

Production Supporting Systems in Factories

ระบบสนับสนุนการผลิตในโรงงานอุตสาหกรรม

Machine learning

| Image classification

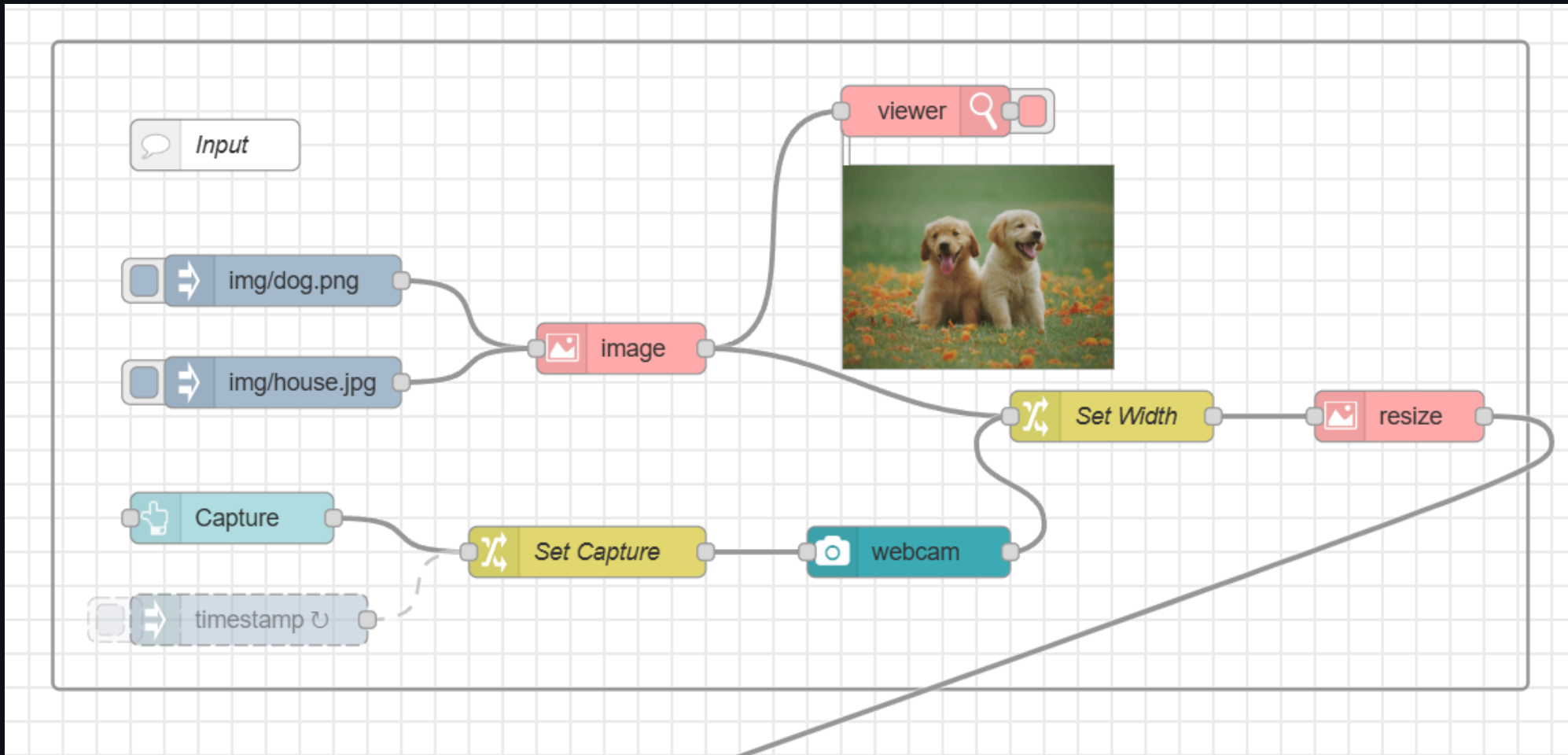
Setting up ML server

- Get [code](#)
- `pnpm install`
- `pnpm run build`
- `pnpm run start`
- Test if server is running.
 - <http://localhost:3003>

Preparing node-red

- `pnpm install node-red-contrib-image-tools node-red-node-ui-webcam node-red-dashboard`

Image input flow



The screenshot displays a Node-RED workflow on the left and the Properties panel for an 'image' node on the right. In the workflow, a 'webcam' node is connected to a 'Set Capture' node, which then connects to an 'image' node. The 'image' node is highlighted with a red rectangle, and a red arrow points from the 'buffer' option in the Properties panel to it. The Properties panel includes fields for Name, image source (set to 'msg. payload'), Function (set to 'none'), Output (set to 'buffer'), and Output Property (set to 'msg. payload').

Properties

Name:

image:
A string containing a file path, URL or base64 image can be used as an image source. NOTE: Passing in an image object is faster as there is no conversion required before processing.

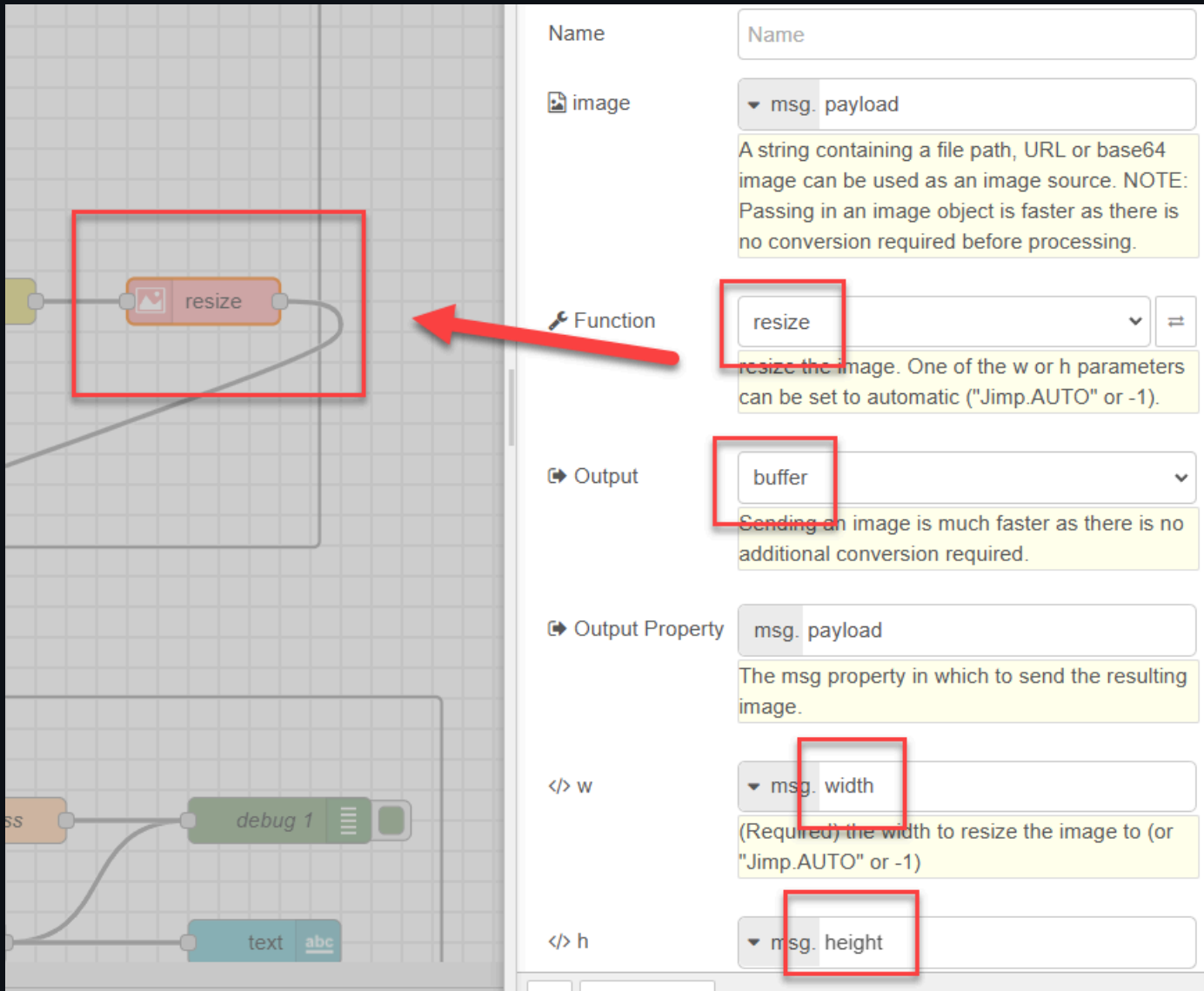
Function:
Just loads the image.

Output:
Sending an image is much faster as there is no additional conversion required.

Output Property:
The msg property in which to send the resulting image.

The image shows a Node-RED interface. On the left, a flow is visible on a grid background. It includes a 'viewer' node (red), a 'Set Width' node (yellow, highlighted with a red rectangle), and a 'webcam' node (teal). The 'Set Width' node is connected to the 'viewer' node. On the right, the 'Properties' panel is open, showing the configuration for the selected 'Set Width' node. The 'Name' field is set to 'Set Width'. Under the 'Rules' section, there are two rules, each highlighted with a red rectangle:

- Rule 1: 'Set' dropdown, 'msg. width' property, 'to the value' dropdown, and '300' value.
- Rule 2: 'Set' dropdown, 'msg. height' property, 'to the value' dropdown, and '-1' value.



The screenshot displays the Node-RED interface. On the left, a flow is visible with a 'resize' node highlighted by a red rectangle. A red arrow points from this node to the configuration panel on the right. The configuration panel for the 'resize' node is shown, with several fields highlighted by red rectangles:

- Name:** A text input field containing the text 'Name'.
- image:** A dropdown menu set to 'msg. payload'. Below it, a note states: 'A string containing a file path, URL or base64 image can be used as an image source. NOTE: Passing in an image object is faster as there is no conversion required before processing.'
- Function:** A dropdown menu set to 'resize'. Below it, a note states: 'resize the image. One of the w or h parameters can be set to automatic ("Jimp.AUTO" or -1).'
- Output:** A dropdown menu set to 'buffer'. Below it, a note states: 'Sending an image is much faster as there is no additional conversion required.'
- Output Property:** A dropdown menu set to 'msg. payload'. Below it, a note states: 'The msg property in which to send the resulting image.'
- </> w:** A dropdown menu set to 'msg. width'. Below it, a note states: '(Required) the width to resize the image to (or "Jimp.AUTO" or -1)'. The 'w' in the label is also highlighted.
- </> h:** A dropdown menu set to 'msg. height'. Below it, a note states: '(Required) the height to resize the image to (or "Jimp.AUTO" or -1)'. The 'h' in the label is also highlighted.

Telegram Mobile Sensor (Backup)

Edit change node

Delete Cancel Done

Properties

Name Set Capture

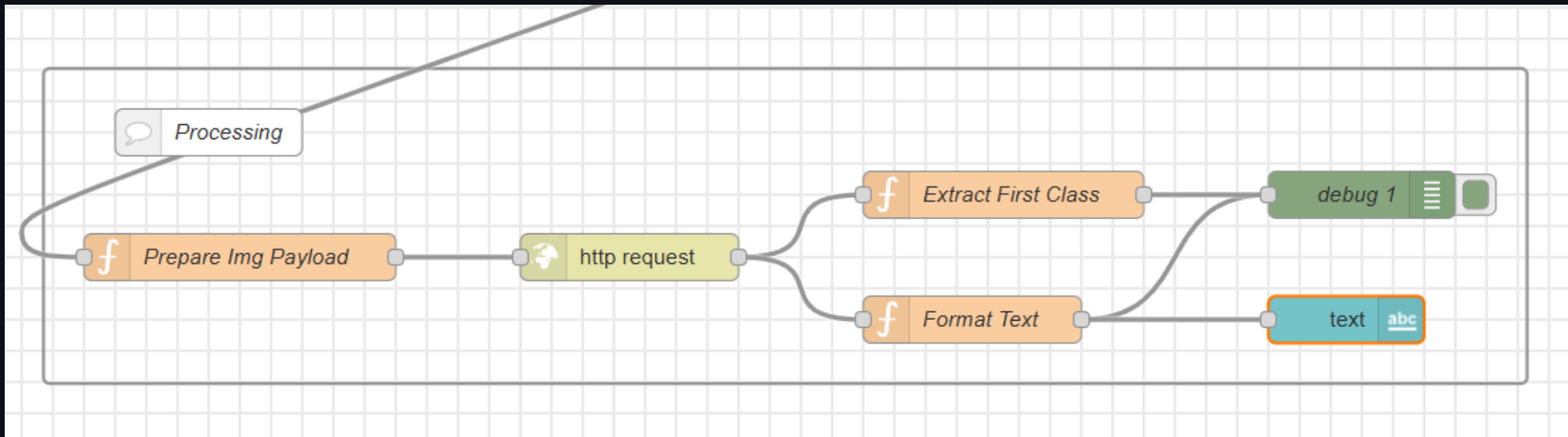
Rules

Set ▼

to the value

- ▼ msg. capture
- ▼ ⌚ true

Image processing flow



function node (Prepare Img Payload)

```
msg.payload = { imageEncoded: Buffer.from(msg.payload).toString("base64") };  
msg.headers = { "Content-Type": "application/x-www-form-urlencoded" };  
msg.url = "http://localhost:3003/upload_base64";  
return msg;
```

function node (Extract First Class)

```
const obj = JSON.parse(msg.payload);  
msg.payload = obj.predictions[0].class;  
return msg;
```

function node (Format Text)

```
const obj = JSON.parse(msg.payload);
let textOut = "";
for (const pred of obj.predictions) {
  const classStr = pred.class;
  const score = pred.score;
  const scoreP = (score * 100).toFixed(1);
  textOut += `👉${classStr} (${scoreP}%) <br/>`;
}
const dt = new Date();
const datestring = dt.toLocaleDateString();
const timestring = dt.toLocaleTimeString();
textOut += `<br/>📅${datestring} ⌚${timestring}`;
msg.payload = textOut;
return msg;
```

Train your own model

- Teachable machine