2023

Supervised Learning Model

AI EV-BATTERIJEN
TADRAŁA,PIOTR P.P.

O-PP-CMK

INHOUDSOPGAVE

1.0 INLEIDING

Om mezelf meer bekend te maken met het bouwen van ML-modellen, heb ik een model gemaakt dat gebruik maakt van Supervised Learning algoritme en als doel heeft om katten van honden te onderscheiden. Dit document is opgesteld na mijn onderzoek naar "Hoe wordt een AI-model ontwikkeld?" en heeft als doel om de stappen die in het onderzoek zijn besproken in de praktijk te demonstreren. In dit document wordt slechts een eenvoudig model en algoritme getoond. Na het onderzoek naar de volgende deelvraag, "Welke algoritmes zijn het meest geschikt?", zal ik meer experimenteren met verschillende algoritmes.

2.0 DATASETS

Voor het trainen van een Supervised Learning model heb je een gelabelde dataset nodig. Hiervoor heb een handige website gevonden die kant-en-klare datasets heeft kaggle.com. Daar heb ik een perfecte dataset voor mijn model gevonden met in totaal 25.000 foto's van katten en honden.

- Sachin, (2019) Cats-vs-Dogs

3.0 FRAMEWORK

Voor mijn model heb ik gekozen voor TensorFlow, omdat ik vind dat de documentatie en code van TensorFlow veel duidelijker is, vooral voor iemand met weinig kennis van Python ML.

4.0 PREPROCESSING

Een dataset moet eerst gepreprocessed worden, dit wil zeggen dat deze opgesplitst moet worden in twee groepen: de trainings- en validatiesets.

Daarna is het handig om de foto's van de trainings-set te augmenteren. Dit houdt in dat een foto meerdere keren gebruikt kan worden door deze bijvoorbeeld te roteren, horizontaal of verticaal te flippen, de helderheid of het contrast aan te passen, of te verschuiven. Hierdoor ontstaan er variaties van dezelfde foto, wat de dataset vergroot zonder dat je meer foto's nodig hebt.

```
train_dir = 'data/train'
validation_dir = 'data/validation'

# Datasets
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Voor mijn model heb ik voor de training 2000 foto's gebruikt (1000 van katten en 1000 van honden). Door de foto's te augmenteren, komt het totale aantal uit op ongeveer 512.000 foto's.

Vervolgens heb ik twee 'generators' gedefinieerd die afbeeldingen vanuit het aangegeven pad importeren en deze naar een specifieke resolutie scalen voordat ze aan het model worden gevoerd.

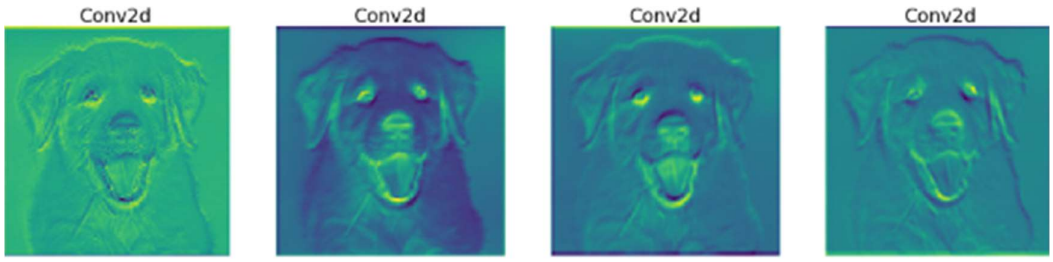
```
# Generators
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=50,
    class_mode='binary'
)
validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(150, 150),
    batch_size=50,
    class_mode='binary'
)
```

5.0 ARCHITECTUUR

Voor mijn model ik 6 layers gebruikt namelijk:

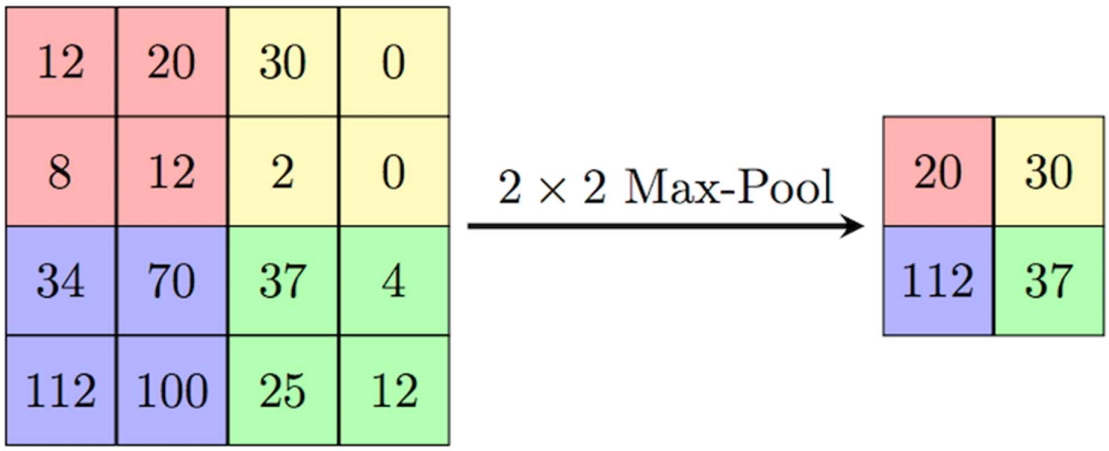
```
model.add(layers.Conv2D( filters: 32,  kernel_size: (3, 3), input_shape=(150, 150, 3)))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense( units: 128, activation='relu'))
model.add(layers.Dense( units: 1, activation='sigmoid'))
```

- **Convo2D:** Werkt als de input layer, neemt images in met resolutie van 150x15 en 3 kleurenkanalen (rgb). Convoluties worden gebruikt om kenmerken in 2D gegevens te herkennen. Output hiervan is een feature map



- **Activation=Relu:** Deze laag zorgt ervoor dat de output van de vorige laag niet-lineair wordt, wat essentieel is voor het leren van complexe patronen in de data.

- **MaxPooling2D:** MaxPooling2D verkleint de feature map door per regio het grootste getal te kiezen, wat data reduceert en belangrijke kenmerken behoudt.



- **Flatten:** De Flatten layer zet een 2D- of 3D-array om in een 1D-array, zodat de data kan worden doorgegeven aan de Dense Layers.

- **Dense,** Een dense layer verbindt elke neuron met alle neuronen uit de vorige layer, wat helpt om complexe relaties in de data te leren. Door de sigmoid functie te gebruiken is de output een binaire classificatie.

6.0 TRAINING

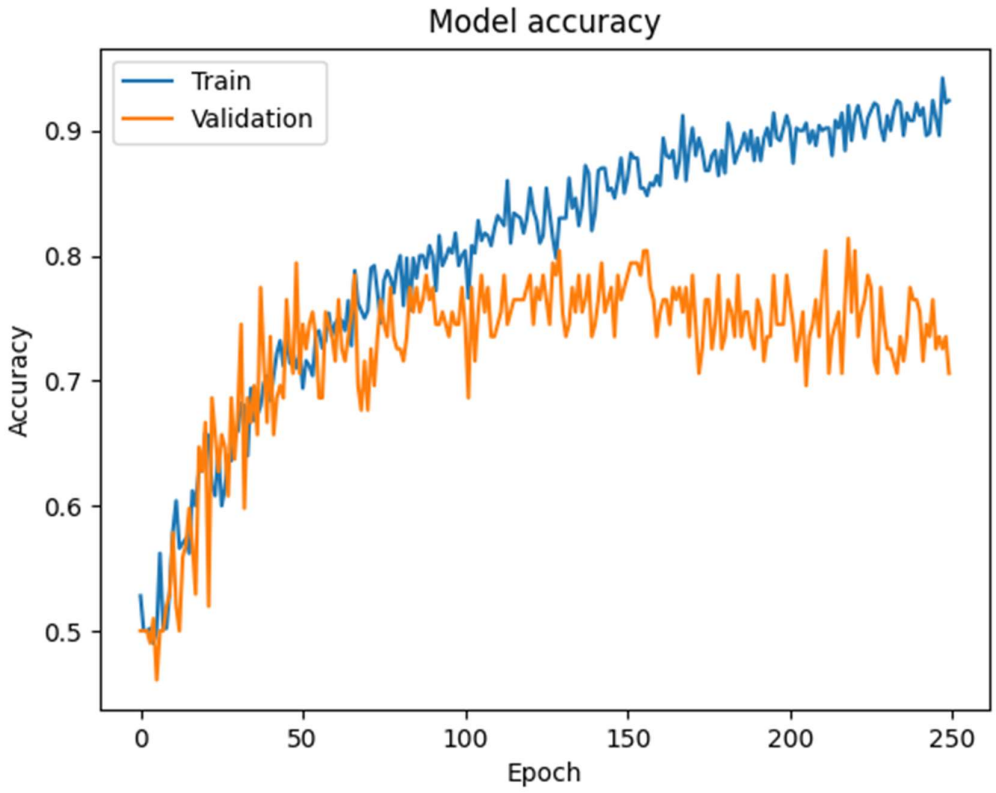
Tot slot wordt het model gecompileerd en getraind. Het Adam-algoritme is gekozen als optimizer omdat die het beste presteert bij CNN-modellen. Voor de loss function is binary_crossentropy gebruikt, omdat deze het meest geschikt is voor binaire classificatie. Daarnaast heb ik aangegeven dat ik de accuracy van het model wil meten.

Het model wordt in iteraties van alle data getraind die epochs heten, zie hooftdstuk 7 voor voorbeelden van resultaten.

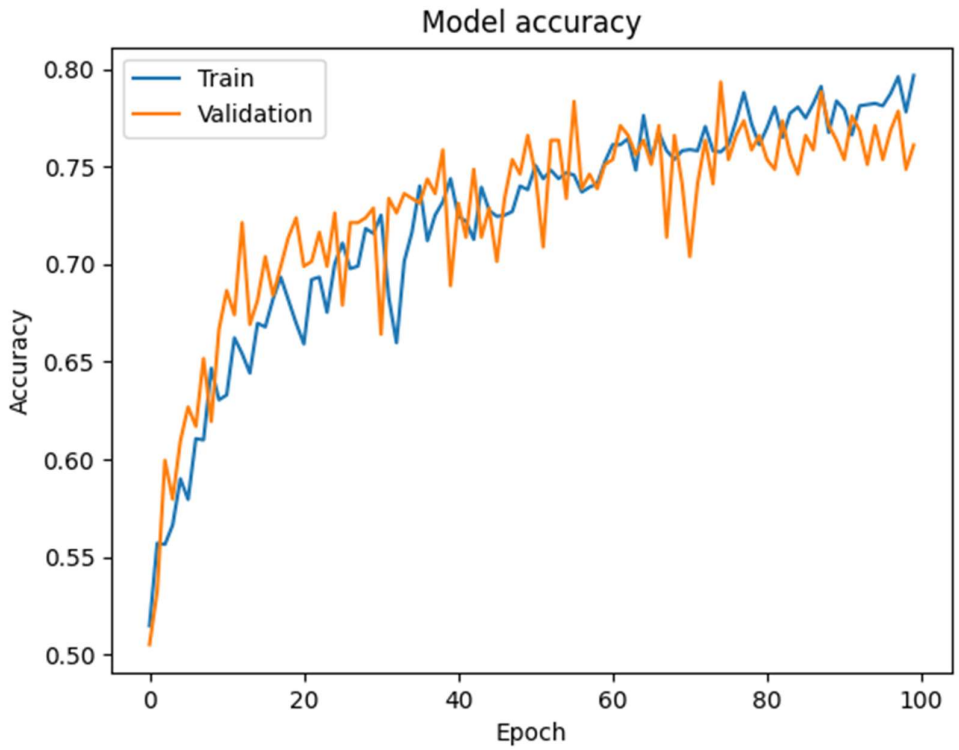
```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

model.summary()

# Training
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator
)
```

7.0 RESULTATEN

Voorbeeld van een model met een dataset van 200 trainingsfoto’s en 50 validatiefoto’s over een periode van 250 epochs. Na 60 epochs is er een duidelijk voorbeeld van overfitting zichtbaar, waarbij het model niet langer algemene kenmerken van afbeeldingen herkent, maar in plaats daarvan specifieke foto's onthoudt. Resultaat hiervan is een lagere nauwkeurigheid bij afbeeldingen die het model nog nooit eerder heeft gezien.



Voorbeeld van een model getraind op 100 epochs van 750 trainingsfoto's en 200 validatiefoto's:



8.0 CODE

https://github.com/Piotr-InforDB/SupervisedLearningModel

9.0 BRONNEN

- Kamal DS, (2023) Deep learning Cat and Dog Classification Using TensorFlow
- TensorFlow, (2024), TensorFlow Keras Layers
- Sachin, (2019) Cats-vs-Dogs