E012: Design of an Automatic Review System

Presented by: Wei Haoran, Team 1 Leader

4 Stages of this Project

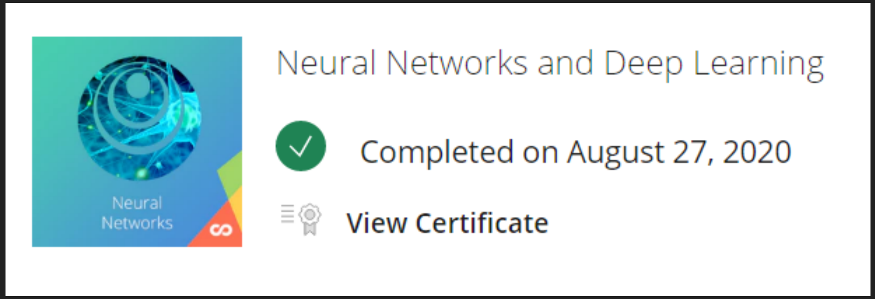
1. Preparation (Wk1-2)
2. Model Testing on Flickr8k (Wk3-5)
3. Applying on Food80k (Wk6-8)
4. Modifications and Outcomes (Wk9-12)

Stage 1: Preparation

1. Research & Study

○ Background Knowledge: CNN, RNN, LSTM, etc.

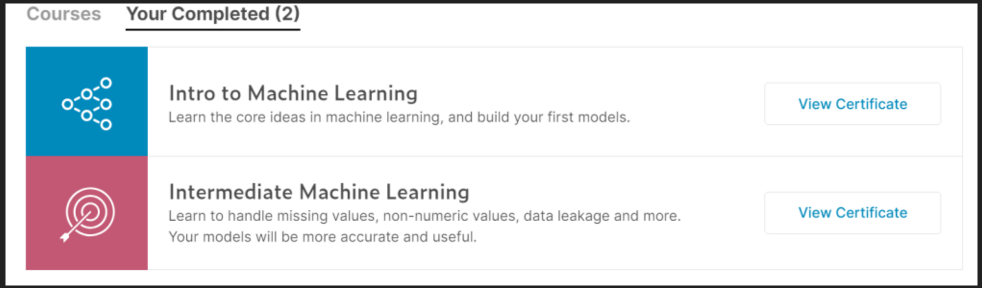
○ Online Courses:



Neural Networks and Deep Learning

✓ Completed on August 27, 2020

View Certificate



Courses   Your Completed (2)

Intro to Machine Learning

Learn the core ideas in machine learning, and build your first models.

View Certificate

Intermediate Machine Learning

Learn to handle missing values, non-numeric values, data leakage and more. Your models will be more accurate and useful.

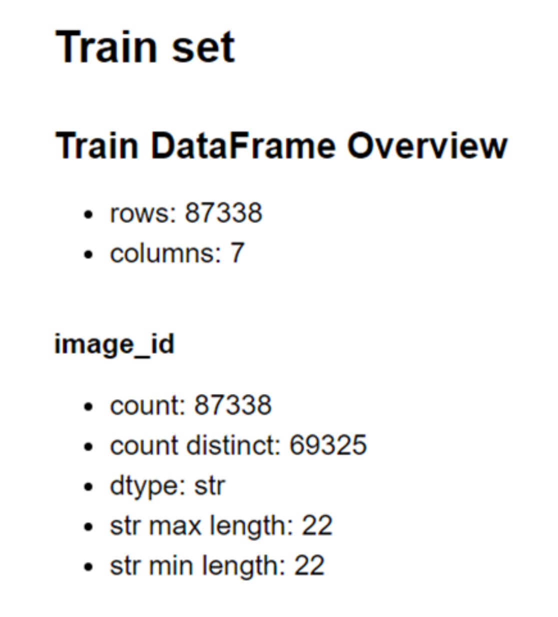
View Certificate

2. Explore Food80k dataset

3. Model Chosen

4. Team 1 members:

Claire, Haoran



Train set

Train DataFrame Overview

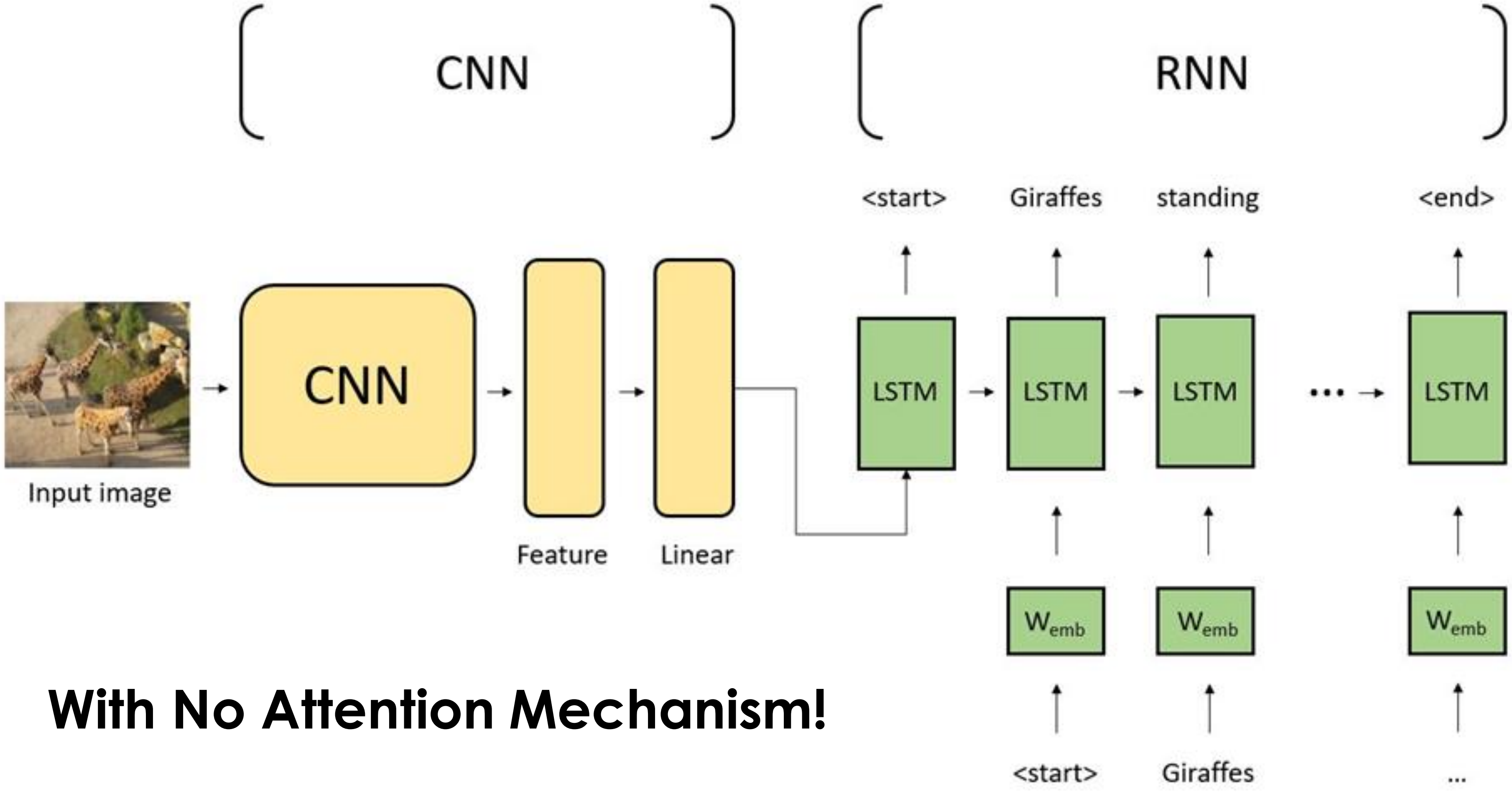
- rows: 87338
- columns: 7

image_id

- count: 87338
- count distinct: 69325
- dtype: str
- str max length: 22
- str min length: 22

Model Chosen

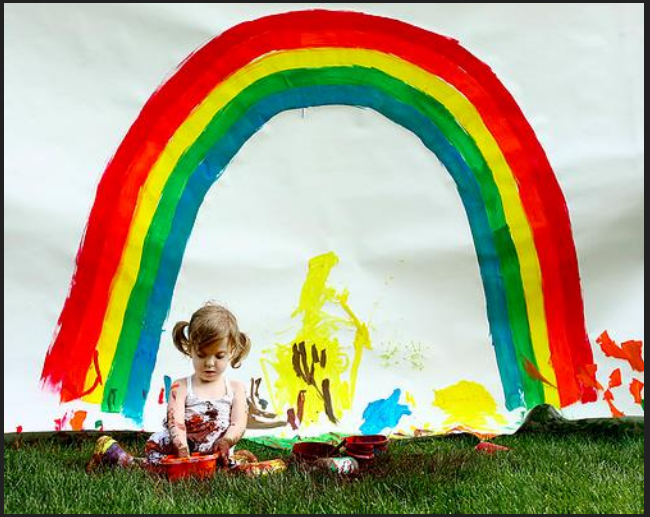
By Aladdin Persson



With No Attention Mechanism!

Stage 2: Model testing on Flickr8k

- Flickr8k Dataset
  - ~8000 images, each image with 5 captions



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 .

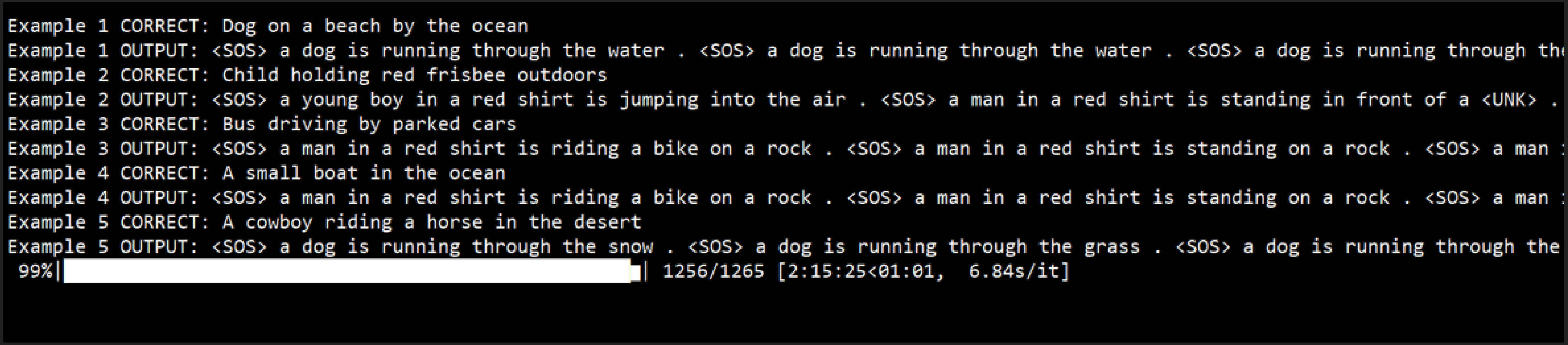
Stage 2: Model testing on Flickr8k

Windows OS =>

Google Drive & Colab

1. Make the model run on Windows10 OS

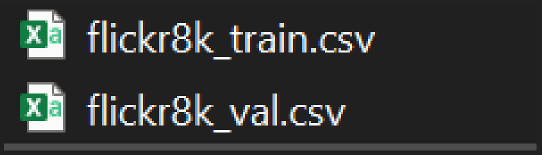
- CPU: time consuming, not device friendly
- Google Colab: easy to share resources, has GPU runtime to accelerate training



```
Example 1 CORRECT: Dog on a beach by the ocean
Example 1 OUTPUT: <SOS> a dog is running through the water . <SOS> a dog is running through the water . <SOS> a dog is running through the
Example 2 CORRECT: Child holding red frisbee outdoors
Example 2 OUTPUT: <SOS> a young boy in a red shirt is jumping into the air . <SOS> a man in a red shirt is standing in front of a <UNK> .
Example 3 CORRECT: Bus driving by parked cars
Example 3 OUTPUT: <SOS> a man in a red shirt is riding a bike on a rock . <SOS> a man in a red shirt is standing on a rock . <SOS> a man
Example 4 CORRECT: A small boat in the ocean
Example 4 OUTPUT: <SOS> a man in a red shirt is riding a bike on a rock . <SOS> a man in a red shirt is standing on a rock . <SOS> a man
Example 5 CORRECT: A cowboy riding a horse in the desert
Example 5 OUTPUT: <SOS> a dog is running through the snow . <SOS> a dog is running through the grass . <SOS> a dog is running through the
 99%|█████████████████████████████████████████▉| 1256/1265 [2:15:25<01:01,  6.84s/it]
```

Stage 2: Model testing on Flickr8k

2. Split Flickr8k Train-Test Set



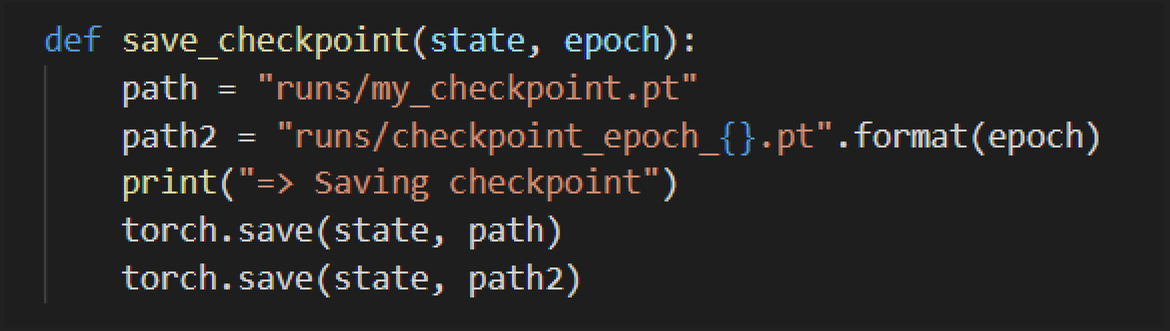
flickr8k_train.csv

flickr8k_val.csv

3. Display **Epoch** while training

- Use While loop instead of For loop
- Current Epoch = int(Current Global Step / Steps per epoch)
- Steps per epoch * batch size = Total images in training

4. Save Every Epoch of Training



```
def save_checkpoint(state, epoch):
    path = "runs/my_checkpoint.pt"
    path2 = "runs/checkpoint_epoch_{}.pt".format(epoch)
    print("=> Saving checkpoint")
    torch.save(state, path)
    torch.save(state, path2)
```

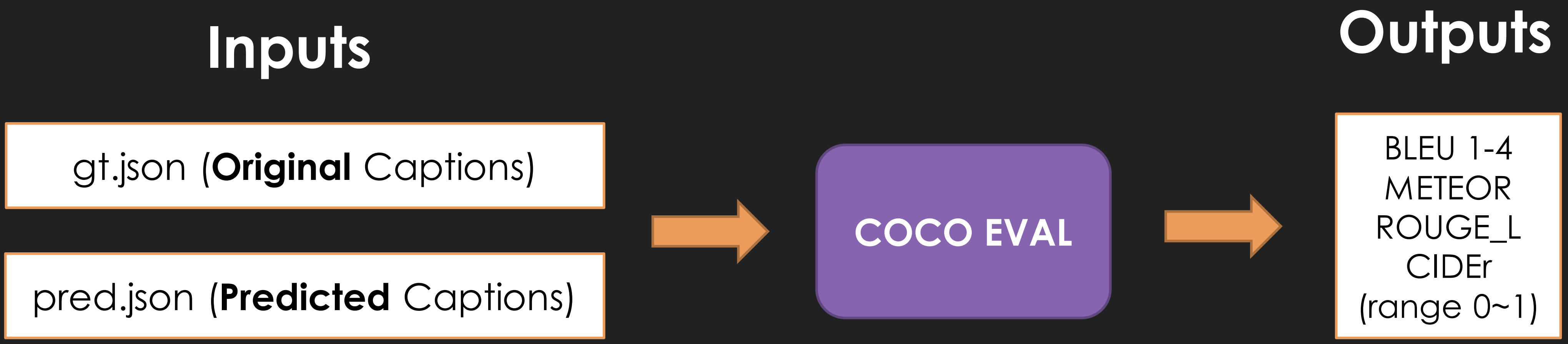
Stage 2: Model testing on Flickr8k

5. COCO Eval:

Quantitative Evaluation for saved prediction result

Inputs: 2 JSON files: ground truth, predicted

Outputs: BLEU1-4, METEOR, ROUGE_L, CIDEr Scores



Inputs

gt.json (**Original** Captions)

pred.json (**Predicted** Captions)

COCO EVAL

Outputs

BLEU 1-4
METEOR
ROUGE_L
CIDEr
(range 0~1)

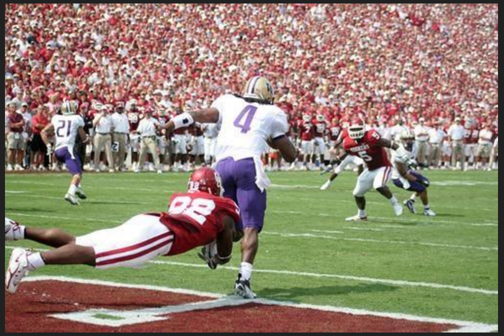
Quantitative Result on Flickr8k

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%

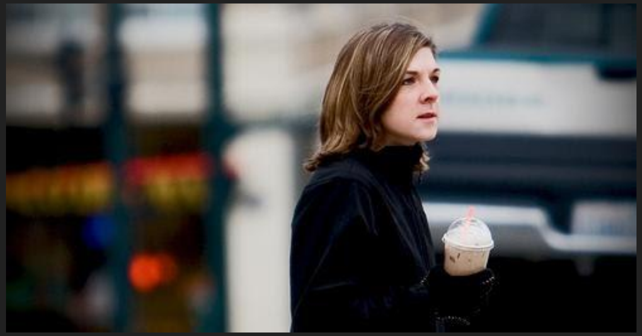
Qualitative Result on Flickr8k

Text Generation ✓

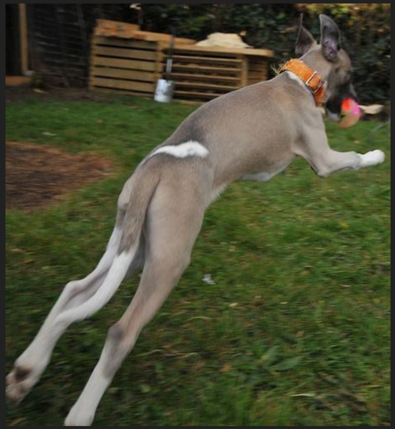
Image Recognition ✓



a soccer player in a red uniform is running on the field .



a woman in a black shirt and jeans is sitting on a bench .



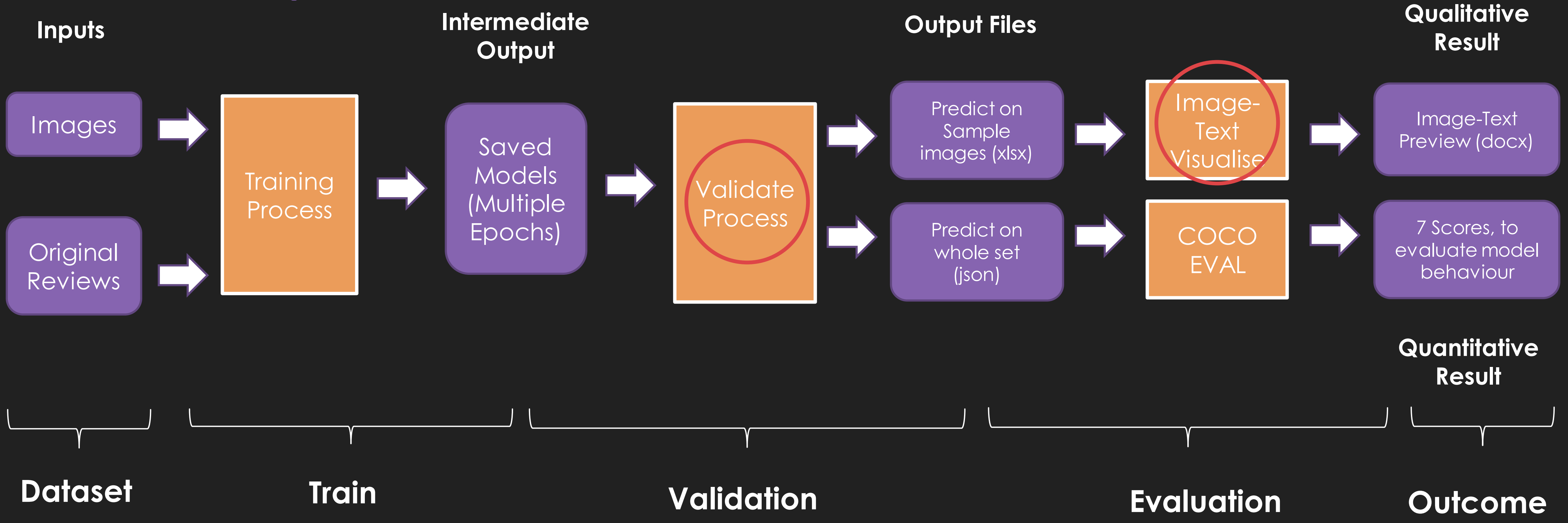
a dog runs through the grass .

Stage 3: Applying on Food80k

My Works:

- Validation codes written (Saved Model to predict on whole val / test dataset)
- Image-Text Preview codes written (make it easier to visualise the result)
- Modify code to save vocabulary (word-index mapping) before training
- Modify code to record training loss per epoch
- Fix some typos in the code
- Count missing images in Food80k downloaded images

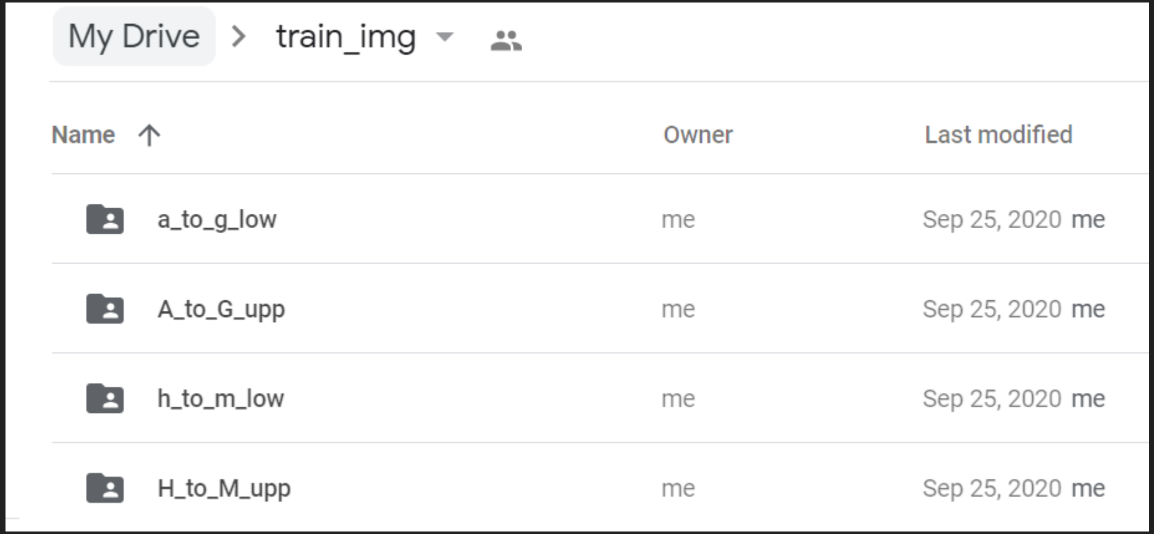
Overview of System



Stage 3: Applying on Food80k

My Works with Google Colab & Google drive:

- Try unzipping images directly on Colab
- Try unzipping images locally and upload all to Google Drive
- Try creating subfolders to handle "Google Drive Timeout"



Name ↑	Owner	Last modified
a_to_g_low	me	Sep 25, 2020 me
A_to_G_upp	me	Sep 25, 2020 me
h_to_m_low	me	Sep 25, 2020 me
H_to_M_upp	me	Sep 25, 2020 me

Google Drive & Colab

=> MLDA server (Linux)

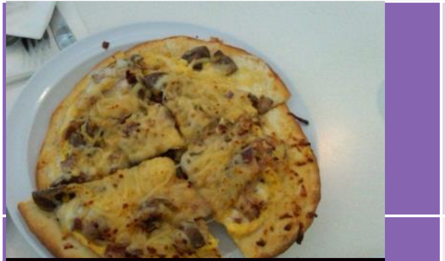
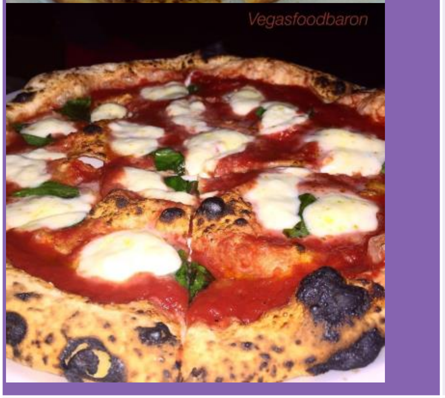
Original Model on Food80k

Text Generation ✓

Image Recognition ✗

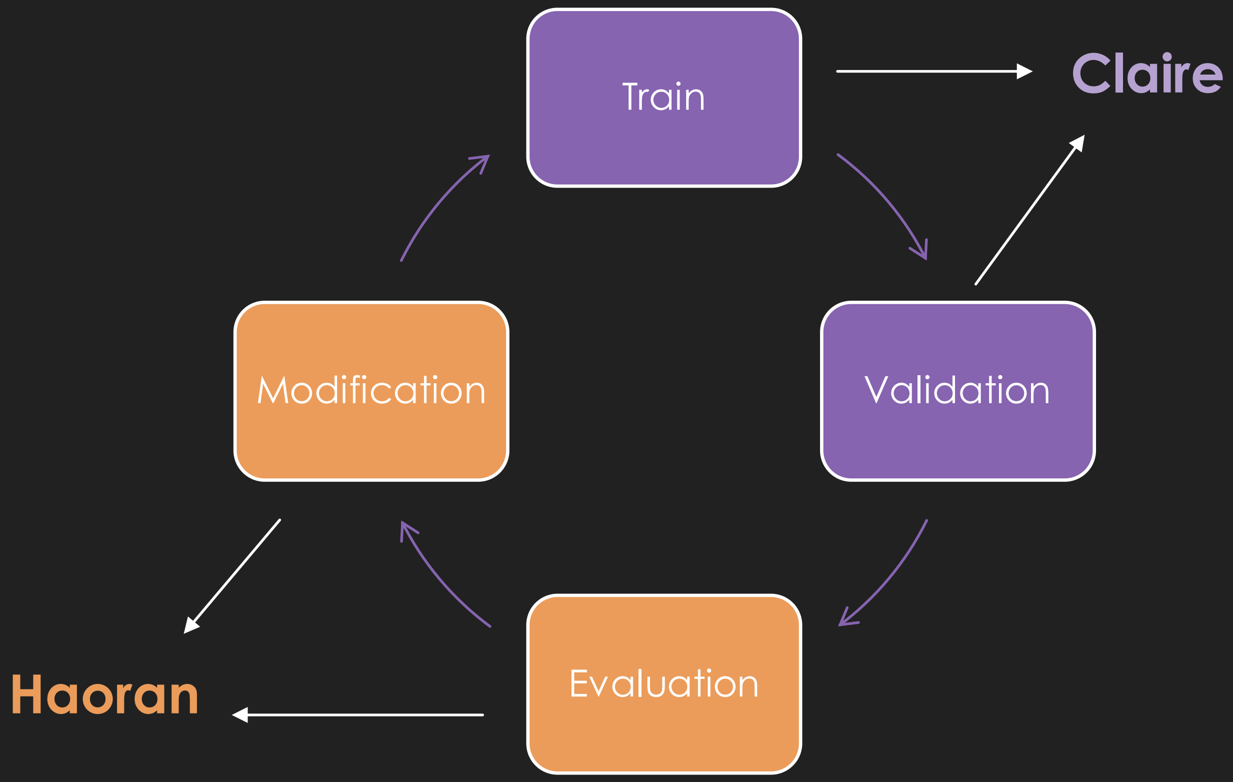
Problems:

1. General Sentences
2. Fail to recognise specific food
3. Same review for different images

	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 .

Modifications!

Stage 4 Modifications: Workflow



Train

Validation

Evaluation

Modification

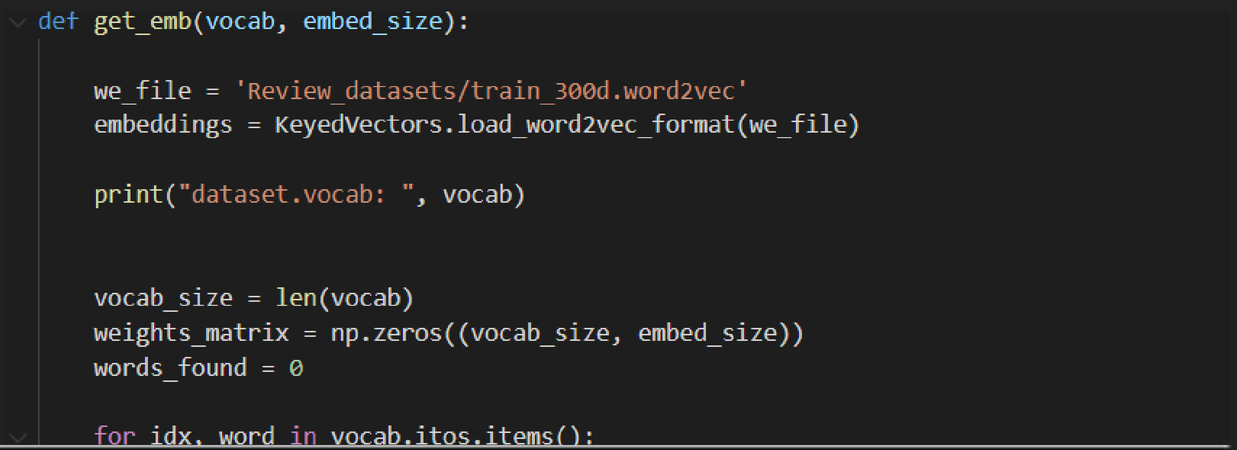
Claire

Haoran

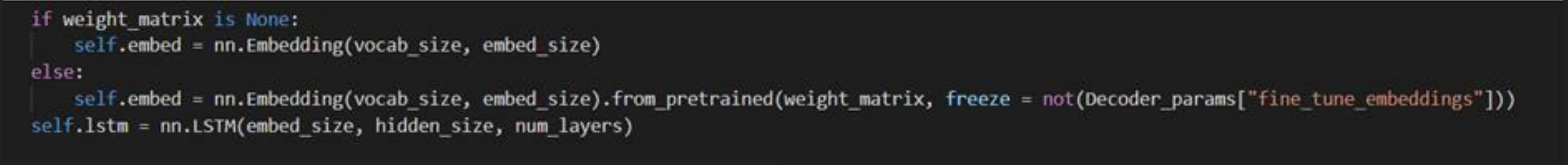
Stage 4: Modifications and Outcomes

Modification 1: Apply pre-trained GloVe Embeddings

- Function Written: get_emb, to extract weight matrix
- Modification in Decoder
- Fine tune word embeddings in training



```python
def get_emb(vocab, embed_size):

    we_file = 'Review_datasets/train_300d.word2vec'
    embeddings = KeyedVectors.load_word2vec_format(we_file)

    print("dataset.vocab: ", vocab)


    vocab_size = len(vocab)
    weights_matrix = np.zeros((vocab_size, embed_size))
    words_found = 0

    for idx, word in vocab.itos.items():
```

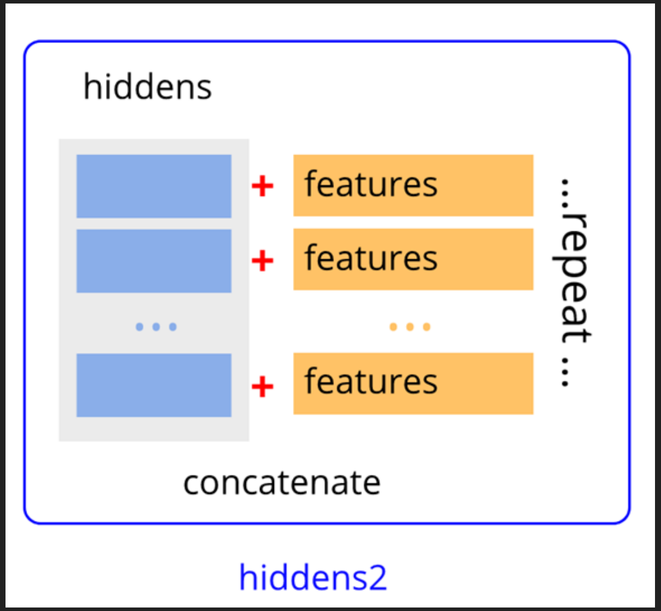


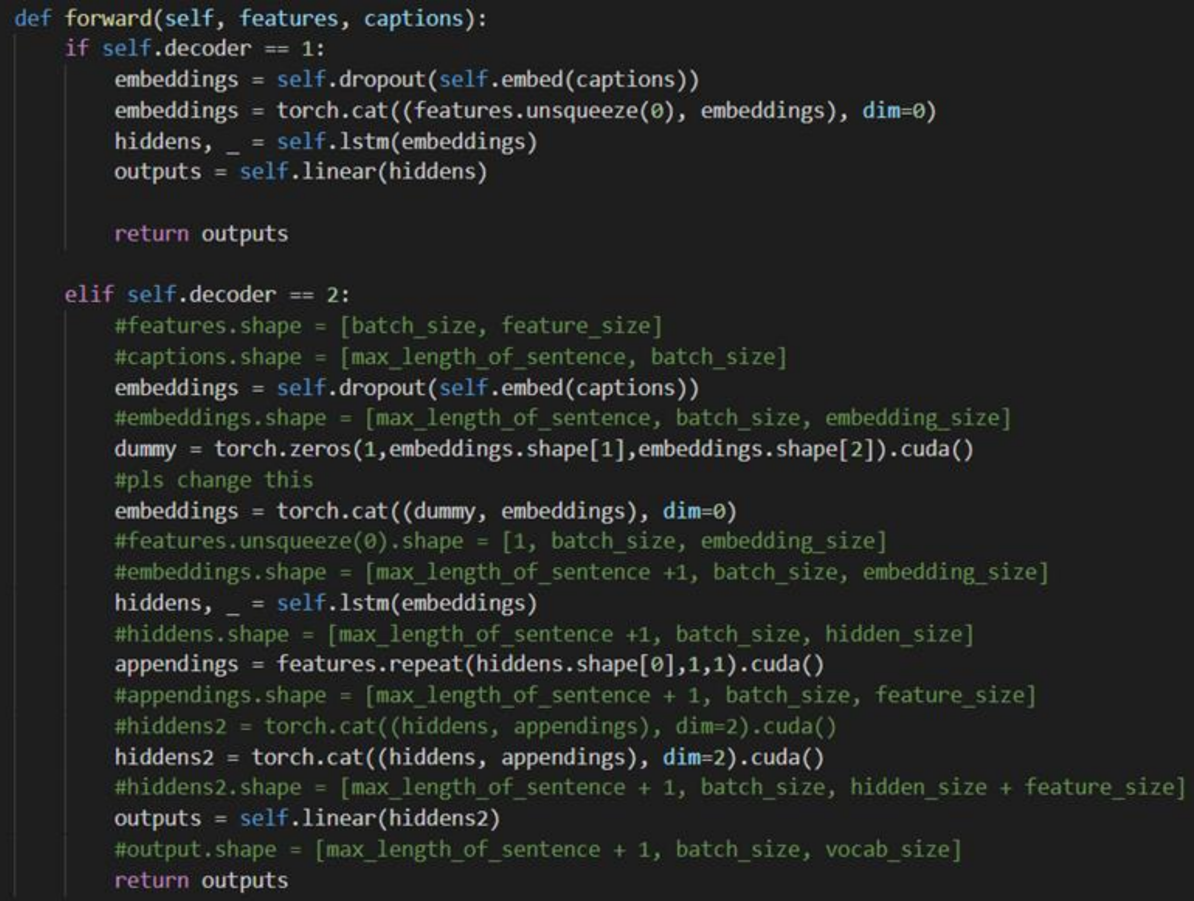
```python
if weight_matrix is None:
    self.embed = nn.Embedding(vocab_size, embed_size)
else:
    self.embed = nn.Embedding(vocab_size, embed_size).from_pretrained(weight_matrix, freeze = not(Decoder_params["fine_tune_embeddings"]))
self.lstm = nn.LSTM(embed_size, hidden_size, num_layers)
```

Stage 4: Modifications and Outcomes

Modification 2: Change the way concatenating information

- Motivation: increase the "importance" of image in final result





```
def forward(self, features, captions):
    if self.decoder == 1:
        embeddings = self.dropout(self.embed(captions))
        embeddings = torch.cat((features.unsqueeze(0), embeddings), dim=0)
        hiddens, _ = self.lstm(embeddings)
        outputs = self.linear(hiddens)

        return outputs

    elif self.decoder == 2:
        #features.shape = [batch_size, feature_size]
        #captions.shape = [max_length_of_sentence, batch_size]
        embeddings = self.dropout(self.embed(captions))
        #embeddings.shape = [max_length_of_sentence, batch_size, embedding_size]
        dummy = torch.zeros(1,embeddings.shape[1],embeddings.shape[2]).cuda()
        #pls change this
        embeddings = torch.cat((dummy, embeddings), dim=0)
        #features.unsqueeze(0).shape = [1, batch_size, embedding_size]
        #embeddings.shape = [max_length_of_sentence +1, batch_size, embedding_size]
        hiddens, _ = self.lstm(embeddings)
        #hiddens.shape = [max_length_of_sentence +1, batch_size, hidden_size]
        appendings = features.repeat(hiddens.shape[0],1,1).cuda()
        #appendings.shape = [max_length_of_sentence + 1, batch_size, feature_size]
        #hiddens2 = torch.cat((hiddens, appendings), dim=2).cuda()
        hiddens2 = torch.cat((hiddens, appendings), dim=2).cuda()
        #hiddens2.shape = [max_length_of_sentence + 1, batch_size, hidden_size + feature_size]
        outputs = self.linear(hiddens2)
        #output.shape = [max_length_of_sentence + 1, batch_size, vocab_size]
        return outputs
```

Stage 4: Modifications and Outcomes

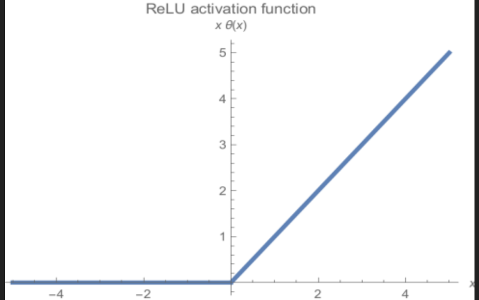
Modification 3: Disable Linear and ReLU

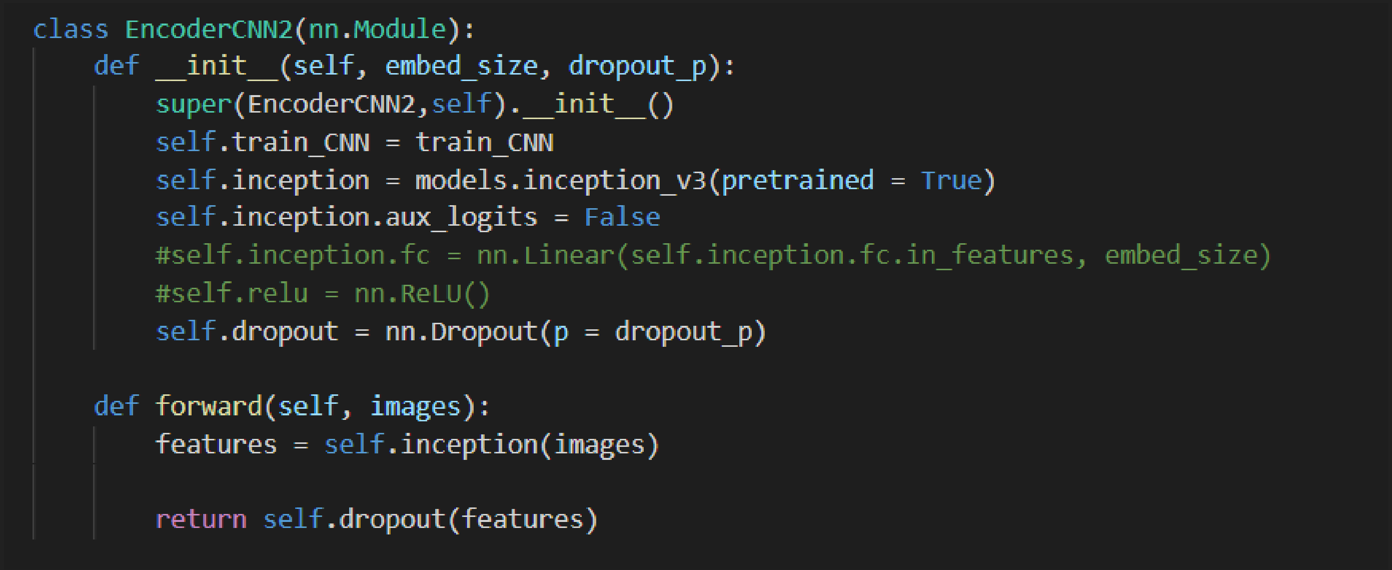
- Linear Transformation:

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

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

- ReLU





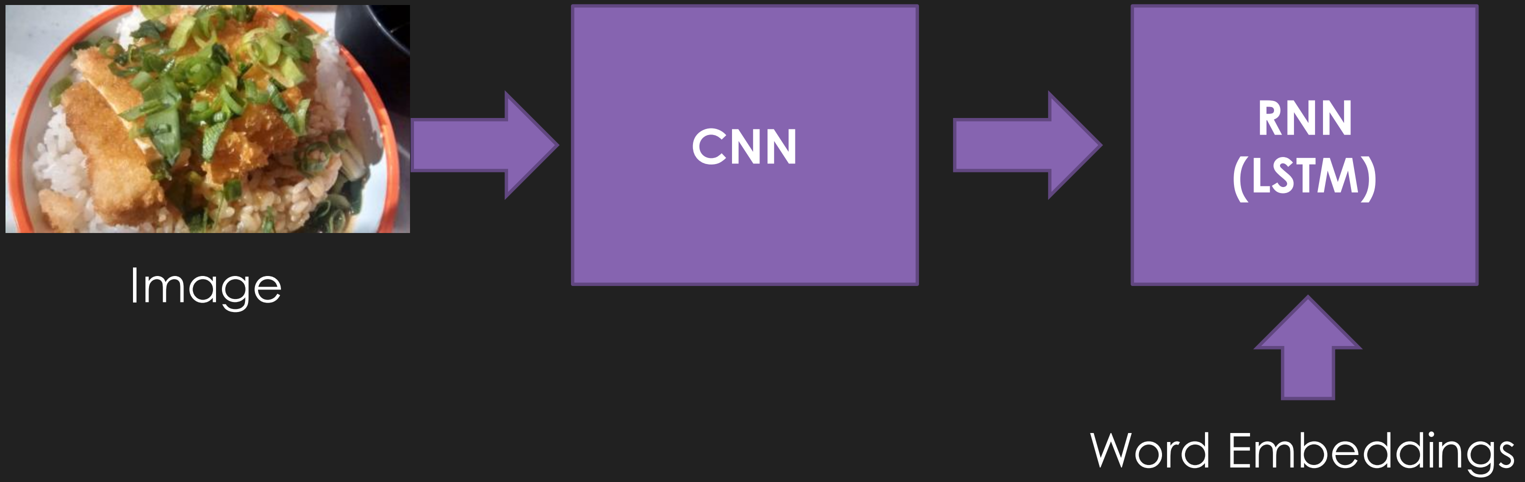
```
class EncoderCNN2(nn.Module):
    def __init__(self, embed_size, dropout_p):
        super(EncoderCNN2,self).__init__()
        self.train_CNN = train_CNN
        self.inception = models.inception_v3(pretrained = True)
        self.inception.aux_logits = False
        #self.inception.fc = nn.Linear(self.inception.fc.in_features, embed_size)
        #self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p = dropout_p)

    def forward(self, images):
        features = self.inception(images)

        return self.dropout(features)
```

Stage 4: Modifications and Outcomes

Modification 4: Another way to extract image features

- Vision Features: helped by Dr Nguyen, in form of dict
- Dictionary mapping: { image id: feature vector }

Pre-extracted vision features



Image

CNN

RNN (LSTM)

Word Embeddings

4 Modification: Quantitative Result

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%

Qualitative Result

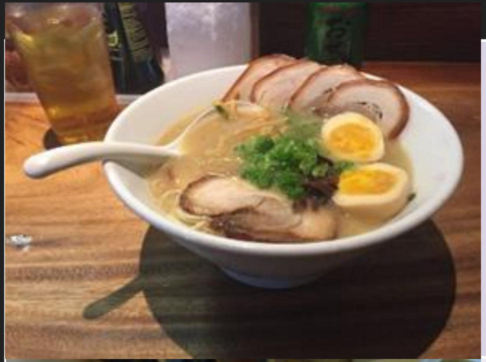
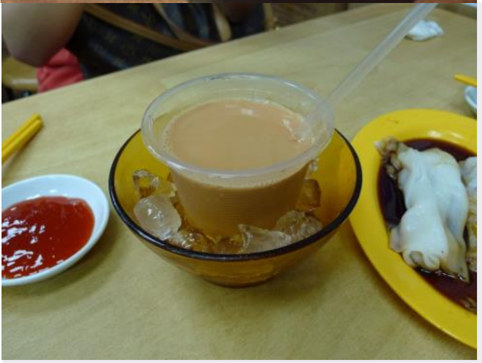
Text Generation ✓

Image Recognition ✓

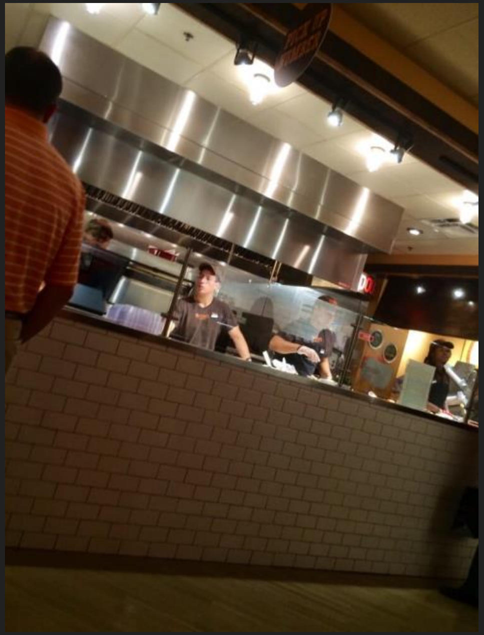
Qualitative Result:

- Multiple sentences, simple
- Able to differentiate images
- Able to recognise food
- Some repeating sentences

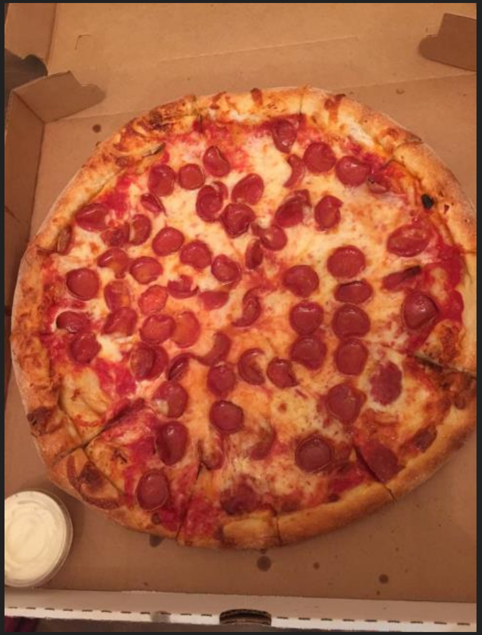
Problem solved!

	i got the shoyu ramen and it was very good . i got the shoyu ramen and it was very good . i got the shoyu ramen and it was very good .
	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 .

More predicted reviews



i have been to a buffet and i have to say it 's a great spot to have a very cool spot . i have been to a buffet . i have been to a buffet and i have to say it 's a great spot to have a



i got the pizza and i got the pizza and it was very good . i got the pizza and it was very good . i got the pizza and it was very good

Stage 4: Modifications and Outcomes

Other works done to manage project:

- Introduce “run sequence” to better manage runs
- Save parameters during training, then read in validation
- Replace positional arguments in model with keyword argument **params

Other modification attempts:

- Different activation functions: Leaky ReLU, PReLU
- Reversely concatenating features & embeddings
- Different vocab size
- Less dropout probability
- More…

Comparison

Team1	i got the shoyu ramen and it was very good . i got the shoyu ramen and it was very good . i got the shoyu ramen and it was very good .
Team3	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.

Q & A