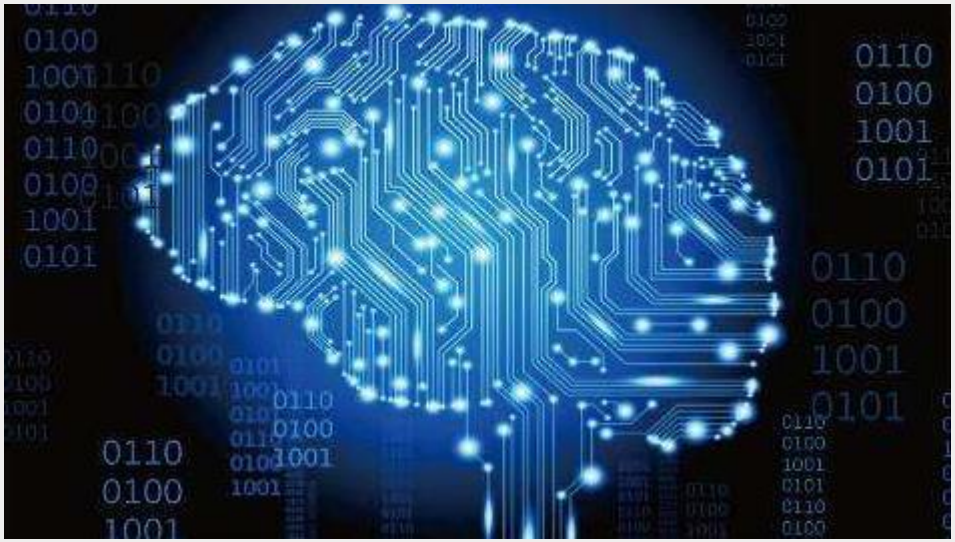
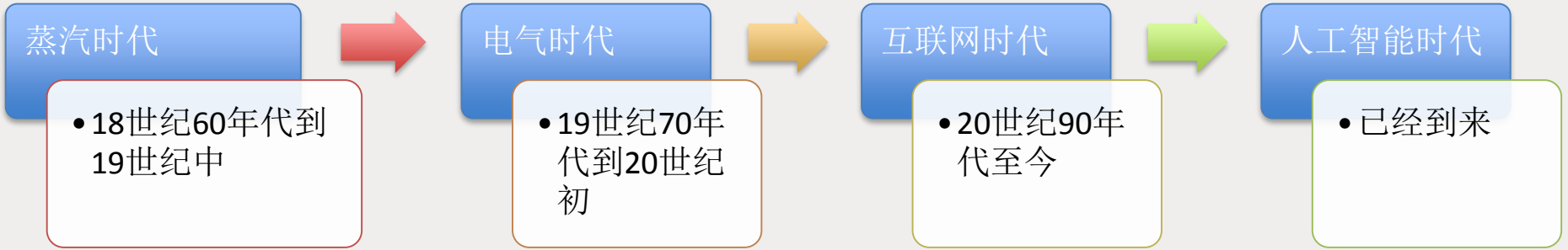
人工智能大时代



一、人工智能时代来啦


蒸汽时代
•18世纪60年代到19世纪中
电气时代
•19世纪70年代到20世纪初
互联网时代
•20世纪90年代至今
人工智能时代
•已经到来

AI诞生

1956

达特茅斯会议

低谷

1970-1980

大规模数据和复杂任务不能完成，计算能力无法突破

专家系统

1982后

神经网络+5代计算机

低谷

1990-2000

DARPA无法实现，政府投入缩减

深度学习

2006-至今

突破性进展，进入发展热潮

1957

Perception 感知神经网络的发明

1986

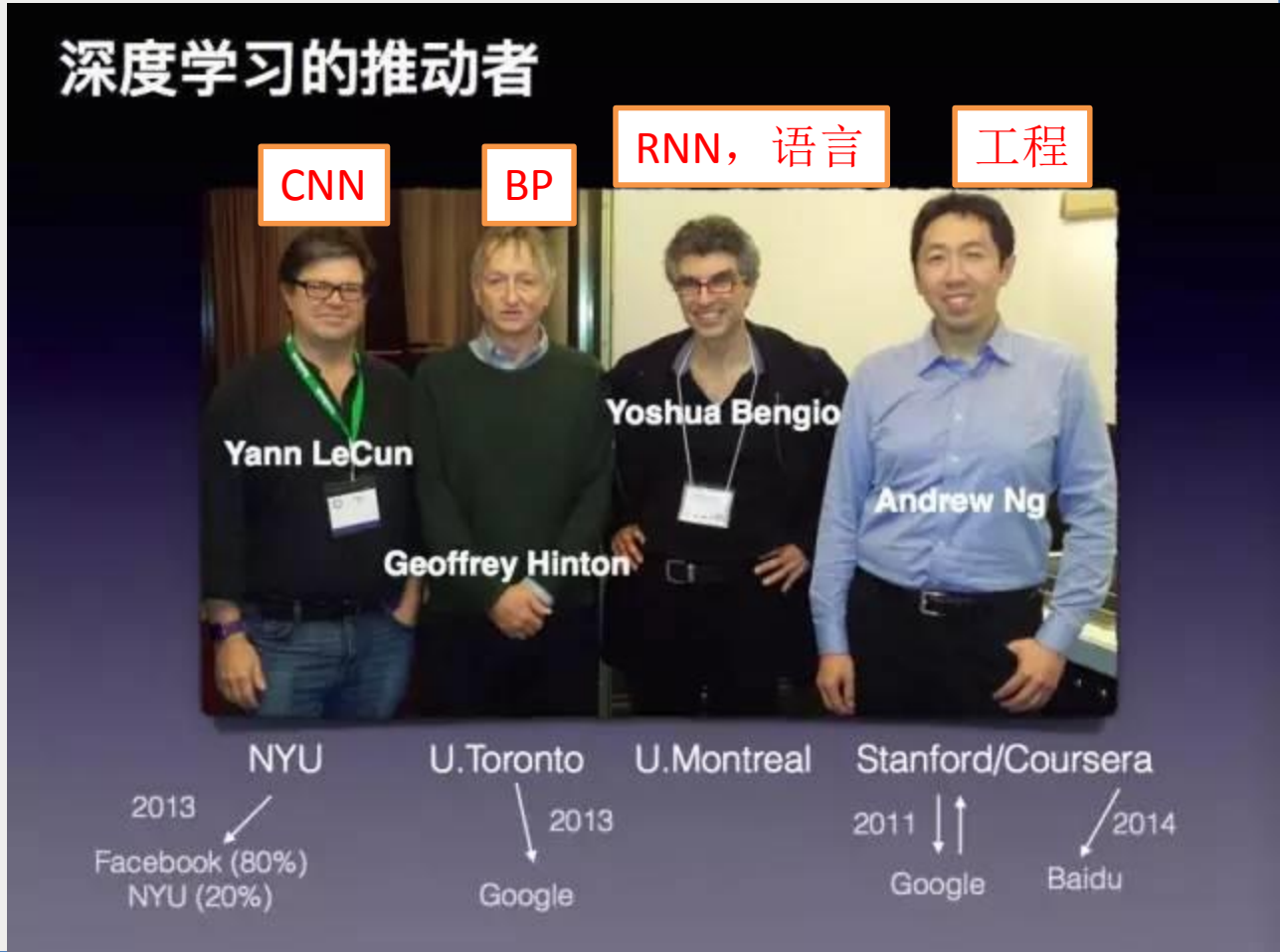
BP 神经网络算法的发明和应用


Google
智能家政
Nest
Google Home
Care .com
Spring Board
手术机器人
智能家居
芯片
感应血糖隐形眼镜
芯片
机器人
App Allo
TPU
无人驾驶汽车
语音
Google Assistant
Brillo
物联网
技术开源
Google Now
Weave
视觉
Tenson Flow
Project Glass
Deep Dream

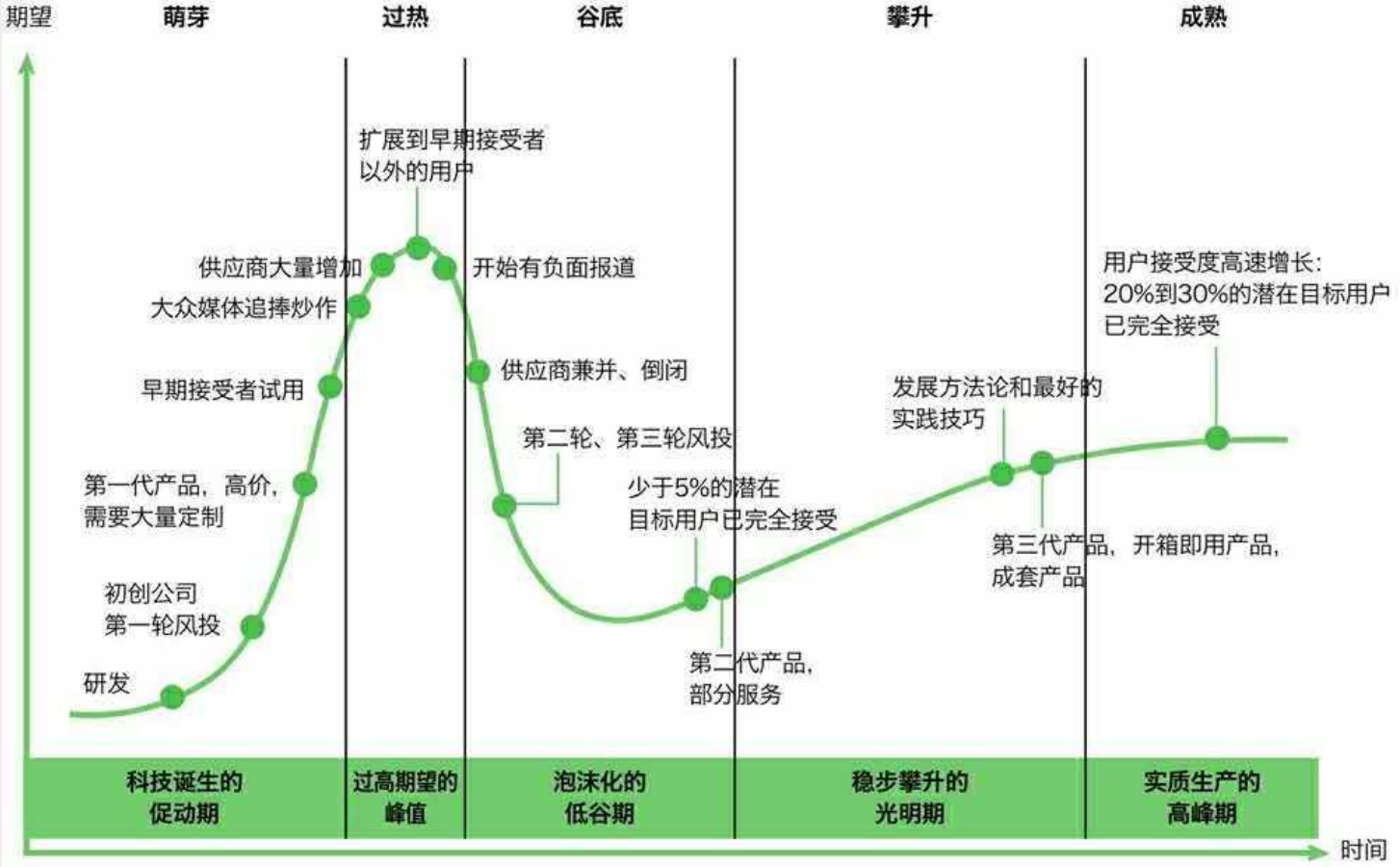
本次人工智能的兴起主要有以下几个特点

一、人工智能时代来啦

hinton是教主，负责挖坑，挖完坑就跑
lecun是独行侠，负责东搞西搞
bengio是金牌打手，负责理论和实验支持



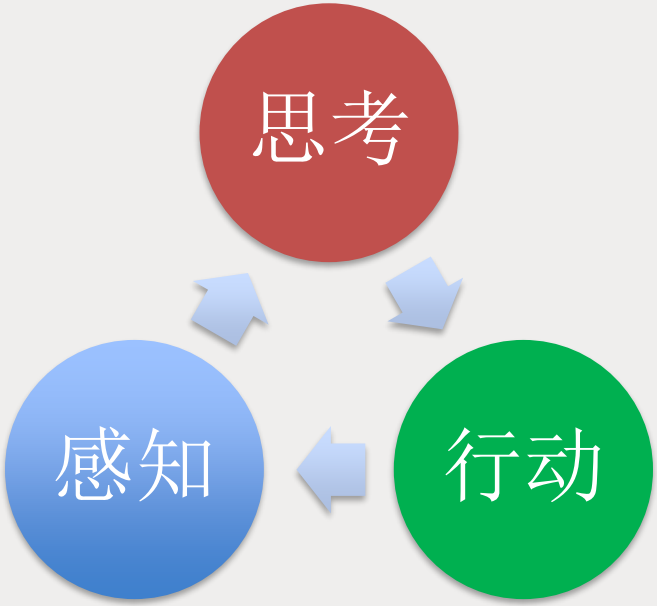
一、人工智能时代来啦


期望
萌芽
过热
谷底
攀升
成熟
扩展到早期接受者
以外的用户
供应商大量增加
开始有负面报道
大众媒体追捧炒作
供应商兼并、倒闭
早期接受者试用
第二轮、第三轮风投
第一代产品，高价，
需要大量定制
少于5%的潜在
目标用户已完全接受
初创公司
第一轮风投
研发
第二代产品，
部分服务
发展方法论和最好的
实践技巧
用户接受度高速增长：
20%到30%的潜在目标用户
已完全接受
第三代产品，开箱即用产品，
成套产品
科技诞生的
促动期
过高期望的
峰值
泡沫化的
低谷期
稳步攀升的
光明期
实质生产的
高峰期
时间

二、什么是人工智能

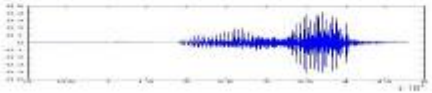
- 人工智能 = 大数据 + 深度学习
- 必要条件
  - 数据
    - 数据收集成本下降
    - 数据储存成本下降
    - 数据处理成本下降
  - 计算能力

二、什么是人工智能



二、什么是人工智能

- Speech Recognition

$f($  $)=$ “How are you”

- Image Recognition

$f($  $)=$ “Cat”

- Playing Go

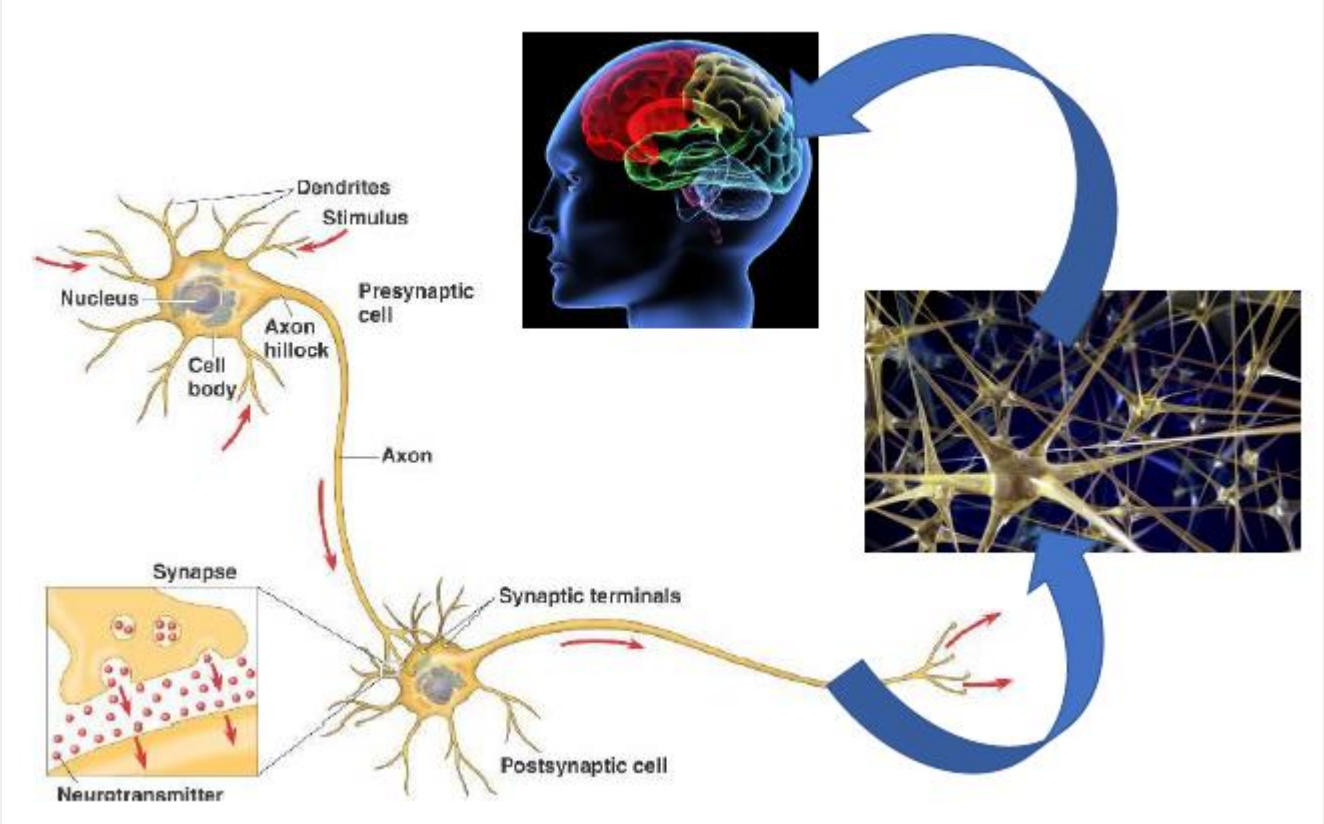
$f($  $)=$ “5-5” (next move)

- Dialogue System

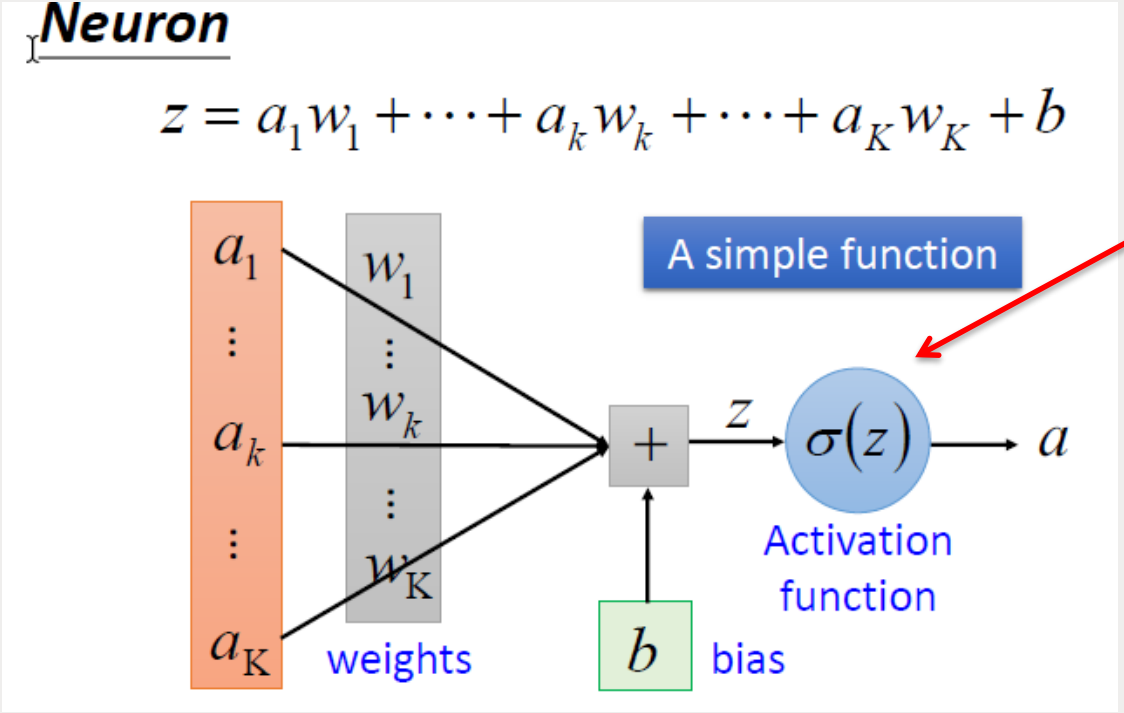
$f($ “Hi” $)=$ “Hello”

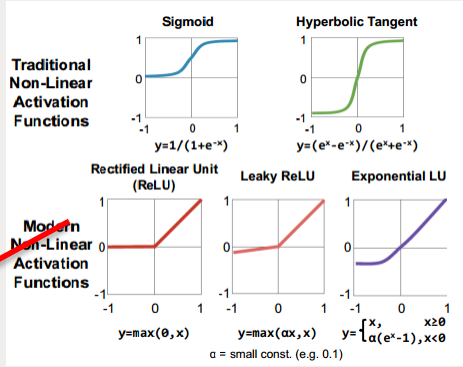
(what the user said) (system response)

二、什么是人工智能


Dendrites
Stimulus
Nucleus
Presynaptic cell
Axon hillock
Cell body
Axon
Synapse
Synaptic terminals
Postsynaptic cell
Neurotransmitter

机器（深度）学习：基本单元



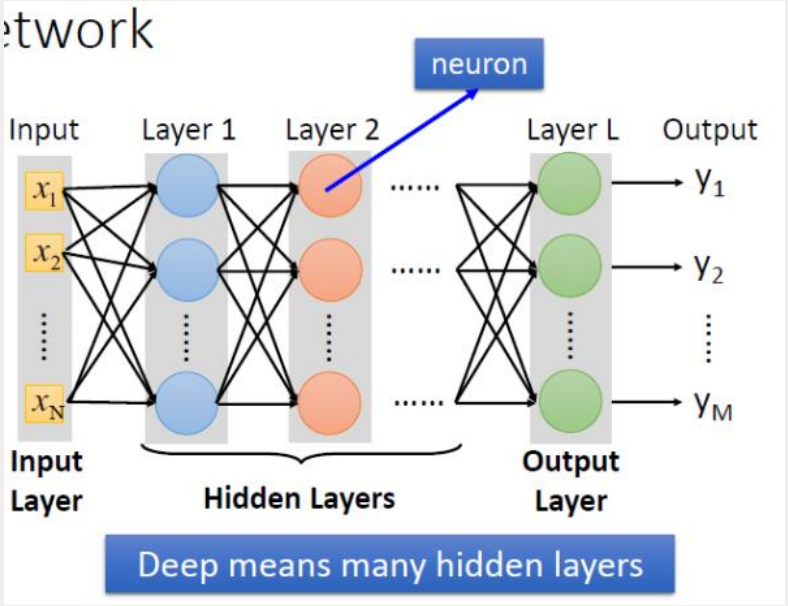


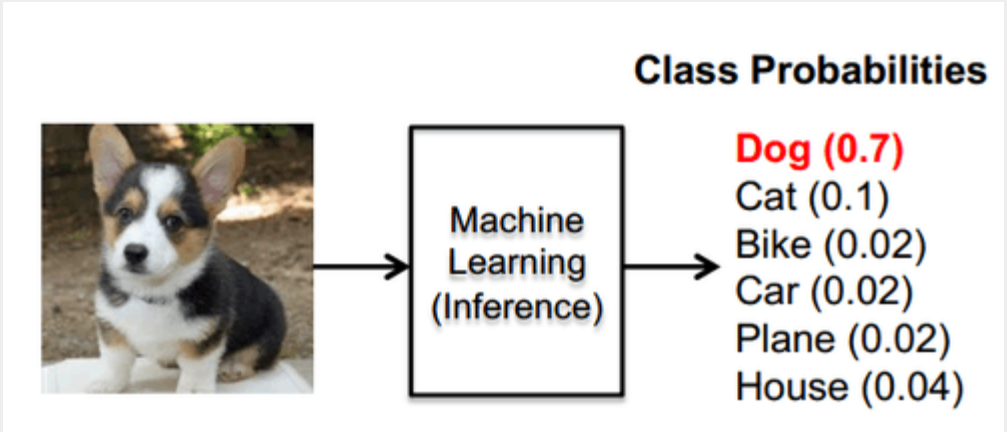
激活函数：sigmoid，relu，tanh等（根据需要选择）

目的：使线性函数变成非线性函数

单一神经单元

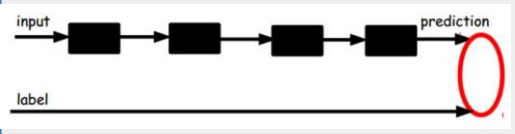
机器（深度）学习：深度神经网络

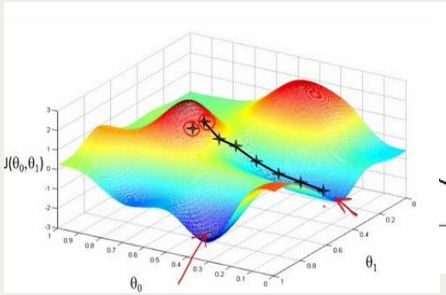


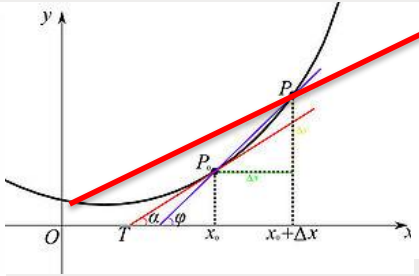


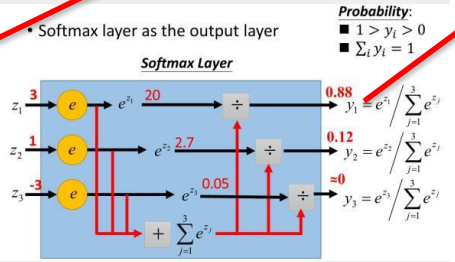
深度学习过程：

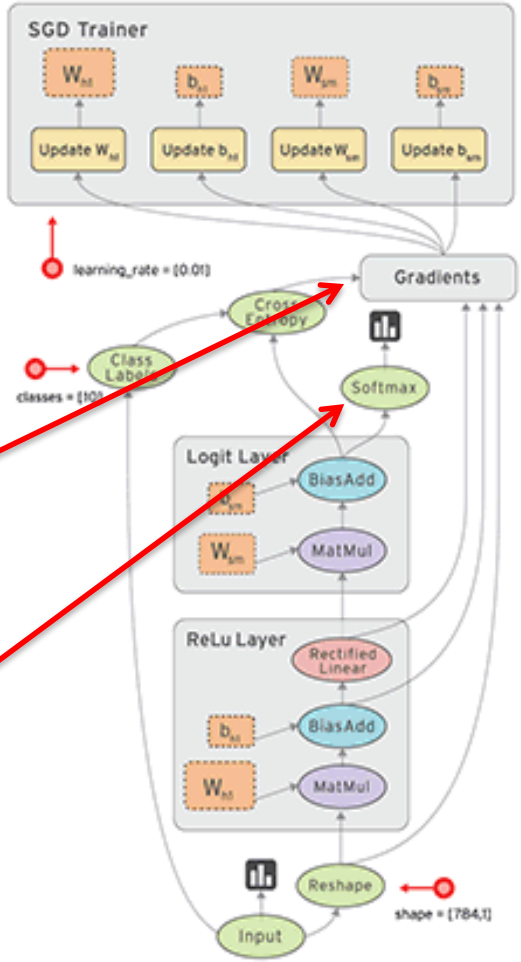
1.设计网络模型
2.准备训练的数据（带有标签）；
3.训练：
---前向计算，结果和标签比较。计算误差。
---反向计算：对误差求导（梯度下降），确定W,B参量
---循环以上过程。









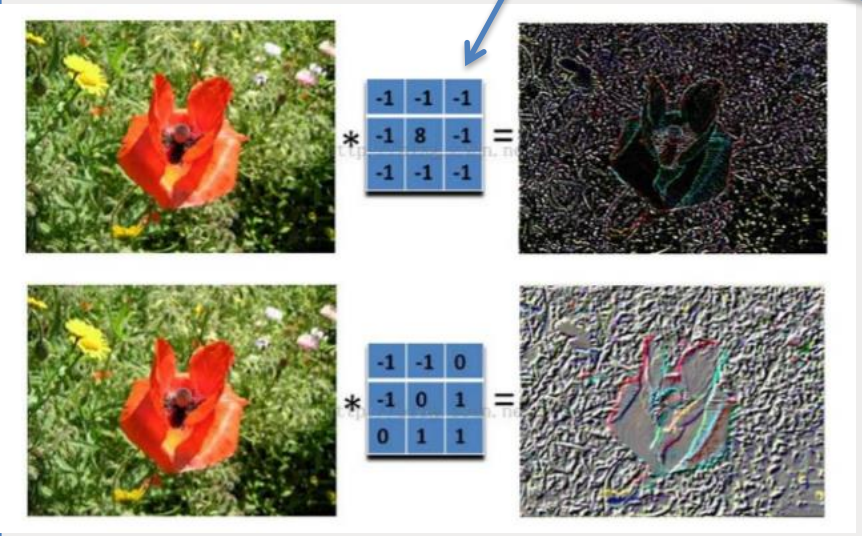


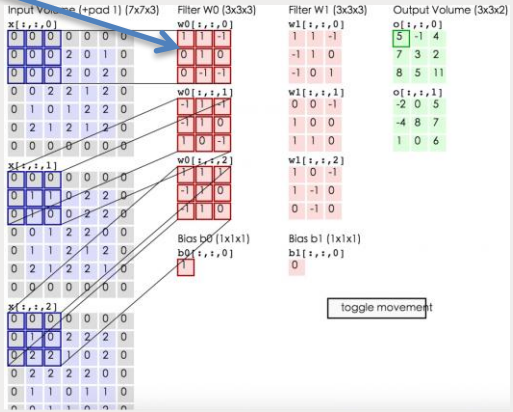
4.学习得到的模型（包括参数），就可以做图像识别等

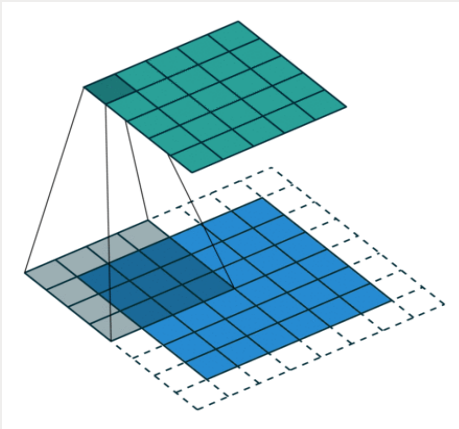
为什么用CNN（卷积）：当然还有RNN,LSTM等

- 全连接网络，参数多，计算量庞大；
- 考虑到图像等，是局部关联，所以使用卷积网络，权值共享的方式。

卷积核

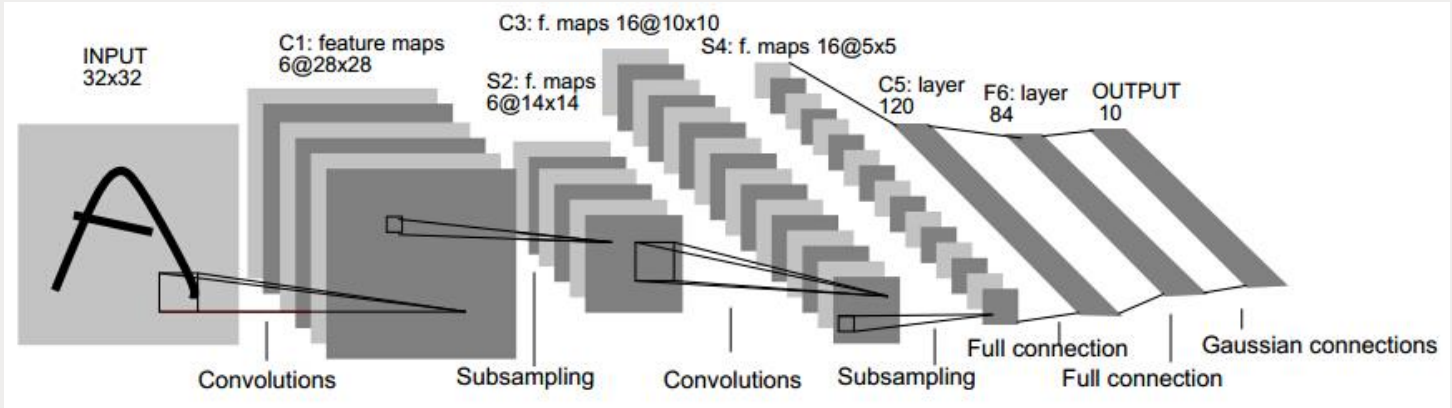






企业界正在在干什么？

1. 使用google tensorflow等类似平台，拿来现成的网络模型（如果有类似开源项目，可以拿来学习好的模型），做自己垂直业务的工程项目；
2. 优化模型（新的模型和去掉接近0参量），使其能用在更小的芯片系统上。
3. 为了识别更好，厂家想办法扩大自己垂直领域的“标记数据”。
4. 什么样的人才合适：数学基础好、编程能力，进一步：图像、语音领域知识。



网络模型：Alexnet(2012)

学术界正在在干什么？

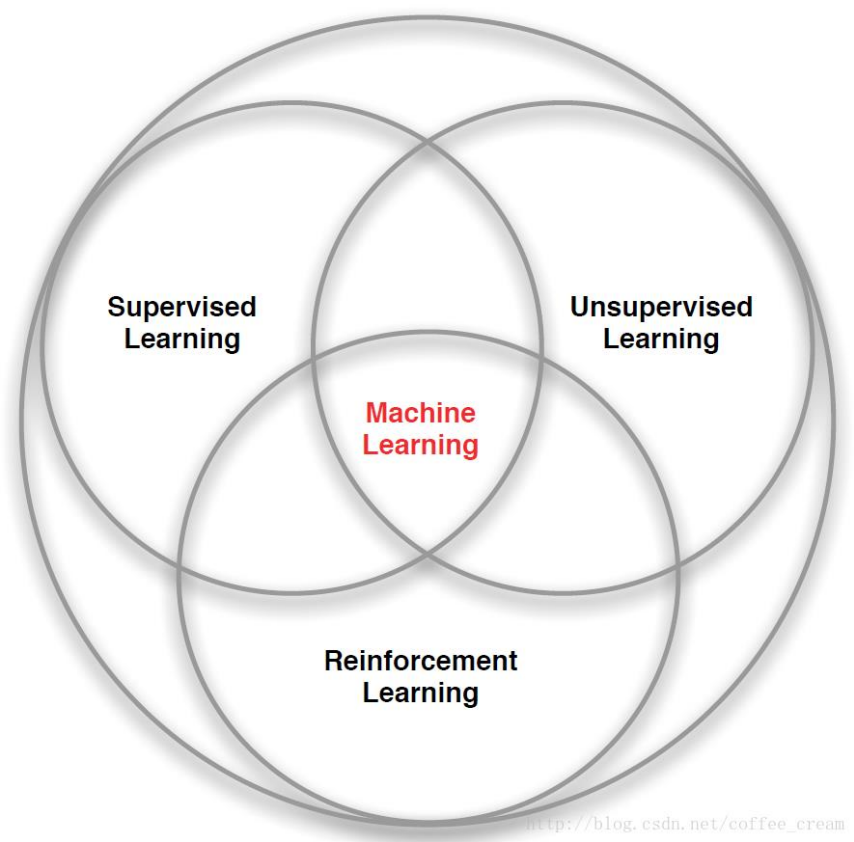
1. 深度学习网络参数的优化：基本图像网络感知网络模型已经相对完善。
2. 如何减少乘法器？如何减少W和b参数个数？（从模型结构入手，例如：mobilenet；压缩模型参数：去掉接近0的参量等方法）
3. 人体姿态、人脸表情等：跟踪人动作，理解人表情
4. 图片分割：物体跟踪
5. 图像语意：一句话表达图像。
6. 视频语意：分析视频情节等等

7. 语音方面：识别，TTS，NLP等。

更深一层：深度网络的基础解释理论、看一眼就会等研究。

二、什么是人工智能

1. 监督学习，优化的迁移学习
2. 非监督学习
3. 强化学习：例如，机器学习玩游戏（奖励机制）



The standard type of RBM has binary-valued (Boolean/Bernoulli) hidden and visible units, and consists of a matrix of weights $W = (w_{i,j})$ (size $m \times n$) associated with the connection between hidden unit h_j and visible unit v_i, as well as bias weights (offsets) a_i for the visible units and b_j for the hidden units. Given these, the energy of a configuration (pair of boolean vectors) (v,h) is defined as

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

or, in matrix notation,

$$E(v,h) = -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h$$

This energy function is analogous to that of a Hopfield network. As in general Boltzmann machines, probability distributions over hidden and/or visible vectors are defined in terms of the energy function:[9]

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)}$$

where Z is a partition function defined as the sum of $e^{-E(v,h)}$ over all possible configurations (in other words, just a normalizing constant to ensure the probability distribution sums to 1). Similarly, the (marginal) probability of a visible (input) vector of booleans is the sum over all possible hidden layer configurations:[9]

$$P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$$

```
import theano
from theano import tensor as T
from theano.sandbox.rng_mrg import MRG_RandomStreams as RandomStreams
import numpy as np
from load import mnist

srng = RandomStreams()

def floatX(X):
    return np.asarray(X, dtype=theano.config.floatX)

def init_weights(shape):
    return theano.shared(floatX(np.random.randn(*shape) * 0.01))

def rectify(X):
    return T.maximum(X, 0.)

def softmax(X):
    e_x = T.exp(X - X.max(axis=1).dimshuffle(0, 'x'))
    return e_x / e_x.sum(axis=1).dimshuffle(0, 'x')

def RMSprop(cost, params, lr=0.001, rho=0.9, epsilon=1e-6):
    grads = T.grad(cast=cost, wrt=params)
    updates = []
    for p, g in zip(params, grads):
        acc = theano.shared(p.get_value() * 0.)
        acc_new = rho * acc + (1 - rho) * g ** 2
        gradient_scaling = T.sqrt(acc_new + epsilon)
        g = g / gradient_scaling
        updates.append((acc, acc_new))
        updates.append((p, p - lr * g))
    return updates

def dropout(X, p=0.):
    if p > 0:
        retain_prob = 1 - p
        X *= srng.binomial(X.shape, p=retain_prob, dtype=theano.config.floatX)
        X /= retain_prob
    return X
```

太复杂，太多公式

看不懂，不明觉厉

Tensorflow代码示例(部分)：

```
#构建网络
x_image = tf.reshape(x, [-1,28,28,1])          #转换输入数据shape,以便于用于网络中
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)      #第一个卷积层
h_pool1 = max_pool(h_conv1)                                   #第一个池化层

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)       #第二个卷积层
h_pool2 = max_pool(h_conv2)                                    #第二个池化层

W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])               #reshape成向量
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)     #第一个全连接层

keep_prob = tf.placeholder("float")
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)                   #dropout层

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])
y_predict=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)    #softmax层
```

IBM PowerAI Platform

PowerAI Software Distribution

Deep Learning Frameworks

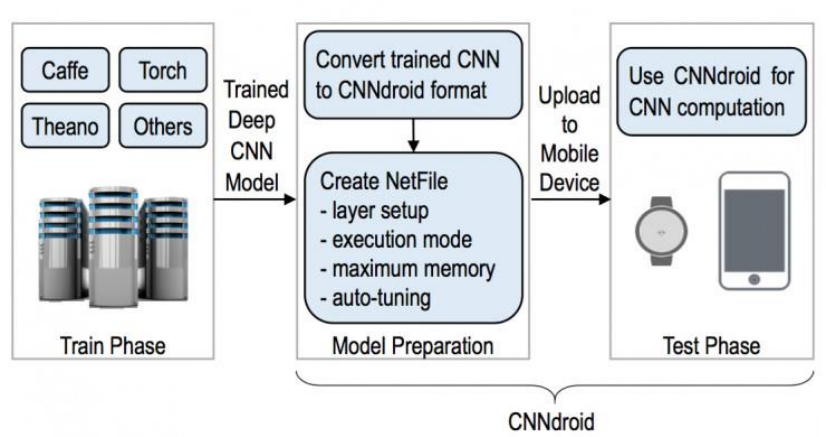
- Caffe
- NVIDIA Caffe
- IBM Caffe
- torch
- TensorFlow
- theano
- Chainer

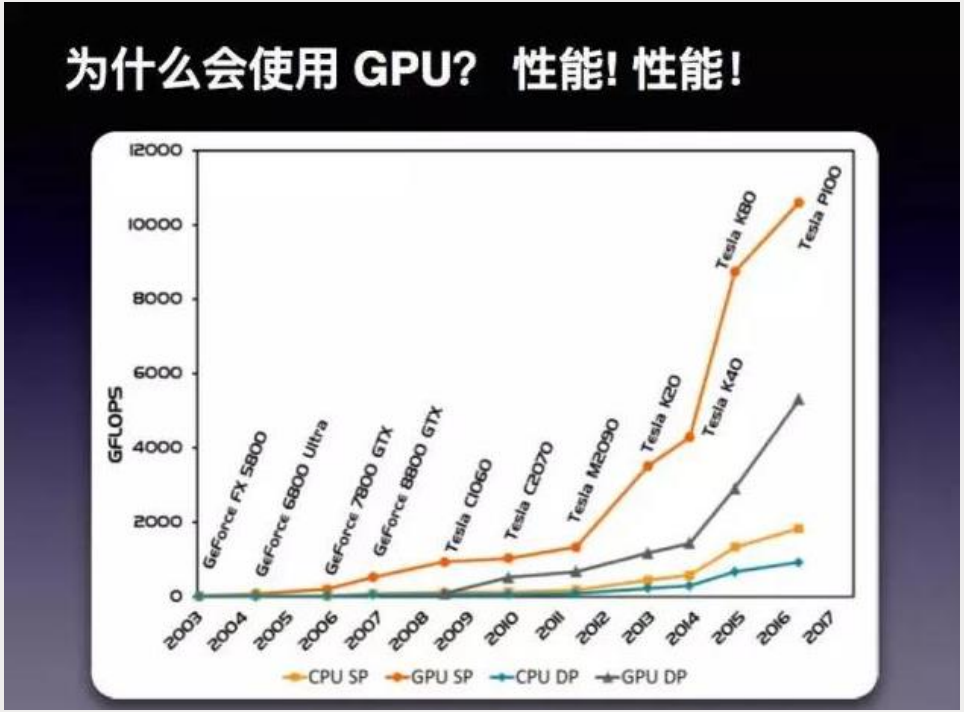
Supporting Libraries

- DIGITS
- OpenBLAS
- Distributed Frameworks
- Bazel
- NCCL

IBM Power System for HPC, with NVLink

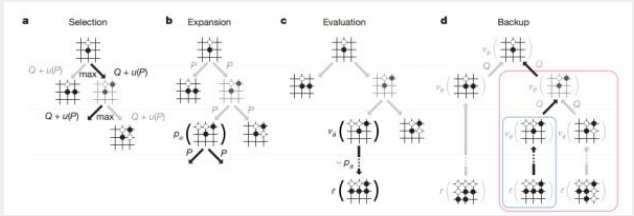
Breakthrough performance for GPU accelerated applications, including Deep Learning and Machine Learning.


Caffe
Torch
Theano
Others
Train Phase
Trained Deep CNN Model
Convert trained CNN to CNNdroid format
Create NetFile
- layer setup
- execution mode
- maximum memory
- auto-tuning
Model Preparation
Upload to Mobile Device
Use CNNdroid for CNN computation
Test Phase
CNNdroid


为什么会使用 GPU? 性能! 性能!
GFLOPS
12000
10000
8000
6000
4000
2000
0
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
GeForce FX 5800
GeForce 6800 Ultra
GeForce 7800 GTX
GeForce 8800 GTX
Tesla C1060
Tesla C2070
Tesla M2090
Tesla K20
Tesla K40
Tesla K80
Tesla P100
CPU SP
GPU SP
CPU DP
GPU DP

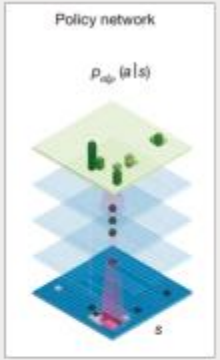
三、人工智能：能干什么？

Alpha zero



d=6

0 0 0 0
0 1 -1 1
0 1 -1 0
0 -1 0 0

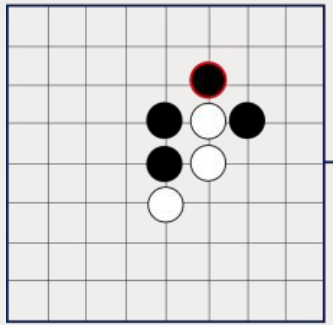


13 层 CNN
深度学习模型

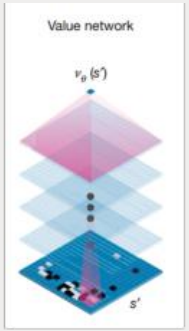
d=7

0 0.1 0.5 0.3
0 1 -1 1
0 1 -1 0
0 -1 0.1 0

d=7



S'（状态）



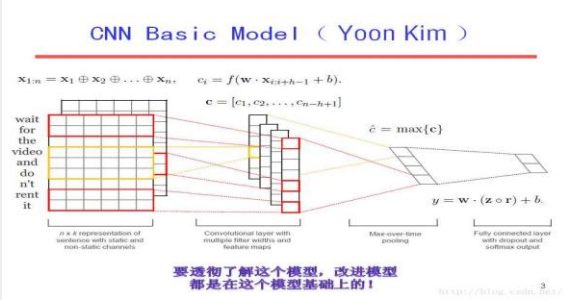
V(S')

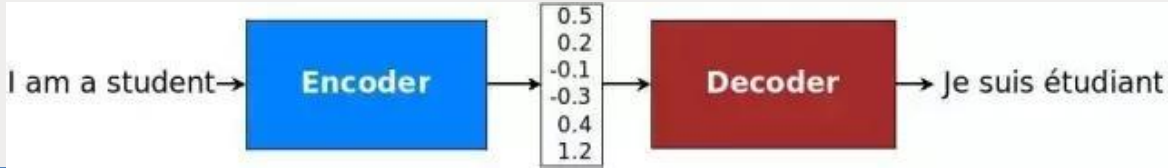
13 层 CNN
深度学习模型

图像识别、分割、跟踪等：

超级分辨率：

语音领域：NLP,STT,TTS：





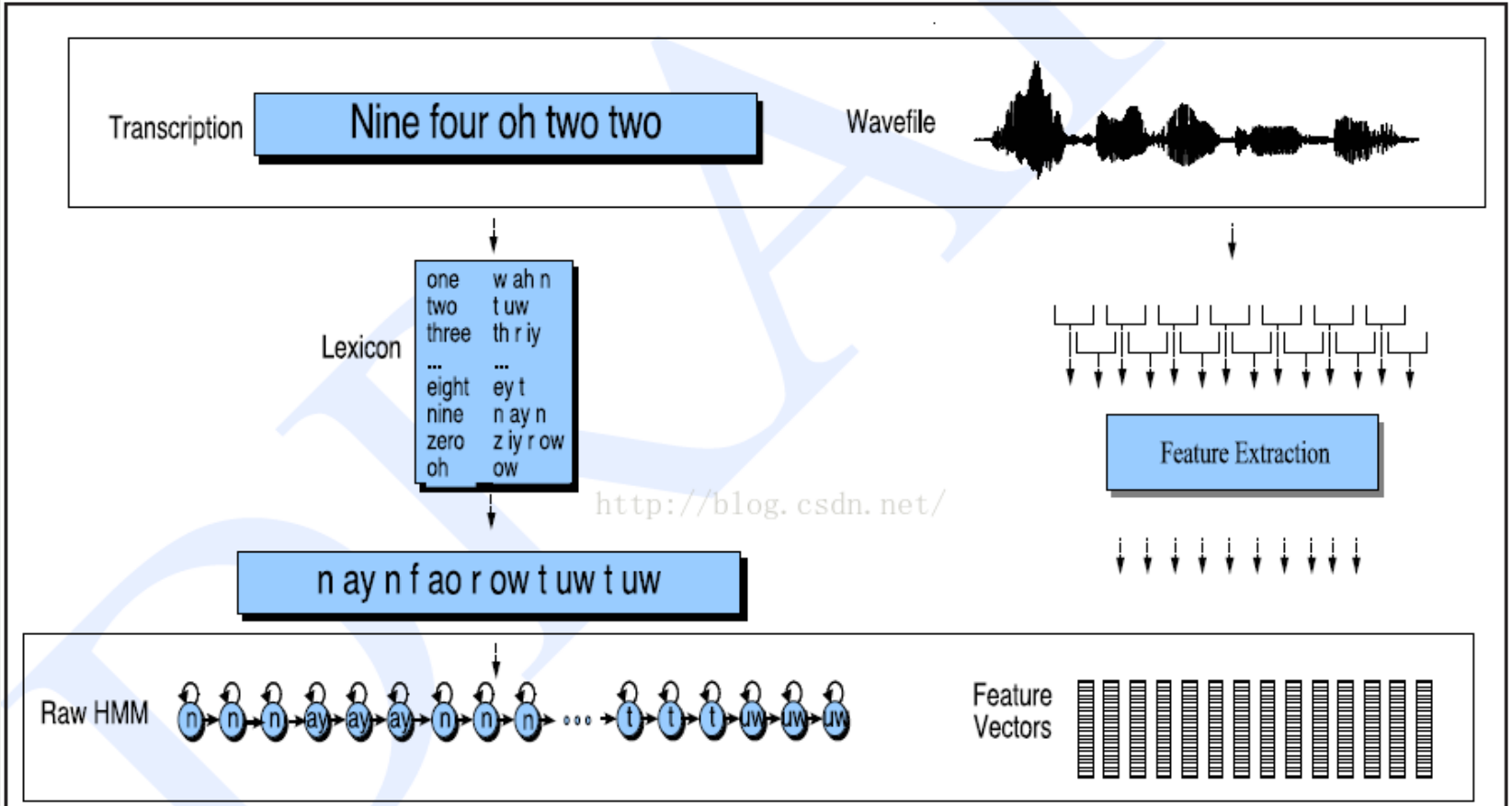
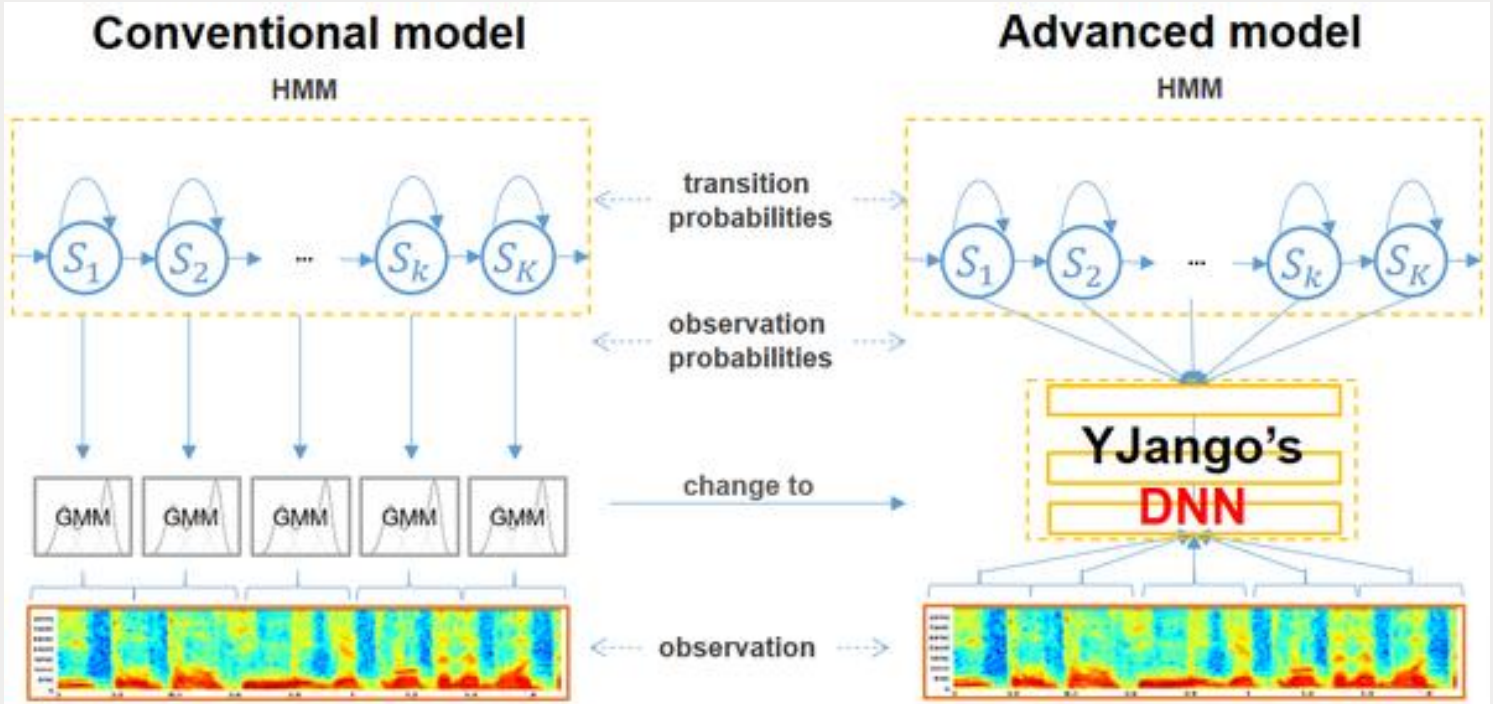
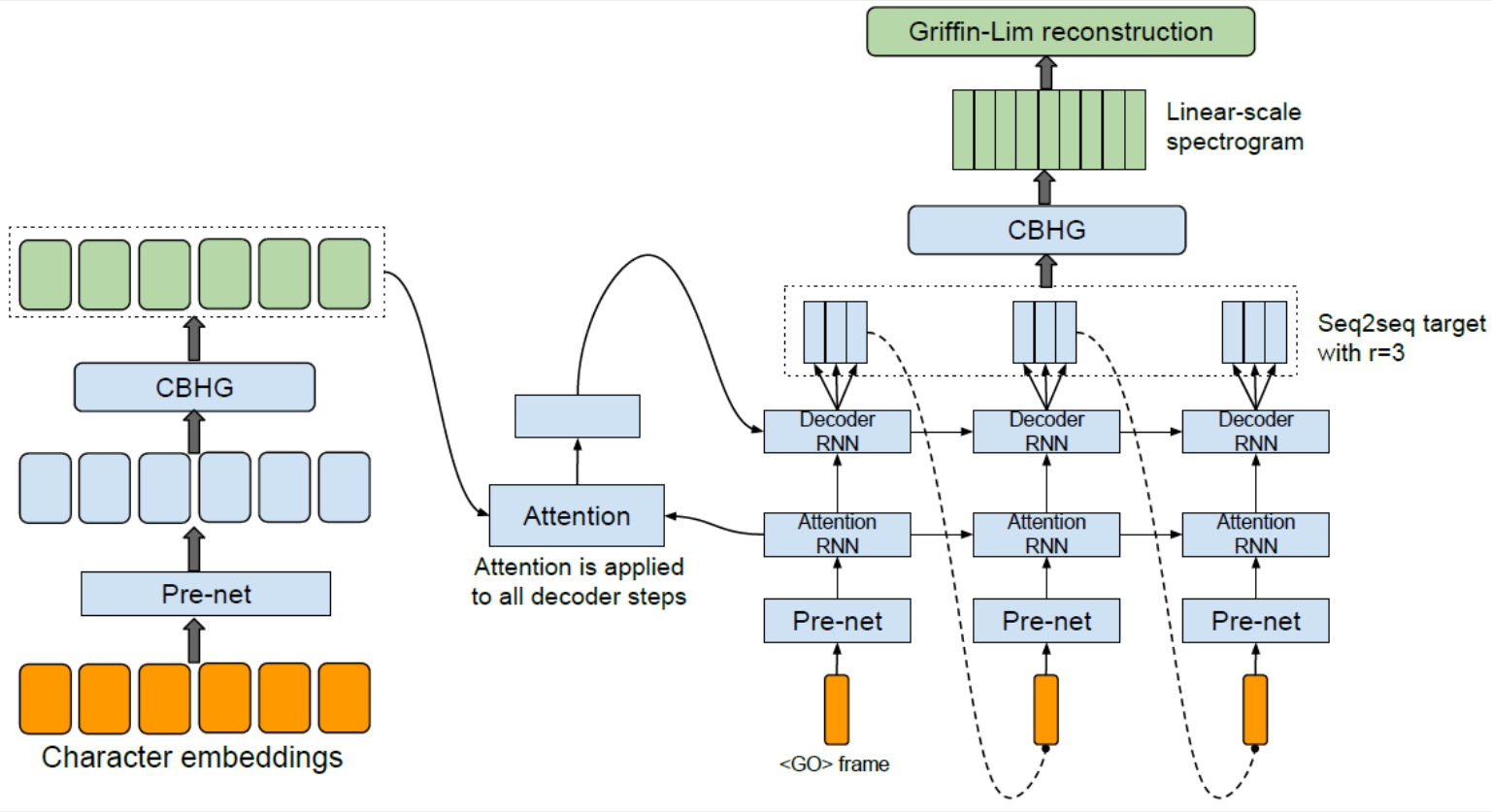


Figure 9.32 The input to the embedded training algorithm; a wavefile of spoken digits with a corresponding transcription. The transcription is converted into a raw HMM, ready to be aligned and trained against the cepstral features extracted from the wavefile.



- $\tilde{w} = \arg\max_{w} P(s|o) = \arg\max_{w} \boldsymbol{P(o|s)}P(s)$
- $\boldsymbol{P(o|s)} = P(s_{t_2}|s_{t_1})\boldsymbol{P(o_{t_1}|s_{t_1})}\cdots$
  - **GMM-HMM**: $\boldsymbol{P(o_{t_1}|s_{t_1})}$ is provided by Gaussian Mixture Model
  - **DNN-HMM**: $\boldsymbol{P(o_{t_1}|s_{t_1})} = \boldsymbol{P(s_{t_1}|o_{t_1})}P(o_{t_1})/P(s_{t_1})$ is calculated from DNN


Griffin-Lim reconstruction
Linear-scale spectrogram
CBHG
Seq2seq target with r=3
Decoder RNN
Decoder RNN
Decoder RNN
Attention RNN
Attention RNN
Attention RNN
Pre-net
Pre-net
Pre-net
<GO> frame
Attention
Attention is applied to all decoder steps
CBHG
Pre-net
Character embeddings