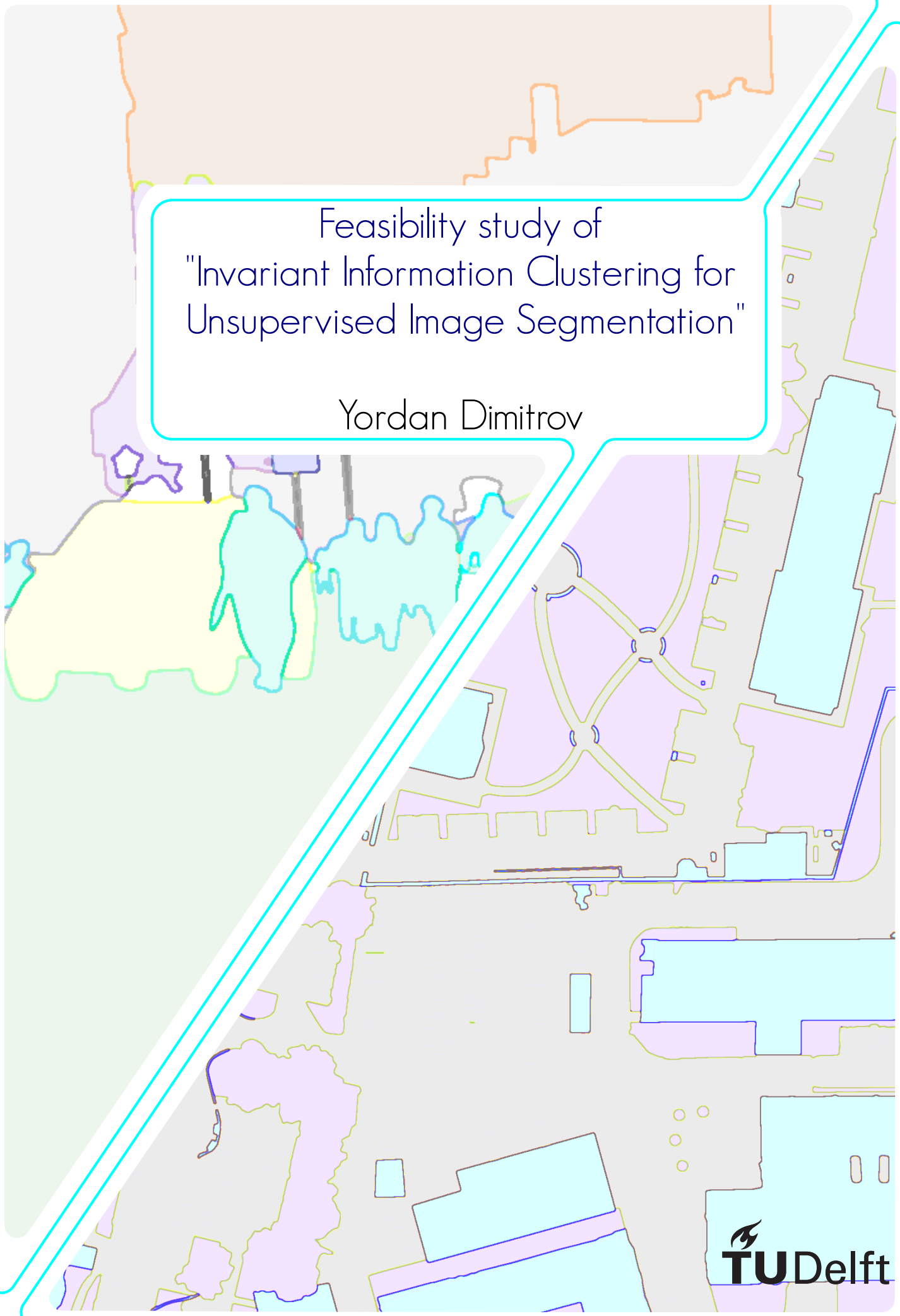
Technische Universiteit Delft

Feasibility study of "Invariant Information Clustering for Unsupervised Image Segmentation"

Yordan Dimitrov



TU Delft

Feasibility study of "Invariant Information Clustering for Unsupervised Image Segmentation"

by

Yordan Dimitrov

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday October 22, 2021 at 12:00 PM.

Student number:	4948874
Project duration:	September 3, 2020 – October 22, 2021
Thesis committee:	Dr. J. C. van Gemert, TU Delft, supervisor
	A. Lengyel, MSc, TU Delft, daily supervisor
	Dr. S. L. Pintea, TU Delft, daily supervisor
	Dr. N. Tömen, TU Delft, daily supervisor
	Dr. K. A. Hildebrandt, TU Delft

TU Delft

Preface

The following report presents my thesis titled *Feasibility study on "Invariant Information Clustering for Unsupervised Image Segmentation"*. The objective of the project was to analyse and evaluate a novel unsupervised segmentation approach which promises unparalleled performance on satellite imagery. The supplementary chapters include background information on Deep Learning, different segmentation approaches and unsupervised representation-learning methods. The thesis has been written to fullfil the requirements to obtain the degree of Master of Science at the Delft University of Technology.

I would like to thank my supervisors Attila Lengyel, Silvia Pintea and Nergis Tömen for their immense help and advice throughout the entire project, this work would not have been as detailed as it is now without their guidance. I am also grateful for all the assistance and advice of my friends and fellow students that helped me whenever I was confronted with a problem.

I am also extremely thankful for my time spent in Delft and for all the friendships that I have made along the way. And last by not least, I would like to express my utmost gratitude towards my parents whose support has made all of this possible.

Yordan Dimitrov
Delft, October 2021

Contents

1

Scientific paper

Feasibility study on "Invariant Information Clustering for Unsupervised Image Segmentation"

Yordan Dimitrov[1], Attila Lengyel[2], Silvia Pintea[2], and Nergis Tömen[2]

[2]Computer Vision Lab, Delft University of Technology
[1] *y.d.dimitrov@student.tudelft.nl*
[2]{*a.lengyel, s.l.pintea, n.tomen*}*@tudelft.nl*

Abstract

In this paper we analyse the performance of a novel clustering objective that optimises a neural network to predict segmentation. We challenge the reported results by replicating the original experiments and conducting additional tests to gain an insight into the algorithm. We analysed the efficiency of the clustering objective on a different architecture, dataset and hyper-parameters. To our surprise the algorithm demonstrated considerably lower results when running on the new setup. Further, in our work we detail the reasons behind the discrepancy and provide configurations under which the method performs best. We show that the objective is highly sensitive to the type of images it is predicting and the complexity of the architecture that is being used with.

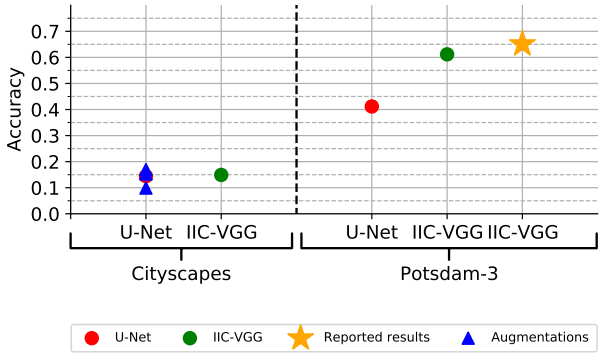


Figure 1. Comparison of the reported results in [18] and the results we obtained with the same setup. We conducted additional experiments which demonstrate the ineffectiveness of the proposed approach in different setups. Each triangle marker denotes an experiment where a single augmentation was modified.

1. Introduction

Convolutional networks have become the target of rapid development over the last decade with the success of the first deep learning models [25, 42]. The availability of large amounts of data and advancement in the technological domain have further accelerated research. Image segmentation as a sub-field of deep learning has gained substantial attention after the introduction of fully convolutional networks [29]. This type of architecture does not use a series of fully-connected layers to generate predictions and instead employs only banks of convolutional, activation and pooling layers. To train such networks at a sufficient level, vast amounts of annotated data are required. Despite the considerable increase of available training datasets, generation of semantic labels is still the single major bottleneck to the performance of deep learning models.

To address this issue many works have proposed solutions which seek to minimise the dependence on labeled data or eliminate it entirely [19, 49, 23]. Many approaches rely on semi-supervised techniques which reduce the use of annotations by utilising synthetic data or joint-architectures [20, 50]. There are however methods which learn CNNs entirely without ground truth supervision [30, 32].

In our study, we analyse one such work which performs segmentation based on contrastive learning and statistical information of the input. The paper by Ji *et al*. [18], which we abbreviate as IIC, maximises the mutual information between 2 input images as an optimisation objective. The approach does not use any labels for training and relies purely on the correlation between images. Typical of the field of contrastive learning [3], IIC generates its second image by applying a random transformation to the first image. Next, a given CNN learns the feature representation between both inputs. Further details of the method can be found in section 3.1.

The results reported in the paper suggest that the method is capable of predicting segmentation with relatively high accuracy for a self-supervised method. On this basis we investigate the documented results and attempt to replicate them on a different setup.

To this end, we propose a set of experiments that aim to explain the performance of IIC when used in a different setup. We conducted tests on a different dataset, architecture, learning rate and image augmentations, as shown in Figure 1. Extensive analysis suggests that the approach is highly sensitive to the choice of inputs and suffers when used with non-uniform datasets. When applied to data which exhibits a wide variety of class instances, IIC performs worse. Additionally, we show that due to design features, deep networks can satisfy the loss function without producing accurate predictions. Empirical results suggest that the learning rate and augmentations have an impact on the method, although not as significant as that of the dataset.

2. Related Work

Semantic segmentation is the process of assigning a class label to each pixel in a given image. In this section we firstly present works on fully- or semi- supervised segmentation. Then some methods which address self-supervised techniques will be introduced.

Fully-supervised techniques Semantic segmentation has seen considerable improvements with the introduction of fully convolutional neural networks which disposed of the fully-connected layers and replaced them with upscaling ones [29]. In later years significant progress was made with supervised approaches which relied on encoder-decoder architectures and skip connections [1, 41, 36]. These consist of a contracting and expansive path, where the first capture features and reduce spatial information and the latter use the feature maps as input to upscale them through deconvolution. He *et al*. [15] achieved further development by merging classification and segmentation models together in their Mask R-CNN. Many works incorporate pyramid pooling methods to extract features at different scales [27, 56, 40]. This allows to achieve better segmentation by using local features along with global information. Several works have explored the feasibility of dilated convolutions [28, 6, 5]. This approach allows for expanding the receptive field of kernels without losing resolution. Huang *et al*. [16] experimented with attention modules which guide the model into learning context specific features of an image. Multiple attention modules and global attention networks have proven effective in single object predictions [12, 57, 43, 24]. DeepLab and other architectures [5, 34] have achieved impressive results with Conditional Random Field (CRF) in the domain of semantic segmentation. CRF is used as a post-processing step to smooth object-based segmentation predictions. Attempts have been made to combine segmentation architectures along with adversarial models organised in a generator-discriminator arrangement [31, 7]. Tu *et al*. [45] have taken a bottom-up approach at the task of scene parsing. They propose an idea to divide the input image in separate regions using a dedicated superpixel algorithm. After that regular CNN feature maps and a custom loss function are used to optimise the network. Couprie *et al*. [10] have addressed the problem in a similar fashion by processing RGB-D images through a Laplacian pyramid. The Laplacian outputs along with superpixels of the original image are fed through a CNN. Multitask architectures have also been studied as potential improvements in the domain. Mousavian *et al*. [34] have proposed a solution that predicts depth and segmentation together. PAD-Net by Xu *et al*. [50] combines depth, contour and surface normals prediction. The described methods rely on a common backbone architecture that extracts essential features and then generates several intermediate auxiliary outputs to aid scene parsing. Recent advancements have been made in recurrent training techniques mainly used in video or frame sequences [24, 47]. They utilise predictions of previous images together with labels to improve predictions of succeeding inputs.

All approaches described here utilise ground truth as the sole supervisory signal to optimise the employed architecture. In contrast, the method which we are analysing, can work without the need of labels.

Semi-supervised techniques These methods rely mostly on self-supervision and at certain training intervals make use of labels. Other techniques may produce several intermediate outputs where one or more are fully-supervised while the others remain unsupervised or self-supervised. Jin *et al*. [20] address the task by training a CNN to construct future frames and labels of a video stream, given the current frame. This is done in a GAN-like architecture. The extracted spatio-temporal information is reused along with generated and existing labels to parse frames. A similar approaches is used by Zhu *et al*. [59]. They expand an existing annotated database of samples by reconstructing video frames and training a CNN on the dataset. Memory gates and attention modules also find an application in the semi-supervised setting. Xie *et al*. [52] have employed a combination of depth-aware attention modules organised in a recurrent structure to generate object segmentation. Depth information is already known. The medical domain is another field that makes extensive use of sparsely annotated data. A number of studies [17, 58, 4] explore the potential of contrastive learning in segmentation models to address the issue of scarce training data. The use of multi-modal architectures extends in the semi-supervised domain as well.

Many approaches employ models of different types together to improve segmentation [44, 19]. These architectures comprise of depth, optical flow, camera location and motion-mask predicting models. Their outputs are channeled together to produce a per-pixel labelling of the input. The aggregation allows for training with fewer samples. Many of the proposed solutions employed depth as an auxiliary method to segmentation. However, several works have introduced methods which utilise depth as a main supervisory signal for this task [11, 48]. These approaches begin by grouping depth pixels either with a dedicated clustering algorithm such as *k*-means or acquiring a kernel probability membership for a specific cluster. After that the obtained clusters are processed to produce the individual objects. Although, these approaches are not as precise as previously discussed methods they do not require any labels.

The presented methods learn prior information on the dataset either by reconstructing video frames, learning salient object features or predicting auxiliary outputs. Similarly, IIC builds upon this and learns the statistical occurrences of class instances from the dataset to optimise a CNN model.

Self-supervised techniques Self-supervision in deep learning pertains to the kind of methods that generate supervisory signal from data itself [51]. Most of these methods create pretext tasks which allow the network to learn useful representations while solving them. Depending on the type of pretext task they can be split into in-painting a missing part of an image [53, 39], colourisation of grey scale images [55, 26], clustering [3, 18], adversarial methods [7, 14, 2] and contrastive learning approaches [18, 8, 33]. There are also tasks such solving jigsaw puzzles [37, 38] and video motion segmentation [54]. These techniques for general learning of input representations have been transferred to the scene parsing and object segmentation domains. Lu *et al*. [30] propose a multi-attention module that employs contrastive learning techniques between video frames to segment objects. A similar approach is adopted in [23]. Another study by Mahendran *et al*. [32] analyses the possibility of employing the in-painting method for optical flow prediction as a pretext task for segmentation. Several works discuss the potential of learning an image prior based on few input images [13, 46], which can be used to denoise and segment images into foreground/background layers. The method described in [49] concatenates two U-Nets [41], which use an N-cut and a reconstruction loss to obtain a segmentation and optimise the model. Kim *et al*. [22] exploit an embedding module to produce feature vectors. These are then normalised and clustered to produce object segmentation.

Unlike the described methods here, IIC does require any pretext tasks to prepare the network, nor does it need post-processing algorithms to obtain useful scene segmentations.

3. Method

3.1. IIC background

IIC is a clustering based approach that learns invariant image representations via contrastive learning. To learn useful feature representations, contrastive approaches rely on augmentation techniques of the input. In the context of IIC, the objective function maximises the mutual information between the feature embeddings of an input image $\mathbf{x}$ and an augmented image $\mathbf{x}'$, Eq. 2. If $\Phi(\cdot)$ is a convolutional network with input $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, the output $\Phi(\mathbf{x}) \in [0,1]^{C\times H\times W}$, generated by a softmax layer, can be regarded as a probability distribution of a random variable z over C classes $P(z=c|\mathbf{x})$ [18]. Considering an augmented image $\mathbf{x}'$, we can construct a $C \times C$ joint probability matrix $\mathbf{P}$ of the outputs $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}'_i)$, as shown in Eq. 1. More specifically, an element at row c and column c' represents $\mathbf{P}_{cc'} = P(z=c, z'=c')$, which is the joint probability of $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}'_i)$ for a given class C of the output tensors.

$$\mathbf{P} = \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}'_i)^{\mathrm{T}} \tag{1}$$

Matrix $\mathbf{P}$ is the input to the objective function, shown in Eq. 2. More concretely, $\mathbf{P}_{cc'}$ is the joint probability computed in Eq. 1. $\mathbf{P}_c$ and $\mathbf{P}_{c'}$ are the marginal probabilities obtained from $\mathbf{P}$. They represent the independent probability of class C as predicted from $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}'_i)$. When computing the mutual information, to avoid neutralising $\mathbf{P}_{cc'}$ and the product of $\mathbf{P}_c$ and $\mathbf{P}_{c'}$, $\mathbf{P}$ is symmetrised along the diagonal using $(\mathbf{P}+\mathbf{P}^{\top})/2$.

$$I(z,z') = I(\mathbf{P}) = \sum_{c=1}^{C}\sum_{c'=1}^{C} \mathbf{P}_{cc'} \cdot \ln \frac{\mathbf{P}_{cc'}}{\mathbf{P}_c \cdot \mathbf{P}_{c'}} \tag{2}$$

In terms of image segmentation, IIC operates on patches u obtained from the input image. Each patch is transformed via a set of geometric and photometric perturbations g, which include colour transformations, affine transformations and flipping. Also, optional perturbations may include translation transformations $t \in T$. Elaborating on Eq. 1, to construct matrix $\mathbf{P}$ the output of the transformed patch $\Phi(g\mathbf{x}_i)$ is inverted back to the original spatial frame by the reverse transformation g^{-1}, as shown in Eq. 3.

$$\mathbf{P}_t = \frac{1}{n|G||\Omega|}\sum_{i=1}^{n}\sum_{g\in G}\overbrace{\sum_{u\in\Omega}\Phi_u(\mathbf{x}_i)\cdot[g^{-1}\Phi(g\mathbf{x}_i)]^{\mathrm{T}}_{u+t}}^{\text{Convolution}} \tag{3}$$

Only the geometric transformations are negated by g^{-1}, while the photometric transformations such as colour augmentations remain. Therefore the goal is to maximise the

information between $\Phi_u(\mathbf{x}_i)$ and $[g^{-1}\Phi(g\mathbf{x}_i)]_{u+t}$ over all images $i = 1, ..., n$, patches $u \in \Omega$ and perturbations $g \in G$.

In order to handle distractor classes in certain datasets or noisy data, the authors employ an additional output 1x1 convolutional layer (head), along the main one. It is trained in the same way as the main layer and uses the same loss function. The key difference is that the main one predicts the same number of classes as the ground truth, while the auxiliary one predicts a larger number of classes. Its purpose is to enhance the learned representation between class instances.

Finally, the authors introduce 2 $\boldsymbol{\lambda}$ constants which weigh the importance of the product of the marginals $\mathbf{P}_c$ and $\mathbf{P}_{c'}$ with respect to the their joint probability $\mathbf{P}_{cc'} = P(z = c, z' = c')$.

3.2. Variation of IIC

In this section we propose our experimental setup whose aim is to examine the poor generalisation of the IIC method in other environments. We demonstrate this through 4 sets of experiments, shown in Table 1. Their objective is to test the method on a variety of different settings, which comprise: architecture, dataset, learning rate and augmentation influence.

3.2.1 Dataset

The main dataset which is used to report performance is Potsdam. In its original format it consists of 6 classes: road, car, building, clutter, vegetation, tree. Consequently, the authors generate Potsdam-3 by merging each of the 3 pairs. Since the majority of quantitative results in the paper are produced from this dataset we decided to base our experiments on it. To determine the importance of a dataset empirically, we test on Cityscapes using an identical hyper-parameter setup. The motivation behind this arises from the characteristics of Potsdam-3. It has little variation in class shapes - mostly rectangular regions with similar distribution and dynamic across all input images. On the other hand, Cityscapes offers 19 classes which may have various shapes, sizes, colours and locations across the inputs.

3.2.2 Architecture

Our assumption is that the architecture proposed by Ji *et al.* has substantial influence on the reported results in [18]. For this reason, we benchmark the results of IIC-VGG against the well-known segmentation network U-Net, shown in Figure 2. In the version that we experimented with, U-Net has close to 13.4M parameters, whereas IIC-VGG has 4.5M parameters. U-Net features a contracting and an expanding path and at the bottleneck feature maps measure at $\frac{1}{16}$ of the original resolution. This allows U-Net to capture the high-level semantics of the image. Furthermore, the model implements skip-connections which recover low-level spatial information from initial layers. In contrast, IIC-VGG contains a single max pooling layer, which extracts features at just half the original size, as shown in Figure 2. We believe that the higher complexity of U-Net allows it to exploit boundary effects, as suggested by Kayhan *et al.* [21]. These effects occur when padding is applied to a feature map, which in turn allows a CNN to learn absolute locations of objects. This is a factor which has a detrimental effect on the segmentation process in our use case. As a consequence a given network can disregard input and predict random patterns across the input image which satisfy the loss function.

3.2.3 Learning rate

The fully-unsupervised experiments conducted on Potsdam-3 featured a learning rate of 1×10^{-5}, and 1×10^{-6} for Potsdam-6, both of which are below the standard value of 1×10^{-4}. Additionally, training did not include learning rate decay or growth. Also while testing, we observed that the setup is sensitive to modifications of the hyper-parameter. The low and unaltered value led us to speculate that barely any weight optimisation is done during training. For this reason we performed a series of experiments with 3 learning rate magnitudes - 1×10^{-4}, 1×10^{-5} and 1×10^{-6}. To establish a benchmark comparison the same 3 tests were conducted on U-Net.

3.2.4 Image augmentations

In the original training procedure the input images are modified by applying colour transformations and horizontal flipping. Optional modifications include Sobel filtering and affine transformations such as scaling, skewing or rotation. Augmentation of inputs is a standard technique in contrastive learning which allows the network to learn high-level representations from input [3]. To determine their influence on segmentation quality we trained U-Net with different augmentations enabled. We investigated the effect of colour transformations by modifying brightness, contrast, saturation and hue at a factor of 0.1, 0.2, 0.4 or none. Each of these factors defines the allowed margin of transformation for the given colour characteristic. Further, we tested with sobel filtering and flipping disabled. Full experimental details can be found in Table 5.

4. Experimental results

In this section we provide details on the datasets that were used to train the models as well as the evaluation metrics. Further, we describe how training is conducted and afterwards we present empirical results on the discrepancy

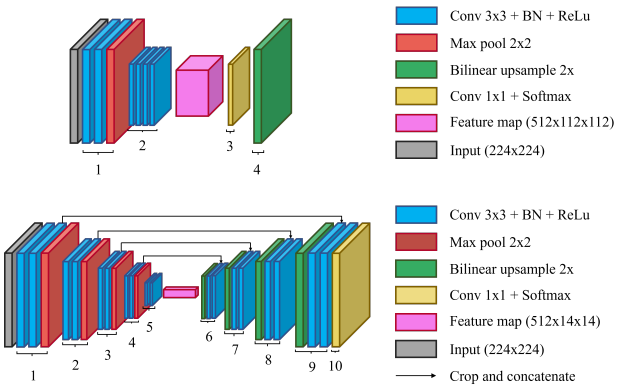


Figure 2. **Architectures** Top: IIC-VGG by Ji *et al*. [18]. The network is a simplified modification of VGG-16. It incorporates no skip connections. Bottom: U-Net by Ronneberger *et al*. [41]. A symmetric encoder-decoder architecture which implements gradual upscaling of feature maps and skip connections to improve low-level detail retrieval.

Architecture	**Test target**
IIC-VGG [18]	Datasets
U-Net [41]	Architecture
IIC-VGG	Learning rate
U-Net	Learning rate
U-Net	Augmentations

Table 1. Groups of experiments. These were designed with the purpose of identifying key factors which are responsible for the lower effectiveness of the objective function in different settings.

of the reported results in [18] and our findings.

4.1. Datasets

Cityscapes is large database of urban street scenes. The dataset offers semantic, instance-wise and depth annotations of over 5000 images. Additionally, it contains approximately 20000 coarsely annotated images. The number of classes are 30 and are divided into 8 distinctive categories. The images were taken in 30 cities and capture a variety of weather conditions and times of the day. The content was originally recorded as video, however specific frames were handpicked which contain dynamic objects and environments.

ISPRS Potsdam is a dataset comprising of 38 high-resolution satellite images of the city of Potsdam, Germany. Each of the images has a resolution of 6000x6000 pixels. The dataset provides semantic annotations grouped in 6 categories. Labels are available for only a part of the data as the rest is used by the authors for test submission by other researchers. The RGB images are also equipped with an infrared channel. The aerial footage is provided by the ISPRS benchmark. [35].

4.2. Evaluation Metrics

In the paper [18] performance of the main approach and baseline methods are measured in *Mean Pixel Accuracy*. Naturally, accuracy is biased when input classes are imbalanced in training datasets. While this characteristic is not present in Potsdam, it is highly pronounced in Cityscapes. A large portion of all images is occupied by the classes *sky* and *road*, albeit at different weather conditions, whereas only some contain the class *person*. To address this issue we also employed Intersection over Union (*IoU*), also known as Jaccard index, to benchmark the setup. The metric overcomes the issue of class imbalance by taking into account the areas of overlap and union of predicted and ground truth classes. The formula of *IoU* is shown in Eq. 4. We reported *IoU* for every class at each validation epoch and then calculated an aggregated average value.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|} \tag{4}$$

In the interest of producing unbiased results we also evaluated the predictions with class normalised accuracy. Let n_i be the total number of pixels in the image, t_i be the total number of pixels per class and c_i - number of correctly predicted pixels per class. Instead of calculating c_i/n_i, we report c_i/t_i. This modification mitigates the effect of imbalanced classes when measuring performance.

4.3. Implementation Details

Architecture The models shown in Figure 2 were used in training. To follow the original approach as closely as possible we implemented the auxiliary overclustering head, as described in [18]. We denote the number of ground truth channels as k_{gt} and the output channels of the auxiliary head as k. In training we set $k = 3 \times k_{gt}$, as suggested by the authors. No further modifications were done to the models.

Training When calculating the joint probability matrix of the IIC method we kept the padding value of 10, as seen in [18]. Additionally, the $\boldsymbol{\lambda}$ entropy coefficients were also set to their default values of 1.0 and 1.5 for the overclustering head and regular head, respectively. We trained all experiments for 35 epochs with a batch size of 40. For the learning rate we used a value of 1×10^{-4}, except for the experiments described in section 3.2.3. When examining the impact of augmentations we employed the setup presented in Table 5, while for the remaining experiments we used entry 5 from

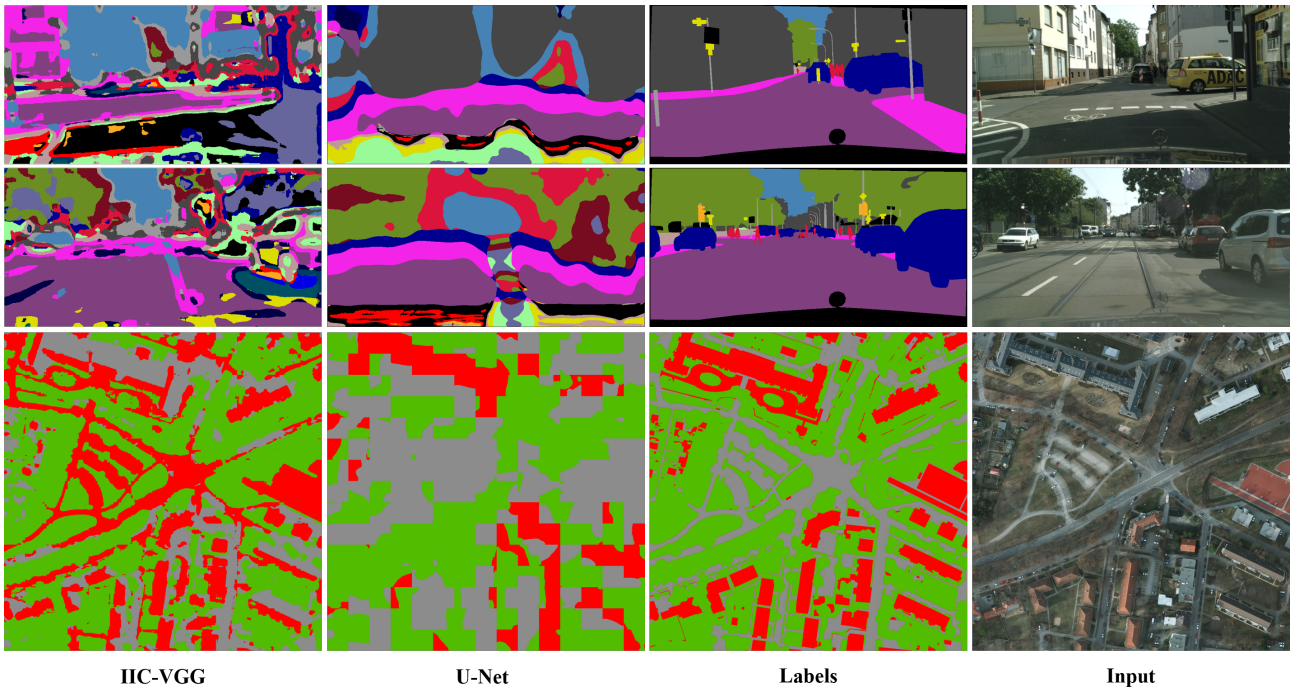


Figure 3. **Example segmentation results (fully self-supervised).** First column: IIC-VGG results, Second column: U-Net results, Third column: Cityscapes and Potsdam-3 labels, Fourth column: Input. As it can be seen, the segmentations by U-Net vaguely resemble the input. Comparatively, IIC-VGG performs well on Potsdam-3, while on Cityscapes it focuses mainly on colour cues.

the same list. Regarding dataset preparation, the high resolution images from Potsdam-3 were scaled down by half and cropped into smaller 200x200 patches. For training we used 7200 samples, while testing and validation each had 675. To gain a better perspective on the parsed outputs from Potsdam-3, we merged together the patch predictions from testing and validation, into the initial 3000x3000 inputs. Likewise, the training inputs from Cityscapes were scaled down by half and cropped to 224x224. All experiments were conducted on an NVIDIA Tesla V100-32GB. The duration of training was approximately 120h.

4.4. Results

4.4.1 Dataset and architecture

Results We investigated the performance of IIC on a different architecture and dataset. The results of the experiments are gathered in Table 2. Below these we present a baseline comparison of the same set of tests conducted on untrained models of each architecture. The applied augmentation setup is referenced from Table 5, setup 6. The results which we produced with IIC-VGG on Potsdam are similar with those reported by Ji *et al*. [18] for the same experiment. However, when IIC-VGG is substituted with U-Net it is evident that performance drops almost by 20%. Furthermore, when the same experimental setup is applied on Cityscapes a drastic decrease in accuracy is registered by both networks. In terms of *mIoU*, U-Net has an advantage of 2% against IIC-VGG, while the opposite can be noted when comparing normalised accuracy, where IIC-VGG prevails. Referring to the untrained models we can conclude that IIC does optimise the networks to some extent, although the results gathered from Cityscapes are close to the baseline.

Analysis We suggest that the poor performance of both networks on Cityscapes is due to the diversity of the dataset. At every batch IIC constructs a probability matrix of the pixel memberships of every class which means that the method benefits from a bigger batch size. Since the majority of Potsdam images follow a similar structure it is easy to capture the distribution of colours and shapes across the dataset. On the other hand, Cityscapes scenes vary greatly and even identical classes appear different. As a result the method fails to generalise between instances of the same class, e.g., a green and a red car observed from a different viewpoint. We infer that IIC as a method which relies on the frequency at which instances appear, expects a dataset with a uniform class prior. This implies that the objective is good at predicting simple, well represented, visually consistent classes such as sky, road and vegetation, or in general those that have little instance variation.

We attribute the loss of performance when using U-Net to the fact that deep convolutional neural networks are capable of exploiting absolute spatial location of objects within an image [21]. Modern CNNs achieve this by learning filters which are highly sensitive to specific locations using boundary effects. Since the receptive field increases with each successive filter this flaw can be exploited even away from the borders of an image. An example is demonstrated in Figure 4. The images were produced by passing a completely white image through each of the models. Nonetheless, both networks still make "predictions" on the blank input. The patterns that we can observe are actually padding artefacts that have been detected by the models when passing through the convolutional layers. The predictions of the shallow IIC-VGG are located mainly near the borders of the image since the architecture comprises of 7 convolutional layers. Additionally, the model features a single upscaling layer. Conversely, the predictions that U-Net made can reach significantly closer to the centre of the image. This is caused by the higher number of convolutional layers - 19 and up-sampling layers - 4, that are part of the architecture. As shown in the Figure 3, U-Net can largely disregard the input image and still produce a prediction which contains all desired classes. Evidence suggest that a translationally equivariant network should not be able to generate such patterns, so we believe that the boundary cues are exploited to satisfy the loss function [21].

We conclude that the choice of dataset is among the most important factors behind the performance of the IIC approach. The results demonstrate that datasets with a uniform distribution and similar class instances are optimal. Second to that we place the design of the employed architecture. Performance on U-Net suggests that deep networks can manipulate the flaws of the objective function to produce low grade predictions.

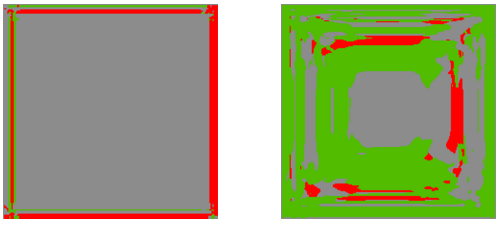


Figure 4. **Boundary effect** Left: IIC-VGG boundary effect. Right: U-Net boundary effect. It is evident that U-Net is much more likely to take advantage of the boundary effect.

4.4.2 Learning rate

Results The learning rate (LR) experiments were conducted on both architectures using Cityscapes with identical augmentation setups, Table 4. The trend seen in Table 2, where results on Cityscapes are significantly lower, can be observed here, as well. All *mIoU* results range between 4% and 6%, with the exception of U-Net, at its highest learning rate, which peaks at 8.9%. Also, at lower learning rates the performance of U-Net and IIC-VGG is comparable to that of the untrained models from Table 2.

Model	Dataset	Learning rate	Augm. setup	mIoU (%)	Normalised accuracy (%)
IIC-VGG	Potsdam	1×10^{-4}	6	45.84	61.16
U-Net	Potsdam	1×10^{-4}	6	26.59	41.17
IIC-VGG	Cityscapes	1×10^{-4}	6	6.40	14.91
U-Net	Cityscapes	1×10^{-4}	6	8.86	14.44
Untrained models					
IIC-VGG	Potsdam	–	–	15.39	33.44
U-Net	Potsdam	–	–	17.10	32.81
IIC-VGG	Cityscapes	–	–	3.97	9.38
U-Net	Cityscapes	–	–	5.15	8.13

Table 2. We trained the U-Net and IIC-VGG under the same conditions on two different datasets and obtained significantly lower results on the second one. The difference in performance on Cityscapes is 3 to 7 times lower than that on Potsdam.

	Average training loss of last epoch			
	U-Net		**IIC-VGG**	
Learning rate	Head **A**	Head **B**	Head **A**	Head **B**
1×10^{-4}	-1.90	-4.84	-1.54	-4.68
1×10^{-5}	-0.59	-3.30	-1.45	-4.57
1×10^{-6}	-0.18	-3.08	-1.07	-4.25

Table 3. Average training loss of the last epochs of each of the learning rate experiments. Each model has 2 output layers marked as **A** and **B**. Head **A** is the overclustering output layer. As the learning rate decreases the average loss at which the models converge also follows the trend.

Analysis As suggested by the findings, the networks seem to benefit from a higher learning rate (LR). When training at low LR values, however, the models converge to degenerate solutions. When we examine the output at different learning rates in Figure 5, we can see that the segmentations at the lowest learning rate focus more on colour cues and edges, while the predictions at the highest LR are smoother, although arguably not more semantically aware. Look-

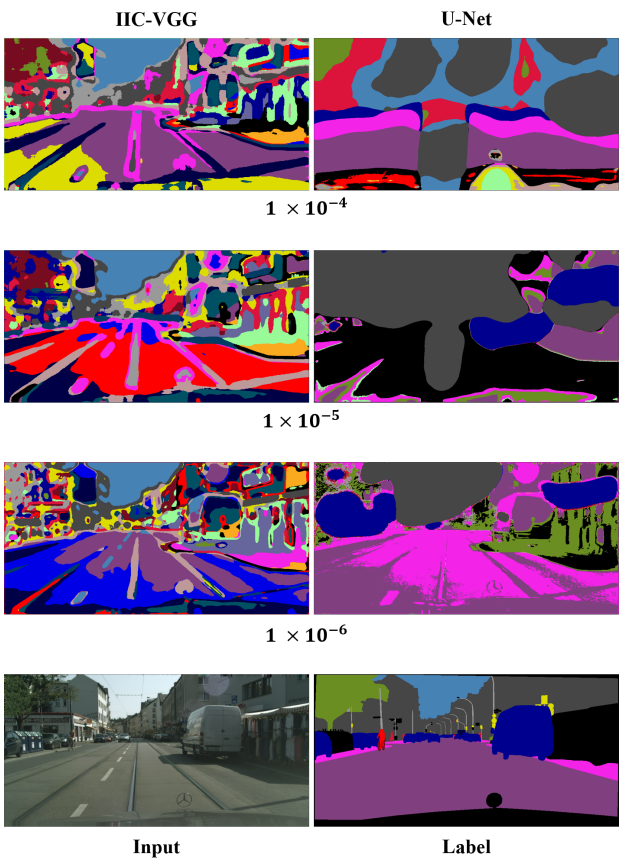


Figure 5. The results obtained at lower learning rates suggest that both models focus on low-level features such as colours and edges. With a learning rate of 1×10^{-6}, we can see that U-Net erroneously segments the road in 2 classes along a light gradient. Conversely, when the learning rate increases the models converge to solutions which can identify higher level features.

ing at the results of IIC-VGG, we can still observe over-fragmentation of the classes at the highest LR, although not as distinctly as with the lower LR. In spite of the slight improvement at LR of 1×10^{-4}, the overall performance of either network is exceptionally low and does not differ greatly from an untrained network. Table 3 shows the average training loss of the learning rate experiments of their last epoch. Naturally, we can observe that the models do not converge to the same level with a lower learning rate, which can be an explanation for the emphasis on low-level features. The higher *mIoU* and *Accuracy* obtained from both untrained models on Potsdam implies that the significance of the LR value comes second to that of the dataset. This further supports the hypothesis that the choice of dataset is crucial to the effectiveness of the approach.

The findings indicate that a higher learning rate does have a positive impact on the approach as it guides the models to focus on higher-level features. Despite this, its significance is negligible in comparison with the dataset, as both models exhibit poor performance regardless of the learning rate.

Model	Dataset	Learning rate	Augm. setup	mIoU (%)	Normalised accuracy (%)
IIC-VGG	Cityscapes	1×10^{-4}	6	6.40	14.91
IIC-VGG	Cityscapes	1×10^{-5}	6	5.46	13.96
IIC-VGG	Cityscapes	1×10^{-6}	6	5.44	13.53
U-Net	Cityscapes	1×10^{-4}	6	8.86	14.44
U-Net	Cityscapes	1×10^{-5}	6	4.85	9.92
U-Net	Cityscapes	1×10^{-6}	6	5.49	9.38

Table 4. Results of experiments on the learning rate scale. Despite the better results at the highest learning rate, both networks performed poorly on Cityscapes.

4.4.3 Image augmentations

Results The results of this analysis are shown in Table 5. The experiments were split in 2 categories: evaluating influence of a Sobel operator and geometric transformations; and evaluating colour modifications (hue, saturation, brightness and contrast). We established a baseline experiment that did not include any augmentation techniques. It is denoted by $\times$ under every perturbation in the aforementioned table. As expected, it demonstrates a remarkably poor result of 3.45% which is even below the score obtained from an untrained model. The best performance was achieved when colour transformation was the single applied augmentation at a factor of 0.1, scoring 9.27%. Similarly, we tested with both geometric transformations enabled, to determine if combined together they enhance contrastive learning. The experiment did not manage to surpass the performance of only colour transformation and produced an *mIoU* of 8.74%. Additionally, we set up 2 experiments where the colour transformation was tested along with *flip* and *affine transformation* separately. Each of the tests registered a slight decrease in *mIoU* compared to using colour transformation by itself. Lastly, we tested with a pre-processing step which applied a Sobel operator to both input images. The purpose of the experiment was to examine if edge detection helps to uniformly segment objects. Again, the results of this test did not display any substantial improvement over previous augmentation techniques.

In addition, we also analysed the impact of the colour transformation scale on the ability of the network to recognise different class instances. The results are presented in Table 5, below the dashed line. We tested at three different scales. Our analysis show an inverse correlation between the colour scale and performance. The *mIoU* decreases as

N°	Model	Dataset	Learning rate	Sobel	Colour transformation	Horizontal flip	Affine transformation	mIoU (%)	Normalised accuracy(%)	Augmentation invariance (%)
1	U-Net	Cityscapes	1×10^{-4}	✗	0.1	✓	✓	8.74	16.97	82.31
2	U-Net	Cityscapes	1×10^{-4}	✗	0.1	✗	✓	7.64	13.62	78.96
3	U-Net	Cityscapes	1×10^{-4}	✗	✗	✗	✗	3.45	8.58	–
4	U-Net	Cityscapes	1×10^{-4}	✗	0.1	✗	✗	9.27	15.45	93.08
5	U-Net	Cityscapes	1×10^{-4}	✓	0.1	✓	✗	8.22	15.29	93.12
6	U-Net	Cityscapes	1×10^{-4}	✗	0.1	✓	✗	8.86	14.44	91.95
7	U-Net	Cityscapes	1×10^{-4}	✗	0.2	✓	✗	8.17	13.31	89.23
8	U-Net	Cityscapes	1×10^{-4}	✗	0.4	✓	✗	4.65	9.88	88.57

Table 5. Results of augmentation experiments. The highest score was achieved when only colour transformations were used in the augmentation setup. The results suggest that although flipping and affine transformations contribute to contrastive learning they are not as effective as colour augmentation.

the colour transformation becomes more pronounced.

Analysis The conducted experiments suggest that geometric transformations have an adverse effect on the contrastive learning process. When *flip* or *affine transformation* are included the results are consistently lower, albeit better than no augmentations. We can observe that *flip* by itself performs better than *affine transformation* by itself. When combined together, the results improve slightly, but are still below the expected. We hypothesise that once an image is rotated or flipped the network struggles to recognise that an object and its flipped version are the same entity. As a result, the joint probability matrix $\mathbf{P}$ from Eq. 1 is not calculated accurately which in turn hampers the ability to optimise the network optimally. In terms of the colour transformation scale, we made a similar conclusion. With the increase of colour augmentation, the network looses the ability to label pixels of different colours identically, despite the fact that they are technically the same object. To analyse these assumptions, we calculated an augmentation invariance score for each experiment in Table 5. The metric reflects the accuracy between the predictions of an unaltered input image and its augmented variant. The higher the value is, the more robust the model is to the augmentations. The obtained results, however, suggest that the network is in fact invariant to transformations and this is not the root cause of the discrepancy in the results.

Based on these findings, we claim that the IIC approach relies heavily on colour cues to extract mutual information between two samples. Ultimately, the use of geometric perturbations is effective, although not as nearly as colour transformations.

5. Discussion

The goal of our analysis was to determine the factors that support or hinder the performance of the IIC approach. Based on our findings, we outline a number of aspects which could have an impact on the method.

Batch and Image size Fundamentally, IIC attempts to estimate a class distribution of the comprising classes at each batch. While training, the batch size that we employed in all of our experiments was set to 40. Taking into account the nature of the algorithm, we believe that in certain use cases IIC would benefit from a batch size that is several orders of magnitude larger. Since Cityscapes offers a wide variety of road scenes, an input that includes all potential class instances and class variations would be optimal. Furthermore, due to hardware limitations the networks are trained on cropped patches which contain a small portion of the original image. Considering that they are handled in a randomised manner, this makes the network prone to lose contextual information of the input. As a result, these factors contribute to worse performance on diverse datasets. Acknowledging these findings, we believe that IIC would be useful for applications and research which have access to large computational power.

Lambda constant In terms of our experimental setup, we set the $\boldsymbol{\lambda}$ values as constants, as described in section 3.1. We reflect that further analysis could be done to fine-tune the values when different datasets are being used. By default, when using the auxiliary overclustering head, section 4.3, $\boldsymbol{\lambda}$ equals 1, while when the regular head is used, $\boldsymbol{\lambda}$ is 1.5. When predicting with the correct number of output classes, a larger $\boldsymbol{\lambda}$ value decreases the mutual information. A larger denominator value in

$$\frac{\mathbf{P}_{cc'}}{\lambda\mathbf{P}_{c} \cdot \lambda\mathbf{P}_{c'}}$$

reduces the overall value of the fraction. As a consequence, distinct objects share considerably less similarity

and boundaries are more discernible. Therefore datasets which contain more classes could benefit from an adjusted $\boldsymbol{\lambda}$ value.

To support our findings we reference the work of Cho *et al*. [9] who have also experimented with IIC and published their results on Cityscapes. According to their reports, they achieve an *mIoU* of 6.35% on the dataset, which largely coincides with our results.

6. Conclusion

In this work we analysed the Invariant Information Clustering method proposed by Ji *et al*. [18]. Its purpose is to segment images by optimising mutual information between image pairs. We present an experimental setup which evaluates the influence of individual components of the algorithm on the quality of segmentation. The purpose of our tests is to analyse the choice of architecture, choice of dataset, magnitude of the learning rate and importance of different augmentations. Our findings suggest that IIC performs better when used with shallow architectures which implement fewer convolutional layers. In terms of input data, we report that datasets which comprise of large, uniform class instances are optimal for this use case. Furthermore, we indicate that a larger learning rate influences the learning process favourably, when presented with a suitable dataset. Finally, among image transformations the most effective technique to generate image pairs is colour transformation.

Based on our observations we claim that the method constructs a statistical prior solely on colours in order to optimise the mutual information. In contrast, object features such as shape, scale and pose are mostly ignored. As a result, the performance of a given CNN will suffer when the dataset is not tailored to the IIC loss.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. 2

[2] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. 3

[3] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 1, 3, 4

[4] K. Chaitanya, E. Erdil, N. Karani, and E. Konukoglu. Contrastive learning of global and local features for medical image segmentation with limited annotations. *arXiv preprint arXiv:2006.10511*, 2020. 2

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 2

[6] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2

[7] M. Chen, T. Artières, and L. Denoyer. Unsupervised object segmentation by redrawing. *arXiv preprint arXiv:1905.13539*, 2019. 2, 3

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 3

[9] J. H. Cho, U. Mall, K. Bala, and B. Hariharan. Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16794–16804, 2021. 10

[10] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013. 2

[11] C. Dal Mutto, P. Zanuttigh, G. M. Cortelazzo, and S. Mattoccia. Scene segmentation assisted by stereo vision. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 57–64. IEEE, 2011. 3

[12] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2

[13] Y. Gandelsman, A. Shocher, and M. Irani. " double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11026–11035, 2019. 3

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 3

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[16] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019. 2

[17] J. Iwasawa, Y. Hirano, and Y. Sugawara. Label-efficient multitask segmentation using contrastive learning. *arXiv preprint arXiv:2009.11160*, 2020. 2

[18] X. Ji, J. F. Henriques, and A. Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019. 1, 3, 4, 5, 6, 10

[19] Y. Jiao, T. D. Tran, and G. Shi. Effiscene: Efficient per-pixel rigidity inference for unsupervised joint learning of optical flow, depth, camera pose and motion segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5538–5547, 2021. 1, 3

[20] X. Jin, X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie, et al. Video scene parsing with predictive feature learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5580–5588, 2017. 1, 2

[21] O. S. Kayhan and J. C. v. Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020. 4, 7

[22] W. Kim, A. Kanezaki, and M. Tanaka. Unsupervised learning of image segmentation based on differentiable feature clustering. *IEEE Transactions on Image Processing*, 29:8055–8068, 2020. 3

[23] Y. Kim, S. Choi, H. Lee, T. Kim, and C. Kim. Rpm-net: Robust pixel-level matching networks for self-supervised video object segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2057–2065, 2020. 1, 3

[24] S. Kong and C. C. Fowlkes. Recurrent scene parsing with perspective understanding in the loop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 956–965, 2018. 2

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classifica-

tion with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[26] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *European conference on computer vision*, pages 577–593. Springer, 2016. 3

[27] H. Li, P. Xiong, J. An, and L. Wang. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018. 2

[28] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019. 2

[29] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1, 2

[30] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3623–3632, 2019. 1, 3

[31] P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*, 2016. 2

[32] A. Mahendran, J. Thewlis, and A. Vedaldi. Self-supervised segmentation by grouping optical-flow. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 1, 3

[33] I. Misra and L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020. 3

[34] A. Mousavian, H. Pirsiavash, and J. Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 611–619. IEEE, 2016. 2

[35] F. Nex, F. Remondino, M. Gerke, H.-J. Przybilla, M. Bäumker, and A. Zurhorst. Isprs benchmark for multi-platform photogrammetry. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2, 2015. 5

[36] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015. 2

[37] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016. 3

[38] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9359–9367, 2018. 3

[39] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 3

[40] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017. 2

[41] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2, 3, 5

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[43] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4481–4490, 2017. 2

[44] F. Tosi, F. Aleotti, P. Z. Ramirez, M. Poggi, S. Salti, L. D. Stefano, and S. Mattoccia. Distilled semantics for comprehensive scene understanding from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4654–4665, 2020. 3

[45] W.-C. Tu, M.-Y. Liu, V. Jampani, D. Sun, S.-Y. Chien, M.-H. Yang, and J. Kautz. Learning superpixels with segmentation-aware affinity loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 568–576, 2018. 2

[46] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 3

[47] C. Ventura, M. Bellver, A. Girbau, A. Salvador, F. Marques, and X. Giro-i Nieto. Rvos: End-to-end recurrent network for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5277–5286, 2019. 2

[48] D. Weikersdorfer, A. Schick, and D. Cremers. Depth-adaptive supervoxels for rgb-d video segmentation. In *2013 IEEE International Conference on Image Processing*, pages 2708–2712. IEEE, 2013. 3

[49] X. Xia and B. Kulis. W-net: A deep model for fully unsupervised image segmentation. *arXiv preprint arXiv:1711.08506*, 2017. 1, 3

[50] D. Xu, W. Ouyang, X. Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018. 1, 2

[51] G. Xu, Z. Liu, X. Li, and C. C. Loy. Knowledge distillation meets self-supervision. In *European Conference on Computer Vision*, pages 588–604. Springer, 2020. 3

[52] H. X. Y. H. A. Xu and J. L. W. Sun. Depth-aware space-time memory network for video object segmentation. *DAVIS challenge on video object segmentation*, 2020. 2

[53] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018. 3

[54] X. Zhan, X. Pan, Z. Liu, D. Lin, and C. C. Loy. Self-supervised learning via conditional motion propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1881–1889, 2019. 3

[55] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 3

[56] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 2

[57] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018. 2

[58] X. Zhao, R. Vemulapalli, P. Mansfield, B. Gong, B. Green, L. Shapira, and Y. Wu. Contrastive learning for label-efficient semantic segmentation. *arXiv preprint arXiv:2012.06985*, 2020. 2

[59] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8856–8865, 2019. 2

2
Introduction

Image segmentation is the process of dividing an image into disparate regions based on object colour, semantics or instance. In this procedure, every pixel of an image is given a label either on the basis of its colour or the object that it belongs to. Being a low-level classification process, the spatial location of the entire scene, captured in a given image has vital importance to the success of the task, as opposed to object detection or classification. Until the emergence of deep learning, the computer vision field was dominated by traditional machine learning algorithms. These included edge detection [2], super-pixels [1, 7], and clustering [3, 23], among others. The effectiveness of the most prevalent algorithms, such as SIFT [5] and HOG [6] which employed handcrafted features, greatly depended on domain experts to construct a reliable representation of the training data. The advent of AlexNet [13] transformed the computer vision field and shifted the focus of research to convolutional neural networks, or CNNs in short. Unlike handcrafted approaches, CNN models do not require additional human intervention, provided that they are supplied with sufficient annotated data. This factor and the unparalleled performance of CNNs contributed to their rapid expansion and increased interest from the research community. Semantic segmentation as a subfield of deep learning has critically influenced academic dialogue in recent years due to the outstanding performance of fully-convolutional networks [14, 16]. This type of architecture disposes of fully-connected layers, and instead comprises solely of convolutional filters. The common encoder-decoder network arrangement has become a standard in the field and has been implemented by many successful segmentation networks such as U-Net [17], SegNet [4], FastFCN [21] and Mask R-CNN [10]. This structure allows to learn high level image features by downsampling the spatial resolution of the input and then upsampling to produce a full resolution label map. This approach introduces a natural bottleneck to the architecture which efficiently stores the most prominent characteristic of the image. Image segmentation has found application most notably in the domains of autonomous driving [8, 22] and medical image diagnostics [15, 24]

2.1. Motivation

Despite the revolutionary success of deep learning, the excelled performance comes at a cost. To reliably train a model, an adequately large and rich dataset is required. Although, the development of CNNs gave rise to a substantial increase in data collection, the generation of a training dataset remains the largest obstacle to achieving high performance in a given task. Since acquiring and annotating data is a time consuming process, discussions regarding alternative methods for training have gained considerable momentum. The emphasis of this effort has been on reducing the required amount of training data or entirely eliminating its need. This has led to the emergence of two main fields of network optimisation, namely: supervised and unsupervised/semi-supervised approaches. Exemplary works from the supervised domain which aim to reduce the use of labels use a mixture of synthetic and real data to train networks [18, 20]. In this study we turn our attention particularly to unsupervised segmentation. By definition, the term *unsupervised learning* refers to a type of machine learning where the algorithm is not provided with labels while learning [11]. As a result, algorithms of this type identify patterns and occurrences in a given dataset autonomously. In the field of neural networks, examples of unsupervised approaches include autoencoders [19] and generative adversarial networks (GANs) [9].

The investigation of Ji *et al.* [12], which we analyse in detail in our study, is part of the unsupervised

representation-learning domain. Their method claims to be capable of performing semantic segmentation without requiring any pre-training or supervision. The approach learns to segment images by maximising the mutual information between an image and its augmented variant. In the current study, the augmentations consist of colour and geometric transformations. The authors argue that the algorithm can achieve impressive results without any supervision on satellite image segmentation. For this reason, we decided to replicate the experiments on a variety of different environments and hyper-parameters to pinpoint the key reasons behind its performance and identify any potential shortcomings.

2.2. Research question

We define the main research question as follows:

What are the generalisation and robustness capabilities of the Invariant Information Clustering method for semantic segmentation?

and address it by investigating these topics in particular:

Q1. What is the influence of the employed architecture?

Q2. How important is the choice of dataset?

Q3. How significant is the role of the learning rate in the optimisation of the network?

Q4. What is the most effective image transformation applied on the second image?

In this work, the authors employ a shallow neural network, hence the goal of the first question is to determine if deep segmentation architectures of the encoder-decoder type have any effect on the performance. Next, we analyse whether training on a dataset of urban street scenes would yield comparable results to those achieved on the satellite images, as reported in the paper. Further, the third question concerns the impact of the learning rate, since the one used in the original study is comparatively low. Finally, we address the use of image transformations and determine their effectiveness.

2.3. Overview

The following chapters will provide background information on the subjects that appertain to the main topic of the paper. Chapter 3 will introduce the basics of Deep Learning and the main building blocks of a convolutional neural network. Next, Chapter 4 will elaborate further into segmentation architectures and the principle behind their operation. Lastly, Chapter 5 will explain the notion behind representation and contrastive learning.

Bibliography

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282.

[2] Al-Amri, S. S., Kalyankar, N., and Khamitkar, S. (2010). Image segmentation by using edge detection. *International journal on computer science and engineering*, 2(3):804–807.

[3] Ali, M. A., Dooley, L. S., and Karmakar, G. C. (2006). Object based image segmentation using fuzzy clustering. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 2, pages II–II. IEEE.

[4] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

[5] Burger, W. and Burge, M. J. (2016). Scale-invariant feature transform (sift). In *Digital Image Processing*, pages 609–664. Springer.

[6] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

[7] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181.

[8] Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360.

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

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

[11] Hinton, G. E., Sejnowski, T. J., et al. (1999). *Unsupervised learning: foundations of neural computation*. MIT press.

[12] Ji, X., Henriques, J. F., and Vedaldi, A. (2019). Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874.

[13] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

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

[15] Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE.

[16] Papandreou, G., Chen, L.-C., Murphy, K. P., and Yuille, A. L. (2015). Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750.

[17] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[18] Sankaranarayanan, S., Balaji, Y., Jain, A., Lim, S. N., and Chellappa, R. (2018). Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3752–3761.

[19] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

[20] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977.

[21] Wu, H., Zhang, J., Huang, K., Liang, K., and Yu, Y. (2019). Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv preprint arXiv:1903.11816*.

[22] Zhang, Z., Fidler, S., and Urtasun, R. (2016). Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 669–677.

[23] Zheng, X., Lei, Q., Yao, R., Gong, Y., and Yin, Q. (2018). Image segmentation based on adaptive k-means algorithm. *EURASIP Journal on Image and Video Processing*, 2018(1):1–10.

[24] Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer.

3

Foundations of Deep Learning

Deep learning is a subfield of machine learning that draws inspiration from the structure of the neuron cell found in the brain. Algorithms of this type are fundamentally mathematical models which attempt to simulate processes that occur in the human brain. The models comprise of multiple layers of artificial neurons linked together, hence where the origin of the term *Artificial Neural Networks* (ANN) is derived from. Although, a network with a single layer is capable of making predictions, additional layers can improve accuracy and generalisation.

3.1. Neural Networks

This section will introduce the general theory behind conventional neural networks and their origins.

3.1.1. Perceptron

A perceptron is the building block of a neural network. It represents a simplified model of a brain neuron. Figure 3.1a illustrates a high-level diagram of the biological unit and Figure 3.1b its artificial counterpart. In the image on the left, we can see the general components of a neuron, with dendrites that receive signals, the cell body that processes input and the axon which transmits the impulse. A perceptron attempts to replicate this functionality by having an input component, processing phase and lastly an activation phase. Similar to the biological neuron, the perceptron processes the input and under the right conditions triggers a signal. The mathematical expression in Eq. 3.1 describes the operation within the perceptron. A weighted summation is performed between the product of vector input x_i and weights w_i, which afterwards is corrected with a bias term b.

$$a_j = \sum_{i=1}^{n} x_i w_i + b \tag{3.1}$$

Defined explicitly, the perceptron is a binary classification algorithm (predicts 2 values), which maps an input x to a resulting value $f(x)$. To extend the prediction capability of the perceptron beyond linearly separable patterns, activation functions are employed to allow non-linearity to be learnt. These take as input the result of Eq. 3.1, as is shown in Figure 3.1b. Common choices for activation functions are ReLu [29] and Sigmoid [30].

3.1.2. Multilayer perceptron

Figure 3.2a shows a diagram of a synapse, which is where transmission of electrical impulses happens between two nerve cells. As described before, signals travel in the neuron from the dendrites through the axon to reach the terminal axon. At this site the neuron is attached to the dendrites of another neuron cell. This chaining creates a network of millions of nerve cells which communicate together and exchange impulses. Equivalently, these processes are modelled in ANNs in smaller scales where neurons are structured together in multiple interconnected layers. The neurons between two adjacent layers are fully connected via weighted nodes, as illustrated in Figure 3.2b. As shown, a network of such type includes three main components: an input layer, a hidden layer(s) and an output layer. The input layer accepts numerically represented data that can be processed by the following layers. Hidden layers are positioned between the input and output layers

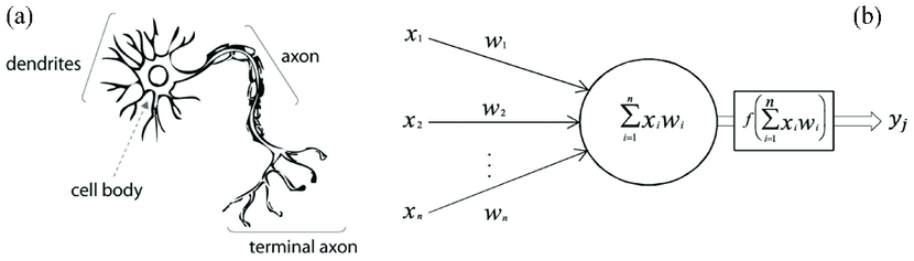


Figure 3.1: Comparison of a biological neuron and an artificial neuron: a) biological neuron; b) model of an artificial neuron [28]

and are responsible for expanding the capacity of the network. The same flow of steps is followed as with a single perceptron. The output layer is where the final result is obtained. In most cases to obtain a prediction, we need a probability distribution derived from the outputs of the final layer. It is usually computed by a softmax function, expressed by the following formula: $f_i(x) = \frac{exp(x_i)}{\sum_j exp(x_i)}$. The diagram in Figure 3.2b is an example of a simple feed forward network, which comprises of 2 hidden layers and 2 outputs nodes. Networks of this type formulate a combination of multiple approximation functions $f^n(f^{n-1}(f^{n-2}(..f^1(x))))$, where f^n corresponds to the first layer and f^n to the n-th layer of the ANN [25]. This factor renders them suitable for tasks which are difficult or impossible to compute with an exact mathematical solution.

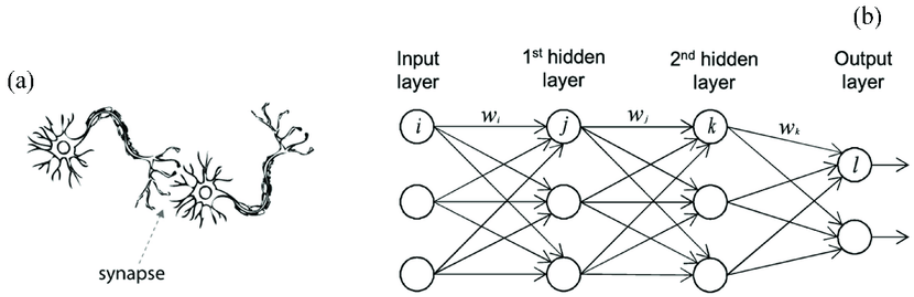


Figure 3.2: Comparison of a biological synapse a) and an artificial neural network b) [28]

3.1.3. Weight optimisation

In order to produce correct output when given specific input, ANNs undergo a procedure called "*training*". It refers to the tuning of the weights w and biases b of all neurons comprising the network. Learning of those parameters happens after each piece of data is processed, such that the margin of error between the output an expected result is minimised. Loss functions are used to measure the discrepancy between the outcome $\hat{y}_i$ and the truth value y_i. Often in neural networks two types of loss functions are used:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (3.2) \qquad\qquad Cross\ entropy = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot log(\hat{y}_i) \qquad (3.3)$$

Updating the weights is performed by computing the gradient of the chosen loss function with respect to each weight and bias of the network. Their values are tuned using gradient descent and the process is commonly referred to as *backpropagation*. Weights adjustment begins at the output layer and is propagated to the initial layers following the derivative chain rule. Training is conducted iteratively with small portions of the available data which are called *mini batches*. A full iteration over all mini batches is known as an epoch. Training usually requires a few dozen epochs.

3.2. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a class of ANNs which are commonly used in image analysis. They differ from standard neural networks in several key aspects with regards to vision applications. Firstly, in ANNs, the number of trainable parameters increase significantly with an increase in image size. As a result, their application in vision tasks is severely limited due to low scalability. Secondly, since the inputs to a standard ANN are fed in the form of a vector, the spatial features of an image are lost. The term *spatial features* refers to the arrangement of pixels in a given image. Additionally, CNNs employ a more advanced approach when analysing an image which allows to reuse weights for different locations of the same image. Finally, they utilise pooling layers (a.k.a. subsampling) which reduce the dimensions of the input by extracting the most significant information. These factors contribute to making CNNs more resource optimised and powerful predictors than ANNs in the vision domain. In the following section we will describe the basics of convolutional neural networks and their application in computer vision.

3.2.1. Convolution

At the core of CNNs lie convolutional layers comprised of filters which perform a convolutional function over some input matrix, hence the name of the layer. Specifically, the main advantage of a convolution layer is instead of having a weight for each input, there is a set of weights known as *kernels* or *filters* which is reused throughout the input and moves around the input matrix to compute outputs [32]. The process is shown in Figure 3.3, where the kernel is typically presented as a small matrix of size $W_{r\times c}$, for an input matrix V.

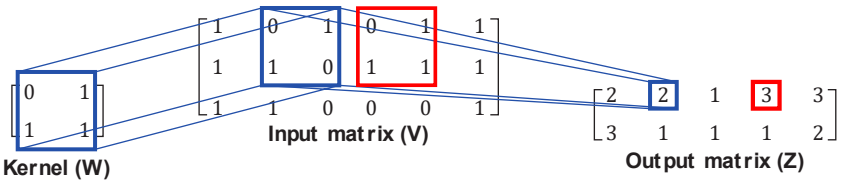


Figure 3.3: Convolution of a 2 × 2 Kernel by a 3 × 6 Input Matrix. [32]

Formally, the convolution function is given by:

$$z_{i,j} = \sum_{k=1}^{r} \sum_{l=1}^{c} w_{k,l} v_{i+k-1,j+l-1} \tag{3.4}$$

Each element from the output matrix is a summation of the one-by-one multiplications of the kernel elements into $r \times c$ window from the input matrix V. If we observe the red square in the input matrix and the corresponding element in the resulting matrix, we can see that the magnitude of each element directly corresponds to the resemblance of the kernel to the convolved input window. Since the window highlighted in red exactly matches the kernel we get a high activation value. In the computer vision domain, the output matrix is usually called a *feature map*, as it contains filtered image features and their relative locations within the image. To scan the entire image a filter slides across all cells of the input matrix in a predetermined step called *stride*. It is applied both horizontally and vertically. Convolutional networks typically have multiple filters stacked in banks in different layers. Filters from initial layers are capable of extracting low-level features such as edges, corners and colour combinations. As we progress deeper in the network, filters start to recognise higher-level features which are parts of entire objects such as eyes and feathers, if we are analysing a bird, for instance. Towards the output of the network, filters are capable of identifying complete objects such as bicycles, cars, dogs or cats. An example of the filtering process is shown in Figure 3.4.

3.2.2. Pooling

The area that a filter in a given layer can cover is known as *receptive field*. As previously described the receptive field of filters gradually increases as we go deeper in the network. This property of CNNs is additionally enhanced by special pooling layers. Their purpose is to decrease the spatial resolution of outputs while still retaining important features. Similar to convolution filters, a *max-pooling* function employs an $r \times c$ sliding window that traverses input features [34]. While moving, it calculates summary statistics of the scanned elements and outputs the maximum value of that neighbourhood. The stride of pooling filters determines

the extent of dimension reduction. Unlike convolution filters, the stride of pooling filters should not overlap with the corresponding $r \times c$ window from the input matrix, since this would distort the calculation. Likewise, an *average-pooling* function takes the average of the output elements, instead of the maximum value. Figure 3.4 illustrates the effect of a max-pooling function applied to a feature map.

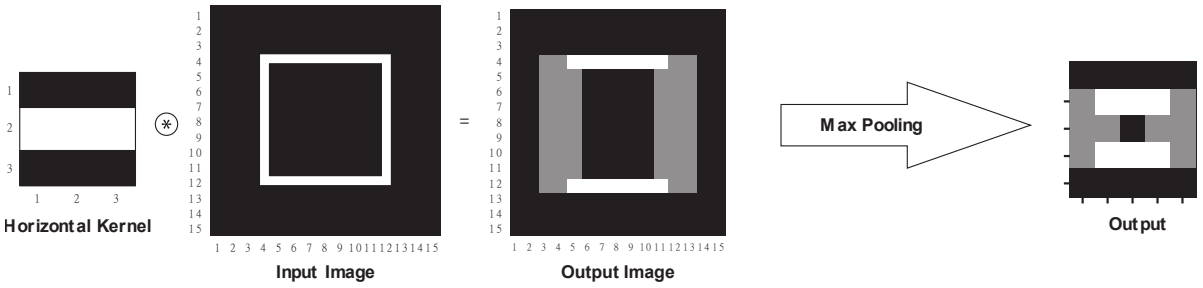


Figure 3.4: Example of extracting horizontal lines with convolution and applying max-pooling to reduce the output size [32]. Values in the output image, coloured in white indicate a higher activation, confirming that a filter has detected a feature.

3.2.3. Preventing overfitting

Most CNNs are trained under the supervision of ground truth labels. A general definition of any machine learning method is stated as approximating a target function $f(\cdot)$ that maps input variables X to an output variable Y. *Overfitting* refers to approximating target function $f(\cdot)$ exactly to the training data. As a consequence, target function $f(\cdot)$ cannot generalise well to unseen data. In other words, a given network will perform poorly when presented with data it was not trained on. The described event can occur when a CNN is trained on too few training samples. Networks with significantly more layers are prone to overfitting when trained on a small training set. The following sections will discuss standard techniques which prevent overfitting.

3.2.3.1 Dropout

Ensembles of neural networks with different model configurations are known to tackle overfitting well [33]. Having different architectures allows each model to learn a specific characteristic of the dataset. However, maintaining and training a set of networks is computationally expensive and time consuming. Instead, a single model can be used to emulate this behaviour by randomly choosing nodes from each layer and removing them during training. The technique is called *dropout* and has proven to be a very effective and cheap solution against overfitting. To implement dropout, a new hyper-parameter is added that defines the probability at which nodes from a given layer will be omitted while training. A typical value for intermediate layers is $p(0.5)$, while $p(0.8)$ is used for final layers. Dropout is not used when predicting. An illustration of the procedure on fully-connected layers in shown in Figure 3.5.

3.2.3.2 Weight regularisation

When training a neural network we learn the weights of the network. As training progresses they become more specialised to the input. As a result, weights grow in value to accommodate any variations and peculiarities seen in the dataset. Although, the model is well adapted to the specific dataset, the magnitude of the weights may make the network unstable, which implies that small differences in the input can cause major changes in the output. This leads to poor generalisation and knowledge transfer. A technique known as *weight regularisation* is introduced to overcome this issue. Weight regularisation is used to encourage the network to keep weight values small which helps the model discard irrelevant or too specific details about the training examples. It is implemented by adding an additional term to the loss function, which represents the current size of all weights. Since the objective is to minimise the loss function, a larger weight term will penalise the network more. The end result will be a more generic and stable network. There exist two approaches of calculating the size of all parameters, namely: L1 (Lasso regression) and L2 (Euclidean Norm) [27].

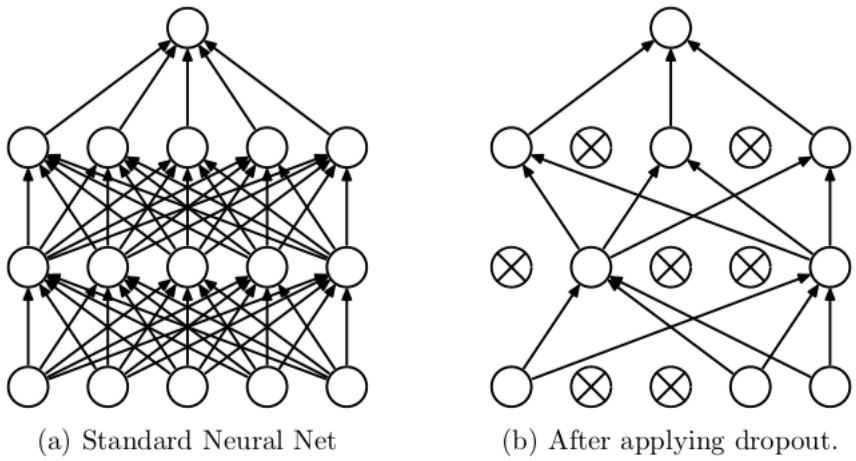


Figure 3.5: Dropout applied on fully-connected layers [33].

3.2.3.3 Early stopping

A common challenge when training a neural network is to determine the number of training epochs, or otherwise stated – the duration of training. Training for too long will overfit the model to the dataset, while not enough training may underfit it. *Early stopping* is a commonly used technique to address this question. Usually, a dedicated validation set is employed alongside the training set, which is not used in training and its purpose is to evaluate the performance on unseen examples. While training the loss on both datasets is monitored. Typically, when the validation loss begins to diverge, it is the best moment to suspend training. Example loss curves which show the process are shown in Figure 3.6.

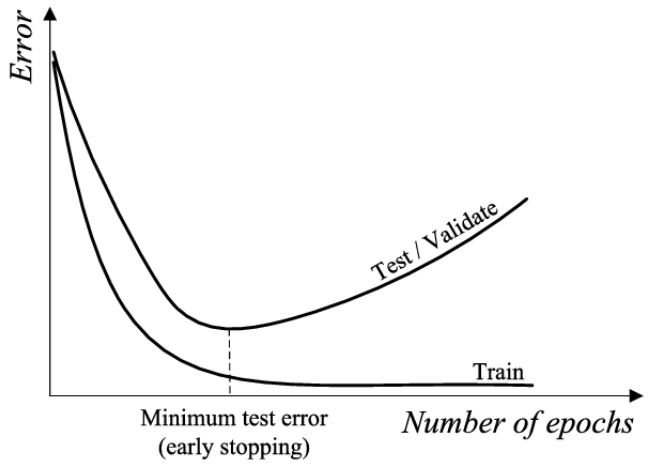


Figure 3.6: Timeline showing best moment to suspend training [31].

3.2.4. Batch normalisation

Upon creation, a neural network will have all its weights randomly initialised. This can be challenging for training because after each batch the inputs from previous layers can change dramatically. Therefore later layers may encounter difficulties adapting to the ever changing input. In backpropagation, when updating the weights of a layer it is assumed that the outputs of the previous layer follow a certain distribution. This, however, is not case since every mini-batch and its feature map outputs are likely to vary significantly [26].

Batch normalisation operates by computing the mean and variance of each batch, following Eq. 3.5. The final normalised input of layer k is computed by Eq. 3.6, where γ and β are learnable parameters which control the scale and shift of the new distribution.

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \tag{3.5}$$

$$y^{(k)} = \gamma \hat{x}^{(k)} + \beta \tag{3.6}$$

This ensures that the distribution which a subsequent layer expects will not differ substantially, thus stabilising the training process. This reduces the required number of epochs to train a model sufficiently and mitigates fluctuations of the loss. Besides these benefits, batch normalisation also has a regularisation effect on training.

Bibliography

[25] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[26] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[27] Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

[28] Meng, Z., Hu, Y., and Ancey, C. (2020). Using a data driven approach to predict waves generated by gravity driven mass flows. *Water*, 12(2):600.

[29] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.

[30] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

[31] Santos, J. M. F. d. et al. (2007). Data classification with neural networks and entropic criteria.

[32] Sharda, R., Delen, D., and Turban, E. (2020). *Analytics, Data Science, & Artificial Intelligence*. Pearson Education, Limited.

[33] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[34] Zhou, Y.-T., Chellappa, R., Vaid, A., and Jenkins, B. K. (1988). Image restoration using a neural network. *IEEE transactions on acoustics, speech, and signal processing*, 36(7):1141–1151.

4

Foundations of Image segmentation

The introduction of CNNs in the domain of Image segmentation has improved overall quality and speed of prediction. In this chapter we will explain the knowledge distillation that happens in a neural network and present various segmentation methods.

4.1. Segmentation in Deep Learning

A very common architecture in the segmentation field is the encoder-decoder type. Unlike standard CNNs which implement several fully-connected layers at the output, the architecture in question does not employ any. Since the use of fully-connected layers requires feature maps to be flattened to a 1-D vector, the spatial information which the maps contain is lost. For this reason the fully-connected layers are discarded and replaced by more convolutional layers. An illustration of such a network is displayed in Figure 4.1.

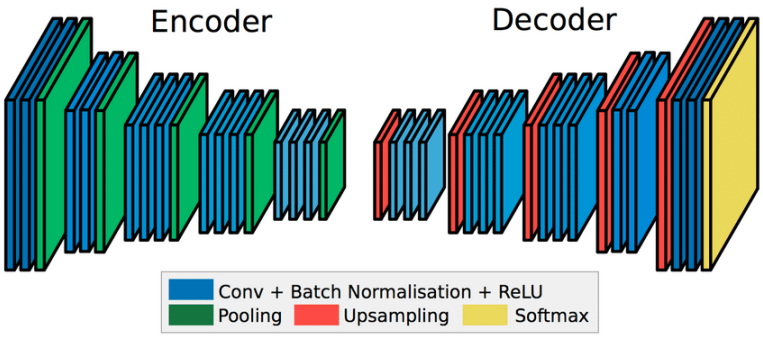


Figure 4.1: Example of an encoder-decoder CNN used for segmentation [35].

The output of the last layer of the encoder is referred to as the *bottleneck*. It represents a low-resolution tensor containing the highest level information encoded in the feature maps. An example of the output of VGG16's fifth convolutional layer [53] is shown in Figure 4.2. As we can see, the outputs do not resemble a meaningful image. Instead, the contents of each box indicate the presence of a high-level feature. Brighter colours denote a higher confidence, whereas dark areas suggest that a specific feature is not detected. The role of the decoder is to "*decipher*" the contents of each box and reconstruct the input image according to a given label. Gradually feature maps are upscaled and each subsequent convolutional layer learns how to decode the outputs of the previous layer. Upsampling is achieved through special layers which use bilinear interpolation or transposed convolutions [38]. The entire process of rebuilding the feature maps to an image segmentation is guided by the labels.

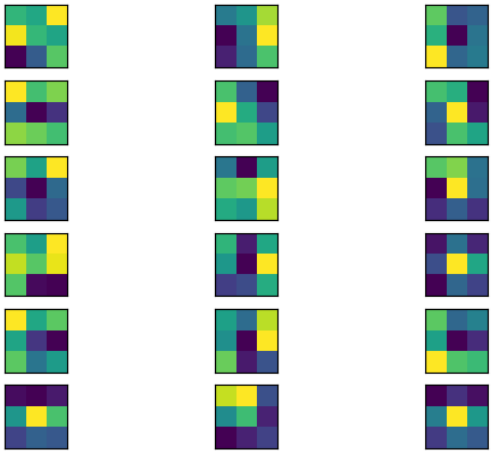


Figure 4.2: Low resolution feature maps produced by the 5th layer of VGG16

4.2. Overview of segmentation architectures

The following sections will present the most prominent approaches to image segmentation in the deep learning domain.

4.2.1. Pyramid and Multi-Scale architectures

Pyramid architectures propose the use of multi-scale convolution to capture objects of different size. One of the fundamental works in the field is Feature Pyramid Network (FPN) by Liu *et al.* [49], Figure 4.3. Feature maps are generated at different scales and are subsequently linked together in order to share feature cues. Each stage is processed by a 3×3 filter to obtain output. Another approach is the Pyramid Scene Parsing Net (PSPNet) [56], Figure 4.4 which focuses on global information. The authors use Pyramid Pooling on extracted feature maps to detect patterns at different scales. Further pooling at four different sizes is performed inside the pyramid and each result is processed by a 1x1 convolutional layer. The outputs of the pyramid are upscaled and concatenated with early feature maps to retain low-level details. Anoher work named RefineNet [48] introduces a multi path network which enhances feature maps from initial layers through residual blocks to obtain high-resolution segmentation mappings. Feature maps are processed at four different scales with the purpose of capturing details with different proportions. Each refinement stage is supplied with pooling and residual connections from previous stages.

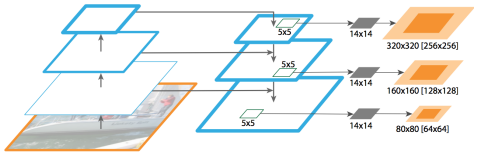


Figure 4.3: Feature Pyramid Network architecture [49]. Each stage is divided into 5×5 regions which are processed by an MLP.

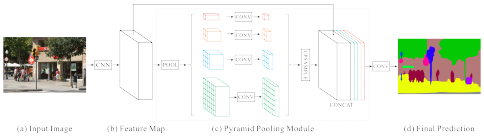


Figure 4.4: Pyramid Scene Parsing Network [56]. Pyramid Pooling is applied to the feature map of a CNN. All outputs are concatenated and segmented by a final convolutional layer.

4.2.2. Atrous convolution

The literal translation of the term "atrous" is derived from the french "à trous", meaning "with holes". It refers to the concept of dilated convolution which introduces spacing inside filter weights with the purpose of expanding their receptive field. Figure 4.5 illustrates a comparison between atrous and standard convolutions. Dilated filters impose no additional computational overhead and are a preferred choice for real-time seg-

mentation. As a result a dilated 3 × 3 filter will cover the same area as a 5 × 5 filter, while using only 9 weights instead of 25.

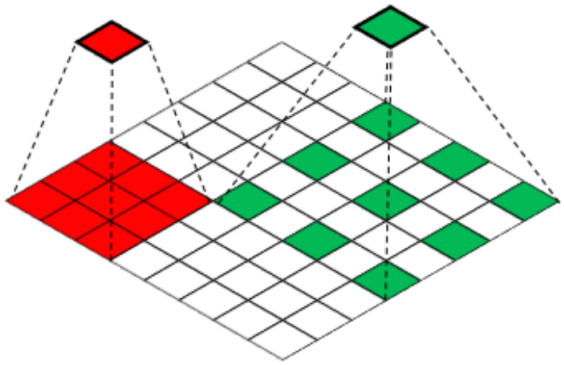


Figure 4.5: Standard convolutions (red) vs Atrous convolution (green) [39].

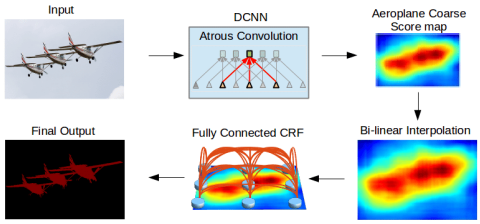


Figure 4.6: DeepLabv1 architecture [36] . Images are processed by atrous filters and then upscaled using bilinear interpolation. Finally, CRF is used to refine the segmentation output.

DeepLab1 [36], Figure 4.6 and DeepLab2 [37] are the most widely known solutions to use such filtering. Segmentation models are usually required to generate full resolution outputs. To achieve this the modification was employed with the purpose of addressing the issue of resolution reduction that occurs with pooling and strided convolution. Furthermore, atrous convolutions are more efficient at capturing contextual information due to the larger receptive field, which otherwise could only be achieved through larger and more computationally expensive filters. Additionally, the DeepLab family of networks employs pyramid pooling features and CRFs [46] for better feature extraction and boundary definition.

4.2.3. R-CNN architectures

R-CNN based models [40, 41, 52] have become popular due to their success in object detection and classification tasks. These networks use a dedicated region proposal sub-network to propose bounding box candidates. The proposals are then processed by the backbone of the architecture to finalise the bounding box coordinates and predict the contained class, as shown in Figure 4.7.

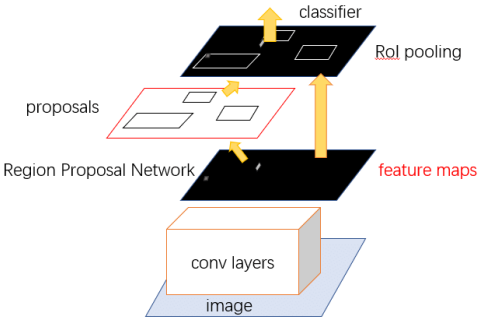


Figure 4.7: Faster R-CNN architecture [52].

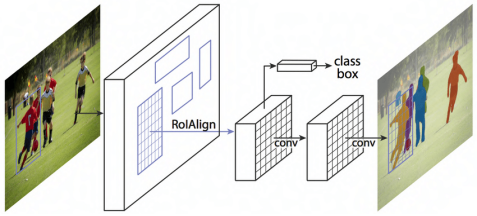


Figure 4.8: Mask R-CNN architecture [43].

The continuation of these papers, Mask R-CNN [43], Figure 4.8 includes object instance segmentation functionality. The model is capable of detecting, classifying and segmenting a given object from an input image. Mask R-CNN is an improved Faster R-CNN [52] with three output heads, one for each type of prediction. The loss function of Mask R-CNN jointly optimises for all three tasks: bounding box regression, classification and binary segmentation. Further models have been developed by other authors which are based on the R-CNN architecture. The Path Aggregation Network (PANet) [50] employs an FPN network to extract features at different scales. To construct a condensed feature tensor the outputs of FPN stage are processed by 3 × 3 filter and then pooled via an adaptive feature pooling method. Similarly to Mask R-CNN, the network contains three output branches for the same prediction tasks.

4.2.4. Recurrent architectures

Recurrent neural networks (RNN) have also found application in segmentation. Research in the field has proven that they can be useful for encoding short and long term dependencies between pixels to improve segmentation performance. With this approach pixel information is chained thus allowing global contextual information to be preserved. ReSeg [54] is a well-known RNN used in segmentation. It comprises of a backbone VGG-16 network that extracts generic features. Following that are RNN layers which scan the image horizontally and vertically to encode relevant global information and patterns. The output of the RNN layers is upsampled to be brought back to the original resolution. Gated Recurrent Units are employed throughout the network to balance processing time and performance quality. Another work by Liang *et al.* [47] employs Long Short-Term Memory (LSTM) [44] along with graphs to construct a segmentation model. The approach creates a graph topology on superpixels (clustering of pixels) rather than on single pixels. A CNN process the image in the standard method and generates confidence maps through a 1×1 filter. These are then appended with global features from the LSTMs. The graph constructed from the superpixels is initially updated by the confidence map and afterwards by each successive LSTM module.

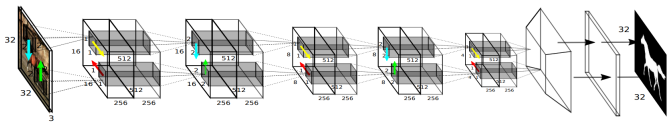


Figure 4.9: ReSeg architecture [54]. Here are shown the RNN modules which scan the image horizontally and vertically.

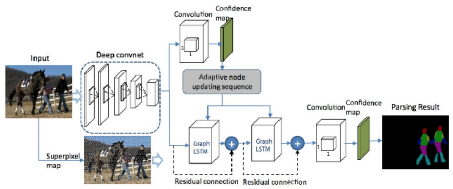


Figure 4.10: LSTM graph architecture [47]. LSTM modules update the graph for successive modules. Local features from standard convolutions are enhanced with global context information by the recurrent modules.

4.2.5. Generative and Adversarial architectures

Generative Adversarial Networks (GAN) [42] have revolutionised the deep learning field and have found application in almost all domains of computer vision, including segmentation. The most straight-forward approach is to train a segmentation CNN together with a discriminator network that differentiates whether a segmentation map is real or synthetic. Such a method is described in [51] and the architecture is illustrated in Figure 4.11. SeGAN by Xue *et al.* [55], Figure 4.12 is a GAN which employs a multi-scale L1 loss for medical image segmentation. Again a standard FCN is used to perform image parsing whose output is fed to a discriminator network. The novelty in the second network is the addition of multi-scale pooling which allows both the segmentator and the discriminator to learn local and global contextual features. Hung *et al.* [45] propose a triple loss for training an adversarial network, which comprises of a cross-entropy loss to optimise a segmentation network, adversarial loss for the discriminator network and a semi-supervised loss on the basis of the confidence map produced by the discriminator.

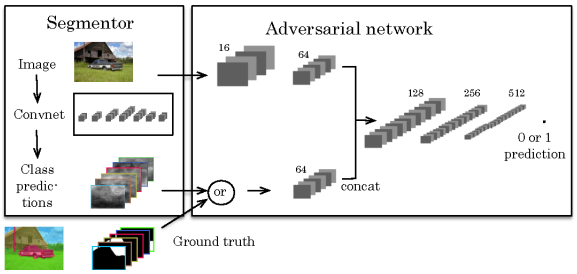


Figure 4.11: Standard approach to segmentation with GANs [51].

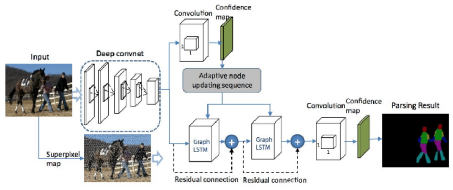


Figure 4.12: SegAN architecture [55]. The discriminator combines several feature map scales in order to capture relevant contextual information. The loss is specifically designed to handle different sized inputs.

Bibliography

[35] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

[36] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

[37] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

[38] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

[39] Ghosh, S., Das, N., Das, I., and Maulik, U. (2019). Understanding deep learning techniques for image segmentation. *ACM Computing Surveys (CSUR)*, 52(4):1–35.

[40] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[41] Gkioxari, G., Hariharan, B., Girshick, R., and Malik, J. (2014). R-cnns for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*.

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

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

[44] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[45] Hung, W.-C., Tsai, Y.-H., Liou, Y.-T., Lin, Y.-Y., and Yang, M.-H. (2018). Adversarial learning for semi-supervised semantic segmentation. *arXiv preprint arXiv:1802.07934*.

[46] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[47] Liang, X., Shen, X., Feng, J., Lin, L., and Yan, S. (2016). Semantic object parsing with graph lstm. In *European Conference on Computer Vision*, pages 125–143. Springer.

[48] Lin, G., Milan, A., Shen, C., and Reid, I. (2017a). Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934.

[49] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017b). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

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

[51] Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*.

[52] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99.

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

[54] Visin, F., Ciccone, M., Romero, A., Kastner, K., Cho, K., Bengio, Y., Matteucci, M., and Courville, A. (2016). Reseg: A recurrent neural network-based model for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 41–48.

[55] Xue, Y., Xu, T., Zhang, H., Long, L. R., and Huang, X. (2018). Segan: Adversarial network with multi-scale l 1 loss for medical image segmentation. *Neuroinformatics*, 16(3):383–392.

[56] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890.

5

Self-supervised Representation Learning

By definition the terms *representation learning* or *feature learning* refer to a set of algorithms which automatically discover patterns and features from given input and reuse this for future referencing such as classification or detection of unseen data [58]. Representation learning can be of two types:

- **Supervised**
- **Unsupervised / Self-supervised**

Supervised methods are those that require annotated data for training. Examples include ANNs, single perceptrons and CNNs, as explained in Chapter 3. On the other hand, unsupervised methods learn without labels. Such types are dimensionality reduction, autoencoders, CNNs with special loss functions and clustering algorithms. This chapter will describe some general unsupervised methods including the technique, also known as *Constrastive Learning*, which is used in the analysed paper from Chapter 1.

5.1. Clustering

Clustering is the task of grouping a set of data points under a predefined number of groups, based on specific characteristics of the data. The end result is such that all points in a given group are most similar between each other than those outside of their category. The groups in which they are allocated are called *centroids* or *clusters*. A criteria for clustering together a set of points may be but not limited to choice of distance metric between elements, close proximity in feature space of members or a statistical distribution. In the sections below are presented examples of two widely used clustering algorithms.

5.1.1. *k*-means

k-means belongs to the family of clustering algorithm. Fundamentally, it operates by dividing a given dataset into n distinctive groups named clusters, where each one is defined by a centroid value. The objective of the algorithm is to associate all points to a given cluster based on a similarity metric. To consider a point as a member of a cluster it needs to be more similar to the centroid of the given cluster than to any other centroid. *k*-means is an iterative algorithm which proceeds in several steps. Initially we begin by picking the number of clusters. Next, as shown in Figure 5.1 centroids are assigned to randomly chosen data points. Following that, Figure 5.2, samples x are grouped with their nearest centroid C such that $\sum_{i=1}^{m} ||x_i - C||^2$ is minimised. In the next step, the centroids are recalculated, according to all current members of the cluster, as given by: $C = \frac{\sum_{i=1}^{m}}{m}$, and illustrated in Figure 5.3. Finally, steps 3 and 4 are repeated until one of the following conditions is satisfied:

- Centroids stop changing.
- Data points do not change cluster membership.
- A predefined number of iterations is reached.

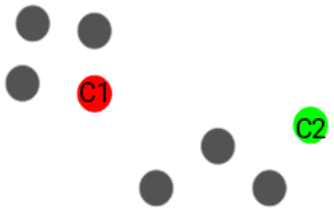


Figure 5.1: *k*-means second step. Random initialisation of cluster centroids [68]

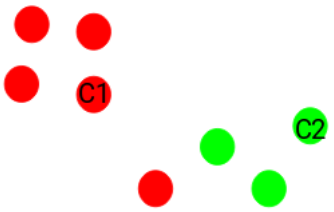


Figure 5.2: *k*-means third step. Assignment of all points to their respective closest cluster centroid [68].

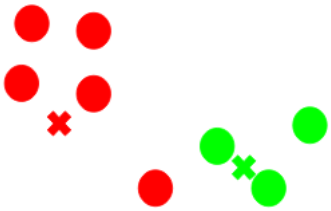


Figure 5.3: *k*-means fourth step. Recomputing new centroids based on all comprising members of a cluster [68].

Figure 5.4: *k*-means fifth step. Repetition of steps 3 and 4 until convergence [68].

5.1.2. Hierarchical clustering

Hierarchical clustering shares similarities with *k*-means since data points are also grouped together based on a distance metric. The algorithm begins by placing each sample in its own cluster. Next, a proximity matrix is computed that reflects the distance between all data points, as shown in Figure 5.6. The values along the diagonal will always be equal to 0, since they show the distance of the sample to itself, hence no distance. The purpose of the matrix is to determine the clusters with the smallest distance to be merged.

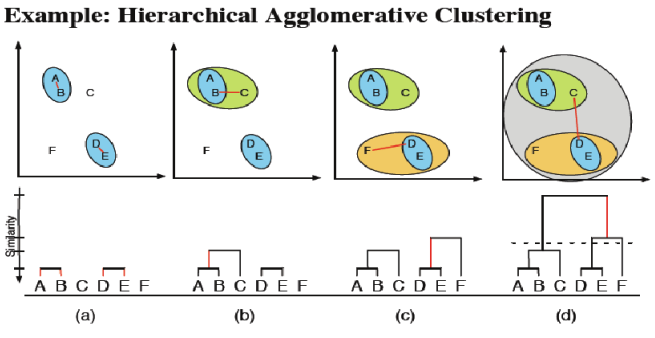


Figure 5.5: Example of agglomerative hierarchical clustering. Clusters are joined based on the distance calculated by a distance metric. The dendrogram at the bottom shows the proximity of each cluster with respect to all remaining. The vertical length of each edge is proportional to the distance between a pair of clusters [62].

Distance Matrix

	A	B	C	D	E	F
A	0	16	47	72	77	79
B	16	0	37	57	65	66
C	47	37	0	40	30	35
D	72	57	40	0	31	23
E	77	65	30	31	0	10
F	79	66	35	23	10	0

Figure 5.6: Distance matrix to be used in agglomerative hierarchical clustering.

Once the closest samples are identified and merged, the distance matrix is recomputed, to account for the coupled clusters, where they are treated as a single newly created cluster. The process is repeated until we obtain a single aggregated group. An example of the algorithm is shown in Figure 5.5. To select the number of final clusters, we dissect the longest edge of the resulting dendrogram, shown at the bottom of Figure 5.5. The number of intersected lines denotes the number of final clusters, in the example from the same Figure, the number of clusters will be 3.

5.2. Dimensionality Reduction

Dimensionality Reduction is the process of transforming data with high dimensionality into data with low dimensionality, such that essential features are preserved, as demonstrated in Figure 5.7. Reducing dimensions of a given input alleviates problems such as *curse of dimensionality* [57] and inefficient computability. It is often employed as a precursory step when performing data visualisation or cluster analysis. Algorithms from this domain are divided in linear and nonlinear. We will focus on a specific linear method commonly referred to as Principal Component Analysis (PCA). Other nonlinear methods include but not limited to Linear Discriminant Analysis (LDA) and Factor Analysis (FA).

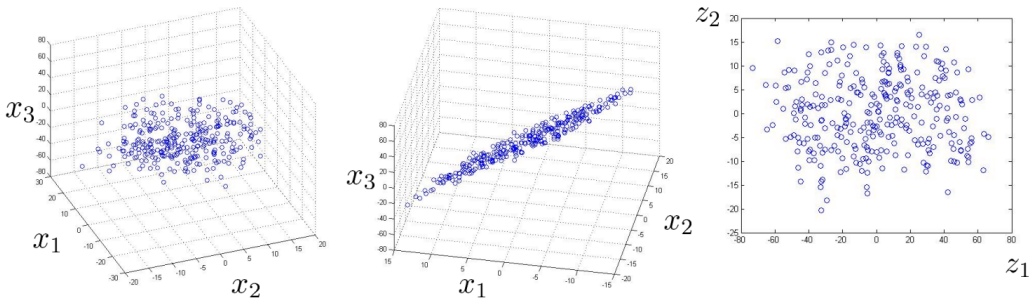


Figure 5.7: Illustration of dimensionality reduction. The points in 3D space in the far-left have been recast to 2D space, as shown in the rightmost plot. Taken from Andrew NG, at Coursera.

By definition a principal component of a collection of samples is represented by a hyper-plane in K-dimensional space that explains the maximum amount of variance, such that most of the information of the data is captured. We can see that such a hyper-plane illustrated in Figure 5.8 under the name *PC1*, which means first principal component. This component represents the line which best approximates the sample space.

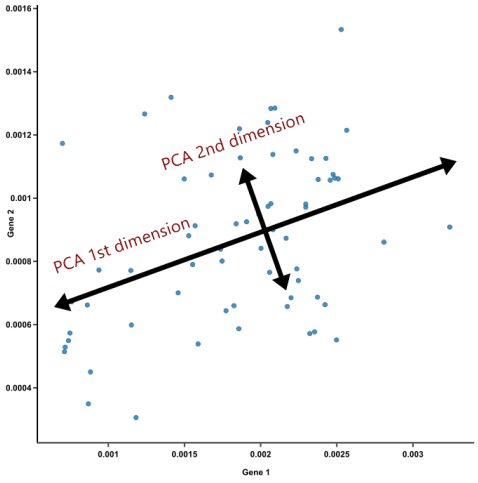


Figure 5.8: Sample of data points, where principal components (PC) 1 and 2 are shown [65].

The first step of PCA is to standardise the data points by scaling their values to unit variance and mean-centering the entire population. Following that, the first principal component is computed, as shown in Figure 5.8. Afterwards the second principal component (PC2) is derived such that it best explains the second largest source of variation in the data, while also being orthogonal to PC1. Considering a 3-dimensional space, the two principal components define a hyper-plane of lower-dimensional sub-space which can be

plotted in 2D space and visualised. This allows to eliminate a given dimension while still preserve meaningful properties of the dataset. PCA is not limited to three dimensions and can be applied on K-dimensional problems.

5.3. Autoencoders

Autoencoders are a type of CNNs that are used to learn compact encodings of unlabelled data. The encoding is learnt by reconstructing the input image as output of the CNN. A diagram of the process is shown in Figure 5.9. To force the network into generating a compressed representation a bottleneck layer is imposed on the network [67]. Typically, autoencoders are employed on tasks which involve anomaly detection, dimensionality reduction, facial recognition and others.

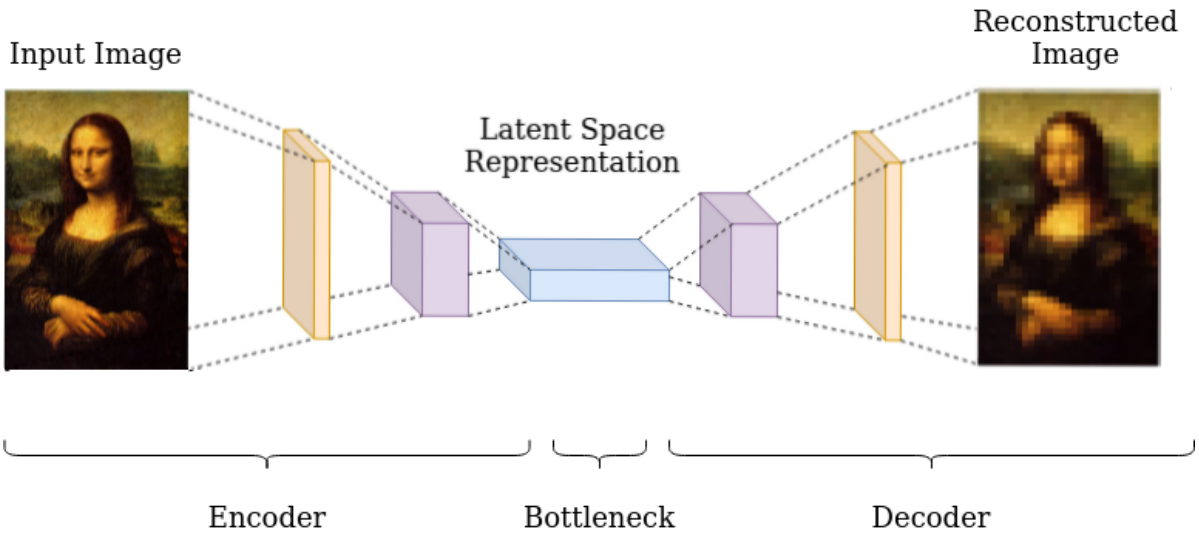


Figure 5.9: An input image is processed by an encoder and compressed into a compact representation. The role of the encoder is to learn to reconstruct the same image given its representation.

In the domain of image reconstruction, the bottleneck layer restricts the flow of information from the input to the output, thus not allowing the network to directly learn the pixel values of the image. Therefore, the network is forced to learn only specific features of data and discard redundant information. In certain implementations the latent space representation in the bottleneck layer is generated by fully connected layers, which implies that it takes the form of a vector. Training of autoencoders is done through backpropagation based on the reconstruction error. Types of autoencoders include sparse [64], denoising [60], contractive [66] and variational [63] amongst others.

5.4. Contrastive Learning

The idea behind contrastive learning is to learn representations such that similar samples remain close to each other, while different samples are far apart, when plotted in feature space. More specifically, in the field of Deep Learning, contrastive learning refers to the act of learning general features of the input data, without the use of labels. This is achieved by enforcing a model to learn which samples are similar to each other. Figure 5.10 illustrates an example between two samples of the same class and a third one which is different.

By observing the animals in Figure 5.10 we can recognise a number of high-level similarities. For instance, cats have pointy ears and short snouts, also usually their eyes are bright coloured. In contrast, dogs' ears are more often relaxed and floppy, while their noses are longer and more projecting from the face. The CNN we use in our use case needs to observe the 2 pairs of images and determine which 2 belong to the same class. This is a very convenient technique as the model does not require annotated data [69].

Figure 5.12 presents a standard contrastive learning framework. Since sorting a dataset by similar samples is time consuming and is a form of supervision the use of augmentations is required to represent similar instances of the same class. Typical augmentations include cropping, resizing, colour and geometric transformations. The goal is to modify the sample while still preserving class specific features. Such is the case

Figure 5.10: Although, the two cats in the images on top are not the same, humans are capable of recognising that they are the same type of animal. Likewise, we can instantly differentiate between the cat and the dog. Contrastive learning attempts to replicate the same approach [69].

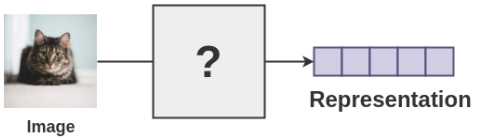


Figure 5.11: To optimise without the use of labels we need to generate a vector representation of a given image. The goal is to generate similar vectors for similar images [69].

with the kitten, which has been recoloured in a blue shade.

Once both images are handled by the CNN, the obtained compact feature map representation is fed to the several stages of fully-connected layers which will generate the vector representation, Figure 5.11. The entire architecture can be viewed as a function $h = f(x)$ which takes input x and produces output h in latent space.

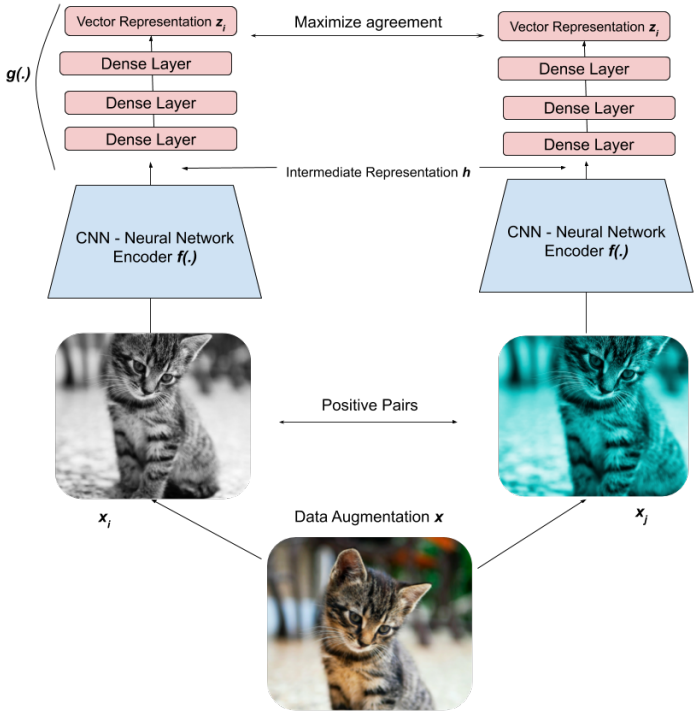


Figure 5.12: SimCLR framework [59, 69]. CNN architecture is ResNet-50 [61]. We can see the real and augmented images being passed to the same model to obtain their representation and compute a similarity score.

The similarity between the generated vectors needs to be maximised and therefore a metric to quantitatively measure their alikeness is required. Given the problem domain, we can measure the angle between two vectors in 2D space to determine their equivalence. This is computed by the cosine similarity, which is given by:

$$similarity = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \quad ||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \sqrt{\sum_{i=1}^{n} {B_i}^2}} \tag{5.1}$$

where A and B are our samples. When the angle between the two vectors is small we can conclude that they

are similar, meaning they overlap and vice versa when the angle is large then the vectors are unalike. In order to optimise the model, a loss function is required that will penalise when the vectors are far apart, and conversely encourage the network to continue optimisation in the given direction when the vectors are close to each other. The authors of [59] use a *Normalised Temperature-Scaled Cross-Entropy* loss, which calculates the probability that the initial images are similar. Let $sim(\cdot)$ denote the cosine similarity between two samples. The function for a positive pair (an image and its augmented variant) is given by:

$$\ell_{i,j} = -log\frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} exp(sim(z_i, z_k)/\tau)} \tag{5.2}$$

where $\mathbb{1}_{[k\neq i]}$ evaluates to 1 *iff* samples k and i are different. τ is the temperature hyper-parameter which controls the degree of how positive or negative a pair is. The $-log(\cdot)$ of the loss function is considered since we are interested in the negative of the maximised probability that the input images are similar or identical. The complete loss is computed on all z_i, z_j positive pairs in a random batch.

Bibliography

[57] Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

[58] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[59] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[60] Cho, K. (2013). Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. In *International conference on machine learning*, pages 432–440. PMLR.

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

[62] Janssen, P., Walther, C., and Lüdeke, M. K. (2012). Cluster analysis to understand socio-ecological systems: a guideline.

[63] Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.

[64] Makhzani, A. and Frey, B. (2013). K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*.

[65] Ng, A. (2018). Machine learning. Dimensionality reduction.

[66] Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., and Glorot, X. (2011). Higher order contractive auto-encoder. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 645–660. Springer.

[67] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

[68] Sharma, P. (2019). Comprehensive guide to k-means clustering. Available at `https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/`.

[69] Tiu, E. (2021). Understanding contrastive learning. Available at `https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607`.