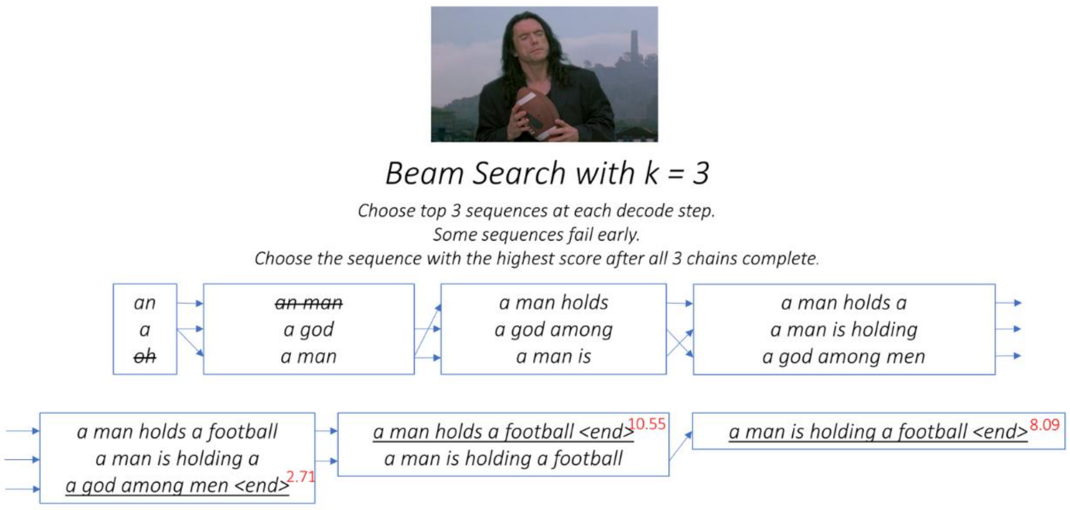

Beam Search with k = 3
Choose top 3 sequences at each decode step.
Some sequences fail early.
Choose the sequence with the highest score after all 3 chains complete.
an
a
~~oh~~
~~an man~~
a god
a man
a man holds
a god among
a man is
a man holds a
a man is holding
a god among men
a man holds a football
a man is holding a
a god among men <end> 2.71
a man holds a football <end> 10.55
a man is holding a football
a man is holding a football <end> 8.09

Figure 7: Beam Search Diagram. Source: (Sgrvined, 2020)

9) Word to Vector

To make machine to understand the meaning of words, we need to transform words into vectors then all relationships of words could be explained by machine in mathematics formulas. This is one example of word to vector map. Based on the complexity of word map, the dimension of vector structure could be increased or decreased. For our model, we use dimension equal to 300 or 512.

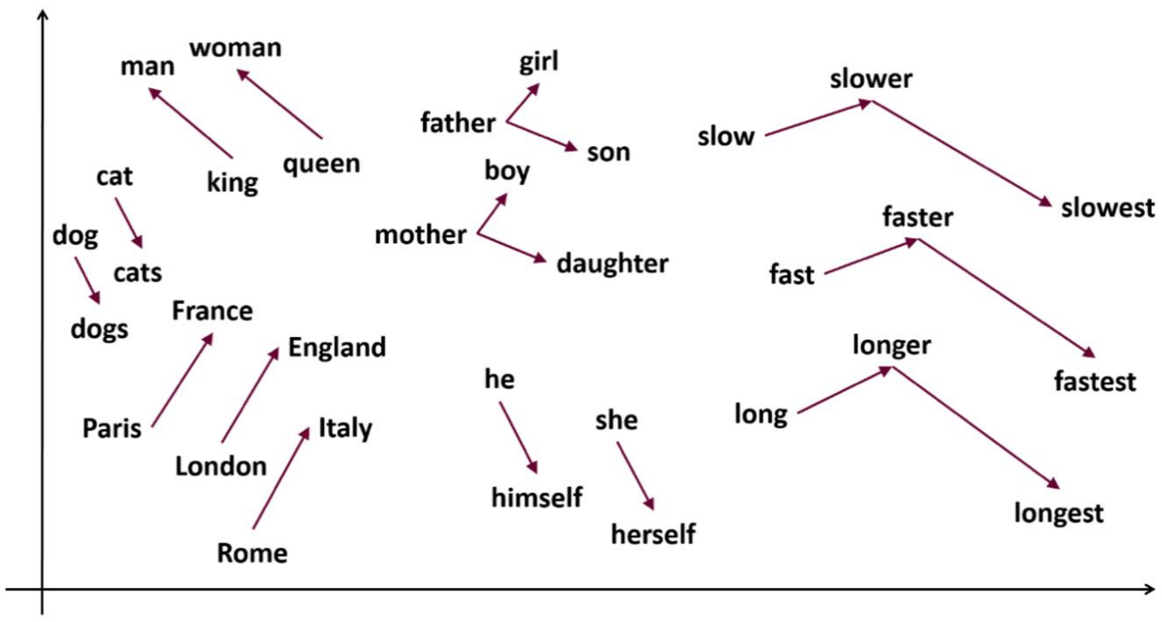


Figure 8: Word to Vector Diagram. Source: (Sgrvined, 2020)

3.2 Overview of models

To tackle our problem of review generation, we looked at different models used for image captioning, as image captioning is a similar problem to ours, having the same type of input (image) and similar outputs (text). The three models that we explored are: Model 1 (basic CNN-RNN), Model 2 (CNN-RNN with attention, pre-trained InceptionV3 encoder), and Model 3 (CNN-RNN with attention, pre-trained ResNet-101 encoder). All these models can be found on GitHub. In this section, a detailed overview of the architecture of each model will be provided.

Model 1:

Encoder-Decoder architecture model without attention mechanism (InceptionV3 + LSTM)

- By Aladdin Persson, available at https://github.com/aladdinpersson/Machine-Learning-Collection.

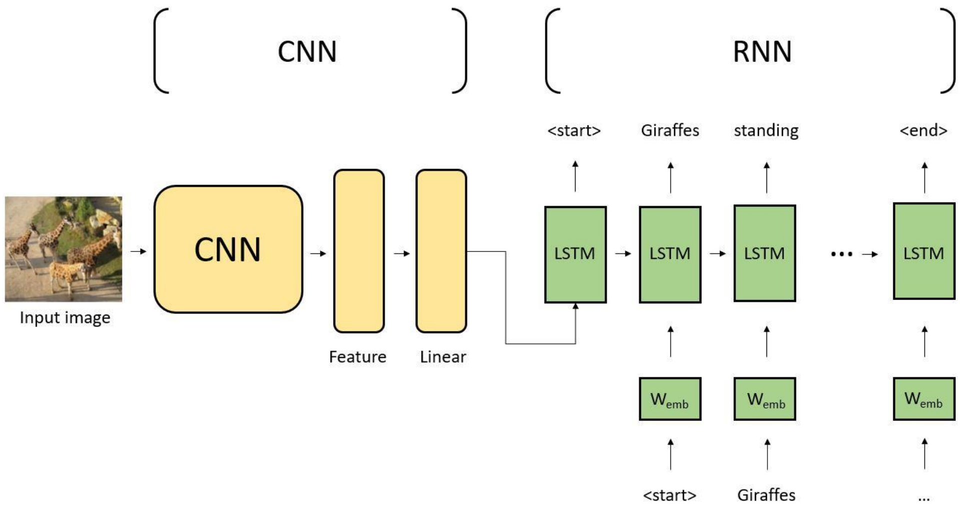


Figure 9: Overview of Model 1. Source: (Aladdin Persson, 2020)

Model 1 utilizes a basic CNN-RNN structure, with a CNN encoder that is pre-trained using InceptionV3 used to extract the vision features from the input image. The extracted vision features are then fed as input into the decoder that is an RNN made up of LSTM cells. The decoder is then responsible to generate sequential text outputs, which are then concatenated to form the desired review/caption. Thus, the input image passes through the model in the following stages: 1) the input image is fed into the encoder and is turned into vision features, 2) the vision features are then fed into the decoder to be used as the initial input, 3) the decoder generates a word at each timestep using the previous time step's output as input for the current time step 4) when the decoder generates an <end> token, the sequence terminates and the outputs from each time step are concatenated to form the desired output, i.e.: review/caption. Model 1 functions as a barebone model, with not much extra features such as attention mechanism or transformers included in its architecture.

Model 2:

Encoder-Decoder architecture model with attention mechanism (InceptionV3 + LSTM)

- By yashk2810, available at https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

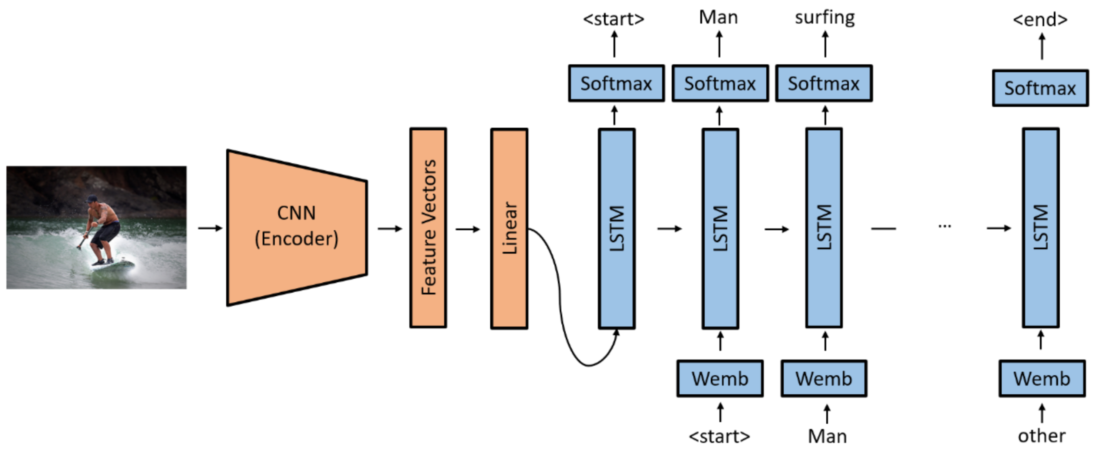
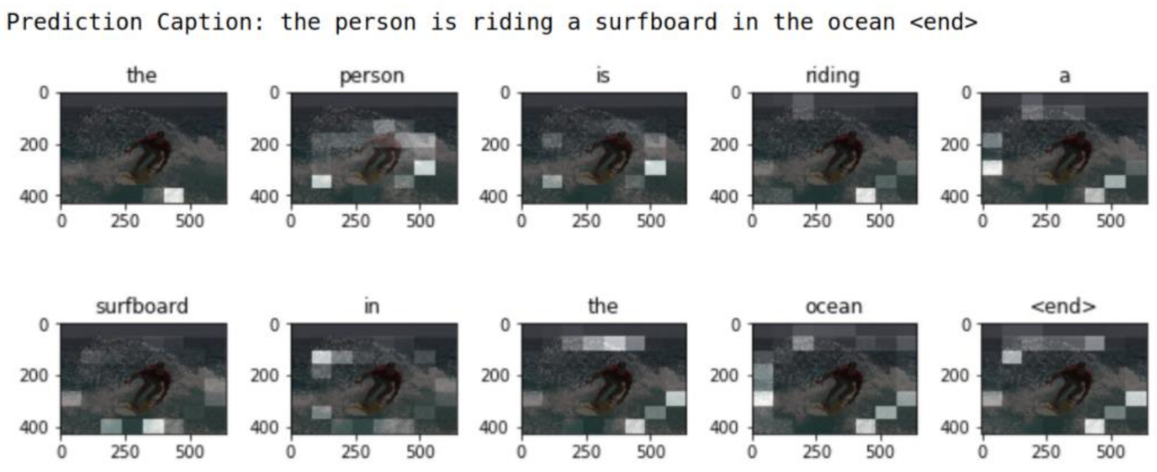


Figure 10: Overview of Model 2. (Source: Faizan Shaikh, 2018)

Model 2 features a similar CNN-RNN structure as Model 1. Like Model 1, it uses a pre-trained InceptionV3 for its encoder. Where it defers from Model 1 is in the addition of an attention mechanism built into the decoder, which allows the decoder to focus on relevant parts of the image while generating the output for each time step.



```
Prediction Caption: the person is riding a surfboard in the ocean <end>
```

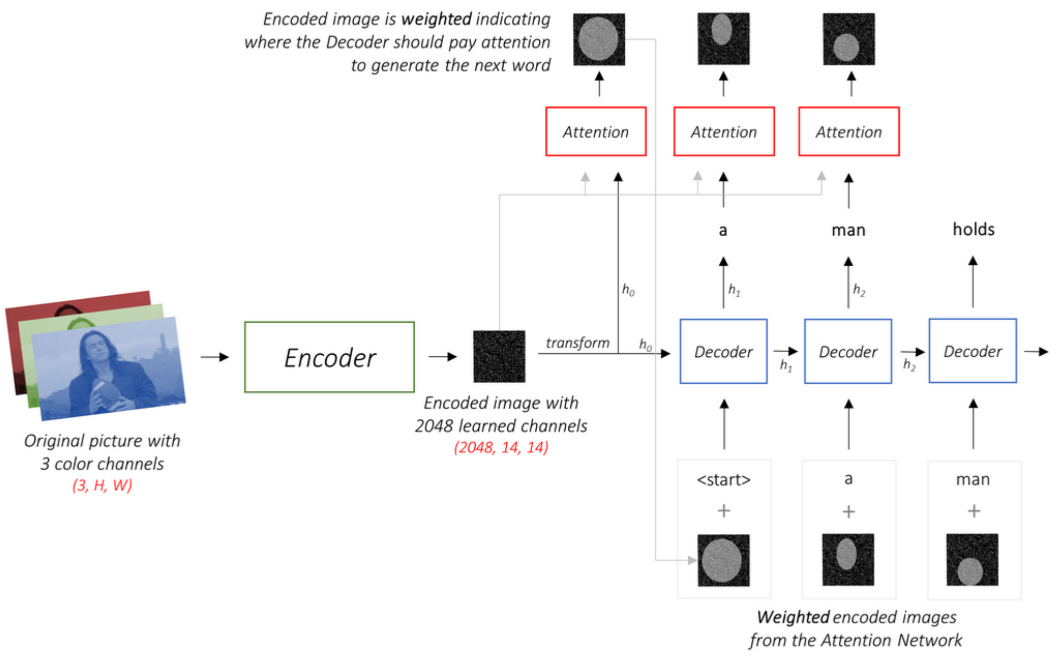
Figure 11: Overview of Model 2 attention. (Source: Faizan Shaikh, 2018)

In this model, it utilises a neural attention mechanism, using a neural network with the ability to focus on a subset of the features extracted from the images. The model learns to attend to specific parts of the image while generating the word describing that part. With an attention mechanism, the image is first divided into n parts, and we compute with a Convolutional Neural Network (CNN) representations of each part h1,…,hn. When the RNN is generating a new word, the attention mechanism is focusing on the relevant part of the image, so the decoder only uses specific parts of the image.

Model 3:

Encoder-Decoder architecture model with attention mechanism (ResNet101 + LSTM)

- By sgrvinod, available at https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning.



Encoded image is weighted indicating where the Decoder should pay attention to generate the next word
Attention
Attention
Attention
a
man
holds
h_0
h_1
h_2
transform
h_0
Original picture with 3 color channels (3, H, W)
Encoder
Encoded image with 2048 learned channels (2048, 14, 14)
Decoder
Decoder
Decoder
h_1
h_2
<start>
a
man
+
Weighted encoded images from the Attention Network

Figure 12: Overview of Model 3. Source: (sgrvinod, 2020)

Model 3 has a similar CNN-RNN structure as the other two models. Similar to Model 2, it also has an attention network built into its decoder so that the decoder is able to know which part of the image it should be paying attention to when generating the next word. However, the encoder used is a pretrained ResNet-101 instead of the InceptionV3 found in the other two models. The decoder consists of LSTM cells, which takes the previous decoder output (or the vision feature generated by the encoder in the case of the first time step) and the attention weighted image features as input to generate its output for each time step.

3.3 Introduction to datasets

Not only the food80k dataset, we also tried to apply our models on general dataset, such as Flickr8k and MSCOCO to check the performance of our models on generating captions.

1. Flickr8k

Flickr8k is a set of images for models to train. It is small in size to allow almost all laptops and desktops to run without any difficulties. There are up to 8000 images in the dataset.

Hence the name, Flickr8k.

There are five captions provided for every image in this dataset as shown below.

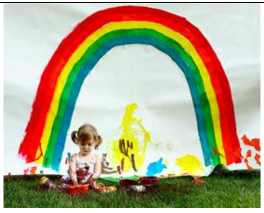
	A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
	A little girl is sitting in front of a large painted rainbow .
	A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
	There is a girl with pigtails sitting in front of a rainbow painting .
	Young girl with pigtails painting outside in the grass .

Table 2: Sample of Flickr8k image and corresponding 5 captions

2. MSCOCO

Microsoft Common Objects in Context (MS COCO) is large image recognition/classification, object detection, segmentation, and captioning dataset. The dataset contains approximately 330K images with more than 220k images annotated. In addition, it contains more than 2M instances in 80 object categories, with 5 captions per image, and 250,000 people with key points.

3. Food80k

Food80k is a set of food images for models to train. It is very big in size so that we need to store all images in Nanyang Technological University Electrical and Electronic Engineering school’s servers. This dataset contains 45810 test images, 87361 train images and 45856 validation images. For each image, there are more than one caption. There are some samples.

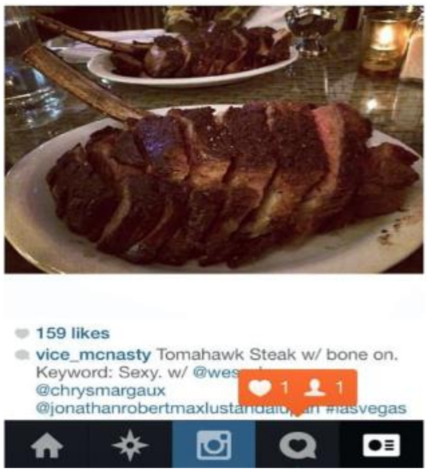
Image	ground truth
	HH is from 3-6 PM and 11-2 AM. Nice to be able to sit down at the bar, get some good food and drink, and watch some basketball at the same time!
	Otherwise I would have loved this dish! I'm still not done with this place and will definitely be back for dinner time to try their other yummy dishes.
	Massive dollops of ricotta and reasonable amounts of meatball made every bite a treat. I didn't even have to add parmesan or red pepper flakes to this one. The thin crust was otherwise similar to the other piece, substantial and chewy and neither soggy or crunchy.
 159 likes vice_mcnasty Tomahawk Steak w/ bone on. Keyword: Sexy. w/ @wes @chrysmargaux @jonathanrobertmaxlustandalo an #lasvegas 1 1	Is it used to attract more yelpers to agree with your review?). The only review she got on her profile and it seems to go against everything I said. Yelper page Created in November of 2013 (the month I wrote the review) Interesting & very Fishy...Ha.

Figure 13: sample of food80k

3.4 Experiments and Evaluation

Model 1: Encoder-Decoder architecture model without attention mechanism (Team 1)

First, we implement the model chosen on Flickr8k dataset, to test the image recognition and text generation capabilities. Then we implement it on the food80k dataset for our project task, where we did some modifications and experiments, finally we come out with the final output of modified model.

A. Output of Original Model

This section shows the quantitative and qualitative results of the original model on Flickr8k and Food80k datasets.

Epoch	**Bleu_1**	**Bleu_2**	**Bleu_3**	**Bleu_4**	**METEOR**	**ROUGE_L**	**CIDEr**
10	**37.07**	**21.23**	**9.74**	**4.39**	**13.24**	**31.53**	**5.17**
20	**43.73**	**21.97**	**9.11**	**4.57**	**11.01**	**32.30**	**6.68**
30	**45.02**	**22.89**	**9.66**	**4.88**	**11.47**	**32.36**	**8.83**
40	**46.46**	**22.76**	**8.25**	**3.38**	**11.11**	**32.12**	**9.10**
50	**45.21**	**21.94**	**7.95**	**3.33**	**11.01**	**31.90**	**8.72**
60	**39.75**	**18.49**	**6.81**	**2.80**	**10.90**	**30.39**	**9.18**

Table 3: Quantitative Result of original model 1 on Flickr8k

Based on the Bleu4 score in the table above, below shows some examples of the images and the predicted captions generated for epoch 30 on flickr8k dataset.

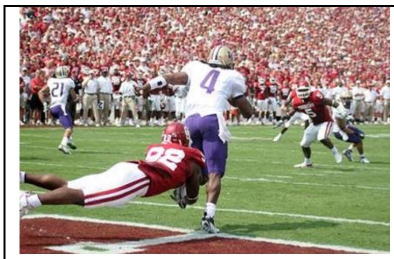
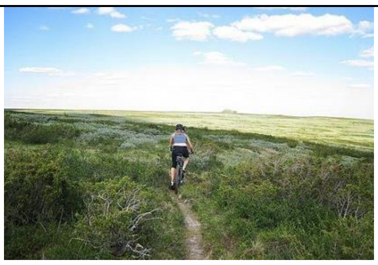
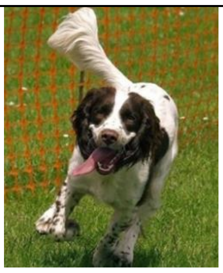
		
a soccer player in a red uniform is running on the field .	a man in a yellow shirt is walking on a rocky path .	a dog is running through a field .

Table 4: Qualitative Result of original model 1 on Flickr8k

Below is a table of the computed scores that tested the original model on Food80k dataset.

Epoch	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	CIDEr
20	32.30	12.76	4.73	1.88	7.59	17.03	0.93
40	38.51	15.71	6.09	2.26	7.84	20.16	0.87
60	25.75	9.49	3.83	1.53	6.03	16.19	0.63
80	34.16	13.04	4.88	1.89	7.23	19.34	0.65
100	34.12	15.07	6.41	2.45	7.33	18.89	0.95

Table 5: Quantitative Result of original model 1 on Food80k

The scores computed for epoch 100 are the best. Hence, we will view the predicted captions for epoch 100.

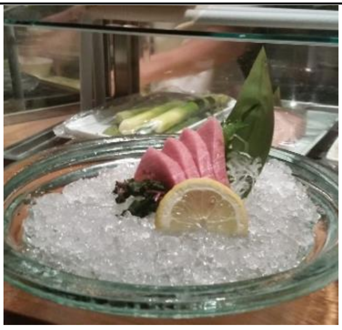
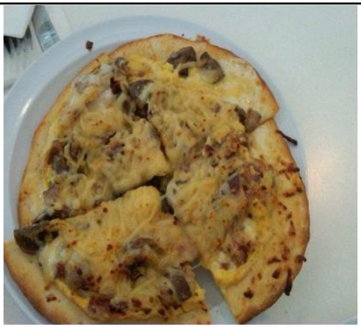
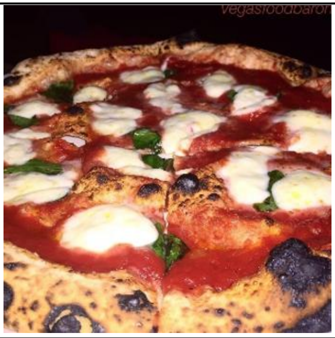
		
i had the chicken and waffles and the chicken and waffles. the waffles were fluffy and the gravy was a little too sweet for me. the waffles were fluffy and the gravy was a little too sweet for me.	the food was good , but the service was great. the food was good , but the service was great.	i 've been here twice now and i 've been to a few times now . i 've been here twice now and i 've been to a few times . i 've been here a few times and i 've never had a bad experience .

Table 6: Qualitative Result of original model 1 on Food80k

As seen from above, the predicted captions do not accurately describe the images.

Comparing between the captions generated for the Flickr8k dataset and the food review dataset, the model generates more accurate captions for Flickr8k. This could be because the food review dataset was more challenging. For the food review dataset, the model had to generate a few sentences for one caption while for the Flickr8k dataset, it only had to generate one sentence for each caption.

B. Modifications made to the model

Some modifications are necessary for the original model, to get the desirable results. The reviews expected should be multiple complex sentences, capable of recognizing different images and detecting specific objects in the images.

Here are the four main modifications we made to the model: application of GloVe pre-trained embeddings, changing the way concatenating information, Disabling ReLU and linear transformation, and finally used another way to extract vision features.

Apply GloVe pre-trained embeddings

Deep learning architectures are unable to process raw text and require numbers as the inputs instead. Word embedding is where each word is mapped to a vector.

Pre-trained word embeddings are embeddings trained on large datasets which can then be used to solve other similar text classification problems. They are often used due to the lack of training data. The dataset needs to have frequently occurring words to build a rich vocabulary. Hence, pre-trained word embeddings are commonly used as most datasets have a large number of rare words.

In order to cover more words in the corpus of food80k dataset, the pre-trained word embedding vectors chosen is Common Crawl, with 840 billion tokens from 2.2 million vocabulary. Each word vector is in the dimension of 300. The downloaded filename is glove.840B.300d.txt.

However, such a large set of word vectors may take more time to load in the training process. Therefore, a pre-processing of the word vectors is necessary, so that the file containing word vectors to load only covers the vocabulary that appears in the food80k dataset. This "word extraction" saves loading time and server space but performs exactly the same as loading the original word vector file.

To approach this extraction of words, the first step is to get a collection of all the words that appears in the food80k dataset.

After the acquisition of a full vocabulary, the corresponding word vectors should then be extracted, and generated as a new file, whose name is glove.vocab.300d.txt. The difference is obvious that the size of the word vector file is reduced from 5.5G to 84MB.

It is important to do so because in the stage of implementation, we met the error of "No space Left on device", as all members in the group are sharing the same server and GPU resource from MLDA. Therefore, monitoring the usage of server space, transferring copies to local computers, and cleaning files not in use, is crucial to keep the whole project on track, without interrupting each other's progress.

With the help of a python library called gensim, the file with the suffix ".word2vec" was generated from the glove.vocab.300d.txt file. This word2vec file could then be loaded and used as a dictionary, with words as keys and pretrained GloVe word vectors as values.

After loading the word2vec file, the next task is to set up a weight matrix, as an input to torch.nn.Embedding, the function used in decoder to get the word embeddings, so that the numerical reviews could be fed into LSTM units.

In the function get_emb, the size of embedding is set to 300, the same as pre-trained GloVe word vectors. Another problem occurred that some words in the food80k dataset did not appear in the GloVe pre-trained word vectors. In this case, the word vector used in the weight matrix is initialised with normal distribution, with samples the same number as the embedding size 300.

Most words in the vocabulary are covered in the pre-trained word collection. For example, with frequency threshold 20, which means only include words that appear more than 20 times in the training reviews, the size of vocabulary chosen is 7164, and the number of words covered in GloVe is 7106, which means 99.2% of word embeddings are pre-trained is this case.

Eventually is to pass the generated weight matrix as a parameter to word embeddings in the decoder. Here, freeze means keeping the weight matrix unchanged through the training, which is not desirable. Therefore, a flag is created in the training file called "fine_tune_embedding", to indicate whether the weights would be updated while training.

After applying the pre-trained word embeddings to the previous model, the result is not improved a lot. Main problems occur again that the model is weak in recognising different images, and the sentences generated are quite general, which seems not related to the picture input fed in. Below are several examples of generated reviews.

However, after evaluation, both the score and the sample review show that the outcome is still not desirable. The possible reason is that although a pre-trained word embedding is applied, the model itself remains unchanged. If the weight of word embeddings leans to the words that are commonly used in daily language, then it could be even more biased.

This result is in correspondence with the result obtained with Flickr8k dataset. When applying the same word embedding to Flickr8k, although there’s not much Flickr8k vocabulary in the word2vec file, the outcome does not seem quite different from the one without word embeddings applied.

Change the way concatenating information

After applying pre-trained word embeddings to the model, the outcome did not improve a lot. Hence, more change has to be made, to try generating better, distinguish reviews for different images.

Therefore, changes are made to the decoder in the way of concatenating information passed to generate the final output. In the forward function of the decoder, there is a parameter called self.decoder which indicates which decoder is selected for training, where 1 is the original decoder and 2 is the modified one.

Considering the performance of these two decoders may both be tested, therefore, the way to pass parameters to decoder, was changed from positional arguments to keywords arguments. This makes the code of class initialisation longer, since it has to read the corresponding parameters from the input dictionary, but it is necessary that different decoder options may require different parameters in the setting up stage. It is also easier to modify the code later so that if any more parameters are used, it can simply add a key-value pair in the dictionary, without major change to the whole model.

Comparing the original decoder, we made two main modifications. One is the information passed to the "head" of LSTM units, the other is the hidden layer information passed to the linear layer to generate the final output.

Firstly, the information passed to LSTM model has been changed. Previously, the LSTM module in decoder takes in the vision features extracted by the pre-trained InveptionV3 as the start, then all the numerical word embeddings are appended. So the image features are transparent to the decoder output, which is formed from the hidden states, the processed outcome of LSTM.
Now in the modified decoder, a dummy zero vector is created to replace the role of vision features in the original model. It is a vector of all zeros, with the same size of pre-trained word embeddings, which is 300. Then these vectors go into LSTM to get the result of the hidden states.

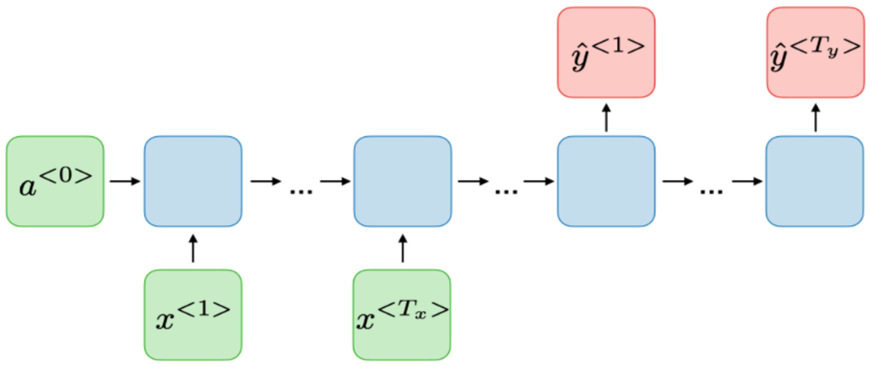


Figure 14: RNN Many-to-Many illustration. Source: (Amidi & Amidi, n.d.)

In this graph illustration, $a^{<0>}$ was the features from encoder, now is replaced by a dummy zero vector. $x^{<1>}$ to $x^{<Tx>}$ are inputs and $y^{<1>}$ to $y^{<Ty>}$ are outputs, which is the hidden layer here.

In order to get more distinguished, image-related review text output, the second main modification is done, to expose more information to the last linear layer in the decoder. Changes are made to the hidden layer, which is the output of LSTM, and the input of the final linear layer of the decoder.

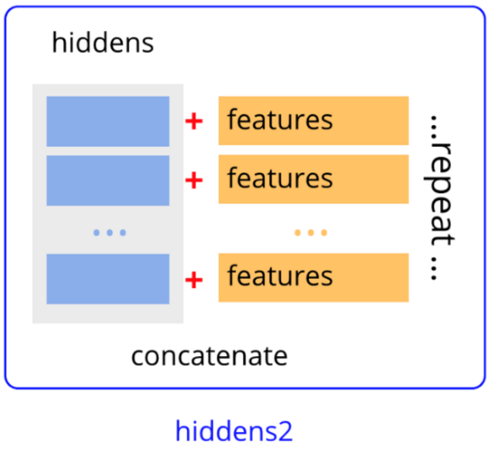


Figure 15: Concatenation illustration. Created by: Wei Haoran

Here we concatenate the vision features extracted by InceptionV3 in the encoder, to the end of the “hiddens”, an output parameter from LSTM, to form a new parameter called “hiddens2”. Then “hiddens2” is passed to the linear layer to get the ouput, instead of “hiddens”

In order to perform the concatenation, the vision feature vector has to repeat k times, where k is equal to the maximum length of sentences in a batch, plus 1, which is the dummy zero vector.

Unfortunately, the outcome is still not ideal. Same problems occurred that the outcome is duplicated, repeating and general.

Disable ReLU and Linear Transformation

After considering the bad outcomes from modification 2, the focus is moved from the decoder to the encoder. There are four main functions working: InceptionV3, linear transformation, ReLU and dropout.

InceptionV3 is the essential pre-trained part for the model to get the vision features and dropout is good in the training process to prevent overfitting. Linear transformation and ReLU are decided to disable in this attempt.

A linear transformation is used in the forward propagation of CNN, as the equation below.

$$Z = W^T . X + b$$

$$Z = \begin{bmatrix} W_{11} & W_{21} & W_{31} & W_{41} \\ W_{12} & W_{22} & W_{32} & W_{42} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Figure 16: Equation of forward propagation in neural networks.
Source: Analytics Vidhya(Singh, 2020)

Where Z is the output of the layer, X is the input vector, W is the weight matrix and b is for bias. Broadcasting is used in matching vector dimensions.

While training, the W and b are updated, through the backward propagation, which is implicitly shown in the code, but in the torch.nn Linear function.

The reason we want to disable such transformation in this step is that InceptionV3 itself is a complex model, with many parameters to update. Linear transform here may make the outcome biased so that it is harder to identify different images for decoder.

ReLU is a commonly used activation function for deep learning, especially in the field of computer vision. However, in our case, when calling a saved model to generate reviews for validating purposes, the output of the encoder shows lots of zeros, even without dropout in testing.

A possible reason for this is that ReLU function takes all negative inputs to zero. If the input vision feature has a lot of negative values, then the output is probably a large portion of zeros. Therefore the ReLU is removed, to keep the negative information in the InceptionV3.

This is also referred to as the "dying ReLU" problem (Lu & Shin, 2019). When the gradient falls to zero, it will never get updated, and some neurons in the network are permanently disabled, which is not preferable in our case.

The figure shows modification in the encoderCNN. Here the Linear transformation and ReLU functions do not operate on the vision features generated out from the InceptionV3 pre-trained model, and only the dropout is used, with a variable indicating the probability of dropout in training, dropout_p.

The decoder is modified to fit the dimensions of new outputs from the encoder. The dimension of "raw" InceptionV3 features is 1000. After concatenating with hidden layers calculated from LSTM with a dummy zero vector as start and pre=trained word embeddings, the dimensions of the finalised hidden layer (hidden2 in the code) is [max length of reviews in a batch + 1, batch size, hidden size + 1000].

After evaluating the model, the problem of over-general reviews and same reviews for different images still occur. Meanwhile, plenty of <UNK> tokens are observed in some epochs. Hence, the <UNK> token is removed before training to avoid this phenomenon.

A strange observation was that the same modified model would work well on Flickr8k dataset but fail to recognise different images for food80k dataset. The scores of the modified model on Flickr8k dataset is even higher than the original scores. Hence, the modifications are proved working, but do not show much improvement in solving our problem.

Another way to extract InceptionV3 features

After revising the model several more times in the order of concatenating, different activation functions used in the encoder, where Leaky ReLU (Sharma, 2019) and PReLU (He et al., 2015) tested, the final focus is the way of extracting vision features.

Suggested by Dr. Nguyen, InceptionV3 pre-trained on ImageNet in TensorFlow Keras library is used for extraction of image features from the last convolutional layer. Applying necessary pre-processing on images, including resizing the image to 299px by 299px and normalization, a dictionary is extracted, with image ids as keys and features as values. The features extracted have a dimension of 2048, which is much more than the previous 1000 in torchvision models.

With the features of images pre-processed and extracted, the data loader now gets information of image features and numerical reviews, instead of images and numerical reviews in the previous model. In the encoder part, no duplicated InceptionV3 in PyTorch, but only the dropout works to prevent overfitting in training. The decoder takes in features with dimension of 2048 and concatenates them to the end of hidden layers from LSTM as before.

C. Result of Modified Model on Food80k

After applying all the four modifications in the previous section, the problems applying the original model on Food80k dataset are solved. Now the modified model is able to differentiate images and also recognize specific food in the test image.

Here are the quantitative scores as the final outcome of our modified model.

Epoch	**Bleu_1**	**Bleu_2**	**Bleu_3**	**Bleu_4**	**METEOR**	**ROUGE_L**	**CIDEr**
10	**21.78**	**7.69**	**3.09**	**1.08**	**5.24**	**16.86**	**0.45**
20	**27.25**	**9.13**	**3.23**	**1.20**	**6.24**	**17.81**	**0.58**
30	**27.34**	**9.34**	**3.59**	**1.10**	**5.06**	**17.45**	**0.51**
40	**32.28**	**12.05**	**4.71**	**1.70**	**6.56**	**18.59**	**0.89**

50	**24.10**	**8.67**	**3.11**	**1.15**	**5.08**	**13.98**	**0.51**
60	**19.31**	**7.24**	**2.62**	**0.95**	**4.71**	**15.07**	**0.29**
70	**29.16**	**11.21**	**4.11**	**1.48**	**6.71**	**17.42**	**0.74**
80	**26.69**	**10.02**	**3.81**	**1.36**	**5.88**	**15.64**	**0.69**

Table 7: Quantitative results of modified model 1

Based on Bleu4 scores, the following generated reviews are extracted from prediction on validation set in epoch 40.

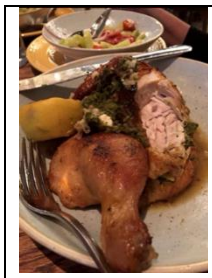
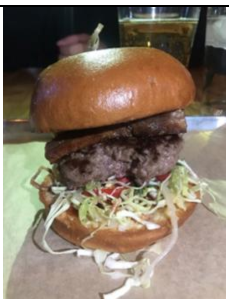
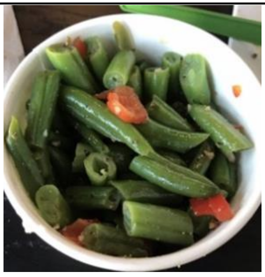
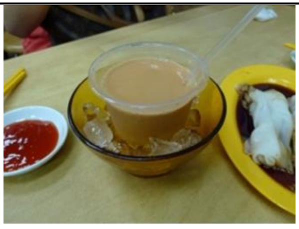
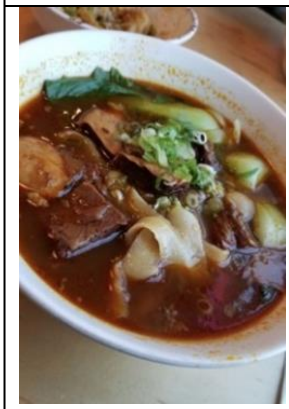
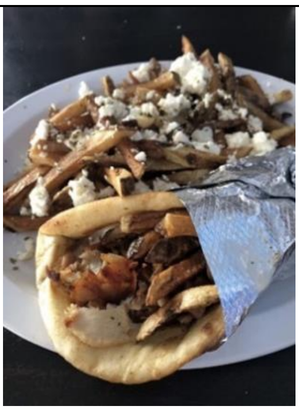
	i had the grilled chicken and i had the lamb chop and i had to say it was a bit dry . i ended up getting the lamb chop and i had to say it was very good . i had the grilled brisket and i had to say		i got the burger and it was so delicious ! i got the burger and it was delicious ! i got the burger and it was so delicious !
	i had the spicy edamame and it was quite spicy . i had the spicy edamame and it was quite spicy . i had the spicy edamame and it was quite spicy .		i got the thai iced tea and i got the thai iced tea . i got the thai iced tea and i got the thai iced tea . i got the thai iced tea and i got the thai iced tea .
	i had the spicy noodles and the spicy noodles . i had the spicy shoyu ramen and it was delicious !		I ordered the carne asada taco and i got the spicy chicken and i got the spicy chicken and it was very good . i ordered the carne asada taco and i got the truffle fries . i got the spicy taco and i got the spicy chicken and

Table 8: Qualitative results of modified model 1

Eventually, this final modification performs much better in terms of qualitative predicted reviews. Firstly, it has the capability to generate different reviews for different images. Secondly, the model can recognise the food in the image. Compared with the previous four results, this is the final version outcome of our model.

The reviews generated have repetitive patterns, which may be a result of the simple decoder, which does not have any attention mechanism or advanced transformer model.

There are still some improvements that can be applied to this model. For example, the predicted reviews have bias on simple sentences like “the food was good”, “the service was good” and the food “chicken and waffles”. Investigation in the dataset and removing some reviews or images to balance the weightages of food and distinct sentences in dataset, may be helpful to avoid such biased result.

Model 2: Encoder-Decoder architecture model with attention mechanism in MSCOCO on Colab (Team 2)

A. General flow of Model

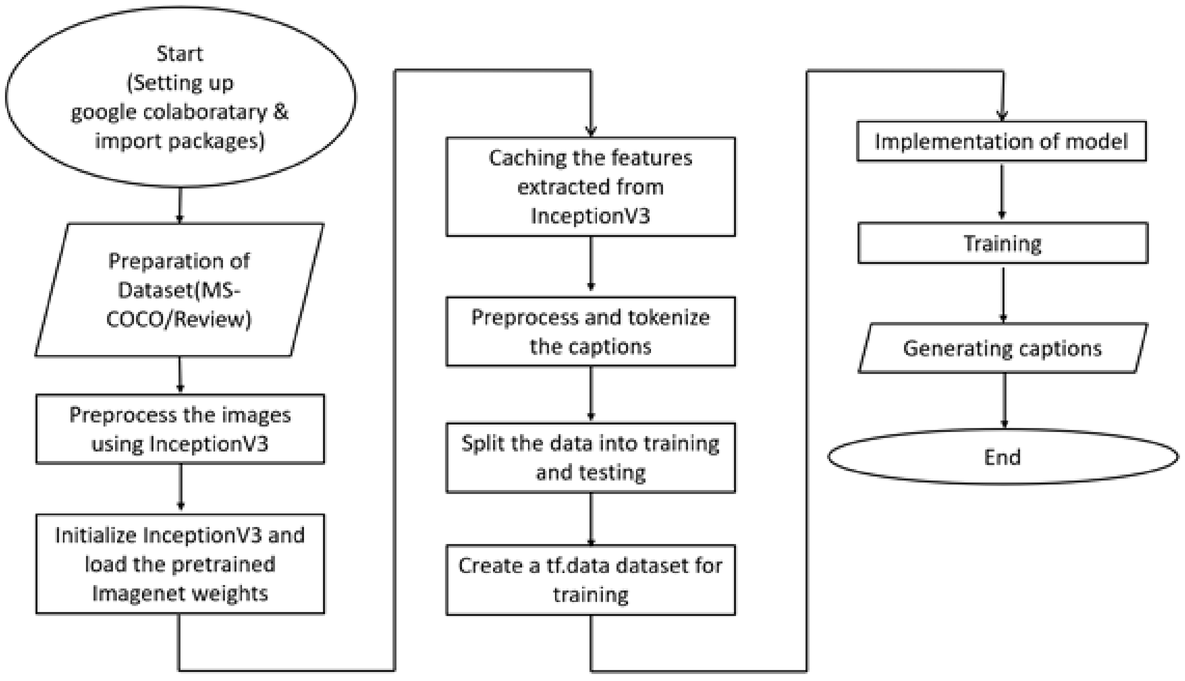
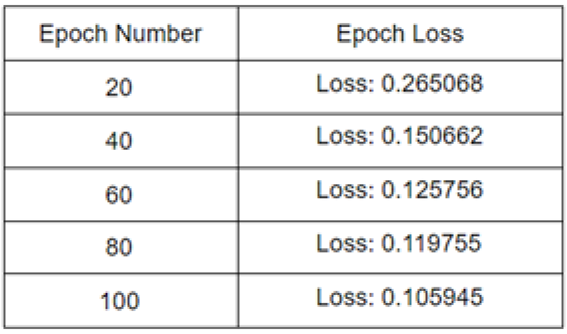
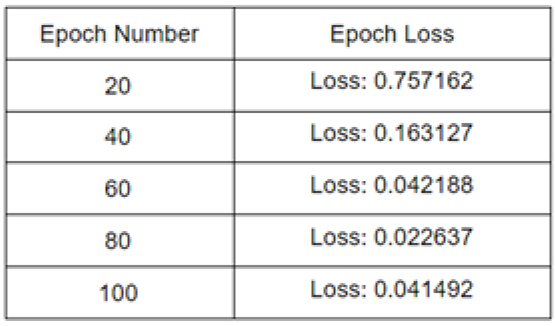
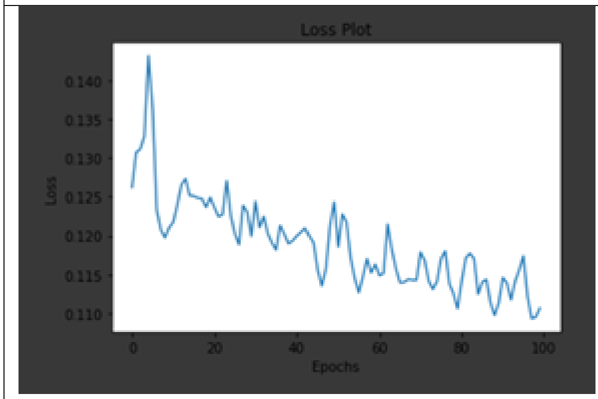
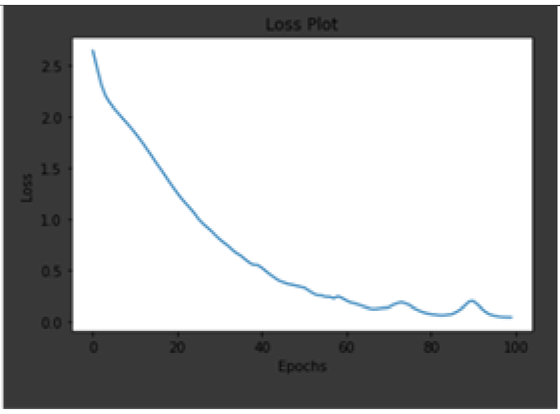

Start (Setting up google colaboratary & import packages)
Preparation of Dataset(MS-COCO/Review)
Preprocess the images using InceptionV3
Initialize InceptionV3 and load the pretrained Imagenet weights
Caching the features extracted from InceptionV3
Preprocess and tokenize the captions
Split the data into training and testing
Create a tf.data dataset for training
Implementation of model
Training
Generating captions
End

Figure 17: General flow of model 2

The figure above shows the general flow of the Image Captioning Model. The model starts with importing the required libraries that are essential in generating the image captions. These libraries can be found in TensorFlow, which is an open source software library which is commonly used in machine learning especially in neural networking models. Following which the dataset is being prepared, for this model we would be training on the MSCOCO & Food80k dataset. Preparation of dataset includes formatting the dataset file in accordance to the required format such that the model is able to read. The model is now able to prepare for its pre-training process, for this model we will be using InceptionV3 to extract the features of the model. Once the features are being extracted, the model then proceeds to tokenize the original captions provided along with each image. By splitting the original captions into individual words, this creates tokens which are then added into the model’s dictionary which will be used later on when predicting the captions for the new images. In every machine learning, a training and testing set is required so that the users will be able to evaluate the quality of the output, thus this model is no different where it splits the datasets(MSCOCO/Food80k) into the training and test dataset. After doing so, the model is now ready to initiate its pre-training by feeding its data obtained from the feature extraction together with the tokenized dictionary into a decoder. The decoder would then produce an output which would be the predicted captions.

B. Comparison of Loss vs Epochs

MS COCO	Food80K
	
	

Epoch Number	Epoch Loss
20	Loss: 0.265068
40	Loss: 0.150662
60	Loss: 0.125756
80	Loss: 0.119755
100	Loss: 0.105945

Epoch Number	Epoch Loss
20	Loss: 0.757162
40	Loss: 0.163127
60	Loss: 0.042188
80	Loss: 0.022637
100	Loss: 0.041492

Table 9: Comparison of Loss vs Epochs on MSCOCO and Food80k

An epoch is defined as the number of passes of the entire training dataset through the neural network. The loss is calculated based off the training and validation, the results are based on how well the model is doing between the training and validation sets. The Loss value determines how poorly or well a model behaves after each iteration of optimization. The lower the loss value, the better the predictions of the model.

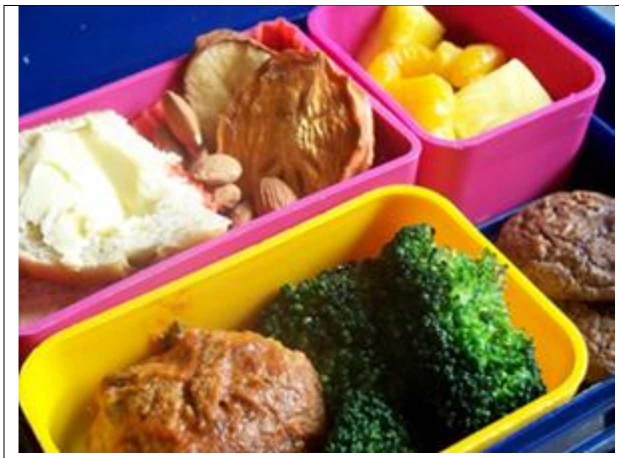
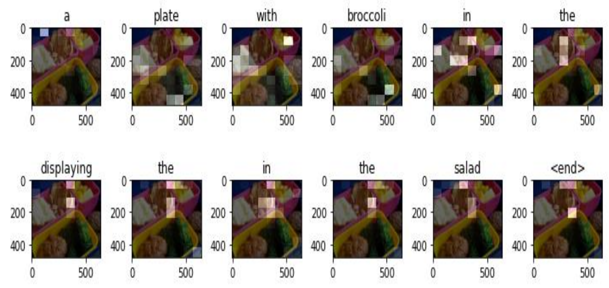
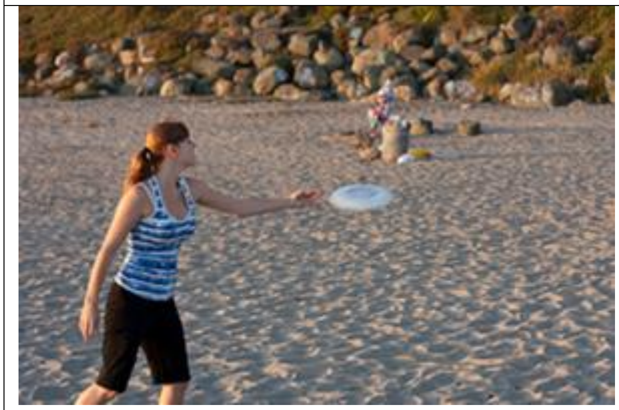
From the graphs above, it is observed that the longer we train our model, the lower the loss becomes thus, the predictions in generating captions are getting better after each epoch.

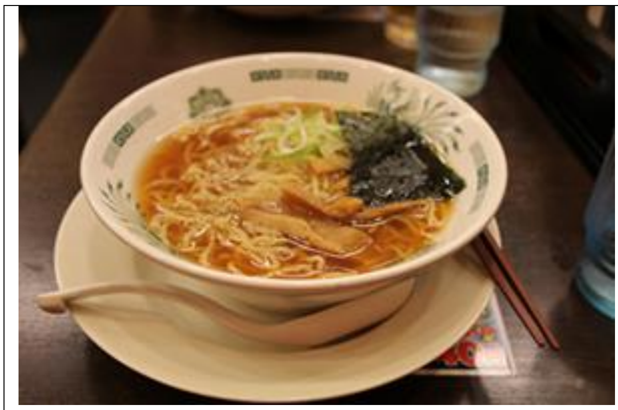
C. Qualitative Results

The table below shows some examples of the reviews generated by MSCOCO dataset and food80k.

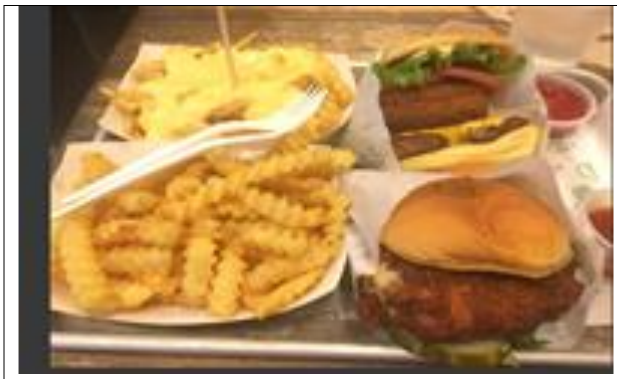
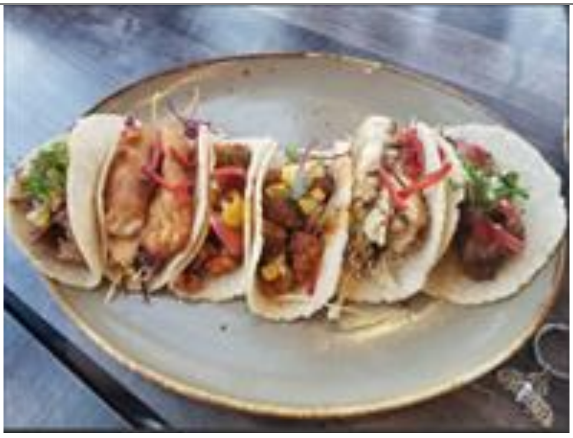
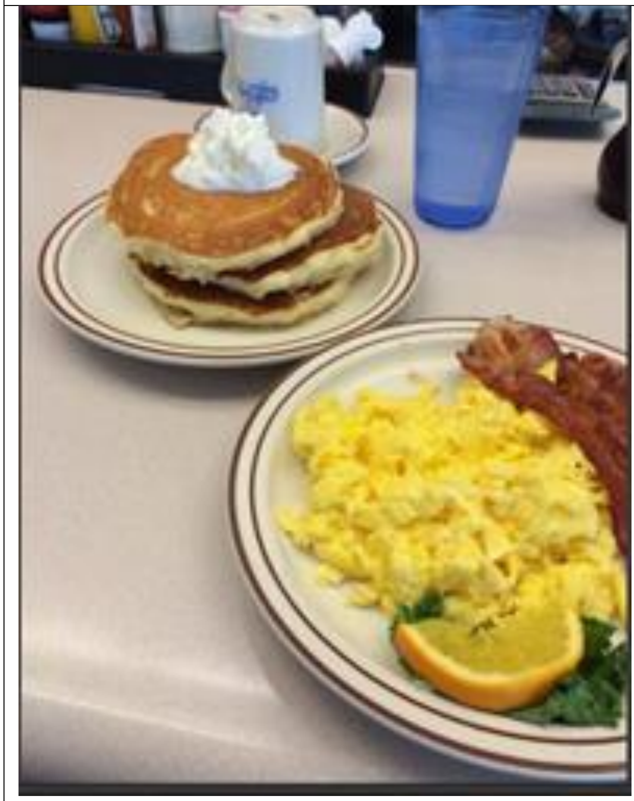
The following table shows the different captions generated by different epoch numbers for the same image. On the left column shows the Image and Original Caption. As for the right column shows the how the Attention Mechanism attend to the image and what word is generated for each attention grid, followed by the 5 generated captions. For the first image, we compare the captions generated for epoch 20,40,60,80,100. For the subsequent 4 examples, we use epoch 100 to generate the predicted captions. This is because epoch 100 has the least loss compare to the other epoch numbers tested.

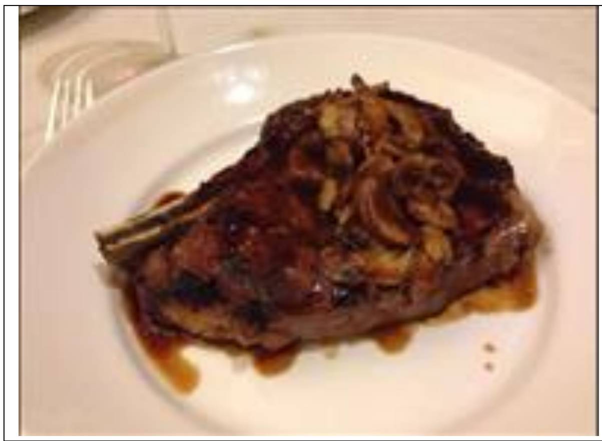
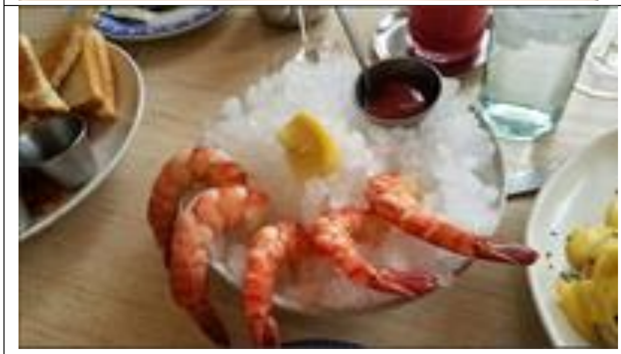
MSCOCO Dataset

  a plate with broccoli in the displaying the in the salad <end>	Original Caption: Colorful dishes holding meat, vegetables, fruit, and bread. Generated Review: A plate with fresh vegetables including vegetables in the sides in a green field. Attention Mechanism: The attention mechanism allows the model to focus and place more “Attention” on the relevant parts of the input sequence as needed. For example, the lighter parts highlighted in the image, giving it more emphasis to the picture and place more attention in identifying a broccoli or salad.
	Original Caption: A young woman enjoying a casual game of Frisbee on a sandy beach. Generated Review: A woman prepares to play frisbee.

	Original Caption: Asian noodle soup, served in a bowl with a spoon and chopsticks Generated Review: A pizza on a plate with food next to eating utensils.
	Original Caption: Several zebras are grazing on a grassy field. Generated Review: A zebra is grazing on grass in the background.
	Original Caption: Two orange buses driving past each other on a street. Generated Review: A bus driving down the street.

Food80k Dataset

	Original Caption: Fries are good on their on - I don't think there was a need for the nacho cheese topping. So glad Shake Shack is on the west coast! ! Generated Review: This is amazing but the eggs scrambled eggs and it was not served up a lot of the chicken taco the only item so on bbq this chicken and it was amazing but the rice it’s not a <unk> couple places in hand and on the best thing we had a crispy golden vegetables my boyfriend is amazing but the beef roll it was amazing but i didn’t have to come back to comeback to be there around.
	Original Caption: i went with a group of co-workers/friends so we ordered a lot of different things. seafood ceviche 3/5 stars mexican street corn (side dish) 5/5 stars guacamole 4/5 stars queso bandito 5/5 stars pulled chicken flautas 4/5 stars chicharron & arugula 5/5 enchiladas (chicken and beef) 4/5 tacos (all of Generated Review: i was very nice and served with all over 10 12 10
	Original Caption: Let's start with some observations ... The fire department folks eat here (good sign). Two busy windows for hot food slamming out of the kitchen (another good sign). Generated Review: in your run out right together

	Original Caption: Read on to find out why! My fiance and I came here for my birthday. And with my friend's recommendation from before I was excited and had high expectations. Generated Review: the french sausage and i enjoyed our taste the food
	Original Caption: the fries were all the space with fries in my needs Generated Review: the fries were all the space with fries in my needs

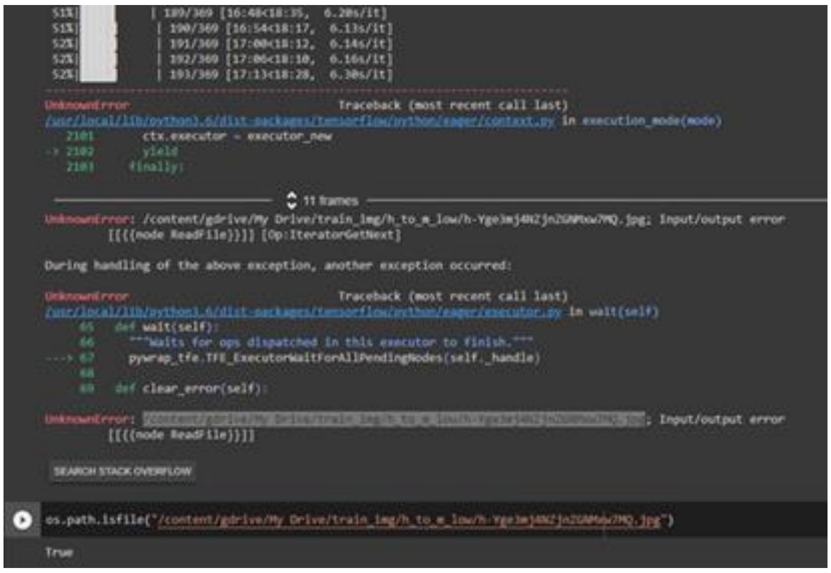
From the results obtained for the MSCOCO and food80k dataset, it can be seen that the captions generated for the MSCOCO dataset was relatively accurate as compared to the captions generated for food80k. The inaccuracy for the captions generated for food80k is possibly due to the dataset being biased to a certain dish causing the model to generate more generation on that particular dish as soon as it deems slightly similar to the biased dish. For example, a piece of meat could be identified as sausage rather than a steak. However, if the dataset contains many pictures of burgers, it will identify burgers correctly. As a result, coming to a conclusion is that, having a wider variety of food images and possible longer training time, we could have yielded better results.

D. Problems encountered

The model encountered multiple timeout error as it was running on Google's Colab. The timeout was mainly caused by the limitation of the ram size of 12.72GB available to each user. Therefore, the model was not able to run large dataset size or high epoch value. The maximum dataset size that was managed was 6000 images and a maximum of 100 epoch per execution.

As the Colab uses Google's Drive for storing the dataset instead of the local drives. Colab was unable to read a single dataset file, therefore the dataset was splitted into 10 smaller subfolders.

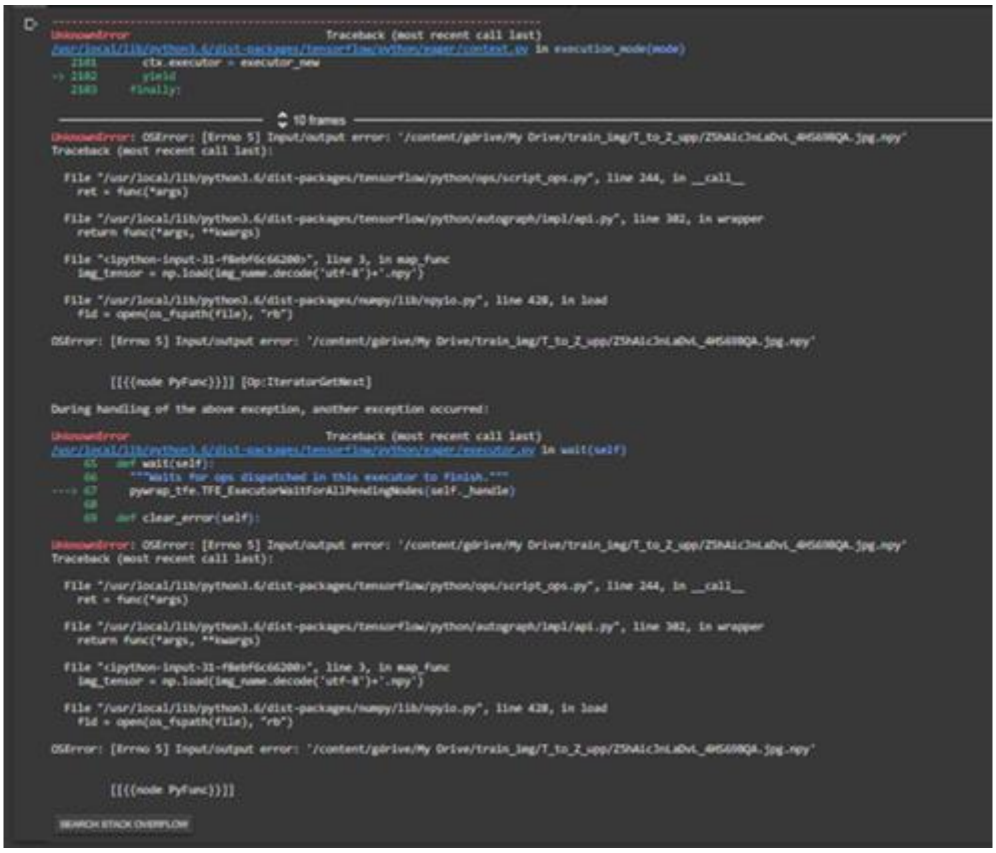
The nature of the given review dataset was more challenging as compared to MS COCO dataset. This is because MS COCO dataset was designed to help train image captioning models. As for the review dataset, there was multiple images that did not have caption that describe the image, rather it was describing the events that was happening which was not constructive in training the model. Each image also had multiple captions tagged to the same image. At random sampling, the dataset can also be found to be biased to a certain dish causing the model to generate more generation on that particular dish as soon as it deems slightly similar to the biased dish.



```
51%|          | 189/369 [16:48<18:35,  6.20s/it]
51%|           | 190/369 [16:54<18:17,  6.13s/it]
52%|           | 191/369 [17:00<18:12,  6.14s/it]
52%|           | 192/369 [17:06<18:10,  6.16s/it]
52%|           | 193/369 [17:13<18:28,  6.30s/it]
---------------------------------------------------------------------------
UnknownError                              Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/context.py in execution_mode(mode)
   2101       ctx.executor = executor_new
-> 2102       yield
   2103     finally:

                              11 frames
UnknownError: /content/gdrive/My Drive/train_img/h_to_m_low/h-Yge3mj4NZjnZGNMxw7MQ.jpg; Input/output error
         [[{{node ReadFile}}]] [Op:IteratorGetNext]

During handling of the above exception, another exception occurred:

UnknownError                              Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/executor.py in wait(self)
     65   def wait(self):
     66     """Waits for ops dispatched in this executor to finish."""
---> 67     pywrap_tfe.TFE_ExecutorWaitForAllPendingNodes(self._handle)
     68 
     69   def clear_error(self):

UnknownError: /content/gdrive/My Drive/train_img/h_to_m_low/h-Yge3mj4NZjnZGNMxw7MQ.jpg; Input/output error
         [[{{node ReadFile}}]]

SEARCH STACK OVERFLOW

os.path.isfile("/content/gdrive/My Drive/train_img/h_to_m_low/h-Yge3mj4NZjnZGNMxw7MQ.jpg")

True
```

Figure 18: Error occurred on Google Colab

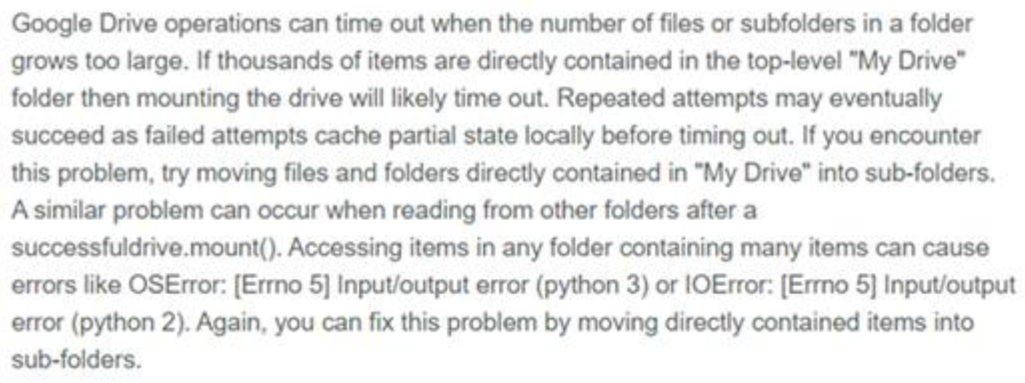
When extracting the features from the images, at times, an input/error will occur which we will need to re-run the code. This posed a challenge as this error occurred several times during a single run.



```
---------------------------------------------------------------------------
UnknownError                              Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/context.py in execution_mode(mode)
   2181       ctx.executor = executor_new
-> 2182       yield
   2183     finally:

                              ⇕ 10 frames
UnknownError: OSError: [Errno 5] Input/output error: '/content/gdrive/My Drive/train_img/T_to_Z_upp/ZShAic3nLaDvL_4HS69BQA.jpg.npy'
Traceback (most recent call last):

  File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/script_ops.py", line 244, in __call__
    ret = func(*args)

  File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/autograph/impl/api.py", line 302, in wrapper
    return func(*args, **kwargs)

  File "<ipython-input-31-f8ebf6c66200>", line 3, in map_func
    img_tensor = np.load(img_name.decode('utf-8')+'.npy')

  File "/usr/local/lib/python3.6/dist-packages/numpy/lib/npyio.py", line 428, in load
    fid = open(os_fspath(file), "rb")

OSError: [Errno 5] Input/output error: '/content/gdrive/My Drive/train_img/T_to_Z_upp/ZShAic3nLaDvL_4HS69BQA.jpg.npy'


         [[{{node PyFunc}}]] [Op:IteratorGetNext]

During handling of the above exception, another exception occurred:

UnknownError                              Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/executor.py in wait(self)
     65   def wait(self):
     66     """Waits for ops dispatched in this executor to finish."""
---> 67     pywrap_tfe.TFE_ExecutorWaitForAllPendingNodes(self._handle)
     68
     69   def clear_error(self):

UnknownError: OSError: [Errno 5] Input/output error: '/content/gdrive/My Drive/train_img/T_to_Z_upp/ZShAic3nLaDvL_4HS69BQA.jpg.npy'
Traceback (most recent call last):

  File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/script_ops.py", line 244, in __call__
    ret = func(*args)

  File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/autograph/impl/api.py", line 302, in wrapper
    return func(*args, **kwargs)

  File "<ipython-input-31-f8ebf6c66200>", line 3, in map_func
    img_tensor = np.load(img_name.decode('utf-8')+'.npy')

  File "/usr/local/lib/python3.6/dist-packages/numpy/lib/npyio.py", line 428, in load
    fid = open(os_fspath(file), "rb")

OSError: [Errno 5] Input/output error: '/content/gdrive/My Drive/train_img/T_to_Z_upp/ZShAic3nLaDvL_4HS69BQA.jpg.npy'


         [[{{node PyFunc}}]]
```

Figure 19: Error occurred on Google Colab

Similarly, an input/output error. This may be due to having a large file size. As directed by the potential problem, we divided the images into subfolders, however the problem persists.



Google Drive operations can time out when the number of files or subfolders in a folder grows too large. If thousands of items are directly contained in the top-level "My Drive" folder then mounting the drive will likely time out. Repeated attempts may eventually succeed as failed attempts cache partial state locally before timing out. If you encounter this problem, try moving files and folders directly contained in "My Drive" into sub-folders. A similar problem can occur when reading from other folders after a successfuldrive.mount(). Accessing items in any folder containing many items can cause errors like OSError: [Errno 5] Input/output error (python 3) or IOError: [Errno 5] Input/output error (python 2). Again, you can fix this problem by moving directly contained items into sub-folders.

The solution we tried is by dividing the images based on their image id into smaller subfolders instead of a large folder by itself.

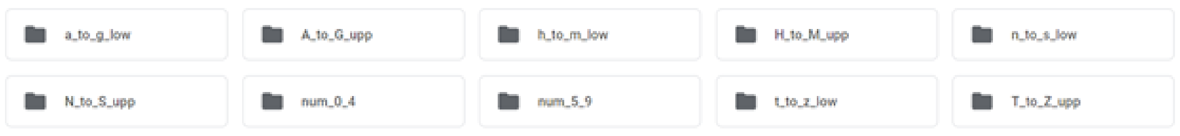


Figure 20: Preview of Subfolders created

Model 3: Apply encoder and decoder framework model with attention in food80k (Team 3)

Evaluation is done to attempt to quantify our model's performance by comparing the set of reviews generated by our model (the "predictions") against the actual reviews contained in the original dataset (the "ground truth") for each images. Thus, we will be considering 6 different metrics generated using different algorithms, namely BLEU1-4, CIDEr, and ROUGE, with BLEU-4 being the main metric used when comparing the performance across different models, as it is a very widely used metric in the field of natural language processing and has also been shown to correlate well with human judgement [1] when it comes to accessing similarity between the prediction set and the ground truth.

For each experiment, the trained models are first evaluated on every 5-10 epochs based on the maximum epoch achieved during training. At this stage, the evaluation is done on the validation set. This data is then used to choose highest scoring epoch to be used as the prediction model for that particular experiment. This chosen model is then evaluated on the test set, and the result obtained can then be used to compare the model with those obtained from different experiments. The chosen model is also the one used to generate reviews that can be manually looked through by humans to get an actual understanding of how it actually performs, as a higher score may not always correlate with better reviews as perceived by humans.

A. Comparison of beam sizes

Beam search is used by the model when generating an output. During each time step, the decoder outputs scores for each word in the vocabulary. Instead of using greedy search, which is to pick the highest scoring word at each time step, beam search allow us to keep track of n number of the highest sentences as the model generates the review, thus allowing us to pick the sequence with the highest overall score only once all n sequences terminate. This number n is referred to as the beam size. Thus, it is clear that beam size will be an important parameter to consider, as having a larger beam size will help us to obtain better reviews as well as have more variety in them. However, having too large of a beam size will affect performance as more memory and calculations need to be done. From our experiments, we have also found that the performance starts to drop past a certain beam size, as shown in Table 10.

Beam size	**Bleu_1**	**Bleu_2**	**Bleu_3**	**Bleu_4**	**ROUGE_L**	**CIDer**
1	**35.20**	**15.40**	**6.70**	**2.80**	**19.10**	**1.20**
3	**39.80**	**17.60**	**7.80**	**3.30**	**20.20**	**1.80**
5	**39.90**	**17.60**	**7.80**	**3.40**	**19.10**	**1.90**
7	**41.30**	**17.50**	**7.10**	**3.00**	**18.10**	**1.90**
10	**40.70**	**17.00**	**6.80**	**2.90**	**17.80**	**1.80**

Table 10: Comparison of performance between different beam sizes

Table 10 shows the evaluation score of the output generated by the same model checkpoint, but with different beam sizes. It is clear to see that the score increases as beam size increase, up to a beam size of 5 after which the performance starts to drop. Based on these results, we have concluded that 5 is the optimal beam size to be used.

B. Comparison between Fine-tuning and freezing image encoder

The image encoder used in our model is a pretrained encoder initialised with weights obtained from ResNet-101. Thus, we have the option of either leaving the image encoder as is by freezing it, or to fine-tune it during the training process. By default, fine-tuning was set to off, meaning the encoder is frozen during training.

	Bleu_1	**Bleu_2**	**Bleu_3**	**Bleu_4**	**ROUGE_L**	**CIDEr**
Fine_tune on	**36.5**	**16.0**	**7.0**	**3.1**	**18.6**	**1.4**
Fine_tune off	**39.9**	**17.6**	**7.8**	**3.4**	**19.1**	**1.9**

Table 11: Comparison of performance with encoder fine-tuning on and off

We found that the model performs better with the encoder frozen, perhaps because the initial weights imported from ResNet-101 were already well-optimised, and any further changes only lead to worse performance.

C. Comparison between different wordmap sizes

The wordmap is the dictionary that maps each word in a reference sentence into a unique number to be fed into the model as input. Thus, the size of the wordmap determines the size of the model's vocabulary. The method to control the size of the wordmap is by setting a threshold for the minimum amount of times a word has to appear in the training set before it is considered as a valid entry in the wordmap. A higher minimum frequency would lead to a smaller wordmap and vice versa. Thus, by limiting the amount of vocabulary the model has to consider during training, hopefully we will be able to train the model to favour more popularly used words used by humans when reviewing, while still maintaining enough word diversity to prevent generated reviews from being dull.

We also experimented with two different methods of tokenizing sentences, the first one being a simple method that utilizes python's built-in .split() method to split apart words separated by spaces, and the second method utilizes the Tokenizer function available in the Natural Language Toolkit (nltk). The main difference between these two methods is that the first method considers a word followed by a punctuation together as one word, while tokenizer splits them into two words (eg: "the food, is good!" into ["the", "food,", "is", "good!"] vs ["the", "food", ",", "is", "good", "!"]).

The table below shows the evaluation score obtained from using the best epoch to generate reviews for the test set, based on different word map sizes as well as different tokenization methods. Since the wordmap size of 8000 paired with the simpler .split() method performed the best in terms of Bleu-4 score, we decided to use this method of generating the wordmap that is used in our final model.

Wordmap size	Bleu_1	Bleu_2	Bleu_3	Bleu_4	ROUGE_L	CIDEr
3000 (split method)	**39.90**	**16.93**	**7.09**	**2.99**	**18.73**	**1.84**
5000 (split method)	**39.64**	**17.20**	**7.29**	**3.09**	**18.75**	**1.92**
8000 (split method)	**39.45**	**17.05**	**7.33**	**3.18**	**18.85**	**1.79**
1000 (nltk tokenizer)	**40.49**	**17.13**	**7.16**	**2.97**	**19.05**	**1.50**
4000 (nltk tokenizer)	**38.99**	**15.84**	**6.32**	**2.62**	**17.34**	**1.43**
5000 (nltk tokenizer)	**35.42**	**15.38**	**6.40**	**2.68**	**16.78**	**1.25**

Table 12: Comparison between different wordmap sizes and tokenization methods.

D. Comparison between pre-trained word-embedding and default values

This experiment is completed based on tokenization mentioned above and GloVe, pre-trained word-embedding. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. As we have word embedding layer in our model, we apply "word to vector" method to transfer words into different vectors in a word map. Therefore, we need to initialize values for our words in word map before we start training. By default way, all these words are initialized with value 0 in word map length times embedded dimension matrix (normally word map length times 300). Then all these 0 will be set specified values while training our model by backward propagation. With GloVe, we could have pre specified values and then reduce the time to reach the expectation. In this experiment, we used word map 5k and 8k, and 1. Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB) and 2. Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB). To prevent overloading the memory of CUDA by importing the GloVe dataset, we filtered out a new GloVe dataset with word map which means we disposed words of GloVe if they do not appear in word map. After all these preparation, we load this new GloVe dataset into embedding layer at the beginning of the whole traning process.

The table below shows the score obtained from using 5k and 8k word map with GloVe.

Dimension = 42B

Wordmap size	Bleu_1	Bleu_2	Bleu_3	Bleu_4
5000 (nltk tokenizer)	**33.6**	**13.8**	**5.7**	**2.4**
8000 (nltk tokenizer)	**35.9**	**15.0**	**6.1**	**2.6**

Dimension = 840B

Wordmap size	Bleu_1	Bleu_2	Bleu_3	Bleu_4
5000 (nltk tokenizer)	**31.4**	**13.2**	**5.4**	**2.3**

8000 (nltk tokenizer)	**34.8**	**14.1**	**5.6**	**2.4**

Table 13: Comparison between different GloVe token sizes

Due to the limitation of time, we cannot choose every model to do comparison then we choose model with GloVe in 5k word map to compare with result in comparison 3.

Wordmap size	**Bleu_1**	**Bleu_2**	**Bleu_3**	**Bleu_4**
5000 (nltk tokenizer with 42B GloVe)	**33.6**	**13.8**	**5.7**	**2.4**
5000 (nltk tokenizer with 840B GloVe)	**31.4**	**13.2**	**5.4**	**2.3**
5000 (nltk tokenizer)	**35.42**	**15.38**	**6.40**	**2.68**

Table 14: Comparison between GloVe and default embeddings, with 5k wordmap size

We figure out that our model performance even poorer then training without GloVe. This result may be caused by situations applied. For GloVe, it is a general situation, but our model is specified for food image only.

E. Quantitative Results

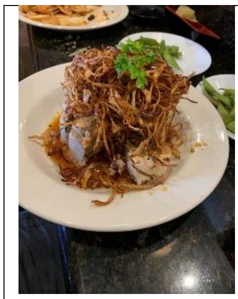
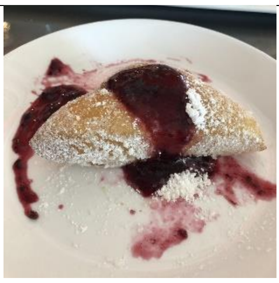
The best settings from each experiment discussed in the previous sections were then combined to train our final model, i.e. the model trained with frozen encoder, wordmap size of 8000 generated with split method, without GloVe embeddings, and a beam size of 5. The score obtained for this model is given as below and is the highest score we have obtained so far.

Bleu_1	**Bleu_2**	**Bleu_3**	**Bleu_4**	**ROUGE_L**	**CIDEr**
39.45	**17.05**	**7.33**	**3.18**	**18.85**	**1.79**

Table 15: Quantitative Results of model 3

F. Qualitative Results

The table below shows some examples of the reviews generated by our final model.

	i ordered the chicken pad thai and it was really good. i had the pad thai and it was the best i've ever had. the service was great and the food was delicious.		it was a little too sweet for me but i think it was a little too sweet for my tastes. i had the chocolate lava cake and it was a little too sweet for my taste. it was a little too sweet.

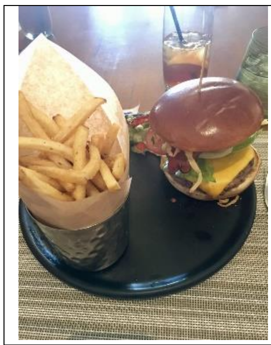
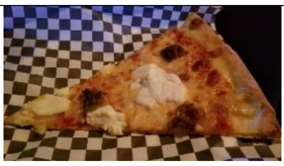
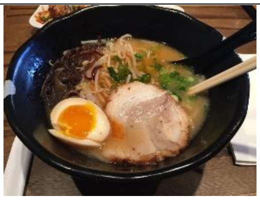
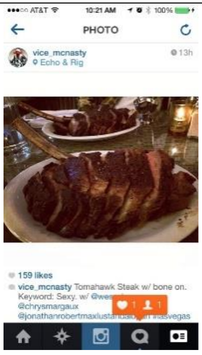
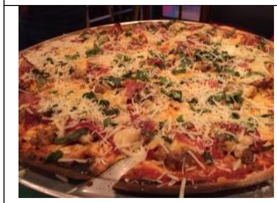
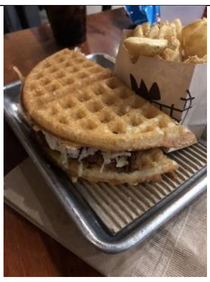
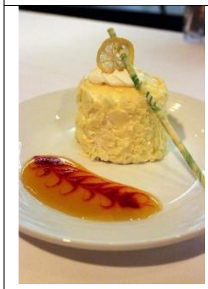
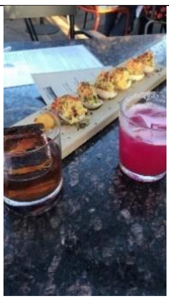
	the burger was good but not the best i've ever had in my life. the meat was tender and juicy and the meat was tender and juicy. the fries were good too.		we ordered the pepperoni pizza and the pepperoni pizza. the pizza was good, but it was a little too salty for my taste. it was a bit bland for me.
	i ordered the tonkotsu ramen and it was really good. the noodles were cooked well and the meat was tender. the broth was rich and creamy.	 PHOTO vice_mcnasty Echo & Rig 159 likes vice_mcnasty Tomahawk Steak w/ bone on.	i had the pleasure of dining at this location and it was my first time here and i was excited to try it out. i had the beef brisket sandwich with a side of rice and it was delicious.
	the crust was very thin and the crust was a little on the dry side. the pizza was good but not the best pizza i've ever had. it was a little too salty for my taste.		i had the chicken fried steak and it was very good. the chicken and waffles were the best i've ever had. it was a little too sweet for my taste
	i had the salmon and my husband had the lobster risotto. i had the filet mignon and it was cooked to perfection. the steak was cooked to perfection.		it was a little too sweet for my liking. i don't know if i would like to go back to try the other items on the menu.

Table 16: Examples of reviews generated by the model

As seen in the table, the model is capable of generating reviews that are mostly grammatically correct, with well-defined sentence structures and clear flow of ideas between sentences. In most cases, it is able to detect the specific food item contained in the image (or at least something close to it). For example, it is able to recognize the burger, pad thai and ramen in the images above. It is also able to generate relevant descriptions to the food it is shown, such as claiming "the broth was rich and creamy" and "the noodle were cooked well" for the image of the ramen. Furthermore, it is able to express its opinions in the reviews, such as claiming that certain foods were the best that it ever had, or that they were too sweet or bland.

However, there are also times when the model fails to correctly recognize the food item shown in the image, as seen in the fifth image on the left of the table. Sometimes, the review generated also contradicts itself, for example in the fourth image in the right of the table where the model generates a review stating that the chicken and waffles were the best it ever had, but then later contradicting itself by saying that it was too sweet for its taste.

Another aspect that may need more work is the generation of generic reviews that have no connection to the actual image used as input. This is shown by the review generated for the image in the bottom right of the table. While these reviews are completely valid and may even be a good representation of how some humans write their reviews, these types of reviews are not what we are looking for, as they provide little value to the user. We would hope that the model would attempt to identify specific food items for every image, even if they turn out to be incorrect.

4. Schedule

PHASE	Planned Milestone Date	Actual Milestone Date
Initiating Phase	Week2	Week2
Planning Phase	Week2	Week2
Execution Phase	Week3 to Week11	Week3 to Week11
Closing Phase	Week12 to Week13	Week12 to Week13
Project End Date	13 Nov 2020	13 Nov 2020

We strictly follow the schedule and deliver good results. By this project, we fully understand how to design and plan a project, and also we learn much knowledge about machine learning.

5. Cost

We do not have cost in this project.

6. Outcomes

In this project, we explored three models which are mentioned above and each team has independent outputs. Based on what we can see in the evaluation part, the model could describe the food and also provide some human feeling related captions. Although there are still some limitations, due to the limitation of the time, I have no more time to improve it. Hopefully, we could deliver this model to our school and finish our project.

7. Reflection

Team 1

Sim Yu Qian, Claire

From this project, I gained a deeper understanding about deep learning. As it was something new to me, I had to read up on the topic a lot and watch tutorials to understand better. When training the model, I ran into a few problems as I did not have some of the libraries installed. After searching the internet, I learnt how to install the libraries and also made modifications to the code so that it could run.

I also learnt how to run the codes on Google Colab and on the school's server. As I was new to using the school's server, I had to read up on Linux commands and also watch YouTube videos. Thanks to Hongyu who shared with us a tutorial video he did on how to use the server, I managed to navigate the server smoothly and run the codes on it. Hence, this project taught me a lot of new skills in a short amount of time. It also taught me teamwork where I got to interact and work with my groupmate, Haoran. I am thankful for my groupmates as they created a supportive environment where everyone is willing to share their knowledge with each other. If any of us were to face difficulties when doing the tasks individually, we would then discuss and try to solve them together. Overall, I learnt to be a more independent learner and how to work in a team.

Wei Haoran

This DIP project is the most challenging project that I have ever done, but also the most beneficial one. As it is the first time I touched deep learning, I researched on the structures and mechanisms of CNN and RNN model a lot. Apart from this project, I also went through some supplementary online courses from Coursera and Kaggle, which provided me with a clearer overview of a deep learning project. Fortunately, when the project is completed, I even grow more interest in deep learning and data processing.

Teamwork in this project maximizes efficiency and encourages self-learning. All of six members of our group are divided in three subgroups, and each group works in parallel, which encourages everyone to engage and learn, and avoid free riders. In our subgroup, not only professional, but also good personal friendships are established between Claire and me. Every week after meeting and reporting, we would start a subgroup meeting to discuss what to do this week, regarding the current process and suggestions from Dr Nguyen.

My programming skills have been improved a lot by this project. Firstly, My Python programming skills were largely improved, and I gained valuable experience on object-oriented programming, and learnt how to handle many files with different functions. Secondly, our subgroup worked on various platforms and environments: Windows 10 OS, Google Colaboratory and Google Drive, Linux virtual machines. I also learnt how to use MLDA GPU resources from Chen Hongyu, which is very helpful in deep learning project with a lot of calculation works. Thirdly, operating on a shared server helped me know how to avoid collision with others, such as clearing the server space usually, and utilizing different GPU resources to ensure everyone's memory usage.

In the beginning stage, we were quite fast in obtaining results from the model we chose on Flickr8k dataset for text generation test. Right after moving on our target dataset, food reviews, we met problem of generating the same reviews from different input images. However, none of our subgroup members gave up, and we tried a lot of modifications to figure out what was wrong with the model. Special thanks to Dr Nguyen, who is really patient and helpful, when our subgroup stuck in the problem of generating the same reviews for different images, he took his time to go through our codes and helped us out of trouble. Finally, we successfully worked out desirable outcomes with Dr Nguyen's suggestions and resources.

If I have the opportunity to work on another deep learning project next time, I would spend more time in the primary stage, to investigate in the characteristics of the model, and create more functions in favor of following stages. Additionally, I would try some version control tools, such as GitHub, to avoid overwriting and improve in collaboration with other group members.

Team 2

Kelvin Ng Hon Seng

Having gone through this design innovation project, it provided me with the opportunity to have a better understanding towards machine learning, specifically image review generation. Writing reviews can be demanding and time consuming, thus there was a need to automatically generate reviews, in our case, generating reviews on food. As a picture is worth a thousand words, there are many ways for the machine to generate reviews, thus our team decided to explore three different models to compare which model yielded the best results.

To start, we divided ourselves into three subgroups tackling on a model. We explored different models using the flicker and MSCOCO dataset. Subsequently, we explored and learned how to tune and adjust the model to fit in our given dataset on food. Out of the many programming software available, I chose to use Google Colaboratory as it allows users to write and execute codes without the need to install packages thus, making it hassle free and user friendly. This was a good alternative for those that want to start on machine learning and data analysis like myself. I have learnt the different types of neural networks such as convolutional neural networks (CNN) and recurrent neural networks (RNN) which was implemented in this model. These two neural networks are essentially the brain behind generating reviews for image captioning.

Throughout this course, the learning journey was not an easy as I was new to the concept of machine learning. Overall, coupled with time constraints and complex terminologies and content of machine learning, I found it challenging to understand and execute the project well. This resulted me in sourcing for my own learning materials from the web and my peers to keep up with the rest of my team members. This was not always sufficient as the learning curve was steep and integrating the review dataset on the model required some adjustments to the code which prompted some errors and resulted in me finding solutions to those errors where I needed to understand more in depth about certain concepts and how these changes will affect the model and overall program.

However, my challenges were mitigated by my supervisors and team members. They were very encouraging and helpful throughout the entire course of the project. They provided me with feedback and solutions on how to tackle the errors in my code as well as guided me on how certain syntax and codes had to be written.

In future, if I were to face with a similar project or problems, I would consult my team leader or supervisor more often to clarify any doubts so as to prevent myself from lagging behind. Since it is a group project, everyone's contribution is equally important in making this project a success.

Yuuki Ikeda

After this Design Innovation Project, I was able to better understand the capabilities and the benefits that image captioning has and will provide to the society.

Before the model is able to generate a caption, multiple steps had to be implemented to train the model. These steps played an important role, as how the model is trained could affect the quality of the produced caption. In the first step, dataset preparation, I noticed that there were multiple images in the dataset that had captions which was not constructive to the training of the model as the caption was not describing the food nor describing the service quality that was received. This might have caused the model to output irrelevant description.

Our model is based on the current Image Captioning models, the Image Captioning models are able to provide description of subjects in the image and in more advanced models the models are able to describe the actions that these subjects are doing. Using the Image Captioning model the group tried to explore ways to improve the generated captions and on top of that, went a step further to generate captions that are able to describe the experience that diners had at the restaurants.

Task allocation to the team was great, the group was split into three teams which allowed us to have an ample amount of opportunities to explore and work on problems that I had with my model. As I am not as fluent in the python programming language, my teammates were very understanding and patient by providing general guidelines and pointers on how I can improve on my model.

Given more time, I think we could work on improving the grammar structure of our model. This is so that it will generate more human-like sentences.

In conclusion, this project has given me the opportunity to improve my knowledge on machine learning. I am able to better understand the machine perspective on how they see each image using the encoder to generate features.

Team 3

Chen Hongyu

The DIP project gives me an opportunity to run a project as a group leader which makes me learn more than the knowledge included in a project only. At the very beginning, I am very confused with the purpose of the project. I was not sure we are going to develop a completed and perfect model or making sure everyone could catch up with each other because each teammate has different coding background. With Mr. Nguyen's suggestion, we were divided into three different teams and different teams has different goals per week. This management nicely resolve our problem. By this way, we could deliver good model at the end and make sure everyone could learn as much as they can. Thankfully, all of my teammates are motivated and responsible which helps us follow our schedule. Along the way, I am learning to be more responsible and helpful to my teammates. Thank support from my teammates and also Dr. Nguyen. Back to the project itself, this is my first project related to Natural Language Processing. It is a wonderful experience to learn about Recurrent Neural Network, many related mechanisms and also these special evaluation tools such as bleu and CIDEr. With this experience, I believe that I could do better in FYP and projects of my career life.

Neo Zhen Ting

Throughout this DIP project, I was able to learn a lot about machine learning algorithms and technique, such as how RNNs and CNNs work, especially in the field of computer vision and natural language processing. More importantly, I was also able to put some of the knowledge that I learned into practice to design a model that is able to suit our specific needs (review generation). Furthermore, I also learned a lot about the hardships involved in a machine learning project, including the collection and cleaning of datasets, the training of models which takes time and GPU resources, tuning of parameters, model evaluation, selection of models, and so on. This gave me a much deeper appreciation of the machine learning/artificial intelligence field, along with what it is capable of doing, and also what some of its limitations are. Thus, I feel incredibly grateful to be able to work on this project as I was able to get a brief glimpse into what it would be like working in the field of machine learning. Regardless of whether or not I choose to continue to pursue this path in the future, I believe that the experiences I obtained from this project would be immensely valuable, as it has also taught me a lot about working in a team, work ethics, as well as problem solving.

8. References

Aladdin, P. (2020, July 13). Pytorch Image Captioning Tutorial. *YouTube.* https://www.youtube.com/watch?v=y2BaTt1fxJU&feature=youtu.be

A. Maheshwari, “Image Captioning using Keras (in Python),” *OpenGenus IQ: Learn Computer Science*, 27-Nov-2019. [Online]. Available: https://iq.opengenus.org/image-captioning-using-keras/.

Baig, M. M. A., Shah, M. I., Wajahat, M. A., Zafar, N., & Arif, O. (2018, December). Image caption generator with novel object injection. In 2018 Digital Image Computing: Techniques and Applications (DICTA) (pp. 1-8). *IEEE.*

Bossard, L., Guillaumin, M., & Van Gool, L. (2014, September). Food-101–mining discriminative components with random forests. In European conference on computer vision(pp. 446-461). *Springer, Cham.*

Chang, H. H., & Meyerhoefer, C. D. (2020). COVID-19 and the Demand for Online Food Shopping Services: Empirical Evidence from Taiwan. *American Journal of Agricultural Economics*.

Faizan ShaikhFaizan “Automatic Image Captioning Using Deep Learning,” 15-May-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-usng-deep-learning/.

Google. (n.d.). Advanced Guide to Inception v3 on Cloud TPU. *Google Cloud.* https://cloud.google.com/tpu/docs/inception-v3-advanced

Graham, Y. (n.d.). Re-evaluating Automatic Summarization with BLEU and 192 Shades of ROUGE. https://pdfs.semanticscholar.org/333f/98412ff246cd646551b4ca6f4b059dc1ea81.pdf

Hinton, G. E. (2012, July 3). Improving neural networks by preventing co-adaptation of feature detectors. https://arxiv.org/pdf/1207.0580.pdf

IBM. (n.d.). BLEU: a Method for Automatic Evaluation of Machine Translation.

https://www.aclweb.org/anthology/P02-1040.pdf

J. Cohen, “Recurrent Neural Networks (RNNs) in Computer Vision: Image Captioning,” *Medium*, 04-Jun-2020. [Online]. Available: https://heartbeat.fritz.ai/recurrent-neural-networks-rnns-in-computer-vision-image-captioning-ea9d568e0077.

Jyotishp, “Introduction,” *neural*. [Online]. Available: https://neural-captioning.jyotishp.ml/.

Kagaya, H., Aizawa, K., & Ogawa, M. (2014, November). Food detection and recognition using convolutional neural network. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 1085-1088).

Lai, H. H., Lin, Y. C., & Yeh, C. H. (2005). Form design of product image using grey relational analysis and neural network models. Computers & Operations Research, 32(10), 2689-2711.

Lu, L., & Shin, Y. (2019, March 15). Dying ReLU and Initialization: Theory and Numerical Examples. *arxiv.org*. https://arxiv.org/abs/1903.06733

Mudambi, S. M., & Schuff, D. (2010). Research note: What makes a helpful online review? A study of customer reviews on Amazon. com. MIS quarterly, 185-200.

Pai, A. (2020, March 16). An Essential Guide to Pretrained Word Embeddings for NLP Practitioners. https://www.analyticsvidhya.com/blog/2020/03/pretrained-word-embeddings-nlp/

Peng, M. L. (2018, May 7). Improving neural networks by preventing co-adaptation of feature detectors. https://medium.com/@lipeng2/dropout-is-so-important-e517bbe3ffcc

Prabhu. (2018, March 4). Understanding of Convolutional Neural Network (CNN) — Deep Learning. Medium. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Rozin, P., Fischler, C., Imada, S., Sarubin, A., & Wrzesniewski, A. (1999). Attitudes to food and the role of food in life in the USA, Japan, Flemish Belgium and France: Possible implications for the diet–health debate. Appetite, 33(2), 163-180.

S. C. -, “What is Feature Extraction? Feature Extraction in Image Processing,” *GreatLearning*, 03-Nov-2020. [Online]. Available: https://www.mygreatlearning.com/blog/feature-extraction-in-image-processing/.

Srivastava, N. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. 30. https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

Wilcock, A., Pun, M., Khanona, J., & Aung, M. (2004). Consumer attitudes, knowledge and behaviour: a review of food safety issues. Trends in Food Science & Technology, 15(2), 56-66.

Zafra, M. (n.d.). Understanding Convolutions and Pooling in Neural Networks: a simple explanation. *Towards Data Science*. https://towardsdatascience.com/understanding-convolutions-and-pooling-in-neural-networks-a-simple-explanation-885a2d78f211

Appendix

Project Members Information

	Name	Project contributions	Report Contribution
1	Chen Hongyu	Group Leader, treasurer, group report editor	Section 1,2 Section 3.1 Section 3.3.1 Section 3.4 team3 Section 4,5,6,7 (reflection of Chen Hongyu),8,9
2	Neo Zhen Ting	Team 3 Member, DIP competition presenter, project demo	Section 3.2 (pg 12-14) Section 3.4 Team 3 A-C, E,F (pg34-35, pg 37-39)
3	Wei Haoran	Team 1 Leader, Modifications and evaluations on Team 1 model, Group report formatter	Section 3.2 Model 1(Pg.12), Section 3.4 Team 1 (Pg.17-26), Concatenation Illustration (Figure 15), Reflection (Pg.41-42)
4	Sim Yu Qian, Claire	Team 1 Member, Training and validating on Team 1 model	Section 3.1 1-6 (Pg. 5-10), Section 3.3.1(Pg.14-15), Section 3.4 Team1 A (Pg. 17), Reflection (Pg.41)
5	Kelvin Ng Hon Seng	Team 2 Leader, Modifications and Training	Pg.13 (Attention Mechanism) Pg.15 (MSCOCO), Pg.26(Figure17) Pg.27-29 Pg.32-33 Pg 42-43 (Reflection)
6	Yuuki Ikeda	Team 2 Member Model Exploration Implementation of InceptionV3 Error Debugging	Pg.26 (General Flow of Model), Pg.27 (Comparison of Loss Vs Epoch), Pg.30-31 (Qualitative Result, Food80k), Pg.31-33 (Problems Encountered), Pg.43-44 (Reflection)