

iAssist

A Deep Learning Application to assist differently abled people to identify picture from captions

CMPE 258 Project

Team Deep Dreamers

Haley | Jocelyn | Princy | Tripura

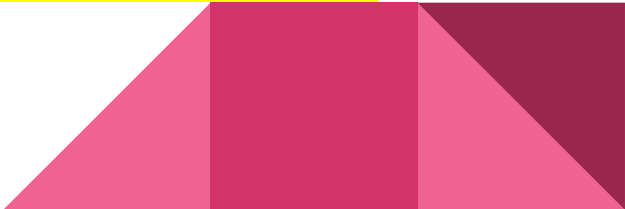
Introduction



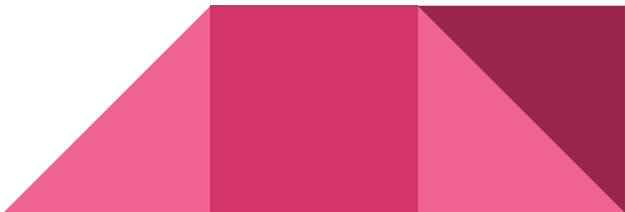
Can you write a caption?

- Goal of the application is to describe a scene in simple words to visually impaired people, just like a friend would if he or she was speaking to them.
 - Challenge is to identify the objects, their relations and generate a natural language description.
 - Solution is based on Encoder-Decoder based deep learning algorithm which combines both image and text.
-

Related work

- Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy, Andrej & Fei-Fei, Li. (2014)
 - Unifying visual-semantic embeddings with multimodal neural language models, Kiros et al (2015).
 - Show and tell: A neural image caption generator, Vinyals et al (2015).
 - **Show, Attend and Tell: Neural Image Caption Generation with Visual Attention**, Xu et al (2015)
- 

Methodology

- Transfer learning using CNN model InceptionV3 on Imagenet
 - Teacher forcing
 - Glove embeddings
 - Bahdanau local attention
 - Long Short-Term Memory(LSTM)
 - Gated Recurrent Units(GRU)
- 

CNN - RNN

- **Convolutional Neural Networks**(CNN) can generate a fixed length vector representation of the image by preserving spatial information.
- We use **InceptionV3**, trained on Imagenet as the feature extractor.
- **Recurrent Neural Networks**(RNN) can process sequential data like a stream of words, represented as time series.
- We experimented with two types of RNN, Long Short-Term Memory(LSTM) and Gated Recurrent Units(GRU)
- Global vectors for word representation, Glove is word-word co-occurrence matrix from a corpus, used to derive semantic relations between words.

Bahdanau Local attention, but why attention?

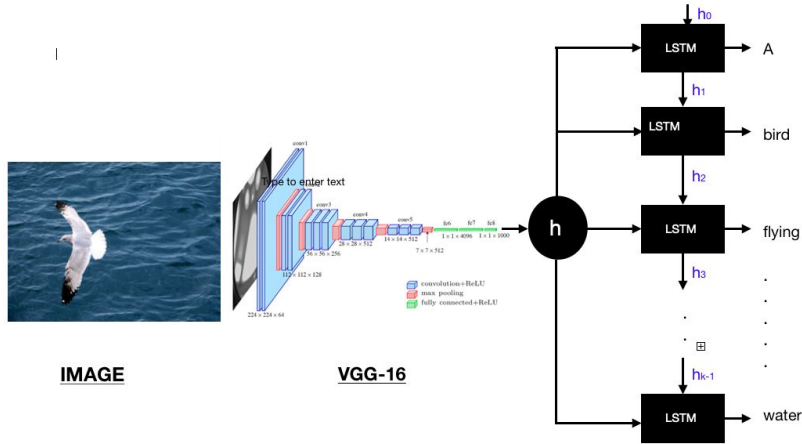
Idea - avoid attempting to learn a single vector representation sentences, instead, pay attention to specific input vectors of the input sequence based on the weights.

Local attention

- chooses to focus only on a small subset of the hidden states of the encoder per target word.
- Global is expensive and impractical while working on long sentences.
- The main part of the attention mechanism is the following two aspects: the decision needs to pay attention to which part of the input; the allocation of limited information processing resources to the important part.



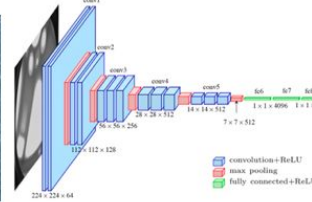
Architecture - with and without Attention



Baseline model Architecture without attention

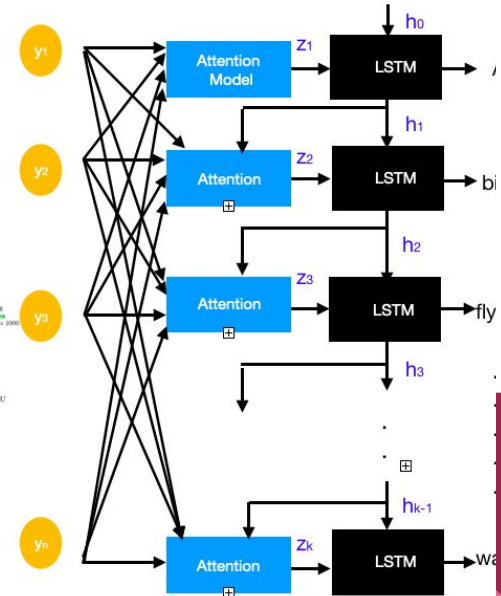


IMAGE



VGG-16

Architecture with attention



Dataset

Choice of datasets : Flickr8k, Flickr30k, MSCOCO

- **Selected Flickr8k because that can be trainable on an 8GB RAM**

Flickr8k_Dataset:

- 8092 images in JPEG format with different shapes and sizes.
- 80% for training, 20% for validation.

Flickr8k_text:

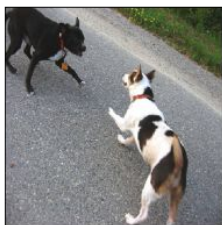
- 5 captions for each image (total 40460 captions).



Dataset - example



- A little girl in a pink dress going into a wooden cabin .
- A little girl climbing the stairs to her playhouse .
- A little girl climbing into a wooden playhouse .
- A girl going into a wooden building .
- A child in a pink dress is climbing up a set of stairs in an entry way .



- Two dogs on pavement moving toward each other .
- Two dogs of different breeds looking at each other on the road .
- A black dog and a white dog with brown spots are staring at each other in the street .
- A black dog and a tri-colored dog playing with each other on the road .
- A black dog and a spotted dog are fighting

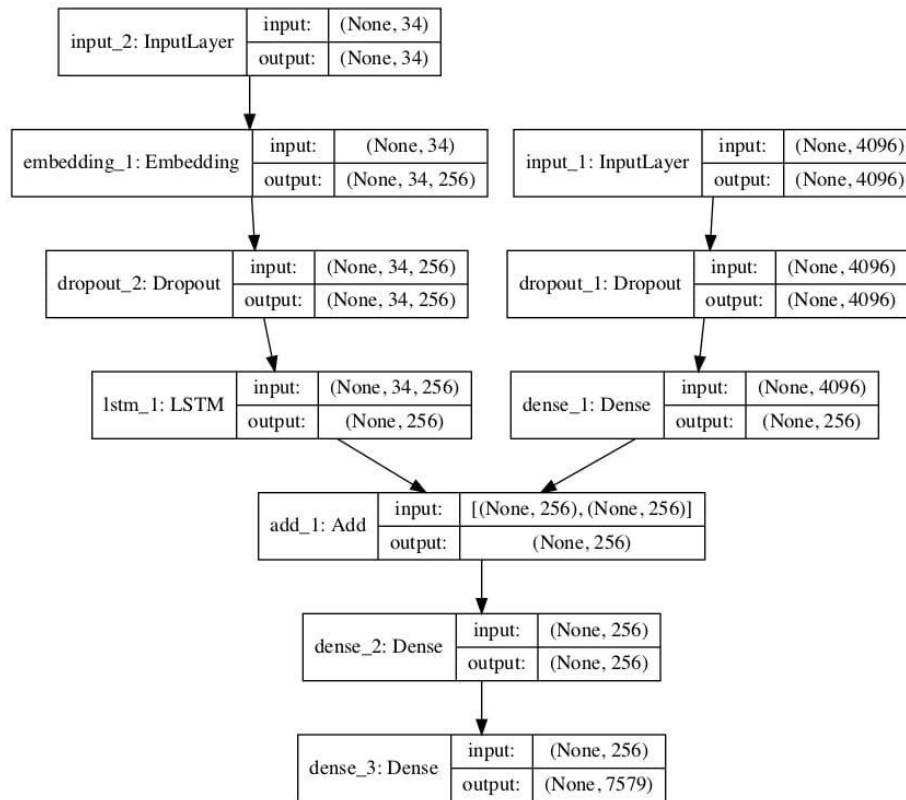


- Young girl with pigtails painting outside in the grass .
- There is a girl with pigtails sitting in front of a rainbow painting .
- A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
- A little girl is sitting in front of a large painted rainbow .
-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

Configurations

Optimizer	Adam/RMSProp
Loss Function	Sparse Categorical Entropy
Batch Size	64
Units	512
Epoch	20~50
Image Array	(2048,)
Caption Vector	37
Embedding Matrix	(8780, 200)
Dropout Layer	0.5

Vanilla Model on Keras Functional API



Tensorflow Subclass Model

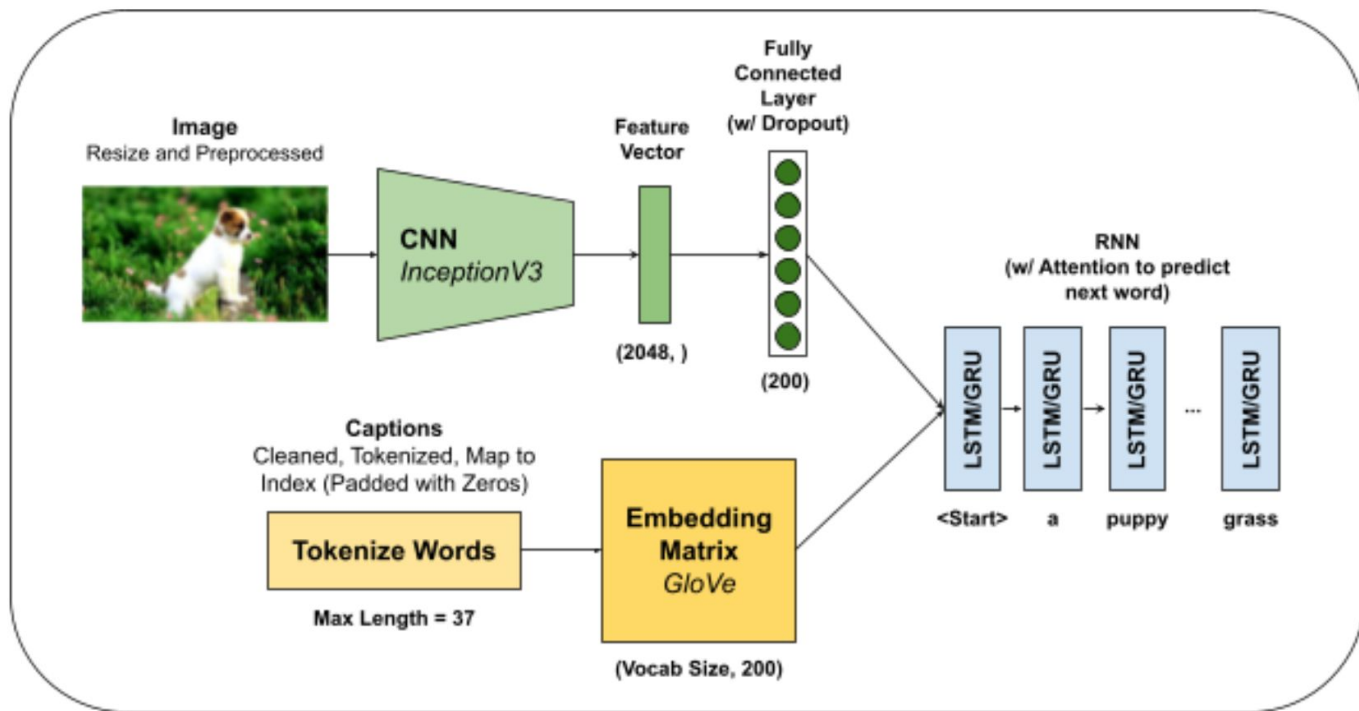
Model: "cnn_encoder_1"

Layer (type)	Output Shape	Param #
=====		
dropout_2 (Dropout)	multiple	0
=====		
dense_6 (Dense)	multiple	409800
=====		
Total params: 409,800		
Trainable params: 409,800		
Non-trainable params: 0		

Model: "rnn_decoder_1"

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	multiple	1756000
=====		
lstm_1 (LSTM)	multiple	1869824
=====		
dropout_3 (Dropout)	multiple	0
=====		
batch_normalization_95 (Batch Normalization)	multiple	2048
=====		
dense_7 (Dense)	multiple	262656
=====		
dense_8 (Dense)	multiple	4504140
=====		
bahdanau_attention_1 (Bahdanau Attention)	multiple	366081
=====		
Total params: 8,760,749		
Trainable params: 7,003,725		
Non-trainable params: 1,757,024		

CNN - RNN architecture with Glove and Attention



Training

```
def train_step(img_tensor, target):
    loss = 0
    # initializing the hidden state for each batch
    # because the captions are not related from image to image
    hidden = decoder.reset_state(batch_size=target.shape[0])
    # The decoder input initial input is the start token
    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * target.shape[0], 1)
    with tf.GradientTape() as tape:
        features = encoder(img_tensor)
        for i in range(1, target.shape[1]):
            # passing the features through the decoder
            predictions, hidden, _ = decoder(dec_input, features, hidden)
            loss += loss_function(target[:, i], predictions)
            # using teacher forcing
            dec_input = tf.expand_dims(target[:, i], 1)
    total_loss = (loss / int(target.shape[1]))
    trainable_variables = encoder.trainable_variables + decoder.trainable_variables
    gradients = tape.gradient(loss, trainable_variables)
    optimizer.apply_gradients(zip(gradients, trainable_variables))

    return loss, total_loss
```

Initial hidden state with
shape: [64, 512]

Decoder input
is initialize with
<start> token,
shape : (64, 1)

Target shape [1] is
37 representing
max caption length

Put ground truth
back to the model

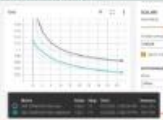
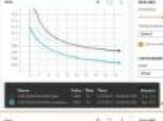
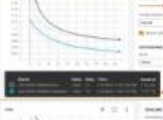


Encoder output
shape: (64, 200)

Evaluation Metrics

BLEU (Bilingual Evaluation Understudy)	GLEU (Google BLEU)	WER (Word Error Rate)
Compares the number of matching n-grams from the predicted caption to the reference captions.	<u>Recall</u> : # of matching n-grams/# total n-grams in the reference <u>Precision</u> : # of matching n-grams/# of total n-grams generated caption	<u>Substitution</u> : number of words need to be substituted to match reference <u>Deletion</u> : number of words dropped from the reference <u>Insertion</u> : number of extra words added compared to the reference <u>Number</u> : the total number of correct words WER = (S+D+I)/N
1.0 - perfect match, 0.0 - perfect mismatch	1.0 - perfect match, 0.0 - perfect mismatch	The smaller the value, the better the prediction

NLP Metrics Source: <https://github.com/gcunhase/NLPMetrics>

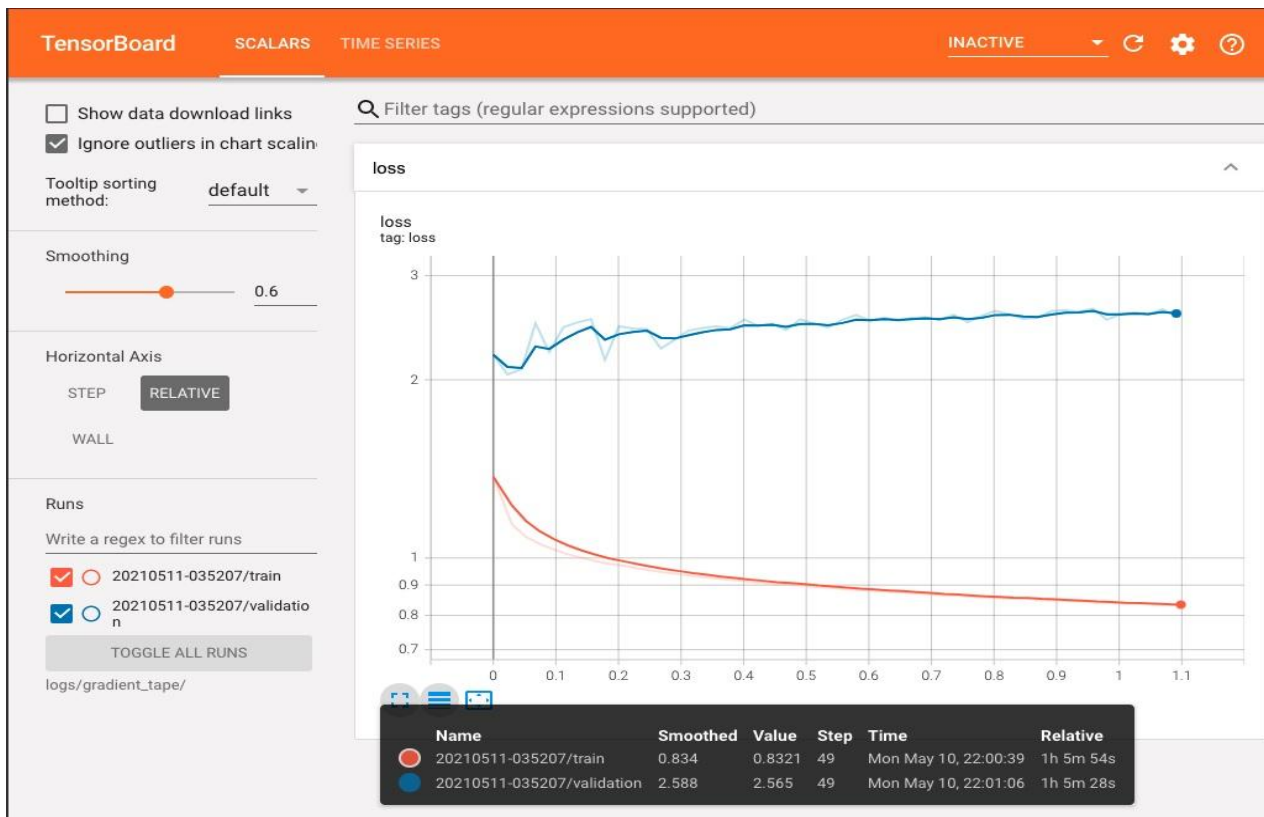
Experiments

# Top Vocab	Encoder	Decoder	Optimizer	Batch, Epoch	Train Loss	Val Loss	Training Time	Batch, Epoch
top_k = 1600	Dense, Relu	Embedding, GRU, Dense, Dense, Attention	Adam	64, 20	1.064	1.027	16m47s with GPU	
top_k = 1600	Dense, Relu	Embedding, GRU, Dropout (0.5), Dense, Dense, Attention	Adam	64, 20	1.063	1.025	17m11s with GPU	
top_k = 1600	Dense, Relu	Embedding, GRU, Dropout (0.5), Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64, 20	1.052	1.024	17m20s with GPU	
top_k = 1600	Dropout, Dense, Relu	Embedding, GRU, Dropout (0.5), Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64, 20	0.888	0.856	17m30s with GPU	
top_k = 1600	Dropout, Dense, Relu	Embedding, LSTM, Dropout (0.5), Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64, 25	0.877	0.839	29m 51s with GPU	
top_k = 5000		Embedding, LSTM, Dense, Dropout (0.5), BatchNorm, Dense, Attention	RMSProp	64, 25	0.998	0.958	32m21 with GPU	
top_k = 5000		Embedding, LSTM, Dropout (0.5), Dense, Dropout (0.5), BatchNorm, Dense, Attention	RMSProp	64, 25	0.999	0.96		
top_k = 5000	Dropout, Dense, Relu	Embedding, GRU, Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64, 50	0.838	2.623		
top_k = 5000	Dense, Dropout, Relu	Embedding, LSTM, Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64, 50	0.832	2.564	57m28s	

Model Tracker:

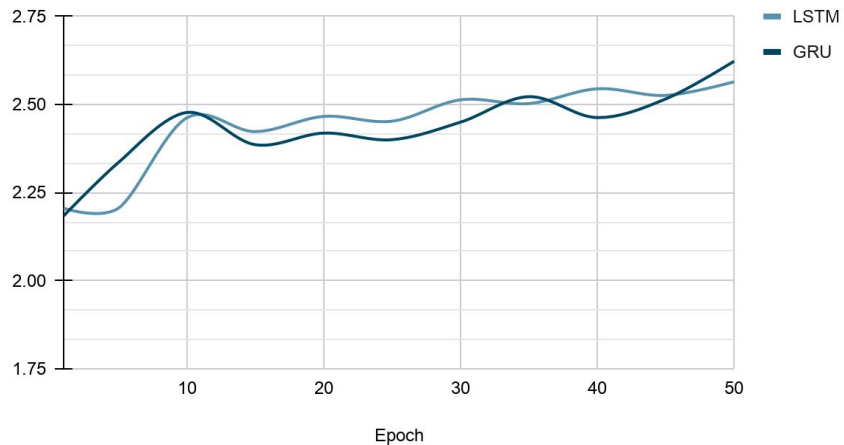
https://docs.google.com/spreadsheets/u/1/d/1VbMC_GL8O9GxUB3ui934VhXeaLtrKatFQrs-JyZHR4I/edit?usp=drive_web&ouid=109949769242133710329

TensorBoard visualization - LSTM

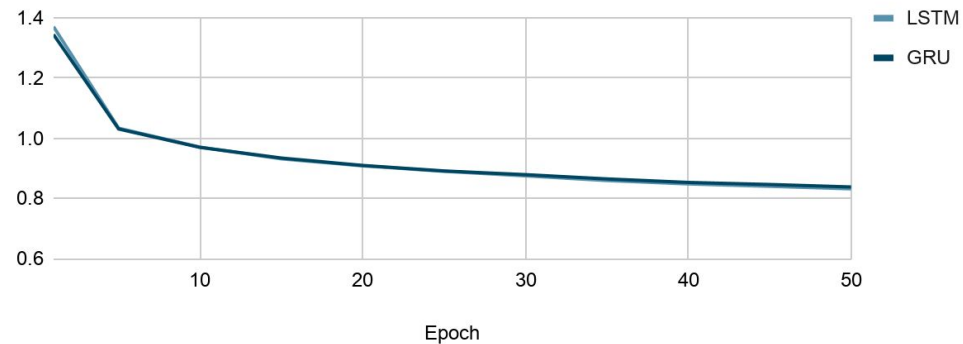


Total train, Validation loss

Validation Loss: LSTM vs. GRU



Total Training Loss: LSTM vs. GRU



Caption Evaluation

* comparison on one image

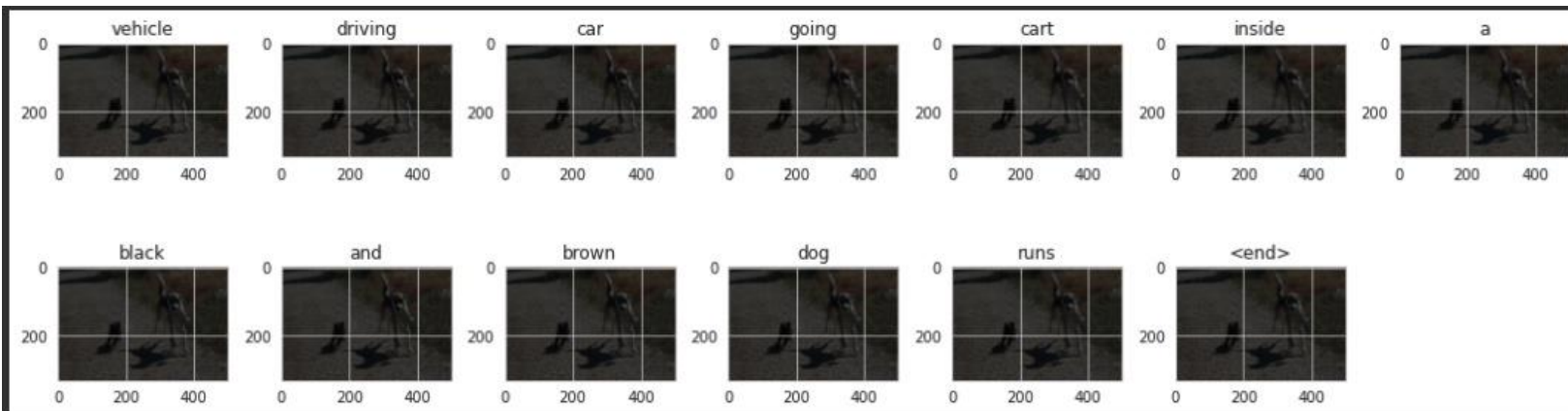
		BLEU Score					
Method	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	GLEU	WER
TF Func. API	Vanilla Model	0.714	0.662	0.634	0.494	0.271	13.4
TF Subclass	LSTM w/ Attention	0.162	0.067	0.197	0.259	0.042	32.8
	+Beam Search	0.138	0.062	0.19	0.25	0.036	32.4
TF Subclass	GRU w/ Attention	0.222	0.114	0.272	0.338	0.075	16.8
	+Beam Search	0.138	0.109	0.113	0.067	0.079	32.2

Reference Caption: a man in a dark shirt and shorts is standing on top of a high graffitied rock

Vanilla Model Generated Caption: a man in a red shirt is standing on a rock overlooking a waterfall



Results - the okay



Real Caption: a little black dog is running beside a large brown and white dog with a large rope in its mouth <end>

Real Caption: a small black dog is running alongside a bigger tricolored dog that is carrying a rope toy in its mouth <end>

Real Caption: a small dog and a large dog with a rope in its mouth <end>

Real Caption: two dogs running down a dirt road <end>

Real Caption: the large dog is holding a rope in its mouth and the little dog is walking beside it <end>

Prediction Caption: vehicle driving car going cart inside a black and brown dog runs <end>

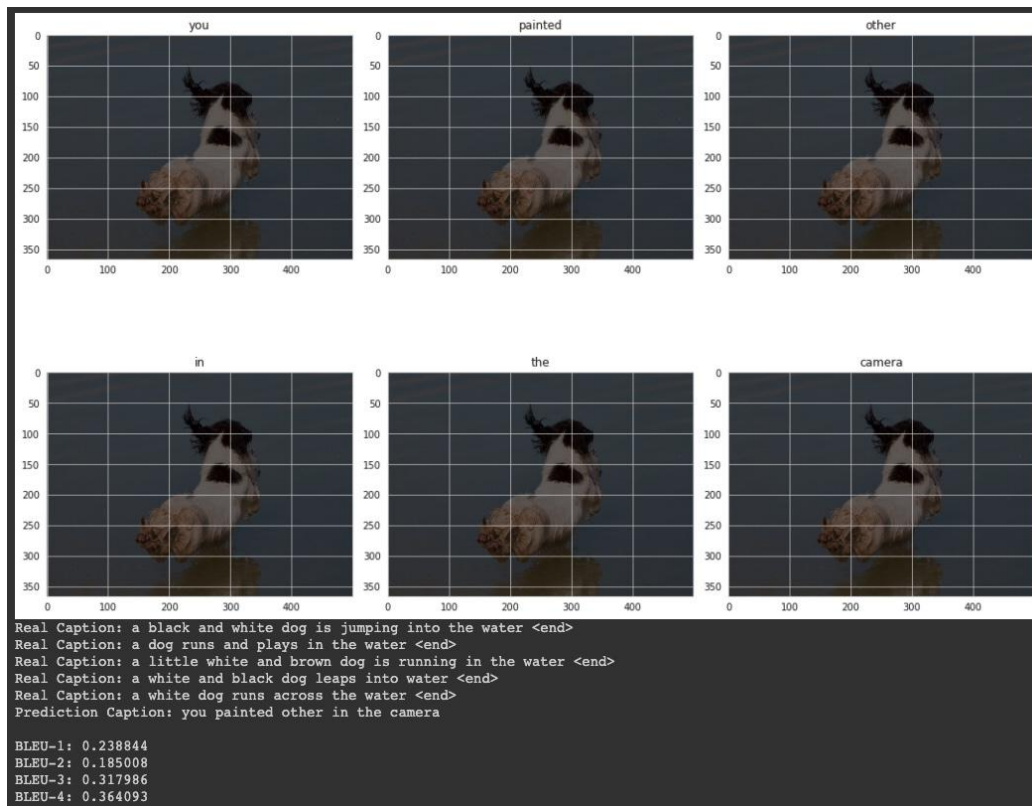
BLEU-1: 0.427367

BLEU-2: 0.629067

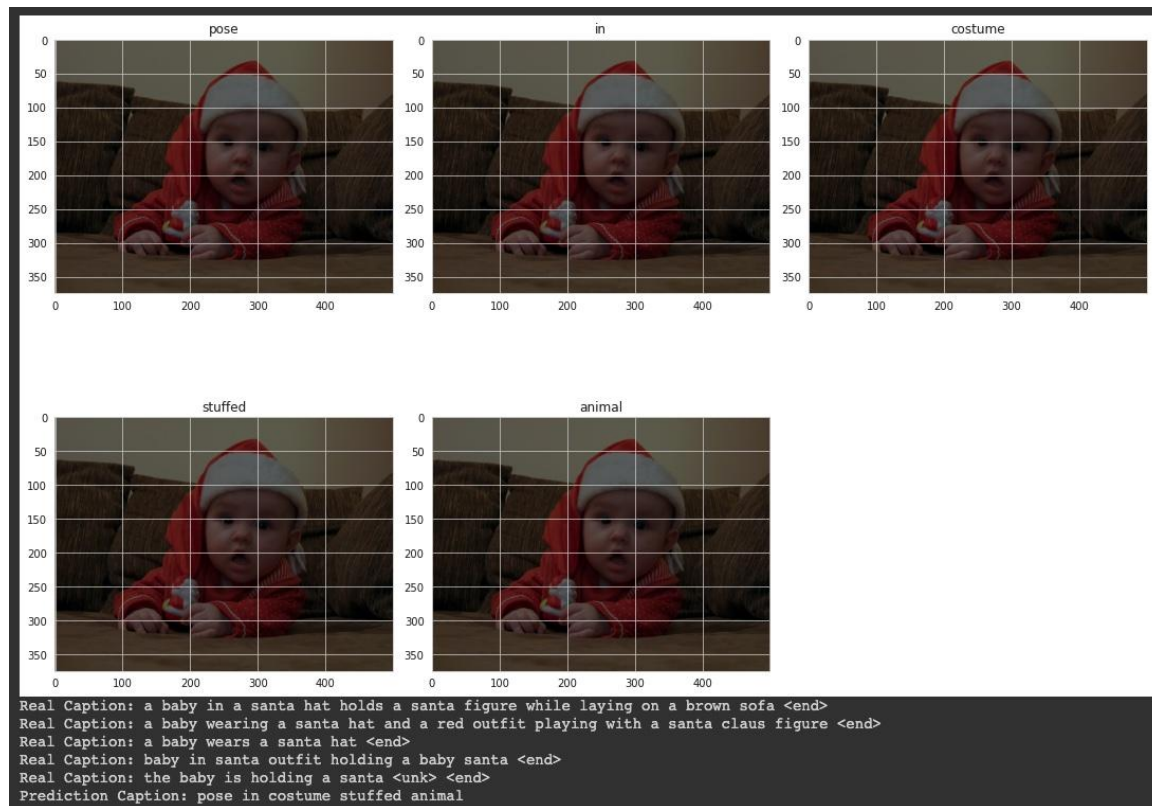
BLEU-3: 0.734269

BLEU-4: 0.763211

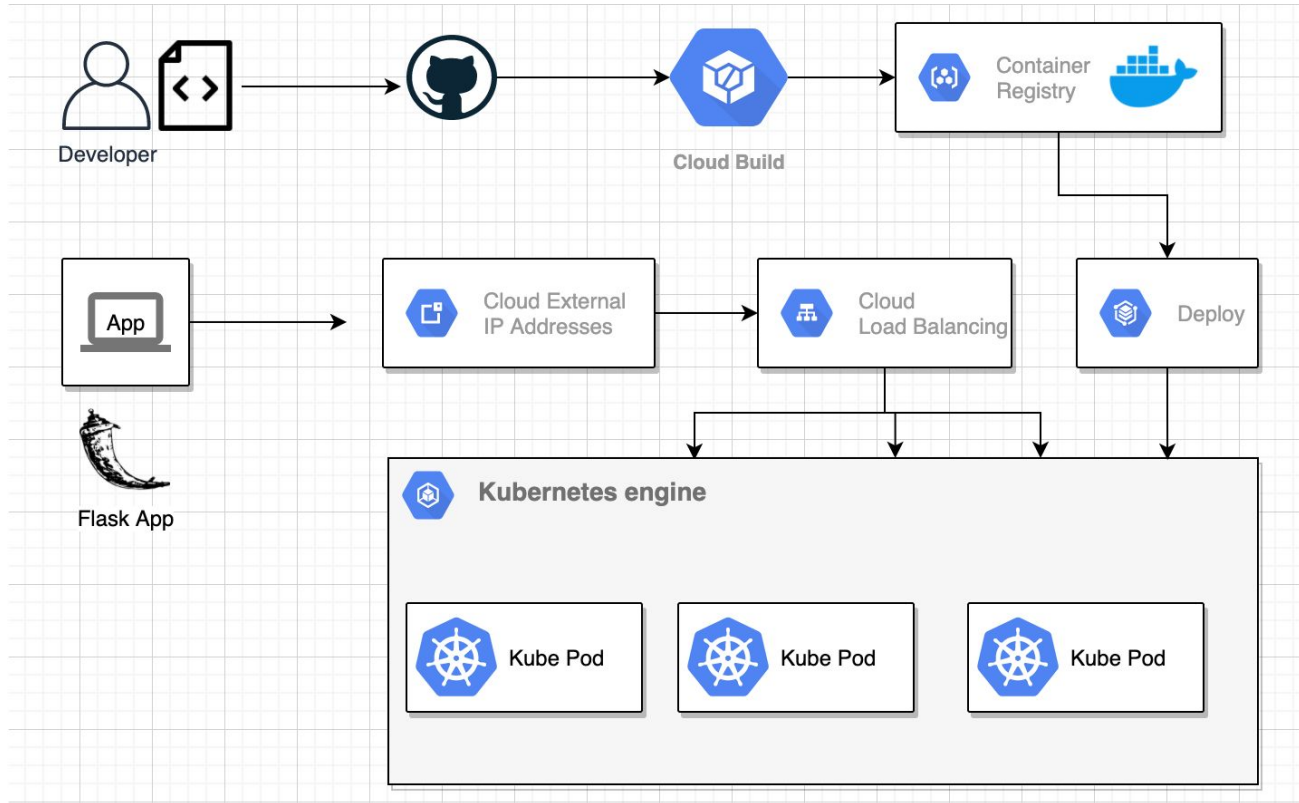
Results - the bad



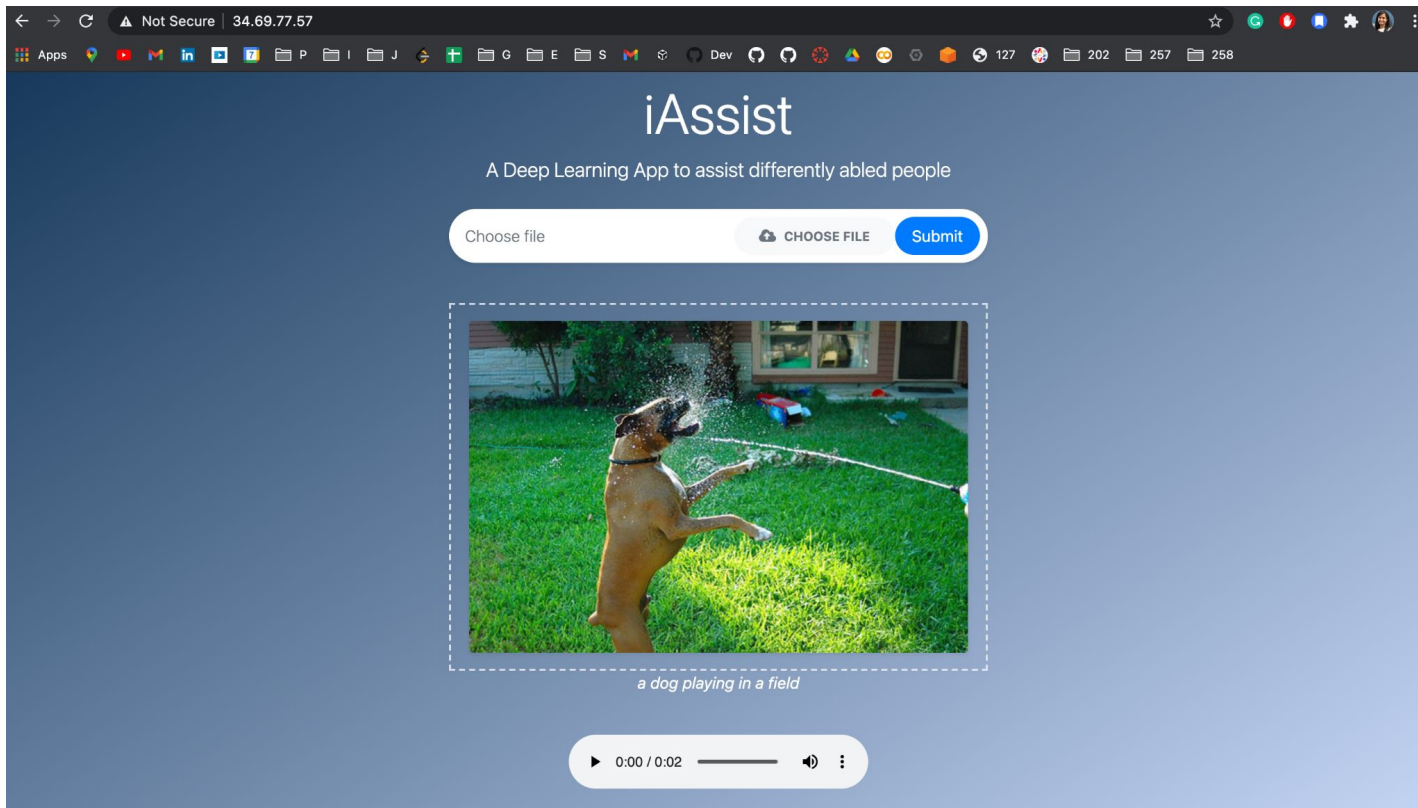
Results - the interesting




Cloud architecture and CI/CD implementation



Web app inference



AI Platform (TFX Kubeflow Pipeline)

- Experimented some trained data model in AI-Platform and TFX Kubeflow Pipeline using Kubernetes container for Image-captioning in [github kubeflow/pipeline](https://github.com/kubeflow/pipeline).
 - Deploying a trained model in AI Platform TFX Kubeflow Pipeline needs some Kubernetes configuration in order to run it.
 - The trained model ms-coco dataset have some issues in downloading to GCP. The tutorial for data loading is not working so we need to load it from the website of MS-COCO dataset and upload it to GCP.
 - Getting setup the ms-coco preprocessing data images is not fully completed.
 - Issues encountered is how to deploy the preprocessing function using GKE (Kubernetes container).
- 

AI Platform (TFX Kubeflow Pipeline)

The screenshot shows a JupyterLab environment. On the left is a file browser pane showing the directory structure of a project. The main area on the right displays a Jupyter notebook with two cells. The first cell contains text explaining the purpose of the notebook: to download a subset of the MS COCO dataset and upload it to a GCS bucket. The second cell contains a Jupyter magic command `[5]:` followed by a shell script that sets environment variables for the GCS bucket and dataset path, and then runs a series of commands to download the dataset using `wget`, unzip it, and sync it to the GCS bucket.

File Browser (Left Pane):

Name	Last Modified
src	5 days ago
(pj)Image Captioning...	a day ago
deep-dreamers-ia...	6 days ago
gcd3_pipeline.zip	a day ago
Image Captioning TF...	a minute ago
README.md	6 days ago
simplekfp.ipynb	a day ago

Jupyter Notebook (Right Pane):

First, you have to download the [MS COCO dataset](#). This sample uses both the 2014 train images and 2014 train/val annotations. The following cells download a small subset (<1000 imgs) of the dataset and the annotations to the GCS bucket specified below with `GCS_DATASET_PATH`.

```
[5]: # Location to download dataset and put onto GCS (should be associated
# with Kubeflow project)
GCS_BUCKET = 'gs://kubeflowpipelines-default'
GCS_DATASET_PATH = GCS_BUCKET + '/ms-coco'
```

Download images

Downloads images to `${GCS_DATASET_PATH}/train2014/train2014`

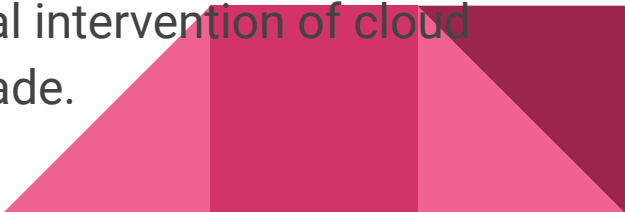
```
[30]: # Tutorial instructions
# Download images (use -x to ignore ~99% of images)
#gsutil -m rsync -x ".*\0.jpg|.*\1.jpg|.*\2.jpg|.*\3.jpg|.*\4.jpg|.*\5.jpg|.*\6.jpg|.*\7.jpg|.*\8.jpg|.*\9.jpg|.*\10.jpg|.*\11.jpg|.*\12.jpg|.*\13.jpg|.*\14.jpg|.*\15.jpg|.*\16.jpg|.*\17.jpg|.*\18.jpg|.*\19.jpg|.*\20.jpg|.*\21.jpg|.*\22.jpg|.*\23.jpg|.*\24.jpg|.*\25.jpg|.*\26.jpg|.*\27.jpg|.*\28.jpg|.*\29.jpg|.*\30.jpg|.*\31.jpg|.*\32.jpg|.*\33.jpg|.*\34.jpg|.*\35.jpg|.*\36.jpg|.*\37.jpg|.*\38.jpg|.*\39.jpg|.*\40.jpg|.*\41.jpg|.*\42.jpg|.*\43.jpg|.*\44.jpg|.*\45.jpg|.*\46.jpg|.*\47.jpg|.*\48.jpg|.*\49.jpg|.*\50.jpg|.*\51.jpg|.*\52.jpg|.*\53.jpg|.*\54.jpg|.*\55.jpg|.*\56.jpg|.*\57.jpg|.*\58.jpg|.*\59.jpg|.*\60.jpg|.*\61.jpg|.*\62.jpg|.*\63.jpg|.*\64.jpg|.*\65.jpg|.*\66.jpg|.*\67.jpg|.*\68.jpg|.*\69.jpg|.*\70.jpg|.*\71.jpg|.*\72.jpg|.*\73.jpg|.*\74.jpg|.*\75.jpg|.*\76.jpg|.*\77.jpg|.*\78.jpg|.*\79.jpg|.*\80.jpg|.*\81.jpg|.*\82.jpg|.*\83.jpg|.*\84.jpg|.*\85.jpg|.*\86.jpg|.*\87.jpg|.*\88.jpg|.*\89.jpg|.*\90.jpg|.*\91.jpg|.*\92.jpg|.*\93.jpg|.*\94.jpg|.*\95.jpg|.*\96.jpg|.*\97.jpg|.*\98.jpg|.*\99.jpg|.*\100.jpg|.*\101.jpg|.*\102.jpg|.*\103.jpg|.*\104.jpg|.*\105.jpg|.*\106.jpg|.*\107.jpg|.*\108.jpg|.*\109.jpg|.*\110.jpg|.*\111.jpg|.*\112.jpg|.*\113.jpg|.*\114.jpg|.*\115.jpg|.*\116.jpg|.*\117.jpg|.*\118.jpg|.*\119.jpg|.*\120.jpg|.*\121.jpg|.*\122.jpg|.*\123.jpg|.*\124.jpg|.*\125.jpg|.*\126.jpg|.*\127.jpg|.*\128.jpg|.*\129.jpg|.*\130.jpg|.*\131.jpg|.*\132.jpg|.*\133.jpg|.*\134.jpg|.*\135.jpg|.*\136.jpg|.*\137.jpg|.*\138.jpg|.*\139.jpg|.*\140.jpg|.*\141.jpg|.*\142.jpg|.*\143.jpg|.*\144.jpg|.*\145.jpg|.*\146.jpg|.*\147.jpg|.*\148.jpg|.*\149.jpg|.*\150.jpg|.*\151.jpg|.*\152.jpg|.*\153.jpg|.*\154.jpg|.*\155.jpg|.*\156.jpg|.*\157.jpg|.*\158.jpg|.*\159.jpg|.*\160.jpg|.*\161.jpg|.*\162.jpg|.*\163.jpg|.*\164.jpg|.*\165.jpg|.*\166.jpg|.*\167.jpg|.*\168.jpg|.*\169.jpg|.*\170.jpg|.*\171.jpg|.*\172.jpg|.*\173.jpg|.*\174.jpg|.*\175.jpg|.*\176.jpg|.*\177.jpg|.*\178.jpg|.*\179.jpg|.*\180.jpg|.*\181.jpg|.*\182.jpg|.*\183.jpg|.*\184.jpg|.*\185.jpg|.*\186.jpg|.*\187.jpg|.*\188.jpg|.*\189.jpg|.*\190.jpg|.*\191.jpg|.*\192.jpg|.*\193.jpg|.*\194.jpg|.*\195.jpg|.*\196.jpg|.*\197.jpg|.*\198.jpg|.*\199.jpg|.*\200.jpg|.*\201.jpg|.*\202.jpg|.*\203.jpg|.*\204.jpg|.*\205.jpg|.*\206.jpg|.*\207.jpg|.*\208.jpg|.*\209.jpg|.*\210.jpg|.*\211.jpg|.*\212.jpg|.*\213.jpg|.*\214.jpg|.*\215.jpg|.*\216.jpg|.*\217.jpg|.*\218.jpg|.*\219.jpg|.*\220.jpg|.*\221.jpg|.*\222.jpg|.*\223.jpg|.*\224.jpg|.*\225.jpg|.*\226.jpg|.*\227.jpg|.*\228.jpg|.*\229.jpg|.*\230.jpg|.*\231.jpg|.*\232.jpg|.*\233.jpg|.*\234.jpg|.*\235.jpg|.*\236.jpg|.*\237.jpg|.*\238.jpg|.*\239.jpg|.*\240.jpg|.*\241.jpg|.*\242.jpg|.*\243.jpg|.*\244.jpg|.*\245.jpg|.*\246.jpg|.*\247.jpg|.*\248.jpg|.*\249.jpg|.*\250.jpg|.*\251.jpg|.*\252.jpg|.*\253.jpg|.*\254.jpg|.*\255.jpg|.*\256.jpg|.*\257.jpg|.*\258.jpg|.*\259.jpg|.*\260.jpg|.*\261.jpg|.*\262.jpg|.*\263.jpg|.*\264.jpg|.*\265.jpg|.*\266.jpg|.*\267.jpg|.*\268.jpg|.*\269.jpg|.*\270.jpg|.*\271.jpg|.*\272.jpg|.*\273.jpg|.*\274.jpg|.*\275.jpg|.*\276.jpg|.*\277.jpg|.*\278.jpg|.*\279.jpg|.*\280.jpg|.*\281.jpg|.*\282.jpg|.*\283.jpg|.*\284.jpg|.*\285.jpg|.*\286.jpg|.*\287.jpg|.*\288.jpg|.*\289.jpg|.*\290.jpg|.*\291.jpg|.*\292.jpg|.*\293.jpg|.*\294.jpg|.*\295.jpg|.*\296.jpg|.*\297.jpg|.*\298.jpg|.*\299.jpg|.*\300.jpg|.*\301.jpg|.*\302.jpg|.*\303.jpg|.*\304.jpg|.*\305.jpg|.*\306.jpg|.*\307.jpg|.*\308.jpg|.*\309.jpg|.*\310.jpg|.*\311.jpg|.*\312.jpg|.*\313.jpg|.*\314.jpg|.*\315.jpg|.*\316.jpg|.*\317.jpg|.*\318.jpg|.*\319.jpg|.*\320.jpg|.*\321.jpg|.*\322.jpg|.*\323.jpg|.*\324.jpg|.*\325.jpg|.*\326.jpg|.*\327.jpg|.*\328.jpg|.*\329.jpg|.*\330.jpg|.*\331.jpg|.*\332.jpg|.*\333.jpg|.*\334.jpg|.*\335.jpg|.*\336.jpg|.*\337.jpg|.*\338.jpg|.*\339.jpg|.*\340.jpg|.*\341.jpg|.*\342.jpg|.*\343.jpg|.*\344.jpg|.*\345.jpg|.*\346.jpg|.*\347.jpg|.*\348.jpg|.*\349.jpg|.*\350.jpg|.*\351.jpg|.*\352.jpg|.*\353.jpg|.*\354.jpg|.*\355.jpg|.*\356.jpg|.*\357.jpg|.*\358.jpg|.*\359.jpg|.*\360.jpg|.*\361.jpg|.*\362.jpg|.*\363.jpg|.*\364.jpg|.*\365.jpg|.*\366.jpg|.*\367.jpg|.*\368.jpg|.*\369.jpg|.*\370.jpg|.*\371.jpg|.*\372.jpg|.*\373.jpg|.*\374.jpg|.*\375.jpg|.*\376.jpg|.*\377.jpg|.*\378.jpg|.*\379.jpg|.*\380.jpg|.*\381.jpg|.*\382.jpg|.*\383.jpg|.*\384.jpg|.*\385.jpg|.*\386.jpg|.*\387.jpg|.*\388.jpg|.*\389.jpg|.*\390.jpg|.*\391.jpg|.*\392.jpg|.*\393.jpg|.*\394.jpg|.*\395.jpg|.*\396.jpg|.*\397.jpg|.*\398.jpg|.*\399.jpg|.*\400.jpg|.*\401.jpg|.*\402.jpg|.*\403.jpg|.*\404.jpg|.*\405.jpg|.*\406.jpg|.*\407.jpg|.*\408.jpg|.*\409.jpg|.*\410.jpg|.*\411.jpg|.*\412.jpg|.*\413.jpg|.*\414.jpg|.*\415.jpg|.*\416.jpg|.*\417.jpg|.*\418.jpg|.*\419.jpg|.*\420.jpg|.*\421.jpg|.*\422.jpg|.*\423.jpg|.*\424.jpg|.*\425.jpg|.*\426.jpg|.*\427.jpg|.*\428.jpg|.*\429.jpg|.*\430.jpg|.*\431.jpg|.*\432.jpg|.*\433.jpg|.*\434.jpg|.*\435.jpg|.*\436.jpg|.*\437.jpg|.*\438.jpg|.*\439.jpg|.*\440.jpg|.*\441.jpg|.*\442.jpg|.*\443.jpg|.*\444.jpg|.*\445.jpg|.*\446.jpg|.*\447.jpg|.*\448.jpg|.*\449.jpg|.*\450.jpg|.*\451.jpg|.*\452.jpg|.*\453.jpg|.*\454.jpg|.*\455.jpg|.*\456.jpg|.*\457.jpg|.*\458.jpg|.*\459.jpg|.*\460.jpg|.*\461.jpg|.*\462.jpg|.*\463.jpg|.*\464.jpg|.*\465.jpg|.*\466.jpg|.*\467.jpg|.*\468.jpg|.*\469.jpg|.*\470.jpg|.*\471.jpg|.*\472.jpg|.*\473.jpg|.*\474.jpg|.*\475.jpg|.*\476.jpg|.*\477.jpg|.*\478.jpg|.*\479.jpg|.*\480.jpg|.*\481.jpg|.*\482.jpg|.*\483.jpg|.*\484.jpg|.*\485.jpg|.*\486.jpg|.*\487.jpg|.*\488.jpg|.*\489.jpg|.*\490.jpg|.*\491.jpg|.*\492.jpg|.*\493.jpg|.*\494.jpg|.*\4
```

Conclusion

- Major challenge - training the model, often met with quota limitations in Colab
- One epoch can train for 80 secs with GPU and **30 mins without GPU**
- Tutorial from Tensorflow github for TFX Image-Captioning have some inaccuracies instructions for loading the dataset.
- Challenges on deployment of model in Kubernetes (TFX Kubeflow) pipeline.
- There was challenges in implementing the TFX pipeline, on how to write the trainer code to deserialize the TFRecords to binary for ingestion and training.



Key Learnings

- Encoder-Decoder model relating to image and text captioning.
 - Transfer learning CNN model Inception V3.
 - Subclassing is useful for customizing model, but can get overly complicated and difficult to debug
 - Deploying Flask web-app model in GCP Cloud
 - Experiments different model methodology BEAM Search, AI Platform, TFX Kubeflow Pipeline.
 - TFX Pipeline in AI Platform
 - CI/CD setup was quite useful and avoided the manual intervention of cloud deployment whenever there was a code/model upgrade.
- 

Future work

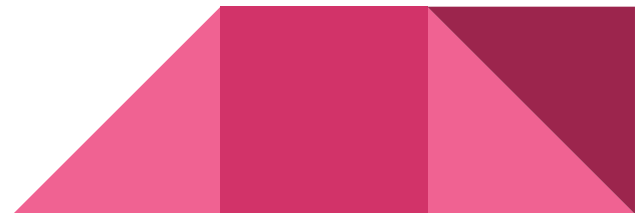
- Larger dataset with better compute resources
- More hyper-parameter tunings.
- Streaming video captioning
- Experiment with Transformer in place of GRU/LSTM
- Soft/Hard attention
- Try EfficientNet which is a CNN with better accuracy and efficiency.
- Self-Critical Sequence Training (SCST)
- Extending to other use cases like image search/retrieval



Collaboration/Responsibility

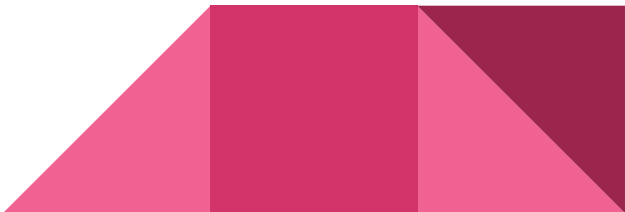
Name	Collaboration	Responsibility
Haley	Princy, Jocelyn, Tripura	Architecture Experiments, Model Evaluation
Jocelyn	Haley, Princy, Tripura	AI-Platform, TFX Pipeline Experiments
Princy	Jocelyn, Haley, Tripura	Web/App Flask Model Deployment
Tripura	Haley, Princy, Jocelyn	Research/Documentation

Taskboard: https://docs.google.com/document/d/11ZsR_8_WC8t1yW9f6PMmL8bO9QUrPch4o_-ivU3hvZU/edit



References

- [1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3156–3164, 2015.
- [2] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In International Conference on Machine Learning, page 2048–2057, 2015.
- [3] Hodosh, P. Young and J. Hockenmaier (2013) "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artificial Intelligence Research, Volume 47, pages 853-899
- [4] Karpathy, Andrej & Fei-Fei, Li. (2014). Deep Visual-Semantic Alignments for Generating Image Descriptions.
- [5] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. In arXiv:1411.2539, 2014.
- [4] https://www.tensorflow.org/tutorials/text/image_captioning



Demo

Colab Baseline model

Colab Attention model

TFX colab

Webapp

GitHub CI/CD

The background is a solid pink color. In the top right corner, there is a decorative pattern of overlapping geometric shapes, including triangles and squares, in various shades of pink and magenta.

Thank you!