Verslag Schilders voorspellen

(zie README structuur.txt file voor extra verklaring over structuur in inhoud van verschillende notebooks)

Introductie

In dit project werd er als opdracht gegeven een web applicatie te maken die instaat voor het voorspellen door welke schilder een opgegeven schilderij werd geschilderd.

Het doel van deze opdracht was onze vaardigheden met Convolutionele Neurale Netwerken te testen door deze te gebruiken om dit project tot een goed einde te brengen. Het opmaken van een web applicatie geeft ook meteen al een praktisch voorbeeld van het gebruiken van een model die we geschreven hebben. Niet alleen het opmaken van een goed model is dus van belang, maar ook het in een praktische omgeving inzetten.

Omgeving

Ik heb voor de uitvoering van dit project gebruikt gemaakt van de GPU op mijn laptop, een NVIDIA GeForce RTX 2060.

Ik heb gewerkt in PyCharm Professional 2022.2.2.

Data

De schilderijen van Mondriaan, Picasso en Rubens waren gegeven. De schilderijen van Rembrandt en Van Gogh heb ik gescraped van onderstaande links.

- Rembrandt
- Van Gogh

In de setup notebook voerde ik data exploration en cleaning uit. Ik verwijderde hier de duplicaten en de corrupte files.

Hier maak ik ook de willekeurige kopieën van images om de aantallen schilderijen van de verschillende schilders even groot te maken, dit omdat ik gekozen heb om te werken met een oversampled sampling strategie. Ik had ook kunnen de extra schilderijen achterwege laten tot we enzelfde aantal schilderijen hebben in iedere klasse (undersampled methode), deze methode heb ik dan in een latere fase van het project ook uitgetest.

De rest van deze notebook bevat een slideshow van de images. Ook het opsplitsen van de schilderijen in train-, validatie- en testset directories gebeurd hier.

Modellen

Voor het opstellen van mijn modellen, maakte ik gebruik van een Hold-out validation. Als evaluatie metrieken gebruik ik de accuracy van het model, aangezien we te maken hebben met een classificatie probleem.

Eigen CNN

Ik begon met zelf eens een convolutioneel model op te stellen. Ik begon met een model met 3 convolutionele lagen met respectievelijk 32, 64 en 128 filters. Deze lagen worden telkens gevolgd door een MaxPooling2D laag en RMSprop als optimizer.

Met dat begin model bekwam ik een accuracy van 91% hebben en een loss van 0.40 op de testset.

```
48/48 [==============================] - 1s 16ms/step - loss: 0.3953 - accuracy: 0.9065
```

De verdere stappen (zoals data augmentation & dropout toevoegen, …) die ik heb ondernomen staan beschreven in de notebook.

Transfer learning

Ik heb ook met behulp van transfer learning enkele modellen opgesteld. Ik heb gewerkt met VGG16 en Resnet.

De verschillende stappen staan wederom beschreven in de notebook betreffende transfer learning (zie readme.txt voor toelichting file structuur).

Fine-tunen model

Nadat ik mijn model met ResNet had gekozen, heb ik deze wat ge-fine-tuned door te experimenteren met verschillende optimizers en learning rates.

Finale model

Voor mijn finale model maak ik gebruik van een model die gebruik maakt van ResNet50. De laatste 4 lagen van deze convbase laat ik hertrainen. Ik maak geen gebruik van data augmentation, want ik krijg betere resultaten zonder data augmentation.

Als laatste paar layers gebruik ik een Flatten en een Dense met 256 units. Ten slotte gebruik ik nog een laatste Dense laag met 5 units, dit omdat we te maken hebben met 5 klassen om te classificeren.

Ik compileer dit model met als optimizer Adam met een learning rate van 1^{e}-6 (=0.000001).

Verder experimenteren met het gekozen model

Nadat ik mijn finale model heb gekozen probeer ik een inzicht te krijgen in de werking ervan met behulp van het plotten van de tussentijdse activaties die het model maakt. Jammer genoeg ben ik hier niet in geslaagd om dit te doen voor mijn ResNet model, dit is wel gelukt voor een CNN die ik zelf heb opgesteld.

Ik heb mijn model ook eens toegepast op een undersampled dataset. Ik heb zowel zonder als met data augmentation uitgetest op de undersampled dataset.

Het leek me overbodig om de imbalanced methode ook eens uit te testen. Er zijn veel meer schilderijen van Picasso, het model zou dit kunnen gebruiken om gewoon telkens Picasso te voorspellen en zo toch een accuracy van rond de 50% te behalen (wat al een beter zou zijn dan de baseline van 25%).

Vandaar dat ik besloot om deze methode niet uit te testen.

Applicatie

De webapp werd gemaakt met behulp van flask.

base.html is het basisbestand voor de andere html paginas. Zo wordt er duplicate code vermeden.

Ik maak gebruik van een form waarin u de gewenste afbeelding kan uploaden. Nadat u deze hebt ingegeven klikt u op de knop “check painter” en dan verschijnt de naam van de schilder dat het model voorspelt heeft.

In **PaintingPredictor.py** vindt u de klasse PaintingPredictor. Deze bestaat uit de constructor en de methode predict_painter. Wanneer de PaintingPredict klasse geïnstantieerd wordt, wordt de filenaam van het bestand waarin het gewenste model is opgeslagen meegegeven. Op basis van dit model wordt dan op basis van de geüploade afbeelding een voorspelling gemaakt.

Samenvatting & conclusies

In dit project heb ik eerst en vooral data verzameld, dit door middel van scraping. Zo heb ik 2 extra schilders kunnen classificeren in deze opdracht, in totaal maakt dat 5 schilders.

Daarna voerde ik data exploration & datacleaning uit op deze dataset. Hierna verdeelde ik de totale dataset in een train-, validatie- en testset van respectievelijk 60%, 20% en 20%.

Om te beginnen heb ik de modellen die aan bod komen in hoofdstuk 8 van het boek “Deep Learning with Python” van François Chollet uitgeprobeerd. Op basis hiervan begon ik dan met mijn eigen CNN op te stellen. Uiteindelijk besloot ik over te stappen naar transfer learning en heb daarmee dan mijn finaal model opgesteld.

Ik begon met VGG16, maar heb dan overgestapt naar ResNet50, met behulp van ResNet50 heb ik dan ook mijn finale model bekomen door de laatste 4 lagen van de convbase trainable te maken. In dit model maakte ik gebruik van de Adam optimizer met een learning rate van 1^{e}-6. Met dit model bekwam ik een **accuracy van 96%** en een **loss van 0.17** op de testset.

Een conclusie die ik kon maken in deze fase van het project is dat data augmentation niet altijd gelijk is aan een beter resultaat, waar ik voordien nog niet bij stil had gestaan.

Het model heb ik dan geïntegreerd in een simpele webapplicatie waarin de gebruiker een afbeelding van een schilderij kan uploaden en de applicatie geeft dan een voorspelling terug met de naam van de schilder die het hoogstwaarschijnlijk geschilderd heeft.

Gebruikte packages (met versies)

beautifulsoup4	4.11.1
Flask	2.2.2
keras	2.10.0
tensorflow	2.10.0
tensorflow-gpu	2.10.0
scikit-learn	1.2.0
matplotlib	3.6.1
numpy	1.23.3
opencv-python	4.6.0.66
Pillow	9.2.0
requests	2.28.1
urllib3	1.26.12
ipython	8.5.0