Inflated Multinomial Matching for Anchor-Free Object Detection

Cesar Hiersemann

Master's thesis
2018:E66

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Abstract

This thesis presents a novel matching strategy, Inflated Multinomial Matching, which enables training of anchor-free object detection models based on convolutional neural networks. An important aspect of detection models is the integral usage of anchor boxes, where an anchor box is a bounding box with a preset and constant location, size and shape in the image. The matching strategy presented utilizes the similarity scores between ground truths and predictions in a stochastic way, which lets detection models obtain several independent submodels where each submodel specializes towards predicting objects of a certain size and shape, essentially mimicking the main benefits of anchors boxes in an unsupervised way. The intended behavior of the matching strategy is confirmed through a number of indicators monitored throughout the training process. Finally, a full scale object detection model is trained with Inflated Multinomial Matching and example detection results are showcased.

Acknowledgements

I would like to thank the incredible team at Sentia for entrusting me with the development of the detection system, where the level of autonomy and responsibility I was given is something I could not have imagined beforehand.

A special thanks goes to my supervisor at Sentia, Mads Ingwar, who was and continues to be a great source of inspiration both professionally and personally. My sincerest thanks also goes to Jacob Würtz Lyngbye and Andreas Møller, for the intense discussions, interesting ideas, valuable input and unhindered encouragement throughout the whole project as well as for the rest of the time at Sentia. And of course to the rest of the team, Fredrik Ferbing, Søren Jensen, Mikkel Andersen and Ida Hjorth for the great times and for always staying positive throughout both the good times and bad times.

I would also like to express my gratitude to my principal supervisor at Lund University, Prof. Anders Heyden, for his immense patience and continuous support. Although the report was severly delayed, Anders positive attitude was a great help in finishing it.

A big thanks to my friends and family for always being supportive and helping me every step of the way.

Finally, thank you Lisa for enduring through the difficult times and for being the greatest source of inspiration by just being you. And Neo, for being the source of motivation through the last stretch.

Contents

List of Figures

List of Tables

Abbreviations

CNN	**C**onvolutional **N**eural **N**etwork
DL	**D**eep **L**earning
ILSVRC	**I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge
IMM	**I**nflated **M**ultinomial **M**atching
IoU	**I**ntersection **o**ver **U**nion
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etworks
RPN	**R**egion **P**roposal **N**etwork
SSD	**S**ingle **S**hot Multibox **D**etector
YOLO	**Y**ou **O**nly **L**ook **O**nce

Chapter 1

Introduction

Object detection refers to the combined task of automatic *localization* of objects of interest in a digital image or video, as well as *classification* of each object. Using visual information for detection and recognition of objects is a deceptively difficult task for a computer, and the fact that it is done subconsciously by humans and other animals at a fraction of a second speaks more of the remarkable computational and relational power of the biological brain than the ease of the task itself [1].

This thesis presents a novel algorithm called *Inflated Multinomial Matching* (IMM), which is used as a step in the training process for *Deep Learning* (DL) models for object detection. IMM introduces a stochastic method to match the ground truth boxes directly with the bounding box proposals generated by a DL detection model, instead of matching with a corresponding *anchor*. The purpose of the algorithm is to enable the training of anchor-free object detection models without sacrificing the models capability of detecting objects of diverse shapes and sizes with high precision, one of the main advantages that come from using anchor priors.

The report is divided into eight chapters. Chapter 2 contains an overview on object classification and detection systems and how DL has impacted the field. Chapter 3 presents the software and hardware used throughout the project, as well as datasets commonly used for training object detection models. Chapter 4 covers some of the important theoretical concepts and methods of implementation to help understand the algorithm and associated experiments. The IMM algorithm is presented in Chapter 5 together with a motivation and a demonstrative example. Experimental results on the algorithm is presented and discussed in Chapter 6 and concluding thoughts on the different aspects covered can be found in Chapter 7. Readers with experience in DL and methods of object detection can choose to skip directly to Chapter 5.

1.1 Context

This Master Thesis project was conducted in partnership with Sentia[1], a Copenhagen-based technology startup in the field of intelligent business analytics. By analysing and merging several different data sources, such as data from cell towers, WiFi traffic, Bluetooth beacons and video cameras, Sentia quantifies customer behavior for businesses as well as gives insight on traffic flow and modal share for sustainable city projects.

At the time of writing, object detection based on CNNs was in large part performed on high-end hardware, primarily on expensive and compute intensive graphics processing units (GPUs). A major focus at Sentia is complete compliance with privacy regulations in the EU, which means that no personally identifiable information, such as images or video, are sent from the data source or stored either locally or on cloud servers. This means that all image processing needs to happen in near realtime on the same device that captured the video. No open-source software or off-the-shelf detection models available at the time would neither run sufficiently fast nor accurately enough for the end-purpose of the product, which meant that a significant portion of the thesis project had to be dedicated to the construction of the software required for a scalable production system. While some details on the implementation and tools used throughout the project is included, the thesis is for the most part focused on the research aspects of the project.

[1] https://sentia.ai/

Chapter 2

Background

The task of object classification and detection has traditionally followed a two-step procedure, where the raw image content is first transformed into a feature representation intended to reduce the number of input variables while preserving the characteristics which distinguishes the objects of interest from each other or from the background class. Haar-like features [2], Local Binary Patterns [3] or Histogram of Oriented Gradients [4] were some of the more popular methods used for this transformation. This feature representation is then used to train a classification model, such as Nearest Neighbours, Decision Trees or Support Vector Machines. As long as the feature extraction algorithm was appropriately chosen and well calibrated, this approach was applied successfully for automation of numerous tasks such as optical character recognition for post- and bank offices, automatic licence plate recognition for toll stations and for law enforcement as well as facial recognition in early digital cameras to name a few.

In 2012, the *AlexNet* model [5] significantly improved on the state of the art on both the classification and localization tasks of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [6]. In contrast to the other competition entries that year, AlexNet utilized a Convolutional Neural Network (CNN) as its base computational engine, which combines the feature extraction and subsequent classification into an end-to-end trainable model. The CNN that AlexNet utilized contained 5 Convolutional Layers followed by 3 Fully Connected (FC) Layers, and the success of this Deep CNN (DCNN) brought about a reignition of DL as well as a paradigm shift in the field of Computer Vision as a whole [7]. Since then, almost all entries to ILSVRC use CNNs as their basic framework and these networks, among with various other DL methods, have been proven to be powerful tools across almost all subfields of image analysis and computer vision.

The highest accuracy object detection models to date are based on a two-stage approach, where first a set of *region proposals* are computed, indicating the regions in the image

which have a high probability of containing an object of interest. Then, the image information or extracted features corresponding to these regions are fed to a second stage network which fine-tunes the localization regression to better match the ground truth box, as well as computes the classification of the objects. This detection framework is commonly known as *Region-based Convolutional Neural Networks* (R-CNN) [8], and has together with its faster and more accurate successor frameworks Fast/Faster R-CNN [9] [10] been the base structure for the winning models of detection challenges since 2014. R-CNN and Fast R-CNN rely on the *Selective Search* [11] algorithm to generate region proposals, which was a huge computational improvement compared to *Exhaustive Search* that was commonly used before. Faster RCNN then replaced the selective search procedure with a *Region Proposal Network* (RPN), which combined with the detection network produced a fully end-to-end trainable method with large improvements in both performance and accuracy. The impact the development of these frameworks have had on the general applicability of object detection cannot be understated, with the execution time on a single image brought down from close to a minute to several times per second as well as producing results of a high enough quality to be usable for cross-domain research and in industry applications.

Comparing the two-stage approach of the methods mentioned so far with the architecture of a standard deep classification model, it is apparent that there are multiple layers of added complexity and possible computational bottlenecks in the two-stage detection models. The final performance in both speed and accuracy of the models depends not only on the depth and architecture of the base network, but also on how well each independent module is optimized and implemented. To combat these issues, the so called single-stage models treat object detection as a regression problem over the entire image in one single forward propagation. Two of the most popular single-stage detectors are *You Only Look Once* (YOLO) [12] [13] and *Single Shot MultiBox Detector* (SSD) [14]. Shared attributes between both YOLO and SSD are that instead of using region proposals, they use the resulting feature maps of the base network directly as feature descriptors for the different image regions. These feature maps are then appended by a set of convolutional layers responsible for bounding box regression and corresponding classification. The main advantages of the single-stage methods are the increase in training and inference speed, in which the main bottleneck is almost entirely the single forward propagation through the base network. The tradeoffs in speed and accuracy based on method and model selection is compherensively investigated in [15].

An important aspect of both the single-stage and the two-stage detection models is the integral usage of *anchor boxes* (also known in literature as prior boxes or default boxes). An anchor box is a bounding box with a preset and constant location, scale and aspect ratio in the image. The purpose of anchor boxes is to simplify the localization

task by training the model to predict the offset from the object to a nearby anchor box instead of predicting the absolute values directly. The first usage of anchors was in the RPN stage of the Faster R-CNN [10] model, where a total of 9 anchors per grid cell of the last feature map were used. These were generated by choosing a box with area and aspect ratio from $\{128^2, 256^2, 512^2\}$ and {1:1,2:1,1:2} respectively, centered at the position of the grid cell. Both YOLO and SSD utilize anchors in a similar way, although improvements have been seen by generating dataset specific anchors by proposing prior scale and aspect ratios through k-means clustering of the ground truth box dimensions [13].

Chapter 3

Datasets and Tools

There is a wide selection of DL frameworks to choose from depending on the programming language used, with considerable differences in their implementation, flexibility and ease of use. Training and inference of models are also made more efficient by using hardware optimized for the selected task in a correctly setup environment. This chapter focuses on the software and hardware setup used as well as the datasets used for the experiments.

3.1 Software

Python[2] with the *NumPy*[3] package is a powerful environment for scientific computing while still being a general purpose programming system. A number of powerful DL libraries exist for Python with *TensorFlow*[4] [16], developed by the Google Brain team and released as open-source in 2015, being one of the most popular. All computational operations in TensorFlow are represented as nodes in a data graph, with directed edges representing the flow of data through the graph. Computational graphs are present across all DL frameworks in large part because of the heavy use of automatic differentiation, a set of techniques for numerically evaluating the gradient of a function defined by a computer program.

With TensorFlow as the computational backbone, machine learning algorithms can be developed, trained and run with considerable ease considering the complexity of the underlying computations. To further simplify the development task, *Keras*[5] [17] is a high-level library for building, evaluating and running neural networks which uses TensorFlow (among other frameworks) as its backend.

[2] http://www.python.org
[3] http://www.numpy.org
[4] http://www.tensorflow.org
[5] https://keras.io

OpenCV[1] is used as the computer vision library for reading image and video data, manipulating color-spaces, efficient resizing and other tools mainly used for image input/output/display and augmentation. If the application does not require any video processing, another excellent image processing library is *Pillow*[2] which arguably is more user friendly than OpenCV.

3.2 Hardware

Training a supervised DL model that can do anything useful typically relies on datasets ranging in size from $10^6 - 10^{15}$ bytes and evaluating and updating somewhere between $10^5 - 10^8$ model parameters repeatedly throughout the training process. The training of vision systems can be especially demanding, where each image is rich in information and commonly used datasets consist of $10^4 - 10^7$ annotated images.

All training and evaluation were carried out on a single **p2.xlarge** instance through *Amazon Elastic Compute Cloud*[3]. A NVIDIA K80 GPU is the main workhorse on this instance with 12 GB of GPU-memory. Having a sufficient amount of GPU-memory is especially important for training of vision systems since each batch of images needs to fit into GPU memory along with allocating memory for the gradient computation and updating of all the model parameters.

3.3 Datasets

The performance of DL models is highly dependent on the quality of the data used to train the model. Three commonly used datasets for object detection are *Pascal VOC*[4] [18], *Microsoft COCO*[5] [19] and *ImageNet*[6] [6] (Table 3.1). Table 3.1 shows number of images, number of object instances and number of classes included in the main detection challenges at the time this project was conducted. COCO and ImageNet are continuously updated with new images and annotations so these numbers are likely to change.

Name	$\mathbf{N}_{images}$	$\mathbf{N}_{objects}$	$\mathbf{N}_{classes}$
Pascal VOC (2007+2012)	27.1K	70.8K	20
Microsoft COCO (2014)	122K	886K	80
ImageNet Detection (2014)	476K	533K	200

TABLE 3.1: Three commonly used datasets for object detection.

[1] http://opencv.org/
[2] https://python-pillow.org/
[3] https://aws.amazon.com/ec2/
[4] http://host.robots.ox.ac.uk/pascal/VOC/
[5] http://cocodataset.org/
[6] http://www.image-net.org/

VOC is considerably easier (in reference to the comparatively high evaluation scores currently achieved by state of the art systems) than the two others, with only 20 classes and relatively few object instances per image. Although COCO contains fewer object classes than ImageNet, the high amount of object instances per image (avg. 7.7) and cluttered scenes make it a difficult dataset to reach high evaluation scores on. Real world applications often have to operate where the environment is complex and cluttered, and the high number of object instances per image provides efficiency since more ground truth samples are seen per image processed.

This thesis project mainly consists of experiments on the proposed algorithm, IMM, with a smaller section on a general purpose detection system trained using the proposed algorithm. Because of its smaller size, Pascal VOC is used for running the algorithm experiments since 20 different parameter settings were tested and each setting incurs a significant training time (around 72 hours in total for the algorithm experiments). For the general purpose detection system, Microsoft COCO is chosen as the main dataset for training, where during the duration of this project it was clearly seen that models trained on COCO performed considerably better on real world tasks compared to Pascal VOC.

A comprehensive table of the per-class object statistics for the COCO dataset is presented in Table A.1 in Appendix A.

Chapter 4

Theory & Methodology

This chapter presents some of the important concepts of object detection systems based on DL, which covers both general theory as well as includes a small section on implementation details. Most of the important theory underlying ML, DL and CNNs have been omitted, where [20] as well as the articles referenced in the Chapter 2 can serve as an excellent resource. Instead, the focus of this chapter is on specific topics important to understand IMM and the experiments conducted.

4.1 Object Detection

4.1.1 Localization

The objective of the localization (loc) task is to compute the coordinates in the image plane describing the position and shape of objects of interest. The localization of each object is commonly represented as a bounding box vector $\mathbf{b} \in \mathbb{R}^4$, which encodes the box by either containing the point coordinates for two opposite corners of the box (minmax encoding) or by containing the center coordinates as well as the width and height of the box (centered encoding), see Figure 4.1.

While these two ways to encode a bounding box are the most commonly seen in annotated datasets and for storing and presenting detection results, there are often additional transformations applied to the coordinates before the box coordinates are used as ground truth for training detection models.

$\mathbf{b}_{minmax} : (x_1, y_1, x_2, y_2)$

$\mathbf{b}_{centered} : (x_c, y_c, w, h)$

FIGURE 4.1:
Example bounding box in minmax encoding.

4.1.2 Classification

The objective of the classification (cls) task is to assign the objects of interest with an appropriate class label. *Binary classification* denotes the objective of only separating between two classes, such as $\{cat, dog\}$ or $\{object, background\}$. In *multi-label classification*, each object can belong to multiple classes simultaneously, in contrary to *multi-class classification* where each object belongs to only one class exclusively. The set of classes can be of varying degrees of granularity, both between and within datasets, where the granularity and choice of class labels can depend on many factors such as the accessibility of data and what the end purpose of the application or research is.

$cls_{species} : bird$

$cls_{subspecies} : chaffinch$

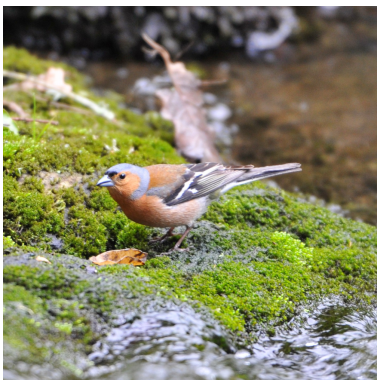


FIGURE 4.2:
Example class labelling.

The object in Figure 4.2 above could be correctly classified as both *bird* and by the name of its subspecies, *chaffinch*. In a classification problem with a set of C different classes, a ground truth object with label $y = c$ where $c \in \{(0), 1, \ldots, C\}$ (zero if including background class) is commonly represented by *one-hot* encoding of the label. One-hot encoding converts the class label (an integer) to a binary vector $\mathbf{y}$ where $y_i = 1$ if $i = c$ and 0 otherwise. A classification model would then output a prediction $\hat{\mathbf{y}}$ with $\hat{y}_i \in (0, 1)\ \forall\ i \in \{(0), 1, \ldots, C\}$, where $\hat{y}_i$ often represents the predicted probability for the object to belong to class i.

4.1.3 Detection

Combining the loc and cls tasks for all objects of interest in an image summarizes object detection. A ground truth object with its bounding box encoded by minmax encoding and with one-hot encoded class label in a 80 class dataset (e.g. COCO), including background class, would then be represented by a vector of 85 numbers:

$$\mathbf{y} = (x_1, y_1, x_2, y_2, 0, \ldots, 1, \ldots, 0) \tag{4.1}$$

Figure 4.3 demonstrates visualization of a model result by the general detection model trained with IMM in this project.

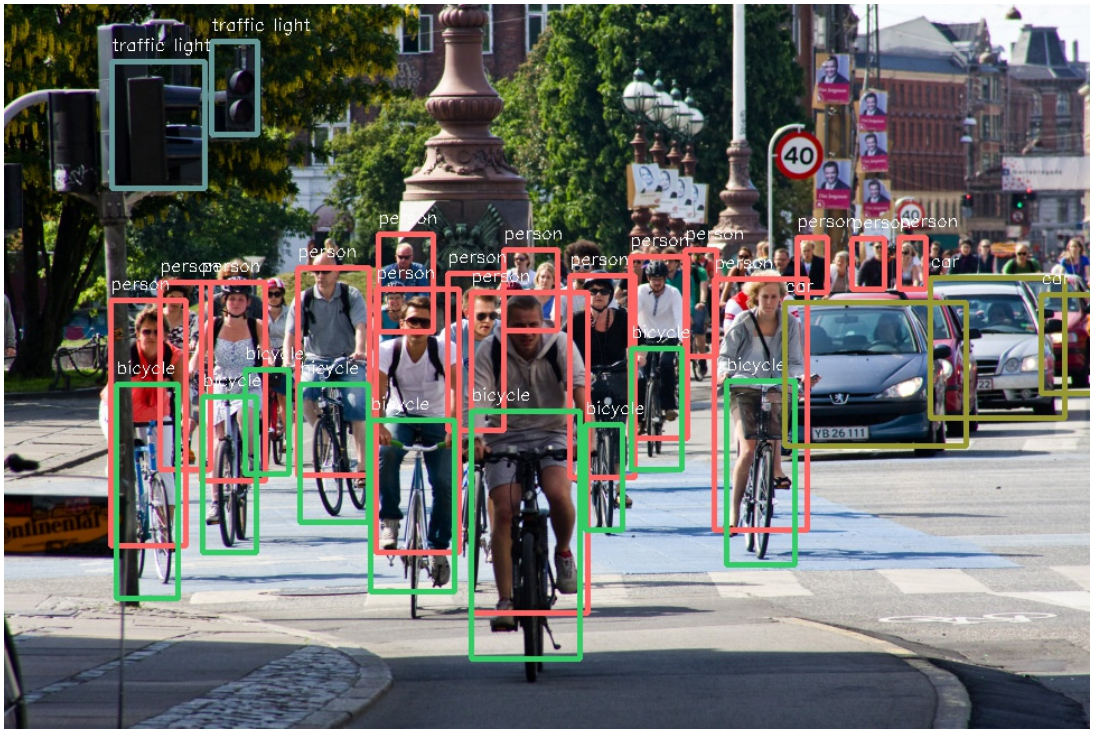


FIGURE 4.3:
Example detection result of our trained detection system.
Rush hour in the streets of Copenhagen. Credit: Mikael Colville-Andersen

4.1.4 Intersection over Union

The *Intersection over Union* (IoU) metric (also known as *Jaccard index*), is a similarity measure between finite sets which is commonly used in object detection for evaluating how good the correspondence is between two bounding boxes. Given bounding boxes $\mathbf{b}$ and $\hat{\mathbf{b}}$, the IoU score is given by:

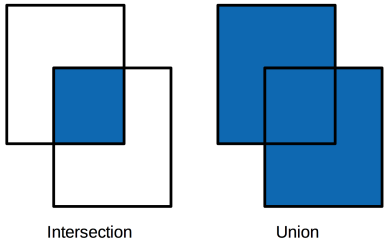


FIGURE 4.4:
Illustration of the regions of Intersection and Union between two bounding boxes.

$$IoU(\mathbf{b}, \hat{\mathbf{b}}) = \frac{|\mathbf{b} \cap \hat{\mathbf{b}}|}{|\mathbf{b} \cup \hat{\mathbf{b}}|} \tag{4.2}$$

where the size of the intersection $|\mathbf{b} \cap \hat{\mathbf{b}}|$ is the area of the overlap between the two boxes, and the size of the union $|\mathbf{b} \cup \hat{\mathbf{b}}|$ is the total shared area of the two boxes. See Figure 4.4.

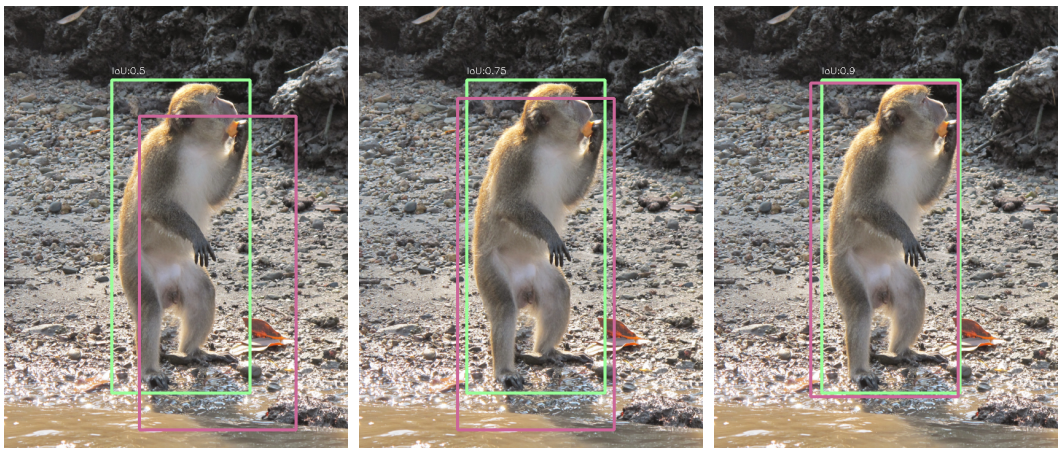


FIGURE 4.5:
Examples of IoU scores for predictions (pink) of a ground truth bounding box (green).
Left: 0.5 IoU, **Middle:** 0.75 IoU, **Right:** 0.9 IoU.
Macaque on Langkawi Island, Malaysia.

In Figure 4.5 an example is shown of three different bounding box predictions and their corresponding IoU scores. A threshold of 0.5 IoU is commonly seen as an "acceptable" detection, and meets the requirement for a true positive in challenges such as Pascal VOC, whereas a score of 0.8-0.9 is a very precise detection.

4.2 Deep Learning for Object Detection

4.2.1 Feature Extraction

Object detection models are generally not trained from scratch on a new dataset, but instead use a pre-trained CNN as a base for the system. CNNs trained to do object classification on a large dataset, e.g. ImageNet, have been shown to be powerful as general feature extractors for different computational tasks on images [21]. One such pre-trained CNN is *MobileNet* [22], which is used throughout this project. MobileNet is composed of an initial convolutional layer, followed by 13 depthwise separable convolutions (see [22] for further details on the network architecture and [23] for a detailed breakdown on depthwise separable convolutions). Each convolutional block outputs a tensor referred to as the *feature maps* of that convolutional block. The feature maps of the last convolutional block of MobileNet are downsampled by a factor of 32 in the spatial dimensions compared to the input image, with a total feature depth of 1024 units. An image of 224 x 224 resolution would therefore result in feature maps of size 7 x 7 x 1024 from the last convolutional block. When referring to the feature maps and individual (x, y) cells of the feature maps, the terms *grid* and *grid cells* will often be used.

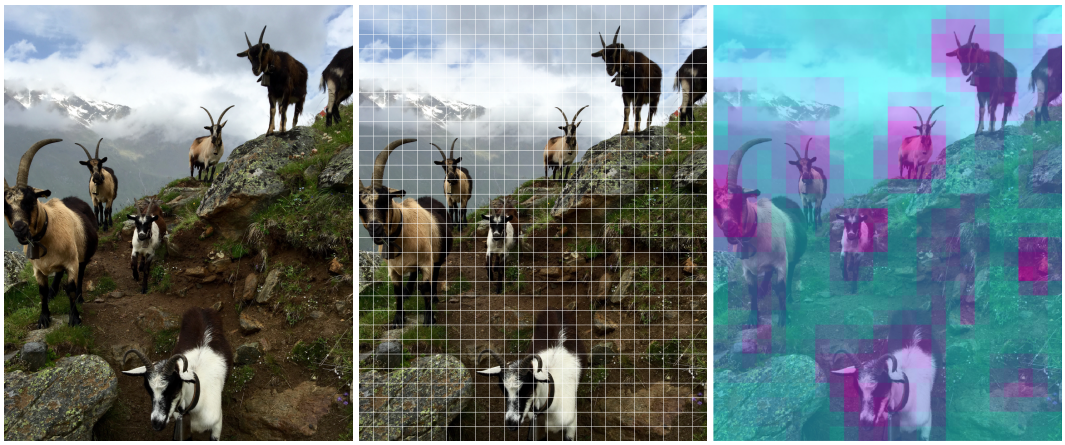


FIGURE 4.6:
Left: 768 x 960 resolution input image, **Middle:** 24 x 30 grid corresponding to the spatiality of the last feature maps, **Right:** Heatmap showing average activation values of the last feature maps of MobileNet (with $\alpha = 1$) when applied to the input image. *Goats roaming the mountainside, Vent, Austria.*

Feature maps from one or several of the last convolutional blocks of pre-trained CNNs are core components in object detection models, and often function as the main information sources for inferring the position and type of objects present in images. Figure 4.6 illustrate how the final feature maps of MobileNet show high average activation values in grid cells where objects of interest are present. In the scenario of classification, a *Pooling* operation would then be applied over the spatial dimensions of the feature maps

to create a single 1024-unit feature vector which acts as the final feature representation for the entire image.

4.2.2 Anchoring & Matching

Anchors are pre-defined reference boxes associated with each grid cell of the feature maps used by object detection models. Each anchor has its own prediction function for bounding box regression and classification, but shared weights across the different grid cells. In other words, an anchor trained to predict objects around 128 x 128 pixels in size operates like a sliding window over the feature maps to determine at the location of each grid cell whether any detections can be inferred based on that feature vector. Some systems, like Faster R-CNN, only use the feature maps of a single scale for all predictions, usually the last or second to last feature maps of the feature extraction network. It is then the job of the anchors to provide multi-scale predictions, where a single feature vector corresponding to one grid cell of the feature maps is used as input for prediction of objects vastly different in size. SSD on the other hand uses feature maps of multiple scales, with individual anchors defined at each scale.

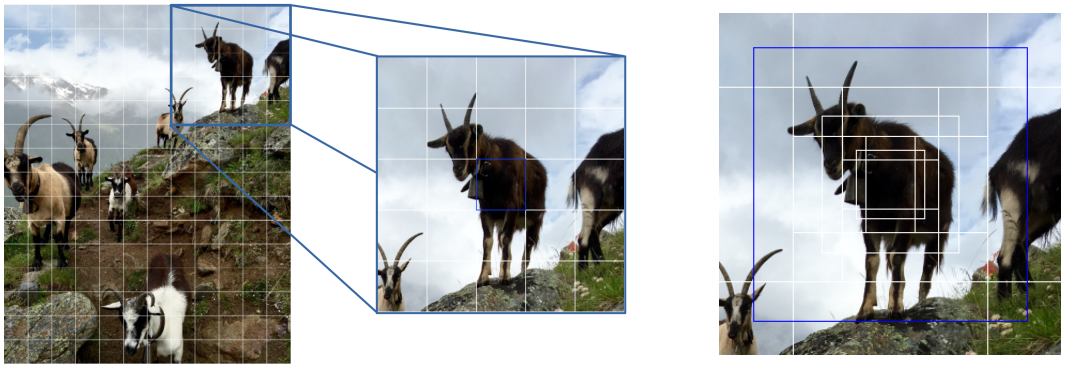


FIGURE 4.7:
Left: Closer look at one grid cell of the feature maps, **Right:** Nine anchor boxes, with the best box marked in blue, for this grid cell.

Figure 4.7 shows a close up of the previous image, with the center grid cell marked on an example feature map grid. Defining nine anchor boxes, Faster R-CNN style, with areas $\{64^2, 128^2, 256^2\}$ pixels and aspect ratios $\{0.5, 1.0, 2.0\}$ for this specific grid cell, results in the set of anchor boxes seen to the right.

During training, every ground truth box is matched with one or several of the anchor boxes based on a matching strategy. Three matching strategies are commonly used:

- Argmax Matching

Ground truth boxes are matched with the anchor boxes with the highest IoU overlap, or with any anchor with an IoU overlap higher than a set threshold (0.5 for SSD and 0.7 for Faster R-CNN). This results in ground truth boxes often matching with several anchor boxes.
Used by: Faster R-CNN, SSD, R-FCN

- Bipartite Matching

  Ground truth boxes are matched with only one of the predictions/anchor boxes. The best pairings between grounds truths/predictions are found by minimizing an assignment cost based on the IoU scores.
  Used by: MultiBox [24]

- Box Center

  Ground truth boxes are matched with the anchor box with highest IoU overlap, where the only anchor boxes considered are those corresponding to the grid cell containing the center point of the ground truth box.
  Used by: YOLOv2

After matching between ground truth and anchor boxes is completed, the final box predictions and object classifications from those specific anchors are compared with the ground truth (generally in a parameterized way, see Section 4.3.2), and the regression and classification losses are calculated.

4.3 Data Preprocessing

Preprocessing refers to the transformations applied to the images and annotations before passing them to the NN.

4.3.1 Image Preprocessing

Images are typically encoded as a tensor of rank 3, where each (x, y) position in the image contains integer values in $[0, 255]$ corresponding to the intensities of each of the color channels red, green and blue (RGB). Preprocessing image data is typically done by zero-centering and/or rescaling the values for the color intensities.

When developing a detection system on top of a pre-trained CNN, it is important to use the same preprocessing scheme as was used when training the pre-trained network. Commonly used preprocessing methods for image recognition and detection are outlined below:

- Mean Subtraction

  Used by the first DL models trained on the ImageNet dataset and performed by subtracting the mean value of each color channel over the entire dataset. Assuming BGR color space (OpenCV default), this means subtracting $(103.939, 116.779, 123.68)$ from the different channels of the input images before passing them to the model.
  Used by: AlexNet, VGG, ResNet

- $[-1, 1]$ Rescaling

  Used by all Google developed models and performed by dividing the pixel values by 255, followed by subtracting 0.5 and multiplying by 2 to confine the pixel values to $[-1, 1]$. The Google models assume RGB color space.
  Used by: Inception, Xception, MobileNet

- $[0, 1]$ Rescaling

  Used by the DarkNet models. Division by 255 assuming RGB color space.
  Used by: DarkNet

In this work, $[-1, 1]$ Rescaling was used since most experiments utilized MobileNet as the feature extractor.

4.3.2 Box Encodings

Bounding box annotations are typically encoded in one of two ways, either in minmax encoding (x_0, y_0, x_1, y_1) or in centered encoding (x_c, y_c, w, h), as defined in Section 4.1.1. Some detection systems (such as the first version of YOLO) train on one of these encodings directly, but it is more common to first perform some sort of preprocessing on the bounding boxes.

The objects in the COCO dataset are heavily skewed towards box widths and heights smaller than 100 pixels, with a distribution similar to a long-tailed gaussian as can be seen in Figure 4.8. A common preprocessing step is log-transforming the widths and heights, resulting in deletion of the long tail and a variance close to 1 for both the width and the height (1.36 for the width and 1.24 for the height).

Additionally, anchor based detection systems typically encode the bounding box coordinates in relation to a matching anchor of the ground truth box. With (x_c, y_c, w, h) representing a ground truth box and (x_a, y_a, w_a, h_a) being the center coordinates, width and the height of a matching anchor, a commonly used encoding ([9], [10], [14], [25]) is to parametrize the coordinates as such:

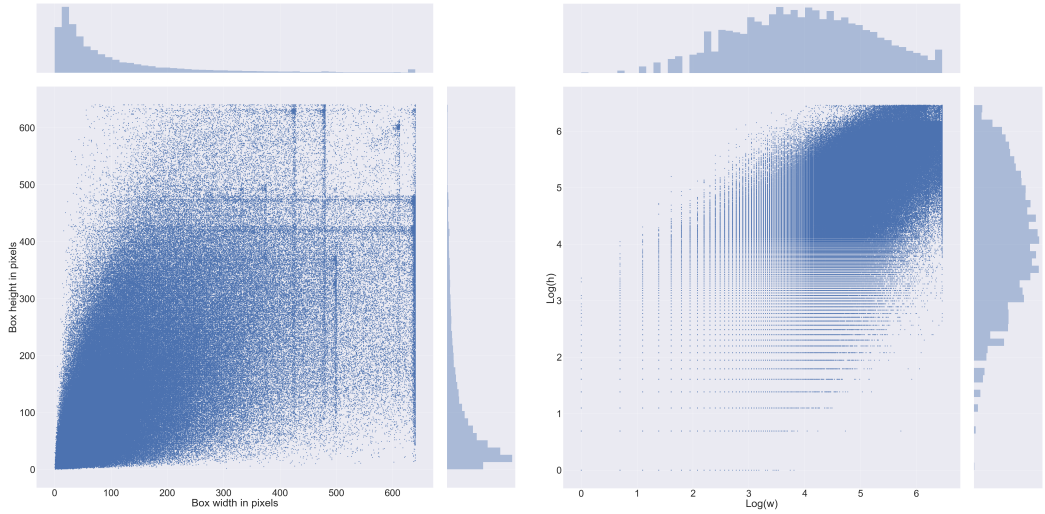


FIGURE 4.8:
Scatter plot and histogram of **Left:** unchanged, **Right:** log-transformed widths and heights of boxes from the COCO dataset.

$$t_x = \frac{x_c - x_a}{w_a},\ t_y = \frac{y_c - y_a}{h_a},\ t_w = log\Big(\frac{w}{w_a}\Big),\ t_h = log\Big(\frac{h}{h_a}\Big) \tag{4.3}$$

The goal of the box regressor is then to infer these parametrized targets given the matching anchor and the features extracted from the image. This encoding has the effect of shifting the regression task towards predicting a scaled offset from the matching anchor box to the ground truth box, instead of predicting the coordinates of the ground truth box directly.

In this project, since no anchors are used, a ground truth box is instead encoded relative to the size and location of the grid cells of the feature maps from the feature extraction. If the center coordinates (x_c, y_c) of a ground truth box are within a grid cell $(x_{gc}, y_{gc}, 32, 32)$ (with x_{gc}, y_{gc} being the center coordinates for the grid cell and 32 being the width and height of the cell), the ground truth box is parametrized as:

$$t_x = \frac{x_c - x_{gc}}{32},\ t_y = \frac{y_c - y_{gc}}{32},\ t_w = log\Big(\frac{w}{32}\Big),\ t_h = log\Big(\frac{h}{32}\Big) \tag{4.4}$$

With a zero-centered initialization of the weights and bias of the network, model predictions $(\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_h)$ will all be close to zero at start of training. This, together with the box encoding in Equation 4.4, leads to the initial box predictions $(\hat{x}_c, \hat{y}_c, \hat{w}, \hat{h})$ from the model to be centered at the location of the grid cell and 32 x 32 pixels in size:

$$\hat{x}_c \approx x_{gc},\ \hat{y}_c \approx y_{gc},\ \hat{w} \approx 32,\ \hat{h} \approx 32$$

4.3.3 Augmentation

Data augmentation is the process of transforming the input data to a learning algorithm while making sure the information relating the input vector to the ground truth label stays intact. The purpose of data augmentation is to increase the quantity of training data, which prevents overfitting and improves robustness of the models.

Image recognition and detection problems are highly susceptible to data augmentation since the input images can be altered to an almost indistinguishable form, as measured by the Frobenius norm or a similar algebraic metric, while still preserving the input-label relationship. The augmentation scheme used in this project consists of:

- Resizing each new batch of training images to a random multiple of 32 in both the x and y direction, where the final image dimension Dim (excluding color channels) becomes:

  $$Dim(i,j) = 32 \cdot (11,11) + 32 \cdot (i,j), \quad (i,j) \sim \mathcal{U}\{0,8\}$$

  Thus the resulting feature map shape is in the range $[11,19]$ and the image dimensions in the range $[352,608]$ in both the x and y directions. The resizing is done by cropping the original image until the aspect ratio matches the new dimensions, followed by bilinear resizing.

- Translation by shifting the image pixels by a random integer $s_{x,y} \sim \mathcal{U}\{-16,16\}$ in both the x,y direction respectively, which ensures that the center position of an object can end up in any one of the $32x32$ locations of a map cell.

- Mirroring with a 50% chance by flipping the image along the x-axis.

- Color distortion by multiplying the pixel values of each color channel by a value in $t_{r,g,b} \sim \mathcal{U}(0.9,1.1)$, followed by rescaling into the $[0,255]$ value range. Additionally, there is a 5% chance of black-and-white recoloring.

- Smoothing with a 50% chance by applying gaussian blur with a kernel standard deviation drawn from $\mathcal{U}(0,1)$.

This scheme ensures that no unique image can be seen twice by the model as well as greatly increasing model robustness to different object sizes. It is also a crucial step to handling common real world situations such as partial occlusions, varying lighting conditions and improves performance on cameras operating with infrared filters (which resembles black-and-white images). Example augmentation output can be seen in Figure 4.9 below.

FIGURE 4.9:
Left: Original image of my family dog Leon. **Right**: Example outputs of augmentation.

4.4 Loss Function

Detection systems generally optimize over a multi-task objective function, combining the regression task and classification task into a single objective. The following descriptions all assume that a prediction has been made by a model and the loss is computed in regards to a single object and its ground truth box.

4.4.1 Regression Loss

As motivated in [9] and used by [10], [14], [25], the Smooth L1 loss is used for the bounding box regression. The Smooth L1 loss for a regression error ϵ is defined as:

$$SmoothL1(\epsilon) = \begin{cases} \frac{\epsilon^2}{2} & \text{if } |\epsilon| \leq 1 \\ |\epsilon| - 0.5 & \text{if } |\epsilon| > 1 \end{cases} \tag{4.5}$$

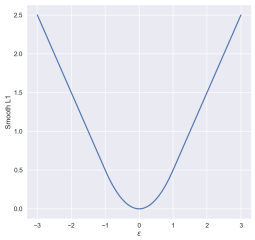


FIGURE 4.10:
Smooth L1.

Given model predictions $(\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_h)$ and ground truth box (t_x, t_y, t_w, t_h), the regression loss L_{reg} per object is the sum of the Smooth L1 loss over the individual regression errors:

$$L_{reg}(t, \hat{t}) = \sum_{i \in \{x,y,w,h\}} SmoothL1(t_i - \hat{t}_i) \tag{4.6}$$

4.4.2 Classification Loss

With C classes, the class prediction of an object by a model is represented by C output values $\{\hat{c}_i\}_{i \in C}, \hat{c}_i \in \mathbb{R}$. The Softmax function, which can be seen as a generalization of the logistic sigmoid for multiple categories, is then applied to each output value:

$$Softmax(\hat{c}_i) = \frac{e^{\hat{c}_i}}{\sum_{j \in C} e^{\hat{c}_j}} \tag{4.7}$$

The output of the Softmax is constrained to $(0, 1)$ and normalized over all classes, which can then be interpreted as the predicted probability distribution (likelihood) for the object classified. With p being the correct class for the object, the classification loss L_{cls} of the prediction is the negative log-likehood for the prediction $\hat{c}_p$:

$$L_{cls}(\hat{c}) = -log(Softmax(\hat{c}_p)) \tag{4.8}$$

This is also known as the *categorical cross-entropy loss* or *multi-class log-loss* and is used as the loss function for the classification task across most popular systems.

Chapter 5

Inflated Multinomial Matching

The main contribution of the proposed algorithm is to provide a method of training DL based object detection models with the capability to detect multiple objects of different shapes and sizes without relying on predetermined anchor boxes to guide the model. Instead of anchor boxes, a set number of *predictors* are defined for each grid cell in the feature maps from the feature extraction network. Lets begin by looking closer at the concept of predictors before presenting the algorithm.

5.1 Predictors

A predictor is similar to an anchor by consisting of a prediction function for box regression and classification with shared weights across all grid cells. The main difference between anchors and predictors is in their bounding box initialization. Each anchor is initialized with its own predefined box width and height and will be responsible for predicting objects of similar size and shape as as result of the commonly used matching strategies, see Section 4.2.2. Predictors on the other hand are all initialized equally and will start out predicting boxes of similar dimensions. Section 4.3.2 covers how the parametrization of the box annotations will affect the initial outputs of the predictors. Figure 5.1 on the next page illustrates the processing pipeline from image to feature maps, predictors and detections.

With all predictors initialized equally, the proposed matching strategy IMM will instead be key to promoting diversity in predicting objects of different shapes and sizes.

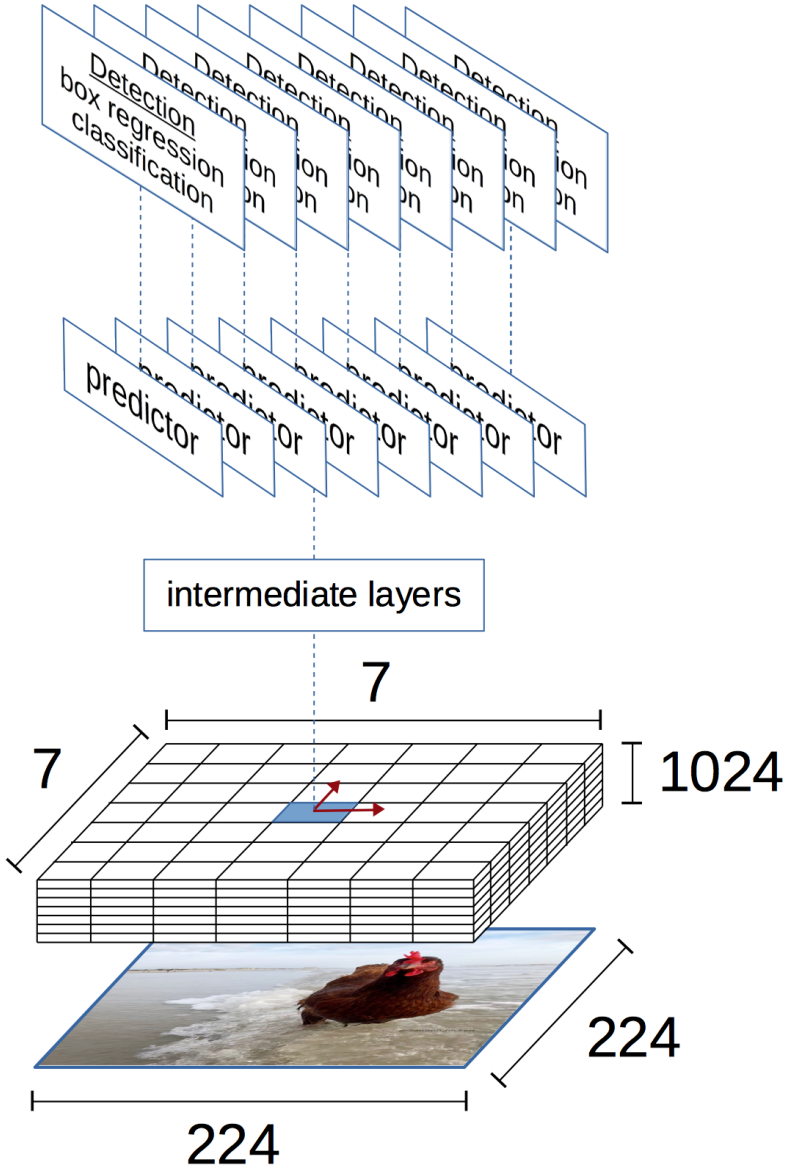


FIGURE 5.1:
With a 224 x 224 resolution image as example input, MobileNet will output feature maps of size 7 x 7 x 1024, where the 1024-unit feature vector of each grid cell acts as a rich feature representation of local image information at and near a 32 x 32 pixel region corresponding to the location of the grid cell. After further enhancing this feature representation through intermediate convolutional layers, typically consisting of 1 x 1 or 3 x 3 filters, we define a set of P predictors where each predictor hold its own set of weights for box regression and classification. Thus, each predictor consists of a 1 x 1 convolution acting on the feature maps from the pre-trained CNN or on the feature maps of the intermediate layers, with a set of $4+1+C$ numbers as output corresponding to the four regression targets and $C+1$ values for the C classes plus background class.

5.2 Algorithm

Let $\mathbf{b}$ denote a ground truth bounding box and $\hat{\mathbf{b}}_i$ denote box prediction i out of a set of m box predictions. The similarity $s_i \in [0, 1]$ between ground truth and prediction i can be evaluated as the IoU score or some other similarity measure:

$$\hat{B} = \begin{pmatrix} \hat{\mathbf{b}}_1 \\ \hat{\mathbf{b}}_2 \\ \vdots \\ \hat{\mathbf{b}}_m \end{pmatrix}, \quad \mathbf{s} = \begin{pmatrix} IoU(\mathbf{b}, \hat{\mathbf{b}}_\mathbf{1}) \\ IoU(\mathbf{b}, \hat{\mathbf{b}}_\mathbf{2}) \\ \vdots \\ IoU(\mathbf{b}, \hat{\mathbf{b}}_\mathbf{m}) \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{pmatrix}$$

Let the normalized similarity vector $\bar{\mathbf{s}}$ represent a probability mass function for a single trial multinomial distribution (also known as categorical distribution), where each individual normalized similarity score $\bar{s}_i$ represents the event probability p_i for box prediction i to be matched with the ground truth box:

$$p_i = \bar{s}_i = \frac{s_i}{\sum\limits_{j \in \{1,m\}} s_j}, \qquad \sum_{i \in \{1,m\}} \bar{s}_i = 1$$

Inflate the probability mass function by executing I iterations of inflation, where inflation is the operation of element-wise raising the vector by power coefficient r, followed by normalization:

$$Inflation(\mathbf{p}, r) = \left(\frac{p_i^r}{\sum\limits_{j \in \{1,m\}} p_j^r} \right)_{i \in \{1,m\}} \tag{5.1}$$

The inflation procedure results in strengthening the probabilities of sampling the better box predictions and weakening the probabilities of sampling the worse box predictions, see Figure 5.3.

Let the inflated normalized similarity scores represent the final probability mass function of the multinomial distribution to be sampled from, where the single trial outcome determines the matching between this ground truth box and box predictions. Finally, compute the (parameterized) regression error between ground truth box and matched prediction box.

The implemented strategy for updating the classification loss is to compare the correct ground truth class with the class of all predictions with IoU-score above a set threshold (where 0.5 was used for the experiments). All predictions with IoU-score below this threshold are considered negative examples and to belong to the background class. A random subset of negative examples with size $N_{neg} = 10 \cdot N_{pos}$ are selected to contribute to the classification loss.

5.3 IMM Example

Consider the following Figure (5.2) with a ground truth box and 5 boxes representing example predictions:

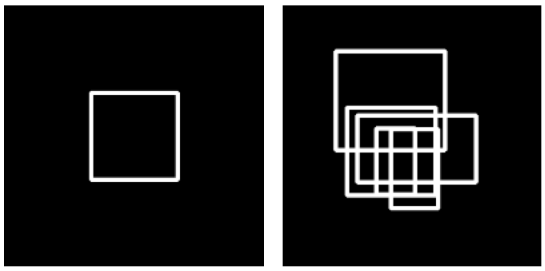


FIGURE 5.2:
Left: Ground truth box, **Right:** 5 example predictions.

The IoU scores $\mathbf{s}$ and corresponding normalized similarity scores $\bar{\mathbf{s}}$ between the ground truth box and the predictions are as follows:

$$\mathbf{s} = (0.24, 0.46, 0.27, 0.56, 0.32), \quad \bar{\mathbf{s}} = (0.13, 0.25, 0.14, 0.30, 0.17)$$

Inflating the probability mass function with power coefficient $r = 2$, results in the following values after $I = \{0, 1, 2, 3\}$ iterations (denoted by $\mathbf{p}_{I=\{0,1,2,3\}}$):

$\mathbf{p}_{I=0} = (0.13, 0.25, 0.14, 0.30, 0.17)$

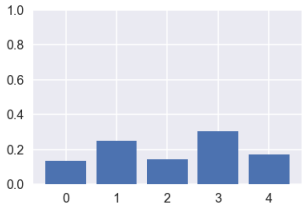
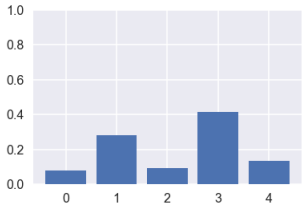


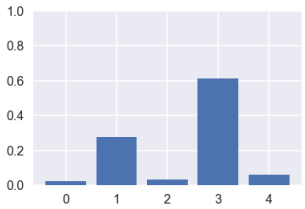
FIGURE 5.3: $I = 0, r = 2.$

$\mathbf{p}_{I=1} = (0.08, 0.28, 0.09, 0.42, 0.13)$



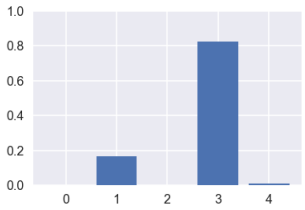
$I = 1, r = 2.$

$\mathbf{p}_{I=2} = (0.02, 0.28, 0.03, 0.61, 0.06)$



$I = 2, r = 2.$

$\mathbf{p}_{I=3} = (0.00, 0.17, 0.00, 0.82, 0.01)$



$I = 3, r = 2.$

Each iteration of inflation results in a redistribution of mass of the probability mass function such that probabilities of outcomes with high relative mass are increased, and probabilities are decreased for outcomes with low relative mass. In the example shown above, zero or one iteration of inflation results in the ground truth box retaining a fairly high relative probability of being matched with one of the worse box proposals, compared to after two or three iterations of inflation where the ground truth box rarely will be matched with a relatively low scoring box proposal.

5.4 Motivation

The core concept of the algorithm is to stochastically match predictions with ground truth boxes throughout the training process. This is performed by assigning matching pairs through sampling from a single trial multinomial distribution, where the event probabilities of matching the ground truth box with each of the box proposals are proportional to the similarity scores between box pairs.

When utilizing predictors and this matching strategy, all box proposals generated by the model will be similar in size and shape at the start of training. Next, when a predictor is matched with a ground truth box and the corresponding model weights are updated based on the regression error, that predictor incurs a slight bias towards predicting boxes with similar characteristics. The stochastic nature of the matching strategy will then act as a feedback loop on this bias, with the same predictor receiving a slightly higher chance to be matched with boxes of similar shape to the previous update because of the higher similarity score. The strategy also has the effect of the same predictor matching numerous times with boxes of completely different dimensions where the similarity score is relatively low. This effect is reduced by increasing the probability of matching with the best predictions through the inflation step.

Through this strategy, the intended end result following training is for each predictor to have independently converged towards predicting boxes of specific shapes and sizes, reflecting the overall distribution of box dimensions present in the dataset. Initial experiments with just using the normalized similarity scores as probabilities for matching, without inflation (this is equivalent to $I = 0$), did not clearly demonstrate this intended behavior. However, after applying inflation, each predictor show a clear tendency towards specializing towards matching with and predicting similarly shaped boxes as was intended. This is discussed in further detail in the next chapter.

The idea of adding the inflation operation comes from *Markov Clustering* [26] [27], where inflation is used together with an *expansion* operation to perform unsupervised clustering on graphs. In Markov Clustering, inflation is used to strengthen and weaken graph connections in a similar manner as presented here in Equation 5.1, where variation of its associated power coefficient r is responsible for regulating the granularity of the clusters. In this work, the power coefficient is kept constant at $r = 2$ and inflation strength is regulated by performing multiple iterations of inflation and subsequent normalization, with the number of iterations I treated as a hyperparameter. Worth mentioning is that similar behavior can be produced by increasing the power coefficient r, as in Markov Clustering, instead of applying multiple iterations of the operation.

Chapter 6

Experiments & Results

This chapter presents experiments conducted on the IMM algorithm, as well as results and example outputs from a full scale detection model trained with IMM.

6.1 IMM Experiments and Results

The main objective of the IMM experiments is to test and confirm the intended behavior of the algorithm, as presented in Section 5.4 in the previous chapter. A number of metrics are monitored for each predictor throughout training, defined to work as indicators of model behavior, where the definition of each metric is presented at introduction during the first experiment. A total of 20 models were trained independently on the same set of training data and validation data, with the same random seed to ensure that all weight initialization and batching logic is identical across training rounds. The 20 different models correspond to setting the number of predictors to a value $P \in \{1, 3, 6, 9, 15\}$ as well as setting the inflation parameter to a value $I \in \{0, 1, 2, 3\}$. The experiment corresponding to setting $P = 9$ and for all values of $I \in \{0, 1, 2, 3\}$ is presented and analyzed in full in this chapter. Additional results with $P = 6$ and $P = 15$ are presented in Appendix B.

6.1.1 Data and Model Setup

The algorithm experiments are conducted on all images from Pascal VOC 07+12 trainval, where 90% is used as training set and 10% is held out as validation set. Most of the predictor specific metrics are evaluated over the training set, while the performance specific metrics such as IoU and localization loss are evaluated on the validation set. The input images were all resized to 416 x 416 pixels using zero-padding to preserve

original image dimensions followed by bilinear resizing. $[-1, 1]$ Rescaling is used as pre-processing scheme, see Section 4.3.1, to match the training of MobileNet and ground truth boxes are encoded as defined in Section 4.3.2. Additionally, no data augmentation was performed during training of these experimental models.

The experimental model is identically defined across all experiments, where MobileNet with $\alpha = 1$ is used as feature extractor with the network weights imported and frozen from a pre-trained ImageNet model denoted `MobileNet_v1_1.0_224`[1] by the TensorFlow team. With Figure 5.1 as reference, we use the feature maps of the last depthwise separable convolutional layer, see Section 4.2.1, and append one additional depthwise separable convolution with 1024 pointwise convolution filters as an intermediate layer before appending P predictors as defined in Section 5.1. The models are trained over a multi-task loss function $L(t, \hat{t}) = \frac{1}{10} L_{reg}(t, \hat{t}) + L_{cls}(t, \hat{t})$ with regression loss L_{reg} as defined in Section 4.4.1 and classification loss L_{cls} as defined in Section 4.4.2. The *Adam* [28] optimizer was used for all experiments, with parameters as suggested in the article, and the models were trained with a batch size of 32 for a total of 5 epochs over the training data.

At each batch update and for each ground truth box, IMM is used to match the ground truth box with one of the box predictions inferred by the P predictors corresponding to the grid cell which contains the center location of the ground truth box. This has the implication of each ground truth box always having exactly P predictions to be compared with, which is important to consider while observing the results from the experiments. This restriction will be mentioned in the next chapter, as an interesting future experiment.

[1] https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md

6.1.2 Experiment: Varying Inflation Iterations with 9 Predictors

Indicator: Scale

Let *scale* denote the average of the width and height of a bounding box, measured in pixels.

Figure 6.1 shows the mean scale of each predictor (represented by the nine lines in the graphs) during the course of training for each value of $I \in \{0, 1, 2, 3\}$ (where each of the four graphs corresponds to one value of I). The mean scales are evaluated over all box proposals generated by each predictor at every grid cell location of the feature maps. The data points in the graphs are updated at a frequency of 5 times per epoch, where the value of each data point is the average scale of each predictor over all batch updates since the previous data point.

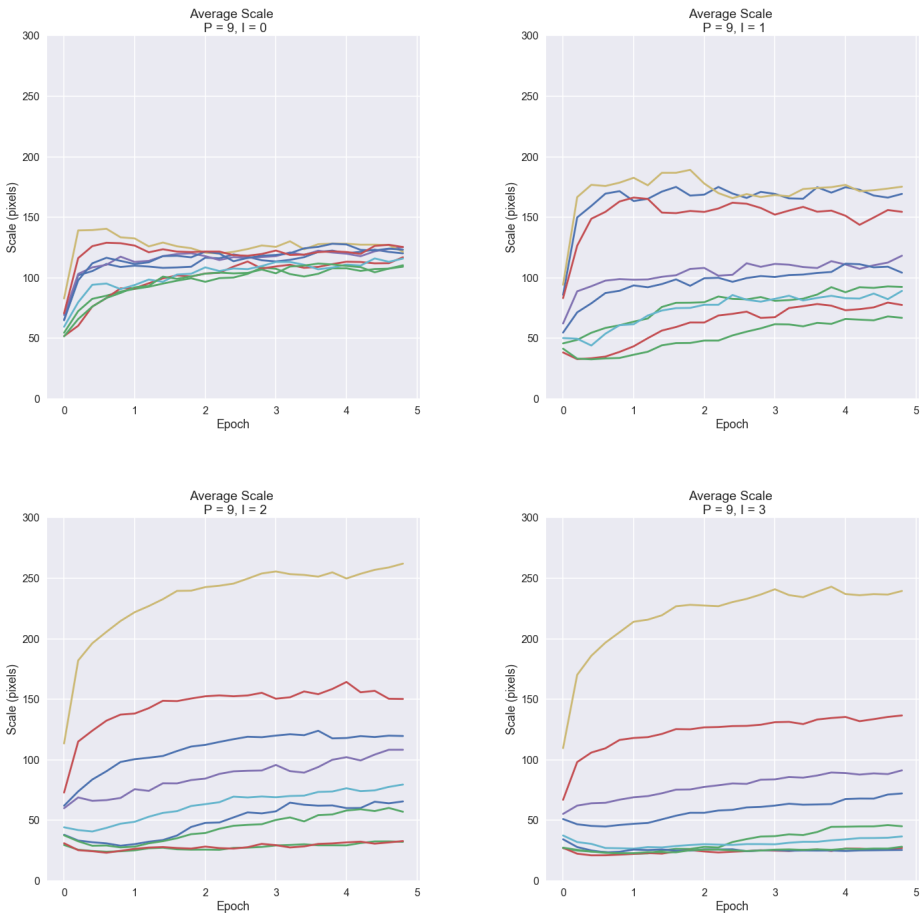


FIGURE 6.1:
Top-left: I = 0, **Top-right:** I = 1, **Bottom-left:** I = 2, **Bottom-right:** I = 3.
Experimental results with 9 predictors on average scale dependent on the inflation parameter.

The first observation of interest is in the first data point at 0 epochs. As mentioned earlier in Section 5.4, we expect to see all predictors inferring boxes of roughly size 32 x 32 pixels at start of training. While this is exactly what was observed during the initial batch updates, this first datapoint corresponds to the first *saved* scale value which happens one fifth of the way into the first epoch (which means after 5000 images have been seen and 156 batch updates performed). This is an unfortunate implementation error, which was not found to be substantial enough to be worthy of rerunning all experiments.

The second observation is the difference in characteristics of the scale values as the number of inflation iterations I are changed. The example presented in Section 5.3 is a useful reminder on the effects I has on the matching procedure. In the top left graph, corresponding to $I = 0$, we observe some initial differentiation of the scale measurements between predictors, with the inter-predictor difference peaking at around 0.6 epochs. We then see this difference decreasing in the scale values, where it is unclear from the data whether the individual predictors have stabilized or whether further convergance towards the same scale value would be observed with further training.

When increasing the value of I and thus imposing a higher probability of sampling the better box predictions at each matching step, we clearly observe a substantial differentiation of the scale values between predictors as training progresses. The top-right graph, showing $I = 1$, shows a clear seperation of the scale values, as compared to $I = 0$. Some predictors (bottom green, middle red) occupy their own segment of the range of box scales, where some other predictors (top yellow and blue) show some interesting interactions on which types of boxes they predict. In the bottom left ($I = 2$) and bottom right graph ($I = 3$), we see a much smoother curve for each of the predictors, where the scale separation occurs early and decisively, and the entire set of predictors appears to converge towards some form of equilibrium. By counting the number of predictors clearly visible in the two graphs, we see two predictors for $I = 2$ and three predictors for $I = 3$ with almost identical curves at scale values 25-30. While this may appear as they are completely unused by the model, we will see in later indicators that this is not completely true.

Indicator: Aspect Ratio

Aspect ratio (AR) denotes the ratio between the width and the height of a bounding box.

Figure 6.2 shows the mean AR of each predictor (represented by the nine lines in the graphs) during the course of training for each value of $I \in \{0, 1, 2, 3\}$ (where each of the

four graphs corresponds to one value of I). As in the case of the scale indicator, the mean AR is taken over all box proposals generated by each predictor at every grid cell location of the feature maps. Same update rules apply as with the scale.

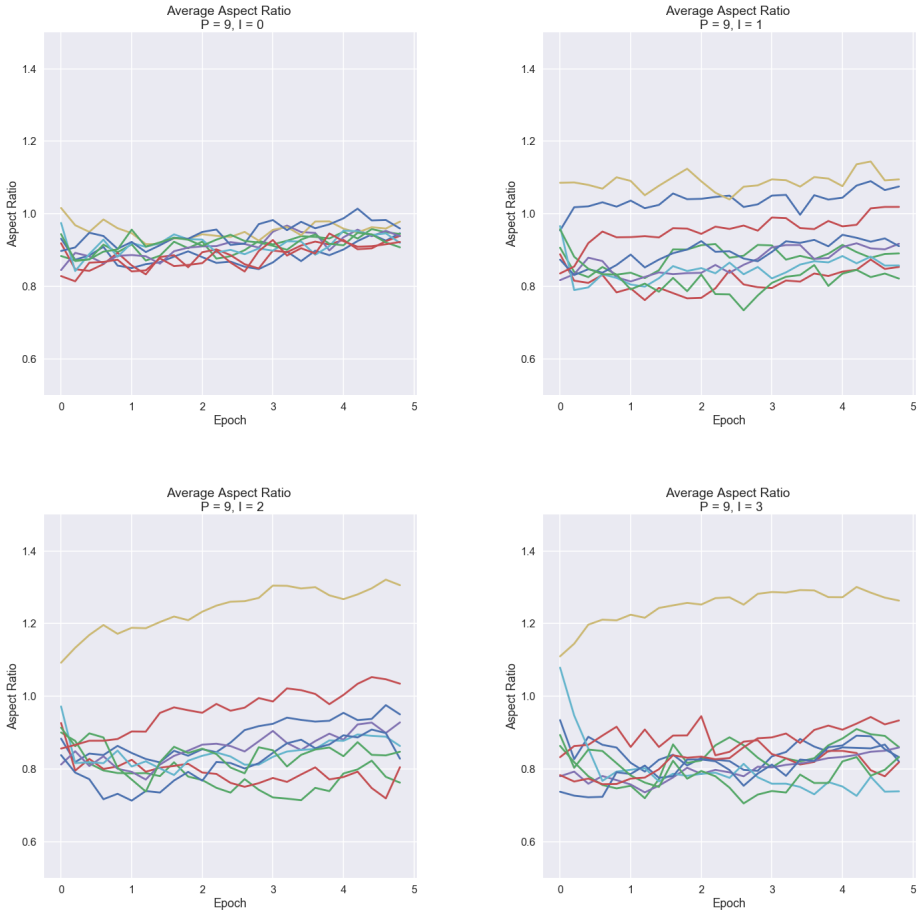


FIGURE 6.2:
Top-left: I = 0, **Top-right:** I = 1, **Bottom-left:** I = 2, **Bottom-right:** I = 3.
Experimental results with 9 predictors on average aspect ratio dependent on the inflation parameter.

Much of the behavior seen by the scale indicator can also be observed in the aspect ratio, but to a lesser degree. Again we observe very similar predictions at $I = 0$ across all predictors, implying together with the scale indicator that all predictors essentially converge towards the average of all boxes in the dataset if no inflation is applied. While we do see a clear distinction when introducing inflation, even at $I = 3$ most of the predictors seem to jitter around $0.7 - 0.9$ AR except for the top yellow predictor, which has specialized towards predicting wide and large objects (the same predictor converged at a scale of 240 pixels in the previous example).

Indicator: Match Ratio

The *match ratio* (MR) for a predictor denotes the proportion of ground truth boxes where this predictor generated the box proposal with highest IoU score.

Figure 6.3 shows the mean MR of each predictor (represented by the nine lines in the graphs) during the course of training for each value of $I \in \{0, 1, 2, 3\}$ (where each of the four graphs corresponds to one value of I). The mean AR is taken over all ground truth boxes, where the top scoring predictor for each ground truth box is counted and compared to the total amount of ground truth boxes at each value update. Same update rules apply as with the scale and AR.

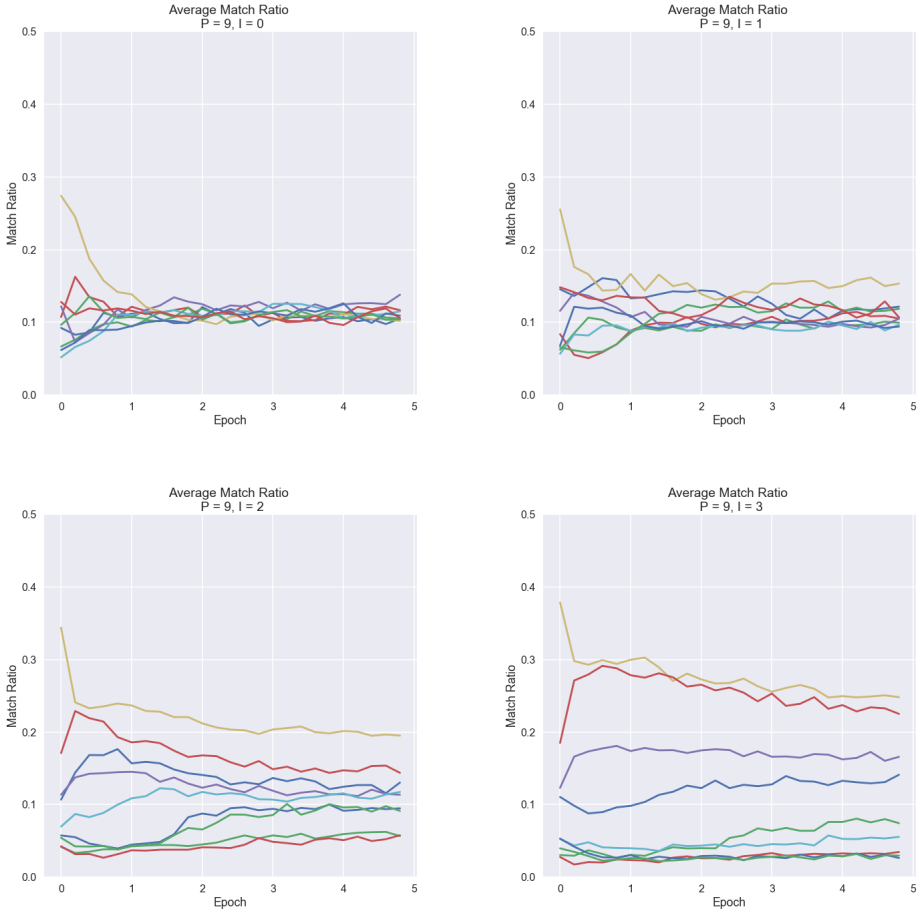


FIGURE 6.3:
Top-left: I = 0, **Top-right:** I = 1, **Bottom-left:** I = 2, **Bottom-right:** I = 3.
Experimental results with 9 predictors on average match ratio dependent on the inflation parameter.

At $I = 0$, the MR of all predictors quickly converge towards the same value, 1/9, again suggesting that all predictors converge towards predicting identical boxes when matching without inflation. After introducing inflation, we see a clear distinction in the MR values

for each predictor. With this indicator, we can settle the dispute from the analysis of the scale results, where we were unsure in the case of $I = 2$ if two predictors went unused by the model. Those predictors are again the same green and red in the bottom of the bottom-left MR graph, where we can observe their contribution in producing the top scoring box proposals for 5.6% and 5.7% of all ground truth boxes. The MR graphs is a useful indicator for determining the value of I. At $I = 3$, we observe three predictors with a MR below 4%, which can be interpreted in two ways. Either the inflation parameter is too high and the probabilities of top scoring boxes are increased such that some predictors never gets updated. Another possibility is that six predictors are enough to capture the entire range of box dimensions in the dataset. The best way to answer these questions is to look at the empirical performance of the models.

Performance: Max IoU

For each ground truth box, the max IoU denotes the maximum IoU value of all the box proposals.

Figure 6.4 shows the mean max IoU over all ground truth boxes during the course of training for each value of $I \in \{0, 1, 2, 3\}$ (represented by the four lines in the graph).

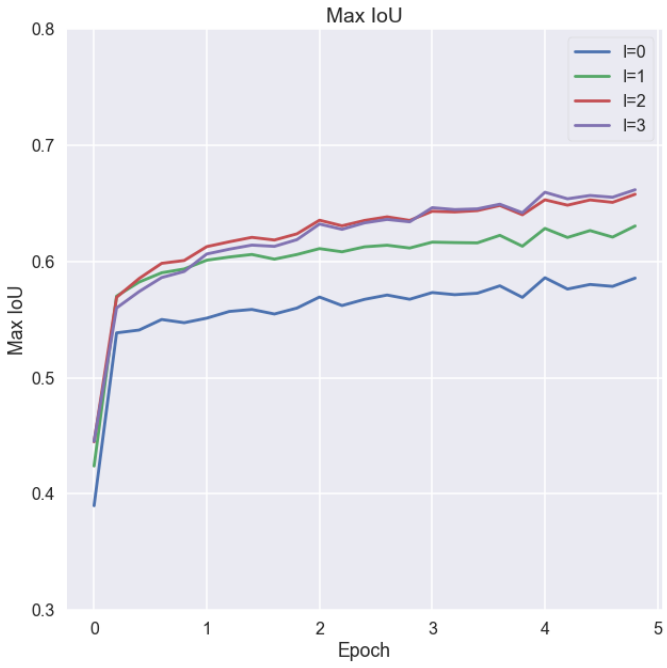


FIGURE 6.4:
Blue: I = 0, **Green:** I = 1, **Red:** I = 2, **Purple:** I = 3.
Experimental results with 9 predictors on Max IoU depending on the inflation parameter.

The first interesting observation in the IoU curves is the similar shapes in the distributions of the highs and lows between the different values of I throughout training. This is due to the random seeding, which ensures that exactly the same images are seen at the same stage of training and also that the same initialization weights are used. Thus, any improvements we see are solely due to the change of the I parameter. We see a large increase in Max IoU between $I = 0$ and $I = 1$, and a smaller increase between $I = 1$ and $I = \{2, 3\}$. There is no significant difference between $I = 2$ and $I = 3$ in the Max IoU performance.

Performance: Regression Loss

The regression loss (loc loss) is computed as in Section 4.4.1 for all matches made by IMM.

Figure 6.5 shows the mean loc loss over all ground truth boxes and their matched predictions during the course of training for each value of $I \in \{0, 1, 2, 3\}$ (represented by the four lines in the graph).

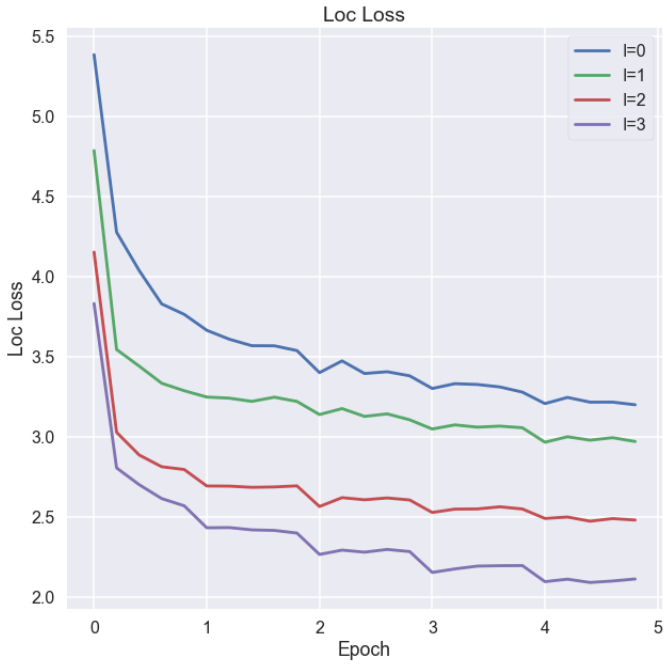


FIGURE 6.5:
Blue: I = 0, **Green:** I = 1, **Red:** I = 2, **Purple:** I = 3.
Experimental results with 9 predictors on Regression Loss depending on the inflation parameter.

Interestingly, for the regression loss we see almost constant improvement for each increase of I.

6.1.3 Experiment: Varying Inflation Iterations and Predictors

Performance: Max IoU

Figure 6.6 shows the mean max IoU for each value of $P \in \{1, 3, 6, 9, 15\}$ (represented by the five lines in the graphs) during the course of training for each value of $I \in \{0, 1, 2, 3\}$ (where each of the four graphs corresponds to one value of I).

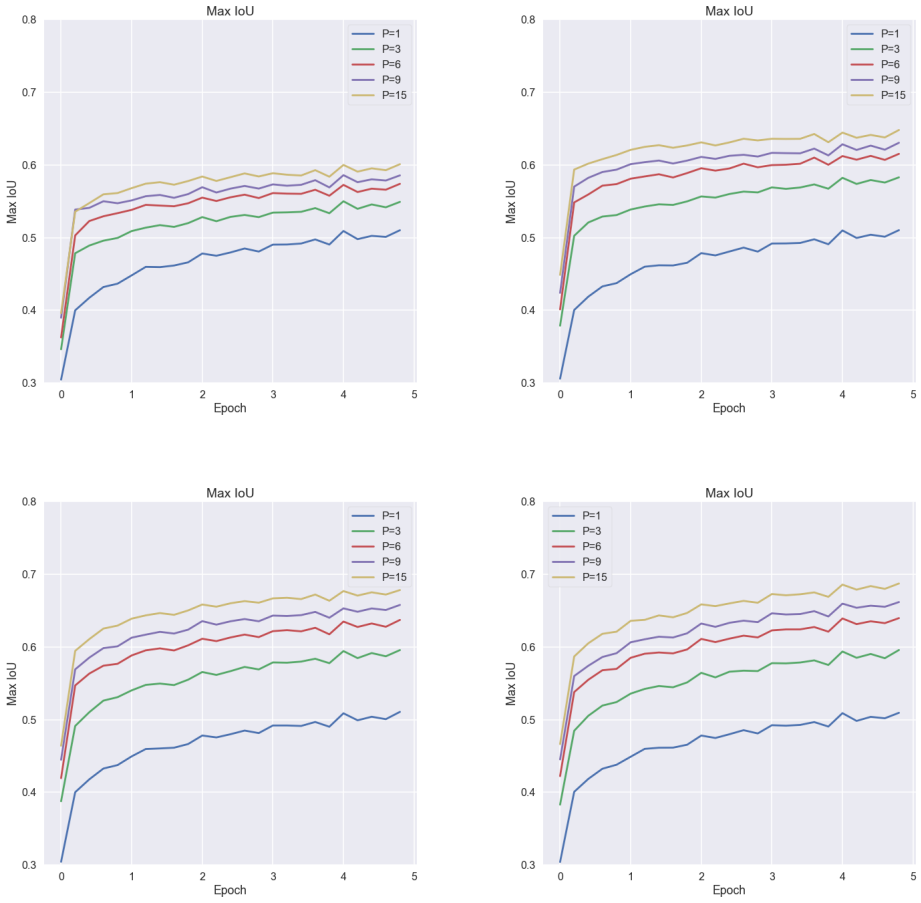


FIGURE 6.6:
Top-left: I = 0, **Top-right:** I = 1, **Bottom-left:** I = 2, **Bottom-right:** I = 3.
Experimental results with varying number of predictors on Max IoU with varying number of inflation iterations. See legend for curve definitions.

This experiment and result shows the mean Max IoU values for each set of P and I. The main observations are that the Max IoU performance increases when adding more predictors, and the introduction of inflation further increases the performance boost of adding predictors. Comparing $I = 2$ and $I = 3$ shows a very small difference in the overall results, which suggests $I = 2$ is a good setting when utilizing IMM (solely based on this set of experiments on the VOC dataset).

6.2 Full Model

For proprietary reasons, the full model definition cannot be presented in the thesis report, and as such this section mostly serves to showcase some detection examples. With the extensive training time and cost associated with training a more complex model on the full COCO dataset, experiments and result analysis similar to the IMM experiments above could not be conducted on the full model. The data and general model setup are included below for the interested reader.

6.2.1 Data and Model Setup

The full model is trained on all images from Microsoft COCO trainval, where again 90% is used as training set and 10% is held out as validation set. When training the full model, the entire data augmentation scheme as defined in Section 4.3.3 is implemented, otherwise the data preprocessing is identical to the setup in Section 6.1.1. This model is more extensive than the experimental model, where again MobileNet with $\alpha = 1$ is used as feature extractor with the same network weights as before, with the main difference that the weights of the last five depthwise separable convolutions are trainable with the earlier layers frozen. It utilizes both the last feature maps of MobileNet as well as the second last feature maps, similar to how SSD defines anchors at multiple scales of the feature extraction network. The second last feature maps has a finer grid structure and the feature vectors are more sensitive to smaller objects in the image, which is overrepresented in the COCO dataset. Due to the size of the grid cells, the default box predictions from this grid will be of size 16 x 16. 15 predictors are defined for each of the feature maps, with 30 predictors in total. Loss function and optimizer are identical to the experimental setup, with the only difference consisting of removing the set limit of training time and instead include early stopping of the training after a full epoch without improvement to the validation loss. IMM with $I = 2$ is used as matching strategy over box proposals from both feature maps, again confined to the proposals corresponding to the grid cells containing the center locations of the ground truth boxes.

6.2.2 Detection Examples

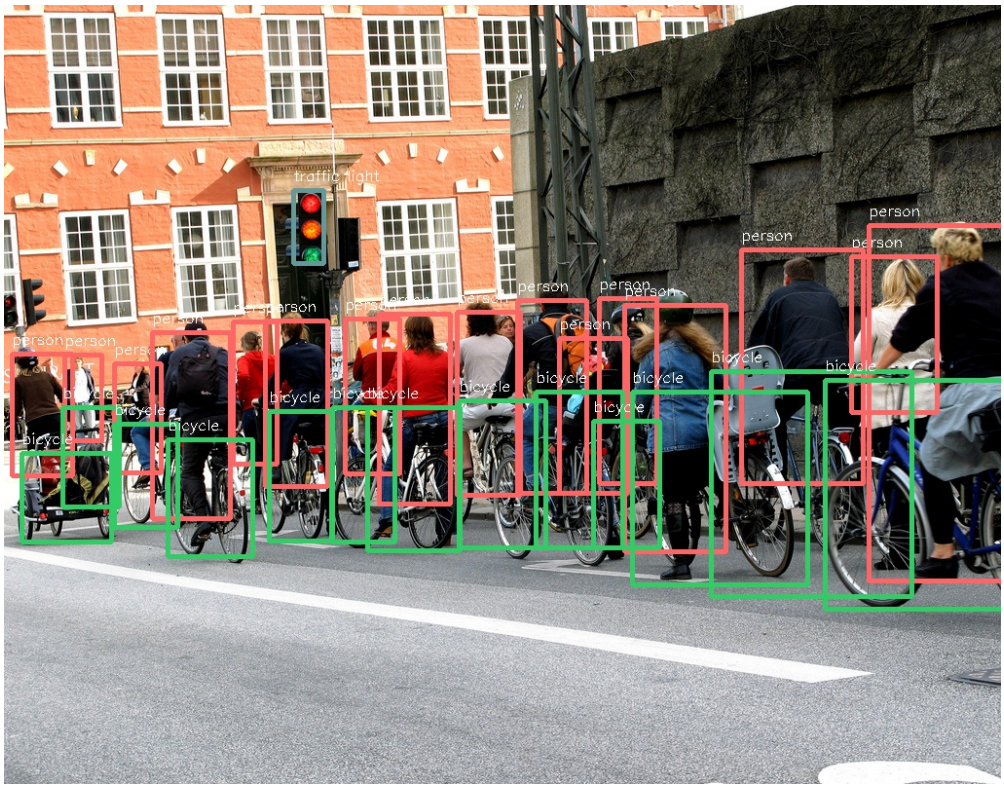


FIGURE 6.7:
Example detection result, processed at 800 x 640 resolution.
Red light in the bike lane, Copenhagen, Denmark. Credit: Mikael Colville-Andersen

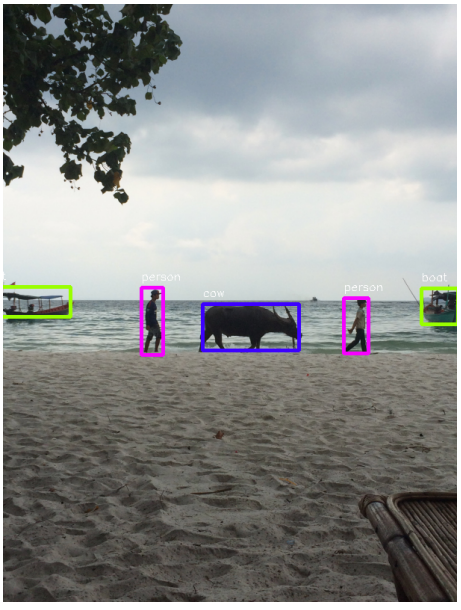


FIGURE 6.8:
Example detection result, processed at 352 x 480 resolution.
Young men with bull, Koh Rong, Cambodia.

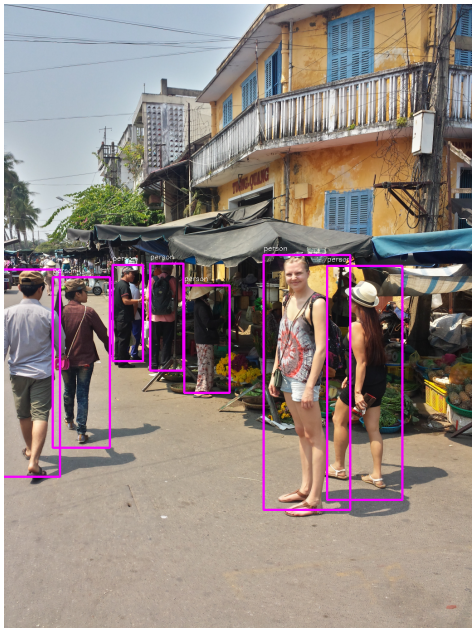


FIGURE 6.9:
Example detection result, processed at 384 x 512 resolution.
Lisa at the Vietnam markets.

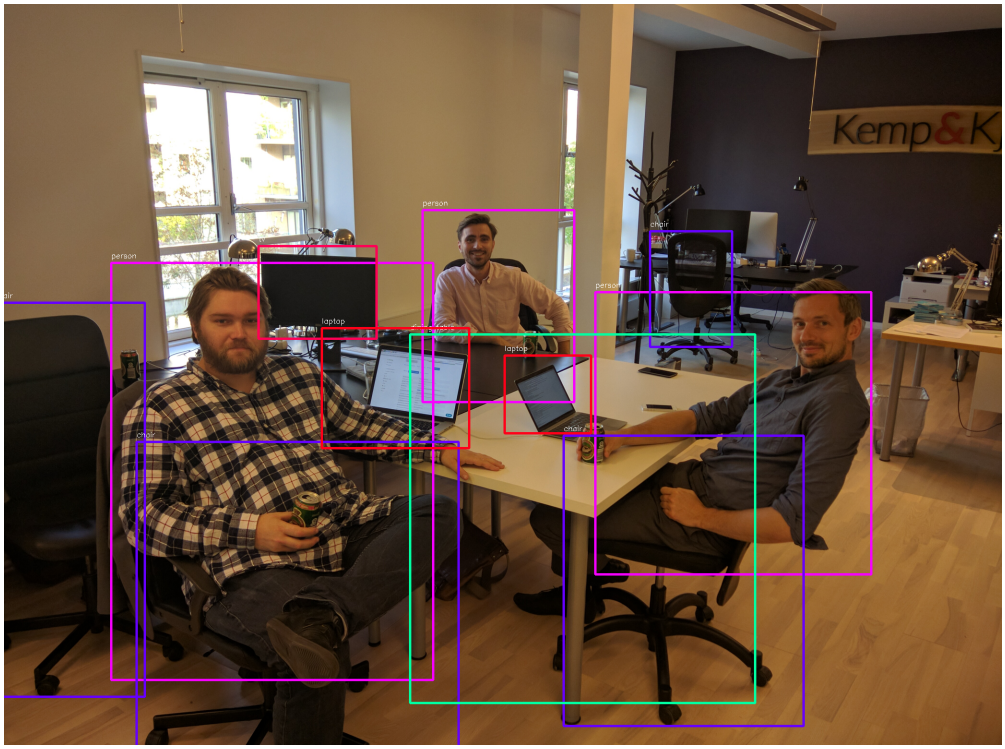


FIGURE 6.10:
Example detection result, processed at 480 x 352 resolution.
First offices of Sentia.

FIGURE 6.11:
Example detection result, processed at 960 x 704 resolution. Note how the coracles (round boats) are classified as *boat* in the water and *bowl* on land.
Shrimp fishing in Mui Ne, Vietnam.

Chapter 7

Conclusions

In this thesis, a novel matching strategy was presented for anchor-free object detection using DL and CNNs. By utilizing the similarity scores between ground truth and predictions in a stochastic way, together with the concept of predictors and the inflation operation, detection models can be trained to attain several submodels where each submodel specializes towards predicting objects of a certain shape and size without the usage of anchor boxes. Through experiments that indicate how the matching strategy affects performance, the intended behavior of IMM is confirmed and example detection results of a full scale model trained with the algorithm is showcased.

For future work, one aspect mentioned in Section 6.1.1 is that the experiments was constrained as to only test the algorithm on a small subset of all predictors. This was due to constraints in processing power, time and cost. Removing this restriction would give a more accurate representation of the full potential of IMM. Furthermore, it would be very interesting to see the mean Average Precision performance on a standard model, such as the MobileNet-SSD or YOLO, with IMM implemented as the matching strategy instead of anchors. Again, due to time and cost constraints, this is an exercise left to the interested reader.

Appendix A

COCO Statistics

Class	person	bicycle	car	motorcycle	airplane	bus	train	truck	boat	traffic light
$\mathbf{N}_{objects}$	268030	7370	45451	9021	5272	6344	4760	10384	11000	13476
Widths	81 (97)	93 (106)	65 (85)	148 (151)	281 (210)	217 (174)	341 (182)	150 (155)	112 (132)	23 (31)
Heights	133 (130)	95 (87)	46 (59)	136 (116)	123 (104)	167 (125)	201 (127)	105 (108)	62 (83)	40 (49)
Scale	138 (135)	115 (111)	70 (88)	169 (153)	283 (210)	235 (172)	355 (178)	158 (157)	120 (135)	41 (50)
Aspect Ratio	0.63 (0.41)	1.01 (0.65)	1.64 (0.99)	1.08 (0.63)	2.52 (1.23)	1.45 (0.96)	2.35 (2.79)	1.60 (0.83)	2.53 (1.99)	0.66 (0.48)
Class	**fire hydrant**	**stop sign**	**parking meter**	**bench**	**bird**	**cat**	**dog**	**horse**	**sheep**	**cow**
$\mathbf{N}_{objects}$	1966	2058	1343	10231	10969	4968	5718	6839	9577	8386
Widths	102 (91)	107 (104)	74 (94)	146 (147)	67 (92)	265 (149)	193 (143)	145 (127)	84 (89)	104 (110)
Heights	188 (153)	118 (107)	135 (147)	86 (96)	64 (88)	231 (126)	184 (131)	160 (127)	77 (80)	96 (101)
Scale	190 (154)	124 (113)	137 (150)	153 (149)	78 (104)	294 (150)	221 (151)	182 (141)	96 (96)	120 (119)
Aspect Ratio	0.56 (0.31)	0.90 (0.37)	0.53 (0.27)	2.57 (3.26)	1.24 (0.73)	1.25 (0.59)	1.15 (0.57)	0.99 (0.54)	1.25 (0.67)	1.28 (0.73)
Class	**elephant**	**bear**	**zebra**	**giraffe**	**backpack**	**umbrella**	**handbag**	**tie**	**suitcase**	**frisbee**
$\mathbf{N}_{objects}$	5736	1365	5535	5360	9085	11672	12882	6700	6411	2796
Widths	179 (140)	248 (144)	169 (128)	179 (125)	57 (61)	116 (115)	48 (55)	44 (57)	124 (123)	67 (64)
Heights	178 (133)	225 (136)	162 (119)	251 (139)	66 (60)	79 (88)	65 (59)	103 (105)	116 (103)	46 (49)
Scale	209 (150)	274 (152)	194 (136)	264 (145)	74 (69)	125 (118)	70 (66)	106 (107)	146 (128)	71 (65)
Aspect Ratio	1.10 (0.57)	1.19 (0.51)	1.15 (0.57)	0.74 (0.39)	0.89 (0.50)	1.99 (1.20)	0.78 (0.48)	0.56 (1.92)	1.17 (0.73)	1.91 (1.17)
Class	**skis**	**snowboard**	**sports ball**	**kite**	**baseball bat**	**baseball glove**	**skateboard**	**surfboard**	**tennis racket**	**bottle**
$\mathbf{N}_{objects}$	6864	2750	6559	9129	3418	3895	5715	6362	5032	25083
Widths	102 (92)	108 (100)	22 (27)	56 (78)	74 (72)	37 (41)	96 (86)	120 (115)	80 (70)	31 (35)
Heights	53 (74)	75 (92)	21 (23)	48 (68)	79 (74)	37 (36)	66 (66)	80 (99)	84 (67)	69 (64)
Scale	116 (102)	134 (113)	23 (27)	65 (86)	106 (83)	42 (43)	109 (91)	146 (129)	104 (78)	70 (65)
Aspect Ratio	3.99 (3.70)	3.07 (3.08)	1.08 (0.36)	1.39 (1.04)	1.61 (1.99)	1.03 (0.37)	2.03 (1.49)	2.97 (2.93)	1.16 (0.93)	0.49 (0.37)
Class	**wine glass**	**cup**	**fork**	**knife**	**spoon**	**bowl**	**banana**	**apple**	**sandwich**	**orange**
$\mathbf{N}_{objects}$	8180	21469	5689	8085	6412	14946	9565	6012	4533	6587
Widths	47 (49)	53 (52)	90 (90)	73 (87)	68 (76)	125 (130)	94 (99)	78 (79)	188 (135)	81 (84)
Heights	86 (82)	65 (62)	75 (75)	62 (69)	59 (61)	89 (102)	89 (89)	66 (67)	151 (111)	70 (70)
Scale	88 (84)	70 (65)	113 (98)	95 (94)	86 (82)	129 (132)	112 (109)	83 (82)	198 (138)	86 (88)
Aspect Ratio	0.60 (0.39)	0.94 (0.72)	1.89 (1.84)	2.01 (2.77)	1.63 (1.74)	1.73 (1.00)	1.20 (0.73)	1.28 (0.64)	1.39 (0.59)	1.24 (0.58)
Class	**broccoli**	**carrot**	**hot dog**	**pizza**	**donut**	**cake**	**chair**	**couch**	**potted plant**	**bed**
$\mathbf{N}_{objects}$	7573	8123	3009	6091	7333	6606	39844	6040	8973	4355
Widths	109 (86)	73 (69)	146 (131)	252 (183)	96 (79)	147 (129)	78 (77)	264 (166)	85 (88)	409 (176)
Heights	96 (73)	64 (58)	111 (107)	163 (137)	78 (70)	115 (107)	86 (78)	191 (112)	108 (108)	267 (137)
Scale	118 (89)	85 (74)	162 (137)	256 (184)	100 (81)	154 (134)	101 (88)	283 (160)	117 (113)	424 (169)
Aspect Ratio	1.20 (0.49)	1.38 (1.03)	1.75 (1.35)	1.93 (1.14)	1.40 (0.56)	1.44 (0.70)	1.15 (0.91)	1.54 (0.96)	0.90 (0.53)	1.84 (1.29)
Class	**dining table**	**toilet**	**tv**	**laptop**	**mouse**	**remote**	**keyboard**	**cell phone**	**microwave**	**oven**
$\mathbf{N}_{objects}$	16390	4328	6091	5191	2367	5983	3007	6684	1727	3477
Widths	346 (225)	151 (94)	137 (105)	183 (131)	58 (59)	54 (68)	192 (144)	59 (73)	138 (117)	189 (152)
Heights	212 (170)	197 (118)	123 (82)	148 (114)	43 (44)	46 (68)	84 (78)	61 (82)	101 (82)	183 (122)
Scale	356 (224)	208 (118)	153 (105)	192 (136)	61 (62)	65 (82)	196 (144)	75 (91)	146 (117)	225 (149)
Aspect Ratio	2.37 (2.17)	0.86 (0.52)	1.18 (0.70)	1.43 (0.82)	1.55 (0.66)	1.64 (1.19)	3.03 (2.08)	1.25 (0.89)	1.47 (0.73)	1.22 (0.99)
Class	**toaster**	**sink**	**refrigerator**	**book**	**clock**	**vase**	**scissors**	**teddy bear**	**hair drier**	**toothbrush**
$\mathbf{N}_{objects}$	234	5834	2760	25206	6587	6851	1500	4919	209	2002
Widths	87 (80)	142 (106)	166 (123)	47 (63)	62 (69)	65 (68)	132 (131)	147 (114)	85 (79)	65 (86)
Heights	79 (77)	74 (73)	282 (145)	47 (49)	66 (69)	103 (100)	123 (113)	169 (126)	87 (75)	84 (96)
Scale	94 (86)	146 (107)	292 (149)	66 (66)	72 (76)	106 (102)	164 (140)	182 (132)	103 (84)	106 (111)
Aspect Ratio	1.23 (0.59)	3.04 (2.62)	0.63 (0.43)	1.51 (1.88)	0.97 (0.47)	0.71 (0.40)	1.34 (1.11)	0.93 (0.39)	1.08 (0.60)	1.21 (1.65)

Class	TOTAL
$\mathbf{N}_{objects}$	886284
Widths	100 (122)
Heights	107 (115)
Scale	131 (137)
Aspect Ratio	1.19 (1.28)

TABLE A.1:
Per class statistics of all the train/val objects (original size) in the Microsoft COCO dataset. The mean and the (standard deviation) is shown for each attribute, with pixels as the unit of measurement. Scale denotes the length of the longest side of the object and Aspect Ratio is the width / height ratio. Statistics over all classes is shown in TOTAL.

Appendix B

Additional Experimental Results

6 Predictors

Indicator: Scale

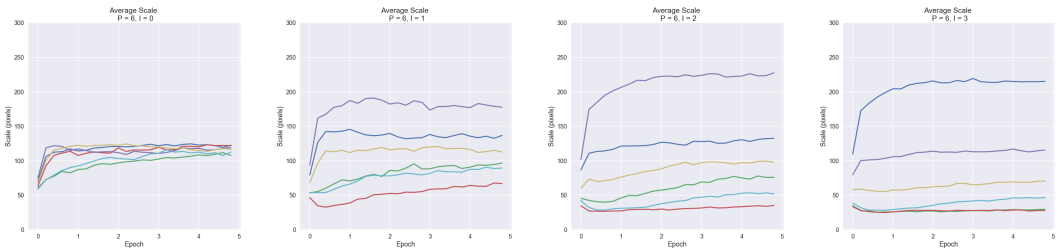


FIGURE B.1:
Left: I = 0, **Middle-left:** I = 1, **Middle-right:** I = 2, **Right:** I = 3.
Experimental results with 6 predictors on average scale dependent on the inflation parameter.

Indicator: Aspect Ratio

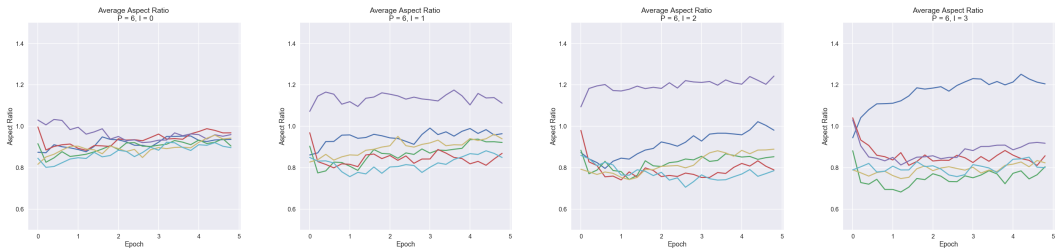


FIGURE B.2:
Left: I = 0, **Middle-left:** I = 1, **Middle-right:** I = 2, **Right:** I = 3.
Experimental results with 6 predictors on average AR dependent on the inflation parameter.

Indicator: Match Ratio

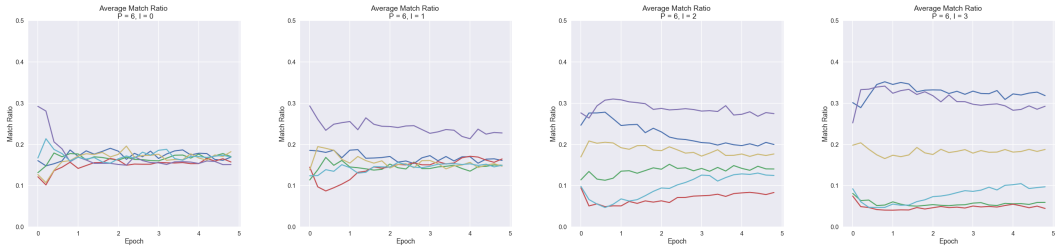


FIGURE B.3:
Left: I = 0, **Middle-left:** I = 1, **Middle-right:** I = 2, **Right:** I = 3.
Experimental results with 6 predictors on average MR dependent on the inflation parameter.

15 Predictors

Indicator: Scale

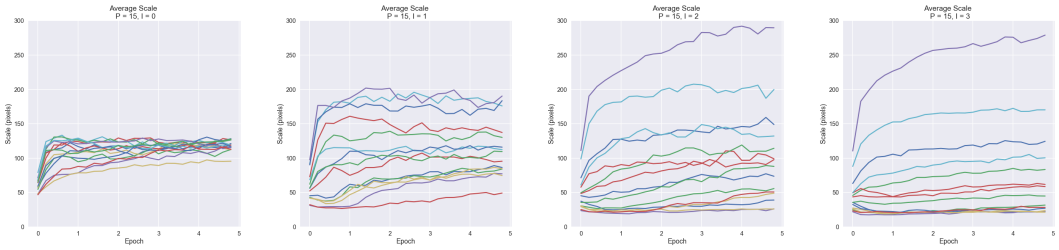


FIGURE B.4:
Left: I = 0, **Middle-left:** I = 1, **Middle-right:** I = 2, **Right:** I = 3.
Experimental results with 15 predictors on average scale dependent on the inflation parameter.

Indicator: Aspect Ratio

FIGURE B.5:
Left: I = 0, **Middle-left:** I = 1, **Middle-right:** I = 2, **Right:** I = 3.
Experimental results with 15 predictors on average AR dependent on the inflation parameter.

Indicator: Match Ratio

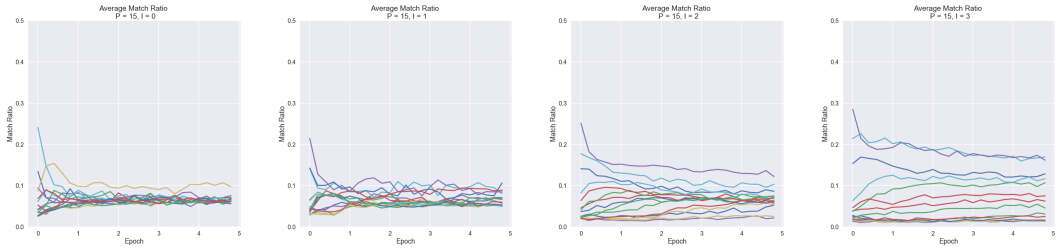


FIGURE B.6:
Left: I = 0, **Middle-left:** I = 1, **Middle-right:** I = 2, **Right:** I = 3.
Experimental results with 15 predictors on average MR dependent on the inflation parameter.

Bibliography

[1] James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73:415–34, 2012 Feb 9 2012. ISSN 1097-4199. URL http://download.cell.com/neuron/pdf/PIIS089662731200092X.pdf?intermediate=true.

[2] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[3] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987, July 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017623. URL http://dx.doi.org/10.1109/TPAMI.2002.1017623.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 05 2015.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[9] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

[10] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[11] 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*, 2013. doi: 10.1007/s11263-013-0620-5. URL http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[12] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

[13] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. URL http://arxiv.org/abs/1612.08242.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL http://arxiv.org/abs/1512.02325.

[15] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016. URL http://arxiv.org/abs/1611.10012.

[16] P. Barham E. Brevdo Z. Chen C. Citro G. S. Corrado A. Davis J. Dean M. Devin M. Abadi, A. Agarwal et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[17] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[18] 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*, 111(1):98–136, January 2015.

[19] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common Objects in Context. *ArXiv e-prints*, May 2014.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[21] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014. URL http://arxiv.org/abs/1403.6382.

[22] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

[23] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL http://arxiv.org/abs/1610.02357.

[24] Christian Szegedy, Scott E. Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *CoRR*, abs/1412.1441, 2014. URL http://arxiv.org/abs/1412.1441.

[25] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016. URL http://arxiv.org/abs/1605.06409.

[26] Stijn van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.

[27] Stijn Dongen. A cluster algorithm for graphs. Technical report, Amsterdam, The Netherlands, The Netherlands, 2000.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Master's Theses in Mathematical Sciences 2018:E66
ISSN 1404-6342

LUTFMA-3366-2018

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/