Review on ImageNet Classification with Deep Convolutional Neural Networks by Alex Krizhevsky et. al

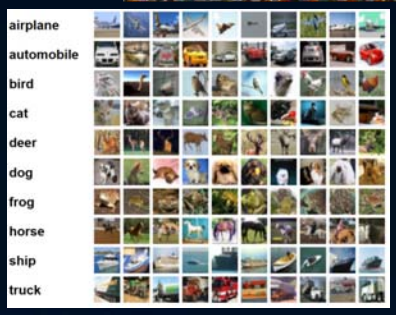
Zhaohui Liang, PhD Candidate, Lassonde School of Engineering
Course #: EECS-6412, FALL 2017
Course Name: Data Mining
Presenting Date: Nov 15, 2017

List of Content

- Background knowledge regarding computer vision
- Deep learning and convolutional neural networks (CNN)
- Roles of different layers in CNN architecture
- Pros and Cons of AlexNet
- Current tools for Deep learning and CNN
- Question / Answer

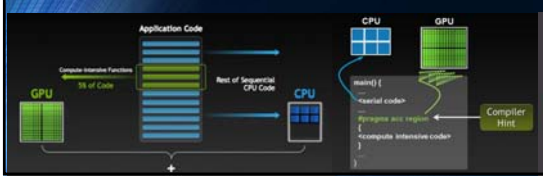
Background knowledge regarding computer vision

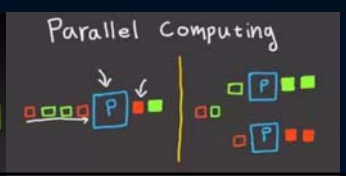
- The ImageNet (2010/2012) dataset: 15 million 227*227 annotated color images, 22,000-class, gold standard for the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) since 2010
- CIFAR-10/100 maintained by The Canadian Institute for Advanced Research, UofT
  - CIFAR-10: 60K 32*32 color images, 10 classes – 1,000 per class
  - CIFAAR-100: 60K 32*32 color images, 100 classes – 100 per class
  - MNIST: 70K 28*28 grey-level handwritten digits for 10 classes (0 to 9)

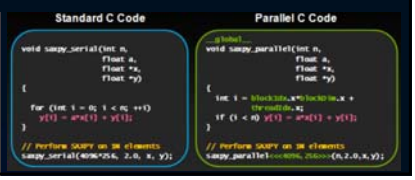


GPU and Parallel Computing

- GPU or Graphics Processing Unit, it works with CPU to accelerate deep learning, analytics, and engineering applications
- GPU offloads compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU
- Parallel computing is the simultaneous use of multiple compute resources to solve a computational problem: A problem is broken into discrete parts that can be solved concurrently.
- CUDA: a parallel computing platform and programming model for NVIDIA GPUs
- cuDNN: CUDA deep neural network library for deep learning running on NVIDIA GPUs





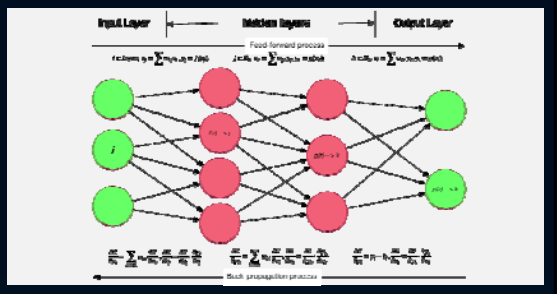


Outline of the AlexNet for ImageNet classification

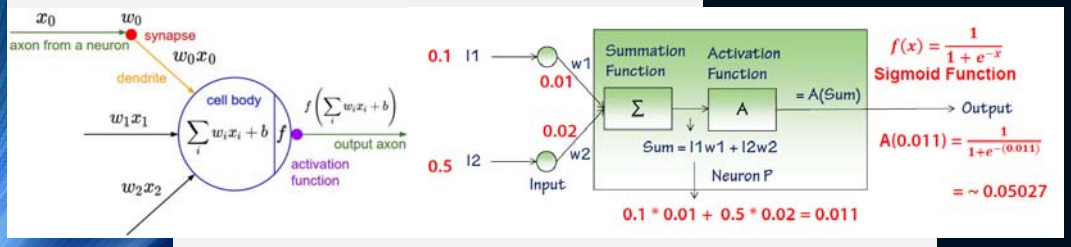
- AlexNet is considered as a breakthrough of deep convolutional neural network to classify the ILSVRC-2010 test set, with the top-5 error 17%, achieving with a CNN with 5 conv layers and 3 dense (fully connected) layers
- Use of multiple GPUs and parallel computing is highlighted in the training of AlexNet.
- The use of ReLU (Rectified Linear Units) as the activation function for image classification by CNN
- Introduction of local normalization to improve learning. It is also called "brightness normalization".
- Use of overlapping pooling. It is considered as a way to reduce overfitting
- Apply two methods: image translation and reflection, and cross color channel PCA to overcome over-fitting
- Apply a 0.5 dropout on the first 2 dense layers to suppress over-fitting

Deep Neural Networks

- A deep neural network is a neural network model with two hidden layers or more
- A deep neural network is a model to perform deep learning for pattern recognition, detection, and segmentation etc.
- It provides the state-of-the-art solution for unstructured data such as text recognition, images, videos, voice / sound, natural language processing

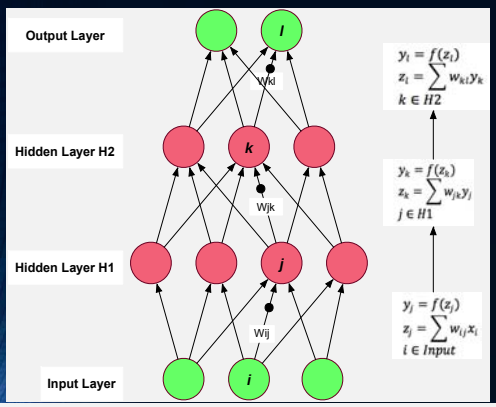


A deep neural network with two hidden layers

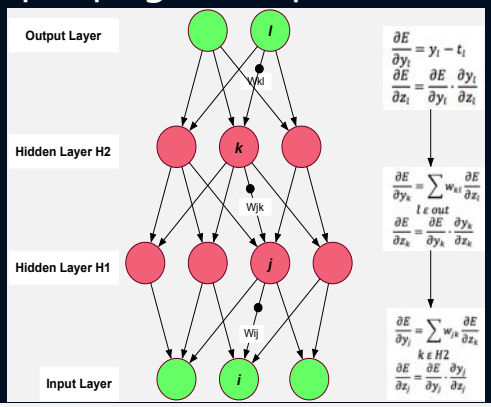


The computing in a single neuron

The feed-forward-back-propagation process



- The output of one layer can be computed by output of all units in the layer
- z is the total input of a unit
- A non-linear f( ) is applied to z to get the output of the unit
- Rectified linear unit (ReLU), tanh, logistic function etc.



- Given the loss function for unit l is $0.5(y_l - t_l)^2$ where t_l is the output, the error derivative is $y_l - t_l$
- The error derivative of output can be converted to the error derivative of the total by multiplying it by the gradient of f(z)

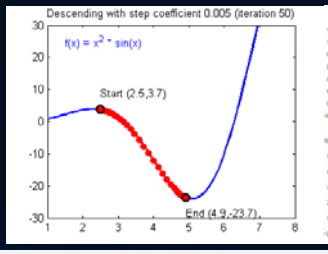
Optimizing a deep neural network

OVER-FITTING VS. UNDER-FITTING

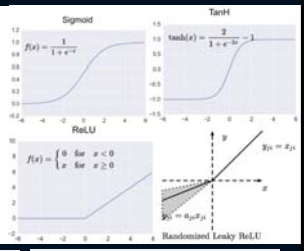
- The goal of training a machine learning: to approximate a representation function to map the input variables (x's) to an output variable (Y).

OVERCOME OVER-FITTING IN DEEP NEURAL NETWORK

- choose the best learning rate: start from 0.1% in AlexNet
- stochastic gradient descent (SGD) - AlexNet
- Alternating activation function – ReLU / Sigmoid



SGD – find minima by derivatives



From Sigmoid to ReLU

Stochastic gradient descent (SGD)

BATCH GRADIENT DESCENT

- $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- $Repeat\{$

m:=total data points

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$\frac{\partial}{\partial\theta_j}J_{train}(\theta)$$

(for j = 0, 1, 2, …, n -- # of neurons)

$\}$

STOCHASTIC GRADIENT DESCENT

- $cost\left(\theta, (x^{(i)}, y^{(i)})\right) = \frac{1}{2}\left(h_\theta(x^{(i)}, y^{(i)})\right)^2$
- $J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$

1. Randomly shuffle the data points in training set
2. $Repeat\{\ from\ 1 \sim a\ small\ subset\ of\ training\ data\ points$ with m rows

$i := 1, \ldots, m\{$

$$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

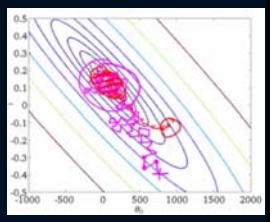
(for j = 0, …, n)

$\}$

$$\frac{\partial}{\partial\theta_j}cost\left(\theta, (x^{(i)}, y^{(i)})\right)$$

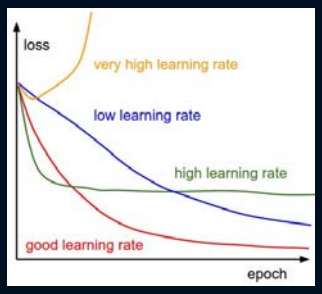
$\}$

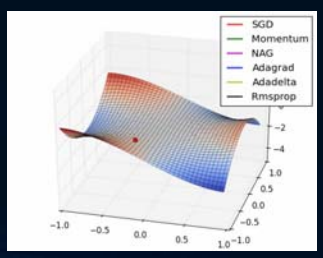
- SGD will not always get the true minima
- But reach the narrow neighbourhood

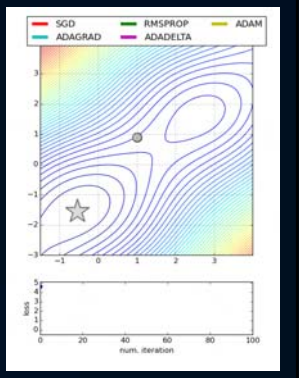


Algorithms to optimize SGD

- Difficult to choose the best learning rate for SGD -- convergence
- Walk out from a saddle point
- Revised SGD algorithms
  1. Momentum / Nesterov momentum - AlexNet
  2. Adaptive gradient algorithm (AdaGrad) / AdaDelta
  3. Root Mean Square Propagation (RMSProp) (Hinton et al. 2012)
  4. Adaptive Moment Estimation (Adam) (Hinton et al. 2014)

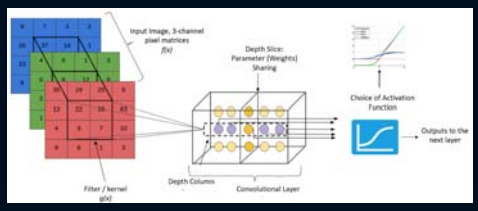


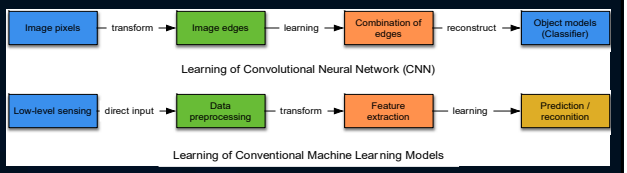




Convolutional Neural Network for image classification

- Convolution neural network (CNN) is a deep learning model particularly designed for learning of two-dimensional data such as images and videos.
- A CNN can be fed with raw input and automatically discover high-dimensional complex representations



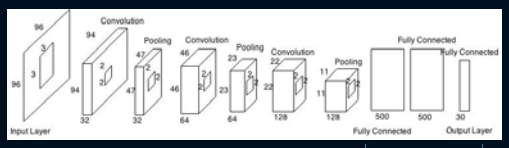


LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015; 521(7553): 436-444

The unique feature of the CNN

- Convolutional layer
- Activation layers
- Pooling layer
- Fully connected layer
- Output layer

Construct sampling unit by the convolutional filters



An input image is a 3-dimensional matrix

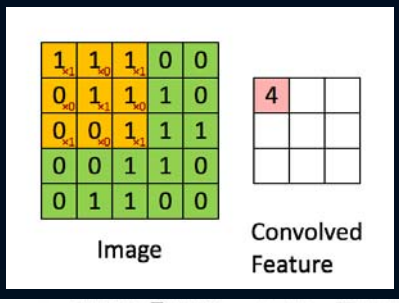
Use the convolutional layer + pooling layer structure to transfer information to a narrow-deep-shape tensor

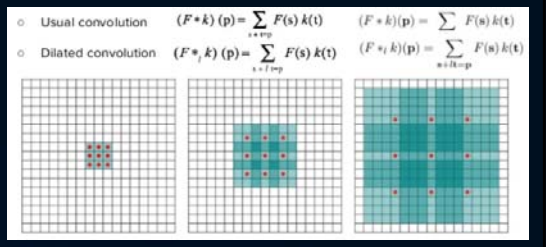
Reshape the tensor to two dimensions for regular NN learning

Convolutional layer

- Convolution operations
  - The convolution of two vectors, u and v, represents the area of overlap under the points as v (filter) slides across u
  - In CNN, the convolutional layer applies a series of filters to scan the raw pixels or the mapped information from the former layer

$$(f * g)(t) = \int_0^t f(t-\tau)g(\tau)d\tau$$





Dilated convolution can aggregate multiscale contextual information without loss of resolution

The convolution operation in detail

The convolution of two functions f and g at the point t, is a function (f*g) with respect to t, i.e.

$$(f * g)(t) = \int_0^t f(t-\tau)g(\tau)d\tau$$

Let $f(t) = sint$, and let $g(t) = cost$, then $(f * g)(t) = \int_0^t \sin(t-\tau)\, cos\tau\, d\tau$

$$(f * g)(t) = \int_0^t \sin(t-\tau)\, cos\tau\, d\tau$$

$$= \int_0^t (\sin t\, cos\tau - sin\tau\, cost)\, cos\tau\, d\tau$$

$$= \int_0^t (sint\, cos^2\tau - cost\, sin\tau\, cos\tau) d\tau$$

$$= sint \int_0^t cos^2\tau\, d\tau - cost \int_0^t sin\tau\, cos\tau\, d\tau$$

$$(\, cos^2\tau = \frac{1}{2}(1 + cos2\tau), u = sin\tau, du = cos\tau\, d\tau)$$

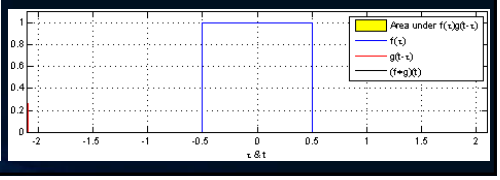
$$= \frac{1}{2}\, sint \int_0^t (1 + cos2\tau) d\tau - cost \int_{\tau=0}^{\tau=t} u\, du$$

$$= \frac{1}{2} sint \left(\tau + \frac{1}{2} sin2\tau\right)|_0^t - cost\, (\frac{1}{2} sin^2\tau)|_0^t$$

$$= \frac{1}{2} t\, sint + \frac{1}{4} sint\, sin2t - \frac{1}{2}\, sin^2 t\, cost$$

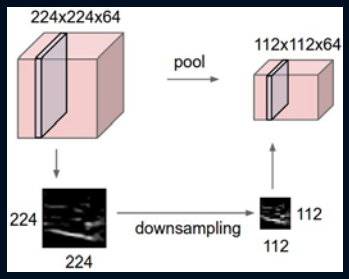
$$= \frac{1}{2}\, t\, sint$$

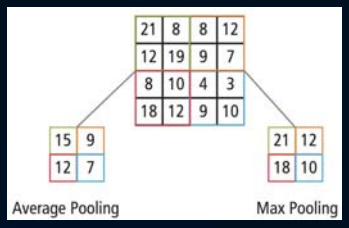
Learning simple features from shallow layers and reassembling to complex features in deep layers



Pooling Layer

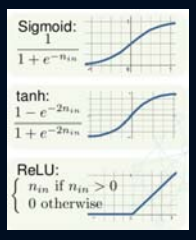
- perform a sub-sampling to reduce the size of the feature map
- merge the local semantically similar features into a more concise representation
- Max pooling – major method
- Average pooling
- The effect of overlapping pooling in AlexNet is not significant

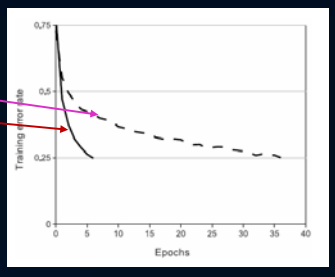




Activation layers

- Activation layers are applied between conv layers to generate learning
- Non-linear functions is the common activation functions in CNN
  - Tanh
  - Sigmoid
  - ReLU (rectified linear unit)
    - can greatly accelerate the convergence of stochastic gradient
    - Low computing cost
    - can easily suppress neurons by replacing any negative input by zero, the died neuron cannot be reactivated

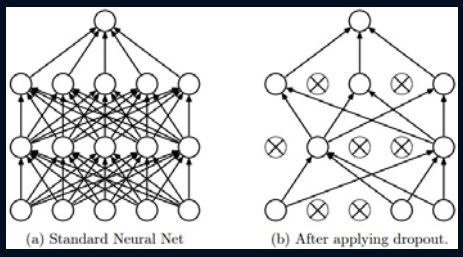


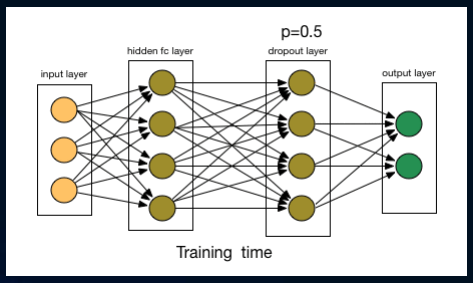


Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 2012 (pp. 1097-1105).

Dropout layer

- Dropout is an effective method to suppress overfitting
- Dropout layer randomly deletes some neurons from the dense layers.
- It can reduce complex co-adaptations of neurons and force the neural network to learn more robust features





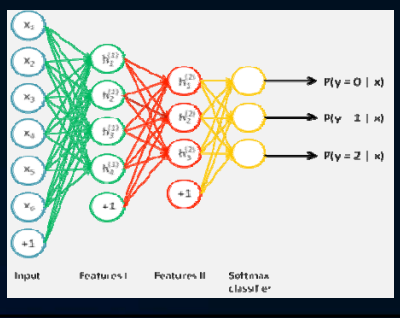
Output layers

- The fully connected layers contain neurons that connect to the entire input volume as other neural networks
- A typical setting for output layers consists of a series of fully connected layers and ends with a Softmax function for outputs

$$P(C_r \mid x) = \frac{P(x \mid C_r)P(C_r)}{\sum_{j=1}^{k} P(x \mid C_j)P(C_j)} = \frac{exp(a_r)}{\sum_{j=1}^{k} exp(a_j)}$$

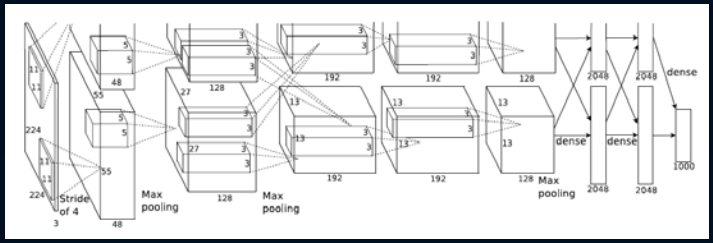
$$(0 \leq P(C_r \mid x) \leq 1 \quad and \quad \sum_{j=1}^{k} P(c_j \mid x) = 1)$$

- The Softmax layer returns the probability regarding the conditional probability of the given class
- also known as the normalized exponential
- can be considered as the multi-class generalization of the logistic sigmoid function



The overall architecture of AlexNet

- The AlexNet has
  - five conv layers
  - Max pooling is applied between every two conv layers
  - After the tensors are flattened, two fully-connected (dense) layers are used
  - The output layer is a softmax layer to compute the softmax loss function for learning



The computing uses two NVIDIA GTX 580 GPUs

AlexNet in Java Code with DL4J

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(seed)
    .weightInit(WeightInit.DISTRIBUTION)
    .dist(new NormalDistribution( mean: 0.0,  std: 0.01))
    .activation(Activation.RELU)
    .updater(Updater.NESTEROVS)
    .iterations(iterations)
    .gradientNormalization(GradientNormalization.RenormalizeL2PerLayer) // normalize to prevent vanishing or exploding gradients
    .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
    .learningRate(1e-2)
    .biasLearningRate(1e-2*2)
    .learningRateDecayPolicy(LearningRatePolicy.Step)
    .lrPolicyDecayRate(0.1)
    .lrPolicySteps(100000)
    .regularization(true)
    .l2(5 * 1e-4)
    .momentum(0.9)
    .miniBatch(false)
```

Use the Nesterovs algorithm

Use learning decay rate of 0.1

Use L2 regularization

AlexNet Architecture in Java

```
.list()
.layer( ind: 0, convInit( name: "cnn1", channels, out: 96, new int[]{11, 11}, new int[]{4, 4}, new int[]{3, 3}, bias: 0))
.layer( ind: 1, new LocalResponseNormalization.Builder().name("lrn1").build())
.layer( ind: 2, maxPool( name: "maxpool1", new int[]{3,3}))
.layer( ind: 3, conv5x5( name: "cnn2", out: 256, new int[] {1,1}, new int[] {2,2}, nonZeroBias))
.layer( ind: 4, new LocalResponseNormalization.Builder().name("lrn2").build())
.layer( ind: 5, maxPool( name: "maxpool2", new int[]{3,3}))
.layer( ind: 6,conv3x3( name: "cnn3", out: 384, bias: 0))
.layer( ind: 7,conv3x3( name: "cnn4", out: 384, nonZeroBias))
.layer( ind: 8,conv3x3( name: "cnn5", out: 256, nonZeroBias))
.layer( ind: 9, maxPool( name: "maxpool3", new int[]{3,3}))
.layer( ind: 10, fullyConnected( name: "ffn1", out: 4096, nonZeroBias, dropOut, new GaussianDistribution( mean: 0, std: 0.005)))
.layer( ind: 11, fullyConnected( name: "ffn2", out: 4096, nonZeroBias, dropOut, new GaussianDistribution( mean: 0, std: 0.005)))
.layer( ind: 12, new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
    .name("output")
    .nOut(numLabels)
    .activation(Activation.SOFTMAX)
    .build())
.backprop(true)
.pretrain(false)
.setInputType(InputType.convolutional(height, width, channels))
.build();
```

Conv layers

Color channel at the third dimension

Pros and cons of AlexNet

STRENGTH

- AlexNet is considered as the milestone of CNN for image classification
- Many methods such as the conv+pooling design, dropout, GPU, parallel computing, ReLU is still the industrial standard for computer vision
- The unique advantage of AlexNet is the directly image input to the classification model.
- The convolution layers can automatically extract the edges of the images, and fully connected layers learning these features
- Theoretically the complexity of visual patterns can be effective extracted by adding more conv layers

WEAKNESS

- AlexNet is NOT deep enough compared to the later model such as VGG Net, GoogLENet, and ResNet
- The use of large convolution filters (5*5) is not encouraged shortly after that
- Use normal distribution to initiate the weights in the neural networks cannot effective solve the problem of gradient vanishing, replaced by the Xavier method later
- The performance is surpassed by more complex models such as GoogLENet (6.7%), and ResNet (3.6%)

Tools for deep convolutional neural networks

- Python
  - TensorFlow by Google
  - Keras coordinated by Google
  - Theano by University of Montreal
  - CNTK by Microsoft
- C++
  - Caffe by UC Berkeley
  - Support Python, MATLAB, and CUDA
- Java
  - Deeplearning4J: includes three core libraries: deeplearning4j, nd4j, DataVec
- Others
  - MATLAB: Neural Network Toolbox, MatConvNet
  - Torch: Lua
  - Encog: C#
  - ConvNetJS: Java Script

Q & A Time

Thank you