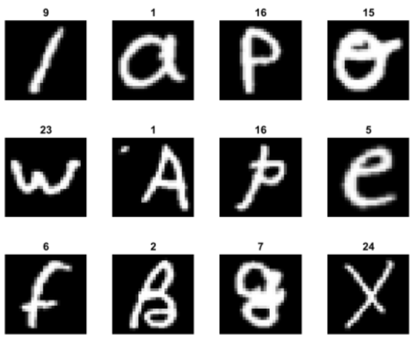
Optical Character Recognition from the handwritten EMNIST character set

Introduction

Optical Character Recognition is the process of converting handwritten text into machine-encoded text through mediums such as documents, written papers, or images. Developing automated methods for this using machine learning and deep learning of large datasets is important for producing AI models that can appropriately recognise and convert this text. In this work, we aim to compare varying classification models for their efficiency and accuracy in creating an AI model for Optical Character Recognition. This has many uses such as data entry in transcribing written records into a digital form.

Data and Preparation



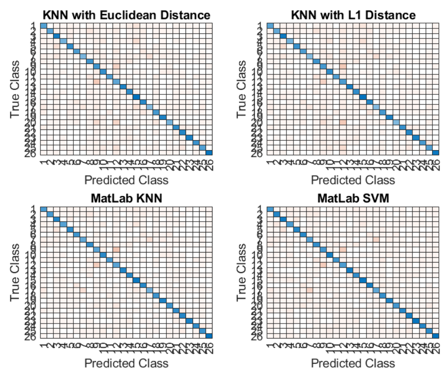
The EMNIST dataset is a collection of 26,000 images. The format of this data is 26,000 28x28 pixel images with each pixel represented by a numerical value. Each image is also provided with a corresponding label as to the correct digital character that the handwritten image refers to. Data is provided as a 1x784 vector and variable labels stored as a column vector. We convert the images into a double type so that it can be scaled accurately unlike an integer. We also convert the labels into a categorical array for further processing. The data is split into training and testing subsets to avoid overfitting. A random permutation of the 26,000 images is created and used to make a 50/50 split of training and testing subsets.

Methodology

In this study we compare various classification models and distance measures to classify the test data. We implement the K-nearest neighbour classification model. This model measures the distance from our test point to the nearest points. It then looks at the k-nearest of these and sets the value of the test point to the most common labelled point nearby. We use the k-value of 28 as a compromise for efficiency and accuracy. In this method we compare the impact of using Euclidean distance against using L1 or Manhattan distance. We then also compare two models from MatLab using pre-defined functions for K-nearest neighbour and SVM for Multiclass. SVM for Multiclass works by splitting data into linearly separated groups.

Results

Classification Model	Time of Processing (s)	Correct Predicts(/13000)	Accuracy (%)
KNN Custom Euclidean	424	9496	73.05
KNN Custom Manhattan	423	9202	70.78
KNN MatLab	21.5	10096	77.66
SVM MatLab	41.3	9518	73.22



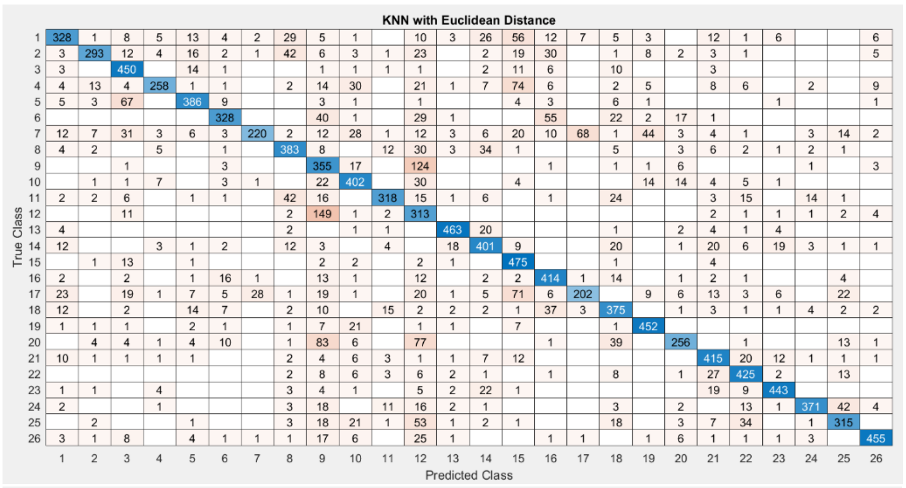
The Custom KNN model is far slower to process than the built-in functions which is to be expected as pre-built functions are optimised in machine code already without the need for compilation or interpretation. The distance measures have very similar time of processing. Manhattan is quicker as it doesn't need calculating of square numbers however this is a large dataset, and this advantage doesn't apply. The MatLab KNN is ~20 times quicker and has a greater accuracy making it more efficient and better. SVM has a slower processing time as it classifies the entire data which takes a lot of time on the dataset. This is also not rewarded as the accuracy is less. SVM is often used for text-processing and is efficient on small datasets, so this observation is expected on a large dataset.

Conclusion

In conclusion, our study shows that the K-nearest neighbour classification model is both the most efficient and accurate algorithm for correctly classifying these handwritten images. KNN is very good for implementing on multiclass models however it can be slow on large datasets and struggle with imbalanced data. An accuracy of 77.66% is not good enough for making this OCR model viable in commercial use and so further testing and training of other models is important for increasing these numbers. Further evaluation could also be done to recognise areas of weakness. For example, looking at the confusion charts we see the models have issues with distinguishing between the letter I and L.

KNN with Euclidean Distance:

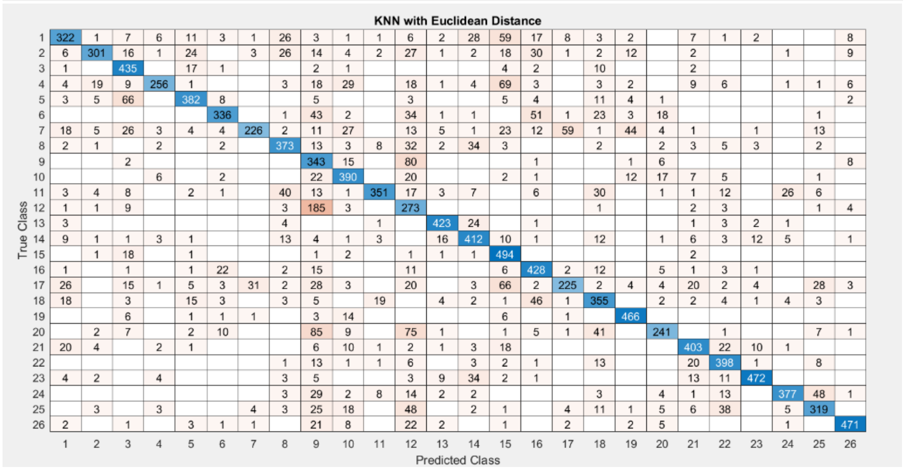
Test 1: 435s, 9496/13000, 73.05%   Test 2: 413s, 9472/13000, 72.86%



KNN with Euclidean Distance

True Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	328	1	8	5	13	4	2	29	5	1		10	3	26	56	12	7	5	3		12	1	6			6
2	3	293	12	4	16	2	1	42	6	3	1	23		2	19	30		1	8	2	3	1				5
3	3		450		14	1			1	1	1	1		2	11	6		10			3					
4	4	13	4	258	1	1		2	14	30		21	1	7	74	6		2	5		8	6		2		9
5	5	3	67		386	9			3	1		1			4	3		6	1				1			1
6						328			40	1		29	1			55		22	2	17	1					
7	12	7	31	3	6	3	220	2	12	28	1	12	3	6	20	10	68	1	44	3	4	1		3	14	2
8	4	2		5		1		383	8		12	30	3	34	1			5		3	6	2	1	2	1	
9			1			3			355	17		124				1		1	1	6				1		3
10		1	1	7		3	1		22	402		30			4				14	14	4	5	1			
11	2	2	6		1	1		42	16		318	15	1	6		1		24			3	15		14	1	
12			11					2	149	1	2	313									2	1	1	1	2	4
13	4							2		1	1		463	20				1		2	4	1	4			
14	12			3	1	2		12	3		4		18	401	9			20		1	20	6	19	3	1	1
15		1	13		1				2	2		2	1		475			1			4					
16	2		2		1	16	1		13	1		12		2	2	414	1	14		1	2	1			4	
17	23		19	1	7	5	28	1	19	1		20	1	5	71	6	202		9	6	13	3	6		22	
18	12		2		14	7		2	10		15	2	2	2	1	37	3	375		1	3	1	1	4	2	2
19	1	1	1		2	1		1	7	21		1	1		7			1	452							
20		4	4	1	4	10		1	83	6		77				1		39		256		1			13	1
21	10	1	1	1	1			2	4	6	3	1	1	7	12						415	20	12	1	1	1
22								2	8	6	3	6	2	1		1		8		1	27	425	2		13	
23	1	1		4				3	4	1		5	2	22	1						19	9	443			
24	2			1				3	18		11	16	2	1				3		2		13	1	371	42	4
25		2			1			3	18	21	1	53	1	2	1			18		3	7	34		1	315	
26	3	1	8		4	1	1	1	17	6		25	1			1	1		1	6	1	1	1	3		455

Predicted Class



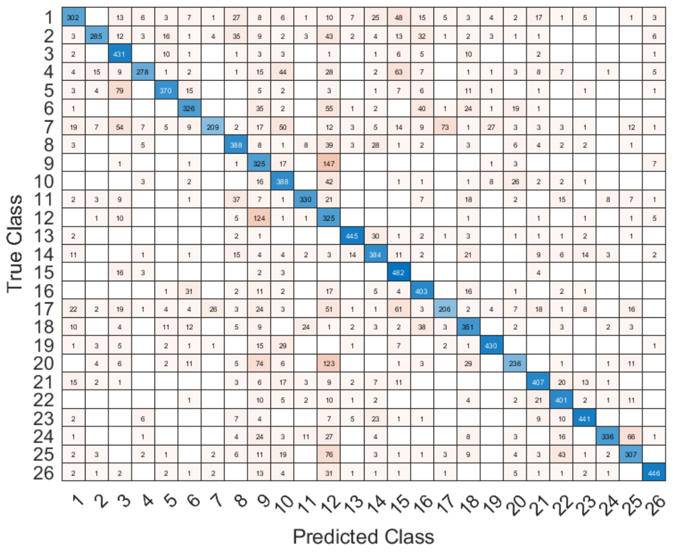
KNN with Euclidean Distance

True Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	322	1	7	6	11	3	1	26	3	1	1	6	2	28	59	17	8	3	2		7	1	2			8
2	6	301	16	1	24		3	26	14	4	2	27	1	2	18	30	1	2	12		2			1		9
3	1		435		17	1			2	1					4	2		10			2					
4	4	19	9	256	1			3	18	29		18	1	4	69	3		3	2		9	6		1	1	6
5	3	5	66		382	8			5			3			5	4		11	4	1						2
6						336		1	43	2		34	1	1		51	1	23	3	18					1	
7	18	5	26	3	4	4	226	2	11	27		13	5	1	23	12	59	1	44	4	1		1		13	
8	2	1		2		2		373	13	3	8	32	2	34	3			2		2	3	5	3		2	
9			2						343	15		80				1			1	6						8
10				6		2			22	390		20			2	1			12	17	7	5			1	
11	3	4	8		2	1		40	13	1	351	17	3	7		6		30		1	1	12		26	6	
12	1	1	9					3	185	3		273						1			2	3			1	4
13	3							4			1		423	24		1					1	3	2	1		
14	9	1	1	3	1			13	4	1	3		16	412	10	1		12		1	6	3	12	5		1
15		1	18		1				1	2		1	1	1	494						2					
16	1		1		1	22		2	15			11			6	428	2	12		5	1	3	1			
17	26		15	1	5	3	31	2	28	3		20		3	66	2	225	2	4	4	20	2	4		28	3
18	18		3		15	3		3	5		19		4	2	1	46	1	355		2	2	4	1	4	3	
19			6		1	1	1		3	14					6		1		466							
20		2	7		2	10			85	9		75	1		1	5	1	41		241		1			7	1
21	20	4		2	1				6	10	1	2	1	3	18						403	22	10	1		
22								1	13	1	1	6		3	2	1		13			20	398	1		8	
23	4	2		4				3	5			3	9	34	2	1					13	11	472			
24								3	29	2	8	14	2	2				3		4	1	13		377	48	1
25		3		3			4	3	25	18		48		2	1		4	11	1	5	6	38		5	319	
26	2		1		3	1	1		21	8		22	2		1		2		2	5				1		471

Predicted Class

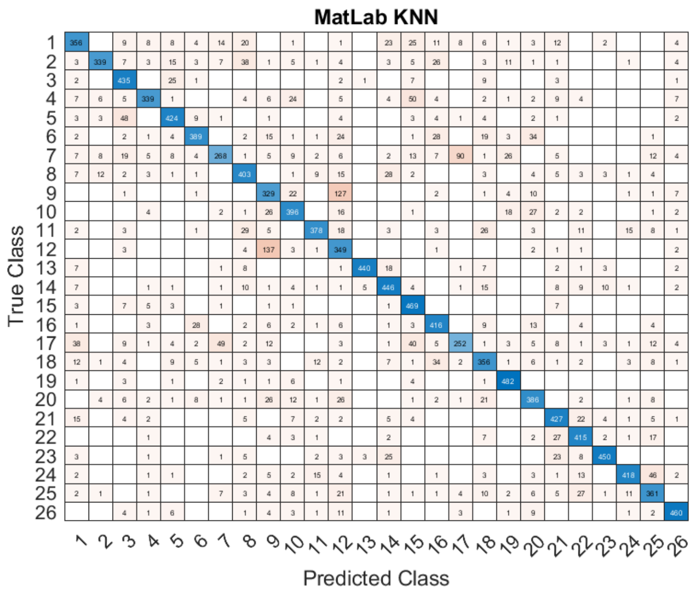
KNN with Manhattan Distance:

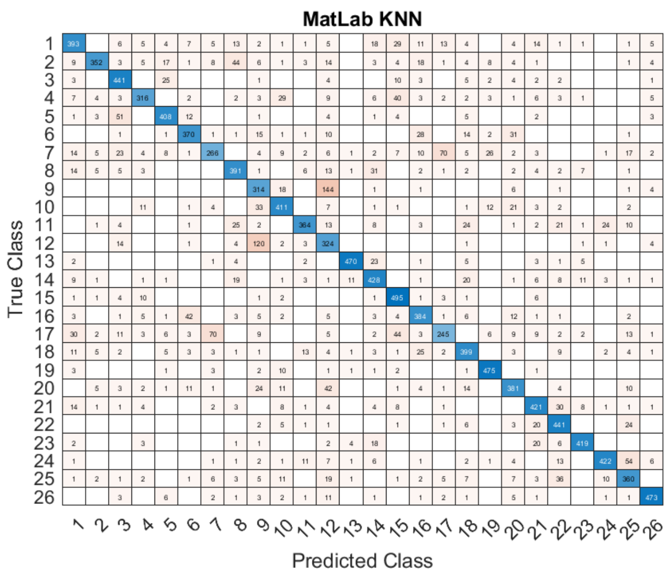
Test 1: 420s, 9202/13000, 70.78%   Test 2: 426s, 9232/13000, 71.02%



<u>KNN MatLab:</u>

Test 1: 21.5s, 10096/13000, 77.66%   Test 2: 21.05s, 10163/13000, 78.18%

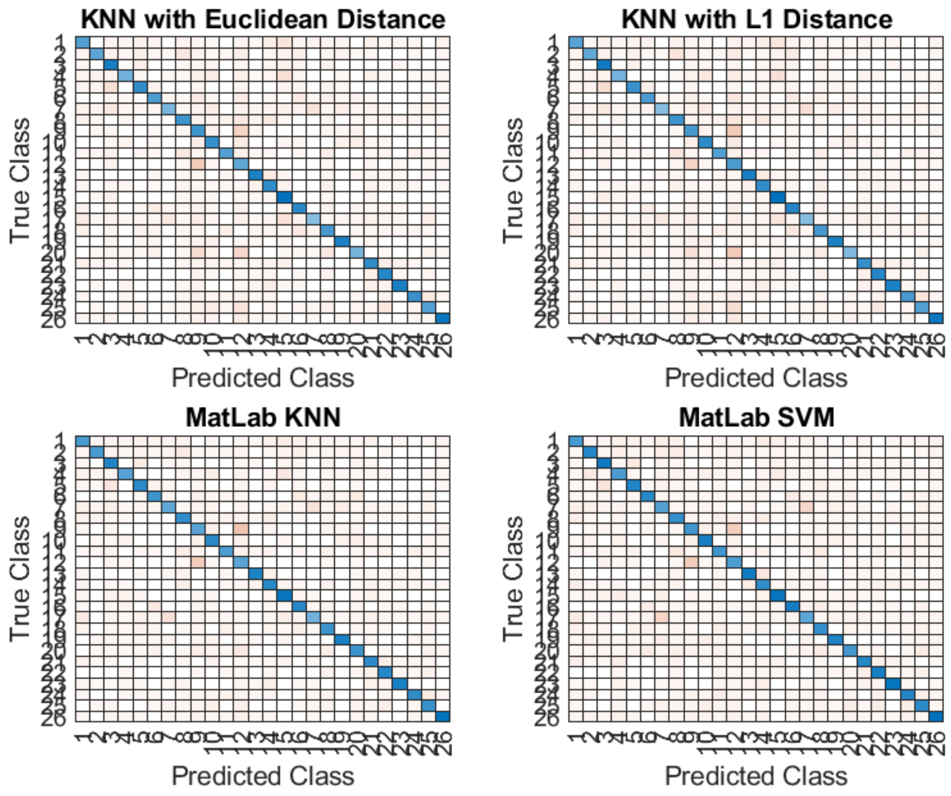




<u>KNN with SVM:</u>

Test 1: 41.3s, 9518/13000, 73.22%   Test 2: 38.9s, 9543/13000, 73.41%

<u>Full Results:</u>


KNN with Euclidean Distance
KNN with L1 Distance
MatLab KNN
MatLab SVM
True Class
Predicted Class
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26