LISTING UITLEG

```
> Start nieuwe regel
regel met spatie na afbreking
regel zonder spatie na afbreking
```

UITVOER

De uitvoer van de code wordt als volgt weergegeven:

```
output 12345
```

kan zelfs voor een groot deel op de Edge TPU zelf, waardoor dit vrij snel gebeurt. Daarvoor heb je wel niet het originele model nodig, maar het originele model zonder de laatste laag. Die vind je op de pagina met modellen onder het kopje “On-device backpropagation (classification)”.

BLOEMEN HERKENNEN

We gaan nu MobileNet, dat 1000 algemene objecten herkent, hertrainen zodat het specifiek diverse soorten bloemen herkent. Maak eerst een directory aan voor je project:

```
> $ DEMO_DIR=$HOME/flowers
> $ mkdir -p $DEMO_DIR
> $ cd $DEMO_DIR
```

Download een dataset met afbeeldingen van bloemen en pak ze uit:

```
> $ wget http://download.tensorflow.org/example_images/flower_photos.tgz
> $ tar xvzf flower_photos.tgz
```

En download dan het MobileNet-model zonder de laatste laag:

```
> $ wget https://github.com/google-coral/edgetpu/raw/master/test_data/mobilenet_v1_1.0_224_quant_embedding_extractor_edgetpu.tflite
```

De code voor het hertrainen vind je in de voorbeeldcode van Google, die je ook eenvoudig installeert:

```
> $ sudo apt install edgetpu-examples
```

Je vindt de voorbeeldcode dan in /usr/share/edgetpu/examples/. Daarna kun je het programma opstarten om het model te hertrainen:

```
> $ python3 /usr/share/edgetpu/examples/backprop_last_layer.py \
>   --data_dir ${DEMO_DIR}/flower_photos \
>   --embedding_extractor_path \
>  ${DEMO_DIR}/mobilenet_v1_1.0_224_quant_embedding_extractor_edgetpu.tflite \
>   --output_dir ${DEMO_DIR}
```

Het programma toont je welke afbeeldingen het allemaal inleest: 600 tot 900 afbeeldingen per bloemsoort, voor madeliefje, tulp, paardebloem, roos, zonnebloem. Na nog geen minuut is het hertrainen gebeurd en vind je het model in `/home/pi/flowers/retrained_model_edgetpu.tflite` en de bijbehorende labels in `/home/pi/flowers/label_map.txt`.

Probeer nu uit of je model deze soort bloemen herkent. Download bijvoorbeeld een willekeurige foto van een zonnebloem en laat deze classificeren:

```
> $ curl -o ${DEMO_DIR}/rose.jpg https://images.pexels.com/photos/1214259/pexels-photo-1214259.jpeg
> $ python3 /usr/share/edgetpu/examples/classify_image.py \
>   --model ${DEMO_DIR}/retrained_model_edgetpu.tflite \
>   --label ${DEMO_DIR}/label_map.txt \
>   --image ${DEMO_DIR}/rose.jpg
> ----------
> sunflowers
> Score :  0.92578125
```

Op dezelfde manier kun je een model trainen voor andere classificaties. Maak gewoon voor elk type object dat je wilt herkennen een map aan met daarin voorbeeldafbeeldingen.

OBJECTEN HERKENNEN

Met het juiste model (MobileNet SSD COCO of Faces) kun je ook objecten en hun locatie in afbeeldingen herkennen. Dat kan als volgt:

```
> $ python3 /usr/share/edgetpu/examples/object_detection.py \
>   --model /usr/share/edgetpu/examples/models/ssd_mobilenet_v2_coco_quant_postprocess_edgetpu.tflite \
>   --label /usr/share/edgetpu/examples/models/coco_labels.txt \
>   --input JEAFBEELDING.JPG \
>   --output ${HOME}/object_detection_results.jpg
----------
person
score = 0.9921875
box = [560.239709854126,
349.3849046230316,
4086.211227416992,
2406.5477085113525]
----------
surfboard
score = 0.12109375
box = [3524.095703125,
35.26763463020325,
5210.838684082031,
2934.12105345726]
----------
cell phone
score = 0.08984375
box = [3595.1825637817383,
1301.93909740448,
3679.4541091918945,
1411.7773611545563]
----------
potted plant
score = 0.06640625
box = [4969.983730316162,
1449.5367743968964,
5241.2756690979,
1829.9384543895721]
```

Dit was een eenvoudige foto van mezelf die met wat elektronische onderdelen aan het knutselen was. Die onderdelen kent het model niet, en daarom zie je hier enkele vreemde objecten ‘herkend’: een surfbord, smartphone en potplant. Geen van die objecten was in de foto aanwezig, maar de persoon werd wel met meer dan 99% zekerheid herkend. Je ziet dat je de uitvoer van het model nog moet filteren: alles onder een zelfgekozen drempelwaarde voor de score laat je het best weg.

Op dezelfde manier kun je ook gezichten in foto’s herkennen.

```
> $ python3 /usr/share/edgetpu/examples/object_detection.py \
>   --model /usr/share/edgetpu/examples/models/ssd_mobilenet_v2_face_quant_postprocess_edgetpu.tflite \
>   --input JEAFBEELDING.JPG \
>   --output ${HOME}/object_detection_results.jpg
----------
score = 0.9921875
box = [2051.762745976448,
240.68063085526228,
2762.9795892238617,
792.7388074696064]
----------
score = 0.98046875
box = [1316.6589984893799,
27.02611466497183,
2043.4382076263428,
591.1641841232777]
----------
score = 0.69140625
box = [1650.4731440842152,
909.7686061859131,
2577.635899603367,
1638.9171080589294]
----------
score = 0.5
box = [551.844685614109,
754.4507395029068,
1434.9394246935844,
1454.9225406050682]
```

Het model MobileNet SSD Faces detecteert menselijke gezichten.

Dankzij de Google Coral USB Accelerator komen krachtige AI-toepassingen binnen handbereik

Je ziet hier dat je geen labels moet opgeven, omdat je geen verschillende klassen objecten detecteert met het model MobileNet SSD Faces, maar slechts één klasse: gezichten. Het resultaat voor de foto die we opgaven zijn vier gezichten, waarvan er twee met vrij hoge waarschijnlijkheid en twee andere met iets minder waarschijnlijkheid werden gedetecteerd.

MEER WETEN?

Er zijn vele voorbeeldprogramma’s in Python te vinden voor de Edge TPU. Bekijk zeker eens de officiële pagina met voorbeelden (**https://coral.ai/examples/**). Maar ook op GitHub vind je allerlei interessante third-party projecten. Je kunt ook realtime de beelden van een Raspberry Pi Camera Module analyseren. Dankzij de Google Coral USB Accelerator komen krachtige AI-toepassingen binnen handbereik van de Raspberry Pi waarvoor je vroeger toegang tot krachtige servers in de cloud nodig had.