

[Return to Classroom](#)[DISCUSS ON STUDENT HUB](#)

# Traffic Sign Classification

REVIEW

HISTORY

## Meets Specifications

## Excellent job!

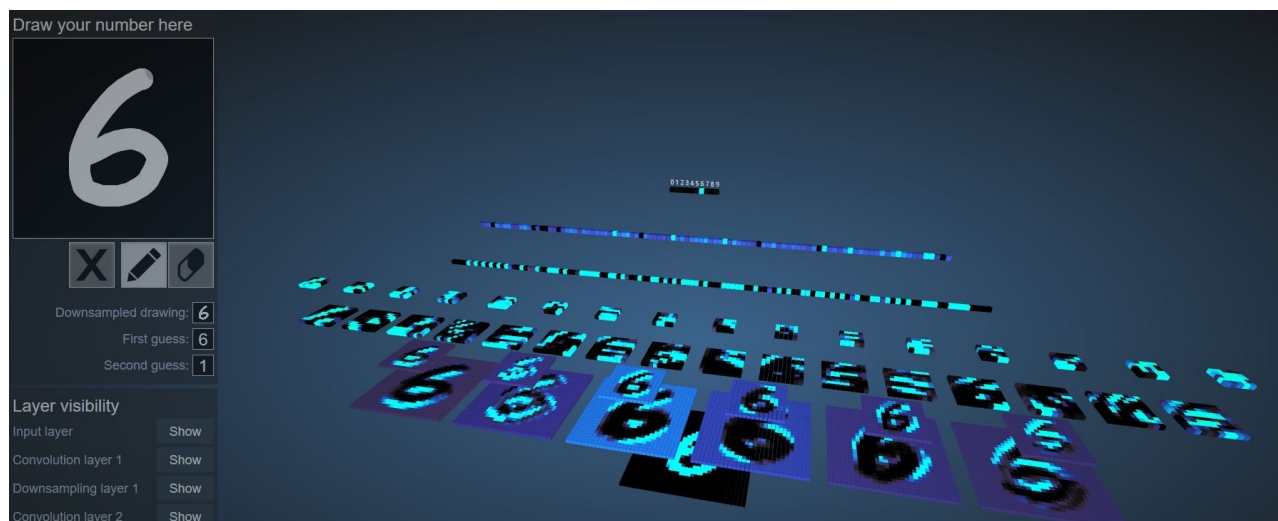
👏 Congratulations! You met all the requirements smoothly! It was a pleasure to go through your code. Please, take into account some comments I added to each part of the rubric below. I'll also suggest using the documentation you created to showcase this work since it is a great project that you achieved here. Perhaps you could even try to blog what you've learned. Keep up the good work! 📄

### Things to highlight from your submission

- This is one of the best reports I have read for this project. Pretty thorough, It felt like reading a paper. In fact, I noticed you used LaTeX!. Also, you got a pretty great accuracy!
- Great work performing perspective transformation to extract traffic signs from google street pictures. In fact, this is sort of what object detection with deep learning does. You can see more [here](#) where an introduction of detecting and classifying an object is explained.
- Finally, I will encourage you to perform the last section in the starter notebook **Visualize the Neural Network's State** where feature maps on your convolutional layers are displayed. In case you don't know, what you are doing in this section is part of the [explainable AI](#) technique, where the goal is to understand AI models with visualizations and other types of illustrations.

#### EXAMPLE

A pretty cool 3D visualization of a CNN classifying numbers trained on the MNIST dataset can be found [here](#). An example for a number 6 is shown below:



## Files Submitted

✓ The project submission includes all required files.

- Ipython notebook with code
- HTML output of the code

Rate this review

START

- A writeup report (either pdf or markdown)

- The `Traffic_Sign_Classifier.ipynb` notebook file with all code cells executed and displaying output. ✓
- An HTML or PDF export of the project notebook with the name `report.html` or `report.pdf`. ✓
- Any **additional datasets or images** used for the project that is not from the German Traffic Sign Dataset ✓
- Your **writeup** report as a markdown or pdf file ✓

Great! Thank you for uploading all required files ☐

## Dataset Exploration

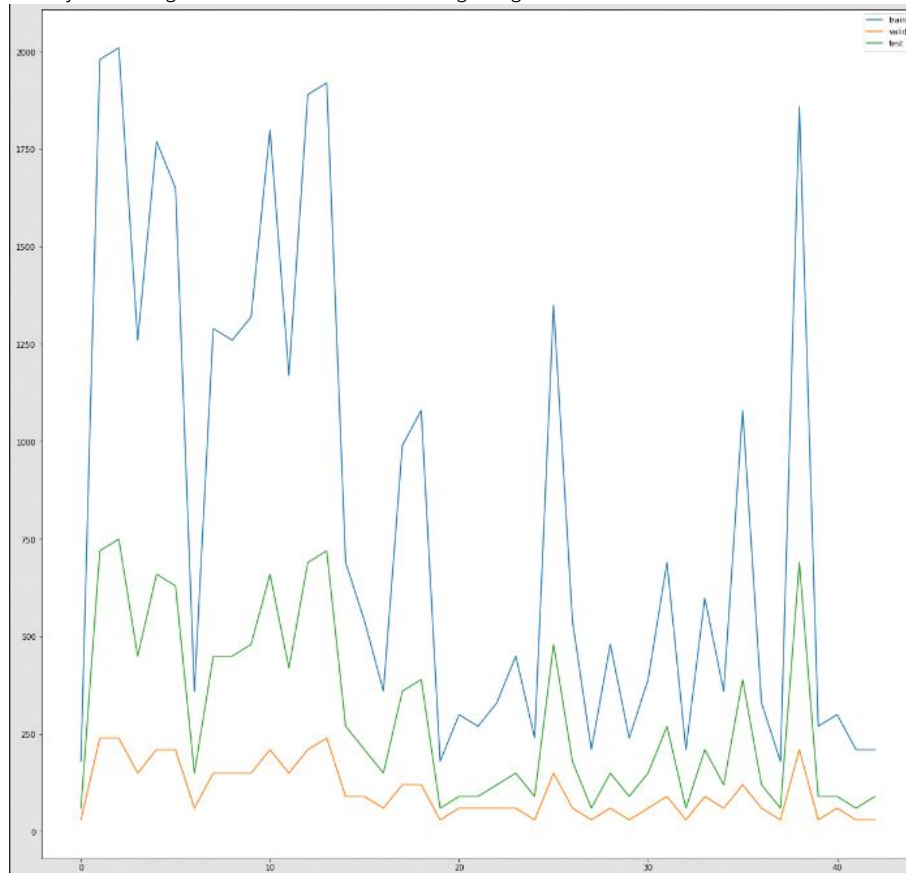
- ✓ The submission includes a basic summary of the data set.

Good job getting the basic summary of the data set as shown below:

- Number of training examples = 34799 ✓
- Number of validation examples = 4410 ✓
- Number of testing examples = 12630 ✓
- Image data shape = (32, 32, 3) ✓
- Number of classes = 43 ✓

- ✓ The submission includes an exploratory visualization on the dataset.

☐ Nice job including the visualization shown below regarding the distribution of the dataset.



**comment:** we usually call this step an Exploratory Data Analysis or EDA. This section usually should take you at least as much time as building the model itself. Exploring the data is crucial because here is where you decide what to refine about the input that will be fed into your model, so that you can make it easier for your model to achieve great performance.

**suggestion:** Always try to draw conclusions based on your analysis. Here, for instance, you could say that the data is uneven. So you can conclude that it will be great to balance the dataset so that the network can avoid skewed learning towards classes that have more data. There are 2 common ways you can mitigate this problem:

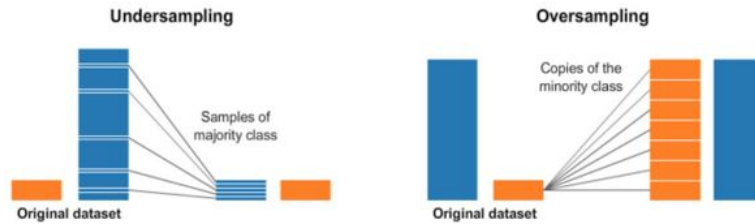
- **Undersampling**, randomly resample some of the observations from the majority class in order to match the numbers with the minority class
  - **Oversampling**, generating synthetic data (data augmentation) that randomly create samples for the minority class
- In terms of images, it usually could be achieved by performing random scaling, random cropping, random

Rate this review

START

mirroring, random rotation, etc.

Below you can see an illustration of the two ideas:



[Reference](#)(Here you can find more details regarding undersampling and oversampling)

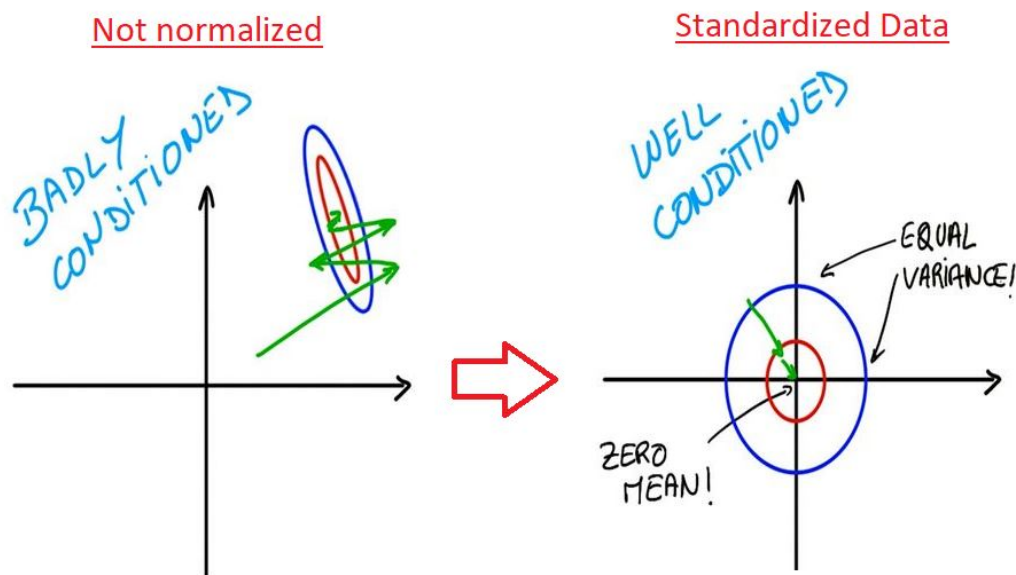
## Design and Test a Model Architecture



The submission describes the preprocessing techniques used and why these techniques were chosen.

- Good job standardizing the dataset. □ This makes convergence faster while training the network
- **Suggestion:** Regarding your Grayscale attempt, have you tried checking the contrast of the images? In fact, when you convert the images to grayscale you might've noticed that some of the images have very poor contrast. That's where Histogram equalization comes in hand, by performing this step you could increase the contrast on low contrast images and consequently increase the potential of your network!

**comment:** To emphasize even more on the importance of normalization or standardization I would like to point out that standardizing your data will deliver a well-conditioned input to your model that will be easier for your network to learn from, as illustrated below:



**suggestion:** I will suggest trying out data augmentation. Let me list all of the augmentation techniques you can implement in this project:

- Geometric augmentations:
  - Rotation ✓
  - Translation ✓
  - Affine transformation ✓
- Cropping
- Scaling
- Shearing
- Shift
- flipping
- sharpening
- Color augmentations
  - Brightness
  - Hue
  - Saturation
  - contrast

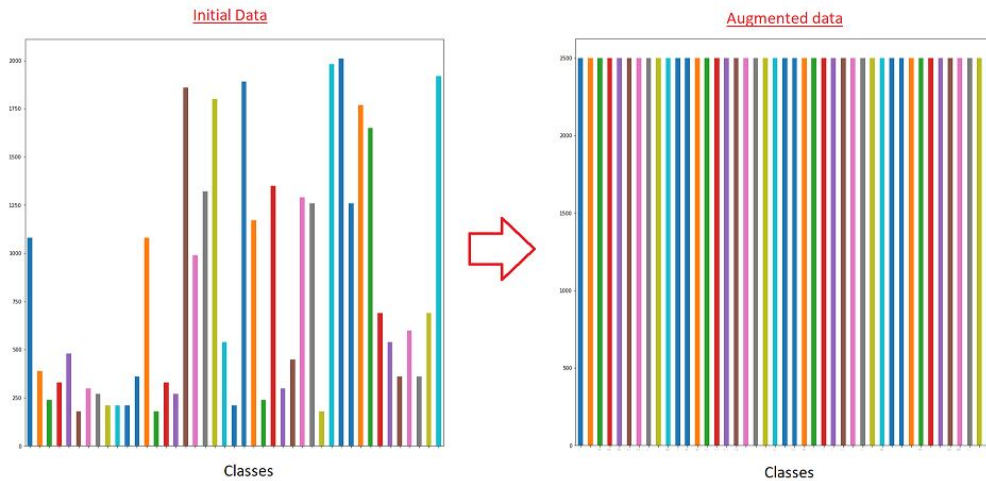
Rate this review

START

- Zooming
- Noise (e.g. Gaussian noise)

You can find some examples about them [here](#) where Tensorflow is used.

**comment:** Data augmentation increases the size of your training data set. Even better, your model will often be more robust (and less prone to overfitting). Also, it is great if you use it to balance an uneven dataset such as the illustration below:



This will help your model avoiding skewed learning towards classes that have more data ☐

### Keep reading in case you want further details on the augmentation approach

**Note:** Data balance is a difficult thing to achieve. In fact, if you augment data to balance an uneven dataset this could not be enough for mitigating the imbalance problem. The reason is that the larger class would have rich variation and the smaller would be many similar images with small affine transformations. That is the reason I suggested you to use as many augmentations as possible to increase variation in the minority classes. However, other alternative approaches, that could work better in case variation on minority classes is not enough, are the following:

- Using `class_weight` argument in your network. The general idea of this is to set higher weights to the classes that have less number of samples. These weights increase the relevance of each class when you are calculating the loss function. i.e. The classes with fewer samples should have higher weights so that the error generated by them has a higher weighing. If you want to find more about this you can look [here](#)
- **Undersampling.** In other words, reducing the size of the majority classes to pair them to the minority class which I explained in the previous section. You can use methods such as bootstrap which is resampling with replacement to achieve that. If you are curious about Bootstrap technique you can find more [here](#) where an example of a bootstrap implementation together with a walkthrough is depicted.
- **Find more data.** In other words, accepting the imbalance and cope with it. To follow this approach you will require more data as deep learning can handle imbalanced data as long as it has enough data to train. That could be a restriction in many cases, so unless you can get more data, try the other approaches.

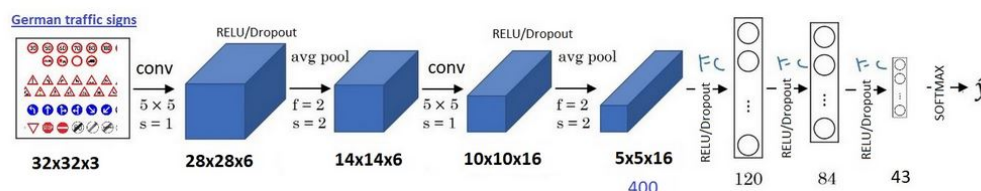


The submission provides details of the characteristics and qualities of the architecture, including the type of model used, the number of layers, and the size of each layer. Visualizations emphasizing particular qualities of the architecture are encouraged.

- Description of the type of architecture used ✓
- Number of layers ✓
- Size of each layer ✓
- (Optional) Sketch of the architecture ✗

Good job describing the layers of the convolutional neural network in your writeup. ☐

**comment:** I will highly suggest you include some visualizations of the architecture you are using. It would be easier for other people to understand your approach or even to showcase your work if you include them. Also having a drawing of what your architecture would be like could help you out coding your model and possibly find some inconsistencies. Take the following diagram as reference:



Rate this review

START



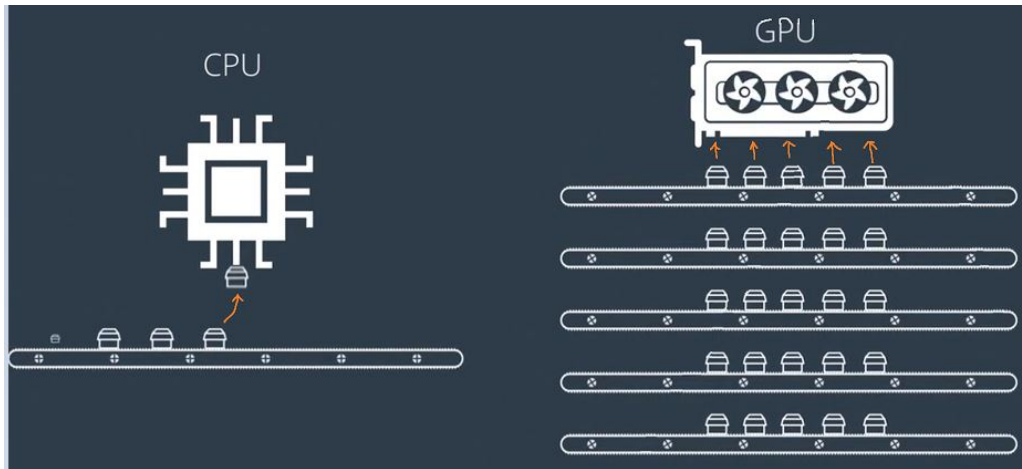
The submission describes how the model was trained by discussing what optimizer was used, batch size, number of epochs and values for hyperparameters.

The discussion involves the following hyper-parameters used for the final model:

- epochs ✓
- batch size ✓
- learning rate ✓
- Optimizer used ✓

Good job mentioning in the writeup all the required details regarding your model. ☐

**comment:** Batch size selection is very important because it will set the pace at which your GPU will get the data as shown below:



Batch size is related to the quantity of data fed to your GPU. In other words, a high batch size will result in a better estimation of the gradient when your optimizer updates the weights. However, it will be at the expense of bursting your GPU memory. That's why whenever you don't have that much of GPU resources you will be forced to reduce the batch size. Try it yourself, increase the batch size to something high and you'll see that TensorFlow throws an OOM error (Out of memory).

- Thus, it will be very beneficial to calculate the **max batch size** to make the most of your GPU power. You can find a great discussion [here](#) on how to calculate it. In a nutshell, it is given by the following relation:

```
Max_batch_size= available_GPU_memory_bytes / 4 / (size_of_tensors + trainable_parameters)
```

- I also noticed you use a 150 batch size which is not a power of 2 number. Usually (depending on the GPU's architecture), we prefer to use a power of 2 batch sizes in case the GPU's physical processor is a power of 2 because of data alignment. However, that all depends on the GPU's architecture, you can check yours and see if it applies, in case it does you should consider modifying it. You can find further discussion on this topic in the following [stackexchange question](#)
- On the other hand, It is also known that small batch sizes could sometimes serve as a regularization technique. Please be aware that if you reduce the batch size then you might require to decrease the learning rate as the steps of gradient descent will be less informed with smaller batch sizes. The downside is that it will take longer for your model to train. In this [blog](#) you can find all the effects that batch size could have in DL models and where this phenomenon is explained.

**comment 2:** I also noticed that you used dropout layers which are a great regularization choice to reduce overfitting and try to help the model generalize better. Therefore, I would also like to introduce you (in case you aren't aware of it yet) to **Batch normalization**. It is also a regularization technique that has gained a lot of popularity in the last few years. Batch normalization is a technique that is typically done by transforming the mini-batch data to zero mean and unit variance before the non-linearity function to try to help improve the gradients of the outputs of the non-linear functions. If you are curious about it you can find more information in [this video](#) where a visual example of a Batch normalization is shown.



The submission describes the approach to finding a solution. Accuracy on the validation set is 0.93 or greater.

- Performance of your model in terms of accuracy:
  - Training: 100% ✓
  - Validation: 97% (required >= 93)✓
  - Testing: 95.2% ✓
- Description of how you get to this result ✓

Values above 95% are very impressive. Great Job!! ☐






SUGGESTION

One thing I will suggest is to create a confusion matrix instead of solely evaluating your network based on accuracy. The reason is that high accuracies can often be misleading especially when dealing with imbalanced data such as the one in

Rate this review

START

this project. An example of a short, but meaningful confusion matrix for this project is shown below:

	0	1	2	3	4	5	6	7	8	9	10	11	12
Ground Truth Class ⇌ Prediction Class ⇓													
0	48	-1	0	0	0	0	0	0	0	0	0	0	0
1	-4	683	-3	0	-3	0	-1	0	0	0	0	0	0
2	0	-15	733	-1	-2	-38	0	-2	-2	0	0	0	0
3	0	0	-1	409	-1	-9	-7	0	-5	0	0	0	0
4	-8	-9	-2	0	616	-5	0	0	0	0	0	0	-1
5	0	-1	-1	-34	-5	552	-1	-8	-5	0	-32	0	0
6	0	0	0	0	0	0	125	0	0	0	0	0	0
7	0	-4	-4	-2	-10	-16	-1	427	-6	0	0	0	0
8	0	-3	-6	0	-8	-9	-7	-12	431	-2	-1	0	0
9	0	0	0	-2	0	-1	-1	0	0	478	-1	0	0
10	0	0	0	-1	0	0	0	0	0	0	615	0	0
11	0	0	0	0	0	0	0	0	0	0	0	391	0
12	0	0	0	-1	-2	0	0	0	0	0	0	-2	683
13	0	0	0	0	-1	0	0	0	0	0	-1	0	0
14	0	0	0	0	-6	0	0	0	0	0	0	0	0
15	0	-2	0	0	-4	0	0	-1	0	0	-2	0	0

## Test a Model on New Images



The submission includes five new German Traffic signs found on the web, and the images are visualized. Discussion is made as to particular qualities of the images or traffic signs in the images that are of interest, such as whether they would be difficult for the model to classify.

- $\geq 5$  traffic sign images taken from the internet are assessed with the model developed. ✓
- Discussion of any particular qualities of the traffic signs chosen that may be of interest ✓



The submission documents the performance of the model when tested on the captured images. The performance on the new images is compared to the accuracy results of the test set.

- Performance of the model, when tested on the  $\geq 5$  images, is displayed. ✓
- The performance of the new images is compared to the accuracy results of the test set. ✓

Great discussion and comparison of your model assessed with the 5 images and with the test set in terms of performance. □

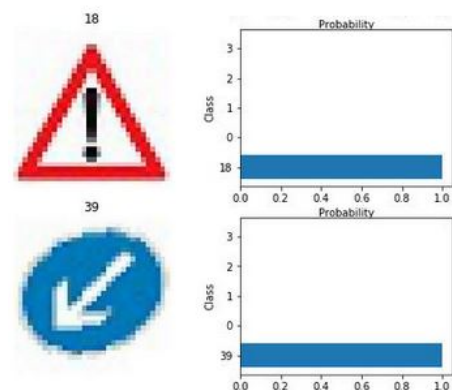


The top five softmax probabilities of the predictions on the captured images are outputted. The submission discusses how certain or uncertain the model is of its predictions.

- Top 5 softmax probabilities are displayed ✓
- Certainty or uncertainty of the model based on the predictions is discussed ✓

Really good work on how you display the top 5 softmax probabilities □

As future work, you can create some plots that show the top\_k probabilities as follows:



Rate this review

START

RETURN TO PATH

Rate this review

START