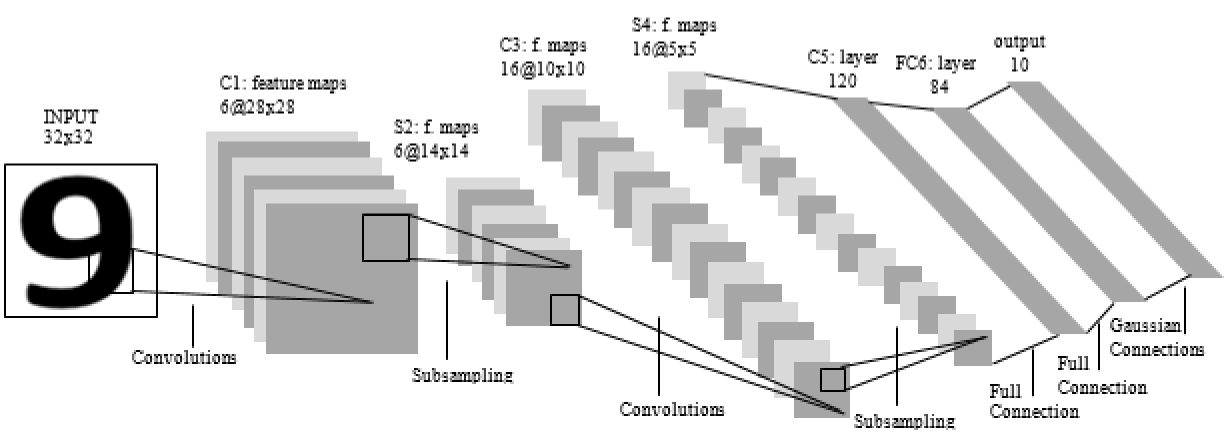
Course Outline

- TOPICS
  1. What is Machine Learning and Image Processing
  2. Traditional Features, K-NN classifier
  3. Linear Classification
  4. Perceptron Algorithm, Sigmoid Activation Function, Gradient Descent
  5. Stochastic Gradient Descent, Back-Propagation
  6. Multi-Layer Neural Network
  7. Convolution and Pooling
  8. Mid-Term Examination
  9. Mid-Term Examination
  10. Convolutional Neural Networks.
  11. Training Convolutional Neural Networks: Hyper-Parameters, Activation functions, initialization, dropout, batch normalization
  12. Recurrent Neural Networks
  13. Applications of Convolutional Neural Networks for Image Segmentation and Object Classification
  14. Project Presentations

Convolution LeNet5

input		32x32	
conv1	5x5 @6	28x28x6	
pool	2x2	14x14x6	
conv2	5x5x6 @16	10x10x16	
pool	2x2	5x5x16	
FC	5x5x16 @120	1x120	
FC	1x120 @84	1x84	
out	1x84 @10	1x10	predicted
		1x10	ground Truth



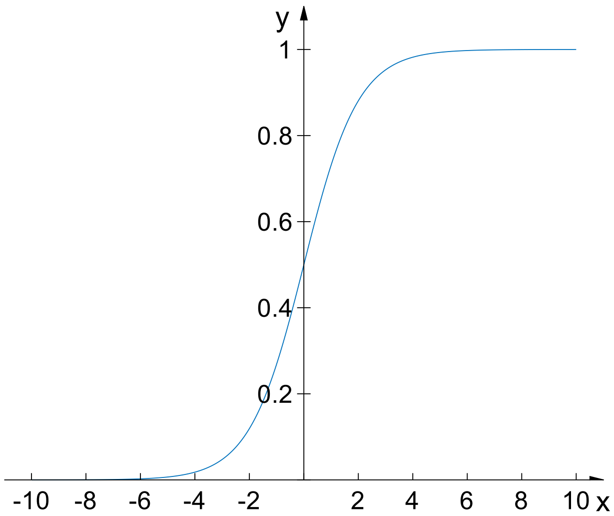
Convolution LeNet5

input		32x32	
conv1	5x5 @6	28x28x6	
reLU		28x28x6	
pool	2x2	14x14x6	
conv2	5x5x6 @16	10x10x16	
reLU		10x10x16	
pool	2x2	5x5x16	
FC	5x5x16 @120	1x120	
FC	1x120 @84	1x84	
out	1x84 @10	1x10	predicted
		1x10	softmax
		1x10	ground Truth

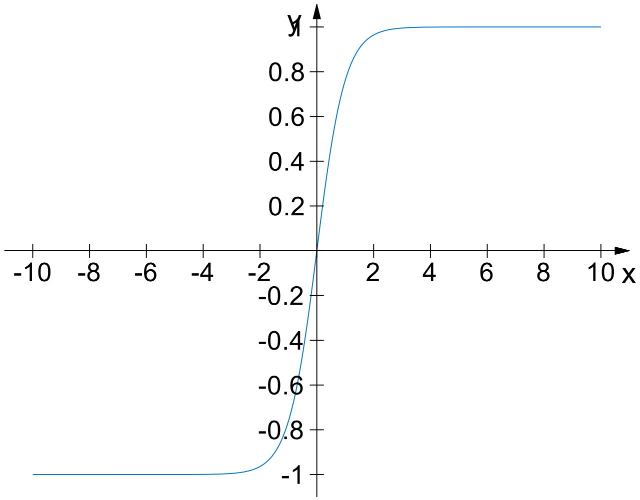
Hyper Parameters

- Training samples: assume that we have n training samples for m classes.
- Learning Rate: used in stochastic gradient descent
  - A.k.a. 0.001. 0.0001. or 0.000001
- Epoch: 1 epoch equals to number of all training samples passed through both of forward and backward process.
- Batch Size:
  - if mini batch size is 1, it means that 1 sample sent to the feed-forward process. Iteration becomes n.
  - if mini batch size is 5, it means that 5 samples sent to the feed-forward process. Iteration becomes n/5.
- Activation Functions: ReLU, Sigmoid, tanh.
- Dropout: used to avoid overfitting thus increasing generalization.
  - If the drop out rate 50%, then 50% nodes set with 0.
- Number of Hidden Layer and Units: number of convolutions, pooling, momentum rate.
- Loss functions: L1, L2, Huber loss etc...
- Weight Initialization: set with random small numbers or Xavier initialization.

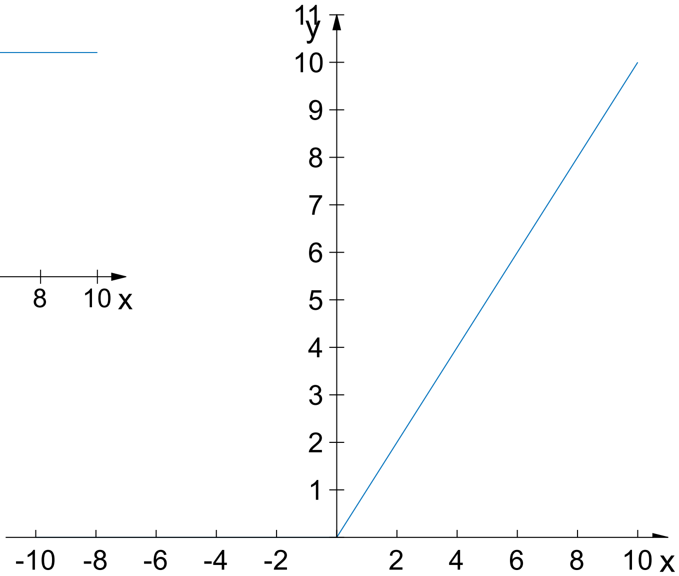
Activation Functions



sigmoid



tanh



ReLU

Initialization of Weights

$\text{input} : 224 \times 224 \times 5$

$f : 3 \times 3 \times 5 \times 64$

$h : 3, w : 3, in : 5, out : 64$

Gaussian

```
type = 'single'
sc = 0.01;
weights = randn(h, w, in, out, type)*sc;
```

Xavier

```
type = 'single'
sc = sqrt(3/(h*w*in)) ;
weights = (rand(h, w, in, out, type)*2 - 1)*sc ;
```

Xavier Improved

```
type = 'single'
sc = sqrt(2/(h*w*out)) ;
weights = randn(h, w, in, out, type)*sc ;
```

Dropout

1	1	3	4

50% dropout

1	0	3	0

1	1	3	4

25% dropout

1	1	3	0

Batch Normalization

- Affects the learning process with SGDM.
- Improve stability.
- Avoid the over-fitting and under-fitting.

$$\hat{x}_i = \frac{x_i}{\sqrt{\sigma_b^2 + eps}}$$

$$\sigma_b = \text{variance of batch}$$

$$eps : 1 \times 10^{-9}$$

data is normalized

Layers Matlab

```
lr = [.1 2] ;

% Define network CIFAR10-quick
net.layers = {} ;

% Block 1
net.layers{end+1} = struct('type', 'conv', ...
                    'weights', {{0.01*randn(5,5,3,6, 'single'), zeros(1, 32, 'single')}}, ...
                    'learningRate', lr, ...
                    'stride', 1, ...
                    'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                    'method', 'max', ...
                    'pool', [2 2], ...
                    'stride', 2, ...
                    'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
```

http://www.vlfeat.org/matconvnet/

Layers Python Tensorflow

```
# TODO: Layer 1: Convolutional. Input = 32x32x1. Output = 28x28x6.
conv1_w = tf.Variable(tf.truncated_normal(shape = [5,5,1,6],mean = mu, stddev = sigma))
conv1_b = tf.Variable(tf.zeros(6))
conv1 = tf.nn.conv2d(x,conv1_w, strides = [1,1,1,1], padding = 'VALID') + conv1_b
# TODO: Activation.
conv1 = tf.nn.relu(conv1)

# TODO: Pooling. Input = 28x28x6. Output = 14x14x6.
pool_1 = tf.nn.max_pool(conv1,ksize = [1,2,2,1], strides = [1,2,2,1], padding = 'VALID')

# TODO: Layer 2: Convolutional. Output = 10x10x16.
conv2_w = tf.Variable(tf.truncated_normal(shape = [5,5,6,16], mean = mu, stddev = sigma))
conv2_b = tf.Variable(tf.zeros(16))
conv2 = tf.nn.conv2d(pool_1, conv2_w, strides = [1,1,1,1], padding = 'VALID') + conv2_b
# TODO: Activation.
conv2 = tf.nn.relu(conv2)

# TODO: Pooling. Input = 10x10x16. Output = 5x5x16.
pool_2 = tf.nn.max_pool(conv2, ksize = [1,2,2,1], strides = [1,2,2,1], padding = 'VALID')
```

https://github.com/sujaybabruwad/LeNet-in-Tensorflow/blob/master/LeNet-Lab.ipynb

Layers Python Keras

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
def getModel():
    model = Sequential()
    #Layer 1
    #Conv Layer 1
    model.add(Conv2D(filters = 6,
                     kernel_size = 5,
                     strides = 1,
                     activation = 'relu',
                     input_shape = (32,32,1)))
    #Pooling layer 1
    model.add(MaxPooling2D(pool_size = 2, strides = 2))
    #Layer 2
    #Conv Layer 2
    model.add(Conv2D(filters = 16,
                     kernel_size = 5,
                     strides = 1,
                     activation = 'relu',
                     input_shape = (14,14,6)))
    #Pooling Layer 2
    model.add(MaxPooling2D(pool_size = 2, strides = 2))
    #Flatten
    model.add(Flatten())
    #Layer 3
    #Fully connected layer 1
    model.add(Dense(units = 120, activation = 'relu'))
    #Layer 4
    #Fully connected layer 2
    model.add(Dense(units = 84, activation = 'relu'))
    #Layer 5
    #Output Layer
    model.add(Dense(units = 10, activation = 'softmax'))
    model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
    return model
```

check the LENET5KERAS.sln

About Final Project

- In your project, you must use the Keras model with respect to facilities behind the Keras.

- The another advantage of using Keras is expressed with that you can migrate any Keras model to Tensorflow, Matlab and other formats.

- Your project topic would be about classification.