Over lerende machines en trekkende vogels

Kan een machine learning model vogeltrek nauwkeurig voorspellen?

Gerwin Kuijntjes | André Vis | Lauren Zwemer

08/01/2021 - Begeleider: Drs. Ir. G. G. van den Top

ABSTRACT

In dit onderzoek zijn de dagtotalen per soort van een selectie aan telposten uit de database van de organisatie Trektellen over de periode van 1999-2019 gebruikt om een zo accuraat mogelijk voorspelling te doen over vogeltrek op basis van data van windrichting, windsnelheid, neerslag en temperatuur, afkomstig van het KNMI. Er is een model ontwikkeld dat gebaseerd is op een Artificial Neural Network met drie *hidden layers*. Om rekening te houden met de *overfitting* die bleek uit de training met de meteorologisch data, zijn er twee *dropout layers* toegevoegd. Een analyse van het model liet een nauwkeurigheid van 34,7% zien. Daarom concluderen wij dat ons model gebruikt kan worden om een indicatie te geven over vogeltrek op soortniveau, maar niet accuraat genoeg is voor een betrouwbare voorspelling. Op basis van literatuur en proeftrainingen zijn er geen aanwijzingen te vinden die wijzen op significante gebreken in het model. De oorzaak van de onnauwkeurigheid van het model is hoogst waarschijnlijk de manier waarop het model de gegeven data verwerkt, wat in vervolgonderzoeken verbeterd kan worden.

ABSTRACT (ENGLISH)

In this research daily species total, from the online database of the organisation Trektellen between 1999 and 2019 from specific migration sites, are used to develop a, as accurate as possible, forecast of bird migration based on data on wind direction, wind speed, precipitation and temperature retrieved from KNMI (Royal Netherlands Meteorological Institute). A model based on an Artificial Neural Network with three *hidden layers* was constructed. To account for apparent *overfitting* during training with data on atmospheric conditions, two *dropout layers* were added. Analyses of the model revealed an accuracy of 34.7%. Therefore, we conclude that our model can be used to indicate the bird migration on species level, but not an accurate forecast. Based on literature and test processes there are no indications to assume significant model imperfections. The cause of the inaccuracy of the model is most likely the way in which our model processes the given data, which could be improved in further research.

INHOUD

1 INLEIDING

Elk jaar komt het resultaat van ruim 30.000 uren aan telling van de vogeltrek in de database van de organisatie Trektellen (Troost, 2020b; Troost & Boele, 2020), waardoor deze database erg waardevol is voor onderzoek naar de vogeltrek. Deze trek hangt af van onder andere de periode van het jaar, iets wat al in de klassieke oudheid bekend was (Aristoteles, 350 B.C.E.), maar ook van abiotische factoren zoals windrichting, windsnelheid, neerslag en temperatuur (Berthold, 2001, para. 6.19; Dijksterhuis, 2018; Kemp, Shamoun-Baranes, van Gasteren, Bouten, & van Loon, 2010). Ondanks de nieuwe ontwikkelingen met betrekking tot het voorspellen van vogeltrek zijn er nog geen accurate modellen die op soortniveau een voorspelling kunnen doen van de trek over Nederland (Van Belle, Shamoun-Baranes, Van Loon, & Bouten, 2007; Van Doren & Horton, 2018).

Voorspellingen over de vogeltrek over Nederland hebben meerdere belangrijke toepassingen. Ten eerste is een voorspelling van de trek over Nederland van belang voor de veiligheid van militaire en civiele luchtvaart (Bouten et al., 2008; Shamoun-Baranes, Van Gasteren, & Ross-Smith, 2018; Van Gasteren & Dekker, 2005). Daarnaast is het ook van belang voor het reduceren van slachtoffers van botsingen tegen windturbines tijdens de trek (Krijgsveld, Kleyheeg-Hartman, Klop, & Brenninkmeijer, 2020). Voor deze twee toepassingen worden al radars gebruikt (Bouten et al., 2020; Van Belle et al., 2007), maar in het geval van de luchtvaart is het nuttig om nog meer inzicht te krijgen in vogeltrek (European Aviation Safety Agency, 2009). Bij windturbines kunnen voorspellingen over heel Nederland van nut zijn, omdat de radarsystemen prijzig zijn en maar voor een klein gebied een voorspelling doen (Bouten et al., 2020). Verder kan de voorspelling gebruikt worden door vogelonderzoekers en trektellers. Als laatste kan het ook gebruikt worden voor het in kaart brengen van de verspreiding van zoönosen die door vogels worden verspreid (Bauer et al., 2017). Door de grote aantallen trekkende vogels en de complexiteit rondom het doen van een accurate voorspelling zijn er nog geen betrouwbare voorspellingen over de trek van vogels over Nederland op soortniveau.

Door de verbeteringen in algoritmes voor het verwerken van grote hoeveelheden data en de verhoogde rekenkracht van de huidige computers (Dokter et al., 2011; Nilsson et al., 2019), is het mogelijk om een voorspelling van vogeltrek te doen op basis van de data van organisatie Trektellen. *Machine learning* door middel van een *Artificial Neural Network* (ANN) kan complexe voorspellingen met meerdere parameters doen. Een ANN is geprogrammeerd om zélf relaties tussen bepaalde parameters en de referentiedata te berekenen, terwijl dat bij andere modellen zoals een *regression* model of een *concept driven* model niet zo is. Daarnaast vereist een ANN minder voorkennis en is het geen slechter medium om vogeltrek te voorspellen (Bouten et al., 2005). Daardoor hebben wij de mogelijkheid om met de data van organisatie Trektellen in combinatie met de meteorologische data van het KNMI een voorspelling te doen van de vogeltrek.

In dit onderzoek ontwikkelen we, met behulp van meteorologische data en dagtotalen van 1999 tot en met 2019, een model, op basis van een ANN, dat een zo accuraat mogelijke voorspelling van het aantal trekkende individuen per soort op basis van abiotische factoren kan doen.

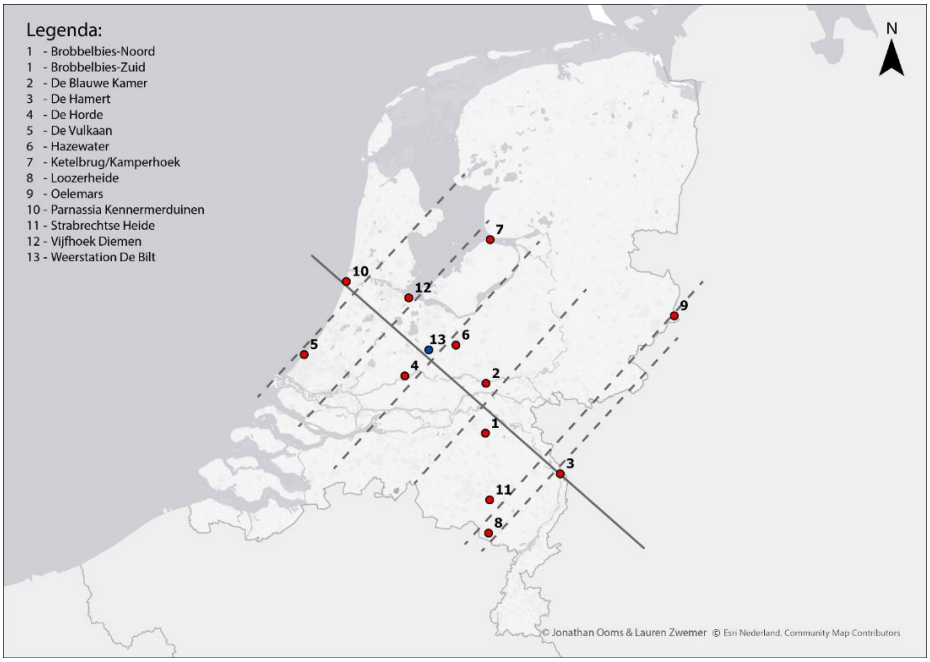
2 MATERIAAL EN METHODE

2.1 BIOLOGISCHE DATA

Om het model een zo accuraat mogelijke voorspelling te laten doen over de soorten en aantallen vogels die zullen vliegen, wordt gebruik gemaakt van gegevens uit de database van de organisatie Trektellen (Troost, 2020b; Troost & Boele, 2020). We maken gebruik van de dagtotalen van de tellingen uit 1999 tot en met 2019 van de soorten die geteld werden, volgens de methode beschreven door Van Dijk (2008).

De geselecteerde telposten zijn te zien in Figuur 1 en in Bijlage 2: Telposten. Voor de keuze van de telposten is rekening gehouden met het totaal aantal teluren, het aantal jaren dat de telpost actief is, of het tellen door meerdere tellers is gedaan en het totaal aantal soorten dat is waargenomen (Troost, 2020a), om zo de kwaliteit en de kwantiteit van de data in balans te houden.

Deze telposten zijn ook gekozen om te kunnen berekenen hoeveel vogels er over Nederland vliegen. Met het wegtrekken in het najaar hebben de vogels die over Nederland vliegen overwegend een zuidwestelijke richting (Nilsson et al., 2019) en in het voorjaar is deze richting tegengesteld (Berthold, 2001). Hierdoor zullen de vogels ongeveer door de denkbeeldige lijn die van noordwest naar zuidoost loopt (zie Figuur 1) vliegen (Van Gasteren, Troost, & Boele, 2015). De richting waarin de vogels vliegen is echter wel variabel (Nilsson et al., 2019) en er wordt niet elke dag op de telposten geteld. Door de dagtotalen per soort van twee telposten (met een uitzondering bij *Brobbelbies-Noord* en *-Zuid*, zie hieronder) te middelen, gelijkend op de methode toegepast door Jaffré et al. (2013), wordt een accurater beeld verkregen van de vogeltrek door de lijn van noordwest naar zuidoost. Om deze reden hebben we ook gekozen voor telposten met minder actieve jaren en teluren (zie Bijlage 2: Telposten, Tabel 1). *Brobbelbies-Noord* en *-Zuid* worden met elkaar gemiddeld alvorens te worden gemiddeld met de *Blauwe Kamer*, omdat *Brobbelbies-Noord* alleen actief is in het najaar (Vogelwacht Uden e.o., 2010). De 'gemiddeldes van de haakse lijnen' worden op hun beurt weer gemiddeld zodat er een gemiddelde uit komt voor een willekeurig punt op de lijn van noordwest naar zuidoost.



Figuur 1 Telposten en weerstation waarvan de data zijn gebruikt in dit onderzoek met daarbij de lijn van noordwest naar zuidoost en de lijnen die daar haaks op staan, die gebruikt zijn voor de keuze van de telposten en het weerstation.

2.2 METEOROLOGISCHE DATA

De abiotische factoren die gebruikt zijn voor het model komen van meteorologische daggegevens van het KNMI (Koninklijk Nederlands Meteorologisch Instituut (KNMI), 2020). We hebben gekozen voor de abiotische factoren wind (DDVEC en FHVEC), temperatuur (TG) en neerslag (TH), omdat wind en neerslag de belangrijkste factoren zijn voor trekactiviteit (Berthold, 2001, par. 6.19) en temperatuur een direct effect heeft op de aanwezigheid van voedsel, wat voor veel vogels een drijfveer is om te trekken (Berthold, 2001, Chapter 2). De data zijn afkomstig van weerstation De Bilt, omdat dit station ongeveer in het midden van de lijn van noordwest naar zuidoost ligt (zie Figuur 1).

2.3 MODEL

Om het *ANN* model te ontwikkelen wordt gebruikt gemaakt van de programmeertaal Python (Python Software Foundation, 2020). In Python zal onder andere gebruik worden gemaakt van de libraries NumPy (Oliphant, 2020), pandas (McKinney, 2020), Matplotlib (Hunter, 2020), Pastas (Collenteur, Bakker, Caljé, Klop, & Schaars, 2019) en TensorFlow (Google Brain Team, 2020).

Voor het maken en trainen van het ANN is een viertal stappen nodig.

2.3.1 ANALYSEREN EN TRANSFORMEREN VAN DATA

Om later het model te kunnen trainen, moeten de data in de juiste vorm beschikbaar zijn. Daarom wordt een selectie gemaakt van de data die nodig zijn, die zo wordt omgevormd dat de selectie gebruikt kan worden voor het model.

2.3.2 HET MODEL ONTWERPEN

Een model bestaat uit meerdere *layers*. Een *input tensor* en een *output tensor* zijn vereist. Daarnaast bestaat de mogelijkheid om *hidden layers* toe te voegen. Een model heeft alleen *hidden layers* nodig als de relaties in de data niet-lineair zijn (Gad, 2018).

Een *tensor* is een vector of matrix van afmetingen die allerlei soorten van data vertegenwoordigt. De *input tensor* is vaak een *flatten layer*, die ervoor zorgt dat de *tensor* omgevormd wordt, zodat hij een vorm heeft die gelijk is aan het aantal elementen die de tensor moet bevatten. In de *input tensor* komt de dataset die verwerkt is binnen in het model, waarna een berekening in de neuronen plaats vindt. Het aantal neuronen dat de *input tensor* heeft hangt af van de vorm van de trainingsdata. Vervolgens geeft de *input tensor* de dataset door aan een andere groep neuronen: een *hidden layer*.

Hidden layers zorgen ervoor dat *ANN's* superieur zijn aan de meeste *machine learning* algoritmes. Bij de meeste problemen is een model met één *hidden layer* genoeg. Hoe meer *hidden layers* gebruikt worden, hoe langer het *neural network* erover doet om de *output* te berekenen, maar ook hoe meer complexe problemen opgelost kunnen worden door het *neural network*. De *input* van een *hidden layer* is gebruikelijke wijs een getal in de reeks 2^n (Google Brain Team, 2020). Het aantal *hidden layers* is dus een afweging tussen nauwkeurigheid en snelheid.

Vervolgens worden de data aan het model gekoppeld en wordt een activatieformule gekozen die bepaalt hoe de structuur van het model eruit gaat zien. Ook de *loss*-functie en de *optimizer* moeten worden bepaald. Deze twee functies bepalen de grootte van de *error* en de mate waarin de gradiënten aangepast moeten worden. Tot slot wordt aangegeven wat de *input* en wat de *output* moet zijn.

2.3.3 HET MODEL TRAINEN, DOOR MIDDEL VAN EEN DEEL VAN DE BESCHIKBARE DATA

Als het model geprogrammeerd is kan het worden getraind met een functie van Tensorflow. Dit gebeurt door een deel van de beschikbare data als *input* te laden in het model. Het model zal dat vergelijken met de *output*. Daarna zal het enkele waarden voor het berekenen van de *output* aanpassen. Dat is het werkelijke trainen. Het aantal keer dat het model met de hele dataset getraind heeft wordt het aantal *epochs* genoemd. Dat aantal moet worden bepaald aan de hand van '*trial and error*', om dan telkens te bekijken of de nauwkeurigheid toeneemt of niet.

2.3.4 HET MODEL TESTEN MET HET ANDERE DEEL VAN DE BESCHIKBARE DATA.

Nadat het model getraind is moet worden getest of het model daadwerkelijk accuraat is. Door maar een deel van de beschikbare data te gebruiken in stap 3, zijn nu geschikte data overgebleven om het model op te laten testen. Als het model goed getraind en nauwkeurig is zal het de verwachte *output* geven op de *input* van de testdata. Als het trainen klaar is wordt het model opgeslagen, zodat deze later geladen kan worden om te kunnen gebruiken.

3 RESULTATEN

3.1 DATA

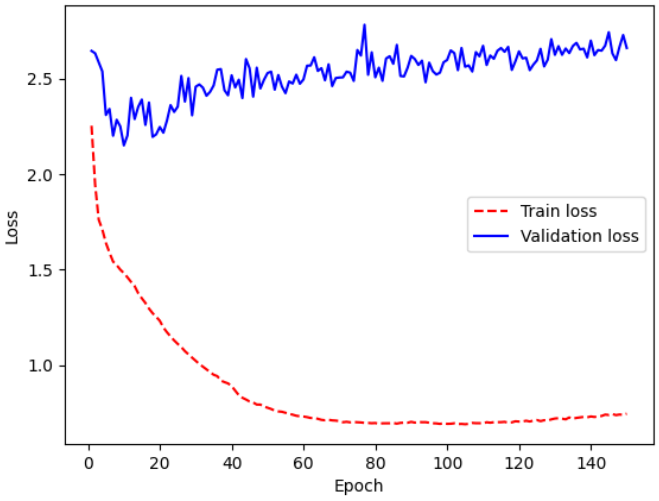
Voordat het model de weerdata en vogeltrekdata tot zich kan nemen worden de data verwerkt in drie stappen. Allereerst worden de trektelgegevens genormaliseerd, door de gegevens van vogelsoorten waar weinig data van beschikbaar zijn te verwijderen. Voor dit onderzoek is een selectie gemaakt van 64 soorten waarvan in totaal meer dan 20.000 individuen zijn opgenomen in de dataset. De dagtotalen van deze soorten zijn verwerkt zoals aangegeven in *paragraaf 2.1 Biologische data*. Daarna worden de tijd-[1] en windgegevens van de weerdata geconverteerd naar sinus en cosinus, omdat op deze manier de circulariteit van gegevens zoveel mogelijk gewaarborgd wordt. Tot slot zijn de tijden waarvoor geen vogeltrekdata van de geselecteerde soorten beschikbaar zijn, geëlimineerd uit de weerdata, met dat doeleinde dat de hoeveelheid significante *input* data en de hoeveelheid significante *target data* equivalent is. Zie Figuur 4 voor een voorbeeld van *input* data.

3.2 ONTWERP MODEL

Het ontwerpproces van het model verloopt in een vijftal stappen.

3.2.1 BEPALING VAN DE SOORT EN HOEVEELHEID LAYERS

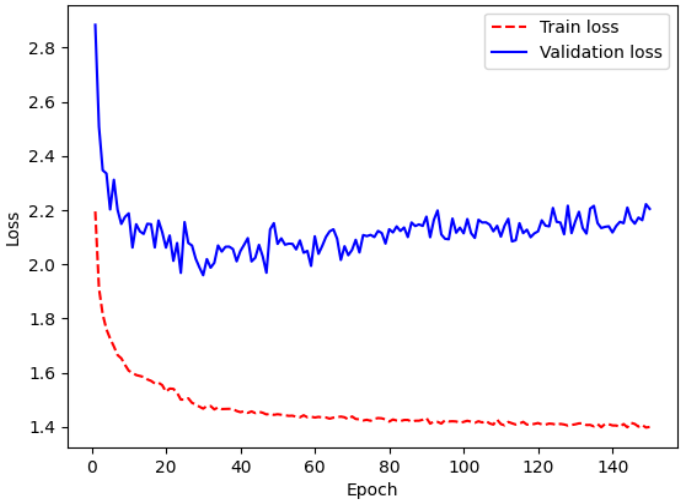
Voor het bouwen van een model zijn meerdere *layers* vereist (zie Figuur 4). De eerste *layer* in het model wordt de zogenaamde *input tensor* en is een *flatten layer*, waardoor de *tensor* wordt omgevormd. Het aantal neuronen dat de *input tensor* heeft hangt af van de vorm van de trainingsdata. Daarom krijgt de *input tensor* in dit model 30 neuronen, want het model krijgt dertig variabelen te verwerken. Vervolgens geeft de *input tensor* de dataset door aan een andere groep neuronen: een *hidden layer*. Vanwege de complexiteit van de data worden er bij dit onderzoek drie *hidden layers* gebruikt, elk met een aantal neuronen van 1024. Dat aantal *layers* en neuronen is als redelijke balans tussen nauwkeurigheid en snelheid gebaseerd op proeftrainingen van het model. Een grotere hoeveelheid had geen significant effect op de accuratie, maar wel op de snelheid van het model. Het verloop van de proeftrainingen is bijgehouden (zie Figuur 2). De *train loss* daalt, terwijl de *validation loss* stijgt. Dat betekent dat het model de trainingsdata steeds beter leert kennen, maar dat het model faalt om die opgedane kennis te generaliseren en toe te passen op onbekende data (Schlüter, 2019).



Figuur 2 De *train loss* en *validation loss* uitgezet tegen het aantal *epochs*. De *train loss* daalt, terwijl de *validation loss* juist toeneemt. Er is sprake van *overfitting*.

[1] Met 'tijd' wordt hier bedoeld: het meervoud van datum. In het vervolg zal dit ook worden aangeduid als tijd. Dit om verwarring tussen data (gegevens) en data (meervoud van datum) te voorkomen.

Om *overfitting* te reduceren hebben we getest wat het effect is van het reduceren van het aantal layers in het model, van het toevoegen van *dropout layers* en van op tijd te stoppen met trainen. Het resultaat daarvan is dat pas een significant verschil optreedt na aanpassing van het toevoegen van *dropout layers*. Daarom zijn twee *dropout layers* toegevoegd aan het model. Door '*trial and error*' is gekozen om *dropout-layer-1* een *rate* van 0,8 te geven en *dropout-layer-2* een *rate* van 0,2. Het effect van beide andere aanpassingen is verwaarloosbaar. Na het toevoegen van de *dropout layers* ziet de grafiek met daarin *train loss* en *validation loss* uitgezet tegen *epoch* eruit als in Figuur 3. Naast een *flatten layer*, *hidden layers* en *dropout layers* heeft het model ook een *output tensor* nodig. Deze neemt de waarden die berekend zijn door de *hidden layers* waarna ze gebruikt worden voor de voorspelling.



Figuur 3 De train loss en validation loss uitgezet tegen het aantal epochs na toevoeging van twee dropout layers

3.2.2 BEPALING VAN DE ACTIVATIEFUNCTIE

De activatie-functie bepaalt de *output* die verkregen wordt door een bepaalde *input*. Deze functie is nodig om een netwerk niet-lineaire patronen te leren. De twee meest gebruikte activatiefuncties zijn *Sigmoid* en *Rectified Linear Unit* (*ReLU*). De uitkomst bij *Sigmoid* zal tussen de 0 en de 1 liggen. Bij *ReLU* zullen alle negatieve waarden een 0 krijgen. Om die reden is voor de activatiefunctie *ReLU* gekozen. Als het model een gemiddeld aantal vogels dat gaat trekken moet voorspellen, is het logisch dat negatieve waarden een 0 krijgen. Bij gebruik van *Sigmoid* zal in de voorspelling niet duidelijk zijn wanneer er gemiddeld 0 vogels zullen trekken. Die waarde zal namelijk ook worden weergegeven tussen 0 en 1. Daarom is voor dit model *ReLU* de beste activatiefunctie. Deze theorie blijkt te kloppen na het draaien van proeftrainingen.

3.2.3 BEPALING VAN DE *LOSS*-FUNCTIE

Nadat de *hidden layers* en de activatiefunctie zijn gespecificeerd is het nodig om de *loss*-functie toe te wijzen aan het model. Er zijn twee globale soorten *loss*-functies, afhankelijk van het type voorspelling dat het model moet geven. Voor binaire classificatie is een van de '*binary cross entropy loss*'-functies het beste. Voor lineaire regressie is een van de '*mean square error*'-functies het beste. Aangezien het model vogeltrek (dus een vorm van lineaire regressie, want het gaat over werkelijke aantallen) moet voorspellen, is voor dit model gekozen voor een functie van de vorm '*mean square error*'. Daarbinnen is keuze uit drie soorten *loss*-functies: de *Mean Squared Error Loss,* de *Mean Squared Logarithmic Error Loss (MSLE)* en de *Mean Absolute Error Loss (MAE).*

Bij dit model is gekozen voor de *MSLE*, omdat het model een vorm van lineaire regressie moet voorspellen, waarbij de controlewaarden erg verspreid liggen. Als in plaats daarvan de *MAE* toegepast

wordt, zal de totale *loss* veel groter zijn, omdat de spreiding zo groot is, dat van een representatief gemiddelde geen sprake kan zijn. Proeftrainingen van het model laten zien dat deze theorie klopt.

3.2.4 BEPALING VAN HET TYPE *OPTIMIZER*

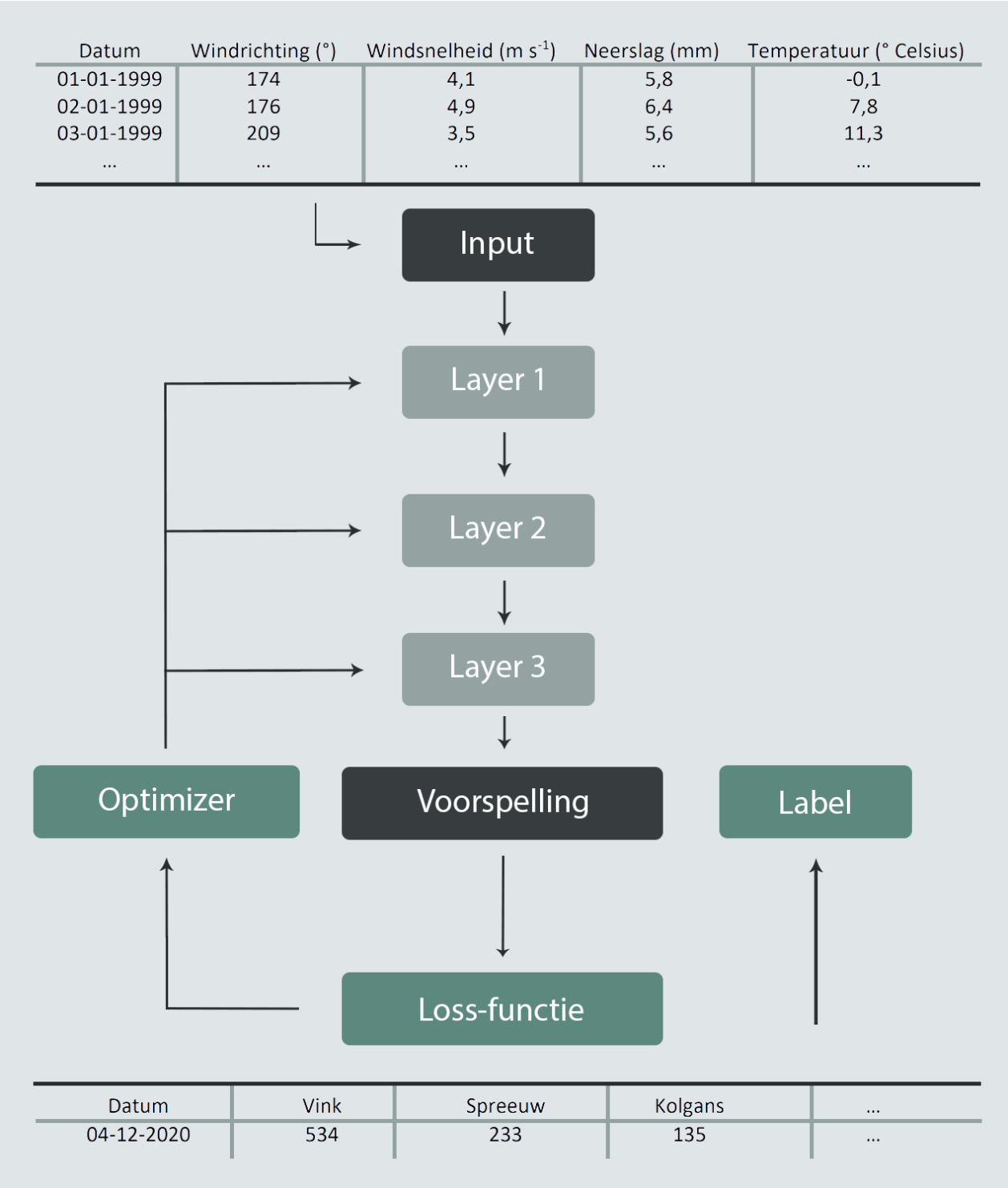
Voor het model is gebruik gemaakt van de *optimizer RMSprop*. Deze *optimizer* berekent het gemiddelde van het kwadraat van de gradiënten. Vervolgens deelt hij de gradiënt door de wortel van dat gemiddelde (Google Brain Team, 2020). Door deze methode blijven sterke verbanden staan en worden zwakke verbanden zwakker. Dit heeft als gevolg dat een model alleen de goede verbanden onthoudt, waardoor alleen de waardevolle informatie behouden blijft. Verder gaf deze methode de kleinste *loss* en daarom is deze methode gebruikt.

3.2.5 BEPALING VAN HET AANTAL *EPOCHS*

Voor het trainen van het model moet worden bepaald hoeveel *epochs* het model moet trainen. Door telkens een ander aantal te kiezen en een proeftraining te draaien kan vernomen worden wanneer het model nauwkeuriger of juist onnauwkeuriger begint te worden. In het eerste geval kan geprobeerd worden het aantal *epochs* te verhogen. In het tweede geval kan geprobeerd worden het aantal *epochs* te verlagen. Voor dit model is gebleken dat het optimum aantal *epochs* ligt rond de 150.

3.3 ACCURATIE

Tijdens het trainen van het model worden de data opgesplitst in drie groepen: *train data, validation data en test data* met respectievelijk een verhouding van 70, 20, 10. Daarvoor is gekozen, omdat met die verdeling relatief veel (70%) van de totale data getraind kan worden en er daarnaast nog genoeg over is (20%) om het model te evalueren. Uiteindelijk blijft dan nog een klein deel (10%) over om daarmee het model te testen. Om de data te scheiden is geen gebruik gemaakt van shuffling (het door elkaar mixen van rijen binnen een dataset), omdat er bij deze dataset sprake is van een chronologische volgorde. De eerste twee delen van de gesplitte data zijn gebruikt voor het trainen van het model. Het laatste deel is gebruikt om het model te testen. Door middel van de functie ‘evaluate()’, afkomstig uit de *Keras library* (Google Brain Team, 2020), is dat samen met de *test data* bewerkstelligd. Daaruit kwam een accuratie voort van 34,7% en een *loss* van -3,0.



Figuur 4 Een versimpelde werking van het model met daarbij voorbeeld input data en een voorbeeld voorspelling.

4 DISCUSSIE

Uit de resultaten blijkt dat het model een nauwkeurigheid heeft van 34,7%. Modellen die een accurate voorspelling doen op basis van radar hebben een nauwkeurigheid van rond de 90% (Bouten et al., 2005; Van Belle et al., 2007), maar er zijn ook modellen met een nauwkeurigheid van rond de 80% (Van Doren & Horton, 2018). Te verwachten is dat de nauwkeurigheid van een model met telgegevens lager ligt, maar toch is ons model met een nauwkeurigheid van 34,7% niet accuraat genoeg om een betrouwbare voorspelling te geven. Het is wel voldoende om een indicatie te geven over vogeltrek. Ons model geeft een voorspelling over de vogeltrek op soortniveau, waardoor het in combinatie met accuratere modellen mogelijk meer dan alleen een indicatie kan geven over de vogeltrek op soortniveau. Het model is echter is wel voldoende accuraat om een indicatie te geven over vogeltrek. Ons model geeft een voorspelling over de vogeltrek op soortniveau waardoor het in combinatie met accuratere modellen mogelijk meer dan alleen een indicatie kan geven over de vogeltrek op soortniveau.

Deze lage mate van nauwkeurigheid is zeer waarschijnlijk niet ontstaan door de opbouw van het model, maar door de manier waarop het model omgaat met data, want zoals uit de resultaten blijkt is elke keuze bij het model zorgvuldig genomen en gebaseerd op andere modellen. Volgens Bouten et al. (2005) is een ANN geen slechter medium om een voorspelling te doen over de vogeltrek. Van Doren en Horton (2018) hebben ook een model gemaakt op basis van een ANN, waaruit een nauwkeurige voorspelling kwam.

Tijdens het ontwerpproces van het model zijn we echter afhankelijk van de middelen die op dit moment beschikbaar zijn binnen de wereld van *machine learning*. Zo is gebruik gemaakt van de activatiefunctie *ReLU* die negatieve waarden substitueert door een nul. Doordat de waarden onder nul verloren gaan, wordt niet alle beschikbare informatie benut. Om dit probleem op te lossen moet een nieuwe activatiefunctie worden ontwikkeld, die de negatieve waarden niet elimineert zonder de informatie die het bevat te benutten. Eenzelfde soort probleem ontstaat door het gebruik van de *RMSprop optimizer*. Deze *optimizer* zorgt ervoor dat zwakke verbanden nog zwakker worden, en sterke verbanden nog sterker worden. Hierdoor worden de subtielere verbanden niet meegenomen in het model. Deze twee functies geven echter de beste resultaten, ondanks hun gebreken. Met vervolgonderzoek op dit wetenschapsgebied, kan het model mogelijk verbeterd worden. Op dit moment is het ontworpen model het beste wat mogelijk is. In Figuur 2 is te zien dat er sprake is van *overfitting* in het model. Om dit op te vangen zijn twee *drop-out layers* toegevoegd. Hoewel de *overfitting* is gereduceerd is er nog steeds sprake van *overfitting*. De waarde van de *validation loss* ligt namelijk nog steeds erg ver bij de *train loss* vandaan.

Aangezien veel geprobeerd is om het model een zo accuraat mogelijke voorspelling te laten doen, is de oorzaak van de lage accuratie waarschijnlijk de manier waarop het model de data verwerkt die zijn aangeleverd. Ten eerste zouden de data van de telposten ook gecorrigeerd moeten worden voor andere factoren dan de meteorologische factoren, namelijk voor de telomstandigheden, zoals het aantal uren, aantal tellers, de bewolking, het zicht, het tellen van zeldzamere soorten, etc. (Soldaat & Poot, 2020; Van Dijk, 2008) en ook voor de geografische ligging van de gebruikte telposten (t.o.v. de kust, hoogte boven NAP, bij topografische kenmerken, etc.). De data zoals zij nu gebruikt zijn voor de voorspelling geven ook niet het aantal van de vogels aan die daadwerkelijk getrokken hebben, maar alleen wat door de tellers waargenomen is. Om dichter bij het werkelijke aantal te komen kan onder andere

gebruik gemaakt worden van een detectiecurve, opgesteld met behulp van *distance sampling* (Newson, Evans, Noble, Greenwood, & Gaston, 2008).

Ten tweede bevatten de data van de telposten veel lacunes, waarmee ons model niet goed overweg kan. Het model zal daarom zo gemaakt moeten worden dat het deze lacunes opvult door middel van *bootstrapping* of zo gemaakt moeten worden dat het model om kan gaan met deze lacunes.

Als laatste zouden ook meteorologische data vanuit het buitenland gebruikt moeten worden om een beter voorspelling te doen, aangezien de meeste vogels die over Nederland vliegen niet vanuit Nederland vertrokken zijn. De meteorologische data van de locaties waar ze vandaan trekken kan meer zeggen over wanneer ze vertrekken en over Nederland vliegen, dan de meteorologische data in Nederland (Bradarić, Bouten, Fijn, Krijgsveld, & Shamoun-Baranes, 2020; Shamoun-Baranes et al., 2006).

Uit dit onderzoek kan geconcludeerd worden dat door de beperkingen van het model, de daarvoor gebruikte techniek en de daaruit volgende beperkte mogelijkheden voor het verwerken van de data die gebruikt zijn voor dit onderzoek, het met ons model niet mogelijk is om een accurate voorspelling te doen over de soorten vogels en over de aantallen die zullen vliegen op basis van abiotische factoren. Wel kan ons model een indicatie geven over het aantal trekkende individuen per soort, kan het in combinatie met ander modellen mogelijk een betere voorspelling doen over de vogeltrek op soortniveau en kan het functioneren als basis voor een vervolgonderzoek waarin het model wordt aangepast zodat het beter om kan gaan met de data die aangeleverd zijn.

DANKWOORD

Ten eerste willen we Gregory van den Top hartelijk bedanken voor de begeleiding in het hele proces van ons onderzoek. Daarnaast gaat onze dank ook uit naar Gerard Troost die ons de mogelijkheid heeft gegeven om gebruik te maken van de data uit de database van organisatie Trektellen. We danken ook alle tellers die in weer en wind geteld hebben, waardoor wij die data konden gebruiken. Verder willen we Willem Mulder danken voor het meedenken van het nemen van de juiste keuzes omtrent de data en het geven van feedback, Arjan Boele voor het geven van feedback en Jonathan Ooms voor het maken van een inzichtelijke kaart. Als laatste danken we iedereen die ons met liefde hebben omringd tijdens het doen van ons onderzoek.

LITERATUURLIJST

Aristoteles. (350 B.C.E.). *Τῶν περὶ τὰ ζῷα ἱστοριῶν (Geschiedenis van de dieren)*. e Typographeo academico. https://books.google.nl/books?id=2Z8NAAAAYAAJ&redir_esc=y

Bauer, S., Chapman, J. W., Reynolds, D. R., Alves, J. A., Dokter, A. M., Menz, M. H., … Shamoun-Baranes, J. (2017). From Agricultural Benefits to Aviation Safety: Realizing the Potential of Continent-Wide Radar Networks. *BioScience*, *67*(10), 912–918. https://doi.org/10.1093/biosci/bix074

Berthold, P. (2001). *Bird migration: a general survey* (2nd ed.). Oxford University Press. https://books.google.nl/books/about/Bird_Migration.html?id=F9JObKNSac8C&redir_esc=y

Bouten, W., Buurma, L., DeFusco, R., Dekker, A., Sierdsema, H., Sluiter, F., … van Loon, E. (2008). *Avian Information Systems: Developing Web-Based Bird Avoidance Models*.

Bouten, Willem, Kleyheeg-Hartman, J. C., Klop, E., Potiek, A., Shinneman, S., & Van Loon, E. (2020). *Haalbaarheidsstudie naar een voorspellen vogeltrekmodel en een stilstandvoorziening om vogesterfte te beperken in Windpark Eemshave*.

Bouten, Willem, Van Belle, J., Van Gasteren, H., Vrugt, J. A., Shamoun, J., & Buurma, L. (2005). *Predicting bird migration: data-driven versus concept- driven models*. http://raob.fsl.noaa.gov/,

Bradarić, M., Bouten, W., Fijn, R. C., Krijgsveld, K. L., & Shamoun-Baranes, J. (2020). Winds at departure shape seasonal patterns of nocturnal bird migration over the North Sea. *Journal of Avian Biology*, *51*(10), 1–11. https://doi.org/10.1111/jav.02562

Collenteur, R. A., Bakker, M., Caljé, R., Klop, S. A., & Schaars, F. (2019). Pastas: Open Source Software for the Analysis of Groundwater Time Series. *Groundwater*, *57*(6), 877–885. https://doi.org/10.1111/gwat.12925

Dijksterhuis, M. (2018). *Oostenwind of westenwind: Analyse naar de samenhang tussen windrichting en aantallen vogels telpost Hazewater | 2006 t/m 2016*. https://www.trektellen.nl/static/doc/Dijksterhuis_M_Analyse_wind_en_trek_Hazewater_2006-2016_v1.0.pdf

Dokter, A. M., Liechti, F., Stark, H., Delobbe, L., Tabary, P., & Holleman, I. (2011). Bird migration flight altitudes studied by a network of operational weather radars. *Journal of The Royal Society Interface*, *8*(54), 30–43. https://doi.org/10.1098/rsif.2010.0116

European Aviation Safety Agency. (2009). *Bird Strike Damage & Windshield Bird Strike Final Report European Aviation Safety Agency*. www.atkinsglobal.com

Gad, A. (2018). *Beginners Ask "How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?"* https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e

Google Brain Team. (2020). *TensorFlow* (2.2.0). www.tensorflow.org

Hunter, J. D. (2020). *Matplotlib* (3.2.1). matplotlib.org

Jaffré, M., Beaugrand, G., Goberville, É., Jiguet, F., Kjellén, N., Troost, G., … Luczak, C. (2013). Long-term phenological shifts in raptor migration and climate. *PLoS ONE*, *8*(11), 1–8. https://doi.org/10.1371/journal.pone.0079112

Kemp, M. U., Shamoun-Baranes, J., van Gasteren, H., Bouten, W., & van Loon, E. (2010). Can wind help explain seasonal differences in avian migration speed? *Journal of Avian Biology*, *41*(6), 672–677.

https://doi.org/10.1111/j.1600-048X.2010.05053.x

Koninklijk Nederlands Meteorologisch Instituut (KNMI). (2020). *Daggegevens van het weer in Nederland - Download*. http://projects.knmi.nl/klimatologie/daggegevens/selectie.cgi

Krijgsveld, K. L., Kleyheeg-Hartman, J. C., Klop, E., & Brenninkmeijer, A. (2020). *Stilstandsvoorziening windturbines Eemshaven Mogelijkheden en consequenties*.

McKinney, W. (2020). *pandas* (1.0.3). pandas.pydata.org

Newson, S. E., Evans, K. L., Noble, D. G., Greenwood, J. J. D., & Gaston, K. J. (2008). Use of distance sampling to improve estimates of national population sizes for common and widespread breeding birds in the UK. *Journal of Applied Ecology*, *45*(5), 1330–1338. https://doi.org/10.1111/j.1365-2664.2008.01480.x

Nilsson, C., Dokter, A. M., Verlinden, L., Shamoun-Baranes, J., Schmid, B., Desmet, P., … Liechti, F. (2019). Revealing patterns of nocturnal migration using the European weather radar network. *Ecography*, *42*, 876–886. https://doi.org/10.1111/ecog.04003

Oliphant, T. (2020). *NumPy* (1.18.5). www.numpy.org

Python Software Foundation. (2020). *Python* (3.8.3). Python Software Foundation. www.python.org

Schlüter, N. (2019). *Don't Overfit! — How to prevent Overfitting in your Deep Learning Models*. Towards Data Science. https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323

Shamoun-Baranes, J., Van Gasteren, H., & Ross-Smith, V. (2018). Sharing the aerosphere: Conflicts and potential solutions. In *Aeroecology* (pp. 465–497). Springer International Publishing. https://doi.org/10.1007/978-3-319-68576-2_18

Shamoun-Baranes, J., Van Loon, E., Alon, D., Alpert, P., Yom-Tov, Y., & Leshem, Y. (2006). Is there a connection between weather at departure sites, onset of migration and timing of soaring-bird autumn migration in Israel? *Global Ecology and Biogeography*, *15*(6), 541–552. https://doi.org/10.1111/j.1466-8238.2006.00261.x

Soldaat, L., & Poot, M. J. M. (2020). *Analyse bruinvisgegevens en evaluatie monitoring Noordzee - Kwaliteitsborging IHM 2019*. https://doi.org/10.13140/RG.2.2.11193.80481

Troost, G. (2020a). *[Trektellen.org] - Teluren*. https://trektellen.nl/stats/hours/1/-1/0

Troost, G. (2020b). *Dutch Migrationcount database*. https://trektellen.nl/

Troost, G., & Boele, A. (2020). Trektellen.org-Store, share and compare migration data. *Bird Census News*, *32*(1–2), 17–26. www.trektellen.org/count/view/1

Van Belle, J., Shamoun-Baranes, J., Van Loon, E., & Bouten, W. (2007). An operational model predicting autumn bird migration intensities for flight safety. *Journal of Applied Ecology*. https://doi.org/10.1111/j.1365-2664.2007.01322.x

Van Dijk, B. (2008). *Trektellen: Vernieuwde handleiding voor het tellen van zichtbare landtrek: Een bewerking van de LWVT-handleiding (Rob Lensink et al. 1985)*. https://docplayer.nl/17705006-Trektellen-vernieuwde-handleiding-voor-het-tellen-van-zichtbare-landtrek-een-bewerking-van-de-lwvt-handleiding-rob-lensink-et-al-1985.html

Van Doren, B. M., & Horton, K. G. (2018). A continental system for forecasting bird migration. *Science*,

361(6407), 1115–1118. https://doi.org/10.6084/m9.figshare.6962810

Van Gasteren, H., & Dekker, A. (2005). EURBASE: military bird strike frequency in Europe. *Annual Meeting International Bird Strike Committee at Athens*. https://www.researchgate.net/publication/267304979

Van Gasteren, H., Troost, G., & Boele, A. (2015, October 15). *Tientallen miljoenen trekvogels over Nederland en België*. https://www.naturetoday.com/intl/nl/nature-reports/message/?msg=22319

Vogelwacht Uden e.o. (2010). *Informatieboekje Vogelwacht Uden e.o.* . https://www.vogelwachtuden.nl/images/PDF-bestanden/secretariaat/Informatieboekje.pdf

BIJLAGEN

BIJLAGE 1: DATA MANAGEMENT PLAN

Data Management Plan

Namen: Gerwin Kuijntjes, André Vis en Lauren Zwemer
Titel van onderzoek (tijdelijk): *PWS Vogeltrek*
Begeleider: Drs. Ir. G.G. Van den Top (Van Lodenstein College Amersfoort)
Korte beschrijving van onderzoek:
Er wordt elk jaar veel informatie verzameld door trektellers. Die gegevens worden ingevoerd op trektellen.org. Deze waardevolle informatie willen wij gaan gebruiken in ons onderzoek. Ons onderzoek richt zich namelijk op het ontwikkelen van een model dat voorspellingen kan doen, op basis van het verwachte weer, over welke en hoeveel vogels er gaan vliegen. Deze voorspelling kan ten eerste inzicht geven in de grootte van trekkende vogels voor de luchtvaart, want er zijn nog regelmatig botsingen tussen vogels en vliegtuigen en daarbij speelt de grootte van de vogels een belangrijke rol. Daarnaast kan het model ook ingezet worden om botsingen met windmolens te voorkomen door de windmolens stop te zetten op dagen waarop sprake is van trek in grote getalen, of van zeldzamere vogels. Als laatste kan deze voorspelling ook gebruikt worden door andere onderzoekers om te kijken naar het moment dat de vogelsoort(en), waar onderzoek naar gedaan wordt, in Nederland aankomen of door trekken. En als een handigheidje zou het uiteindelijk ook gebruikt kunnen worden door vogeltrektellers, om te weten wanneer ze welke soorten kunnen verwachten.
Dit voorspellende model zal een zogenoemd Artificial Neural Network (ANN) zijn dat getraind zal worden met data van de organisatie Trektellen gecombineerd met data van het KNMI. Een ANN is een gecompliceerd netwerk, maar het komt op het volgende neer: er is een input van verschillende parameters, in dit geval van het weer en de datum. Door meerdere berekeningen zal daarna een bepaalde waarde uit het model komen dat de voorspelling voor een bepaalde vogelsoort zal zijn. In het begin zullen deze voorspellingen fout zijn omdat het ANN nog niks 'geleerd' heeft, maar het ANN gaat de uitkomst vergelijken met wat er eigenlijk uit zou moeten komen (hier dus wat er geteld is bij de specifieke parameters). Op basis hiervan verandert het model de berekening een heel klein beetje en kijkt of dit tot een betere voorspelling leidt. Door dit vaak genoeg te doen, dus ook met meer data, zal het ANN 'leren' hoe de input zich verhoudt tot de output en daarmee kunnen dan voorspellingen gedaan worden.
Uiteindelijk zullen deze voorspellingen zeker niet perfect zijn, maar het zal in ieder geval een beeld geven van wat er zal gaan vliegen. De natuur is uiteindelijk nooit voorspelbaar, maar aangezien het weer een belangrijke rol speelt in de vogeltrek zal het een indicatie geven, die gebruikt kan worden voor de bovengenoemde doeleinden.
Soort onderzoekdata:
Voor dit onderzoek hebben we de data nodig op welke datum welke soorten op de telposten gezien zijn. Dit halen we uit de data van organisatie Trektellen die ons aangeleverd wordt door Gerard Troost. Dit zal gaan om de volgende telposten (met de namen zoals aangegeven op Trektellen): *De Vulkaan (Den Haag), Parnassia, Kennemerduinen, Vijfhoek Diemen, Ketelbrug/Kamperhoek, De Horde (Lopik), Hazewater (Amersfoort), Brobbelbies-Noord, Brobbelbies-Zuid, De Blauwe Kamer, Strabrechtse Heide, Oelemars (Losser), Loozerheide (bij Weert) en De Hamert*. De data zullen gestandaardiseerd worden zodat het gebruikt kan worden als vergelijkingsmateriaal vormen voor het ANN. Daarnaast zullen details van de telling (op Trektellen *kopgegevens* genaamd) gebruikt worden om extremen in de data te controleren.
Hard- en software:
Het ANN zal geprogrammeerd worden in de programmeertaal *Python*, op onze persoonlijke computers en op die van het van Lodenstein College te Amersfoort.
Bestandsformaten:
De data zullen worden aangeleverd worden als .csv Omdat het ANN in *Python* geprogrammeerd zal worden zal het ANN uit .py of daaraan verwante bestanden bestaan. Ook zal het .csv bestand omgezet worden naar de vorm waarop het leesbaar is voor *Python*.

Verwachte dataomvang:
De totale dataset zal zo ongeveer rond de 50MB zijn.
Databeheer:
Alle data zullen beheerd worden door Lauren Zwemer, maar beschikbaar zijn voor alle onderzoekers (Gerwin Kuijntjes, André Vis, Lauren Zwemer). Met onze toestemming zullen de data ook te zien zijn voor onze begeleider. De data zullen alleen gebruikt worden voor dit onderzoek en zal ook niet openbaar gemaakt worden.
Opslag tijdens onderzoek:
Tijdens het onderzoek zullen de data zich bevinden op de harde schijven van alle onderzoekers. Daarnaast zal er ook een kopie van de data zich bevinden in de cloud voor het geval de data op enigerlei wijze zouden beschadigen of kwijtraken.
Documentatie:
Er zal geen bestand toegevoegd worden dat uitleg geeft over de data aangezien het alleen binnen ons onderzoek gebruikt zal worden.
Ethische aspecten:
Privacygevoelige informatie, zoals namen van de waarnemers, die zich bevindt in de data zal alleen zichtbaar zijn voor de onderzoekers binnen dit onderzoek.
Juridische aspecten:
Trektellen blijft de eigenaar van de data en zal daarom ook de volledige zeggenschap hebben over wat wel en niet toegestaan is om te doen met de data.
Tijdpad:
Het onderzoek is van start gegaan op 18 mei 2020 en zal geheel beëindigd worden op D.V. 17 februari 2021. Eind november 2020 hopen wij u een conceptversie van ons onderzoek te verstrekken.
Gedurende dit onderzoek wordt contact opgenomen met *Trektellen* en de trektellers over hoe de data worden gebruikt en gepresenteerd.

BIJLAGE 2: TELPOSTEN

Trektelpost	Totaal aantal teluren	Aantal jaren actief	Totaal aantal soorten[2]
Brobbelbies-Noord (tussen Uden en Oss)	14379	21	188
Brobbelbies-Zuid (tussen Uden en Oss)	14539	25	194
De Blauwe Kamer	5638	21	215
De Hamert	5122	13	215
De Horde (Lopik)	12030	24	258
De Vulkaan (Den Haag)	38987	47	350
Hazewater (Amersfoort)	5292	16	244
Ketelbrug/Kamperhoek	9903	18	263
Loozerheide (bij Weert)	11719	17	221
Oelemars (Losser)	11259	20	209
Parnassia Kennemerduinen	7021	68	239
Strabrechtse Heide	14147	33	215
Vijfhoek Diemen	5876	43	228

Tabel 1 Telposten met hun eigenschappen op basis waarvan de keuze gemaakt is van welke telposten we de data gebruiken in dit onderzoek.

[2] Alleen vogelsoorten, maar inclusief verzamelsoorten, hybrides en exoten

BIJLAGE 3: CODE

In deze bijlage staat alle code. Het is verdeeld in vier bestanden met elk hun eigen toelichting. Voor de meest recente versie, zie www.github.com/Hard-Biter/pwsvogel.

ANN.PY

Dit bestand bevat de kern. Hierin worden de data aangepast, het model getraind en opgeslagen.

```
import os
from pandas.core.frame import DataFrame
from six import print_
import data
import pandas
from data import Convert_data
from data import Get_data
from model import Model as modelClass
import tensorflow
import numpy
import matplotlib.pyplot as plt
from tensorflow import keras
# main functions

# ----------------------------------------------------------------------------------------
# Read CSV file for weather and parse dates into pandas dataframe.
# ----------------------------------------------------------------------------------------

root_path = os.getcwd()
path_to_csv_weather = os.path.join(root_path, "data/weather_data.csv")
path_to_csv_bird = os.path.join(
  root_path, "data/bird_migration_per_specie.csv")

weather_data_raw = data.Get_data.raw_weather_data(path_to_csv_weather)
bird_data_unfiltered = data.Get_data.bird_data(path_to_csv_bird)


weather_data_raw = Get_data.transform_weather_data(weather_data_raw)

weather_data = Get_data.filtered_weather_data(
  weather_data_raw, bird_data_unfiltered)
weather_data = weather_data.drop(['DATE'], axis=1)
bird_data = bird_data_unfiltered.drop('DATE', axis=1)


# ----------------------------------------------------------------------------------------
# Call : Build the model.
# ----------------------------------------------------------------------------------------
```

```
model = modelClass.build_model()

# ----------------------------------------------------------------------------------------
# Get number of rows and calculate the amount needed for respectively train, validation,
and test data.
# Define model, set target_data and pecify the number of epochs.
# ----------------------------------------------------------------------------------------

length_dataframe = len(weather_data) # count numbers of rows in csv file

# takes 70% of that number of rows
train_weather_dataframe = weather_data[0:int(0.7*length_dataframe)]
# takes the next 20% of that number of rows
validation_weather_dataframe = weather_data[int(
  0.7*length_dataframe):int(0.9*length_dataframe)]
# takes the last 10% of that number of rows
test_weather_dataframe = weather_data[int(
  0.9*length_dataframe):length_dataframe]

train_bird_dataframe = bird_data[0:int(0.7*length_dataframe)]
validation_bird_dataframe = bird_data[int(
  0.7*length_dataframe):int(0.9*length_dataframe)]
test_bird_dataframe = bird_data[int(0.9*length_dataframe):length_dataframe]


# target_data = Data_processing.process_bird_migration_data(bird_data) # TODO
# train_dataset = Convert_data.dataframe_to_tf_dataset(
#   train_weather_dataframe, train_bird_dataframe) # , train_dataframe)
epochs = 200

# export dataframe


Get_data.exportFile(train_bird_dataframe, "bird.csv")
Get_data.exportFile(train_weather_dataframe, "weather.csv")


# ----------------------------------------------------------------------------------------
# Train model and save the model.
# ----------------------------------------------------------------------------------------

print(len(train_bird_dataframe.columns))
print(train_bird_dataframe.columns[38])
for column in range((len(train_bird_dataframe.columns) - 1), 10, -1):
  train_bird_dataframe = train_bird_dataframe.drop(
```

```
    train_bird_dataframe.columns[column], axis=1)
for column in range((len(validation_bird_dataframe.columns) - 1), 10, -1):
  validation_bird_dataframe = validation_bird_dataframe.drop(
    validation_bird_dataframe.columns[column], axis=1)
for column in range((len(test_bird_dataframe.columns) - 1), 10, -1):
  test_bird_dataframe = test_bird_dataframe.drop(
    test_bird_dataframe.columns[column], axis=1)


print(validation_bird_dataframe.shape)

history = model.fit(x=train_weather_dataframe, y=train_bird_dataframe, epochs=epochs,
verbose=2, shuffle=True,
          validation_data=(validation_weather_dataframe, validation_bird_dataframe))

training_loss = history.history['loss']
validation_loss = history.history['val_loss']
epoch_count = range(1, len(training_loss)+1)

plt.plot(epoch_count, training_loss, 'r--')
plt.plot(epoch_count, validation_loss, 'b-')
plt.legend(['Training Error', 'Validation Loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.savefig('loss.png')
plt.show()

model.save(os.path.join(root_path, "pwsvogelmodel"))
predictions = model.predict(x=test_weather_dataframe)
predictions = pandas.DataFrame(data=predictions)
Get_data.exportFile(predictions, "test_predictions.csv")

path = os.path.join(os.getcwd(), 'pwsvogelmodel')
model = keras.models.load_model(path)
validation_loss = model.evaluate(x=test_weather_dataframe, y=test_bird_dataframe)
print(validation_loss)
```

MODEL.PY

Dit bestand bevat functies om het model te bouwen en functies die voorspellen aan de hand van een opgeslagen model.

```
import tensorflow
from tensorflow import keras
from tensorflow.keras import layers
```

```
class Model: # base class for model
  # ------------------------------------------------------------------------------
  # Build the model
  # ------------------------------------------------------------------------------
  def build_model():
    model = tensorflow.keras.Sequential([
      layers.Flatten(input_shape=(30,)),
      layers.Dense(1024, activation='relu'),
      layers.Dropout(.8),
      layers.Dense(1024, activation='relu'),
      layers.Dropout(.2),
      layers.Dense(1024, activation='relu'),
      layers.Dense(66, activation='relu')
    ])

    model.summary()
    model.compile(optimizer='RMSprop', loss=keras.losses.MeanSquaredLogarithmicError(
    ), metrics=['accuracy'])
    return model

    # ------------------------------------------------------------------------------
    # Predict with model.
    # ------------------------------------------------------------------------------
  def predict_with_model(model, input_data):
    predicted_data = model.predict(x=input_data)
    return predicted_data
```

DATA.PY

Dit bestand bevat een aantal classes die in verschillende bestanden gebruikt worden. Allereerst is er een class met functies om data op te halen en op te slaan. Vervolgens is er een class om de weer en wind data om te zetten naar sinus en cosinus. De laatste class genereert een bestand, waar elke rij het weer van vijf dagen bevat.

```
import numpy as numpy
import pandas as pandas
import tensorflow as tensorflow
import datetime
import os


class Get_data:
  # ------------------------------------------------------------------------------
  # Read CSV file for weather and parse dates into pandas dataframe.
```

```
# ------------------------------------------------------------------------------
def raw_weather_data(path_to_csv):
  raw_dataframe = pandas.read_csv(path_to_csv)
  return raw_dataframe

def transform_weather_data(raw_weather_dataframe):
  raw_weather_dataframe.info()
  date = raw_weather_dataframe['DATE']
  raw_weather_dataframe = Convert_data.time_data(
    raw_weather_dataframe)
  clean_dataframe = Convert_data.weather_data(
    raw_weather_dataframe)
  transformed_data_frame = Transform_data.dataframe(
    clean_dataframe)
  transformed_data_frame['DATE'] = date

  length_of_weather_data = len(transformed_data_frame)
  for index in range(length_of_weather_data - 4, length_of_weather_data, 1):
    transformed_data_frame = transformed_data_frame.drop([index])
  return transformed_data_frame
# ------------------------------------------------------------------------------
# Read CSV file for birds and parse dates into pandas dataframe.
# ------------------------------------------------------------------------------

def bird_data(path_to_csv):
  raw_dataframe = pandas.read_csv(path_to_csv)
  raw_dataframe.info()
  Convert_data.process_bird_migration_data(raw_dataframe) # TODO
  return raw_dataframe

def filtered_weather_data(weather_data, bird_data):
  all_bird_dates = bird_data['DATE']
  all_weather_dates = weather_data['DATE']

  deleteRows = []

  indices = pandas.Index(list(all_weather_dates))
  for date in all_weather_dates:

    if not (all_bird_dates == date).any():
      deleteRows.append(indices.get_loc(date))
    else:
      pass

  weather_data.drop(deleteRows, inplace=True)
  weather_data.reset_index(drop=True, inplace=True)
```

```
    Get_data.exportFile(weather_data, "cleanedWeatherData.csv")
    return weather_data

  def exportFile(dataFrame, fileName):
    csv = pandas.DataFrame.to_csv(dataFrame)
    outfileName = os.path.join(os.getcwd(), fileName)
    outFile = open(outfileName, "w")
    outFile.write(csv)
    outFile.close()


class Convert_data: # to sin, cosin

  # -----------------------------------------------------------------------------------
  # Process and convert time data into cosin and sin.
  # -----------------------------------------------------------------------------------
  def time_data(date_dataframe):
    seconds_in_year = 365.2425 * 24 * 60 * 60
    date_time = pandas.to_datetime(
      date_dataframe.pop('DATE'), format='%Y-%m-%d')
    timestamp_seconds_since_epoch = date_time.map(
      datetime.datetime.timestamp)
    moment_of_year_cos = numpy.sin(
      timestamp_seconds_since_epoch * (2 * numpy.pi / seconds_in_year))
    moment_of_year_sin = numpy.cos(
      timestamp_seconds_since_epoch * (2 * numpy.pi / seconds_in_year))
    date_dataframe['moment_of_year_cos'] = moment_of_year_cos
    date_dataframe['moment_of_year_sin'] = moment_of_year_sin
    return date_dataframe
  # -----------------------------------------------------------------------------------
  # Process and convert weather data into vectors.
  # -----------------------------------------------------------------------------------

  def weather_data(weather_dataframe):
    wind_rotation_degrees = weather_dataframe.pop('DDVEC')
    wind_velocity = weather_dataframe.pop('FHVEC')
    wind_rotation_radian = wind_rotation_degrees * 2 * numpy.pi / 360
    wind_x = wind_velocity * numpy.cos(wind_rotation_radian)
    wind_y = wind_velocity * numpy.sin(wind_rotation_radian)
    weather_dataframe['wind_x'] = wind_x
    weather_dataframe['wind_y'] = wind_y
    return weather_dataframe
  # -----------------------------------------------------------------------------------
  # Convert pandas dataframe to tensorflow dataset
  # -----------------------------------------------------------------------------------
```

```
  def dataframe_to_tf_dataset(pandas_dataframe_input, pandas_dataframe_target):
    if (pandas_dataframe_target.shape[0] != pandas_dataframe_input.shape[0]):
      raise ValueError(
        "shape input dataframe has to be equal to output dataframe")
    tf_dataset = tensorflow.data.Dataset.from_tensor_slices(
      (pandas_dataframe_input.values, pandas_dataframe_target.values))
    return tf_dataset
  # ----------------------------------------------------------------------------
  # Transform the dataframe.
  # ----------------------------------------------------------------------------


class Transform_data:

  #! removes last 4 rows
  def dataframe(dataframe_with_weather_data_for_1_day_per_row):
    dataframe = dataframe_with_weather_data_for_1_day_per_row
    series_with_all_columns = dataframe.columns
    for days_back in range(-1, -5, -1):
      for a in range(0, 6):
        column = dataframe.iloc[:, a]
        column_name = series_with_all_columns[a]
        temp_column_name = f'{column_name}_{days_back}'
        temp_column = Transform_data.column(
          column, days_back)
        dataframe[temp_column_name] = temp_column
        dataframe[f'{temp_column_name}'] = temp_column
    return dataframe

  def column(old_column, days_back):
    new_column = old_column
    for x in range(0, -1*days_back, 1):
      new_column = new_column.drop(index=x)
    new_column.reset_index(drop=True, inplace=True)
    return new_column
```

KNMI.PY

Dit bestand laadt het eerder gegenereerde model en gebruikt het om de vogeltrek te voorspellen aan de hand van het weer van de laatste vijf dagen.

```
import os
from datetime import datetime, timedelta
from pandas import DataFrame
from tensorflow.keras import models
```

```
from os import path, getcwd
from data import Get_data
from pastas.read.knmi import KnmiStation

date_of_today = datetime.now().date()
date_of_5_days_back = date_of_today - timedelta(5)
knmi = KnmiStation.download(start=date_of_5_days_back, end=date_of_today)
knmi_data = knmi.data
arrayWithVariables = ['FHVEC', 'DDVEC', 'TG', 'RH']

def getNeededVariables(arrayWithVariables):
  path_to_model = path.join(getcwd(), 'pwsvogelmodel')
  neededKNMIdata = DataFrame()
  for variable in arrayWithVariables:
    neededKNMIdata[variable] = knmi_data.pop(variable)

  neededKNMIdata = neededKNMIdata.reset_index()
  neededKNMIdata.rename(
    columns={neededKNMIdata.columns[0]: "DATE"}, inplace=True)

  neededKNMIdata = Get_data.transform_weather_data(
    neededKNMIdata).drop('DATE', axis=1)
  model = models.load_model(path_to_model)
  print(model.summary())
  predictions = model.predict(x=neededKNMIdata)
  Get_data.exportFile(DataFrame(predictions), 'predictions.csv')
  return predictions

prediction_based_on_knmi_date = getNeededVariables(arrayWithVariables)
print(prediction_based_on_knmi_date)
```