Show and Tell

– A Neural image caption generator

Presented by Siqi Miao

Team Members: Siqi Miao, Shicheng Zhou, Ketong Xie

Overview

1

Results

2

Experiments

3

Possible Future Improvements

Results – Dataset used

Training set:
118k images from MSCOCO 2017

Validation set:
5k images from MSCOCO 2017

Testing set:
41k images from MSCOCO 2014

- No available server found for evaluating testing set of MSCOCO 2017
- No overlap between training set of MSCOCO 2017 and testing set of MSCOCO 2014
- Testing set of MSCOCO 2017 and MSCOCO 2014 differ only ~100 images

Results – Performance of a single model

Perform better than Google's

Around 0.5%~1% off than Google's

User	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr-D	
	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40
siqimiao	0.721	0.896	0.551	0.805	0.412	0.695	0.309	0.583	0.248	0.333	0.528	0.674	0.930	0.940
Google	0.713	0.895	0.542	0.802	0.407	0.694	0.309	0.587	0.254	0.346	0.530	0.682	0.943	0.946

*Both evaluated on testing set of MSCOCO 2014

Google used ensembling models!

User	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr-D	
	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40
siqimiao	0.724	-	0.554	-	0.416	-	0.314	-	0.251	-	0.532	-	0.966	-

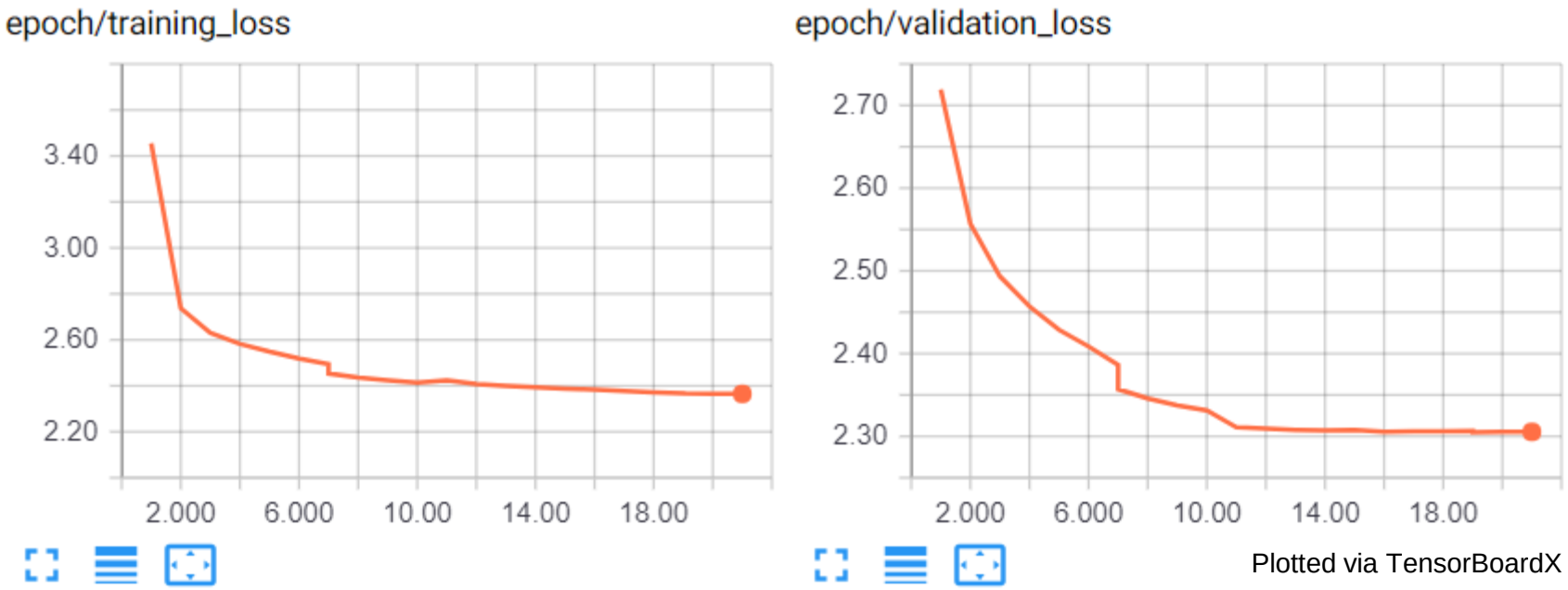
*Evaluated on validation set of MSCOCO 2017

Scores can be found at:
http://cocodataset.org/#captions-leaderboard
https://competitions.codalab.org/competitions/3221#results

Results – Details of our best model – Loss

- Trained for 21 epochs
- Best performance at 18th epoch
- 133.86 GPU hours used for this single model

Loss at 18th epoch*
- Training loss: 2.372
- Validation loss: 2.306



Plotted via TensorBoardX

*1) Have dropout layer;
2) Accumulate training loss during training (under model.trian());
3) Calculate validation loss separately (under model.eval()).

Results – Details of our best model – Implementation

1. **Data augmentation and transformation**
   - Resize every image to (224, 224) directly
   - Apply random changes via ColorJitter
   - Flip images horizontally with p=0.5
   - Apply z-score normalization

2. **Model architecture**
   - Pretrained ResNet152
     - Retrain its FC layer
     - Image embedding size = 512
   - LSTM
     - One layer
     - Hidden size = 512
     - Word embedding size = 13003x512

3. **Regularization**
   - Dropout both for CNN and RNN
   - L2 weight decay for ADAM optimizer

4. **Minor details**
   - Clip Gradients to [-5, 5]
     - To avoid exploding gradients
   - Use ADAM optimizer with learning rate decay
     - Need to take care of overflow in BWs
   - Beam search with a size of three
     - Suggested by Google for this model

Results – Details of our best model – Training procedure

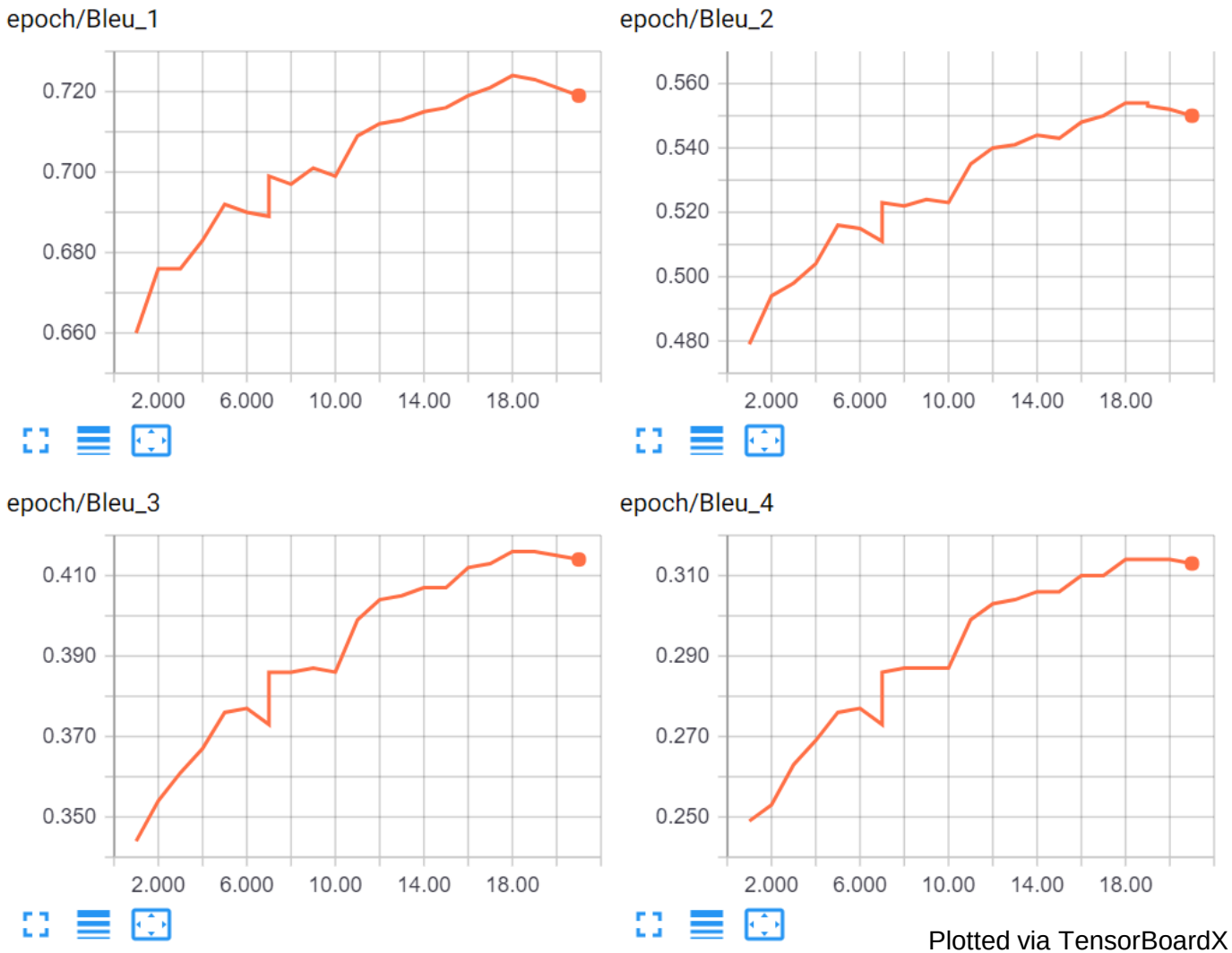
Epoch 1-6:
- Learning rate = 10^-4 = 0.0001
- Weight decay = 10^-4 = 0.0001

Epoch 7-10
- Learning rate = 10^-5 = 0.00001
- Weight decay = 10^-5 = 0.00001

Epoch 11-13
- Learning rate = 10^-6 = 0.000001
- Weight decay = 10^-6 = 0.000001

Epoch 14-18
- Learning rate = 10^-6 = 0.000001
- Weight decay = 0

Epoch 19-21
- Learning rate = 10^-7 = 0.0000001
- Weight decay = 0

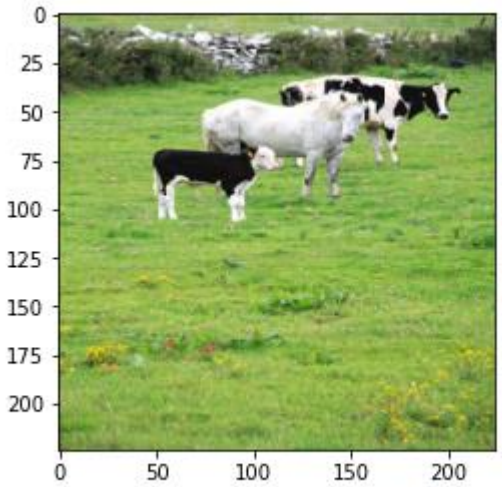


Plotted via TensorBoardX

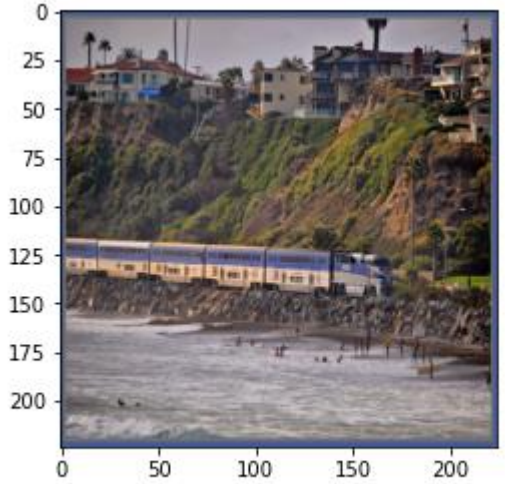
a) Epoch 1-10: Freeze weights of ResNet152 except the FC layers; batch size = 128; ~4h/epoch
b) Epoch 11-21: Unfreeze weights and fine-tune CNN; batch size = 16; ~13h/epoch
c) Train LSTM and Word Embedding layers all the time

Results – Some resulting captions

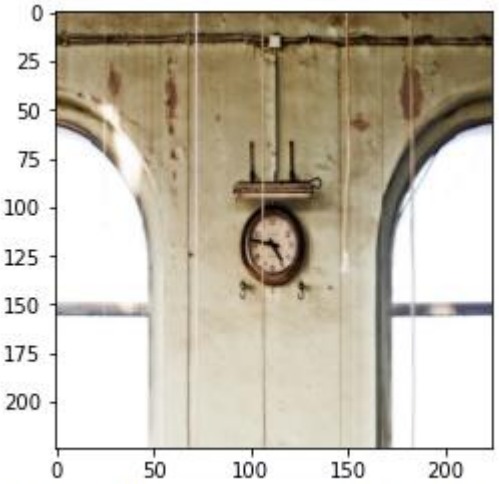
beam size = 3; randomly selected from testing set



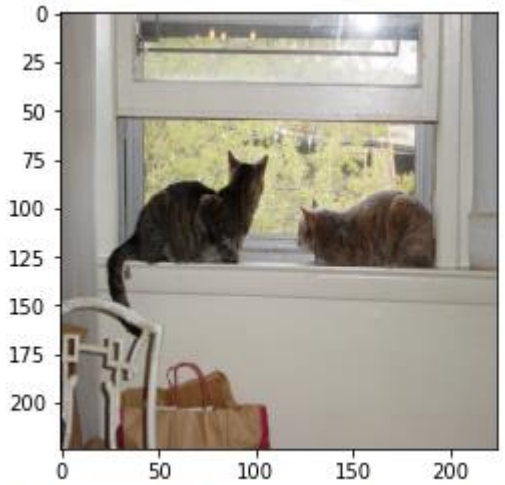
STK a group of cows standing in a field . EDK



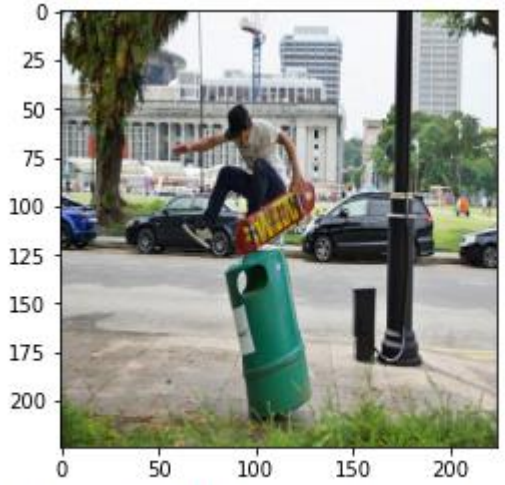
STK a large long train on a steel track . EDK



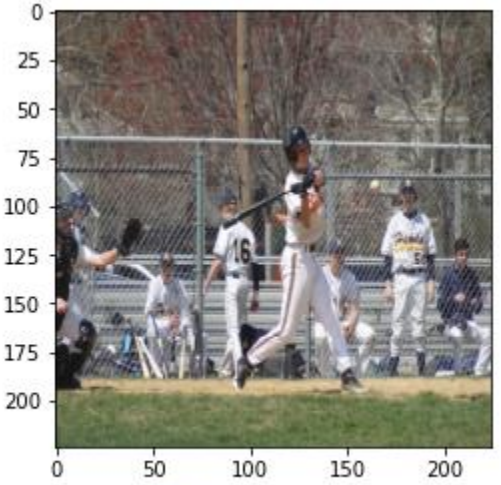
STK a clock on the side of a building . EDK



STK a cat sitting on top of a window sill . EDK



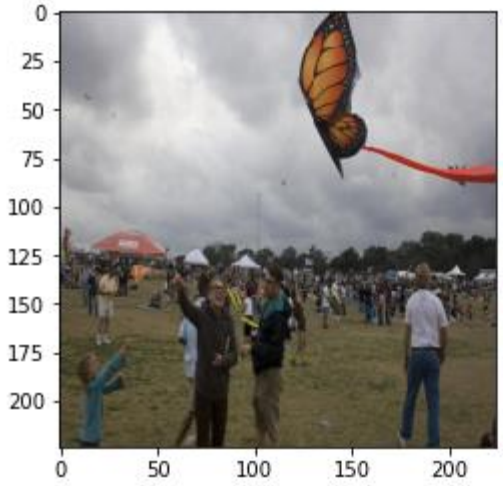
STK a man is doing a trick on a skateboard . EDK



STK a baseball player swinging a bat at a ball EDK

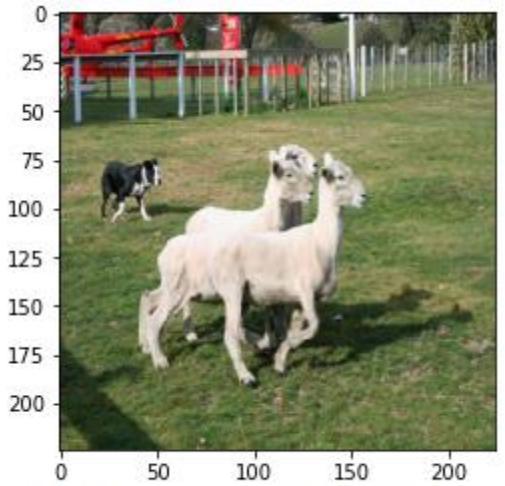
Results – Some resulting captions

beam size = 3; randomly selected from testing set

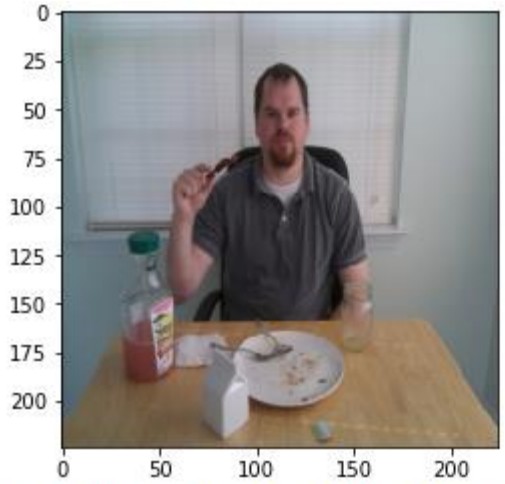


STK a group of people flying kites in the sky . EDK

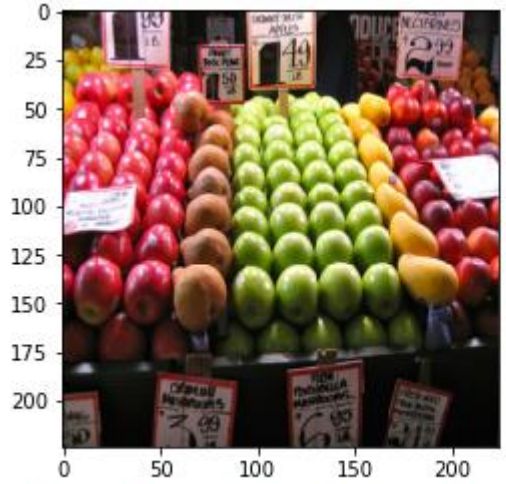
STK a hotel room with a bed and a window EDK



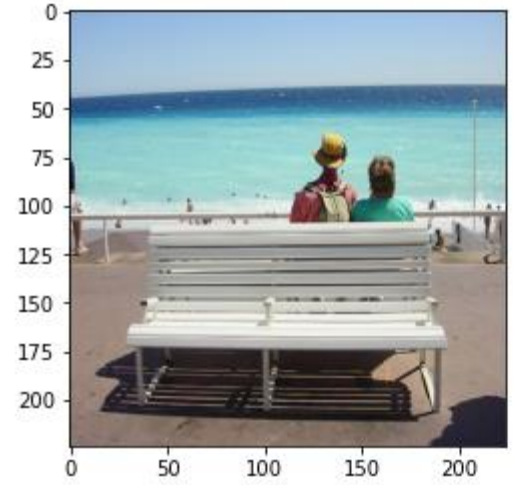
STK a group of sheep standing on top of a lush green field . EDK



STK a man sitting at a table with a plate of food . EDK



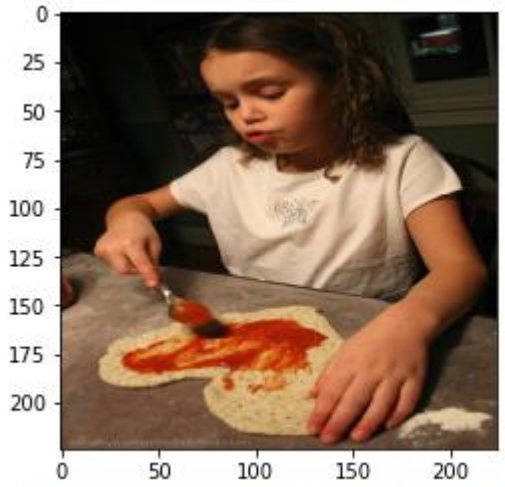
STK a variety of fruits and vegetables on display . EDK



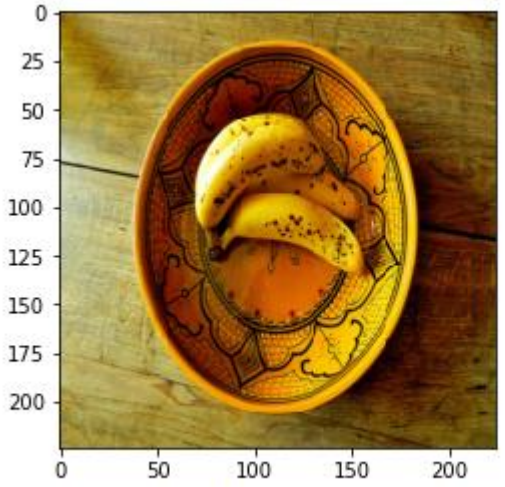
STK a couple of people sitting on a bench in the sand . EDK

Results – Some resulting captions

beam size = 3; randomly selected from testing set

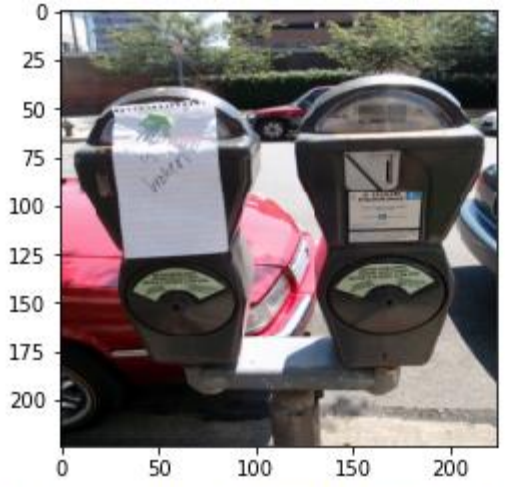


STK a little girl eating a slice of pizza . EDK

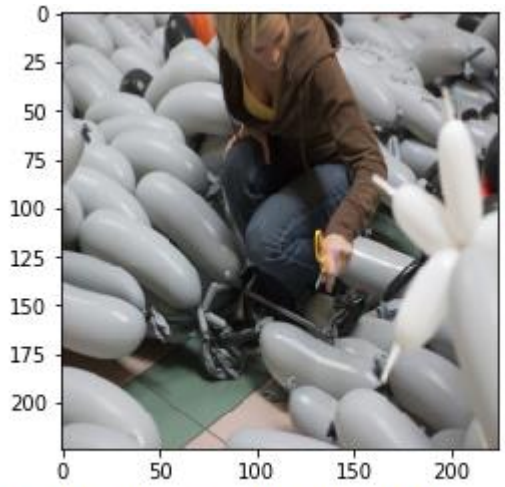


STK a banana sitting on top of a white plate . EDK

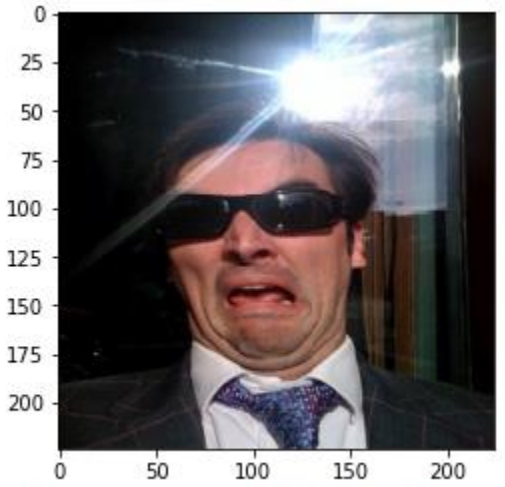
STK two horses pulling a carriage down a street . EDK



STK two parking meters sitting next to each other . EDK



STK a woman is sitting on a toilet seat . EDK



STK a man in a suit and tie posing for a picture . EDK

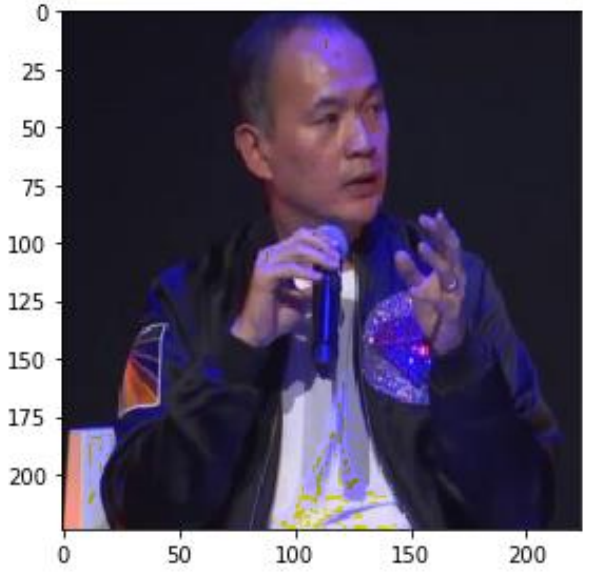
Experiments – Pretrained word embeddings?

Model 1 (with weights of ResNet and Word Embeddings freezed)

- SGD optimizer
- ResNet101
- Pretrained GloVe word embedding with size = 17003x300
- No regularization techniques used
- Trained for 22 epochs with ~66 hours used

Results

- Overfitted easily
- Generated captions are bad but readable
- Pretrained word embeddings may not help
    - The initial learning rate of SGD can be as high as 2.0




```
In [18]: print(' '.join(caption[0]))
a man holding a nintendo wii controller in a living room .
```

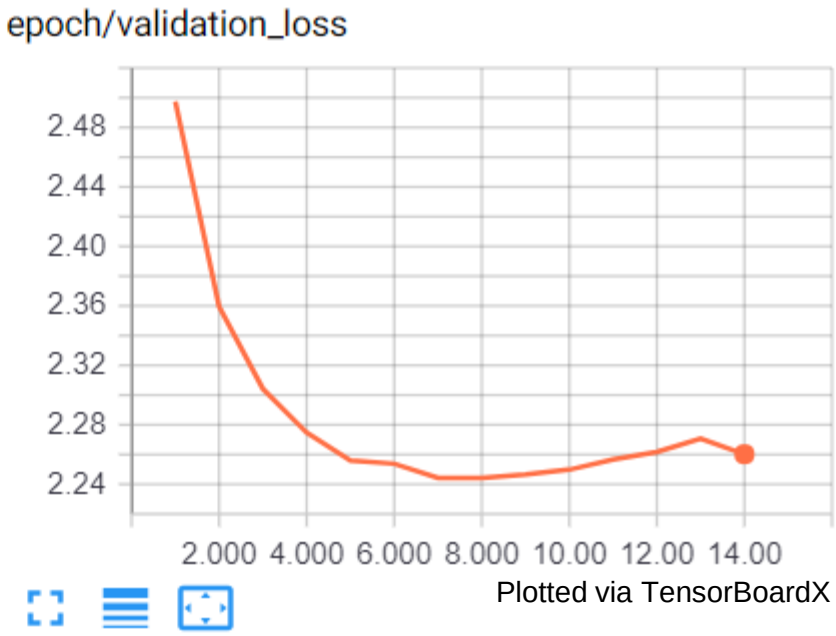
Experiments – No regularization?

Model 2 (with weights of ResNet freezed)

- ADAM optimizer
- ResNet101
- Randomly initialized word embeddings with size = 13003x512
- No regularization techniques used
- Trained for 14 epochs with ~45 GPU hours used

Results

- Overfitted extremely fast
- Results are OK on valset
  - BLEU-4 = ~0.270



Plotted via TensorBoardX

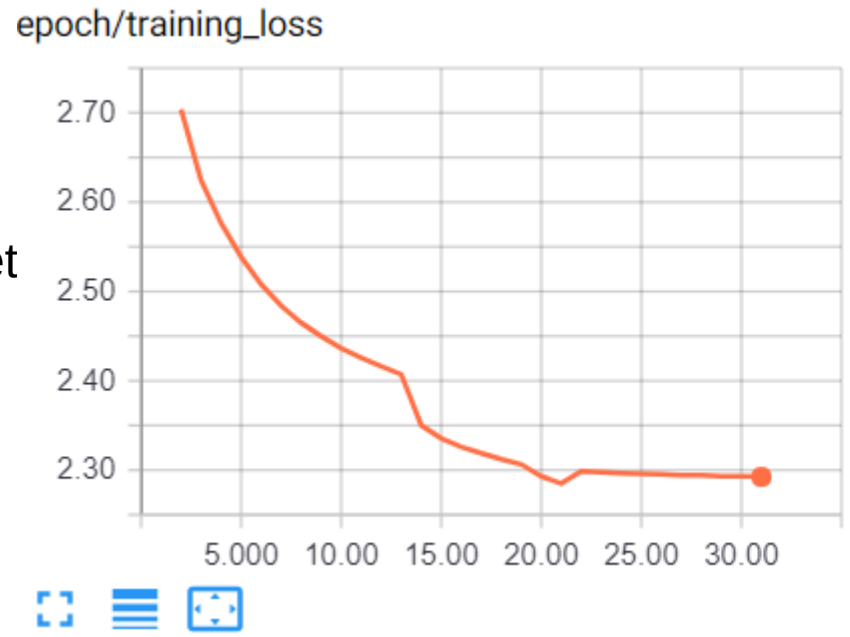
Experiments – ResNet101?

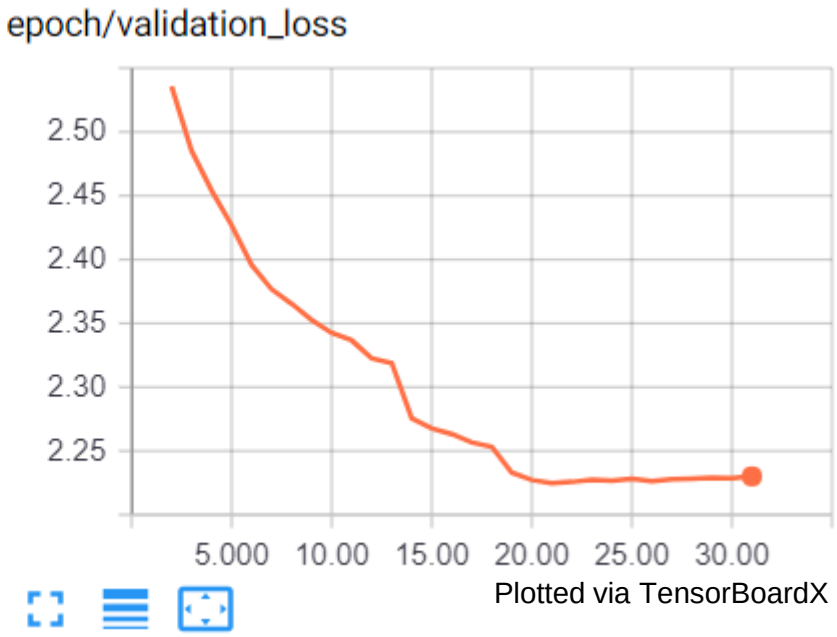
Model 3 (with similar training procedure of our best model)

- ADAM optimizer
- ResNet101
- Randomly initialized word embeddings with size = 13003x512
- Dropout and L2 weight decay added
- Trained for 31 epochs with ~142 GPU hours used

Results

- No obvious overfitting
- Results are good on valset
  - BLEU-4 = ~0.307





Plotted via TensorBoardX

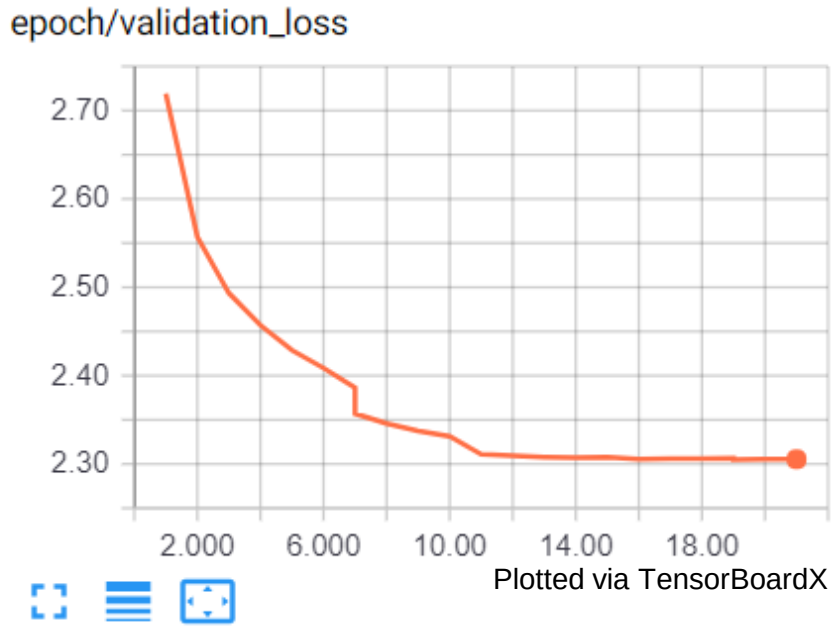
Experiments – Our best model

Model 4 (with the training procedure introduced before)

- ADAM optimizer
- ResNet151
- Randomly initialized word embeddings with size = 13003x512
- Dropout and L2 weight decay added
- Trained for 21 epochs with ~134 GPU hours used

Results

- No obvious overfitting
- Results are better on valset
  - BLEU-4 = ~0.314



Plotted via TensorBoardX

Experiments – Conclusion

Keys of training

- Randomly initialized word embedding
- Dropout and L2 weight decay
- ADAM optimizer with learning rate decay
- Train RNN first then train both CNN and RNN

Dropout for CNN

```
if dropout_prob:
    self.fc_output = nn.Sequential(
            nn.Linear(pretrained_cnn.fc.in_features, pretrained_cnn.fc.in_features),
            nn.ReLU(),
            nn.BatchNorm1d(pretrained_cnn.fc.in_features),
            nn.Dropout(p=dropout_prob),

            nn.Linear(pretrained_cnn.fc.in_features, no_word_embeddings)
            )
else:
    self.fc_output = nn.Linear(pretrained_cnn.fc.in_features, no_word_embeddings)
```

Dropout for RNN

```
if dropout_prob:
    self.fc_output = nn.Sequential(
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.BatchNorm1d(hidden_size),
            nn.Dropout(p=dropout_prob),

            nn.Linear(hidden_size, vocab_size)
            )
else:
    self.fc_output = nn.Linear(hidden_size, vocab_size)
```

Possible Future Improvements

a) Try ensembling models

b) Add attention systems

Thank You!