Het beheersen van stedelijke complexiteit met Graph Neural Networks

Foto: Andrii Turtsevych

In een verkeersnetwerk is alles met alles verbonden. Dat zorgt ervoor dat een ingreep *hier* tot onverwachte gevolgen *daar* kan leiden. Standaard verkeersmodellen weten niet goed raad met deze complexe verbanden, maar de relatief nieuwe loot van *Graph Neural Networks* wél. Hoe slagen deze AI-modellen erin om informatie tussen verschillende delen van een netwerk uit te wisselen?

Voor we de techniek van *Graph Neural Networks*, GNN's, induiken, is het goed stil te staan bij de noodzaak van GNN's. Want waarom zouden andere, toch ook superslimme AI-modellen minder of niet voldoen voor het rekenen aan verkeersnetwerken?

Het antwoord ligt in de geometrie. Standaard AI-modellen, met name *Convolutional Neural Networks*, zijn ontworpen voor ***grids***. Denk aan een digitale afbeelding: een perfect schaakbord van pixels, netjes gerangschikt in rijen en kolommen, waarbij elke pixel exact hetzelfde aantal buren heeft. Dit wordt in de wiskunde een *Euclidische ruimte* genoemd.

Een stad is echter geen schaakbord, maar een onregelmatig web en inherent inconsistent. Zo sluit het ene kruispunt op vijf wegvakken aan en het andere op maar twee. Wanneer je zo'n *niet-Euclidische* realiteit probeert te forceren in een rigide grid om standaard AI toe te passen, verlies je de meest waardevolle informatie: de topologie.

GNN's vereisen geen grid, respecteren de unieke, onregelmatige vorm van het netwerk en verwerken data zoals die in werkelijkheid bestaat. Dát maakt ze zo nuttig in het verkeersdomein.

HOE WERKT EEN GNN?

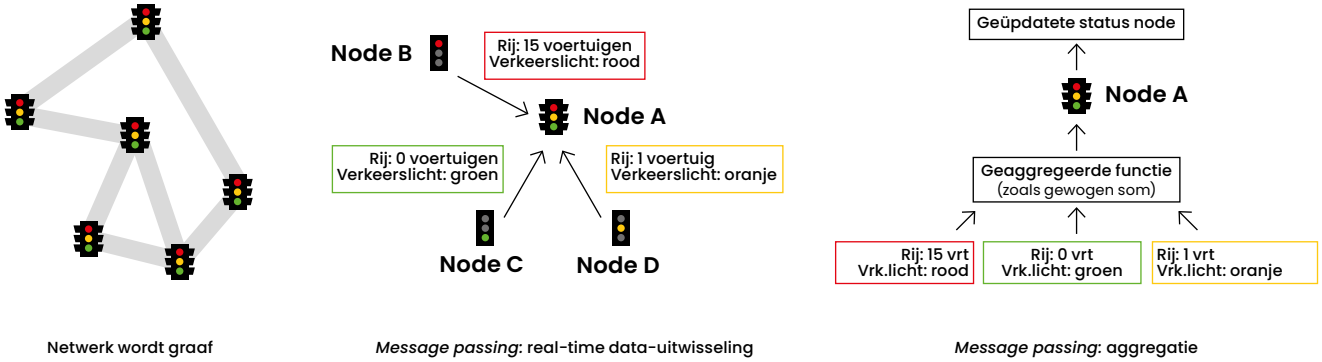
Hoe verwerkt een GNN de data? Het unieke zit 'm in het begrip ***graph*** – in het Nederlands: ***graaf*** – in de naam GNN. Simpel gezegd is een graaf een collectie van punten waartussen verbindingen lopen.

Vergelijk het met een metronetwerkkaart. Het doel van gewone stadskaarten is om punten, zoals metrostations, op de goede coördinaten weer te geven. Maar bij de metronetwerkkaart draait het om de *relaties* tussen metrostations. Of die stations geografisch precies op hun plek staan, is ondergeschikt aan de topologie: de kaart moet vooral duidelijk maken hoe station A verbonden is met station B.

Nodes en edges

Uit bovenstaande definitie blijkt al dat een graaf leunt op twee elementen: de punten, die we ***nodes*** noemen, en de verbindingen, ***edges***.

Waar die *nodes* en *edges* precies voor staan, hangt af van het probleem dat je wil oplossen. Bij een analyse van verkeerscongestie zijn de



Figuur 1:
*De interne werking van een Graph Neural Network voor verkeersvoorspelling. **Links** zien we hoe een fysiek wegennetwerk wordt getransformeerd naar een graaf, waarbij kruispunten 'nodes' worden en wegen 'edges'. De figuur in het **midden** demonstreert 'message passing': naburige kruispunten (de nodes B, C en D) delen real-time data, zoals wachtrijlengte en verkeerslichtstatus, met een centrale node (A). De figuur **rechts** laat zien hoe de binnenkomende berichten via een 'aggregation function' worden verwerkt om de toestand (state) van node A te actualiseren.*

nodes misschien de kruispunten en de *edges* de wegvakken tussen die kruispunten. Maar bij een analyse van de verkeersbewegingen door een stad zijn de *nodes* waarschijnlijk herkomsten en bestemmingen (scholen, bedrijventerreinen enzovoort) en de *edges* de verplaatsingen van personen tussen die locaties.

Gewogen verbindingen en de nabijheidsmatrix

Nu zijn niet alle verbindingen even sterk of belangrijk. In een graaf maken we dat onderscheid door ***gewichten*** aan de *edges* toe te kennen. Wiskundig organiseren we deze in een *adjacency matrix* of ***nabijheidsmatrix***. De (positieve) waarden in deze tabel geven de sterkte van de verbindingen tussen twee *nodes* aan; een nul betekent dat er geen directe verbinding bestaat.

Hoe we die gewichten bepalen, hangt af van de context. In een wegennetwerk zijn gewichten vaak omgekeerd evenredig aan de afstand. Oftewel: als de afstand tussen twee kruispunten klein is, is de verbinding 'sterk'. De congestie op het ene kruispunt heeft immers al snel effect op het andere. Maar in een logistiek netwerk zijn economische relaties vaak belangrijker. Twee hoofdsteden kunnen fysiek ver uit elkaar liggen, maar toch een hoge verbindingssterkte hebben vanwege een intens handelsvolume.

Overigens kunnen in hetzelfde graafmodel meerdere nabijheidsmatrices gebruikt worden. Bijvoorbeeld één voor fysieke afstand, één voor snelheidslimieten en één voor vrachtwagenintensiteiten. Hiermee ontstaat een rijk, meerlagig beeld van het verkeers- en vervoerssysteem.

Het mechanisme: message passing

We snappen nu hoe de graaf van een GNN is opgebouwd. Maar op welke wijze 'leert' het model van deze structuur?

Hier komen we bij het hart van een GNN: het proces van ***message passing***. Je kunt dit zien als een grootschalig, gelijktijdig gesprek dat plaatsvindt over het volledige netwerk. Waar in een traditioneel model een datapunt geïsoleerd wordt behandeld, vraagt in een GNN elke *node* actief informatie op bij z'n buren om de eigen situatie beter te begrijpen.

Neem als voorbeeld kruispunten die de verkeerssituatie voor de komende tien minuten proberen te voorspellen – zie ook bijgaande figuur. Die 'luisteren' naar berichten van aangrenzende kruispunten door informatie op te halen over bijvoorbeeld congestie of incidenten stroomafwaarts. De *node* verzamelt deze signalen en aggregeert ze tot een samenvatting, gewogen volgens de nabijheidsmatrix (= meer prioriteit voor belangrijke buren). Vervolgens combineert de *node* deze externe informatie met zijn eigen interne data, zoals actuele intensiteit of groenfase, om zijn toestand te actualiseren.

Door dit proces meerdere keren te herhalen kan een *node* uiteindelijk informatie incorporeren van locaties kilometers verderop. Dankzij *lokale* interacties worden zo problemen in het *netwerk* als geheel opgelost.

Waar zit het neural-deel?

In dit *message passing*-proces zit ook het ***neural***-deel uit de naam. Neurale netwerken fungeren in dit proces namelijk als de 'vertalers'.

Voordat een *node* een bericht uitzendt, zal het eerst proberen de 'ruwe' signalen, zoals een eenvoudige melding van een ongeval, om te zetten naar een informatierijkere boodschap. Hierdoor ontvangen andere *nodes* niet slechts kale statistieken van hun buur, maar context en patronen die zijn gecodeerd in een vorm die het model diepgaand kan interpreteren en verwerken.

TOEPASSINGEN

Wat zijn mogelijke toepassingen van GNN in verkeer en vervoer? We noemen een paar voorbeelden uit de praktijk.

Reistijdschatting

Traditionele modellen behandelden wegen vaak als onafhankelijke entiteiten, zonder rekening te houden met hoe congestie zich verspreidt tussen alternatieve routes. Het GNN-model ConSTGAT,[1] toegepast binnen Baidu Maps, kijkt daarentegen naar parallelle routes die overloopverkeer opvangen, toeritten die extra verkeersvolume voeden en naar bottlenecks die zich nog moeten manifesteren.

Door een *Graph Attention Network* te combineren met een temporele component, worden zowel de ruimtelijke interacties tussen wegvakken als de tijdsdynamiek gemodelleerd. Het resultaat is niet slechts een voorspelling, maar systeembewustzijn.

Netwerkbrede snelheidsvoorspelling

Op een enkele corridor kunnen we de snelheid eenvoudig voorspellen met een tijdreeksmodel. Maar op netwerkschaal met honderden sensoren voldoet zo'n model minder: een snelheidsdaling kan zich ongemerkt verspreiden naar toevoerwegen kilometers verderop.

Het model DCRNN[2] benadert netwerkbrede snelheidsvoorspelling als een diffusieproces. Het netwerk wordt gecodeerd als een bidirectionele *random walk*, waarmee ruimtelijke en temporele afhankelijk-

1 Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation.
2 Diffusion Convolutional Recurrent Neural Network.

heden gelijktijdig worden geleerd. Het model wacht niet tot congestie arriveert, maar anticipeert op het pad dat deze waarschijnlijk zal volgen.

Openbaar vervoer

Het voorspellen van openbaar vervoer is een uitdaging op zich. Een bus bijvoorbeeld beweegt zich binnen het algemene verkeer en eventuele vertragingen daar beïnvloeden weer de reizigersaantallen bij haltes verderop. Deze *crowding* kan zich over de hele lijn verplaatsen.

Het model TMS-GNN[3] integreert real-time verkeerscondities direct in de graaf van haltes en routes. In een *multi-task framework* voorspelt het zowel verkeersstaten als reizigersstromen en modelleert het hoe vertragingen zich verspreiden. Voor vervoersautoriteiten betekent dit een verschuiving van statische dienstregelingen naar dynamisch inzicht: waar zal overbezetting optreden, hoe moet de inzet worden aangepast, welke informatie is cruciaal voor reizigers? Enzovoort.

Strategische planning

Bij strategische planning gaat het niet alleen om de vraag of bijvoorbeeld een nieuw deelfietsstation gebruikt zal worden, maar ook hoe zo'n station de systeemdynamiek beïnvloedt.

Een GNN ontwikkeld voor toepassingen in Toronto en Vancouver modelleert station-tot-station-verplaatsingen als een graaf en simuleert *what-if*-scenario's. Door latente afhankelijkheden te leren kan het model niet alleen de vraag op de nieuwe locatie voorspellen, nog voordat de infrastructuur er ligt, maar ook de verschuivingen op verbonden stations. Planners krijgen hiermee inzicht in tweede-orde-effecten.

Energie en smart charging

Bij dynamisch smart charging is vermogensverdeling een coördinatievraagstuk tussen voertuigen, laadstations en tijd. Het aan de TU Delft ontwikkelde GNN-DT[4] modelleert elk voertuig en elk laadpunt als *nodes* in een deelbare graaf. *Edges* representeren geschiktheid en temporele afhankelijkheden.

Via *message passing* leert het model de structuur van deze interacties, anticipeert het op conflicten, past het zich real-time aan en optimaliseert het collectief.

AANDACHTSPUNTEN

Er zijn dus al genoeg (relevante!) toepassingen voor GNN in verkeer en vervoer. Maar het succesvol vertalen van GNN-onderzoek naar operationele systemen vraagt om meer dan een goed algoritme. Het vereist een doordachte *ontwerpstrategie*. Let daarbij op de volgende punten.

1. Definieer de juiste graaf

Het definiëren van de graaf is een fundamentele stap. Kijk verder dan louter fysieke nabijheid en focus op de relaties die het systeem daadwerkelijk sturen.

Ook is het essentieel om te bepalen of het vraagstuk een ***statische structuur*** vereist (bijvoorbeeld vaste infrastructuur), of een ***dynamische structuur*** die zich real-time aanpast aan incidenten en veranderende omstandigheden.

De graaf is het primaire instrument waarmee domeinkennis wordt gecodeerd. Wanneer één perspectief onvoldoende is, combineer dan meerdere invalshoeken. Een voorbeeld hiervan is de DG4b-methode,[5] ontwikkeld aan de TU Delft. Deze methode combineert twee afzonderlijke grafen: een die de statische weginfrastructuur representeert en een andere die dynamische fietsroutes modelleert. Door deze te integreren wordt de nauwkeurigheid van reistijdschattingen voor fietsverkeer aanzienlijk verbeterd.

2. Houd de architectuur gericht en beheersbaar

Bij het ontwerpen van het neurale netwerk geldt vaak: eenvoud is kracht. Gedreven door het *message passing*-mechanisme vergroot elke GNN-laag het 'receptive field' van een *node* met precies één stap. Zo vangt een tweelaags netwerk effectief de context van *neighbours-of-neighbours*.

Té veel lagen leiden echter tot het fenomeen *over-smoothing*. Hierbij vervagen unieke lokale kenmerken, omdat elke *node* uiteindelijk informatie uit vrijwel de hele graaf aggregeert. De representaties van *nodes* worden dan onderling nauwelijks onderscheidbaar.

Voor de meeste verkeers- en vervoersnetwerken adviseren wij een diepte van ***twee tot vier lagen***.

3. Plan voor schaal en snelheid

De stap van een testdataset naar een stedelijk netwerk met duizenden wegvakken verandert het computationele speelveld fundamenteel.

Om efficiënt met deze schaal om te gaan, adviseren wij technieken als ***graph sampling*** en ***training op subgrafen***. Hiermee leert het model representatieve patronen zonder bij elke iteratie het volledige netwerk te hoeven verwerken.

Voor real-time toepassingen – waar voorspellingen binnen seconden beschikbaar moeten zijn – kan ***knowledge distillation*** uitkomst bieden. Hierbij train je eerst een complex en zeer nauwkeurig *teacher model*. De inzichten die dat oplevert 'comprimeer' je vervolgens naar een lichter *student model* dat eenvoudiger inzetbaar is.

CONCLUSIE

De overstap naar GNN's is meer dan een algoritmische upgrade. We bewegen van een benadering waarin transport wordt gezien als een verzameling geïsoleerde datapunten, naar een ***systeemvisie*** waarin verbindingen centraal staan.

Dat wil niet zeggen dat GNN de nieuwe 'standaard' moet worden in verkeer en vervoer. Voor professionals geldt een hiërarchie van noodzaak. Is het probleem lokaal en zijn interacties zwak? Dan blijven klassieke methoden robuust en efficiënt. Maar domineren netwerkeffecten, waarbij de toestand van een *node* onlosmakelijk verbonden is met zijn buren, dan ligt de inzet van GNN's zeker voor de hand.

De potentiële opbrengst is in ieder geval groot. Door het 'levende web' van onze steden adequaat te modelleren, openen GNN's de deur naar transportsystemen die niet alleen reactief zijn, maar daadwerkelijk intelligent en voorspellend functioneren. ●

De auteurs
Ting Gao is onderzoeker van het DAIMOND Lab van TU Delft. Zij wordt begeleid door dr. ir. Winnie Daamen, dr. Elvin Isufi en prof. dr. ir. Serge Hoogendoorn.
Daniel Arcadio Chaves Acuña is onderzoeker aan de KTH Royal Institute of Technology. Zijn begeleiders zijn dr. ir. Wilco Burghout, prof. dr. Erik Jenelius en dr. Matej Čebecauer.

3 Traffic-Aware Multistep Graph Neural Network for Bus Passenger Flow Prediction.
4 Graph Neural Network Enhanced Decision Transformer.
5 Bicycle Travel Time Estimation via Dual Graph-Based Neural Networks.