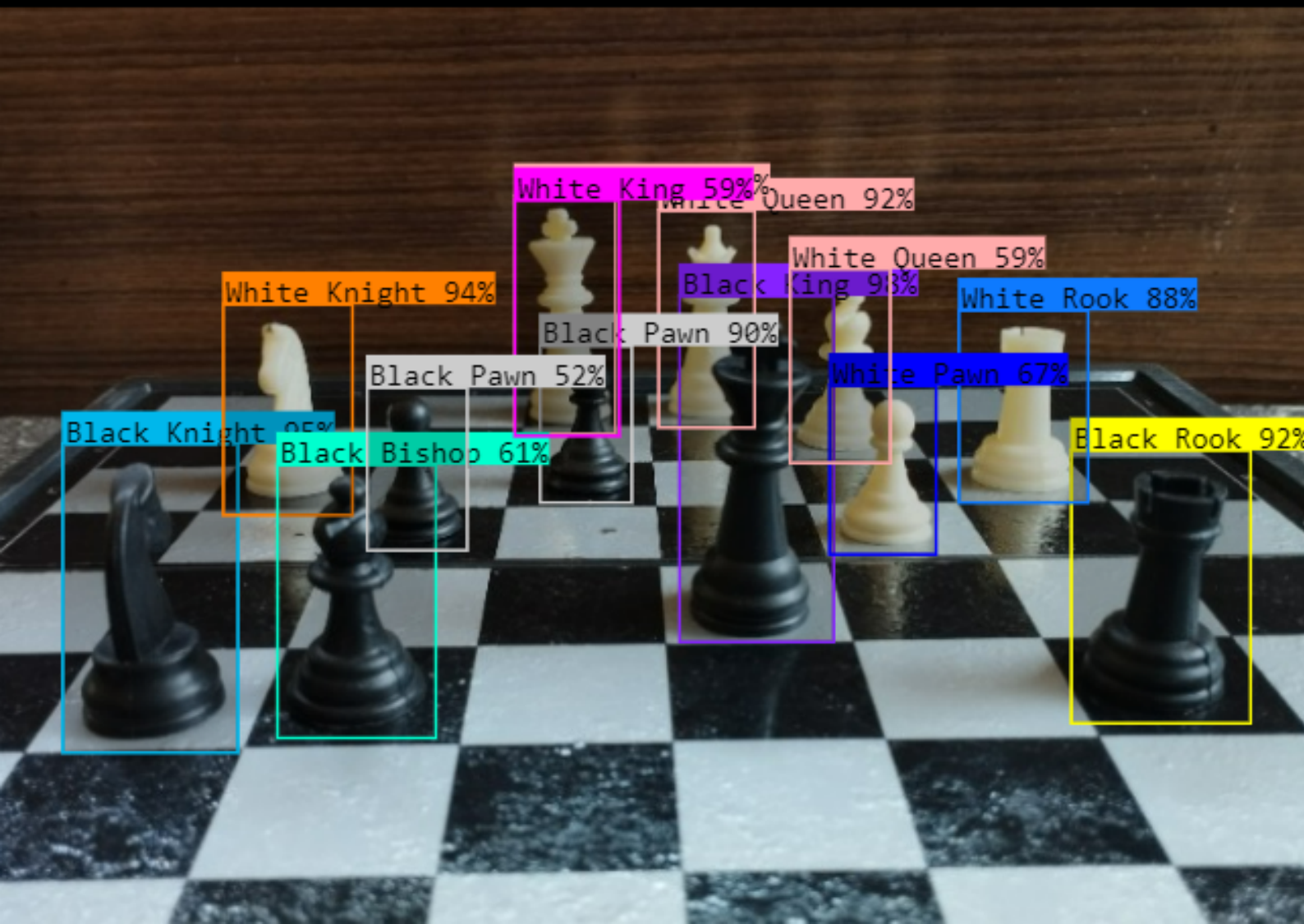
Color Equivariant Object Detection

Integrating Color Equivariance into the Faster R-CNN Object Detection Architecture

MSc Thesis Computer Science

Devin Lieuw A Soe

TU Delft



Color Equivariant Object Detection

Integrating Color Equivariance into the Faster R-CNN Object Detection Architecture

by

Devin Lieuw A Soe

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 30, 2023 at 15:00 PM.

Student number:	4933028	
Project duration:	November 14, 2022 – August 30, 2023	
Thesis committee:	Assoc. Prof. dr. J.C. van Gemert,	TU Delft
	Assoc. Prof. dr. S.E. Verwer,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This report describes the work that I have done for my thesis for the Master of Computer Science at Delft University of Technology.

Firstly, I would like to give a special thanks to my professor Jan van Gemert. It was very helpful to be able to schedule a meeting with him whenever I needed it. During the progression of my thesis, I had to make many considerations and decisions, and Jan was always able to help me in these processes by giving feedback. Secondly, I want to thank Attila Lengyel for all the conversations that I had with him, as well as his critical input and support by providing me with details about his research. Lastly, I would like to thank my friends and family for their continuous support throughout this project.

Devin Lieuw A Soe
Delft, August 2023

Contents

1 Introduction

The field of computer vision has seen a significant rise in recent years. Object detection is an area in computer vision that deals with locating and identifying objects within images. Similarly to how our eyes and brain work, we teach computers to recognize the objects. More specifically, we train an algorithm by exposing it to example images and we help them to recognize patterns. With this training, we aim to make the algorithm better at performing the task, also when dealing with new, unseen images. Face recognition and car license plate detection are two great examples of applications of object detection [1, 2].

Faster R-CNN (Region-based Convolutional Neural Network) [3] is one of the most commonly used neural architectures designed for object detection. Like many other algorithms, Faster R-CNN relies on patterns and structures in the images to identify objects. However, it can occur that the same objects appear in different colors when looking at different images. There could be various causes for this, such as changes in lighting or the use of different kinds of cameras. This raises the question of whether the Faster R-CNN model will still perform well when faced with significant color variation in the image data. As is discovered in this thesis, color variations pose a challenge for the standard Faster R-CNN architecture, and therefore this thesis is focused on improving the robustness of the algorithm to these variations.

Inspired by the work of Cohen et al. [4], this research attempts to enhance the robustness to color variations of Faster R-CNN by making changes to its standard network architecture. These changes improve the ability of the model to use the information that it has learned about patterns and structures within images for different colors, including colors that it has not seen before. Several strategies are analyzed and compared, providing a better understanding of the influence of the adaptations within the network.

This report has the following structure: Chapter 2 is the scientific article of this thesis. It delves into the details of the research, providing an in-depth explanation of the methodologies, experiments, and findings. Chapter 3 explains the core concepts used in the scientific article. The technical background of all used methods is described in a comprehensive manner. Chapter 4 describes and visualizes the image datasets that are used for the experiments of this work.

2

Scientific Article

Color Equivariant Object Detection with Faster R-CNN

Devin Lieuw A Soe[1] , **Jan van Gemert**[1]
[1]TU Delft
d.l.lieuwasoe@student.tudelft.nl

Abstract

This paper studies the effect of integrating color equivariance and invariance into object detection, in particular into the Faster R-CNN architecture. To better understand the influence of this integration, we introduce modifications to the traditional convolutional layers of the standard Faster R-CNN model. By employing group theory in a similar way as Group Equivariant Convolutional Networks (G-CNNs), we replace the convolution operations with operations that are equivariant to hue transformations. The modified models are tested on several different datasets in which variations and imbalances in color distributions are present. Our toy experiments demonstrate that the replacement of the convolutional layers can lead to significant improvements in performance, especially in scenarios where the data contains a substantial amount of color variation. The findings of this work suggest that incorporating color equivariance and invariance into the design of convolutional layers can enhance object detection, proposing interesting possibilities for future research on real-world tasks.

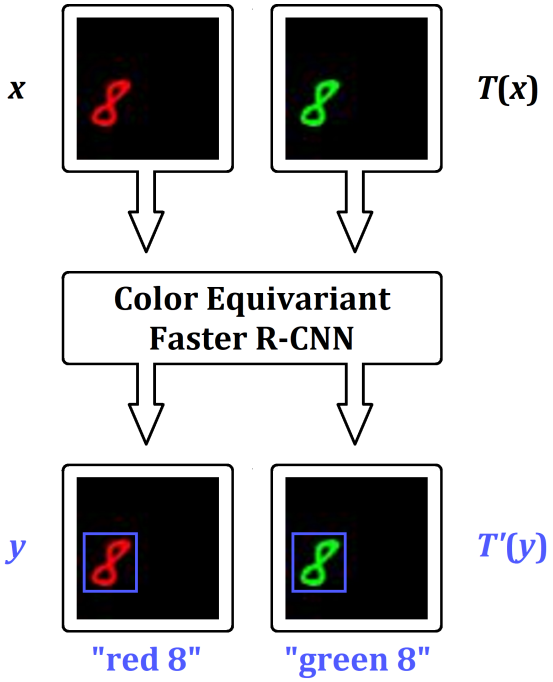


Figure 1: Our modified versions of the Faster R-CNN architecture are color equivariant. More specifically, if a certain hue shift T is applied to an input image x, it will result in a corresponding transformation T' to the output y, which consists of the bounding box and class predictions. An example of a hue shift is the transformation from a red MNIST digit to a green one, as is shown in the figure.

1 Introduction

Color is an integral characteristic of visual perception and has an important role in object detection, as it directly affects the performance of object detection algorithms. The color features of objects can provide essential information that makes it easier to identify or recognize them. Unwanted variations in color, however, which can be caused by factors such as lighting changes or object occlusions, introduce a challenge for these algorithms. Recent advancements in the field of object detection and recognition have achieved significant progress because of the deployment of convolutional neural networks (CNNs) [1]. Despite this, algorithms within this field often struggle to deliver accurate results when dealing with datasets containing a significant amount of color variation or imbalance. This limits their reliability and applicability, especially in scenarios where color variations are prevalent.

The Faster Region-based Convolutional Neural Network (Faster R-CNN) [2] is one of the most commonly used examples of object detection algorithms. It has significantly improved the efficiency and effectiveness of object detection. It introduced an architecture that combines the advantages of both region proposal and classification, enabling object detection. Like other CNN-based models, however, Faster R-CNN struggles with datasets that contain a significant amount of color variation or imbalance. Because of this, its performance in various applications, such as medical imaging [3], autonomous driving [4], and surveillance systems [5], is limited. This forms the motivation of this study to explore how color robustness can improve the effectiveness of Faster R-CNN.

Equivariance and invariance to transformations [6] offer potential strategies to tackle this challenge. Equivariance to a certain transformation implies that transforming the input leads to a corresponding transformation of the output [7]. This can improve the performance of a network on datasets containing variations and imbalances. When a network is invariant to a certain transformation, it means that the output of the network does not change when applying the transformation to the input [8]. Invariance is a special case of equivariance since the corresponding transformation is an identity mapping. This property is often desirable when the classification of an object is independent of its position or orientation for instance. In this work, the influence of color equivariance and invariance on object detection is investigated, in particular on Faster R-CNN.

We introduce a novel approach that is inspired by the framework of Group Equivariant Convolutional Networks (G-CNNs) [9]. The use of G-CNNs provides robustness to various transformations, such as rotation and flipping. G-CNNs achieve this by employing group theory to build transformation equivariant features. This results in the fact that applying a certain transformation to the input will result in a predictable transformation to the output of a convolutional layer. We propose versions of Faster R-CNN that are equivariant and invariant to color changes, in particular hue shifts, see Figure 1. In doing so, we aim to improve the performance of Faster R-CNN on datasets with color variety or imbalance by making use of group equivariant convolutional layers.

In our experiments, we use a simple convolutional neural network as the backbone for the studied Faster R-CNN versions, inspired by [10], and we replace the convolutional layers of the architecture with color equivariant and invariant counterparts. We then assess the performance of the modified models on several toy datasets that are derived from the MNIST dataset [11]. The results of these experiments empirically demonstrate that the replacement of the convolutional layers has a positive effect on the color robustness of the network when dealing with data that contains a significant amount of color variation or imbalance. Furthermore, we conduct experiments on small versions of a dataset containing 2D chess boards and the PASCAL VOC2012 dataset to evaluate the performances on data containing more complex color and shape variations.

The contributions of this research can be summarized as follows:

- We propose and implement modifications to the Faster R-CNN architecture by integrating color equivariant convolutional layers.
- We design comprehensive toy experiments to evaluate the performance of our proposed models in scenarios with varying color distributions. The results of these experiments show that the modifications improve the color robustness with respect to the standard architecture.
- We conduct more complex experiments of which the results suggest that the integration of equivariance within the Faster R-CNN model is an interesting approach to do future research on with real-world experiments.

The next section touches upon research that is done on the equivariance, invariance, and robustness of convolutional neural networks. Following that, the third section of this paper describes the methodology used in the experiments, also providing the details of the studied model architectures. Section 4 discusses the results of the experiments, comparing the standard and modified versions Faster R-CNN. Section 5 concludes the paper and discusses the limitations of this work as well as ideas for future work.

2 Related Work

This section reviews relevant works that focus on improving the robustness to color changes and other transformations. Several approaches integrate equivariance or invariance into network architectures to attempt to improve the robustness, while others use data augmentation to improve the ability of a network to generalize.

Robustness to Color Variation The use of color information can significantly enhance the performance of object detection and recognition. However, datasets containing a substantial amount of color variation, imbalance, or bias often pose challenges for these algorithms. This is because these color variations, which can be caused by factors such as lighting changes, can have a significant influence on the reliability of object detection and recognition algorithms [12, 13]. Several strategies have been proposed to address these challenges by focusing on robustness against color variation. Color constancy, for example, enhances the robustness of color descriptors in image recognition tasks with changing illumination conditions [14]. Further research has also experimented with learning color-constant descriptors which maintain invariance to color transformations [15]. While these methods offer potential strategies for improving the accuracy of object recognition algorithms under color variations, this study focuses more on color robustness of object detection, in particular Faster R-CNN, by incorporating color equivariance and invariance into the network. We investigate the influence of this on the performance of the algorithm.

Group Equivariant Convolutional Networks (G-CNNs) G-CNNs make a notable contribution in the progression of invariant and equivariant neural networks [9]. They employ group theory to build transformation equivariant features, resulting in the fact that applying a certain transformation to the input will result in a corresponding transformation to the output of a convolutional layer. The use of G-CNNs has led to significant advancements in applications such as medical imaging and pattern recognition [16, 17]. Moreover, further research has extended the concepts of G-CNNs to also handle more complex transformations [18]. These studies mainly focus on spatial transformations such as rotation and flipping. In this research, however, we focus on the use of the G-CNN concepts for handling color transformations. By doing this, we attempt to improve the robustness of Faster R-CNN to color variation and imbalance in datasets.

Data Augmentation using Color Transformations Besides the strategy of employing equivariance and invariance in neural network architectures, color variations in datasets

are also commonly handled by performing data augmentation using color transformations. Data augmentation implies enhancing the dataset by applying transformations to its original images, thereby improving the ability of the network to generalize [19]. Color transformations can be executed in different color spaces, for example the RGB and HSV spaces. In the RGB space, techniques like brightness adjustments, color jittering, and channel shuffling have shown to be able to effectively augment data and improve the performance of deep learning models [20]. As for the HSV space, randomly altering the hue, saturation, and values of an image could significantly enhance the performance in object detection tasks [21]. In contrast to making adjustments to the data on which a network is performing to improve its accuracy, we conduct research in altering the network to achieve robustness.

3 Color Equivariant Faster R-CNN

In this section, we describe the methodology used for creating color equivariant and invariant versions of the Faster R-CNN object detection network. We cover the structure of the Faster R-CNN architecture, the principles of Group Equivariant Convolutional Networks (G-CNNs), and the implementation details of the introduced Faster R-CNN variants.

3.1 Faster R-CNN

Faster R-CNN is a powerful object detection model that consists of several primary components. The backbone is a deep convolutional neural network that serves as a feature extractor. The Region Proposal Network (RPN) generates region proposals that potentially contain objects by sliding over the convolutional feature map that is provided by the backbone CNN. The RPN is followed by a detection network for the classification of objects within the proposed regions and refining their spatial locations [2].

3.1.1 Backbone

The backbone of Faster R-CNN is a deep convolutional network that extracts features from the input image. In this work, we use a simple CNN consisting of only three convolutional layers and two subsampling operations as the backbone. The mathematical operation of a convolutional layer can be described as:

$$[f \star \psi](x) = \sum_{y \in \mathbb{Z}^2} \sum_{c=1}^{C^l} f_c(y) \psi_c(y - x). \tag{1}$$

where:

- $[f \star \psi](x)$ is the convolutional operation of the function f with the kernel ψ at position x.
- $y \in \mathbb{Z}^2$ indicates that y is iterating over a 2D integer grid, representing spatial positions in the input.
- c is the channel index iterating from 1 to C^l, where C^l is the total number of channels in layer l.
- $f_c(y)$ is the value of function f at position y for channel c.
- ψ_c is the value of the kernel ψ for channel c.

3.1.2 Region Proposal Network (RPN)

The RPN is a fully convolutional network that generates proposals for object regions. The feature map that is the output of the backbone is used as the input for the RPN. With a sliding window approach and a given set of anchors, the RPN predicts the objectness scores for each anchor, which is a quantification of the likelihood that a certain anchor box contains an object, and therefore not only background. Also, the RPN predicts bounding box adjustments, which are essentially refinements for each anchor box.

3.1.3 Region of Interest (RoI) Pooling and Detection Network

The region proposals that are generated by the RPN are fed into the RoI pooling layer. This layer crops these regions from the feature map produced by the backbone CNN. Subsequently, these crops are fed into the detection network, which consists of a series of fully connected layers for classification and further adjustment of the bounding boxes.

3.1.4 Total Loss Function

The total loss function for Faster R-CNN combines the RPN and detection network losses and can be represented as:

$$\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{det}} \tag{2}$$

By combining both losses, the process of backpropagation ensures that the weights of both the RPN and the detection network are updated such that the model is able to return more accurate bounding boxes and class scores.

3.2 Group Equivariant Convolutions

Group equivariant convolutions offer a way to build neural networks that respect specific symmetries. [9]. This concept is particularly useful when data consists of certain symmetries that the network should consider.

3.2.1 Equivariance

A convolutional neural network layer Φ is said to be *equivariant* to a symmetry group G if the transformations in the group act similarly on the input and the resulting feature mapping that is the output of the layer. In mathematical terms, equivariance can be represented as:

$$\Phi(T_g x) = T'_g \Phi(x), \quad \forall g \in G, \tag{3}$$

where T_g and T'_g are the transformation operators corresponding to a group action g on the input and output, respectively.

Translation Equivariance A special instance of the concept of equivariance is when T_g and T'_g are identical, such as in the case of translation equivariance. If shifting the input results in an equally shifted feature map, the equivariance condition simplifies to:

$$\Phi(T_g x) = T_g \Phi(x), \quad \forall g \in G. \tag{4}$$

Invariance Another special instance of equivariance is when the transformation on the input leaves the output feature map unchanged, implying that T'_g is the identity mapping. In this case, the property is known as *invariance*:

$$\Phi(T_g x) = \Phi(x), \quad \forall g \in G. \tag{5}$$

3.2.2 Extending Convolutional Operations

The standard convolution operation in a CNN layer can be described using the correlation between feature maps f and filters ψ, see Equation (1). By considering a group action g instead of translation x, this definition can be extended to G-CNNs:

$$[f \star \psi](g) = \sum_{h \in G} \sum_{c=1}^{C^l} f_c(h) \psi_c(g^{-1} y). \tag{6}$$

In the context of G-CNNs, it is often desirable to achieve invariance to a specific subgroup H of the symmetry group G. This can be accomplished by applying a max pooling operation over the cosets of the subgroup, thereby reducing the sensitivity of the network to transformations in H. Given a feature map f, the max pooling operation over the cosets of H can be defined as:

$$f_{\text{pooled}}(gH) = \max_{h \in H} f(gh), \quad \forall g \in G, \tag{7}$$

where gH denotes the coset of H under the action of g, and the max pooling is performed over all elements h in the subgroup H. This operation allows the network to focus on the most prominent features that are consistent across the subgroup, which improves the ability to recognize patterns that are invariant to transformations in H.

3.2.3 Color Equivariant Convolutional Layer (CEConv)

Color equivariant convolutions provide a robust feature representation that stays consistent under changes in hue. The hue value is encoded by an angular scalar value within the HSV color space. A hue shift is performed as follows:

$$H' = (H + \theta) \mod 360 \tag{8}$$

where H and H' are the original and transformed hue values, respectively, and θ is the rotation angle. When projecting the HSV representation onto the RGB color space, the same hue shift becomes a rotation along the $[1, 1, 1]$ diagonal vector.

In the context of group equivariant convolutions, we can define a color equivariant convolutional layer (CEConv) by considering the group $G = \mathbb{Z}^2 \times H_n$, which represents a direct product of the 2D translations and discrete hue shifts.

CEConv Layer The CEConv layer in the input can be defined as:

$$[f \star \psi](x, k) = \sum_{y \in \mathbb{Z}^2} \sum_{c=1}^{C^l} f_c(y) \cdot H_n(k) \psi_c(y - x), \tag{9}$$

where f is the input, ψ is the filter, and $H_n(k)$ represents the orthogonal matrix corresponding to a hue shift by k.

Equivariance to Hue Shifts The main idea of the CEConv layer is that it is equivariant to hue shifts. We define an operator $L(t, m)$ as:

$$[L(t, m) f](x) = H_n(k) f(x - t). \tag{10}$$

where t is a translation and m is a hue shift, both acting on input f. Using the orthogonal property of H_n, we can derive the relationship between a hue-shifted input and a filter:

$$H_n f \cdot \psi = f \cdot H_n^{-1} \psi. \tag{11}$$

The above relationship leads to the equivariance property:

$$\begin{aligned}[[L(t, m) f] \star \psi](x, k) = \sum_{y \in \mathbb{Z}^2} & f(y) \cdot H_n(k - m) \\ & \cdot \psi(y - (x - t)),\end{aligned} \tag{12}$$

where we apply the hue shift and translation to both the input and the filter. By rearranging, we have:

$$\begin{aligned}[[L(t, m) f] \star \psi](x, k) &= \sum_{y \in \mathbb{Z}^2} f(y) \cdot H_n(k - m) \\ &\quad \cdot \psi(y - x + t) \\ &= [L'(t, m)[f \star \psi]](x, k).\end{aligned} \tag{13}$$

In this way, the CEConv layer ensures that the transformations in the color space (hue shifts) are reflected in the resulting feature maps, preserving the relationships between different colors in the images.

3.3 Implementation of CEConv Layers in Faster R-CNN

In our approach, the CEConv layers are implemented in a way that is similar to Group Convolution (GConv) [9] by extending the feature map tensor with an additional dimension to account for the color transformations. This allows for the incorporation of spatial and hue transformations within the convolutional layers of the Faster R-CNN architecture. We describe the main components of this implementation, as well as the implemented versions of Faster R-CNN that are experimented with.

3.3.1 Extended Feature Map Tensor

The feature map tensor X is extended with an extra dimension G^l of size $[C^l, G^l, H, W]$, where C^l represents the number of channels, G^l denotes the transformations that leave the origin invariant (including hue shifts), and H and W represent the height and width of the feature map at layer l. The dimension corresponding to the batch size has been excluded.

3.3.2 GConv Filter

A GConv filter $\tilde{F}$ is of size $[C^{l+1}, G^{l+1}, C^l, G^l, k, k]$, where the dimensions correspond to the channels and transformations of the next layer and the current layer, as well as the spatial dimensions of the filter. We define the GConv in terms of tensor multiplication operations:

$$X^{l+1}_{c',g',:,:} = \sum_{c=1}^{C^l} \sum_{g=1}^{G^l} X^l_{c,g,:,:} \tilde{F}^l_{c',g',c,g,:,:}. \tag{14}$$

This operation essentially captures the relationship between spatial structures and color transformations in the feature map and the filter, resulting in a new feature map that preserves the spatial and color relationships.

3.3.3 Implemented Versions of Faster R-CNN

We experiment with several different versions of the PyTorch implementation of the Faster R-CNN model[1], all using a simple CNN with three convolutional layers and two subsampling operations as the backbone:

- **Standard Faster R-CNN (ST):** This version is the standard implementation of the Faster R-CNN model with the simple CNN integrated as the backbone.
- **Grayscale Faster R-CNN (GS):** For this version, the standard implementation of Faster R-CNN is used as the architecture, exactly as for the ST version. However, the input images are converted to grayscale images before they are fed to the model. This eliminates the ability of the model to use color information.
- **Faster R-CNN with a Color Equivariant Backbone (CEB):** In this version, the Faster R-CNN architecture is modified such that the simple backbone is color equivariant. The convolutional layers of the backbone are replaced by CEConv layers, max pooling subsampling operations are replaced by group max pooling subsampling operations, and a 3D convolutional layer serves as a fully connected layer for each pixel, eliminating the group dimension.
- **Faster R-CNN with a Color Invariant Backbone (CIB):** This version is similar to the CEB version, but the last layer of the backbone is a group coset max pooling that eliminates the group dimension and provides the color invariance property.
- **Fully Color Equivariant Faster R-CNN (CE):** This version features a fully color equivariant architecture. The backbone is adapted with CEConv layers and group max pooling subsampling operations. There is no last pooling layer in the backbone, so the output feature map retains the group dimension. The convolutional layer in the RPN is replaced by a CEConv layer which is followed by a 3D convolutional layer.
- **Fully Color Invariant Faster R-CNN (CI):** This version features a fully color invariant architecture and is similar to the CE version. However, a group coset max pooling follows the CEConv layer in the RPN, providing the invariance property.

All CEConv layers that are used to replace the standard convolutions have 3 rotations of 120°, implying that the extra dimension of the feature map tensors that represents the transformation group is of size 3. This also implies that there is an increase in the total amount of parameters of the modified versions. This is accounted for by lowering the number of hidden channels within the backbone CNN. We perform an ablation experiment with the amount of parameters in the appendix.

The replacement of the convolutional layers in the modified Faster R-CNN versions enables the models to share shape information across different colors. This allows the models to generalize shape features over different colors, which can for example be useful for recognizing an object that appears in different colors but maintains the same shape.

4 Experiments

In this section, we conduct experiments with the implemented versions of Faster R-CNN. We aim to compare the levels of color equivariance, invariance, and robustness by experimenting with different kinds of toy datasets. The goal of these experiments is to gain a better understanding of the influence of the modifications in the Faster R-CNN network on the detection results. Furthermore, we conduct an experiment on a 2D chess board dataset that contains more complex color and shape variations.

All experimented models are trained for 200 epochs using the Stochastic Gradient Descent (SGD) optimizer [22] with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.0005. Each model is trained 5 times with different random initializations, and the results show the average of the highest measured Mean Average Precision (mAP) [23] scores.

Experiment 1: Color Equivariance In the first experiment, we compare the ability of the Faster R-CNN versions to distinguish objects that differ in color. For this, we generate a variant of the MNIST dataset [11] that contains images of size 64 x 64 with a randomly scaled digit on a random position within the image frame. The digit is either red, green, or blue, and each possible combination of the digits and the colors is assigned a class label, implying that there are 30 classes in total. Each image has a gray background that contains noise. The train set consists of 1514 images. In this train set, we introduce a class imbalance by drawing the number of samples per class from a power law distribution. This results in different class frequencies. On the other hand, the test set consists of 960 images where each class contains 32 images.

[1] https://pytorch.org/vision/main/models/faster_rcnn.html

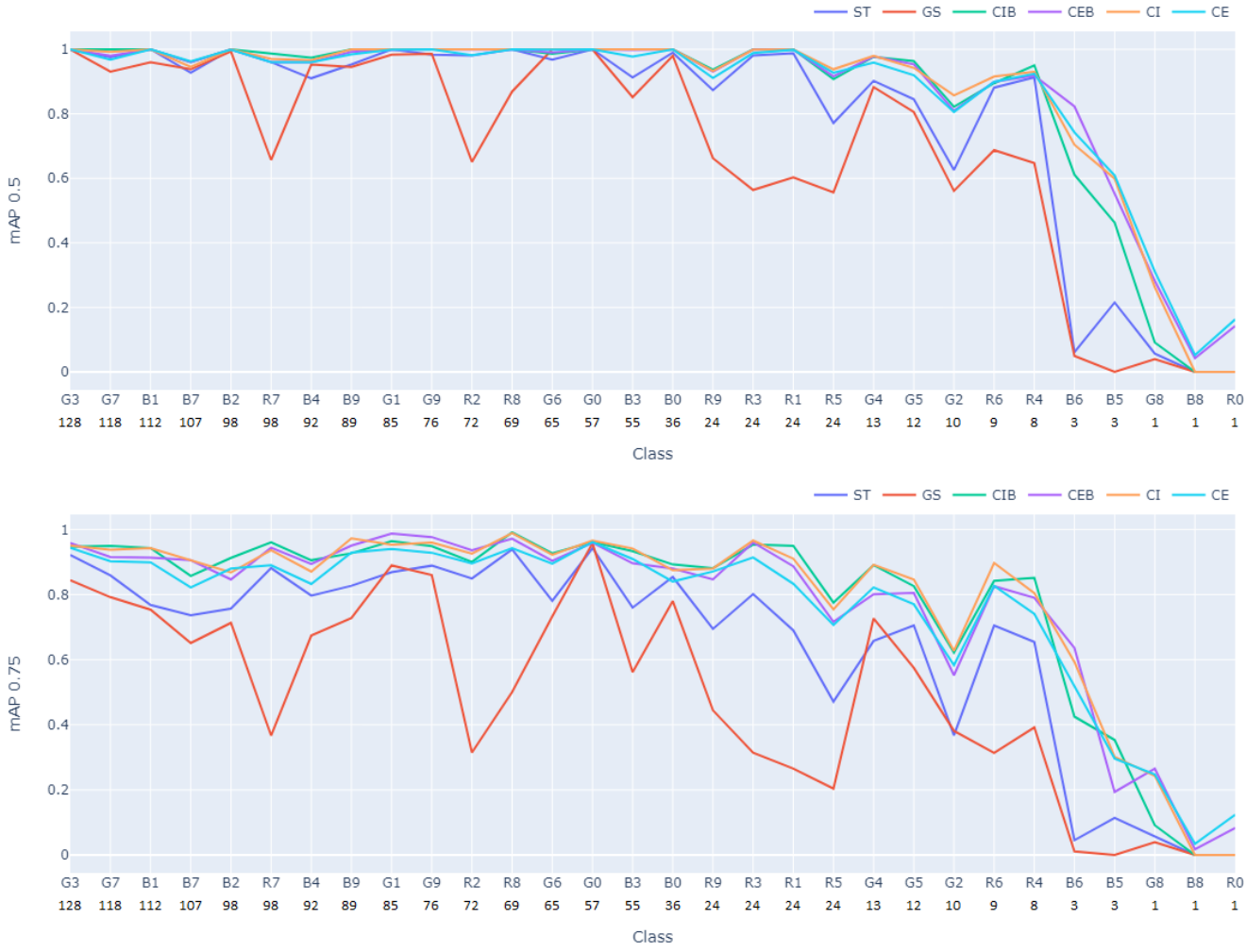


Figure 2: The Mean Average Precision (mAP) results of Experiment 1. The top figure shows the mAP measurements for an Intersection over Union (IoU) of 0.5 and the figure below shows the mAP measurements for an IoU of 0.75. On the horizontal axis, all 30 classes are listed along with the corresponding amount of samples (drawn from a power law distribution) contained in the training dataset. G3 denotes a green 3, B0 denotes a blue 0, R5 denotes a red 5, etc. The results show that the replacement of the convolutional layers with CEConv layers has a positive effect on the performance, especially for the classes that have lower frequencies in the train set.

By using these datasets for training and testing, we investigate the influence of replacing the convolutional layers of the Faster R-CNN architecture on the efficiency in sharing shape information across different colors. The models are trained with a batch size of 8.

The results, presented in Figure 2, show that the replacement of the convolutional layers has a positive effect on the performance. Especially for the classes that have lower frequencies in the train set, the performances of the modified models are significantly higher than that of the standard model. Furthermore, the Color Equivariant Backbone (CEB) and Fully Color Equivariant (CE) versions perform slightly better than the other versions on the classes with extremely low amounts of samples. This indicates that the shape information across different colors is maintained in the output of these architectures, showcasing their color equivariance property.

Experiment 2: Color Invariance Next, we perform an experiment on a dataset that is similar to the version of the MNIST dataset from Experiment 1 to validate the color invariance properties of the modified Faster R-CNN architectures. Exactly as in the original MNIST dataset, the data that is used in this experiment is now comprised of 10 classes, each corresponding to a digit. Several combinations of grayscale colors and the colors red, green, and blue are used for both the digits and the background. The train set consists of 1152 images with each color combination containing 64 images, and the test set consists of 576 images, each color combination having 32 images. Since the hue differences between the colors red, green, and blue match the 120° RGB rotations of the CEConv layers that are integrated into the modified architectures, we expect a significant increase in performance relative to the standard architecture. In this experiment, a batch size of 16 is used.

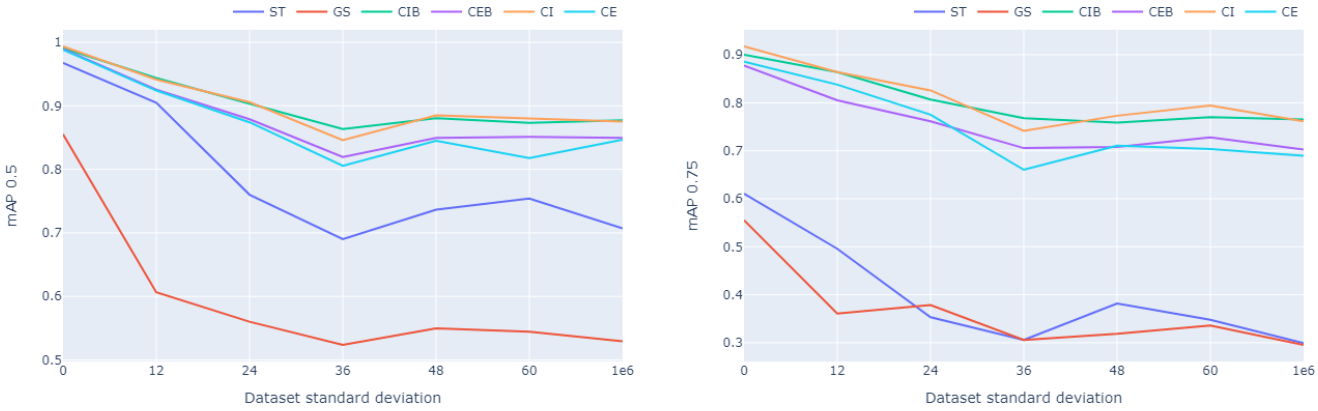


Figure 3: The Mean Average Precision results of Experiment 3. The left figure shows the mAP measurements for an IoU of 0.5 and the right figure shows the mAP measurements for an IoU of 0.75. The standard Faster R-CNN (ST and GS) models show a significantly stronger decrease in efficiency compared to the architectures with the CEConv layers (CIB, CEB, CI, and CE).

Figure 4: The Mean Average Precision results of Experiment 4. The left figure shows the mAP measurements for an IoU of 0.5 and the right figure shows the mAP measurements for an IoU of 0.75. The modified versions (CIB, CEB, CI, and CE) achieve overall better performance than the standard version (ST and GS) across all tested dataset sizes.

model	mAP 0.5	mAP 0.75
ST	69.1 ± 4.2	52.0 ± 4.2
GS	75.0 ± 5.5	57.6 ± 4.7
CIB	$\mathbf{93.5 \pm 0.5}$	$\mathbf{89.6 \pm 1.1}$
CEB	92.8 ± 0.3	89.4 ± 1.2
CI	93.1 ± 0.4	88.5 ± 0.7
CE	92.2 ± 0.4	88.1 ± 1.8

Table 1: The Mean Average Precision results of Experiment 2. The results show that the output of the equivariant architectures is robust to the difference of the colors red, green, and blue.

Table 1 shows that the versions of Faster R-CNN with the CEConv layers are indeed more efficient. As we would expect, the color invariant variants (CIB and CI) achieve the overall highest Mean Average Precision scores. This points to the fact that the output of the architectures is invariant to the difference between the colors red, green, and blue. However, the color equivariant versions (CEB and CE) also perform surprisingly well, indicating that even when the output is transformed the robustness to the color variations maintains.

Experiment 3: Color Robustness For the third experiment, we evaluate the models on a dataset in which a color bias is introduced to compare the robustness to color variation. We generate multiple MNIST variants with each variant containing digits of which the colors provide different amounts of representativeness to their digit. More specifically, each of the 10 classes is assigned a specific hue value that is defined in degrees. Multiple datasets are created that randomly shift the hue values of the colors by a specific amount, derived from a normal distribution. The normal distributions corresponding to the different datasets have standard deviations ranging from 0 to 10^6. In the dataset where the standard deviation is 0, the color of the digits perfectly represents their class. The higher the standard deviation, the less useful the color of the digits is for their classification. The train sets contain 1024 images and the test sets 512. Again, the models are trained with a batch size of 16.

When comparing the performances of the architectures, visualized in Figure 3, we can see that the standard Faster R-CNN (ST and GS) models show a significantly stronger de-

crease in efficiency compared to the architectures with the CEConv layers. This showcases the increase in robustness to color variation that is provided by the CEConv layers. Besides, the color invariant architectures (CIB and CI) perform slightly better than the color equivariant architectures (CEB and CE). This behavior is expected since the color information is not reliable for determining the class in this case. Hence, these results demonstrate that color invariance can be beneficial in scenarios where color is not a reliable feature.

Experiment 4: 2D Chess Boards The fourth experiment aims to investigate whether the replacement of the convolutional layers in the simple Faster R-CNN architecture is also beneficial when dealing with a dataset that contains more complex variations of shapes and colors. For this, we experiment with a dataset consisting of images of 2D chess boards[2]. We take random samples of the entire dataset that are relatively small and compare the implemented versions of Faster R-CNN on each of them. All trained models are tested on a test set consisting of 256 images. Furthermore, all train and test images are modified in such a way that each image only contains a single chess piece. The models are trained using a batch size of 8.

The results of this experiment are shown in Figure 4, where it is visible that the modified versions (CIB, CEB, CI, and CE) achieve overall better performance than the standard version (ST and GS) across all tested dataset sizes. This shows that even when dealing with more complex variations of shapes and colors, the replacement of the convolutional layers can have an interestingly positive influence on the performance of Faster R-CNN.

Experiment 5: PASCAL VOC2012 In the last experiment of this study, we use a small sample of the PASCAL Visual Object Classes 2012 (VOC2012) dataset [24, 25] to compare the models on more real-world images. This dataset is widely used for object detection, segmentation, and image classification. It contains images from 20 different object classes such as various animals, vehicles, and indoor items. The height and width dimensions of the images vary between 100 and 500 pixels, but we resize all of them to 128 x 128. The train dataset used in this experiment consists of 2048 images and the test set contains 512 images. A batch size of 8 is used. The results of this experiment are shown in Table 2.

model	mAP 0.5	mAP 0.75
ST	16.1 ± 3.9	6.0 ± 3.4
GS	14.8 ± 2.8	5.8 ± 2.8
CIB	$\mathbf{17.9 \pm 2.7}$	$\mathbf{7.8 \pm 2.6}$
CEB	17.3 ± 2.3	7.4 ± 2.1
CI	17.8 ± 2.7	7.7 ± 2.7
CE	16.2 ± 3.2	6.1 ± 2.9

Table 2: The Mean Average Precision results of Experiment 5. The achieved performances are relatively low, but we can still see that the modified versions (CIB, CEB, CI, and CE) perform slightly better than the ST and GS versions.

[2]https://www.kaggle.com/datasets/koryakinp/chess-positions

Although the achieved performances are relatively low, we can still see that the modified versions (CIB, CEB, CI, and CE) perform slightly better than the ST and GS versions. The low mAP scores follow most likely from the fact that the backbone is relatively small and a low amount of small-sized images is used as training data. Because of these low performances, we cannot make reliable conclusions from this experiment about the translation of the color robustness improvements to real-world data. Despite all of this, the experiment does show that it will be interesting to compare color equivariant versions of Faster R-CNN with more complex backbones on real-world datasets.

5 Conclusion

To conclude, this work presents a comprehensive investigation of the integration of color equivariance and invariance into the Faster R-CNN object detection framework. With the conducted experiments, we demonstrate that the replacement of the standard convolutional layers in Faster R-CNN with color equivariant and invariant operations leads to significant improvements in performance when dealing with datasets containing diverse color distributions. The color equivariant and invariant models (CEB, CE, CIB, and CI) showed improved performance in comparison to the standard Faster R-CNN version on data with color variations and imbalances. These results reaffirm that color variations and their inferences play an important role in the field of object detection, which highlights the significance of the introduction of the color equivariance and invariance properties.

Limitations and Future Work Although the results of the toy experiments show a significant improvement in performance when replacing the standard convolutional layers in Faster R-CNN with color equivariant layers, this study conducts only a limited amount of research on real-world data. This is because of the use of a simple CNN for the backbone, as well as the long duration of the training processes combined with time constraints. However, this study does show that it would be interesting to conduct further research on the influence of the modifications when using more complex, state-of-the-art backbone CNNs for Faster R-CNN. Furthermore, this research performs an ablation study (see the appendix) that consists of only a single toy experiment in which the influence of adding more parameters to the CIB and CEB models on their learning curve is measured. However, a more extensive ablation study that also includes experimenting with the amount of parameters by varying the amount of rotations of the CEConv layers, which is fixed to 3 in this research, would provide more insight into this.

References

[1] I. S. Krizhevsky, Alex and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[2] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[3] Z. Liang, G. Zhang, J. Huang, and Q. Hu, "Faster r-cnn based glaucoma detection using fundus images," *Future Generation Computer Systems*, vol. 90, pp. 62–71, 2019.

[4] Y. Hou, Z. Zhang, and Q. Wu, "Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining," *Remote Sensing*, vol. 9, no. 2, p. 173, 2019.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, Cham, 2016, pp. 21–37.

[6] N. K. Diego Marcos, Michele Volpi and D. Tuia, "Rotation equivariant vector field networks," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5048–5057, 2017.

[7] M. Weiler and G. Cesa, "General e(2)-equivariant steerable cnns," *Advances in Neural Information Processing Systems*, 2018.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] T. S. Cohen and M. Welling, "Group equivariant convolutional networks," *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2990–2999, 2016.

[10] O. S. Kayhan and J. C. van Gemert, "Evaluating context for deep object detectors," *Computer Vision Lab, Delft University of Technology*, 2022.

[11] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[12] J. Geusebroek, G. Burghouts, and A. Smeulders, "The amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.

[13] A. Khosla, B. An, J. Lim, and A. Torralba, "Looking beyond the visible spectrum," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[14] A. Gijsenij and T. Gevers, "Color constancy using natural image statistics," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[15] K. Barnard, L. Martin, A. Coath, and B. Funt, "A comparison of computational color constancy algorithms. i: Methodology and experiments with synthesized data," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 972–984, 2002.

[16] B. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant cnns for digital pathology," *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, vol. 11071, pp. 210–218, 2018.

[17] S. Ravanbakhsh, F. Lanusse, R. Mandelbaum, M. Schneider, and B. Poczos, "Enabling dark energy science with deep generative models of galaxy images," *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[18] R. Kondor and S. Trivedi, "On the generalization of equivariance and convolution in neural networks to the action of compact groups," *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 2747–2755, 2018.

[19] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of the International Conference on Learning Representations*, 2015.

[21] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, 2017.

[22] L. Bottou, "Online algorithms and stochastic approximations," *D. Saad (Ed.), Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK*, 1998.

[23] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.

[24] M. Everingham, L. V. 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.

[25] M. Everingham, S. A. Eslami, L. V. Gool, C. K. 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.

3

Methods

This chapter gives comprehensive technical explanations of the methods that are used in the scientific article from Chapter 2.

3.1. Computer Vision by Deep Learning

Computer vision is about teaching machines to interpret visual data, similar to how humans are able to perceive and understand images and videos [5]. Various techniques that range from traditional image processing techniques to more advanced machine learning algorithms are being used in the field of computer vision. In recent years, however, deep learning has revolutionized the approach to performing computer vision tasks.

Deep learning is a specific subset within the area of machine learning [6]. It uses algorithms, known as neural networks, that are comprised of layers that each perform a mathematical operation. The "deep" in deep learning refers to the high number of layers in these architectures. Traditional neural networks contain only a couple of layers, while deep networks can have hundreds. Deep learning models learn directly from data and there is no need for providing them with instructions. In general, the more training data is given to these networks, the higher their performance on computer vision tasks.

A neural network learns by updating the learnable weights of each of its mathematical operations in such a way that the loss in for example a computer vision task is minimized. This loss can be defined as a quantification of the difference between the predictions of the network and the actual ground truth targets. The exact updates are calculated by during the training process, in which the network is shown a large amount of examples. Because this is a thesis in the field of computer vision, these examples are represented by images in this work. Typically, the more example data that is used for training a network, the better the network is able to perform the task.

For the calculation of the weight updates, a loss function is used. This loss function can for example be the Mean Squared Error (MSE) or the Cross-Entropy Loss, and it essentially quantifies the error of the neural network that is being trained. By performing backpropagation, the gradients of the loss function with respect to each weight are calculated such that each weight can be updated accordingly.

3.2. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a kind of deep learning models that are designed for tasks with grid-like data, such as images [7]. Essentially, an image is a grid of pixels. CNNs can learn patterns within the structures of these images, from simpler patterns such as edges and textures in the initial layers of the networks, to more complex patterns in the deeper layers. This layered approach results in

the fact that a CNN is able to automatically learn the most important features in an image, without having to provide the network with instructions, making them perform accurately in image recognition tasks such as object detection.

3.2.1. Convolutions in CNNs

An important aspect of CNNs is the convolution operation, which combines two functions together [8]. In the context of CNNs, these functions are an input image or feature map, and a kernel that serves as a filter. A kernel is a matrix, usually of smaller height and width, and its depth corresponds to the number of channels of the input, which is 3 for an RGB image. The convolution operation can be described as an element-wise multiplication of the values of the kernel with the image pixel values, followed by a summation of the resulting values. This operation is performed for every spatial position of the input, resulting in an output feature map that can have a different amount of channels than the input image. It can be expressed with the following formula:

$$[f \star \psi](x) = \sum_{y \in \mathbb{Z}^2} \sum_{c=1}^{C^l} f_c(y)\psi_c(y - x). \tag{3.1}$$

Here is a breakdown of the components of the formula:

- $[f \star \psi](x)$ denotes the resulting feature map of the convolution of the function f with the kernel ψ at a specific spatial position x. For the first layer of a CNN, f is an input image and x is a pixel from the image in the context of this thesis. The result $[f \star \psi](x)$ can be seen as a feature map that captures specific patterns from the original image and can be passed as the input function for the next layer within the network.
- The term $y \in \mathbb{Z}^2$ means that the operation is performed over a 2D grid. In the context of images, this represents the fact that the operation iterates over all pixel positions in a two-dimensional space, where y stands for a particular position.
- c denotes an index for channels. In a typical RGB image, there are 3 channels, namely for the colors red, green, and blue. However, as images are processed through a CNN, the feature maps that come out of its layers can have many more channels. The value C^l denotes the total number of channels within the feature map of layer l.
- $f_c(y)$ represents the value of the input function f at position y for channel c.
- ψ_c is the convolutional kernel for channel c. In a CNN, these kernels are the learnable parameters, meaning that during the training process, these values are adjusted in such a way that the performance of the whole convolutional neural network is maximized for the specific computer vision task that it is used for.

Essentially, the convolution operation involves sliding the kernel over the image and computing the sum of element-wise products at each pixel, see Figure 3.1.

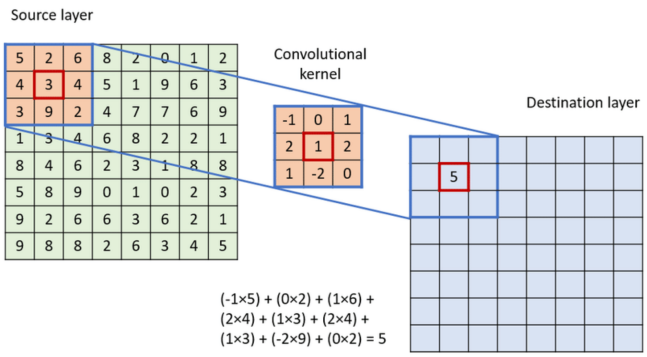


Figure 3.1: A schematic visualization of a convolution operation [9]. Using a sliding window approach, the values for each spatial location of the resulting feature map are derived by calculating a sum of products of the values of the input and kernel functions.

3.2.2. Subsampling by Pooling

Subsampling has as its main purpose that it reduces the spatial dimensions of the feature maps, which also reduces the computational complexity so that the network can focus on the most important features within the feature maps. Two of the most commonly used strategies for subsampling are max pooling and average pooling. In max pooling, a window slides over the input feature map, similar to a convolution operation. For each position of the window on the feature map, the maximum value within the window is selected for the resulting pooled output. With average pooling, an average of the values within the window is taken instead of the maximum value. While average pooling is sometimes utilized, max pooling is more commonly used since it retains the most important features, often leading to better efficiency. The max pooling and average pooling operations can be mathematically expressed as:

$$p_{\max}(x) = \max_{y \in W_x} f_c(y), \tag{3.2}$$

$$p_{\text{avg}}(x) = \frac{1}{|W_x|} \sum_{y \in W_x} f_c(y), \tag{3.3}$$

where:

- $p_{\max}(x)$ is the resulting pooled value at spatial position x after applying max pooling.
- $p_{\text{avg}}(x)$ is the resulting pooled value at spatial position x after applying average pooling.
- W_x denotes the set of spatial positions in the pooling window when it is centered at x.
- $f_c(y)$ represents the value of the input feature map f at spatial position y for channel c.

Both operations are visualized in Figure 3.2. The height and width dimensions are halved since the size of the pooling window is 2 x 2 for both pooling operations.

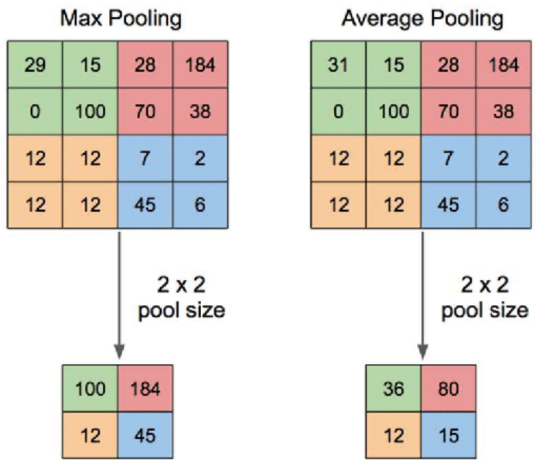


Figure 3.2: A schematic visualization of the max pooling and average pooling operations [10]. The height and width dimensions are halved since the size of the pooling window is 2 x 2 for both operations.

3.2.3. A Visualization of a CNN

Figure 3.3 shows a schematic example of a simple CNN that contains only three convolutional layers and two max pooling layers. The max pooling layers serve as subsampling layers by using a 2 x 2 pooling window to halve the height and width dimensions of their input feature maps. Since there are two max pooling layers, the output feature map has height and width dimensions that are only 25% the size of the height and width dimensions of the original input image. The figure also shows how the convolutional layers are able to change the number of channels within the feature map. This exact example network is used in this thesis as the backbone CNN for the Faster R-CNN object detection model, as will be explained in the next section.

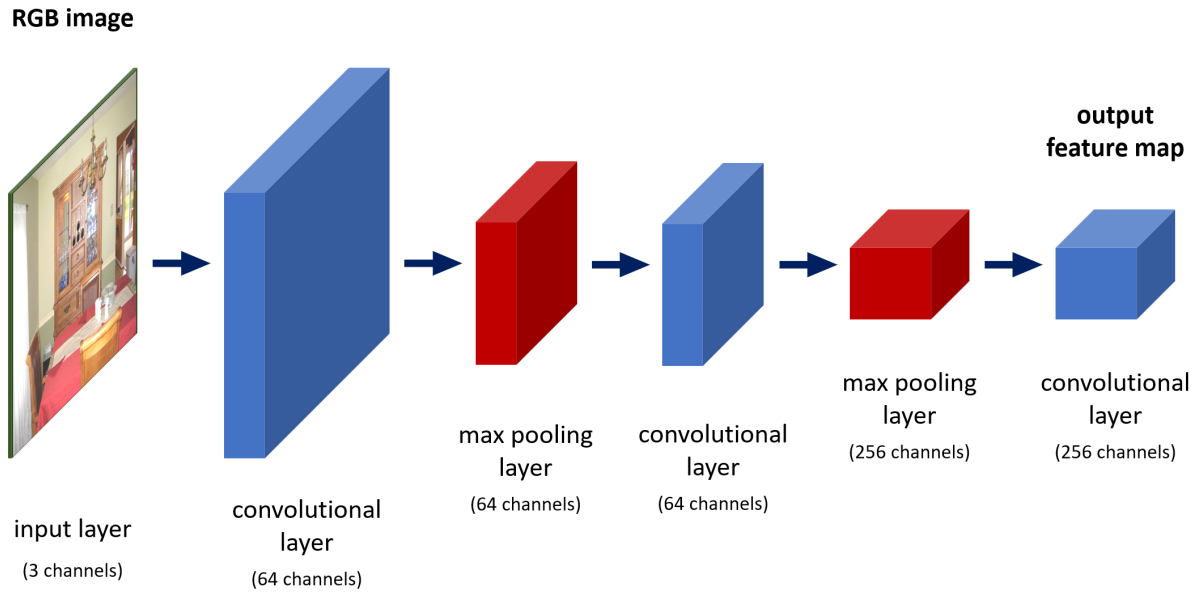


Figure 3.3: A schematic example of a simple CNN that contains only three convolutional layers and two max pooling layers. The blue blocks represent the output feature maps of the convolutional layers, and the red blocks those of the max pooling layers. For each layer, the amount of channels of its output feature map is also specified. This exact example network is used in this thesis as the backbone CNN for the Faster R-CNN object detection model.

3.3. Object Detection with Faster R-CNN

Object detection is a computer vision task that tries to locate objects within an image, and also to identify them. Faster R-CNN (Region-based Convolutional Neural Network) is one of the most notable examples of object detection architectures [3]. It has a backbone CNN that produces feature maps that are processed by the Region Proposal Network (RPN) to predict object bounding boxes, and by the detection network to classify the objects within those bounding boxes. Figure 3.4 visualizes an overview of the Faster R-CNN architecture.

3.3.1. Backbone

The backbone CNN extracts features from the input image, resulting in a feature map. The CNN architecture that is depicted in 3.3 is the exact architecture that is used for the backbone in the models that are studied in this thesis. It is composed of three convolutional layers and two max pooling layers that each have a 2 x 2 pooling window. This results in the fact that the output feature map of this backbone CNN has height and width dimensions that are 25% the height and width dimensions of the original input image. This feature map is then processed by the Region Proposal Network (RPN) and the classification network.

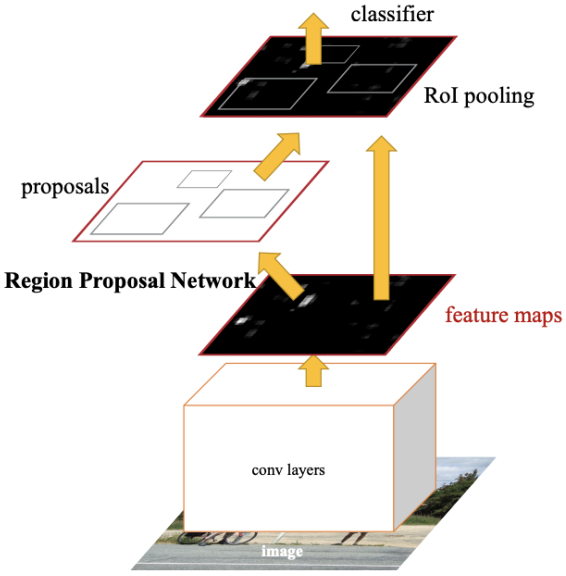


Figure 3.4: An overview of the Faster R-CNN architecture. The input image is processed by the backbone network to produce a feature map that is used by the RPN to propose object regions. These proposals are classified and refined by the final layers of the network [3].

3.3.2. Region Proposal Network (RPN)

The RPN is a fully convolutional network that generates proposals for object regions. The feature map that is the output of the backbone is used as the input for the RPN. With a sliding window approach and a given set of anchors, the RPN predicts the objectness scores for each anchor, which is a quantification of the likelihood that a certain anchor box contains an object, and therefore not only background. Also, the RPN predicts bounding box adjustments, which are essentially refinements for each anchor box coordinates to more precisely fit around the targeted object in the image. The loss function for RPN consists of two parts:

- **Objectness Loss** ($\mathcal{L}_{\text{obj}}$): This loss measures the differences between the predicted objectness scores (p) and the ground-truth labels (p^*). We use the logistic loss [11], which is the most common approach for computing this metric.
- **Regression Loss** ($\mathcal{L}_{\text{reg}}$): This loss measures the difference between the predicted bounding box adjustments (t) and the ground-truth adjustments (t^*). This metric is typically computed using the smooth L1 loss [12], which we also utilize in our approach.

The overall loss function for the RPN is:

$$\begin{aligned}\mathcal{L}_{\text{rpn}}(\{p_i\},\{t_i\}) = &\frac{1}{N_{\text{obj}}}\sum_i \mathcal{L}_{\text{obj}}(p_i, p_i^*) \\ &+ \lambda \frac{1}{N_{\text{reg}}}\sum_i p_i^* \mathcal{L}_{\text{reg}}(t_i, t_i^*)\end{aligned} \tag{3.4}$$

where i is the index of an anchor box, p_i and t_i are the predicted objectness score and bounding box adjustments, and p_i^* and t_i^* are the corresponding ground-truth values. λ is a weighting factor that balances the contributions of both losses to the total RPN loss.

3.3.3. Region of Interest (RoI) Pooling and Detection Network

The region proposals that are generated by the RPN are fed into the RoI pooling layer. This layer crops these regions from the feature map produced by the backbone CNN. Subsequently, these crops are fed into the detection network, which consists of a series of fully connected layers for classification and further adjustment of the bounding boxes. The loss function for the detection network consists of:

- **Classification Loss** ($\mathcal{L}_{\text{cls}}$): This part of the loss measures the discrepancy between the predicted class scores (c) and the ground-truth class labels (c^*). This metric is computed using softmax loss.
- **Bounding Box Loss** ($\mathcal{L}_{\text{box}}$): This metric compares the predicted bounding box coordinates (b) to the ground-truth coordinates (b^*). Similar to the RPN, we compute this metric using the smooth L1 loss.

The overall loss for the detection network is calculated as follows:

$$\begin{aligned}\mathcal{L}_{\text{det}}(\{c_i\},\{b_i\}) = \frac{1}{N_{\text{cls}}}\sum_i \mathcal{L}_{\text{cls}}(c_i, c_i^*) \\ + \beta\frac{1}{N_{\text{box}}}\sum_i c_i^*\mathcal{L}_{\text{box}}(b_i, b_i^*)\end{aligned} \tag{3.5}$$

where i is the index of a bounding box prediction, c_k and b_k are the predicted class score and bounding box coordinates, and c_k^* and b_k^* are the corresponding ground-truth values. β is a weighting factor that balances the contributions of both losses to the total detection network loss.

3.3.4. Total Loss Function

The total loss function for Faster R-CNN combines the RPN and detection network losses and can be represented as:

$$\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{det}} \tag{3.6}$$

By combining both losses, the process of backpropagation ensures that the weights of both the RPN and the detection network are updated such that the model is able to return more accurate bounding boxes and class scores. It is important to understand the structure of the Faster R-CNN object detection architecture, as well as its loss function, to be able to reason about certain results that follow from the experiments of this work.

3.4. Group Equivariant Convolutional Neural Networks (G-CNNs)

Group Equivariant Convolutional Neural Networks (G-CNNs) are CNNs that are modified in such a way that they take specific symmetries within the input data into consideration [4]. This can make them more robust to specific variations or imbalances in datasets. This section provides a comprehensive description of how G-CNNs use the theory of symmetry groups to achieve equivariance to certain transformations in the data. The concept of G-CNNs is also applied in this research by designing versions of the Faster R-CNN object detection model that are equivariant to color transformations that change the hue values of an image.

3.4.1. Symmetry Groups

A symmetry of an object is a transformation that is applied on the object without changing it, for example rotating the object by 90°. If two symmetry transformations have the property that combining both transformations results in another symmetry transformation, and if the inverse of both transformations also results in symmetries, the two transformations are said to be in the same symmetry group. One

example of a symmetry group is the group $p4$, see Figure 3.5. This group consists of all possible combinations of spatial translations and rotations by 90° around any spatial center of rotation within a feature map. It can be parameterized by three integers r, u, and v:

$$g(r,u,v) = \begin{bmatrix} \cos\left(\frac{r\pi}{2}\right) & -\sin\left(\frac{r\pi}{2}\right) & u \\ \sin\left(\frac{r\pi}{2}\right) & \cos\left(\frac{r\pi}{2}\right) & v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

where:

- r captures the rotational symmetries of the square. r can be either 0, 1, 2, or 3, specifying no rotation, 90° rotation, 180° rotation, and 270° rotation, respectively. The rotational transformation is captured in the matrix by the $\cos\left(\frac{r\pi}{2}\right)$ and $\sin\left(\frac{r\pi}{2}\right)$ terms.
- u and v encode the translational symmetries on a discrete plane. The parameters take values in the integer lattice $\mathbb{Z}^2$, representing translations in the x and y directions, respectively. In the matrix representation, u and v appear as translational components.

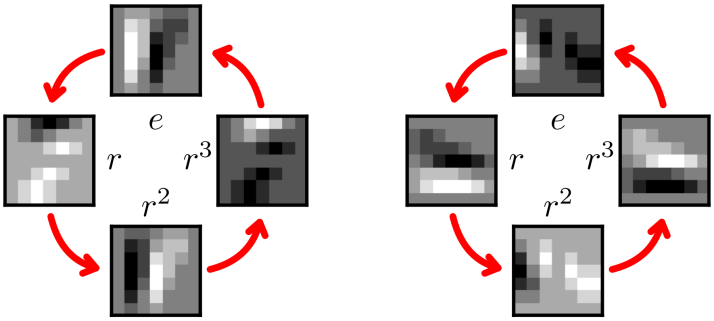


Figure 3.5: A $p4$ feature map and its rotation by r [4].

3.4.2. Equivariance and Invariance

Equivariance and invariance are two concepts that can be integrated into deep learning networks with the goal of making them consider symmetry groups. A convolutional neural network layer Φ is said to be equivariant to a symmetry group G if the transformations in the group act similarly on the input and the resulting feature mapping that is the output of the layer. In mathematical terms, equivariance can be represented as:

$$\Phi(T_g x) = T'_g \Phi(x), \quad \forall g \in G, \tag{3.8}$$

where T_g and T'_g are the transformation operators corresponding to a group action g on the input and feature space, respectively.

A special instance of the concept of equivariance is when T_g and T'_g are identical, such as in the case of translation equivariance. If shifting the input results in an equally shifted feature map, the equivariance condition simplifies to:

$$\Phi(T_g x) = T_g \Phi(x), \quad \forall g \in G. \tag{3.9}$$

Another special instance of equivariance is when the transformation on the input leaves the output feature map unchanged, implying that T'_g is the identity mapping. In this case, the property is known as invariance:

$$\Phi(T_g x) = \Phi(x), \quad \forall g \in G. \tag{3.10}$$

3.4.3. Integrating Color Equivariance into Faster R-CNN

In this thesis, several versions of the Faster R-CNN object detection architecture with the simple backbone from Figure 3.3 are implemented in which color equivariance is integrated. This work defines color equivariance as equivariance to hue shifts, hue being a component of the Hue-Saturation-Value (HSV) color space which provides an alternative representation of colors to the standard Red-Green-Blue (RGB) color space. The RGB space represents colors as combinations of the colors red, green, and blue, and the HSV space describes colors in terms of tint, shade, and tone. The HSV color space consists of three components:

- **Hue (H):** Representing the type of color, such as red or green, and is measured in degrees from 0° to 360°.
- **Saturation (S):** Specifying the vibrancy of the color, ranging from 0 (gray) to 1 (pure color).
- **Value (V):** Denoting the brightness of the color, with 0 being black and 1 being the full brightness of the color.

A color in the RGB space can be converted to HSV using the following formulas:

$$H = \begin{cases} \frac{60(G-B)/(V-m)}{mod\ 6} & \text{if } V = R \\ \frac{60(B-R)/(V-m)+2}{mod\ 6} & \text{if } V = G \\ \frac{60(R-G)/(V-m)+4}{mod\ 6} & \text{if } V = B \end{cases} \tag{3.11}$$

$$S = \begin{cases} 0 & \text{if } V = 0 \\ \frac{V-m}{V} & \text{otherwise} \end{cases} \tag{3.12}$$

$$V = \max(R, G, B) \tag{3.13}$$

where $m = \min(R, G, B)$. When projecting the HSV representation onto the RGB color space, the same hue shift becomes a rotation along the $[1, 1, 1]$ diagonal vector. In this thesis, we define the symmetry group H_n of multiples of $360/n$-degree rotations about the $[1, 1, 1]$ diagonal vector in R^3 space. We can parameterize H in terms of integers k, n as:

$$H_n(k) = \begin{bmatrix} \cos\left(\frac{2k\pi}{n}\right) + a & a - b & a + b \\ a + b & \cos\left(\frac{2k\pi}{n}\right) + a & a - b \\ a - b & a + b & \cos\left(\frac{2k\pi}{n}\right) + a \end{bmatrix} \tag{3.14}$$

with n the total number of discrete rotations in the group, k the rotation,

$$a = \frac{1}{3} - \frac{1}{3}\cos\left(\frac{2k\pi}{n}\right) \tag{3.15}$$

and

$$b = \sqrt{\frac{1}{3}}\sin\left(\frac{2k\pi}{n}\right). \tag{3.16}$$

A group action in this group is a matrix multiplication that transforms RGB pixel values. In the color equivariant versions of Faster R-CNN that are implemented in this research, the convolutional layers are replaced by color equivariant convolutional layers that are equivariant to this transformation group. Figure 3.6 shows examples of hue shifts on images to which the modified Faster R-CNN versions are equivariant.

(a) (b)

Figure 3.6: Examples of hue shifts on images with $n = 3$ (a) and $n = 4$ (b).

4

Datasets

The experiments of this thesis, described in the scientific article in Chapter 2, are conducted on several kinds of image datasets. This chapter describes the details of these datasets and how they are modified to generate different versions. The images of the datasets are also visualized in this chapter.

4.1. MNIST Variants

The MNIST (Modified National Institute of Standards and Technology) dataset [13] is a large database of images containing handwritten digits, see Figure 4.1. It is commonly used for training image processing algorithms. It consists of 60,000 train images and 10,000 test images, all distributed over 10 classes. Each image in the dataset is a 28 x 28 pixel grayscale image of a single digit labeled as a number from 0 to 9. Because of the small size and simplicity of the images, the MNIST dataset is widely used for evaluating and comparing model architectures in computer vision.

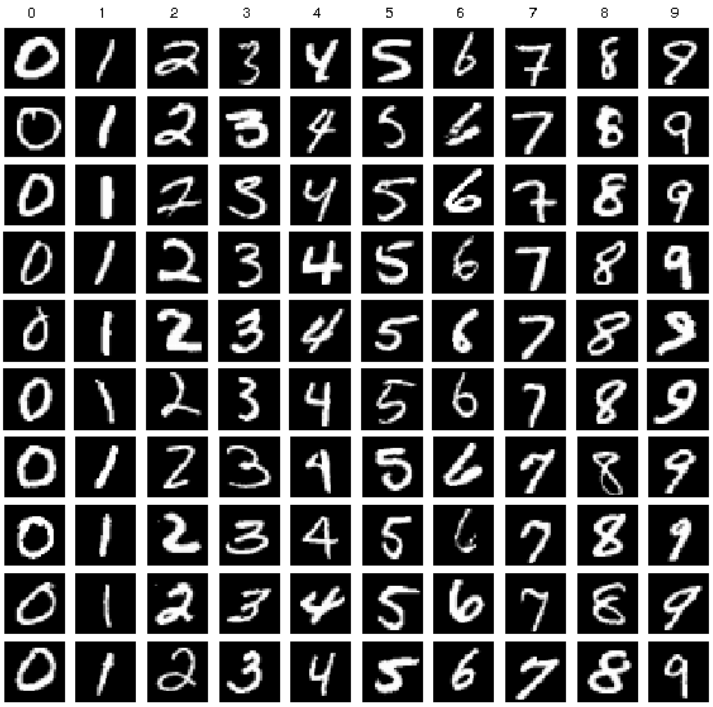


Figure 4.1: Example images from the MNIST dataset with the corresponding class labels at the top of the figure [14].

Experiment 1

For the first experiment of this work, a variant of the MNIST dataset is generated that contains images of size 64 x 64 with a randomly scaled digit on a random position within the image frame, see Figure 4.2. The scale factor is drawn from a uniform distribution and is between 0.5 and 2.0. The random position is chosen in such a way that the full MNIST image is contained within the resulting image frame. The digit is colored either red, green, or blue, and each possible combination of the digits and the colors is assigned a class label, implying that there are 30 classes in total. Each image has a gray background with an intensity of 0.33, and noise is added with a standard deviation of 0.1. The train set consists of 1514 images. In this train set, a class imbalance is introduced by drawing the number of samples per class from a power law distribution. This results in different class frequencies, see Table 4.1. On the other hand, the test set consists of 960 images where each class contains 32 images.

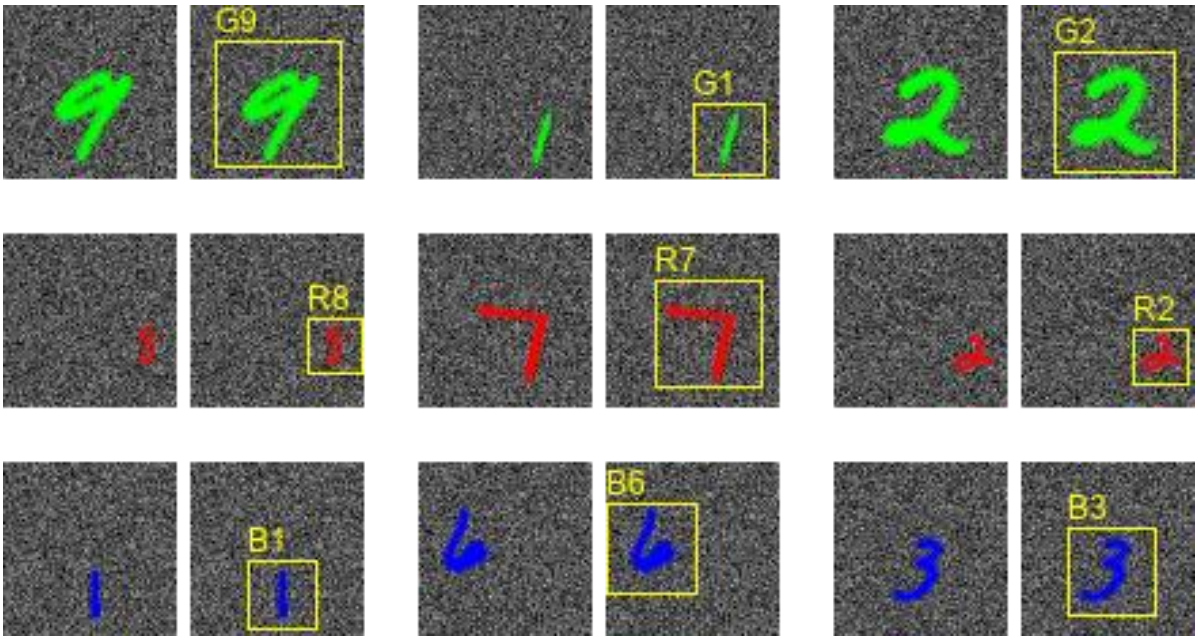


Figure 4.2: Example images from the MNIST variant that is used in Experiment 1. For each image, the corresponding bounding box and class label are also visualized. G9 denotes a green 9, B1 denotes a blue 1, R2 denotes a red 2, etc.

G3	G7	B1	B7	B2	R7	B4	B9	G1	G9	R2	R8	G6	G0	B3	B0	R9	R3	R1	R5	G4	G5	G2	R6	R4	B6	B5	G8	B8	R0
128	118	112	107	98	98	92	89	85	76	72	69	65	57	55	36	24	24	24	24	13	12	10	9	8	3	3	1	1	1

Table 4.1: Class labels and corresponding amounts of samples of the train dataset used in Experiment 1. G9 denotes a green 9, B1 denotes a blue 1, R2 denotes a red 2, etc.

Experiment 2

The second experiment is conducted with a generated variant of MNIST where the images are 64 x 64 pixels and the digits are randomly scaled and positioned in a similar way as in Experiment 1. However, the 10 class labels of the original MNIST dataset are maintained. 18 different combinations of grayscale colors and the colors red, green, and blue are used for both the digits and the background, see Figure 4.3. The backgrounds do not contain any noise. The train set consists of 1152 images with each color combination having 64 images, and the test set consists of 576 images with each color combination having 32 images.

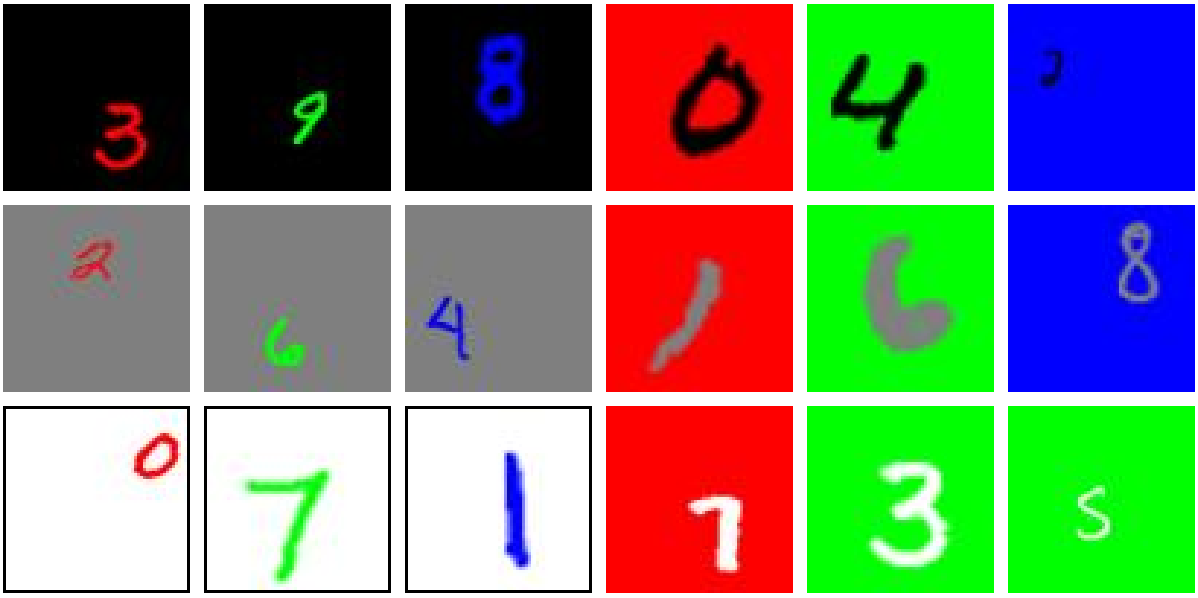
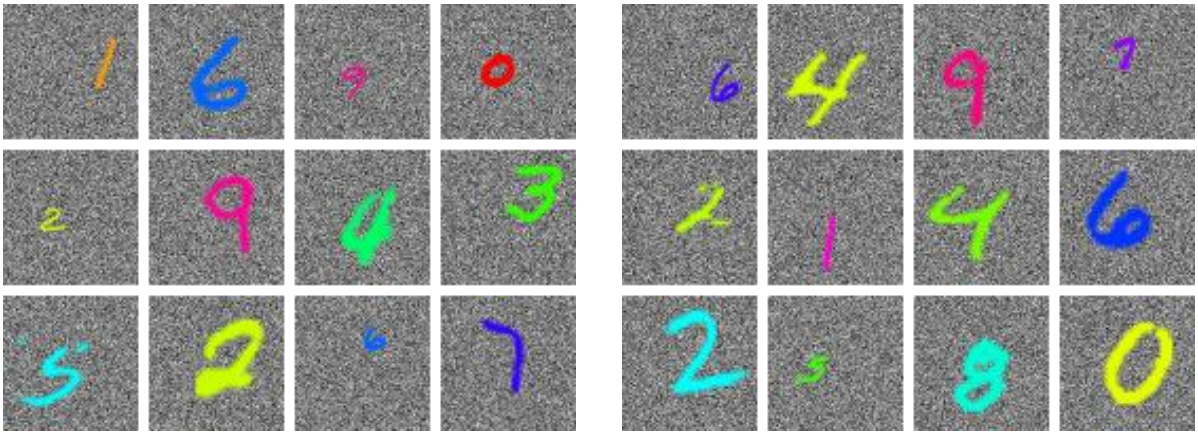


Figure 4.3: Example images from the MNIST variant that is used in Experiment 2. For each of the 18 color combinations for the digits and backgrounds, a single example image is shown. In this figure, a black outline is added to the images with a white background for visualization purposes. However, the dataset does not have any outlines around images.

Experiment 3

The third experiment is the last experiment of this work that is conducted on MNIST digits. Similarly to Experiment 1, the images are of size 64 x 64 and the digits are randomly scaled and positioned. In this experiment, however, the 10 class labels of the original MNIST dataset are maintained. 7 variants are generated with each variant containing digits of which the colors provide different amounts of representativeness to their digit. More specifically, each of the 10 classes is assigned a specific hue value that is defined in degrees. Each variant randomly shifts the hue values of the colors by a specific amount, derived from a normal distribution. The normal distributions corresponding to the 7 datasets have the standard deviations 0, 12, 24, 36, 48, 60, and 10^6. In the dataset where the standard deviation is 0, the color of the digits perfectly represents their class. The higher the standard deviation, the less useful the color of the digits is for their classification, see Figure 4.4. Again, the backgrounds of the images are grayscale, now with an intensity of 0.5, and noisy, with a standard deviation of 0.1. The train sets contain 1024 images and the test sets 512.



(a) **(b)**

Figure 4.4: Example images from two MNIST variants that are used in Experiment 3. (a) shows the variant where the standard deviation of the hue shifts is 0, and (b) shows the variant where it is 32.

4.2. 2D Chess Boards

For Experiment 4 of this thesis, a dataset consisting of images of 2D chess boards[1] is utilized. The 80,000 train and 20,000 test images of this dataset are of size 400 x 400 and contain randomly generated 2D chess positions. The chess pieces represent the objects that need to be detected and are labeled one of 6 possibilities; pawn, bishop, knight, rook, queen, or king. 28 different styles of chess boards and 32 styles of chess pieces were used. The number of chess pieces in the images varies between 5 and 15, but in this study, the amount of pieces is fixed to a single piece per image. This is done by modifying the original images by replacing random pieces with random empty squares.

Experiment 4

In this experiment, the 2D chess board images are modified in such a way that each image has a single chess piece on the board, see Figure 4.5.

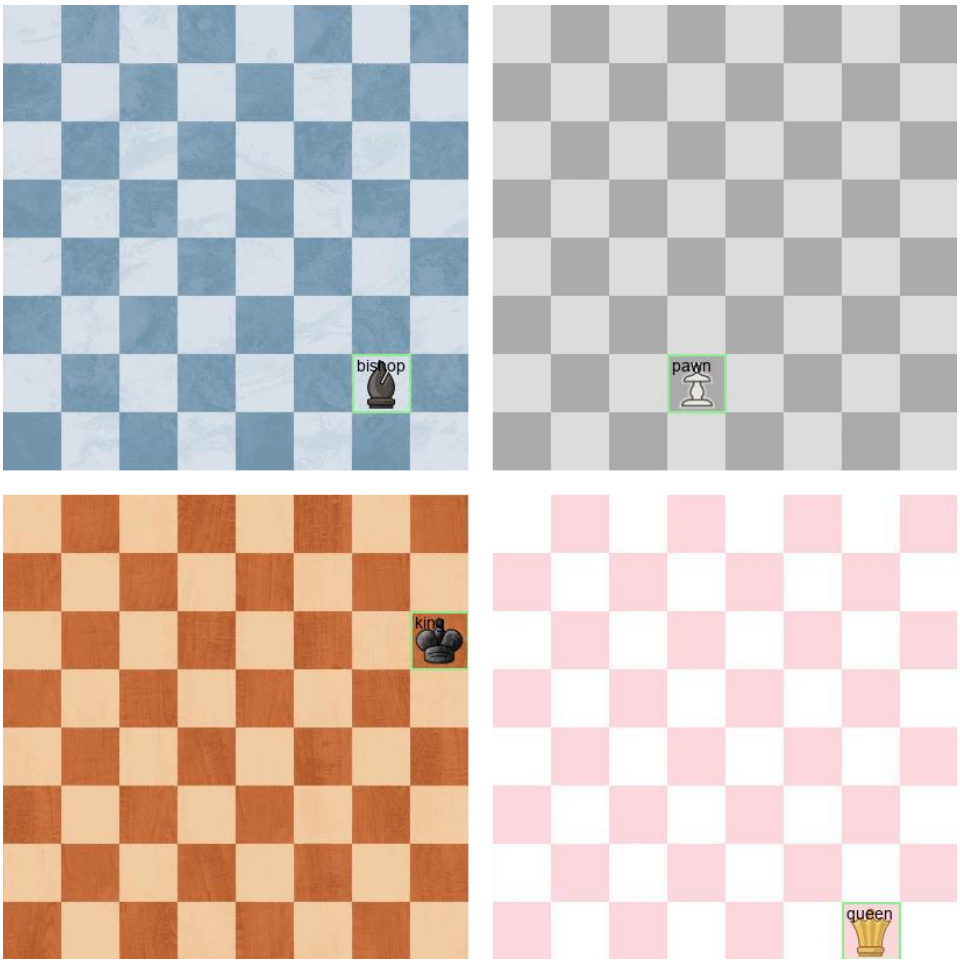


Figure 4.5: Example images from the 2D chess board dataset that is used in Experiment 4. The bounding boxes and corresponding labels of the pieces are also visualized in this figure.

[1] https://www.kaggle.com/datasets/koryakinp/chess-positions

4.3. PASCAL VOC2012

The PASCAL Visual Object Classes Challenge 2012 (VOC2012)[2] is a widely used dataset for object detection, segmentation, and image classification. It contains images from 20 different object classes such as various animals, vehicles, and indoor items, see Figure 4.6. More specifically, the class labels are person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, and tv/monitor. The height and width dimensions of the images vary between 100 and 500 pixels.

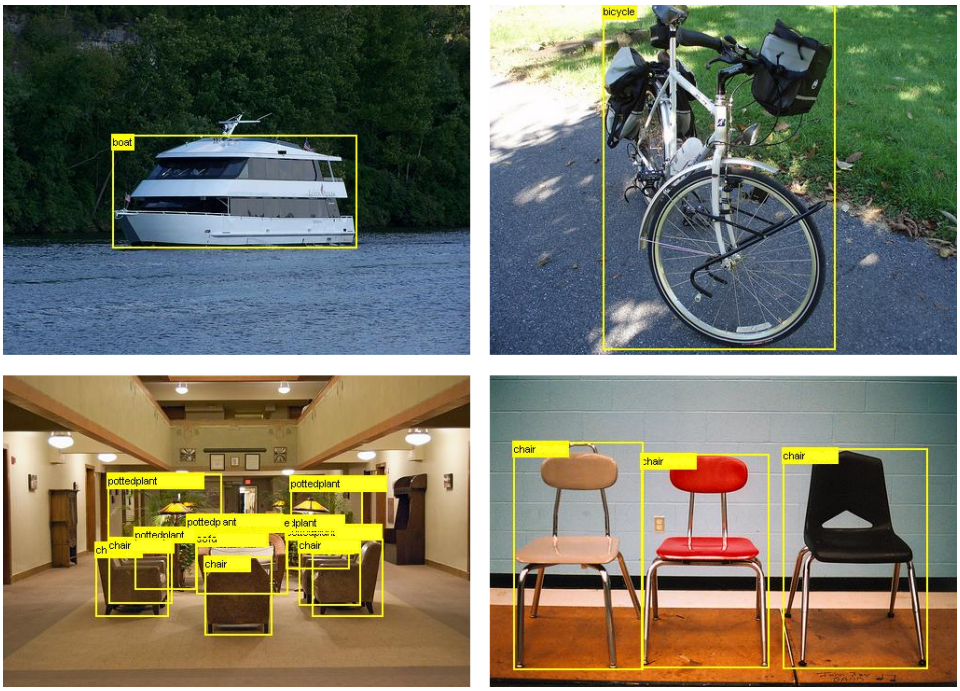


Figure 4.6: Example images from the PASCAL VOC2012 dataset that is used in Experiment 6. The bounding boxes and corresponding labels of the objects are also visualized in this figure.

Experiment 5

For Experiment 5, a random subsample of the original PASCAL VOC2012 dataset is taken. The train dataset used in this experiment consists of 2048 images and the test set contains 512 images. All images are resized to 128 x 128.

[2] `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html`

Bibliography

[1] K. Zhang et al. “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks”. In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503.

[2] H. Li and P. Shen. “A Robust and Scalable Approach to Automatic License Plate Detection”. In: *Pattern Recognition* 52 (2016), pp. 33–43.

[3] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems* (2015), pp. 91–99.

[4] Taco S. Cohen and Max Welling. “Group Equivariant Convolutional Networks”. In: *Proceedings of The 33rd International Conference on Machine Learning* (2016), pp. 2990–2999.

[5] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[7] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105.

[8] Adam Gibson and Josh Patterson. *Deep Learning: A Practitioner’s Approach*. O’Reilly Media, 2017.

[9] Damian Podareanu et al. *Best Practice Guide - Deep Learning*. 2019.

[10] Muhamad Yani, Budhi Irawan, and Casi Setianingsih. “Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail”. In: *Journal of Physics: Conference Series* 1201 (2019).

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2009.

[12] Ross Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448.

[13] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[14] Seung-Hwan Lim, Steven Young, and Robert Patton. *An analysis of image storage systems for scalable training of deep neural networks*. Oak Ridge National Laboratory, 2016.

Appendix

Ablation Study

In the experiments of this thesis, a simple CNN is used with only three convolutional layers and two downsampling operations as the backbone for the studied Faster R-CNN versions, see Figure 3.3. The replacement of a traditional convolutional layer with a CEConv results in a more computationally expensive operation since the amount of learnable parameters is increased. Similarly, the complexity of the Faster R-CNN object detection architecture increases when integrating the color equivariant convolutions. For the ST and GS versions of the model, all convolutional layers have 256 output channels, whereas the first two convolutional layers of the other versions (CIB, CEB, CI, and CE) only have 64 output channels. This is implemented this way to account for the increase in complexity that is caused by the modifications. The choice of these amounts of output channels, however, leads to the fact that the CIB and CEB models have significantly fewer parameters than the ST model. We perform an ablation study in which we compare the CIB and CEB models to more complex versions of themselves and we label these versions as CIBm and CEBm, respectively. This extra complexity is achieved by simply doubling the output channels of both the first layer and the second layer of the CNN backbone from 64 to 128. Table 2 shows a comparison of the number of parameters for each of the Faster R-CNN versions.

model	number of parameters
ST	15845064
GS	15845064
CIB	15212040
CIBm	15988296
CEB	15408904
CEBm	16185160
CI	15802376
CE	16196104

Table 2: A comparison of the number of parameters for each of the Faster R-CNN versions.

We perform an ablation study on the training processes of the CIB, CIBm, CEB, and CEBm models, and we compare the test mAP curves from CIB and CEB with those from CIBm and CEBm, respectively. We use the MNIST dataset variant from Experiment 2 of this thesis for this ablation study. The models are trained for 200 epochs using the SGD optimizer with a learning rate of 0.01, a momentum of 0.9, a weight decay of 0.0005, and a batch size of 16. Each model is trained 5 times with different random initializations. The results for the comparison of CIB and CIBm are visible in Figure 7, and the results for the comparison of CEB and CEBm are visible in Figure 8. From the mAP curves, we can see that there is no significant difference in performance when adding complexity to the backbone of the models.

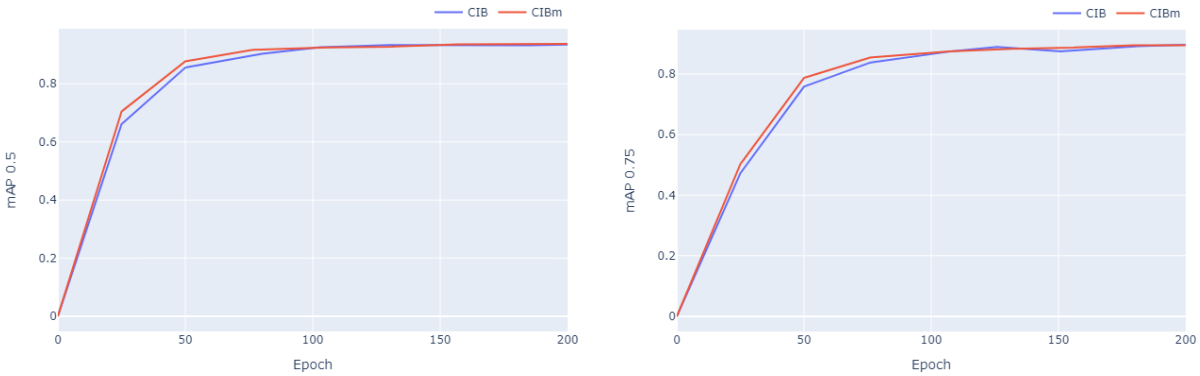


Figure 7: The Mean Average Precision results of the comparison of the CIB and CIBm models. The left figure shows the mAP measurements for an IoU of 0.5 and the right figure shows the mAP measurements for an IoU of 0.75. There is no significant difference in performance.

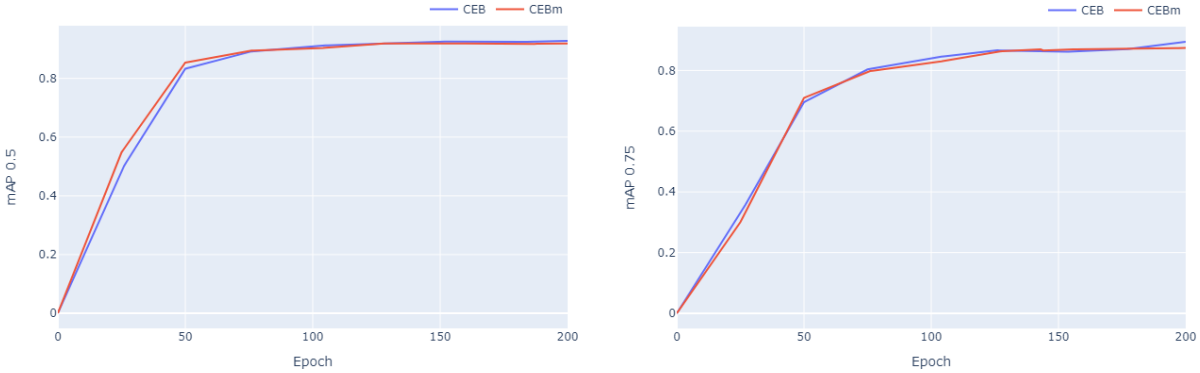


Figure 8: The Mean Average Precision results of the comparison of the CEB and CEBm models. The left figure shows the mAP measurements for an IoU of 0.5 and the right figure shows the mAP measurements for an IoU of 0.75. There is no significant difference in performance.