IMAGE ANALYSIS IN DIGITAL PATHOLOGY BASED ON MACHINE LEARNING TECHNIQUES & DEEP NEURAL NETWORKS

Paris-Panagiotis Amerikanos

University of Piraeus

Department of Digital Systems

Big Data & Analytics

June 2020

Supervisor: Dr. Ilias Maglogiannis

Abstract

Detection of regions of interest (e.g. mitosis or histologic primitives) in Whole Slide Images in a clinical setting is a highly subjective and labor-intensive task. In this thesis we explore recent developments in Machine Learning and Computer Vision algorithms to assess their possible usage and performance in tasks such as the above, in order to enhance and accelerate healthcare procedures. A state-of-the-art Deep Learning framework (Detectron2) is trained on the TUPAC16 dataset for object detection, and on the JPATHOL dataset for instance segmentation. We evaluate its predictions against competing models and discuss further possible improvements.

Keywords: Machine Learning, Deep Learning, Convolutional Neural Networks, Digital Pathology, Image classification, Object Detection, Instance Segmentation, Tumor Proliferation, Breast Cancer, Nuclei Segmentation, Epithelium Segmentation, Tubule Segmentation

Acknowledgements

I would like to thank Prof. Ilias Maglogiannis of the Dept. of Digital Systems at the University of Piraeus for supervising this research and providing the guidance needed to complete this thesis.

Also, I would like to thank God, my family, and friends (esp. Spyros & George) for their continual support.

Table of Contents

Chapter 1: Introduction

This thesis shall examine the feasibility of using a state-of-the-art Deep Learning framework to segment instances of tumor in breast cancer histopathological WSIs, as well as its performance. The two fields that shall be explored are Digital Pathology from the medical side where the problem lies, and Machine Learning/Computer Vision from the IT side, which offers us the techniques for a cost- and time-effective solution.

Thematic Area

The massive amounts of data gathered in the medical field has allowed experts to access enormous volumes of data about individual patients, but the sheer amount of it would simply forbid any human to process, use or understand it. As a response to that challenge, Statistical, Machine & Deep Learning techniques are being used (with the help of Big Data frameworks) to detect patterns and trends in all that raw data and produce understandable insights useful to experts.

The applications of ML techniques are spread across most of the medical fields and, while specialized software and algorithms have been around for some time, the differences are substantial. Until recently, the recommendation (or feature extraction) algorithm was hard coded and had a rigid nature, and the decision criteria were usually based on external research that could not be representative of the actual use cases as environment and populations could differ. On the other hand, ML algorithms can dynamically update with newer data provided by the expert that consults the software, providing an increased accuracy on the produced diagnosis.

The tasks that ML is applied to are numerous and wide-ranging. Identifying potential drug combinations, assisting the health practitioner’s diagnosis using the available symptoms data, or classifying medical images to the correct disease classes are some of the challenges ML is tasked to overcome.

The development of information systems for the automated diagnosis of medical images constitutes a field of ever-growing scientific research in the last decade. Digital medical images are present in most diagnostic labs, providing easy manipulation through various information systems. The digital processing of medical images by multiple feature extraction techniques can lead to the accumulation of numerous features in a reliable and reproducible way. The analysis of biomedical images through the values of extracted features is a process that can be carried out by Machine Learning algorithms (through use of various classifiers) and ultimately augment the decision making processes of medical experts by providing automated diagnosis insights. The information gain of such systems is significant as it enhances the timely and reliable identification of important patient cases. Systems like that can be incorporated in local information systems at diagnostic centers but can also be a part of a telehealth information system.

Structure

The present thesis will follow the following structure:

The second chapter presents previous work in the field, and relevant background theory concerning the problem from a medical perspective.

The third chapter gives a brief description of Machine Learning, and traditional and state-of-the-art Computer Vision approaches to Image Analysis.

The fourth chapter introduces the image datasets (TUPAC16 & JPATHOL) we will train our models on, as well as various related research that will facilitate our own research.

The fifth chapter presents the experimentation procedure followed, the DL framework used, and the prediction results achieved.

In the final two chapters we compare our results to other similar work, discuss improvements and draw our conclusions.

Chapter 2: Background & Related Work

Breast Cancer

Breast cancer is cancer that develops from breast tissue. Signs of breast cancer may include a lump in the breast, a change in breast shape, dimpling of the skin, fluid coming from the nipple, a newly inverted nipple, or a red or scaly patch of skin. In those with distant spread of the disease, there may be bone pain, swollen lymph nodes, shortness of breath, or yellow skin.

Risk factors for developing breast cancer include being female, obesity, a lack of physical exercise, alcoholism, hormone replacement therapy during menopause, ionizing radiation, an early age at first menstruation, having children late in life or not at all, older age, having a prior history of breast cancer, and a family history of breast cancer. About 5–10% of cases are the result of a genetic predisposition inherited from a person's parents, including BRCA1 and BRCA2, among others. Breast cancer most commonly develops in cells from the lining of milk ducts and the lobules that supply these ducts with milk. Cancers developing from the ducts are known as ductal carcinomas, while those developing from lobules are known as lobular carcinomas. There are more than 18 other sub-types of breast cancer. Some, such as ductal carcinoma in situ, develop from pre-invasive lesions. The diagnosis of breast cancer is confirmed by taking a biopsy of the concerning tissue. Once the diagnosis is made, further tests are done to determine if the cancer has spread beyond the breast and which treatments are most likely to be effective.

The balance of benefits versus harms of breast cancer screening is controversial. A 2013 Cochrane review found that it was unclear if mammographic screening does more harm than good, in that a large proportion of women who test positive turn out not to have the disease. A 2009 review for the US Preventive Services Task Force found evidence of benefit in those 40 to 70 years of age, and the organization recommends screening every two years in women 50 to 74 years of

age. The medications tamoxifen or raloxifene may be used to prevent breast cancer in those who are at high risk of developing it. Surgical removal of both breasts is another preventative measure in some high-risk women. In those who have been diagnosed with cancer, several treatments may be used, including surgery, radiation therapy, chemotherapy, hormonal therapy, and targeted therapy. Types of surgery vary from breast-conserving surgery to mastectomy. Breast reconstruction may take place at the time of surgery or later. In those in whom the cancer has spread to other parts of the body, treatments are mostly aimed at improving quality of life and comfort.

The five-year survival rates in England and the United States are between 80 and 90%. In developing countries, five-year survival rates are lower. Worldwide, breast cancer is the leading type of cancer in women, accounting for 25% of all cases. In 2018 it resulted in 2 million new cases and 627,000 deaths. It is more common in developed countries and is more than 100 times more common in women than in men.

Diagnosis

Most types of breast cancer are easy to diagnose by microscopic analysis of a sample (biopsy) of the affected area of the breast. Also, there are types of breast cancer that require specialized lab exams.

The two most used screening methods, physical examination of the breasts by a healthcare provider and mammography, can offer an approximate likelihood that a lump is cancer, and may also detect some other lesions, such as a simple cyst. When these examinations are inconclusive, a healthcare provider can remove a sample of the fluid in the lump for microscopic analysis (a procedure known as fine needle aspiration, or fine needle aspiration and cytology, FNAC) to help establish the diagnosis. A needle aspiration can be performed in a healthcare provider's office or

clinic. A local anesthetic may be used to numb the breast tissue to prevent pain during the procedure but may not be necessary if the lump is not beneath the skin. A finding of clear fluid makes the lump highly unlikely to be cancerous, but bloody fluid may be sent off for inspection under a microscope for cancerous cells. Together, physical examination of the breasts, mammography, and FNAC can be used to diagnose breast cancer with a good degree of accuracy.

Other options for biopsy include a core biopsy or vacuum-assisted breast biopsy, which are procedures in which a section of the breast lump is removed; or an excisional biopsy, in which the entire lump is removed. Very often the results of physical examination by a healthcare provider, mammography, and additional tests that may be performed in special circumstances (such as imaging by ultrasound or MRI) are sufficient to warrant excisional biopsy as the definitive diagnostic and primary treatment method.

One of the main problems found in the medical field, and which is being mitigated by help of Machine Learning, is the huge amount of medical data that has to be processed in order to reach conclusions about e.g. patient treatment, diagnosis, drug efficacy, pandemic prevention, and general research. Initially, medical experts had to manually sort through this data in its multitude of forms (images, text, numerical, etc.) in order to reach a conclusion. However, with the great advances in computational power during the last decade, Machine Learning (ML) and Deep Learning (DL) have placed this power at the feet of the experts who can reach their desired conclusions with great savings in time and cost, and, in some cases, greater accuracy than what any human could achieve.

Digital Pathology

Over the last decade, the nature of diagnostic healthcare has changed rapidly owing to an explosion in the availability of patient data for disease diagnosis (Madabhushi, 2009). Traditional

methods of analysis of cancer samples were limited to a few variables, usually stage, grade, and the measurement of a few clinical markers, such as estrogen receptor, progesterone receptor, HER2 for breast cancer and prostate specific antigen for prostate cancer (CaP). The pathologist was trained to synthesize this information into a diagnosis that would help the clinician determine the best course of therapy. These data were also used to try to understand the molecular basis of cancer with the goal of improving therapy. With the recent advent and cost-effectiveness of whole slide digital scanners, tissue histopathology slides can now be digitized and stored in digital image form. With the availability and analysis of a much larger set of variables combined with sophisticated imaging and analysis techniques, the traditional paradigm of a pathologist and a microscope could rapidly be replaced with a digital pathologist relying on a large flat screen panel to view and rapidly analyze digitized tissue sections.

Diagnosis: Dramatic increases in computational power and improvement in image analysis algorithms have allowed the development of powerful computer assisted analytical approaches to biomedical data. Just as with digital radiology over two decades ago, digitized tissue histopathology has now become amenable to the application of computerized image analysis and machine learning techniques for accurate diagnosis. In the context of CaP, for example, of the approximately 1 million biopsies performed in the USA every year, only 20% are found to be positive for cancer. This implies that pathologists are spending a large fraction of their time looking at benign tissue, which in most cases is easily distinguishable from cancer. This represents a huge waste of time and resources that might be better spent analyzing patients who indeed have CaP, or to focus on the cases where the disease is difficult to identify/classify or presents with nonstandard features. Consequently, several researchers have begun to develop computer aided diagnosis methods by applying image processing and computer vision techniques to try and identify spatial

extent and location of diseases such as breast cancer, CaP, neuroblastomas and meningiomas on digitized tissue sections.

One of the principal challenges in analysis of digital histopathology data is the enormous density of data that the algorithms must contend with, compared with radiological and other imaging modalities. For instance, the largest radiological datasets obtained on a routine basis are high-resolution chest CT scans comprising approximately 512 × 512 × 512 spatial elements or approximately 134 million voxels. A single core of prostate biopsy tissue digitized at 40× resolution is approximately 15,000 × 15,000 elements or approximately 225 million pixels. To put this in context, a single prostate biopsy procedure can comprise anywhere between 12 and 20 biopsy samples or approximately 2.5–4 billion pixels of data generated per patient study. Thus, unlike computer-aided detection (CAD) algorithms previously proposed for radiology, histopathology CAD algorithms are typically constructed within a multiresolution framework for them to be rapid, efficient, and accurate.

Prognosis: A second important role of computerized image analysis of Digital Pathology (DP) is to identify prognostic markers and to predict disease outcome and survival. For instance, in both breast cancer and CaP, cancer grade is known to be highly correlated to patient outcome and long-term survival. One of the issues with grade determination by a pathologist is the high degree of inter- and intra-observer variability. Since pathologist grade is reflected in tissue architecture and nuclear arrangement, graph-based algorithms have been proposed to quantitatively characterize spatial arrangement and distribution of histological structures such as cancer nuclei, lymphocytes, and glands. It is conceivable that these image-based predictors may in the future become powerful and accurate enough to be able to rival more expensive molecular prognostic assays in predicting disease outcome.

Theragnosis: It has always been accepted that cancer is a complex disease that we do not yet fully understand. In the clinic, the same treatment applied to two patients with diseases that look very similar have vastly different outcomes. A part of this difference is undoubtedly patient specific, but a part must also be a result of our limited understanding of the relationship between disease progression and clinical presentation. There is a consensus among clinicians and researchers that a more detailed approach, using computerized imaging techniques to better understand tumor morphology, combined with the classification of diseases into more meaningful molecular subtypes, will lead to better patient care and more effective therapeutics. The variables that can be used in such an analysis are the molecular features of a tumor (as measured by gene expression profiling or real-time PCR and FISH), results from the imaging of the tumor cellular architecture and microenvironment (as captured in histological imaging), the tumor 3D tissue architecture and vascularization (as measured by dynamic contrast enhanced MRI) and its metabolic features (as seen by metabolic or functional imaging modalities e.g., magnetic resonance spectroscopy or PET).While digital pathology offers very interesting, highly dense data, one of the exciting challenges in the future will be in the area of multimodal data fusion for making therapy recommendations (theragnosis), especially as it pertains to personalized medicine.

Role of the pathologist: While image analysis methods for digital pathology are rapidly finding application in the clinic, both imaging, computer scientists and pathologists alike need to appreciate that the primary purpose of these tools is to complement the role of the pathologist. They will not in the short or medium term be able to replace the vast domain of expertise that a pathologist brings to the table; a lesson that we can appreciate from radiology where the availability of commercial CAD systems over the last two decades has not in any way diminished the role of the radiologist. Most histopathology image analysis researchers are computer vision

researchers. As such, it is important to maintain a constant collaboration with clinical and research pathologists throughout the research process. There are unique challenges to analysis of histopathology imagery, particularly in the performances required for eventual use of the technique in a clinical setting. It is the pathologist who can best provide the feedback on the performance of the system, as well as suggesting new avenues of research that would provide beneficial information to the pathologist community. Additionally, it is the pathologist that is best equipped to interpret the analysis results considering underlying biological mechanisms which, in turn, may lead to new research ideas.

Outlook: We are living in an exciting time when disease diagnostics and treatment are becoming more accurate and patient specific. Computerized imaging methods are beginning to assist the pathologist and radiologist in making an accurate diagnosis of disease and identify morphological features correlated with prognosis. Molecular profiling of disease promises to help the clinician understand the underlying biology of the disease and suggest new and more effective therapeutics. We stand at the threshold of an era when predictive, preventive, and personalized medicine will transform medicine by decreasing morbidity in cancer. We believe this transformation will be driven by the integration of multiscale heterogeneous data. The goal of the research of many scientists is aimed at a future when disease diagnostics will involve the quantitative integration of multiple sources of diagnostic data, including genomic, imaging, proteomic and metabolic data acquired across multiple scales/resolutions that can distinguish between individuals or between subtle variations of the same disease to guide therapy. Quantitative crossmodal data integration will also allow disease prognostics, enabling physicians to predict susceptibility to a specific disease as well as disease outcome and survival. Finally, the analysis will provide theragnostic; the ability to predict how an individual will react to various treatments.

Such a theragnostic profile would be a synthesis of various biomarkers and imaging tests from different levels of the biological hierarchy. It would be used as the 'signature' of an individual patient, useful in predicting her/ his response to drug treatment. A collection of these profiles, followed up over time, would provide insights into the disease process and be useful for improvements in developing future treatment options.

Existing approaches vs Deep Learning

Digital pathology is becoming increasingly common due to the growing availability of whole slide digital scanners. These digitized slides afford the possibility of applying image analysis techniques to DP for applications in detection, segmentation, and classification. Algorithmic approaches have shown to be beneficial in many contexts as they have the capacity to not only significantly reduce the laborious and tedious nature of providing accurate quantifications, but to act as a second reader helping to reduce inter-reader variability among pathologists. Several image analysis tasks in DP involve some sort of quantification or tissue grading and invariably require identification of histologic primitives (e.g., nuclei, mitosis, tubules, epithelium, etc.). As a result, there is a strong need to develop efficient and robust algorithms for analysis of DP images.

While there have been a few papers in the area of computational image analysis of DP images for the purposes of object detection and quantification in the last few years, there appear to be two main drawbacks to existing approaches. First, the development of task specific approaches tends to require long research and development cycles: an algorithmic scheme needs to be developed which can account for as many of the variances as possible while not being too general as to result in false positive results or too narrow as to result in false negative errors. This process can become quite unwieldy as it is often infeasible to view all the outlier cases a priori, and thus an extensive iterative trial and error approach needs to be undertaken. The second

drawback with existing approaches is the required but limited implicit knowledge of how to find or adjust optimal parameters often resides solely with the developers of the algorithms and thus are not intuitively understood by external parties. Together, these create a strong hindrance for researchers to leverage or extend the available technology to investigate their clinical hypothesis.

Deep learning is an example of the machine learning paradigm of feature learning; wherein it iteratively improves upon learned representations of the underlying data with the goal of maximally attaining class separability. There are no preexisting assumptions about any task or dataset in the form of encoded domain-specific insights or properties which guide the creation of the learned representation. The DL approach involves deriving a suitable feature space solely from the data itself. This is a critical attribute of the DL family of methods, as learning from training exemplars allows for a pathway to generalization of the learned model to other independent test sets. Once the DL network has been trained with an adequately powered training set, it is usually able to generalize well to unseen situations, obviating the need of manually engineering features.

DL is suited to analyze big data repositories where employing a feature engineering or approach would require several algorithmic iterations and substantial effort to capture a similar range of diversity. Many manually engineered or feature-based approaches are not implicitly poised to manipulate and distil large datasets into classifiers in an efficient way. DL approaches, on the other hand, function well under these circumstances: they have the potential to become the unifying approach for the many tasks in DP, having previously been shown to produce state-of-the-art results across varied domains. As such, the focus of this manuscript is to discuss the usage of a single framework which can be marginally tweaked to apply to a diverse set of unique use cases.

Chapter 3: Machine Learning & Computer Vision

Machine Learning

Machine Learning is a research field in computer science and engineering. As a branch of Artificial Intelligence it allows the extraction of meaningful patterns from examples, just as human intelligence allows us to do (Erickson, 2017). A computer that performs repetitive and well-defined tasks can cover a variety of use cases as it will perform a given task consistently and efficiently, unlike humans. In the last decades, computers have demonstrated the ability to learn and even have become proficient in tasks that were thought to be too complex for machines, showing that ML algorithms can be critical components of computer-aided diagnosis and decision support systems. An intriguing finding is that in some instances computers may be able to discern patterns that are imperceivable to humans. This discovery has naturally led to strong and heightened enthusiasm in the field of ML, especially along with the latest substantial increases in computational performance and available data.

Of great interest is how ML could be applied to medical imaging. Using ML algorithms to perform computer-aided detection and diagnosis can help medical experts interpret medical imaging findings and reduce analysis times. Examples of challenging tasks where these algorithms have been used are: pulmonary embolism segmentation with computed tomographic (CT) angiography, polyp detection with virtual colonoscopy or CT in the setting of colon cancer, breast cancer detection and diagnosis with mammography, brain tumor segmentation with magnetic resonance (MR) imaging, and detection of the cognitive state of the brain with functional MR imaging to diagnose neurologic diseases (e.g., Alzheimer disease).

A broadly accepted definition of ML is: If an ML algorithm is applied to a dataset and some knowledge (ground truth) related to that, then the algorithm system can learn from the training

data and apply what it has learned to make a prediction. An example would have medical images of tumors as data, classification of benign or malignant per instance as knowledge, and whether a new image is depicting benign or malignant tumor tissue as the output prediction. The algorithm is considered to be learning a task as the system optimizes its variables in such a way that its performance metrics improve, meaning more test cases are diagnosed correctly.

Machine Learning now has many areas of application outside of medicine, with a central role in tasks such as language translation, speech recognition, product recommendations, and autonomous vehicle navigation. Even though some of those were unattainable up to a while ago, recent advances have made them feasible. Older ML algorithms demanded structured input, and some techniques would fail learning if any point of data were missing, but newer algorithms can accommodate omissions in data and, in some instances, purposefully create data omissions to make the algorithm more robust.

Four branches of machine learning

Machine Learning algorithms generally fall into four broad categories, as described in the following sections: supervised, unsupervised, self-supervised and reinforcement learning.

Supervised learning: This is the most common case and consists of learning to map input data to known targets (annotations), given a set of examples (often annotated by humans). Examples of supervised learning algorithms include support vector machine, decision tree, linear regression, logistic regression, naive Bayes, k-nearest neighbor, random forest, AdaBoost, and neural network methods. Generally, almost all applications of DL that are in the spotlight these days belong in this category, such as optical character recognition, speech recognition, image classification, and language translation. Although supervised learning mostly consists of classification and regression, there are more exotic variants as well, including:

Sequence generation: Given a picture, predict a caption describing it. Sequence generation can be reformulated as a series of classification problems (e.g. repeatedly predicting tokens).

Syntax tree prediction: Given a sentence, predict its decomposition into a syntax tree.

Object detection: Given a picture, draw a bounding box around certain objects inside the picture. This can also be expressed as a classification problem (given many candidate bounding boxes, classify the contents of each one) or as a joint classification and regression problem, where the bounding-box coordinates are predicted via vector regression.

Image segmentation: Given a picture, draw a pixel-level mask on a specific object. In a related example of a brain tumor classification problem (malignant vs benign vs normal), supervised learning involves gaining experience by using images of brain tumor examples that contain important information (benign vs malignant labels) and applying the gained expertise to predict benign and malignant neoplasia on unseen new brain tumor images (test data).

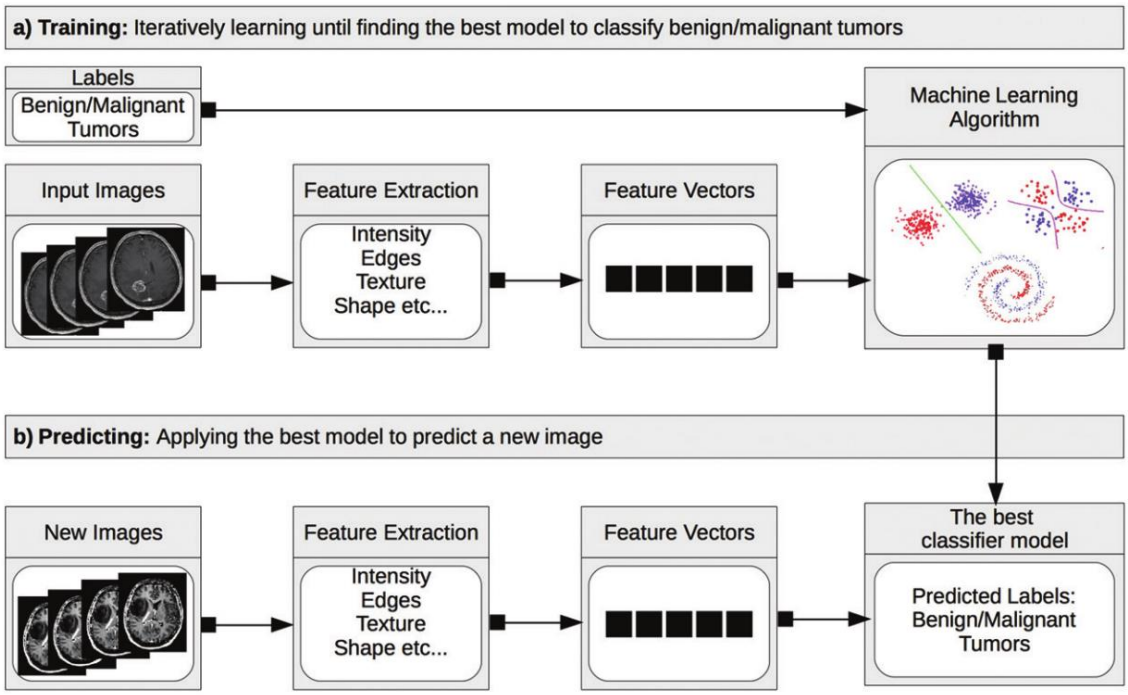


Figure 1: ML model development and application model for medical image classification tasks.

Unsupervised learning: This branch of ML consists of finding interesting transformations of the input data without the help of any targets, for the purposes of data visualization, data compression, or data denoising, or to better understand the correlations present in the data at hand. Unsupervised learning is the main means of Data Analytics and it is often a necessary step in better understanding a dataset before attempting to solve a supervised-learning problem. Dimensionality reduction and clustering are well-known categories of unsupervised learning. Examples of unsupervised learning algorithm systems include K-means, mean shift, affinity propagation, hierarchical clustering, DBSCAN (density-based), Gaussian mixture modeling, Markov Random Fields SODATA (iterative self-organizing data), amongst others.

In the Digital Pathology field, a common task is where data are processed in order to separate the images into groups - for example, benign and malignant tumors - without providing the algorithm with information regarding what the groups are; the algorithm determines the group count and the approach of separation.

Self-supervised learning: Self-supervised learning is supervised learning without human-annotated labels. There are still labels involved (because the learning must be supervised by something), but they are generated from the input data, typically using a heuristic algorithm.

For instance, autoencoders are a well-known instance of self-supervised learning, where the generated targets are the unmodified input. In the same way, trying to predict the next frame in a video, given past frames, or the next word in a text, given previous words, are instances of self-supervised learning (temporally supervised learning, in this case: supervision comes from future input data). Note that the distinction between supervised, self-supervised, and unsupervised learning can be blurry sometimes, as these categories are more of a continuum without solid borders. Self-supervised learning can be reinterpreted as either supervised or unsupervised

learning, depending on whether you pay attention to the learning mechanism or to the context of its application.

Reinforcement learning: Long overlooked, this branch of ML recently started to get a lot of attention after Google DeepMind successfully applied it to learning to play Atari games. In reinforcement learning, an agent receives information about its environment and learns to choose actions that will maximize some reward. For instance, a neural network that "looks" at a videogame screen and outputs game actions in order to maximize its score can be trained via reinforcement learning. Currently, reinforcement learning is mostly a research area and has not yet had significant practical successes beyond games. In time, however, reinforcement learning is expected to take over an increasingly large range of real-world applications: self-driving cars, robotics, resource management, education, and so on.

In this thesis, we focus on supervised learning, since it is the most common training style applied to medical images.

Feature Extraction and Selection

The primary action in ML is to retrieve the features that contain the information on which insights will be based. Even though feature learning visually is easy for humans, e.g. during medical training, computing and representing a feature is a complex task. A usual approach is assigning a numeric value to an optical texture. Visual features must be robust enough to overcome morphological variations, the most common of whom observed in medical imaging data being rotations, noise, and intensity differences.

It is possible to retrieve a great number of features from a given image, but too many features can stunt the learning of the true characteristics of a decision and lead to overfitting. In order to make the best possible predictions, a subset of the features has to be selected; this process

is known as feature selection. A popular feature selection technique is looking for correlations amongst features: large numbers of correlated features may mean that some features may be omitted, and their number can be reduced with minimal information loss. Nonetheless, depending on the case, a complex relationship may exist and evaluating an isolated feature may be unsound.

For example, let us imagine a binary classification list with people's weights, where each weight indicates obesity or absence thereof. By adding a heights attribute, the accuracy would likely improve: a high weight value together with a low height value are more probable to indicate obesity than a high weight value together with a high height value.

Training and Testing

Supervised ML takes its name from the need for available samples from each class to be learned. Too few samples will prevent a model from recognizing the needed features of an object that will allow it to discern between the various classes. The required number of samples for every class to be learned depends mainly on the discreteness of each class.

The cross-validation technique is a popular approach to evaluating the accuracy of an ML model when there is a limited dataset available. Using cross-validation on a given dataset, a split is made on the dataset, using its major partition of samples for training, and designating the remainder for testing purposes. After training of the model has completed on the training subset, the learned model is tested on the testing subset. The process is repeated multiple times, but with a different split of training and testing samples from the full dataset of samples. The most extreme version would call removing just one sample for testing and using the rest for every training iteration, which is known as leave-one-out cross-validation. Despite being considered a good method for accuracy evaluation, cross-validation is limited in that each training and testing iteration results in a separate model, not providing a single final model ready for use.

Types of Machine Learning Algorithms

There is a great variety of ML algorithms available for obtaining the optimum weights for features, based on differing assumptions concerning the data and depending on different methods for adjusting the feature weights. Some of the most used algorithms are described below.

Neural Networks: The archetypal ML method is neural networks, whose learning schema is divided in three functions: (a) the error function that quantifies the output quality for a given set of inputs, (b) the search function that defines the derivative (direction and magnitude of change) needed to minimize the error function, and (c) the update function that determines how the weights of the network are refined based on the search function values. During training, as samples are presented to the neural network, the error for each is computed, and the total error is measured. For each iteration, the search function determines the gradient according to the total error, and the update function uses this to adjust the weights. The weights are continuously updated until the error is steadily minimized.

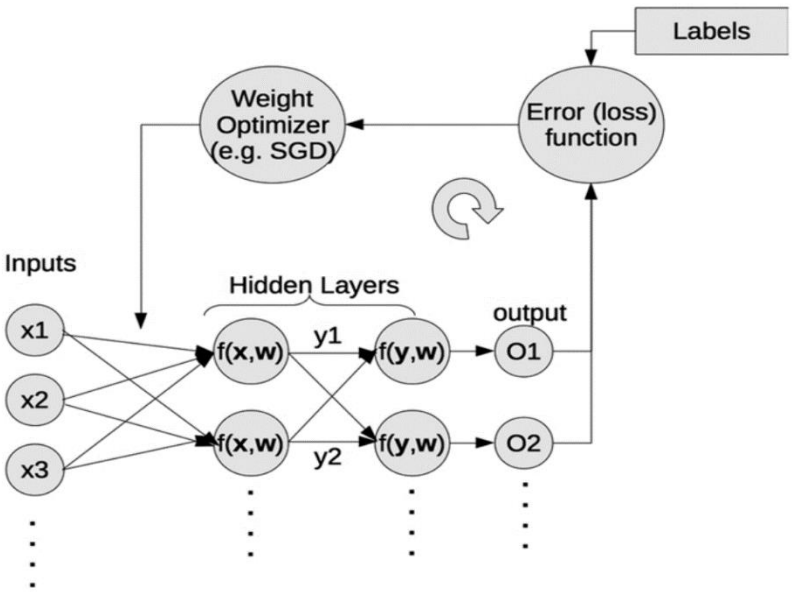


Figure 2: Example neural network with input nodes, hidden layers, and an output layer.

Let us take as an example a neural network with n-input nodes, two hidden layers, and an output layer with several output nodes (Figure 2). The output nodes are summed and compared with the desired output by the error function, returning a loss metric, with which the weights in the neural network are updated. Real-world neural networks generally are built with multiple hidden layers and utilize more complex functions.

k-Nearest Neighbors: With k-nearest neighbors we classify a collection of features for an unknown sample by assigning it to the most related classes. We define neighbors as the nearest samples that determine the classes our sample object may belong. If the neighbor count is 1, then the unknown sample is directly attached to the class of that single neighbor. It is crucial the normalization of the feature vectors is implemented properly, so that the similarity function, which determines the distances amongst the samples, can return valid results. This function is usually defined as a Minkowski or Euclidean.

Support Vector Machines: SVMs transform input data so that maximum separation between the classes is achieved, by splitting them with the widest possible plane or support vector. They allow flexibility in setting the tradeoff point between widest plane of separation and the number of misclassified points. SVMs were developed in the ‘90s, but gained traction in the most recent years thanks to the addition of functions that can map points to other dimensions by using nonlinear relationships, thus allowing classification of non-linearly separable samples, which is considered an important advantage over the other ML methods.

Decision Trees: Compared to many other ML methods described, decision trees have the significant advantage that the rules they generate can be understood by humans, with regard to classification of samples. They are recognizable to most people and use the yes/no scheme, e.g. whether a feature value is above a threshold. Another advantage of decision trees is the ability to

rapidly search through the many possible combinations to find the one that will form the simplest and most accurate tree. When run, two parameters need to be set: maximal depth (number of decision points) and maximal breadth, which establish the tradeoff between accurate results and more decision points.

In some cases, an ensemble method can be used to improve results, where multiple decision trees are constructed, the most common of whom are random forest and boosting with aggregation techniques. When boosting with aggregation (bagging), multiple decision trees are generated by repeatedly resampling the training data through replacement, and voting on them to reach a consensus. The random forest technique uses a few decision trees to improve accuracy without resampling the data, and, unlike bagg ing where all features are considered for splitting nodes, only a subset of those is selected at random and each node is split using the best split feature from the subset.

Naive Bayes Algorithm: According the Bayes theorem, which is one of the oldest ML methods, the probability of an event is a function of related events as defined by its formula: $P(y|x) = \frac{P(y)\times P(x|y)}{P(x)}$. In ML, wherever multiple input features are available, the probabilities of each feature must be chained together to calculate the probability of a class, given the provided array of features. The NBA is different from most ML algorithms in that a single computation is defines the relationship between an input and an output feature set. Therefore, unlike most other ML methods, this one does not involve an iterative training process, but other training/testing data issues still do apply. The “Naïve” reference emphasizes that all features are assumed to be independent amongst themselves, even though this may lead to inaccurate results in real-life problems. Still, even when this assumption is breached, useful estimates of performance can be obtained.

Finally, the NBA often leads to better results with fewer available samples or possibilities. These factors highlight the importance of having reliable prior probabilities, apart from accuracy alone, when measuring performance metrics.

Deep Learning

Deep Learning (DL) is a novel and rapidly developing research area yielding very interesting results. Due to limitations in computing power and difficulties in the backpropagation process, primitive neural networks had mostly less than 5 layers (shallow nets); by the current definition, DL concerns neural networks typically deeper than 20 layers. These challenges have been overcome through leveraging the parallel computing power of game-oriented GPUs, mostly developed by NVidia. Different DL NN architectures have been developed for varying purposes, such as speech recognition, language translation, classification and segmentation on images, etc. Some of the available DL algorithms are stacked auto encoders, Boltzmann machines, Deep Neural Networks (DNNs), Recurring Neural Networks (RNNs), and Convolutional Neural Networks (CNNs).

This thesis will concentrate on CNNs as these networks are typically used for problems involving images (photos, medical and scientific imaging, etc.). The main point of CNNs is that they assume that the input arrays have some sort of localized geometric relationship between their

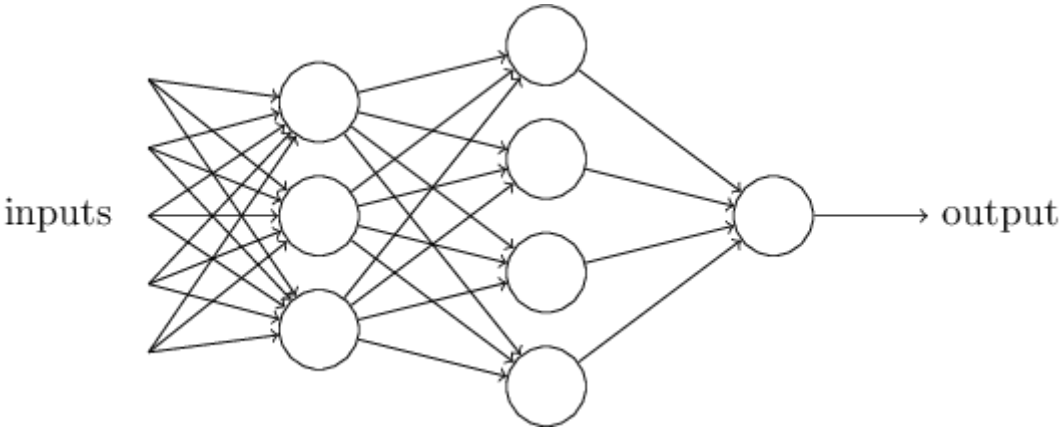


Figure 3: Example neural network with two hidden layers

rows and columns of pixels. The input layer nodes of a CNN are set to produce a moving convolution of a matrix (called kernel) over the input image, computing an output value at each location.

Other layers found in a CNN apart from the convolution kernels are activation layers and pooling layers. Activation layers initially implemented functions designed to simulate the sigmoidal activation function of neurons, but modern ones are much simpler, typically implementing a rectified linear unit (ReLU) which returns zero value for negative input values, otherwise returns the positive input value. The pooling layer implements a max-pool function: it takes the output of a convolution kernel and outputs only its greatest value, rewarding so the optimal convolution function that derives the most significant features of an image.

During training of DNNs, it is crucial to implement regularization, where weights between layers are rescaled to a more efficient range. One regularization approach is application of Dropout: random weights between nodes of layers (typically more than half) are set to zero with each learning round (epoch). Dropout may improve performance by preventing overfitting. Through enough iterations, if random weights are set to zero and a number of samples are passed for training, then only the important weights will have a significant effect on performance and will be preserved until the end of training.

Concerning the DNN's architecture, there are no set rules to define the correct number, type, size or ordering of layers for a given problem – it is still a trial and-error process. Some DNN architectures that have been successful in ML competitions (e.g. the ImageNet Challenge) are LeNet, GoogleNet, AlexNet, VGGNet, and ResNet.

In comparison with traditional ML techniques, CNN DL algorithms have the important benefit of not needing to compute initial features. The CNN can efficiently extract the important

features during training. eliminating the testing bias, computational overhead, and selection procedure of features that a human believes to be of importance.

Deep Learning in Computer Vision

Rapid progressions in DL and improvements in device capabilities including computing power, memory capacity, power consumption, image sensor resolution and optics have improved the performance and cost-effectiveness of vision-based applications. Compared to traditional CV techniques, DL enables CV engineers to achieve greater accuracy in tasks such as image classification, semantic segmentation, object detection and Simultaneous Localization and Mapping (SLAM) (O'Mahony, 2019). Since neural networks used in DL are trained rather than programmed, applications using this approach often require less expert analysis and fine-tuning and exploit the tremendous amount of video data available in today's systems. DL also provides superior flexibility because CNN models and frameworks can be re-trained using a custom dataset for any use case, contrary to CV algorithms, which tend to be more domain specific. Taking the problem of object detection as an example, we can compare the two types of algorithms for computer vision, traditional vs DL.

The traditional approach is to use well-established CV techniques such as feature descriptors (SIFT, SURF, BRIEF, etc.) for object detection. Before the emergence of DL, a step called feature extraction was carried out for tasks such as image classification. Features are small descriptive or informative patches in images. Several CV algorithms, such as edge detection, corner detection or threshold segmentation may be involved in this step. As many features as practicable are extracted from images and form a definition (a bag-of-words) of each object class. At the deployment stage, these definitions are searched for in other images. If a significant number of features from one bag-of-words are in another image, the image is classified as containing that

specific object (i.e. chair, horse, etc.). The difficulty with this traditional approach is that it is necessary to choose which features are important in each given image. As the number of classes to classify increases, feature extraction becomes more and more cumbersome. It is up to the CV engineer's judgment and a long trial and error process to decide which features best describe different classes of objects. Moreover, each feature definition requires dealing with a plethora of parameters, all of which must be fine-tuned by the CV engineer.

DL introduced the concept of end-to-end learning where the machine is just given a dataset of images which have been annotated with what classes of object are present in each image. Thereby a DL model is 'trained' on the given data where neural networks discover the underlying patterns in classes of images and automatically works out the most descriptive and salient features with respect to each specific class of object for each object. It has been well-established that DNNs perform far better than traditional algorithms, albeit with trade-offs regarding computing requirements and training time. With all the state-of-the-art approaches in CV employing this methodology, the workflow of the CV engineer has changed dramatically where the knowledge and expertise in extracting hand-crafted features has been replaced by knowledge and expertise in iterating through DL architectures as depicted in Figure 4:

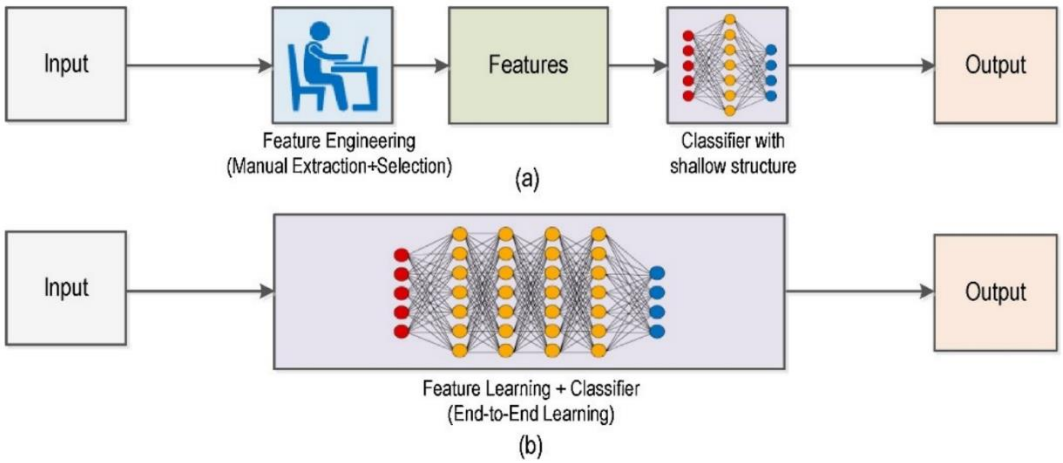


Figure 4: (a) Traditional Computer Vision workflow vs. (b) Deep Learning workflow.

The development of CNNs has had a tremendous influence in the field of CV in recent years and is responsible for a big jump in the ability to recognize objects. This burst in progress has been enabled by an increase in computing power, as well as an increase in the amount of data available for training neural networks. The recent explosion in and wide-spread adoption of various DNN architectures for CV is apparent in the fact that the seminal paper *ImageNet Classification with Deep Convolutional Neural Networks* (Krizhevsky, 2012) has been cited over 63,000 times.

Neural Networks

A Neural Network is a graph constructed by artificial neurons called a perceptrons. Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts (Rosenblatt, 1958). Today, it is more common to use other models of artificial neurons like the sigmoid neuron. A perceptron takes several binary inputs and produces a single binary output, as seen in Figure 5.

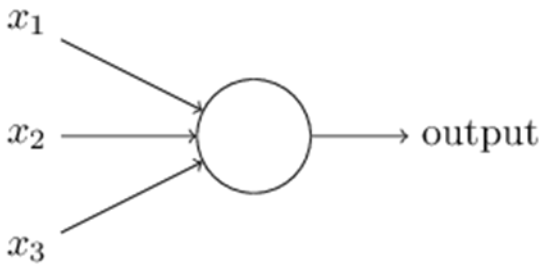


Figure 5: Perceptron with three inputs and one output

In the example shown, the perceptron has three inputs, but, in general, it could have more or fewer inputs. Rosenblatt proposed a simple rule to compute the output where he introduced real numbers as weights expressing the importance of the respective inputs to the output. The neuron's output, 0 or 1, is determined by whether the weighted sum is less than or greater than some threshold value. Like the weights, the threshold is a real number which is a parameter of the neuron.

A perceptron is a device that makes decisions by weighing up evidence, and it should seem plausible that a complex network of perceptrons could make quite subtle decisions. In the following network, the first column of perceptrons is making three very simple decisions, by weighing the input evidence. Each of the second layer perceptrons is making a decision by weighing up the results from the first layer of decision-making. In this way a perceptron in the second layer can decide at a more complex and more abstract level than perceptrons in the first

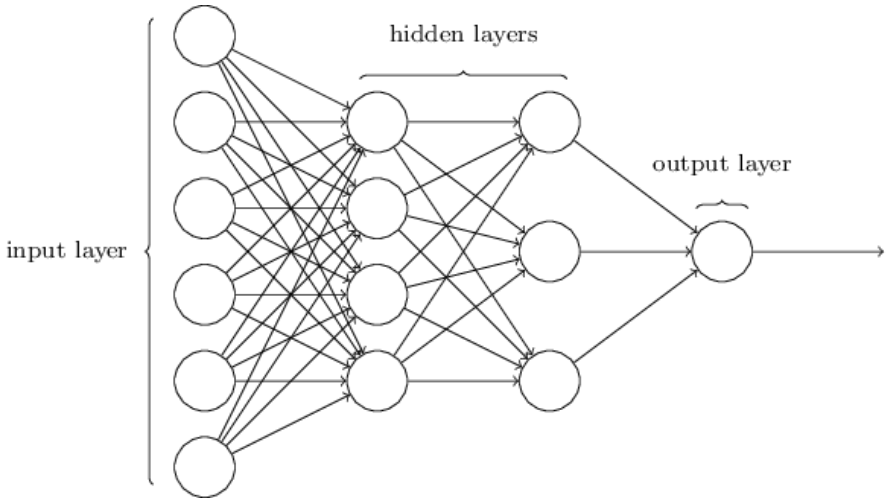


Figure 6: Multilayer Perceptron with input, two hidden, and output layers

layer. Even more complex decisions can be made by the perceptron in the third layer. In this way, a multi-layer network of perceptrons can engage in sophisticated decision making.

Sigmoid neurons are similar to perceptrons but modified so that small changes in their weights and bias cause only a small change in their output, which is the crucial fact that will allow a network of sigmoid neurons to learn. Just like a perceptron, the sigmoid neuron has inputs, but instead of being just 0 or 1, these inputs can also take on any values between 0 and 1. Also just like a perceptron, the sigmoid neuron has weights for each input and an overall bias. However, the output is not 0 or 1; instead, it is $\sigma(w \cdot x+b)$, where σ is called the sigmoid function. One important difference between perceptrons and sigmoid neurons is that sigmoid neurons don't just output 0 or 1 but can have as output any real number between 0 and 1. This can be useful, for example, if we

want to use the output value to represent the average intensity of the pixels in an image input to a neural network.

The leftmost layer in this multilayer perceptron (despite being made up of sigmoid neurons) is called the input layer, and the neurons within the layer are called input neurons. The rightmost or output layer contains the output neurons, or, as in this case, a single output neuron. The middle layer is called a hidden layer since the neurons in this layer are neither inputs nor outputs.

This network where the output from one layer is used as input to the next layer is called a feedforward neural network. This means there are no loops in the network - information is always fed forward, never fed back. However, there are other models of artificial neural networks in which feedback loops are possible and these are called Recurrent Neural Networks (RNNs). The idea in these models is to have neurons which fire for some limited duration of time, before becoming quiescent. That firing can stimulate other neurons, which may fire a little while later, also for a limited duration, and so over time we get a cascade of neurons firing.

RNNs have been less influential than feedforward networks, in part because the learning algorithms for recurrent nets are usually less powerful, but they are still extremely interesting. They are much closer in spirit to how our brains work than feedforward networks and it is possible that RNNs can solve important problems which can only be solved with great difficulty by feedforward networks.

Convolutional Neural Networks

Convolutional Neural Networks are very similar to ordinary neural networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other, and they still have a loss function on the last (fully connected) layer. The main difference is ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the number of parameters in the network.

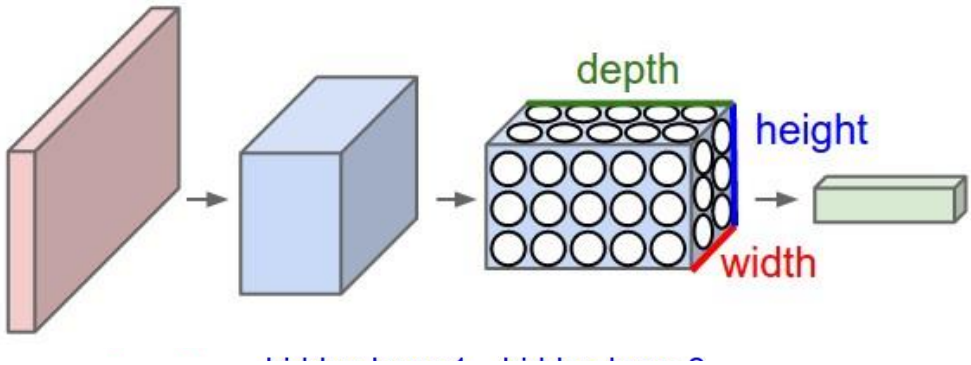


Figure 8: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.

Unlike a regular Neural Network (Figure 7), the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth (Figure 8). The neurons in a layer will only be connected to a small region of the layer before it, instead of all the neurons in a fully connected manner. Moreover, the final output layer would for e.g. CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture the full image will be reduced into a single vector of class scores, arranged along the depth dimension.

Layers used to build ConvNets: A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. Three main types of layers are stacked to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully Connected Layer. In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores.

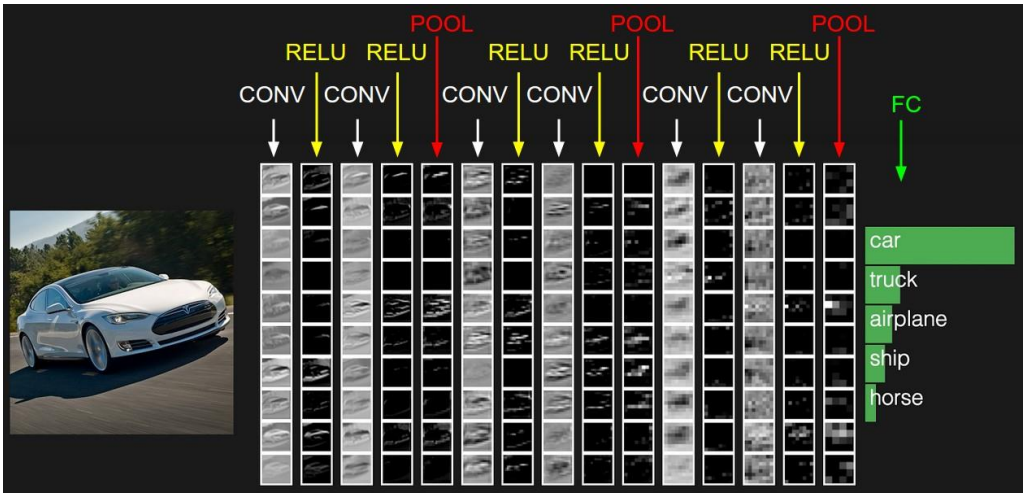


Figure 9: The activations of an example ConvNet architecture. The initial volume stores the raw image pixels (left) and the last volume stores the class scores (right). Each volume of activations along the processing path is shown as a column.

Convolutional Layer: The Conv layer is the core building block of a CNN that does most of the computational heavy lifting. Its parameters consist of a set of learnable filters. Every filter is small spatially (along width and height) but extends through the full depth of the input volume. During the forward pass, we convolve each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume, we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in

each CONV layer, and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, we will connect each neuron to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (equivalently this is the filter size). The extent of the connectivity along the depth axis is always equal to the depth of the input volume.

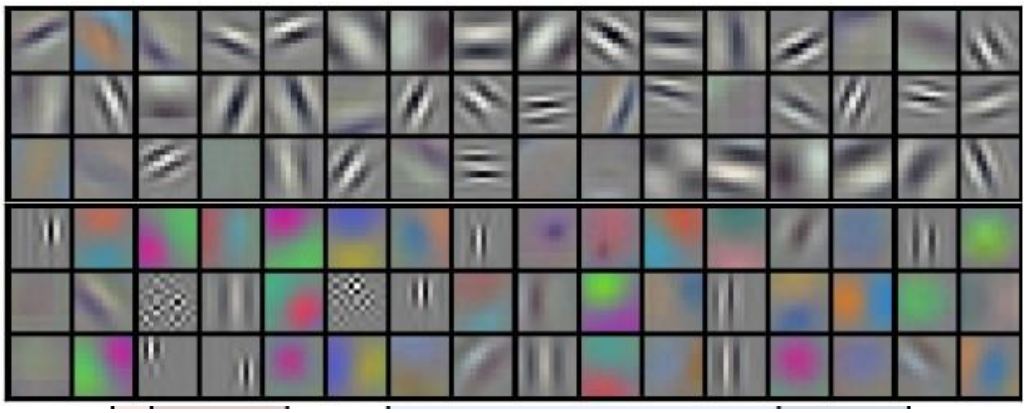


*Figure 10: Example filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice.*

Figure 11: An example input volume in red, and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth.

Three hyperparameters control the size of the output volume: the depth, stride and zero-padding. We can compute the spatial size of the output volume as a function of the input volume size (WW), the receptive field size of the Conv Layer neurons (FF), the stride with which they are applied (SS), and the amount of zero padding used (PP) on the border. Calculating how many neurons fit is given by (W−F+2P)/S+1(W−F+2P)/S+1.

Pooling Layer: It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network (Figure 12), and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation (Figure 13). The depth dimension remains unchanged. It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: A pooling layer with F=3,S=2F=3,S=2 (also called overlapping pooling), and more commonly F=2,S=2F=2,S=2. Pooling sizes with larger receptive fields are too destructive.

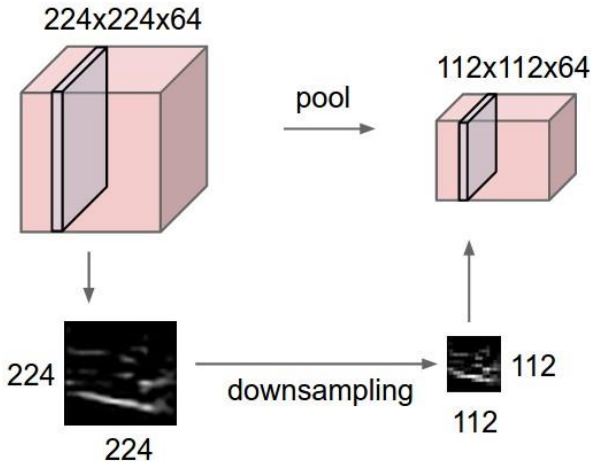


Figure 12: Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume - the volume depth is preserved.

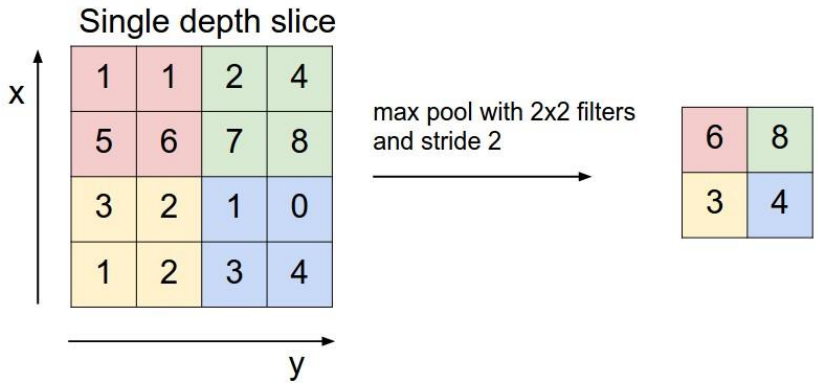


Figure 13: The most common downsampling operation is max (pooling), here shown with a stride of 2.

Fully connected layer: Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

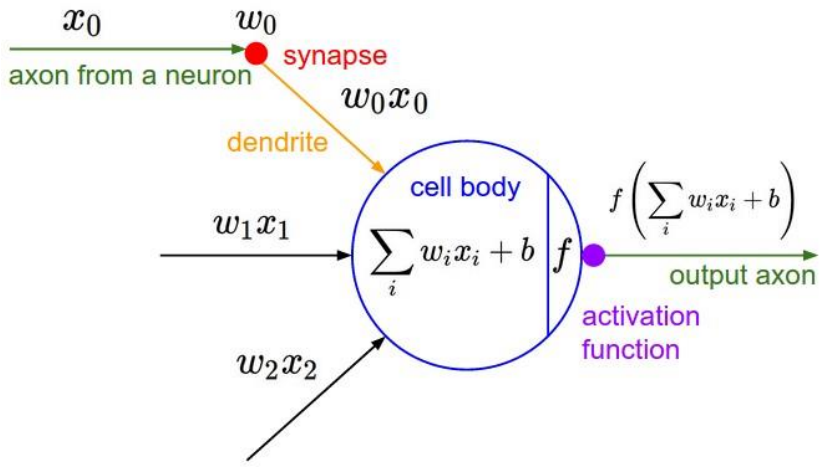


Figure 14: The neurons of a CNN compared to a regular Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

ConvNet Architectures: The most common form of a ConvNet architecture stacks a few CONV-RELU layers, follows them with POOL layers, and repeats this pattern until the image has been merged spatially to a small size. At some point, it is common to transition to fully connected layers. The last fully connected layer holds the output, such as the class scores. In other words, the most common ConvNet architecture follows the pattern: INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC where the * indicates repetition, and the POOL? indicates an optional pooling layer. Moreover, N >= 0 (usually N <= 3), M >= 0, K >= 0 (usually K < 3).

It should be noted that the conventional paradigm of a linear list of layers has recently been challenged in Google's Inception architectures and in state-of-the-art ResNets from Microsoft Research Asia. Both feature more intricate and different connectivity structures. In practice we use whatever works best on ImageNet: in 90% or more of applications the same structure can be applied. Instead of rolling our own architecture for a problem, we could look at whatever

architecture currently works best on ImageNet, download a pretrained model and finetune it on our data. Rarely is there a need to train or design a CNN from scratch. The most common CNN architectures used are: LeNet, AlexNet, ZF Net, GoogLeNet, VGGNet, ResNet (Kayalibay, 2017) (Kraus, 2016).

Computational Considerations: The largest bottleneck to be aware of when constructing CNN architectures is the memory bottleneck (CS, n.d.). There are three major sources of memory use to keep track of:

- The intermediate volume sizes: These are the raw number of activations at every layer of the CNN, and their gradients (of equal size).
- The parameter sizes: These are the numbers that hold the network parameters, their gradients during backpropagation, and commonly also a step cache. Therefore, the memory to store the parameter vector alone must usually be multiplied by a factor of at least 3 or so.
- Miscellaneous: Every implementation must maintain memory for various objects, such as the image data batches, their augmented versions, etc.

Once a rough estimate of the total number of values (for activations, gradients, and misc.) is calculated, the number should be converted to size in GB. If the network does not fit, a common heuristic to force it is to decrease the batch size, since most of the memory is usually consumed by the activations.

Computer Vision

Computer Vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos (Ballard & Brown, 1982) (Huang, 1996). From the perspective of engineering, it seeks to automate tasks that the human visual system can do. Computer vision is concerned with the automatic extraction, analysis and understanding

of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, CV is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, CV seeks to apply its theories and models for the construction of CV systems.

Computer Vision was meant to mimic the human visual system, as a stepping stone to endowing robots with intelligent behavior. What distinguished computer vision from the prevalent field of digital image processing was a desire to extract three-dimensional structure from images with the goal of achieving full scene understanding. Studies in the 1970s formed the early foundations for many of the CV algorithms that exist today, including extraction of edges from images, labeling of lines, non-polyhedral and polyhedral modeling, representation of objects as interconnections of smaller structures, optical flow, and motion estimation.

Later, studies were published based on more rigorous mathematical analysis and quantitative aspects of computer vision. Researchers also realized that many of these mathematical concepts could be treated within the same optimization framework as regularization and Markov random fields. By the 1990s, some of the previous research topics became more active than the others. Research in projective 3-D reconstructions led to better understanding of camera calibration. At the same time, variations of graph cut were used to solve image segmentation. This decade also marked the first time that statistical learning techniques were used in practice to recognize faces in images (cf. Eigenface). Toward the end of the 1990s, a significant change came about with the increased interaction between the fields of computer graphics and CV. This included image-based

rendering, image morphing, view interpolation, panoramic image stitching and early light-field rendering.

Recent work has seen the resurgence of feature-based methods, used in conjunction with ML techniques and complex optimization frameworks (Sonka, Hlavac, & Boyle, 2008). The advancement of DL techniques has brought further life to the field of CV. The accuracy of DL algorithms on several benchmark CV data sets has allowed significant progress in tasks including, and not limited to:

Image Classification: Classify an image based on the dominant object inside it.

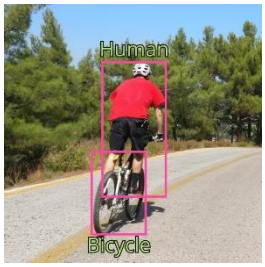
Example datasets: MNIST, CIFAR, ImageNet

Object Localization: Predict the image region that contains the dominant object. Then image classification can be used to recognize object in the region.

Example dataset: ImageNet

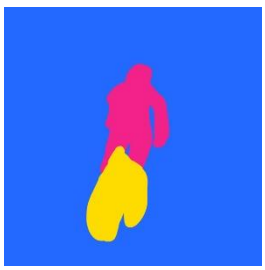
Object Recognition: Localize and classify all objects appearing in the image. This task typically includes proposing regions (ROIs), then classify the object inside them.

Example datasets: PASCAL, COCO



Semantic Segmentation: Label each pixel of an image by the object class that it belongs to.

Example datasets: PASCAL, COCO



Instance Segmentation: Label each pixel of an image by the object class and object instance that it belongs to.

Example datasets: PASCAL, COCO

Keypoint Detection: Detect locations of a set of predefined keypoints of an object, such as keypoints in a human body, or a human face.

Example dataset: COCO

CV is being used today in various real-world applications (Szeliski, 2010):

Optical Character Recognition: Reading handwritten postal codes on letters and automatic number plate recognition

Machine inspection: Rapid parts inspection for quality assurance using stereo vision with specialized illumination to measure tolerances on aircraft wings or auto body parts or looking for defects in steel castings using X-ray vision

Retail: Object recognition for automated checkout lanes

Photogrammetry: Fully automated construction of 3D models from aerial photographs

Automotive safety: Detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar or lidar do not work well

Match move: Merging CGI with live action footage by tracking feature points in the source video to estimate the 3D camera motion and shape of the environment

Medical imaging: Registering pre-operative and intra-operative imagery or performing long-term studies of people's brain morphology as they age.

The principles of computer vision are like ML tasks. The first step is to find a suitable representation of the content of a digital image so that features can be extracted and use them as an input for the algorithm. The next step is to train the algorithm on those features and produce a prediction on the content. Traditionally, there have been multiple techniques for feature extraction, such as Eigenfaces or Histograms of Oriented Gradients (HOG). After obtaining the feature vector, a classifier like Support Vector Machines (SVMs) can be trained on those structures and predict the class of an image or perform object detection by checking the existence or absence of them.

Chapter 4: Use Cases

Introduction

The focus of this thesis is to discuss the usage of a single DL framework which can be tweaked and applied to a diverse set of unique use cases.

The use cases examined in this project, Object Detection of mitotic figures in TUPAC16 (Veta M. H., 2016) and Instance Segmentation of histologic primitives (Nuclei, Epithelium, Tubules) in JPATHOL (Janowczyk, 2016), demonstrate how DL can be applied to a spectrum of the most common image analysis tasks in DP. We shall leverage the open source DL framework Detectron2 running on PyTorch, using the Mask R-CNN architecture pretrained on the COCO dataset.

We examine if a single training and model-building paradigm can be applied to each task, solely by determining and modifying its hyperparameters, and yet generate results that are comparable to competing DL solutions, or better than handcrafted approaches. This convergence to a unified approach not only allows for a low maintenance overhead, but also implies that image analysis researchers or DP users face a minimal learning curve, as the overall learning paradigm and hyperparameters remain constant across all tasks.

Object Detection on TUPAC16

Tumor proliferation is an important biomarker indicative of the prognosis of breast cancer patients. Patients with high tumor proliferation have worse outcomes compared to patients with low tumor proliferation (van Diest, 2004). The assessment of tumor proliferation influences the clinical management of the patient – patients with aggressive tumors are treated with more aggressive therapies and patients with indolent tumor are given more conservative treatments that are preferred because of fewer side-effects (Fitzgibbons, 2000).

Tumor proliferation in a clinical setting is traditionally assessed by pathologists. The most common method is to count mitotic figures (dividing cell nuclei) on hematoxylin & eosin (H&E) histological slides under a microscope. The pathologists will assign a mitotic score of 1-2-3, where a score of 3 represents high tumor proliferation. Although mitosis counting is routinely performed in most pathology practices, this highly subjective and labor-intensive task suffers from reproducibility problems (Veta M. H., 2016). One solution is to develop automated computational pathology systems to efficiently, accurately and reliably detect and count mitotic figures on histopathological images. Mitosis detection in WSIs is currently an active field of research (Albarqouni et al., 2016; Chen et al., 2016; Li et al., 2018; Tellez et al., 2018).

Description

This interest was in large part supported by the availability of public datasets in the form of medical image analysis challenges. The first challenge on the topic of mitosis detection was MITOS 2012 hosted at the International Conference of Pattern Recognition (ICPR) (Roux L. R., 2013). In 2013, Veta et al. organized AMIDA13 in conjunction with the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) (Veta M. V.-R., 2015). Mitosis detection was also one of the tasks of the MITOS-ATYPIA-14 challenge, organized as part of ICPR 2014, with the other task being scoring of nuclear atypia (Roux L. , 2014).

A large limitation of all previous challenges was that they focused solely on mitosis detection in predetermined tumor regions of interest (ROIs). However, in a real-world scenario, automatic mitosis detection is performed in WSIs and an automatic method should ideally be able to produce a breast tumor proliferation score given a WSI as input. To address the above problem, (Veta M. H., 2016). organized the Tumor Proliferation Assessment Challenge 2016 on prediction of tumor proliferation scores from WSIs.

The challenge had three main tasks to predict tumor proliferation:

1. Predict Mitotic Scores: Reproduce the most common method of assessing tumor proliferation by a pathologist.

2. Predict Gene Expression Based PAM50 Proliferation Scores: Determine whether molecular scores can be predicted from tissue morphology/WSIs.

3. ROI & Mitosis Detection: Design of a WSI tumor proliferation scoring system by following a two-step approach to emulate how a pathologist would assess a slide for tumor proliferation: identify ROIs followed by mitotic counting. This is the task and dataset we will examine.

The mitosis detection dataset consists of WSIs from 73 breast cancer cases from three pathology centers with annotated mitotic figures by consensus of three observers.

Of the 73 cases, 23 were previously released as part of the AMIDA13 challenge (Veta M. V.-R., 2015). These cases were collected from the Department of Pathology at the University Medical Center in Utrecht, The Netherlands. Each case was represented with varying number of HPFS extracted from WSIs acquired with the Aperio ScanScope XT scanner at 40× magnification with a spatial resolution of 0.25 μm/pixel. The remaining 50 cases previously used to assess the inter-observer agreement for mitosis counting were from two other pathology centers in The Netherlands (Symbiant Pathology Expert Center, Alkmaar and Symbiant Pathology Expert Center, Zaandam) (Veta M. H., 2016). Each case was represented by one WSI region with an area of 2 mm^2. These WSIs were obtained using the Leica SCN400 scanner (40× magnification and spatial resolution of 0.25 μm/pixel).

The annotated mitotic figures are the consensus of at least two pathologists, similar to the AMIDA13 challenge. In total, the mitosis detection auxiliary dataset contained 1552 annotated

mitotic figures. Of the 656 provided images, only the 587 of those that contained the annotated mitotic figures mentioned previously were used for training and validation; these images contained between 1 and 67 mitotic figures each.

The ROI & Mitosis Detection task was and related to the AMIDA13 challenge. The top scoring method for the third task had an F-score of 0.652 on mitosis detection. This is a slight improvement over the top scoring method of AMIDA13 challenge which had an F-score of 0.612.

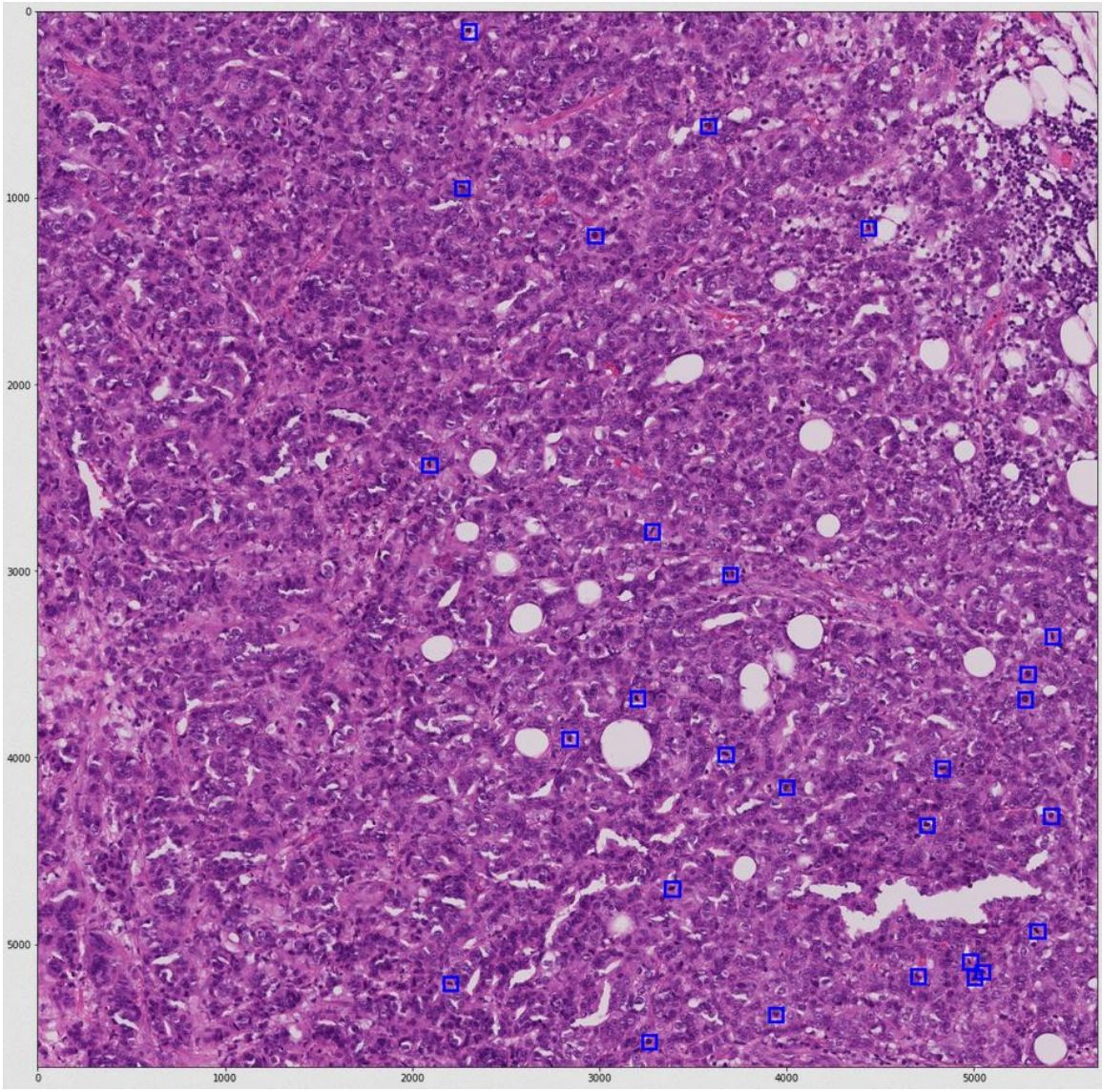


Figure 15: Example from the mitosis detection auxiliary dataset with annotated mitotic figures (blue bounding boxes). These annotated mitotic figures are the consensus of at least two pathologists.

Instance Segmentation on JPATHOL

The JPATHOL paper (Janowczyk, 2016) presents 7 use cases that represent the ensemble of critical components necessary for most of the pertinent pathology tasks and thus span the current challenges in the DP image analysis space split in 3 main task categories, each with their corresponding datasets. Our focus shall be on the segmentation tasks, where the delineation of accurate boundaries for histologic primitives (nuclei, epithelium, tubules) is required to extract precise morphological features.

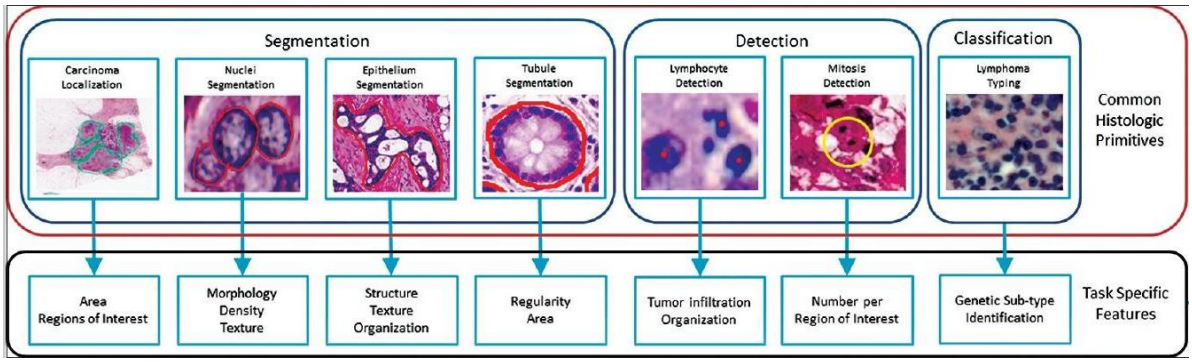


Figure 16: The flowchart shows a typical workflow for digital pathology research. Histologic primitives are identified, after which biologically relevant features are extracted for use in higher order research

The ground truth annotations are usually done by an expert delineating object boundaries or annotating pixels corresponding to a region or tissue of interest. In computational approaches, this level of annotation precision is critical so that supervised classification systems can be optimized. Generating these annotations, though, is often an arduous task due to the large amount of time and effort needed. Pathologists are typically not available to perform the large amounts of laborious manual annotations at the high resolutions needed for training and evaluating supervised object detection and classification algorithms. As a result, annotations are rarely pixel level precise, usually done at a lower magnification, and tend to contain numerous false positives and negatives.

Our goal is to tune a DL framework for three segmentation use cases (nuclei, epithelium, tubules), using all three datasets both separately and simultaneously, as seen in Table 1.

Table 1: Digital Pathology task descriptions

Task	Biological motivation	Dataset
Nuclei segmentation	Pleomorphism is used in current clinical grading schemes	141x2000x2000 @40× ROIs of ER+ BCa, containing subset of 12000 annotated nuclei
Epithelium segmentation	Epithelium regions contribute to identification of tumor infiltrating lymphocytes (TILs)	42x1000x1000 @20× ROIs from ER+ BCa, containing 1735 regions
Tubule segmentation	Area estimates in high power fields are critical towards BCa grading schemes	85x775x522 @40× ROIs from Colorectal cancer, containing 795 delineated tubules

Nuclei segmentation is an important problem because nuclei configuration is correlated with outcome, and nuclear morphology is a key component in cancer grading schemes. Manually annotating all of the nuclei in a single hematoxylin and eosin (H&E) stained estrogen receptor positive (ER+) breast cancer image is laborious and does not generalize to all of the other variances present by other patients and their stain/protocol variances. As a result, time is better invested annotating sub-sections of each image, even though this creates a challenging situation for generating training patches. Typically, one would use the annotations as a binary mask created for the positive class, and the negation of that mask as the negative class, randomly sampling from both to create a training set. In this case, though while one can successfully randomly sample from the positive mask, the randomly sampling from the complement image may or may return unmarked nuclei belonging to the positive class. To compensate, the standard approach is extended with intelligently sampled challenging patches for the negative class training set. Using a basic color deconvolution thresholding approach to select random negative patches further segmented nuclei are obtained, even though the network is unable to accurately identify nuclear boundaries. To enhance these boundaries, an edge mask is produced by morphological dilation and negative training patches are selected, which are inherently difficult to learn due to their similarity with the

positive class. This patch selection technique results in clearly separated nuclei with more accurate boundaries.

Epithelium identification is significant since regions of cancer are typically manifested there. Work by (Beck, 2011) suggest that histologic patterns within the stroma might be critical in predicting overall survival and outcome in breast cancer patients. Thus, from the perspective of developing algorithms for predicting prognosis of disease, the epithelium-stroma separation becomes critical. This task is unique in that it is less definitive than the more obvious tasks of mitosis detection and nuclei segmentation where the expected results are quite clear. Epithelium segmentation, especially the subcomponent of identifying clinically relevant epithelium, is typically done more abstractly by experts at lower magnifications. Due to discrepancies which can cause training and evaluation difficulties, an additional expert evaluation metric is considered to validate the results.

Given that the AlexNet approach constrains input data to a 32 × 32 window, the task is scaled to fit into this context. The principle is that a human expert should be able to make an educated decision based solely on the context present in the patch supplied to the DL network. This implies that an appropriate magnification must be selected a priori from which to extract the patches and perform the testing. Networks with larger patch sizes could use higher magnifications, at the cost of longer training times. Similar to the nuclei segmentation, the aim is to reduce the presence of uninteresting training examples in the dataset, which can be done by applying a threshold of 0.8 to the grayscale image, thus removing fat/background pixels from the patch selection pool. In addition, to enhance the classifier's ability to provide crisp boundaries, samples are taken from the outside edges of the positive regions.

Tubule segmentation can facilitate automation of the area estimation with decreasing inter-/intra-reader variances, and greater specificity, which can potentially lead to better stratifications associated with prognosis indication. Tubules are complex structures that not only consist of numerous components (e.g., nuclei, epithelium, lumen), but also their boundaries are determined by them. There is a large variance in the way tubules present given the underlying aggressiveness and stage of cancer. In benign cases tubules show in a well-organized fashion with similar size and morphological properties making their segmentation easier, while in cancerous cases the organization structure breaks down and accurately identifying the boundary becomes challenging even for experts. Also, tubules as an entity are much larger compared to the individual components, thus requiring a greater viewing area to provide sufficient context to make an accurate assessment.

In this use case, (Janowczyk, 2016) introduce the concept of using cheap preprocessing to help identify challenging patches: per image, a random selection of pixels belonging to both classes to act as training samples is made, and a limited set of texture features (i.e., contrast, correlation, energy, and homogeneity) is computed. Next, a naive Bayesian classifier determines posterior probabilities of class membership for all the pixels in the image. This way, pixels which would potentially produce false positives and negatives are identified and the training set is improved by removing trivial samples, without requiring any additional domain knowledge. Finally, knowing that benign cases are easier to segment than malignant cases, patches are disproportionally selected from malignant cases to further help with generalizability.

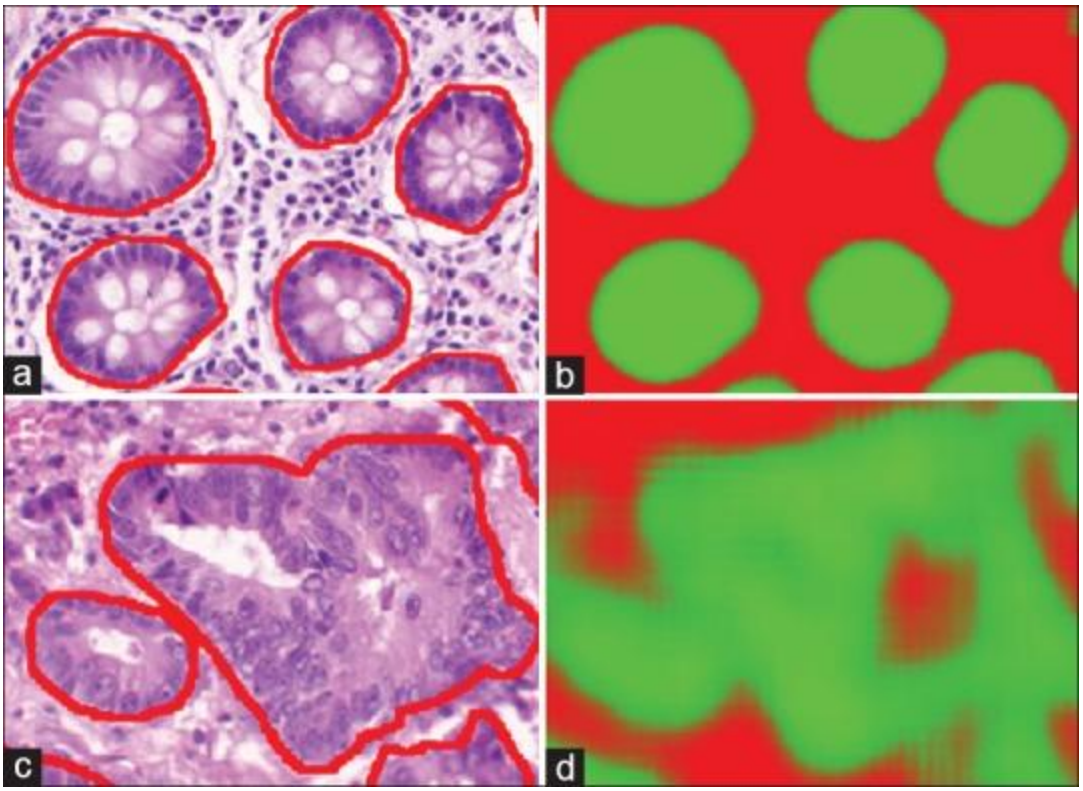


Figure 17: Benign tubules (a), outlined in red, are more organized and similar, as a result the model can provide clear boundaries (b). Malignant tubules (c) are abstract and detection is difficult (d).

Figure 17 shows that benign sections of tissue do well as a result of being able to generalize well from the dataset. Malignant tubules, on the other hand, are far more abstract and tend to have the hallmarks of a tubule, such as clear epithelial ring around a lumen, less obvious making them harder to generalize to. This is potentially one of the downfalls of machine learning techniques, which make inferences from training data; when insufficient examples are provided to cover all cases expected to be viewed in testing phases the approaches begin to fail. On the other hand, in this case, especially these challenges could be addressed by providing a larger database of malignant images.

Network Architecture

The DL network used for each of the individual tasks outlined in (Janowczyk, 2016) was a stock AlexNet architecture, identical to the one provided by Caffe. Its configuration is presented

in Table 2 and its hyperparameters shown in Table 3 were held constant for all tasks to illustrate how parameter tweaking and tuning was not important in achieving good quality results. Experiments using dropout showed no improvement in the results, and lack of overfitting evidence dissuaded an approach using dropout.

Table 2: AlexNet configuration

Layer	Type	Kernels	Kernel Size	Stride	Activation
0	Input	3	32x32		
1	Convolution	32	5x5	1	
2	Max Pool		3x3	2	ReLU
3	Convolution	32	5x5	1	ReLU
4	Mean Pool		3x3	2	
5	Convolution	64	5x5	1	ReLU
6	Mean Pool		3x3	2	
7	Fully Connected	64			Dropout + ReLU
8	Fully Connected	2			Dropout + ReLU
9	SoftMax				

Table 3: AlexNet hyperparameters

Variable	Setting
Batch size	128
Learning rate	0.001
Learning rate schedule	Adagrad
Rotations	0, 90
Num Iterations	600,000
Weight decay	0.004
Random minor	Enabled
Transformation	Mean-centered

Chapter 5: Experiment Results

The problem we are examining, the use of DL frameworks to run Object Detection and Instance Segmentation tasks on separate medical datasets with as limited configuration changes as possible, is split in the following subprocesses: framework selection, data wrangling, training & validation, and results evaluation.

Deep Learning Framework

Selection Process

Initially, given the two different medical datasets and tasks to be accomplished, and in order to familiarize with different DL frameworks and CNN architectures for educational purposes, it was decided to use TensorFlow (TensorFlow.org, n.d.) with MobileNet for the TUPAC16 object detection task, and PyTorch with Mask R-CNN-benchmark (FBAI, n.d.) for the JPATHOL instance segmentation task. Unfortunately, despite great effort to operate them concurrently but separately, issues with the Python environments and CUDA drivers made them virtually incompatible and were abandoned.

During the search for a single DL framework that could run both required tasks, Facebook AI Research released Detectron2 (Yuxin Wu, 2019), a next-generation platform for object detection and segmentation. Detectron2 is a ground-up rewrite of Detectron (originating from Mask R-CNN-benchmark), is powered by PyTorch, trains faster, and includes features such as Panoptic Segmentation, Densepose, R-CNN, rotated bounding boxes, and more.

Network Specs

The model utilized by Detectron2 is based on Mask R-CNN (He, 2017), a DL framework for instance segmentation. Instance segmentation combines elements from the classical CV tasks of object detection, where the goal is to classify individual objects and localize each using a

bounding box, and semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances. Mask R-CNN extends Faster R-CNN (Ren, 2015) by adding a parallel branch for predicting segmentation masks on each RoI, in parallel with the existing branch for classification and bounding box regression, adds only a small computational overhead, and surpasses all previous state-of-the-art single-model results on the COCO object detection and instance segmentation tasks.

Initially, the Region-based CNN approach was to attend to a manageable number of RoIs and evaluate CNNs independently on each of them. Fast R-CNN extended this to allow attending to RoIs on feature maps using RoIPool (Girshick, 2015), and Faster R-CNN advanced this by learning the attention mechanism with a Region Proposal Network (RPN). Faster R-CNN consists of two stages: a Region Proposal Network (RPN) which proposes candidate object bounding boxes and a Fast R-CNN that extracts features using RoIPool from each candidate box and performs classification and bounding box regression. The features used by both stages can be shared for faster inference. Mask R-CNN adopts the same two-stage procedure, with an identical first RPN stage, and in the second stage, in parallel to the class and box offset prediction, a binary mask for each RoI is also output, contrary to most recent systems where classification depends on mask predictions.

Mask R-CNN is comprised of two sections, the convolutional backbone architecture used for feature extraction over an entire image, and the network head for bounding box recognition and mask prediction that is applied separately to each RoI. The backbone architecture used in our implementation is called a Feature Pyramid Network (FPN) and employs a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R-CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid

according to their scale, but otherwise the rest of the approach is similar to vanilla ResNet. Using a ResNet-FPN backbone with a depth of 50 layers for feature extraction with Mask R-CNN gives excellent gains in both accuracy and speed. For the network head we extend the Faster R-CNN box heads from the ResNet and FPN papers, as shown in Figure 18.

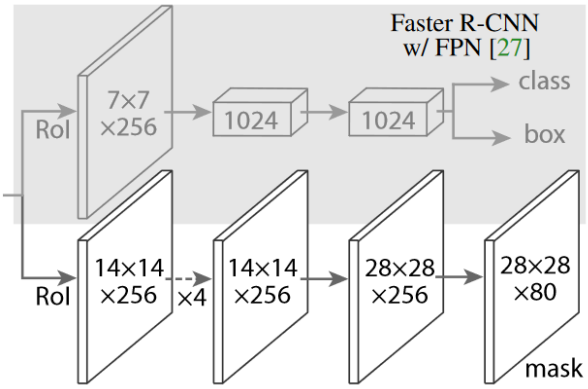


Figure 18: Heads for the ResNet-FPN backbone, with added mask branch. Numbers denote spatial resolution and channels.

The ResNet (Residual Network) is a CNN architecture most popularly used for image classification, is easy to optimize, and can gain accuracy from considerably increased depth. It was designed to ease the training of networks that are substantially deeper than older architectures using explicitly reformulated layers for learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. It tackles the vanishing gradient and degradation problems of learning by using residual mapping modules to form complete CNN networks, as shown in Figure 19 and Figure 20.

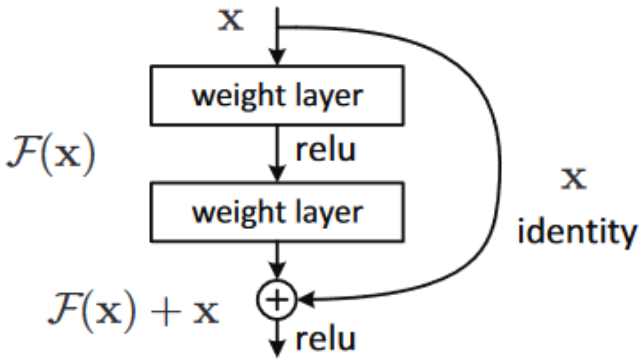


Figure 19: Residual learning: a building block.

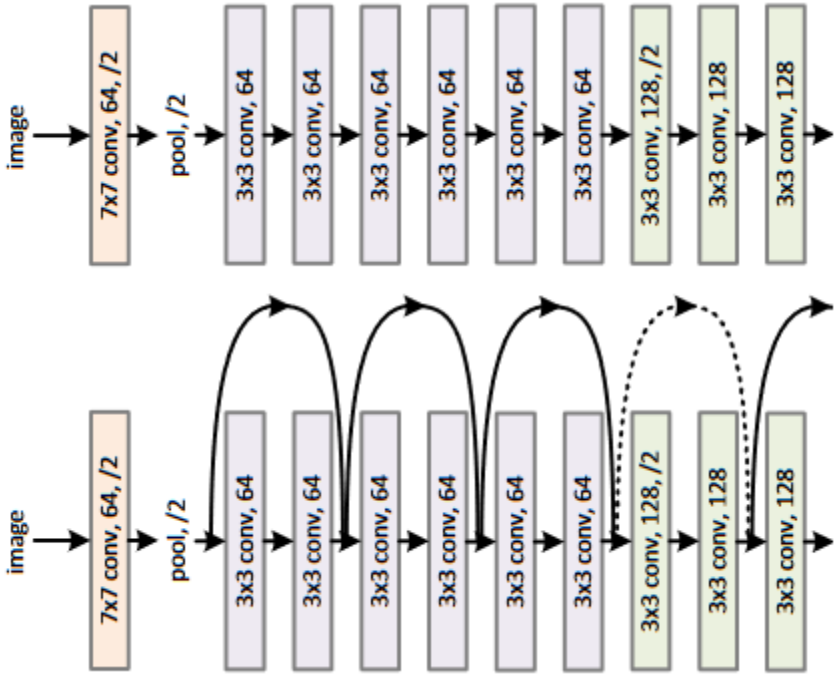


Figure 20: Example network architectures with plain (upper) and residual (lower) modules

The Detectron2 framework can be run within a Docker container, but due to the nature of the current problems involving CV and image processing, it was preferred to run it within Jupyter Notebooks in a separate Python 3.6 environment on a dedicated Linux 18.04 installation. The system running the setup is comprised of 6-core 3.00 GHz CPU, 16 MB 3000MHz DDR4 RAM, M.2 SSD and Nvidia RTX2070 8GB GPU.

Data Processing & Consumption

For standard tasks such as object detection or instance segmentation the standard representation for a dataset to be consumed by Detectron2 has a specification similar to COCO's JSON annotations. This means that apart from the folder with images to be processed, a JSON file with a list of dictionaries (one per image) must be provided, one for each subset of training, validation and testing data. The fields each image dictionary contains are the image file path, dimensions, a unique ID, and a list of annotations for every instance featured in the image. Each annotation contains the bounding box coordinates, the label and the segmentation mask of the

instance, either as a list of polygons or as a per-pixel bitmap segmentation mask in COCO's RLE format, as is our case. These JSON files are created after organizing and preprocessing the images, and arranging them into training, validation and test subsets.

TUPAC16: This dataset contains 73 WSI images in PNG format, annotated by a list of text files containing the coordinates of the central points of each mitosis figure present in each of the images. In total, the dataset contains 1552 mitotic figures.

Since we are performing object detection, but the annotations are coordinate tuples instead of bounding boxes or polygons, we created bounding boxes around each mitotic figure with side dimensions of 80px. This value was initially set roughly by viewing many images with their annotations overlaid and ensuring the bounding boxes contained a significant portion of the mitotic figures. At a later stage, the bounding box dimension was used as a hyperparameter for the model to be tuned for maximum accuracy.

Another phase of preprocessing was the splitting of images into smaller parts. As the input window for the model is 224 x 224 px, the initial image dimensions (2000 x 2000 px) would cause every image fed to the model to be resized leading to significant loss of information and detail due to antialiasing. All images with dimensions greater than the CNN window were contiguously split into 224 x 224 px subimages and saved separately in JPEG format, along with their corresponding annotations. After training/prediction, the subimages may be rejoined to recreate the original image along with their annotations, if needed (Figure 21).

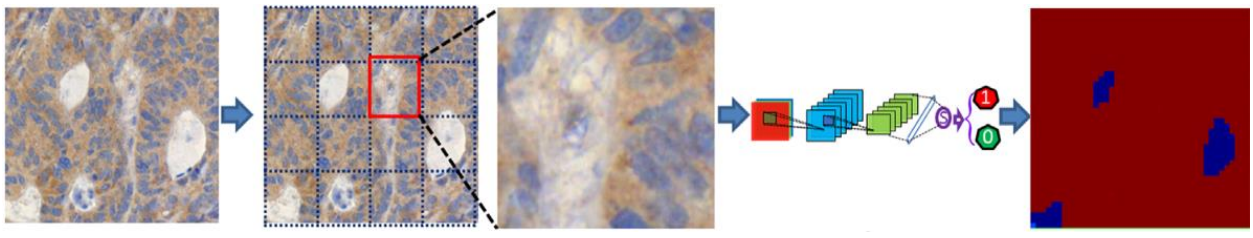


Figure 21: Subimage splitting process

Any resulting subimages smaller than the CNN window were padded with black pixels to conform with the dataset. Subimages with no annotations were removed from the dataset, as they would not be used for training by the model by default, and the rest were split into training, validation and testing subsets with a 0.6:0.2:0.2 ratio. To facilitate formation of the JSON file, a CSV file was created during this phase containing info on all subimages and annotations.

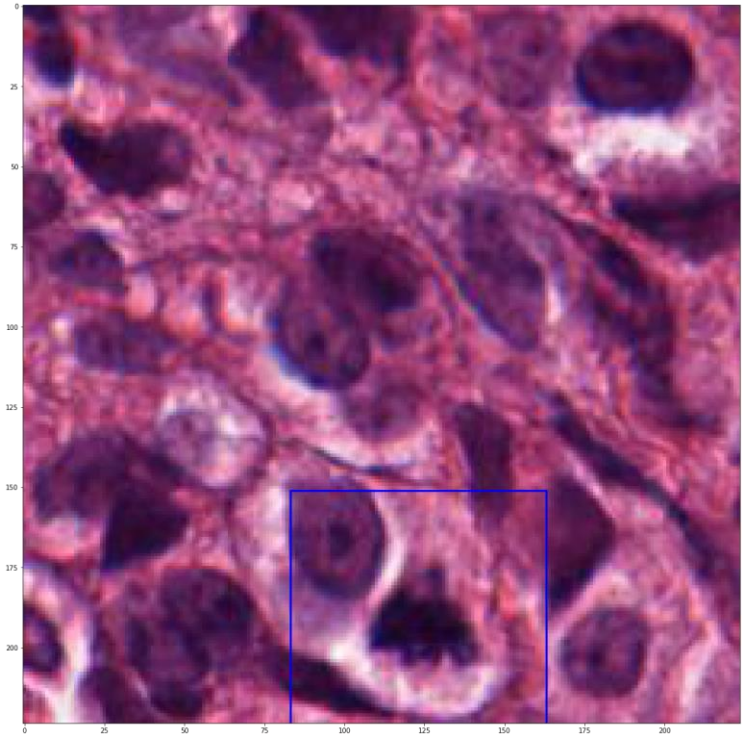


Figure 22: Sample subimage (224 x 224 px) resulting from splitting original image into CNN-window dimensions. It is apparent a much greater degree of detail is retained for training.

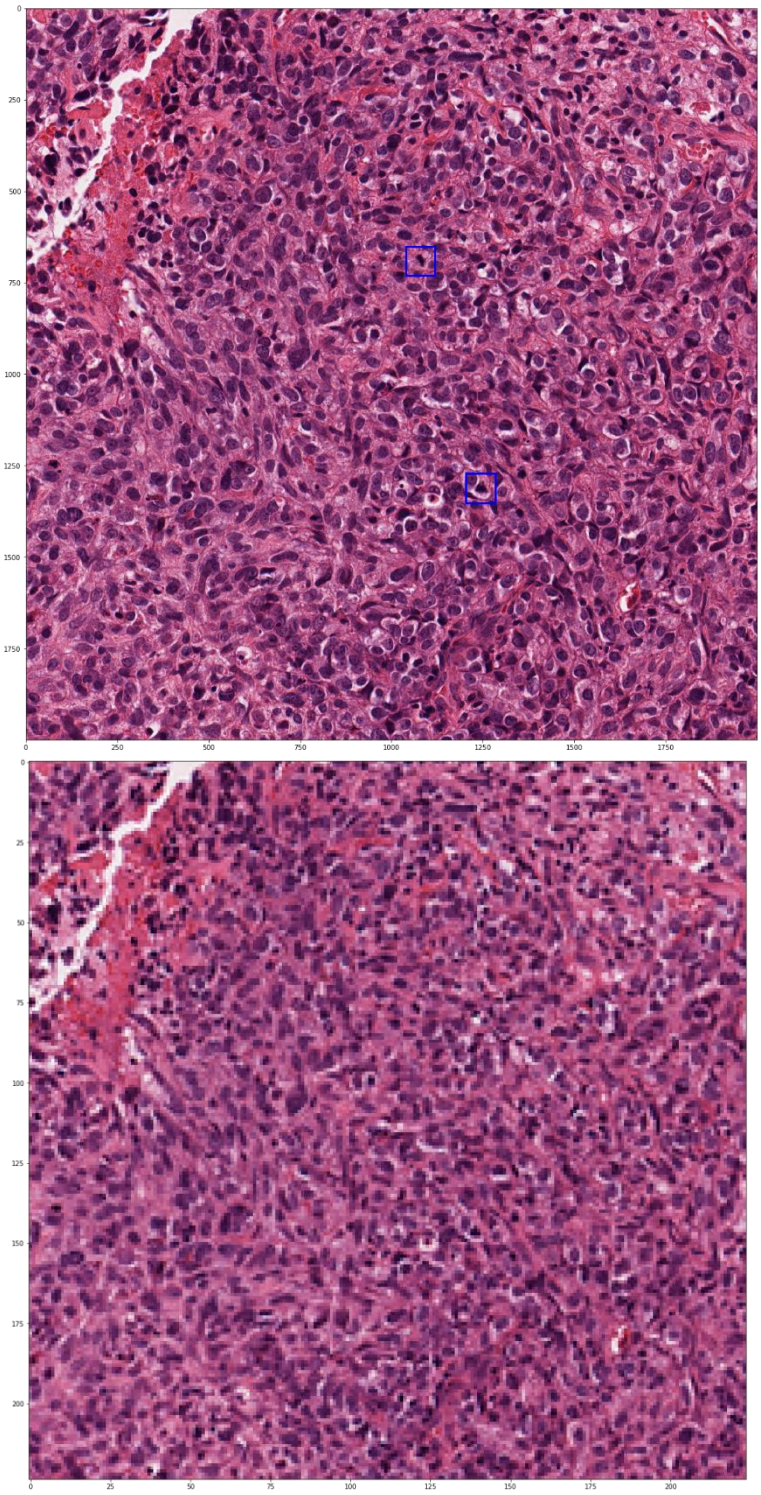


Figure 23:(Up) Sample image of the initial dataset with original dimensions (2000 x 2000 px). Bounding boxes (blue) are centered around the mitotic figure coordinates to contain its greatest portion possible. (Down) Same image resized to CNN window dimensions (224 x 224 px). Note the loss of detail.

JPATHOL: This dataset is comprised of three separate medical datasets featured in (Janowczyk, 2016), prepared for experimentation with nuclei, epithelium and tubule segmentation tasks. Each dataset is comprised of images of different count, format, dimensions, and magnification, as found in Table 1. The first stage in preprocessing this dataset is to ensure all datasets are of the same magnification, so the epithelium images were zoomed in 2×, resulting in their new dimensions 2000 x 2000 px. Also, since the images in the tubule dataset do not have the same height as width, they are padded with black pixels in order to make them rectangular without affecting their aspect ratio or magnification, changing from 775 x 522 px to 775 x 775 px. The bitmap annotations undergo the same process in order to retain correct ground truth format.

The second phase, as with the TUPAC16 dataset, is splitting the images into smaller subimages analogous to the CNN window dimensions, in order to avoid excessive resizing and information loss. In this case, however, we shall resize to 261 px as the least common denominator of the three data sets' dimensions to reduce cases of subimages consisting of mostly black padding.

The final phase, before the JSON file formulation, is to prepare the segmentation masks for each instance and image. The ground truth for all three datasets is provided as 1- or 3-channel bitmap files of dimensions equal to their corresponding images, which are transformed into single channel binary arrays and are encoded into COCO RLE format with the Pycocotools library.

After this process, the imageset consisting of subimages of the three datasets is randomly split into training, validation and testing subsets with a 0.6:0.2:0.2 ratio. As before, subimages with no annotations are discarded. For each subset, a JSON file is prepared containing all necessary metadata outlined previously.

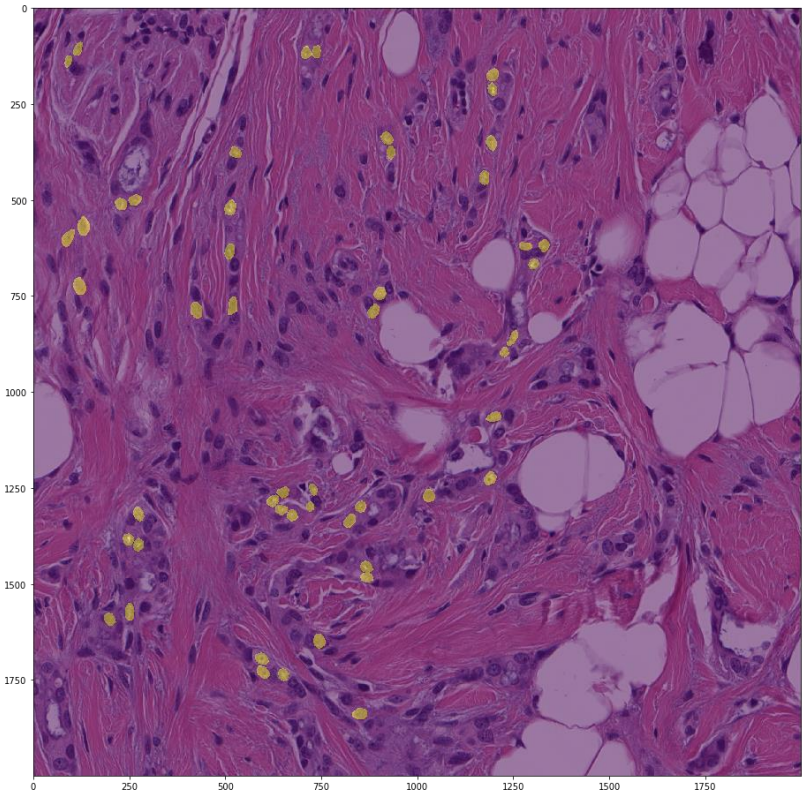


Figure 24: Sample image from the nuclei dataset in its original dimensions. The nuclei are annotated with yellow pigment – not all of them are annotated

Parameterization & Training

After the images have been preprocessed and registered, they can be consumed by Detectron2 to train and evaluate an instance segmentation model. This model is based on Mask R-CNN with a ResNet R50-FPN backbone, pretrained on the COCO dataset (trained on Train2017 and evaluated on Val2017). In order to train the model on our datasets with the maximum possible accuracy, we must tune the model's hyperparameters. The hyperparameters tuned in our research are presented in Table 4, along with the range of values they were tested with, their initial values, and their final values.

Table 4: Detectron2 hyperparameter values

Hyperparameter	Tuning Range	Starting Value	Optimum Value
Number of workers	[2, 8]	2	8
Images per batch	[2, 8]	2	4
Learning Rate	[0.00025, 0.1]	0.00025	0.025
Max iterations	[300, 12000]	300	9000
Batch size per image	[128, 1024]	128	512
Testing threshold	[0.01, 0.7]	0.7	0.07
Bounding box dimension	[40, 120]	80	80
Subimage split dimension	[224, 1000]	1000	500

Finding the optimum values is a process that can be done either manually or automatically. For the automatic approach, a script is prepared that iteratively trains models using all the consecutively or randomly chosen points in the hyperparameter grid, whose limits are suggested values found in the documentation, or just arbitrary within a logical scope. The downsides of this approach are the time cost, as many models far from the optimum values are needlessly trained, and there is limited overview concerning script errors. With the manual approach we start training models using random or suggested initial hyperparameter values and we try to follow a fashion of gradient descent towards the optimum values by changing only one or two values for every iteration. The downside of this approach is the possibility of getting stuck in a local minimum in the hyperparameter space.

During the hyperparameter tuning phase, the resulting models are evaluated with the validation subset exclusively. For each hyperparameter value combination we train and evaluate the same model multiple times so that we can get more precise mean and standard deviation values of each model's accuracy, mitigating the variance caused by the CNN's stochastic nature. The evaluation stage returns 12 performance metrics, as used by COCO; our decision process is based on Average Precision for the whole area (AP), traditionally called Mean Average Precision (mAP):

```
Average Precision (AP) IoU=0.50:0.95 | area= all   | AP
Average Precision (AP) IoU=0.50      | area= all   | AP50
Average Precision (AP) IoU=0.75      | area= all   | AP75
Average Precision (AP) IoU=0.50:0.95 | area= small | APs
Average Precision (AP) IoU=0.50:0.95 | area=medium | APm
Average Precision (AP) IoU=0.50:0.95 | area= large | APl
Average Recall  (AR)   IoU=0.50:0.95 | area= all   | AR
Average Recall  (AR)   IoU=0.50:0.95 | area= all   | AR50
Average Recall  (AR)   IoU=0.50:0.95 | area= all   | AR75
Average Recall  (AR    IoU=0.50:0.95 | area= small | ARs
Average Recall  (AR)   IoU=0.50:0.95 | area=medium | ARm
Average Recall  (AR)   IoU=0.50:0.95 | area= large | ARl
```

TUPAC16: One of the major hurdles in histopathology image analysis is the variability of tissue appearance. The staining color and intensity can be significantly different between WSIs due to variation in tissue preparation, staining and digitization processes. To address this, for most similar tasks staining normalization is performed as a preprocessing step. The most used method is the one proposed by (Macenko, 2009), where an unsupervised method heuristically estimates the absorbance coefficients for the H&E stains for every image and the staining concentrations for every pixel; afterwards, normalization is performed by recomposing the RGB images from the staining concentration maps using common absorbance coefficients.

We attempted to approximate this method by normalizing the 3-channel RGB histograms of the images jointly and separately. This greatly distorted the color balance and distorted the visual features, resulting in significantly lower accuracy compared to the initial unprocessed images. This method was rejected after some trial runs on the TUPAC16 dataset.

JPATHOL: Given the size and complexity of this dataset, the Detectron2 hyperparameters for this task were taken directly from the previous task, as they are a good baseline and the datasets are of similar structure. Normalization was not performed as it was found to not offer any improvements in accuracy. Dropout during training was also not performed as in JPATHOL

experiments showed no improvement in metrics and (Srivastava N, 2014) requires a smaller optimal dataset.

The important element of this dataset is the experimentation it requires concerning patch generation. The generation of annotations for a dataset such as this is a cumbersome process, due to the large amount of time and effort needed. For example, the nuclei annotation dataset used in this work took over 40 hours to annotate 12,000 nuclei, and yet represents only a small fraction of the total number of nuclei present in all images. Unfortunately, this creates a challenging situation for generating training patches. Typically, one would use the annotations as a binary mask created for the positive class, and the negation of that mask as the negative class, randomly sampling from both to create a training set. In this case, however, while one can successfully randomly sample from the positive mask, the randomly sampling from the complement image may or may return unmarked nuclei belonging to the positive class.

Consequently, extended image patches need to be generated to more fully represent the available but unannotated ground truth in the training and validation sets. This stage requires modest domain knowledge in order to ensure a good representation of diversity in the training set. Selecting appropriate image patches for the specific task could have a dramatic effect on the outcome. Especially in the domain of histopathology, there can be substantial variance present within a single target class, such as nuclei. This is especially pronounced in breast cancer nuclei, where nuclear areas can vary upwards of 200% between nuclei. Ensuring that a sufficiently rich set of exemplars is extracted from the images is perhaps one of the most key aspects of effectively leveraging and utilizing a DL approach.

For each of the three classes of images in JPATHOL a detailed description of approaches is suggested that allows for tailoring of training sets towards improving the specific detection tasks.

Nuclei Segmentation: A standard approach involves selecting patches from the positive class and using a threshold on the color-deconvolved image to determine examples of the negative class. This rationale is based on the fact that non-nuclei regions tend to weakly absorb hemotoxin. The resulting network has very poor performance in correctly delineating nuclei since these edges are underrepresented in the training set.

To compensate, the above approach is extended with intelligently sampled challenging patches for the negative class training set. Through identifying positive pixels and the basic color deconvolution thresholding approach to select random negative patches, the segmented nuclei are obtained. However, the network may be unable to accurately identify nuclear boundaries, so an edge mask is produced by morphological dilation where negative training patches are selected. A small proportion of the stromal patches is still included to ensure that these exemplars are well represented in the learning set. This patch selection technique results in clearly separated nuclei with more accurate boundaries.

Epithelium Segmentation: Similar to the nuclei segmentation task, the presence of uninteresting training examples in the dataset has to be reduced, so that learning time can be dedicated to more complex edge cases. Epithelium segmentation can have areas of fat or the white background of the stage of the microscope removed by applying a threshold at conservative level of 0.8 to the grayscale image, thus removing those pixels from the patch selection pool. In addition, to enhance the classifiers ability to provide crisp boundaries, samples are taken from the outside edges of the positive regions, as discussed for the nuclei segmentation task.

Tubule Segmentation: In this use case, a form of primitive preprocessing is used to help identify challenging patches. Per image a number of pixels belonging to both classes is randomly selected to act as training samples and compute a limited set of texture features (i.e., contrast,

correlation, energy, homogeneity). Next, a naive Bayesian classifier determines posterior probabilities of class membership for all the pixels in the image, and pixels are identified which would potentially produce false positives and negatives and would benefit from additional representation in the training set. These pixels are selected based on their magnitude of confidence, such that false positives with posterior probabilities closer to 1.0 are selected with greater likelihood than those with 0.51. This approach further helps to bootstrap our training set, by removing trivial samples, without requiring any additional domain knowledge. Finally, knowing that benign cases are easier to segment than malignant cases, patches are disproportionally selected from malignant cases to further help with generalizability.

For experimental purposes, the problem was not treated as individual single-class instance segmentation tasks but as a combined multi-class instance segmentation task, where the detector would have to both classify the input image to the correct class and detect and segment their instances. Unfortunately, different focus did not allow implementation of the above processing suggestions on any of the subtasks at this time. The expected model performance would be very low, given that the ground truth available had limitations. However, as this part was more of an implementation for proof of concept, the performance is considered not as important as having a functioning model. Implementation of those suggestions, with or without use of additional knowledge, is expected to dramatically increase the performance of the model to a level where it could raise interest for further research.

Results

Here we present the performance results for the final (and some intermediate) models trained for the two different datasets and tasks. All the results presented are Average Precision metrics run on the validation subset, which was chosen to be 20% of the whole initial dataset. The

first experiments with low AP were run just a few times, but as AP improved multiple runs were made (up to 12 runs) per hyperparameter combination in order to achieve a more precise average value and standard deviation. Outliers were discarded, and it was found that at least 5 runs were required per hyperparameter combination to achieve a stable result. For each of the two final models only, a separate testing subset (also 20% of the whole initial dataset) was used to compute their Average Precision and performance on previously unseen data.

TUPAC16

Table 5: Results for object detection task on TUPAC16

Images/ batch	Learning Rate	Max iterations	Batch/ image	Split dimension	Train Time (H:M:S)	AP (%)	STDEV (±%)
4	0.025	600	512	1000x1000	0:04:56	43.32	6.58
4	0.025	600	1024	1000x1000	0:05:23	40.11	8.65
8	0.025	600	1024	1000x1000	0:11:35	43.79	5.69
6	0.010	9000	1024	1000x1000	2:02:55	55.74	3.26
4	0.025	9000	512	1000x1000	1:13:18	58.11	2.96
4	**0.025**	**9000**	**512**	**500x500**	**1:11:49**	**65.02**	**4.05**

In Table 5 a few key hyperparameter value combinations are displayed, along with the AP their model achieved on the validation set. Also mentioned is the average training time spent for each. In all, more than 40 hyperparameter combinations were used to train an equal number of models, and more than 170 runs were done to validate these models.

The optimum combination of hyperparameters that resulted in the model with the highest Average Precision is the last row of Table 4. The training time required is around 70 mins and the Average Precision achieved on the validation set was 65.02 ± 4.05%. On the unseen testing subset the Average Precision is 65.14%, which shows overfitting was avoided. The average F-score for this model is $F_1 = 2\frac{AP \cdot AR}{AP+AR} = 0.628$.

Image Predictions: In Figure 25 are presented a few random images from the test set, with the ground truth bounding boxes (left), juxtaposed to bounding boxes predicted by the model (right):

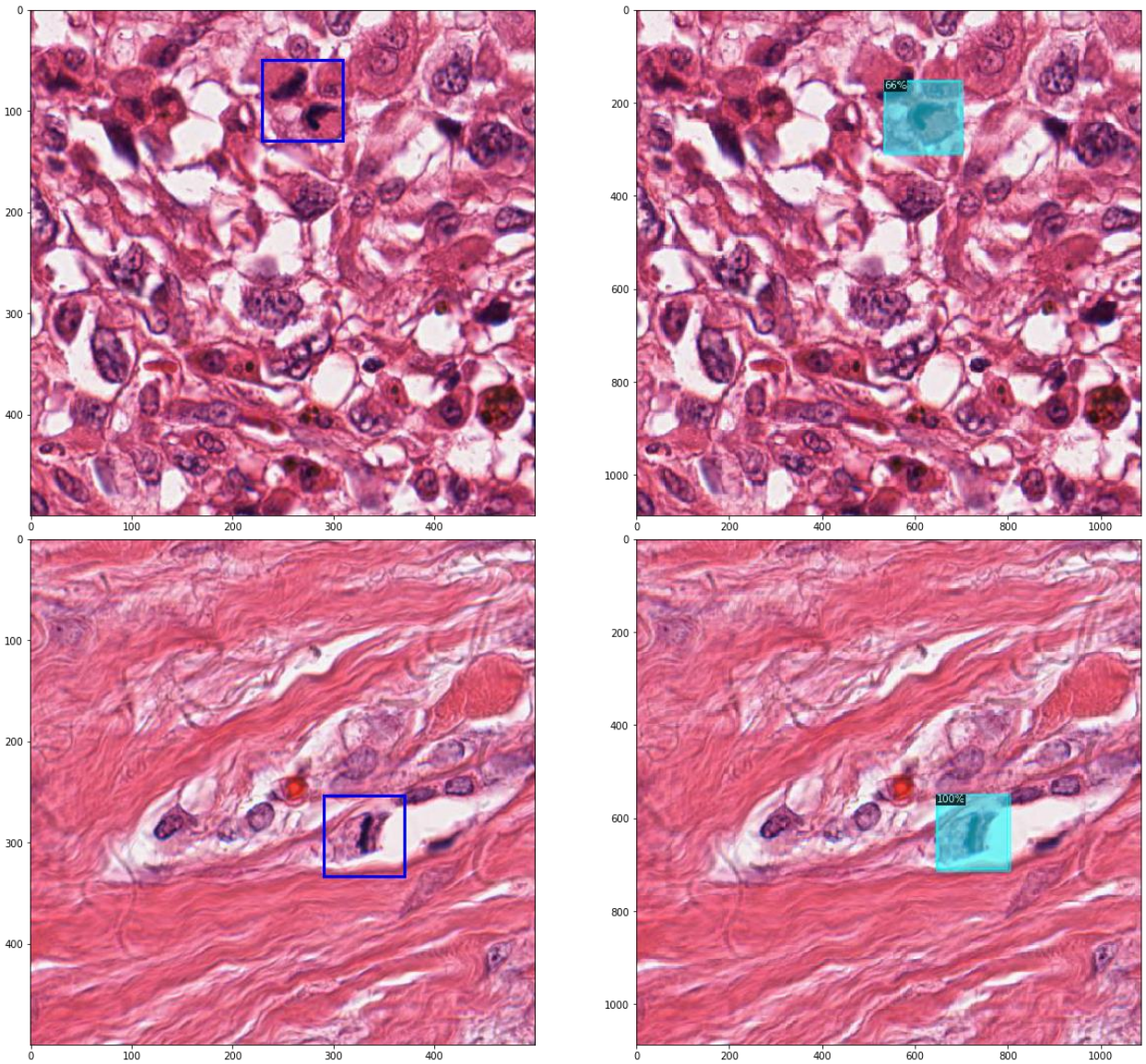
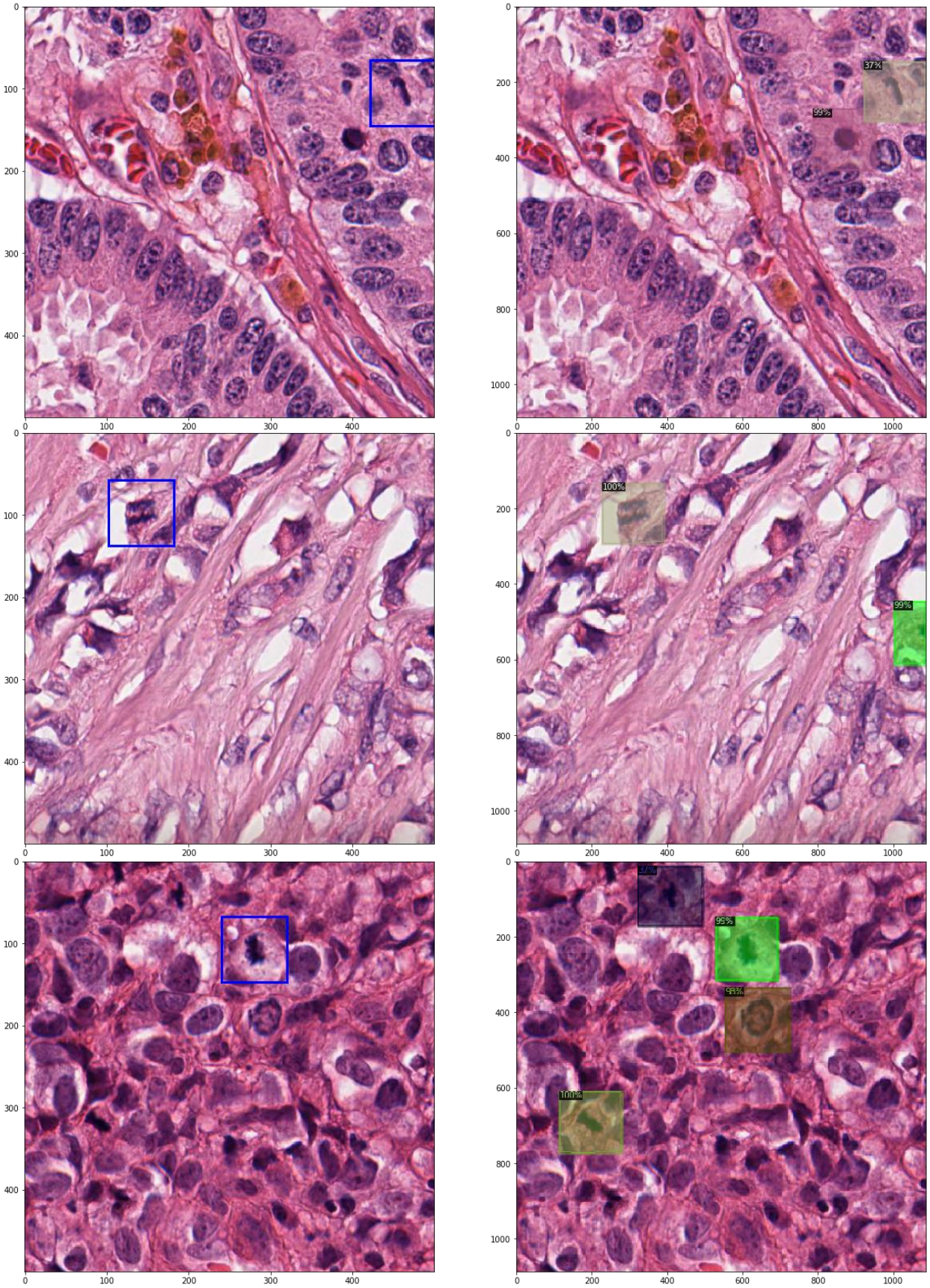


Figure 25: Sample WSIs from the test set with annotated ground truth (left) and predictions (right)


37%
99%
100%
99%
97%
95%
98%
100%

JPATHOL

Table 6: Results for instance segmentation task on JPATHOL

Dataset	AP (%)	STDEV (±%)
Nuclei	19.53	1.68
Epithelium	5.15	3.21
Tubule	35.22	5.16
Nuc/Epi/Tub	**8.03**	**2.09**

Table 6 presents the performance of the models trained on each of the Nuclei/Epithelium/Tubule datasets separately, as well as of the model trained on all of them simultaneously. All models were trained on the optimum hyperparameters found from the TUPAC16 task (Table 4), apart from *max iterations* which had to be tuned for each independently to avoid cases of disappearing gradients. Since significant hyperparameter tuning was not needed, only around 20 hyperparameter value combinations were evaluated. Run count was also limited to less than 50 due to lack of substantial performance improvement.

The metrics in Table 6 are based on evaluation of the model on the validation subsets. On the unseen testing subset the model's metrics are:

```
'AP'       : 14.41
'AP50'     : 23.72
'AP75'     : 15.31
'APs'      :  5.16
'APm'      : 22.42
'APl'      : 17.67
'AP-Nuc'   : 15.26
'AP-Epi'   :  2.06
'AP-Tub'   : 25.91
'F1-score' : 0.220
```

Image Predictions: In Figure 26 are presented a few randomly selected images of the test set with the ground truth segmentation masks (left), juxtaposed to segmentation masks predicted by the model (right):

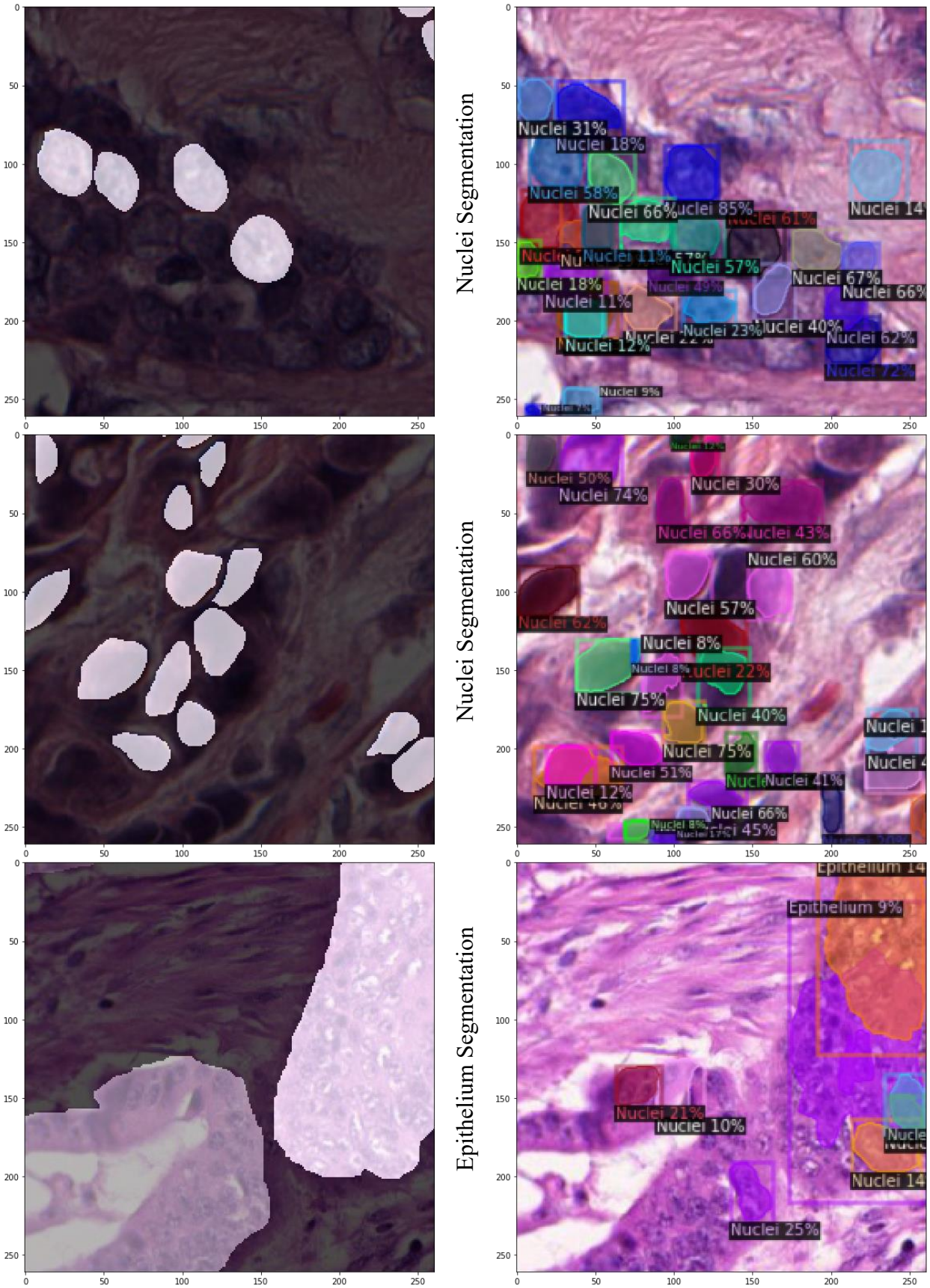
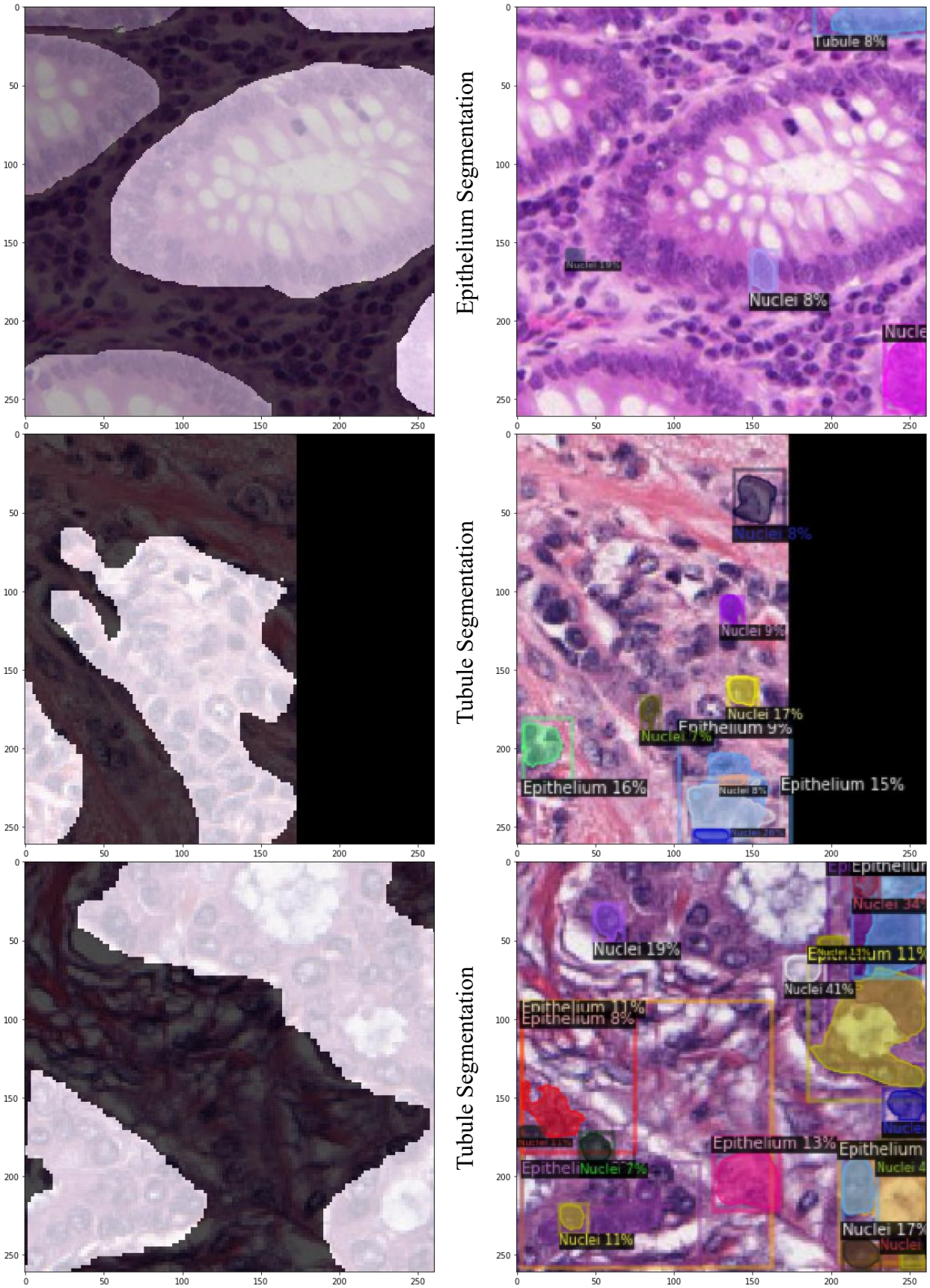


Figure 26: Sample WSIs from the test set with annotated ground truth (left) and predictions (right)


Epithelium Segmentation
Tubule Segmentation
Tubule Segmentation
Tubule 8%
Nuclei 8%
Nuclei 8%
Nuclei 9%
Nuclei 17%
Epithelium 9%
Nuclei 7%
Epithelium 16%
Epithelium 15%
Nuclei 19%
Nuclei 41%
Epithelium 11%
Epithelium 8%
Epithelium 13%
Nuclei 11%
Nuclei 17%

Chapter 6: Discussion

State of the Art

TUPAC16: All teams that performed mitosis detection as part of the Tumor Proliferation Assessment Challenge used CNNs (IMAG/e, 2016). Most teams trained a two-class classification model with patches centered at a mitotic figure, and background patches. On the testing dataset, the model evaluated every pixel location and produced a mitosis probability map that could be further processed to identify mitotic figures and/or produce a mitotic score for a ROI. The neural network architectures applied to this problem vary from relatively shallow with only a few convolutional layers to deep ResNets. Team results for the challenge are shown in Table 7.

Table 7: TUPAC16 Mitosis Detection team entries & results vs. ours

Team	F-score	Additional data
HUST, Wuzhen, China	0.669	No
Lunit Inc., Korea	0.652	No
IBM Research, Zurich and Brazil	0.648	ICPR 2012/14
University of Warwick, UK	0.640	No
University of Piraeus, Greece	0.628	20% less data
Chinese University of Hong Kong	0.620	ICPR 2012/14
Contextvision, Sweden	0.616	No
IBM CODAIT, USA	0.601	ICPR 2012/14
Microsoft Research Asia, China	0.596	ICPR 2012/14
PIEAS, Pakistan	0.571	ICPR 2012/14
Radboud UMC, The Netherlands	0.541	No
University of Alberta, Canada	0.487	No
University of Heidelberg, Germany	0.481	No
University of South Florida, USA	0.440	No
Shiraz University of Technology, Iran	0.330	No
Inha University, Korea	0.251	No
Instituto Politécnico Nacional, Mexico	0.135	No
IIT Madras, India	0.017	No

Since mitoses are generally rare events, the mitosis detection problem is very unbalanced. Two main strategies were used to mitigate this: data augmentation by geometric transformations

and hard negative mining. The mitosis detection problem is invariant to rotations, flipping and small translation and scaling, so can be exploited to create new plausible training samples to enrich the training data. The other strategy was hard negative mining (Cireşan, 2013) which is a boosting-like technique where an initial mitosis detection method is trained with random sampling for the background class, and then used to detect difficult negative instances that are used to train a second method. In practice, models trained with random sampling for the background class result in many false positives since all hyperchromatic objects are detected as mitoses. The output of the initial mitosis detection method can be used to sample such difficult background samples and train a second mitosis detection method, which can lead to improvements of the mitosis detection accuracy.

JPATHOL: For the three instance segmentation tasks described in JPATHOL, its authors provided some initial processing methodologies to promote further research on their datasets.

Nuclei Segmentation: Using the procedure outlined in Chapter 4 and an AlexNet Network structure, the authors developed a 5-fold cross-validation set of about 100 training and 28 testing images. Qualitatively, the network returns crisper results at 40× magnification, rather than at 20×. Quantitatively, the detection rate, i.e., the ability to find nuclei in the image, is very high, with the network identifying 98% of all nuclei at the 40× magnification, and dropout appears to negatively impact the metrics, as presented in Table 8.

Table 8: Nuclei Segmentation results

Method	Detection	F-score
20×	0.95	0.80
20× + Dropout	0.90	0.79
40×	0.98	0.83
Our model - 40×	0.14	0.220

Epithelium Segmentation: Five folds of 34 training and 8 test images are preprocessed as mentioned above and are passed through the same AlexNet framework as with the previous task. The threshold is used as a hyperparameter for each fold in search of the best possible F-score. Pathologists often treat this task as a higher-level abstraction instead of a pixel level classification, without removing white background pixels. The approach followed by (Janowczyk, 2016) can identify smaller regions ignored by pathologists because they are considered clinically irrelevant. After review of their results by a clinical collaborator, they were found to be suitable for use in conjunction with other classification algorithms, for example prognosis prediction. This is one of the first attempts at direct segmentation and quantification of epithelium tissue in breast tissue. The mean F-score for 5-fold cross-validation is 0.84.

Tubule Segmentation: Each of the 5-fold cross validation sets has about 21 training images and 5 test images. The mean F-score, using a threshold of 0.5, was 0.827 ± 0.05. When optimized with threshold on a per fold basis this measure rose slightly to 0.836 ± 0.05. Combining all the test sets together, the p-value equals 0.33, indicating that there was no significant difference between the expected clinical grade associated with our approach versus and expert's ground truth annotation. Two other state-of-the-art approaches claim 86% accuracy and 0.845 object-level dice coefficient.

Further research for use of similar DL techniques on this dataset could not be found.

Discussion of our results

TUPAC16: Our object detection model results can be considered satisfactory when examined on their own. An F-score of 0.628 gives us a good baseline for further fine-tuning and improving the model. Training and inferring with this model is fast; acceptable models can be obtained within an hour, and prediction results are near instantaneous. Average accuracy is also

satisfactory with limited data preprocessing and fine-tuning. Even examining the prediction images shows that many false predictions can be considered to resemble human-like mistakes, especially acknowledging that part of the ground truth mitoses may not have been annotated at all, conceivably generating several false negatives.

JPATHOL: Unfortunately, our instance segmentation model could only be judged as a first-stage proof of concept. The data pipeline with preprocessing, training and inferring is functioning, but the final model performance metrics are mediocre, both for the three datasets separately, as well as jointly. The image predictions corroborate our numerical performance metrics, demonstrating our model's difficulty in detecting particularly the Epithelium class, and any smaller instances in general. Also, they present multiple overlapping bounding box predictions from different classes, which should not be a possible outcome with these datasets.

Comparison with SOTA

TUPAC16: Compared to the challenge entries presented in Table 7, the performance of our model is found to be more than adequate, giving it fourth place in the challenge standings. Our model's performance stands out even more if we consider the technical limitations, the lack of experience and unavailability of domain knowledge we had to overcome. Given that the entries with better F-scores are close to ours, it is hopeful that with a better process and the improvements outlined below we might be able to gain at least a couple of places in the challenge.

A further fact to be considered is that the other entries were evaluated with an external unpublished testset, whereas our model was evaluated with a subset of the published dataset, giving us a ~20% smaller training subset. Also, some entries used external datasets to further augment their training subset and help with generalization, while we selected to focus on technical specifications and settings. Last, our approach used an off-the-shelf, easily set and tuned DL

framework, instead of custom ANN architectures intertwined with complex non-DL CV procedures.

JPATHOL: A direct comparison between our approach and the presented methods in JPATHOL would be an unproductive effort. The methods presented for each of the three datasets are focused on a preliminary experimentation to arrive at a basic process for training single-class instance segmentations models using partially annotated data and non-exhaustive tuning. Our goal was to create a multi-class instance segmentation model, which, though functioning, could not reach nearly the performance of the above methods, mainly due to lack of experience and domain knowledge.

The tasks examined are solutions for different sets of problems, and the presented methods should only be viewed as facilitators in the development and improvement of our model, not as competing proposals. As a functioning proof of concept, our model can be used as a testbed for further improvements suggested below and its class performance could be methodically evaluated using the JPATHOL results as benchmarks.

Future Work

TUPAC16: The trained model has been found to be a good base for further experimentation, despite some oversights during its development. Improvements to be considered:

- **Data augmentation:** The primary oversight, due to wrongful assumptions, was the omission of a data augmentation process. As the TUPAC16 dataset is invariant to rotation, flipping and scaling, it would be beneficial to incorporate this in our process and enlarge our training subset.
- **Hyperparameter grid:** Instead of a manual path of gradient descent in the hyperparameter grid, and given the short training duration of most models, automated scan of a

hyperparameter grid in either a randomized or sequential fashion could help dodge local minima during training, or give us a tighter range of hyperparameter values to be tested more thoroughly.

- **Different CNN models:** Detectron2 has many pretrained models available for testing, some of which may retrieve faster or more accurate predictions. Thanks to its structure, swapping different models is a very straightforward process.

- **Staining normalization:** Due to variability in tissue appearance and lack of standard staining processes, different staining normalization methods could be tested to diminish these differences. This could help with working on training data from different sources and annotators.

- **Full annotations:** The annotations of this dataset do not cover all instances of mitosis, which will weaken our model's training ability and performance. Therefore, steps should be taken, as previously shown, to extract as accurately and as thoroughly possible any other mitosis instances present in the dataset. This could be undertaken as an unsupervised learning problem with non-DL techniques, such as SVM or boosting.

- **Domain knowledge:** A medical expert would be invaluable in inspecting our initial dataset and annotations, as well as visually evaluating our annotation extraction process and predictions.

- **Evaluation on external dataset:** Our best model can be retrained using the current testing subset along with the training subset for training. The final evaluation can be done by submitting it to the TUPAC16 organizers who will run it on a separate unseen test set. This will both slightly improve its performance and give us an official place in the challenge standings.

- **Evaluation of training process:** Tensorboard will allow us to track, visualize and log any parameters or metrics of the CNN in real time.

- **Rejoining images:** As an operational feature, prediction subimages could be joined with the rest of the related subimages to form the initial full-resolution image.

JPATHOL: The main problem that encumbered the development of an adequate model is the backwards process we followed. In retrospect, we should have started with development of three separate instance detection models trained on each of the sub-datasets separately, then three single-class models on all three sub-datasets, and, eventually, the one multi-class model. This way, the necessary preprocessing for each dataset could be determined more exactly, any deficiencies in the datasets could be rectified earlier, and any possible performance loss due to instance class or size can be examined more systematically.

Further improvements would be the same as those mentioned for the TUPAC16 challenge, but the one that would bring the greatest performance enhancement in this case is the refinement of separate preprocessing schemes for each class of data. Our approach made minimal use of class-specific processing, which hindered its training capability. Starting with the suggestions stated in the JPATHOL article, we should create processes that can extract accurate and thorough annotations and reduce uninteresting training features. Also, for instances like Epithelium and Tubule, which as entities are much larger compared to individual components, a greater viewing area than 261 x 261 px may be needed to provide sufficient context to make an accurate assessment.

These improvements would not necessarily make it a useful framework for medical staff in real-world applications, but the proof of concept will stand and could lead to further advances in multi-class classification and detection tasks in the Digital Pathology field.

Chapter 7: Conclusions

The main goal of this thesis was to gain valuable knowledge and experience in the domain of Deep Learning, especially where it may be applied to the field of Digital Pathology. Two separate medical image datasets were chosen for that objective, and two different problems were formulated around them, one for object detection and one for instance segmentation.

We selected an open source and easily customizable DL framework specialized in Computer Vision tasks, Detectron2, which proved to be a solid choice. Its flexibility and tunability allowed quick training/evaluation cycles which led to a very competent model for the TUPAC16 challenge. Using that model as a starting point for the JPATHOL instance segmentation task, we managed to develop a functioning, though performance lacking, multi-class segmentation model. A range of improvements has been suggested, most of which are straightforward, that are expected to significantly boost our model's performance.

Summarizing, we have successfully developed a baseline model for single-class object detection tasks, as well as a proof-of-concept model for multi-class instance segmentation tasks based on WSI/medical histology slide datasets.

References

Ballard, D. H., & Brown, C. M. (1982). *Computer Vision.* Prentice Hall ISBN 978-0-13-165316-0.

Beck, A. H. (2011). Systematic analysis of breast cancer morphology uncovers stromal features associated with survival. *Science translational medicine, 3(108)*, 108ra113-108ra113.

Cireşan, D. C. (2013). Mitosis detection in breast cancer histology images with deep neural networks. *International conference on medical image computing and computer-assisted intervention* (pp. 411-418). Berlin: Springer.

CS, S. (n.d.). *CS231n Convolutional Neural Networks for Visual Recognition*. Retrieved from Convolutional Neural Networks (CNNs / ConvNets): https://cs231n.github.io/convolutional-networks/#comp

Erickson, B. J. (2017). Machine learning for medical imaging. *Radiographics, 37(2)*, 505-515.

FBAI. (n.d.). *Mask R-CNN*. Retrieved from https://github.com/facebookresearch/maskrcnn-benchmark

Fitzgibbons, P. L. (2000). Prognostic factors in breast cancer: College of American Pathologists consensus statement 1999. *Archives of pathology & laboratory medicine 124*, 966-978.

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE international conference on computer vision*, (pp. 1440-1448).

He, K. G. (2017). Mask R-CNN. *Proceedings of the IEEE international conference on computer vision*, (pp. 2961-2969).

Huang, T. V. (1996). Computer Vision : Evolution And Promise. *19th CERN School of Computing* (pp. 21–25). Geneva: CERN ISBN 978-9290830955.

IMAG/e, M. I. (2016). *Tumor Proliferation Assessment Challenge 2016*. Retrieved from Results Task 3: Mitosis detection: http://tupac.tue-image.nl/node/62

Janowczyk, A. &. (2016). Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *Journal of pathology informatics, 7*.

Kayalibay, B. J. (2017). CNN-based segmentation of medical imaging data. *arXiv:1701.03056.* Retrieved from arXiv:1701.03056.

Kraus, O. Z. (2016). Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics, 32(12)*, i52-i59.

Krizhevsky, A. S. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097-1105.

Macenko, M. N. (2009). A method for normalizing histology slides for quantitative analysis. *IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (pp. 1107-1110). IEEE.

Madabhushi, A. (2009). Digital pathology image analysis: opportunities and challenges. *Imaging in medicine*, p. 7.

Nielsen, M. (2019, 12). *Neural Networks and Deep Learning*. Retrieved from Chapter 6: Deep learning: http://neuralnetworksanddeeplearning.com/chap6.html

O'Mahony, N. C. (2019). Deep learning vs. traditional computer vision. *Science and Information Conference* (pp. 128-144). Springer.

Ren, S. H. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 91-99.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review, 65(6)*, 386.

Roux, L. (2014). *MITOS-ATYPIA-14 contest home page*. Retrieved from https://mitos-atypia-14.grand-challenge.org/

Roux, L. R. (2013). Mitosis detection in breast cancer histological images An ICPR 2012 contest. *Journal of pathology informatics, 4*.

Sonka, M., Hlavac, V., & Boyle, R. (2008). *Image Processing, Analysis, and Machine Vision.* Springer.

Srivastava N, H. G. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J Mach Learn Res*, 1929-58.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* Springer Science & Business Media.

TensorFlow.org. (n.d.). Retrieved from https://github.com/tensorflow/tensorflow

van Diest, P. J. (2004). Prognostic value of proliferation in invasive breast cancer: a review. *Journal of clinical pathology, 57(7)*, 675-681.

Veta, M. H. (2016). Predicting breast tumor proliferation from whole-slide images: the TUPAC16 challenge. *Medical image analysis, 54*, 111-121.

Veta, M. V.-R. (2015). Assessment of algorithms for mitosis detection in breast cancer histopathology images. *Medical image analysis, 20(1)*, 237-248.

Yuxin Wu, A. K.-Y. (2019). *Detectron2*. Retrieved from https://github.com/facebookresearch/detectron2