Master's Thesis

Authors:
Christoffer Bøgelund Rasmussen

Supervisors:
Kamal Nasrollahi

Aalborg University
VGIS
10th semester
Group 17gr1041

TBA

AALBORG UNIVERSITY

Title:
Master's Thesis
Subject:
Vision, Graphics and Interactive Systems
Project period:
1/2-2017 to TBA
Project group:
17gr1041
Participants:
Christoffer Bøgelund Rasmussen

TBA

Supervisor:
Kamal Nasrollahi
Printed copies:
TBA
Number of pages:
TBA
Appendix media:
AAU digital exam zip file
Finished:
TBA

Preface

Reading Guide

Tables, code listings and figures are numbered sequentially within each chapter. Citations are written as [x] where x denotes the reference number used in the bibliography. Code classes and functions are written as `class` and `function()`, respectively. Additional files have been uploaded to the AAU Digital Exam.

Christoffer Bøgelund Rasmussen

Contents

1 Introduction

Object detection is a fundamental area of computer vision that has had a great amount of research over the past decades. The general goal of object detection is to find a specific object in an image. The specific object is typically from within a pre-defined list of categories that are of interest for a given use case. Object detection generally consists of two larger tasks; localisation and classification. It is assumed that the objects of interest are not already located in the image and as objects can vary in number of pixels depending on factors such as distance and scale, objects must be both localised in an image and classified accurately. Localisation is typically done by with a bounding-box indicating where a given object is in the image. However, other methods such as objects' centres and closed boundaries can also be used [4]. Not only is object detection an important task in localising and classifying, it is also a necessary earlier step in larger computer vision pipelines. For example, object detection is needed within the tasks such as activity and event recognition, scene understanding, and robotic picking.

Object detection is a challenging problem due to both some large scale issues and minute differences. Firstly, there is the challenge of differentiating objects between classes. Depending on the problem at hand the sheer number of potential categories present can be into the thousands or tens of thousand. On top of this separate object categories can be both very different in appearance, for example an apple and an aeroplane, but separate categories can also be similar in appearance, such as dogs and wolves.

Current state-of-the-art within object detection is also within the realm of deep learning with Convolutional Neural Networks (CNNs). This is exemplified with almost all leading entries in benchmark challenges such as Pattern Analysis, Statistical Modelling and Computational Learning Visual Object Classes (PASCAL VOC) [1], ImageNet [2], and Microsoft Common Objects in Context (MS COCO) [3] consisting of CNN-based approaches. However, improvements are still needed before object detection can be used in real-world scenarios that require a high level of precision, accuracy, and performance.

1.1 Initial Problem Statement

An initial problem statement can be formed as follows:

How is object detection performed with CNNs?

Based upon this, the following chapter will cover these challenges. On top of this, related work into current state-of-the-art object detection will be researched.

2 Problem Analysis

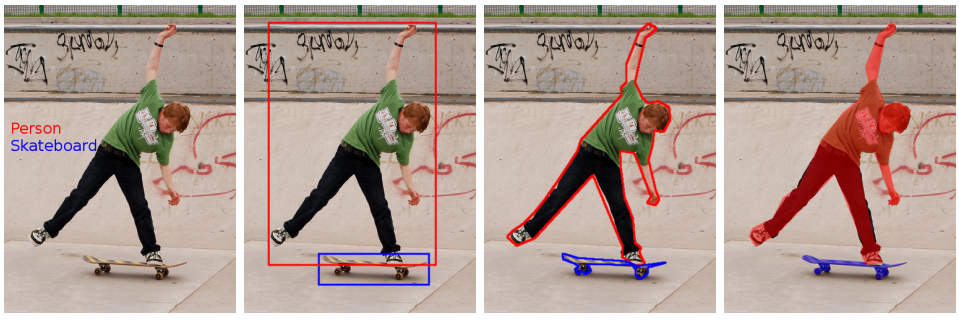
This chapter will outline object detection and it's key challenges. This includes aspects within robustness, computational-complexity and scalability. Once completed the current key works within object detection will be analysed, both current state-of-the-art.

what else is covered in this chapter

2.1 Object Detection

As mentioned in Section *1.1 Initial Problem Statement*, object detection consists of two larger tasks; classification and localisation. Depending on the problem at hand, object detection can be split into two categories. If only a single class is of interest, such as detecting a specific traffic sign, the object detection task is denoted as class-specific detection. Whereas, in the more general case when multiple classes are of interest, it is denoted as multi-class detection [4]. Key challenges such as PASCAL VOC, ImageNet and MS COCO are of the latter task. This thesis will be within the multi-class detection domain and take these challenges into account when analysing related works and determining the algorithm to be implemented. The goal of an object detector is to output a list of labels from a predefined list of categories indicating which objects are present and where they are located in an image. Object detection has a number of related fields which share the common goal of categorising relevant objects. This can be seen in Figure 2.1. In all four instances the goal is to categorise the two objects person and skateboard, however, the difference lies in the level of localisation precision. In Figure 2.1a, object categorisation aims to only classify the objects in the image without providing any indication as to where the objects are located. Object class detection in Figure 2.1b, localises the classified objects with the use of bounding-boxes, where ideally the bounding-boxes are placed as tightly around the given object as possible. Figure-ground segmentation in Figure 2.1c, indicates localisation with a lasso outline around the objects. Finally, in Figure 2.1d, semantic-segmentation localises objects at a pixel-level classifying each pixel that is related to the given object.

correct section refs to above



(a) **(b)** **(c)** **(d)**

Figure 2.1: Example of vision tasks related to object detection. All tasks have the common goal of categorising predefined objects. Methods are: object categorisation (a), object class detection (b), figure-ground segmentation (c), semantic Segmentation (d). Image and class labels taken from MS COCO [3].

A recent boom in the domain of segmentation has been that of instance segmentation. Instance segmentation differs to semantic segmentation in that individual instances of ob-

jects are classified as such. If multiple instances of the same object is present, such as an image of a crowd with many people, in semantic segmentation all people will be given the same label as one large group. However, in instance segmentation the people are still given the same label but individual instances of a person is also found. This area of research within segmentation is relatively new, however, is beginning to become more popular in comparison to semantic segmentation. For example, the MS COCO segmentation challenge which has been held in 2015 and 2016 only accepts instance segmentation entries.

2.2 Main Challenges

The challenges of object detection can be split into two groups as per [4]:

- Robustness-related.
- Computational-complexity and scalability-related.

The following sections will outline the above.

2.2.1 Robustness-related Challenges

Robustness-related refers to the challenges in appearance within the both of intra-class and inter-class. Intra-class is the differences in appearance of objects which are of the same class. For example, as seen in Figure 2.2, all of the images belong to the superclass chair from the ImageNet training set [2], however, vary greatly in their overall appearance.

(a) **(b)** **(c)** **(d)**

Figure 2.2: Examples of intra-class appearance variation. All images have the label chair in the ImageNet training set [2].

An object detection system must be able to learn the appearance variations that can occur intra-class. These variations can be categorised into two types as per [5]:

- Object variations.
- Image variations.

Object variations consist of appearance differences between instances of colour, texture, shape, and size. Image variations are differences not related to the object instances themselves but rather consist of conditions such as lighting, viewpoint, scale, occlusion,

and clutter. Based upon these conditions the task of both classifying a given object as a given class but also differentiating the potentially largely varying objects into the same class challenging.

Robustness-related challenges can also occur with inter-class appearance differences. This refers to the differences between objects that are regarded as different categories. Challenges arise in scenarios where an object detector must decide if an instance is between classes that are very similar. For example using images and their respective classes from ImageNet [2], in Figure 2.3 and Figure 2.4 the differences between the two examples are very similar, however, their class labels are different. In Figure 2.3a and Figure 2.3b the class labels are mini-bus and delivery truck respectively. In Figure 2.4a and Figure 2.4b the labels are white wolf and German shepherd.

(a) **(b)**

Figure 2.3: Examples of inter-class appearance variation. Both images are from the ImageNet training set [2] and have the labels mini-bus (a) and delivery truck (b).

(a) **(b)**

Figure 2.4: Examples of inter-class appearance variation. Both images are from the ImageNet training set [2] and have the labels White wolf (a) and German shepherd (b).

It should be noted that this is a task-specific if inter-class appearance similarities is a problem or not. It can be argued that both the examples in Figure 2.3 and Figure 2.4 can be

grouped into a larger superclass label if the given task does not require training of a model to such granularity. In both examples the classes stated are of the lowest class available in the overall hierarchy. ImageNet has labels available for each image along a larger array of classes and sub-classes. Figure 2.5 visualises the granularity possible where both images belong to the superclass animal.

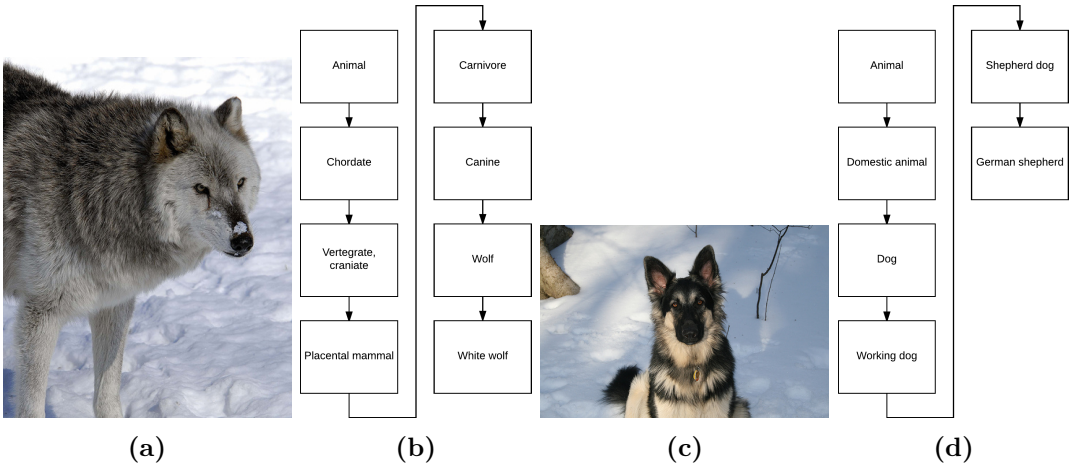


Figure 2.5: Visualisation of the hierarchy of potential classes for two examples in the ImageNet training set [2].

2.2.2 Computational-complexity and Scalability-related Challenges

The second challenge as per [4] is related to the potential scale of object detection. When deciding on which type of model to use it must be complex enough to be able to capture the previously mentioned challenges both in inter- and intra-class. On top of this there is an extremely large number of potential classes in object detection. If the aim is to train a model to classify between an extreme number of classes then naturally a large number of images are also needed for each category. The large number of images need also to be representative enough in training to capture the necessary visual features to generalise on non-training images. In 2016 the ImageNet object detection challenge there is a total of 200 object categories, with 456,567 images comprising the training set. Therefore, a complex enough model is needed in order to learn and generalise on such a dataset but this of course places high requirements on the amount of training needed.

Issues can also arise over time when designing an object detection system. Over time the visual appearance of an object can change, which is very difficult to take into account when training a model. For example, the visual appearance of televisions have changed greatly in the past century. If a system were only to be trained on images from an earlier time period it may not be able to generalise on new instances. Therefore, it is important that a model is able to be updated as the appearance of objects change. On top of this, new categories of objects can come to fruition which may needed to be added to a model.

2.3 Benchmark Datasets

This section will outline some of the commonly used object detection datasets. This will include their purpose and the general setup.

2.3.1 PASCAL Visual Object Classes Challenge

The PASCAL VOC challenge [1] was held yearly between 2005 to 2012 and provided datasets for benchmarking within vision tasks of visual object category recognition and detection. Between that time period is was considered the top benchmark for the respective challenges. While being an annual competition, PASCAL VOC evaluation in state-of-the-art literature is most often performed on data from the years 2007 and 2012. The competition saw a large shift in the former year as the number of classes increased from 10 to 20, in turn also significantly increasing the total amount of data. Additional data was added individually for the various recognition tasks between 2007 and 2012 and the performance metric was altered slightly between this time period, however, the overall ecosystem remained largely the same from 2007 until the competition's end. This section will be largely based upon the two retrospective papers, [6] and [7], published by authors who were involved in the challenge and for the most part will be in respect to the challenge after 2007.

Images were obtained from the website flickr [8] with the aim in mind to collect natural images for the recognition challenges. Ideally the dataset was to contain a significant level of visual variability in regards to object size, orientation, pose, illumination, position, and occlusion. A total of 20 classes were present in the 2007 challenges and these remained the same until 2012. The classes can be considered as a part of a taxonomy with 4 main branches, where each has finer-grain objects in sub-classes. The 20 classes and the branching taxonomy can be seen in Table 2.1.

Table 2.1: Taxonomy of the 20 classes introduced in VOC2007.

Vehicles	Household	Animals	Other
Aeroplane	Bottle	Bird	Person
Bicycle	Chair	Cat	
Boat	Dining table	Cow	
Bus	Potted plant	Dog	
Car	Sofa	Horse	
Motorbike	TV/Monitor	Sheep	
Train			

A total of 500,000 potential images were collected randomly based upon different combinations of queries for a given class. For example of class bird, potential queries were bird, birdie, birdwatching, nest, sea, aviary, birdcage, bird feeder, and bird table. Of these potential images the majority were discarded for potential annotation due to not meeting the considerations of visual variability mentioned earlier. The annotation process was completed by a team from the University of Leeds based upon strict guidelines. The aim was to ensure that the annotations resulted in a consistent, accurate, and exhaustive dataset. The annotations are stored in XML format which contains the following information:

- Class: one of the 20 shown in Table 2.1.
- Bounding box: axis-aligned bounding-box around the visual extent of the object.
- Viewpoint: viewpoint to the object.
- Truncation: whether or not object is truncated. An object is truncated when the bounding-box does not cover the full extent of the object. Truncation can occur if the object extends outside the image or is partially occluded.
- Difficult: A subjective evaluation on if the object is difficult to detect. This is determined based on object size, illumination, or image quality.

An example of the XML format for the object chair can be seen in Code 2.1 and its corresponding image in Figure 2.6.

Code 2.1 Example of XML annotation for the object chair.

```
<object>
    <name>chair</name>
    <pose>Rear</pose>
    <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>263</xmin>
                <ymin>211</ymin>
                <xmax>324</xmax>
                <ymax>339</ymax>
        </bndbox>
</object>
```

Figure 2.6: Image from the PASCAL VOC 2007 dataset. The bounding box represents the annotated XML data shown in Code 2.1.

Of the 500,000 potential images, 9,963 were annotated for the VOC2007 challenges based upon the annotation guidelines. PASCAL VOC datasets is split into two subsets; `trainval`, consisting of training and validation data and `test`, consisting of the testing data.

A histogram showing the frequency of an object class in an image and the total number of objects for the VOC2007 dataset can be seen in Figure 2.7.

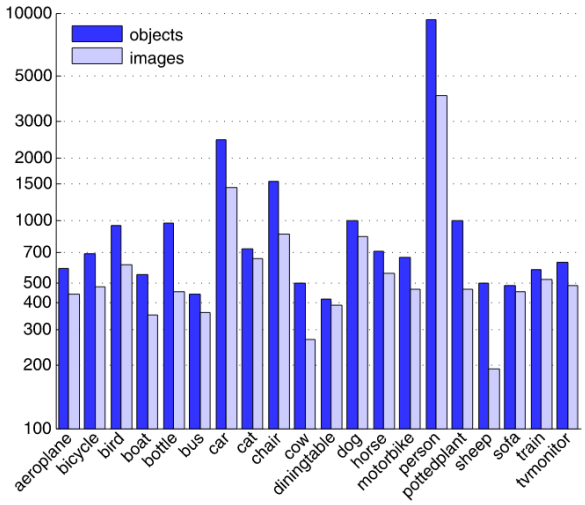


Figure 2.7: Image from the PASCAL VOC 2007 dataset. The bounding box represents the annotated XML data shown in Code **??**.

Evaluation of a object detector on PASCAL VOC is based upon Average Precision (AP) which summarises the precision and recall of detections. The metric requires that for each detection both a bounding-box and an associated confidence. With these a precision/curve can be calculated and the overall AP for the curve represents the performance of the detector. By altering the threshold corresponding to the confidence of each detection a precision and recall curve can be calculated and the AP summarises the shape of the this. Precision is the number of true positives in relation to both true positives and false positives. Whereas, recall is the number of true positives in relation to all detections. However firstly, a detection must be classified as either a true or false positive. This is determined by measuring the bounding-box overlap between the detection and ground truth annotation. In PASCAL VOC a bounding box is a true positive if the overlap is above 50% according to the following calculation:

$$a_o = \frac{area(B_p \bigcap B_{gt})}{area(B_p \bigcup B_{gt})} \tag{2.1}$$

where $B_p \bigcap B_{gt}$ is the intersection of the predicted bounding box and the ground truth and $B_p \bigcup G_{gt}$ is the union between the two. It should also be noted that for a given ground truth object it is only possible to have one true positive detection. If multiple detections satisfy Equation 2.1 the remaining will be denoted as a false positive. Now that detections can be as classified as either a true or false positive AP can be calculated. Before 2007, in PASCAL VOC this was calculated as the mean precision at eleven equally space recalls [0, 0.1, ..., 1] by:

$$AP = \frac{1}{11} \sum_{r \epsilon [0, 0.1, ..., 1]} p_{interp}(r) \tag{2.2}$$

where r is the recall and p_{interp} is the interpolated precision.

explanation of interpolated precision

explanation of new metric

2.3.2 ImageNet Large Scale Visual Recognition Challenge

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an extremely large benchmark in object recognition. The challenge has been held since 2010 with multiple challenge tasks such as image classification, scene recognition and object detection. Currently, in 2017 the only challenges being held are object detection in both images and video. The aim of ILSVRC was to create an image recognition challenge on such a scale that had not been previously seen. Before this, the main challenge was PASCAL VOC with 20 categories in the object detection challenge. ILSVRC has 200 classes, roughly 500,000 annotated positive and negative images and just under 500,000 annotated objects in the positive examples. Once again, the goal of an algorithm is to learn object detection towards to 200 classes and return a bounding-box with an associated confidence for each object detected in an image.

Creating such a large dataset is a difficult and cumbersome task. The majority of images come from another ILSVRC challenge of single-object localisation, where the aim was to only return a single object detection. Additional images were supplemented from Flickr queries. Examples of images from the dataset can be seen in Figure 2.8.

(a) **(b)**

Figure 2.8: Example of images in the ILSVRC object detection challenges.

Evaluation of detections was inspired by the AP metric used in PASCAL VOC. Again detections are determine to be correct or incorrect based on if the Intersection-Over-Union (IoU) is above 0.5. While being a challenging dataset in terms of the number of classes present, the object sizes and number of objects is similar to that of PASCAL VOC. Next will be an explanation of a new benchmark that addresses such challenges.

2.3.3 Microsoft Common Objects in Context

MS COCO [3] is a relatively new dataset within the realm of object recognition appearing in 2015. MS COCO holds competitions in object detection and segmentation. The object

detection challenge is similar to PASCAL VOC, in that detections must be shown using bounding-boxes. However, for the segmentation challenge MS COCO requires the results to be of more challenging instance form rather than semantic. The creators of the dataset had three core research problems they wanted to be present, these include:

1. Detecting non-iconic views of objects.
2. Contextual reasoning between objects.
3. Precise 2-dimensional localisation of objects.

The first problem is present by having object instances in images that are closer to everyday scenarios. Iconic views of objects are when the instance is near the centre of the image, is unobstructed and taken in a controlled scenario. Objects taken in such conditions are much easier to detect but practical applications are limited. Therefore, non-iconic views of objects are collected by having images that can have background or other objects present, objects being partially occluded and being amongst clutter. By having a dataset of non-iconic views, the second research problem is addressed as objects are in scenarios where context with respect to the scene can be used. Finally, a higher level of precision is provided in the MS COCO segmentation challenge. As mentioned, results are required to be instance and pixel-wise. Therefore, the objects in the dataset are annotated precisely at a pixel-level but also with coarser bounding-boxes. These goals resulted in a dataset has a total of 91 object classes of which 82 have more than 5,000 labelled instances. Considerably higher that that of PASCAL VOC. In addition to having a large number of classes the number of object instance per image is also relatively high. On average there is 7.7 instances per image, considerably higher than PASCAL VOC with 2.3 and ImageNet with 3.0.

The object classes chosen are similar to those in PASCAL VOC, where the categories should represent a common objects that are relevant to practical applications. Also the categories should be such that a high number of images that respect the core research problems can be found. The 91 categories were chosen to be at a higher-level of taxonomy such that they would be the commonly used label by a typical person. The categories and number of instances in each can be seen in Figure 2.9.

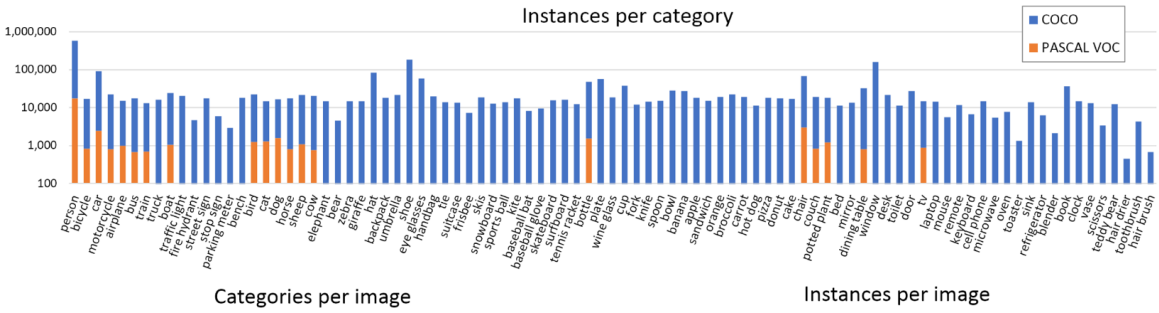


Figure 2.9: Object categories and number of instances in each in the MS COCO dataset [3].

The image collection process was done using queries through Flickr inspired by the PASCAL VOC method of collection. Having only a single categories as a query resulted

in higher chances of iconic views, therefore multiple categories were used to collect images. Once collected and annotated the total number of images in the 2015 release was 328,000, split as 165,482 for training, 81,208 validation and 81,434 for testing.

Apart from having a larger number of categories and there being a large number of instance per image (7.7), there are also other items that make this dataset more challenging. Firstly, the number of categories per image is larger than PASCAL VOC, with 3.5 compared to 1.7. Additionally, the dataset is made up of much smaller images which are typically more difficult to detect. Roughly 65% of object instances make up only 4-6% of the total image size. This is in comparison to PASCAL VOC at roughly 45%.

MS COCO uses 12 metrics to evaluate the performance of object detection, where 6 are variants of AP and the remaining 6 variants of Average Recall (AR). The AP metric is calculated in the same manner as in PASCAL VOC, however, the primary metric is also averaged over multiple IoUs. In PASCAL VOC AP is only calculated at IoU=0.5, however, in MS COCO AP is average in the range of [0.50, 0.95] at intervals of 0.05. The MS COCO AP is also evaluated on detections across ground truth image scales, as the number of small objects in the dataset is as mentioned relatively high. The scales covered is AP for small objects that have a ground truth bounding-box are less than 32^2 pixels, medium objects area between 32^2 and 96^2, and large objects with area above 96^2. Apart from the primary metric and the three image scales, AP is also calculated at two fixed IoUs. Firstly, at IoU 0.50 which results in the same metric as in PASCAL VOC and at relatively strict IoU of 0.75. The AR metric is also average across multiple IoUs, but also measure at a limitation of the maximum number of detections per image. This makes up three AR metrics where maximum detections are 1, 10 and 100 per image. Finally the remaining three metrics evaluate the AR across the same object scales mentioned earlier.

2.4 Related Work

One of the first methods to show that CNNs could significantly improve object detection was that of R-CNN [9]. The method obtains the name R-CNN based upon a CNN is used on regions of the image. Many earlier object detection approaches were used in a sliding window fashion testing all areas of an image. This can lead to a huge amount of potential testing windows especially if the object detection is done at a multitude of different scales. The method was heavily inspired by the AlexNet model that started the deep learning renaissance in 2012 winning the classification challenge in the ILSVRC. The authors of R-CNN aimed to show that the advances in classification with a model such as AlexNet could also be done in object detection. In R-CNN the CNN model is used as a feature extractor from which a class-specific linear Support Vector Machine (SVM) can be trained on top of. The AlexNet-based feature extractor is firstly pre-trained on a large dataset designed for classification, in this case the training set from ILSVRC 2012. This pre-trained model is then adapted to the new domain of object detection by fine-tuning the model accordingly. In this instance the authors fine-tuned warped training instances from the PASCAL VOC dataset. The AlexNet model was also altered to classify the 20 classes present in PASCAL VOC rather than the 1000 classes in ILSVRC. The pipeline of the R-CNN is split into 3 modules as:

1. Region proposals.
2. Feature extraction.
3. Class-specific linear SVMs.

In this first module, region proposal, there is a large number of potential methods to produce a suitable number of windows in comparison to a sliding window approach. R-CNN is agnostic to the region proposal method chosen, and in the original work SelectiveSearch [10] is used. Module two, as explained earlier, is the use of a CNN as a feature extractor. This is in the form of a 4096-dimensional feature vector from the domain-specific PASCAL VOC trained AlexNet model. These feature vectors are used in the third module, class-specific linear SVMs. In the case of PASCAL VOC a total of 21 SVMs are trained, one for each of the 20 classes in the challenge and one for a background class. The training of the SVMs is done by forward propagating a large number of both positive and negative region proposals found with SelectiveSearch and storing each 4096-dimensional feature vector to disk. After this the appropriate labels are applied to each vector and a linear SVM is optimised for the 21 classes. At test time, for a given image SelectiveSearch is used to produce around 2000 proposals. Each of the proposals are propagated through the network to extract their respective feature vectors. Each feature vector is then tested against every SVM to produce a score for each class. Finally greedy Non-maximum Suppression (NMS) is applied to remove overlapping detections. The approach outlined in R-CNN produced a significant improvement in object detection of roughly 13%, compared to previous state-of-the-art methods. Similar results were also found on the PASCAL VOC 2011/12 test set. Despite the significant improvements with a CNN-based method on region proposals there are still issues with the R-CNN. Firstly, the testing time per image is slow, at roughly 47 seconds on an Nvidia K40 GPU. Also extracting features for each proposal to train the SVMs takes a large amount of disk space and may not be feasible on all hardware configurations. Finally, as the R-CNN is made up of 3 modules the training is done in a multi-stage manner rather than end-to-end. Therefore, the loss calculation when optimising the SVMs are not used to update the CNN parameters.

The R-CNN method was improved the following year with Fast R-CNN [11] and aimed to improve speed and accuracy. One of the significant changes is that the detection training done is now end-end rather than in the multi-stage pipeline in R-CNN. Due to this the large requirements of disk space due to feature caching is no longer required. The Fast R-CNN method takes both an image and a set of pre-computed object proposals, as in R-CNN. A CNN forward propagates the entire image, rather than individual proposals in R-CNN, through several convolutional and max-pooling layers to produce a feature map. Features are extracted for each proposal in their corresponding location in the computed feature map with a Region of Interest (RoI) pooling layer. The RoI feature is calculated by splitting the $h \times w$ proposal into $H \times W$ sub-windows of size $h/H \times w/W$. Where h and w is the height and width respectively of a proposal. H and W are hyper-parameters specifying the fixed spatial extent of the extracted feature. Each sub-window has max-pooling applied and with the resulting value being placed in the corresponding output cell. Once the RoI pooling layer has been applied to a pre-computed object proposal the forward pass continues through two fully-connected layers followed by two sibling output layers. The sibling outputs are a

softmax classification layer that produces probabilities for the object classes and another layer for bounding-box regression. These two layers replace the respective external modules in R-CNN and make it possible to train the entire detection network in a single-stage. As in R-CNN, pre-training a CNN on a large classification dataset and fine-tuning towards detection and a specific object classes is done in a similar fashion. In R-CNN, the only deep network used was AlexNet [12], however, in Fast R-CNN the authors experiment with networks of different size. It was found that the deeper network VGG-16 [13] for computing the convolutional feature map gave a considerable improvement in Mean Average Precision (mAP). For a fair comparison of results against R-CNN, its CNN was the same pre-trained VGG-16 network and it was found that Fast R-CNN improves AP by 3-4% on PASCAL VOC.

However, as the name Fast R-CNN implies the main improvement is the speed in respect to both training and testing. By computing a convolutional feature map for an entire image rather than per object proposal the number of passes in the network is lowered significantly. The training time is speed up 8.8× and test per image is 146× faster. While Fast R-CNN provided improvements in both accuracy and speed, the increase in speed is only in relation to the actual object detection and assumes that the region proposals are pre-computed. Therefore, there is still a significant bottleneck per image as a region proposal method can typically take a couple of seconds.

The region proposal bottleneck was addressed in the third iteration of the R-CNN network with Faster R-CNN [14]. In this method, it was shown that region proposals could be computed as part of the network. This part is called a Region Proposal Networks (RPN) and shares the convolutional layers and feature map used for computing features with RoI pooling in Fast R-CNN. As these layers are already computed on the entire image for the classification pipeline, the added time for proposals using the RPN is negligible in comparison to a method such as SelectiveSearch. Apart from the change in how region proposals are computed, there is no difference in comparison to Fast R-CNN. An RPN takes the last convolutional feature map as input and returns a number of object proposals. Each proposals is fed into two sibling layers, similar to that in Fast R-CNN, one layer scoring how likely to be an object or background and another performing bounding-box regression. The proposals are found through a method denoted as anchors. At each sliding-window location proposals are found with user-defined reference boxes for how an object proposal may be formed. The anchors can be built based upon scale and aspect ratio altering the size. These anchors are then placed on the feature map and the sibling layers calculate the likelihood of an object and regress the anchor as necessary. Once the proposals have been found with the RPN these are placed on the same convolutional feature map as earlier and the rest of the pipeline is identical to Fast R-CNN, classifying and regressing bounding-boxes with another set of sibling layers. As the only change is the addition of computing proposals in the network with RPN, the results are similar in respect to mAP. Only a slight improvement in made on PASCAL VOC 2007 and 2012, from 66.9% to 69.9% and 65.7% to 67.0% respectively. However, the main contribution to the work is the speed-up of the entire object detection pipeline as the object proposal time is now minimal. On average processing an image on PASCAL VOC 2007 with an Nvidia K40 with Fast R-CNN including proposals took 2 seconds per image. While in Faster R-CNN with the same hardware takes 0.2 seconds per image. A speed-up of 10× from Fast R-CNN to Faster R-CNN and 250×

from the original R-CNN. The Faster R-CNN methods has also proved to be the foundation for the winning entry in multiple detection challenges including MS COCO. The results for this challenge with a VGG-16 model for Fast R-CNN were 35.9% mAP@0.5 IoU and 19.7% mAP@[0.5, 0.95]. Faster R-CNN improved this to 42.7% mAP@0.5 and 21.9% mAP@[0.5, 0.95].

Much of the recent work within object detection has been based upon the Faster R-CNN framework. This is exemplified by looking at the MS COCO detection leaderboard [16], with 15 of the 21 approaches being Faster R-CNN related as of early 2017. Firstly, the winner of the MS COCO 2015 and ILSVRC 2015 detection challenge was with the use of deep residual networks (ResNets) [15]. As is well known with CNNs, deeper networks are able to capture richer higher-level features. The authors showed that this is also beneficial in the object detection domain. In [15] an ensemble of three deep residual networks with 101 layers was trained for object detection and another ensemble of three used for region proposals with the RPN while being based on the Faster R-CNN framework. In addition to the ensemble, the winning entry also added box refinement, global context, and multi-scale testing to the Faster R-CNN.

The current leading method on MS COCO is an extension of the previously explained ResNets [15]. This method dubbed G-RMI on the MS COCO leaderboard [16] is an ensemble of five deep residual networks based upon ResNet [15] and Inception ResNet [17] feature extractors. No work has been published yet on G-RMI at this time, however, a short explanation of the entry is included in a survey paper from the winning authors [18]. The approach was to train a large number of Faster R-CNN models with varying output stride, variations on the loss function, and different ordering of the training data. Based upon the collection of models, five were greedily chosen based upon performance on a validation set. While performance on the models were important, the models were also chosen such that they were not too similar. It should also be noted that apart from the ensemble of models, G-RMI did not include any extras such as multi-scale training, box refinement, or global context which are often used in benchmark challenge entries.

Another variant is that of MultiPath [19], placing second in MS COCO 2015. Which aimed to address the many small objects present in MS COCO by modifying Fast R-CNN. Firstly, rather than only having a single classification head, MultiPath has four. Each classification head observes different scaled regions around the RoI which aims to add context around the object. The output of each of the four are concatenated for classification and regression. Also MultiPath uses skip connections. The RPN and consequent RoI-pooling in Faster R-CNN and Fast R-CNN is only performed once at a number of convolutional layers. At which point the input image has been down-sampled a number of times, therefore, small objects are potentially not represented very well any more. With the use of skip connections higher-resolution features from earlier convolutional layers can be added giving the RPN and classier information about smaller objects.

Inside-Outside Net (ION) [20] also adds contextual and multi-scale information on top of Fast R-CNN. ION was the third place entry in MS COCO 2015. The multi-scale information is also added with skip connections. Whereas global context is added through

the use of Recurrent Neural Networks (RNNs) passing information about the image both vertically and horizontally.

Global context has also been added to the Faster R-CNN framework in [21] with the use of semantic segmentation as a form of top-down information. A segmentation module is augmented onto the framework and is calculated using the same initial convolutional layers as Faster R-CNN. The segmented result is then added after the RPN and RoI-pooling is performed on both the convolutional layers and corresponding RoI segmentation area.

Additional work on adding finer details for smaller objects with Faster R-CNN was performed in [22] who aimed to improve skip connections with additional top-down information. While skip connections is useful method for finding higher-resolution features, the authors argue that with Top-down Modulation (TDM) features are taken from an appropriate lower layer. TDM is incorporated into the Faster R-CNN framework and can be trained along side it.

The use of hard example mining was conducted in [23]. In this work the problem of small objects in ILSVRC and MS COCO is also addressed. The authors present a method for training a Fast R-CNN object detector for these objects with Online Hard Example Mining (OHEM). Inspired by bootstrapping, with OHEM a modification is made to Stochastic Gradient Descent (SGD) training by selecting RoIs that the network currently has a high loss on and backpropagating accordingly.

The methods covered so far have all followed a region-based paradigm of first finding a selection of object proposals and second classifying these into one of the appropriate classes, while also regressing bounding-boxes. These methods can be computationally expensive and therefore recent work has attempted to combine the two steps into a single feed-forward CNN. These methods can be denoted single shot object detectors, with he most recent approach is that of Single Shot Detector (SSD) [24]. It is the first deep approach that does not resample pixels of features to perform object detection such as RoI pooling in region-based methods [11] [14]. Rather, convolutional feature layers are added to the end of a network and a small filter is applied on these for simultaneous localisation and classification. The truncated layers become progressively smaller and allow SSD to find objects at multiple scales. The predictors used on these layers are of pre-determined size and aspect ratio similar to the anchor boxes used in Faster R-CNN [14]. The additional layers and predictors can be added to any classification-based CNN and SSD test using the standard VGG-16 network. On top of this the authors train two separate instances of the network, one for low-resolution input SSD300 (300×300) and one for high-resolution SSD512 (512×512). Overall the higher-resolution network performs best with 1-2% improvements in comparison to Faster R-CNN on PASCAL VOC 2007 and MS COCO test-dev 2015. In terms of speed there is a considerable difference, the authors found Faster R-CNN on average took 0.14 s/image, SSD300 0.02 s/image, and SSD512 0.05 s/image on PASCAL VOC 2007 testing with a Titan X GPU.

One of the original single shot CNN-based methods for object detection was OverFeat [25] with a sliding-window approach. Methods such as OverFeat have not recently been as

popular due to deeper and more powerful networks being too computationally expensive to be run across the entire image at multiple scales. However, at the time OverFeat won the ILSVRC 2013 localisation challenge using an altered AlexNet [12] CNN. The main alteration was a regression layer for added accuracy in localisation.

A precursor to SSD was that of MultiBox [26] and the improved version in [27]. Again the goal was to directly predict the bounding-box of an object directly with a CNN for a given class. The MultiBox method was originally designed to prove that object proposals with a CNN could be an improvement of hand-engineered methods such as SelectiveSearch [10] in R-CNN [9] and Fast R-CNN [11]. MultiBox is similar to the RPN in Faster R-CNN [14] where object locations are predicted on a grid with a number of default predictions of different sizes. Additionally, MultiBox ranks the proposals according to a loss in relation to both the confidence of being an object and location of the bounding-box. With this ranking MultiBox is able produce and classify only 15 proposals per image while being competitive to other methods such as R-CNN at the time.

Another CNN method that only uses a single network is You Only Look Once (YOLO) [28] and its successor YOLOv2 [29]. In the original YOLO method, a single shot approach was taken by producing bounding box locations and class scores from a fixed grid in an image. Various combinations of these grids are used as potential bounding-boxes. However, in YOLO the accuracy of the localisation was poor and this was addressed in YOLOv2 with a number of changes. Firstly, the RPN from Faster R-CNN [14] was adapted with this use of anchor boxes. But rather than using pre-determined anchor sizes k-means clustering is run on the training set to determine what appropriate sizes. Also to address the issue of small objects not being represented in the convolutional feature map in the RPN after many convolutional operations, additional features are added from an earlier convolutional layer. Other improvements to YOLOv2 include batch normalisation, multi-scale training, and high-resolution inputs. Overall the method produces competitive results against approaches such as Faster R-CNN and SSD. But the main improvement is in speed, where at inference time is up to 182× and 5× faster than Faster R-CNN and SSD respectively.

Recently, a newer approach to region-based methods has been proposed with the use of Fully Convolutional Networks (FCNs) through the Region-based Fully Convolutional Network (R-FCN) [30]. The authors argue that in region-based methods the act of cropping features from RoIs in the same layer adds an unnatural condition. There has been an issue in the two step pipeline in region-based methods, as the classification is translation-invariant, whereas detection in translation-variant. Due to this difference region-based methods have been adjusted towards the invariant properties of classification by pooling features and classifying them. However, [30] argue that translation-variant representations are important in object detection as the position of an object inside a RoI can provide meaningful information. Therefore, [30] present their fully convolutional approach with R-FCN. The overall approach is similar to that used in region-based methods such as [9], [11] and [14]. First compute RoIs using a region proposal method and second perform classification on these regions. R-FCN uses the RPN from Faster R-CNN [14] for class-agnostic RoI computation. However, rather than extracting features with RoI-pooling, fully convolutional position-sensitive score maps are computed. The score maps are split up to represent a relative

position in a $k \times k$ grid, with each cell presenting information relative to the spatial position of an object. For example, the upper-left cell represents scores that pixels are present at that relative position to the object. A bank for position-sensitive score maps are found for each class, generating a total of $k^2(C+1)$ where C is the number of classes plus a background class. After computing the bank, the R-FCN computes a position-sensitive RoI-pooling layer for each class. For each RoI found with the RPN each cell aggregates the response from the appropriate score map from the bank of maps. While the ordering and methodology of RoI-pooling is different in R-FCN to that of Faster R-CNN the same backbone CNN can be used. In the experiments conducted by the authors a ResNet-101 network is chosen. Overall R-FCN is an improvement on the Faster R-CNN approach on benchmarks such as PASCAL VOC and MS COCO. It is also competitive with the MS COCO 2015 winning entry [15], while not having any additions such as global context or iterative box regression. Additionally it is considerably faster in training and testing in comparison to Faster R-CNN.

The use of FCNs is currently the leading method for segmentation; both semantic [31] and instance [32]. The latter, named Fully Convolutional Instance-aware Segmentation (FCIS) won the 2016 MS COCO instance segmentation challenge and is also the current second place in bounding box object detection. It uses a similar approach with position-sensitive score maps for pixel-level likelihood for an object category to produce bounding boxes. From these, instance segmentation is performed to produce the pixel-level classification. The main differences between R-FCN and FCIS is the addition of ensemble of ResNet models, multi-scale testing and training, and horizontal flipping.

2.5 Problem Statement

As outlined in the introduction, the general goal of object detection is to find objects in an image based upon pre-defined object categories. As mentioned in Section *2.2 Main Challenges*, the main challenges can be within object detection can be defined in two groups as robust-related and computational complexity and scalability-related as per [4]. The robust-related challenges are with respect to variations in the objects, this can include colour, texture, shape and size. The other challenge in this group is variations in the images which can differ in terms of lighting, viewpoint and quality. Both object and image variations can occur intra- and inter-class. These robust-related challenges lead into the computational-complexity and scalability-related. As object detection can be a quite difficult task the choice of model must be sufficient to capture such large variations. Additionally, this puts requirements on the quantity and quality of the data needed to train such a model. Based on the the works covered in Section *2.4 Related Work*, current leading methods are CNN-based. These methods are the current leaders in choice for tackling the robust-related challenges. Additionally through the use of high-quality datasets such as PASCAL VOC and MS COCO research within object detection has grown considerably within recent years. However, there is still number of challenges present in relation to both object and image variations that many not have been addressed properly yet. Many leading methods find smaller objects challenging to detect. Additionally variations in the quality of the image is an area which not many have addressed. Despite CNN-based methods being the current state-of-the-art and are becoming increasingly more complex they may yet find it difficult to generalise across many of the robust-related challenges.

Therefore, the following questions will be investigated in this work:

- *How can specific robust-related challenges be addressed in CNN-based object detector*?

3 Technical Analysis

As mentioned in Section *2.4 Related Work*, the current leading methods within object detection are within the domain of deep learning. This chapter will cover the core concepts of deep learning which will include general architecture of CNNs, typical layers and optimisation strategies. Also covered will be aspects of deep learning that are more specific to object detection with CNNs.

3.1 Convolutional Neural Networks

CNNs are an extension of artificial neural networks which have existed for decades. Neural networks consist of neurons that receive inputs and have learned parameters such that the input can be altered in some manner. In the neuron, the dot product is computed between the input and parameters. For CNNs, the key difference is the first input to the network is an image and the parameters in the neurons are filters which are trained to activate towards certain inputs. One of the first successful CNN methods was LeNet for hand-written digit classification in 1989 [33]. However, after this point, deep learning research became stagnant mostly due to the large amount of processing needed in training. The return of deep learning is often attributed to AlexNet in 2012 [12], which gave significant improvements in image classification on ImageNet.

The general architecture of a CNN is shown in Figure 3.1. The network takes an image as an input, this can be a single channel as depicted in the figure or multiple such as an colour image. Convolutional operations with learned filters are applied to an area of the input image dependent on the filter size to produce an output at a given layer shown by the red dot. The size of the filters at a given layer constitutes the receptive field of that layer. For example, a 9×9 filter has a larger receptive field than a 3×3 filter to produce a given response. Each filter is individually trained and shown as the arrows leading to the dots. In the second convolutional layer is where the network starts to be considered deep. Again convolutional operations are performed to produce an output. Depending on the architecture of the network many convolutional layers can be present, generally deeper networks are able to find more abstract features for the given task. Finally the network may have a fully-connected layer that produces confidence scores. These scores can be used to determine how well an input image represents a given class for a classification problem.

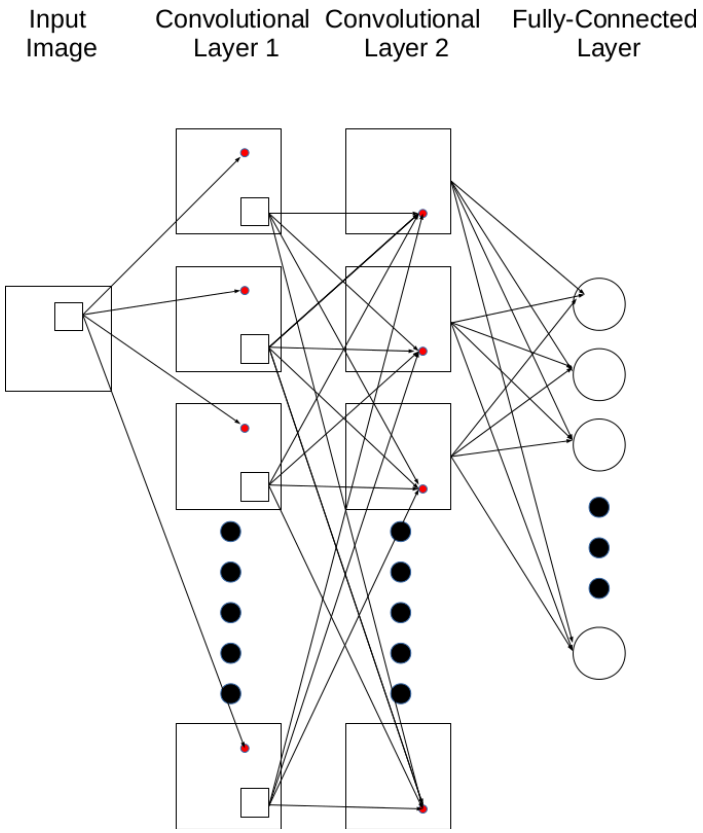


Figure 3.1: An example of a general CNN with convolutional and fully-connected layers.

The activation function within a convolutional layer is another key aspect to neural networks. The activation layer is used to add non-linearity to the network and measures how well a given convolutional operation and associated bias fires for a patch in either the input image or a previous layer. Typically activation functions output between 0 and 1 to represent this measurement. In earlier adaptations of CNNs the a sigmoid activation function was popular to map the output of a convolutional layer between 0 and 1. However, most current CNNs take advantage of the Rectified Linear Units (ReLUs) activation function. ReLU is a simple thresholding function that maps negative outputs to 0 and positive outputs are kept unchanged.

In order to learn the parameters in a CNN an optimisation strategy is required. The training process is to minimise a loss function in respect the inputs. Typically the learning is done through gradient descent with backpropgation. The intuition here is to update parameters after each forward iteration such that the loss calculated between input samples and their labels is decreased. Generally for each forward pass the loss is calculated as the average loss over a mini-batch of samples. This is both more efficient and produces a less stochastic learning process. Once the loss is found the gradient indicates which direction to update parameters and this information is backpropagated right through to the initial parameters.

A key aspect of training CNNs is how the parameters in a network are initialised. Poor

initialisation of parameters can make the training process slow or impossible if the initial operations fire the activation functions too violently. Common approaches to initialisation include sampling the weights from a Gaussian distribution and setting all biases to zero. Other alternative dynamic approaches do exist, such as Xavier initialisation. In this case the architecture of the network is measured, such as number of filters in a layer, and weights are sampled according to this information. Another initialisation method is fine-tuning parameters from a pre-trained network. A pre-trained network is typically trained on a large set of data and has learnt parameters to that given task, then by updating the parameters they can be changed towards a new task. This can provide a number of benefits. Firstly, the amount of training time can be drastically reduced as strong general parameters have already been learned. Also, if the amount of training data is sparse, fine-tuning can aid such that the risk of overfitting is reduced.

3.2 Object Detectors with Convolutional Neural Networks

This section will perform a technical analysis of some of the current leading CNN-based object detectors. This includes Faster R-CNN [14], R-FCN [30] and YOLOv2 [29]. On top of the analysis, results for the detectors will be discussed for PASCAL VOC and MS COCO. This should given an indication as to which CNN-based object detector will be used to address robustness-related challenges.

3.2.1 Faster Region-Convolutional Neural Network

A primary CNN-based object detector over the previous years has been Faster R-CNN [14] and its predecessors, Fast R-CNN [11] and R-CNN [9]. As mentioned in Section *2.4 Related Work*, 15 of the 21 current entries in MS COCO are Faster R-CNN based. The general method of Faster R-CNN can be split into two parts, region proposals and region classification. Region proposals aims to reduce the amount of windows that need to be tested at inference time in comparison to the previously often used sliding window approach. Rather than testing a plethora of potential object window locations, scales and aspect ratios, region proposals find a lower number of windows that are likely to contain an object. Additionally, it also allows for using more expensive classification techniques such as CNNs. There are a large number of different methods to compute region proposals. However, the RPN in Faster R-CNN is one of the more popular and is used in other CNN-based approaches. The proposals are efficiently computed with the RPN, as proposals are found directly in the network, sharing convolutional layers with the classification step. The framework of the Faster R-CNN can be seen in Figure 3.2, where the convolutional layers are used to compute feature maps. On the last feature map, the RPN computes region proposals. These are then placed back onto the last feature map and RoI pooling is used to compute features for classification.

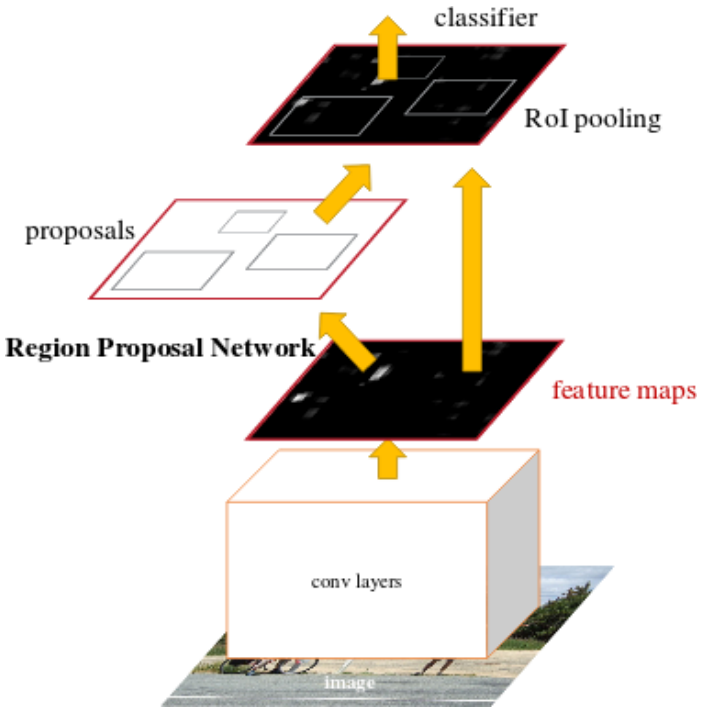


Figure 3.2: Faster R-CNN framework. A CNN computes a feature map from which a RPN finds region proposals. Given these proposals and the same feature map proposals are classed accordingly [**?**].

There are a number of different possible CNN models that can be used the compute feature maps through the convolutional layers. In the original Faster R-CNN VGG nets [13] were experimented with. However, since then more efficient and accurate networks have come forward, with one of the most popular being the ResNet [15] architecture. Both of the models will be covered in more depth later in this chapter. Standard practice is to pre-train the network for classification on ImageNet followed by fine-tuning it towards object detection. Independent of the chosen network the RPN takes as input the last feature map from the convolutional layers of the network. A small network traverses the feature map which feeds the result into two sibling fully-connected layers, a box-classification layer and a box regression layer. The box-classification layer classifies the RoI into either an object or background, with a probability being associated to each. While the box-regression layer attempts to fit the bounding-box to the object of interest. In order to take into account different scales and aspect ratios of objects in the feature map, the RPN uses a set of pre-defined RoIs at each sliding window location. These pre-defined RoIs are denoted as anchors. At each sliding window location a maximum of k possible region proposals can be computed based upon the k anchors. The anchors are user-defined into different sizes and aspect ratios. An often used default setting for the anchors is $k = 9$, which corresponds to all combinations of 3 scales and 3 aspect ratios. The computation of the k anchors at a sliding window location followed by the sibling layers is visualised in Figure 3.3.

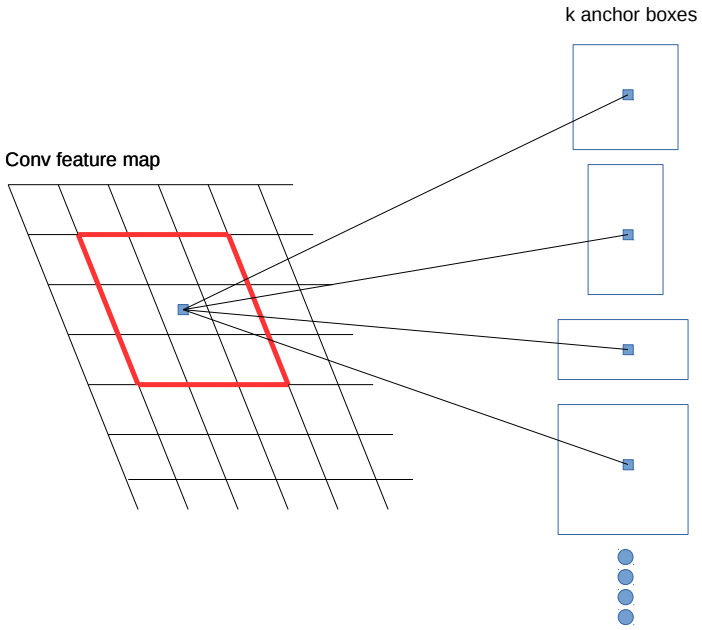


Figure 3.3: RPN framework. The k anchor boxes are placed at each sliding window location on the last feature map. The RPN uses two sibling layers to compute the classification of object or background and perform bounding-box regression.

Given the set of region proposals from the RPN, objects are classified in the $C + 1$ categories based upon the same approach as in Fast R-CNN. Where C are the total number of object classes plus one background class. Features are cropped for each proposal are their respective location from the same feature map given to the RPN. Features are computed using a RoI pooling layer that uses max pooling to convert the cropped area into a pooled map of fixed spatial extent $(H \times W)$, where H and W are hyper-parameters. In order convert each RoI into a fixed max pooled size, the RoI of size $h \times w$ is split into a grid of $H \times W$, with each sub-window being of size $h/H \times w/W$. Max pooling is then applied at each sub-window and placed accordingly in the $H \times W$ pooled layer. Following the RoI pooling layer, two fully-connected layers feed sibling layers into a classification layer and a box-regression layer, similar to those in the RPN. However, in this instance the classification layer computes the probabilities for each of the $C + 1$ classes.

3.2.2 Region-Based Fully-Connected Network

One of the current leading object detection methods is the R-FCN [30], which as mentioned in Section *2.4 Related Work*, takes a different approach to that of the region-based methods such as Faster R-CNN. The authors of R-FCN were inspired by the recent advances in FCN classification networks, such as ResNets, and argue that the addition of the RoI-pooling layer in the Faster R-CNN pipeline is unnatural and adds computational complexity. The authors hypothesise that the reasoning behind this addition is due to the trade-off between using a classification approach in an object detection pipeline. A defining factor in object detection is that the method should be able to respect translation variance, that translation of an object inside an object proposal should given a good indication as to how well the proposal fits the object. Whereas classification is more translation invariant, as the shifting

of an object in an image does not effect how the system returns it's output. The use of the RoI-pooling layer placed in between convolutional layers means that any convolutions after this point are not translation invariant as it is not region specific. Rather than using this popular feature extractor, R-FCN uses position-sensitive score maps computed by a bank of convolutional layers. The maps add translation variance into the detection pipeline by computing scores in relation to position information with respect to the relative spatial position of an object. A RoI-pooling layer is added after the score-maps, however, no convolutional operations are done after this point ensuring translation variance.

The overall approach of the R-FCN also consists of the popular two-stages of region proposal and region classification. Region proposal is done using the RPN from Faster R-CNN followed by the position-sensitive score maps and RoI pooling for region classification. The overall architecture of the R-FCN can be seen in Figure 3.4. Similar to Faster R-CNN, convolutional layers are applied on the input image and the RPN computes region proposals. After this position-sensitive score maps aid in classification.

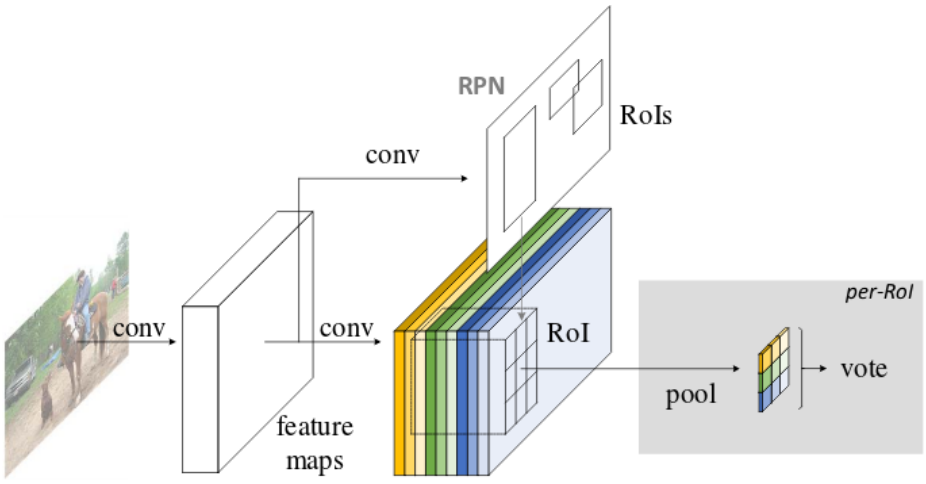


Figure 3.4: Architecture of R-FCN. Region proposals are found using the RPN followed by classification based on a bank of position-sensitive score maps [30].

The added translation variance post finding proposals with the RPN is done by producing a bank of k^2 score maps for each object category. Therefore, there are a total of $k^2(C+1)$ maps. The number of k^2 maps is due to a $k \times k$ spatial grid representing relative positions. Typically $k = 3$, therefore, nine score maps represent position-sensitive scores for a given object category. This is illustrated in Figure 3.5, each of the 9 coloured rectangles on the left of the figure represent the k^2 score maps. Each colour represents one of the relative positions. For example, the three shades of blue are positions in the bottom of a RoI, where the darkest is bottom-right, then bottom-centre and lightest bottom-right. For a given RoI placement the vote for relative position is sampled from their respective map in the bank.

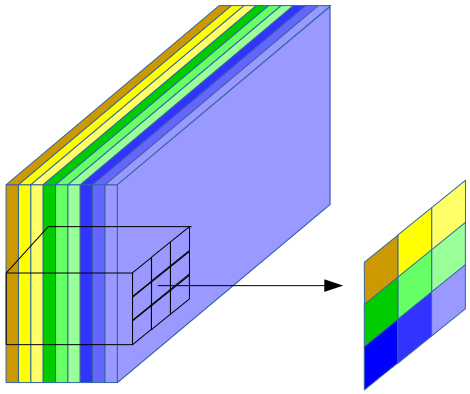


Figure 3.5: A bank of score maps are present for each object category. For a given RoI, the score is sampled from the respective position in the corresponding score map.

Once the bank of score maps have been computed, position-sensitive RoI-pooling is found for region classification. Each individual $k \times k$ bin pools from its corresponding location in the relevant score map. For example, the top left bin pools from that position in the top-left score map and so on. The RoI-pool is computed using average pooling for each bin which can be seen in Figure 3.6. The final decision for a given class is determined by a vote where each of the bins are averaged, producing a $(C+1)$-dimensional vector for each RoI.

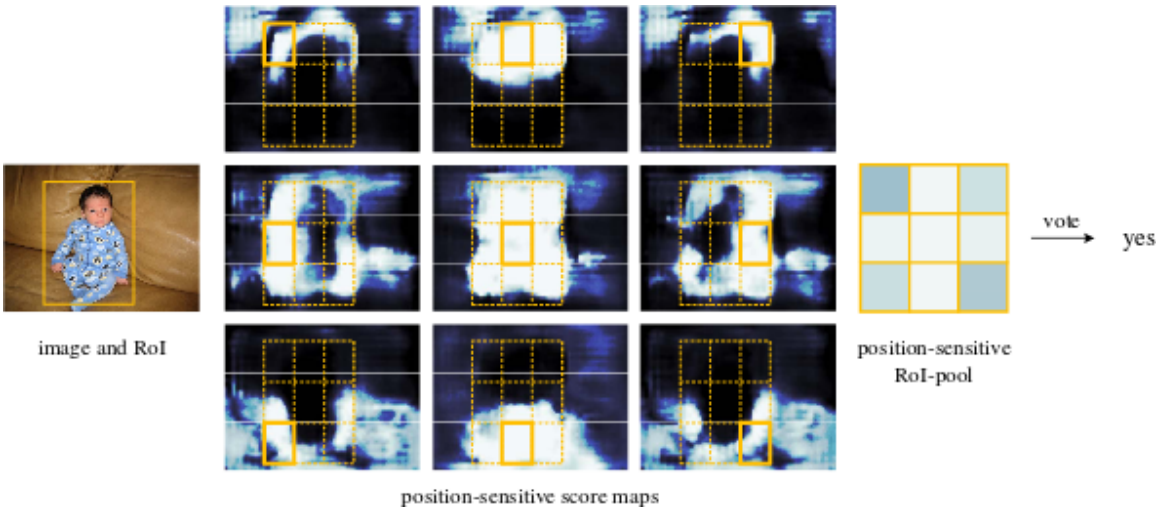


Figure 3.6: Position-sensitive RoI-pooling operation for a given class [30].

3.2.3 You Only Look Once

YOLOv2 [29] is one of the current best performing single shot detectors, with results on par with more commonly used object detectors while being considerably faster at test time. YOLOv2 uses a different approach than the common 2-step method of region proposal and region classification seen in Faster R-CNN and R-FCN by directly computing class probabilities on each RoI. Some of the distinct difference between YOLOv2 and region-based methods is the use of directly predicting bounding boxes, using a modified model,

and altering how the priors for anchor boxes are computed during region proposals with the RPN. The distinct differentiator for YOLO and YOLOv2 is that bounding boxes for a given object are predicted directly rather than predicting offsets to anchors with the RPN. This is done by splitting the image into $S \times S$ grid cells, with each cell predicting B bounding boxes. Each of the B boxes predict a total of 5 values: $[t_x, t_y, t_w, t_h, t_o]$. Where t_x, t_y are the coordinates of the centre of the given cell. The values t_w, t_h are the width and height relative to the entire image. Finally, t_σ is the confidence of how well the predicted box fits the ground truth. The location of the bounding box is determined by these values with respect to a given cell and the offset of the cell from the top left corner of the image (c_x, c_y) and the size of the anchor box is p_w, p_h. Then the bounding box predictions are calculated by:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h}. \end{aligned} \tag{3.1}$$

Finally, the probability that the given bounding box fitting given the probability of their being an object is:

$$Pr(object) * IoU(b, object) = \sigma(t_o). \tag{3.2}$$

Each of the S^2 cells predicts C conditional probabilities of it containing a given class and also being object by $Pr(Class_i|Object)$. With the predicted bounding boxes and class probabilities calculated for each cell the final detections can be determined by adjusting a threshold based upon the calculation:

$$Pr(Class_i|Object) * Pr(object) * IoU(b, object). \tag{3.3}$$

This process of using grid cells, bounding box prediction, cell class probabilities and final detections can be seen in Figure 3.7.

update figure

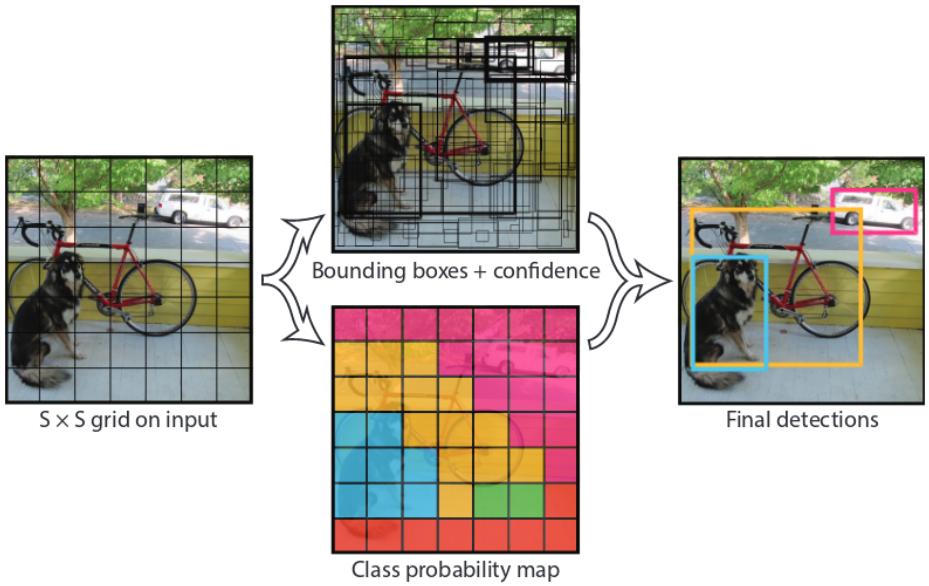


Figure 3.7: PLACEHOLDER. YOLO model.

As mentioned region proposals are found using the RPN from Faster R-CNN [14]. However, instead of using hand-picked priors for the anchor boxes, YOLOv2 proposed a method to learn more suitable sizes and aspect ratios. This is done by running k-means clustering on the annotated bounding boxes from the training set using a custom distance measurement. The custom measurement replaces Euclidean distance as these distances would create a bias due to more error on likely occurring on larger anchors. The custom distance measurement is designed for favourable IoU scores and is as follows:

$$d(box, centroid) = 1 - IoU(box, centroid) \tag{3.4}$$

where *box* is the ground truth bounding box from the training set and *centroid* is the predicted anchor box. By learning the priors YOLOv2 is able to use five anchor boxes at the same level of recall as the nine used in a typical RPN.

YOLOv2 also goes against the grain in comparison to other state-of-the-art object detectors in regards to the choice of classification model. Rather than using the common networks such as VGG or ResNets, YOLOv2 propose their own 19 layer model dubbed Darknet-19. The model is of similar paradigm to VGG nets in that it uses mostly 3×3 convolutions and doubles the number of channels after pooling, which is also present in ResNets. But it is of considerably lower complexity than VGG-16 which consists of 15.3 billion Floating Point Operations (FLOPs), with only 5.58 billion FLOPs. The baseline model has competitive results on ImageNet which can be seen in Table 3.1. The baseline can be improved using standard data augmentations and also by initially training on 224×224 images followed by fine-tuning on 448, this is also shown as Darknet-19++ in Table 3.1.

To aid in the detection of small objects the Darknet-19 model is also pre-trained on high-resolution images from ImageNet prior to training for object detection. Also fine-grained features are passthrough from an earlier layer when performing prediction. This gives features from a 26×26 feature map instead of the 13 size at the RPN. Finally multi-scale training is also performed.

Table 3.1: ILSVRC classification results for the Darknet-19 model.

Model	top-1 error (%)	top-5 error (%)
Darknet-19	27.1	8.8
Darknet-19++	23.5	6.7

3.2.4 Benchmark Results

This section will outline the results of the aforementioned CNN-based object detectors on leading benchmarks PASCAL VOC and MS COCO. This includes results on the methods with different combinations of CNN models, training data, and additions such as multi-scale training. All results are taken from the respective authors papers.

PASCAL VOC

A summary of the results on the test set of PASCAL VOC 2007 can be seen in Table 3.2. The first column denotes which method is used while also stating the underlying CNN model, for example VGG-16 or ResNet-101. Improvements to some of the baseline methods are also included in the first column if relevant. The improvements are online hard example mining (OHEM), multi-scale training (MSTR), multi-scale testing (MSTE), box refinement (BR), and global context (GC). Training data used in the various methods include the train set of PASCAL VOC 2007 (07), train set of PASCAL VOC 2012 (12), and trainval set from MS COCO (COCO). In entries when COCO is included, the detector is first trained on COCO followed by fine-tuning on 07+12. The best mAP result for each combination of training data is shown in bold.

A clear initial improvement is the use of ResNet-101 in comparison to the VGG-16 network, both with Faster R-CNN and R-FCN. ResNet-101 gives a mAP improvement of 3.2% in this scenario, from 73.2% to 76.4%. This was improvement was clear to the authors of R-FCN and therefore the only network used in their work is ResNet-101 for object detection. A small improvement of 0.2% can also been seen for R-FCN over Faster R-CNN when using ResNet-101, both with and without OHEM. The best performing detector with 07+12 training data is R-FCN with both OHEM and MSTR (80.5%), indicating that the addition of multi-scale training improves the result by 1%. YOLOv2 scores slightly lower that R-FCN and Faster R-CNN. The best performing YOLOv2 network is trained to inputs of resolution 544×544, resulting in a mAP of 78.6. When using the method of training on COCO followed by fine-tuning on 07+12, Faster R-CNN with ResNet-101 and BR/GC/MSTE scoring 85.6%. However, it is difficult to directly compare methods in this instance as the other method that uses the same training data, R-FCN with ResNet-101 and OHEM/MSTR, uses different variants of additions to the overall method. However, a general trend is that the training scheme of COCO+07+12 results in a significant improvement, with the comparable R-FCN method improving 3.1%.

Similar results can be seen on the PASCAL VOC 2012 testing set, shown in Table 3.3. A general standard for training on this test set is to include both the trainval and test from PASCAL VOC 2007 and 2012 trainval, denoted as 07++12. As in the 2007 test set, Faster R-CNN is improved with the deeper features from ResNet-101 by 3.4% when training with 07++12. The best result using the training set of 07++12 is with R-FCN with OHEM/MSTR, improving upon Faster R-CNN with ResNet-101 by 3.8%. However,

Table 3.2: PASCAL VOC 2007 results.

method	**training data**	**mAP (%)**
Faster R-CNN VGG-16 [14]	07	69.9
Faster R-CNN VGG-16 [14]	07+12	73.2
Faster R-CNN ResNet-101 [15]	07+12	76.4
Faster R-CNN ResNet-101 OHEM [15]	07+12	79.3
R-FCN ResNet-101 [30]	07+12	76.6
R-FCN ResNet-101 OHEM [30]	07+12	79.5
R-FCN ResNet-101 OHEM/MSTR [30]	07+12	**80.5**
YOLOv2 288 × 288 DarkNet-19 [29]	07+12	69.0
YOLOv2 544 × 544 DarkNet-19 [29]	07+12	78.6
Faster R-CNN ResNet-101 BR/GC/MSTE [15]	COCO+07+12	**85.6**
R-FCN ResNet-101 OHEM/MSTR [30]	COCO+07+12	83.6

again it is difficult to compare due to the additions of OHEM and MSTR. The high-resolution version of YOLOv2 is again a number of percentage points behind resulting in 73.4%. The best result is again of Faster R-CNN with ResNet-101 and BR/GC/MSTE when using COCO+07++12 as the training data with 83.8%. R-FCN with ResNet-101 and OHEM/MSTR is similarly behind as in the 2007 test, scoring 1.8% lower.

MS COCO

The final benchmark dataset to be compared is MS COCO. The results for this benchmark is more comprehensive than that shown in the PASCAL VOC challenge. mAP is calculated across a number of different IoUs. Also ground truths are split into three difference categories depending on the size of the object, denoted as either small, medium, or large. Training and testing data is done in two separate ways. Firstly, training is done on the train set of MS COCO, followed by testing on the validation set val. Secondly, training can be done on a combination of the aforementioned train and val (trainval), followed by testing on the test-dev set. The main results for the object detectors can be seen in Table 3.4. A final testing set is also used for the YOLOv2 method, denoted trainval35k. Which is made up of the same images in trainval, however, 5,000 are removed for other validation purposes.

Table 3.3: PASCAL VOC 2012 results.

method	**training data**	**mAP (%)**
Faster R-CNN VGG-16 [14]	12	67.0
Faster R-CNN VGG-16 [14]	07++12	70.4
Faster R-CNN ResNet-101 [15]	07++12	73.8
R-FCN ResNet-101 OHEM/MSTR [30]	07++12	**77.6**
YOLOv2 544×544 DarkNet-19 [29]	07++12	73.4
Faster R-CNN VGG-16 [14]	COCO+07++12	75.9
Faster R-CNN ResNet-101 BR/GC/MSTE [15]	COCO+07++12	**83.8**
R-FCN ResNet-101 OHEM/MSTR [30]	COCO+07++12	82.0

As in the PASCAL VOC challenges, baseline R-FCN performs better than Faster R-CNN with ResNet-101, with AP@.5 scoring 0.5% and AP@[.5, .95] 0.4% higher. R-FCN with MSTR gives the best result when using the training set of MS COCO only. Interestingly this best result is not consistent when comparing AP across the three object scales. For small object R-FCN with ResNet-101 is best at 8.9%, slightly above R-FCN with ResNet-101 and MSTR (8.8%). However, the later method is best for medium sized objects at 30.8%, 0.3% better than the next best method. Faster R-CNN with ResNet-101 performs best for large objects with 45.0%, 2.8% higher than the next best result, despite being considerably worse performing in the small and medium sized objects. When using the trainval set for training and test-dev for testing the best performing method is again Faster R-CNN with ResNet-101 and BR/GC/MSTE across all AP modes. Again comparison is difficult as the methods do not all have the same additions to their baselines. R-FCN with ResNet-101 and MSTR/MSTE is competitive to the best result. According to the authors of the best method [15], the additions of box refinement gives roughly 2% improvement, while global context gives about 1%. This could account for the 2.5% difference in AP@.5. Finally YOLOv2 with DarkNet-19 performs considerably worse on MS COCO. This is especially present on smaller objects scoring 5.0% AP.

3.3 Models

intro explaining backbone of models and what is covered in this section

Table 3.4: MS COCO test-dev results.

method	**training data**	**test data**	**AP @.5**	**AP @ [.5, .95]**	**AP small**	**AP medium**	**AP large**
Faster R-CNN VGG-16 [14]	train	val	41.5	21.2	-	-	-
Faster R-CNN ResNet-101 [15]	train	val	48.4	27.2	6.6	28.6	**45.0**
R-FCN ResNet-101 [30]	train	val	48.9	27.6	**8.9**	30.5	42.0
R-FCN ResNet-101 MSTR [30]	train	val	**49.1**	**27.8**	8.8	**30.8**	42.2
Faster R-CNN VGG-16 [14]	trainval	test-dev	42.7	21.9	-	-	-
Faster R-CNN ResNet-101 BR/GC/MSTE [15]	trainval	test-dev	**55.7**	**34.9**	**15.6**	**38.7**	**50.9**
R-FCN ResNet-101 [30]	trainval	test-dev	51.5	29.2	10.3	32.4	43.4
R-FCN ResNet-101 MSTR [30]	trainval	test-dev	51.9	29.9	10.8	32.8	45.0
R-FCN ResNet-101 MSTR/MSTE [30]	trainval	test-dev	53.2	31.5	14.3	35.5	44.2
YOLOv2 DarkNet-19 [29]	trainval35k	test-dev	44.0	21.6	5.0	22.4	35.5

3.3.1 VGG

A popular model for classification and object detection tasks are the VGG nets [**?**]. At the time of their creation they displayed superior results on the ImageNet classification challenge in 2014, largely due to their significantly deep networks compared to previous CNN approaches. The network is overall simple having 16 or 19 convolutional weight layers stacked on top of each other that only have filters of size 3×3. The overall architecture of the 19 layer VGG-19 network can be seen in Figure 3.8. The number of channels in the convolutional layers starts at 64 and increases by a factor of 2 after each max-pooling layer to a maximum size of 512 channels. The convolutional layers have zero-padding added to preserve the spatial resolution. Therefore, the volume is only decreased by the five max-pooling layers that consist of 2×2 windows at a stride of 2. After the final max-pooling layer there are a total of three fully-connected layers, two with 4096 outputs and a 1,000 output layer for the number of classes in ILSVRC.

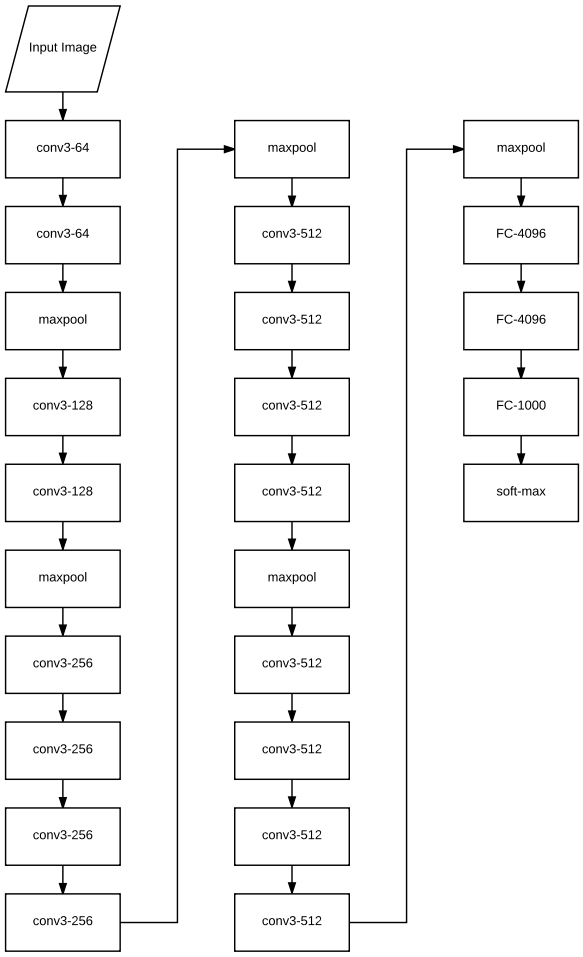


Figure 3.8: Core concept of residual blocks used in ResNets.

At the time of creation the VGG nets used relatively small receptive fields in comparison to other CNN classification networks. Rather than having larger 7×7 filters, VGG nets use multiple 3×3 filters. An advantage to this there are more functions each with their own non-linear ReLUs, which has the effect of creating more discriminative functions.

As mentioned, the results on ILSVRC were amongst the top performing in 2014. The entry took advantage of ensemble methods by averaging the softmax outputs of their top two best performing complimentary models. On top of using an ensemble, the networks were trained and tested at multiple scales. Additionally, multiple crops were taken from a given image and their softmax results averaged to give the output for each model. Using these strategies with VGG-19 models the top-1 error was 24.4% and top-5 7.1%. Despite these results there are a number of issues with the VGG networks. Firstly, the networks are notoriously difficult to train. The aforementioned 16 and 19 layer networks were initialised using a shallower 11-layer network as convergence was so hard to achieve. Secondly, despite using smaller receptive fields the number of parameters in the networks is very large. Apart from training difficulties, this puts a large requirement on this size of GPU needed.

3.3.2 Residual Networks

After the impressive advances of various challenges with ResNets in 2015 the use of these have become a standard for object detection systems, with many of the entries in MS COCO and ILSVRC being based upon ResNets. The intuition of deep neural networks is that as the model becomes deeper richer representations of the original input are found. However, as shown in the work of ResNets [15], it is difficult to train deeper versions of commonly used CNN architectures. Before the appearance of ResNets one of the standard architectures were VGG nets [13], consisting of 16 to 30 layers. The intuition of deeper architectures lead to better networks was explored in [15] by stacking additional layers and creating a 56 layer CNN. It was found that stacked deeper networks have a degradation problem and the networks converge to a testing error higher than the corresponding shallower networks. A hypothesis could be made that this is due to a deeper network overfitting the dataset, however, it was determined that this was not the case as deeper models also exhibit higher training error. The solution to this degradation problem was to to construct deeper models using identity mapping and residuals, dubbed a deep residual learning framework. With this framework a given layer learns a residual mapping between the previous layer output and operations on the output. Using this reformulation the training error in the current layer should be no greater that the previous. This core concept of a residual block is visualised in Figure 3.9. The input x is passed into the block where a mapping is computed with two weight layers with convolutional operations with a ReLU operation between them, representing an alteration with $F(x)$. After this the original input (identity) is added through a shortcut connection to the mapping by $F(x) + x$. This formulation forces the convolutional layers to compute weights to learn the residual mapping $F(x)$.

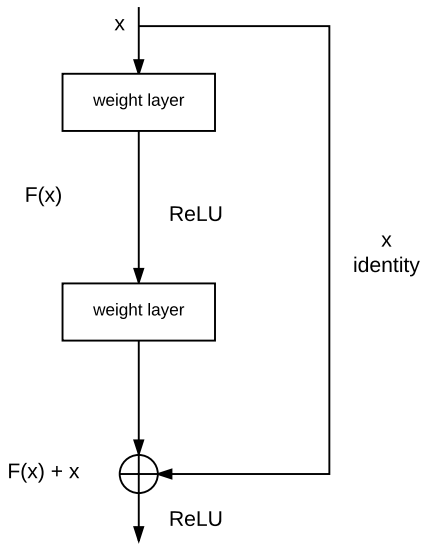


Figure 3.9: Core concept of residual blocks used in ResNets.

The formulation of $F(x) + x$ requires that the dimensions of F and x are equal. In situations where this is not the case a linear projection $W_s x$ is performed on x such that the dimensions match. The final formulation of a residual block is:

$$y = F(x, W_i + W_s x) \tag{3.5}$$

where W_i are the weights of the convolutional layers.

In the original work, experiments on a number of different architectures with residual blocks are conducted. Firstly, ImageNet classification is evaluated using naively stacked plain networks and ResNets. The plain networks are inspired by VGG nets [13] with two design criteria. Firstly, for a given output feature map size the number of filters must be equal. Secondly, if the feature maps size is halved the number of filters in the layer are doubled. The ResNets are variants of these plain networks but with residual connections in each block. In order to perform a fair comparison the linear projection performed in ResNets when dimensions are altered is done with zero-padding so that no extra parameters are added. Both sets evaluated are with 18 and 34 layers. An example of plain and ResNets can be seen in Figure 3.10.

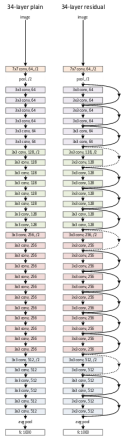


Figure 3.10: PLACEHOLDER. Overview of 34 layer plain and residual networks.

On top of the use of residual blocks, ResNets are also trained with scale augmentation and batch normalisation. During inference, multi-scale testing is conducted. Results on ImageNet validation top-1 error showed that the use of residual blocks aided in the optimisation of deeper architectures. Table 3.5 shows that the deeper plain networks exhibited troubles in optimisation with increased error with a deeper network. However, ResNets provided a decrease of 2.85% between the two respective architectures.

Table 3.5: Top-1 error(%) on ImageNet validation set.

	plain	**ResNet**
18 layers	27.94	27.88
34 layers	28.54	25.03

Haven shown that ResNets aid in optimisation of deep networks, the authors experimented with even deeper networks of 50, 101, and 152 layers. Due to concerns in the training time the residual blocks are altered in comparison to that shown in Figure 3.9. The new block shown in Figure 3.11, F consists of 3 convolutional layers of size $1 \times 1, 3 \times 3,$

Table 3.6: Results of various deep ResNet architectures on ImageNet validation set.

Method	top-1 error (%)	top-5 error (%)
ResNet-34	21.84	5.71
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	19.38	4.49

and 1×1. The two sets of 1×1 layers are used to reduce complexity by reducing the input to the 3×3 layer and restoring the resulting output.

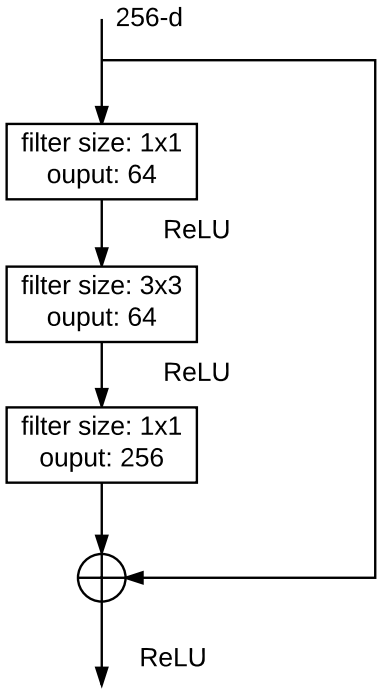


Figure 3.11: Residual block used in deeper ResNet architectures.

The deeper ResNets prove to give impressive results on the ImageNet validation sets as seen in Table 3.6, with the very deep ResNet-152 providing the lowest error.

3.4 Ensemble Methods

Ensembles of classifiers is a popular method to increase the performance of many machine learning application and problems. In object detection, most current top performing systems are ensemble based. As mentioned in Section *2.4 Related Work*, this includes the top two performing methods on MS COCO that use Faster R-CNN ensemble with variants of ResNets. This section will give an overview of ensemble methods in machine learning, including some popular methodologies. The section is largely inspired on the concepts

from the comprehensive of ensemble methods in regards to methods and applications in [34].

One of the main goals of ensemble systems is to reduce the variance incorporated in the training process. By reducing variance a key issue that appears in the bias-variance trade-off problem in machine learning can be addressed. Bias is the error that arises from incorrect assumptions in the learning algorithm, high bias can result in an algorithm to miss important patterns in the given problem. Whereas variance is fluctuations in the training data, if there is a high amount of variance a model can overfit to random noise. Typically systems with low bias tend to have high variance and have models that are more complicated. Therefore, in ensemble methods a goal can be to have multiple classifiers that have a similarly low bias but are different in regards to the variance in training data. By combining these models the overall variance is reduced and hence accuracy improved. An example of having different variance is to train classifiers on different subsets of the data. By doing this the assumption is that the classifiers will make different errors on a given data point, however, by combining the classifiers the errors will be cancelled out by the increased strength from lower individual variance. Each classifier is considered an ensemble member in the overall system and can have be used in one of two settings. Firstly a member can be used for classifier selection, here each classifier is trained such that it is an expert in a local part of the feature space. Then during inference one of the members are selected to answer the problem based upon a distance measurement of the data in the feature space. Alternatively the members can be weighted according to their distances to the data and combined to produce a decision. The second way in which ensemble members can be used is in classifier fusion. With this method all members are trained over the entire feature space and fused to make a composite classifier. Due to differences in training, such as ordering of training data, the individual members are slightly different and fusing them leads to lower variance.

3.4.1 Building an Ensemble System

According to [34] there are three main strategies to building an ensemble system. Namely:

1. Data sampling and selection: selection of training data for individual classifiers.
2. Training member classifiers: specific procedure used for generating ensemble members.
3. Combining ensemble members: combination rule for obtaining ensemble decision.

The firstly strategy aims to increase the diversity of the individual ensemble members. A common method as mentioned earlier is to train the members on different subsets of the training data. Ideally the members should not give the same output for a given data point, otherwise the ensemble is superfluous. While important that members have their individual strengths in producing correct predictions that are different, even more important is that the members produce different errors. Again, if the members produce the same errors on a data point it is not possible to reach the correct outcome. However, if members produce different errors there is the potential that individual member error can be fixed when combining outcomes. The second strategy is in regards to how the members are trained. Variability in the ensemble can be reduced by using different strategies. Third is the final step in an ensemble system, how to combine the members. This step is dependent on the type of

output from the classifiers. For example, a support vector machine may only return the class label without any additional information. In this instance the popular choice is to use a majority voting strategy where labels with the highest sum is taken as the ensemble output. However, if the output of the members is continuous, such as with accompanying confidence values in neural networks, more options are present. These can include multiple arithmetic methods such as mean, average, minimum, maximum and median.

4 Design

4.1 Design Overview

Now that an analysis of the technical aspects of object detection with deep learning has been conducted an overview of the design of the system will be made in this section. Multiple choices can be made with respect to the overall architecture of the CNN-based object detector. As covered in Section *3.2 Object Detectors with Convolutional Neural Networks*, two of the best performing systems are Faster R-CNN and R-FCN. Both methods have similarities in their overall architecture. Such as taking advantage of an RPN to efficiently find region proposals. Additionally the current core classification model used in both is the ResNet architecture. As the addition of ResNets significantly increases performance the use of these in this work is deemed as crucial part. However, the choice of either Faster R-CNN or R-FCN is not immediately as clear. Both methods perform similarly with respect to benchmarks such as PASCAL VOC and MS COCO. However, as the decision has been made to incorporate ResNets a decision on this matter was indirectly made. The GPU available in this project while being large in regards to memory was only available to train R-FCN with the ResNet-101 model. Unfortunately, due to the internal architecture of Faster R-CNN the 8Gb memory on the NVIDIA GPU was not able to store all parameters while training a Faster R-CNN with ResNets. However, due to the more efficient classification module in R-FCN, a ResNet backbone could be trained.

As mentioned, leading object detection systems take advantage of ensemble methods. However, many of them are trained with regards to the internal architecture and not specifically training experts towards solving specific challenges. Therefore, the system in this project will take advantage of the first point in Section *3.4.1 Building an Ensemble System*, namely data sampling and selection. The aim will be to train R-FCN with ResNet-101 on different subsets of training data with the aim to create expert ensemble members in regards to factors present. Two separate factors will be chosen, one with respect to variations in the object and the other in terms of image variations. The first factor chosen in object size, as seen in Section *3.2.4 Benchmark Results*, in general object detection systems find it challenging to detect and classify smaller objects. Therefore, if a system can be trained towards a subset of sizes in the training data, ideally the individual ensemble members will increase their performance on the respective sizes. The second factor chosen is in with image quality. As mentioned in Section **?? ??**, the quality of an image can be a factor in the overall performance of CNN-based classification systems. Therefore members will also be trained towards subsets data split based upon this. Lastly, individual members predictions much be combined in an ensemble system. Therefore, approaches must be taken to combine outputs. The combination strategy is greater than only voting on which class a potential object is associated to. Bounding-boxes and the confidence of each detection is used in the calculation of metrics in both PASCAL VOC and MS COCO. Therefore these must be combined in the ensemble system as well.

Based upon these issues the following design requirements are set with respect to the previously discussed items.

- Object Detector Architecture.

  - CNN-based method.
  - ResNets as backbone model.

- Ensemble Data Sampling and Selection.

  - Ability to measure object and image variations with respect to:

    * Object size.
    * Image Quality.

- Ensemble Training of Classifiers.

  - Must be kept constant to measure effect of differing data sampling strategies.

- Ensemble Combination.

  - Method to combine individual bounding-boxes and confidences between individual ensemble members.

Now that the general requirements have been outlined the following sections will cover the architectural considerations to ensure that the above requirements can be met.

4.1.1 Training R-FCN

The training of the R-FCN object detectors will be done in the Convolutional Architecture for Fast Feature Embedding (Caffe) [35]. This was chosen due to the research being provided by the authors of R-FCN through training code and pre-trained Caffe models. However, as there is the requirement mentioned in the previous section, that in order to combine detections between ensemble members the detection must be found based upon the same input to the model. One solution to this is to use the region proposals found using the R-FCN's RPN. In a standard R-FCN the RPN is an internal part of the network and is trained end-to-end. However, as these proposals must be constant between all ensemble members this method is not appropriate. Additionally, due to the nature of the Caffe framework, once a network has been defined and trained it is difficult to change it. For example, a standard R-FCN takes the entire image as inputs but in this work the requirement is that it takes smaller region proposals. The solution to these points is to train the networks using a method inspired by the 4-step alternating training method presented by the Faster R-CNN authors [14]. The process can be seen in Figure 4.1.

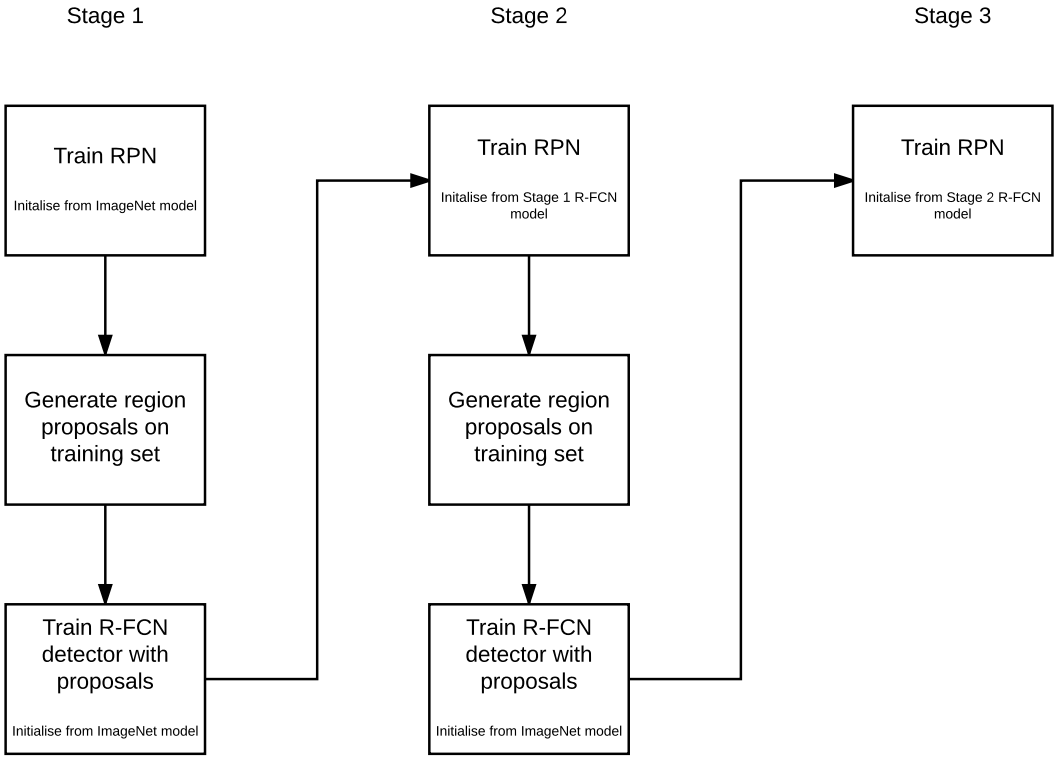


Figure 4.1: Flow chart showing the 4-step alternating training method.

In this approach the overall network in trained in multiple steps rather than an end-to-end method. In the first step a RPN is trained to determine region proposals, the RPN is initialised from a pre-trained ImageNet model and fine-tuned to the proposal task. Next a R-FCN is trained based upon the proposals found in the previous step. This network is also initialised with a pre-trained ImageNet model. In step three, another RPN is trained but initialised using the R-FCN from step two. In this step the convolutional layers that are shared between the R-FCN and RPN are fixed and only the layers unique to the RPN are updated. By training a model with this approach a testing image is able to run through the same steps as a R-FCN trained end-to-end, however, as the networks are split into different models it is also possible to use the stages of the method individually. Creating a solution for finding region proposals with an RPN and having a R-FCN that can take the proposals as inputs.

An additional benefit to training R-FCNs in this manner is that as the aim is to train ensemble members to different subsets of data, once a baseline model has been created only one part needs to be re-trained. This being the final step in stage 2, training the R-FCN detector. The RPN in stage 3 should be kept constant based on the baseline model as it will provide the shared proposals for test images. Therefore, once a systematic approach has been found for splitting data for both train and test based on the data sampling and selection requirements the detection part of the R-FCN can be trained towards its expert area. The following sections will explain how the subsets of data will be selected.

Object Size Data Sampling

The area of a region proposal gives an indication as to the approximate size of a potential objects. Therefore, the area for all proposals on the training set can be computed from the output of the second step in stage 2 shown in Figure 4.1. Once computing the area of all proposals an appropriate split of the data can be determined depending on the distribution for the given training data set. The main requirement in creating the subsets of data is that equal number of ground truth samples should be present in both.

Image Quality Data Sampling

There are many choices for computing the quality of an image. A popular area of research for this purpose is Image Quality Assessment (IQA). These methods aim to determine the subjective quality of an image. There are two popular forms of IQA, Full-Reference Image Quality Assessment (FR-IQA) and No-Reference Image Quality Assessment (NR-IQA). FR-IQA approaches require the original, undistorted reference image in order to determine quality. Whereas, NR-IQA do not have this information available. As the aim is to determine the level of image quality on one of the benchmark datasets no reference image is present, therefore, an NR-IQA method is required. Current state-of-the-art within NR-IQA is also deep learning based and works are typically trained on IQA datasets. Such datasets include Laboratory for Image & Video Engineering (LIVE) dataset [36] [37], TID2013 [38] and CSIQ [39]. The datasets consist of source reference image and have artificially created distorted counterparts with varying levels of distortion. Distortions include, such as in the LIVE dataset, JPEG2000 compression, JPEG compression, additive white Gaussian noise, Gaussian blur and bit errors from a fast fading Rayleigh channel. Models can then be trained to predict subjective quality based on ground truth user determined quality measurement.

Based upon this, an NR-IQA method can be used to determine the level of image quality with respect to a number of different distortions. Then as in object size training, if possible, the data will be split into different training subsets. A leading CNN-based NR-IQA method will be used and the specific network and implementation details will be covered in Section *5.2 Image Quality Assessment*.

R-FCN Training

Training of the baseline R-FCN model shown in Figure 4.1 is done using SGD optimisation with largely the same parameters across the five different training parts. The parameters are adapted from [30] and can be seen in Table 4.1. All models start with a base learning rate of 0.001 which is dropped by a factor of 0.1 once in the process. This is done after 80,000 iterations for the R-FCN models and after 60,000 for the RPNs. The learning rate is controlled with a momentum of 0.9 and weight decay of 0.0005. The two R-FCN models are trained for 120,000 iterations, while the three RPNs are trained for 80,000.

Both networks are trained with a batchsize of one example per iteration. In the RPN models the batches are simply one ground truth example per iteration. Whereas, the training of the R-FCNs sample mini-batches of size 128 from a given image. These mini-batches can consist of both object class samples and background samples. The only data

Table 4.1: Common SGD optimisation parameters for the 5 training parts of the baseline R-FCN model.

Parameter	**Value**
Base learning rate	0.001
Learning rate policy	step
Gamma	0.1
Momentum	0.9
Weight decay	0.0005

augmentation used in training is horizontal flipping of images, effectively creating double as many training examples.

5 Implementation

5.1 Resolution-Aware Object Detection

Object detectors are commonly more accurate on objects that cover a larger number of pixels in an image. This is intuitive as objects with a lower resolution objectively have less details that can describe them. The poorer performance can be seen in Table 3.4, for all object detectors the AP is considerably lower for smaller objects in comparison to both medium and large. The best performing detector from [15], has an AP difference of 35.3%, from 50.9% for large objects to 15.6% for small. A potential method of tackling this issue is to train multiple detectors on separate partitions of the training data according to the size of the object. While deep-based CNN have millions of parameters to generalise from training to tesitng, the difference between small and large objects may skew the learning towards the latter. In order to test this hypothesis an initial test will be conducted on the PASCAL VOC dataset. However, PASCAL VOC does not have the same definition of objects sizes as in MS COCO. Therefore, the distribution of the bounding boxes from the training set must be analysed in order to determine an appropriate split of data based on object size. This was done by parsing all of the bounding box coordinates in the 07++12 training set and calculating the area. A histogram of the area can be seen in Figure 5.1. There is a clear tendency to smaller objects in the training set with a clear skew towards the left of the figure.

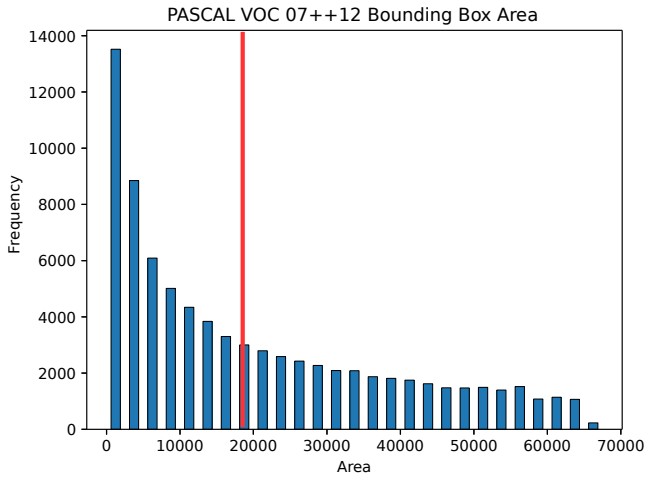


Figure 5.1: Histogram of the PASCAL VOC 07++12 bounding box area.

update following explanation

Ensemble of above networks proved not to be possible due to Caffe network architecture. Once networks have been trained as an end-to-end method, the networks were not able to take object proposals as inputs. Solution instead to train the network in 5 stages. Much slower. Result on VOC2007: 79.59 vs end to end: 76.35

While Table **??** shows that it is possible for a R-FCN network to be optimised to objects of a certain size, a more powerful approach is to combine the two networks in an ensemble during testing. Given a potential object, the ensemble should weight one of the networks

outputs accordingly. A simple example in this case would be to weight one of the outputs by 100%, if the potential object is below or above the threshold found in the training data. The results gathered in Table **??** are much more naive than this, simply splitting the testing data and only using one trained network at a time. Object proposals will needed to be known during inference to ensure that the outputs of the networks in an ensemble are performing classification in the same locations. On top of this, new networks will needed to be trained on subsets of smaller and larger data due to the internal architecture of Caffe networks. The networks in the ensemble will need to take object proposals as inputs, however, the above mentioned networks are trained based upon the end-to-end versions of the R-FCN. These do not take proposals as inputs but rather have an internal RPN. Therefore, the new networks are trained using 4-step alternating training used in the original Faster R-CNN paper [14].

A potential shortcoming of using RPN proposals as inputs to training a R-FCN rather than using ground truth bounding boxes is that a RPN likely finds many more examples of objects that actually exist in an image. For a given image, an RPN may find hundreds of potential objects despite an image only containing a couple of true positive examples. This can be solved by setting the proposals with the highest confidence as the ground truth examples and labelling the remaining proposals as the background class. This is a stark comparison to the end-to-end training approach as there are now many more training examples and a large skew towards the background class. Whereas the end-to-end approach does not train on any background examples. With this approach, the total number of training examples is increased from 80,116 to 9,979,345. The median of the almost 10 million proposals is 4,684 pixels, significantly smaller than the threshold of 19,205.5 determined using only ground truth boxes. This large increase in training examples and skew in data poses a question on how to split the RPN proposals such that two networks can be trained towards small and large objects respectively. As mentioned earlier, a pre-requisite in creating subsets of data is that the multiple sets should be roughly equal in size in order for training to be conducted fairly. If the subsets were split by the median of the RPN proposals (4,684) the two sets of data would have equal numbers of examples. However, upon inspection there seems to be a large skew in RPN proposals to smaller objects as there are significantly more true positive examples in the subset of data containing larger objects. This can be seen in Table 5.1, where despite there being an almost even split in data subsets there are significantly more ground truth annotations in the RPN_{larger} subset.

Table 5.1: My caption

Data	**RPN**$_{small}$	**RPN**$_{larger}$
Ground Truth	19,992	60,116
Background	4,969,369	4,929,297
Total	4,989,361	4,989,413

Another option is to use the threshold of 19,205.5 found on only ground truth boxes used in the initial test. The data distribution based on this threshold can be seen in Table 5.2. In this instance there is significantly more data in the RPN_{larger} subset, however, the skew is solely due to the many more background examples. The ground truth annotations are shared equally with 40,058 in each.

Table 5.2: My caption

Data	**RPN**$_{small}$	**RPN**$_{larger}$
Ground Truth	40,058	40,058
Background	3,528,370	6,370,859
Total	3,568,428	6,410,917

As the overall goal of object detectors is to find objects within the C classes, the decision was made to use the threshold of 19,205.5 to create the split in data. Despite there being significantly more background examples.

A baseline 4-step R-FCN with ResNet-101 model was trained on all the data (07++12).

Table 5.3: My caption

Train Data	**AP**
Small	46.74%
Large	62.48%
All	**79.59%**

Table 5.4: My caption

Train Data	**AP**
Small	**55.00%**
Large	10.86%
All	43.80%

5.2 Image Quality Assessment

To evaluate the amount of distortions in the dataset a method for IQA is needed. A recent state of the art method is that of Deep IQA [40]. Deep IQA is a CNN-based No-Reference (NR) IQA method that can be trained to measure the visual quality of an image. It is deeper than previous deep-based IQA methods with the architecture being inspired by VGG nets [**?**]. Deep IQA consists of 14 convolutional layers, 5 max-pooling layers and 2 fully-connected layers. The architecture is shown in Figure 5.2. The convolutional layers are all 3 × 3 convolution kernels and activated using ReLU. Inputs to each convolutional layer are zero-padded to ensure output size is equal to the input. Max-pooling layers consist of 2 × 2 sized kernels. The network is trained on mini-batches of 32 × 32 patches. During inference non-overlapping patches are sampled from the image and image quality scores are predicted for each instance. The patch scores are averaged for the final score for the entire image.

Table 5.5: My caption

Train Data	AP
Small	21.28%
Large	**80.10%**
All	75.14%

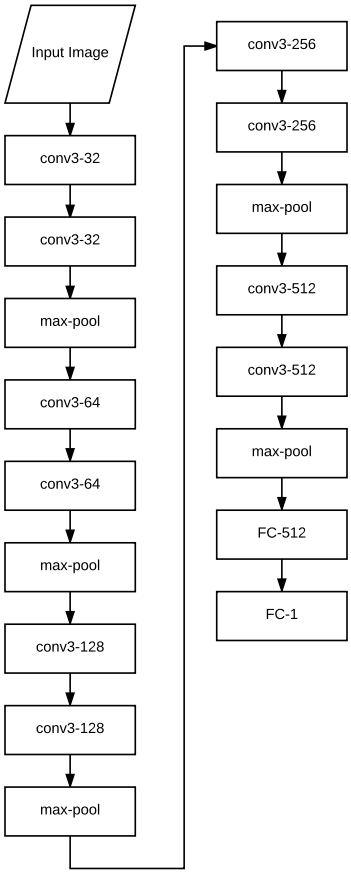


Figure 5.2: Architecture of the Deep IQA network. Notation for convolutional layers are conv(receptive field size)-(number of channels) and fully-connected layers are FC(number of channels).

Training Deep IQA requires a database of annotated images with both reference images and distorted counterparts. The following section will outline the database used in this project for IQA training.

5.2.1 LIVE Image Quality Database

Deep IQA assesses three different datasets for this purpose. These are LIVE which consists of 5 different distortions [36], TID2013 [38] with 24 different distortions and CSIQ [39] with 5 types. For simplicity purposes the only LIVE dataset is chosen for this project. The dataset was made for the purposes of evaluating the subjective visual quality of images in regards to the five distortion types. The distortions are generated from 29 colour reference images that are of both high-resolution and high quality. An example of images from the

dataset can be seen in Figure 5.3. The references image were collected to have a wide variety in different content, this includes faces, people, animals, nature and man-made objects.

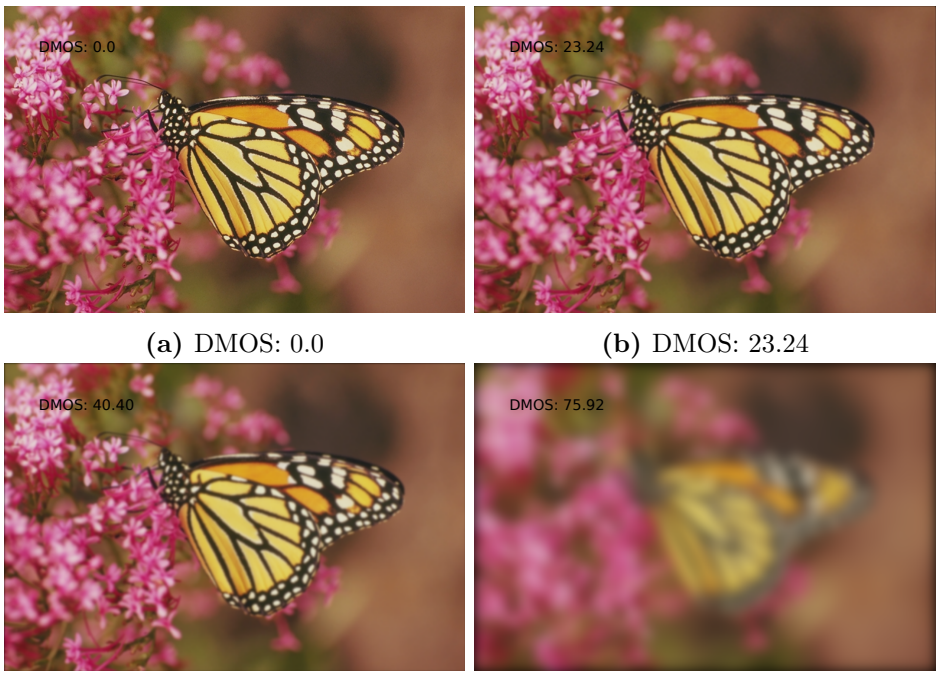
Figure 5.3: Examples of reference images from the LIVE dataset.

The five distortions generated from the reference images are Gaussian blur, white noise, JPEG compression, JPEG2K compression and fast fading. By varying the parameters used in creation of the distortions a larger database is created for each type. The total number of images is 982 where 174 are for Gaussian blur, white noise and fast fading. JPEG and JP2K compression have 233 and 227 images respectively. The distorted images were created as follows:

- Gaussian blur: blur is added to the images using a circular-symmetric Gaussian kernel of standard deviation σ_B. The values of σ_b are sampled between the range of 0.42 to 15 pixels.
- White noise: Gaussian white noise of standard deviation σ_N is added to all RGB pixels. Firstly, pixel values are scaled to between 0 and 1. σ_N varying between 0.012 and 2.0 is added, afterwards pixel values are rescaled back between 0 and 255.
- JPEG compression: compression artefacts are added to the reference bitmap images with JPEG at bit rates between 0.15 Bits per Pixel (BPP) to 3.34 BPP.
- JP2K compression: artefacts added ranging between 0.028 BPP to 3.15 BPP.
- Fast fading: this distortions represents errors that can occur when a JP2K bitstream is transmitted over a wireless channel. The receiver signal-to-noise-ratio is varied between 15.5 to 26.1dB for bit errors.

Subjective image quality values were calculated by showing human subjects all images, including reference images, and asking them to rate the image as either bad, poor, fair,

good, or excellent. The rating was done using a slider on a graphical interface with the five possibilities being evenly spaced. A value between [1, 100] was then found depending on where the subject paced their rating. Difference Mean Opinion Score (DMOS) were calculated for each image and averaged between all users for the final image quality annotation for an image. A low DMOS represents high image quality and a high DMOS is a low quality image. Figure 5.4 shows an example of an image with four varying levels of Gaussian blur and their DMOS values.



(a) DMOS: 0.0 **(b)** DMOS: 23.24

(c) DMOS: 40.40 **(d)** DMOS: 75.92

Figure 5.4: Four example images from the distortion set. Respective DMOS scores are shown on the image and below.

Given the annotated DMOS values a system can be trained to predict the image quality of an image. The following section will outline how to train Deep IQA for this purpose.

5.2.2 Training Deep IQA

As per the original authors [40], the models are trained for each distortion type in the LIVE IQA second release dataset [36] [37]. In the original work the Deep IQA model is trained for all five distortion types present in LIVE. While this can provide insights into general image quality, individual models are needed to create a more powerful ensemble. The NR model was provided by the authors and by taking advantage of the fine-tuning technique, models for each of the individual distortions can be trained in a timely manner. Fine-tuning takes a previously trained model and uses these parameters as a starting point, rather than other commonly used initialisation techniques such as using a Gaussian distribution. As the model is already trained towards all distortions the assumption can be made that only a shorter training cycle is necessary to the new task. The five fine-tuned models are trained

in a similar manner to that of the provided model following the guidelines in the original work [40].

In the LIVE dataset there are 29 reference images from where the respective distortions have been created. When training, the reference images are split into 17 training images, 6 validation images and 6 test images. The deep IQA models are trained using mini-batches consisting of a total of 128 patches per forward/backward pass. The patches are sampled from four randomly selected images from the training split and each image accounts for 32 of the 128 patches in the mini-batch. This process is continued until no more patches are available for mini-batch sampling. This constitutes a completed epoch and all patches are again available for the next epoch. The model provided by the authors was trained for 3,000 epochs, however, as mentioned fine-tuning can drastically reduce the number of epochs required. Therefore, the models for the five distortions are trained for only 500 epochs. The optimisation method for parameter updates is Adam [41]

explanation of adam solver

. The optimisation settings for Adam are unchanged to that of those used in training the original model. They are as $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and $\alpha = 10^{-4}$. A total of 10 models are trained for each distortion type, each on their individual random split of the 29 reference images. After each of the 500 epochs the model is evaluated on the validation set and the epoch with the best performance is chosen as the final model for testing. The evaluation metrics used for both the validation and test set is Pearson Linear Correlation Coefficient (LCC) and Spearman Rank Order Coefficient (SROCC).LCC is used for prediction accuracy as it is a measure of the linear correlation between two sets of data. SROCC evaluates the prediction monotonicity by measuring the rank correlation between the two sets. For both metrics a value of +1 indicates a positive correlation, 0 is no correlation, and -1 is a negative correlation.

The mean results for each of the distortion types can be seen in Table 5.6. Each best performing model on the respective validation sets are run on the testing sets and averaged.

add results for all model

Table 5.6: Average Results

Distortion Type	LCC	SROCC
Gaussian Blur	0.9750	0.9681
White Noise	0.9957	0.9887
JPEG	0.9805	0.9523
JP2K	0.9788	0.9600
FF	0.9679	0.9505

5.2.3 PASCAL VOC Data Split

Each model for the five distortion types and run through the 07++12 dataset in order to give an indication to the respective distributions, as done for the object sizes in Sec-

tion *5.1 Resolution-Aware Object Detection*. The distributions can been seen in the histograms in Figure 5.5.

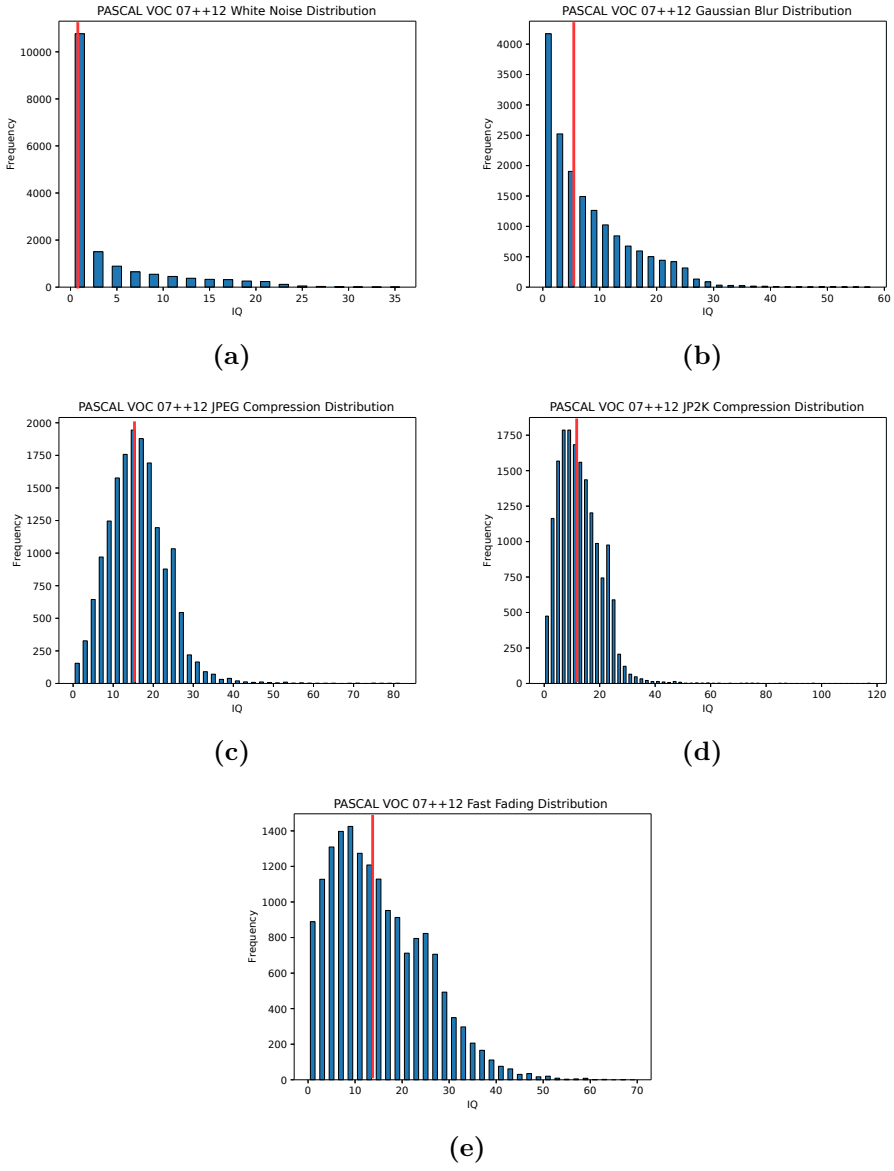


Figure 5.5: Histograms representing the distribution of image quality for the five distortions trained from the LIVE image quality dataset. The distortions shown are white noise (a), Gaussian blur (b), JPEG compression (c), JP2k compression (d), fast fading (e).

The distribution for white noise and Gaussian blur is skewed towards a higher image quality as seen in Figure 5.5a and Figure 5.5b and also to a lesser extent in fast fading in Figure 5.5e. Whereas the image quality for compression distortions is somewhat of a Gaussian nature in Figure 5.5c and Figure 5.5d. For determining an appropriate manner to split the data the same constraints are made as in that for object sizes, namely that both subsets of data should have an equal number of ground truths to train on. Again by taking

the median for each of the five distributions can satisfy this. The respective medians can be seen in Table 5.7.

Table 5.7: My caption

Distortion Type	**Median**
White Noise	0.599
Gaussian Blur	5.607
JPEG Compression	15.660
JP2K Compression	11.747
Fast Fading	13.373

A skewed distribution of data was also present for object sizes, however, creating two subsets of data for high and low white noise image quality does not appear to be feasible. The combination of both the heavy skew and half of the data lying below 0.599 indicates that a minimal amount of white noise distortion is present in the 07++12 dataset. Therefore, this distortion is not considered for part of the ensemble. While the Gaussian blur image quality is also skewed it is similar to that of the the object sizes and therefore is deemed appropriate to split based upon its median of 5.607. The remaining distributions are much less skewed and therefore a total of eight R-FCN models will be trained for the high and low levels of image quality for the distortions Gaussian blur, JPEG compression, JP2K compression and fast fading. Therefore, in total there will be ten R-FCN models trained including the two for smaller and larger object sizes.

The 07++12 dataset has a total of 16,551 images and as this data is to be distributed into eight different subsets there is a possibility that there is a high level of overlap between the sets. As the aim of the ensemble is to learn to detect objects based upon different information, if subsets are two similar there may be potentially no advantage gained between two or more models. Therefore, a comparison matrix is used to evaluate how much the different combinations of 07++12 match. This can be seen for higher quality subsets in Table 5.8, lower quality in Table 5.9 and between lower and higher quality in Table 5.10.

Table 5.8: Higher Quality

	Gaussian Blur	**JPEG**	**JP2K**	**FF**
Gaussian Blur		74.14%	70.78%	83.12%
JPEG	74.14%		80.62%	77.98%
JP2K	70.78%	80.62%		72.75%
FF	83.12%	77.98%	72.75%	

For the subsets of data for both higher and lower quality there are a few instances of relatively high overlap in images. The largest is between Fast Fading (FF)

gls for FF

and Gaussian blur with 83.12% and 83.11%. Other instances of overlap also appears between JPEG and JP2K compression which could be intuitively explained due to their similarities in their distortions. However, in general it is deemed that enough difference is present between the splits to train variants of R-FCN networks.

Table 5.9: Lower Quality

	Gaussian Blur	**JPEG**	**JP2K**	**FF**
Gaussian Blur		74.14%	70.78%	83.11%
JPEG	74.14%		80.62%	77.98%
JP2K	70.78%	80.62%		72.75%
FF	83.11%	77.98%	72.75%	

Table 5.10: Lower / Upper

	Gaussian Blur$_{Lower}$	**JPEG**$_{Lower}$	**JP2K**$_{Lower}$	**FF**$_{Lower}$
Gaussian Blur$_{Higher}$	0%	25.86%	29.22%	16.88%
JPEG$_{Higher}$	25.86%	0%	19.38%	22.02%
JP2K$_{Higher}$	29.22%	19.38%	0%	27.25%
FF$_{Higher}$	16.88%	22.02%	27.25%	0%

Finally, Table 5.10 shows the comparison matrix between all eight data subsets. There is much less overlap between these sets as much of the overlaps are present in respective higher and lower configurations as seen in Table 5.8 and Table 5.9.

5.2.4 Evaluating Image Quality Experts

As in the resolution-aware R-FCN networks individual tests are run to evaluate whether or not the models trained on the above data are candidate experts. Firstly, using the same measures as in Section *5.1 Resolution-Aware Object Detection*, the 07 test set is split into lower and upper subsets for each distortion type according to their respective medians. The two respective experts trained on each of their subset and the baseline R-FCN model are evaluated on each split of the test set to see if experts have been trained.

Firstly, the results for the Gaussian blur experts can be seen in Table **??**. Unfortunately, the models trained on each of the subsets of data do not appear to give any advantage over training on all of the 07++12 data. Regardless of the test data the best performing model is that trained on all of the data, outperforming by 3-5%. Similar results can be seen in Table 5.12, Table 5.13 and Table 5.14 for each the remaining models trained on subsets of JPEG, JP2K, and fast fading distortions. This seems to indicate that the distortions are not as apparent for the R-FCN models such that it is possible to train expert members to either subset.

Regardless, of this result the following section will present a method to ensemble these members and the models trained for smaller and larger object sizes. Future methods may still be able to benefit from an ensemble where a clearer factor that the four distortions covered here.

5.3 Ensemble

A number of different strategies for combining the ensemble members will be described in this section. This includes averaging and weighted averaging the detections. The method

Table 5.11: Gaussian Blur Experts

Train Data	**Test Data**	**AP (%)**
07++12 Gaussian Blur_{lower}	07 Gaussian Blur_{lower}	75.76%
07++12 Gaussian Blur_{upper}	07 Gaussian Blur_{lower}	74.75%
07++12	07 Gaussian Blur_{lower}	**79.08%**
07++12 Gaussian Blur_{lower}	07 Gaussian Blur_{upper}	76.33%
07++12 Gaussian Blur_{upper}	07 Gaussian Blur_{upper}	76.50%
07++12	07 Gaussian Blur_{upper}	**80.22%**
07++12 Gaussian Blur_{lower}	07	76.25%
07++12 Gaussian Blur_{upper}	07	75.39%
07++12	07	**79.59%**

Table 5.12: JPEG Experts

Train Data	**Test Data**	**AP (%)**
07++12 JPEG_{lower}	07 JPEG_{lower}	74.33%
07++12 JPEG_{upper}	07 JPEG_{lower}	73.46%
07++12	07 JPEG_{lower}	**78.51%**
07++12 JPEG_{lower}	07 JPEG_{upper}	76.69%
07++12 JPEG_{upper}	07 JPEG_{upper}	76.27%
07++12	07 JPEG_{upper}	**80.05%**
07++12 JPEG_{lower}	07	76.01%
07++12 JPEG_{upper}	07	75.26%
07++12	07	**79.59%**

for inferring each of the ensemble will be the same apart from the combination set. This is shown in Figure 5.6. Firstly, for a object proposal in an image found with the RPN each network will infer a bounding box and associated confidence for all classes. After this the given ensemble combination method determines the final detection.

Table 5.13: JP2K Experts

Train Data	Test Data	AP (%)
07++12 JP2K$_{lower}$	07 JP2K$_{lower}$	74.75%
07++12 JP2K$_{upper}$	07 JP2K$_{lower}$	74.65%
07++12	07 JP2K$_{lower}$	**79.18%**
07++12 JP2K$_{lower}$	07 JP2K$_{upper}$	75.86%
07++12 JP2K$_{upper}$	07 JP2K$_{upper}$	76.66%
07++12	07 JP2K$_{upper}$	**80.01%**
07++12 JP2K$_{lower}$	07	75.69%
07++12 JP2K$_{upper}$	07	75.64%
07++12	07	**79.59%**

Table 5.14: FF Experts

Train Data	Test Data	AP (%)
07++12 FF$_{lower}$	07 FF$_{lower}$	75.94%
07++12 FF$_{upper}$	07 FF$_{lower}$	74.56%
07++12	07 FF$_{lower}$	**79.02%**
07++12 FF$_{lower}$	07 FF$_{upper}$	77.18%
07++12 FF$_{upper}$	07 FF$_{upper}$	76.26%
07++12	07 FF$_{upper}$	**80.79%**
07++12 FF$_{lower}$	07	76.40%
07++12 FF$_{upper}$	07	74.93%
07++12	07	**79.59%**

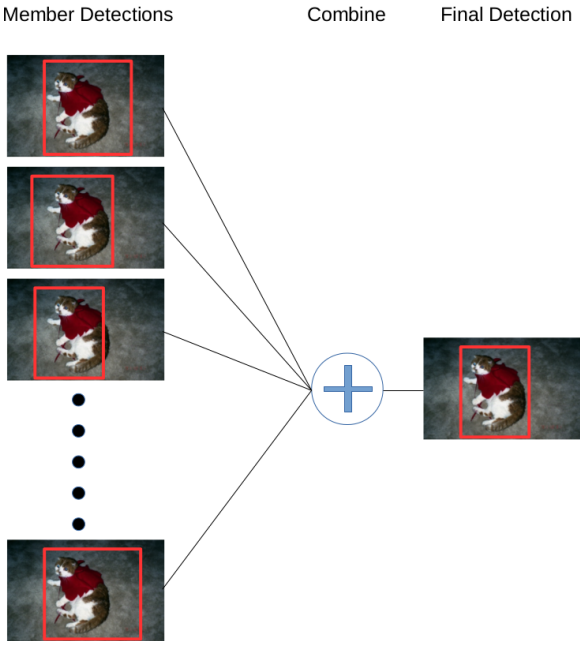


Figure 5.6

A number of different combination strategies will be presented and evaluated in the remainder of this section.

5.3.1 Average Ensemble

One of the combination strategies is similar to that used when evaluating the resolution-aware object detector in Section *5.1 Resolution-Aware Object Detection*. Each of the five ensemble factors are weighted evenly in the overall ensemble. Within each ensemble factor pair, the detection for one of the pairs will be chosen and the other discarded. This is determined by where the given factor lies in relation to the training data distribution. For example, if for the given test image it is measured with Deep IQA to have JPEG compression below the threshold used to split the data, then the detection found using the model trained on that data will be used. This results in five detections that will be weighted equally to find the final detection by:

$$E_j = \frac{1}{n} \sum_{i=1}^{n} p_{i,j} \tag{5.1}$$

where n is the number of detections found by the n ensemble factor, p is the detection result to be averaged and i represents one of the ensemble factors. Finally, j is one of the five values found by each detection, namely the four corners of the bounding-box and the associated confidence.

5.3.2 Weighted Average Ensemble

Each of then 10 trained networks will be used on all object proposals found using the RPN. Weights will be distributed evenly across each of the five different types of factors. The weighted average ensemble is determined for each bounding-box and the associated confidence by:

$$E_j = \frac{1}{n} \sum_{i=1}^{n} w_i p_{i,j} \tag{5.2}$$

Weights are determined in pairs for each of the 5 ensemble factors, where the total sum of weights is equal to n. If each detection were to be weighted equally all w would be equal to 1. As the weights are calculated in pairs each ensemble factor is overall weighted equally as the pair of weights can at most be equal to 2. By using this tactic in between the two sets of network for a given factor can be weighted differently but overall each factor is weighted equally. Weights for a given factor is found according to where the the test image lies for that factors training distribution data. For example, the subsets of training data for Gaussian blur was determined according to the line shown in Figure 5.7.

line showing split in training data

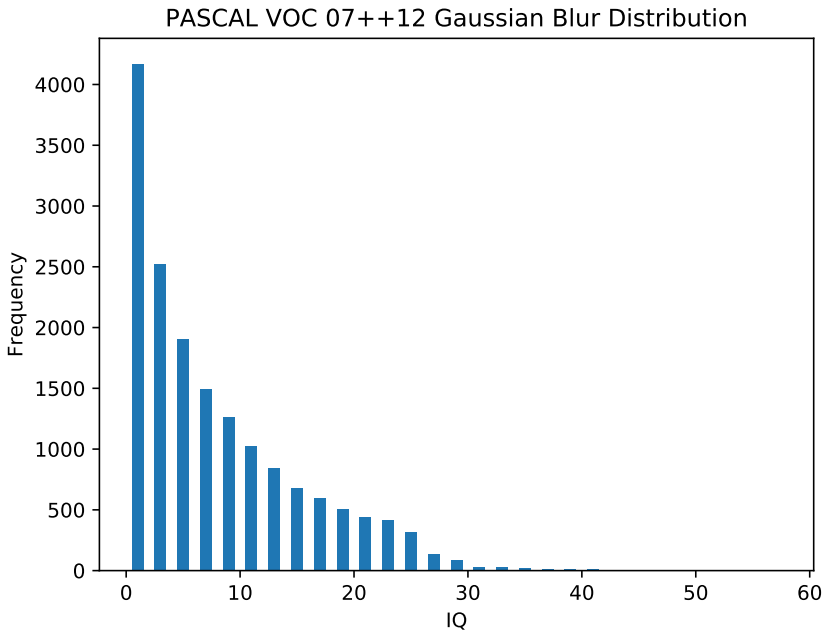


Figure 5.7

The quality, q_i with respect to blur for a given image is determined using the appropriate Deep IQA model, if the quality is below the value used to split the data the weights are calculated for the detection found with the given lower network by:

$$w_{Lower} = 2 - \frac{split - q_i}{split - minq_i} \tag{5.3}$$

and the weight for the upper network w_{Upper} by:

$$w_{Upper} = 2 - w_{Lower} \tag{5.4}$$

where *split* is the value used to split the training data and $minq_i$ is the minimum quality for the given factor in the training set.

However, if the quality is above *split* the w_{Upper} is calculated by:

$$w_{Upper} = 2 - \frac{maxq_i - q_i}{maxq_i - split} \tag{5.5}$$

and lower weight w_{Lower}:

$$w_{Lower} = 2 - w_{Upper}. \tag{5.6}$$

5.3.3 Ensemble Results

In this section the results for the two above mentioned ensemble combinations strategies will be presented. Each presentation will be accompanied with the result for the baseline R-FCN ResNet-101 model trained on all of the 07++12 training data and will be dubbed

as baseline. The results presented will be on the 2007 PASCAL VOC test set as also shown in earlier preliminary results in this report.

The results for both combination strategies can be seen in Table 5.15.

Table 5.15: Results for the two ensemble combination strategies and for the baseline model on the 2007 test set.

Method	**AP (%)**
Average	79.21%
Weighted Average	79.13%
Baseline	**79.59%**

While neither of the combinations provide an improvement over the baseline method both provide an increase in performance in comparison to the image quality expert results found shown in Section *5.2.4 Evaluating Image Quality Experts* where individual members were 3-4% worse in performance in comparison to the baseline model on their trained expert areas. Additionally, the weighted average only performs slightly worse than that of the non-weighted version. This is interesting as the intra-factor experts for the image quality factors are similar in performance, however, while disregarding this and weighting models still provides a performance increase.

Next, to the evaluate the contribution of both the eight quality factor ensemble members and the two resolution members these were combined separately based on the two strategies. The results for the average ensemble can be seen in Table 5.16 and the weighted ensemble in Table 5.17.

Table 5.16: Results for the the image quality ensemble members and resolution members individually combined using average strategy on the 2007 test set.

Ensemble Members	**AP (%)**
Image Quality	78.15%
Resolution	78.13%

Table 5.17: Results for the the image quality ensemble members and resolution members individually combined using the weighted average strategy on the 2007 test set.

Ensemble Members	**AP (%)**
Image Quality	78.44%
Resolution	75.00%

By separating the quality and resolution members Table 5.16 shows that the performance decreases by over 1% for both in comparison the the average ensemble result of 79.21%. This appears to indicate that the two complement each other well and have their own expertises for this problem. Table 5.17 also shows a decrease in performance when separating the members based on their expertise factors. The weighted average combination strategy does not show as large of a decrease in performance for only image quality as the average combination does, however, there is still a performance drop from 79.13% to 78.44%. There

is a significant decrease in performance for the two resolution members showing an AP of 75.00% on the test set. This seems to show that the addition of weighing individual detections based on proposal size as a poorer approach. Comparing the two tables seems to indicate that image quality members are well suited to adding a weight to detections but the resolution members are better suited to simply taking the detection from the appropriate model. Therefore, combinations of average and weighted average ensembles could be of interest. The results for these can be seen in Table 5.18. The two strategies are shown as either Image Quality or Resolution followed by the subscript $_{Avg}$ or $_{WAvg}$ indicating the combination strategies of average or weighted average respectively.

Table 5.18: Results for the the image quality ensemble members and resolution members with both combinations of average and weighted average on the 2007 test set.

Ensemble Members	**AP (%)**
Image Quality$_{WAvg}$ / Resolution$_{Avg}$	**79.90%**
Image Quality$_{Avg}$ / Resolution$_{WAvg}$	78.71%
Baseline	79.59%

Results in Table 5.18 show that by using separate strategies with image quality members are weighted and when one of the resolution members are weighted 100% the performance on the test set surpasses the baseline model. The increase is slight from 79.59% to 79.90%. However, again it appears that the members of the ensemble compliment each other well both intra-factor and inter-factor. As suspected the opposite strategy of average combination for image quality and weighted average for resolution does not surpass previous results.

The results so far have only been with different combinations of the expert ensemble members. However, another strategy is to include the baseline model trained on all of the 07++12 data. As the baseline model performs well by itself the other ensemble members will act as support, as ideally there are parts of the PASCAL VOC data that they perform better on due to the reduced training variance. The methods for ensemble are used as earlier, except that there is an additional member in the ensemble. Also it should be noted that as there is no complementary member to the baseline, its detections are weighted by 1.0 regardless of ensemble combination strategy. Firstly, the results for the average and weighted average ensemble, both with the baseline model can be seen in Table 5.19. The inclusion of the baseline model is shown by the subscript $_{base}$. The table shows that in both strategies the inclusion increases the overall performance. Using the weighted average the performance is increased by 0.22%. While the average strategy is increased above the baseline result by 0.65% to 79.86%.

Next the addition of the baseline model with respective to each ensemble factor using the average ensemble strategy can be seen in Table 5.20. Both factors have a significant increase in AP performance with the extra ensemble member. The image quality experts gain 0.77%, while the two resolution members have their performance increased by 1.96%. The result of 80.09 is higher than the result shown in Table 5.18 even without having members trained towards image quality factors.

Adding the baseline model to the factors and using the weighted average strategy does not result in an improvement over the baseline result as shown in Table 5.21. However, both factors see a larger increase in performance that that of the average combination

Table 5.19: Results for the two ensemble combination strategies and for the baseline model on the 2007 test set. Shown is both the results with the expert ensemble members only and experts plus the baseline model.

Method	**AP (%)**
Average	79.21%
Average$_{base}$	**79.86%**
Weighted Average	79.13%
Weighted Average$_{base}$	79.35%
Baseline	79.59%

Table 5.20: Results for the the image quality ensemble members and resolution members individually combined using average strategy on the 2007 test set. Shown is both the results with the expert ensemble members only and experts plus the baseline model.

Ensemble Members	**AP (%)**
Image Quality	78.15%
Image Quality$_{base}$	78.92%
Resolution	78.13%
Resolution$_{base}$	**80.09%**
Baseline	79.59%

in Table 5.20. Image quality performance is increased by 0.71% and resolution members increase by 3.21%. Clearly regardless of ensemble strategy the addition of the baseline model aids in overall object detection on PASCAL VOC.

Table 5.21: Results for the the image quality ensemble members and resolution members individually combined using weighted average strategy on the 2007 test set. Shown is both the results with the expert ensemble members only and experts plus the baseline model.

Ensemble Members	**AP (%)**
Image Quality	78.44%
Image Quality$_{base}$	79.15%
Resolution	75.00%
Resolution$_{base}$	78.21%
Baseline	**79.59%**

While improvements are seen for both strategies with the addition of baseline, the tendency is still that the resolution members perform best with the average ensemble and image quality with weighted average. Therefore, the two combinations of ensembles with the addition were tested. This is shown in Table 5.22 and shown is that this provided the best result of any ensemble combination. Image quality with the weighted average and resolution with average ensemble results in 80.15%, an increase of 0.56% in comparison to the baseline R-FCN.

Table 5.22: Results for the the image quality ensemble members and resolution members with both combinations of average and weighted average on the 2007 test set. Shown is both the results with the expert ensemble members only and experts plus the baseline model.

Ensemble Members	**AP (%)**
Image Quality$_{WAvg}$ / Resolution$_{Avg}$	79.90%
Image Quality$_{WAvg}$ / Resolution$_{Avg\ base}$	**80.15%**
Image Quality$_{Avg}$ / Resolution$_{WAvg}$	78.71%
Image Quality$_{Avg}$ / Resolution$_{WAvg\ base}$	79.10%
Baseline	79.59%

6 Discussion

7 Conclusion

Bibliography

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft COCO: Common Objects in Context*. Cham: Springer International Publishing, 2014, pp. 740–755. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10602-1_48

[4] X. Zhang, Y.-H. Yang, Z. Han, H. Wang, and C. Gao, "Object class detection: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 10:1–10:53, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2522968.2522978

[5] F. Schroff, *Semantic Image Segmentation and Web-supervised Visual Learning*. University of Oxford, 2009. [Online]. Available: https://books.google.dk/books?id=4EqZYgEACAAJ

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0275-4

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11263-014-0733-5

[8] flickr. flickr. [Online]. Available: https://www.flickr.com/

[9] R. Girshick, J. Donahue, T. Darrell, and J. 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.

[10] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11263-013-0620-5

[11] R. Girshick, "Fast R-CNN," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran

Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Neural Information Processing Systems (NIPS)*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[16] M. COCO. (2017) Ms coco detections leaderboard. [Online]. Available: http://http://mscoco.org/dataset/#detections-leaderboard

[17] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[18] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[19] S. Zagoruyko, A. Lerer, T. Lin, P. H. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár, "A multipath network for object detection," *CoRR*, vol. abs/1604.02135, 2016. [Online]. Available: http://arxiv.org/abs/1604.02135

[20] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," *CoRR*, vol. abs/1512.04143, 2015. [Online]. Available: http://arxiv.org/abs/1512.04143

[21] A. Shrivastava and A. Gupta, *Contextual Priming and Feedback for Faster R-CNN*. Cham: Springer International Publishing, 2016, pp. 330–348. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_20

[22] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta, "Beyond skip connections: Top-down modulation for object detection," *CoRR*, vol. abs/1612.06851, 2016. [Online]. Available: http://arxiv.org/abs/1612.06851

[23] A. Shrivastava, A. Gupta, and R. B. Girshick, "Training region-based object detectors with online hard example mining," *CoRR*, vol. abs/1604.03540, 2016. [Online]. Available: http://arxiv.org/abs/1604.03540

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[25] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013. [Online]. Available: http://arxiv.org/abs/1312.6229

[26] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 2155–2162. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.276

[27] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *CoRR*, vol. abs/1412.1441, 2014. [Online]. Available: http://arxiv.org/abs/1412.1441

[28] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[29] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[30] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: http://arxiv.org/abs/1605.06409

[31] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[32] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," *CoRR*, vol. abs/1611.07709, 2016. [Online]. Available: http://arxiv.org/abs/1611.07709

[33] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems (NIPS 1989)*, D. Touretzky, Ed., vol. 2. Denver, CO: Morgan Kaufman, 1990.

[34] C. Zhang and Y. Ma, *Ensemble Machine Learning*. Springer US, 2012.

[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[36] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3440–3451, Nov 2006.

[37] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. Live image quality assessment database release 2. [Online]. Available: http://live.ece.utexas.edu/research/quality

[38] N. Ponomarenko, O. Ieremeiev, V. Lukin, K. Egiazarian, L. Jin, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C. C. J. Kuo, "Color image database tid2013: Peculiarities and preliminary results," in *European Workshop on Visual Information Processing (EUVIP)*, June 2013, pp. 106–111.

[39] E. Larson and D. M. Chandler, "Consumer subjective image quality database," 2009.

[40] S. Bosse, D. Maniry, K. Müller, T. Wiegand, and W. Samek, "Deep neural networks for no-reference and full-reference image quality assessment," *CoRR*, vol. abs/1612.01697, 2016. [Online]. Available: http://arxiv.org/abs/1612.01697

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

Appendices

A Appendix A

A.1 Resolution-Aware Object Detection

Table A.1: Results Test07$_{small}$

Train Data	Test Data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
07++12$_{small}$	07Test$_{small}$	**34.10**	**28.32**	**31.47**	**50.08**	**52.30**	**46.47**	**17.04**	**50.08**	**13.36**	**45.58**	**63.96**	0.38	**16.54**	**15.72**	**34.19**	**55.16**	**31.02**	**65.25**	**2.52**	**9.07**	**53.46**
07++12$_{larger}$	07Test$_{small}$	3.08	1.54	4.91	4.95	2.96	1.42	2.90	3.44	1.04	5.58	2.79	0.34	2.68	2.50	3.42	3.02	2.20	5.74	0.94	4.22	4.98
07++12	07Test$_{small}$	22.16	21.49	20.70	28.18	30.57	35.72	12.04	35.48	5.83	39.40	41.18	**0.52**	10.56	10.08	15.07	28.92	20.93	44.11	1.58	8.76	32.13

Table A.2: Results Test07$_{larger}$

Train Data	Test Data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
07++12$_{small}$	07Test$_{larger}$	36.40	53.91	46.30	29.28	12.45	13.91	62.50	35.16	71.09	4.57	22.06	43.96	61.90	56.13	44.99	17.84	5.09	17.14	52.30	64.13	13.35
07++12$_{larger}$	07Test$_{larger}$	**76.60**	**87.49**	**78.95**	**76.57**	**66.74**	**67.98**	**86.49**	**86.40**	**88.44**	**50.11**	**82.56**	74.15	**86.59**	**86.16**	**82.57**	**84.00**	**46.13**	**67.78**	**78.31**	**84.39**	**71.11**
07++12	07Test$_{large}$	64.34	78.18	71.17	58.76	43.91	36.72	82.71	73.40	85.64	25.73	54.67	**74.47**	82.66	82.25	77.02	63.90	31.53	46.35	77.67	84.07	54.92

Table A.3: Results 07

Train Data	Test Data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
07++12$_{small}$	07	59.65	60.70	68.59	58.54	46.91	22.21	76.58	56.20	84.76	39.22	51.84	66.49	79.93	79.00	68.58	54.64	27.72	47.86	75.42	78.99	48.95
07++12$_{larger}$	07	62.65	75.46	69.93	69.03	50.15	51.35	74.68	80.54	77.36	40.38	75.05	40.65	73.59	69.38	67.94	59.37	31.35	68.72	54.13	69.66	54.23
07++12	07	**76.35**	**79.14**	**80.16**	**76.97**	**68.68**	**57.11**	**86.36**	**85.95**	**88.34**	**60.06**	**86.85**	**67.12**	**87.91**	**86.51**	**80-36**	**78.14**	**45.67**	**77.74**	**76.98**	**83.75**	**73.30**

A.2 Deep IQA Models

Table A.4: Gaussian Blur

Model	Best Epoch	LCC	SROCC	Mean Difference
1	5	0.9879	0.9825	2.218
2	1	0.9552	0.9547	2.395
3	1	0.9872	0.9750	2.499
4	6	0.9767	0.9772	2.266
5	23	0.9864	0.9913	2.103
6	6	0.9633	0.9216	2.875
7	10	0.9914	0.9807	2.785
8	7	0.9686	0.9756	2.188
9	6	0.9807	0.9807	4.013
10	2	0.9528	0.9415	3.389

Table A.5: White Noise

Model	Best Epoch	LCC	SROCC	Mean Difference
1	50	0.9958	0.9900	1.387
2	44	0.9935	0.9817	0.862
3	93	0.9940	0.9875	1.2946
4	8	0.9952	0.9862	1.312
5	96	0.9953	0.9886	1.3604
6	5	0.9953	0.9896	1.5867
7	5	0.9976	0.9952	0.9822
8	3	0.9953	0.9856	1.728
9	129	0.9981	0.9897	1.412
10	286	0.9968	0.9945	1.196

Table A.6: JPEG

Model	Best Epoch	LCC	SROCC	Mean Difference
1	4	0.9861	0.9584	3.025
2	44	0.9766	0.9523	2.094
3	14	0.9766	0.9374	3.067
4	46	0.9838	0.9565	2.716
5	8	0.9783	0.9304	2.415
6	31	0.9761	0.9560	1.986
7	14	0.9927	0.9553	3.023
8	10	0.9771	0.9728	2.981
9	6	0.9834	0.9500	3.165
10	16	0.9732	0.9539	2.580

Table A.7: JP2K

Model	Best Epoch	LCC	SROCC	Mean Difference
1	36	0.9823	0.9624	3.819
2	32	0.9879	0.9703	2.822
3	1	0.9658	0.9525	3.243
4	26	0.9861	0.9775	4.011
5	17	0.9790	0.9792	3.552
6	48	0.9885	0.9804	2.720
7	25	0.9836	0.9675	2.916
8	51	0.9871	0.9654	3.345
9	25	0.9714	0.9447	2.953
10	4	0.9567	0.9597	2.265

Table A.8: Fast Fading

Model	Best Epoch	LCC	SROCC	Mean Difference
1	2	0.9611	0.9379	2.585
2	17	0.9528	0.9343	2.740
3	10	0.9692	0.9521	2.341
4	2	0.9627	0.9585	3.004
5	3	0.9764	0.9487	3.620
6	39	0.9848	0.9748	3.121
7	8	0.9504	0.9376	3.445
8	1	0.9772	0.9668	4.188
9	59	0.9747	0.9477	2.578
10	2	0.9697	0.9467	3.105

Notes