***Neural Networks***

**Week 1:**

Discussion topics:

* Wat is Amazon Mechanical Turk, om je data te laten labelen?
* Wat is k-fold cross-validation, en hoe moet je dit zien in de context van deep learning?
* Bespreek de “goodness” van een model tijdens training, en tijdens test fase. Voor zowel classification als regression.
* Wat is een epoch, en een batch (vooral: wanneer doe je updating van je learnable parameters)?
* Bespreek bias - variance, en wat dat te maken heeft met # of features (bias is meestal redelijk goed te begrijpen, maar focus op variance!).
* Wat is data imputation?
* Bespreek normalization versus standardization, en wat is het effect hiervan op data met outliers?

**Week 2:**

Discussion topics:

* Een Neural Net is een hiërarchisch model. Leg uit.
* Wat is een affine transformation (in de context van een perceptron)?
* Om een multilayer perceptron, met minstens 1 hidden layer, te laten werken als universal approximator, heb je non-linear activation functions nodig. Verklaar.
* Leg de ReLu activation function uit, en waarom je soms een leaky ReLu, of ELU nodig hebt.
* Leg de loss function uit in de context van een regression problem, en in de context van een classification problem.
* Leg uit waarom de gradiënt van de loss function belangrijk is.
* Bij backpropagation gaan we de gradiënt van de loss function laten terugwerken naar het begin, via de chain rule. Daarom hebben we ook de gradients van de activation functions nodig(niet echt die van affine linear transforms). Leg in deze context het probleem van vanishing gradients uit (zie ook vraag over ReLu activation function).

**Week 3:**

Discussion topics:

* Leg uit wat stochastic gradient descent is (tov gewone ‘batch’ gradient descent), en wat het begrip ‘mini-batch’ in deze context betekent.
* Leg uit wat een goede learning rate is, en wat simulated annealing betekent.
* Leg de One Cycle learning rate uit.
* Leg de verschillende soorten regularisaties uit: L1 of L2 weight decay, early stopping, dropout, ensemble technieken, data augmentation en batch normalization.
* Leg verschillende soorten optimizations uit: batch normalization, momentum technieken (RMSProp, Adam), initializers (Xavier).

**Week 4:**

Discussion topics:

* Leg uit welke pre-processing technieken meestal nodig zijn om met images te kunnen werken in een CNN.
* Waarom gebruiken we geen DNN om computer vision te doen op images?
* Leg uit wat een convolution is (in de context van CNNs), en gebruik hiervoor de termen ‘kernel’, ‘locale receptive field’, ‘stride’ en ‘feature map’.
* Leg uit wat een 2D convolution bewerking is met een 3D kernel op een RGB image.
* Leg uit waarom we padding gebruiken bij convolution bewerkingen op images, en geef enkele voorbeelden van padding.
* Waarom is het belangrijk dat je de output\_size van een convolution bewerking moet weten?
* Leg uit waarvoor we pooling technieken gebruiken, en geeff enkele voorbeelden.
* Leg een typische CNN architectuur uit voor image classification (convolutions, poolings, activation functions, normalisation layers, flattening layer, … ).

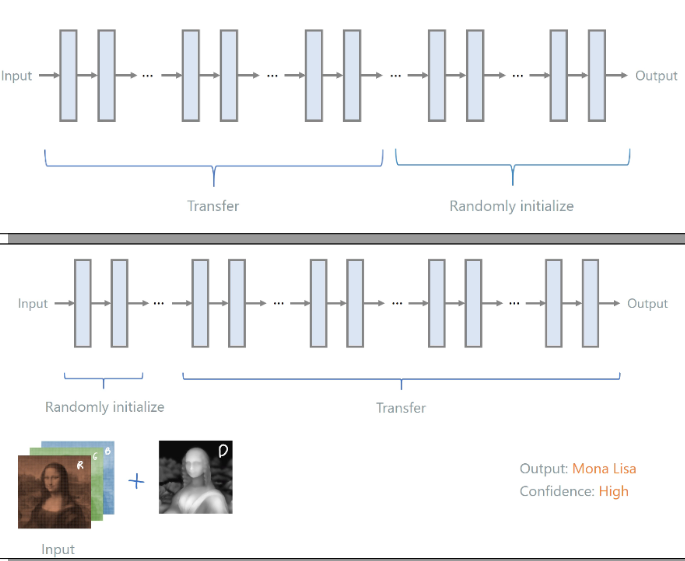
**Week 5:**

Discussion topics:

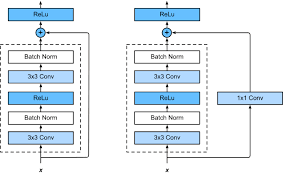
* Leg uit waarom deep learning data hungry is, maar dat transfer learning daar een mogelijke oplossing voor is. Geef hierbij ook aan welke limitaties transfer learning heeft.
  + Deep learning mist momenteel een mechanisme voor het leren van abstracties door expliciete, verbale definitie, en werkt het beste wanneer er duizenden, miljoenen of zelfs miljarden trainingsvoorbeelden zijn. Transfer learning gebruikt zijn kennis van een geleerde taak om een soortgelijke taak op dezelfde manier op dezelfde manier op te lossen (minder data en compute). Wanneer er een te groot verschil is tussen de architecturen van de 2 models, kunnen we moeilijk de knowledge, de weights, zomaar gaan transfereren (limitation).



* Leg verschillende mogelijke strategieën (en hun context) uit om transfer learning toe te passen.



* De bovenste keuze hier is de meest gebruikte manier, maar de onderste manier wordt gebruik als je gelijkaardige input hebt die toch een klein beetje verschilt.
* De bovenste is de meest gebruikte om de main features te hergebruiken, terwijl de onderste kan werken als je gelijkaardige input hebt die een klein beetje anders is. De eerste lagen moeten dan eerst leren die input wat te projecteren in andere dimensies en kan daarna min of meer hetzelfde doen zoals bij het eerste netwerk.
* Wat is ImageNet, en waarom is dit zo belangrijk geweest voor deep learning?
  + ImageNet is een grote database die correct gelabeld is met 1000 categorieën en 1,2 miljoen training images en een test set van 100 duizend images. Dit is belangrijk geweest omdat dit enorm veel werk was en mensen hebben hier gebruik van gemaakt en is daardoor een heel grote boost door gekomen op het gebied van image recognition.
* Leg uit hoe de 1x1 convolution (identity) een optimalisatie is in de compute van CNNs.
  + Door een 1x1 te gebruiken zorg je ervoor dat er minder dimensies zijn en dus ook minder compute kracht nodig is om dit te berekenen. Dit lijkt niet logisch maar als je de wiskunde bekijkt zie je dat je bij een 1x1 veel minder berekeningen moet doen dan bij een 5x5 bijvoorbeeld.
* Leg de verschillende innovatieve verbeteringen uit van AlexNet, VGG, Inception (GoogLeNet), ResNet, DenseNet.
  + AlexNet: stacking of conv layers (no pooling in between), local response normalization (sterk geactiveerde neurons gaan tegen dichtbijzijnde neurons zeggen dat ze moeten zwijgen zodat neurons vechten om te specialiseren in verschillende features). Om overfitting te voorkomen is er een heel hoge drop out rate van 50% zodat niet 1 neuron instaat voor een bepaalde feature, maar een paar neurons hiervan. Randomly shifting training images by offsets helpt ook tegen de overfitting.
  + VGG: in plaats van een grotere filter te nemen neemt VGG meerdere kleinere filters na elkaar zonder een pooling layer om zo exponentieel minder parameters moet trainen en dus ook minder dimensies gaat hebben. Daarom heb je dus minder compute kracht nodig.
  + Inception (GoogLeNet): bij kleinere kernels is de detectie van info meer lokaal en bij grotere kernels meer globaal, daarom neemt het een mix van de twee. Diepere networks hebben meer de neiging om te gaan overfitten dus gaan we hier breder en dieper, het stacken van conv layers is wel heel compute intensive.
  + ResNet: extreem diep netwerk (152 layers), skipped connecties dit forceert het model om de rest te leren, wat de andere lagen niet doen. We kunnen dieper gaan met dit network omdat connecties skippen het makkelijker maakt for the gradient to backprop.



* DenseNet: bestaat uit dense blokken die fully connected zijn, hier ‘skippen’ connecties naar elke sublaag van het blok: BatchNorm layer, ReLu activation, Conv2D. Een voordeel hiervan is dat je minder parameters hebt om te trainen en je kan veel layers droppen. Elke layer heeft toegang tot de gradients van de loss function en de originele input image.

