Verslag

Dataset

Voor mijn dataset gebruik ik +- 3000 hoge resolutie afbeeldingen (minimaal 1440x720) deze zijn verzameld door middel van een flickr scrape script, dit script downloadt heel veel afbeeldingen van flickr (een site waar fotografen hun foto's posten). Het commando wat gebruikt is om afbeeldingen te downloaden is als volgt:

python3 scraper.py –search "Zoek term" –bbox "regio" –start-page 0 –max-pages 4 –original

--max-pages 4 voorkomt dat hij te veel afbeeldingen van dezelfde foto's downloadt, al zijn hier achteraf handmatig nog controles voor geweest.

--original wil zeggen dat hij de originele grootte van de afbeelding gebruikt.

--start-page geeft aan welke pagina we beginnen, wij downloaden de meest recente foto's (pagina 0)

--bbox geeft aan in welke regio van de wereld je wilt downloaden, ik koos dit niet te limiteren en gebruikte dus bbos -180 -90 180 90.

--search geeft aan welke zoek term je wilt gebruiken, ik heb dit heuristiek gedaan met een aantal zoektermen:

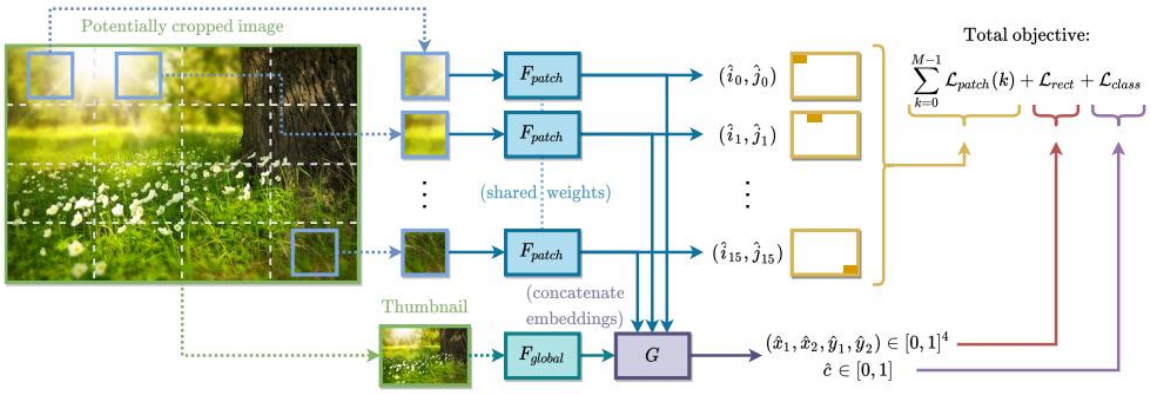
- Travel
- Tourism
- Summer
- Street
- Sport
- Protest
- Portrait
- Photography
- Park
- Outside
- Online
- News
- Mountains
- Morning
- Military
- Life
- Landscape
- International
- Horizon
- Demonstration
- Celebrity
- Boat
- Beach

Nadat de afbeeldingen waren gedownload gingen hier twee scripts overheen, als eerste een filter script. Het filter script verwijderd alle afbeeldingen die niet aan onze eisen voldoen, de eisen die wij

stellen zijn eigenlijk alleen dat de afbeeldingen groter zijn dan 1440x720 en niet bewerkt zijn. Aangezien kleine crops (denk aan het rechttrekken van de horizon) van nature bij fotografie horen zit er een kleine marge in wat we onder bewerking verstaan. Na dit gezegd te hebben zijn we dus op zoek naar afbeeldingen met een aspect ratio van 1,5 oftewel 2:3. We zoeken ook alleen naar horizontale afbeeldingen, staande afbeeldingen roteren wij. Alle afbeeldingen die overblijven worden verplaatst naar de opgegeven sdir (sorted directory).

Ten tweede hebben we een script die alle afbeeldingen sorteert naar train/ test/ val/ directories, dit wordt gedaan met een split van 75/ 15/ 10/.

Neurale Netwerk



Ik wil het netwerk gaan implementeren in 3 stappen, Ik wil beginnen met alleen het Lclass netwerk. Als dit lukt wil ik deze uitbereiden met het Lrect netwerk en als dit werkt wil ik ten slotte ook het Fpatch netwerk implementeren, bij elke stap wil ik de accuracy van het netwerk opslaan en kijken hoeveel impact de volgende stap heeft.

Lclass

Het was in eerste instantie de bedoeling om Lclass te maken, dit netwerk moet een percentage van de crop kunnen voorspellen. Lclass is een 3-layer perceptron, deze ontvangt een concatenated input van alle outputs van het patch netwerk en het thumbnail netwerk, ik begin met alleen een input van het thumbnail netwerk. Vervolgens geeft Lclass 5 outputs, de eerste 4 geven de percentages van de resolutie ten opzichte van de originele resolutie. En als laatste de crop factor of size factor, dit is het percentage waarmee de afbeelding gecropt of geresized is.

Tijdens het implementeren liep ik tegen een groot probleem aan, alle outputs waren altijd hetzelfde. Ik merkte dat als ik maar 3 epochs deed met een learning rate van 0,001 dat ik dan nog wel variabele outputs kreeg. Naarmate ik meer trainde met een learning rate van 0,001 kreeg ik steeds minder variabele outputs. Na wat advies van andere en wat onderzoek besloot ik om de learning rate te verlagen, uiteindelijk kwam ik op een leraning rate van 1 * 10^-6, hier wijk ik wel af van de auteur van mijn paper die een exponentieel afnemende learning rate heeft van 1 * 10 ^-5 tot 5 * 10^-6, deze learning rate neemt af op basis van epochs. Verder heb ik ook de verdeling tussen gecropte en niet gecropte afbeeldingen geskewed, naar 80% afbeeldingen die gecropt zijn en 20% die niet gecropt zijn, dit in poging om het netwerk harder te straffen voor outputs van alleen maar 0,0,0,0,0 wat eerst wel vaak het geval was. Wederom wijk ik hieraf van de auteur die een verdeling heeft van 50/50.

Lrect

Lrect is best wel straight forward, het is een ResNet34 model die als output een length-64 embedding heeft. Verder krijgt Lrect een gedownscalede thumbnail van de invoer afbeelding of dit een crop is of niet. De downscaled resolutie is (224, 149).

Lpatch

Lpatch is een ResNet 18 model die voor elke patch die hij binnenkrijgt geeft hij een probability distribution van de geschatte locatie. Dit zijn dus 16 outputs per patch.

De patches zelf worden genomen door de afbeelding (crop of niet) op te delen in een grid van 16 cellen, uit elke cel wordt vervolgens een patch van 96x96 genomen. Elke patch wordt met hetzelfde netwerk getraind. Het is uiteindelijk niet gelukt om Lpatch in het grote netwerk te knopen, ik heb dit netwerk dus afzonderlijk getest.

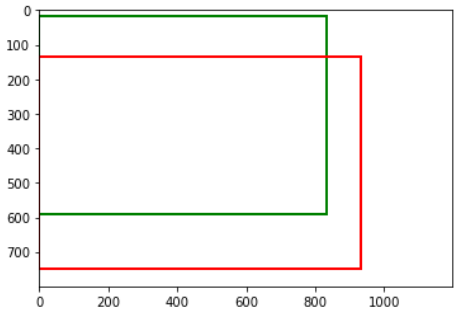
Optimalizaties

Het trainen duurde erg lang, dit kwam voornamelijk omdat alle training op mijn CPU gedaan werd, per batch van 256 afbeeldingen kon dit tot wel 20 seconden duren, het inladen van de afbeeldingen duurde echter nog langer. Om dit proces te versnellen heb ik alle afbeeldingen die groter waren van 2048 in de breedte geresized naar een random breedte tussen [1024..2048] en alle afbeeldingen verplaatst naar een M2 SSD, dit verbeterde de laad tijd van de afbeeldingen drastisch. Ik heb dit echter terug moeten draaien toen ik begon aan het patch netwerk.

De tweede optimalisatie was het gebruik maken van de GPU, om dit te doen heb ik de NVIDIA CUDA driver moeten installeren en het tensorflow-gpu pakket. Dit versnelde het trainen enorm.

Experimenten

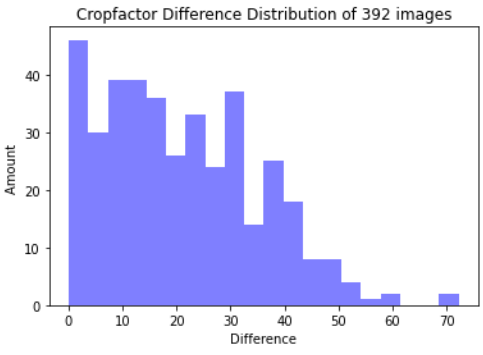
Thumbnail + Class



Deze resultaten zijn behaald door 500 epochs te draaien over een dataset van 2000 afbeeldingen.

Om te visualiseren of mijn netwerk een beetje juiste voorspellingen maakt heb ik mijn test data door mijn netwerk heen gehaald en de uitkomst met matplotlib mooi geprobeerd weer te geven. In de afbeelding hier links zien we een plot met de resolutie van de originele afbeelding, de groene rectangle is de crop resolutie en de rode rectangle is de crop resolutie die het netwerk heeft voorspeld.

Verder heb ik ook een experiment gedaan op alleen de crop factor voorspelling, ik ben hier door mijn hele test set heen gegaan en heb telkens de afwijking berekend (echte crop factor – geschatte crop factor), deze waardes heb ik allemaal bij elkaar opgeteld waarna ik ze heb gedeeld door het totaal aantal afbeeldingen. Ik kom dan uit op een gemiddelde afwijking van 21%. In de afbeelding hiernaast kan je de verdeling zien van de cropfactor redelijk goed wordt voorspeld, er is nog ruimte voor verbetering maar zeker niet slecht.

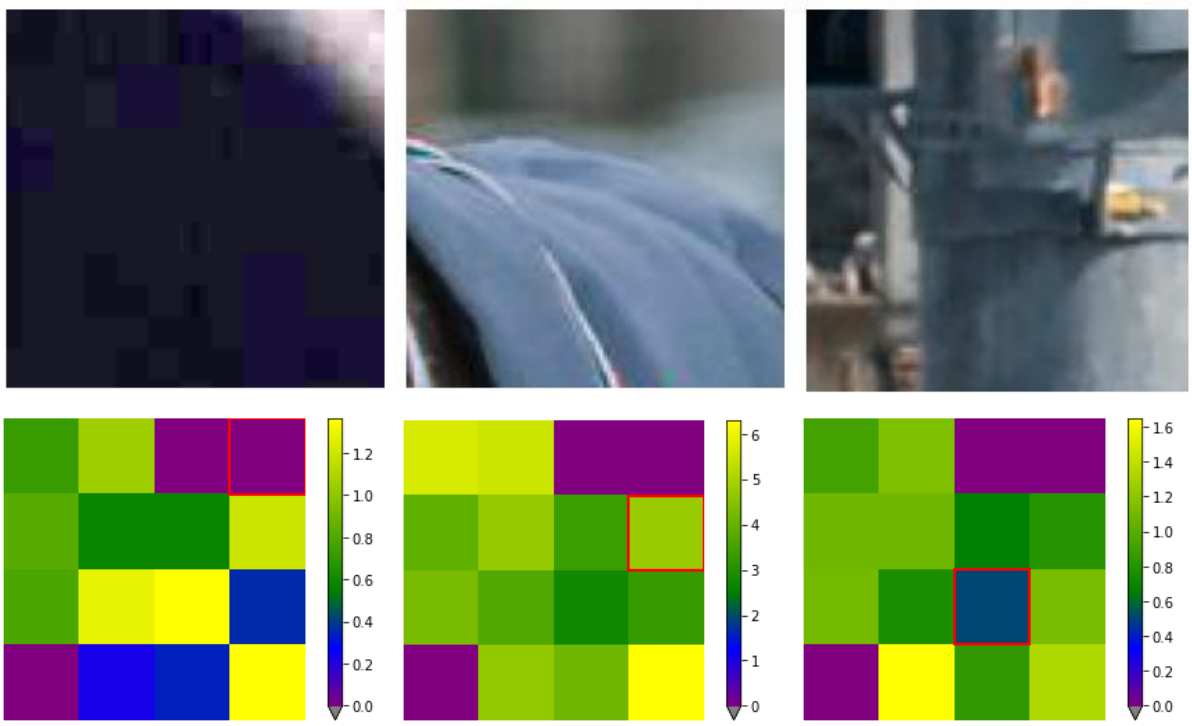


Patch

Voor het patch netwerk heb ik de output van mijn data geprobeerd te visualiseren door middel van een zogenaamde “heatmap”. Hier links kunnen we een afbeelding zien met daaronder de voorspelling van het Patch netwerk. De rode rectangle geeft aan wat de echte positie van de patch is ten opzichte van de hele afbeelding.

Ik heb het patch netwerk getraind op een dataset van +- 2000 afbeeldingen met 140 epochs, de resultaten hiervan waren teuleurstellend om het zacht uit te drukken. Alle resultaten waren hetzelfde. Na wat onderzoek denk ik dat dit komt door mijn relatief gezien kleine dataset. Aan mijn learning rate lag het niet, ik heb namelijk geëxperimenteerd met een learning rate van 1*10^-3 tot 1*^10-6 en allemaal gaven ze (altijd) hetzelfde resultaat.

Een tweede poging met een dubbel zo groote dataset gat echter wel resultaten, nog niet enorm goede maar je kon zien dat er wat gebeurde. De resultaten van deze poging staan hieronder:



De resultaten in de bovenstaande afbeelding zijn getraind op een dataset van +- 4000 afbeeldingen met 6 epochs (dit in verband met de tijd). Je ziet een aantal herhalende patronen, hij denkt bijvoorbeeld nooit dat de afbeelding rechtsboven of rechtsonder is. Ook denkt het model redelijk vaak dat e afbeelding linksboven of rechtsonder is. Deze patronen zijn denk ik weg te werken door een stuk meer te trainen.

Opmerkingen

Er zijn een aantal dingen die in eerste instantie niet helemaal duidelijk waren aan de paper, om hier toch duidelijkheid in te vinden heb ik zelf door de auteur zijn code moeten scrollen. De onduidelijkheden gingen hier om de activatie functies van de embedding layers en het perceptral, na wat zoeken kwam het erop neer dat dit allemaal ‘relu’ functies waren.

De tweede onduidelijkheid zat hem in het patch model, de uitvoer van het patch model in het Lclass model is een length-64 embedding, om hier de patch locaties van te krijgen moet je hier nog eens een lengt-16 embedding overheen gooien.

Afwijkingen

- Er zijn eigenlijk 2 grote punten waarop ik afwijk met de autheur, ik heb een vele male kleinere dataset (4000 ten opzichte van 700.000).
- Ik heb geen exponentieel afnemende learning rate functie. Ik heb dus standaard al een lagere learning rate dan de auteur.