Lecture 5 - Image Classification part 2

Watch this video and this video first!

Codes here (1.3.1 - Using Colab to Curate and Upload a Dataset)

Introduction

Now that you have created a dataset, let's extract features and send them to Edge Impulse to train our first model. You have a couple of options for this project:

1: Run the pre-made Google Colab script to extract features and upload raw features to your Edge Impulse project

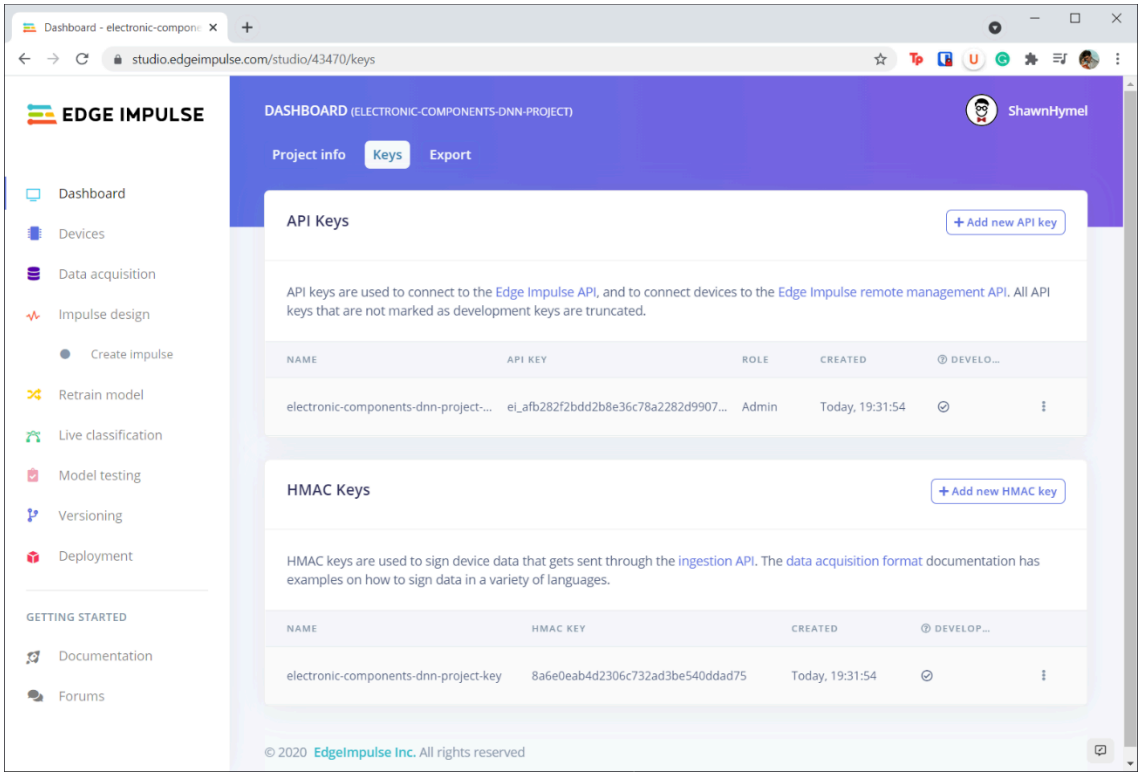
2: Edit the pre-made Colab to curate the dataset in a different manner. This will be more difficult, but it will give you practice using Python to manipulate files.

Either option is fine. I recommend choosing the second option if you want a harder challenge, as you will actually need to read, understand, and modify my code.

Create Edge Impulse Project

If you have not done so already, create a profile on https://www.edgeimpulse.com/. Click your profile picture, and click **Create new project**.

Give your project a name. If you get a pop-up window, close it. Go to **Dashboard > Keys**. You will need to copy both the *API Key* and the *HMAC Key* into your Colab notebook in the next part.



Note: Even though the key(s) might look truncated, if you double-click on the key string and copy (e.g. ctrl+c), the entire key will be copied.

Curate Dataset

Here is where you get to choose one of two options for this project. Either way, you will need to open this Colab notebook.

You should use the images you collected in the previous project. You are also welcome to use my dataset: https://github.com/ShawnHymel/computer-vision-with-embedded-machine-learning/blob/master/Datasets/electronic-components-png.zip

Option 1: No Editing (Easy Mode)

Simply follow the directions in the notebook. You will need to paste in the *API Key* and *HMAC Key* from your Edge Impulse project.

Additionally, you will need to upload your image dataset as shown in the description to create a particular directory structure. The individual folders should have the same names as the labels.

Option 2: Edit the Notebook (Hard Mode)

Instead of a multi-class classifier that uniquely identifies all of your classes, let's say you want to train a model that identifies one of your classes versus the others.

For example, instead of

- background
- capacitor
- diode
- led
- resistor

as your classes, let's say you wanted to identify just resistors. You might have resistor vs. other (binary classifier). However, I might recommend keeping the background class (multi-class classifier). So, you would end up with:

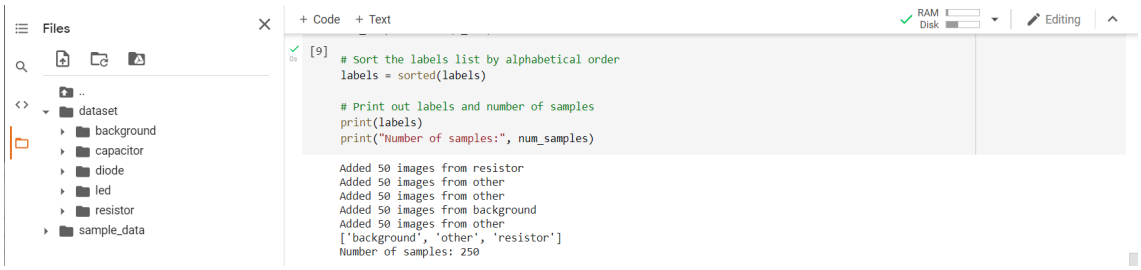
- background
- other
- resistor

The easy way to do this would be to copy all the files from the other classes into an "other" folder and create a new dataset. However, this might be infeasible when you are working with very large datasets.

You should be able to do this in code: as you load each image, pay attention to the folder. If it's "background" or "resistor," simply leave the label alone. If it's anything else, change the label to "other."

I recommend modifying the cell that starts with the comment `### Load images as Numpy arrays` to accomplish this task.

I have 50 images in each category. When I modify this cell to be resistor vs. other vs. background, it should show that there are 50 images in *resistor*, 50 images in *background*, and 150 images in *other*.

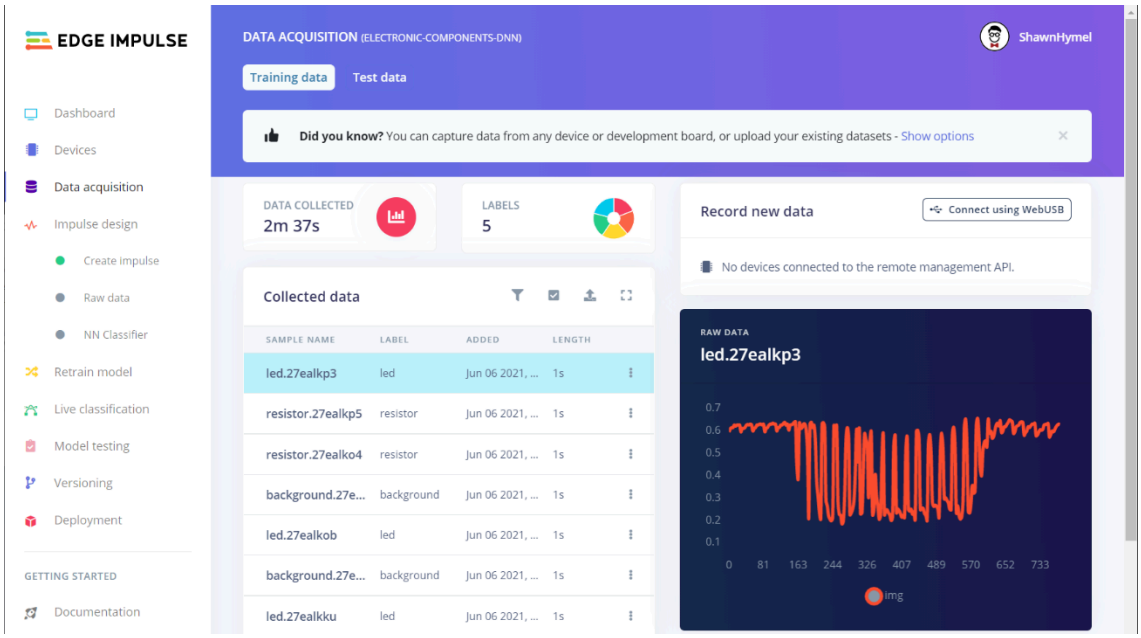


I recommend trying this option, as it will force you to read and understand Python code for data curation! If you get stuck or want to compare answers, [my solution can be found here](#).

Train Model

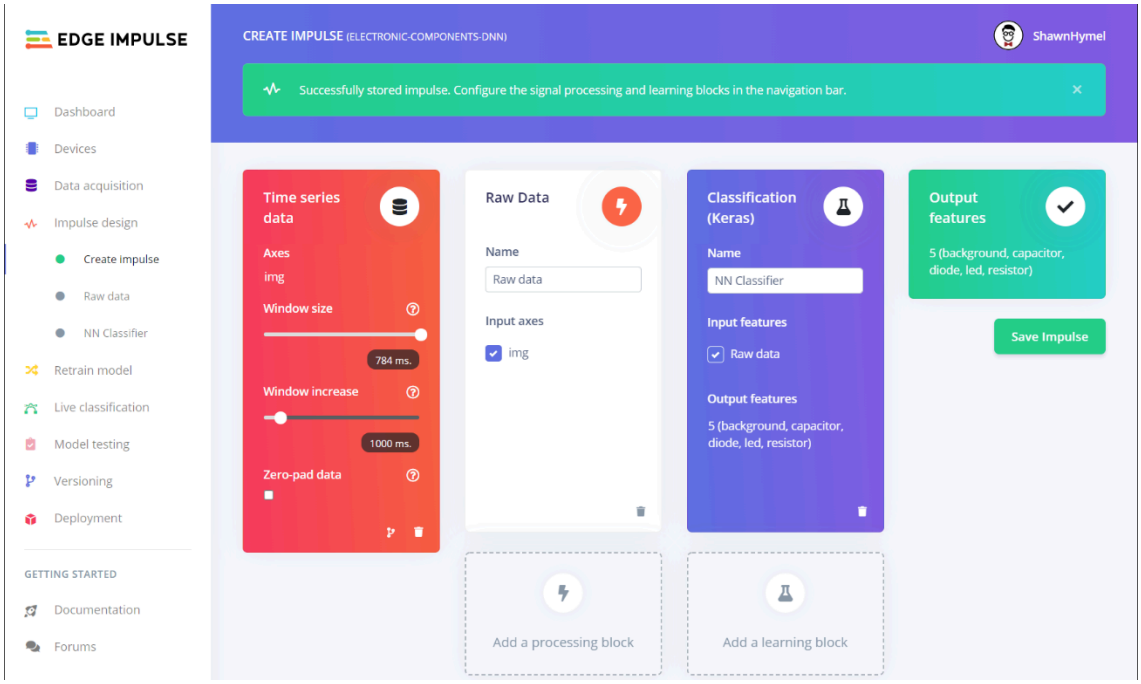
Go back to your Edge Impulse project and click on **Data acquisition**. Feel free to look through your data to make sure everything was uploaded properly.

Note: Edge Impulse thinks we're using "time series" data for this project, as there's no way to upload raw without any sort of units at this time (it's either time series or 2D images). Remember, we've normalized and flattened our images to 1D vectors! This is a bit of a hack, as we are tricking Edge Impulse to think our raw 1D vectors are "time series" data. However, it will still work. Normally, you'd want Edge Impulse to do feature extraction for you on images, which we'll explore later.



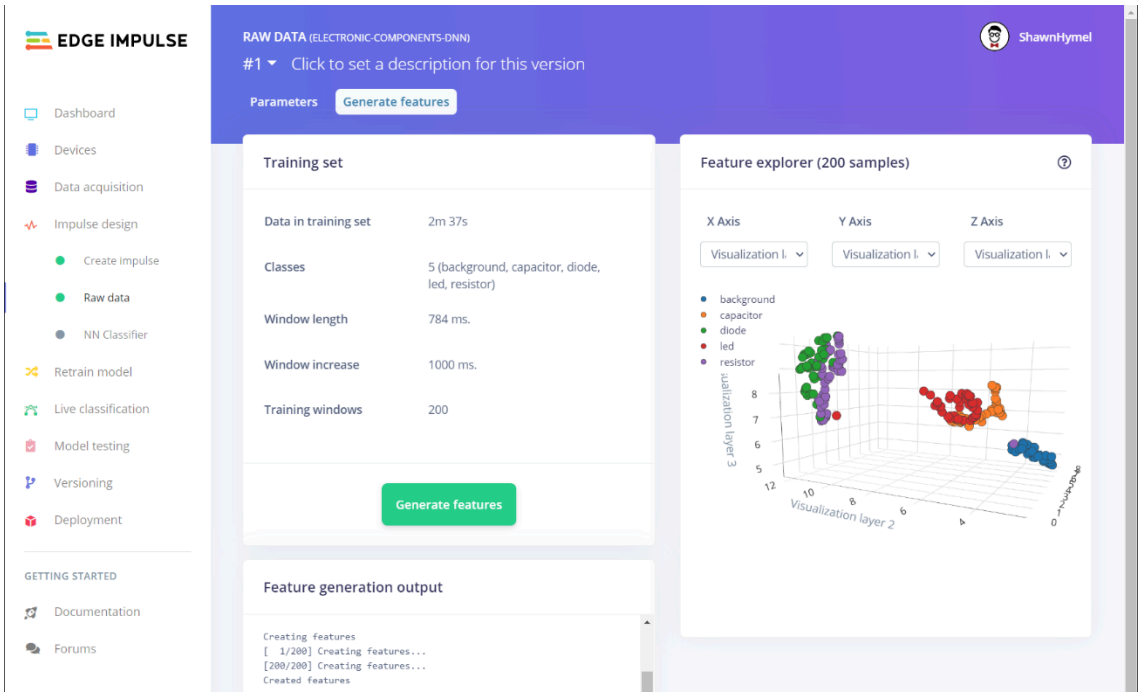
Click on **Impulse design** and you should see a *Time series data* block. By default, the *Window size* is set to 1000 ms. However, we only have 784 data points per sample, so we need to change it to **784**.

Add a **Raw Data** block for your *Processing block* and a **Classification (Keras)** block for your *Learning block.* Click **Save Impulse**.



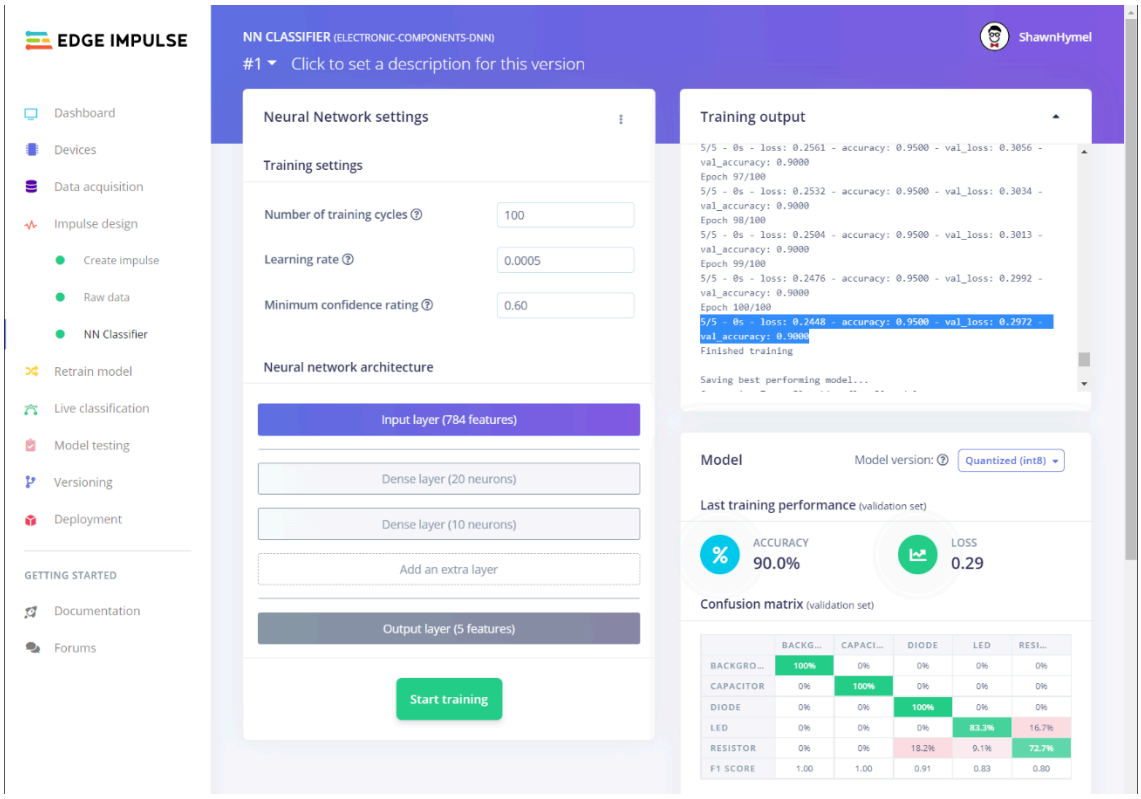
Click on **Raw Data** in the explorer pane on the left side. Click on the **Generate features** tab, and click **Generate features**. Wait a moment while Edge Impulse extracts features (there are no real features to extract here--it's just wrapping our vectors up in a way that can easily be used by Keras and reducing some dimensions for the Feature Explorer).

Feel free to look at the *Feature Explorer* window to see if your classes are easily separable.



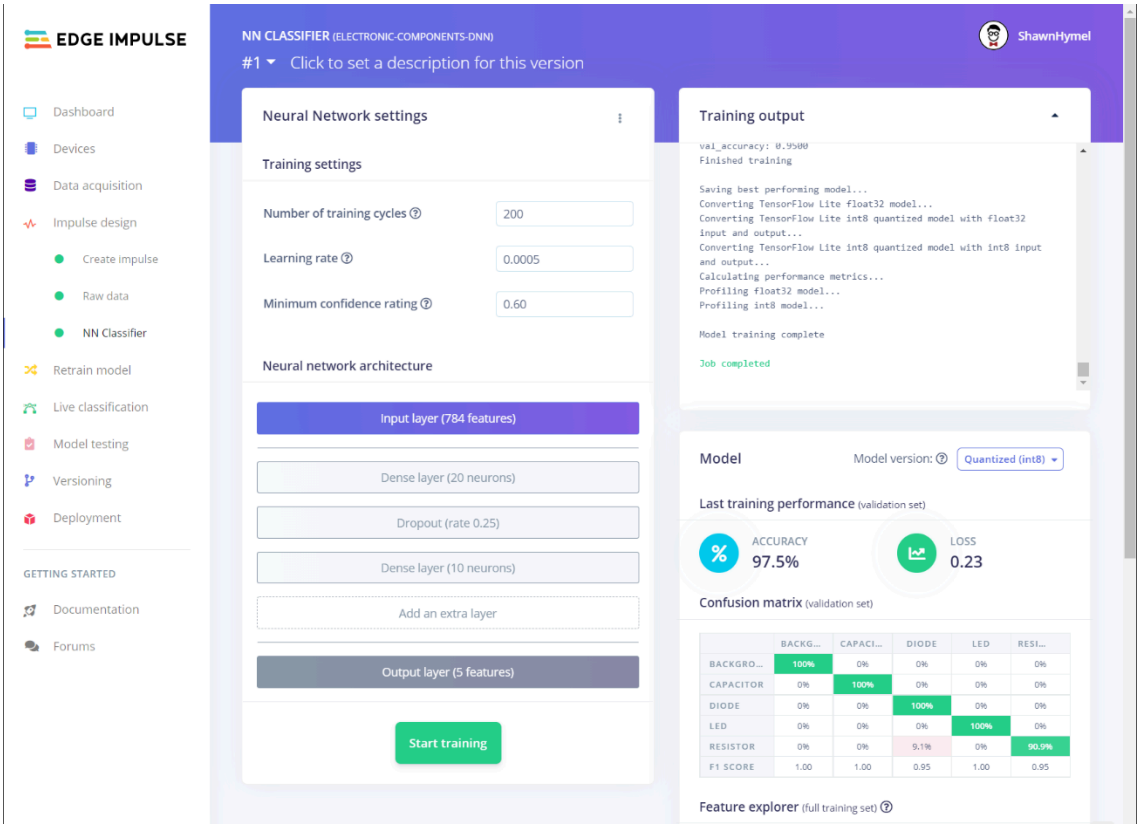
Click on **NN Classifier**. I recommend changing the number of *training cycles* to 100, as 30 did not seem to be enough to have the model converge for me. Click *Start training*. Once training is done, take a look at the accuracy score and confusion matrix for the validation set.

Scroll through the training output to compare the training loss/accuracy to the validation loss/accuracy.



How well did your model perform? It looks like there was some overfitting for my model, as the training accuracy was 95% and the validation accuracy was 90%.

Try changing the number of neurons in the first layer, adding additional dense layers, or adding dropout layers to see if that helps to fix any issues you might have, such as overfitting or underfitting. Note that as you make your model more complex, you might need to increase the number of training cycles.



For example, I added a dropout layer (with 0.25 dropout) and increased the training cycles to 200. Those changes seemed to help my model performance

Note: Remember that we are working with a relatively small and simple dataset here! In future lectures, we will explore better models and learn how to increase the size of our dataset to make our model more robust.

(Optional) Train a Model with Keras

If you would like to see how to use code (Keras) to train a similar model, open my Colab example here. Detailed explanation available here.

Upload your dataset and work through the notebook to train a model. How does the model trained with Keras compare with the one trained on Edge Impulse?

You are welcome to try making changes to the model architecture (changing the number of nodes, adding dropout layers, etc.). Here is the Keras Layers API if you would like to see the reference documentation: https://keras.io/api/layers/