

---

---

# Visual Question Answering

Final project for the Deep Learning course

---

---

# Task

## Input:

image + free-form, open-ended, natural  
language question



## Output:

natural language answer

AI capability required	Questions examples
fine-grained recognition	"What kind of cheese is on the pizza?"
object detection	"How many bikes are there?"
activity recognition	"Is this man crying?"
knowledge-base reasoning	"Is this a vegetarian pizza?"
commonsense reasoning	"Is this person expecting company?"
other	"...."



# Proposed approaches

Visual Question Answering



**Random baseline**

**Prior yes baseline**

**Approach 1: CNN + LSTM**

**Approach 2: Generative LXMERT**

# Dataset

Visual Q&A v2.0 <https://visualqa.org/download.html>

- Images from MS COCO and Abstract scene datasets

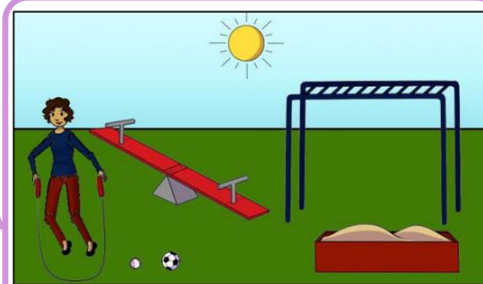
123.287 images  
443.753 questions

29.998 images  
60.000 questions

- At least 3 questions per image (5.4 on average)
- 10 ground truth answers per question
- Confidence (*yes, maybe, no*) associated to each answer
- 3 plausible (but likely incorrect) answers per question



How many bikes are there?	2	3
	2	4
	2	12
What number is the bus?	48	4
	48	46
	48	number 6



How many balls are there?	2	1
	2	2
	2	3
What side of the teeter totter is on the ground?	right	left
	right	left
	right side	right side

# Preprocessing

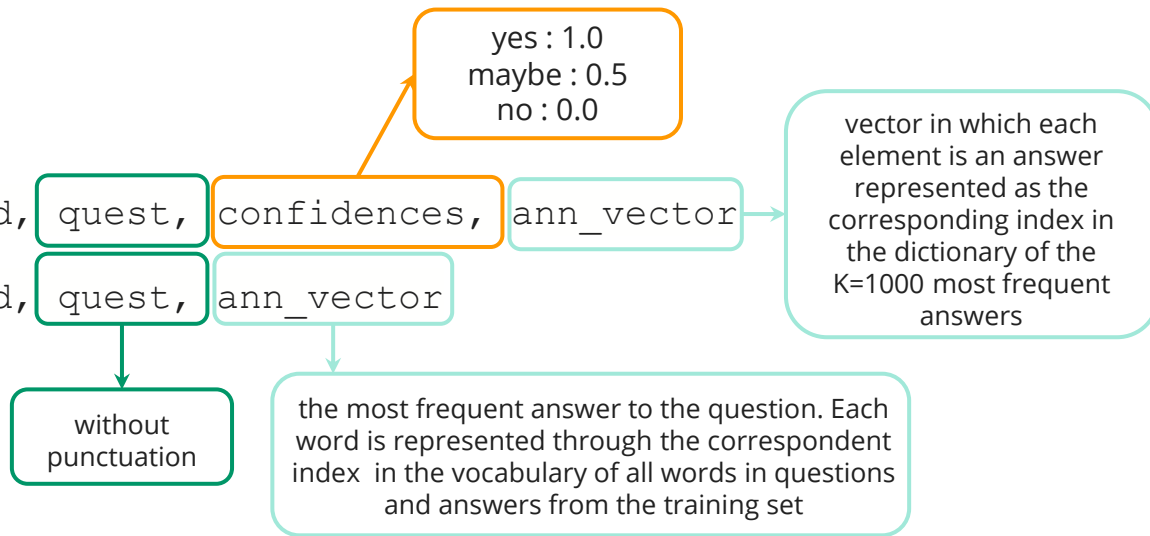
In order to lighten the computational burden:

- Build a dictionary with image\_id-image\_filename pairs
- Randomly consider just a portion of the total number of samples (10%)
- Perform some data preprocessing and save it in a file before the actual creation of the dataset

Format of each sample in the file:

approach 1: `img_filename, qst_id, quest, confidences, ann_vector`

approach 2: `img_filename, qst_id, quest, ann_vector`



# Dataset creation

After loading back the preprocessed samples, we further process them as follows:

## Approach 1

- Encode questions using GloVe embeddings.



`image_filename, question_id,  
question, preferences, labels`

## Approach 2

- Encode questions using LXMERT tokenizer, which outputs `input_ids`, `token_type_ids`, `attention_mask`.
- Pad all answers to the same length.



`image_filename, question_id,  
question, labels, input_ids,  
attention_mask, token_type_ids`

# Evaluation



Before performing the evaluation, both answers and predictions are preprocessed as follows:

- Make them lowercase
- Substitute numbers with digits
- Remove punctuation and articles

Metric robust to inter-human variability in phrasing the answers:

$$Acc(ans) = \min\left(\frac{\text{number of humans that said } ans}{3}, 1\right)$$

# Trivial baselines: Random and Prior yes

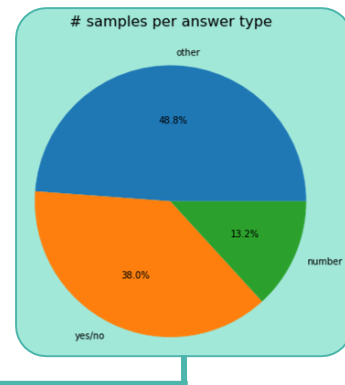
As trivial baselines, we have chosen to implement:



1) **Random**: the output answer is taken randomly from the K (1000) most frequent answers



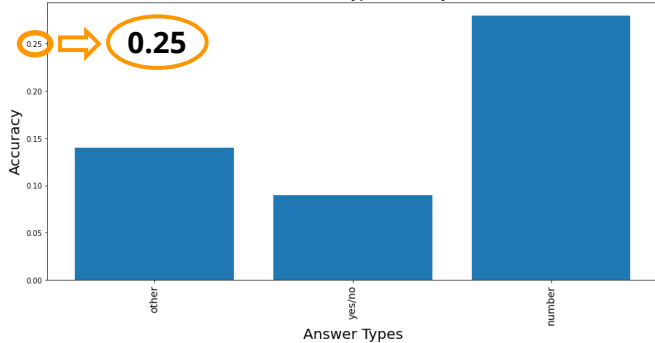
2) **Prior yes**: for every sample, the prediction is yes





# Trivial baselines: evaluation

Per Answer Type Accuracy

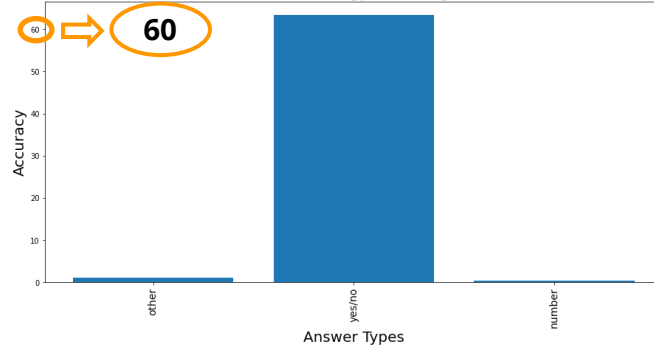


Random

Overall Accuracy	0.14
Per Answer Type Accuracy is the following:	
other	0.14
yes/no	0.09
number	0.28

Prior yes

Per Answer Type Accuracy



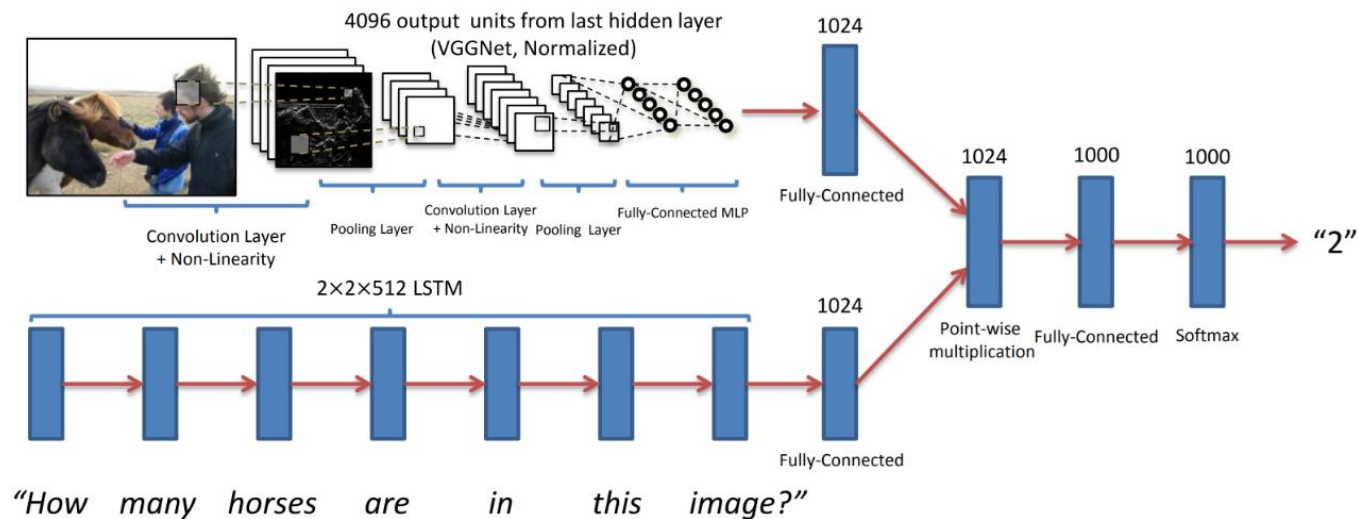
Overall Accuracy	24.99
Per Answer Type Accuracy is the following:	
other	0.99
yes/no	63.43
number	0.32

# Approach 1

## CNN + LSTM

Vision  
channel

Language  
channel

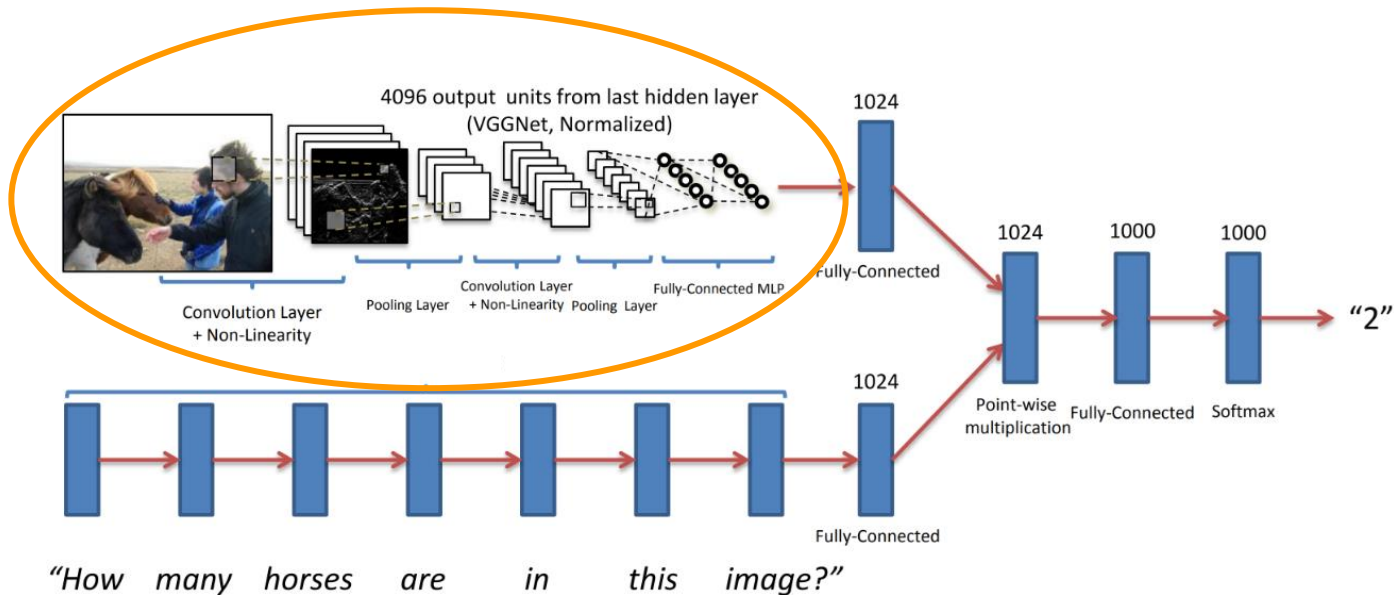


# Approach 1

## CNN + LSTM

4096-dim activation values from the last hidden layer of pre-trained VGG are l2 normalized.

Vision  
channel



Language  
channel

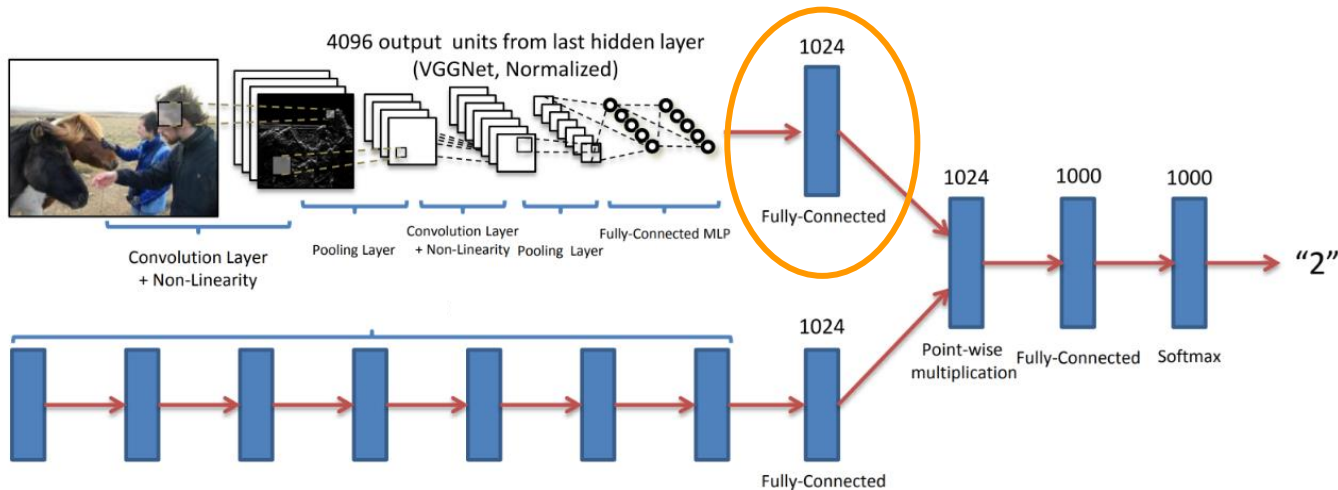
"How many horses are in this image?"

# Approach 1

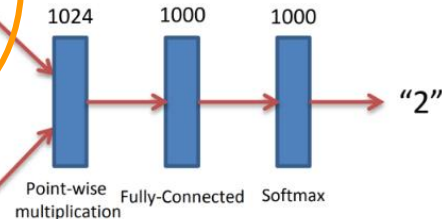
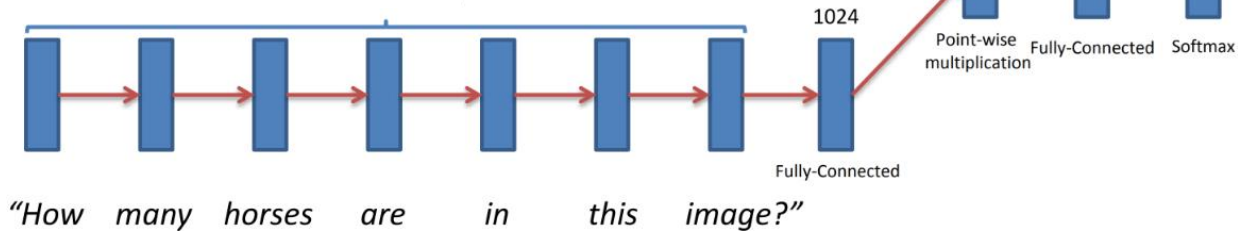
## CNN + LSTM

Fully-connected layer with tanh nonlinearity to make image embedding 1024-dimensional.

Vision  
channel



Language  
channel



# Approach 1

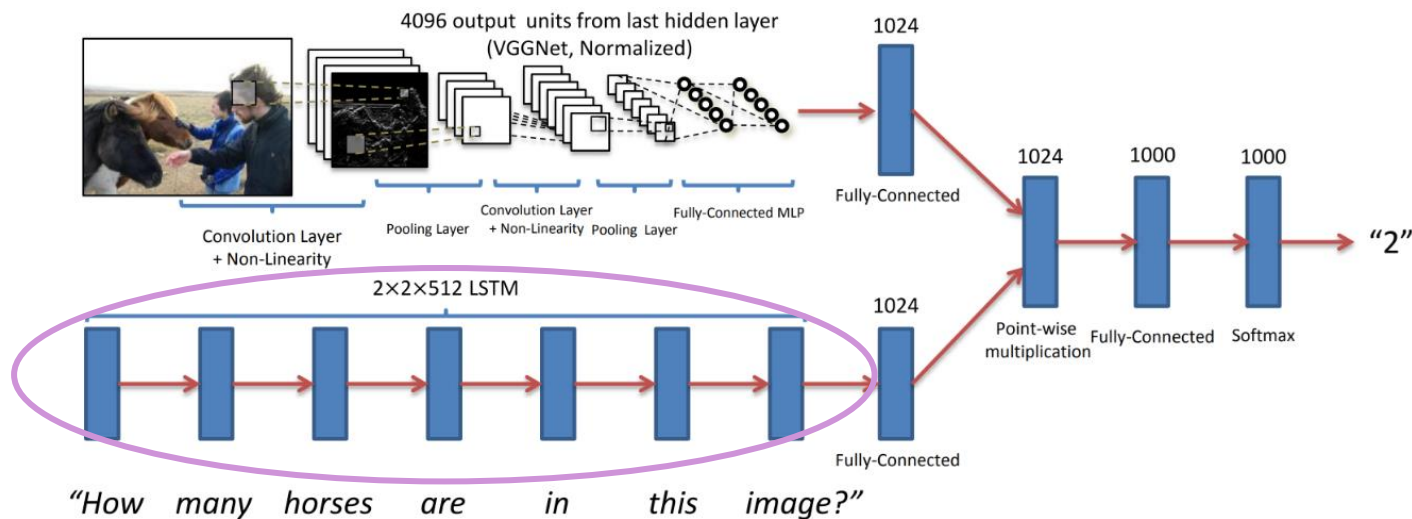
## CNN + LSTM

LSTM with 2 hidden layers. Last hidden state and cell state are concatenated to obtain a 2048-dimensional embedding.

GloVe pre-trained embeddings are used in the embedding layer.

Vision  
channel

Language  
channel

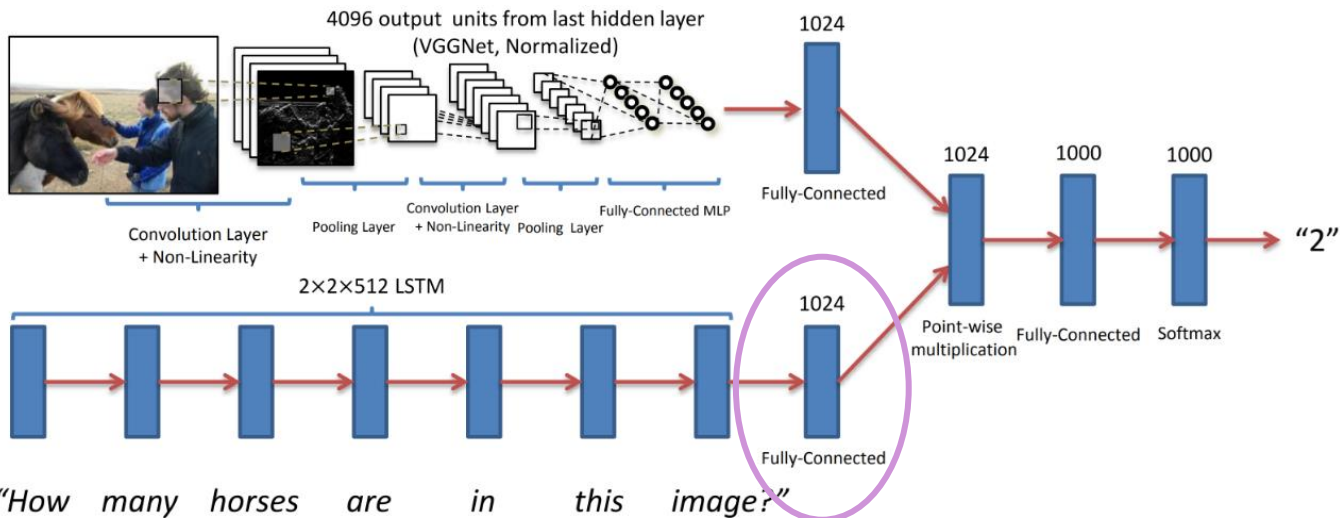


# Approach 1

## CNN + LSTM

Fully-connected layer with tanh nonlinearity to make language embedding 1024-dimensional.

Vision  
channel



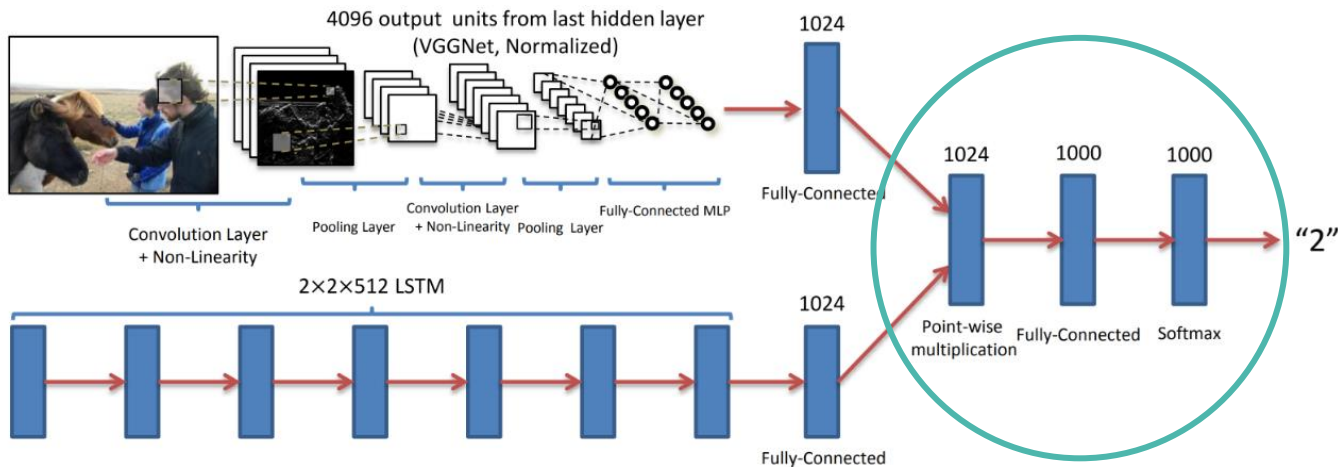
Language  
channel

# Approach 1

## CNN + LSTM

Image and question embeddings are combined via element-wise multiplication. The combined embedding is passed to an MLP with a linear layer of 1000 hidden units followed by tanh non-linearity and 0.5 dropout, and a final layer with K units followed by the softmax activation function, to obtain a probability distribution over the K most frequent answers.

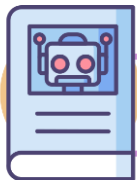
Vision  
channel



Language  
channel

# Approach 1: Losses

- ❖ Just answers among the K (1000) most frequent ones in the training set are considered.
- ❖ According to the reference paper, **cross-entropy loss** has been used.



## Standard loss

**Label:** index of the most frequent answer among the possible ones for the given question.



All other answers will be considered as wrong if predicted by our model.

## Novel loss

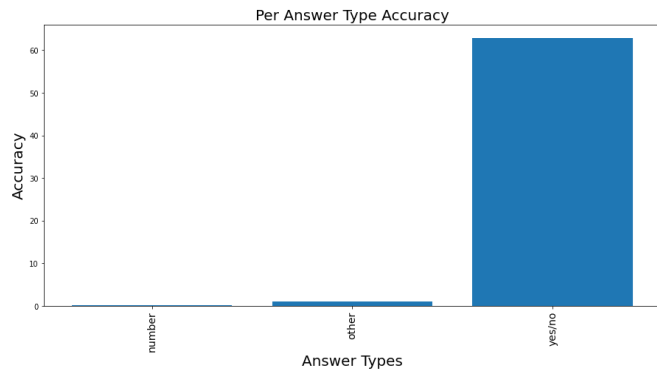


**Label:** K-dimensional vector values:

- 1 for answers with preference 'yes';
- 0.5 for answers with preference 'maybe';
- 0 for answers with preference 'no' or for not given answers.



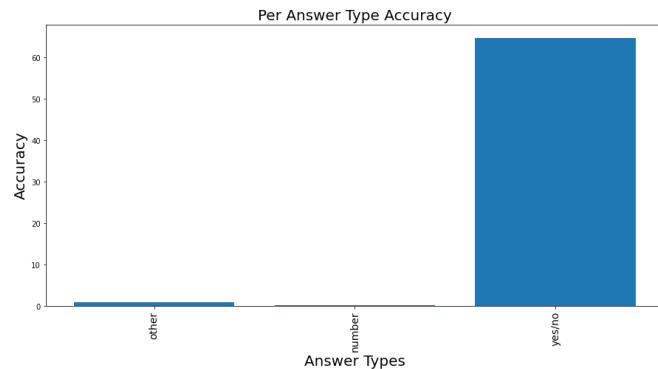
# Approach 1: Evaluation



**Standard  
loss**

Overall Accuracy	25.79
Per Answer Type Accuracy is the following:	
other	1.13
yes/no	62.90
number	0.17

**Novel  
loss**



Overall Accuracy	26.45
Per Answer Type Accuracy is the following:	
other	0.99
yes/no	64.71
number	0.20

# Approach 1: Results



Question: Is there a person in the picture?

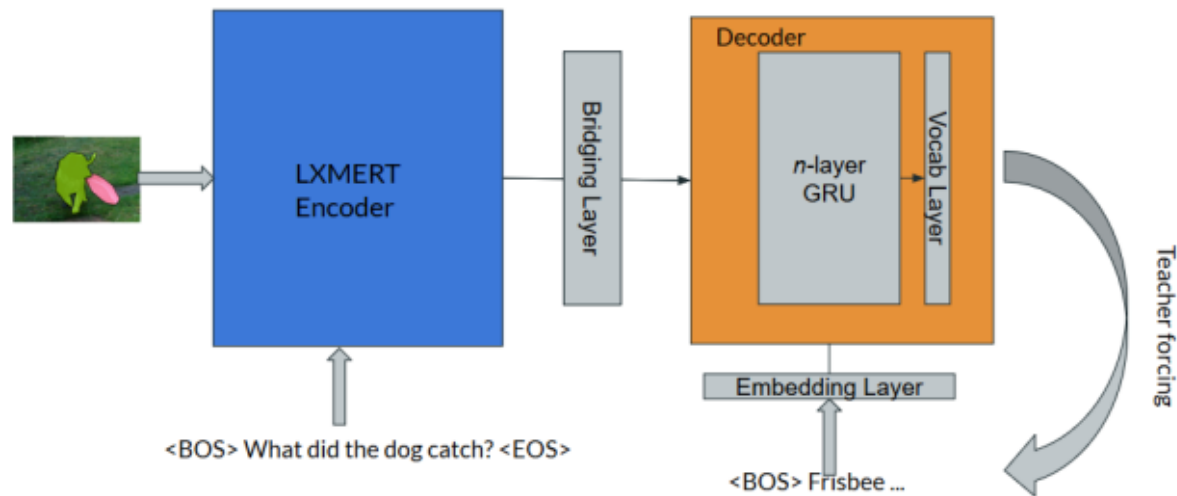
Answer 1: yes  
Answer 2: yes  
Answer 3: yes  
Answer 4: yes  
Answer 5: yes  
Answer 6: yes  
Answer 7: yes  
Answer 8: yes  
Answer 9: yes  
Answer 10: yes

Generated answer (accuracy 100.0)

Answer: yes

# Approach 2: Generative LXMERT

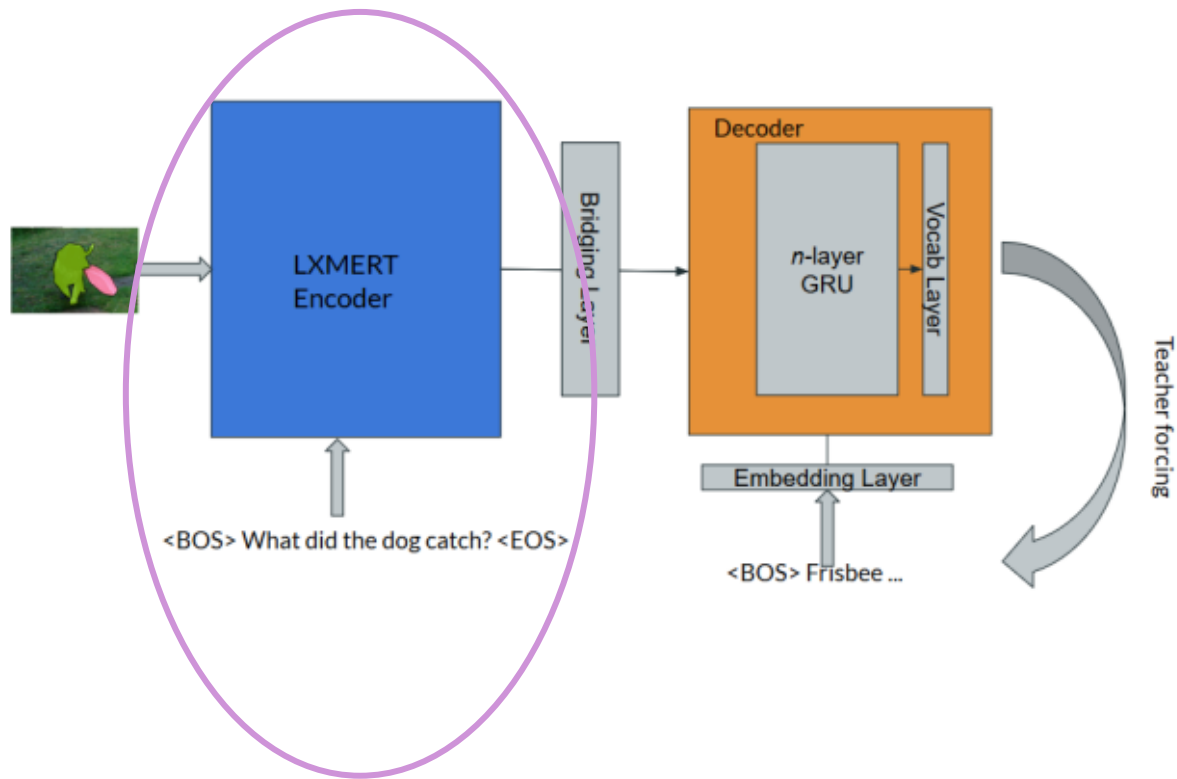
Encoder-decoder architecture which generates one answer's word at a time.



Reference paper "Generative Visual Question Answering using Cross-Modal Visual-Linguistic Embeddings", Guan, Chen:  
[Generative\\_Visual\\_Question\\_Answering\\_using\\_Cross-Modal\\_Visual-Linguistic\\_Embeddings.pdf](#)

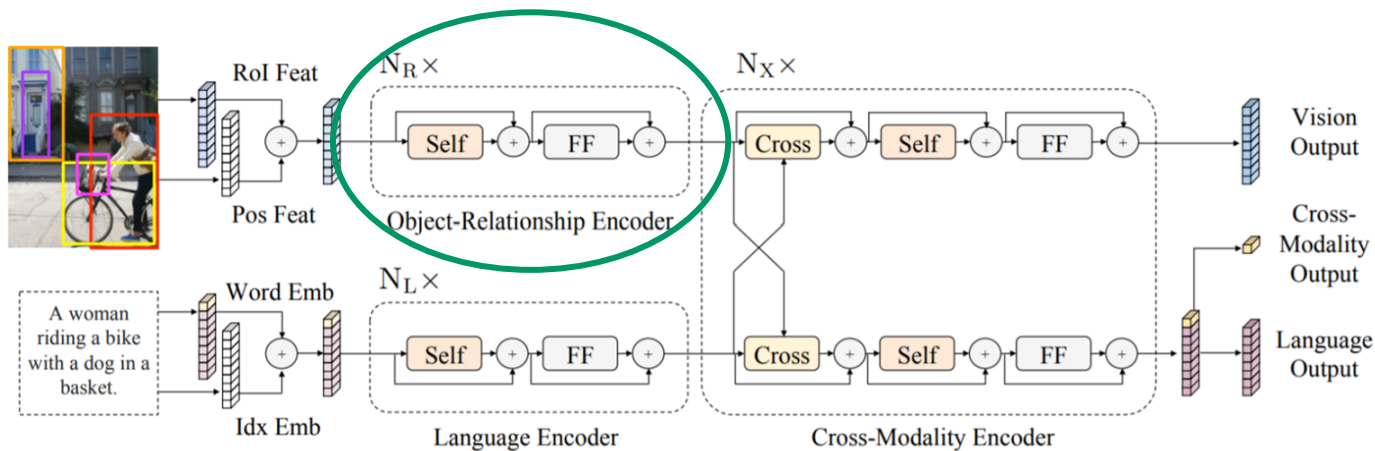
# Approach 2: Generative LXMERT

**Encoder:** pre-trained LXMERT model



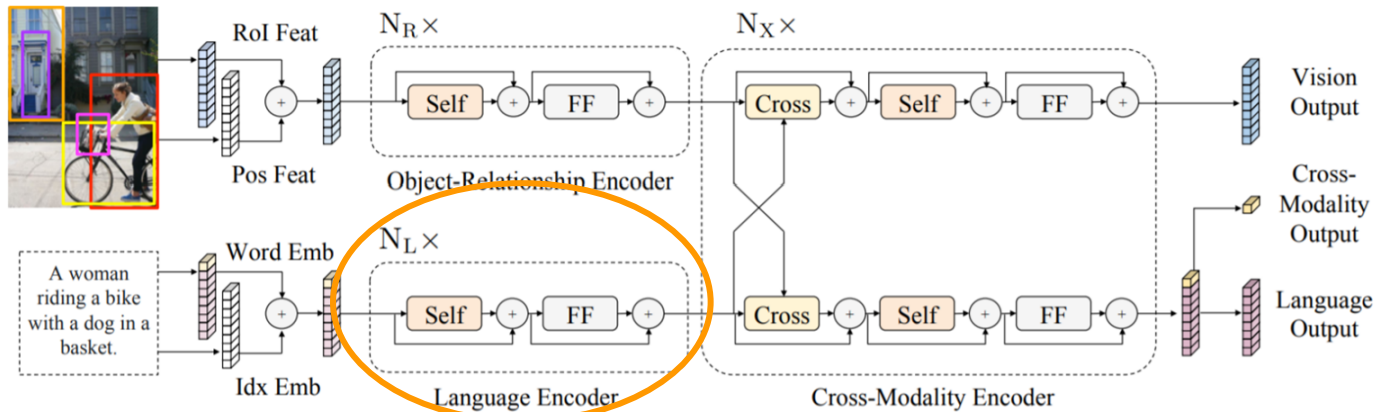
# Encoder: LXMERT model

- The **object-relationship encoder** uses the Region of Interest (RoI) features extracted by faster RCNN backbones and processes them using a BERT-style encoder.



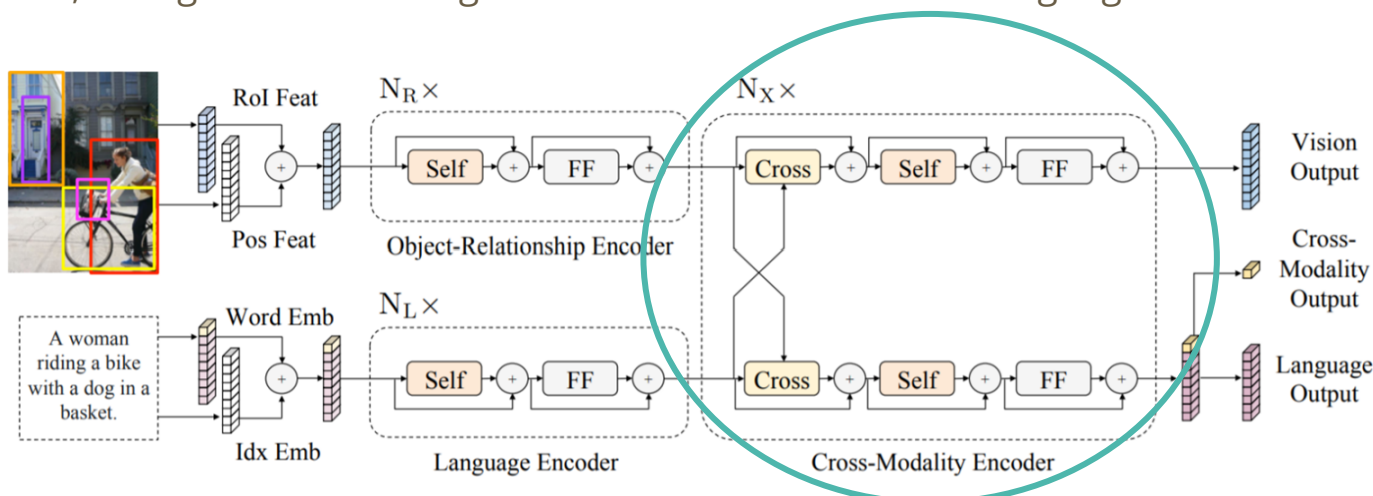
# Encoder: LXMERT model

- The **object-relationship encoder** uses the Region of Interest (RoI) features extracted by faster RCNN backbones and processes them using a BERT-style encoder.
- The **language encoder** is essentially the same as a BERT style self-attention based language encoder.



# Encoder: LXMERT model

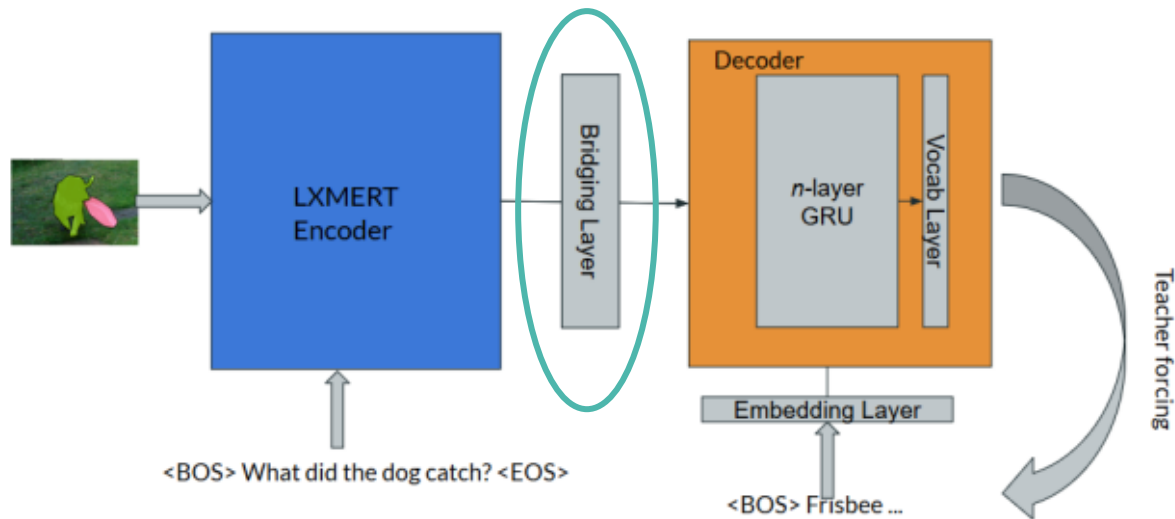
- The **object-relationship encoder** uses the Region of Interest (RoI) features extracted by faster RCNN backbones and processes them using a BERT-style encoder.
- The **language encoder** is essentially the same as a BERT style self-attention based language encoder.
- A **cross-modality encoder** allows language and visual embeddings to attend to each other, fusing information together for downstream visual-language related tasks.



# Approach 2: Generative LXMERT

## Bridging layer:

MLP with a single linear layer and ReLU activation function. Another linear layer is used to reduce the sequence length to 3, so that the output dimension matches the hidden dimension of the GRU in the decoder.

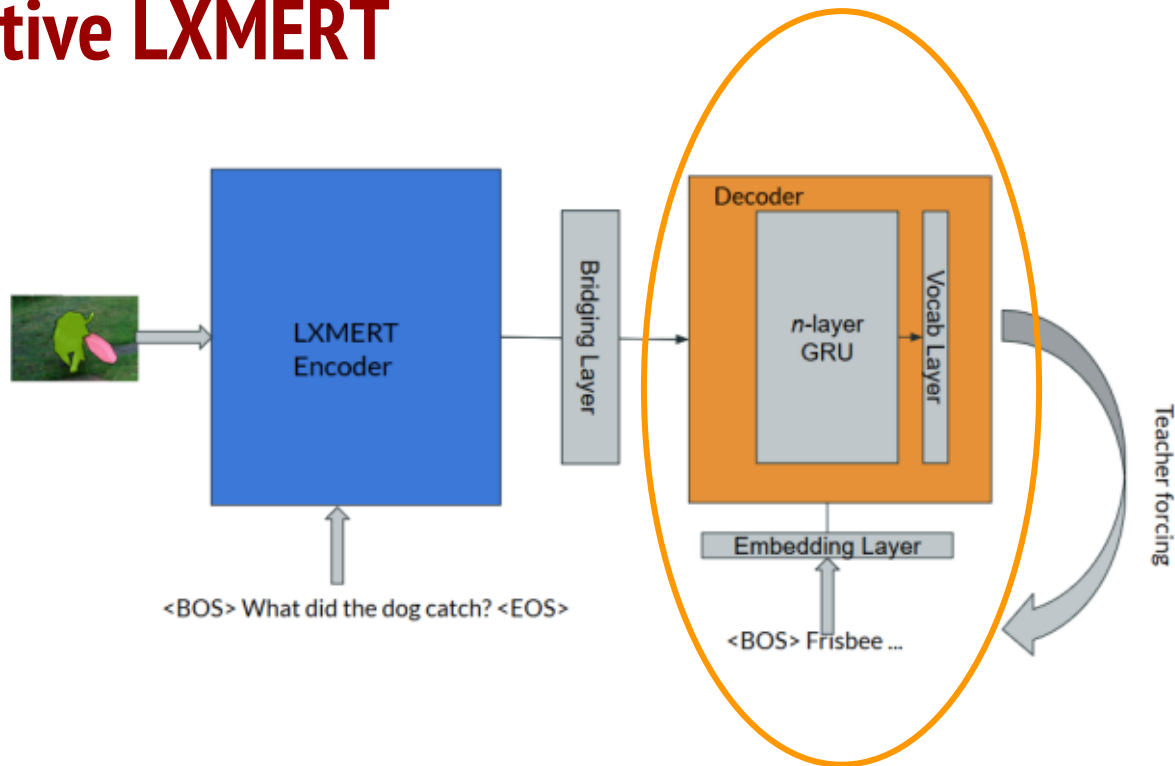




# Approach 2: Generative LXMERT

## Decoder:

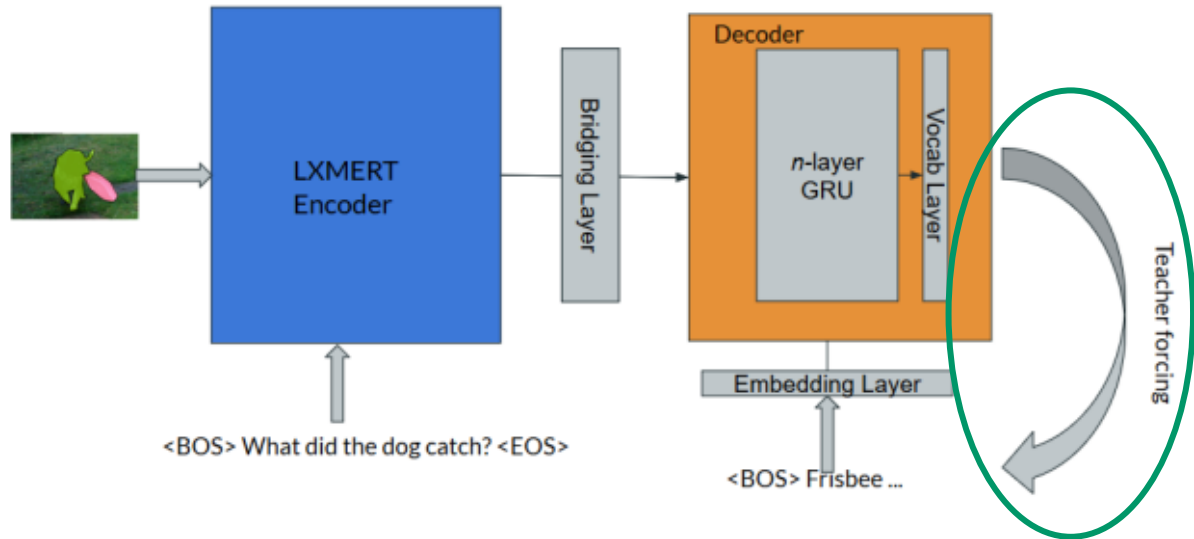
- 300-dimensional word embedding layer
- 3-layer GRU RNN
- Vocabulary layer of size  $|V|$ , which is the size of the vocabulary containing all the words in questions and answers in the training set



# Approach 2: Generative LXMERT

## Teacher forcing:

Once token has been generated by the decoder, with probability  $r$  we set the input token for the next decoding step equal to the ground truth one.



## Approach 2: modifications and evaluation



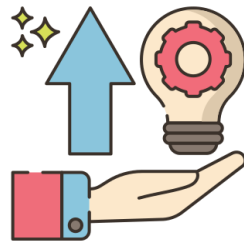
**Issue:** model always predicts [PAD] token after some time

We have tried several changes:

- Add *gradient clipping*
- *Weight* the [PAD] token in the loss computation
- Test different optimizers:
  - *Adam* → [PAD] token occurs too early during the training
  - *SGD* → MAX\_ANSW\_LEN random tokens prediction
  - *AdamW* → [PAD] token occurs later during the training

Overall Accuracy	0.00
Per Answer Type Accuracy is the following:	
other	0.00
yes/no	0.00
number	0.00

# Possible improvements (with more resources)



- ★ Train for more epochs
- ★ Use the entire dataset during training
- ★ Increase the batch size (accumulate\_grad\_batches has been already used)
- ★ (2) Fine-tune LXMERT and the embedding layers
- ★ (2) Try to combine word2vec and GloVe embeddings
- ★ (2) Use teacher forcing with a higher probability in the first epochs and stabilize the 0.5 probability after a certain number of epochs.



— **Thank you** —

---

---