Neural Network Theory and Applications：Lecture 5
神经网络理论及应用：第五讲

Bao-Liang Lu（吕宝粮）
Dept. of Computer Science & Engineering
Shanghai Jiao Tong University
Office: 3-431 Dianxin Bld, Minhang
Tel: 3420-5422
bllu@sjtu.edu.cn

Office Hour

- Time: PM 3:30-4:30, Tuesday
- Venue：3-414, Dianxin Building (电信学院3号楼414教室）
- Teaching Asistants:

刘伟

罗赟

赵黎明

Outline of Lecture Five

- Deep Learning
- Deep Networks
  - Deep Belief Networks (DBNs)
  - Deep Autoencoder
  - Convolutional Neural Networks

What Is Deep Learning?

- More neurons than traditional feed-forward multilayer networks (Shallow)
- More complex ways of connecting layers
- "Cambrian" explosion of computing power to train
- Automatic feature extraction

Defining Deep Networks

- Unsupervised Pretrained networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Recursive Neural Networks

Why Deep Learning

- Biological Plausibility – e.g. Visual Cortex
- <u>Learning features, not just handcrafting them</u>
- Unsupervised feature learning
- Learning multiple levels of representation
- Highly varying functions can be efficiently represented with deep architectures
  - Less weights/parameters to update than a less efficient shallow representation
- Sub-features created in deep architecture can potentially be shared between multiple tasks
  - Type of Transfer/Multi-task learning

Deep Learning = Learning Features

- Traditional model of pattern recognition
  - Fixed features + trainable classifier

- Deep Learning
  - Trainable features + trainable classifier

Computer Vision Features

SIFT

Spin image

HoG

RIFT

Textons

GLOH

人脸特征（ MLGBP）

Original Image
Normalized Image
GIED
GIPP
MLGBP Feature
Feature Vector
Face Location
Multi-resolution Operation
Gabor Filters
LBP Operation
Regional Histogram
Histogram projection using CCL or LDA

多特征性别分类框架

Deep Learning = Learning Hierarchical Representations

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Learning Feature Hierarchy

Deep Learning

- Deep architectures can be representationally efficient.
- Natural progression from low level to high level structures.
- Can share the lower-level representations for multiple tasks

Trainable Feature Hierarchy

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story
- Speech
  - Sample → spectral band → sound → … → phone → phoneme →word

The Mammalian Visual cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina – LGN – V1 – V2 – V4 – PIT – AIT ....
- Lots of intermediate representations

Trainable Feature Hierarchies

- A hierarchy of trainable feature
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories

MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

Additive Manufacturing

Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.

Baxter: The Blue-Collar Robot

Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.

Memory Implants

A maverick neuroscientist believes he has deciphered the code by which the brain forms long-term memories. Next: testing a prosthetic implant for people suffering from long-term memory loss.

Smart Watches

The designers of the Pebble watch realized that a mobile phone is more useful if you don't have to take it out of your pocket.

Ultra-Efficient Solar Power

Doubling the efficiency of a solar cell would completely change the economics of renewable energy. Nanotechnology just might make it possible.

Big Data from Cheap Phones

Collecting and analyzing information from simple cell phones can provide surprising insights into how people move about and behave – and even help us understand the spread of diseases.

Supergrids

A new high-power circuit breaker could finally make highly efficient DC power grids practical.

Major Breakthrough in 2006

- Biological Ability to train deep architectures by using layer-wise unsupervised learning, whereas previous purely supervised attempts had failed

- Unsupervised feature learners:
  - RBMs
  - Auto-encoder variants
  - Sparse coding variants

2019 ACM Turing Award

The New York Times

Scientists See Promise in Deep-Learning Programs

John Markoff
November 23, 2012

Speech Recognition Progress: NIST evaluations

After no improvement for 10+ years by the research community…

…MSR used deep learning to reduce error rate from **~23%** to **~13%** on SWBD (and under 7% for Rick Rashid's demo)!

Deep Learning Today

- AlphaGo-Zero
- Speech Recognition and machine translation
  - A few long-standing performance records were broken with deep learning methods
  - Microsoft and Google have both deployed DL-based speech recognition system in their products
  - Microsoft, Google, IBM, Nuance, AT&T, and all the major academic and industrial players in speech recognition have projects on deep learning
- Computer Vision
  - Feature engineering is the bread-and-butter of a large portion of the CV community, which creates some resistance to feature learning
  - But the record holders on ImageNet and Semantic Segmentation are convolutional nets

Difficulties of Training Deep Networks

- Difficulties of supervised training of deep networks
- Early layers of MLP do not get trained well
  - Diffusion of Gradient – error attenuates as it propagates to earlier layers
  - Leads to very slow training
  - Need a way for early layers to do effective work
  - Often not enough labeled data available while there may be lots of unlabeled data
  - Can we use unsupervised/semi-supervised approaches to take advantage of the unlabeled data?
- Deep networks tend to have more local minima problems than shallow networks during supervised training

Three Types of Training Protocols

- Purely Supervised
  - Initialize parameters randomly
  - Train in supervised mode
- Unsupervised, layer-wise + supervised classifier on top
  - Train each layer unsupervised, one after the other
  - Train a supervised classifier on top, keeping the other layers fixed
  - Good when very few labeled samples are available
- Unsupervised, layer-wise + global supervised fine-tuning
  - Train each layer unsupervised, one after the other
  - Add a classifier layer, and retrain the whole thing supervised
  - Good when label set is poor

Multilayer feed-forward neural network

Unsupervised, layer-wise + Supervised

Greedy Layer-Wise Training

- Train first layer using unlabeled data
- Then freeze the first layer parameters and start training the second layer using the output of the first layer as the unsupervised input to the second layer
- Repeat this for as many layers as desired
- This builds our set of robust features
- Use the outputs of the final layer as inputs to a supervised layer/model and train the last supervised layer(s) (leave early weights frozen)
- Unfreeze all weights and fine tune the full network by training with a supervised approach, given the pre-processed weight settings

Greedy Layer-Wise Training-1

- Greedy layer-wise training avoids many of the problems of trying to train a deep net in a supervised fashion
  - Each layer gets full learning focus in its turn since it is the only current "top" layer
  - Can take advantage of the unlabeled data
  - Once you finally start the supervised training portion the network weights have been adjusted so that you are already in a good error basin and now just need to fine tune. This helps with problems of
    - ✓ Ineffective early layer learning
    - ✓ Deep network local minima

Multilayer feed-forward neural network

Three Types of Deep Architectures

- Feed-Forward: multilayer neural nets, convolutional nets

- Feed-Back: Stacked Sparse Coding, Deconvolutional Nets

- Bi-Drectional: Deep Boltzmann Machines, Stacked Auto-Encoders

Core Components of Deep Networks

- Parameters (weights)
- Layers
- Activation functions (sigmoid)
- Loss functions (squared loss)
- Optimization methods (first-order)
- Hyperparameters

Hyperparameters

- Layer size
- Magnitude (momentum, learning rate)
- Regularization (dropout, drop connect, L1)
- Activation (and activation function families)
- Weight initialization strategy
- Loss functions
- Settings for epochs during training (mini-batch size)
- Normalization scheme for input data

Mini-batching

- With mini-batching, we send more than one input vector (a group or batch of vectors) to be trained in the learning system
- This allows us to use hardware and resources more efficiently at the computer-architecture level
- This method also allows us to compute certain linear algebra operations in a vectorized fashion
- In this scenario we also have the option of sending vectorized computations to GPUs if they are present

Building Blocks of Deep Networks

- Feed-forward multilayer neural networks
- Restricted Boltzmann Machines (RBMs)
- Auto-encorders

Outline of Lecture Five

- Deep Learning
- Deep Networks
  - **Deep Belief Networks (DBNs)**
  - Deep Autoencoder
  - Convolutional Neural Networks

Restricted Boltzmann Machines (RBMs)

- The Boltzmann distribution is named after Ludwig Boltzmann who first formulated it in 1868 during his studies of the statistical mechanics of gases in thermal equilibrium.
- Boltzmann Machines were invented in 1985 by Geoffrey Hinton and Terry Sejnowski
- A Boltzmann machine is a type of stochastic recurrent neural network
- Feature extraction
- Dimensionality reduction

Boltzmann Machines

A graphical representation of an example Boltzmann machine. Each undirected edge represents dependency. In this example there are 3 hidden units and 4 visible units. This is not a restricted Boltzmann machine.

Restricted Boltzmann Machines (RBMs)

- Graphical representation of a restricted Boltzmann machine.
- The four blue units represent hidden units, and the three red units represent visible states.
- In restricted Boltzmann machines there are only connections between hidden and visible units, and none between units of the same type .

Visible and hidden layers

- Every single node of the visible (input) layer is connected by weights to every single node of the hidden layer
- Hidden units are feature detectors, learning feature from the input data
- Units (nodes) in the visible layer are "observable" in that they take training vectors as input
- Each node performs computation based on the input to the node and outputs a result based on a stochastic decision whether or not to transmit data through an activation

Connections and weights

- Nodes make decisions about whether to be on or off through acts of computation
- "On" means that they pass a signal further through the net
- "Off" means that they don't
- Usually, being "on" means the data passing through the node is valuable; it contains information that will help the network make a decision
- Being "off" means the network thinks that particular input is irrelevant noise

Reconstruction

- Deep neural networks with unsupervised pretraining (RBMs, autoencorders) perform feature engineering from unlabeled data through <u>reconstruction</u>
- In pretraining, the weights learned through unsupervised pretrain learning are used for weight initialization in networks such as Deep Belief Networks

Reconstructing MNIST digits with RBMs

Deep Belief Networks (DBNs)

- DBNS are composed of layers Restricted Boltzmann Machines (RBMs) for the pretrained phase and then a feed-forward network for the fine-tune phase
- We use RBMs to extract higher-level features from the raw input vectors
- It was discovered that we could train better neural networks by letting RBMs learn <u>progressively</u> higher-level features using the learned features from a lower level RBMs pretrain layer as the input to a higher-level RBMs pretrain layer

Learning higher-order features automatically

- Learning features in an unsupervised fashion is considered the pretrain phase of DBNs
- Each hidden layer of the RBM in the pretrain phase learns progressively more complex features from the distribution of the data
- These higher-order features are progressively combined in nonlinear ways to do elegant automated feature engineering

RBM: Weights → Energies → Probabilities

- Joint distribution $\mathrm{p}(\mathbf{v},\mathbf{h};\theta)$ is defined in terms of an energy function $\mathrm{E}(\mathbf{v},\mathbf{h};\theta)$

$$\mathrm{p}(\mathbf{v},\mathbf{h};\theta) = \frac{exp\big(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\big)}{Z}$$

- For a Bernoulli–Bernoulli RBM

$$\mathrm{E}(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j$$

- For a Gaussian–Bernoulli RBM

$$\mathrm{E}(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j + \frac{1}{2}\sum_{i=1}^{V}(v_i - b_i)^2 - \sum_{j=1}^{H} a_j h_j$$

- $\mathrm{p}(\mathbf{v},\mathbf{h};\theta)$ → generative model!

Restricted Boltzmann Machines (RBMs)

- Conditional probabilities are very easy to calculate
- For a Bernoulli–Bernoulli RBM

Inference

$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right)$$

synthesis

$$p(v_i = 1|\mathbf{h};\theta) = \sigma\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i\right)$$

- For a Gaussian–Bernoulli RBM

Inference

$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right)$$

synthesis

$$p(v_i|\mathbf{h};\theta) = N\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i, 1\right)$$

- Proof next page. (This is a "neural net" with stochastic units rather than the deterministic MLP that you may be more familiar with)

RBM Conditionals Factorize

$$
\begin{aligned}
P(\mathbf{h}|\mathbf{x}) &= \frac{\exp(\mathbf{b}'\mathbf{x} + \mathbf{c}'\mathbf{h} + \mathbf{h}'W\mathbf{x})}{\sum_{\tilde{\mathbf{h}}} \exp(\mathbf{b}'\mathbf{x} + \mathbf{c}'\tilde{\mathbf{h}} + \tilde{\mathbf{h}}'W\mathbf{x})} \\
&= \frac{\prod_i \exp(\mathbf{c}_i\mathbf{h}_i + \mathbf{h}_i W_i\mathbf{x})}{\prod_i \sum_{\tilde{\mathbf{h}}_i} \exp(\mathbf{c}_i\tilde{\mathbf{h}}_i + \tilde{\mathbf{h}}_i W_i\mathbf{x})} \\
&= \prod_i \frac{\exp(\mathbf{h}_i(\mathbf{c}_i + W_i\mathbf{x}))}{\sum_{\tilde{\mathbf{h}}_i} \exp(\tilde{\mathbf{h}}_i(\mathbf{c}_i + W_i\mathbf{x}))} \\
&= \prod_i P(\mathbf{h}_i|\mathbf{x}).
\end{aligned}
$$

RBM Conditionals Factorize

With $\mathbf{h}_i \in \{0,1\}$, recall $\text{Energy}(\mathbf{x},\mathbf{h}) = -\mathbf{b}'\mathbf{x} - \mathbf{c}'\mathbf{h} - \mathbf{h}'W\mathbf{x}$

$$
\begin{aligned}
P(\mathbf{h}_i = 1|\mathbf{x}) &= \frac{e^{1\mathbf{c}_i + 1W_i\mathbf{x} + other\ terms}}{e^{1\mathbf{c}_i + 1W_i\mathbf{x} + other\ terms} + e^{0\mathbf{c}_i + 0W_i\mathbf{x} + other\ terms}} \\
&= \frac{e^{\mathbf{c}_i + W_i\mathbf{x}}}{e^{\mathbf{c}_i + W_i\mathbf{x}} + 1} \\
&= \frac{1}{1 + e^{-\mathbf{c}_i - W_i\mathbf{x}}} \\
&= \text{sigm}(\mathbf{c}_i + W_i\mathbf{x}).
\end{aligned}
$$

since $\text{sigm}(a) = \frac{1}{1+e^{-a}}$.

Maximum likelihood learning for RBM

Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\frac{\partial \log p(v)}{\partial w_{ij}} = <v_i h_j>^0 - <v_i h_j>^\infty$$

Maximum likelihood learning for RBM

$$\ln P(\mathbf{v}) = \ln\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right) - \ln\left(\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right)$$

$$\begin{aligned}
\frac{\partial \log P(\mathbf{v})}{\partial \boldsymbol{\theta}} &= \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} - \frac{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \\
&= \sum_{\mathbf{h}}\left(P(\mathbf{h}|\mathbf{v})\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)\right) - \sum_{\mathbf{v},\mathbf{h}}\left(P(\mathbf{v},\mathbf{h})\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)\right) \\
&= \mathbb{E}_{P(\mathbf{h}|\mathbf{v})}\left[-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right] - \mathbb{E}_{P(\mathbf{h},\mathbf{v})}\left[-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right]
\end{aligned}$$

Training RBMs

- $\Delta w_{ij} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$
- Approximate $\langle v_i h_j \rangle_{model}$
  i. Initialize $\mathbf{v_0}$ at data
  ii. Sample $\mathbf{h_0} \sim \boldsymbol{p}(\mathbf{h}|\mathbf{v_0})$
  iii. Sample $\mathbf{v_1} \sim \boldsymbol{p}(\mathbf{v}|\mathbf{h_0})$
  iv. Sample $\mathbf{h_1} \sim \boldsymbol{p}(\mathbf{h}|\mathbf{v_1})$
  v. Call $(\mathbf{v_1}, \mathbf{h_1})$ a sample from the model.
- $(\mathbf{v}_\infty, \mathbf{h}_\infty)$ is a true sample from the model. $(\mathbf{v_1}, \mathbf{h_1})$ is a very rough estimate but worked
- Contrastive divergence algorithm (CD)

Deep Belief Network Training

- Same Greedy Layer-wise approach
- First train lowest RBM (h^0 – h^1) using RBM update algorithm
- Freeze weights and train subsequent RBM layers
- Then connect final outputs to a supervised model and train that model
- Finally, unfreeze all weights, and train full DBN with supervised model to fine-tune weights

Fine-tuning for discrimination

- First learn one layer at a time greedily.
- Then treat this as “pre-training” that finds a good initial set of weights which can be fine-tuned by a local search procedure.
- Backpropagation can be used to fine-tune the model for better discrimination.

Fine Tuning DNN after pre-training: Optimization view

- Stacking RBMs one layer at a time scales well to really big networks
- Do not start back-propagation until sensible feature detectors are found by RBM pre-training that should already be very helpful for the discrimination task.
- Back-propagation only needs to perform a local search from a sensible starting point.

Fine Tuning DNN after pre-training: Regularization view

- Information in the pre-trained weights comes from modeling the distribution of input vectors in an "unsupervised" manner.
- The input vectors generally contain a lot more information than the labels.
- The precious information in the labels is only used for the final fine-tuning.
- The fine-tuning only modifies the features slightly to get the category boundaries right. No need to discover "features".
- Hence less prone to overfit (unlike the old neural nets with typically random weight initialization)
- This type of backpropagation works well even if most of the training data is unlabeled.
- The unlabeled data is still very useful for discovering good features.

Deep Belief Network

Generating samples from DBN

Figure 9: Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is initialized by an up-pass from a random binary image in which each pixel is on with a probability of 0.5. The first column shows the results of a down-pass from this initial high-level state. Subsequent columns are produced by 20 iterations of alternating Gibbs sampling in the associative memory.

Hinton et al, A Fast Learning Algorithm for Deep Belief Nets, 2006

Comparison of SVM and DBN

Feature	Classifier	Delta	Theta	Alpha	Beta	Gamma	Total
PSD	SVM	58.03/15.39	57.26/15.09	59.04/15.75	**73.34/15.20**	71.24/16.38	59.60/15.93
	DNN	60.05/16.66	55.03/13.88	52.79/15.38	60.68/21.31	**63.42/19.66**	61.90/16.65
DE	SVM	60.50/14.14	60.95/10.20	66.64/14.41	80.76/11.56	79.56/11.38	**83.99/09.72**
	DNN	64.32/12.45	60.77/10.42	64.01/15.97	78.92/12.48	79.19/14.58	**86.08/08.34**

Wei-Long Zheng, **Bao-Liang Lu**, Investigating Critical Frequency Bands and Channels for EEG-based Emotion Recognition with Deep Neural Networks, IEEE Transactions on Autonomous Mental Development, 2015

Outline of Lecture Five

- Deep Learning
- Deep Networks
  - Deep Belief Networks (DBNs)
  - **Deep Autoencoder**
  - Convolutional Neural Networks

概要

- 什么是自编码器（Auto-Encoder）？
- 自编码器的结构、数学模型、性质
- 深度自编码器。
- 常见的自编码器/深度自编码器模型
- 自编码器/深度自编码器的应用实例

什么是Auto-Encoder？

- 自编码器是一种神经网络，不同于其他的网络，自编码器网络的输出用来重建网络的输入。

- 自编码器网络，包括编码阶段和解码阶段。编码阶段可以按照非监督学习的方式，找到输入数据的某种表示方式；解码阶段将找到的这种表示方式恢复成原始的输入。
- 自编码器可以用来降维、提取特征、预训练深度神经网络。

自编码器的结构、数学模型、性质

- 最简单的自编码器结构是一个只有一个隐层的神经网络。
- 这个网络的输入为X
- 编码阶段用方程 f 表示
- 解码阶段用方程 g 表示
- 网络的输出为 $\hat{X}$
- 数学表示为：$\hat{X} = g\big(f(X)\big)$

网络训练的目的，就是让重构的变量和输入的变量差距尽量小。即最小化损失函数：$L(\hat{X}, X)$

常见的损失函数是最小平方误差函数：$L(\hat{X}, X) = \|\hat{X} - X\|_2^2$

自编码器的结构、数学模型、性质

- 以只有一个隐层的自编码器为例，随机初始化权重W和偏执b，使用编码阶段和解码阶段的激活函数均为sigmoid函数，误差函数为平方误差函数。
- 编码阶段：$\sigma(W_{encode}X + b_{encode})$
- 解码阶段：
  $\hat{X} = \sigma(\mathrm{W}_{decode}\sigma(W_{encode}X + b_{encode}) + b_{decode})$
- 损失函数为

$$L(\hat{X}, X) = \|\hat{X} - X\|_2^2$$

- 利用BP算法可以对网络进行优化求解

自编码器的结构、数学模型、性质

- 自编码器可以用来进行降维，当编码函数和解码函数为线性函数时，可以证明自编码器和**PCA**降维一致。当编码函数和解码函数为非线性函数式，自编码器可以当做非线性降维算法。
- 自编码器隐层可以作为输入数据的特征，供其他机器学习算法使用。
  - 当输入的数据是原始信号，自编码器隐层可以作为新的特征来使用——即可以自动提取所需要的特征。
  - 当输入的数据是人工提取的特征时，可以将自编码器看做是一种特征变换的工具，将原始的特征映射到一个新的特征空间中。

深度自编码器

- 单隐层自编码器网络结构简单，易于实现，但是这种结构不能学到复杂的特征。
- 深度自编码器利用多个隐层，对输入的数据进行多层次的特征学习，能够学习到比较复杂的特征。
- 实现深度自编码器的方式有多种：
  - Stacked Auto-Encoder (SAE)，
  - Deep Auto-Encoder (DAE) – RBM based
  - 卷积自编码器（Convolutianl Auto-Encoder，CAE）
  - 与LSTM网络结合的自编码器结构，常用于序列学习的编码和解码。

深度自编码器 – DAE

- **Deep Auto-Encoder**一般指利用受限玻尔兹曼机(**RBM**)进行预训练，最后进行参数微调的深度自编码器。
- 该结构最早由**Hinton**于**2006**年提出。

深度自编码器 – DAE

- 如图所示，深度自编码器利用**RBM**对每一层进行预训练。
- 输入图像首先转换成一维的向量，作为第一个**RBM**的可见层，其隐层需要人为指定，图中为**2000**个节点。
- 第一个**RBM**训练好之后，其隐含层作为第二个**RBM**的可见层，训练新的**RBM**。图中，第二个**RBM**的隐含层有**1000**个节点。
- 然后以同样的方式继续训练两个新的**RBM**。

深度自编码器 –DAE

- 有了训练好的**RBM**之后，我们将这些**RBM**堆叠起来，作为编码部分，即**Encoder**。
- **Decoder**的构造比较简单，将**Encoder**的结构对称一下，系数使用**RBM**训练好的系数的转置作为初始值即可。
- 由于每一个**RBM**训练都会学到一些信息，并且权重使用相同的权重，因此网络可以实现重建输入的功能。
- 这个形成深度自编码器的过程，叫做**Unrolling** 或者 **Unfolding**。

深度自编码器 – DAE

- 直接使用转置的权重初始化后得到的深度自编码器，虽然能够实现一些重建输入数据的功能，但是效果并不理想。
- 需要使用BP算法对网络的参数进行整体的调整，减小重建图像和原始图像之间的误差。
- 这种参数的微调成为 **Fine-tuning.**

Fine-tuning

深度自编码器降维

- 数据降维
  - A图是PCA对MNIST降维结果
  - B图是Deep Auto-Encoeder的降维结果
  - B图降维效果好于A图

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.

深度自编码器 – SAE

- **Stacked Auto-Encoder**采用逐层贪婪训练算法进行训练。
- 首先对输入层建立单隐层自编码器，隐层作为**Feature 1**
- 然后以第一个自编码器得到的**Feature 1**作为输入，建立新的单隐层自编码器，隐层作为**Feature 2.**

深度自编码器 – SAE

- 然后以第一个自编码器得到的**Feature 1**作为输入，建立新的单隐层自编码器，隐层作为**Feature 2.**
- 对于**Feature 2**的处理按照任务不同有多种，常见的是接一个**softmax**层作为分类器来使用；也可以展开该结构，作为多隐层的自编码器。

深度自编码器 – SAE

- 若最后接softmax层用来分类，那么最终的网络结构：

- 若最后展开形成多层的自编码器，解码阶段的权重**W**取编码阶段预训练得到的解码权重作为初始值即可。

深度自编码器 – SAE

- 无论是展开形成多层的自编码器，还是接**softmax**用来分类，预训练后的网络参数往往表现不太理想，这时，需要进行参数微调 —— **Fine Tuning.**
- 参数微调，是指用**BP**算法对整个网络的参数作调整。
  - 预训练阶段，网络参数是分层训练的，直接整合起来效果不一定理想。
  - 参数微调阶段，是将网络里初始化后的参数作为整体进行调整，使最终参数适应所处理的问题。

去噪自编码器模型

- **Denoising Auto-Encoders**
- 将输入X以概率分布q_D"污染"- X中随机位置的值变为0
- 用污染后的$\hat{X}$得到隐层y，并利用y解码得到X的重建Z
- 优化 X 和 Z 之间的重建误差。
- 这样做可以提高网络的稳定性，同时加入噪声可以迫使网络学到"有用的"表示。

Figure 1. An example $\mathbf{x}$ is corrupted to $\tilde{\mathbf{x}}$. The autoencoder then maps it to $\mathbf{y}$ and attempts to reconstruct $\mathbf{x}$.

Vincent, H. Larochelle Y. Bengio and P.A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08), pages 1096 - 1103, ACM, 2008.

稀疏自编码器模型

- **Sparse Auto-Encoder**，稀疏自编码器。
- 稀疏自编码器就是在普通的自编码器网络上，加上了稀疏性限制。稀疏性限制可以理解为：使用**sigmoid**激活函数时，大部分的隐层神经元的激活值都接近**0.**
- 激活度：隐层神经元的激活值的平均值为激活度。
- 稀疏性限制：为激活度设置一个阈值，迫使大部分的神经元激活值接近于**0.**

https://web.stanford.edu/class/archive/cs/cs294a/cs294a.1104/sparseAutoencoder.pdf

多模态深度自编码器

- 多模态深度自编码器
  - 融合不同模态的数据，得到高级表示。
  - 可以看做是一种特征变换。

(a) Video-Only Deep Autoencoder

(b) Bimodal Deep Autoencoder

Ngiam, Jiquan, et al. "Multimodal deep learning." *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011.

多模态深度自编码器的应用

- 使用多模态深度自编码器融合脑电和眼动特征
  - 无监督地融合脑电和眼动特征，得到融合后的高层次特征
  - 对融合后的高层次特征进行建模，进行情绪识别。
  - 相较于**SVM**，在SEED3类情绪数据集上取得了更好的结果。

Convolutional Neural Networks

Position-, rotation-, and scale-invariant

- The image is scale-invariant, rotation-invariant and position-invariant.
- However, some image processing methods, e.g. traditional CNN, lack geometric invariance, which limits their robustness for classification and matching of highly variable scenes.

Position-invariant

- **Figures below show the translation and the classification accuracies of traditional CNN and MOP-CNN* for four randomly selected classes: arrival gate, florist shop, volleyball court, and ice skating.**

**Gong Y, Wang L, Guo R, et al. Multi-scale orderless pooling of deep convolutional activation features. European conference on computer vision. Springer, Cham, 2014: 392-407.*

Rotation-invariant

- **Figures below show the rotation with different degree and the classification accuracies of traditional CNN and MOP-CNN* for four randomly selected classes: arrival gate, florist shop, volleyball court, and ice skating.**

**Gong Y, Wang L, Guo R, et al. Multi-scale orderless pooling of deep convolutional activation features. European conference on computer vision. Springer, Cham, 2014: 392-407.*

Scale-invariant

- **Figures below show the picture with different scaling ratio and the classification accuracies of traditional CNN and MOP-CNN* for four randomly selected classes: arrival gate, florist shop, volleyball court, and ice skating.**

Gong Y, Wang L, Guo R, et al. Multi-scale orderless pooling of deep convolutional activation features. European conference on computer vision. Springer, Cham, 2014: 392-407.

卷积神经网络

- CNN的生物学基础
- CNN的历史
- CNN的结构元件
- CNN的结构：LeNet-5
- CNN VS 全连接网络

CNN的生物学基础: Hubel & Wiesel

- 猫的大脑中第一层的视觉神经元（V1），只处理局部视觉中的简单基础结构信息（如某一方向的直线、点），且一种神经元只对一种特定的基础结构做出反应。
- 而且V1神经元是会保留拓扑结构信息的，即相邻的神经元的感受野在图像中也相邻。
- 视觉神经元是有层次的。

Hubel, David H., and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." *The Journal of physiology* 160.1 (1962): 106-154.

CNN的生物学基础

- 猫的大脑中第一层的视觉神经元（V1），只处理局部视觉中的简单基础结构信息（如某一方向的直线、点），且一种神经元只对一种特定的基础结构做出反应。

CNN的历史

- **Fukushima 1980: Neurocognitron**

  (“三明治” 结构, 局部感受野, 池化层)

Prof. Fukushima

CNN的历史

- Lecun 1998: LeNet-5 (工业界的应用：手写数字识别）

CNN的历史

- Krizhevsky, Sutskever, Hinton, 2012: AlexNet (IMAGENET数据集上分类准确率远高于传统方法)

CNN的结构元件

- 卷积的定义
- 卷积层
- 激活层
- 池化层
- 全连接层
- 损失函数

卷积的定义

- 在泛函分析中，卷积(**Convolution**)是通过两个函数**f**和**g** 生成第三个函数的一种数学算子，表征函数**f**与**g**经过翻转和平移的重叠部分的面积。如果将参加卷积的一个函数看作区间的指示函数，卷积还可以被看作是“滑动平均”的推广。
- 卷积是在信号与线性系统的基础上或背景中出现的，讨论的就是信号经过一个线性系统以后发生的变化（就是输入，输出和所经过的所谓系统，这三者之间的数学关系）。
- 实际上，线性系统的传递函数和输入信号，在数学上的形式就是所谓的卷积关系(一维卷积)。而数字图像处理中的卷积关系是二维卷积。

离散一维卷积

对于两个长度为m和n的序列$f(i)$和$g(j)$，

$$h(i)=f(i)*g(i)=\sum_{j}f(j)g(i-j)$$

给出长度为$N=m+n-1$的输出序列。

其矩阵计算形式为

$$\mathbf{h}=\mathbf{g}\bullet\mathbf{f}=\begin{bmatrix}\mathbf{g_p}(1) & \mathbf{g_p}(\mathrm{N}) & \cdots & \mathbf{g_p}(2)\\ \mathbf{g_p}(2) & \mathbf{g_p}(1) & \cdots & \mathbf{g_p}(3)\\ \vdots & \vdots & \vdots & \vdots\\ \mathbf{g_p}(\mathrm{N}) & \mathbf{g_p}(\mathrm{N}-1) & \cdots & \mathbf{g_p}(1)\end{bmatrix}\begin{bmatrix}\mathbf{f_p}(1)\\ \mathbf{f_p}(2)\\ \vdots\\ \mathbf{f_p}(\mathrm{N})\end{bmatrix}$$

离散二维卷积

- 将卷积形式从一维推广到二维，我们得到离散二维卷积的形式如下。二维卷积一般被应用在图像预处理之中(平滑，滤波等)。此时，**F**为卷积函数(又称算子，卷积核)，**G**为原始图像，**H**为经过卷积后的图像。

$$H = F * G$$

$$H(i,j) = \sum_m \sum_n F(m,n) G(i-m, j-n)$$

Locally connected and Share Weights

- Example: 1000*1000 image
  - Fully-connected, 1000,000 hidden units
  - 1000 *1000*1000,000= 10^{12} parameters
  - Locally-connected, 1000,000 hidden units 10x10 fields = 10^{8} parameters
  - Local connections capture local dependencies
  - 1000,000 hidden units share the same weights
  - The number of parameters are decreased from 10^{8} to 100!

CNN的结构元件：卷积层

- O = convolution(I , filters)
  - O = {convolution(I , filter), for filter in filters}
  - 每个滤波器生成输出的一个通道，滤波器数=输出通道数
- 输入I：$H*W*C_1$，其中H,W分别是输入的高度和宽度，C_1是输入的通道数（深度）。
- 滤波器组filters：C_2*滤波器，其中C_2是滤波器的数目。
  - 滤波器：F*F*{C}+bias+P+S，其中F是局部感受野的边长，bias是偏移量 (通常为1)，P是填充宽度，S是步长，{C}是输入通道的组合，通常{C}={1,2,...,C_1}
- 输出O：$H'*W'*C_2$，由输入和滤波器组唯一确定。
  - H'=(H-F+2*P)/S+1，W'=(W-F2*P)/S+1，C_2=C_2

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

CNN的结构元件：卷积层

- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积
- 窗口按照给定的步长**S**在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。

Image

Convolved Feature

CNN的结构元件：池化的意义

- 在卷积神经网络中，经常会进行池化操作，池化层往往在卷积层后面，通过池化来降低卷积层输出的特征向量，同时改善结果（不易出现过拟合）。
- 图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在另一个区域同样适用。因此，为了描述大的图像，一个很自然的想法就是对不同位置的特征进行聚合统计，例如，人们可以计算图像一个区域上的某个特定特征的平均值 (或最大值)来代表这个区域的特征。这就是池化操作，通过池化我们可以降低特征的维度。

CNN的结构元件：池化层

- 池化层：**F*F+S**，其中**F**是窗口长度，**S**是移动步长。
- 常见的**CNN**都是使用最大化池化层，即输出窗口中的最大值。
- 均值池化层(输出窗口内的均值)，最近常用于替代全连接层。
- 下图是**F=2,S=2**最大化池化的例子：

CNN的结构元件：全连接层&损失函数

- 全连接层：常见的前向传播网络，每个神经网络节点与前一层的所有节点连接。
  - 全连接层可以看做**F=H=W, P=0**的卷积层。
  - 大型网络中常用均值池化层替代全连接层(减少参数个数)
- 损失函数：常见的有交叉熵，**hinge,** 平均平方误差等。

CNN的典型结构

- 每层卷积层之后都接激活层
- 若干层卷积层之后接池化层
- 最后使用全连接层+损失函数
- LeNet, AlexNet, VGGNet都符合这一范式

LeNet-5

- 早在1989年，Yann LeCun (纽约大学; Facebook)和他的同事们就发表了卷积神经网络（Convolution Neural Networks， 简称CNN）的工作。
- CNN是一种带有卷积结构的深度神经网络，通常至少有两个非线性可训练的卷积层，两个非线性的固定卷积层（又叫Pooling Laye）和一个全连接层，一共至少5个隐含层。
- LeNet-5就是一个典型的CNN结构。

LeNet-5的功能

- LeNet-5是应用在手写数字识别任务中的。
- **输入：32*32**的手写字体图片，这些手写字体包含**0~9**数字，也就是相当于**10**个类别的图片。
- **输出：**分类结果，**0~9**之间的一个数。
- 因此我们可以知道，这是一个多分类问题，总共有十个类，因此神经网络的最后输出层必然是**SoftMax**问题，然后神经元的个数是**10**个。

LeNet-5: C1层

- 输入图片大小：32*32
- 卷积窗大小：5*5
- 卷积窗种类: 6
- 输出特征图数量：6
- 输出特征图大小：28*28 (32-5+1)
- 神经元数量：4707 [(28*28)*6)]
- 连接数： 12304 [(5*5+1)*6]*(28*28)
- 可训练参数：156 [(5*5+1)*6]

LeNet-5: C1层

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: la
120
Convolutions
Subsampling
Convolutions
Subsamp

LeNet-5: S2层

- 输入图片大小：(28*28)*6
- 卷积窗大小：　2*2
- 卷积窗种类：　6
- 输出亚采样图数量：6
- 输出亚采样图大小：(14*14)*6
- 神经元数量：　1176 (14*14)*6
- 连接数：　5880 (4+1)*(14*14)*6
- 可训练参数：　12(6*2)
- 每个单元的四个输入相加，乘以一个可训练参数，再加上一个可训练偏置，结果通过Sigmoid函数计算

LeNet-5: S2层

C1: feature maps
6@28x28
C3: f. maps 16@10x10
S4: f. maps 16@5x5
S2: f. maps
6@14x14
C5: la
120
ıvolutions
Subsampling
Convolutions
Subsamp

LeNet-5：C3层

输入图片大小：　(14*14)*6

卷积窗大小：　5*5

卷积窗种类：　15

输出特征图数量：　16

输出特征图大小：　10*10　(14-5+1)

神经元数量：　1600 [(10*10)*16)]

连接数：　151600 [(60+16)*25]*(10*10)（部分连接）

60+16)*25]

LeNet-5：C3层

C1: feature maps
6@28x28
C3: f. maps 16@10x10
S4: f. maps 16@5x5
S2: f. maps
6@14x14
C5:
120
volutions
Subsampling
Convolutions
Subsar

LeNet-5：S4层

输入图片大小：(10*10)*16

卷积窗大小：2*2

卷积窗种类：16

输出下采样图数量：16

输出下采样图大小：(5*5)*16

神经元数量： (5*5)*16=400

连接数：(4+1)*(5*5)*16= 2000

可训练参数：(16*2)= 32

LeNet-5：C5层（NN）

输入图片大小：(5*5)*16

卷积窗大小：5*5

卷积窗种类：120

输出特征图数量：120

输出特征图大小：1*1 (5-5+1)

神经元数量：120 (1*120)

连接数：（16*5*5+1）*120(全连接）=48120

可训练参数：（16*5*5+1）*120=48120

LeNet-5：F6层（NN）

输入图片大小：　(1*1)*120

卷积窗大小：　1*1

卷积窗种类：　84

输出特征图数量：　84

输出特征图大小：　1

神经元数量：　84

连接数：（120+1）*84=10164（全连接）

可训练参数：（120+1）*84=10164

LeNet-5：OUTPUT层（NN）

输入图片大小：　1*84

输出特征图数量：　1*10

LeNet-5： MINIST Dataset

60,000 original datasets

Test error: 0.95%

540,000 artificial distortions

+ 60,000 original

Test error: 0.8%

LeNet-5: Misclassified Examples

LeNet-5：OUTPUT层（NN）

- Trained with stochastic gradient descent
- on two NVIDIA GTX 580 3GB GPUs
- for about a week

- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- 5 convolutional layer, 3 fully connected layer
- Final feature layer: 4096-dimensional

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton,

Advances in Neural Information Processing Systems 2012

ImageNet

- 15M images
- 22K categories
- Images collected from Web
- Human labelers (Amazon's Mechanical Turk crowd-sourcing)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
    - 1K categories
    - 1.2M training images (~1000 per category)
    - 50,000 validation images
    - 150,000 testing images

- RGB images
- Variable-resolution, but this architecture scales them to 256x256 size

ImageNet-1

Classification goals:

- ❑ Make 1 guess about the label (Top-1 error)
- ❑ make 5 guesses about the label (Top-5 error)

The CNN Architecture

The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in the kernel map. 224/4=56

The pooling layer: form of non-linear down-sampling. Max-pooling partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum value

Experimental Setting

- Trained with stochastic gradient descent
- on two NVIDIA GTX 580 3GB GPUs
- for about a week

- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- 5 convolutional layer, 3 fully connected layer
- Final feature layer: 4096-dimensional

The First Convolutional Layer

96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images.

The top 48 kernels were learned on GPU1 while the bottom 48 kernels were learned on GPU2

Experimental Results

Results on the test data:

top-1 error rate: 37.5%

top-5 error rate: 17.0%

ILSVRC-2012 competition:

15.3% accuracy

2nd best team: 26.2% accuracy

CNN VS 全连接网络

- 局部感受野
  - 保留图像的拓扑结构
  - 提取图像的基础特征
- 权值共享
  - 减少可训练参数→防止过拟合
  - 基础特征适用于图像的所有位置→平移、扭曲不变性
- 降采样
  - 提取层次化特征
  - 减少具体位置对于最终结果的影响→平移、扭曲不变性

参考资料

- Hubel, David H., and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." *The Journal of physiology* 160.1 (1962): 106-154.
- Hubel, David H., and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." The Journal of physiology 148.3 (1959): 574-591.
- Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." Competition and cooperation in neural nets. Springer Berlin Heidelberg, 1982. 267-285.
- LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

参考资料-1

- UNDERSTANDING CONVOLUTIONAL NEURAL NETWORKS FOR NLP [http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/]
- CS231n: Convolutional Neural Networks for Visual Recognition [http://cs231n.stanford.edu/]
- Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
- He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

Deep Learning Reading List

1. **Deep learning**(2015), Y Lecun, Y Bengio， G Hinton（review）
2. **Teaching machines to read and comprehend** (2015), K. Hermann et al. (Natural Language Processing)
3. **Batch normalization: Accelerating deep network training by reducing internal covariate shift** (2015), S. Loffe and C. Szegedy. (Optimization / Training Techniques)
4. **Inception-v4, inception-resnet and the impact of residual connections on learning** (2016), C. Szegedy et al. (Convolutional Neural Network Models)
5. **You only look once: Unified, real-time object detection** (2016), J. Redmon et al. (Image: Segmentation / Object Detection)

Deap Learning Reading List-1

6. **Fully convolutional networks for semantic segmentation** (2015), J. Long et al.. (Image: Segmentation / Object Detection)
7. **Deep residual learning for image recognition** (2016), K. He et al. (Convolutional Neural Network Models)
8. **Fast R-CNN** (2015), R. Girshick.(Image: Segmentation / Object Detection)
9. **Image Super-Resolution Using Deep Convolutional Networks** (2016), C. Dong et al. (Image / Video / Etc)
10. **Neural Architectures for Named Entity Recognition** (2016), G. Lample et al (Natural Language Processing)
11. **End-to-end attention-based large vocabulary speech recognition** (2016), D. Bahdanau et al. (Speech / Other Domain)
12. **Learning deep physiological models of affect,** HP Martinez, Y Bengio, GN Yannakakis