Instance Segmentation With Contrastive Self-supervised Learning

Mohammadmatin Kaviani

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical engineering Engineering) at

Concordia University

Montréal, Québec, Canada

November 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Mohammadmatin Kaviani**
Entitled:	**Instance Segmentation With Contrastive Self-supervised Learning**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical engineering Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair
Dr. Chunyan Wang

______________________________ External Examiner
Dr. Yiming Xiao

______________________________ Examiner
Dr. Chunyan Wang

______________________________ Supervisor
Dr. Wei-Ping Zhu

______________________________ Co-supervisor
Dr. Iman Moazzen

Approved by ______________________________
Yousef Shayan, Chair
Department of Electrical and Computer Engineering

__________ 2022 ______________________________
Dr. Debabbi, Dean
Faculty of Engineering and Computer Science

Abstract

Instance Segmentation With Contrastive Self-supervised Learning

Mohammadmatin Kaviani

Object detection or localization gradually progresses from coarse to fine digital image inference. It provides the classes of the image objects and the location of the image objects that have been categorized. The location is provided in the shape of bounding boxes or centroids. Semantic segmentation provides fine inference by indicating labels for every pixel in the input image. Each pixel is labelled according to the object class within which it is surrounded. Moreover, instance segmentation gives different sets of pixels of interest for separate instances of objects. Thus, instance segmentation is defined as solving the problem of object detection and semantic segmentation simultaneously.

This thesis is introducing a new self-supervised framework for instance segmentation and background removal. We make use of a state-of-art self-supervised method called bootstrap your own latent for our pretraining section and then by transfer learning, we transfer the representation learnt in this pretraining to downstream task which is instance segmentation using Mask R-CNN. Two other existing state-of-art methods of instance segmentation, namely, instance segmentation scheme using Mask R-CNN along with random initial weights and that with ImageNet initial weights, respectively are implemented and compared to our proposed framework. Comparing our method with those known methods in instance segmentation and background removal, we find out that self-supervised learning outperforms both methods despite using no labelled data in pretraining. Our experimental results indicate that the proposed framework outperforms the instance segmentation and background removal using ImageNet initial weights and random initial weights by 0.866% and 14.06% ,respectively, in average precision (AP). Moreover, the proposed self-supervised framework has been designed to minimize the computations and the need for annotated datasets. This design demonstrates that the proposed system can perform high-quality processing at a meagre computation cost for many applications. Hence, instance segmentation using self-supervised techniques is a practical approach to lowering the barrier of computation resource requirements and labelled datasets to make them more implementable and applicable to the general public.

Acknowledgments

I would like to express my best appreciation to my supervisor Dr. Wei-Ping Zhu and my co-supervisor Dr. Iman Moazzen for their support during my study in Concordia University. They always give me brilliant guidance when I need them. Moreover, Dr. Zhu's focus on details in research influences me deeply. It is my great honor and fortune to work under his supervision. I would also like to thank my friends, Ali Salmasi, for the help in the last two years. Additionally, special thanks to my parents and my uncle who were beside me during this tough time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background information

Image segmentation, such as instance segmentation and semantic segmentation, is an important element in different visual understanding systems. It concerns partitioning pictures (or video frames) into multiple components or objects Szeliski (2010). Segmentation plays a significant part in a wide range of applications Forsyth and Ponce (2011), including medical image analysis (e.g., tumour boundary extraction and measurement of tissue volumes), autonomous vehicles (e.g., navigable surface and pedestrian detection), video surveillance, and background removal to name a few. In other words, Image segmentation is the task of clustering components of an image that belong to the identical object category. This procedure is also called pixel-level classification. Classification may be described as the procedure of predicting a complete input, i.e., prediction of the class of an object in the image or delivering a list of classes of objects in an image based on their classification scores. Detection or object localization is an incremental step from coarse to fine inference, which provides categories of the image objects and delivers the location of the classified image objects in the shape of bounding boxes or centroids.

As we mentioned earlier, image segmentation can be pictured as a classification problem of pixels with semantic labels (semantic segmentation) or partitioning of separate objects (instance segmentation). Semantic segmentation conducts pixel-level labelling with a set of object classes (e.g., animal, human, table, laptop) for all image pixels. Thus, it is typically a more challenging task than image classification, which indicates a single label for the whole image. This segmentation aims to obtain acceptable inference by assigning labels for each image pixel. Every pixel is class-labelled according to the object or region it is surrounded by. Instance segmentation further develops the semantic segmentation coverage by detecting and drafting each object of interest in the image (e.g., partitioning of individual cars). Simply, instance segmentation can be defined as the task of finding simultaneous solution to object detection as well as semantic segmentation.

One of the important problems in computer vision is salient object detection. This problem can also be called image background removing, as in this problem foreground extraction or background removal is the principal object. In recent years, salient object detection (SOD) has drawn a great number of computer vision researchers' attention. As mentioned, the goal of this task is to segments objects in the foreground as well as classify the corresponding pixels and detect the background at the same time. Before deep learning methods have been developed, this task had been done by traditional approaches such as thresholding Otsu (1979), histogram-based bundling, region-growing Nock and Nielsen (2004), k-means clustering Dhanachandra, Manglem, and Chanu (2015), watersheds Najman and Schmitt (1994), to more advanced algorithms such as active contours Kass, Witkin, and Terzopoulos (1988), graph cuts Boykov, Veksler, and Zabih (2001), conditional and Markov random fields Plath, Toussaint, and Nakajima (2009), and sparsity-based Starck, Elad, and Donoho (2005)- Minaee and Wang (2019) methods.

However, nowadays deep learning methods are of interest to numerous researchers and scientists, this method is the most widely used technique to resolve salient object detection related subjects. This task widely applied to a great number of applications which investigates objects in foreground and has tons of uses. For instance, in computer vision, applications like image understanding, image captioning, object detection, semantic segmentation, instance segmentation, video summarization, etc. can be named. In computer graphics, SOD plays a significant role in non-rendering photorealistic, image retargeting, image recroping and so on. SOD has other applications in medical domains Wang et al. (2021).

Therefore, improvement of deep learning methods has increased the use of this technique in recent years in SOD problems. After the first use of deep learning in solving SOD problems in 2015, SOD algorithms based on deep learning represented marvelous results compared with conventional methods.

In figure 1.1 the comparison of visual recognition tasks in computer vision is shown. Sub figure (a)

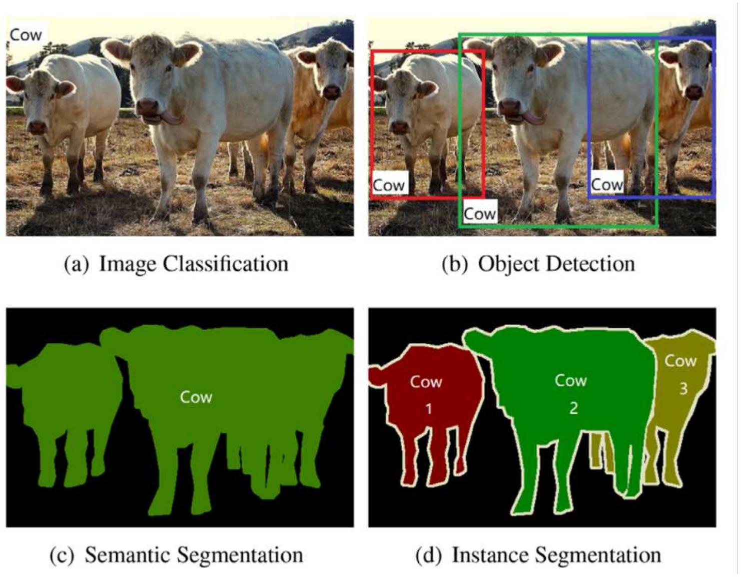


Figure 1.1: Comparison of different visual recognition tasks in computer vision X. Wu et al. (2020)
.

represents image classification in which only labels are assigned to the picture. In sub figure (b) object detection predicting labels as well as drawing bounding boxes to localize objects, is shown. Sub figure (c) and (d) are showing 2 types of segmentation. In (c), semantic segmentation classifies objects without differentiating object instances and in (d), instance segmentation differentiates objects by assigning pixel-level segmentation tasks X. Wu et al. (2020).

1.2 Motivation and contributions

In order to train deep neural networks to obtain better functionality and performance while solving different problems of computer vision, having a considerable number of labelled data is mandatory. To avoid collecting and annotating massive datasets, self-supervised learning as a branch of unsupervised learning methods has been developed to learn a good representation of unlabeled images. Due to the excellent capability of Convolutional Neural Networks (CNN) in learning features at different levels of visual representation, these networks are considered the foundation of many computer vision tasks like object detection, semantic segmentation, instance segmentation, etc. Models that were already trained on massive datasets like ImageNet are mostly known as pre-trained methods and are fine-tuned to solve various computer vision problems. There are two reasons these pre-trained models on ImageNet dataset Deng et al. (2009), are used as mentioned below:

- Learnt parameters from datasets with various data provide a great initial point to do different mainstream tasks such as object detection and image segmentation. Hence, training networks for computer vision problems like instance segmentation, semantic segmentation, etc, are done much faster, and the defined objective function or loss function converges to the optimum point more quickly.

- Pretrained networks on massive datasets have learned good hierarchical features, and this can prevent the over-fitting issue in computer vision tasks. The importance of this matter would be even more visible when the labelled dataset is not as huge Jing and Tian (2020).

The performance of deep convolutional networks is affected by the depth of the network and the amount of training data. Different types of these networks have been developed to increase the generalization of the models, these including AlexNet Simonyan and Zisserman (2014), VGG Szegedy et al. (2015), Resnet Ku, McKinley, and Kenney (2002). In addition, significant datasets like ImageNet and OpenImage Neumann, Pinto, Zhai, and Houlsby (2020) have also been created which we can use to train our models. With these models and training datasets, outstanding precisions have been acquired in most computer vision problems Girshick, Donahue, Darrell, and Malik (2014) Long, Shelhamer, and Darrell (2015) Vinyals, Toshev, Bengio, and Erhan (2015). However, collecting and annotating massive datasets is too time-consuming and costly. For instance, a massive dataset used to train deep convolutional neural networks is named ImageNet, containing about 1.3 million 2-dimensional images of 1000 classes. Humans manually labelled this dataset, and this procedure takes a very long time for a human being to annotate the whole dataset. Compared to natural image datasets, labelling other data sets like videos or overhead imagery and medical images is more challenging to annotate. Accordingly, we were motivated to utilize one of the best self-supervised learning methods, which have been developed in the past two years, without manual labelling and utilizing pre-trained models as initial weights to better solve our SOD related problem. In this thesis, we used a specific algorithm called BYOL, a self-supervised learning or unsupervised learning based on contrastive learning, to learn visual representations. COCO 2014 dataset was utilized for pretraining. Then the weights obtained from this process are used to solve our downstream task, which is instance segmentation followed by background removal. In the training section, 20,000 images of COCO 2017 dataset are used. ImageNet pre-trained models have been widely used for a long time in numerous computer vision applications in supervised ways, and acceptable results have been acquired. This thesis aims to replace these supervised pre-trained models with self-supervised ones. Solving instance segmentation and background removal problems proves the effectiveness of this method compared with pre-trained models on the ImageNet dataset. Our contributions are as follows:

- We have verified the generalizability of self-supervised learning methods for solving background removal and instance segmentation problems using the BYOL algorithm..
- We have suggested simple and minor changes to the BYOL algorithm to reduce convergence time in pre-training to an acceptable level.
- We used the weights obtained using the BYOL algorithm to solve the instance segmentation and SOD problem and compared the obtained model with the state where the initial weights of the ImageNet database are obtained in a supervised way.
- For the first time, we use self-supervised pre-training methods to solve SOD-related problems and examine the generalizability of contrastive self-supervised learning methods to SOD and segmentation problems. In solving the instance segmentation and SOD problems, we have used only a relatively small part of the coco 2017 dataset and assumed that it is challenging to obtain labelled data for these problems, which is a correct assumption.

1.3 Scope and organization

To extract useful features from input images and classify the pixels accurately for segmentation applications, without need of having a large amount of labeled data, we could select better initial point to start our training and this comes true by generating the initial weights of our mainstream tasks using self-supervised learning. The contribution of this thesis contains 2 main parts:

- Obtaining initial weights by pretraining a deep neural network using unlabelled COCO dataset 2014 to better start our segmentation.

- Training a deep neural network with mentioned initial weights using COCO dataset 2017 in order to remove backgrounds of still images as well as segmenting instances in the end.

This thesis's organization is explained in the following. Description of self-supervised learning, pretext tasks, contrastive learning and a brief history of instance segmentation are given in chapter 2. The proposed method to segment objects and remove background with detailed explanation of the algorithm is shown in chapter 3. Chapter 4 provides the evaluation, numerical and visual comparisons. In particular, the comparison is made between three methods, segmentation using self-supervised pretrained models, segmentation using Imagenet pretrained model, and segmentation using random initial weights. A conclusion of the research is presented in chapter 5.

Chapter 2

Previous work using deep learning in instance segmentation

2.1 Introduction

Having expandable pretrained features causes scientists to be able to transfer obtained weights to different problems and datasets. This enables computer vision problems to be solved more accurately and decreases need of massive labelled datasets and reduce convergence time as well. Transfer learning is an approach in which a model is trained in a big dataset, and if the model reaches satisfactory results in defined problem, we could transfer the weights to other problems and similar datasets. Therefore, pre-training is usually performed on a large dataset called the upstream dataset, and then the knowledge gained using this process is transferred to another set of data called downstream problems. For example, pretrained weights on the ImageNet dataset, which is for 1000-class classification, are transferable to related tasks such as semantic segmentation Girshick et al. (2014), object recognition Long et al. (2015), etc., and on these downstream tasks, we fine tune ImageNet initial weights, and as we mentioned, in this case the need for labeled data is very low. Pretrained weights on the ImageNet dataset are often transferred to other computer vision issues, but other approaches can be considered to avoid supervised training. One of these approaches is to use unsupervised methods to pre-train the features and then transfer the learning. To learn the visual features of unlabeled data, a general solution is to define various auxiliary problems to be solved by neural networks, when these networks are able to reach the objective of such problems by minimizing the cost function, they learn well and these weights can be used for other computer vision problems as well. Auxiliary problems that have been developed for learning visual features include coloring gray images Zhang, Isola, and Efros (2016), painting images Pathak, Krahenbuhl, Donahue, Darrell, and Efros (2016), image puzzle problems Noroozi and Favaro (2016), and so on. Auxiliary problems should usually have two important characteristics, which are:

(1) One can capture the features of videos or images by solving auxiliary problems (pretext tasks) by convolutional networks.

(2) For pretext tasks, pseudo-tags can be extracted using the data themselves based on the features and characteristics of the video and video data sets, automatically or semi-automatically without human intervention Jing and Tian (2020).

The general trajectory of how to use self-monitoring learning methods is shown in figure 2.1.

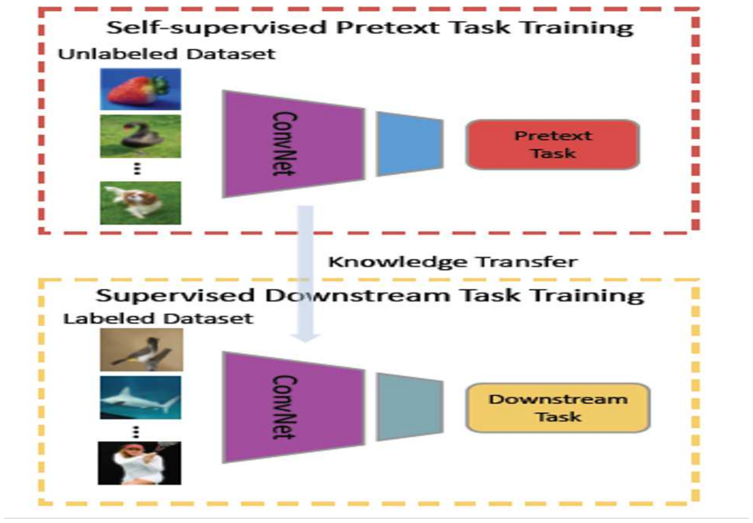


Figure 2.1: Framework of self-supervised learning.

During the self-supervised pre-training phase, a pretext task is designed for convolutional neural networks to solve, and pseudo-tags of this problem are obtained automatically based on the characteristics of the dataset; The convolutional network is then trained to solve the objective function of these tasks. After the self-supervised training phase is completed, the learned features can be transferred to the main tasks, which are also called downstream tasks. For core tasks, we usually have datasets with few samples; For this reason, pre-learned networks using self-supervised learning methods are employed, which avoids the problem of over-fitting the data of the main issue. In general, the initial layers of neural networks learn general features such as edges, corners, and textures; The deeper layers learn the top-level features associated with the task [6]. In this chapter, we first define some of the concepts and terminologies in learning methods, then describe self-supervised learning methods based on pretext-task solving, and finally introduce contrastive self-supervised learning methods as well as the common architectures.

2.2 Term definition

First, we identify and introduce the words in this area, to make it easier for the reader to continue.

Human labels: Refers to the labels provided by humans.

Pseudo-labels: Labels that are generated automatically or semi-automatically based on the dataset characteristics used without human intervention or with minimal human intervention Jing and Tian (2020).

Pretext tasks: Tasks are pre-designed to be solved by neural networks, and visual properties are learned by solving the objective function of these tasks Jing and Tian (2020).

Downstream task: They are computer vision applications that are used to evaluate the features learned using self-monitoring learning methods. These applications can be highly learned from pre-taught models; Benefit when human labeled data is rare. In general, human tags are necessary to solve the downstream problem. However; In some applications, the downstream problem can be equated with the auxiliary problem, in which case data labels are not used to solve the auxiliary problem Jing and Tian (2020).

Supervised learning: Supervised learning is a technique in which human-annotated labels are used to train a network Jing and Tian (2020).

Unsupervised learning: Refers to methods that do not use human-produced labels to solve the problem Jing and Tian (2020).

Semi-supervised learning: Refers to learning methods that use a small amount of labeled data along with large amounts of unlabeled data to learn features.

Weakly-supervised learning: Refers to feature learning methods that use weak or incorrect labels to solve a problem that normally requires stronger labels Jing and Tian (2020).

Self-supervised learning: These methods are a subset of unsupervised feature learning methods learning. By its classical definition, refers to feature learning methods in which neural networks are explicitly trained using data whose labeling process is automated. The second definition of self-supervised learning methods is that if we use a method to learn features that uses parts of the data available to us to estimate other parts or information of other parts of the data, our method can be a self-supervised learning method Jing and Tian (2020).

Because no human annotations are required to produce pseudo labels during self-monitoring training, largescale unlabelled datasets can be operated for self-supervised training. self-supervised approaches acquired good results and the gap with supervised techniques in performance on downstream tasks gets smaller by training with these pseudo labels.

2.3 Commonly-used feature learning methods

Based on data labelling types, feature learning methods can fall into one of four categories: supervised learning, semi-supervised learning, weakly supervised learning, and unsupervised learning. In this section, we will explain the different learning methods in more detail.

2.3.1 Supervised leaning

Having a data set such as X, in the supervised learning for each instance of the X_i data from the X dataset, there is also a corresponding man-made tag with the Y_i attribute. Having an N set of training data, i.e. $D = \{X_i, i = 0, ..., N\}$, the cost function of training data at the time of training is defined as follows:

$$\operatorname{loss}(D) = \min_{\Theta} \frac{1}{N} \left(\sum_{i=1}^{N} \operatorname{loss}\left(X_i, Y_i\right) \right) \tag{1}$$

Using man-made labels, supervised learning methods have shown good solutions to many computer vision problems Szeliski (2010) Forsyth and Ponce (2011) Otsu (1979) Donahue, Krähenbühl, and Darrell (2016). However; Collecting and tagging large datasets is usually very costly and difficult, and in some cases, such as medical and satellite imagery, should be done by experts in the field of tagging. For this reason, semi-supervised learning methods, weakly supervised learning and non-supervised learning methods have been proposed to minimize the cost of data labelling Jing and Tian (2020).

2.3.2 Unsupervised learning

Learning without supervision refers to methods of learning features that perform learning without the need for man-made labels. This learning method involves two approaches: (1) completely observer-free learning, in which the methods used do not require labels, (2) and self-supervised learning, which uses automatically generated pseudo-labels and learn the properties of data. In this thesis, we will focus exclusively on self-supervised learning methods.

2.3.3 Semi-supervised learning

To learn visual representations in a semi-supervised way, with a small tagged data set X and a large unlabeled dataset such as Z, for each instance of the X_i data set in the X set, the corresponding human-generated label with the Y_i attribute also exists. In this case, for an N set of human-labeled training data, i.e.$D_1 = \{P_i\}_{i=0}^{N}$ and for an M set of unlabeled data, i.e. $D_2 = \{Z_i\}_{i=0}^{M}$ at the time of training, the objective function is defined as follows:

$$\operatorname{loss}(D_1, D_2) = \min_{\theta} \frac{1}{N}\left(\sum_{i=1}^{N} \operatorname{loss}(X_i, Y_i)\right) + \frac{1}{M}\left(\sum_{i=1}^{N} \operatorname{loss}(X_i, R(Z_i, X))\right) \tag{2}$$

In this regard, the expression $R(Z_i,\ X)$ is a function that is defined according to the problem to determine the relationship between unlabeled educational data and human-labeled educational set Jing and Tian (2020).

2.3.4 Weakly-supervised learning

To learn the visual features of weakly supervised data, with a data set such as X, there will also be a weak, incomplete C_i label for each instance of X_i data from the X dataset. Now if the defined data set has N training samples, ie $D = \{X_i, i = 0, ..., N\}$, the cost function model is defined as follows when teaching:

$$\operatorname{loss}(D)=\min_{\theta} \frac{1}{N}\left(\sum_{i=1}^{N} \operatorname{loss}\left(X_{i}, C_{i}\right)\right) \tag{3}$$

Because the cost of labelling in weak supervision is much lower than in labelling in strong supervision, collecting and annotating large datasets is easy. Recently, in some works, images collected from the internet using hashtags as category label has been used Mahajan et al. (2018) Li, Wang, Li, Agustsson, and Van Gool (2017) to extract visual features from images, thereby good accuracy has been achieved.

2.3.5 Self-supervised learning

Compared to supervised learning methods that require input data X_i and human-generated Y_i labels; Self-supervised learning only uses X_i input data along with P_i pseudo-labels, which automatically generate P_i for predefined auxiliary problems. P_i pseudo-labels can be generated automatically using image or video attributes. For example, pseudo-labels can be generated based on image content or traditional image processing methods Mahendran, Thewlis, and Vedaldi (2018), Sayed, Brattoli, and Ommer (2018), Korbar, Tran, and Torresani (2018), Owens and Efros (2018), Kim, Cho, and Kweon (2019), Jing, Yang, Liu, and Tian (2018), Fernando, Bilen, Gavves, and Gould (2017) and Z. Ren and Lee (2018).

Having a set of N training data, ie $D = P_{i}{}_{i=0}^{N}$, the cost function for training data is defined in equation 4:

$$\operatorname{loss}(D)=\min_{\theta} \frac{1}{N}\left(\sum_{i=1}^{N} \operatorname{loss}\left(X_{i}, P_{i}\right)\right) \tag{4}$$

As long as pseudo-labels are generated automatically and without human intervention, the method used is self-monitoring. In this section, we examine self-supervised learning methods from various aspects such as network architecture, pretext task problems, and their applications.

2.4 Common architectures of deep neural networks

Regardless of the category of learning methods, whether supervised learning, or unsupervised, etc., most of these methods use the same or similar architectures. In this section, we briefly introduce common architectures for learning image properties. Various two-dimensional convolution neural networks have been

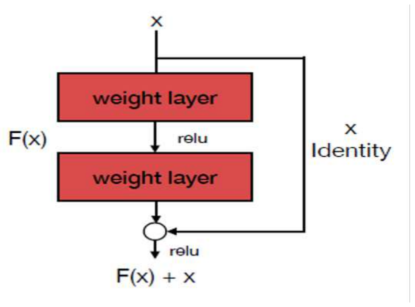


Figure 2.2: Residual network (ResNet) architecture.

developed to learn image properties, such as AlexNet, VGG, GoogLeNet Z. Ren and Lee (2018), ResNet And DenseNet Huang, Liu, Van Der Maaten, and Weinberger (2017). To further explore each of these architectures, we recommend the original article of each of these models to readers. However, because we used the ResNet50 model in our dissertation, we describe the architecture of this model.

Residual network architecture (ResNet)

We know from experience that deeper networks are likely to perform better, however, training deeper networks is very difficult for two reasons: gradient fade and gradient explosion. The ResNet architecture provides solutions to help us avoid these problems. This model uses escape connections in convolutional blocks by sending feature maps from the previous block to the next block to prevent the gradients from fading and exploding. Details of escape connections are shown in figure 2.2.

Using escape connections, it becomes very easy to train deep neural networks on GPU-equipped systems. ResNet networks with different widths have been developed to categorize images. Due to the small model size, low number of parameters, and excellent performance, ResNet models with different depths are often used as the base model for many computer vision tasks. Convolution blocks with escape joints are also widely used in many architectures.

2.5 A brief of pretext and downstream tasks

Most self-supervised learning approaches follow a schematic such as figure 2.3.

In general, a pretext task problem is defined for the convolutional network to solve it, by doing this task, the visual properties of the images are learned. For the defined pretext task, P-labels can be obtained

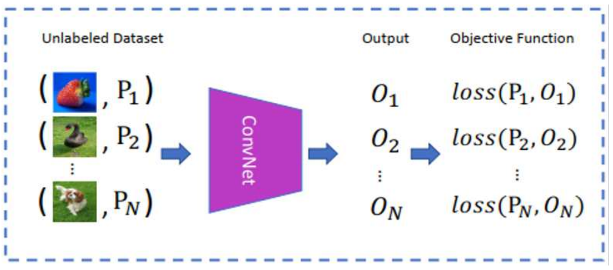


Figure 2.3: Feature representation schematic using self-supervised learning algorithms.

automatically or semi-automatically, without human intervention, based on the characteristics of the data itself. The convolutional network optimizes its parameters by minimizing the error between the network prediction and the P-labels. After the training on the pretext task is completed, the convolutional models have learned good visual properties that can be transferred to other computer vision problems Jing and Tian (2020).

2.5.1 Learning visual features by solving pretext tasks

To get rid of very large data set labeling, a pretext task is generally defined for convolutional neural networks to solve these problems using pseudo-tags that are generated automatically with minimal human intervention. So far, many problems have been designed and used, such as slicing objects on the screen Jing et al. (2018), painting Pathak et al. (2016), clustering Z. Ren and Lee (2018), coloring Fernando et al. (2017), jigsaw puzzle, etc. do. Effective pretext tasks ensure that meaningful features are well learned during the problem-solving process. For example, consider coloring an image where the goal is to turn gray images into color. To produce true color images, it is necessary that networks learn the content information and the structure of the images. In this case, the existing data, the X set, are gray images obtained by linear conversion on RGB color images, and our pseudo-labels, or Ps, are RGB color images themselves. X_i and P_i training data pairs can be generated in real time at very low cost. Self-supervised learning with other pretext tasks has a similar approach to coloring images.

2.5.2 Famous pretext tasks that have been used abundantly

Based on the characteristics of the dataset, which are used to design pretext tasks, we can divide these problems into 4 categories, which are: Generation Based Methods, Context-Based Methods, Free Semantic-Label Based Methods and Cross Modal-Based Methods.

Generation Based Methods: Visual features are learned during the process by which the image is generated. As an example of these methods, we can name image colorization Zhang et al. (2016), super resolution Ledig et al. (2017), image impainting Pathak et al. (2016) and produce images using Generative adversarial network Goodfellow et al. (2014) Chen, Zhai, and Houlsby (2018).

Context-based methods: In designing this type of task, the content features of images and videos, such as context similarity, spatial structure, local structure, etc., are considered. We briefly describe each of these cases.

- Context similarity: These are pretext tasks defined based on the similarity of the content between different parts of the image. for example; Image clustering methods are part of this category Caron, Bojanowski, Joulin, and Douze (2018).
- Spatial Context Structure: These types of pretext tasks used to train convolutional neural networks are based on spatial relationships between different parts of the image. For example, the jigsaw puzzle problem is predicting the lost content of the image and detecting geometric transformations applied to the images Jing et al. (2018).
- Temporal Context Structure: The temporal arrangement of video frames is used as a signal to train convolutional neural networks. The convolutional network is trained to determine whether video frames have entered the network in the correct order or not or predict the order of the frames Lee, Huang, Singh, and Yang (2017).

Free semantic-label based methods:

These types of auxiliary problems train neural networks using semantic labels that are generated automatically. Hardcode algorithms Faktor and Irani (2014), Stretcu and Leordeanu (2015) or game engines

Z. Ren and Lee (2018) generate the labels. For example; Fragmentation of moving objects and prediction of relative depth prediction Jiang, Larsson, Shakhnarovich, and Learned-Miller (2018), contour detection Z. Ren and Lee (2018), etc., are among these issues.

Cross Modal-Based Methods: This kind of pretext task train ConvNets to verify that there are two different channels input data matches each other for example; They examine the harmony of sound with image Korbar et al. (2018); Sayed et al. (2018).

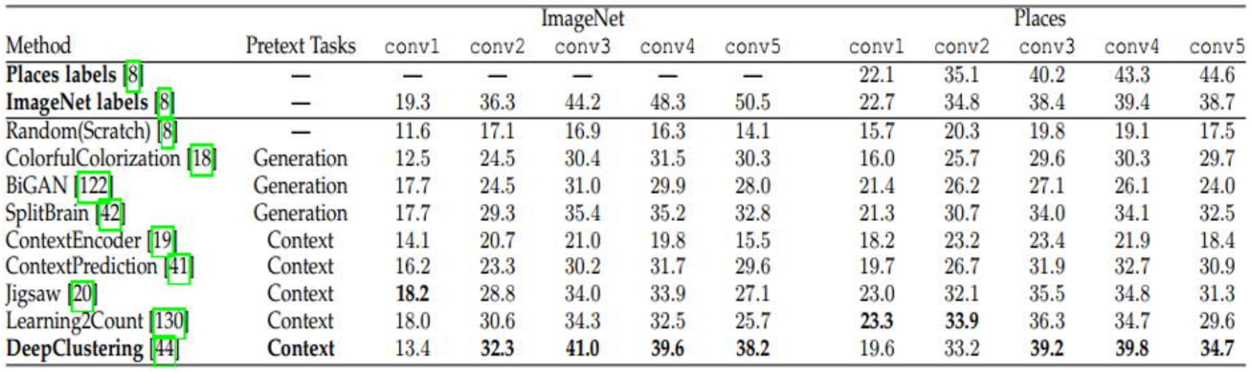
2.6 Comparing self-supervised learning methods based on solving pretext tasks

Supervised learning is not only dependent on large amounts of labeled data, but also grapples with other challenges such as generalization error, spurious correlation, and adversarial attack Liu et al. (2021). Recently, self-supervised learning methods have combined productive and reciprocal approaches to learn the intrinsic properties of data using unlabeled data. In this area, the common approach is to define various pretext tasks that help to learn features without the use of data tags. Things like painting pictures, coloring pictures gray, enhancing the quality of the image damaged by noise, predicting the frames of a video, matching the video with the sound, etc., have been useful for learning the features of images and videos.

This section compares the effectiveness of self-supervised learning methods, features of images and videos. For self-supervised learning, the learnt parameters by self-supervised learning are utilized as pre-trained models and after that, these parameters are fine-tuned on downstream tasks such as image classification, object identification, semantic segmentation, etc. are evaluated.

As mentioned earlier, the effectiveness of self-supervised learning methods is achieved by adjusting the weights learned on downstream tasks such as image categorization, semantic segmentation, and so on. The table below shows the performance of image classification on ImageNet Deng et al. (2009) and Places Kuehne, Jhuang, Garrote, Poggio, and Serre (2011) datasets.

During the self-supervised training, most of the methods were trained on the ImageNet dataset with the AlexNet neural network. After the self-supervised training step, a linear classification is conducted on different cannulation layers that are also frozen. The classification performance on the two datasets shows



		ImageNet					Places				
Method	Pretext Tasks	conv1	conv2	conv3	conv4	conv5	conv1	conv2	conv3	conv4	conv5
Places labels [8]	—	—	—	—	—	—	22.1	35.1	40.2	43.3	44.6
ImageNet labels [8]	—	19.3	36.3	44.2	48.3	50.5	22.7	34.8	38.4	39.4	38.7
Random(Scratch) [8]	—	11.6	17.1	16.9	16.3	14.1	15.7	20.3	19.8	19.1	17.5
ColorfulColorization [18]	Generation	12.5	24.5	30.4	31.5	30.3	16.0	25.7	29.6	30.3	29.7
BiGAN [122]	Generation	17.7	24.5	31.0	29.9	28.0	21.4	26.2	27.1	26.1	24.0
SplitBrain [42]	Generation	17.7	29.3	35.4	35.2	32.8	21.3	30.7	34.0	34.1	32.5
ContextEncoder [19]	Context	14.1	20.7	21.0	19.8	15.5	18.2	23.2	23.4	21.9	18.4
ContextPrediction [41]	Context	16.2	23.3	30.2	31.7	29.6	19.7	26.7	31.9	32.7	30.9
Jigsaw [20]	Context	**18.2**	28.8	34.0	33.9	27.1	23.0	32.1	35.5	34.8	31.3
Learning2Count [130]	Context	18.0	30.6	34.3	32.5	25.7	**23.3**	**33.9**	36.3	34.7	29.6
DeepClustering [44]	**Context**	13.4	**32.3**	**41.0**	**39.6**	**38.2**	19.6	33.2	**39.2**	**39.8**	**34.7**

Table 2.1: Comparison of the accuracy of self-supervised learning methods when transferring to Places and ImageNet datasets from different layers of the AlexNet model Jing and Tian (2020)
.

the quality of the features learned by solving various pretext tasks.

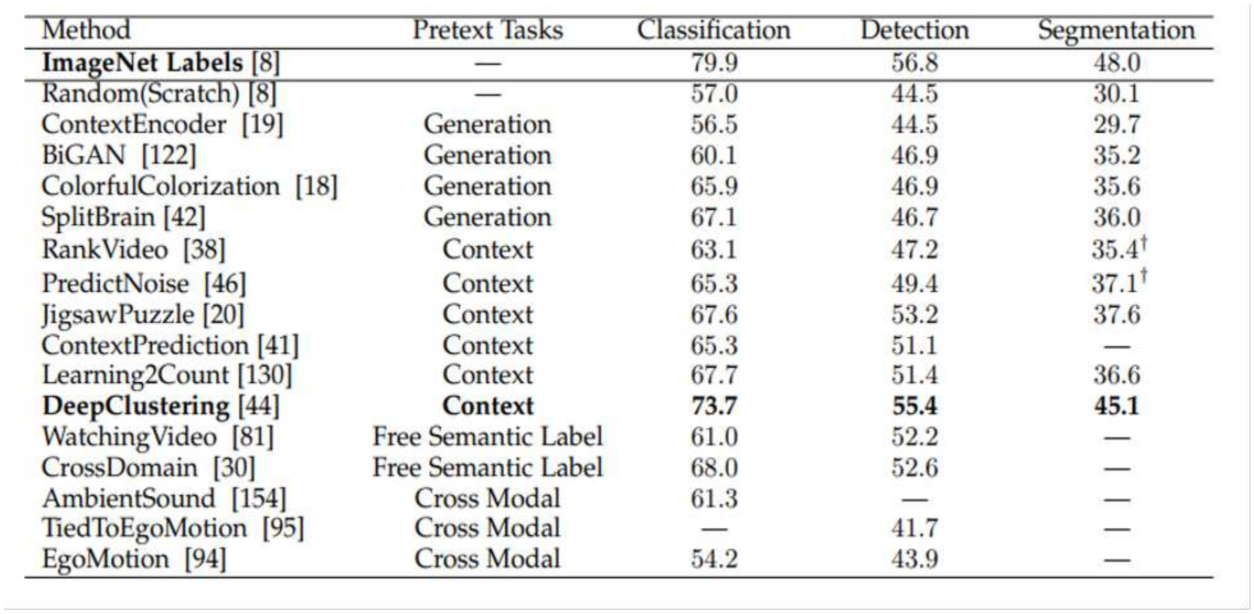
As shown in table 2.1, the overall performance of self-supervised learning methods is lower than that of the models trained to monitor both the ImageNet and Places datasets. Of all the self-supervised learning methods, the DeepCluster algorithm has the highest efficiency. Based on the performance shown in the table above, we conclude:

(1) Features at different layers get better by solving pretext tasks. The efficiency of self-supervised learning methods is always better than the performance of algorithms trained from scratch.

(2) All self-supervised learning methods improve performance by using the features of conv3 and conv4 layers, while they perform worse on downstream tasks by using the features of conv1, conv2 and conv5 layers. This decrease in accuracy may be due to the fact that surface layers extract more general low-level features from images, and deeper layers record high-level features associated with pretext tasks, which in both cases solving high-level downstream problems alone is not enough.

(3) When the datasets used for training the pretext task and the downstream task are different, Self-supervised features learning methods can achieve comparable accuracy to supervised methods in the ImageNet dataset.

In addition to image classification, object recognition and semantic segmentation are also used as downstream issues to evaluate the performance of features learned through self-supervised learning methods. In



Method	Pretext Tasks	Classification	Detection	Segmentation
ImageNet Labels [8]	—	79.9	56.8	48.0
Random(Scratch) [8]	—	57.0	44.5	30.1
ContextEncoder [19]	Generation	56.5	44.5	29.7
BiGAN [122]	Generation	60.1	46.9	35.2
ColorfulColorization [18]	Generation	65.9	46.9	35.6
SplitBrain [42]	Generation	67.1	46.7	36.0
RankVideo [38]	Context	63.1	47.2	35.4†
PredictNoise [46]	Context	65.3	49.4	37.1†
JigsawPuzzle [20]	Context	67.6	53.2	37.6
ContextPrediction [41]	Context	65.3	51.1	—
Learning2Count [130]	Context	67.7	51.4	36.6
DeepClustering [44]	**Context**	**73.7**	**55.4**	**45.1**
WatchingVideo [81]	Free Semantic Label	61.0	52.2	—
CrossDomain [30]	Free Semantic Label	68.0	52.6	—
AmbientSound [154]	Cross Modal	61.3	—	—
TiedToEgoMotion [95]	Cross Modal	—	41.7	—
EgoMotion [94]	Cross Modal	54.2	43.9	—

Table 2.2: Self-supervised learning methods accuracy on different computer vision tasks Jing and Tian (2020)
.

this case, the ImageNet dataset is commonly used in pretraining. Also, the very famous AlexNet model has been selected as the base network. Table 2.2 lists the performance of image classification, object recognition and semantic segmentation on the PASCAL VOC2007 dataset. As shown in table 2.2, the performance of some self-supervised learning methods used for solving pretext tasks on semantic segmentation and object identification datasets is similar to that of fully supervised learning on the ImageNet dataset.

The difference in accuracy obtained for semantic segmentation and object identification in self-supervised and supervised learning is less than %3 higher for supervised pre-training, which indicates that the features learned by self-supervised learning algorithms have the ability to generalize to other tasks like instance segmentation and SOD. Among the self-supervised learning methods, the DeepCluster Caron et al. (2018) algorithm has shown the best accuracy on all problems.

Despite all the creativity that researchers have expended in defining various pretext-tasks that are defined according to their data characteristics, pre-trained models using these methods in solving other computer vision problems could not perform better than the previously trained ImageNet models in a supervised manner. For this reason, it was necessary to develop other methods to minimize the gap between supervised and unsupervised learning methods. One of these approaches is contrastive self-supervised learning, which we will explain in detail in the following sections.

2.7 Contrastive self-supervised learning methods

As we have mentioned, in recent years, self-supervised learning methods have become very popular among other machine learning methods because they eliminate our need for large-scale labeled data. These methods can use pseudo-labels extracted from the data itself as monitoring signals and apply the obtained visual representations or features to other computer vision problems, also known as downstream problems. In particular, contrastive self-supervised learning methods have become very important among other self-supervised learning methods, and in computer vision applications and natural language processing, etc., the highest accuracy has been achieved by using contrastive learning methods. The general goal of contrastive learning methods is to have similar properties in the feature space of different versions of an image obtained by applying different data processing methods, while trying to keep other images in the data set or category as far away as possible in the feature space. In this chapter, we want to review self-supervised learning methods that are based on creating competition between data in a category or a whole set of data. Also, We look at common pretext tasks used for feature interaction learning, and take a closer look at common architectures developed for this purpose in recent years. Then the performance of different methods using several downstream problems such as image categorization, object identification, and so on are compared.

Recent advances in deep learning have made these methods an integral part of many intelligent systems. The ability to learn rich and valuable patterns from the vast amount of data available today has made deep neural networks an extremely important component of computer vision tasks such as image sorting, object recognition, semantic segmentation, and more. Also, deep neural networks are also widely used for natural language processing applications such as language models, sentence categorization, and more. However, supervised approaches to learning features from labelled data have become saturated, as they require us to manually label millions of data. For example, labelling an ImageNet dataset at the image level with about fourteen million images takes 22 years. And we must keep in mind that image-level labeling will not be accurate for other computer vision tasks, such as semantic segmentation. We also need experts in the same field to label some types of data, so that data labeling can be done accurately. For example, for medical or satellite image data, it is necessary for experts to do the labeling, which in some cases costs a lot of time and money to projects. The need for large data labelling is due to the fact that many modern computer systems that are also monitored try to extract a form of image features by finding a pattern between tags and data points. In this regard, works such as GRAD-CAM Selvaraju et al. (2017) have been done to demonstrate a technique for visually describing the decisions made by the model.

Traditional supervised learning approaches are highly dependent on the number of labelled data. Furthermore, Although access to large amounts of data in today's world is in many cases simple, the lack of data tags has led researchers to find alternative ways to learn features without the need for human-labelled data. Here, self-supervised learning methods play a key role in shaping the development of deep learning methods without the use of labeled data. In these methods, according to the characteristics of the data itself, a supervising signal is provided.

Unlike generative models, contrastive-learning is an approach that aims to group similar samples side by side and dissimilar data far apart. The main idea of these methods is shown in figure 2.4. To achieve this, it is necessary to use a similarity metric to measure the proximity of images in the feature space. Specifically for computer vision tasks, a cost function or cross-objective function is defined based on the properties extracted from the images. For example, one data is selected from a training sample and another version is obtained by applying different data augmentation methods. During the tutorial, as shown in figure 2.4, the augmented version of the image is considered a positive pair, and the other examples in the dataset or category (depending on what method is used) is considered as a negative pair.

Then the model is trained in a way to separate positive samples from negative ones. To separate these samples, we use the pretext tasks that are described in the following. By doing this, the model can learn good indicators that can be useful for performing other computer vision tasks. The idea for this comes from an interesting experiment conducted in 2016 by Epstein Epstein (18.02.2016/n.d.). The story goes that Mr. Epstein asks his students to draw a picture of a dollar bill on paper without looking at it. The results of this experiment show that the human brain does not need to see the detailed information of each object to be able to distinguish between two different objects, but instead uses more general features to identify and separate objects.

Much work has been done in this area which combines sampling approach at the sample level with

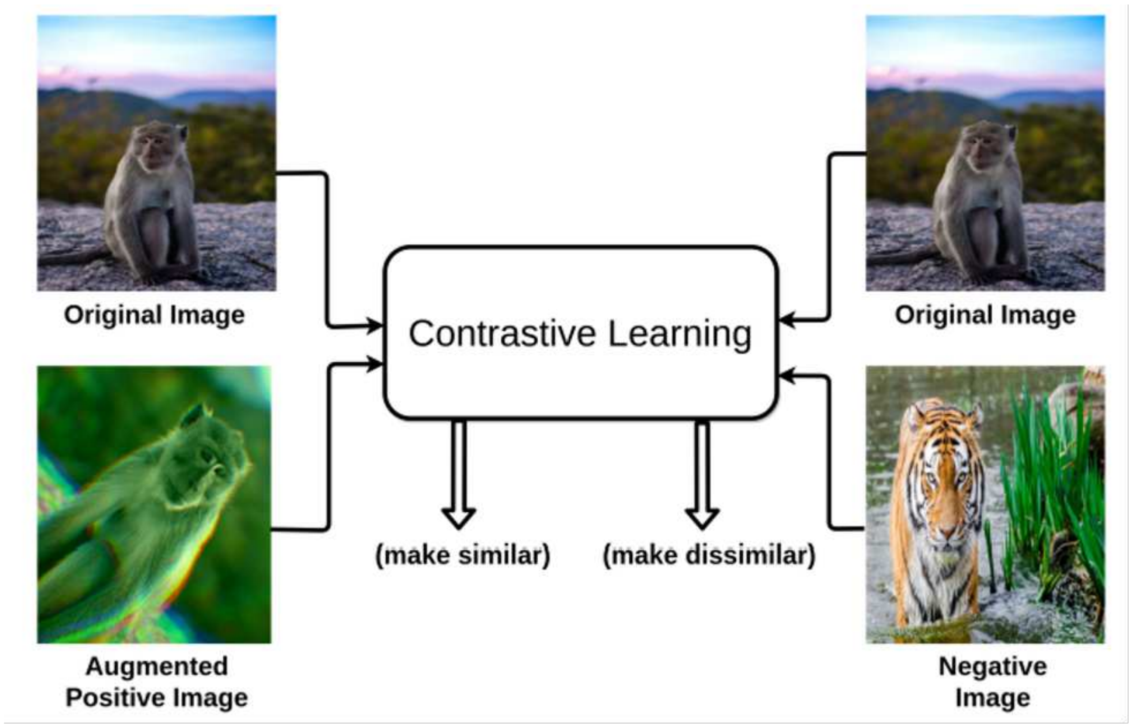


Figure 2.4: The main idea in cross-learning: the original image and the augmented data version are brought closer together and the original image is separated from the negative examples Jaiswal et al. (2020)
.

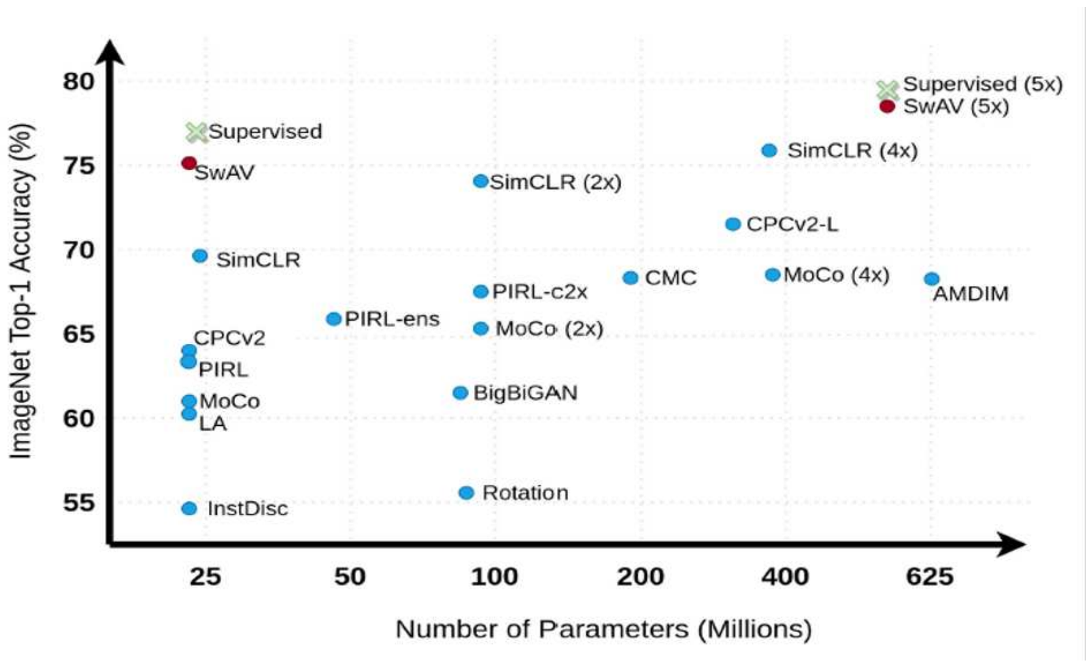


Figure 2.5: Accuracy of classifying different contrastive learning methods on the ImageNet dataset Jaiswal et al. (2020)
.

contrastive learning as a self-supervised approach, and thus achieve very good accuracy. However, recent methods such as SwAV Caron et al. (2020), MoCo He, Fan, Wu, Xie, and Girshick (2020) and SimCLR Chen et al. (2020) algorithms use advanced approaches and show results that are very comparable to the best supervised learning methods on ImageNet Deng et al. (2009) dataset. Figure 2.5 shows a comparison of the efficiency of these methods.

2.7.1 Common architectures for self-supervised contrastive learning

Contrastive learning methods are highly dependent on the number of samples or negative pairs in the data set or category to be able to learn good image representations or features. These tasks can be considered as a matter of searching for a word within a dictionary. In some cases, the size of this dictionary is the size of the total images in the training dataset, and in some cases, its size is a subset of a training dataset. A very interesting way to categorize the methods in contrastive learning can be based on the technique used in these methods to collect and construct negative samples compared to positive samples or pairs when teaching

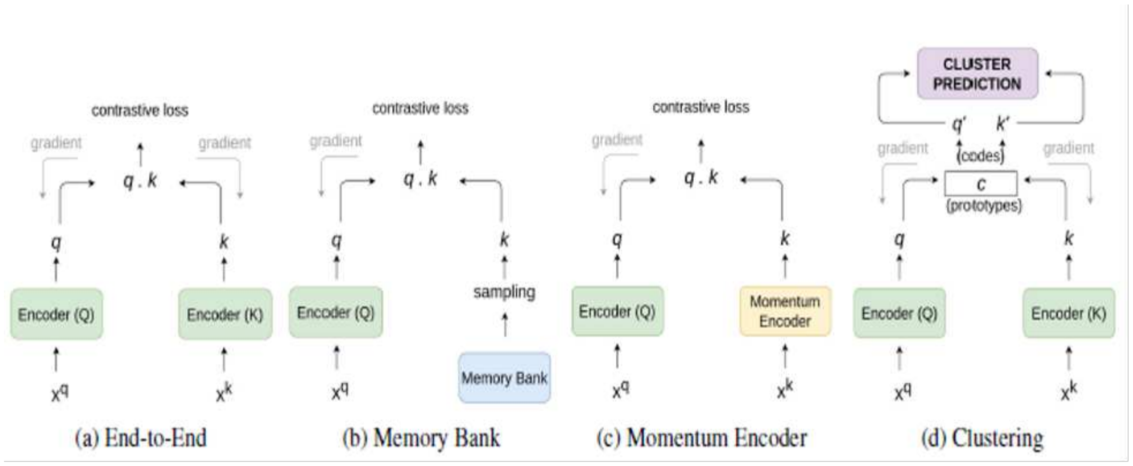


Figure 2.6: Common architectures in contrastive self-supervised learning

.

the model. Based on which approach is chosen, we divide the methods in contrastive learning into four categories which is shown in figure 2.6.

Methods based on the use of end-to-end architectures

End-to-end learning is a very complex learning system that uses gradient-based learning, so that all the modules used are derivative. This architecture prefers very large batch sizes to be able to collect a large number of negative samples, forcing these negative samples to compete with the positive ones, and to learn. Except for the original images and their augmented version, the rest of the images in the category are considered negative examples. The procedure is to use two encoder networks, which are query encoder Q and K key encoder, as shown in figure 2.6. The two selected encoder networks can be different or identical and can be updated end-to-end during the training using the backpropagation algorithm. Using the contrastive loss function, the network converges to a direction where the positive samples are close together in the feature space and the negative samples are as far apart as possible. In these methods, the query encoder is trained using the original image, and the key encoder is trained on the positive pair of images and other negative samples. The feature vectors q and k are generated using these encoders and are used to calculate the similarity to the corresponding input using a similarity metric described below. In many cases, the metric that measures the similarity of these vectors is a function of cosine similarity, which is the interior product of the normalized state of the vectors obtained from the corresponding encoders.

Recently, SimCLR Chen et al. (2020), a successful end-to-end model has been introduced with three different versions which in the first version, the batch size is 4096 and with 1000 epochs, the model is trained. The figure 2.7 shows that despite the simplicity of the end-to-end architectures, the methods that follow this architecture can work better with large batch sizes and a large number of training epochs. Another well-known work that follows end-to-end architecture is presented in the article by Oord, Li, and Vinyals (2018). In this article, learning the features of high-dimensional time series data is done by predicting the future in the property space by autoregressive models with the help of contrastive function. This approach makes the model traceable by using negative pair data sampling. In algorithms that follow this architecture, the number of negative samples used to properly train the network depends on the size of the batch, as the negative batch is selected from within the batch. Because batch size is limited to GPU memory size, the ability to scale large batch sizes for methods that use this architecture is a fundamental challenge. Furthermore, for large batch sizes, these methods suffer from Mini-Batch Optimization Problem and require effective optimization methods, which are described in the article Goyal et al. (2017).

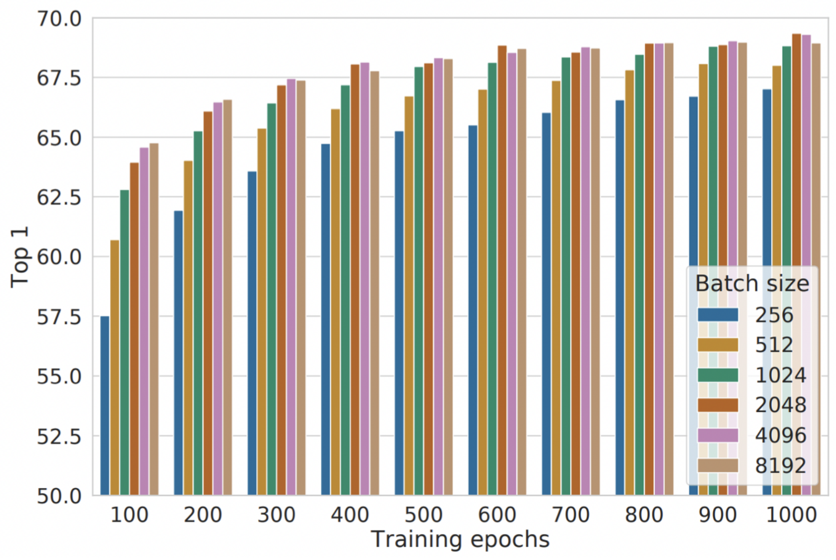


Figure 2.7: Linear evaluation models (ResNet-50) trained with various batch sizes and epochs Chen et al. (2020)

.

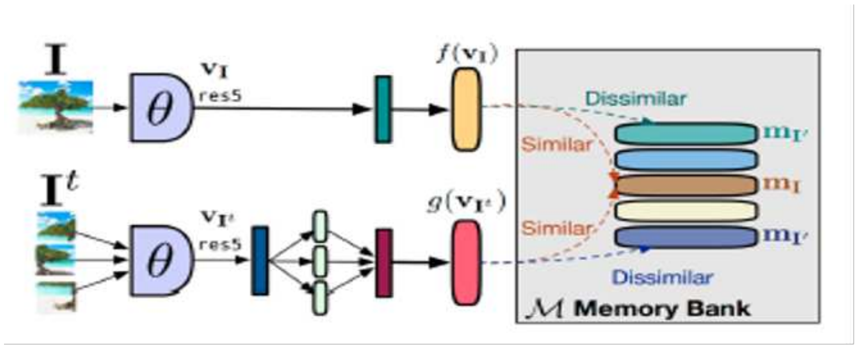


Figure 2.8: Use of memory bank in PIRL algorithm: Memory bank includes the average displacement of negative sample representations that are used in mutual learning Misra and Maaten (2020)
.

Methods based on using a memory bank

Because of the problems with large batch sizes and their negative effect on optimization during training, one possible solution is to keep a separate dictionary called a memory bank. In the following, we will explain the concept of memory bank more.

Memory bank: The purpose of maintaining a memory bank is to accumulate a large number of features of different samples that are used as negative samples during training. To achieve this, a dictionary is created that stores and updates the features of the samples that were last entered in the model. The memory bank (M) contains a feature representation (m_I) for each instance I in the D dataset. The m_I indicator is an exponential moving average of the indicators or attributes calculated during previous epochs. Memory bank, replacing negative instances of m_I with the corresponding representation in the memory bank, provides instruction without the need to resize. That is, negative samples that are necessary for mutual learning are extracted from the memory bank. The representation of each instance in the memory bank is updated the last time it is given to the model. That is, as soon as the data is last given to the model, its corresponding properties in the memory bank are updated and replaced with the last representation obtained from the sample. Therefore, the sampled keys related to encoder are in several different steps during the previous epochs. The PIRL algorithm Misra and Maaten (2020) is one of the recent successful methods that extracts good features from images and uses a memory bank to provide negative samples during training. In figure 2.8, we see how the PIRL algorithm works.

This approach requires algorithms to construct image representations that are covarient to any pretext problem used. Basically, the PIRL algorithm chose the jigsaw puzzle as the pretext problem. Another well-known work that uses the idea of a memory bank is designed in the article Misra and Maaten (2020), in which a non-parametric version of the Soft Max classifier is implemented, which has a higher scalability for large data applications. However, maintaining a memory bank during training can be very complicated. One of the disadvantages of this approach is that updating the memory bank requires a lot of computing resources because the features in the memory bank become obsolete quickly and require frequent updates.

Methods based on using a momentum encoder

To solve the problems that arose for the memory bank, we can replace the memory bank with a separate module called momentum encoder. The momentum encoder produces a series of encoded keys in a dictionary. In this dictionary, the keys encoded in the current category are added to the queue, and the keys for the oldest category are removed from the queue. The dictionary keys are defined on the fly by a set of data samples in the category during the training. The momentum encoder has some of the same parameters as the request encoder or Q shown in figure 2.6. After performing any forward propagation in the momentum encoder, no error propagation is performed and instead, the parameters of this encoder are updated based on the request encoder parameters using equation 5 He et al. (2020).

$$\theta_{\mathrm{k}} = \mathrm{m} * \theta_{\mathrm{k}} + (1 - \mathrm{m}) * \theta_{\mathrm{q}}, \mathrm{m}[0, 1) \tag{5}$$

In this equation, m, is a number between zero and one, called the momentum coefficient. Only the request encoder parameters or $theta_q$ are updated during the training using the error propagation algorithm. Updating the momentum encoder causes the weights of this network to change at a slower speed than the Q network. As a result, although the keys in the queue are encoded by different encoders (in different categories), the difference between the representations obtained from this encoder in different steps is very small.

The advantage of using this architecture over the previous two methods is that there is no need to teach two separate models. Furthermore, there is no need to maintain a memory bank, which itself has a lot of computation at the time of the update.

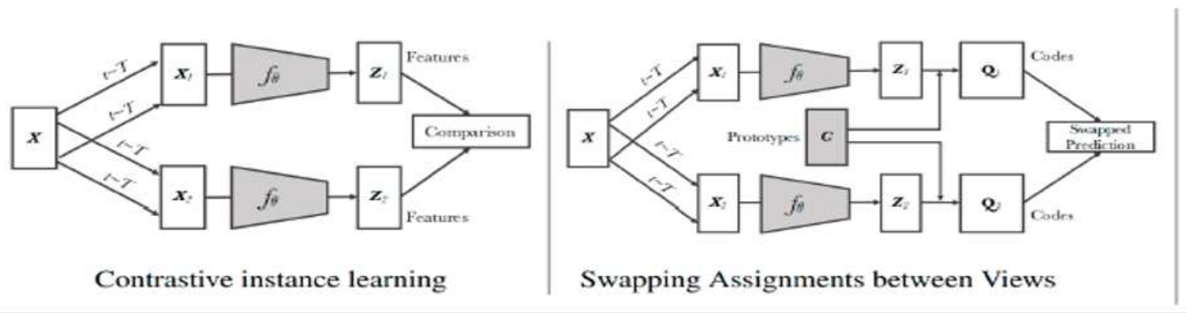


Figure 2.9: Comparing contrastive learning algorithm with SWAV algorithm Misra and Maaten (2020)
.

Clustering-Based Methods

All three architectures described above focus on comparing positive and negative examples using a similarity criterion, and attempt to place similar items in the feature space close together and dissimilar items in the feature space, keep away from each other. By doing this, The model can learn great features. In contrast, clustering-based learning architecture follows an end-to-end approach using two encoders that share their parameters; But instead of using a contrastive approach between samples, it uses clustering algorithms to cluster similar features. One recent work that uses the cluster-based feature learning approach is the SwAV Misra and Maaten (2020) algorithm, which is shown in figure 2.9.

Figure 2.9 shows the difference between contrastive learning architecture and clustering-based method in the best way. The goal here is not only to bring similar pairs of specimens closer together, but also to ensure that other similar features are included in a cluster. For example, in the image property space, the properties obtained for the cat image should be similar to or close to the dog class characteristics, as both are in the animal cluster. Also, the characteristics of these two classes should be as similar as possible to the characteristics obtained for the home class, because it has a separate cluster. In sample-based learning, each instance in the data set is considered as a separate class. This factor causes the input sample to be considered a negative pair relative to the original image when the input sample is compared to other samples in the same class, and because the network tries to separate the negative pairs as much as possible, network weights are updated incorrectly. To illustrate the problem in more detail, let us give an example. Suppose we have a picture of a cat class among a bunch of images that are going to enter the network altogether. When giving this batch of images to the model, other images in the category are considered as negative examples. Now the problem arises when, in addition to the cat image, there are other cat images other than the one we mentioned in the batch, in this case all other images of the cat class are considered as negative examples and the network is not directed to the right point. This mode forces the model not to consider two images of the cat class similar during training. This problem is explicitly solved using clustering-based algorithms.

2.7.2 Common encoders in contrastive self-supervised learning architecture

It is correct to say that encoders play the most important role in any self-supervised learning system, as they are responsible for mapping input samples to the feature space. Figure 2.10 shows the role of an encoder in self-monitoring learning methods.

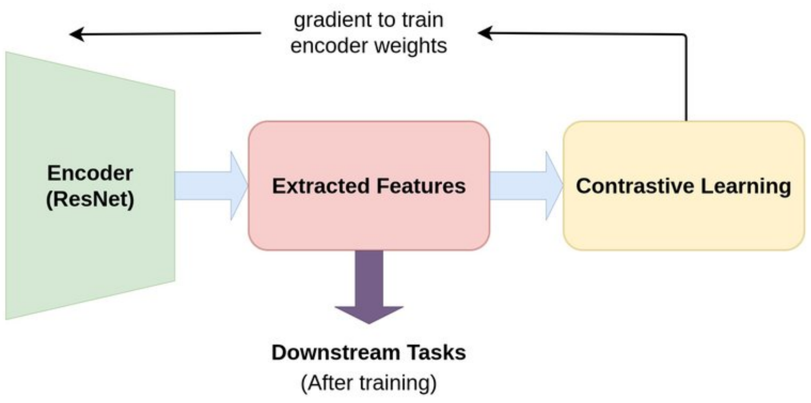


Figure 2.10: Transfer learning in self-supervised learning methods Jaiswal et al. (2020)

.

Without effective features, the category model will face a fundamental problem in learning the differences and distinctions between different categories. Much of the work done in the field of self-supervised-contrastive-learning has used the ResNet model as an encoder. Among the various models of ResNet architecture, the ResNet50 network has been used extensively due to its balance between learning power and network size. In an encoder, the output of a specific layer is merged to provide a one-dimensional feature vector for each instance. Depending on the approach chosen, this feature vector can be upsampled or downsampled. For example, in the work presented by Misra and Maaten (2020), the ResNet50 architecture is used in such a way that the output features of the fifth block (res5) are merged on average so that for each sample or input image, a vector containing 2048 numbers is produced. They then apply a linear mapping or linear layer to this vector and convert the feature vector to a 128-dimensional vector. They also examined the features of other blocks, such as res2, res3, and res4, to evaluate the performance of the features learned at different layers. As might be expected, the properties extracted from the last blocks of the encoder have better properties than the output of the primary layer blocks, and respond better to other computer vision problems. Similarly, in the paper Chen, Zhai, Ritter, Lucic, and Houlsby (2019) a traditional ResNet model is used as the encoder in which the features are extracted from the output of Global Average-Pooling. In addition, a multi-layer perception block with only one linear layer is applied to the obtained features to map these feautres to a space with other dimensions, now in the current space, the cost function or contrastive loss function is calculated Lorre, Rabarisoa, Orcesi, Ainouz, and Canu (2020) Tao, Wang, and Yamasaki (2020).

2.7.3 Contrastive self-supervised models training

To train encoder, a pretext task is executed by using the contrastive cost function to propagate the error in the network. As we have already mentioned, the main idea in contrastive learning is to bring similar examples closer together in the feature space and to separate the negative examples as much as possible. To achieve this, a similarity criterion is employed to calculate and determine the proximity of two samples in the feature space. In cross-learning, the criterion for similarity in many cases is cosine similarity, which is used as a basic method in many cross-learning methods. The cosine similarity of two variables or vectors is the cosine of the angle between these two vectors and is mathematically defined as:

$$\cos(\theta) = \frac{\mathrm{A} \cdot \mathrm{B}}{\|A\| \|B\|} \tag{6}$$

The focus of cross-learning is on comparing the features obtained for the samples using the noise contrastive estimation function Gutmann and Hyvärinen (2010). This function is defined as:

$$L_{NCE} = -\log \frac{\exp\left(\operatorname{sim}\left(q, k_{+}\right)/\tau\right)}{\exp\left(\operatorname{sim}\left(q, k_{+}\right)/\tau\right) + \exp\left(\operatorname{sim}\left(q, k_{-}\right)/\tau\right)} \tag{7}$$

In equation (7), q represents the original sample, k_+ represents the positive sample, and k_- represents the negative sample. And the superparameter has been used in many recent methods and is called the Temperature Hyper-Parameter. The basic idea behind the NCE cost function was to calculate a nonlinear logistic estimate that would separate the observed data from the artificially generated noise. If the number of negative instances increases, another type of NCE cost function called InfoNCE can be used, namely,

$$L_{NCE} = -\log \frac{\exp\left(\frac{\text{sim}(q,k_+)}{\tau}\right)}{\exp\left(\frac{\text{sim}(q,k_+)}{\tau}\right) + \sum_{i=0}^{K} \exp\left(\frac{\text{sim}(q,k_i)}{\tau}\right)} \tag{8}$$

In equation 8, k_i represents negative examples. Like other deep learning methods, contrastive learning can use many optimization algorithms for training. The learning process is such that the parameters of the encoder network are learned by minimizing the cost function. The SGD optimization algorithm is one of the most popular algorithms used to optimize the objective function in contrastive-learning methods Bojanowski and Joulin (2017) Z. Wu, Xiong, Yu, and Lin (2018) Z. Wu et al. (2018) Misra and Maaten (2020). The estimate for the cost function gradient relative to the parameters made using this algorithm is approximate. Because it replaces the actual gradient of the cost function with respect to the parameters that should be calculated based on all training samples with the calculation of the gradient based on the cost function calculated on a small subset of training data. The crucial parameter that must be determined for the Stochastic Gradient Descent algorithm is the learning rate. In practice, the rate of learning should decrease over time. In many deep learning approaches, an upgraded mode of the SGD algorithm such as momentum is used.

Another optimization method used in some of these methods is the adam algorithm. Because some end-to-end methods use large batch sizes during network training, using SGD-based optimization algorithms that scale the learning rate linearly is not possible. To stabilize the training, the Layer wise Adaptive Rate Scaling (LARS) You, Gitman, and Ginsburg (2017) algorithm with cosine learning rate is used.

Chapter 3

Instance segmentation using modified bootstrap your own latent (BYOL)

3.1 Proposed self-supervised learning framework for instance segmentation

Generally, computer vision pipelines that use self-supervised learning concern executing two tasks: a pretext task and a downstream task. Downstream tasks are application-specific tasks that employ the knowledge learned during the pretext task. Figure 3.1 illustrates how knowledge is transferred to a downstream task. In this figure, the higher line represents self supervised pretraining using BYOL algorithm and the lower line is trained using Mask R-CNN for instance segmentation and background removal eventually.Transfer learning execution on these high-level vision tasks proves the generalization ability of the learned features. Some of the common downstream tasks in computer vision are classification, detection, segmentation, future prediction, etc. In our work, instance segmentation using self-supervised learning is performed as downstream task followed by background removal.

As mentioned in the previous chapter, a number of self-supervised learning methods known as instance segmentation using contrastive learning, perform better in representation learning of ImageNet dataset images than other recent methods which can minimise the accuracy gap between supervised and self-supervised learning. Since these self-supervised methods, do not need labelled data in pretraining, we could benefit from the numerous unlabelled data in real world, as collecting unlabelled dataset is not difficult in plenty of applications. Our downstream task results obtained using COCO 2014 with almost 82,000 unlabelled images for pretraining using BYOL are comparable with the results using ImageNet initial weights trained on 1.3 million labelled data in a supervised way.

As mentioned earlier, we are utilising COCO 2014 dataset and BYOL algorithm which is a self-supervised method, for learning a rich representation of unlabeled images for dense prediction tasks like instance segmentation and then background removing which was done by applying a post processing to instance segmentation. It is worth mentioning that before solving background removing problem, at first three different weights, namely, random initial weights, ImageNet supervised initial weights and pretrained weights obtained from BYOL algorithm are used to initialise Mask R-CNN on COCO dataset 2017 and then instance segmentation problem is solved using these three initial weights. Afterwards, for each model a certain type of post-processing is applied to solve our downstream task or background removal. Obtained results prove the effectiveness of self-supervised pretraining using BYOL algorithm in instance segmentation and background removing problems.

Instance segmentation and background removal combined with self-supervised learning to obtain initial weights is so innovative. It outperforms Mask R-CNN using random initial weights and can compete with Mask R-CNN using ImageNet initial weights obtained from training on almost 1.3 million labelled images. In addition, we made minor changes in the pretraining self-supervised model (BYOL) middle layers to reduce the computations and speed up our training and convergence. Furthermore, in Mask R-CNN paper He et al. (2017), the final goal is instance segmentation. Therefore, to add background removal of still images, we added a post-processing to Mask R-CNN to remove the background or detect foreground in input images. In general, self-supervised learning is a technique that pretrains the network to learn visual features and then the pretrained network can be fine-tuned using the annotated data. In the following subsections, self-supervised pretraining using BYOL and then instance segmentation and background removal using Mask R-CNN are explained in details. To train the models, NVIDIA Tesla K80 is used.

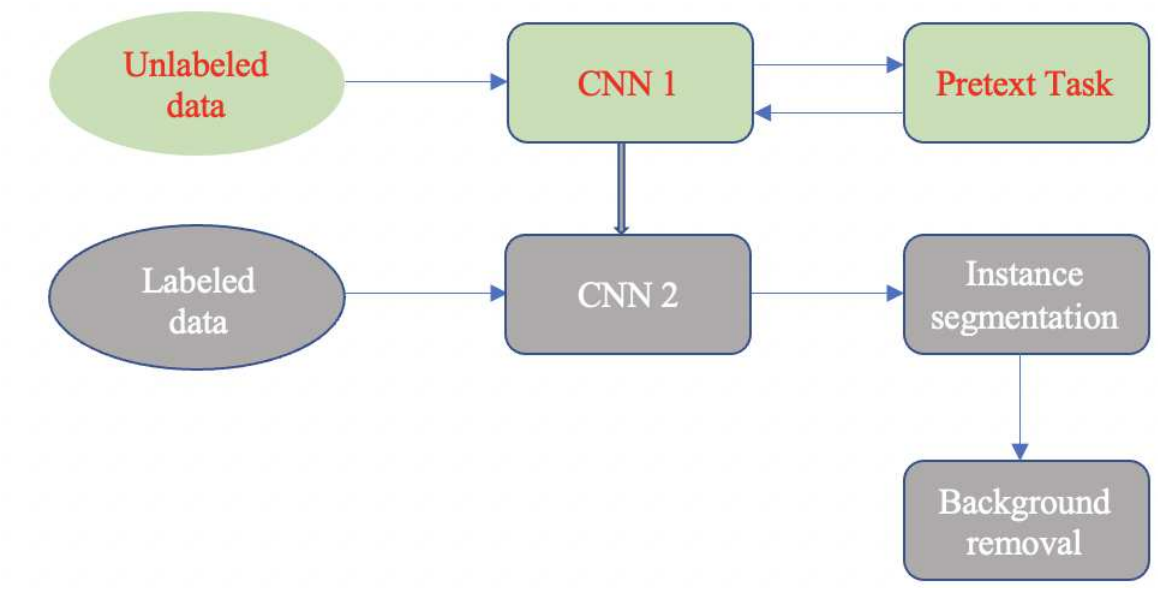


Figure 3.1: The framework of self-supervised instance segmentation and background removal

.

3.2 BYOL algorithm for pretraining

3.2.1 A brief about BYOL algorithm

BYOL is an algorithm based on contrastive learning that does not require negative samples which is the foundation of some other methods like MoCo and SimCLR. This factor causes BYOL to be computationally more efficient compared to other methods like SimCLR. Furthermore, this algorithm outperforms other methods in terms of efficiency. In the following chart, we show the performance of this method compared to other existing algorithms:

In Figure 3.2, X axis represents the number of the parameters of each model and Y axis shows accuracy during on the linear evaluation on ImageNet dataset using ResNet-50 as backbone. A linear evaluation is a standard evaluation protocol to train a linear classifier on top of (frozen) representations learnt by self-supervised approaches. As shown, pretrained weights using BYOL algorithm, despite having fewer parameters than other algorithms like MoCo and AMDIM, could reach a considerable accuracy so that the difference between this algorithm and supervised one is only 2%. Accordingly, this method is utilised for pretraining visual features in this thesis. The general schematic of the BYOL algorithm is shown in figure

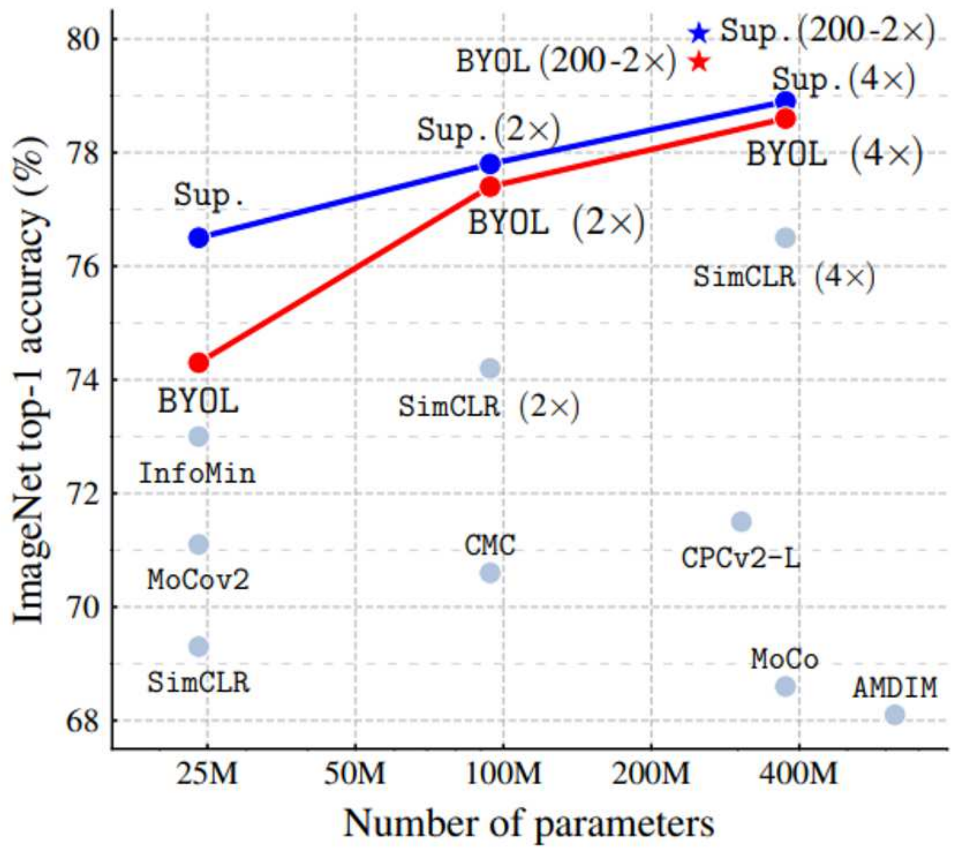


Figure 3.2: Demonstration of accuracy of BYOL compared with supervised and other unsupervised models Grill et al. (2020)
.

3.3. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $s_g(z'_\xi)$, where θ is the trained weights, ξ is an exponential moving average of θ and s_g means stop-gradient. Stop gradient is an operation that prevents the models from getting trivial solutions. At the end of training, only f_θ is retained for next steps, and y_θ is used as the image representation only used in pretraining Grill et al. (2020).

Before diving into the mathematical aspect of figure 3.3, let's consider an example. Assume that x is the input image of a cat. T and T' are two different augmented views of this cat. For instance, random cropping is applied to the first view and gaussian blurring to the second view. The goal is to predict the second view's features using the first view's features. Indeed, they both have to have the same representations as both views are generated from one image. Hence, if the first view shows the representations of a cat, the second view has to follow and shows similar representations. Once the prediction of the second view is different than first view, the online and target networks' parameters are updated.

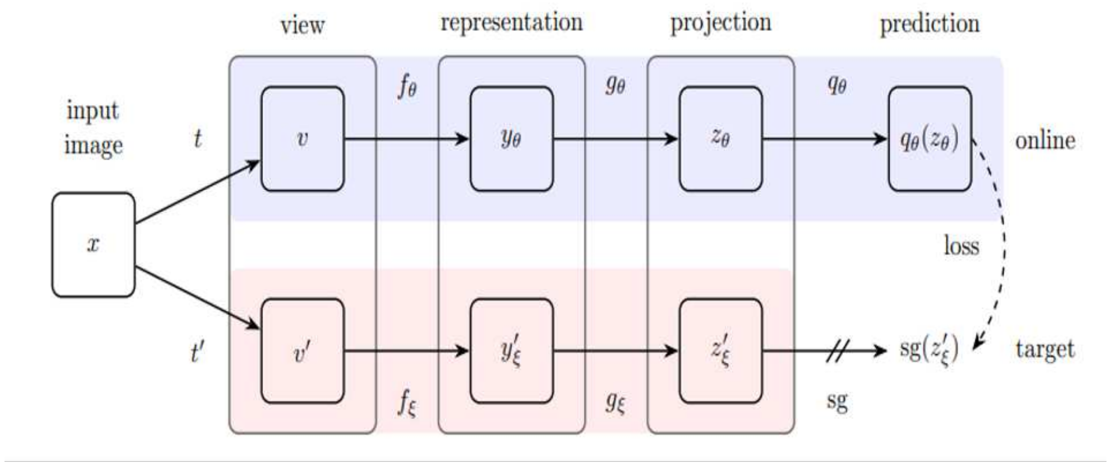


Figure 3.3: BYOL's architecture Grill et al. (2020)

.

BYOL's objective is to learn a representation y_θ which can be utilsed later for downstream tasks. BYOL uses two online and target neural networks to learn. The online network is a set of weights and consists of three stages: an encoder f_θ, a projector g_θ and a predictor q_θ as shown in figure 3.3. The target network has the identical architecture as the online network, with a different set of weights. The target network provides the regression targets to train the online network, and its parameters are an exponential moving average of the online parameters . Indeed, given a target decay rate $\tau \in [0, 1]$, after each training step we perform the following update:

$$\xi \leftarrow \tau\xi(1-\tau)\theta \tag{9}$$

Given a set of images D, an image x is sampled uniformly from D, BYOL produces two augmented views v=t(x) and v' = t'(x) from x by applying respectively image augmentations t $\sim$ T and t' $\sim$ T'. From the first augmented view v, the online network outputs a representation $y_\theta = f_\theta(v)$ and a projection $z_\theta = g_\theta(y)$. The target network outputs $y'_\xi = f_{xi}(v')$ and the target projection $z'{}_\xi = g_\xi(y')$ from the second augmented view v'. Finally, we obtain a prediction $q_\theta(z_\theta)$ of $z\prime_\xi$ and l_2 - normalize both $q_\theta(z_\theta)$ and z'_ξ to obtain $\overline{\mathrm{q}}_\theta\left(\mathrm{z}_\theta\right) = \frac{\mathrm{q}_\theta(\mathrm{z}_\theta)}{\|\mathrm{q}_\theta(\mathrm{z}_\theta)\|_2}$ and $\bar{z}'_\xi \triangleq= z'_\xi / \left\| z'_\xi \right\|_2$ Grill et al. (2020). The loss function used is mean squared error which is the difference between l2 normalized online and target networks' representations.

$$\mathcal{L}_{\theta}^{\mathrm{BYOL}} \triangleq \left\|\overline{q_{\theta}}\left(z_{\theta}\right)-\bar{z}_{\xi}^{\prime}\right\|_{2}^{2}=2-2 \cdot \frac{\left\langle q_{\theta}\left(z_{\theta}\right), z_{\xi}^{\prime}\right\rangle}{\left\|q_{\theta}\left(z_{\theta}\right)\right\|_{2} \cdot\left\|z_{\xi}^{\prime}\right\|_{2}} \tag{10}$$

3.2.2 Image augmentations

BYOL's image augmentations are similar to SimCLR Chen et al. (2020). SimCLR is a framework for contrastive learning of visual representations. It learns representations by maximizing agreement between differently augmented views of the same data example through a contrastive loss in the latent space. Initially, a randomly selected patch is resized to 224*224. Then a random horizontal flip followed by a color distortion are applied. In the end, Gaussian blurring and solarization (this is optional) are applied to the chosen patch. A detailed explanation is in the following. Choosing a right augmentation is of importance in contrastive learning. For example, SimCLR does not perform properly once we remove color distortion from image augmentations. As an explanation, SimCLR demonstrates that crops of the same image usually share their colour histograms Chen et al. (2020). Therefore, when a contrastive task only depends on arbitrary crops as image augmentations, it can be solved mainly by focusing on colour histograms alone. Hence, the representation only keeps colour histograms information. To solve this , SimCLR adds colour distortion to its image augmentations. Instead, BYOL keeps any information seized by the target representation in its online network to enhance its predictions. Thus, even if augmented views of an identical image share the identical colour histogram, BYOL is still incentivized to maintain other features in its representation. Due to that, we believe BYOL is more potent for the selection of image augmentations than contrastive methods.

For self-supervised training, BYOL utilises following image augmentations Grill et al. (2020):

- Random cropping: a random patch of the image, with an area of 8% to 100% is chosen and then resized to 224*224 which is our target size;
- Left-right flip: it is a feature that turns an image horizontal or vertical;
- Colour jittering: the brightness, contrast and saturation of the image are changed or shifted uniformly and they are applied to the chosen patches.;

- Colour dropping: A conversion to greyscale is applied, namely, for a pixel with colour intensity (r,g,b), the greyscale intensity is computed as 0.2989r + 0.5870g + 0.1140b;
- Gaussian blurring: A kernel of 23*23 with a standard deviation uniformly sampled over [0.1,2.0] is utilised.
- Solarization: a color transformation is used to invert all pixels above a threshold value of magnitude.

3.2.3 Demonstration of BYOL workflow

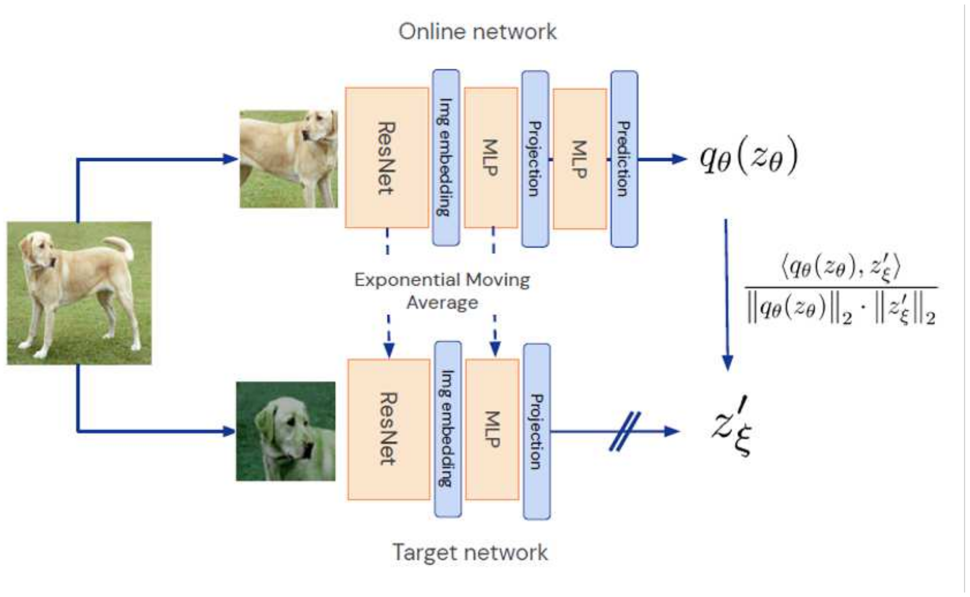


Figure 3.4: BYOL example and the two augmented views for an arbitrary input image He et al. (2017)
.

Figure 3.4 represents the visual architecture of the BYOL method. As explained, it does not utilize negative samples; instead, it directly lowers the similarity of feature representation of the identical picture under various augmentation. Negative pairs are pictures from different batches unlike positive pairs. BYOL requires smaller batch sizes which cause fewer computations. For simplicity, we can call the online network student, and the target network teacher. The teacher model is updated only through an exponential moving average from the student's parameters.

As mentioned earlier, an exponentially moving average of the online network's parameters is used for the target network. The network consists of an encoder (Resnet50) and a projector (dense layer with a ReLU). The online network has a predictor in addition which is the same as the projector. The architectures are shared between these two networks but not the weights. The output of the encoder module is used as the final representation in the downstream task. A single training iteration includes generating two augmentations of one image, sending them through the two networks to acquire the projections, and using the prediction module in the online (student) network to obtain the predicted projection of the target (teacher) network. The online network is updated by gradient descent, and the target network is updated by an exponential moving average.

3.2.4 Architecture used for pretraining

A 50-layer convolutional neural network, ResNet-50, He, Zhang, Ren, and Sun (2016), is used as our base parametric encoders (backbone) f_θ and f_ξ, as shown in figure 3.3. Followed by this encoder (backbone), we have a projection head which is a multi-layer perceptron (MLP). MLP is composed of one or more layers of neurons. Data is provided to the input layer; there may be one or more hidden layers providing levels of abstraction. Predictions are made on the output layer, also named the visible layer. The linear output of this MLP has 4096 dimensions. Therefore, this layer maps the output of adaptive average pooling layer in backbone (ResNet-50) that has 2048 to 4096 dimensions. Afterwards, the projection head module uses a ReLU activation function layer and another linear layer to map 4096 features to 256.

According to our experiments on COCO 2014 in pre-training using BYOL algorithm, we found that if we reduce the number of middle layer neurons in MLP from 4096 to 2048, the defined objective function for BYOL algorithm converges faster that is our goal. After the projection head with 2048 linear neurons, we design a prediction head module whose number of linear neurons is set to 256 based on experiments. This prediction head module has exactly the same architecture as projection head. In addition, in our pretraining with BYOL, instead of adopting random initial weights, we employed ImageNet initial weights which decreases the convergence time as well as improves the final results in instance segmentation.

3.3 Downstream task: instance segmentation

At first, a quick explanation of Mask R-CNN is provided and then the implementation is described:

3.3.1 Mask R-CNN

Mask R-CNN is a state-of-the-art model for instance segmentation which extends Faster R-CNN by adding a similar unit to bounding box recognition branch S. Ren et al. (2015) for predicting segmentation masks. Since Mask R-CNN is a newer version of Faster R-CNN, we will first take a look at Faster R-CNN.

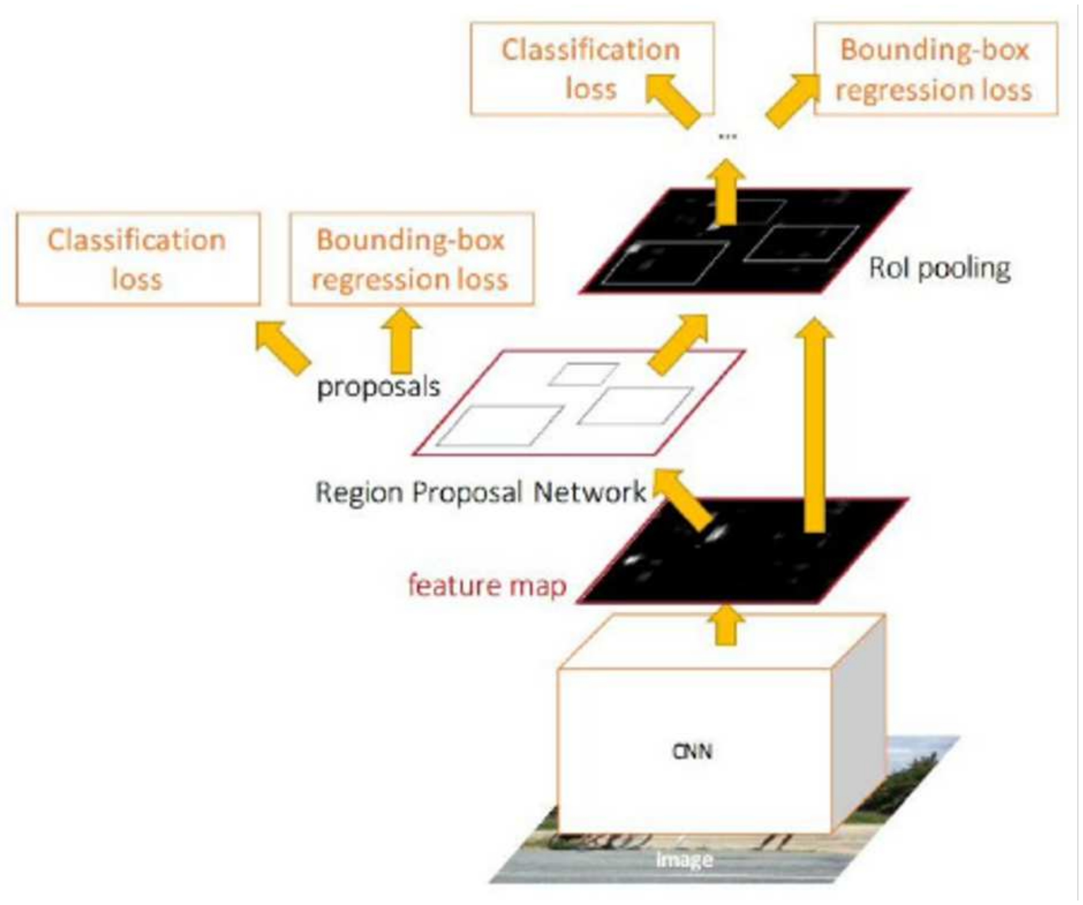


Figure 3.5: System flow in Faster R-CNN S. Ren et al. (2015)

.

As shown in figure 3.5, the system, called Faster RCNN, consists of two phases. The first phase is a deep convolutional network generating proposals, and the second phase is a Fast RCNN Girshick (2015) making use of these proposed regions. The following gives a detailed description of the two phases.

Phase 1: The first phase as shown in figure 3.6, is a deep convolutional network with Region Proposal Network (RPN), which proposes regions of interest (ROI) from the feature maps output by the convolutional neural network. The input image goes into a CNN, called backbone, which is usually a pretrained network such as ResNet50. Figure 3.6 shows that the RPN utilizes a sliding window strategy to estimate anchor boxes from the feature maps. At each spatial window, multiple anchor boxes are of pre-calculated scales and aspect ratios, allowing the model to detect objects of a broad range of scales and aspect ratios in the image. Then we will have a binary classification (cls1) showing whether the anchor has an object or not (and categorize into classes foreground and background) and then a bounding box regression (bbox1) will refine bounding boxes. If the anchor has highest intersection over union (IOU) with respect to ground truth, it is categorized as positive label which is foreground class. At each sliding window area, various proposals (max k) are anticipated corresponding to anchor boxes. So, the reg layer has 4k results containing the locations of k boxes, and the classification layer shows 2k scores that calculate likelihood of existence of an object for every region of proposal.

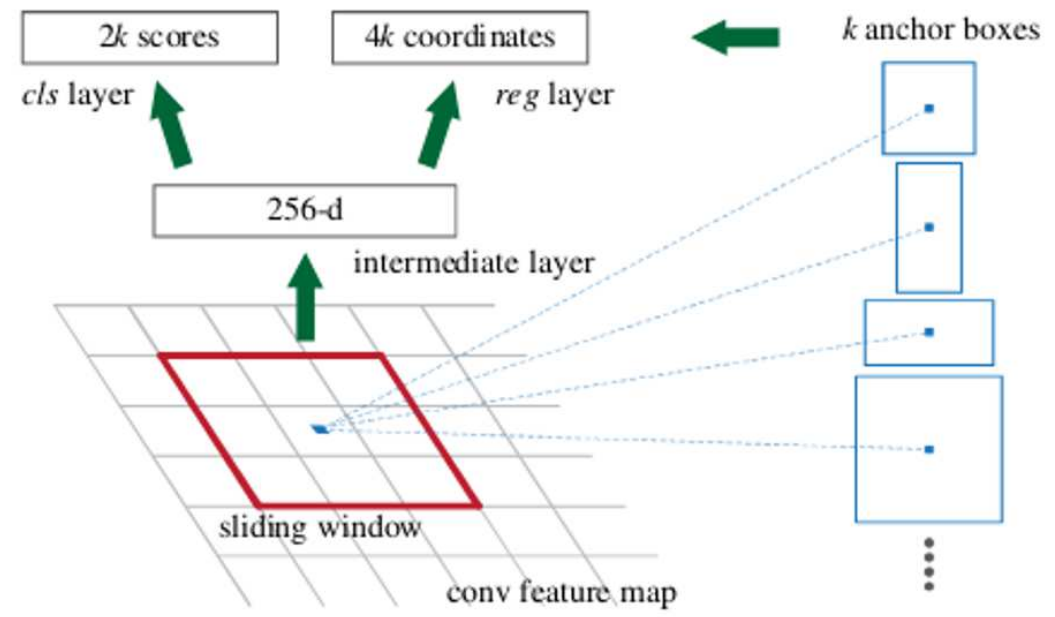


Figure 3.6: Region proposal network S. Ren et al. (2015)

.

In Faster R-CNN there are two losses that are bbox binary classification loss, L_{cls} and bbox regression loss, L_{bbox}. Binary classification loss is a loss function that is used in binary classification tasks that answer a question with only two choices such as yes or no. The bounding box loss is a loss function that provides the error between the predicted and ground truth bounding box.

Hence, at this stage, there are two losses i.e. binary classification loss, Lcls1 and bounding box regression loss, Lbbox1.

The top (positive) anchors output by the region proposal network (RPN), called proposals or ROI are handed to the following phase.

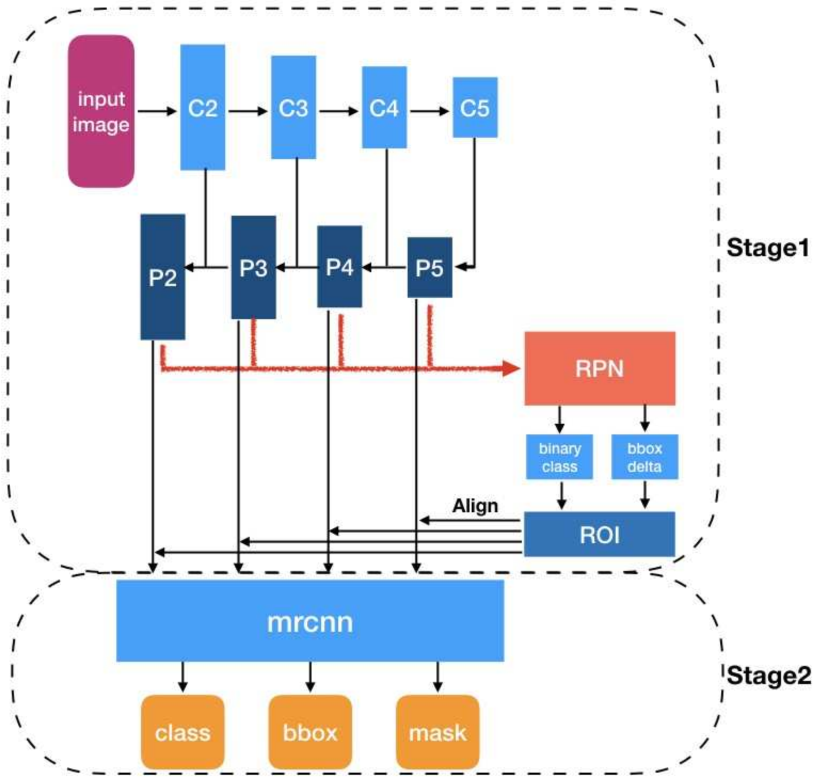


Figure 3.7: Representation of the two phases of Mask R-CNN He et al. (2017)

.

Phase 2: The subsequent stage is basically Fast R-CNN in addition to ROI pooling layer, separates feature maps from every ROI, and performs classification and bounding box regression. The ROI pooling layer changes over the segment of feature map corresponding to every (variable estimated) ROI into fixed size to be sent into a fully connected layer. For instance, as shown in figure 3.8, for a 8x8 feature map, the ROI is 7x5 in the base left corner, and the ROI pooling layer sends out a proper size 2x2 feature map. Then, at that point, the accompanying activities would be performed:

- Divide the ROI into 2*2;
- Perform max-pooling on each subsection;

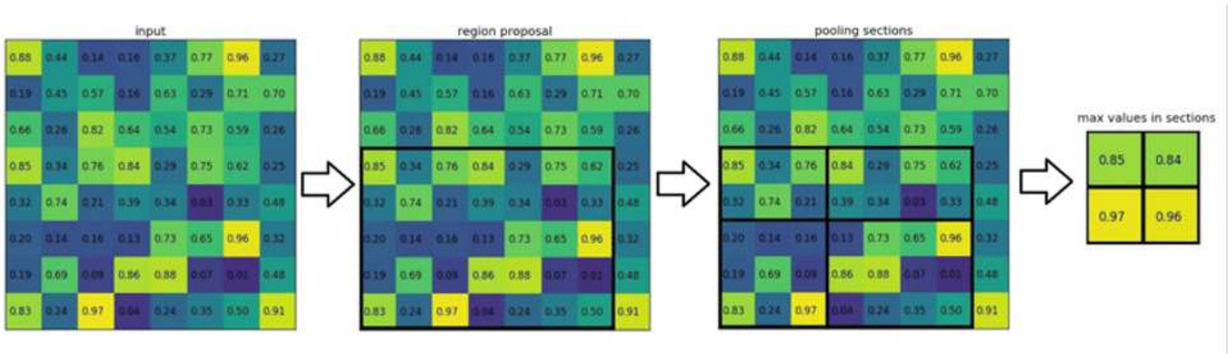


Figure 3.8: Region projection and pooling section

.

Then the softmax classification of objects such as vehicles, people, etc. is performed by fc layer and at the second stage, a multi-task loss on each sampled RoI is defined i.e., object classification loss (into multiple classes), Lcls2, and bbox regression loss, Lbbox2.

Mask R-CNN employs the same first phase as the Faster R-CNN but its second phase comprises binary mask prediction as well as class score and bounding box as shown in figure 3.7. The mask uses positive ROIs and predicts mask utilising fully convolutional network (FCN). In other words, Mask R-CNN is Faster R-CNN plus FCN as shown in figure 3.9.

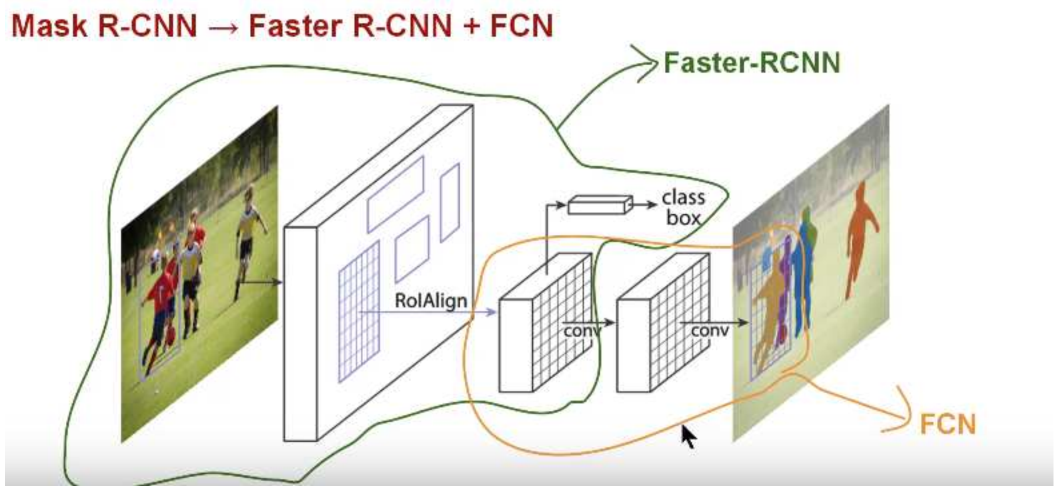


Figure 3.9: Mask R-CNN architecture He et al. (2017)

.

The loss function of Mask R-CNN is defined as:

$$L = Lcls + Lbbox + Lmask \tag{11}$$

Suppose we train for an RoI with a ground-truth class g and a bounding-box regression target r. Then p is the output of our softmax sibling layer, which is a probability distribution on the K data classes, including the background class. We represent the output of the regressor sibling layer as

$$t^k = \left(t_x^k, t_y^k, t_w^k, t_h^k\right) \tag{12}$$

a tuple of bounding box for each class k. As a result, our multi-task loss is given by:

$$L\left(p, g, t^g, r\right) = L_{cls}(p, g) + \lambda[g \geq 1] L_{loc}\left(t^g, r\right) \tag{13}$$

with,

$$L_{cls}(p, g) = -\log p_g \tag{14}$$

where p_g is the probability of class g. In equation (12), $L_{loc}\left(t^g, r\right)$ represents a smooth L_1 loss which is defined as:

$$L_{loc}\left(t^{g}, r\right)=\sum_{i \in\{x, y, w, h\}} \operatorname{smooth}_{L_{1}}\left(t_{i}^{g}-r_{i}\right) \tag{15}$$

where

$$\operatorname{smooth}_{L_{1}}(x)=\begin{cases}0.5 x^{2} & |x|<1 \\ |x|-0.5 & \text{otherwise}\end{cases} \tag{16}$$

Furthermore, λ indicates a hyperparameter (often assigned to 1 in He et al. (2017)) that specifies the relative weight of the regression loss vs. the classification loss to the overall loss.

The Lcls (Lcls1+Lcls2) is the classification loss, which tells how correct the predictions are to the right class, and Lbbox (Lbbox1+Lbbox2) is the bounding box loss, which tells how accurate the model is at localization, as discussed above.

In Mask R-CNN, there is also Lmask, as well as two mentioned losses, for mask prediction, which is calculated by taking the binary cross-entropy between the predicted mask and the ground truth. This loss penalizes wrong per-pixel binary classifications. Mask R-CNN assigns a binary mask for each class for each of the RoIs, and the mask loss for a particular RoI is calculated based only on the mask corresponding to its true class, which helps prevent the mask loss from being affected by class predictions.

In total there are five losses used in Mask R-CNN:

- RPN-class-loss, Lcls1: RPN (bbox) anchor binary classifier loss;
- RPN-bbox-loss, Lbbox1: RPN bbox regression loss;
- fastrcnn-class-loss, Lcls2: loss for the classifier head of Mask R-CNN.
- fastrcnn-bbox-loss, Lbbox2: loss for Mask R-CNN bounding box refinement.
- maskrcnn-mask-loss, Lmask: mask binary cross-entropy loss for the mask head

3.3.2 Training our model for instance segmentation

Mask R-CNN is used to conduct our instance segmentation downstream task. To train Mask R-CNN, we first import all required libraries and datatset. 20,000 images are used to train our model, 5,000 for validation and 5,000 for test. afterwards, we need to transfer our initial weights obtained in our pre-training section. In our implementation, we used Detectron 2 which is FAIR's (Facebook AI research) next-generation platform for object detection and segmentation. As discussed, we have three different initial weights to start our training and in the first, initial weights acquired by self supervised pre-training are used. then, the results are compared to the same models with random initial weights and ImageNet initial weights. Using the training dataset, the weights are being fine-tuned. The training for downstream task was conducted on Google Colab using NVIDIA Tesla K80.

3.3.3 Background removal

Using the masks generated by instance segmentation, we can apply a post-processing to remove the background of any input images. In this research we assume that images have one instance. First we read the image and then predict the instance using DefaultPredictor. DefaultPredictor Create a simple predictor with the given configuration that runs on a single input image to predict the instance. Then we create a mask from the predicted output on image and remove the background of object by checking the value of every pixel and if the pixel does not belong to the mask, that pixel is background and its value is set to zero. The following is the pseudo code for the background removal:

Algorithm 1 Background removal

```
Result: Background removed images
Input : Load instance segmentation model to make mask prediction
Im                                          load an image to remove its background
Set training data-set path
predictor = DefaultPredictor(cfg)           Create predictor (model for inference)
mask = outputs["instances"].pred_masks      predict the masks in the image
i = len(mask[0][0])                         number of the rows of the image
j = len(mask[0])                            number of the columns of the image
test = im (mask[0])                         store the original image in a variable
while i,j are less than the number of rows and columns do
    if not mask then
        test[j1,i1] = 255;                  it is background
    else
        it is foreground;
    end
end
Output : Background removed image
```

3.4 Summary

The detailed explanation of our method for instance segmentation and background removal is given in this chapter. This system is composed of three parts, self-supervised pre-training, instance segmentation using Mask R-CNN and background removal. The pretrainig is to learn the knowledge needed for the downstream task by solving the pretext tasks and then we use the backbone (weights) to better conduct our instance segmentation. Then the result of instance segmentation which is pixel-wise localisation of instances is used for background removal and Finally, the post-processing was added to remove the background from input images using instance segmentation mask predictions. In addition, the proposed method which is instance segmentation followed by background removal using BYOL self-supervised initial weights is compared with two other existing methods, namely, instance segmentation and background removal once using ImageNet initial weights and the other time using random initial weights.

In conclusion, the proposed method designed based on self-supervised learning segments instances precisely and we obtained very good results in instance segmentation and background removal.

Chapter 4

Evaluation

4.1 Introduction

The proposed framework for instance segmentation and background removal has been explicated in detail in the previous chapter. It will be trained and tested with commonly used dataset, namely COCO 2017, for its performance evaluation in this chapter. As discussed, instance segmentation using a self-supervised method called BYOL is implemented and compared to two other existing methods. The details concerning conducted experiments in self-supervised pretraining step using BYOL algorithm to be trained and tested with COCO 2014 dataset are elaborated. The experimental work is divided into two main steps which are: 1. Self-supervised weights pretraining using BYOL algorithm on COCO 2014 train dataset. 2. Transferring acquired weights from the last step to instance segmentation (downstream task) and using COCO 2017 dataset to train and test, followed by background removal. All training and testing were performed on NVIDIA TESLA K80.

4.2 Dataset

To conduct our experiments in the thesis both versions of COCO dataset are used. For self-supervised pretraining, 2014 version of COCO is used that contains 82,783 training images. For the downstream task which is instance segmentation followed by background removal, 20k images are randomly selected from COCO dataset 2017 for training. Then, 5k images are selected as validation data and 5k as test data.

As we know, COCO dataset contains 81 different classes labelled both box-wise and pixel-wise and are mostly utilised for dense prediction problems such as object detection and instance segmentation. Table 4.1 represents the distribution of intances in validation data:

category	#instances	category	#instances	category	#instances
person	10777	bicycle	314	car	1918
motorcycle	367	airplane	143	bus	283
train	190	truck	414	boat	424
traffic light	634	fire hydrant	101	stop sign	75
parking meter	60	bench	411	bird	427
cat	202	dog	218	horse	272
sheep	354	cow	372	elephant	252
bear	71	zebra	266	giraffe	232
backpack	371	umbrella	407	handbag	540
tie	252	suitcase	299	frisbee	115
skis	241	snowboard	69	sports ball	260
kite	327	baseball bat	145	baseball gl..	148
skateboard	179	surfboard	267	tennis racket	225
bottle	1013	wine glass	341	cup	895
fork	215	knife	325	spoon	253
bowl	623	banana	370	apple	236
sandwich	177	orange	285	broccoli	312
carrot	365	hot dog	125	pizza	284
donut	328	cake	310	chair	1771
couch	261	potted plant	342	bed	163
dining table	695	toilet	179	tv	288
laptop	231	mouse	106	remote	283
keyboard	153	cell phone	262	microwave	55
oven	143	toaster	9	sink	225
refrigerator	126	book	1129	clock	267
vase	274	scissors	36	teddy bear	190
hair drier	11	toothbrush	57		
total	36335				

Table 4.1: Self-supervised learning methods accuracy on different computer vision tasks Jing and Tian (2020)

.

4.3 Evaluation metrics

This section describes the evaluation metrics used to compare the performances of different models.

4.3.1 Intersection over union (IoU)

Intersection over union (IoU) is a metric that measures how perfectly a prediction, such as a mask, overlays with the ground truth. More specifically, it is described as:

$$IoU = \frac{\text{area (prediction } \cap \text{ ground truth)}}{\text{area (prediction } \cup \text{ ground truth)}} \tag{17}$$

A perfect overlap gives IoU = 1, and if there is no overlap IoU = 0. Figure 4.1 below depicts a graphical illustration of the concept of Intersection over Union (IoU).

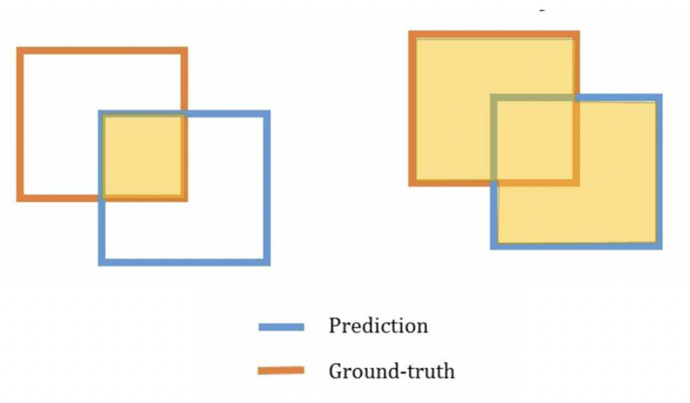


Figure 4.1: The left subfigure is the area of the intersection and the right subfigure is the area of union

.

4.3.2 Average precision

One standard metric used to assess instance segmentation tasks is average precision (AP) for some IoU threshold. AP is defined based on two metrics, namely precision and recall. Employing the subsequent denotations:

- TP = true positive (the number of correctly predicted objects)
- FP = false positive (the number of predictions that incorrectly predicted the presence of an object)
- FN = false negative (the number of missed objects)

the precision and recall are defined as:

$$\text{recall} = \frac{TP}{TP+FN} \tag{18}$$

$$\text{precision} = \frac{TP}{TP+FP} \tag{19}$$

where a prediction is usually categorized as a true positive (TP) if the IoU between the prediction and a ground truth is bigger than some threshold. Otherwise, it is categorized as FP. So the average precision is calculated as:

$$AP = \int_0^1 p(r)dr \tag{20}$$

where p(r) is the precision at recall r. Estimating the AP over some interpolated points for every recall value is also common. The average precision with N interpolation points is:

$$AP = \frac{1}{N} \sum_{r\in\{0.0,\ldots,1.0\}} p_{\text{interp}}(r) \tag{21}$$

where Pinterp(r) is the maximum precision for recalls greater than r, i.e. Everingham, Van Gool, Williams, Winn, and Zisserman (2010),

$$\text{Pinterp}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{22}$$

4.3.3 Cosine similarity

Cosine similarity is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words. It is defined as the division between the dot product of two vectors and the product of the euclidean norms or magnitude of each vector. Let A and B be two vectors for comparison. Using the cosine measure as a similarity function, we have

$$similarity(A,B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{23}$$

where $||A||$ is the Euclidean norm of vector A = (A_1 , A_2 , A_3 , ..., A_p) defined as $\sqrt{A_1^2 + A_2^2 + \cdots + A_p^2}$ and the same definition applies to $||B||$.

4.4 Pretraining using BYOL algorithm

To learn visual operators that are useful and transferable to the downstream task like instance segmentation and then background removal, BYOL algorithm is selected and pre-trained on COCO 2014 dataset. As mentioned ealier, BYOL algorithm has Siamese architecture so that only one side of this architecture weights are updated by back propagation algorithm. The architecture of this algorithm is totally symmetric and the only difference is in stop-gradient operator which applies to one part of the model. To run our expretiments, ResNet50 is utilised as backbone in BYOL model. As we know, there are various types of ResNet such as ResNet18, ResNet101, etc. Altough ResNet50 is more preferred as it has more complex patterns learning capacity and less computational resources and it becomes a standard in contrastive visual operator learning. As mentioned earlier, we use this model as backbone in our experiments. Following this backbone, we have projection head which is a MLP whose linear layer output has dimensions of 4096. Hence, this layer maps the output of Adaptive Average Pooling in ResNet50 backbone that has 2048 layers to a vector with 4096 numbers. This linear layer is followed by a projection head module using a ReLU activation function layer and eventually another linear layer is used to map the 4096 dimensional vector to 256 dimension space. According to the experiments conducted in pretraining step using BYOL algorithm, we have found experimentally that if we reduce the number of middle layer neurons from 4096 to 2048, the objective function defined for the BYOL algorithm converges faster, which is desirable. Following the projection head module, the BYOL algorithm has a prediction head module that has exactly the same architecture as projection head. For this module, based on our experimental tests, the number of linear layers is set to 2048 and 256, respectively. It should be noted that at each time of pre-training, after making any modifications like changing the number of linear layers in the BYOL architecture, we have trained the resulting model on COCO2014 dataset and the mode in which the model could minimize the loss function the best is selected as the final model whose weights can be used in our downstream task. Furthermore, in self-supervised pretraining, batch size is set at 64 and for 250k and 350k iterations, we have trained this model to choose the best mode where the cost function is minimized. Figure 4.2 shows the changes in the cost function of the BYOL algorithm with respect to the number of iterations of each experiment:

According to figure 4.2, we can see that the dark orange model has been able to minimize the cost function related to the BYOL algorithm, which is actually a cosine similarity function. The hyperparameters

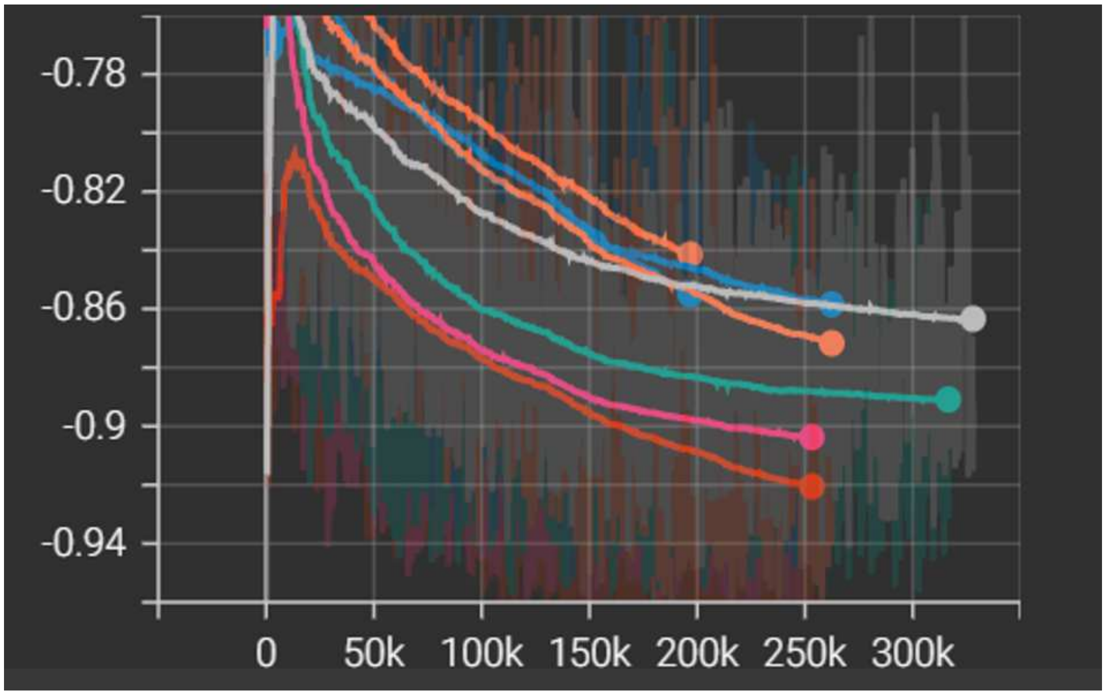


Figure 4.2: Loss function changes over iteration

.

of the model whose cost function is marked in dark orange color are as follows: 2048 and 256 neurons are placed in the linear layers of the projection head module, respectively. Projection head is shown in figure 3.3.

Figure 4.3 below shows how the learning rate changes with respect to the number of iterations:

As shown in figure 4.3, we first set the learning rate to large values and then after a little while of training the model, we decrease the learning rate by using a learning rate schedule in which the learning rate changes over time (training epochs). We have set other hyperparameters such as weight decay and gradient clip to 1e-4 and 1e-1 respectively. We have used the PyTorch and Pytorch-Lightning frameworks to implement the mentioned items. Pytorch-Lightning is a wrapper written on PyTorch that makes training self-supervised learning models very simple. Also, this framework is very suitable for distributed training or training using hardware equipped with TPU. In the following sections, we carry out the second stage of the work and use the obtained weights to solve the instance segmentation problem.

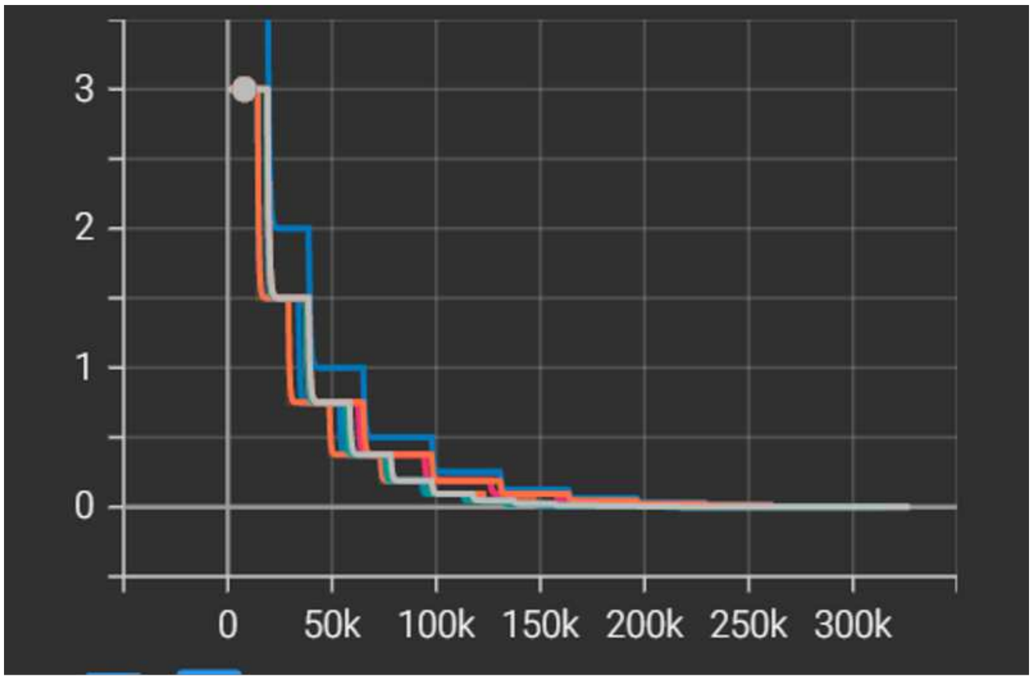


Figure 4.3: Learning rate changes over iteration

.

4.5 Mask R-CNN hyperparameters fine-tuning

For Mask R-CNN we constructed a model using ResNet50-FPN backbone. The model is trained for 100000 iterations with 8 images per batch and by experimenting with different values of learning rates in our work, lr=0.03 is chosen and we set two step decays in which the learning rate changes over time (training epochs) at 40,000 and 80,000. Learning rate schedule seeks to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule. In other words, we use adaptive learning rates instead of a constant learning rate. The dataloader number of workers is set to 4. $Num\_workers$ tell the data loader how many sub-processes to use for data loading. If the num_worker is zero, the GPU has to wait for the CPU to load data. Theoretically, having greater $num\_workers$ results in more efficiently loading data in the CPU and less wait time for GPU. For the training, we firstly use initial weights generated by BYOL self-supervised algorithm and then compare it with random initial weights and ImageNet initial weights results in the end.

4.6 Testing results

As mentioned earlier, the results are compared in three different conditions. In the first subsection, four losses , namely, bounding box loss, classification loss, mask loss and total loss for three different methods are shown. In the following subsection, the average precision (AP) of all three methods are calculated and shown. Finally, a visual demonstration of background removal using these three methods is provided and a numerical and visual comparison were made between these methods.

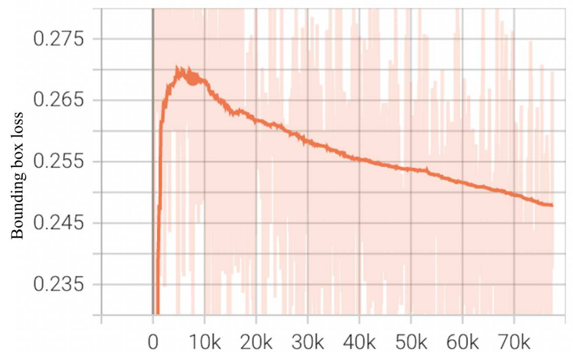
In the following charts, there are two curves shown in each subfigure. The faded curves show the actual values, and the solid lines represent the same curve after smoothing using an exponential moving average. The function that carries out this smoothing is defined as:

$$V_t = \beta V_{t-1} + (1 - \beta)\theta_t \tag{24}$$

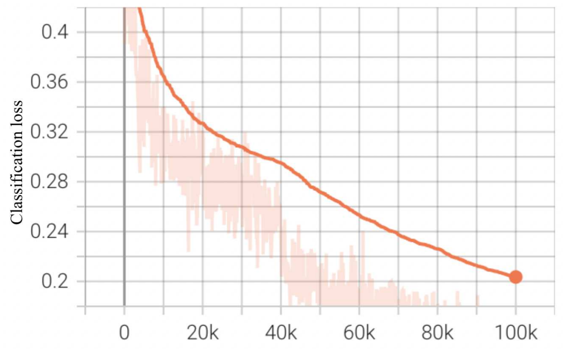
where V_t is the value of moving average at epoch t, β is the smoothing factor which is set to 0.99 and θ_t is the value at epoch t.

4.6.1 Loss charts

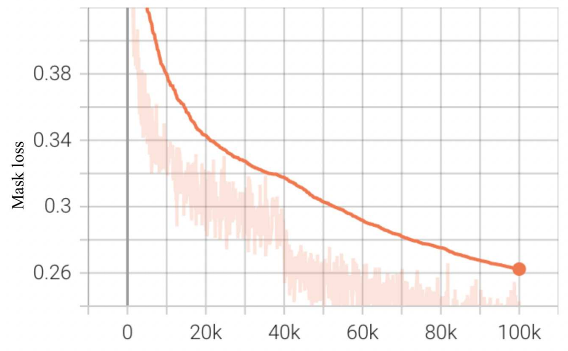
Loss is the penalty for a poor prediction. It is a number indicating how bad the model's prediction was on a single example. If the model's prediction is ideal, the loss is zero; otherwise, the loss is greater. Training a model aims to discover a set of weights and biases that have a low loss. The first results, figure 4.4, are representing the losses of instance segmentation using BYOL self-supervised method. A loss function is a measurement of how satisfactorily a prediction model is able to predict the desired outcome. As mentioned, the model is trained on 20,000 images of COCO dataset 2017. In each series of experiments, four losses related to MASK R-CNN which are loss_box, loss_cls, loss_mask and total loss are shown:



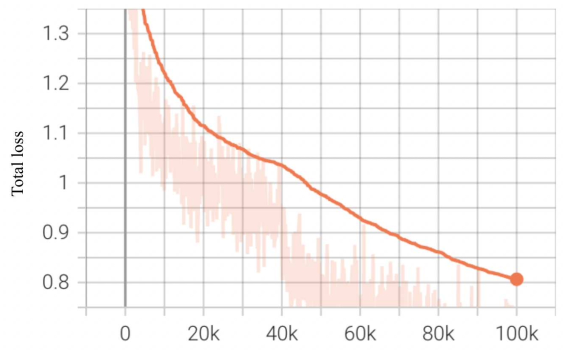
(a) Bounding box loss



(b) Classification loss



(c) Mask loss

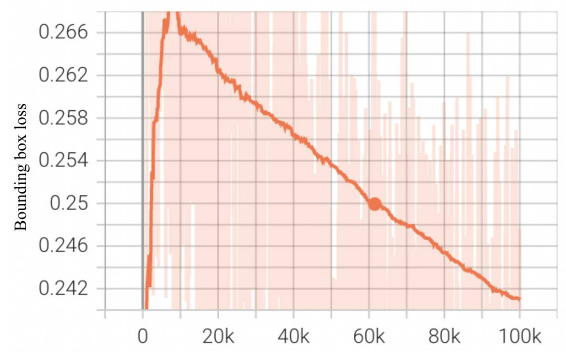


(d) Total loss

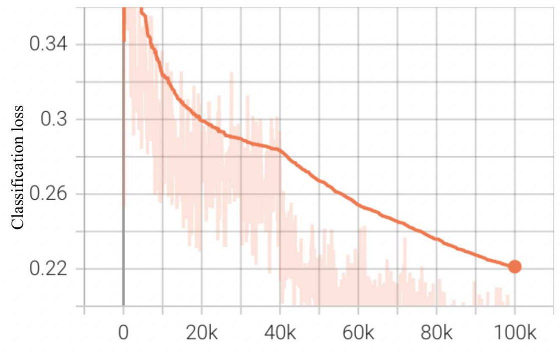
Figure 4.4: These four charts represent losses using BYOL algorithm initial weights to train our model: a) Top left sub_figure represents bounding box loss of training over iteration. b) Top right sub_figure represents classification loss of our training over iteration. c) Lower left sub_figure represents mask loss of our training over iteration. d) Lower right sub_figure represents total loss which results from the three other losses.

In figure 4.4, it is seen that in all subfigures, the loss curves are trending downward which implies that the model is performing well.

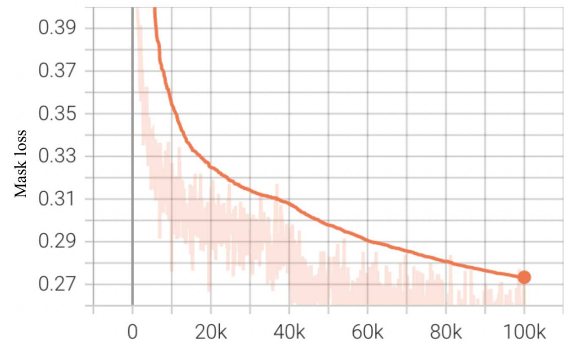
In figure 4.5, we depict the same charts which are bounding box loss, classification loss, mask loss and total loss using fully supervised ImageNet initial weights.



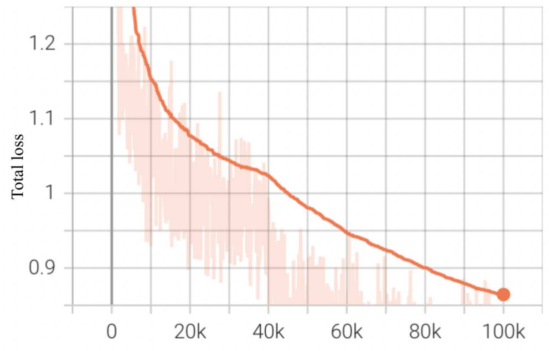
(a) Bounding box loss



(b) Classification loss



(c) Mask loss

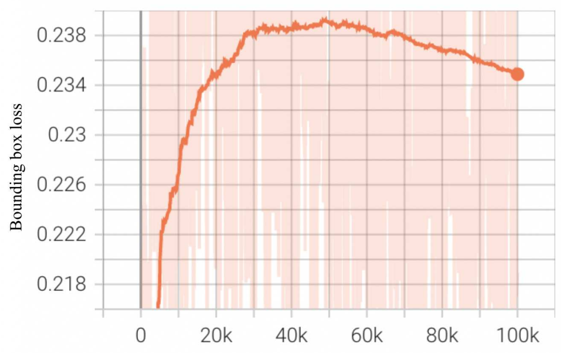


(d) Total loss

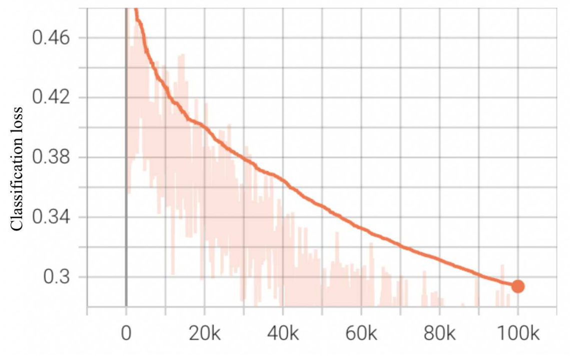
Figure 4.5: These four charts represent losses using ImageNet initial weights to train our model. a) Top left sub_figure represents bounding box loss of training over iteration. b) Top right sub_figure represents classification loss of our training over iteration. c) Lower left sub_figure represents mask loss of our training over iteration. d) Lower right sub_figure represents total loss which results from the other three shown losses.

In figure 4.5, we observe that the curves are converging. the final value of each loss curve is shown in table 4.2.

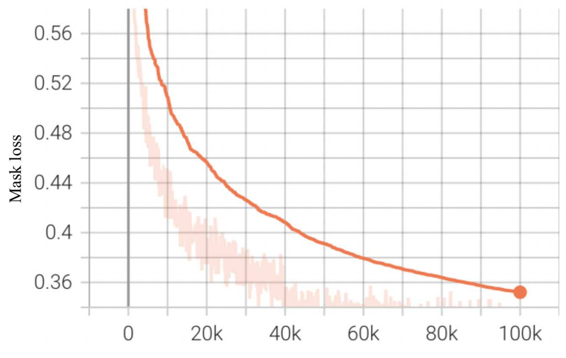
The final experiment was done using random initial weights on the same 20,000 images of COCO 2017. Like before, there are 4 losses to be shown which are bounding box loss, classification loss, mask loss and total loss. In figure 4.6, these charts are presented:



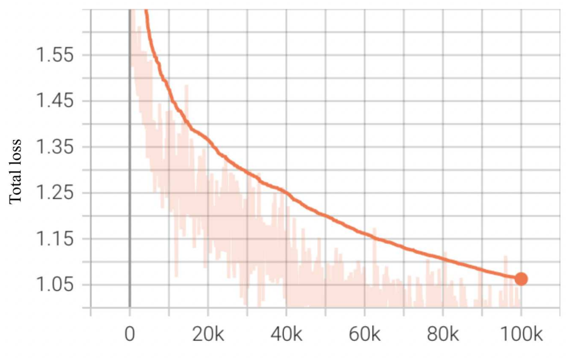
(a) Bounding box loss random initial weights



(b) Classification loss random initial weights



(c) Mask loss random initial weights



(d) Total loss random initial weights

Figure 4.6: These four charts represent losses using random initial weights to train our model. a) Top left sub_figure represents bounding box loss of training over iteration. b) Top right sub_figure represents classification loss of our training over iteration. c) Lower left sub_figure represents mask loss of our training over iteration. d) Lower right sub_figure represents total loss which results from the addition of three shown losses.

The corresponding losses to our training for instance segmentation using three different initial weights are depicted in table 4.2 for each of the methods. The total loss when we are using random initial weights to train our model with is 0.9601 which shows the effectiveness of not using random initial weights in our training for instance segmentation.

Method	Loss BB	Loss Cls	Loss Mask	Total loss
ImageNet weights	**0.2316**	0.2572	0.2686	0.7574
BYOL weights	0.2456	**0.2369**	**0.252**	**0.7345**
From scratch	0.2934	0.3296	0.337	0.9601

Table 4.2: Final loss values of three methods for instance segmentation after 100 epochs.

As it can be seen, our proposed framework for instance segmentation using BYOL initial weights outperforms the instance segmentation using ImageNet initial weights by 0.0229 in total loss. Moreover, we are able to see the effectiveness of transfer learning in a self-supervised way as we achieve lower total loss value using BYOL algorithm compared to the two method using ImageNet initial weights despite using way lower number of images for pretraining.

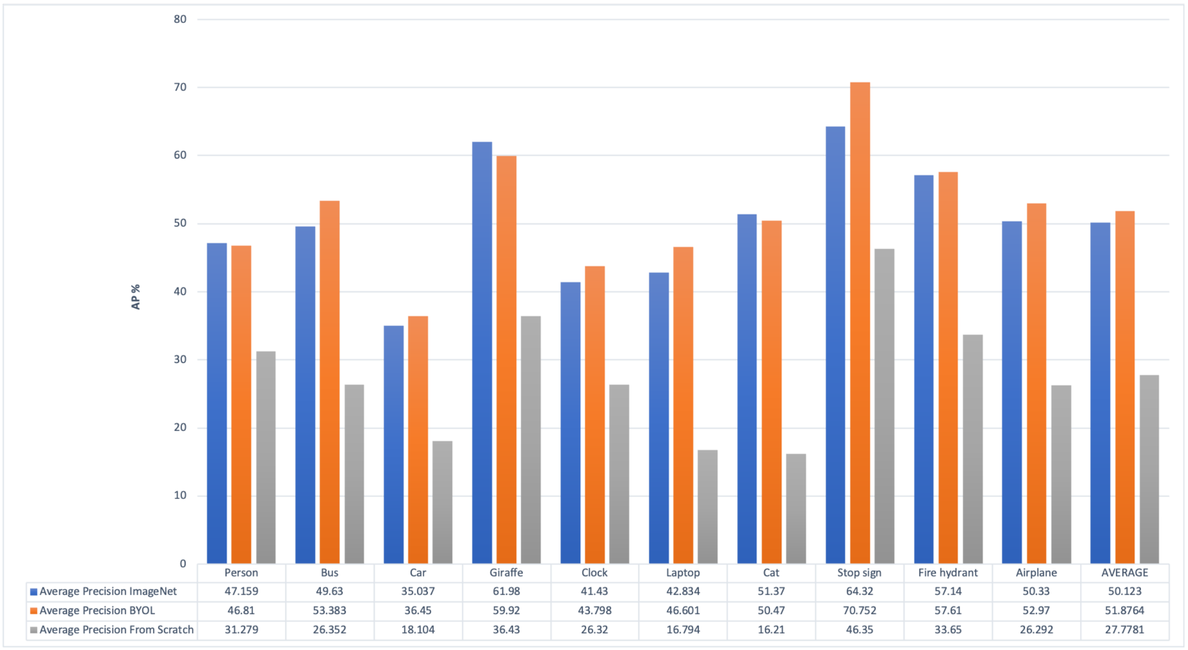
4.6.2 Average precision (AP)

In table 4.3, the Average Precision (AP) of each of the models is shown for comparison. We evaluate with instance segmentation using Mask R-CNN architecture He et al. (2017), then fine-tune with 20,000 images of COCO 2017 and report the results with 5,000 images of test COCO 2017 using the standard AP, AP50, AP75, AP_s, AP_m and AP_l metrics.

Method	Backbone	AP	AP50	AP75	AP$_s$	AP$_m$	AP$_l$
ImageNet weights	ResNet-50 FPN	27.62	43.764	30.152	**13.366**	28.90	36.98
BYOL weights	ResNet-50 FPN	**28.486**	**44.474**	**31.094**	10.03	**30.98**	**38.573**
From scratch	ResNet-50 FPN	13.56	23.496	14.133	4.774	13.96	19.254

Table 4.3: Instance segmentation benchmark results.

Furthermore, a comparison between APs in instance segmentation for different items is presented in figure 4.7. It represents how much better BYOL performs in segmenting ten classes out of all eighty classes defined by COCO dataset.



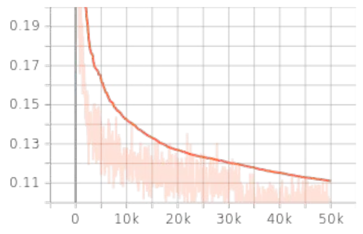
	Person	Bus	Car	Giraffe	Clock	Laptop	Cat	Stop sign	Fire hydrant	Airplane	AVERAGE
🟦 Average Precision ImageNet	47.159	49.63	35.037	61.98	41.43	42.834	51.37	64.32	57.14	50.33	50.123
🟧 Average Precision BYOL	46.81	53.383	36.45	59.92	43.798	46.601	50.47	70.752	57.61	52.97	51.8764
◼ Average Precision From Scratch	31.279	26.352	18.104	36.43	26.32	16.794	16.21	46.35	33.65	26.292	27.7781

Figure 4.7: APs of ten classes compared between our proposed method and two other existing methods

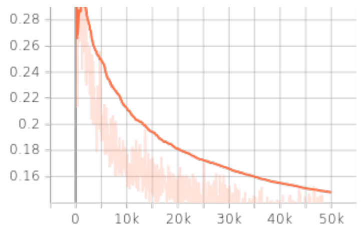
As can be seen in Figure 4.7, in these classes, for example, classes of giraffe, cat and person, the instance segmentation network with initial weights of self-supervised learning have a better performance. This is because the model has learned more distinctive features than the other two weighting methods because of the contrastive loss. Although the supervised initial weights based on the ImageNet dataset work well, in self-supervised learning, the model has learned better discriminator features by exploiting unlabeled data.

4.6.3 False negative, false positive and accuracy

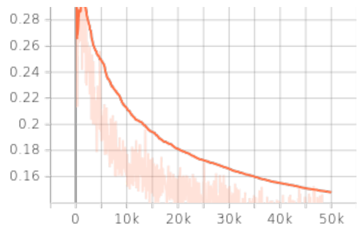
In addition to loss charts, false negative and false positive as well as the accuracy of our proposed method is compared to two other existing methods. In this experiment, the model is trained on the same dataset and for 50,000 iterations. The batch size is set to 8 and the data loader number of workers is set to 4. Learning rate is 0.03 like before and we set one step decay at 32,000. In the following parts, we will first compare false negative charts, then false positive charts and finally accuracy charts of these three methods. Figure 4.8 represents false negative charts for the three methods:



(a) False negative BYOL



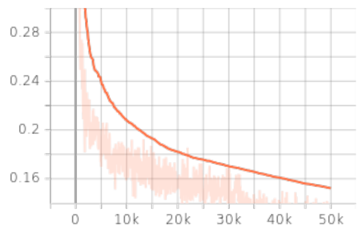
(b) False negative ImageNet



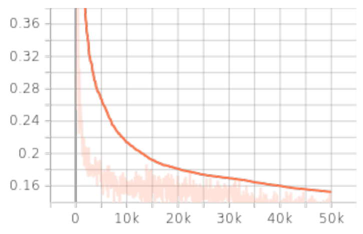
(c) Falase negative without initial weight

Figure 4.8: These three charts represent false negative rate of a) instance segmentation using BYOL self supervised weights b) instance segmentation using ImageNet fully supervised weights c) instance segemntation using random initial weights.

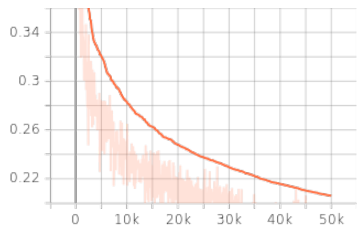
As it is shown in figure 4.9, the rate of false negatives decrease over time and our proposed method achieves comparable results compared to the method ImageNet weights are used and much better results than instance segmentation using random initial weights.



(a) False positive BYOL



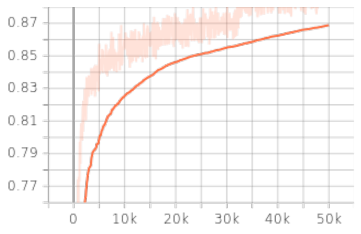
(b) False positive ImageNet



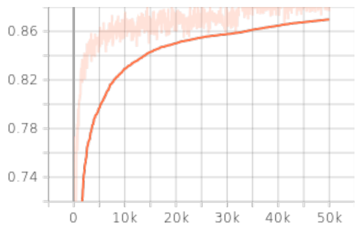
(c) False positive without initial weight

Figure 4.9: These three charts represent false positive rate of a) instance segmentation using BYOL self supervised weights b) instance segmentation using ImageNet fully supervised weights c) instance segemntation using random initial weights.

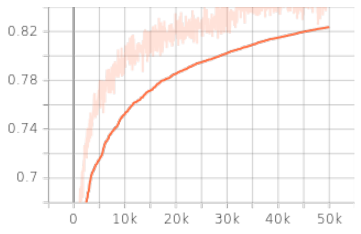
The results in figure 4.10 represent the performance of our proposed method and getting almost the same results as instance segmentation using ImageNet initial weights despite having much fewer labelled data. The results while not using initial weights or using random initial weights show the importance of initial weights in our final results.



(a) Accuracy BYOL



(b) Accuracy ImageNet



(c) Accuracy without initial weight

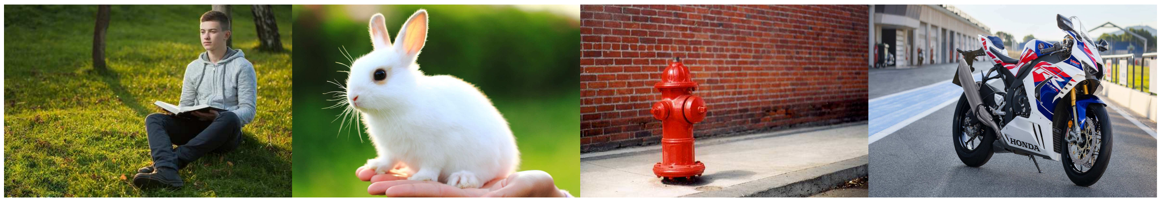
Figure 4.10: These three charts represent accuracy of a) instance segmentation using BYOL self supervised weights b) instance segmentation using ImageNet fully supervised weights c) instance segmentation using random initial weights.

According to the experimental results, the accuracy obtained from BYOL initial weights and that from ImageNet initial weights are so close.

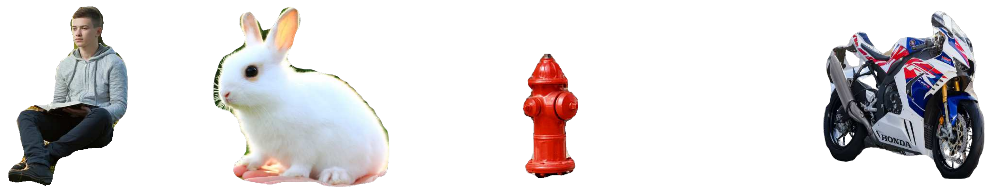
4.6.4 Background removal

After instance segmentation, our final goal is background removal using a post-processing process as explained in 3.3.3. In this section we are applying our model to different images with different background complexity and compare the results using three different methods. 1) Background removal using instance segmentation with our initial weights obtained from BYOL algorithm. 2) Background removal using instance segmentation with ImageNet initial weights. 3) Background removal using instance segmentation with random initial weights.

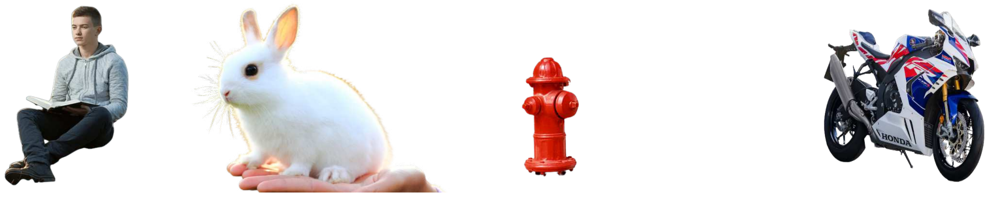
Figure 4.11 depicts the background removed images using our method with comparison to ground truth:



Original images



Background removing using BYOL algorithm to generate initial weights



Ground truth

Figure 4.11: The results of the proposed method are shown in the middle row. The upper photos are the original ones and the lower photos are the ground truth pictures.

The second series of results represent background removing using ImageNet initial weights along with ground truth for comparison as shown in figure 4.12:

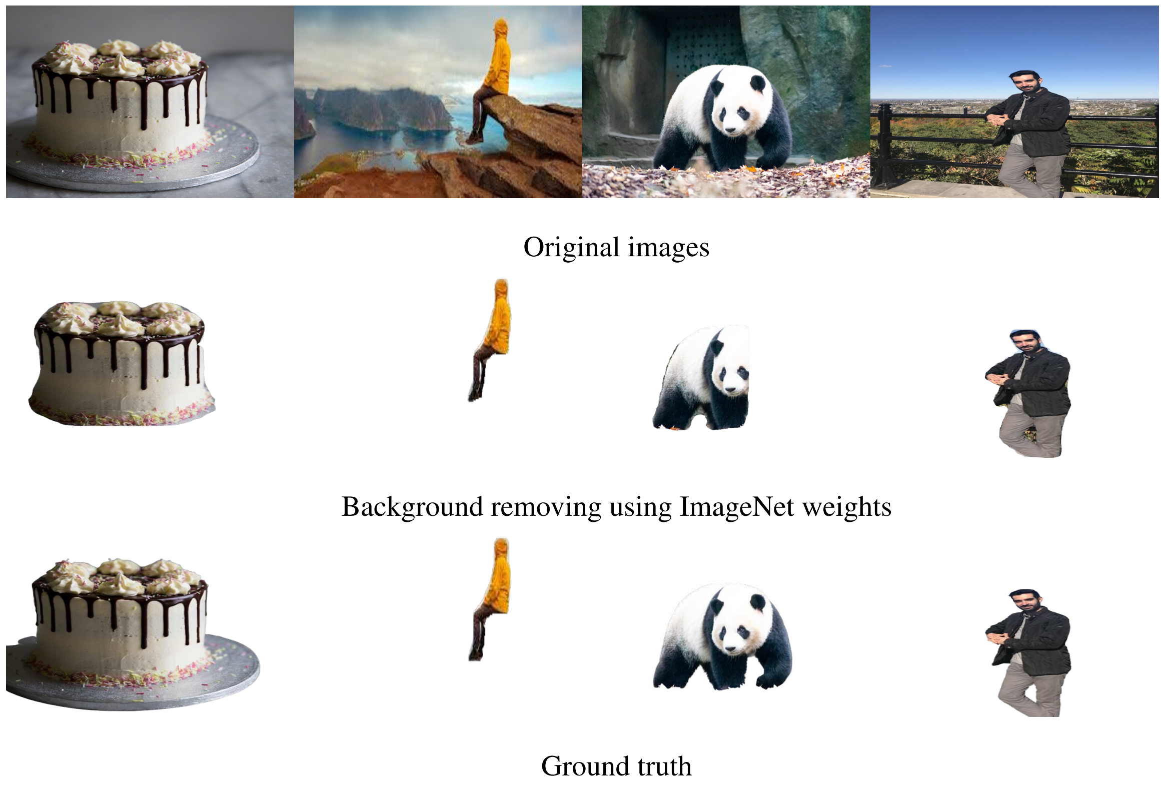


Figure 4.12: The results of the background removal using ImageNet initial weights are shown in the middle row. The upper photos are the original ones and the lower photos are the ground truth pictures.

The third and last series of the pictures are obtained by using random initial weights for training and they are compared with ground truth as shown is figure 4.13:

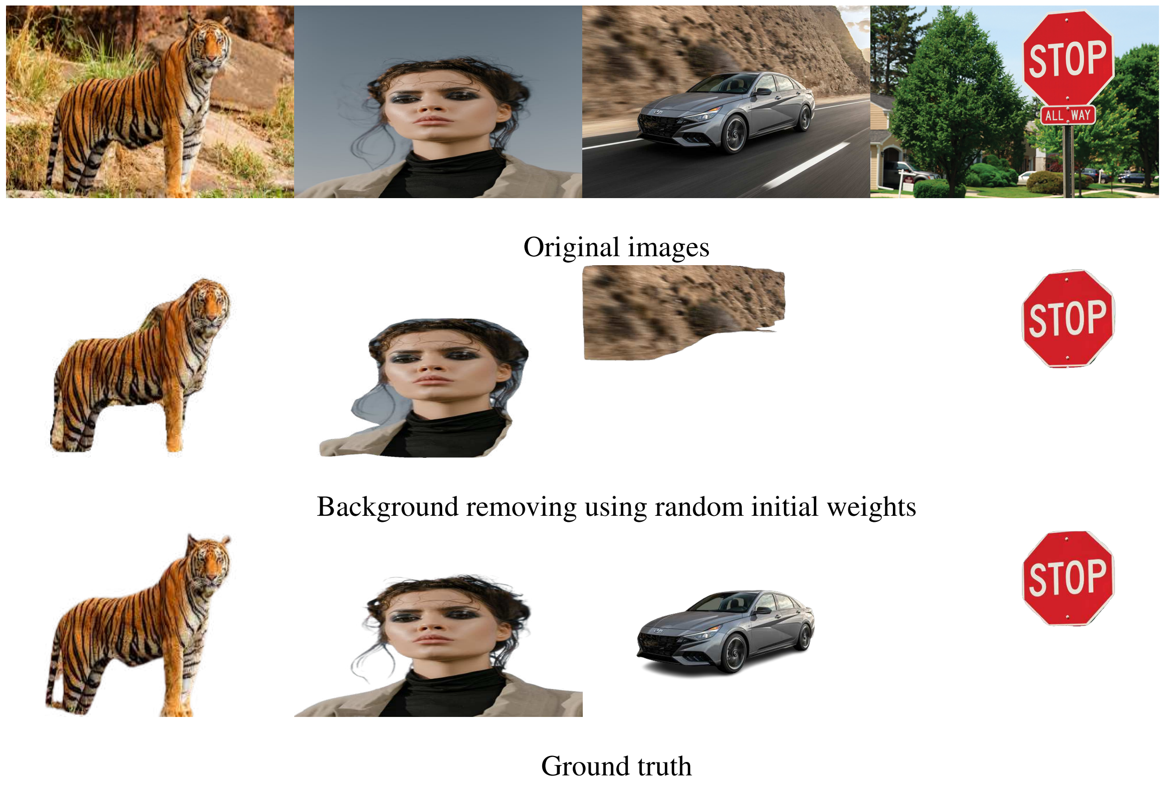


Figure 4.13: The results of the background removal using random initial weights are shown in the middle row. The upper photos are the original ones and the lower photos are the ground truth pictures.

In order to compare these methods on the same images, we now apply the models on a number of images and place them next to each other as shown in figure 4.14. This comparison shows the effectiveness of using BYOL initial weights in instance segmentation and background removal in practice which is because of the more distinctive features learned by the model compared to the two other methods. In addition to the visual comparison, we also measure the similarity of the results using cosine similarity. For every result obtained from each method, cosine similarity is measured using the result and ground truth from the same image. The output is a value in percentage meaning that the closer the obtained value measure is to 100%, the better the model performed the background removal. We compute this measure on 63 images and obtain the average score as shown in table 4.4 for each of three methods: Background removal using BYOL initial weights, background removal using ImageNet initial weights and Background removal using random initial weights:

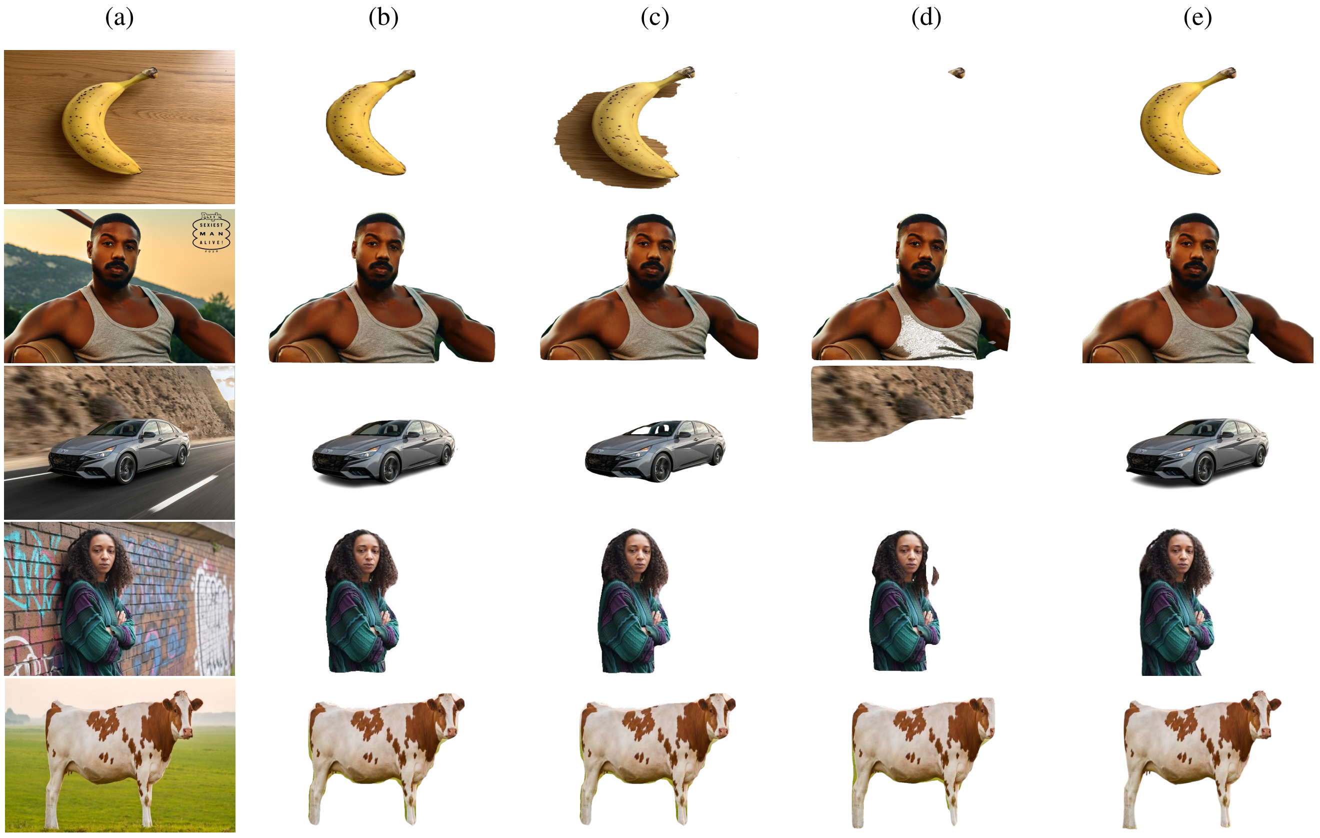


Figure 4.14: Comparison of three methods for background removal, Column (a) represents original images in each row, Column (b) represents background removal using BYOL initial weights, Column (c) represents background removal using ImageNet initial weights, Column (D) represents background removal using random initial weights and finally, column (e) represents ground truth for each row.

Method	BYOL	ImageNet	Random
Average	**91.83**%	89.11%	83.29%

Table 4.4: The average percentage of the results by calculating cosine similarity for 72 images.

According to the obtained average score, our proposed method outweighs background removal using ImageNet initial weights and that using random initial weights methods by 2.82% and 8.54%, respectively.

Training our proposed model using BYOL initial weights on Google colab took 38 hours for 100,000 iterations. Training the model with ImageNet initial weights took 36.5 hours approximately and with random initial weights took almost 37 hours for the same number of iterations. The greatest privilege that our proposed method has over ImageNet is that in our method we do not need labelled data in our pretraining phase which is way more cost effective and less time consuming. In other words, ImageNet uses about 1,000,000 labelled images to generate the weights used in computer vision tasks such as instance segmentation. In our proposed method, we only use 82,000 unlabelled images to generate the weights for instance segmentation and still obtain comparable results.

4.7 Summary

The proposed system is evaluated with dataset COCO 2017. It performs very well thanks to our self-supervised initial weights using modified BYOL. To test the consistency of the proposed system, the experiments have been repeated three times, i.e., training and testing the proposed system from initial state. The losses and average precision of the test results obtained from the three experiments are very similar. Moreover, our proposed framework is compared to two other cutting edge methods using same amount of training data and the results show that our proposed method can perform even better than those trained supervisedly.

To test the generalizability of the proposed system, the model is applied to 63 images of different complexity to remove their background. The obtained visual and numerical results represent that our method handles most of the conditions. The results of the proposed system are compared with those of the two CNN systems reported in recent years. The proposed system yields good results under the lowest computation complexity. It has: very good average precision, i.e., 28.486; the highest accuracy in cosine similarity compared with ground truth in background removal, i.e., 91.83%; and the best visual results as shown in figure

4.14. The high performance of the proposed system owes to the efficient self-supervised initial weights and the modifications made on the mid layers of the CNN in pretraining. The test results also confirm that the self-supervised initial weights efficiently contribute to the high performance.

Chapter 5

Conclusion

Instance segmentation and background removal are the two tasks we worked on in this thesis and they are of high importance in image segmentation. Instance segmentation is a special form of image segmentation that deals with detecting instances of objects and demarcating their boundaries. The accuracy of the segmentation has been very important and with that we can detect object boundaries or types. we use instance segmentation to do background removal and the accuracy of both instance segmentation and background removal using our proposed method are compared with two other existing methods in this thesis.

To achieve this goal, a new method which is a combination of three parts is developed, implemented and tested on different samples. this method consists of a pretraining being carried out using BYOL architecture and a training using Mask R-CNN and a post-processing to fulfill the background removal. Furthermore, we developed a method by which instance segmentation can be done not only for still images, but for different types of medical , multi spectral, gray scale images.

The pretraining block is implemented on COCO 2014 dataset containing around 82,000 training images and around 42000 validation and test set. This first step is to produce the initial weights being used in our next block as an initial point to improve the accuracy. All the images used in this section are 100% unlabelled.

The second block which segments instances and then removes backgrounds, is a Mask R-CNN using those generated initial weights. By fine-tuning these weights we achieve our goal of instance segmentation and background removal and then we compared our proposed framework with existing state-of-art methods.

The system has been trained and tested by using COCO 2017 dataset, and the results including the four losses which are bounding box loss, mask loss, classification loss and total loss are compared with two other state-of art methods. The number of iterations in this part is 100,000 on 20,000 training images of COCO 2017 and 5,000 test images of COCO 2017.

The final block concerns the step after instance segmentation, called background removal. In this block, we apply a post-processing to segmented images to acquire background removed images. 150 images are compared with two existing state-of-art methods and then to the ground truth.

To evaluate the consistency of the performance, each experiment is conducted 3 times. In each of the experiments, training data, validation and test data are kept the same to make our comparisons fair. The test results are compared to other existing methods. The average precision of our proposed system is 28.486 which is better compared to two other existing methods which are 13.56 and 27.62 for instance segmentation using random initial weights and instance segmentation using ImageNet initial weights, respectively. Moreover, in background removal, our proposed system over weighs background removal using random initial weights and background removal using ImageNet initial weights by 8.54% and 2.72%, respectively.

It should be noted that the proposed method can be used for any type of object detection, segmentation and background removal in images and videos. Compared to ImageNet that used 1.3 million labelled images to train and generate its initial weights for different purposes, in our system we used only about 82,000 images that are unlabelled and the final results suggest that our proposed method outperforms that method. The work presented in this thesis indicates that in order to train models for segmentation purposes, we do not have to label a huge amount of data, instead we can rely on self-supervised techniques and obtain comparable result which is less expensive and less time consuming.

References

Bojanowski, P., & Joulin, A. (2017). Unsupervised learning by predicting noise. In *International conference on machine learning* (pp. 517–526).

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, *23*(11), 1222–1239.

Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *Proceedings of the european conference on computer vision (eccv)* (pp. 132–149).

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, *33*, 9912–9924.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607).

Chen, T., Zhai, X., & Houlsby, N. (2018). Self-supervised gan to counter forgetting. *arXiv preprint arXiv:1810.11598*.

Chen, T., Zhai, X., Ritter, M., Lucic, M., & Houlsby, N. (2019). Self-supervised gans via auxiliary rotation loss. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 12154–12163).

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

Dhanachandra, N., Manglem, K., & Chanu, Y. J. (2015). Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, *54*, 764–771.

Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. *arXiv preprint* *arXiv:1605.09782*.

Epstein, R. (n.d.). *The empty brain.* Retrieved 14.01.2016, from http://www.wpcentral.com/ie9-windows-phone-7-adobe-flash-demos-and-development-videos

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*(2), 303–338.

Faktor, A., & Irani, M. (2014). Video segmentation by non-local consensus voting. In *Bmvc* (Vol. 2, p. 8).

Fernando, B., Bilen, H., Gavves, E., & Gould, S. (2017). Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3636–3645).

Forsyth, D., & Ponce, J. (2011). *Computer vision: A modern approach.* Prentice hall.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., … He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., … others (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, *33*, 21271–21284.

Gutmann, M., & Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 297–304).

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9729–9738).

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

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

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4700–4708).

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., & Makedon, F. (2020). A survey on contrastive self-supervised learning. *Technologies*, *9*(1), 2.

Jiang, H., Larsson, G., Shakhnarovich, M. M. G., & Learned-Miller, E. (2018). Self-supervised relative depth learning for urban scene understanding. In *Proceedings of the european conference on computer vision (eccv)* (pp. 19–35).

Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *43*(11), 4037–4058.

Jing, L., Yang, X., Liu, J., & Tian, Y. (2018). Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv preprint arXiv:1811.11387*.

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, *1*(4), 321–331.

Kim, D., Cho, D., & Kweon, I. S. (2019). Self-supervised video representation learning with space-time cubic puzzles. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 8545–8552).

Korbar, B., Tran, D., & Torresani, L. (2018). Cooperative learning of audio and video models from self-supervised synchronization. *Advances in Neural Information Processing Systems*, *31*.

Ku, H., McKinley, M. D., & Kenney, J. S. (2002). Quantifying memory effects in rf power amplifiers. *IEEE Transactions on microwave theory and techniques*, *50*(12), 2843–2849.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). Hmdb: a large video database for human motion recognition. In *2011 international conference on computer vision* (pp. 2556–2563).

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., … others (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4681–4690).

Lee, H.-Y., Huang, J.-B., Singh, M., & Yang, M.-H. (2017). Unsupervised representation learning by sorting sequences. In *Proceedings of the ieee international conference on computer vision* (pp. 667–676).

Li, W., Wang, L., Li, W., Agustsson, E., & Van Gool, L. (2017). Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., & Tang, J. (2021). Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Lorre, G., Rabarisoa, J., Orcesi, A., Ainouz, S., & Canu, S. (2020). Temporal contrastive pretraining for video action recognition. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 662–670).

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., … Van Der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *Proceedings of the european conference on computer vision (eccv)* (pp. 181–196).

Mahendran, A., Thewlis, J., & Vedaldi, A. (2018). Cross pixel optical-flow similarity for self-supervised learning. In *Asian conference on computer vision* (pp. 99–116).

Minaee, S., & Wang, Y. (2019). An admm approach to masked signal decomposition using subspace representation. *IEEE Transactions on Image Processing*, *28*(7), 3192–3204.

Misra, I., & Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 6707–6717).

Najman, L., & Schmitt, M. (1994). Watershed of a continuous function. *Signal Processing*, *38*(1), 99–112.

Neumann, M., Pinto, A. S., Zhai, X., & Houlsby, N. (2020). Training general representations for remote sensing using in-domain knowledge. In *Igarss 2020-2020 ieee international geoscience and remote sensing symposium* (pp. 6730–6733).

Nock, R., & Nielsen, F. (2004). Statistical region merging. *IEEE Transactions on pattern analysis and* *machine intelligence*, *26*(11), 1452–1458.

Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision* (pp. 69–84).

Oord, A. v. d., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, *9*(1), 62–66.

Owens, A., & Efros, A. A. (2018). Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the european conference on computer vision (eccv)* (pp. 631–648).

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2536–2544).

Plath, N., Toussaint, M., & Nakajima, S. (2009). Multi-class image segmentation using conditional random fields and global classification. In *Proceedings of the 26th annual international conference on machine learning* (pp. 817–824).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

Ren, Z., & Lee, Y. J. (2018). Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 762–771).

Sayed, N., Brattoli, B., & Ommer, B. (2018). Cross and learn: Cross-modal self-supervision. In *German conference on pattern recognition* (pp. 228–243).

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the ieee international conference on computer vision* (pp. 618–626).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Starck, J.-L., Elad, M., & Donoho, D. L. (2005). Image decomposition via the combination of sparse representations and a variational approach. *IEEE transactions on image processing*, *14*(10), 1570–1582.

Stretcu, O., & Leordeanu, M. (2015). Multiple frames matching for object discovery in video. In *Bmvc* (Vol. 1, p. 3).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Tao, L., Wang, X., & Yamasaki, T. (2020). Self-supervised video representation learning using inter-intra contrastive framework. In *Proceedings of the 28th acm international conference on multimedia* (pp. 2193–2201).

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3156–3164).

Wang, W., Lai, Q., Fu, H., Shen, J., Ling, H., & Yang, R. (2021). Salient object detection in the deep learning era: An in-depth survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wu, X., Sahoo, D., & Hoi, S. C. (2020). Recent advances in deep learning for object detection. *Neurocomputing*, *396*, 39–64.

Wu, Z., Xiong, Y., Yu, S. X., & Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3733–3742).

You, Y., Gitman, I., & Ginsburg, B. (2017). Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.

Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision* (pp. 649–666).