CS50's Introduction to Artificial Intelligence with Python

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

Brian Yu (https://brianyu.me)
brian@cs.harvard.edu

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/) (https://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Traffic

The latest version of Python you should use in this course is Python 3.12. **This is particularly the case for this project, due to interactions with TensorFlow**.

Write an AI to identify which traffic sign appears in a photograph.

```
$ python traffic.py gtsrb
Epoch 1/10
500/500 [==============================] - 5s 9ms/step - loss: 3.7139 - accuracy: 0.1545
Epoch 2/10
500/500 [==============================] - 6s 11ms/step - loss: 2.0086 - accuracy: 0.4082
Epoch 3/10
500/500 [==============================] - 6s 12ms/step - loss: 1.3055 - accuracy: 0.5917
Epoch 4/10
500/500 [==============================] - 5s 11ms/step - loss: 0.9181 - accuracy: 0.7171
Epoch 5/10
500/500 [==============================] - 7s 13ms/step - loss: 0.6560 - accuracy: 0.7974
Epoch 6/10
500/500 [==============================] - 9s 18ms/step - loss: 0.5078 - accuracy: 0.8470
Epoch 7/10
500/500 [==============================] - 9s 18ms/step - loss: 0.4216 - accuracy: 0.8754
Epoch 8/10
500/500 [==============================] - 10s 20ms/step - loss: 0.3526 - accuracy: 0.8946
Epoch 9/10
500/500 [==============================] - 10s 21ms/step - loss: 0.3016 - accuracy: 0.9086
Epoch 10/10
500/500 [==============================] - 10s 20ms/step - loss: 0.2497 - accuracy: 0.9256
333/333 - 5s - loss: 0.1616 - accuracy: 0.9535
```

When to Do It

By Wednesday, January 1, 2025 at 1:59 AM GMT-3 (https://time.cs50.io/20241231T235900-0500)

How to Get Help

1. Ask questions via Ed (https://cs50.edx.org/ed)!
2. Ask questions via any of CS50's communities!

Background

As research continues in the development of self-driving cars, one of the key challenges is computer vision (https://en.wikipedia.org/wiki/Computer_vision), allowing these cars to develop an understanding of their environment from digital images. In particular, this involves the ability to recognize and distinguish road signs – stop signs, speed limit signs, yield signs, and more.

In this project, you'll use TensorFlow (https://www.tensorflow.org/) to build a neural network to classify road signs based on an image of those signs. To do so, you'll need a labeled dataset: a collection of images that have already been categorized by the road sign represented in them.

Several such data sets exist, but for this project, we'll use the German Traffic Sign Recognition Benchmark (http://benchmark.ini.rub.de/?section=gtsrb&subsection=news) (GTSRB) dataset, which contains thousands of images of 43 different kinds of road signs.

Getting Started

- Download the distribution code from https://cdn.cs50.net/ai/2023/x/projects/5/traffic.zip (https://cdn.cs50.net/ai/2023/x/projects/5/traffic.zip) and unzip it.
- Download the data set (https://cdn.cs50.net/ai/2023/x/projects/5/gtsrb.zip) for this project and unzip it. Move the resulting `gtsrb` directory inside of your `traffic` directory.
- Inside of the `traffic` directory, run `pip3 install -r requirements.txt` to install this project's dependencies: `opencv-python` for image processing, `scikit-learn` for ML-related functions, and `tensorflow` for neural networks.

Understanding

First, take a look at the data set by opening the `gtsrb` directory. You'll notice 43 subdirectories in this dataset, numbered `0` through `42`. Each numbered subdirectory represents a different category (a different type of road sign). Within each traffic sign's directory is a collection of images of that type of traffic sign.

Next, take a look at `traffic.py`. In the `main` function, we accept as command-line arguments a directory containing the data and (optionally) a filename to which to save the trained model. The data and corresponding labels are then loaded from the data directory (via the `load_data` function) and split into training and testing sets. After that, the `get_model` function is called to obtain a compiled neural network that is then fitted on the training data. The model is then evaluated on the testing data. Finally, if a model filename was provided, the trained model is saved to disk.

The `load_data` and `get_model` functions are left to you to implement.

Specification

Complete the implementation of `load_data` and `get_model` in `traffic.py`.

- The `load_data` function should accept as an argument `data_dir`, representing the path to a directory where the data is stored, and return image arrays and labels for each image in the data set.
  - You may assume that `data_dir` will contain one directory named after each category, numbered `0` through `NUM_CATEGORIES - 1`. Inside each category directory will be some number of image files.
  - Use the OpenCV-Python module (`cv2`) to read each image as a `numpy.ndarray` (a `numpy` multidimensional array). To pass these images into a neural network, the images will need to be the same size, so be sure to resize each image to have width `IMG_WIDTH` and height `IMG_HEIGHT`.
  - The function should return a tuple `(images, labels)`. `images` should be a list of all of the images in the data set, where each image is represented as a `numpy.ndarray` of the appropriate size. `labels` should be a list of integers, representing the category number for each of the corresponding images in the `images` list.
  - Your function should be platform-independent: that is to say, it should work regardless of operating system. Note that on macOS, the `/` character is used to separate path components, while the `\` character is used on Windows. Use `os.sep` (https://docs.python.org/3/library/os.html) and `os.path.join` (https://docs.python.org/3/library/os.path.html#os.path.join) as needed instead of using your platform's specific separator character.
- The `get_model` function should return a compiled neural network model.
  - You may assume that the input to the neural network will be of the shape `(IMG_WIDTH, IMG_HEIGHT, 3)` (that is, an array representing an image of width `IMG_WIDTH`, height `IMG_HEIGHT`, and `3` values for each pixel for red, green, and blue).
  - The output layer of the neural network should have `NUM_CATEGORIES` units, one for each of the traffic sign categories.
  - The number of layers and the types of layers you include in between are up to you. You may wish to experiment with:
    - different numbers of convolutional and pooling layers
    - different numbers and sizes of filters for convolutional layers
    - different pool sizes for pooling layers
    - different numbers and sizes of hidden layers
    - dropout
- In a separate file called *README.md*, document (in at least a paragraph or two) your experimentation process. What did you try? What worked well? What didn't work well? What did you notice?

Ultimately, much of this project is about exploring documentation and investigating different options in `cv2` and `tensorflow` and seeing what results you get when you try them!

You should not modify anything else in `traffic.py` other than the functions the specification calls for you to implement, though you may write additional functions and/or import other Python standard library modules. You may also import `numpy` or `pandas`, if familiar with them, but you should not use any other third-party Python modules. You may modify the global variables defined at the top of the file to test your program with other values.

Hints

- Check out the official Tensorflow Keras overview (https://www.tensorflow.org/guide/keras/overview) for some guidelines for the syntax of building neural network layers. You may find the lecture source code useful as well.
- The OpenCV-Python (https://docs.opencv.org/4.5.2/d2/d96/tutorial_py_table_of_contents_imgproc.html) documentation may prove helpful for reading images as arrays and then resizing them.
- Once you've resized an image `img`, you can verify its dimensions by printing the value of `img.shape`. If you've resized the image correctly, its shape should be `(30, 30, 3)` (assuming `IMG_WIDTH` and `IMG_HEIGHT` are both `30`).
- If you'd like to practice with a smaller data set, you can download a modified dataset (https://cdn.cs50.net/ai/2023/x/projects/5/gtsrb-small.zip) that contains only 3 different types of road signs instead of 43.

Testing

If you'd like, you can execute the below (after setting up `check50` (https://cs50.readthedocs.io/projects/check50/en/latest/index.html) on your system) to evaluate the correctness of your code. This isn't obligatory; you can simply submit following the steps at the end of this specification, and these same tests will run on our server. Either way, be sure to compile and test it yourself as well!

```
check50 ai50/projects/2024/x/traffic
```

Execute the below to evaluate the style of your code using `style50`.

```
style50 traffic.py
```

Remember that **you may not import any modules** (other than those in the Python standard library) **other than those explicitly authorized herein**. Doing so will not only prevent `check50` from running, but will also prevent `submit50` from scoring your assignment, since it uses `check50`. If that happens, you've likely imported something disallowed or otherwise modified the distribution code in an unauthorized manner, per the specification. There are certainly tools out there that trivialize some of these projects, but that's not the goal here; you're learning things at a lower level. If we don't say here that you can use them, you can't use them.

How to Submit

Beginning **Monday, January 1, 2024 at 2:00 AM GMT-3 (https://time.cs50.io/20240101T000000-0500)**, the course has transitioned to a new submission platform. If you had not completed CS50 AI prior to that time, **you must join the new course pursuant to Step 1, below**, and also must resubmit all of your past projects using the new submission slugs to import their scores. We apologize for the inconvenience, but hope you feel that access to `check50`, which is new for 2024, is a worthwhile trade-off for it, here!

1. Visit this link (https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b), log in with your GitHub account, and click **Authorize cs50**. Then, check the box indicating that you'd like to grant course staff access to your submissions, and click **Join course**.
2. Install Git (https://git-scm.com/downloads) and, optionally, install `submit50` (https://cs50.readthedocs.io/submit50/).
3. If you've installed `submit50`, execute

   ```
   submit50 ai50/projects/2024/x/traffic
   ```

   Otherwise, using Git, push your work to `https://github.com/me50/USERNAME.git`, where `USERNAME` is your GitHub username, on a branch called `ai50/projects/2024/x/traffic`.

If you submit your code directly using Git, rather than `submit50`, **do not** include the `gtsrb` directory as part of your submission. (It's too large, and the autograder will almost certainly time-out trying to deal with it.) `submit50` will automatically exclude this for you.

Work should be graded within five minutes. You can then go to https://cs50.me/cs50ai (https://cs50.me/cs50ai) to view your current progress!

Acknowledgements

Data provided by J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011 (http://benchmark.ini.rub.de/index.php?section=gtsrb&subsection=dataset#Acknowledgements)