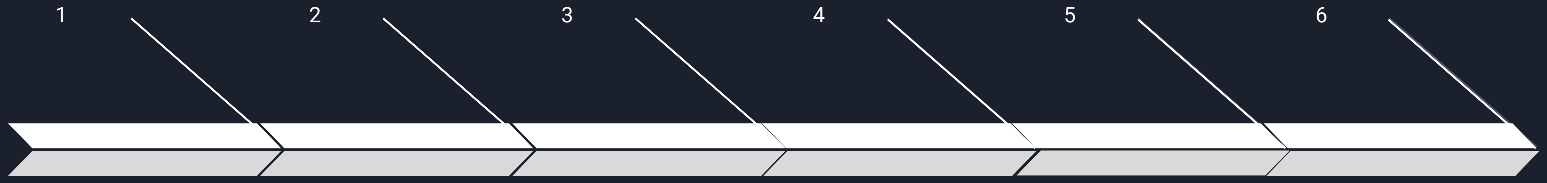
Image Captioning with Conditioned LSTM Generators

Group 7: Lichun He / Shanyue Zeng / Shiqi Tang / Huiying Wang / Jiuru Wang

Outline



1 **Background**

2 **Dataset**

flickr8k dataset

3 **Process**

4 **Baseline**

VGG16

5 **Model 2&3**

InceptionResNetV2 & Inception V3 w/ Conditioned LSTM Generators

6 **Evaluation**

BLEU scores comparison

Background

Image captioning, automatically generating natural language descriptions according to the content observed in an image, is an important part of scene understanding, which combines the knowledge of computer vision and natural language processing.

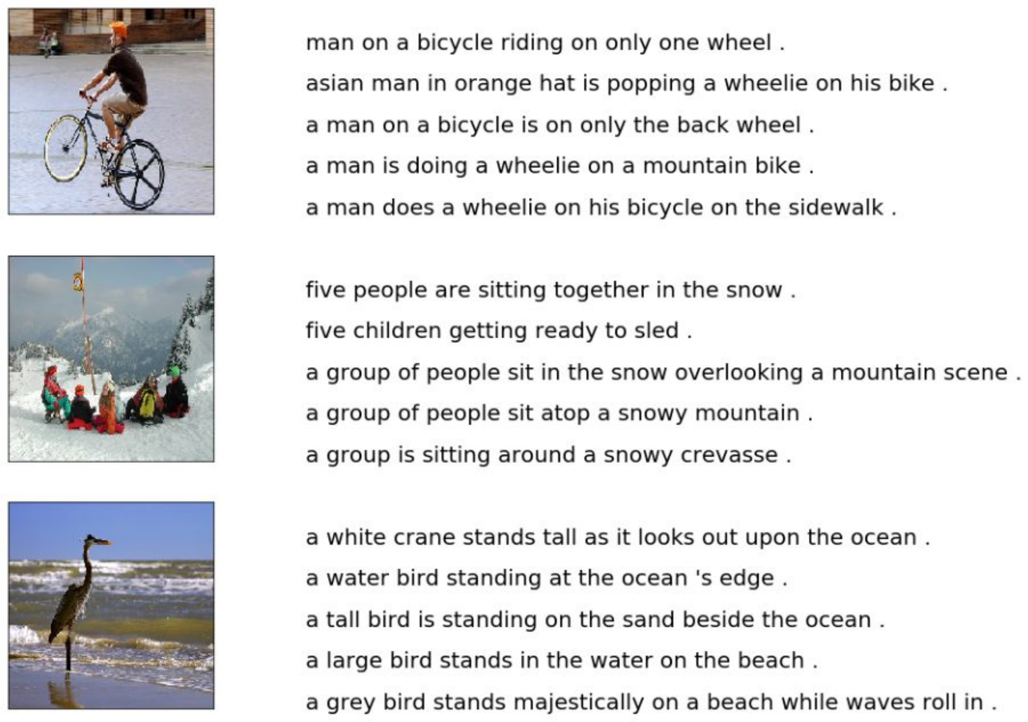
Image captioning is a challenging task where computer vision and natural language processing both play a part to generate captions. This technology can be used in many new fields like helping visually impaired, medical image analysis, geospatial image analysis etc.

Dataset: Flickr8k

A new benchmark collection for sentence based image description and search.

This dataset includes around 8,000 images along with 5 different captions written by different people for each image. The images are all contained together while caption text file has captions along with the image number appended to it.

In our project, 6000 are used for training, 1000 for test and 1000 for development.

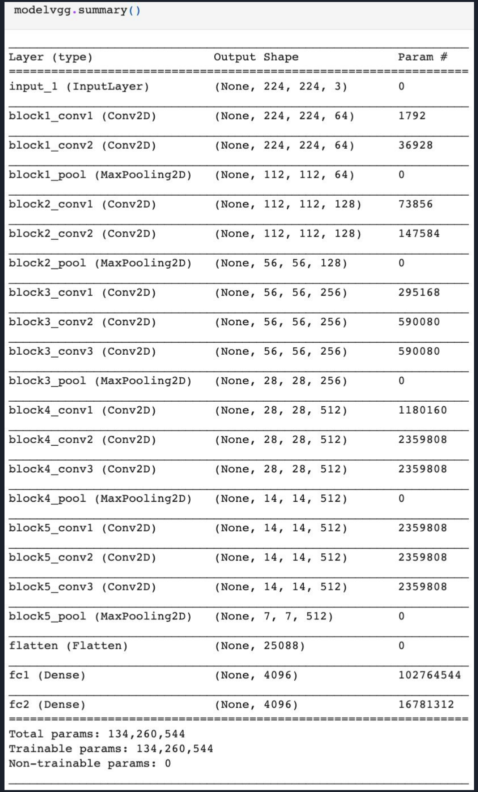


man on a bicycle riding on only one wheel .
asian man in orange hat is popping a wheelie on his bike .
a man on a bicycle is on only the back wheel .
a man is doing a wheelie on a mountain bike .
a man does a wheelie on his bicycle on the sidewalk .

five people are sitting together in the snow .
five children getting ready to sled .
a group of people sit in the snow overlooking a mountain scene .
a group of people sit atop a snowy mountain .
a group is sitting around a snowy crevasse .

a white crane stands tall as it looks out upon the ocean .
a water bird standing at the ocean 's edge .
a tall bird is standing on the sand beside the ocean .
a large bird stands in the water on the beach .
a grey bird stands majestically on a beach while waves roll in .

1. 0 Baseline: Extracting features from images using: **VGG-16**



```
modelvgg.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0         
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544 
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312  
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

- Instead of using the pre-trained model for image classification as it was intended to be used. We just use it for extracting the features from the images. In order to do that we need to get rid of the last output layer from the model. The model then generates 4096 features from taking images of size (224,224,3).

1. 1 Extract features using: **Inception-ResNet-V2**

Inception-ResNet-V2 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories, such as the keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299, and the output is a list of estimated class probabilities.

More detailed information on the model can be found here: Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

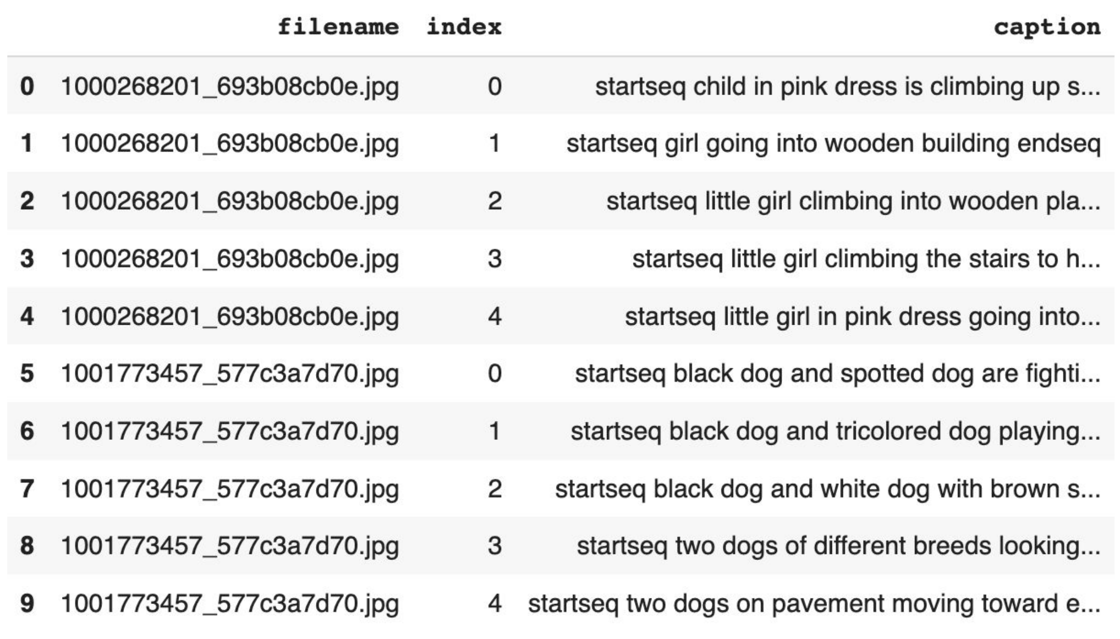
1. 2 Pre-trained image encoder: the **Inception V3 network**

Inception V3 incorporated all of the exploration for Inception V2, and in addition used the following:

- RMSProp Optimizer.
- Factorized 7x7 convolutions.
- BatchNorm in the Auxillary Classifiers.
- Label Smoothing (A type of regularizing component added to the loss formula that prevents the network from becoming too confident about a class. Prevents over fitting).

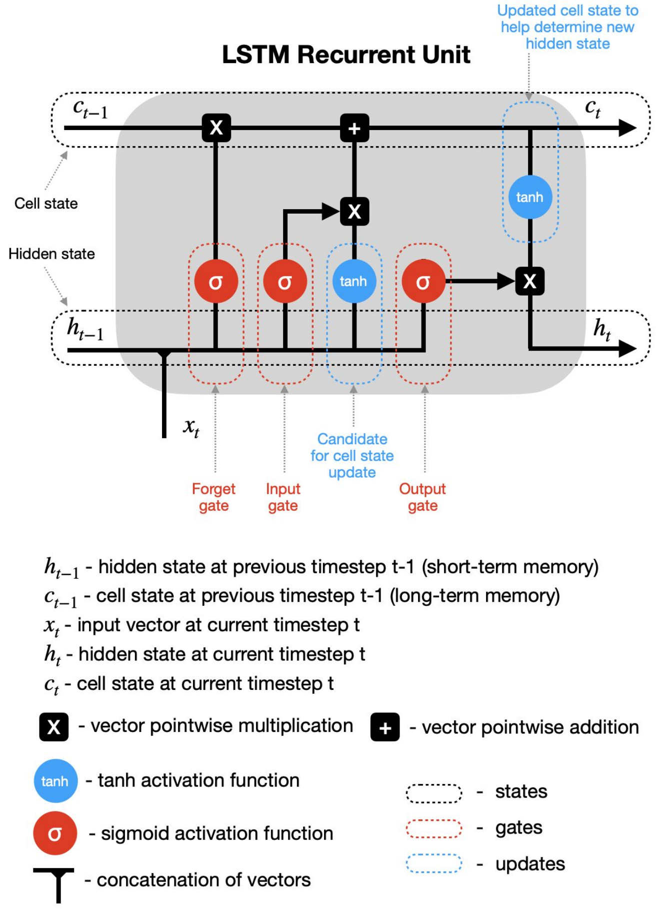
2. Merging the captions and images

Next, we need to load the image captions and generate training data for the generator model.

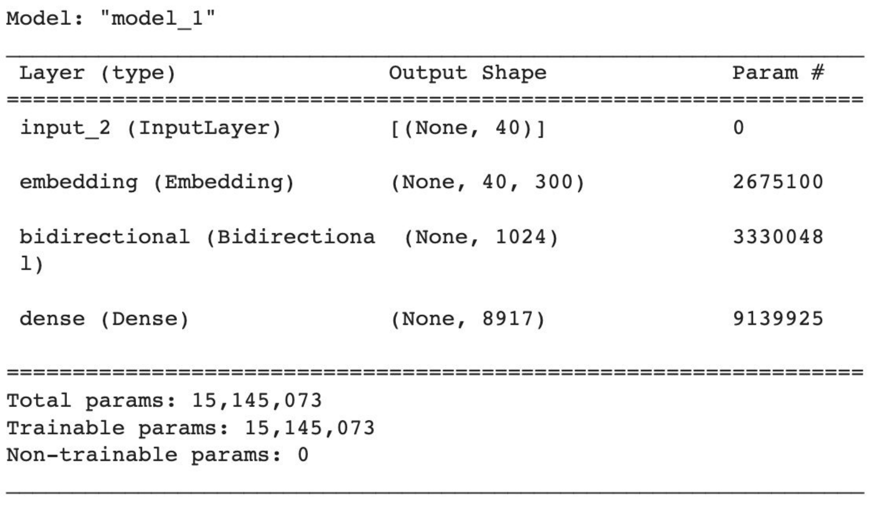


	filename	index	caption
0	1000268201_693b08cb0e.jpg	0	startseq child in pink dress is climbing up s...
1	1000268201_693b08cb0e.jpg	1	startseq girl going into wooden building endseq
2	1000268201_693b08cb0e.jpg	2	startseq little girl climbing into wooden pla...
3	1000268201_693b08cb0e.jpg	3	startseq little girl climbing the stairs to h...
4	1000268201_693b08cb0e.jpg	4	startseq little girl in pink dress going into...
5	1001773457_577c3a7d70.jpg	0	startseq black dog and spotted dog are fighti...
6	1001773457_577c3a7d70.jpg	1	startseq black dog and tricolored dog playing...
7	1001773457_577c3a7d70.jpg	2	startseq black dog and white dog with brown s...
8	1001773457_577c3a7d70.jpg	3	startseq two dogs of different breeds looking...
9	1001773457_577c3a7d70.jpg	4	startseq two dogs on pavement moving toward e...

3. Build LSTM model for training

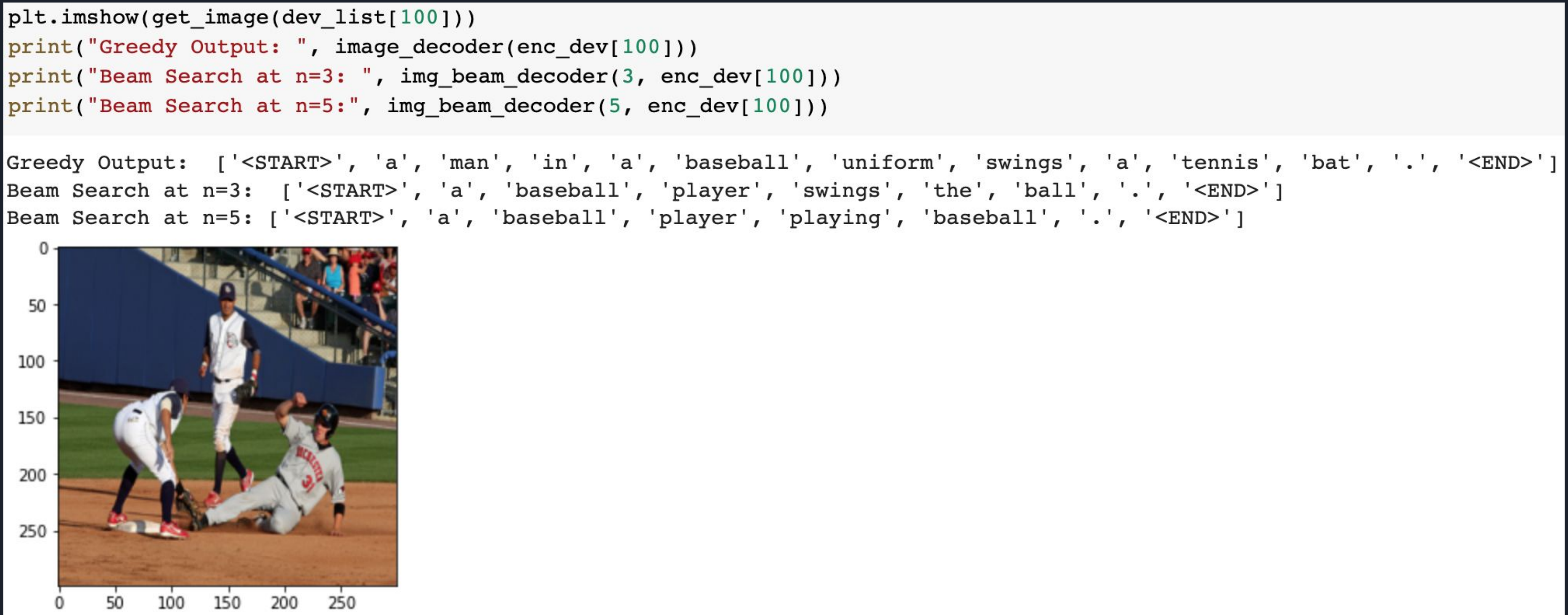

LSTM Recurrent Unit
Updated cell state to help determine new hidden state
c_{t-1}
c_t
Cell state
Hidden state
X
+
X
X
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
Forget gate
Input gate
Candidate for cell state update
Output gate
h_{t-1} - hidden state at previous timestep t-1 (short-term memory)
c_{t-1} - cell state at previous timestep t-1 (long-term memory)
x_t - input vector at current timestep t
h_t - hidden state at current timestep t
c_t - cell state at current timestep t
X - vector pointwise multiplication
+ - vector pointwise addition
tanh - tanh activation function
σ - sigmoid activation function
- concatenation of vectors
- states
- gates
- updates

- LSTM model is used because it takes into consideration the state of the previous cell's output and the present cell's input for the current output. This is useful while generating the captions for the images.
- We will use the model to predict one word at a time, given a partial sequence. For example, given the sequence ["START","a"], the model might predict "dog" as the most likely word.



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 40)]              0

 embedding (Embedding)       (None, 40, 300)           2675100

 bidirectional (Bidirectiona  (None, 1024)             3330048
 l)

 dense (Dense)               (None, 8917)              9139925

=================================================================
Total params: 15,145,073
Trainable params: 15,145,073
Non-trainable params: 0
_________________________________________________________________
```

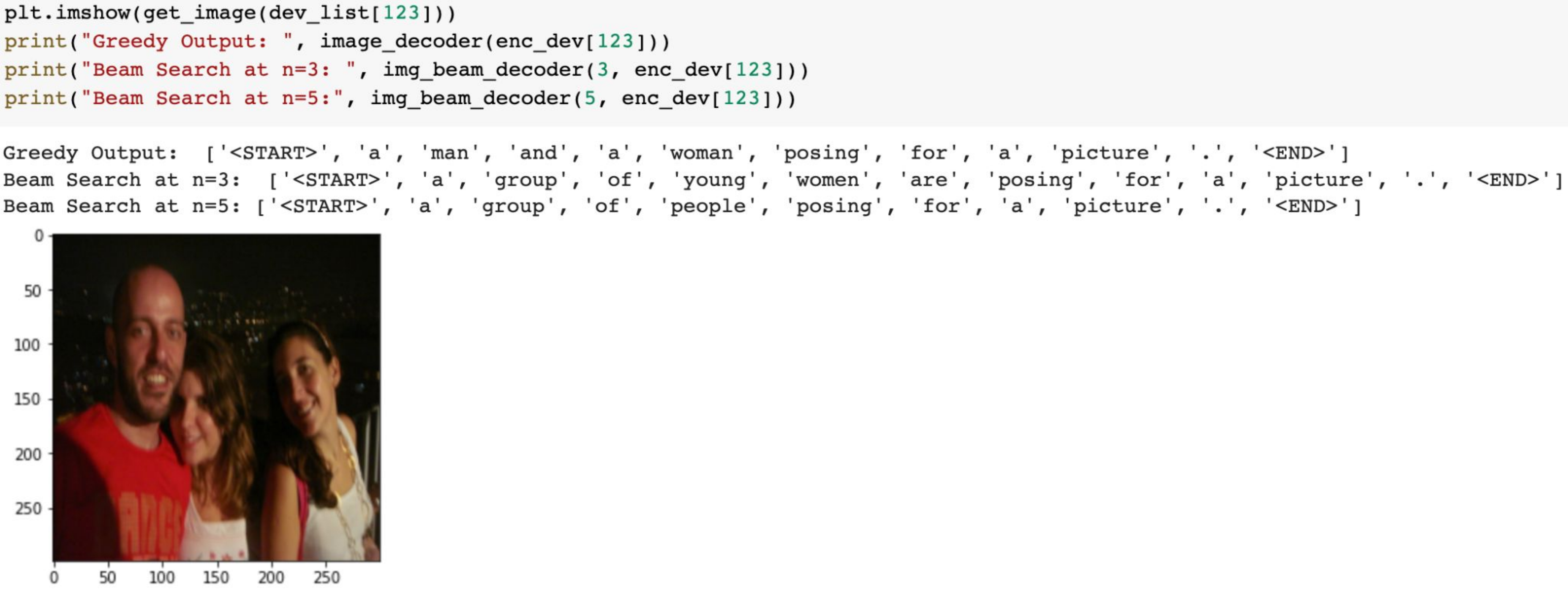
4. Prediction on Dataset



```python
plt.imshow(get_image(dev_list[100]))
print("Greedy Output: ", image_decoder(enc_dev[100]))
print("Beam Search at n=3: ", img_beam_decoder(3, enc_dev[100]))
print("Beam Search at n=5:", img_beam_decoder(5, enc_dev[100]))
```

```
Greedy Output:  ['<START>', 'a', 'man', 'in', 'a', 'baseball', 'uniform', 'swings', 'a', 'tennis', 'bat', '.', '<END>']
Beam Search at n=3:  ['<START>', 'a', 'baseball', 'player', 'swings', 'the', 'ball', '.', '<END>']
Beam Search at n=5: ['<START>', 'a', 'baseball', 'player', 'playing', 'baseball', '.', '<END>']
```

4. Prediction on Dataset cont.



```
plt.imshow(get_image(dev_list[123]))
print("Greedy Output: ", image_decoder(enc_dev[123]))
print("Beam Search at n=3: ", img_beam_decoder(3, enc_dev[123]))
print("Beam Search at n=5:", img_beam_decoder(5, enc_dev[123]))
```

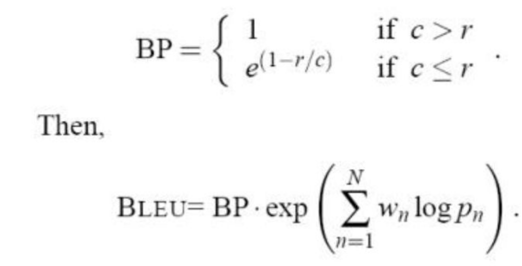
```
Greedy Output:  ['<START>', 'a', 'man', 'and', 'a', 'woman', 'posing', 'for', 'a', 'picture', '.', '<END>']
Beam Search at n=3:  ['<START>', 'a', 'group', 'of', 'young', 'women', 'are', 'posing', 'for', 'a', 'picture', '.', '<END>']
Beam Search at n=5: ['<START>', 'a', 'group', 'of', 'people', 'posing', 'for', 'a', 'picture', '.', '<END>']
```

5. Evaluation using **BLEU** scores

BLEU, or the **Bilingual Evaluation Understudy**, is a score for comparing a candidate translation of text to one or more reference translations.

The primary programming task for a BLEU implementer is to compare **n-grams of the candidate with the n-grams of the reference translation** and count the number of matches. These matches are position-independent. **The more the matches, the better the candidate translation is.**

Specifically,



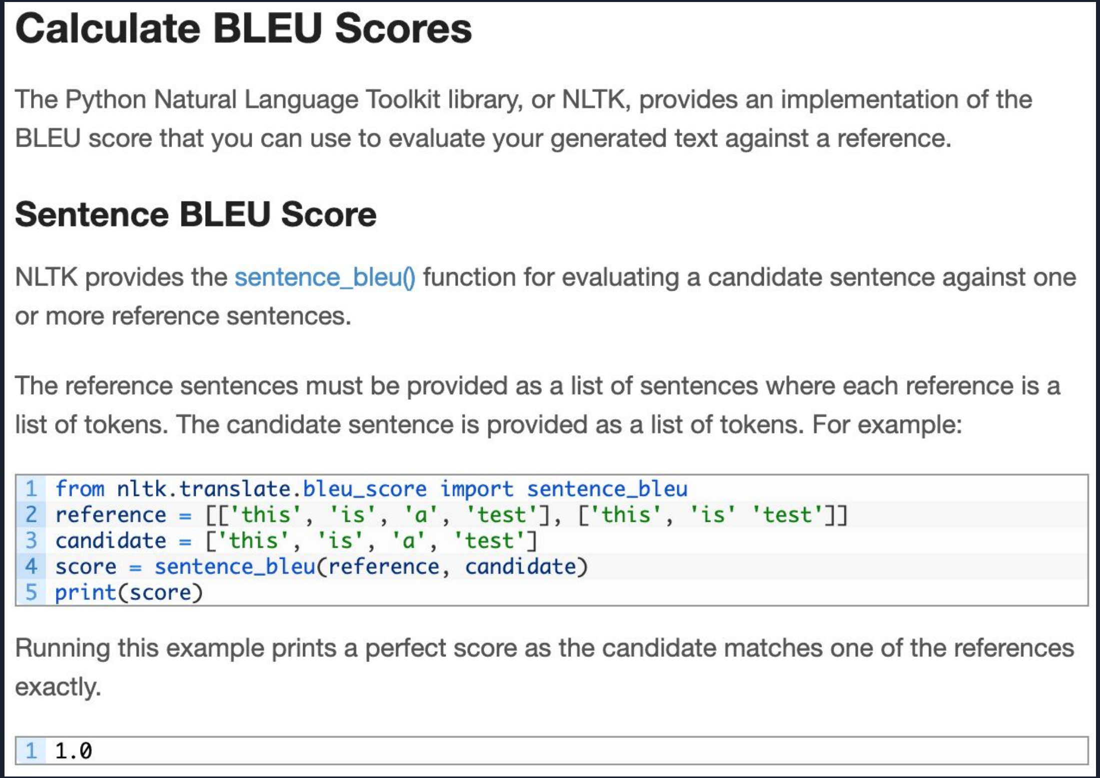
$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} .$$

Then,

$$\mathrm{B{\small LEU}} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n \right) .$$

where c is the number of unigrams (length) in all the candidate sentences, and r is the best match lengths for each candidate sentence in the corpus.

5. Evaluation using **BLEU** scores cont.



Calculate BLEU Scores

The Python Natural Language Toolkit library, or NLTK, provides an implementation of the BLEU score that you can use to evaluate your generated text against a reference.

Sentence BLEU Score

NLTK provides the sentence_bleu() function for evaluating a candidate sentence against one or more reference sentences.

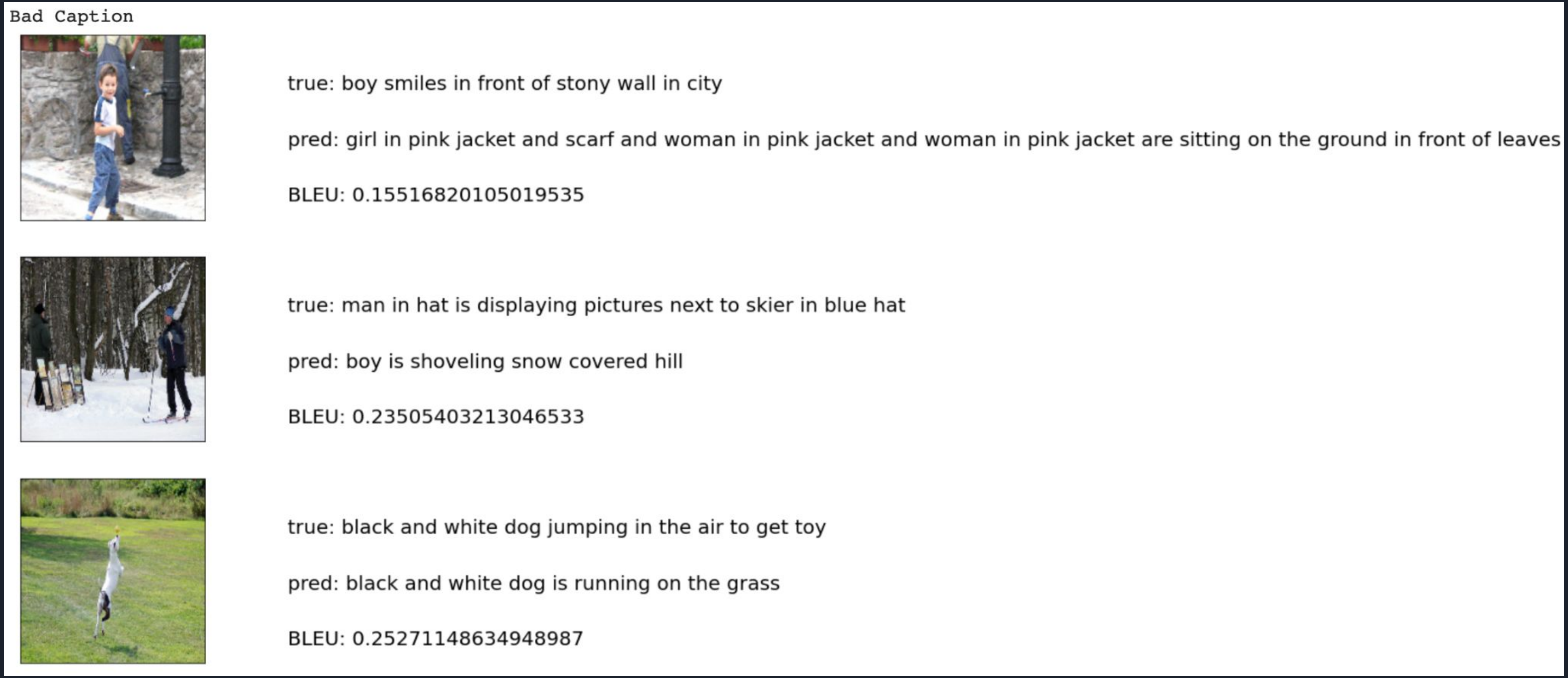
The reference sentences must be provided as a list of sentences where each reference is a list of tokens. The candidate sentence is provided as a list of tokens. For example:

```python
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'a', 'test'], ['this', 'is' 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate)
print(score)
```

Running this example prints a perfect score as the candidate matches one of the references exactly.

```
1.0
```

5. Evaluation using **BLEU** scores cont.



Bad Caption

true: boy smiles in front of stony wall in city

pred: girl in pink jacket and scarf and woman in pink jacket and woman in pink jacket are sitting on the ground in front of leaves

BLEU: 0.15516820105019535

true: man in hat is displaying pictures next to skier in blue hat

pred: boy is shoveling snow covered hill

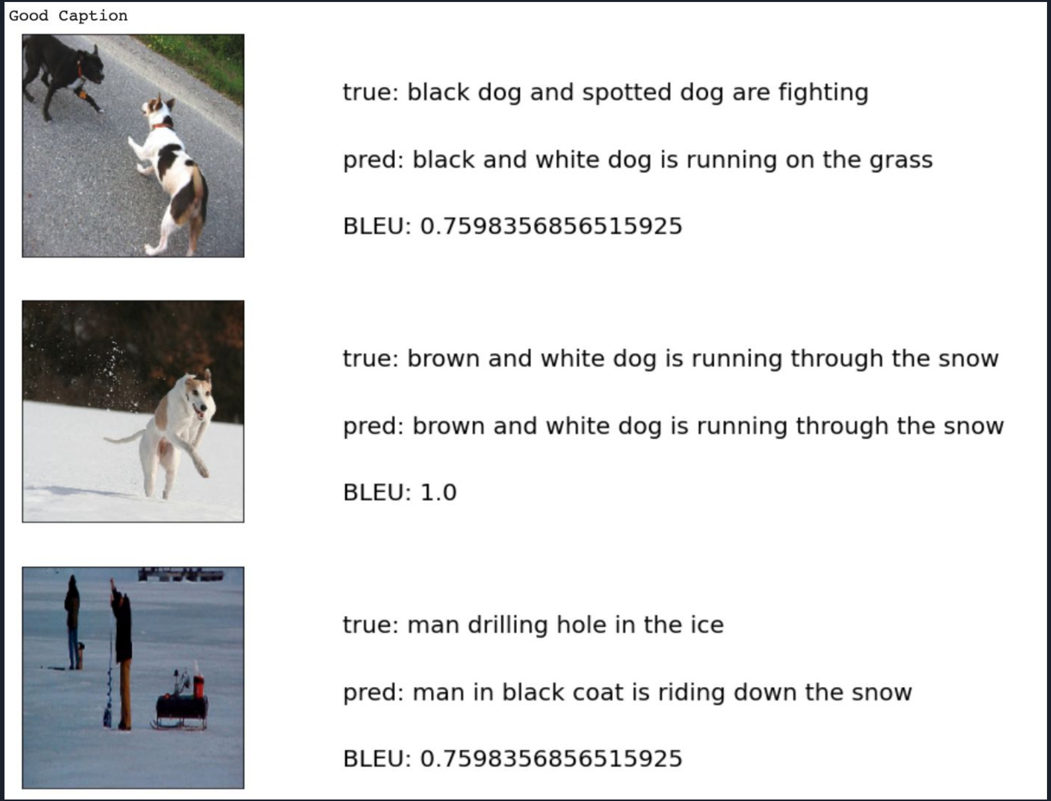
BLEU: 0.23505403213046533

true: black and white dog jumping in the air to get toy

pred: black and white dog is running on the grass

BLEU: 0.25271148634948987

5. Evaluation using **BLEU** scores cont.



Good Caption

true: black dog and spotted dog are fighting

pred: black and white dog is running on the grass

BLEU: 0.7598356856515925

true: brown and white dog is running through the snow

pred: brown and white dog is running through the snow

BLEU: 1.0

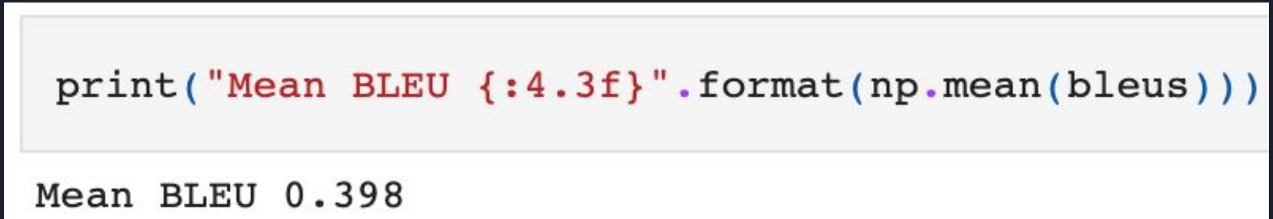
true: man drilling hole in the ice

pred: man in black coat is riding down the snow

BLEU: 0.7598356856515925

5. Evaluation
using **BLEU** scores cont.

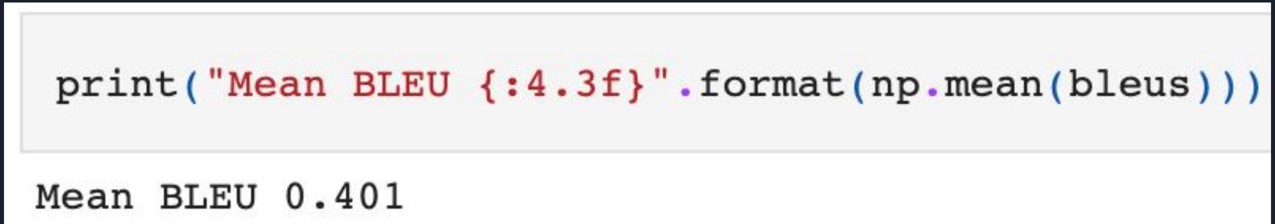
VGG-16:



```
print("Mean BLEU {:4.3f}".format(np.mean(bleus)))
```

```
Mean BLEU 0.398
```

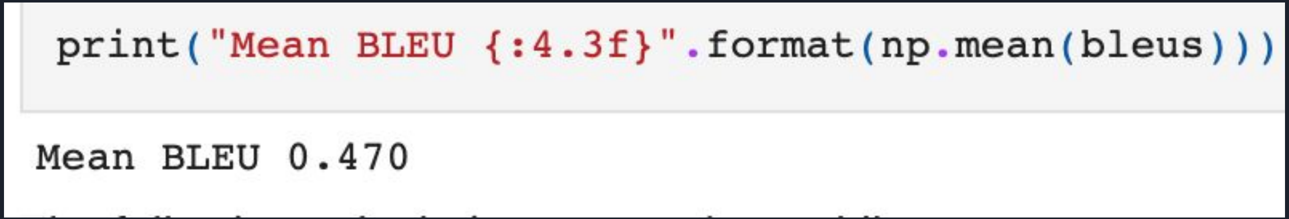
Inception V2:



```
print("Mean BLEU {:4.3f}".format(np.mean(bleus)))
```

```
Mean BLEU 0.401
```

Inception V3:



```
print("Mean BLEU {:4.3f}".format(np.mean(bleus)))
```

```
Mean BLEU 0.470
```

Future Work

- Other evaluation metrics? E.g. ROUGE [Lin, 2004]
- Testing with different hyperparameters
- Flickr30k dataset
- Other models? E.g. Inception V4

Reference List

- M. 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.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
- Raj, B., 2018. A Simple Guide to the Versions of the Inception Network. [online] Medium. Available at: <https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202> [Accessed 27 April 2022].
- Brownlee, J., 2022. A Gentle Introduction to Calculating the BLEU Score for Text in Python. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/calculate-bleu-score-for-text-python/> [Accessed 27 April 2022].
- Dobilas, S., 2022. LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past. [online] Medium. Available at: <https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e> [Accessed 27 April 2022].