***Neural Networks***

**Week 1 recap:**

* Wat is Amazon Mechanical Turk, om je data te laten labelen?

*Amazon Mechanical Turk is een crowdsourcing marketplace waar bedrijven hun taken kunnen outsourcen. In de context van data te labelen betekent dit dat een bedrijf hier mensen betaald om hun data te labelen (bv dat de worker moet zeggen of hij een foto van een hond of een kat ziet).*

* Wat is k-fold cross-validation, en hoe moet je dit zien in de context van deep learning?

K-fold cross-validation is een resampling procedure die gebruikt wordt om een model te evalueren op een gelimiteerde data sample. De parameter K verwijst naar het aantal groepen waarin een data sample gesplit zou moeten worden.

Cross-validation wordt vooral gebruikt om te schatten hoe goed een model is op data die het model nog nooit gezien heeft.

* Bespreek de “goodness” van een model tijdens training, en tijdens testfase. Voor zowel classification als regression.

Tijdens training: wat voor type probleem? Regressie of classificatie?

Classificatie:

wiskundige berekening op logaritmen (standaard voorgekozen)

Regressie:

mean-squared error (hoever is men data punt verwijderd van men voorspelling)

Tijdens test: zware berekening (éénmalig), bij classificatie (confusion matrix), bij regressie (R-squared)

* Wat is een epoch, en een batch (vooral: wanneer doe je updating van je learnable parameters)?

Epoch:

Één iteratie over het deel meegegeven dataset.

Batch:

Een deel van de dataset dat door het neurale netwerk gestuurd wordt.

Het updaten van de learnable parameters doen we na elke batch update in het netwerk.

* Bespreek bias - variance, en wat dat te maken heeft met # of features (bias is meestal redelijk goed te begrijpen, maar focus op variance!).

Bias maakt assumptions van de data voor variance errors tegen te gaan (bias-variance tradeoff).

Variance neemt alle features (zelfs de onbelangrijke).

Bians generaliseerd features (te weinig features).

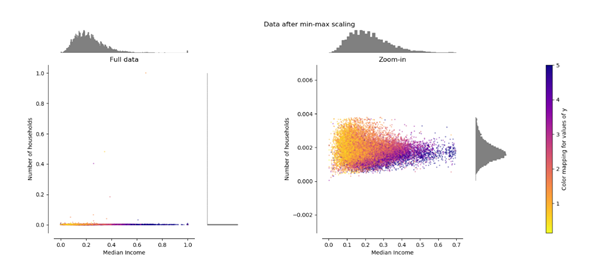
Balans tussen deze 2 proberen te zoeken (overfit ↔ underfit).

* Wat is data imputation?

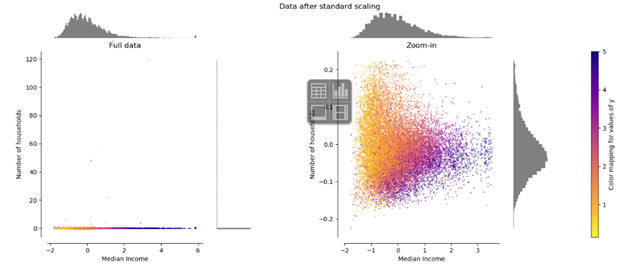
Missing data vervangen en opvullen met assumptions rekening houdend met andere features

* Bespreek normalization versus standardization, en wat is het effect hiervan op data met outliers?

Normalization:



Standardization:



**Week 2 neural network basics:**

* 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:**

* 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:**

* Leg uit welke pre-processing technieken meestal nodig zijn om met images te kunnen werken in een CNN.

Uniform aspect ratio:

(Zo goed als alle libraries gebruiken vierkante images (gemakkelijk om te roteren, transpose,…))

- crop images to be square

- center of image most important

Uniform image size:

(Alle images in een batch (zelfs over alle batches heen, niet alleen in een ‘mini’-batch, maar in de ganse batch) moeten dezelfde afmetingen hebben

Truck: meestal gaat men de images zelfs eerst verkleinen, model snel trainen, en dan transfer learning techniek gebruiken om te starten met trained netwerk, images terug te vergroten, en dan fine tuning te doen)

- fit image size to CNN feature maps (pixels to convolutional layer)

- up-scaling and down-scaling

Mean and perturbed images:

(Dit is een pure ‘insight’ techniek, om feeling te krijgen voor de images. De meeste conv.nets doen geen specifieke preprocessing op mean, of normalisation niveau. Toch niet bij de inputs)

- average pixel across training dataset

Normalized image inputs:

(Scaling doet men wel regelmatig (0-1 range) (hoewel dit niet niet altijd nodig is), maar normalisation doet men meestal in een ‘batch normalize’ layer, na de convolution layer, en voor de ReLU activation. De convolution neurons hebben immers geen echte activation function, dus ook geen nood aan normalizing…)

- scaling en uitschieters wegdoen

Dimensionality reduction:

- reduce the 3 rgb channels to grayscale for just one channel (reduces dimensionality of all image tensors and the size of the problem so training completes faster)

Data augmentation:

- foto’s inzoomen, draaien, spiegelen voor meer data en overfitten te verminderen en het zo meer robuust te maken

* Waarom gebruiken we geen DNN om computer vision te doen op images?

DNN is te complex en fully connected met veel layers en gebruikt veel meer neuronen. CNN is meer shallow en traint ook sneller.

* Leg uit wat een convolution is (in de context van CNNs), en gebruik hiervoor de termen ‘kernel’, ‘locale receptive field’, ‘stride’ en ‘feature map’.

Kernel: een matrix met gewichten die toegepast wordt op de local receptive field

Local receptive field: deel van de image waar momenteel naar gekeken wordt

Stride: afstand dat uw local receptive field verschuift

Feature map: het resultaat van al de kernel berekeningen

* Leg uit wat een 2D convolution bewerking is met een 3D kernel op een RGB image.

Kleur images, met meerdere channels, vragen wel voor een 3D (of meerdere dimensies) kernel, maar het is een 2D convolution: we bewegen de kernel in een 2D vlak over de image (wel met meerdere channels tegelijkertijd). Output is ook een 2D matrix.

* Leg uit waarom we padding gebruiken bij convolution bewerkingen op images, en geef enkele voorbeelden van padding.

Bij CNN kan het zijn dat de randen van de image niet zo belangrijk worden dus kan er een lege border rond worden gezet zodat deze ‘genegeerd’ wordt ipv de image zelf.

Er kan ook padding gebruikt worden als met de strides het local receptive field niet past.

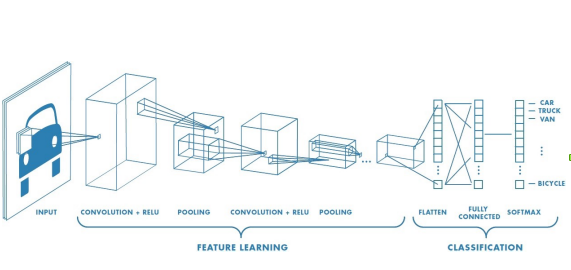
* Waarom is het belangrijk dat je de output\_size van een convolution bewerking moet weten?

Principe is niet zo belangrijk in praktijk, omdat de meeste frameworks zoals keras en fastai dit zelf opvangen maar bij tensorflow of fastai is dit wel belangrijk omdat je moet kunnen doorgeven aan je volgende layer hoe groot deze moet zijn, dit is vooral belangrijk als je van je CNN naar je fully connected gaat gaan omdat je moet weten hoe groot het gaat worden met je flattening layer.

* Leg uit waarvoor we pooling technieken gebruiken, en geef enkele voorbeelden.

Reduce the parameters of the image, when your image is too big. (max, average, sum pooling). Reduce memory usage, overfitten tegengaan, make NN recognize features independent of location.

* Leg een typische CNN architectuur uit voor image classification (convolutions, poolings, activation functions, normalisation layers, flattening layer, … ).

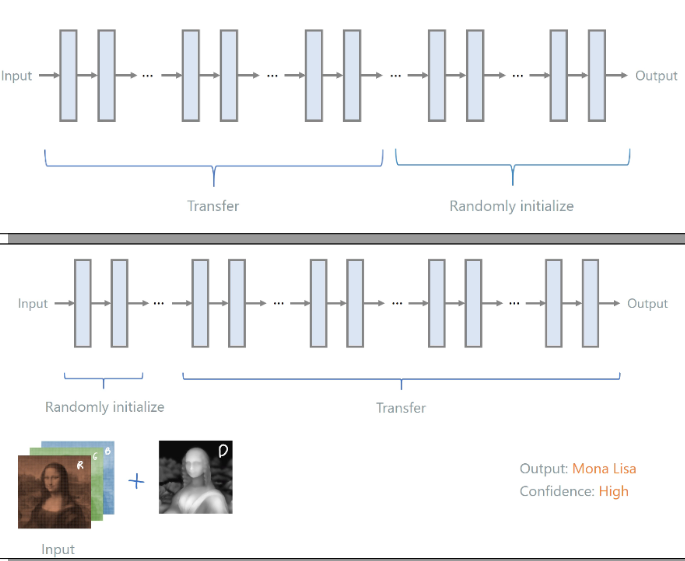


**Week 5:**

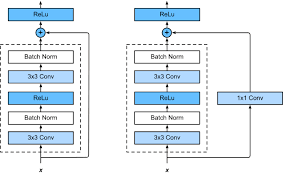
* Leg uit waarom deep learning data hungry is, maar dat transfer learning daar een mogelijke oplossing voor is. Geef hierbij ook aan welke limitaties transfer learning heeft.
  + Deep learning mist momenteel een mechanisme voor het leren van abstracties door expliciete, verbale definitie, en werkt het 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).



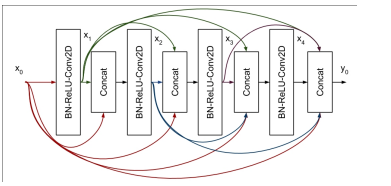
* Leg verschillende mogelijke strategieën (en hun context) uit om transfer learning toe te passen.



* 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 dichtstbijzijnde 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: Batch Norm 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.



**Week 6:**

* Leg uit hoe, en waarom, we Active Learning gebruiken om data sneller te labelen (bv via een Machine Learning-human-in-the-loop-approach).
* Leg uit waarom het soms aangewezen is om, eerder dan voor een grote hoeveelheid labelled data, voor meer quality (tov quantity) van labelled data te gaan.
* Leg de verschillende stappen van het kwaliteitsvol labellen van data, via een Active Learning approach uit. Leg ook uit waarom de querying stap hierin zo cruciaal is.
* Leg kort uit wat model distillation betekent (of knowledge distillation) in de context van Edge computing.

**Week 7:**

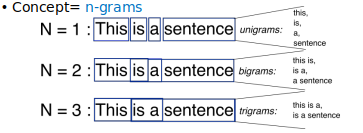
* Leg het principe van tokenization uit, in de context van NLP (gebruik hiervoor: n-grams, corpus, dictionary, vocabulary, normalization en cleaning van tekst, special tokens, stemming/lemmatization, stop words).

**Tokenization** is een veel voorkomende taak in NLP, het is in feite een tekst in verschillende delen splitsen, zodat je meaningful chunks krijgt (tokens) en eventueel bepaalde karakters weggooien.

Dan voer je token **normalization** uit door verschillende vormen van hetzelfde woord dezelfde token te geven er zijn 2 verschillende processen (Stemming & Lemmatization). **Stemming** is de achtervoegsels verwijderen en vervangen en zo een basis woord vormen (stem) (vb. getting → get) en **lemmatization** maakt gebruik van **vocabularies** zoals wordnet om zo het basis woord te vinden (vb. wolves -> wolf of feet → foot).

**Stop words** zijn woorden die beter gefilterd kunnen worden tijdens de pre processing fase. (vb. the)

Een **n-gram** hier geeft gewoon aan naar hoeveel woorden je tegelijk naar de tekst kijkt (San Francisco is een 2-gram, ik ben hier is een 3-gram).



**Corpus** verwijst naar de collection van teksten. En vocabulary geeft aan welke woorden uw network kent.

* Leg de vectorization van tokens uit: frequency based (BOW, tf-idf), en probability based word-embeddings (word2vec).

Het proces van het omzetten van NLP-tekst in getallen wordt vectorisatie genoemd. Verschillende manieren om tekst om te zetten in vectoren zijn:

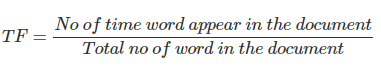
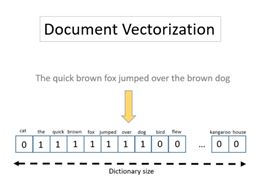
Frequency based: kijkt hoeveel het woord voorkomt in de dictionary

**BOW** (Bag of Words) is een basis model gebruikt in nlp waarbij elke volgorde van de woorden in het document worden verwijderd. Het kijkt alleen als een woord aanwezig is in het document of niet.

**TF-IDF** (Term Frequency-Inverse Document Frequency) geeft het belang van het woord in het corpus of het document. TF geeft weer hoe vaak een woord voorkomt en IDF duidt het belang aan. Deze 2 tables worden vermenigvuldigd. Het vermindert in feite de waarden van veelvoorkomende woorden die in een ander document worden gebruikt.

Probability based: context en volgorde is wel belangrijk

**Word-embedding** zorgt voor het verbeteren van het netwerk door data weer te geven als dimensionele vectors. Dit wordt gedaan door een numerieke vector aan elk woord in een dictionary te koppelen, zodat de afstand tussen twee willekeurige vectoren een sematic relatie vormen. De geometrische ruimte gevormd door deze vectoren wordt een inbeddingsruimte genoemd. Deze vectors noemen embeddings. De geometrische afstand tussen deze vectors wordt een embedding space genoemd. Een van de populairste word embeddings is **Word2vec** van google.





* Leg uit wat het long term dependency problem is bij word embeddings, en hoe we een Language Model kunnen trainen via RNN/LSTM/GRU of 1-D CNNs om dit probleem in beperkte mate op te lossen

1-D CNN’s kunnen de volgorde modelleren door op te treden als n-gram feature extractors voor embeddings.<https://towardsdatascience.com/beyond-word-embeddings-part-2-word-vectors-nlp-modeling-from-bow-to-bert-4ebd4711d0ec>

* Leg de verschillende evoluties in het NLP domein bondig uit. Focus op hoe iedere evolutie een verbetering is van de vorige manier van werken ((BOW/one hot encoding word embeddings (word2vec, glove,fasttext) RNNs/CNNs attention enrichment Transformers Transfer learning (ULMFit, ELMo, BERT, GPT-2))

**Week 11:**

* Om sequentiële data te verwerken, moeten we context kunnen mee-modelleren. Hiervoor hebben we een CNN of RNN nodig. Leg het basisprincipe van beide approaches uit (in de context van sequentiële data).

Bij sequentiële data hebben we contextinformatie nodig, er moet gekeken worden naar het verleden om zo goede voorspellingen te doen (time series prediction)

Via een CNN kunnen we contextinformatie gebruiken. Een kernel vat immers een gans gebied, context, samen tot 1 getal. Die kernel verschuiven we dan over de verschillende time steps (verschillende woorden in het verleden / of toekomst).

RNN leert van het verleden en zijn beslissing beïnvloed door wat hij al geleerd heeft van het verleden.

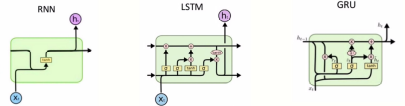
* Als we een RNN gebruiken, lopen we tegen het probleem van vanishing gradients aan (long term dependencies). Leg uit waarom, en hoe dit wordt opgelost bij RNNs, en waarom we niet de ‘standard’ oplossing hiervoor kunnen gebruiken, zoals bij FeedForward netwerken

Short-Term memory en de vanishing gradient probleem is vanwege de aard van back-propagation. Een gradient wordt gebruikt om de interne network weights aan te passen waardoor het netwerk kan leren. Hoe groter de gradient, hoe groter de aanpassing. Wanneer back-propagation gebeurt berekend iedere node zijn gradients rekening houdend met het effect van de gradients ervoor. Als de aanpassing van de layer ervoor al klein is is de aanpassing aan de huidige layer nog kleiner. Dit zorgt ervoor dat gradients exponentieel verminderen tijden back-propagation. Dat is het vanishing gradients probleem.

Een RNN maakt gebruikt van back-propagation in time waarbij elke stap een layer is, dus gradients verminderen exponentieel door tijd. Hierdoor kan een RNN geen long term dependencies leren in tijdstappen.

Om het kortetermijngeheugen te verminderen zijn er 2 gespecialiseerde RNN’s, LSTM (Long Short-Term Memory) en GRU (Gated Recurrent Units). Ze functioneren net hetzelfde als een RNN maar ze kunnen long-term dependencies leren aan de hand van het ‘gates’ mechanisme. Deze ‘gates’ zijn verschillende tensor operations dat kan leren welke informatie er moet toegevoegd worden en welke verwijderd mag worden van de hidden state.

Vanishing gradient kan je niet oplossen met batch normalization. Daarvoor heb je immers een batch nodig, zodat je average en mean kan uitrekenen per batch om zo de data terug naar sweet spot te brengen. Maar bij RNN veranderen de elementen in een batch bij iedere time step, of aanpassing. Het volgende element is immers afhankelijk van het vorige, maar als je het vorige zou aanpassen via een batch norm, heeft dit weer invloed op de volgende, waardoor die batch norm weer niet goed was,…



* Leg uit wat een LSTM netwerk is (Long Short Term Memory), en waarvoor we dit kunnen gebruiken. Om een LSTM model meer power te geven om complexe data te modelleren, kunnen we daar een multi-layered (zelfs bi-directional) model (deep LSTM) van maken, maar dan lopen we de kans om snel te overfitten. Leg uit hoe we dit kunnen oplossen.

De cell state lijkt op een transportband. Het loopt recht door de hele keten, met slechts enkele kleine lineaire interacties. Informatie kan niet worden toegevoegd of verwijderd aan de cell state, dit word gereguleerd door structures genaamd gates. Een gate is altijd opgesteld door een sigmoid neural net layer en een pointwise multiplication operation. LSTM heeft 3 gates voor de cell state te controleren en beschermen.



De forget gate is een sigmoid layer die beslist welke informatie we weg gaan gooien van de cell state. Output is een nummer tussen 0 en 1 voor elk nummer in de cell state. 1 staat voor ‘completely keep this’ en 0 staat voor ‘completely get rid of this’.

De input gate layer beslist welke nieuwe informatie we gaan toevoegen aan de cell state, dit bestaat uit 2 delen. Eerst een sigmoid layer die beslist welke values we gaan updaten en dan een tanh layer die de vector bepaald van de nieuwe candidat values.

De output gate is gebaseerd op onze cell state maar een gefilterde versie. De cell state word door een tanh gestuurd en multiplied door de output van de sigmoid gate zodat we alleen maar de output hebben die we willen.

**Week 12:**

* Leg uit wat multiclass- en multilabel classification is. Leg de activation- en loss functions van beide uit.

Multiclass classification:

Categorical crossentropy = softmax activation function + cross entropy loss

De logits = uitkomst van de affine functions (aW+b), dat daarna door een non-linear part wordt gedaan, de softmax (squashing naar probabilities tussen verschillende categorieën).

Multilabel classification:

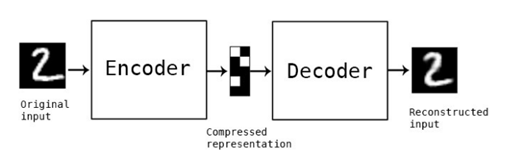
Binary crossentropy = sigmoid activation + cross entropy loss

* Leg uit wat segmentation van een image is. Leg de typische architectuur uit van een CNN netwerk dat kan segmenteren. U-Net gebruikt in zijn architectuur ook skip connections. Leg uit wat dit doet.

Image segmentation is de methode om het beeld in verschillende segmenten te verdelen, waarbij elk segment een andere entiteit heeft.

//TODO: CNN MET SEGMENTEREN EN U-NET UITLEGGEN

* Leg uit wat de basisarchitectuur van een auto-encoder is, en hoe we dit kunnen gebruiken om een image bv te denoisen. Leg in deze context ook uit wat het verschil is tussen een L1- en L2 loss.



Encoder:

Het model leert hier hoe je de input dimensies vermindert en de input data compressed naar een encoded representation.

Bottleneck:

Dit is de laag die de compressed representatie van de input data heeft, dit is de laagst mogelijke dimensie van de input data.

Decoder:

Hier leert het model hoe hij de data moet reconstrueren van de encoded representatie en dit probeert die zo hard mogelijk op de originele input te laten lijken.

Reconstruction Loss:

Dit is de methode die meet hoe goed de decoder zijn werk aan het doen is en hoe dichtbij de originele input de decoder geraakt is.

Als het netwerk getraind is kan er een image met noise worden doorgestuurd en de decoder genereert dan terug een image zonder de noise erop.

De L2 los neemt de squared euclidean distance terwijl de L1 de manhattan distance gebruikt dus de edges bij L2 zijn meer blurry terwijl die van L1 scherp en squary zijn waardoor de image scherper is bij L1.

* Leg de basisarchitectuur van een GAN uit, en waarom we in deze architectuur een apart netwerk trainen om als loss function te dienen (tov de gewone L2 loss bv)