Comparing ML Model Performance against CNN (Deep Learning) within the MNIST dataset

Michael Barrett

EC Utbildning

Kunskapskontroll 2 – Machine Learning

2025/03

Innehållsförteckning

1 Inledning

The MNIST dataset consists of 70,000 unique handwritten digits written in black ink on a white background. The images are greyscale, and are by default 28x28 pixels. Each image can therefore be understood as a 2D array. Once flattened, this becomes a 784 dimensional vector with each vector holding a value between 0 and 255, with 0 being white and 255 being black (LeCun et al 1998).

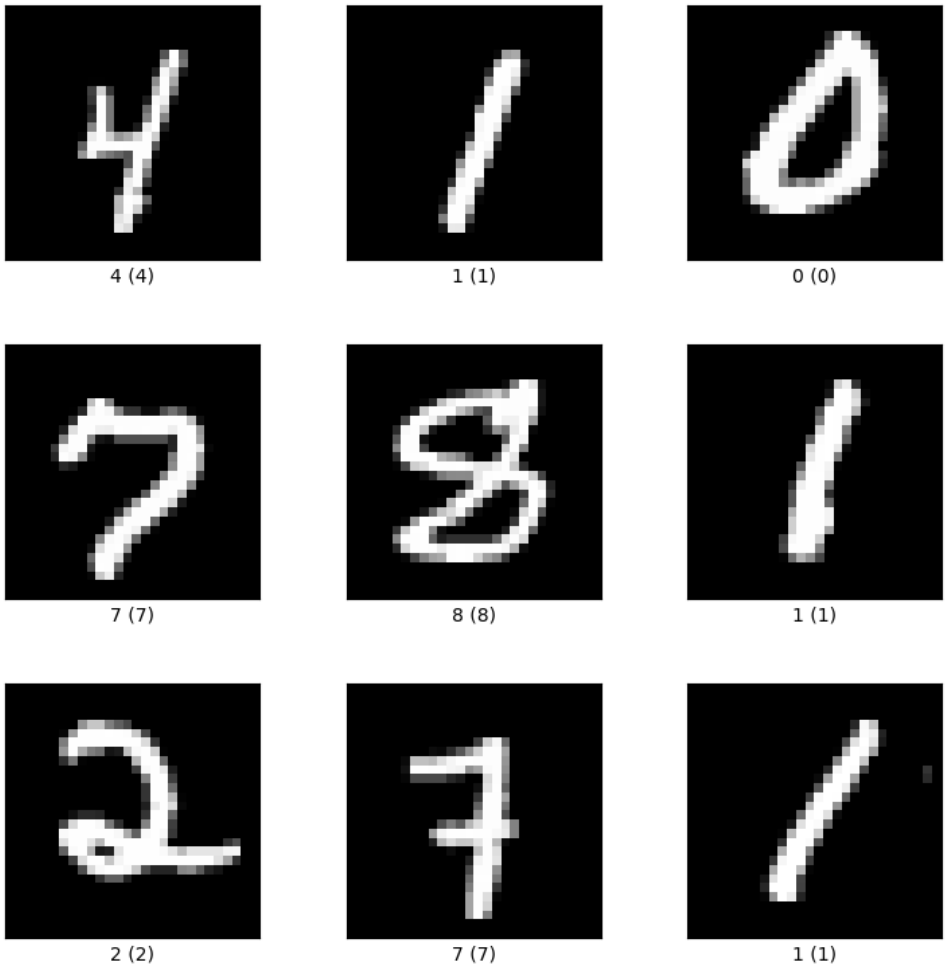


Figure 1, MNIST Data example (with inverted greyscale) (TensorFlow.org, 2024)

MNIST's format makes it an excellent dataset for Machine Learning and Deep Learning, wherein different ML models can be built. By utilizing the 60,000 training and validation observations to train and shape the model one can then run the trained model on the 10,000 observations strong test set. Comparing these models and their proficiency at accurately predicting the test set is a useful way of introducing the concept of model building in all of its stages, from exploring the data to hyperparameter tuning.

For this study, I thought that it would be interesting to build three models of varying complexity that I could compare their performance on a Streamlit application and tally up a score based on data that comes from outside the MNIST dataset entirely. This data would be in the form of figures drawn with a mouse into a 200x200 square on a Streamlit application, which would then be transformed and read by the three different models.

I wanted three models that performed at clearly different levels, in order to make it visually clear how different models perform not only on the test data, but also on real-world data that is entirely separate from the MNIST dataset. I was particularly curious to see whether some models performed better on the MNIST test data but performed comparatively worse on real-world data.

For this analysis, the three models chosen to compete against each-other were as follows:

- Logistic Regression model – Basic Machine Learning model.
- Random Forest model – More Complex Machine Learning Model.
- CNN (Convolutional neural network) – Deep Learning model.

It was my expectation that the logistic regression model would be the weakest overall, and that it would also perform poorest on real-world data. Random forest would perform between the CNN model and the logistic regression model, and finally, that the CNN model would be by far the most well-performing model in both the test data and real-world data.

2 Teori

In order to undertake this task, we should justify that the three models chosen for this classification exercise are suited to being trained on and then predicting MNIST and Streamlit application data.

The task at hand is that of a classification – we are training our models to label an image with a class (number). Regression models are concerned with likelihoods; how likely is (*x)* event to happen when the characteristics surrounding it are (*y).* By this logic, there are regression models that can also be tasked with classification problems (Geron, 2019).

Logistic regression, for example, can output a probability for each 'class' (number) in the MNIST dataset, and the class with the highest probability can be put forward as the predicted value. This therefore makes it a viable if not weak model for outputting predictions of this nature.

The Random Forest Machine Learning algorithm has it's roots in Brieman's 'bagging method' (1996), wherein he generated 'multiple versions of a predictor' and then used these to aggregate a prediction. This idea was later developed into the RF algorithm, which can best be understood as consisting of multiple decision trees. These decision trees are trained on data, and then 'vote' or rather push their prediction into an aggregate of the other decision trees. Once there, the most likely prediction is put forward. In the case of a classification problem – as we are handling here in this study – it would likely be predicted via majority voting (IBM, 2025). The class with the most votes will be put forward as the output.

The final model being included in this analysis is a CNN (Convolutional Neural Network). This is a Deep Learning model which is specialized for image and video recognition and classification, which makes it a prime model for our goal here. This model is built on several layers (one or more convolutional layers, one or more pooling layers, and finally a FC (fully-connected) layer). Each layer is specialized with different tasks that are made to strengthen the understanding of the data that it is being modelled on (IBM, 2025). Ultimately, the model runs these layers and eventually attempts to classify the image at hand.

3 Metod

The MNIST dataset was imported from LeCun's (2025) source site. From here, a basic EDA was undertaken that explored the form of the dataset. This EDA looked primarily at how the distribution of figures looked within the dataset, and the datasets characteristics. This EDA discovered that over the 70,000 observations they were relatively evenly distributed, with 5 having the least observations (a little over 6000 total), and 1 having the most observations (almost 8000).

The data was then split into training, validation and test sets, totalling 50,000 (training), 10,000 (validation), and (10,000) test. A fourth set was made that consisted of both the training and validation set, totalling 60,000. The training/validation set was used to train the model once hyperparameters had been selected to provide as large of a database for the model to train on as possible.

Logistic Regression Model

The first model that was fitted to the MNIST data was the logistic regression model. This model was ran with baseline parameters as follows:

{log_reg = LogisticRegression(solver="lbfgs", max_iter=100, multi_class="multinomial", n_jobs=-1)}

This was to get a broad idea of model performance prior to utilizing hyperparameter tuning to improve the model performance. The first performance of this model using baseline parameters came to just below 92% on the training data.

The next step was to perform a GridSearchCV to find the best parameters for the logistic regression model. The best hyperparameters were found to be as follows:

{log_reg_best = LogisticRegression(C=1, solver='saga', max_iter=1000, random_state=42)}

Running these hyperparameters saw a small drop in the model accuracy score, which is due to the way that the GridSearchCV folds data – a drop in accuracy does not mean that the hyperparameters chosen are worse, but rather that all the hyperparameters included in the GridSearch CV are being tested differently to how the parameters were utilized in the baseline model.

Furthermore, these may still not be the best hyperparameters. Within the GridSearchCV limits that were set, they are, however this GridSearchCV after having ran for approximately thirty minutes, found that the coefficient had not converged and that it had reached the 1000 iteration point. This meant that with more iterations it may find better hyperparameters, but I felt that this would yield a very small improvement in the model overall.

The model was trained using both the training and validation datasets with specific hyperparameters and then evaluated on the MNIST test set, achieving an accuracy of 81.69%. This is notably lower than the 92% accuracy attained with the initial Logistic Regression model. However, it is important to emphasize that the primary goal of this model is not to maximize performance on the MNIST test set, but rather to improve generalization to real-world handwritten digits inputted via the Streamlit application. The decrease in MNIST test accuracy does not necessarily indicate poor model quality; instead, it may reflect design choices intended to enhance robustness to variations in handwriting styles and drawing conditions outside of the MNIST dataset. This trade-off between MNIST accuracy and real-world adaptability is a key factor in evaluating the model's performance. Therefore, the true measure of success is the difference between test set accuracy and real-world performance in

the Streamlit application rather than a direct comparison to MNIST benchmarks. This justification applies consistently across all three models tested in this analysis.

Once the model had been saved it was imported into the first Streamlit application that I had built and tested there. I ran 100 (10 of each digit) test and achieved a 67% accuracy on real-world data after some modifications were made to the application to best standardize the real-world imagery to MNIST image characteristics.

Random Forest Model

The Random Forest (RF) was then built in a similar process. The data was first flattened, because RF does not handle images, and instead works with vectors. This means that each pixel in the 28x28 MNIST image is treated as a data point for the model, totalling 784 features per image.

I ran the RF training, which resulted in a 100% accuracy. This was difficult to understand – and I am still not entirely certain what occurred here. After I ran hyperparameter testing I got instead a 96.57% accuracy on the validation set instead, which is more normal. Once the model had been finalized, I ran it on the test set and pulled an accuracy of 96.45%, which is close to the previous validation accuracy. This is a good sign for model fit.

After this, visualizations of the model were produced and then the model was saved to be imported into a Streamlit application. After having issues with having multiple models running in the same Streamlit application I decided instead to instead run the three models in separate Streamlit apps. This was done for ease of use, and because the goal is just to compare the three models on their Streamlit performance and test performance, which is not hindered by this choice.

Within the application, similar normalizations of the image were done to try to best imitate MNIST imagery, and the results came to a 73% accuracy on real-world data.

CNN (Convolutional Neural Network)

The final model built for this analysis was the CNN (Convolutional Neural Network). The MNIST data was reshaped to four dimensions for the CNN, and then ran on 20 epochs, which saw the test accuracy run up to 98.72%. This would undoubtedly go higher if given more training time.

This was first trained on the training data with 10 epochs. There was no hyperparameter tuning done for this model. Then the model was run on the test data and initially received a score of approximately 98.45%. A second CNN model was built that utilized image augmentation to attempt to improve the applicability of the model to the Streamlit application data. This model was trained using image augmentation techniques and with an epoch of twenty but otherwise was identical to the previous model.

This augmented CNN model was implemented into the final Streamlit application and achieved the best application performance of all the three models, at 80% over 200 runs (20 per number).

4 Resultat och Diskussion

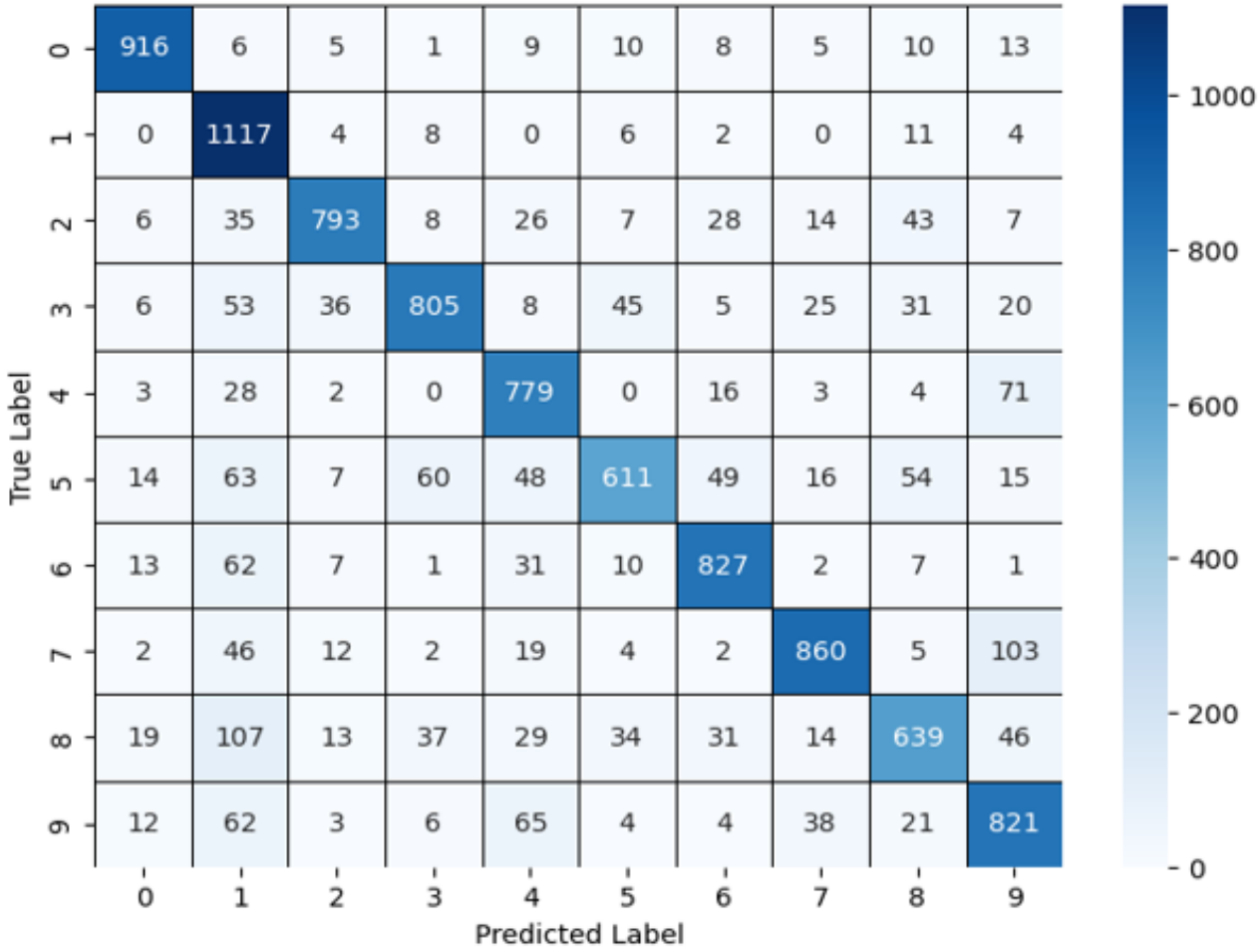
Model Name	Test Set	Application	+/-
Logistic Regression	81.69%	67.00%	-14.69%
Random Forest	96.45%	73.00%	-23.45%
CNN (Convolutional Neural Network)	98.72%	80.00%	-18.72%

Above, we can see the performance of all of our models on both the test MNIST data and the data that is ran through the application. There is an obvious improvement in model performance in both the test data set and the application as the models become more complex. It is clear that the models struggle to replicate their accuracy when being provided with non-MNIST data. This suggests that to some extent all of the models are being overfit on the MNIST data, but there is also the possibility that the format of the MNIST dataset was not adequately replicated in the application before the image was passed to the model.

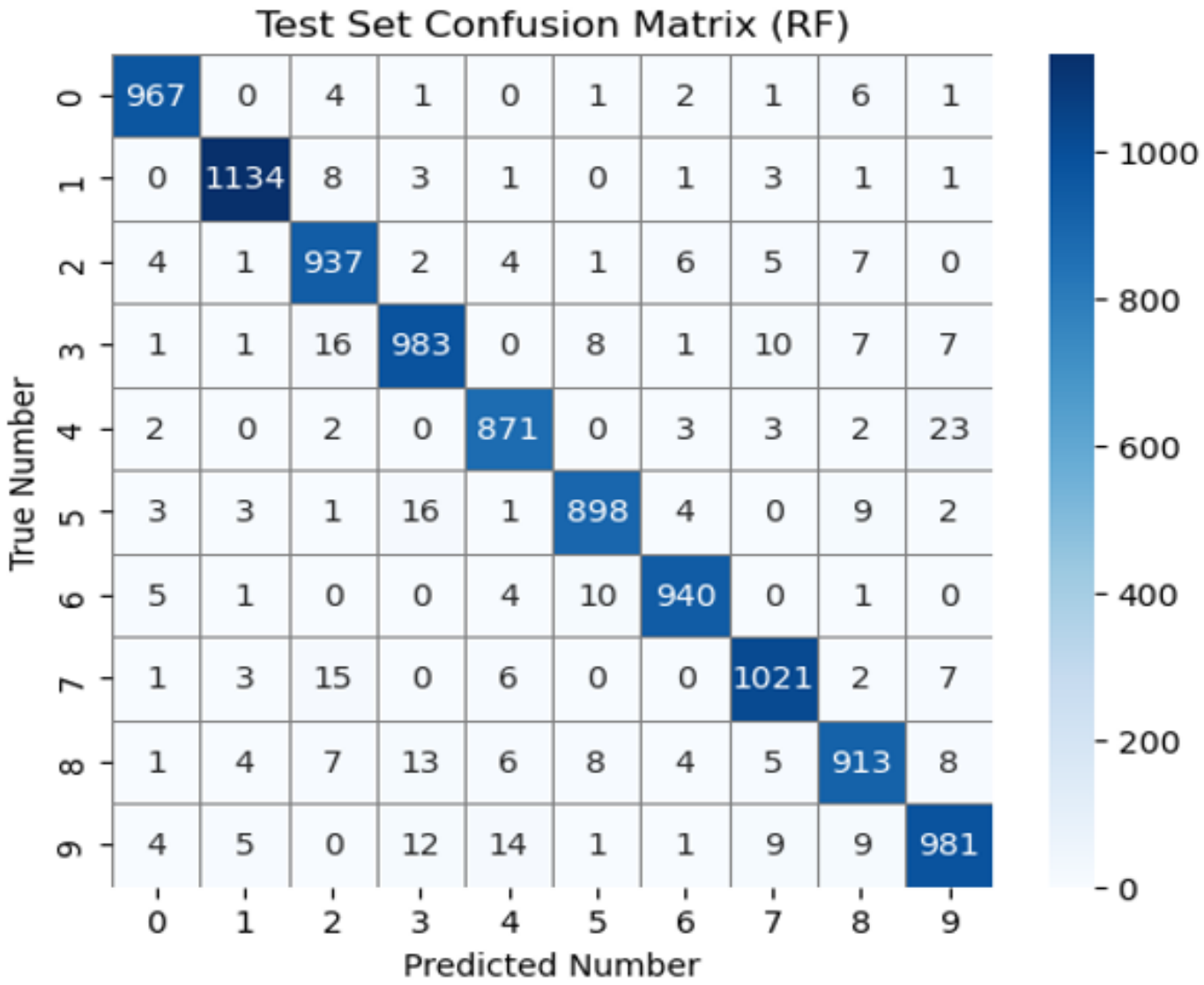
Another important note is that the MNIST dataset are handwritten digits that always fall within the central 20x20 grid of a 28x28 square. Our final model was not restrained to this 20x20, and was also not handwritten – instead utilizing a mouse. These small differences in image generation for the model could account for losses in accuracy. Furthermore, depending on where and when the data was collected for the MNIST dataset, it may no longer be representative of written digit habits today. The age of technology has changed how certain digits are written in certain ways, and writing habits also vary over cultures.

The application test that was undertaken in this study had digits only generated by myself which allows bias to easily creep in to the application performance. This is especially clear when I realized that testing the application accuracy had led to the model instead training me on what forms and edges each digit should have to best be read by the models.

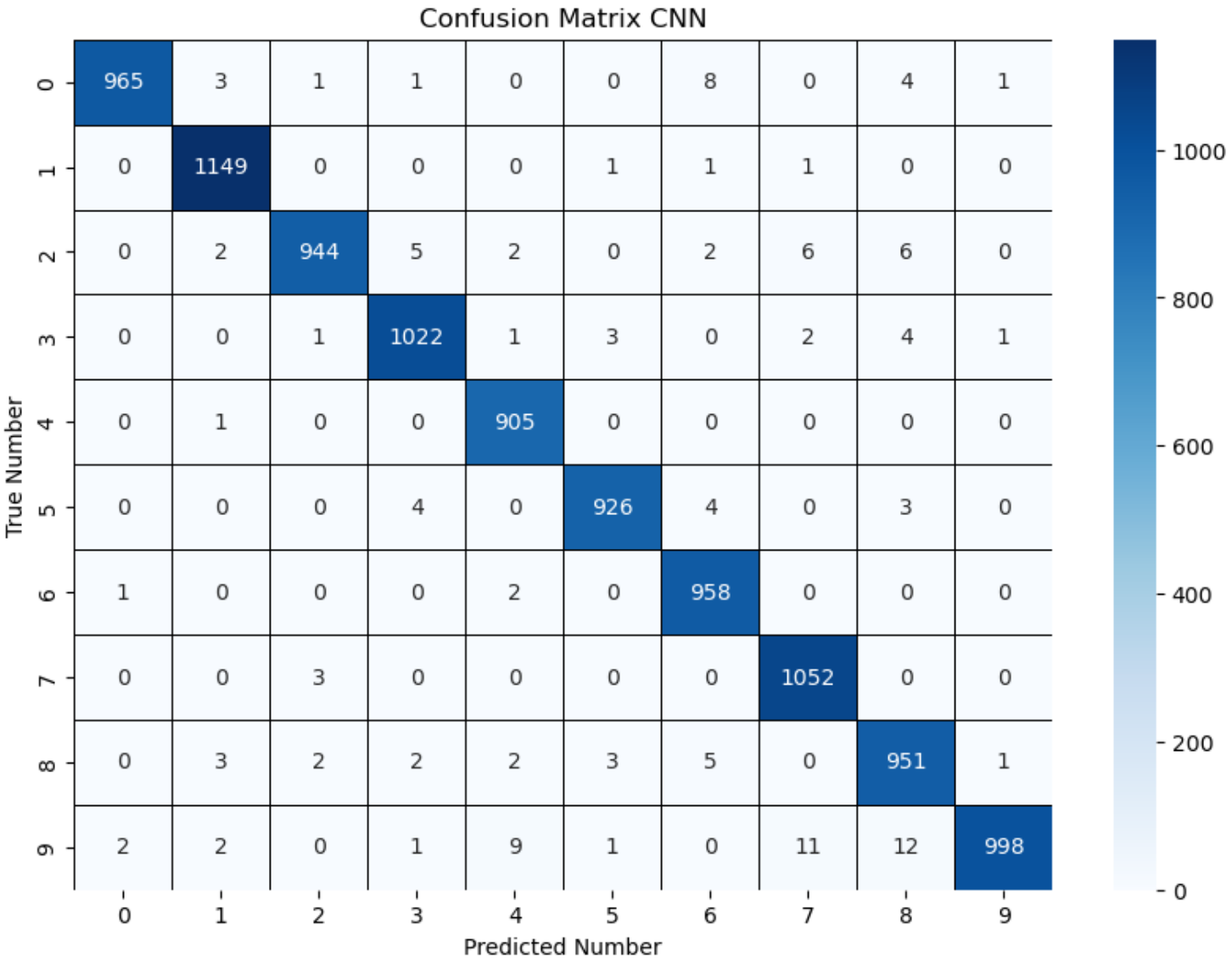
In conclusion, the CNN model has shown to perform the best on both the MNIST test data and perform the best on the Streamlit application, but still experiences a noticeable drop in accuracy between the two. Our CNN and Random Forest models have been trained to the MNIST dataset exceptionally well, whilst the logistic regression sacrificed some test accuracy to improve it's application accuracy, which led to the logistic regression model experiencing the least accuracy loss between the two. Generally, the application accuracy is lower than I had hoped to be able to achieve for the study – with my initial personal goal being to get a model to at least 90% accuracy on the Streamlit application.



Tabell 1, Confusion Matrix for the Image augmented Logistic Regression Model.



Tabell 2, Confusion Matrix for the RF (Random Forest) model.



Tabell 3, CNN Confusion Matrix

5 Teoretiska frågor

1.

A training set is used in regression analysis as the split of a dataset that is used to train the model on the relationships that exist within the data. It is also at this point that variable selection is undertaken for the model. If splitting a singular dataset then the training set should be the largest, as models value larger volumes of good data to function.

A validation set is used to evaluate the model's performance after training. Here, hyperparameter tuning and model selection are utilized to maximize the performance of the model. If there are issues or inconsistencies that occur between this set and the training set in the analysis, then the cause for these inconsistencies should also be investigated and evaluated.

The test set is the data that the model uses once the model has been fully trained and refined. The results of the test set are what are used to evaluate the 'true' relationship that the analysis was initially interested in. This data needs to be kept separate from the training and validation set to avoid contamination in the model-building process, as its primary goal is to provide an unbiased evaluation of the final model's performance.

2.

K-fold cross-validation process can be utilized to create a validation set that can be used to evaluate the three models for their performance. This technique can be used to fold a dataset into any number of smaller partitions (k) and subsequently use each fold for validation and the remaining folds for training. In this case, it would be wise to split the training set into (k=5) folds and use k-fold cross-validation to then test each of the three models for their average RMSE values over each fold to discover which model performs best.

3.

Regression analysis is a method used to analyse the relationship between a dependent variable and several independent variables. The most basic form of regression is one which describes the linear relationship between a dependent and independent variable, a linear regression. An example of the realistic application of such a model could be the relationship between ice-creams sold and temperature, wherein as temperature rises, so too does ice-cream. Theoretically, this makes sense, but then, how well does a linear regression capture what happens when temperatures reach extreme highs? Would the amount of ice-cream sold at 80^{o}c truly be higher than the ice-creams sold at 30^{o}c? Not likely.

The non-linearity of many relationships has driven the development of more complex regression models. Many regression models are variations of linear regression, but with added caveats that build upon the model to better fit the data. One example that has previously been explored in this course is the Lasso regression model, which can identify significant independent variables and reduce poorly associated variables coefficients within the model. This can be useful because it can reduce the importance of manually fine-tuning which independent variables you might want to include in a model and therefore prevent overfitting.

4.

RMSE (or the Root Mean Squared Error) refers to a metric that can be utilized for measuring how well a model performs at predicting values in regression. When looking at the RMSE value of a regression model, a model is understood to perform better the lower the RMSE is (the gap between the predicted value and the actual value is, on average, lower). This can be especially useful when comparing models, because it allows us to see which model performs better on unseen data. It is also useful when referring to model fit – if a model has a low RMSE on data that is from outside of the original dataset then it has good generalization and therefore is likely not overfit to the training data.

5.

A classification issue refers to how we can use machine learning to predict categorical outcomes, for example – instead of a traditional model predicting income based on certain variables, a classification problem would be interested in what category of income someone falls into based on certain variables. Many models are equipped to deal with both classification and regression, but some are better suited to one or the other. Random forest, for example, is excellent for classification problems.

A confusion matrix can be used to visualize how effective a classification model is at predicting the classification correctly. It is also valuable for seeing where false positives and false negatives are most prevalent within a classification grid. An example from this assignment would be that the confusion matrix was highly valuable for identifying which numbers were often mistaken for each other – for example the difficulty of many models to distinguish between 7 and 9.

6.

K-means refers to a machine learning algorithm that can be utilized for clustering data points based on the characteristics of the data points. It is a good example of a machine learning algorithm that can be unsupervised, as it can simply process data via the model and assign it to the cluster that it most closely resembles. One common example is customers in a bank – one cluster may consist of high-spending, high-income individuals, and another of low-spending, high-income individuals. Both clusters exist in a high-income dimension but are separated by another variable; spending – which the high-income low-spending shares more with a low-income low-spending cluster, therefore developing a distinct cluster from both other groups. It is also important to highlight that these groups are based on means of the group and so can group characteristics can morph over time as well as individuals within that cluster falling in or out of the group based on real-time characteristic changes.

7.

Ordinal coding refers to how we can make categorical data numerical in its form, for example, educational attainment may be ‘Primary School’, ‘High School’, ‘Bachelors’, ‘Masters’, ‘Doctorate’, which can be translated to 1, 2, 3, 4, 5 in order of attainment, wherein 5 now represents the highest level of education and 1 the lowest. The categorical variables must have some kind of hierarchy to them for the variable to have analytical value.

If something does not have an order, for example gender, in a database. Using one-hot encoding would mean that there are two columns in the data, one for male, and one for female. If the row contained a man, then the male column would read ‘1’, and the female column ‘0’.

Dummy variables are very similar to one-hot encoding, but we can use the same example as in the previous paragraph to best understand the difference. In one-hot, there are two columns – one for male, one for female. In dummy variables, we can include just one column, and set either male or female to 1, and use the absence of 1 to represent the alternative gender. This is useful for when you want to avoid multicollinearity in regression.

8.

Julia is correct in this case – in this example, context, and therefore order, has been introduced to the red shirt example. By default, the red/green/blue shirts are not ordered in any ranking. This is true of many variables that can be used in regression. When we add a ‘beauty’ factor to the colour of the shirts, then the shirts can be used in an ordinal form, but that is a transformation of the data and not the default case.

9.

Streamlit is an open-source framework for developing applications with Python. It is especially useful for building applications that involve ML models built in Python because it allows for ML models to be implemented websites with relative ease. It is also easy to develop and edit the application due to being able to re-run the application live as it is edited. All-in-all, it is a relatively accessible tool that is simple to learn and implement ideas into.

6 Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

There were a couple of challenges that I faced during this assignment.

Firstly, I realized during a lecture that I may have cross-contaminated my training/validation sets with my test sets during my analysis. I went over my code and confirmed the case – I had trained the model on the test and validation as usual, but then pulled the test accuracy, before running a GridSearch. I fixed this issue and I believe that my ML flow should be within the correct practices.

Secondly, I struggled particularly with discrepencies in my models performance between the test data and data that was inputted into the Streamlit application. I found that when I was first testing the Streamlit application on a logistic regression model that had scored upwards of 92% accuracy in the test dataset that I was seeing accuracy as low as 40-50%, which was concerning. I spent a lot of time working with image augmentation in the training model as I believed that the issue lay with a logistic regression model struggling to adapt to non-MNIST data. I believe that theoretically this is a good approach, but the issue again lays with how logistic regression functions: data augmentation might actually hinder the model instead of help it, as it 'muddies' the waters and therefore makes the model even less suited to predicting digits.

In the future I would also need to be more careful with my data management and abiding by good ML practices. I believe that everything that I have run in this assignment is now correct, but I also learned a great deal over the course of the assignment. If I could go back and re-build my code from scratch I believe it would be much more organized and the risk for mistakes in the ML pipeline would be much lower.

2. Vilket betyg du anser att du skall ha och varför.

Frankly, I am writing this before I have finished my Streamlit application, and if I haven't managed to build a functional application that boasts something more than a 85% accuracy on my strongest model then probably a G. If I have managed to solve the application by submission, then hopefully a VG! I feel I have at least put the time in to it, but currently what I have to show for it is not living up to the VG grade.

3. Något du vill lyfta fram till Antonio?

Nothing specific. I enjoyed this course, and I feel that the assignment is a fantastic learning tool that forces you to really interact with the data and the models in such a way that you have to understand them to move along. Big assignment though! Sometimes it can feel a bit uncertain about how much is required for the assignment, and I personally like to have boundaries or limits in place for how much is expected.

Källförteckning

Breiman, L. (1996) Bagging Predictors. *Machine Learning.* Vol. 24 pp.123-140. Accessed from: https://link.springer.com/content/pdf/10.1007/BF00058655.pdf (Date Accessed: 21/03/2025)

Geron, A. (2019) Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. Accessed from: hooshio.com (Date Accessed: 20/03/2025)

IBM (2025) What is random forest? Accessed from: https://www.ibm.com/think/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems. (Date Accessed: 20/03/2025)

Lecun, Y., Cortes, C., & Burges, C. (2025) MNIST Website. Accessed from: http://yann.lecun.com/exdb/ (Date Accessed 18/03/2025)

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324. Online Article accessed from: https://ieeexplore.ieee.org/document/726791 (Date Accessed: 19/03/2025)

TensorFlow.org (2024) Datasets/Catalog/MNIST. Online Article accessed from: https://www.tensorflow.org/datasets/catalog/mnist (Date Accessed 20/03/2025)