白皮书

使用 MATLAB 实现信号处理的深度学习

MathWorks®

简介

深度学习网络用于图像分类已长达数年，但它们其实也是强大的数字信号处理工具。深度学习网络具备数学模型的一切功能，而您无需知道要将哪些信号特征引入模型。在训练过程中，网络会自行决定需要引入的特征。

无论输入数据是一维信号、时序数据还是文本，深度学习网络（如卷积神经网络 (CNN)）都能采用全新的方式处理数据。它们为非专业人士敞开了大门 — 不必成为信号处理专家，也能通过 CNN 处理信号 — 而且可以极为迅速地得出准确的结果。

本白皮书将回顾一些深度学习基础知识，然后会讲解三个信号处理示例：

- 语音命令识别
- 剩余使用寿命 (RUL) 预估
- 信号去噪

这些示例将向您展示如何通过 MATLAB® 实现深度学习来加速执行信号处理任务并获得更准确的结果。

深度学习基础

深度学习是机器学习的一个子类型。在该类型中，模型直接从图像、文本或声音中学习执行分类和回归任务。采用机器学习或传统信号处理技术时，您需要手动选择相关数据特征。而在采用深度学习的情况下，模型会在数据流经网络时自动学习并提取相关信息。

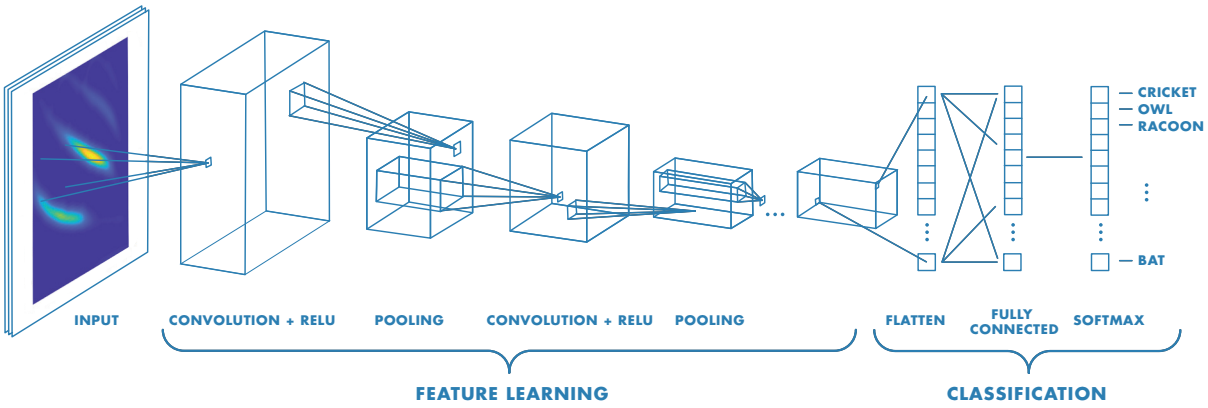
深度学习通常使用神经网络架构来实现。“深度”一词是指网络中的层数 — 层数越多，网络越深。各层通过节点或神经元相互连接，每个隐藏层使用前一层的输出作为其输入。

深度学习网络

现阶段最热门的两种深度学习网络分别是：

- 卷积神经网络（CNN 或 ConvNet）
- 长短期记忆 (LSTM)

CNN 由一个输入层、一个输出层和中间的多个隐藏层组成。卷积层、激活（或 ReLU）层和池化层是最常见的三层结构。这些操作在几十层或几百层上反复进行，每一层都学习检测不同的输入数据特征。



CNN 架构包含卷积层、ReLU 层和池化层等多个常用层次结构。

LSTM 是一种循环神经网络 (RNN)，可学习时间步长序列数据之间的长期依赖关系。与 CNN 不同，LSTM 可以记住预测之间的网络状态。

LSTM 网络的核心组件是序列输入层和 LSTM 层。序列输入层将时序数据加入网络。LSTM 层会学习一段时间内序列数据时间步长之间的长期依赖关系。

选择网络

CNN 通常用于信号和时序预测，此时输入信号从一维转换为二维表示，信号随之转换为“图像”。

LSTM 适用于序列和时序数据分类，此时必须基于记忆的数据点序列进行网络预测或输出。

信号数据注意事项

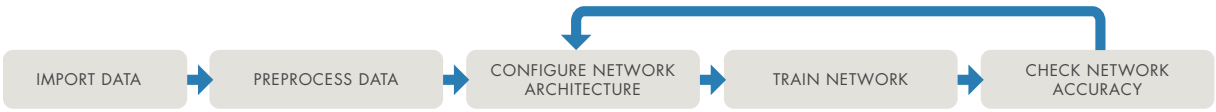
相较于使用图像数据进行深度学习，通过信号数据实现深度学习通常需要完成更多的预处理和特征提取。通常，您可以将原始图像输入到神经网络并获得结果，但是这一工作流程几乎不可能适用于处理信号。大部分原始信号数据嘈杂、多变而且比图像数据小。小心谨慎地对可用数据进行预处理可以确保得到尽可能理想的模型。

是否应该改用机器学习？

如果您理解数据但数据量有限，则使用机器学习并手动提取最相关的特征可能会获得更好的结果。尽管结果令人印象深刻，但相较于深度学习，这种方法需要掌握更多的信号处理知识。

工作流程

深度学习工作流程的主要步骤如下:



无论采用哪一种网络配置，也不论利用深度学习解决哪一类问题，流程始终要不断迭代：如果网络未达到预期的准确率，请返回更改设置，看看是否有方法加以改进。

以下每一个示例均遵循这项基本工作流程。

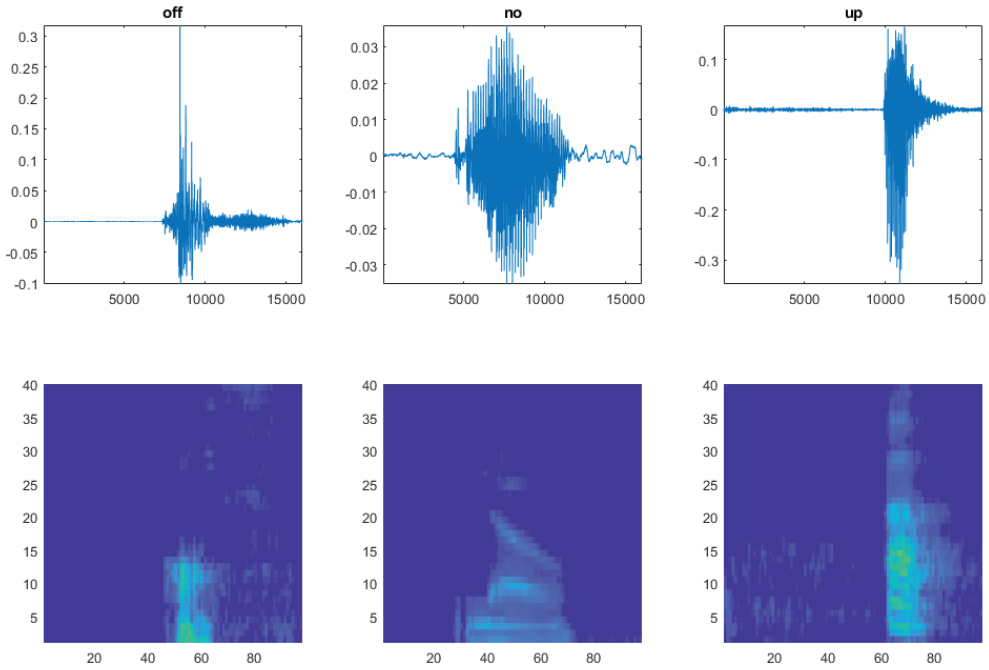
示例 1.语音命令识别：语音文件分类

我们希望训练一个简单的深度学习模型来识别并确定录音中的各项语音命令。

在经典的信号处理过程中，我们需要从信号中识别并提取相关特征（识别语音的信号分量），再将特征传递给解码器模型。小心谨慎地进行信号预处理（例如，降低背景噪音或混响）并选择一组特定特征，仍有可能生成不良结果，除非整个系统巧妙适应了现存的问题。

采用深度学习后，尽管仍然需要从信号中提取相关语音特征，但之后便可以将问题作为图像分类问题加以解决。首先将音频文件转换为频谱图 — 鉴于频谱图是一维音频文件的二维信号可视化效果，因而可以将它用作 CNN 输入，如同使用实际图像一样。

结果将得到一个支持处理信号细微差别的稳健模型。 若不采用深度学习，信号处理工程师就得手动识别、发现及处理信号中的相关信息。



上图：原始语音信号。下图：语音信号对应的频谱图。

导入数据

本示例使用一种用于执行高性能数值计算的开源软件库 TensorFlow™ 中的数据集。

> 在 MATLAB 中同步学习：*获取完整示例和数据集链接*

准备数据

为了准备数据以便对 CNN 进行有效训练，我们使用 **`melSpectrogram`** 命令（MATLAB 中的一种比较简单的 spectrogram 函数变体）将语音波形转换为频谱图。语音处理是音频处理的一种特殊形式，具有以特定频率定位的特征（识别语音的信号分量）。由于我们想要 CNN 关注语音最相关的频率区域，所以我们使用 Mel 频率间隔，它基于人耳听觉特性分配敏感度。

频谱图只是将原始信号数据表示为二维图像的众多时频变换方法之一。小波变换是另一种常用技术。

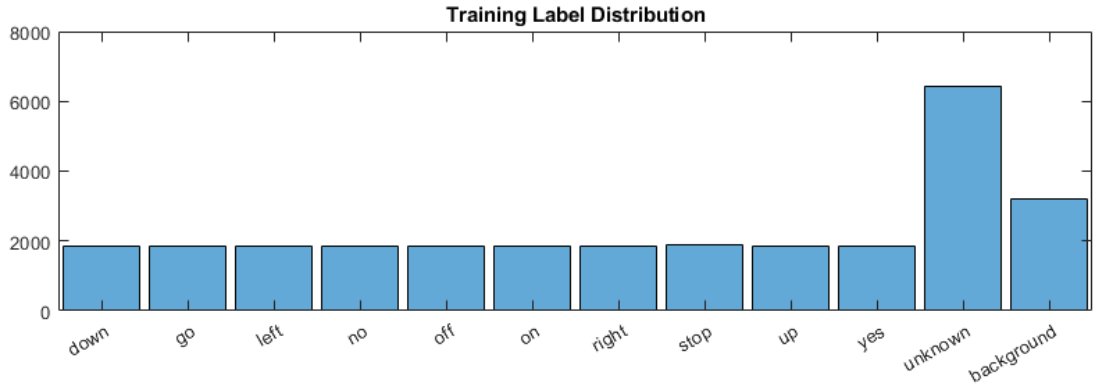
小波变换（如尺度谱和连续小波变换）可以生成类似频谱图的二维表示，但时间分辨率更高，非周期性信号分量表现尤为显著。

小波还广泛应用于名为*小波散射*或“不变散射卷积网络”的自动特征提取技术。这项技术可模仿在 CNN 前几层中学习的特征。但是，它们采用固定权重模式，无需从数据中开展学习，从而大大降低了网络复杂度及训练数据量要求。

接下来，将数据分为三组：

- **训练数据：**原始输入数据，网络将利用这些数据学习和理解特征
- **验证数据：**在网络训练过程中用于验证算法是否学习特征并归纳理解数据
- **测试数据：**网络从未遇到过的数据，经过训练后再将数据传递到网络，用于评估网络性能，同时确保结果无偏差

我们将训练数据均匀分布在要分类的文字类之间。



均匀分布的训练数据。

为了减少误报，我们包括了一个可能与预期类别混淆的文字类别。例如，如果目标文字是“on”，那么与“mom”、“dawn”和“won”类似的文字便放置在“未知”类别中。网络不需要识别这些文字，只要知道它们不是需要识别的目标文字即可。

配置网络架构

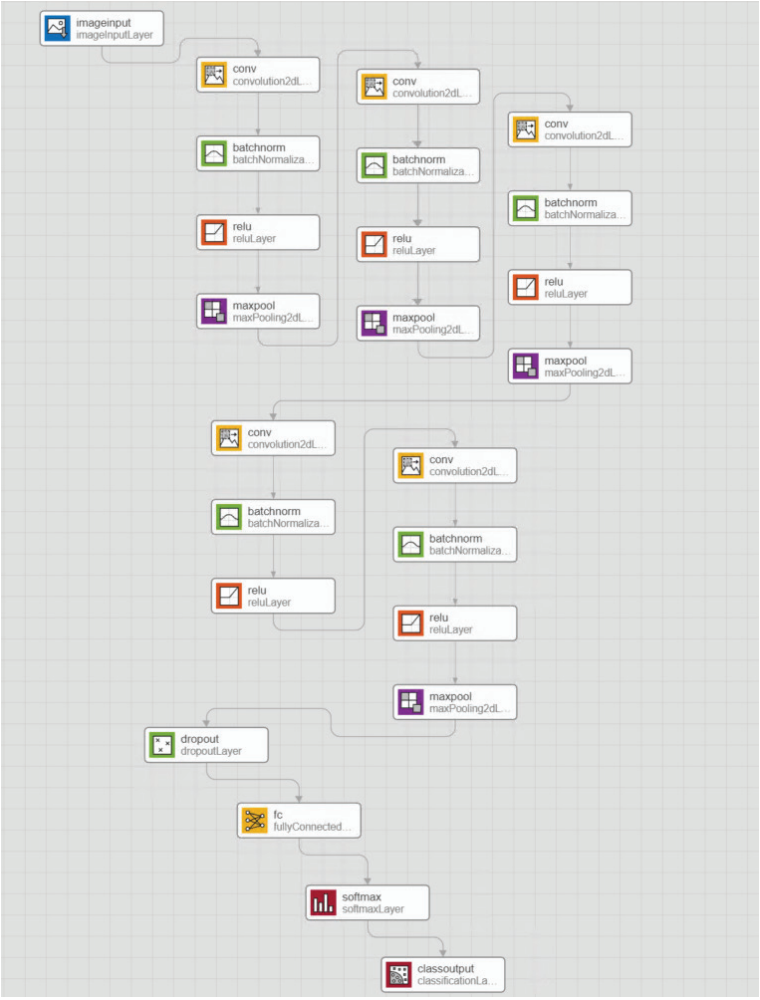
因为我们使用频谱图作为输入，所以 CNN 的结构可以与我们用于图像分类的其他网络相似。在此应用中，我们将使用一个 48 层的网络。

由 48 层构成的网络听上去很复杂，但基本结构很简单：不过是一系列反复出现的卷积层、批归一化层和 ReLU 层。

如何定义网络架构？

一种常见的方法是借鉴研究报告或其他已发布来源中的网络，然后根据需要对其进行修改。

整个模型可能如下所示：



网络设计者自行确定执行卷积、批归一化和 ReLU 的迭代次数；但必须谨记，网络越复杂，训练和测试网络所需的数据就越多。

训练网络

开始训练之前，先指定训练选项，如：

- 训练代数（完整传输训练数据）
- 学习速率（训练的速度）
- 处理器（通常是指 CPU 或 GPU）

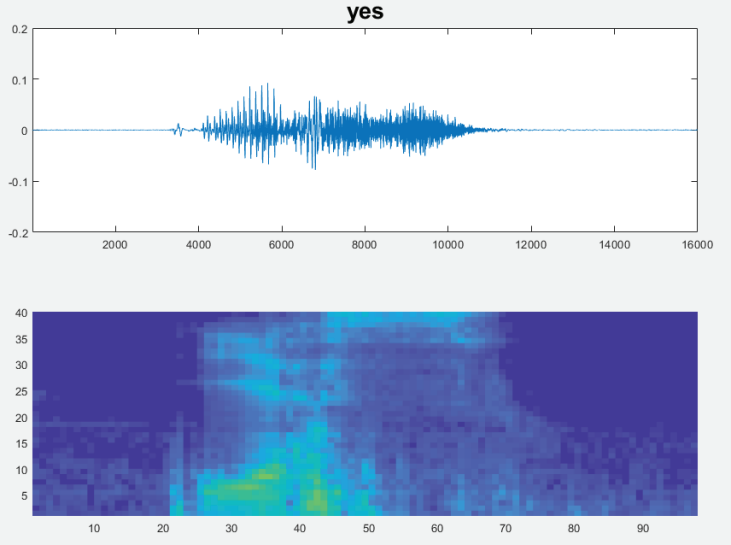
然后，我们将信号样本传递到网络中，一边训练一边绘制进度。

网络将学习识别这些样本中的独特特征并相应进行分类。处理频谱图数据时，学习过程实质上是图像分类过程：网络前几层学习光谱图中的普遍特征，如线条和形状的颜色、粗细及方向。随着训练的深入，网络会识别更精细的单个信号特征。

检查网络准确率

完成模型训练后，无论因为完成了我们指定的训练代数还是达到了指定的停止点，模型都会将输入图像（频谱图）归入适当的类别。

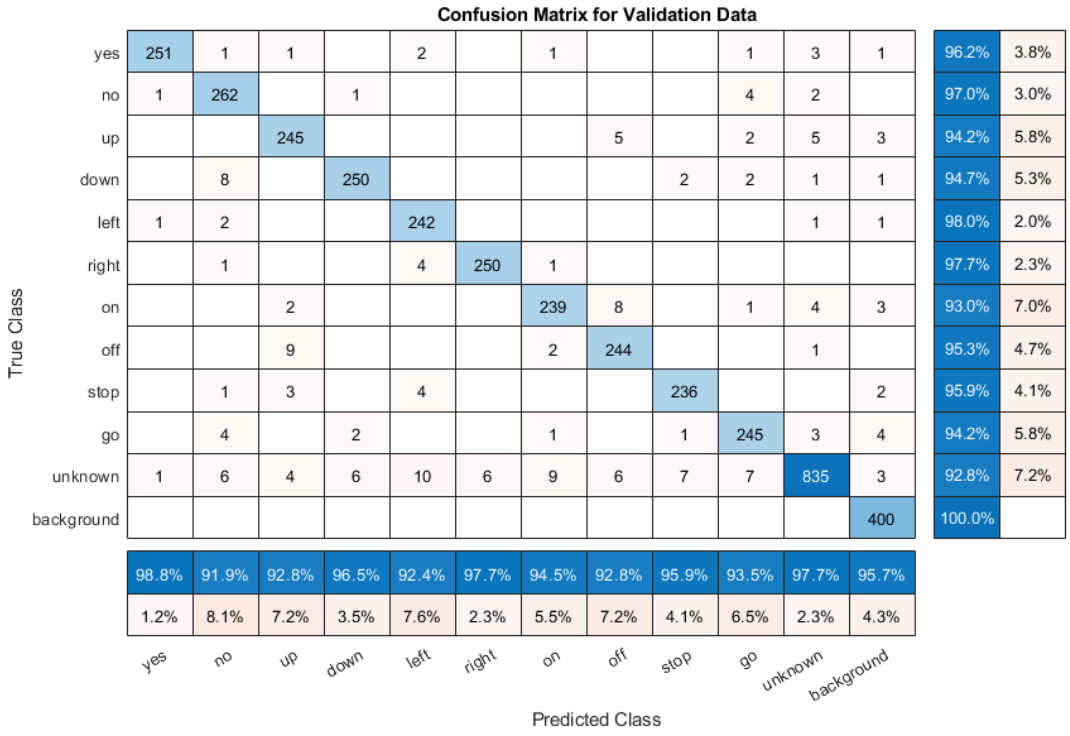
为了测试结果，我们将网络预测与对应的音频剪辑进行对比。验证集的准确率约为 96%。



归类为“yes”的文字的绘图和频谱图。

对于简单的语音识别任务而言，此结果或许可以接受；但是如果希望在安全系统或语音激活设备中实现该模型，则需要提高准确率。

这也是深度学习方法的真正价值所在：它为我们提供了多种准确率提升选项。我们只要将更多训练样本添加到网络中即可改善结果。为了深入了解网络存在哪些预测失误，我们可以对混淆矩阵进行可视化处理。



混淆矩阵显示了网络预测类（文字）与真实类。蓝色阴影表示正确预测，而米色则表示错误预测。

我们可以通过深度学习继续开展学习，直至达到预期准确率。采用传统信号处理技术时，必需掌握大量专业知识才能理解信号的细微差别。

示例 2.预测剩余使用寿命：序列到序列回归

在本例中，我们希望预测发动机再过多久会发生故障，也就是其*剩余使用寿命* (RUL)。RUL 是安排发动机维护时间并避免计划外延误的关键指标。

尽管很多深度学习网络预测的是一个类或类别，但我们希望输出数字（再经过多少个周期，发动机会发生故障），因而属于回归问题。

我们使用“运行故障”(run-to-failure) 数据计算 RUL，该数据显示传感器值随系统相对运行状况变化的趋势。

若不采用深度学习，将按以下工作流程解决此问题：

1. 选择哪些传感器是系统运行状况的最佳指示器。
2. 推导使用上述传感器组合的指标。
3. 在一段时间内持续跟踪数据，确定系统是否面临故障危险。
4. 拟合返回故障概率或故障时间的模型。

选择传感器（特征选择）是第一步，但这绝非易事。系统往往包含超过 50 个传感器，并非所有传感器都会在系统濒临故障时表现出压力迹象。采用深度学习，模型将为我们完成特征提取。网络会识别哪些传感器是最佳问题预测元。

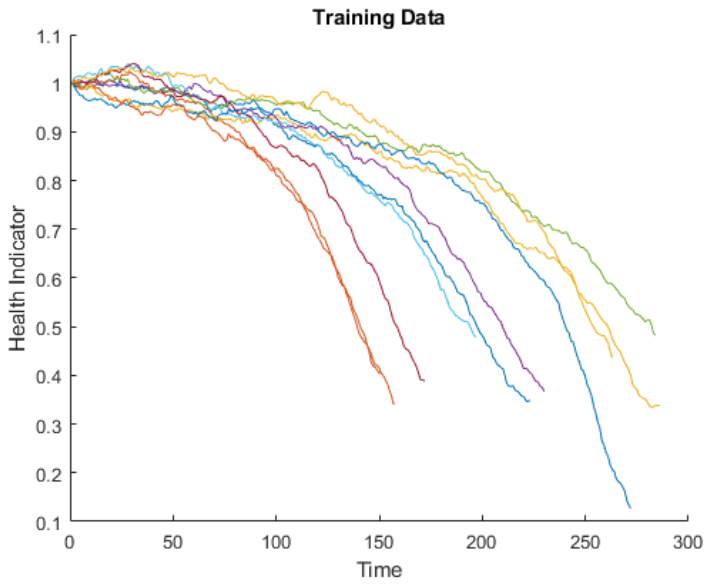
导入数据

首先从 NASA 预测数据存储库中的涡扇发动机退化仿真数据集开始。

> 在 MATLAB 中同步学习：*获取完整示例和数据集链接*

该数据集不仅包含训练和测试观测值，还包括用于验证的实际 RUL 值矢量。

每个时间序列代表一台发动机。各个发动机在时序开始时均正常运行，之后在该时序的某一时刻发生故障。在训练集中，该故障逐步升级为系统故障。



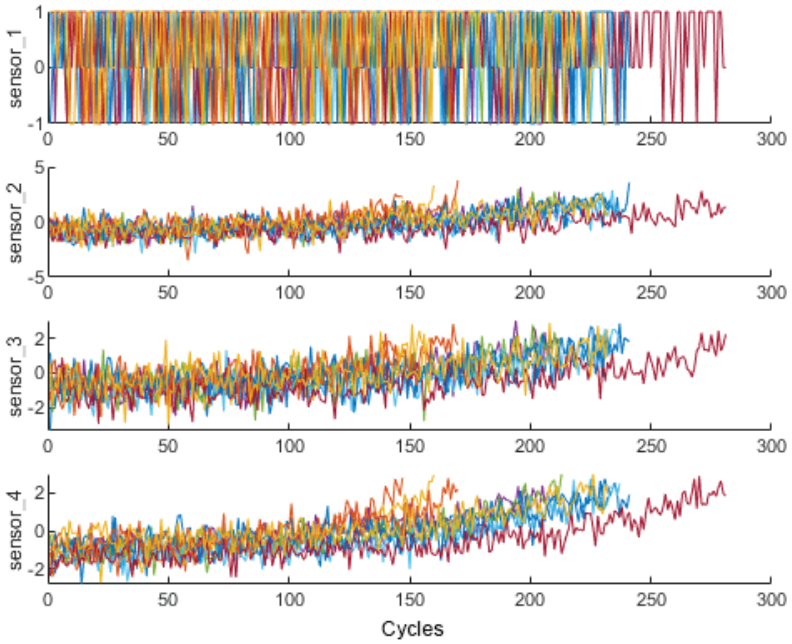
从 0 到 300 周期的训练数据的视图。

准备数据

为了确保获得准确的预测，我们希望为网络提供尽可能理想的待处理数据。要简化并清理数据，我们可以执行以下操作：

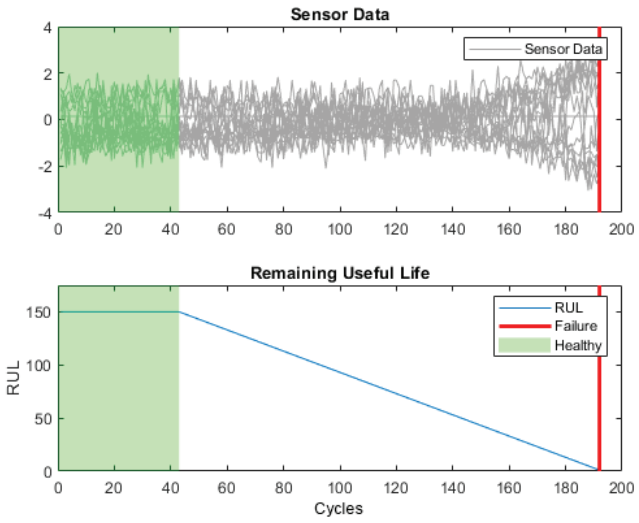
删除具有恒定值的信号。所有时间步长均保持恒定的信号可能对训练模型准确率没有帮助，因此可以删除最小值和最大值相同的信号。

训练预测元归一化。鉴于不同信号跨越的范围不同，我们需要对数据进行归一化。在本例中，我们假设零均值，且数值范围介于 -4 到 4 之间。为了计算所有观测值的平均值和标准方差，水平连接序列数据。



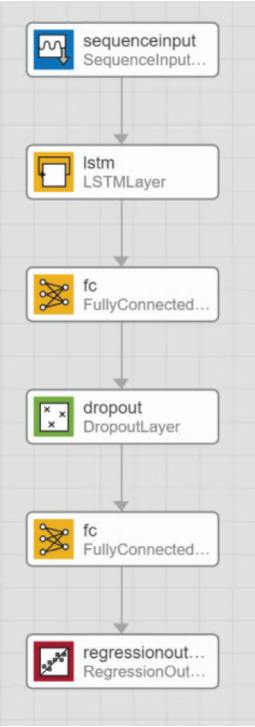
传感器 1-4 中的归一化数据。

裁剪响应数据。裁剪响应数据可以让网络重点关注系统开始出现故障的点。下图显示了一台发动机的传感器数据（上图）以及每一点的传感器剩余使用寿命数据（下图）。自发生故障起超过 150 个周期的信号初始值（显示为绿色）的剩余使用寿命上限值为 150，使得网络预测寿命留有一定余量。



上图：一台发动机的传感器数据。下图：各点传感器数据的 RUL。

配置网络架构



LSTM 十分适合此类应用，因为它可以学习时间序列数据之间的依赖关系。传感器通常相互连接，甚至两个周期前发生的事件也可能对未来系统性能造成影响。

我们定义了一个 LSTM 网络，该网络由一个包含 200 个隐藏单元的层与丢弃概率为 0.5 的 dropout 层构成。

丢弃概率是用于防止过拟合的工具。网络使用该值随机跳过一些数据，避免记忆训练集。隐藏单元往往多达数百个。确定要包含的隐藏单元数是一个折衷过程。如果数量过少，模型将不具备开展学习的充足内存。如果数量过多，网络可能会过拟合。

训练网络

首先设置两个训练选项：

- 训练代数设为 60，小批量大小设为 20（使用求解器 `'adam'`）
- 学习速率设为 0.01

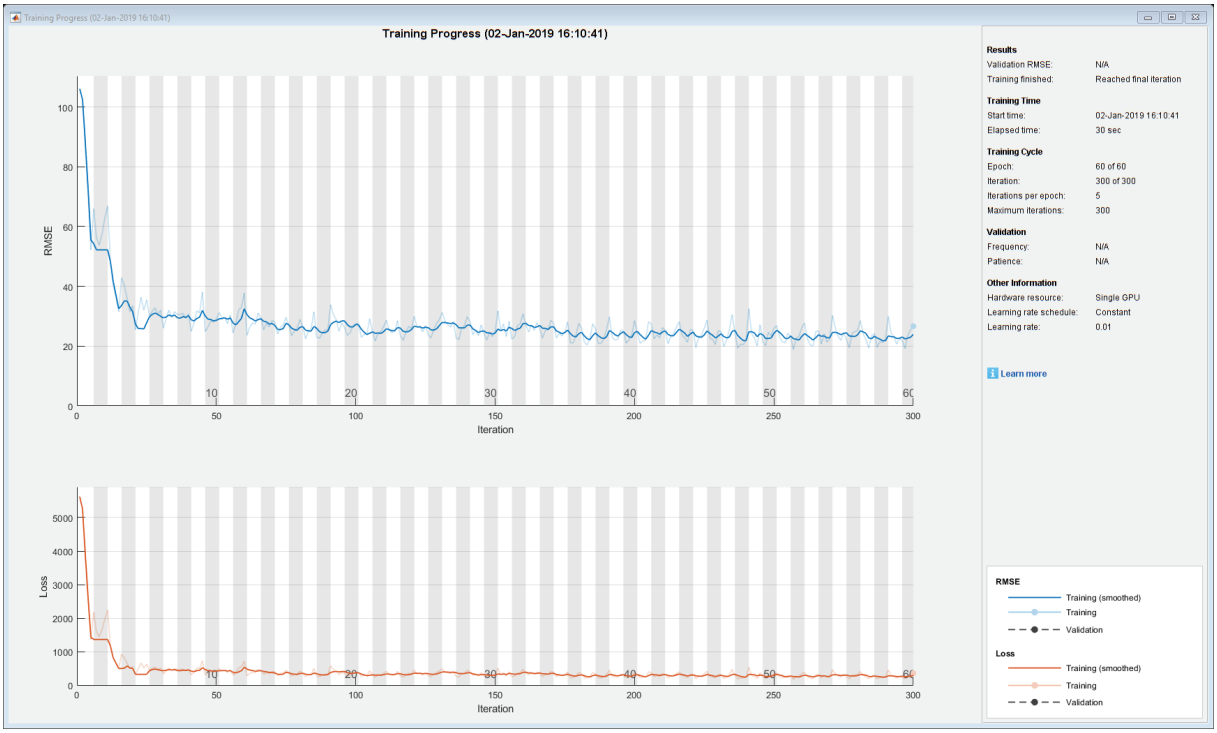
使用这些设置，训练网络大概需要两分钟。在网络运行期间绘制训练进度。

什么是求解器？

求解器是用于优化神经网络的算法。

`'adam'`（自适应矩预估）是最常与 LSTM 网络搭配使用的求解器，通常可以采用默认设置。`'adam'` 用于记录按元素计的参数梯度及其均方值的移动平均值。

*获取 'adam' 及*其他求解器（包括 `'sgdm'` 和 `'rmsprop'`）的更多信息。



RUL 模型的训练结果。

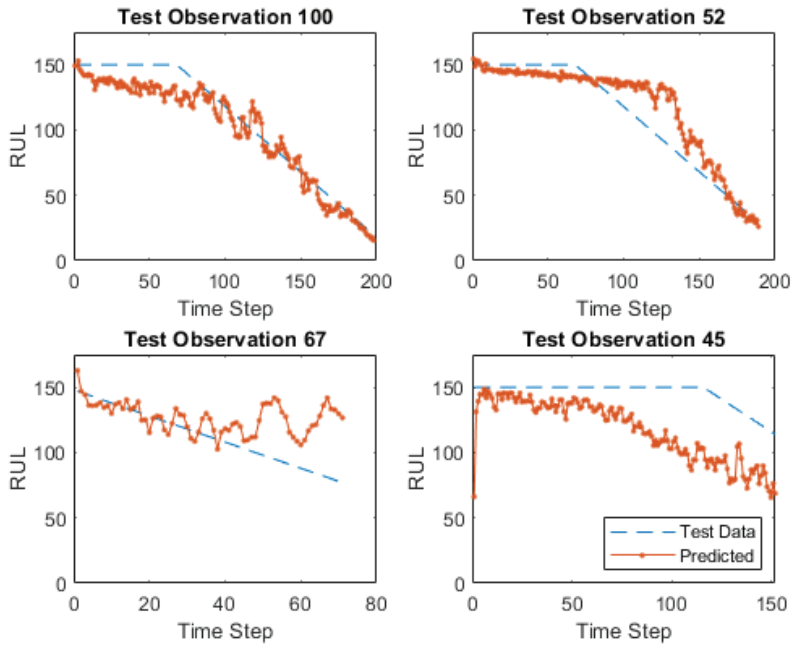
由于预测 RUL 属于回归问题，因此第一个图显示的是均方根误差 (RMSE) 序列，而不是准确率序列。数值越小，预计故障时间越接近实际值。

检查网络准确率

训练网络后，我们将根据测试数据对其进行验证。

LSTM 网络对部分序列做出预测，每次预测一个时间步长。在每一周期，网络都会更新其内部状态并预测故障时间长度 — RUL 值。

为了监控网络性能，我们选择四台测试发动机的数据，并根据预测绘制真实剩余使用寿命。



针对四台发动机（100、52、67 和 45）进行预测建模，比较实际 RUL（蓝色虚线）与预估 RUL（橙色线）。

LSTM 对发动机 100 和 45 的故障时间预估相当准确，但对发动机 52 和 67 的预估则不那么准确。该结果可能表明，在那些情况下，我们需要增加代表发动机性能的训练数据。或者也可以调整网络训练设置，看看性能是否有所改善。

在本例中，我们设计、训练并测试了一个简单的 LSTM 网络，并且使用该网络对剩余使用寿命进行了预测。通过使用深度学习代替传统信号处理技术，免去了艰巨而又耗时的特征提取任务。

示例 3.使用全连接神经网络实现语音去噪

我们希望消除语音信号中的洗衣机噪音，同时提高信号的质量和清晰度。工程师对信号进行去噪以提高数据质量 — 例如，向基于云的语音助手引擎传递更清晰的录音或者提高 ECG 的信噪比。

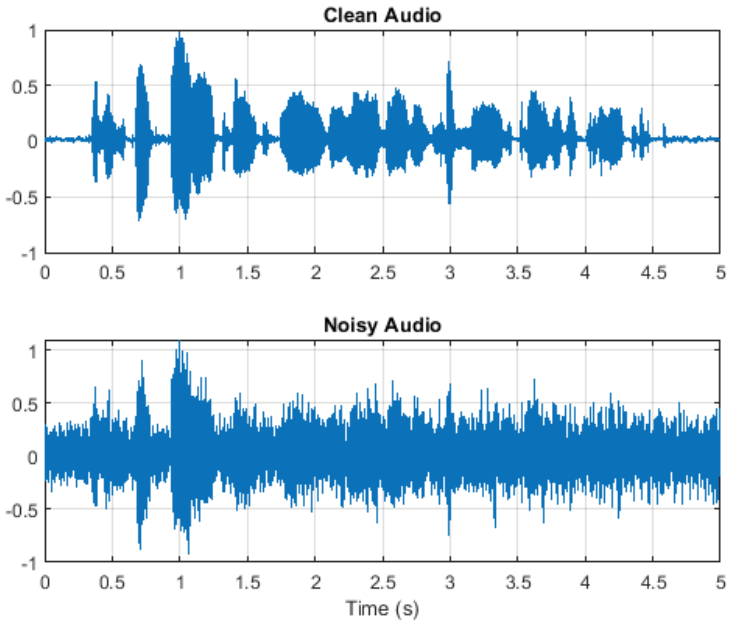
经典信号处理技术早已（并且仍将）用于过滤噪音。这些方法（谱减法、噪音门等）面临的挑战在于，良好信号的质量有时也会被无意中过滤掉，继而导致某些原始信号信息丢失。

在本例中，通过使用深度学习进行语音去噪，将能够消除语音信号中的洗衣机噪音，同时最大限度地弱化输出语音的意外失真现象。网络输出之后可用于过滤其他类似噪音环境下的噪音。

导入数据

我们读入干净的语音信号，然后使用这个干净的信号生成噪音信号。为此，需将预先录制的 wav 文件中的洗衣机噪音添加到该信号中。这样，只需比较真实干净的信号与最终去噪信号，即可按照此过程了解算法的效果。

在 MATLAB 中同步学习：*获取示例文件和完整示例*



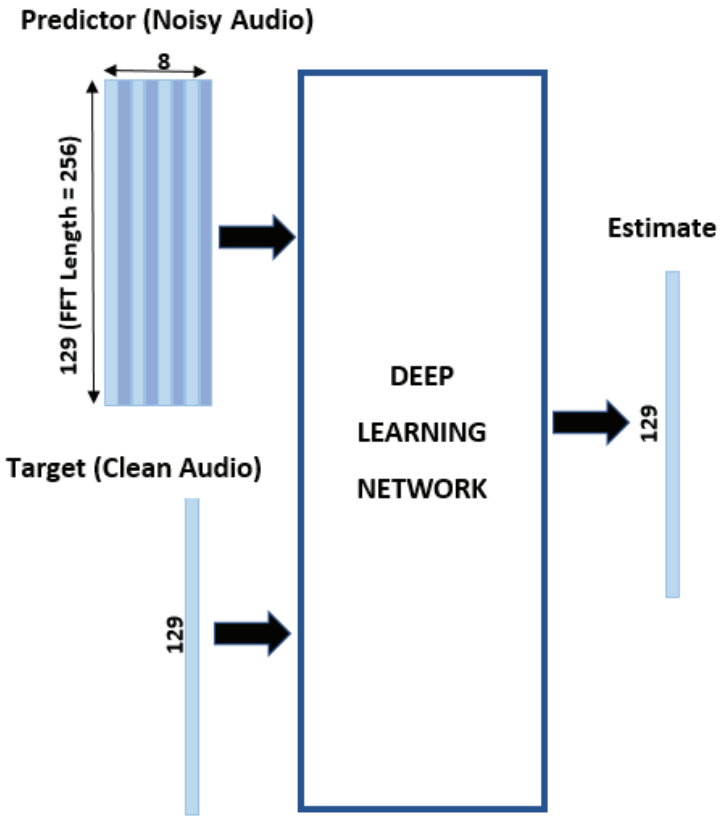
原始语音图和有噪语音图。

准备数据

我们准备了一组有噪语音示例，用于测试经过训练的算法。

在本例中，我们使用短时傅立叶变换 (STFT) 将信号转换为二维信号。该技术包括两部分：首先将信号分割为多个重叠片段，然后计算各个片段的快速傅立叶变换 (FFT)。流信号分割为若干片段后，将可以在获取信号时查看预测。输入时，我们累积了八个片段，以便及时为网络提供一些“内存”— 因而不仅可以查看当前信号段，还能查看与当前信号密切相关的一些过往信号。输出时，要求网络每次预测一个片段。

整个过程如下图所示。



预测元输入由八个连续的有噪 STFT 矢量构成，因此每个 STFT 的输出估计值根据当前有噪 STFT 和用作学习背景的七个过往有噪 STFT 矢量计算得来。

STFT 目标和预测元

使用短时傅立叶变换 (STFT) 将音频转换为频域时，需要设置一些系统参数。

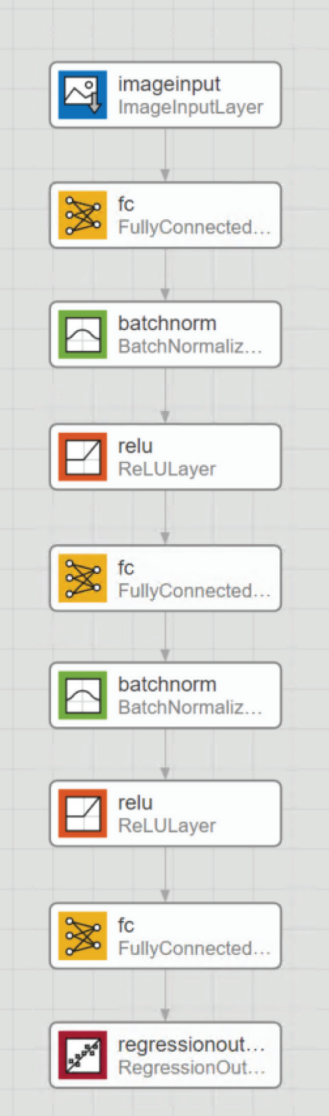
窗口长度。窗口越长，分辨率和复杂度越高。在给定可用数据的情况下，本例中的窗口长度 256 是对高频率分辨率和低计算复杂度进行合理折衷的结果。

重叠。我们将重叠率设置为 75%，使网络具有更精确的时间分辨率，以便在等量时间内通过网络传输更多信息。

输入采样率。该输入数据以 48 KHz 的速率录制。

频率采样率。将输入数据采样率降至 8KHz 足以满足人类语音需求，同时我们的网络也会变得更加紧凑。

信号段数。在本例中，我们选择了八个信号段，包括当前有噪 STFT 和网络将用作学习背景的七个过往信号段。由于每个信号段重叠率达 75%，因此大概相当于三个完整的时间段。



配置网络架构

我们将遵循与第一个示例相同的基本工作流程。虽然语音命令识别示例使用了一系列卷积层，但在本例中，我们使用的是全连接层。

全连接层与上一层的所有激活相连。全连接层将二维空间特征“扁平化”为一维矢量。

批归一化层将输出的平均和标准方差归一化。在训练阶段，随着参数或前几层的变化，当前层必须重新调整为新的分布，这需要一定的时间。为加快卷积神经网络的训练速度并降低对网络初始化的敏感度，我们在卷积层与非线性层之间使用了批归一化层，如 ReLU 层。

ReLU 层对每个输入元素执行阈值运算，所有小于零的值均设置为零。

最后以回归层结束，因而会产生半均方误差损失。回归层输出连续数据预测，如角度、距离或本例中的去噪信号。

训练网络

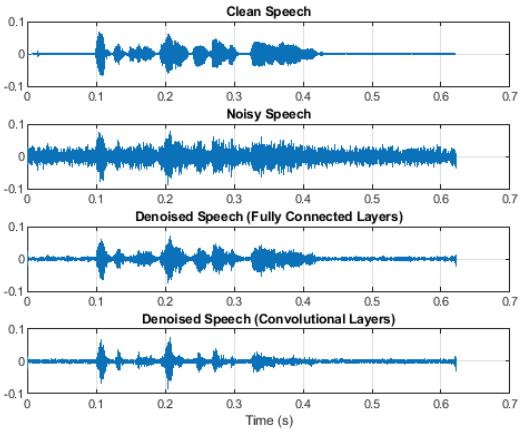
接下来，设置训练选项，如下所示：

- 最大训练代数设置为 3（网络将传输三次训练数据）
- 小批量大小设置为 128（网络每次将查看 128 个训练信号）
- `'Plots'` 设置为 `'training-progress'`（此设置将显示训练图）
- `'Shuffle'` 设置为 `'every-epoch'`
- `'LearnRateSchedule'` 设置为 `'piecewise'`（每经过一个训练代数，都将导致学习速率降低指定的系数 (0.9)）

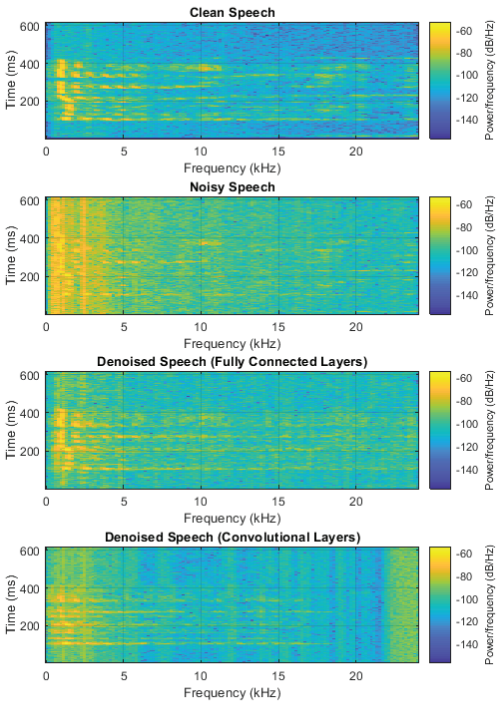
此类规模的训练集可能需要几分钟才能运行。

检查网络准确率

完成网络训练后，我们可以将留作测试的有噪信号传递到网络并可视化结果。我们可以通过绘图或频谱图显示信号。



时域图表明背景噪音水平已大幅降低。



频谱图显示了模型在不同频率下的性能的更多信息。例如，我们可以看到它在消除大部分语音频谱集中区域的低频率噪音方面特别有效。

由于是音频数据，您可以侦听语音以对质量进行主观对比，从而了解结果是否令人满意。

经过训练的网络现已准备就绪可供实时使用，我们最终得到一个可重用的精确模型，用于进行语音去噪。

在 MATLAB 中运行 **`speechDenoisingRealtimeApp`**，以仿真实时流版本去噪网络。

小结

信号处理深度学习不仅可以帮助工程师创建可重用的精确模型，还有助于降低提取和选择信号特征并通过更改来改善模型性能所需的专业知识要求。

在语音识别和信号去噪示例中，我们展示了如何准备及转换音频数据以便用在 CNN 中。我们对模型性能进行了可视化，并研究了如何在信号处理专业知识匮乏的情况下做出更改。

在 RUL 示例中，我们使用 LSTM 并对传感器数据进行预处理，以便尽量为网络提供最佳数据，无需手动提取和选择特征即创建了工作模型。

其他资源

使用 MATLAB 实现深度学习

信号和声音的深度学习