Neural Network implementation

Daan Zimmerman van woesik & Noah titarsole

**Call tree**

Main

….tf.keras.datasets.mnist.load\_data

….reshape

….reshape

….astype

….astype

….sequential.\_\_init\_\_

….add

….add

….add

….add

….add

….add

….compile

….fit

….evualuate

**Documentation**

Eerst laden we de mnist dataset op met de functie **datasets.mnist.load\_data()**. Deze functie haalt de data op als (x\_train, y\_train), (x\_test, y\_test). Hierin is x de foto en y het daadwerkelijke getal. Daarnaast is de train de training\_set en test de test\_set.

Ten eerste moeten we de training\_set en test\_set **reshapen** naar 28x28x1 foto’s. Anders kan je errors krijgen als sommige fotos niet goed zijn. Daarna **normalisen** we de data door een keras.utils functie waardoor alle waardes in de foto’s tussen de 0 en 1 zitten. Hierdoor werkt het NN beter.

Voor het model gebruik ik een sequential model, dit is het basis model die ik gebruik. Voor de eerste layer de functie **Flatten.** Deze functie heb je nodig om de inputs te krijgen voor je nodes. De pixels in een foto worden hierbij omgezet naar input nodes. Daarna gebruik ik de **Dense** functie en dit is eigenlijk een functie waar je aangeeft hoeveel nodes er in een layer zitten. Ook zullen al deze nodes verbonden zijn met elkaar. Daarnaast staat er ook nog een activation functie. tf.nn.relu is een lineaire lijn die alleen vanaf 0 begint met stijgen. Voor de eerste laag heb ik 196 neurons staan. Dit heb ik gedaan want elke foto is 28x28=784 pixels, dan heb ik die 784 gedeeld door 4 en dan krijg je 196 nodes. Voor de tweede layer heb ik wederom gedeeld door 4 en dan krijg je 49 nodes. Voor de laatste layer gebruik ik 10 neurons. Dit is nodig omdat je maar 10 cijfers hebt en ik gebruik hiervoor geen tf.nn.relu activation voor maar softmax. De softmax activation functie zorgt ervoor dat onder de 0.5 een 0 uitkomt en erboven een 1. Dit is handig voor clasificatie.

Daarna **compile** ik het model met de optimizer adam omdat snel en goed leert. Daarna gebruik ik de sparse\_categorical\_crossentropy loss parameter. Dit geeft ons een loss per categorie. Ten slotte de metrics geven we de accuracy mee. We willen namelijk weten hoe accuraat ons model is dus moeten we dat meegeven.

Nu gaan we ons model **fitten** en dit is kort gezegd het trainen van ons NN. Dit doen we 5 verschillende keren voor een goed resultaat.

Daarna bekijk ik de **summary** waarin we bepaalde specs van ons model kunnen zien. Dit heb ik gebruikt om te kijken of ik het goed deed.

Ten slotte gebruiken we de **evaluate** functie om te kijken hoe goed ons model is. Hiervoor gebruiken we de test dataset. Het antwoord hiervan printen we.