面向程序员的深度学习

胡键

关于我

胡键

- 上海圭步 CTO
- 国际 SCRUM 联盟认证 CSM
- 华为云 MVP
- TFUG 西安组织者（机器学习技术社区）
- 前 InfoQ 中文站 SOA 社区首席编辑
- 技术书籍翻译者
- 兴趣
  - 工业物联网
  - 区块链（以太坊、超级账本 Fabric、Polkadot / Substrate）
  - 机器学习
  - 大数据处理

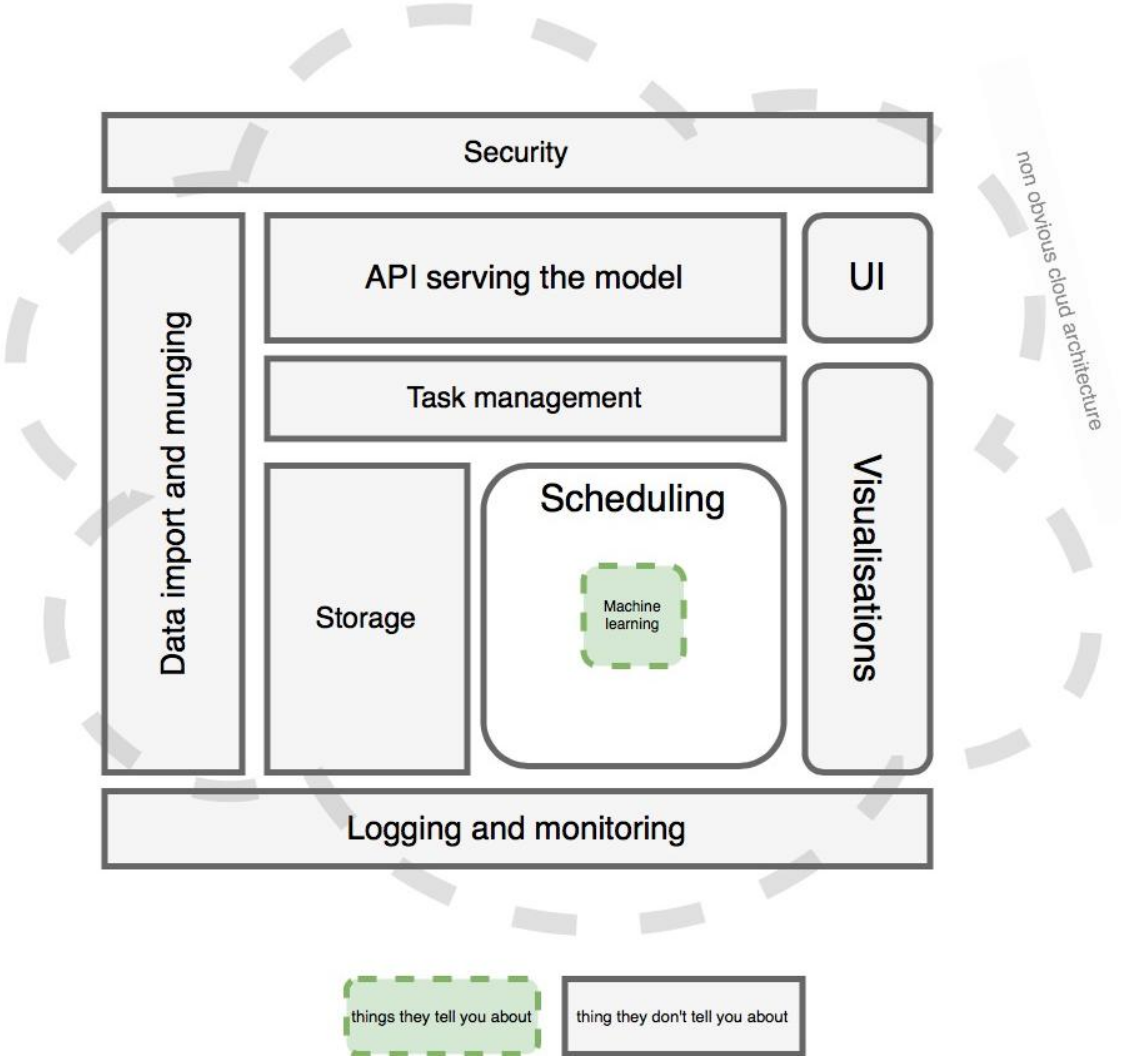
jian.hu@shifudao.com

目 录

CONTENTS

先从分工谈起

ML 工程师要搭建的系统



来源：https://medium.com/@tomaszdudek/but-what-is-this-machine-learning-engineer-actually-doing-18464d5c699

ML 工程师的技能：ML + 工程师的结合体

工程能力仍是首选

- 在产品环境实现模型（训练和推理）

具备 ML 知识

- 数据科学家支撑新模型

理解业务知识

- 领域专家支撑更专业的要求

来源：https://medium.com/@tomaszdudek/but-what-is-this-machine-learning-engineer-actually-doing-18464d5c699

普通工程师如何学习深度学习

纸上得来终觉浅，绝知此事要躬行

01

找感觉

- 例子驱动学习
- 先工具，后理论
- 手动实现NN概念模型

02

找队友

- Kaggle 练级，找同好
- 边练边学，缺啥补啥
- 社区输出

03

找方案

- 搜集模型、论文和案例
- 尝试阅读源码
- 扩展工具和框架

普通工程师如何学习深度学习

光看不练

01

- 结合工具实战练习

克服心理障碍

02

- Keras 和 TF 之类工具的诞生推动了深度学习的平民化。
- 就 ML 工程师而言，数学的要求并不太高。

神秘化 ML 类项目

03

- ML 项目就是用 ML 模型解决实际问题，与一般项目没有本质区别。

模型自己写

04

- 预训练模型本质上就是类库。

不重视数据

05

- 好数据才能得出好结果。

要避免的学习误区

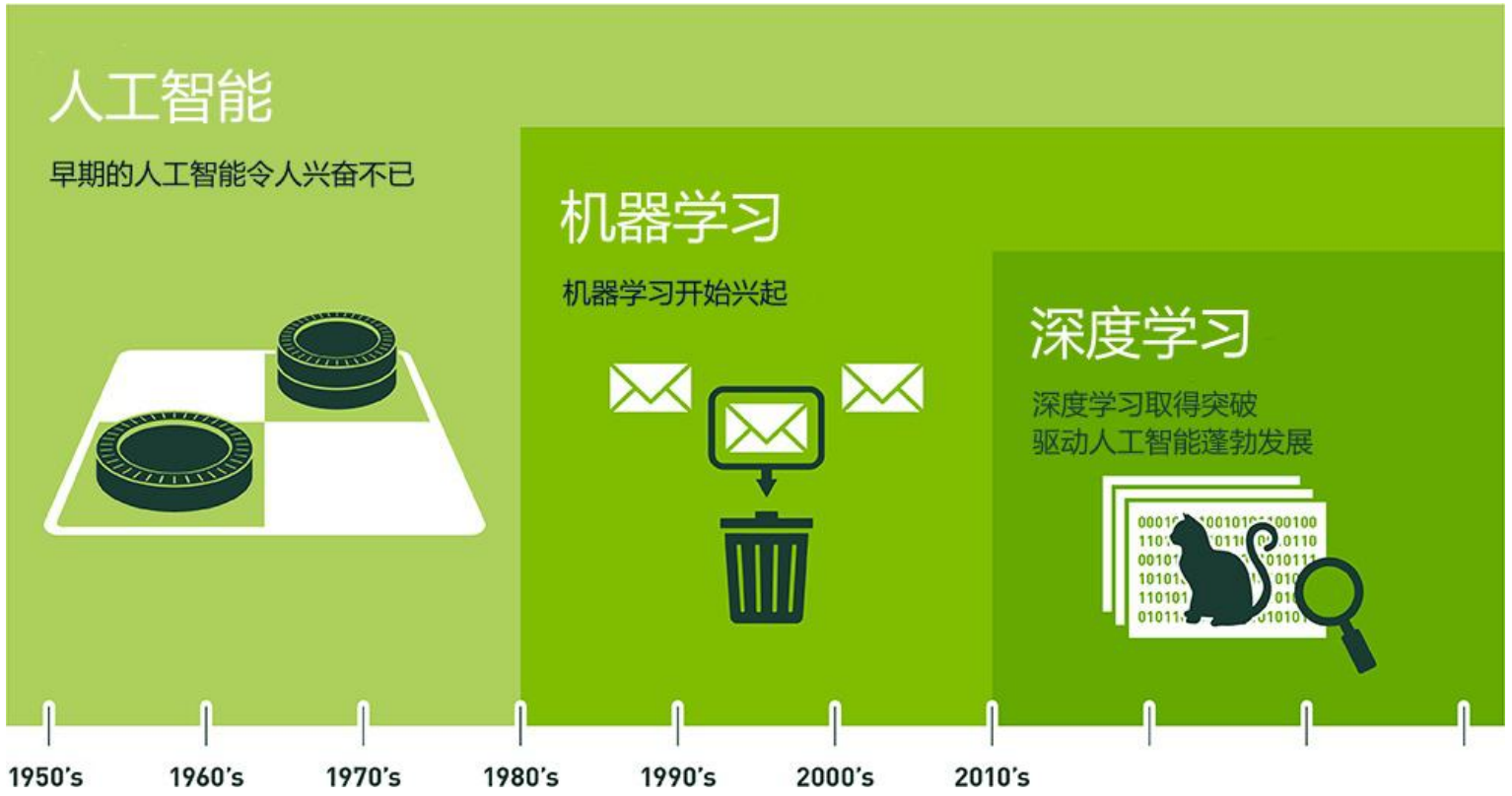
/02

深度学习基础概念

人工智能、机器学习和深度学习

三者的区别

- 人工智能，梦想构造复杂的、拥有与人类智慧同样本质特性的机器，一般指强人工智能。当前研究多半集中在弱人工智能部分。
- 机器学习，一种实现人工智能的方法，使用算法来解析数据、从中学习，然后对真实世界中的事件做出决策和预测。
- 深度学习，一种实现机器学习的技术。



促使当下人工智能繁荣的原因

天时：算力的发展

人工智能的发展伴随着个人电脑、互联网、移动互联网的成熟和兴起，在这个过程中算力呈增加趋势。充沛的算力为人工智能的研究提供了动力。

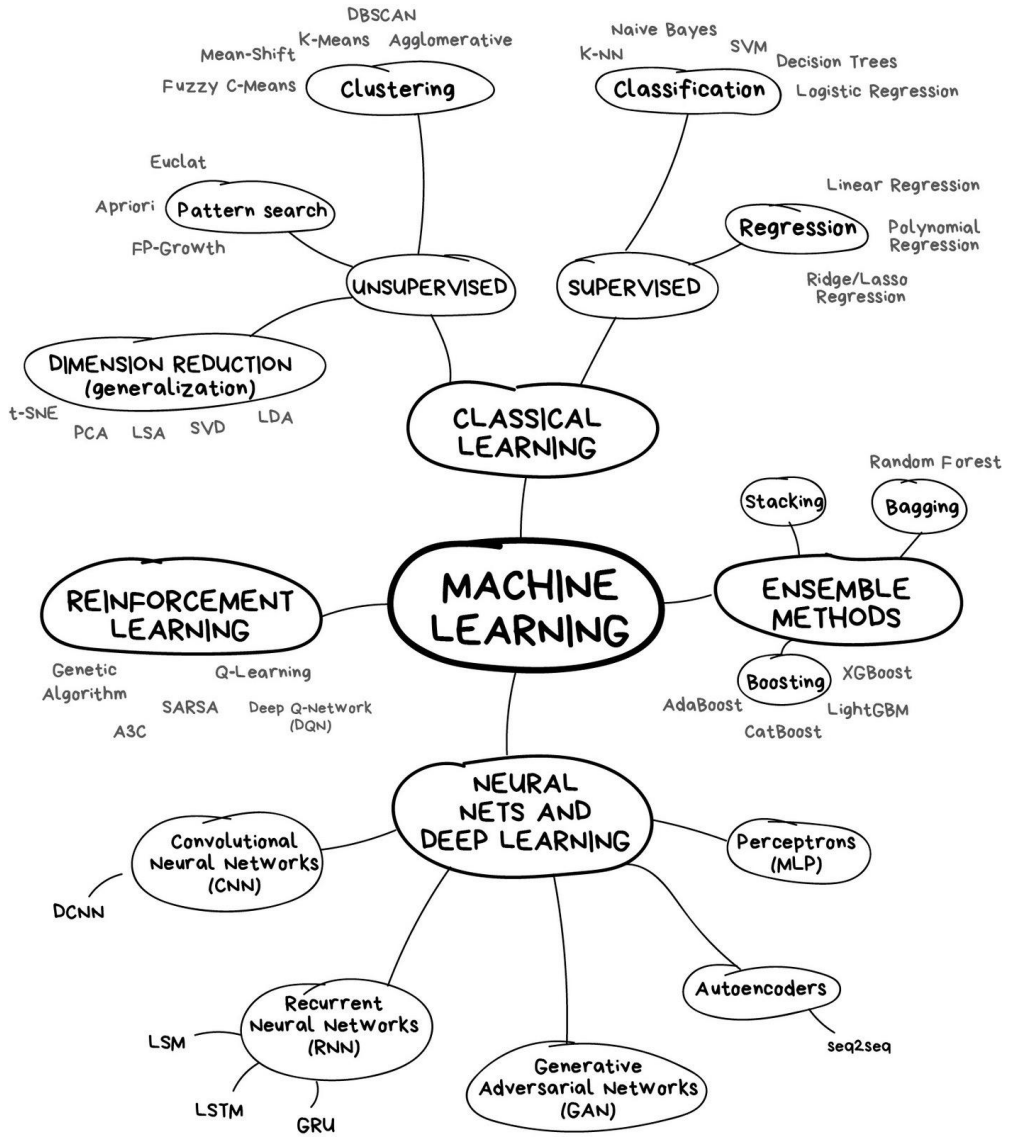
地利：日益增加的数据

联网设备和互联网用户的增加导致数据的增加，丰富的数据为人工智能的研究提供了必要的原材料。

人和：模型和算法的推陈出新

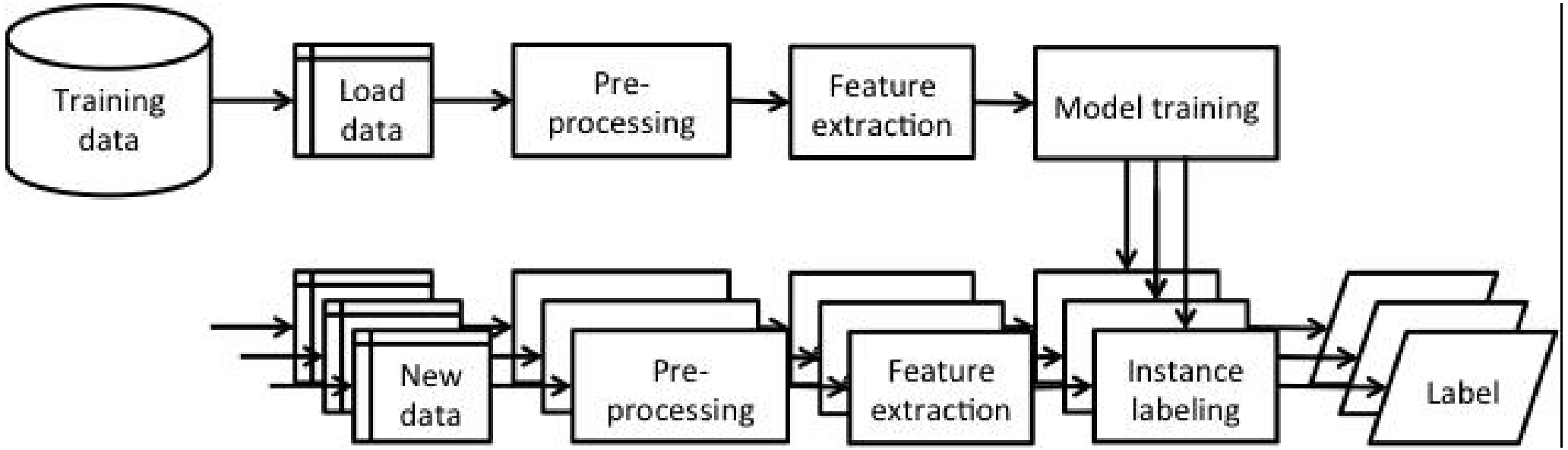
可以有效利用算力和数据的算法和模型被开发出来，它们为人工智能的研究提供了匹配的引擎。

机器学习的算法地图



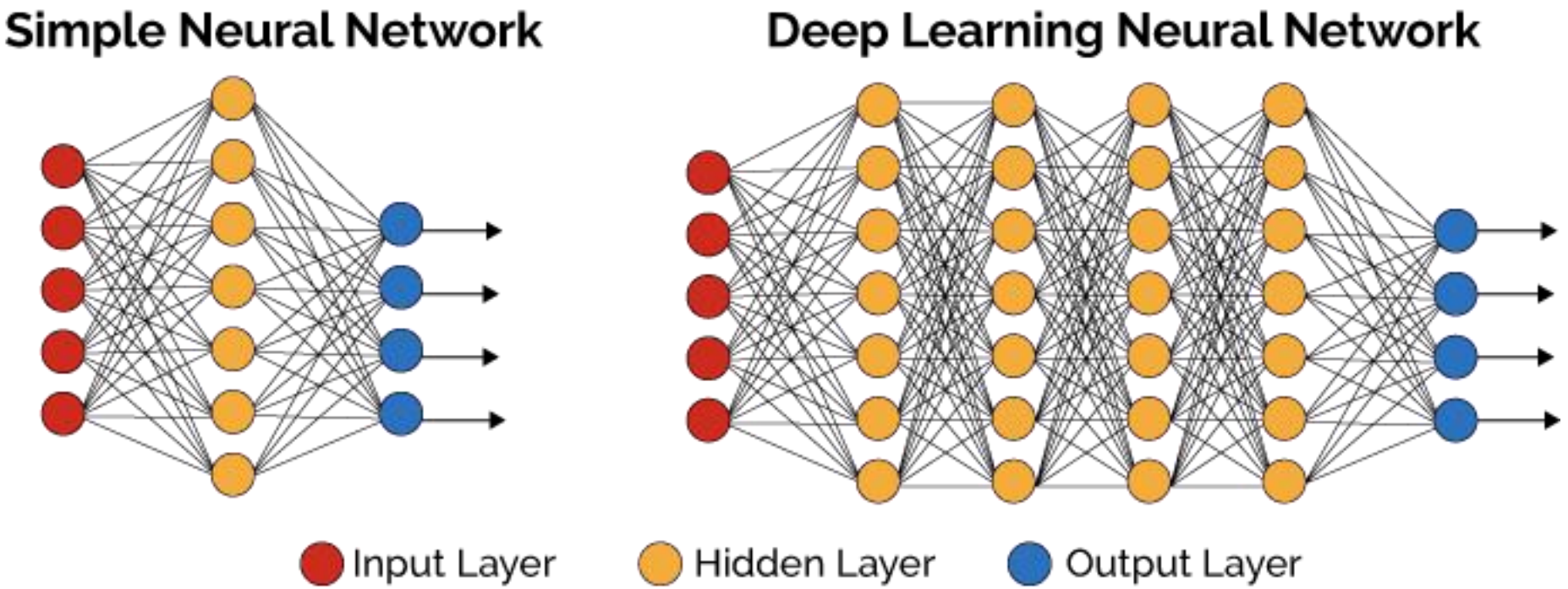
来源：https://vas3k.com/blog/machine_learning/

机器学习应用的工作流程



- 上方，训练。
- 下方，使用。注意：数据要经过同样的数据处理。
- 通过更新训练数据，得到新模型，然后替换旧的即可。

什么是深度网络？



- 传统浅层网络 <= 3层
- 深度网络：层数 > 3
- 理论上两层网络即可完成任何任务，引入多层的价值：
  - 计算的可行性
  - 关联的灵活性

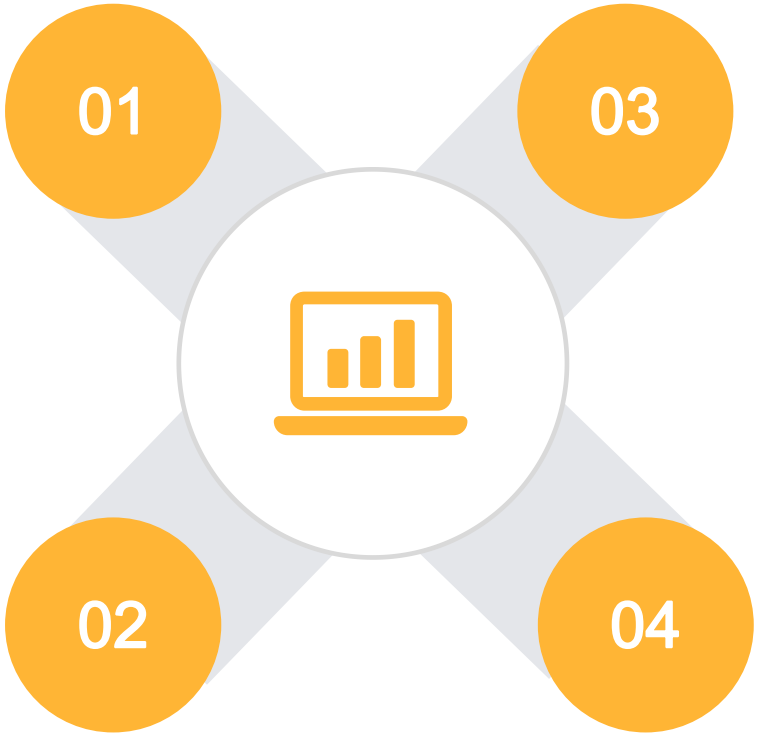
如何使用深度网络？

训练阶段

1. 构建网络
2. 训练数据数值化处理
3. 前向传播计算偏差
4. 后向传播调整权重和偏置
5. 重复3/4直到偏差不再缩小
6. 保持模型

训练目标

- 权重矩阵和偏置矩阵



推理阶段

1. 加载模型（网络 + 权重/偏置）
2. 转化输入符合训练时的输入格式
3. 调用模型预测结果
4. 解释结果

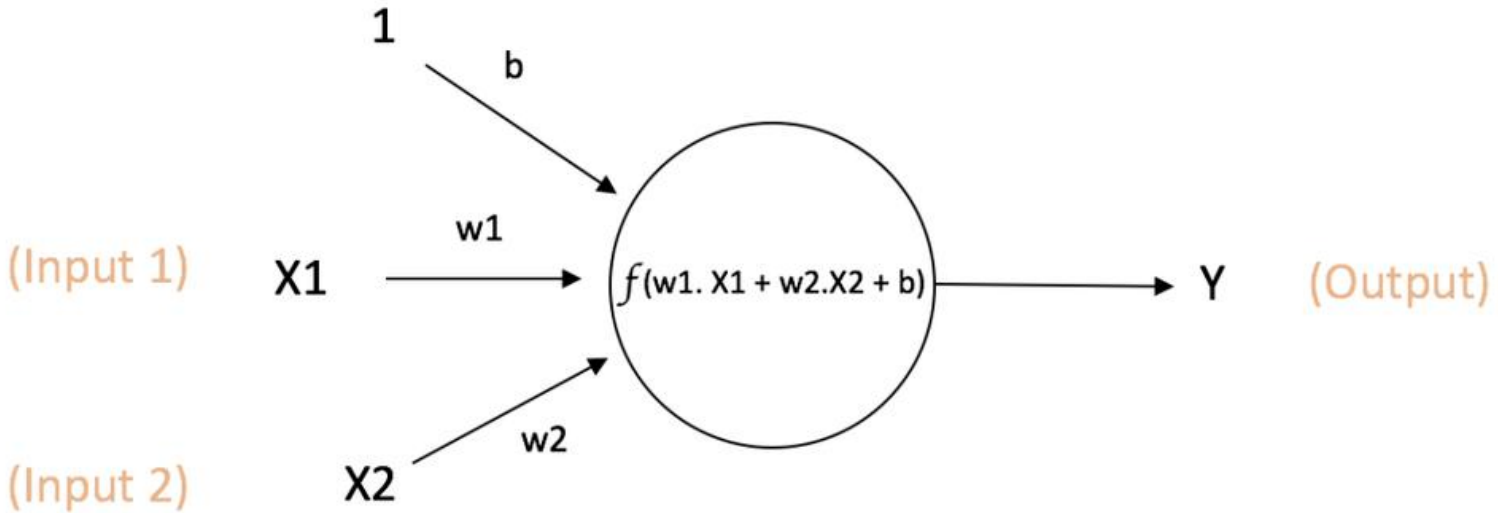
模型来源

- 训练结果
- （解决同类问题的）预训练模型

深度网络构成：神经元

剖析神经元

- X，输入，Y，输出
- W + b = 知识
- 激活函数（f），决定神经元输出
  - f((sum(x*w) + bias)
  - 判定当前神经元的重要程度



Output of neuron = Y= f(w1. X1 + w2.X2 + b)

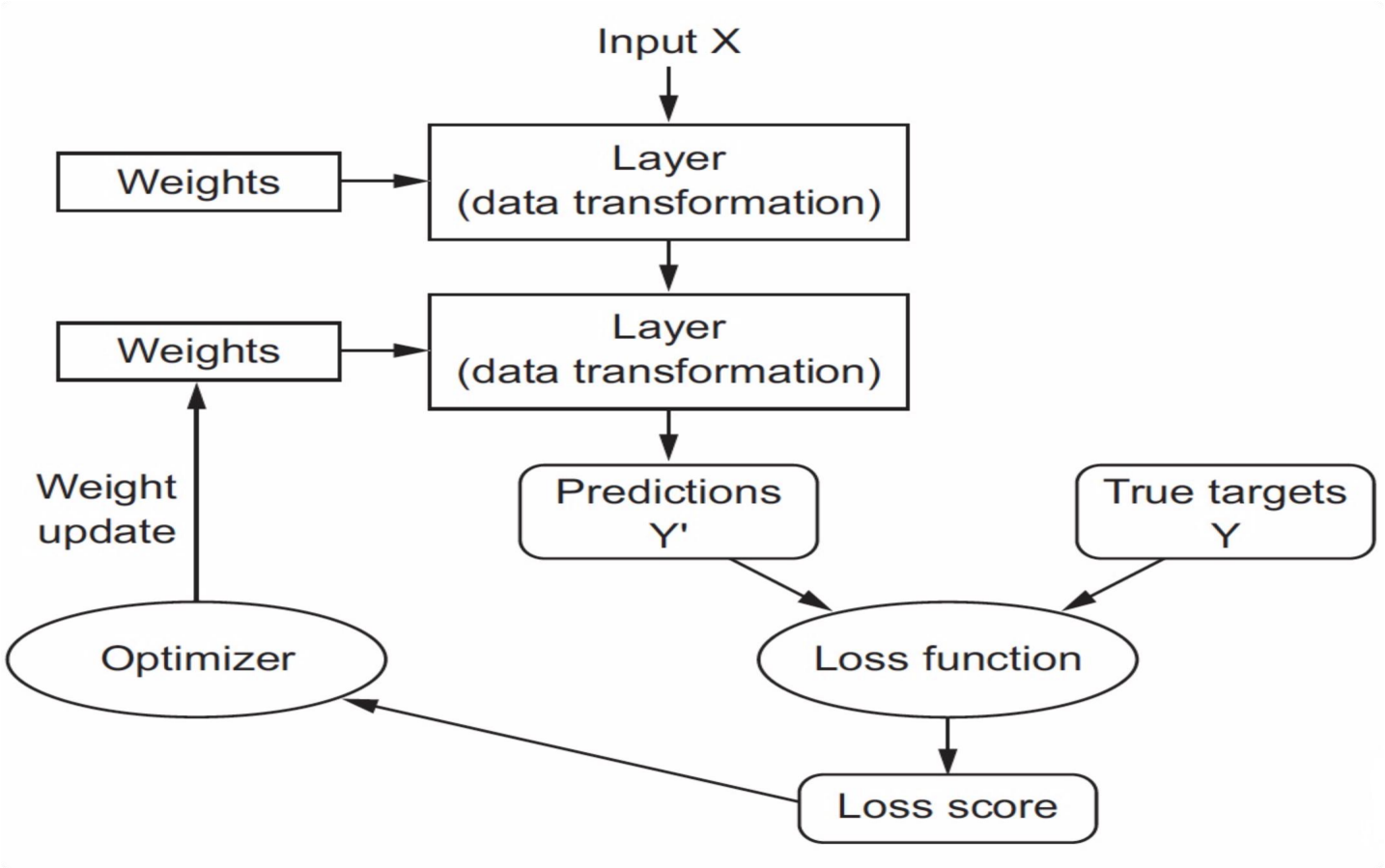
深度网络构成：层

- 层 = 一组神经元，可视为单个神经元的矩阵化
  - 输入（1 x m）、输出（1 x n）、W（m x n）、偏置（1 x n）
  - 输出 = 激活函数(输入 * 输出) + 偏置
- 将每个层视为数据的加工处理函数，其输出为中间数据集，并作为下一层的输入。
  - 层的组合形成了数据处理流水线，即网络结构。
- 对于层的激活函数
  - 隐藏层常用：sigmod、relu、tanh 等
  - 输出层根据问题类型决定：
    - 回归问题（任意值），无
    - 回归问题（0~1），sigmod
    - 分类问题（n选1），softmax
    - 分类问题（2选1），sigmod

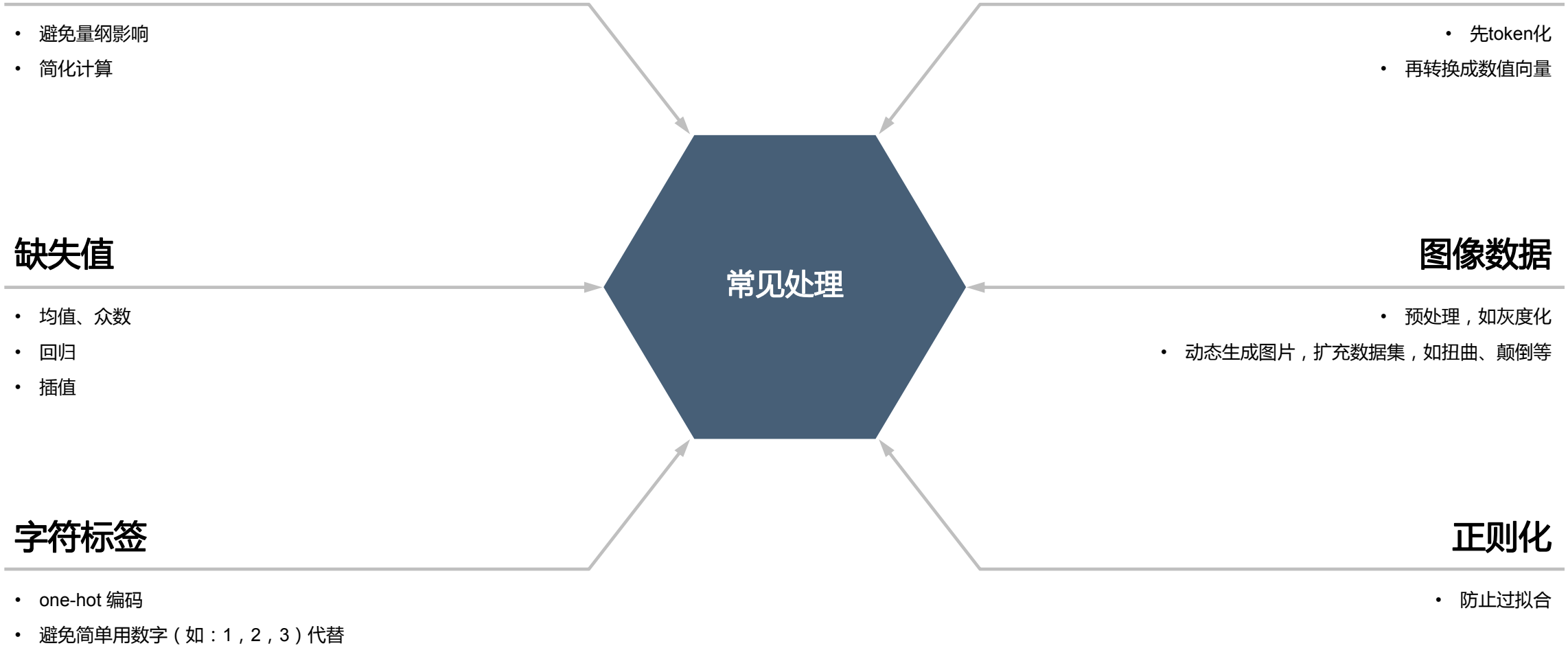
深度网络的训练

训练要点

- 损失函数，评判预测和实际的差距，不同类型的问题使用不同类型的损失函数。
  - 回归问题：MSE
  - 分类问题：Cross Entropy
- 优化器，调整权重和偏置，常见SGD。



训练用输入数据常见处理



训练数据集的划分

常用术语

- 1 pass，一条数据处理完毕
- 1 iteration，一批样本数据中所有数据处理完毕
- 1 epoch，所有批次所有样本数据处理完毕

训练模型需考虑

- batch 大小，权重更新时机
- epoch 大小，训练次数

数据集的划分

- 训练集，训练数据
- 验证集，训练过程中验证训练效果没有下降
- 测试集，训练完毕后检验训练效果

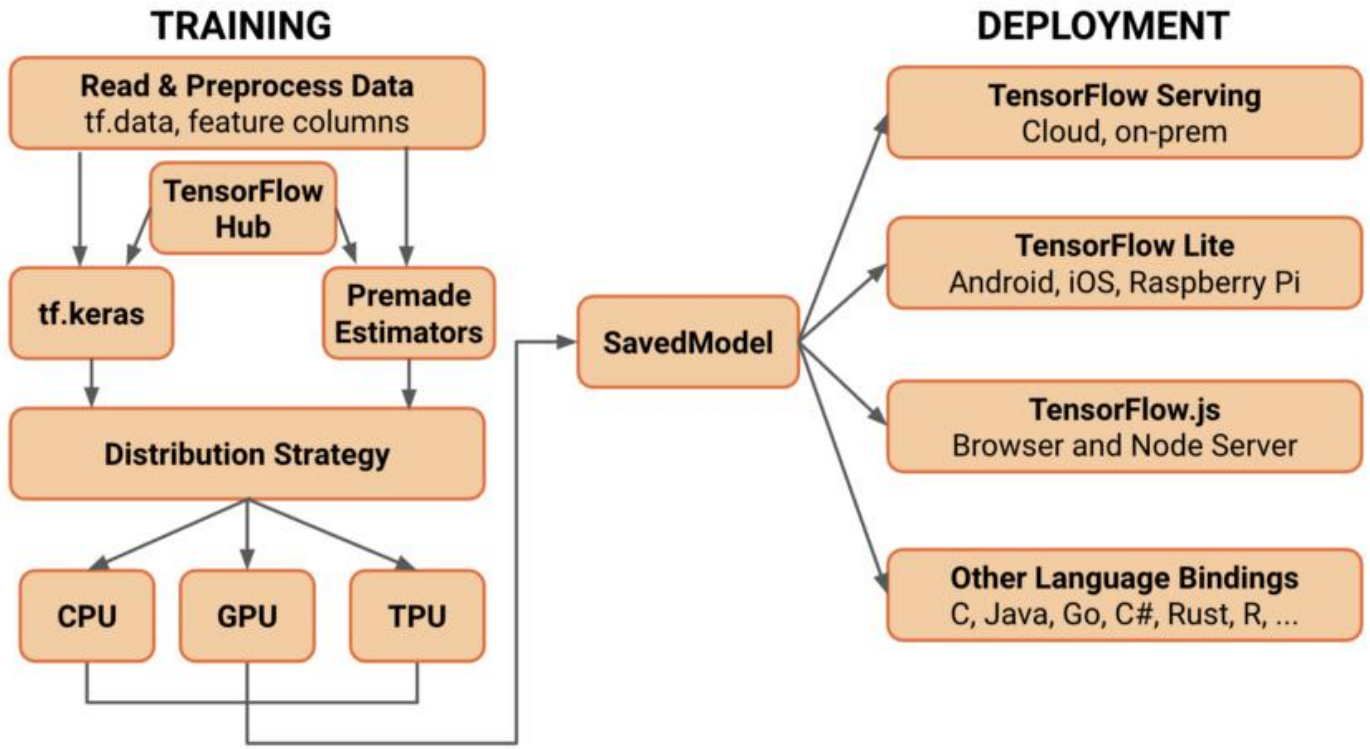
训练集、验证集、测试集常见比例

- 60:20:20
- 70:10:20
- 超大数据集：95:2:3

深度模型的部署

- 训练完的模型等同于类库
- 方式1：API
  - 如：Rest 框架 + 模型
- 方式2：嵌入应用
  - 如：MVC 框架 + 模型

- 注意事项
  - 输入数据需符合模型预期
  - 模型输出需应用自己解释
  - 训练和使用可以是不同框架，如在 DL4J 环境中使用 TF 训练的模型



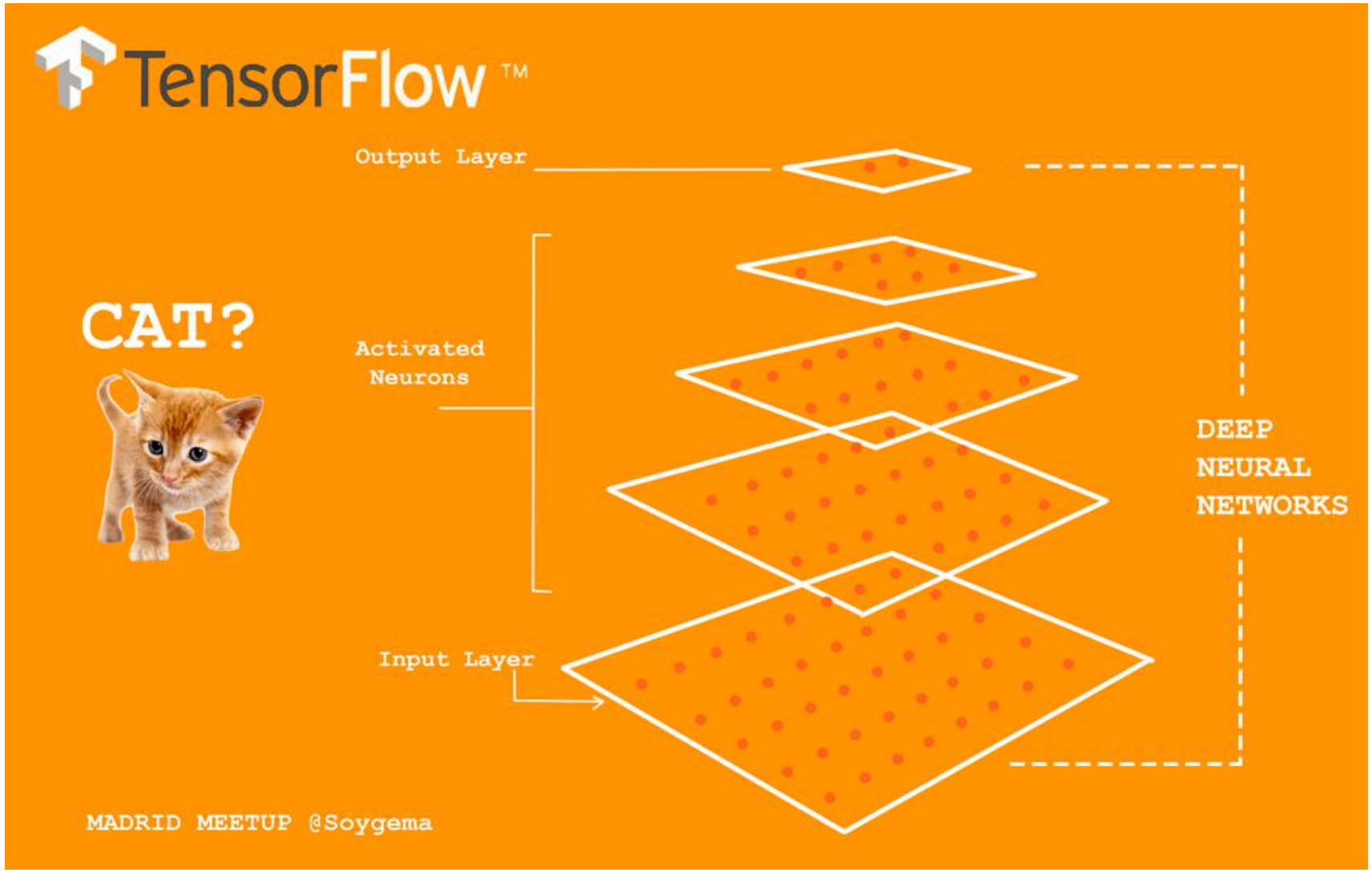
/03

深度学习进阶概念

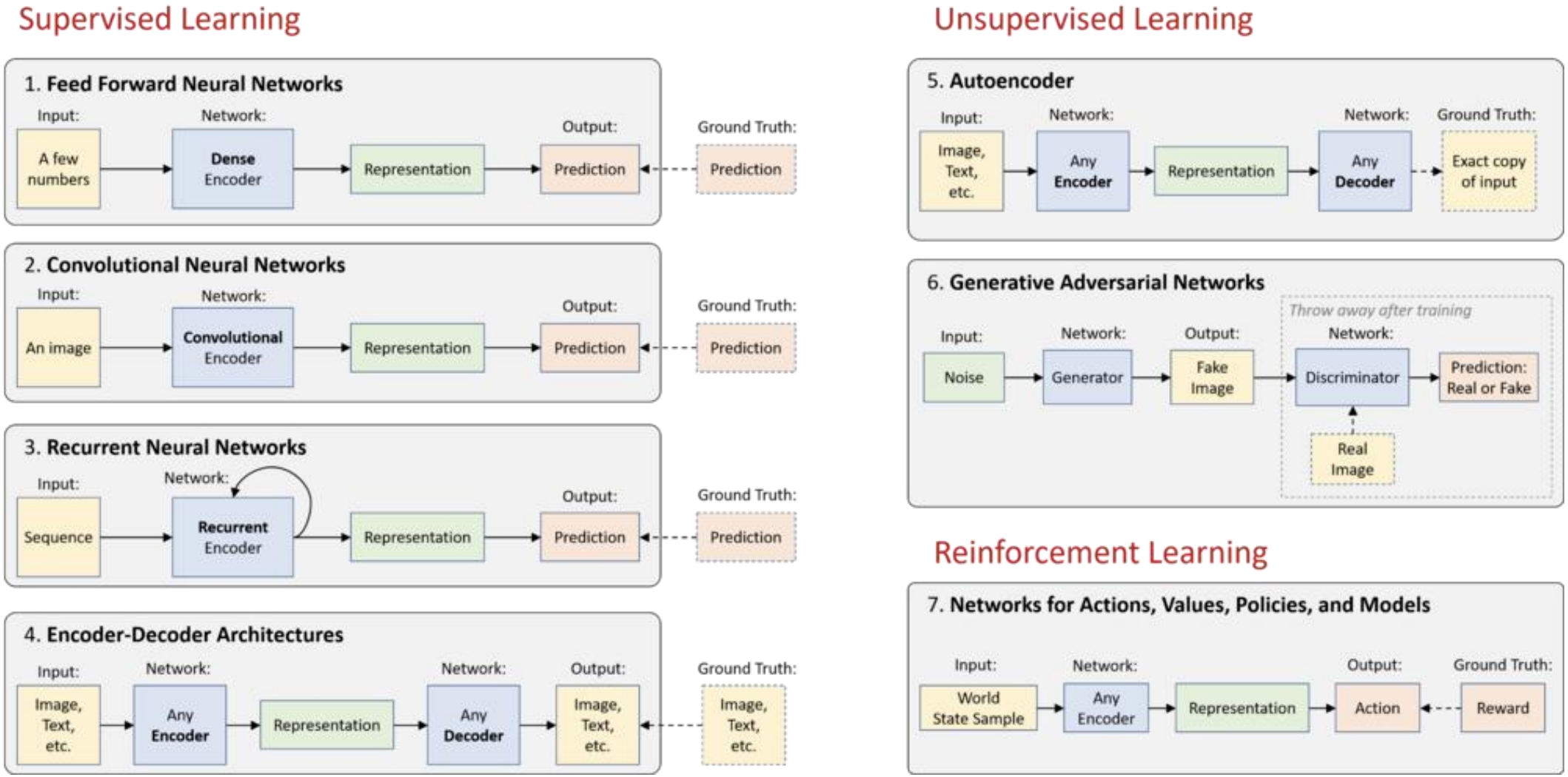
再谈“深度网络中的层”

- 不同类型的层完成不同类型的工作，典型：
  - Convolution，抽取局部特征
  - Pooling，跟 Convolution 配合使用，典型 MaxPooling 、AvgPooling 等
  - DropOut，随机去掉一定比例神经元
  - Dense，全连接
  - Embedding，词向量转换
  - Recurrent，处理序列数据，如文本和时序
  - Merge，合并多个输入层数据
  - ……
- 构建网络的过程就是使用不同层进行组合的过程
  - 本质上等同于编程语句的使用构成程序
- 一般情况下，输出层都为 Dense

深度神经网络是如何看到猫的？



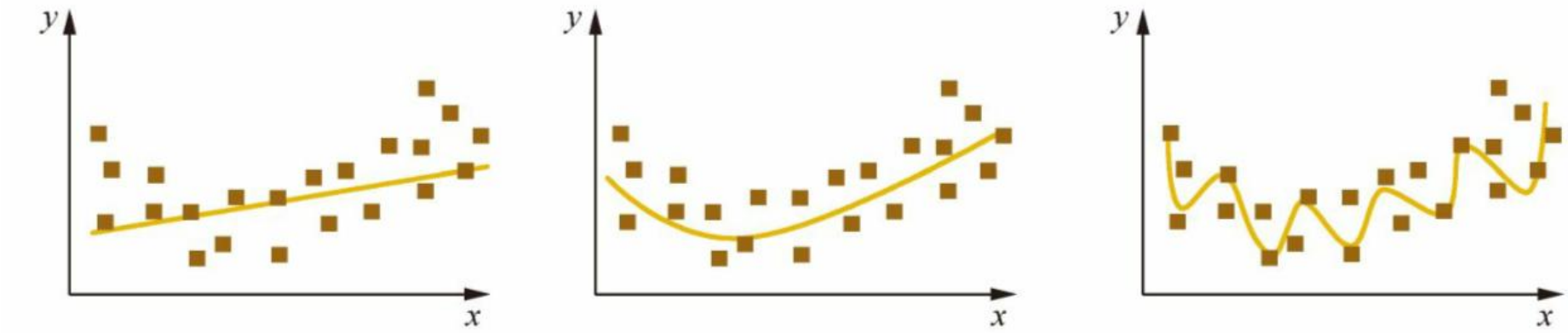
常见的网络结构


Supervised Learning
1. Feed Forward Neural Networks
Input:
A few numbers
Network:
Dense Encoder
Representation
Output:
Prediction
Ground Truth:
Prediction
2. Convolutional Neural Networks
Input:
An image
Network:
Convolutional Encoder
Representation
Output:
Prediction
Ground Truth:
Prediction
3. Recurrent Neural Networks
Input:
Sequence
Network:
Recurrent Encoder
Representation
Output:
Prediction
Ground Truth:
Prediction
4. Encoder-Decoder Architectures
Input:
Image, Text, etc.
Network:
Any Encoder
Representation
Network:
Any Decoder
Output:
Image, Text, etc.
Ground Truth:
Image, Text, etc.
Unsupervised Learning
5. Autoencoder
Input:
Image, Text, etc.
Network:
Any Encoder
Representation
Network:
Any Decoder
Ground Truth:
Exact copy of input
6. Generative Adversarial Networks
Throw away after training
Input:
Noise
Network:
Generator
Output:
Fake Image
Network:
Discriminator
Prediction: Real or Fake
Real Image
Reinforcement Learning
7. Networks for Actions, Values, Policies, and Models
Input:
World State Sample
Network:
Any Encoder
Representation
Output:
Action
Ground Truth:
Reward

欠拟合和过拟合

- 欠拟合
  - 学习能力低下，即使对训练数据也都表现很差
  - 典型原因
    - 小网络、大数据
  - 对策
    - 增加每层神经元
    - 增加网络层级

- 过拟合
  - 死记硬背，对训练数据几乎能达到100%，而对测试数据，则表现不如人意
  - 典型原因
    - 大网络、小数据
  - 对策
    - 数据正则化：L1和L2
    - DropOut



（a）欠拟合　（b）正常模型　（c）过拟合

优化过程中的鞍点

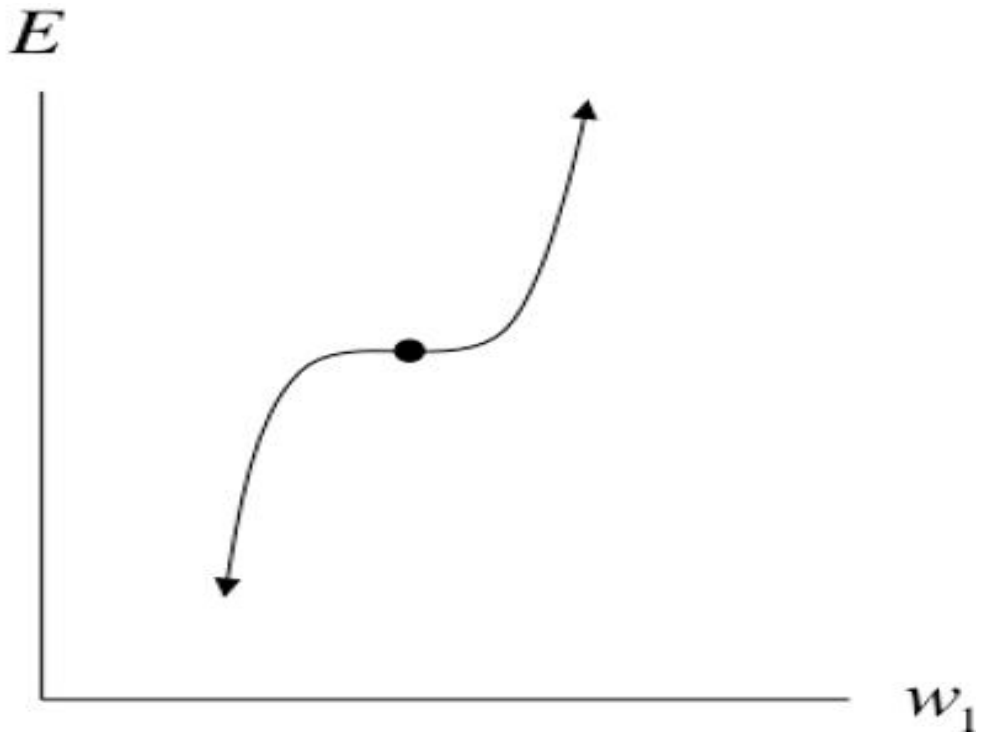
鞍点 = 局部最小值

负面影响

- 学习性能低下，迟迟无法收敛
- 无法获得理想结果

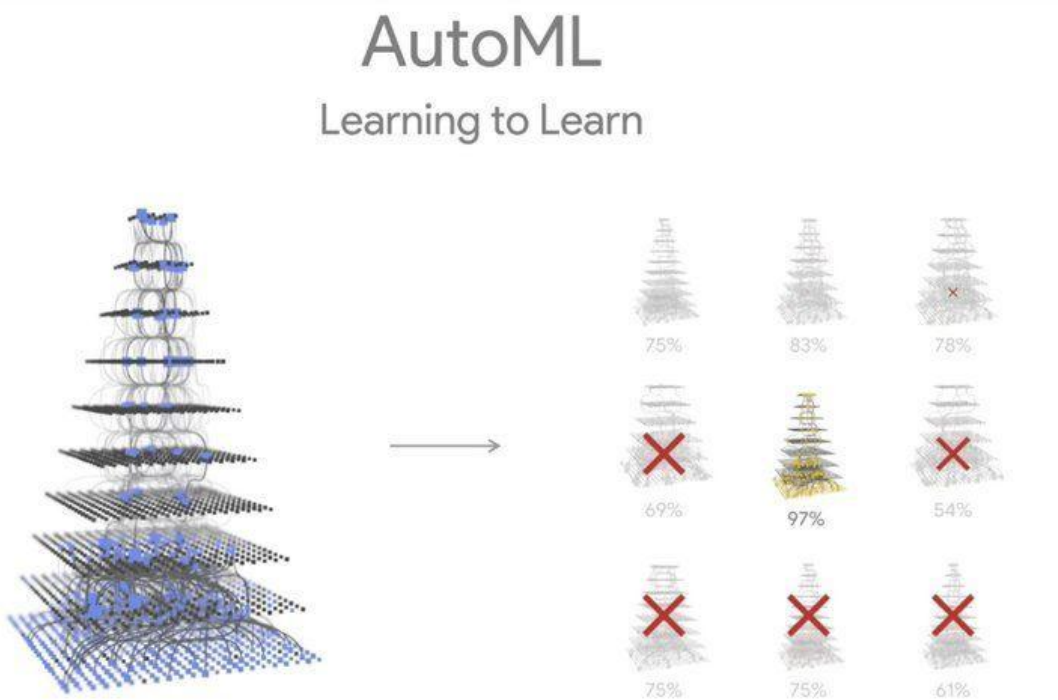
对策

- 使用有动量的优化算法，如 Adagrad、RMSProp。
- 自适应学习率
- 学习率决定了权重调整的大小



深度网络的超参数

- 超参数即元数据，无法训练获得。典型有：
  - 神经元个数
  - 网络层数
  - 激活函数
  - 优化器
  - 学习率
  - Epoch数
  - Batch大小
- 超参数调优
  - 人工、Grid Search、随机、贝叶斯。



/04

深度学习实战

环境搭建

- 设置 pip 代理：~/.pip/pip.conf
  - [global]
  - index-url = https://pypi.douban.com/simple/
- 安装 Mini-Conda
  - 下载并安装
- 安装TF环境
  - conda create -n tfenv python=3.6.x
  - source activate tfenv
  - Pip 安装：numpy、pandas、scipy、pillow、h5py、matplotlib、scikit-learn、keras、tensorflow
    - 对于matplotlib，可能需要
      - 创建：~/.matplotlib/matplotlibrc
      - 内容：backend: TkAgg

IDE 配置：vscode

之前，我都会推荐用pyCharm来作为Python编辑器，但现在更偏爱更轻量的vscode：

1. 安装微软官方的Python插件
2. 选择conda为python解释器：
    - 进入【设置】页面，选择【用户设置】，选择【扩展】-【Python】
    - 找到【Conda Path】，将它设置为conda所在位置，如：/Users/foxgem/miniconda3/bin/conda
3. 为了能在编辑文件时可以享受到自动联想，需要配置当前用的Python解释器使用conda env（这一步很关键，否则你在自己环境中安装的那些包如keras无法在编辑时被联想到）：
    - 参考：https://code.visualstudio.com/docs/python/environments
    - 如果不想看上面的长篇大论，这里介绍一下最简单的方式：在文件左下角状态栏点击【解释器】，然后选择合适的解释器就好了。
4. 配置代码格式化插件
    - 输入：shift + cmd + f，此时会提示安装格式化插件
    - 选择用pip安装，它会打开终端激活环境，在当前环境下使用pip install你选择的格式化包（一般用yapf）。

Minist 例子 TF 2.0 版：加载并训练模型

In [30]:
```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf
```

In [31]:
```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

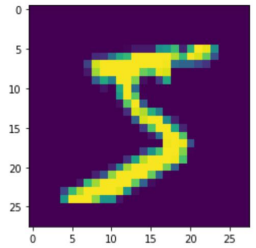
In [32]:
```
x_train.shape
```

Out[32]:
```
(60000, 28, 28)
```

In [33]:
```
%matplotlib inline
import matplotlib.pyplot as plt

print('This first picture in training set is %x' % y_train[0])
plt.figure()
plt.imshow(x_train[0])
plt.show()
```

```
This first picture in training set is 5
```



In [34]:
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [35]:
```
model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test,  y_test, verbose=2)
```

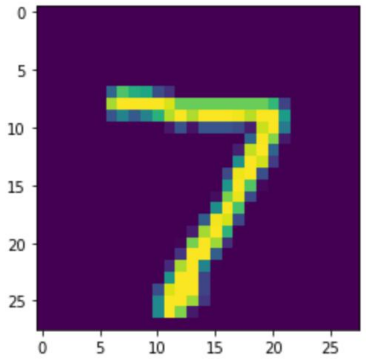
```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 2s 41us/sample - loss: 0.2968 - accuracy: 0.9133
Epoch 2/5
60000/60000 [==============================] - 2s 39us/sample - loss: 0.1445 - accuracy: 0.9567
Epoch 3/5
60000/60000 [==============================] - 2s 37us/sample - loss: 0.1058 - accuracy: 0.9676
Epoch 4/5
60000/60000 [==============================] - 2s 41us/sample - loss: 0.0875 - accuracy: 0.9729
Epoch 5/5
60000/60000 [==============================] - 2s 39us/sample - loss: 0.0736 - accuracy: 0.9764
10000/1 - 0s - loss: 0.0448 - accuracy: 0.9787
```

Out[35]:
```
[0.07673446817761287, 0.9787]
```

Minist 例子 TF 2.0 版：模型预测

In [36]:
```
print('This predicted value of first picture in test set is %x' % model.predict_classes(x_test[:1])[0])
print('This real value of first picture in test set is %x' % y_test[0])
plt.figure()
plt.imshow(x_test[0])
plt.show()
```

```
This predicted value of first picture in test set is 7
This real value of first picture in test set is 7
```



In [37]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               100480
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

Minist 例子 TF 2.0 版：保存并重新加载模型

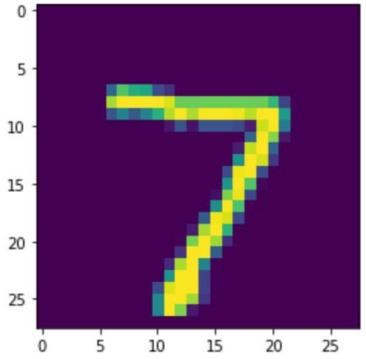
In [38]:
```
model.save('minist.h5')
```

In [39]:
```
saved_model = tf.keras.models.load_model('minist.h5')
saved_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               100480
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
print('This predicted value of first picture in test set is %x' % saved_model.predict_classes(x_test[:1])[0])
print('This real value of first picture in test set is %x' % y_test[0])
plt.figure()
plt.imshow(x_test[0])
plt.show()
```

```
This predicted value of first picture in test set is 7
This real value of first picture in test set is 7
```

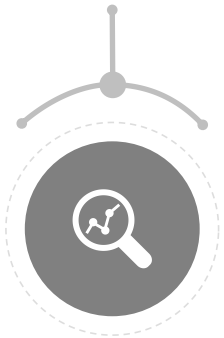


/05

总结

总结

深度学习入门并没有想象中那么困难

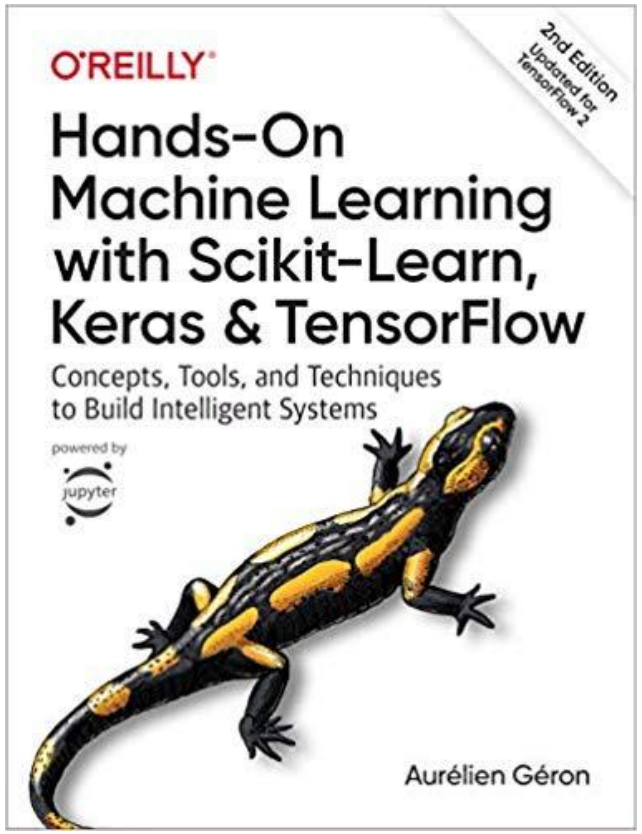


借助 Keras 等工具可以快速实现网络模型

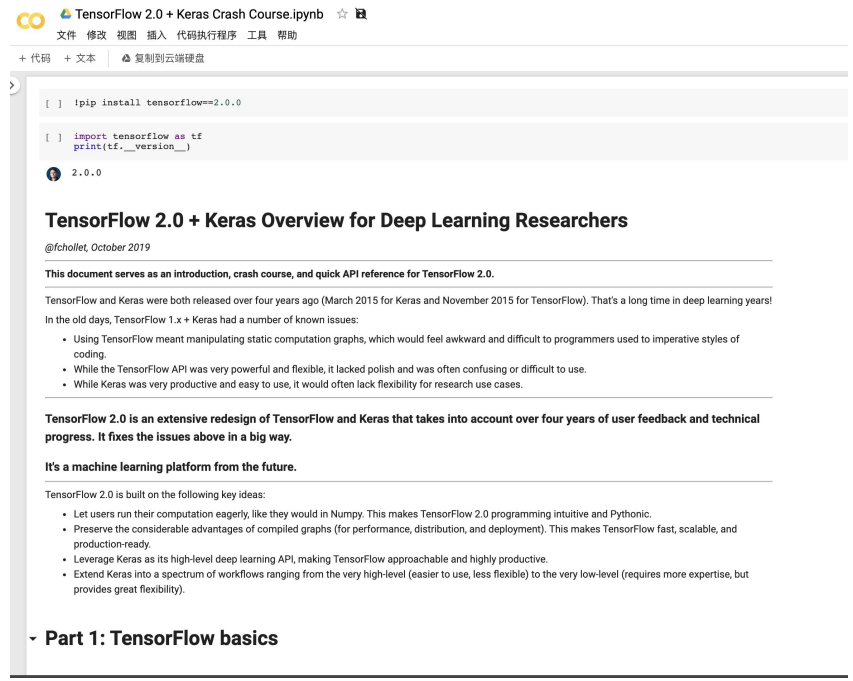
深度网络的调参虽是个脏活累活，但 AutoML 和 AutoKeras 这类工具的出现，意味着自动调参时代即将到来。

Make Your Hands Dirty

推荐资料


O'REILLY
2nd Edition
Updated for TensorFlow 2
Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by
jupyter
Aurélien Géron


TURING
MANNING
全彩印刷
Deep Learning with Python
Python
深度学习
[美] 弗朗索瓦·肖莱 著
张亮 译
Keras之父、Google人工智能研究员
François Chollet执笔，深度学习领域力作
通俗易懂，帮助读者建立关于机器学习和深度学习核心思想的直觉
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS


TensorFlow 2.0 + Keras Crash Course.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助
+ 代码 + 文本 复制到云端硬盘
[ ] !pip install tensorflow==2.0.0
[ ] import tensorflow as tf
print(tf.__version__)
2.0.0
TensorFlow 2.0 + Keras Overview for Deep Learning Researchers
@fchollet, October 2019
This document serves as an introduction, crash course, and quick API reference for TensorFlow 2.0.
TensorFlow and Keras were both released over four years ago (March 2015 for Keras and November 2015 for TensorFlow). That's a long time in deep learning years!
In the old days, TensorFlow 1.x + Keras had a number of known issues:
Using TensorFlow meant manipulating static computation graphs, which would feel awkward and difficult to programmers used to imperative styles of coding.
While the TensorFlow API was very powerful and flexible, it lacked polish and was often confusing or difficult to use.
While Keras was very productive and easy to use, it would often lack flexibility for research use cases.
TensorFlow 2.0 is an extensive redesign of TensorFlow and Keras that takes into account over four years of user feedback and technical progress. It fixes the issues above in a big way.
It's a machine learning platform from the future.
TensorFlow 2.0 is built on the following key ideas:
Let users run their computation eagerly, like they would in Numpy. This makes TensorFlow 2.0 programming intuitive and Pythonic.
Preserve the considerable advantages of compiled graphs (for performance, distribution, and deployment). This makes TensorFlow fast, scalable, and production-ready.
Leverage Keras as its high-level deep learning API, making TensorFlow approachable and highly productive.
Extend Keras into a spectrum of workflows ranging from the very high-level (easier to use, less flexible) to the very low-level (requires more expertise, but provides great flexibility).
Part 1: TensorFlow basics

THANKS

胡键 / 上海圭步 CTO

jian.hu@shifudao.com

个人微信

公众号