Metastases Detection in Lymph Nodes using Transfer Learning

Master's Thesis Computing Science – Data Science

MICHEL KOK

August 2019

SUPERVISORS

RADBOUD UNIVERSITY, ICIS

Prof. dr. ir. Arjen de Vries

Chris Kamphuis, Msc

RADBOUD UNIVERSITY MEDICAL CENTER, DIAG

Dr. Geert Litjens,

Pétér Bándi, Msc

SECOND READER

Dr. Francesco Ciompi

Radboud University

IN·DEI·NOMINE·FELICITER

Radboudumc university medical center

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PROBLEM STATEMENT

Cancer is the leading cause of death worldwide [1]. When someone is diagnosed with cancer, chances of survival plummet once the cancer has metastasized, as metastasized cancer is responsible for over 90% of cancer-related deaths. Different types of cancer spread via blood or via the lymphatic system from its tumorous origin to other parts of the body.

Challenges like CAMELYON16 [2] indicate the potential of machine learning techniques for pathology. The goal of this challenge was to detect metastasized breast cancer in the first lymph node where the lymph drains to from the tumor region, the so-called *sentinel* lymph node. In clinical practice, once somebody is diagnosed with breast cancer the sentinel lymph node is often used to investigate whether the tumor has metastasized, as this is most likely to contain tumor cells [3]. The sentinel lymph node is excised and analyzed by a pathologist. Given that this examination is tedious and time-consuming, the CAMELYON16 challenge was hosted [4, p. 2].

Submissions to the CAMELYON16 challenge achieve similar effectiveness as "an expert pathologist interpreting slides without time constraints" [2, p. 2208]. Follow-up studies have outperformed the winners' solution [5] and have even led to a digital assistant called LYNA designed to be used in clinical practice [6].

Most submissions to the CAMELYON16 challenge exploit deep convolutional neural networks (25 of 32) – including the submission yielding the highest effectiveness [2]. These networks are trained to perform well on a specific task. However, these networks can sometimes be finetuned to related tasks [7]. The learned information is transferred to a new task, hence the umbrella term of *transfer learning*. For example, many networks are first trained to recognize images by training on the ImageNet dataset[8], and then trained on a specific image analysis task at hand.

In general, in this master's thesis project, it is asked *to what extent transfer learning can be applied when creating a network for the CAMELYON16 challenge and transferring the information to different cancer metastases*. Specifically, four aspects corresponding with the four subquestions below are singled out when trying to answer this question.

1.1.1 Architecture

Regarding the submissions to the CAMELYON16 challenge, it was concluded that the chosen architecture only plays a marginal role, and that "auxiliary strategies to improve system generalization and performance seemed more important" [9, pp. 2205–2207]. These strategies for example deal with stain variations or address class imbalance [9]. For transfer learning the network should be able to grasp the data structures in the CAMELYON16 dataset well, before it can even be used to do transfer learning. Smaller architectures are easier and faster trained with less data. On the other hand, there is less capacity to store the structure of the data. So, ideally, there is a minimal architecture that fits the data well enough.

Liu et al. used Inception (V3) [10] in their solution. This architecture has been used to achieve the current state of the art results on the CAMELYON16 dataset [5], [6]. They experimented with several 'versions' of Inception V3. A slimmed-down architecture, for example, uses fewer neurons per layer and is therefore less prone to overfitting and computationally less heavy. The number of trainable parameters is reduced

from roughly 200 million to 300,000. In Keras [11] or Tensorflow [12] this can be achieved by setting the following two parameters: *depth_multiplier=0.1* and *min_depth=16*. Performance only slightly decreased when using this slim architecture.

However, this still results in a many-layered architecture exploiting ingenious structures. To see whether a simple, straightforward convolutional neural network will do, the first subquestion is *if the architecture can be made smaller and simpler while maintaining performance when auxiliary strategies remain the same.*

1.1.2 Training set size

Transfer learning is often applied in image analysis in a straightforward manner. First, a model is trained on some source task. Then, a network is created to train for the target task and, importantly, this network is initialized with values learned during the source task. Transfer learning comes in handy when there is limited training data for the second task or when the data is expensive. The second model already incorporates much information from the source task before training is started, and thus its weights only have to be finetuned. Machine learning problems in the domain of histopathology often deal with small datasets, and corresponding annotations are expensive. This domain is therefore suitable for transfer learning.

Usually, transfer learning in the histopathological domain only occurs where the first task is to train on the ImageNet challenge [8]. The task is to learn to classify an image in one of 1000 classes, e.g. dog or car. However, the ImageNet dataset does not contain histopathological images. Datasets of different types of cancer in lymph nodes are more closely related than ImageNet as the lymph nodes are the main part of both datasets and only the specific tumor regions are specific per dataset. Datasets containing histopathological images may therefore be preferable to NatureNet as a source dataset.

Depending on the domain and how related the source and target tasks are, a different amount of training data is required to adequately finetune the network to the new task. An important aspect when transferring weights is how big the size of the training set should be. In general the smaller the training set used to finetune the network the better, as transfer learning can then be applied to small, related datasets. Therefore, the second subquestion is *to what extent training set size matters when the straightforward approach to transfer learning is applied between different histopathological systems.*

1.1.3 Learning rate

The following two aspects focus on the fact that convolutional neural networks are connectionist systems. This entails that there is no separate memory except for the connections in the model. When training a network, the weights on the connections are optimized towards the task at hand. However, when finetuning towards a new task, the connections are altered and thus performance on the old task decreases. This effect is more severe the more the network is allowed to change its parameter configuration during transfer learning. This phenomenon is referred to as catastrophic forgetting.

This is not necessarily a problem. Most people that use ImageNet as their source model do not care about their performance on the ImageNet challenge, but only about performance on the target task. High target performance can mostly be achieved faster and cheaper by using the learned structures in the data from the source model. However, sometimes performance on the initial task needs to be contained. For example, there are certain benefits to having one single model that can detect different types of cancer

in the lymphatic system. The model does not miss rare cancer types and can do with much less training data compared to using many models where every model detects only one specific cancer type.

One of the ways to overcome this is to use a very small learning rate. This prevents the network from altering the weights with high values, as it can only marginally tune them in the right direction. On the other hand, the network may not be able to learn the new dataset as well as would have been the case with a high learning rate. This leads to the third question: *what is the effect of the learning rate when doing transfer learning?*

1.1.4 Elastic Weight Consolidation

Another, more sophisticated method is Elastic Weight Consolidation [13]. It is one of the recent techniques that has successfully been applied in several domains. This does not require any alternative learning rate or architecture but uses a different loss function. This loss function updates parameters in such a way that the new solution is as close as possible to the solution space of the source task. Using this alternative loss function, EWC aims to achieve good performance on both the source and target task. Therefore, the fourth question is *to what extent we can prevent forgetting using EWC*.

For the remainder of this chapter, the background of this project is provided. In the second chapter three datasets are described, each containing lymph nodes of different regions. To answer the research questions, experiments are done which are described in chapter 3 and of which the results are provided in the fourth chapter. Finally, the last chapter answers the research questions using the obtained results.

1.2 BACKGROUND

The background of this project consists of three parts. As the majority of solutions to the CAMELYON16 challenge used deep convolutional neural networks, these are first described. Then, the exact definition of transfer learning is given, including the description of EWC. Third, related work is discussed in general.

1.2.1 Deep convolution neural networks

As with all supervised machine learning techniques the aim is to learn a model or function $f(x, \theta) = y$ that is good at predicting the right label y for unseen data samples x. Often these functions have the additional input θ which is typically used to denote parameters or weights of the function. Fitting such a function means finding the optimal weights θ for f. This function should resemble the data distribution in general. For example, when faced with a classification problem, samples from class A should be separated from samples of class B by the hyperplane described by f. Underfitting f would mean a function that does not adequately distinguish between elements from A and B. Overfitting f means that all training samples are perfectly distinguished but that the model is bad at distinguishing unseen test data. This means that f has learned the training set 'by heart' instead of the general distribution of the data.

1.2.1.1 Training procedure

Giving input (x, θ) to the model yields a prediction label $\hat{y}$. The difference between this prediction and the ground truth is computed by a loss function. Given the loss function L_i for a training sample, a cost function J can be defined as well:

$$J = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{1}$$

where there are N samples in the training set. During training the aim is to find the global minimum of this cost function. In other words: to obtain predictions of the whole dataset that are as close to the ground truth as possible.

To find this global minimum, optimizers or optimization functions are used. Almost all currently used optimizers are variants of gradient descent [14]. A training procedure is typically started with random weights, resulting in a random position somewhere on the cost function. The best direction along which the weights should be changed towards, is then given by the gradient ∇ of the cost function. Especially for layered models that have many parameters, computing gradients can be computationally heavy. Weights are then altered as follows:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2}$$

where η is the learning rate parameter. This influences by how much the gradient should update the current weights. When the loss function is differentiable the gradient computation is efficient and feasible with the backpropagation algorithm that exploits the chain rule [15].

Often the dataset is too big to fit into memory at once, so the data is loaded into memory sequentially via mini-batches. Gradient descent can then be defined for n training examples as follows:

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{3}$$

However, (3) is vulnerable to get stuck in a local minimum, especially when a low learning rate is used. Simply increasing the learning rate is often not an option, as this causes the algorithm to make bigger steps. These big steps prevent from gradually descending down the slope and approaching the minimum. This makes it harder to find the exact global minimum as only big steps are taken along the cost function. Another drawback is that every weight is updated according to the same learning rate.

One of the most used variants of the intuitive gradient descent function is Adaptive Moment Estimation (Adam) [16]. Basically, this method computes adaptive learning rates for each parameter in θ based on previously seen gradients. This generally results in faster convergence and works well in practice [14].

In summary the procedure consists of three steps. First a mini-batch is sampled from the training dataset. Second, a so-called forward pass in which predictions are computed on the mini-batch. The cost as a combination of losses is then computed by comparing the predictions to the ground truth. Last, the gradients of the cost function are computed by the backpropagation algorithm which are used to update the parameters of the model.

1.2.1.2 Convolutional neural network models

Solutions to the CAMELYON16 challenge use a procedure like this to distinguish tumor from non-tumor cells. The contemporary state-of-the-art models obtained by this procedure comprise millions of trainable parameters and are called convolutional neural networks.

1.2.1.2.1 Artificial neural network

Artificial neural networks consist of a number of layers that each contain artificial neurons. An artificial neuron receives several input values with associated weights and a bias. As datasets are possibly non-linear, the neuron applies a non-linear function to this weighted input.

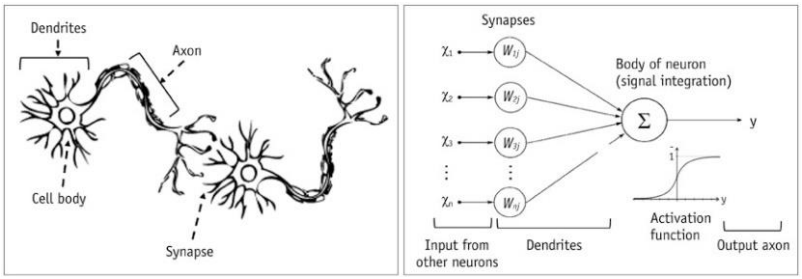


Figure 1.1: Analogy between real and artificial neurons by Lee et al. [10]

Formally an artificial neuron can be described as

$$y = f\left(\sum_i w_i x_i + b\right) \tag{4}$$

where $w \in \theta$ and b is a bias which can be used to shift the activation function f.

The layers of neurons are connected by using the output of layer n as input for neurons in layer $n+1$. The first layer usually receives the input from an external data source and a random set of weights, while the last layer outputs the final prediction per class. The term 'deep learning' or 'deep networks' often refers to networks using multiple (or many) hidden layers in the model, see Figure 1.2. The last layer contains the number of output classes.

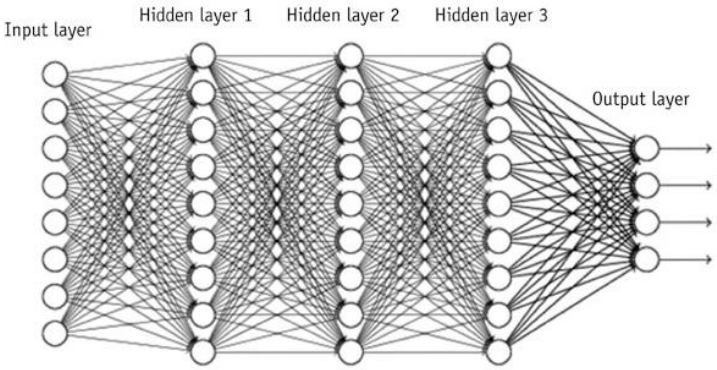


Figure 1.2: Fully connected layered neural network by Lee et al. [10]

1.2.1.2.2 Convolutional neural network

A special type of neural networks specifically designed for image analysis are convolutional neural networks (CNN). The input of a convolutional neural network is a numerical matrix representing an image and many layers in a convolutional neural network are 2D-convolutions.

A convolutional operation requires an input matrix and a kernel (often also referred to as filter) and adding the element-wise multiplication as illustrated in Figure 1.3 [17]. The values in the kernel correspond to the weights in an artificial neural network. The aim of the training procedure is therefore to find the best values of the filters. The result of a convolutional operation is a feature map. This feature map exists of values that have information of multiple pixels which is one of the big advantages when treating images. The layers are not fully connected as not all input values are necessary to compute one output value. The values in the feature map thus contain information of a region of the original input matrix. Due to the convolutional operation the feature map size is always reduced with respect to the input matrix. Typically when many convolutions have to be done, the input matrix is padded before the convolution to compensate for these decreasing sizes.

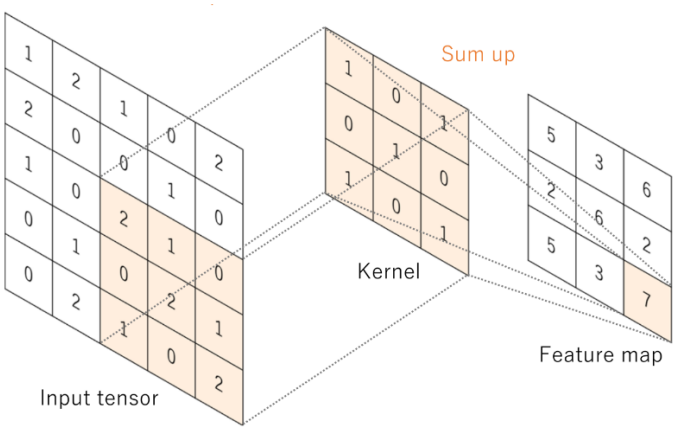


Figure 1.3: The convolution operation by Yamashita et al. [9]

Convolutional layers are often followed by a pooling layer. This layer connects e.g. a region of 2x2 in the feature map and outputs the maximum or average value of this region. The network will become less sensitive to small shifts or distortions in the data and the size of the feature maps is effectively reduced [18]. A contemporary CNN builds up an increasingly complex representation of the data over the layers and contains millions of parameters. With such numbers the network can easily overfit on the training data. Therefore, a third important type of layer worth mentioning is a dropout layer, that randomly drops neurons [19]. This forces the network to not rely on single neurons and keep generalizing the learned patterns.

1.2.1.2.3 Contemporary example: Inception V3

Although successors have been are already around for multiple years already [20], Inception V3 is still widely used and achieves many state-of-the-art results in imaging challenges – like CAMELYON16. Inception V4 adds more Inception modules to the architecture, and the architecture as a whole is a little bit simplified [20]. Liu et al. show that Inception V3 can already adequately fit the dataset – even with less parameters than the standard Inception V3 network prescribes [5]. Both Inception-ResNet V1 and V2 typically require less training data to converge, but their effectiveness is generally a little lower than Inception V3. So, the successors seem to target tasks that need deeper models or have smaller datasets. So, given Inception V3 produces the current state-of-the-art result, and that there is enough data to train a model that is less sophisticated than the standard Inception V3, the focus here lies on Inception V3.

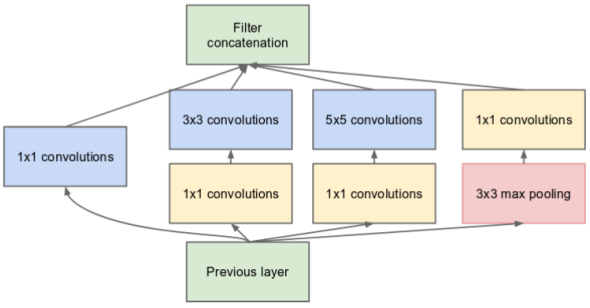


Figure 1.4: Inception module [13].

The first Inception network paper describes the so-called Inception module which is depicted in Figure 1.4 [21]. This module makes the network wider instead of only deeper as there are a number of different convolutional operations at the same level which are then concatenated and given as input to the next layer. This means that the network is able to learn which size of convolutions contributes most without the network becoming extremely deep.

Deep networks can suffer from a vanishing gradient problem [22]. As the output of most activation functions is between 0 and 1, and the gradients are computed via the chain rule, the gradients in the first layers become exponentially smaller with a high number of layers. As a result, these layers – or eventually the whole network – could stop learning. One-by-one convolutions reduce input channel size and are computationally cheaper than doing 5x5 convolutions. Successors of Inception V1 implemented several variants of this module that are mostly computationally cheaper. E.g. replacing a 5x5 convolution with two 3x3 convolutions reduces computation already and then doing 3x3 convolutions sequentially as a 3x1 and 1x3 convolution reduces computation time again. Inception architectures stack 9 or 10 of these modules onto each other which makes the network still deep. So, the new way of computing the convolutions in the Inception module together with some technical solutions to minimize effects of the vanishing gradient and speed up computation describe the changes from Inception V1 to Inception V3 [10].

1.2.2 Transfer learning

A contemporary CNN often has many layers and thus many trainable weights. During training the weights of a CNN are optimized towards the specific task at hand. In recent years, many techniques have been proposed to transfer these weights from some source domain to a different target domain [7].

1.2.2.1 Formal definition

Weiss et al. emphasize the importance of the relation between a source and target domain when they describe transfer learning as "a need to create a high-performance learner for a target domain trained from a related source domain" [7, p. 1] and "for transfer learning to be feasible, the source and the target domains must be related in some way." [7, p. 19] In fact, humans tend to learn in a similar fashion. For example, a skilled guitarist will learn to play the piano much faster than a non-musician, as the guitarist can *transfer* his musical knowledge from playing guitar to playing piano. In this case, the transfer is made possible as there is the common higher domain of music that links both subdomains. In the same fashion, a pathologist specialized in lymph nodes with breast cancer, can with relatively little effort learn to detect colon cancer metastases in lymph nodes as well. So, although the specific types of cancer are different, there is the shared upper domain of the lymphatic system.

Transfer learning can formally be defined as follows: a domain D consists of a feature space χ (i.e. the dataset) and marginal probability distribution $P(X)$ where $X = \{x_1, \dots, x_n\} \in \chi$. A learning task T can then be defined by the label space y and a predictive function $f(\cdot)$. Transfer learning happens between a given D_{source} and corresponding T_{source} on the one hand and D_{target} and T_{target} on the other. The goal is to optimize $f_{target}(\cdot)$ where $D_{source} \neq D_{target}$ or $T_{source} \neq T_{target}$. In the case of detecting metastasized cancer in different lymph nodes, the dataset is different but the task remains the same. Because $D_{source} \neq D_{target}$ it follows that $X_{source} \neq X_{target}$ and this is what Weiss et al. call heterogeneous transfer learning [7].

The predictive function $f_{source}(\cdot)$ contains the iteratively updated weights after training on the source domain. Common practice when doing transfer learning with CNNs is to use the network that is trained on the source dataset and continue to train on the target dataset with additional iterations of updating the network weights. So, the weights parameter of $f_{target}(X_{target}, \theta_{target})$ is not initialized at random – as is usually done – but initialized with θ_{source}. When training, a lower learning rate can be used as the network already performs well on a related dataset, and thus only needs some finetuning. As basic structures in the data are often learned in the lower layers, the lowest layers are often 'frozen'. This means that the weights in those layers will not be updated at all when training on the target dataset.

1.2.2.2 Catastrophic forgetting

However, there are still several (and often many) weights that are changed during transfer learning. So, the weights for θ_{source} are finetuned to perform also on T_{target}. And as $\theta_{source} \neq \theta_{target}$, the network will probably perform worse on the source dataset after transfer learning. In other words, there is no memory in CNNs except for the learned connections. Function $f_{source}(\cdot)$ is the predictive function as well as the memory at once. During transfer learning the connections are altered as $f_{source}(\cdot)$ transitions to $f_{target}(\cdot)$, thus affecting the previously learned information. This characteristic phenomenon of CNNs (or all connectionist systems) is called *catastrophic forgetting*.

If aiming to learn a specific task with limited data, transfer learning is still often applied. E.g. Inception V3 is often first trained on the ImageNet dataset, where it learns basic image structures and is then finetuned to the specific task at hand. The lost performance on the ImageNet dataset is not important in that case. However, when developing a model that needs to do well on different domains, catastrophic forgetting needs to be dealt with. In clinical practice one system detecting cancer in lymph nodes can be preferred over many models each responsible for one type of cancer. It is often not feasible to hire pathologists specialized in one cancer type only. A hospital may prefer to deal with one system only which can be applied in all tumor types in the lymphatic system. In this way new, unknown or rare types of tumor will not be ignored, whereas with specialized models, chances are that there is no system for it.

1.2.2.3 Elastic Weight Consolidation

As contemporary CNNs often have millions of parameters, there are multiple unique sets of parameter configurations that perform well on a task. All possible sets of parameters – the solution space – that perform well enough are denoted by θ^* and depicted as ellipses in Figure 1.5. To prevent from catastrophic forgetting, a configuration should be found that has good performance on both tasks. So the aim is to find a solution θ_{target} that is 'close' to the previously found solution θ_{source}. Specifically, 'close' means that $\theta_{target} \in \theta^*_{source}$. In Figure 1.5 the goal is to end up where both ellipses overlap and there is low error for both tasks.

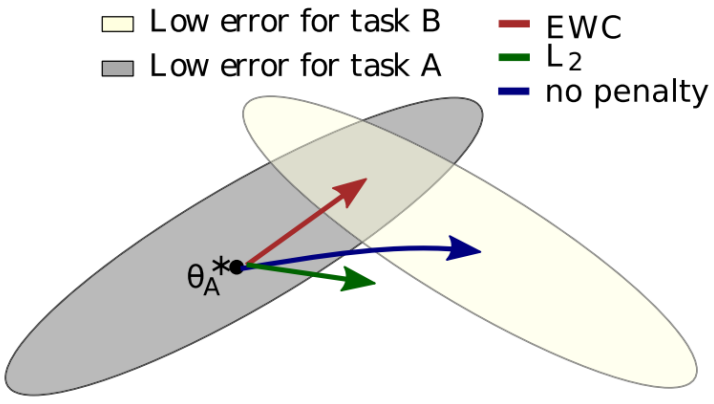


Figure 1.5: Elastic Weight Consolidation by Kirkpatrick et al. [5, p.3522].

The naïve way of transfer learning is depicted by the blue line. The model is adapted such that a low error on T_{target} is achieved without any constraints. It is successful in learning the new task, but by changing its weights, its parameters have changed weights in such a way that performance on T_{source} is lost.

To constrain the parameters to stay close to θ_{source}, an option that is often explored is the use of some quadratic penalty like the L_2 regularization. It penalizes weights that are farther away from the original solution in a quadratic manner. Only solutions that lie relatively close around θ_{source} are accessible. In Figure 1.5 the possible solutions that can be obtained by transfer learning are constrained to a circle with the radius depicted by the green line. However, this green line demonstrates why this is not a great solution. The weights are being adjusted towards the solution space of the target task, but it is not able to reach good performance on T_{target} due to the constraint. Moreover, by changing the weights, it has moved away from the low error space of T_{source} and thus has also forgotten the previous task.

Kirkpatrick et al. [13] observe that using a quadratic penalty assumes that all weights are equally important in obtaining good performance on a task. In terms of Figure 1.5 it can be described as the assumption that the solution space is a circle. However, Kirkpatrick et al. argue that some weights are more important than others in solving a problem. This explains why, in Figure 1.5, the shape of the solution spaces is like an ellipse and not a circle.

Therefore, their aim is to find a penalty that is greater for weights that affect performance most. The constraint when doing transfer learning should not be a circle but an ellipse as well. Ideally, the shape of the constraint is equal to the solution space of the source task.

Another way to understand the idea is to image the circle that describes the possible target solution space with L_2 regularization, but then stretched like some elastic fabric towards parameters that affect performance least. This explains why this method is names Elastic Weight Consolidation (EWC).

It is thus essential to find the importance per parameter. Kirkpatrick et al. use a probabilistic perspective towards the training procedure to compute this importance. Given the available data D the training problem can be defined with Bayes' rule as follows:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \qquad (Bayes'\ Theorem)$$

$$\log p\ (\theta|D) = \log p(D|\theta) + \log p(\theta) - \log p\ (D) \qquad (5)$$

where $\log p\ (D|\theta)$ is the negative loss for the problem. To apply this function to transfer learning, the dataset is split per task:

$$\log p\ (\theta|D) = \log p\big(D_{target}\big|\theta\big) + \log p(\theta|D_{source}) - \log p\ \big(D_{target}\big) \qquad (6)$$

Note that by defining the posterior as T_{source} this formula can be applied for many sequential tasks. Also, all information of the source task is captured in the posterior $\log p(\theta|D_{source})$ obtained by (5). It describes exactly the possible solution space. There is no (catastrophic) forgetting when the problem is defined in this Bayesian manner. Moreover, the posterior "must contain information about which parameters were important to task *A* and is therefore key to implementing EWC." [13, p. 3522]

Unfortunately, the true posterior is computationally intractable as this involves high-dimensional integrals. However, there are ways to approximate the posterior. Kirkpatrick et al. use a Gaussian with a μ given by the parameters θ^*_{source}. To compute the variance, n Fisher matrices should be computed. A Fisher matrix calculates the covariance matrix of θ_{source} for a sample input. One of the advantages of this approximation is that it "can be computed from first-order derivates alone and is thus easy to calculate even for large models" [13, p. 3522]. Once these matrices are computed for $x_i \in \{\mathrm{x}_0 \dots \mathrm{x}_n \in D_{source}\}$ and averaged, the diagonal can be used as the diagonal precision for the Gaussian distribution that approximates the posterior. So, the Fisher matrix is used to approximate the importance of every parameter.

With this Gaussian approximation, the loss L when using EWC is defined as

$$L(\theta) = L_{target}(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{source,i})^2 \tag{7}$$

where i labels every parameter and λ indicates the importance of T_{target} compared to T_{source}. The regularization is thus implemented as a quadratic penalty like L_2, but including now the Fisher matrix that approximates the importance of every parameter – as if the L_2 constraint is pulled like elastic fabric. This allows for less important weights to be altered more than the more important parameters. The possible solution space thus obtained is an ellipse approximating the ellipse that depicts the low error space for T_{source} in Figure 1.5.

1.2.3 Related work

The CAMELYON16 challenge was followed by the CAMELYON17 challenge and although these challenges are closely related, the tasks are slightly different. Where CAMELYON16 focusses on slide level analysis, attention shifted to patient level analysis in the CAMELYON17 challenge and much more data was included [4], [9]. At the start of this project, the results of this challenge were not officially published. Also, much more data would have to be processed. As CAMELYON16 data is already able to achieve very high performance on itself, and the goal of this research is not to improve upon the existing state-of-the-art, this data is not included. However, one could expect to improve upon the findings by using the extra data.

Transfer learning has obtained quite some attention over the years. As commonly done with the Inception architecture, in histopathological image analysis transfer learning is sometimes applied by pretraining on ImageNet [8]. Both tasks are then related by the general upper domain of 'image', but the shared image structures are apparently good enough to achieve state-of-the-art performance [23]. Komura and Ishikawa state that "no networks learned from tasks using other pathological images are available" [24, p. 38] and they still appear to be unavailable as of now.

In recent years several methods have been proposed to prevent from catastrophic forgetting. Experiments regarding catastrophic forgetting do not seem to be used in the domain of histopathology, but are usually conducted in well-known imaging tasks like the dataset [13], [25]. Although on some tasks it is outperformed by other methods like Incremental Moment Matching [26], EWC has generally been effectively used in different imaging areas with continual learning.

2 DATA

When diagnosed with cancer, often a lymph node is excised and placed upon a glass slide for analysis by a pathologist [2]. These glass slides containing tissue can also be digitized via a high-speed whole slide scanner. These scanners are able to scan at a very high resolution (e.g. 240 nm per pixel). One digitized glass slide amounts to an image with "a size on the order of 10 gigapixels." [4, p. 2] The technique to acquire a digitized histopathology slide is called *whole slide imaging* and the obtained results are called *whole slide images* (WSIs).

All glass slides are stained with hematoxylin and eosin (H&E). Hematoxylin colors the cell nuclei and eosin mainly stains the extracellular matrix and cytoplasm in a slightly lighter shade [27]. It is a basic stain and used in daily practice as it is cheap and reliable. As can be seen from the stained slides in Figure 2.1 this staining results in a pink-purple colored slide. The slight variations in color and size are caused by different lab circumstances, scanning equipment, staining protocols or tissue types.

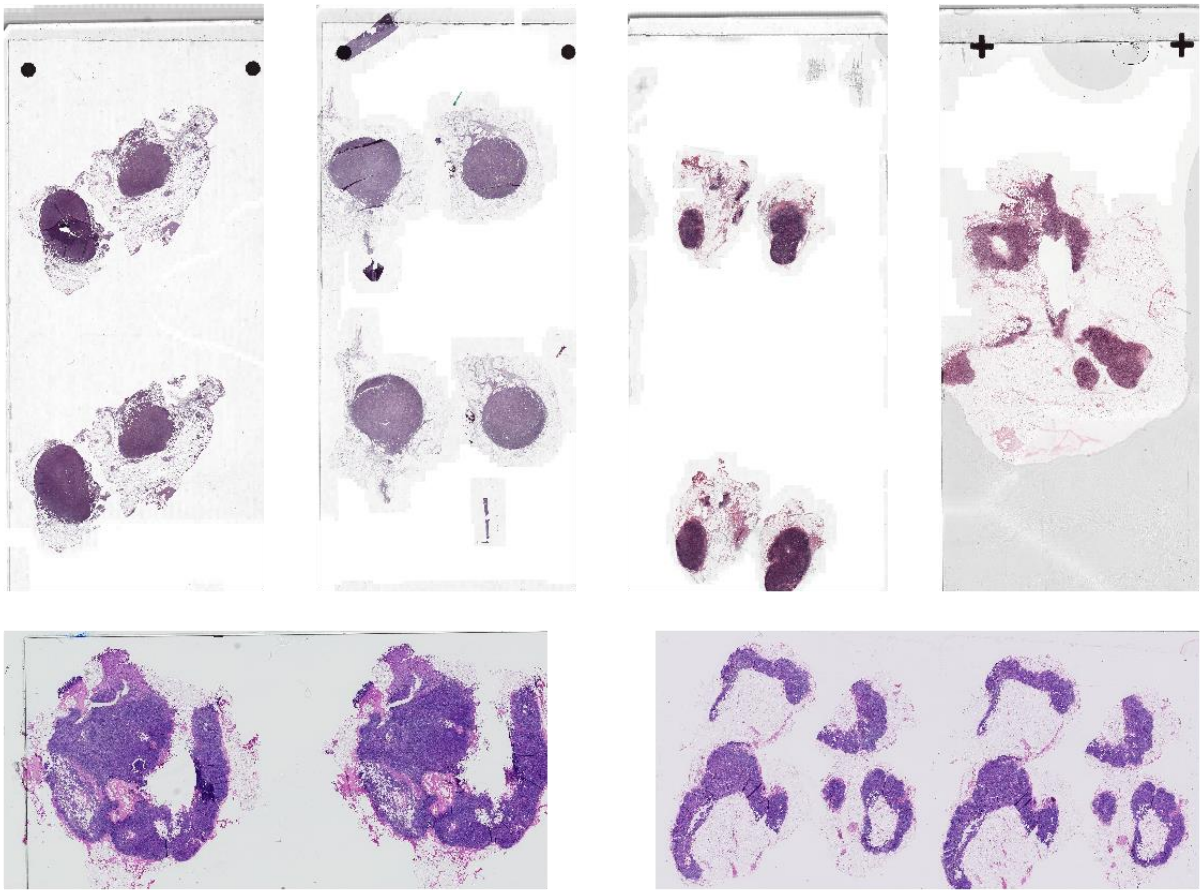


Figure 2.1: Low-resolution examples of different Whole Slide Images.

Three datasets containing lymph nodes of different lymphatic regions are used. After providing details of these datasets, the general preprocessing steps taken to process WSIs are described.

2.1 DATASET 1: BREAST

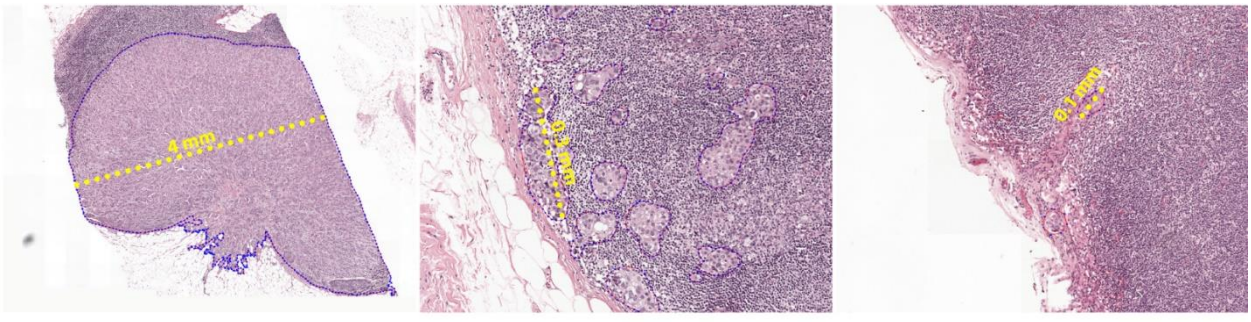
The first dataset contains 399 sentinel lymph nodes (SLNs) in the breast, made available by the Radboud University Medical Center (UMC) and the Utrecht University Medical Center. This dataset was available for the CAMELYON16 and part of the CAMELYON17 challenges [4]. All CAMELYON16 slides are used for this project, including a test set of 129 slides that was published only after the challenge had ended.

There are 159 slides with and 240 without metastasized breast cancer. Metastases is clinically divided in three categories based on the size of the largest available metastasis [4, p. 2]. The details of the metastases division for this dataset are provided by the CAMELYON challenge and are shown in Table 2.1. A visual example of the different categories is shown in Figure 2.2.

Isolated tumor cells (ITCs) are ignored in the CAMELYON16 challenge as "the clinical value of having only isolated tumor cells in an SLN is disputed" [2, p. 2200]. Therefore, there are no slides in the CAMELYON16 dataset that only contain ITCs.

Table 2.1: Division of classes in CAMELYON16 data.

CLASS	SUBCLASS	DESCRIPTION	COUNT PER SUBCLASS	COUNTS PER CLASS
METASTASIS				159
	Macro-metastasis	Larger than 2 mm	82	
	Micro-metastasis	Larger than 0.2 mm and/or containing more than 200 cells, but not larger than 2 mm	77	
	ITCs	Single cells or clusters containing less than 200 cells	0	
NO METASTASIS				240
TOTAL				**399**



(a) Macro-metastasis (b) Micro-metastasis (c) Isolated tumor cells

Figure 2.2: Different sizes of breast cancer metastases by Litjens et al [6, p.4].

After the CAMELYON16 challenge, a few changes have been made to the dataset. Originally, one slide was duplicated and has therefore been removed. Another slide was wrongly indicated to contain no metastasis and an annotation was added to describe the tumor region. Furthermore, 15 slides are not exhaustively

annotated. Exhaustive annotations are still not available for all slides. In this project the improved dataset – but still including non-exhaustively annotated slides – is used. This is the same dataset used by Liu et al. in their state-of-the-art solution [5].

2.2 Dataset 2: Colon

The other two datasets used in this project are not publicly available. They are however accessible within the Radboud UMC. The second dataset is provided for by the Rijnstate hospital and contains lymph nodes extracted around the colon. The dataset consists of 138 contain metastasized colon cancer and 102 slides not containing any metastasis, making a total of 240 WSIs.

A difficulty with this dataset is the variation in annotations. Some annotations were very accurate and detailed, while others roughly marked the area containing metastases, see Figure 2.3. In the other datasets used, the annotations are always very detailed.

Table 2.2: Division of classes for colon data.

CLASS	COUNT PER CLASS
METASTASIS	138
NO METASTASIS	102
TOTAL	**240**

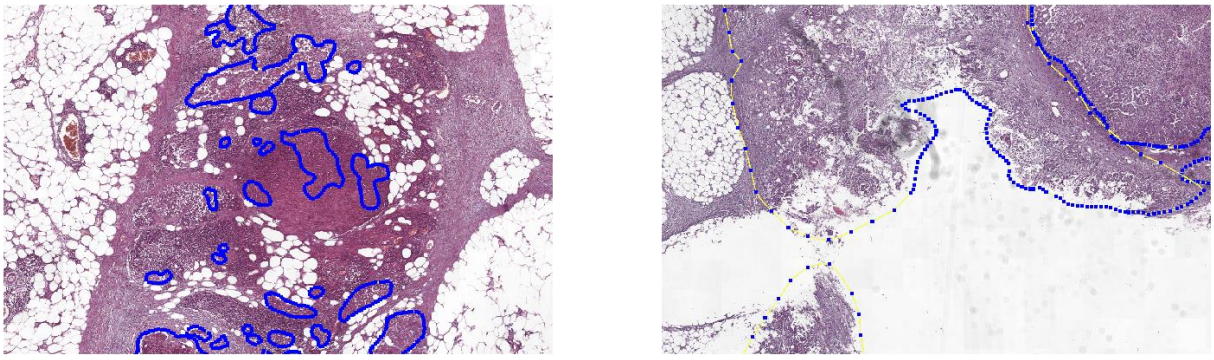


Figure 2.3: Detailed annotations in the left slide, while on the right annotations coincide and include background.

2.3 Dataset 3: Head-Neck

A small dataset containing lymph nodes extracted from the junction of the head and neck, is also made available by the Amsterdam University Medical Center (location: VUmc). This dataset has been annotated by medicine students. Some of these annotations have been checked by a pathology resident at the Radboud UMC. The details can be found in Table 2.3.

Table 2.3: Division of classes for head and neck data.

CLASS	SUBCLASS	COUNT PER SUBCLASS	COUNT PER CLASS
METASTASIS			23
	Annotations checked by resident.	5	
NO METASTASIS			8
TOTAL			**47**

2.4 PREPROCESSING

2.4.1 Converting raw data

All scanned slides were initially available as MRXS files – the proprietary format of the used 3DHistech scanner. Different scanners store WSIs in different ways. In order to prevent from altering algorithms to handle different file formats, the WSIs are converted to the open source Tagged Image File Format (TIFF) using the ASAP package [28]. A TIFF image file is stored as a series of small JPEG compressed image tiles (512x512 pixels) on different magnification levels. At the highest magnification used in these datasets at 20X which corresponds with ~0.24 micrometer pixel spacing.

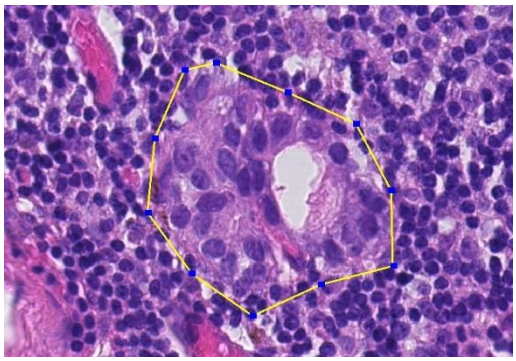


Figure 2.4: Annotations are stored as polygons.

The annotations are stored as polygons. These polygons consist of a list of coordinates describing the tumor border and are stored as a list in eXtensible Markup Language (XML) [4]. The ASAP package provides tools to parse and display the annotations and to create or alter existing annotations as well [28].

2.4.2 Generating masks

Only part of the slide contains the lymph node tissue. As the gigapixel images are already computationally heavy to process, a tissue mask of all WSIs is computed. The removal is done by applying a not yet published, in-house developed tissue segmentation model. It has the same architecture as the fully convolutional neural network in [29], but it has been trained with much more data. Moreover, it can be used on a range of levels as the new model is resolution agnostic.

Annotations are delivered in XML-format which are converted to a mask as well. For WSIs including metastases, the two masks are combined by addition. This results in the following pixel mask values:

0. Background
1. Normal tissue
2. Tumor tissue

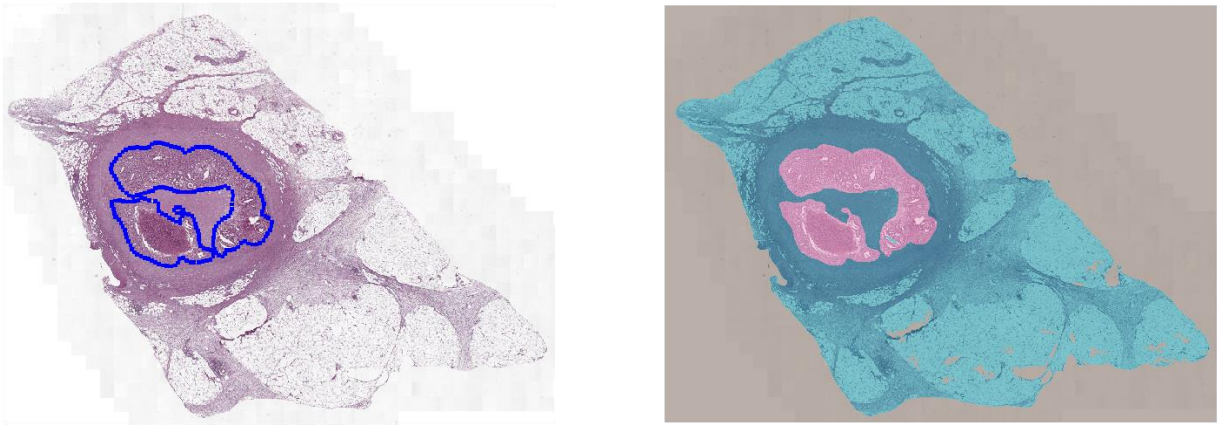


Figure 2.5: Example of a generated mask. The mask in the right slide displays the three pixel values with a different color.

2.5 DATA USAGE

2.5.1 Training

The input size for the Inception and NatureNet architectures should be 299x299 and 128x128 pixels respectively. Although the final LYNA paper processed the images at the maximum level of magnification (i.e. 0.25 micrometer pixel spacing) [6], the networks were trained using input data with ~0.5 micrometer pixel spacing. This ensures that the network is provided enough context, especially the NatureNet as the input size is only 128x128 pixels. This extra context may help the network to distinguish better between different tissue types as well. Liu et al. already outperformed the CAMELYON16 winners' solution by using the ~0.5 micrometer spacing value in the first LYNA paper and using a higher spacing value speeds up the training procedure and requires much less memory [5].

It is common practice to keep the distribution during training equal per class. However, the datasets suffer from major class imbalances [4], [5]. The majority of the WSIs do not contain metastases and even the slides that do contain metastasis only a part of the whole slide contains metastasized cancer. This is not per se a problem of the dataset, but these characteristics reflect clinical practice. Therefore, the original tumor to non-tumor distribution in the dataset should not be ignored altogether by sampling an equal number of patches per class.

Therefore, the following patch sampling strategy was adopted. Every epoch an equal number of WSIs per class is opened. These slides are chosen at random with a 1:1 ratio of the slide containing metastases or not. Patches are then sampled from these slides randomly with a 3:1 tumor to non-tumor ratio (background patches are ignored) to account for the majority of tissue being non-cancerous. The patch label is the value of the central pixel (or the upper left neighbor in case of the 128x128 patches) in the patch. These patches are all normalized on all RGB-channels from [0, 255] to [0, 1].

2.5.2 Inference

There are two ways of doing inference. First, inference can be done in a fully-convolutional manner [30]. This results in a pixelwise prediction heatmap based on the input patch as shown at the left in in Figure 2.6 and referred to as a heatmap. However, deep architectures containing many convolutional layers must usually pad the input image with zeros at several stages [10]. The classification layer gets used to these padded and then cropped images during training. However, a test patch does not go through a process that adds padding at several stages upon inference. This means that there will be inference points that have an input field of view that is not similarly subjected to zero-padding as it was during training, leading to false results and cause the network to lose performance significantly.

To prevent this from happing one could use a different approach. An alternative way of doing inference divides a test image into Regions Of Interest (ROI) of 128x128 pixels, based on the patch extraction process of [5]. To give the network some context and to achieve the same input size as during training, a patch of 299x299 is cropped centered on the ROI. Each of these patches is classified and the resulting confidence value counts for the ROI it was centered on. This means that the confidence score for one ROI is the same for all the 128x128 pixels. The original image is padded with zeros around the edges to not go out of bounds when extracting the patches. The heatmap thus obtained consists of tiny squares that cover 128x128 pixels on the level with 0.5 micrometer pixel spacing. All pixels in this square share the confidence value, as displayed on the right in Figure 2.6.

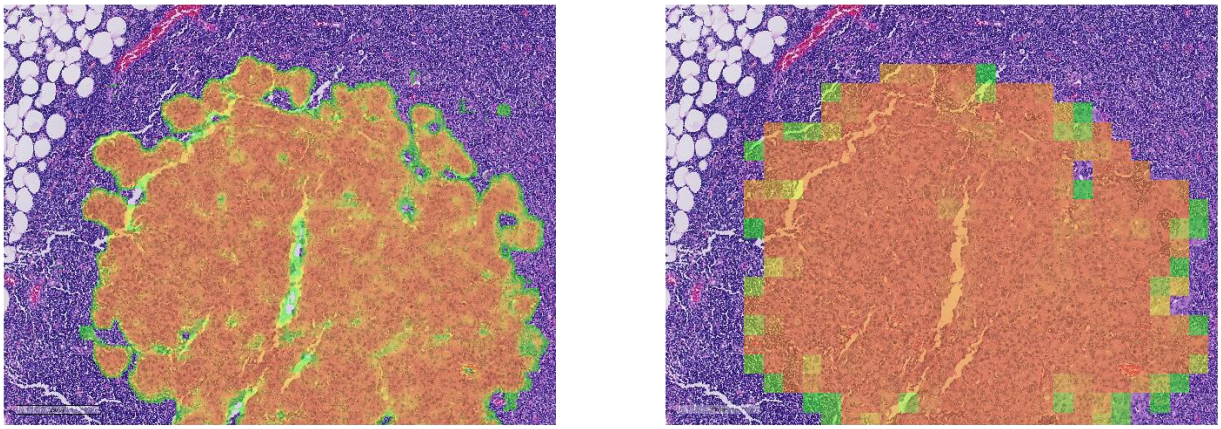


Figure 2.6: The difference in the obtained heatmaps when using a fully convolutional (left) or a ROI-based approach (right).

2.6 DATA AUGMENTATION

Often, augmentation is done to compensate for a shortage of training data. As the CAMELYON16 dataset contains many slides and each slide contains around 10,000 to 400,000 (median 90,000) 299x299 patches [5], the need for augmentation is not obvious. Still, every patch used during training is augmented twice due to differences in WSIs that occur during clinical practice.

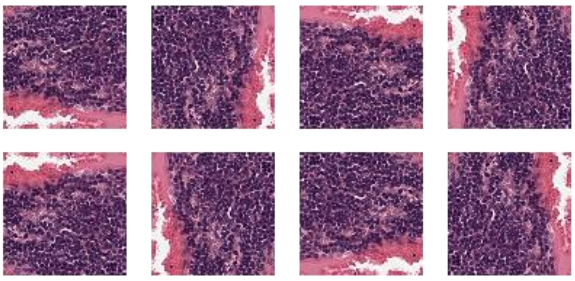


Figure 2.7: Rotation augmentations.

There is no natural orientation of the tissue as it can be put on the slide in many positions. Glass slides can be used horizontally or vertically depending on tissue or hospital practice. The network should be indifferent to position of tissue. Figure 2.7 displays the spatial augmentation used which are achieved by rotating a patch 0, 90, 180 or 270 degrees and flipping the original patch vertically followed by rotating again to any of the four multiples of 90 degrees. This gives eight possible augmentations. One of these eight spatial augmentations is included per originally sampled patch.

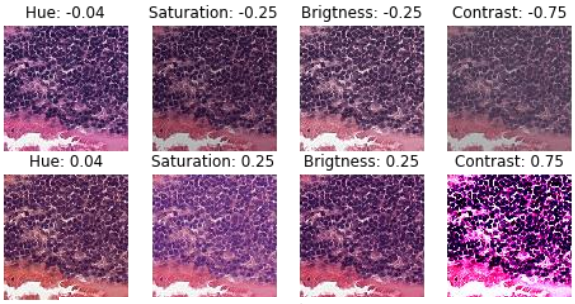


Figure 2.8: Color augmentations.

Second, staining can be different due to a number of factors as depicted in Figure 2.1. To compensate for these differently colored WSIs the color of the training patches is augmented. Figure 2.8 depicts the maximum differences in color augmentations on the HSB color space. The hue-channel is perturbed between -0.4 to 0.4, saturation and brightness between -0.25 and 0.25. These augmentations are followed by perturbing the contrast between -0.75 and 0.75. These ranges follow the approach by Liu et al. [5] who optimized these ranges during training.

2.6.1 Test time augmentation

During test time augmentation, every ROI of the image is rotated in all eight possible directions, see Figure 2.7. So, the network sees all patches from 8 perspectives. All these patches are then classified by the network and the result will be the geometric mean of the eight obtained heatmaps.

3 EXPERIMENTS

In the CAMELYON16 challenge, the task is 'only' to distinguish metastasized tumor from tissue not containing cancerous cells. However, these classes are not easily divided as both "can be highly heterogeneous" [31]. For example, fat tissue also belongs to the 'normal' class and there is no differentiation between different types of tumor. As with CAMELYON16, the goal is to distinguish metastases from other tissue. When speaking of 'normal' regions or slides, this means all types of tissue that simply do not contain any metastases.

3.1 ARCHITECTURE

To answer the question if the architecture can be made smaller and simpler while maintaining performance when auxiliary strategies remain the same, two networks are trained. For reference, the slim version of Inception (V3) is used [10]. The Inception architecture has been used to approach the current state of the art results on the CAMELYON16 dataset [5], [6]. The smaller and simpler architecture consists of only five convolutional layers and a final *softmax* activation function. This architecture is referred to as 'NatureNet' and is compared to the slim Inception architecture.

3.1.1 Learning procedure

The CAMELYON16 challenge demonstrated that current state of the art performance is comparable with expert pathologists [2]. After this challenge the LYmph Node Assistant (LYNA) was developed – first described in [5] but later improved leading to [6]. Both algorithms outperformed the winning solution of the CAMELYON16 challenge [6]. To achieve comparable performance with the current state of the art, developing the source model closely follows the approach taken to develop LYNA and uses the CAMELYON16 data.

3.1.1.1 Training

The original train-test distribution of the CAMELYON16 challenge is kept intact. This means that 129 (49 tumor – 80 no tumor) of the 399 WSIs are used only for evaluation. The other 270 slides are randomly split in a training (80%) and validation (20%) set preserving tumor non-tumor ratio.

To develop the source model the total number of images open in memory and available for sampling per epoch is limited to 100 for computational reasons. Per training epoch the network uses 25600 batches of 32 patches and during validation 3200 batches of 32 patches. The Adam optimizer [16] is used with an initial learning rate of 0.05. This learning rate was multiplied by 0.5 every four epochs[1] that the network did not improve on patch level validation accuracy. The training stopped after 12 consecutive epochs that did not improve the network. This means that during these 12 epochs the learning rate is decreased twice.

3.1.1.2 Inference

One of the advantages of NatureNet is that it can be applied to the test set in a straightforward fully convolutional manner [30]. The Inception architecture can also be applied in a fully convolutional way. However, as this is a much deeper network, there is a lot more zero-padding [10]. When using the

[1] See for the exact definition of an 'epoch' section 2.5.

Inception architecture, roughly 20% of the image is affected by this. Therefore, when using the Inception architecture, the ROI-based approach is used to do inference.

3.1.1.3 Evaluation

From the obtained heatmaps performance is measured on three different levels. Slide-level and lesion-level metrics are statistical analyzed by a bootstrapping approach [32]. Randomly, n samples from the test set are chosen which over which the metric is computed, where n is the length of the test set. The 2.5 and 97.5 percentile are reported. In correspondence with Liu et al. this procedure is repeated 2000 times [5].

3.1.1.3.1 Slide-level: ROC-AUC

First, the network needs to be able to distinguish slides containing metastases from slides containing normal tissue only. An intuitive approach here is used as the highest confidence score in the whole slide is simply used as the confidence score for the whole slide. To evaluate the scores, the area under the curve of the receiver operating characteristic curve (ROC-AUC) is computed.

3.1.1.3.2 Lesion-level: FROC

Second, the network needs to identify lesions (chunks of tumor cells). Lesions were identified from the heatmap by adopting the non-maxima suppression method by Liu et al [5]. The non-maxima suppression method repeats two steps:

1. Report the highest value as a lesion.
2. A circle with some radius r_{lesion} zeros out the region in the heatmap centered on the reported lesion.

These two steps are continued until the highest value found is below a certain minimal threshold t_{lesion}.

From the reported coordinates and the confidence value at those coordinates a free-response receiver operator characteristic curve (FROC) can be computed [33]. This curve should be interpreted as plotting the true-positive fraction against the mean number of false-positive detection in slides free from metastasis [2]. When computing the FROC score, ITCs are ignored in correspondence with the CAMELYON16 challenge. Computing the FROC score requires setting two hyperparameters. The radius $r_{lesion}$is set to the ROI size of 128 pixels. In correspondence with Liu et al. a conservative threshold of 0.5 is chosen for t_{lesion}.

3.1.1.3.3 Pixel-level: adapted Dice

These slide- and lesion-level evaluation metrics were also used in CAMELYON16. To obtain high effectiveness for these metrics it is not very important to identify metastases on pixel-level. Identifying the center of lesions will also suffice to get high scores. However, to adequately measure performance on new datasets, pixel level performance is informative. Pixel level effectiveness in image segmentation is usually measured with the Dice score – which is equivalent to the F1 score. The problem at hand is not a segmentation challenge but identifying metastases on pixel-level comes close. Contrary to a segmentation challenge, there are many slides that do not contain metastases. As false positives can occur, the Dice metric is adapted as follows:

$$dice'(x)(: \begin{cases} dice(x) & \text{if } x \text{ contains tumor regions} \\ 1 - dice(x) & \text{otherwise} \end{cases} \quad (8)$$

For slides not containing tumor regions $dice'$ inverts the Dice score for mistaken predictions. A Dice score can only be computed from a binary image. To obtain this binary image from the heatmap requires a certain threshold t_{pixel} after which every confidence value below t_{pixel} will be 0 and every pixel value above t_{pixel} becomes 1.

To find the optimal t_{pixel} value, a subset of the CAMELYON16 test data is used. Specifically, this subset consists of 15 slides containing no tumor, 15 slides with micro-metastases and another 15 containing macro-metastases – all chosen randomly from the total test set. This subset is also used to obtain the optimal value for hyperparameter for Dice computation.

3.2 TRANSFER LEARNING

Experiments for subquestions 2 – 4 will be carried out simultaneously as one would expect the training set and EWC to have an effect on the learning rate and vice versa. Therefore, experiments to answer these questions are combined based on these three parameters.

3.2.1 Developing bounds

To see how effective transfer learning in general is, it makes sense to compute a lower and an upper bound of effectiveness on the target task. The lower bound can be found by applying the source model on the test images from the colon and head and neck dataset – i.e. without any extra training. To obtain an upper bound, a network can be trained from scratch with all available training data of the target type of data in the same way as described when developing the source model. However, this is only an option for the colon dataset as the head and neck dataset is so small that it is probably insufficient to train a sophisticated network architecture like Inception and all data is already used during transfer learning.

3.2.2 Regular transfer learning procedure

The described, straightforward approach of doing transfer learning is applied to all available training sets. Cells and tissue shape, size and color may look a bit different compared to the source dataset. Because even the very basic information can differ, all layers will be trainable. However, the learning rate is kept lower than 0.001 and multiplied by 0.5 every three consecutive non-improving epochs and after nine epochs in which the network did not improve, training is stopped.

A maximum of 50 images are open for sampling during an epoch. Per training epoch the network uses 3200 batches of 32 patches and during validation 1600 batches of 32 patches. The tumor to non-tumor ratio, patch extraction process, normalization and augmentation are not altered compared to the development of the source model.

3.2.3 Parameter 1: Training set size

A well-trained source model that achieves excellent effectiveness is a prerequisite for all types of transfer learning. The architecture with highest effectiveness is therefore used as the source model to do all transfer learning experiments. The setup (e.g. way of doing inference, evaluation metrics) is largely the same as when developing the source model. Anything done differently will be described explicitly.

To find out to what extent training set size matters, several experiments on the colon dataset are done. For all experiments on the colon dataset, the test set consists of 50 WSIs randomly sampled from the dataset with a 1:1 slide label ratio, i.e. 25 images with and 25 images without metastasis. Several training sets are created to experiment with the amount of data required to do transfer learning. In order to

compare the differences in dataset, the first training set consists of 12 WSIs. As with the head-neck dataset, the training-validation split is 6-6. Training sets are created with 10 and 20 training images. Those two datasets have the same additional validation set of 10 WSIs. So, these training sets consists of respectively 20 and 30 WSIs. The smaller training sets are subsets of the biggest training set. The different datasets are summarized in Table 3.1. The head-neck dataset cannot be used to experiment with different training set sizes as the highest possible training to validation ratio is 6:6. Experiments are run independently of each other, so training on 20 images does not mean including 10 images after training on 10 images first.

Table 3.1: Dataset splits for the different experiments.

DATASET	TRAINING SET SIZE	VALIDATION SET SIZE	TEST SET SIZE
COLON			
	6	6	50
	10	10	50
	20	10	50
HEAD-NECK			
	6	6	12

3.2.4 Parameter 2: Learning rate

The source network has already learned most of the information available. Therefore the learning rate during transfer learning is usually decreased by a big factor. When a very low learning rate is already able to perform well on the target dataset, chances are that only a little bit of forgetting will occur. On the other hand, there is a risk that the training procedure will get stuck in a local minimum as the learning rate is not big enough to get out of it.

As the third subquestion asks for the effect of the learning rate, two different learning rates are chosen that are typical learning rate values. First, the experiments are done with a very small learning rate of only 10^{-5} , which is based on the assumption that almost all information has already been learned. Second, a fairly aggressive learning rate of 10^{-3} is used. As the difference in order of magnitude of these learning rates is large, the difference in performance will highlight what effect the learning rate has on transfer learning.

This experiment can be done on the colon dataset as well as on the head-neck dataset. Only the annotations that were checked by the pathology resident were used as training data. Slides that were not checked were used only as potential test data. There are 10 test slides to get a test set that is on the one hand representative enough of the data and, on the other hand leaves training data to do transfer learning. Only a very small training set can be created: 6 WSIs containing metastasis and 6 WSIs not containing metastasis.

3.2.5 Parameter 3: Elastic Weight Consolidation

The extent to which EWC prevents catastrophic forgetting, while minimizing loss of effectiveness on the source task, is being investigated for all regular transfer learning experiments. First, effectiveness of the networks obtained after doing regular transfer learning on the source data is measured. Then, another network will be trained with the same setup, except that EWC is included during training.

EWC can be implemented by first computing the Fisher matrix of the source model. This Fisher matrix and the trainable weights of the source model can then be used to compute the constraint per layer. In Keras,

this constraint can be implemented by setting the *bias_regularizer* and *kernel_regularizer* parameter for convolutional layers and the *beta_regularizer* for batch normalization layers to the constraint. The *bias_regularizer* should only be set if indeed a bias is used in this layer.

Also EWC cannot be measured separately from other settings. Therefore, experiments with different training set sizes and learning rates are all done with and without EWC. So, every experiment is defined by the three different parameters: training set size (6, 10, 20), EWC (yes or no) and learning rate (10^{-3} or 10^{-5}). All combinations of these parameters define the experiments for the subquestions 2 – 4.

4 RESULTS

4.1 ARCHITECTURE

There is a notable difference in effectiveness between the two different architectures as displayed in Table 4.1. Test time augmentation on the Inception architecture behaves according to expectation by improving effectiveness on the CAMELYON16 metrics.

Table 4.1: Architecture results.

ARCHITECTURE	TEST TIME AUGMENTATION	FROC	ROC-AUC	DICE
NATURENET				
	×	0.560 (0.556 – 0.562)	0.852 (0.850 – 0.854)	0.792
INCEPTION				
	×	0.737 (0.739 – 0.743)	0.961 (0.960 – 0.962)	**0.863**
	✓	**0.750 (0.753 – 0.757)**	**0.972 (0.971 – 0.972)**	0.857

4.1.1.1 Hyperparameters

As the ROC and FROC indicate that the Inception architecture performs better than the NatureNet, the tuning of t_{pixel} (i.e. to binarize the heatmap) is done with heatmaps generated by the Inception network. The Dice score is computed for different threshold in a range from 0.75 to 0.95. Combined, the scores are best at a threshold of 0.85. However, there is a peak for micro-metastases at a threshold of 0.9. Because The difference with 0.9 is marginal (0.85: *0.800*, 0.9: *0.799*) and to prevent the network from missing small tumor regions, the threshold is set to 0.9.

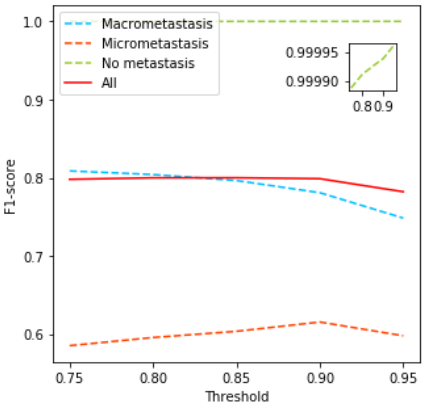


Figure 4.1: Hyperparameter optimization.

4.2 TRANSFER LEARNING

As experiments on training set size, learning rate and EWC are not tested independently of each other, they are reported at once.

The results of applying the source model (Inception architecture) on both target datasets are reported in Table 4.2. Furthermore, the upper bound for the colon dataset is reported after the exact same training procedure and architecture that is used to get the highest results on the CAMELYON dataset.

Table 4.2: Lower and upper bounds for target datasets.

	DATASET	FROC	ROC-AUC	DICE
LOWER BOUND				
	Head-Neck	0.341	0.950	0.205
	Colon	0.229 (0.249 – 0.255)	0.896 (0.894 – 0.899)	0.664
UPPER BOUND				
	Colon	0.528 (0.532 – 0.539)	0.970 (0.967 – 0.970)	0.845

4.2.1 Colon

Results of transfer learning on the colon dataset are visually summarized in Figure 4.2. Exacts results for transfer learning on the colon dataset can be found in Supplement A.

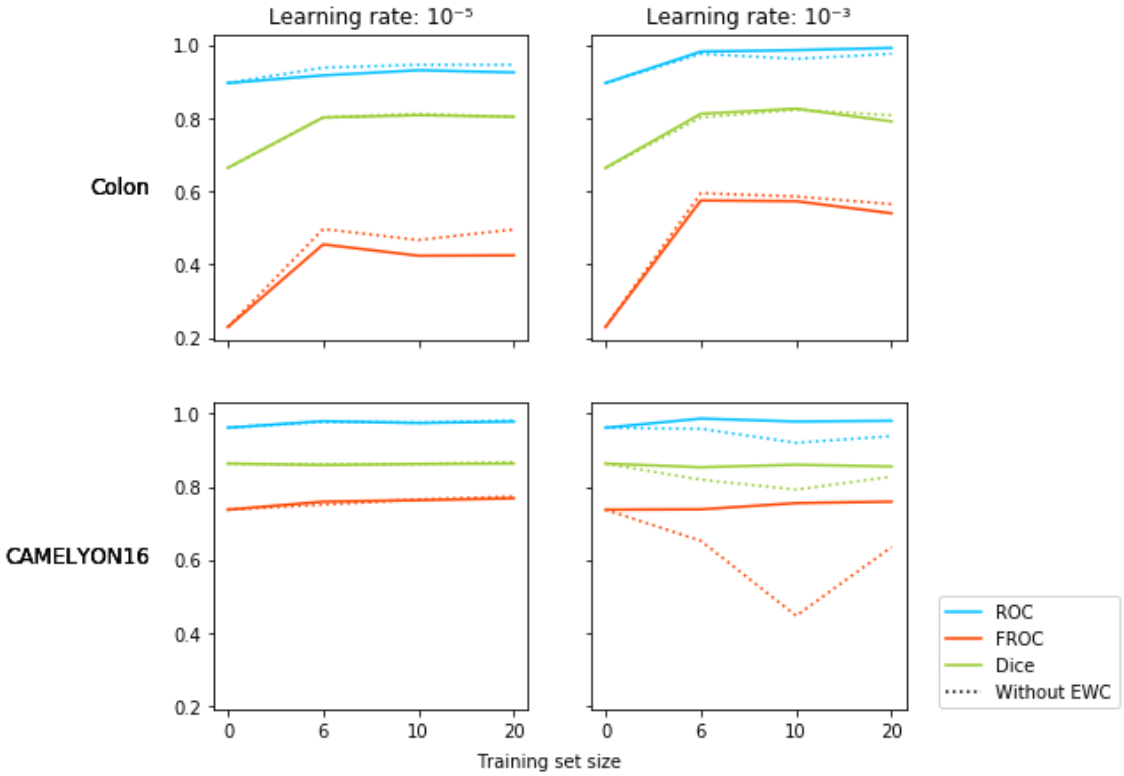


Figure 4.2: Experimental results on the colon dataset.

The rows display effectiveness on the colon and CAMELYON16 dataset respectively. The difference in learning rate is displayed by comparing the plots on the left with those on the right. On the x-axis the size of the training set is shown. Training set 0 means directly applying the network trained on CAMELYON16. So, these correspond to the results reported for CAMELYON16 in section 4.1 without test time augmentation.

As most of the lines are relatively horizontal after adding target data, in general, colon results do not benefit from adding much data. A small dataset suffices to achieve maximum performance. Rather, with the higher learning rate effectiveness on two of the three metrics decreases after adding data. Adding data, however, does slightly improve the effectiveness scores on the source dataset.

The effect of increasing the learning rate on the other hand is big, especially for the FROC metric as it increases substantially. However, the countereffect of this – catastrophic forgetting – occurs as can be seen from the dotted line, as results on the CAMELYON16 dataset are significantly lower.

Elastic weight consolidation in general decreases results on the target dataset. However, the changes are not big, and with the high learning rate the ROC metric is even improved. With the high learning rate, the effects of catastrophic forgetting are mitigated by using EWC.

4.2.2 Head-neck

Results for the head-neck dataset are not along the same lines as the colon dataset. The network performs best on the target dataset when using EWC and the aggressive learning rate. Scores on the CAMELYON16 challenge however drop very quickly, even with the lower learning rate. Performance on the source task is best when disabling EWC and using the higher learning rate.

Table 4.3: Results on the head-neck dataset.

Learning rate	EWC	Test set	FROC	ROC	DICE
0.001	×	Head-neck	0.460	0.900	0.551
		CAMELYON	**0.554** (0.555 – 0.561)	**0.919** (0.917 – 0.919)	**0.808**
0.001	✓	Head-neck	**0.512**	**0.950**	**0.581**
		CAMELYON	0.427 (0.427 – 0.434)	0.901 (0.899 – 0.902)	0.794
0.00001	×	Head-neck	0.488	0.500	0.488
		CAMELYON	0.446 (0.462 – 0.468)	0.906 (0.903 – 0.906)	0.790
0.00001	✓	Head-neck	0.492	0.450	0.497
		CAMELYON	0.448 (0.448 – 0.453)	0.901 (0.900 – 0.903)	0.788

5 DISCUSSION

When interpreting the results the focus will be on the FROC-score as identifying tumorous regions is eventually the most important task and there are "approximately twice as many tumors as slides." [5, p. 5] Based on the results of the experiments, answers to the research questions are formulated in this chapter. Further, limitations of this project and future work are discussed per question and in general.

5.1 ARCHITECTURE

Although the architecture should not make a big difference, the results show that using a minimal network like NatureNet, does not suffice to adequately fit the data. NatureNet can be applied fully convolutional, leading to pixel level predictions. When looking at Figure 2.6, one would expect the fully convolutional approach to have a higher Dice score. It is much more detailed around the edges and detects very small regions. However, NatureNet performs worse than Inception on the Dice metric. So, this drawback of using a bigger network like Inception, i.e. not being able to apply the network fully convolutional, is apparently compensated for by the sophisticated architecture. So, although Bejnordi et al. stated that the architecture is less important than auxiliary strategies [2], this does not mean a straightforward, simple convolutional neural network will do the job. For future work, one could find out to what extent the results of the Inception architecture can be approached by a slightly more complex architecture than NatureNet but that can still be applied in a fully convolutional manner.

Table 5.1: Comparing architectures.

	FROC	ROC
Liu et al. ***(SOTA – slim version)***	0.855	0.986
Wang et al. ***(Winner)***	0.807	0.994
Proposed	0.750	0.972

In Table 5.1 the obtained results are compared to the winners' solution and to the current state-of-the-art solution by Liu et al. The approach by Wang et al. does not use a single system but uses an additional network for lesion detection. The obtained scores by a single system are relatively close to these CAMELYON16 winners who report a FROC of 0.807 and ROC of 0.994 [2], [34]. The obtained scores for the Inception architecture are slightly lower than the scores reported by Liu et al [5]. They report a FROC of 0.855 (0.810 – 0.897) and ROC-AUC of 0.986 (0.967 – 1.00).

Although the approach followed here closely resembles that of Liu et al., there are several differences that may explain the gap in performance. First, they used a somewhat different patch sampling approach where all available patches are seen during training. However, it is unlikely that this causes the difference. In their approach, every slide is divided into ROIs of 128x128 pixels which are then viewed by the network during training. This guarantees that all training data is seen during one epoch. When sampling randomly, this cannot be guaranteed. However, when sampling large numbers, as is the case, it approaches seeing the full dataset. Moreover, random sampling has the benefit that a ROI with offset (50, 26) is possible where otherwise only multiples of 128 are valid offsets.

Second, they train with validation metrics of the challenge, which may tune the network a bit more towards the metrics at hand. However, our approach does not need to perform high on specific metrics and is therefore not trained with these metrics. Our network is trained with patch-level accuracy as this is computationally less heavy. The accuracy reported during training is reflected in the metrics upon evaluation.

Third, and this is probably responsible for the biggest gap, they use an ensemble of multiple networks to obtain their final score. Their technical description of the solution speaks of no less than 10 networks, but "gave diminishing returns after 3 models" [5, p. 7]. A limitation of the study at hand is that it has not been analyzed experimentally which difference is mainly responsible for the gap in performance, as this (especially the last reason) is computationally heavy.

5.2 TRAINING SET SIZE

The second subquestion was to what extent training size matters when the straightforward approach to transfer learning is applied between different histopathological systems. First, it must be noted that, although not done before, transfer learning is possible between different histopathological systems. Scores on the target datasets obtained via transfer learning are even higher than training on the target dataset from scratch. This not only means that a network trained on the CAMELYON16 dataset will likely be performing well when detecting other cancer metastases in lymph nodes, but that it benefits from being initialized with these weights. The network is thus better in discriminating normal from tumor tissue when information of the CAMELYON16 challenge is available.

In Figure 4.2 the training set size is displayed on the x-axis. As the performance lines are more or less horizontal amongst these different sizes, it can be concluded in general that the training set size does not seem to matter a great deal. However, the size does change the scores in some interesting ways. It seems that training with less data is beneficial. Effectiveness on two of the three metrics on the target dataset decreases when the amount of data is increased.

5.2.1 Dataset quality

There are two remarkable trends visible in the scores. First, effectiveness on the colon set is always significantly lower than on the CAMELYON16 dataset as FROC score never exceeds 0.5. Second, when adding data effectiveness on the target task becomes slightly lower.

The CAMELYON16 dataset is a relatively 'clean' dataset. Annotations are accurate and mostly complete, and the scanned quality of the slides is high. The colon set is of inferior quality as there is colon tissue and annotations are not as accurate. This begs the question of whether the quality of the colon dataset is responsible for this loss in performance. Both trends in the scores can then be explained as effectiveness is indeed worse when the data is bad and that adding data actually means corrupting the data, which explains why effectiveness drops when data is added.

5.2.1.1 Cleaning the dataset

This suspicion is enhanced by the fact that the colon dataset originally contained 58 more WSIs. When doing the first experiments results were very low (under 0.2). Consequently, many experiments were conducted excluding the possibility of there being a problem with the model or training procedure. Experiments were done by freezing different Inception-modules, by training with different learning rates, and using a different amount of training data. Lastly, the optimizer was changed from Adam to SGD as this latter optimizer does not use an adaptive function. With a very sophisticated trained model like the source model, Adam can cause some basic weights to be only minimally updated as these weights did not need updating before. On the target data these weights sometimes need finetuning as well. With SGD all parameters are updated by the same factor – also the 'seemingly' less important weights.

However, all these changes did not improve the results. The highest possible FROC score of 0.184 is then obtained when training on 40 slides with a learning rate of 0.001 and all layers trainable. Training from scratch only slightly improved this result.

To find out whether the original dataset was responsible for these low results, the data was inspected by a PhD student doing image analysis research regarding colon tissue at the Diagnostic Image Analysis Group at the Radboud UMC. Especially, the worst performing WSIs – two slides had over 800 false lesion detections – were inspected. The student found that the network was correct in detecting these lesion detections, but that the tissue was not annotated as it was a tumor region in colon tissue and not in the lymph node – this slide contained both a lymph node and colon tissue. The student also found that there were several slides containing tumor that were not annotated as such, quite some slides having large out-of-focus regions due to some scanning problem and that in some WSI no lymph nodes were present (or only marginally) while there was colon tissue – with or without tumor cells – present on the slide.

5.2.1.2 Lower performance in general

The dataset was filtered by removing 58 images. This resulted in the colon dataset as described in Section 2.2. Results greatly improved, but results are still significantly lower than for the CAMELYON16 dataset, even after training from scratch. Although the PhD student is not a pathologist, and the dataset is probably still not equal to the CAMELYON16 dataset in terms of accuracy and scanning quality, it is unlikely that the poorer quality of the dataset is fully responsible for the loss in performance.

Otherwise, one would expect results from EWC to be higher as more information is incorporated from the source dataset than training without EWC, but on the colon dataset EWC slightly decreases performance on the target task. Moreover, upon inspection of the worst performing test slides, other factors appear to cause the confusion of the model. Three slides containing many false positive lesion detections are shown in Figure 5.1 and Figure 5.2. The false positives in first image are caused by the tissue being inflammatory, while the other two contain necrosis. So, in a sense, the lymph node looks abnormal,

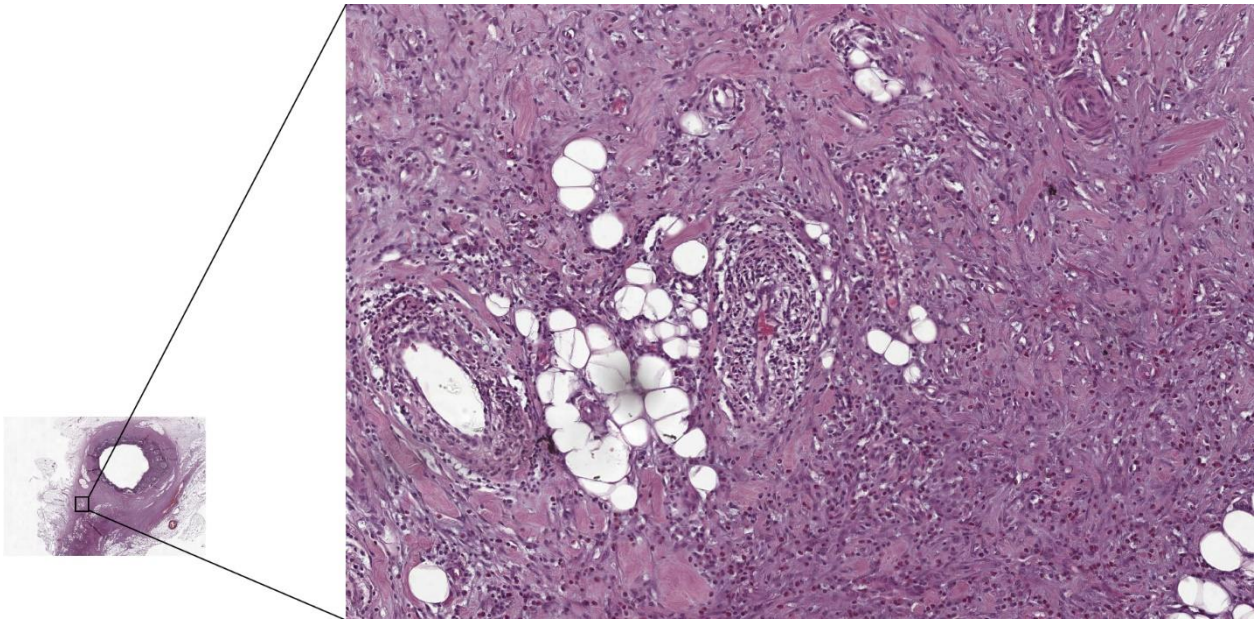


Figure 5.1: Inflammatory tissue.

although the anomaly is not cancerous. Lastly, experiments on the colon dataset used a relatively high number of epochs during training, even with the high learning rate.

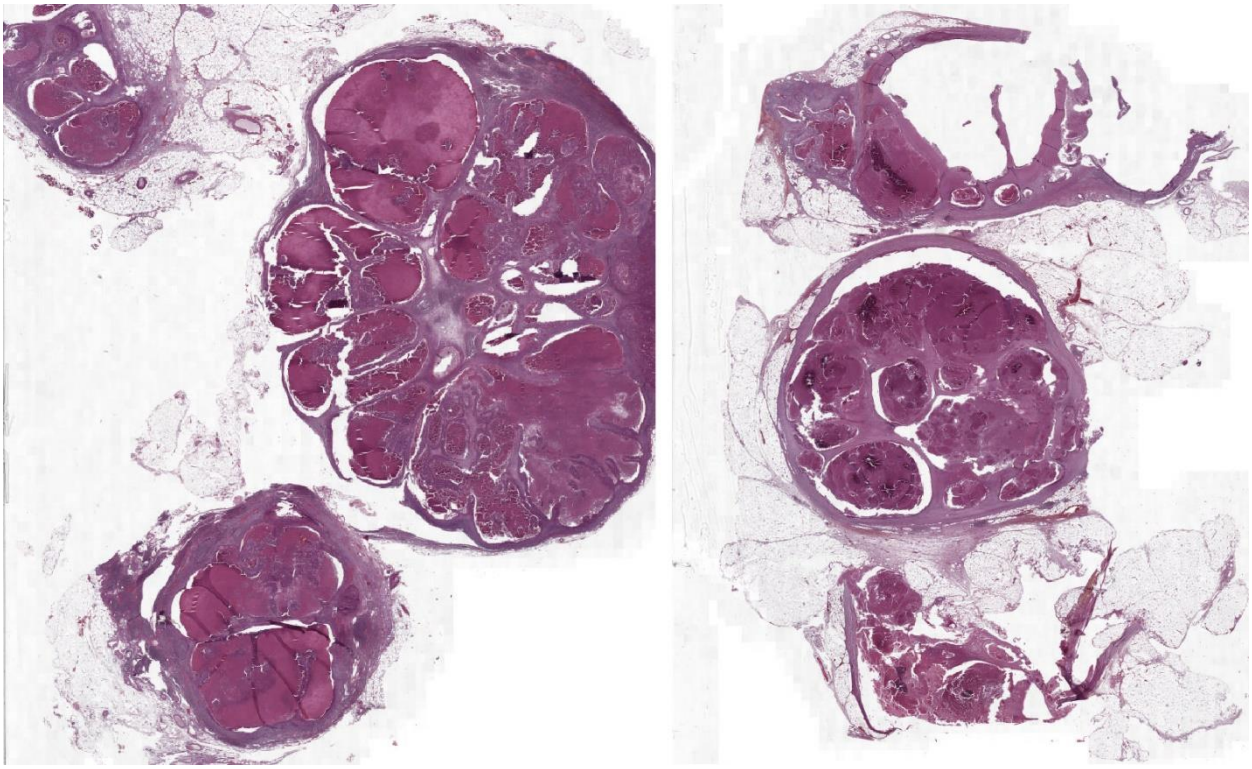


Figure 5.2: Lymph nodes full of necrosis (red tissue).

All this leads us to believe that detecting colon metastases is a harder task than detecting breast cancer metastases. In the future one may try improving upon effectiveness by doing hard negative mining which typically increases performance if the dataset contains rare, hard cases. Furthermore, one may investigate to what extent colon metastases is harder to detect and why.

5.2.1.3 Lower performance when adding data

However, the conclusion that colon metastases is harder to detect does not explain why the results decrease when using 20 instead of 10 training images. This appears to be caused by the model slightly overfitting towards the second task when using 20 training images. When training on 10 and 20 images, the validation set remains the same and has a size of 10. The training on 10 images reported 0.966 patch-level accuracy after 17 epochs where 20 images reached 0.968 after 24 epochs. So, the latter model appears to have slightly overfitted on the training set. The patch level accuracy for training with six images is even higher. However, to compare results of this training set with the head-neck dataset, the validation set only contains six images. The high accuracy is probably due to the validation set being very small. Therefore, this network also quickly overfits. A straightforward solution would be to add to the validation set in order to prevent this from happening. If the extra data is not available, careful tuning of the learning rate may also mitigate overfitting effects.

5.2.2 Dataset differences

Although to answer this research question, experiments on the head-neck dataset were not included, there are differences when observing the results of the head-neck dataset. The size is the same as the smallest colon training set size, but results are hardly comparable.

For the colon dataset, additional training does improve effectiveness. However, effectiveness remains low. This can have multiple reasons. First, it can be due to the test set being too small. Especially for the FROC metric, a few more false detections can greatly influence the score with a test set this small. In the colon test set, some rare cases were observed in the test set. However, effectiveness on most of the slides was much better, thus compensating for these rare test cases. With the current size of the head-neck test set, there is no such room for compensation.

Second, unlike the colon dataset, head-neck data may be easier to learn or, third, the dataset size in general can be too small to learn the new domain. The models with the higher learning rate overfit the training set, as effectiveness on the test set is not high but patch-level accuracy during training was as high as 0.991. With the lower learning rate, the patch-level accuracy reaches 0.971, which is a little higher but comparable to training accuracies reported for the colon dataset. With the lower learning rate, the maximum score is obtained already after 1 epoch. So, only a little training seems to be sufficient and using a higher learning rate tends to overfit the dataset quickly. However, for both learning rates, high effectiveness – even on the source dataset – cannot be achieved, which indicates that the dataset is too small.

To sum up, learning on the new images increases effectiveness notably, but there are only small differences between different training set sizes. However, this conclusion is dependent on the target domain. Differences should be sought in the size of the training-validation set in correspondence with the learning rate, when the model is unable to perform on both source and target tasks.

5.3 LEARNING RATE

To answer what the effect of the learning rate is, when doing transfer learning, five different conclusions can be distinguished. First, and most obvious, the higher learning rate is necessary to properly tune the network towards the target task. Performance on the target task is always higher when using the more aggressive learning rate.

Second, however, the more aggressive learning rate sometimes appears to be a little too aggressive. During training the network tends to overfit quickly, especially when a small training and validation set is used. This is discussed for the colon dataset in the previous section, but this extends also to the head-neck dataset.

Third, when using the lower learning rate, there is relatively little effectiveness loss on the source task. When, on the other hand, using the higher learning rate, catastrophic forgetting occurs as effectiveness drops on the source task. So, training with the lower learning rate does prevent from forgetting as it can just take smaller steps in altering the weights obtained by training on the source task. This is clearly visible from all experiments on the colon dataset. However, this is not the case for the head-neck task. For some domains it may therefore be necessary to use a learning rate that is in between both learning rates that have been experimented with, as the smaller is unable to learn the target data well and the larger overfits.

The fourth effect that the learning rate has, is that having an extra dataset can also be used as extra training data. Namely, effectiveness on the CAMELYON16 is increased after doing transfer learning. This is probably caused by the network seeing more tissue of a different kind, which can cause it to be a little more nuanced in difficult tissue parts. Although not effective for transfer learning, it is one of the unexpected outcomes of the experiments. If increased effectiveness on the source dataset is the goal, then the lower learning rate is better used.

Fifth, the learning rate is dependent on the type of data. Whereas on the colon dataset, effectiveness is best with the higher learning rate, on the head-neck dataset results are lower. These lower results are due to overfitting as training ended after an obtained patch level accuracy of 0.991 during training. It could be that the structure in the head-neck slides is learned faster compared to the colon dataset. However, this is just an indication as this is only based on experiments with a very small number of slides. It could also just be that the used slides are relatively easy cases.

5.4 Elastic Weight Consolidation

EWC can effectively be used to prevent catastrophic forgetting from happening. It comes at a very small cost, as effectiveness on the target task is sometimes for some metrics slightly worse. When only interested in the target task, EWC is maybe better left out. When aiming for good performance on both source and target tasks, EWC should be implemented. Especially when using the larger learning rate, which appeared to be essential to tune sufficiently towards the target task, EWC has a huge effect. Namely, effectiveness on the CAMELYON16 without EWC drops, but remains stable when EWC is implemented.

This conclusion is supported by all experiments on the colon dataset, but not by the experiment on the head-neck dataset with the higher learning rate, but, as explained in the previous paragraph, this is due to the network overfitting the target dataset.

5.5 Conclusion

In general, in this master's thesis project, it is asked to what extent transfer learning can be applied when creating a network for the CAMELYON16 challenge and transferring the information to different cancer metastases.

On the one hand, some domains are in general harder to learn. Although the domains share the upper domain of the lymphatic system, and effectiveness increases after transfer learning, results are still not close to CAMELYON16 performance. Even when training from scratch, the effectiveness scores were significantly lower. Whereas six training slides would suffice for the colon dataset, this was not enough data to tune towards the head-neck dataset.

On the other hand, transfer learning based on the CAMELYON16 challenge is possible. The architecture cannot be simple but does not need to be huge either. With a relatively high learning rate, the network is able to finetune towards the target dataset. Catastrophic forgetting then does occur but this can be prevented by exploiting EWC, which approaches maximum performance on the target task while maintaining high performance on the source task.

Transfer learning can thus be used to increase performance on both the source and target tasks. When developing a model that is related to the lymphatic system, initializing this model with CAMELYON16

weights is beneficial. A small amount of the target dataset can suffice to tune the network effectively towards the task at hand. This offers a range of possibilities for small or expensive datasets containing different types of cancer in lymph nodes. It is possible to use a single system to detect multiple types of cancer in the lymphatic area. The combined model is more effective than training from scratch. All this can be achieved with less training time and resources.

Future work could focus on optimizing the hyperparameters to achieve maximum performance. A high-quality dataset with verified annotations by pathologists can greatly contribute to comparing the effectiveness to the CAMELYON16 challenge and with regards to clinical practice. Different lymphatic domains can be added to analyze EWC in a continual learning setting. This could then be used to develop a single system distinguishing metastases from normal lymph node tissue in all of the lymphatic area.

BIBLIOGRAPHY

[1] H. Ritchie and M. Roser, "Causes of Death," *Published online at OurWorldInData.org*, 2018. [Online]. Available: https://ourworldindata.org/causes-of-death.

[2] B. Ehteshami Bejnordi *et al.*, "Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer," *Journal of the American Medical Association*, vol. 318, no. 22, p. 2199, Dec. 2017.

[3] A. E. Giuliano *et al.*, "Effect of Axillary Dissection vs No Axillary Dissection on 10-Year Overall Survival Among Women With Invasive Breast Cancer and Sentinel Node Metastasis: The ACOSOG Z0011 (Alliance) Randomized Clinical Trial.," *Journal of the American Medical Association*, vol. 318, no. 10, pp. 918–926, 2017.

[4] G. Litjens *et al.*, "1399 H&E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON dataset," *GigaScience*, vol. 7, no. 6, Jun. 2018.

[5] Y. Liu *et al.*, "Detecting Cancer Metastases on Gigapixel Pathology Images," *arXiv preprint arXiv:1703.02442*, 2017.

[6] R. Steiner, David F MacDonald *et al.*, "Impact of deep learning assistance on the histopathologic review of lymph nodes for metastatic breast cancer," *The American journal of surgical pathology*, vol. 42, no. 12, pp. 1636–1646, 2018.

[7] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, Dec. 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097--1105.

[9] P. Bandi *et al.*, "From detection of individual metastases to classification of lymph node status at the patient level: the CAMELYON17 challenge," *IEEE Transactions on Medical Imaging*, pp. 1–1, 2018.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision." pp. 2818–2826, 2016.

[11] F. Chollet, "Keras." 2015.

[12] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems." 2015.

[13] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," Dec. 2016.

[14] S. Ruder, "An overview of gradient descent optimization algorithms," *Computing Research Repository*, vol. abs/1609.0, 2016.

[15] P. J. Werbos, "Beyond regression : new tools for prediction and analysis in the behavioral sciences," 1974.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*.

[17] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview

and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018.

[18] J. G. Lee *et al.*, "Deep Learning in Medical Imaging: General Overview," *Korean J Radiol*, vol. 18, no. 4, pp. 570–584, 2017.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[21] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-term Dependencies with Gradient Descent is Difficult," *IEEE transactions on neural networks 5.2 (1994)*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[23] Y. Xu *et al.*, "Large scale tissue histopathology image classification, segmentation, and visualization via deep convolutional activation features," *BMC Bioinformatics*, vol. 18, no. 1, p. 281, May 2017.

[24] D. Komura and S. Ishikawa, "Machine Learning Methods for Histopathological Image Analysis," *Computational and Structural Biotechnology Journal*, vol. 16, pp. 34–42, 2018.

[25] B. Pfülb, A. Gepperth, S. Abdullah, and A. Kilian, "Catastrophic forgetting: still a problem for DNNs," in *International Conference on Artificial Neural Networks*, 2018, pp. 487–497.

[26] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in neural information processing systems*, 2017, pp. 4652–4662.

[27] J. K. C. Chan, "The Wonderful Colors of the Hematoxylin–Eosin Stain in Diagnostic Surgical Pathology," *International Journal of Surgical Pathology*, vol. 22, no. 1, pp. 12–32, 2014.

[28] G. J. S. Litjens, "Automate Slide Analysis Platform (ASAP)," 2018. [Online]. Available: https://github.com/geertlitjens/ASAP. [Accessed: 01-Dec-2018].

[29] P. Bandi *et al.*, "Comparison of different methods for tissue segmentation in histopathological whole-slide images," *2017 IEEE 14th International Symposium on Biomedical Imaging*, pp. 591–595, 2017.

[30] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[31] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017.

[32] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in Statistics*, Springer, 1992, pp. 569–593.

[33] D. P. Chakraborty, "Recent developments in imaging system assessment methodology, FROC analysis and the search model," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 648, pp. S297–S301, Aug.

2011.

[34] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, “Deep learning for identifying metastatic breast cancer,” *arXiv preprint arXiv:1606.05718*, 2016.

SUPPLEMENTS

A. COLON TRANSFER LEARNING

Exact results of all experiments done on the colon dataset. The results are given in the two tables below, with the former having a learning rate of 10^{-5} and the latter 10^{-3}.

Training – Validation slide ratio	EWC	Test set	FROC	ROC	DICE
6 – 6	×	Colon	**0.497** (0.500 – 0.507)	0.938 (0.936 – 0.940)	0.802
		CAMELYON	0.751 (0.750 – 0.754)	0.977 (0.976 – 0.977)	0.861
6 – 6	✓	Colon	0.455 (0.458 – 0.466)	0.917 (0.914 – 0.918)	0.802
		CAMELYON	0.759 (0.756 – 0.760)	0.979 (0.978 – 0.979)	0.859
10 – 10	×	Colon	0.467 (0.474 – 0.482)	**0.946** (0.944 – 0.947)	**0.812**
		CAMELYON	0.765 (0.765 – 0.770)	0.975 (0.975 – 0.976)	0.862
10 – 10	✓	Colon	0.424 (0.434 – 0.442)	0.931 (0.930 – 0.934)	0.808
		CAMELYON	0.763 (0.763 – 0.767)	0.974 (0.973 – 0.975)	0.862
20 – 10	×	Colon	0.496 (0.500 – 0.508)	**0.946** (0.945 – 0.948)	0.803
		CAMELYON	**0.773** (0.774 – 0.779)	**0.981** (0.981 – 0.982)	**0.867**
20 – 10	✓	Colon	0.425 (0.426 – 0.434)	0.925 (0.923 – 0.926)	0.804
		CAMELYON	0.768 (0.768 – 0.772)	0.978 (0.977 – 0.978)	0.863

Training – Validation slide ratio	EWC	Test set	FROC	ROC	DICE
6 – 6	×	Colon	**0.595** (0.598 – 0.605)	0.976 (0.975 – 0.977)	0.802
		CAMELYON	0.652 (0.655 – 0.661)	0.958 (0.957 – 0.959)	0.819
6 – 6	✓	Colon	0.575 (0.571 – 0.578)	0.982 (0.981 – 0.982)	0.812
		CAMELYON	0.738 (0.739 – 0.744)	0.986 (0.986 – 0.986)	0.853
10 – 10	×	Colon	0.586 (0.595 – 0.603)	0.962 (0.960 – 0.963)	0.823
		CAMELYON	0.447 (0.449 – 0.454)	0.920 (0.919 – 0.921)	0.792
10 – 10	✓	Colon	0.573 (0.578 – 0.586)	0.986 (0.985 – 0.987)	0.826
		CAMELYON	0.755 (0.754 – 0.759)	0.978 (0.977 – 0.979)	**0.860**
20 – 10	×	Colon	0.565 (0.571 – 0.579)	0.976 (0.976 – 0.977)	0.808
		CAMELYON	0.635 (0.638 – 0.643)	0.938 (0.936 – 0.938)	0.827
20 – 10	✓	Colon	0.540 (0.544 – 0.552)	**0.992** (0.992 – 0.992)	0.791
		CAMELYON	**0.759** (0.758 – 0.763)	0.980 (0.980 – 0.981)	0.855