IT8303
AI HUMAN INTERFACE

Lab 1
Introduction to Deep Learning

What you will learn / do in this lab

1. *Explore Deep Learning concepts and applications*
2. *Conduct an Image application experiment using Deep Learning*
3. *Conduct a Sentiment Analysis experiment using Deep Learning*

TABLE OF CONTENTS

1.
OVERVIEW

In this practical we will be exploring what is deep learning. We will also be running some online demonstrations that use deep learning on data that is unstructured (no rows, no columns, no attributes, no labels).

INTRODUCTION TO DEEP LEARNING

Deep learning is part of a broader family of machine learning methods based on **learning data representations**, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning neural networks have many layers (8 or more layers).

Deep learning is a class of machine learning algorithms that:

- *use a cascade of multiple layers of nonlinear processing units for **feature extraction** and **transformation**. Each successive layer uses the output from the previous layer as input.*
- *learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.*
- ***learn multiple levels of representations** that correspond to different levels of abstraction; the levels form a **hierarchy of concepts**.*

APPLICATIONS OF DEEP LEARNING

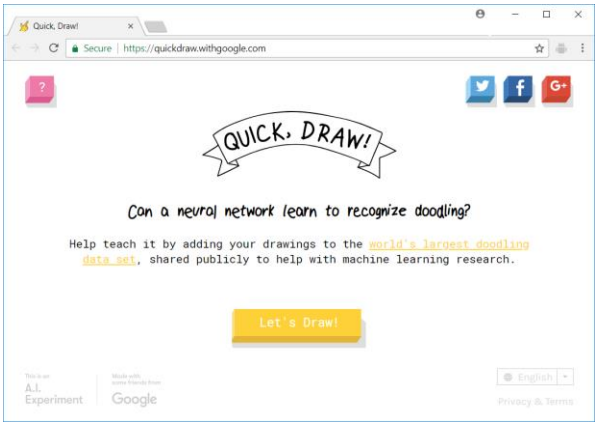
Deep learning stands out in its ability to handle large amounts of data and for unstructured data such as video, images, speech, audio and text in natural language.

The following are a list of top applications for Deep Learning:

- *Automatic speech recognition: Voice Search & Voice-Activated Assistants*
- *Video analysis: Self-driving cars*
- *Image recognition*
- *Image restoration: super-resolution, automatic colourization of black and white photos*
- *Natural language processing: Sentiment analysis*
- *Automatic Machine Translation*
- *Recommendation systems*
- *Question answering system: Chatbots*
- *Automatic Image Caption Generation*

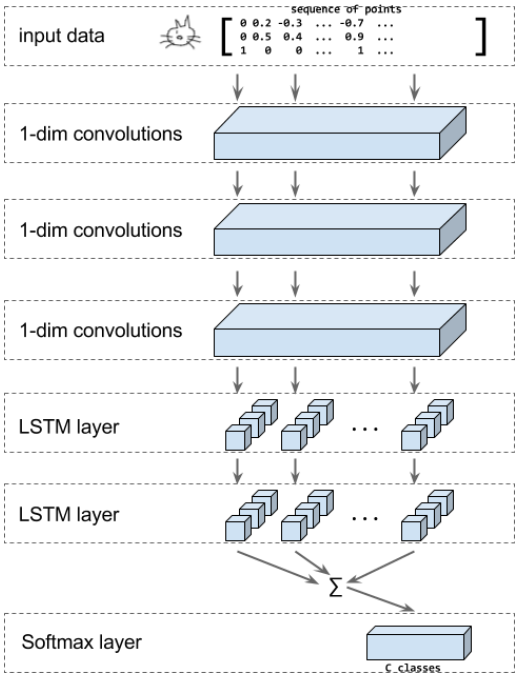
Activity

- *Take 15 minutes to explore this application.*
- *Use URL: https://quickdraw.withgoogle.com/*



How it works:

Under the hood, the model takes sequences of strokes your draw on the canvas and feed into a combination of convolution layer and recurrent network. Finally, the class digits will be generated from the softmax output layer. And here is the illustration of the structure of the model.



2.
IMAGE APPLICATIONS

In the section, we will explore two application areas of deep learning related to images.

IMAGE RECOGNITION

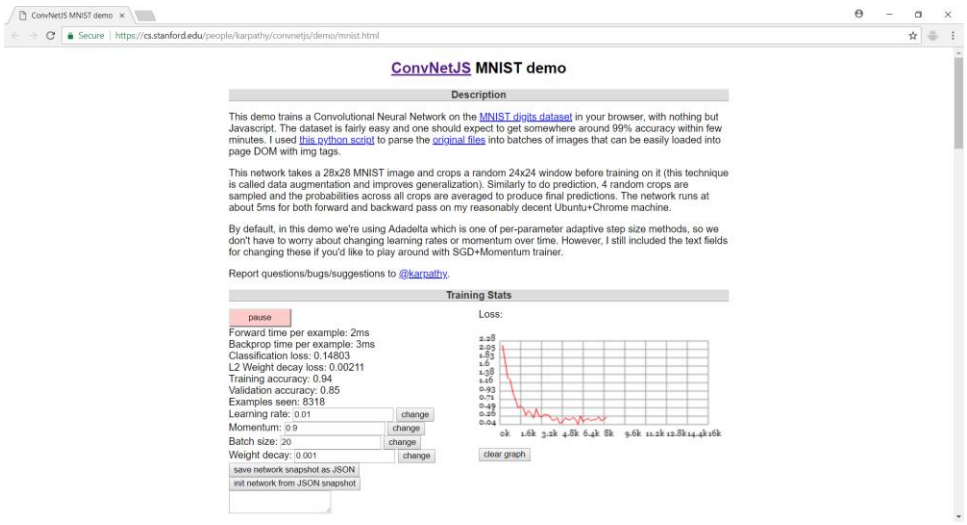
Image recognition, in the context of machine vision, is the ability of software to identify objects, places, people, writing and actions in images.

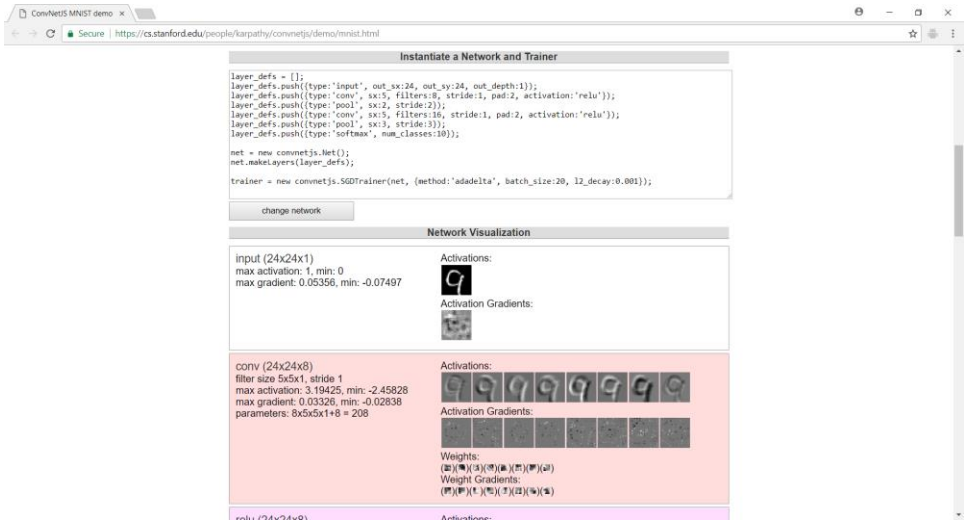
Image recognition is used to perform a large number of machine-based visual tasks, such as labeling the content of images with meta-tags, performing image content search and guiding autonomous robots, self-driving cars and accident avoidance systems.

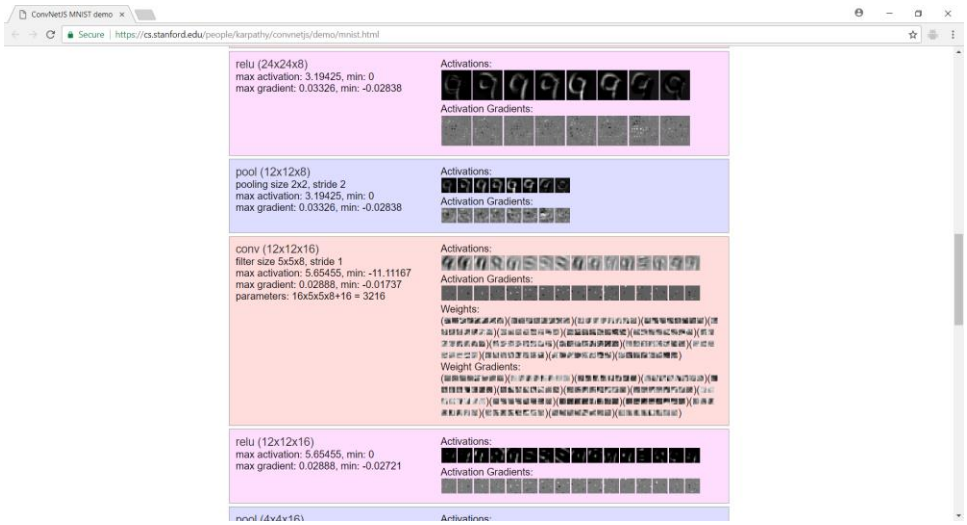
A common evaluation set for image classification is the MNIST database data set. MNIST is composed of handwritten digits and includes 60,000 training examples and 10,000 test examples.

Activity

- *Take 15 minutes to explore this application.*
- *Use URL: https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html*


ConvNetJS MNIST demo
Secure | https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html
ConvNetJS MNIST demo
Description
This demo trains a Convolutional Neural Network on the MNIST digits dataset in your browser, with nothing but Javascript. The dataset is fairly easy and one should expect to get somewhere around 99% accuracy within few minutes. I used this python script to parse the original files into batches of images that can be easily loaded into page DOM with img tags.
This network takes a 28x28 MNIST image and crops a random 24x24 window before training on it (this technique is called data augmentation and improves generalization). Similarly to do prediction, 4 random crops are sampled and the probabilities across all crops are averaged to produce final predictions. The network runs at about 5ms for both forward and backward pass on my reasonably decent Ubuntu+Chrome machine.
By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.
Report questions/bugs/suggestions to @karpathy.
Training Stats
pause
Forward time per example: 2ms
Backprop time per example: 3ms
Classification loss: 0.14803
L2 Weight decay loss: 0.00211
Training accuracy: 0.94
Validation accuracy: 0.85
Examples seen: 8318
Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change
save network snapshot as JSON
init network from JSON snapshot
Loss:
clear graph


ConvNetJS MNIST demo
Secure | https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html
Instantiate a Network and Trainer
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'softmax', num_classes:10});
net = new convnetjs.Net();
net.makeLayers(layer_defs);
trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20, l2_decay:0.001});
change network
Network Visualization
input (24x24x1)
max activation: 1, min: 0
max gradient: 0.05356, min: -0.07497
Activations:
Activation Gradients:
conv (24x24x8)
filter size 5x5x1, stride 1
max activation: 3.19425, min: -2.45828
max gradient: 0.03326, min: -0.02838
parameters: 8x5x5x1+8 = 208
Activations:
Activation Gradients:
Weights:
Weight Gradients:


ConvNetJS MNIST demo
Secure | https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html
relu (24x24x8)
max activation: 3.19425, min: 0
max gradient: 0.03326, min: -0.02838
Activations:
Activation Gradients:
pool (12x12x8)
pooling size 2x2, stride 2
max activation: 3.19425, min: 0
max gradient: 0.03326, min: -0.02838
Activations:
Activation Gradients:
conv (12x12x16)
filter size 5x5x8, stride 1
max activation: 5.65455, min: -11.11167
max gradient: 0.02888, min: -0.01737
parameters: 16x5x5x8+16 = 3216
Activations:
Activation Gradients:
Weights:
Weight Gradients:
relu (12x12x16)
max activation: 5.65455, min: 0
max gradient: 0.02888, min: -0.02721
Activations:
Activation Gradients:

Note: Use chrome browser. Clear browser cache and reload if it does not start properly.

How it works:

ConvNetJS is a Javascript library for training Deep Learning models (Neural Networks) entirely in your browser. See documentation at: https://cs.stanford.edu/people/karpathy/convnetjs/docs.html

The network use for the demo is given by

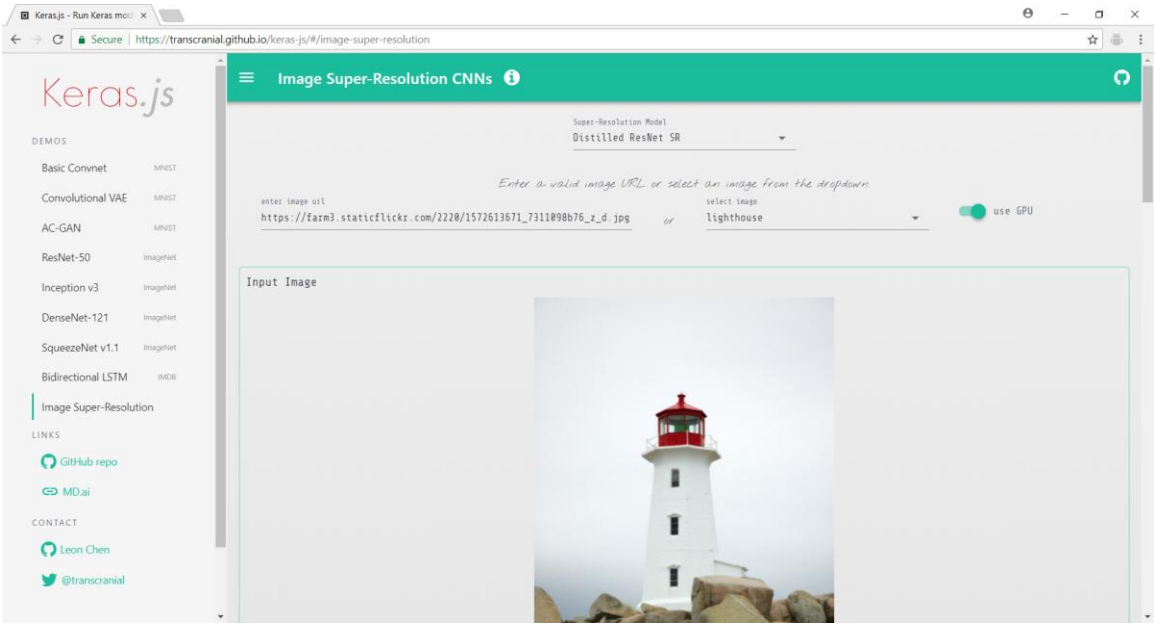
```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2,
        activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2,
        activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);

trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20,
        l2_decay:0.001});
```

IMAGE RESTORATION

View and edit this document in Word on your computer, tablet, or phone. You can edit text; easily insert content such as pictures, shapes, or tables; and seamlessly save the document to the cloud from Word on your Windows, Mac, Android, or iOS device.

Activity

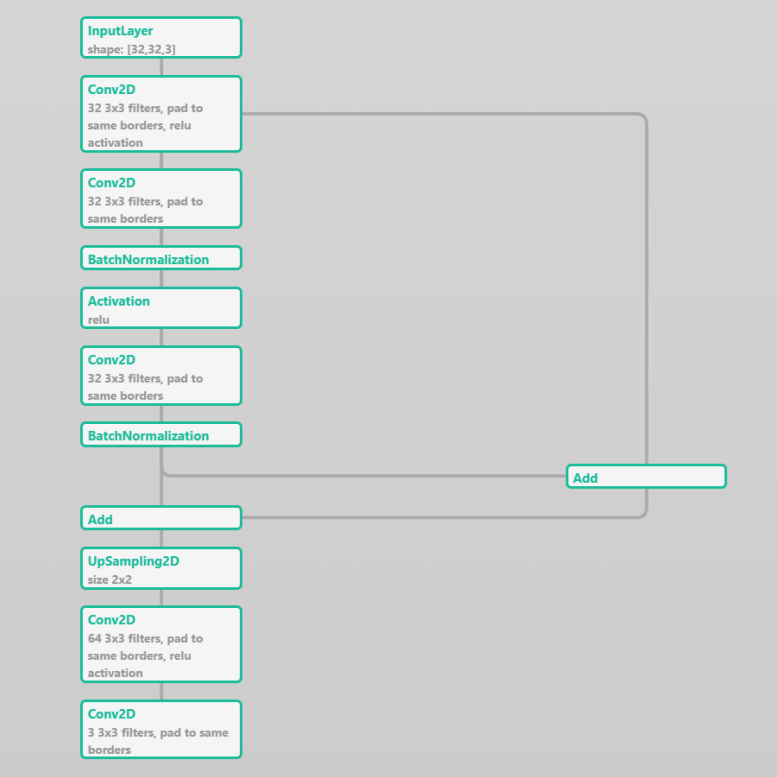
- *Take 15 minutes to explore this application.*
- *Use URL: https://transcranial.github.io/keras-js/#/image-super-resolution*



How it works:

See URL: https://github.com/titu1994/Image-Super-Resolution

Implementation of Image Super Resolution CNN in Keras from the paper *Image Super-Resolution Using Deep Convolutional Networks*.


InputLayer
shape: [32,32,3]
Conv2D
32 3x3 filters, pad to same borders, relu activation
Conv2D
32 3x3 filters, pad to same borders
BatchNormalization
Activation
relu
Conv2D
32 3x3 filters, pad to same borders
BatchNormalization
Add
Add
UpSampling2D
size 2x2
Conv2D
64 3x3 filters, pad to same borders, relu activation
Conv2D
3 3x3 filters, pad to same borders

3.
NLP APPLICATION

In this section we will be using deep learning for a different kind of data involving sequences. Unlike image data where pixels in the neighbourhood of each other are likely to be related, in sequence data the ordering of the data is import. Consider the sentence: "grandmother bit the dog" vs "dog bit the grandmother": the same words are used but they have completely different meanings depending on the sequence of words.

NATURAL LANGUAGE PROCESSING (NLP)

Natural-language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to fruitfully process large amounts of natural language data.

Challenges in natural-language processing frequently involve speech recognition, natural-language understanding, and natural-language generation.

SENTIMENT ANALYSIS

Sentiment analysis (opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and

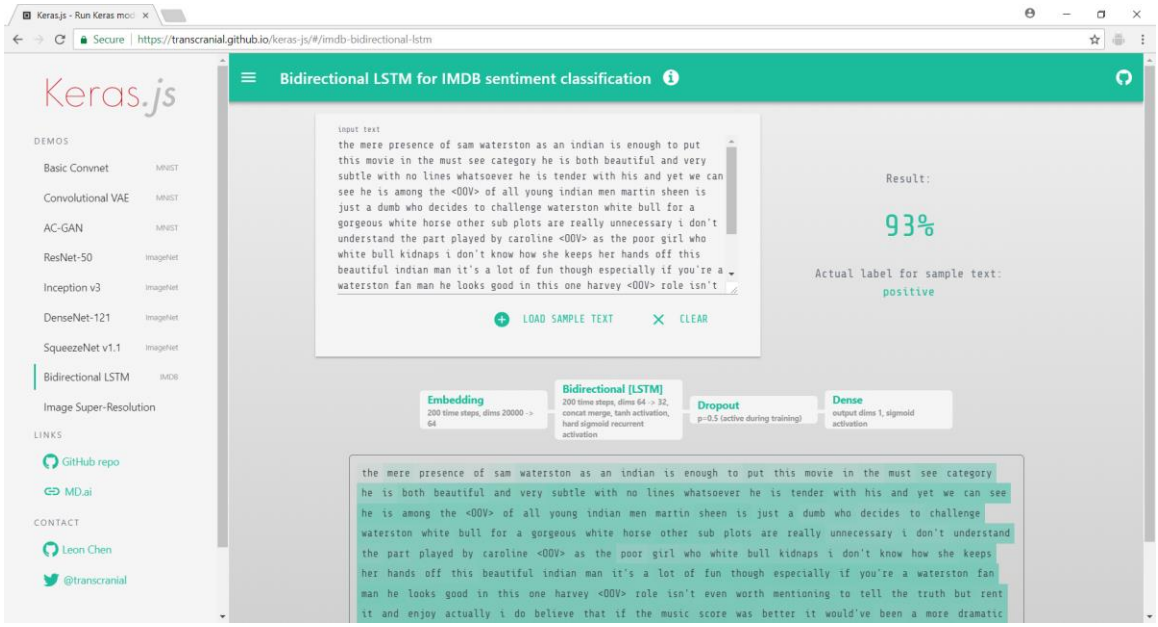
healthcare materials for applications that range from marketing to customer service to clinical medicine.

A basic task in sentiment analysis is classifying the **polarity** of a given text at the document, sentence, or feature/aspect level — whether the expressed opinion in a document, a sentence or an entity feature/aspect is **positive**, **negative**, or **neutral**.

Advanced sentiment classification looks, for instance, at emotional states such as "angry", "sad", and "happy".

Activity

- *Take 15 minutes to explore this application.*
- *Use URL:* *https://transcranial.github.io/keras-js/#/imdb-bidirectional-lstm*



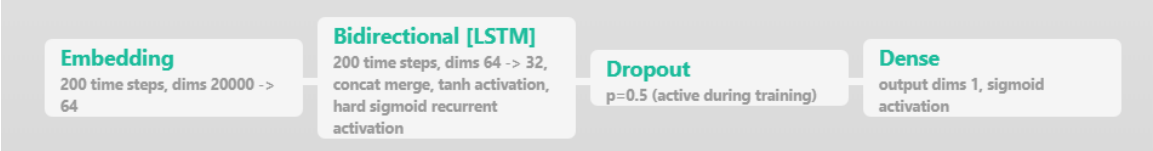
How it works:

Recurrent neural networks are a subclass of neural networks, designed to perform a sequences recognition or prediction. They have a flexible number of inputs and they allow cyclical connections between their neurons. This means that they are able to remember previous information and connect it to the current task.

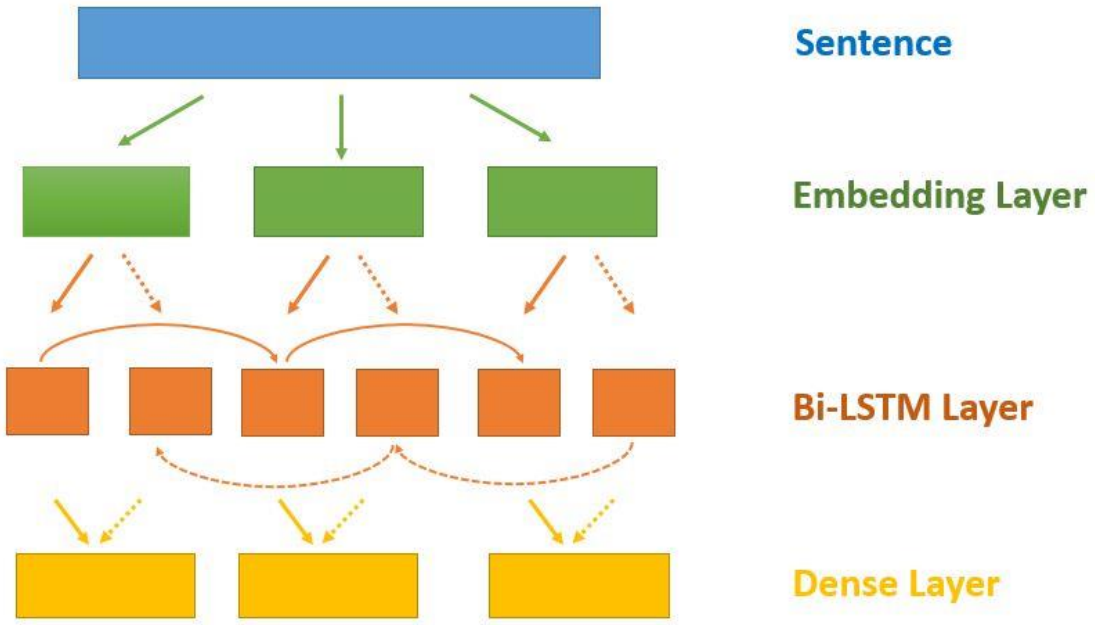
Long Short Term Memory networks (LSTM) are a subclass of RNN, specialized in remembering information for a long period of time. More

over the Bidirectional LSTMs keep the contextual information in both directions.

A bidirectional LSTM is used to implement a deep recurrent neural network. It checks for sequences of words that are used to train the neural network for positive sentiment and negative sentiment.



The Dropout layer is added to prevent overfitting.



As described in the image above, we need to have three layers:

- ***Embedding Layer*** *- modifies the integer representation of words into dense vectors*
- ***Bidirectional LSTM Layer*** *- connects two hidden layers of opposite directions to the same output*
- ***Dense Layer*** *- output layer with softmax activation*

See https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ for explanation of word embedding