KU LEUVEN

TECHNOLOGIECAMPUS DE NAYER SINT-KATELIJNE-WAVER

Workshop Deep Learning

10/05/2017 – Evoluon Eindhoven

Ir. Steven Puttemans
EAVISE Research Group, KU Leuven, Campus De Nayer,
Sint-Katelijne-Waver, Belgium

Wie ben ik?

Steven Puttemans – PhD onderzoeksmedewerker

- **Opleiding**
  - Industrieel ingenieur Elektronica-ICT – *KHBO Brugge/Oostende*
  - Master na Master in Artificial Intelligence – *KU Leuven*
- **Actief binnen OpenCV community**
  - Open-source beeldverwerkingspakket
  - Actief als developer/moderator

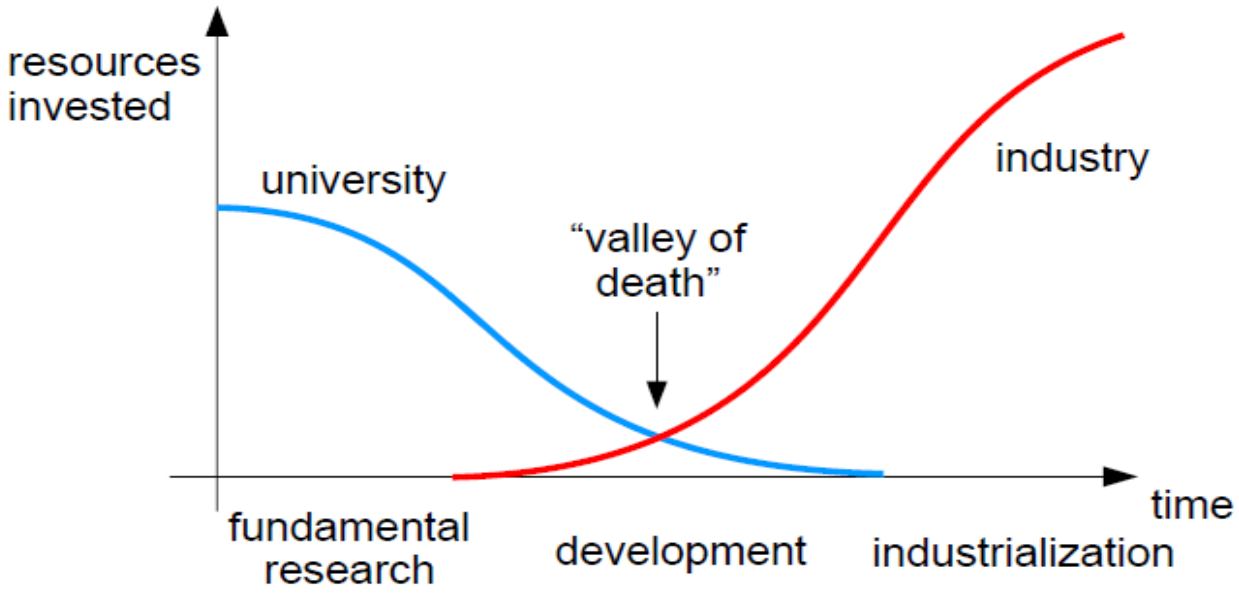
OpenCV GitHub

- **+- 6 jaar onderzoekservaring**
  - 2 jaar op IWT-TETRA project TOBCAT: industriële toepassingen van object categorisatie
  - 5 jaar als doctoraatstudent: optimale object categorisatie onder applicatie specifieke voorwaarden

KU LEUVEN

EAVISE Onderzoeksgroep?

Valley of death



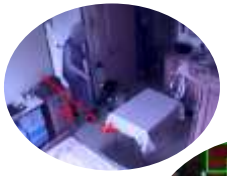
EAVISE: **E**mbedded and **A**rtificially Intelligent **Vis**ion **E**ngineering

- Vertalen van **state-of-the-art** algoritmes naar oplossingen voor **industrie** specifieke problemen, dit zowel vanuit de **beeldverwerking** als de **artificiële intelligentie**.

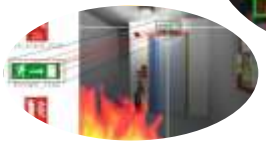
- **Optimalisatie** algoritmes om te voldoen aan industriële eisen:
  - Real-time performantie: **1-2 FPS** is vaak onvoldoende
  - Robuuster en accurater: **95%** accuraatheid onvoldoende

- Implementeren van geavanceerde algoritmes en applicaties op **embedded systemen** zoals FPGA, DSP, GPU, multicore CPU, clusters, …

Onderzoekstopics



Object detectie



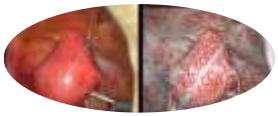
Persoons-detectie en tracking

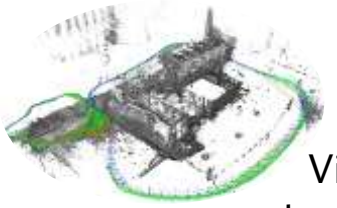
OpenCV

OpenCL

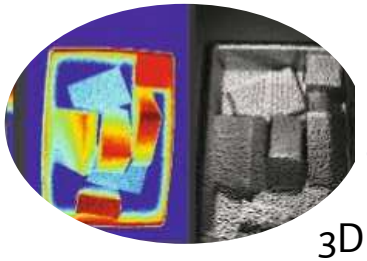
pcl

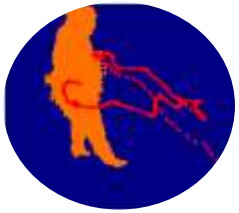
Real-time (embedded) implementatie





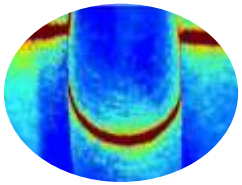
Visuele localisatie

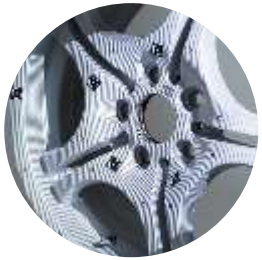




3D systemen

Visie gebaseerde robotsturing (bvb. drones)





KU LEUVEN

Onderzoekers @ EAVISE

Prof. Toon Goedemé
research leader
Computer Vision

Prof. Joost Vennekens
research leader
AI & KR

Stijn De Beugher
Eyetracking

Steven Puttemans
2D Objectdetectie

Inge Coudron
Visuele navigatie

Kristof Van Beeck
Blindehoekcamera

Wiebe Van Ranst
GPU & Mobile

Bram Aerts
Kennisrepresentatie voor cinematografie

Wim Abbeloos
3D Bin Picking

Dries Hulens
Robotic UAV

Timothy Callemein
AR & Aut. Cinematography

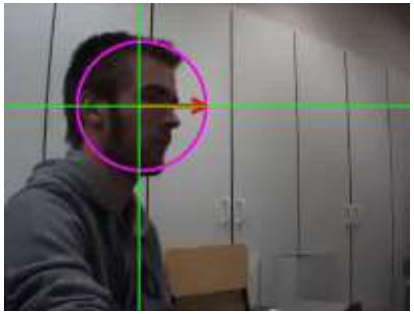
Maarten Vandersteegen
IR persoonsdetectie

Kristof Van Engeland
AI Surveillance

Shana Van Dessel
Timetabling

KU LEUVEN

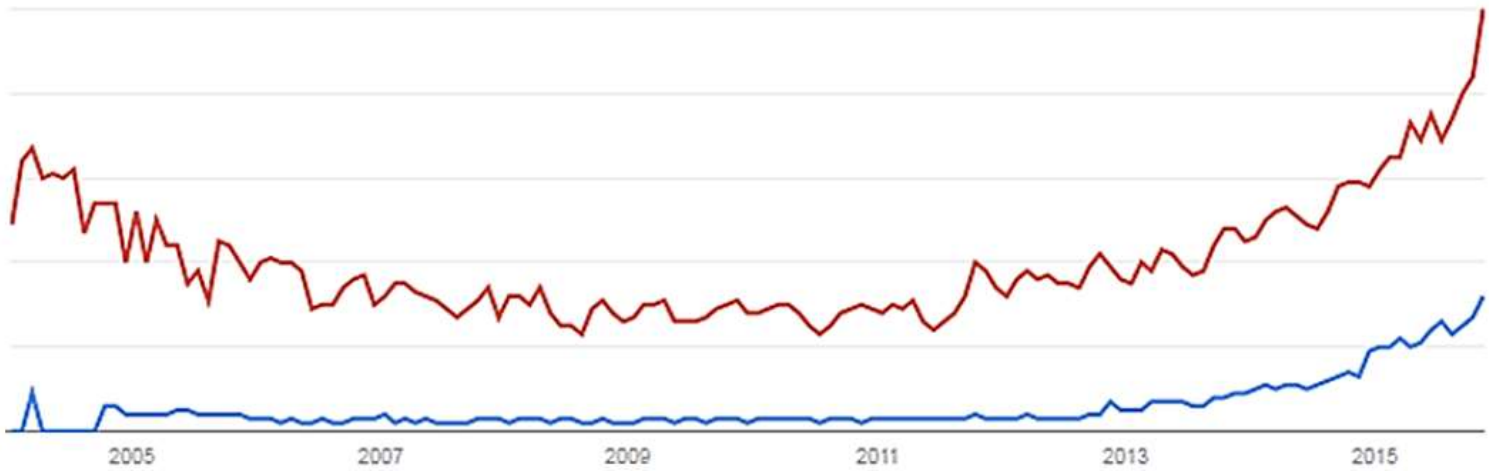
Voorbeeldprojecten



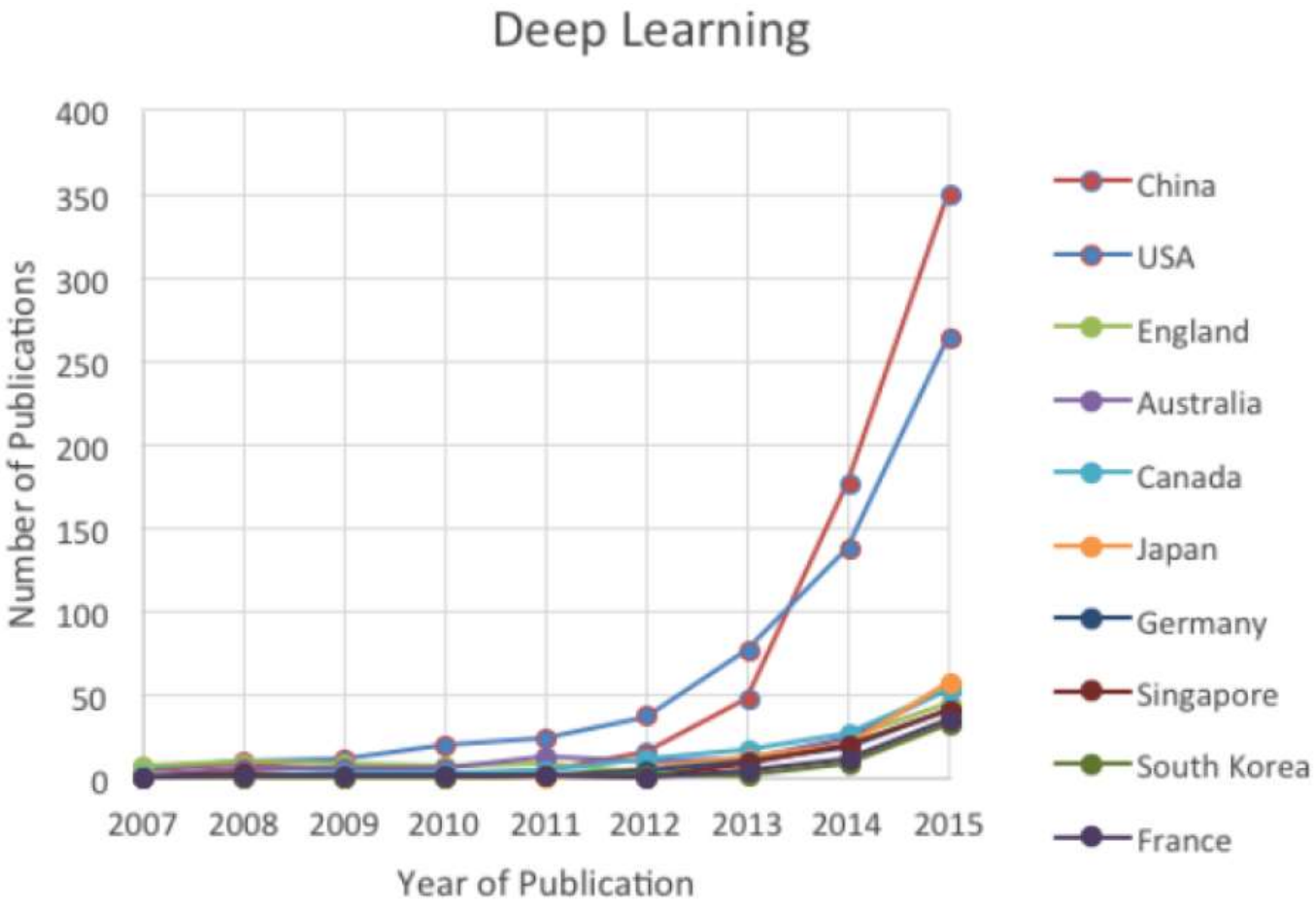
DEEL 1:
Wat is deep learning?

Presentation based on deep learning presentation given by Alison B Lowndes of NVIDIA
@ Campus De Nayer during International Days 2017

Situering deep learning

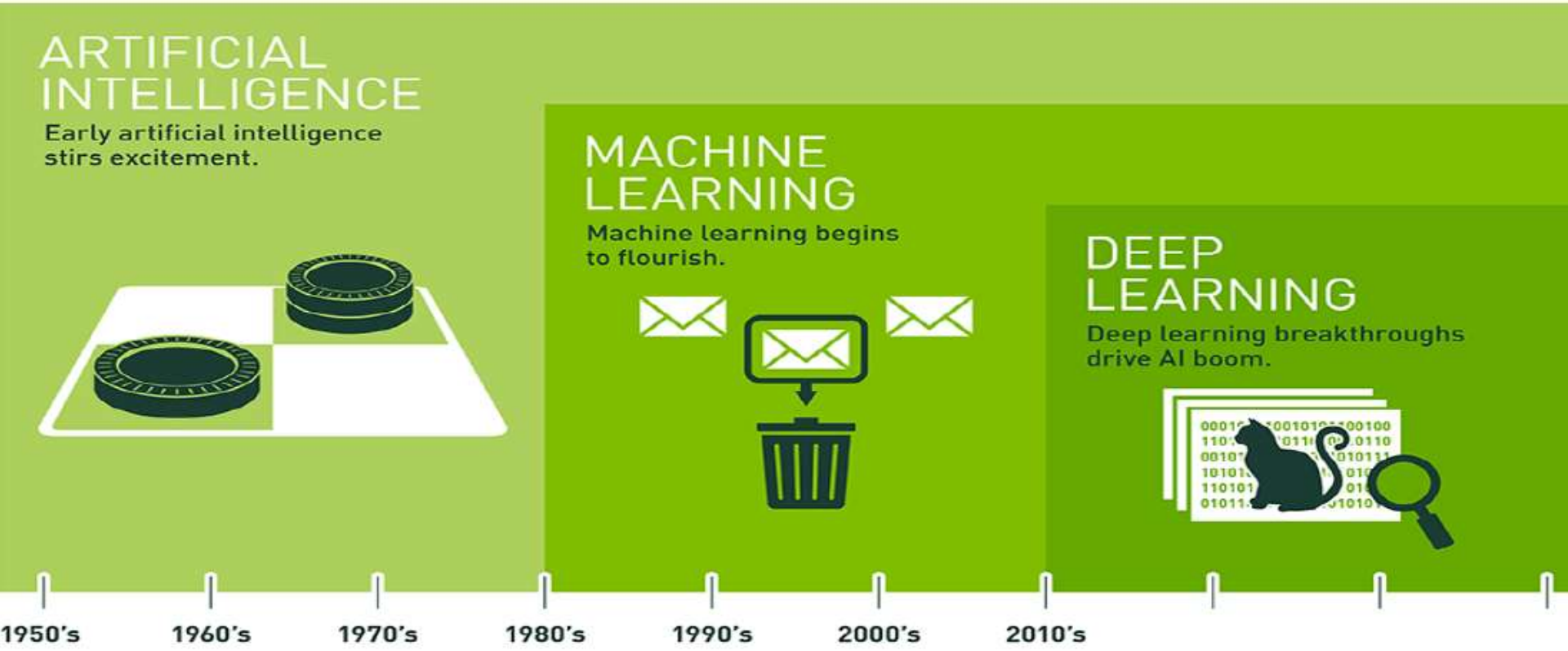


Situering deep learning



Situering deep learning

"We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College"

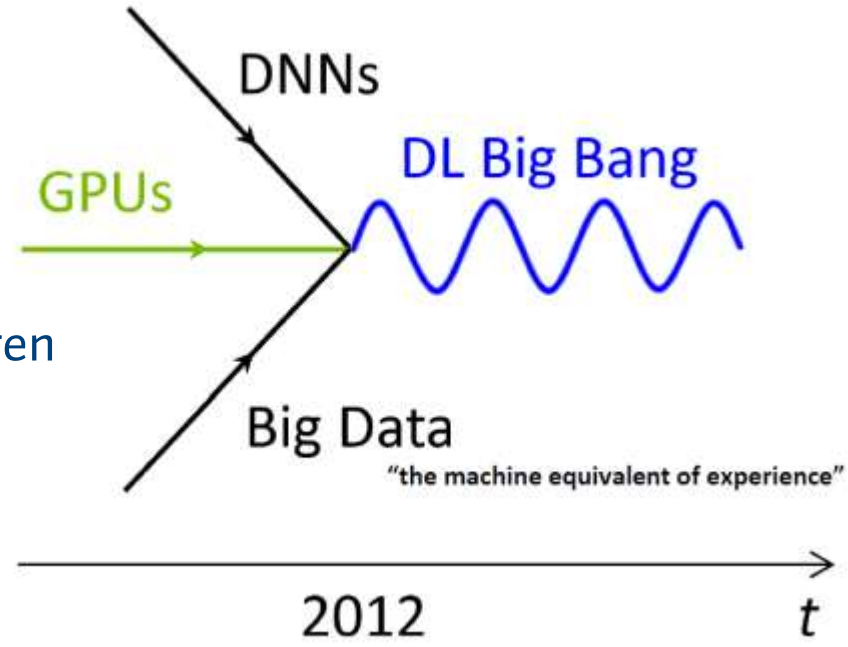


Situering deep learning

Waarom heeft deep learning nu pas een doorbraak?

- Yann Le Cun: A theoretical framework for back-propagation 1998
  - Theoretisch <-> praktische haalbaarheid
  - Software nog niet matuur genoeg

- In **2012** eindelijk een doorbraak
  - Voldoende publieke datasets
  - Voldoende toegankelijke architecturen
  - Nodige hardware word betaalbaar



Shallow/classical learning vs deep learning

- Deep learning is slechts een onderdeel van machine learning.
- Omdat deep learning momenteel een echte `hype` is, merken we een drastische wijziging in educatieve pakketten.
  - Klassieke technieken SVM, NaiveBayes, Gaussian Mixture Models, Random Forests, … krijgen steeds minder aandacht.
  - Om mee te zijn met de trend focust men steeds meer op neurale netwerken (ANN, CNN, DNN, RNN, AutoEncoders, GAN, …).

- We mogen echter niet vergeten dat klassieke machine learning nog steeds zijn plaats kent, en ook belangrijk is tot het bekomen van krachtige en robuuste beeldverwerkingsalgoritmes.

- We moeten deep learning als bouwsteen beschouwen.

Deep learning, hoe werkt dit?

- Deep learning = automatisch neurale netwerken parametriseren.

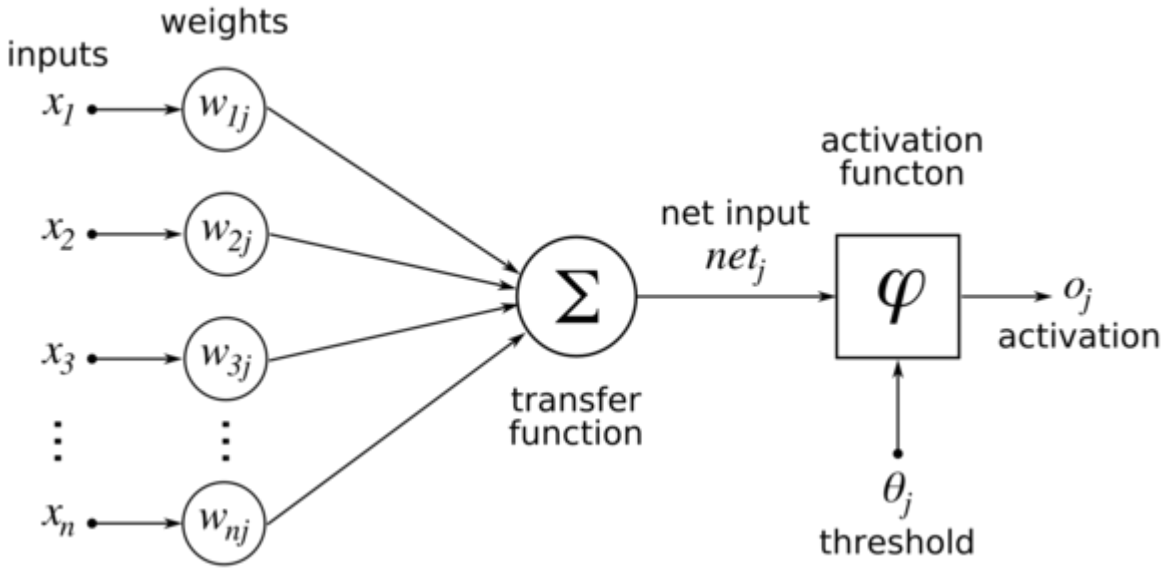
- Neurale netwerken bestaan al heel lang, en werden eigenlijk ook in de klassieke beeldverwerken reeds gebruikt:
  - SLP = single layer perceptron
  - MLP = multilayer perceptron

- Pas wanneer we heel wat complexe lagen van perceptronen gaan combineren met tussenliggende operaties (*convolutie*, *max-pooling*, …) kunnen we spreken over deep learning.

Deep learning, hoe werkt dit?

SLP = Single Layer Perceptron

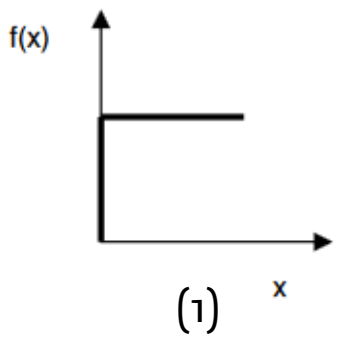
- Simpelste model van een neuron gebruikt in beeldverwerking
- Doel: herkennen van patronen in lineair scheidbare klassen
- Neuron + aanpasbare gewichten per input + bias



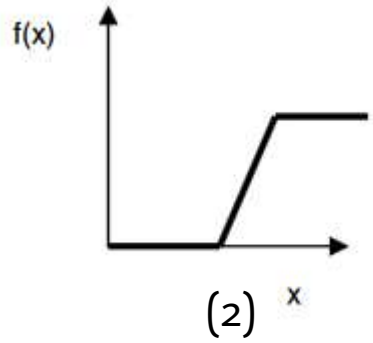
Deep learning, hoe werkt dit?

SLP = Single Layer Perceptron

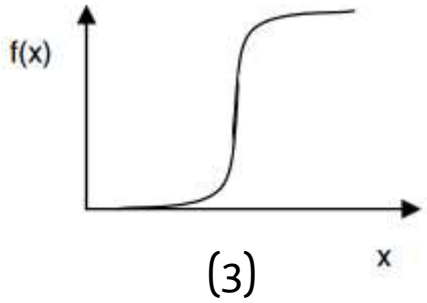
- Verschillende activatiefuncties mogelijk
  1. Threshold f(x) = 1 **if** x >= 0
     f(x) = 0 **if** X < 0
  2. Piecewise linear f(x) = 1 **if** x >= 0,5
     f(x) = x + 0,5 **if** -0,5 <= x <= 0,5
     f(x) = 0 **if** x <= -0,5
  3. Sigmoid f(x) = 1 / (1 + e^(-x))



(1)



(2)



(3)

Deep learning, hoe werkt dit?

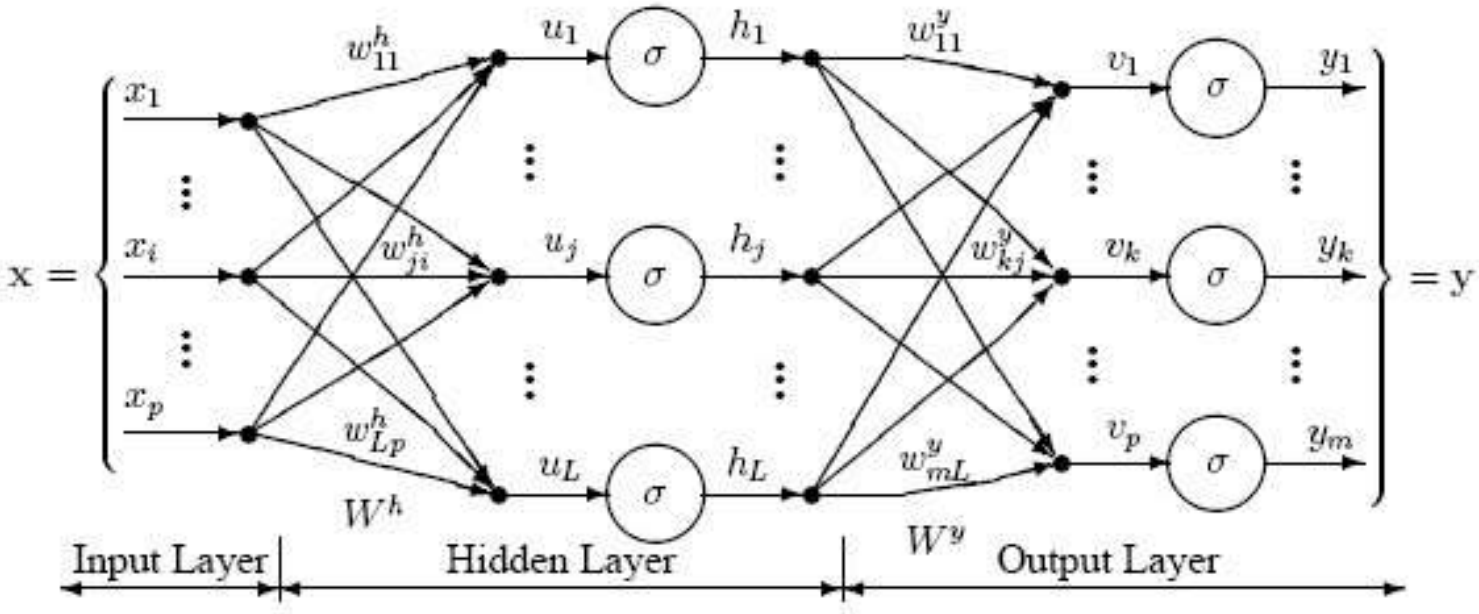
MLP= Multi Layer Perceptron

- Een netwerk van aan elkaar geschakelde perceptronen
- Feedforward netwerk (geen feedback tijdens inferentie)
- Backpropagation (supervised learning technique) voor het trainen van het netwerk
  - Bepalen van de gewichten van de onderlinge nodes
  - Bepalen van de activatiefunctie gekoppeld aan elke node
- Combinatie van input & output layers
  - Inputlayer = 1 node per inputwaarde / feature
  - Outputlayer = 1 node per klasse / label
- Daartussen verschillende hidden layers
- Fully connected netwerk
  - Elke node in laag T gelinkt aan elke node van laag T+1

Deep learning, hoe werkt dit?

MLP= Multi Layer Perceptron

- Kunnen vergeleken met SLP niet lineair scheidbare problemen oplossen door hun complexere structuur



Deep learning, hoe werkt dit?

Toepassingen van SLP & MLP

- Optical Character Recognition (OCR)
- Speech recognition
- Image recognition

MLP zijn tot +-2012 de enige vorm van Deep Learning netwerken die actief gebruikt werden voor heel wat toepassingen.

- Met een beperkt aantal hidden layers toch heel accurate resultaten.
- Je kan aan deze neurale netwerken nog steeds je eigen hand-crafted features meegeven, en het netwerk enkel als classifier gebruiken.

Deep learning, hoe werkt dit?

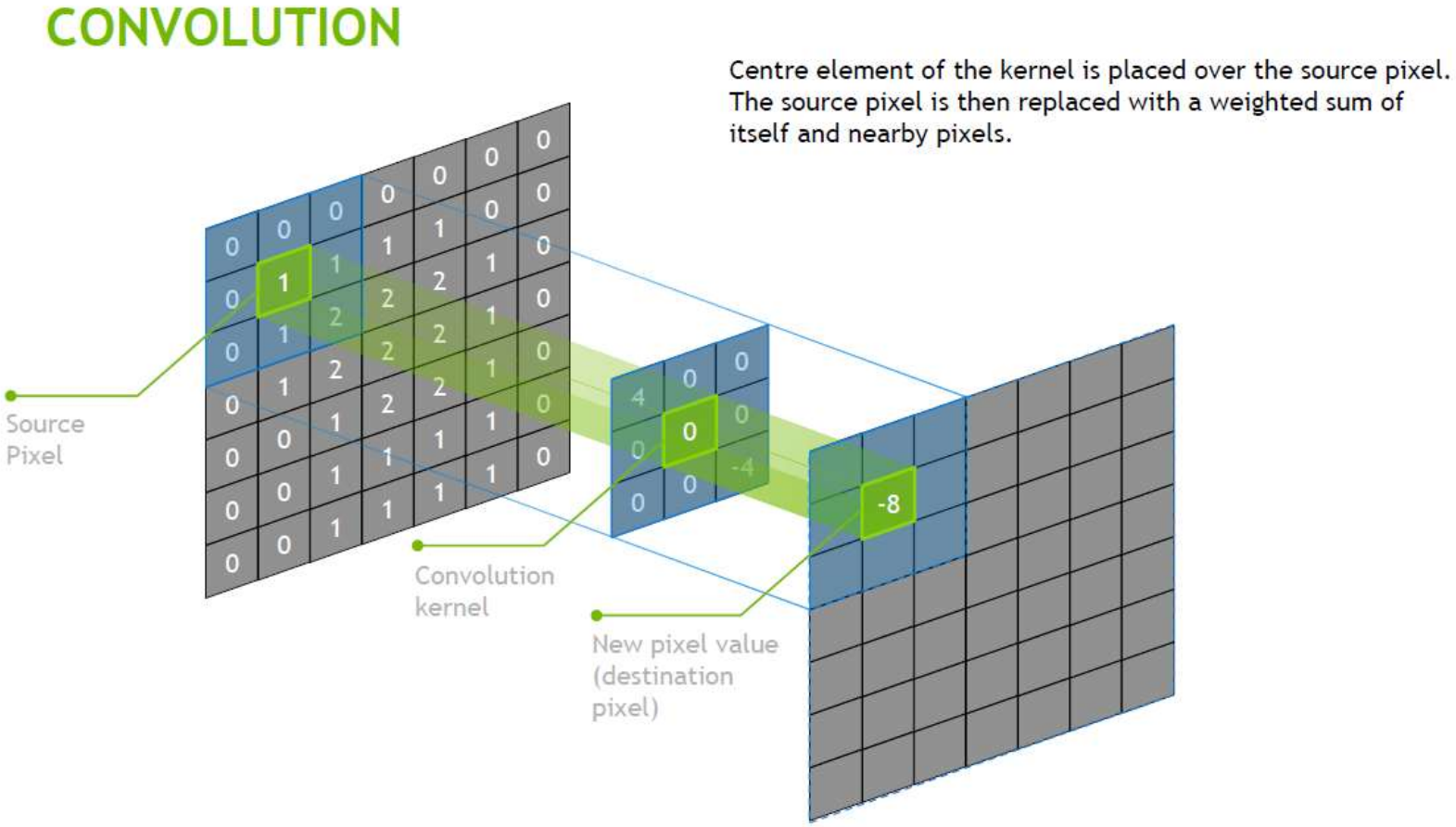
Grootste nadeel aan shallow learning is dat features voor classificatie hand-crafted zijn. In heel wat processen is dit een tijdsrovend en intensief gegevens.

Deep learning

- Past convolutiefilters toe op de gehele afbeelding.
- Het netwerk leert zelf intern complexe representaties vanuit de pixelinformatie die aan het netwerk aangeboden word.

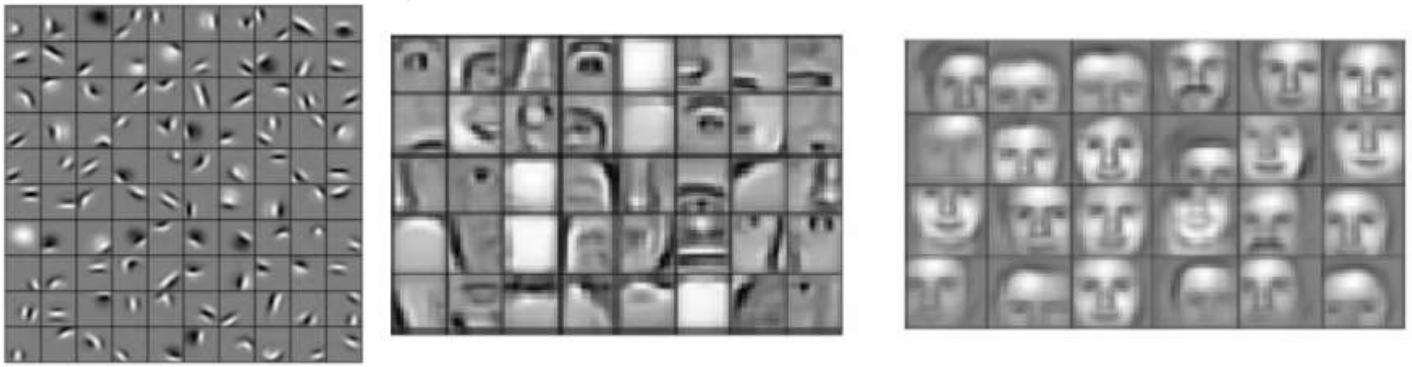
Ontstaan van Convolutional Neural Networks (CNN), die tot op heden een van de frequentst gebruikte architectuur is in computer visie oplossingen.

Deep learning, hoe werkt dit?



Deep learning, hoe werkt dit?

Convolution layers gaan op zoek naar gemeenschappelijke patronen over de trainingsdata, die steeds toenemen in complexiteit.

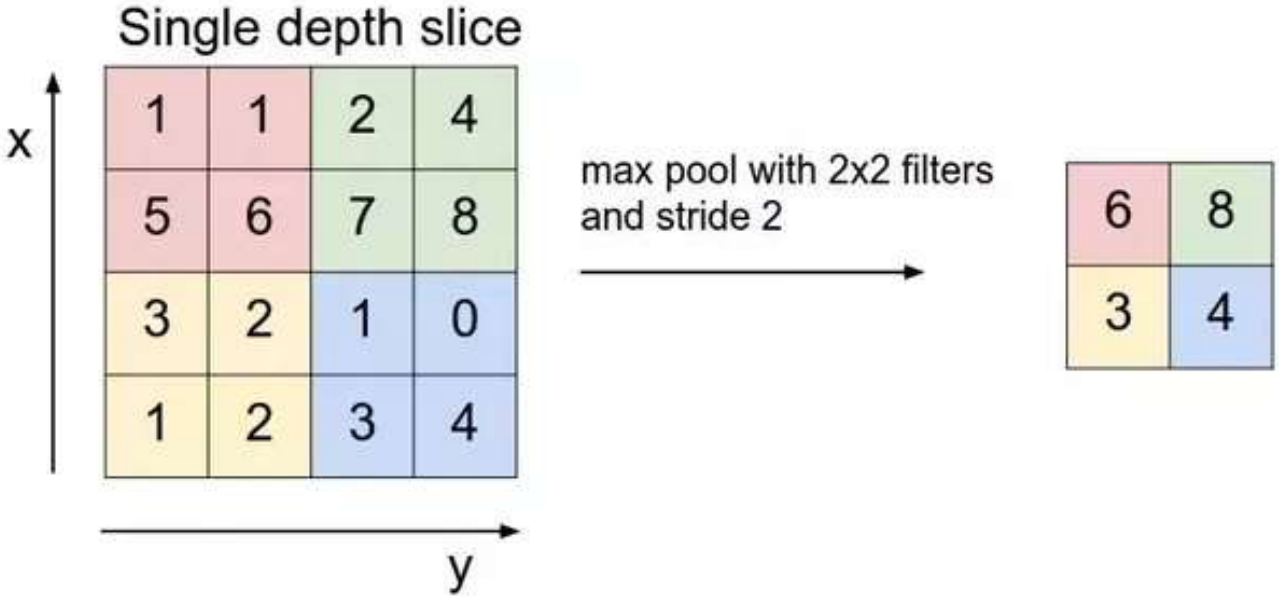


Na elke convolution layer volgt een subsampling layer.

- Verdere reductie van de dimensionaliteit.
- Wordt ook wel een pooling layer genoemd.
- Zorgt voor invariantie tegen rotatie en translatie.

Deep learning, hoe werkt dit?

Er zijn verschillende types pooling/subsampling layers, maar meest gebruikte zijn max-pooling en average-pooling.

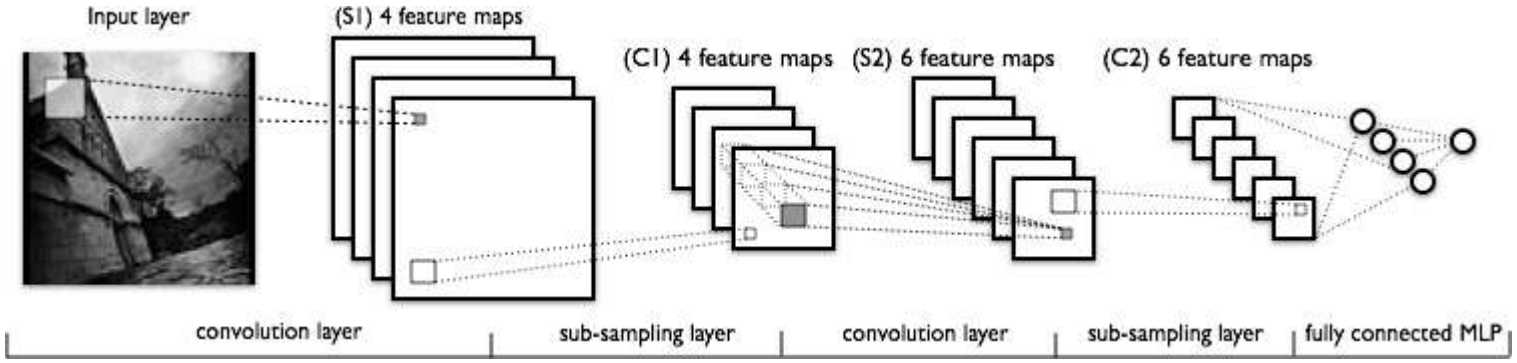


Deep learning, hoe werkt dit?

De reductie in dimensionaliteit (aantal convolutional layers) heeft rechtstreeks verband met

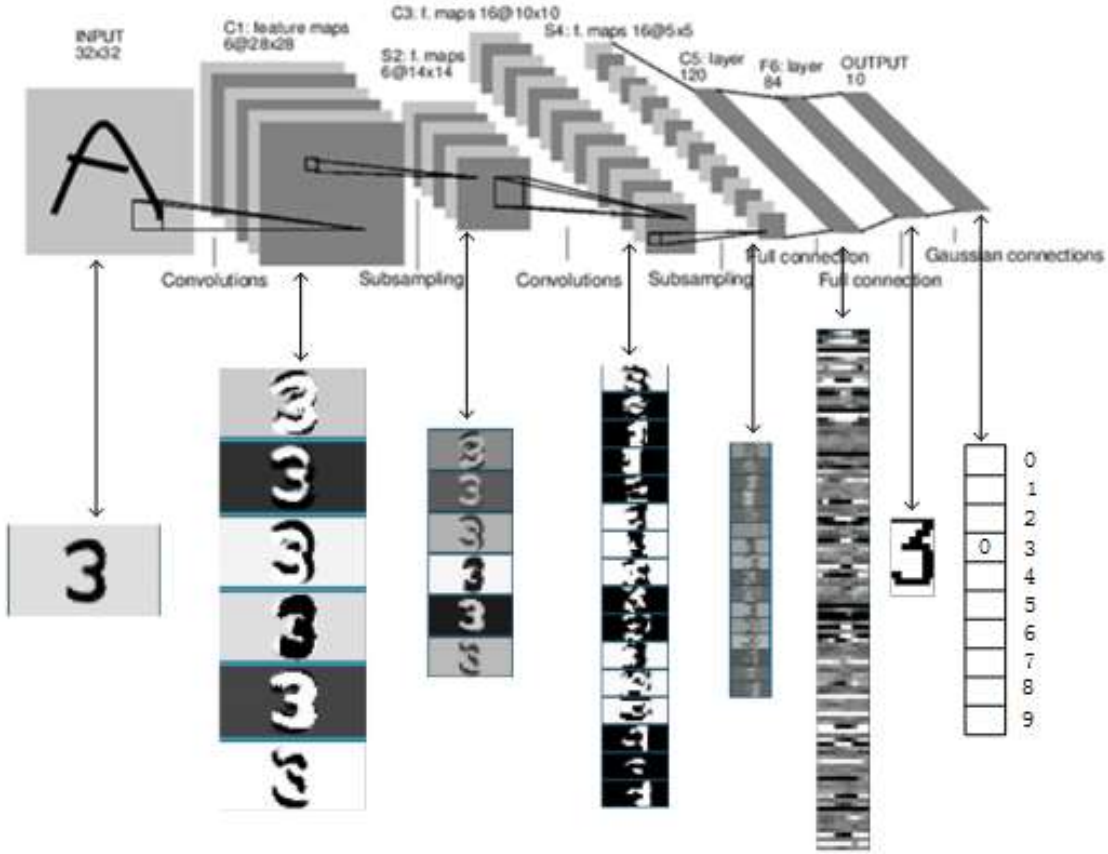
- De inputgrootte van de data: we starten immers op pixel niveau
- De gewenste finale feature grootte: eerste deel totale netwerk

Uiteindelijk kunnen we het geheel opsplitsen in een feature extraction netwerk en een classification netwerk, meestal bestaand uit fully connected layers.



Voorbeeld van een CNN

Eerste CNN van Yann Le Cun (LeNet), gebruikt voor classificatie van digits:



Data versus accuraatheid

Deep learning = optimalisatie van een complex systeem

- Elke neuron in het complexe netwerk heeft een gewicht per connectie.
- Dit leid al snel tot miljoenen parameters die geoptimaliseerd dienen te worden.
- Hiervoor is veel trainingsdata nodig, om een relatie te kunnen opbouwen tussen input (image pixels) en output (labels).

Er zijn heel wat publieke datasets

- Pascal VOC challenge (http://host.robots.ox.ac.uk/pascal/VOC/)
- Microsoft COCO dataset (http://mscoco.org/)
- KITTI vision benchmark (http://www.cvlibs.net/datasets/kitti/)

Jammer genoeg focussen deze nog steeds op de common classes: personen, voertuigen, dieren, …

Data versus accuraatheid

Hoe meer data beschikbaar, hoe accurater we het model kunnen trainen.

- Een vuistregel = (#number_of_weights)^2.
- Echter een gemiddeld netwerk heeft al snel 10000 weights …
- Je merkt dus hoe snel de nodige datavereiste kan ontploffen.

Om dus zelf je eigen model te trainen, heb je (los van de hardware) uiteraard ook deze data nodig.

- Er moet voldoende variatie in de data zitten.
- Het capteren van al deze data kan een tijdsintensief proces zijn.
- Al deze data moet finaal ook manueel geannoteerd worden met labels.

Data versus accuraatheid

Wanneer er weinig data beschikbaar is, dan is het trainen van een CNN mogelijk, maar de accuraatheid zal hieronder lijden.

- Klassieke technieken zoals een SVM vinden hier met veel minder data beter een optimale scheiding.
- Vooral omdat klassieke technieken veel minder parameters hoeven te optimaliseren.

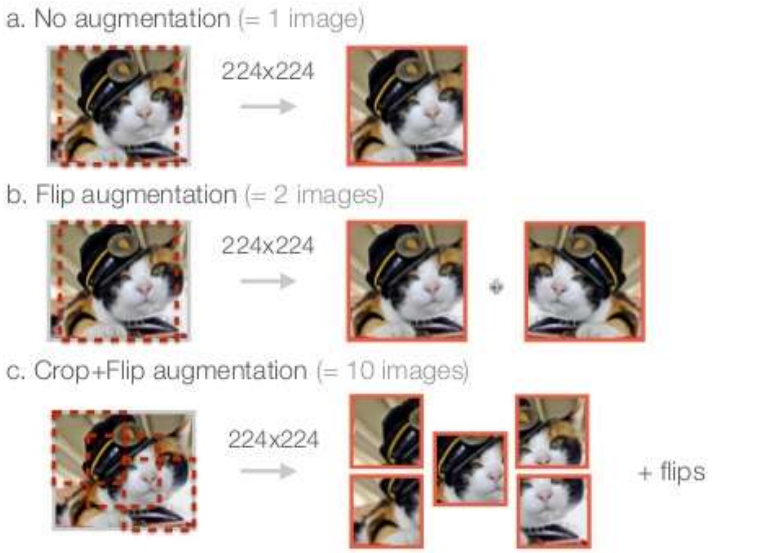
We moeten de hoop echter niet opgeven, want om dit probleem aan te pakken maakt men heel vaak gebruik van data augmentation.

Data augmentation

Data augmentation is het principe waarbij we zoveel mogelijk variatie in het proces introduceren op basis van de beschikbare trainingsdata.

- Horizontale en vertikale flips
- Rotaties rond het centerpunt
- Random crops uit de annotaties (robuustheid tegen occlusie)
- Translaties van de inputdata tov annotaties
- Kleurcasting + convertie naar andere color spaces
- Relighting van beelden
- Geometrische vervormingen
- …

Data augmentation



KU LEUVEN

Data augmentation

Maar het gaat nog veel verder!

Nieuwste trend in computer vision is data augmentation via computer generated content.

- Computer graphics slaagt erin uiterst realistische scenes na te bouwen.
- Dit laat ons toe om objecten te plaatsen, exact waar we het willen.

Grote bedrijven zoals Google & Facebook betalen werknemers om b.v.b. GTAV te spelen, om scenes te capteren, waarvan ze via de engine eigenlijk weten waar elk object zich bevindt.

- Introduceren van uiterst moeilijke condities wordt nu gemakkelijk (regen, mist, nacht/dag, crowds, …)
- Miljoenen frames worden zo gecombineerd to efficiente trainingsdata.
- Er zijn geen manuele annotaties meer nodig!

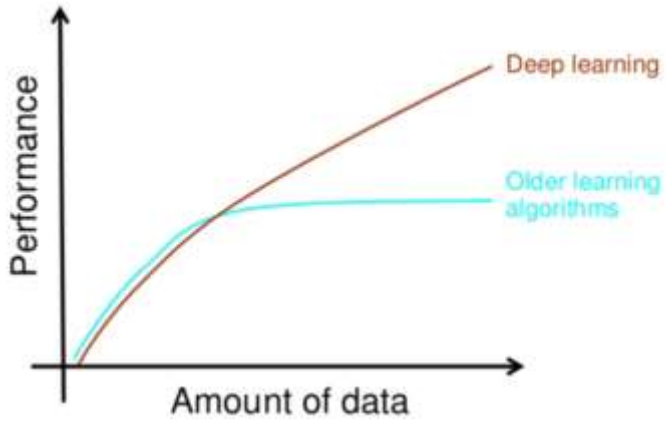
Data augmentation

FIGURE 1 : Various time and weather conditions possible in GTA5.

FIGURE 2 : Signs found in GTA5 match those described in Manual on Uniform Traffic Control Devices (10).

Hoe maak ik nu de keuze?

De keuze tussen traditionele machine learning en deep learning hangt af van heel wat parameters, maar toch vooral van hoeveelheid data.



Opgelet! Dit klopt in unconstrained object detection scenarios. Maar industriele toepassingen hebben net het voordeel heel wat constraints te bevatten. Klopt deze vergelijking dan wel atijd?

Hoe maak ik nu de keuze?

Traditionele Machine Learning

- Je kent de descriptieve features voor je applicatie.
- Je hebt een beperkte hoeveelheid data beschikbaar.
- Het toepassingsveld is drastisch verschillend van huidige deep learning onderzoek.
- Een duidelijke scheiding tussen de verschillende klassen.

Deep Learning

- Je kent de descriptieve feature voor je applicatie NIET.
- Je hebt onbeperkte trainingsdata ter beschikking.
- Je toepassingsveld valt midden in het toepassingsveld van huidig deep learning onderzoek.

Opgelet! Tot 2013-2015 was dit vrijwel een goeie opsplitsing.

Tegenwoordig kunnen deze standpunten met nieuwere technieken weerlegd worden.

Hoe maak ik nu de keuze?

`Toepassingsveld applicatie = toepassingsveld deep learning onderzoek.`

Op dat moment

- Vervalt de nood aan beschikbare trainingsdata.
- Heel wat onderzoekers stellen immers hun getrainde modellen ter beschikking, zoals de Caffe model zoo. (https://github.com/BVLC/caffe/wiki/Model-Zoo)
- Meest gebruikte klassen
  - 20 klassen van Pascal VOC challenge: person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor
  - 90 klassen van COCO dataset: zie competitie
  - 1000 klassen van IMAGENET: zie competitie

Hoe maak ik nu de keuze?

`Toepassingsveld applicatie = toepassingsveld deep learning onderzoek.`

- De uitdaging ligt hem dan vooral in het aan de praat krijgen van de onderzoeks- / opensource software.
- Training zonder GPU is onmogelijk
  - Door de grote hoeveelheid data
  - Gemiddeld model zou weken tot maanden trainen voor optimale gewichten op alle nodes van het model
- Inferentie zonder GPU is wel mogelijk, puur CPU gebaseerd
  - Het uitvoeren van een getrained model op CPU werkt als je geen real-time vereisten hebt binnen je toepassing.
  - B.v.b. een 640x480 pixel afbeelding kan je door een deep learning segmentatie techniek laten onderverdelen in segmenten
    - CPU: 2-3 min per afbeelding
    - GPU: 100 msec per afbeelding

Hoe maak ik nu de keuze?

`Toepassingsveld applicatie =/= toepassingsveld deep learning onderzoek.`

Op dat moment

- Heb je trainingsdata nodig…
- Wil dat dan zeggen dat je geen deep learning kan uitvoeren?

NEEN! Deep learning is nog steeds mogelijk via een techniek genaamd transfer learning.

- Je vertrekt vanuit een bestaand model, met bestaande weights voor alle parameters, die je inlaad in je trainingsprotocol.
- Je voegt extra kennis toe vanuit een beperkte gelabelde dataset, specifiek gekoppeld aan de klassen die je wilt terugvinden.

Transfer learning

Algemeen gaan we ervan uit dat deep learning

- In zijn convolutional layers op zoek gaat naar universele object features.
- In zijn fully connected layers deze features `verbind` tot een object.

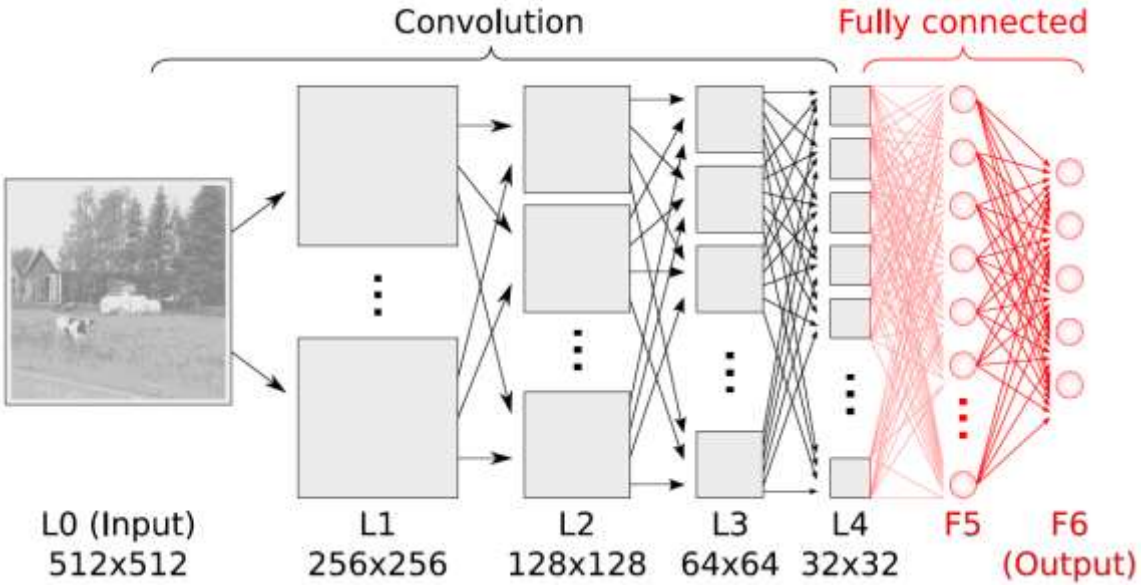
Vanuit dat oogpunt kunnen we extra kennis toevoegen aan een reeds getrained model, of de kracht van een bestaand model gebruiken

- Convolutional neural network als feature extraction techniek
- Fine-tuning van een convolutional neural network

CNN als feature extraction techniek

Hier gebruiken we de kracht van een CNN om features te filteren uit een inputbeeld van pure pixels, op basis van de getrainde klassen.

- Verwijder fully connected layers van het model.
- Hierdoor krijg je een convolutional feature vector van 32x32 = 1024 feature values.
- Gebruik dit in een klassieke machine learning classifier zoals SVM, boosting, random trees, …



CNN als feature extraction techniek

Voordelen van CNN als feature extraction techniek

- Feature representatie van een object zonder manual crafted features.
- CNN features hebben momenteel een grote performance boost t.o.v. meeste combinaties van hand crafted features.
- Door trainen van model over heel wat verschillende klassen, abstraheer je eigenlijk het concept object, en splits je dit op als een combinatie van onderdelen.

Nadelen van CNN als feature extraction techniek

- Zonder de nodige hardware ter ondersteuning (GPU) is dit een traag process, wat niet aan real-time processing eisen voldoet.
- Kunnen een makkelijk scheidbaar proces ook overcomplex maken. Is dus zeker niet altijd het beste alternatief!

Fine-tuning van een CNN

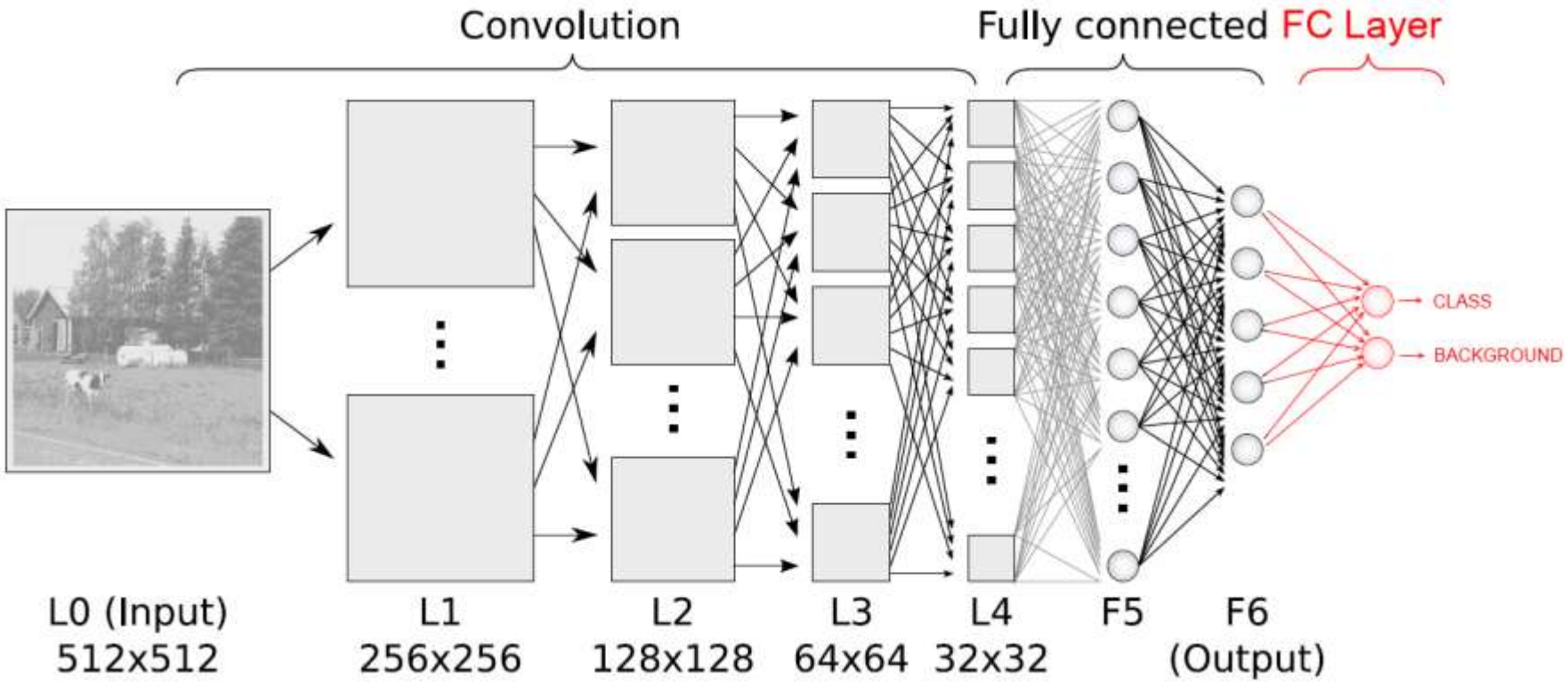
Hier gaan we het bestaande model verder finetunen via het backpropagation trainings algoritme op de beschikbare data.

De te volgen stappen

1. Aanpassen van de fully connected layers aan het aantal klassen specifiek voor de applicatie waar je het netwerk wil gebruiken.
   - Voorzien van een extra fully connected layer
   - Laatste fully connected layer vervangen
2. Beslissen welke weights van welke lagen je gaat blokkeren.
   - OF vaste gewichten voor de convolutional layers EN fully connected layers
   - OF vaste gewichten voor de convolutional layers zonder FC layers
   - OF alle gewichten updaten aan nieuwe classificatie probleem

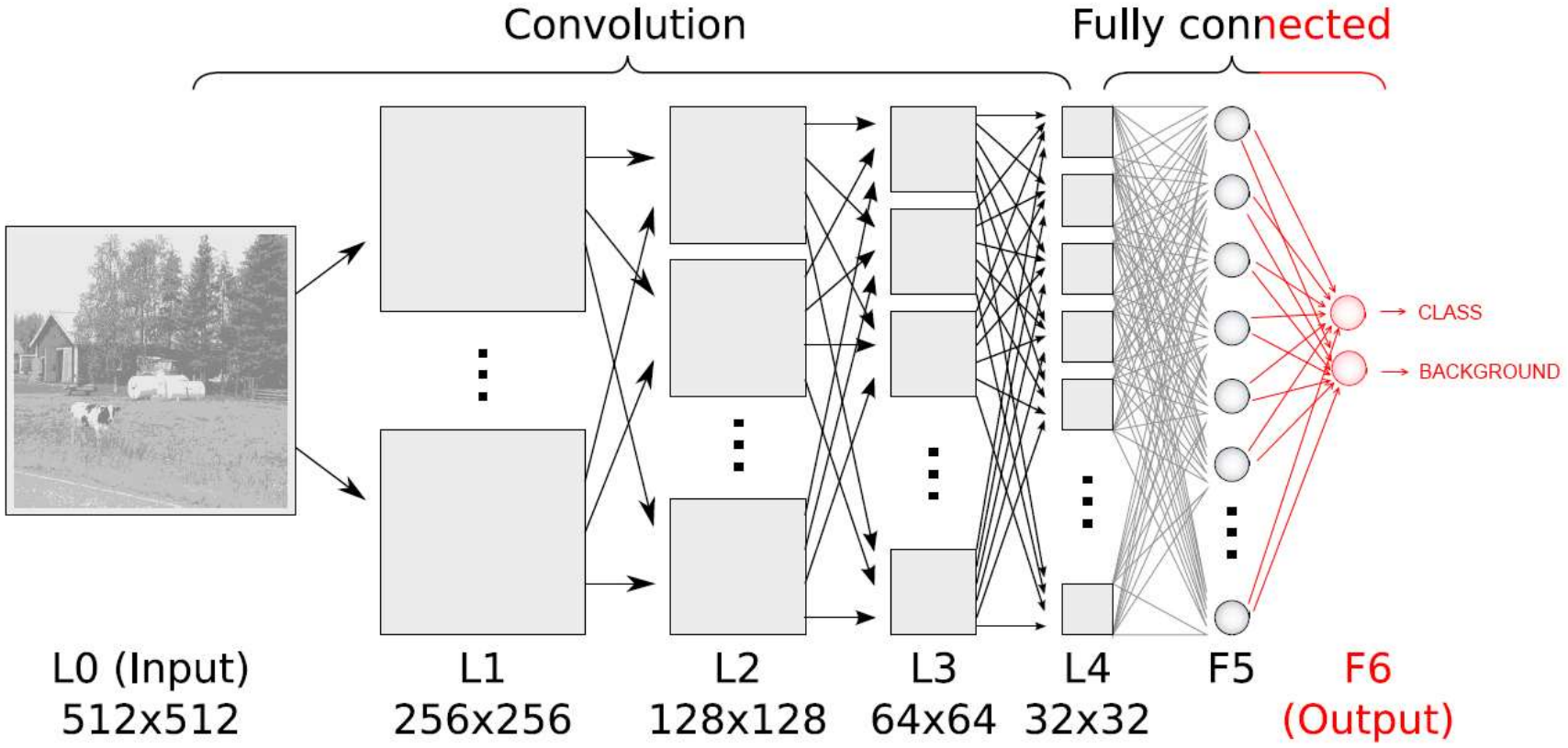
Fine-tuning van een CNN

Aanpassen van de fully connected layers via extra FC layer



Fine-tuning van een CNN

Aanpassen van de fully connected layers via wijzigen van de laatste fully connected layer naar nodig aantal klassen.



Fine-tuning van een CNN

Opletten met vastzetten of verder updaten van weights

- Convolutional layers zijn er vooral om generische features te halen uit de input samples, terwijl de fully connected layers meer dienen om de klasse specifieke combinaties te maken.
- Alle lagen verder updaten zou kunnen leiden tot overfitting van de trainingsdata.

Voordeel aan gebruik van fine-tuning

- Moderne CNN's op ImageNet trainen
- 2-3 weken over meerdere GPU's.
- Hiervoor worden 1,2 millioen trainingsbeelden gebruikt, waarop data augmentation wordt toegepast.
- Fine-tuning kan drastisch sneller. Op een volledig nieuwe klasse kan je een model finetunen op enkele uren.

Fine-tuning van een CNN

Nieuwe kleine dataset + data is gelijkaardig aan trainingsdata

- Finetuning weights kan snel leiden tot overfitting trainingsdata.
- Vooral omdat de higher level features van originele model relevant zijn tov de nieuwe trainingsdata.
- Beste oplossing → lineaire classifier op CNN features / codes.

Nieuwe grote dataset + data is gelijkaardig aan trainingsdata

- Meer data geeft ons zekerheid dat we het model niet overfitten.
- Finetuning van volledige netwerk is mogelijk.

Nieuwe kleine dataset + data is verschillend aan trainingsdata

- De data staat enkel toe een lineaire classifier te leren op basis van CNN features.
- Omdat data echter heel verschillend is nemen we beter output van wat vroegere layers, waar nog geen klasse specifieke features geleerd worden.

Nieuwe grote dataset + data is verschillend aan trainingsdata

- Aangezien de data zeer verschillend is, finetunen we best alle weights van het netwerk.
- Voorgetrainde weights heeft nog steeds zin, in plaats van random initialisatie te nemen.

Fine-tuning van een CNN

Belangrijk om te onthouden bij fine-tuning

- De input data moet steeds voldoen de eisen van de originele trainingsdata. Dit mede door het feit dat de convolutional layers en de fully connected layers afgestemd zijn op een bepaalde resolutie (initieel aantal pixels).
- Het kan dus zijn dat je een groter input beeld, sliding window-gewijs, zal moeten opsplitsen in sub-beelden, zodanig dat je elk deel afzonderlijk classificeert via het convolutional neural network.
- In het trainingsproces moet je een learning rate opgeven. Het is bij finetuning belangrijk om de learning rate lager te nemen dan bij het origineel model. We gaan er immers van uit dat de weights reeds iets betekenisvol doen en dus willen we die niet te snel drastisch veranderen.

Deep learning als black box

- Een neuraal netwerk, wat doet dat nu exact?
- Hoe komen we te weten wat er intern in elke laag gebeurt?
- Hoe zien de gevonden features in een netwerk er nu juist uit?

Heel wat ontwikkelaars / academici gebruiken dit argument nog steeds om aan te geven dat deep learning heel complex is, en dat het moeilijk is om voorbij de theorie te kijken.

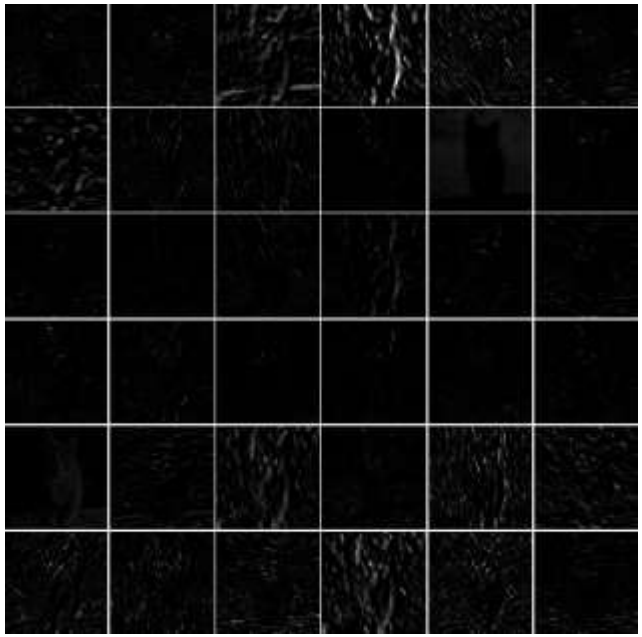
Daarom gebeurt er heel wat onderzoek rond de visualisatie van de verschillende stappen binnen deep learning.

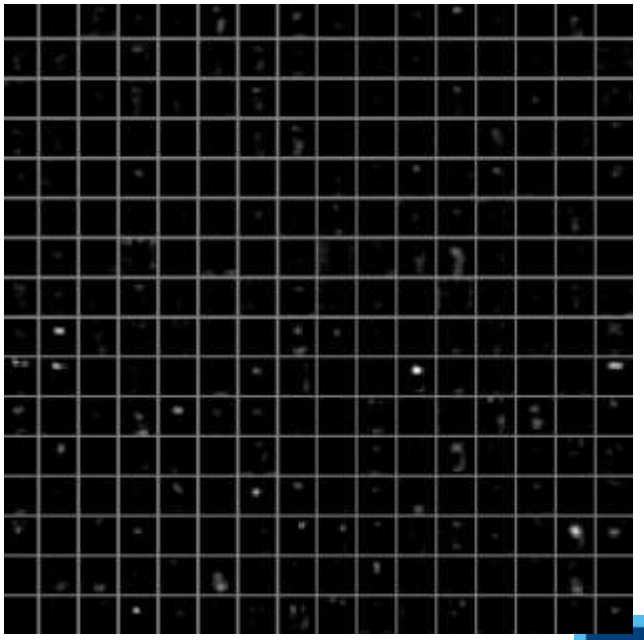
Visualisatie van deep learning

Een eerste stap is het visualiseren van de layer activations.

- In begin blobby en dense, naar eind van model steeds specifieker en lokaler.
- Als we merken dat veel activation maps leeg zijn, kan dit een indicatie zijn van `dead filters`, wat zorgt voor grote learning rates.

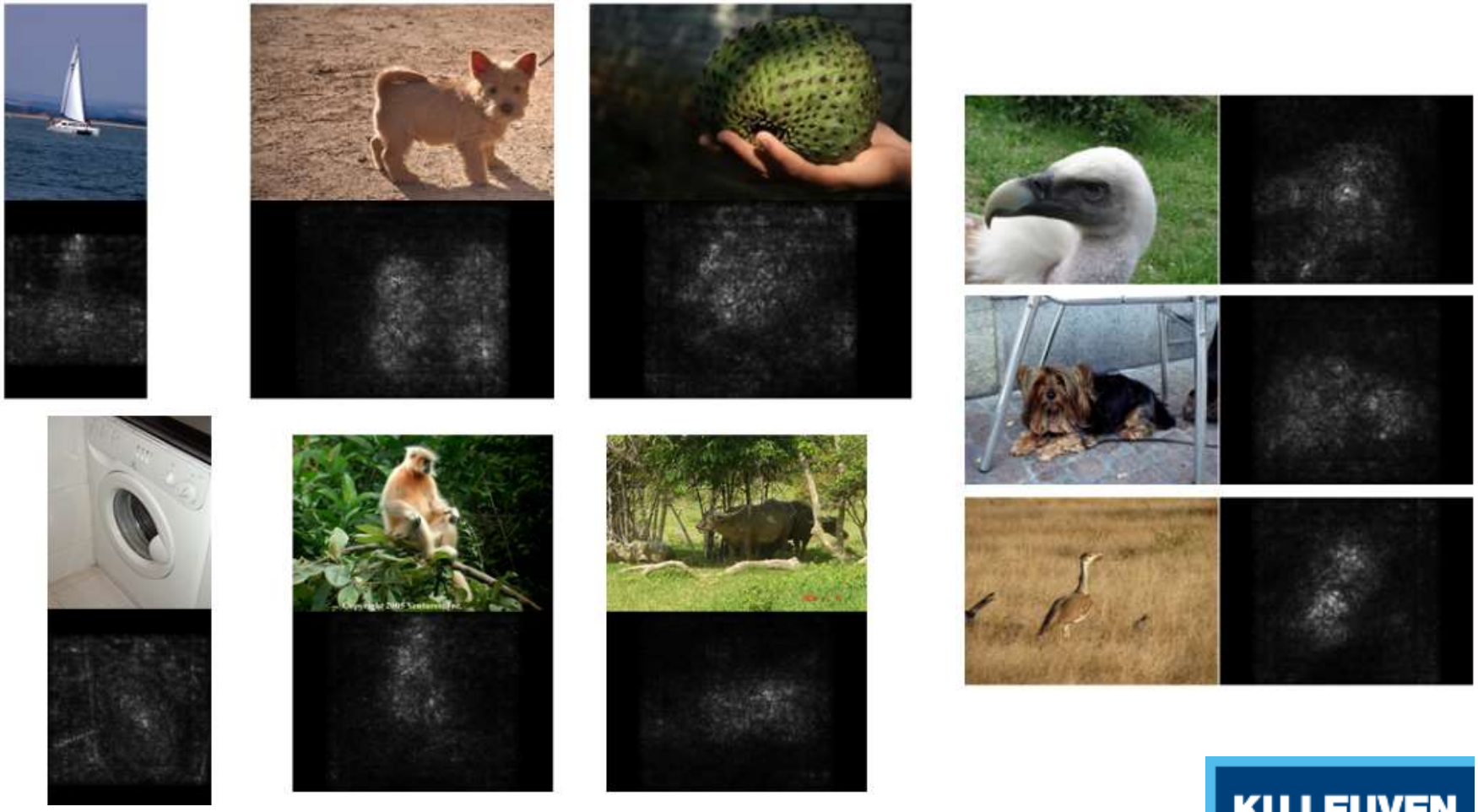
CONV1 & CONV5 layer bij bekijken van kat via AlexNet





Visualisatie van deep learning

Een eerste stap is het visualiseren van de layer activations.

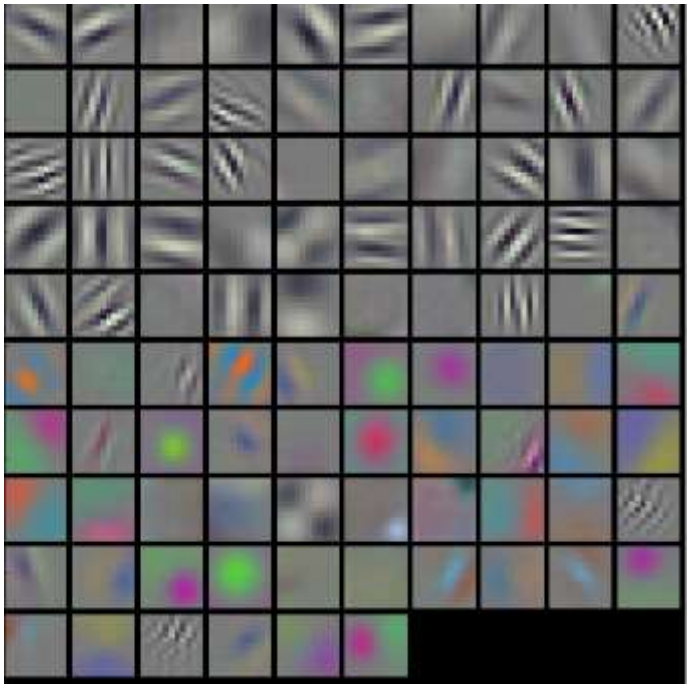


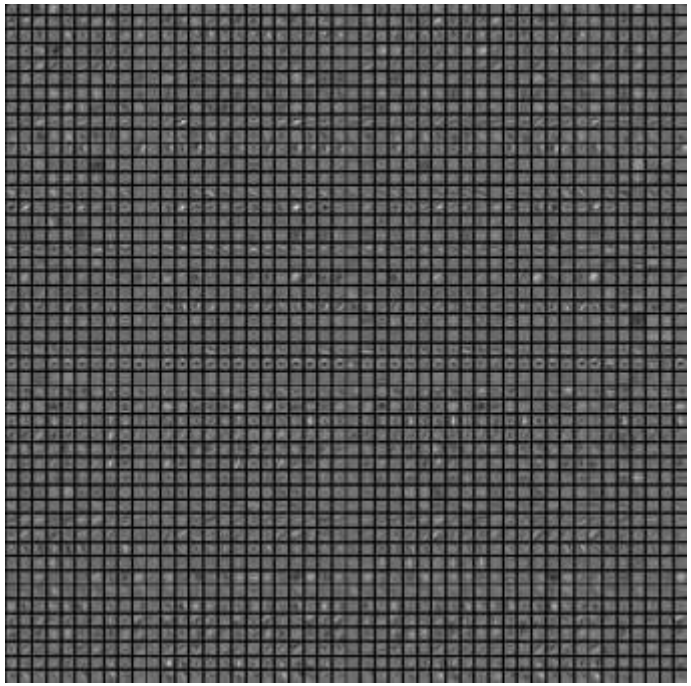
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman

Visualisatie van deep learning

Een tweede mogelijkheid is het visualiseren van de weights

- Deze zijn het duidelijkst in de beginnende lagen, en geven een duidelijke filterstructuur weer.
- Hoe smoother de structuren, hoe beter de lagen geconvergeerd zijn naar een optimale oplossing.

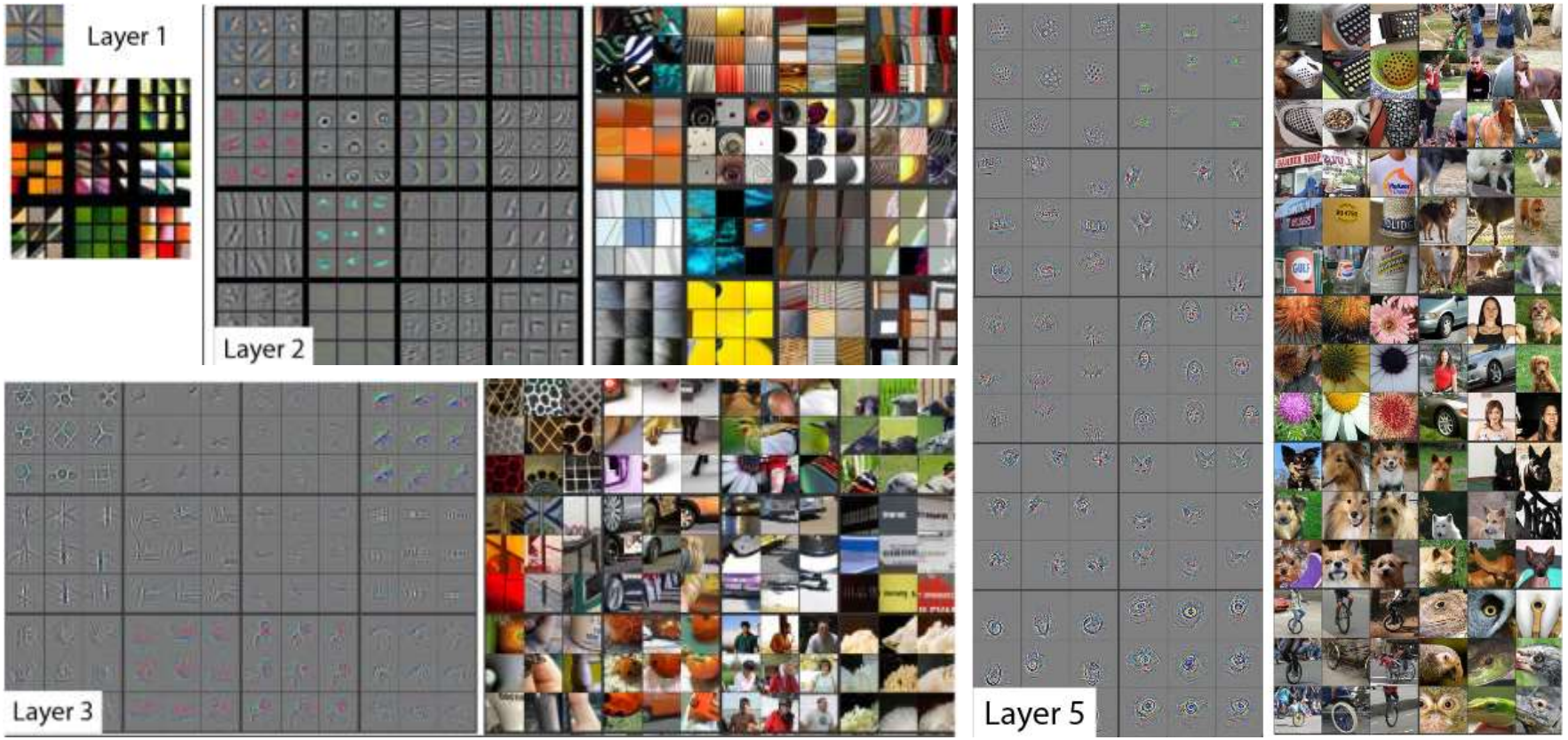




CONV1 en CONV2 layers voor AlexNet

Visualisatie van deep learning

Een tweede mogelijkheid is het visualiseren van de weights



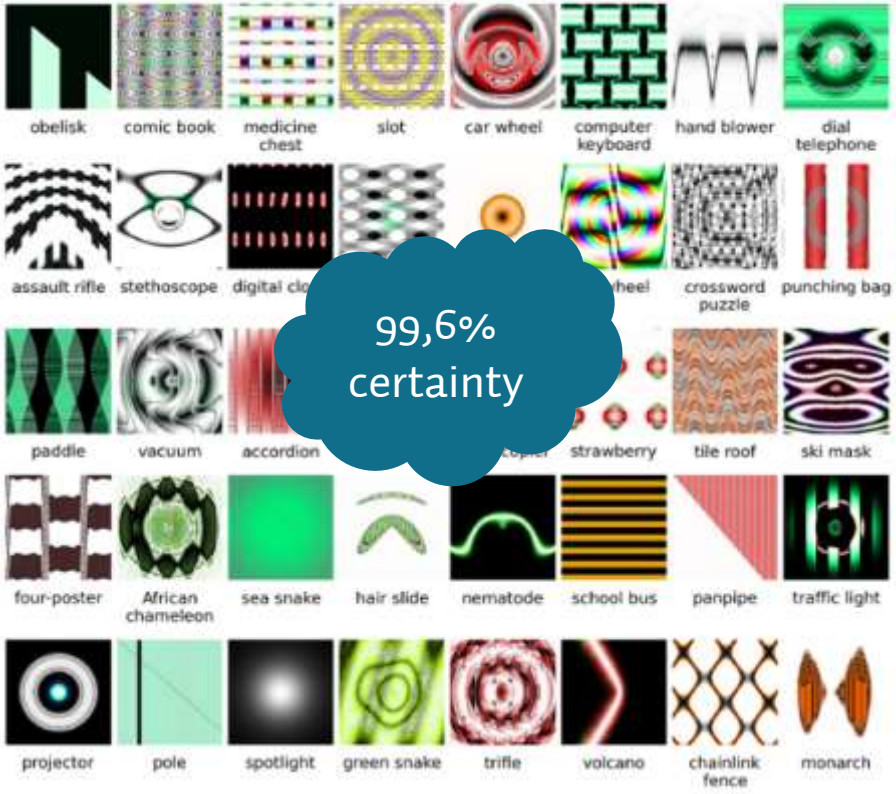
Visualisatie van deep learning

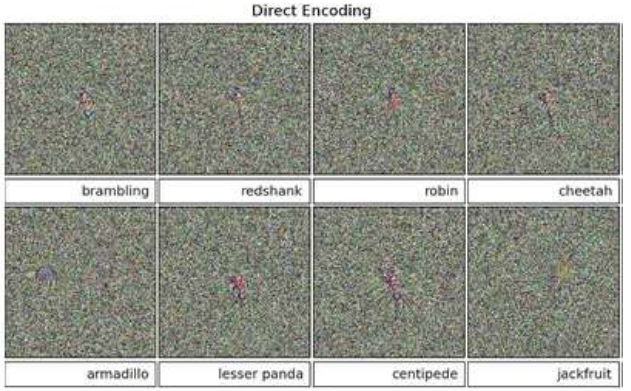
De derde mogelijkheid is de activatie van een bepaalde node gaan bekijken terwijl we data doorheen het netwerk propageren. Zo weten we waar de neuronen naar op zoek zijn.

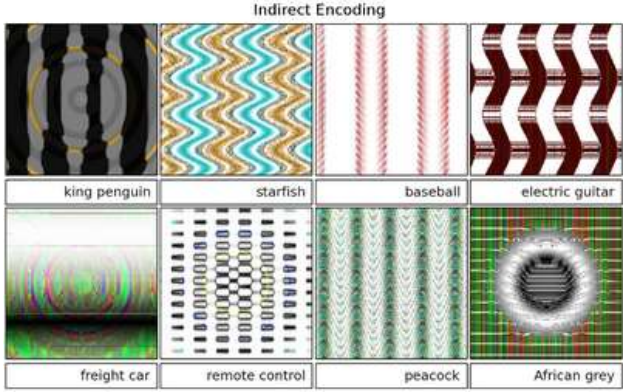
Beelden met maximale triggering op 5th max-pooling layer van AlexNet

Fooling CNN's

Door diepgaand onderzoek naar visualisaties van de interne lagen van een deep learning model, werd de vraag gesteld of we artificiele data kunnen genereren die specifiek triggered waar nodig.







KU LEUVEN

Is GPU altijd beter dan CPU?

- NEE dat is het niet! Is een veelgehoorde mythe binnen computervisie communities, dat alles beter oplosbaar is met een GPU.
- Heel afhankelijk van de hardware naast de GPU, beschikbaar in je systeem.
  - Hoeveel CPU cores heb je beschikbaar?
  - Kan het systeem gebruiken van optimalisaties zoals SSE, AVX, NEON, …
  - Heb je parallelle frameworks ter beschikking op je systeem?
  - …
- Veel computer visie bibliotheken maken gretig gebruik van deze CPU instructie sets. Maak er dus gebruik van.
- Er is immers altijd een bottleneck, wanneer je data van CPU naar het GPU geheugen brengt en terug.

Is GPU altijd beter dan CPU?

Een toepassing van gezichtsdetectie op CPU en GPU, gevolgd door facial landmark detectie en tracking.

CPU

Intel(R) Xeon(R) CPU E5

GPU

TitanX(P)

DEEL 2:
Deep learning technieken en toepassingen

Originele insteek deep learning

Deep learning = vertaling van een patch pixels → label

- Oplossing voor classificatietaken
- Geen (halve) oplossing voor localisatietaken
  - Sliding window gewijs kan je door het beeld lopen en op elke patch bepalen hoe groot de kans is dat er een object voorkomt
  - Om geen objecten te missing is overlap tussen patches nodig, maar dat maakt het heel traag

- Geen (ruwe) oplossing voor segmentatietaken
  - Opnieuw kan een patch based approach gebruikt worden. Deze patches moeten nadien echter efficient geclustered worden, wat opnieuw een zeer complex gegeven is.

Daarom heeft men doorheen de tijd heel wat nieuwe en complexere architecturen uitgevonden om aan de bovenstaande vraag te voldoen.

Deep learning architecturen

Er bestaat een wildgroei aan architecturen binnen het deep learning domein.

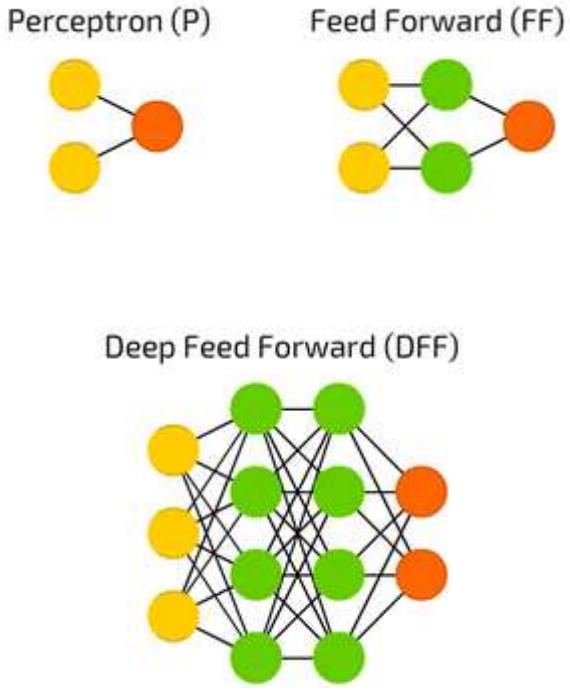
- Volledig overview: Asimov institute – Neural Network Zoo
- Niet alle modellen/architecturen zijn bruikbaar voor beeldverwerking
- Basis van heel wat modellen ligt in de speech recognition and textual analysis.

Elke architectuur heeft heel wat toepassingen, we bespreken alvast de belangrijkste architecturen binnen de beeldverwerking.

Deep learning architecturen

De basisnetwerken

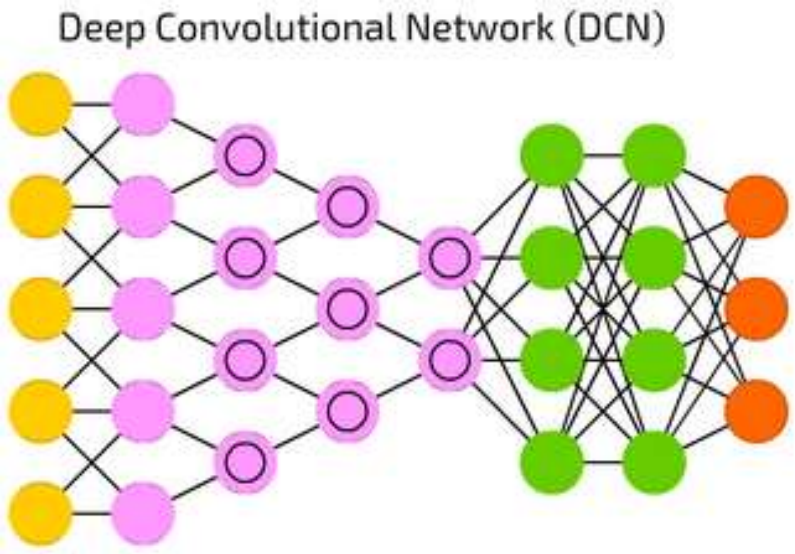
- SLP & MLP (zie eerste blok)
- Deep Feed Forward Network
  (fully connected network)
- Geen data cycles/loops mogelijk
- Training via backpropagation
  (meest populaire techniek)
  (error per patch terug door netwerk)
- Optimalisatie gewichten via gradient descent



Deep learning architecturen

Deep convolutional neural networks

- Een combinatie van convolution filters en pooling filters.
- Van daaruit komen we tot een feature representatie.
- Daarop word een reeks van fully connected hidden layers gekoppeld.
- Deze dienen als classificatie tot de eindklassen.



Deep learning architecturen

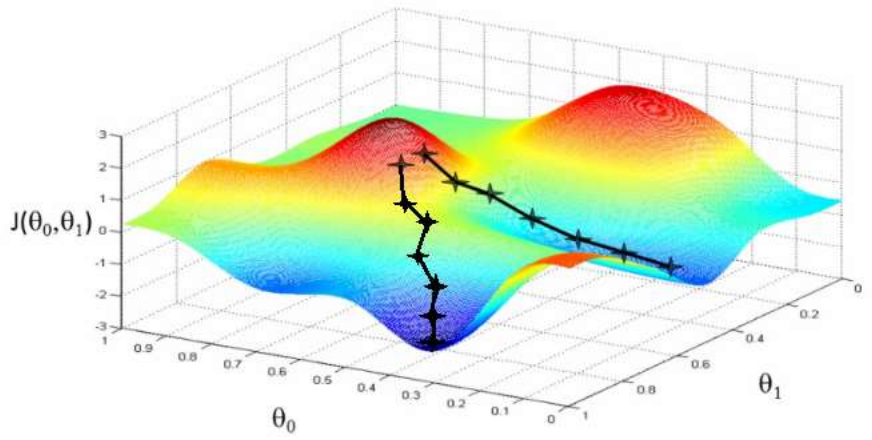
Gevaren bij feed forward netwerking

- Backpropagation werkt enkel als het voldoende keren herhaald wordt (aantal trainings cycli) en met wat `geluk` bij gezette stappen.

  ---> Hierdoor kan convergentie naar minima lang duren

- Netwerk convergeert wanneer de error op samples+label voldoende klein is.
- Gradient descent (weights aanpassen zodat we dalen op de error curve) heeft als gevaar vast te zitten in een lokaal minima.

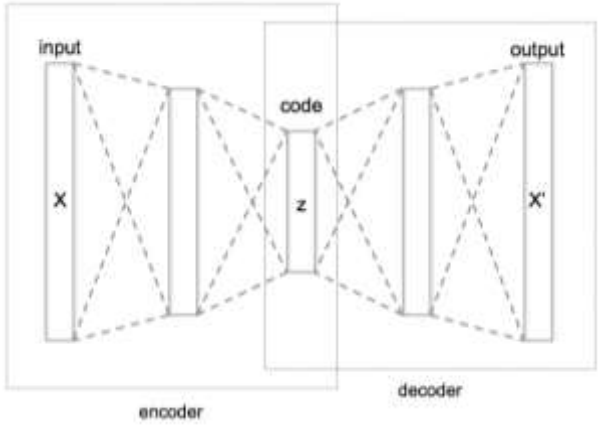
  ---> Oplossing: herhalen met verschillende random initialisaties



Deep learning architecturen

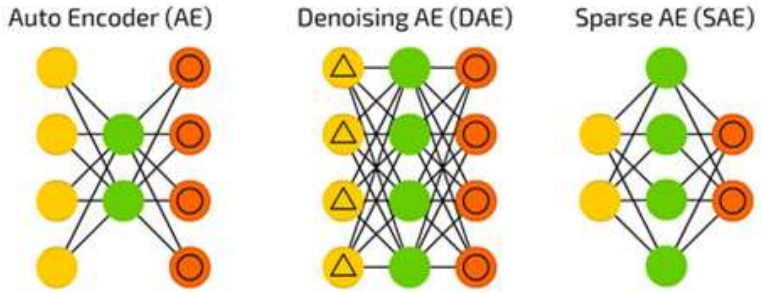
Auto-encoders

- Leren van efficiente codering / representaties van data.
- Wordt dikwijls gebruikt als dimensionaliteitsreductie.
- Altijd zelfde aantal input en output nodes, waardoor het netwerk zelf probeert zijn input te herconstrueren.
- Opgesplitst in encoder en decoder deel, waarbij de optimale encodering tussen encoder en decoder te vinden is.



Deep learning architecturen

Auto-encoders: soorten



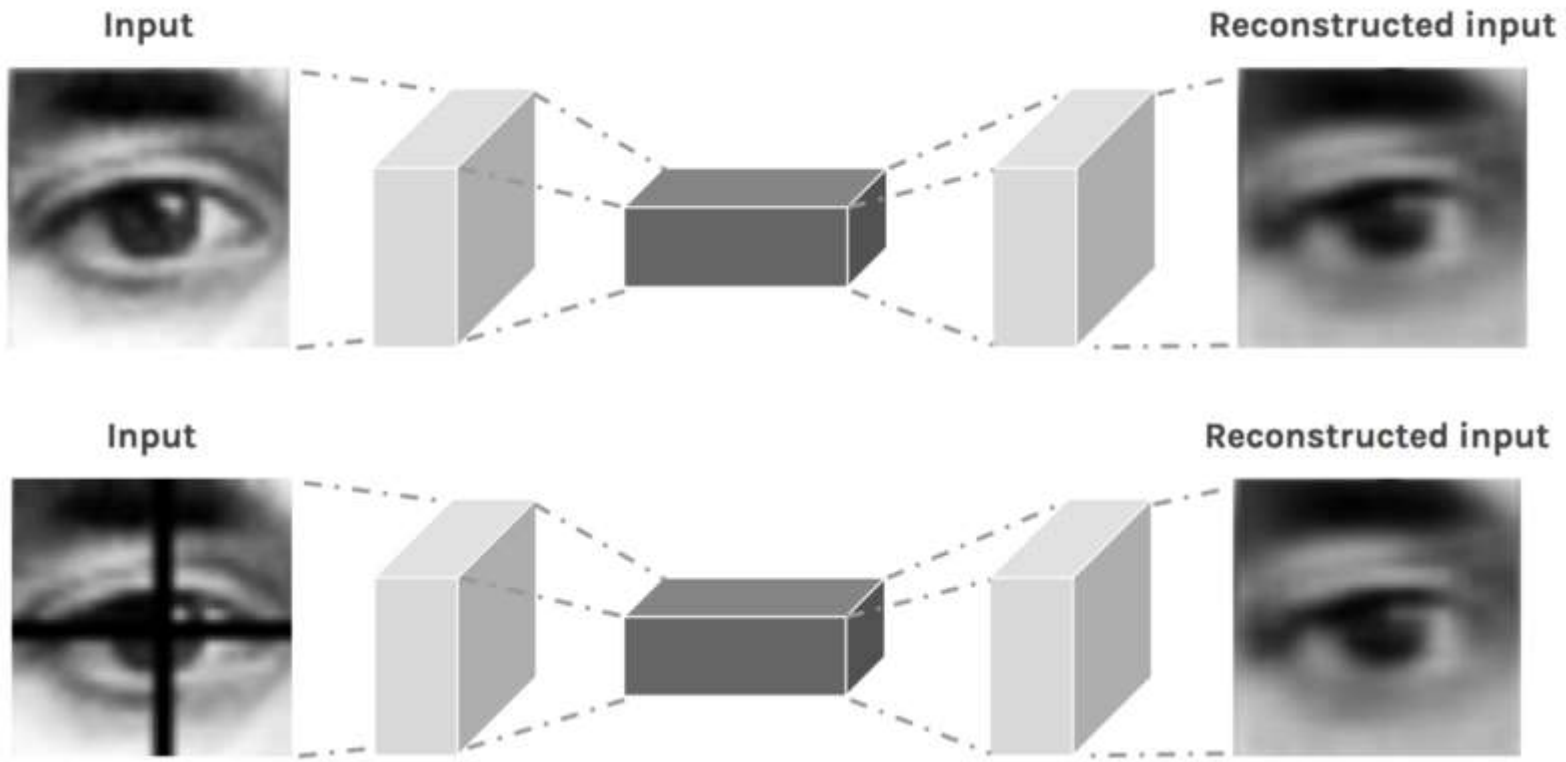
- Denoising AE: op basis van corrupte input, proberen het origineel signaal te reconstrueren. Wordt ingetrained op manueel gecorrupteerde data.

- Sparse AE: door meer hidden nodes te gebruiken dan inputnodes, forceren we het `sparse` gedrag van de data. Hierdoor kan een AE belangrijke structuren uit inputdata filteren.

- Wanneer je weinig gelabelde data hebt, dan toont onderzoek aan dat een AE getrained op die data, met daarbovenop een lineaire SVM als classificatie, betere resultaten geeft dan CNN.

Deep learning architecturen

Auto-encoders – removing artefacts from images



Deep learning architecturen

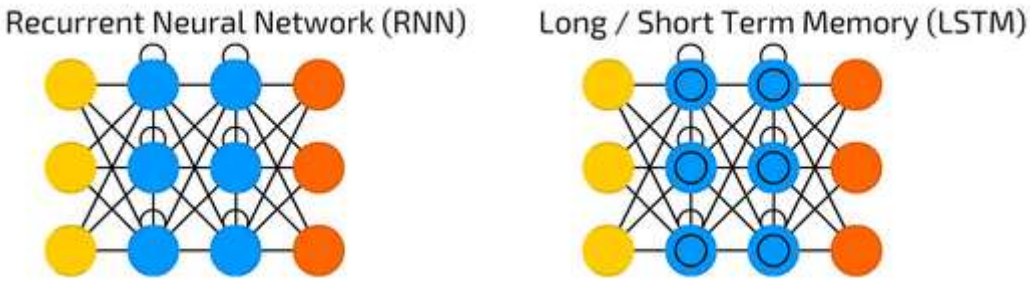
Auto-encoders – enhancing images

Deep learning architecturen

Recurrent neural networks

- In de hidden layers van het model, zijn gestuurde cycles/loops toegestaan.
- Op die manier gebruiken ze hun state als een soort geheugen.
- Dit staat ons toe om relaties in sequenties te leren, waardoor ze meer kunnen dan enkel per pixelset een output leveren.

- LSTM is een specifieke vorm van RNN
  - Door backpropagation training, kunnen errors bij RNN soms incorrect terug door het netwerk geduwd worden. [verdwijnen/exploderen]
  - Door invoeren van forget gates, wordt dit probleem vermeden.



Deep learning architecturen

Recurrent neural networks

- RNN en LSTM zijn vooral populair in speech recognition, waar ze nog steeds state-of-the-art resultaten behalen.
  - Google text-to-speech synthesis: android assistant / google voice search
  - Baidu speech recognition

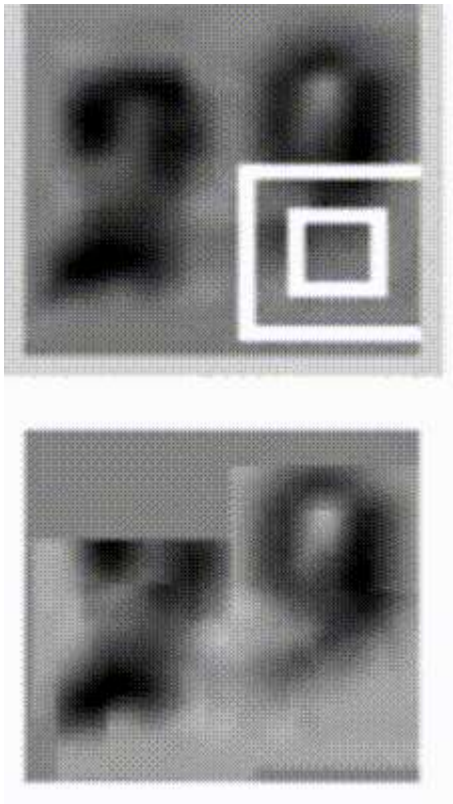
- Haalde ook heel nauwkeurige resultaten bij written digit recognition.

- Gecombineerd met CNN haalt dit state-of-the-art resultaten voor image captioning.

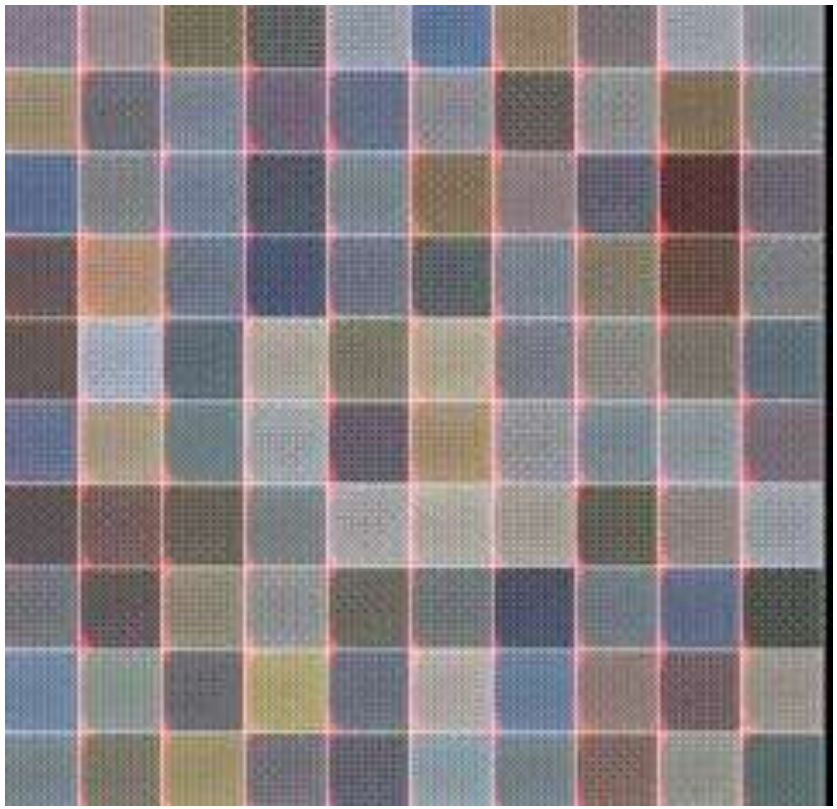
Deep learning architecturen

Recurrent neural networks (Google street view)

Herkennen van huisnummers

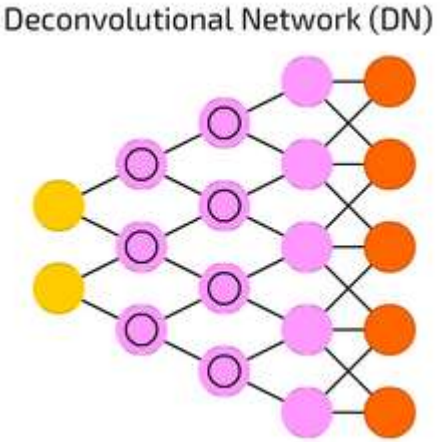


Een nummer laten genereren door hetzelfde netwerk

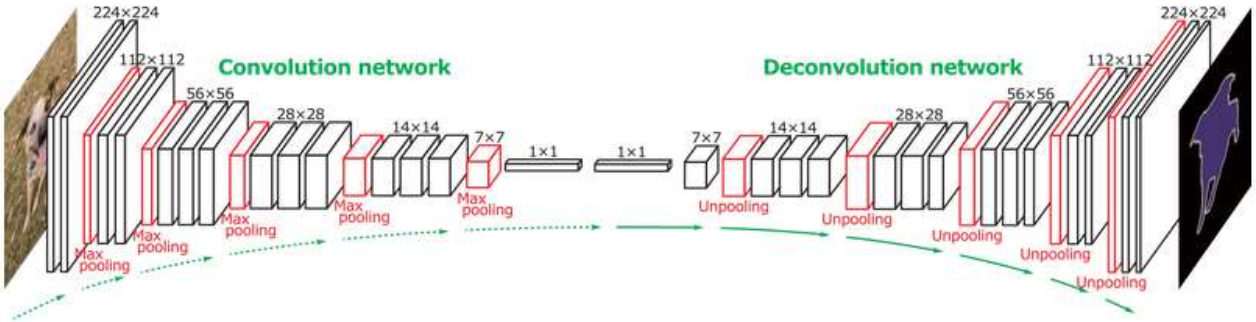


Deep learning architecturen

Deconvolutional networks



- Laat ons in eerste plaats toe om een visualisatie te maken van de convolutie filters toegepast in een netwerk.
- Maar laat ons ook toe een stap verder te gaan door CNN te koppelen met een DN, en zo een netwerk te krijgen die een complete mapping leert.
- End – to – end deep learning voor bijvoorbeeld segmentatie.



Deep learning architecturen

Deconvolutional networks

- Deconvolutional neural networks volgen de structuur van hun tegenhangende convolutional neural networks.
- Zowel de convolution als de pooling filters moeten geinverteerd worden.

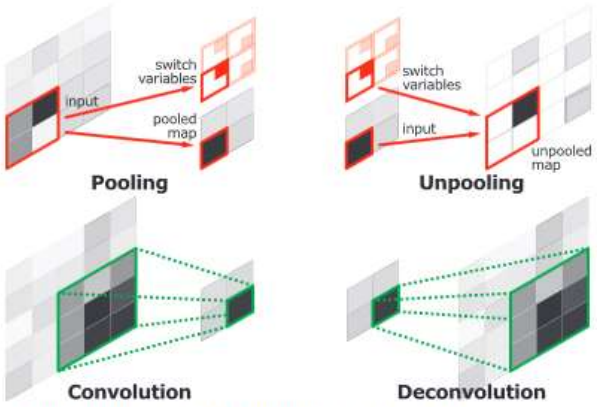
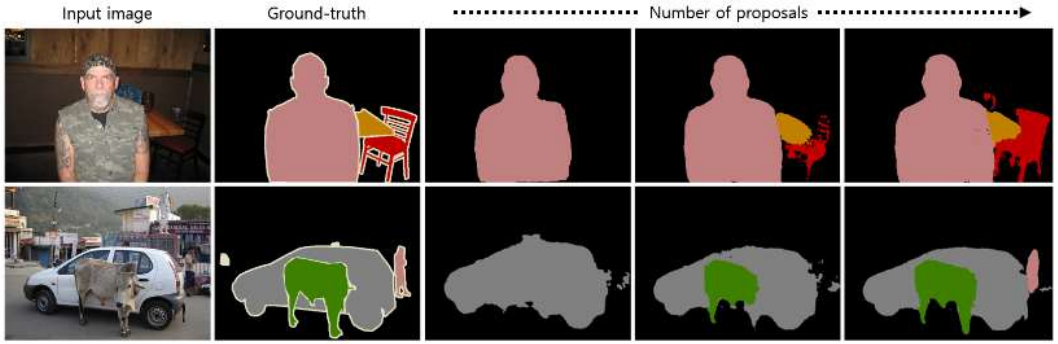
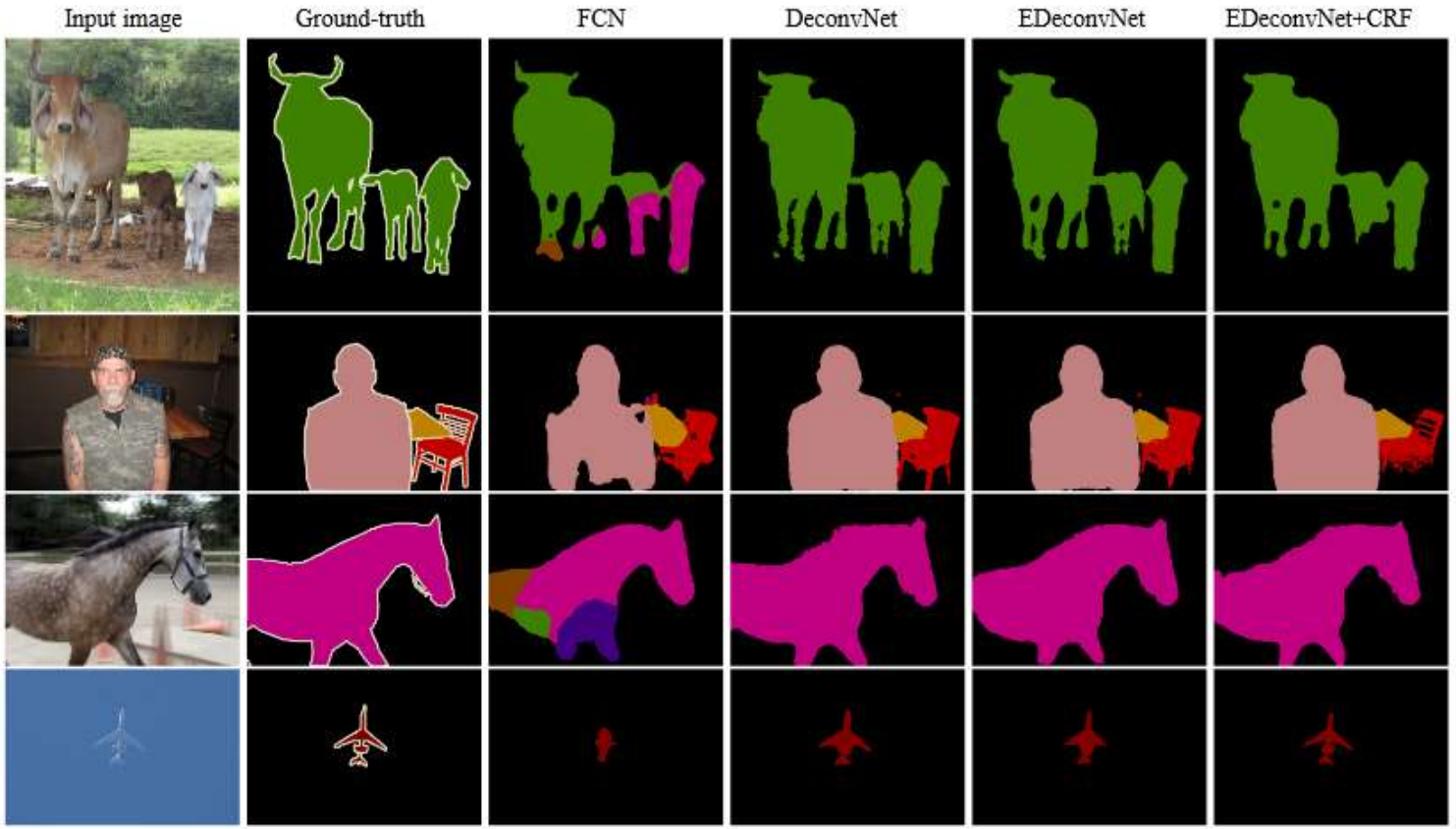


Figure 3. Illustration of deconvolution and unpooling operations.



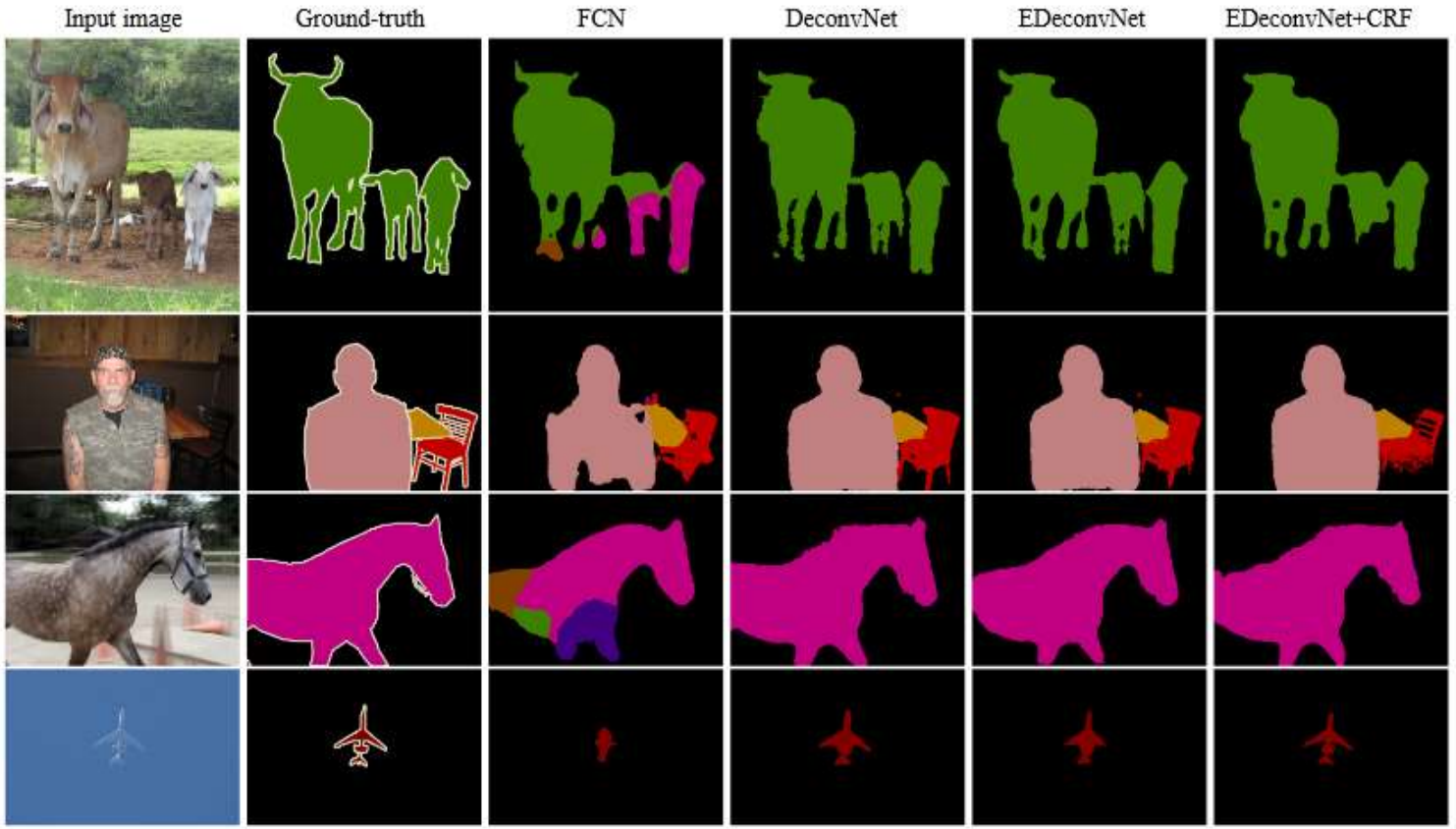
Deep learning architecturen

Deconvolutional networks - segmentation



Deep learning architecturen

Deconvolutional networks - segmentation



Problemen bij deep learning

- Convolutional neural networks zijn veelbelovend, zeker in combinatie met andere netwerkstructuren.

- Er zijn echter nog veel nadelen aan deze techniek verbonden:
  - Je moet een groot inputbeeld nog steeds doorkruisen via een set van overlappende patches waarna elke patch een classificatie krijgt.
  - Hierdoor is de memory footprint voor een enkel afbeelding gigantisch.
  - Zeker als je dit nog eens in een multi-scale context gaat bekijken.

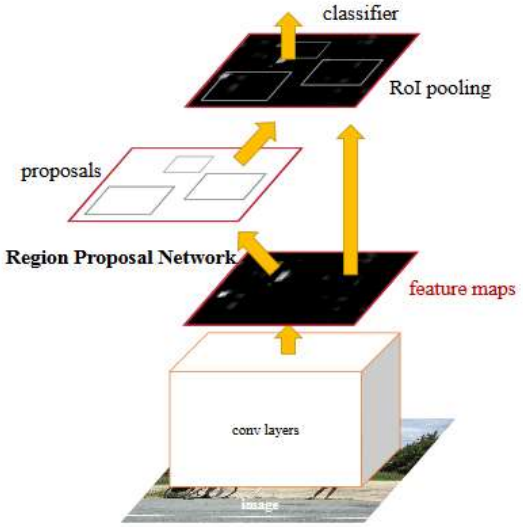
- Hierdoor lijken convolutional neural networks geen optie wanneer object detectie het doel van applicatie wordt of wanneer men real-time processing als basisvereiste heeft.

Region proposal networks

Om dit probleem aan te pakken, werden region proposal networks in het leven geroepen *(RCNN, Faster RCNN)*

Een region proposal network (RPN)

- Neemt een beeld als input en geeft een set van rechthoekige object proposals terug elk met hun eigen objectness score.
- Idee is gebaseerd op een FCN.
- Region proposal networks pikken in na de convolutional layers, en voor de fully connected layers, zodat de convolutie niet dubbel hoeft te gebeuren.

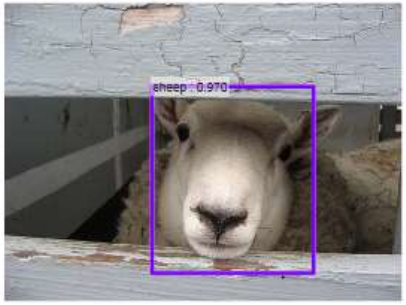


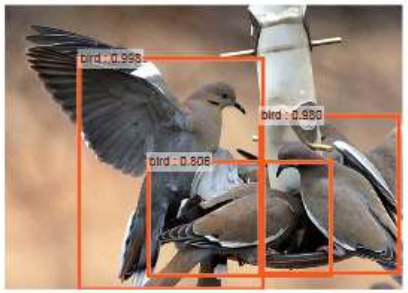
Region proposal networks

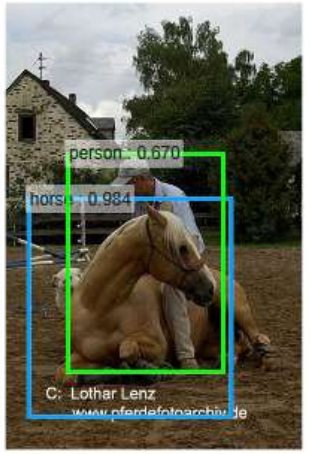
Gevolg is dat in plaats van meerdere duizenden tot miljoenen kandidaat windows geprocessed worden, er een selecte hoeveelheid region proposals door de CNN geduwt worden.

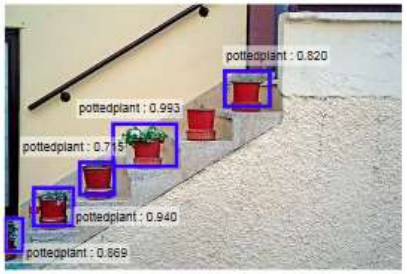
- Faster RCNN → 2000 best scoring region proposals.
- Onderzoek toont echter aan dat met +-300 proposals en een sterk netwerk, voldoende accuraadheid behaald kan worden.
- Het verlies in accuraatheid t.o.v. evalueren van alle regions is minimaal.

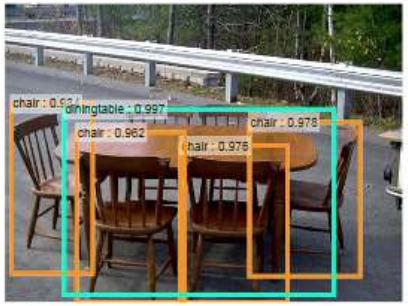
RPN + CNN for object detection

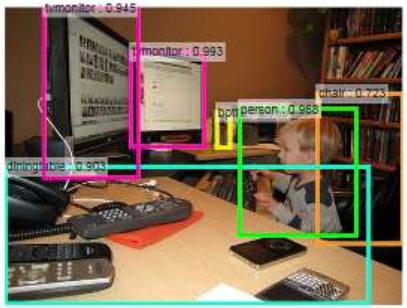


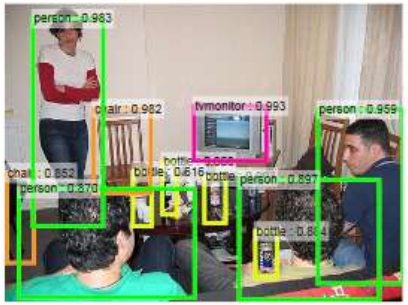












KU LEUVEN

Zelf aan de slag met deep learning

Om zelf aan de slag te gaan met deep learning

1. Goeie hardware die het mogelijk maakt om real time processing te doen.

2. Nood aan een softwarepakket dat een goeie interface aanlevert met de onderliggende deep learning bibliotheken.

3. Datasets / pretrained modellen om via transfer learning zelf een oplossing op te bouwen.

Deep learning hardware

NVIDIA domineert het deep learning landschap.

- Zo wat elk deep learning pakket maakt gebruik van NVIDIA software
  - CUDA: NVIDIA parallel computing platform voor GPU
  - CUFFT: Fast Fourier transform library voor GPU
  - CUDNN: NVIDIA deep learning primitives voor GPU
- De alternatieven in OpenCL zijn
  - Minder performant
  - Onbestaand
  - Minder actief ontwikkeld
- Dit zorgt ervoor dat b.v.b. AMD GPU's momenteel geen waardig alternatief zijn voor deep learning.

Deep learning hardware

"Wij doen geen deep learning wegens de hoge kost aan nieuwe hardware."

- Een veelgehoord argument bij industriele partners die in beeldverwerking / image processing aan de slag zijn.
- Tot 2015 was dit een geldig excuus, GPU setup's die krachtig genoeg waren om deep learning te doen, waren zeer duur.
- De dag van vandaag is dit argument echter niet meer geldig.

Deep learning hardware

NVIDIA voorziet verschillende lijnen, geschikt voor deep learning, maar zelfs de goedkopere GeForce lijn is een optie.

EDGE COMPUTING	HYPERSCALE HPC	VISUALIZATION	RESEARCHERS/ EARLY ADOPTERS
			
JETSON	TESLA	QUADRO	DGX-1

Deep learning hardware

Waar moet je rekening mee houden bij het aanschaffen van deep learning hardware?

- Op voorhand weten welke applicatie je wil uitwerken!
- Ga je aan de slag met object training?
- Of kies je enkel voor inferentie?

Inclusief model training

- Zorg voor minimaal 4 GB dedicated on board RAM

Enkel model inferentie

- Compacte modellen kunnen uitgevoerd worden op een GPU met 2GB

Algemeen

- Zorg voor een voldoende zware voeding! Die en je GPU en je andere hardware onderdelen van voldoende stroom kan voorzien.
- Zorg voor voldoende airflow in je toestel.
- Hoe meer CUDA cores, hoe sneller je kaart rekent!

Deep learning hardware

Interessante hardware indien je een compact embedded deep learning systeem wil opbouwen.

- Jetson TX1 en TX2
  - Supercomputer ter grootte van een bankkaart
  - Bevat 256 CUDA cores (Maxwell TX1, Pascal TX2)
  - 4GB (TX1) / 8GB (TX2) on board memory
- Vooral de developer kits zijn interessant!

TX1 /TX2
450
euro

Deep learning hardware

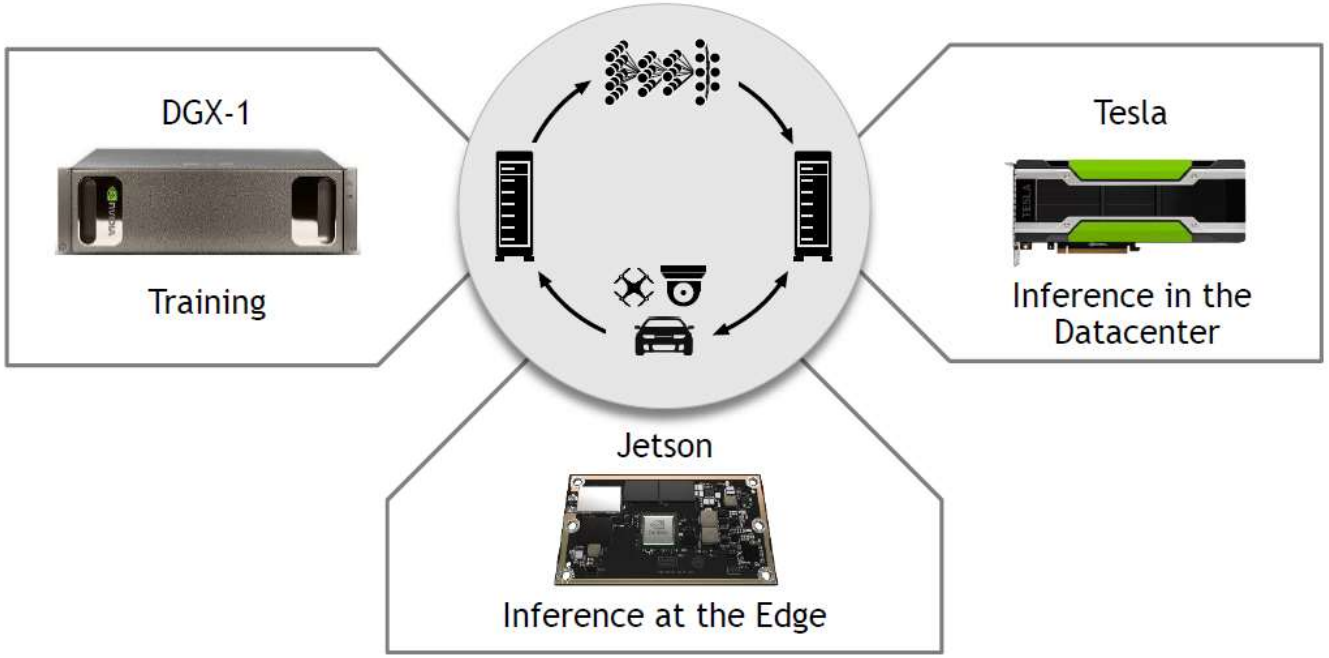
Interessante hardware indien je een desktop systeem hebt om testen op uit te voeren:

- NVIDIA Titan X (Pascal)
  - 1370 euro
  - 12GB dedicated memory
  - 3584 CUDA cores

- NVIDIA Gefore GTX 1080
  - 600-700 euro
  - 8GB dedicated memory
  - 2560 CUDA cores

Deep learning hardware

Eigenlijk stuurt NVIDIA aan op een interactie tussen hun verschillende structuren, om zo een multi-platform oplossing te bieden.



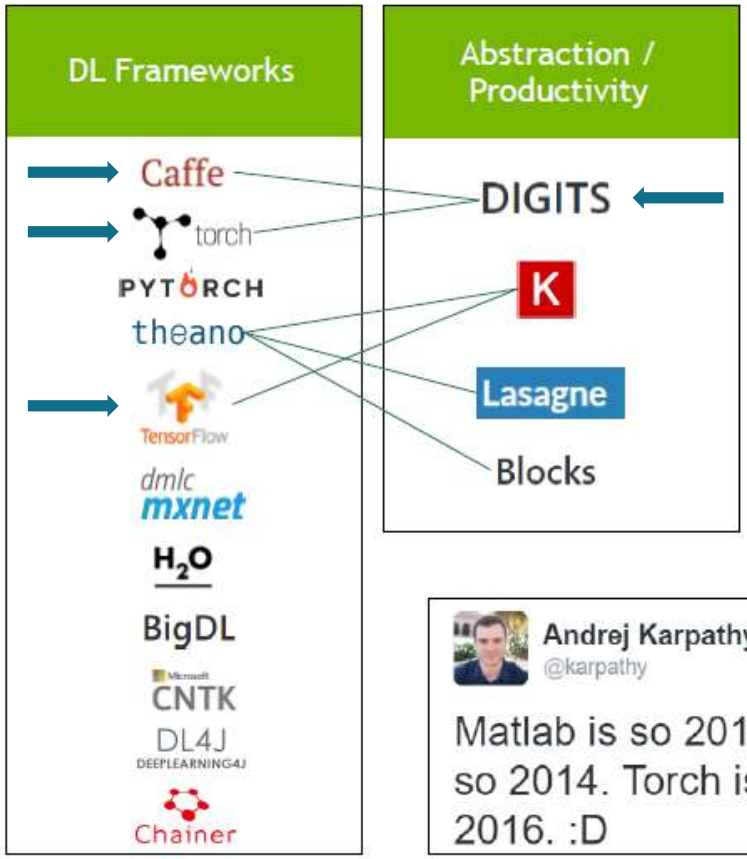
Deep learning software

Deep learning software is gelaagd opgebouwd:

developer.nvidia.com/deep-learning-software

Deep learning software

De frameworks laag is de laag die ons interesseert. Dit is namelijk de interface laag die je toelaat op een intuitieve manier met deep learning om te gaan.



DEEP LEARNING FRAMEWORKS

Deep learning software

Voordelen van een framework / abstraction layer

- Voldoende tutorials en documentatie beschikbaar.
- In de meeste gevallen een community waar je als user bij terecht kunt.
- Nadeel: zelfs abstracties blijken soms vrij complex wanneer je wilt afstappen van de standaard voorbeelden / toepassingen.

Daarnaast is er ook een veelbelovend pakket rond deep learning, dat een interface voorziet via C en CUDA, namelijk darknet.

- YOLO model → object detectie
- ImageNet model → object classificatie
- RNN model → time series analyse
- TinyYOLO → object detectie op embedded hardware

Deep learning via OpenCV

- OpenCV - https://github.com/opencv/opencv
  - Open source computer vision door Intel
  - Beeldverwerkingsbibliotheek die heel wat functionaliteit bevat.
  - Zorgt via wrappers (abstractie) een interface naar het bekende CAFFE pakket voor deep learning
  - Tiny-DNN - https://github.com/tiny-dnn/tiny-dnn
    - Een lightweight deep learning bibliotheek
    - Puur via header files, waardoor je niet moet compileren
    - Bevat momenteel enkel deep learning inferentie, via CPU
    - Belooft in toekomst deep learning inferentie en training, via GPU

Deep learning via OpenCV

- Snelste manier om deep learning op te starten in OpenCV
  - Zorg voor de laatste OpenCV 3.2 master branch
  - Zorg voor de contributed repository (https://github.com/opencv/opencv_contrib)
  - Zorg voor een werkende CAFFE installatie (https://github.com/BVLC/caffe/)
  - Zorg er voor dat bovenstaande pakketten met CUDA en CUDNN support gebuild zijn

- Via de Caffe model zoo download je het model waarmee je deep learning wil doen (segmentatie/detectie/classificatie/…)

Deep learning via OpenCV

- Via de voorbeeld samples, krijg je een zicht op hoe de deep learning modellen aan te spreken

  (https://github.com/opencv/opencv_contrib/tree/master/modules/dnn/samples)

```
int main(int argc, char **argv)
{
    cv::dnn::initModule();

    String modelTxt = "bvlc_googlenet.prototxt";
    String modelBin = "bvlc_googlenet.caffemodel";
    String imageFile = (argc > 1) ? argv[1] : "space_shuttle.jpg";

    Net net = dnn::readNetFromCaffe(modelTxt, modelBin);

    Mat img = imread(imageFile);
    resize(img, img, Size(224, 224));
    Mat inputBlob = blobFromImage(img);

    net.setBlob(".data", inputBlob);          //set the network input
    net.forward();                            //compute output
    Mat prob = net.getBlob("prob");           //gather output of "prob" layer

    int classId;
    double classProb;
    getMaxClass(prob, &classId, &classProb);  //find the best class

    std::vector<String> classNames = readClassNames();
    std::cout << "Best class: #" << classId << " '" << classNames.at(classId) << "'" << std::endl;
    std::cout << "Probability: " << classProb * 100 << "%" << std::endl;

    return 0;
} //main
```

Meest gebruikte architecturen/modellen

Binnen de deep learning community zijn een aantal modellen afkomstig van onderzoeksinstellingen toonaangevend en common practice.

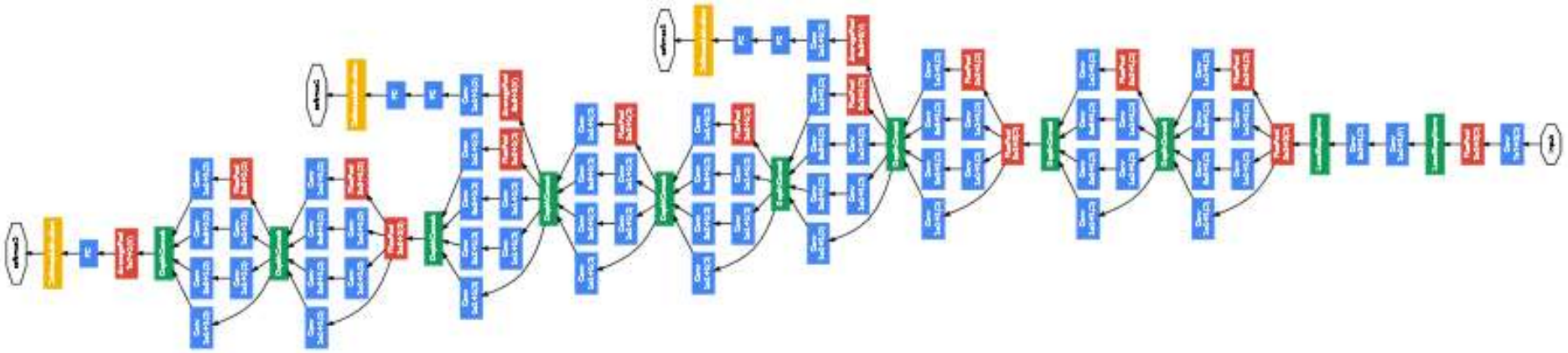
- LeNet: 5 layer CNN ontwikkeld voor digit recognition
- AlexNet: een stacked 9 stage CNN voor ImageNET classificatie
- GoogLeNet: going even deeper – 22 layer deep CNN, of which 9 inception modules, leading to over 100 layers in total
- Oxford group: VGG models – large scale visual recognition

  VGG16, VGG19, VGGfusion

De trend was om steeds dieper te gaan, meer complexiteit toe te voegen, om dan uiteindelijk een zeer hoog performant netwerk te bekomen, dat zeer rekenintensief is en bijna onbruikbaar.

Meest gebruikte architecturen/modellen

GoogLeNet architectuur



Deep learning – memory footprint

- Doordat de meeste beschikbare open source modellen vrij complex en vrij diep zijn (veel opeenvolgende layers), zijn de resulterende weight files heel groot.
  - Configuratie files (opbouw layers), zijn compact.
  - De weight files daarentegen zijn gigantisch (VGG 350 MB)

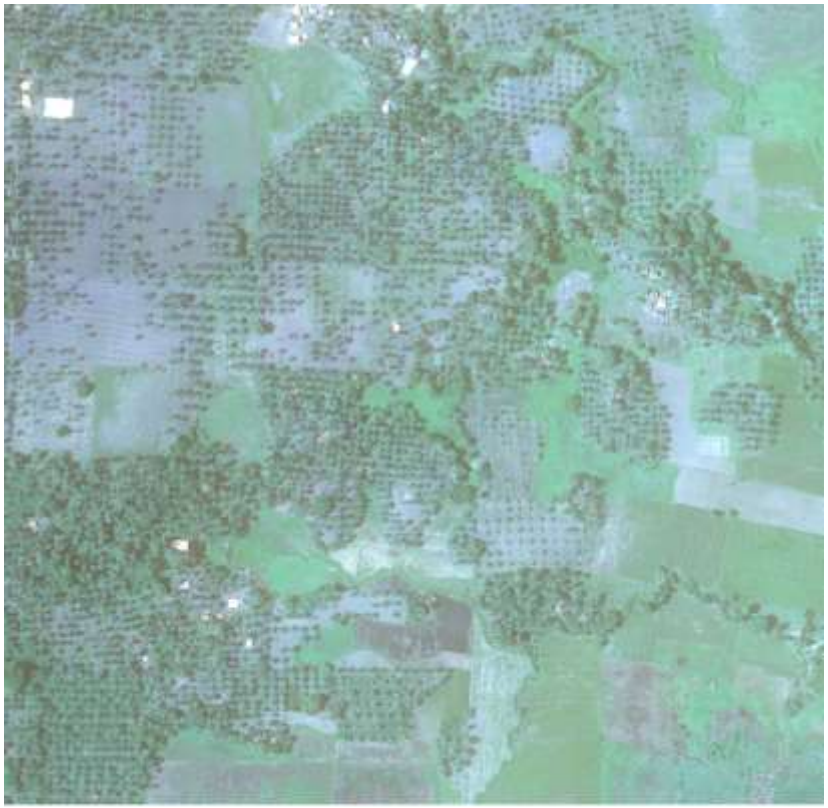
- Uiteraard is dit problematisch op embedded systemen. Daarom werden verscheidene technieken uitgevonden om data van krachtige diepe modellen te comprimeren.

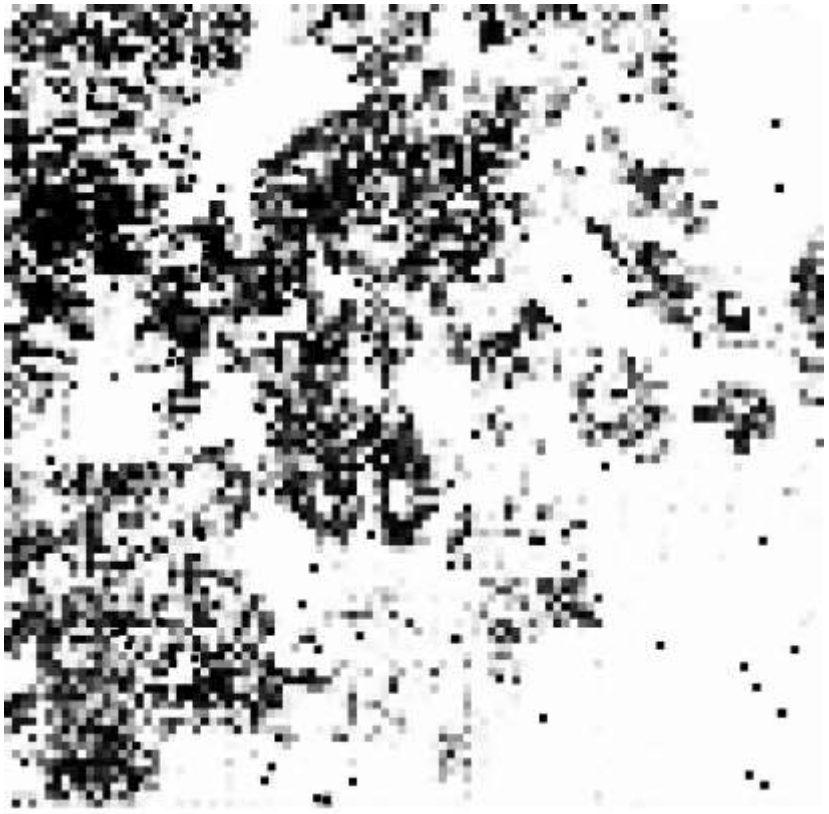
Deep learning – memory footprint

- Voorgestelde oplossingen voor model compressie
  - Model compressie via model hashing: encodering van model
  - Toepassen van singular value decomposition op CNN models
  - Network pruning: wegnemen van weights onder een threshold, op 0 zetten om tot een sparse matrix representatie te komen
  - Toevoegen van quantizatie aan het network pruning
  - Deep compression: gebruik van huffman encoding in deep networks
  - SqueezeNet:
    - Vervangen van convolution filters met kleiner equivalent
    - Verminderen van het aantal inputkanalen
    - Downsamplen pas laat in het netwerk, zodanig dat we grote activatiemapping krijgen in het netwerk in de convolution layers.
- This all lead to a tremendous decrease in size, resulting in models between 4 – 10MB, opening deep learning for embedded hardware!

Deep learning examples

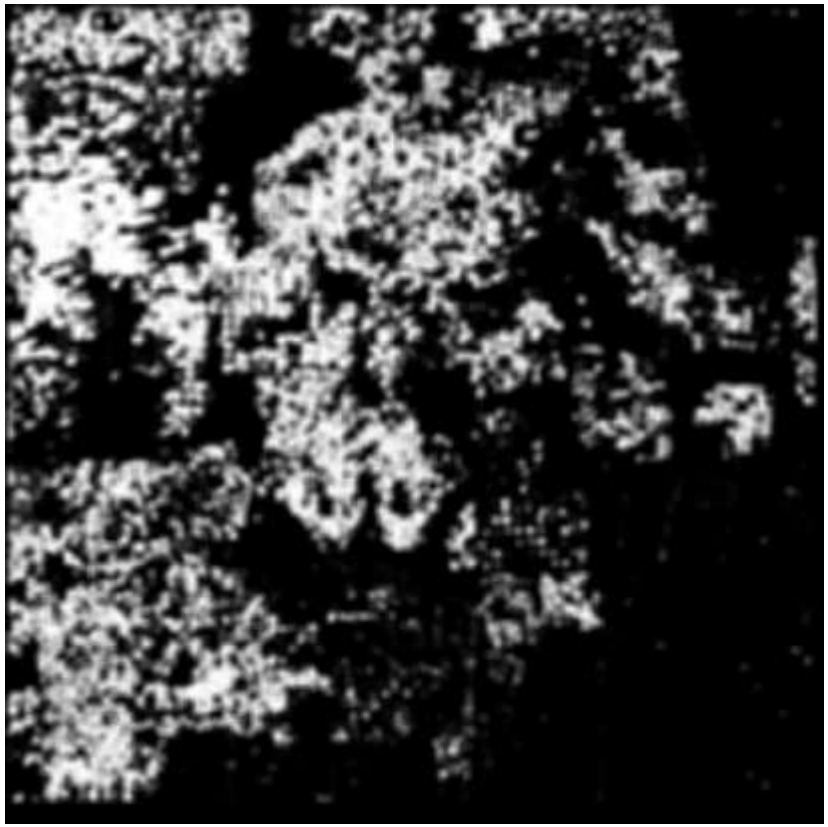
Transfer learning using Google's Inception v3 model on only 35 class and non object class samples (no overlapping windows)

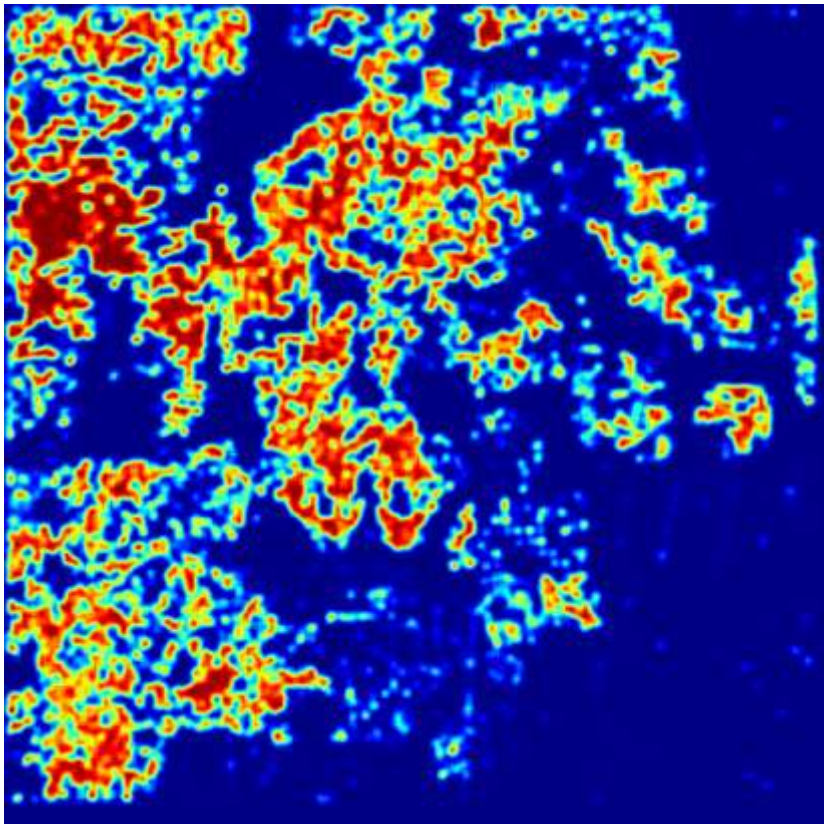




Deep learning examples

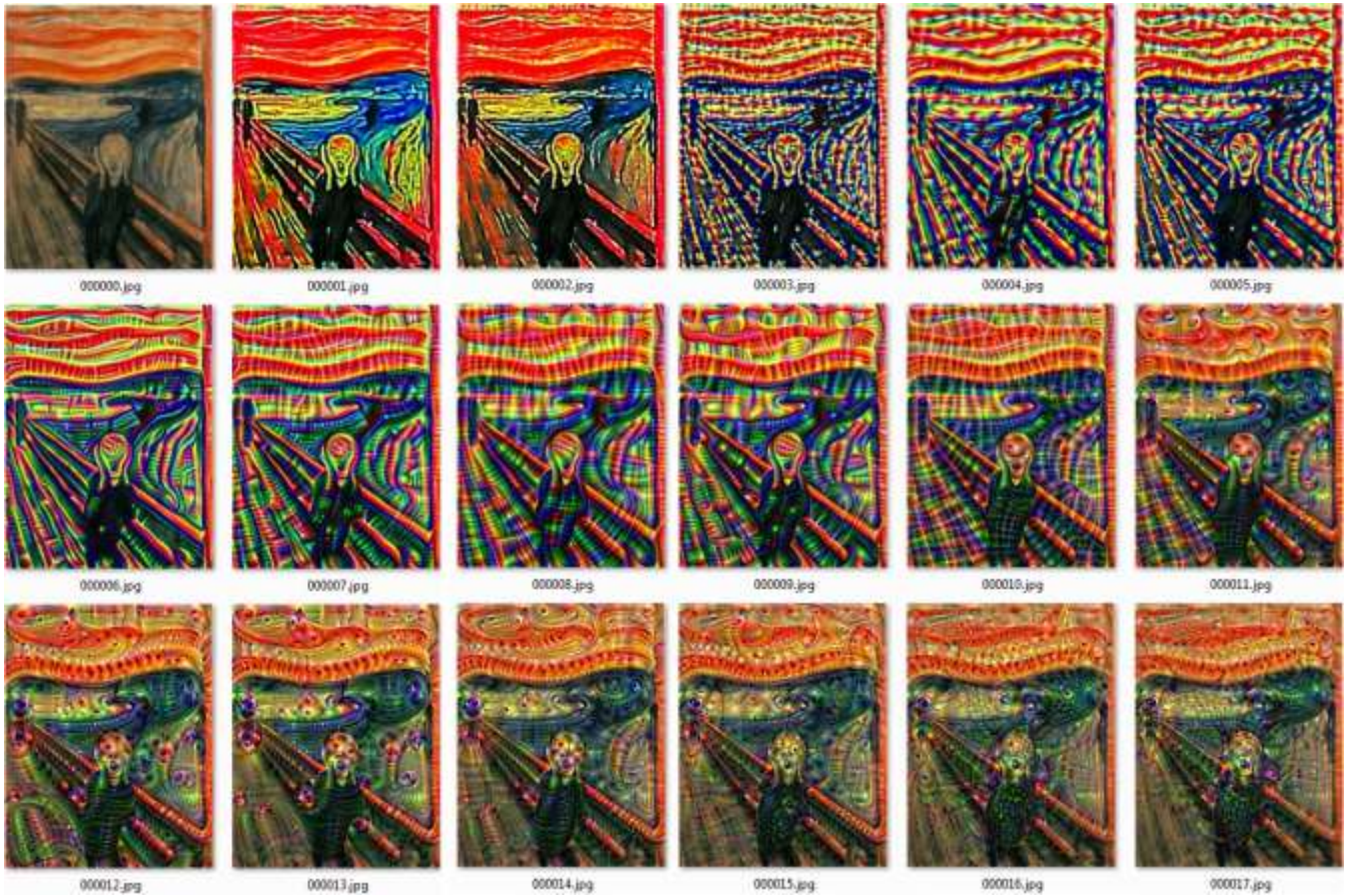
Transfer learning using Google's Inception v3 model on only 35 class and non object class samples (overlapping windows)





Deep learning examples

Deconvolution model op een schilderij, wat ziet het model intern?

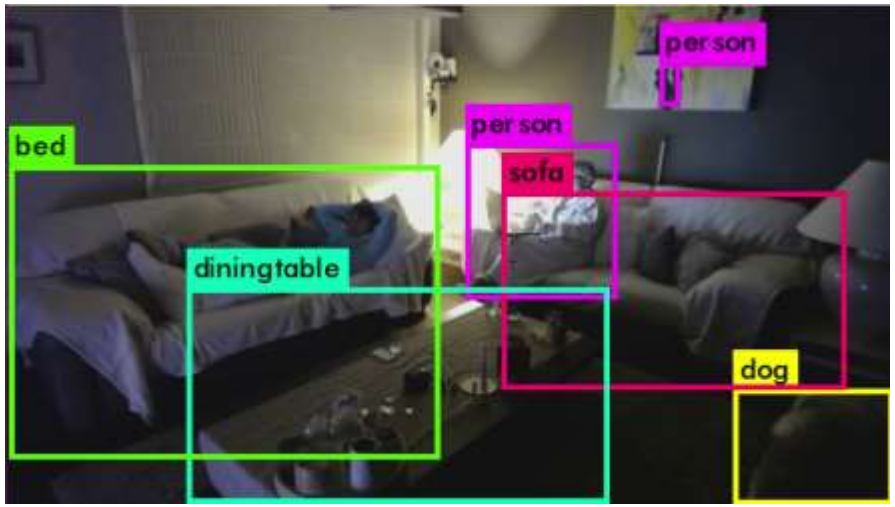


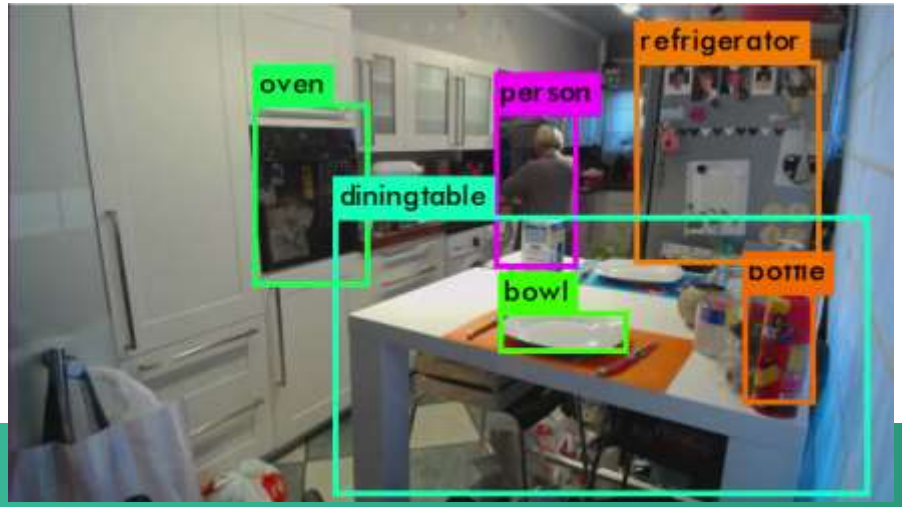
Deep learning examples

Deconvolution model op een veranda, wat ziet het model intern?

Deep learning examples

YOLOv2 object detectie in uitdagende situaties (belichting, clutter, camerastandpunten, …) waar andere detectoren falen.



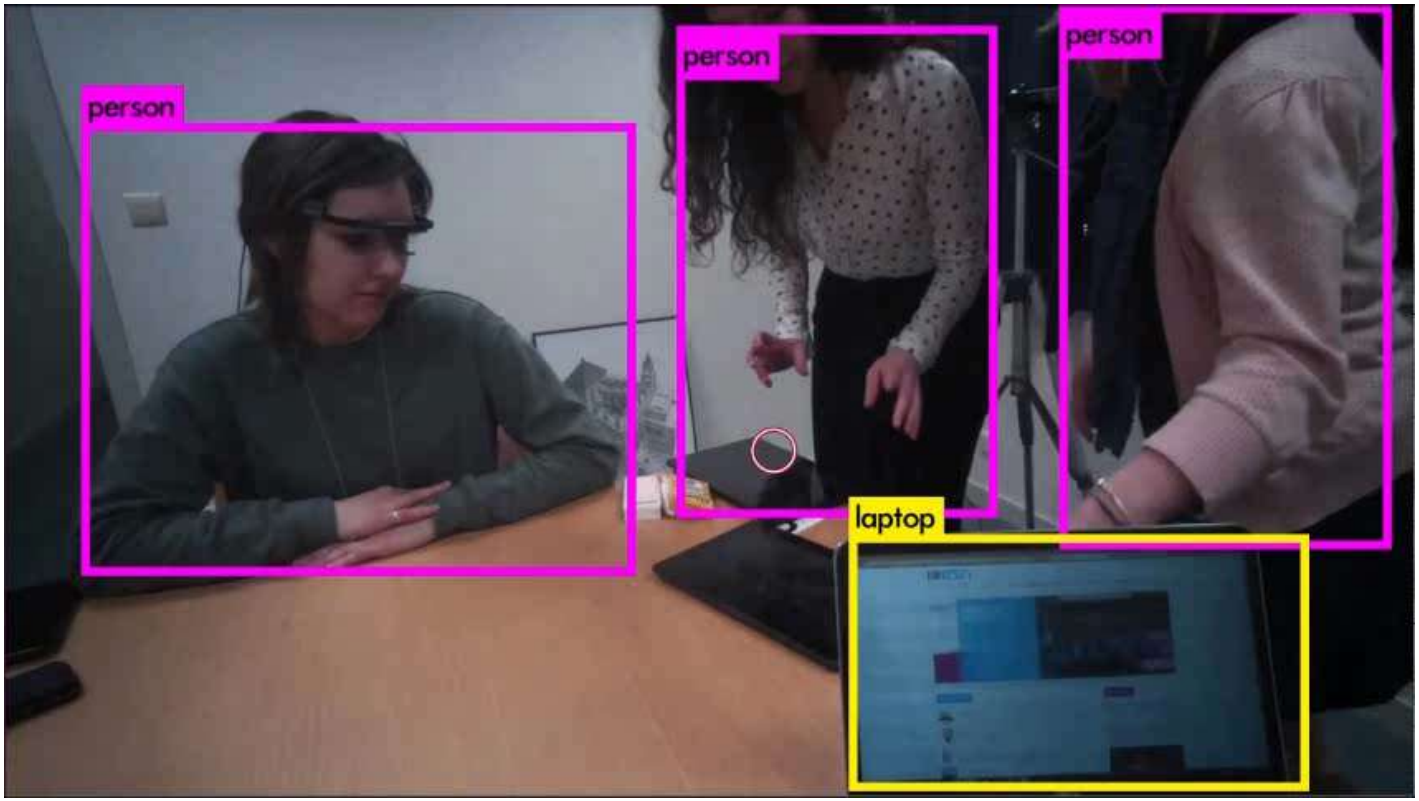


Deep learning examples

YOLOv2 object detectie in een verkeersanalyse toepassing.

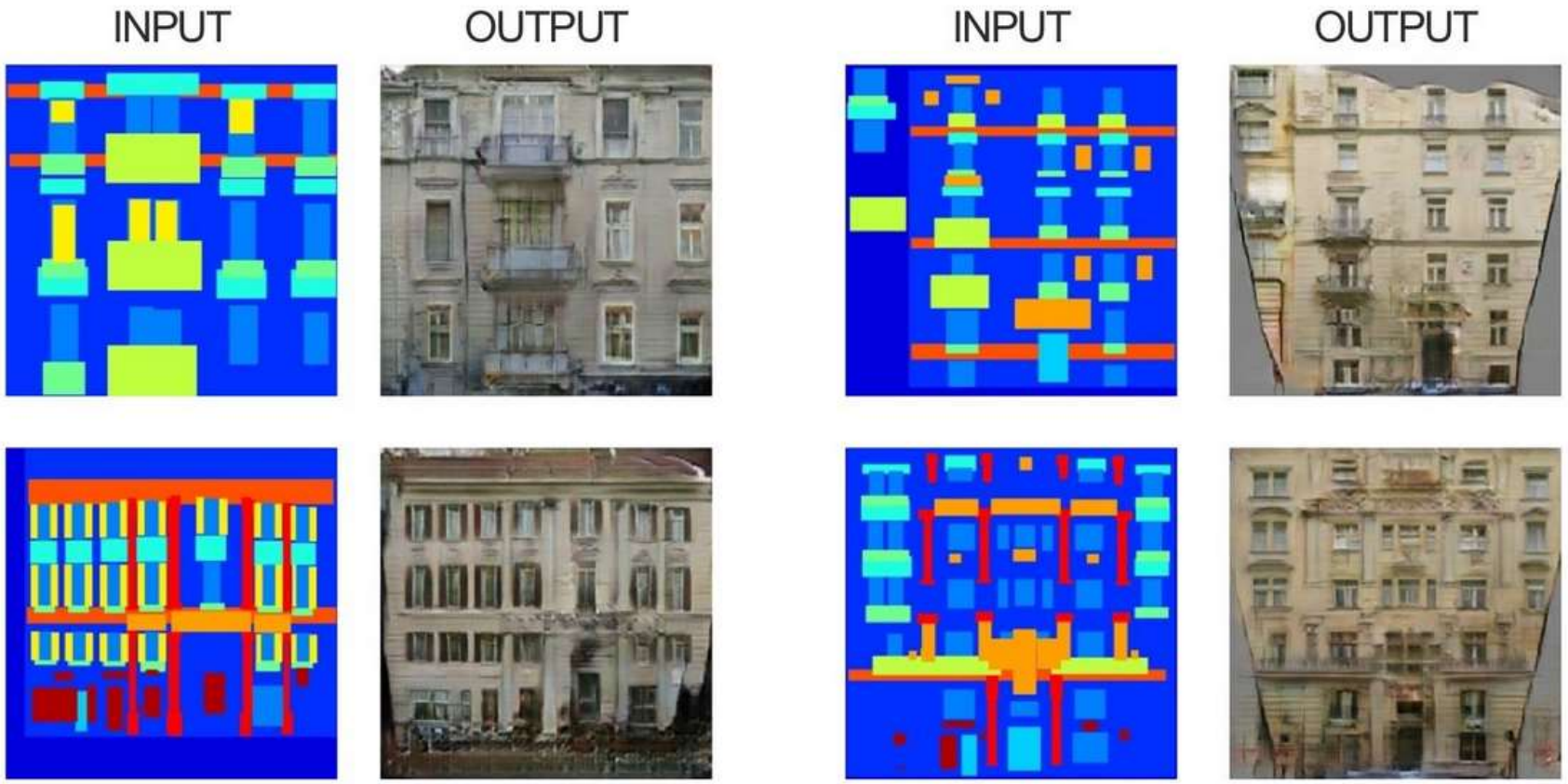
Deep learning examples

YOLOv2 object detectie in een eyetracker opname, detecteren van personen in elke mogelijke opstelling.



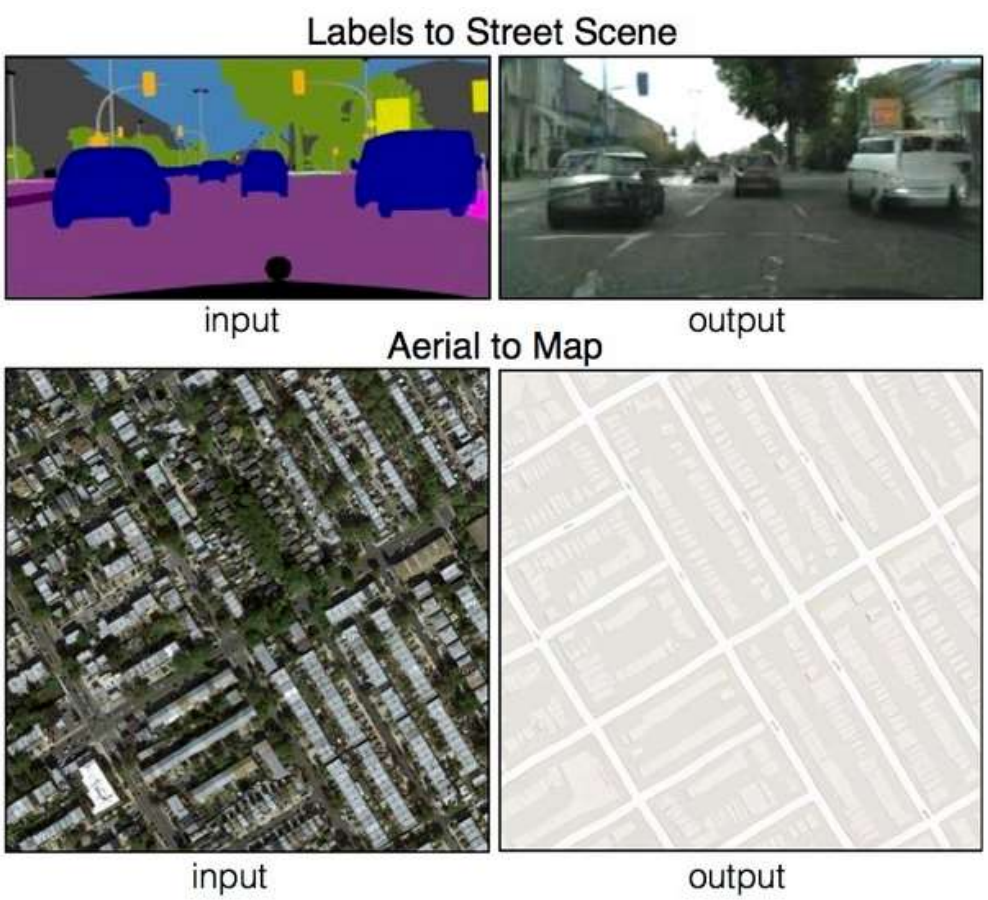
Laatste nieuwtjes in deep learning

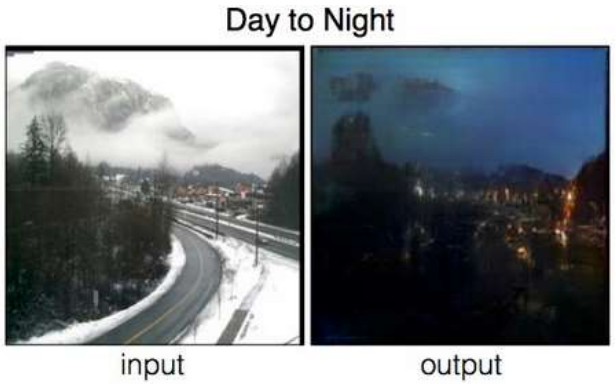
GAN = Generative Adversarial Nets - bvb. pix2pix

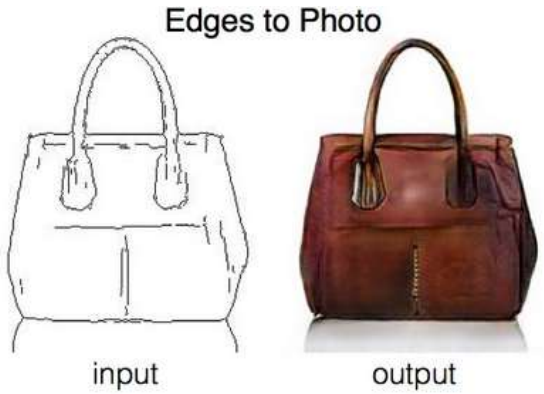


Laatste nieuwtjes in deep learning

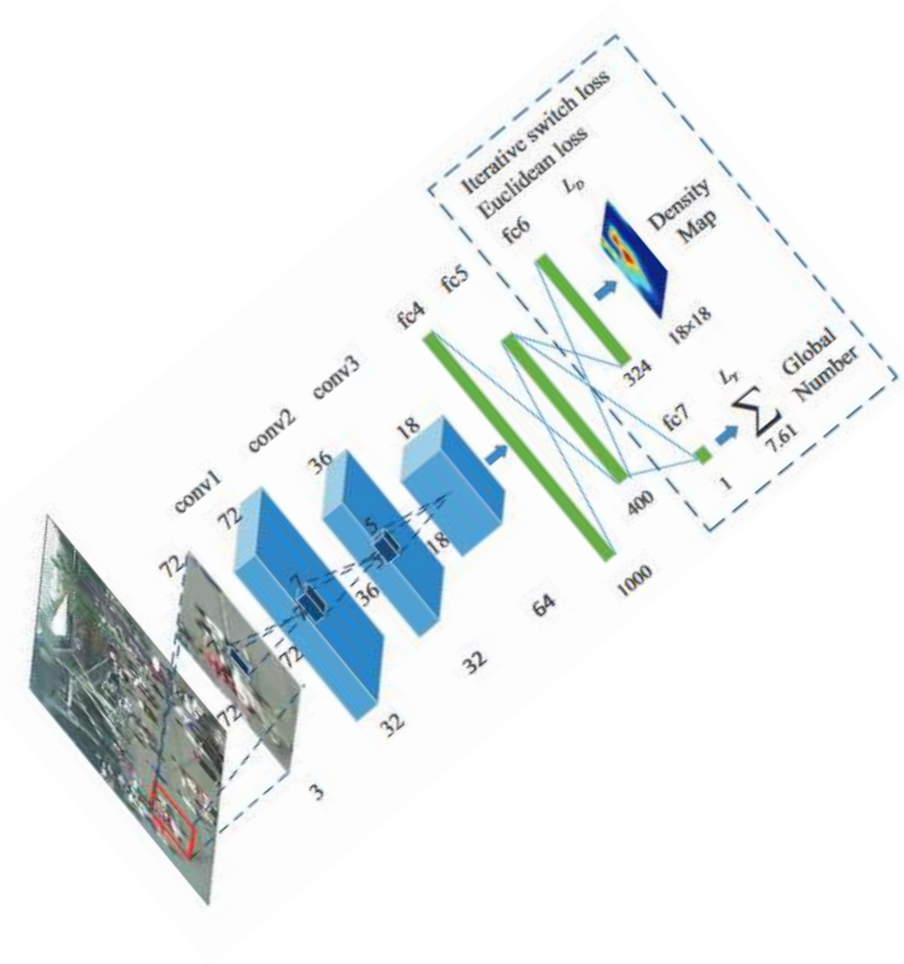
GAN = Generative Adversarial Nets - bvb. pix2pix

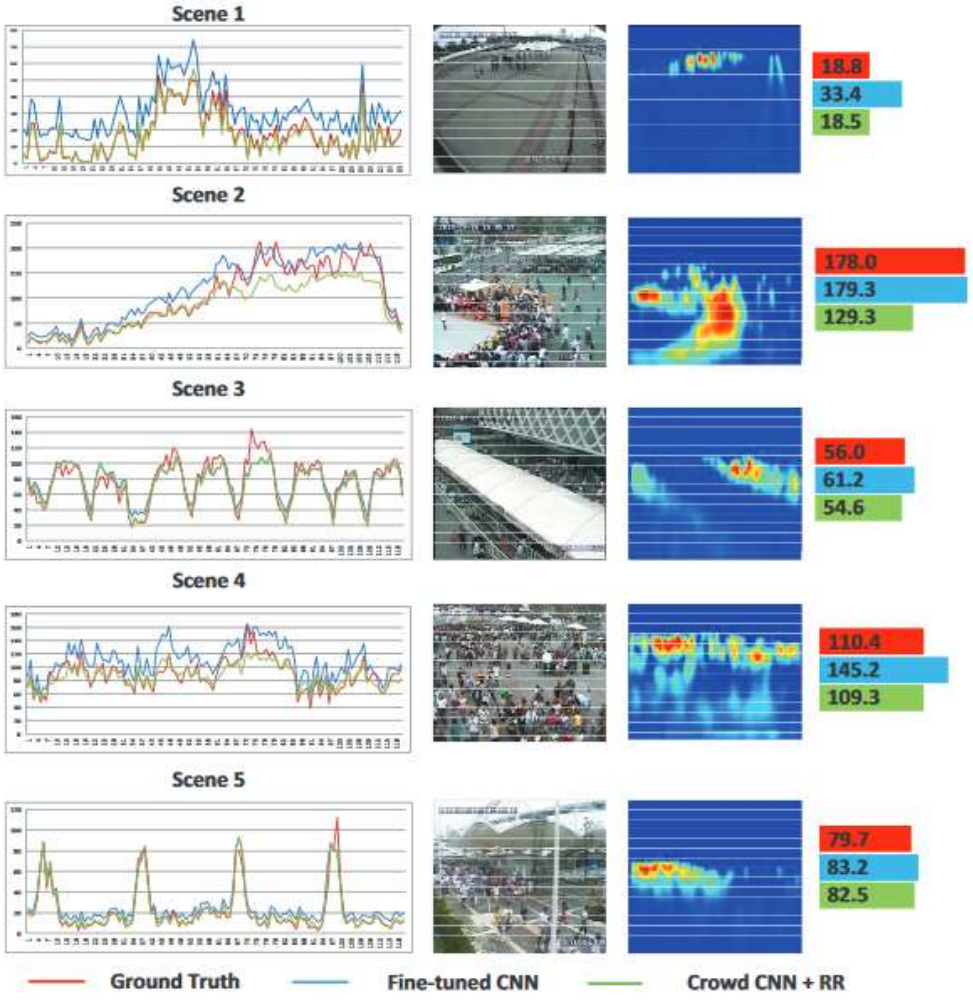






Integratie van convolutional outputs voor crowd counting / analysis





KU LEUVEN

Bedankt voor uw aandacht!

Meer info:

- http://www.eavise.be
- stevenputtemans.github.io
- steven.puttemans@kuleuven.be

KU LEUVEN