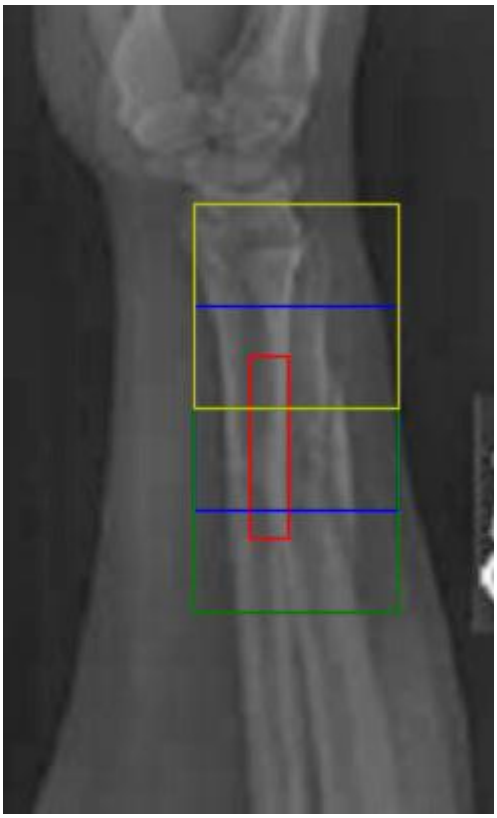


BOOTBREUK DETECTIE VERSLAG

ABSTRACT

Met een simpel model en vergroting van een kleine dataset is het mogelijk geweest om botbreuk te classificeren als geen of wél botbreuk. De detectie op grotere afbeeldingen is bereikt maar de nauwkeurigheid wisselde echter sterk, onder andere door de aanwezigheid van voorwerpen zoals gipshoezen, die sterk lijken op botbreuken op x-ray images. Een betere scheiding van beeldlagen via segmentatie zou de prestaties verder kunnen verbeteren

Alexander Navarro

Background

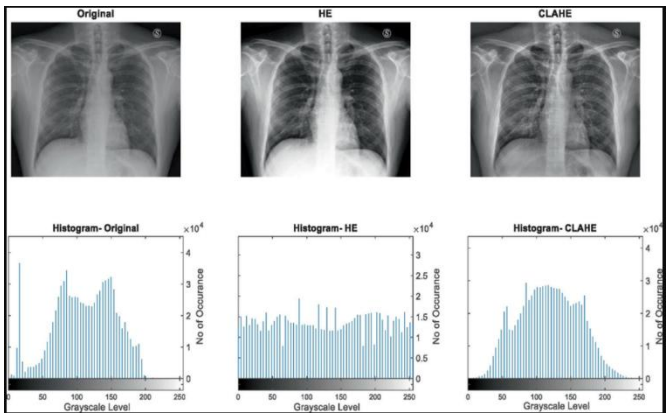
CLAHE

Contrast Limited Adaptive Histogram Equalization (CLAHE) is een techniek die helpt om de details in een afbeelding duidelijker te maken. In het artikel van Qiuru Wang (Wang, 2024) gaat het om oude afbeeldingen van orakelbotten (botten van runderen). Deze foto's zijn vaak donker, vaag of hebben een slecht contrast, waardoor het moeilijk is voor een model om de tekens goed te herkennen. CLAHE zorgt dus ervoor dat het verschil tussen licht en donker in kleine stukjes van de afbeelding wordt versterkt, zonder dat het te veel ruis toevoegt. Hierdoor worden letter of tekens op de bot beter zichtbaar.

Het idee blijft hetzelfde voor de detectie van breuken. Breuken zijn in het algemeen dunne en lege sleufjes met onregelmatige vormen. CLAHE helpt met de verbetering van het contrast tussen de botstructuur en de breuklijn, zodat het model deze beter kan onderscheiden van de achtergrond en andere weefsels. Hierdoor worden ook kleine of vage breuken beter zichtbaar in de afbeelding, wat leidt tot een nauwkeurigere herkenning.

Histogram Equalization (**HE**) is het originele concept over de herdistributie van de intensiteitsniveau van de pixels door de hele afbeelding heen. Eerst wordt het aantal pixels in de volledige afbeelding geteld die een bepaalde intensiteitsniveau tonen en het resultaat wordt op een histogram weergegeven. Vervolgens wordt de distributie van de intensiteit uniform uitgevoerd op het hele plaatje door gebruik te maken van een intensiteitstransformatie functie. Dit kunnen we waarnemen in figuur 1-1 onder het HE-plaatje.

Daarentegen maakt **CLAHE** gebruik van een kleine matrix (bv. 3x3 of 8x8) om het histogram van lokale plekken in de afbeelding aan te maken en daar de intensiteitstransformatie functie toepassen. De distributie van het intensiteitsniveau ziet er dus minder uniform uit dan bij HE omdat het uitkomst relatief is aan de gemiddelde intensiteit van iedere gebied. Het voordeel hiervan is dat de textuur van de bot wordt beter gehouden en onderscheidt zich van andere structuren zoals spierweefsels. Bij zones in de afbeelding waar de helderheid zeer hoog of laag is, werkt CLAHE efficiënter dan HE. Dit komt doordat in sommige gebieden het niet per se nodig is om de intensiteit van de pixel aan te passen. In figuur 1 zien we dat in het originele plaatje de wervels minder zichtbaar zijn. Bij de HE-plaatje zijn deze weer niet zichtbaar, maar het verschil per wervels is wel aan te merken bij het plaatje waar CLAHE is toegepast.



FIGUUR 0-1 VERSCHIL IN DE DISTRUBUTIE VAN INTENSITEITSNIVEAU TUSSEN HE EN CLAHE

Gabor Filters

Gabor-filters worden in verschillende toepassingen gebruikt zoals detectie en informatie halen uit röntgenstraling afbeeldingen van trabeculaire botten (Catherine M Gdyczynski, 2014). Hier kunnen Gabor filters herkennen in welke richting structuren liggen in de afbeelding. Ze lijken een beetje op hoe ons oog lijnen en patronen ziet. In dit geval helpen ze om te meten hoe de trabeculae (de fijne botstructuren) georiënteerd zijn.

Methode

De methode die ik gebruikt heb is de zogenaamde "*Sliding Window Object Detection*". Het doel van deze methode is dat een rechthoekig "*window*" (in mijn geval een *window* van 64px bij 64px) verplaatst wordt over een test afbeelding die het model nooit eerder heeft gezien. Als een vastgestelde drempelwaarde wordt overschreden tekent het model een *window frame* op het plaatje waar de verwachte bootbreuk in zit. Verschillende drempelswaardes zijn getest tijdens deze opdracht. In de Experimenten sectie van dit verslag wordt de keuze voor iedere waarde verder uitgelegd.

Deze methode is heel eenvoudig te implementeren maar levert lage nauwkeurigheid op omdat het model contekst mist over wat een gezonde onderarm in zijn geheel is. Deze contekst kan het model leren door *image segmentation* op de plaatjes toe te passen, maar de dataset bevat geen segmentatie wat veel meer tijd voor het labelen zou kosten.

Om het model basale kenmerken van een botbreuk te laten leren, zoals veranderingen in de richting van scherpe randen of plotselinge continuïteitsafwijkingen van het bot, worden de trainings- en validatieplaatjes voorbewerkt met verschillende image manipulation methodes.

Image preprocessing

De images zijn verwerkt op z'n manier dat de textuur van de bot duidelijk is.

1. Hiervoor wordt eerst contrast op de images uitgevoerd m.b.v. CLAHE.
2. Ruis uit het contrast stap verwijderd met een kleine 3x3 Gaussian kernel.
3. Gabor Filters helpen de edges te detecteren. Deze kunnen als een kanaal worden toegeveogd.

Dataset strategie

De images uit de dataset worden geknipt volgens de aangegeven coördinaten in de bijbehorende labels en aan het model gevoerd voor binaire classificatie.

Training en validation subsets

De subset voor training en validation zijn heel klein waardoor **data augmantation** nodig is. Hiermee kunnen we *overfitting* voorkomen. Om dit aan te pakken worden de afbeelding drie keer 90° geroteerd en verschillende contrast niveaus uitgevoerd.

Voor afbeeldingen waar er geen botbreuk in zit bevatten de labels geen coördinaten en klasse ID. Dit is iets normaals in YOLO-notatie. Dit aspect van de dataset vereist dus dat plaatjes worden gegenereerd om als 64x64 niet-botbreuk afbeeldingen te classificeren. Omdat de dataset meer dan 3000 images met verschillende breuk klasses (pols, been, etc) heeft en wordt niet aangegeven of er een gezonde onderarm in zit, heb ik dus ervoor gekozen deze images random te genereren. Het enige parameter voor het generen van deze images is dat de gemiddelde helderheid van de image minstens groter is dan 30 (in een gray schaal). Op deze manier leert het model dat sommige complexe vormen geen botbreuken zijn.

In total zijn er 1572 images gegenereerd uit 262 botbreuk images en 262 random gegenereerde images voor het trainen van de het model.

Test subset
De dataset bevat slechts 6 images voor het testen van het model. Hiermee is er te weinig om te achterhalen of het model goed genoeg werkt om de sliding window te implementeren. Daarvoor heb ik dus uit de training dataset 40 images (20 positief + 20 negatief) genomen en verplaatst naar de test afbeeldingen.

Model trainen
Het model moet capabel zijn om meerdere vormen te herkennen waar er een gemeenschappelijk feature in zit, het feite dat er een lege plek zit tussen de boten.

Twee aspecten kunnen het leerproces in belangrijke mate beïnvloeden. Allereerst de aanwezigheid van kraakbeen tussen de botten, wat visueel overeenkomt met lege ruimtes en daardoor verwarring kan veroorzaken voor het model. Daarnaast speelt de textuur van gipshoezen een rol. Deze texturen worden vooraf bewerkt, maar het kraakbeen dat zich bij de gewrichten bevindt, blijft aanwezig en beïnvloedt het leerproces van het model aanzienlijk, omdat het lijkt op lege ruimtes en daarmee bijdraagt aan ambiguïteit tijdens de training.

Tijdens het trainen worden het aantal hyperparameters aangepast. Eerst wordt er gekeken of een groot aantal parameters een goed resultaat oplevert. Met goed resultaat wordt er bedoeld dat het model minstens 70% accuraat is. Het aantal *convolutional kernels* is stellen we groot in, maar wordt verminderd naarmate de accuraatheid hoger resulteert.

Voor de classificatie worden tijdens de eerste architectuurontwerpen alleen met *convolutional layers* gewerkt. Daarna voegen we *dense layers* toe om het verschil in prestatie waar te nemen. In theorie zouden alleen convolutional layers voldoende informatie en features uit de plaatsjes kunnen opnemen omdat het model alleen de aanwezigheid van de bootbreuk voorspelt. De complexiteit in de praktijk is echter breder vanwege de onregelmatige vorm van de randjes en de aanwezigheid van gewrichten. Hoewel deze aspecten meer te maken hebben met ruimtelijke features, kan de toevoeging van *dense layers* de classificatie benutten.

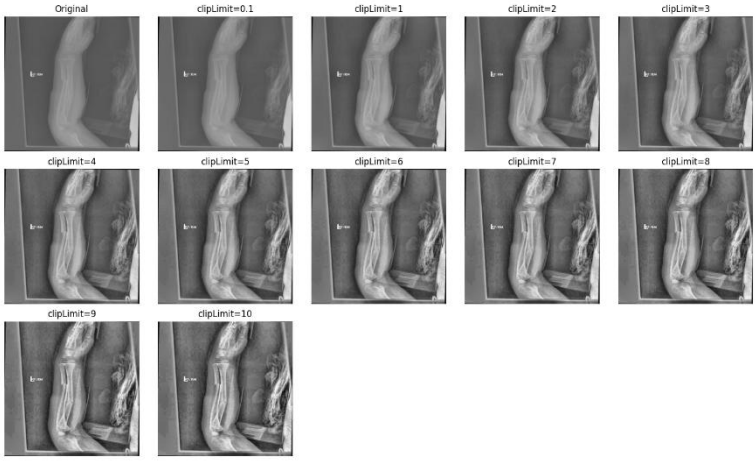
TEST
Voor het testen van de accuraatheid wordt een simpel gemiddelde berekend van het aantal afbeeldingen dat correct wordt herkend. Daarna wordt er gekeken naar de confusion matrix van het model. Dit helpt om te bevestigen of de afbeeldingen met een botbreuk genoeg informatie en variatie bevatten en of het voor het model duidelijk is waar het op moet letten.

Sliding window test
Bij deze test wordt een *window* van 64x64 pixels over een test Plaatje verplaatst. De "Classifier" moet aangeven of in de *window* een botbreuk zit. De voorspellingen van de classifier worden weergegeven op het testplaatje met een *bounding box* en de bijbehorende voorspellingswaarde. Hoe roder de randen van de bounding box is, hoe zekerder het model is dat er in die *window* een botbreuk zit.

Experimenten
Het bepalen van de juiste contrast niveau
CLAHE bestaat ook uit een clipping waarde. Deze waarde is een voorgeschreven limietwaarde voor iedere histogram bin en wordt meestal gebruikt tussen 3 en 4. Deze clipping beperkt de amplificatie van de intensiteitswaarde in de buurt van een specifieke

pixel. In andere worden, clipping vooraf de vergroting van de intesiteit in de pixels van de *tile* (dus het gebied van de image die wordt aangepast) bepaalt hoeveel ruis wordt doorgelaten maar vermindert het contrast.

De onderstaande set van afbeeldingen heb ik uit de training subset gehaald en meerdere clipping waarde erop toegepast. Deze image heb ik gekozen omdat de spier en de bot vergelijkbare pixelsintensiteit hebben. Naarmate de clipping waarde toeneemt, worden de randen en textuur van de bot scherper.



FIGUUR 0-1 VERSCHILLENDE CLIP LIMIT WAARDES

Het model is getraind op basis van een clipping waarde van 4 omdat deze de beste contrast past bij meerdere images.

NB.: Tijdens het doorkijken van de images, is het uitgevonden dat er meerdere kopies zijn van de images, die al tussen de 15 en 30 graden zijn geroteerd en hebben verschillende contrastsniveaus. Als er nog extra images worden gegenereerd met waarschijnlijk dezelfde contrast en inhoud, dan ontstaat er mogelijk *overfitting*. Hierdoor is het besloten om alleen een specifieke waarde van contrast te gebruiken.

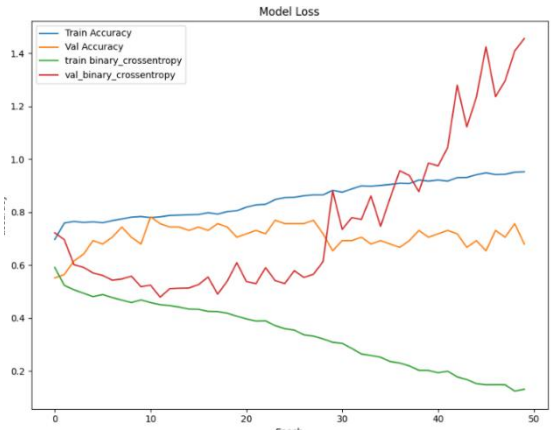
Model structuren

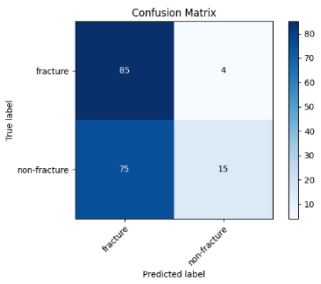
Bij het eerst model worden alleen kleine kernels (3x3 en 5x5) pixels gebruikt. Hier is een overzicht van het gebruikte model:

```
        layers.Conv2D(20, (5, 5), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(20, (3,3), activation='relu',),
        layers.MaxPooling2D(2,2),

        layers.Conv2D(20, (5,5), activation= 'relu' ),
        layers.MaxPooling2D(2,2),
        layers.Conv2D(10, (3,3), activation= 'relu' ),
        layers.Conv2D(num_classes, (3, 3), activation='sigmoid', name='output_layer'),
        layers.Flatten()
```

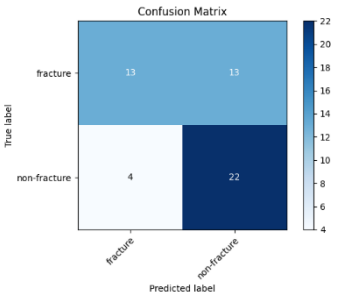
Binaire classificatie test

Dit model en dataset leveren de volgende accuracygrafiek en confusion matrix op:





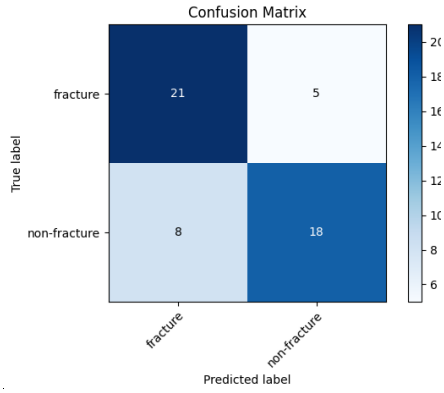
FIGUUR 0-3. 1/3 NON-FRACTURE IMAGES



FIGUUR 0-2. 1 OP 1 TRAINING SUBSET

Overfitting begint al vanaf epoch 15, wat waarschijnlijk wijst op een gebrek in de dataset, namelijk te weinig variatie om de data goed te generaliseren. Het model deelt de meest van de 46 images uit de test subset als fracture in. Bij de trainingset waren de afbeeldingen niet evenredig verdeeld: ongeveer 1/3 was non-fracture en de rest wel. Dit was opzettelijk gedaan om te bevestigen dat, ook al worden er weinig non-fracture afbeeldingen gevoed, het model deze alsnog kan onderscheiden en niet alle images als fracture indeelt. Vervolgens is het aantal non-fracture en fracture images gelijk verdeeld over de trainingsubset.

Aangezien een groot aantal parameters voor overfitting in lage epochs zorgt zijn het aantal kernels verminderd naar 20, 10, 10 en 10 respectieflijk. Deze zijn de resultaten.

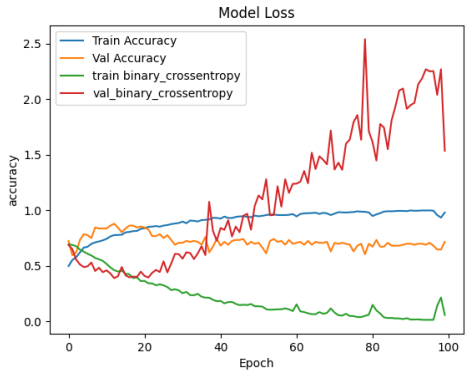


Het is op te vallen dat de prestatie van het model lijkt te verbeteren en dat het kruispunt tussen de train en validation *binary crossentropy* vanaf een latere epoch overfitting begint aan te tonen. De training accuracy blijft net niet op de 80%.

Vervolgens is er een dense layer van 20 neuronen toegevoegd aan het model om abstracte aspecten te realiseren, zoals het concept van discontinüiteit of variatie.

```
layers.MaxPooling2D((2,2)),# output size :(2,2,20)
layers.Flatten(),
```

```
layers.Dense(20, activation= 'relu'),
layers.Dense(num_classes, activation='sigmoid')
```

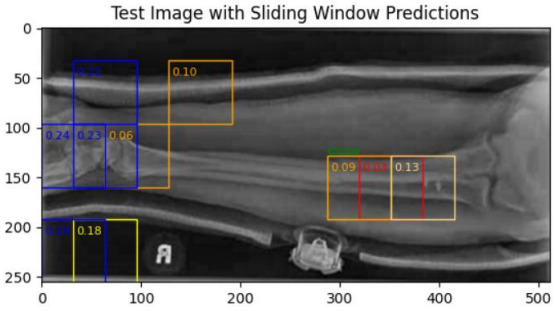


De validation binary crossentropy (loss function) stijgt heel vroeg in de epochs wat aantont dat het model data uit de training dataset aan het onthouden is en niet algemene features uit de dataset haalt.

Om de overfittingsprobleem te voorkomen zijn Dropout layers en Gaussian noise layers toegevoegd. De dropout layers zorgen ervoor dat het model meer generaliseert door random sommige kernels en neuronen uit te schakelen tijdens het trainen. Bij de volgende forward propagation worden (afhankelijk van het gekozen percentage) een nieuwe set van neuronen geselecteerd en gedeactiveerd. Op deze manier wordt het model niet afhankelijk van ontstaande padden en kan dus het netwerk verder andere patronen proberen te vinden. De onderstaande tabel laat de laatste berekende waardes tijdens de trainingsfase.

Model	Dropout (%)	Layers	Epochs	Train. loss	Val. loss	Binary clas. test (%)	Opmerkingen
1	**10**	- Conv: 4 - Polling: MaxPooling - Geen dense layer	50	0.1	0.98	81	- Overfitting vanaf epoch nr 10. De CM (Confusion Matrix) lijkt zich accurater, maar model deelt nog veel non-fracture images als fracture.
2	**20**	- Conv: 4 - Polling: MaxPooling - Geen dense layer	50	0.2	0.55	77	- Overfitting vindt nog steeds plaats, maar het kruispunt tussen train en val loss start vanaf een latere epoch, namelijk epoch 22. - Daarnaast is de validation crossentropy verlaagd ten opzichte van de vorige test. - De loss curve is minder fluctuerend en volgt een betere pad. Dit is een indicatie dat het model andere padden probeert te verkennen. Zie **figuur 9**. - Gipshoezen en complexe botstructuren blijken nog een probleem te zijn. Zie plaatje nr 10.
3	**30**	- Conv: 4	50	0.25	0.65	0.68	Dit model geeft minder zekeheid op de voorspellingen. Het is alsof het model te veel

		- Polling: MaxPooling					generaliseerd dat de onzekerheid van te voren kan worden achterhaald door validation loss.



FIGUUR 0-4 SLIDING WINDOW TEST VAN MODEL 2. ALLEEN VOORSPELINGEN ONDER 0.3 WORDEN WEERGEGEVEN.

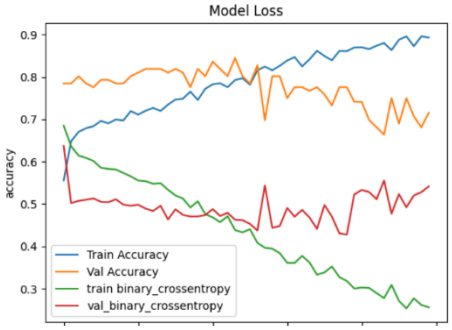


FIGURE 1 LOSS CURVE MODEL 2

De toevoeging van Dropout Layers heeft met het verlagen van overmatige stijging van de *validation loss* geholpen. Deze waarde blijft nog steeds te hoog waardoor de toevoeging van Gaussian noise werd aangenomen. Het model met 0.2 Dropout layer werd verder getest met verschillende Gaussian Noise waardes en dense layers. Hieronder zijn de resultaten en opmerkingen uit de training en test fases:

ID	Dropout (%)	Layers	Gaussian Noise	Epochs	Train loss (laatste epoch)	Val loss	Binary clas. test (%)	Opmerkingen
4	0.2	Hetzelfde als model 2, maar met 3 dense layer extra Aantal neuronen per layer: 30, 10 en output layer	0.05	100	0.23	0.65	77	Het model is minder gevoelig op ruis, maar de loss curve laat nog steeds zien dat er overfitting plaatsvindt.
5	0.2	Hetzelfde als model 2, maar zonder dense layers	0.1	100	0.24	0.75	0.62	We zien dat de val loss curve en de accuracy curve bij elkaar blijven. Dit betekent dat het model is op een limiet aangekomen. Een complexer model lijkt nodig te worden. Door te veel generalisatie begint het model de meest voorkomde label te voorspellen welke niet botbreuk is.

ID	Dropout (%)	Layers	Gaussian Noise	Epochs	Train loss (laatste epoch)	Val loss	Binary clas. test (%)	Opmerkingen
6	0.2	Hetzefde als model 2, maar **met** dense layers		100	0.15	0.85	67	De toevoeging van dense layers zorgt ook voor complexiteit in het model, maar blijkbaar zit er een limiet in de dataset omdat deze niet heel gevarieerd is. Dit wordt verder in de Discussie besproken.

Conclusie

Met deze opdracht is het waargenomen dat generalisatie van patronen in kleine dataset kan worden bereikt door de juiste technieken (in deze opdracht Dropout en Gaussian Noise) toe te passen en de resultaten van experimenten te analyseren. Daarnaast draagt data augmentation aan een verbetering in het overfittingsprobleem bij door variatie binnen de dataset te genereren.

Te sterke focus op het meest waarschijnlijk patroon kan echter ervoor zorgen dat het model minder zeker is van zijn voorspellingen en dat de output van het model een biased uitkomst kan opleveren. Door te veel generalisatie kan *underfitting* ook voorkomen. Om balans tussen een hoge bias en high variance aan te pakken zijn de resultaten uit de experimenten van groot belang geweest. Hiermee kunnen we afleiden dat een *dropout* waarde groter dan 20% en Gaussian noise van > 0.05 de prestatie van het model tijdens het trainen vermindert. De test resultaten uit de sliding window methode geven ook aan dat hoe meer het model patronen overmatig generaliseert uit deze dataset, hoe meer onverwachte resultaten het model levert.

Ten slotte kan er worden geconcludeerd dat generalisatie technieken minder kostbare oplossingen bieden dan data argumentatie voor het trainen van een binary classifier. Naast deze aspect helpen *sliding window* methode en confusion matrix de afwijkingen in de voorspellingen van het model beter zichtbaar te maken. Hiermee kunnen de foutpatronen van het model geëvalueerd worden.

Discussie

Geen segmentatie of maskers in de dataset

Het was beter geweest als ik was begonnen met segmentatie (segmentation) — dat is een techniek waarbij je eerst de onderdelen van de afbeelding los herkent, zoals bot, kraakbeen, spierweefsel en achtergrond. Maar in deze dataset zijn geen masks aanwezig die aangeven waar deze onderdelen precies zitten.

Als ik zulke segmentatiemaskers had, kon het model veel beter leren wat een echte botstructuur is (in dit geval de onderarm, dus twee lange “stokjes” zonder enige onderbreking), en zo ook beter herkennen wanneer die kapot is (bij een breuk).

Met segmentatie-maskers kan het model eerst leren wat het normale botpatroon is in de onderarm. Daarna kan het makkelijker herkennen wanneer dat patroon onderbroken is — dus bij een echte breuk. Ook kan het leren dat dingen zoals gips, die buiten de botstructuur vallen, niet hetzelfde zijn als de structuur en textuur van een bot.

Zonder die extra hulp moet het model alles zelf leren uit ruwe beelden (raw X-rays), en dat is veel moeilijker en gevoeliger voor fouten.

Bibliografie

Catherine M Gdyczynski, A. M. (2014). *On estimating the directionality distribution in pedicle trabecular bone from micro-CT images.* Semantic scholars.

Wang, Q. (2024). *Research on Image Preprocessing Based on the CLAHE Algorithm and Oracle Bone Inscription Image Segmentation Based on the YOLOv5s Algorithm.* 6th International Conference on Power, Intelligent Computing and Systems (ICPICS).