CASA0018 Deep Learning

Design and build of a handwriting to digital text AI-powered stylus

Ethan Low

INTRODUCTION

This project seeks to test the feasibility of a low-cost AI-powered smartpen concept, by recognizing gestures specific to the handwritten numerals 0 to 7 using a Convolutional Neural Network (CNN) classifier.

RELATED WORK

One of the oldest problems in human computer interaction has been the transfer of human gestures from drawing or writing into digital formats. As early as the 1970s, this problem was solved using electronic graphics tablets, which make use of a stylus, paired with a sensor pad, to capture marks drawn by a user as a data, with the invention of the first modern electronic tablet largely attributed to the RAND corporation (Machover, 1978; Ware, 2008). Though remarkable for its time, the RAND tablet was extremely expensive, limiting its sales (Bauman, 2018). Contemporary tablets such as those built by Wacom \* are far more affordable in price but are still limited in that they still require a sensor surface for collecting pen stroke data.

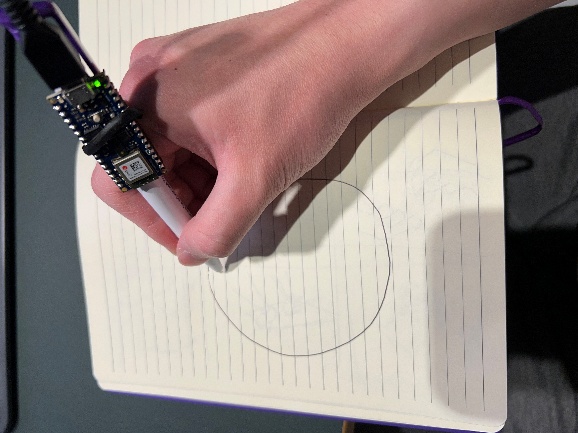
Recognition of pen gestures remains at the cutting edge, with new research and commercial products continually emerging. For example, (Tian *et al.*, 2013) investigated the use of pen tail movements to initiate interactions with a computer, while the pen tip was being occupied by the user for another task. However, the researchers only examined and proposed gestures that could be recognized by classical machine learning classifiers based on orientation and panning data from the pen tail (Tian *et al.*, 2013). More recently, (Bi, Zhang and Chen, 2020) used a CNN deployed on a smartwatch to detect and correct poor pen holding gestures when writing. Though (Bi, Zhang and Chen, 2020) used more modern Deep Learning methods, they did not seek to try and recognize specific characters from handwriting. Nevertheless, (Bi, Zhang and Chen, 2020) do highlight that smartpens are significantly weightier than their original counterparts, and this can have negative implications for the product’s ergonomics.

The commercial field, a variety of smartpens can be found for sale, with notable examples including Nuwa pen (*Nuwa Pen | AI-powered Ballpoint Pen*, 2025) and the Neo Smartpen (*N2 – Writing experience as a pen with digital convenience*, 2020). The Nuwa pen is particularly notable for not requiring the use of specialized paper, instead relying on internal micro cameras to collect pen stroke data (*Nuwa Pen | AI-powered Ballpoint Pen*, 2025). However, the translation of pen strokes to digital text appears to be done only on the server-side (*Nuwa Pen | AI-powered Ballpoint Pen*, 2025).

INITIAL TESTING

Data collection methodology

For this project, it was decided to use the Arduino Nano 33 BLE Sense for the device, due to its small size, its available 256 KB SRAM which would be more than sufficient for a tiny CNN model, and its included onboard Inertial Measurement Unit (IMU) (*Nano 33 BLE Sense | Arduino Documentation*, no date). Initial testing was performed with this board to ascertain whether the IMU would be sensitive enough to the small-scale gestures of handwriting.

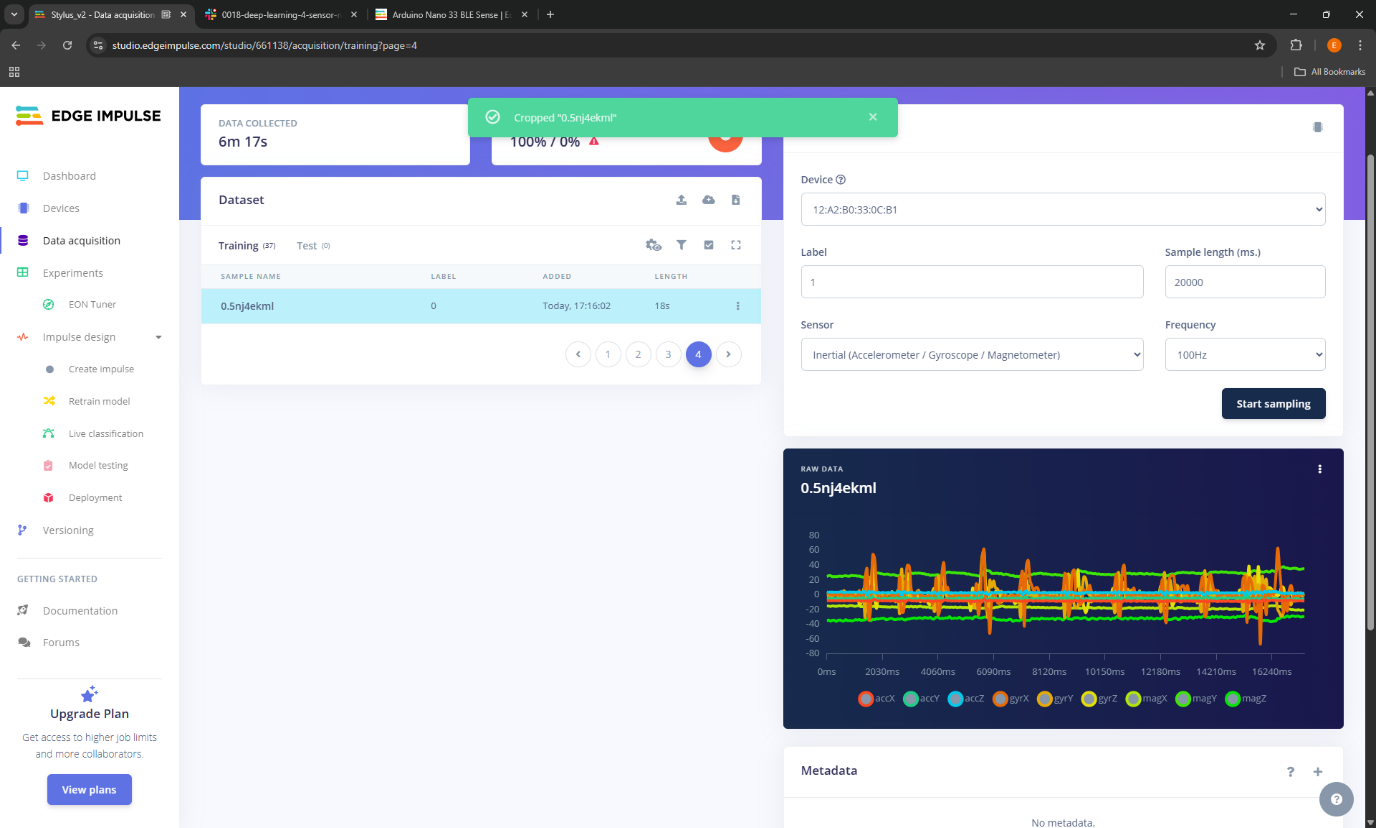
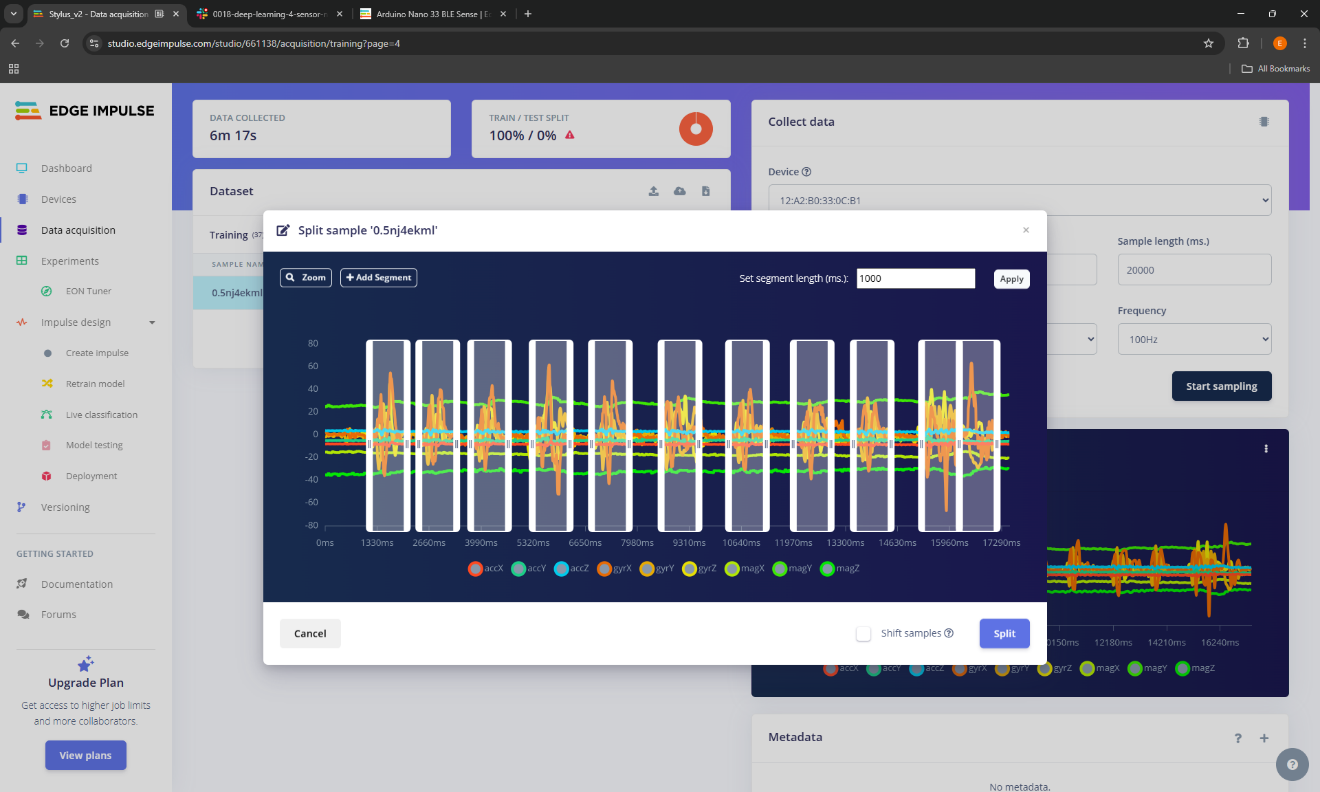
A hand holding a pen and writing on a notebook

AI-generated content may be incorrect.

*A large-scale handwriting gesture test with the Arduino Nano 33’s onboard IMU.*

Once it was clear that the IMU gyroscopic and acceleration data was of sufficient quality to be used, the full set of data for all the classes (numerals 0-7) was collected and processed using Edge Impulse.

Data processing & augmentation

*Screenshots from Edge Impulse showing the collected set of data for the handwritten numeral ‘0’ being split*

Data was augmented using Edge Impulse’s built-in sliding window tool, creating windows of 1500ms at 100ms steps from each raw sample. Initially, shorter samples were collected and a shorter window size used, but these were found to perform poorly during live model testing. The window size was therefore increased, along with the additional collection of longer samples where each numeral character was drawn at a slower pace.

It was found that distinctiveness between samples from each class could be increased by scaling the axes, which intuitively makes sense as the raw waveforms from handwriting do not have a large amplitude.

The decimation ratio of the input signal was also tired in the digital signal processing (DSP) block to try and reduce some of the noise in the data, but increasing this value did not result in any meaningful changes to the clustering. In fact, as more classes were added to the developing project, it was found that decimation made it harder to distinguish between some of the most easily confused classes such as 0 and 6 or 2 and 3.

FULL TESTING & REFINEMENT

Model architecture and hyperparameter refinement

Model refinement in deep learning is increasingly being regarded as an empirical science (Goodfellow, Bengio and Courville, 2016). In this spirit, a series of experiments was carried out, using Edge Impulse’s EON tuner to define a search space and run a battery of tests on different model variations.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

*Model performance at default 30 neurons 2nd layer and trained for 30 epochs*

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

*Model performance at default 30 neurons 2nd layer and trained for 60 epochs*

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

*Model performance at default 30 neurons 2nd layer and trained for 90 epochs*

* Epoch no.
  + ~90 epochs needed for 2 classes
  + ~120-150 epochs for 3 classes
  + 150 – 200 epochs for all 7 classes
* Neurons
  + 2nd layer - 30 more than sufficient for 2 classes, incrementally increased to 50 for 3 classes
  + 5th layer – 10 more than sufficient for 2 classes, incrementally increased to 25 for 3 classes
* Confusion
  + Matrix indicates 1 and 2 classes are where the greatest confusion lies
* Num of layers
  + Increasing number of dense layers to 3 seems to cause overfitting – increased loss in training

A relatively limited amount of data, compared to modern deep learning training workflows, was collected due to the time constraints of this project, with just over 100 samples collected for each class. Intuitively, it was therefore decided to add a dropout layer after each dense layer to combat overfitting, the risk of which was considered quite high considering the small dataset. According to pioneers of deep learning such as Hinton, a dropout rate of 0.5 is optimal for most tasks and therefore this was the rate that was defaulted to for the classification layers (Srivastava *et al.*, 2014).

Model with convolutional layers far outperformed model with the standard DSP block. This makes sense intuitively, as what we are trying to sense is a two-dimensional pattern with specific visual features. In fact, (Warden and Situnayake, 2019) specifically mention the use of convolutional layers for a very similar example project, in which gestures from a wand are recognized.

Nevertheless, the CNN-based model still benefitted from earlier adjustments to the dense layers, which were still employed within the classification block of this updated architecture.

DEPLOYMENT

Enclosure build

To prevent excessive vibration from being transferred to the accelerometer, a simple enclosure was designed in Fusion360 and used to securely attach the Arduino Nano to a pen.

Circuit

A simple circuit was created, soldered on a proto-board and attached as a shield to the Arduino Nano. Its purpose was to provide support for a 8-strip Neopixel LED strip, which would indicate the inferred classes once the model was deployed.

CONCLUSION

* Complexity of model (neurons & layers) is determined by the complexity of the problem. In a classification problem, more classes = more complex. Esp. if classes are hard to distinguish.
* Beyond a certain point, tuning hyperparameters and model architecture leads to very limited improvement => law of diminishing returns
* Single open line characters, e.g. 1 and 2 are less easy to distinguish compared to 1 vs 0 (a closed line char).