Go Cognitive with Python

TensorFlow

amazon alexa

python™

Go Cognitive with Python

Your buddy

Rahul Kumar

rahul.kumar@happiestminds.com
Happiest Minds Technologies Pvt. Ltd.

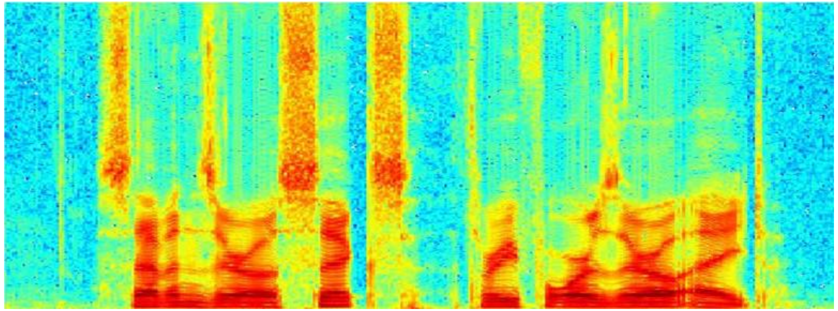
Setup --install all the things

- Clone or download this repo : http://tinyurl.com/hybub7a
- Install Anaconda for Python 2.7 : https://www.continuum.io/downloads
- Install Tensorflow : https://www.tensorflow.org/get_started/os_setup
- Windows installation:

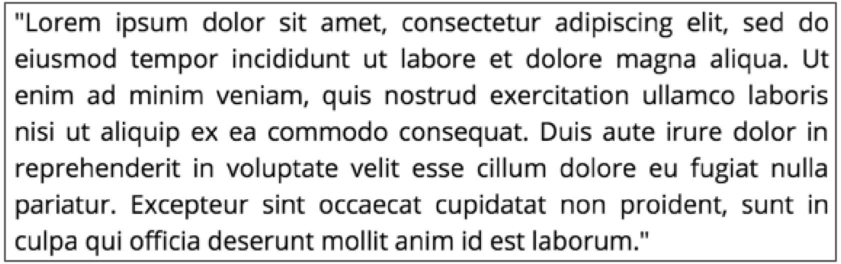
```
conda create --name tensorflow python=3.5
activate tensorflow
conda install jupyter
conda install scipy
pip install tensorflow
```

- Install Flask-ask : `pip install flask-ask`
- Setup Amazon developer account: https://developer.amazon.com/edw/home.html

In many real-world applications input data have **structure**.



Voice: Spectrograms



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Text

Images

Let's get COGNITIVE in

Image Processing

Natural Language Processing

Speech Processing

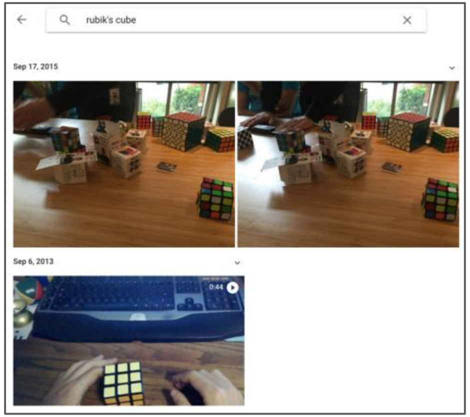
TensorFlow

python™

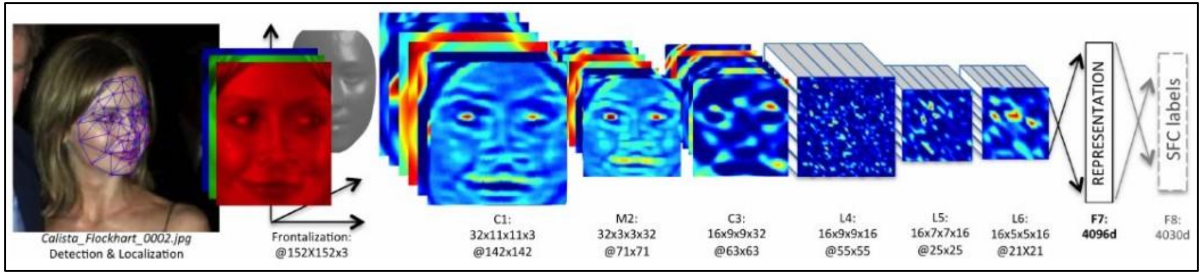
Cognitive:
Image Processing

Refer: /Session1-ImageProcessing/

Artificial Intelligence is everywhere…



e.g. Google Photos search



Face Verification, Taigman et al. 2014 (FAIR)

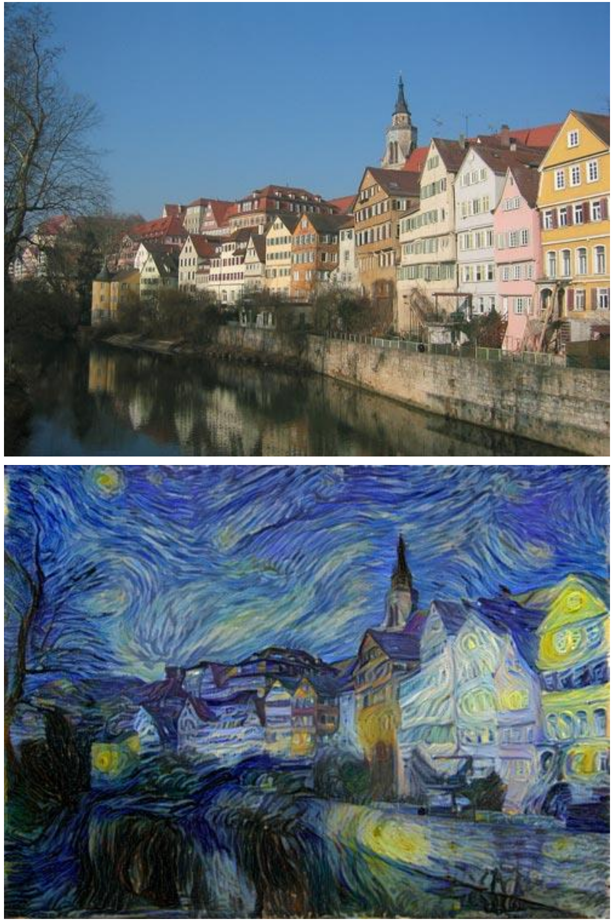
[Goodfellow et al. 2014]

Self-driving cars

WaveNet, van den Oord et al. 2016

Artificial Intelligence is everywhere…

DeepDream *reddit.com/r/deepdream*

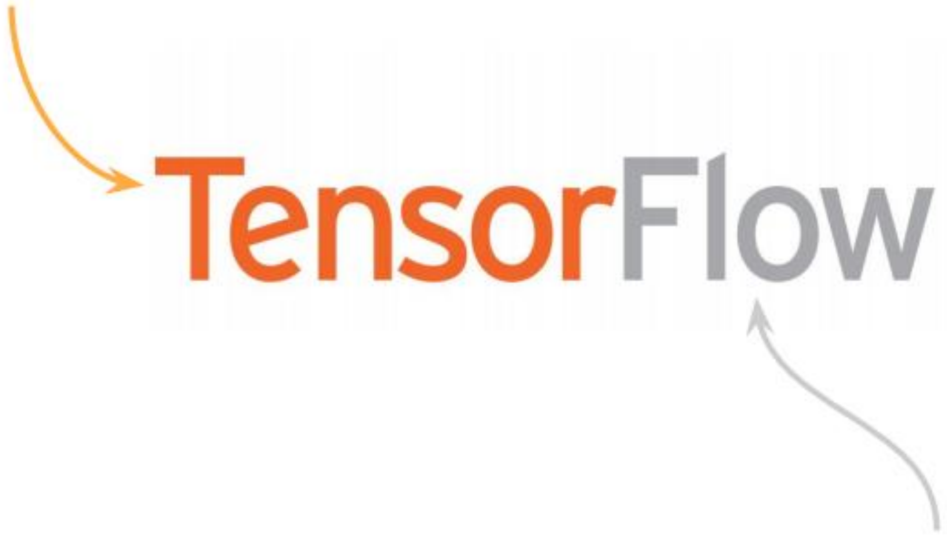


NeuralStyle, Gatys et al. 2015
deepart.io, Prisma, etc.

What’s TensorFlow?

- Open source Machine Learning library
- Especially useful for Deep Learning
- For research and production
- Apache 2.0 license
- www.tensorflow.org

A multidimensional array.

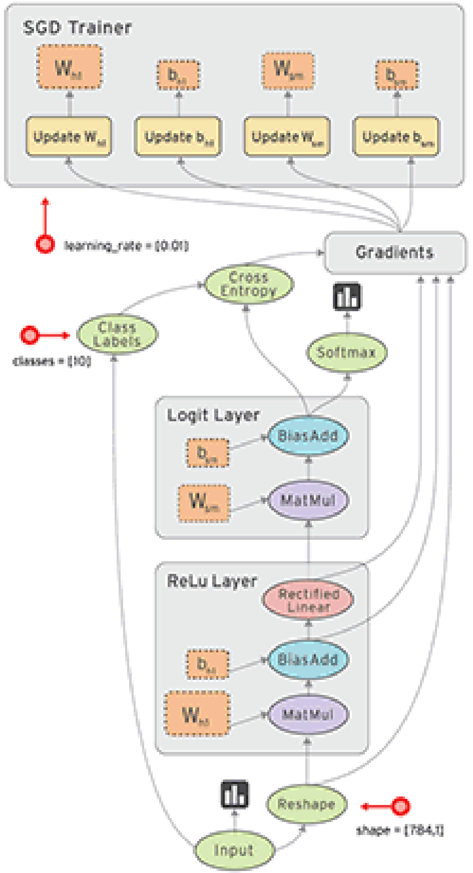


A graph of operations.

Operates over **tensors**: n-dimensional arrays

Using a **flow graph**: data flow computation framework

- Flexible, intuitive construction
- automatic differentiation
- Support for threads, queues, and asynchronous computation; distributed runtime
- Train on CPUs, GPUs, ...and coming soon, TPUS...
- Run wherever you like: Linux, Windows, OSX

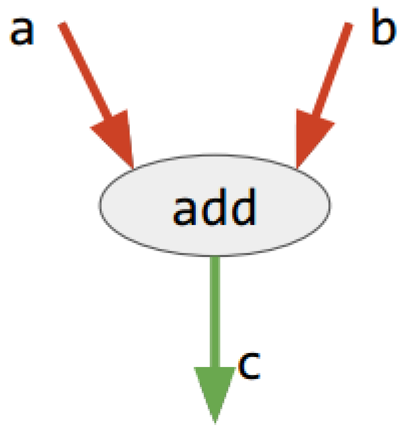


Hello World !!!

with TensorFlow

`Refer: /Session1-ImageProcessing/Hello_tensorflow/`

Build a graph; then run it.



```
...
c = tf.add(a, b)

...

session = tf.Session()

value_of_c = session.run(c, {a=1, b=2})
```

Refer: /Session1-ImageProcessing/Hello_tensorflow/

Give me {code}

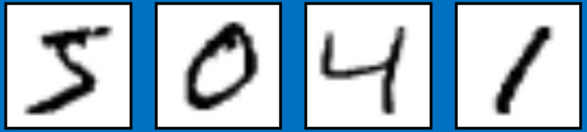
Refer: /Session1-ImageProcessing/Hello_tensorflow/

Handwriting Recognition

with TensorFlow

`Refer: /Session1-ImageProcessing/MNIST/`

--using MNIST dataset



http://colah.github.io/posts/2014-10-Visualizing-MNIST/

TensorFlow : Handwriting recognition

```
# Import MINST data
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets(‘data_dir’, one_hot=True)
```

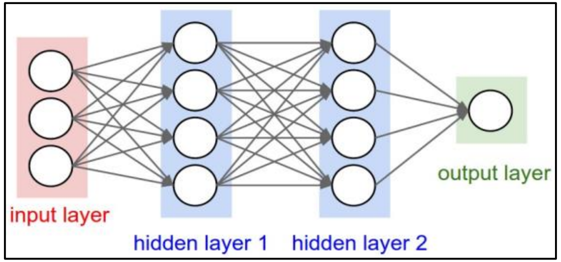
Load library and MNIST data

TensorFlow : Handwriting recognition

```
# Import MINST data
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets(‘data_dir’, one_hot=True)
```

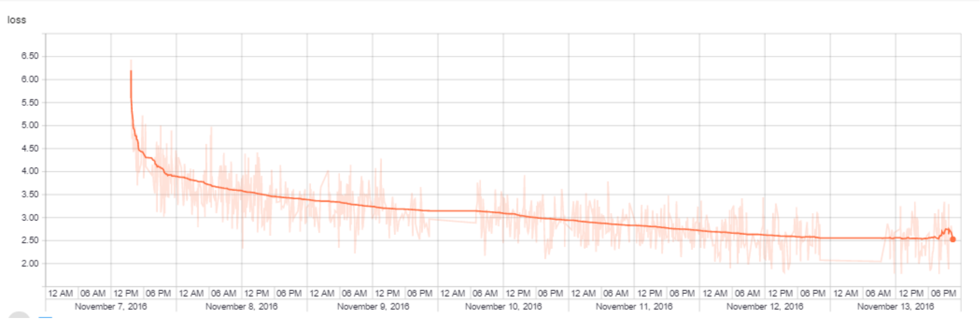
```
# Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b
```

Design neural network architecture



TensorFlow : Handwriting recognition

```
# Import MINST data
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets(‘data_dir’, one_hot=True)

# Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b
```



```
# Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y, y_))
  train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Select optimization algorithm

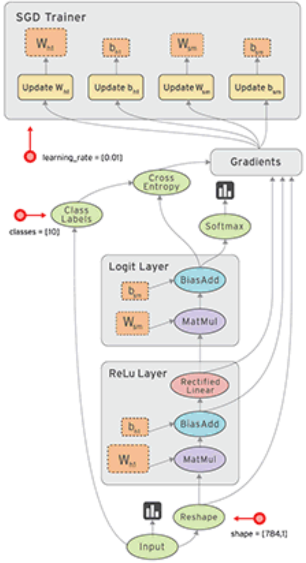
TensorFlow : Handwriting recognition

```
# Import MINST data
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets(‘data_dir’, one_hot=True)

# Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b

# Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y, y_))
  train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

  sess = tf.InteractiveSession()
  tf.global_variables_initializer().run()
```

Initialize the session and variables

TensorFlow : Handwriting recognition

```
# Import MINST data
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets(‘data_dir’, one_hot=True)

# Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b

# Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y, y_))
  train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

  sess = tf.InteractiveSession()
  tf.global_variables_initializer().run()
 # Train
  for _ in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```



Train the model

TensorFlow : Handwriting recognition

```
# Import MINST data
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets(‘data_dir’, one_hot=True)

# Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b

# Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y, y_))
  train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

  sess = tf.InteractiveSession()
  tf.global_variables_initializer().run()
 # Train
  for _ in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

90.3%

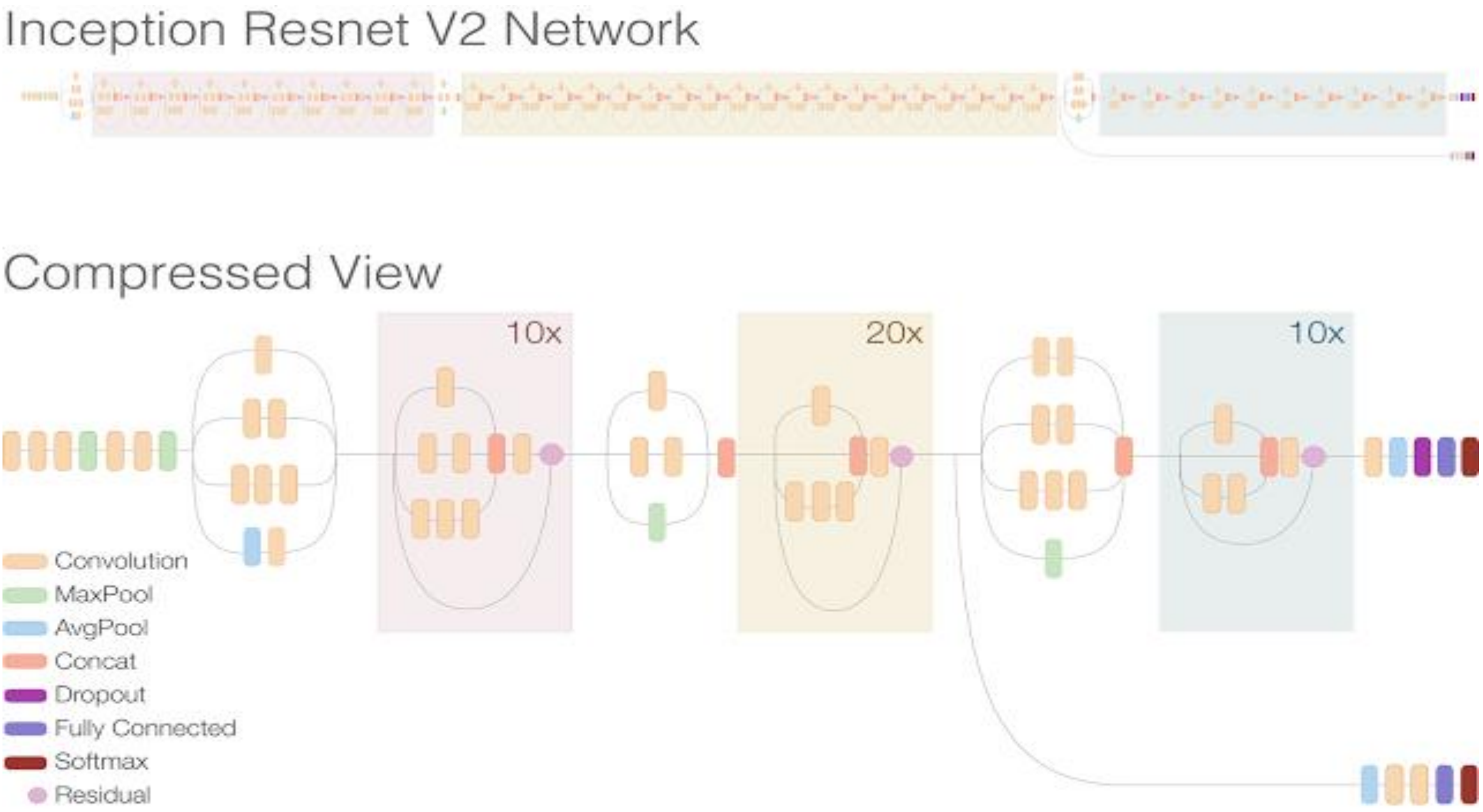
Give me {code}

Refer: /Session1-ImageProcessing/MNIST/

Open Source Models

github.com/tensorflow/models

Inception Model

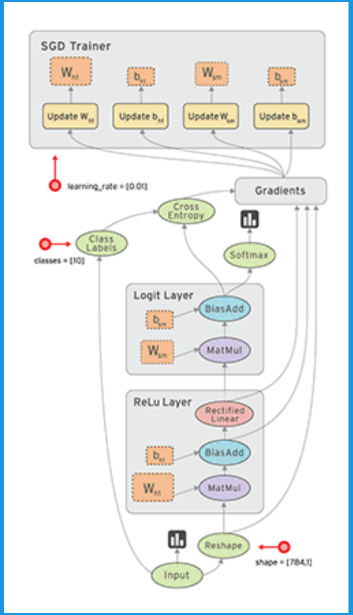


Follow: https://research.googleblog.com/2016/08/improving-inception-and-image.html

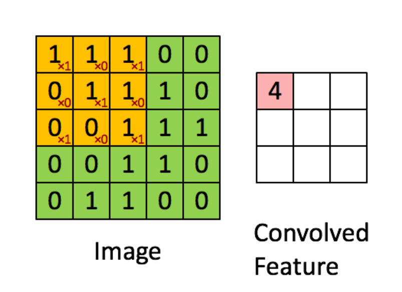
Inception Model

An Alaskan Malamute (left) and a Siberian Husky (right). Images from Wikipedia

TensorFlow



Convolutional NN



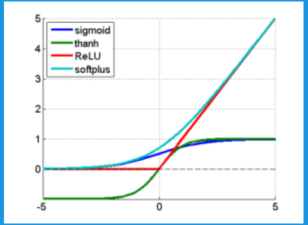
$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

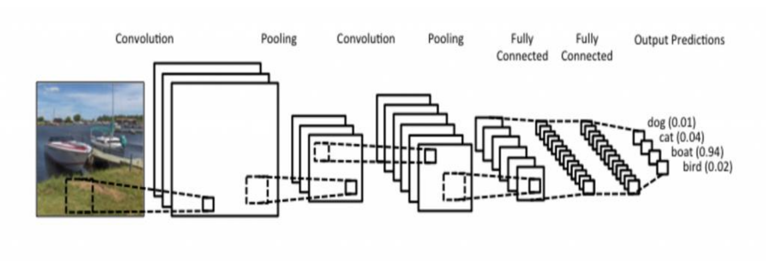
4		

Convolved Feature

Convolution Operation



Activation functions



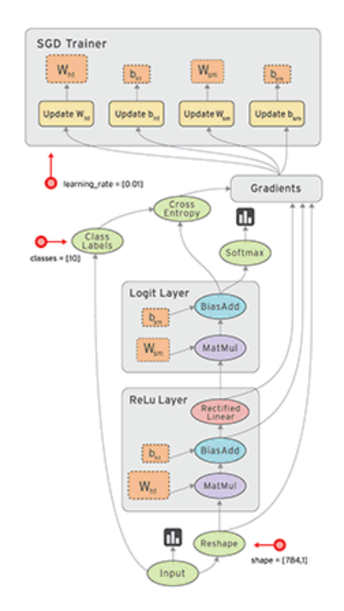
Give me {code}

Refer: /Session1-ImageProcessing/Inception/

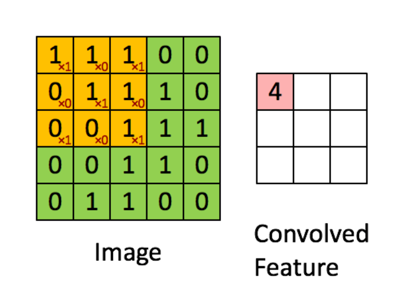
Cognitive: Natural Language Processing

Refer: /Session2-NaturalLanguageProcessing/

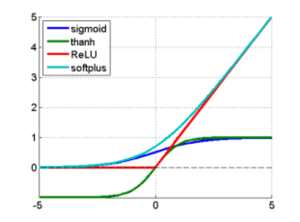
TensorFlow



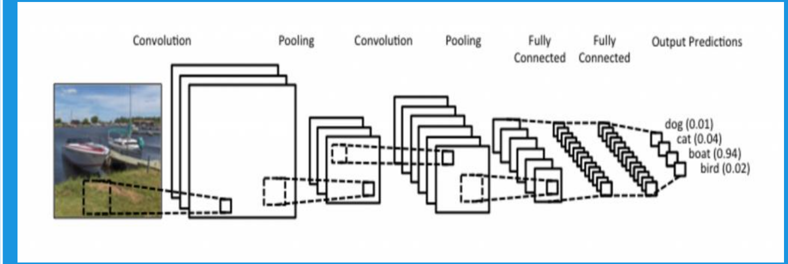
Convolutional NN



Convolution Operation

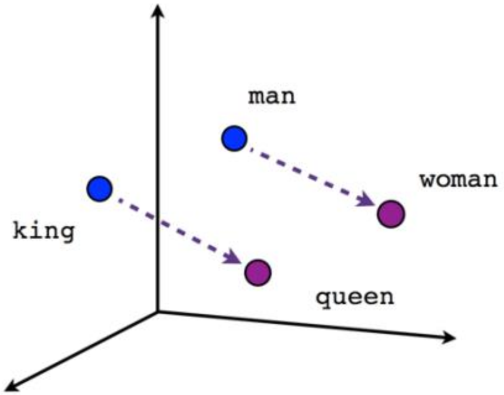


Activation functions

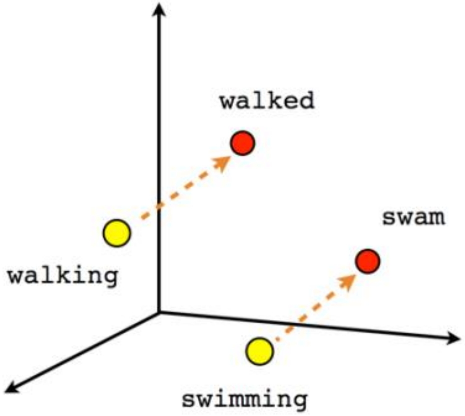


Word2vec

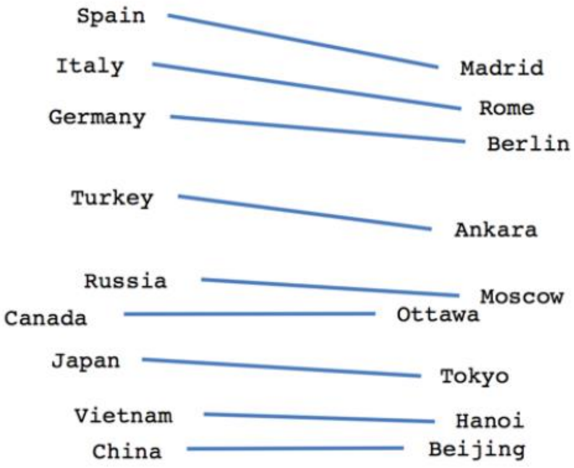
Word2vec



Male-Female



Verb tense



Country-Capital

https://www.tensorflow.org/tutorials/word2vec/

Give me {code}

Refer: \Session2-NaturalLanguageProcessing\word2vec

Text Classification

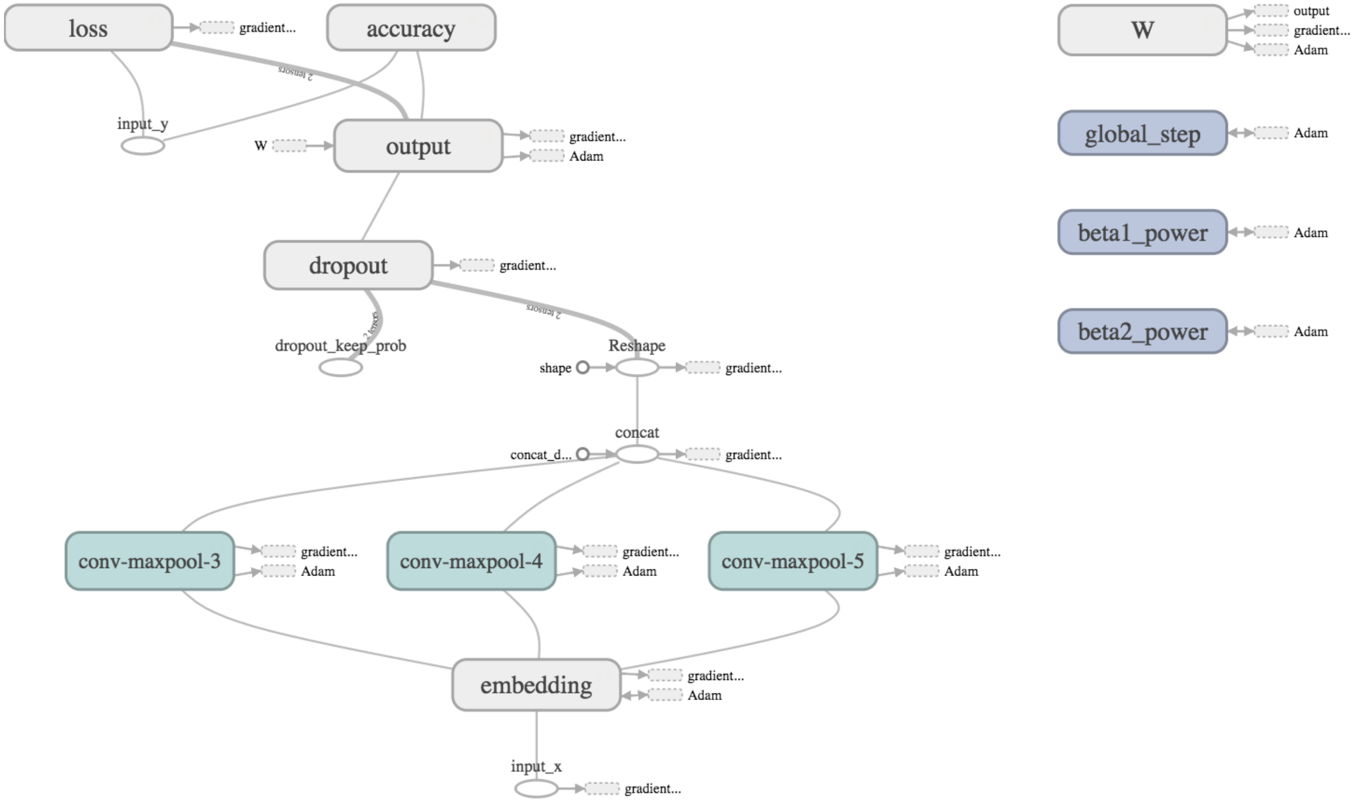
with TensorFlow

`Refer: /Session2-NaturalLanguageProcessing/CNN/`

--using reddit and twitter dataset

https://storage.googleapis.com/oscon-tf-workshop-materials/processed_reddit_data/news_aww/prepared_data.tar.gz

Text Classification using CNN



Give me {code}

Refer: \Session2-NaturalLanguageProcessing\CNN

Cognitive: Speech Processing

Refer: /Session3-AudioProcessing/

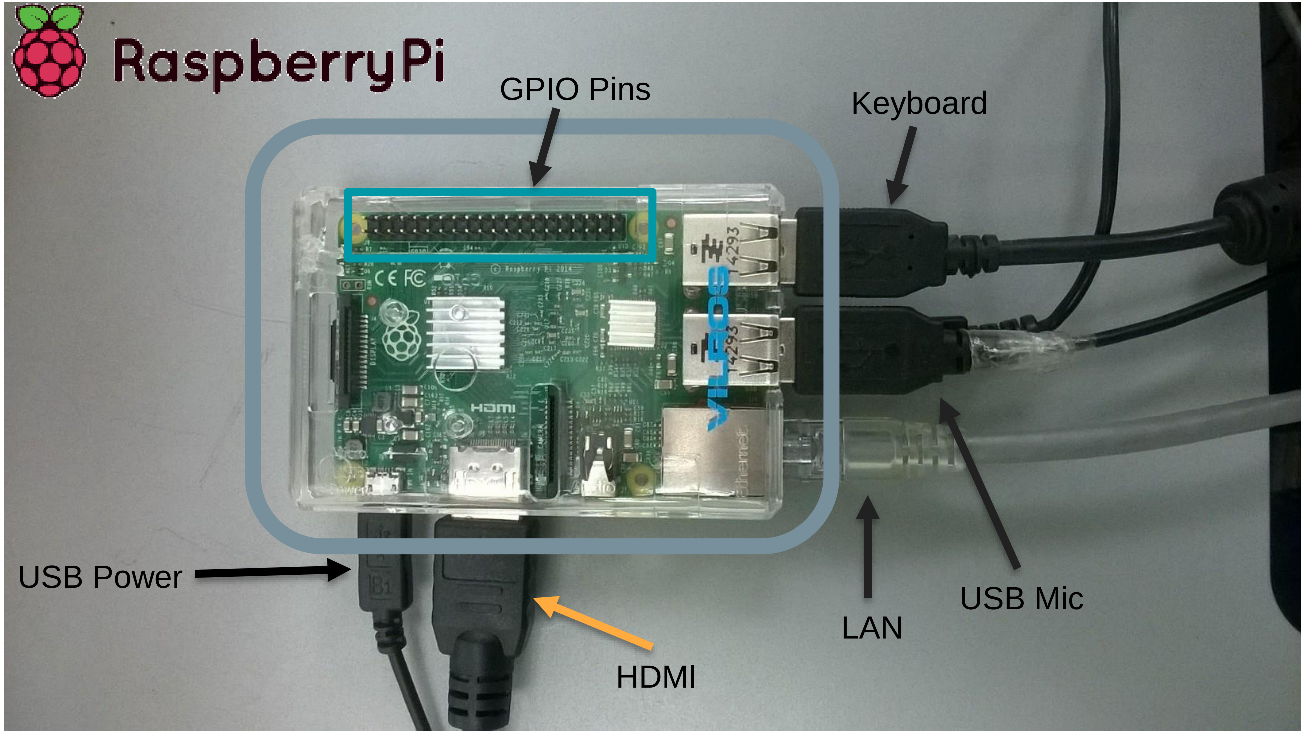
Conversational BOTs

with Amazon Alexa

`Refer: /Session3-AudioProcessing`

--using Flask-ask

AlexBot



Refer: https://github.com/goodrahstar/ALexBot

Give me {code}

Refer: \Session3-AudioProcessing\

"Deep learning" neural networks offer us great power - and pose unique risks. Can we solve for this?

Thank you!