Digit recognizer: computer vision fundamentals

Bavo Maes

Erasmushogeschool Brussel

Abstract

In deze paper wordt de Kaggle uitdaging 'Digit Recognizer' [1] onderzocht en enkele mogelijke oplossingen aangeboden. In deze uitdaging wordt gevraagd om een zo accuraat mogelijke oplossing te creëren voor het herkennen van handgeschreven letters. Voor deze letters wordt gebruikt gemaakt van de MNIST dataset. Als eerste wordt uitgegaan van dat het probleem een simpel classificatieprobleem is, waarbij er gebruik gemaakt wordt van twee verschillende algoritmen: '*decision tree*' en '*K-nearest neighbors*'. Er wordt gebruik gemaakt van '*K-fold cross validation*' om de nauwkeurigheid van het model correct in te schatten. Bij het decision tree algoritme wordt een beste gemiddelde nauwkeurigheid van 86,34% bereikt. Bij het K-nearest neighbors algoritme wordt een beste gemiddelde nauwkeurigheid van 87,96% bereikt. Vervolgens wordt ook een mogelijke oplossing aangeboden via een neuraal netwerk dat geïmplementeerd is met Keras. Hierbij wordt een gemiddelde nauwkeurigheid van 95,83% bereikt.

1. Introductie

De Kaggle uitdaging genaamd 'Digit Recognizer' is een online uitdaging waarbij de basisprincipes van Machine Learning op proef worden gesteld, door de gebruiker te vragen om een model te ontwikkelen dat handgeschreven letters kan classificeren.

2. Data

Er wordt gebruik gemaakt van een vooraf bepaalde selectie uit de MNIST ("Modified National Institute of Standards and Technology") dataset [2]. De data is opgesplitst in een set die dient als training voor het model, en een aparte set als dataset om te testen. De dataset om te testen heeft geen labels, enkel de dataset die dient om het model te trainen is voorzien van labels die de waarde van afgebeelde cijfer beschrijven. Beide datasets worden aangeleverd in een CSV-bestand. De afgebeelde cijfers worden weergegeven als een afbeelding van 28 op 28 pixels. De dataset bestaat echter niet uit deze afbeeldingen, maar uit numerieke waarden de kleurintensiteit van een bepaalde pixel voorstelt. Deze waarden liggen steeds tussen 0 en 255. Voor elk handgeschreven cijfer zijn er dus 784 waarden gegeven. Er zijn 42.000 handgeschreven cijfers aanwezig in de trainingsset. In de dataset om te testen zijn er 28.000 handgeschreven cijfers aanwezig.

3. Methode

Om dit probleem op te lossen zijn er twee verschillende redeneringswijzen mogelijk.

1. Aangezien alle cijfers een unieke vorm hebben, kan er van uitgegaan worden dat er sprake is van een klassiek classificatieprobleem waarbij het verschil in handschrift van de cijfers weinig invloed zal hebben op het onderverdelen van de cijfers.
2. Handschrift zal een grote invloed hebben op de interpretatie van de cijfers waardoor een klassiek classificatie algoritme een lagere nauwkeurigheid zal hebben dan een oplossing die gebruikt maakt van deep learning.

Als eerste werd er vanuit gegaan dat klassieke classificatie algoritmen het probleem zouden kunnen oplossen. Om de scope van het project te beperken werd er gekozen voor een uitwerking

met 2 verschillende algoritmes: het decision tree algoritme en het K-nearest neighbors algoritme. Aangezien het om een grote dataset gaat besloot ik om rekenkracht te besparen en het model te trainen op de eerste 21.000 records (de helft van de totale grootte) van de training dataset. In de testfase liet testte ik het model op de tweede helft van de dataset, maar na feedback op het project besloot ik om gebruik te maken van K-fold cross validation. Hierbij wordt de dataset opgesplitst in een vast aantal delen, en vervolgens wordt het model getraind op alle delen buiten het deel dat dient om te testen. Dit wordt gedaan tot als elk deel een testset is geweest, om zo de gemiddelde nauwkeurigheid van het model te bepalen.
Het decision tree algoritme werd op twee manieren getest: met 3 folds en met 5 folds. De resultaten hiervan zijn terug te vinden in tabel 1 en tabel 2. Bij dit algoritme werd een beste gemiddelde score van 86,34% bereikt, bij 5 folds. Bij het K-nearest neighbors algoritme werd ook de invloed van het aantal neighbors (variabele k) getest op de nauwkeurigheid. De resultaten hiervan zijn terug te vinden in tabel 3 tot en met tabel 8. Bij dit algoritme werd een beste gemiddelde score van 87,96% behaald, met 3 folds en 10 als waarde voor de variabele k. Deze resultaten zijn terug te vinden in tabel 7. Deze scores zijn niet uitzonderlijk hoog, dus werd ook de tweede denkpiste uitgewerkt. Er werd een neuraal netwerk opgezet [3] bestaande uit 3 lagen. De ingangslaag hiervan heeft 784 neuronen (voor 28 x 28 pixels), en de uitgangslaag heeft 10 neuronen (aantal mogelijkheden van handgeschreven cijfer). De middenste laag wordt als variabele gebruikt, samen met de batch size en het aantal epochs. De resultaten hiervan zijn terug te vinden in tabel 9 tot en met tabel 11. De beste nauwkeurigheid was 96,52% met 30 neuronen als middenste laag, 20 epochs en een batch size van 500, terug te vinden in tabel 10.

4. Resultaten

Tabel 1: Decision tree, criterion = "gini", splitter = "best", folds = 3

Fold	Nauwkeurigheid
1	86,05%
2	86,39%
3	86,37%

Gemiddelde nauwkeurigheid: 86,27%

Tabel 2: Decision tree, criterion = "gini", splitter = "best", folds = 5

Fold	Nauwkeurigheid
1	86,19%
2	86,19%
3	86,34%
4	86,32%
5	86,68%

Gemiddelde nauwkeurigheid: 86,34%

Tabel 3: K-nearest neighbors, folds = 5, k = 3

Fold	Nauwkeurigheid
1	88,69%
2	87,05%
3	86,26%
4	86,39%
5	85,27%

Gemiddelde nauwkeurigheid: 86,73%

Tabel 4: K-nearest neighbors, folds = 5, k = 10

Fold	Nauwkeurigheid
1	87,82%
2	85,86%
3	85,43%
4	84,69%
5	83,53%

Gemiddelde nauwkeurigheid: 85,33%

Tabel 5: K-nearest neighbors, folds = 5, k = 25

Fold	Nauwkeurigheid
1	82,83%
2	81,20%
3	80,77%
4	79,33%
5	79,78%

Gemiddelde nauwkeurigheid: 80,78%

Tabel 6: K-nearest neighbors, folds = 3, k = 3

Fold	Nauwkeurigheid
1	88,37%
2	87,80%
3	86,94%

Gemiddelde nauwkeurigheid: 87,70%

Tabel 7: K-nearest neighbors, folds = 3, k = 10

Fold	Nauwkeurigheid
1	87,96%
2	88,32%
3	87,59%

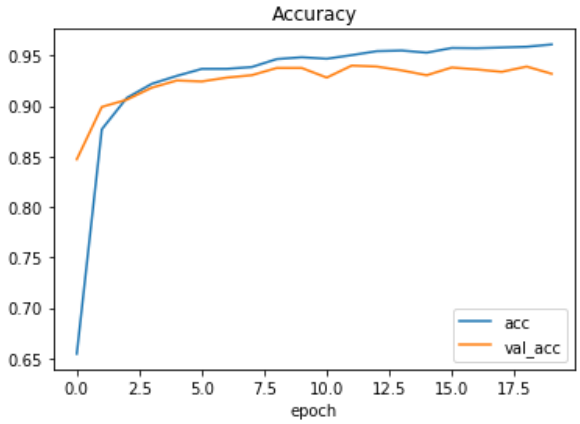
Gemiddelde nauwkeurigheid: 87,96%

Tabel 8: K-nearest neighbors, folds = 3, k = 25

Fold	Nauwkeurigheid
1	87,25%
2	88,23%
3	87,86%

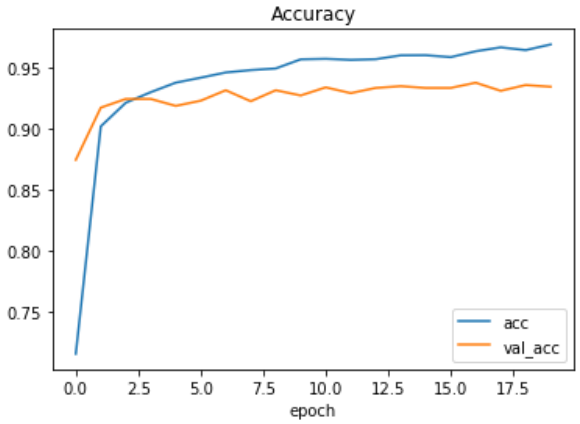
Gemiddelde nauwkeurigheid: 87,78%

Tabel 9: Neural network, middenste laag 30 neuronen, epochs = 20, batch size = 200, validation split = 0.1



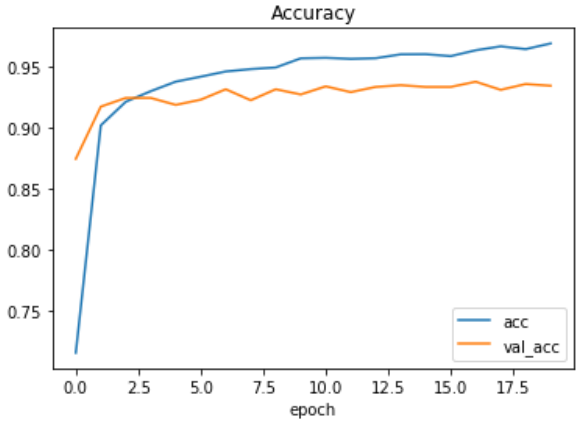
Nauwkeurigheid: 95,25%

Tabel 10: Neural network, middenste laag 30 neuronen, epochs = 20, batch size = 500, validation split = 0.1



Nauwkeurigheid: 96,52%

Tabel 11: Neural network, middenste laag 50 neuronen, epochs = 10, batch size = 1000, validation split = 0.1



Nauwkeurigheid: 95,25%

5. Conclusie

De hoogste nauwkeurigheid die werd behaald was 95,83%, aan de hand van een neuraal netwerk met 20 epochs en een batch size van 200. Deze test maakt duidelijk dat handschrift een duidelijke invloed heeft op het succes of falen van het herkennen van een handgeschreven cijfer, en daarom voor dit soort computer vision toepassing beter wordt gewerkt met een neuraal netwerk dan klassieke classificatie algoritmen.

6. Future work

Een hogere score kan waarschijnlijk behaald worden door het uittesten van neurale netwerken met meer als 3 lagen. Ook kan dit experiment nog uitgebreid worden door andere klassieke classificatie algoritmen te testen.

7. Referenties

[1] Kaggle. (2019). Digit Recognizer, https://www.kaggle.com/c/digit-recognizer

[2] Yann LeCun, Corinna Cortes, Christopher J.C. Burges. (2020). "The MNIST database of handwritten digits", http://yann.lecun.com/exdb/mnist/

[3] Özbek, Abdullah Furkan. (2019). How to Train a Model with MNIST dataset. https://medium.com/@afozbek_/how-to-train-a-model-with-mnist-dataset-d79f8123ba84