University of São Paulo
Institute of Mathematics and Statistics
Bachelor of Computer Science

A study on image segmentation with convolutional neural networks

Text segmentation problem in manga

Pedro Henrique Barbosa de Almeida

Final Essay

mac 499 — Capstone Project

Program: Computer Science
Advisor: Profª. Drª. Nina S. T. Hirata

During this work, the author was supported by São Paulo Research Foundation (FAPESP), grant 2020/02891-3.

São Paulo
January 30th, 2021

Acknowledgements

"Ne me demandez pas qui je suis et ne me dites pas de rester le même"
— Michel Foucault

I would like to thank:

My parents, for always giving their best to provide me everything that I need, for teaching me solid values of how to be a decent human, and for stand by me in every tough situation I faced in this journey.

My supervisor, for her supervision, attention, patience, didactic, empathy, and for opening doors in my academic journey.

My *fiancé*, for loving me, for being there for me in the hours of despair, for being my reviewer and critic.

My little brother, for being the motivation of my strength to overcome obstacles and of my desire to go beyond.

My college friends, for helping me in everything that I needed, for hearing me when I needed to let off steam.

My extra-college friends, for being my relief valve in these hard times.

IME-USP, for providing me the privilege of being in touch with highly skilled specialists and admirable professors that were inspirations to me.

FAPESP, for allowing me to dedicate myself exclusively to this project, through its grant.

Abstract

Pedro Henrique Barbosa de Almeida. **A study on image segmentation with convolutional neural networks: *Text segmentation problem in manga***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2021.

The text segmentation problem in mangas consists of breaking images of manga's pages into "text" or "non-text" regions. This classification task can be modeled by a local function that will be learned by a supervised learning algorithm. To learn this function, the algorithm will perform a minimization of the difference between the expected and target outputs, adjusting its weights. In this work, we investigate the impacts of varying some parameters and aspects of a model, namely U-Net, in the context of few training examples. To achieve that, we had to build a network using the Keras framework and applied it to the text segmentation problem in mangas, to measure the model's performance. Among the results we obtained, we noticed that a model with depth 3, input normalization, and with generalized dice loss function is robust enough to generalize for different comic titles, even when it is trained on one title and tested on another. In summary, we have obtained a very strong network for this problem. Even though it may not be the optimal model, it may serve as a guide to answer further research questions.

Keywords: Deep learning. Small training data. Manga. Text segmentation.

Resumo

Pedro Henrique Barbosa de Almeida. **Um estudo sobre segmentação de imagens com redes convolucionais**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

O problema de segmentação de texto em mangás consiste em quebrar as imagens das páginas do mangá em regiões de "texto" ou "não texto". Essa tarefa de classificação pode ser modelada por uma função local, que será aprendida por um algoritmo de aprendizado supervisionado. Para aprender essa função, o algoritmo fará a minimização da diferença entre a saída esperada e a desejada, ajustando seus pesos. Neste trabalho, investigamos os impactos da variação de alguns parâmetros e aspectos de um modelo, chamado U-Net, em um contexto com poucos exemplos de treinamento. Para isso, construímos uma rede utilizando o framework Keras e a aplicamos no problema de segmentação de texto em mangás, a fim de medir o desempenho do modelo. Dentre os resultados obtidos, notamos que um modelo com profundidade 3, normalização de entrada, com função de perda Generalized Dice Loss é robusto o suficiente para generalizar para diferentes títulos de quadrinhos, mesmo quando é treinado em um título e testado em outro. Em resumo, obtivemos uma rede muito forte para esse problema. Mesmo que não seja o modelo ótimo, poderá servir como um guia para responder a outras questões de pesquisa futuras.

Palavras-chave: Aprendizado profundo. Conjunto de treino pequeno. Mangá. Segmentação de texto.

List of Abbreviations

AH	AosugiruHaru
CCs	Connected components
CE	Cross-Entropy
CNNs	Convolutional Neural Networks
Concat	Concatenation
Conv	Convolution
EL	EvaLady
FL	Focal Loss
GDL	Generalized Dice Loss
JBF	JijiBabaFight
MSNN	MariaSamaNihaNaisyo
Std	Standard deviation
WCE	Weighted Cross-Entropy

List of Symbols

Ψ	Text/non-text classifier
Ψ_w	Local function Ψ centered on window w
$\mathcal{I}$	Set of input gray scale images
$\mathcal{O}$	Set of ouput binary images (predictions)
$\mathcal{T}$	Set of target binary images (ground-truth)

List of Figures

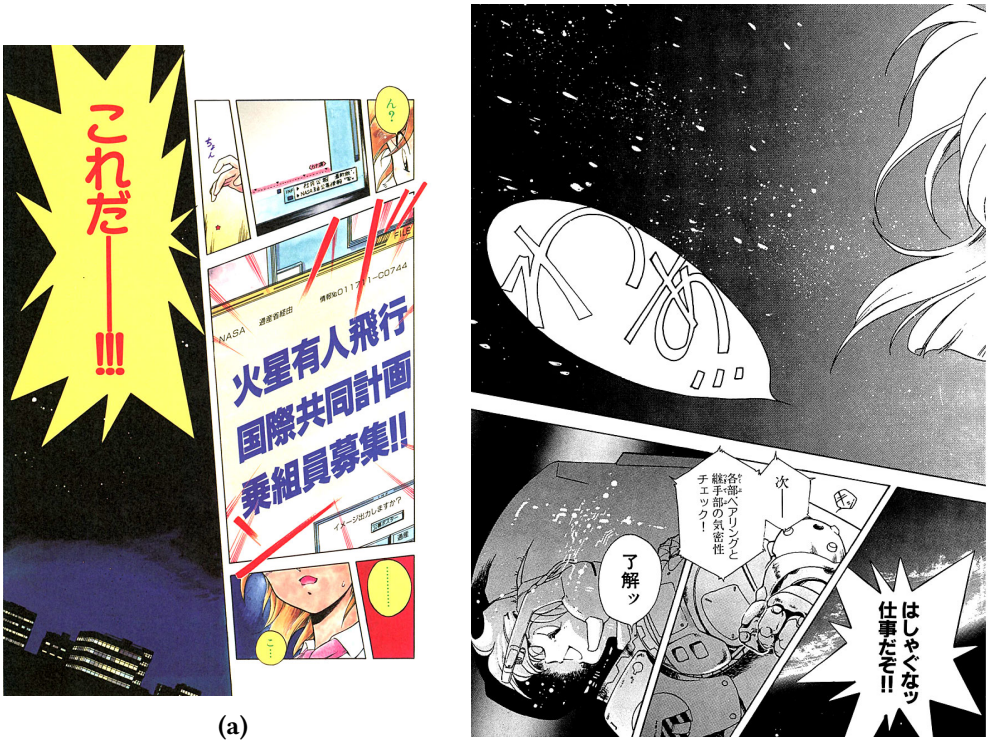
List of Tables

Contents

Chapter 1

Introduction

Supervised machine learning algorithms can be defined as optimization problems (Vapnik, 1998), in which weights are updated to minimize a certain loss function. Traditional models usually take as input some numeric characteristics of a particular example and also a target for those values. It is a quite hard task selecting what take into consideration as features when designing such a model, especially when the data are images (Coates *et al.*, 2011).



(a) (b)

Figure 1.1: *Illustration of the variance in pages from the same comic (EvaLady).*

Deep learning architectures are intended to solve this complex question. In essence, these networks have the capacity of learning what is relevant in each example (Good-

fellow *et al.*, 2016), updating specific weights for each abstract feature detected and learned. Particular examples of a deep learning architecture are the Convolutional Neural Networks (CNNs) (LeCun *et al.*, 2015). Although CNNs came to light during the 1980's (Fukushima and Miyake, 1982), only recently there is good enough computer machinery to turn this architecture attractive, popular, and suitable to employ in different fields (Anthes, 2013).

Besides the hardware has evolved during the last decades, it is also valuable that the algorithms become lighter, faster, and more portable (Wang *et al.*, 2017). This enhancement is important to employ these programs in different contexts, like devices that do not have great processing or energy power, as those part of the Internet of Things or even smartphones.

In this work, we study the impact of varying some parameters of a CNN, namely U-Net, in the context of few training examples. To analyze the effects of the changes, we chose to apply the network on the problem of text segmentation in manga, the Japanese comic-book story style. More specifically, we want to train a classifier that predicts which pixels belong to the class "text" and which do not.

In a context where there is a growth in the number of publications of these comics online (Fujimoto *et al.*, 2016), text segmentation enables further character recognition, topic identification, sentiment analysis, and it can also help with translation of these comic-books. Each page from these books is represented as an image. The challenge is that there is great variability of fonts, scales, graphical elements, and orientations, as can be seen in figure 1.1. Additionally, we provide an incremental parameter tuning, explaining explicitly the choices made. Thus, the main goal of this work is to discuss how each parameter affects the learning, not to build an optimal network that solves the aforementioned problem. To achieve that, we have used images from the Manga109 (Fujimoto *et al.*, 2016) data set.

This text is organized as follows: in chapter 2, we discuss some important concepts to the understanding of this study; in chapter 3, we bring and comment on the related works; following up, in chapter 4, we present the experiments we have done and the achieved results; lastly, in chapter 5, we discuss the conclusions we can take from the metrics observed.

Chapter 2

Background

2.1 Image representation

Images are computationally represented as matrices. In the RGB system, each image color is formed by the overlay of tonalities from three different colors: red, green, and blue. Each one of these colors is represented by a matrix (or channel). The resulting colored image is then constituted of a stack of these three same-sized matrices, just like the schema on figure 2.1, where each value at a matrix serves as the contribution of that color intensity to the final color. Moreover, each matrix entry usually ranges from 0 to 255.

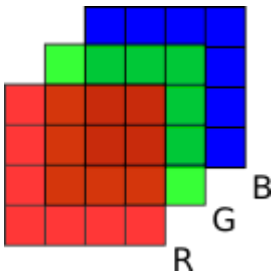


Figure 2.1: *RGB matrix representation.*

On the other hand, grayscale images are built of just one matrix, in which each value represents how "white" is the pixel color. In other words, if a pixel has value 0, its color is black; otherwise, if it is 255, then the color is white. Between 0 and 255, there are shades of gray. Although there are many methods to convert an RGB image into a grayscale one, in this project, we used images that were already gray, i.e., images that had only one channel.

2.2 Segmentation problem

Segmenting an image means breaking this entity into several disjoint parts. This segmentation can be done by several methods, like SLIC (Achanta *et al.*, 2012), watershed (Tarabalka *et al.*, 2010), components connected identification, binarization, among many

others. Each of these methods groups pixels in its semantic unity, i.e., a segment could be a part of a letter, a letter, a group of letters, and so on. An example of a segmentation done by the SLIC algorithm is shown in figure 2.2.

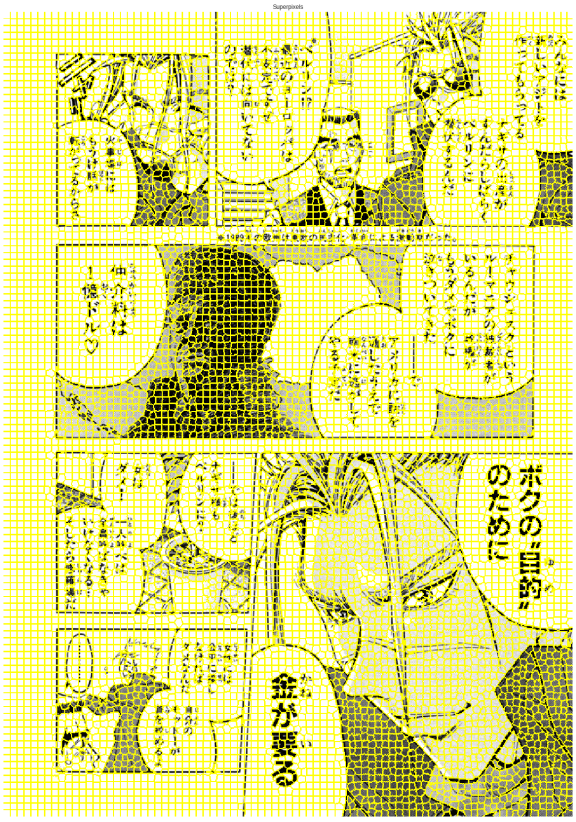


Figure 2.2: *Segmentation done by the SLIC algorithm.*

The segmentation problem can also be seen as a classification problem, in which, a class is assigned to each of the many segments of an image. This modeling allows us to represent the problem mathematically. The classification is done by a local function Ψ, centered in a certain pixel k, that iterates a window W of size $p \times p$ with its center through the image and maps it into a number, which will be the value of the central pixel on the output image. An exaggerated schema is display on figure 2.3.

This local function is what will enable a machine learning approach to solve this problem. As it will be further detailed, this function has weights that will be adjusted to allow it to make the best mapping to the output image. Moreover, in this case, the output binary image has the same size as the input and only one channel. Each entry can have the values 0 if the input pixel is not part of a character, or 1 if the input pixel is part of a text.

2.3 Data

The comics used in this work are from the Manga109 data set, which compiles 109 pieces from Japanese artists that were published between 1970 and 2010 (Fujimoto *et al.*, 2016). Only four comics were selected to train and evaluate the models: AosugiruHaru (AH),

Figure 2.3: *Local function schema.*

EvaLady (EL), JijiBabaFight (JBF), and MariaSamaNihaNaisyo (MSNN). Detailed information (Aizawa Yamasaki Matsui Laboratory, n.d.) about each comic can be checked on table 2.1.

Title	Author	Age	Publisher	Target	Genre
AH	Okuda Momoko	2000's	Shueisha	lady	love romance
EL	Miyone Shi	1990's	Takeshobo	boy	science fiction
JBF	Nishikawa Shinji	1990's	Kodansha	boy	humor
MSNN	Konohana Akari	1990's	Shueisha	lady	love romance

Table 2.1: *Data about the four comics used in this work.*

Many images were discarded because they had atypical text formatting (for example, summary, cover, and credits pages) or even because they did not have any text. RGB images were also excluded, remaining only the grayscale examples. The resting number of pages by title is on table 2.2.

Title	Number of pages
AH	105
EL	185
JBF	20
MSNN	192

Table 2.2: *Number of pages per title.*

For each title, statistics regarding the number of pixels belonging to a certain class or metrics about the components connected (area, height, and width) were calculated from the remaining pages. The standard deviation (std) of the components connected (CCs) metrics was taken over the averages per image of those values. Those statistics can be seen on table 2.3.

Title	Metric	Pixels "non-text"	Pixels "text"	CCs	Area	Height	Width
AH	Average	961839.491	5750.510	81.471	77.799	12.676	11.451
	Std	4422.614	4422.614	65.271	38.304	3.223	3.228
EL	Average	951005.254	16584.746	191.914	132.274	13.578	12.187
	Std	16036.411	16036.411	189.274	188.759	5.645	4.859
JBF	Average	955582.100	12007.900	261.100	46.717	9.807	8.190
	Std	4365.898	4365.898	100.150	5.827	0.899	0.568
MSNN	Average	957618.492	9971.508	135.417	160.300	14.404	12.180
	Std	7277.946	7277.946	76.126	620.957	17.424	11.759

Table 2.3: *Statistics about the number of pixels and number, area, height, and width of connected components.*

From table 2.3, we see that MSNN has the biggest CCs, while JBF the smallest, considering pixel area. CCs from JBF are also more alike since the standard deviations of all the metrics (area, height, and width) are the smallest. Differently, the metrics from MSNN are more diffuse, since this comic has the highest standard deviations. JBF is also the comic that has more CCs in the set, whilst EL, the lowest.

2.4 Convolutional Neural Networks

A convolution consists of a window (or the kernel), computationally represented by a matrix, that slides its center over an image, calculating this center mapping (which, in this case, is a pixel) by the dot product between the values of this kernel with the pixel values that this window is hover (O'Shea and Nash, 2015). The kernel is also called filter.

Figure 2.4 depicts the first step of a 3x3 (the kernel size) convolution. The center of the kernel is highlighted in light red. This center of the filter is positioned in the pixel of the input image that is pointed up in light blue. The output of this first step is the value emphasized in light green in the feature map.

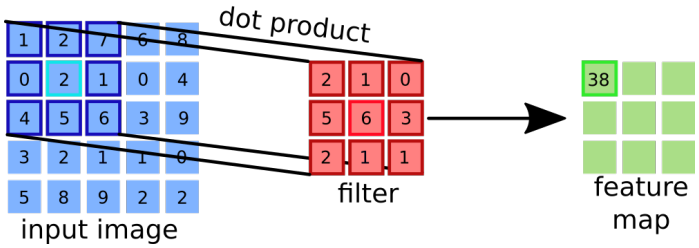


Figure 2.4: *Convolution schema.*

The values of the kernel's matrix are what we call weights. These numbers will be learned during the optimization of the loss function.

In a Convolutional Neural Network (CNN), usually, there are many stacked kernels (or filters) that convolve throughout the image. Each filter generates a feature map, that is a matrix, in which, each cell is the result of a dot product of the kernel centered in a certain pixel and the pixels within the window. As every filter has its own weights, it is commonly

said that each filter learns to detect particular characteristics of the input images, like, colors, or vertical edges, diagonal edges, and so on.

Differently from fully connected architectures, weights from different filters that are on the same "stack" are independent of each other. This is one of the factors that enables parallel processing in CNNs. A trade-off is that these networks tend to be more computationally costly to train.

2.5 Model

To learn the function Ψ_W, a supervised machine learning framework was used. More specifically, we elected a Convolutional Neural Network model to be trained on the data described in section 2.3. Specifically, given the $\mathcal{T}$ set of target binary images and the $\mathcal{I}$ set of input gray-scale images, the model should find the weights for the classifier that best generalize the predictions on never-seen examples. That is to say, the model will solve the following optimization problem:

$$\min_{weights} ||\Psi_W(k) - j||, \forall k \in \mathcal{I}, j \in \mathcal{J} \tag{2.1}$$

The CNN model used was a U-Net. This network has a contracting path followed by an expanding path, hence the "U" in the model's name. The contracting path is a sequence of pairs of convolutions with kernel size $3x3$ followed by a $2x2$ max-pooling (Ronneberger *et al.*, 2015). For the sake of understanding, let's call each pair of convolution followed by a max-pooling an encoding *plateau*.

In the meanwhile, the expanding path is composed of pairs of convolutions proceeded by an up-convolution, which is an up-sampling operation succeeded by a $2x2$ convolution (Ronneberger *et al.*, 2015). Similarly, let's name each pair of convolution and an up-convolution a decoding *plateau*.

The effect caused by each encoding *plateau* is to halve the image size, whereas, in each decoding plateau, the image size is doubled, reconstituting the original dimensions. Each encoding *plateau* is connected to its corresponding decoding pair by having its filters concatenated to first convolution's filters inside the decoding *plateau*. The number of encoding *plateaux* (or of decoding *plateaux*) is called the network's depth. In figure 2.5 there's a diagram that illustrates a depth-three model's organization. After each convolution, the activation ReLU is applied. Furthermore, still on figure 2.5, on the top of each blue bar, there are the numbers of filters from each convolution.

Although we have experimented with different parameters, we considered being our standard U-Net a model with depth 3, loss function Categorical Cross-Entropy, learning rate 10^{-4}, optimizer Adam, batch size of one image, patch size equals the full image and no normalization. In all experiments, this base model was used, except in where it was explicitly specified the opposite.

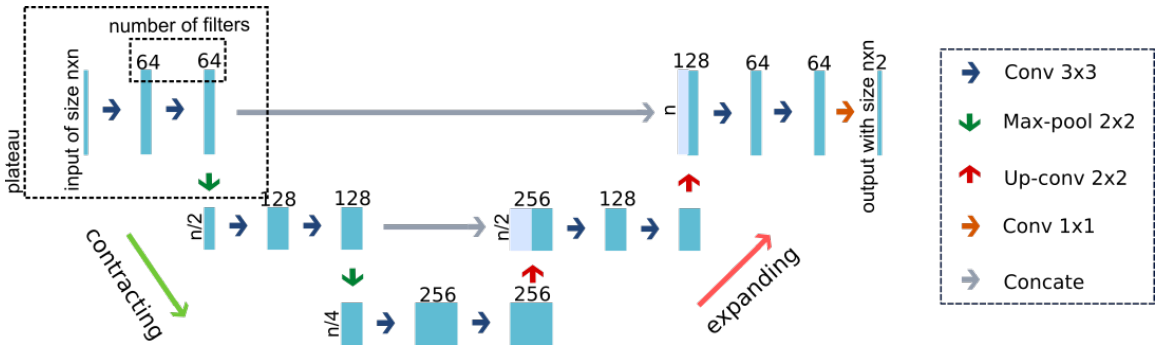


Figure 2.5: *U-Net diagram.*

2.6 Training and evaluating frameworks

Instead of using the whole available data to train and to evaluate the networks, we have built some subsets: 10 images with most CCs (D_1), 10 first pages of the story (D_2), 5 images with the most connected components (D_3). Additionally, it was randomly[1] selected 10 examples for evaluating the performance, from which 5 were used to appraise the model after each epoch (V_1 subset) and 5 others were used to assess the performance after the training (V_2 subset). The properties of these subsets are:

1. $D_3 \subset D_1$;
2. $V_1 \cap V_2 = \emptyset$;
3. $V_1 \cap D_1 = \emptyset$, $V_1 \cap D_2 = \emptyset$, $V_1 \cap D_3 = \emptyset$;
4. $V_2 \cap D_1 = \emptyset$, $V_2 \cap D_2 = \emptyset$, $V_2 \cap D_3 = \emptyset$.

The weights that achieved the least loss on the V_1 set were saved on an h5f5 file. After the training was completed, the best weights were reloaded to the model to evaluate it on V_2.

2.7 Experiments planning

In each experiment, we chose to vary only one aspect of the network or the training/evaluation schema. The best choices found at each stage were propagated to the next trials. The order of experiments was the following:

1. Training models with different depths;
2. Training models with different normalization techniques;
3. Training models with different loss functions;
4. Training models on different training subsets;
5. Training the same model on different comics;
6. Training models on a comic and evaluating on another;

[1]For random choices, it was used 42 as a fixed random seed for numpy operations.

7. Training with different numbers of examples.

2.8 Loss functions

The loss functions are the equations that measure how far or near are the predictions from the target outputs. Each loss function "sees" the differences in a particular way, thus, their outputs range in different intervals. The challenge arises when we have a very imbalanced dataset, which is the case. For that reason, we experimented with different loss functions. Some of them try to minimize the effects of the abundance of one particular class.

Among the loss functions we experimented are: Cross-Entropy loss (CE, eq. 2.2), (Lin *et al.*, 2017), Weighted Cross-Entropy loss (WCE, eq. 2.3) (Panchapagesan *et al.*, 2016), Generalized Dice loss (GDL, eq. 2.4) (Sudre *et al.*, 2017), and Focal loss (FL, eq. 2.5) (Lin *et al.*, 2017). In the next equations, p_n is the probability of the example be of class 1 (text), while t_n is the target class (0 or 1). Weights w_0 and w_1 are calculated with the inverse of frequency from each class in the generalized dice loss. Finally, in the focal loss, accordingly to the researchers (Lin *et al.*, 2017), α balances the importance of positive/negative examples, without differentiating easy/hard examples, and the easy examples are down weighted with a modulating factor $(1 - p_n)^\gamma$, with a focusing parameter γ.

$$CE = -\frac{1}{N}\sum_{n=1}^{N} t_n log(p_n) + (1 - t_n) log(1 - p_n) \tag{2.2}$$

$$WCE = -\frac{1}{N}\sum_{n=1}^{N} 0.45 \cdot t_n \cdot log(p_n) + 0.55 \cdot (1 - t_n) \cdot log(1 - p_n) \tag{2.3}$$

$$GDL = 1 - 2\frac{w^0 \sum_n t_n \cdot p_n + w^1 \sum_n (1 - t_n) \cdot (1 - p_n)}{w^0 \sum_n (t_n + p_n) + w^1 \sum_n [(1 - t_n) + (1 - p_n)]} \tag{2.4}$$

$$w^0 = \frac{1}{\left(\sum_n (1 - t_n)\right)^2}$$

$$w^1 = \frac{1}{\left(\sum_n t_n\right)^2}$$

$$FL = -\alpha\frac{1}{N}\sum_{n=1}^{N} t_n (1 - p_n)^\gamma log(p_n) + (1 - t_n)(p_n)^\gamma log(1 - p_n) \tag{2.5}$$

2.9 Metrics

The models have their performance evaluated on training, validation, and testing sets through metrics, which have different semantics. In this work, we have used accuracy (Sokolova *et al.*, 2006), precision (Davis and Goadrich, 2006) and recall (Davis and Goadrich, 2006). The accuracy expresses the proportion of correctly classified (positives and negatives) examples in the set. Precision measures the proportion of correctly classified

positive examples, taking into account the total of examples that were predicted as positives. In the meanwhile, recall measures the percent of correctly classified positive examples, among truly positives examples present in the set. Let true positives, true negatives, false positives, and false negatives be, respectively, TP, TN, FP, and FN. So, these metrics can be defined as:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.6}$$

$$precision = \frac{TP}{TP + FP} \tag{2.7}$$

$$recall = \frac{TP}{TP + FN} \tag{2.8}$$

Chapter 3

Related works

Bounding boxes are the most common ground-truth annotation for text detection in mangas. A common idea explored for text detection is to build region proposals and then classify them as text or non-text. To build region proposals, Aramaki *et al.*, 2016, first classify connected components as text or non-text based on geometrical features and then group components classified as text into rectangular regions. Then these regions are classified as text or non-text using features extracted with pre-trained deep networks and SVM. Chu and Yu, 2018 employ object detection methods commonly used in generic computer vision tasks, and report results comparable or superior to the ones in Aramaki *et al.*, 2016.

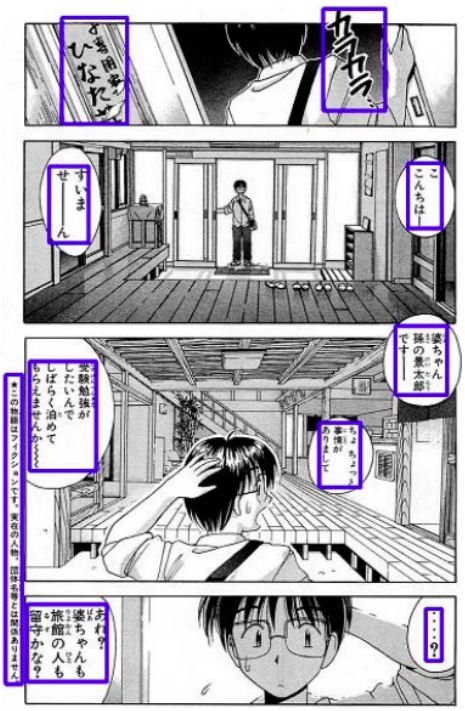


Figure 3.1: *Example of bounding box annotation used in Aramaki et al., 2016 as ground-truth.*

Region proposals are obtained using the selective search algorithm. Features extracted

with CNNs are used for classification and also for regression (to learn the region offsets). Authors report that using this method, a similar performance to the ones reported in Aramaki *et al.*, 2016 were achieved, and even better performance was obtained using the well known Faster-RCNN model.

In Piriyothinkul *et al.*, 2019, final text regions are built by first applying the Stroke Width Transform on the input image to find letter candidates, then classifying them as a letter or non-letter using SVM, and finally grouping the components classified as letters in text regions based on heuristic criteria, and better results than the ones in Aramaki *et al.*, 2016 are reported. These works all use up to 100 pages for training and 100 pages for evaluation from 6 manga titles in the Manga109 data set (although not the same pages), following the evaluation method of the first work.

Pixel-level segmentation is less explored (Hirata *et al.*, 2016; Ko and Cho, 2020). In Hirata *et al.*, 2016 a method for pixel-level segmentation modeled as an image transformation problem and computed based on a sliding window approach is presented, however without any quantitative evaluation. In Ko and Cho, 2020, a method for inpainting text region in mangas is proposed. This method includes segmentation of typeset as well as hand-drawn text as an intermediary step to help to remove the text before inpainting. A U-Net-like architecture is used, with a total of 200 training, 57 validation, and 28 test images. Since there are no images annotated at the pixel level, one part of the training data was generated relying on the available bounding box annotations and the other part was manually segmented. The performance was evaluated in terms of the IoU metric since only bounding-box ground-truth was available.

According to the authors, two loss functions, weighted cross-entropy, and Jaccard, were tested and the best performance was obtained with the latter.

U-Net-like models are also used for speech balloon detection Dubray and Laubrock, 2019, which is an effective way of detecting text when most of them are located within speech balloons.

Chapter 4

Experiments and results

4.1 Baseline performance

Before varying several aspects, we have built a standard model and studied its performance. For this baseline model, we have used a U-Net with depth 3, loss function Categorical Cross-Entropy, batch size equals one image, learning rate 10^{-4}, optimizer Adam, no normalization, and D_1 of EvaLady as the training set. The learning curve during 100 epochs is on figure 4.1.

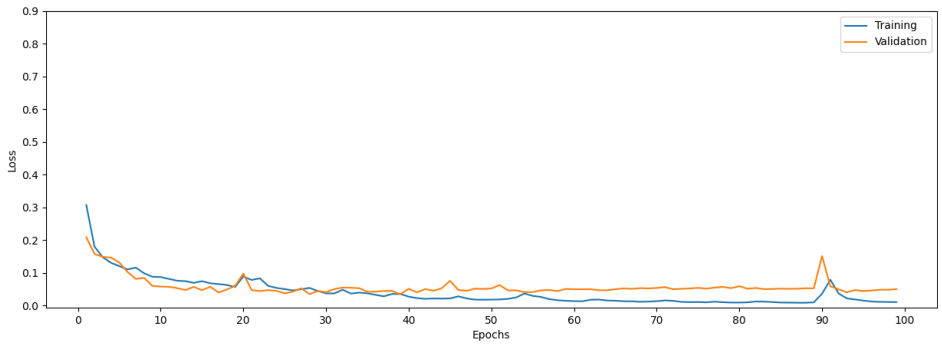


Figure 4.1: *Learning curve of the baseline model.*

As can be seen, the model overfits around epoch 30, that's when the training loss continues decreasing while the validation curve keeps stable. It is also important to note that the training is converging, i.e., the weights are being optimized, as the loss is coming down and approximating to zero.

4.2 Training models with different depths

The first step was to vary the model's depth, the other arguments remained the same. This parameter determines the number of convolutions and max-pooling in the contracting path (or the number of convolutions and up-sampling in the expanding path). Strictly

speaking, the deeper the network, the more parameters to learn. We have experimented with depths 1, 2, 3, 4, and 5. The behaviour observed for the training during 100 epochs is on figure 4.2 and in the table 4.1.

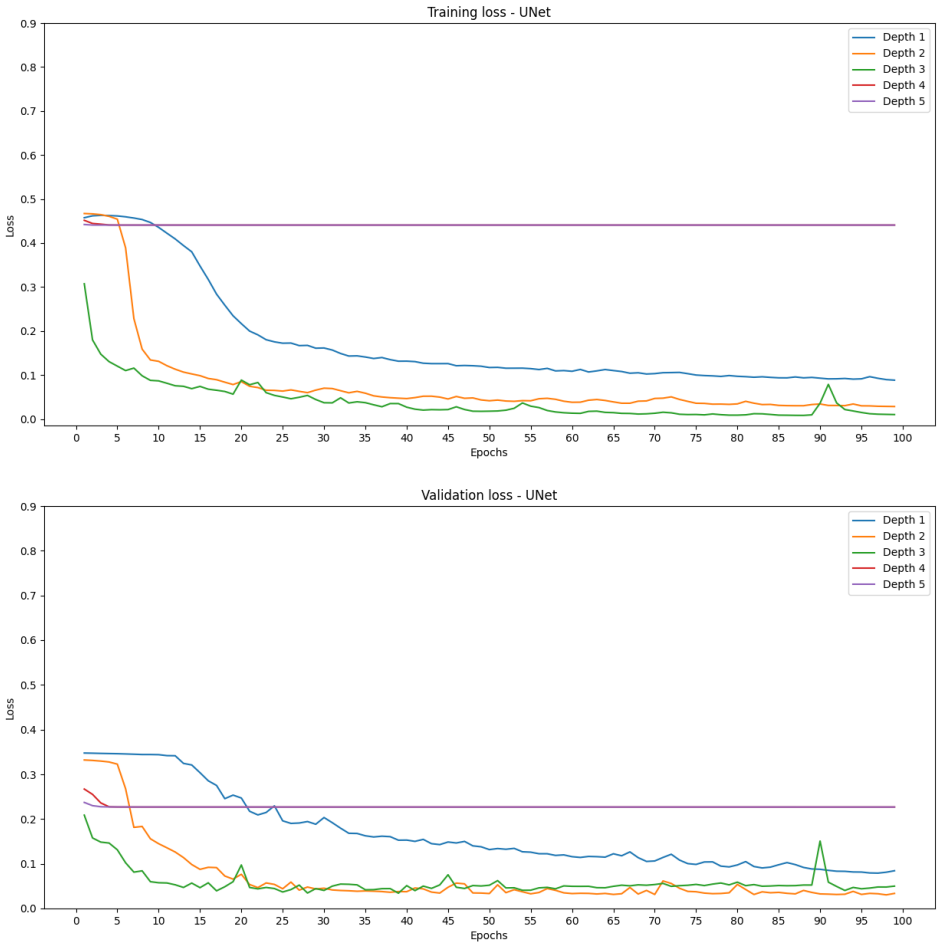


Figure 4.2: *Learning curves of depths with no normalization.*

From the learning curves (fig. 4.2) that converged, we see that depth 1 presented the worst result, while depth 3 the best. For depths 4 and 5, the curves did not come down, despite the slight fall in the very beginning. These results can also be seen in the validation after the training (tab. 4.1), where depth 3 performed the best accuracy and precision. Depths 4 and 5 probably learned to classify everything as negative ("non-text"), which explains the NaN precision and zero recall. Thus, depth 3 appears to be an adequate size for the model to generalize the examples from the training.

depth	seconds elapsed during training	min loss on training	min loss on V_1	accuracy on V_2	precision on V_2	recall on V_2
1	2361.793	0.088	0.079	0.988	0.433	0.193
2	10835.544	0.028	0.031	0.994	0.735	0.670
3	15516.197	0.008	0.034	0.993	0.753	0.578
4	19371.455	0.441	0.227	0.989	-	0.000
5	22481.266	0.441	0.227	0.989	-	0.000

Table 4.1: *Metrics of different depths with no normalization*

4.3 Training models with different normalization techniques

Following up, we tested different normalization schemes for the already mentioned five depths. Input normalization, batch normalization, and both techniques combined were the normalization schemas experimented.

4.3.1 Input normalization

With this normalization, every pixel intensity of the inputs is converted to the $[0, 1]$ interval. In order to do so, all values are divided by the maximum intensity, which is 255. Other parameters without exception were the same from the previous step. The learning curve for this case is on figure 4.5.

depth	seconds elapsed during training	min loss on training	min loss on V_1	accuracy on V_2	precision on V_2	recall on V_2
1	2298.055	0.084	0.055	0.989	0.484	0.020
2	10439.136	0.034	0.025	0.993	0.826	0.516
3	16006.453	0.009	0.035	0.990	0.932	0.090
4	19428.981	0.008	0.041	0.995	0.834	0.689
5	22557.089	0.006	0.029	0.997	0.941	0.763

Table 4.2: *Metrics of different depths with input normalization.*

In this scenario, every training converged (fig. 4.3). We can see that the bigger the model, in terms of learnable parameters, the lower the training loss achieved, although this does not reflect on validation on V_1 loss. Even though the model with depth 5 had the best metrics on V_2 validation, it was 40% slower than depth 3 to train (tab. 4.2). Similar to the case of depth 4, which is the second-best and is 21% slower than depth 3, in terms of time in seconds to learn. On the other side, depth 3 also achieved a 99% accuracy.

4.3.2 Batch normalization

This normalization is done by a layer in Keras[1] applied after every ReLU activation. The values from each filter are normalized by subtracting the mean and dividing by the

[1]Keras is the python framework used to build the model. More info: https://keras.io/

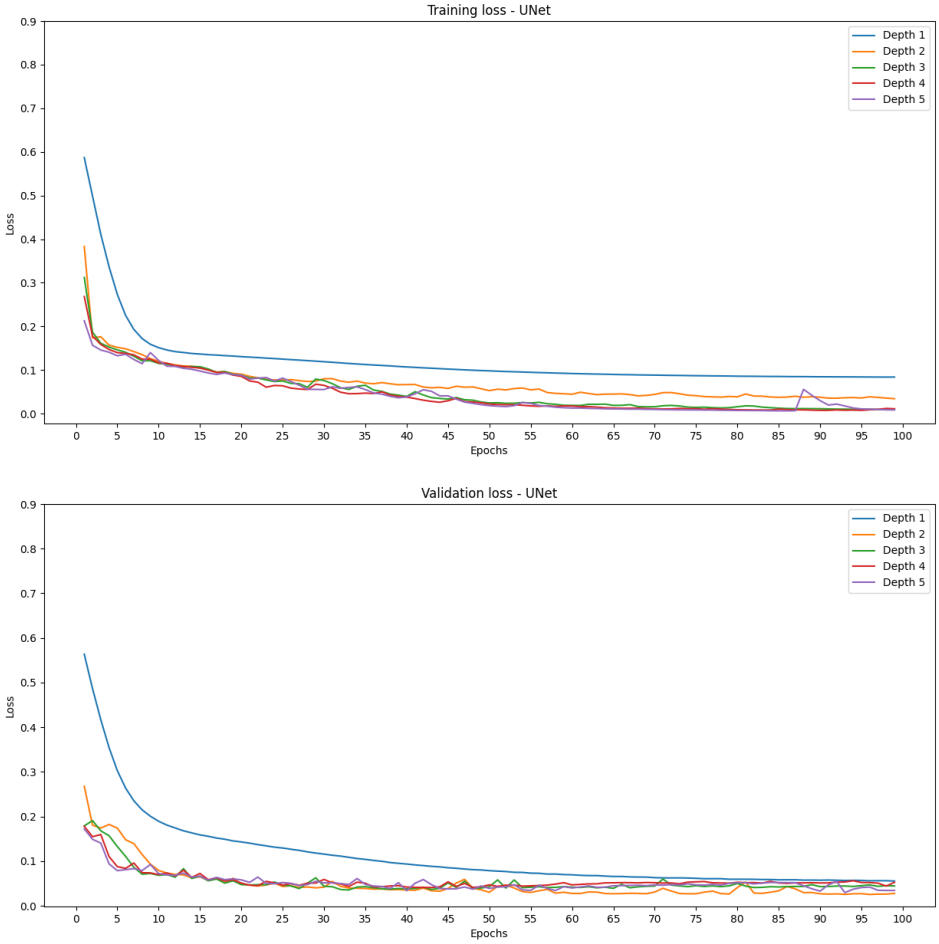


Figure 4.3: *Learning curves of depths with input normalization.*

standard deviation, like on equation 4.1.

$$\text{intensity normalized} = \frac{intensity - mean}{std} \tag{4.1}$$

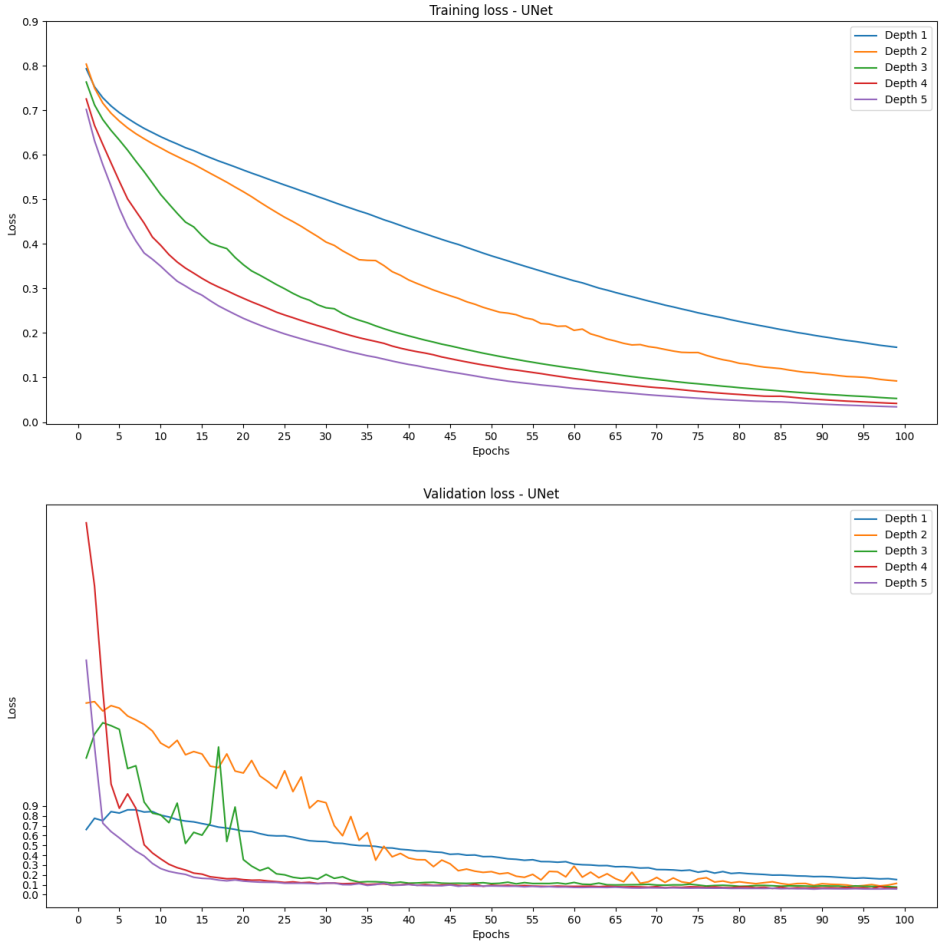


Figure 4.4: *Learning curves of depths with batch normalization.*

Again, all curves converged (fig. 4.4). However, the loss decrease was slower. In this normalization scheme, we can see clearly that, the bigger the model, the lower the loss achieved during training. In this case, this behavior also happens in the validation on V_1 loss curve, i.e., bigger models also achieved lower V_1 loss.

On the other hand, training with batch normalization turned the learning very slower, compared to normalization only on the inputs. To be precise, it was 53% slower for depth 1 and 30% for depth 5 (tabs. 4.2 and 4.3).

Another important fact to mention is that recall improved for every depth, while the

depth	seconds elapsed during training	min loss on training	min loss on V_1	accuracy on V_2	precision on V_2	recall on V_2
1	3531.379	0.168	0.154	0.988	0.456	0.475
2	15855.795	0.092	0.086	0.996	0.843	0.747
3	22294.151	0.053	0.077	0.997	0.864	0.857
4	26279.252	0.042	0.060	0.998	0.917	0.869
5	29429.710	0.034	0.057	0.997	0.912	0.797

Table 4.3: *Metrics of different depths with batch normalization.*

precision and accuracy had no significant difference. This hints that the model was more penalized for guessing everything as "non-text", making it correctly classify more positive examples (tab. 4.3).

4.3.3 Batch and input normalization

In this scenario, we tried to combine both normalizations previously mentioned to check if they could improve the metrics. The results can be checked on figure 4.5 and table 4.4.

depth	seconds elapsed during training	min loss on training	min loss on V_1	accuracy on V_2	precision on V_2	recall on V_2
1	3345.867	0.179	0.177	0.987	0.450	0.485
2	15382.582	0.092	0.079	0.996	0.919	0.664
3	22022.136	0.054	0.073	0.997	0.926	0.836
4	26358.042	0.046	0.060	0.998	0.937	0.873
5	29570.680	0.035	0.054	0.997	0.920	0.752

Table 4.4: *Metrics of different depths with input and batch normalization.*

As we can see, the curves were very similar to the ones with only batch normalization, despite some oscillations. Even though precision improved for every depth, it was not a very wide increase. The other metrics did not change significantly, including the seconds elapsed to train.

4.4 Training models on different training subsets

As explained in section 2.6, we made three subsets to use them as training sets: D_1, with the top 10 images with most CCs; D_2 with the first 10 pages from the comics; D_3 with the top 5 images with most CCs. The model that was used was the same as described in section 4.1, but with input normalization.

Although the $D3$, which has fewer examples, contributes to slightly worse performance, there is no significant difference between the subsets $D1$ and $D2$ (fig. 4.6). In terms of validation loss, after 50 epochs, all the models achieve almost the same minimum loss on V_1. Furthermore, all models have achieved a 99% accuracy (tab. 4.5).

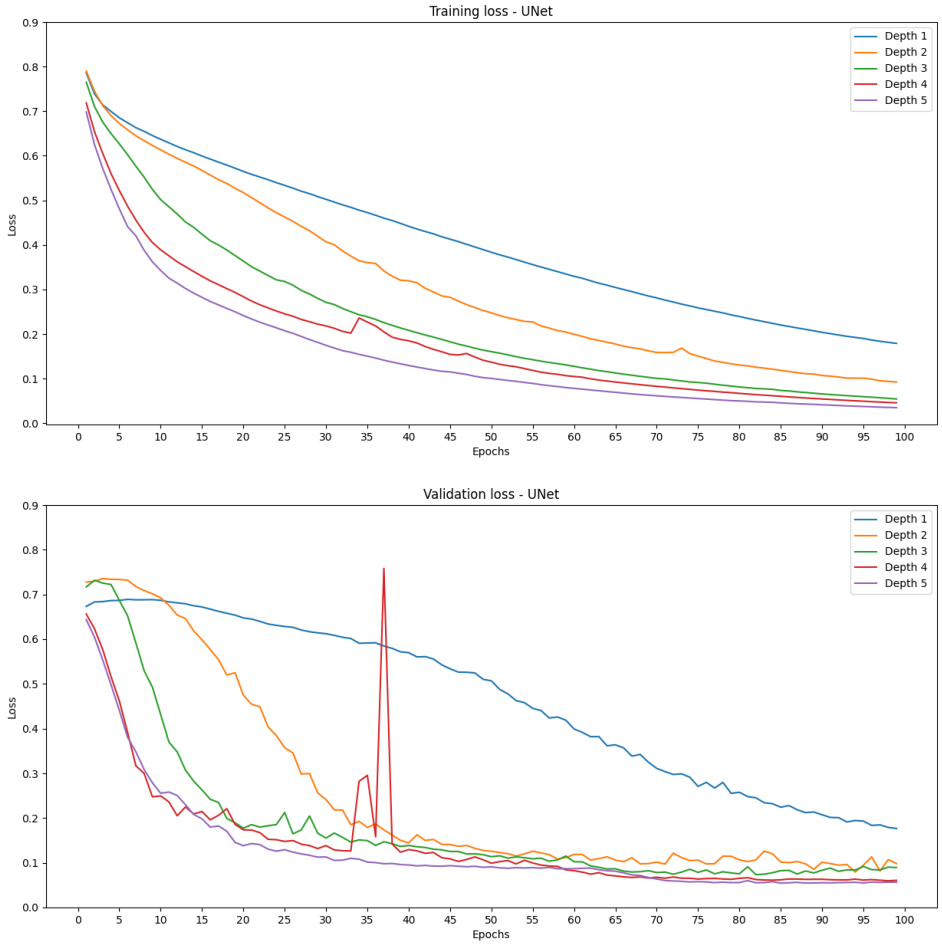


Figure 4.5: *Learning curves of depths with input and batch normalization.*

subset	seconds elapsed during training	min loss on training	min loss on V_1	accuracy on V_2	precision on V_2	recall on V_2
D1	15549.119	0.009	0.035	0.990	0.932	0.090
D2	15698.767	0.007	0.042	0.994	0.777	0.692
D3	8627.472	0.020	0.035	0.992	0.907	0.342

Table 4.5: *Metrics of different training subsets.*

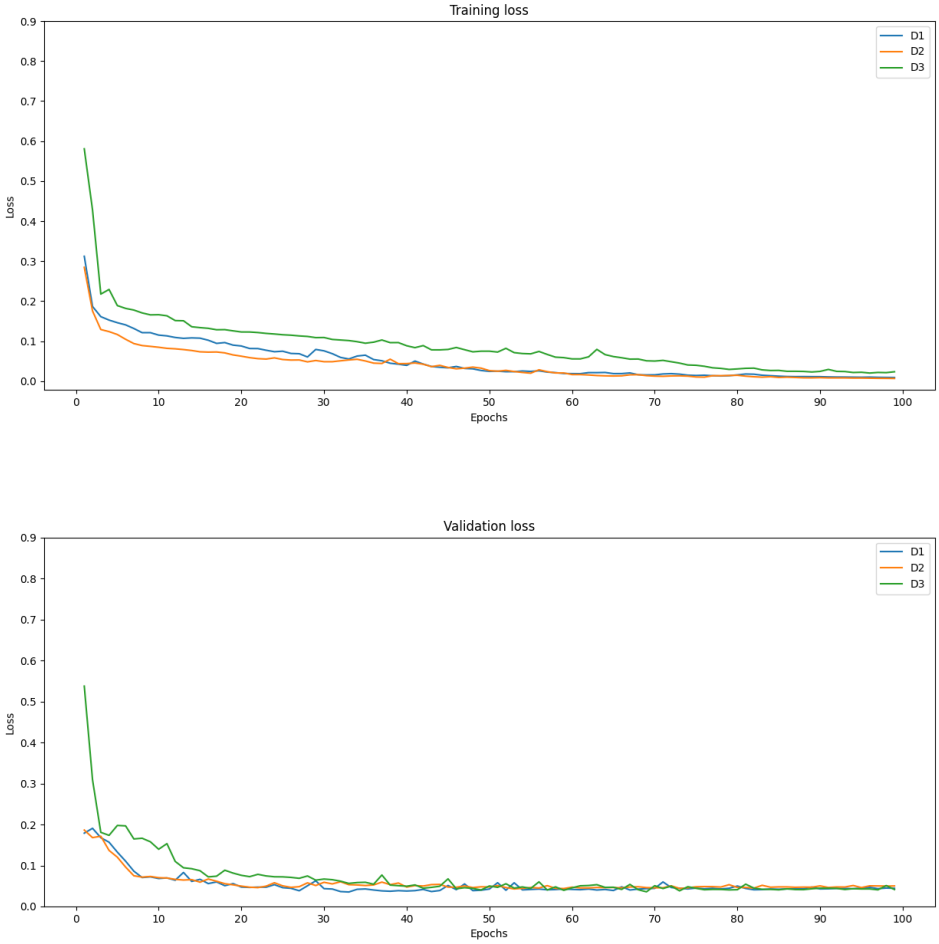


Figure 4.6: *Learning curves of different training subsets.*

4.5 Training models with different loss functions

Like it was said in section 2.8, the loss function is the mathematical expression that measures how far or near the predictions are from the expected outputs. The model, as already mentioned, tries to minimize these functions by updating the weights. What differs from one function to another is how the formula "views" and expresses this distance.

In this work, we have experimented with the functions: Generalized Dice Loss (GDL), Weighted Cross-Entropy (WCE), Cross-Entropy (CE), and Focal Loss (FL). As each metric ranges on its own domain, in this section, we chose not to display the learning curves of each loss functions, as they are not directly comparable. However, some metrics on V_2, which are comparable, due to be on the same scale, are on table 4.6.

loss function	seconds elapsed during training	min loss on training	min loss on V_1	loss on V_2	accuracy on V_2	precision on V_2	recall on V_2
GDL	15729.872	0.065	0.263	0.156	0.997	0.880	0.823
WCE	15926.927	0.005	0.019	0.142	0.991	0.886	0.215
CE	15549.119	0.009	0.035	0.028	0.990	0.932	0.090
FL	15591.891	0.001	0.002	0.033	0.994	0.778	0.635

Table 4.6: *Metrics of different loss functions.*

From the table 4.6, we see that the model trained with the GDL function has the best performance, with the highest accuracy and recall and very high precision. That means that, even though there is a high imbalance in the data set, the model can be generalized well on never seen examples, and also that it does not tend to predict only the most frequent class.

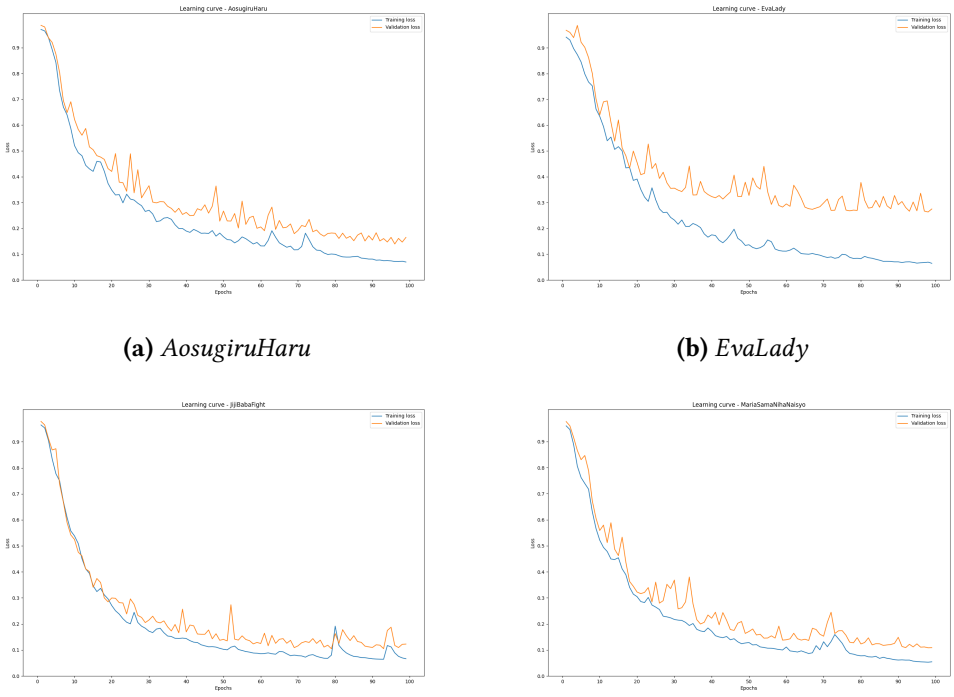
We also notice that different loss functions do not affect significantly the seconds spent to train the network, as this metric is similar for all losses. The difference of time taken to train between the fastest and slowest is approximately 6 minutes.

4.6 Training the same model on different comics

Following up, we tested the standard model with GDL and input normalization on four comics from Manga109. Learning curves are on figure 4.7 and there are some metrics on table 4.7.

comic	seconds elapsed during training	min loss on training	min loss on V_1	loss on V_2	accuracy on V_2	precision on V_2	recall on V_2
EL	15626.725	0.065	0.263	0.156	0.997	0.880	0.823
AH	15576.702	0.069	0.139	0.149	0.999	0.792	0.941
JBF	15606.121	0.064	0.104	0.091	0.998	0.890	0.943
MSNN	15660.370	0.053	0.108	0.276	0.996	0.656	0.820

Table 4.7: *Metrics of different comics.*



(a) *AosugiruHaru* **(b)** *EvaLady*

(c) *JijiBabaFight* **(d)** *MariaSamaNihaNaisyo*

Figure 4.7: *Learning curves of different comics.*

We see that JBF, which has smaller (in terms of area) and more CCs (in absolute number), has the best performances in terms of precision and recall on V_2. On the other hand, MSNN, which has the biggest CC's average area, has the worst performance in terms of precision and recall.

4.7 Training models on a comic and evaluating on another

We also trained the aforementioned model with loss GDL and input normalization on a comic and evaluated on another to analyze the behavior. From table 4.8, we see that, as expected, the models best perform when trained and tested on the same comic.

Aggregating the values per training comic, and averaging them by only consider when the models are evaluated on comics different than the used on training, we see that EL and MSNN have the highest mean accuracy; JBF has the highest mean precision; EL has the highest mean recall. It is also important to recall that JBF has the CCs with the smallest areas.

Trained on	Tested on	Loss on V_2	Accuracy on V_2	Precision on V_2	Recall on V_2
EL	EL	0.089	0.996	0.857	0.813
EL	AH	0.207	0.998	0.735	0.821
EL	JBF	0.215	0.996	0.677	0.956
EL	MSNN	0.133	0.995	0.700	0.815
AH	EL	0.167	0.992	0.772	0.577
AH	AH	0.144	0.998	0.839	0.853
AH	JBF	0.157	0.997	0.814	0.883
AH	MSNN	0.152	0.994	0.738	0.660
JBF	EL	0.169	0.992	0.848	0.513
JBF	AH	0.202	0.998	0.867	0.702
JBF	JBF	0.091	0.998	0.890	0.943
JBF	MSNN	0.152	0.995	0.805	0.593
MSNN	EL	0.129	0.994	0.867	0.639
MSNN	AH	0.235	0.998	0.799	0.676
MSNN	JBF	0.228	0.996	0.745	0.810
MSNN	MSNN	0.097	0.996	0.844	0.786

Table 4.8: *Metrics of cross-evaluations.*

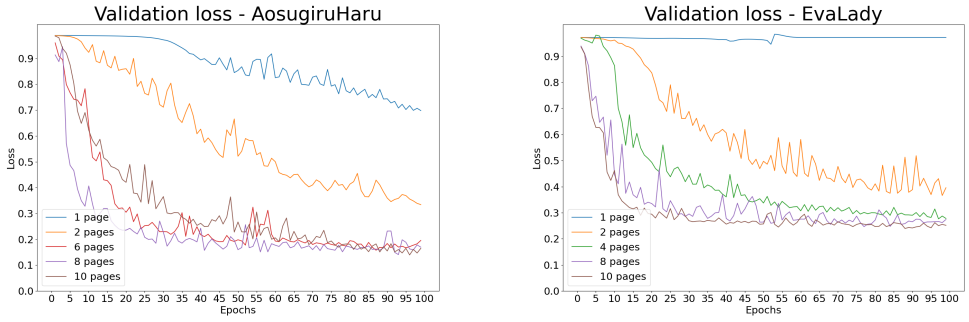
4.8 Training models with different number of examples

Lastly, we experimented with varying the number of training examples used to learn. We tested training with 1 to 10 images. In figure 4.8, not all curves are plotted due to lack of convergence of the training. Probably this divergence happens because of a stochastic characteristic of the learning process. However, we could not isolate hypotheses to confirm that.

Considering the cases where the training converged and the loss decreased, we can see from table 4.9 that the more examples, the higher are the metrics on V_2, as expected. Another fact observed is that sometimes a number of training examples allowed the model to converge on one comic, but not on another. This hints us the random behavior of the training.

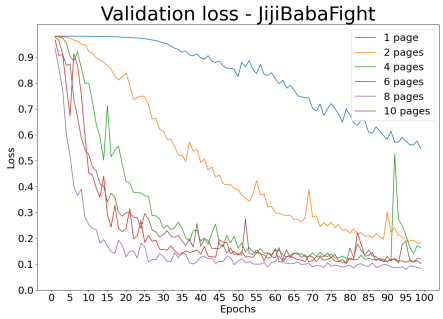
comic	number of training examples	seconds elapsed during training	min loss on V_1	loss on V_2	accuracy on V_2	precision on V_2	recall on V_2
EL	1	2607.570	0.946	0.961	0.936	0.102	0.602
EL	2	4039.904	0.366	0.281	0.995	0.814	0.683
EL	3	5489.714	0.296	0.199	0.996	0.869	0.761
EL	4	6958.265	0.276	0.195	0.996	0.860	0.769
EL	5	8382.049	0.565	0.593	0.982	0.332	0.556
EL	6	9825.591	0.791	0.840	0.896	0.089	0.895
EL	7	11346.325	0.964	0.939	0.937	0.025	0.120
EL	8	12812.275	0.247	0.147	0.997	0.911	0.809
EL	9	14263.480	0.718	0.780	0.952	0.144	0.667
EL	10	15726.339	0.240	0.150	0.997	0.937	0.782
AH	1	2669.277	0.698	0.710	0.994	0.278	0.321
AH	2	4174.078	0.334	0.313	0.997	0.612	0.854
AH	3	5785.595	0.912	0.966	0.859	0.022	0.853
AH	4	6958.837	0.953	0.981	0.996	0.191	0.039
AH	5	8406.155	0.911	0.962	0.956	0.039	0.458
AH	6	9916.660	0.158	0.143	0.999	0.803	0.936
AH	7	11351.098	0.154	0.147	0.999	0.782	0.953
AH	8	12803.125	0.141	0.138	0.999	0.779	0.973
AH	9	14284.857	0.123	0.116	0.999	0.817	0.973
AH	10	15658.323	0.880	0.945	0.969	0.061	0.503
JBF	1	2613.289	0.546	0.580	0.990	0.449	0.424
JBF	2	4065.148	0.182	0.194	0.997	0.848	0.808
JBF	3	5524.634	0.107	0.106	0.998	0.885	0.924
JBF	4	6972.755	0.111	0.094	0.998	0.903	0.928
JBF	5	8444.949	0.091	0.069	0.999	0.926	0.947
JBF	6	9880.643	0.102	0.076	0.999	0.899	0.961
JBF	7	11358.774	0.111	0.109	0.998	0.887	0.913
JBF	8	12810.470	0.080	0.058	0.999	0.948	0.945
JBF	9	14264.388	0.074	0.060	0.999	0.924	0.964
JBF	10	15632.639	0.937	0.930	0.956	0.029	0.121
MSNN	1	2583.022	0.779	0.872	0.970	0.080	0.346
MSNN	2	4033.580	0.344	0.501	0.992	0.421	0.660
MSNN	3	5495.866	0.168	0.329	0.995	0.618	0.751
MSNN	4	6940.666	0.923	0.948	0.870	0.035	0.729
MSNN	5	8386.843	0.128	0.424	0.993	0.459	0.779
MSNN	6	9855.683	0.116	0.264	0.997	0.730	0.750
MSNN	7	11366.824	0.096	0.211	0.997	0.787	0.797
MSNN	8	12756.922	0.096	0.230	0.997	0.723	0.829
MSNN	9	14272.052	0.094	0.231	0.997	0.734	0.810
MSNN	10	15705.678	0.094	0.198	0.997	0.801	0.806

Table 4.9: *Metrics of different numbers of training examples per each comic.*

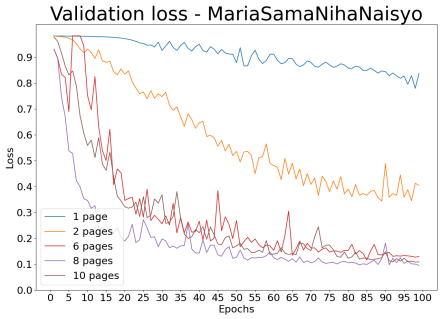


(a) *AosugiruHaru*

(b) *EvaLady*



(c) *JijiBabaFight*



(d) *MariaSamaNihaNaisyo*

Figure 4.8: *Learning curves of different numbers of training examples per each comic.*

Chapter 5

Conclusion

In this work, we investigated the impact of varying some parameters and aspects of a convolutional neural network, namely U-Net, in a context with few examples. We applied our networks on the text segmentation problem of mangas. Among the aspects of the U-net, we experimented with the model depth, normalization techniques, loss functions, different training subsets, different comics, and a different number of examples. We also conducted a cross-evaluation between models trained on a comic and assessed on another.

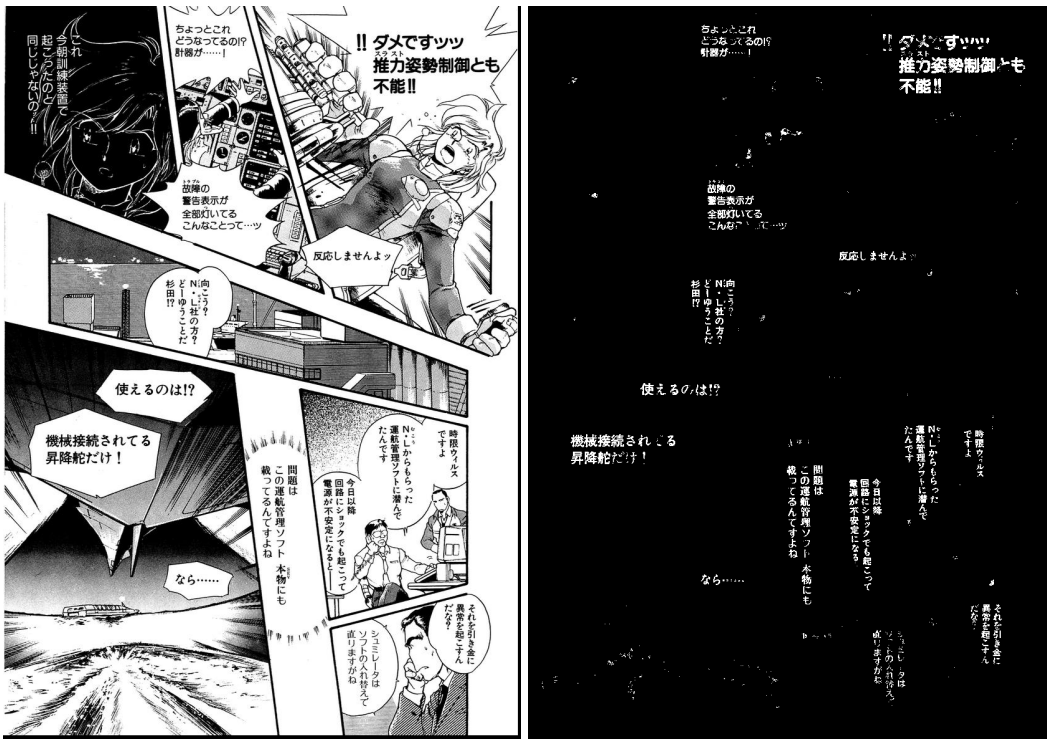
We observe that a U-net of depth 3 is sufficient to efficiently generalize the problem of segmentation of text in mangas, achieving good metrics and not taking so long to train. Moreover, input normalization can help the converge of the training, while the improvements brought by batch normalization do not compensate for the worsening in the convergence speed. Combining both normalizations did not bring any relevant improvements.

About using the first 10 pages or the 10 pages with the most connected components (and training the model with input normalization), we did not notice any relevant improvement, especially after 50 training epochs. Regarding the loss functions, we see that, in a very imbalanced data set like ours, the generalized dice loss appeared to be the most adequate function to explain the difference between the prediction and the expected and, thus, the most suitable to be minimized by the model, as this is the function that allowed the network to achieve the highest accuracy and recall, and a very elevated precision.

When training the model on different comics, we notice a relationship between the average size of the connected components (CCs) and their precision and recall: JBF, which has smaller CCs, has the best precision and recall; whereas MSNN, which has the biggest CCs, has the worst performance in terms of the same metrics. We also trained the model on a comic and evaluated it on another, which presented a very good performance in every case.

Finally, we also varied the number of training examples, from 1 to 10. Although not all training curves converged, probably due to a stochastic behavior, we observe that, in general, the more examples, the higher are the metrics, as expected.

In conclusion, although maybe this is not the optimal network for this problem, we



(a) *Input image* **(b)** *Output image*

Figure 5.1: *Example of prediction made by a network build in this study.*

have conducted a parameter investigation and justified the choices made based on observed metrics, which we believe yields a strong baseline, as it can be seen in figure 5.1.

The possible future works are: use different sizes of patches for training and check their relation to the smoothness of the validation curve and training convergence; comparison of the bounding box and pixel-wise approaches in terms of performance and processing cost; increase the number of target classes. The networks build in this project should be good enough to guide the development of solutions for these questions.

References

[Achanta *et al.* 2012] Radhakrishna Achanta *et al.* "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282 (cit. on p. 3).

[Aizawa Yamasaki Matsui Laboratory n.d.] Aizawa Yamasaki Matsui Laboratory. *Project Manga109 website*. URL: http://www.manga109.org/ (cit. on p. 5).

[Anthes 2013] Gary Anthes. "Deep learning comes of age". In: *Communications of the ACM* 56.6 (2013), pp. 13–15 (cit. on p. 2).

[Aramaki *et al.* 2016] Y. Aramaki, Y. Matsui, T. Yamasaki, and K. Aizawa. "Text detection in manga by combining connected-component-based and region-based classifications". In: *IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 2901–2905 (cit. on pp. 11, 12).

[Chu and Yu 2018] W. Chu and C. Yu. "Text Detection in Manga by Deep Region Proposal, Classification, and Regression". In: *IEEE Visual Communications and Image Processing (VCIP)*. 2018, pp. 1–4 (cit. on p. 11).

[Coates *et al.* 2011] Adam Coates *et al.* "Text detection and character recognition in scene images with unsupervised feature learning". In: *2011 International Conference on Document Analysis and Recognition*. IEEE. 2011, pp. 440–445 (cit. on p. 1).

[Davis and Goadrich 2006] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 233–240 (cit. on p. 9).

[Dubray and Laubrock 2019] D. Dubray and J. Laubrock. "Deep CNN-Based Speech Balloon Detection and Segmentation for Comic Books". In: *International Conference on Document Analysis and Recognition (ICDAR)*. 2019, pp. 1237–1243 (cit. on p. 12).

[Fujimoto *et al.* 2016] Azuma Fujimoto *et al.* "Manga109 dataset and creation of metadata". In: *Proceedings of the 1st international workshop on comics analysis, processing and understanding*. 2016, pp. 1–5 (cit. on pp. 2, 4).

[Fukushima and Miyake 1982] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285 (cit. on p. 2).

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016 (cit. on p. 1).

[Hirata *et al.* 2016] N. S. T. Hirata, I. S. Montagner, and R. Hirata Jr. "Comics image processing: learning to segment text". In: *Proc. of the 1st International Workshop on coMics ANalysis, Processing and Understanding*. ACM, 2016, 11:1–11:6 (cit. on p. 12).

[Ko and Cho 2020] U-Ram Ko and Hwan-Gue Cho. "SickZil-Machine: A Deep Learning Based Script Text Isolation System for Comics Translation". In: *Document Analysis Systems*. Ed. by Xiang Bai, Dimosthenis Karatzas, and Daniel Lopresti. Cham: Springer International Publishing, 2020, pp. 413–425 (cit. on p. 12).

[LeCun *et al.* 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444 (cit. on p. 2).

[Lin *et al.* 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988 (cit. on p. 9).

[O'Shea and Nash 2015] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015) (cit. on p. 6).

[Panchapagesan *et al.* 2016] Sankaran Panchapagesan *et al.* "Multi-task learning and weighted cross-entropy for DNN-based keyword spotting." In: *Interspeech*. Vol. 9. 2016, pp. 760–764 (cit. on p. 9).

[Piriyothinkul *et al.* 2019] B. Piriyothinkul, K. Pasupa, and M. Sugimoto. "Detecting Text in Manga Using Stroke Width Transform". In: *International Conference on Knowledge and Smart Technology (KST)*. 2019, pp. 142–147 (cit. on p. 12).

[Ronneberger *et al.* 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on p. 7).

[Sokolova *et al.* 2006] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation". In: *Australasian joint conference on artificial intelligence*. Springer. 2006, pp. 1015–1021 (cit. on p. 9).

[Sudre *et al.* 2017] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations". In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248 (cit. on p. 9).

[Tarabalka *et al.* 2010] Yuliya Tarabalka, Jocelyn Chanussot, and Jon Atli Benediktsson. "Segmentation and classification of hyperspectral images using watershed transformation". In: *Pattern Recognition* 43.7 (2010), pp. 2367–2379 (cit. on p. 3).

[Vapnik 1998] Vladimir N. Vapnik. *Statistical learning theory*. New York: Wiley-Interscience, 1998 (cit. on p. 1).

[Wang *et al.* 2017] Yunhe Wang, Chang Xu, Chao Xu, and Dacheng Tao. "Beyond filters: Compact feature map for portable deep model". In: *International Conference on Machine Learning*. 2017, pp. 3703–3711 (cit. on p. 2).