02456 DEEP LEARNING, DTU COMPUTE, FALL 2020

HEDIA FOOD RECOGNITION WITH FASTER RCNN

Authors: Jia Yi Johnny Ye, Kia Kafaei, Aslak Rønnow Franzen, Jiayi Han & Hjalte Reuss Schmidt

Technical University of Denmark DTU Compute Lyngby

1. ABSTRACT

This report provides an overview on how a faster RCNN is constructed to identify food and its location in images. The neural network can be split into 3 networks, which is a VGG16 pre-trained convolutional neural network, RPN and a detection network. An explanation of how the networks are constructed and the processing between the networks will be explained. A walk through of the code will be shown in the form of a flowchart. The data set given has class imbalance problems and not enough images to train on for each food class categories. This leads to a classification problems, since the algorithm will tend to predict the most presented classes.

***Index Terms*—** Object Detection, Faster R-CNN, Food Detection

2. INTRODUCTION

This project is done in collaboration with Hedia, which is a company that develops innovative digital solutions for diabetic patients. In relation to their work on the APP Hedia Diabetes Assistant (HDA 1.0) where patient's glucose intake can be monitored, we have decided to work with image recognition of food, which could be interesting to incorporate into further improvement of the app functionalities. The object detection algorithm Faster RCNN (Regional Convolutional Neural Network) is chosen for this task.

3. RELATED WORK

A series of object detection models were considered in the beginning of the project. These models are able locate and classify the objects in the images. All were based on the convolution neural network (CNN) which is standard for image detection. In all cases Faster-RCNN were found to have the highest mean accuracy precision which for this project was more important than high analyzing speeds of the real-time systems [1]. Faster-RCNN is explained in depth in section 4.

3.1. RCNN

RCNN, the first model of the R-CNN family, was first considered. In object detection algorithm the region proposals are extracted using a selective search algorithm, a fixed model that cannot be trained, which propose around 2000 regions each of which are fed to a CNN model for feature extraction. The feature mapping is then fed to a SVM (Support Vector Machine) to classify objects within each region proposal as well as refinement of bounding boxes in order to avoid cut off.[2]

3.2. YOLO

The YOLO (You Only Look Once), a regression based network, the image is divided into grid cells and then only once run through a CNN. Hereafter there is simultaneously calculated a conditional class probability for the different classes for each grid cell as well as object confidence scores for a number of bounding boxes set up around each grid cell. These probabilities are multiplied to give a class-specific confidence score which both reflects the probability of the class of the object in the individual boxes as well as how good the box fits the object. This way of only running through a CNN once give YOLO high run-through speeds and by finding class probabilities throughout the entire image relational information are found between objects.[1]

3.3. Mask RCNN

Mask RCNN is an extension of Faster-RCNN with inclusion of a final branch (the other being the class predictor and the BBox regressor) that determines whether individual pixels belong to a given object and thereby giving information about the shape of the object. However, as this feature is more complex, Mask RCNN was dismissed in favor of Faster-RCNN.[3]

4. FASTER RCNN

Faster RCNN is a regional based CNN algorithm suited for object detection with the output being both the location and the classification of object. It is composed of different components that will be explained in the following sections.

4.1. Feature Map

Feature maps is generated using pretrained VGG–16, the reason for using a pretrained network is that, it, from a computational perspective, saves time as VGG-16 is trained on ImageNet which is a database containing 14 million images of 1000 classes. Feature extraction is the first step of faster RCNN. VGG-16 is a large network with 16 layers with weights. The network has around 15 million parameters in total. The VGG-16 has an output of 512 feature maps[4].

4.2. Anchor

The purpose for anchor boxes is to lay out ROI (region of interests) in order to make predictions upon. Special about the Faster RCNN algorithm and many other object detection algorithms is that they use anchor boxes for object detection. For each feature map pixel, there will be generated k number of anchor boxes of different sizes. The number of of anchor boxes generated for each pixel is 9 in this project. The pixel that the boxes are centering are called anchor points.

At the end of an object detection algorithm, only the boxes with actual objects will remain. Therefore the selection of the correct amount and sizes of boxes is crucial for performance of object detection [5, 6].

4.3. RPN

Region proposal network (RPN) enables detection and classification of objects within an image. RPN is a convolutional neural network. It consists of an intermediate layer then one classifier and one regressor that both are fully connected layers. The classifier gives the probability of proposal having an object or background and regressor gives the coordinates of that object. The algorithm distinguishes whether it is an object or background. The proposals are generated upon anchors. The number of proposals per anchor is a free parameter. The RPN gives locations of each anchor box together with a score for the probability of an object is within the box. [5, 7, 8]

4.4. NMS

Non maximum suppression (NMS) is a process that reduces the number of anchor boxes that overlap with each other. This procedure comes after RPN where there has been generated a large number of region proposals. NMS uses two parameters; threshold and confidence score to filter the region proposals. Confidence score is a measure of the possibility of an object being within the region proposal. The algorithm takes each proposal, starts with the bounding box with highest confidence score, then compares each of the proposals with the rest. If the IOU (intersection over union) score is higher than the threshold in the remaining batch of region proposals, those bounding boxes will be removed. This process is done iteratively. IOU between two bounding boxes is a measure of overlap, therefore high IOU indicates that two boxes are representing the same object proposal. [9]

4.5. Sample ROI

The ROI's are now batched in sets of 128 samples. The ROIs that are sampled are the ROIs that are identified as objects or the ROIs that are identified as background. The ROIs that have IOU values in between these two thresholds are ignored. Since it is unclear to the network in these cases whether the ROI is identifies as an object or background. These 128 ROI samples is then send further through the network to the ROI pooling layer.[10]

4.6. ROI Pooling

ROI pooling is a neural network layer used to standardize the shape of the ROI's such that the input dimensions for the detection network is fixed. The ROI pooling layer takes feature maps and the ROI coordinates as input. It works by splitting the region of interest into an AxB area, which in this case is a 7x7 area. Just like Max pooling it then takes the maximum value from each area and outputs it to a 7x7 matrix, which is then used as input to the detection network.[11]

4.7. Detection Network

Just like the RPN, the detection network consists of a regressor and a classifier. The classifier is used to classify what the object in the ROI is. The regressor is used to determine where on the image the object is. The network consists of a max pooling layer and two fully connected layers.

4.8. Loss

The total loss of the entire network is the sum of the losses from the RPN network and the losses from the detection network. Each of the networks consists of a classification error and a regression error, shown in the general formula below.

$$L\left(p,u,t,v\right) = L_{cls}(p,u) + \lambda * L_{loc}(v,t) \tag{1}$$

p is in this case the predicted class and u is the actual class in the classification loss. v is the location of the predicted class and t is the location of the actual class in the regression loss.

λ is a hyper parameter indicating how significant the location error should be compared to the classification error. Bigger λ means that the location loss will have a larger significance in the total loss.

The classification loss of the RPN network indicates how wrong the model is in detecting whether a specific region is an object or background. The classification loss of the detection network tells us how wrong the network is in determining what the object is.

The location loss for both networks describe the error associated with the placement of the ROI compared to the ground truth box.

For each network we use the cross entropy loss function to determine the classification losses and the smooth L1 loss function to determine the location losses.[10] The smooth L1 loss function is a combination of the L1 and L2 loss functions. The loss function acts as a L1 function for arguments with high values and behaves as the L2 loss function when the value of the argument is low. This threshold is determined using a hyper-parameter. The equation for the smooth L1 loss function is shown below.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta \\ \delta\left(|a| - \frac{1}{2}\delta\right), & \text{otherwise} \end{cases} \tag{2}$$

Here a is the argument and δ is the hyper-parameter. [12]

5. IMPLEMENTATION

As described in section 4 a large number of different algorithms are used in faster RCNN. A flowchart over the code architecture is shown in figure 1. The detailed code is in https://github.com/KiaKafaei1/Hedia_project

From figure 1 we start with feeding the input image into the VGG-16 network that is a 16 layered convolution network, which outputs a feature map. The feature map is used in the RPN in order to find the regions of interest(ROI) and anchor boxes. The ROI and anchor boxes are used to calculate the loss of the RPN. The non maximum suppression(NMS) algorithm then looks at the ROI and reduces the number of ROI on how much overlap there is between ROIs. Sample ROI then takes a sample of 128 of the reduced ROI and gives them to the ROI pooling. The ROI pooling also takes the feauture map and uses an adaptive max pooling on the feature map and the sample ROIs. The output of the ROI pooling is used in the detection network in order to classify the objects in the image and location. From the detection network and the loss of RPN the total loss can be found.

The full architecture of our faster RCNN includes 136.873.389 weights which were trained with back-propagation using a steepest gradient descent method for optimization ($\mu = 0.9$). This is distributed over the three networks as seen in table 1.

Network	**# Weights**
VGG16	14.714.688
RPN	2.387.510
Detection	119.771.191

Table 1

VGG16 is as already mentioned an pre-trained model, but it was still updated during training. With that many weights a larger dataset might be better in order to train the algorithm more effectively.

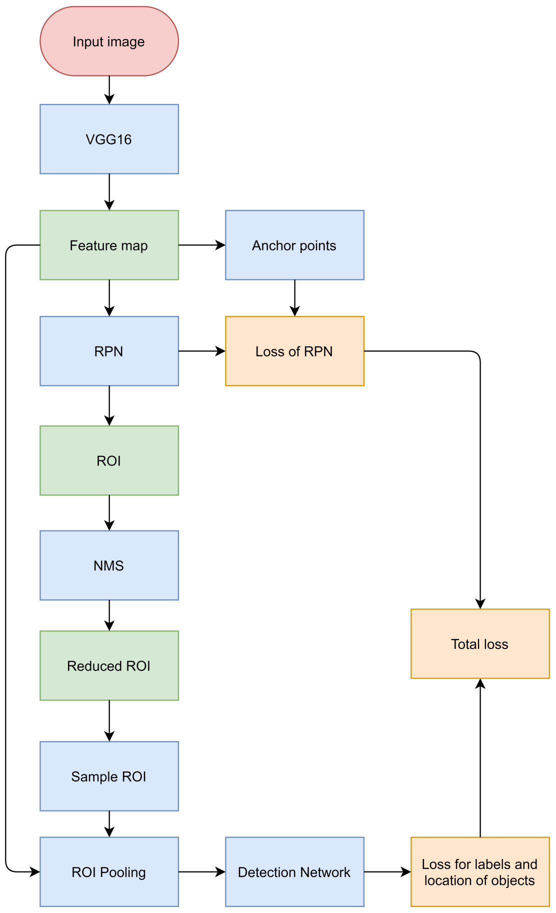


Fig. 1: Flowchart of faster RCNN. Here we have color coded the red boxes as input, blue boxes as algorithms part, orange boxes for loss and green boxes indicates important outputs

6. DATA

The data set used in the project is food images from Hedia with pre-assigned regions of food and labels in the VASCO data format. The data set is split up into a training, test and validation set. We have a total of 10 different food categories over 5124 images. The distribution of food categories can be seen in figure 2 and the number representing the categories can be seen on table 2 In general we see that there is an imbalance of classes in the given data set. As shown in figure 2 the categories are not equally distributed. From figure 2 it is clear that the data set is heavily tilted towards meatballs and therefore it might learn to predict meatballs more than other classes. This is called the accuracy paradox[13].

We cannot perform have sufficient training due to the limited amount of data. As a rule of thumb, there should at least be 1000 image examples represented for each class[14]. This is not the case here since plain rice, plain spaghetti etc doesn't have 1000 images in the data. Another issue in our data set is the food category glass of water which might be hard for the

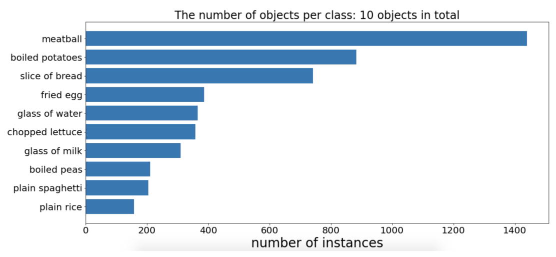


Fig. 2: Distribution of images over 10 different food categories for all images for the training data

Meatballs	1
Slice of bread	2
boiled potatoes	3
Fried egg	4
Glass of water	5
Chopped lettuce	6
Glass of milk	7
Boiled peas	8
Plain spaghetti	9
Plain rice	10

Table 2: Here can be seen the 10 different food categories and their corresponding number given by the network when predicting the label of the object.

neural network to detect, since water is transparent.

7. RESULTS

The network was trained on 2023 images consisting of 10 different categories of food. The final hyper-parameters used for training the network can be seen in table 3.

Learning rate	5E-5
Weight decay	5E-4
Momentum	0.9
Weight of location loss in RPN, λ_{RPN}	1
Weight of location loss in ROI, λ_{ROI}	1
Samples from anchor generation	256
Ratio of foreground sampled in anchor generation	0.5
Threshold for foreground IoUs in anchor generation	0.9
Threshold for background IoUs in anchor generation	0.3
Samples from Sample ROI	128
Ratio of foreground sampled in Sample ROI	0.25
Threshold for foreground IoUs in Sample ROI	0.9
High threshold for background IoUs in Sample ROI	0.3
Low threshold for background IoUs in Sample ROI	0.0
Allowed overlapping area in NMS	0.7
Samples pre NMS	12000
Samples post NMS	2000
Minimum size of bounding box in nms	16x16

Table 3: Hyper-parameters used in training the network

Using the hyper-parameters seen in table 3 the network is trained and the corresponding loss graphs can be seen in figures 3, 4 and 5.

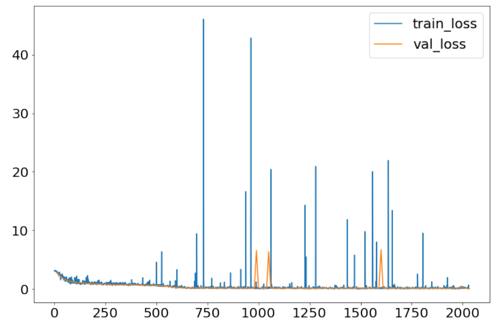


Fig. 3: The training and validation loss for the network when training. At earlier iterations learning can be observed while minor learning can also be seen at iteration 500. The learning is hard to observe due to the spikes of higher loss values assumed due to images deviating from the norm.

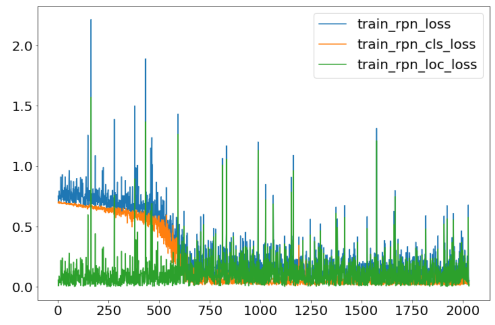


Fig. 4: RPN training loss given by location and classification loss. Here it can be seen that the learning happening at iteration 500 is due to learning in RPN classification.

In figure 3 the training loss and validation loss can be seen. From the figure it can be observed that in the early training iterations there is some learning happening. But at later training iterations some spikes occur which is probably due to outliers in the training and validation data that does not conform to the underlying principles learned from the rest of the training data-set. These spikes have a much higher loss obscuring some of the learning achieved by the network. When looking at the loss for the RPN part of the network, figure 4, it can be seen that learning also happens for the classification part of the RPN at training iteration 500. Comparing figure 3 and figure 4 with figure 5 it can be observed that it is the ROI loss that dominates the total loss for the network.

Some selected prediction for the fully trained network can be seen in figure 6.

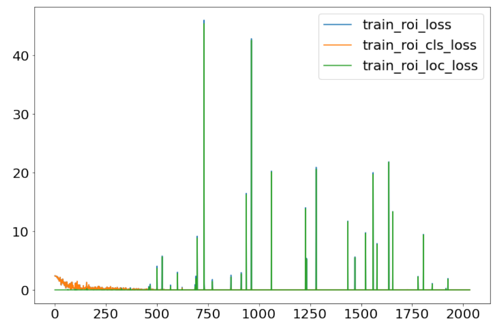
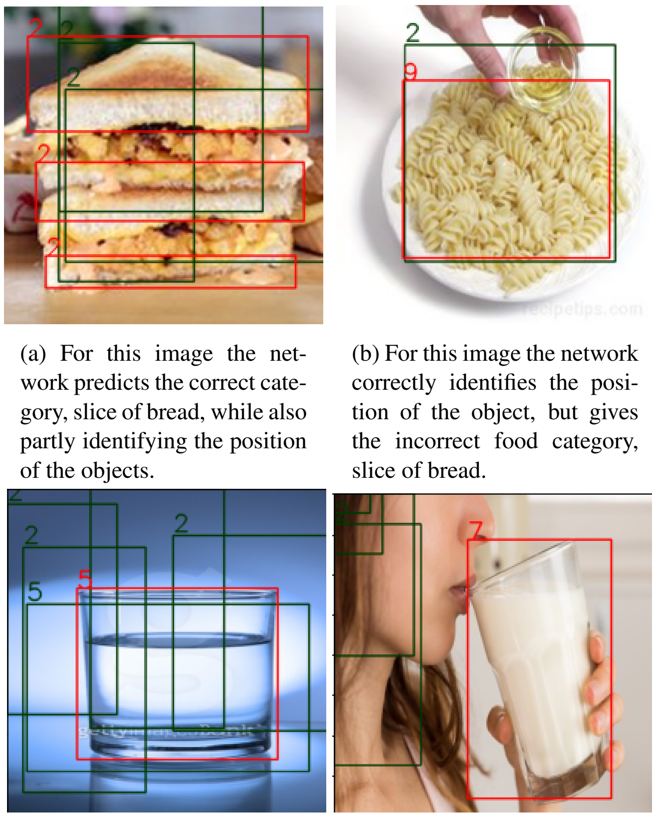


Fig. 5: ROI training loss given by location and classification loss. It can here be seen that the early learning is due to learning in ROI classification. It can also be seen that the ROI loss is the dominate loss for the total training loss.



(a) For this image the network predicts the correct category, slice of bread, while also partly identifying the position of the objects.

(b) For this image the network correctly identifies the position of the object, but gives the incorrect food category, slice of bread.

(c) For this image the networks top 5 predictions are shown. It can be seen that for one of them the network have both the correct position and correct category for the object, water, which are given by the number 5 in the image.

(d) For this image it can be seen that the network's top 5 predictions fail to identify the position of the object. The category of the object is also wrong as the network in this image predicts slice of bread for all top 5 predictions.

Fig. 6: Predictions of the trained network. Red boxes are the ground-truth of the object while the red number it is category. The green boxes and numbers are the prediction of the network.

The fully trained network has a hard time correctly classifying the different food categories. When going through its prediction it is found that the food categories that have an overabundance in the data-set such as meatballs, boiled potatoes and slice of bread are over-represented in the networks prediction. An example of this can be seen in figure 6d, 6c and 6b. Here examples of the network making the category predictions of slice of bread, one of the three most abundant categories in the data-set. This outcome is in accordance with expectations for a data-set with an imbalance in the classification categories [14]. Something the network does achieve with a high rate of success is to identify where the objects of interest are located. In figure 6b a clear example of the network correctly predicting the location of the food is seen. Furthermore in figure 6a the network has partly predicted the correct location and in figure 6c it can be seen that one of the top predictions of the network also includes the correct location of the object. From figure 4 it can be seen that learning happens for the classification of background and object for the RPN part of the network. It can therefore be concluded that the RPN network learns which features given by the VGG16 network that are of interest when identifying the food objects. It can also be observed that almost no learning happens for the location loss in either ROI or RPN. This could be due to the VGG16 network used in the over-all network being pre-trained. It is thus already primed to identify features in the images, and the rest of the network therefore only has to learn which features is of interest when working with the 10 different food categories.

8. FURTHER WORK

There is still a lot of room for improvement for the network. An obvious improvement to the network will be further fine-tuning of the network's hyper-parameters. The network would undoubtedly benefit from a more rigorous testing of the different hyper-parameters than allowed for in this project due to time constraints. Such a fine-tuning of the hyper-parameters could result in improved performance of the network and could also result in reduced training time.

Another promising area of improvement would be an expansion of the data-set. With only 2032 training images there is a size deficiency for all 10 categories of food images in the project when counting individual images and a size deficiency for most of the categories when counting instances of the food [14]. With a larger sample size to train the network on there will be a large chance that the network can find stronger underlying factors which will help the network make stronger and more accurate predictions. Even though an expansion of the data-set can be a time consuming endeavor, it is a fairly simple and straight-forward task making it an easy improvement to implement. There is also a category imbalance in the data-set which will have to be taken into account when collecting further data. As a patch to the data already available, one could sort the images so the instances of the different categories are roughly equal. The instances of the categories varies from image to image so this can be slightly difficult to achieve in a time efficient manner.

9. REFERENCES

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[4] "Step by step vgg16 implementation in keras for beginners," https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c, Accessed: 2020-12-28.

[5] R. Girshick S. Ren, K. He and J. Sun, "Faster r-cnn: Towards real- time object detection with region proposal networks," 6 Jan 2016.

[6] "Anchor boxes — the key to quality object detection," https://medium.com/@andersasac/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9, Accessed: 2020-12-28.

[7] "Faster r-cnn explained for object detection tasks," https://blog.paperspace.com/faster-r-cnn-explained-object-detection/, Accessed: 2020-12-28.

[8] "Region proposal network (rpn) — backbone of faster r-cnn," https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9, Accessed: 2020-12-28.

[9] "Non maximum supression," https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c, Accessed: 2020-12-28.

[10] "Guide to build faster rcnn in pytorch," https://medium.com/@fractaldle/guide-to-build-faster-rcnn-in-pytorch-95b10c273439, Accessed: 2021-01-01.

[11] "Region of interest pooling explained," https://deepsense.ai/region-of-interest-pooling-explained/, Accessed: 2021-01-01.

[12] "Huber loss," https://en.wikipedia.org/wiki/Huber_loss, Accessed: 2021-01-03.

[13] "8 tactics to combat imbalanced classes in your machine learning dataset," https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/, Accessed: 2020-12-30.

[14] "37 reasons why your neural network is not working," https://blog.slavv.com/37-reasons-why-your-neural-network-is-not-working-4020854bd607, Accessed: 2020-12-30.