Evaluating Machine Learning Algorithms For Handwritten Digit Recognition

1st Sunny Hsieh
Faculty of Sciences
University Utrecht
Utrecht, The Netherlands
s.hsieh@uu.nl

2nd Kathleen de Boer
Faculty of Sciences
University Utrecht
Utrecht, The Netherlands
k.k.deboer@students.uu.nl

3rd Tamara Sessink
Faculty of science
University of Utrecht
Utrecht, The Netherlands
t.r.sessink@students.uu.nl

***Abstract*—Handwritten digit recognition is a relatively simple static classification task, but a classic problem in pattern recognition, and it has been widely used as a test case for researchers to explore the data using machine learning techniques and pattern recognition techniques. In this research paper, we use the MNIST database, which consists of 70000 handwritten digits. First, we will fit multinomial logit models using 3 different feaures and combinations of these 3 features to see which combination can best distinguish between the different digit classes. Further, we are interested in evaluating and analysing a few common Machine Learning algorithms on the MNIST dataset. The algorithms that will be used are the Regularized multinomial logit model using the LASSO penalty, support vector machines, and feed-forward neural networks. The algorithms are compared using the accuracy, recall, precision metric, and confusion matrix. Moreover, the model performances are compared and tested on significance, using the McNemar significance test. The Feed-forward neural network has the best performance at classifying the handwritten digits, with an average accuracy score on all the class labels of 91,6 %. This is significantly higher than the other two algorithms.**

***Index Terms*—MNIST dataset, Support Vector Machine, Logistic Regression, Feed-forward Neural Network, Regularized multinomial logit model, Lasso**

I. Introduction

Handwritten digit recognition is a relatively simple static classification task, but a classic problem in pattern recognition. It comes down to the ability of a computer to recognize the human handwritten digits from different sources like images or papers and classify them into 10 predefined classes (0-9). It has been widely used as a test case for researchers to explore the data using machine learning techniques and pattern recognition techniques. The free Modified National Institute of Standards and Technology (MNIST) the database has helped the exploration of machine learning methods and ables the analysing of important tasks such as feature and model selection as reflected upon in this paper. The MNIST dataset has been used considerably often to run all kinds of leading algorithms, which is the reason why it is often used to assess the relative performance of a novel algorithm. The dataset contains a total of 70,000 handwritten digits. A more elaborate description of the MNIST dataset will be given later in this paper.

In the literature convolutional neural networks have the best overall performance on the dataset as can be seen in the study from Ciresan et al. 2011 [2]. However, without the use of a convolution structure and or a special type of preprocessing, the lowest error rate in the literature, 0.83%, is achieved using a deep stacking neural net. [3] (Deng et al. 2011). Shallower neural nets tend to result in a higher error rate. Support vector machines also have a good performance on the data-set. Chychkarov et al. implemented a few algorithms including a Support Vector Machine, a Multilayer perceptron, and CNN on the MNIST dataset [1]. They found that with a simple setup all classifiers demonstrated a high accuracy (96-98%), the CNN resulting in the highest score. However, the CNN took an exponential amount of computing time. In all previous cases preprocessing and eliminating useless variables is fundamental for performance. For this study, the main objective is to evaluate a few common Machine Learning algorithms on the MNIST dataset. Further, we will

also fit a multinomial logit models using a different combination of features to see which combination can best distinguish between the different digit classes. Different hyperparameters will be evaluated for tuning for the machine learning algorithms. In the paper of Kim et al. [4], they used three machine learning algorithms to classify digits, namely the support vector machine, decision tree, and k-nearest neighbors. In most cases, the support vector machine algorithm achieved the highest accuracy by classifying digits. Therefore, we are interested to compare the support vector machine algorithm with other common machine learning algorithms. The algorithms that will be experienced include the regularized multinomial logit model using the LASSO penalty, support vector machines, and feed-forward neural networks. As these are known to be among the best performers on the MNIST dataset as discussed before. Moreover, we will evaluate the performance of the ink feature, horizontal symmetry feature, and vertical symmetry feature that constructed by the authors. These feature are used to fit a multinomial logit model. The analysing of the models will be visualized with the use of plots and charts that have been implemented with the use of matplotlib. The findings of the experiments will be tested on significance and this will be discussed.

II. Description of the data

A. Dataset

For our research we used the dataset MNIST (Mixed National Institute of Standards and Technology). The MNIST database was introduced by LeCun et al. in 1998 [5]. The dataset consist of 785 columns, were the first column represent the label. The label can be a value from 0 up and till 9 and represent the digit that is written by the user. The other 784 columns represent the pixels of the image, and is represented by 28 pixels in height and 28 pixels in width. The value of the pixel is a value between 0 and 255. The values represent a scale from 0 up and till 255. How higher the number how darker the pixel is.

The MNIST database contains preprocessed and formatted, including segmentation and normalization, handwritten digit recognition that saves enormous effort for researchers such that they can mainly focus on the classification task. There are 60,000 instances for training and 10,000 for testing, so the database contains in total of 70,000 instances. These training and test sets are established so that a same writer of a handwritten digit would not be found in both datasets. For the training set alone, there are more than 250 writers. Examples of the handwritten digits can be seen in figure 1.

Fig. 1. Example of the MNIST images.

III. Exploratory analysis

We noticed that the data had many useless pixels in it. For the pixels 0 , 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ,11, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 52, 53, 54, 55, 56, 57, 82, 83, 84, 85, 111, 112, 139, 140, 141, 168, 196, 392, 420, 421, 448, 476, 532, 560, 644, 645, 671, 672, 673, 699, 700, 701, 727, 728, 729, 730, 731, 754, 755, 756, 757, 758, 759, 760, 780, 781, 782, 783 they all had a pixel value of zero, in total 86 useless pixels. We can see in 2 where the yellow pixels denote the pixels that were never used. It looks like there are many but if we look in more detail, we can see that not a single row or single column are all containing a value of zero, which means that each row and each column do carry some information around them. That is also he reason why we choose to use the data which contains 28x28 pixels instead of cropping all data into 14x14 pixels in order to remain all information each row and column carries around. However, we can't say with confidence that if we used 14x14 pixels, we would get a totally different result, but we do think that we will be losing at least some information by doing so, which

would mean that the result and accuracy would be at least slightly affected.

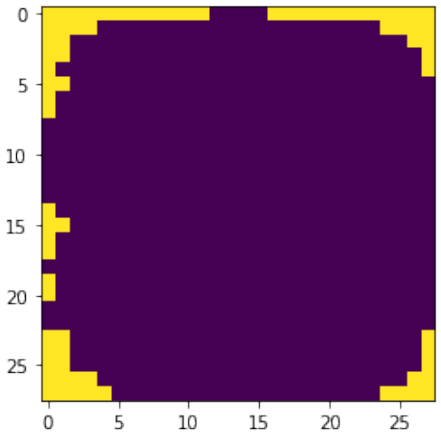

Fig. 2. None used pixels

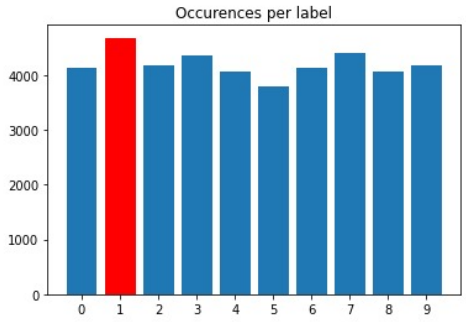

Fig. 3. Occurrences per label

Next, we noticed that among the labels, the class 1 occurred mostly among all cases (0 to 9). The majority class, 1, is contributing around 11% of the 42,000 instances. If we simply predict the majority class we would get an accuracy of 11%. We will use this as our baseline through our experiments. In figure 3, the occurrence per label can be seen.

IV. Experimental setup

The first part of our experiment is to invent two features ourselves. The first feature we came up with is ink, where ink quantifies "how much ink" a digit costs, in other words, the total sum of pixels of a digit. In table I, there is the average amount of ink and standard deviation per class displayed. We expect from classes such as 0 and 1 that they will be predicted better than other classes due to the extraordinary difference in mean. The have the lowest and highest mean so it might be easier for the model to classify these two classes.

TABLE I
SHOW THE MEAN AND STANDARD DEVIATION OF INK PER CLASS

Class	Mean	Std
0	34632	8462
1	15188	4409
2	29871	7653
3	28320	7574
4	24233	6375
5	25836	7527
6	27735	7531
7	22931	6168
8	30184	7777
9	24553	6465

Fig. 4. Examples of horizontal symmetry

The second feature we came up with is horizontal symmetry, where we calculated the summed pixels of the upper-half image (pixel 0 to pixel 391) and divided it by the total summed pixel. In figure 4 examples of horizontal symmetry by digits is displayed. Further, in table II the average amount of ink per digits (based on the upper-half) and standard deviation per class is displayed. We see that some classes differs a lot, such as class 0 and class 6, and because of the low standard deviation we do expect that some classes will be predicted pretty well. On the other hand, we expect that classes such as 0 and 1 or class 3, 8 and 9 will be hardly distuinguised by the model.

The third feature we came up with is the vertical symmetry, where we calculated the summed pixels of the left-part of the image and divided it by the total sum of pixels in the image. In table III, the average amount of ink per digits (based on the left part) and standard deviation per class is displayed. Further, in figure 5 examples of vertical symmetry by digits is displayed.

TABLE II
SHOW THE MEAN AND STANDARD DEVIATION FOR THE HORIZONTAL SYMMETRY

Class	Mean	Std
0	0.4795	0.0228
1	0.4741	0.0207
2	0.4090	0.0467
3	0.4905	0.0373
4	0.4392	0.0487
5	0.4919	0.0444
6	0.3965	0.0424
7	0.5444	0.0478
8	0.4907	0.0281
9	0.4970	0.0457

TABLE III
SHOW THE MEAN AND STANDARD DEVIATION FOR THE VERTICAL SYMMETRY

Class	Mean	Std
0	0.4266	0.0310
1	0.2100	0.1318
2	0.3736	0.0405
3	0.3382	0.0445
4	0.3782	0.0534
5	0.3909	0.0456
6	0.3907	0.0513
7	0.3286	0.0609
8	0.3731	0.0521
9	0.3383	0.0485

Fig. 5. Examples of vertical symmetry

We extracted these three features individually and trained four multinomial logistic regression models:

1) A logistic regression model with only the feature ink.
2) A logistic regression model with only the feature horizontal symmetry.
3) A logistic regression model with only the feature vertical symmetry.
4) A logistic regression model with all the features, ink, horizontal and vertical symmetry

For all our multinomial logistic regression models we used all 42,000 instances as training set and we also used the same 42,000 instances to evaluate our model.

The second part of our experiment we will build three different model and using these three models to analyse the data. These three models are:

1) The regularized multinomial logit model (using the LASSO penalty).
2) Support vector machine.
3) Feed-forward neural network.

We drew a random sample of size 5,000 from the data and used these 5,000 examples for training and model selection (using cross-validation). In this part, we used the 784 raw pixel values themselves as features. In the section exploratory analysis, we saw that 86 pixels never contain any information (in all 42,000 these pixels had the value 0). If we now cropped the 28x28 data into a 14x14, that would mean we would have 196 pixels left as features, we certainly would get rid of most useless pixels but we would also delete a lot of features that do contain useful information. Furthermore, when passing down all 784 features would mainly affect the running time, which even wasn't excessively long. Also, the weights of these 86 useless features (pixels), would likely be set 0 after training. So we don't see any disadvantage of passing down all 784 pixels as features (except the running time which was doable.)

For choosing the right parameters, we divided our training sample into 5 parts, such that each part contained 1,000 instances and then we used cross-validation. We estimated the error and used the parameters that derived the cross-validation score in our final model, and eventually evaluated our model, after training on the 5,000 instances, on the remaining 37,000 instances. For each type of model, we used the same 5,000 instances for training and the same 37,000 instances for evaluation, otherwise, the comparison wouldn't be fair.

Deriving the parameters of each model we used a slightly different setup.

A. *Python functions*

To give a clear view of what we did we will elaborate on the functions we used. First of all, we did our experiment in the language python. We used a function for tuning the parameters this was

used for all the models. The name of the library is sklearn.model_selection with the function GridSearchCV(). This function has also an integrated cross-validation generator, which can be called by the variable cv. To give a more clear view, cv = 5 means a 5-fold cross-validation.

For the first model namely regularized multinomial logit model (using the LASSO penalty) we used the library sklearn.linear_model. In the library, we used the function LogisticRegression(). We used the GridSearchCV() to tune the parameters The parameters we used are C value and Penalty. The array of different possibilities for C is described in section IV B. The value for the penalty = L1, because logistic regression with an L1 regularization is standing for the Lasso penalty.

For the second model, the support vector machine, we used the library sklearn.svm and the function linearSVC. LinearSVC is a very fast implementation of Support Vector Classification in the case of a linear kernel. First, we used the GridSearchCV() to tune the parameters. We used different parameters for the C value, which will be described in section IV C. Further, we used for cv = 5 and estimator = linearSCV(). After tuning the parameters and training we used the value for C as the new value for the testing phase.

For the third model, feed-forward neural network, we used the library sklearn.neural_network with the function MLPClassifier(). We also used the GridSearchCV() to tune the parameters. In this case, there is not just one parameter we want to test but five different parameters, namely hidden_layer_sizes, activation, solver, alpha, and learning_rate. The values which we tested for those parameters are discussed in section IV D.

B. The regularized multinomial logit model (using the LASSO penalty)

For obtaining the parameters of the regularized multinomial logit model we passed down a C array containing the values:

$$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$$

After using cross-validation it returned the best score among these alpha's of 88.7 for C = 0.1. We want to build a model that neither underfits or overfits and here it seems a smaller value for alpha will be appropriate. For the L1 case this means that not a lot of coefficients will be set to zero, the sparsity will decrease. Note that a higher alpha implies stronger regularization and a higher restriction on dimensions, thus a weaker penalty. The different scores of values for C will be discussed later and can be seen in figure 11.

C. Support vector machine

For obtaining the parameters of the support vector machine classifier we passed down a C array containing the values:

$$0.001, 0.01, 0.1, 1, 10, 100$$

The C parameter tells the SVM optimisation how much the model wants to avoid misclassifying each training instance. When we have larger values of C, the optimisation will have a smaller margin hyperplane. After using cross-validation it returned the best score among these C's of 0.83639 for C = 0.001. A value of 0.001 for C will result in a smooth decision boundary and will help against overfitting.

D. Feed-forward neural network

For the feed-forward neural network classifier, we used five parameters, namely hidden layer sizes, activation function, the solver, alpha, and the learning rate. For the solver, we had the values:

$$adam, sgd$$

The hidden layer sizes was containing the values:

$$(100,), (500,), (600,), (700,), (100, 100, 100), (100, 50)$$

The activation array was containing the values:

$$Tanh, ReLU$$

The alpha array was containing the values:

$$0.01, 0.025, 0.05$$

The learning rate array was containing the values:

$$constant, adaptive$$

After using cross-validation it returned the best initial score among these parameters of 0.9224 for solver = adam, hidden layer sizes = (500,), activation = ReLU, alpha = 0.05 and learning rate = constant. The network thus has only one hidden layer with 500 units. The average score on all class labels for recall and precision is 92 %.

E. Statistical test McNemar

For determining the best machine learning model and analyse the performance, a statistical test will be performed. Because we are interested in comparing different algorithms on the same test set we will be using the McNemar statistical significance test to conclude if the performance of the algorithms significantly differs. Typically, the test is used to compare only two different algorithms, hence all algorithms will be checked pairwise. The McNemar test gets its results from a contingency table. In the contingency table, two algorithms will be compared by the amount of correct and incorrect outputs. The McNemar statistic will be based on the bottom left and top right cell values. Those values represent the number of outputs one algorithm has correct, which the other has incorrect and vice versa. These values check whether the disagreements on the test are set to match or not. The null hypothesis entails that the two models disagree on the same amount. We use a significance threshold of α = 0.05. If the p-value $<=$ 0.05 the null hypothesis will be rejected.

V. Experimental results

A. A logistic regression model with only the feature ink

To visualize the performance of the logistic regression model with an ink feature, we will look at the confusion matrix. The confusion matrix can be seen in figure 6. In the figure can be seen that the labels 4, 5, 6, and 8 are never predicted by the model because the value of those is always zero. The reason for this is that those labels look like another label, which is more in the data. Therefore, it will never classify some labels. When looking at the occurrences per label the class with the lowest occurrence is label 5 with 3795. The second label is label 8 with an occurrence of 4063. Thirdly, it is label 4 with an occurrence of 4072 and as fourth, it is label 6 with an occurrence of 4137. We can thus conclude that labels 4,5,6 and 8 are never predicted because those occur less in the data compared to the other labels. Therefore, it will predict another label that it resembles. The ink feature resulted in an accuracy of 22.6 %. The low accuracy can be explained by the number of labels the model never predicts.

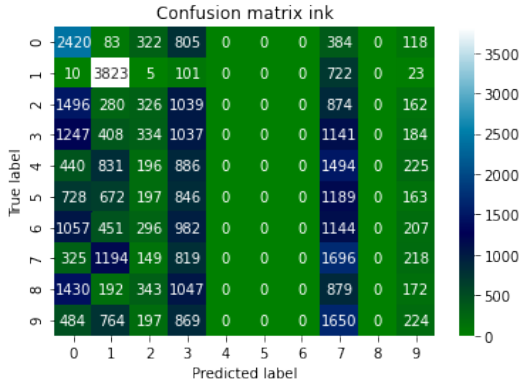

Fig. 6. Confusion matrix for the ink feature

B. A logistic regression model with only the feature horizontal symmetry

To visualize the performance of the logistic regression model with a horizontal symmetry feature, we will look at the confusion matrix. The confusion matrix can be seen in figure 7. In the figure can be seen that the labels 0, 5, and 8 are never predicted by the model because the value of those is always zero. In table II, the mean and standard deviation is displayed for the horizontal symmetry feature. A reason that 5 and 8 are never predicted is the same as with the ink feature, namely those labels have the lowest occurrence in the data. In contrary to the ink feature, is for the horizontal symmetry feature the label 0 never predicted. A reason for this could be that the mean for label 0 (0.4795) and label 1 (0.4741) are almost the same. Therefore, it will always predict the majority class 1. This can also be seen in the confusion matrix because the value for the predicted label 1 by true label 0 is high. The horizontal symmetry feature resulted in an accuracy of 27.1 %. The accuracy is a bit higher than the accuracy of the ink feature, which means more labels are predicted correctly.

C. A logistic regression model with only the feature vertical symmetry

The confusion matrix for the vertical symmetry feature can be seen in figure 8. Some labels are never predicted by the model, namely labels 4, 5, 8, and 9. The label 8 is mainly predicted as 0 which

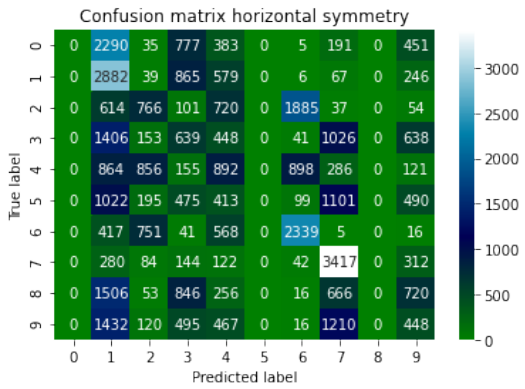

Fig. 7. Confusion matrix for the horizontal symmetry feature

is logical since the are identical in vertical symmetry (total symmetry). Furthermore, it is remarkable that 4 and 5 are mainly predicted as 0 as well, but if we look into the details, our vertical symmetry is defined as total ink of the left part of the image divided by the total ink. The number 4 and 5, especially 5, will have most of the times equal ink on the left and right part of the image, which would explain why they are mainly classified as the label 0. The number 9 is mainly predicted as 7, which is explainable by having a lower score of symmetry. Both numbers have more ”ink” on the left side due to their form, which would explain the similarities. The vertical symmetry feature resulted in an accuracy of 23.8 %. We could conclude that the accuracy is a bit lower compared to the horizontal feature. The reason for this could be that the horizontal feature has only three labels that are never predicted and the vertical feature has four.

D. Comparing the three logistic regression models with only one feature

If we look in more detail, the logistic regression model with only the feature ink predicts the numbers 0,1 and 3 more often correctly and the logistic regression model with only the feature horizontal symmetry predicts the numbers 2, 4, 6, 7, and 9 more often correctly. Further, the feature vertical symmetry predicts the numbers 0 and 1 most of the time correct. There can be seen that the logistic regression model with feature horizontal symmetry predicts more labels more often correctly. Also, the logistic regression model with feature horizontal symmetry has higher accuracy. The possible reason for the higher accuracy of the logistic regression model with the horizontal symmetry feature is because symmetry might contain more information than ink, in other words, if we train a model with these two features the model might give the feature symmetry a higher weight. The horizontal feature has higher accuracy than the vertical symmetry 23.8% $<$ 27.1 %. The reason for this could be that more numbers have vertical symmetry (label 0, 3, and 8) than horizontal symmetry (label 0 and 8). Additionally, it is worth noticing that labels 5 and 8 are never predicted correctly in both models.

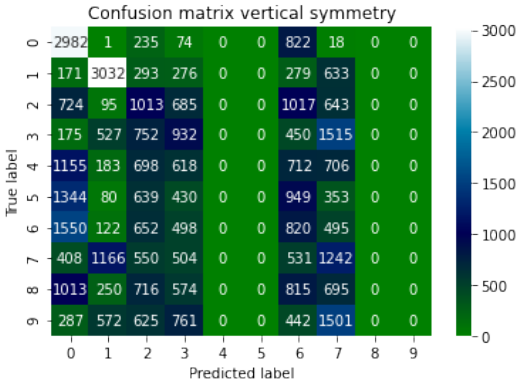

Fig. 8. Confusion matrix for the vertical symmetry feature

E. A logistic regression model with both features, ink and horizontal symmetry

The horizontal feature performed better than the vertical feature, therefore we will now look at the performance of the ink and horizontal feature working together. To visualize how the logistic regression model performs with two features, namely the ink and horizontal symmetry, we will look at the confusion matrix. The confusion matrix can be seen in figure 9. The combination of both the features resulted in a better outcome, because know there is only one label that is never predicted (label 5). The two features resulted in an accuracy of 37.6 %. This is a substantial improvement compared to just using the ink feature in the logistic regression model. We saw that in both individual models they weren't been able to predict any instances with label 5 or 8 correctly, but now we see that this is only the case for label 5. Label 8 is predicted for 408 instances correctly where it was 0 before for both models with only one feature. This would indicate

that using the feature symmetry and the feature ink together, the logistic regression model is now able to predict some instances correctly. Moreover, the logistic regression model with both features predicts the numbers 1,2,4, 8, and 9.

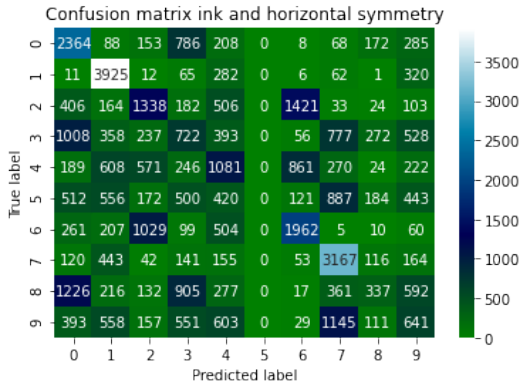

Fig. 9. Confusion matrix for the ink and horizontal symmetry feature

F. A logistic regression model with all the features, ink, horizontal and vertical symmetry

In figure 10, is visualized how the three features work together. In the confusion matrix of ink and horizontal symmetry, there was number 5 that was never predicted correctly. However, in the confusion matrix of figure 10 is that not the case anymore. Therefore, the addition of vertical symmetry has a positive outcome. This can also be seen in the increased accuracy. Using the three features together resulted in an accuracy of 42.7 %. This is quite an improvement compared to the accuracy of using only the ink feature, which was 22.6 %.

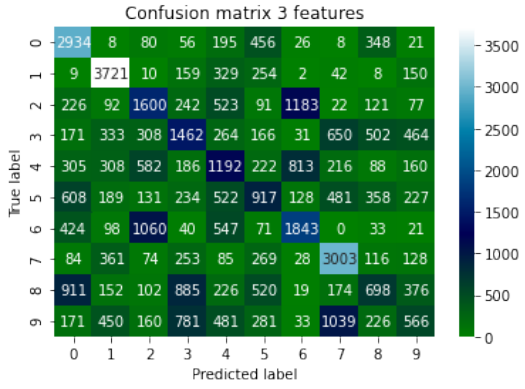

Fig. 10. Confusion matrix for the three feature together (horizontal and vertical symmetry and ink feature)

G. The regularized multinomial logit model (using the LASSO penalty)

We used the value of C = 0.1 for the testing part because it returned the highest initial score as can be seen in figure 11. The regularized multinomial logit model resulted in an accuracy of 88.7 %. The precision score resulted in 89.0 as well as for the recall score. The confusion matrix for the regularized multinomial logit model can be seen in figure 12. It is interesting to visualize the weights of the coefficients. In figure 13 the coefficients of the model are shown in an image. The digits can clearly be distinguished just by looking at the blue in the images of the coefficients for the different classes.

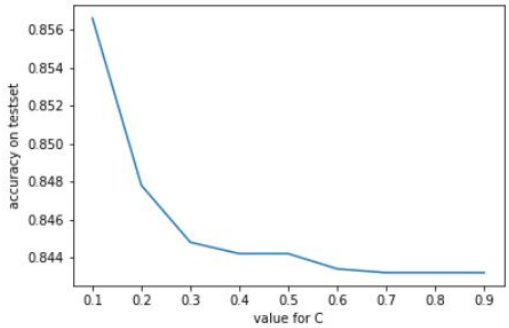

Fig. 11. Accuracy score for different values of C using gridsearchSV

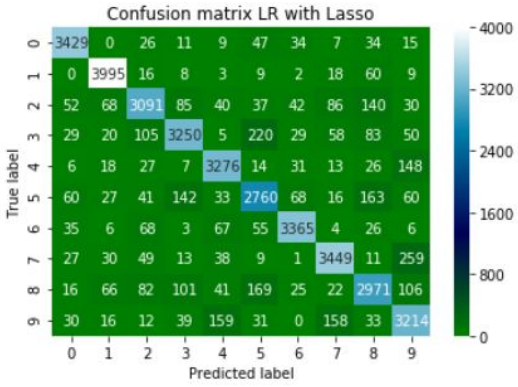

Fig. 12. Confusion matrix for LR with Lasso

H. Support vector machine

We used the value of C = 0.001 for the testing part because it returned the best score as can be seen in figure 15. The accuracy of the classifier resulted in 84.1 % The recall and precision score both resulted

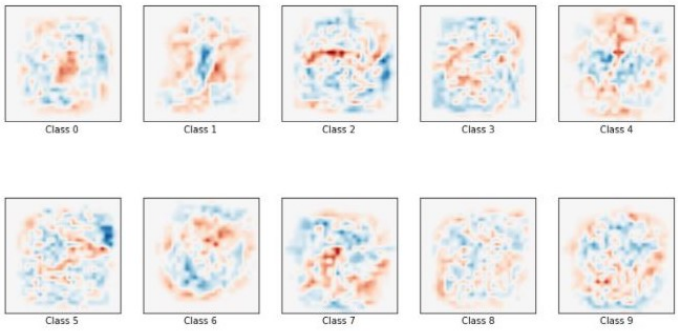

Fig. 13. Visualization of the coefficients as images per class

in 84.0 %. Training on the 5000 data examples took 2.42 seconds. The confusion matrix for the support vector machine classifier can be seen in figure 14.

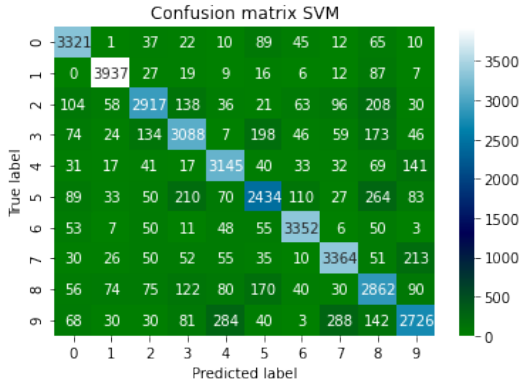

Fig. 14. Confusion matrix for SVM

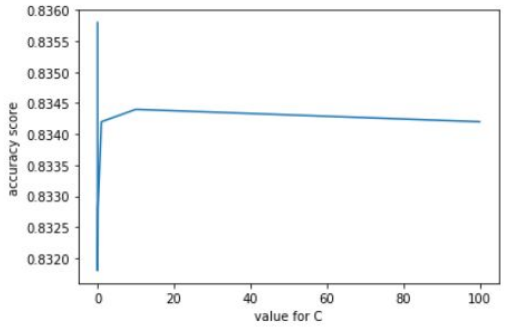

Fig. 15. Accuracy score for different values of C using gridsearchSV

I. Feed-forward neural network

For implementing the feed-forward neural network the sklearn multilayer perceptron is used. A multilayer perceptron is a class of feedforward artificial neural networks that consist of at least three layers: input, hidden, and output. All neurons use a non-linear activation function except for the input neurons, in this research the activation functions Tanh and ReLU were tested. The resulted hyperparameters that are used for testing are hidden layer sizes = (500,), activation = ReLU, alpha = 0.05, solver = adam and learning rate = constant. Using those parameters resulted in the accuracy of 91.6 % of the classifier. The confusion matrix for the Feed-forward neural network classifier can be seen in figure 16. Training the multiplayer perceptron on the 5000 data examples took 17,5 seconds (which is quite slow in comparison to the SVM).

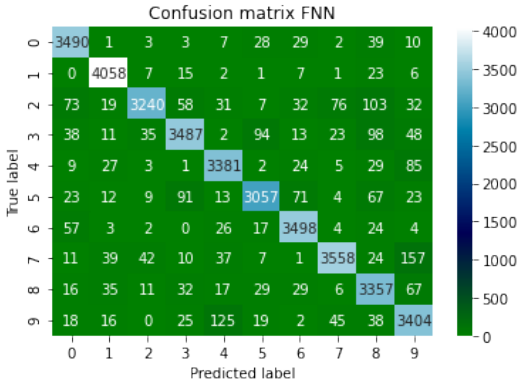

Fig. 16. Confusion matrix for FNN

J. Best performers

We will summarize the different performances of the algorithms on the MNIST dataset. The accuracy, precision, and recall are computed from the confusion matrix. The formulas for those are desribed below.

- Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Recall = TP / (TP + FN)
- Precision = TP / (TP + FP)

The regularized multinomial logit model (with Lasso penalty) resulted in an estimated accuracy of 88.7 %. The support vector machine resulted in an estimated accuracy of 84.1 %. Additionally, the feed-forward neural network resulted in an estimated accuracy of 91.6 %. Further, the regularized multinomial logit model resulted in a recall and precision score of 89 %. For the support vector machine, the recall and precision score was 84%.

Lastly, the recall and precision score for the feed-forward neural network was 92 %. In table IV, there can be seen a summary per algorithm. the best performer on this dataset based on all used metrics would be the feed-forward neural network.

TABLE IV
SUMMARY OF DIFFERENT MODELS

Algorithm	Accuracy	Precision/Recall
The regularized multinomial logit model	88.7 %	89 %
Support vector machine classifier	84.1 %	84 %
Feed-forward neural network	91.6 %	92%

As discussed earlier, we will compare the different machine learning algorithmes using the McNemar statistical significance test to conclude if the performance of the algorithms significantly differs. Typically, the test is used to compare only two different algorithms, hence all algorithms will be checked pairwise. The McNemar test gets its results from a contingency table. The three contigency tables are displayed in table V, VI and VII. In table V, are the support vector machine and feed-forward neural network compared. Further, in table VI the support vector machine and regularized multinomial logit are compared. Lastly, in table VI, the feed-forward neural network and regularized multinomial logit are compared. We compared all three algorithms and the results are the following: For the comparing of the Regularized multinomial logit model and the Support vector machine: chi = 868 and p-value = 1.44e-175 $<$ 0.05 so the null hypothesis will be rejected. Comparing the Support vector machine and the Feed-forward neural network resulted in: chi = 806 and p-value = 0.00 $<$ 0.05, so reject the null hypothesis. When comparing the Regularized multinomial logit model and the Feed-forward neural network the values are: chi = 865 and p-value = 2.4e5-147 $<$ 0.05, so the null hypothesis is rejected. This statistical test suggests that all algorithms differ significantly in performance.

TABLE V
THE CONTINGENCY TABLE FOR THE COMPARISON OF THE SUPPORT VECTOR AND FEED-FORWARD NEURAL NETWORK

	Support vector correct	Support vector incorrect
Feed-forward neural network correct	31934	2290
Feed-forward neural network incorrect	865	1911

TABLE VI
THE CONTINGENCY TABLE FOR THE COMPARISON OF THE SUPPORT VECTOR AND REGULARIZED MULTINOMIAL LOGIT ALGORITHM

	Support vector correct	Support vector incorrect
regularized multinomial logit correct	30332	2467
regularized multinomial logit incorrect	868	3333

TABLE VII
THE CONTINGENCY TABLE FOR THE COMPARISON OF THE FEED-FORWARD NEURAL NETWORK AND REGULARIZED MULTINOMIAL LOGIT ALGORITHM

	Feed-forward neural network correct	Feed-forward neural network incorrect
regularized multinomial logit correct	30394	806
regularized multinomial logit incorrect	3830	1970

VI. DISCUSSION

First, we would like to talk about the limitations of the individual features. The symmetry feature was designed to capture the proportion of the upper half of the image or left half of the image to the whole image. However, it isn't able to distinguish 5 as asymmetric due to the proportion on the left part of 5 is equal to the right part of the image. Thus, the feature symmetry, either horizontal or vertical, could be improved such that it can really capture whether the image is symmetric. Furthermore, other features might be implemented such as roundness (which would capture that 8 has two rounds for example).

Second, about the models that use all 784 pixels as features, we could have used a more variety

of parameters for cross validation. More variety of parameters would improve the chance of getting higher accuracy. Also, combining a logistic regression model with Lasso, SVM or feed-forward network model with one or all three feature that we designed would be very interesting. To see whether combining the models that are trained on the 784 pixel features with our features can be taken into account for future work.

VII. Conclusion

In our research, we fitted a multinomial logit model using three features and looked at which combination resulted in the most improvement for classifying digits. The three different features we used are how much ink a digit costs, horizontal symmetry, and vertical symmetry. We can conclude that the best combination of features is using all three of them together, as expected. This resulted in an accuracy of 42.7 %. If we would only use the amount of ink per digit as a feature we resulted in an accuracy of 22.6 %. Therefore, we can conclude that the use of all three features together increased the accuracy by more than 20 %. Thus, the three features together can better distinguish between the different classes than only using the ink feature. Moreover,if we only look at each feature individually, horizontal symmetry scored the highest with 27.1 %. Further, the paper has implemented three Machine Learning models namely Support Vector Machine Classifier, a regularized multinominal logit model with the LASSO (L1) penalty, and a Feed-forward neural network using MNIST datasets. The algorithms are compared using the accuracy, recall, precision metric. Also, the model performances are compared and tested on significance, using the Mc-Nemar significance test. The Feed-forward neural network has the best performance at classifying the handwritten digits, with an average accuracy score on all the class labels of 91,6 %. This is significantly higher than the other two algorithms, which means it improves on both of them.

References

[1] Y. Chychkarov , A. Serhiienko, I. Syrmamiikh, and A. Kargin, "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks," In CMIS, 2021, pp. 496-509

[2] D. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Convolutional neural network committees for handwritten character classification," in Proc. ICDAR, 2011

[3] L. Deng and D. Yu, "Deep convex network: A scalable architecture for speech pattern classification," in Proc. Interspeech, August 2011

[4] G. Kim, S. Kapetanovic, R. Palmer and R. Menon, "Lensless-camera based machine learning for image classification, " arXiv preprint arXiv:1709.00408, 2017

[5] Y. LeCun, C. Cortes and C.J.C. Burges, "The MNIST Database of Handwritten Digits," 1998