Arabic Handwritten Characters Classification Using Logistic Regression, SVM, and Neural Networks

Henrikas (Henry) Krukauskas
New York University Shanghai
New York, USA
henry.krukauskas@nyu.edu

Almadi Shiryayev
New York University Shanghai
New York, USA
as12936@nyu.edu

Abstract

The Arabic Letter dataset (with 16,800 32x32 RGB images) was used in this research project. The dataset was separated into training set (13,440 images) and testing set (3,360 images). Three machine learning algorithms, namely Support Vector Machine (SVM), Logistic Regression and Convolutional Neural Network (CNN) were trained on the data. Linear, radial basis function (RBF) and sigmoid function kernel were used when training the data with SVMs, with the penalty parameter, C, of the error term varying over a range 0.0001 - 100. While training our CNN, different values of alpha (in ReLU) and activation functions were used to find the most accurate model. In addition, increase in the number of hidden layers resulted in very little change in accuracy. As the result, the Convolutional Neural Network (CNN) (with L2 reg. term = 0.001 and 4 hidden layers) was found to produce the best results with a classification accuracy of 94.73%, a slightly poorer accuracy of 75.83% with a radial basis function (RBF) (with C = 100), and the least accurate was the Logistic Regression with accuracy of 41.85%. Index Terms—Logistic, Neural Networks, Convolution, SVM.

1 Introduction

The Arabic handwritten characters dataset was chosen for this research project, which consists of 16,800 32x32 black-and-white images that were obtained from 60 participants, age 19-40, and 90% of participants are right-handed. Images are classified in 28 classes, where each class represents an Arabic letter that the image with handwritten data is capturing. Handwritten text recognition is a typical machine learning algorithm problem for classification.

The images were converted into data tables or flattened vectors. Hence, various models (such as logistic regression, SVM, NN, or CNN) can easily be trained on this dataset. The dataset has a large amount of information, and applicability to prototype different models provided incentive for us to choose this type of data. Moreover, many researchers and startups are concentrating on digitalizing many handwritten documents in order to create more structured and usable data. Therefore, handwritten Arabic letter recognition is an important and valid question to look at, especially, if we think about digitalization of documents that are handwritten in Arabic.

For the purposes of this research, we choose three machine learning algorithms that are the Support Vector Machine (SVM), the Logistic Regression, and the Convolutional Neural Network (CNN). These algorithms are actively used in classification problems, thus, we decided to test our data on these models. For comparison, we are going to look at the accuracy metric for each unique model.

The dataset consists of training set (13,440 images) and testing set (3,360 images). However, we decided to normalize data by dividing every value by 255, so that the data would change to be values in range from 0 to 1.

2 Classification Experiments

For the first set of classification experiments, the SVM model was tested with different kernels and regularization constants. We imported .csv files that had already pre-flattened vectors. Thus, we decided to use provided vectors for our SVM models.

2.1 SVM Kernels and Regularization

We used three types of kernels to train the data, which are linear kernel, the radial basis function (RBF) kernel and the sigmoid function kernel. For the purposes of our research, we decided to use training and testing sets provided in the dataset. 13,440 images were used to train the data. The

Kernel	C	Train Accuracy	Test Accuracy
Linear	0.0001	22.54%	21.48%
	0.001	33.83%	32.26%
	0.01	52.55%	46.48%
	0.1	67.95%	49.94%
	1	85.04%	45.89%
	10	96.14%	43.15%
	100	98.99%	42.32%
RBF	0.0001	36.33%	33.03%
	0.001	36.33%	33.03%
	0.01	36.33%	33.03%
	0.1	45.71%	40.92%
	1	86.18%	65.29%
	10	99.73%	72.23%
	100	100%	72.53%
Sigmoid	0.0001	24.71%	23.81%
	0.001	24.71%	23.81%
	0.01	24.76%	23.81%
	0.1	35.27%	33.03%
	1	34.76%	32.44%
	10	26.57%	23.33%
	100	25.79%	21.25%

Table 1: Test Accuracies for Different SVM Models

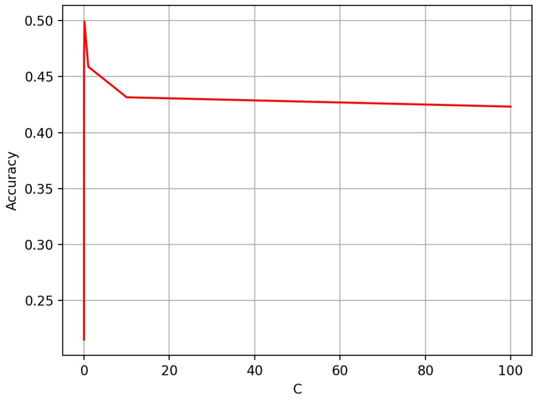

Figure 1: Accuracy against C for SVM Linear Kernel

accuracy was acquired from testing set, where we adjusted our penalty parameter, C, for each run. The penalty parameter, C, of the error term was varied over the range from 0.0001 to 100. The results are summarized in Table 1.

In Table 1, the best test accuracy of 72.53% was obtained in RBF kernel for C = 100. The second best test accuracy was obtained in Linear kernel with value of 49.94%, with a penalty parameter C = 0.1. The plot of accuracy against C for the linear kernel is shown in Figure 1. The performance of the SVM on the test set improves substantially until C = 0.1, after which the performance diminishes.

The RBF kernel was tested next, with the penalty parameter, C, ranging from 0.0001 to 100. In this case, the best test accuracy of 72.53% was obtained for C = 100. As it is our last C value tested, we don't know if beyond C = 100 the accuracy values would decrease. However, from the accuracy values that we have, we can see that at C=10 and beyond, the slope is converging to 0, which might mean that beyond the C=100, the test accuracy might start diminishing. The results are plotted in Figure 2.

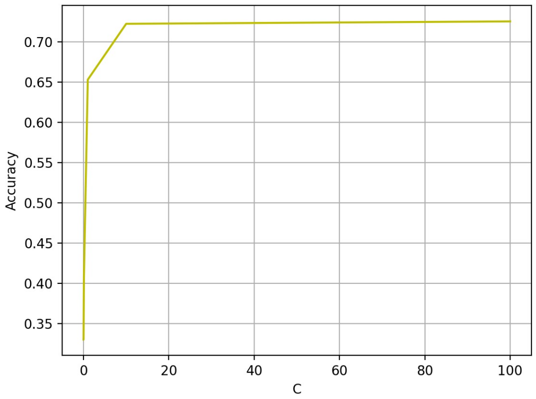

Figure 2: Accuracy against C for SVM RBF Kernel

Thereafter, the Sigmoid kernel was tested, with the penalty parameter, C, ranging from 0.0001 to 1000. In this case, the best test accuracy of 33.04% was obtained for C = 0.1. After C = 0.1, test accuracy started decreasing, which means that the model is overfitting. The results are plotted in Figure 3.

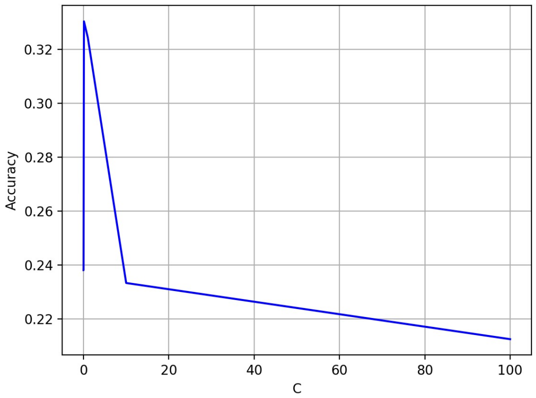

Figure 3: Accuracy against C for SVM Sigmoid Kernel

Selected SVM Model The best results were obtained with the RBF kernel SVM for C = 100 with test accuracy of 72.53%. The confusion matrix of the best result is shown in the Figure 4 below.

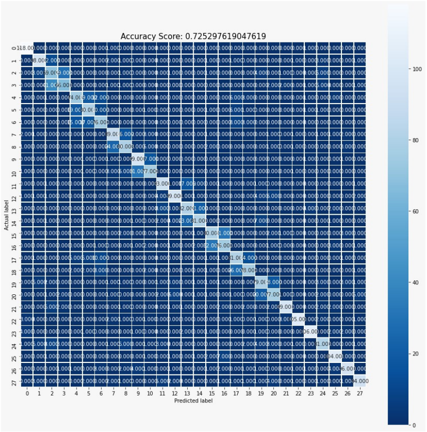

Figure 4: Confusion Matrix of the Best Accuracy Model

2.2 Logistic Regression with Regularization and Feature Transformation

For logistic regression, we used flattened vectors from .csv files. The data was transformed with feature transformation of degree 2 and feature transformation of degree 3, that could be used for different logistic regression model with feature transformations. Once again, we used preprocessed and split data similarly as for SVM models. 13,440 images were used to train the data. The accuracy was acquired from testing set, where we adjusted our penalty parameter, C, for each run. The penalty parameter, C, of the error term was varied over the range from 0.0001 to 100. The results are summarized in Table 2.

Transform.	$\mathcal{C}$	Train Accuracy	Test Accuracy
None	0.0001	28.96%	26.64%
	0.001	34.64%	31.55%
	0.01	44.85%	39.52%
	0.1	53.94%	41.85%
	1	60.19%	39.29%
	10	61.41%	36.67%
	100	61.29%	35.77%
Squared	0.0001	24.81%	25.92%
	0.001	29.94%	29.58%
	0.01	40.62%	36.10%
	0.1	47.62%	38.36%
	1	51.70%	35.86%
	10	51.91%	34.29%
	100	51.44%	33.33%
Cubed	0.0001	21.29%	25.71%
	0.001	25.31%	28.75%
	0.01	35.43%	33.96%
	0.1	41.55%	36.13%
	1	44.44%	34.50%
	10	44.45%	33.18%
	100	44.31%	32.32%

Table 2: Test Accuracies for Different LR Models

In Table 2, the best test accuracy of 41.85% was obtained in no transformation model for $\mathcal{C}$ = 0.1. From transformed data models, the best test accuracies were 38.36% for squared data transformation with $\mathcal{C}$ = 0.1 and 36.13% for cubed data transformation with same $\mathcal{C}$ value.

The plot of accuracy against $\mathcal{C}$ for the no transformation model is shown in Figure 5. The performance of the logistic regression model on the test set improves substantially until $\mathcal{C}$ = 0.1, after which the performance diminishes.

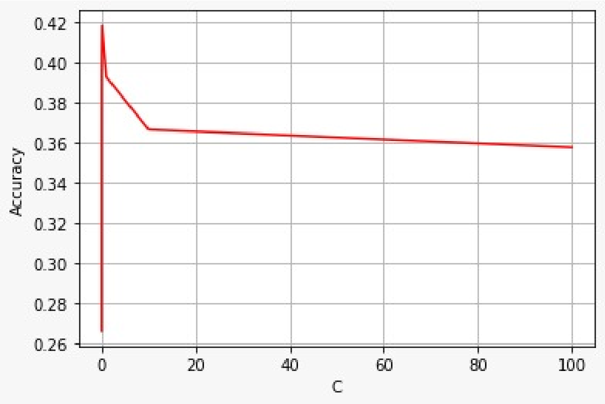

Figure 5: Accuracy against $\mathcal{C}$ for Simple LR Kernel

The plot of accuracy against $\mathcal{C}$ for the squared transformation model is shown in Figure 6. The performance of the logistic regression model on the test set improves substantially until $\mathcal{C}$ = 0.1, after which the performance diminishes.

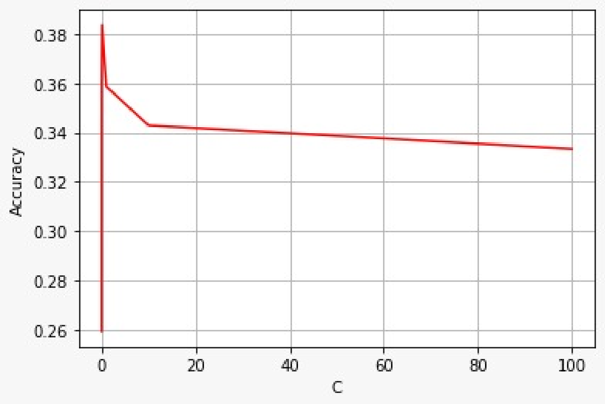

Figure 6: Accuracy against $\mathcal{C}$ for Squared LR Kernel

The plot of accuracy against $\mathcal{C}$ for the cubed transformation model is shown in Figure 7. The performance of the logistic regression model on the test set improves substantially until $\mathcal{C}$= 0.1, after which the performance diminishes.

Selected Logistic Regression Model The best results were obtained with the no data transformation model for C = 0.1 with test accuracy of 41.85%.

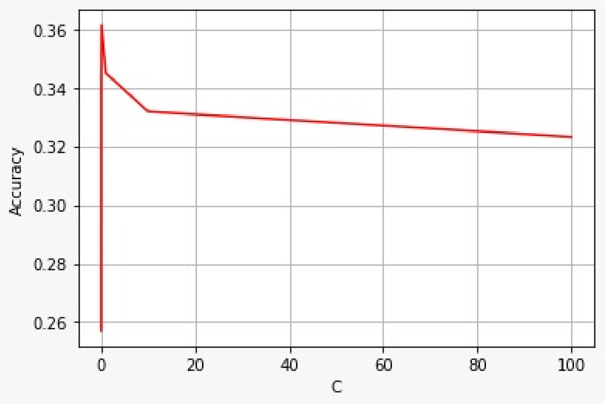

Figure 7: Accuracy against C for Cubed LR Kernel

The confusion matrix of the best result is shown in the Figure 8 below.

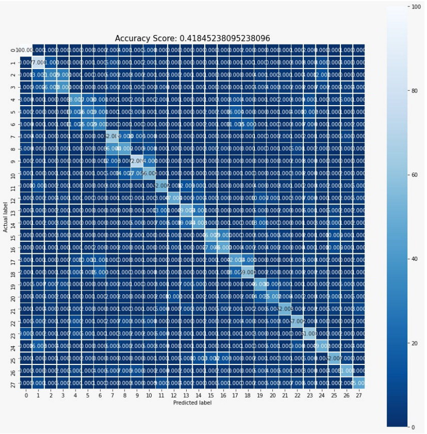

Figure 8: Confusion Matrix of the Best Accuracy Model

2.3 Neural Networks

Before training our CNN models, from our training set, we split the data to create validation set to produce more accurate results for our CNN models. We used four types of activation functions in our CNN to train the data such as Linear, Leaky ReLU, Sigmoid and Tanh. Each time we trained our model 5 times (5 epochs).

Each of the graph above shows the change in the training and validation accuracy after each cycle of training (after each epoch). Every activation function, except for the sigmoid, showed accurate results. The best results were obtained by using Tanh activation function with a 91.63% of validation accuracy. Additionally, we analyzed how the alpha term affects the model's accuracy when using Leaky ReLU activation function. The results are in Table 4.

Activation	Epochs	Train Accuracy	Validation Accuracy
Linear	1	0.5392	0.7682
	2	0.8324	0.8516
	3	0.8943	0.8679
	4	0.9259	0.8802
	5	0.9449	0.8917
Leaky ReLU	1	0.4531	0.7050
	2	0.7875	0.8170
	3	0.8751	0.8624
	4	0.9090	0.8943
	5	0.9369	0.8906
Sigmoid	1	0.0353	0.0387
	2	0.0350	0.0346
	3	0.0354	0.0320
	4	0.0354	0.0387
	5	0.0338	0.0342
Tanh	1	0.5299	0.7816
	2	0.8373	0.8631
	3	0.9055	0.8962
	4	0.9450	0.8999
	5	0.9614	0.9163

Table 3: Validation Accuracies for Different CNN Models

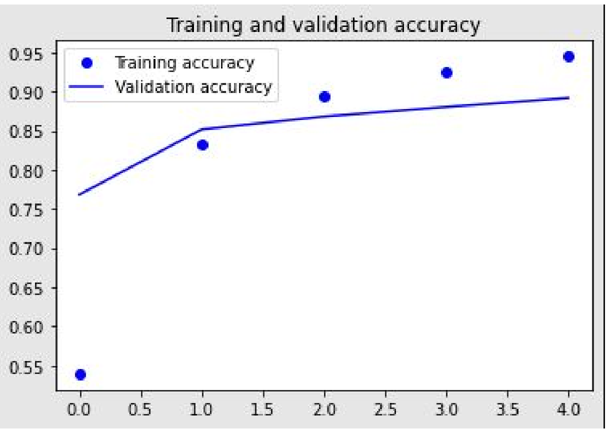

Figure 9: Accuracy Results for CNN with Linear Activation Function

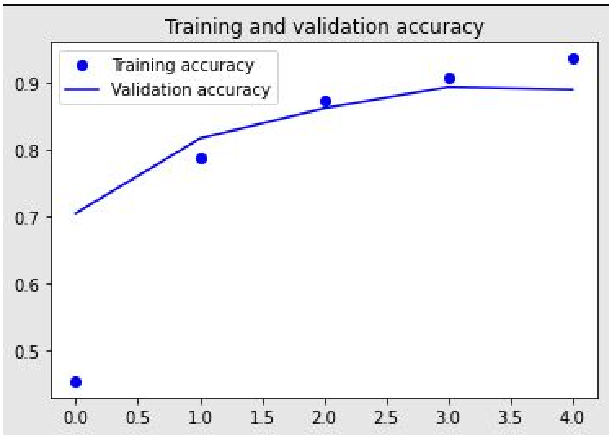

Figure 10: Accuracy Results for CNN with ReLU Activation Function

Each time we trained our model 5 times (5 epochs). The best accuracy when using the Leaky

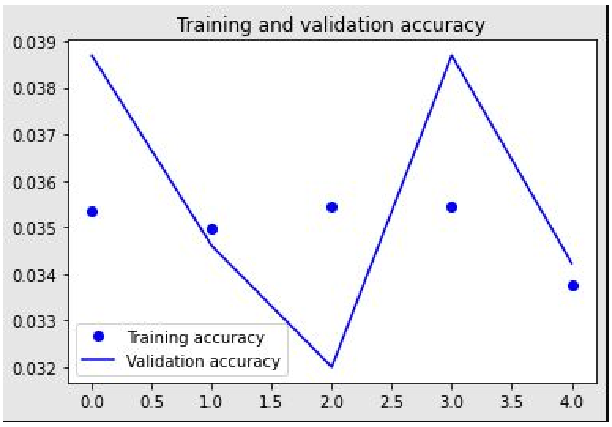

Figure 11: Accuracy Results for CNN with Sigmoid Activation Function

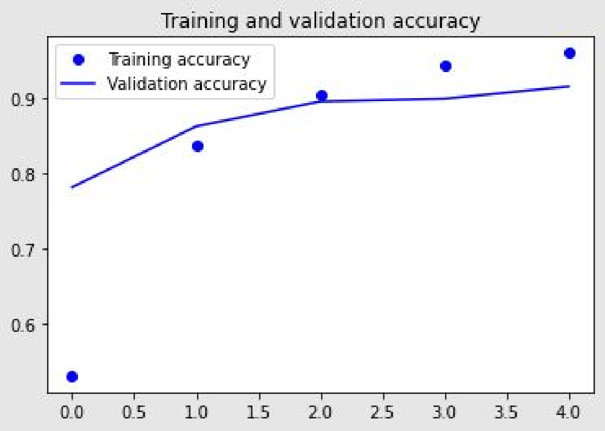

Figure 12: Accuracy Results for CNN with Tanh Activation Function

Activation	Alpha Value	Val. Accuracy
Leaky ReLU	0.0001	0.8754
	0.001	0.9044
	0.1	0.9182
	1	0.8969
	10	0.7589
	100	0.2556

Table 4: Validation Accuracies for Different Alpha Values

ReLU is obtained, when value of alpha is equal to 0.1. When alpha equals to 0.1, our model's test accuracy is 91.82%, which is a great result.

In addition, with each type of activation function, we trained our models with L2 Regularization term. The results are in Table 5.

The data in the Table 5 shows that using the Tanh activation function with L2 regularization term, which equals to 0.0001, gives us the highest validation accuracy.

Activation	L2 Reg.term	Train Accuracy	Val. Accuracy
Linear	0.0001	0.9502	0.9010
	0.001	0.9358	0.9077
	0.1	0.7706	0.7719
	1	0.5836	0.6124
	10	0.3780	0.4010
	100	0.0365	0.0342
Leaky ReLU	0.0001	0.9291	0.8873
	0.001	0.9186	0.8761
	0.1	0.7554	0.7474
	1	0.5474	0.5796
	10	0.0365	0.0290
	100	0.0350	0.0290
Sigmoid	0.0001	0.0344	0.0298
	0.001	0.0348	0.0387
	0.1	0.0314	0.0417
	1	0.0362	0.0357
	10	0.0352	0.0379
	100	0.0384	0.0335
Tanh	0.0001	0.9620	0.9263
	0.001	0.9556	0.9193
	0.1	0.7255	0.7124
	1	0.5414	0.5815
	10	0.0348	0.0320
	100	0.0358	0.0298

Table 5: Validation Accuracies for Different L2 Values

# of Hidden Layers	Val. Accuracy
3	0.9312
4	0.9568
5	0.9554

Table 6: Validation Accuracies for Different # of Hidden Layers

Table 6 shows the validation accuracy results of the model with different amount of hidden layers. The highest result was obtained when 4 hidden layers were used in the model. Thus, in the next model we are going to use 4 hidden layers to get the most accurate result.

By analyzing the results of each of the analysis we made, we decided to use Tanh activation function with L2 regularization term 0.0001.

Next, to check accuracy of our model, we created a confusion matrix (See Figure 13), which shows that there are a lot of True Positive values, which means that our model predicted almost everything accurately.

Selected CNN Model After running the model, the final accuracy is equal to around 0.947. In other words, based on our validation set, our model will be accurate and correct in 94.7% of all cases. However, the final testing accuracy after 20 cycles (20 epochs) of training our model is equal to 1.0. This infers overfitting, which is why the overall accuracy of the model is less than its testing accuracy.

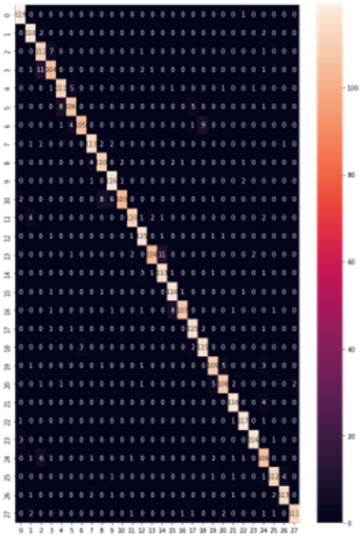

Figure 13: Confusion Matrix of the Best Accuracy Model

3 Conclusion

Out of the three machine learning algorithms used to train our dataset, the best results produced for each of the three algorithms were as follows: (i) SVM - 72.53%, (ii) Logistic Regression - 41.8% and (iii) and CNN - 94.7%. Several hyperparameters were varied when training the dataset with each of the three learning algorithms. Finally, the CNN architecture with 4 fully convolutional layers and L2 Reg. term = 0.001 was observed to produce the best results, with an accuracy of 94.7%. Any further increase in the number of hidden layers shows extremely small changes in final accuracy. A slightly poorer accuracy of 75.83% with a radial basis function (RBF) and C = 100, and the least accurate was the Logistic Regression with its best accuracy of 41.85%.

References

[1] https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

[2] https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

[3] https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python