Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Dataset Generation for the Training of Neural Networks Oriented toward Histological Image Segmentation

Supervisor:
Dr. Enrico Giampieri

Submitted by:
Alessandro d'Agostino

Academic Year 2019/2020

Acknowledgements:

Abstract

Abstract.....

Contents

Introduction

In the last decades, the development of Machine Learning (ML) and Deep Learning (DL) techniques has contaminated every aspect of the scientific world, with interesting results in many different research fields. The biomedical field is no exception to this and a lot of promising applications are taking form, especially as Computer-Aided Detection (CAD) systems which are tools coming in support to physicians during the diagnostic process. Medical doctors and the Healthcare system in general collect a huge amount of data from patients during all the treatment, screening, and analysis activities in many different shapes, from anagraphical data, to blood analysis to clinical images.

In medicine the study of images is ubiquitous and countless diagnostic procedures rely on it, such as X-ray imaging (CAT), nuclear imaging (SPECT, PET), Magnetic resonance, and visual inspection of histological specimens after biopsies. The branch of Artificial Intelligence in the biomedical field that handles image analysis to assist physicians in their clinical decisions goes under the name of Digital Pathology Image Analysis (DPIA). In this thesis work, I want to focus on some of the beneficial aspects introduced by DPIA in the histological images analysis and some particular issues in the development of DL models able to handle this kind of procedure.

Nowadays the great majority of analysis of histological specimens occurs through visual inspection, carried out by highly qualified experts. Some analysis, as cancer detection, requires the ability to distinguish if a region of tissue is healthy or not with high precision in very wide specimens. This kind of procedure is typically very complex and requires prolonged times of analysis besides substantial economic efforts. Furthermore, the designated personnel for this type of analysis is often limited, leading to delicate issues of priority assignment while scheduling analysis, based on the estimated patient's clinical development. Some sort of support to this analysis procedure is therefore necessary.

The problem of recognizing regions with different features within an image and detect their borders is known in computer vision as segmentation task, and it's quite spread in many different applications, allowing a sort of automatic interpretation of the image. The segmentation problem is usually faced as a supervised task, hence the algorithm in order to be trained properly requires a reasonable quantity of pre-labeled images, from which learn the rules through which distinguish different regions. This means that the development of segmentation algorithms for a specific application, as would be the

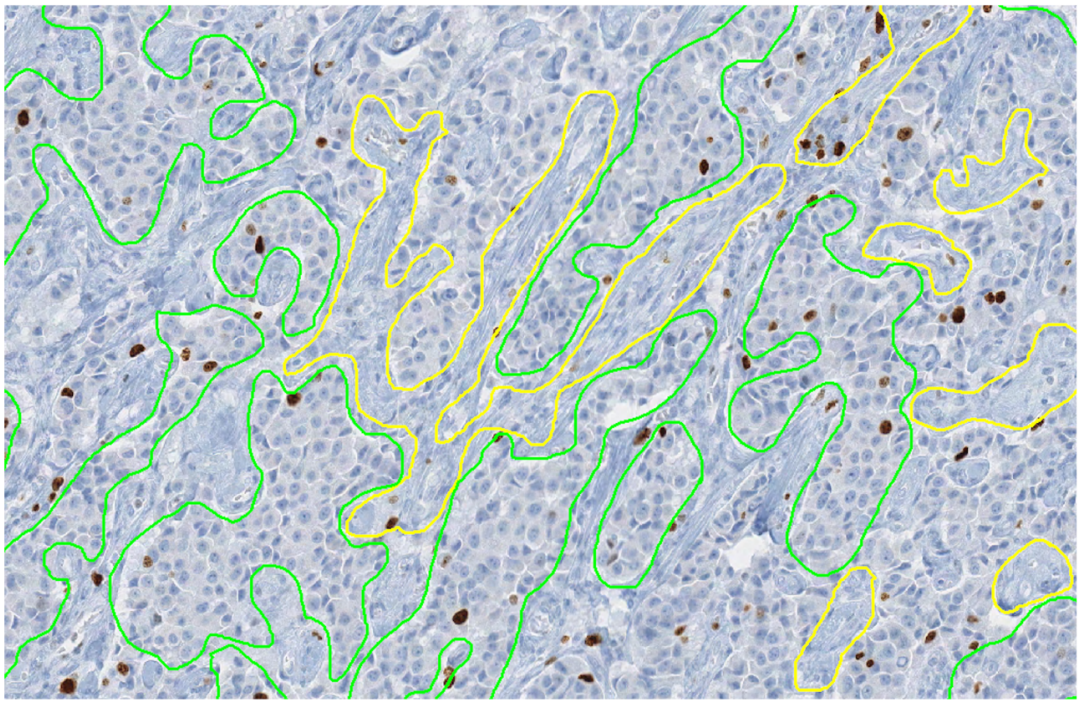

Figure 1: *Interleaving of tumor (green annotation) and non-tumor (yellow annotation) regions [7].*

one with histological images, requires a lot of starting material independently analyzed from the same qualified expert encharged of the visual inspection I mentioned before. A human operator thus is required to manually track the boundaries, for example, between healthy and tumoral regions within a sample of tissue and to label them with their identity, as in Figure 1. The more the algorithm to train is complex the more starting material is required to adjust the model's parameters and reach the desired efficacy.

The latest developed segmentation algorithms are based on DL techniques, hence based on the implementation of intricated Neural Networks (NN) which process the input images end produces the correspondent segmentation. Those models are typically very complex, with millions of parameters to adjust and tune, therefore they need a huge amount of pre-labeled images to learn their segmentation rules. This need for data is exactly the main focus of my thesis work. The shortage of ground truth images is indeed one of the toughest hurdles to overcome during the development of DL-based algorithms. Another important aspect to bear in mind is the quality of the ground truth material. It's impossible for humans to label boundaries of different regions with pixel-perfect precision, while for machines the more precise is the input the more tuned is the resulting algorithm.

There have already been explored different approaches to overcome this problem, and

they are mainly based on the generation of synthetic data to be used during the training phase. Some techniques achieve data augmentation manipulating already available images and then generating "new" images, but as we will see later this approach suffers from different issues. The technique that I propose in this work follows a generation from scratch of entire datasets suitable for the training of new algorithms, based on the 3D modelization of a region of human tissue at the cellular level. The sectioning of the virtual histological samples yields the synthetic images with their corresponding ground truth. Using this technique one would be able to collect sufficient material for the training (the entire phase or the preliminary part) of a model, avoiding then the shortage of hand-labeled data.

The 3D modeling of a region of particular human tissue is a very complex task, and it is almost impossible to capture all the physiological richness of a histological system. The models I implemented thus are inevitably schematic in their representation of the target biological structures. I'll show two models: one of pancreatic tissue and another of epidermic tissue, besides all the tools I used and the choices I made during the design phase.

In order to present organically all the steps of my work the thesis is organized in chapters as follows:

1. Structure
2. Of the
3. Thesys

Chapter 1

Histo & Deep Learning & SOA

1.1 Histological Images

Description of followed Approach

1.1.1 Traditional Preparation of Histological Samples

How to prepare a sample

1.1.2 Important Aspects

Problems with hand-labeling Important aspects in final images

1.2 Introduction to Deep Learning

Deep Learning is part of the broader framework of Machine Learning and Artificial Intelligence. Indeed all the problems typically faced using ML can also be addressed with DL techniques, for instance, regression, classification, clustering, and segmentation problems. We can think of DL as a universal methodology for iterative function approximation with a great level of complexity. In the last decades, this technology has seen a frenetic diffusion and an incredible development, thanks to the always increasing available computational power, and it has become a staple tool in all sorts of scientific applications.

1.2.1 Perceptrons and Multilayer Feedforward Architecture

As other artificial learning techniques, those models aim to "learn" a relationship between some sort of input and a specific kind of output. In other words, approximating numeri-

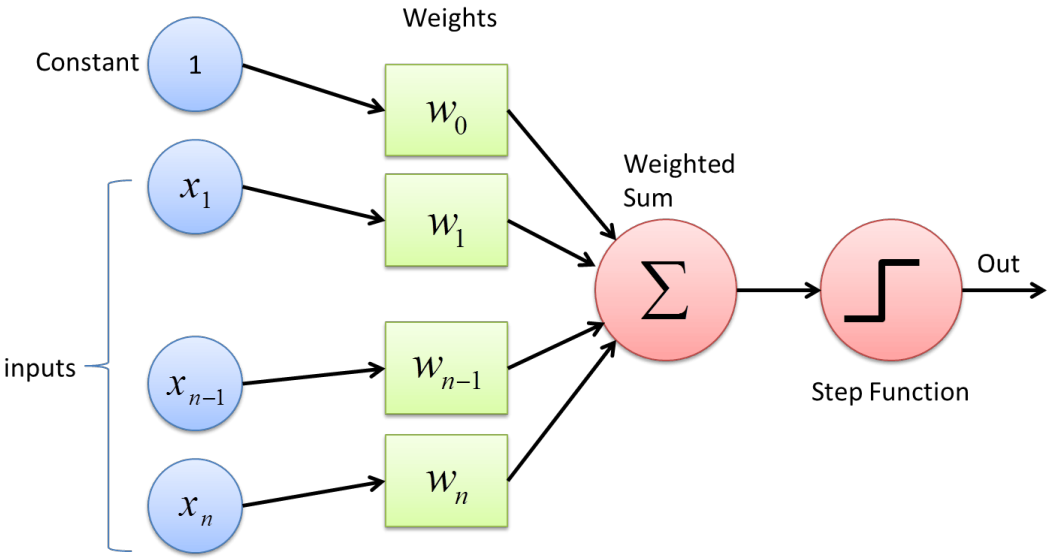

Figure 1.1: *Schematic picture of a single layer perceptron. The input vector is linearly combined with the bias factor and sent to an activation function to produce the numerical output.*

cally the function that processes the input data and produces the desired response. For example, one could be interested in clustering data in a multidimensional features space, or in the detection of objects in a picture, or in text manipulation and generation. The function is approximated by means of a greatly complex network of simple linear and non-linear mathematical operations arranged in a so-called Neural Network (typically with millions of parameters). In fact, the seed idea behind this discipline is to recreate the functioning of actual neurons in the human brain: their entangled connection system and their "ON/OFF" behavior [11].

The fondamental unit of a neural network is called perceptron, and it acts as a digital counterpart of a human neuron. As shown in Figure 1.1 a perceptron collects in input a series on n numerical signals $\vec{x} = 1, x_1, ..., x_n$ and computes a linear wieghted combination with the weights vectors $\vec{w} = w_0, w_1, ..., w_n$, where w_0 is the bias vector:

$$f(\vec{x}, \vec{w}) = \chi(\vec{x} \cdot \vec{w}) \tag{1.1}$$

The results of this linear combination is given as input to a non-linear function $\chi(x)$ called activation function. Typical choices as activation function are any sigmoidal function like $sign(x)$ and $tanh(x)$, but in more aadvanced applications other functions like ReLU [1] are used. The resulting function $f(\vec{x}, \vec{w})$ has then a simple non linear behaviour. It produces a binary output: 1 if the weighted combination is high enough, 0 otherwise.

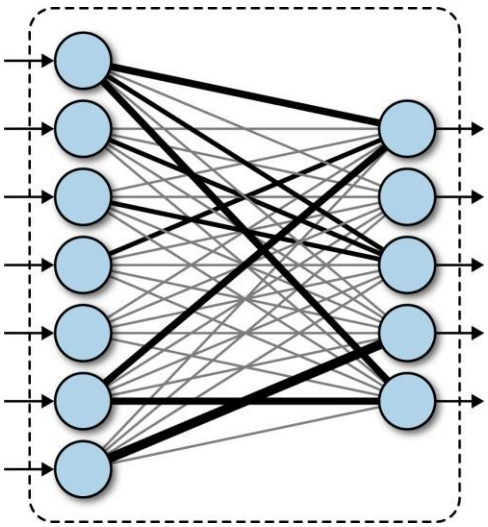

Figure 1.2: *Schematic representation of a fully connected (dense) layer. Every neuron from the first layer is connected with every output neuron. The link thickness represent the absolute value of the combination weight for that particular value.*

The most common architecture for a NN is the so-called feed-forward architecture, where many individual perceptrons are arranged in chained layers, which take as input the output of previous layers along the information flux. More complex architectures could implements also recursive connection, linking a layer to itself, but it should be regarded as an exception to the standard case. There are endless possibilities of combination and arrangement of neurons inside a NN but the most simple one is known as fully connected layer, where every neuron is linked with each toher neuron of the following layer, as shown in Figure 1.2. Each connection has its weight, which contributes to modulate the overall combination of signals. The training of a NN consists then in the adjustment and fine-tuning of all the Network's weights and parameters through iterative techniques until the desired precision is reached in the output generation.

Although a fully connected network represents the simplest linking choice, the insertion of each weight increases the number of parameters, and so the complexity of the model. Thus we want to create links between neurons smartly, avoiding the less useful ones. Depending on the type of data under analysis there are many different established typologies of layers. For example, in the image processing field, the most common choice is the convolutional layer, which implements a sort of discrete convolution on the input data, as shown in Figure 1.3. While processing images, the convolution operation confers to the perception some kind of correlation between adjacent pixels of an image and their color channels, allowing a sort of spatial awareness.

As a matter of principle a NN with just two successive layers, which is called a *shallow*

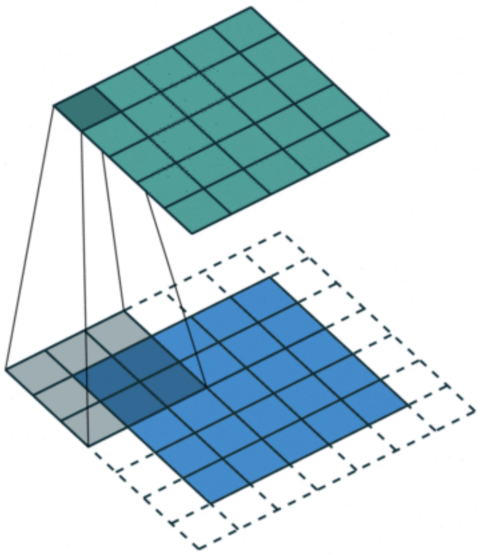

Figure 1.3: *Schematical representation of a convolutional layer. The input data are processed by a window kernel that slides all over the image.*

network, and with an arbitrary number of neurons per layer, can approximate arbitrary well any kind of smooth enough function [8]. However, direct experience suggests that Networks with multiple layers, called *deep* networks, can reach equivalent results exploiting a lower number of parameters overall. This is the reason why this discipline goes under the name of *deep* learning: it focuses on deep networks with up to tents hidden layers. Such a deep structure allows the computation of which are called deep features, so features of the features of the input data, that allows the network to easily manage concepts that would be bearly understandable for humans.

1.2.2 Training of a NN - Error Back-Propagation

Depending on the task the NN is designed for, it will have a different architecture and number of parameters. Those parameters are initialized to completely random values, tough. The training process is exactly the process of seeking iteratively the right values to assign to each parameter in the network in order to accomplish the task. The best start to understanding the training procedure is to look at how a supervised problem is solved. In supervised problems, we start with a series of examples of true connections between inputs and correspondent outputs and we try to generalize the rule behind those examples. After the rule has been picked up the final aim is to exploit it and to apply it to unknown data, so new problem could be solved. In opposition to the concept of supervised problems there are the *unsupervised* problems, where the algorithm do not try

Figure 1.4: *Sample grid of images from the CIFAR10 dataset. Each one of the 32 × 32 image is labeled with one of the ten classes of objects: plane, car, bird, cat, deer, dog, frog, horse, ship, truck.*

to learn a rule from practical example but try to devise it from scratch. A task typically posed as unsupervised is clustering, when different data are separated in groups based on the values of their features in the feature space. Usually only the number of groups is taken in input from the algorithm, and the subdivision is completely performed by the machine. In the real world by the way, there are many different and sophisticated shapes between pure supervised and pure unsupervised learning, based on the actual availability of data and specific limitations to the individual task.

A good example of supervised problems is the classification of images. Let's assume we have a whole dataset of pictures of different objects as cats, dogs, cars, etc. like the CIFAR10 [4] dataset. This famous dataset is made of over $60K$ labeled images 32×32 divided into 10 categories of objects as shown in Figure 1.4. We could be interested in the creation of a NN able to assign at every image its belonging class. This NN could be arbitrarily complex but it certainly will take as input a $32\times32\times3$ RGB image and the output will be the predicted class. A typical output for this problem would be a probability distribution over all the 10 classes like:

$$\vec{p} = (p_1, p_2, \dots, p_{10}), \tag{1.2}$$

$$\sum_{i=1}^{10} p_i = 1, \tag{1.3}$$

and it should be compared with the truth, that is represented just as a binary sequence $\vec{t}$ with the bit correspondent to the belonging class set as 1, and all the others value set to 0:

$$\vec{t} = (0, 0, \ldots, 1, \ldots, 0, 0). \tag{1.4}$$

Every time an image is given to the model an estimate of the output is produced. Thus, we need to measure the 'distance' between that prediction and the true value, to quantify the error made by the algorithm and try to improve the model's predictive power. The functions used for this purpose are called loss functions. The most common choice is the Mean Squared Error function that is simply the averaged L^2 norm of the difference vector between $\vec{p}$ and $\vec{t}$:

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(t_i - p_i)^2. \tag{1.5}$$

Let's say the NN we are training has L concecutive layers, each one with its activation function f^k and its wights vector $\vec{w}^k$, hence the prediction vector $\vec{p}$ could be seen as the result of the concesutive, nested, application through all the layers:

$$\vec{p} = f^L(\vec{w}^L \cdot (f^{L-1}(\vec{w}^{L-1} \cdot ... \cdot f^1(\vec{w}^1 \cdot \vec{x})))). \tag{1.6}$$

From both 1.5 and 1.6 it is clear that the loss function could be seen as a function of all the weights vectors of every layer of the network. So if we want to reduce the distance between the NN prediction and the true value we need to modify those weights to minimize the loss function. The most established algorithm to do so for a supervised task in a feed-forward network is the so-called Error Back-Propagation. The backpropagation method works essentially computing the gradient of the loss function with respect to the weights using the derivative chain rule and updating by a small amount the value of each parameter to lower the overall loss function. Each weight is *moved* counter-gradient, and summing all the contribution to every parameter the loss function approaches its minimum. In equation 1.7 is represented the variation applied to the j^{th} weight in the i^{th} layer:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \tag{1.7}$$

where E is the error function, and η is the *learning coefficient*, that modulate the effect of learning through all the training process. This iterative procedure is applied completely to each image in the training set several times, each time is called an *epoch*. The great majority of the dataset is exploited in the training phase to keep running this trial and error process and just a small portion is left out (typically 10% of the data) for a final performance test.

The loss function shall inevitably be differentiable, and its behavior heavily influences the success of the training. If the loss function presents a gradient landscape rich of local minima it's very probable that the gradient descent process would get stuck in one of them. More sophisticated algorithms capable of avoiding this issue have been devised, with the insertion of some degree of randomness in them, as the Stochastic Gradient Descent algorithm, or the wide used *Adam* optimizer [3].

The training phase is the pulsing heart of a DL model development and it could take even weeks on top-level computers for the most complicated networks. In fact, one of the great limits to the complexity of a network during the designing phase is exactly the available computational power. There are many more further technical details necessary for proper training, the adjustment of which can heavily impact on the quality of the algorithm.

However, after the training phase, we need to test the performance of the NN. This is usually done running the trained algorithm on never seen before inputs, the test dataset, and comparing the prediction with the ground-truth value. A good way to evaluate the quality of the results is to use the same function used as loss function during the training, but there is no technical restriction to the choice of this quality metric. The average score on the whole test set is then used as a numerical score for the network, and it allows straightforward comparison with other models, trained for the same task.

All this training procedure is coherently customized to every different application, depending on which the problem is posed as supervised or not and depending on the more or less complex network's architecture. The leitmotif is always finding a suitable loss function that quantifies how well the network does what it has been designed to do and trying to minimize it, operating on the parameters that define the network structure.

1.3 Deep Learning-Based Segmentation Algorithms

In digital image processing, image segmentation is the process of recognizing and subdividing an image into different regions of pixels that show similar features, like color, texture, or intensity. Typically, the task of segmentation is to recognize the edges and boundaries of the different objects in the image and assigning a different label to every detected region. The result of the segmentation process is an image with the same dimensions of the starting one made of solid color regions, representing the detected objects. This image is called *segmentation mask*. In Figure 1.5 is shown an example of segmentation of a picture of an urban landscape: different colors are linked to different classes of objects like persons in magenta and scooters in purple. This technology has a significant role in a wide variety of application fields such as scene understanding, medical image analysis, augmented reality, etc.

A relatively easy problem and one of the first to be tackled could be distinguishing an object from the background in a grey-scale image. The easiest technique to per-

Figure 1.5: *Example of the resulting segmentation mask of an image of an urban landscape. Every interesting object of the image is detected and a solid color region replaces it in the segmentation mask. Every color corresponds to a different class of objects, for example, persons are highlighted in magenta and scooters in purple. The shape and the boundaries of every region should match as precisely as possible the edges of the objects.*

form segmentation in this kind of problem is based on thresholding. Thresholding is a binarization technique based on the image's grey-level histogram: to every pixel with luminosity above that threshold is assigned the color *white*, and vice versa the color *black*. However, this is a very primitive and fallacious yet very fast method, and it manages poorly complex images or images with un-uniformity in the background.

Many other traditional techniques improve this first segmentation method. Some are based on the object's edges recognition, exploiting the sharp change in luminosity typically in correspondence of the boundary of a shape. Other techniques exploit instead a region-growing technology, according to which some *seed* region markers are scattered on the image, and the regions correspondent to the objects in the image are grown incorporating adjacent pixels with similar properties.

1.3.1 State of the Art on Deep Learning Segmentation

Similarly to many other traditional tasks, also for segmentation, there has been a thriving development lead by the diffusion of deep learning, that boosted the performances resulting in what many regards as a paradigm shift in the field [6].

In further detail, image segmentation can be formulated as a classification problem of pixels with semantic labels (semantic segmentation) or partitioning of individual objects (instance segmentation). Semantic segmentation performs pixel-level labeling with a set of object categories (e.g. boat, car, person, tree) for all the pixels in the image, hence it is typically a harder task than image classification, which requires just a single label for the whole image. Instance segmentation extends semantic segmentation scope further by detecting and delineating each object of interest in the image (e.g. partitioning of individual nuclei in a histological image).

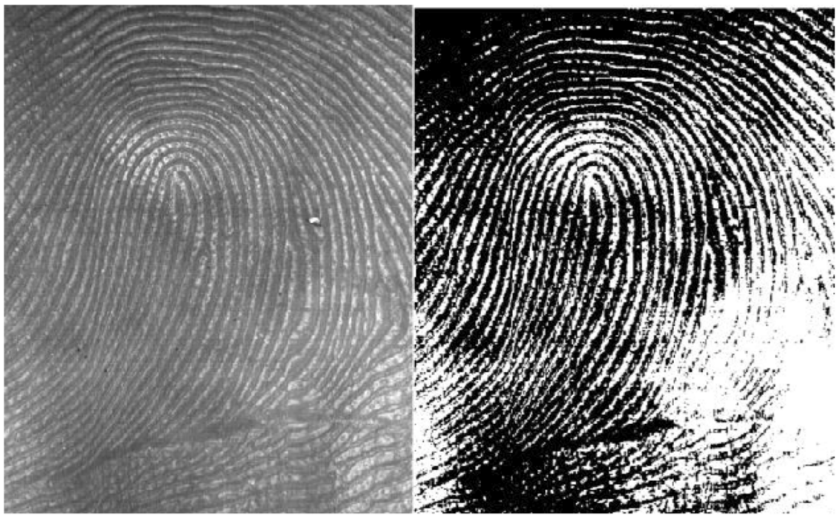

Figure 1.6: *Example of the resulting segmentation mask of an image of a fingeprint obtained trhough a thresholding algorithm. The result is not extremely good, but this techinque is very easy to implement and runs very quickly.*

There are many prominent Neural Network architectures used in the computer vision comunity nowadays, based on very differnet concepts such as convolution, recursion, dimensionality reduction and image generation. This section will provide an overview on the state of the art of this techlogy and will dwell briefly on the details behind some of those innovative architectures.

Recurrent Neural Networks (RNNs) and the LSTM

The typical application for RNN is processing sequential data, as written text, speech or video clips or any other kind of time-series signal. In this kind of data there is a strong dependece between values at a given time/position and values previously processed. Those models try implement the concept of *memory* weaving connections, outside the main information flow of the network, with the previous NN input. At each time-stamp the model collects the input from the current time X_i i and the hidden state from the previous step h_{i-1}, and outputs a target value and a new hidden state Figure 1.7. Typically RNN cannot manage easily long-term dependences in long sequences of signal. There is no theoretical limitation in this direction, but often it arises vanishing (or exploding) gradient problematics. A specific type of RNN has been designed to avoid this situation, the so called Long Short Term Memory (LSTM) [2]. The LSTM architecture includes three gates (input gate, output gate, forget gate), which regulate the flow of information into and out from a memory cell, which stores values over arbitrary time intervals.

Generative Adversarial Networks (GANs)

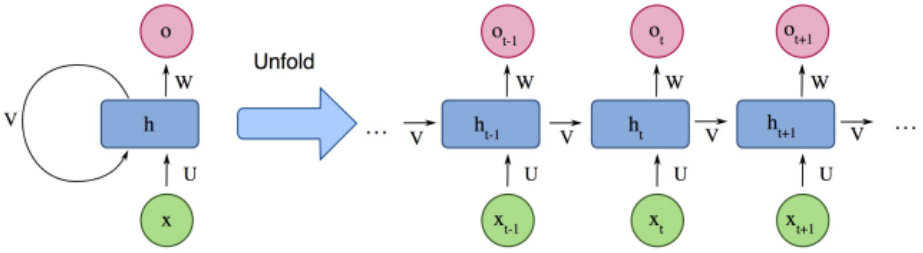

Figure 1.7: *Example of the structure of a simple RNN from [6].*

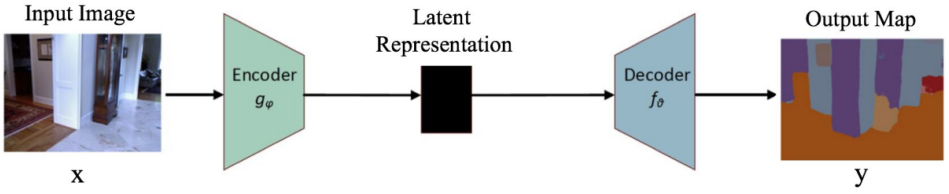

Figure 1.8: *Example of the structure of a simple ED NN from [6].*

TO DO

Encoder-Decoder and Auto-Encoder Models

Encoder-Decoder models try to learn the relation between an input and the correspondening output with a two steps process. The first step is the so called *encoding* process, in which the input x is compressed in what is called the *latent-space* representation $z = f(x)$. The second step is the *decoding* process, where the NN make a prediction of the output starting from the latent-spece representation $y = g(z)$. The idea underneath this approach is to capture in the latent-space representation the underlying semantic information of the input that is useful for predicting the output. ED models are widely used in image-to-image problems (where both input and output are images) and for sequential-data processing (like Natural Language Processing NLP). In Figure 1.8 is shown a schematich representation of this architecture. Usually these model follow a supervised training, trying to reduce the restruction loss between the predicted output and the ground-truth output provided while training. Typical application for this technology are image enhancing techinque like de-noising or super-resolution, where the output image is an improved version of the input image.

Convolutional Neural Networks (CNNs)

As stated before CNNs are a staple choice in image processing DL applications. They mainly consist of three type of layers:

i convolutional layers, where a kernel window of parameters is convolved with the image pixels and produce numerical features maps.

ii nonlinear layers, which apply an activation function on feature maps (usually element-wise). This step allow the network to introduce non-linear behaviour and then increasing its modeling capabilities.

iii pooling layers, which replace a small neighborhood of a feature map with some statistical information (mean, max, etc.) about the neighborhood and reduce spatial resolution.

Given the arrangment of successive layers, each unit receives weighted inputs from a small neighborhood, known as the receptive field, of units in the previous layer. The stack of layers allow the NN to perceive different resolutions: the higher-level layers learn features from increasingly wider receptive fields. The leading computational advantage given by a CNN architecture lyes in the sharing of kernels' weights within a convolutional layer. The result is a significantly smaller number of parameters than fully-connected neural networks. Some of the most notorious CNN architectures include: AlexNet [5], VGGNet [10], and U-Net [9].

For the purposes of this work the U-net architechture is particular interesting. The U-net model was initially developed for biomedical image segmentation, and in its structure reflects characteristics of both CNN and Encoder Decoder models. Ronneberger et al. [50] proposed this model for segmenting biological microscopy images. Their network and training IMAGE SEGMENTATION DATASET COMMONLY USED FROM [6].

Chapter 2

Model Generations

Chapter 3

Conclusions

3.1 conclusions

Bibliography

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018.

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

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[4] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[7] Muhammad Niazi, Thomas Tavolara, Vidya Arole, Douglas Hartman, Liron Pantanowitz, and Metin Gurcan. Identifying tumor in pancreatic neuroendocrine neoplasms from ki67 images using transfer learning. *PLOS ONE*, 13:e0195621, 04 2018.

[8] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143195, 1999.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[11] Sandro Skansi. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2018.