Bachelor Project - Sign Language

Tobias, Bertram, Silas, Patrick

February 29, 2024

Abstract

Your abstract.

Pre-introduction

Data

We've done all initial training with a dataset downloaded from Kaggle (asl-alphabet). The dataset in its current form is unlikely be the final (even if only for the ASL-alphabet). The dataset has a few shortcomings:

- The dataset contains duplicate images. There are a total of 3000 images for each letter, but identical images may appear ten (or more) times.
- The dataset contains only static images of the letters 'j' and 'z' which both require some form of hand movement.

That said, we chose the dataset because of it's size one GB in total. Such a size was manageable (could fit in memory) while we were learning the basics of machine learning.

Training a model

Following along the book, we began our machine learning endeavour.

What kind of input would we feed the model?

We drew inspiration from OUR BOY when deciding on the format of the model input, meaning we straight up copied his Python functions. Feeding the dataset through the MediaPipe Hands library yielded hand landmarks for all the ASL letters. Using the functions of **OUR BOY** the landmarks would be converted to image coordinates, then to relative coordinates, and lastly normalized to values between -1 and 1.

That is given the $base\_x$ and $base\_y$ coordinates of the index the relative coordinate of C is calculated as:

$$x = C_0 - base\_x$$

$$y = C_1 - base\_y$$

This process turned every ASL letter into a vector of 21 coordinates, one for each landmark. The coordinates would be relative to the position of the index, index being the point where the hand connects to the rest of the arm (wrist).

SGDClassifier

Our first classifying model would be a *linear regression model* using Stochastic Gradient Descent as the learning algorithm. It would be a binary classifier, capable of determining whether a sign was an ASL 'A' or not.

```
array([[3832,    9],
       [   4,  142]])
```

Figure 1: Confusion Matrix for A or not a

PUT IN PRECISION AND RECALL? Judging by the confusion matrix (figure 5) it was all looking very good.

Multi-class classification

A binary classification model was a good first step but still not too exciting. The next goal was to combine a bunch of binary classifiers into a single multi-class classification model.

One way to accomplish such is to use the *One versus One* strategy, where you train a binary classifier for each pair of labels, that is a model to distinguish 'A's and 'B's, 'A's and 'C's, and so on.

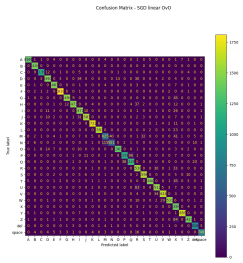

Figure 2: Confusion Matrix for multi-class

It was looking great! The model was doing very good overall, although it had some weaknesses, like mistaking 'U' for 'R' as can be seen on figure 2.

SOMETHING SOMETHING, PRECISION AND RECALL?

Training another one

We decided to also try a *logistic regression model*, specifically *Softmax* because of its ability to directly support multi-class classification.

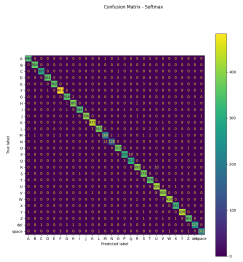

Figure 3: Confusion matrix for softmax

And the results were mind-blowing, what a win. Just look at the confusion matrix figure 3

CLASSIFICATION REPORT DUMP? Softmax did a lot better in general and almost completely eliminated the confusion between 'U' and 'R'.

0.1 Rights?..

The pictures of hand signs for letters used by our frontend is from https://www.wpclipart.com/ and is completely free to use, even for commercial purposes.

1 Appendix

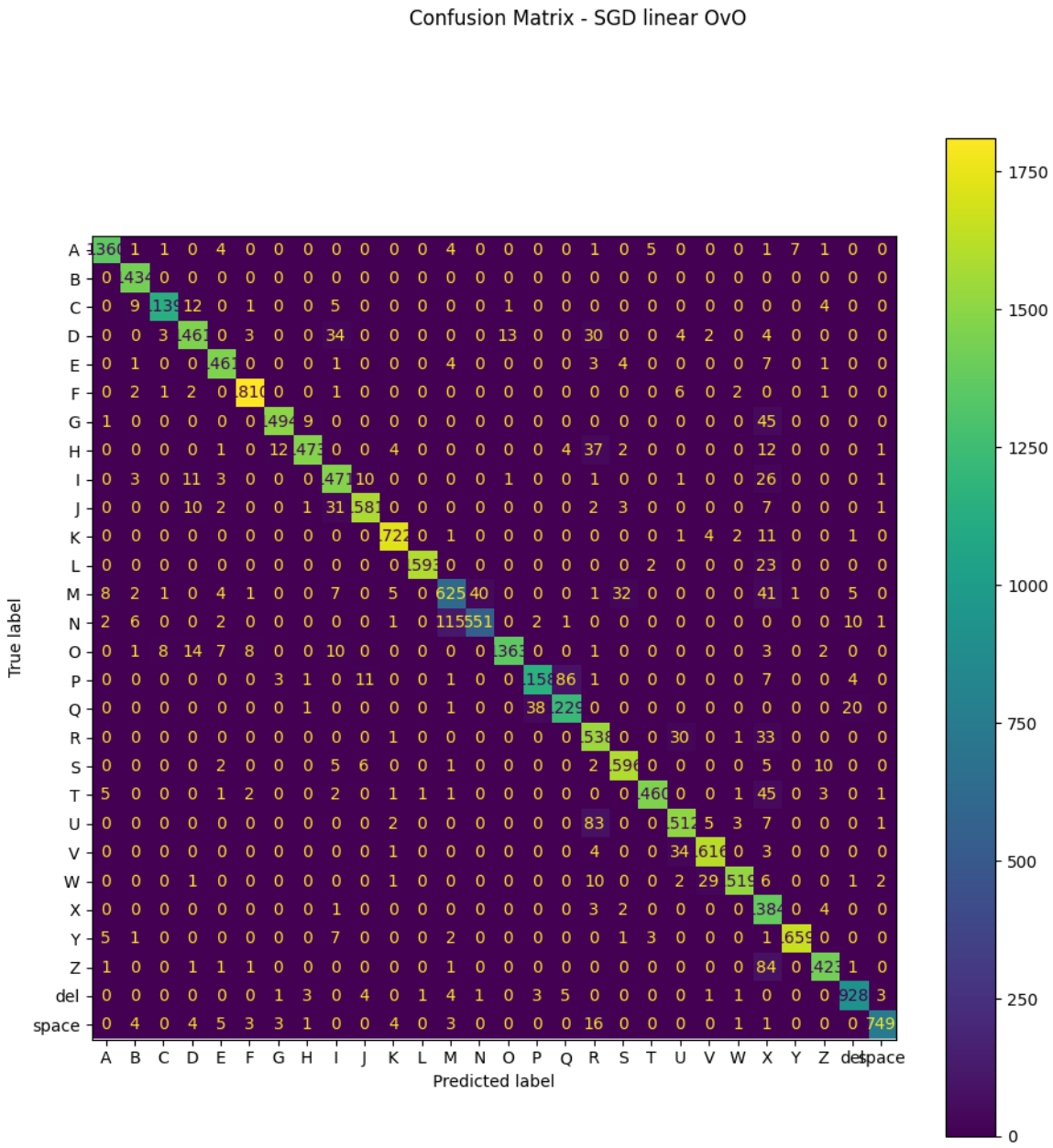

Figure 4: Confusion Matrix for linear model, trained using stocastic gradient descent

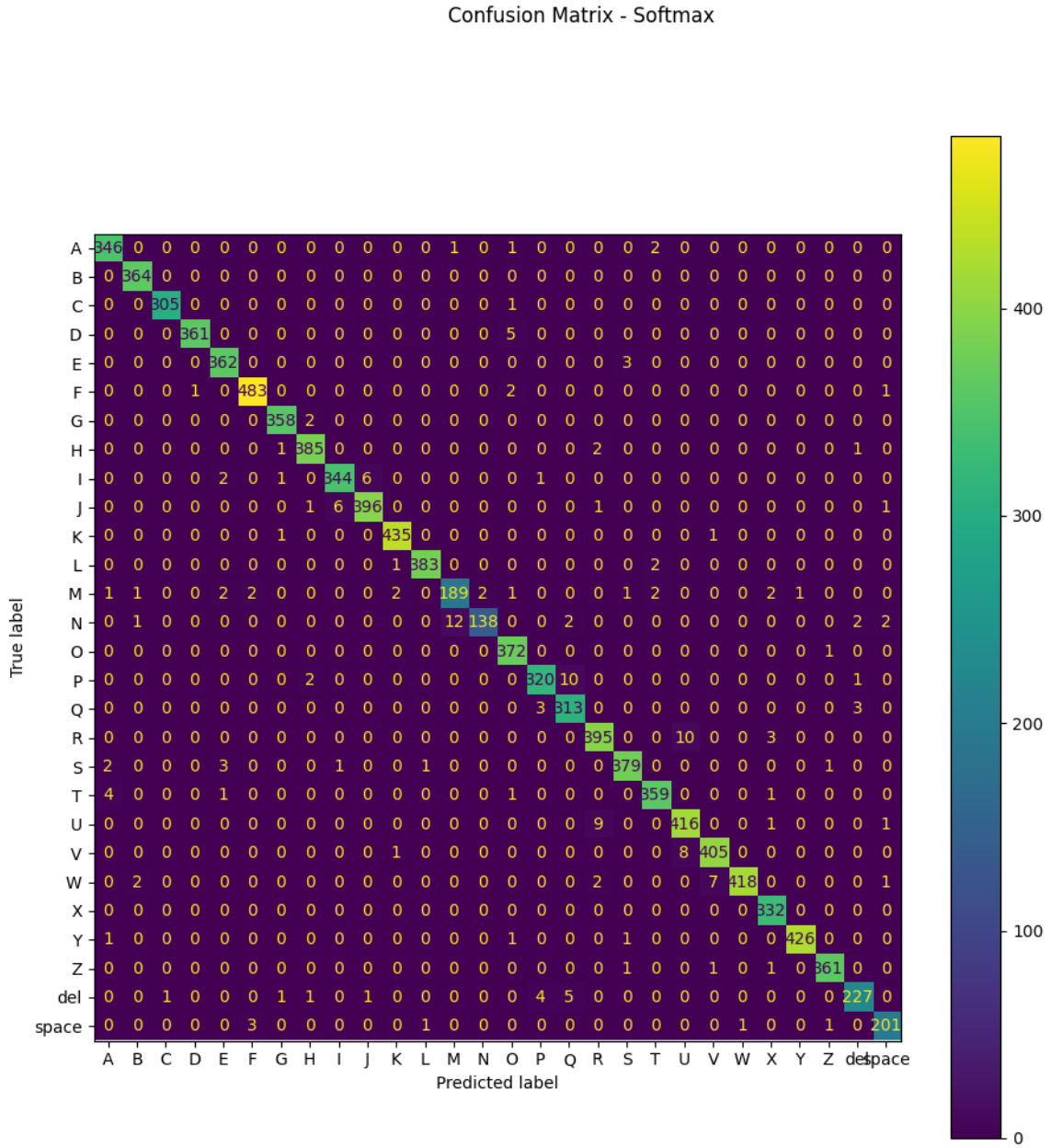

Figure 5: Confusion Matrix Softmax multiclass classification

References