CS 224N: TensorFlow Tutorial

Lecture and Live Demo

Nishith Khandwala & Barak Oshri

31 January, 2017

Intro to Deep Learning Frameworks

- Scales machine learning code
- Computes gradients!
- Standardizes machine learning applications for sharing
- Zoo of Deep Learning frameworks available with different advantages, paradigms, levels of abstraction, programming languages, etc
- Interface with GPUs for parallel processing

In some ways, rightfully gives Deep Learning its name as a separate *practice*

What is TensorFlow?

- Open source software library for numerical computation using data flow graphs
- Originally developed by Google Brain Team to conduct machine learning research
- “Tensorflow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms”

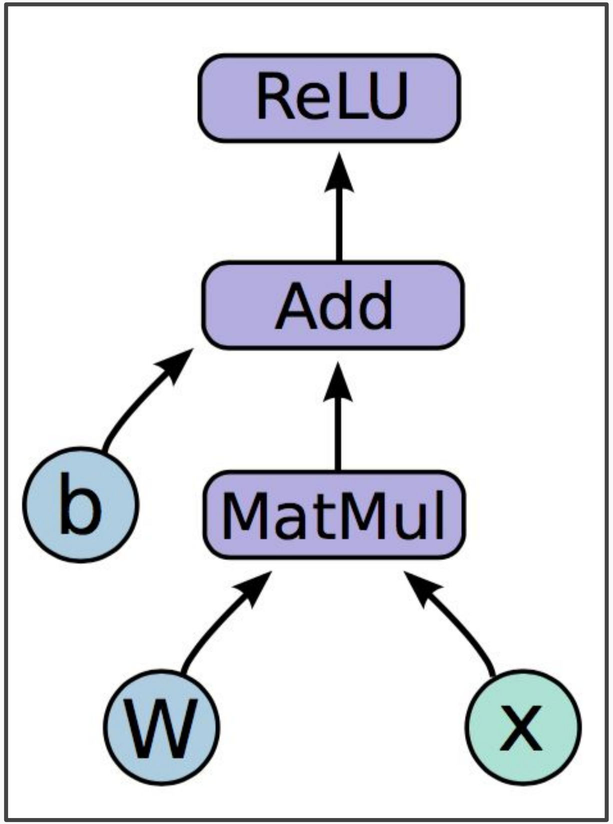
Programming model

Big idea: express a numeric computation as a **graph**.

- Graph nodes are **operations** which have any number of inputs and outputs
- Graph edges are **tensors** which flow between nodes

Programming model

$$h = ReLU(Wx + b)$$



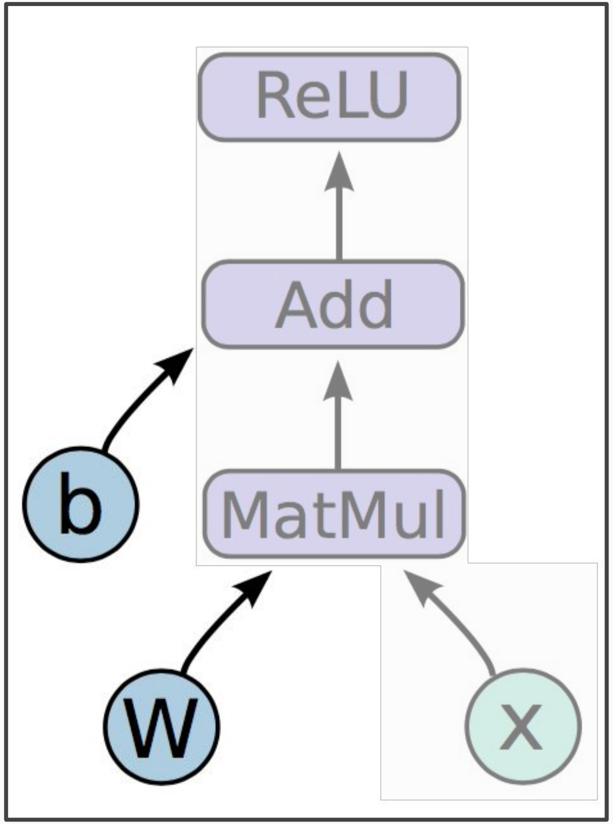
Programming model

$$h = ReLU(Wx + b)$$

Variables are stateful nodes which output their current value.

State is retained across multiple executions of a graph

(mostly parameters)

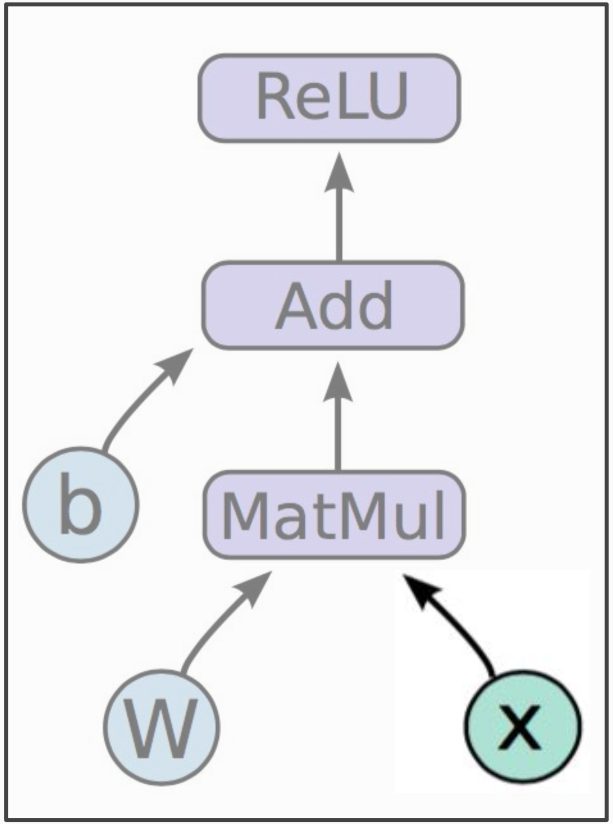


Programming model

$$h = ReLU(Wx + b)$$

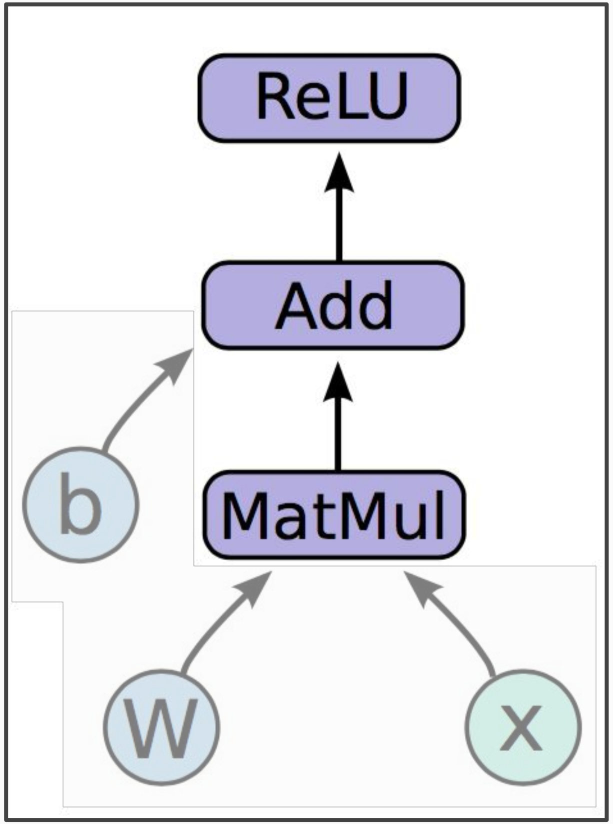
Placeholders are nodes whose value is fed in at execution time

(inputs, labels, ...)



Programming model

$$h = ReLU(Wx + b)$$



Mathematical operations:

MatMul: Multiply two matrix values.
Add: Add elementwise (with broadcasting).
ReLU: Activate with elementwise rectified linear function.

In code,

1. Create weights, including initialization

   W ~ *Uniform*(-1, 1); b = **0**

2. Create input placeholder x

   m * 784 input matrix

3. Build flow graph

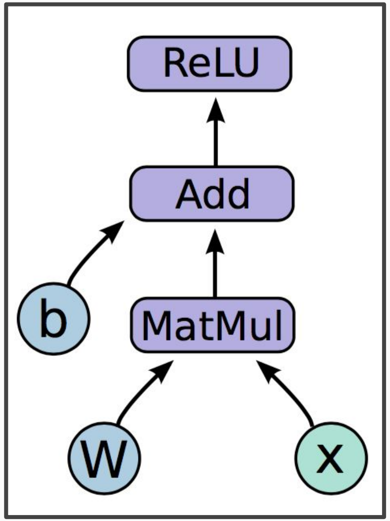
```
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100), -1, 1))

x = tf.placeholder(tf.float32, (100, 784))

h = tf.nn.relu(tf.matmul(x, W) + b)
```

$$h = ReLU(Wx + b)$$



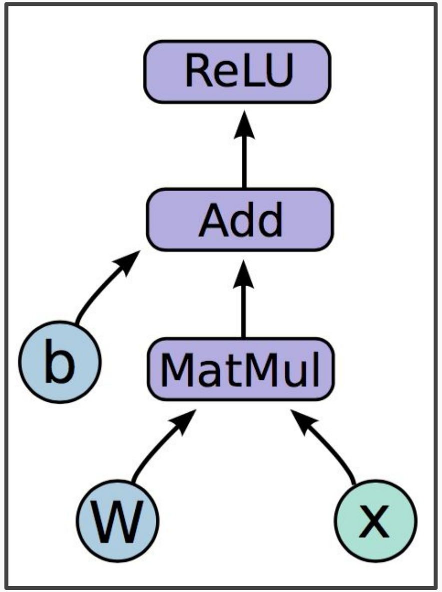
But where is the graph?

New nodes are automatically built into the underlying graph!

tf.get_default_graph().get_operations():

zeros/shape
zeros/Const
zeros
Variable
Variable/Assign
Variable/read
random_uniform/shape
random_uniform/min
random_uniform/max
random_uniform/RandomUniform
random_uniform/sub
random_uniform/mul
random_uniform
Variable_1
Variable_1/Assign
Variable_1/read
Placeholder
MatMul
add
Relu == h

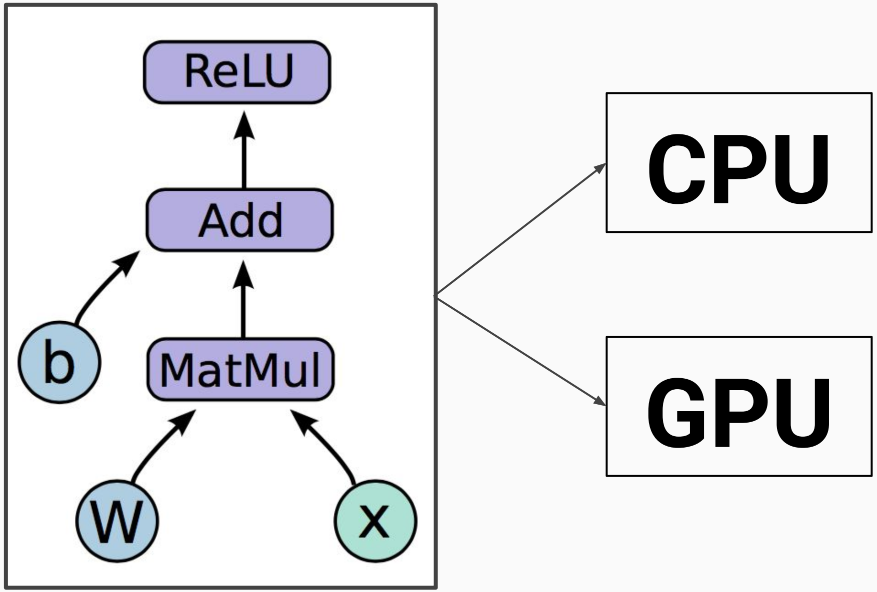
h refers to an op!



How do we run it?

So far we have defined a **graph**.

We can deploy this graph with a **session**: a binding to a particular execution context (e.g. CPU, GPU)



Getting output

```
sess.run(fetches, feeds)
```

Fetches: List of graph nodes. Return the outputs of these nodes.

Feeds: Dictionary mapping from graph nodes to concrete values. Specifies the value of each graph node given in the dictionary.

```python
import numpy as np
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100),
                 -1, 1))

x = tf.placeholder(tf.float32, (100, 784))
h = tf.nn.relu(tf.matmul(x, W) + b)

sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h, {x: np.random.random(100, 784)})
```

So what have we covered so far?

We first built a **graph** using **variables** and **placeholders**

We then deployed the graph onto a **session**, which is the **execution environment**

Next we will see how to **train** the **model**

How do we define the loss?

Use **placeholder** for **labels**

Build loss node using labels and **prediction**

```
prediction = tf.nn.softmax(...)  #Output of neural network
label = tf.placeholder(tf.float32, [100, 10])

cross_entropy = -tf.reduce_sum(label * tf.log(prediction), axis=1)
```

How do we compute Gradients?

```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

- `tf.train.GradientDescentOptimizer` is an **Optimizer** object
- `tf.train.GradientDescentOptimizer(lr).minimize(cross_entropy)` adds optimization **operation** to computation graph

TensorFlow graph **nodes** have **attached gradient operations**

Gradient with respect to **parameters** computed with **backpropagation**

...automatically

Creating the train_step op

```
prediction = tf.nn.softmax(...)
label = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(-tf.reduce_sum(label * tf.log(prediction),
reduction_indices=[1]))

train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Training the Model

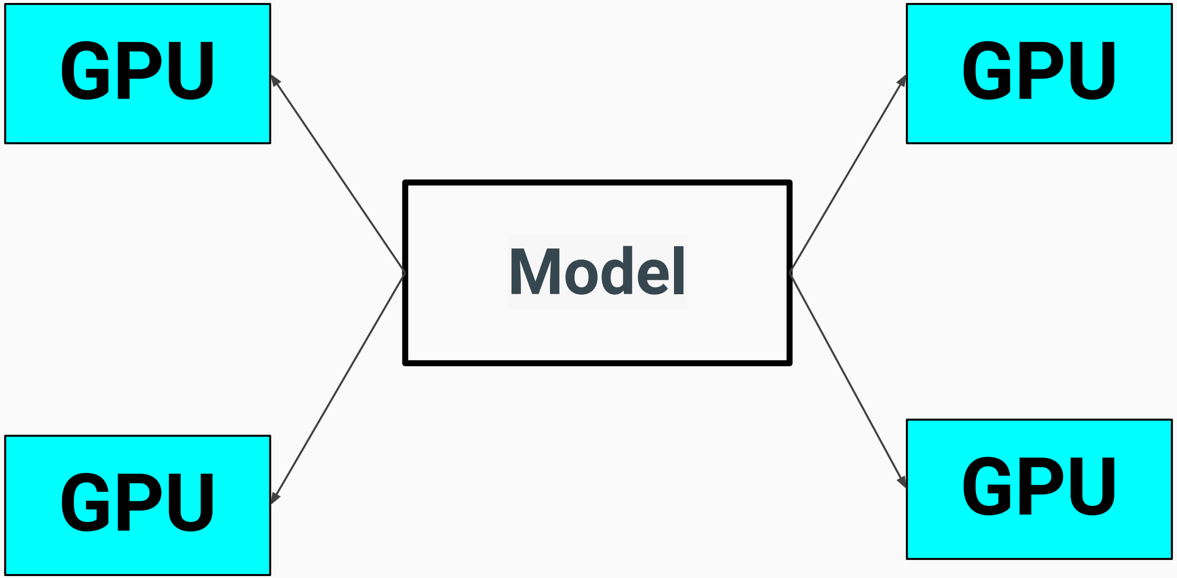
```
sess.run(train_step, feeds)
```

1. Create Session
2. Build training schedule
3. Run `train_step`

```
sess = tf.Session()
sess.run(tf.initialize_all_variables())

for i in range(1000):
    batch_x, batch_label = data.next_batch()
    sess.run(train_step, feed_dict={x: batch_x,
                                    label: batch_label}
```

Variable sharing



Variable sharing: naive way

```
variables_dict = {
    "weights": tf.Variable(tf.random_normal([782, 100]),
                           name="weights")
    "biases": tf.Variable(tf.zeros([100]), name="biases")
    }
```

Not good for encapsulation!

What’s in a Name?

`tf.variable_scope()` provides simple name-spacing to avoid clashes

`tf.get_variable()` creates/accesses variables from within a variable scope

```
with tf.variable_scope("foo"):
    v = tf.get_variable("v", shape=[1])  # v.name == "foo/v:0"

with tf.variable_scope("foo", reuse=True):
    v1 = tf.get_variable("v")      # Shared variable found!

with tf.variable_scope("foo", reuse=False):
    v1 = tf.get_variable("v")      # CRASH foo/v:0 already exists!
```

In Summary:

1. Build a graph
    a. Feedforward / Prediction
    b. Optimization (gradients and train_step operation)
2. Initialize a session
3. Train with `session.run(train_step, feed_dict)`

Acknowledgments

Jon Gauthier, Natural Language Processing Group, Symbolic Systems

Bharath Ramsundar, PhD Student, Drug Discovery Research

Chip Huyen, Undergraduate, teaching CS20SI: TensorFlow for Deep Learning Research!

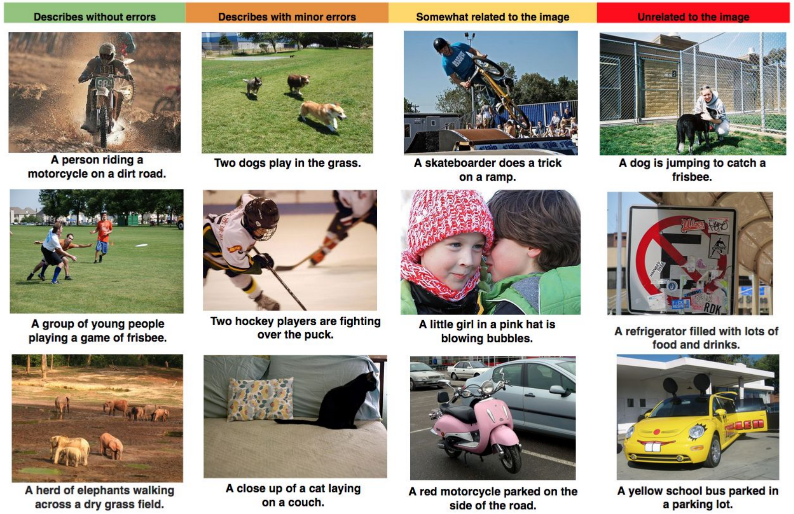
Visual Dialog

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, Dhruv Batra

Presented by: Alan Luo

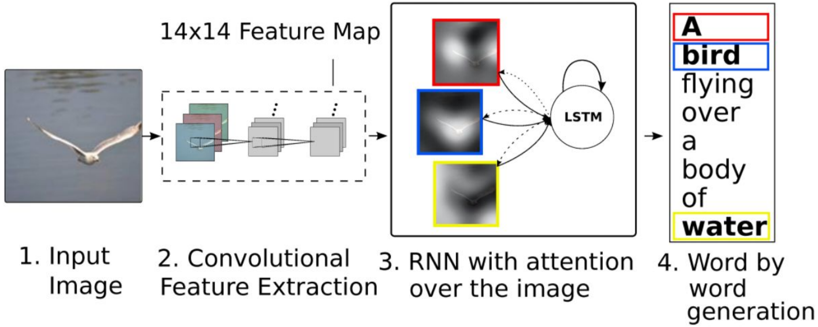
Introduction Natural Language Processing + Computer Vision

- Aiding visually impaired users in understanding their surroundings or social media content
- Interacting with an AI assistant



Related Work Image/Video Captioning

Image Captioning



a man is throwing a frisbee in a park

a man riding a wave on top of a surfboard

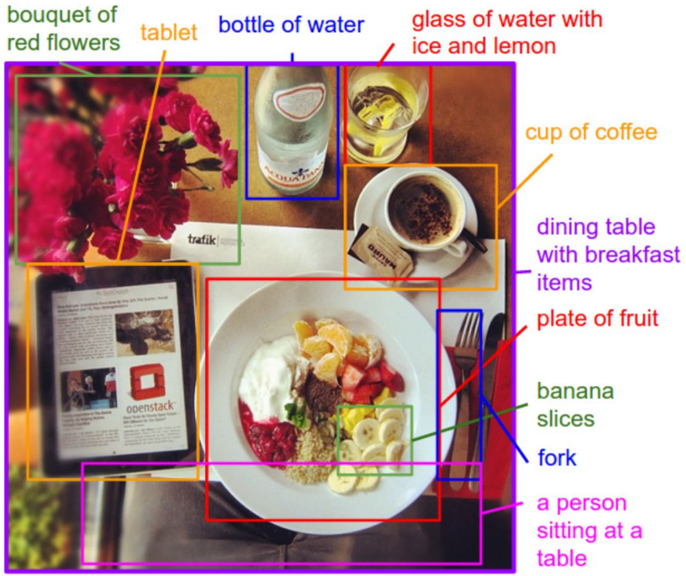
Video Summary

Input video:

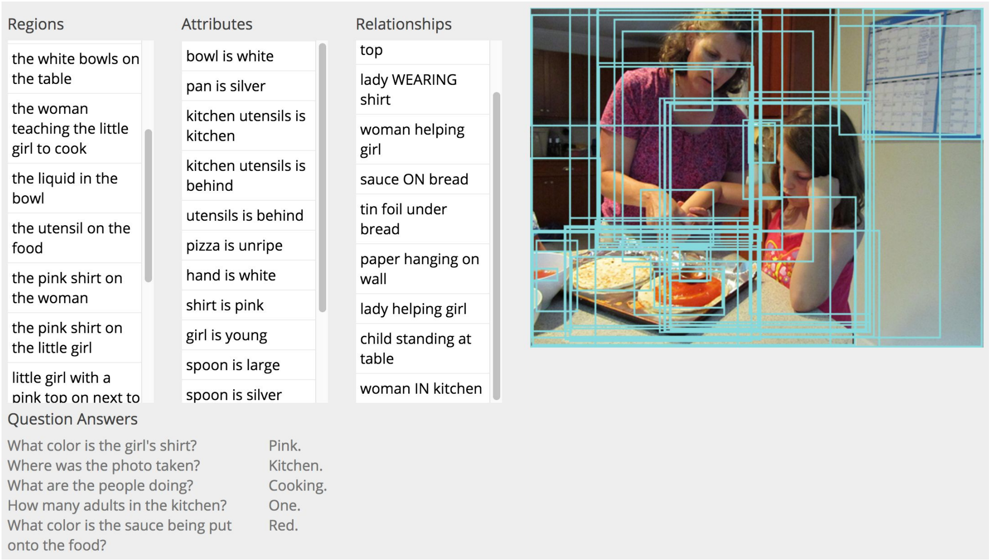
Our output: *A cat is playing with a toy.*
Humans: *A Ferret and cat fighting with each other. / A cat and a ferret are playing. / A kitten is playing with a ferret. / A kitten and a ferret are playfully wrestling.*

Related Work Visual-Semantic Alignments

Visual-Semantic Alignments



Datasets



Related Work Visual Q&A

How many pickles are on the plate?	1 1 1	1 1 1
What is the shape of the plate?	circle round round	circle round round

What does the sign say?	stop stop stop	stop stop yield
What shape is this sign?	octagon octagon octagon	diamond octagon round

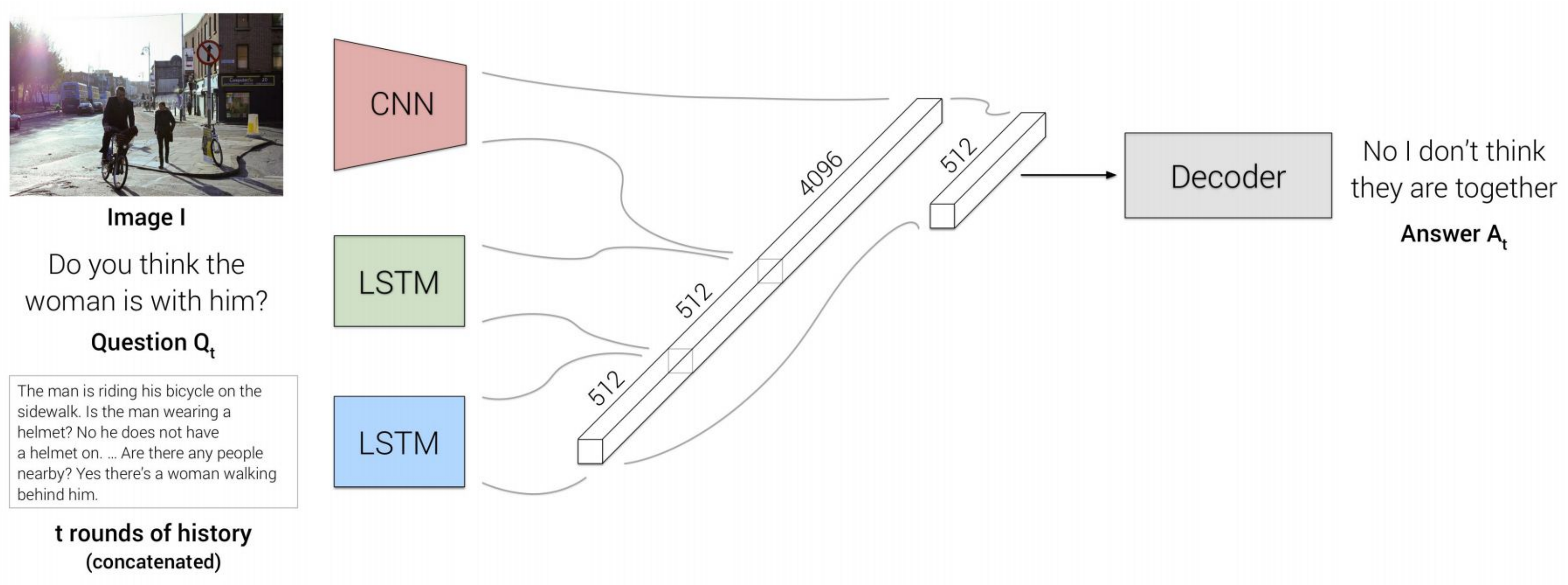
Contributions

1. Propose a new AI task: Visual Dialog

2. Develop a novel two-person chat data-collection protocol and introduce a new dataset

3. Introduce a family of neural encoder-decoder models for Visual Dialog

Technical Details With Late Fusion Encoder



Dataset VisDial

Qualitative

Caption: A statue depicting a bear breaking into a car.

Person A (1): how big is statue

Person B (1): about size of real full grown bear

Person A (2): so is car full size then as well

Person B (2): yes replica of car

Person A (3): is statue all 1 color

Person B (3): no brown and black

Person A (4): what color is car

Person B (4): dark red

Person A (5): where is this, do you think

Person B (5): in wooded area someplace

Person A (6): do you see any people in image

Person B (6): yes 1 man

Person A (7): how old is man

Person B (7): 35-40

Person A (8): what is man doing

Person B (8): sitting in car behind replica

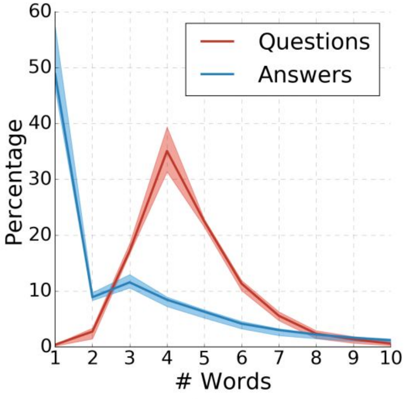
Person A (9): do you see any signs

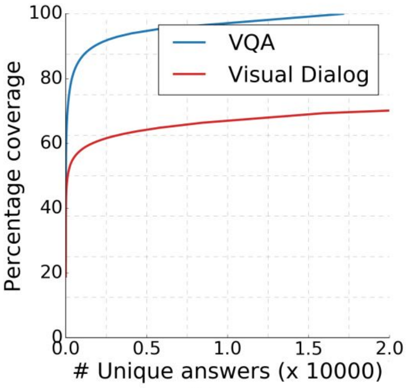
Person B (9): yes, on car door warning sign

Person A (10): what else can you tell me about this image

Person B (10): there are many trees in background

Quantitative





Results

Qualitative Results

Quantitative Results

	Model	MRR	R@1	R@5	R@10	Mean
Baseline	Answer prior	0.311	19.85	39.14	44.28	31.56
	NN-Q	0.392	30.54	46.99	49.98	30.88
	NN-QI	0.385	29.71	46.57	49.86	30.90
Generative	LF-Q-G	0.403	29.74	50.10	56.32	24.06
	LF-QH-G	0.425	32.49	51.56	57.80	23.11
	LF-QI-G	0.437	34.06	52.50	58.89	22.31
	LF-QIH-G	0.430	33.27	51.96	58.09	23.04
	HRE-QH-G	0.430	32.84	52.36	58.64	22.59
	HRE-QIH-G	0.442	34.37	53.40	59.74	21.75
	HREA-QIH-G	0.442	34.47	53.43	59.73	21.83
	MN-QH-G	0.434	33.12	53.14	59.61	22.14
	MN-QIH-G	**0.443**	**34.62**	**53.74**	**60.18**	**21.69**
Discriminative	LF-Q-D	0.482	34.29	63.42	74.31	8.87
	LF-QH-D	0.505	36.21	66.56	77.31	7.89
	LF-QI-D	0.502	35.76	66.59	77.61	7.72
	LF-QIH-D	0.511	36.72	67.46	78.30	7.63
	HRE-QH-D	0.489	34.74	64.25	75.40	8.32
	HRE-QIH-D	0.502	36.26	65.67	77.05	7.79
	HREA-QIH-D	0.508	36.76	66.54	77.75	7.59
	MN-QH-D	0.524	36.84	67.78	78.92	7.25
	MN-QIH-D	**0.529**	**37.33**	**68.47**	**79.54**	**7.03**
VQA	SAN1-QI-D	0.506	36.21	67.08	78.16	7.74
	HieCoAtt-QI-D	0.509	35.54	66.79	77.94	7.68
	Human Accuracies					
Human	Human-Q	0.441	25.10	67.37	-	4.19
	Human-QH	0.485	30.31	70.53	-	3.91
	Human-QI	0.619	46.12	82.54	-	2.92
	Human-QIH	0.635	48.03	83.76	-	2.83