Group 26: A solver for cryptic crossword puzzles

Introduction

A cryptic crossword is a type of crossword in which each clue is a word puzzle. Each clue typically has one or more words corresponding to a definition of the answer at either the beginning or end of the clue, with the rest of the clue forming a word puzzle which also corresponds to the answer. Cryptic crosswords tend to be exceedingly difficult, with even multiple-time Times Crossword Championship winner Mark Goodliffe taking 22 minutes to solve 3 championship crosswords [1] - the average user can take hours, if they even manage to solve them at all.

For both cryptic crossword regulars and beginners, there is a lack of accessible tools to assist and enhance their crossword experience. When working through a crossword users were not able to solve certain clues, or believed they had the right answer to a clue but did not understand the wordplay. Therefore, our users wanted the ability to know the solution to some clues, as well as their explanations.

Official solutions for crosswords are normally only released a week after the crossword, and even then they lack explanations. There are websites such as `https://www.fifteensquared.net` which have crowd-sourced solutions and explanations, but these are not available for all crosswords and lack a standard format. There are some websites and apps which offer a solution to a single typed-in clue, however, these cannot solve cryptic crossword clues and simply search a database of previously seen clues, without explanations or adaptation to new clues (e.g. `https://crossword-solver.io/`). This can be time consuming if you are typing in lots of clues from the same crossword, does not effectively help the user understand the clues and answers often simply fail to solve the clue.

The app “Crossword Genius” offers an additional ability to solve entire crosswords. However, this is mainly suited to the crosswords already in the app. Image recognition of a physical crossword is very limited. Crossword Genius has a cryptic crossword solving AI, Ross, which is very good at solving clues, but not very good at explaining the answer. Printed or partially completed crosswords are a common use case because people will often get part way through a printed (i.e. in a newspaper) crossword before getting stuck, unfortunately, this app struggles to read the majority of photos and is completely unable to process crosswords which have been partially completed. Additionally, it is only available on iOS and takes a long time (> 5mins) to solve entire crosswords.

In our project, we endeavoured to build an app to help people solve crosswords, capable of solving and explaining full crosswords when given a photo or screenshot of the crossword, even if it is partially completed. We aimed to improve upon the best current solution to our problem, Crossword Genius, by having better explanations, the ability to read partially filled crosswords, a new and better image recognition system, cross-platform support, and faster full grid solving while still allowing individual clue solving.

The Project Artefact

From our supervisor, we were supplied with a GitLab repository for a crossword solver called Morse. We created a branch ‘microprocess’ in the repository and made any necessary additions to the repository there. This included making an API interface to the existing code.

We were also supplied access to the API of an external crossword solver written by Unlikely AI, called Ross. This solver is a black box to us, but we are able to utilise API calls to solve clues.

Our code utilises calls to both of these solvers as they have different strengths and weaknesses. Ross is very good at solving clues with no known letters. Morse is less accurate at solving but when provided with the correct answer it is very reliable at finding parsing the clue and producing an explanation. Combining these two solvers we get the basis for our project.

Main repo: `https://gitlab.doc.ic.ac.uk/g226002126/cryptic-crossword-solver`
Morse Solver repo: `https://gitlab.com/skellystudios/crossword-solver`
Web app: `http://www.wardware.co.uk`

We created a web app which is hosted on the above URL. This domain was already owned by a group member and so could be used with no additional cost. Currently, the domain only hosts on HTTP to avoid mixed content errors with our HTTP API calls. This could be solved with additional funding.

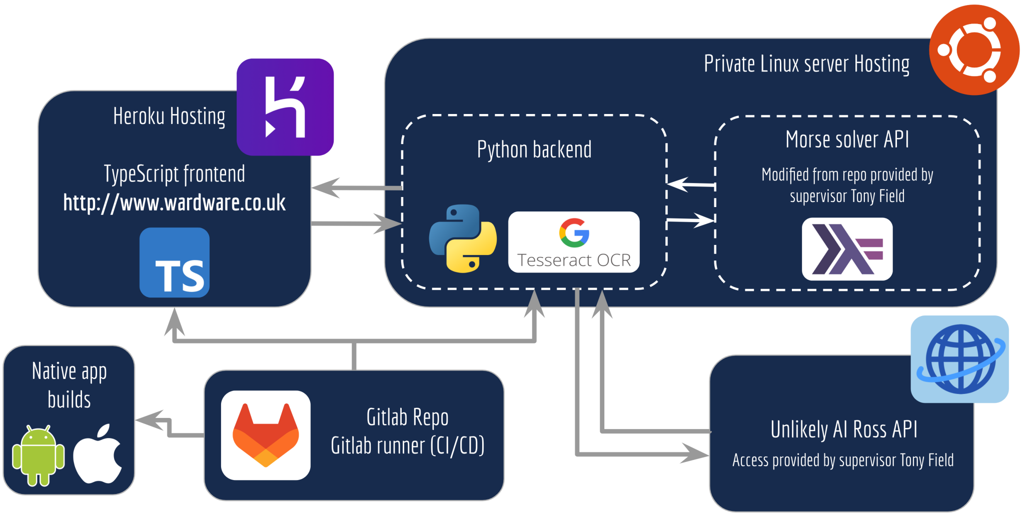


Figure 1: System architecture diagram

We developed a flask-based API that calls the crossword solvers based on input from the front-end application. This is served on a HP EliteDesk 800 G2 SFF PC. This device has 16GB of DDR4 Memory, with an Intel i7-6700 Quad Core Processor. This device is not extremely powerful but is energy efficient enough that it can be left running, as well as being a spare device and as such was good for our needs. It runs Ubuntu 22.10, with Python 3.10.8.

A feature that was kept from the 1st iteration, shown in Figures 2 and 3, is the ability for the user to input a single cryptic crossword clue, length and partial solution for the app to solve. The app outputs the top solutions, the confidence level of each solution and the explanation of each solution. At most 3 solutions are shown.

Alternatively, the user is able to upload a photo from the gallery or take a photo in our app. The selected image is displayed on the screen as shown in Figure 4, or a default example image if no image is selected.

On receiving an image, after preprocessing we find the largest approximately square contour (outline of a shape), assuming this is the crossword grid. We then warp the grid as necessary to produce a flat square and divide it into 15x15 segments. We decide if squares are white or black based on a threshold value for the number of black pixels, and decide if white squares are filled with a written character based on another threshold value applied to a subsection of the square.

For written squares, we use contour hierarchy information to remove any square numbers and borders. The square is then cropped to fit the character with a small padding and rescaled to fit the input size of our neural network.

We then run our TensorFlow neural network handwriting recognition model (trained on: `https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format`) to predict what character is in filled squares. We assume that users fill in entire clues, so stray characters are removed, and clues are corrected using a dictionary.

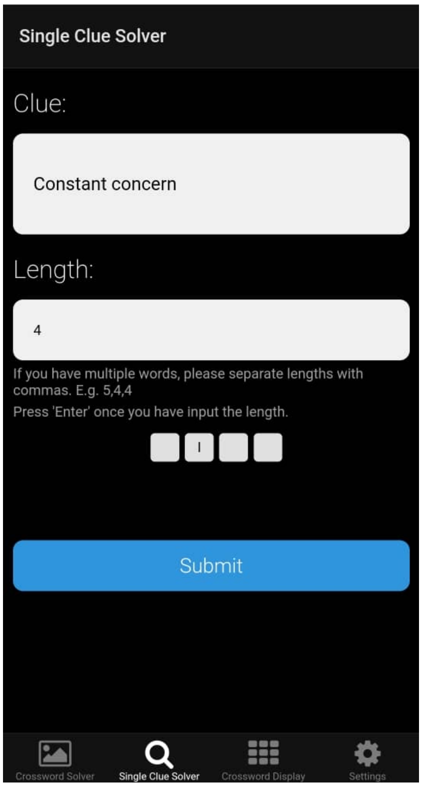
The original image is then passed to the clue-reading system. We use Tesseract OCR to detect the font size of text on the image. We attempt to detect the font size and process the data to remove outliers. If we deem the font size to be reliably detected (standard deviation less than a set value) we use it as a guide to scale our image, which ensures good performance in our image processing pipeline.

We use the detected font size as a guide, and explore nearby sizes to see if they are more optimal. To create a robust computer vision model, we pre-process 7 slightly differently scaled images in parallel. During preprocessing we remove shadows, apply a binary threshold, and remove noise.

The pre-processing pipeline has a fixed kernel size, so differently scaled images will have different levels of readability. We take all 7 of the processed images and use Tesseract OCR to read their text. We try to parse this text as a crossword and pick the best output, which is the crossword with the most clues correctly identified.

We correct any misread words or missed spaces by trying to split words apart and using Enchant's autocorrect to check words are in the dictionary. Our parsing system uses a Levenshtein automata to efficiently find Across/Down tags, which we wrote using the algorithm described in Schulz & Mihov, 2002 [2] and the Apache Lucene source code as references.

The crossword grid display was built upon the GitHub repository `https://github.com/JaredReisinger/react-crossword`. This repository is under the MIT licence so we are free to modify and distribute this code.

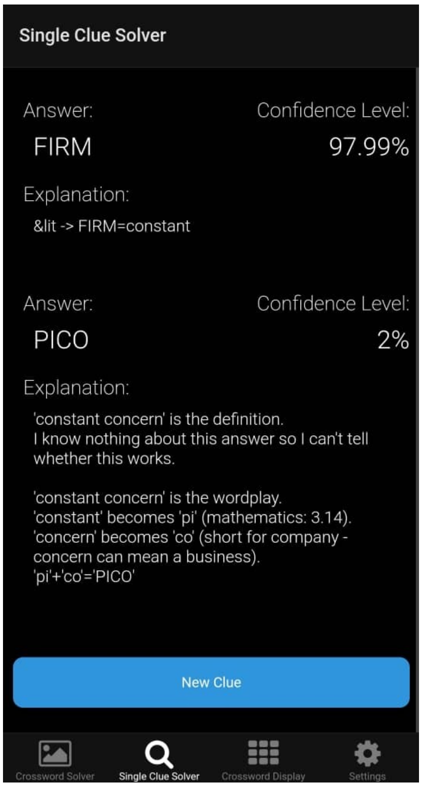


Constant concern

4

If you have multiple words, please separate lengths with commas. E.g. 5,4,4
Press 'Enter' once you have input the length.

Figure 2: Single clue solver page



Answer: FIRM
Confidence Level: 97.99%

Explanation:
&lit -> FIRM=constant

Answer: PICO
Confidence Level: 2%

Explanation:
'constant concern' is the definition.
I know nothing about this answer so I can't tell whether this works.

'constant concern' is the wordplay.
'constant' becomes 'pi' (mathematics: 3.14).
'concern' becomes 'co' (short for company - concern can mean a business).
'pi'+'co'='PICO'

Figure 3: Single clue answer page

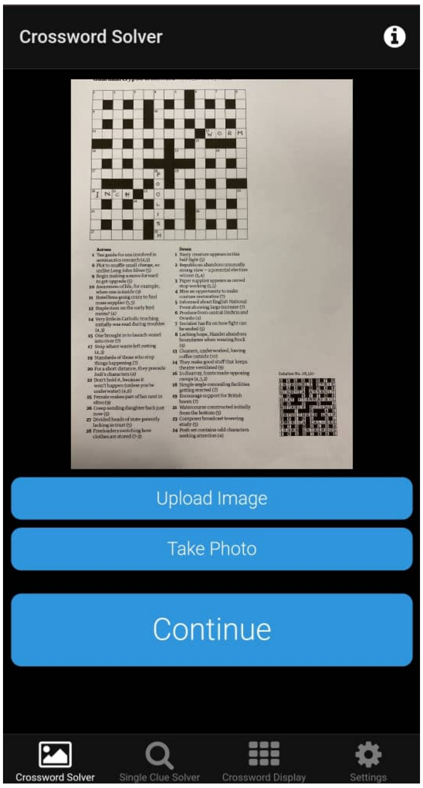
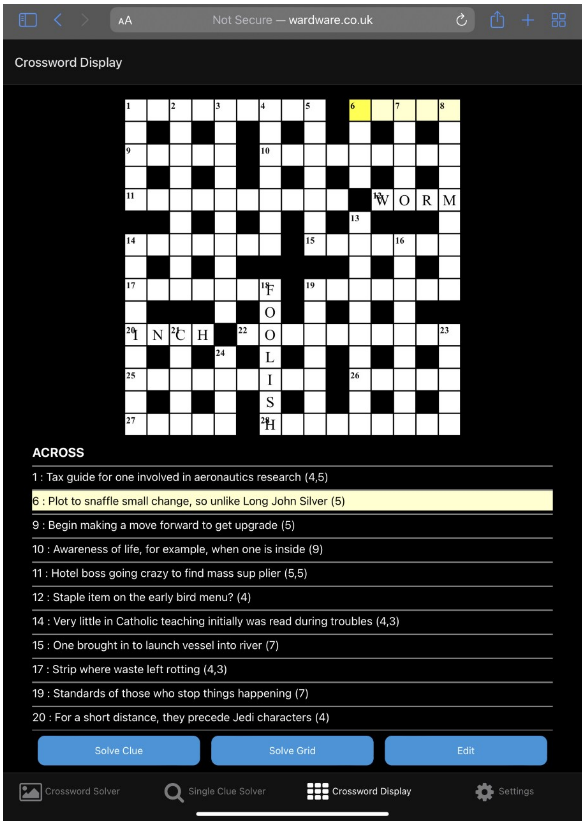


Figure 4: Crossword Input Page

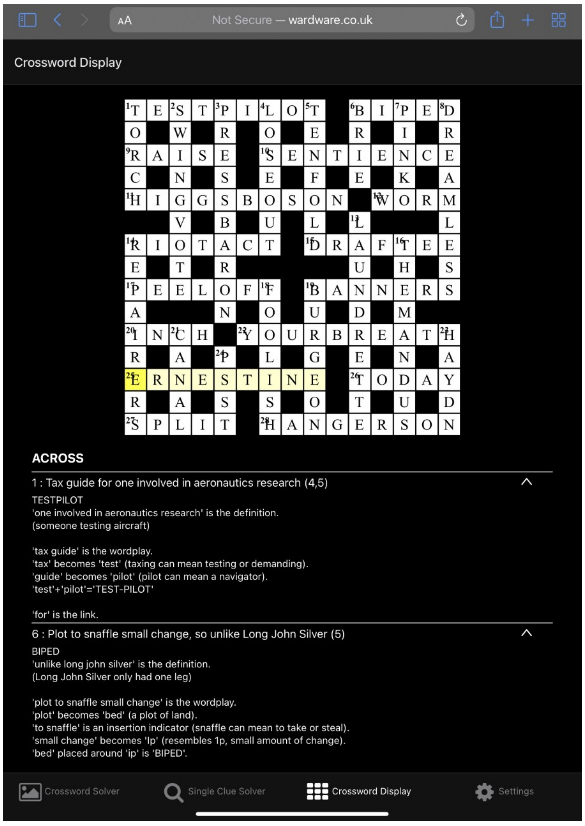


1		2		3		4		5	■	6		7		8
	■		■		■		■		■		■		■	
9					■	10								
	■		■		■		■		■		■		■	
11										■	12 W	O	R	M
■	■		■		■		■		■	13	■	■	■	
14							■	15				16		
	■		■		■	■	■	■	■		■		■	
17						18 F	■	19						
	■	■	■		■	O	■		■		■		■	■
20 I	N	21 C	H	■	22	O								23
	■		■	24	■	L	■		■		■		■	
25						I			■	26				
	■		■		■	S	■		■		■		■	
27					■	28 H								

ACROSS

1 : Tax guide for one involved in aeronautics research (4,5)
6 : Plot to snaffle small change, so unlike Long John Silver (5)
9 : Begin making a move forward to get upgrade (5)
10 : Awareness of life, for example, when one is inside (9)
11 : Hotel boss going crazy to find mass sup plier (5,5)
12 : Staple item on the early bird menu? (4)
14 : Very little in Catholic teaching initially was read during troubles (4,3)
15 : One brought in to launch vessel into river (7)
17 : Strip where waste left rotting (4,3)
19 : Standards of those who stop things happening (7)
20 : For a short distance, they precede Jedi characters (4)

Figure 5: Read Grid



1 T	E	2 S	T	3 P	I	4 L	O	5 T	■	6 B	I	7 P	E	8 D
O	■	W	■	R	■	O	■	E	■	R	■	I	■	R
9 R	A	I	S	E	■	10 S	E	N	T	I	E	N	C	E
C	■	N	■	S	■	E	■	F	■	E	■	K	■	A
11 H	I	G	G	S	B	O	S	O	N	■	12 W	O	R	M
■	■	V	■	B	■	U	■	L	■	13 L	■	■	■	L
14 R	I	O	T	A	C	T	■	15 D	R	A	F	16 T	E	E
E	■	T	■	R	■	■	■	■	■	U	■	H	■	S
17 P	E	E	L	O	F	18 F	■	19 B	A	N	N	E	R	S
A	■	■	■	N	■	O	■	U	■	D	■	M	■	■
20 I	N	21 C	H	■	22 Y	O	U	R	B	R	E	A	T	23 H
R	■	A	■	24 P	■	L	■	G	■	E	■	N	■	A
25 E	R	N	E	S	T	I	N	E	■	26 T	O	D	A	Y
R	■	A	■	S	■	S	■	O	■	T	■	U	■	D
27 S	P	L	I	T	■	28 H	A	N	G	E	R	S	O	N

ACROSS

1 : Tax guide for one involved in aeronautics research (4,5)
TESTPILOT
'one involved in aeronautics research' is the definition.
(someone testing aircraft)

'tax guide' is the wordplay.
'tax' becomes 'test' (taxing can mean testing or demanding).
'guide' becomes 'pilot' (pilot can mean a navigator).
'test'+'pilot'='TEST-PILOT'

'for' is the link.

6 : Plot to snaffle small change, so unlike Long John Silver (5)
BIPED
'unlike long john silver' is the definition.
(Long John Silver only had one leg)

'plot to snaffle small change' is the wordplay.
'plot' becomes 'bed' (a plot of land).
'to snaffle' is an insertion indicator (snaffle can mean to take or steal).
'small change' becomes 'lp' (resembles 1p, small amount of change).
'bed' placed around 'lp' is 'BIPED'.

Figure 6: Solved Grid

Changes are as follows: React to React Native click and touch behaviour consistent across platforms, style changes, jump to clues, added explanations for clues, edit clues and cells, integration of solving and updating displays during solve.

An 'edit' button is provided to correct inaccurately read clues or handwritten letters on the grid. It can also be used for the user to provide their own answers on the grid.

Full Grid Solving One distinct advantage that our app has over Ross, is its ability to leverage the grid when solving an entire crossword. Ross and Morse both solve clues in isolation, without considering the intersections between answers, which could narrow down the range of possibilities.

Thus, we developed a simple and quick method for best-effort full-grid solving. It operates as follows:

1. The clues are placed on the grid, and an intersection map () is generated for mapping a clue to its 'waiters' which will have more information, should the original be solved.
2. A todo list is created from the clues, ordered by their number of waiters (starting important clues early can speed up all their waiters, if solved)
3. A runners map maps clues to an AbortController, allowing their queries to be aborted
4. Now, for each clue in the todo list:
   (a) A new AbortController is added to the runners map, then a query is sent to the backend with this AC, as well as the current contents of the clue's answer cells on the grid.
   (b) Once the query returns, it either:
      i. Fails, due to insufficient confidence (<0.95%) from Ross.
      ii. Succeeds, in which case the answer is written to the grid, and a solve-with-answer query is sent to Morse for explaining the answer. Then, all other runners for the same clue are aborted, since the solution has been found, and the solve method (from 4(a)) is called for each of the waiters for the solved clue.

Compared to more naive fullgrid-solving approaches –namely (a) ignoring the grid, and solving each clue individually, and (b) dispatching queries for every unsolved clue given the current board state and waiting for all of them to return before sending out more– this algorithm showed vast improvements in avg. % of clues solved compared to (a) (44% vs 87% on recent The Guardians), and a huge avg. speedup over (b) (4 minutes with 20:00 mins worst-case observed vs. 1:24 mins and aborted after 2 mins)

All the tools, libraries and resources we used, except Ross API, were used in accordance with their licence and can be used in a commercial application if we chose to publish our Cryptic Crossword Solver in the future. We would need to provide the Apache 2 licence along with the end product.

Evaluation

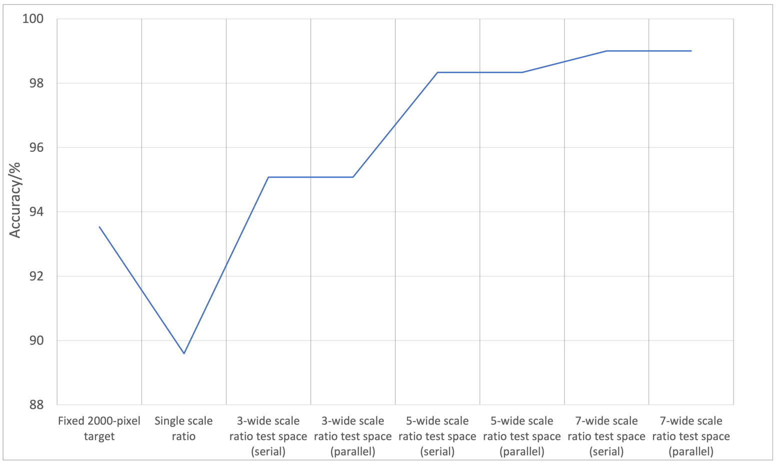


Figure 7: Computer Vision processing time depending on the quality of the image and method used

We believe we have met the user needs specified in the introduction. In particular, our system has improved on the current offerings by providing better explanations with Morse, improving the number of machine-solvable clues by unifying Ross and Morse, and creating a robust and accurate computer vision system to parse crosswords from images with handwritten answers. We believe we have created an app which is easy to use and has functions which fulfil different user needs.

The size of the text on a page is very important to the accuracy of our computer vision pipeline. Too small images have their text eroded away to be unreadable, but too large images do not get cleaned up enough to be readable. We tested different approaches to picking the best scaling ratio for our input image: A fixed image size (2000pi), and a scheme where we read font size and scale to 1/3/5/7 similarly sized images, to allow the rather inaccurate font size detection system to point us in the right general region, and to find the optimal solution

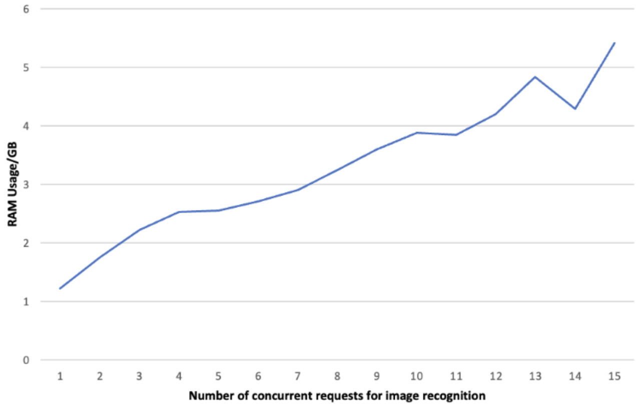


Figure 8: Scalability: Computer Vision RAM use

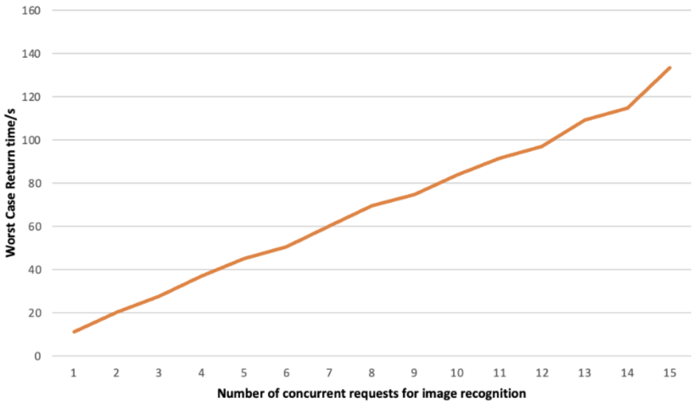


Figure 9: Scalability: Computer Vision Request time

by trial and error. In Figure 7 we compare how accurate (normalised on output length) the results are, while also considering the time taken to return back to the user. We test on 10 different images and average the performance of the methods. We see that using the font detection method improves accuracy, and a wider scale-ratio test space improves accuracy further at the expense of processing time. Our users did not want to correct the read-in crossword manually as they wanted a seamless experience, so we prioritise accuracy over the processing time. Furthermore, parallelising the processing reduces the time penalty to a manageable 10s.

We want our system to serve many users, so we tested the impact of increasing the number of concurrent users on both the independent clue solver and computer vision systems. In Figure 8 and 9 we see that RAM usage and processing time increase linearly on the computer vision task, suggesting even one request saturates the computational capabilities of the machine. To make a future scalable solution we should separate the computer vision and solver servers. We can distribute the computer vision tasks among multiple servers using a consistent hashing scheme as its request independence lends itself well to horizontal scaling. The Morse solver servers, however, can be configured to build a cache that speeds up subsequent similar requests, so investing in vertical scaling (e.g. increasing RAM) for those servers could be more beneficial to increase maximum concurrent users.

We tested the performance of grid character recognition by filling in crosswords with characters and seeing how many characters are correctly predicted out of 456 total character predictions.

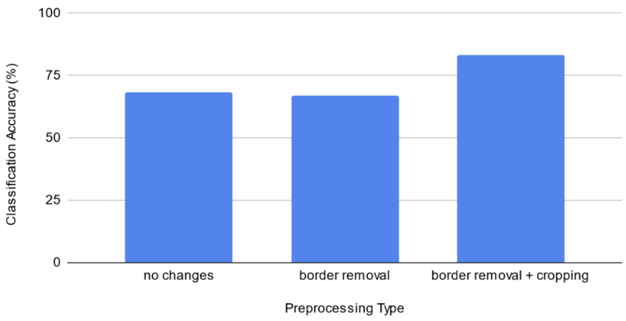


Figure 10: Preprocessing methods effect on accuracy

We started with an average classification accuracy of 68.2%. From our testing, in Figure 10 we see border removal does not affect the classification accuracy much, probably because the neural network has very low weights for pixels on the margin as it does not expect writing there. However, our classification accuracy increases to 83.2% if we utilise both border removal and cropping. While this accuracy isn't that high, our model tends to underperform on letters which are not that common in use, and our dictionary is able to correct words with a few mispredicted letters, resulting in very few misread words.

One weakness of the border removal algorithm is that it requires the letter in the cell to be a single connected stroke. If the letter is segmented for some reason only one segment will be kept, and if it touches a cell number or border that number/border won't be removed.

In the future, we should look into improving the handwriting recognition performance to ensure our users don't need to correct any entries to make sure the experience is as hands-off as possible.

References

[1] Alan Connor. Crossword blog: watching a champion solver at work, 2014. URL `https://www.theguardian.com/crosswords/crossword-blog/2014/oct/20/crossword-blog-watching-a-champion-solver-at-work`. [Accessed on 2023-01-09].

[2] Klaus U. Schulz and Stoyan Mihov. Fast string correction with levenshtein automata. *International Journal on Document Analysis and Recognition*, 5(1):67–85, 2002. doi: 10.1007/s10032-002-0082-8.