Gradio简介

预计阅读时间：15分钟

目标

完成本阅读后，您将能够：

- 解释 Gradio 的基础知识
- 演示在 PyTorch 中实现图像分类的示例

介绍

Gradio 是展示您的机器学习模型的方式，提供用户友好的网页界面，使每个人都可以在任何地方使用它。Gradio 是一个开源的 Python 包，允许您快速构建机器学习模型、API 或任何任意 Python 函数的演示或网页应用程序。然后，您可以使用 Gradio 内置的共享功能分享您的演示或网页应用程序的链接。无需 JavaScript、CSS 或网页托管经验。

为什么使用 Gradio?

Gradio 有几个有用的原因：

- 易用性：Gradio 只需几行代码即可创建模型接口。
- 灵活性：Gradio 支持多种输入和输出，如文本、图像、文件等。
- 分享与协作：接口可以通过唯一的 URL 与他人共享，便于轻松协作和收集反馈。

任务 1：开始使用 Gradio

要开始使用 Gradio，您首先需要安装该库。您可以使用 pip 安装 Gradio：

```
pip install gradio
```

创建你的第一个 Gradio 界面

你可以在你喜欢的代码编辑器、Jupyter notebook、Google Colab，或任何其他你编写 Python 的地方运行 Gradio。让我们来编写你的第一个 Gradio 应用：

```
import gradio as gr
def greet(name, intensity):
  return "Hello, " + name + "!" * int(intensity)
demo = gr.Interface(
  fn=greet,
  inputs=["text", "slider"],
  outputs=["text"],
)
demo.launch(server_name="127.0.0.1", server_port= 7860)
```

如果从文件运行，下面的演示将在浏览器中打开 http://127.0.0.1:7860。如果您在笔记本中运行，演示将嵌入在笔记本中。

在左侧的文本框中输入您的名字，拖动滑块，然后按提交按钮。您应该能在右侧看到一个友好的问候。

理解 Interface 类

请注意，为了制作您的第一个演示，您创建了 `gr.Interface` 类的一个实例。`Interface` 类旨在为接受一个或多个输入并返回一个或多个输出的机器学习模型创建演示。

`Interface` 类有三个核心参数：

- **fn:** 要包装用户界面的函数
- **inputs:** 用于输入的 Gradio 组件。组件的数量应与您函数中的参数数量相匹配。
- **outputs:** 用于输出的 Gradio 组件。组件的数量应与您函数中的返回值数量相匹配。
  fn 参数是灵活的——您可以传递任何您想用用户界面包装的 Python 函数。在上面的示例中，您看到的是一个相对简单的函数，但该函数可以是从音乐生成器到税务计算器，再到预训练机器学习模型的预测函数。

input 和 **output** 参数接受一个或多个 Gradio 组件。正如我们将看到的，Gradio 包含超过 30 个内置组件（如 `gr.Textbox()`、`gr.Image()` 和 `gr.HTML()` 组件），这些组件专为机器学习应用设计。

如果您的函数接受多个参数，如上所示，请将输入组件的列表传递给 **inputs**，每个输入组件应按顺序对应于函数的一个参数。如果您的函数返回多个值也是如此：只需将组件列表传递给 **outputs**。这种灵活性使得 `Interface` 类成为创建演示的非常强大的方式。
让我们为图像字幕模型创建一个简单的接口。BLIP（Bootstrapped Language Image Pretraining）模型可以为图像生成字幕。以下是如何为 BLIP 模型创建 Gradio 接口。

```
pip install transformers
pip install torch
```

```
import gradio as gr
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
def generate_caption(image):
    # Now directly using the PIL Image object
    inputs = processor(images=image, return_tensors="pt")
    outputs = model.generate(**inputs)
    caption = processor.decode(outputs[0], skip_special_tokens=True)
    return caption
def caption_image(image):
    """
    Takes a PIL Image input and returns a caption.
    """
    try:
        caption = generate_caption(image)
        return caption
    except Exception as e:
        return f"An error occurred: {str(e)}"
iface = gr.Interface(
    fn=caption_image,
    inputs=gr.Image(type="pil"),
    outputs="text",
    title="Image Captioning with BLIP",
    description="Upload an image to generate a caption."
```

```
)
iface.launch(server_name="127.0.0.1", server_port= 7860)
```

在这里，我们使用 `transformers` 库中的 `BlipProcessor` 和 `BlipForConditionalGeneration` 来设置图像字幕生成模型。这个示例演示了如何使用 Gradio 创建一个网页接口，其中输入参数指定一张图片，输出是生成的文本字幕。标题和描述参数通过提供上下文和用户指令来增强接口。

用例：图像描述

图像描述模型如 BLIP 是在多个领域中非常强大的工具，从帮助视觉障碍人士理解图像内容到高效地组织和搜索大型照片库。为这样的模型创建 Gradio 界面，使非技术用户能够与之互动并从这项技术中受益。例如，摄影师或数字资产管理人员可以使用您的应用程序自动生成图像的描述性名称，从而增强他们数字库的可用性和可搜索性。

任务 2：在 PyTorch 中进行图像分类

现在让我们探索另一种计算机视觉任务 — **图像分类**。图像分类是计算机视觉中的核心任务。构建更好的分类器以识别图像中存在的物体是一个活跃的研究领域，因为它在从自动驾驶汽车到医学成像等多个应用中都有广泛应用。

这样的模型非常适合与 Gradio 的图像输入组件一起使用。在本教程中，我们将构建一个使用 Gradio 进行图像分类的网络演示。我们可以用 Python 构建整个网络应用程序。

第一步：设置图像分类模型

首先，我们需要一个图像分类模型。在本教程中，我们将使用预训练的 Resnet-18 模型，因为它可以从 PyTorch Hub 轻松下载。您可以使用其他预训练模型或训练自己的模型。

```
import torch
model = torch.hub.load('pytorch/vision:v0.6.0', 'resnet18', pretrained=True).eval()
```

第2步：定义预测函数

接下来，我们需要定义一个函数，该函数接受用户输入，在本例中是图像，并返回预测结果。预测结果应作为字典返回，其中键是类名，值是置信概率。我们将从这个文本文件中加载类名。

对于我们的预训练模型，它将如下所示：

```
import requests
from PIL import Image
from torchvision import transforms
# Download human-readable labels for ImageNet.
response = requests.get("https://git.io/JJkYN")
labels = response.text.split("\n")
def predict(inp):
 inp = transforms.ToTensor()(inp).unsqueeze(0)
 with torch.no_grad():
  prediction = torch.nn.functional.softmax(model(inp)[0], dim=0)
  confidences = {labels[i]: float(prediction[i]) for i in range(1000)}
 return confidences
```

让我们来详细分析一下。该函数接受一个参数：

`inp`：输入图像，作为PIL图像

该函数将输入图像转换为PIL图像，然后再转换为PyTorch张量。在通过模型处理张量后，它以名为confidences的字典形式返回预测结果。字典的键是类别标签，值是相应的置信概率。

在这一部分，我们定义了一个预测函数，该函数处理输入图像以返回预测概率。该函数首先将图像转换为PyTorch张量，然后将其传递通过预训练模型。在最后一步，我们使用softmax函数来计算每个类别的概率。softmax函数至关重要，因为它将模型的原始输出logits（可以是任何实数）转换为概

率，使其总和为1。这使得解释模型的输出作为每个类别的置信水平变得更加容易。

第3步：创建Gradio界面

现在我们已经设置好了预测函数，可以围绕它创建一个Gradio界面。

在这种情况下，输入组件是一个拖放图像组件。为了创建这个输入，我们使用 `Image(type="pil")`，它创建了组件并处理预处理，将其转换为PIL图像。

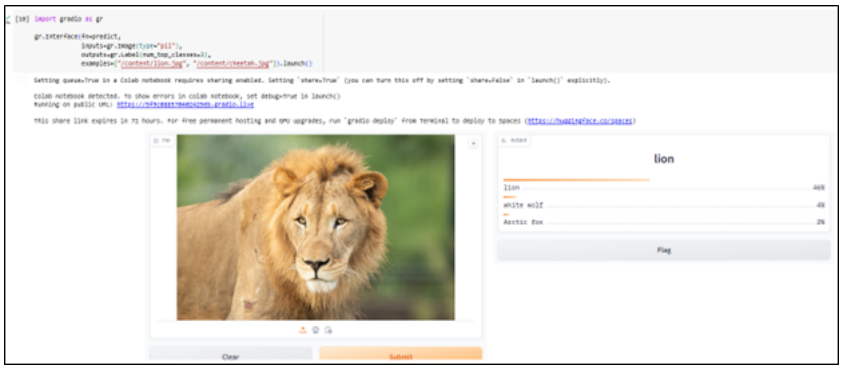
输出组件将是一个标签，显示最顶部的标签，以一种美观的形式呈现。由于我们不想显示所有1,000个类别标签，因此我们将其定制为仅显示前3个类别，构造为 `Label(num_top_classes=3)`。

最后，我们将添加一个参数，即示例参数，它允许我们用一些预定义的示例预填充我们的界面。Gradio的代码如下：

```
import gradio as gr
gr.Interface(fn=predict,
        inputs=gr.Image(type="pil"),
        outputs=gr.Label(num_top_classes=3),
        examples=["/content/lion.jpg", "/content/cheetah.jpg"]).launch()
```

提供的示例路径 `/content/lion.jpg` 和 `/content/cheetah.jpg` 是占位符。您应该将这些替换为您系统或服务器上实际保存的图像路径，以便进行测试。这确保当您或其他人使用 Gradio 界面时，示例能够正确加载，并可以用于演示您的图像分类器的功能。

这将生成以下界面，您可以在浏览器中尝试（尝试上传您自己的示例）。



完成了！这就是构建图像分类器网络演示所需的所有代码。如果您想与他人分享，请在 `launch()` 界面时尝试设置 `share=True`！

结论

Gradio 简化了构建机器学习模型交互式网页演示的过程。通过将 Gradio 与像 BLIP 这样的模型集成进行图像描述，您可以创建实用且用户友好的应用程序，利用 AI 的力量解决现实世界的问题。这个工具不仅有助于展示您模型的能力，还能收集有价值的反馈以便进一步改进。

Changelog

| 日期 | 版本 | 更改者 | 更改描述 |
|2025年4月2日|0.2|Wojciech “Victor” Fulmyk|添加了 server_name 和 port 以启动|