

The background features abstract, overlapping green geometric shapes in various shades, creating a modern and dynamic visual effect.

第二十讲

Node-RED (IV)

Lecture 20

Node-RED (IV)

明玉瑞 Yurui Ming
yrming@gmail.com

声明

Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识产权，所引材料之知识产权均归原著者所有；若原著者介意之，请联系作者更正及删除。

The time limit for the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for remedy including deletion have you had any concern.

TensorFlow 节点

TensorFlow Nodes

- ▶ 当下人工智能技术如火如荼地开展，因此，将人工智能与物联网结合起来是一个很直观的想法。作为成熟的人工智能框架，TensorFlow提供了从研究到工程、从训练到部署、从服务器到嵌入式的成熟的方案。因此，我们可以借用TensorFlow.js，在Node-RED中加入对人工智能应用的支持。

The research and applications of artificial intelligence (AI) is riding an unprecedented tidal wave currently, therefore, it is worthy of combining AI and Node-RED together. As a mature AI framework, TensorFlow almost provides an ubiquitous solution covering from research to engineering, from training to deployment, from server to embedded systems. Node-RED can be accordingly enhanced by virtue of TensorFlow.js to support a variety of AI applications.

- ▶ 虽然基于浏览器的TensorFlow.js与基于Node.js的TensorFlow.js在语法与应用上是一致的，但由于Node.js运行在服务端，其实际上是借助TensorFlow C++适配程序直接运行的，因此性能要比基于浏览器的TensorFlow.js性能要好。

Although it is not necessary for users to distinguish the browser based TensorFlow.js from Node.js based TensorFlow.js, especially from syntax and programming perspective, however, Node.js based TensorFlow.js is actually built upon the TensorFlow C++ adapter and run on local hardware directly, which boost the performance highly surpassing the browser based version.

TensorFlow 节点

TensorFlow Nodes

- ▶ 在我们深入讨论如何将Node.js和TensorFlow结合之前，让我们回顾一下Node.js的起源。事实上，JavaScript的核心特性使该语言具有与其他编程语言相比的高并行性等优势，因此人们想知道是否可能将其用于服务器端程序，这促使了Node.js的创建。然而，在服务器端运行意味着它需要直接与系统进行资源相关操作。一个相当直观的解决方案是绑定到用C或C++编写的模块以提高性能。由于JavaScript也在不断发展，因此绑定需要符合一些标准，即所谓的NAPI版本。

Before we dive into how we can combine Node.js and TensorFlow together, let's recap the origination of Node.js. Actually, the core features of JavaScript provide the language some advantages such as high parallelism compared with other programming languages, so, people wonder whether it's possible to adopt it for server-side programs and this motivated the creation of Node.js. However, running on the server side means it needs to directly interact with the system for resource-related operations. A quite intuitive solution is to bind to modules/addons written in C or C++ to boost the performance. Due to the fact JavaScript also evolves, so the binding needs to conform to some standard, the so-called NAPI version.

TensorFlow 节点

TensorFlow Nodes

- ▶ 右图展示了与Node.js版本对应的NAPI版本号。注意，当在新的Node.js环境中用某些插件时，可能由于NAPI的不一致，要对之前的扩展模块重新编译。

The right figure shows the correspondence between the Node.js version and NAPI numbers. Notably, when extending the functionalities of new Node.js via might be legacy addons, you might need to compile them for version mismatch in order to solve the compatibility issues.

Node-API version	Supported In
9	v18.17.0+, 20.3.0+, 21.0.0 and all later versions
8	v12.22.0+, v14.17.0+, v15.12.0+, 16.0.0 and all later versions
7	v10.23.0+, v12.19.0+, v14.12.0+, 15.0.0 and all later versions
6	v10.20.0+, v12.17.0+, 14.0.0 and all later versions
5	v10.17.0+, v12.11.0+, 13.0.0 and all later versions
4	v10.16.0+, v11.8.0+, 12.0.0 and all later versions
3	v6.14.2*, 8.11.2+, v9.11.0+*, 10.0.0 and all later versions
2	v8.10.0+*, v9.3.0+*, 10.0.0 and all later versions
1	v8.6.0+**, v9.0.0+*, 10.0.0 and all later versions

TensorFlow 节点

TensorFlow Nodes

- ▶ 由于Node.js的功能扩展主要通过插件来完成，因此，需要一种机制来完成为数众多的插件管理。实际上，我们前面讲了Node.js的包管理器npm，这里我们再强调一下npm的特点以及其与Python包管理器pip的区别。

Due to the fact that extending the functionalities of the program is mainly via addons, so it is quite nature to provide some mechanisms to maintain this bunch of packages. Actually, we introduced the NPM package manager previously, however, here we want to readdress some of its features and its distinction from pip, the package manager of Python.

- ▶ 通常，利用pip包管理器安装Python包时，不管pip被执行时的当前工作路径，这些包总会被安装在默认的标准位置。而使用npm安装Node.js包时，默认包是会被安装在当前工作目录下的。

Usually, when installing Python packages using the pip package manager, regardless of the current working directory in which pip is executed, these packages will always be installed in the default standard location. However, when installing Node.js packages using npm, by default, the packages are installed in the current working directory.

TensorFlow 节点

TensorFlow Nodes

- 通常，我们不会从头开始做一个项目。虽然对于具有依赖关系的软件包，软件包管理器可以自动解析依赖关系，但新项目需要用户显式安装相关的软件包。在这种情况下，用户可以评估要安装的软件包的通用性，例如它被使用的可能性有多大。如果必要的话，用户可以通过添加-g标志在全局范围内安装它，就像我们之前在安装Node-RED时所做的那样。

Usually, we don't do a project from scratch. Although for package with dependencies, the package manager can automatically resolve the dependencies, new project needs users explicitly install the dependent packages. In this circumstance, users can assess like the generality the package to be installed, like how much the chance it get used. If necessary, users can install it in the global scope by adding the flag -g, as we did previously when installing node-red.

- 我们可以通过npm ls来查看当前工作路径下或全局（此时加-g）安装的包。

We can check the installed package by the command npm ls in the current working directory or globally (adding -g).

TensorFlow 节点

TensorFlow Nodes

- ▶ 右图展示了当前路径下与全局安装的包：

The right figure shows the installed package under the current working directory and globally:

```
C:\Users\ming\.node-red>npm ls
node-red-project@0.0.1 C:\Users\ming\.node-red
+-- @tensorflow/tfjs-node@4.17.0
+-- acorn-walk@8.3.2
+-- acorn@8.11.3
+-- node-red-contrib-open@1.0.0
+-- node-red-contrib-tf-function@0.1.1
+-- node-red-contrib-tf-model@0.1.12
+-- node-red-contrib-tfjs-coco-ssd@ extraneous
`-- node-red-dashboard@3.6.2
```

```
C:\Users\ming\.node-red>npm ls -g
C:\Users\ming\AppData\Roaming\npm
+-- node-gyp@10.0.1
+-- node-red@3.1.3
+-- npm@10.5.0
`-- pm2@5.3.1
```


TensorFlow 节点

TensorFlow Nodes

- ▶ 对于Windows系统，Node-RED的默认工作目录是位于用户主目录下的.node-red文件夹。因此，通过“管理调色板”菜单安装的每个软件包都会安装在.node-red文件夹中。因此，有两种方法可以为Node-RED安装额外的节点，一种是通过“管理调色板”，另一种是通过将当前工作目录设置为.node-red文件夹直接在终端中安装它。

For the Windows system, the default working folder for Node-RED is .node-red folder under user's home directory. So each package installed via the Manage palette menu is got installed in the .node-red folder. So there are two ways to install extra node for Node-RED, one is via Manage palette, the other is to directly install it in the terminal by setting current working directory to .node-red folder.

- ▶ 实际上，在终端上安装包的一个好处是，如果安装过程中有任何问题，可以方便地查看到，并且当通过管理调色板安装失败时，只能通过命令行的方式进行逐步解决。

In fact, one benefit of installing packages via the terminal is that if there are any issues during the installation process, they can be easily viewed, and when installation fails through the Manage palette, troubleshooting can only be done step by step via the command line.

TensorFlow 节点

TensorFlow Nodes

- Because Node.js plugins are typically written in C/C++ languages that adhere to Node.js specifications, they need to be compiled into modules that Node.js can load. This means on the Windows platform, tools such as Visual Studio or MS Build Tool need to be installed to accomplish these building tasks. Users can download these build tools from the Microsoft website to fulfill the dependencies.

- 由于Node.js插件通常是由符合Node规范的C/C++语言编写，因此需要编译成Node.js可以加载的模块。这意味在Windows平台上，需要安装例如Visual Studio或MS Build Tool等构建工具来完成相关工作。用户可以通过微软网站，下载构建工具进行安装，完成依赖条件。

Visual Studio Installer

[已安装](#) 可用

所有安装都是最新的。



Visual Studio Community 2022

17.9.3

功能强大的 IDE，供学生、开放源代码参与者和个人免费使用

[发行说明](#)

修改(M)

启动(L)

更多 ▾



Visual Studio 生成工具 2022 (2)

17.9.3

Visual Studio 生成工具允许生成本机和基于 MSBuild 的托管 .NET 应用程序，而不需要 Visual Studio IDE。还可以选择安装 Visual C++ 编译器和库、MFC、ALT 和 C++/CLI 支持。

[发行说明](#)

修改(M)

启动(L)

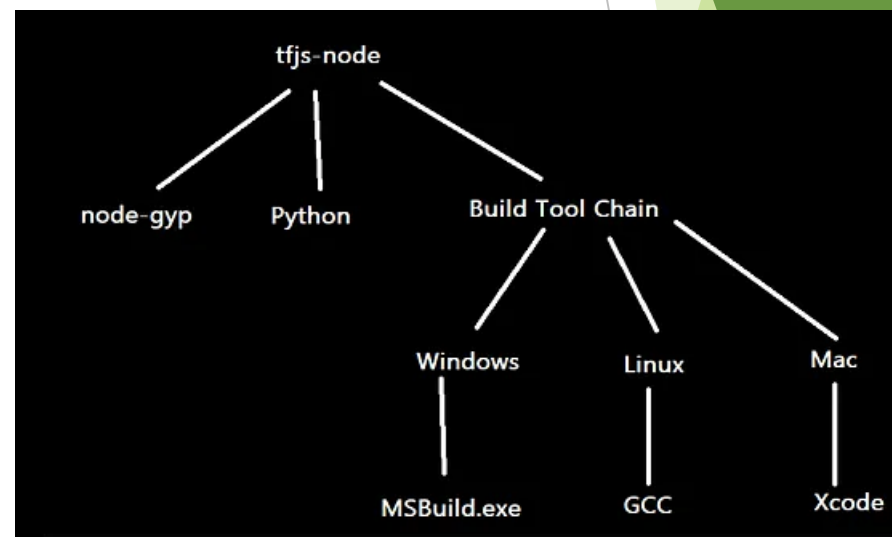
更多 ▾

TensorFlow 节点

TensorFlow Nodes

- 实际上，Windows上的构建过程如果不借助图形化工具，有时还是挺复杂的。因此，Node提供了相应的工具node-gyp，通过配置文件，辅助构建。该构建工具可自动执行构建过程的相关工作，包括处理多种编程语言并打包。例如，右图展示了当处理tfjs节点，即Node.js对TensorFlow的包装，相应的构建依赖关系。

In fact, the build process on Windows can be quite complex without the aid of visual tools. Therefore, Node provides corresponding tool node-gyp to assist with building through configuration files. This tool can automatically perform various tasks related to the build process, including handling multiple programming languages and packaging. The right figure demonstrates when processing the tfjs node, the wrap of TensorFlow, the overall dependencies.



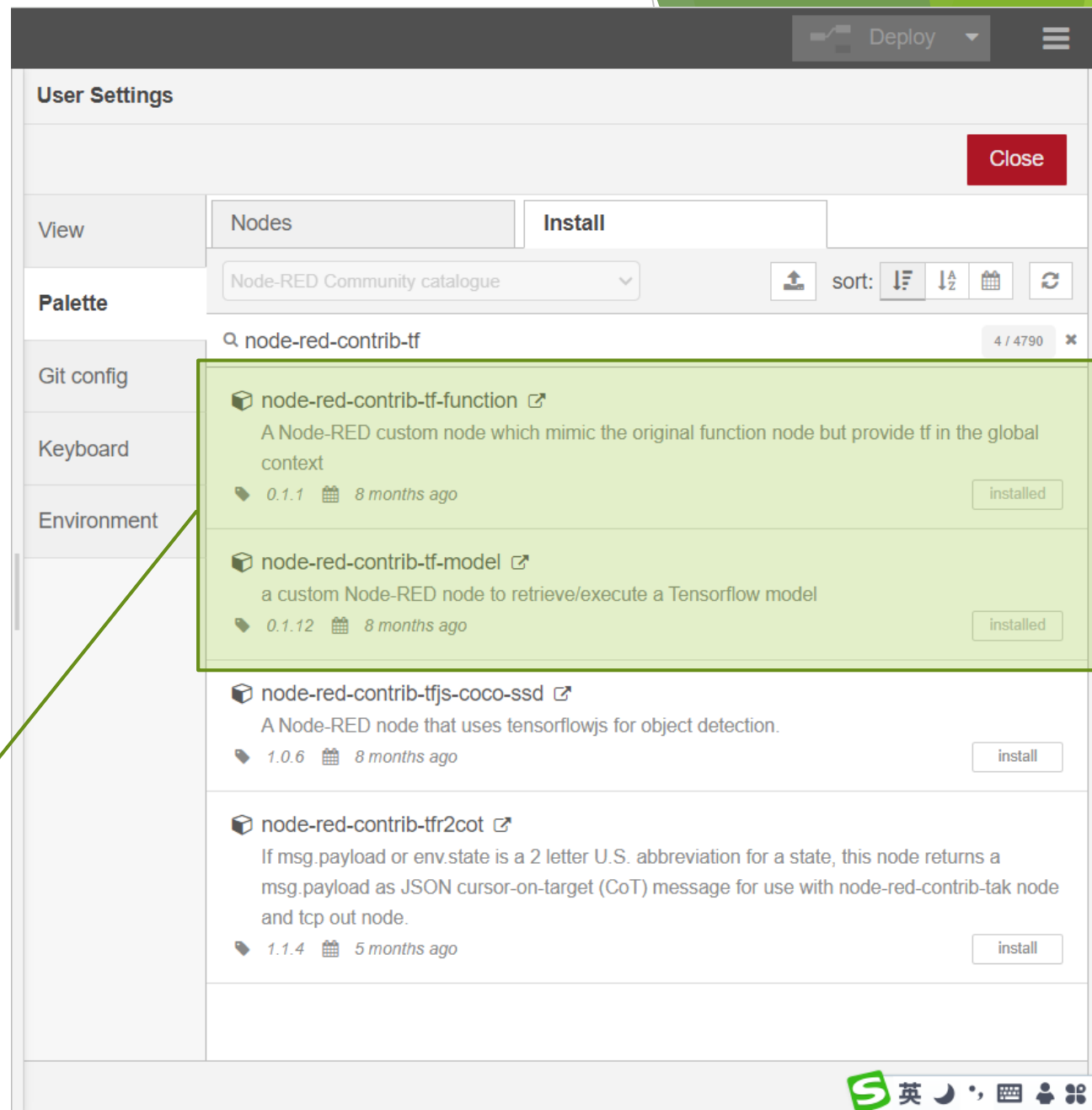
TensorFlow 节点

TensorFlow Nodes

- 使用TensorFlow.js节点的第一步是安装相关依赖，如右图所示：

The first step to use TensorFlow.js is to install the relevant dependency, as indicated in the right figure:

选择图示的两项。
Select these two options as shaded.



TensorFlow 节点

TensorFlow Nodes








- 但由于实际上，上述两个安装包都依赖于 tfjs-node，我们需要首先安装 tfjs-node，解决依赖问题。由于我们不打算基于 Node.js 做太多的开发，因此，我们仅对 Node-RED 进行安装，这意味着我需要进到 .node-red 目录进行安装：

But due to the fact both packages are dependent on the tfjs-node, so we need to tfjs-node first to solve the dependency issue. However, we have no intention to develop based on Node.js, so we just install it in the .node-red folder targeting Node-RED:

```
npm install @tensorflow/tfjs-node-gpu
```

- 由于我们安装的均为较新的版本，因此我们需要重新构建。但为适配新的版本，我们进行安装文件目录，进行一些更改。

But due to the fact all we installed are the most up-to-date version, so we need to rebuild the packet. However, to adapt to the new version, we need to make some modification in the installed package folder.

名称	修改日期	类型
 binding.gyp	2024/3/15 14:43	GYP 文件
 DEVELOPMENT.md	2024/3/15 14:43	MD 文件
 package.json	2024/3/15 21:04	JSON File
 README.md	2024/3/15 14:43	MD 文件
 tsconfig.json	2024/3/15 14:43	JSON File
 tsconfig.test.json	2024/3/15 14:43	JSON File
 WINDOWS_TROUBLESHOOTING.md	2024/3/15 14:43	MD 文件

TensorFlow 节点

TensorFlow Nodes

- ▶ 由于我们安装的Node.js的API版本为9，因此我们要在文件package.json文件中，对生成的tfjs_binding的最新版本增加至9，然后将命令行工作目录切换到tfjs-mode包所在的文件夹，并运行如下命令：

Since the API version for the installed Node.js is 9, so we need to append the newest version number 9 in the package.json, in order to generate the corresponding tfjs_binding file. After that, switch the working directory to the package folder of tfjs-node and execute the following command:

```
npm run build-addon-from-source
```



```
76  "host": "https://storage.googleapis.com/tfjs-tfjs/tfjs-binding-  
77  "remote_path": "./napi-v{napi_build_version}-  
78  "napi_versions": [  
79    3,  
80    4,  
81    5,  
82    6,  
83    7,  
84    8,  
85    9,  
86  ],  
87  },  
88 }
```

TensorFlow 节点

TensorFlow Nodes

- 在此之后，用户可以继续选择用“管理调色板”安装相关节点，或者选用命令行安装相关节点。但选择命令行进行安装时，一定要进入.node-red目录进行安装：

After that, users can choose to install related nodes via “Manage palette” or via terminal. However, if users prefer to install via command line, make sure the current working directory is .node-red:

```
npm install node-red-contrib-tf-model
```

```
npm install node-red-contrib-tf-function
```

- 注意，安装之后，要对Node-RED设置文件settings.js进行修改。具体修改如下图所示：

Notably, after installation, you need to modify the configuration file settings.js, as indicated by the figure below:

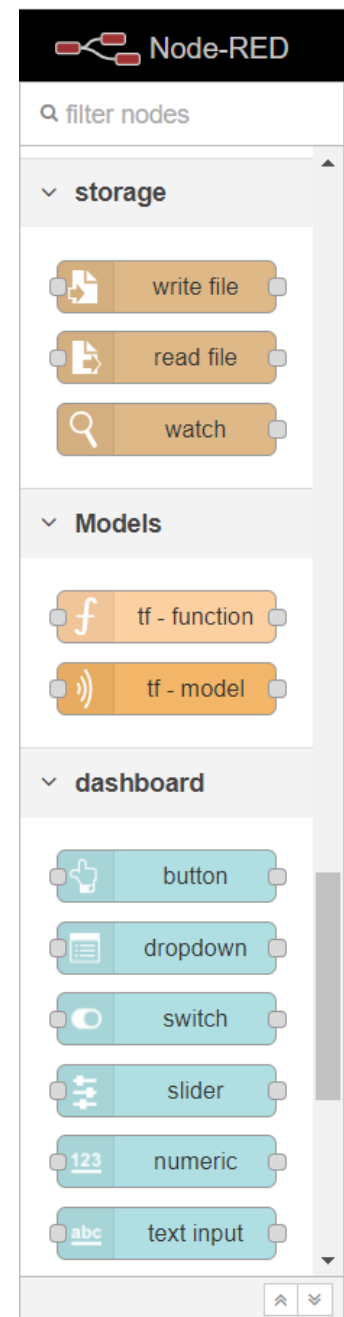
```
functionGlobalContext: {  
  // os:require('os'),  
  tf:require('@tensorflow/tfjs-node'),  
},
```


TensorFlow 节点

TensorFlow Nodes

- 进行上述配置后，原则上便可以使用基于 TensorFlow.js 的相关功能。如果在 Node-RED 启动过程中报错，典型地如模块缺失，则安装对应包即可。如果完全无误，则用户应该能看到如图所示：

After the above configuration, in principle TensorFlow.js is available in Node.js. However, there might still be problems during the startup process of node-red, for example, missing of some modules. The usual solution is to install the packages. If everything goes smoothly, users should be able to see the nodes shown in right figure:

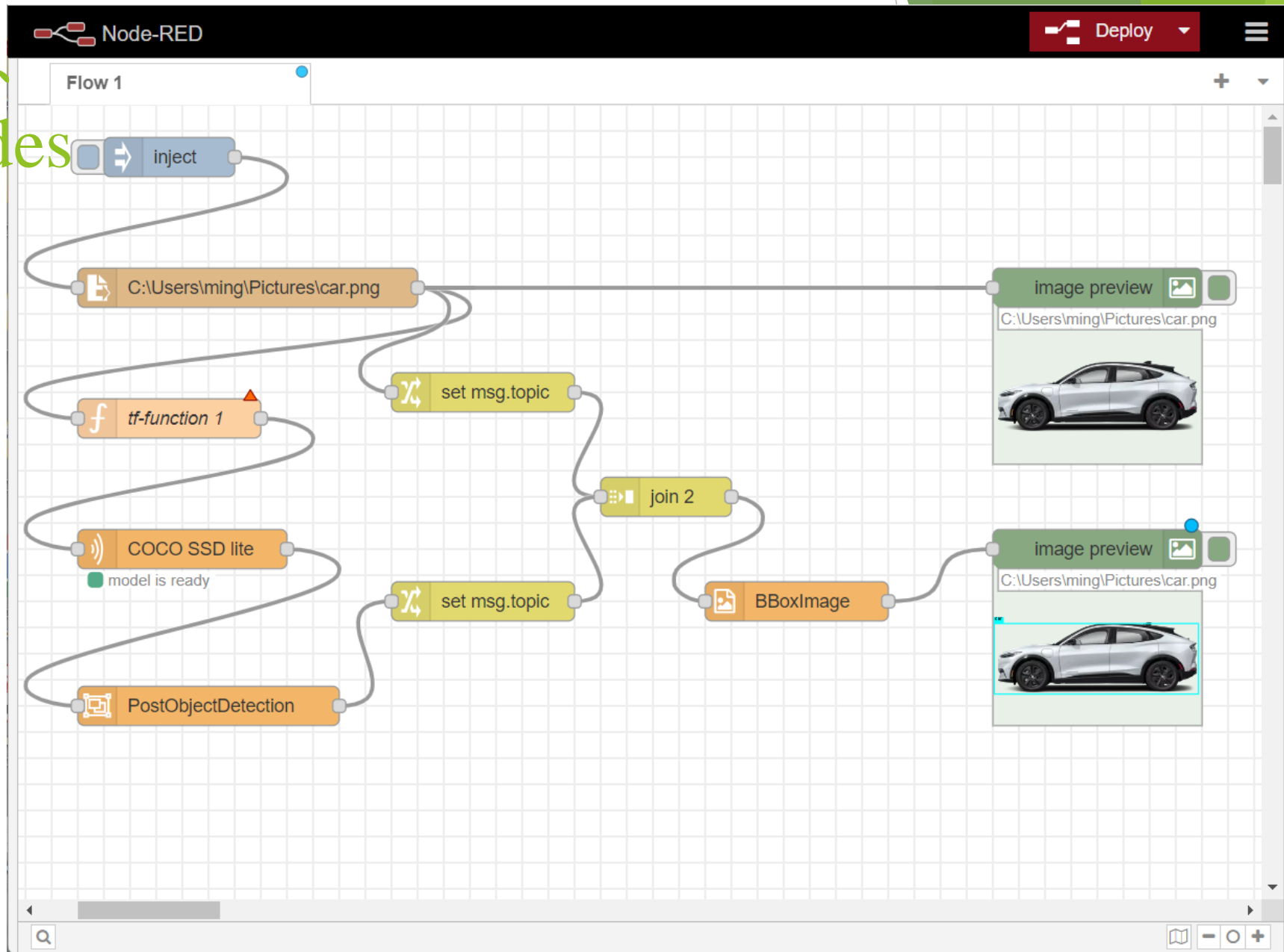


TensorFlow节点

TensorFlow Nodes

- 下面我们以目标识别为例，来介绍如何基于tfjs-node实现智能识别算法。其整体流程图所示：

In the following, by exemplifying with object detection, we introduce some intelligent algorithms based on tfjs-node. The overall flow diagram is shown in the right:

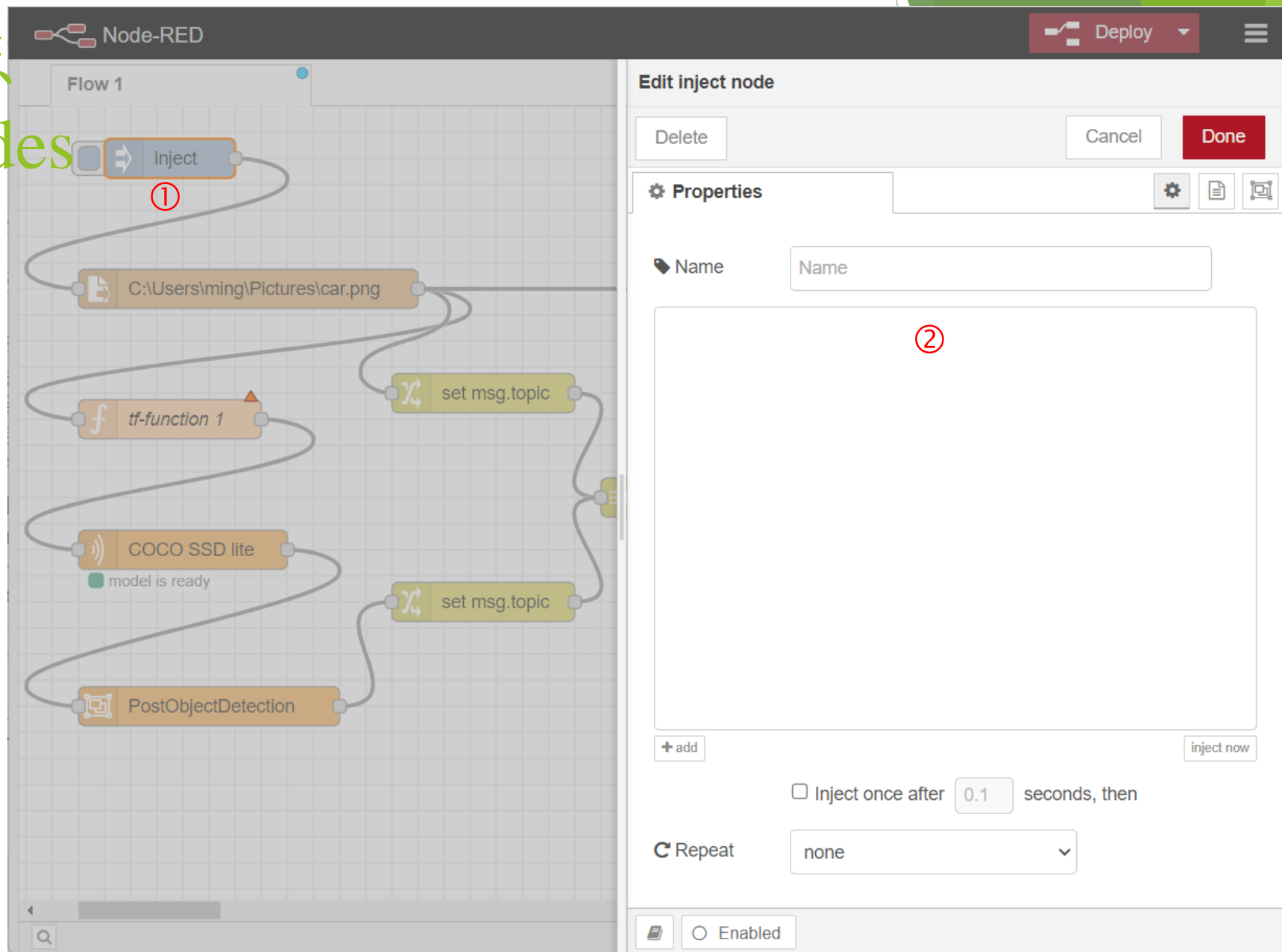


TensorFlow 节点

TensorFlow Nodes

- 我们首先插入“嵌入”节点，但由于主要用于激发流，因此，并不需要其有任何实际输出。

We first insert an inject node, which is used to activate the flow. Therefore, nothing output from it necessary.

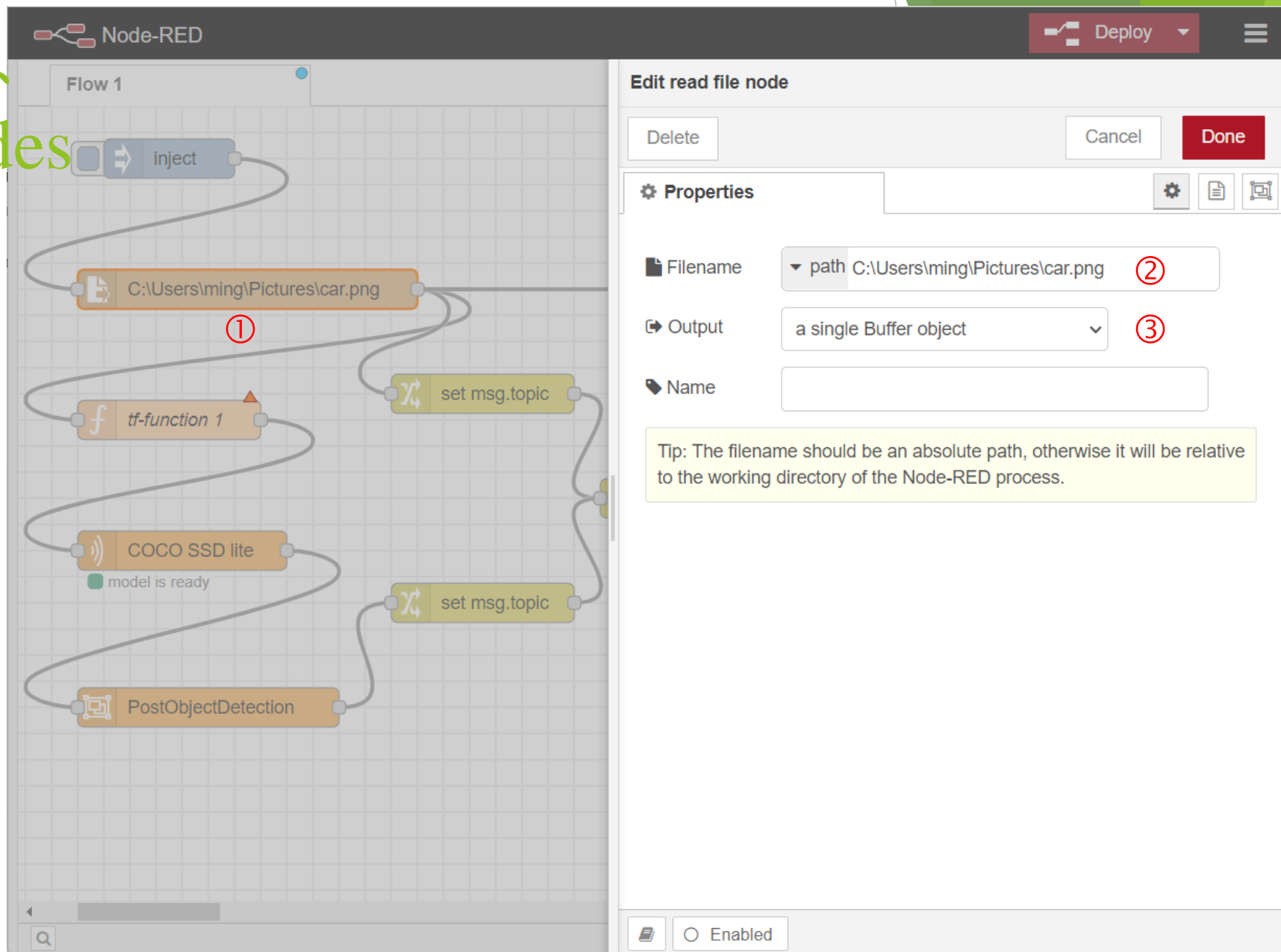


TensorFlow 节点

TensorFlow Nodes

- 我们再次插入“读入”节点，其是文件操作节点。我们需要指定要读入的文件的路径，并将输出设置成单一的缓冲区对象。

We insert the read in node next, which is actually a file node. We need to set the path of image file and the format of output, which is a single buffer.



TensorFlow 节点

TensorFlow Nodes

- 我们接下来插入tf-function节点，其主要的是将读入的图片缓冲区对象解码，并转化成TensorFlow需要的张量格式。

We insert the following tf-function node, which decodes the image buffer, and converts it to the format conformable to the TensorFlow requirement.

The screenshot displays the Node-RED web interface. On the left, a workflow named 'Flow 1' is visible, consisting of an 'inject' node, a file node pointing to 'C:\Users\ming\Pictures\car.png', a 'tf-function 1' node (marked with a red circle ①), a 'COCO SSD lite' node (with a 'model is ready' indicator), and a 'PostObjectDetection' node. On the right, the 'Edit tf-function node' dialog is open, showing the 'On Message' tab. The code editor contains the following JavaScript code (marked with a red circle ②):

```
1 const image = tf.tidy(() => {  
2   return tf.node.decodeImage(msg.payload, 3).expandDims(0);  
3 });  
4  
5 return { payload: { image_tensor: image } };  
6
```

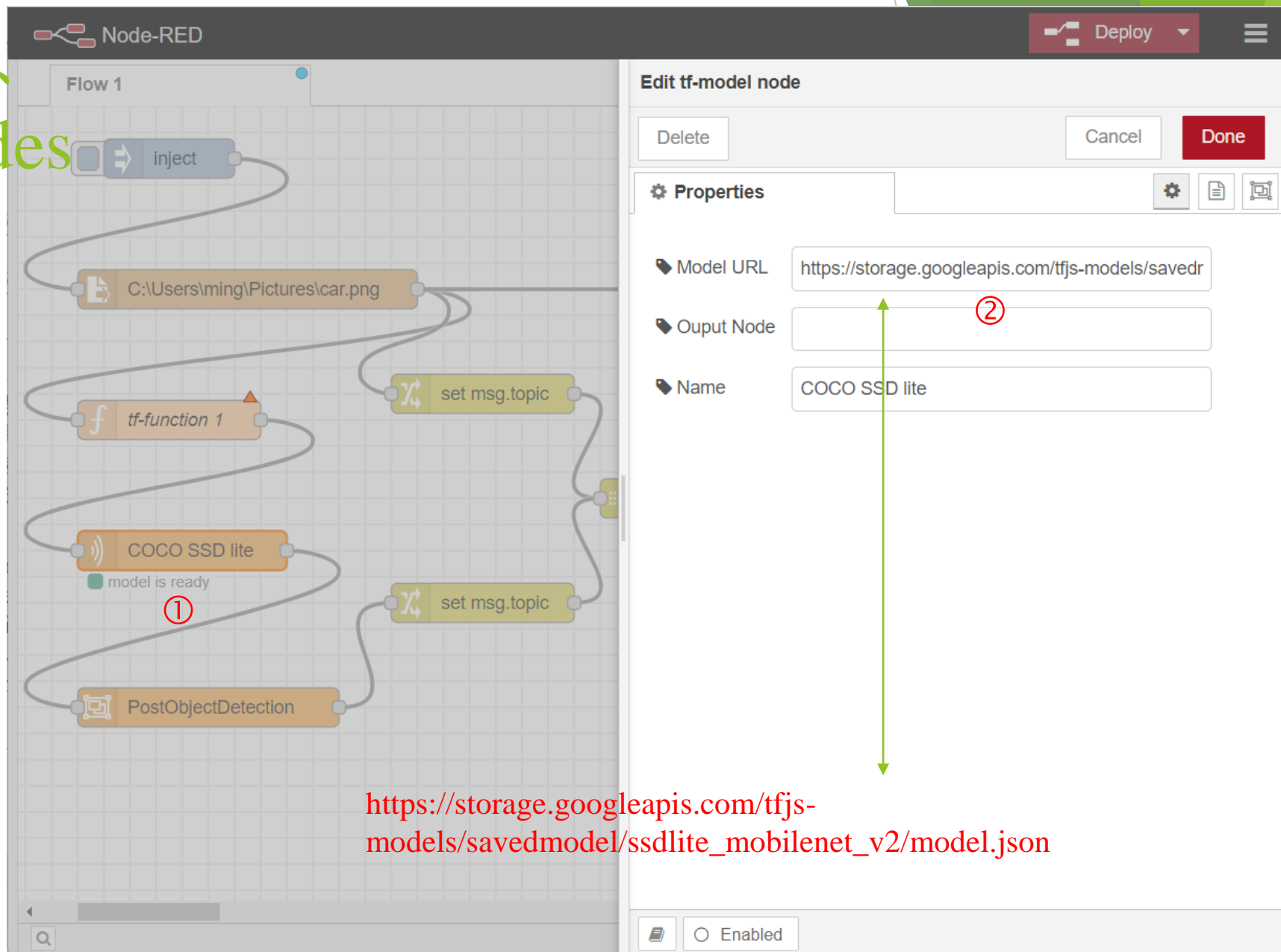
At the bottom of the dialog, there is a checkbox labeled 'Enabled' which is currently unchecked.

TensorFlow 节点

TensorFlow Nodes

- 我们接下来插入tf-model节点，其主要是根据预训练模型，进行目标识别，并输出目标框与目标类别数字。

We insert the following tf-model node, which detects the objects in the target image, and outputs the bounding boxes and label numbers.



TensorFlow 节点 TensorFlow Nodes

- 由于需要将类别数字转变为类别名称，并合并边界框，因此我们需要加入对象检测后处理节点。注意我们需要配置类名称文件的URL。

Due to need to convert label number to label name, and combine the bounding box, we insert the post object detection node. Note we need to configure the URL of class name file.

The screenshot displays the Node-RED web interface. On the left, a flow named 'Flow 1' is visible, containing an 'inject' node, a file input node for 'C:\Users\ming\Pictures\car.png', a 'tf-function 1' node, a 'COCO SSD lite' node (marked 'model is ready'), and a 'PostObjectDetection' node (marked with a red circle ①). The 'PostObjectDetection' node is connected to a 'set msg.topic' node. On the right, the 'Edit post-object-detection node' configuration panel is open. It includes a 'Delete' button, 'Cancel' and 'Done' buttons, and a 'Properties' section. The 'Properties' section contains the following fields:

- Class URL: <https://raw.githubusercontent.com/yhwang/node-red-contrib-tf-model/master/examples/object-detection/classes.json> (marked with a red circle ②)
- IoU: 0.5
- Min Score: 0.5
- Name: Name

A green double-headed arrow points from the 'IoU' field to the 'Min Score' field. At the bottom of the configuration panel, there is a checkbox labeled 'Enabled' which is currently unchecked.

<https://raw.githubusercontent.com/yhwang/node-red-contrib-tf-model/master/examples/object-detection/classes.json>

`npm install node-red-contrib-post-object-detection`

TensorFlow节点

TensorFlow Nodes

- 为了显示侦测效果，需要将边界框与标签加到原图像上，因此我们需要添加如图所示的BBoxImage节点，同时，设置合适的图像与侦测结果属性。

To display the result, a BBoxImage node needs to be inserted to aggregate the bounding boxes and labels. In addition, one needs to configure the properties of image and detected objects.

The screenshot displays the Node-RED web interface. On the left, a workflow in 'Flow 1' is visible, consisting of two 'msg.topic' input nodes connected to a 'join 2' node, which then feeds into a 'BBoxImage' node (labeled with a red circle ①). The 'BBoxImage' node is connected to an 'image' output node. On the right, the 'Edit bbox-image node' configuration panel is open. It includes a 'Delete' button, 'Cancel', and 'Done' buttons. The 'Properties' section contains the following settings:

- Stroke Width: 2 px
- Font Size: 10 px
- Objects: msg.payload.objects (labeled with a red circle ②)
- Image: msg.payload.image (labeled with a red circle ③)
- Name: Name

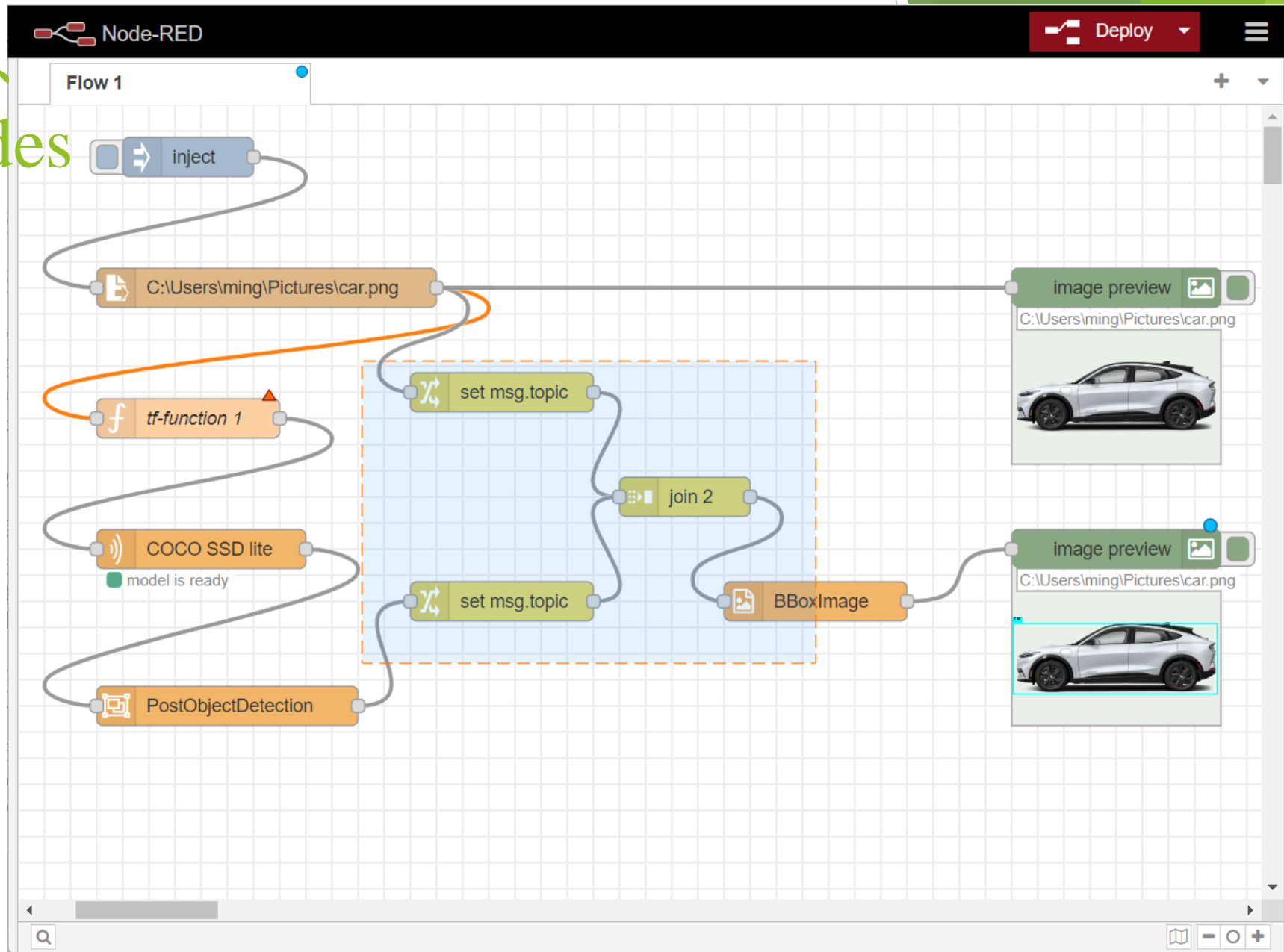
At the bottom of the configuration panel, there is an 'Enabled' toggle switch.

TensorFlow 节点

TensorFlow Nodes

- 由于BBoxImage节点对输入的格式有一定的要求，因此我们需要用join节点将原图像检测结果连在一起。

Due to the requirement of input for BBoxImage node, we need to use a join node to stitch the output from image and that from detection together.



TensorFlow节点

TensorFlow Nodes

- 最后，我们添加图像预览节点，用于预览原图像与加工过后的图像，展示算法工作效果。

At last, we add the image preview node, to preview the image and processed image, to demonstrate how the algorithm works.

The screenshot shows the Node-RED web interface. In the center workspace, a flow named 'Flow 1' contains two 'image preview' nodes. The top node is connected to a message input, and the bottom node is connected to a 'BoxImage' node. Both nodes display a car image. The right sidebar shows the 'Edit image node' configuration with properties like 'msg.payload', 'Width: 160', and 'Name'. At the bottom right, there is a text box with the command: `npm install node-red-contrib-image-output`.