Eindopdracht Beeldverwerking en Computer Vision
COVID-19 face mask detector

Student: Merry Ashji
Klas: B
Datum: 26/03/2022

Inhoudsopgave

Inleiding

Vanaf 1 december 2020 is het dragen van een mondkapje voor iedereen vanaf 13 jaar verplicht in het openbaar vervoer en publieke binnenruimtes. Het dragen van een mondkapje beschermt ons en anderen tegen de verspreiding van het coronavirus. Het is een van de maatregelen die ons samen beschermen tegen het virus. Daarom is het van belang dat iedereen zich aan de regels houdt en een mondkapje draagt. Om te kunnen controleren of mensen een mondkapje dragen hebben we een systeem nodig. Face mask detector is het onderwerp dat ik heb voor de eindopdracht van het vak vision. Dit systeem controleert of de mensen (op foto's) een mondkapje dragen. Dit systeem kan handig worden gebruikt op het station, op school, in de bibliotheek of in de trein.

Requirements

Om het project te kunnen runnen en testen zijn er een paar libraries nodig:

- Tensorflow
- Opencv
- Os
- Random
- Numpy
- Matplotlib

Dataset

De dataset die ik gebruik om face mask detector te trainen wordt door Prajna Bhandary gemaakt. De dataset bestaat uit 1376 foto's (690 met mondkapjes, 686 zonder mondkapjes).
Zij heeft de facial landmarks gedetecteerd op foto's van mensen zonder mondkapje en dan wordt met een Python script de foto van het mondkapje op de kin en neus gezet.

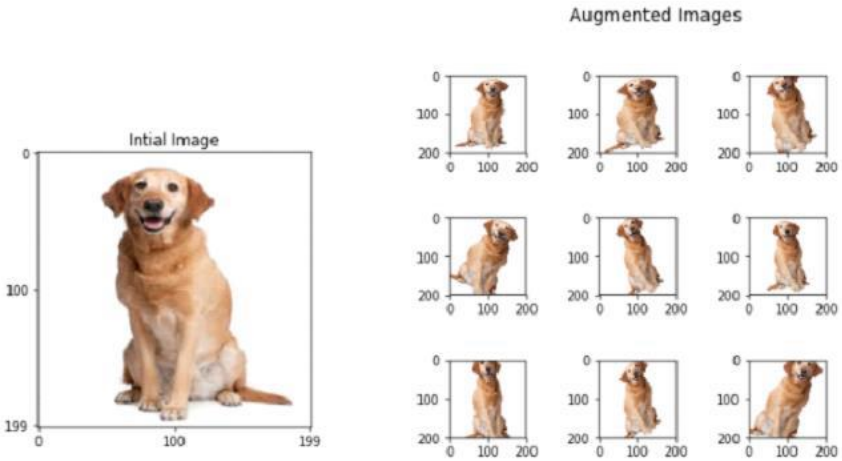
Link naar de dataset: https://github.com/prajnasb/observations/tree/master/experiements

Code

- ## Data preprocessing
    1. Ik heb een functie data_preprocessing() gemaakt. Met die functie wordt de data in een array gezet en klaar gemaakt om getraind te worden. Binnen die functie heb ik deze functies gebruikt:
        - load_img
        - img_to_array
        - preprocess_input

- ## Training dataset voorbereiden om te kunnen trainen

1. Train_test_split: Ik heb de data gesplitst, 30% voor testen en 70% voor trainen. Dus 963 voor trainen en 413 voor testen. Ik heb het op deze manier gesplitst omdat ik een relatief kleine dataset heb (dataset < 10000).
2. ImageDataGenerator: ik heb de keras ImageDataGenerator() functie gebruikt om mijn dataset uit te breiden en mijn model beter te kunnen trainen. Het itertion train is het aantal train data gedeelte door batch_size. De batch size is default (32) en het aantal train data is 963. Dus in elk epoch worden 30 foto's getraind. De ImageDataGenerator functie past transformatie toe (rotation_range, zoom_range, width_shift_range, height_shift_range, shear_range, horizontal_flip, vertical_flip en fill_mode) en gebruikt de getransformeerde foto's voor training. Het aantal getrainde foto's is gelijk in elk epoch maar met random transformatie.



CNN maken om de dataset te kunnen trainen

Om mijn dataset te kunnen trainen heb ik gekozen om mobeilNet2 en 5 layers te gebruiken. Hier is het basismodel voor het gebruiken van mobeilNet2 en 5 layers:

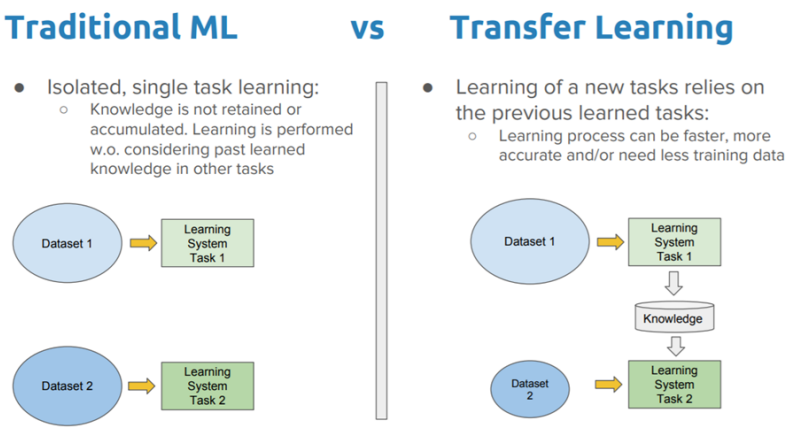
```
block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)   0           ['block_16_depthwise_BN[0][0]']

block_16_project (Conv2D)       (None, 7, 7, 320)   307200      ['block_16_depthwise_relu[0][0]']

block_16_project_BN (BatchNorm  (None, 7, 7, 320)   1280        ['block_16_project[0][0]']
alization)

Conv_1 (Conv2D)                 (None, 7, 7, 1280)  409600      ['block_16_project_BN[0][0]']

Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)  5120        ['Conv_1[0][0]']

out_relu (ReLU)                 (None, 7, 7, 1280)  0           ['Conv_1_bn[0][0]']

==================================================================================================
Total params: 2,257,984
Trainable params: 2,223,872
Non-trainable params: 34,112
```

In de volgende alinea leg ik uit waarom ik voor mobielNet2 gekozen heb, wat de vijf layers zijn en waarom die layers nodig zijn.

- ## Transfer learning using mobileNet2

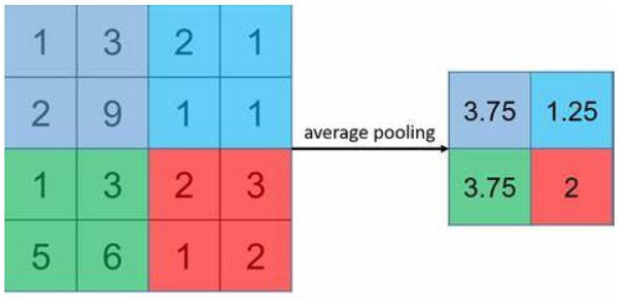
  Transfer learning is een populaire methode bij machine learning, het is hergebruiken van een pre-trainend model voor een nieuw taak**.** Hoe meer de taken op elkaar lijken, hoe meer de kennis van de ene bruikbaar zal zijn voor de andere. Een model dat een goed onderscheid kan

maken tussen honden en katten, kan snel leren om ook een onderscheid te maken tussen leeuwen en tijgers.
Ik heb deze methode gekozen omdat het sneller is, er relatief weinig data en weinig computerkracht nodig is om het model de nieuwe taak te laten uitvoeren.



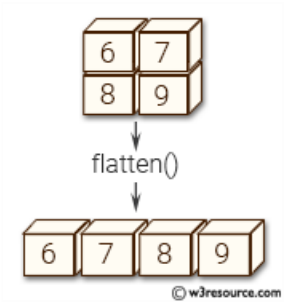
- AveragePooling2D

Pooling layers vermindert de dimensionaliteit van elke feature map, maar tegelijkertijd zorgt het ervoor dat alle belangrijke informatie behouden blijft. Pooling layers kunnen van verschillende typen zijn: MaxPooling, AveragePooling, MinPooling etc. Ik heb de AveragePooling gebruikt om de foto's gladder te maken.



- Flatten

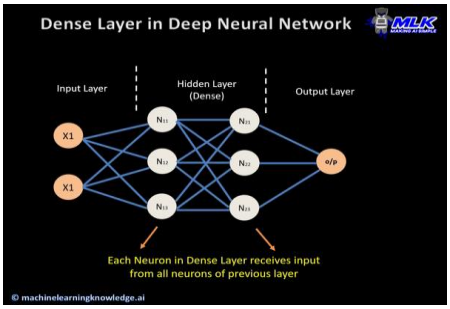
De flatten functie is het converteren van de data naar een 1-dimensionale array om deze in de volgende layer als input te gebruiken.
Ik heb de flatten functie gebruikt om van een multi-dimensional input een 1-dimensionaal array te maken. Dat is nodig voor de volgende layer want de input van de volgende layer moet een 1-dimensionaal array zijn.



- ## Dense layer

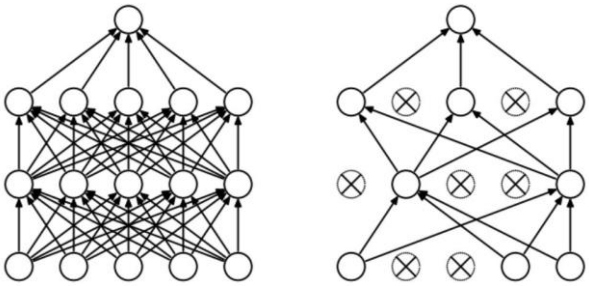
  Dense layer is een neural network layer die diep verbonden is, dat betekent dat elke neuron in de dense layer input ontvangt van alle neurons van de vorige layers. Dense layer is het meest gebruikte layer. Ik heb twee dense layers gebruikt om neural networks te bouwen.

  1. Dense(128, activation='relu') heb ik als unit 64 gekozen want met 128 krijg ik minder loss. Ik heb ReLu activation function gebruikt om alle negatieve waarden uit de gefilterde foto te vervangen met nul.
  2. Dense(2, activation='softmax') heb ik als unit 2 gekozen omdat ik alleen twee outputs heb (with mask, without mask) en ik heb softmax activation function gebruikt om de verschillen te vergroten door te normaliseren.



- ## Dropout:

  Dropout is een layer in cnn en wordt gebruikt om de overfitting te voorkomen. Ik heb het gebruikt om de accuracy toe te laten nemen. Ik heb de default (0.5) value gebruikt.


(a) Standard Neural Net (b) After applying dropout.

Hier is het model na het gebruiken van mobeilNet2 en 5 layers dat ik gebruikt heb om de data te trainen.

```
out_relu (ReLU)                 (None, 7, 7, 1280)   0           ['Conv_1_bn[0][0]']

average_pooling2d_13 (AverageP  (None, 1, 1, 1280)  0           ['out_relu[0][0]']
ooling2D)

Flatten (Flatten)               (None, 1280)         0           ['average_pooling2d_13[0][0]']

dense_26 (Dense)                (None, 64)           81984       ['Flatten[0][0]']

dropout_13 (Dropout)            (None, 64)           0           ['dense_26[0][0]']

dense_27 (Dense)                (None, 2)            130         ['dropout_13[0][0]']

==================================================================================================
Total params: 2,340,098
Trainable params: 2,305,986
Non-trainable params: 34,112
```

Experimenten

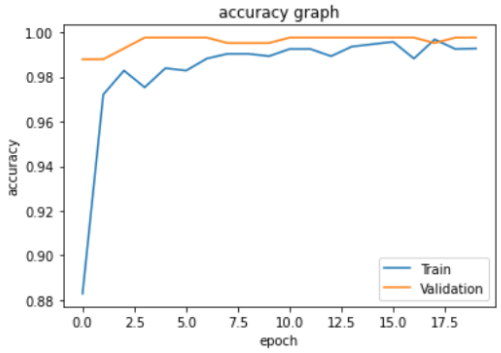
- ## Accuracy

  Na het compilen heb ik de model.evaluate functie gebruikt om het model te evalueren. Die functie geeft de loss en de accuracy weer.
  De loss is 0.0087 en de accuracy is 0.9976.

- ## Accuracy and Loss Graph

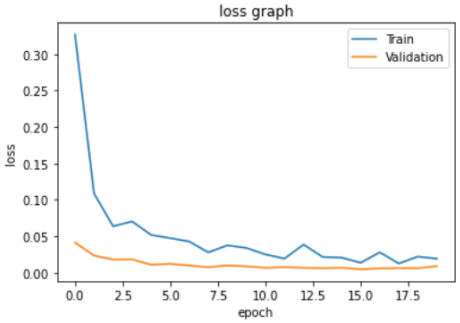
  Ik heb de history of accuracy en de loss geplot en uit de graph blijkt dat het model goed is getraind.

  1. Accuracy graph



  De accuracy en validation accuracy neemt toe.

  2. Loss graph



  De loss en de loss validation neemt af.

- ## Test code

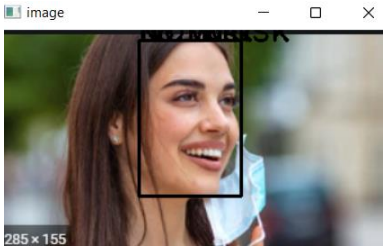
  Ik heb het caffe_model_for_dace_detection-master model gebruikt om het gezicht te detecteren. Het model is te vinden in de github repository.
  Na het detecteren van het gezicht heb ik het model dat ik in de vorige stappen heb getraind, gebruikt om te detecteren of het gezicht met mondkapje of zonder mondkapje is.
  Ik heb een test code om mijn getrainde model te kunnen testen. Ik heb de predict functie gebruikt en op basis van het resultaat van predict wordt bepaald of het gezicht met face mask is of niet.

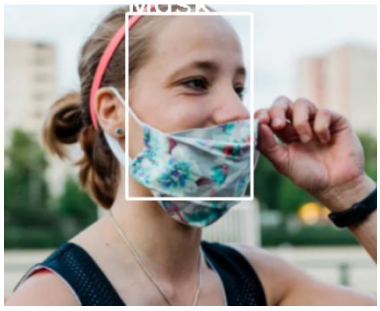
- ## Run test code

  1. Open de test data file.
  2. Run the jupyter notebook face_mask_detector_test_versie1.
  3. Je krijgt een melding om één van de gegeven foto te kiezen.
  4. Kies één van de foto's.

- ## Voorbeelden

  Ik heb mijn model op 24 foto's getest. 12 foto's met mondkapjes en 12 foto's zonder mondkapje. De foto's zijn met verschillende mondkapjes design en verschillende kleuren. Ik laat een paar voorbeelden zien.





Conclusie

- De test code is afhankelijk van twee modellen. De eerste is caffe_model_for_dace_detection-master. Dat model is verantwoordelijk voor het detecteren van het gezicht. Het tweede model is face_mask_detector_model. Dat is het model dat ik getraind heb in de vorige stappen en dat model is verantwoordelijk voor het detecteren van face mask bij het gedetecteerd gezicht in een foto. Dus om te kunnen bepalen of een persoon een mondkapje draagt, moeten we eerst het gezicht kunnen detecteren. Als er geen gezicht wordt gevonden, kan de face mask detector niet worden toegepast. Dat was dus het probleem bij voorbeeld nummer 5.

- Ik heb het model alleen met foto's met één gezichtsuitdrukking getest, dus een aanbeveling voor mensen die er verder aan zouden willen werken is om de test-code uit te bereiden zodat het model met foto's met meerdere gezichtsuitdrukkingen getest kan worden.

- Voor mensen die er verder aan zouden willen werken is het nuttige als het model real time kan werken en getest kan worden.

Bronnenlijst

- Prajna Bhandary (2020, march) link naar de dataset https://github.com/prajnasb/observations/tree/master/experiements

- Shreenidhi Sudhakar (2017, 10 July) Custom Image Augmentation https://towardsdatascience.com/image-augmentation-14a0aafd0498

- Cem Dilmegani (2020, 5 July) Transfer Learning in 2022: What it is & how it works https://research.aimultiple.com/transfer-learning/

- NumPy Array manipulation: ndarray.flatten() function https://www.w3resource.com/numpy/manipulation/ndarray-flatten.php

- Onbekende Gebruiker (2017, 2 februari) Keras Dense Explained for Beginners https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/

- Govinda Dumane (2020, 2 march) Introduction to Convolutio Neural Network (CNN) using tensorflow https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83

- Onbekende Gebruiker (2017,19 oct) Recent Advances in Convolutional Neural Networks https://arxiv.org/pdf/1512.07108.pdf

- Ujjwalkarn (2016, 11 August) An Intuitive Explanation of Convolutional Neural Networks https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

- Adithya Challa (onbekend) What is a ReLU layer? https://www.educative.io/edpresso/what-is-a-relu-layer

- Daniel Matthias, Chidozie Managwu (January 2021) FACE MASK DETECTION PAPER (1) https://www.researchgate.net/publication/348870477_FACE_MASK_DETECTION_PAPER_1

Bijlage

1. De student kan vision-algoritmes implementeren ten behoeve van een geselecteerde Vision taak.
   Ik heb een vision taak gekozen (face mask detection) en ten behoeve van deze taak heb ik een model neuraal netwerk gebouwd. Om het model te kunnen trainen heb ik transfer learning met mobielNet2 en 5 layers geïmplementeerd. (CNN maken om de dataset te kunnen trainen).

2. De student kan papers uit de wetenschappelijke literatuur over computer vision lezen, begrijpen en omzetten naar code.
   Ik heb een challenge gekozen. Ik heb de juiste dataset gebruikt om het model te trainen en een test data set verzameld om het getraind model te testen. Om het probleem te kunnen oplossen heb ik een paar papers gelezen en meerdere methodes uitgeprobeerd.

3. De student kan een opdracht analyseren en op basis hiervan het benodigde werk (waaronder het implementeren) verdelen en plannen.
   Ik heb een planning gemaakt en de taken verdeeld zodat ik op tijd een werkend product kan inleveren.

4. De student kan omgaan met softwareoplossingen voor versiebeheer (GIT) en kan deze software inzetten voor het ontwikkelen van code.
   Voor de eindopdracht heb ik een Git repository gemaakt. Daar heb ik de dataset, test data, requirements, planning, face mask detector, caffe_model_for_dace_detection-master, face_mask_detector_model en het verslag ingeleverd. In de readme staat er een korte uitleg over het project.
   Hier is een link naar GIT https://github.com/Merryashji/Beeldverwerking-en-Computer-Vision

5. De student kan een experiment opzetten en uitvoeren om aan te tonen hoe de geïmplementeerde vision-methode werkt, hoe effectief (zowel in kwaliteit als in snelheid) deze methode en implementatie is en waar grenzen liggen van de gekozen methode en implementatie.
   Ik heb de accuracy en de loss gemeten en het model geëvalueerd. Daarna Ik heb de history of accuracy en de loss geplot (accuracy, accuracy en loss graph).

6. De student kan een rapport schrijven om het gekozen en uitgevoerde experiment te beschrijven, de resultaten te beschrijven en conclusies te trekken.
   Ik heb een verslag geschreven over de gekozen methoden en het uitgevoerd werk. Binnen het verslag heb ik alle methoden die ik gekozen heb uitgelegd en waarom het nodig was voor het project.

7. De student kan een zelf-opgezet experiment uitvoeren en de resultaten op gestructureerde wijze verzamelen.
   Ik heb een test-code geschreven waarmee de gebruiken het getraind model kan testen. Ik heb in mijn verslag een paar voorbeelden laten zien. Ik heb het resultaat van het model behandeld en een paar ideeën gegeven voor degene die er verder aan wil werken. (Experimenteren).