Convolutional Neural Networks: MNIST

Noemie Lamontagne

August 15, 2023

Abstract

This paper was part of a personal effort to learn more about Machine Learning and computer vision by training a convolutional neural network (CNN) using images as input. Additionally, the paper served to learn how to deploy a CNN model for practical use.

The MNIST database (Modified National Institute of Standards and Technology database) was used as a training and testing dataset. The inputs were images of hand-written numbers from 0 to 9. The outputs were labels or predictions based on the following classes: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Several CNN models were trained and then compared using confusion matrices and classification reports.

Many CNN models were trained and the best model was deployed using Gradio and Hugging Face after achieving good results.

1 Introduction

Within Machine Learning, a convolutional neural network (CNN) is a type of neural network architecture that involves processing pixel data and is often used for image recognition [1]. A CNN involves convolutional and pooling layers that extract features or pixel data from the input image. These features or pixel data can then be reshaped and passed into fully connected layers or linear layers that produce an output. Feature extraction occurs in the convolutional and pooling layers, and classification occurs in the fully connected layers. Feature extraction refers to the process of transforming raw data (such as images) into relevant numerical features that can be processed better than the raw data [2]. Classification is a supervised learning approach in which the computer program learns from the input data (numerical features) and then uses this learning to classify new observations [3].

Using the Modified National Institute of Standards and Technology (MNIST) database, the goal is to produce a simple convolutional neural network (CNN) model that predicts hand-written numbers from 0 to 9 and to deploy the model to be used by the public.

Several CNN models with differing parameters and convolutional, pooling, and fully connected layer architecture were trained. The models were compared using confusion matrices and classification reports. Using the best CNN model, a Machine Learning interface was built and deployed using Gradio and Hugging Face.

2 Dataset

2.1 MNIST

The Convolutional Neural Network was trained using the MNIST database as a dataset. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems [4]. The dataset includes 70,000, 28 x 28 images of handwritten digits from 0 to 9 [4]. 60,000 training images and 10,000 testing images [4]. Each image containing one hand-written digit in grey-scale.

Figure 1: Hand-written digit examples.

3 Models and Methodology

CNN models with different architectures and parameters were trained and then compared using confusion matrices as well as their accuracy, recall, precision, and F1 score. The CNN model with the best results, along with its architecture, was saved to be used in a Machine Learning interface. This interface was later built and deployed using Gradio and Hugging Face.

3.1 CNN Model

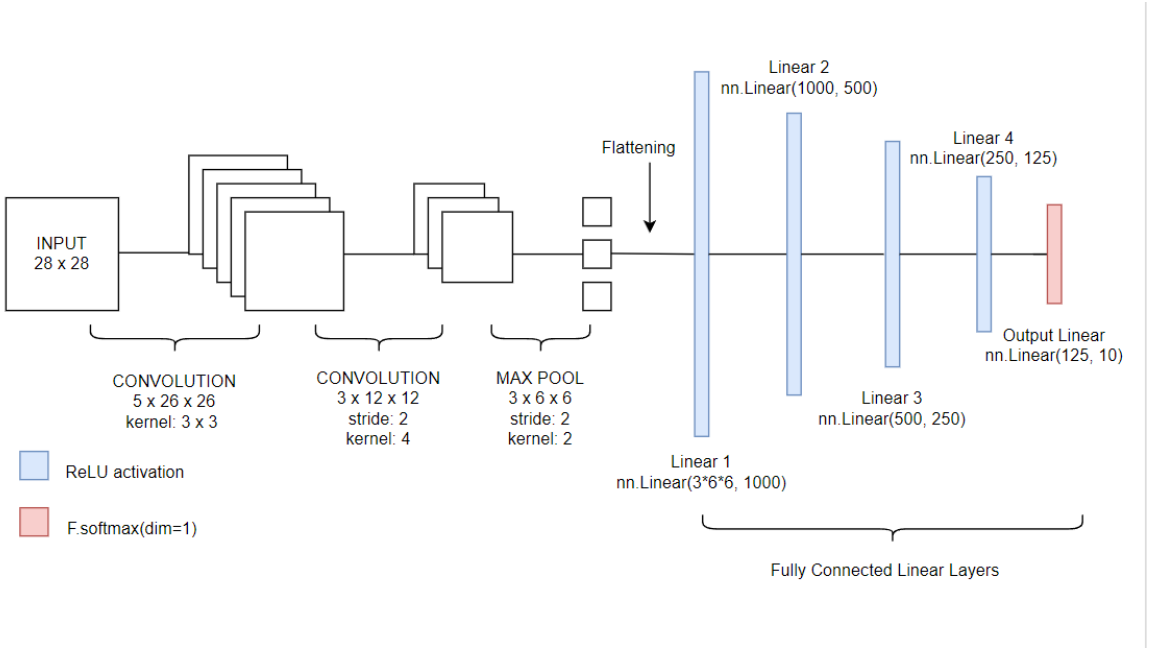

Figure 2: Diagram of CNN model architecture.

The CNN model takes a tensor representing a 28 by 28 greyscale image as input and predicts one of the following 10 classes: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. This CNN model architecture has 2 convolutional layers, 1 pooling layer and 5 fully connected linear layers:

- 1st Convolutional Layer:
  - in_channels = 1
  - out_channels = 5

  - kernel_size = 3
  - Expects a 28 by 28 matrix representing a grey-scale image
  - Passes on 5 26 by 26 matrices to the next convolutional layer
- 2nd Convolutional Layer:
  - in_channels = 5
  - out_channels = 3
  - kernel_size = 4
  - stride = 2
  - Expects 5 26 by 26 matrices
  - Passes on 3 12 by 12 matrices to the pooling layer
- Max-Pooling Layer:
  - kernel_size = 2
  - stride = 2
  - Expects 3 12 by 12 matrices
  - Passes on 3 6 by 6 matrices to the linear layers
  - Before being passed on the matrices are flattened using flatten(start_dim = 1). This means the matrices are flattened by reshaping them into a one-dimensional tensor.
- 1st Linear Layer:
  - Expects 3*6*6 features
  - Passes on 1000 output features
  - ReLU activation
- 2nd Linear layer:
  - Expects 1000 features
  - Passes on 500 output features
  - ReLU activation
- 3rd Linear Layer:
  - Expects 500 features
  - Passes on 250 output features
  - ReLU activation
- 4th Linear Layer:
  - Expects 250 features
  - Passes on 125 output features
  - ReLU activation
- 5th or Output Linear Layer:
  - Expects 125 features
  - Produces 10 output features
  - The 10 output features are passed on to a softmax function which assigns a decimal probability to each of the 10 output features. The result is a list of 10 probabilities for each class. The index of each probability corresponds to each of the 10 classes. Therefore the first probability in the list is the probability for class '0' since its index will be 0.
  - The argmax function can then be used to return the index of the highest probability and therefore returns the predicted class.

A confusion matrix was used for graphical representation of the model's classification performance.

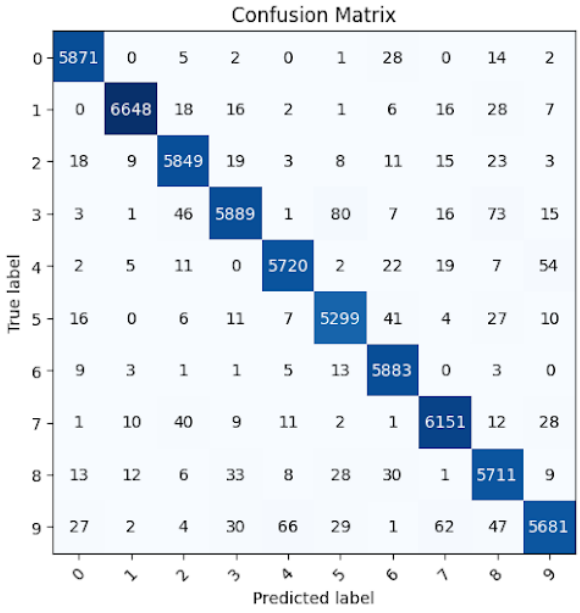

Figure 3: Example of a confusion matrix for the best multi-class CNN model.

4 Results and Discussion

The false positives (FP), true positives (TP), false negatives (FN), and true negatives (TN) for each class are shown in the model's confusion matrix. The confusion matrix gives a graphical representation of the model's performance overall and for each class. The sklearn.metrics module along with the two corresponding lists of predicted labels and true labels were used to measure classification performance.

The 11th CNN model whose architecture is described above produced the best test results.

4.1 Training-Split Results

These are the results from testing model 11 on the training-split.
Accuracy: 0.98
Micro Precision: 0.98
Micro Recall: 0.98
Micro F1-score: 0.98
Macro Precision: 0.98
Macro Recall: 0.98
Macro F1-score: 0.98
Weighted Precision: 0.98
Weighted Recall: 0.98
Weighted F1-score: 0.98

Table 1: Classification report for the testing-split

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
0	99	99	99	5923
1	99	99	99	6742
2	98	98	98	5958
3	98	96	97	6131
4	98	98	98	5842
5	97	98	97	5421
6	98	99	98	5918
7	98	98	98	6265
8	96	98	97	5851
9	98	95	97	5949
accuracy			98	60000
macro avg	98	98	98	60000
weighted avg	98	98	98	60000

4.2 Testing-Split Results

These are the results from testing model 11 on the testing-split.
Accuracy: 0.97
Micro Precision: 0.97
Micro Recall: 0.97
Micro F1-score: 0.97
Macro Precision: 0.97
Macro Recall: 0.97
Macro F1-score: 0.97
Weighted Precision: 0.97
Weighted Recall: 0.97
Weighted F1-score: 0.97

Table 2: Classification report for the testing-split

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
0	97	99	98	980
1	99	99	99	1135
2	98	98	98	1032
3	98	96	97	1010
4	98	97	97	982
5	96	98	97	892
6	98	98	98	958
7	98	97	97	1028
8	95	97	96	974
9	97	94	96	1009
accuracy			97	10000
macro avg	97	97	97	10000
weighted avg	97	97	97	10000

4.3 Discussion

Support is the number of actual occurrences of the class in the specified dataset. The classification report shows a relatively well-balanced occurrence for each class within the testing-split and training-split of the MNIST dataset as there are no obvious imbalances. Additionally, the classification reports show good testing results.

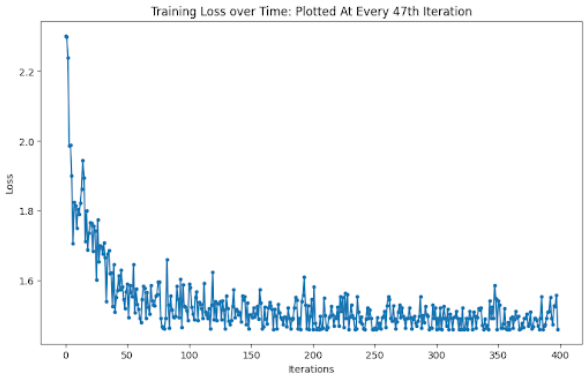

Figure 4: CNN Loss Function.

The graph of the loss function shows that the model learned well over time. However, as the iterations or epochs increase, the loss eventually plateaus, and improvement becomes sporadic or minimal.

Considering the good testing results from model 11, the model was deployed to a wider audience using Gradio on Hugging Face. Gradio is an open-source Python package that allows you to quickly create an easy-to-use, user interface for your Machine Learning model as well as deploy it. Hugging Face is a website that allows users to share and use Machine Learning models and interfaces.

5 Conclusion

A CNN model for hand-written digit recognition was successfully trained and deployed for public use. The model is available for use as a space (interface on hugging face called NumberClassification.

It can be found at this link:
https://huggingface.co/spaces/NoemieL/NumberClassification

It is possible that the interface may not be accessible if it has been paused due to inactivity.

6 References

[1] Awati, R. (2023, April 24). What are convolutional neural networks?: Definition from TechTarget. Enterprise AI. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network#:~:text=A%20CNN%20is%20a%20kind,the%20network%20architecture%20of%20choice.

[2] Feature extraction explained. Explained - MATLAB & Simulink. (n.d.). https://www.mathworks.com/discovery/feature-extraction.html#:~:text=Feature%20extraction%20refers%20to%20the,directly%20to%20the%20raw%20data.

[3] Sidana, M. (2020, April 9). Intro to types of classification algorithms in machine learning. Medium. https://medium.com/sifium/machine-learning-types-of-classification-9497bd4f2e14#:~:text=In%20machine%20learning%20and%20statistics,learning%20to%20classify%20new%20observations.

[4] Wikimedia Foundation. (2023, August 8). MNIST database. Wikipedia. https://en.wikipedia.org/wiki/MNIST_database#:~:text=The%20MNIST%20database%20(Modified%20National,the%20field%20of%20machine%20learning.