2023

Supervised Classification Model

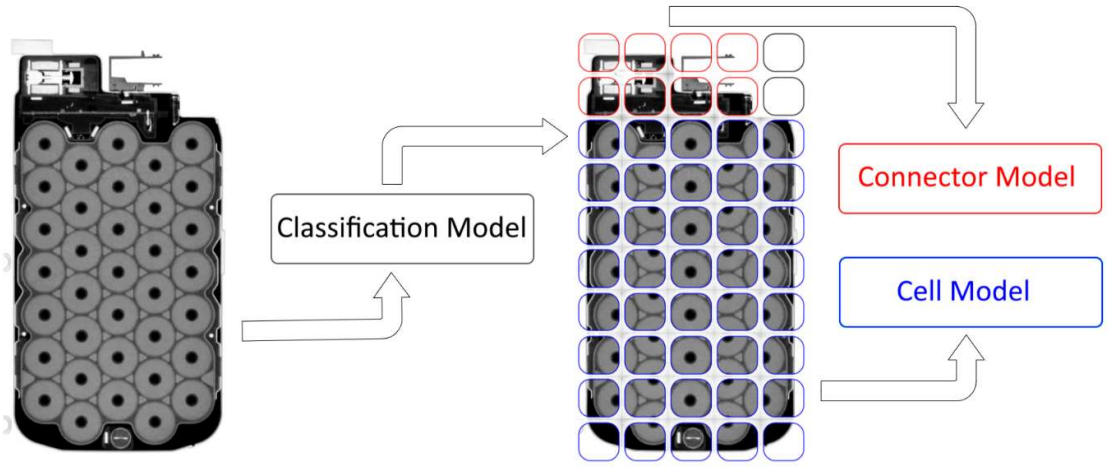
AI EV-BATTERIJEN
TADRAŁA,PIOTR P.P.

O-PP-CMK

INHOUDSOPGAVE

1.0 INLEIDING

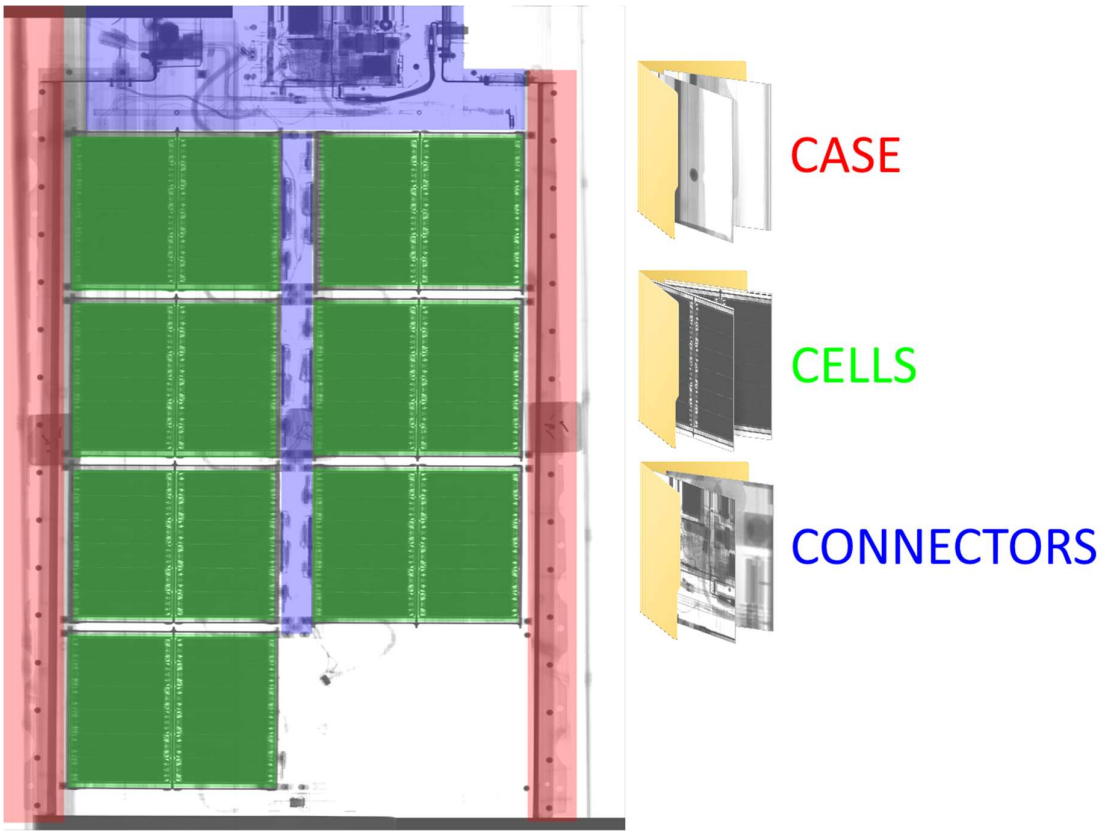
Om het probleem van het gebrek aan data om ons model op te trainen aan te pakken, heb ik een mogelijke oplossing bedacht: component-specifieke modellen. Hierbij wordt een foto van een EV-batterij opgedeeld in een grid, waarbij elk blokje een nieuwe afbeelding is. Deze afbeeldingen worden vervolgens door een model geclassificeerd, bijvoorbeeld als een batterijcel, een connector, case, of gewoon niks.



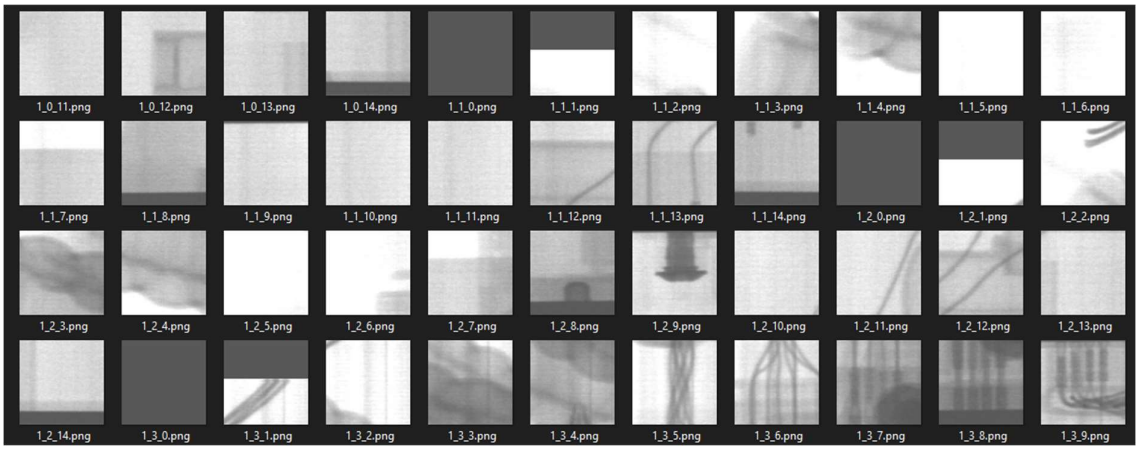
2.0 DATA PREPROCESSING

In eerste instantie heb ik handmatig een afbeelding van een CT-scan opgesplitst in losse afbeeldingen. Hiermee heb ik drie klassen gecreëerd: Case, Cells en Connectors, met in totaal ongeveer 10 afbeeldingen.

Deze afbeeldingen heb ik gelabeld. Het doel is dat het uiteindelijke model kleine samples van bijvoorbeeld 100x100 px gaat herkennen. Daarom heb ik met behulp van een script de handmatig uitgeknipte afbeeldingen verder opgesplitst in kleine blokjes van 100x100 px. Dankzij de hoge resolutie van de CT-scan heb ik in totaal ongeveer 10.000 afbeeldingen kunnen genereren.

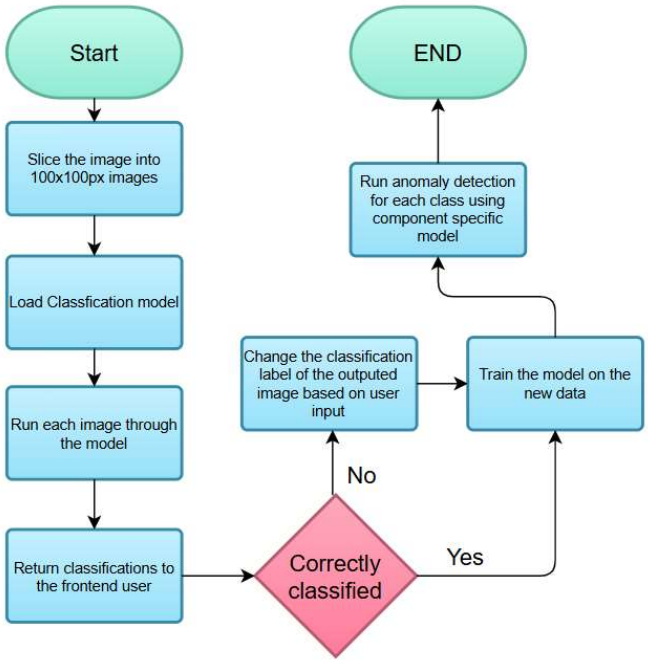


Voorbeeld van de 100x100 blokjes van de connectors.



3.0 FLOW-CHART

Het proces begint wanneer een afbeelding wordt geüpload waarvan de componenten geclassificeerd moeten worden.



4.0 ARCHITECTUUR

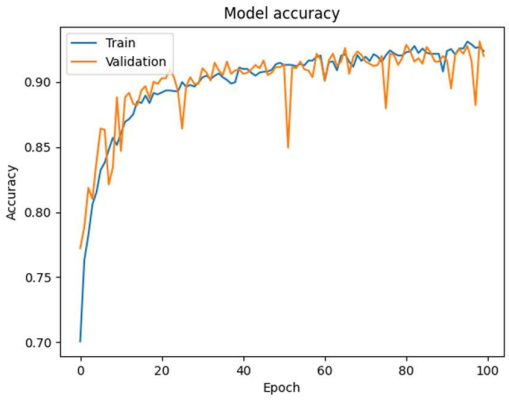
Voor de classificatie heb ik ook gebruikgemaakt van een CNN-model, waarbij de output layer de softmax-activatiefunctie gebruikt. Softmax zorgt voor een 'probability distribution' door de output van elk neuron te schalen, zodat de totale som gelijk is aan 1. Hierdoor kunnen we de waarschijnlijkheid van elke class bepalen.

```python
# Model
model = models.Sequential()

model.add(layers.Conv2D( filters: 32,  kernel_size: (3, 3), activation="relu", input_shape=(100, 100, 3)))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D( filters: 64,  kernel_size: (3, 3), activation="relu"))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D( filters: 128,  kernel_size: (3, 3), 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: 3, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```


- Esra Soylu, (2024) Creating a CNN Model for Image Classification With TensorFlow
- TensorFlow, (2024) Convolutional Neural Network

5.0 RESULTATEN

Dankzij deze aanpak heb ik veelbelovende resultaten kunnen behalen. Door een scan op te splitsen in blokjes van 100x100 px en hiermee een dataset van 10.000 afbeeldingen te genereren, heb ik een classificatie-accuratesse van 93% bereikt.



6.0 CODE

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

7.0 BRONNEN

- Esra Soylu, (2024) Creating a CNN Model for Image Classification With TensorFlow
- TensorFlow, (2024) Convolutional Neural Network