

第 8 章 授人以鱼不如授人以渔： 迁移学习

教材： 王万良 《人工智能导论》（第4版）

<https://www.icourse163.org/course/ZJUT-1002694018>

社区资源： <https://github.com/Microsoft/ai-edu>

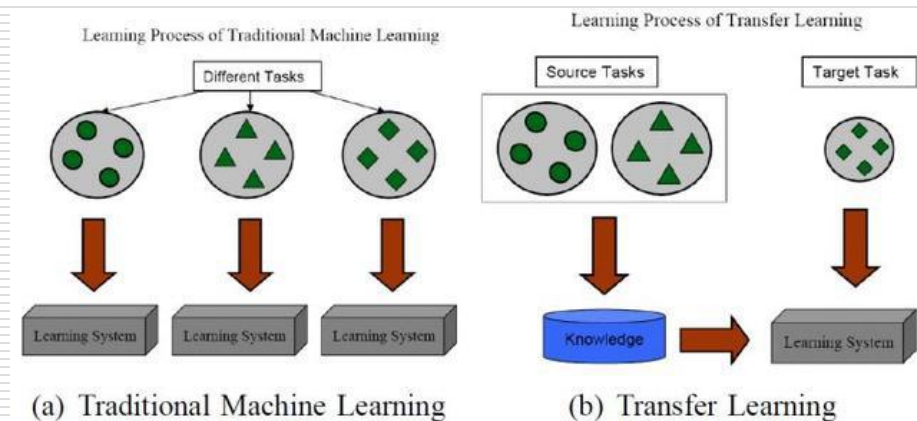
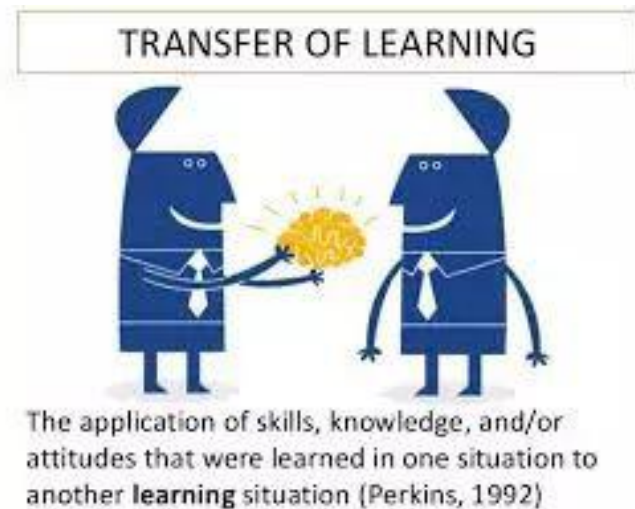
参考MOOC：

授人以鱼不如授人以渔：迁移学习

- 什么是迁移学习？
- 迁移学习分类
- 热门研究方向

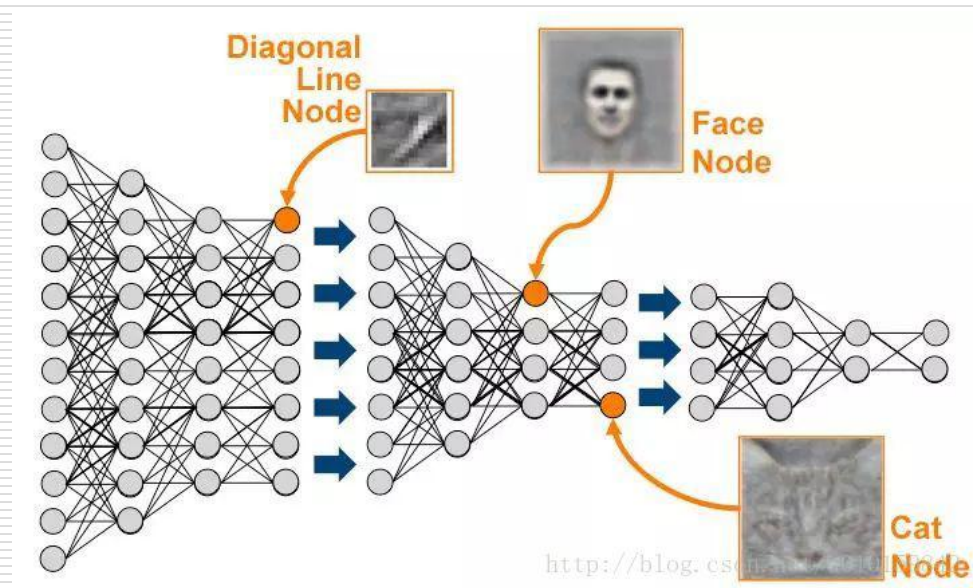
授人以鱼不如授人以渔：迁移学习

- 举一反三
- 运用已学习的知识来求解不同但相关领域问题的新的机器学习方法，目的是让机器“学会学习”
- 传统机器学习对不同的学习任务需要建立不同的模型，学习不同的参数，而对于迁移学习，只需要利用源域中的数据将知识迁移到目标域，就能完成模型建立



授人以鱼不如授人以渔：迁移学习

- 一层层网络中每个节点的权重从一个训练好的网络迁移到一个全新的网络里，而不是从头开始，为每个特定的任务训练一个神经网络
- 假设已经有了一个可以高精度分辨猫和狗的深度神经网络，之后想训练一个能够分辨不同品种的狗的图片模型，需要做的不是从头训练那些用来分辨直线，锐角的神经网络的前几层，而是利用训练好的网络，提取初级特征，之后只训练最后几层神经元，让其可以分辨狗的品种





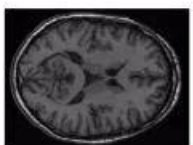


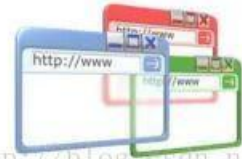
授人以鱼不如授人以渔：迁移学习

□ 不必重新发明轮子

Why?

<http://www.bigr.nl/website/structure/main.php?page=researchlines&subpage=project&id=64>

<http://www.spear.com.hk/Translation-company-Directory.html>

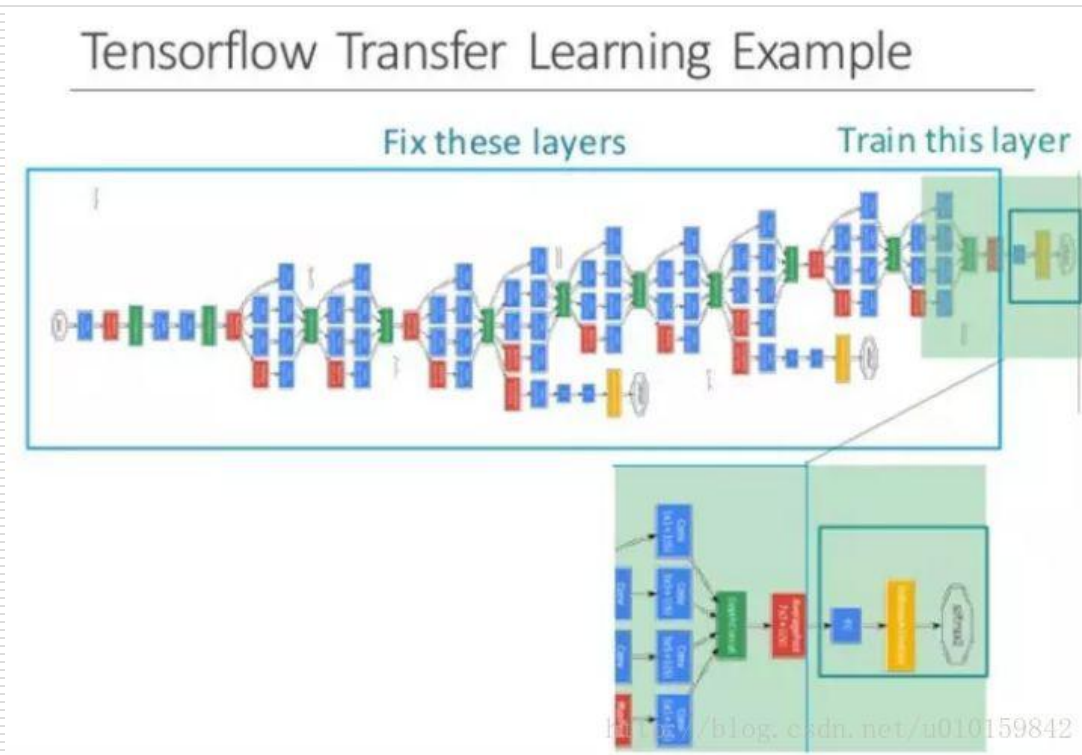
Task Considered		Data not directly related
Speech Recognition	 Taiwanese	 English Chinese
Image Recognition	 Medical Images	
Text Analysis	 Specific domain	 Webpages

<http://blog.csdn.net/u010159842>

授人以鱼不如授人以渔：迁移学习

□ 图像识别

- 训练一个神经网络，来识别不同的品种的猫
 - 若是从头开始训练，需要百万级的带标注数据，海量的显卡资源。
 - 若使用迁移学习，可以使用Google发布的Inception或VGG16这样成熟的物品分类的网络，只训练最后的softmax层，只需要几千张图片，使用普通的CPU就能完成，而且模型的准确性不差。



授人以鱼不如授人以渔：迁移学习

□ 两个假设

- 训练样本和测试数据必须处于相同的特征空间并具有相同的分布
 - 时间差异导致分布规律变化
- 有足够的高质量训练样本作为学习资源
 - 费时费力，知识产权

授人以鱼不如授人以渔：迁移学习

□ 为什么需要进行迁移学习？

- 数据的标签很难获取，当有些任务的数据标签很难获取时，就可以通过其他容易获取标签且和该任务相似的任务来迁移学习。
- 从头建立模型是复杂和耗时的，也即是需要通过迁移学习来加快学习效率。

授人以鱼不如授人以渔：迁移学习

□ 迁移学习的必要性和价值体现在

- 复用现有知识域数据，已有的大量工作不至于完全丢弃；
- 不需要再去花费巨大代价去重新采集和标定庞大的新数据集，也有可能数据根本无法获取；
- 对于快速出现的新领域，能够快速迁移和应用，体现时效性优势

授人以鱼不如授人以渔：迁移学习

□ 迁移学习能解决那些问题？

■ 小数据的问题

- 比方说新开一个网店，卖一种新的糕点，没有任何的数据，就无法建立模型对用户进行推荐。
- 但用户卖饮料，已经有了很多很多的数据，利用这些数据建一个模型，结合用户买饮料的习惯和买糕点的习惯的关联，就可以把饮料的推荐模型给成功地迁移到糕点的领域，这样，在数据不多的情况下可以成功推荐一些用户可能喜欢的糕点

■ 个性化的问题

- 比如每个人都希望自己的手机能够记住一些习惯，这样不用每次都去设定它
- 可以通过迁移学习把一个通用的用户使用手机的模型迁移到个性化的数据上面。

授人以鱼不如授人以渔：迁移学习

□ 相关定义

- 源域：已有的知识（包括样本数据集及其分布）
- 目标域：要学习的新知识
- 源任务
- 目标任务

□ 严格定义

- 给定源域 $D_s = \{X_s, F_s(X)\}$ 和学习任务 T_s ，目标域 $D_t = \{X_t, F_t(X)\}$ 和学习任务 T_t ，迁移学习旨在源域不同于目标域或学习任务 T_t 不同于学习任务 T_s 的条件下通过使用学习任务 T_s 和源域 $D_s = \{X_s, F_s(X)\}$ 所获取的知识来帮助学习目标的在目标域 D_t 的预测函数 $F_t(\cdot)$

授人以鱼不如授人以渔：迁移学习

迁移学习的分类

按迁移情景分

- 归纳式迁移学习（Inductive TL）：源域和目标域的学习任务不同
- 直推式迁移学习（Transductive TL):源域和目标域不同，学习任务相同
- 无监督迁移学习（Unsupervised TL):源域和目标域均没有标签

		源Domain & 目标Domain	源Task & 目标Task	源Data & 目标Data	任务方法
传统机器学习		相同	相同	有标签 有标签	
迁移学习	归纳式迁移学习	相同/相关	相关	多任务学习 - 有标签 有标签 自我学习 - 无标签 有标签	分类回归
	直推式迁移学习	相关	相同	有标签 无标签	分类回归
	无监督迁移学习	相关	相关	无标签 无标签	聚类降维

授人以鱼不如授人以渔：迁移学习

□ 迁移学习的分类

■ 按迁移学习的基本方法分

□ 基于实例的迁移学习方法

- 在源域中找到与目标域相似的数据，把这个数据的权值进行调整，使得新的数据与目标域的数据进行匹配。然后进行训练学习，得到适用于目标域模型。



迁移学习的实现方法（1）：样本迁移



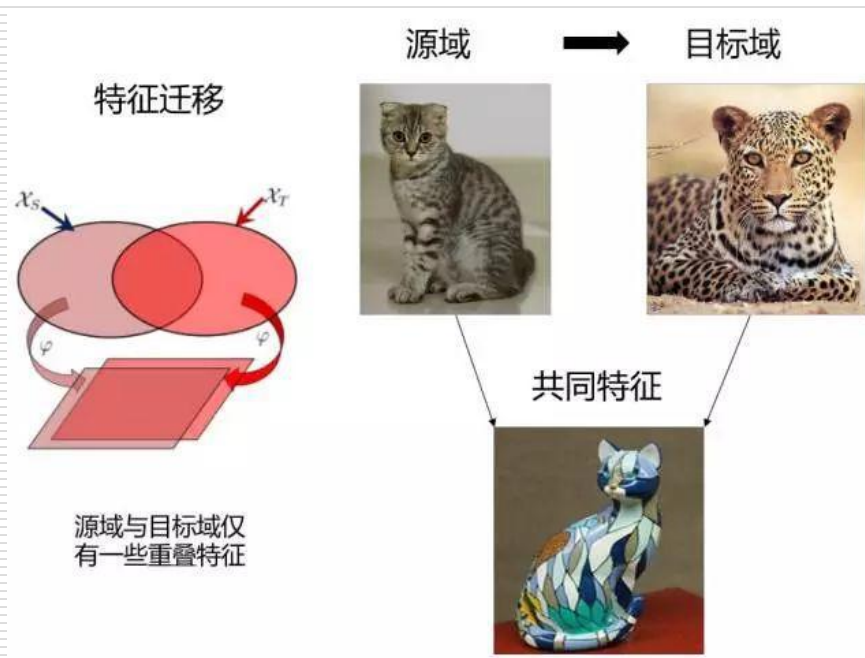
授人以鱼不如授人以渔：迁移学习

□ 迁移学习的分类

■ 按迁移学习的基本方法分

□ 基于特征的迁移学习方法

- 当源域和目标域含有一些共同的交叉特征时，我们可以通过特征变换，将源域和目标域的特征变换到相同空间，使得该空间中源域数据与目标域数据具有相同分布的数据分布，然后进行传统的机器学习。优点是对大多数方法适用，效果较好。缺点在于难于求解，容易发生过适配



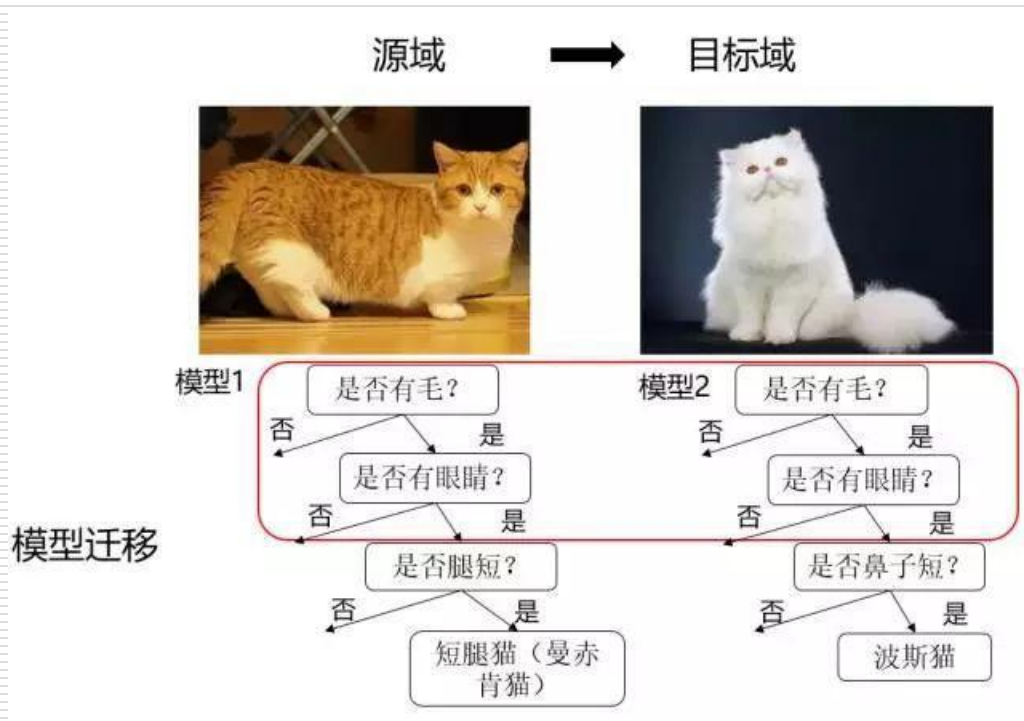
授人以鱼不如授人以渔：迁移学习

□ 迁移学习的分类

■ 按迁移学习的基本方法分

□ 基于模型的迁移学习方法

- 源域和目标域共享模型参数，也就是将之前在源域中通过大量数据训练好的模型应用到目标域上进行预测。比较直接，优点是可以充分利用模型之间存在的相似性。缺点在于模型参数不易收敛



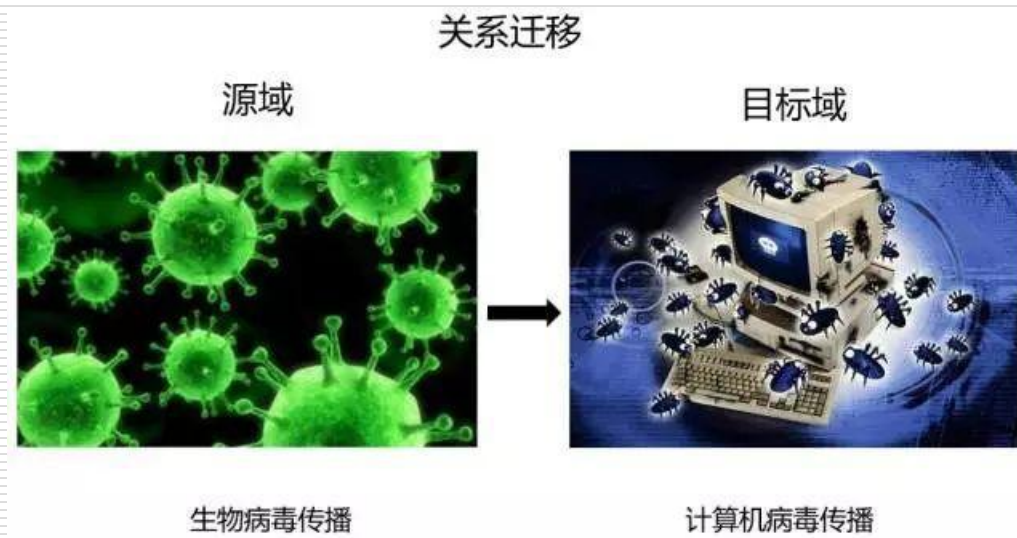
授人以鱼不如授人以渔：迁移学习

□ 迁移学习的分类

■ 按迁移学习的基本方法分

□ 基于关系的迁移学习方法

- 当两个域是相似的时候，那么它们之间会共享某种相似关系，将源域中学习到的逻辑网络关系应用到目标域上来进行迁移，比方说生物病毒传播规律到计算机病毒传播规律的迁移。这部分的研究工作比较少。典型方法就是mapping的方法



授人以鱼不如授人以渔：迁移学习

	说明	归纳式	直推式	无监督
基于样本的迁移学习	通过调整 源Domain的标签（辅助） 和 目标Domain标签的权重，协同训练得到目标模型。 典型方法：TrAdaBoost	√	√	
基于特征的迁移学习	找到 “好”特征 来减少源Domain和目标Domain之间的不同，能够降低分类、回归误差。 典型方法：Self-taught learning, multi-task structure learning	√	√	√
基于参数的迁移学习	发现源Domain和目标Domain之间的共享参数或先验关系。 典型方法：Learning to learn, Regularized multi-task learning	√		
基于相关性的迁移学习	建立源Domain和目标Domain之间的相关知识映射。 典型方法：Mapping 方法	√		

授人以鱼不如授人以渔：迁移学习

- 在从源域到目标域的迁移过程中，迁移学习要解决的问题
 - 迁移什么
 - 迁移学习的作用对象
 - 迁移的对象应该是普适性的知识
 - 能不能迁移
 - 源域和目标域具有较高的相关性
 - 如何迁移

实例：基于keras框架

- 在ImageNet中，图像的中物体的类别多达1000种。当面对另外一个image dataset时，我们能否最大程度地利用已经训练好的VGG16来完成我们的任务，或者说是将VGG16中已经学到的经验“迁移”到新到问题中呢？

实例：基于keras框架

□ 问题介绍

- 设计能够分辨图像中的动物是猫还是狗的方法



Dogs vs. Cats

Create an algorithm to distinguish dogs from cats

215 teams · 4 years ago

<https://blog.csdn.net/baimafujinji>

实例：基于keras框架

□ 数据集

- 原始数据集中包含12,500张猫的图片 and 12,500张狗的图片，从中各取前1000张来作为训练数据，并再各去400张来作为验证数据。然后创建如下图所示的目录结构，用来存放上述提到的这些图像文件：



实例：基于keras框架

□ 数据预处理

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```


实例：基于keras框架

□ 产生一些图片，并将它们存入一个临时的文件夹

```
img = load_img('data/train/cats/cat.0.jpg') # this is a PIL image
x = img_to_array(img) # this is a Numpy array with shape (3, 150, 150)
x = x.reshape((1,) + x.shape) # a Numpy array with shape (1, 3, 150, 150): 一张图, 三个通道

# the .flow() command below generates batches of randomly transformed images
# and saves the results to the `preview/` directory
i = 0
for batch in datagen.flow(x, batch_size=1,
                          save_to_dir='preview', save_prefix='cat', save_format='jpeg'):
    i += 1
    if i > 20:
        break # otherwise the generator would loop indefinitely
```

实例：基于keras框架

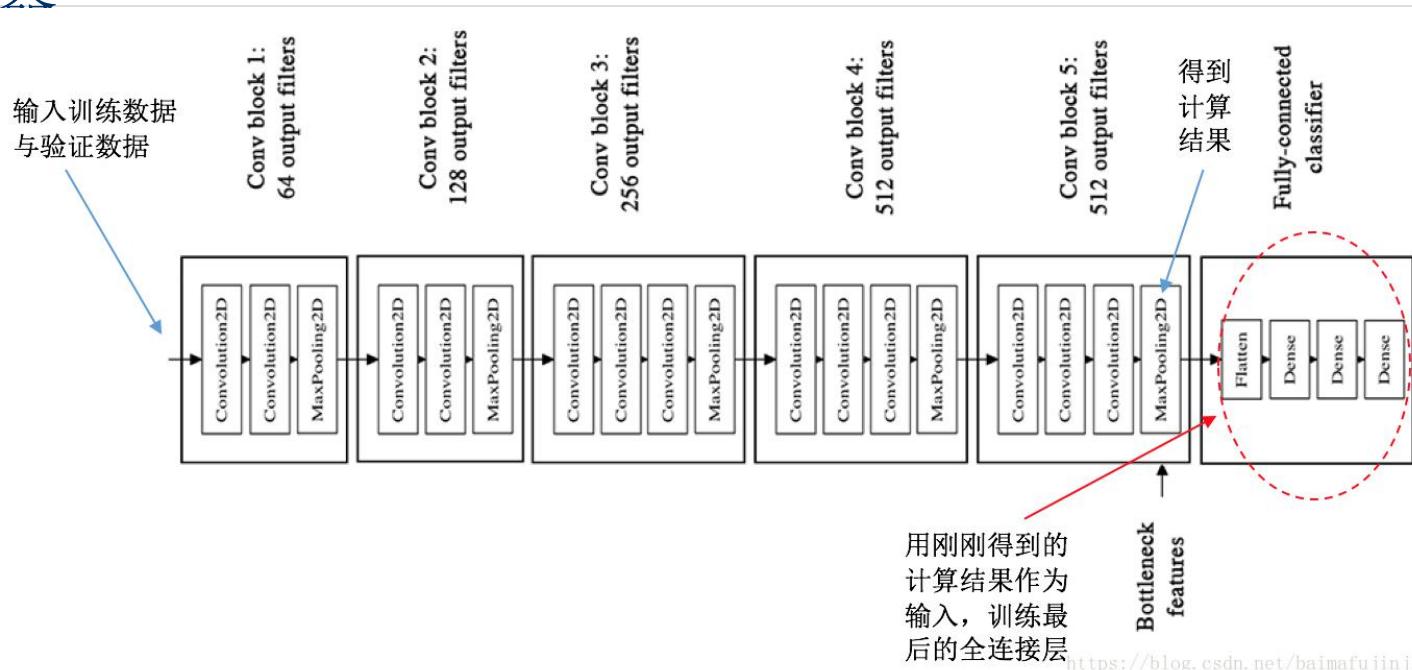
□ 一张图就被参数化地生成了多张图



实例：基于keras框架

□ 修改VGG16，得到一个90%准确率的分类器

- 用前面的若干卷积层来作为图像特征的提取和编码器，经过一次计算，训练数据和验证数据都被“编码”了，而这些编码中就蕴含着它们的特征。然后我们将这些新的编码输入到最后的全连接层，用来训练最后的softmax分类器



实例：基于keras框架

□ 首先引入必要的包，并定义一些全局变量

```
import numpy as np
from PIL import Image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications

# dimensions of our images.
img_width, img_height = 150, 150

top_model_weights_path = 'bottleneck_fc_model.h5'
train_data_dir = 'data/train'
validation_data_dir = 'data/validation'
nb_train_samples = 2000
nb_validation_samples = 800
epochs = 50
batch_size = 16
```

实例：基于keras框架

- 经由最后一个卷积层获得（测试集和验证集）特征编码，并将其存储起来备用

```
def save_bottlebeck_features():  
    datagen = ImageDataGenerator(rescale=1. / 255)  
  
    # build the VGG16 network  
    model = applications.VGG16(include_top=False, weights='imagenet')  
  
    generator = datagen.flow_from_directory(  
        train_data_dir,  
        target_size=(img_width, img_height),  
        batch_size=batch_size,  
        class_mode=None,  
        shuffle=False)  
    bottleneck_features_train = model.predict_generator(  
        generator, nb_train_samples // batch_size)  
    np.save('bottleneck_features_train.npy', bottleneck_features_train)  
  
    generator = datagen.flow_from_directory(  
        validation_data_dir,  
        target_size=(img_width, img_height),  
        batch_size=batch_size,  
        class_mode=None,  
        shuffle=False)  
    bottleneck_features_validation = model.predict_generator(  
        generator, nb_validation_samples // batch_size)  
    np.save('bottleneck_features_validation.npy', bottleneck_features_validation)
```

实例：基于keras框架

□ 最后，定义自己的softmax层，并进行训练

```
def train_top_model():
    train_data = np.load('bottleneck_features_train.npy')
    train_labels = np.array(
        [0] * (nb_train_samples // 2) + [1] * (nb_train_samples // 2))

    validation_data = np.load('bottleneck_features_validation.npy')
    validation_labels = np.array(
        [0] * (nb_validation_samples // 2) + [1] * (nb_validation_samples // 2))

    model = Sequential()
    model.add(Flatten(input_shape=train_data.shape[1:]))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy', metrics=['accuracy'])

    model.fit(train_data, train_labels,
              epochs=epochs,
              batch_size=batch_size,
              validation_data=(validation_data, validation_labels))
    model.save_weights(top_model_weights_path)
```


实例：基于keras框架

- 通过函数调用，来将整个过程串接起来

```
save_bottlebeck_features()  
train_top_model()
```

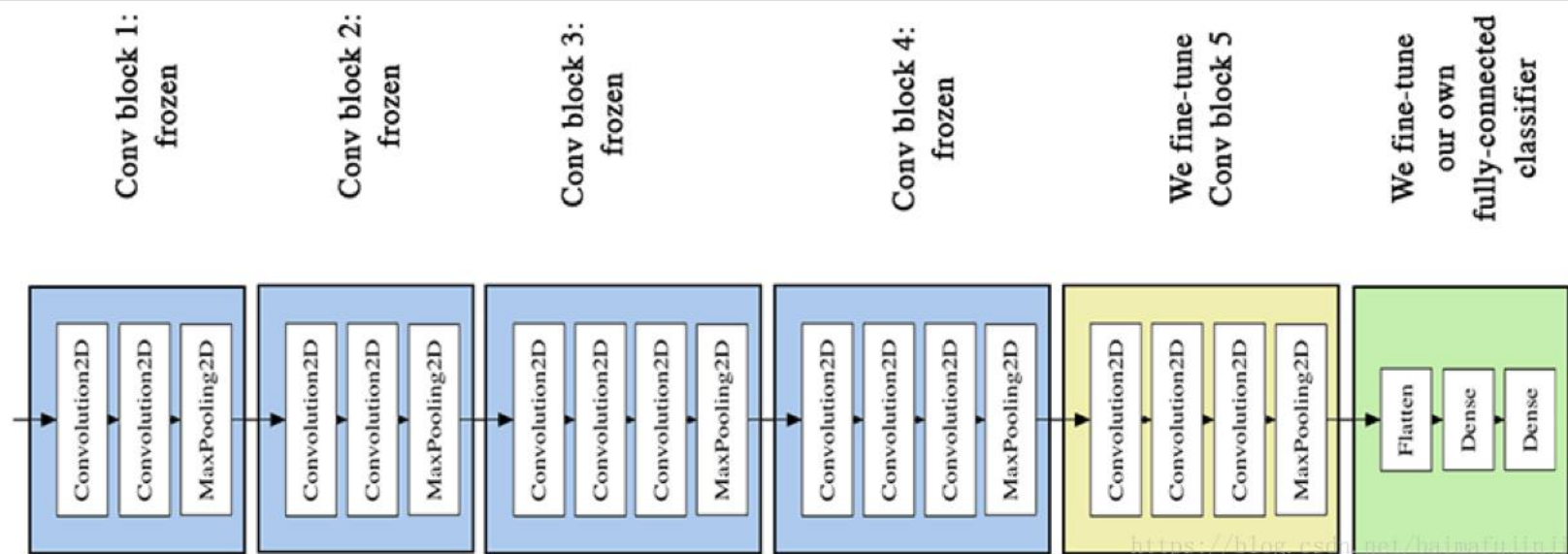
- 在很短的训练时间内，分类器的准确率就已经可以达到90%

```
2000/2000 [=====] - 4s 2ms/step - loss: 0.0289 - acc: 0.9910 - va  
Epoch 24/50  
2000/2000 [=====] - 4s 2ms/step - loss: 0.0281 - acc: 0.9900 - va  
Epoch 25/50  
2000/2000 [=====] - 4s 2ms/step - loss: 0.0261 - acc: 0.9915 - va  
Epoch 26/50
```


实例：基于keras框架

□ 更进一步，得到一个94%的分类器

- 首先初始化VGG16并加载；然后，在top的位置加入先前定义的全连接网络（绿色部分），并载入训练好的权重；然后将VGG模型中的前四个卷积块（蓝色部分）冻结，即训练时不更新它们的权重



实例：基于keras框架

□ 引入必要的包，并定义一些全局变量

```
import numpy as np
from PIL import Image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, Model
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from keras import optimizers

# dimensions of our images.
img_width, img_height = 150, 150

train_data_dir = 'data/train'
validation_data_dir = 'data/validation'
nb_train_samples = 2000
nb_validation_samples = 800
epochs = 50
batch_size = 16

# path to the model weights files.
top_model_weights_path = 'bottleneck_fc_model.h5'
```

实例：基于keras框架

- instantiate the convolutional base of VGG16 and load its weights

```
# build the VGG16 network  
base_model = applications.VGG16(weights='imagenet', include_top=False, input_shape=(150,150,3))  
print('Model loaded.')
```

- 加入全连接层

```
# build a classifier model to put on top of the convolutional model  
top_model = Sequential()  
top_model.add(Flatten(input_shape=base_model.output_shape[1:]))  
top_model.add(Dense(256, activation='relu'))  
top_model.add(Dropout(0.5))  
top_model.add(Dense(1, activation='sigmoid'))  
  
# note that it is necessary to start with a fully-trained  
# classifier, including the top classifier,  
# in order to successfully do fine-tuning  
top_model.load_weights(top_model_weights_path)  
  
# add the model on top of the convolutional base  
model = Model(inputs=base_model.input, outputs=top_model(base_model.output))
```

实例：基于keras框架

□ 将最后一个卷积块之前的卷积层全部冻结

```
# set the first 15 layers (up to the last conv block)  
# to non-trainable (weights will not be updated)  
for layer in model.layers[:15]:  
    layer.trainable = False  
  
# compile the model with a SGD/momentum optimizer  
# and a very slow learning rate.  
model.compile(loss='binary_crossentropy',  
              optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),  
              metrics=['accuracy'])
```

实例：基于keras框架

□ 准备数据

```
# prepare data augmentation configuration
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary')
```

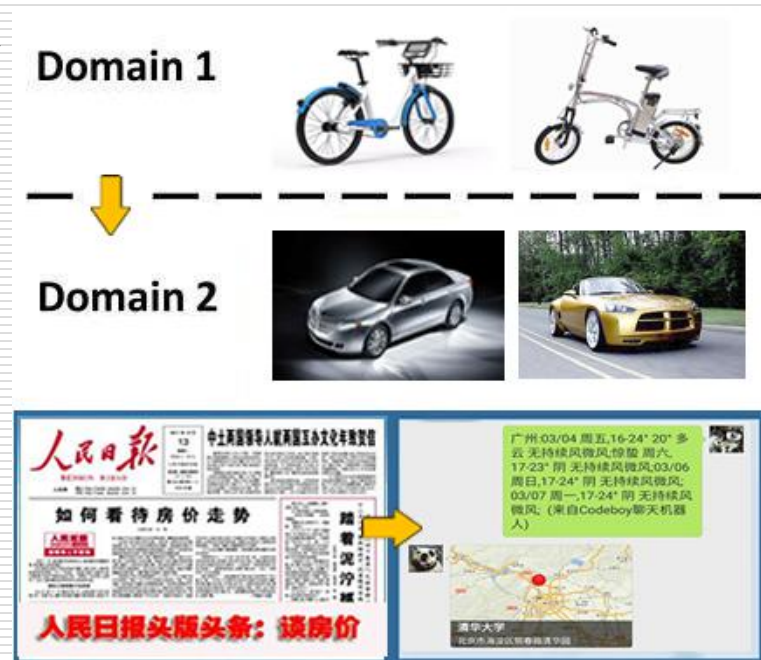
实例：基于keras框架

□ 训练新的模型

```
# fine-tune the model
model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)
```

迁移学习算法

- ❑ 通过 原有数据 和 少量新领域数据混淆训练；
- ❑ 将原训练模型进行分割，保留基础模型（数据）部分作为新领域的迁移基础；
- ❑ 通过三维仿真来得到新的场景图像（OpenAI的Universe平台借助赛车游戏来训练）；
- ❑ 借助对抗网络 GAN 进行迁移学习



授人以鱼不如授人以渔：迁移学习

□ 迁移学习热门研究方向

- 域适配问题(domain adaptation): 有标签的源域和无标签的目标域共享相同的特征和类别，但是特征分布不同，如何利用源域标定目标域。
 - 解决Domain adaptation问题主要的思路就是将source训练好的模型能够用在target上，而域适配问题最主要的也就是如何减少source域和target域不同分布之间的差异。
 - 代表性论文有：Domain adaptation via transfer component analysis--基于特征的迁移方法；Density ratio estimation in machine learning--基于实例的迁移方法；Cross-domain video concept detection using adaptive svms--基于模型的迁移方法等等

授人以鱼不如授人以渔：迁移学习

□ 迁移学习热门研究方向

- 多源迁移学习(multi-source TL):多个源域和目标域，通过进行有效的域筛选，从而进行迁移。
 - 多源迁移学习可以有效利用存在的多个可用域，综合起来进行迁移，达到较好的效果。
 - 当然现在如何衡量多个域之间的相关性和多个域的利用方法还是一个比较大的问题
 - 代表性论文有：Boosting for transfer learning; Multi-source transfer learning with multi-view adaboost等等

授人以鱼不如授人以渔：迁移学习

□ 迁移学习热门研究方向

- 深度迁移学习(deep TL):特别是近年来由于深度学习的火爆,越来越多研究者利用深度神经网络的结构进行迁移学习,深度学习可以深度表征域中的知识结构,也大大增强了模型的泛化能力,可以说利用深度学习做迁移学习的前景还是很好的。

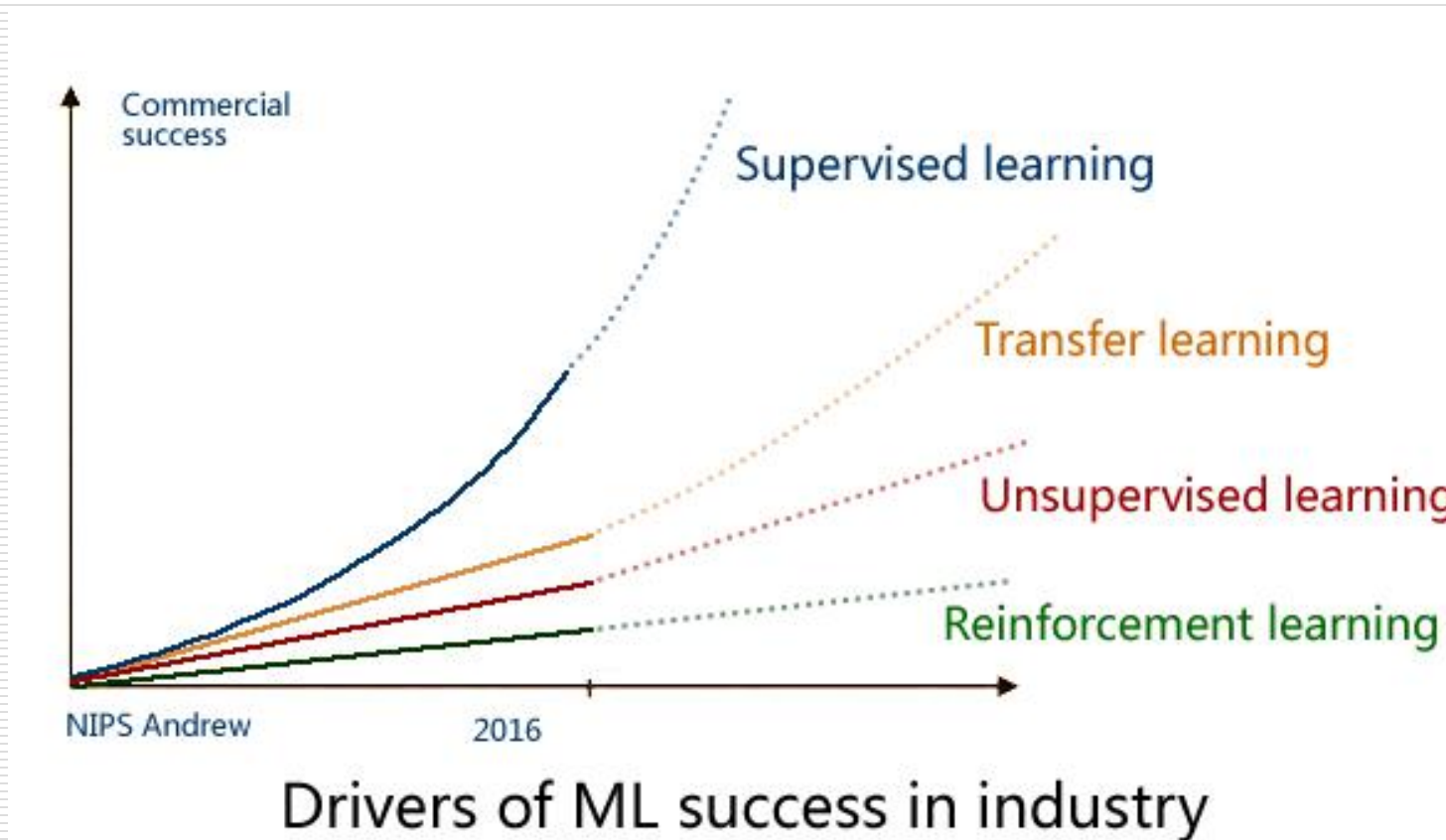
- 代表性论文有: Simultaneous deep transfer across domains and tasks; Multi-source transfer learning with multi-view adaboost; Learning Transferable Features with Deep Adaptation Networks 等等

总结与思考

- ❑ 迁移学习是运用已学习的知识来求解不同但相关领域问题的新的机器学习方法
- ❑ 迁移学习适用于跨领域和小数据（例如传感器数据、文本分类、图片分类等）的学习任务
- ❑ 迁移学习的任务类型可以分为归纳迁移学习、直推式迁移学习和无监督迁移学习
- ❑ 迁移学习的学习方法包括基于样本、基于特征、基于模型和基于关系的学习方法

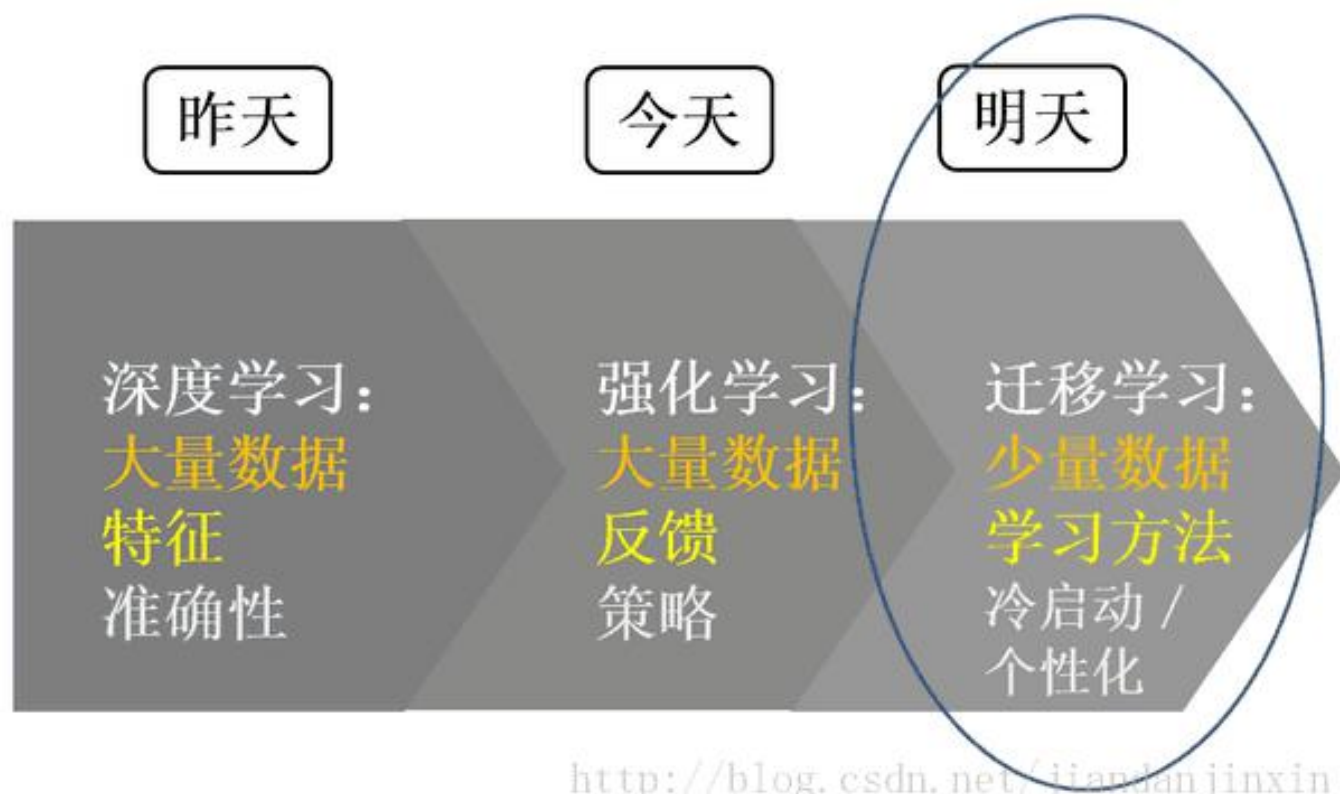
总结与思考

- 2016年，人工智能大咖吴恩达表示“继深度学习之后，迁移学习将引领下一波机器学习技术”。
- 那么，你如何看待迁移学习的前景呢？



总结与思考

机器学习：昨天，今天，明天





THE END