Universiteit Utrecht

De prijs van Bitcoin voorspellen met neurale netwerken

Een onderzoek naar de nauwkeurigheid van een standaard neuraal netwerk, een Long Short-Term Memory netwerk en een Gated Recurrent Unit netwerk

Saïd el Khouani

3538060

Kunstmatige Intelligentie

Bachelor scriptie

20-08-2019

Begeleider: dr. M. van Ommen

Abstract

Cryptovaluta zijn een populair onderwerp voor wetenschappelijk onderzoek, waarbij getracht wordt de prijs van een cryptovaluta te voorspellen op basis van machine learning. De laatste jaren wordt er steeds meer onderzoek gedaan om deze voorspellingstaak te vervullen met behulp van neurale netwerken.

Het huidig onderzoek richt zich specifiek op het voorspellen van een niet-lineaire tijdreeks, de koers van Bitcoin. Deze voorspellingstaak wordt met drie verschillende typen neurale netwerken uitgevoerd: een standaard neuraal netwerk en twee typen recurrent neurale netwerken, een Long Short-Term Memory netwerk en een Gated Recurrent Unit netwerk. Het onderzoek bestaat uit een experimenteel onderzoek dat theoretisch onderbouwd is door een literatuuronderzoek. Het experimenteel onderzoek maakt gebruik van twee datasets en drie externe variabelen (*Market Capitalization*, *Payment Count* en *Realized Capitalization)* om de prijs van de populairste cryptovaluta (Bitcoin) tussen de periode van 23 november 2018 en 10 juni 2019 te voorspellen. Het onderzoek analyseert hoe nauwkeurig de prestaties van standaard en de twee typen recurrent neurale netwerken zijn in het voorspellen van de prijs van cryptovaluta's en of deze prestatie verbetert kan worden met het gebruik van externe variabelen. Het onderzoek toont aan dat een Long Short-Term Memory netwerk het best presteert in deze voorspellingstaak, hoewel het Gated Recurrent Uni netwerk een vergelijkbare prestatie levert. Het standaard neurale netwerk presteert het minst nauwkeurig in het voorspellen van de Bitcoin prijs. Het onderzoek toont verder aan dat het gebruik van een feature de nauwkeurigheid van de voorspelling in elk van de gebruikte netwerken verbetert.

Lijst met afkortingen

BTC – Bitcoin

GRU – Gated Recurrent Units

LSTM – Long Short-Term Memory

NN – Neural Network

RNN – Recurrent Neural Network

Inhoudsopgave

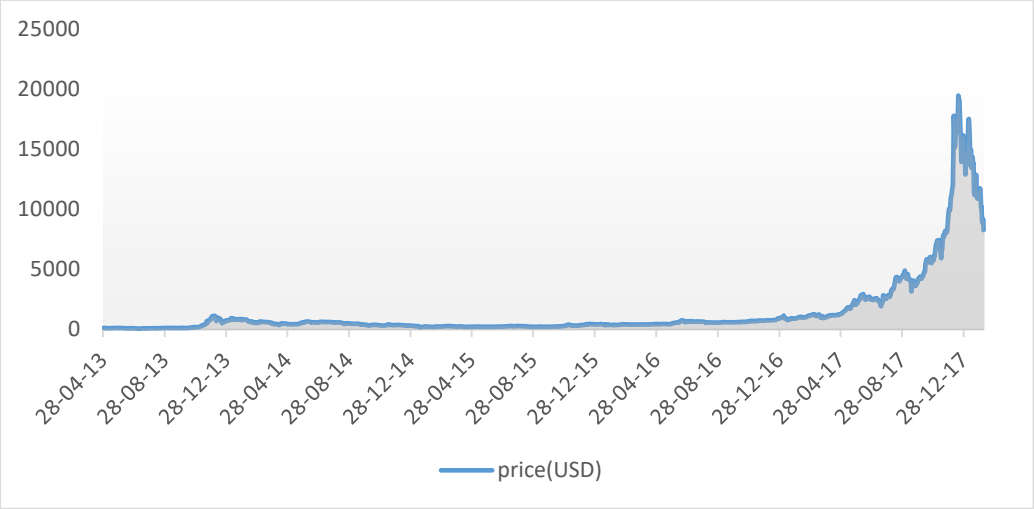
1 Introductie

Dit hoofdstuk geeft een overzicht van de gehele scriptie, waarin het scriptieonderwerp cryptovaluta en de motivatie voor het kiezen van dit onderwerp besproken wordt. Vervolgens wordt het doel van het onderzoek en daarbij horende hoofd- en deelvragen toegelicht. Ten slotte wordt het plan van aanpak en de structuur van de scriptie beschreven.

1.1 Inleiding

Cryptovaluta's (Engels: "cryptocurrencies") zijn een vorm van digitale valuta. Deze valuta werkt op basis van cryptografie waarbij op een veilige en sterk versleutelde manier transacties gecontroleerd en verzonden worden zonder tussenkomst van een derde persoon. Deze gedecentraliseerde manier van bankieren is uitgevonden door een onbekend persoon met de pseudoniem Satoshi Nakamoto, die de eerste gedecentraliseerde cryptovaluta Bitcoin heeft uitgevonden (Nakamoto, 2008). Dit digitaal systeem was ontwikkeld om partijen de mogelijkheid te bieden valuta rechtstreeks naar elkaar toe te zenden op basis van een minimale structuur zonder dat transacties via een centrale autoriteit moeten (Nakamoto, 2008).

De cryptovaluta markt bereikte een hoogtepunt ter waarde van 800 miljard dollar in het begin van januari 2018, waarvan Bitcoin ruim 330 miljard dollar waard was (CoinMarketCap). Op het moment van het schrijven van deze scriptie is de markt kalmer geworden en schommelt de marktwaarde van Bitcoin rond de 140 miljard dollar. Omdat deze markt snelle prijsschommelingen meemaakt, is het geliefd onder speculanten die trachten hun fortuin op deze markt te maken.



Figuur 1.1. Grafiek van de prijs van Bitcoin in USD tussen de periode van 28 april 2013 en 5 februari 2018.

De cryptomarkt heeft enkele duidelijke verschillen van de traditionele aandelenmarkt. Cryptovaluta's zijn bijvoorbeeld erg bewegelijk in koers, omdat de waarde van cryptovaluta zeer speculatief is en mensen niet goed een waarde kunnen geven aan deze valuta (DeVries, 2016). De prijs is gebonden aan wat mensen verwachten dat de prijs is, waardoor er grote schommelingen in prijs kunnen zijn. Een verschil met de traditionele beurshandel is dat aandelen een percentage in een bedrijf representeren en daarmee een stuk tastbaarder zijn dan cryptovaluta (DeVries. 2016). Een cryptovaluta kan namelijk met verschillende bedoelingen gestart worden, waaronder fondsenwerving, als token voor het kopen van specifieke producten als games en nieuws of als betaalmiddel om content online te plaatsen. Daarnaast is de markt

amper tot niet gereguleerd, waardoor geruchten, fraude, manipulatie, handel met voorkennis en bedrog geen consequenties hebben voor mensen die slecht gezind zijn. Cryptomarkten zijn bovendien kwetsbaar voor hackers, waarbij een geslaagde hack de koers kan laten crashen. Dit was het geval bij de Japanse cryptomarkt Mt. Gox, die in februari 2014 gehackt was (Ledger, 2019). Deze markt had toentertijd ruim 70 procent van alle Bitcoin transacties in handen, totdat ongeveer 850.000 bitcoins gestolen werd. De reactie van de markt deed de waarde van Bitcoin ruim 40 procent dalen in enkele weken. Daarnaast is een belangrijk verschil dat op de cryptomarkt continu gehandeld kan worden. Dit houdt in dat transacties altijd kunnen plaatsvinden, in tegenstelling tot de traditionele beurshandel.

Onderzoek naar mogelijkheden om financiële markten te voorspellen valt onder de noemer tijdreeksen voorspellingsprobleem (Engels: “time-series prediction). In de financiële sector wordt dit type probleem steeds vaker met behulp van machine learning getracht op te lossen, waarbij de laatste decennia meer gekeken wordt naar het gebruik van neurale netwerken als mogelijke oplossing voor dit probleem (Kaastra & Boyd, 1996). Het onderwerp tijdreeksen zal verder toegelicht worden in hoofdstuk 2.

1.2 Trends in tijdreeks voorspellingsmodellen

Voor het voorspellen van tijdreeksen kunnen lineaire modellen zoals Autoregressive integrated moving average (ARIMA) gebruikt worden (Box & Jenkins, 1976). Deze statistische modellen maken gebruik van historische prijsdata waarbij de voorspelling geschied met een lineaire functie of een variatie hiervan. Resultaten van deze modellen zijn goed beoordeeld en worden nog steeds veel gebruikt (Abonazel & Abd-Elftah, 2019; Bakar & Rosbi, 2017; Dritsaki, 2015), hoewel neurale netwerken betere prestaties lijken te leveren (Adebiyi, Adewumi & Ayo, 2014; Siami-Namini & Namin, 2018; Yao, Tan & Poh, 1999). De veelbelovende resultaten die neurale netwerken bieden hebben ertoe geleid dat er al veel onderzoek gedaan is naar het voorspellen van tijdreeksen met dit type netwerk (Siami-Namini & Namin, 2018; McNally et al., 2018; Ly et al. 2018; Kaastra & Boyd, 1996; Kara et al. 2011).

Het vinden van patronen in financiële tijdreeksen met behulp van neurale netwerken gebeurt al sinds de jaren 90. Kamijo en Tanigawa (1990) voerden met recurrent neurale netwerken een onderzoek uit om patronen te voorspellen in beursprijzen, waarbij dit succesvol bleek te zijn in 15 van 16 experimenten. Er is veel literatuur te vinden over het toepassen van neurale netwerken op financiële tijdreeksen. Hoewel cryptovaluta relatief jong zijn, is er relatief veel literatuur te vinden op het gebied van prijsvoorspellingen. Kim et al. (2016) analyseerden bijvoorbeeld berichten op het internet over drie cryptovaluta’s, om de koers van deze valuta’s te voorspellen. Ly et al. (2018) gebruikten een standaard neuraal netwerk op de prijs van Bitcoin te voorspellen. Phaladisailoed & Numnonda (2018) en McNally et al. (2018) deden onderzoek naar de prestaties van verschillende machine learning modellen om de prijs van Bitcoin te berekenen.

Na het bestuderen van literatuur van de afgelopen jaren lijkt de voorkeur voor een algoritme verschoven te zijn van tree-based algoritmes (waaronder Random Forest) naar neurale netwerken **(**McNally, Roche & Caton, 2018; Borovykh et al., 2019). Dit kan deels komen omdat er in de wetenschap veel onderzoek wordt gedaan naar nieuwe toepassingen van algoritmes en neurale netwerken relatief nieuw zijn in het gebruik van het voorspellen van financiële tijdreeksen. Daarnaast is de snelheid van een neuraal netwerk gunstiger dan die van tree-based algoritmes (Buciluǎ, Caruana, & Niculescu-Mizil, 2006) en andere algoritmes (Kara et al., 2011) met een hogere nauwkeurigheid. In het bijzonder lijken recurrent neurale netwerken (RNNs) en RNN varianten zoals Long Short-Term Memory (LSTM) en Gated Recurrent

Units (GRU) een hoge nauwkeurigheid te behalen in prijsvoorspellingen van valuta's die grote prijsveranderingen ondergaan (Hsu, 2017; Phaladisailoud & Numnonda, 2018; McNally, Roche & Caton, 2018), hoewel andere onderzoekers stellen dat de nauwkeurigheid van voorspellingen niet beter presteert dan eenvoudigere modellen (Torres & Qiu, 2018). In deze scriptie ligt de focus daarom op onderzoek naar het voorspellen van de prijs van Bitcoin. Dit wordt gedaan aan de hand van meerdere typen neurale netwerken.

1.3 Onderzoeksvraag

Het doel van dit onderzoek is om te analyseren hoe nauwkeurig de prestaties van standaard en twee typen recurrent neurale netwerken zijn in het voorspellen van de prijs van Bitcoin en of deze prestatie verbetert kan worden met het gebruik van externe variabelen (Engels: features). Features zijn variabelen die bepaalde kenmerken uitdrukken, zoals het aantal Bitcoin transacties die uitgevoerd zijn of de totale marktwaarde van Bitcoin op een dag. Dit wordt verder toegelicht in paragraaf 5.3.

Om dit doel te kunnen bewerkstelligen zijn twee deelvragen opgesteld die bijdragen aan het beantwoorden van de hoofdvraag.

- Eerst wordt onderzocht hoe recurrent neurale netwerken presteren in de prijsvoorspelling van Bitcoin ten opzichte van een standaard neuraal netwerk. Hierbij worden twee typen recurrent neurale netwerken gebruikt, Long Short-term memory (LSTM) en Gated recurrent Units (GRU).
- Vervolgens wordt er, gebaseerd op literatuuronderzoek en een experimenteel onderzoek, geanalyseerd of het gebruik van features de nauwkeurigheid van een standaard en recurrent neuraal netwerk kan verbeteren. Features zijn variabelen die bepaalde kenmerken uitdrukken, zoals het aantal Bitcoin transacties die uitgevoerd zijn of de totale marktwaarde van Bitcoin op een dag. Het doel is te onderzoeken of kenmerken van een cryptovaluta kunnen bijdragen aan een hogere nauwkeurigheid van een (recurrent) neuraal netwerk bij de prijsvoorspelling van Bitcoin.

Deze vragen zullen beantwoord worden aan de hand van literatuuronderzoek en een hierop voortbouwend experimenteel onderzoek. Het literatuuronderzoek legt een theoretische basis door verschillende onderzoeken te analyseren. Deze onderzoeken leggen de focus op prijsvoorspellingen van verschillende tijdsreeksen, waaronder traditionele valuta's en cryptovaluta's, aan de hand van neurale netwerken. Het doel van het literatuuronderzoek is om meer kennis te vergaren over de werking van standaard en recurrent neurale netwerken, te begrijpen welke keuzes onderzoekers hebben genomen om modellen op te stellen en welke problemen zij aantroffen bij voorspellingstaken. Met het experimenteel onderzoek wordt vervolgens getracht antwoord te geven op de bovenstaande onderzoeksvragen.

Het experimenteel onderzoek bestaat uit vier experimenten. In alle experimenten worden met een standaard en twee typen recurrent neurale netwerken prijsvoorspellingsmodellen opgesteld. Deze modellen worden vervolgens getraind en gevalideerd, waarop de prestaties van deze drie modellen onderling vergeleken worden.

De modellen in het eerste experiment maken gebruik van een dataset die gelijk is aan de dataset van het onderzoek van Ly et al. (2018). In dit experimenteel onderzoek is een standaard neuraal netwerk gebruikt om de prijs van Bitcoin van een week later te voorspellen op basis van de afgelopen vijf dagen. Deze dataset bestaat enkel uit de dagprijzen van Bitcoin tussen de periode 28 april 2013 en 05 februari 2018. In het tweede experiment wordt gebruik gemaakt van een dataset over dezelfde periode als in het

eerste experiment. Het verschil met het eerste experiment is dat er telkens een feature is toegevoegd aan de dataset. Het doel is om per gebruikte feature de toe- of afname van de nauwkeurigheid in voorspelling te analyseren ten opzichte van het eerste experiment.

Het derde en vierde experiment maken gebruik van een dataset die over een langere periode loopt dan de eerste twee experimenten, namelijk van 28 april 2013 tot 10 juni 2019. Deze dataset loopt hiermee ruim een jaar langer door dan de gebruikte datasets van de eerste twee experimenten. De reden hiervoor is dat de dataset van Ly et al. een grote prijsstijging aan het einde bevat (figuur 1.1), waardoor de prestaties van de gebruikte neurale netwerken mogelijk vertekend raken. In het derde experiment worden, net als het eerste experiment, enkel de dagprijzen gebruikt om de modellen te trainen. In het vierde experiment wordt een feature toegevoegd aan de dataset van het derde experiment om de toe- of afname van nauwkeurigheid in voorspelling te analyseren ten opzichte van het derde experiment.

Door de resultaten van het experimenteel onderzoek te koppelen aan de bestaande literatuur wordt getracht de hoofdvraag te beantwoorden.

In hoofdstuk 2 wordt uitgelegd wat tijdreeksen zijn en welk wetenschappelijk onderzoek er is uitgevoerd naar het voorspellen van tijdreeksen, hierbij wordt er gekeken naar verschillende patronen in tijdreeksen en het voorspellen van tijdreeksen. In hoofdstuk 3 wordt de architecturen van verschillende typen neurale netwerken belicht. In hoofdstuk 4 wordt er vervolgens gekeken naar voorgaande onderzoeken op het gebied van tijdreeksvoorspellingen met behulp van neurale netwerken. Dit hoofdstuk koppelt zodoende de informatie van hoofdstuk 2 en 3. Het experimentele onderzoek wordt in hoofdstuk 5 verder toegelicht. Hierna volgen de resultaten in hoofdstuk 6. Een analyse van deze experimentele resultaten wordt in hoofdstuk 7 gekoppeld aan het theoretisch kader, dit is de conclusie. De scriptie wordt afgesloten met de discussie in hoofdstuk 8, waar zowel de beperkende factoren van het onderzoek als de suggesties voor vervolgonderzoek aan bod komen.

2 Tijdreeksen

Een tijdreeks is een verzameling observaties van een variabele in een groep tijdstippen (Taylor, 2008). Deze tijdstippen hoeven niet even ver van elkaar af te staan. In dit onderzoek wordt er, wanneer er over tijdreeksen gesproken, vanuit gegaan dat de tijdstippen even ver van elkaar af liggen. De afstand tussen tijdstippen wordt een tijdstap genoemd.

Een univariate tijdreeks verzameling X bestaat uit t tijdstippen, waarvan elk tijdstip in deze verzameling weergeven wordt als een vector $X = x_1, x_2, \dots, x_{t-1}, x_t$. In het voorspelling van de prijs van Bitcoin zou een tijdstip x bestaan uit de prijs. Als er van een multivariate tijdreeks verzameling X wordt gesproken, bestaat ieder tijdstip uit meerdere variabelen (Engels: features). Deze verzameling tijdstippen wordt weergeven als een matrix X (Figuur 2.1).

$$\begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^{v-1} & x_1^v \\ x_2^1 & x_2^2 & \cdots & x_2^{v-1} & x_2^v \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{t-1}^1 & x_{t-1}^2 & \cdots & x_{t-1}^{v-1} & x_{t-1}^v \\ x_t^1 & x_t^2 & \cdots & x_t^{v-1} & x_t^v \end{bmatrix}$$

Figuur 2.1. Een multivariate tijdreeks verzameling weergegeven in een matrix met *t* tijdstippen en *v* features. Rechtsboven elk getal in de matrix stelt de kolomwaarde (feature) voor, rechtsonder elk getal wordt het tijdstip aangegeven.

Hier staat v voor het aantal features. Het getal rechtsboven elke waarde geeft de kolom en feature aan, het getal rechtsonder geeft het tijdstip aan. Als het getal rechtsboven een 1 is en deze kolom de feature prijs bevat (x_t^1), stelt dit de laatste prijs in de tijdreeks weer. De waarden $x_t^1, x_t^2, \dots, x_t^{v-1}, x_t^v$ stellen de features van tijdstip t voor.

Er wordt getracht een voorspelling te maken op basis van historische data over een tijdstip x_{t+n}, waarbij n het aantal dagen in de toekomst is waarvan de prijs voorspeld moet worden. Het verschil tussen een voorspelde prijs $\hat{x}_{t+n}$ en daadwerkelijke prijs x_{t+n} wordt hierop berekend om een uitspraak te doen over de effectiviteit van een model middels de voorspellingsfout (Engels: prediction error) $e_{t+n} = x_{t+n} - \hat{x}_{t+n}$. Op basis van deze voorspellingsfout wordt een aanpassing in het model teruggekoppeld om een betere prestatie te verkrijgen.

2.1 Patronen in tijdreeksen

Binnen de literatuur is veel informatie te vinden over patronen in tijdreeksen. Brockwel & Davis (2002) spreken onder andere over trends, seizoensgebonden patronen en cyclische patronen. Er wordt over een trend gesproken als er een toe- of afname is in waarde is over een langere tijd, waarbij deze verandering niet lineair hoeft te zijn. Als de waarden in een reeks toe- of afnemen, wordt er respectievelijk gesproken van een positieve of een negatieve trend. Er wordt over een seizoen patroon gesproken als seizoensgebonden factoren de toe- of afname van waarden beïnvloeden. Hier moet gedacht worden aan tijdstappen zoals een kwartaal, maand, weekdag of uur van de dag, waarbij het patroon een herhaalbaar effect toont op de waarden. Een cyclisch patroon strekt zich, net als een trend, eveneens over een langere periode. Het verschil hier is dat een cyclisch patroon geen herhaald effect toont op bepaalde tijdstappen zoals bij een seizoensgebonden patroon (Fischer & Krauss, 2018). Een voorbeeld van een cyclische patroon kan bijvoorbeeld een politieke cyclus zijn, waarbij verandering van politieke leiders economische reacties kunnen veroorzaken. Een vierde factor die invloed kan hebben op waarden zijn uitschieters. Dit

zijn patronen die zich vaak niet herhalen en ontstaan door invloeden van buitenaf, zoals het nieuws. De lengte van data dat nodig is om deze patronen te vinden kan erg verschillen, maar over het algemeen wordt er gesproken over jaren van data om met zekerheid te zeggen dat er een patroon aanwezig is (Fischer & Krauss, 2018).

2.2 Voorspellen van tijdreeksen

Naar het voorspellen van financiële tijdreeksen is veel onderzoek gedaan. Er is vanuit de literatuur een overeenstemming dat lineaire modellen geen nauwkeurige voorspellingen kunnen maken over een tijdreeks (Clements et al., 2004; Franses & van Dijk, 2000). Taylor (2008) concludeerde in zijn boek over het modelleren van financiële tijdreeksen dat een redelijk model een niet-lineair model moet zijn. Het hoeft echter niet te betekenen dat niet-lineaire modellen automatisch de oplossing zijn voor een tijdreeks probleem (Azoff, 1994; Terui & van Dijk, 2002). Desondanks is er binnen de literatuur voornamelijk aandacht voor het gebruik van niet-lineaire modellen. Financiële tijdreeksen tonen bijvoorbeeld vaker grote dalingen in de prijs dan grote stijgingen in de prijs. Dit soort gedrag wordt ook wel asymmetrisch gedrag genoemd. Deze asymmetrie kan veel beter in kaart gebracht worden in een niet-lineair model dan een lineair model (Franses & van Dijk, 2000). Een ander asymmetrische gedraging van financiële tijdreeksen is dat periodes van grote prijsdalingen vaak een voorbode zijn voor een periode van heftige prijsschommelingen, waartegen dit niet het geval is in een periode van een grote prijsstijging (Taylor, 2008).

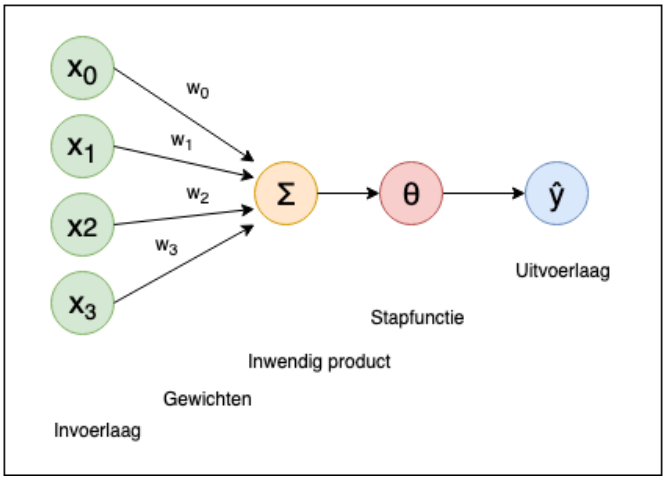
Omdat er binnen de literatuur de consensus is dat niet-lineaire modellen toepasbaar zijn op financiële tijdreeksen, richt dit onderzoek zich ook op dergelijke modellen. Het volgende hoofdstuk zal verder ingaan op de architecturen van enkele niet-lineaire modellen. Hierbij gaat het om verschillende typen neurale netwerken, waaronder een standaard neuraal netwerk en twee typen recurrent neurale netwerken (Long Short-Term Memory netwerk en Gated Recurrent Unit netwerk).

3 Netwerk architecturen

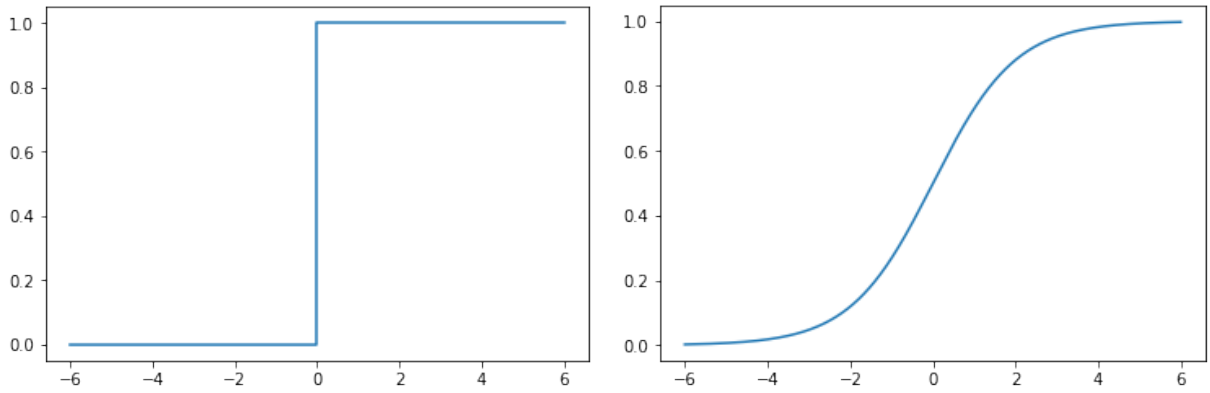
3.1 Single-Layer Perceptron

De meest simpele vorm van een kunstmatig neuraal netwerk (Engels: Artificial Neural Network) is de perceptron (Engels: Single Layer Perceptron). Een SLP bestaat uit vier onderdelen: de invoerlaag X, gewichten W met een bias b, het inwendig product hiervan en de activeringsfunctie (figuur 3.1). Een SLP heeft hiermee één laag aan verbindingen tussen de invoer- en uitvoerlaag. De invoerlaag bestaat uit een inputvector X en een outputvector Y, die respectievelijk bestaan uit de input- en outputvariabelen x en y. De gewichten worden in het begin van het model willekeurig gekozen.

Het doel van de gewichten is om aan te geven hoe sterk de verbinding tussen twee neuronen is, hoe groter het gewicht wordt hoe belangrijker de input is op het netwerk. De bias waarde b is een gewicht, vermenigvuldigd met een constante, die wordt gebruikt om de output aan te passen en wordt als grenswaarde gezien (Goodfellow et al., 2016). Er zijn verschillende activeringsfuncties die het inwendig product van de gewichten- en inputvector transformeert naar een waarde wat informatief is voor het model. Activeringsfuncties die veel gebruikt worden in classificatieproblemen zijn onder andere de stapfunctie (Engels: Heaviside step function) en de logistische functie (Engels: sigmoid function), die een waarde tussen 0 en 1 teruggeven (figuur 3.2). De gewenste output bepaalt welke activeringsfunctie gebruikt moet worden.



Figuur 3.1. Een representatie van een single-layer perceptron.



Figuur 3.2. Links en rechts wordt respectievelijk de stapfunctie en de sigmoid functie weergeven.

De invoer van een perceptron netwerk bestaat uit een vector X met $n+1$ waarden. De eerste waarde bevat de bias waarde x_0 die 1 toegekend krijgt. Elke waarde in vector X wordt vermenigvuldigd met een corresponderende rij uit gewichtenvector W. Deze vector bestaat uit $n+1$ waarden van gewichten, waarbij de eerste waarde w_0 gelijk staat aan b. Het inwendig product van de twee vectoren, $W^T X$, wordt vervolgens door een activeringsfunctie gehaald. Het model transformeert dit vervolgens naar bijvoorbeeld een classificatie van 0 als de output kleiner is dan de grenswaarde of 1 als de output groter is dan de grenswaarde (figuur 3.3). Dit proces wordt forward propagation genoemd.

$$w_0 x_0 + w_1 x_1 + \cdots + w_{n-1} x_{n-1} + w_n x_n \leq 0$$

$$w_0 x_0 + w_1 x_1 + \cdots + w_{n-1} x_{n-1} + w_n x_n > 0$$

Figuur 3.3. Als de grenswaarde kleiner of gelijk is aan 0 (bovenste formule), dan geeft de stapfunctie een waarde van 0 terug. Als de grenswaarde groter is dan 0 (onderste formule), geeft de stapfunctie een waarde van 1 terug (figuur 3.2).

Door de gewichten na elke ronde te updaten over de trainingsdata, kan het model leren de gewenste output zo dicht mogelijk te benaderen. Dit gebeurt door telkens een verkeerd geclassificeerd punt x_i te pakken, waarbij geldt dat $sign(W^T X) \neq y_i$. Daarna kunnen alle gewichten W tegelijk aangepast met een gewicht updateregel: $w_i = w_i + learning\ rate * (verwachte\ y_i - voorspelde\ \hat{y}_i) * x_i$. Door deze stappen te herhalen wordt de ideale lineaire functie bereikt door het perceptron. Dit proces wordt in een SLP de delta regel of het LMS algoritme (Least Mean Squared) genoemd.

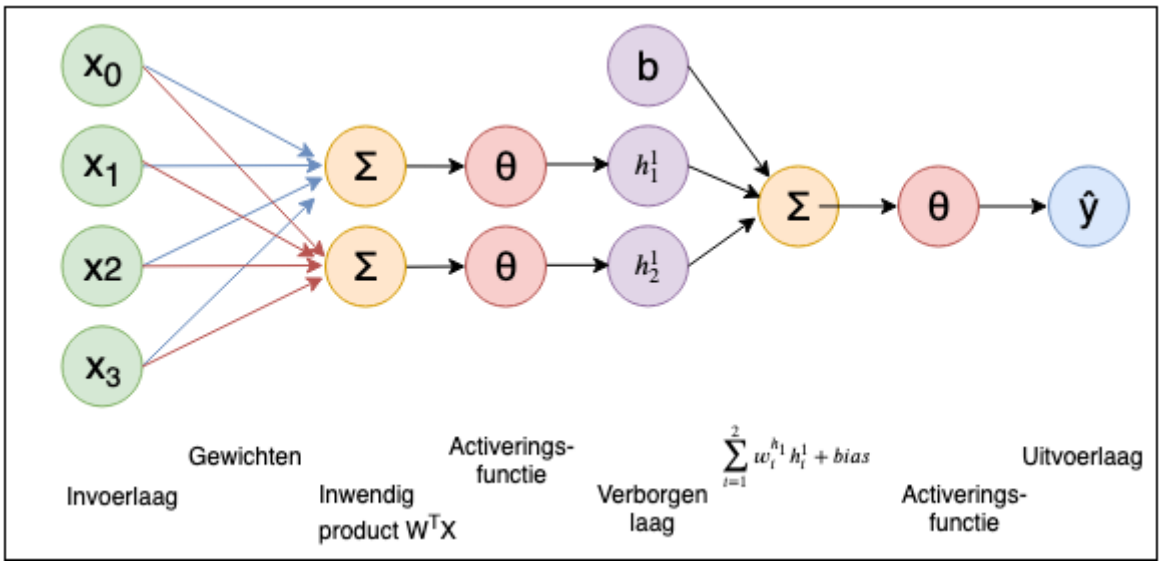
3.2 Multi-Layer Perceptron

Een SLP kan enkel lineaire verbanden vinden. Dit is in de praktijk niet bruikbaar, omdat bijna geen enkele dataset een lineair verband heeft. Met meerdere lagen (MLP) is een netwerk van perceptrons in staat ingewikkeldere verbanden te vinden. Een MLP is een neuraal netwerk dat bijna dezelfde structuur bevat als een SLP, met als groot verschil dat er een of meerdere verborgen lagen tussen de invoer- en uitvoerlaag bevinden (figuur 3.4). De output van een laag wordt de input van de opeenvolgende laag in dit geval. Een MLP is in staat niet-lineaire verbanden te vinden, iets waartoe een SLP niet in staat is (Minsky & Papert, 1969). Hornik, Stinchcombe & White (1989) bewezen dat een twee-laag netwerk met sigmoid activeringsfuncties in de verborgen lagen en lineaire functies in de output laag praktisch elke functie kan benaderen zolang er voldoende verborgen neuronen gebruikt worden. Veenema (1999) concludeerde dat dit soort modellen echter niet geschikt zijn voor de predictie van tijdreeksen.

Het netwerk wordt via de invoerlaag een vector X aangeboden, die via de gewogen verbindingen (gewichten) het signaal doorgeeft door de lagen aan de uitvoerlaag. Het netwerk wordt vervolgens getraind door middel van backpropagation (Kröse & van der Smagt, 1993). De uitvoer van de invoervectoren wordt vergeleken met de werkelijke uitvoer en wordt teruggekoppeld om de gewichten aan te passen. Het meest eenvoudige backpropagation algoritme is de Gradient Descent (Gençay & Liu, 1997). Deze functie verandert de gewichten in de richting waarin de fout het meest zal dalen.

$$nieuw\ gewicht = gewicht - (learning\ rate * gradient)$$

Een MLP heeft echter geen besef van tijdsvolgorde en beschouwt de input als enkel input. Hiermee is het niet geschikt om een tijdreeks probleem op te lossen.



Figuur 3.4. Een multi-layer perceptron netwerk.

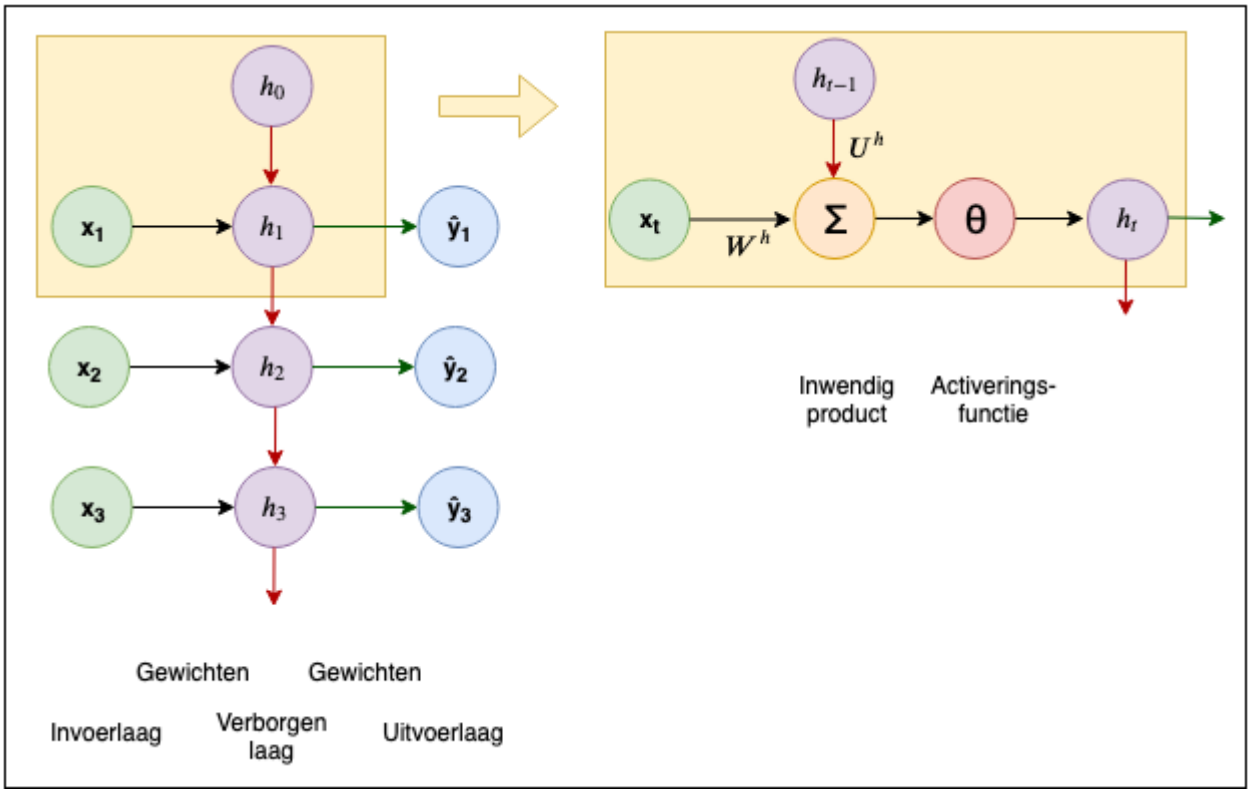
3.3 Recurrent Neuraal Netwerk

Een model dat een tijdreeks probleem kan modelleren is een recurrent neuraal netwerk (RNN). Een RNN is een netwerk met feedback, waarbij output van het vorige tijdstip deel wordt met de input van het huidige tijdstip (figuur 3.5). Op deze manier kan een RNN met de uitvoer die verkregen wordt van tijdstip x_{t-1} de beslissing over tijdstip x_t beïnvloeden. Het inwendig product van de input van het huidige tijdstip x_t met de gewichten W^h en de voorgaande verborgen laag h_{t-1} met de gewichten U^h worden opgeteld en gaan door een activeringsfunctie, waar het resultaat de verborgen laag van het huidige tijdstip h_t wordt (figuur 3.5). Hiermee kan gesteld worden dat een RNN een geheugen heeft in een verborgen laag dat telkens aangepast en doorgegeven wordt naar het volgende tijdstip.

$$nieuwe\ verborgen\ laag\ h_t = \ tanh(W^h x_t + U^h h_{t-1})$$

$$output\ y_t = \ W^y h_t$$

De hoeveelheid berekeningen in een RNN kan snel oplopen. Een nadeel met een RNN trachten een tijdreeks probleem op te lossen is dat het geen informatie over een langere periode kan leren. Het probleem wordt veroorzaakt doordat de gradiënt zeer klein kan worden na een bepaald hoeveelheid tijdstippen. Omdat deze gradiënt exponentieel kleiner wordt per tijdstip, leert het netwerk na deze hoeveelheid tijdstippen niet meer en verdwijnt deze kennis. Een RNN heeft hiermee een zogezegd korte termijn geheugen en is niet in staat verbanden te vinden over langere periodes, omdat het niet kan beslissen welke informatie in een tijdstip onthouden moet worden en welke informatie vergeten moet worden. Dit probleem wordt ook wel vanishing gradient genoemd (Hochreiter, 1991).



Figuur 3.5. Een recurrent neuraal netwerk. De zwarte pijlen in het linkergedeelte van de afbeelding stellen de gewichten W^h voor, de groene gewichten zijn W^y en de rode pijlen stellen de gewichten U^h voor. Aan de rechterkant is het proces van een RNN cel geïllustreerd.

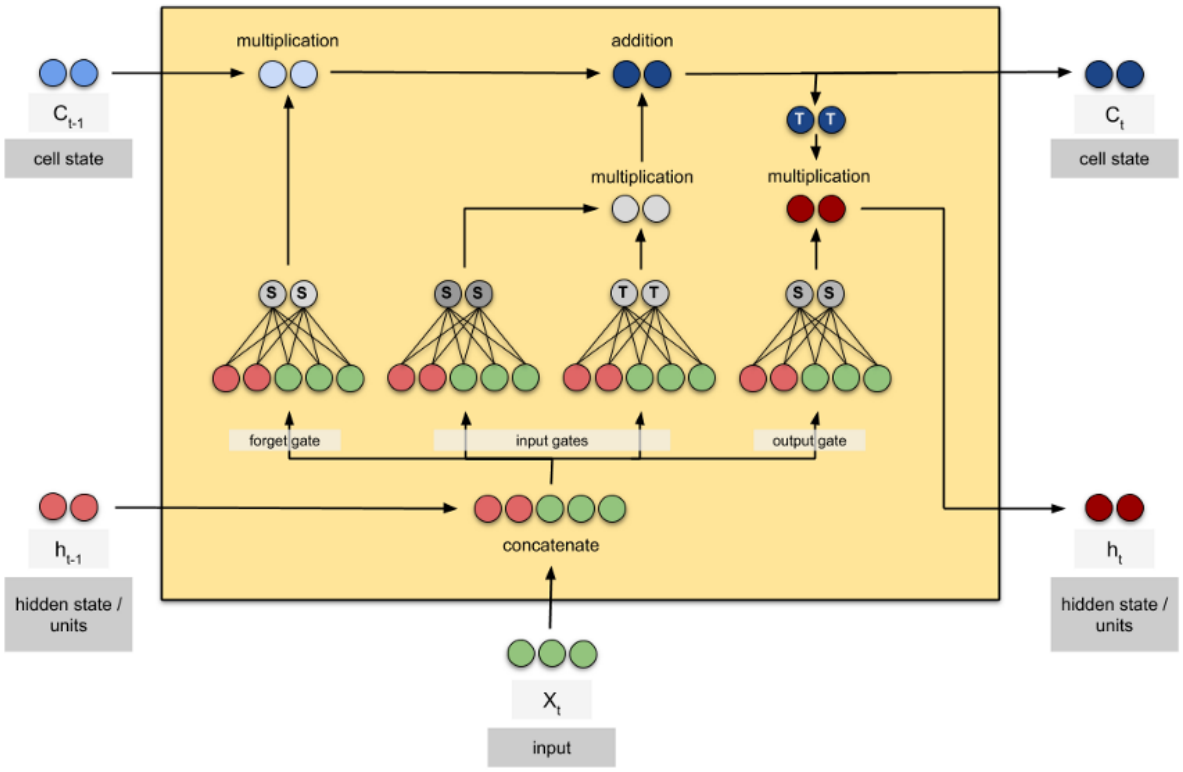
3.4 LSTMs

In 1997 is door Hochreiter, met de introductie van LSTMs, een oplossing gevonden voor het vanishing gradient probleem. Deze oplossing gebruikt poorten (Engels: gates) die de informatiedoorloop controleren door een beslissing te maken welke informatie vergeten of onthouden moet worden.

Een LSTM cel bezit een geheugencel en drie poorten. De input die een LSTM cel binnenkrijgt is informatie van de verborgen laag uit de vorige verborgen laag h_{t-1} en de input van huidige tijdstip x_t, wat vergelijkbaar is met een RNN cel (figuur 3.6). Het verschil echter is dat er ook informatie uit de geheugencel van het voorgaande tijdstip, c_{t-1}, meegegeven wordt als verborgen laag die de tijdreeks tot op tijdstip x_{t-1} heeft *gezien*. Deze functie wordt, samen met het maken van een output voorspelling, in een RNN cel door h_t vervuld. Eerst worden de processen van de forget gate en input gates uitgelegd, daarna de functie van het cel geheugen en ten slotte het proces van de output gate en welke output de LSTM cel levert.

De functie van de forget gate is het laten vergeten of onthouden van informatie uit het geheugencel van het vorige tijdstip c_{t-1}. Dit gebeurt door het inwendig product van de voorgaande verborgen laag h_{t-1} en de huidige input x_t met hun gewichten W^f en U^f door een sigmoid functie te halen. Hier komen waarden tussen 0 en 1 uit, waarbij 0 compleet vergeten en 1 compleet onthouden betekent. Deze output wordt later vermenigvuldigd met cel geheugen c_{t-1}. De forget gate beslist hiermee hoeveel van de informatie uit de voorgaande laag opgeslagen moet blijven in het geheugencel.

$$forget\ gate\ f_t = \sigma(W^f x_t + U^f h_{t-1})$$



Figuur 3.6. LSTM cell. Verkregen op 13 juni 2019 van https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45.

De mogelijkheid om relevante informatie toe te voegen aan het geheugencel van het voorgaande tijdstip c_{t-1} gebeurt met behulp van de **input gate**. Het inwendig product van de input x_t wordt met de verborgen laag h_{t-1} en hun gewichten W^i en U^i wederom door een sigmoid functie gehaald. Deze output tussen 0 en 1 beslist hoe belangrijk de informatie is die aangereikt is aan de cel. Daarnaast wordt input x_t en h_{t-1} vermenigvuldigd met de gewichten W^c en U^c, waarop vervolgens een tanh activeringsfunctie gebruikt wordt die output tussen -1 en 1 levert. De output van deze tanh functie wordt ook wel **kandidaat geheugencel** $\hat{c}_t$ genoemd. De output van input gate i_t en kandidaat geheugencel $\hat{c}_t$ worden hierop vermenigvuldigd met elkaar. Dit wordt vervolgens opgeteld met de multiplicatie van de output van de forget gate en het geheugencel $\hat{c}_{t-1}$, waarop de uitkomst van deze optelling cel geheugen c_t vormt.

$$input\ gate\ i_t = \sigma(W^i x_t + U^i h_{t-1})$$

$$kandidaat\ geheugencel\ \hat{c}_t = tanh(W^c x_t + U^c h_{t-1})$$

De informatie uit het geheugencel van het vorige tijdstip c_{t-1} wordt met de corresponderende elementen vermenigvuldigd (Hadamardproduct) met de output van de forget gate f_t. Als informatie uit het verleden minder belangrijk geacht wordt, zal de output van de forget gate dicht bij 0 zitten en zal deze multiplicatie ervoor zorgen dat de waarde van het geheugencel $\hat{c}_t$ kleiner wordt. Daarna wordt informatie uit de input gate met $\hat{c}_t$ opgeteld om het netwerk te laten weten of nieuwe informatie die binnen in gekomen belangrijk geacht wordt. Dit leidt tot het nieuwe geheugencel c_t. Dit nieuwe geheugencel c_t

bestaat hiermee uit het onthouden gedeelte van de voorgaande geheugencel met een gedeelte van de geheugencel $\hat{c}_t$.

$$nieuw\ cel\ geheugen\ c_t = c_{t-1} \odot f_t + \hat{c}_t$$

Ten slotte wordt informatie door de **output gate** gestuurd. De output gate helpt beslissen in hoeverre de volgende verborgen laag h_t bijgestuurd moet worden, zodat dit achteraf doorgegeven wordt als extra input h_{t-1} in het volgende tijdstip. Het inwendig product van de verborgen laag h_{t-1} wordt met de input x_t en hun gewichten W^o en U^o voor de laatste keer door een sigmoid functie gehaald. Hierop gaat informatie uit het nieuwe geheugencel c_t door een tanh functie, wat een waarde tussen -1 en 1 levert. Van de output van output gate o_t en de tanh functie $tanh(c_t)$ worden vervolgens het Hadamardproduct berekend. Dit vormt de nieuwe verborgen laag h_t, die doorgegeven zal worden met het geheugencel c_t.

$$output\ gate\ o_t = \sigma(W^o x_t + U^o h_{t-1})$$

$$nieuwe\ verborgen\ laag\ h_t = o_t \odot tanh(c_t)$$

Het voordeel van een LSTM is dat het mogelijk is patronen over grotere periodes te vinden. Daarnaast lijken LSTMs geen noodzaak te hebben voor zorgvuldig kiezen van parameters. Een nadeel is dat het gewichten voor elke gate gebruikt, waardoor het gebruik meer berekeningen en hiermee meer rekenkracht nodig heeft (Hochreiter & Schmidhuber, 1997).

3.5 GRUs

In 2014 is er een simpelere variant van de LSTM geïntroduceerd, de Gated Recurrent Unit (GRU). Het verschil met een LSTM cel is dat de GRU cel een poort minder heeft. In plaats van de drie gates en de geheugencel die een LSTM gebruikt, wordt in deze structuur slechts gebruik gemaakt van twee gates, de update en reset gate (figuur 3.7).

Een GRU cel start in het begin met de **update gate**. De update gate beslist, net als de forget gate, welke van de informatie uit het vorige tijdstip behouden moet worden. Het verschil met de forget poort is dat dit op een andere volgorde gebeurt. Het voordeel van een GRU is dat ze iets sneller werken en minder rekenkracht vergen (Chung et al., 2014).

Het inwendig product van de voorgaande verborgen laag h_{t-1} en de huidige input x_t met hun bijbehorende gewichten worden door een sigmoid functie gehaald. In de formule hieronder staan W^z voor de gewichten van input x_t en U^z voor de gewichten over de verborgen laag h_{t-1}.

$$update\ gate\ u_t = \sigma(W^z x_t + U^z h_{t-1})$$

De reset gate beslist, net als de input gate van een LSTM cel, hoeveel informatie uit het voorgaande tijdstip vergeten moet worden. De formule voor de reset gate is hetzelfde als die van de update gate. Het verschil is dat de output van de reset gate later meegenomen wordt om te beslissen hoe belangrijk de nieuwe informatie is die binnen is gekomen.

$$reset\ gate\ r_t = \sigma(W^r x_t + U^r h_{t-1})$$

Het Hadamardproduct van de reset gate r_t met de vermenigvuldiging van de verborgen laag van tijdstip h_{t-1} en gewichten U worden opgeteld met de vermenigvuldiging van input x_t en gewichten W. Het resultaat hiervan gaat door een tanh activeringsfunctie heen en wordt kandidaat verborgen laag

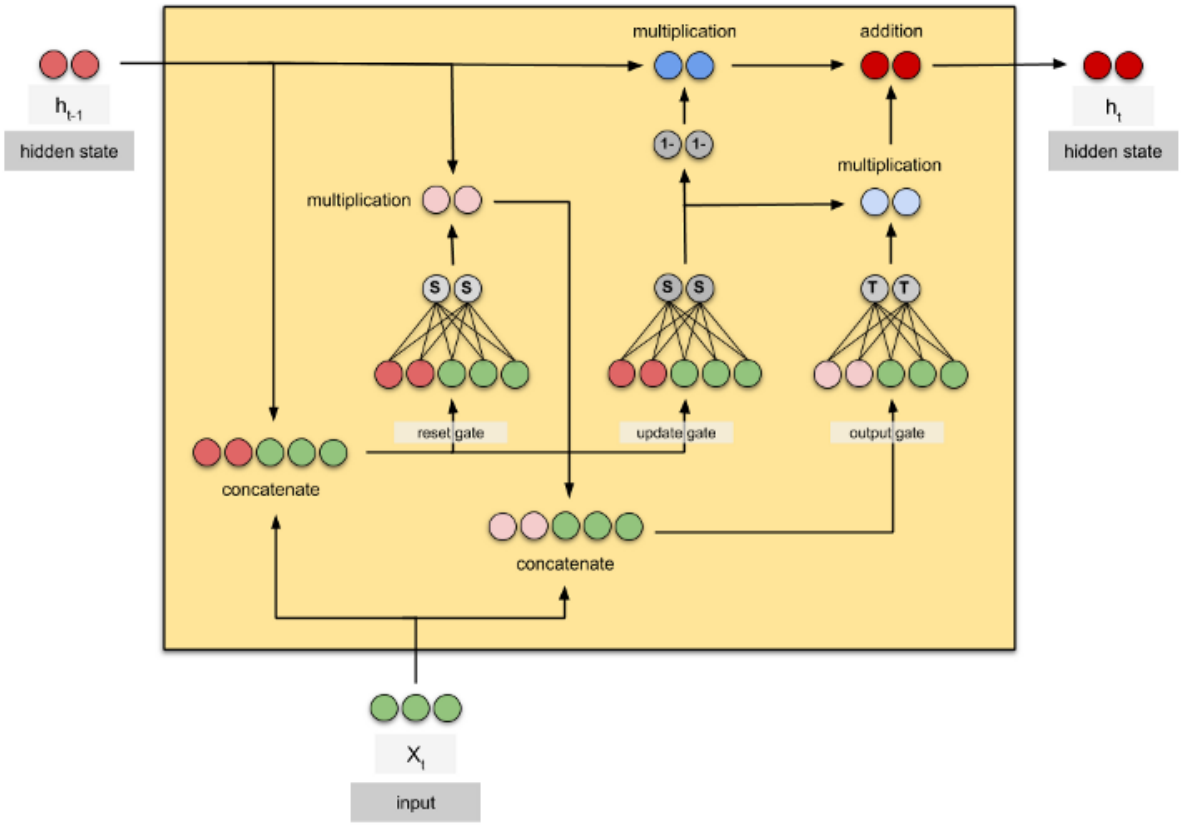
wordt $\hat{\mathrm{h}}_t$. Deze kandidaat $\hat{\mathrm{h}}_t$ bestaat hiermee uit een combinatie van de ingevoerde informatie en een selectie van onthouden informatie uit het vorige tijdstip. In het geval dat r_t enkel enen bevat wordt alle informatie uit de vorige verborgen laag onthouden. Uiteindelijk zorgt de activeringsfunctie voor een output tussen -1 en 1.

$$kandidaat\ hypothese\ \hat{\mathrm{h}}_t = tanh(Wx_t + r_t \odot Uh_{t-1})$$

Uiteindelijk wordt de verborgen laag h_t gevormd door het Hadamardproduct van update gate u_t met de voorgaande verborgen laag h_{t-1} op te tellen met het Hadamardproduct van $1 - u_t$ met de kandidaat verborgen laag $\hat{\mathrm{h}}_t$. In het geval dat $1 - u_t$ een vector is met getallen dicht bij de 0, betekent dit dat vector u_t voornamelijk getallen dicht bij de 1 bevat. Daarmee wordt veel informatie uit de voorgaande verborgen laag h_{t-1} minder belangrijk wordt geacht en veel informatie kandidaat verborgen laag $\hat{\mathrm{h}}_t$ meegenomen wordt naar de huidige verborgen laag h_t.

$$verborgen\ laag\ h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \hat{\mathrm{h}}_t$$

Als de reset gate r_t alleen enen bevat en de update gate u_t enkel nullen, is de GRU cel in essentie gelijk aan een RNN cel.



Figuur 3.7. GRU cell. Verkregen op 13 juni 2019 van https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

4 Gerelateerd onderzoek

Er zijn op verschillende benaderingswijzen onderzoeken uitgevoerd naar prijsvoorspellingen van valuta met neurale netwerken. Zo zijn bijvoorbeeld het prijsverleden van een valuta, externe variabelen zoals de het aantal koop- en verkooporders (Guo, 2018) of het online sentiment van een valuta (Kim et al., 2016) gebruikt om de prijs te voorspellen van verschillende traditionele valuta's en cryptovaluta's. In dit hoofdstuk worden enkele benaderingswijzen die toegepast zijn om prijsvoorspelling uit te voeren geanalyseerd. Daarnaast worden werkwijzen van onderzoekers en resultaten van experimentele onderzoeken bestudeerd om te begrijpen hoe verschillende typen netwerken hebben gepresteerd in voorspellingstaken van financiële tijdsreeksen.

4.1 Onderzoek van invloed externe variabelen

Er is relatief veel geschreven over de invloed van externe variabelen (features) op het voorspellen van de prijs van traditionele en cryptovaluta's. In het paper 'Predicting short-term Bitcoin price fluctuations from buy and sell orders' uit 2018 analyseert Guo het effect van koop- en verkooporders op de prijs van Bitcoin. Het doel was om schommelingen in de prijs van Bitcoin te voorspellen door modellen op te stellen die lineaire combinaties van normaalverdelingen gebruiken (Engels: Mixture models). Alessandretti et al. (2018) hebben in hun onderzoek 1681 cryptovaluta's geanalyseerd. Het doel van hun onderzoek was om met XGBoost en LSTM netwerken toekomstige waarden te voorspellen van deze valuta's. Zij concludeerden dat het mogelijk is om de prijzen daadwerkelijk te benaderen. Nelson et al. (1994) hebben onderzoek gedaan naar het herkennen van seizoensgebonden patronen in een tijdreeks. Zij stelden dat standaard neurale netwerken niet capabel zijn om deze patronen te herkennen.

Er zijn ook veel onderzoeken uitgevoerd die technische analyse gebruiken om de prijs van valuta's te voorspellen. Bij technische analyse wordt gebruik gemaakt van historische informatie om beslissingen te nemen over het kopen of verkopen van een valuta. Hier wordt gebruik gemaakt van het herkennen van patronen in prijsgrafieken of het calculeren van indicatoren die de prijsrichting moeten voorspellen. Radityo et al. (2017) gebruiken enkele technische indicatoren als feature om neurale netwerken te trainen om de prijs van Bitcoin te voorspellen. Het onderzoek van Jang & Lee (2017) richtte zich op het gebruik van features, waaronder het aantal transacties per dag en de marktwaarde van Bitcoin, om de prijs van deze valuta te voorspellen.

4.2 Onderzoek voorspellen op basis van prijs

Er is ook onderzoek te vinden over het voorspellen van de prijs van een valuta op basis van enkel de historische prijsgeschiedenis. Kian en Liu (1995) onderzochten het voorspellende vermogen van standaard en recurrent neurale netwerken. Modellen werden getraind op de dagprijzen van vijf wisselkoersen van de Amerikaanse dollar over een periode van vijf jaar. Zij vonden dat de resultaten gunstig waren voor twee van de vijf wisselkoersen, maar stelden ook dat dit niet genoeg bewijs was om de bruikbaarheid van neurale netwerk modellen te rechtvaardigen.

Het onderzoek van Torres & Qiu (2018) richtte zich op de analyse van twee typen recurrent neurale netwerken, een LSTM en GRU netwerk. Torres & Qiu (2018) onderzochten welk type netwerk de hoogste nauwkeurigheid behaalde in de voorspelling van volatiele beursdata en of deze netwerken beter presteerden dan traditioneel gebruikte ARIMA modellen. Er bleek uit hun onderzoek dat er geen verschil was in de nauwkeurigheid van voorspelling tussen deze twee typen netwerken. Er werd wel vastgesteld dat GRU netwerken een kortere trainingstijd hadden dan LSTM netwerken. Gers et al. (2002) concludeerden dat LSTMs niet beter presteerde dan andere modellen bij simpele tijdreeksvoorspellingen.

Zij stelden dat LSTMs beter gebruikt kunnen worden bij complexere voorspellingstaken. Het onderzoek van Gers et al. was echter niet uitgevoerd op financiële tijdreeksen, waarvan gesteld mag worden dat dit een complexe voorspellingstaak is. Zhang et al. (1998) schreven in hun literatuur review dat neurale netwerken veelbelovend, maar ook wispelturig presteerden in het voorspellen van de prijs van valuta's. Zij stelden dat het selecteren en prepareren van de training en test data cruciaal is om een model goed te trainen en dat dit vaak fout geschied. Hann en Steurer (1995) vonden dat een neuraal netwerk beter presteerde in het voorspellen van de wisselkoers dan lineaire modellen op wekelijkse data. Echter presteerden beide modellen hetzelfde als zij getraind werden op maandelijkse data. Batres-Estrada (2015) vond in zijn onderzoek dat neurale netwerken beter presteerden op financiële data dan andere netwerken. Deze financiële data bestonden uit dagelijkse prijzen van de S&P 500 (een Amerikaanse aandelenindex) over een periode van 22 jaar.

5 Experimenteel onderzoek

Dit hoofdstuk beschrijft de methode en data die gebruikt zijn voor het trainen van de modellen voor prijsvoorspelling van Bitcoin. In paragraaf 5.1 wordt het experiment onderzoek beschreven, paragraaf 5.2 beschrijft de data die gebruikt wordt in het model en paragraaf 5.3 beschrijft de model opzet die opgesteld wordt. Ten slotte omschrijft paragraaf 5.4 wat hyperparameters zijn en het proces dat gevolgd is voor de optimalisatie van hyperparameters.

5.1 Experiment

Om te bewijzen of welk type neuraal netwerk de beste resultaten levert op prijsvoorspelling van Bitcoin, is het experimenteel onderzoek van Ly, Timaul, Lukanan, Lau en Stainmetz (2018) met enkele aanpassingen gereproduceerd. In dit experimenteel onderzoek worden de verschillende neurale netwerken gebruikt om de prijs van Bitcoin te voorspellen van een week later te voorspellen op basis van de afgelopen vijf dagen. Net als het onderzoek van Ly et al. is er gebruik gemaakt van de Python bibliotheek Keras, een open-source neuraal netwerk bibliotheek dat ontwikkeld is om te experimenteren met neurale netwerken. Het doel van dit onderzoek is om aan te tonen hoe de verschillende neurale netwerken presteren in de prijsvoorspelling van Bitcoin en of het gebruik van features de nauwkeurigheid van deze neurale netwerken verbeteren.

5.2 Data

Het geschreven programma maakt gebruik van de data die gebruikt is in het nagebootst onderzoek van Ly et al. (2018). Omdat de herkomst van de data niet vermeld stond in het originele paper, is de data gedownload van de website investing.com. Eventuele prijsverschillen tussen verschillende cryptomarkten zullen aanwezig, doch verwaarloosbaar zijn. De modellen die gebruikt zijn om het experimenteel onderzoek van Ly et al. na te bootsen, maken gebruik van de sluitingsprijzen per dag tussen de periode van 28 april 2013 en 05 februari 2018. Deze periode heeft in het laatste jaar grote schommelingen in de prijs meegemaakt, waardoor het mogelijk lastig is voor het model om de prijs te voorspellen. Om deze reden is er besloten een tweede dataset te maken die bestaat uit de sluitingsprijzen per dag tussen de periode van 28 april 2013 en 10 juni 2019. Er wordt aan iedere dataset telkens een feature toegevoegd om te onderzoeken of de nauwkeurigheid van voorspellen verbeterd. Deze gebruikte features worden in hoofdstuk 5.3 toegelicht.

Wegens het gebrek aan rekenkracht is enkel gekeken naar de sluitingsprijzen per dag. Waar andere literatuur ook kijkt naar andere tijdstappen zoals per halve dag, per uur of zelfs per minuut, is dit niet mogelijk met de apparatuur waarmee de modellen worden getraind. De periode waarop het model getraind moet worden zal snel groeien tot een paar dagen per model, in tegenstelling tot de 6 tot 10 uur die nodig zijn om een model met dagwaarden te trainen.

5.3 Feature selectie

Er is een selectie van de features in de derde dataset aanwezig. Deze features zijn:

- Het aantal unieke Bitcoin adressen die in gebruik zijn op een dag (Engels: Active Addresses).
- Het aangepaste volume van de waarde txVolume (Engels: Adjusted Tx Volume).
- De gemiddelde moeilijkheidsgraad om een nieuwe Bitcoin te genereren (Engels: Average Difficulty).
- De gemiddelde grootte van een Bitcoin block in MB's (Block Size).

- De volume van de grootste cryptomarken, waaronder Bitfinex en GDAX (Engels: Exchange Volume).
- De totale aantal transactiekosten om Bitcoin te versturen die gemaakt zijn op een dag (Engels: Fees).
- Het aantal nieuwe Bitcoins die gemijnd zijn op een dag (Engels: Generated Coins).
- Totale marktwaarde van Bitcoin, berekend door het aantal Bitcoins die in omloop zijn te vermenigvuldigen met de prijs van Bitcoin (Engels: Market Capitalization).
- De gemiddelde kosten per transactie, berekend door het totale aantal transactiekosten te delen met de het aantal transacties (Engels: Median Fee).
- Het gemiddelde aantal transacties per block (Engels: Median Tx).
- De totale opbrengst van alle Bitcoin mijners op een dag, berekend door het aantal Bitcoin dat gemijnd is op een dag te vermenigvuldigen met de prijs en hier de transactiekosten bij op te tellen (Engels: Payment Count).
- Een alternatieve berekening van de totale marktwaarde van Bitcoin in USD, waarbij de afzonderlijke valuta de waarde krijgt van het laatste moment dat het verhandeld of verplaatst is (Engels: Realized Capitalization).
- Aantal unieke transacties die op het Bitcoin blockchain verstuurd zijn op een dag (Engels: Tx Count).
- Het volume van de waarde (in USD) van alle transacties op de Bitcoin blockchain op een dag (Engels: Tx Volume).

Er is gebruik gemaakt van de Python bibliotheek Linselect om er stapsgewijs (Engels: stepwise selection algorithm) achter te komen welke van de bovenstaande features een positieve bijdrage levert aan de nauwkeurigheid van de voorspellingsmodellen. Het doel is om een selectie van de bovenstaande features te nemen, met de achterliggende reden om computerrekenkracht te besparen. Het trainen van neurale netwerken op veel features duurt namelijk zeer lang (paragraaf 5.5), waardoor de keuze is gemaakt om enkel drie features te selecteren uit de lijst.

Bij de selectie van features zijn twee feature zoekalgoritmen gebruikt, FwdSelect en RevSelect (EFavDB, 2018). Deze zoekalgoritmen lijken op het eerste gezicht niet geschikt te zijn om feature selectie toe te passen met betrekking tot neurale netwerken, omdat ze lineaire verbanden proberen te vinden tussen de features en de prijs van Bitcoin. Deze algoritmen zijn toch als heuristische methode gebruikt om te bepalen welke features een lineair verband hebben met de prijs van Bitcoin en welke juist geen correlatie tonen (EFavDB, 2018). Het algoritme FwdSelect leverde helaas geen duidelijke kandidaat features, omdat alle bovenstaande features geen correlatie bleken te hebben met de prijs van Bitcoin. Met dit algoritme wordt telkens een feature toegevoegd om te berekenen of de prestatie van het model beter wordt. Het algoritme RevSelect doet het omgekeerde en verwijderd telkens een feature uit de featurelijst, om te berekenen welke subset van de featurelijst de prestatie het minst doet dalen (EFavDB, 2018). Met RevSelect zijn uiteindelijk de beste voorspellende features geselecteerd, namelijk de totale marktwaarde (Market Capitalization), de totale opbrengst van alle Bitcoin mijners op een dag (Payment Count) en de alternatieve totale marktwaarde (Realized Capitalization).

5.4 Model setup

Het programma is geschreven in Python met behulp van enkele bibliotheken, waaronder Numpy, Pandas en Keras. In het eerste en derde experiment bestaat de inputdata, zoals voorheen vermeld, uit de

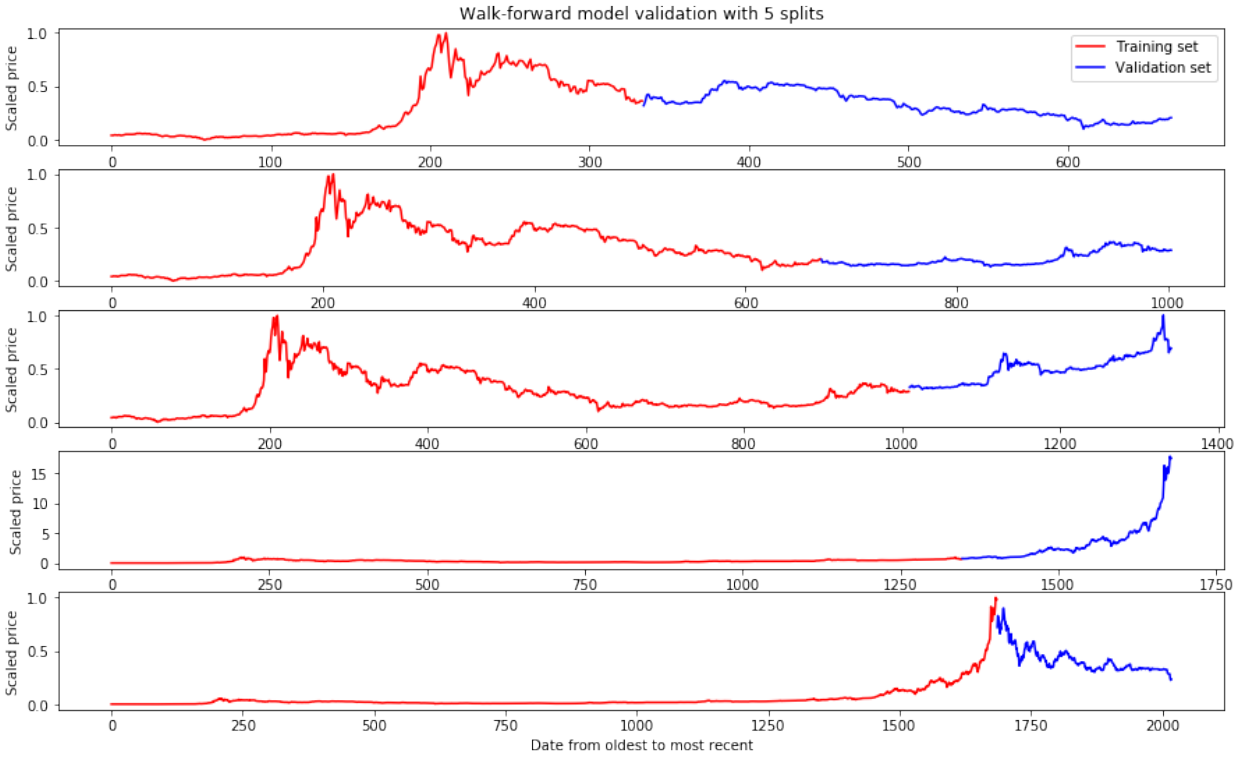
sluitingsprijzen per dag voor de bovenstaande periode (paragraaf 5.2). In het tweede en vierde experiment wordt een van de drie gekozen features toegevoegd aan respectievelijk het eerste en derde experiment. Na het opstellen en trainen van ieder netwerk model met de training data, wordt de prijs van de week erop voorspelt. Dit vormt de outputdata. Nadat deze outputdata verzameld is, wordt gekeken wat het verschil tussen de originele prijs en de voorspelde prijs is. Om de resultaten te kunnen vergelijken met het nagebootst onderzoek van Ly et al. is dezelfde foutmaat genomen, de MSLE (Mean Squared Logarithmic Error). Deze functie is een variant van de MSE (Mean Squared Error). MSLE gebruikt het verschil van het logaritme van de voorspelde en de daadwerkelijke waarde, om dit vervolgens te kwadrateren. De MSE daarentegen kwadrateert enkel het daadwerkelijke verschil tussen de twee waarden (Figuur 5.1). Het voordeel van het gebruik van MSLE is dat grote prijsverschillen in de voorspelde waarde en de daadwerkelijke waarde niet hard afgestraft worden als beide waarden grote getallen zijn. Daarnaast straft de MSLE onderschatte waarden harder af dan overschatte waarden (Socaci & Lemnaru, 2018).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2$$

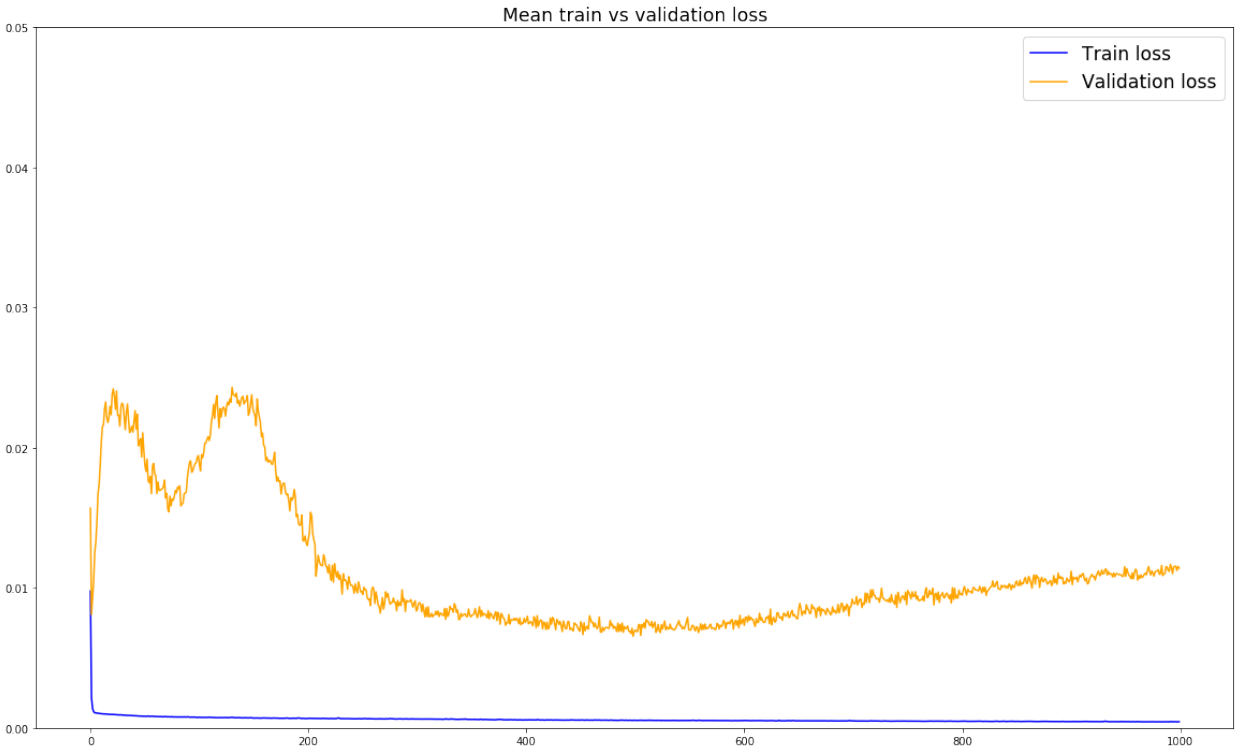
$$MSLE = \frac{1}{n}\sum_{i=1}^{n}(\log(y^i + 1) - \log(\hat{y}^i + 1))^2$$

Figuur 5.1. MSE en MSLE formules. De waarde *n* staat in dit geval voor het aantal tijdstippen, *y* voor de daadwerkelijke waarde en *ŷ* voor de voorspelde waarde op tijdstip *i*.

Er wordt getracht de prijs van Bitcoin van een week later te voorspellen op basis van de afgelopen vijf dagen. Een dataset is verdeeld in twee delen: een training set en een test set. De training set wordt vervolgens in meerdere delen gesplitst, zodat crossvalidatie uitgevoerd kan worden. Deze crossvalidatie voorkomt dat een model under- of overfit raakt. Er is ervoor gekozen een variatie op deze methode te gebruiken om de training set in vijf training-validatie sets (splits) te verdelen. Na het splitten van de training set wordt de training set van elke split afzonderlijk geschaald naar waarden tussen de 0 en 1 en wordt de validatie set van elke split respectievelijk geschaald middels dezelfde schaling van de training set (Figuur 5.2). Deze methode voorkomt dat kennis van de te voorspellen waarden in de validatie set kan doorlekken tijdens de training van het model (Kaastra & Boyd, 1996). De gebruikte crossvalidatie methode wordt ook wel *walk-forward model validation* genoemd (Figuur 5.3). Deze methode wordt in paragraaf 5.5 verder toegelicht.



Figuur 5.2. Grafieken van de afzonderlijke training en validatie sets, waarbij elke training set afzonderlijk geschaald is per split naar waarden tussen de 0 en 1 en de test set geschaald is naar deze parameters.



Figuur 5.3. Grafiek met de scores van de gemiddelde training loss en validatie loss per epoch over de training data dat verdeeld is over vijf splits.

Elk model bestaat uit een input laag, twee verborgen lagen, een eventuele dropout tussen de twee verborgen lagen (paragraaf 5.5) en een output laag (Engels: Dense layer). De input laag bestaat uit een reeks prijswaarden, waarvan de lengte bepaald wordt door het aantal tijdstippen (Engels: window size) waar naar teruggekeken wordt (Figuur 5.4). In dit onderzoek bestaat de reeks prijswaarden uit de prijs van Bitcoin over 5 dagen genomen. Een verborgen laag bestaat uit een aantal neuronen (Engels: units), waarbij elke neuron verbonden is aan alle neuronen in de voorgaande en uitgaande laag (hoofdstuk 3).

t_0 [135.3][134.44][144.][139.][116.38]	t_0 [0.003442][0.00339769][0.00389029][0.00363265][0.00246711]
t_1 [134.44][144.][139.][116.38][106.25]	t_1 [0.00339769][0.00389029][0.00363265][0.00246711][0.00194514]
t_2 [144.][139.][116.38][106.25][98.1]	t_2 [0.00389029][0.00363265][0.00246711][0.00194514][0.0015252]
t_3 [139.][116.38][106.25][98.1][112.9]	t_3 [0.00363265][0.00246711][0.00194514][0.0015252][0.0022878]
t_4 [116.38][106.25][98.1][112.9][115.98]	t_4 [0.00246711][0.00194514][0.0015252][0.0022878][0.0024465]
...	...
t_n-3 [6522.27][6442.6][6386.13][6413.63][6411.76]	t_n-3 [0.33254342][0.32843827][0.32552854][0.32694553][0.32684918]
t_n-2 [6442.6][6386.13][6413.63][6411.76][6373.19]	t_n-2 [0.32843827][0.32552854][0.32694553][0.32684918][0.32486178]
t_n-1 [6386.13][6413.63][6411.76][6373.19][6351.24]	t_n-1 [0.32552854][0.32694553][0.32684918][0.32486178][0.32373076]
t_n [6413.63][6411.76][6373.19][6351.24][5736.15]	t_n [0.32694553][0.32684918][0.32486178][0.32373076][0.29203702]

Figuur 5.4. Voorbeeld van reeksen die worden ingevoerd in de input laag. Links staan de prijzen in USD per tijdstip, rechts staan de geschaalde waarden per tijdstip.

Voor het aantal lagen en maximaal aantal neuronen per laag dat gebruikt moet worden in de verborgen laag zijn er verschillende richtlijnen te vinden in de literatuur. Gebruik maken van te veel verborgen neuronen zorgt ervoor dat het model overfit raakt, terwijl te weinig verborgen neuronen het model laat underfitten. In het eerste geval loopt de trainingstijd van het model erg omhoog. In het tweede geval heeft het model moeite met het herkennen van patronen in een dataset. Om in dit onderzoek een juiste balans tussen deze twee scenario's te vinden zijn richtlijnen van Huang (2003) meegenomen. Huang stelt dat het gebruik van twee verborgen lagen ervoor zorgt dat het aantal neuronen dat nodig is drastisch verminderd kan worden. Het totale aantal verborgen neuronen dat nodig is om een neuraal netwerk te trainen met een dataset van n lengte en m output neuronen is: $2\sqrt{n(m+2)}$. De eerste verborgen laag zou maximaal $\sqrt{n(m+2)} + 2\sqrt{\frac{n}{(m+2)}}$ neuronen moeten bevatten, de tweede laag maximaal $m\sqrt{\frac{n}{(m+2)}}$ neuronen. De output laag zorgt er tenslotte voor dat de output die het model voorspelt in de vorm van één enkele waarde is.

In het eerste (en tweede) experiment bestaat de training set uit de eerste 1545 dagen van de beschreven periode en de test set uit de laatste 200 dagen (Figuur 1.1), net als het originele onderzoek van Ly et al (2018). In het derde (en vierde) experiment bestaat de training set uit 2035 dagen van de beschreven periode en de test set uit de laatste 200 dagen (Figuur 5.5). Het script van voor het opstellen van deze modellen en de gebruikte datasets zijn te vinden op https://github.com/aithesis3538060.

Figuur 5.5. Grafiek van de prijs van Bitcoin in USD tussen de periode van 28 april 2013 en 10 juni 2019.

5.5 Hyperparameter optimalisatie

Hyperparameters zijn waarden die gebruikt worden om een model bij te stellen om te voorkomen dat het model under- of overfit raakt. Deze waarden worden niet door het model zelf aangepast. Voor het optimaliseren van het model zijn verschillende hyperparameters in het script aanwezig. Het gaat om het aantal epochs, het aantal neuronen dat een verborgen laag heeft, het aantal lagen dat gebruikt wordt en de dropout rate. Daarnaast kan ook de batch size en learning rate aangepast worden. Deze hyperparameters worden hieronder nader toegelicht. Om de beste parameters te bepalen is, zoals in paragraaf 5.4 vermeld is, crossvalidatie gebruikt. Met behulp van de Python bibliotheek Talos zijn de best presterende hyperparameters gezocht over een subset van de hyperparameter ruimte. De hyperparameter ruimte bestaat uit alle mogelijke combinaties van gegeven hyperparameters. Er zijn verschillende waarden per type hyperparameter ingevoerd, waarop Talos het model traint voor elke verzameling hyperparameters in de willekeurig gekozen subset van de hyperparameter ruimte. Het proces voor het kiezen van de subset wordt aan het einde van de paragraaf toegelicht.

De learning rate aanpassen zorgt ervoor dat de mate waarin gewichten tijdens het trainen van het model groter of kleiner wordt. Een grotere learning rate zorgt ervoor dat er minder training epochs nodig zijn omdat de gewicht aanpassingen groter zijn, maar het kan er ook voor zorgen dat het model niet de beste gewichten kan vinden doordat het te grote sprongen in gewicht toe- of afname maakt. Daarentegen kan een te kleine learning rate ervoor zorgen dat de gewicht aanpassingen te klein zijn om tot de juiste gewichten te komen, waardoor er meer epochs nodig zijn (Kaastra & Boyd, 1996).

De dropout voorkomt dat het model overfit raakt door een bepaald percentage neuronen te laten vallen tijdens een epoch (Srivastava et al., 2014). Het laten vallen van een neuron betekent dat deze neuron en zijn inkomende en uitgaande verbindingen tijdelijk genegeerd wordt, waardoor andere neuronen zich hierop moeten aanpassen. Het negeren van een neuron zorgt voor ruis (Engels: noise), wat het model minder laat overfitten. Omdat er gebruik wordt gemaakt van relatief kleine datasets, is het waarschijnlijk dat het gebruiken van een dropout laag de nauwkeurigheid van het model ten goede komt (Li., Chen, Hu & Yang, 2019).

Het aantal epochs is het aantal keer dat een model een training dataset compleet doorloopt. De batch size is het aantal samples, in dit geval inputreeksen met prijzen en eventuele features, die verwerkt worden door het model tijdens een epoch voordat de gewichten aangepast worden.

Voor de learning rate van de optimizer Adam (Keras-Team, 2019) zijn de 100 waarden tussen 0.001 en 0.1 gekozen, waarvan 0.001 de default waarde is van de optimizer. Op basis van de vermelde richtlijnen die in paragraaf 4.4 besproken zijn, is er gekozen om voor de variablen n en m respectievelijk 1500 en 1 te nemen. Hiermee is berekend dat er in de eerste verborgen laag maximaal 100 neuronen en in de tweede verborgen laag maximaal 20 neuronen mogen bevinden. Voor de eerste verborgen laag zijn 10 waarden tussen 1 en 100 neuronen gekozen, voor de tweede verborgen laag zijn dit 10 waarden tussen 1 en 20 neuronen. Voor de batch size is gekozen voor een grootte van 32 en 64**.** Verder is het model telkens getraind op 10 waarden tussen 10 en 100 epochs en met een dropout van 10 waarden tussen 0 en 0.1.

Er zijn in totaal $2 * 10^6$ verschillende combinaties aanwezig voor elke parameter, wat betekent dat in totaal 10^7 verschillende modellen getraind moeten worden voor elk model, als elke combinatie over elke split uitgevoerd moet worden. Omdat het weken zou duren om de beste hyperparameters te vinden per vraagstuk, gebruikt dit onderzoek een optimalisatie strategie in Talos. Deze strategie neemt 0.01% van alle mogelijke variaties, zodat er enkel 200 variaties overblijven per model. Het vinden van de beste hyperparameters onder de gekozen variaties kost hiermee ruim 10 uur per model. Op basis van de gemiddelde waarden over de splits per gebruikte hyperparameter combinatie zijn uiteindelijk de beste hyperparameters gekozen om de modellen op te trainen.

Uiteindelijk is er per experiment voor elk van de drie typen neurale netwerken, na hyperparameter optimalisatie, een model opgesteld. Voor ieder model is de gemiddelde validatie loss over alle splits berekend om te bepalen of het model onder de gebruikte hyperparameters goed heeft gepresteerd tijdens de crossvalidatie. Vervolgens is er per model de best presterende hyperparameter samenstelling geselecteerd om het model te trainen en te valideren. De resultaten zijn in hoofdstuk 6 gepresenteerd.

6 Resultaten

In dit hoofdstuk worden de resultaten van het experimenteel onderzoek weergeven. Voor elk neuraal netwerk zijn de gevonden hyperparameters van de hyperparameter optimalisatie (paragraaf 5.5) genoteerd ter reproductie van de resultaten. Er is geen analyse uitgevoerd op mogelijke verbanden tussen de gevonden hyperparameters en de prestatie van een neuraal netwerk. Daarnaast is er voor ieder netwerk zowel de behaalde training als de test loss van de voorspellingstaak vermeld.

In het paper van Ly et al. gaf hun prijsvoorspellingsmodel de beste resultaten na 10 epochs. De beste resultaten behaalden zij met de optimizer Stochastic Gradient Descent (SGD), waarbij een training loss van 0.0135 werd behaald en een test loss van 0.02809. Het gebruiken van de optimizers RMSProp en Adam gaf een training los van respectievelijk 0.0160 en 0.0189. Het onderzoek noemde bij deze optimizers niet hoe hoog de test loss was, maar vermelden wel dat bij alle modellen de test loss hoger was dan de training loss.

Er is voor dit onderzoek gekozen om de optimizer Adam te gebruiken, omdat literatuur (Kingma & Ba, 2014) erop wijst dat deze optimizer het beste presteert bij neurale netwerken wanneer er gekeken wordt naar de balans tussen snelheid en nauwkeurigheid.

De baseline die gebruikt is bestaat uit de prijs waarden van een week geleden. Dit is de meest simpele vorm van het SMA (Engels: Simple Moving Average), die bestaat uit het gemiddelde van de som van alle prijzen over *n* dagen (in dit geval is *n = 1*).

$$SMA = \frac{x^1 + x^2 + .. + x^{n-1} + x^n}{n}$$

Deze baseline wordt gebruikt om de nauwkeurigheid van de getrainde modellen te vergelijken, zodat er een uitspraak kan worden gedaan over de prestaties van deze modellen.

6.1 Model resultaten

In het eerste experiment is er een dataset gebruikt die gelijk is aan de dataset die gebruikt is in het onderzoek van Ly et al. (2018). Deze dataset bestaat uit de dagprijzen van Bitcoin over de periode van 28 april 2013 en 05 februari 2018. In het experiment van Ly et al. werd een standaard neuraal netwerk gebruikt om de prijs van Bitcoin van een week later te voorspellen. Een standaard en twee typen recurrent neurale netwerken zijn getraind op dezelfde dataset. Er is een baseline training loss van 0.01268 en een baseline test loss van **0.02746** gevonden.

Het standaard neurale netwerk behaalde een training loss van 0.00150 en een test loss van **0.02662**. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00062 en een test loss van **0.02501**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00081 en een test loss van **0.02642** behaald. Deze resultaten werden behaald met de onderstaande hyperparameters die gevonden zijn na hyperparameter optimalisatie.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	1	0	18	75	32	0.060
LSTM	40	0.07	10	50	32	0.003
GRU	60	0.02	16	30	64	0.030

In het tweede experiment werd er één externe variabele (feature) toegevoegd aan de dataset van het eerste experiment. Het toevoegen van een feature aan deze dataset leverde de volgende resultaten op, die hieronder zijn genoteerd per toegevoegde feature.

Als de feature **Market Capitalization** toegevoegd wordt aan de dataset, wordt er met een standaard neuraal netwerk een training loss van 0.00119 en een test loss van **0.02641** behaald. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00082 en een test loss van **0.02418**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00133 en een test loss van **0.02559**.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	20	0.04	8	20	32	0.025
LSTM	50	0.09	18	10	32	0.001
GRU	80	0.05	12	80	32	0.070

Als de feature **Payment Count** toegevoegd wordt aan de dataset, wordt er met een standaard neuraal netwerk een training loss van 0.00075 en een test loss van **0.02605** behaald. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00184 en een test loss van **0.02426**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00102 en een test loss van **0.02568**.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	30	14	10	60	32	0.020
LSTM	80	0.09	18	50	32	0.015
GRU	30	0.01	2	10	64	0.045

Als de feature **Realized Capitalization** toegevoegd wordt aan de dataset, wordt er met een standaard neuraal netwerk een training loss van 0.00094 en een test loss van **0.02598** behaald. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00113 en een test loss van **0.02417**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00072 en een test loss van **0.02541**.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	10	0.07	14	50	64	0.008
LSTM	20	0.03	10	30	32	0.003
GRU	80	0.07	14	10	32	0.055

In het derde experiment is er een dataset gebruikt die bestaat uit de dagprijzen van Bitcoin over een langere periode dan het eerste experiment. Deze dataset bestaat uit de dagprijzen van Bitcoin tussen de periode van 28 april 2013 en 10 juni 2019. Wederom zijn een standaard en twee typen recurrent neurale netwerken getraind op de dataset om de prijs van een week later te voorspellen. Er is een baseline training loss van 0.01400 en een test loss is **0.01064** gevonden.

Het standaard neurale netwerk behaalde een training loss van 0.01400 en een test loss van **0.01163**. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00052 en een test loss van **0.01061**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00050 en een test loss van

0.01057 behaald. Deze resultaten werden behaald met de onderstaande hyperparameters die gevonden zijn na hyperparameter optimalisatie.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	10	0.05	12	60	32	0.050
LSTM	80	0.05	18	80	64	0.020
GRU	40	0.03	14	70	32	0.005

In het vierde experiment is er één externe variabele (feature) toegevoegd aan de dataset van het derde experiment. Het toevoegen van een feature aan deze dataset levert de volgende resultaten op, die hieronder zijn genoteerd per toegevoegde feature.

Als de externe variabele **Market Capitalization** toegevoegd wordt aan de dataset, wordt er met een standaard neuraal netwerk een training loss van 0.00081 en een test loss van **0.01060** behaald. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00050 en een test loss van **0.01048**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00148 en een test loss van **0.01047**.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	30	0.04	8	40	32	0.035
LSTM	80	0.05	18	80	64	0.020
GRU	40	0.03	14	70	32	0.005

Als de externe variabele **Payment Count** toegevoegd wordt aan de dataset, wordt er met een standaard neuraal netwerk een training loss van 0.00098 en een test loss van **0.01020** behaald. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00057 en een test loss van **0.00984**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00056 en een test loss van **0.00974**.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	60	0.08	14	50	64	0.020
LSTM	80	0.05	18	80	64	0.020
GRU	40	0.03	14	70	32	0.005

Als de externe variabele **Realized Capitalization** toegevoegd wordt aan de dataset, wordt er met een standaard neuraal netwerk een training loss van 0.00189 en een test loss van **0.01043** behaald. Het LSTM recurrent neurale netwerk behaalde een training loss van 0.00053 en een test loss van **0.00947**. Het GRU recurrent neurale netwerk behaalde een training loss van 0.00130 en een test loss van **0.01003**.

	1ste verborgen laag	dropout	2de verborgen laag	Aantal epochs	Batch size	Learning rate
NN	70	0.04	12	90	32	0.010
LSTM	60	0.06	14	40	64	0.006
GRU	70	0.05	18	10	32	0.008

7 Conclusie

7.1 Analyse resultaten

In het experimenteel onderzoek zijn drie typen neurale netwerken getraind op meerdere datasets. In het eerste experiment blijkt dat van de geteste neurale netwerken het LSTM netwerk het beste presteert, gevolgd door het GRU netwerk en het standaard neurale netwerk (respectievelijk 0.02501, 0.02642 en 0.02662). Alle drie de netwerken presteren in dit experiment beter dan de baseline (0.02746). Opvallend is dat het beste resultaat uit het onderzoek van Ly et al. lager uitvalt dan de baseline die in dit experiment gevonden is (0.02809). Een reden hiervoor kan zijn dat er in dit experiment andere hyperparameters zijn gebruikt voor het model van het standaard neuraal netwerk. Opmerkelijk is dat het GRU netwerk niet veel beter presteert dan het standaard neuraal netwerk in dit experiment. Dit gaat tegen de verwachtingen van de gevonden literatuur (Torres en Qiu, 2018), waar de prestaties van een GRU netwerk met die van een LSTM netwerk vergelijkbaar gevonden wordt.

Het (logische) vermoeden is dat alle neurale netwerken beter presteren als ze getraind zijn op een grotere dataset. Dit blijkt inderdaad het geval te zijn. De dataset in het derde experiment omvatte namelijk een extreme prijsstijging die eind 2018 plaatsvond. Deze prijsstijging valt in de validatie set van het eerste experiment, wat betekent dat de modellen in dit experiment hier niet op getraind zijn en mogelijk deze grote prijsstijgingen niet kunnen voorspellen. In het derde experiment zijn de modellen wel getraind op deze prijsstijging. Het GRU netwerk behaalt in dit experiment de beste test loss (0.01057). Het LSTM netwerk heeft niet veel slechter gepresteerd op deze dataset (0.01061). Het verschil tussen deze twee recurrent neurale netwerken en de gevonden baseline (0.01064) is echter kleiner dan het verschil tussen de baseline en de twee recurrent neurale netwerken in het eerste experiment. Verder is opvallend dat het standaard neuraal netwerk een slechtere test loss levert dan de baseline in dit experiment. Het standaard neuraal netwerk model behaalde een test loss van 0.01163, wat niet strookt met de resultaten uit het eerste experiment. Een mogelijke reden hiervoor is dat er zich geen goede hyperparameters bevinden binnen de subset van de hyperparameter ruimte, waardoor een model met de best presterende hyperparameters uit deze subset de test set niet goed kon valideren.

Het tweede en vierde experiment maakten gebruik van de dataset uit respectievelijk het eerste en derde experiment met een toegevoegde feature. Het blijkt dat de beste test loss in het vierde experiment werd behaald door het LSTM netwerk met de feature *Realized Capitalization*, een alternatieve berekening van de totale marktwaarde (0.00947). Dit is een groot verschil met de score die behaald werd als het netwerk getraind werd met enkel de dagprijzen van Bitcoin. Er werd met het LSTM netwerk ook een verbetering in de behaalde test loss geobserveerd met de features *Market Capitalization* (de totale marktwaarde van Bitcoin; 0.01048) en *Payment Count* (de totale opbrengst van alle Bitcoin mijners op een dag; 0.00984). Deze verbetering in test loss met de feature *Realized Capitalization* werd ook geconstateerd in het tweede experiment, waarbij de test loss van het LSTM netwerk verbeterde van 0.02501 (in het eerste experiment) naar 0.02417.

Het tweede experiment toonde aan dat, net als in het vierde experiment, het gebruik van een feature in de dataset de test loss positief beïnvloedde in alle typen netwerken. In verhouding met het eerste experiment was de grootste prestatie verbetering van het GRU netwerk ten opzichte van het standaard neurale netwerk te zien bij het gebruik van de feature *Payment Count* (respectievelijk van 0.02642 naar 0.02568 en van 0.02662 naar 0.02605). In het tweede experiment behaalde het LSTM netwerk de beste resultaten, ongeacht de gebruikte feature. De prestaties van het LSTM netwerk in alle

experimenten komen overeen met wat Alessandretti et al. (2018) beargumenteren, namelijk dat een LSTM netwerk in staat is de prijzen van Bitcoin te benaderen.

Het is interessant te vermelden dat het onderzoek van Raghava-Raju (2018) de features Market Capitalization en Payment Count (respectievelijk btc_market_cap en btc_miners_revenue genoemd) verwijderde uit de dataset van zijn onderzoek. De reden hiervoor is dat deze features een te hoge correlatie zouden hebben met de Bitcoin prijs en kunnen leiden tot overfitting. Deze features zijn in de onderzoeken van Alessandretti et al. (2018) en Jang & Lee (2017) wel weer met succes gebruikt. In deze scriptie hebben de gebruikte features de laagste correlatie met de Bitcoin prijs.

Uit het uitgevoerde experimenteel onderzoek blijkt dat het gebruik van features de nauwkeurigheid bij het voorspellen van Bitcoin verbetert bij het gebruik van een neuraal netwerk, waarbij de beste resultaten behaald worden met de feature *Payment Count* en *Realized Capitalization.* Daarnaast lijken de beste prestaties voor de voorspelling van de Bitcoin prijs geproduceerd te worden door het LSTM netwerk, hoewel er ook gelijkwaardige prestaties worden behaald met een GRU netwerk. Dit komt overeen met de conclusie uit het onderzoek van Torres & Qiu (2018). Het blijkt dat een standaard neuraal netwerk niet goed in staat is om de prijs te voorspellen op basis van enkel de prijs van Bitcoin, maar dat het gebruik van features de resultaten positief kunnen beïnvloeden.

Experiment	Feature	NN Test Loss	LSTM Test Loss	GRU Test Loss
1	-	0.02662	0.02501	0.02642
2	Market Capitalization	0.02641	0.02418	0.02559
	Payment Count	0.02605	0.02426	0.02568
	Realized Capitalization	**0.02598**	**0.02417**	**0.02541**
3	-	0.01163	0.01061	0.01057
4	Market Capitalization	0.01060	0.01048	0.01047
	Payment Count	**0.01020**	0.00984	**0.00974**
	Realized Capitalization	0.01043	**0.00947**	0.01003

Figuur 7.1. Resultaten van de modellen.

7.2 Conclusie

Uit het onderzoek is gebleken dat de prestaties van een standaard neuraal netwerk minder nauwkeurig zijn in het voorspellen van de prijs van Bitcoin ten opzichte van de prestaties van beide typen recurrent neurale netwerken. Het onderzoek heeft daarnaast aangetoond dat het gebruik van features van groot belang zijn voor het optimaliseren van prijsvoorspelling middels neurale netwerken. De hoogste nauwkeurigheid werd in het onderzoek behaald door het LSTM netwerk, wat overeenkomt met voorgaand onderzoek van McNally et al. (2018), Wu et al. (2018) en Torres & Qiu (2018).

Het uitgevoerde onderzoek toont daarnaast aan dat de prestaties van een GRU netwerk vergelijkbaar zijn met die van een LSTM netwerk. Het voordeel van het gebruik van een GRU netwerk ten opzichte van een LSTM netwerk is dat een GRU netwerk minder poorten bezit en hiermee minder berekeningen nodig heeft (paragraaf 3.5). Dit betekent dat GRU netwerken sneller getraind zijn zonder dat er veel aan nauwkeurigheid ingeleverd wordt.

Al met al is er uit dit onderzoek gebleken dat de nauwkeurigheid van het voorspellen van de Bitcoin-prijs afhankelijk is van het type neurale netwerk, de gebruikte dataset en of deze dataset features bevat.

8 Discussie

8.1 Beperkende factoren

Er zijn enkele beperkende factoren die de resultaten van het onderzoek mogelijk hebben beïnvloed. De rekenkracht die nodig is om een recurrent neuraal netwerk model te trainen loopt snel op bij het gebruik van grotere datasets. Ook het trainen van een netwerk over meer epochs en het gebruiken van meerdere features in een dataset leidt tot meer berekeningen en daarmee tot een langere trainingstijd. Elk ander wetenschappelijk onderzoek naar neurale netwerken maakt gebruik van een grafische kaart. Helaas was het gebruik van een grafische kaart niet mogelijk, waardoor de hyperparameter optimalisatie van een simpel model al snel meer dan zes uur kon duren. Dit heeft er eveneens toe geleid dat er niet gekeken kon worden naar kleinere tijdstappen. Hierdoor is de keuze gemaakt om enkel naar dagprijzen van Bitcoin te kijken. Het zou interessant geweest zijn om te onderzoeken of het gebruik van kleinere tijdstappen in een dataset invloed heeft op de nauwkeurigheid van voorspellen. Daarnaast hebben de beperkingen in rekenkracht ertoe geleid dat ik niet meer features tegelijkertijd in een dataset heb kunnen toevoegen, een onderwerp dat ik graag verder had willen verkennen. Een laatste beperking aan dit onderzoek is de feature selectie. De features zijn gekozen op basis van de minste correlatie hebben met de prijs. Het onderzoek zou mogelijk beter zijn geweest wanneer de features waren gekozen op basis van andere criteria.

8.2 Vervolgonderzoek

Het uitgevoerde onderzoek geeft veel ruimte voor vervolgonderzoek. In deze scriptie is er bijvoorbeeld enkel gekeken naar de voorspelde Bitcoin prijs over een week. Onderzoek naar de relatie tussen nauwkeurigheid en het effect van een kortere of langere periode van voorspelling is een gebied wat ik graag verder zou willen verkennen. Het effect van technische indicatoren (technische analyse, hoofdstuk 4) op de nauwkeurigheid van prijsvoorspelling is een onderwerp waar al veel over geschreven is. Ik heb echter nog geen onderzoek kunnen vinden waarbij een GRU netwerk is gebruikt met deze features. Dit biedt dus mogelijkheden voor vervolgonderzoek.

Nieuwe recurrent neurale netwerken zoals de Minimal Gated Unit (MGU) zijn verder nog weinig onderzocht, dit zou ook een goed uitgangspunt zijn om vervolgonderzoek naar te doen. Ten slotte zijn er nog veel andere features die onderzocht kunnen worden om het effect op de prijsvoorspelling te analyseren, waaronder de prijs van andere cryptovaluta's en het aantal Facebook-, Instagram- en Twitter-berichten over Bitcoin.

Literatuurlijst

Abonazel, M. R., & Abd-Elftah, A. I. (2019). Forecasting Egyptian GDP Using ARIMA Models.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. Journal of Applied Mathematics, 2014.

Alessandretti, L., ElBahrawy, A., Aiello, L. M., & Baronchelli, A. (2018). Anticipating cryptocurrency prices using machine learning. Complexity, 2018.

Azoff, E. M. (1994). Neural network time series forecasting of financial markets. John Wiley & Sons, Inc..

Bakar, N. A., & Rosbi, S. (2017). Autoregressive integrated moving average (ARIMA) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction. International Journal of Advanced Engineering Research and Science, 4(11).

Bilberto Batres-Estrada. “Deep learning for multivariate financial time series

Borovykh, A., Oosterlee, C. W., & Bohte, S. M. (2019). Generalisation in fully-connected neural networks for time series forecasting. arXiv preprint arXiv:1902.05312.

Box, G. E., & Jenkins, G. M. (1976). Time series analysis: Forecasting and control San Francisco. Calif: Holden-Day.

Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 535-541). ACM.

Clements, M. P., Franses, P. H., & Swanson, N. R. (2004). Forecasting economic and financial time-series with non-linear models. International Journal of Forecasting, 20(2), 169-183.

CoinMarketCap - Cryptocurrency Market Capitalizations. (2019). Geraadpleegd van https://coinmarketcap.com/.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

DeVries, P. D. (2016). An Analysis of Cryptocurrency, Bitcoin, and the Future. International Journal of Business Management and Commerce, 1(2), 1-9.

Dritsaki, C. (2015). Forecasting real GDP rate through econometric models: an empirical study from Greece. Journal of International Business and Economics, 3(1), 13-19.

EFavDB. (2018, June 03). EFavDB/linselect. Geraadpleegd van https://github.com/efavdb/linselect.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Franses, P. H., & van Dijk, D. (2000). Non-linear time series models in empirical finance. Cambridge University Press.

Gençay, R., & Liu, T. (1997). Nonlinear modelling and prediction with feedforward and recurrent networks. Physica D: Nonlinear Phenomena, 108(1-2), 119-134.

Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01 (pp. 193-200). Springer, London.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning.

Gurney, K. (2014). An introduction to neural networks. CRC press.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München, 91(1).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.

Hsu, D. (2017). Time series forecasting based on augmented long short-term memory. arXiv preprint arXiv:1707.00666.

Huang, G. B. (2003). Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Transactions on Neural Networks, 14(2), 274-281.

Jang, H., & Lee, J. (2017). An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. Ieee Access, 6, 5427-5437.

Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. Neurocomputing, 10(3), 215-236.

Kamijo, K. I., & Tanigawa, T. (1990, June). Stock price pattern recognition - a recurrent neural network approach. In 1990 IJCNN International Joint Conference on Neural Networks (pp. 215-221). IEEE.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. Expert systems with Applications, 38(5), 5311-5319.

Keras-Team. (2019). Keras-team/keras. Geraadpleegd van https://github.com/keras-team/keras/blob/master/keras/optimizers.py#L436.

Kim, Y. B., Kim, J. G., Kim, W., Im, J. H., Kim, T. H., Kang, S. J., & Kim, C. H. (2016). Predicting fluctuations in cryptocurrency transactions based on user comments and replies. PloS one, 11(8), e0161197.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Kröse, B. & Smagt, P. (1993). An introduction to neural networks.

Kuan, C. M., & Liu, T. (1995). Forecasting exchange rates using feedforward and recurrent neural networks. Journal of applied econometrics, 10(4), 347-364.

Ledger. Hack Flasback: The Mt.Gox Hack - The Most Iconic Exchange Hack. (15 Mei 2019 Geraadpleegd van https://www.ledger.com/hack-flasback-the-mt-gox-hack-the-most-iconic-exchange-hack/

Li, X., Chen, S., Hu, X., & Yang, J. (2019). Understanding the disharmony between dropout and batch normalization by variance shift. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2682-2690).

Ly, B., Timaul, D., Lukanan, A., Lau, J., & Steinmetz, E. (2018). Applying Deep Learning to Better Predict Cryptocurrency Trends.

McNally, S., Roche, J., & Caton, S. (2018, March). Predicting the price of Bitcoin using Machine Learning. In 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP) (pp. 339-343). IEEE.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Nelson, M., Hill, T., Remus, B., O'Connor, M., 1994. Can neural networks be applied to time series forecasting and learn seasonal patterns: An empirical investigation. In: Proceedings of the Twenty seventh Annual Hawaii International Conference on System Sciences, pp. 649–655.

Phaladisailoed, T., & Numnonda, T. (2018). Machine Learning Models Comparison for Bitcoin Price Prediction. In 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE) (pp. 506-511). IEEE.

Radityo, A., Munajat, Q., & Budi, I. (2017, October). Prediction of Bitcoin exchange rate to American dollar using artificial neural network methods. In 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 433-438). IEEE.

Raghava-Raju, A. A Machine Learning Approach to Forecast Bitcoin Prices. International Journal of Computer Applications, 182(24).

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

Siami-Namini, S., & Namin, A. S. (2018). Forecasting economics and financial time series: Arima vs. lstm. arXiv preprint arXiv:1803.06386.

Socaci, I. A., & Lemnaru, C. (2018, December). A Comparative Study of Methods Used in the Analysis and Prediction of Financial Data. In International Conference on Mining Intelligence and Knowledge Exploration (pp. 309-320). Springer, Cham.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Steurer, E., & Hann, T. H. (1995). Exchange rate forecasting comparison: Neural networks symbolic machine learning and linear models. In Neural Networks in Financial Engineering, Proceedings of the Third International Conference on Neural Networks in the Capital Markets. World Scientific, London (pp. 113-121).

Taylor, S. J. (2008). Modelling financial time series. world scientific.

Terui, N., & Van Dijk, H. K. (2002). Combined forecasts from linear and nonlinear time series models. International Journal of Forecasting, 18(3), 421-438.

Torres, D. G., & Qiu, H. O. N. G. L. I. A. N. G. (2018). Applying Recurrent Neural Networks for Multivariate Time Series Forecasting of Volatile Financial Data.

Venema, R. S. (1999). Aspects of an integrated neural prediction system.

Verma, K., & Singh, P. K. (2015). An insight to soft computing based defect prediction techniques in software. International Journal of Modern Education and Computer Science, 7(9), 52.

Wu, C. H., Lu, C. C., Ma, Y. F., & Lu, R. S. (2018, November). A New Forecasting Framework for Bitcoin Price with LSTM. In 2018 IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 168-175). IEEE.

Yao, J., Tan, C. L., & Poh, H. L. (1999). Neural networks for technical analysis: a study on KLCI. International journal of theoretical and applied finance, 2(02), 221-241.