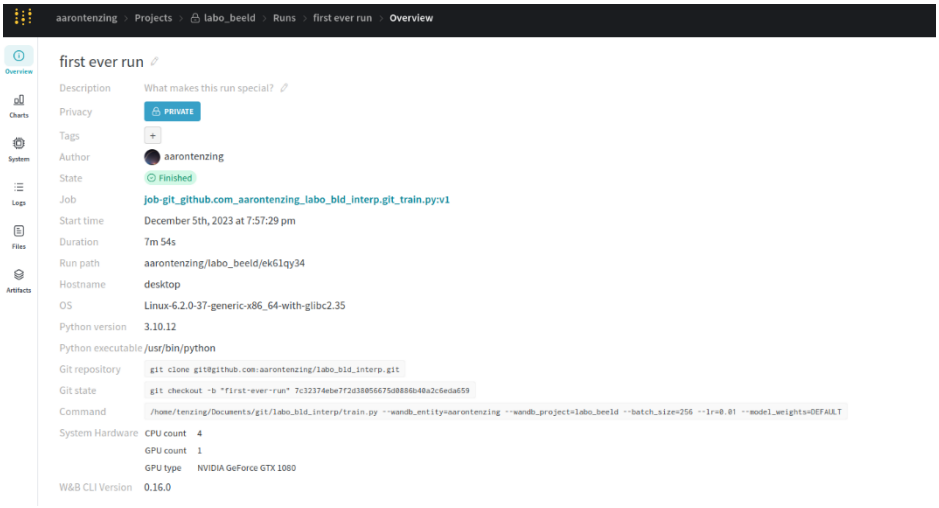
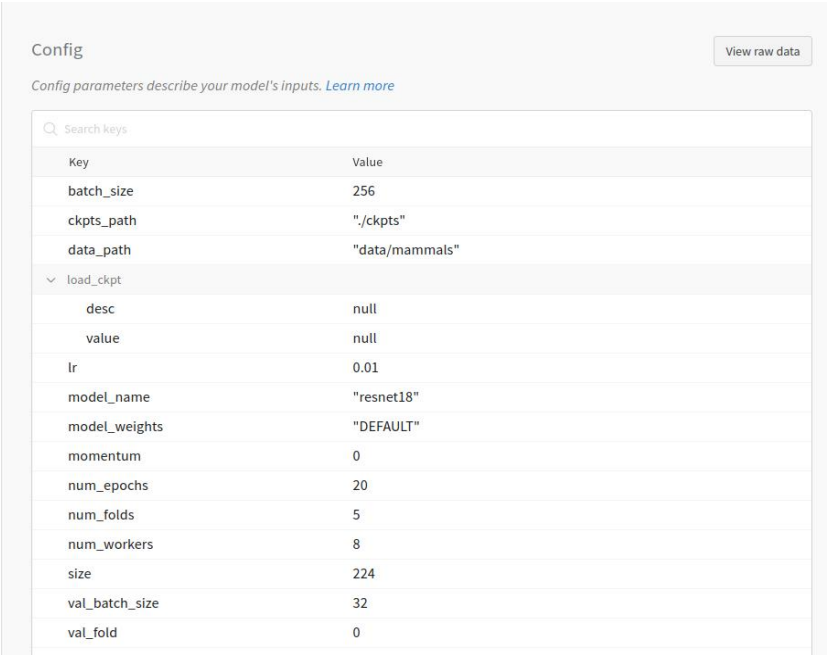
Aaron Van Campenhout

Verslag beeldverwerking

Pas code van andere notebooks en de .py files aan naar de nieuwe data zodat je met python train.py een correct werkende training kan opstarten.

Volgende screenshot toont aan dat het mij gelukt is een training uit te voeren op de Mammals dataset.
https://wandb.ai/aarontenzing/labo_beeld/runs/ek61qy34/overview?workspace=user-aarontenzing





Aaron Van Campenhout

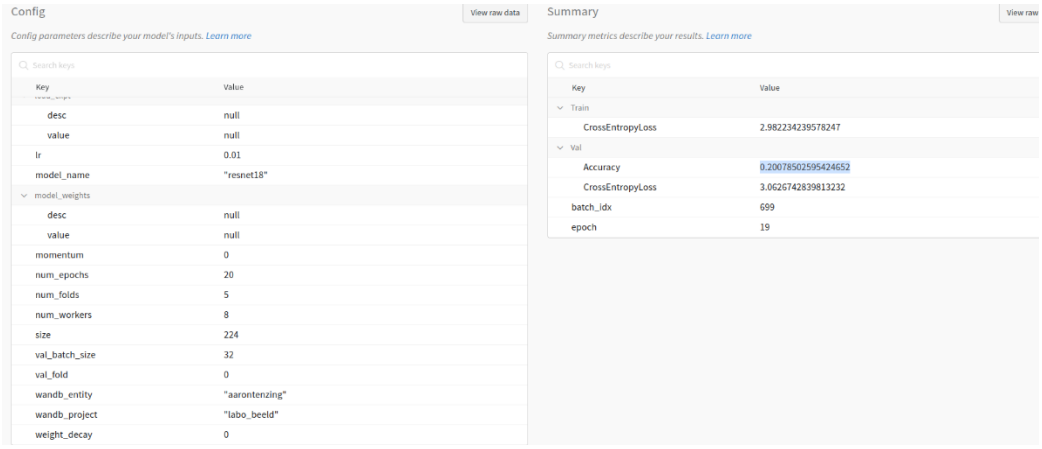
Voer de **volgende** experimenten uit:

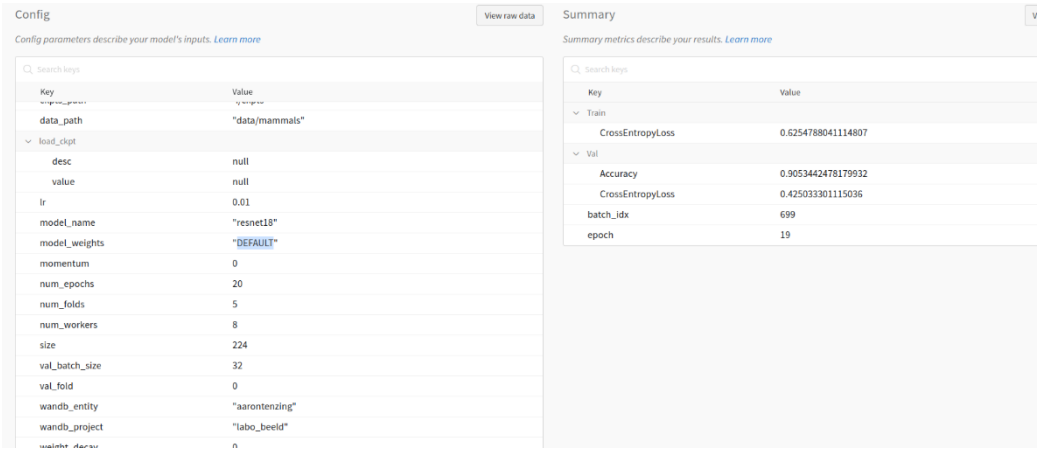
1. **Vergelijk accuracy van model dat met random weights en pretrained weights getraind wordt**

De accuracy van het model bij training zonder pretrained weights bedraagt: 20.079%.

De accuracy met het gebruik van pretrained weights ligt hoger namelijk: 90.534%.

Voor weights te gebruiken geven we –model_weights=DEFAULT “resnet18(pretrained=DEFAULT)”. Torch zal vervolgens automatisch weights initialiseren voor het Resnet18 model. Deze gewichten zijn bepaalt tijdens training op een grote verzameling van afbeeldingen uit verschillende categorieën (de ImageNet-dataset) en worden vaak gebruikt als startpunt. Deze gewichten zorgen er dus voor dat we betere resultaten bekomen, omdat we hierdoor niet van nul beginnen te trainen.





Aaron Van Campenhout

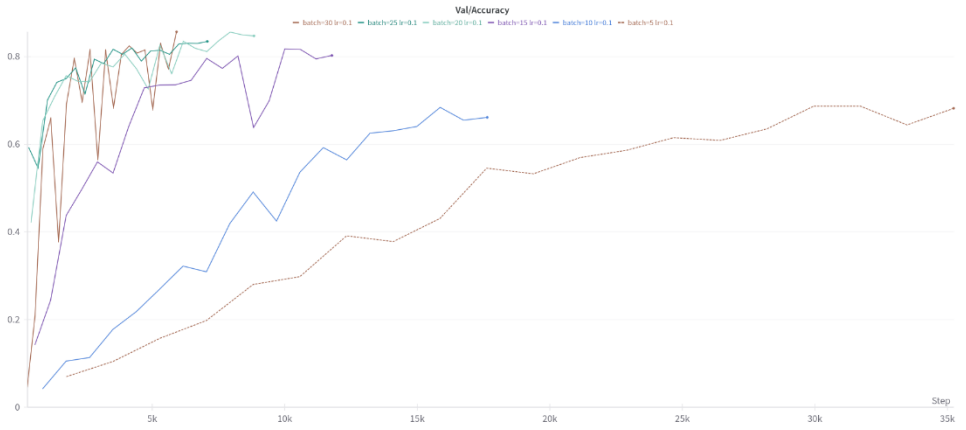
Bepaal de beste learning rate voor batch sizes 5, 10, 15, 20, 25, 30 en bespreek de resultaten.

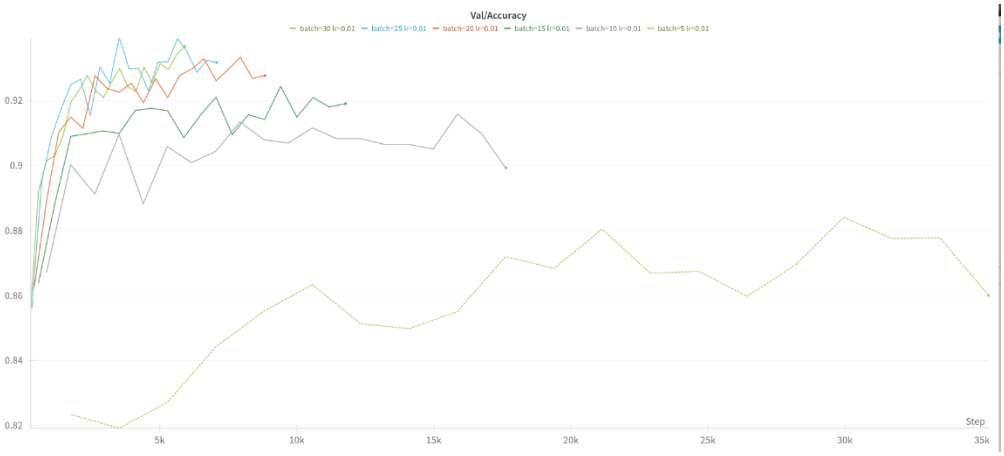
aarontenzing training model

51de188 · 44 minutes ago History

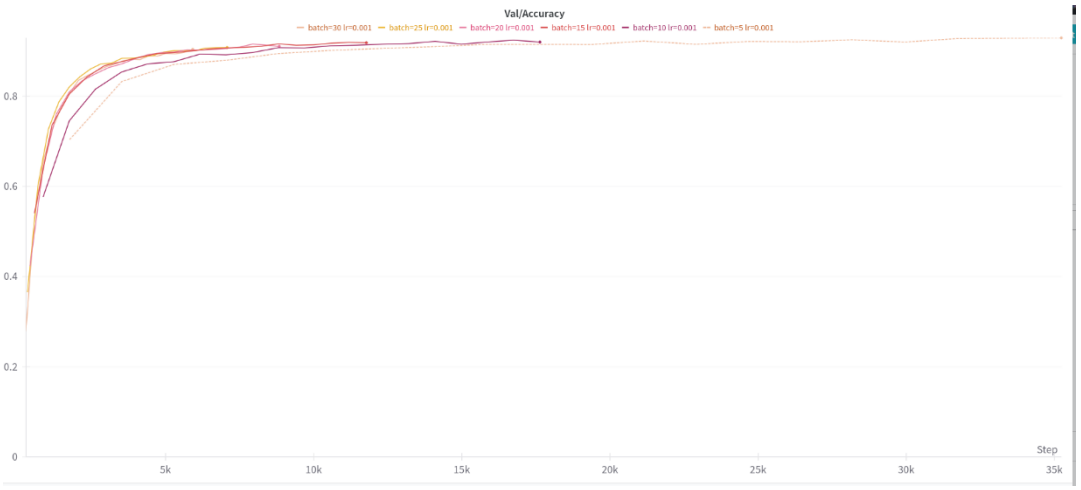
Code Blame Executable File · 16 lines (12 loc) · 427 Bytes Code 55% faster with GitHub Copilot Raw

```
# Array met verschillende batchgroottes
batch_sizes=(5 10 15 20 25 30)

# Array met verschillende learningrates
learning_rates=(0.001 0.01 0.1)

# Loop over batchgroottes en learningrates
for batch_size in "${batch_sizes[@]}"
do
    for lr in "${learning_rates[@]}"
    do
        python3 train.py --wandb_entity=aarontenzing --wandb_project=labo_beeld --batch_size=$batch_size --lr=$lr --model_weights=DEFAULT
    done
done
```





Aaron Van Campenhout



We zien hoe groter de batch size is, hoe minder iterations er nodig zijn om al onze epochs te door lopen. Kleiner batch grootte zorgt ervoor dat elke stap bij gradient descent minder accuraat is (aangezien kleine hoeveelheid data niet representatief is t.o.v. de hele dataset), dus het duurt langer voor te convergeren. Heel grote batch size is ook niet optimaal, hierbij gaat het heel snel convergeren maar het levert geen goede accuracy.

Hier is duidelijk te zien dat als we een te hoge learning rate gebruiken, er meer instabiliteit optreedt. Bij een learning rate van 0.01 zien we dat het model overshoot ervaart bij het zoeken naar het minimum van de lossfunctie. Het optimale resultaat wordt behaald wanneer we kleine learning rate nemen.

Kort besluit:

- Kleine learning rate: Bijvoorbeeld, 0.001, 0.0001 - kunnen geschikt zijn om langzamer te leren en minder fluctuaties kunnen veroorzaken.

- Matige leersnelheden**:** Bijvoorbeeld, 0.01, 0.005 - kunnen een redelijk compromis zijn tussen snel leren en stabiliteit, maar het kan variëren afhankelijk van het model en de dataset.

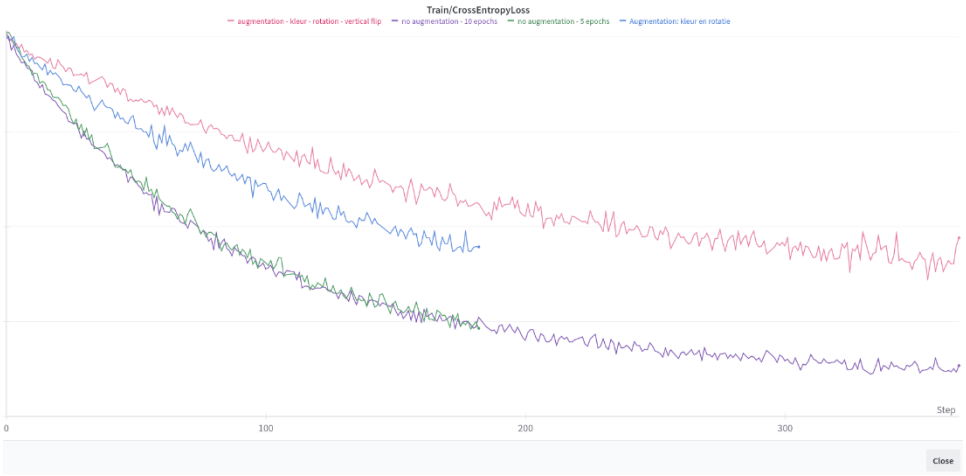
- Grotere leersnelheden**:** Bijvoorbeeld, 0.1, 0.5 - Deze leersnelheden kunnen sneller convergeren, maar ze kunnen ook leiden tot instabiliteit of heel slechte acccuracy.

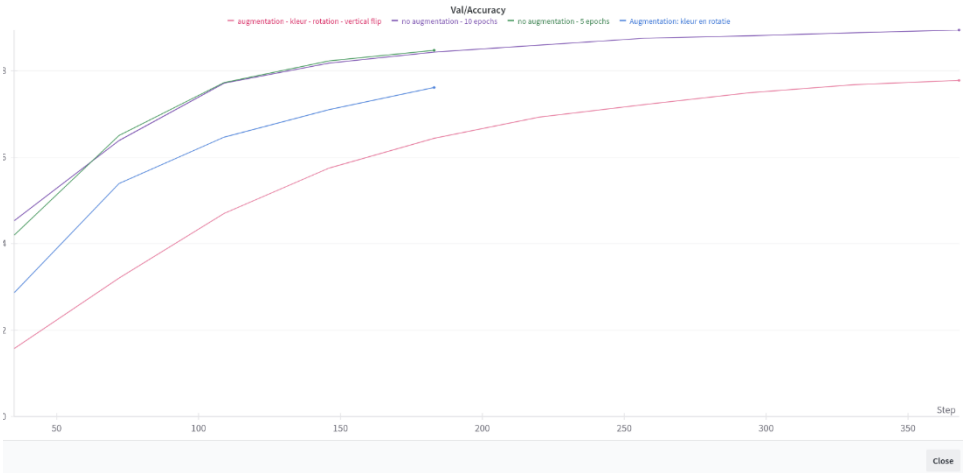
Breid de data transformation pipeline uit met augmentation(s) die jou zinvol lijk(en) voor de gekozen data. Visualiseer het effect van de pipeline op verschillende, willekeurig gekozen afbeeldingen. Train de data eens met en eens zonder de extra augmentations en bediscussieer de resultaten.

Door data augmentatie toe te passen, kan het model robuuster worden in het classificeren van zoogdieren. Als we bijvoorbeeld kleurvervorming gebruiken, leert het model niet alleen te vertrouwen op specifieke kleuren, maar richt het zich meer op de unieke kenmerken van het dier. Hierdoor kan het model dieren herkennen die misschien genetisch een iets andere kleur hebben gekregen, doordat het

meer aandacht besteedt aan de karakteristieke eigenschappen van het dier, in plaats van alleen aan de kleur.


```
color_jitter = v2.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1)
transforms_3 = v2.Compose([
    v2.RandomHorizontalFlip(),
    v2.RandomRotation(30),
    v2.RandomResizedCrop(224, antialias=True),
    color_jitter
])

visualize_df(df_sample, transforms_3)
```



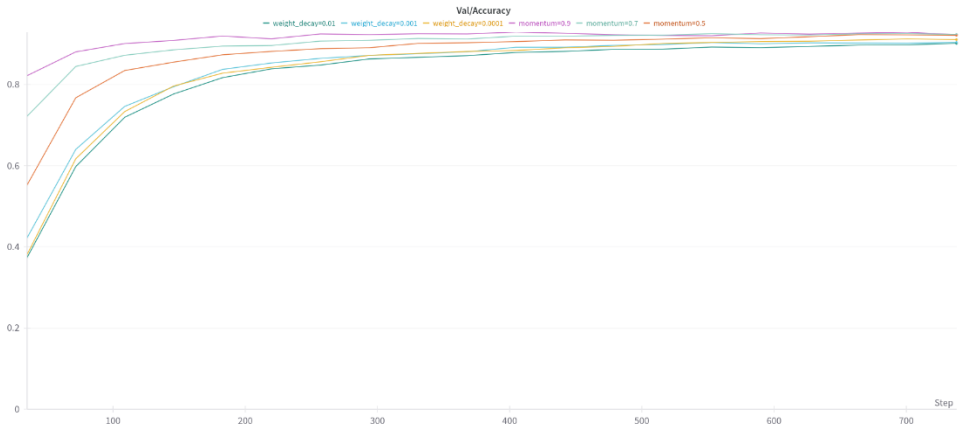


Als er geen augmentatietransformaties worden toegepast, merken we een snellere daling van de lossfunctie en een snellere toename van de accuracy.

Dit kan mogelijk worden verklaard doordat onze validatiedataset geen foto's bevat met augmentaties, waardoor deze beter overeenkomen met de originele dataset zonder augmentaties.
Het vergelijken van de robuustheid die augmentatie biedt, kan pas plaatsvinden wanneer we beschikken over een meer gevarieerde validatiedataset.

Voer de vorige experimenten uit met 5-fold cross-validatie (behalve eerste experiment)
Door 5-fold cross-validatie te gebruiken zal het model beter geëvalueerd worden op verschillende delen van de dataset. De evaluatie zal minder afhankelijk zijn van de toevallige verdeling van de dataset.
De optie die we hier voor gebruiken is --val_fold=None, hierbij gaan we alle folds één keer gebruiken als validatieset.

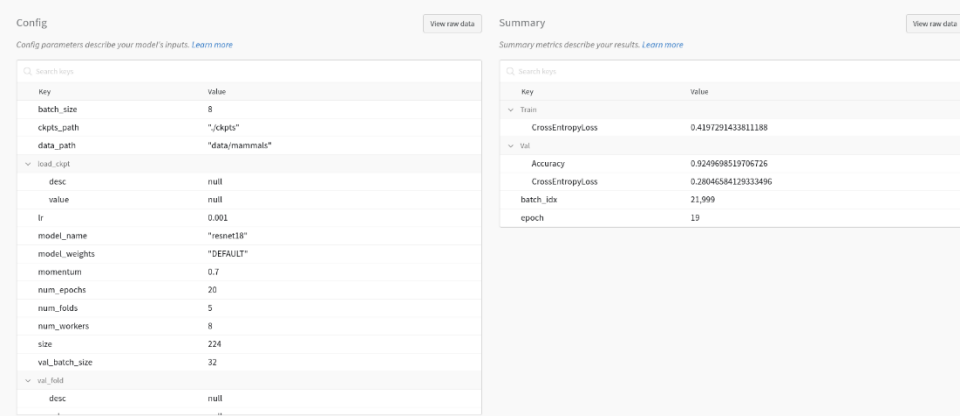
Bepaal de beste hyperparameters na cross-validatie, train een nieuw model met alle folds.
Met een weight_decay van 0.0001 bereiken we de hoogste nauwkeurigheid.
Voor de momentum zie ik dat bij 0.9 de nauwkeurigheid het snelst toeneemt, maar uiteindelijk resulteert een momentum van 0.7 in een ietwat betere nauwkeurigheid. Als je dus met minder epochs traint, is het beter om te kiezen voor 0.9.



- **Evalueer de globale accuracy**

Vervolgens trainde ik een resnet18 met de gevonden hyperparameters, en een batch size die niet al te groot is (8). Ik haalde een accuracy van 92.497%.

Aaron Van Campenhout



- **Evalueer de accuracy van de 5 grootste en 5 kleinste klassen.**

- **Visualiseer de classificaties met hoge confidence die fout zijn en classificaties met lage confidence die juist zijn**

Bonus/malus van max. +/- 2 ptn mogelijk op basis van code
Maak verslag en stuur op als **PDF**, samen met link naar jouw publieke **GitHub repo** (geforked van originele repo) en de link naar jouw publiek **wandb project**