实战：从0搭建完整 AI 开发环境写出第一个 AI 应用

邹欣

全文大约：5600字

读完可能需要下面这首歌的时间

随着 AI 越来越深入的发展，智能革命的浪潮隐约到来，悄然的影响着软件行业。

那么，作为多年的程序员，或者准备着成为新一代程序员的读者们，该如何为智能时代做好准备，成为 AI 时代的程序员呢？

网上铺天盖地的教程，大部分都是从高数、线代、概率讲起的算法基础，或者是讨论某某网络的算法，比较模型的效果，偏研究和数据科学。

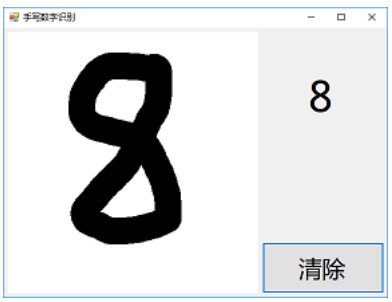
多学一些知识，当然是很好的，但毕竟需要不少的时间，在软件开发中积累的经验好像也用不上太多。

如果程序员们从自己擅长的领域出发，逐步融入 AI 热潮中，岂不是取长补短，事半功倍？从另一方面来看，以后是不是只需要开发 AI 模型就够了呢？显然不是，AI 模型会带来智能革命，但传统的软件开发仍然是基础。

就像工业革命发生后，人类仍然需要农业提供足够的粮食；信息革命发生后，也需要工业革命所带来的巨大工业生产力。因此，智能革命也会基于信息革命所构建的巨大的软件制造经验和能力之上。

本文帮你入门 AI 应用开发，写出第一个 AI 程序 -- 手写识别（见下图）。同时会穿插着讲一些必备的 AI 知识和背景，带你走入 AI 新世界的大门。

找个网络环境较好的地方，计划好空闲的时间，放松心情，一步步学习下来，保准你收获满满。



注意：安装过程中要下载的软件较多，建议在网速稳定且较快的环境下进行。整个时长取决于预先安装的情况，以及网络状况。通常一个多小时即可完成环境搭建和 AI 应用。

1. 配置 AI 开发环境

1. 安装要求

- Windows 64 位版本

强烈推荐升级到 Windows 10 的最新发行版，并安装上所有更新。Windows 7，Windows 8 也需要64位版本，推荐安装所有系统更新。

- Visual Studio

本教程将安装 Visual Studio 2017。如果有旧版的 Visual Studio 且条件允许，最好先行卸载。特别是 Visual Studio 2012，可能会出现兼容性问题。

- Python

在安装 Visual Studio 2017 时，会安装 Python 3.6 版本。如果系统已经有 Python 3.6 将不会再安装它。如果是这种情况，在本文中配置路径时，应配置成当前 Python 3.6 的路径。

- AI 框架

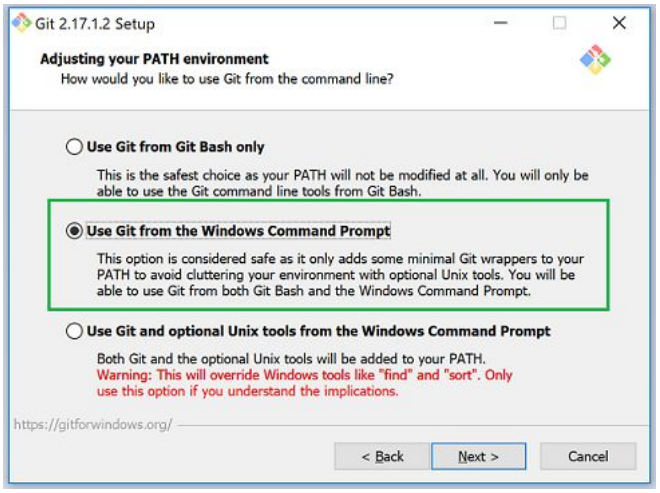
本安装过程会安装 TensorFlow，CNTK，pytorch，Keras，Caffe2，Theano，MXNe，Chainer 等流行的 AI 框架。如果你的 Python 环境已经有了一些框架并正在使用中，建议最好选择干净的 Python 环境来安装。以免产生版本冲突，或影响正在使用已安装框架的代码。

2. 检查并安装 Git

Git 是流行的源代码版本管理工具，应用非常广泛。在接下来的安装过程中，会通过它下载一些 AI 组件。

先打开命令行或终端窗口，输入 git，看是否能找到此命令。如果命令行返回了 Git 命令的用法说明，则表示已经安装了 Git，可跳过 Git 安装部分。

如果没有安装 Git，打开安装页面，自动下载 Git 安装包。下载完后，点击安装。为了方便使用，在安装向导中需选择在命令提示符中使用 Git（见下图）。其它步骤直接点击 Next。



3. 检查是否安装 NVIDIA 显卡机器学习包

在机器学习中，有的算法在并行计算下速度会得到很大的提升。而 GPU 由于要进行快速的图形处理，且这类计算可并行程度很高，所以 GPU 有很强的并行计算能力。在运行一些机器学习算法时，同等价位的 GPU 的速度会比 CPU 快上数十倍、甚至百倍。

NVIDIA 的显卡是机器学习领域中最流行的硬件之一，几乎所有框架都集成了对它的支持。安装过程也包括了 NVIDIA 显卡的配置。

如果没有 NVIDIA 的显卡，则可跳过这步，直接安装或配置 Visual Studio 2017。

确认自己的显卡是 NVIDIA 的，并支持 CUDA

CUDA（Compute Unified Device Architecture）是 NVIDIA 公司推出的通用并行计算架构，能将显卡用于解决各种复杂的计算问题。通过显卡的成百甚至上千个并行核心来加速计算。

访问 NVIDIA 的桌面或笔记本显卡列表，并找到自己的显卡型号，点开详情页面。如果在左边能看到下图的高亮部分，则表示支持 CUDA（部分高端显卡会进入购买页面，这也表示支持 CUDA），否则开始安装或配置 Visual Studio 2017。

如果不太清楚如何检查 GPU 型号或找不到自己的显卡，可先跳过下面 CUDA 与 cuDNN 的安装过程，在接下来安装AI框架时能够自动检测 GPU 是否受支持，是否安装了 CUDA，cuDNN。

如果发现显卡是支持的，但没有安装 CUDA 与 cuDNN，可以在安装结束后再进行这一步。

4. 安装 NVIDIA 机器学习相关组件

如果确认了是支持 CUDA 的 NVIDIA 显卡，根据下面的步骤来完成配置。

1. 确认安装了最新的显卡驱动

    如已安装，接下来我们安装 CUDA

2. 安装 CUDA 9.0

为了减少 TensorFlow 等机器学习框架的兼容性问题，这里选择了安装兼容性最佳的 CUDA 9.0。

首先，进入 CUDA 9.0 下载页面。

然后，选择对应的操作系统、CPU 架构、操作系统版本来确认安装包。安装包本身也比较大，如果网速稳定，可以选择网络（network）版本来按需安装。如果网速不稳定，或者要在多台机器上安装，可选择本地安装（local）版本一次性完成下载。

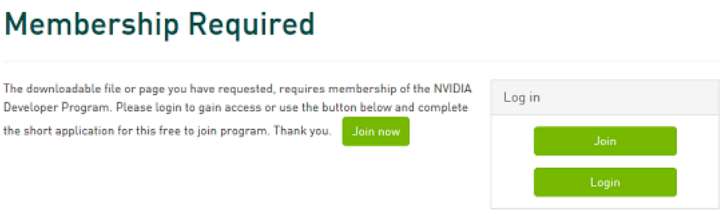
选择好后，直接点击打开或另存到本地再运行。

注意：如果安装过程中出现了以下错误，可能是由于显卡较新，而 CUDA 9.0 中不包含对应的驱动。这时候可以重新开始安装过程，选择**自定义安装**，并取消勾选显卡驱动（Driver Components）再试一次。

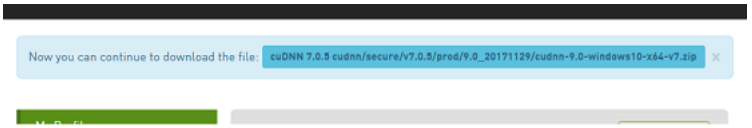
3.配置 cuDNN

cuDNN（CUDA Deep Neural Network）是 NVIDIA 基于 CUDA 的专为深度神经网络优化的计算库。cuDNN 的下载安装过程比较复杂，需要在 NVIDIA 网站进行注册后才可下载。

首先进入 cuDNN 下载页面，点击展开 Download cuDNN v7.0.5 (Dec 5, 2017), for CUDA 9.0。然后根据对应的操作系统选择对应的版本，随后会进入下面的登录界面。



注册或者登录后，就能看到下载链接，或者重新点击下载页面再来一遍。



解压开会有一个 dll 扩展名的动态链接库文件，将其拷贝到 CUDA 安装目录的 bin 目录中即可。CUDA 的默认安装路径路径如下：

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\bin
```

安装或配置 Visual Studio 2017

注意：如果有 Visual Studio 的其它版本，尽可能卸载后，再安装 Visual Studio 2017。特别是 Visual Studio 2012 或者更低版本，否则在使用时有可能发生版本冲突相关的问题。

如果已经安装了 Visual Studio 2017，只需要从开始菜单中搜索并打开 Visual Studio Installer。选择更多中的修改来确认是否安装了 Python开发 与 .NET 桌面开发这两个工作负载。如果没有，则选上并安装。

1.进入下载页面，指向下载 Visual Studio，并点击 Community 2017。

Visual Studio Community 版是完全免费的，包含有 Visual Studio 的大部分基础功能，也能全面的支持 AI 应用开发。只需要用微软账户登录后，就可以一直使用。

2.运行安装程序后，会打开工作负载的选择界面。至少要选择 Python 开发与 .NET 桌面开发。

Visual Studio 将多种场景下的开发都按照工作负载的方式进行了组合，只需要勾选上，就能顺畅的进行对应场景下的开发了。安装完成后，可以通过开始菜单中的 Visual Studio Installer 再次打开这个界面，安装其它工作负载。

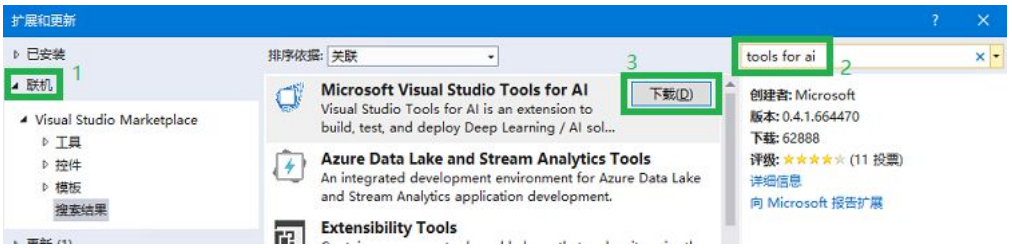
随后点击右下方的安装按钮，即可开始安装。根据选择的内容多少以及网络情况，安装时长也会很不一样。此处需耐心等待。

安装 Visual Studio Tools for AI

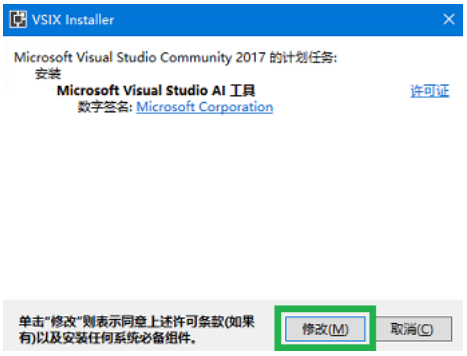
Visual Studio Tools for AI 是 AI 集成开发环境中较核心的部分，包含了训练任务管理、模型推理等功能。

1.启动 Visual Studio，在菜单栏中选择工具 -> 扩展和更新。

2. 点击**联机**，并在右上方输入 `tools for ai` 后回车。如果第一次没有搜索出来，可再增加一个空格，并再次回车搜索。搜索结果出来后，点击 `Microsoft Visual Studio Tools for AI` 的下载按钮。



3. 下载完成后，关闭所有 Visual Studio 窗口，来触发插件的安装过程。关闭 Visual Studio 后，稍等数秒钟，即会出现安装界面，点击**修改**。等待整个安装过程结束，然后点击**关闭**。



4. 再次打开 Visual Studio，在菜单栏会看到 `AI 工具`，表示安装完成。

安装 AI 框架

在机器学习中，特别是深度学习中，经常会因为各种原因，需要在不同的 AI 框架之间切换。而不同的框架可能还依赖于不同的底层库版本。因此，搭建一个顺手的 AI 开发环境，颇会花些时间和精力。

本文通过微软 AI 示例库中的一键安装脚本，简化了 AI 框架的安装过程。

一次性就能装上 TensorFlow，CNTK，pytorch，Keras，Caffe2，Theano，MXNet，Chainer 等流行框架，以及 jupyter，matplotlib，pandas，scipy，onnx，tf2onnx，scikit - learn，xgboost，libsvm 等流行工具。这个 1000 多行的安装脚本会节省大量的安装配置时间，也让新手不再被卡在安装配置环境这一步上。

下载微软 AI 样例库

打开命令提示符或终端窗口，选择并进入某个用来存放代码的目录，如 `%USERPROFILE%` 。运行下列命令，来下载微软 AI 示例库，完成后进入该目录。

注意：请确保整个路径中没有中文。不少机器学习框架对多语言支持并不好，为了防止因此出现的问题，保证整个路径没有中文等扩展字符集。

注意：推荐用管理员权限打开命令提示符窗口。以免 Python 安装目录需要管理员权限才能写入文件。步骤：打开**开始**菜单 -> 用键盘输入 `cmd` -> 右击出现的**命令提示符** -> 选择**以管理员身份运行**-> 通过 `cd` 命令选择一个合适的目录，如 `cd /d %USERPROFILE%` 。

```
git clone https://github.com/Microsoft/samples-for-ai.git
cd samples-for-ai
```

运行安装脚本

1.命令提示符下可能没有配置 Python 的路径。运行 python --version 检查一下。

```
C:\>python --version
'python' 不是内部或外部命令，也不是可运行的程序
或批处理文件。
```

如出现上图的内容，表示系统路径中没有 Python ，则根据系统中 Visual Studio 的安装路径，用下列命令来设置 Python 路径。

注意：如果修改过 Visual Studio 的默认安装路径，则此命令也需要做相应的修改。

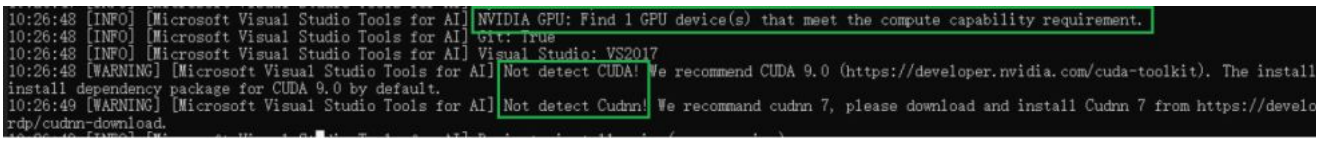
```
set PATH="C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64";%PATH%
```

2.执行 install.py

```
python installer/install.py
```

3.检查并配置 CUDA、cuDNN

如果前面的步骤中，因为不确定显卡是否支持 CUDA 等原因，而跳过了 CUDA 安装部分。这里可以看一下输出。如果在输出的前面几行里发现了以下字样，则表示支持 CUDA，且没有正确安装好 CUDA 或（和）cuDNN。等待安装脚本运行完后，参考前文完成 CUDA 和 cuDNN 的安装。



安装结束后，如果有任何红色的错误文字，请参考常见问答：运行 install.py 时出现红色错误文字时，该如何处理？。

命令行窗口完成后回车一下，但不要关闭，接着开始训练第一个模型。

训练第一个模型

下载的 samples-for-ai 中包含了大量的机器学习训练和应用的示例。使用 TensorFlow 的 MNIST 示例来测试一下环境安装是否成功，也为下一步准备好 AI 模型。

```
python examples/tensorflow/mnist/mnist.py
```

运行过程中会打印出一些 error 字样。别担心，这都是 AI 训练过程中正常的错误率信息。随着训练过程的进行，模型在训练时的错误率会逐步下降，这表示 AI 模型推理预测出的结果越来越准确了。

```
Step 8000 (epoch 9.31), 18.2 ms
Minibatch loss: 1.646, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Step 8100 (epoch 9.43), 16.8 ms
Minibatch loss: 1.630, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Step 8200 (epoch 9.54), 15.2 ms
Minibatch loss: 1.624, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.8%
Step 8300 (epoch 9.66), 18.2 ms
Minibatch loss: 1.622, learning rate: 0.006302
Minibatch error: 1.6%
Validation error: 0.8%
Step 8400 (epoch 9.77), 18.5 ms
Minibatch loss: 1.596, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.8%
Step 8500 (epoch 9.89), 15.8 ms
Minibatch loss: 1.609, learning rate: 0.006302
Minibatch error: 1.6%
Validation error: 0.9%
Test error: 0.8%

C:\Users\squir\source\repos\samples-for-ai>
```

2. 创建第一个 AI 应用

克隆代码，并导入训练好的模型，就可以试试自己的第一个 AI 应用了！当然，不仅要知其然，也知其所以然，接下来还会分析一下核心的代码。

把程序跑通

克隆代码

使用下面的命令来克隆 AI 应用的代码。代码里有一个应用的窗体项目，预先写好了所有的代码。同上，要选择好放置代码的路径，如：cd /d %USERPROFILE%

注意：放置克隆代码的路径中不能含有中文。

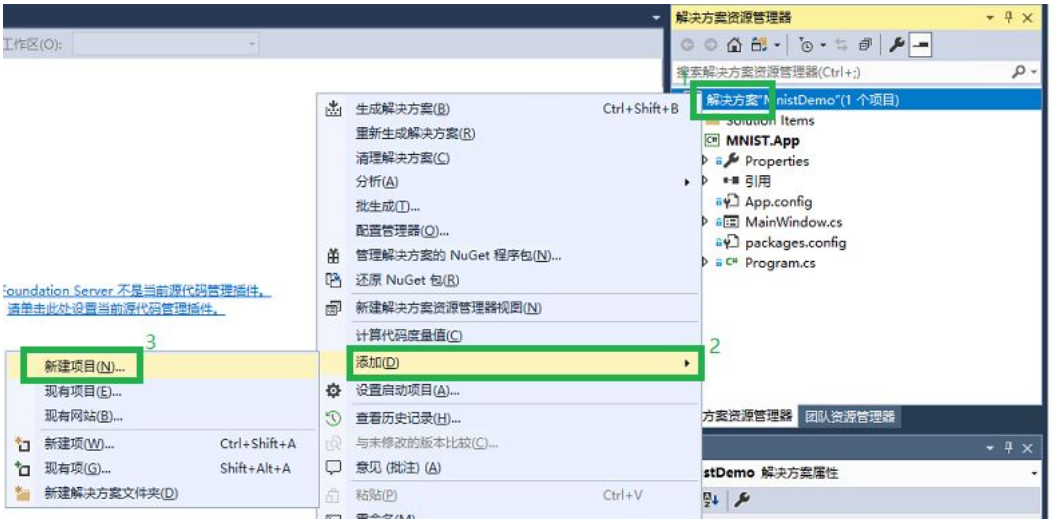
```
git clone https://github.com/squirrelsc/sample-mnist.git
start sample-mnist\MnistDemo.sln
```

运行完上面的脚本后，就会在 Visual Studio 2017 中打开这个解决方案。

引用模型

1. 首先创建模型项目。在**解决方案资源管理器**中，右击**解决方案**，指向**添加**，再点击**新建项目**。

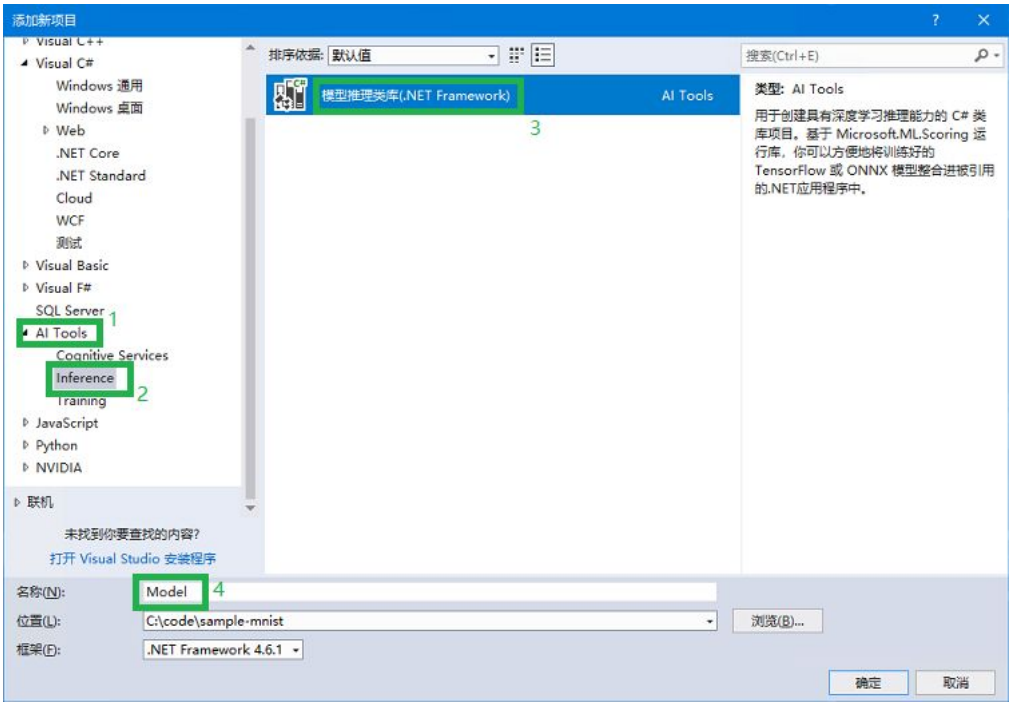
注意，一定要在解决方案上右击，否则不会出现**新建项目**的菜单。



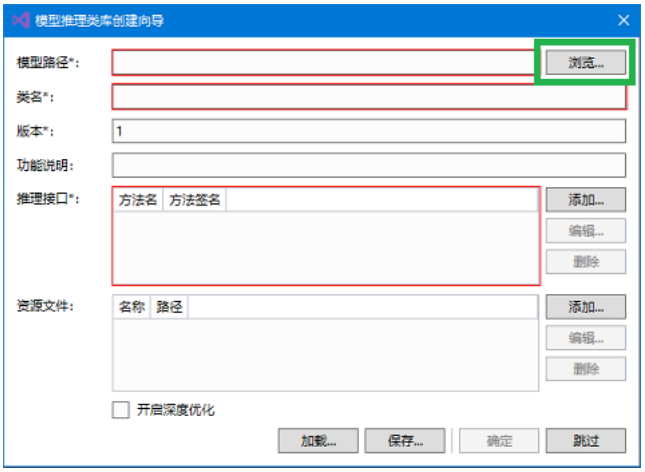
2. 在弹出的添加新项目的对话框里，选择 AI Tools 下的 Inference 后，在右边选择**模型推理类库（.NET Framework）**。

然后在下面的**名称**处改为 Model，并点击确定。

注意：名称一定要保持一致（包括大小写），这是生成代码的命名空间。

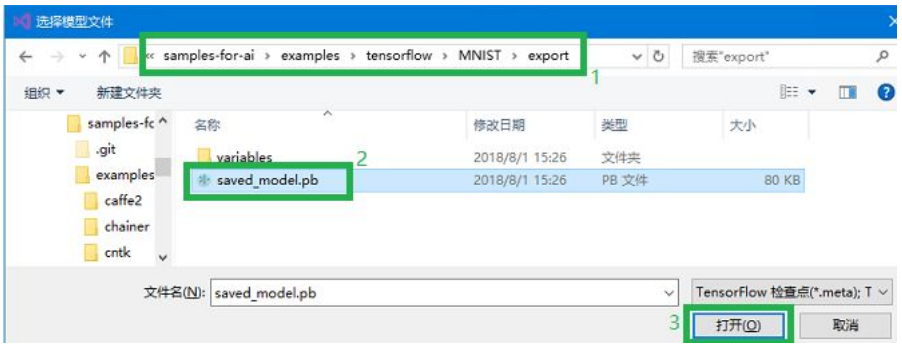


3.点击确定后，Visual Studio 会提示在检查环境，完成后会显示下图。红色的框先不用担心，点击浏览。



4.浏览到示例代码的下载路径，并继续选择到如下路径中的 saved_model.pb 文件，并点击打开。

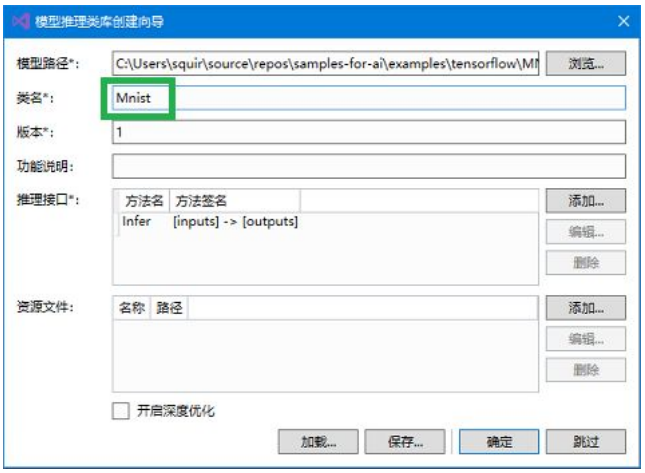
注意：参考前文训练模型部分的代码来找到示例的路径。



5.点击打开后，会出现分析模型的过程，完成后，就会如下图。这时候，再在类名中输入 Mnist，最后点击确定。

第一次导入模型会下载所依赖的库，因此，正在创建项目 “ Model ” ... 的对话框会显示较长时间。

注意：Mnist 会是生成代码的类名，所以也要保持一致（包括大小写）。另注意单词中 n 和 i 的位置。



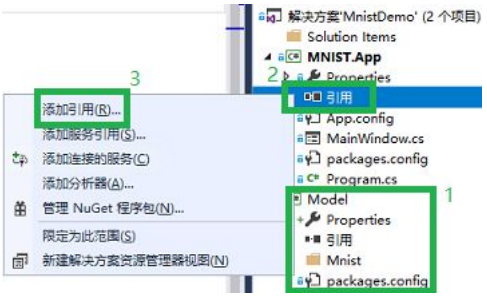
分析模型会调用 TensorFlow 来分析模型文件的输入输出等信息，以便生成相应的代码，有时会花一两分钟。

6.一旦创建项目完成后，先检查一下是不是如下图产生了 Model 项目，并且里面有 packages.config 文件。如果没有看到此文件，通常是由于网速过慢，下载超时造成的。请参考**常见问题：创建模型项目时出现错误，该如何处理？**

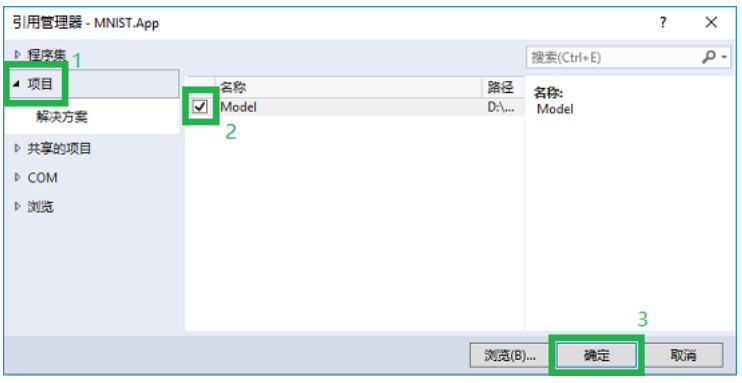
然后，右击 MNIST.App 项目的**引用**，并点击**添加引用**。

如果有兴趣的话，可以点开 Mnist 目录看看生成的代码。这里面还包含了优化后的模型数据文件。

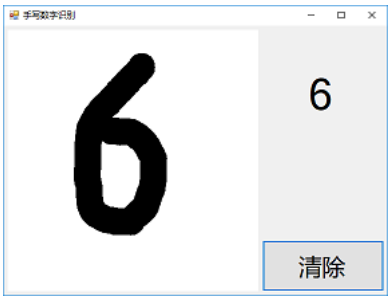
注意：要在**引用**上右击。



7.在弹出的对话框中，点击**项目**，并在右侧将 Model 勾上。这样就能在窗体项目中引用 Model 项目了。



8.现在按下 CTRL + F5，或者点击工具栏上的**启动**按钮。你的第一个 AI 应用就运行起来了！666。



理解代码

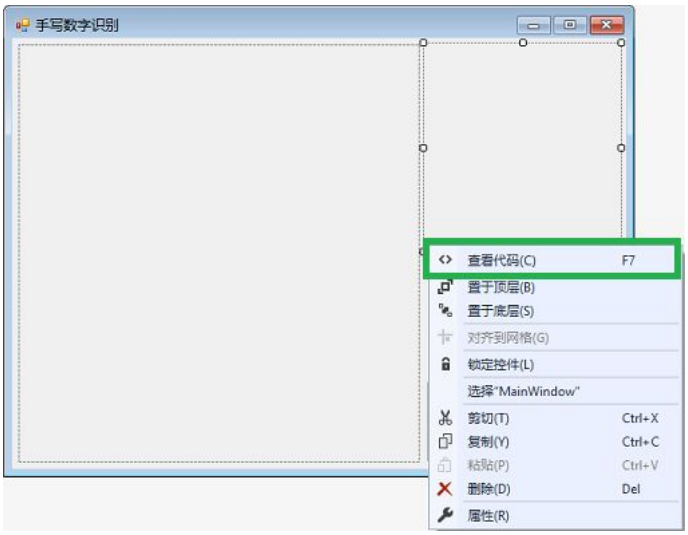
知其然，还要知其所以然，所以要简单介绍一下代码以及必须的知识。代码的逻辑都在 MNIST.App 项目的 MainWindow.cs 文件中，剩下的都是自动生成的代码。该文件包括了界面联动、数据预处理两部分的代码，以及一行推理预测的代码。界面联动是为了实现手写输入时的良好体验，而数据预处理部分是在推理前，将用户输入的笔迹变为模型所需要输入的浮点数组。

注意：代码中包含了非常详尽的注释。建议在读完本章节后，再通读代码中的注释，以便更深入的理解整个代码逻辑。

界面联动

1.展开 MNIST.App 项目，找到 MainWindow.cs 文件。所有的界面设计和代码都在这里。先双击它，打开设计界面。可以看到，设计界面和程序实际运行起来的效果非常像。窗体上包含了三个控件：PictureBox，Label，以及Button。稍后会介绍控件对应的变量名称，以及添加的事件、用途等。

2.在窗体的设计界面上右击，并选择**查看代码**，可以看到核心的代码。



3.打开代码后，可以看到，几乎每一行代码都有对应的注释。除了类里的 ImageSize 等几个变量外，剩下的逻辑几乎都在事件响应函数中。这些响应函数是在控件属性的事件面板中添加的。每个控件都在代码中可以通过变量名称来使用。

具体信息如下：

变量名称	控件类型	添加的事件	用途
Form1	Form	Form1_Load	主窗体。 加载时会一次性初始化部分变量。
writeArea	PictureBox	`writeArea_MouseDown`, `writeArea_MouseMove`, `writeArea_MouseUp`	手写区域。在鼠标操作时，响应鼠标的按下（MouseDown)、移动（MouseMove）、释放（MouseUp）事件。 在触摸屏操作时， 与鼠标操作类似， 会响应手指的接触屏幕、在屏幕上移动、 离开屏幕的事件。
outputText	Label		文本标签。 显示推理结果的数字。
clean	Button	clean_click	清除按钮。 在每次推理前清除手写区内容， 及文本标签显示的数字。

数据预处理

数据预处理是 AI 应用的重要一环。

在大部分 AI 应用中，特别是本文的图片分类应用中，通过监督学习来训练模型。即先提供一些标记过分类的图片来训练出模型，然后输入未知的图片，推理预测出此图片的类别。因此，在训练和推理过程中，每次输入模型的数据格式必须完全一致，这样才能保证预测推理的效果。

本示例已知了训练数据的输入形式，所以按图索骥就能写出代码。在 AI 的实际应用中，一定要了解模型输入数据的格式细节，严格的实现它。如果数据格式细节不一致，通常会降低推理结果的正确率。而这类问题几乎不会产生编译或运行错误，而且数据是不易直观理解的浮点数组，所以对此类问题的诊断和修正较困难。

下面会介绍一下本例中的数据预处理过程，从而体会一下数据预处理中的细节问题。

1. 数据预处理的第一步，在窗体设计时，手写区域调整为了正方形，和训练数据的形状保持一致。

2. 定义了类变量 ImageSize 常量等于 28。这是训练数据的实际图片尺寸。

界面上的正方形最终会缩小为 28 x 28。以下均为C#代码

```
1 private const int ImageSize = 28;
```

3.clear 函数中设置了手写区域的背景为白色。训练数据是黑白的，需要将前景、背景颜色同样设置，从而与训练数据一样，达到最大的对比度。

```
graphics.Clear(Color.White);
```

4. writeArea_MouseMove 事件中设置了手写笔风格。手写笔迹宽度是 40，颜色为黑色，开始、结束位置设成圆头。

笔迹宽度与图片尺寸的比例基本匹配了训练数据的比例。但无法控制用户输入的文字大小，还会有一些误差。好在深度学习的模型适应性较强，对识别准确率的影响不太大。

黑色笔迹配合了白色背景，形成最大的对比度。

笔迹的开始、结束位置为圆头。在书写过程中会多次调用到鼠标移动事件中，每次根据上一次的结束位置到当前位置画了一条直线。如果不将笔头设置为圆头，就会像下图一样，这些直线会形成矩形拼接起来，形成很多不连续的位置。既影响识别，也不美观。

```
graphics.Clear(Color.Pen penStyle = new Pen(Color.Black,
40) { StartCap = LineCap.Round, EndCap = LineCap.Round
};White);
```

5. writeArea_MouseUp 事件中包含了其它的数据处理逻辑。首先构造了 28 x 28 的图片，将手写的图片缩放到了新的图片对象中。

```
Bitmap clonedBmp = new Bitmap(writeArea.Image, ImageSize,
ImageSize);
```

6. 按行、列遍历了 28 x 28 位图中的所有节点，并取出了每个像素，处理后存入数组中。

```
for (int y = 0; y &lt; ImageSize; y++)
  {
      for (int x = 0; x &lt; ImageSize; x++)
      {
          Color color = clonedBmp.GetPixel(x, y);
          // 此处略过处理过程，见下文...
          image.Add((float)reversed);
      }
  }
```

7. 将红绿蓝通道加和并平均，完成了像素灰度化。手写识别模型的输入数据是黑白图。

手写区是白底黑字，每个颜色通道的值其实是一样的。为了逻辑上的严密和便于理解，所以对值取了均值，进行灰度化。

```
double average = (color.R + color.G + color.B) / 3.0;
```

8. 将取值范围变换到了 0 ~ 1。机器学习中取值范围变化很大，因此绝大部分机器学习模型都用浮点数进行计算。

```
double oneValue = average / 255;
```

9. 将数值翻转，并做了 0.5 的位移。这一步减少输入数据中零值的数量。过多的零，会让中间结果也出现更多的零，在神经网络中容易丢失信息。

白色背景通过灰度化之后是 255，变换后为 - 0.5；黑色笔迹灰度化之后是 0，变换后为 0.5。这样处理后，大部分值都成为了非零值。

对于数据的取反，是经验的做法。通过试验，在很多情况下取反后的训练效果会更好。MNIST 数据集的数据也是取反保存的。

```
double reversed = 0.5 - oneValue;
```

推理

推理即输入数据并获取模型的预测结果。这一步非常简单，输入变换后的一维数组，输出预测结果即可。推理函数每次调用可以输入多张图片，进行批量预测。

```
long inferResult = model.Infer(new List<IEnumerable<float>>
{ image }).First().First();
```

实际场景

本示例非常简单，还不足以成为实际的手写识别产品。但迈出这小小的一步，就进了 AI 应用开发的门槛。这里先抛砖引玉，再提出一些实际中会遇到的问题，并给出部分解决方案供参考。

在真正应用时还会发现更多的问题，如连笔文字，歪斜矫正，梯形校正等等。因此，加入 AI 模型后，能将不可能变为可能，但所需要的软件开发工程量仍然很大，传统软件开发中的挑战一样也不会少。

在思考实际场景的同时，也会逐渐培养出 AI 的应用意识。做产品时，如果遇到了有很多数据，但很难找到规律的情况。不妨思考一下，能否用 AI 模型解决这个问题？有了这样的 AI 意识，就能更好的将 AI 和传统软件相结合，创造出让人眼前一亮、用户体验自然的产品。

在大部分 AI 应用中，数据非常重要。经常可以看到网上的讨论认为：数据是 AI 应用的基础。在下面的解决方案中也能看出，很多方案依赖于大量的、有代表性的数据。

传统方案很易于理解和实现，但需要巧妙的算法，处理噪音等非常规情况也比较棘手。AI 模型的质量则取决于数据、特征工程及算法，不需要钻研算法。

究竟哪个更好，要在实际场景中验证后，才能有结论。也可以同时使用多种方案，再通过一个 AI 模型来决定使用哪个方法的结果。

大小不一或没有居中

当字写得小一点，或者写偏一点，识别错误的情况就会变多。这是因为训练数据为了提高模型的识别率，对数据都进行了居中、缩放的处理。可能的解决方案如：

1. 采用一些图像算法，如：找到笔迹像素的分布中心，将其居中，并根据外围的笔迹像素来进行缩放，适配到识别区中。此方法逻辑清晰，不需要数据进行训练。缺点是如果手写数据来源于摄像头，会有很多噪点，会造成误判。

2. 训练一个目标检测（Object Detect）的 AI 模型来自动框出合适的大小。这种方法会将需要的目标（即字符）用矩形框出来、并基本保证其居中。将矩形做适当的缩放后，即可作为输入。如果数据量足够且有代表性，这种方法的效果会不错。但目标检测需要的数据标记工作却很繁琐，需要对每个训练数据中的字符画框标记，标记的质量也直接影响到了识别的质量。

多字符识别

在书写时，通常是连续的一段话或者一个等式，很少只输入一个字符。需要将其分割为单字符，再逐个识别。有两个问题需要解决：一是将图片分割出单个字符，二是将字符进行排序。

- 分割字符
    - 传统的解决方案较多，如：

- 找到相邻的点进行扩展，找到每个连通区域，然后将重合的矩形合并起来，作为一个字。这种方法的缺点是如果在识别中文等情况下，遇到左右或者上下结构，可能会被识别成两个字符。
- 检查每一列像素，如果没有任何笔迹像素，则将左右两侧切割开来。但这样在左右结构时仍然会遇到问题，也不支持纵向书写。
- 用 AI 模型，可以使用上文提到的目标检测（Object Detect）方法，框出每个字符。

- 字符排序。

单个字符本身的意义还不够，语言文字中需要将字符组成词语和句子，才有更丰富的含义。在上一步中，分割出了单个的字符，完成识别后，要将它们组合成词语和句子。

除了常见的横向书写外，还有纵向书写。如果纸上没有画线，还可能越写越歪。这些问题都给字符排序带来了麻烦。这一步可以用传统的方法根据各个方向的距离等因素综合判断是否是连续的字符串。

个人风格迥异

解决此问题，主要依靠搜集不同的写作风格来解决。如果有了有代表性的数据集，这个问题就迎刃而解了。

识别字母、符号、中文

同上，数据是最重要的。另外，随着问题规模的变大，使用的 AI 模型有可能也要修改，要有足够的复杂度来容纳更多的信息。

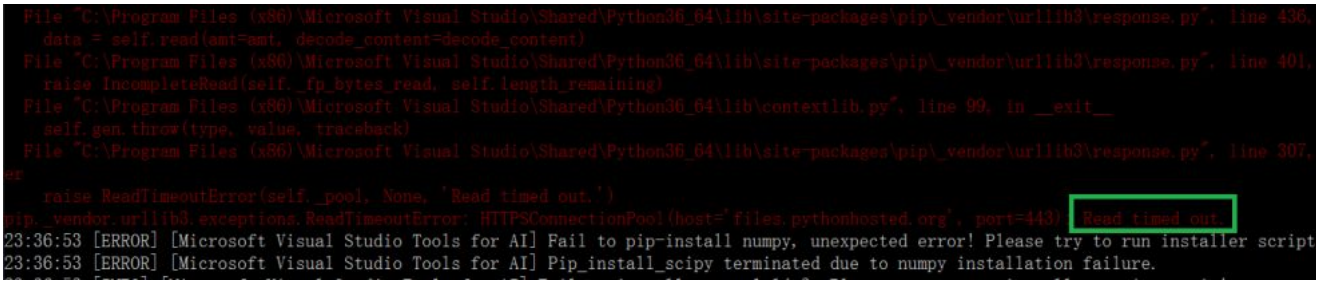
如：中文字符比较复杂，可能 28x28 的大小不够表达足够多的细节，需要原始输入的尺寸更大。

3. 常见问答

运行 install.py 时出现红色错误文字时，该如何处理？

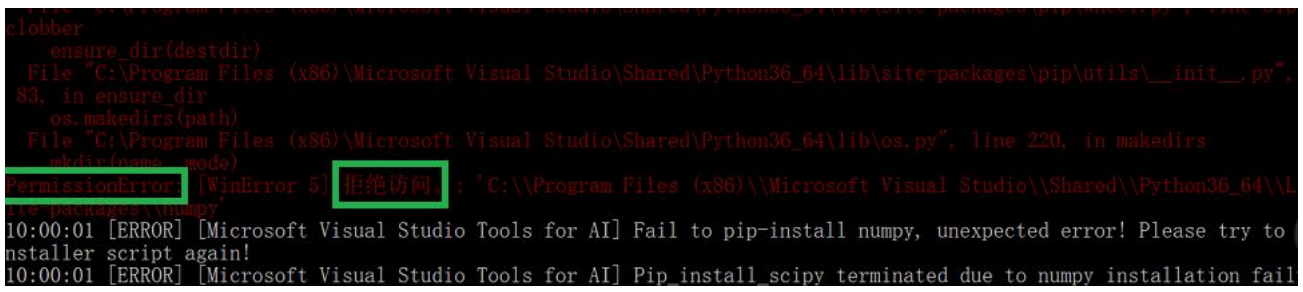
因为 Python 语言的异常信息通常是在最下面的一行，所以需要认真阅读红色文字的最后一行，寻找关键字。根据不同的问题进行处理：

- time out



一般是因为网速不够或网络不稳定造成下载文件超时。可以等整个脚本完成后再重试。或者是换个网络环境再试试。重试时，已经装好的包不会再次安装，所以会更快一些。

- access denied 或“拒绝访问”



这是因为 Python 安装在了系统目录中，需要管理员权限才能有写入权限。用管理员权限打开命令行窗口才能安装。步骤：打开开始菜单 -> 用键盘输入 cmd -> 右击出现的命令提示符-> 选择以管理员身份运行。然后重新进行命令行相关的安装工作。

- 其它错误 大部分其它错误都可能是网络问题，可以参考 time out 部分的处理方法。 另外还有可能有版本兼容性问题。如果有多个 Python 环境，推荐选择干净的 Python 环境进行安装。

系统路径下已有 Python 时，该如何处理？

Python 是 AI 训练中使用得最多的语言，绝大部分 AI 框架和库都基于 Python。

一个操作系统中可以安装多个 Python 实例，但并不是每个 Python 环境都能用来做 AI 开发。在本安装过程中使用了 64 位的 Python 3.6。原因如下：

- Python 2.x 下受支持的 AI 框架逐渐变少，但 3.x 下几乎全部支持。
- Python 32 位下的 AI 框架也很少。
- Python 3.6 是相对较新，且目前兼容性较好的版本。
- pip 命令只会将 Python 的包安装在一个 Python 实例中，只有在当前实例中安装了某个包，才能使用它。

如果以前安装过机器学习的软件包，推荐安装全新的 Python，从而获得最好的兼容性。如果没安装过，则参考下面的步骤，检查当前环境是否能直接使用。

- 参考前文 python --version 命令的结果，如果输出结果不是 Python 3.6.x 或 Python 3.5.x，则必须安装新的 Python 环境。下图为 3.6.5，是正确的 Python 版本。

```
C:\>python --version
Python 3.6.5
```

- 运行 python -c "import platform;print(platform.architecture())"

来检查是否是 64 位 Python。下图中包含了 “ 64bit ” 字样，Python 的平台满足需求。

```
C:\>python -c "import platform;print(platform.architecture())"
('64bit', 'WindowsPE')
```

如果上述任何条件不满足，或安装过其它机器学习组件，推荐安装新的 Python。

克隆代码时经常出现错误，该如何处理？

在教育网等网络环境里，访问 GitHub 较不稳定。如果不能找到其它网络环境，只能在网络的非高峰使用期尝试。

创建模型项目时出现错误，该如何处理？

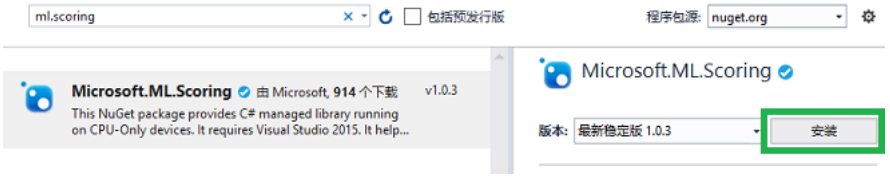
如果在添加模型项目完成后，没有在目录下看到 packages.config 文件。通常是因为网络不稳定，下载超时造成的。参考下面的过程手工下载依赖的库。

- 右击 **Model** 项目，选择 **管理 NuGet 程序包...**。

- 在弹出的 NuGet 页面里，点击**浏览**，然后输入 **ml.scoring**，来手工搜索需要的 NuGet 包。

- 点选搜索出的 **Microsoft.ML.Scoring**，点击右边的**安装**，然后点击**确定。**

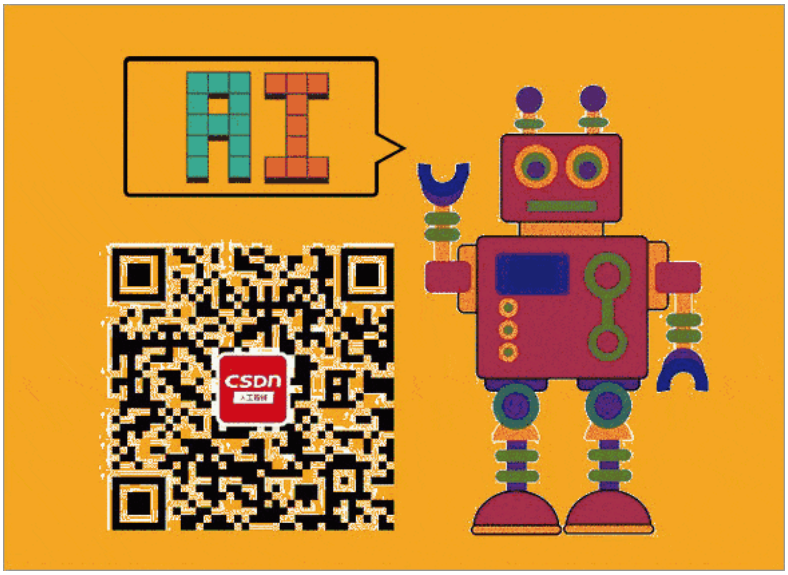
会在输出界面看到开始添加 Microsoft.ML.Scoring 的信息。如果网速较慢，这一步会花费一些时间。完成后，即可根据本文内容进行下一步操作。



运行时提示 “ 系统找不到指定的路径 ”，该如何处理？

一般是由于路径中有中文字符造成的，需要将整个解决方案移动到不包含中文等扩展字符集的路径中。

本文为特约作者邹欣撰写


AI
CSDN