

Building a training data generator for poker hand classification.

Jennifer Ada

School of Applied Computing, Faculty of Architecture, Computing and Engineering, University of Wales Trinity St David, Mount Pleasant, Swansea, SA1 6ED, Email: 0851115@student.uwtsd.ac.uk

Risky Data Sources

- After evaluating other poker hand classifiers it was concluded that some existing datasets such as the popular UCI's dataset[1] presented risks for mis-training neural networks which classify poker hands by the rank or value of the hand[2][3].
- The probability of a hand appearing in the UCI dataset is proportionate to how likely it is to be drawn from a deck of cards (Figure 1). Therefore, to mitigate the small sample of high value hand labels in the dataset (I.e. Straight Flush), it would be necessary to produce equal samples of each set of values.
- This is necessary to ensure that the data did not cause a classifier to only learn to classify some poker hands and mostly fail to classify others. One classifier had an accuracy of 92% which is also the total of 2 out of 8 of the classes of hands that exist in the dataset.

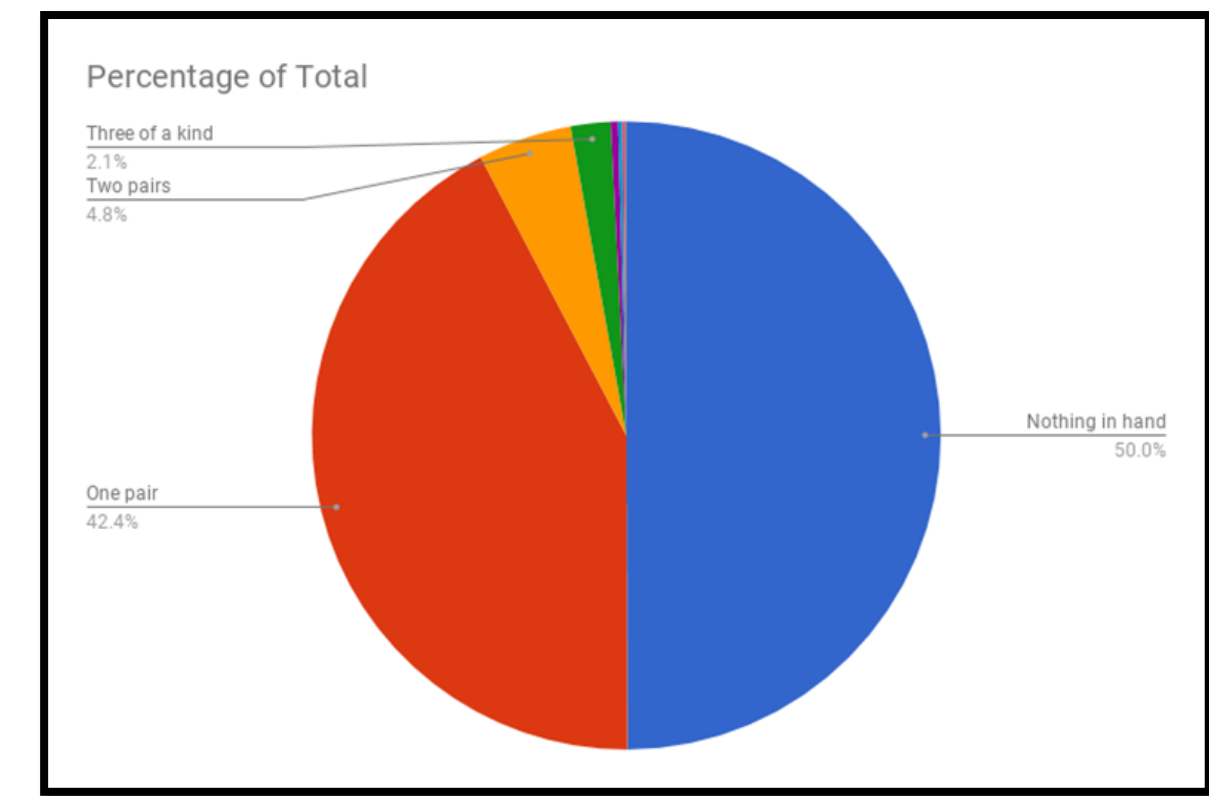


Figure 1 – Percentage of hand combination instances in UCI training data.

Objectives

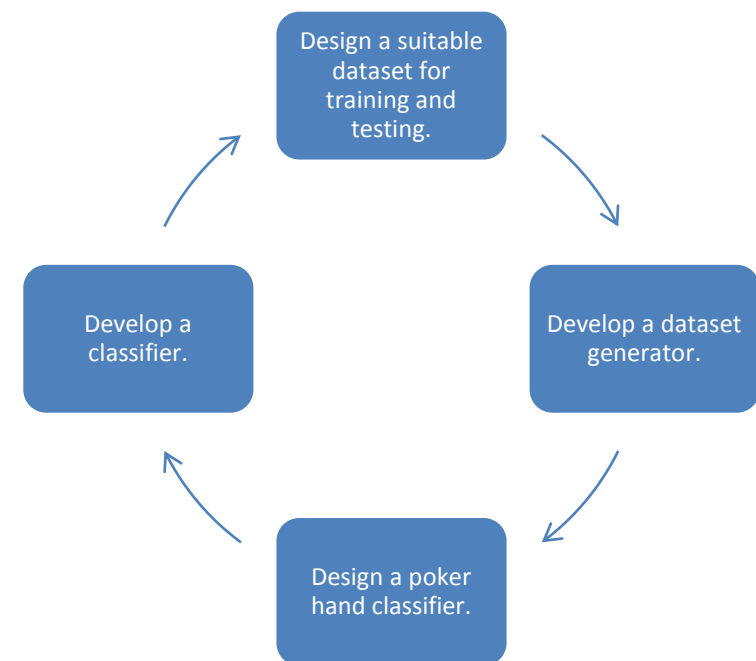


Figure 2 – Aims achieved through iteration.

Methods

- Objectives were composed of features.
- Each feature was prototyped separately as components which are then integrated into the core (Fig. 2).
- The TensorFlow[4] machine learning framework was used to design the target convolutional neural network allowing a layer of abstraction (Fig. 5,6).
- Python was used to develop a rules-based classifier (Fig. 4).
- The classifier was used to develop a data generator (Fig. 4), which adhered to the Unified Representation for Poker Games[5] (Fig. 3).

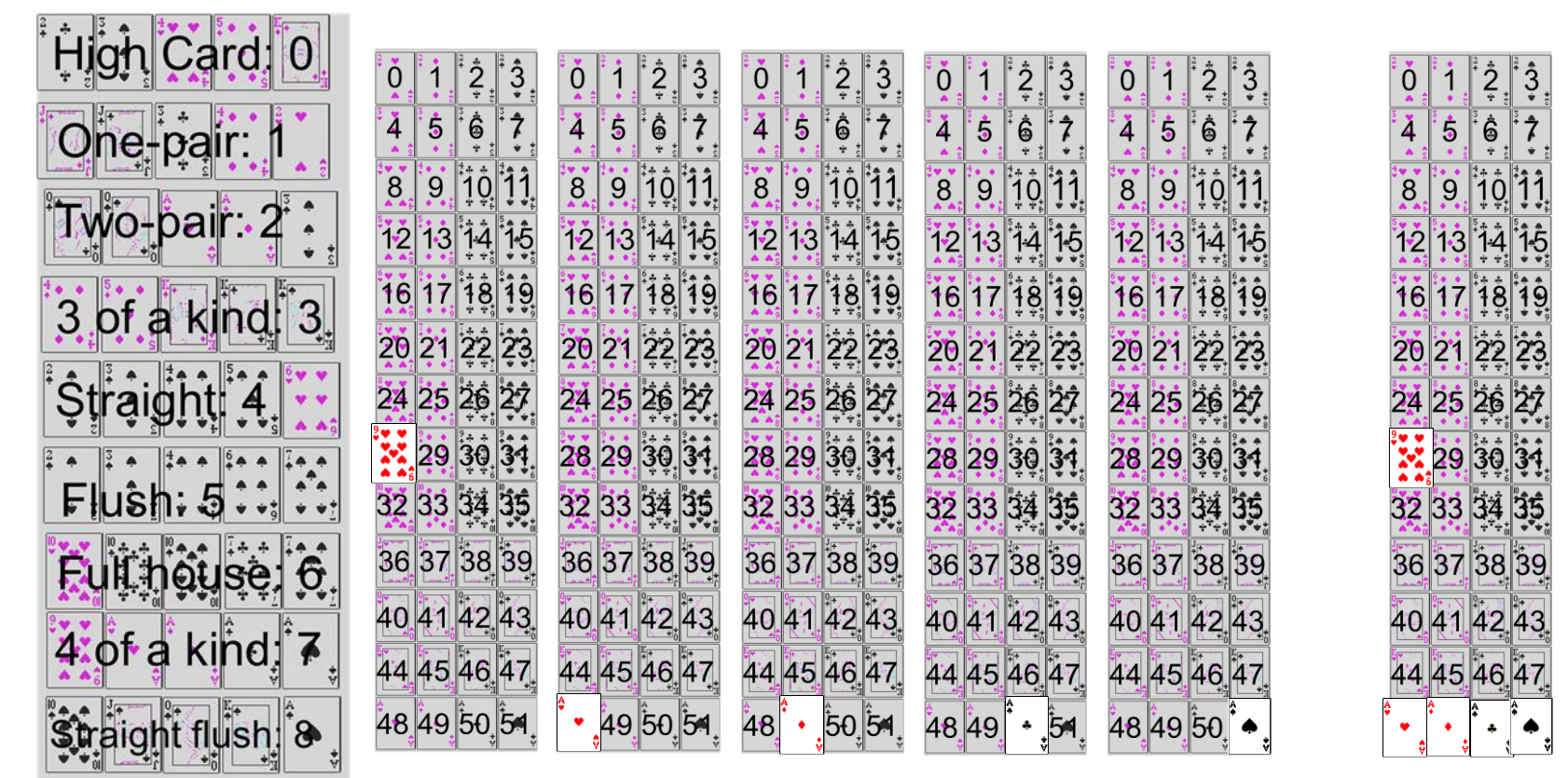


Figure 3 – Unified Representation for Poker Games Four of a Kind in CSV: "28,48,49,50,51,7"

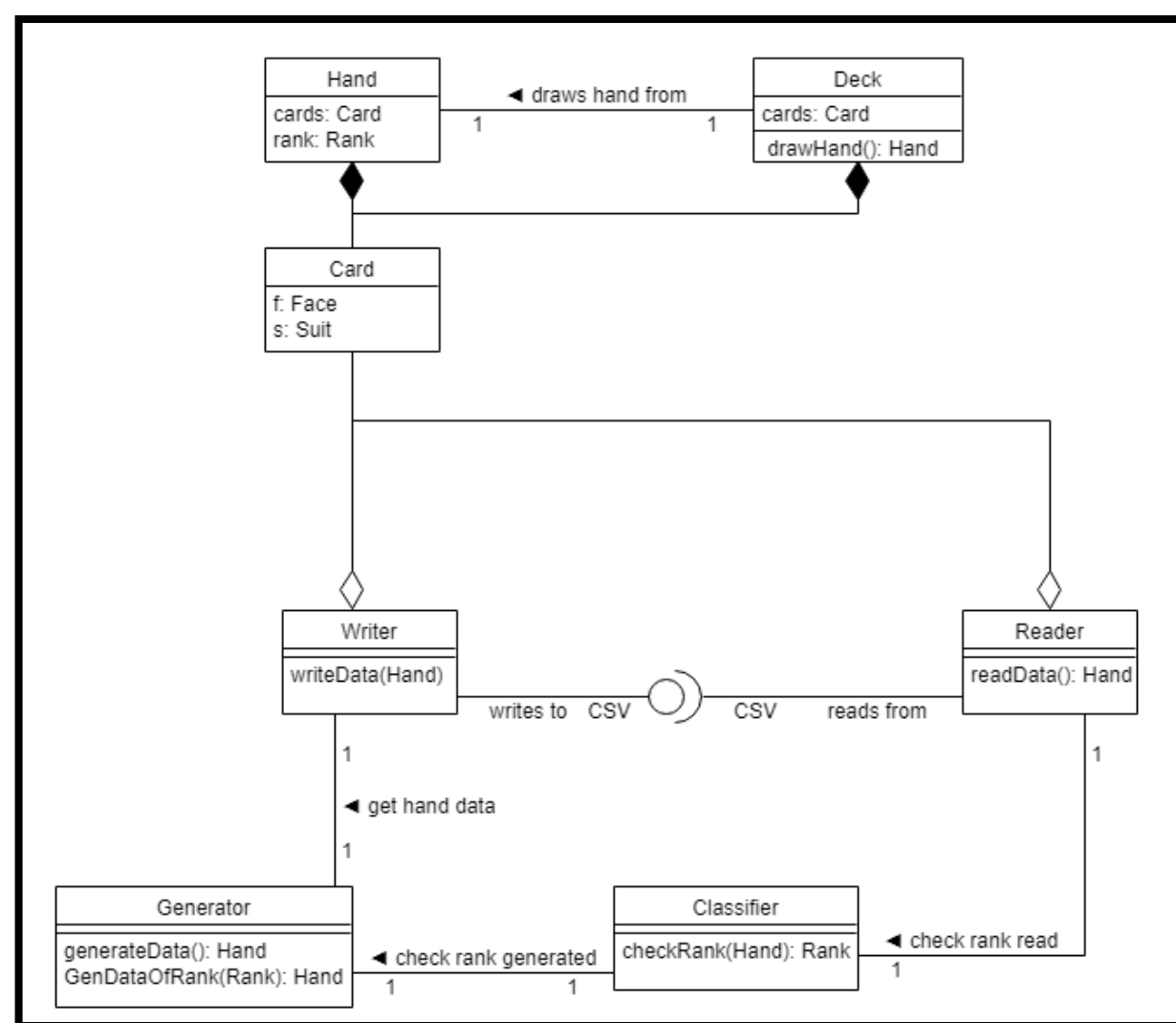


Figure 4 – Dataset Generator Design.

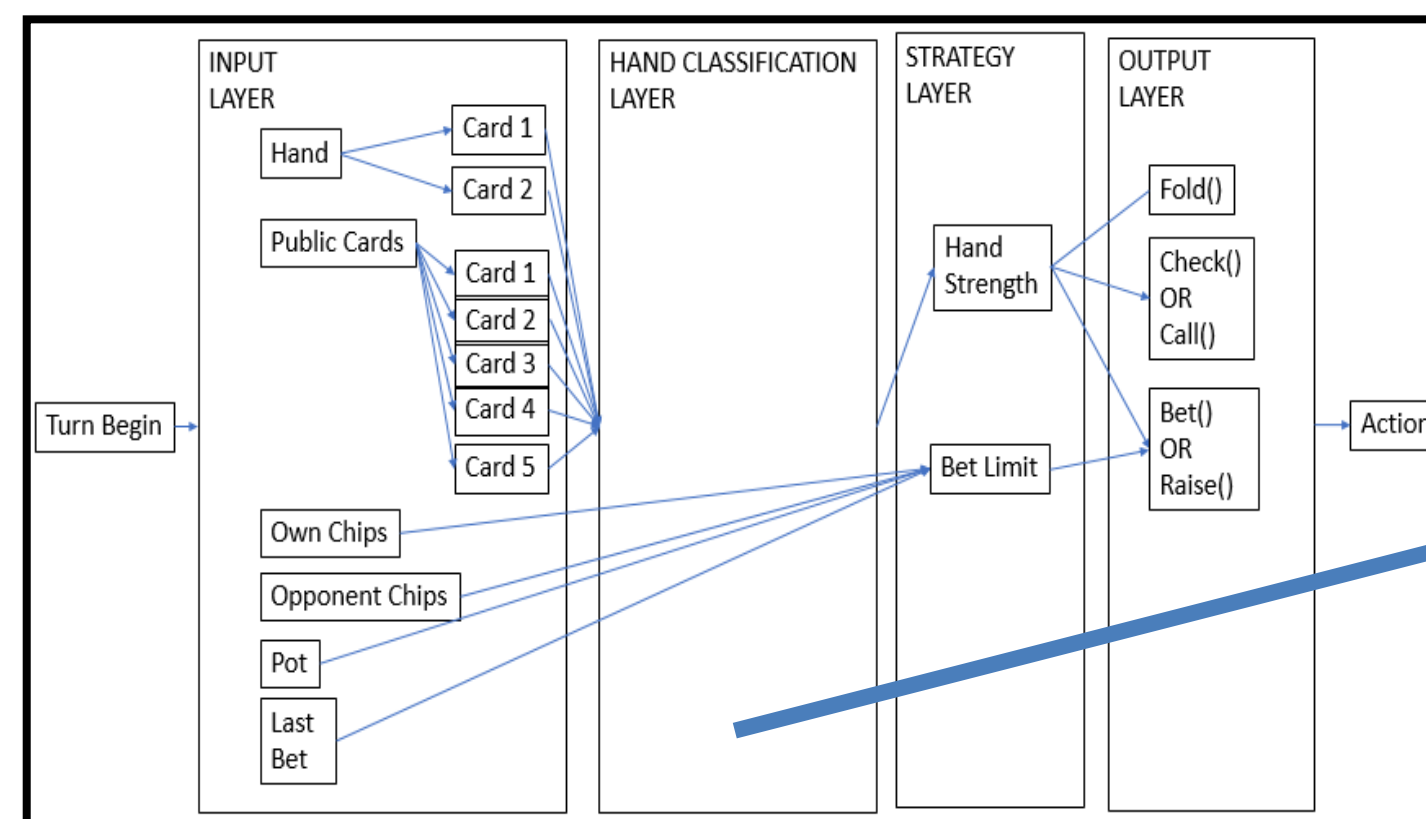


Figure 5 – Target Neural Network: Layers constitute modules based on the poker AI Loki's architecture[6].

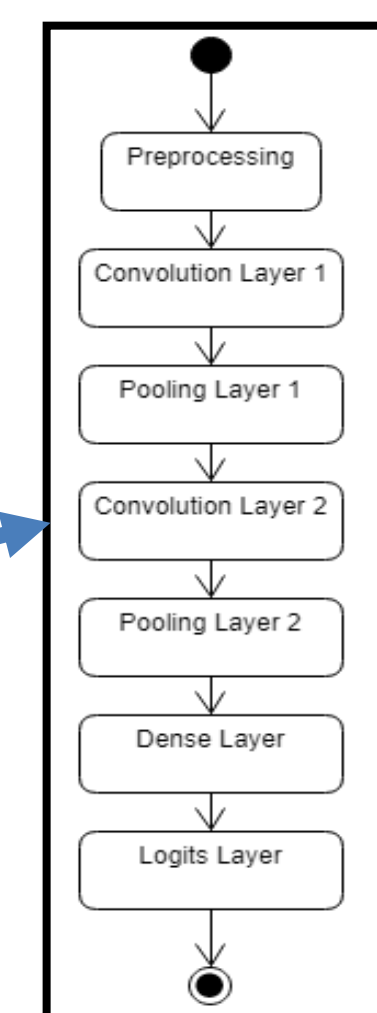


Figure 6 – Hand classifier convolutional neural network design based on Poker-CNN[5] and TensorFlow layers[4].

Outcomes

- Straight Flush hands were being generated slower than the other hands by a significant margin (3-4 hours for 1000 samples) because of the brute-force algorithm. The generator functions were benchmarked for time and the hypothesis was confirmed (Fig. 7,8).
- As a side-effect of developing a data generator, it was concluded that a rules-based classifier can achieve 100% accuracy as a result, this rules-based classifier may be used as a pre-processor for the neural network with the intention that this will reduce training time and resources while costing accuracy or flexibility.

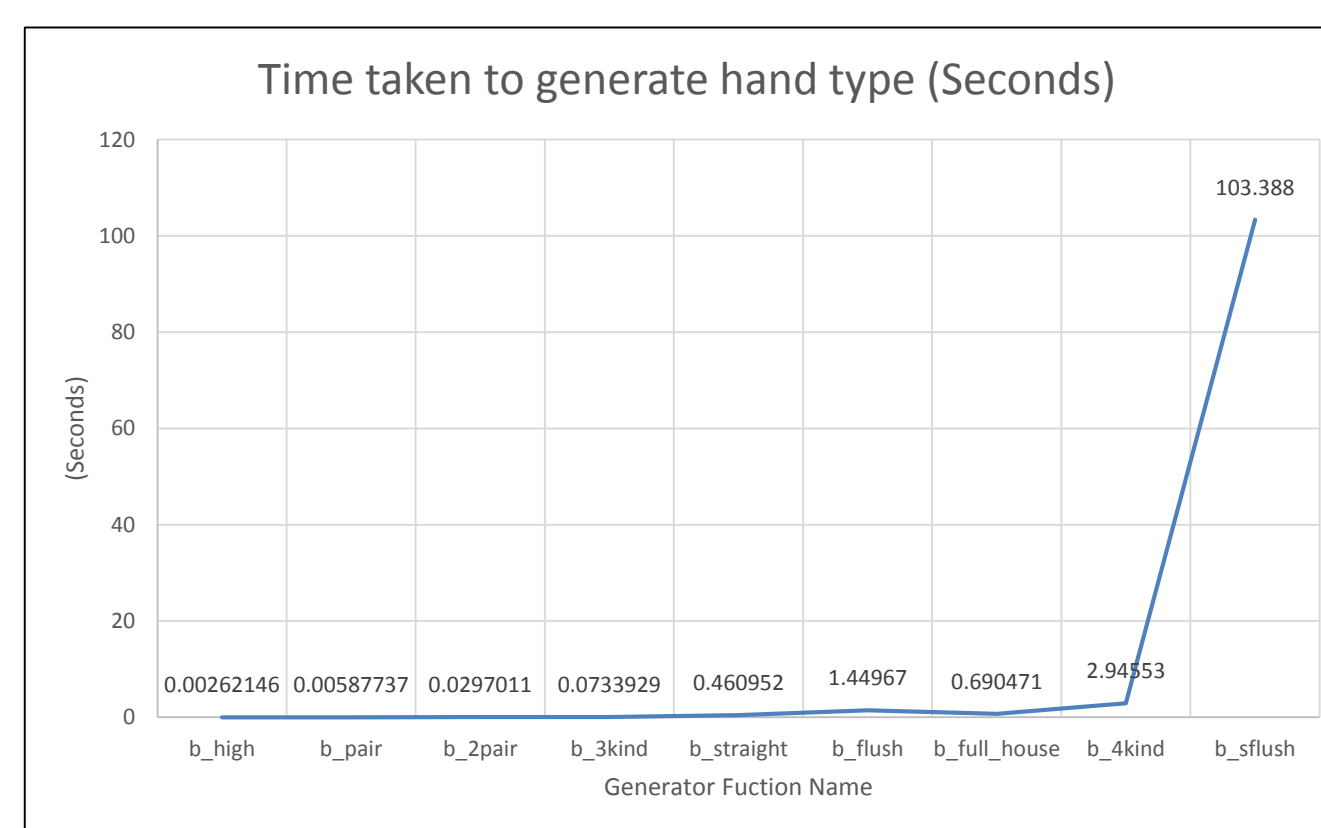


Figure 7 – Initial benchmarking data for hand generation functions in GetHandOfRank().

	Straight Flush (Seconds)	High Card (Seconds)
Largest value in Batch 1	324.2262	0.001387135
Largest value in Batch 2	80.70618	0.001200312
Largest value in Batch 3	60.4747	0.001842314
Largest value in Batch 4	59.34799	0.001567479
Smallest value in Batch 1	0.076061	0.000122569
Smallest value in Batch 2	0.059754	0.000101511
Smallest value in Batch 3	0.131379	0.000100971
Smallest value in Batch 4	0.027454	0.00010961
Average value in Batch 1	16.60548	0.000285814
Average value in Batch 2	11.22539	0.000270206
Average value in Batch 3	11.1925	0.000259257
Average value in Batch 4	9.944892	0.000263304
Largest value in all batches	324.2262	0.001842314
Smallest value in all batches	0.027454	0.000100971
Average value in all batches	12.24207	0.000269645
Sum of Batch 1	4234.397	0.072882622
Sum of Batch 2	2862.475	0.068902641
Sum of Batch 3	2854.088	0.066110553
Sum of Batch 4	2436.499	0.0671424

Figure 8 – Benchmarking summary for 250 sample batches.

Line #	Hits	Time	Per Hit	% Time	Line Contents
14					@profile
15					def parallel():
16	1	8.0	8.0	0.0	jobs = []
17	1	16.0	16.0	0.1	for i in range(4):
18	1	165.0	165.0	0.8	p = multiprocessing.Process(target=worker_sflush)
19	1	6.0	6.0	0.0	jobs.append(p)
20	1	19301.0	19301.0	99.0	p.start()

Figure 9 Line-by-line benchmarks parallelised code proved inconclusive.

Conclusions and Future Work

- Investigation into parallelising the brute force algorithm indicates that it is faster than serial. Adequate benchmarking could not objectively verify (Fig. 9).
- UCI's dataset and data generated from this project should be used to train separate classifiers and their accuracy compared to confirm the initial hypothesis.
- Although inflexible, the rules based classifier which was necessary to develop for the data generator is accurate enough to be relied on as a classifier in its own right.