Representations in Object Detection and Instance Segmentation

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M. Sc. Hamd ul Moqeet Riaz
aus Gujranwala, Pakistan

Tübingen
2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	31.01.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	Prof. Dr. rer. nat. Andreas Zell
2. Berichterstatter/-in:	Prof. Dr. -Ing. Hendrik P. A. Lensch

Dedicated to my lovely daughter
Yashal

Abstract

Object detection and instance segmentation are emerging technologies, which are applied in various fields, including autonomous driving, health, fashion, sports, etc. The current research effort aims at reducing the complexity and speed of these models and at the same time improving the detection performance. One of the ways to achieve that is to represent objects in images more efficiently than the state-of-the-art methods. In this dissertation, we aim to investigate various representations in object detection and instance segmentation, which compresses the information, reduces the training effort, and at the same time provides meaningful insights into the representation.

We first show how class activation maps (CAMs) provide a reasonable estimate of the location of persons in thermal camera images, in neural networks trained on only class labels (weakly supervised). Furthermore, we show that CAMs representation could be employed to generate bounding boxes with decent accuracy.

Thereafter, we shift the focus of this thesis towards the task of instance segmentation. We show that the mask information could be compressed in coefficients of the Fourier series. We experiment with this using a single-stage object detection framework and compare the performance of our model *FourierNet*. We illustrate that low-frequency components of the Fourier series hold the overall shape information of the objects and higher frequencies contain the corner and edge information. Our model predicts the mask in polar coordinates, which can only generate star-shaped objects.

To overcome this, we shift our focus to implicit representations. First, we show a connection of Fourier series with implicit neural networks. We introduce an integer Fourier mapping and show that it forces periodicity in implicit neural networks. We explore and analyze the effect of the number of elements and standard deviation on our performance. Finally, we show that implicit neural representations can be employed for instance segmentation. We show that sub-sampling the pixel coordinates in an implicit neural network generates higher resolution output, which improves the qualitative and quantitative performance.

Kurzfassung

Objekterkennung und Instanzsegmentierung sind neue Technologien, die in verschiedenen Bereichen wie autonomes Fahren, Gesundheit, Mode, Sport usw. eingesetzt werden. Die aktuelle Forschung zielt darauf ab, die Komplexität und Geschwindigkeit dieser Modelle zu verringern und gleichzeitig die Erkennungsleistung zu verbessern. Eine Möglichkeit, dies zu erreichen, ist die effizientere Darstellung von Objekten in den Bildern im Vergleich zu den aktuellen Methoden. In dieser Dissertation wollen wir verschiedene Repräsentationen in der Objekterkennung und Instanzsegmentierung untersuchen, die die Informationen komprimieren, den Trainingsaufwand reduzieren und gleichzeitig aussagekräftige Einblicke in die Repräsentation liefern.

Wir zeigen zunächst, wie *Class Activation Maps* (CAMs) in neuronalen Netzen, die nur auf Klasseninformationen trainiert werden (schwach überwacht), eine vernünftige Schätzung der Position von Personen in Wärmekamerabildern liefern. Darüber hinaus zeigen wir, dass die CAMs-Darstellung verwendet werden kann, um Bounding Boxes mit angemesse Genauigkeit zu erzeugen.

Danach verlagern wir den Schwerpunkt dieser Arbeit auf die Aufgabe der Segmentierung von Instanzen. Wir zeigen, dass die Maskeninformation in Koeffizienten der Fourier-Reihe komprimiert werden kann. Wir experimentieren damit unter Verwendung eines einstufigen Objekterkennungssystems, vergleichen die Leistung und nennen unser Modell *FourierNet*. Wir zeigen, dass die niederfrequenten Komponenten der Fourier-Reihe die allgemeine Forminformation der Objekte enthalten, während die höheren Frequenzen die Ecken und Kanteninformationen enthalten. Unser Modell sagt die Maske in Polarkoordinaten voraus, wodurch nur sternförmige Objekte erzeugt werden können, und nicht-konvexe Masken nicht möglich sind.

Um dies zu überwinden verlagern wir unseren Schwerpunkt auf die implizite Darstellung. Zunächst stellen wir eine Verbindung zwischen der Fourier-Reihe und impliziten neuronalen Netzen her. Wir führen das Integer-Fourier-Mapping ein und zeigen, dass es eine Periodizität in impliziten neuronalen Netzen erzwingt. Wir untersuchen und analysieren die Auswirkungen der Anzahl der Elemente und der Standardabweichung auf unsere Leistung. Schließlich zeigen wir, dass die implizite neuronale Repräsentation für die Segmentierung von Instanzen verwendet werden kann. Wir zeigen, dass die Unterabtastung der Pixelkoordinaten in einem impliziten neuronalen Netz eine höher aufgelöste Ausgabe erzeugt, die die qualitative und quantitative Leistung verbessert.

Acknowledgments

I would like to thank my parents for their constant support and encouragement throughout my life, which allowed me to complete this journey. I would like to thank my colleagues and co-authors, especially Nuri Benbarka and Timon Hoefer, with whom I shared the successful and tough moments of my work and who made this journey a memorable one. I would like to thank my professor Prof. Andreas Zell, whose guidance and feedback were critical in improving the standard of my work. I would also like to thank my friend Waleed Gondal, who gave me technical and emotional support throughout these years.

Most importantly, I would like to thank my wife whose love, support, and compromise in my personal life were instrumental in finishing this work. Finally, I would like to thank Allah Almighty for His countless blessings on me.

Contents

Chapter 1

Introduction

In the last two decades, the computer vision and machine learning communities have been transformed. A major contributor to this advancement has been the availability of data and the convenience of generating new data from devices such as mobile phones/-cameras. The ample amount of data along with high-performance machines has enabled computer scientists to train machine learning models for real-world tasks. In recent years, these models have been successfully deployed in industries, including automotive, security, medical and commerce. The pivotal factors which pushed the industries to machine learning solutions are the state-of-the-art performance both in terms of accuracy and speed. Also, the cost of training a model has recently made comparable and viable to competitive solutions.

It must be emphasized that recently deep learning has shown promise in fields that were never thought to be a natural choice for machines. For example, GPT-3 (Brown *et al.* (2020)) generates human-like text and performs incredibly well on language tasks. Similarly, with StyleGAN (Karras *et al.* (2020)), one can generate realistic human faces with the ability to control facial features for the user. These tasks were traditionally considered the artist's domain, which required extensive human time and patience. However, a computer user can generate art effortlessly now with the help of these models.

With the advancements in research methods, there is still a need to deploy these models in real-world systems. For industrial applications, it is of utmost importance that the models are robust to varying conditions. For example, an autonomous car perception system must account for weather conditions and unseen scenarios. This is important, since these are safety-critical systems, and human lives could be endangered by wrongly perceived environments. Also, the perception systems should run in real-time on limited hardware on the systems. This typically adds some constraints to the size and type of models that could be employed.

Another important aspect is the interpretability of the systems. Humans interacting with autonomous systems are always more confident and comfortable when they know the reasons for the decision-making. Deep Learning models are typically considered a black box because they do not provide any intermediate representations. This usually concerns humans, especially in situations where a decision contradicts the user, even though being equally plausible. Therefore, it is desirable to have intermediate represen-

tations.

Lastly, the financial and environmental cost of training deep learning models should justify their use. Normally industries will always converge on a cheaper solution that fulfills their requirements. Therefore, researchers need to keep these aspects in mind while developing new ideas.

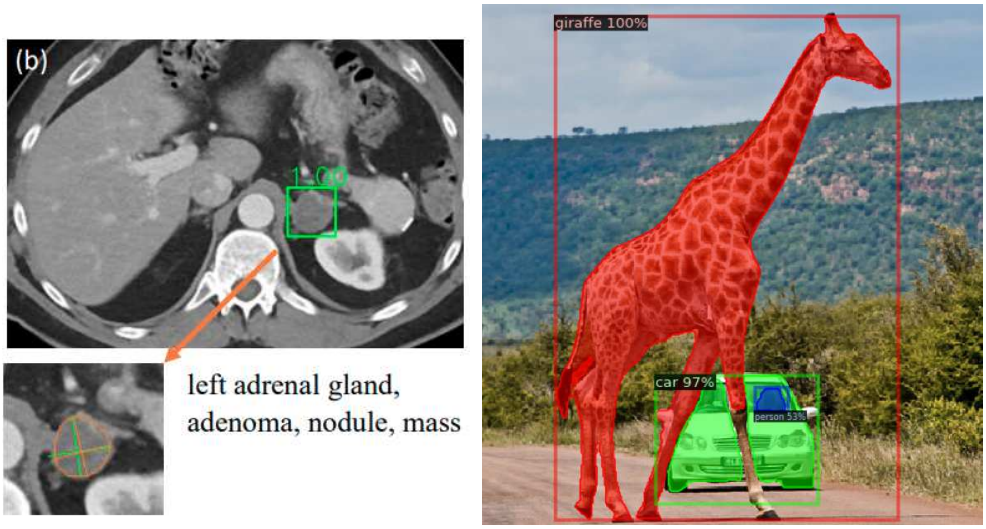
1.1 Object detection and instance segmentation

Object detection is one of the most widely researched topics in computer vision. Formally, *object detection* refers to classifying the type of objects (e.g person, cat, chair, etc) and finding their spatial locations in a given image. The spatial location of objects is typically represented by a bounding box. In certain situations, it is desirable to have a finer localization than just a bounding box. In such scenarios, one can classify each pixel of an object as being part of an object or not. This is a pixel-wise classification of all instances of objects termed as *instance segmentation*.

The most widely developed models of object detection and instance segmentation are in the autonomous driving industry. For any autonomous system, perception is the foremost part of its pipeline. A typical pipeline of autonomous cars is shown in figure 1.1. The systems perceive the environment first, before predicting the objects' future behavior and planning their trajectory. A wrongly perceived environment could be catastrophic for the whole pipeline. Studies have shown that until recently, the major cause of poor performance in autonomous driving is mis-detections. There is a strong effort from the computer vision community to improve the object detection frameworks and make them more robust.

One of the most recent use cases of object detection and segmentation has been in the medical industry. Computed Tomography (CT) scans and Magnetic Resonance Imaging (MRI) scans provide detailed images of internal body parts and organs. These images are critical for healthcare providers in identifying lesions and diagnosing diseases such as cancer. However, this diagnosis is difficult because the areas of concern are not identifiable by humans and sometimes experts' opinions also differ. For this purpose, deep learning frameworks have shown promise in finding lesions, characterizing and measuring them, and then describing them in a radiological report. For example, MULAN (Yan *et al.* (2019)) is a network that jointly detects, tags, and segments lesions in a variety of

Figure 1.1: A typical pipeline of autonomous driving vehicles.



(a) Internal CT scan and the detection from MULAN (Yan *et al.* (2019))

(b) An example of instance segmentation from our network.

Figure 1.2: Some examples of object detection and instance segmentation

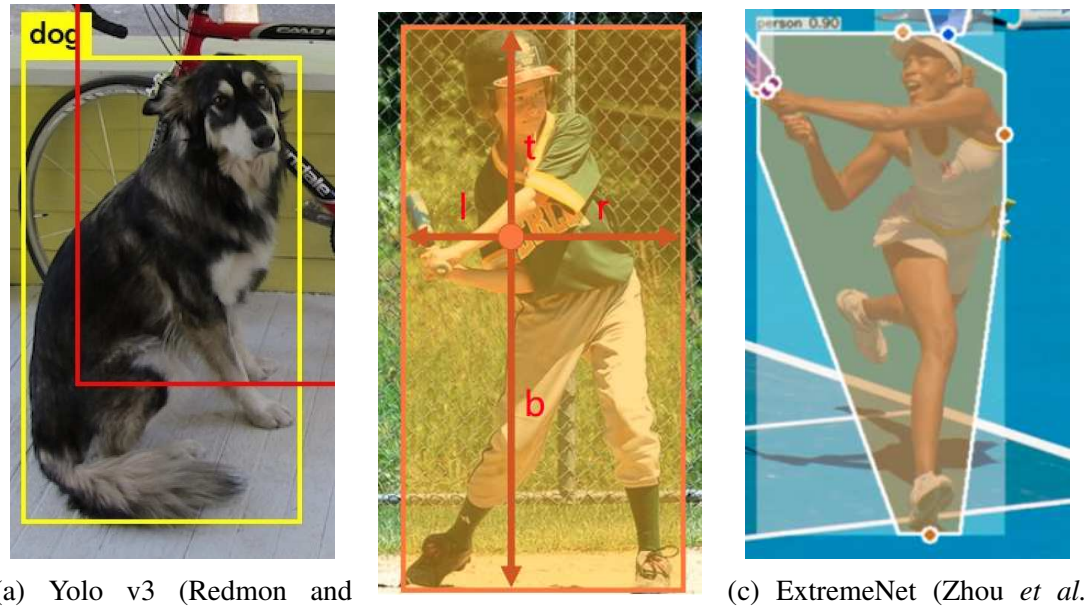
body parts. Figure 1.2a shows one of the examples of this network.

In agriculture, object detection has been widely used in many applications such as plant/fruit disease detection, animal health, crop monitoring, etc. Object detection is also a useful tool in retail where its often employed in people counting systems and customer interests. Other applications also include surveillance in security critical places.

1.2 Object Representations

One of the fundamental questions concerning object detection is the way objects can be represented. Representations play a pivotal role in the performance and speed of the models. Similarly, different representations can affect how users interpret the predictions of the model. In this section, we will provide an overview of possible ways objects can be represented using bounding boxes and masks and analyze their characteristics.

The typical approach for localizing objects in images is to predict a bounding box around the objects. A bounding box can be represented by the pixel locations of the top left corner of the box and the height and width of the box. This approach has been the standard for famous object detectors such as YOLO (Redmon and Farhadi (2018)) and Mask R-CNN (He *et al.* (2017)). Figure 1.3a shows an example prediction from YOLO v3. Recently, FCOS (Tian *et al.* (2019)) first predicts the centroid of the object and then the distance to the top, bottom, left, and right of the box (see figure 1.3b). This proved helpful for objects whose center of mass does not match the center of the box. One can also represent an object with a polygon. ExtremeNet (Zhou *et al.* (2019a))



(a) Yolo v3 (Redmon and Farhadi (2018))

(b) FCOS (Tian *et al.* (2019))

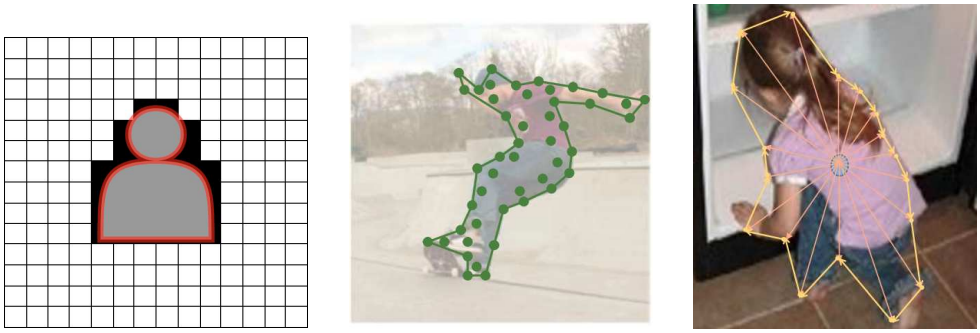
(c) ExtremeNet (Zhou *et al.* (2019a))

Figure 1.3: Some coarse polygon representations

predicts four extreme points around an object and constructs a polygon (figure 1.3c). This representation can be interpreted as a coarse segmentation mask having only four contour points.

The most straightforward and widely employed representation for instance segmentation masks is a grid representation (see figure 1.4a). Mask R-CNN (He *et al.* (2017)) is a two-stage detector, which first predicts a region of interest (ROI) and then classifies each pixel in the ROI to be either inside or outside the object boundary. This is an intuitive strategy that also suits the framework of a convolutional neural network, since it preserves spatial information. Two-stage methods consistently show better performance in terms of mask quality in the literature. However, these two-stage methods are generally slow and complicated during operation. Also, one can also argue that predicting all pixels is redundant because we only need to know the boundary contour of an object. Especially, the amount of redundant pixels increases manifolds in case of higher resolution output and larger objects. Therefore, there is a need to find compact representations and faster architectures.

Dense RepPoints (Yang *et al.* (2019)) predicts points near and inside the boundary of objects (figure 1.4b) and generates a contour by making a concave hull from these dense points. This certainly requires fewer parameters to define a mask compared to grid representation. However, dense points still require a large number of points to represent a mask. PolarMask (Xie *et al.* (2020)) uses polar coordinates to represent the mask. They predict the length of rays to the boundary of the object from a fixed center point as shown in figure 1.4c. This representation has the advantage of having few parameters and an in-



(a) Grid representation (He *et al.* (2017)) (b) Dense Representation (Yang *et al.* (2019)) (c) Polar representation (Xie *et al.* (2020))

Figure 1.4: Mask representations

tuitive representation. However, there are also some limitations regarding the complexity of the masks they can represent. Polar representation is limited to star-shaped objects e.g non-convex contours like donuts are not possible. Similarly, split masks are not possible with this approach, since we are predicting ray lengths from only one center point. A slightly different approach is to represent masks using parametric functions. Mask information could be encoded inside the parameters of a model/function and decoded to generate masks whenever needed. ESE-Seg (Xu *et al.* (2019)) employed Chebyshev polynomials to explicitly decode the shape information of the objects (figure 1.5). The advantage of a parametric function is that it is independent of the output resolution because it is continuous. Furthermore, such functions have the capability to be compressed according to the limitations of the application.

1.3 Aim and contribution

In this work, we aim to achieve object representations that are compact, fast, and interpretable. As discussed previously, it is critical in autonomous driving scenarios to

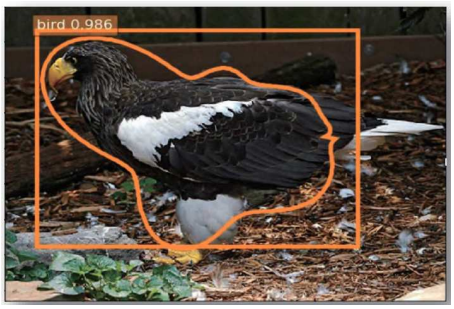


Figure 1.5: Parameteric representation ESE-Seg (Xu *et al.* (2019))

operate the perception systems at real-time speeds. At the same time, it is equally important that the models are memory efficient and computationally inexpensive. We aim to tackle this in our work through architecture search, non-complex models, and compact representations.

One of the main concerns about deep learning models is that they are a 'black box' and the predictions are not backed up by reasonable interpretations. In our work, we aim for interpretable representations for object detection tasks. Finally, we aim to achieve higher resolution and more accurate mask representations for instance segmentation.

Following is the summary of our contribution to the object and mask representations:

1. We show that humans can be represented by heatmaps in thermal camera images using the technique *class activation mapping* (CAM) in a weakly supervised setting. These heatmap representations from CAMs can be used to localize humans in the image.

2. We show that these human detection models can be optimized for hardware constraints and can be efficiently deployed to safety critical systems to make them more robust.

3. We show that a mask can be encoded inside the parameters of a Fourier series, which achieves a compact and meaningful representation.

4. We propose a differentiable shape decoder, which is able to learn the coefficients of a Fourier series and achieve automatic weight adjustment.

5. We show that implicit representations can be applied to the task of instance segmentation.

6. As implicit functions are continuous in the domain of input coordinates, we show that we can sub-sample the pixel coordinates to generate higher-resolution masks during inference.

7. We verify and illustrate that the rendering strategy from PointRend (Kirillov *et al.* (2020a)) brings significant qualitative gains for FourierMask. Our renderer MLP *FourierRend* significantly improves the mask boundary of FourierMask.

1.4 Structure of the thesis

The second chapter describes the foundations and theoretical background necessary to understand the rest of the work. It explains the neural network architectures used in this thesis. It also covers the details needed to understand the object detection framework. Furthermore, it briefly describes the datasets used in this thesis and the evaluation metrics used for analyzing the results. We also explain current work on implicit representations.

The third chapter describes the work done for people detection in thermal camera images in skiing resorts. The aim of this work was to improve the safety of skiers by detecting humans close to an operating snow groomer. We applied a weakly supervised strategy, in which we trained the models only on class labels, but predicted bounding boxes during inference. By using Class activation mapping (CAM), we could identify the regions in the image where people were present and consequently localize them using bounding boxes. We also performed an architecture search to optimize the models to run in real-time on limited hardware. Finally, the models were deployed on the real system and tested on the skiing resorts successfully.

The fourth chapter discusses the FourierNet (Riaz *et al.* (2021)), which is a network that uses a Fourier representation to represent masks for instance segmentation. This representation is compact since only a few parameters can generate reasonable masks compared to other methods using a similar number of parameters. Furthermore, we show that this representation is meaningful, since low-frequency components hold the overall shape of the object and high-frequency components are responsible for the edges and corners. Finally, we also conclude that FourierNet is a differentiable shape decoder, which is able to learn the coefficients of a Fourier series and achieve automatic weight adjustment. Since the Fourier series is differentiable, we apply the loss on the final contours, without the need to manually weigh each frequency component of the Fourier series in the loss function.

In the fifth chapter, we discuss how implicit representations and Fourier series are connected. We experiment with implicit neural representations on the task of image regression. We show that our integer mapping strategy enforces the periodicity of the network output. We also show that the progressive training strategy helps in generalization.

The sixth chapter uses the findings from the fifth chapter to apply them to the task of instance segmentation using our model FourierMask (Riaz *et al.* (2022)). We show that since the implicit representations are continuous in the input domain, we can sub-sample the input pixel location to generate higher resolution output, which is smoother and more accurate. We show that a rendering strategy from PointRend (Kirillov *et al.* (2020b)) can be applied to FourierMask and significant qualitative and quantitative gains can be achieved. The last chapter summarizes and concludes the contributions of this thesis.

Chapter 2

Foundations

This chapter focuses on the foundations and theoretical background of the thesis. It covers the topics which are common among all the chapters. Moreover, it contains concepts that need a deeper explanation to understand the later chapters. This chapter includes the necessary literature which is related to the work in this dissertation.

2.1 Backbones

This section describes the common neural network backbone architectures employed in this work for object detection. Typically, object detection and instance segmentation models have a *backbone* network which extracts meaningful features. These features are passed on to *heads*, which perform specific functions like classification, bounding box and mask predictions, etc. In this section, we will describe common backbones used in this thesis.

One of the first and most widely used backbones was VGG-16 (Simonyan and Zisserman (2014)). The architecture of VGG-16 is shown in figure 2.1. Briefly, it uses 3×3 convolutional kernels with a stride of 1. It applies max pooling operations to reduce the feature map size, while at the same time increasing the number of filters. This squeezes the spatial dimension and extracts more global features from the image. For the classification task, it employs fully connected layers at the end to predict meaningful output from global features. VGG was one of the standard baselines for a variety of computer vision tasks because of its intuitive architecture. However, the major disadvantage of this model was the size of its parameters and high memory and computational needs. VGG-16 has more than 130 million parameters, for which fully connected layers are mostly to blame.

ResNet (He *et al.* (2016a)) provided an alternative to train deeper networks by introducing residual learning blocks. In figure 2.2a, it is illustrated that each residual layer has an identity skip connection from input $\mathbf{x}$ to the output $\mathbf{y}$, which enforces the network to learn the residual mapping $\mathcal{F}(\mathbf{x}, W_i)$. The output can be calculated as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, W_i) + \mathbf{x}. \tag{2.1}$$

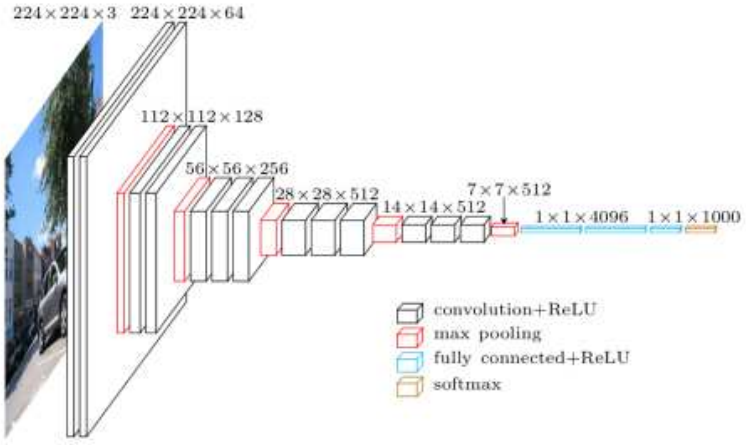
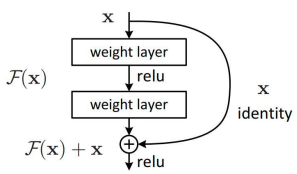


Figure 2.1: The VGG-16 architecture (Simonyan and Zisserman (2014))

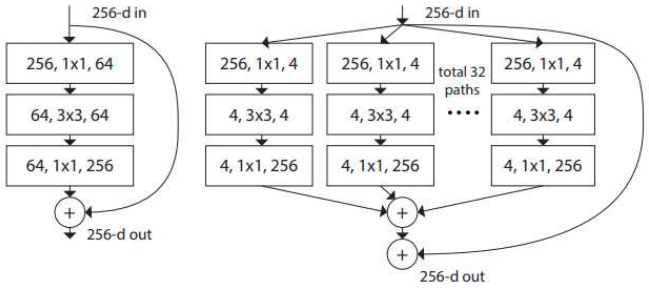
ResNet also added a global average pooling layer before the fully connected layers, which significantly reduced the number of parameters compared to VGG. This allowed ResNet to train very deep networks (152 layers) while still managing to keep the memory footprint low.

ResNeXt (Xie *et al.* (2016)) introduced a multi-branch architecture, which aggregated a set of transformations with the same topology. This architecture is shown in figure 2.2b. This ”cardinality” proved to be more significant in improving the performance than increasing the width or depth of the network.

Feature Pyramid Networks (FPN) (Lin *et al.* (2017a)) have proved to be a really important feature for multi-scale object detection. Semantic features for large and small objects need to be extracted at different scales in the backbone to have a decent detection performance. (Lin *et al.* (2017a)) used feature pyramids with marginal extra cost compared to previous architectures. Figure 2.3 shows a feature pyramid, which predicts objects at all feature levels and thus proves better for both large and smaller objects.



(a) Residual learning block (He *et al.* (2016a))



(b) **Left:** ResNet block, **Right**: ResNeXt (Xie *et al.* (2016))

Figure 2.2: ResNet variants

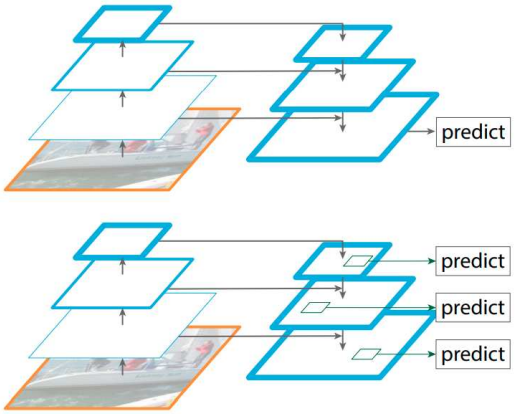


Figure 2.3: Feature Pyramid Network (Lin *et al.* (2017a))

2.2 Object detection framework

Object detection frameworks can be broadly classified into single-stage and two-stage methods. In two-stage methods, the networks first propose regions with a high probability of objects in them. Then, they predict classes, bounding boxes and masks based on these regions. These methods generally show better performance in terms of accuracy, but they are slow and complicated. On the contrary, single-stage methods detect the objects in a single shot without the need for intermediate proposals. These methods are generally fast and straightforward but they lack the high accuracy of two-stage methods.

2.2.1 Single-stage methods

Single-stage methods generally divide the spatial dimensions of the image in a grid. For each grid cell, they predict whether an object is present, its class and the bounding box coordinates. FCOS (Tian *et al.* (2019)) is an example of single stage detector, which predicts the bounding boxes at each grid cell of the feature map with a height H and width W as shown in figure 2.4. It also predicts the centerness, which refers to the probability of that feature map location being the center of an object.

Another example of a single-stage instance segmentation method is PolarMask (Xie *et al.* (2020)). Rather than predicting the bounding box, polarmask predicts the ray distances to the boundary of the object from the center point.

2.2.2 Two-stage methods

Two-stage instance segmentation splits the task into two sub-tasks, object detection and then detection/segmentation. The most prominent object detection method is Faster R-CNN (Ren *et al.* (2015)), which employs a region proposal network (RPN) to first

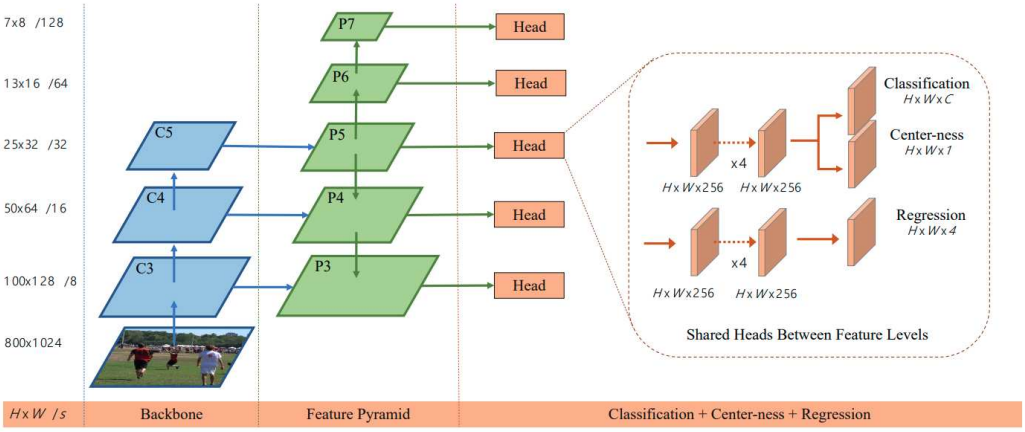


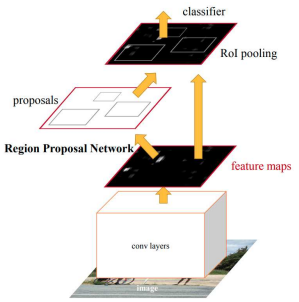
Figure 2.4: FCOS architecture (Tian *et al.* (2019))

propose candidate regions. Then it uses region of interest (ROI) pooling to generate fixed-sized feature maps for heads. Figure 2.5a shows the idea of the network.

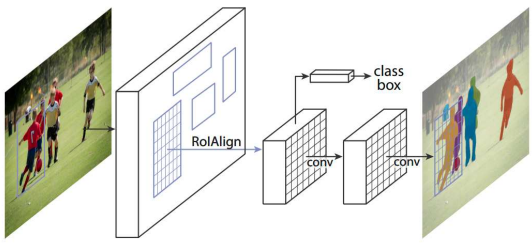
Mask R-CNN (He *et al.* (2017)), which is constructed on top of Faster R-CNN (Ren *et al.* (2015)) by adding a mask branch parallel to the bounding box and the classification branches. further, they used RoI-Align instead of RoI-Pooling. Figure 2.5b shows the architecture of the Mask R-CNN head.

Region proposal networks

Region proposal networks (RPN) were introduced in Faster R-CNN (Ren *et al.* (2015)) to generate candidate regions for objects in a fully convolutional manner. Figure 2.7 shows the working of RPN. They use a $n \times n$ sliding window on the last convolutional feature map of the backbone network. This sliding window is mapped to a lower dimensional feature and then fed into two sibling fully connected layers —a box-regression layer (reg)

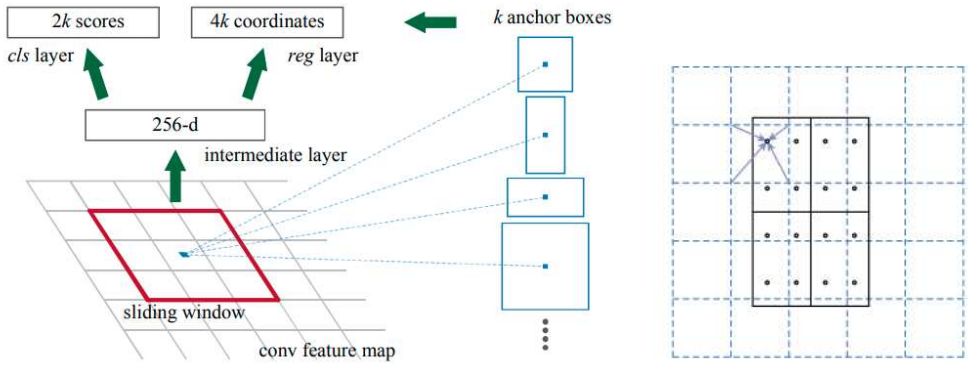


(a) Faster R-CNN (Ren *et al.* (2015))



(b) Mask R-CNN (He *et al.* (2017))

Figure 2.5: R-CNN architectures



(a) Region proposal Network (b) ROI align

Figure 2.6: Object detection components

and a box-classification layer (cls). Since it's a sliding window, it shares the weights at all spatial locations in the feature map.

ROI Align

Region of interest (ROI) pooling is an important component of two-stage object detectors because they generate fixed-sized feature maps which are essential for the network design. Especially in heads that have fully connected layers and are not fully convolutional, it is necessary to know the number of features during the design process. Faster R-CNN used a ROI pooling layer to achieve this. They divide the proposed regions into a fixed-sized grid and choose the maximum value from the feature map region which corresponds to that grid cell. Since there is quantization in ROI pooling, a lot of information is lost in the max pooling operation. Therefore, Mask R-CNN introduced ROIAlign, which is shown in figure 2.6b. In ROIAlign, the values are calculated using bi-linear interpolation instead of max pooling. This allows information to be passed on from multiple feature locations and improves the performance of the pooling operation.

2.3 Microsoft COCO Dataset

The Microsoft COCO (Common Objects in Context) (Lin *et al.* (2014)) is an object detection, segmentation and captioning dataset. The dataset contains images of common objects in natural contexts in everyday scenes. The objects in the images are labeled with bounding boxes and instance segmentation annotations. The original COCO 2014 dataset contained 91 object categories, which could be recognized easily by children. It has a total of 328,000 images, in which a total of 2.5 million instances are present.

Compared to other object detection datasets, MS COCO was state-of-art at its time. Figure 2.8 shows a comparison between MS COCO and various classification and de-

(a) Image (b) Instance segmentation label

Figure 2.7: Example from MS COCO dataset (Lin *et al.* (2014))

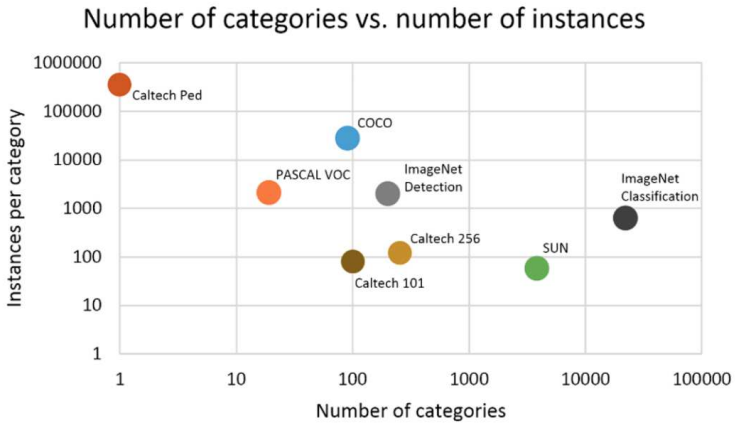


Figure 2.8: Comparison of different object detection datasets (Lin *et al.* (2014)).

tection datasets. From the figure, it can be seen that COCO has the most instances per category for datasets having multiple classes. MS COCO has more categories and instances per category than Pascal VOC (Everingham *et al.* (2012)), which is the most similar in type and distribution to MS COCO.

2.4 Evaluation Metrics

To analyze and compare the performance of object detection and instance segmentation models, it is necessary to evaluate them on criteria that focus on the localization capability of the models. For classification tasks, metrics such as accuracy (Top-1/Top-5) could be used. One could also evaluate precision, recall and F1 score to get further insights into the performance of the networks from different aspects. However, in the case of object detection and segmentation tasks, one can only consider a prediction accurate if the predicted location or pixels are close to the target regions. Therefore, the first step is

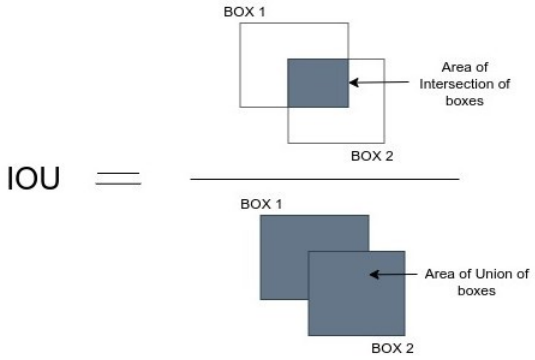


Figure 2.9: Intersection over Union

to formulate a metric that calculates the overlap of predictions and targets.

2.4.1 Intersection over Union

Intersection over Union (IOU) is the ratio of the area of intersection and the area of union between two regions. In the case of bounding boxes, IOU can be calculated as shown in figure 2.9. In case the predicted bounding box completely overlaps the target, the IOU has a value of 1 (maximum) because the intersection and union areas are equal. In case there is no overlap, the intersection is 0 (minimum) and consequently, IOU is also zero. In the case of instance segmentation, we calculate the ratio of overlapping pixels by the total pixels covering the prediction and targets.

2.4.2 Mean Average Precision (mAP)

IOU provides a measure of the localization ability of a detector. However, we need a metric that calculates the precision of a detector for all datasets. For this, we first need to calculate the precision and recall. Precision and Recall are defined as:

$$Precision = \frac{TP}{TP+FP}, \tag{2.2}$$

$$Recall = \frac{TP}{TP+FN}, \tag{2.3}$$

where True Positives (TP) are those predictions that have IOU greater than a threshold (typically 0.5). There are multiple possibilities for False Positives (FP):

- The examples which have an IOU less than the threshold
- There is a mismatch in the predicted and target class.

Figure 2.10: Precision-Recall curve

- There is more than one prediction for a single target. Then the highest confidence prediction is considered TP and the rest are categorized as FP.

False Negatives (FN) are the instances where there was no prediction while a ground truth label was present. A high precision suggests that most of the predictions are correct. On the other hand, a high recall suggests that most of the ground truth targets were predicted by the detector.

The average precision (AP) is defined as the area under the precision-recall curve. Figure 2.10 shows one of the examples. This curve is generated by calculating the precision and recall at each confidence threshold of the prediction. For example, at a high confidence threshold, the precision is really high because there are few predictions and most of them are correct (few FP). However, the recall is very low because of the large number of FN. On the contrary at a low confidence threshold, a lot of objects are detected. which results in low FN and high FP. This results in a high recall but low precision. For COCO, precision and recall are calculated at 10 different confidence thresholds and plotted on a precision-recall (PR) curve. The PR curve follows a kind of zig-zag pattern as recall increases absolutely, while precision decreases overall with sporadic rises. At each recall level, we use the maximum precision value to the right of that recall level, as shown in figure 2.10. Finally, the area under this smoothed out curve is calculated which gives us the Average Precision (AP). The mean average precision (mAP) is the AP averaged over all classes in the predictions. COCO does not differentiate between AP and mAP.

Typically, the IOU threshold is set to 0.5 in object detection metrics. However, MS COCO uses 10 evenly spaced IOU thresholds between 0.5 and 0.95 and takes the average over all the IOU thresholds. They claim that averaging IOUs rewards detectors with better localization performance.

2.4.3 Boundary IOU

IOU rewards the pixels/area inside an object equally. However, for instance segmentation tasks, it is important to focus on the boundary of the object. Especially for larger objects,

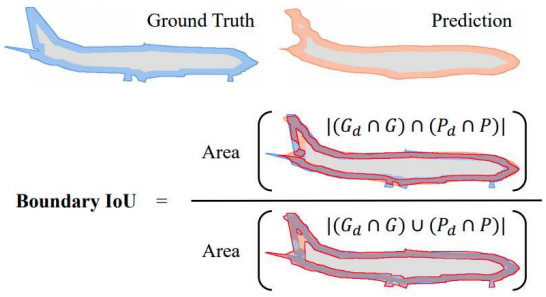


Figure 2.11: Boundary IOU

misclassifications around the boundary of the objects become insignificant because most of the pixels are inside the objects. Therefore, metrics like mean average precision fail to highlight the difference between objects with poor boundary predictions. Boundary IOU (Cheng *et al.* (2021)), explicitly focuses on the boundary pixels of the objects and therefore portrays the actual difference between good and bad mask quality. Figure 2.11 illustrates the way boundary IOU is calculated. First, it finds the intersecting pixels near and inside the boundary of the ground truth mask and the predicted mask. Then, it calculates the union of the same boundary pixels of ground truth and prediction. Finally, it takes the ratio of intersection and union similar to the normal IOU. Boundary IOU gives a better estimate of the mask quality of the model. Just by replacing IOU with boundary IOU, one can calculate boundary mean average precision or mAP_{bound}

2.5 Training and loss functions

Deep learning models must be trained on a large amount of data. In this work, we employ neural networks only and therefore we limit our scope to common methods of training such networks. Neural networks are typically trained using *backpropagation*. Backpropagation is a method that calculates the gradient of the loss function with respect to the weights of the neural network. This allows gradient-based optimization methods such as gradient descent to train these networks. For large datasets, a variant of gradient descent 'Stochastic Gradient Descent (SGD)' is typically used. SGD is an iterative method for optimizing the loss function, which uses a randomly selected subset of data to estimate the gradient in each iteration. However, in this work, the primary focus is on object detection and segmentation. Therefore, we do not delve deep into optimization methods and backpropagation and rather focus on loss functions employed in this work for detection and segmentation.

2.5.1 Binary cross entropy

Loss functions allow an evaluation of the network prediction compared to the targets. Especially for calculating the gradients for backpropagation, error or loss functions are needed. The most common loss function for classification tasks is binary cross-entropy, which is defined as:

$$H_p(q) = -\frac{1}{N}\sum_{i=0}^{N} y_i \log(p(y_i)) + (1-y_i)\log(1-p(y_i)), \tag{2.4}$$

where N are the total number of examples, y_i is the target and $p(y_i)$ is the predicted probability for that example. The first term $y_i \log(p(y_i))$ in the sum refers to the log-likelihood of that class and the second term $(1-y_i)\log(1-p(y_i))$ refers to the log-likelihood of other classes based on this prediction. In many cases the target classes are mutually exclusive, for example, the result of a coin toss could either be heads or tails but not both. In such cases, the second term is always zero because y_i is always 1. Therefore, for mutually exclusive classes with one-hot encoding, we can rewrite the above equation in the following way

$$H_p(q) = -\frac{1}{N}\sum_{i=0}^{N} y_i \log(p(y_i)). \tag{2.5}$$

Natural images generally contain multiple objects such as people, vehicles and animals etc. Therefore, a classification of an image containing multiple objects of different categories is not mutually exclusive. One could extend equation 2.4 to be applied to multiple classes by taking the sum for all classes in a particular dataset.

2.5.2 Polar IOU loss

Calculating the IOU between the ground truth and predicted masks is computationally expensive. (Xie *et al.* (2020)) introduced Polar IOU, which provided a more suitable evaluation metric for polar representations. Figure 2.12 illustrates the process of calculating polar IOU. Polar IOU is the ratio of the minimum ray distances to the maximum ray distances between two contours. If there are infinite rays, this effectively becomes equivalent to the ratio of intersecting and the union area. For discrete cases, Polar IOU can be defined as:

$$PolarIOU = \frac{\sum_{i=1}^{N} d_{min}}{\sum_{i=1}^{N} d_{max}}, \tag{2.6}$$

where N is the total number of rays, and d_{min} and d_{max} are the ray distances explained above. Polar IOU loss is simply the binary cross entropy of Polar IOU, which can be written as

$$PolarIOU\ Loss = \log \frac{\sum_{i=1}^{N} d_{max}}{\sum_{i=1}^{N} d_{min}}. \tag{2.7}$$

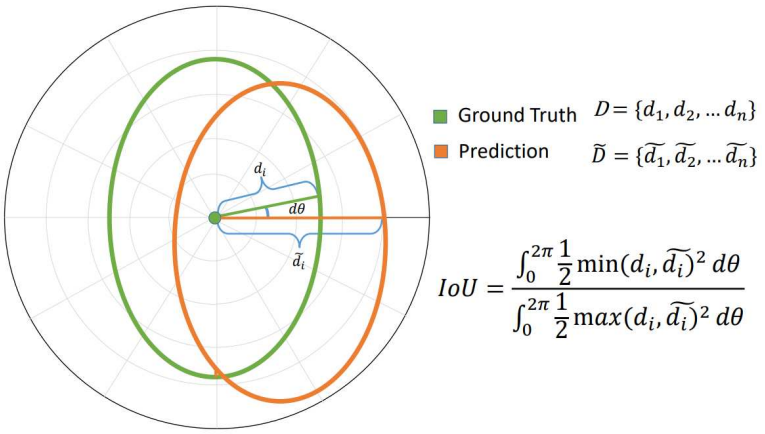


Figure 2.12: Polar IOU (Xie *et al.* (2020))

Polar IOU loss provides a reasonable estimate of the quality of the mask prediction, which can be easily replaced with typical loss functions for mask IOU calculations.

2.6 Fourier Series

Fourier series is a way by which any periodic signal can be represented by a sum of sine and cosine terms. If enough of these terms are present, theoretically any periodic signal can be represented by the Fourier series. Each sine and cosine term has a harmonic frequency (integer multiple of the fundamental frequency of the periodic function). In sine-cosine form, the Fourier series is defined as:

$$s_N(x) = \frac{a_0}{2} + \sum_{n=1}^{N} (a_n \cos(\frac{2\pi}{P} nx) + b_n \sin(\frac{2\pi}{P} nx)), \tag{2.8}$$

where N is the total number of harmonics, a_0 is the 0^{th} harmonic or the constant term, a_n and b_n are the coefficients of the n^{th} harmonic cosine and sine terms respectively, P is the period of the function. To represent a periodic signal as Fourier series, the coefficients a_0, a_n and b_n can be calculated as follows.

$$a_0 = \frac{2}{P} \int_0^P s_N(x) dx \tag{2.9}$$

$$a_n = \frac{2}{P} \int_0^P s_N(x) \cos(\frac{2\pi}{P} nx) dx \tag{2.10}$$

$$b_n = \frac{2}{P} \int_0^P s_N(x) \sin(\frac{2\pi}{P} nx) dx \tag{2.11}$$

Fourier series can also be formulated in exponential form. It is defined as

$$s_N(x) = \sum_{n=-N}^{N} (c_n \exp^{\imath 2\pi nx/P}), \tag{2.12}$$

where c_n are the complex Fourier coefficients. This is the typical form to represent a function as a complex-valued function.

Fourier series can only represent functions that are periodic. For non-periodic functions, the Fourier series can be extended to a Fourier transform. The Fourier transform assumes non-periodic functions as periodic functions with infinite periods. Using the Fourier transform, a frequency domain representation can be generated of non-periodic functions and consequently advantages of Fourier approximation can be exploited for many applications. For signals which are finite sequences of equally-spaced samples (for example images), the Discrete Fourier Transform (DFT) is a natural choice. A DFT generates a complex-valued function of frequency that has the same length as the original signal. A DFT can be considered a frequency domain representation of the original signal. A DFT of a complex number sequence of length N $\mathbf{x}_n := x_0, x_1, \ldots, x_{N-1}$ is another complex number sequence $\mathbf{X}_k := X_0, X_1, \ldots, X_{N-1}$, which is defined by

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-\imath 2\pi}{N} kn} \tag{2.13}$$

This is the discrete form of the formula for the coefficients of the Fourier series. Therefore, the inverse discrete Fourier transform (IDFT) is equivalent to a Fourier series. It is defined as

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{\imath 2\pi}{N} kn} \tag{2.14}$$

This relationship is really useful in our application. We could represent our signal, for example, objects, masks or contours as x_n in the above equation. If our networks could learn to predict the Fourier coefficients X_k in the frequency domain, an IDFT can be applied to retrieve the original representation. This could potentially have a variety of

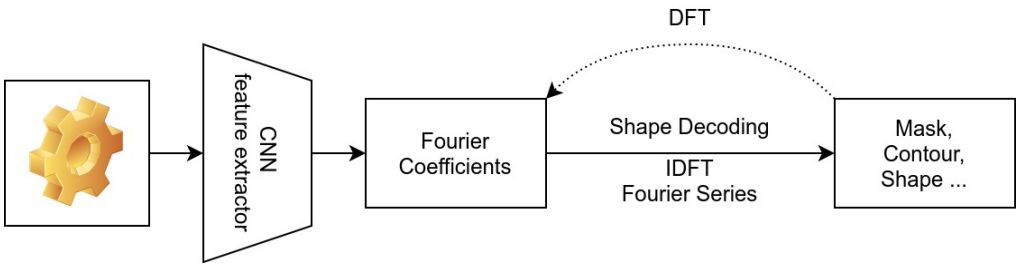


Figure 2.13: Fourier series flowchart for Instance segmentation

advantages such as compression and better interpretability.

Figure 2.13 shows that a convolutional neural network feature extractor can predict the Fourier coefficients of a particular object. An IDFT (Fourier series) can be applied using these Fourier coefficients to decode a mask, contour or shape of the object. Since the Fourier series is a linear operation, gradients can flow through the function when networks are trained using backpropagation. Therefore, the models can be trained in an end-to-end manner and typical loss functions on masks or contours can be employed and the Fourier coefficients are learned automatically.

Alternatively, it is also possible to apply a DFT on the mask or contour and retrieve the Fourier coefficients analytically. Then, a loss for regression such as L2 can be applied to the Fourier coefficients directly. However, in this case, it is necessary to weigh each coefficient independently according to its effect on the actual shape of the object. This makes training more complicated and introduces unwanted hyper-parameters.

2.7 Implicit Neural Representations

Implicit neural representations (INRs) learn a continuous differentiable signal in the parameters of a neural network. Typically, signal representations are discrete, for example, images are discrete pixels, and 3D signals can be represented as point clouds or discrete voxels. Implicit representations represent these signals as continuous in the input domain. For example, an image's input coordinates are mapped to the RGB value at that location. Figure 2.14 shows an example of an image being represented by an Implicit neural network. The word implicit refers to the fact that these representations are not tractable and it's not possible to write an analytical formula for such signals. Since these representations are continuous, they have a variety of benefits. They are not tied to the spatial resolution of the signals and therefore the memory requirement is independent of the resolution and depends on the complexity of the signal. Tasks such as

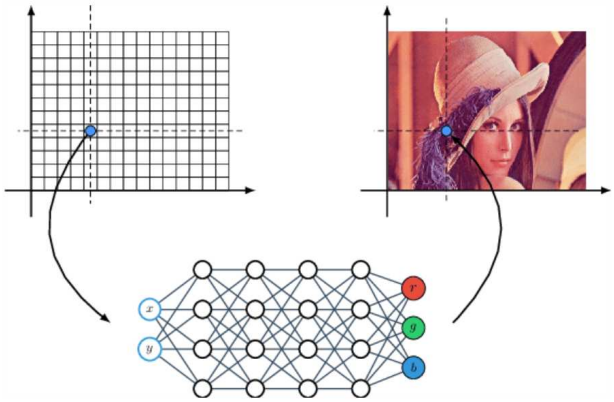
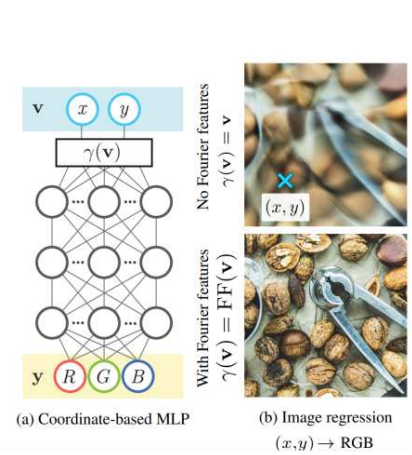
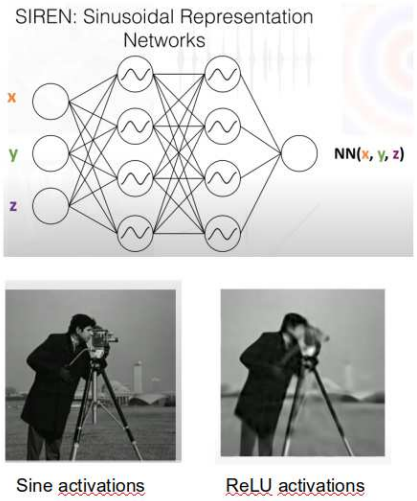


Figure 2.14: Implicit Neural Network (Skorokhodov *et al.* (2021))



(a) Fourier Features (Tancik *et al.* (2020))



(b) Sirens (Sitzmann *et al.* (2020a))

Figure 2.15: Recent works in Implicit neural networks

super-resolution are an ideal fit for such representations because INRs have an infinite resolution theoretically. Also, the signals with higher dimensions such as 3D scenes require very high memory and INRs are suitable for 3D tasks.

Recent work in implicit neural networks has shown that these representations are able to capture high-frequency details in the signals. (Tancik *et al.* (2020)) illustrated that if a Fourier mapping γ is applied to the input coordinates before passing it through the MLP, the MLP learns high-frequency details better than without the Fourier mapping. This is illustrated in figure 2.15a, where it can be seen that Fourier features generate sharper images when applied to the image regression task.

Sirens (Sitzmann *et al.* (2020a)) showed that instead of using activation functions such as ReLU, periodic activation functions are ideally suited for representing complex natural signals and their derivatives. They demonstrate the effectiveness of sinusoidal activations for the representation of images, wavefields, video, sound, and their derivatives. Figure 2.15b shows a comparison between ReLU and sine activations when used on image regressions task.

2.8 Weakly supervised object detection

One of the major challenges in training well-generalizing models is the availability of data. For large deep-learning models, it is imperative to have millions of annotated training examples. For image-level classification labels, it is fairly quick and straightforward to annotate the images. Also, there is no need for expert skills in humans to label images. However, for dense labeling tasks such as pixel-wise segmentation annotation, it requires

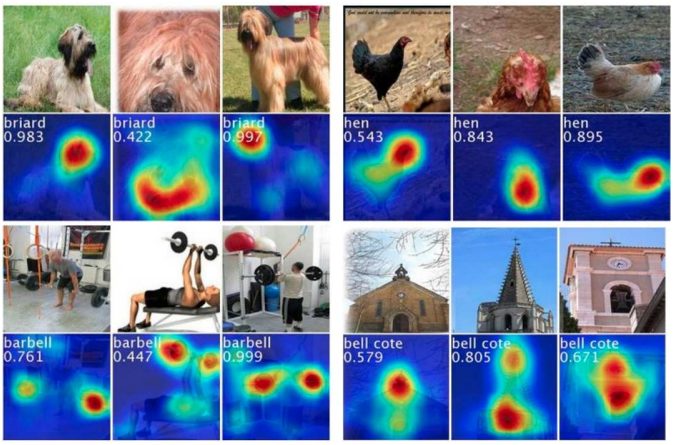


Figure 2.16: Class activation mapping (Zhou *et al.* (2016))

time and patience from skilled annotators.

A decent amount of research has been done on developing tools and methods that can label data without human input. Also, researchers have recently explored methods, which do not require fully annotated data for their applications. Weakly supervised is such a training technique by which models can perform tasks on which they haven't been explicitly trained. The term *weak* refers to the fact that the labels used are incomplete or contain insufficient information for training the model for that task and therefore they cannot be termed fully supervised. An example of a weakly supervised model is a convolutional neural network trained on classification labels only but predicting bounding boxes as well during inference.

Weakly supervised models exploit the information extracted from the structure of the models. In weak supervision, we assume that while training for the higher level task, the models inherently learn correlations and features which could lead us to further insights. For example, it is possible to trace the pixels responsible for predicting a particular class in an image. (Zhou *et al.* (2016)) showed that we can trace the activations in the convolutional feature maps by a technique called class activation mapping (CAM). Using the inherent structure of global average pooling, (Zhou *et al.* (2016)) takes a linear combination of activations in the feature maps, which generates a heatmap highlighting the relevant pixels for that class. Figure 2.16 shows an example of class activation mapping for various classes.

Weak supervision is even more valuable in the case of instance segmentation because labeling pixels for segmentation takes manifolds more resources than bounding box labels. (Zhou *et al.* (2018)) used only image-level labels to predict the segmentation masks of objects. They used local maxima in the class response maps to find the instances in the images. The peaks were then backpropagated to find the pixels contributing to that local maximum. Figure 2.17 shows how the peaks are backpropagated to the image pixels for instance segmentation.

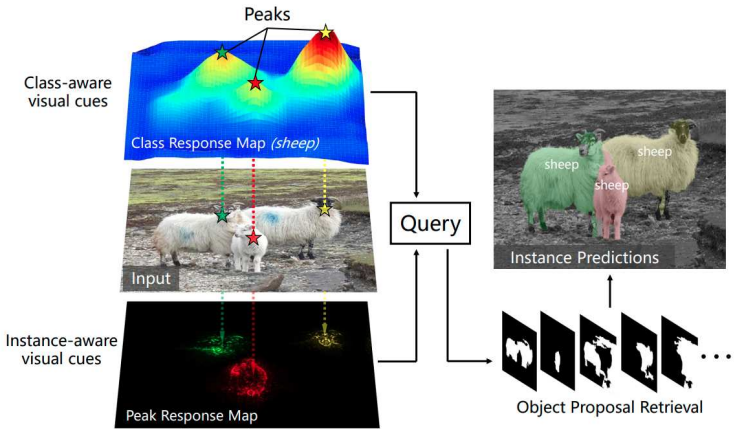


Figure 2.17: Class peak response (Zhou *et al.* (2018))

Chapter 3

People detection using weak supervision

3.1 Introduction

One of the foremost challenges in autonomous driving is perceiving the environment. An accurate depiction of the environment becomes crucial when the safety of people is at stake. Therefore, the task of object detection has been of major interest in computer vision during the last decade. In urban scenarios, the recent object detection frameworks have been able to robustly classify and localize objects in the environment. This has led to vehicle manufacturers confidently adopting object detection models in their autonomous driving systems.

Although technology is moving towards fully autonomous vehicles, there are certain environments in which it's still challenging to perceive the environment confidently. Objects, which are articulated or change orientation, are hard for machines to recognize. Especially with traditional computer vision methods with hand-crafted features, it was even more challenging. Furthermore, in scenarios where object features are indistinguishable from the background due to occlusions or color similarities, object detection becomes more difficult.

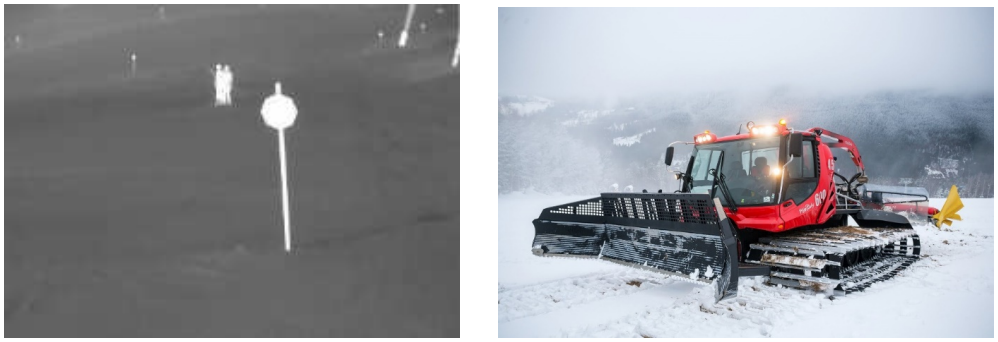
With the recent success of deep learning methods, the aforementioned scenarios have been tackled to some extent. However, deep learning training requires large amounts of labeled datasets. The labeling process is tedious, and generally requires a sizeable amount of manual labeling hours and computational resources. One of the ways to overcome the manual labeling process is to adopt *weakly supervised* training. In weakly supervised methods, the models are trained using partial/limited labels compared to the task at hand. For example, a weakly supervised object detector could only be trained on class labels, while it predicts both class labels and bounding boxes of objects during inference. These methods are applicable for multiple tasks, for example, labeling, finding patterns, and an elementary solution for simple problems.

3.2 Task

In this chapter, we explain the work done on a similar scenario, where the objects had articulation, changing orientation and similarity with the background. Formally, we wanted to explore the task of detecting people in skiing resorts using thermal imaging cameras. There is a constant requirement in skiing resorts to groom the snow to maintain a good surface for skiers. Snow groomers (figure 3.1b) are employed for this task which level and smooth the snow. The snow groomers are tracked vehicles and thus maintain strong traction during operation. However, these heavy machines come with a cost of safety hazards for skiers and employees working near the vehicle. There have been multiple fatal accidents reported. Since the skiers often fall and get covered in snow, it is even more challenging for the vehicle operators to identify people around. Furthermore, low visibility during snowfall and fog is also a common occurrence in these resorts, which increases the safety hazard. To overcome these challenges, vehicle manufacturers search for an automated safety solution.

To develop an automated safety system, these vehicles are attached with four thermal imaging cameras, one each at the corners of the passenger compartment facing outside. These cameras are suitable for skiing resorts because most of the background is snow, which does not emit any infrared signature and the objects of interest (people) are easily distinguishable because of a comparably high temperature. This gives an advantage over RGB cameras because the visibility and object appearance does not play a role in detection. Therefore, people can still be better distinguished in stormy and foggy conditions. Furthermore, people with white clothes or slightly covered in snow are better visible. An example image is shown in the figure 3.8.

Along with the advantages of thermal images, there are also some drawbacks. Having no color information means that the shape of objects becomes really critical in identifying objects. For example, people far away could look similar to signposts. Backgrounds of trees and rocks could also pose problems because they are at a higher temperature than



(a) Thermal camera example (b) Pistenbully

Figure 3.1: A dataset example and a snow groomer

snow. Traditional computer vision methods, which extract hand-crafted features to detect objects, fall short in these scenarios. Therefore, we wanted to test deep learning methods for detecting people in thermal infrared images.

3.3 Dataset

Four thermal imaging cameras (FLIR (2022)) were attached at the corners of the passenger compartment of the snow groomer (Pistenbully) shown in figure 3.1b. The Forward Looking Infrared (FLIR) cameras detect passive infrared from the field of view and generate its heatmap. In our case, this heatmap was represented as a grayscale image of resolution 382×286 pixels. Since the cameras operated at 30 frames per second, for each second of the video, 30 grayscale images were generated. We received the data from all four cameras simultaneously. Data was collected from people skiing at two different locations namely Kaunertal and Steibis glaciers. For data diversity, data was collected at different times and conditions e.g different days, weather conditions, places at the resort, etc. Since it was important to detect non-typical scenarios, particular attention was given to examples of people who were huddled together (occlusions), fallen, moving very close to the camera, and in bright backgrounds.

The table 3.1 shows the statistics of the data set. In positive images, there is at least one person in the frame and in negative images, there is none. We can observe that the percentage of positive images is less than 10%, which indicates a high imbalance in the data set. During training, we adopt an under-sampling strategy to balance the positive and negative samples. This makes sure our model is not biased towards always predicting no person.

We split the data into training and test sets in the ratio 0.85 to 0.15 respectively. A typical way to choose the test data is to take frames uniformly randomly from the whole dataset. However, since it is a video sequence, this would make the test examples highly similar and correlated to the training examples. We wanted to make the test set similar to a practical test scenario. Therefore, we used the last set of video sequences of each day/scenario for test data. This would match a natural test scenario because we test

Dataset	**All data**	**Train**	**Test**	**Test Ratio**
Total Images	2,809,041	2,382,372	426,669	15%
Positive Images	277,394	248,297	29,107	10%
Negative Images	2,531,647	2,134,075	397,562	15%
Positive Ratio	9.8%	11.6%	6.8%	-

Table 3.1: The statistics of the dataset. The positive images contain at least one labeled person. The table values highlight the imbalance in the dataset since only less than 10% images have persons in them.

our systems on separate occasions after models are trained. Also, using our strategy, the test examples would be independent of training examples but still be in a similar environment. Note that the positive examples ratio in the test set (6.8%) does not match the positive examples in the training set (11.6%) and this could potentially lower our test accuracy. However, we did not force the ratios to be similar because then there would be human interference involved in picking the examples and also disturb the natural way of splitting described above.

In the dataset each person was labeled with three sub-classes: Standing, lying, and blurred. However, most of the time people were standing, which is natural for humans in skiing resorts. Therefore, there was a further imbalance of examples in the human category. To avoid this, we did not use any sub-classes and relied on the higher-level class of people only. It is worth mentioning that a CNN feature extractor could learn to be invariant to human poses and orientation if enough examples with high variance are provided.

3.4 Method

In this section, we will discuss the architecture of our deep-learning model and techniques for localizing the objects in the image. As discussed before, we had the task of classifying the images in the categories 'Person' and 'No Person' and predicting an enclosing bounding box around the objects. We employed a weakly supervised model for this work i.e we did not use the bounding box labels for training but predicted the labels during inference.

3.4.1 Architecture

To achieve this, we employed a classification network derived from VGG-16 (Simonyan and Zisserman (2014)). VGG-16 is a baseline architecture for many modern classification and detection models. It was one of the earliest convolutional neural networks which evaluated increasing depth using an architecture with very small (3x3) convolution filters. This gradually decreasing feature map size using max pooling also became the baseline method while designing convolutional neural networks for image and vision-related tasks.

One of our network architectures is shown in figure 3.2. This architecture is derived from the VGG-16 model with the following modifications:

- Instead of having 13 convolutional layers, we experimented with 8 - 11 layers. The architecture in figure 3.2 has 9 convolutional layers.

- We did not use any fully connected layers but rather applied a global average pooling to the last feature map from convolutions.

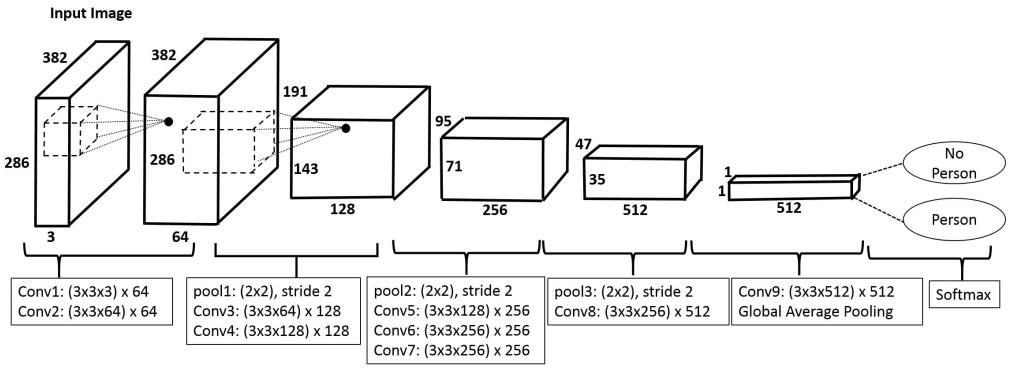


Figure 3.2: The architecture of our classification head.

- We only have 2 classes instead of 1000 categories needed for the ImageNet dataset (Deng *et al.* (2009)).

Note that we used a VGG model, which was pre-trained on the ImageNet dataset. Since we had only two classes, our task was much simpler than the ImageNet dataset having 1000 categories. Therefore, we experimented with less number of layers. Furthermore, we needed to deploy our model to an embedded device (NVIDIA Jetson series). We performed a comparison of accuracy vs. speed on various CNN model sizes, which is discussed later in section 3.5.3.

3.4.2 Global Average Pooling

Fully connected (FC) layers hold most of the parameters in the original VGG-16 model and they are prone to over-fitting. Therefore, we replaced the FC layers with *Global Average Pooling* (GAP) (Lin *et al.* (2013)). The GAP operation is illustrated in figure 3.3. For a given feature volume, GAP takes the average of all values in each channel separately. Therefore, the channels with overall high activations, generate a higher response after a GAP operation. Since it is an average operation, it does not have any learnable parameters. In figure 3.3, a fully connected layer of size m would have $H \times W \times D \times m$ parameters compared to none in GAP. This is a major reduction in parameters and therefore the network avoids over-fitting. In our model, we apply GAP to the last feature map of the convolutions and later apply a fully connected layer to predict the classes.

3.4.3 Class Activation Mapping

The main objective of object detection is to localize objects in the images. In our case, our model should predict bounding boxes around the people. As mentioned before, we apply a weakly supervised model i.e we train our model without bounding box labels but predict the bounding boxes during training. To achieve this, we trained a classification

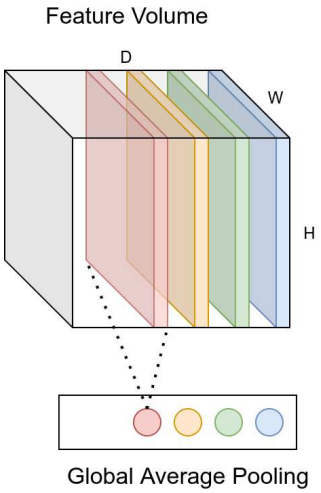


Figure 3.3: The Global Average pooling operation

model and used Class activation mapping (CAM) from Zhou *et al.* (2016) to predict the bounding boxes. Given the feature volume of the last convolutional layer V, the activation map of a particular class can be generated by

$$CAM_c = \sum_{i=0}^{N} w_{ci} F_i, \tag{3.1}$$

where c is the class in question, N is the number of channels in the feature volume V, w_{ci} is the weight connecting the i_{th} GAP feature and the output neuron of class c and F_i is i_{th} channel in V. The process of CAM is illustrated in figure 3.4.

Global average pooling (GAP) plays a critical role in achieving CAM. Each feature after a GAP operation holds the average activation of a particular channel F_i of the feature volume. Since these GAP features are fully connected to the output neurons, the weights of the fully connected layer signify the effect each channel has on that class. Therefore, by multiplying the fully connected weights w_c by the feature maps F, we weigh each feature map by its importance to that particular class. Eventually, a sum of these weighted features generates a heatmap, which highlights highly relevant regions for predicting that class.

3.4.4 Post Processing

To generate bounding boxes from CAM heatmaps, we apply some post-processing steps. First, we resize the heatmap to the original size of the input image. Then, we apply binary thresholding to the heatmap which generates a hard boundary of different objects in the image. Finally, we find contours in the image which gives us the boundary points of each contour in the thresholded image. By finding the minimum and maximum in the x and y

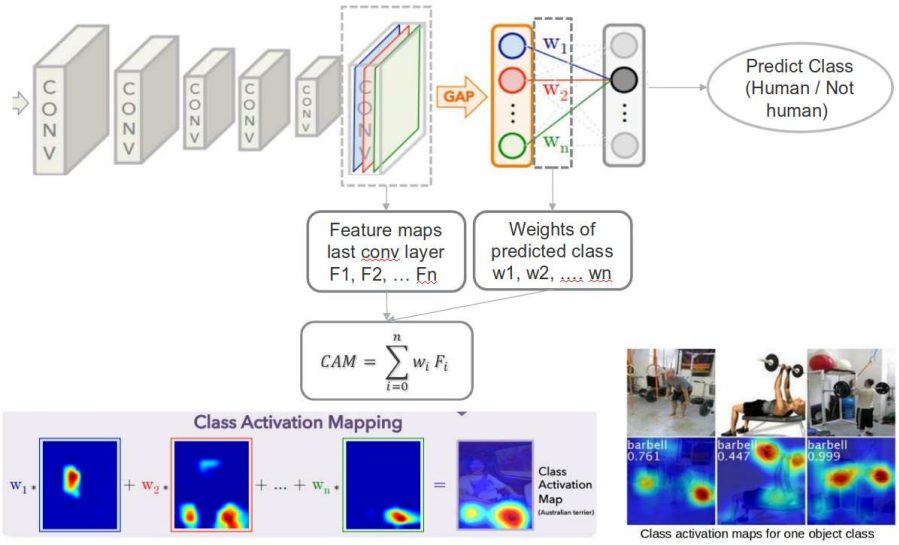


Figure 3.4: The modified version class activation mapping from Zhou *et al.* (2016). Here we use only 2 classes (Human/Not Human)

direction of the image, we find the boxes around the objects. The process is illustrated in figure 3.5

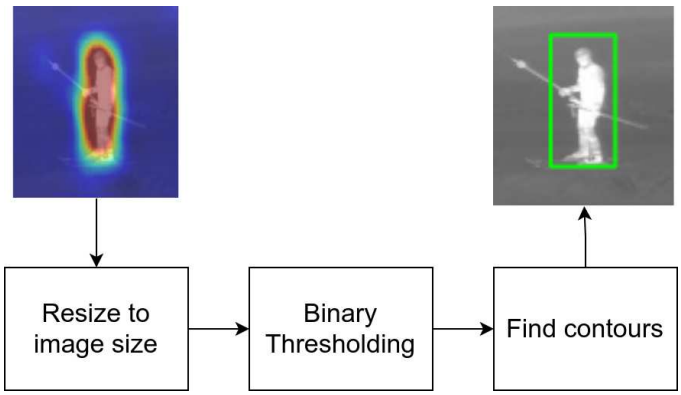


Figure 3.5: The post-processing cycle for bounding box generation.

3.5 Experiments

This section explains the experiments we performed to accomplish the tasks. First, we elaborate on the training methods we employed. Then, we focus on our best qualitative and quantitative results. Finally, we illustrate the different architectures we trained to achieve the best performance on limited hardware.

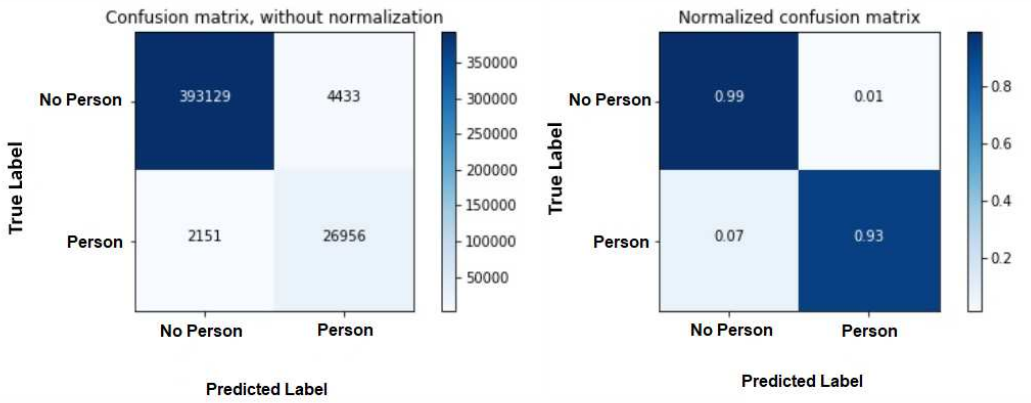


Figure 3.6: Confusion matrices

3.5.1 Training details

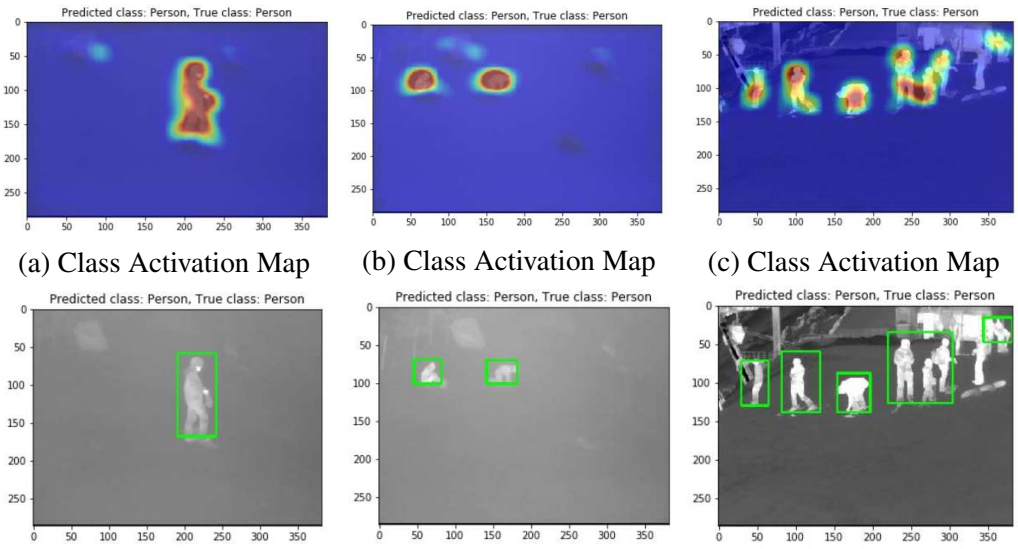
We trained our models on the thermal imaging dataset explained in section 3.3. To achieve better generalization, we applied data augmentation techniques during our training. The most intuitive and effective augmentation in our scenario was random brightness and contrast variation. For each input image, we chose contrast variation, brightness variation, or no change uniformly randomly.

Since we had a high imbalance in our classes (approx 90/10), we used undersampling to avoid this problem. For each epoch, we picked N random samples using a uniform distribution of the 'No person' class, where N is the total number of images of the class 'Person'. Therefore, we had an equal number of both classes in each epoch. In every epoch, we took a new sample to promote diversity in the images. Note that the data is a video sequence and losing so many samples from undersampling does not significantly influence our training because the images have low variance. Strictly speaking, an epoch must contain all samples in the training dataset, but we use only a subset. However, we still use the term epoch in our case to avoid confusion.

We trained our model using binary cross entropy loss with softmax on network output. We used the ADAM optimizer (Kingma and Ba (2014)) as the gradient descent method. We trained on a single GPU (NVIDIA GTX1080Ti) and our code was implemented on the Tensorflow package of Python. We employed a pre-trained model of VGG-16 on ImageNet and fine-tuned our dataset. Note that we only employed pre-trained weights for the layers which were unmodified.

3.5.2 Results

We evaluated our models qualitatively and quantitatively. For quantitative comparison, we used accuracy and confusion matrices as our metrics. Figure 3.6 shows the confusion

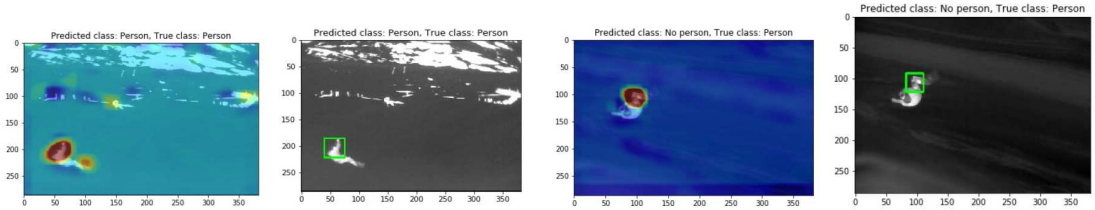


(a) Class Activation Map (b) Class Activation Map (c) Class Activation Map

(d) Bounding Box (e) Bounding Box (f) Bounding Box

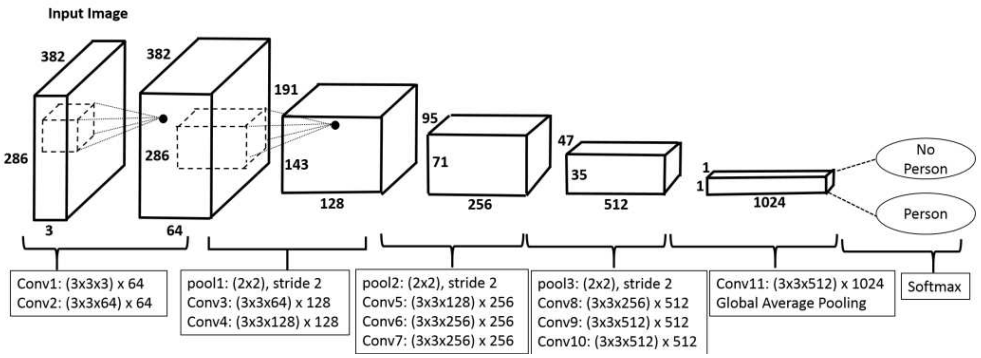
Figure 3.7: Some detections from our model

matrix and its normalized version when the network shown in figure 3.3 was used on our test dataset. The overall accuracy of the model is 98.45%. Since there is an imbalance in the classes, the confusion matrix gives a better estimate of where the errors lie. Among the examples with 'No person', the model predicts with high confidence of above 99%. From the examples, where there are persons, the network predicts it correctly 93% times and fails 7%. This indicates that our system will rarely produce false positives but may miss-detect some people in certain scenarios. A system that is designed to save lives must be conservative (cautious) in its operation and failing to detect people might not be ideal for such as system. Figure 3.7 and 3.8 illustrate some examples of people being detected using our model. It can be seen that upright and fallen people both can be detected using our method. Also, in case of a blurry/hazy environment, the model can detect people reasonably well. In the case of people clustered together, it combines some instances together. This behavior is expected in our model since we do not follow a

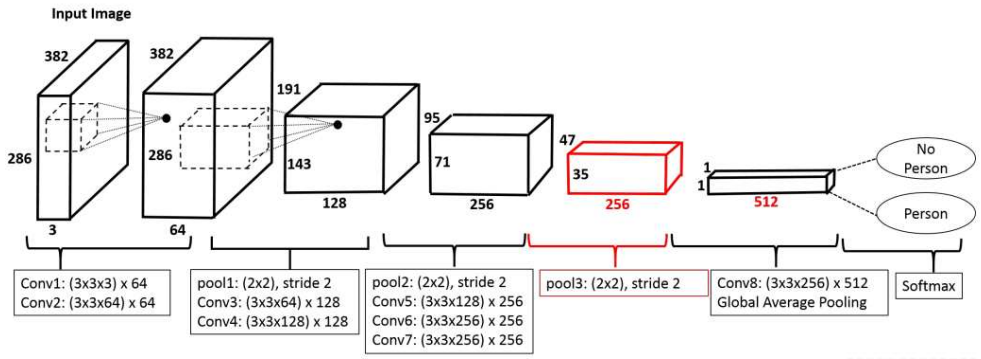


(a) CAM (b) Bbox (c) CAM (d) Bbox

Figure 3.8: Examples of fallen people



(a) 11 Convolutional Layers – 12.3M parameters



(b) 8 Convolutional Layers – 4.6M parameters

Figure 3.9: The structures variation for architecture search

two-stage strategy (find regions of interest, then detect) or divide the output in a spatial grid of detections. We simply find the contours from global information and therefore an overlapping class activation mapping of objects would be regarded as a single object.

3.5.3 Architecture Search

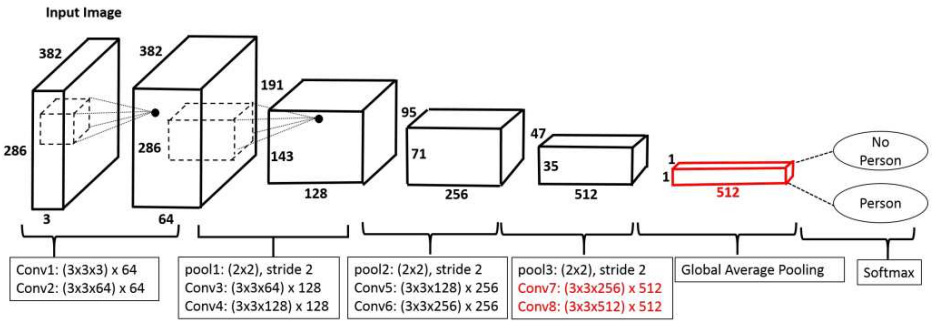
Since one of the tasks was to optimize the models according to the limited hardware available in the snow groomers, we performed an architecture search on our models. We restricted our search to VGG-16 model variants only. We only either removed or modified the final layers of the network and kept the initial layers the same. This allowed us to use the pre-trained feature extractors which is helpful for generalization. The list below contains six of these networks, which were trained on a slightly smaller dataset, and their structures are illustrated in figure 3.9 and 3.10:

1. 11 Convolutional Layers – 12.3M parameters

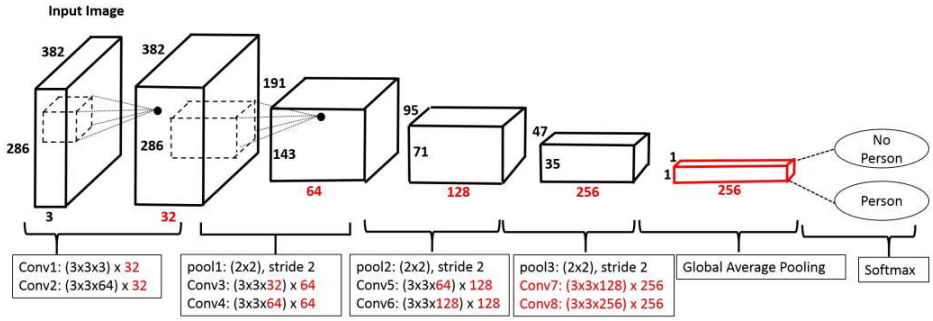
2. 8 Convolutional Layers – 4.6M parameters

3. 8 Convolutional Layers – 2.9M parameters – Structural difference
4. 8 Convolutional Layers – 1.1M parameters – Reduced Filters (1/2 times)
5. 8 Convolutional Layers – 0.2M parameters – Reduced Filters (1/4 times)
6. 11 Convolutional Layers – 12.3M parameters – Input image size halved

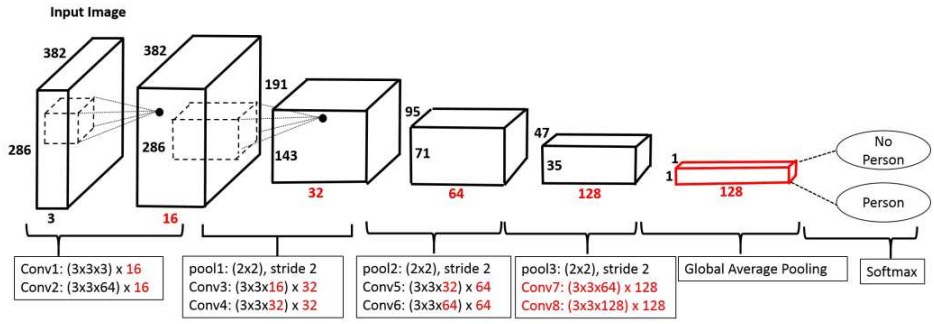
.



(a) 8 Convolutional Layers – 2.9M parameters – Structural difference



(b) 8 Convolutional Layers – 1.1M parameters – Reduced Filters (1/2 times)



(c) 8 Convolutional Layers – 0.2M parameters – Reduced Filters (1/4 times)

Figure 3.10: The structures variation for architecture search

The table 3.2 below shows the results of these networks. The models are evaluated on

Metric	1	2	3	4	5	6
Parameters	12.3M	4.6M	2.9M	1.1M	0.2M	12.3M
Test Accuracy (%)	99.47	98.84	98.09	92.05	89.58	99.27
True Positives	4427	4113	4198	3693	2819	4116
True Positive rate	0.80	0.77	0.79	0.69	0.53	0.77
False Positives	1056	2090	5221	25689	33548	1445
False Positive Rate	0.00	0.01	0.02	0.07	0.09	0.00
Mean IOU	0.39	0.26	0.21	0.00	0.00	0.01
FPS (GTX 1080Ti)	91.0	126	135	238	357	205
FPS (Jetson TX2)	4.32	6.84	7.09	14.9	19.2	12.6
FPS (Jetson TK1)	0.82	1.24	1.26	3.68	10.8	2.84

Table 3.2: The performance of various networks on the test set.

different metrics to highlight the speed-accuracy trade-offs while optimizing networks. We also calculate the mean intersection over union (IOU) of the models to show their localization accuracy. Also, the frames per second are evaluated on various limited hardware platforms.

As a general trend we see that the accuracy decreases as we decrease the number of parameters of the network, which is expected. A similar trend can be observed in true positives and false positives. In the case of mean IOU, it can be seen that the performance drops sharply even with a slight decrease in a number of parameters. This happens, since the networks are trained using only the classification loss (weakly supervised) and in case, we have a limited capacity of the model, the localization suffers the most.

For comparing the speed of our models, we employed three different systems. The first was an NVIDIA GTX1080Ti installed on a desktop PC and the other two were NVIDIA Jetson TX2 and TK1 developmental boards. Jetson TK1 being the oldest proved to be the slowest as expected. It also failed to achieve a reasonable speed (above 5 FPS) on all except the 5th model. However, the accuracy of the 5th model is not acceptable, especially for safety-critical systems.

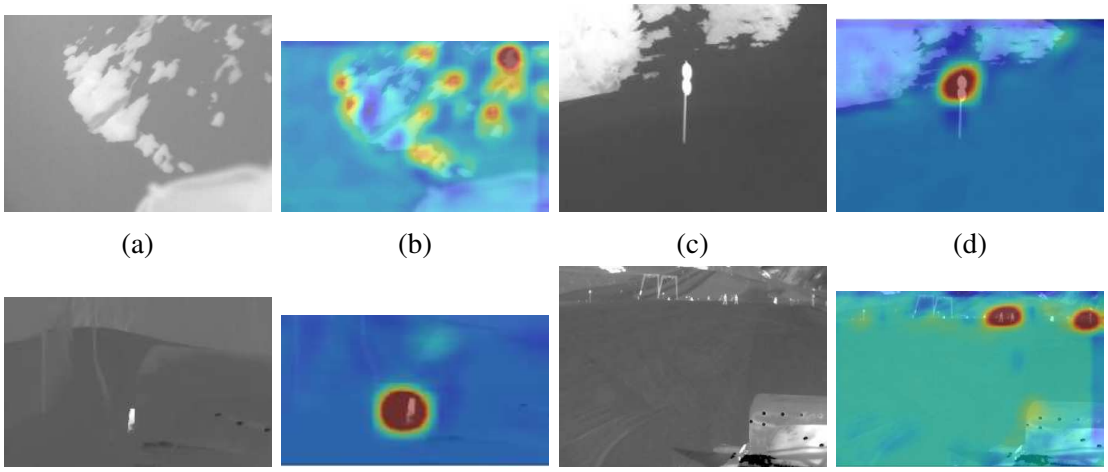
For Jetson TX2, the 3rd, 4th, and 5th models seem to run at a reasonable speed and therefore, these models were eventually tested on the real platform. Naturally, on GTX1080Ti, the speed performance is really high compared to the embedded platforms. However, this cannot be used inside the snow groomer vehicle and is presented here only to show the difference between an embedded platform and a desktop PC.

3.6 Error Analysis

It's important for safety critical systems to analyze the shortcomings of the system. This gives some insights into how to tackle failure cases and further improve the system.

Therefore, in this section, we illustrate the false positives and false negative predictions from our best model and analyze the reasons for the behavior.

Figure 3.11 shows some of the false positive examples from the model. One can observe in figure 3.11a and 3.11b that when there are high-frequency background features like rocks or trees, the model sometimes predicts them as a person. From figure 3.11c and 3.11e, we can deduce that in case of similar features to humans, for example, poles or vehicle parts, the network also may give false positives. One must consider that there is no color information in our images and if an object is radiating thermal radiation and looks similar to humans, it could be detected positively. In figure 3.11g, we can see that small humans are detected, which is wrongly classified as a false positive because it was wrongly labeled.



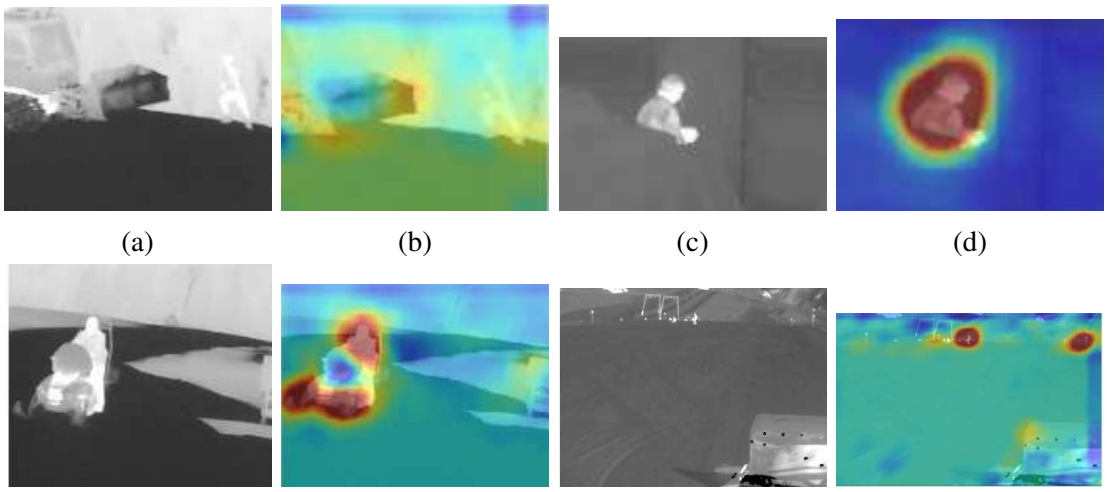
(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.11: Examples of False Positives

Figure 3.12 shows some of the false negatives predicted by our model. Figure 3.12a highlights an instance in which a human is leaning against a wall. Since the relative difference in thermal radiation between the person and the wall is too little, it cannot recognize the person. This is a drawback of using thermal imaging cameras since an RGB image could have easily distinguishable human features if the person is wearing colored clothes.

Figure 3.12c and 3.12e show instances where the human is occluded. Occlusion is a well-studied problem in the literature for object detection models. For both these examples, the class activation maps (figure 3.12d and 3.12f) show that it highlights the human regions correctly, however, since all body parts are not visible, the model predicts a lower confidence, which falls below the threshold and results in a false negative. In the last figure 3.12g, a false negative occurs due to a very small human. This is opposite to the behavior seen in figure 3.11g. In either case, since the human is far away in these images, it's not a safety concern for the actual vehicle.



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

(e) (f) (g) (h)

Figure 3.12: Examples of False Negatives

Chapter 4

Single Stage Instance Segmentation using Fourier Series

This chapter deals with the task of instance segmentation using a single-stage (single-shot) object detection framework. Single-stage methods, as described in section 2.2.1, are simple and fast compared to their counterparts. For bounding box detections, single-stage methods typically predict the box coordinates at each spatial location. However, mask prediction for instance segmentation using single-stage methods is a more complicated problem. We need to predict the masks at all spatial locations in the image in parallel during a single forward pass. Therefore, mask information is normally encoded in a shape vector. This opens new avenues of research in the type of encoding, which could represent the masks most optimally. This chapter focuses on a *Fourier* encoding, which is a compact representation for segmentation masks, and we call our model *FourierNet*.

4.1 Introduction

With the recent emergence of deep learning, combined with readily available data and higher computational power, the use of autonomous machines has become a realistic option in many decision-making processes. In applications such as autonomous driving and robot manipulation, the first and foremost task is to perceive and understand the scene before a decision is made.

Instance segmentation is one of the techniques used for scene understanding (Janai *et al.* (2017)). It categorizes each pixel/region of an image by a specific class and, at the same time, distinguishes different instance occurrences. Among the instance segmentation methods are *two-stage* methods that produce a bounding box and then classify the pixels within that box as foreground or background (He *et al.* (2017); Liu *et al.* (2018); Huang *et al.* (2019); Chen *et al.* (2019b); Lee and Park (2019); Kuo *et al.* (2019)). Although these are still dominant in terms of prediction accuracy, they are computationally intensive.

There is a growing trend to use more straightforward, faster *single-stage* instance segmentation methods that do not require initial bounding box proposals (Bolya *et al.*

Figure 4.1: FourierNet prediction by using **10 coefficients** (20 parameters) of Fourier series. Note that 90 contour points are used to generate this mask.

(2019); Xie *et al.* (2020); Xu *et al.* (2019); Ying *et al.* (2019); Zhou *et al.* (2019a); Yang *et al.* (2019)). In one of the latest approaches, ESE-Seg (Xu *et al.* (2019)) have encoded the objects' contours using function approximations such as Chebychev polynomials and Fourier series. They trained a network to predict a shape vector (a vector of coefficients), in which a numerical transform converts it into contour points in the polar representation. The main advantage of this method is that it requires fewer parameters to represent the mask as opposed to the binary grid or polygon representations (Liang *et al.* (2019)). However, ESE-Seg (Xu *et al.* (2019)) regresses the shape vector directly. We argue that the direct regression of the shape vector does not weigh each coefficient according to its impact on the mask and prevents the model from learning the actual data distribution.

Therefore, we propose an alternative training method in which the network outputs are passed through a *differentiable shape decoder* to obtain contour points that are used to calculate the loss. In this case, the losses of other polygon representation methods, e.g. PolarIOUloss (Xie *et al.* (2020)) and Chamfer loss (Fan *et al.* (2017)), can be used, and the network is trained for its main task. The gradients of these losses are back-propagated through the decoder and the weight balancing of the different shape vector's coefficients is done automatically.

The contribution of this chapter is summarized as follows:

1. We show that the mask information can be encoded in the coefficients of a Fourier series optimally. This compressed representation can be a drop-in replacement for

other single-stage methods.

2. We propose a *differentiable shape decoder* for training, which allows the segmentation masks to be trained directly on the shape of the object instead of regressing the coefficients. This achieves automatic weight balancing of the Fourier coefficients.

3. Finally, we show insights into the meaning of Fourier coefficients. We illustrate that the lower frequency components hold the overall shape information and the higher frequencies contain the fine details (like corners) of the object.

4.2 Related work

This section contains the specific work related to instance segmentation, which is relevant to this chapter. First, we briefly discuss the recent scientific work in two-stage instance segmentation. In the later section, we elaborate on single-stage instance segmentation and the related work to this chapter.

4.2.1 Two stage instance segmentation

Two-stage instance segmentation splits the task into two subtasks, object detection and then segmentation. The most prominent instance segmentation method is Mask R-CNN (He *et al.* (2017)), which is constructed on top of Faster R-CNN (Ren *et al.* (2015)) by adding a mask branch parallel to the bounding box and the classification branches. further, they used RoI-Align instead of RoI-Pooling.

Following on from Mask R-CNN, PANet (Liu *et al.* (2018)) improved the information flow from the backbone to the heads using bottom-up paths in the feature pyramid and adaptive feature pooling. In Mask Scoring R-CNN (Huang *et al.* (2019)), the network estimates the IoU of the predicted mask and uses it to improve the prediction scores. HTC (Chen *et al.* (2019b)) introduced the cascade of masks by merging detection and segmentation features and achieved enhanced detections. ShapeMask (Kuo *et al.* (2019)) introduced class-dependent shape priors and used them as preliminary estimates to obtain the final detection. Instead of building Faster R-CNN, CenterMask (Lee and Park (2019)) built its work on FCOS (Tian *et al.* (2019)) and applied spatial attention for mask generation. The above methods use the binary-grid representation of masks.

In contrast, PolyTransform (Liang *et al.* (2019)) uses a polygon representation and requires a mask for the first stage. The initial mask is refined by a deforming network to obtain the final prediction. These methods accomplish state-of-the-art accuracy, but they are generally slower than one-stage methods.

4.2.2 One stage instance segmentation

YOLACT (Bolya *et al.* (2019)) generated prototype masks and simultaneously produced bounding boxes and combination coefficients. They cropped the prototype masks with the bounding boxes and made a weighted sum of the cropped prototype masks using the combination coefficients to construct the final mask. Likewise, Embedmask (Ying *et al.* (2019)) generated pixel embeddings that differentiate each instance in the image and simultaneously produced bounding boxes and proposal embeddings. Here, they formed the mask by comparing the proposal embedding with all pixel embeddings in the produced bounding box area. In addition to the previous binary-grid representation approaches, there are a few methods that employ polygon representation.

ExtremeNet (Zhou *et al.* (2019a)) used keypoint detection to obtain the extreme points of an object. Then a rough mask was created by forming an octagon from the extreme points. Polarmask (Xie *et al.* (2020)) performed a dense regression of the distances from the mask center to points on the outer contour in polar coordinates. Additionally, since FCOS (Tian *et al.* (2019)) showed that the detections near object boundaries were generally inaccurate, they likewise adopted the concept of centerness, which gave greater importance to the detections near the center and enhanced the prediction quality. ESE-Seg (Xu *et al.* (2019)) trained a network to predict a shape vector that is transformed into contour points in the polar representation. Although their mask representation requires fewer parameters than the other representations, their training method is not optimal, as mentioned in section 4.1. Therefore, we propose employing *differentiable shape decoders* for training, which we explain in the next section.

4.3 Our method

FourierNet is an anchor-free, fully convolutional, single-shot network, and figure 4.2 illustrates its design. Following its backbone, it has a top-down feature pyramid network (FPN) (Lin *et al.* (2017a)) with lateral connections, in which we connect five heads with different spatial resolutions. These heads predict a set of classification scores, centerness, and Fourier coefficients at each spatial location in the feature map. The classification branch predicts scores for each class. Centerness is a term that measures the closeness of a feature point to the mask's center, and we explain it in section 4.3.2. Moreover, we describe the Fourier coefficients in the following mask representation section.

4.3.1 Mask representation

FourierNet uses polygon representation to represent masks. The network generates these polygons by a sequence of contour points, represented by either polar or Cartesian coordinates. The following two sections describe polar and Cartesian representations, respectively.

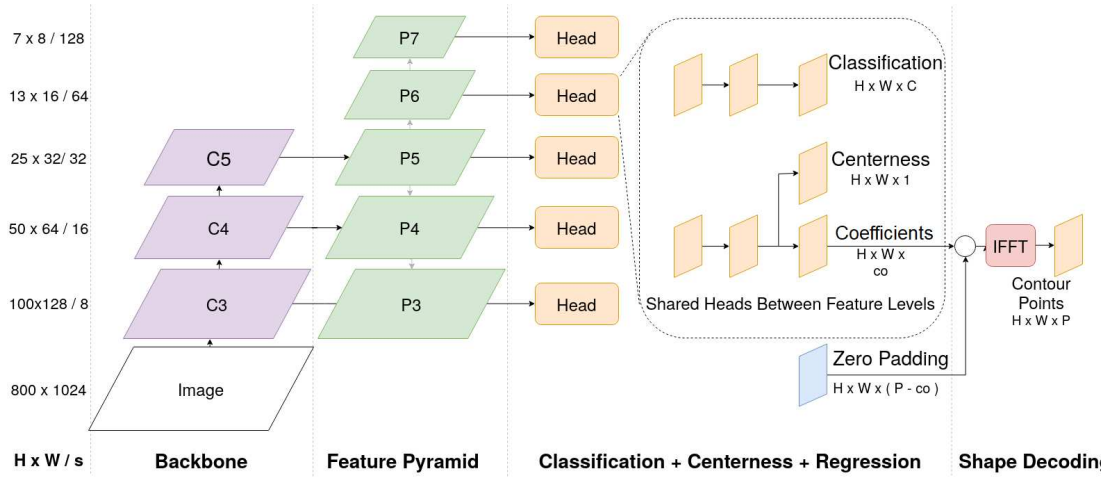


Figure 4.2: The FourierNet architecture. The network head predicts Fourier coefficients, which are transformed to contour points using an Inverse Fast Fourier Transform (IFFT). We zero-pad the coefficients when the number of coefficients are less than the number of contour points. (Note: The head shown in the figure is for polar representation only)

Polar representation

In polar mask representations, for each feature point i near the contour's center, we extend N rays to the point of intersection with the object boundary, as shown in figure 4.3. The angle between the rays $\Delta\theta$ is constant and defined by $360°/N$. The length of these rays from the center point is described by $P_i = \{p_{0,i}, p_{1,i}, ..., p_{N-1,i}\}$. If there is more than one intersection point, the point with the longest distance is selected. Furthermore, a constant $\varepsilon = 10^{-6}$ is assigned to rays that do not have intersection points, which occurs when the feature point i is outside or on the contour's boundary. Note that the emerged contour would only approximate the ground truth contour even with a high number of rays; however, the IOU values can reach up to 0.95 (Xie *et al.* (2020)).

To determine P_i, we apply an *Inverse Fast Fourier Transform* (IFFT) to the coefficients predicted by the network (figure 4.2). The inverse discrete Fourier transform is defined by

$$p_{n,i} = \frac{1}{N} \sum_{k=0}^{N-1} x_{k,i} e^{\frac{j2\pi kn}{N}}, \tag{4.1}$$

where $p_{n,i}$ is the n_{th} ray in P_i and $x_{k,i}$ is the k_{th} coefficient of X_i, which is the Fourier transform of P_i. In cases where we predict more rays than Fourier coefficients, the network predicts a subset of the coefficients $S_i \subset X_i$ and then we replace the rest of the output tensor (higher frequency components) with zeros. This is done to equalize the dimensions before and after the IFFT. Note that the IFFT is differentiable and therefore the training is done directly on rays (P_i) and thus justifies the name *differentiable shape decoder*.

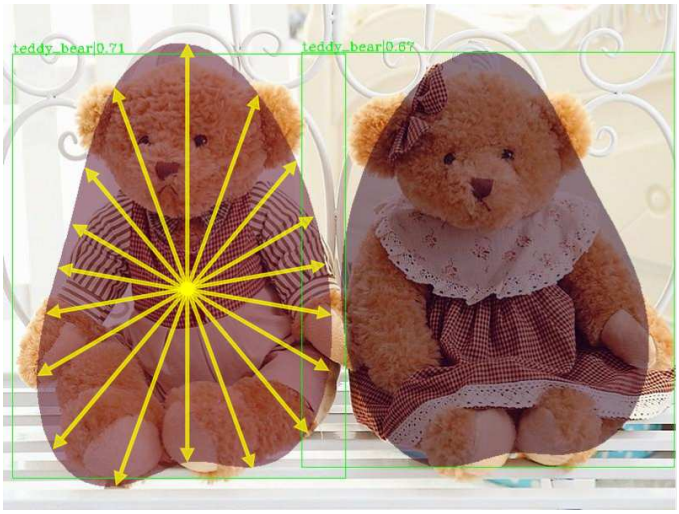


Figure 4.3: On the left object, **18 rays** are extended by the lengths P_i from a feature point i (a potential center point). The contour points are the endpoints of these rays. The ground truth center point is the mean value of the contour points. Note that this figure is simplified for illustration purposes. The actual mask generated in this image has 90 contour points

Cartesian representation

Polar representations generate star-shaped masks, since, for each angle, there is only one possible ray length. To represent arbitrary masks, we can use Cartesian coordinates. For Cartesian representations, we modified the FourierNet head to predict the x and y (Cartesian) coordinates of each contour point of the mask rather than the ray lengths (Polar). Figure 4.4 shows the modified structure of the FourierNet head for Cartesian representations. Since x and y are independent entities, two separate branches of Fourier coefficients are utilized instead of a single one. An IFFT is applied to each of these branches separately. For the Cartesian case, the $p_{n,i}$ in equation 4.1 refers to the distance of the n_{th} contour point from the i_{th} feature point, in either x or y directions. For cases where contour points are more than Fourier coefficients, we pad the output tensor with zeros as in polar representation.

4.3.2 Centerness

Centerness is a term that measures the closeness of a feature point to the center of a mask. In many grid-based detection methods (Tian *et al.* (2019), Zhou *et al.* (2019b)), a center point is predicted for each object. However, multiple center points can predict high confidence for the same object due to close proximity. (Tian *et al.* (2019)) utilizes centerness to filter out weak detections during inference. We utilize *polar centerness* in the case of polar representation and *Gaussian centerness* in the case of Cartesian

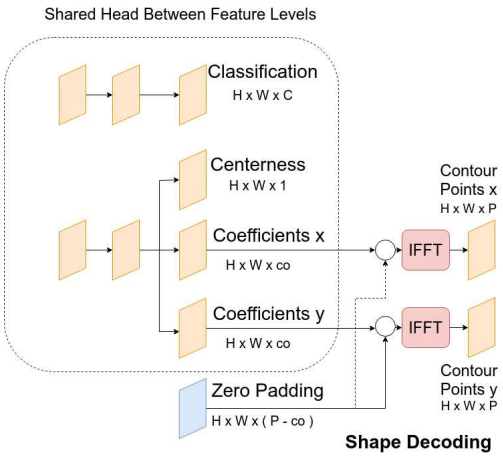


Figure 4.4: The FourierNet head for Cartesian representation. We predict separate coefficients for both x and y coordinates.

representation. Both are detailed in the following sections, respectively.

Polar centerness

Polar Centerness (PC) (Xie *et al.* (2020)) is defined for the i_{th} feature point as

$$PC_i = \sqrt{\frac{\min(p_{0,i}, p_{1,i}, \ldots, p_{N-1,i})}{\max(p_{0,i}, p_{1,i}, \ldots, p_{N-1,i})}}, \tag{4.2}$$

where $p_{n,i}$ are the ray lengths. During inference, we multiply this value with the classification score to keep the locations which could produce the best detection.

We argue that this metric would be low if the object's mask shape is not circular, and since we multiply it by the classification score, it will lower the probability of predicting such objects. PolarMask introduced a hyperparameter called *Centerness Factor* (CF) to overcome this problem, which is added to the centerness to increase its value. To the best of our knowledge, this offset defeats the purpose of centerness, since it artificially raises confidence and sometimes even exceeds 1. Moreover, it does not explicitly solve the problem of low centerness of non-circular objects. Therefore, we introduce *Normalized Centerness* (*NC*) which is defined for a feature point *i* by

$$NC_i = \frac{PC_i}{PC_{\max}}, \tag{4.3}$$

where PC_{max} is the polar centerness of the *center of mass* of an instance. The maximum value of the NC_i is clamped to one when the center of mass does not have the highest polar centerness value.

Gaussian Centerness

In the case of centerness for Cartesian representation, we can not adopt equation 4.2 directly. Accordingly, we apply a Gaussian distribution to represent the probability of a point being at the object's center. For the ith feature point having the location (m,n) in the feature map, Gaussian centerness (GC) is defined as

$$GC = e^{-\alpha(\frac{m-\mu_x}{\sigma_x})^2} e^{-\alpha(\frac{n-\mu_y}{\sigma_y})^2}, \tag{4.4}$$

where μ_x and μ_y are the means (center points), and σ_x and σ_y are the standard deviations of a mask instance in x and y directions respectively and α is a hyperparameter used for controlling the decaying rate. Note that we multiply the two Gaussians, which enforces a probability of 1 only if both m and n are at the object's center. On all the other locations, the decaying functions' product reduces the centerness depending upon the standard deviation in both x and y directions of the mask instance. Notice that GC solves the problem of low centerness for non-circular objects, and therefore the centerness factor can be completely avoided.

4.3.3 Loss Function

The overall loss function comprises four components, which are defined as:

$$L_{total} = L_{cls} + L_{box} + L_{cent} + L_{mask}. \tag{4.5}$$

We use *focal loss* (Lin *et al.* (2017b)) for the classification loss L_{cls} and *IOU loss* (Yu *et al.* (2016)) for the bounding box loss L_{box}. Note that the bounding box branch is an optional branch and therefore not explicitly shown in figure 4.2. For centerness loss L_{cent}, we employ *binary cross entropy* for both *Polar centerness* and *Gaussian Centerness*. For mask loss L_{mask}, we utilize two different loss functions for polar and Cartesian representations. For polar representations, we adopt *Polar IOU loss* from (Xie *et al.* (2020)). For Cartesian representations, we employ both *smooth L1* loss and *chamfer distance* loss (Fan *et al.* (2017)). In the following section, chamfer distance loss has been explained.

Chamfer distance loss

To train the (x, y) contour points, chamfer distance loss (Fan *et al.* (2017)) is adopted. It is defined as

$$CD = \sum_{a \in S_1} \min_{b \in S_2} \parallel D(a,b) \parallel_2^2 + \sum_{b \in S_2} \min_{a \in S_1} \parallel D(a,b) \parallel_2^2, \tag{4.6}$$

where S_1 and S_2 are the sets of predicted contour points and ground truth contour points respectively, a and b are elements (individual contour points (x,y)) of the sets S_1 and S_2 respectively and $D(a,b)$ is the euclidean distance between any two points a and b respectively. Please note that the centroid of the object is taken as a reference for the contour

points. We normalize the chamfer distance by dividing it by the average of height and width of the ground truth bounding box. Without the normalization, chamfer distance becomes exceptionally large, which leads to overflows and exploding gradients. Moreover, normalization avoids the problem of manually weighing classification, centerness, and mask losses.

Chamfer losses employ the nearest neighbor approach to pick the associations between predicted and ground truth contour points. A naive approach would be to associate predictions and targets index-wise. Since the indices of predictions do not always start from the same angle (or relative physical location), there is a potential for a mismatch between predictions and target associations. Due to this offset, the model might ignore the edges and learn the average mask (i.e., an ellipse). Since chamfer loss utilizes the nearest neighbor for associations between prediction and target, it overcomes this problem.

The nearest-neighbor approach also poses a risk that only the closest points are trained. For example, if we take the nearest neighbors of all the predicted points only, some significant target points are overlooked while training, and therefore complicated contours may never be predicted. Similarly, if we take the nearest neighbors of all the target points only, then errors do not back-propagate through all prediction points and thus may generate an uneven distribution of predicted contour points. Therefore nearest neighbors of both predictions and targets must be considered separately, as done in chamfer distance.

4.3.4 Weakly supervised training

One of the main challenges in segmentation is the effort in labeling masks or contours. Especially, compared to image-level or bounding box labels, segmentation labels require multi-folds of labeling hours and computing resources. For example, labeling an object with 60 contour points may require 30 times as much time as a bounding box label (2 points only). Therefore, the scientific community is continuously looking for ways to either generate labels with minimal effort or train machine learning models without true labels.

Weakly supervised methods make use of inexact information to train a model. For instance segmentation, either image-level classification labels or bounding box labels are employed to train models, which predict segmentation masks for each object in the image. This processes either utilizes the information a model learns inherently for a trivial task to generate much more complex output (Zhou *et al.* (2018)), or they use readily available algorithms to generate pseudo labels from weak labels and train the networks on these pseudo labels.

GrabCut algorithm

In our work, we use the GrabCut algorithm (Rother *et al.* (2004)) to generate pseudo segmentation labels using the bounding box labels in the MS COCO dataset. The GrabCut algorithm works on the idea of graph cuts. Using a Gaussian Mixture Model (GMM),

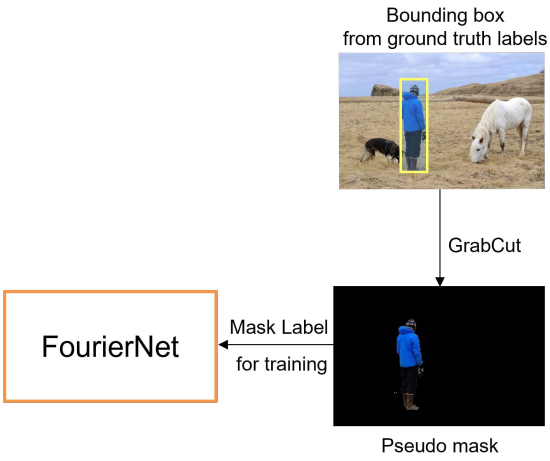


Figure 4.5: The process of generating pseudo mask labels for training our model. It uses GranCut (Rother *et al.* (2004)) algorithm, which has been briefly explained in section 4.3.4

the GrabCut creates a color distribution in the bounding box of the object. It employs a Morkov random field with an energy function that enforces connected regions to have the same category. Figure 4.5 illustrates the process of generating pseudo mask for training our model.

4.4 Experiments

We conducted the experiments on the COCO 2017 benchmark (Lin *et al.* (2014)) divided into 118K training and 5K validation data splits. We based our work on PolarMask (Xie *et al.* (2020)) implementation, which uses the mmdetection framework (Chen *et al.* (2019a)). Unless otherwise stated, we did all the experiments using a pre-trained ResNet-50 (He *et al.* (2015)) on the ImageNet (Deng *et al.* (2009)). We trained the networks for 12 epochs with an initial learning rate of 0.01 and a mini-batch of 4 images. The learning rate was reduced by a factor of 10 at epochs 8 and 11. We used Stochastic gradient descent (SGD) with momentum (0.9) and weight decay (0.0001) for optimization. We resized the input images to 1280×768 pixels.

4.4.1 Cartesian representation vs Polar representation

Table 4.1 shows a comparison between various networks trained on cartesian representations using smooth L1 loss and chamfer distance loss. The training setup and hyper-parameters are the same as described above. The network with smooth L1 loss performs worst as its index-wise associations make the masks ellipse-like (as discussed in section

Coeff.	Loss mask	Loss centerness	mAP
8	Smooth L1	Gaussian	13.6
36	Smooth L1	Gaussian	13.5
8	Chamfer	Gaussian	22.9
36	Chamfer	Gaussian	22.4
36	Polar IOU	Polar	28.0

Table 4.1: Comparison of mAP for the cartesian and polar representations. Clearly, the polar representation (with Polar IOU loss) performs significantly better than the Cartesian representation (Chamfer or Smooth L1 loss). Among the cartesian representations, the chamfer loss performs better than the smooth L1 loss because of the nearest neighbor approach in contour point associations.

4.3.3). Figures 4.7a and 4.7b show that the visual difference in masks when using 8 or 36 coefficients is also insignificant.

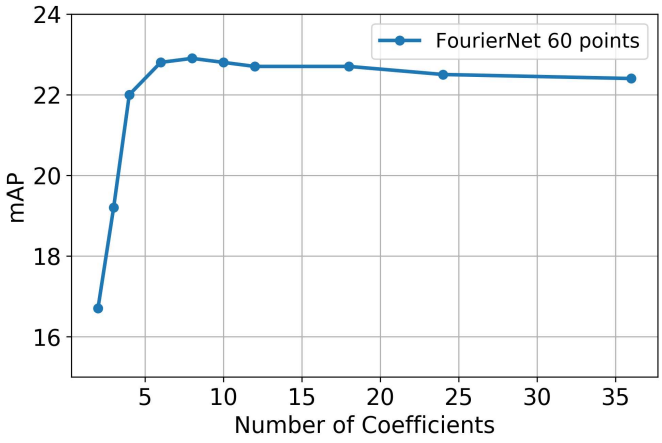
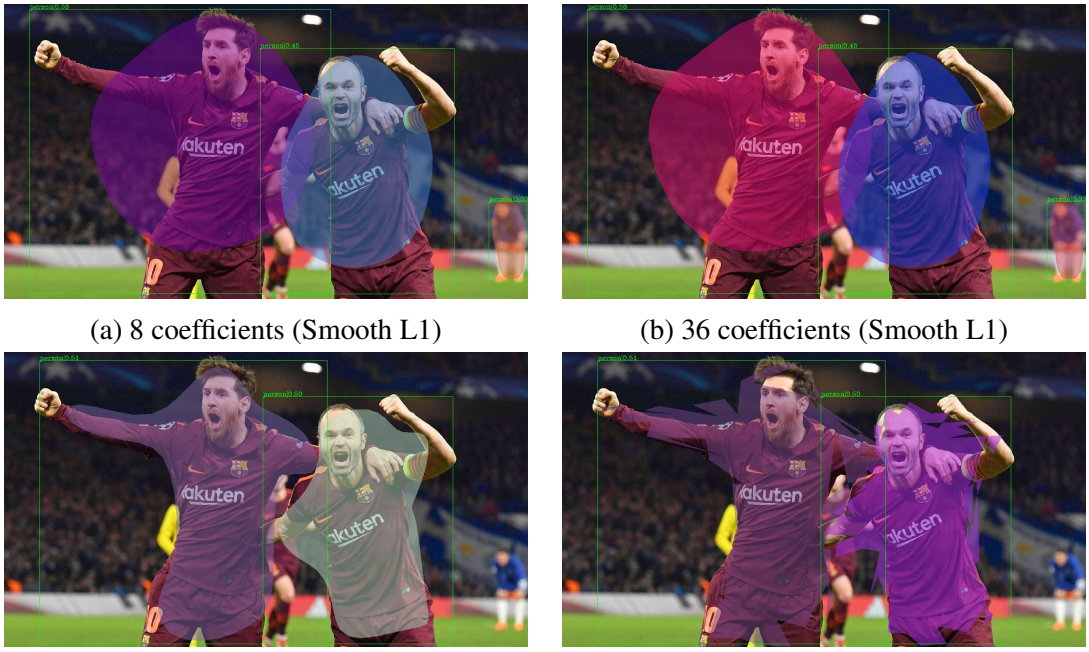


Figure 4.6: Evolution of performance of FourierNet with changing coefficients in cartesian representation (Chamfer loss). The maximum performance is seen at 8 coefficients (22.9 mAP).

The network trained with the Chamfer distance loss was first pre-trained for one epoch on smooth L1 loss as a warm-up. This provides a good initialization since chamfer loss greatly benefits from elliptical predictions at the start. It shows the best performance with 22.9 mAP in the Cartesian domain but still falls short of polar representation (28.0 mAP). In all the experiments, $\alpha = 10$ is used, which provides a reasonable balance between a high probability for a point at the center of the object and low values at the mask edges.

Figure 4.6 shows the evolution of mAP with respect to the number of coefficients of the Fourier series. It can be seen that the maximum mAP is reached when using 8 coefficients only. Figure 4.7 illustrates the masks generated by the network using



(a) 8 coefficients (Smooth L1)

(b) 36 coefficients (Smooth L1)

(c) 8 coefficients (Chamfer distance)

(d) 36 coefficients (Chamfer distance)

Figure 4.7: Examples of the network using cartesian coordinates. Sub-figure 4.7a and 4.7b are trained using smooth L1 loss. Sub-figure 4.7c and 4.7d are trained using chamfer distance loss. The quality of results from chamfer distance loss is clearly better than the smooth L1 loss.

cartesian representation. Counter-intuitively, the masks with 8 coefficients are smoother and have a better IOU than the masks with 36 coefficients. As seen in figure 4.7d, when using 36 coefficients, there are undesired oscillations that make the contour worse.

One possible reason could be that the gradients are very low for the higher frequency coefficients during training because they affect the output very little. This eventually leads to under-trained higher frequency coefficients which show erratic results when visualized. However, this hypothesis needs further investigation, which is not part of this work. Since polar representation showed better performance, in the rest of the work, polar representation has been employed.

4.4.2 Ablation study

All the experiments done in this ablation study section adopt polar representations for masks. We employ a ResNet-50 backbone for our model and use the settings described at the start of this section 4.4.

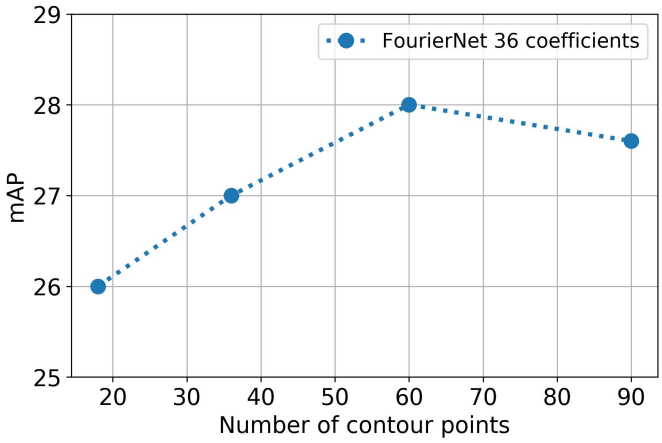


Figure 4.8: The relation of mAP with a varying number of contour points. The maximum performance (28.0 mAP) is achieved with 60 points. The FourierNet in this experiment has 36 coefficients of Fourier series and a Resnet-50 backbone.

Number of contour points

We trained multiple FourierNets having 18, 36, 60, and 90 contour points and 36 complex coefficients (72 parameters). Figure 4.8 shows that more contour points generally lead to a higher mAP until it saturates, and then the performance deteriorates.

The FourierNet with 90 points has a lower mAP (27.6) than FourierNet with 60 points (28.0). A possible reason could be that the added complexity (in terms of contour points) makes the problem harder for the optimizer to learn. Furthermore, while more contour points seem more appealing for large and complicated masks, for smaller objects, it means adding unwanted complexity, which could lower AP_s if not learned correctly, leading to an overall negative effect on performance.

Number of coefficients

From the results of the previous section 4.4.2, we choose a FourierNet with 60 contour points for this study. Figure 4.9 illustrates the progression of the accuracy of the FourierNet for a varying number of parameters. We generated the curve by testing the network multiple times. Each point in the curve refers to a test where we use a subset of the network output tensor with the lowest frequency coefficients. The rest of the higher frequency coefficients were replaced with zeros to inhibit their effects (explained in section 4.3.1).

As the number of parameters increases, the mAP sharply increases until around 18 parameters, and after 36 parameters, it saturates. We also observed that the FourierNets with 18, 36, and 90 points showed the same trend of this curve as FourierNet with 60 points, so we did not plot them. Furthermore, We visualize in figure 4.10 the network

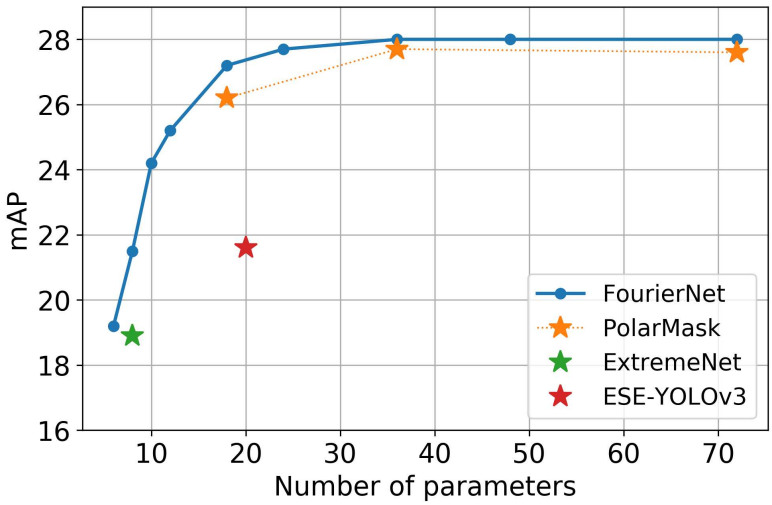


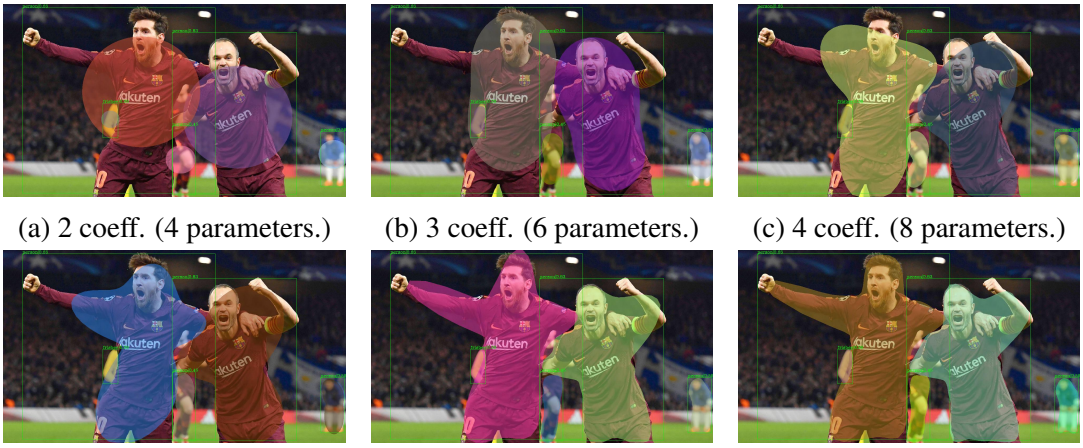
Figure 4.9: The FourierNet in this experiment has a **Resnet-50** backbone and **60 contour points**. The number of parameters is the complex (real+imaginary) coefficients of the Fourier series i.e 36 coefficients = 72 parameters. The Polarmask used in this experiment has the same backbone as well.

outputs with a different number of suppressed higher frequencies. We obtain smooth contours if we use a low number of coefficients, and when we utilize only two coefficients, all the predictions become ellipses. For 36 coefficients (figure 4.10f), we acquire a reasonably good prediction, with some limitations in the non-convex regions of the mask due to polar representation.

Note that the Fourier series achieve compression, since each frequency component individually learns to fit the contour to the ground truth mask according to its capacity. Therefore, users could choose the number of frequencies that fit their use case and achieve a compromise between speed and performance.

Coefficients regression (CR) vs. Differentiable shape decoding (DSD)

The lower frequency coefficients of a Fourier series have a higher impact on the contour, which can be inferred from the experiments in section 4.4.2. However, unweighted direct coefficient regression focuses equally on all coefficients during training, which is not optimal for shape decoders. On the contrary, when trained on contour points, the optimizer can inherently learn to prioritize the lower frequency coefficients of the Fourier series and achieve *automatic weight balancing*. To verify this hypothesis, we trained a network with 18 coefficients and regressed the coefficients directly using a smooth L1 loss. It attained an mAP of 5.3 (table 4.2), which is poor compared to a similar network trained on contour points (26 mAP from figure 4.9) and it validates our initial intuition. Moreover, the qualitative results of CR showed out-of-size masks which is a sign of



(a) 2 coeff. (4 parameters.) (b) 3 coeff. (6 parameters.) (c) 4 coeff. (8 parameters.)

(d) 6 coeff. (12 parameters.) (e) 12 coeff. (24 parameters.) (f) 36 coeff. (72 parameters.)

Figure 4.10: Comparison between the predicted mask of using a varying number of Fourier coefficients using polar representation. As the number of coefficients increases, the mask fits more closely to the boundary of the persons. Note that polar representations can only predict star-shaped masks and we can observe this limitation in the right player's arm.

Method	mAP	AP_{50}	AP_{75}
Coefficient Regression	5.3	14.9	3.1
Differentiable Shape Decoding	26	46.6	25.8

Table 4.2: Coefficients regression (CR) vs. Differentiable shape decoding (DSD)

errors in low-frequency coefficients.

ESE-Seg (Xu *et al.* (2019)) also used CR and compared various function approximators. They reported the best performance on Chebyshev polynomials and argued that they have the best numerical distribution. However, we argue that if they had used optimized weights for Fourier coefficients during training, they would have reached better performance. This was verified with our results using DSD because it does automatic weight balancing.

Method	CF	mAP	AP_{50}	AP_{75}
Polar	0	26.3	42.8	**27.7**
Normalized	0	**27.0**	**47.8**	26.9
Polar	0.5	**27.7**	46.4	**28.6**
Normalized	0.5	27.0	**47.9**	26.9

Table 4.3: Polar centerness vs. Normalized polar centerness

Polar centerness (PC) vs. Normalized centerness (NC)

Two networks with 90 contour points and 36 coefficients were trained on Normalized Centerness (NC) and Polar Centerness (PC). From the results in table 4.3, it can be seen that NC is better than PC when the CF is set to zero, which means that it is generally a better centerness metric. Normalized centerness seems to be unaffected by changing CF and therefore is a more stable method. However, to obtain the best performance (27.7 mAP), we still need to use the CF hyperparameter (CF=0.5) along with Polar Centerness.

4.4.3 Comparison to other state-of-the-art methods

A FourierNet-640 was trained with an image resolution of 640 x 360 to compare with a ESE-Seg-416 (Xu *et al.* (2019)). With a comparable backbone and the same number

Method	B.Bone	Rep.	Param.	mAP	AP_{50}	AP_{75}	FPS	GPU
Mask RCNN He *et al.* (2017)	RX-101	BG	784	37.1	60.0	38.4	5.6	1080Ti
PANet Liu *et al.* (2018)	RX-101	BG	784	**42.0**	**65.1**	**45.7**	-	-
HTC Chen *et al.* (2019b)	RX-101	BG	784	41.2	63.9	44.7	2.1	TitanXp
ESE-Seg-416 Xu *et al.* (2019)	DN-53	SE	20	21.6	**48.7**	22.4	**38.5**	1080Ti
FourierNet-640	R-50	SE	20	**24.3**	42.9	**24.4**	26.6	2080Ti
ExtremeNet Zhou *et al.* (2019a)	HG-104	P	**8**	18.9	44.5	13.7	3.1	-
FourierNet	RX-101	SE	**8**	**23.3**	**46.7**	**21.1**	**6.9**	2080Ti
EmbedMask Ying *et al.* (2019)	R-101	BG	†	**37.7**	**59.1**	**40.3**	13.7	V100
YOLACT-700 Bolya *et al.* (2019)	R-101	BG	†	31.2	50.6	32.8	**23.4**	TitanXp
PolarMask Xie *et al.* (2020)	RX-101	P	36	32.9	55.4	33.8	7.1*	2080Ti
FourierNet	RX-101	SE	36	30.6	50.8	31.8	6.9	2080Ti

Table 4.4: Comparison with state-of-the-art for instance segmentation on COCO test-dev. The methods above the double line are two stage methods and the methods below are faster one stage methods. † The number of parameters is dependent on the size of the bounding box cropping the pixel embedding or mask prototype. * speed tested on our machines. BG: Binary Grid. SE: Shape Encoding. P: Polygon.

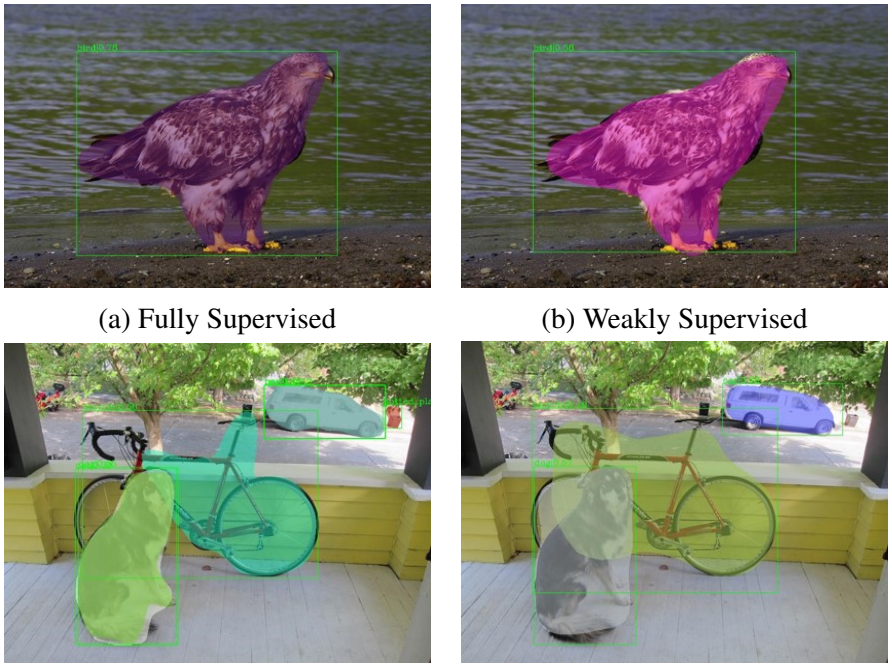
of parameters, our result is 2.7 mAP higher and it runs in real-time. To compare to state-

of-the-art methods, a FourierNet with a ResNeXt101 backbone (Xie *et al.* (2017)), 90 contour points, and 36 coefficients was trained. The quantitative results are shown in table 4.4 and an example of a prediction is shown in figure 4.10.

Compared to ExtremeNet (Zhou *et al.* (2019a)), using 8 parameters, our results are better, especially with AP_{75}, with an increase of 7.4. This means that our mask quality is superior when using a few parameters. It can be seen that FourierNet is comparable to PolarMask when using the same number of parameters, with a small loss in speed due to the IFFT. However, the qualitative results are visually better with smoother contours. In general, our method is comparable to polygon methods but falls short in performance compared to binary grid methods.

4.4.4 Weakly supervised instance segmentation

As described in section 4.3.4, we used the Grabcut (Rother *et al.* (2004)) algorithm to generate pseudo labels and train our network on these labels. We used a ResNet-50 backbone with 36 Fourier coefficients in our model. All other settings were the same as the experiments above. We reached a mean average precision of 18.8 on the COCO validation dataset. The qualitative results are shown in figure 4.11.



(a) Fully Supervised (b) Weakly Supervised

(c) Fully Supervised (d) Weakly Supervised

Figure 4.11

We can see in the figures that weakly supervised results are slightly worse than the fully supervised results, which is expected due to noisy ground truth labels. A weakly

supervised network does not predict high-frequency details in the masks. It must be noted that in occluded scenarios and in cases where two objects overlap, GrabCut fails to generate a reasonable segmentation mask. This happens because it considers all objects inside a bounding box as foreground. Therefore, with noisy pseudo labels, such a result is expected.

4.5 Conclusion

FourierNet is a single-stage anchor-free method for instance segmentation. It uses a novel training technique with IFFT as a differentiable shape decoder. Moreover, since lower frequencies impact the mask the most, we obtained a compact representation of masks using only those low frequencies. Therefore, FourierNet outperformed all methods which use less than 20 parameters quantitatively and qualitatively. Figure 4.12 shows an example of qualitative comparison between our method and ESE-Seg (Xu *et al.* (2019)). Even compared to object detectors, FourierNet can yield better approxima-

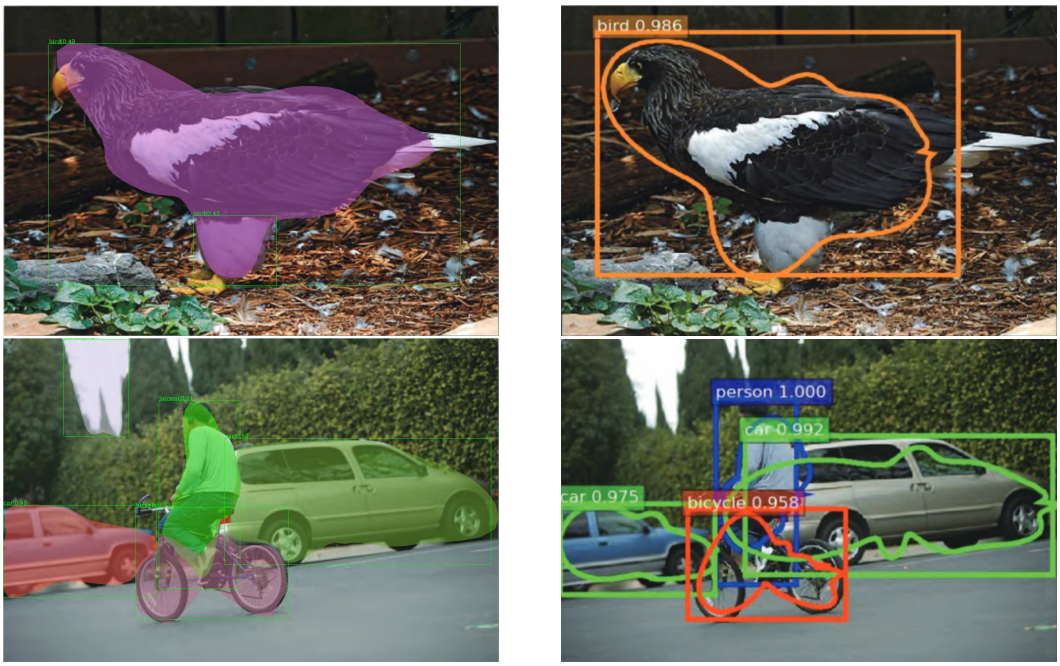


Figure 4.12: Qualitative comparison between masks prediction. Left: FourierNet, Right: ESE-Seg (Xu *et al.* (2019)).

tions of objects using slightly more parameters. Our FourierNet-640 achieves a real-time speed of 26.6 FPS (NVIDIA RTX2080Ti GPU). We hope this method can inspire the use of differentiable decoders in other applications. Figure 4.13 shows some examples from FourierNet.

Figure 4.13: Examples of qualitative results from FourierNet (ResNeXT-101 backbone)

Chapter 5

Fourier series in coordinate-based multi-layer perceptrons

This chapter focuses on some insights into the idea of employing the Fourier series in coordinate-based multi-layer perceptrons (MLPs). We describe how our Fourier mapping of the input could help coordinate-based MLPs, in learning high-frequency details in the signal better. We show this on the task of image regression in this chapter. Note that this chapter forms the basis for our work on instance segmentation in the next chapter. Also, we use the term Implicit Neural representation (INR) and coordinate-based MLP interchangeably, depending on the context of the discussion.

5.1 Introduction

Implicit neural representations (INRs) are a novel field of research in which the traditional discrete signal representation (e.g., images as discrete grids of pixels, 3D shapes as voxel grids or meshes) are replaced with continuous functions that map the input domain of the signal (e.g., coordinates of a specific pixel in the image) to a representation of color, occupancy or density at the input location. However, these functions typically are not analytically tractable; INRs approximate those functions with fully connected neural networks (also called multi-layer perceptrons (MLPs)).

INRs are not coupled to the spatial resolution (e.g., voxel size in a 3D scene) and theoretically have infinite resolution. Therefore, these representations are naturally suited to applications with high-dimensional signals and heavy memory consumption. Also, since they are differentiable, they are suitable for gradient-based optimization and machine learning. In addition, the application of INRs for images (Henzler *et al.* (2020); Stanley (2007)), volume density (Mildenhall *et al.* (2020)), and occupancy (Mescheder *et al.* (2019)) enhanced the performance on various tasks such as shape representation (Chen and Zhang (2019); Deng *et al.* (2020); Genova *et al.* (2019, 2020); Jiang *et al.* (2020); Michalkiewicz *et al.* (2019); Park *et al.* (2019)), texture synthesis (Henzler *et al.* (2020); Oechsle *et al.* (2019a)), and shape inference from images (Liu *et al.* (2020, 2019)).

However, early architectures lacked accuracy in high-frequency details. Sitzmann

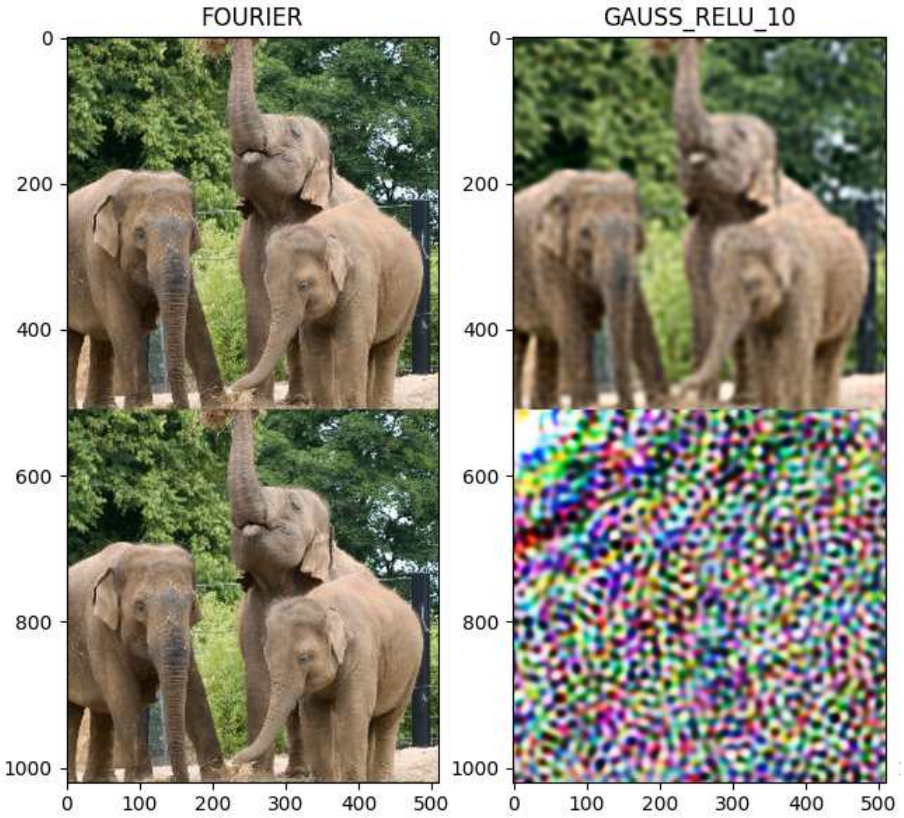


Figure 5.1: Visualization of Gauss ReLU (Tancik *et al.* (2020)) and our method (Fourier). The top row is the first period and the bottom row is the second period, which shows our method of enforcing periodicity.

et al. (2020b) proposed SIRENs, which could represent high frequencies. They argued that sinusoidal activations work better than ReLU networks because ReLU networks are piecewise linear, and their second derivative is zero. As a result, they are incapable of modeling data contained in higher-order derivatives of signals. However, a concurrent work (Mildenhall *et al.* (2020)) proposed positional encoding, which also enabled the networks to learn high-frequency information. The positional encoding uses a heuristic sinusoidal mapping to input coordinates before passing them through a ReLU network. They did follow-up work (Tancik *et al.* (2020)) exploring the general Fourier mapping and explaining why it worked using a Neural Tangent Kernel (NTK) framework (Jacot *et al.* (2018)). They found out that the Fourier mapping transforms the NTK into a shift-invariant kernel. And modifying the mapping parameters enables tuning the NTK's spectrum, therefore controlling the range of frequencies the network can learn. They also showed that a random Fourier mapping with low standard deviation learns only low frequencies of the signal. On the contrary, a high standard deviation lets the network

learn high frequencies only, which leads to over-fitting. They recommended a linear search to find the optimal value of the standard deviation for the corresponding task. They also showed that increasing the number of parameters in the mapping improves the performance constantly.

The recent works in implicit neural representations generate further questions, which need deeper investigation. For example, the connection between SIRENs and Fourier mapping has not been researched yet. Also, the effect of the number of mapping parameters and type of Fourier mapping (random) has not been fully studied. Moreover, techniques for better generalization and avoiding over-fitting need more attention.

To investigate these questions, we explored the mathematical connection between Fourier mappings and SIRENs and showed that a Fourier-mapped perceptron is structurally like one hidden layer SIREN. However, in the SIREN case, the mapping is trainable, and it is represented in the amplitude-phase form instead of the sine-cosine form in the case of Fourier mappings.

Also, we looked at the functions we want to learn, and we observed that they have a limited input domain (e.g., the height and width of an image), and their values are defined on a finite set. Hence, we can assume that they are continuous and periodic over their input bound, which satisfies all the requirements to represent them with a Fourier series. Furthermore, we determined the d-dimensional Fourier series's trigonometric form and showed that it is precisely a single perceptron with an integer lattice mapping applied to its inputs. The weights of that perceptron are the Fourier series coefficients. As the Fourier series can theoretically represent any periodic signal, this perceptron can represent any periodic signal if it has an infinite number of frequencies in its mapping. However, in practice, the Fourier series coefficients are finite, and we can get them by sampling the signal at the Nyquist rate (twice the bandwidth) and applying a fast Fourier transform (FFT) to the signal. Thus, the number of Fourier coefficients is the theoretical upper bound of the number of parameters needed in the mapping.

Moreover, we modified the coarse-to-fine (CTF) training strategy of (Lin *et al.* (2021)), where we train the lower frequencies in the initial training phase and gradually add the higher frequency components as the training progresses. As a result, we show that our *coarse-to-fine* training strategy avoids the problem of over-fitting. Finally, we tested our proposed *Integer Lattice* mapping in the image regression and novel view synthesis tasks. We found out that the main contributor to the mapping performance is the number of parameters and the standard deviation, as was shown in (Tancik *et al.* (2020)).

In this chapter, first, we introduce an integer Fourier mapping and prove that a perceptron with this mapping is equivalent to a Fourier series. We also explore the mathematical connection between Fourier mappings and SIRENs and show that a Fourier-mapped perceptron is structurally like one hidden layer of SIREN. Also, we show that the integer mapping forces the periodicity of the network output. Furthermore, we modify the coarse-to-fine training strategy of (Lin *et al.* (2021)) and show that it improves the generalization of the interpolation task. We compare the different mappings on the image regression and novel view synthesis tasks and verify the previous findings of (Tancik

et al. (2020)) that the main contributor to the mapping performance is the number of elements and standard deviation.

5.2 Related work

Inspired by INRs' recent success, by outperforming grid-, point- and mesh-based representations (Park *et al.* (2019); Mescheder *et al.* (2019); Chen (2019)), many works based on INRs achieved state-of-the-art results in 3D computer vision (Atzmon and Lipman (2020); Gropp *et al.* (2020); Jiang *et al.* (2020); Peng *et al.* (2020); Chabra *et al.* (2020); Sitzmann *et al.* (2020b)). Moreover, impressive results are obtained across different input domains, e.g., from 2D supervision (Sitzmann *et al.* (2019); Niemeyer *et al.* (2020); Mildenhall *et al.* (2020)), 3D supervision (Saito *et al.* (2019); Oechsle *et al.* (2019b)), to dynamic scenes (Niemeyer *et al.* (2019)) which can be represented by space-time INR.

In early architectures, there was a lack of accuracy in the fine details of signals. Mildenhall *et al.* (2020) proposed positional encodings to tackle this problem, then Tancik *et al.* (2020) further explored positional encodings in an NTK framework, showing that mapping input coordinates to a representation close to the actual Fourier representation before passing them to the MLP lead to a good representation of the high-frequency details. Furthermore, they showed that random Fourier mappings achieved superior results than if one takes the simple positional encoding. Sitzmann *et al.* (2020b) also attempted to solve the problem of getting high-frequency details. They proposed SIRENs and demonstrated that SIRENs are suited for representing complex signals and their derivatives. In both solutions, they used a variant of Fourier neural networks (FNN) for the first layer of the MLP. FNN are neural networks that use either sine or cosine activations to get their features (Liu (2013)).

The first attempt to build an FNN was by (Gallant and White (1988)). They proposed a one-layer hidden neural network with a cosine squasher activation function and showed if they hand-wire certain weights, it will represent a Fourier series. Silvescu (1999) proposed a network that did not resemble a standard feedforward neural network. However, they used a cosine activation function to get the features. Liu (2013) introduced the general form for Fourier neural networks in a feedforward manner. They also proposed a strategy to initialize the frequencies of the embedding, which helped with convergence. Our work will show another way to initialize the embedding, which results in a neural network that is precisely a Fourier series.

5.3 Method

This section introduces the Fourier mapping we employed in our work. The connection between SIREN and Fourier mapping is also discussed. The training strategies are also discussed in this section.

5.3.1 Fourier mapping

This section explains how a perceptron with an integer lattice Fourier mapping applied to its inputs is equivalent to a Fourier series. First, we present the Fourier-mapped perceptron equation and then link it to the Fourier series's general equation. The fundamental building block of any neural network is the perceptron, and it is defined as

$$\mathbf{y}(\mathbf{x}, \mathbf{W}', \mathbf{b}) = g(\mathbf{W}' \cdot \mathbf{x} + \mathbf{b}). \tag{5.1}$$

Here $\mathbf{y} \in \mathbb{R}^{d_{out}}$ is the perceptron's output, $g(\cdot)$ is the activation function (usually non-linear), $\mathbf{x} \in \mathbb{R}^{d_{in}}$ is the input, $\mathbf{W}' \in \mathbb{R}^{d_{out} \times d_{in}}$ is the weight matrix, and $\mathbf{b} \in \mathbb{R}^{d_{out}}$ is the bias vector. Now, if we let $g(\cdot)$ to be the identity function and apply a Fourier mapping $\gamma(x)$to the input we get

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \mathbf{W} \cdot \gamma(\mathbf{x}) + \mathbf{b}, \tag{5.2}$$

where $\gamma(x)$ is the Fourier mapping defined as

$$\gamma(\mathbf{x}) = \begin{pmatrix} \cos(2\pi \mathbf{B} \cdot \mathbf{x}) \\ \sin(2\pi \mathbf{B} \cdot \mathbf{x}) \end{pmatrix}. \tag{5.3}$$

$\mathbf{W} \in \mathbb{R}^{d_{out} \times 2m}$, $\mathbf{B} \in \mathbb{R}^{m \times d_{in}}$ is the Fourier mapping matrix, and m is the number of frequencies. Equation 5.2 is the general equation of a Fourier-mapped perceptron, and we will relate it to the Fourier series's general equation.

A Fourier series is a weighted sum of sines and cosines with incrementally increasing frequencies that can reconstruct any periodic function when its number of terms goes to infinity. In applications that use coordinate-based MLPs, the functions we want to learn are not periodic. However, their inputs are naturally bounded (e.g., the height and width of an image). Accordingly, it doesn't harm if we assume that the input is periodic over its input's bounds to represent it as a Fourier series. We will explain later why this assumption has many advantages. A function $f : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ is periodic with a period $p \in \mathbb{R}^{d_{in}}$ if

$$f(\mathbf{x} + \mathbf{n} \circ \mathbf{p}) = f(\mathbf{x}) \quad \forall \mathbf{n} \in \mathbb{Z}^d, \tag{5.4}$$

where $\circ$ is the Hadamard product. As it is plausible to normalize the inputs to their bounds, we assume that each variable's period is 1. The Fourier series expansion of function (5.4) with $\mathbf{p} = \mathbf{1}^d$ is defined by (Osgood (2019)):

$$f(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{Z}^d} c_{\mathbf{n}} e^{2\pi i \mathbf{n} \cdot \mathbf{x}}, \tag{5.5}$$

where $c_{\mathbf{n}}$ are the Fourier series coefficients, and they are calculated by:

$$c_{\mathbf{n}} = \int_{[0,1]^d} f(\mathbf{x}) e^{-2\pi i \mathbf{n}\mathbf{x}} d\mathbf{x}. \tag{5.6}$$

For real-valued functions, it holds that $c_{\mathbf{n}} = c^*_{\text{-}\mathbf{n}}$ where $c^*_{\mathbf{n}}$ is the conjugate of $c_{\mathbf{n}}$. Using Euler's formula and mathematical induction we showed that equation (5.5) can be written as (see section 5.6.1):

$$f(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{N}_0 \times \mathbb{Z}^{d-1}} a_{\mathbf{n}} \cos(2\pi \mathbf{n} \cdot \mathbf{x}) + b_{\mathbf{n}} \sin(2\pi \mathbf{n} \cdot \mathbf{x}) \tag{5.7}$$

$$\begin{aligned} a_{\mathbf{0}} &= c_{\mathbf{0}}, \\ a_{\mathbf{n}} &= \begin{cases} 0 & \exists j \in \{2, \dots, d\} : n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0 \\ 2\mathrm{Re}(c_{\mathbf{n}}) & \text{otherwise,} \end{cases} \\ b_{\mathbf{n}} &= \begin{cases} 0 & \exists j \in \{2, \dots, d\} : n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0 \\ -2\mathrm{Im}(c_{\mathbf{n}}) & \text{otherwise.} \end{cases} \end{aligned} \tag{5.8}$$

And if we write equation (5.7) in vector form, we get

$$f(\mathbf{x}) = (a_{\mathbf{B}}, b_{\mathbf{B}}) \cdot \begin{pmatrix} cos(2\pi \mathbf{B} \cdot \mathbf{x}) \\ sin(2\pi \mathbf{B} \cdot \mathbf{x}) \end{pmatrix}, \tag{5.9}$$

where $a_{\mathbf{B}} = (a_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}}$, and $b_{\mathbf{B}} = (b_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}}$. Now, if we compare 5.2 and 5.9, we find similarities. We see that $(a_{\mathbf{B}}, b_{\mathbf{B}})$ is equivalent to $\mathbf{W}$, $\mathbf{b}$ is zero and $\mathbf{B} = \mathbb{N}_0 \times \mathbb{Z}^{d-1}$, is the concatenation of all possible permutations of $\mathbf{n}$. For practicality we limit $\mathbf{B}$ to

$$\mathbf{B} = \{0, \dots, N\} \times \{-N, \dots, N\}^{d-1} \setminus \mathbf{H}, \tag{5.10}$$

where N will be called the frequency of the mapping, $\mathbf{H} = \{\mathbf{n} \in \mathbb{N}_0 \times \mathbb{Z}^{d-1} | \exists j \in \{2, \dots, d\} : n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0\}$, then the perceptron represents a Fourier series. Hence, we calculate the dimension m of all possible permutations (see section 5.6.2)

$$m = (N+1)(2N+1)^{d-1} - \sum_{l=0}^{d-2} N(2N+1)^l. \tag{5.11}$$

In practice, we can find the Fourier series coefficients by sampling the function uniformly with a frequency higher than the Nyquist frequency and applying a Fast Fourier Transform (FFT) on the sampled signal. The resulting FFT coefficients are the Fourier series coefficients multiplied by the number of sampled points. And in theory, if we initialize the weights with the Fourier series coefficients, our network should give the training

target at iteration 0.

5.3.2 Fourier mapped perceptron as a SIREN

In this section, we want to show that a Fourier-mapped perceptron is structurally like a SIREN with one hidden layer. If we evaluate $\mathbf{W} \cdot \gamma(\mathbf{x})$ in equation (5.2), using (5.3) and combine the sine and cosine terms, we get:

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \mathbf{W} \cdot sin(2\pi \mathbf{C} \cdot \mathbf{x} + \phi) + \mathbf{b}, \tag{5.12}$$

where $\phi := (\pi/2, \ldots, \pi/2, 0, \ldots, 0)^T \in \mathbb{R}^{2m}$ and $\mathbf{C} := (\mathbf{B}, \mathbf{B})^T$. Here we see that $\mathbf{C}$ is acting as the weight matrix applied to the input, ϕ is like the first bias vector, and $sin(\cdot)$ is the activation function. Hence, the initial Fourier mapping can be represented by an extra initial SIREN layer, with the difference that $\mathbf{B}$ and ϕ are trainable in the SIREN case. This finding closes the bridge between Fourier frequency mappings and sinusoidal activation functions which have recently attracted a lot of attention.

5.3.3 Coarse-to-fine optimization

Lin *et al.* (2021) introduced a training strategy for coarse-to-fine registration for Neural Radiance Fields (NeRFs) (Mildenhall *et al.* (2020)) which they called BARF. Their idea is to mask out the positional encoding's high-frequency activations at the start of training and gradually allow them during training. Their work showed how to use this strategy on positional encodings only to improve camera registration. In our work, we will show how to run this strategy on an arbitrary Fourier mapping and show that it improves the generalization of the interpolation task. We weigh the frequencies of γ as follows:

$$\gamma^{\alpha}(\mathbf{x}) := \begin{pmatrix} w_{\mathbf{B}}^{\alpha} \\ w_{\mathbf{B}}^{\alpha} \end{pmatrix} \circ \gamma(\mathbf{x}) \tag{5.13}$$

where $w_{\mathbf{B}}^{\alpha}$ is the element wise application of the function $w_{\alpha}(z)$ on the vector of norms of $\mathbf{B}$ on the input dimension:

$$w_{\mathbf{B}}^{\alpha} := w_{\alpha} \begin{pmatrix} ||B_1||_2 \\ \vdots \\ ||B_m||_2 \end{pmatrix}. \tag{5.14}$$

where $w_{\alpha}(z)$ is defined as:

$$w_{\alpha}(z) = \begin{cases} 0 & \text{if } \alpha - z < 0 \\ \frac{1-\cos((\alpha-z)\pi)}{2} & \text{if } 0 \leq \alpha - z \leq 1 \\ 1 & \text{if } \alpha - z > 1 \end{cases} \tag{5.15}$$

Here, $\alpha \in [0, \max((\|B_i\|_{d_{\text{in}}})_{i\in\{1,\dots,m\}})]$ is a parameter which is linearly increased during training. This strategy forces the network to train the low frequencies at the start of training, ensuring that the network will produce smooth outputs. Later, when high-frequency activations are allowed, the low-frequency components are trained, and the network can focus on the left details. This strategy should reduce the effect of overfitting, which was introduced by Tancik *et al.* (2020) when using mappings with large standard deviations.

5.3.4 Frequency selection

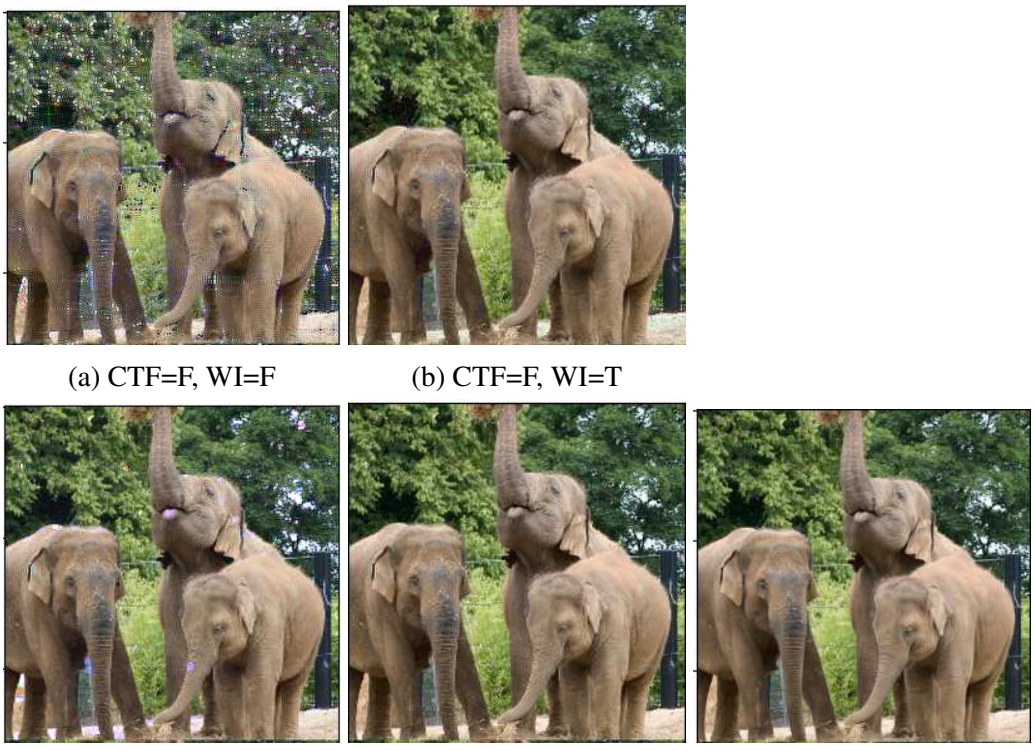
The standard way of using equation (6.2) is by defining a value N and taking the whole set $\mathbf{B}_N$. High-dimensional tasks lead to high memory consumption, and it is not clear whether this subset of $\mathbb{Z}^d$ brings the best performance. We, therefore, propose a way to select a more appropriate subset through data pruning. A pruning $pr(N,M)$ is done as follows: Assume we have $N,M \in \mathbb{N}$ with $M >> N$ and $|\mathbf{B}_N| = n$, $|\mathbf{B}_M| = m$. We train a perceptron with an integer mapping given by $\mathbf{B}_M$. After training we define $\mathbf{D}$ such that $\mathbf{D}$ contains only those elements of $\mathbf{B}_M$ where the respective weights are greater than a margin, that is chosen to yield $|\mathbf{D}| = n$. While $\mathbf{B}_N$ and $\mathbf{D}$ now have the same size, we believe that $\mathbf{D}$ will yield better performance because it contains the most relevant frequencies of the signal we want to reconstruct.

5.3.5 Fourier Mapping in MLPs

Although we showed in section 5.3.1 that we can represent any bounded input function with only one Fourier-mapped perceptron, in practice, these networks can become very wide to give high performance. As a result, the number of calculations will increase. To compromise between performance and speed, one can add depth and reduce the width of the network.

First, it is natural that using MLPs rather than perceptrons increases performance. However, it remains unclear why our proposed integer mapping should perform better than competing mappings for multilayer networks.

One could argue that if a mapping gives the perceptron a high representation power, it will also provide a high representation power to the MLP and vice versa. First, however, we should verify this claim with experiments. In addition, we remind the reader that a periodic function has integer frequencies. And because our assumption that the signal we want to reconstruct is periodic, it will have only integer frequencies. Also, the activation functions we are using only introduce integer frequencies when applied to a periodic function, as shown for the 1D case in the supplementary material. With this, we reduce the search space for frequencies from $\mathbb{R}$ to $\mathbb{Z}$, which could make the optimization easier as the search space is more compact and approachable.



(a) CTF=F, WI=F (b) CTF=F, WI=T

(c) CTF=T, WI=F (d) CTF=T, WI=T (e) Ground Truth

Figure 5.2: A visualization of the outputs of Fourier mapped perceptrons of $N = 128$. CTF stands for coarse-to-fine training and WI stands for weight initialization. T/F stands for True/False, respectively.

5.4 Experiments

5.4.1 Weight initialization and coarse-to-fine training

In this section, we want to confirm our mathematical claims through experiments. First, we will show that the derivation of the integer mapping indeed represents the Fourier series. Secondly, we want to check whether coarse-to-fine training helps with generalization.

We conducted our experiments on the image regression task. This task aims to make a neural network memorize an image by predicting the color at each pixel location. We use ten images with a resolution of 512×512, which can be found in the supplementary material, and report the mean peak signal-to-noise ratio (PSNR). We divide the image into train and test sets, where we use every second pixel for training and take the complete image for testing. We utilize 3 Fourier-mapped perceptrons with $N = 128$ (Nyquist frequency), one for each image channel. We normalize the input ($\mathbf{x}$) to have an interval between [0,1] in both width (x) and height (y) dimensions.

In this experiment, we made an ablation: With and without weight initialization using the normalized FFT coefficients of the image's training pixels, with and without the coarse-to-fine training scheme explained in section 5.3.3. For coarse-to-fine training, α was linearly increased from 0 to its maximum value at 75% of training iterations. In training, we only make an update step after we accumulate the gradients of the whole image. We did not study learning schedules in this work, and the reader is encouraged to try different schedules. Figure 5.2 shows a visualization of one of the images, and figure 5.3 shows the training progress, where the solid line is the mean PSNR and the shaded area shows the standard deviation. As can be deducted from figure 5.3, one can

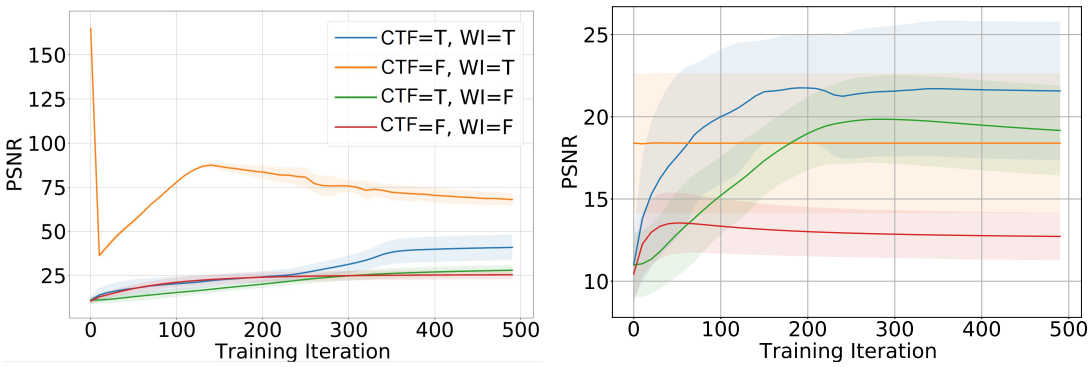


Figure 5.3: The training progress of Fourier mapped perceptrons with $N = 128$. The left and right figures report the train and test PSNR, respectively. Weight initialization without CTF yields a PSNR of 160 which one can consider as the ground truth proving that the perceptron is a Fourier series. Note: The y-axis limits are different in both plots.

see that the train PSNR starts at an optimum at the start of training when we use weight initialization (WI), and we don't use coarse-to-fine training (CTF). This fact underlines our claim that *a perceptron with an integer lattice mapping is indeed a Fourier series*. Note that in case both WI and CTF are used, the train PSNR is not optimal at the start because the CTF masks out high-frequency activations.

We can also see from figure 5.3 that whenever we use coarse-to-fine training, it always shows a higher test PSNR, which confirms that *coarse-to-fine training helps with generalization*. Lastly, when we did not employ both CTF and WI, the perceptron overfits to the training pixels, and this can be seen quantitatively with a very low test PSNR (red line in figure 5.3) and qualitatively with grid-like artifacts (in figure 5.2a)).

5.4.2 Perceptron experiments

In this experiment, we want to compare the representation power of the different mappings in the single perceptron case. We conducted our experiments in the same setting

as in section 5.4.1, where we used coarse-to-fine training and did not use weight initialization.

In the integer mapping, we increased N's value from 4 until half the training image dimension (Nyquist frequency) and calculated all possible permutations $\mathbf{B}_N$, as discussed in section 5.3.1. For the Gaussian mapping, we sample $m = |\mathbf{B}_N|$ parameters from a Gaussian distribution with a standard deviation of 10 (which was the best value for this task in our experiments). Also, we test a one-layer SIREN with one hidden layer having the same size m. Finally, we adopt the positional encoding (PE) scheme from (Mildenhall *et al.* (2020)) and limit its values to N. Figure 5.4 shows our experiments' results on the train and test pixels, respectively. Figure 5.5 shows the networks' outputs trained on one of the images.

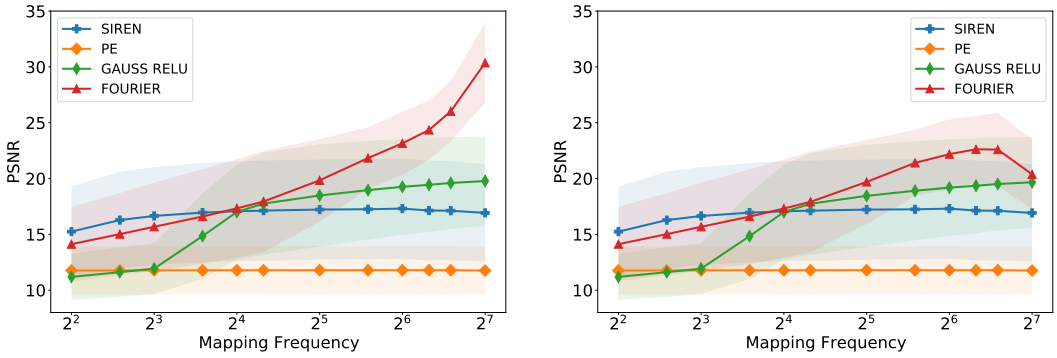


Figure 5.4: Perceptron experiments with different values for the mapping frequency N. We report the train PSNR on the left and the test PSNR on the right. For high values of N our integer mapping outperforms all competing mappings.

At low N values, we see that the Gaussian mapped perceptrons do not work because the number of sampled frequencies is low, so there is a low chance that samples will be near the image's critical frequencies. On the other hand, the integer-mapped perceptrons have slightly lower performance because they can only learn low frequencies. The SIREN performs relatively well in this case, and we think this is because SIRENs naturally inherit a learnable Fourier mapping that is not restricted to the initial sampling, as described in section 5.3.2. PE can only produce horizontal and vertical lines because it has diagonal frequencies (only one non-zero frequency is allowed), and this effect is persistent at any value of N.

As N increases, SIREN, Gauss, and integer mapping performance increase giving similar performance around $N = 16$. For high values of N, we see that in figure 5.5, the integer lattice mapping of the Fourier coefficients outperforms the competing mappings, clearly displaying more details in the reconstruction. On the other hand, the PSNR of the SIREN and the Gaussian mapped perceptrons saturates. We think this is because both mappings rely on sampling the frequencies. Although we can get many of the critical frequencies

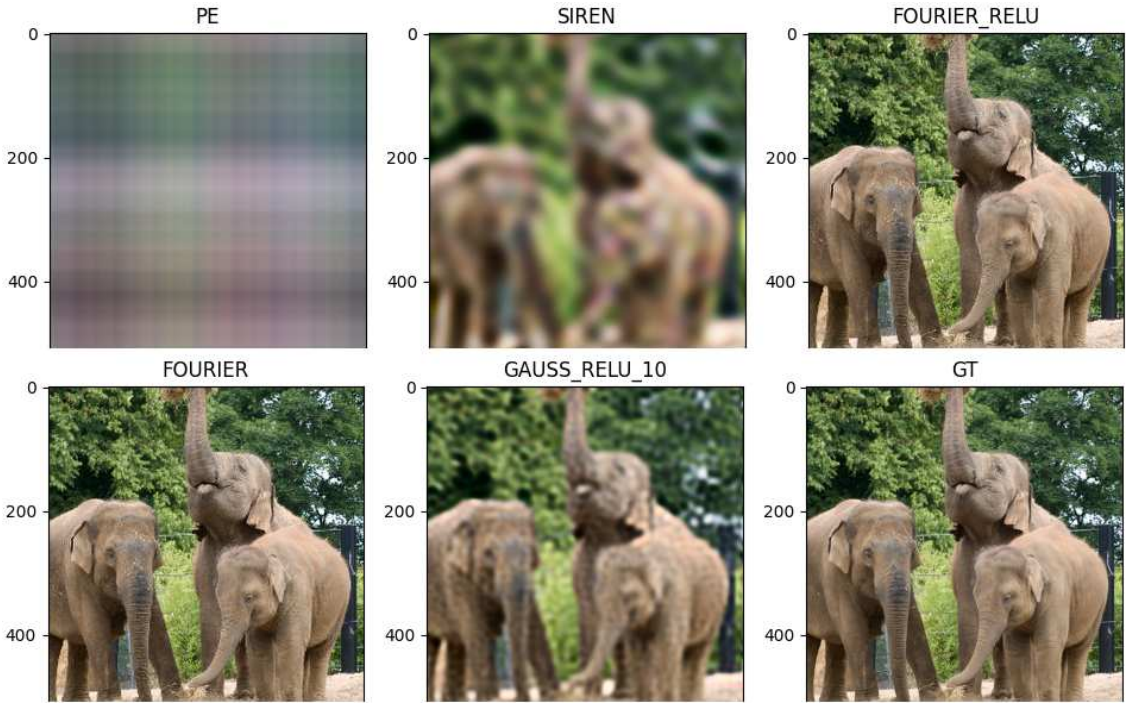


Figure 5.5: The visualization of the Fourier mapped perceptrons and the one layer SIREN with $N = 128$.

of the image with sampling, it is improbable to get all of them simultaneously. Even the trainability of the SIREN mapping did not help in this case.

5.4.3 MLP experiments

Our theory for integer mapping assumes an underlying function that is periodic. However, it is not clear that we will end up with a periodic function if we go the other way, using an integer mapping. In this experiment, we want to check if applying an integer mapping forces periodicity. Secondly, we want to validate our claim (in section 5.3.5) that if a mapping gives the perceptron a high representation power, it will also give a high representation power to the MLP and vice versa. We compared ReLU networks with integer, Gaussian, PE, and pruned integer mapping (section 5.3.4). We also compared SIRENS with no mapping (extra layer), integer, or pruned mapping. We made a grid search of the parameters N=[8, 16, 32], depth=[0, 2, 4, 6] (depth=0 represents a perceptron), and fixed the width to 32. For the pruned mapping, we used a $pr(N, 128)$. And for the Gaussian mapping, we had two settings. The first one had a standard deviation of 10 (σ_{10}), which had the best performance in the perceptron experiments. In the second one, we set the standard deviation the same as the pruned integer mapping's standard deviation (σ_{pr}) to check its effect. Tables 5.1 and 5.2 show the mean train and

Activ.	Map.	N=8 m=113				N=16 m=481			
		d=0	d=2	d=4	d=6	d=0	d=2	d=4	d=6
Sine	No	**16.65**	22.15	**23.26**	**24.07**	17.07	22.09	23.84	19.76
	Int.	15.68	**22.31**	22.41	20.94	**17.33**	27.66	27.06	**27.33**
	Pr.	15.28	21.03	22.40	23.00	16.76	**28.17**	**27.68**	24.66
Relu	P.E.	11.78	16.61	17.37	17.77	11.78	16.87	17.79	17.95
	Gs. σ_{10}	11.93	21.90	21.68	21.69	17.01	24.53	24.26	25.13
	Gs. σ_{pr}	14.06	20.23	20.78	20.88	12.69	26.02	26.40	26.72
	Int.	15.68	20.51	20.65	20.62	**17.33**	24.42	24.09	24.49
	Pr.	15.28	20.35	20.92	20.96	16.76	25.87	26.23	26.33

Activ.	Map.	N=32 m=1985			
		d=0	d=2	d=4	d=6
Sine	No	17.22	14.90	14.67	13.63
	Int.	**19.84**	33.78	26.98	23.60
	Pr.	18.48	37.34	30.41	19.74
Relu	P.E.	11.78	17.05	18.15	18.15
	Gs. σ_{10}	18.48	26.10	26.30	27.48
	Gs. σ_{pr}	13.01	37.69	**37.90**	**37.74**
	Int.	**19.84**	31.57	32.14	32.79
	Pr.	18.48	**37.70**	36.81	37.48

Table 5.1: The mean train PSNR results of network type comparison experiment with varying network depth (d), number of frequencies (N). We use the following abbreviations: Activ. = Activation function, Map. = Mapping type, Int. = Integer, Pr.= Pruned Integer, P.E. = Positional Encoding, Gs. = Gaussian. Here, m is the mapping size and σ is the standard deviation.

test PSNRs, respectively.

Figure 5.1 shows a visualization of the network's outputs at N = 128 and width = 32 for the first period and next period in the height and width directions ($f([x+1, y+1])$). We see that *the integer mapping forces the network's underlying function to be periodic* unlike ReLU network with Gauss mapping, which proves our first hypothesis.

From the table 5.1 we see that if a mapping at $d = 0$ gives the highest PSNR, this does not mean that it will give the highest PSNR for $d > 0$ and vice versa. One clear example at $N = 32$ is the Gauss σ_{pr}, where it has a PSNR of 13.01 dB at $d = 0$, which is lower than integer mapping (19.84 dB), but has the highest PSNR at $d = [4, 6]$. This result disproves our initial assumption that if a mapping gives the perceptron a high representation power, it will also give a high representation power to the MLP. We see also that the pruned integer mapping has comparable results with the Gauss σ_{pr}, and this shows that the main contributor to the performance is the mappings' standard deviation.

From the tables, we can also observe some trends. First, networks with sine activations

Activ.	Map.	N=8 m=113				N=16 m=481			
		d=0	d=2	d=4	d=6	d=0	d=2	d=4	d=6
Sine	No	**16.65**	21.63	**21.85**	**21.99**	17.06	21.28	22.03	18.50
	Int.	15.68	**21.75**	21.53	20.06	**17.31**	**23.48**	**22.67**	22.28
	Pr.	15.28	20.49	21.22	21.45	16.75	22.00	21.39	22.17
Relu	P.E.	11.78	16.60	17.33	17.70	11.78	16.85	17.73	17.87
	Gs. σ_{10}	11.93	20.67	21.06	20.90	17.00	22.96	22.78	23.04
	Gs. σ_{pr}	14.06	19.89	20.22	20.21	12.69	22.46	22.48	22.16
	Int.	15.68	20.27	20.35	20.23	**17.31**	22.93	22.65	**22.50**
	Pr.	15.28	19.98	20.33	20.21	16.75	22.31	22.26	22.09

Activ.	Map.	N=32 m=1985			
		d=0	d=2	d=4	d=6
Sine	No	17.22	13.57	13.29	12.37
	Int.	**19.70**	16.85	17.89	16.36
	Pr.	18.39	20.49	15.15	13.13
Relu	P.E.	11.79	17.02	18.06	18.02
	Gs. σ_{10}	18.45	23.66	23.61	**23.73**
	Gs. σ_{pr}	12.99	23.12	23.48	23.33
	Int.	**19.70**	**24.36**	**24.02**	**23.73**
	Pr.	18.39	23.24	23.18	23.30

Table 5.2: The mean test PSNR results of network type comparison experiment. For abbreviations see table 5.1.

and large mappings collapse during performance worse than Relu networks. Second, the integer mapping usually gives the best test PSNR, demonstrating its effectiveness in the MLP case. Third, the pruned integer mapping shows consistently better train PSNR than the normal integer mapping at $d > 0$. We believe this is because pruned mapping has a higher standard deviation. Finally, the PE is worse in every case because we cannot easily control the standard deviation, and it has very few parameters.

5.4.4 2D to 3D experiments

This section wants to see if our findings in the image regression task transfer to the novel view synthesis (NVS) task. In NVS, we are given a set of 2D images of a scene, and we try to find its 3D representation. With this representation, one can render images from new viewpoints. In contrast to the 2D experiments, the inputs are (x, y, z) coordinates that are mapped to a 4-dimensional output, the RGB values, and a volume density. For this experiment, a simplified version of the official NeRF (Mildenhall *et al.* (2020)) is used, where the view dependency and hierarchical sampling are removed. Here, we experiment with the input mappings used in section 5.4.3. Unless otherwise stated, we

adopt the settings from the image regression task. We set the network width to be 64.

As the mapping size increases exponentially, we do our experiments with lower frequencies than in the 2D case. Specifically, we used integer mapping on four frequencies. The frequencies of our mapping were limited to the maximum network size which we could fit on a NVIDIA GTX-2080Ti. The pruning is given by $pr(4,8)$. We conduct our experiments on the bulldozer scene, which is commonly used for NeRF experiments. For training, we used a batch size of 128, 50.000 epochs and a learning rate of 5×10^{-4}.

As seen in Table 5.3, in the perceptron case ($d = 0$), SIREN provides the best performance, which aligns with our image regression results at low values of N. We observe that the pruned mapping increases the performance compared to normal mapping for both Relu and sinusoidal activation. This increase in performance is because pruned mapping has a higher standard deviation than normal mapping. Gauss gives comparable results to pruned integer mapping because they have the same standard deviation. These findings align with our conclusions from image regression experiments. However, due to memory limitations, we could not test a perceptron with frequencies higher than 8, which was superior in image regression.

Act.	Map.	$N = 4$				N=8
		d=0	d=2	d=4	d=6	d=0
Sine	No	**20.37**	23.08	**23.55**	23.35	OM
	Int.	18.42	22.22	22.95	22.97	**19.31**
	Pr.	19.15	**23.12**	**23.58**	23.36	-
Relu	P.E.	16.30	21.48	22.64	23.51	16.40
	Gs.	18.93	22.81	**23.64**	**23.82**	19.29
	Int.	18.42	21.81	22.68	23.28	**19.31**
	Pr.	19.15	22.78	23.61	**23.89**	**-**

Table 5.3: Validation PSNR scores of NERF experiments using a mapping of frequency 4. OM stands for out of memory. For other abbreviations see table 5.1.

5.5 Conclusion

In this work, we identified a relationship between the Fourier mapping and the general d-dimensional Fourier series, which led to the *integer lattice mapping*. We also showed that this mapping forces the periodicity of the neural network underlying function. From experiments, we showed that one perceptron with frequencies equal to the Nyquist rate of the signal is enough to reconstruct it. Furthermore, we showed that the coarse-to-fine training strategy improves the generalization of the interpolation task. Lastly, we confirmed the previous findings that the main contributor to the mapping performance is its size and the standard deviation of its elements.

5.6 Proofs

5.6.1 Equation (5.7)

We will prove the validity of equation (5.7) in the general case of dimension $d \in \mathbb{N}$. We use the concept of mathematical induction for this task. Therefore we show, that the equation is true for $d = 1$ and additionally prove, that if the equation holds for dimension $d-1$ it is also valid for dimension d.
$\underline{d = 1:}$

$$\begin{aligned}
f(x) &= \sum_{n\in\mathbb{Z}^1} c_n e^{2\pi i n\cdot x} \\
&= \sum_{n\in\mathbb{N}} c_n e^{2\pi i n\cdot x} + \sum_{n\in\mathbb{N}} c_{-n} e^{-2\pi i n\cdot x} + c_0 \\
&\overset{c_n^* = c_{-n}}{=} \sum_{n\in\mathbb{N}} \big(\mathrm{Re}(c_n) + i\mathrm{Im}(c_n)\big)\big(\cos(2\pi nx) + i\sin(2\pi nx)\big) \\
&\quad + \sum_{n\in\mathbb{N}} \big(\mathrm{Re}(c_n) - i\mathrm{Im}(c_n)\big)\big(\cos(2\pi nx) - i\sin(2\pi nx)\big) \\
&\quad + c_0 \\
&= \sum_{n\in\mathbb{N}} 2\mathrm{Re}(c_n)\cos(2\pi nx) - 2\mathrm{Im}(c_n)\sin(2\pi nx) \\
&\quad + c_0 \\
&= \sum_{n\in\mathbb{N}_0} a_n\cos(2\pi nx) + b_n\sin(2\pi nx),
\end{aligned} \tag{5.16}$$

where

$$a_0 = c_0,\ a_n = 2\mathrm{Re}(c_n),\ b_n = -2\mathrm{Im}(c_n). \tag{5.17}$$

$\underline{\text{Assumption of the induction:}}$
We will assume that the equation holds for $d-1$, where $d \geq 2$.

$\underline{\text{Induction step: } d-1 \to d\text{:}}$
As the Fourier series of any periodic and continuous function is absolutely convergent,

we are allowed to rearrange the sum in $(*)$ and receive

$$\begin{aligned}
&\sum_{\mathbf{n}=(n_1,\dots,n_d)\in\mathbb{Z}^d} c_{\mathbf{n}} e^{2\pi i \mathbf{n}\cdot\mathbf{x}} \\
&\overset{(*)}{=} \sum_{n_1\in\mathbb{N}} \sum_{(n_2,\dots,n_d)\in\mathbb{Z}^{d-1}} c_{\mathbf{n}} e^{2\pi i \mathbf{n}\cdot\mathbf{x}} \\
&\quad + \sum_{n_1\in\mathbb{N}} \sum_{(n_2,\dots,n_d)\in\mathbb{Z}^{d-1}} c_{-\mathbf{n}} e^{-2\pi i \mathbf{n}\cdot\mathbf{x}} \\
&\quad + \sum_{n_1=0}^{0} \sum_{(n_2,\dots,n_d)\in\mathbb{Z}^{d-1}} c_{\mathbf{n}} e^{2\pi i \mathbf{n}\cdot\mathbf{x}} \\
&\overset{c_{\mathbf{n}}^*=c_{-\mathbf{n}}}{=} \sum_{\mathbf{n}\in\mathbb{N}\times\mathbb{Z}^{d-1}} 2\mathrm{Re}(c_{\mathbf{n}})\cos(2\pi\mathbf{n}\cdot\mathbf{x}) - 2\mathrm{Im}(c_{\mathbf{n}})\sin(2\pi\mathbf{n}\cdot\mathbf{x}) \\
&\quad + \sum_{\mathbf{n}\in\{0\}\times\mathbb{Z}^{d-1}} c_{\mathbf{n}} e^{2\pi i \mathbf{n}\cdot\mathbf{x}} \\
&\overset{\text{Ind. asm.}}{=} \sum_{\mathbf{n}\in\mathbb{N}\times\mathbb{Z}^{d-1}} 2\mathrm{Re}(c_{\mathbf{n}})\cos(2\pi\mathbf{n}\cdot\mathbf{x}) - 2\mathrm{Im}(c_{\mathbf{n}})\sin(2\pi\mathbf{n}\cdot\mathbf{x}) \\
&\quad + \sum_{\mathbf{n}\in\{0\}\times\mathbb{N}_0\times\mathbb{Z}^{d-2}} a'_{\mathbf{n}}\cos(2\pi\mathbf{n}\cdot\mathbf{x}) + b'_{\mathbf{n}}\sin(2\pi\mathbf{n}\cdot\mathbf{x}),
\end{aligned} \tag{5.18}$$

where

$$\begin{aligned}
a'_{\mathbf{0}} &= c_{\mathbf{0}}, \\
a'_{\mathbf{n}} &= \begin{cases} 0 & \exists j\in\{3,\dots,d\} : n_2 = \cdots = n_{j-1} = 0 \wedge n_j < 0 \\ 2\mathrm{Re}(c_{\mathbf{n}}) & \text{otherwise,} \end{cases} \\
b'_{\mathbf{n}} &= \begin{cases} 0 & \exists j\in\{3,\dots,d\} : n_2 = \cdots = n_{j-1} = 0 \wedge n_j < 0 \\ -2\mathrm{Im}(c_{\mathbf{n}}) & \text{otherwise.} \end{cases}
\end{aligned} \tag{5.19}$$

Combining these two summands we get

$$\sum_{\mathbf{n}\in\mathbb{N}_0\times\mathbb{Z}^{d-1}} a_{\mathbf{n}}\cos(2\pi\mathbf{n}\cdot\mathbf{x}) + b_{\mathbf{n}}\sin(2\pi\mathbf{n}\cdot\mathbf{x}), \tag{5.20}$$

where

$$
\begin{aligned}
a_{\mathbf{0}} &= c_{\mathbf{0}}, \\
a_{\mathbf{n}} &= \begin{cases} 0 & \exists j \in \{2,\dots,d\} : n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0 \\ 2\mathrm{Re}(c_{\mathbf{n}}) & \text{otherwise,} \end{cases} \\
b_{\mathbf{n}} &= \begin{cases} 0 & \exists j \in \{2,\dots,d\} : n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0 \\ -2\mathrm{Im}(c_{\mathbf{n}}) & \text{otherwise.} \end{cases}
\end{aligned} \tag{5.21}
$$

5.6.2 Equation (5.11)

In the following we use $|\cdot|$ to talk about the number of elements in a set. Furthermore, we use the notation $[\![n]\!] := \{0,\dots,n\}$ for $n \in \mathbb{N}$ and $[\![m,l]\!] := \{m,\dots,l\}$ for $m,l \in \mathbb{Z}$ and $m < l$. We have

$$
\begin{aligned}
\mathbf{B} = \{0,\dots,N\} \times \{-N,\dots,N\}^{d-1} \setminus \{\mathbf{n} \in \mathbb{N}_0 \times \mathbb{Z}^{d-1} : & \qquad (5.22) \\
\exists j \in \{2,\dots,d\} : n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0\}. & \qquad (5.23)
\end{aligned}
$$

It is immediately clear, that

$$
|\{0,\dots,N\} \times \{-N,\dots,N\}^{d-1}| = (N+1)(2N+1)^{d-1}, \tag{5.24}
$$

therefore the only thing we need to show is, that

$$
\begin{aligned}
|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!]^{d-1} : \exists j \in \{2,\dots,d\} : & \qquad (5.25) \\
n_1 = \dots = n_{j-1} = 0 \wedge n_j < 0\}| & \qquad (5.26) \\
= \sum_{l=0}^{d-2} N(2N+1)^l. & \qquad (5.27)
\end{aligned}
$$

We will do this proof with mathematical induction. We start with $\underline{d = 2}$:

$$
\begin{aligned}
&|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!] : \exists j \in \{2\} : n_1 = 0 \wedge n_j < 0\}| && (5.28) \\
&= |\{\mathbf{n} \in \{0\} \times [\![-N,-1]\!]\}| && (5.29) \\
&= N && (5.30)
\end{aligned}
$$

Assumption of the induction:
We will assume that the equation holds for some d, where $d \geq 2$.

Induction step: $d \to d+1$:

$$
\begin{aligned}
&\left|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!]^d : \exists j \in [\![2,d+1]\!] : \right. && (5.31)\\
&\quad \left. n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0\}\right| && (5.32)\\
&= \left|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!]^d : \exists j \in [\![3,d+1]\!] : \right. && (5.33)\\
&\quad \left. n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0\}\right| + && (5.34)\\
&\quad \left|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!]^d : \exists j \in \{2\} : \right. && (5.35)\\
&\quad \left. n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0\}\right| && (5.36)\\
&= \left|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!]^d : \exists j \in [\![3,d+1]\!] : \right. && (5.37)\\
&\quad \left. n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0\}\right| + && (5.38)\\
&\quad \left|\{\mathbf{n} \in \{0\} \times [\![-N,-1]\!] \times [\![-N,N]\!]^{d-1}\}\right| && (5.39)\\
&= \left|\{\mathbf{n} \in [\![N]\!] \times [\![-N,N]\!]^{d-1} : \exists j \in [\![2,d]\!] : \right. && (5.40)\\
&\quad \left. n_1 = \cdots = n_{j-1} = 0 \wedge n_j < 0\}\right| + && (5.41)\\
&\quad N(2N+1)^{d-1} && (5.42)\\
&\overset{\text{Ind. asm.}}{=} \sum_{l=0}^{d-2} N(2N+1)^l + N(2N+1)^{d-1} = \sum_{l=0}^{d-1} N(2N+1)^l. && (5.43)
\end{aligned}
$$

Chapter 6

Two-stage instance segmentation using Fourier series

In this chapter, we focus on using a Fourier representation of instance segmentation masks on a two-stage object detection framework. Two-stage models have the advantage of accurately detecting objects in the image and then predicting their attributes, for example class, bounding box, and segmentation masks. We employed Mask R-CNN (He *et al.* (2017)) as our baseline two-stage model and developed a mask head, which we call *FourierMask*.

6.1 Introduction

In the past decade, we have witnessed a shift from classical approaches toward deep learning methods for a variety of real-world tasks. Due to an ample amount of real and synthetic datasets and high computation power, it has been possible to reliably use these 'black box' models on highly complex and critical use cases. In the field of autonomous driving, perceiving and understanding the environment is crucial. Instance segmentation is one of those tasks which allows autonomous systems to semantically separate different regions in their percepts (images) and at the same time separate objects from each other.

In recent years, the majority of the methods have employed CNNs for the task of instance segmentation. There are methods that generate instance segmentation masks by classifying each pixel of a region of interest as either foreground or background e.g Mask R-CNN (He *et al.* (2017)). These methods generally show the best performance but suffer from high computation needs and slow speed. There are methods that predict the contour points around the boundary of the object (Yang *et al.* (2019)) (Xie *et al.* (2020)). Although faster, these methods fail to match the performance of pixel-wise classification methods. Alternatively, there are methods that try to encode the mask contours in a compressed representation (Xu *et al.* (2019)) (Riaz *et al.* (2021)).

These mask representations are compact and meaningful but lack the superior capabilities of pixel-wise methods. Our approach employs a pixel-wise mask representation and additionally we generate the mask using a compact Fourier representation. In the case of segmentation masks, the Fourier series' low-frequency components hold the gen-

(a) Mask R-CNN (b) PointRend (c) FourierMask

Figure 6.1: Comparison between Mask R-CNN (He *et al.* (2017)), PointRend (Kirillov *et al.* (2020a)) and FourierMask.

eral shape and high-frequency components hold the edges of the mask. Therefore, our representation is meaningful and can be compressed according to the use case.

FourierMask is an *implicit neural representation*. For the image regression task, *implicit representations* learn to predict the RGB values of a particular image given a pixel coordinate. Tancik *et al.* (2020) showed that using a Fourier Mapping of coordinates instead of actual coordinates locations as inputs, allows the implicit networks to learn high-frequency details in images and 3D scenes. Our work draws inspiration from this work and applies their findings to the task of generating masks for instance segmentation.

These representations are inherently trained on a single image and have not yet been adopted to a general task of instance segmentation on natural images. In our work, we employ the implicit representations by applying a *Fourier mapping* $\gamma(\mathbf{x})$ to the input coordinates of the implicit network. Essentially, FourierMask predicts the weights $\mathbf{W}$, which are the coefficients of each term in the mapping $\gamma(\mathbf{x})$. Since $\mathbf{W}$ is adapted to a specific ROI, FourierMask learns to weigh the effect of each term in the Fourier series for that particular object. The mapped features are passed through an implicit neural network. The combination of Fourier mapping and the implicit neural representation lets the model learn high-frequency details and produce crisp masks.

Implicit representations have the advantage of learning and reconstructing fine details, which traditional representations cannot do as effectively in such compact models. As implicit functions are continuous in the domain of input coordinates, we can sub-sample the pixel coordinates to generate higher-resolution masks during inference. Our contributions are as follows:

1. We developed FourierMask, which can replace any mask predictor that uses a region of interest (ROI) to predict a binary mask. It is fully differentiable and end-to-end trainable.

2. We show that implicit representations can be applied to the task of instance segmentation. We achieve this by learning the coefficients of the Fourier mapping of a particular object.

3. As implicit functions are continuous in the domain of input coordinates, we show that we can sub-sample the pixel coordinates to generate higher-resolution masks during inference. These higher-resolution masks are smoother and improve the performance on MS COCO.

4. We verify and illustrate that the rendering strategy from PointRend (Kirillov *et al.* (2020a)) brings significant qualitative gains for FourierMask. Our renderer MLP *FourierRend* significantly improves the mask boundary of FourierMask.

6.2 Related work

In *two-stage* instance segmentation, the network first detects (proposes) the objects and then predicts a segmentation mask from the detected region. The baseline method for many two-stage methods is Mask R-CNN (He *et al.* (2017)). Mask R-CNN added a mask branch to Faster R-CNN (Ren *et al.* (2015)), which generated a binary mask that separated the foreground and background pixels in a region of interest. Mask Scoring R-CNN (Huang *et al.* (2019)) had a network block to learn the quality of the predicted instance masks and regressed the mask IoU. ShapeMask (Kuo *et al.* (2019)) estimated the shape from bounding box detections using shape priors and refined it into a mask by learning instance embeddings. CenterMask (Lee and Park (2020)) added a spatial attention-guided mask on top of FCOS (Tian *et al.* (2019)) object detector, which helped to focus on important pixels and diminished noise. PointRend (Kirillov *et al.* (2020a)) tackled instance segmentation as a rendering problem. By sampling unsure points from the feature map and its fine-grained features from higher-resolution feature maps, it was able to predict really crisp object boundaries using a fully connected MLP. Rather than employing binary grid representation of masks, PolyTransform (Liang *et al.* (2020)) used a polygon representation. These methods accomplish state-of-the-art accuracy, but they are generally slower than one-stage methods.

One stage instance segmentation methods predict the instance masks in a single shot, without using any proposed regions/bounding boxes as an intermediate step. YOLACT (Bolya *et al.* (2019)) linearly combined prototype masks and mask coefficients for each instance, to predict masks at real-time speeds. Likewise, Embedmask (Ying *et al.* (2019)) employed embedding modules for pixels and proposals. ExtremeNet (Zhou *et al.* (2019a)) predicted the contour (octagon) around an object using keypoints of the object. Similarly, Polarmask (Xie *et al.* (2020)) used the polar representation to predict a contour from a center point (centerness from FCOS (Tian *et al.* (2019))). Dense RepPoints (Yang *et al.* (2019)) used a large set of points to represent the boundary of objects. FourierNet (Riaz *et al.* (2021)) employed inverse fast Fourier transform (IFFT) to generate a contour around an object represented by polar coordinates. The network learned the coefficients of the Fourier series to predict those contours.

Implicit representations learn to encode a signal as a continuous function. Mescheder

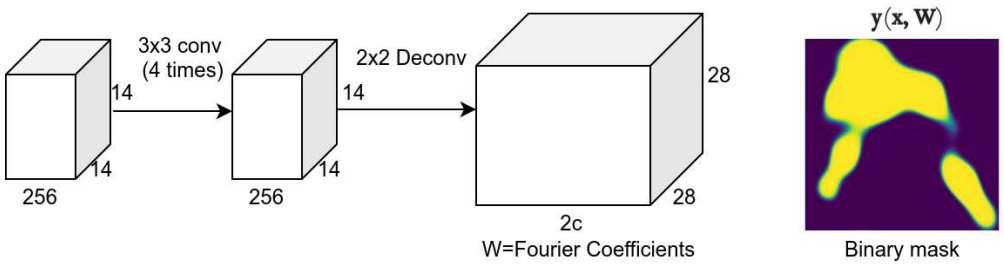


Figure 6.2: FourierMask head architecture for a ROI Align size of 14x14. The network predicts Fourier coefficients **W** for each location in the feature map.

et al. (2019) proposed occupancy networks, which implicitly represented the 3D surface as the continuous decision boundary of a deep neural network classifier. Mildenhall *et al.* (2020) used implicit neural networks to learn 3D scenes and synthesize novel views. Tancik *et al.* (2020) showed that mapping input coordinates using a Fourier feature mapping, allows MLPs to learn high-frequency functions in low-dimensional domains. Sirens (Sitzmann *et al.* (2020a)) showed that using periodic activation functions (such as sine) in implicit representations, lets the MLPs learn the information of natural signals and their derivatives better than when using other activation functions such as ReLU.

6.3 Method

6.3.1 FourierMask head architecture

This section explains the network architecture of FourierMask. We employ Mask R-CNN (He *et al.* (2017)) as our baseline model. We use a ResNet (He *et al.* (2016b)) backbone pre-trained on ImageNet dataset (Deng *et al.* (2009)), with a feature pyramid network (FPN) (Lin *et al.* (2017a)) architecture. Following from Mask R-CNN, we use a small region proposal network (RPN), which generates k proposal candidates from all feature levels from FPN. To generate fixed-size feature maps from these proposal candidates, we use a ROI Align (He *et al.* (2017)) operation. This produces a (k,d,m,m) sized feature map for the mask head, where m is the fixed spatial size after the ROI Align operation and d is the number of channels.

We apply four convolutions consecutively on these ROIs, each with a kernel size 3×3 and a stride of 1. Then we apply a deconvolution layer with $2c$ number of filters, which generates a spatial volume of size $2c \times 2m \times 2m$. We call this feature volume **W**, which holds $2c$ Fourier coefficients for each spatial location. The structure of the head is shown in figure 6.2.

6.3.2 Fourier Mapping

In this section, we explain how we achieve the Fourier mapping from the coefficients **W**, which is the core contribution of this paper. The Fourier mapping is given as

$$\gamma(\mathbf{x}) = [cos(2\pi\mathbf{x}\cdot\mathbf{B}), sin(2\pi\mathbf{x}\cdot\mathbf{B})]. \tag{6.1}$$

Here $\mathbf{x} \in \mathbb{R}^{p\times 2}$ are the pixel coordinates (i, j) normalized to a value in range $[0, 1]$ and p are the total number of pixels in the image. Since sine and cosine have a period of 2π, by normalizing the pixel coordinate **x** to a range $[0, 1]$, we ensure one complete image lies in the period of 2π. Although images are not periodic signals, since they are bounded by an image resolution, we can safely apply our method to predict 2D binary masks. $\mathbf{B} \in \mathbb{Z}^{2\times c}$ is the integer lattice matrix that holds the possible combinations of harmonic frequency integers of Fourier series for both dimensions in the image. **B** contains the elements of the set S, defined by

$$S = \{0, \ldots, f\} \times \{-f, \ldots, f\} \setminus \{0\} \times \{-f, \ldots, -1\}, \tag{6.2}$$

where f is the total number of frequencies. This way of defining **B** is motivated by the 2D Fourier representation. If we do not limit **B** by the total number of frequencies f, we would obtain the original 'sine plus cosine' form of the Fourier series. This property is shown in the next section (6.3.3). For the case of images (2D input), the possible permutations c can be calculated by:

$$c = (f+1)(2f+1) - f \tag{6.3}$$

For example, for images with $f = 1$, **B** is defined as:

$$B = \begin{bmatrix} 0 & 0 & 1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{6.4}$$

Fourier Features are generated as follows:

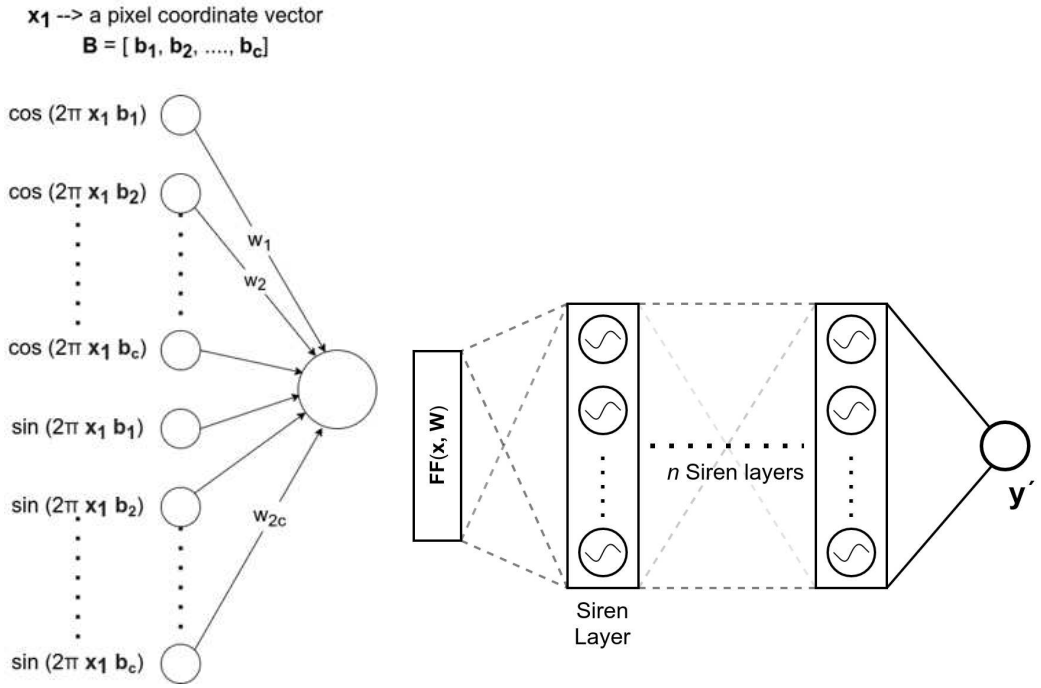
$$\mathbf{FF(x, W)} = \gamma(\mathbf{x}) \circ \mathbf{W}, \tag{6.5}$$

where $\circ$ is the element-wise product, $\mathbf{W} \in \mathbb{R}^{p\times 2c}$ is the weight matrix predicted by the FourierMask. Note that we flatten the spatial dimension of the prediction beforehand ($p = 2m \times 2m$). Let $\boldsymbol{ff}_i$ be the i_{th} column of **FF**; then the binary mask **y** is defined as

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \phi\left(\sum_{i=1}^{2c} \boldsymbol{ff}_i\right). \tag{6.6}$$

Here ϕ is the sigmoid activation function, which we use to bind the output between 0 and 1. Note that $\mathbf{y}(\mathbf{x}, \mathbf{W})$ can be interpreted as an implicit representation with a single

perceptron because it is a linear combination of Fourier features followed by a nonlinear activation function. This representation can be seen in figure 6.3a.



(a) Fourier perceptron (b) MLP with sine activation functions and FF input.

Figure 6.3

6.3.3 Proof of Fourier Mapping

The 2D Fourier series for a function with period $(1,1)$ is:

$$f(\mathbf{x}) = \sum_{\mathbf{n}\in\mathbb{Z}^2} c_{\mathbf{n}} e^{2\pi i \mathbf{n}\cdot\mathbf{x}},$$

here $c_{\mathbf{n}}$ are the Fourier series coefficients, given by

$$c_{\mathbf{n}} = \int_{[0,1]^2} f(\mathbf{x}) e^{-2\pi i \mathbf{n}\mathbf{x}} d\mathbf{x}.$$

In our case we are searching for the true underlying function, that predicts the binary mask of an object in the image. As we normalized the input domain, we only defined the function values on a finite subset of $[0,1]^2$. Therefore, we can and we will assume that the function is continuous and periodic over its input domain limits, resulting in a function with period $(1,1)$. As described above, we know that there exists a Fourier

representation of this function. We will show that the following equation holds:

$$f(\mathbf{x}) = \sum_{\mathbf{n}\in\mathbb{N}_0\times\mathbb{Z}} a_{\mathbf{n}}\cos(2\pi\mathbf{n}\cdot\mathbf{x}) + b_{\mathbf{n}}\sin(2\pi\mathbf{n}\cdot\mathbf{x}). \tag{6.7}$$

For the proof we will use Euler's formula and the fact, that for real-valued functions, $c_{\mathbf{n}} = \overline{c}_{\text{-}\mathbf{n}}$ where $\overline{c}_{\mathbf{n}}$ is the conjugate of $c_{\mathbf{n}}$. We are allowed to reorder the sum, therefore we get:

$$\begin{aligned}
&\sum_{\mathbf{n}\in\mathbb{Z}^2} c_{\mathbf{n}} e^{2\pi i\mathbf{n}\cdot\mathbf{x}} \\
&= \sum_{n_1\in\mathbb{N}}\sum_{n_2\in\mathbb{Z}} c_{\mathbf{n}} e^{2\pi i\mathbf{n}\cdot\mathbf{x}} + \sum_{n_1\in\mathbb{N}}\sum_{n_2\in\mathbb{Z}} c_{-\mathbf{n}} e^{-2\pi i\mathbf{n}\cdot\mathbf{x}} \\
&+ \sum_{n_1=0}^{0}\sum_{n_2\in\mathbb{Z}} c_{\mathbf{n}} e^{2\pi i\mathbf{n}\cdot\mathbf{x}}
\end{aligned}$$

$$\begin{aligned}
&\overset{\overline{c}_{\mathbf{n}}=c_{-\mathbf{n}}}{=} \sum_{\mathbf{n}\in\mathbb{N}\times\mathbb{Z}} 2\mathrm{Re}(c_{\mathbf{n}})\cos(2\pi\mathbf{n}\cdot\mathbf{x}) - 2\mathrm{Im}(c_{\mathbf{n}})\sin(2\pi\mathbf{n}\cdot\mathbf{x}) \\
&+ \sum_{\mathbf{n}\in\{0\}\times\mathbb{Z}} c_{\mathbf{n}} e^{2\pi i\mathbf{n}\cdot\mathbf{x}} \\
&= \sum_{\mathbf{n}\in\mathbb{N}\times\mathbb{Z}} 2\mathrm{Re}(c_{\mathbf{n}})\cos(2\pi\mathbf{n}\cdot\mathbf{x}) - 2\mathrm{Im}(c_{\mathbf{n}})\sin(2\pi\mathbf{n}\cdot\mathbf{x}) \\
&+ \sum_{\mathbf{n}\in\{0\}\times\mathbb{N}} 2\mathrm{Re}(c_{\mathbf{n}})\cos(2\pi\mathbf{n}\cdot\mathbf{x}) - 2\mathrm{Im}(c_{\mathbf{n}})\sin(2\pi\mathbf{n}\cdot\mathbf{x}) \\
&+ \sum_{\mathbf{n}\in\{0\}\times\{0\}} c_{\mathbf{0}} e^{2\pi i\mathbf{n}\cdot\mathbf{x}} \\
&= \sum_{\mathbf{n}\in\mathbb{N}_0\times\mathbb{Z}} a_{\mathbf{n}}\cos(2\pi\mathbf{n}\cdot\mathbf{x}) + b_{\mathbf{n}}\sin(2\pi\mathbf{n}\cdot\mathbf{x}),
\end{aligned}$$

where

$$\begin{aligned}
a_{\mathbf{0}} &= c_{\mathbf{0}}, \\
a_{\mathbf{n}} &= \begin{cases} 0 & \text{if } n_1 = 0 \wedge n_2 < 0 \\ 2\mathrm{Re}(c_{\mathbf{n}}) & \text{otherwise,} \end{cases} \\
b_{\mathbf{n}} &= \begin{cases} 0 & \text{if } n_1 = 0 \wedge n_2 < 0 \\ -2\mathrm{Im}(c_{\mathbf{n}}) & \text{otherwise.} \end{cases}
\end{aligned}$$

For practicality we limit the sum by some upper value f, in the following way:

$$S = \{0, \dots, f\} \times \{-f, \dots, f\} \setminus \{0\} \times \{-f, \dots, -1\},$$

where $\mathbf{B}$ is the matrix with the elements of the set S. With this, we can rewrite equation (6.7) to

$$f(x) = [\cos(2\pi x \cdot \mathbf{B}), \sin(2\pi x \cdot \mathbf{B})] \begin{pmatrix} a_{\mathbf{B}} \\ b_{\mathbf{B}} \end{pmatrix},$$

where $a_{\mathbf{B}} = (a_n)_{n \in \mathbf{B}}$ and $b_{\mathbf{B}} = (b_n)_{n \in \mathbf{B}}$. If we further define $W := \begin{pmatrix} a_{\mathbf{B}} \\ b_{\mathbf{B}} \end{pmatrix}^T$, then we can rewrite the equation to

$$f(x) = \sum_{y \in \gamma(x) \circ W} y.$$

In other words: We approximate the function by summing up the **FF(x,W)** in equation (6.5) in the dimension of coefficients c.
This proof was done for a single input x, obviously, this generalizes to multiple input data.

6.3.4 Fourier Features based MLP

As shown by (Sitzmann *et al.* (2020a); Tancik *et al.* (2020)), implicit representations can learn to generate shapes, images, etc. from input coordinates very effectively. Following the work from (Tancik *et al.* (2020)), we saw that Fourier mapping of input coordinates lets the MLP learn higher frequencies and consequently generate images with finer detail compared to MLPs without Fourier mapping. Furthermore, (Sitzmann *et al.* (2020a)) showed that using periodic activation functions works better compared to ReLU in implicit neural networks. We employ an MLP with sine activation functions, in which Fourier features (FF) are the input and mask $\mathbf{y}'$ is the output. We have 3 hidden layers (Siren layers), each with 256 neurons. The MLP has a single output neuron, on which we apply a sigmoid function to bind it between 0 and 1 (see figure 6.3b for illustration of architecture).

The Fourier features (FF) are generated by the equation (6.5) and they are parameterized by coefficients $\mathbf{W}$ learned by the network and therefore adapted for a specific input ROI. Although coordinate-based MLPs encode the information of one particular image or shape, by parameterizing them with learned Fourier coefficients $\mathbf{W}$ of each object, we can generalize them to generate a binary mask of any object in general.

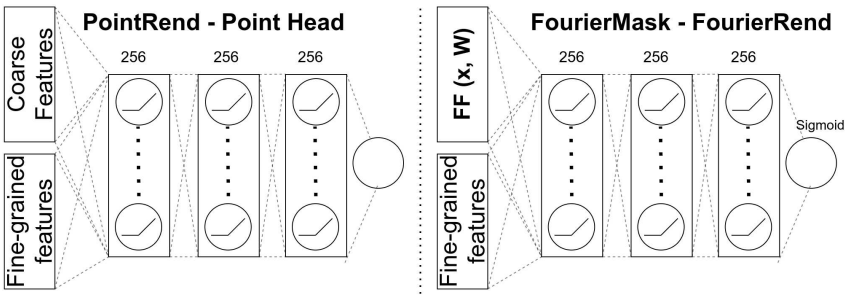


Figure 6.4: Difference between point head from PointRend and FourierRend.

6.3.5 MLP as a renderer - FourierRend

Apart from an Implicit MLP which predicts all pixels of the image, we also employed a renderer MLP (FourierRend) which specialized only on the uncertain regions of the mask predicted by equation (6.6). Most of the wrong (or uncertain) predictions fall near the boundary of objects, which hinders the capability of models to generate crisp masks. By focusing on the contour of objects, one could generate boundary-aligned masks, which has already been pointed out by PointRend (Kirillov *et al.* (2020a)). Moreover, such a rendering strategy saves precious computation and memory resources during inference.

We adopted the rendering strategy (sub-division inference) from PointRend and made the following modification in the PointHead (figure 6.4). Rather than sampling coarse mask features in the mask head, we sample the Fourier features (**FF**) from equation (6.5) at uncertain mask prediction coordinates (the locations where the predictions are near 0.5). We concatenate these Fourier features and fine-grained features (from the 'p2' level FPN feature map). We replace the mask predictions from equation (6.6) (coarse predictions), with the fine-detailed predictions from FourierRend. Consequently, we replace uncertain predictions at the boundary with more accurate predictions of the renderer, resulting in crisp and boundary-aligned masks.

6.3.6 Training and loss function

We concatenate the output $\mathbf{y}$ from equation (6.6) and $\mathbf{y}'$ from the MLP and train both masks in parallel. By training the output $\mathbf{y}$, we learn the coefficients $\mathbf{W}$ of a Fourier series in their true sense. We need these coefficients because we assume that the input for the MLP is Fourier features. We use IoU loss for training the binary masks defined as:

$$\textbf{IoU loss} = \frac{\sum_{i=0}^{N} \min(y_{p_i}, y_{t_i})}{\sum_{j=0}^{N} \max(y_{p_j}, y_{t_j})} \tag{6.8}$$

y_{p_i} is the predicted value of the pixel i, y_{t_i} is the ground truth value of the pixel i and N is the total number of pixels in the predicted mask.

6.4 Experiments

For all our experiments we employ a Resnet 50 backbone with a feature pyramid network pre-trained on ImageNet (Deng *et al.* (2009)) unless otherwise stated. We use the Mask R-CNN default settings from detectron2 (Wu *et al.* (2019)). We train on the MS COCO (Lin *et al.* (2014)) training set and show the results on its validation set. We predict class agnostic masks, i.e. rather than predicting a mask for each class in MS COCO, we predict only one mask per ROI. For the baseline, we trained a Mask R-CNN and PointRend (Kirillov *et al.* (2020a)) with class agnostic masks.

6.4.1 Spectrum Analysis MS COCO

To validate that the Fourier Mapping (equation (6.6)) works for instance mask prediction, we performed a spectrum analysis on the MS COCO training dataset. Along with verifying our method, this analysis gave us insight into an optimal number of frequencies for the dataset. We performed this experiment by applying a fast Fourier transform on

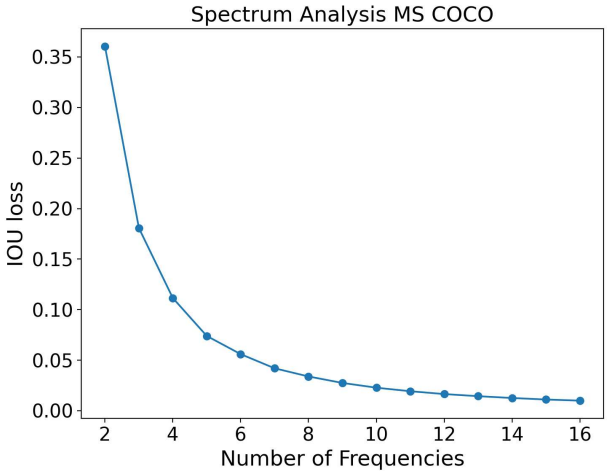
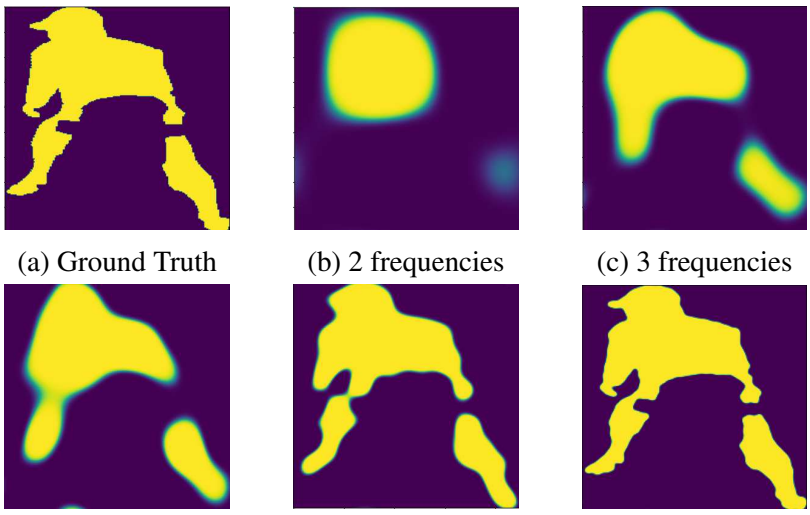


Figure 6.5: Spectrum test on COCO dataset.

all the target object masks in the COCO training dataset. This Fourier transform gave us the coefficients of a Fourier series, which hold the same meaning as the coefficients prediction **W** of FourierMask.

Firstly, we sampled only the lower frequency coefficients of the Fourier series and reconstructed the object's mask by applying equation (6.6). We did this for all the objects' masks in the COCO training set and evaluated the IoU loss of the reconstruction compared to the target. Then we incrementally added higher frequency coefficients and repeated the above-mentioned procedure until we reached the maximum number of frequencies.



(a) Ground Truth (b) 2 frequencies (c) 3 frequencies

(d) 4 frequencies (e) 8 frequencies (f) 25 frequencies

Figure 6.6: The ground truth vs its reconstructions at various frequencies.

Figure 6.5 shows the mean IoU loss at various frequencies. It can be seen that the loss decreases exponentially. We choose the maximum frequency as 12 since it has a low enough reconstruction loss and fits comfortably in our GPU memory. Figure 6.6 illustrates a visual comparison between the ground truth and reconstructions using varying numbers of frequencies.

Figure 6.7 shows the magnitude of the coefficients depicted as a heatmap. The segmentation mask (the top image in figure 6.7) is from a diagonally placed thin object, for example, a spoon on a table. The axis of the heatmap plot in the lower image represents the frequencies in the width and height direction of the image (Note that we do not show the negative frequencies here because of space constraints).

It can be seen that some of the diagonal frequencies have a higher magnitude (yellow or dark blue), while many of the frequencies have zero magnitudes (green). Such plots could be useful for those datasets which have a clear bias in their distribution towards certain shapes. Near-zero magnitude frequencies could be pruned, which could make models resource efficient and potentially faster. However, in this work, we conclude our investigation at this point.

6.4.2 Number of frequencies

To validate our experiment from the previous section, we trained a FourierMask with a similar configuration. Rather than predicting a set of coefficients for each pixel, we modified the architecture to predict a single vector for the whole image (see figure 6.8). We applied strided (stride=2) 3×3 convolutions 2 times (on the ROI) to reduce the feature size by 1/4th and then used a fully connected layer to predict the coefficients. The

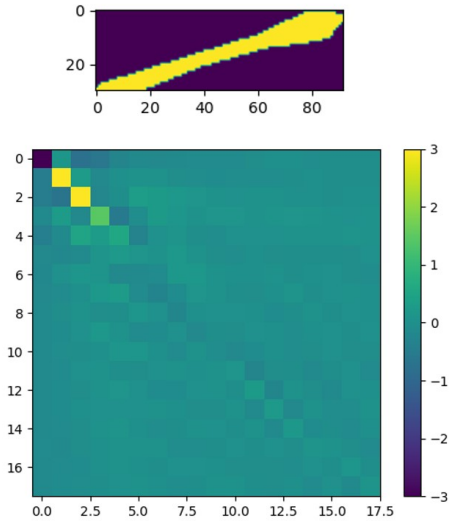


Figure 6.7

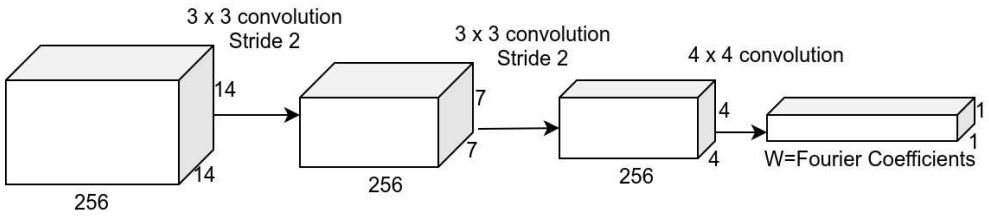


Figure 6.8: Modified FourierMask architecture with spatial size reduced to 1.

network architecture is shown in figure 6.8.

We applied the equation (6.5) and (6.6) for generating the mask. We copied the predicted Fourier coefficients p times to match the dimensions for matrix multiplication in equation (6.5). We trained the network with 12 frequencies and an output resolution of 56×56 using IoU loss. In this experiment, we did not add an MLP branch and trained only the equation (6.6).

We evaluated the mAP precision of the network on the COCO validation dataset when using a subset of Fourier component frequencies. The network was trained on 12 frequencies, but during inference, we incrementally added the higher frequency components starting from the first component. Due to memory limitations, we could not train our network with more than 12 frequencies. Figure 6.9 shows the result of this test. The mAP shows a similar trend as seen in figure 6.5 and therefore validates the spectrum analysis and the choice of 12 maximum frequencies. Figure 6.10 shows an example of how the masks change when different frequencies are used.

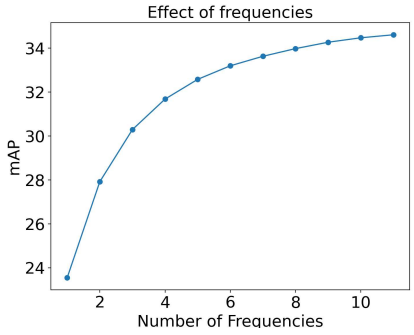
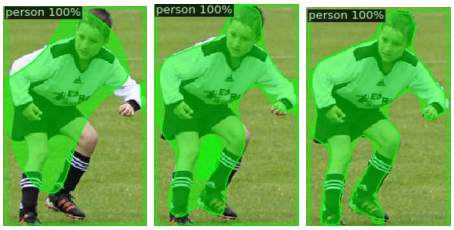


Figure 6.9: The mAP when using a subset of trained frequencies.



(a) 1 (b) 4 (c) 12

Figure 6.10: Mask predictions using various frequencies

Table 6.1: Comparison of various FourierMask architectures with Mask R-CNN.

Model	Backbone	mAP
Mask R-CNN	ResNet-50	34.86
FM	ResNet-50	34.89
FM + MLP	ResNet-50	34.97
FM + MLP	ResNeXt-101	39.09

6.4.3 Fourier Features based MLP

To validate that the Fourier Feature-based MLP improves the performance of FourierMask, we trained 2 networks with the architecture shown in figure 6.2. In this architecture, the network predicts separate Fourier coefficients for each spatial location. The first network was trained on the masks obtained using $\mathbf{y}$ in equation (6.6) and output $\mathbf{y}'$ of MLP (FM + MLP). The second network was trained only using equation (6.6) (FM). Both networks used 12 Fourier frequencies and had an output resolution of 28×28 pixels.

We used class agnostic masks and therefore predicted only one class for each region of interest, rather than a mask for each class in the COCO dataset. We had two hidden layers (both with sine activations and 256 neurons) and a single output neuron with sigmoid activation. For the first network (FM + MLP), we took the mean of the masks predicted by $\mathbf{y}$ (equation (6.6)) and an output $\mathbf{y}'$ of MLP during inference.

The results are shown in the table 6.1. As can be seen in the table, the network with an MLP shows the best performance among the models with the ResNet-50 backbone. We also trained the same network with a larger ResNeXt-101 (Xie *et al.* (2017)) backbone. The improvement of more than 4 mAP over the Resnet-50 model shows that our model scales well to bigger backbones.

(a) Step = 1 (b) Step = 1/2 (c) Step = 1/4 (d) Step = 1/8

Figure 6.11: Subsampling the pixels smoothes out the boundaries of the mask.

6.4.4 Higher resolution using pixel sub-sampling

One of the advantages of our method is that it can predict masks at sub-pixel resolution because implicit representations are continuous in the input domain. We analyzed this by evaluating both the trained networks in section 6.4.3 on the MS COCO validation set on various pixel steps. For the input $\mathbf{x}$ in the equation (6.1), rather than using integer values of pixels (pixel step of 1), we used a pixel step of $1/2^{s-1}$, where $s \in \mathbb{Z}^+$ is the scaling factor. This effectively scaled both the height and width of the input $\mathbf{x}$ by a factor of s. To match the size of input pixels $\mathbf{x}$, we upsampled the coefficients $\mathbf{W}$ in equation (6.5) in the spatial dimension using bilinear interpolation, by a scaling factor 2^{s-1}.

Table 6.2 shows the evaluation using the two networks explained in section 6.4.3. We can observe that using a lower pixel step improves the mAP.

Table 6.2: The effect of sub-sampling the pixels during inference. The speed is tested on NVIDIA GTX 2080Ti GPU.

Model	Pixel Step	Resolution	mAP	Speed (ms)
Mask R-CNN	1	28×28	34.86	48.7
FM	1	28×28	34.89	50.3
FM	1/2	56×56	35.13	59.1
FM	1/4	112×112	**35.18**	68.3
FM + MLP	1	28×28	34.97	52.1
FM + MLP	1/2	56×56	**35.18**	67.0

Figure 6.11 shows how the mask boundary smoothes out when sub-sampling the pixels. Note that we trained the network on 28×28 output resolution, but we can generate higher resolution output during inference, which is a considerable advantage over other methods. (See the ablation study for analysis with a varying number of frequencies at various pixel steps.)

6.4.5 Higher resolution using FourierRend

To generate higher resolution masks, we used FourierRend (section 6.3.5) along with the subdivision strategy from PointRend (Kirillov *et al.* (2020a)). Instead of using the coarse predictions of PointRend, we replaced them with the predictions from equation (6.6). This resulted in masks that were crisper and boundary aligned. For training FourierRend, we employ the default settings of the point selection strategy along with the point loss from PointRend. The results are shown in the table 6.3.

Here, we also evaluate the mask quality using the *Boundary IOU* (Cheng *et al.* (2021)) metric (mAP_{bound}), which penalizes the boundary quality more than overall correct pixels. Compared to Mask R-CNN, we see a decent improvement of more than 0.7 mAP_{mask} and 1.6 mAP_{bound} with comparable speeds. We can clearly see visual improvements especially in boundary quality (see figure 6.1). Compared to PointRend, we observe that the masks are more complete (see figure 6.1) with a reasonably lower inference time.

Table 6.3: The effect of subdivision inference.

Model	Sub. steps	Resolution	mAP_{mask}	mAP_{bound}	Speed (ms)
Mask R-CNN	0	28×28	34.86	21.2	48.7
FourierRend	0	28×28	35.01	21.0	48.7
FourierRend	1	56×56	35.63	22.8	52.4
FourierRend	2	112×112	35.64	22.8	55.7
FourierRend	3	224×224	35.64	22.9	59.4
PointRend	5	224×224	36.12	23.5	81.6

6.5 Ablation studies

6.5.1 Binary cross entropy vs IoU loss

We trained the network shown in figure 6.8 using binary cross entropy loss rather than IoU loss. We reached a mean average precision of 32.1 which was clearly lower than the IoU loss. Therefore, we used IoU loss for the rest of our experiments.

6.5.2 ReLU vs Sinusoidal activations

We needed to investigate if sinusoidal activations in MLP indeed perform better than MLP with ReLU activations. Therefore, we trained a network with the same settings and architecture as in the section 6.4.3, but replaced sine activations with ReLU. Table 6.4 clearly shows that ReLU activation showed poor performance compared to sine.

Table 6.4: Comparison between Sine and ReLU.

Model	Backbone	mAP
FM + MLP (Sine)	ResNet-50	**34.97**
FM + MLP (ReLU)	ResNet-50	34.41

6.5.3 Sub-sampling and frequency analysis

Similar to the section 6.4.2 in the paper, we wanted to test the performance of our networks when using only a subset of frequencies. Previously, we performed the experiment on the network shown in figure 6.8, which has one set of coefficients for all pixel locations. For the network in figure 6.2 (with a separate set of coefficients for each pixel location), we performed the same experiment. Additionally, we repeated the experiment with various pixel steps. Figure 6.12 shows the comparison of these networks. We can

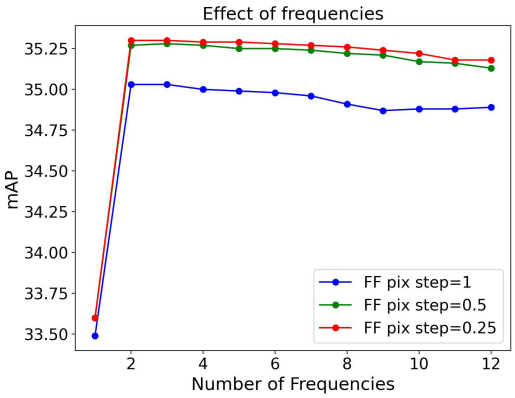


Figure 6.12: The mAP on the COCO validation dataset when using a subset of trained frequencies using different pixel steps.

see that the networks with a smaller pixel step always show better performance. Moreover, we observe that till 2-3 frequencies, performance improves for all the variants, and afterward, the performance saturates and slightly decreases. The saturation could be attributed to the fact that we are predicting a set of coefficients for each pixel location and higher frequency coefficients are not needed in this case (may also add noise to the output in some simple masks).

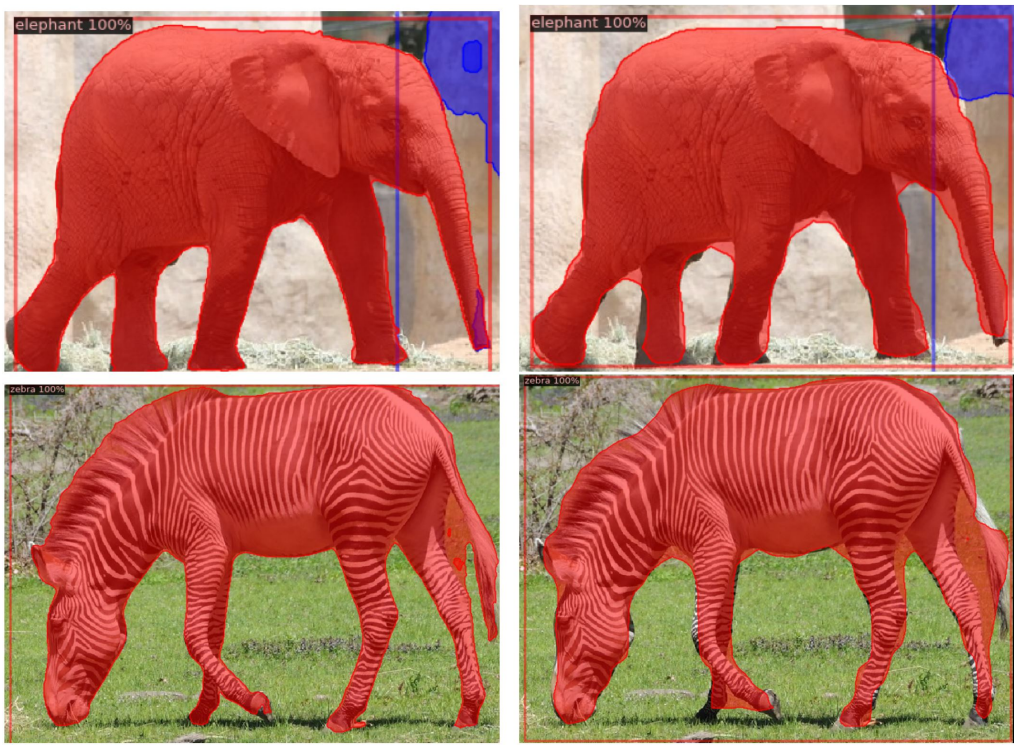


Figure 6.13: Left image FourierRend, Right image Mask R-CNN

6.6 Conclusion

In this chapter, we showed how implicit representations combined with the Fourier series can be applied to the task of instance segmentation to generate high-quality masks. We illustrated that the masks generated using our Fourier mapping are compact and meaningful. The lower Fourier frequencies hold the shape and higher frequencies hold the sharp edges. Furthermore, by sub-sampling the pixel coordinates in our implicit MLP, we can generate higher resolution masks during inference, which are visually smoother and improve the mAP over our baseline Mask R-CNN with similar settings and model capacity. We also show that our renderer MLP FourierRend improves the boundary quality of FourierMask significantly and consequently the quantitative performance is improved. Figure 6.13 shows a qualitative comparison between FourierRend and Mask R-CNN.

Chapter 7

Conclusion

This work focused on various representations which play important roles in the quality of object detection and instance segmentation. Particularly, we investigated how Fourier representations could help in improving the efficiency of models and at the same time provide a deeper understanding of deep learning frameworks employed in segmentation tasks.

In chapter 3, we showed that weakly supervised models could be used for bounding box detection in thermal camera images. Since labeling is a major bottleneck in training machine learning models, weak supervision using only image-level labels provided a more effective way to train our models. Also, we illustrated that the Class activation maps (CAMs) representation provides a reasonable estimation for localizing humans in thermal camera images. We also observed some limitations in the case of occluded persons because CAMs represent all objects of the same class as a single contour.

In chapter 4, we used a Fourier series representation of segmentation masks in a single-stage object detector framework. We showed that the mask information could be encoded in Fourier coefficients efficiently and our *FourierNet* could thus be a replacement for other single-stage segmentation heads. Our *differentiable shape decoder* allows the segmentation masks to be trained directly on the shape of the object and achieve automatic weight balancing of the Fourier coefficients. We illustrated that the low-frequency components of the Fourier Series contain the overall shape information of the object and higher frequencies hold the edges and corners. Therefore, our representation could be adapted to the use case, where a smaller model with only low-frequency components could be used for tasks with low requirements (e.g localizing) and larger models could be employed for generating crisp masks. One of the limitations of the polar representation that we experienced was its inability to represent non-star-shaped objects. This was tackled in our work later.

In chapter 5, we show the connection of the Fourier series to implicit neural representations. We introduced an integer Fourier mapping and proved that a perceptron with this mapping is equivalent to a Fourier series. We showed that a Fourier-mapped perceptron is structurally like a SIREN with one hidden layer. Furthermore, we showed that the integer mapping forces periodicity of the network output. We also proved that the main contributor to the mapping performance are the number of elements and the

standard deviation of the mapping frequencies. This chapter forms the basis of using implicit neural networks for image-related tasks and thus makes a reasonable argument to employ implicit representations for instance segmentation.

In chapter 6, we developed a mask head *FourierMask* for two-stage object detectors. FourierMask predicts the coefficients of a Fourier series, which are then provided to an implicit neural representation. We showed that implicit representations could be employed for the task of instance segmentation. We showed that we can sub-sample the pixel coordinates to generate higher resolution masks during inference. We verified and illustrated that the rendering strategy from PointRend Kirillov *et al.* (2020a) brings significant qualitative gains for FourierMask. Our renderer MLP *FourierRend* improves the mask boundary of FourierMask significantly.

This work provided some insights and investigated different representations for object detection and instance segmentation. We hope that the experiments and findings prove fruitful for further research in the related fields.

Abbreviations

3D	3 Dimensional
AP	Average Precision
ANN	Artificial Neural Network
CAM	Class Activation Maps
CF	Centerness Factor
CNN	Convolutional Neural Networks
COCO	Common Objects in COntext
CR	Coefficient Regression
CTF	Coarse-to-fine training
DFT	Discrete Fourier Transform
DSD	differentiable shape decoder
FCOS	Fully Convolutional One-Stage object detector
FFT	Fast Fourier Transform
FC	Fully connected
FF	Fourier Features
FM	Fourier Mapping
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
FPS	Frames per second
GAP	Global Average Pooling
GC	Gaussian Centerness
GPU	Graphics processing unit
HTC	Hybrid Task Cascade
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
INR	Implicit neural representation
IoU	Intersection-over-Union
mAP	mean Average Precision
MLP	Multi-Layer Perceptron
NC	Normalized Centerness
NTK	Neural Tangent Kernel
NVS	Novel View Synthesis

PC	Polar Centerness
PE	Positional Encoding
PSNR	Peak signal-to-noise ratio
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RoI	Region-of-Interest
RPN	Region Proposal Network
SIREN	SInusoidal REpresentation Networks
SGD	Stochastic Gradient Descent
TP	True Positive
WI	Weight Initialization
YOLO	You only look once

Bibliography

Atzmon, M. and Lipman, Y. (2020). Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574.

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019). Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9157–9166.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.* (2020). Language models are few-shot learners. *Advances in neural information processing systems*, **33**, 1877–1901.

Chabra, R., Lenssen, J. E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., and Newcombe, R. (2020). Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer.

Chen, K. *et al.* (2019a). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., *et al.* (2019b). Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983.

Chen, Z. (2019). *IM-NET: Learning implicit fields for generative shape modeling*. Ph.D. thesis, Applied Sciences: School of Computing Science.

Chen, Z. and Zhang, H. (2019). Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948.

Cheng, B., Girshick, R., Dollár, P., Berg, A. C., and Kirillov, A. (2021). Boundary IoU: Improving object-centric image segmentation evaluation. In *CVPR*.

Deng, B., Lewis, J. P., Jeruzalski, T., Pons-Moll, G., Hinton, G., Norouzi, M., and Tagliasacchi, A. (2020). Nasa neural articulated shape approximation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 612–628. Springer.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

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

Fan, H., Su, H., and Guibas, L. J. (2017). A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613.

FLIR (2022). Thermal imaging cameras.

Gallant, A. R. and White, H. (1988). There exists a neural network that does not make avoidable mistakes. In *ICNN*, pages 657–664.

Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W. T., and Funkhouser, T. (2019). Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164.

Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T. (2020). Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866.

Gropp, A., Yariv, L., Haim, N., Atzmon, M., and Lipman, Y. (2020). Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Henzler, P., Mitra, N. J., and Ritschel, T. (2020). Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8356–8364.

Huang, Z., Huang, L., Gong, Y., Huang, C., and Wang, X. (2019). Mask scoring r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6409–6418.

Jacot, A., Hongler, C., and Gabriel, F. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*.

Janai, J., Güney, F., Behl, A., and Geiger, A. (2017). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Arxiv*, pages arXiv–1704.

Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T., *et al.* (2020). Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirillov, A., Wu, Y., He, K., and Girshick, R. (2020a). Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kirillov, A., Wu, Y., He, K., and Girshick, R. (2020b). Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808.

Kuo, W., Angelova, A., Malik, J., and Lin, T.-Y. (2019). Shapemask: Learning to segment novel objects by refining shape priors. *arXiv preprint arXiv:1904.03239*.

Lee, Y. and Park, J. (2019). Centermask: Real-time anchor-free instance segmentation. *arXiv preprint arXiv:1911.06667*.

Lee, Y. and Park, J. (2020). Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13906–13915.

Liang, J., Homayounfar, N., Ma, W.-C., Xiong, Y., Hu, R., and Urtasun, R. (2019). Polytransform: Deep polygon transformer for instance segmentation. *arXiv preprint arXiv:1912.02801*.

Liang, J., Homayounfar, N., Ma, W.-C., Xiong, Y., Hu, R., and Urtasun, R. (2020). Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9131–9140.

Lin, C.-H., Ma, W.-C., Torralba, A., and Lucey, S. (2021). Barf: Bundle-adjusting neural radiance fields. *arXiv preprint arXiv:2104.06405*.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liu, S. (2013). Fourier neural network for machine learning. In *2013 International Conference on Machine Learning and Cybernetics*, volume 1, pages 285–290. IEEE.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768.

Liu, S., Saito, S., Chen, W., and Li, H. (2019). Learning to infer implicit surfaces without 3d supervision. *Advances in Neural Information Processing Systems*, **32**, 8295–8306.

Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., and Cui, Z. (2020). Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470.

Michalkiewicz, M., Pontes, J. K., Jack, D., Baktashmotlagh, M., and Eriksson, A. (2019). Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer.

Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2019). Occupancy flow: 4d reconstruction by learning particle dynamics. In *International Conference on Computer Vision*.

Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515.

Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., and Geiger, A. (2019a). Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540.

Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., and Geiger, A. (2019b). Texture fields: Learning texture representations in function space. In *International Conference on Computer Vision*.

Osgood, B. G. (2019). *Lectures on the Fourier transform and its applications*, volume 33. American Mathematical Soc.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174.

Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Riaz, H., Benbarka, N., and Zell, A. (2021). Fouriernet: Compact mask representation for instance segmentation using differentiable shape decoders. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7833–7840, Los Alamitos, CA, USA. IEEE Computer Society.

Riaz, M., Benbarka, N., Hoefer, T., Zell, A., *et al.* (2022). Fouriermask: Instance segmentation using fourier mapping in implicit neural networks. *Image Analysis and Processing – ICIAP 2022*, pages 587–598.

Rother, C., Kolmogorov, V., and Blake, A. (2004). " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, **23**(3), 309–314.

Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., and Li, H. (2019). Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314.

Silvescu, A. (1999). Fourier neural networks. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 1, pages 488–491. IEEE.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. (2019). Scene representation networks: continuous 3d-structure-aware neural scene representations. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 1121–1132.

Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., and Wetzstein, G. (2020a). Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020b). Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, **33**.

Skorokhodov, I., Ignatyev, S., and Elhoseiny, M. (2021). Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10753–10764.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, **8**(2), 131–162.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). Fcos: Fully convolutional one-stage object detection. *arXiv preprint arXiv:1904.01355*.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. `https://github.com/facebookresearch/detectron2`.

Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., and Luo, P. (2020). Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12193–12202.

Xie, S., Girshick, R. B., Dollár, P., Tu, Z., and He, K. (2016). Aggregated residual transformations for deep neural networks. *CoRR*, **abs/1611.05431**.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xu, W., Wang, H., Qi, F., and Lu, C. (2019). Explicit shape encoding for real-time instance segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Yan, K., Tang, Y., Peng, Y., Sandfort, V., Bagheri, M., Lu, Z., and Summers, R. M. (2019). Mulan: multitask universal lesion analysis network for joint lesion detection, tagging, and segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 194–202. Springer.

Yang, Z., Xu, Y., Xue, H., Zhang, Z., Urtasun, R., Wang, L., Lin, S., and Hu, H. (2019). Dense reppoints: Representing visual objects with dense point sets. *arXiv preprint arXiv:1912.11473*.

Ying, H., Huang, Z., Liu, S., Shao, T., and Zhou, K. (2019). Embedmask: Embedding coupling for one-stage instance segmentation. *arXiv preprint arXiv:1912.01954*.

Yu, J., Jiang, Y., Wang, Z., Cao, Z., and Huang, T. (2016). Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

Zhou, X., Zhuo, J., and Krahenbuhl, P. (2019a). Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 850–859.

Zhou, X., Wang, D., and Krähenbühl, P. (2019b). Objects as points. *arXiv:1904.07850*.

Zhou, Y., Zhu, Y., Ye, Q., Qiu, Q., and Jiao, J. (2018). Weakly supervised instance segmentation using class peak response. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3800.