**Traffic Sign Classifier**

**Write up** August 10th, Kenta Kumazaki

**1. Purpose**

The goals / steps of this project are the following:

* Load the data set
* Explore, summarize and visualize the data set
* Design, train and test a model architecture
* Use the model to make predictions on new images
* Analyze the softmax probabilities of the new images
* Summarize the results with a written report

**2. Output**

* Make a pipeline that run deep learning to classify traffic signs.  
  I used the Workspace in Udacity for this project.
* Reflect on my work in a written report.

**3. Submission**

**(1) GitHub**

<https://github.com/kkumazaki/Self-Drivig-Car_Project3_Traffic-Sign-Classifier-Project.git>

**(2) Directory**

<folder: main>

* **Writeup\_of\_Lesson14.pdf**: This file
* **Traffic\_Sign\_Classifier.ipynb**: Pipeline file (Jupyter Notebook)
* **Traffic\_Sign\_Classifier.html**: Pipeline file (HTML)
* Image files are saved as following:

**<folder: Test\_pics>**

Additional Test images downloaded by the following site.

<https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/version/1>



**<folder: Test\_results>**

Modified Test images for Traffic Sign Classification.



**3. Reflection**

**(1)Description of my pipeline and results**

My pipeline consisted of 4 steps as following, including the preparation Step: 0.

Step 0: Load the Data

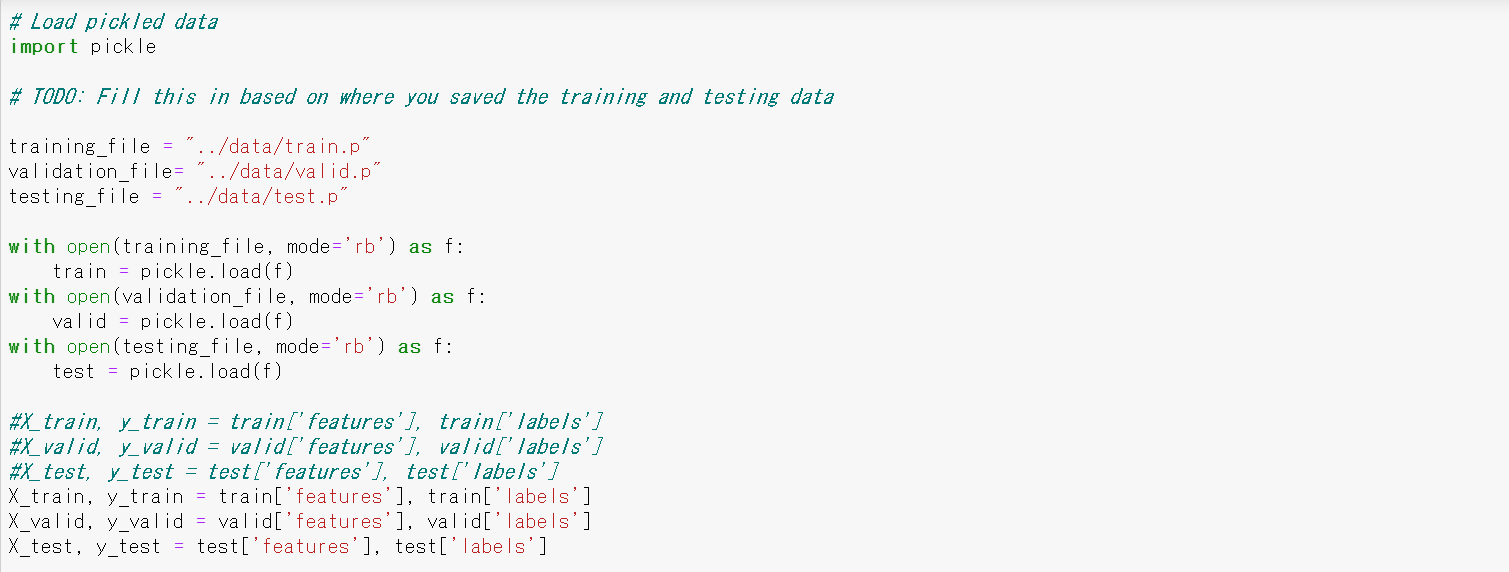
Step 1: Dataset Summary & Exploration

Step 2: Design and Test a Model Architecture

Step 3: Test a Model on New Images

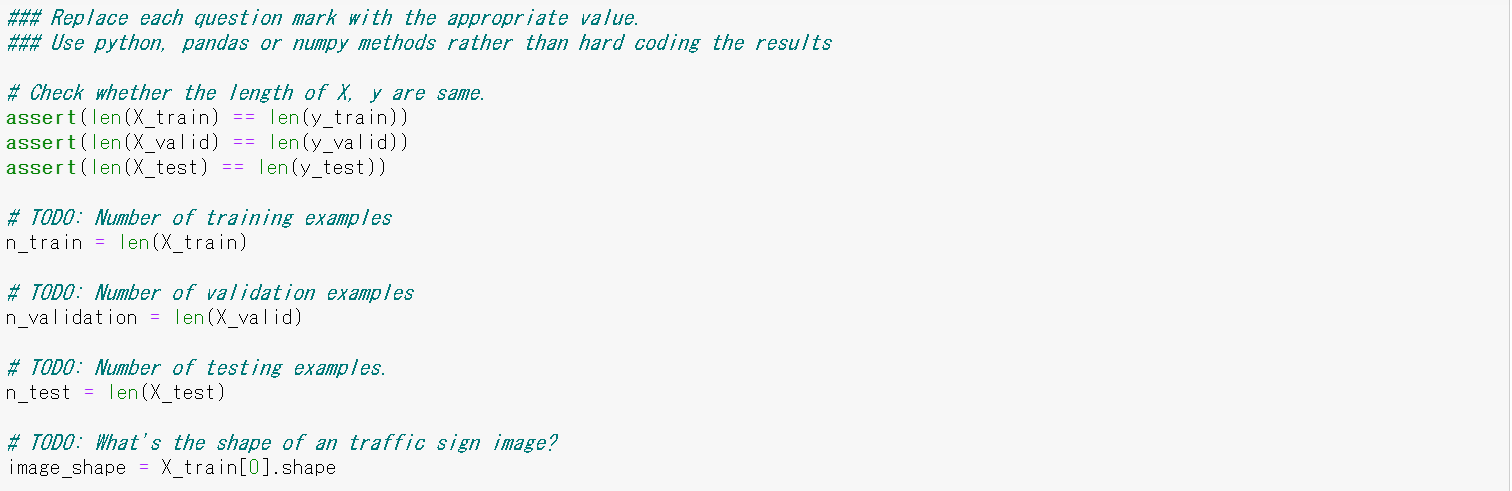
**Step 0: Load the Data**

Load the pre-uploaded datasets at the workspace.



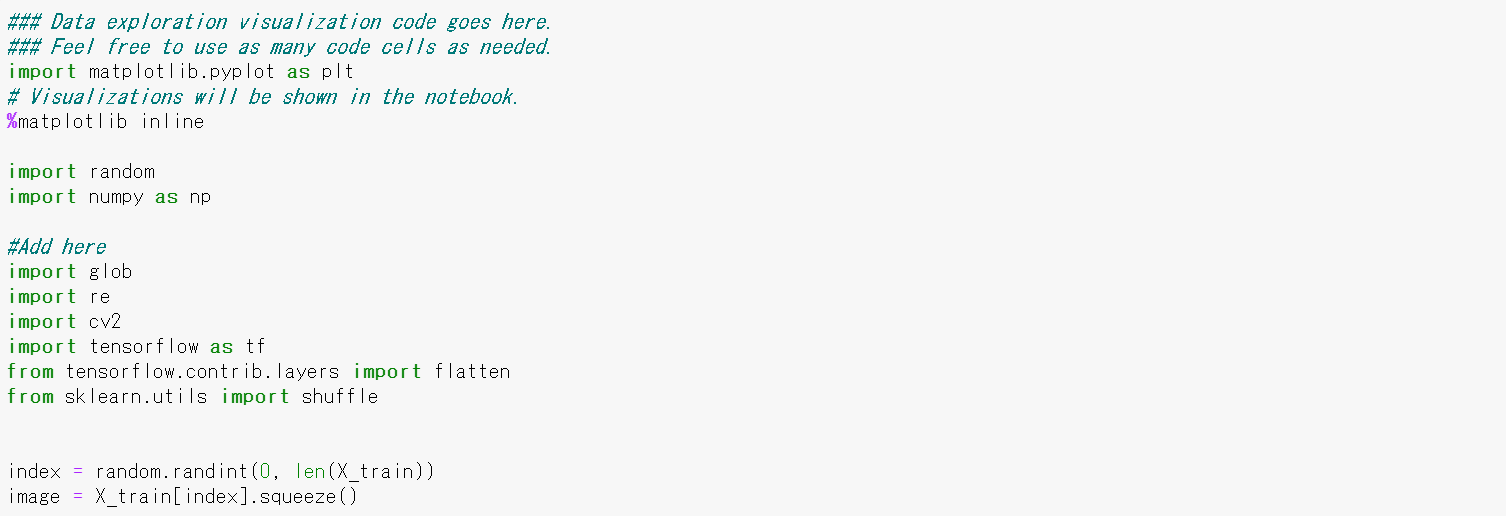
**Step 1: Dataset Summary & Exploration**

At first, I check the dimensions of each necessary data.





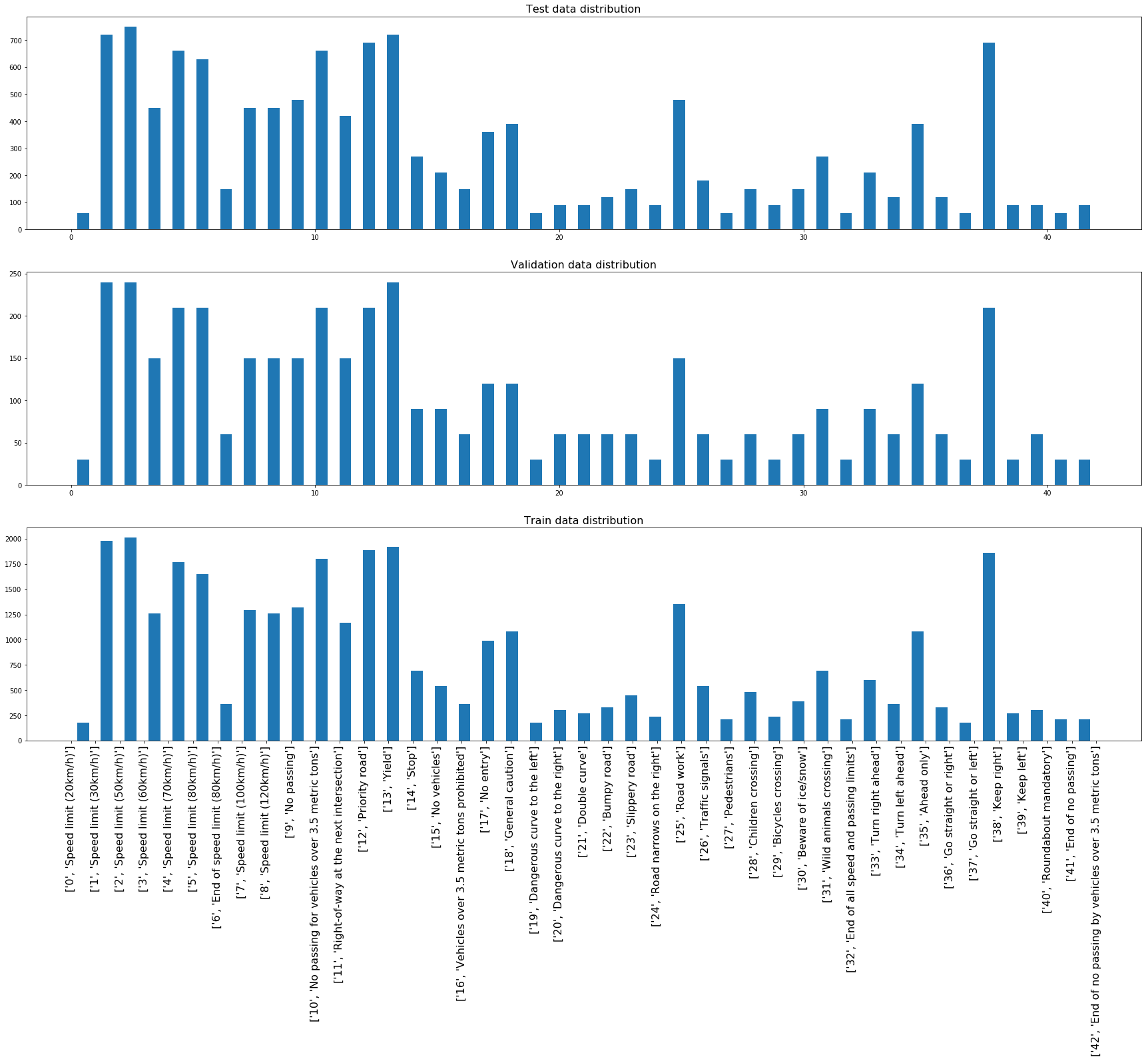
Next, I visualize the datasets with bar graphs.

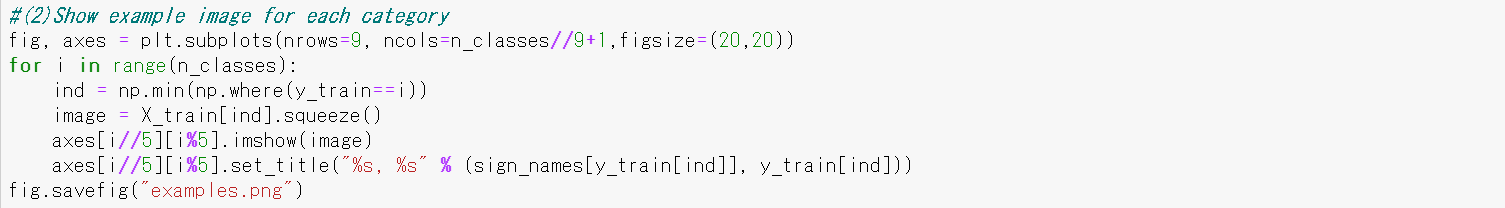




The distributions of Test Data, Validation Data and Tran Data are shown below.

There’s big disperse between each class (0 ~ 42), but there’s small disperse between each datasets,   
so it will be OK to proceed.





The following images are examples of Traffic Signs.



**Step 2: Design and Test a Model Architecture**

First in the Step2, I pre-process the Data Set.

I proceeded with the following steps:

[Test1] I trained the model without any preprocess nor any change from the original model,

and the Validation Accuracy was just 89.2%. Target Accuracy is 93%, so it’s not good enough.

[Test2] I made the images from color to gray scale, and normalized them.

Validation Accuracy became 92.5%, but it’s not good enough.

[Test3] I added training data by rotating the original training data by +15/-15 degree.

The number of training data became triple from original, which is 104,397.

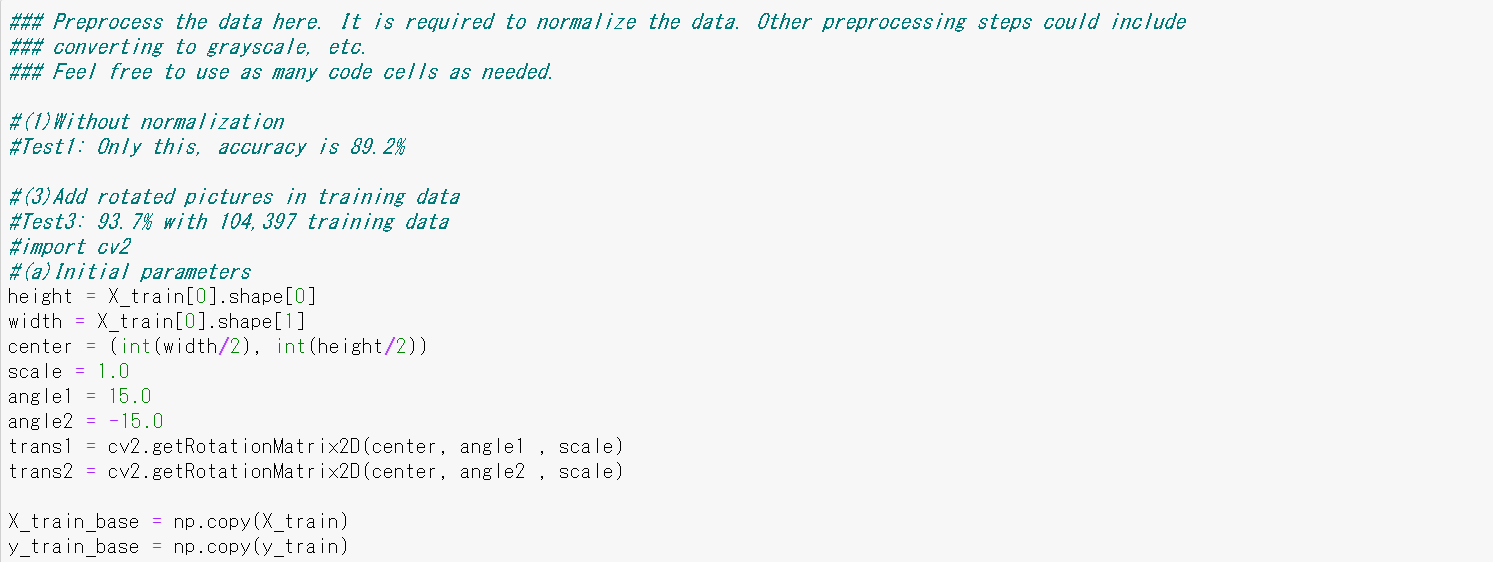
Validation Accuracy became 93.7%, but it’s better to make it more robust.

[Test4] I added training data by zooming up/down the original training data by +2/-2 pixels.

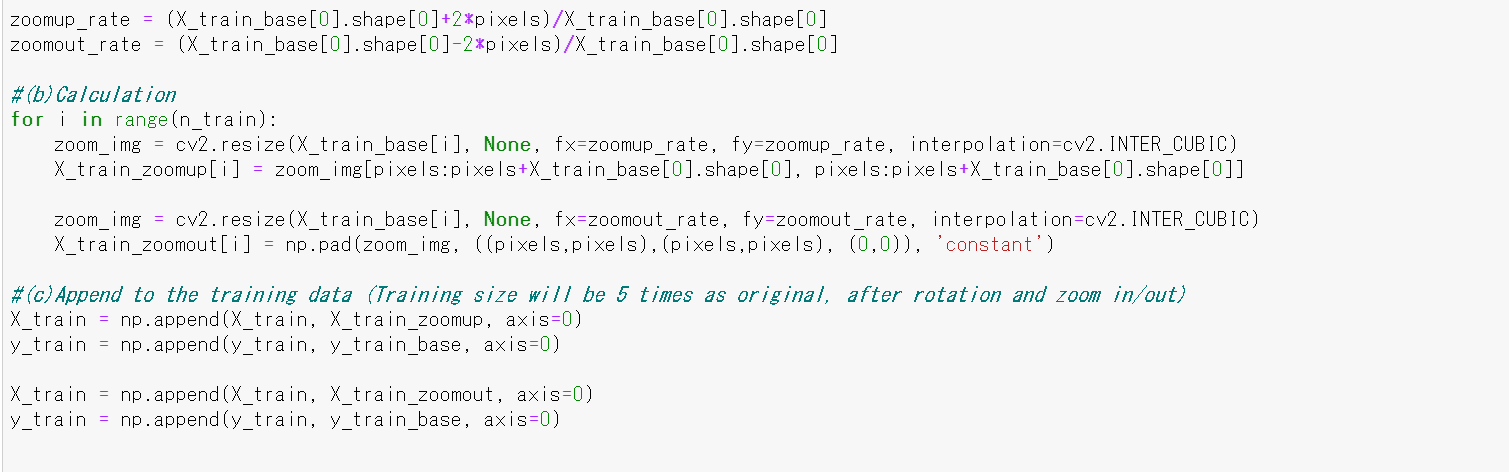
The number of training data became five times from original, which is 173,995.

Validation Accuracy became 94.3%, but it’s not good enough because Test Accuracy was 92.4%.

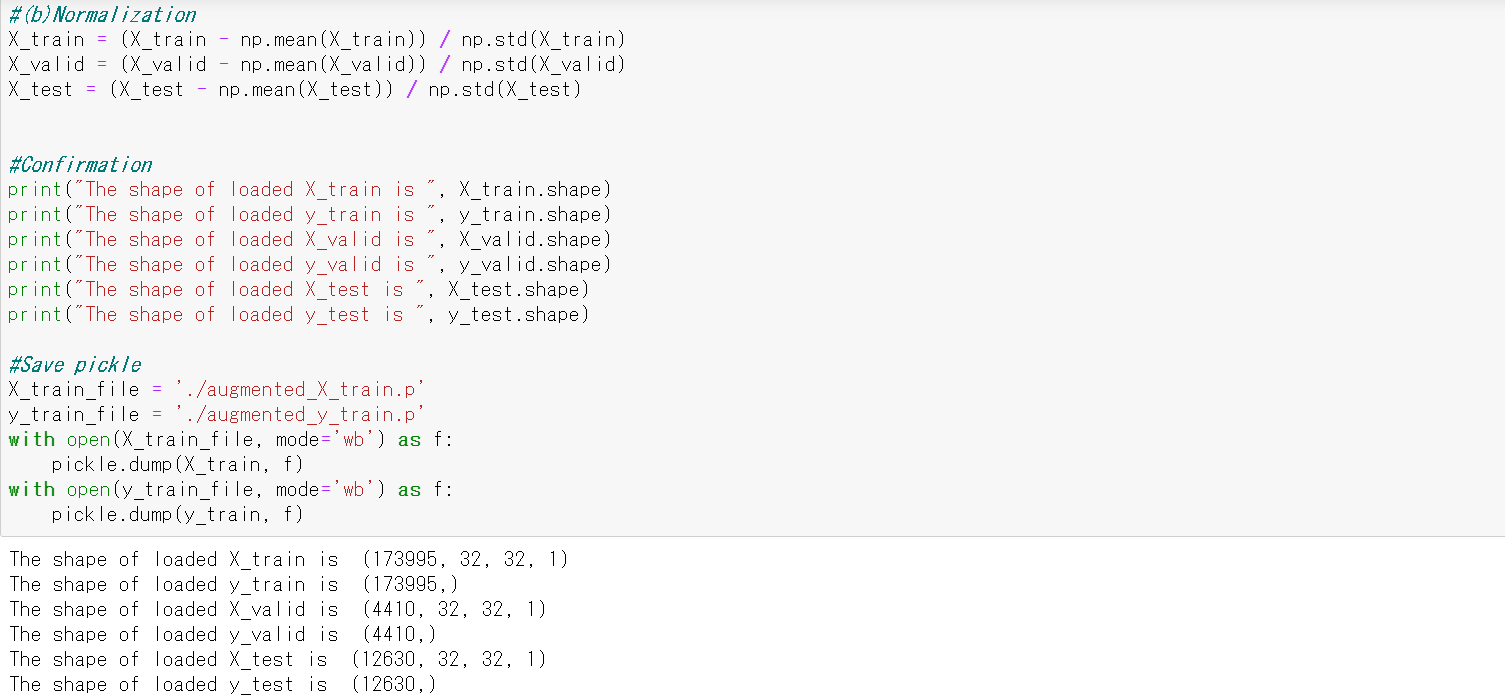
I found that it’s better to change the parameters of the model to get better results.







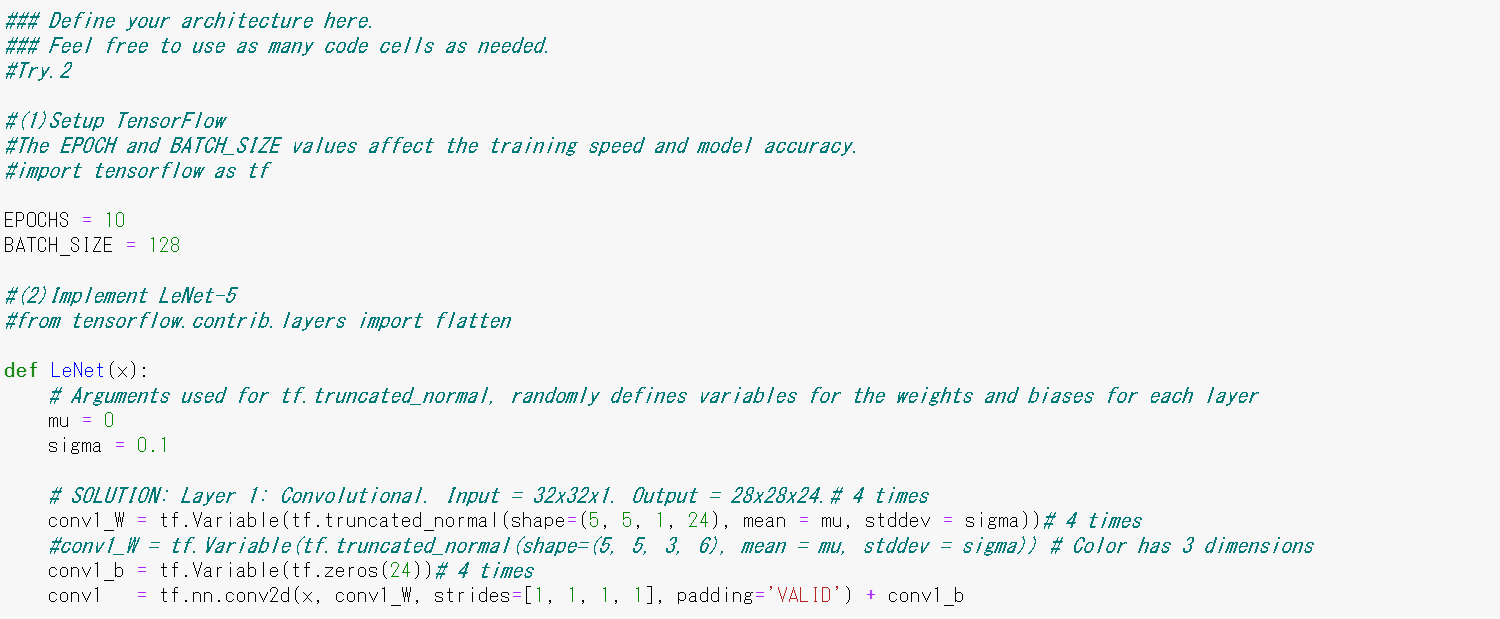


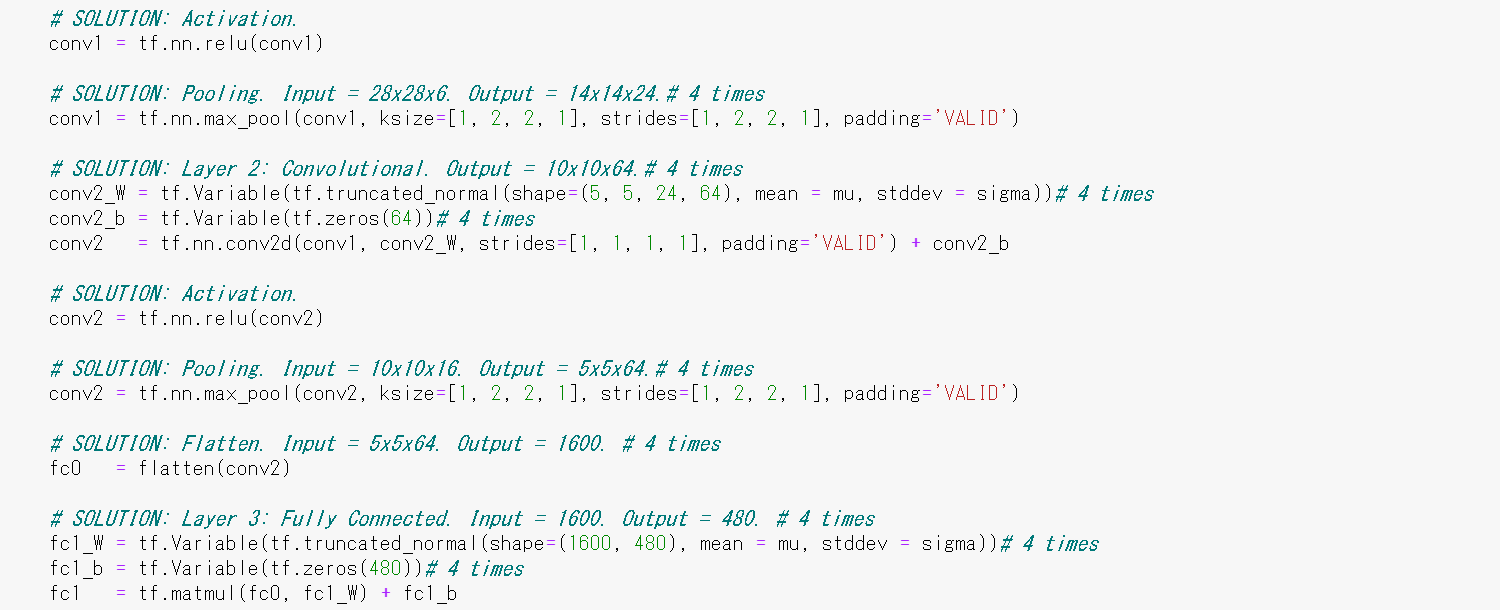


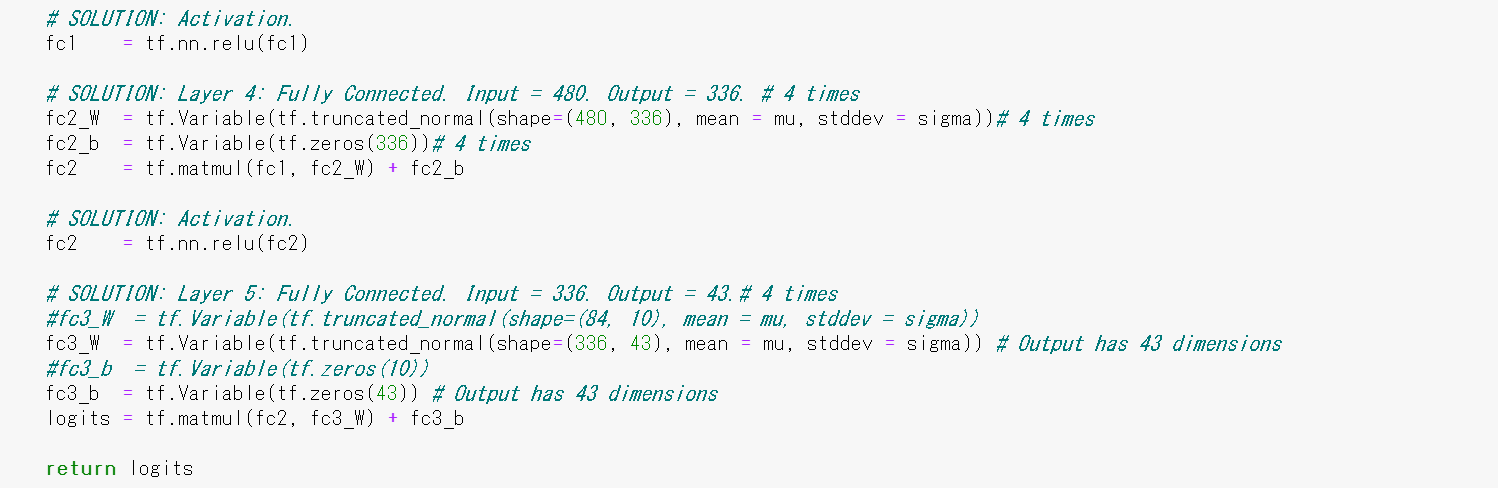
The following code shows the final model of this project.

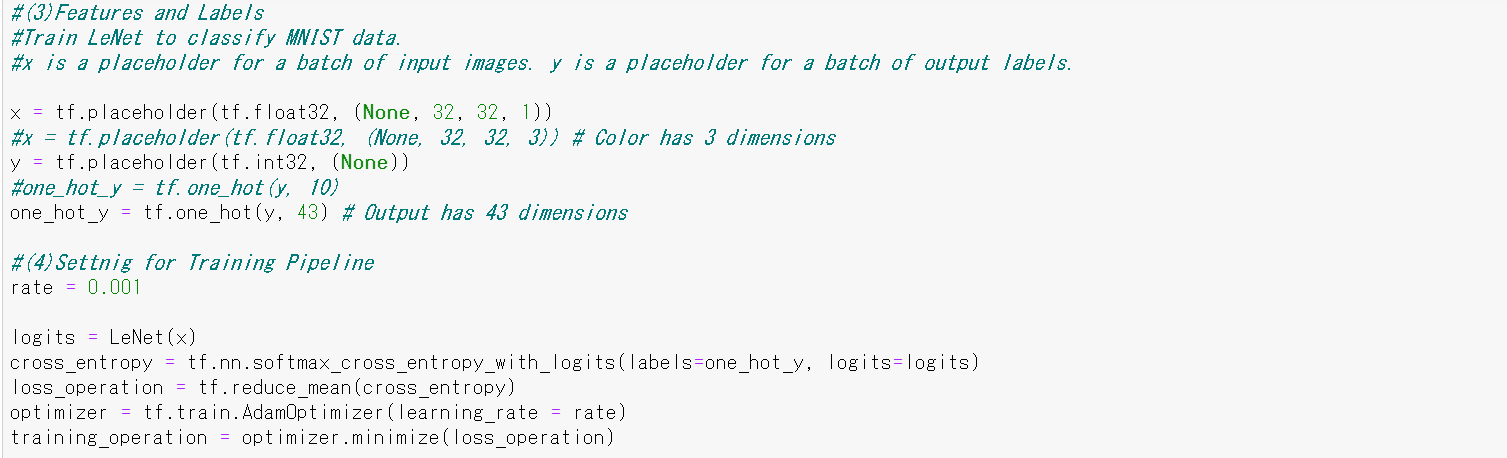
[Test5] The original LeNet has output dimension: 10 (0, 1, 2, …, 8, 9), but our project has **output dimension 43**.

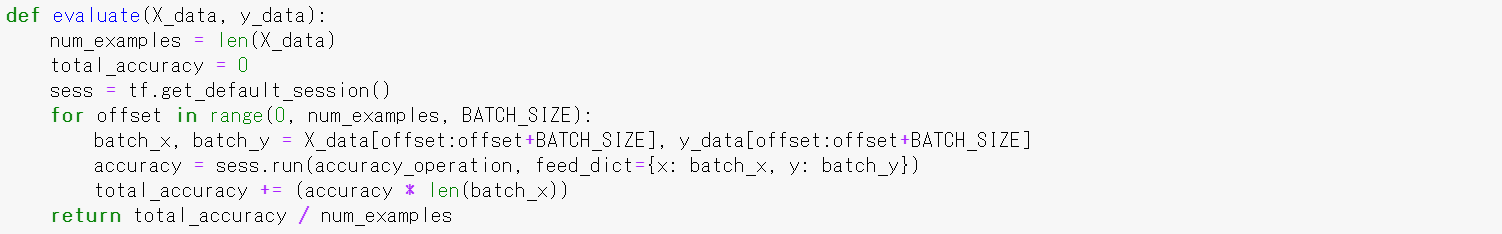
It’s almost 4 times bigger than original one, so I made the depth of the hidden layers 4 times bigger   
than original.

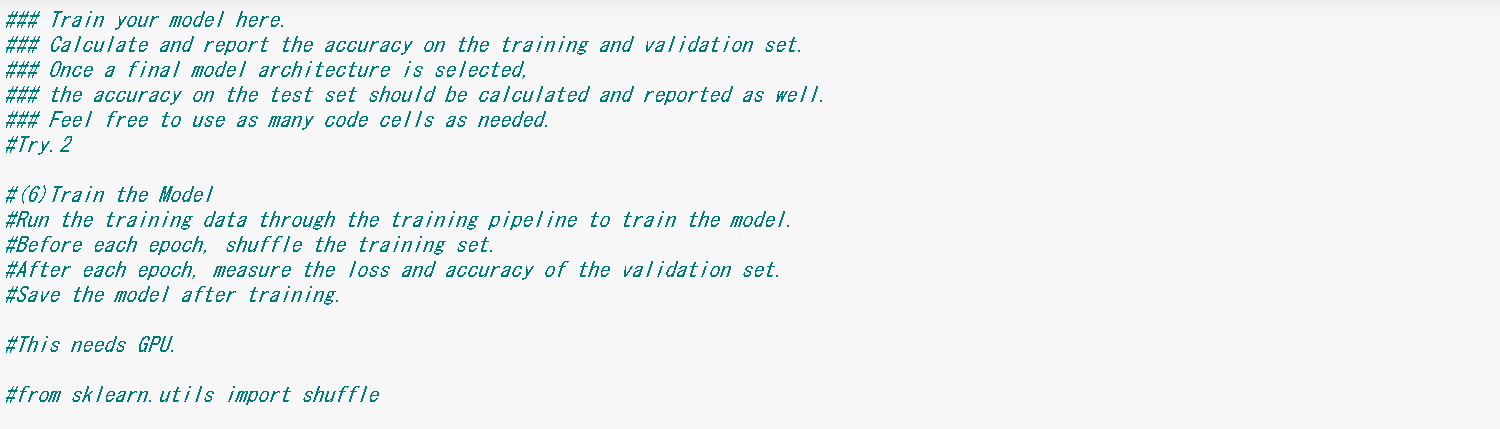


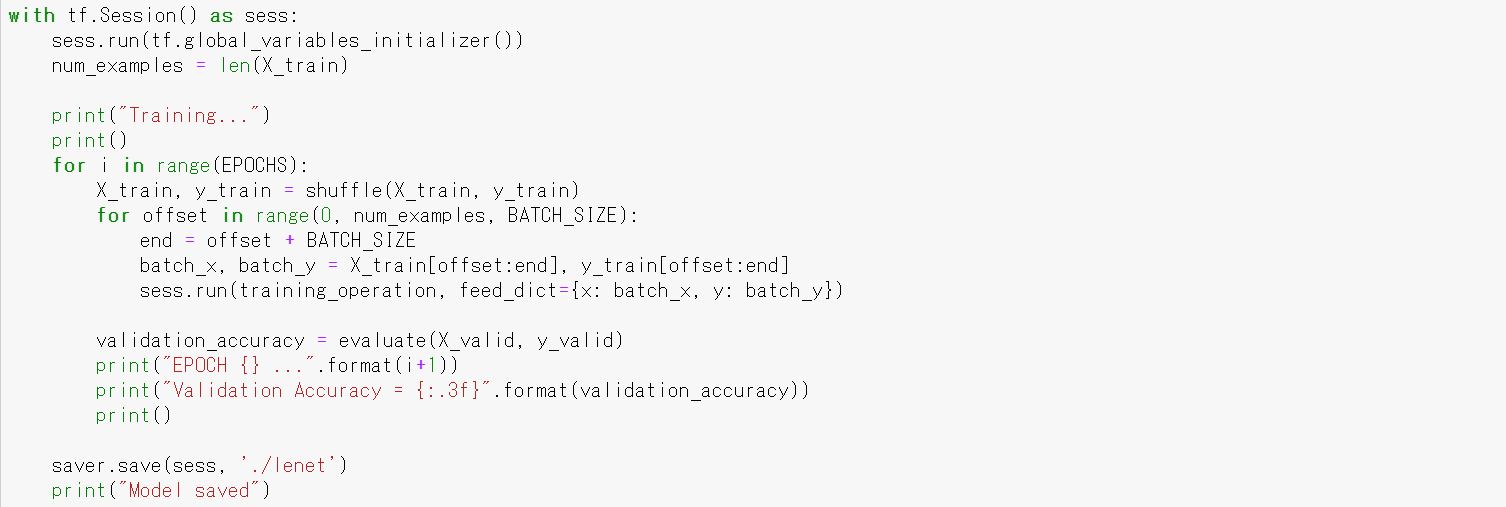




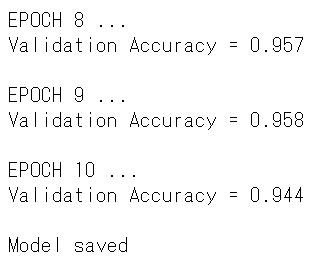
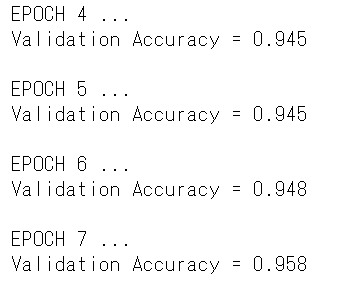
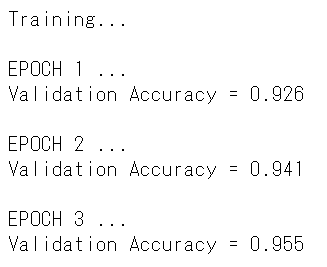
****

****

****

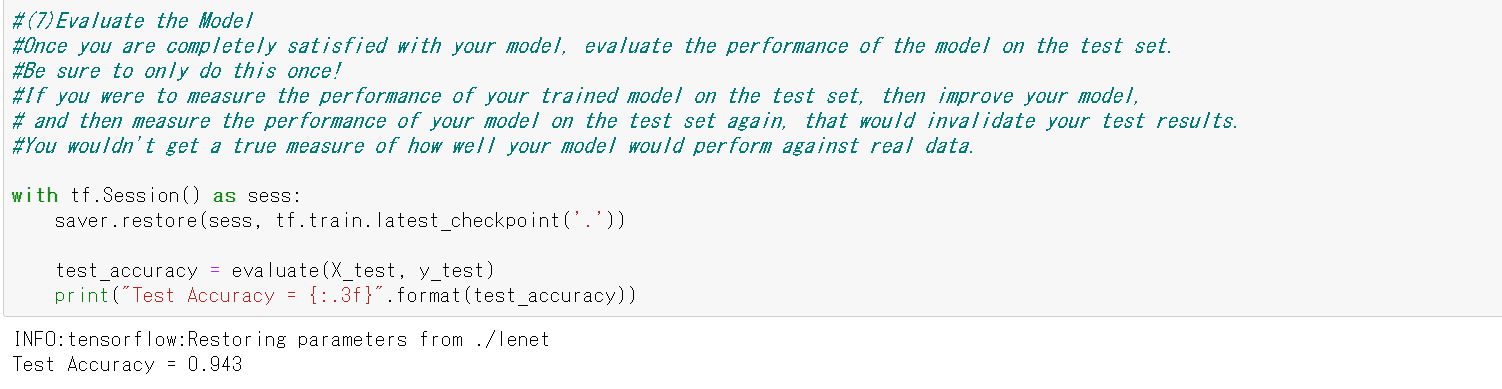
****

The results of Validation Accuracy is shown below. Final Validation Accuracy is **94.4%**, which is better than the Target Accuracy: 93%. It can have better result by Early Termination at **EPOCH 7~9**, but it may have better results with other Test Data, so I proceed by taking this trained model.

****

I evaluated the Model with Test Data.

Test Accuracy is 94.3%, so it passes the Target Accuracy.

****

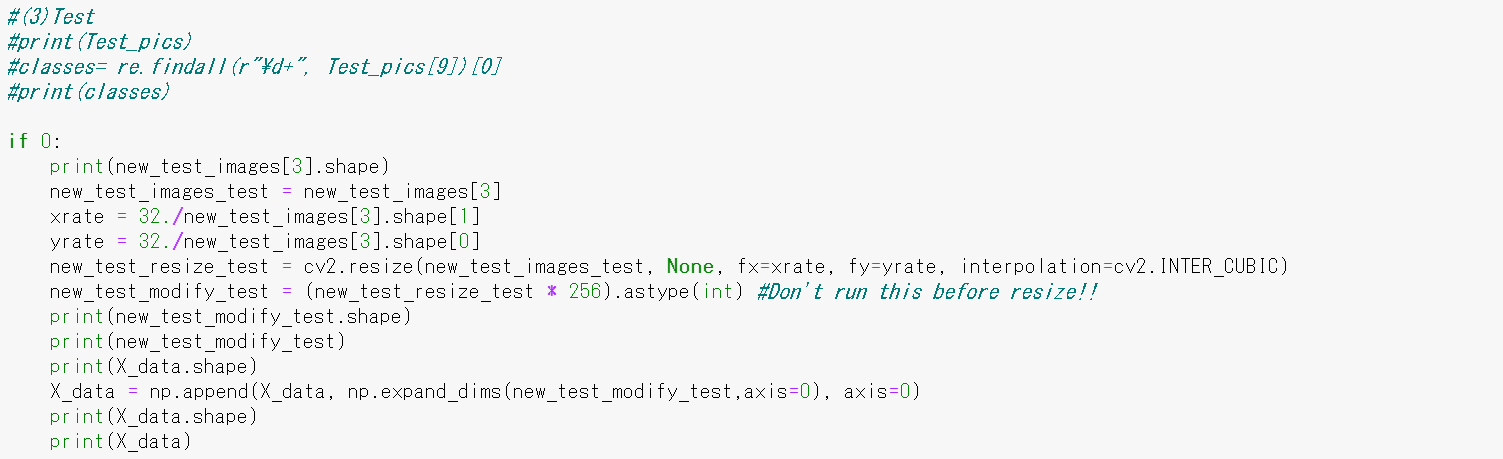
**Step 3: Test a Model on New Images**

I downloaded **10 new Test Images** by the following Website, and evaluated the Model with them.

<https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/version/1>

These images have different image size and data type is float, so I modified them into good data type with the Model. The size of Test Data by new images is **10(image numbers)x32x32(size)x3(RGB colors)**.









I chose 10 images. **Some of them are challenging for classification**.



Leaning

Blurred &

Shaded

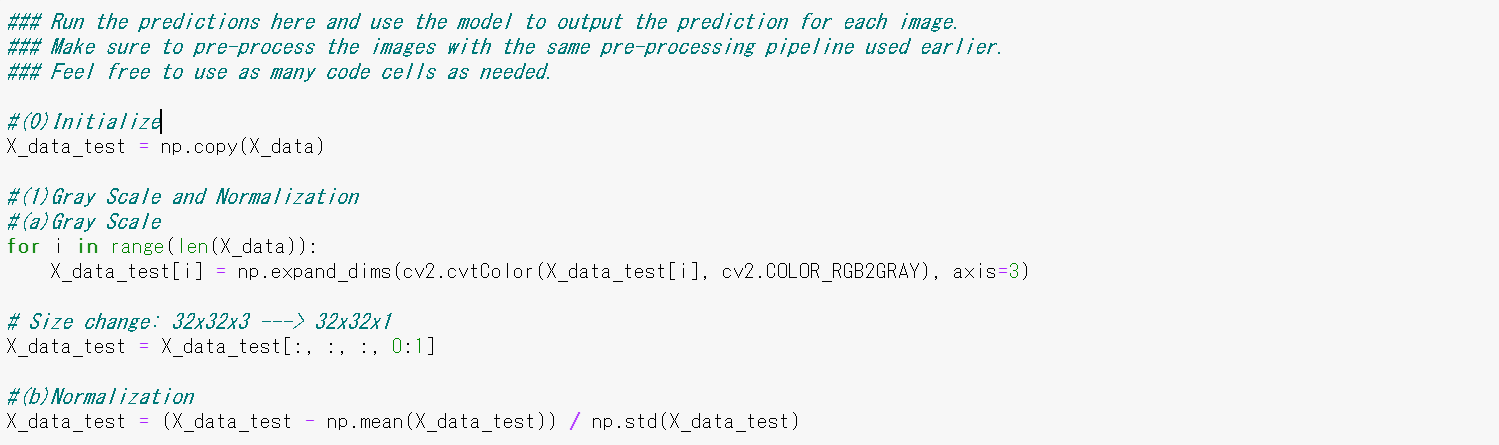
Blurred

Dark

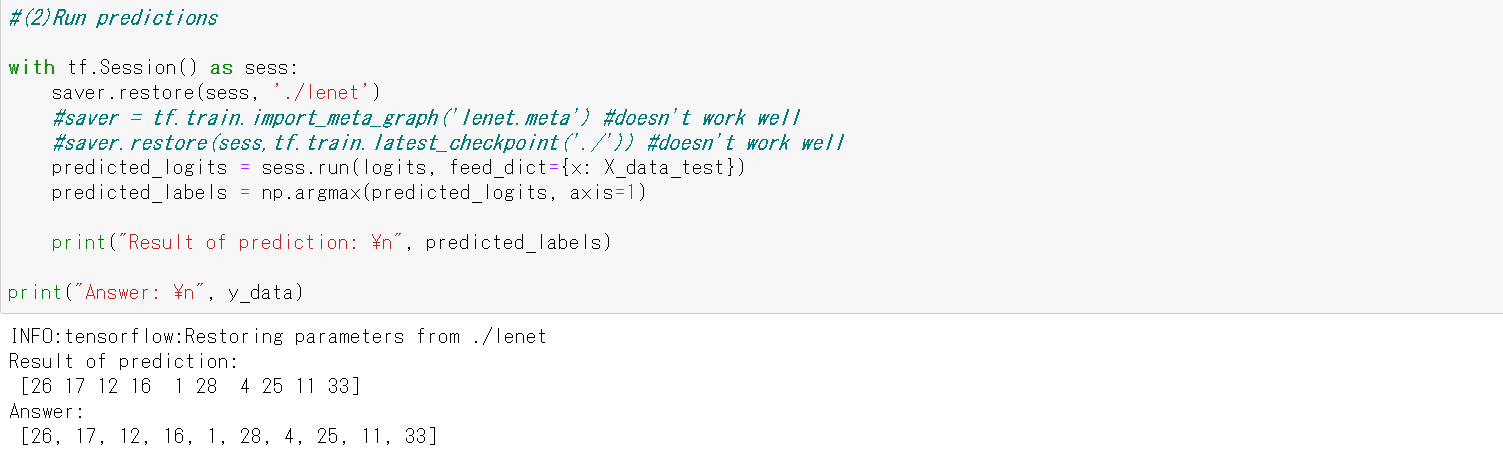
color

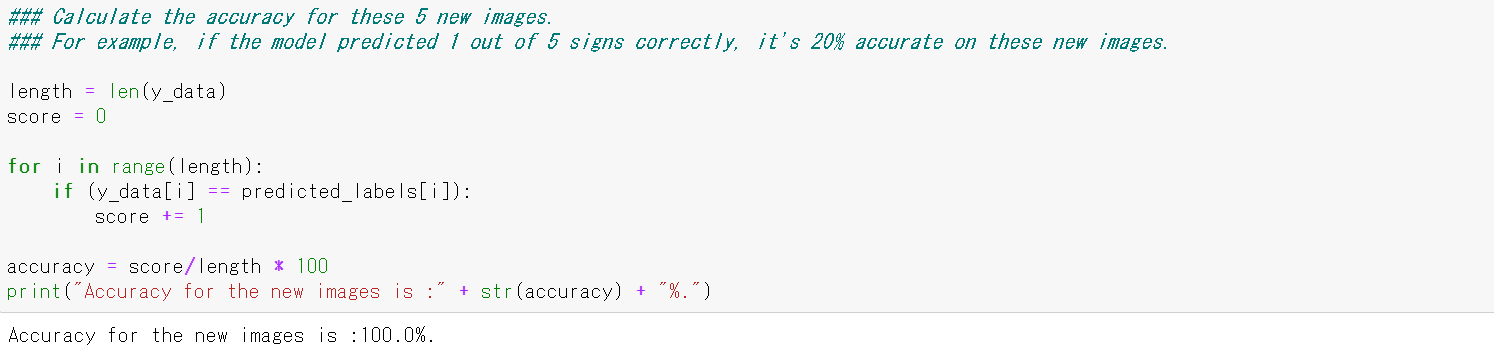
Crushed shape

I adapted Gray scale and Normalization to the new 10 images same as other image data, and I ran the prediction of the Sign Type for each image.

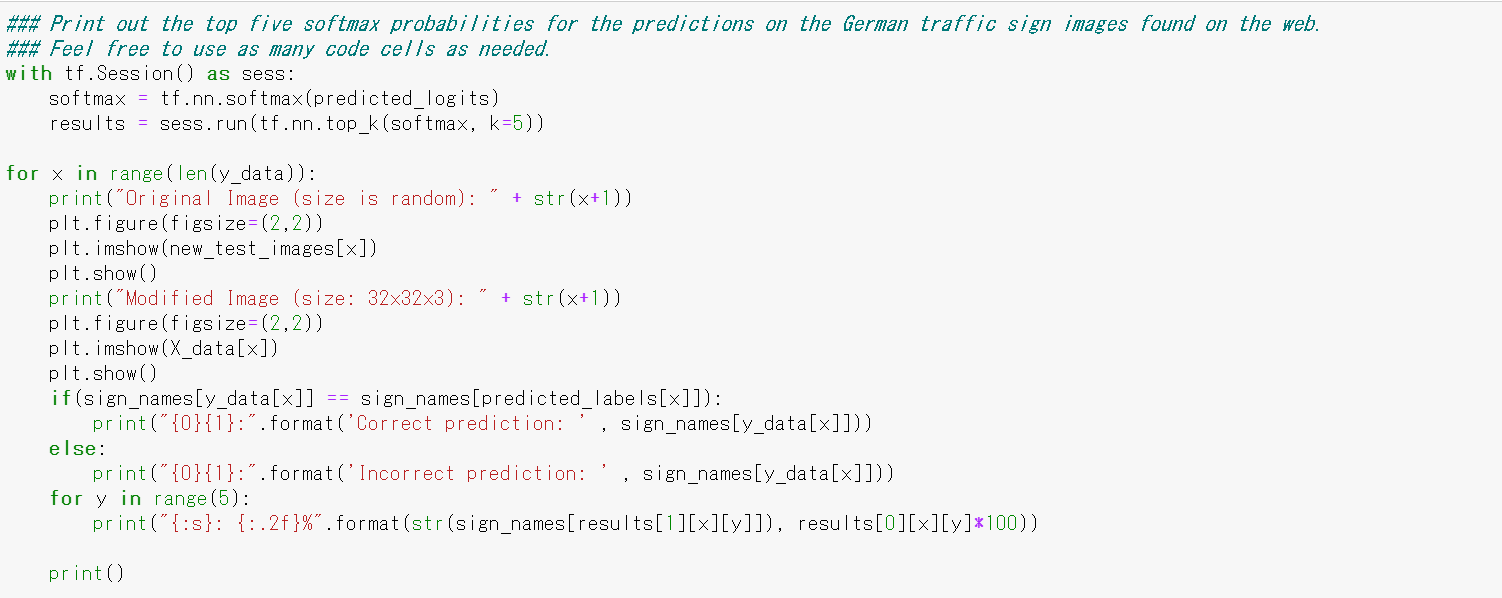


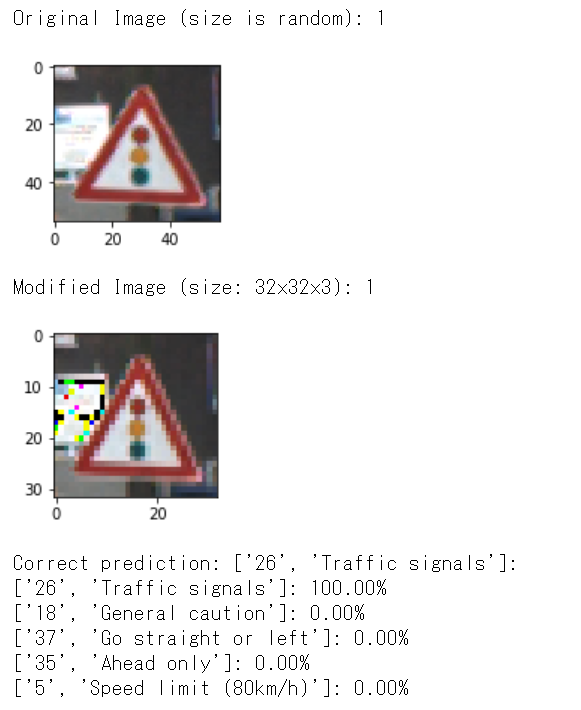
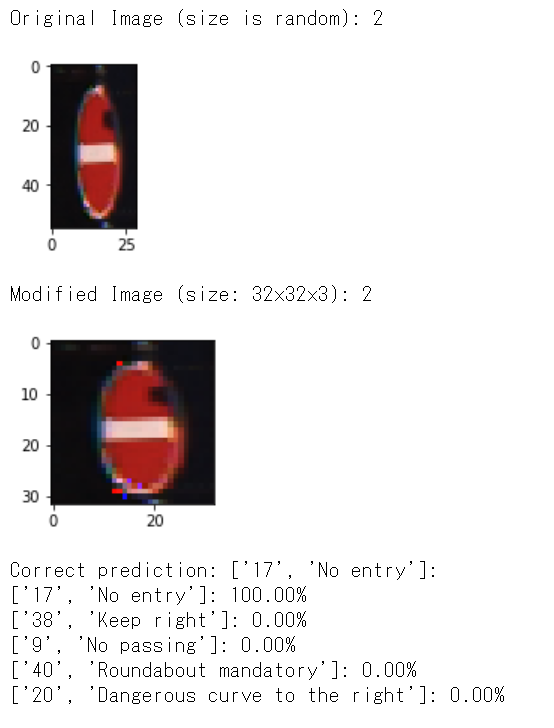
The result of prediction matched the labeled answer **100%**, so my Model is well trained for new images as well.

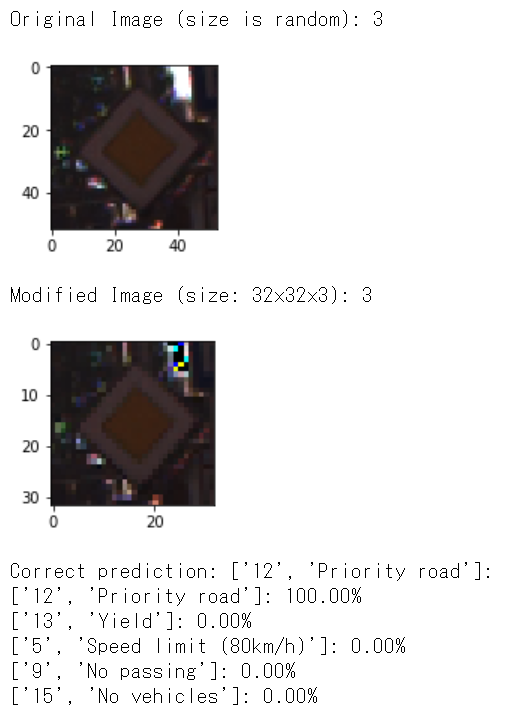
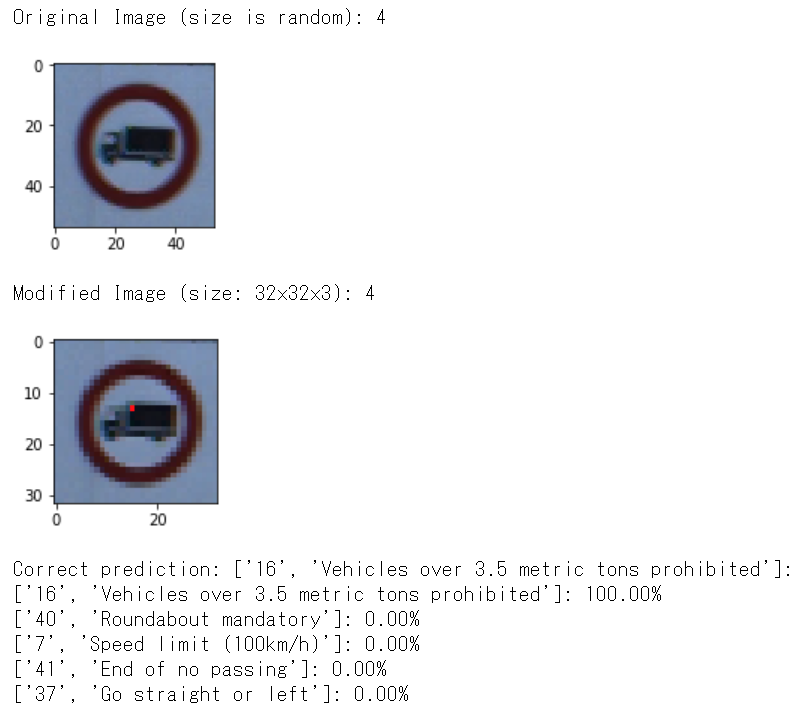


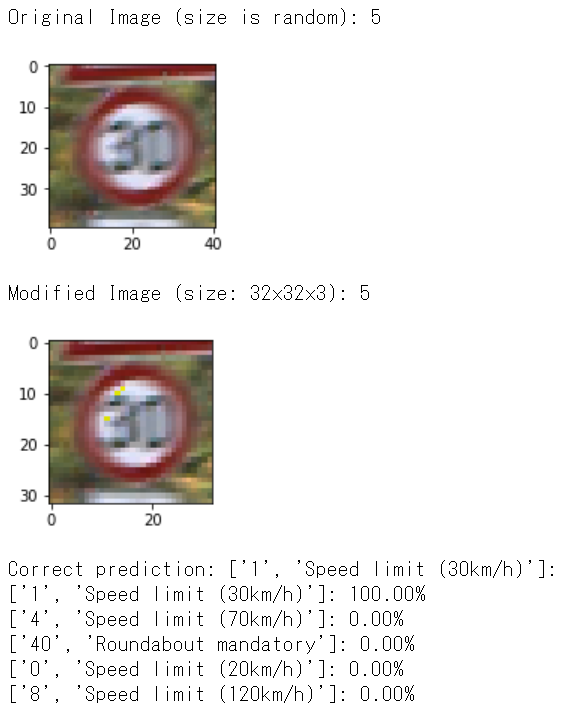
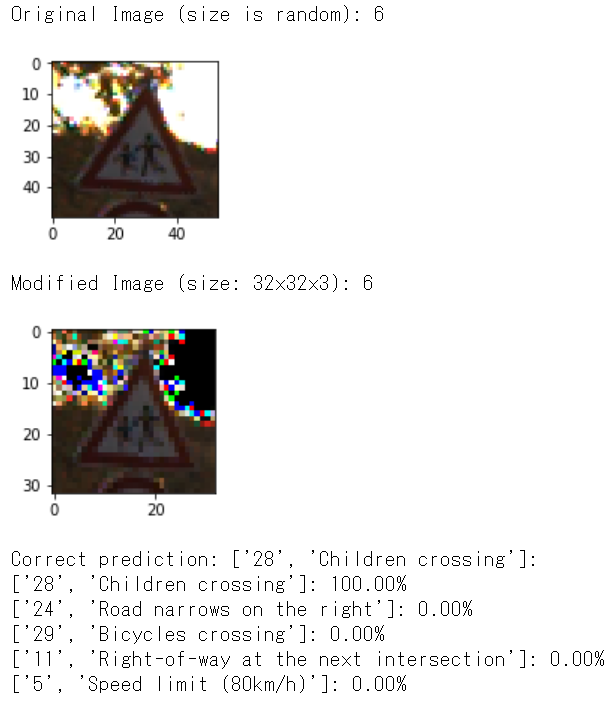


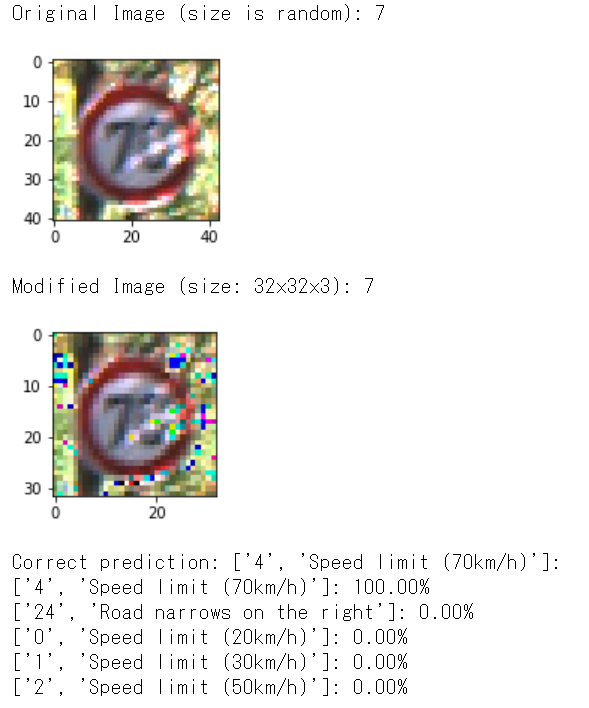
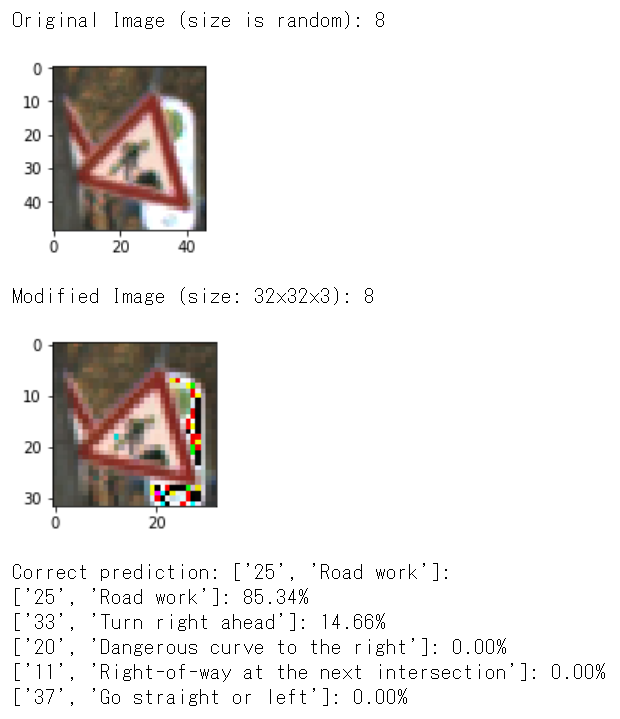
Finally, I show the Output Top 5 Softmax Probabilities for each image found on the Web site.

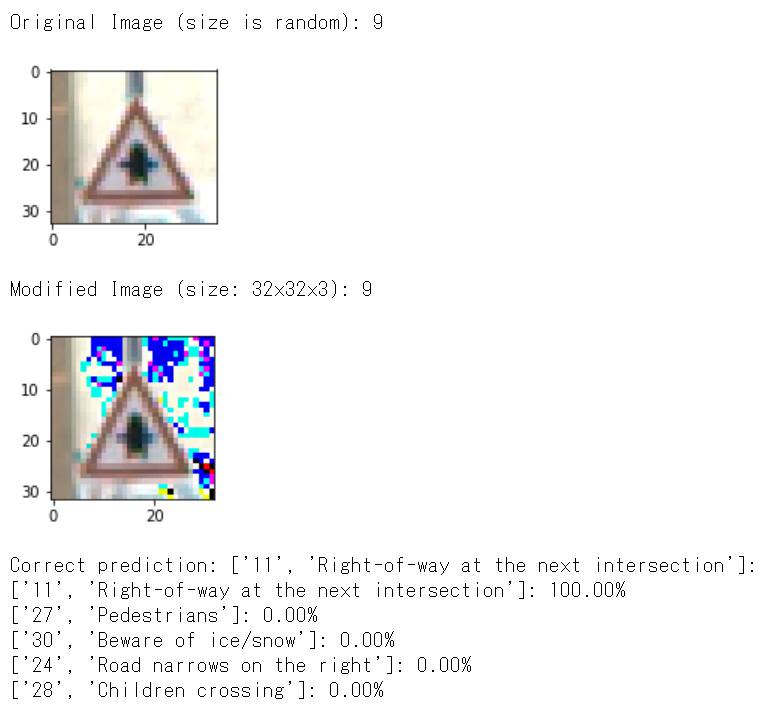
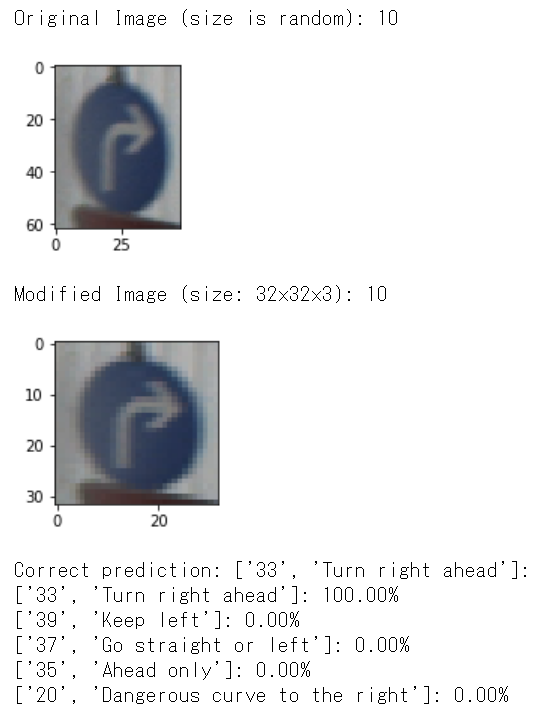
****

** **

** **

** **

**** ****

**** ****

**1st prediction is 85.3% and 2nd prediction is 14.7% with No.8**, but **mostly the 1st prediction is 100%.**