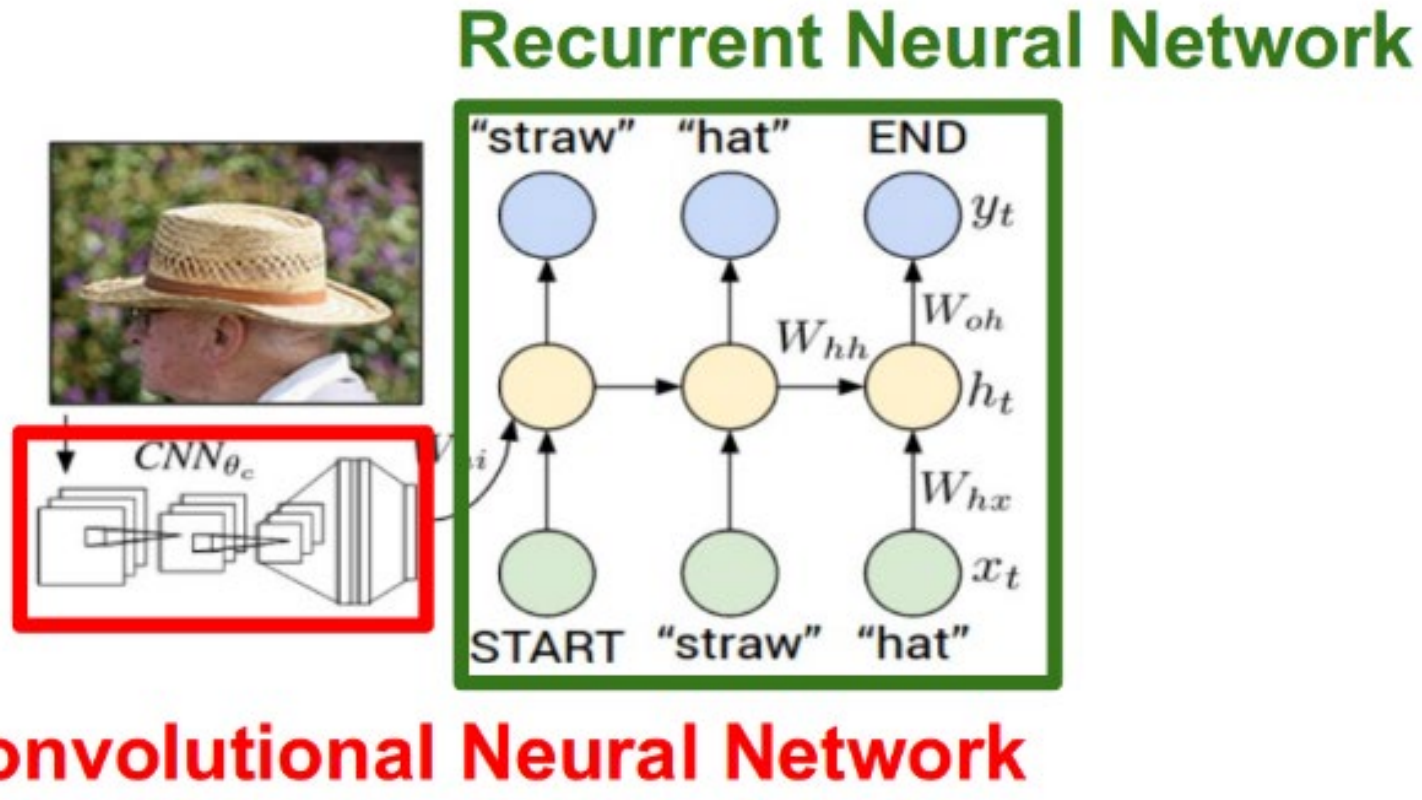
IMAGE CAPTIONING WITH BIDIRECTIONAL LSTM AND BEAM SEARCH METHOD: INVESTIGATING ATTENTION

Arnab Karmakar (SC15B079) and Samvram Sahu (SC15B132)

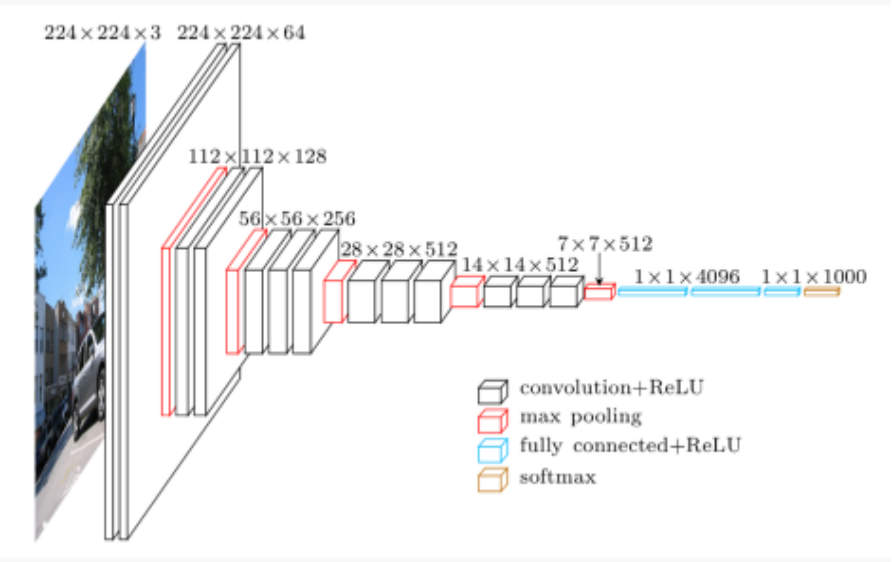
INTRODUCTION

BASIC IMAGE CAPTIONING

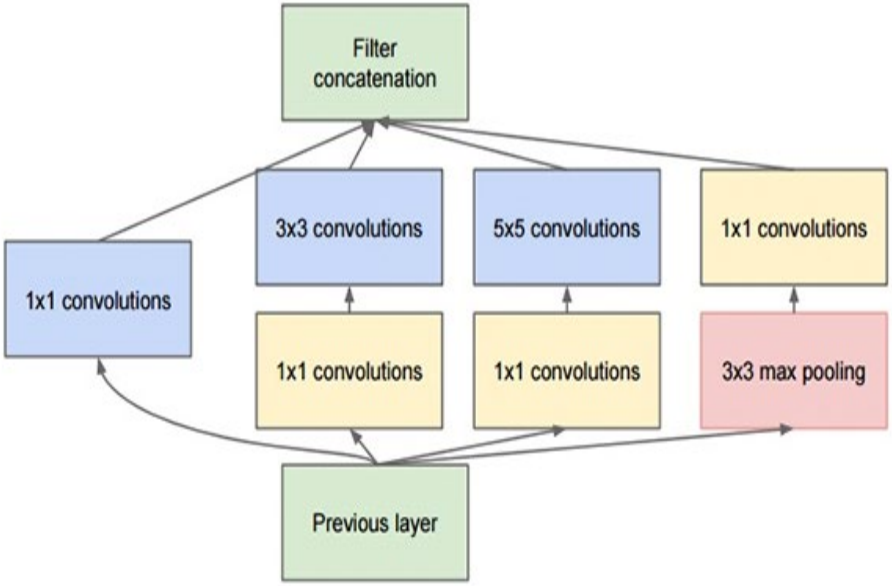


CNN's output of **FC** layer is fed to a **RNN**, which converts it into meaningful descriptions

VGG-16 vs. INCEPTION-V3

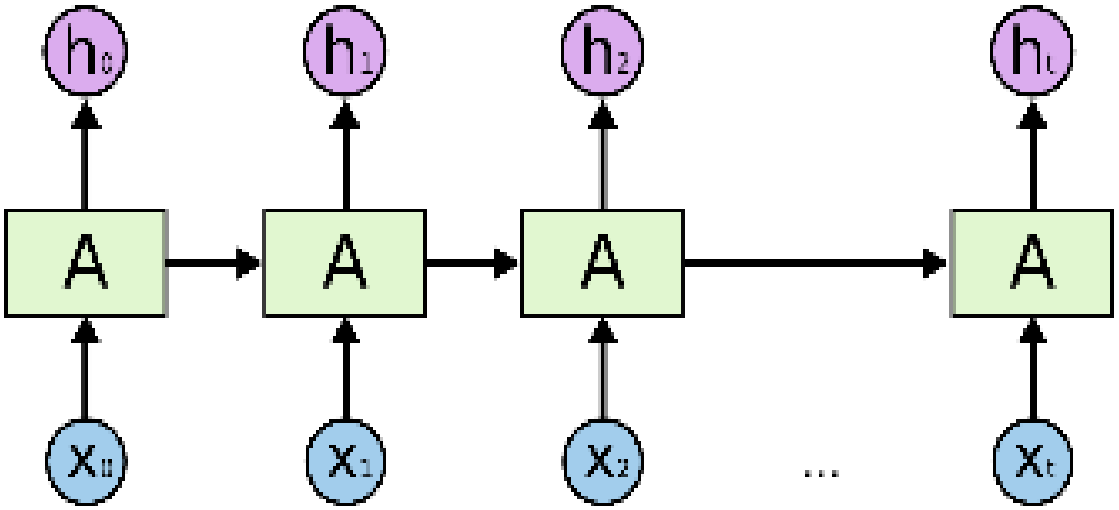


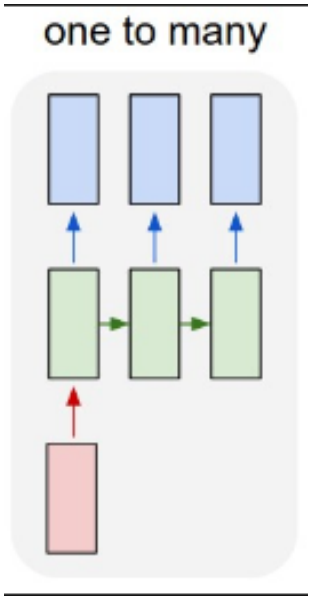
- Unfortunately, there are two major drawbacks with VGGNet:
  - It is *painfully slow* to train.
  - The network architecture weights themselves are quite large (in terms of disk/bandwidth).



- Few advantages of Inception V_3:
  - Act as a "multi-level feature extractor"
  - The weights for Inception V3 are smaller than both VGG and ResNet

RNN : RECURRENT NEURAL NETWORK

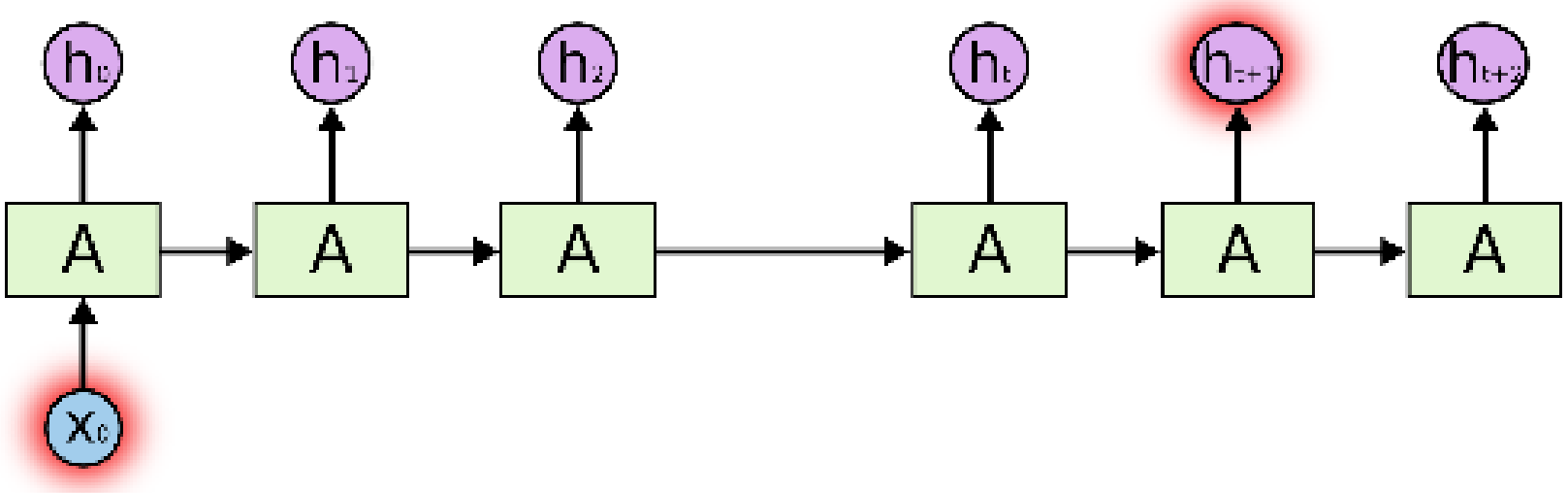




- By default, a RNN is used for covering time series dependencies on the extracted features, and hence is used in this application. However we will first see the problems with using RNN as the tokenizer.

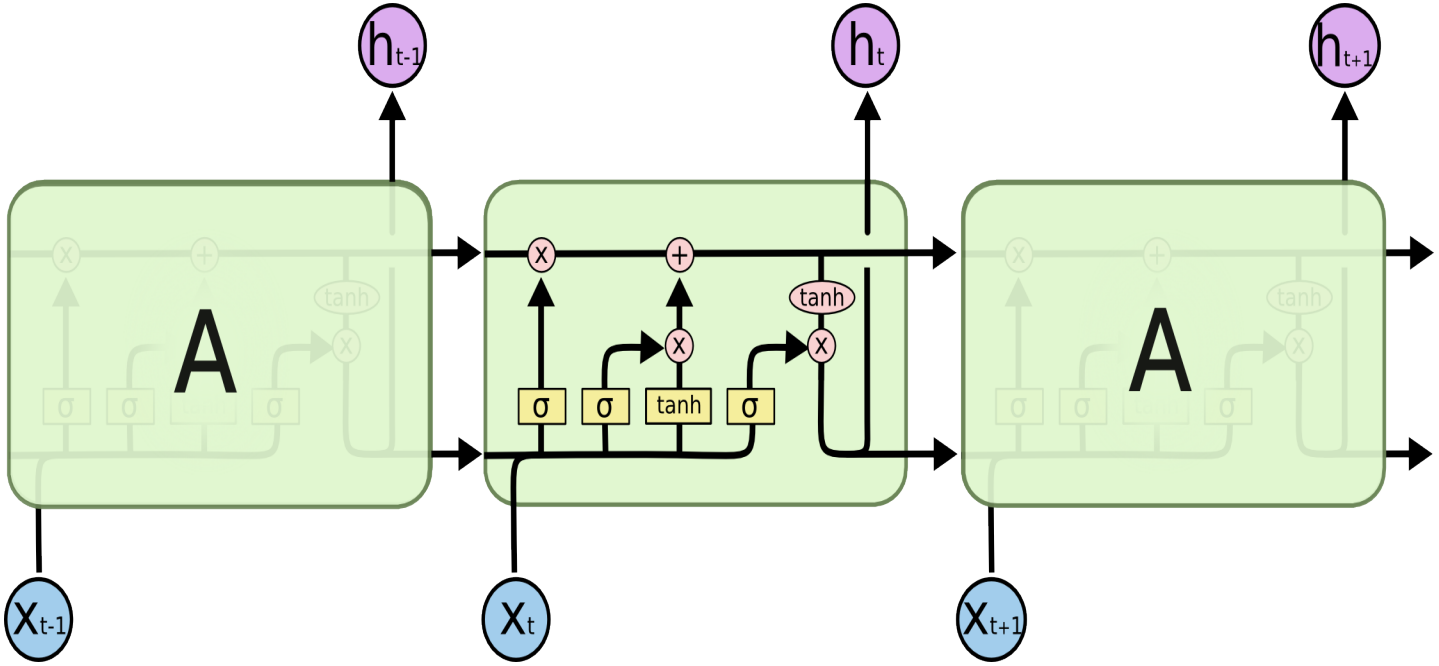
- We use one to many RNN as it gives an output consisting of many words, from a single feature vector of an image.

WHY RNNs ARE INEFFICIENT



- RNNs are popular for their ability to use previous stage data to make correct predictions in future stages, however at each stage the output of the previous layer is completely modified. Hence, if a dependency is very long, it might not even capture that effect!

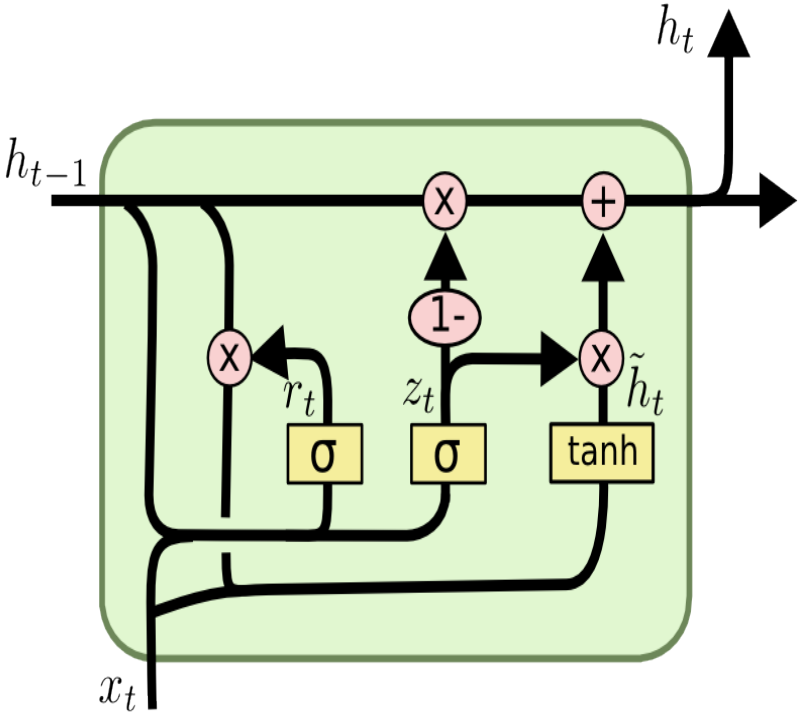
LSTM



- Two features of LSTM which make it advantageous over RNN:

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

LSTM



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

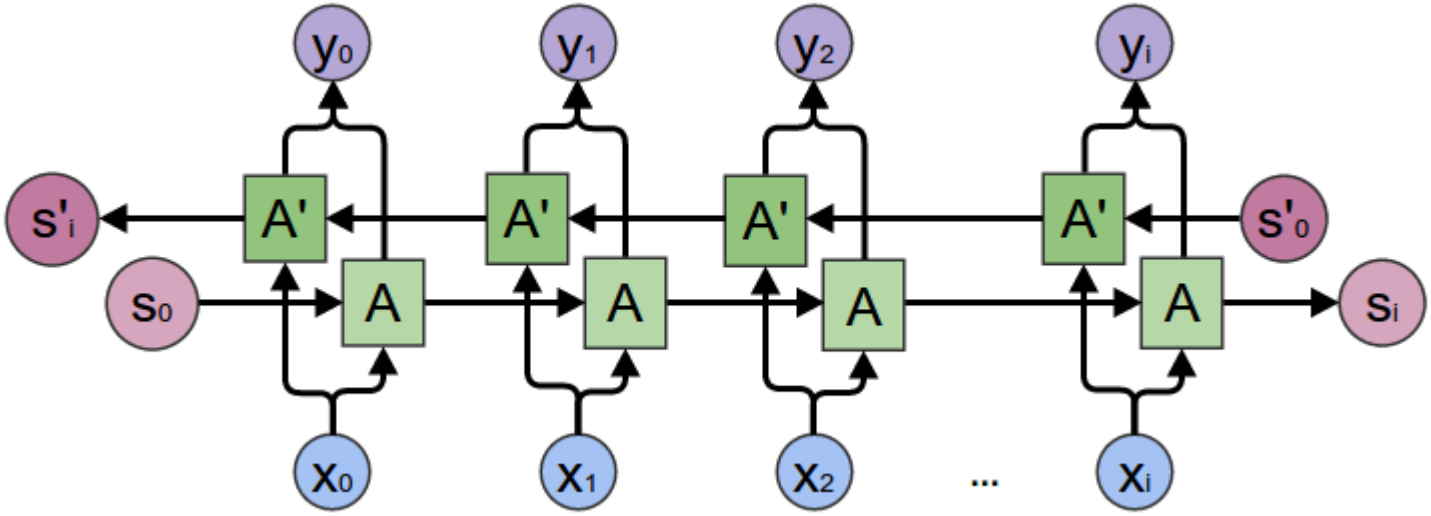
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

BIDIRECTIONAL LSTM

- A major issue with all of the above networks is that they learn representations from previous time steps. Sometimes, you might have to learn representations from future time steps to better understand the context and eliminate ambiguity.



BIDIRECTIONAL LSTM

- Take the following examples,
  - "He said, Teddy bears are on sale"
  - "He said, Teddy Roosevelt was a great President"
- In the above two sentences, when we are looking at the word "Teddy" and the previous two words "He said", we might not be able to understand if the sentence refers to the President or Teddy bears. Therefore, to resolve this ambiguity, we need to look ahead. This is what Bidirectional LSTMs accomplish.
- The repeating module in a Bidirectional LSTM could be a conventional RNN, LSTM or GRU. The structure and the connections of a bidirectional LSTM are represented in figure (previous slide). There are two type of connections, one going forward in time, which helps us learn from previous representations and another going backwards in time, which helps us learn from future representations.

BIDIRECTIONAL LSTM

- Forward propagation is done in two steps:
- We move from left to right, starting with the initial time step we compute the values until we reach the final time step
- We move from right to left, starting with the final time step we compute the values until we reach the initial time step

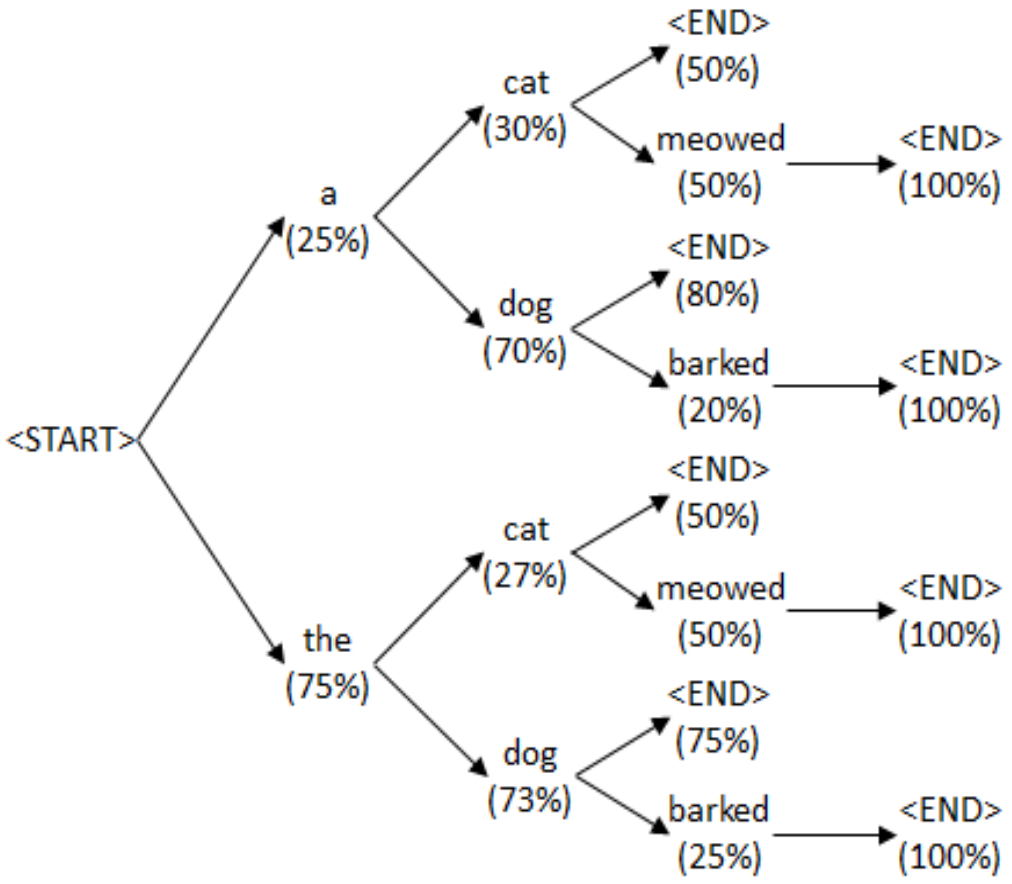
PREDICTING THE CAPTIONS

- **Argmax Search** is where the maximum value index(argmax) in the predicted vector is extracted and appended to the result. This is done until we hit <end> or the maximum length of the caption.

- **Beam Search** is where we take top **k** predictions, run again through the model prediction and then sort them using the probabilities returned by the model. So, the list will always contain the top **k** predictions. In the end, we take the one with the highest probability and go through it till we encounter <end> or reach the maximum caption length
  - "The local beam search algorithm keeps track of k states rather than just one. At each step, all the successors of all k states are generated. If any one is a goal, the algorithm halts. Otherwise, it selects the k best successors from the complete list and repeats."

BEAM SEARCH

- By searching through specific combinations of words, and creating different possible outputs, beam search constructs a whole sentence without relying too heavily on any individual word from the ones which the LSTM may generate at any specific time step. **Beam search, therefore, can rank a lot of different sentences according to their collective, or holistic, probability.**



DATASET

- We have used Flickr8k dataset (size: 1GB). Flickr8K has
- The standard **dataset** division that are used for this set is 6000 for training, 1000 for validation, 1000 for testing.
- Each image has 5 full sentence level caption for each image.

BLEU (BILINGUAL EVALUATION UNDERSTUDY) SCORE

- BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations, i.e. it is a measure of evaluating a generated sentence to a reference sentence.
- Benefits of BLEU:
  - It is quick and inexpensive to calculate.
  - It is easy to understand.
  - It is language independent.
  - It correlates highly with human evaluation.
  - It has been widely adopted.

HOW IS BLEU EVALUATED?

- The approach works by counting matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. The comparison is made regardless of word order.
- BLEU 1 - Comparing each word without respect to order of words.
- BLEU 2 - Bigram model where each pair of word is compared for equality.
- BLEU 3 - Trigram model where each group of 3 words is compared.
- BLEU 4 - Group of 4 words are compared together for equality.
- Weighted BLEU and Cumulative BLEU scores also exist

RESULTS

RESULTS

RESULTS

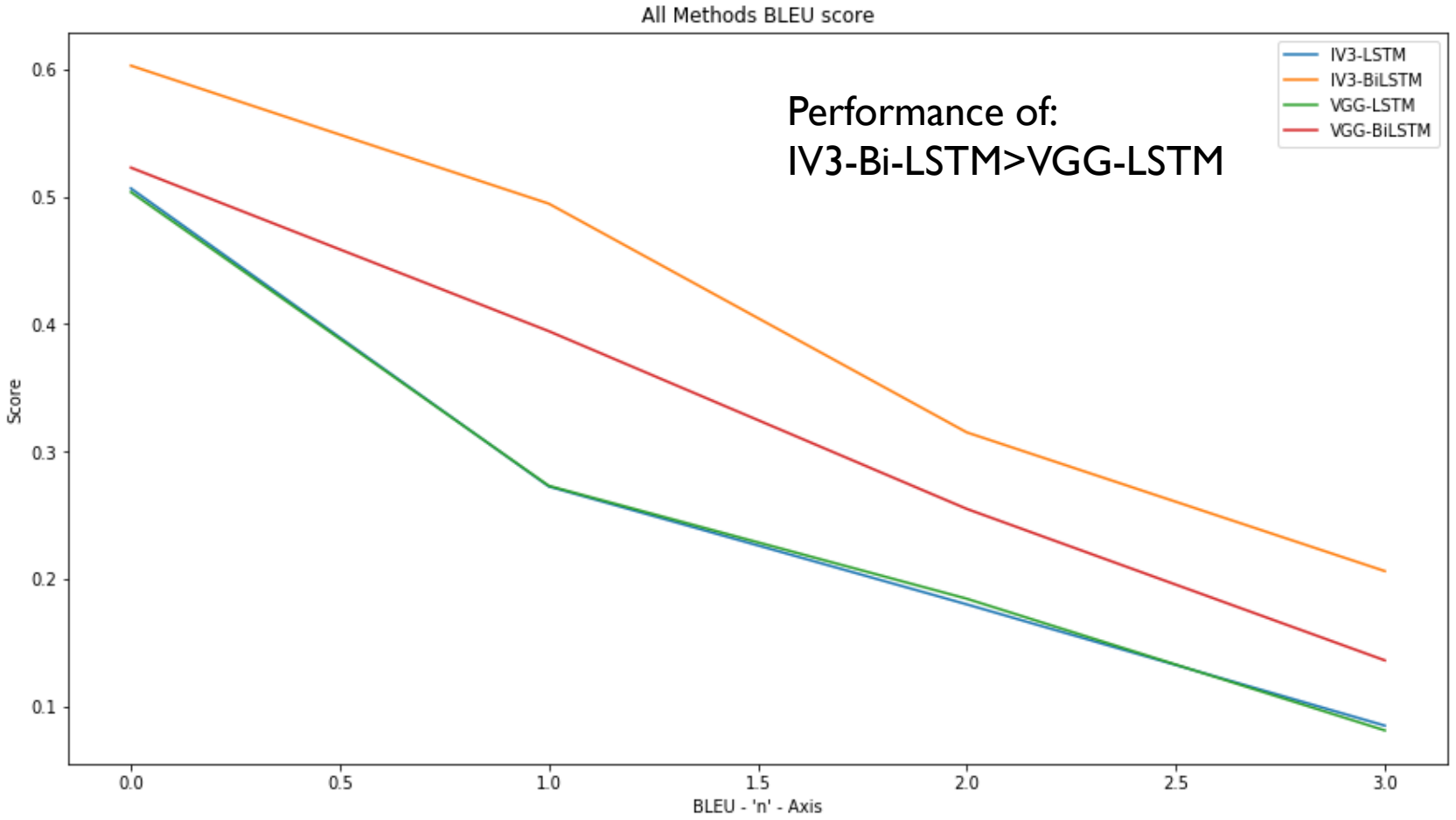
- iv3-lstm : man in black shirt and black hat is standing on bench with man in red shirt
- vgg-lstm: man in blue shirt and black hat is standing in front of the bench
- vgg-bilstm: man in red shirt and blue shirt is sitting on the camera over a bench

RESULTS

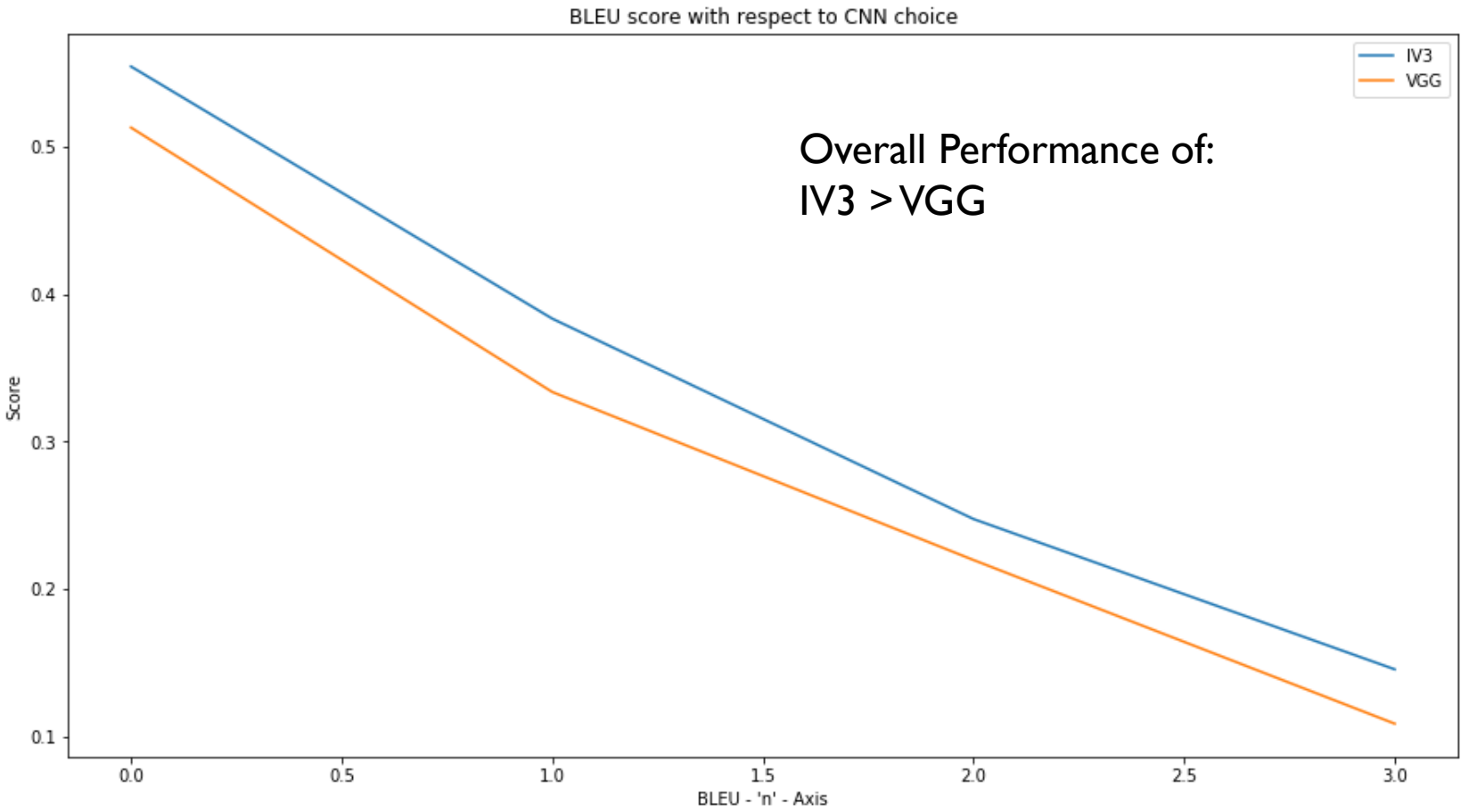
RESULTS

- IV3-lstm
  - man in red shirt is riding bike down the street
  - man in red shirt is standing on rock overlooking the ocean
- VGG-lstm
  - man in red shirt is standing in front of the street
  - children are playing with the camera in the grass
- VGG-bilstm
  - red car sitting on the street
  - man in red shirt is standing on rock down the street

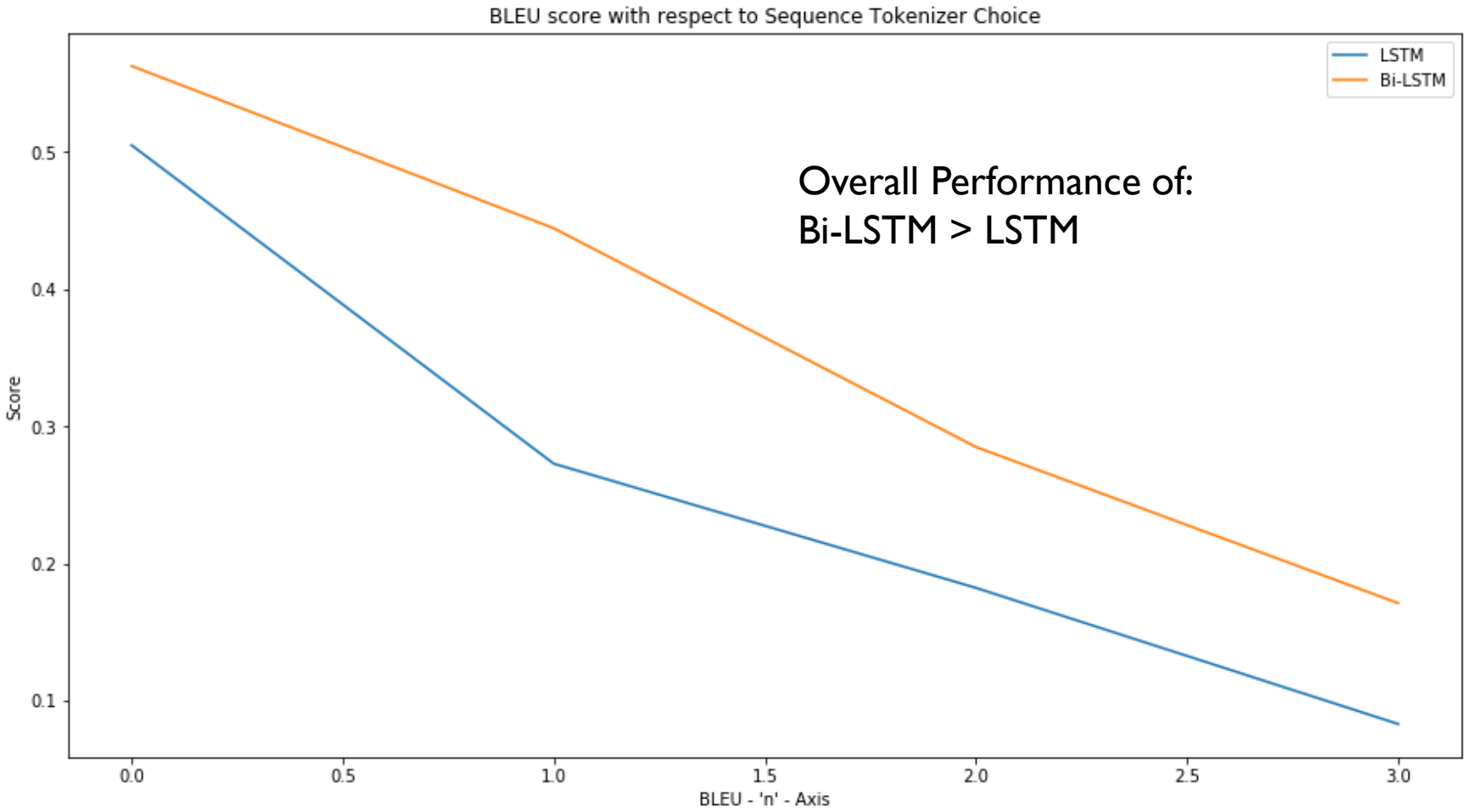
INFERENCES


All Methods BLEU score
IV3-LSTM
IV3-BiLSTM
VGG-LSTM
VGG-BiLSTM
Performance of:
IV3-Bi-LSTM>VGG-LSTM
Score
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
BLEU - 'n' - Axis

INFERENCES


BLEU score with respect to CNN choice
IV3
VGG
Overall Performance of:
IV3 > VGG
Score
0.1
0.2
0.3
0.4
0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
BLEU - 'n' - Axis

INFERENCES


BLEU score with respect to Sequence Tokenizer Choice
LSTM
Bi-LSTM
Overall Performance of:
Bi-LSTM > LSTM
Score
0.5
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
BLEU - 'n' - Axis

ATTENTION MODEL

- Attention as a technique, in this context, refers to the ability of the CNN **to weight regions of the image differently**. Broadly, it can be understood as a tool to direct the allocation of available processing resources towards the most informative parts of the input signal. Rather than summing up the image as a whole, we would like to know how CNN assigns weights to the '*salient*' parts of the image.
- In this work, we would like to focus for each caption, how CNN is perceiving the different parts of the image.
- We introduce cropped up images to our network, as well as occluded subjects in different forms, and we investigate how our model predicts the output.

ATTENTION MODEL

Iv3-lstm: two dogs are playing in the grass
vgg-lstm: dog is running through the grass
vgg-bilstm: man with dog play in the grass

ATTENTION MODEL

- Iv3-lstm: young boy in red shirt is running through the grass
- Vgg-lstm: two children are playing with the camera in the grass
- Vgg-bilstm: girl in pink shirt is sitting on the grass

ATTENTION MODEL

- Iv3-lstm: dog is running through the grass
- Vgg-lstm: dog is running through the grass
- Vgg-bilstm: dog is running on the grass

ATTENTION MODEL

- Iv3-lstm: two children are playing in the water
- Vgg-lstm: dog is running through the grass
- Vgg-bilstm: the brown dog is running on the grass

CONCLUSION

- Bi-Directional LSTM works far better than LSTM, due to its capability of mapping both forward and backward sequences.
- Dataset is small, hence the generalization capacity can be improved.
- With limited resources, it was our best attempt
- Training time (used Early Stopping)
  - Bi-Directional LSTM – 105 minutes per epoch
  - LSTM – 45 minutes per epoch
- CNNs have not been trained for this purpose.

FUTURE WORK

- Using Region-Proposal CNNs (R-CNN) or Faster R-CNN
- Using Object Detection and Localization Networks might improve feature extraction capability
  - YOLO or SSD

THANK YOU