License Plate Character Classification

Karenna Rodriguez and Kori Vernon

Professor Sellie

Machine Learning Final Project

Fall 2021

Table of Contents

Introduction

Background

The goal of this project is to implement character and digit recognition on license plates and test how well we can identify license plate characters with different methods of Machine Learning: Logistic Regression, Support Vector Machines (SVM), and Neural Networks.

Datasets and Data Pre-Processing

We used three separate datasets to train our neural network to recognize License Plate characters.

Figure 1: dataset_chars

1) The dataset_chars dataset was collected from a license plate recognition project. The dataset contains characters 37,623 examples that were sized 28x28 of different fonts. No preprocessing on the dataset_chars dataset was performed.

2) We collected a number of license plates (test_lps) ourselves by taking pictures of passer-by license plates. Some photographs from a license plate recognition project were also used as test license plates. With regard to preprocessing, we imported a module to aid in the segmentation of individual characters. After the individual characters were segmented, they were size 60x30. For the Logistic Regression and Support Vector Machine models, the individual License Plate Characters were resized to 28x28.

Figure 2: test_lps

3) For the Neural Network, we used both the MNIST digits sklearn dataset and dataset_chars. MNIST digits consist of 8x8 pixel images of digits 0-9. In order to use MNIST digits to train our model, and test_lps to test our model, we resized our segmented license plate character images to be 8x8 rather than 60x30.

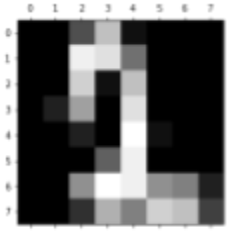


Figure 3: MNIST digit

Figure 4: License Plate Character transformed with Padding and Resizing

For our project, we implemented various data transformations including resizing, RGB to greyscale, padding our images, and using sklearn's StandardScaler. As mentioned previously, the license plate images were resized to 28x28 for Logistic Regression and SVMs and resized to 8x8 for Neural Networks in order to match the training datasets. We also used padding to make the images in test_lps look more like the characters in dataset_chars. This transformation can be seen in Figure 4. The greyscale transformation removes the depth of an image and made it easier for us to reshape the arrays of images in order to use them in our models. Sklearn's StandardScaler transforms features by removing the mean and scaling to unit variance.

Supervised Learning - Logistic Regression

I. *Logistic Regression using dataset_chars*

No Penalty, C = 1.0
Score: 0.8843019135364989
Classification Report:

	precision	recall	f1-score	support
0	0.75	0.74	0.75	166
1	0.85	0.91	0.87	169
2	0.96	0.95	0.95	148
3	0.94	0.92	0.93	161
4	0.92	0.96	0.94	143
5	0.94	0.89	0.92	167
6	0.93	0.89	0.91	154
7	0.96	0.92	0.94	145
8	0.83	0.88	0.85	158
9	0.85	0.90	0.87	163
A	0.89	0.90	0.89	155
B	0.88	0.87	0.87	162
C	0.92	0.93	0.92	141
D	0.92	0.94	0.93	188
E	0.89	0.88	0.89	154
F	0.84	0.86	0.85	139
G	0.90	0.87	0.88	170
H	0.86	0.85	0.85	153
I	0.81	0.79	0.80	157
J	0.82	0.92	0.87	142
K	0.95	0.96	0.96	151
L	0.94	0.92	0.93	165
M	0.91	0.88	0.89	176
N	0.87	0.87	0.87	149
O	0.73	0.75	0.74	138
P	0.92	0.91	0.92	185
Q	0.89	0.87	0.88	154
R	0.89	0.85	0.87	157
S	0.89	0.82	0.85	144
T	0.89	0.91	0.90	126
U	0.90	0.86	0.88	176
V	0.85	0.88	0.86	149
W	0.90	0.85	0.88	155
X	0.88	0.92	0.90	179
Y	0.91	0.92	0.91	168
Z	0.86	0.90	0.88	137
accuracy			0.88	5644
macro avg	0.88	0.88	0.88	5644
weighted avg	0.89	0.88	0.88	5644

L2, C = 0.001
Score: 0.8871367824238129
Classification Report:

	precision	recall	f1-score	support
0	0.75	0.78	0.76	166
1	0.86	0.90	0.88	169
2	0.95	0.95	0.95	148
3	0.93	0.92	0.92	161
4	0.94	0.95	0.94	143
5	0.94	0.90	0.92	167
6	0.93	0.90	0.92	154
7	0.97	0.93	0.95	145
8	0.82	0.89	0.85	158
9	0.86	0.91	0.88	163
A	0.90	0.90	0.90	155
B	0.88	0.85	0.87	162
C	0.90	0.93	0.92	141
D	0.91	0.94	0.92	188
E	0.88	0.88	0.88	154
F	0.82	0.87	0.85	139
G	0.90	0.86	0.88	170
H	0.87	0.84	0.85	153
I	0.80	0.82	0.81	157
J	0.83	0.92	0.87	142
K	0.95	0.95	0.95	151
L	0.94	0.92	0.93	165
M	0.92	0.89	0.90	176
N	0.88	0.87	0.87	149
O	0.76	0.74	0.75	138
P	0.92	0.92	0.92	185
Q	0.89	0.87	0.88	154
R	0.91	0.87	0.89	157
S	0.88	0.82	0.85	144
T	0.88	0.91	0.90	126
U	0.92	0.86	0.89	176
V	0.86	0.88	0.87	149
W	0.91	0.85	0.88	155
X	0.88	0.93	0.90	179
Y	0.91	0.92	0.91	168
Z	0.87	0.90	0.88	137
accuracy			0.89	5644
macro avg	0.89	0.89	0.89	5644
weighted avg	0.89	0.89	0.89	5644

II. *Logistic Regression using License Plates*

With Image Padding

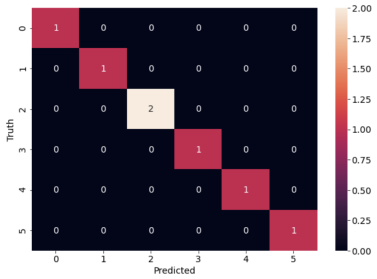
L2, C = 10

Score: 1.0

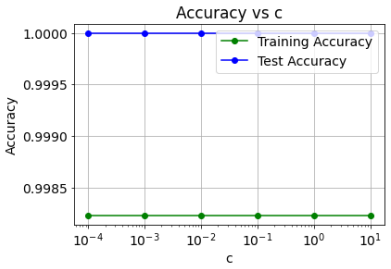
Classification Report

	precision	recall	f1-score	support
3	1.00	1.00	1.00	1
5	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	1
M	1.00	1.00	1.00	1
P	1.00	1.00	1.00	1
accuracy			1.00	7
macro avg	1.00	1.00	1.00	7
weighted avg	1.00	1.00	1.00	7

Confusion Matrix



Accuracy vs. C



Actual

Predicted

```
['M' '8' '7' '5' '7' '3' 'P']
```

All of the license plate characters were correctly classified.

Without Image Padding

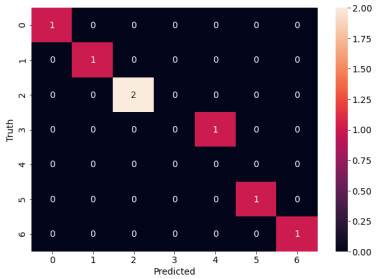
L2, C = 10

Score: 0.857

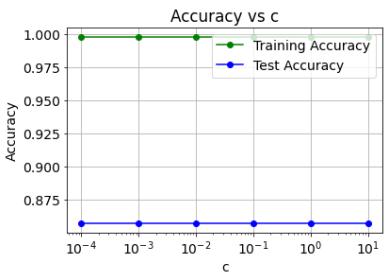
Classification Report

	precision	recall	f1-score	support
3	1.00	1.00	1.00	1
5	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	0
M	1.00	1.00	1.00	1
P	1.00	1.00	1.00	1
accuracy			0.86	7
macro avg	0.71	0.71	0.71	7
weighted avg	0.86	0.86	0.86	7

Confusion Matrix



Accuracy vs. C



Actual

Predicted

```
['M' '9' '7' '5' '7' '3' 'P']
```

One of the license plate characters was incorrectly classified; ‘8’ was classified as a ‘9’ without padding.

Supervised Learning - Support Vector Machines

III. *Support Vector Machines using dataset_chars*

RBF Kernel

C = 1, gamma=0.0001

Score: 0.2680960396916807

	precision	recall	f1-score	support
0	0.96	0.19	0.32	339
1	1.00	0.23	0.37	300
2	0.04	1.00	0.07	311
3	1.00	0.24	0.38	314
4	1.00	0.25	0.40	292
5	1.00	0.21	0.35	343
6	1.00	0.21	0.35	323
7	1.00	0.24	0.39	318
8	1.00	0.21	0.35	340
9	1.00	0.22	0.36	330
A	1.00	0.26	0.41	316
B	1.00	0.25	0.40	324
C	1.00	0.24	0.38	306
D	1.00	0.26	0.41	351
E	1.00	0.25	0.40	302
F	0.99	0.33	0.49	279
G	1.00	0.22	0.36	337
H	1.00	0.27	0.42	331
I	1.00	0.33	0.50	309
J	1.00	0.25	0.40	308
K	1.00	0.25	0.41	291
L	1.00	0.30	0.47	312
M	1.00	0.28	0.43	320
N	1.00	0.25	0.40	306
O	0.96	0.23	0.37	279
P	1.00	0.24	0.39	327
Q	1.00	0.23	0.38	312
R	1.00	0.24	0.38	316
S	1.00	0.23	0.37	304
T	1.00	0.29	0.45	303
U	1.00	0.27	0.42	319
V	1.00	0.26	0.41	280
W	1.00	0.24	0.38	297
X	0.99	0.22	0.36	333
Y	1.00	0.23	0.38	323
Z	1.00	0.28	0.44	292
accuracy			0.27	11287
macro avg	0.97	0.27	0.39	11287
weighted avg	0.97	0.27	0.39	11287

Linear Kernel

C = 1, gamma=0.0001

Score: 0.9181359085673784

	precision	recall	f1-score	support
0	0.72	0.80	0.76	339
1	0.86	0.90	0.88	300
2	0.95	0.97	0.96	311
3	0.91	0.97	0.94	314
4	0.95	0.97	0.96	292
5	0.93	0.94	0.93	343
6	0.94	0.91	0.92	323
7	0.96	0.95	0.96	318
8	0.93	0.90	0.91	340
9	0.94	0.94	0.94	330
A	0.94	0.92	0.93	316
B	0.89	0.90	0.90	324
C	0.96	0.94	0.95	306
D	0.93	0.96	0.94	351
E	0.95	0.94	0.94	302
F	0.92	0.95	0.93	279
G	0.93	0.92	0.93	337
H	0.90	0.91	0.91	331
I	0.79	0.87	0.83	309
J	0.91	0.93	0.92	308
K	0.94	0.97	0.96	291
L	0.96	0.94	0.95	312
M	0.93	0.95	0.94	320
N	0.92	0.92	0.92	306
O	0.72	0.72	0.72	279
P	0.95	0.95	0.95	327
Q	0.94	0.89	0.92	312
R	0.96	0.89	0.92	316
S	0.92	0.91	0.91	304
T	0.95	0.91	0.93	303
U	0.92	0.88	0.90	319
V	0.91	0.91	0.91	280
W	0.98	0.92	0.95	297
X	0.94	0.96	0.95	333
Y	0.97	0.89	0.93	323
Z	0.95	0.94	0.94	292
accuracy			0.92	11287
macro avg	0.92	0.92	0.92	11287
weighted avg	0.92	0.92	0.92	11287

Polynomial Kernel
C = 1, gamma=0.0001
Score: 0.8786214228758749

	precision	recall	f1-score	support
0	0.68	0.81	0.74	339
1	0.80	0.87	0.84	300
2	0.94	0.95	0.94	311
3	0.89	0.91	0.90	314
4	0.93	0.92	0.93	292
5	0.93	0.90	0.91	343
6	0.96	0.87	0.91	323
7	0.91	0.93	0.92	318
8	0.88	0.88	0.88	340
9	0.92	0.89	0.91	330
A	0.93	0.90	0.91	316
B	0.80	0.86	0.83	324
C	0.93	0.92	0.92	306
D	0.96	0.94	0.95	351
E	0.94	0.88	0.91	302
F	0.74	0.94	0.83	279
G	0.78	0.91	0.84	337
H	0.84	0.90	0.87	331
I	0.53	0.82	0.64	309
J	0.91	0.86	0.88	308
K	0.94	0.92	0.93	291
L	0.97	0.84	0.90	312
M	0.96	0.91	0.93	320
N	0.94	0.86	0.90	306
O	0.71	0.77	0.74	279
P	0.95	0.88	0.91	327
Q	0.87	0.78	0.82	312
R	0.93	0.85	0.89	316
S	0.95	0.87	0.91	304
T	0.96	0.81	0.88	303
U	0.96	0.87	0.91	319
V	0.95	0.86	0.90	280
W	0.91	0.90	0.90	297
X	0.94	0.91	0.93	333
Y	0.97	0.82	0.89	323
Z	0.91	0.92	0.91	292
accuracy			0.88	11287
macro avg	0.89	0.88	0.88	11287
weighted avg	0.89	0.88	0.88	11287

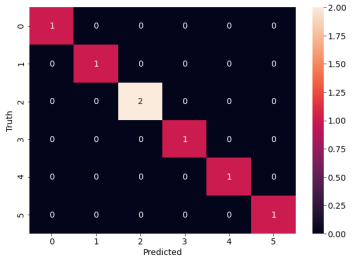
IV. *Support Vector Machines using License Plates*

Linear Kernel
With Image Padding, C = 1, gamma=0.0001
Score: 1.0

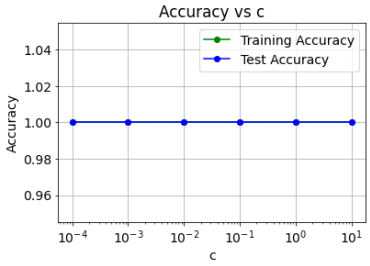
Classification Report

	precision	recall	f1-score	support
3	1.00	1.00	1.00	1
5	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	1
M	1.00	1.00	1.00	1
P	1.00	1.00	1.00	1
accuracy			1.00	7
macro avg	1.00	1.00	1.00	7
weighted avg	1.00	1.00	1.00	7

Confusion Matrix



Accuracy vs. C



Actual

Predicted

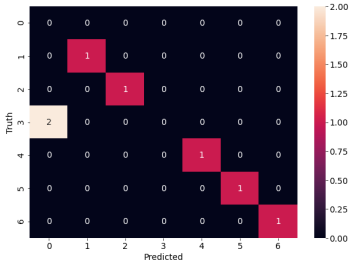
```
['M' '8' '7' '5' '7' '3' 'P']
```

Linear Kernel
Without Image Padding, C = 1, gamma=0.0001
Score: 0.7142857142857143

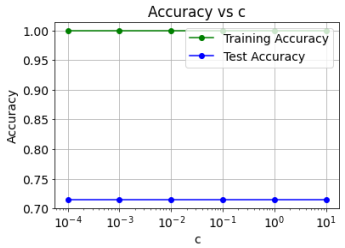
Classification Report

	precision	recall	f1-score	support
2	0.00	0.00	0.00	0
3	1.00	1.00	1.00	1
5	1.00	1.00	1.00	1
7	0.00	0.00	0.00	2
8	1.00	1.00	1.00	1
M	1.00	1.00	1.00	1
P	1.00	1.00	1.00	1
accuracy			0.71	7
macro avg	0.71	0.71	0.71	7
weighted avg	0.71	0.71	0.71	7

Confusion Matrix



Accuracy vs. C



Actual

Predicted

```
['M' '8' '2' '5' '2' '3' 'P']
```

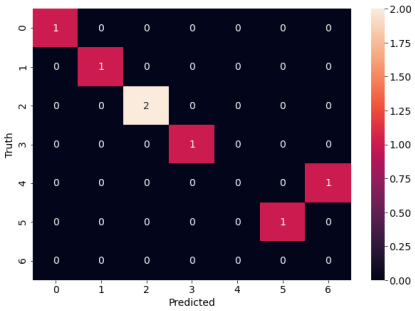
Both instances of ‘7’ were classified as a ‘2.’

Polynomial Kernel
With Image Padding, C = 1, gamma=0.0001
Score: 0.8571428571428571

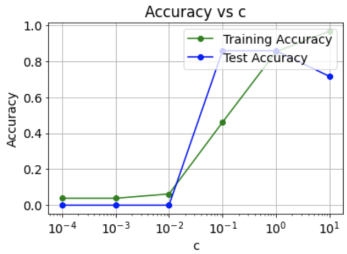
Classification Report

	precision	recall	f1-score	support
3	1.00	1.00	1.00	1
5	1.00	1.00	1.00	1
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	1
M	0.00	0.00	0.00	1
P	1.00	1.00	1.00	1
R	0.00	0.00	0.00	0
accuracy			0.86	7
macro avg	0.71	0.71	0.71	7
weighted avg	0.86	0.86	0.86	7

Confusion Matrix



Accuracy vs. C



Actual

Predicted

```
['R' '8' '7' '5' '7' '3' 'P']
```

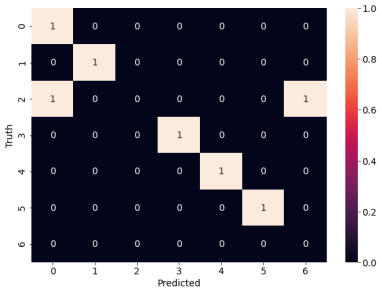
The letter ‘M’ was predicted as the letter ‘R.’

Polynomial Kernel
Without Image Padding, C = 1, gamma=0.0001
Score: 0.7142857142857143

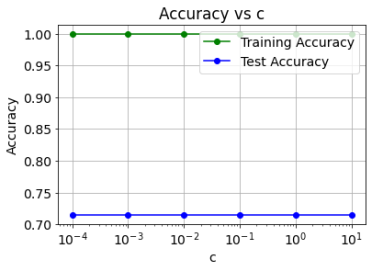
Classification Report

	precision	recall	f1-score	support
3	0.50	1.00	0.67	1
5	1.00	1.00	1.00	1
7	0.00	0.00	0.00	2
8	1.00	1.00	1.00	1
M	1.00	1.00	1.00	1
P	1.00	1.00	1.00	1
R	0.00	0.00	0.00	0
accuracy			0.71	7
macro avg	0.64	0.71	0.67	7
weighted avg	0.64	0.71	0.67	7

Confusion Matrix



Accuracy vs. C



Actual

Predicted

```
['M' '8' 'R' '5' '3' '3' 'P']
```

'7' was classified as '3' and 'R.'

Supervised Learning - Neural Networks

Neural Network Using dataset_chars

Using Sigmoid Activation Function
With Image Padding,
Neural Network Structure = (64, 40, 36),
alpha = 0.25,
iter_num = 3000
Score: 0.04920429250141201

Classification Report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	870
1	0.00	0.00	0.00	836
2	0.00	0.00	0.00	884
3	0.00	0.00	0.00	820
4	0.00	0.00	0.00	815
5	0.00	0.00	0.00	870
6	0.00	0.00	0.00	866
7	0.00	0.00	0.00	825
8	0.00	0.00	0.00	851
9	0.01	0.01	0.01	841
A	0.00	0.00	0.00	850
B	0.00	0.00	0.00	859
C	0.00	0.00	0.00	835
D	0.00	0.00	0.00	875
E	0.06	0.00	0.00	830
F	0.00	0.00	0.00	823
G	0.00	0.00	0.00	843
H	0.05	0.03	0.04	862
I	0.00	0.00	0.00	799
J	0.00	0.00	0.00	801
K	0.97	0.29	0.45	854
L	0.00	0.00	0.00	843
M	0.01	0.01	0.01	827
N	0.00	0.00	0.00	818
O	0.08	0.73	0.14	800
P	0.00	0.00	0.00	850
Q	0.00	0.00	0.00	822
R	1.00	0.00	0.00	835
S	0.00	0.00	0.00	817
T	0.00	0.00	0.00	824
U	0.00	0.00	0.00	834
V	0.03	0.75	0.06	787
W	0.00	0.00	0.00	827
X	0.00	0.00	0.00	828
Y	0.00	0.00	0.00	837
Z	0.00	0.00	0.00	841
accuracy			0.05	30099
macro avg	0.06	0.05	0.02	30099
weighted avg	0.06	0.05	0.02	30099

The neural network using dataset_chars performed significantly worse (and slower) than MNIST. Only 6 characters were correctly classified.

Neural Network using MNIST

Using Sigmoid Activation Function

With Image Padding,
Neural Network Structure = (64, 40, 36),
alpha = 0.25,
iter_num = 3000
Score: 0.9151599443671766

Classification Report

	precision	recall	f1-score	support
0	0.97	0.99	0.98	74
1	0.88	0.82	0.85	78
2	0.96	0.92	0.94	72
3	0.95	0.92	0.94	64
4	0.97	0.90	0.93	63
5	0.89	0.97	0.93	73
6	0.99	0.94	0.96	77
7	0.91	0.97	0.94	66
8	0.76	0.89	0.82	76
9	0.94	0.84	0.89	76
accuracy			0.92	719
macro avg	0.92	0.92	0.92	719
weighted avg	0.92	0.92	0.92	719

Neural Network using License Plates (MNIST)

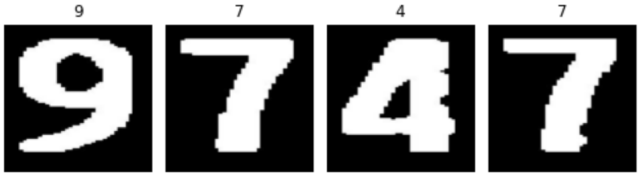
Using Sigmoid Activation Function

With Image Padding, Neural Network Structure = (64, 40, 36), alpha = 0.25, iter_num = 3000
Score: 0.25

Classification Report

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	0
2.0	0.00	0.00	0.00	0
4.0	1.00	1.00	1.00	1
7.0	0.00	0.00	0.00	2
8.0	0.00	0.00	0.00	0
9.0	0.00	0.00	0.00	1
accuracy			0.25	4
macro avg	0.17	0.17	0.17	4
weighted avg	0.25	0.25	0.25	4

Actual



Predicted

```
[0. 2. 4. 8.]
```

The 9 was misclassified as a ‘9’, the ‘2’ misclassified as a ‘7’, however, the ‘4’ was correctly classified, yielding a score of 0.25 on a license plate with only numbers.

Analysis and Conclusion

Logistic Regression

With regard to using Logistic Regression on dataset_chars, we achieved an accuracy score of 88.4% without penalty and using a C-value of 1.0. This resulted in a weighted average accuracy of 89% for 5644 test examples. We added L2 Regularization and decreased the C-value to 0.001. This resulted in an average accuracy of 88.7%. This is slightly better

than if we had no penalty at all. In order to identify the license plate photos we have collected using dataset_chars, we first transformed our original cropped images of the license plate characters. We added padding to the sides of the characters and we found our model actually scored better than it did without padding. Without padding, using L2 regularization, a C-value of 0.001, we had an accuracy of 85.7%; when adding padding, that accuracy rose to 1.0 (100%). Another interesting observation is that when we increased our training size, the accuracy on the license plate decreased. We found that using <20% of our total examples yielded the best results when applying the model to our license plate photos. This suggests that overfitting is an issue and there is high bias and low variance; if we have too many training examples, it makes it harder for our model to predict license plate characters. We also found that decreasing the C-value and increasing the number of training examples, causes the model to misclassify '8' as a '9', 'B', or 'R'.

Support Vector Machines

With regard to using Support Vector Machines on dataset_chars, we achieved an accuracy score of 26.8% in the RBF Kernel, using a C-value of 1, and gamma of 0.0001. Using the Linear Kernel, a C-value of 1, and gamma of 0.0001, a score of 91.8% was achieved. This score is comparatively better than the score using L2 regularization in Logistic Regression. With regard to predictions and accuracy, this model could be the better model to use to predict characters on license plates. Additionally, the testing and training accuracy for the Linear Kernel was the same meaning there is low bias, high variance, and no overfitting. Finally, with regard to using the Polynomial Kernel on dataset_chars, a score of 87.86% was achieved, using a C-value of 1, and a gamma of 0.0001.

In the application of the models to our license plate data, we found that using padding to transform our images greatly improved our accuracy. In the Linear Kernel with image padding, a C-value of 1, and gamma of 0.0001, the model achieved a score of 100% accuracy on the test license plate; however, without image padding and the same C-value and gamma, the model achieved a score of 71.4%. We decided to omit the RBF Kernel model as its test accuracy was too low to deem its application to license plates. Another interesting observation is the way the linear and polynomial kernel misclassified images without padding. For the same license plate image with no padding, the linear kernel misclassified both instances of the number '7' as '2,' while the polynomial kernel misclassified '7' as both 'R' and '3.'

Neural Network

Initially, we used the dataset_chars dataset in order to train our model. In order to make the model more efficient and decrease the run-time of the neural network, a transformation was applied to resize the license plate characters (with padding) from a size of 28x28 to a size of 8x8. In order to further decrease the run-time of the neural network, we only used 20% of examples to train. This resulted in a testing accuracy of 4.9%. This is extremely low, so we resorted to using the handwritten digit MNIST dataset to see if we could achieve a higher accuracy. When using the handwritten MNIST dataset, the neural network model achieved an accuracy of 91.5%. When applied to the test license plate characters, the model correctly predicted 25% of license plate characters.

Conclusion

While several models successfully identified characters with 100% accuracy after the transformation of the license plate photos by adding padding, the model that performed the best was the Support Vector Machine using the Linear Kernel, a C-value of 1, and a gamma of 0.0001 with a test accuracy of 91.8%. Transforming the data by adding padding made a notable improvement to the model accuracy. Future improvements could be made when implementing a more accurate neural network model, however, due to limited computing power, it is more feasible to go with the Support Vector Machine model as an ideal model for License Plate Character Classification.

Credits

1. dataset_characters (dataset) and Character Segmentation: https://github.com/quangnhat185/Plate_detect_and_recognize
2. mnit sklearn dataset: sklearn library
3. Professor Linda Sellie Slides

GitHub Repository: https://github.com/Karennarodriguez/license_plate_detection