Road Sign Predictions

Course CSE 450 Brad Strange, Jacob Ferris, Landon Davis, James Hall, Keaton Kesler

Model

I. Model Introduction and evaluation

We created a convolutional neural network that is capable of distinguishing between different types of traffic signs. Our model can process and predict over 1,000 images per second with 99.5% accuracy. We decided that accuracy would be the best metric to assess our model's performance and reliability because it focuses on how often the model is correct overall. We got more specific and analyzed how accurate the model was with each specific type of sign, such as stop signs, but there wasn't much of a discrepancy between that and the overall accuracy, which is nearing 100%.

II. Model Chosen

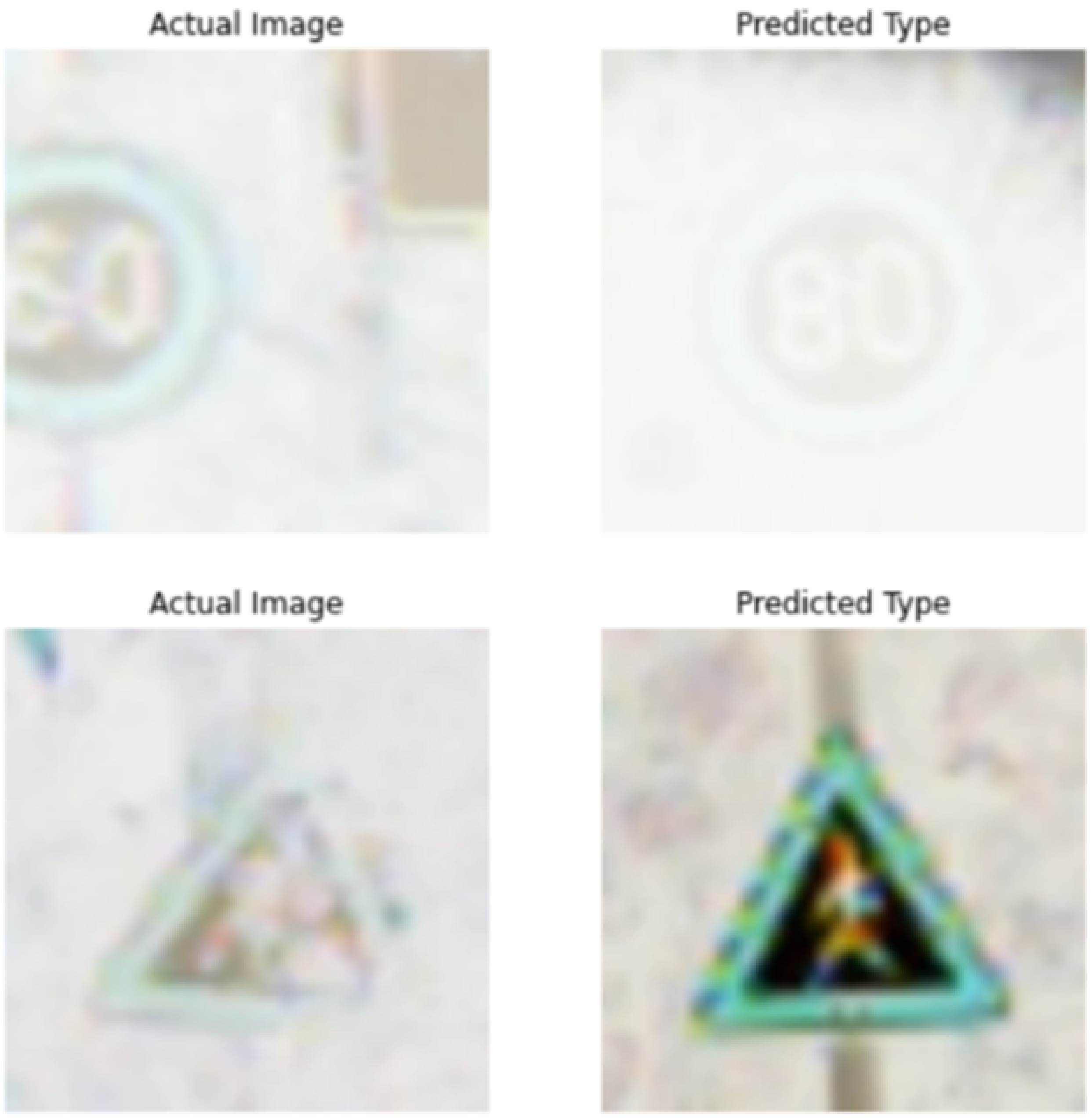
We decided to go with a pre-built network because it had the best performance in terms of accuracy. We tried different types of pre-built networks ranging from Xception, VGG16, and EfficentNetB7. We tried our luck with creating the models by hand but found better success in pre-built networks (see Figures 4-6 to see the differences between all the models we created). We also tried to see how a non-convolutional neural network would perform. Surprisingly it performed just as good as our hand created model without batch normalization. Ultimately, we went with InceptionV3 as it performed the best and had a somewhat easy implementation.

III. Model Testing

We needed to test the model's accuracy with new data that it hadn't seen before while being trained. To do this, we used the first 200 images of the holdout data set, with their correct labels, for testing our model after we finished training it. The model correctly identified those signs with a 99.5% accuracy, only missing one image (see Figure 2 below). Although we felt confident in our model, we wanted to take an extra measure to ensure it would be reliable, and decided to go out and take our own images to see if the model could correctly identify them. We specifically decided to test the stop sign, as it is the only sign we found here in the U.S. that was comparable to the German signs the model was trained to recognize, and it is arguably the most important sign for the model to correctly identify. Our model predicted each stop sign with 100% accuracy (see Figure 3 below).

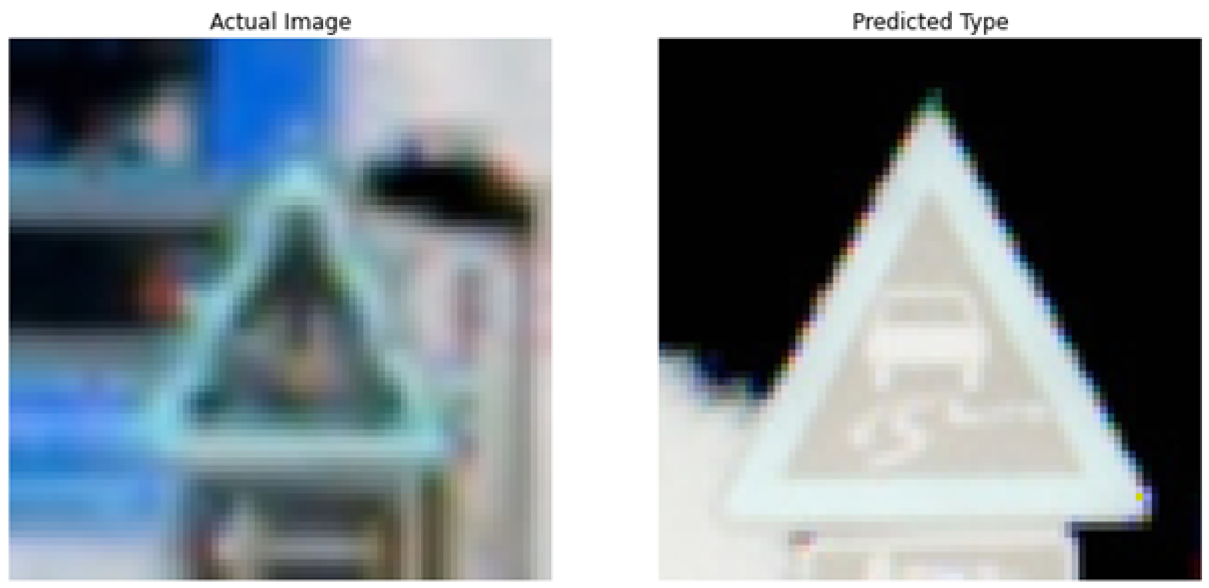
Visualizations

Figure 1. Images misidentified in testing data during model adjusting.



In our figure above, we see an example of how our model was working and what it was looking for in recognizing signs. This was early in our model process and these were examples of pictures our model guessed incorrectly. The first one we recognized that since the picture was in the corner it mistook the 6 for an 8 on the speed limit sign. We were able to quickly address this and believe that the next instant the car would see the sign more centered and predict it well anyway. For the other you can see all the noise of white spots and can see why it may have been predicted wrong but we still see the major points hit and again we were able to address this issue later in our model.

Figure 2. Picture of the only sign our model mislabeled in our validation.



The image on the left is the only image our model predicted incorrectly in both our validation and testing data sets after we finished training our final model. It seems unlikely that a human could recognize the sign either, and none of us could. When applied in real life settings, the pictures taken should not be this blurred and we feel confident in our model predicting signs correctly in most circumstances, especially considering that multiple pictures can be taken of each sign in quick succession.

Figure 3. One of many pictures taken on campus of stop signs. Used to test model.

Figures 4-6 Note: “cnn” refers to the model that we created by hand that has batch normalization, “nbn” refers to the model that we created by hand that has no batch normalization, “inc” refers to the model that we chose in the end (Inception).

Figure 4. Time taken to train each CNN.

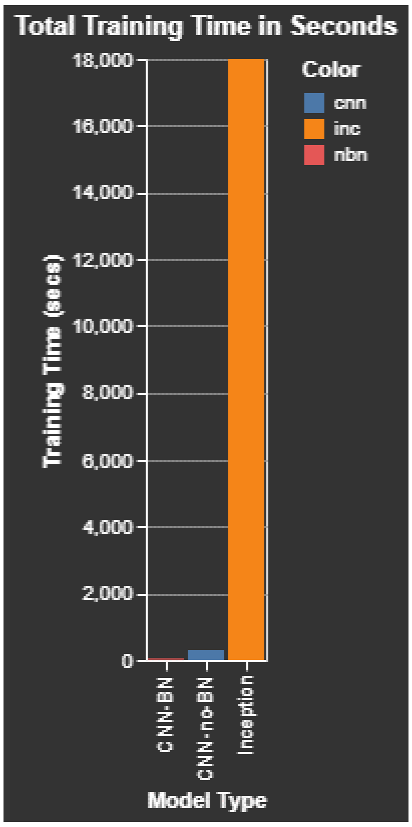


Figure 5. Time taken per image for each model.

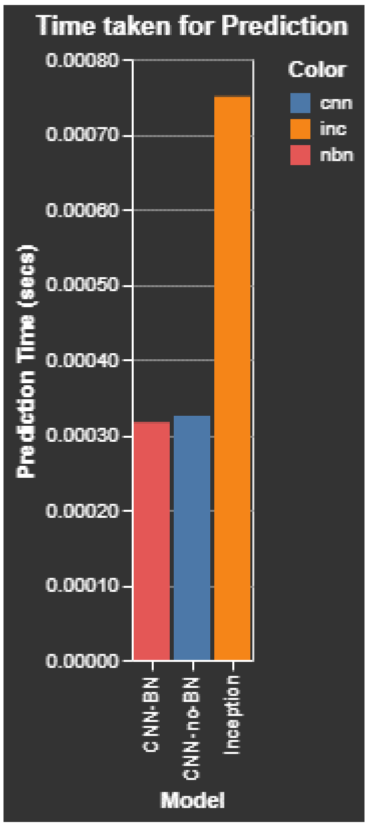
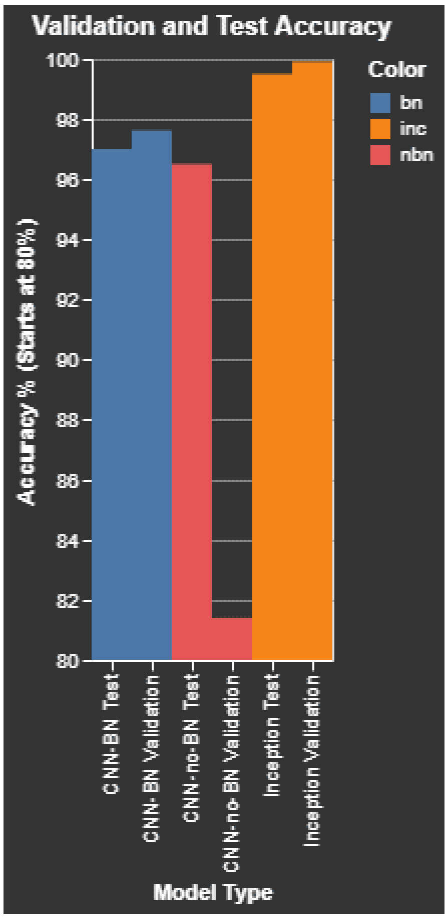


Figure 6. Validation and test accuracy for each model.



Discussion Responses

Below are our final answers to the case discussion questions:

Will you be using an existing architecture as a starting point, or do you think it'll be better to design your own?

- **We decided to go with existing architecture, but using new weights, as it gave us the best performance.**

Do you think we need to do any preprocessing before using the data to train the model?

- **We rescaled all the images to 75 by 75 pixels and decided to use data augmentation to train on rotations and other transformations of the data.**

What strategy would you suggest to recognize signs that don't look like our data?

- **We did some preprocessing data augmentation before running our models to help us line up our photos to make sure they are in the same style and can help our model recognize the same patterns easier.**

Straight accuracy might not be the best metric for model evaluation, but what do you think?

- **We do think accuracy is a good model for this because we want our car to predict all signs correctly, not just certain styles and getting a sign wrong either way can be punishing depending on what sign it is. We have an F1 measurement as well but we feel accuracy is important and will be used.**

What performance would you require before you'd ride in the car?

- **We have a test accuracy of 99.5% right now. We feel that accuracy is good enough for us to feel safe in the car given that our accuracy is 100% on important signs such as stop and yield signs.**

APPENDIX A (PYTHON SCRIPT)

Below are Google Colab links to the notebooks we used during this case study:

- https://colab.research.google.com/drive/1D8iy2Xo5gp6Uw8EK4gqj32WIWJy8APvj
- https://colab.research.google.com/gist/jamesbhall423/146f80706137ff99c938a5677663ffa4/starter_signs_v2_student.ipynb
- https://colab.research.google.com/drive/1CK7APTQe_l8wIjy1txuxYthKVdqnY2P1?usp=sharing