**Training a CNN in Edge Impulse**

**Introduction**

Now that we have covered how convolutional neural networks (CNNs) work, let’s train one! We’re going to use the same dataset from the first module to train a CNN model on Edge Impulse rather than a simple dense neural network.

You are welcome to use [my electronic components dataset](https://github.com/ShawnHymel/computer-vision-with-embedded-machine-learning/raw/master/Datasets/electronic-components-png.zip) if you did not collect your own.

**Convert Images**

At this time, Edge Impulse only accepts .png and .jpg images. If you originally collected your dataset as .bmp files, you will need to convert them to one of those two categories. I recommend .png, as it is a [lossless format](https://en.wikipedia.org/wiki/Portable_Network_Graphics#Compression).

* Windows: use something like [Bulk Image Converter](https://sourceforge.net/projects/bulkimageconver/)
* Mac: use the [built-in tools](https://www.makeuseof.com/tag/batch-convert-resize-images-mac/) or something like [XnConvert](https://apps.apple.com/us/app/xnconvert/id436203431)
* Linux: [ImageMagick](https://www.imagemagick.org/script/index.php) has some good command line tools for converting images

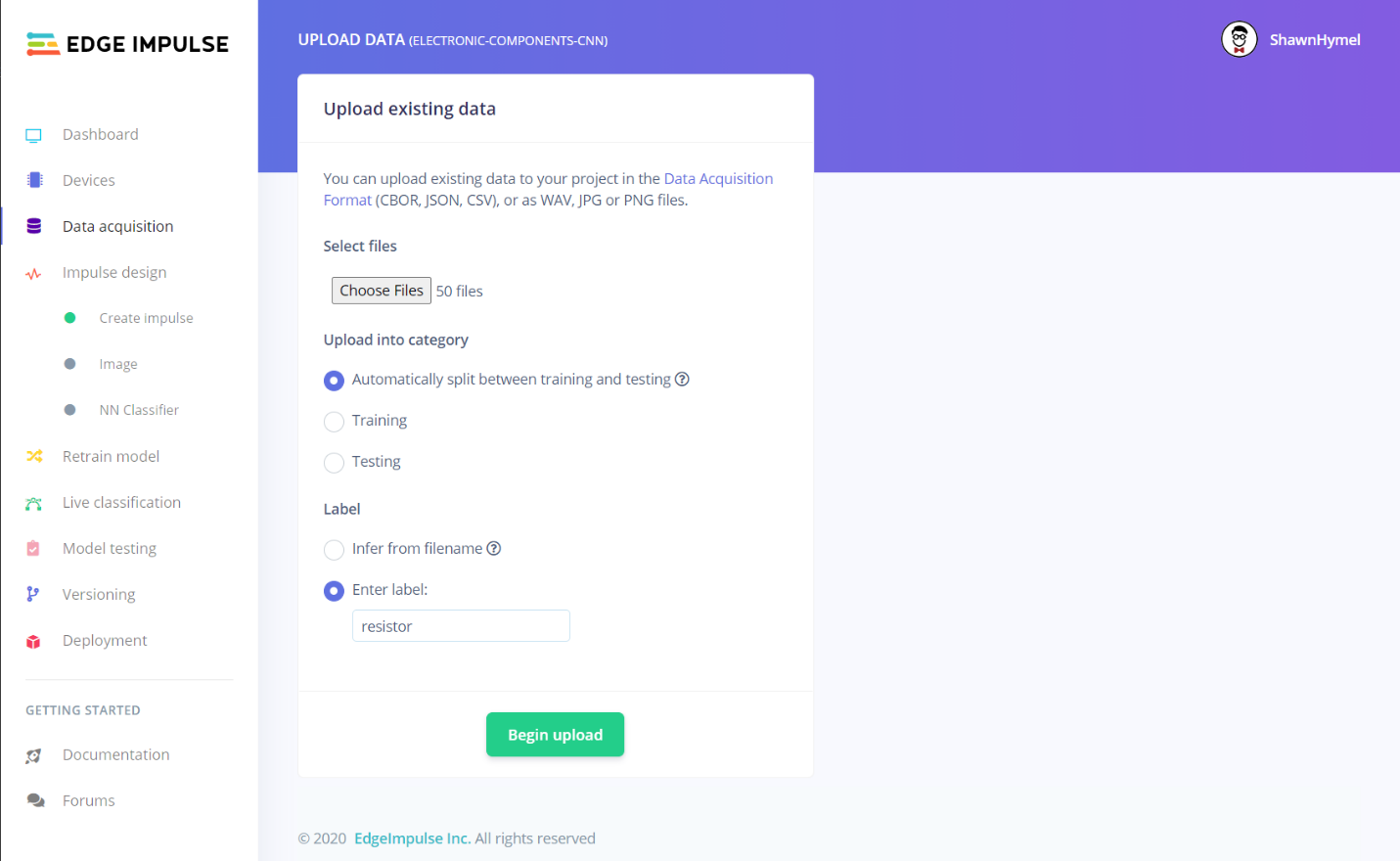
Or if you’re feeling particularly enthusiastic, you could write a quick Python script (using something like [PIL](https://pillow.readthedocs.io/en/stable/reference/Image.html)) to get the job done.

I recommend keeping the directory structure the same so that you can determine the label of each sample by just looking at the name of the containing folder.

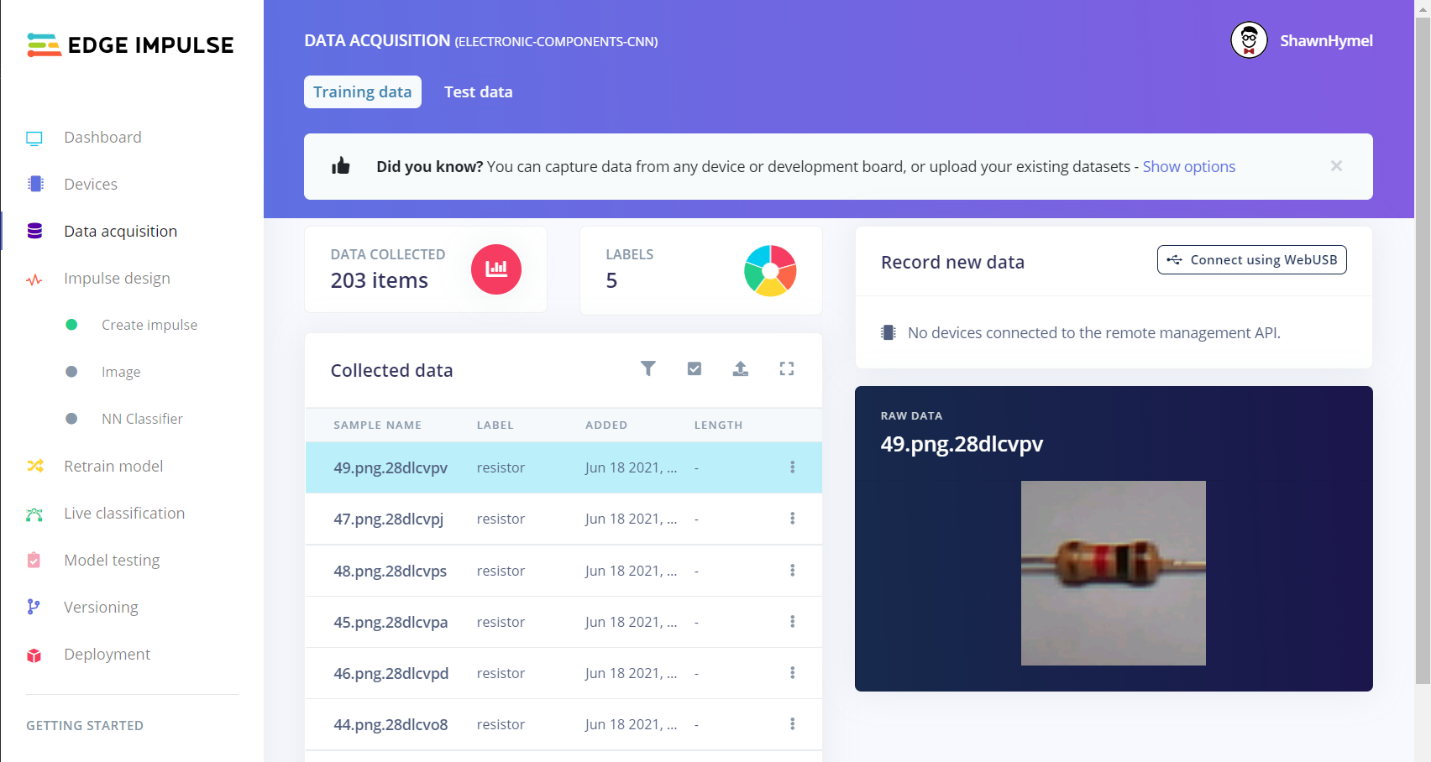
**Upload Dataset**

The good news is that we can upload image data directly to Edge Impulse rather than having to manually extract features using Colab.

Start a new Edge Impulse project. Go to **Data acquisition**. Click the button to **Upload existing data**. Click **Choose Files** and select all images for one class (e.g. resistor class). Leave *Automatically split between training and testing* selected. For *Label*, enter a label for your first category (e.g. “resistor”).



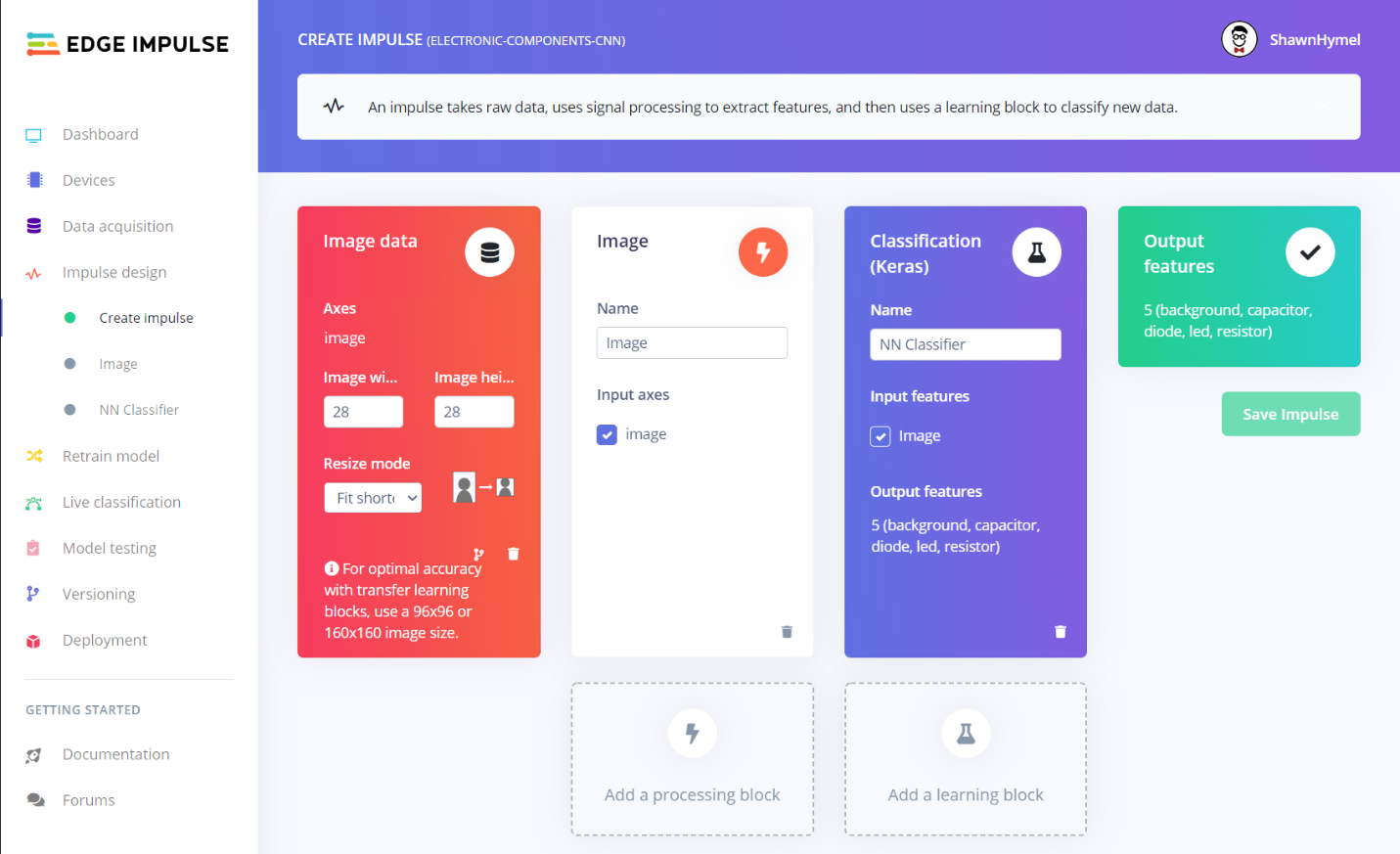
Click **Begin upload**. Repeat this process for the rest of the categories in your dataset. When you are done, go to **Data acquisition**. Make sure that you have a good representation of samples in your training and test sets.



**Create Impulse**

Head to **Impulse design**. Change the *Image data* setting to **28 x 28** (width x height). If your original images are not all square, you can adjust the *Resize mode* as needed to crop or squish the images.

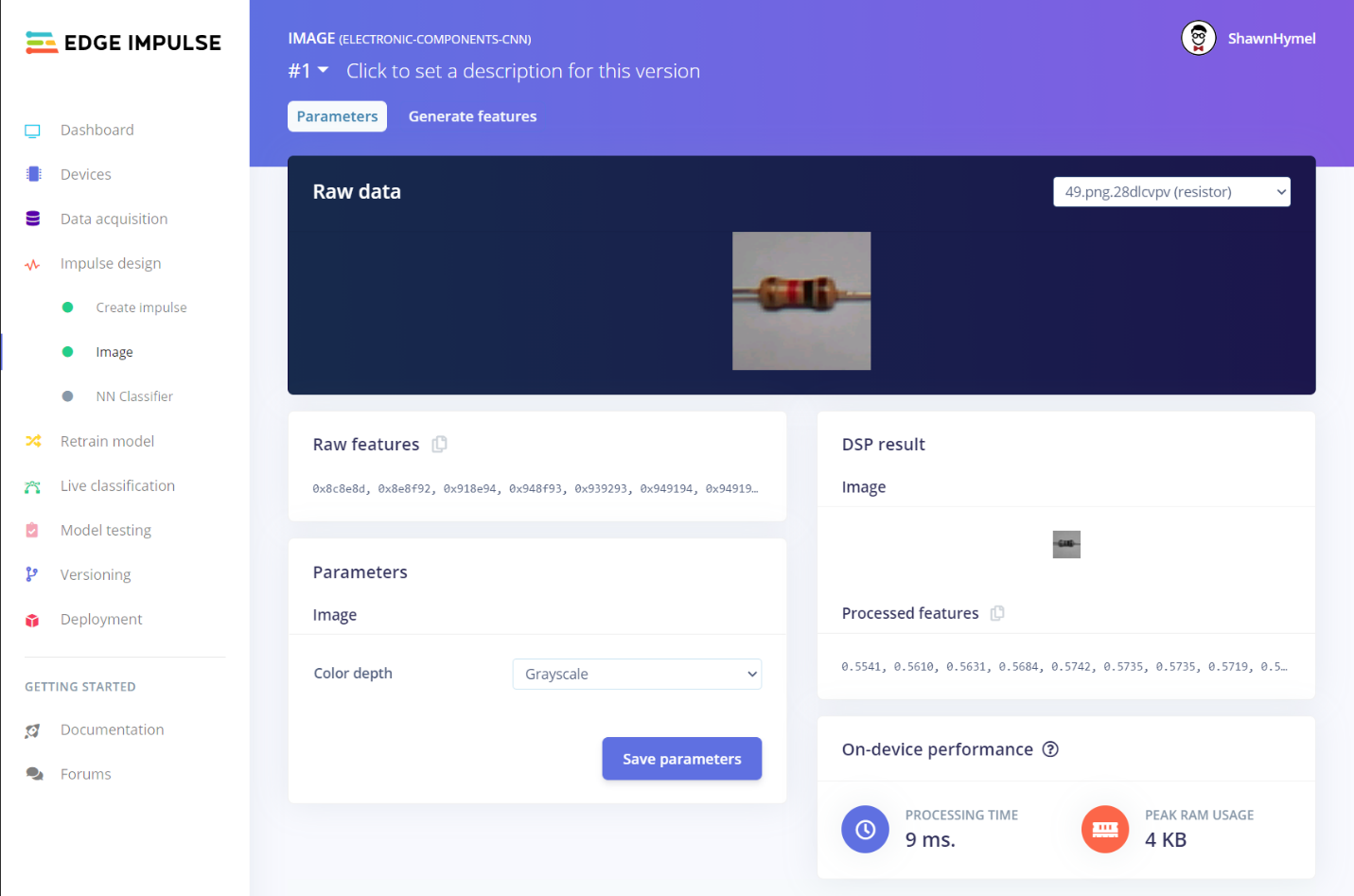
 Add an **Image** processing block. Add a **Classification (Keras)** learning block.



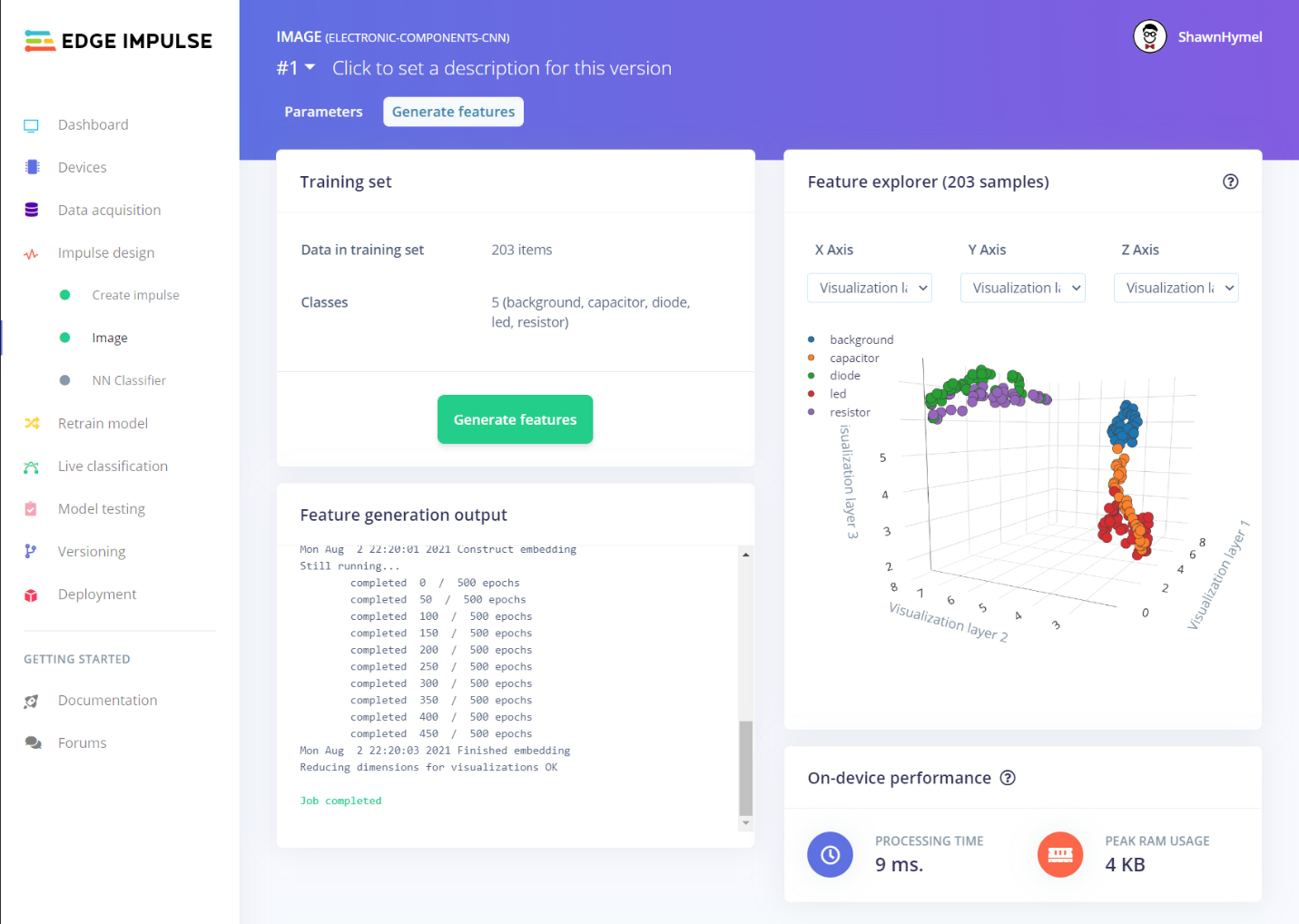
Click **Save Impulse**.

**Extract Features**

Go to **Image** under *Impulse design* on the left-side navigation bar. Change *Color depth* to **Grayscale**. Click **Save parameters**.

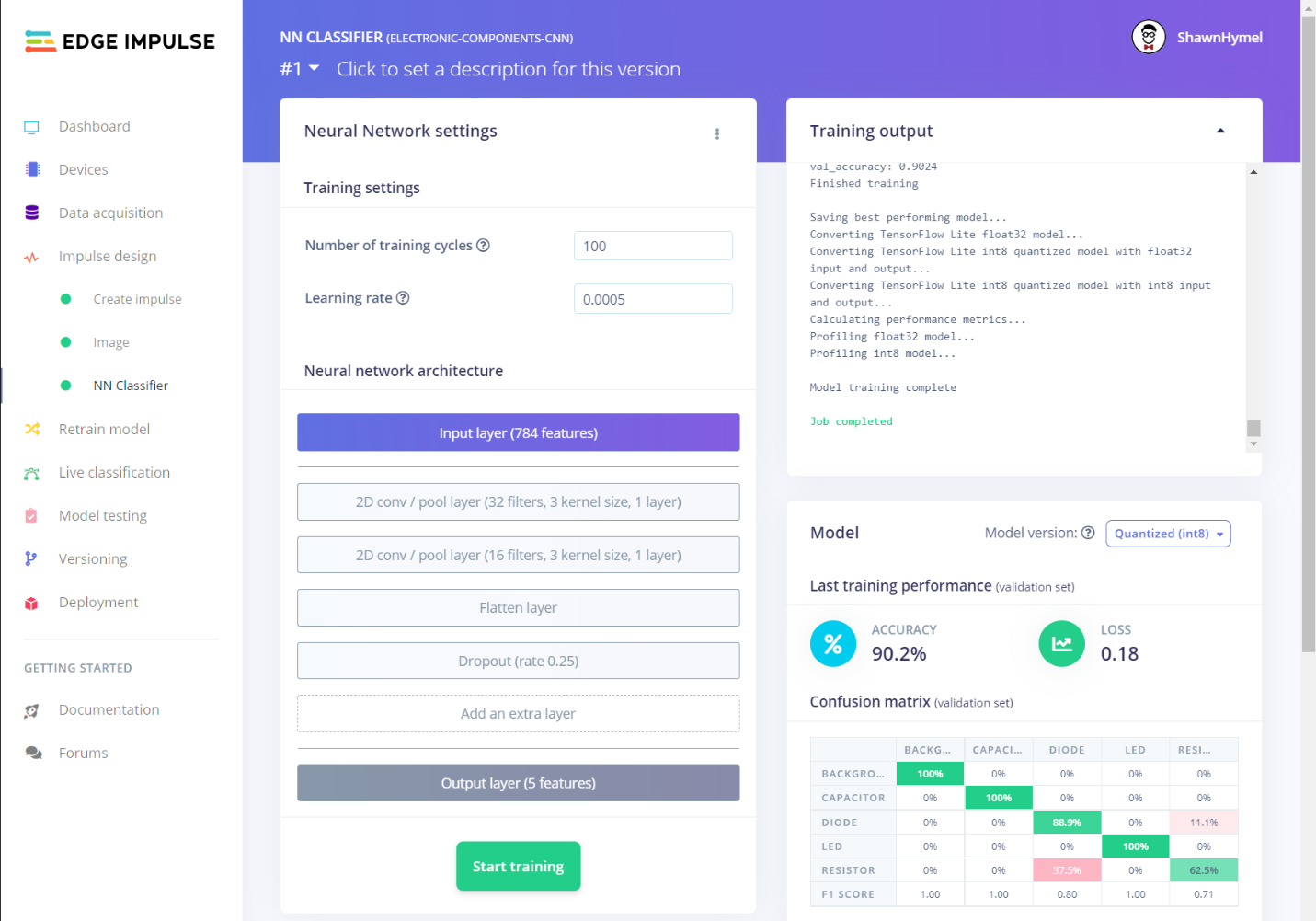


On the next screen, click **Generate features**. Wait until the feature generation script is complete. Here, images are being converted to grayscale and 28x28 pixels.

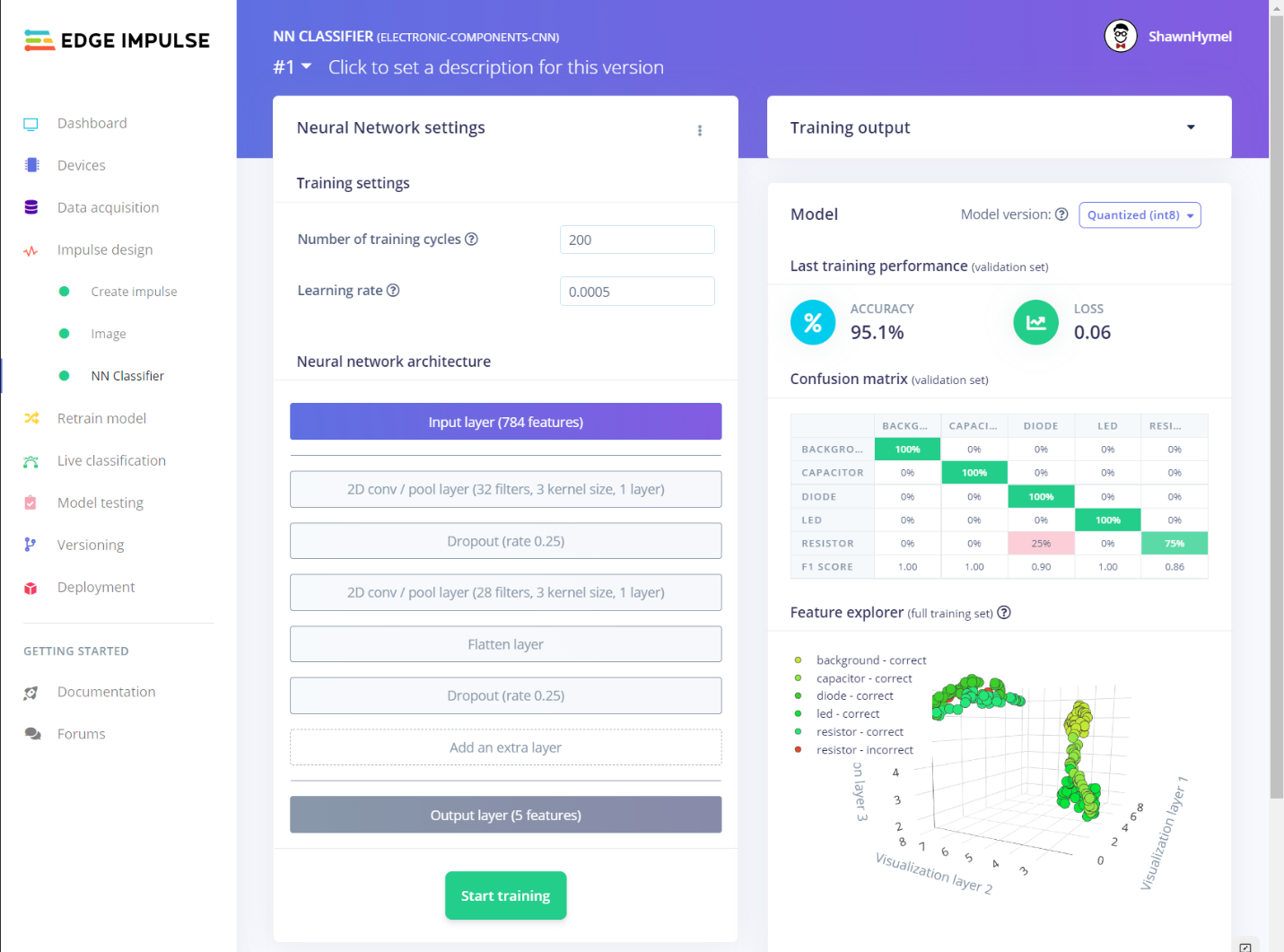


**Train!**

Go to **NN Classifier**. I recommend changing the *Number of training cycles* to something between 100 and 200 (as we are working with a small dataset). Click **Start training**. When training is done, how well did your model perform?



Try adjusting the model hyperparameters, such as number of training cycles, number of filters, and kernel sizes. You can try adding and removing layers. For example, I added a dropout layer after the first convolution layer and changed the second convolution layer to have 28 filters. This seemed to provide a better accuracy on the validation data.



**Conclusion**

Comparing the accuracy results of this model with those of the dense neural network example might be misleading. Remember that we are working with a relatively small dataset (50 samples per class), and the samples are likely randomly distributed between training and test sets. Moving a single sample in or out of the training set could cause the validation accuracy results to sway by 0.5% (assuming 200 samples in the training set). A larger dataset would help prevent overfitting and give you a much better idea of the model’s viability. We will explore ways to create larger datasets in a future lecture and project.