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.

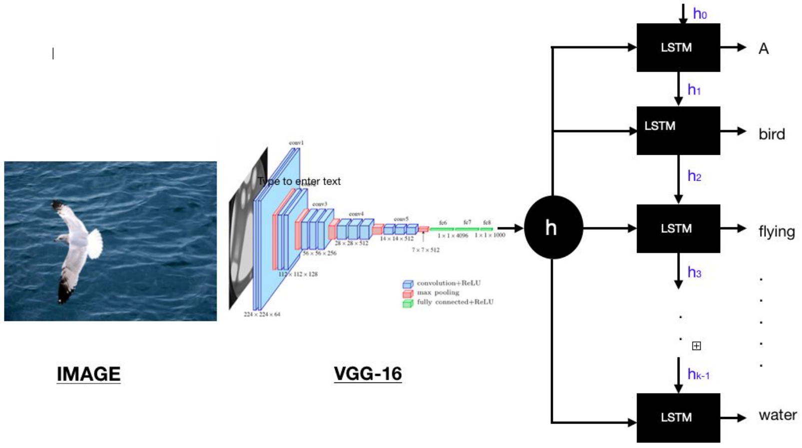
Local attention

- 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.

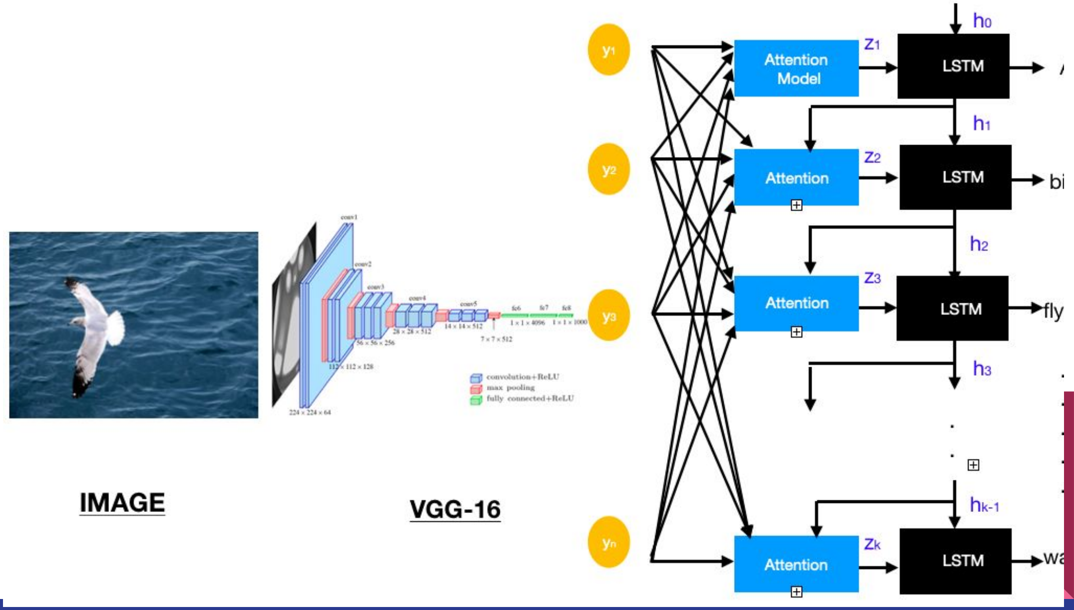
- The main part of the attention mechanism is the following two aspects: the decision needs to pay attention to which part of the input; the allocation of limited information processing resources to the important part.

Architecture - with and without Attention



Baseline model Architecture without attention

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 .

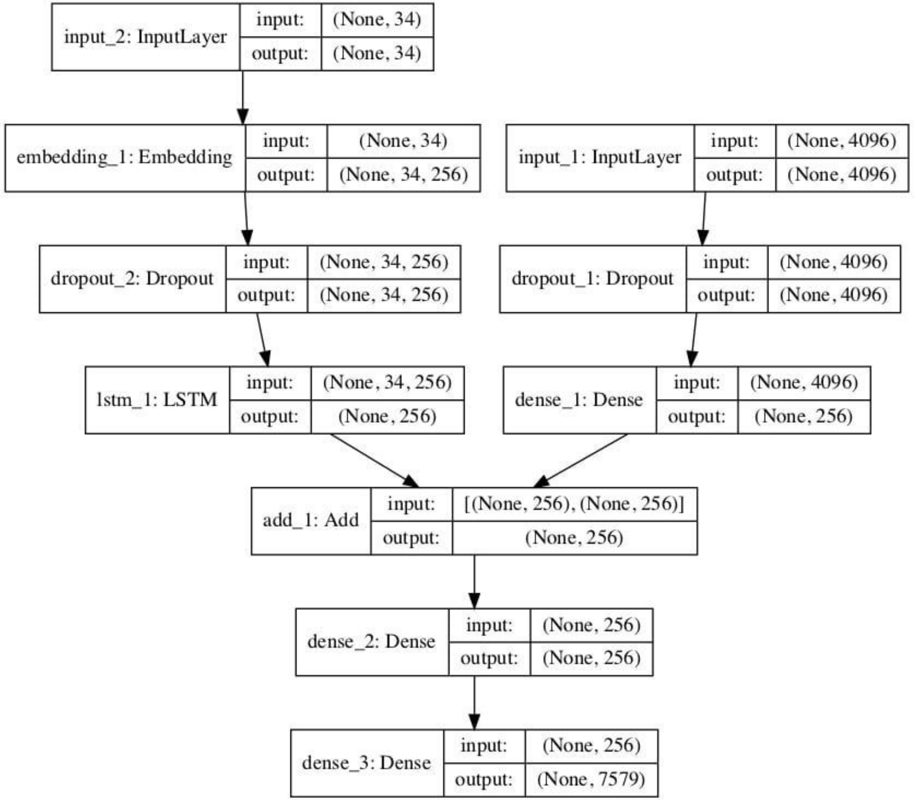
Two dogs on pavement moving toward each other .
Two dogs of different breeds looking at each other on the road .
A black dog and a white dog with brown spots are staring at each other in the street .
A black dog and a tri-colored dog playing with each other on the road .
A black dog and a spotted dog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

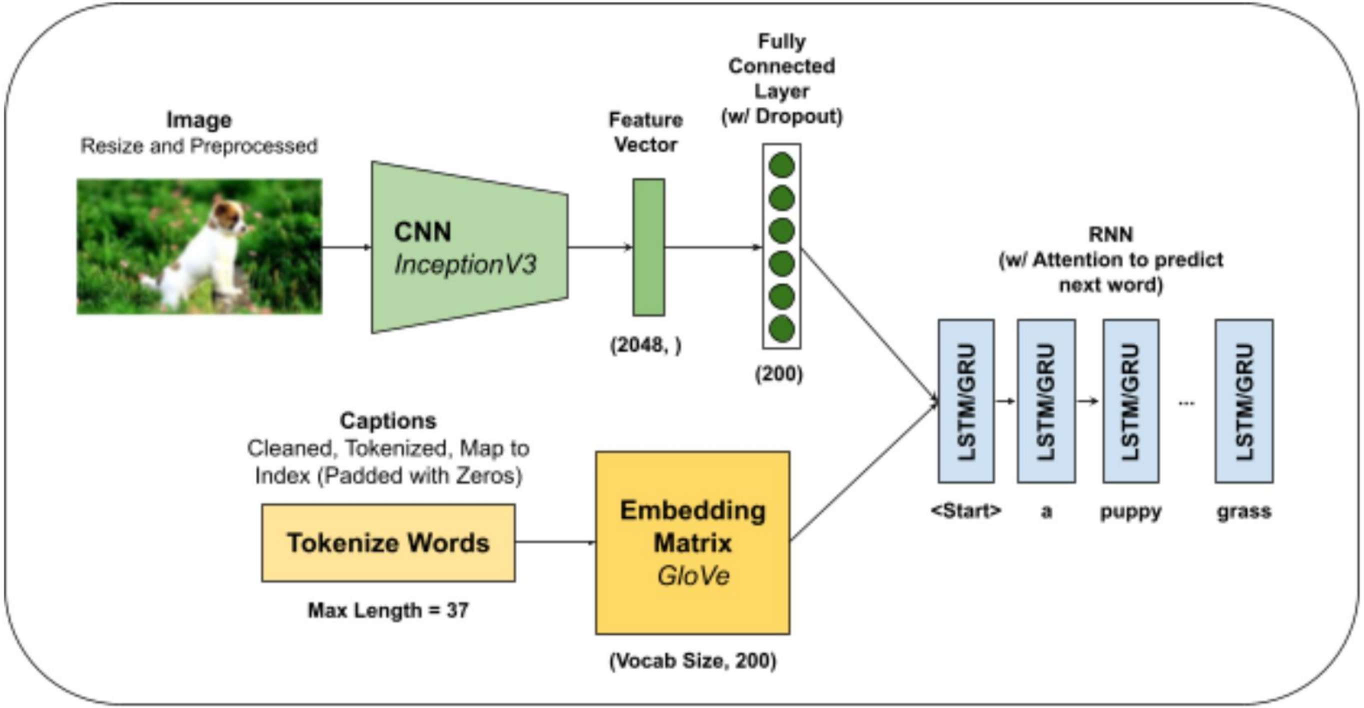


Tensorflow Subclass Model

```
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 (Batc multiple                  2048
_________________________________________________________________
dense_7 (Dense)              multiple                  262656
_________________________________________________________________
dense_8 (Dense)              multiple                  4504140
_________________________________________________________________
bahdanau_attention_1 (Bahdan 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

```python
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 initital 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

Encoder output shape: (64, 200)

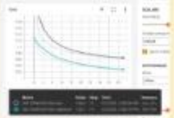
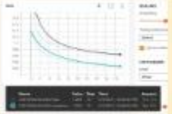
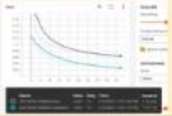
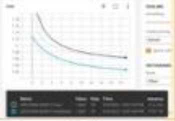
Put ground truth back to the model

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.	Recall: # of matching n-grams/# total n-grams in the reference Precision: # of matching n-grams/# of total n-grams generated caption	Substitution: number of words need to be substituted to match reference Deletion: number of words dropped from the reference Insertion: number of extra words added compared to the reference Number: 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***

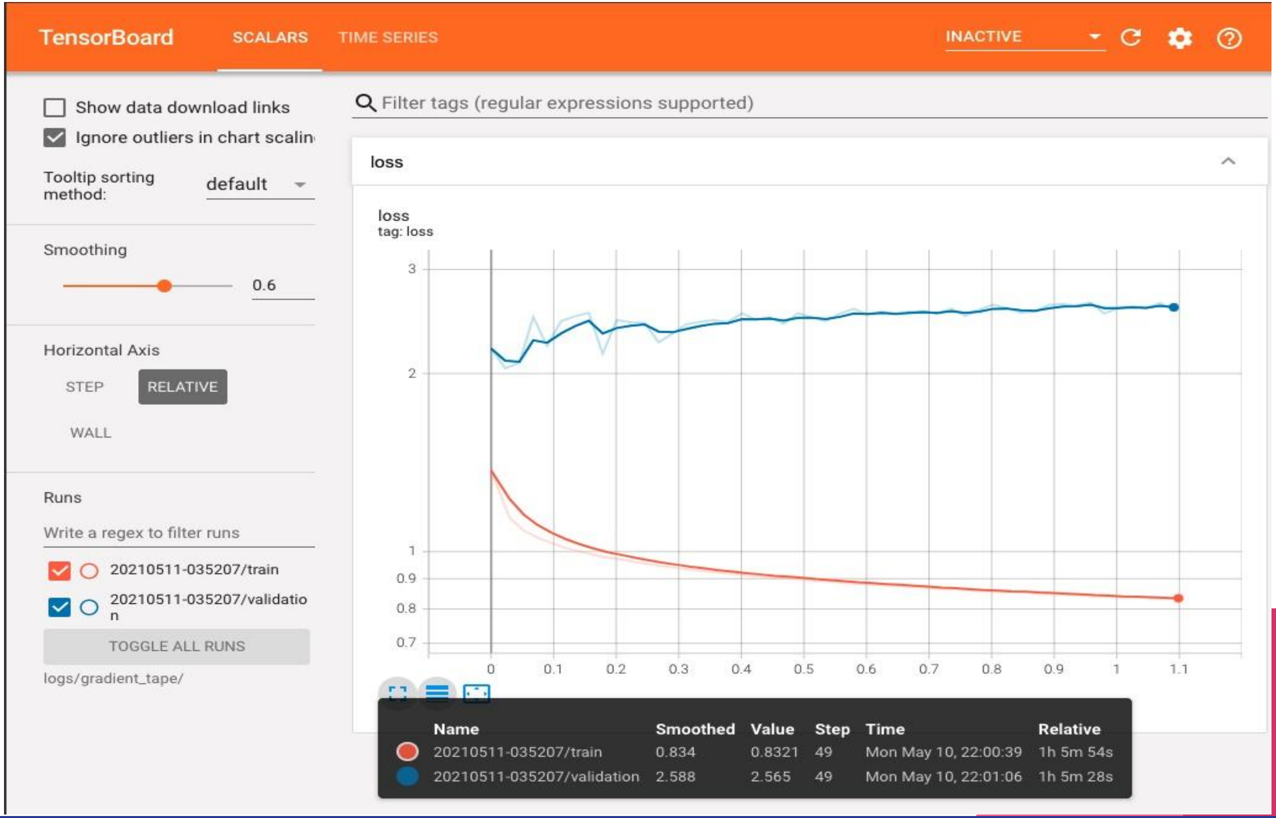
NLP Metrics Source: https://github.com/gcunhase/NLPMetrics

Experiments

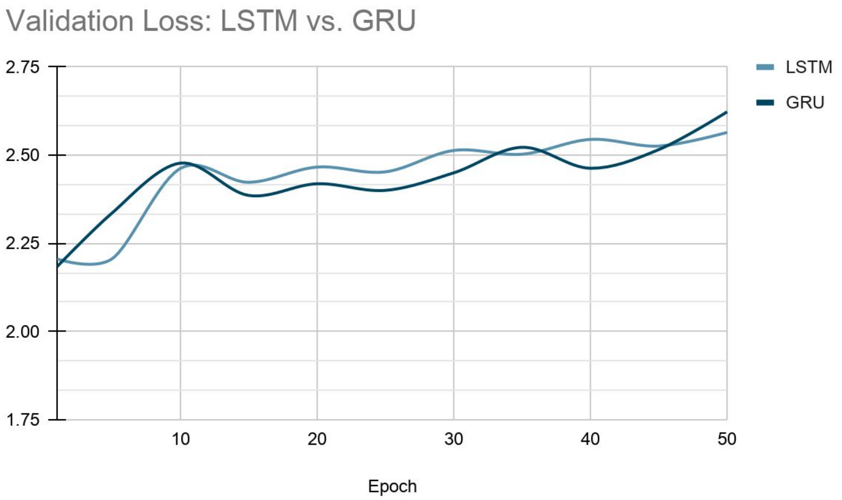
# Top Vocob	Encoder	Decoder	Optimizer	Batch, Epoch	Train Loss	Val Loss	Training Time	Batch, Epoch
top_k = 1600	Dense, Relu	Embedding,GRU, Dense, Dense, Attention	Adam	64, 20	1.064	1.027	16m47s with GPU	
top_k = 1600	Dense, Relu	Embedding, GRU, Dropout (0.5), Dense, Dense, Attention	Adam	64, 20	1.063	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	Embeddin, GRU, Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64. 50	0.838	2.623		
top_k = 5000	Dense, Dropout, Relu	Embeddin, LSTM, Dense, Dropout (0.5), BatchNorm, Dense, Attention	Adam	64. 50	0.832	2.564	57m28s	

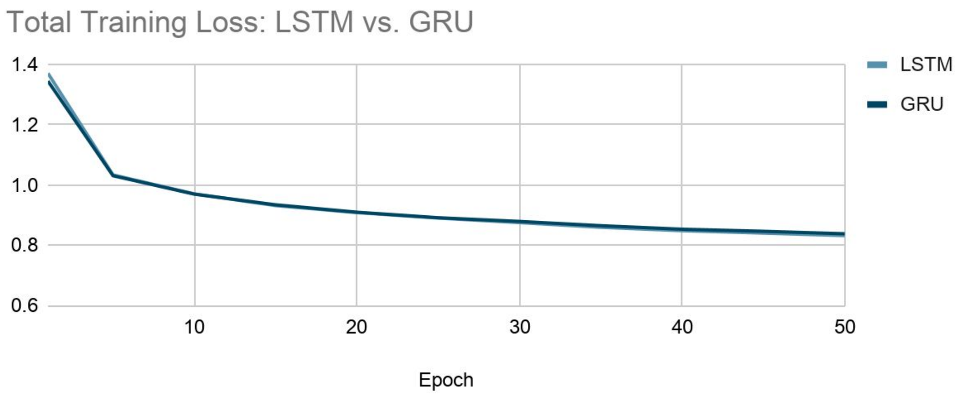
Model Tracker:
https://docs.google.com/spreadsheets/u/1/d/1VbMC_GL8O9GxUB3ui934VhXeaLtrKatFQrs-JyZHR4I/edit?usp=drive_web&ouid=109949769242133710329

Tensorboard visualization - LSTM



Total train, Validation loss





Caption Evaluation

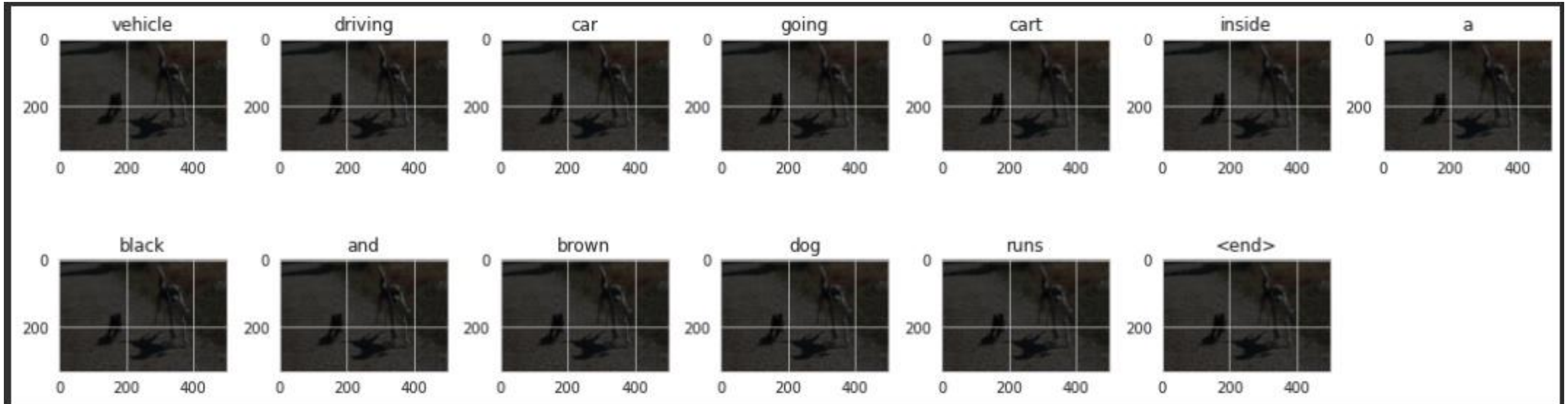
* comparison on one image

Method	Model	BLEU Score 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

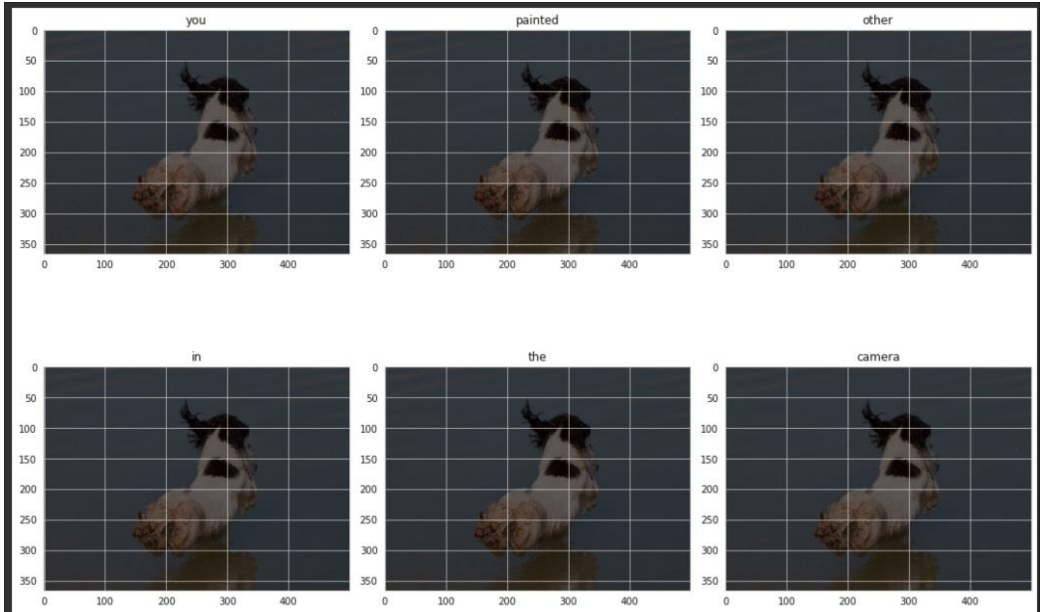
Vanilla Model Generated Caption: a man in a red shirt is standing on a rock overlooking a waterfall

Results - the okay



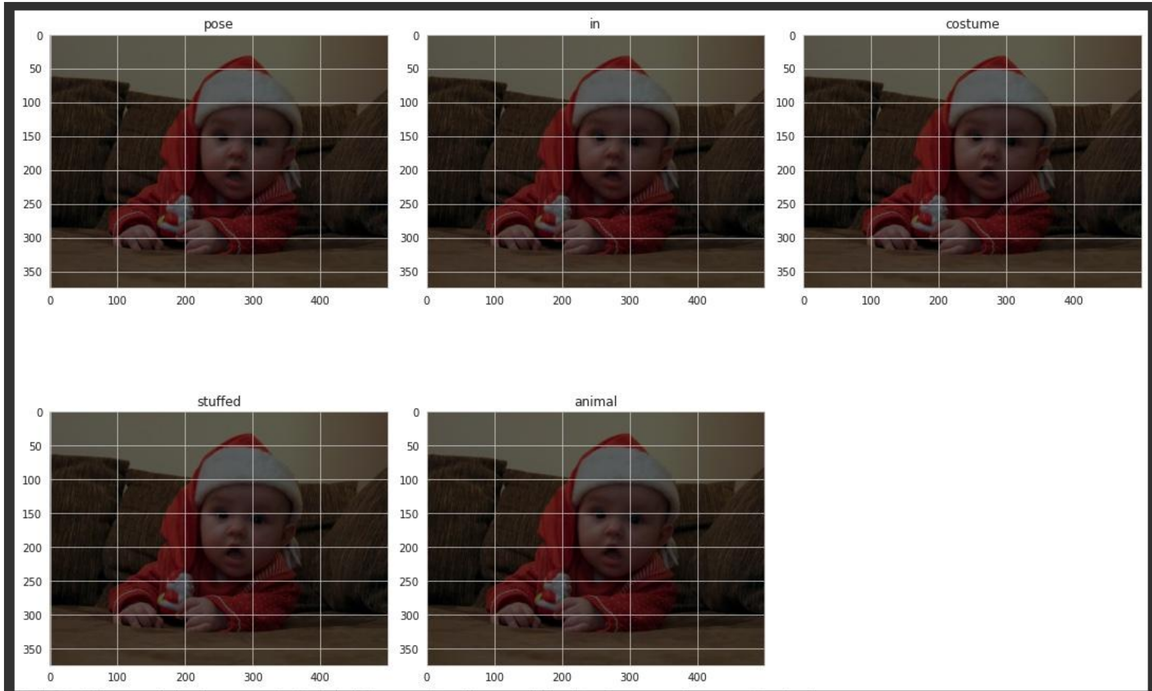

```
Real Caption: a little black dog is running beside a large brown and white dog with a large rope in its mouth <end>
Real Caption: a small black dog is running alongside a bigger tricolored dog that is carrying a rope toy in its mouth <end>
Real Caption: a small dog and a large dog with a rope in 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



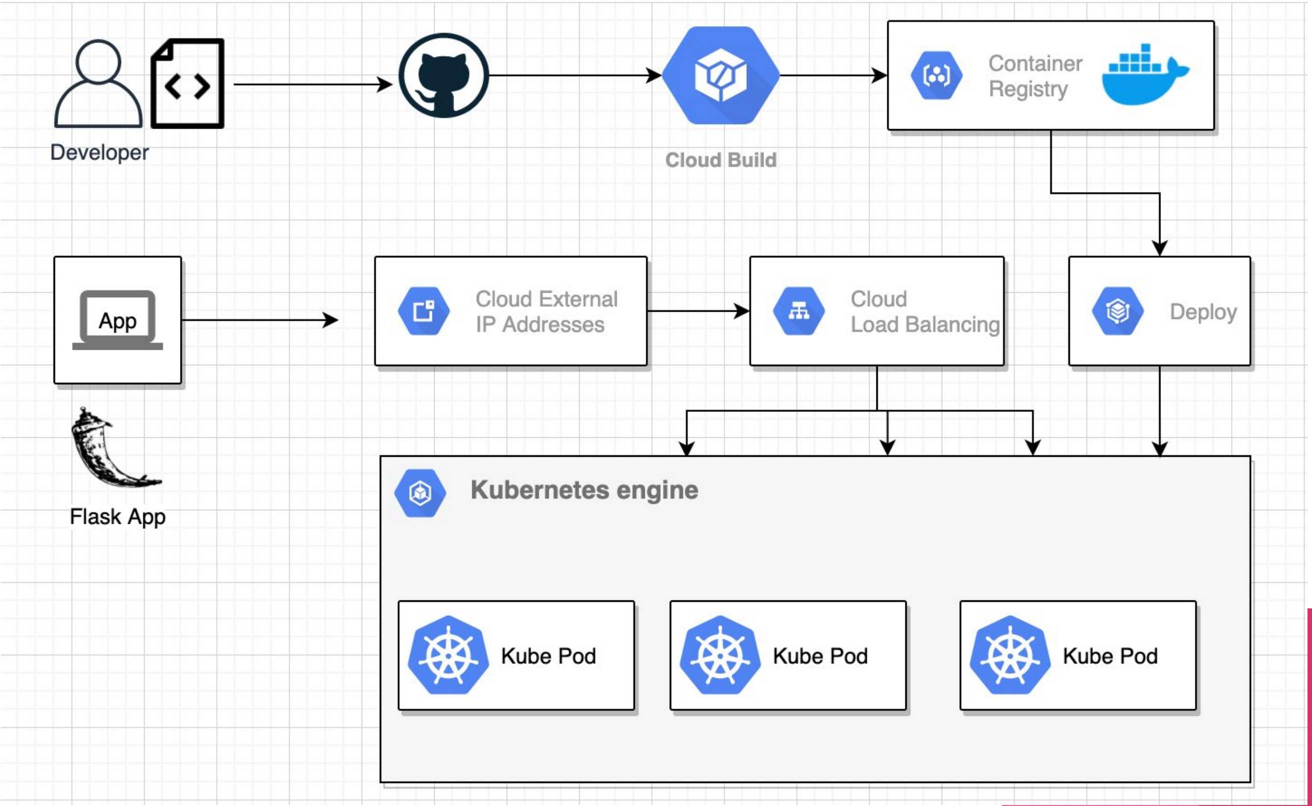

```
Real Caption: a black and white dog is jumping into the water <end>
Real Caption: a dog runs and plays in the water <end>
Real Caption: a little white and brown dog is running in the water <end>
Real Caption: a white and black dog leaps into water <end>
Real Caption: a white dog runs across the water <end>
Prediction Caption: you painted other in the camera

BLEU-1: 0.238844
BLEU-2: 0.185008
BLEU-3: 0.317986
BLEU-4: 0.364093
```

Results - the interesting

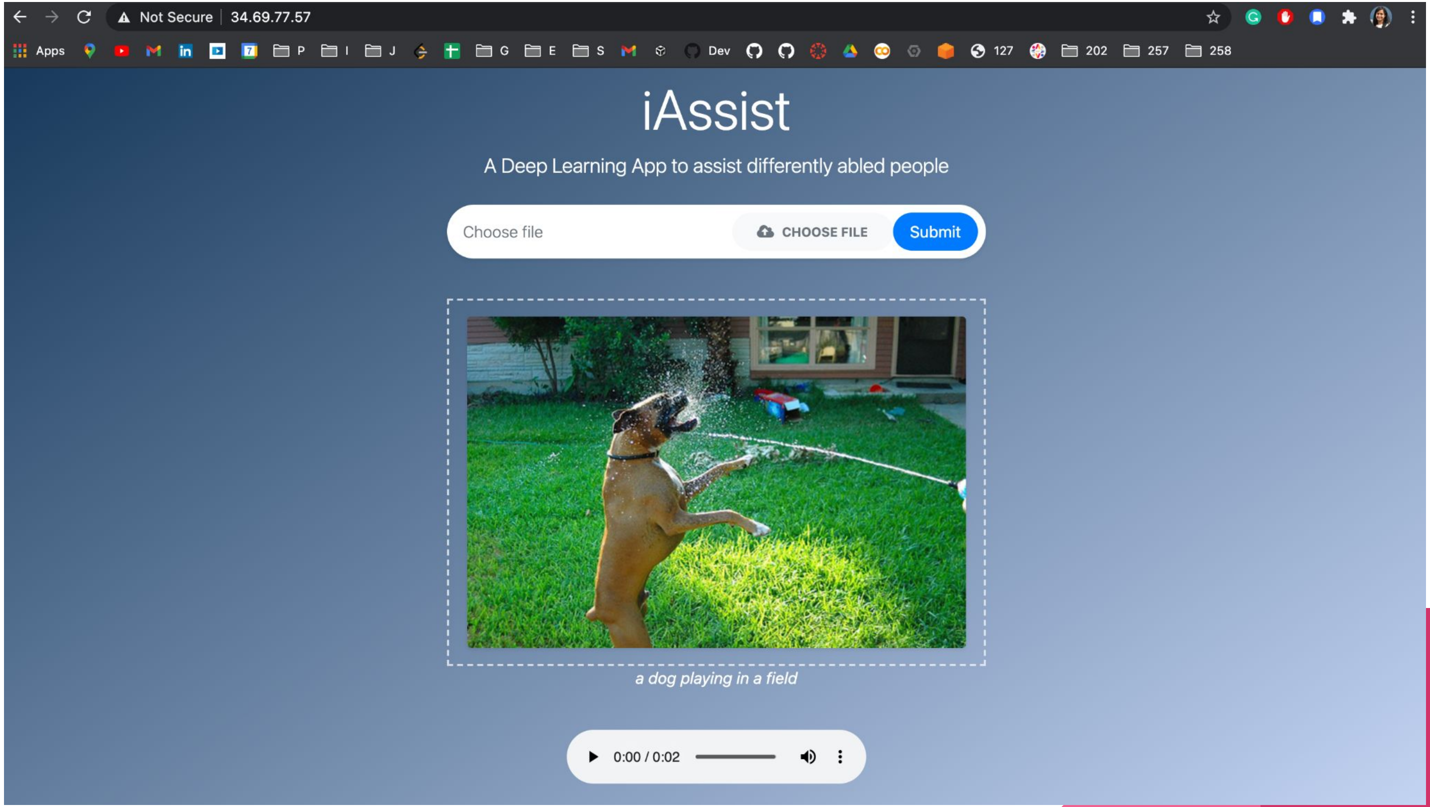



```
Real Caption: a baby in a santa hat holds a santa figure while laying on a brown sofa <end>
Real Caption: a baby wearing a santa hat and a red outfit playing with a santa claus figure <end>
Real Caption: a baby wears a santa hat <end>
Real Caption: baby in santa outfit holding a baby santa <end>
Real Caption: the baby is holding a santa <unk> <end>
Prediction Caption: pose in costume stuffed animal
```

Cloud architecture and CI/CD implementation



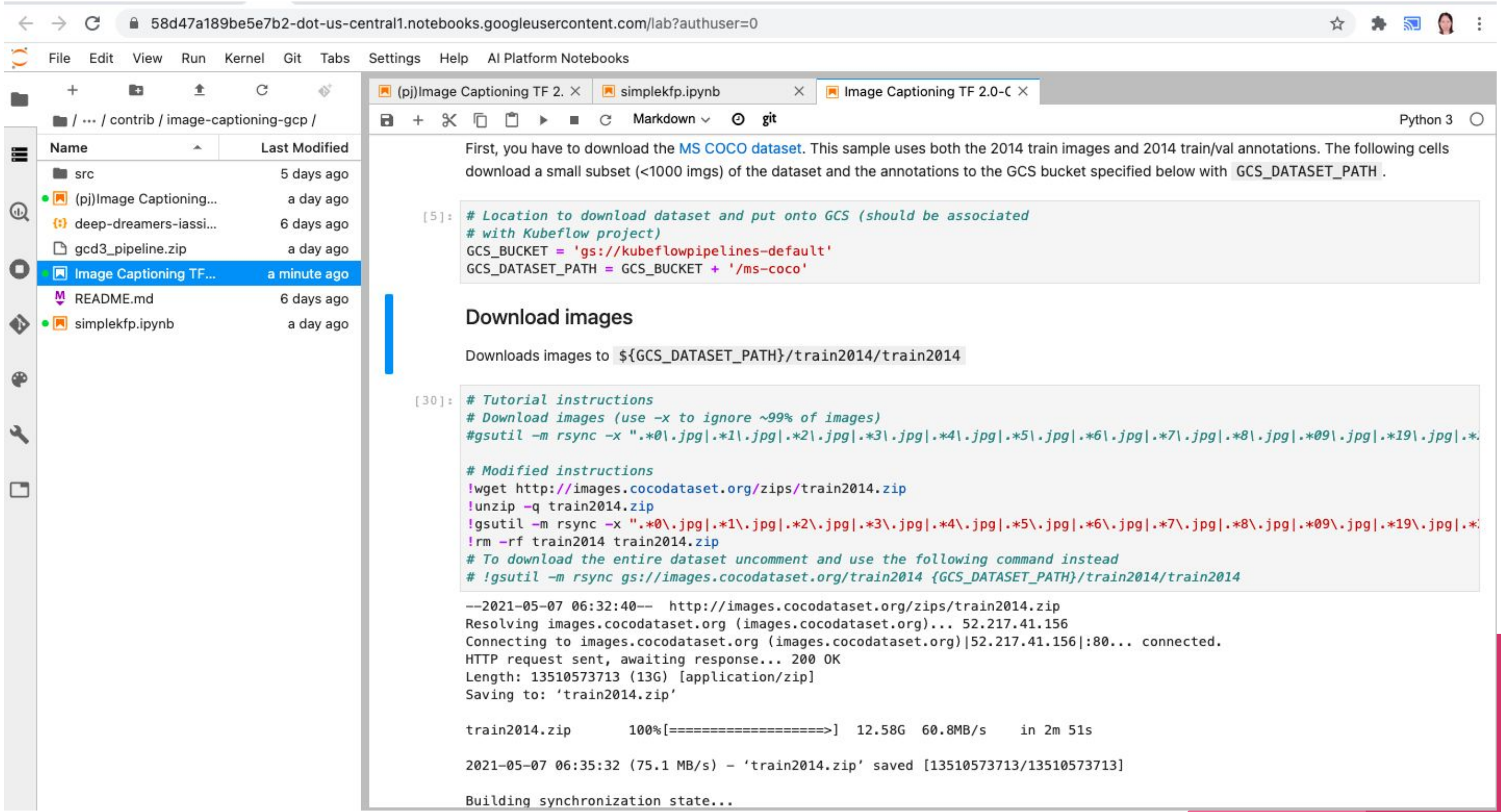
Web app inference



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.
- 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)



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

Thank you!