Projekt Objekterkennung

Deep Learning Traffic Sign Detection:

Geschwindigkeitsbegrenzungszeichen WS 22/23AI

Gruppe 6:

- Leon Walter, 710379
- Kevin Gottbrecht, 710378
- Reinmar Finkler, 710380
- Alban Becker,
- Michael Stadie, 710782
- Vigoleis Hühn, 690452

Inhaltsangabe

1. Aufgabenteilung S.2
2. Aufgabe 1: ohne Augmentierung S.3
3. Aufgabe 2: mit Augmentierung S.5
4. Aufgabe 3: direkte Klassifikation S.8

Aufgabenteilung:

Aufgabe 1+3: Alban Becker, Vigoleis Hühn
Aufgabe 2: Leon Walter, Reinmar Finkler
Dokumentation: Kevin Gottbrecht, Michael Stadie

Erstes Netz

Im Schritt Eins sollen wir mithilfe des ResNet50 die Region-Detection ausführen und der DigitTrainDataStoreAugmented das Kategorisieren vornehmen. Anschließend die Genauigkeit angegeben werden. Die erzielten Schritte, werden später mit dem zweiten Teil dann erkannt.

Das ResNet-50

ResNet-50 ist ein Convolutionales neurales Netzwerk, dass 50 verschiedene Ebenen umfasst. Es erhält einen Input der Größe von 224 zu 224. Es kann viele verschiedene Objekte erkennen und wird hier dazu genutzt, in den Bildern die Verkehrszeichen zu ermitteln. Diese werden dann im folgenden Schritt Zwei erkannt und genauer ausgegeben.

Umsetzung im Programm code

Im ersten Teil werden die Daten eingelesen. Dabei zum einen die Bilddateien im DataDir, die Daten von TA und die fignDatasetGroundTruth.mat. Diese werden dann mit der Func_setupData aufgeräumt und die Grounddata erstellt falls noch nicht existent.

```
dataDir = 'Picturedata'; % Destination-Folder for provided (img) Data
zippedDataFile = 'PicturesResizedLabelsResizedSignsCutted.zip'; %Data provided by TA
grDataFile = 'signDatasetGroundTruth.mat';
func_setupData(dataDir, zippedDataFile, grDataFile);
```

Nachdem die Daten geladen wurden, werden Sie zufällig aufgeteilt in einen Anteil von 60% für die Trainingsdaten, einen weiteren von 10% zum Bestätigen und die Restlichen Daten zum Testen des Trainierten Detektoren. Danach werden sie benannt und mit dem Label verknüpft.

```
rng(0)
shuffledIndices = randperm(height(traficSignDataset));
idx = floor(0.6 * height(traficSignDataset));

trainingIdx = 1:idx;
trainingDataTbl = traficSignDataset(shuffledIndices(trainingIdx),:);
```

```
imdsTrain = imageDatastore(trainingDataTbl{:,'imageFilename'});
bldsTrain = boxLabelDatastore(trainingDataTbl(:,'sign'));
```

```
trainingData = combine(imdsTrain,bldsTrain); % erzeugt 'CombinedDatastore
```

Nach Vollendung der Vorbereitung wird eine der Trainingsdaten Dargestellt.

```
while 1==0  %hasdata(preprocessedTrainingData)
    data = read(preprocessedTrainingData);
    I = data{1};
    bbox = data{2};
    annotatedImage = insertShape(I,'Rectangle',bbox);
    annotatedImage = imresize(annotatedImage,4);  % nur fuer Darstellung
    figure(1)
    imshow(annotatedImage)
    pause(0.100)
end
```

Zu Beginn werden die Anchors-Boxen definiert und ausgewählt. Und das Resnet50 Netzwerk ausgewählt, der Layer ausgewählt und die Funktion ausgewählt. Mit Hilfe der FasterRCNNLayers, werden die Positionen der Anchor-Box ermittelt. Wenn das Augmentieren erfolgen soll wird im nächsten Stepp die daten hierfür erweitert und transformiert.

```
if doAugmentation

    augmentedTrainingData = transform(trainingData,@augmentData);
    trainingData = transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
    validationData = transform(validationData,@(data)preprocessData(data,inputSize));

end
```

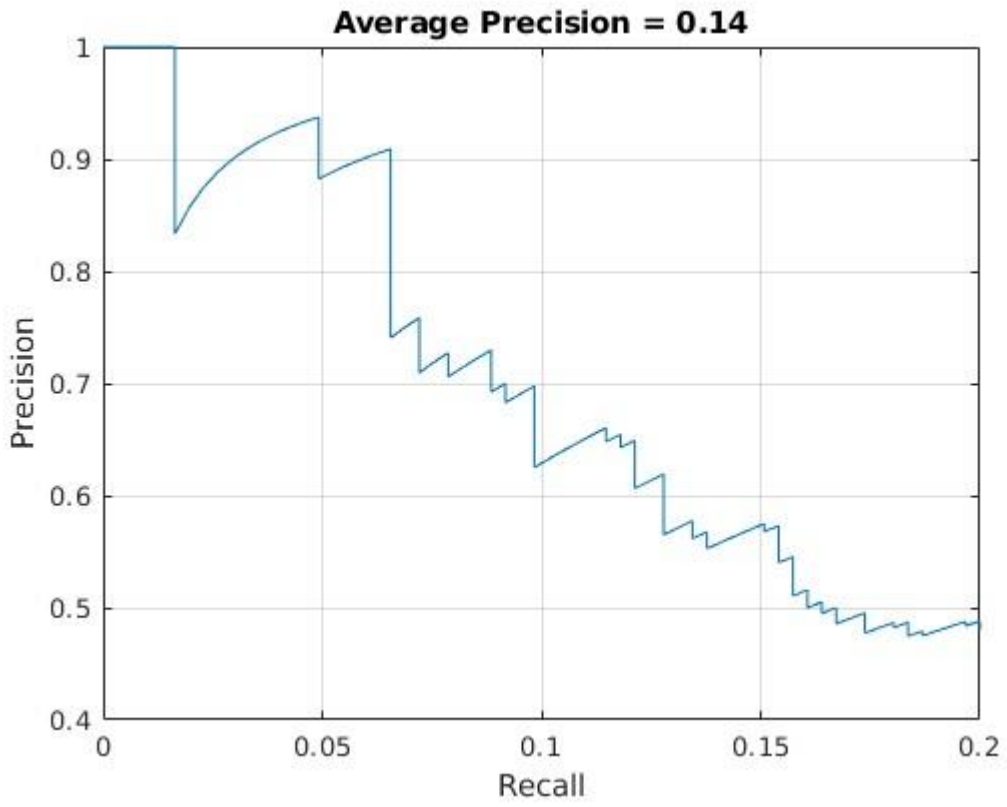
Beim Training werden mit der Funktion trainFasterRCNNOBjectDetector unser Netzwerk trainiert. Dazu geben wir der Funktion die Trainingsdaten, unser Netzwerk (lgraph) und die weiteren Optionen. Dies sind in diesem Fall die Negative bzw. positive Überlappungsweite. Und nachdem Training soll das trainierte Netzwerk. Sollte diese Einstellung falsch sein, lädt es einfach das gespeicherte Netz.

```
if doTraining
    % Train the Faster R-CNN detector.
    % * Adjust NegativeOverlapRange and PositiveOverlapRange to ensure
    %   that training samples tightly overlap with ground truth.
    [detector, info] = trainFasterRCNNObjectDetector(trainingData,lgraph,options, ...
        'NegativeOverlapRange',[0 0.3], ...
        'PositiveOverlapRange',[0.6 1]);
    save netname detector;
else
    % Load pretrained detector for the example.
    load netname detector;
end
```

Im Folgenden werden nun der eigentliche Vorgang gestartet. Zuerst werden die Größen überprüft, im Anschluss diese transformiert und so in das benötigte Format gebracht. Dann wird der Detektor angewendet auf alle Daten. Die Resultate werden auf Präzision geprüft und ausgegeben.

```
% ----- Testing

testData = transform(testData,@(data)preprocessData(data,inputSize));

% Run the detector on all the test images.

detectionResults = detect(detector,testData,'MinibatchSize',4);

% Evaluate the object detector using the average precision metric.

[ap, recall, precision] = evaluateDetectionPrecision(detectionResults,testData);
% The precision/recall (PR) curve highlights how precise a detector is at varying levels of recall. The i
deal precision is 1 at all recall levels. The use of more data can help improve the average precision but
might require more training time. Plot the PR curve.

figure
plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f', ap))
```

Die Dabei erzielten Ergebnisse sehen wie folgend aus: Die Erzielte Genauigkeit ist eher mäßig erfolgreich aber das bessert sich dann im folgenden Script



Umsetzung des zweiten Netzwerks

Zur Umsetzung des zweiten Netztes, welches die vorher erkannten und zugeschnittenen Bilder verarbeitet und das dementsprechende Geschwindigkeitslimit 144741

Das Alexnet

Das AlexNet ist ein neuronales Netzwerk, welches 8 Schichten tief ist. Es ist möglich eine vortrainierte Version des Netzes zu laden, die mit mehr als einer Million Bildern aus der „ImageNet-Datenbank“ trainiert wurde. Das Netzwerk hat eine feste Bildeingabe-Auflösung von 227x227 Pixel.

Umsetzung im Programmcode

Zuerst setzen wir eine Variable an welche zum einen den Speicherort der zugeschnittenen Bilder wiedergibt, folgend wird festgelegt das alle Unterordner mit einbezogen werden sollen.

```
imds=imageDatastore( ...
    'SignsCutted', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
```

Folgend geben wir die Anzahl der gefundenen Bilder an und teilen diese zufällig 70/30 in „Lernbilder“ und Bilder zur Validierung auf.

```
fprintf("Anzahl Bilder: %d\n", length(imds.Labels));
rng(7);
[imdsTrain, idmsValidation] = splitEachLabel(imds,0.7,'randomized');
```

Da die letzten drei Schichten des AlexNet auf 1000 verschiedene Inhaltstypen voreingestellt sind, müssen wir diese aus dem Netz entfernen. Dazu Extrahieren wir alle Schichten, bis auf die letzten drei aus dem Netz.

```
layersTransfer = net.Layers(1:end-3);
```

Daraufhin werden die extrahierten Schichten einer neuen Aufgabe übergeben und es werden die folgenden drei Layer neu hinzugefügt.

```
numClasses = numel(categories(imdsTrain.Labels));

layers = [
    layersTransfer
    fullyConnectedLayer(numClasses, ...
        'WeightLearnRateFactor',20, ...
        'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer
];
```

Nun setzten wir die zum Trainieren des Netzes verschiedene Einstellen die unter anderem aus Epochenanzahl, Lernrate, und Validation der Daten.

```
options = trainingOptions('sgdm', ...
    MiniBatchSize=10, ...                       % 10
    MaxEpochs=6, ...                            % 6
    InitialLearnRate=1e-4, ...                  % 1e-4
    ValidationData=idmsValidation, ...
    ValidationFrequency=3, ...                  % 3
    ValidationPatience=10, ...                  % 5
    Verbose=false, ...                          % false
    Plots='training-progress');
```

Folgend beginnen wir mit dem Trainieren des Netzes, es wird in der Standarteinstellung die GPU 1 verwendet.

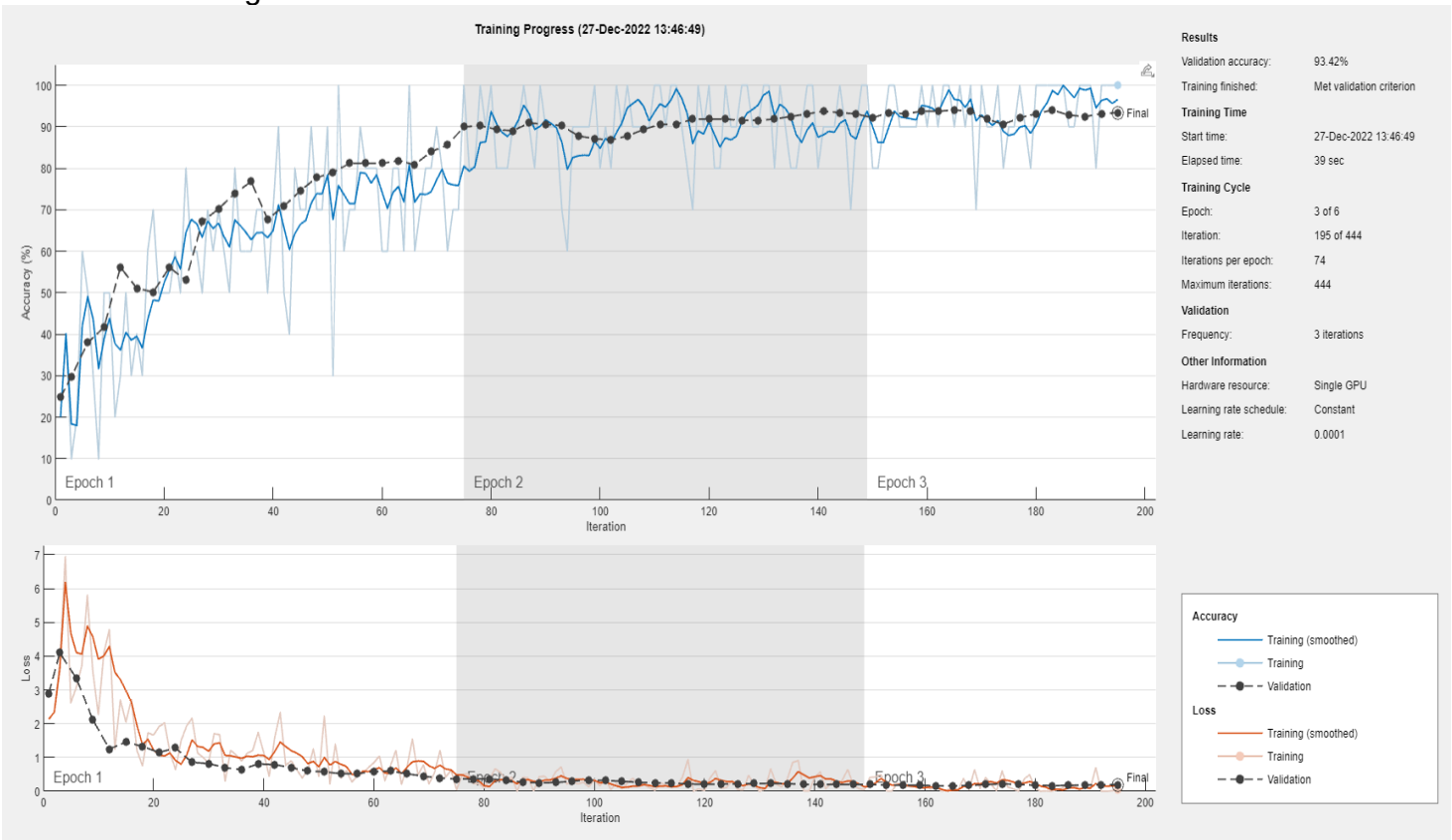
```
netTransfer = trainNetwork(imdsTrain, layers, options);
```

Final geben wir wie folgend sowohl den Trainingsverlauf als auch 64 der genutzten Bilder mit eindeutiger Markierung der nicht erkannten Bilder.

```
[yPred, scores] = classify(netTransfer, idmsValidation);

idx=randperm(numel(idmsValidation.Files), 64);

figure
for i=1 : 64
    subplot(8,8,i)
    I = readimage(idmsValidation, idx(i));
    imshow(I)
    label = yPred(idx(i));
    title(string(label));
end
```

Das nun erstellte AlexNet hat in diesem Beispiel eine Erkennungsrate von ~95% (61 von 64 erkannt).

Final haben sind mit allen genutzten Bildern auf eine validierte Genauigkeit von 93.42% gekommen.



Das dritte Netz (Faster RCNN)

Die Aufgabe für das dritte Netz bestand darin, mit mehreren Bild-Kategorien zu trainieren und auch die Klassifikation vorzunehmen. In Unserem Programmcode beginnen wir damit, die Bilddateien zu Laden bzw. die Zip Datei zu entpacken. Zudem geben wir unsere Groundtruth Datei an.

```
dataDir = 'Picturedata'; % Destination-Folder for provided (img) Data
zippedDataFile = 'PicturesResizedLabelsResizedSignsCutted.zip'; %Data provided by
TA
grDataFile = 'signDatasetGroundTruth_stepthree.mat';
func_setupData_stepthree(dataDir, zippedDataFile, grDataFile);
```

Folgend laden wir diese Datei und teilen die Bilder 60/10/30 in Training, Validation und Testbilder auf.

```
data = load(grDataFile);
traficSignDataset = data.DataSet;

rng(0)
shuffledIndices = randperm(height(traficSignDataset));
idx = floor(0.6 * height(traficSignDataset));

trainingIdx = 1:idx;
trainingDataTbl = traficSignDataset(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = traficSignDataset(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = traficSignDataset(shuffledIndices(testIdx),:)
```

Daraufhin erstellen wir Storages für die Bilder während des Trainingsprozesses

```
imdsTrain = imageDatastore(trainingDataTbl{:,'imageFilename'});
bldsTrain = boxLabelDatastore(trainingDataTbl(:,2:end))

imdsValidation = imageDatastore(validationDataTbl{:,'imageFilename'});
bldsValidation = boxLabelDatastore(trainingDataTbl(:,2:end))

imdsTest = imageDatastore(testDataTbl{:,'imageFilename'});
bldsTest = boxLabelDatastore(trainingDataTbl(:,2:end))
```

Nun erstellen wir das Faster-RCNN und geben an, dass wir nur Verkehrsschilder erkennen wollen.

```
% ----- Create Faster R-CNN Detection Network

inputSize = [224 224 3];

preprocessedTrainingData = transform(trainingData,
@(data)preprocessData(data,inputSize));
% Achtung: dieser DS wird nur zur Ermittlung der BoundingBoxes verwendet


% Auswahl der anchor boxes
%   Infos dazu: https://de.mathworks.com/help/vision/ug/estimate-anchor-boxes-
from-training-data.html
numAnchors = 3;
anchorBoxes = estimateAnchorBoxes(preprocessedTrainingData,numAnchors);

% und das feature CNN
featureExtractionNetwork = resnet50;
featureLayer = 'activation_40_relu';
numClasses = width(traficSignDataset)-1;    % also hier: 1, es sollen nur
Verkehrsschilder erkannt werden
```

Zudem müssen, falls das Netz trainiert wird, verschiedene Einstellungen getätigt werden. Diese bestehen aus zwei Overlap-Ranges.

```
if doTraining
    % Train the Faster R-CNN detector.
    % * Adjust NegativeOverlapRange and PositiveOverlapRange to ensure
    %   that training samples tightly overlap with ground truth.
    [detector, info] = trainFasterRCNNObjectDetector(trainingData,lgraph,options,
...
        'NegativeOverlapRange',[0 0.3], ...
        'PositiveOverlapRange',[0.6 1]);
    save netname detector;
else
    % Load pretrained detector for the example.
    load (netname, 'detector');
end
```

Final ist im Programmcode ein Test hinterlegt, welcher Testdaten verwendet, das Ergebnis evaluiert und einen Plot ausgibt.

```
% ----- Testing

testData = transform(testData,@(data)preprocessData(data,inputSize));

% Run the detector on all the test images.

detectionResults = detect(detector,testData,'MinibatchSize',4);

% Evaluate the object detector using the average precision metric.

[ap, recall, precision] = evaluateDetectionPrecision(detectionResults,testData);
% The precision/recall (PR) curve highlights how precise a detector is at varying
levels of recall.
% The ideal precision is 1 at all recall levels. The use of more data can help
improve the average precision but might require more training time.
% Plot the PR curve.

figure
plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f', ap))
```

Leider sind unsere Ergebnisse mit diesem Netz nicht zufriedenstellend.
Wie in dem folgenden Beispielbildern zu sehen, ist es dem Netz nicht möglich ein Geschwindigkeitsschild richtig zu erkennen.