Caption Generator for Images powered by Watson Visual Recognition

Project report submitted by

Dr. B. Ramani,

Sri Sivasubramaniya Nadar College of Engineering

Introduction:

Image Captioning refers to the process of generating textual description from an image, based on the objects and actions in the image. The ability to recognize image features and generate accurate, syntactically reasonable text descriptions is important for many tasks in computer vision. For example:

(1) Self-driving cars:

Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self-driving system.

(2) Aid for visually challenged:

This can help us create a product for the blind which will guide them to travel on the roads without the support of anyone else. This can be implemented by first converting the scene into text and then the text to voice.

CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents.

Proposed Solution :

This project aims at building an application which takes input as image analyses it and generate the captions in the form of speech. To achieve this, IBM Services like node-red service is used to build a web UI where user uploads a picture. This picture is analyzed by Watson visual recognition service and the analyzed description is then converted in to speech using text to speech service.

Workflow of the project:

The project involves the 3 processes mentioned below.

Process 1: Create IBM Academic Initiative Account

Step 1: IBM Academic Initiative Account

Step 2: Create Node-red Starter Service

Step 3: Create Visual recognition Service

Step 4: Create Text to speech service

Process 2: Build basic node red flow

Step 1: Install browser-utils

Step 2: Install dashboard nodes

Step 3: Nodes to be used

Process 3: Create Node-red Dashboard

Step 1: Insert template node to create Basic html Page

Step 2: Deploy and Open the page

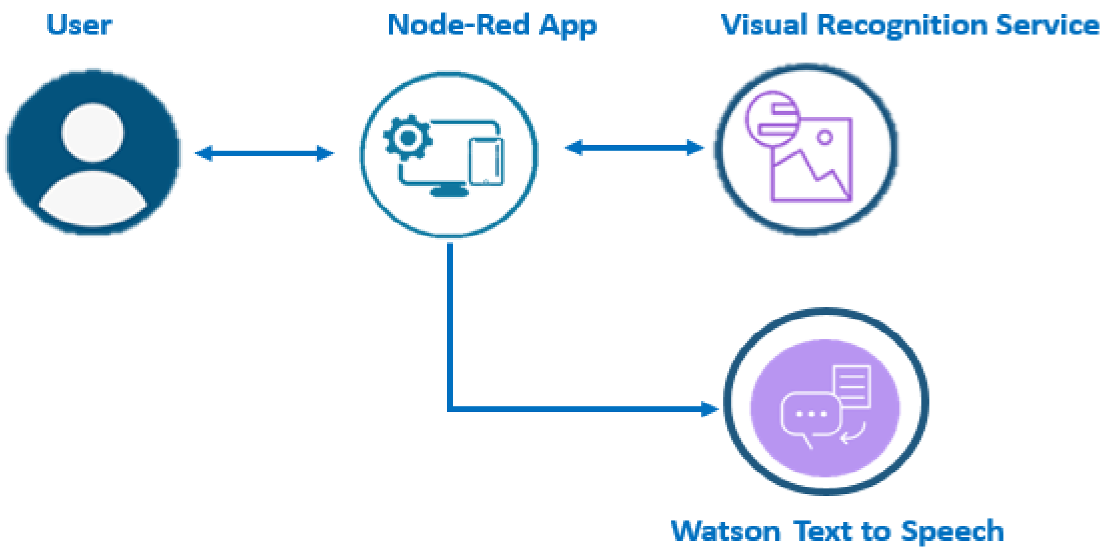


Fig. 1 Work flow of the caption generator

Implementation of the project:

Visual Recognition Service:

The project is implemented by using the pre-built model in Visual Recognition service, which generates class keywords that describe the image. Given any input image, the classes are identified and given a confidence score which is nothing but the probability with which each class is identified.

Text to Speech Service:

The text to speech service enables us the provide speech output for any text given to it as input.

Node-Red Service:

Once the classes in the image is classified, in order have a web UI, node-red service is used. The web UI design is designed such a way that it gets an image as input from the user. This is then analyzed by Visual Recognition service and we obtain the class and the corresponding confidence scores. The caption text to be displayed is decided based on the confidence score and this is fed to the text to speech service to read that text. Fig. 2 shows the different nodes deployed for the web UI using node-red.

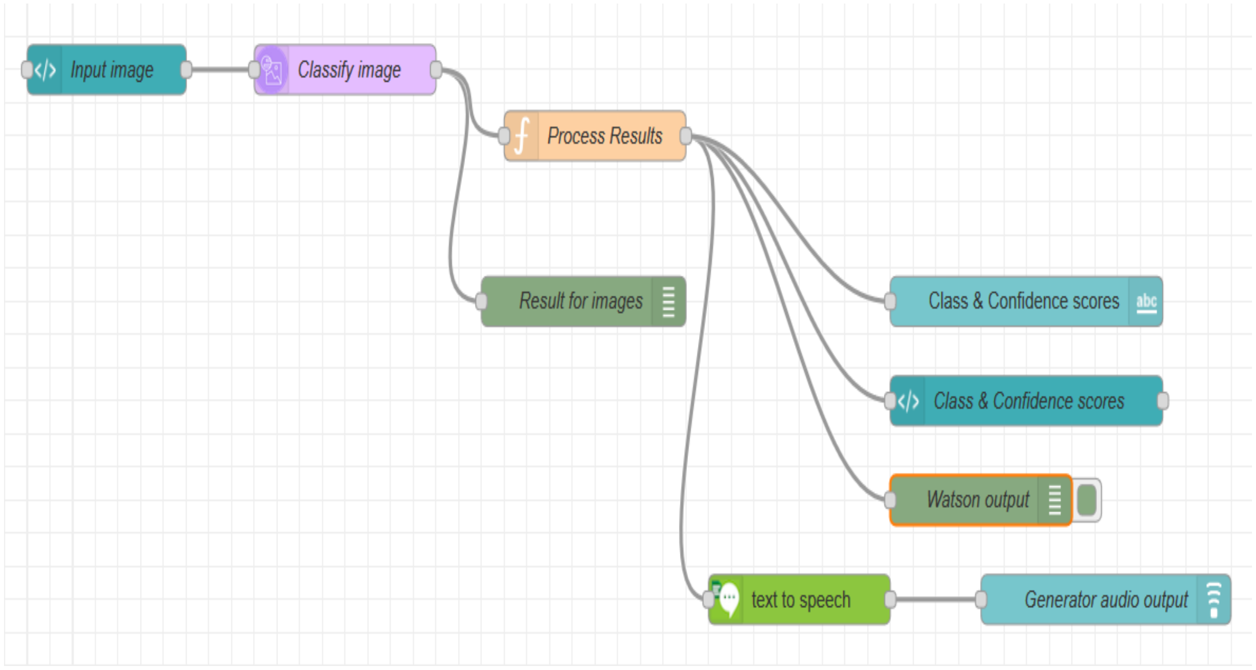


Fig.2 Nodes deployed in node-red

Experimetal Results:

Once the nodes are deployed, the web UI is displayed as shown in Fig. 3.

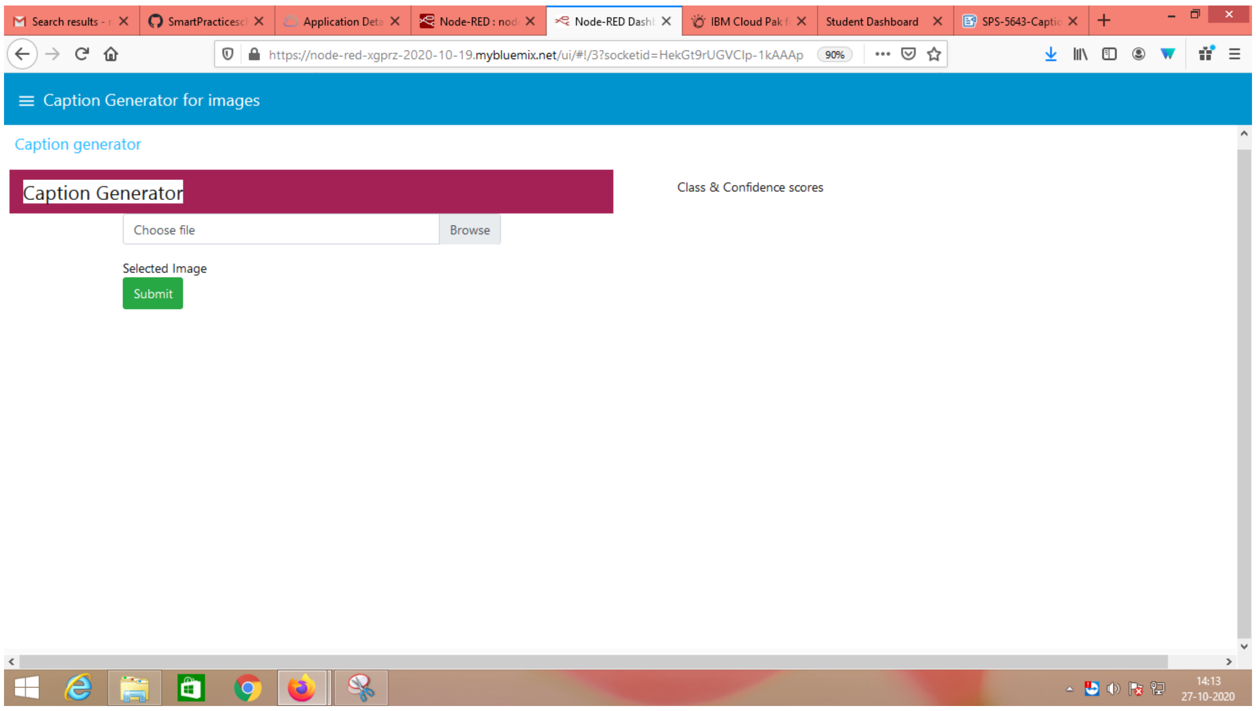
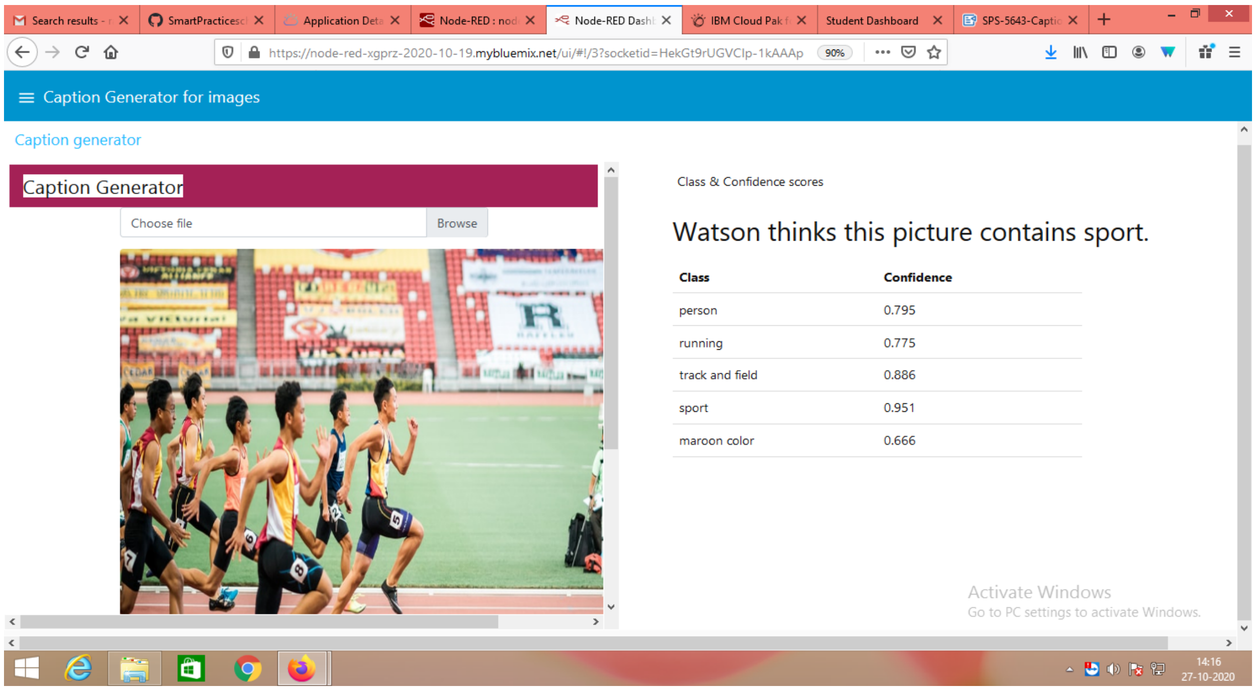


Fig. 3 Deployment of web UI using node-red service

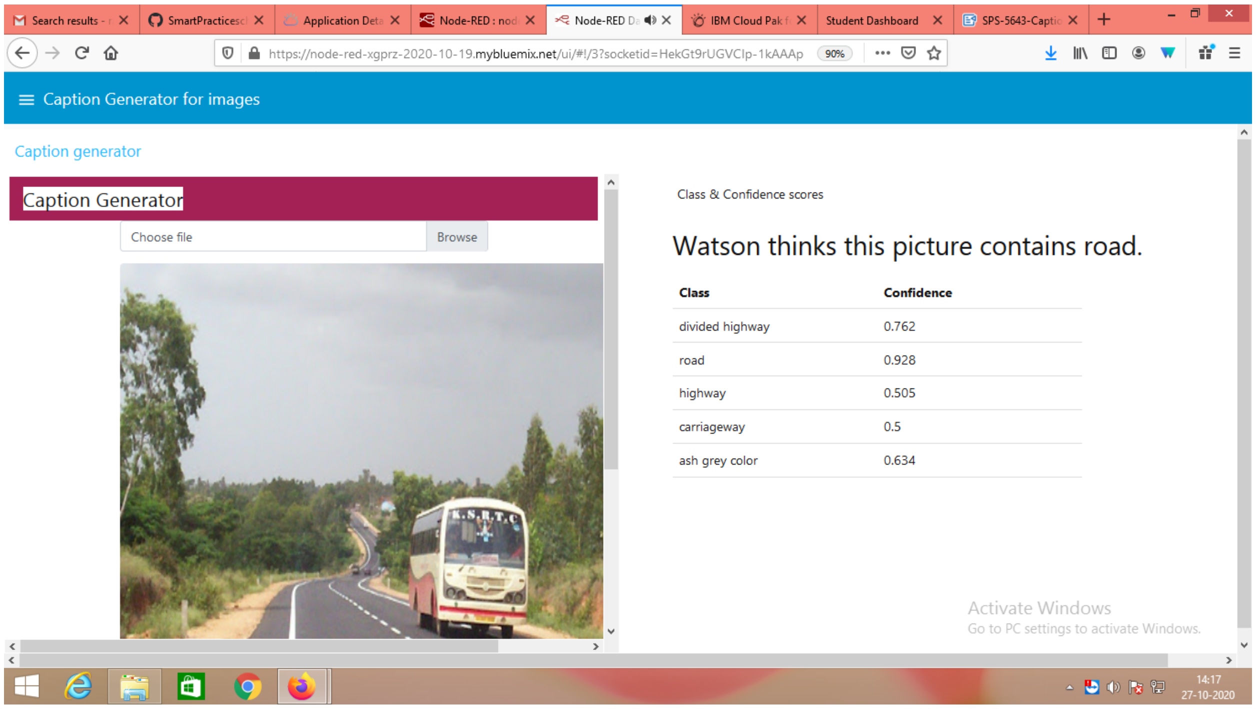
By uploading different images, the results are obtained as shown in Figs. 4-6.



Watson thinks this picture contains sport.

Class	Confidence
person	0.795
running	0.775
track and field	0.886
sport	0.951
maroon color	0.666

Fig. 4 Output for image 1

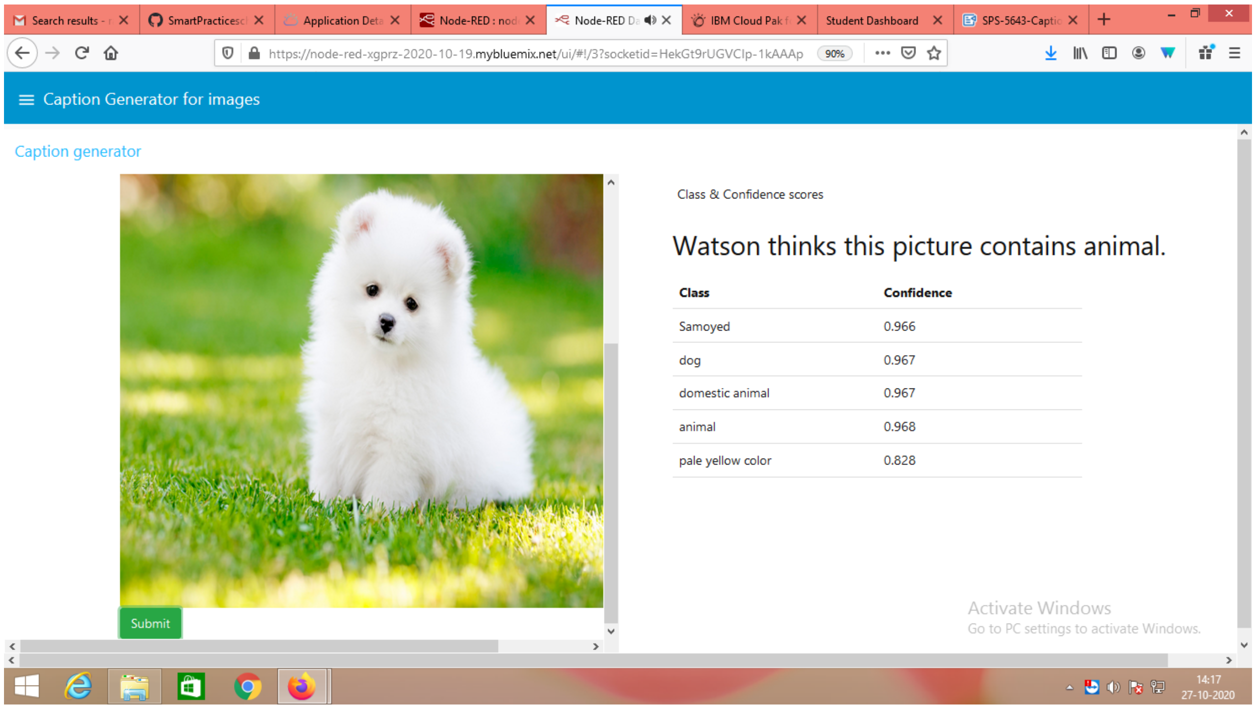


Class & Confidence scores

Watson thinks this picture contains road.

Class	Confidence
divided highway	0.762
road	0.928
highway	0.505
carriageway	0.5
ash grey color	0.634

Fig. 5 Output for image 2


Caption Generator for images
Caption generator
Submit
Activate Windows
Go to PC settings to activate Windows.

Class & Confidence scores

Watson thinks this picture contains animal.

Class	Confidence
Samoyed	0.966
dog	0.967
domestic animal	0.967
animal	0.968
pale yellow color	0.828

Fig. 6 Output for image 3

Conclusion:

The caption generator for images is developed using several Watson services such as visual recognition, text to speech and node-red. This caption generator can be used for several applications such as autonomous driving, aid for visually challenged, alarming the authorities in case of accidents or crime scenes.

Future scope:

The developed caption generator can be made effective by giving the proper caption instead of identifying a single keyword.

Appendix

A. Source code to obtain input image from user

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Visual Recognition</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/boo
tstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquer
y.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/
```

```
umd/popper.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
  <style>
  .bg-light {
    background-color: #A42156!important;
        }
        dashboard-template h4 {
    color: #111111 !important;
    background-color: #A42156 !important;
    }

        .custom-file {
          margin-bottom: 14px;
          }

        .table .thead-dark th {
          color: #fff;
          background-color: #A42156;
          border-color: #A42156;
        }
        h5{
          text-align: center;
            color: #EB9E30;
          }
          .text-center {
            text-align: center;
          }
```

```
            .imgdiv
            {
               align:left;
            }
            </style>
</head>
<body>
            <nav class="navbar navbar-expand-sm bg-light">
               <div class="justify-content-center">
                        <h4 class="text-center">Caption Generator</h4>
               </div>
            </nav>
            <br><br>
    <div class="container">
               <div class="row">
                        <div class="col-sm-2">
                        </div>
                        <div class="col-sm-8">

                                        <div class="custom-file">

                                                  <input type="file" name="pic"
accept="image/*" onchange="readURL(this);" class="custom-file-input"
id="customFile">

                                                  <label class="custom-file-label"
for="customFile">Choose file</label>
                                        </div>
                                        <br>
                        <div class="imgdiv">
                            <img src="#" id="blah" class="rounded" alt="Selected Image">
                        </div>
                        <!--<md-button ng-click="send({payload:action()})">
                            Predict
                        </md-button>-->
                        <button type="submit" ng-click="send({payload:action()})" class="btn btn-success">Submit</button>


                </div>

</body>
</html>
<script>
var x="";
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();

            reader.onload = function (e) {
                $('#blah')
                    .attr('src', e.target.result)
                    .width(700)
                    .height(700);
            };

```

```
                reader.readAsDataURL(input.files[0]);
                x= input.files[0]


            }
        }
        function getdata(data)
            {
                var html = '';
                if(data != 0)
                    {
                                $.each(data, function(i){
                                    var row = data[i];
                                    console.log(row);
                                    html += '<tr>';
                                    html += '<td>';
                                    html +=  row.class;
                                    html += '</td>';
                                    html += '<td>';
                                    html +=  row.score;
                                    html += '</td>';
                                    html += '</tr>';
                            });
                            }
                            else
                                    html+="<div>No Data</div>";
                            $('#scoretable').html(html);
                }
           (function(scope) {
            scope.$watch('msg.payload', function(data) {
                console.log('Position 2');
                console.dir(data);
                getdata(data);

            });
        })(scope);
// or overwrite value in your callback function ...
this.scope.action = function() { return x; }

</script>
```

B. Source code to process the results obtained from Visual Recognition service

```
if (typeof msg.result == 'undefined') {
    return null;
}

// Text Extraction
if (typeof msg.result.images[0].text != 'undefined') {
    var image_text = msg.result.images[0].text;
    msg.payload = image_text;
    msg.template = image_text;
    if( image_text.length >0 ) {
       msg.template= "Watson found the words: "+image_text;
    }
    return msg;
}
```

```
var c_id = 0;
var say = "";
var item;

for ( c_id=0; c_id < (msg.result.images[0].classifiers.length); c_id++ ){
    for( i =0; i<(msg.result.images[0].classifiers[c_id].classes.length); i++ ){
   var object = msg.result.images[0].classifiers[c_id].classes[i].class;
     }
  var bestItem = 0;
  var itemScore = 0;
  for( i =0; i<(msg.result.images[0].classifiers[c_id].classes.length); i++ ){
   var object = msg.result.images[0].classifiers[c_id].classes[i].class;
   if ( !object.includes("color") ) {
    if( msg.result.images[0].classifiers[c_id].classes[i].score > itemScore){
      bestItem = i;
      itemScore = msg.result.images[0].classifiers[c_id].classes[i].score;
    }
   }
```

```
    }
    item = msg.result.images[0].classifiers[c_id].classes[bestItem].class;
    say = say + " Watson thinks this picture contains " + item +".";
}
msg.payload =  say;

var picInfo = msg.result.images[0].classifiers[0].classes;
var arrayLength = picInfo.length;

msg.template="<style>h4 { text-align: center; margin: 10px; }";
msg.template=msg.template+"table {    width: 500px; margin-top: 10px; }";
msg.template=msg.template+"th, td { padding: 8px; text-align: left; border-bottom: 1px solid #ddd; background-color: #FFFFFF; width: 50%;}";
msg.template=msg.template+".classifier {background-color: rgb(85,150,030);text-align: center;}";
msg.template=msg.template+".title { background-color:LightGrey;}</style>";

msg.template=msg.template+"<h2>"+say+"</h2><table span=100%><tr><th>Class</th><th>Confidence</th></tr>";
for (var i = 0; i < arrayLength; i++) {
  msg.template = msg.template + "<tr><td>" + picInfo[i].class + "</td><td>" + picInfo[i].score + "</td></tr>";
```

```
}
msg.template = msg.template + "</table>"


return msg;
```

Node-red URL for web UI:

https://node-red-xgprz-2020-10-19.mybluemix.net/ui/#!/3?socketid=ud4MRnsZFGdStlDmAAAs