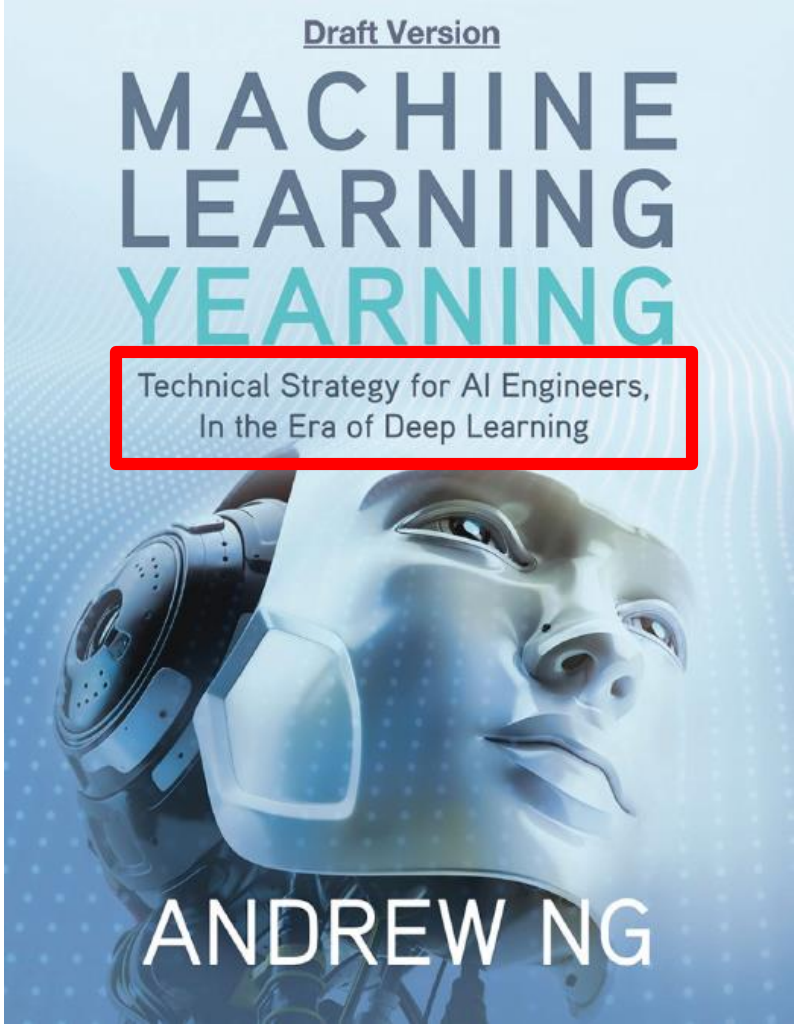
Machine Learning Yearning



- 作者 Andrew Ng

Lectures contents

- 设置开发集与测试集
- 基础误差分析
- 偏差与方差
- 学习曲线

- **与人类表现水平对比**
- **在不同的分布上训练与测试**
- **端到端深度学习**
- **根据组件进行误差分析**

Lectures contents

- 设置开发集与测试集
- 基础误差分析
- 偏差与方差
- 学习曲线

- **与人类表现水平对比**
- 在不同的分布上训练与测试
- 端到端深度学习
- 根据组件进行误差分析

与人类表现水平对比

- **为何与人类表现水平进行对比**

  有许多理由表明在处理人类擅长的任务时，构建一个机器学习系统会更加简单……

  - **易于从人为标签中获取数据；**
  - **基于人类直觉进行误差分析；**
  - **使用人类表现的优异水平来估计最优错误率，并设置可达到的“期望错误率”；**

与人类表现水平对比

- **为何与人类表现水平进行对比**

连人类都不擅长的任务也是存在的……比如预测股票市场走势，又比如判断下面监控中图像两人是否是同一个人。

=?

与人类表现水平对比

- **如何定义人类表现水平**

假设你正在做一个医学成像应用程序，它可以自动依据X射线图像进行诊断:

- 除了一些基础的训练外，一个没有任何医学背景的人在该任务上的错误率为 15% 。
- 一名新手医生的错误率为10% 。
- 而经验丰富的医生可以达到 5% 。
- 由小型的医生团队对每一幅图像进行单独的讨论，错误率将降低至 2% 。

上述的哪一种错误率可以定义为“人类表现水平”呢？

与人类表现水平对比

- **超越人类表现水平**

在许多重要的机器学习应用程序中，机器已经超越了人类的水平。例如，机器可以更好地预测电影分级，一辆送货车到某个地方需要多长时间，或者是否批准贷款申请。

其他机器学习/深度学习超越人类表现的例子？

当人类很难识别出算法明显出错的样本时，可应用的技术范围则会受到局限。因此在机器已经超越人类水平的问题上，进展通常比较慢，而当机器仍在试图赶上人类水平时，进展速度反而更快。

Lectures contents

- 设置开发集与测试集
- 基础误差分析
- 偏差与方差
- 学习曲线

- 与人类表现水平对比
- **在不同的分布上训练与测试**
- 端到端深度学习
- 根据组件进行误差分析

在不同的分布上训练与测试

- **何时在不同的分布上训练与测试**

假设用户已经向你的猫咪图片程序上传了10000张图片，且图片已被人为标记为含有猫与不含猫两类。同时你也从互联网上下载了规模更大的 200000 张图片集，此时训练集、测试集与开发集应该如何定义呢？

一种可行的做法是：不将用户上传的所有10000个图像放到开发/测试集合中，而是将其中 5000 张放入。 这样的话，训练集中的 205000 个样本的分布将来自现有的开发/测试集，以及 200000 张网络图片。我们将在后续讨论为什么这个方法是有帮助的。

在不同的分布上训练与测试

- **何时在不同的分布上训练与测试**

  -由于用户的 10000 张图片密切地反映了你想要处理的数据的实际概率分布，因此你可以将它们（部分）作为开发集与测试集。如果你正在训练一个数据量需求极大的深度学习算法，则可能需要使用额外200000 张网络图片来进行训练。这样的话，你的训练集与开发集/测试集将服从不同的概率分布。这对你的工作会有什么影响呢？

（答案后续揭晓……）

在不同的分布上训练与测试

- **何时在不同的分布上训练与测试**

  -与传统方法相比，在大数据时代，我们可以使用大型的训练集，比如猫的网络图像。即使训练集的分布不同，**我们仍然希望使用它来学习**，因为它可以提供大量的信息。但重要的是你要明白，不同的训练和开发/测试集分布将带来一些特殊的挑战。

在不同的分布上训练与测试

- **如何决定是否使用你所有的数据**

  回看例子：

  10000 张用户上传的图片：这些数据来自相同的数据分布且将5000作为单独的开发/测试集，同时也代表着你关心的将要处理的数据分布。

  互联网下载的额外200000张图片：海量图片，但可能与你关心的数据分布不同。

  此时你是否应该为你的学习算法提供所有的200000 + 5000 张图片作为它的训练集，或者丢弃这200000 张网络图片，以免它会影响你的学习算法呢？

在不同的分布上训练与测试

- **如何决定是否使用你所有的数据**

  添加额外的 200000 张图片会产生以下影响：

  1. 它给你的神经网络提供了**更多关于猫咪外貌**的样本。这是很有帮助的，因为互联网图片和用户上传的移动应用图片确实有一些相似之处。你的神经网络可以**将从互联网图像中获得的一些知识应用到移动应用图像中**。

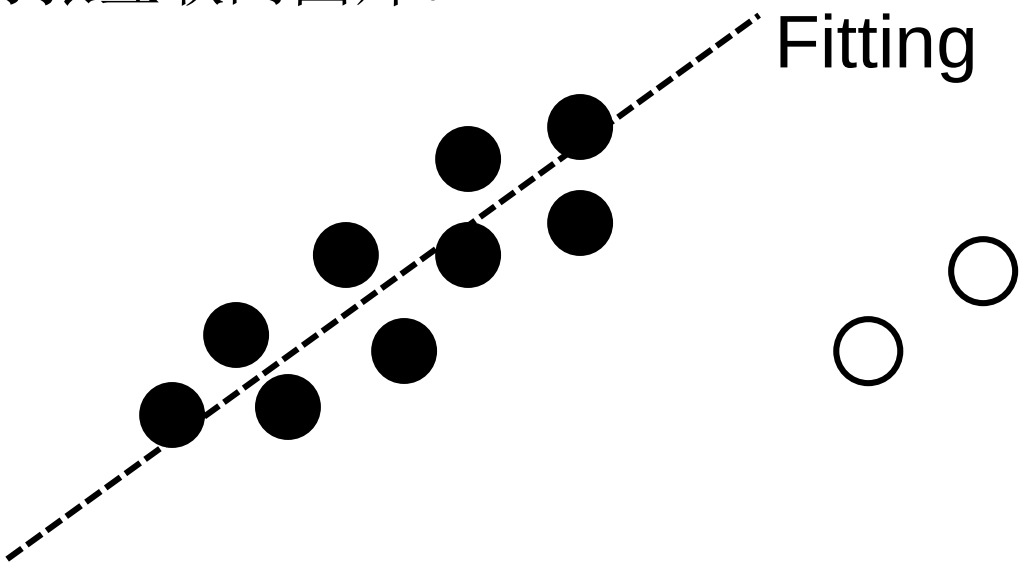
  2. 它迫使神经网络花费部分容量来学习网络图像的特定属性（比如更高的分辨率，不同画面结构图像的分布等等）。如果这些属性与移动应用图像有很大的不同，那么它将“耗尽”神经网络的一些表征能力，导致从移动应用图像的分布识别数据的能力就会降低，而这正是你真正关心的东西。从理论上讲，这可能会损害算法的性能。

在不同的分布上训练与测试

- **如何决定是否使用你所有的数据**

  添加额外的 200000 张图片会产生以下影响：

  3. 在使用早期的学习算法（比如人为设计的计算机视觉特征，然后使用一个简单的线性分类器）时，真正的风险在于：合并这两种类型的数据会导致算法的表现更差。因此，一些工程师会警告你不要加入20万张互联网图片。

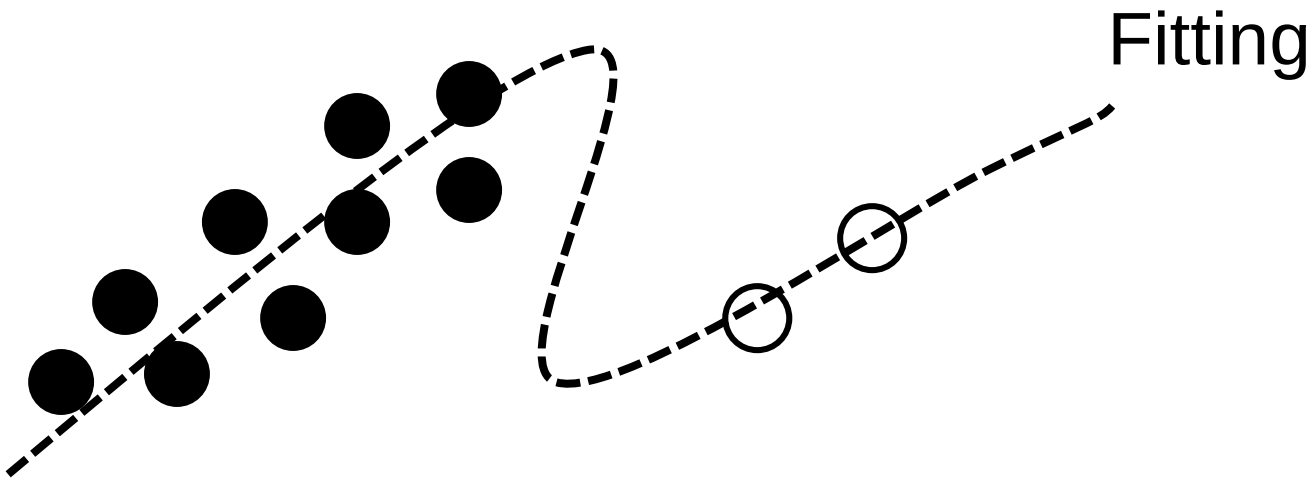


在不同的分布上训练与测试

- **如何决定是否使用你所有的数据**

  添加额外的 200000张图片会产生以下影响：

  4.但是有了现代强大而灵活的学习算法——比如大型的神经网络——这种风险已经大大降低了。如果你能够构建一个有足够多的隐藏单元/层的神经网络，你可以安全地将 200000 张图片添加到你的训练集。此时添加图片则更有可能提升算法的性能。



在不同的分布上训练与测试

- **如何决定是否使用你所有的数据**

也就是说，如果你有足够的计算能力来构建一个足够大的神经网络，那么你会有足够的能力从互联网和移动应用图像中学习，而不会存在两种类型的数据在容量上的竞争，即是说，你的算法的“大脑”足够大。

如果你没有足够大的神经网络（或者另一个高度灵活的学习算法），那么你应该更加关注训练数据，需要与开发集/测试集的分布相匹配。

在不同的分布上训练与测试

- **给数据添加权重**

假设你有20万张来自互联网的图片，还有来自移动应用用户的 5000 张照片。数据集的大小之间有一个 40:1 的比率。从理论上讲，只要你建立了一个庞大的神经网络，并在所有205000 张图片上进行足够长的时间训练，那么在网络图像和移动图像上将算法都训练得很好是没有害处的。

但在实际操作中，拥有 40 倍的网络图像可能意味着，相比只使用 5000 张图片，你需要花费40 倍（或更多）的计算资源来对两者进行建模。如果你没有巨大的计算资源（包括神经网络的大小），你可以给互联网图片一个较低的权重作为妥协。

在不同的分布上训练与测试

- 给数据添加权重（损失函数）

$$\min_{\theta} \sum_{(x,y)\in \text{MobileImg}} \left(h_\theta(x) - y\right)^2 + \sum_{(x,y)\in \text{InternetImg}} \left(h_\theta(x) - y\right)^2$$

$$\Downarrow$$

$$\min_{\theta} \sum_{(x,y)\in \text{MobileImg}} \left(h_\theta(x) - y\right)^2 + \beta \sum_{(x,y)\in \text{InternetImg}} \left(h_\theta(x) - y\right)^2$$

在不同的分布上训练与测试

- **解决数据分布不匹配问题**

假设你已经开发了一个语音识别系统，它在训练集上做得很好。但是，它在你的开发集上做得很差：这表明有一个数据分布不匹配的问题。你会怎么做呢?

建议如下：

（i）尝试理解数据属性在训练集和开发集分布之间的差异。

（ii）尝试找到**更多的训练数据**，以便更好地匹配你的算法碰到的开发集样本。

在不同的分布上训练与测试

- **人工合成数据**

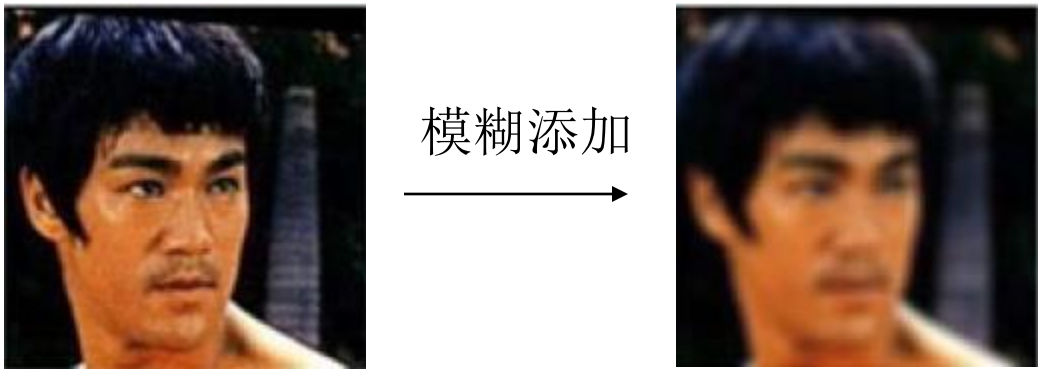
例如，你的语音系统需要更多的训练数据，它们听起来就像是从车里录制得到的。与其在开车的时候收集大量的数据，不如通过人工合成数据来获取这些数据。

假设你获得了大量的汽车/道路噪音的音频剪辑。你可以从几个网站下载这些数据。假设你也有一群在安静的房间里说话的人。如果你把一个人的音频片段“添加”到一个汽车/道路噪音的音频片段，你会得到一个音频剪辑，听起来就好像那个人在嘈杂的汽车里说话一样。使用这个过程，你可以“合成”大量的数据，听起来就像是在汽车里收集的……

在不同的分布上训练与测试

- **人工合成数据**

更一般的情况是，在一些情况下，人工合成数据允许你创建一个与开发集相当匹配的巨大数据集，让我们使用**视频人脸识别（Video-based Face Recognition**）作为第二个例子：你注意到，开发集的图像有更多的动态模糊，因为视频拍摄时人或摄像机都可能移动。你可以从互联网人脸图像的训练集中获取非模糊的图像，并将模拟的动态模糊添加到它们中，从而使它们更类似于开发集。



在不同的分布上训练与测试

- **人工合成数据**

请记住，人工数据合成存在一定的挑战：创建一个对人而言真实的合成数据比创建对计算机而言真实的数据要容易得多。例如，假设你有 1000 小时的语音训练数据，但只有 1 小时的汽车噪音。如果你反复使用相同的 1 小时的汽车噪音，从最初的 1000 小时的训练数据中，你将会得到一个合成的数据集，然而同样的汽车噪音会不断重复。听这段音频的人可能无法分辨——所有的汽车噪音对我们大多数人来说都是一样的——但是某种学习算法可能会“过拟合”1小时的汽车噪音。因此，它可能无法很好地泛化到一个新的音频剪辑片段，里面汽车的噪音听起来是不同的。

在不同的分布上训练与测试

- **人工合成数据**

另一种情况，假设你有1000个小时的汽车噪音片段，但所有的噪音都是从 10 辆不同的车上提取的。在这种情况下，一种算法可能会“过拟合”这 10 辆车，如果在不同的汽车上进行音频测试，性能则会很差。不幸的是，**由于神经网络的黑盒特性**，这些问题很难被发现。

在不同的分布上训练与测试

- **人工合成数据**

　　又例如，你正在建立一个计算机视觉系统来识别汽车：你正与一家电脑游戏公司合作，该公司拥有几辆汽车的计算机图形3D模型。为了训练你的算法，你可以使用这些模型来生成汽车的合成图像。即使合成的图像看起来非常真实，但这种方法（已经被许多人独立提出）可能不会很好地工作。在整个电脑游戏中，可能只有 20 种汽车设计（制造一辆汽车的 3D 模型价格是非常昂贵的）如果你在玩这个游戏，你可能不会注意到你正在一遍又一遍地看到同样的车，也许只是换了一种颜色。

在不同的分布上训练与测试

- **人工合成数据**

即这些数据对你来说很真实。但是，与所有在道路上行驶的汽车相比——也就是你可能在开发/测试集里看到的——这组根据 20 辆汽车模型合成的汽车只捕获了世界上销售的汽车的极小一部分。因此，如果你的 10 万个训练样本都来自这 20 辆车，你的系统将会“过拟合”这 20 款特定的汽车设计，而且它将无法很好地泛化到包含其他汽车设计在内的开发/测试集。

在不同的分布上训练与测试

- **人工合成数据**

当你在合成数据时，请考虑一下你是否真的在合成一组具有代表性的样本。**尽量避免给出合成数据的属性（即多加入一些随机性和多样性）**，否则这将使学习算法有可能将**合成**和**非合成**的真实样本区分开来——例如，所有的合成数据是否来自 20 个汽车设计中的某一个，或者所有的合成音频是否都来自于某个小时的汽车噪音。

在处理数据合成过程时，有的团队会花上几周的时间来生成带有细节的数据，这些数据与实际的数据分布非常接近，从而产生显著的效果。如果你能够正确地获取这些细节，你可以突然获得比以前更大的训练集。

Lectures contents

- 设置开发集与测试集
- 基础误差分析
- 偏差与方差
- 学习曲线

- 与人类表现水平对比
- 在不同的分布上训练与测试
- **端到端深度学习**
- 根据组件进行误差分析

端到端深度学习

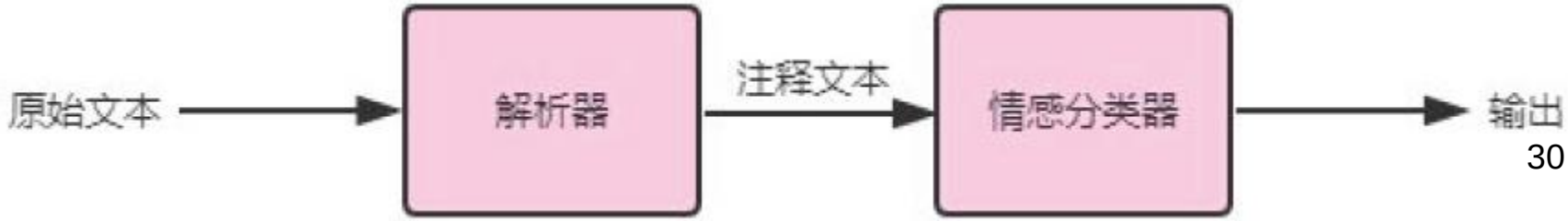
- **端到端学习的兴起**

假设你想要构建一个系统来对产品的线上评论进行检查，并且要能够自动地告诉你给出评论的人是否喜欢这个产品：

客户1：这个拖把非常好用

客户2：拖把的质量好差，我后悔买它了

这种识别正面与负面评论的问题被称为“情感分类”（sentiment classification）。想要构建一个这样的系统，你的流水线模块需要有以下两个组件：解析器和情感分类器↓。



端到端深度学习

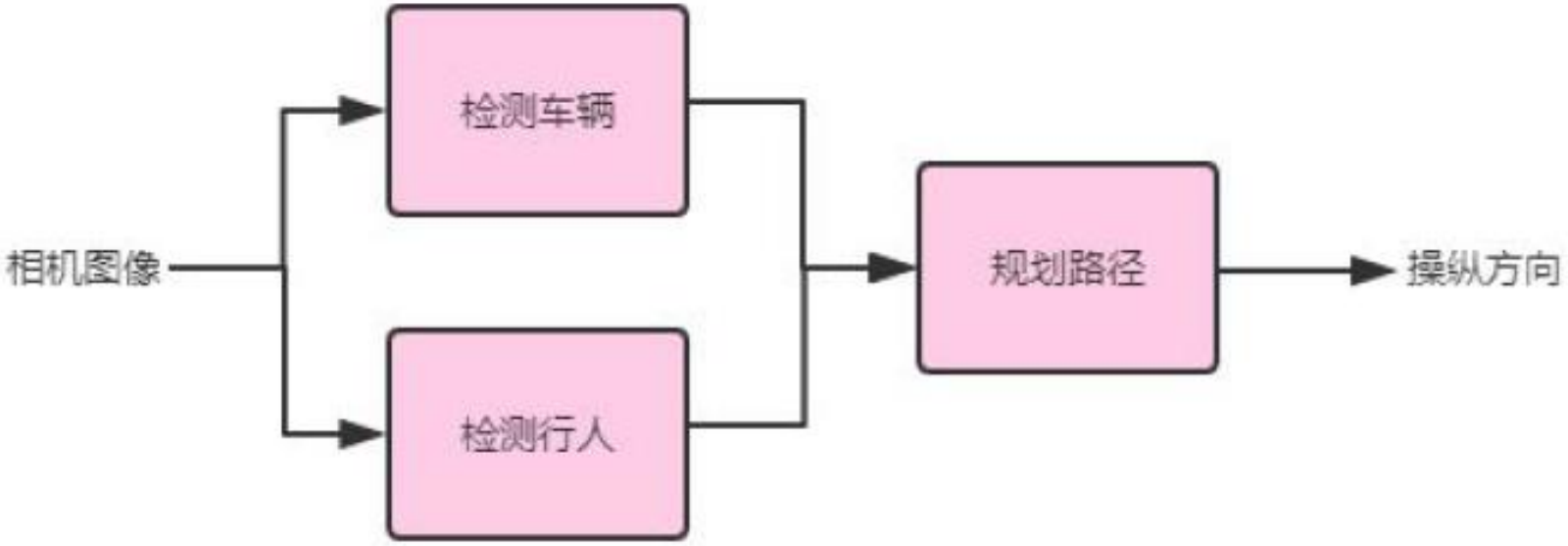
- **端到端学习的兴起**

最近兴起的趋势更倾向于用一种单一的学习算法取代此类流水线。该任务的“端到端（end-to-end）学习算法” 只需输入一个原始的文本 *“这个拖把非常好用！”*，接着尝试直接识别其中的情感。

端到端深度学习

- **端到端学习的更多例子**

在数据量十分丰富的问题上，端到端系统往往很奏效，但它并不总是一个很好的选择，请见如下几个例子。例如，这是一个自动驾驶汽车的简单流水线架构：



端到端深度学习

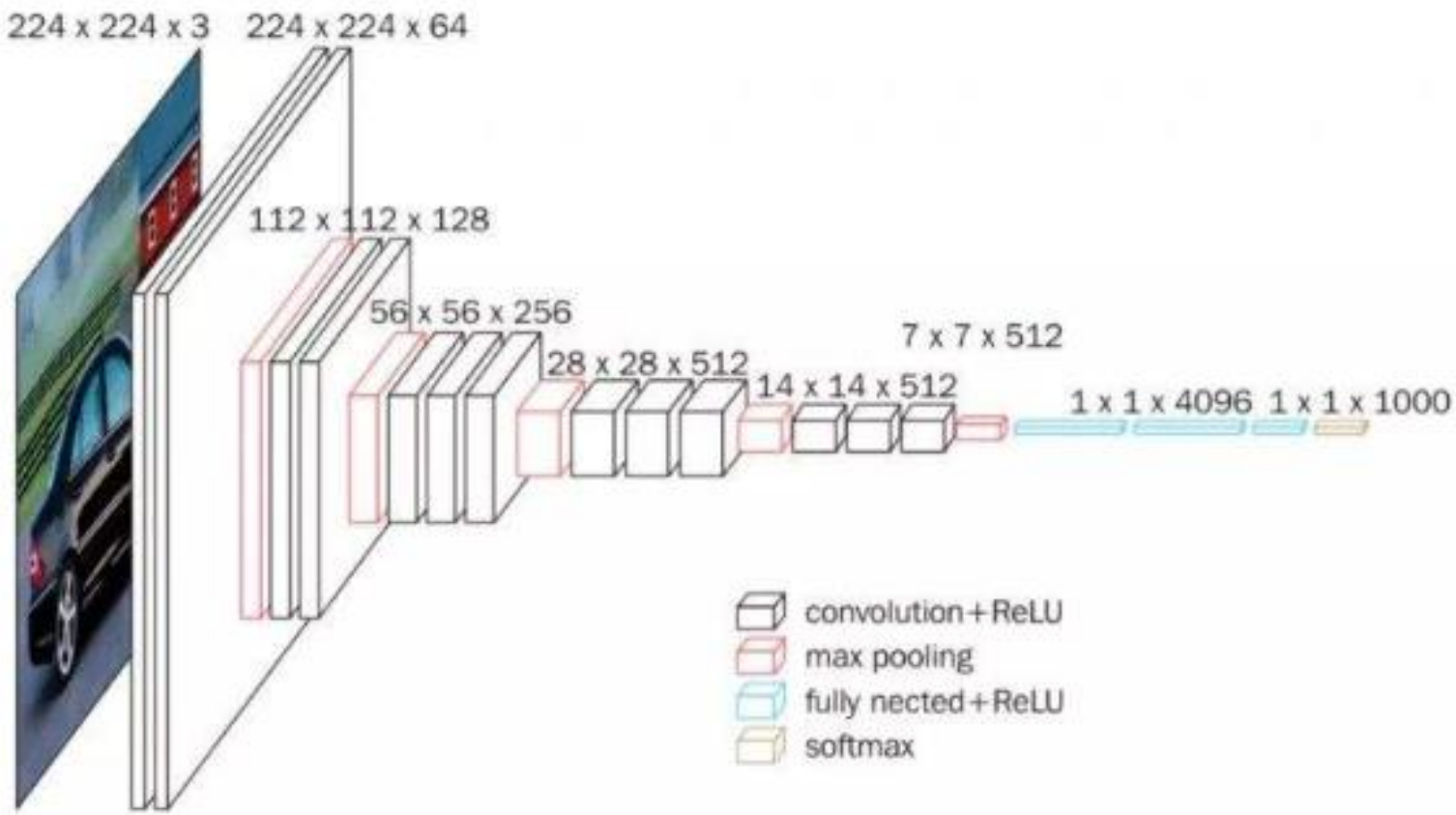
- **端到端学习的更多例子**

如果采用端到端学习，其可能会尝试从传感器获取输入并直接输出转向方向:

尽管端到端学习已经在许多领域取得了成功，但它并不总是最佳方案。端到端的语音识别功能很不错，但自动驾驶的端到端学习却可能相反（作者的观点）。后续将会对此进行解释……

端到端深度学习

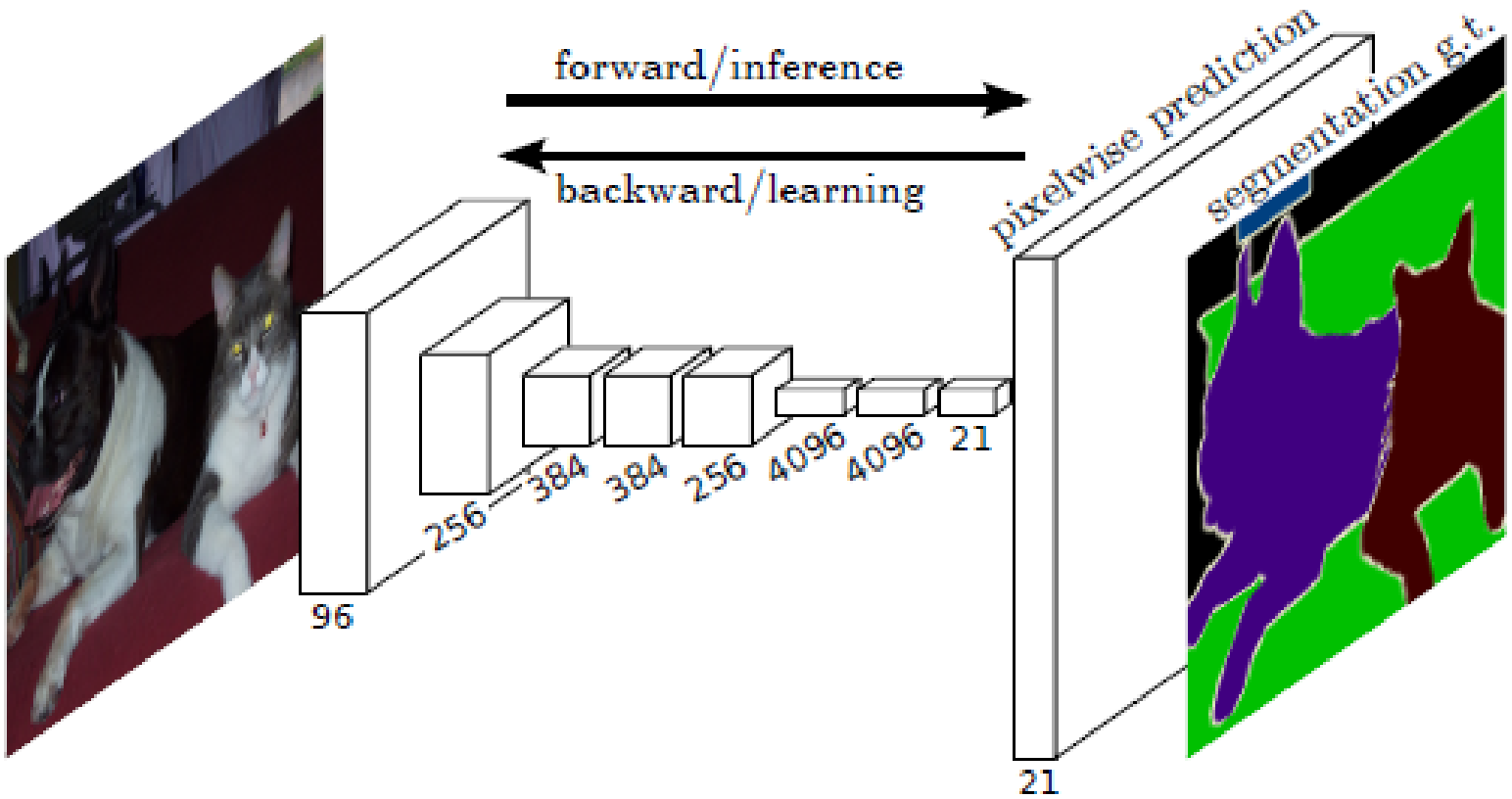
- **端到端学习的更多例子**



图像分类(Image classification)

端到端深度学习

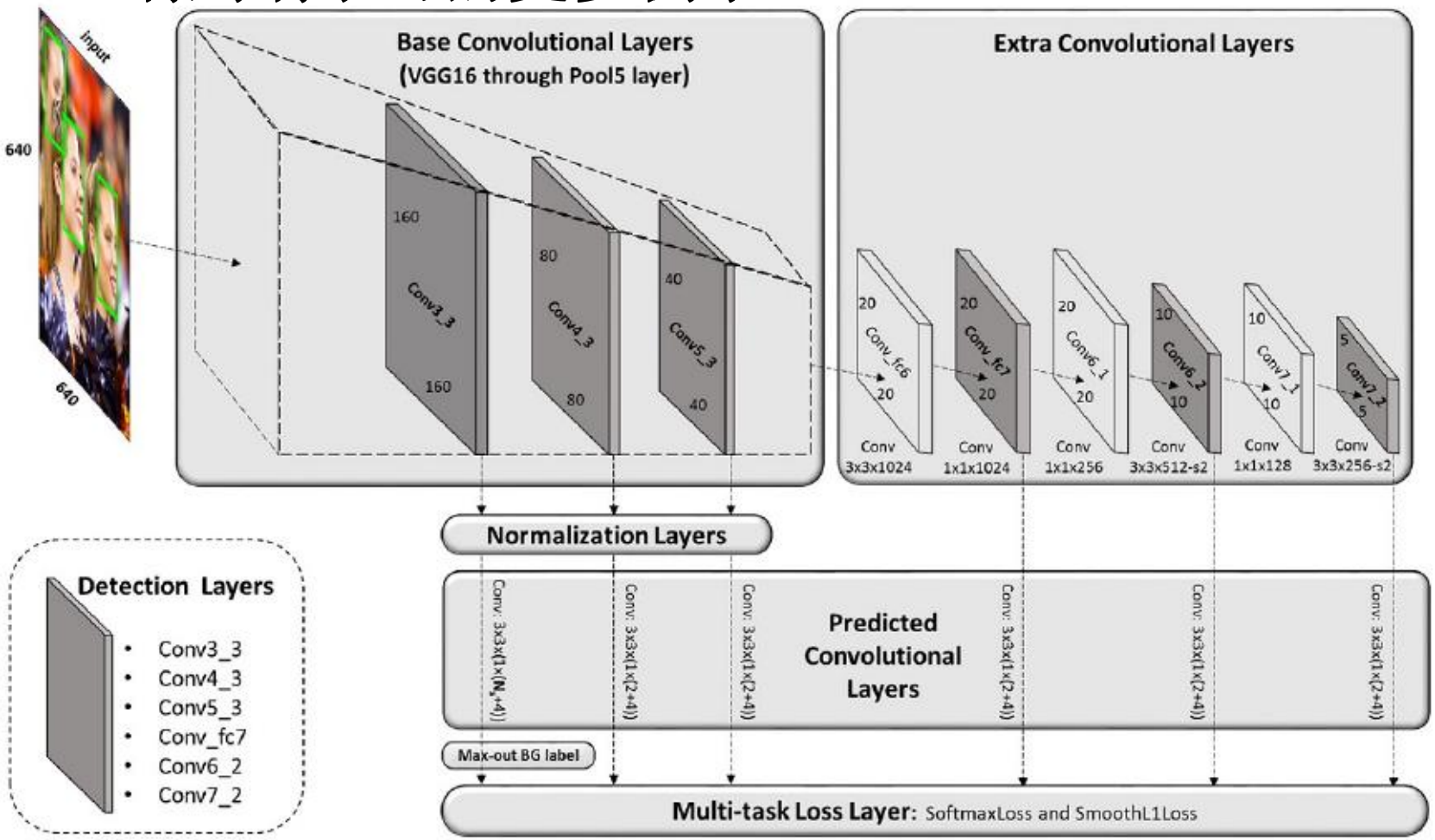
- **端到端学习的更多例子**



语义分割(Semantic Segmentation)

端到端深度学习

- 端到端学习的更多例子



人脸检测(Face Detection)

端到端深度学习

- **端到端学习的更多例子**

(c) John Wesley Shipp

(d) Leymah Gbowee

(e) Princess Haya Bint Al Hussein

(f) Julio César Chávez Jr.

(g) Roy Jones Jr.

(h) Ruby Lin

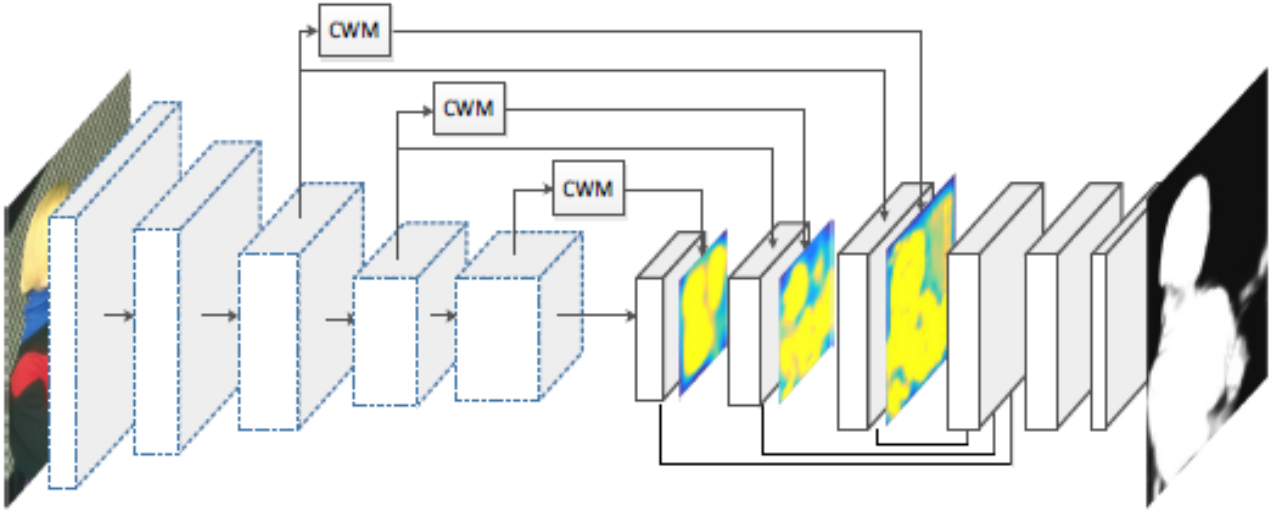
(i) Additi Gupta

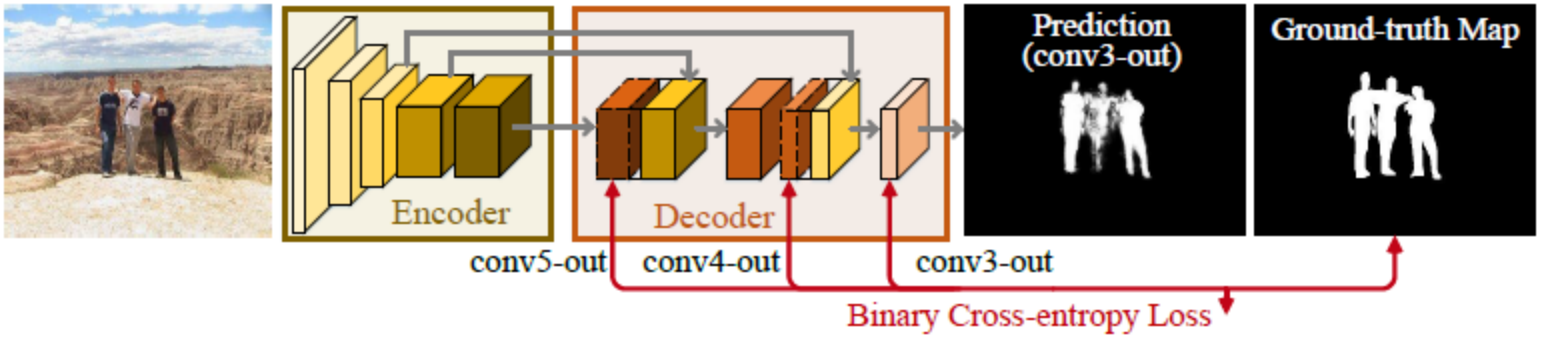
(j) Lee Joon-gi

人脸识别(Face Classification)

端到端深度学习

- **端到端学习的更多例子**

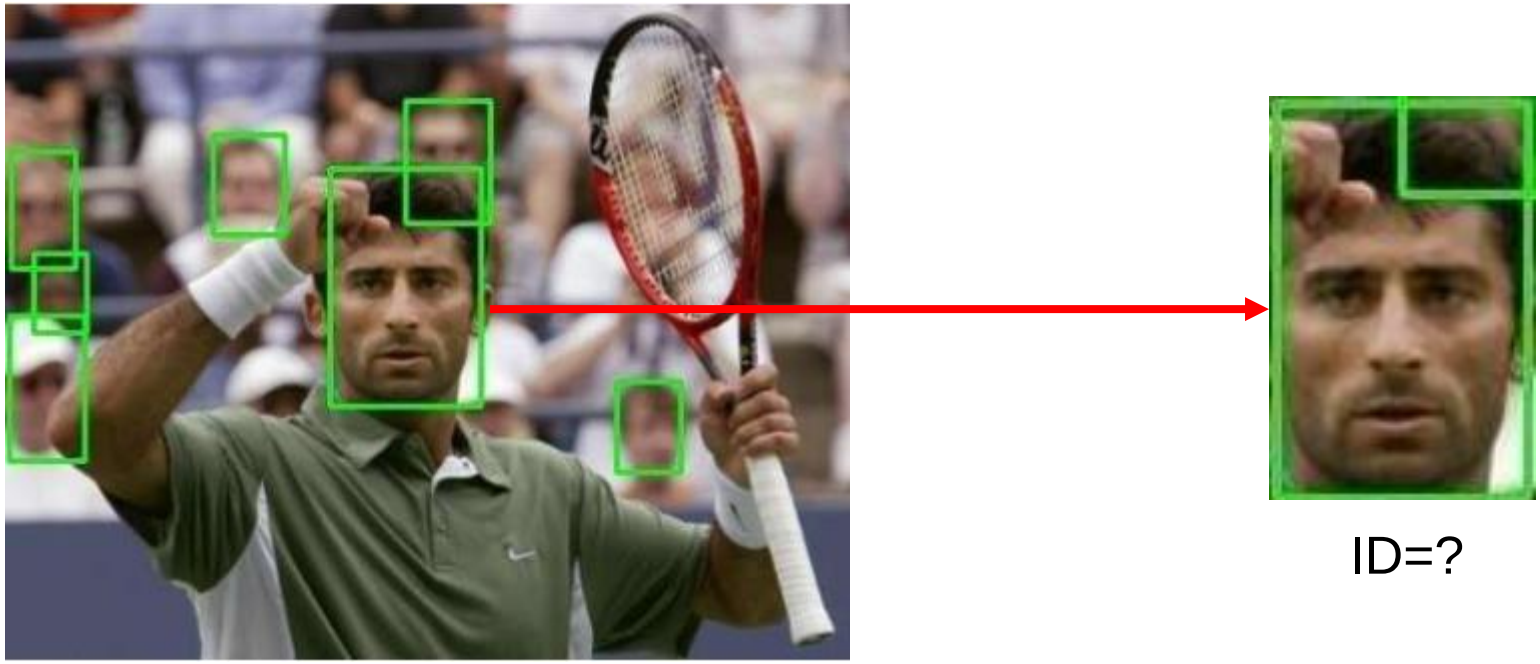




显著物体检测(Salient Object Detection)

端到端深度学习

- 非端到端学习的例子：完整的人脸识别系统



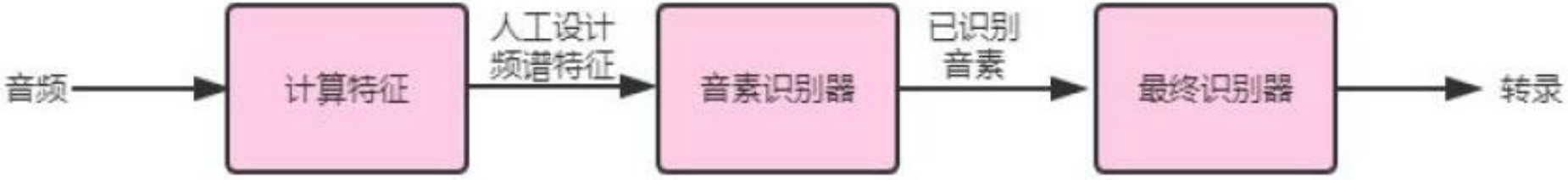
Complete scene image

完整的人脸识别系统基本无法实现端到端！

端到端深度学习

- **端到端学习的优缺点**

考虑某个语音识别系统的流水线：



该流水线中的许多部分都是“人工设计”的，拥有更多的人工设计成分通常可以让语音系统学习更少的数据，当我们的数据量不是很多时，这些知识是非常有用的！

端到端深度学习

- **端到端学习的优缺点**

考虑端到端的系统

这个系统缺乏人工设计知识，因此当训练集很小的时候，它的表现可能比人工设计的流水线更加糟糕！

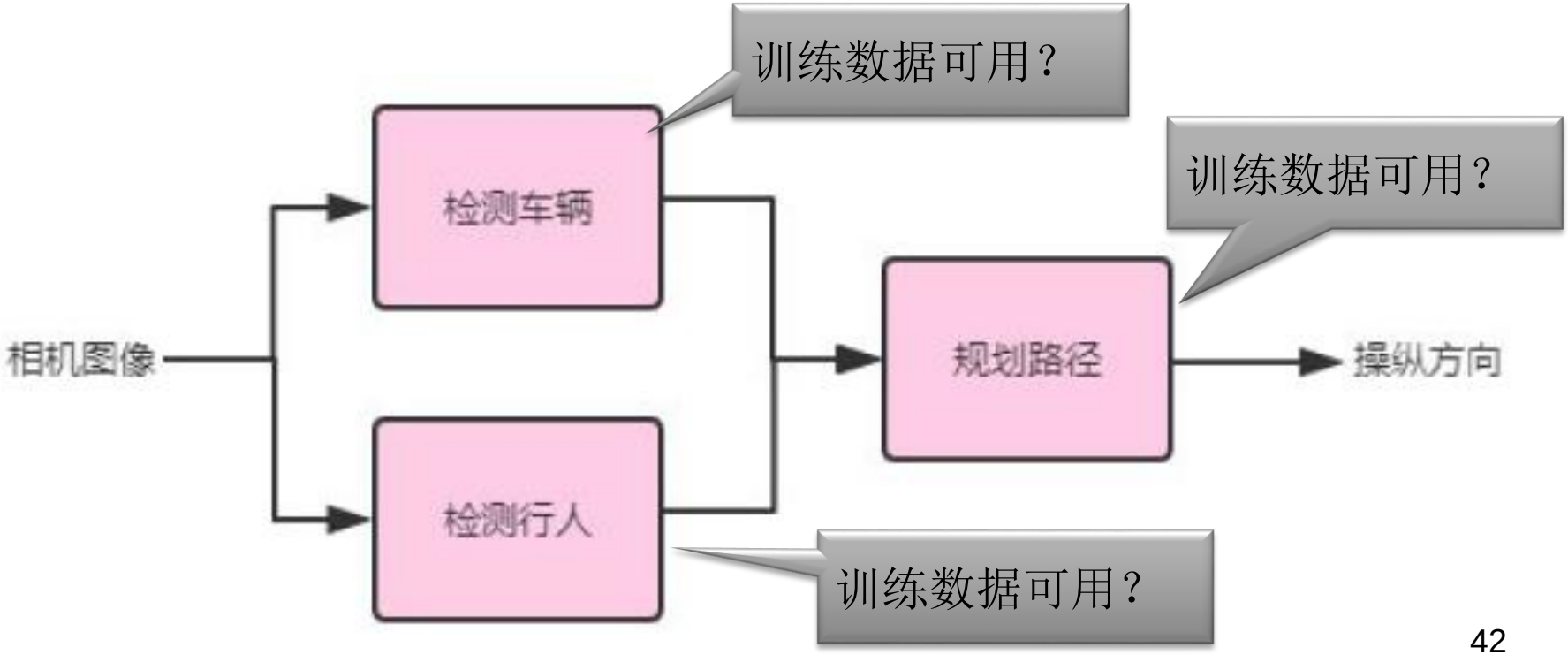
然而当**<u>训练集很大</u>**时，它不会受到人工特征/知识的限制。如果学习算法是一个足够大的神经网络，且输入进去许多的训练数据，就有可能做得更好，甚至达到最优错误率。

端到端深度学习

- **流水线组件的选择：数据可用性**

在构建非端到端的流水线系统时，什么样的流水线组件可作为合适的选项呢？数据可用性！

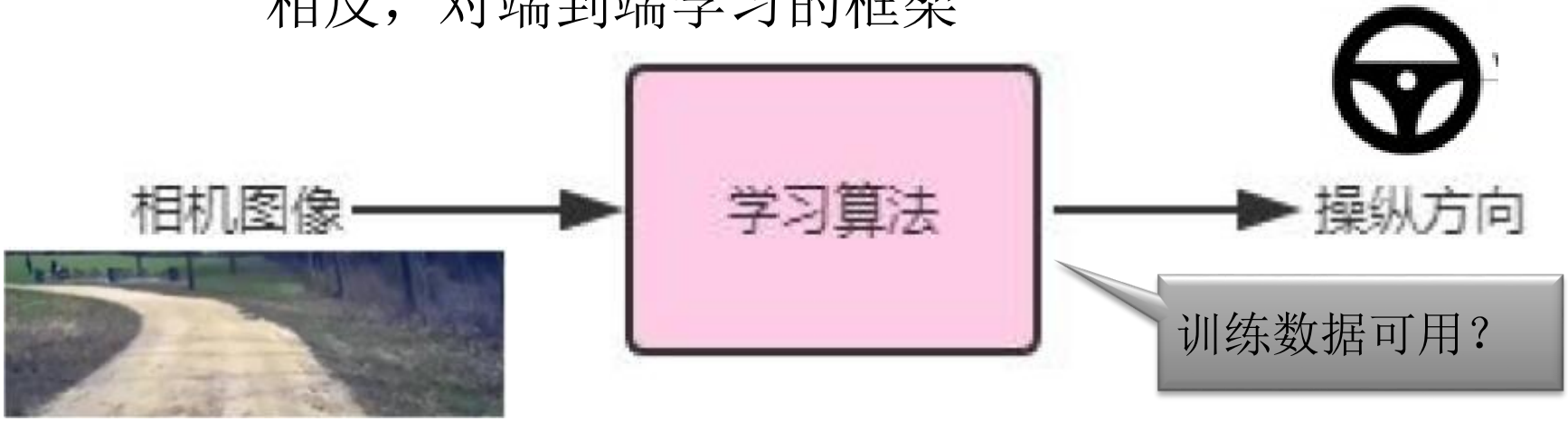
例如，考虑下面这个自动驾驶架构：



端到端深度学习

- **流水线组件的选择：数据可用性**

  相反，对端到端学习的框架

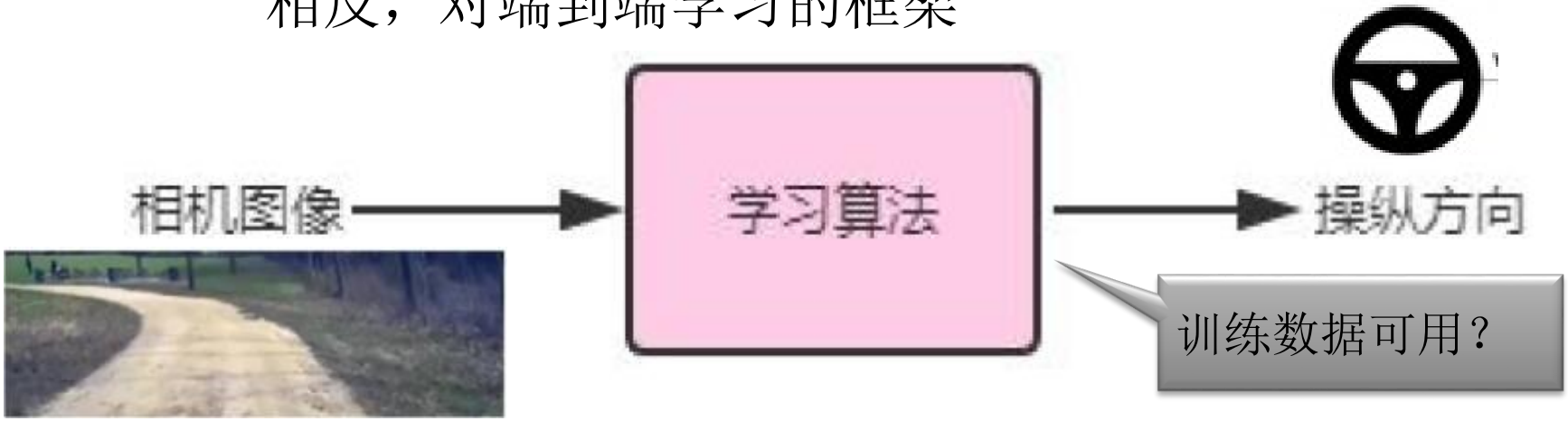


为了训练这样一个系统，我们需要一个包含 <图像，操纵方向> 数据对的大型数据集。然而让人们在驾驶汽车时收集汽车的操纵方向的数据是非常费时费力的，你需要一辆特殊配置的汽车，且需要巨大的驾驶量来涵盖各种可能的场景。这就使得端到端系统难以进行训练，获得大量带标记的行人或者是汽车图像反而要容易得多。

端到端深度学习

- **流水线组件的选择：数据可用性**

  相反，对端到端学习的框架



所以作者认为：“在更多端到端数据变得可用之前，我相信非端到端的方法对于自动驾驶而言是更有希望的——它的体系架构更匹配于数据的可用性。”

端到端深度学习

- **流水线组件的选择：任务简单性**

除了数据可用性之外，你还应该考虑流水线组件选择的第二个因素，独立的组件使得任务简单了多少。

考虑下面列出的机器学习任务，它们的难度逐级递增↑：

1. 分类判断图片是否过度曝光；
2. 分类判断图片拍摄于室内还是室外；
3. 分类判断图片中是否有猫；
4. 分类判断图片中是否有黑白两色皮毛的猫；
5. 分类判断图片中是否有暹罗猫（特殊的猫）

端到端深度学习

- **流水线组件的选择：任务简单性**

除了数据可用性之外，你还应该考虑流水线组件选择的第二个因素，独立的组件使得任务简单了多少。

考虑下面列出的机器学习任务，它们的难度逐级递增↑：

的猫；

的猫）

端到端深度学习

- **流水线组件的选择：任务简单性**

如果你能够完成一个复杂的任务，并将其分解为更简单的子任务，然后**显式编写子任务步骤代码（即任务的输入输出定义十分明确）**，那么你就会给算法一些先验知识，从而帮助它更有效地学习任务。

端到端深度学习

- **流水线组件的选择：任务简单性**

假设你正在构建一个暹罗猫检测器，选用下面哪种？

端到端学习方式：

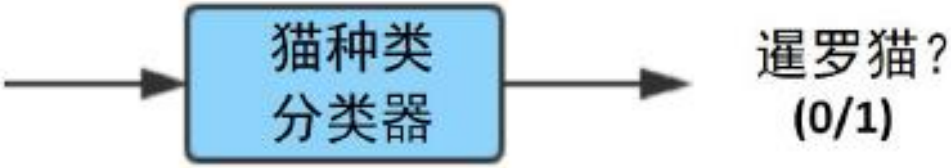
流水线组件方式：

第一步：（猫检测器）检测图像中所有的猫。

第二步：将每一块被检测到的猫的图像传送给猫种类分类器（每次一张），如果其中任何一只猫是暹罗猫，则在最后输出 1.

端到端深度学习

- 流水线组件的选择：任务简单性



端到端深度学习

- **流水线组件的选择：任务简单性**

最后让我们回顾一下自动驾驶流水线：

通过使用该流水线架构，你可以告诉算法总共有三个关键的步骤：

（1）检测其他车辆，

（2）检测行人，

（3）为你的车规划一条道路。

以上，每一个步骤都是相对简单的功能——因此可以用更少的数据来学习——而不是纯粹的端到端方法。

总而言之，当决定流水线组件的内容组成时，试着构建这样的流水线，其中每个组件都是一个相对 “简单” 的功能，因此只需要从少量的数据中学习。

Lectures contents

- 设置开发集与测试集
- 基础误差分析
- 偏差与方差
- 学习曲线

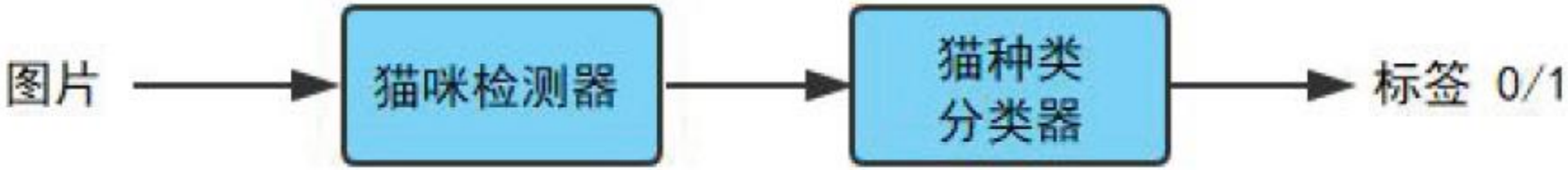
- 与人类表现水平对比
- 在不同的分布上训练与测试
- 端到端深度学习
- **根据组件进行误差分析**

根据组件进行误差分析

- 当有多个组件时，如何进行误差分析？

假设你的系统由复杂的机器学习流水线所构建，并且你希望提高该系统的性能，那应该从流水线的哪一部分开始改进呢？

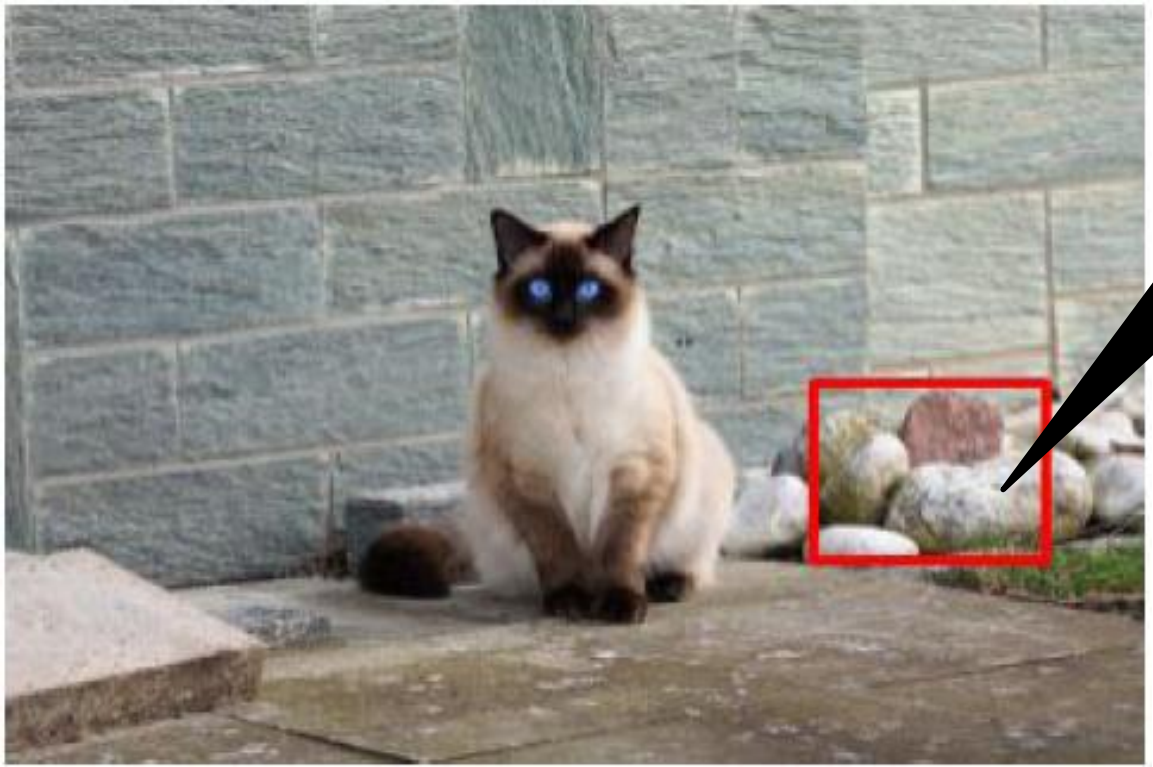
我们使用暹罗猫分类器的例子来进行说明：



在上图的流水线中，第一部分是猫检测器，它能够检测出猫，并将它们从图像裁剪出来；第二部分是猫的品种分类器，决定它是否是暹罗猫。 改进两个组件中的任何一个都有可能花费数年的时间……

根据组件进行误差分析

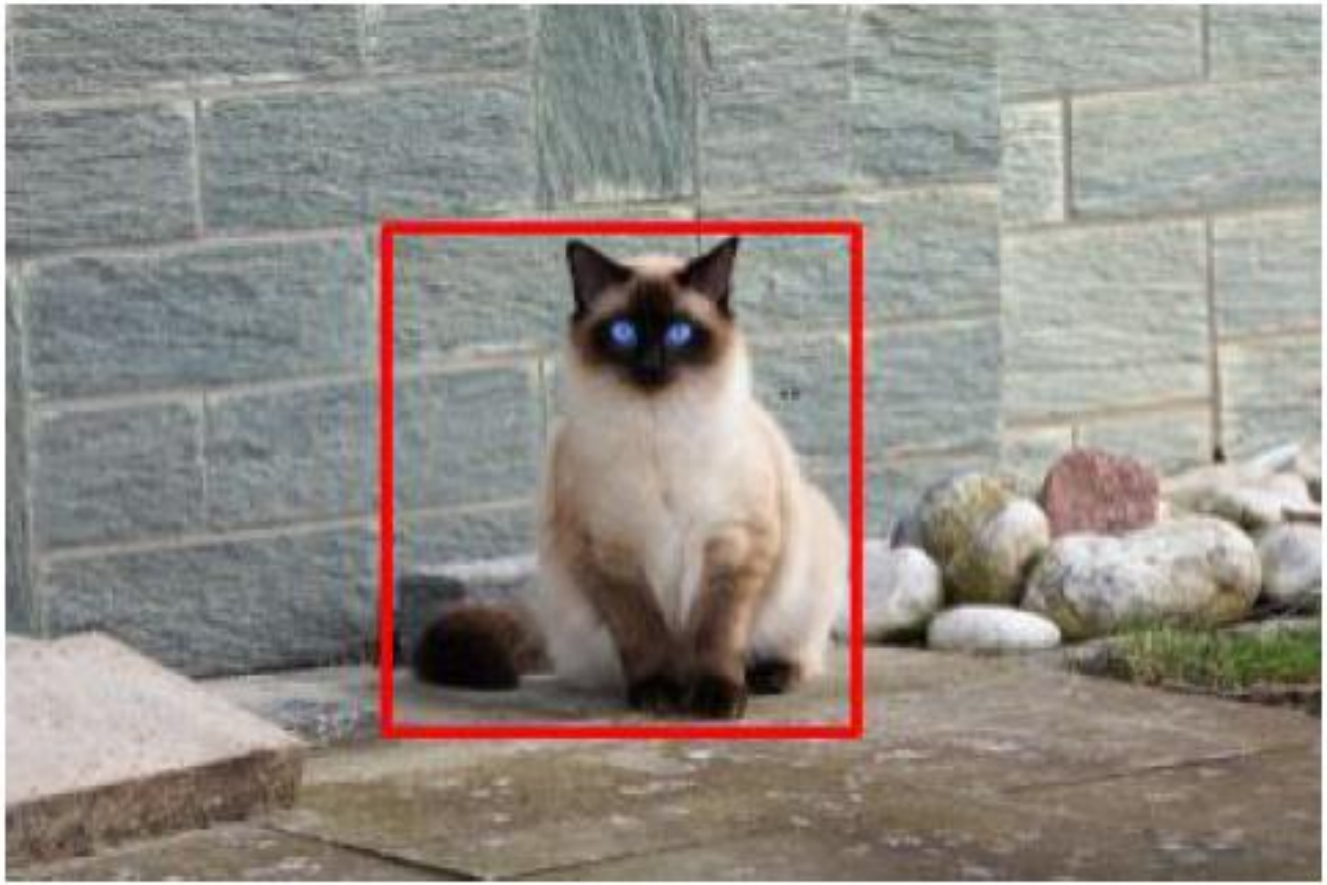
通过“按组件进行误差分析”，你可以尝试将每一个算法造成的误差<u>归因于</u>流水线的某个（有时是两个）组件。例如对如下某张算法判断错误的图像，让我们人为地检查一下算法两个步骤的执行过程。假设暹罗猫检测器从下图中检测出一只猫↓



这表示猫检测器给出了这样的图片

根据组件进行误差分析

另一方面，如果猫探测器输出了以下边界框：

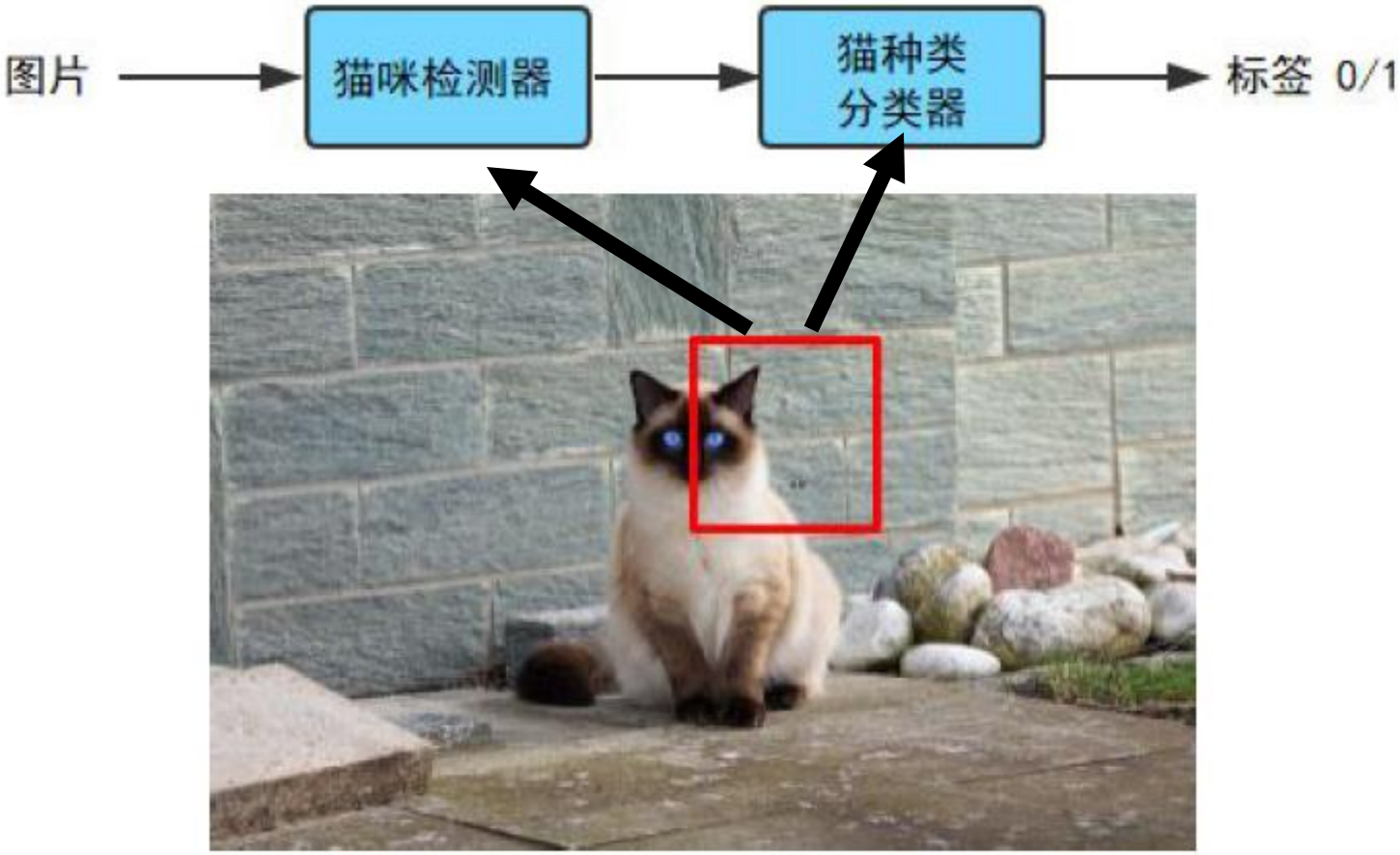


你会得出如下结论，猫探测器已经完成了它的工作，有缺陷应该是猫品种分类器。

根据组件进行误差分析

- **误差归因至某个组件**

让我们继续使用这个例子：



根据组件进行误差分析

- **误差归因至某个组件**

  1. 用手动标记的边界框替换猫检测器的输出。
  2. 通过猫品种分类器处理相应的裁剪图像。

如果猫品种分类器仍将其错误地分类，则将误差归因于猫品种分类器。否则，将此误差归因于猫检测器。

换而言之，进行一个实验，在其中为猫品种分类器提供“完美”输入。有两种情况：

情况1：即使给出了一个“完美”的边界框，猫品种分类器仍然错误地输出。在这种情况下，猫品种分类器很明显存在着问题。

情况2：给定一个“完美”的边界框，品种分类器现在正确输出。这表明只要猫检测器给出了一个更完美的边界框，那么整个系统的输出就是正确的。因此，将误差归因于猫检测器。

根据组件进行误差分析

- **误差归因的一般情况**

以下是误差归因的一般步骤。假设在流水线中有三个步骤 A，B 和 C，其中 A 直接输出到 B，B直接输出到 C。

根据组件进行误差分析

- **误差归因的一般情况**

对于系统在开发集上存在的每个错误样本

1. 尝试人为修改 A 的输出为“完美” 输出（例如，猫的“完美”边界框），并在此输出上运行流水线其余的 B，C 部分。 如果算法现在给出了正确的输出，那么这表明，只要 A 给出了更好的输出，那么整个算法的输出就是正确的；因此，你可以将此误差归因于组件 A。否则，请继续执行步骤 2。

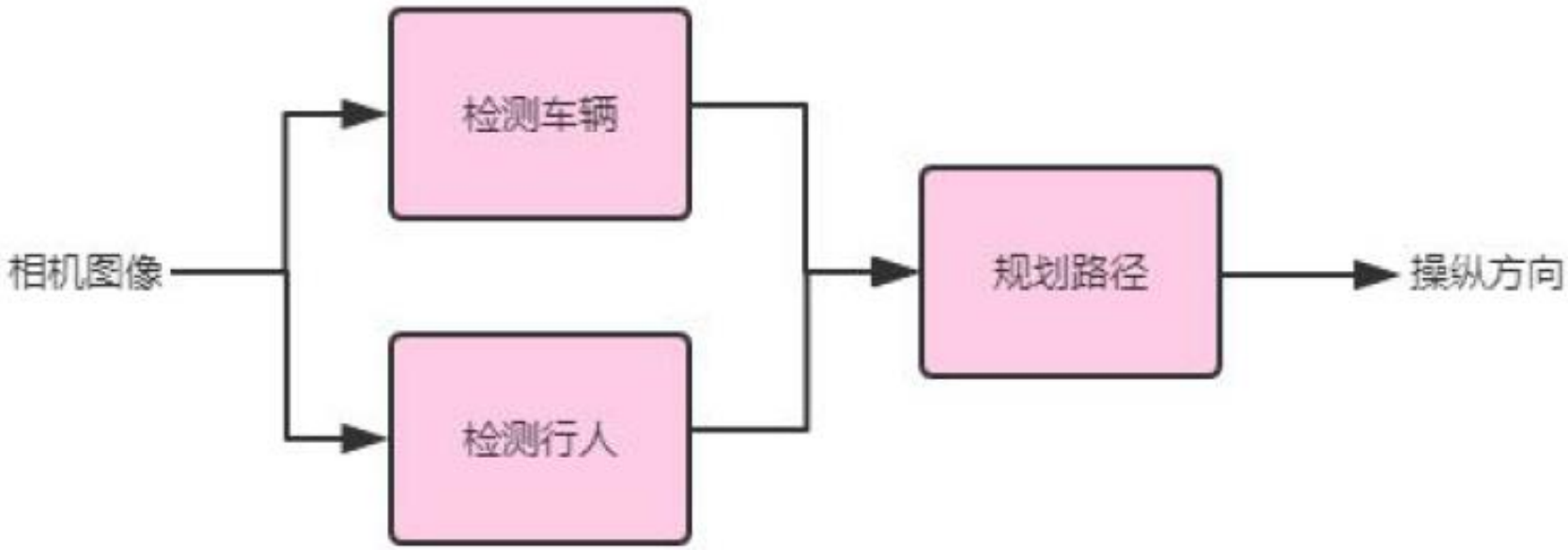
2. 尝试人为修改 B 的输出为 “完美” 输出。如果算法现在给出正确的输出，则将误差归因于组件 B. 否则，继续执行步骤 3。

3. 将误差归因于组件 C。

根据组件进行误差分析

- **误差归因的一般情况**

你的自动驾驶汽车将使用上面的流水线技术。如何根据组件进行误差分析来决定专注于哪个（些）组件呢？



根据组件进行误差分析

- **误差归因的一般情况**

  误差归因至某个组件的最终结果……

组件名称

图像	A	B	C	D	备注
…					
98				✔	标注者忽略了背景中的猫
99		✔			
100				✔	猫的画像；非真猫
占全体比例	8%	43%	61%	6%	

根据组件进行误差分析

- **发现有缺陷的机器学习流水线**

如果你的机器学习流水线的每个单独组件在人类水平性能或接近人类水平性能上执行，但总体流水线性能却远远低于人类水平会怎么样？这通常意味着流水线存在缺陷，需要重新设计。

换言之，误差分析还可以帮助你了解是否需要重新设计流水线。

根据组件进行误差分析

- **发现有缺陷的机器学习流水线**

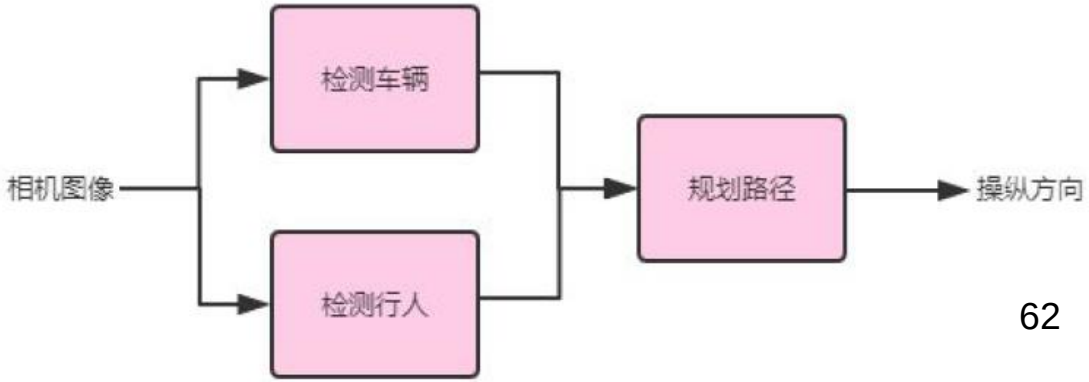
例如说：

1. 汽车检测部件（大概）是人类级别的性能，用于从摄像机图像中检测汽车。

2. 行人检测组件（大概）是人类级别的性能，用于从摄像机图像中检测行人。

3. 与仅根据前两个流水线组件的输出（而不是访问摄像机图像）规划汽车路径的人相比，路径规划组件的性能处于类似水平。

然而，你的自动驾驶汽车的整体性能远远低于人类水平



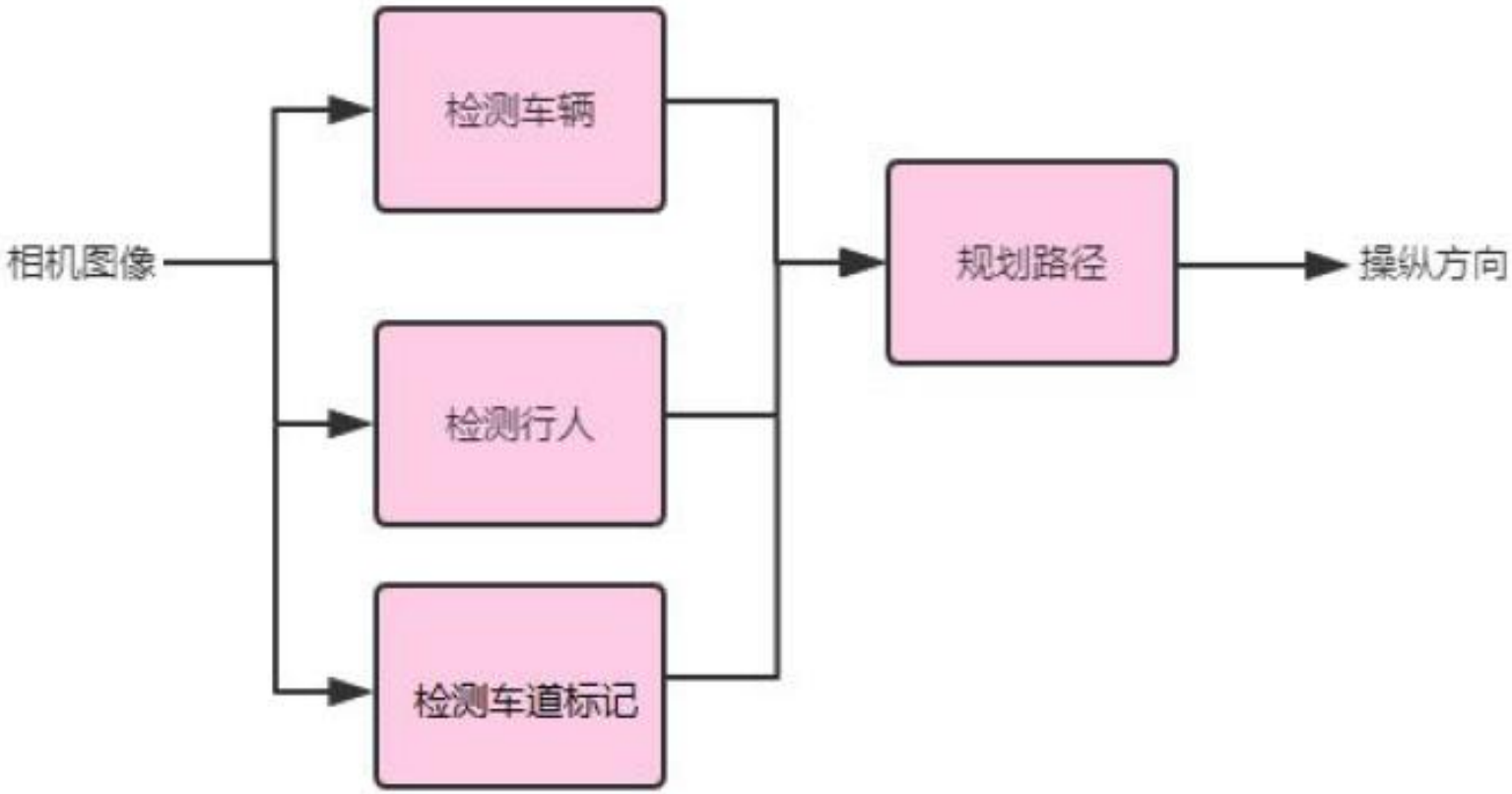
根据组件进行误差分析

- **发现有缺陷的机器学习流水线**

唯一可能的结论是， 流水线存在着缺陷。在这种情况下，路径规划组件在给定输出的情况下可以做得很好，但输入没能包含足够的信息。你应该询问自己，除了两个早期流水线组件的输出之外，还需要哪些其他信息来为汽车的驾驶辅助规划路径。换句话说，熟练的人类驾驶员还需要什么其他信息？例如，假设你意识到人类驾驶员还需要知道**车道标记的位置**。这表明你应该按如下方式重新设计流水线。

根据组件进行误差分析

- 发现有缺陷的机器学习流水线



小结

- **本堂小结**

- **使用人工合成数据时需谨慎**
- **端到端学习的优缺点：**
  - 优：方法直接，输入即得输出，无需手工设计中间环节
  - 缺：数据驱动，依赖较大量的训练数据与算力
- **流水线组件的选择需考虑两方面：**
  - 数据可用性、任务简单性
- **如何将误差归因至某个组件**

小结

Congratulations on finishing this book!

In Chapter 2, we talked about how this book can help you become the superhero of your team.

The only thing better than being a superhero is being part of a superhero team. I hope you'll give copies of this book to your friends and teammates and help create other superheroes!

End of Unit 2