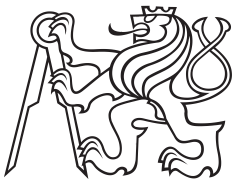
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

# Detection of Microscopic Fungi and Yeast in Clinical Samples

**Jakub Paplhám**

**Supervisor: Ing. Vojtěch Franc, Ph.D.**
**Field of study: Cybernetics & Robotics**
**May 2022**

# Acknowledgements

First and foremost, I would like to thank my supervisor, Ing. Vojtěch Franc, Ph.D., for his guidance, patience, and invaluable feedback.

My sincere thanks also go to MUDr. Daniela Lžičařová for collecting the dataset, providing expert insight into the task, and for continued interest in the project.

My heartfelt gratitude goes to my family, mother, brother, sister, and father, who has always been and always will be an inspiration to me.

Thank you also to my girlfriend Helena for her support and for putting up with me working long days instead of spending time with her.

Finally, I'd like to express my heartfelt gratitude to my friends.

---

I gratefully acknowledge the Center for Machine Perception's assistance in providing computational resources for this project.


# Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 19. May 2022

# Abstract

Early detection of yeast and filamentous fungi in clinical samples is critical in treating patients predisposed to severe infections caused by these organisms. The patients undergo regular screening, and the gathered samples are manually examined by trained personnel.

This thesis investigates the use of deep neural networks to detect filamentous fungi and yeast in the clinical samples to simplify the work of the human operator by filtering out samples that are clearly negative and presenting the operator with only samples suspected of containing the contaminant.

Domain-specific data augmentation techniques utilizing Poisson image editing and gradient-based localization are proposed to alleviate the scarcity of the data, and the model performance is compared against expert and novice level humans.

State-of-the-art models are successfully used to detect the contaminant. The method achieves human-level performance, theoretically reducing the amount of manual labor by 86.9%, given a true positive rate of 99%.



**Keywords:** filamentous fungi, yeast, convolutional neural networks, automated detection, automated microscopy, fluorescence staining, Poisson image editing, gradient-based localization



**Supervisor:** Ing. Vojtěch Franc, Ph.D.
Katedra kybernetiky, ČVUT FEL
Na Zderaze 269/4,
121 35 Praha 2



# Abstrakt

Včasná detekce kvasinek a vláknitých hub v klinických vzorcích má zásadní význam pro léčbu pacientů náchylných k závažným infekcím způsobeným těmito organismy. Pacienti podstupují pravidelný screening a odebrané vzorky jsou manuálně vyšetřovány školeným personálem.

Tato práce se zabývá využitím hlubokých neuronových sítí k detekci vláknitých hub a kvasinek v klinických vzorcích, za účelem zjednodušení práce lidské obsluhy tím, že se nejprve odfiltrují vzorky, které jsou jednoznačně negativní, a obsluze se následně předloží pouze vzorky s podezřením na obsah kontaminantů.

Pro zmírnění nedostatku dat jsou v práci navrženy techniky rozšíření dat specifické pro tuto úlohu, které využívají Poissonovu úpravu obrazu a lokalizaci založenou na gradientu. Funkčnost modelu je následně srovnána s odborníky a začátečníky.

K detekci kontaminantu jsou úspěšně použity state-of-the-art modely. Metoda dosahuje výsledků na úrovni člověka a teoreticky snižuje množství manuální práce o 86.9% při sensitivitě 99%.



**Klíčová slova:** vláknité houby, kvasinky, konvoluční neuronové sítě, automatická detekce, automatizovaná mikroskopie, fluorescenční barvení, Poissonova úprava obrazu, lokalizace založená na gradientu

**Překlad názvu:** Detekce mikroskopických hub v klinickém materiálu

# Contents

# Figures

# Tables

# Chapter 1

# Introduction

Early detection of yeast and filamentous fungi in clinical samples is critical in treating patients predisposed to severe infections caused by these organisms. Fluorescence microscopy is a suitable method for this detection, where after application to a slide, the material is stained with a fluorescent dye (e.g., Calcofluor White), which binds to chitin contained in the fungal cell wall. This staining process is non-specific, as other structures that may occur accidentally in the sample (e.g., dust, pollen, arthropods) can also bind the dye. The samples commonly consist of respiratory secretions or non-invasive tissue biopsy samples; the aforementioned foreign bodies are therefore routinely present.

Severe infections caused by filamentous fungi are sporadic but severe; thus, patients with a risk factor undergo regular screening. It follows that a considerable number of slides must be carefully examined, most of which do not contain yeast or filamentous fungi. Fungi and yeast cells are currently detected manually by trained personnel based on their typical morphology. Laboratory staff then spend a significant amount of time examining negative samples, which leads to job dissatisfaction, and development of musculoskeletal disorders caused by repetitive stress injuries, [George, 2010].

Further, there has recently been an increase in demand for clinical screening as evidence has shown an association between severe respiratory diseases and fungal infections, e.g., prevalent pulmonary aspergillosis in patients with acute respiratory distress syndrome, [Lai and Yu, 2021]. The combination of increased demand and a shortage of skilled personnel[1] [Garcia et al., 2021] makes the detection of fungal infections an ideal candidate for automation.

Automatic detection and diagnosis of infectious diseases from clinical samples is an area of ongoing research. Classically, the automation was performed by detecting specific morphological attributes of fungi and other manually designed features in microscopic images and yielded unsatisfactory results. With the advent of machine learning, manually designed methods were gradually replaced by deep learning, particularly by convolutional neural networks (CNNs).

Recently, automated slide scanners were used for slide imaging and CNNs

[1]The worker shortage is reported by The American Society for Clinical Pathology; however, the expert consensus is that similar issues are present in Europe.

for evaluation of the data. This allowed for fully automatic detection and classification of bacteria in Gram stains of blood culture, [Smith et al., 2018], and detection of intestinal protozoa in trichrome stained stool samples, [Mathison et al., 2020]. Gram staining dyes were also applied to samples containing yeast and yeast-like fungi, allowing their successful classification, [Zieliński et al., 2020]. Perhaps most similar to the goals of this work, [Gao et al., 2021] fully automate the process of scanning and classification of fluorescent stained skin samples containing fungi.

This work was done in collaboration with Motol University Hospital, whose staff amassed a unique dataset of fluorescence microscopy images over a period of several years in order to develop an automated system to detect microscopic filamentous fungi and yeast. The ultimate goal is to develop a fully automated system that reduces the amount of manual work by using an automated slide scanner and an automatic classifier. The automated system's mode of operation would be to filter out samples that are easily distinguishable as negative and present the remaining potentially positive samples to a medical expert for verification. An example of positive and negative images is shown in Figures 1.1 and 1.2 respectively.

This work's goal is to be an intermediate step toward the creation of the automated system. The work's contributions are the following:

- proof-of-concept detector based on deep CNNs is created,
- novel data augmentation techniques specific to the task are proposed,
- performance of the developed model is evaluated and compared against expert and novice level humans.

The main challenges of the work are twofold: (i) the amount of available data is relatively low, and (ii) positive and negative images have a high degree of similarity. The dataset contains high-resolution images, which are a priori assumed to be negative. The images become positive when structures specific to the contaminant are detected, even if low-resolution and present only in a small portion of the image, i.e., a positive and a negative image can be identical except for a small region of the image. An example of such a case, when the contaminant is present in a very small portion of the image, is shown in Figure 1.3.

The structure of the thesis is as follows. Chapter 2 examines state-of-the-art approaches to medical image classification. The methodology is described in Chapter 3. The experiments and their results are described in Chapter 4. Chapter 5 discusses the findings and Chapter 6 concludes the thesis.

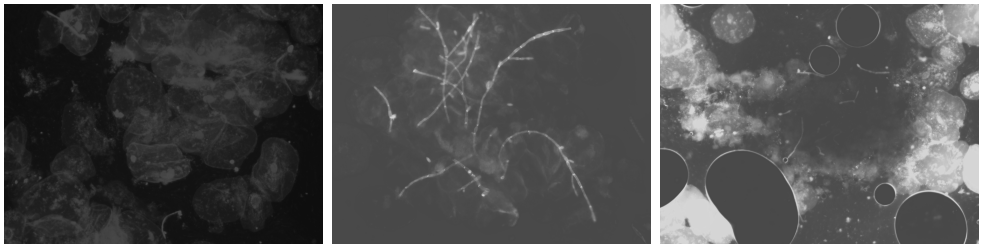


**Figure 1.1:** Example of positive images from the dataset.

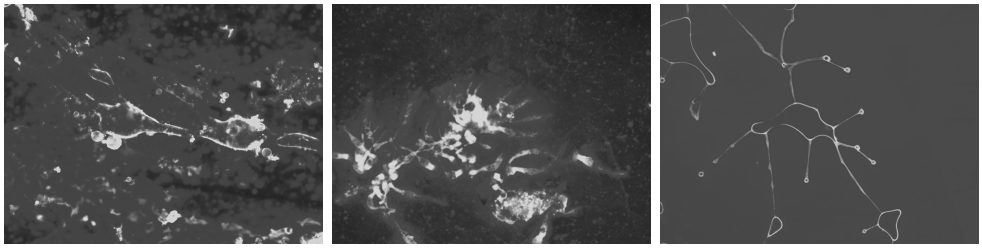


**Figure 1.2:** Example of negative images from the dataset.

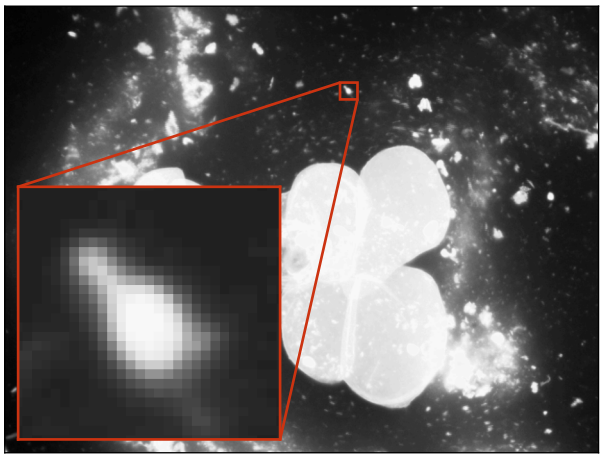


**Figure 1.3:** Except for a single budding yeast cell, there is no contaminant in the image. The yeast cell takes up only a small portion of the image, but it is entirely responsible for the final classification.

# Chapter 2

# The State-of-the-art

Historically, machine learning techniques have attained promising results on a wide range of problems. Still, they have not been widely used in medical diagnostics due to a lack of trust in a machine's decisions regarding human health. However, as data-driven methods began to outperform human capabilities, these automated techniques gradually crept into medical image diagnostics, where they are successfully used to detect a wide range of diseases. Models trained on the ChestX-ray8 dataset, [Wang et al., 2017], facilitate early detection of common thorax diseases from chest radiography. Images of dermatoscopic melanoma, [Rotemberg et al., 2020], enable discerning between benign and malign skin lesions, and histopathological images collected by [Koziarski et al., 2021] allow for reliable prostate cancer detection. [Ali et al., 2020] implement a deep-learning-based system capable of early heart disease detection.

The distrust of having a machine make decisions about one's health has somewhat diminished, and deep learning has become common practice in many healthcare facilities.

This chapter reviews state-of-the-art approaches to image classification, Section 2.1, and successful application of these methods to tasks similar to the detection of microscopic fungi and yeast in clinical samples, Section 2.2.

## 2.1 Deep Learning for Image Classification

### 2.1.1 Model Architectures

Image classification has been dominated by CNNs since the victory of AlexNet, [Krizhevsky et al., 2012], in the 2012 ImageNet competition. CNNs have become increasingly popular each year, and significant breakthroughs have been made both in the design of the networks and in the training procedure.

It was discovered that the performance of CNNs improves when more layers are added. Training deep networks was, however, complicated. The skip-connections introduced in the ResNet architecture, [He et al., 2016a], allowed for easy training of these deep networks, and yet deeper models continued to be developed. A race started of who could train the most extensive network. The state-of-the-art models grew from millions of parameters to hundreds

of millions. Still, the diminishing returns motivated the researchers to shift their interest to discovering more efficient architectures.

[Tan and Le, 2019] show that increasing the depth, width, and resolution of the network in unison is superior to increasing the depth in isolation. They propose EfficientNet, a CNN architecture that outperforms previous methods yet is computationally tractable with regular hardware.

The success of CNNs in computer vision tasks can largely be attributed to their intrinsic inductive biases, such as locality and invariance to translation. They were primarily inspired by the visual cortex of the brain[1] and were designed such that they mimic how humans process images. However, recently [Dosovitskiy et al., 2021] show that these inductive biases are unnecessary or even detrimental when vast amounts of training data are available. They train a Vision Transformer model, utilizing *attention* blocks introduced by [Vaswani et al., 2017] for natural language processing, and reach new state-of-the-art.

Results competitive with the Vision Transformer (ViT) are also attained by [Tolstikhin et al., 2021], utilizing a model based exclusively on multi-layer perceptrons (MLPs). This finding supports the claim that the inductive biases of CNNs are superfluous and that more general models with sufficient expressivity can better utilize large amounts of training data.

As of the time of writing, CNNs remain competitive with transformers by utilizing semi-supervised training, [Xie et al., 2020], [Pham et al., 2021] and scaling up previously established architectures, [Kolesnikov et al., 2020], [Tan and Le, 2021], with modifications.

### 2.1.2 Data Augmentation

At the moment, the most effective techniques are data-driven. When there is enough data, stronger knowledge can be extracted directly from the data rather than relying on manually designed features. However, data is frequently limited, and gathering additional data is costly. Domain adaptation techniques are one approach to dealing with this. Another strategy is to use data augmentation techniques.

Data augmentation is the process of increasing the amount of data available by creating synthetic data from existing data. Assuming that the available data are generated by some underlying probability distribution, data augmentation techniques would ideally sample new data from the distribution. This can sometimes be accomplished by applying transformations that do not change the class to existing data, such as creating a new realistic image by slightly rotating an existing image. As obtaining samples from the underlying distribution is often impossible, an alternative approach to data augmentation is to generate samples that, while not from the distribution, contain certain features that allow or even force the models to learn the distribution's features.

[1] For more information, refer to the theoretical background in [Paplhám, 2020].

Some standard image augmentation techniques produce a realistic-looking new image by applying natural transformations, such as horizontal/vertical mirroring, rotation, scaling, or deforming the image. Other augmentations, such as the addition of noise and minor changes in the image's brightness, hue, or contrast, attempt to change the values of the pixels while maintaining the overall visual content of the image. Another type of standard augmentations removes information from the images, e.g., cropping or occluding random portions of the image.

Other augmentations are less intuitive. For example, the mixup augmentation, [Zhang et al., 2017], generates convex combinations of pairs of data samples and their labels, resulting in strong regularisation and increased model robustness. CutMix, [Yun et al., 2019], uses a similar technique to combine data samples, swapping cropped regions of the images and assigning a convex combination of the labels to the result.

Much potential has recently also been attributed to augmentation methods utilizing generative models. E.g., generative adversarial networks (GANs) have been used to generate synthetic, realistic-looking scans of liver lesions, [Frid-Adar et al., 2018], for a task with a meager amount of available data. Furthermore, the output of the generative models can be guided to produce very specific augmentations; for example, [Aakash Saboo and Wang, 2021] show how GANs in combination with latent-space optimization can be used to generate images depicting consistently progressing stages of a disease.

An alternative use of generative models is to generate synthetic data directly in the model's latent space, [Du et al., 2022], rather than generating realistic synthetic data to be used as an input.

## 2.2 Relevant Medical Applications

In the following section, Section 2.2.1, we review works most relevant to our goal, attempting to automate the evaluation of clinical samples observed through a microscope. We provide a summary of the approaches in Section 2.2.2.

### 2.2.1 Prior Research

This section describes previous research in a similar domain to our work, analyzing medical images collected with a microscope.

#### Automated Interpretation of Blood Culture Gram Stains by Use of a Deep Convolutional Neural Network

[Smith et al., 2018] use an automated slide scanner and a 40× optical lens to record non-overlapping images from predefined locations of Gram-stained blood culture samples containing bacteria and classify them into four classes.

They follow the industry-standard approach of taking a pre-trained CNN[2]

[2]Inception-V3, [Szegedy et al., 2015].

and fine-tuning the fully-connected classifier head on a downstream task. Each captured image is classified individually with a stringent, manually selected classification threshold[3] $\theta = 0.99$, to minimize the amount of false-positive predictions. The classification for the whole slide is obtained using a majority class vote. If no class accumulates six (of 192 total) or more votes, the slide is classified as negative.

The authors propose standardizing the sample acquisition and staining process. This would reduce the number of modalities the network needs to learn, e.g., multiple possible backgrounds. It would also allow the development of a specialized focusing algorithm suitable for the task, as some of the images taken by the automated scanner were out of focus. They achieve a slide-level accuracy of 92.5%.

### Deep Learning Approach to Describe and Classify Fungi Microscopic Images

[Zieliński et al., 2020] classify microscopic images of yeast-like fungi, colored using standard Gram staining equipment, into nine separate classes. The images were taken using a 100× optical lens.

They utilize a pre-trained CNN[4] as the fixed backbone and remove the fully-connected classification head to replace it with a bag-of-words encoding followed by a support-vector-machine (SVM) classifier. They show that their modification provides better results than the former classifier head. An identical approach was previously used by [Zieliński et al., 2017] for the classification of bacterial colonies. It should be noted that the backbone was *locked* and not modified. Therefore, it is unclear whether the novel approach would achieve consistently better results if the backbone were allowed to fine-tune its weights and a more sophisticated training protocol was used.

Image of the stained slide is captured using an automated slide scanner. Regions of interest, "patches", are discovered by thresholding the image, and each patch is classified individually. The classification for the whole slide is obtained using a majority class vote.

The authors present their method as a means of identifying the species rather than simply detecting its presence. They claim that their approach makes the last stage of conventional biochemical identification unnecessary and thereby replaces the currently used *fluorescence in situ hybridization* (FISH) or *polymerase chain reaction* (PCR) analysis of replicated DNA. They achieve patch-level and slide-level accuracy of 82.4% and 93.9% respectively.

### Detection of Intestinal Protozoa in Trichrome-Stained Stool Specimens by Use of a Deep Convolutional Neural Network

[Mathison et al., 2020] use an automated slide scanner to record images of trichrome stained stool, with a 40× lens and an optical doubler, in three

[3] We refer the reader to Section 3.2 for detailed definition.
[4] AlexNet, [Krizhevsky et al., 2012] or ResNet [He et al., 2016a].

planes of Z-stack. The images are processed using a pre-trained CNN[5], which classifies the images into one of 11 classes and predicts bounding boxes around the contaminants of interest - the object detection/localization task, with the goal of detecting intestinal protozoa.

They show that the CNN-based method can detect protozoa in heavily diluted solution, up to the ratio of $1 : 256$, while human personnel is able to detect the specimen in dilutions of at most $1 : 8$ ratio. Further, their method achieves slide-level accuracy equal to a human evaluation.

They successfully deploy the model in a real-life setting, with the model presenting its results to a human expert for confirmation.

### 2.2.2 Summary

All of the listed methods utilize the standard technique of fine-tuning a pre-trained CNN. They employ an automated scanner to obtain a large number of images from each sample, classify the images separately, then aggregate the result over the images to classify the sample. An identical approach can be observed in other research in the field of automatic evaluation of digital microscopy.

The methods achieve human-level performance, motivating future large-scale deployment. Some novelty can be observed in replacing the linear classification head with bag-of-words encoding followed by an SVM classifier. To our knowledge, no notable domain-specific modifications of the standard techniques were used in these works.

[5] A single-shot multibox detector Inception-V2, [Szegedy et al., 2015].

# Chapter 3

# Methods

The methodology used to carry out the experiments is described in this Chapter. Section 3.1 introduces the dataset. Section 3.2 defines the task of binary classification and introduces notation used in the remainder of the thesis. Section 3.3 defines and analyzes the evaluation metric used to compare different models. Section 3.4 contains technical details of the implementation, lists the models used, and establishes the training protocol. Section 3.5 describes a novel data augmentation technique, tailored to the task of contaminant detection. Section 3.6 describes the evaluation procedure.

## 3.1 Dataset

The dataset used for training and evaluation of the models was collected and manually annotated by a medical expert from January 2018 to April 2021. Clinical material (sputum, endotracheal or bronchial aspirate, bronchoalveolar fluid, tissue, pleural fluid, pericardial fluid, cerebrospinal fluid, liquid or solid contents of pathological cavities) was smeared on a sterile slide and dried. The dried slides were dyed with Calcofluor White mixed 1 : 1 with 20% potassium hydroxide solution and immediately covered with cover slides and examined. Fluorescence microscopy was performed manually with the use of Olympus BX 53 fluorescence microscope, UplanFLN 20x objective lens, FN 26,5. The entire slide was examined, and a representative section of the slide was selected, i.e., for positive slides, the section containing the contaminant. The image of the selected section was captured using an Olympus DP72 microscope digital camera.

The manually captured dataset contains a total of 1244 high-resolution images. The dimensions of the images are not identical. However, the aspect ratio is constant throughout the dataset, and each image captures the same field of view. The number of images with corresponding dimensions is shown in Table 3.1.

Annotations are given in the form of the binary label (positive/negative) for each image, that is, the region of interest is not specified. In other words, the annotation provides no information on the location or size of the specimen within the image.

It should be noted that multiple images in the dataset could originate

from the same slide. The images are thus not statistically independent. The dependency was ignored, as there was no possibility of retrospectively deciding which slide an image was taken from.

| Width × Height | Annotation | |
|---|---|---|
| | Positive | Negative |
| 4140 × 3096 | 374 | 546 |
| 2040 × 1536 | 77 | 3 |
| 1360 × 1024 | 231 | 13 |
| Total | 682 | 562 |

**Table 3.1:** Dimensions of images in the dataset.

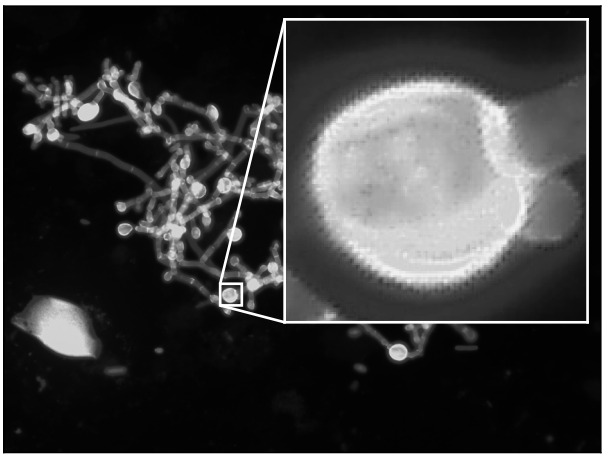


**Figure 3.1:** Imaging artifacts in the dataset. Because of the artifacts, the images can most likely be downsampled without significant information loss.

The positive images were further separated into either (i) fungi: 545, (ii) yeast: 61, or (iii) both yeast & fungi: 75 images, with the majority of images containing solely fungi. Note, however, that this separation was performed by the author of the thesis, who is not an expert in medical biology, i.e., the labels can be noisy.

The images were obtained encoded in JPG format in the RGB color space. The data was converted to PNG format to reduce the number of artifacts created in further processing and converted to grayscale colorspace. The motivation and the procedure are explained in detail in Section 4.1.5.

### Scale

As the images have approximately the same aspect ratio of $\frac{\text{Width}}{\text{Height}} \approx 1.33$, they can be resized to a uniform shape. This allows for mini-batching of the data, improving the training efficiency.

We consider the smallest shape among images in the dataset, $1360 \times 1024$ pixels, as the *base* shape. To discuss different image sizes, we define *scale* of an image as the coefficient which relates the image size to the base size, e.g., $scale = 50\%$ signifies images downscaled to shape $680 \times 512$ pixels.

Training networks on smaller scales is less computationally demanding, whereas higher scales preserve more detail, potentially allowing for superior performance. The original, high-resolution images, however, are not precisely in focus and contain artifacts, as shown in Figure 3.1. This further encourages downscaling because the loss of information should be minor.

We evaluate and report the effects of image scale on performance in Section 4.1.1.

## 3.2 Classification

Detection of yeast and fungi in images is treated as a binary classification problem. In this section, we formalize the task and introduce the necessary notation used later in the thesis.

**Binary Classifier.** Binary classification is a task of assigning a binary class label to an observation of features. Formally, a binary classifier is a mapping

$$h : \mathcal{X} \to \mathcal{Y}, \tag{3.1}$$

from the set of features $\mathcal{X}$ to the set of labels $\mathcal{Y} = \{1+, -1\}$.

**Sample.** By a sample, we refer to either (i) a clinical sample, biological material prepared in a slide, (ii) the pair $(x, y) \in \mathcal{X} \times \{+1, -1\}$, or (iii) the feature $x \in \mathcal{X}$, depending on the context. We associate the label $+1$ with presence of the contaminant and refer to samples $\{(x, +1) \mid x \in \mathcal{X}\}$ as positive samples. Complementarily, we associate the label $-1$ with absence of the contaminant and refer to samples $\{(x, -1) \mid x \in \mathcal{X}\}$ as negative samples.

The samples $(x, y)$ are assumed to be generated by a random process with the distribution $p(x, y)$ defined on $\mathcal{X} \times \{+1, -1\}$. We use $p(x)$ and $p(x \mid y)$ to denote the marginal and the class conditional distribution of the inputs $x$, respectively. The prior probability of the positive and the negative class are denoted $p(y = +1)$ and $p(y = -1)$, respectively. The expected value of $f(z)$ w.r.t. the distribution $p(z)$ is denoted by $\mathbb{E}_{z \sim p(z)} f(z)$.

**Image Classification.** Many classification models exist and are chosen based on the specific properties of the input features. The task of image classification necessitates special consideration due to its uniqueness in two ways: (i) the feature space is vast, and (ii) the feature values (pixels) are interdependent and only weakly connected to the class label when considered in isolation, i.e., knowing the color of a single pixel provides minimal information. Prior to the rise of deep learning, the traditional approach to processing images was to reduce the size of the feature space by using hand-designed feature extractors, such as edge detectors or texture detectors, to express the general

properties of the image more compactly. E.g., instead of each pixel color being encoded individually, the overall color tones of the image are expressed. Current state-of-the-art models, such as convolutional neural networks, which are used in this thesis, learn these feature extractors from the data.

**Binary Image Classifier.** The task at hand is a binary image classification task. We continue with a formal definition. Let us denote the common pixel domain as

$$\mathcal{D} \subset \mathbb{Z}^2, \tag{3.2}$$

the color domain and the monochromatic domain are defined, respectively as

$$\mathcal{C} \subset \{(r, g, b) \mid r, g, b \in \mathbb{R}\}, \qquad \mathcal{M} \subset \{x \mid x \in \mathbb{R}\}, \tag{3.3}$$

where lower and upper bounds of the values and machine precision are ignored for simplicity. The set of all possible images is then

$$\mathcal{I} = \mathcal{C}^{\mathcal{D}} \cup \mathcal{M}^{\mathcal{D}}. \tag{3.4}$$

A binary image classifier is the mapping

$$h : \mathcal{X} \to \mathcal{Y}, \tag{3.5}$$
$$\mathcal{X} = \mathcal{I}, \tag{3.6}$$
$$\mathcal{Y} = \{+1, -1\}. \tag{3.7}$$

Often the classifier is further decomposed as

$$h = f \circ d, \tag{3.8}$$
$$f : \mathcal{X} \to \mathbb{R}, \tag{3.9}$$
$$d : \mathbb{R} \to \{+1, -1\}, \tag{3.10}$$

with the decoding mapping $d$ defined as

$$d(x, \theta) = \begin{cases} +1 & \text{for } x \geq \theta, \\ -1 & \text{for } x < \theta, \end{cases} \tag{3.11}$$

for some fixed threshold $\theta \in \mathbb{R}$.

**Ensemble of Binary Image Classifiers.** In the rest of the thesis, by an ensemble of binary classifiers, we refer to a binary classifier where the prediction rule $f$ is defined as

$$f(x) = \frac{\sum_{i=1}^{n} f_i(x)}{n}, \qquad f_i : \mathcal{X} \to \mathbb{R}. \tag{3.12}$$

By *weighted* ensemble, we refer to a binary classifier with the prediction rule

$$f(x) = \frac{\sum_{i=1}^{n} \alpha_i \cdot f_i(x)}{\sum_{i=1}^{n} \alpha_i}, \qquad f_i : \mathcal{X} \to \mathbb{R}. \tag{3.13}$$

Sometimes, the ensemble is also used to refer to the set of classifiers

$$\{f_1, f_2, \ldots, f_n \mid f_i : \mathcal{X} \to \mathbb{R}\}, \tag{3.14}$$

used in the prediction rule $f(x)$.

## 3.3 Metrics

The development of any machine learning system includes evaluation of a performance metric. The metric is used to measure the model's performance and numerically express the expected results so that the best method can be selected for the final implementation.

In the case of a detection task, the model is a binary classifier. The contaminant of interest is detected, or it is not detected. Traditional metrics such as accuracy, precision, and recall are helpful, but we argue that they are not ideal for selecting a model to be used within a medical facility. Instead, we propose an alternative metric, the *saved time* metric, which is a scalar measure of the model's utility for clinical use. The saved time metric is defined as a portion of input samples which do not need to be manually evaluated by laboratory staff. Equivalently, the metric directly represents the number of man-hours that the model saves. The introduced metric has two advantages. First, it has a clear interpretation. Second, the scalar metric allows for easy model comparison, accelerating the development. We continue with a formal definition and analysis.

### 3.3.1 Saved Time Metric

Assuming that manual examination of each sample takes constant time, the saved time is directly proportional to the number of samples which need not be examined by human personnel. If the clinical sample is to be classified as positive, manual confirmation is required. This ensures that a patient is not diagnosed with infection until reviewed by a qualified human expert, increasing the patients' confidence in the diagnosis. Therefore, the saved time is proportional to the number of samples classified as negative.

One could achieve the maximal value of such a metric simply by classifying all samples as negative. This is undesirable. We, therefore, define the saved time metric as the portion of samples classified as negative while guaranteeing that the true positive rate is higher than a specified level.

Formally, we evaluate the prediction rule $h\colon \mathcal{X} \to \{+1, -1\}$ in terms of two metrics. First, the true positive rate (a.k.a. sensitivity)

$$\mathrm{TPR}(h) = \mathbb{E}_{x \sim p(x|y=+1)}[\![h(x) = +1]\!], \tag{3.15}$$

which is the probability that a positive sample is correctly predicted as positive. Secondly, the saved time

$$\mathrm{ST}(h) = \mathbb{E}_{x \sim p(x)}[\![h(x) = -1]\!], \tag{3.16}$$

equal to the probability that an input sample is predicted as negative. It is useful to rewrite the saved time as

$$\begin{aligned} \mathrm{ST}(h) = {} & [1 - p(y = +1)] \cdot [1 - \mathrm{FPR}(h)] \\ & + p(y = +1) \cdot [1 - \mathrm{TPR}(h)], \end{aligned} \tag{3.17}$$

where $p(y = +1)$ is the prior probability of the positive class, and

$$\mathrm{FPR}(h) = \mathbb{E}_{x \sim p(x|y=-1)}[\![h(x) = +1]\!] \tag{3.18}$$

is the false positive rate, i.e., the probability that a negative sample is incorrectly classified as positive. The equation (3.17) shows that the two metrics, $\mathrm{TPR}(h)$ and $\mathrm{ST}(h)$, are antagonistic, i.e., increasing one leads to a decrease of the other and vice versa.

The predictor is a binary image classifier of the form

$$h(x;\theta) = \begin{cases} +1 & \text{for } f(x) \geq \theta, \\ -1 & \text{for } f(x) < \theta, \end{cases} \tag{3.19}$$

where $f\colon \mathcal{X} \to \mathbb{R}$ is a score trained from examples and $\theta \in \mathbb{R}$ is a decision threshold used to tune the operating point of the predictor. With a slight abuse of notation, we use $\mathrm{TPR}(\theta)$ and $\mathrm{ST}(\theta)$ as a shortcut for $\mathrm{TPR}(h(\cdot;\theta))$ and $\mathrm{ST}(h(\cdot;\theta))$, respectively.

The number of positive and negative samples in the test set is approximately the same, which does not match the real distribution at the deployment time. According to Motol University Hospital's staff, approximately 9 out of 10 samples that arrive at the laboratory for examination are negative. As a result, for the remainder of the thesis, we assume that the incidence rate is $p(y = +1) = 0.1$. However, the methodology as a whole is general and, if necessary, can be applied to any incidence rate. When evaluating the metrics from the data, we resolve the mentioned mismatch as follows. Given a test set $\{(x_i, y_i) \in \mathcal{X} \times \{+1, -1\} \mid i = 1, \ldots, n\}$, we first compute the empirical estimates of $\mathrm{TPR}(\theta)$ and $\mathrm{FPR}(\theta)$ as

$$\widehat{\mathrm{TPR}}(\theta) = \frac{1}{n_+} \sum_{i=1}^{n} [\![h(x_i;\theta) = +1 \wedge y_i = +1]\!], \tag{3.20}$$

$$\widehat{\mathrm{FPR}}(\theta) = \frac{1}{n_-} \sum_{i=1}^{n} [\![h(x_i;\theta) = +1 \wedge y_i = -1]\!], \tag{3.21}$$

where $n_+ = \sum_{i=1}^{n} [\![y_i = +1]\!]$ and $n_- = \sum_{i=1}^{n} [\![y_i = -1]\!]$. Then, we fix the positive class prior to the expert estimate of the incidence rate, $p(y = +1) = 0.1$, and compute the empirical estimate of the saved time

$$\begin{aligned} \widehat{\mathrm{ST}}(\theta) = {} & [1 - p(y = +1)] \cdot [1 - \widehat{\mathrm{FPR}}(\theta)] \\ & + p(y = +1) \cdot \left[1 - \widehat{\mathrm{TPR}}(\theta)\right]. \end{aligned} \tag{3.22}$$

We evaluate the predictor $h$ by a curve $\{(\widehat{\mathrm{TPR}}(\theta), \widehat{\mathrm{ST}}(\theta)) \mid \theta \in (-\infty, \infty)\}$ which summarizes the entire space of achievable true positive rates and saved times. As a reference, we also plot the best achievable saved time curve as a function of TPR, i.e., we plot the curve $\{(\mathrm{TPR}, \mathrm{ST}^*(\mathrm{TPR})) \mid \mathrm{TPR} \in (0, 1)\}$ where $\mathrm{ST}^*(\mathrm{TPR}) = p(y = +1) \cdot [1 - \mathrm{TPR}] + [1 - p(y = +1)]$, which is obtained from (3.17) when assuming an ideal predictor with zero $\mathrm{FPR}(h)$.

In case we need to evaluate the predictor by a single scalar, e.g., when ranking different models, we report the saved time at desired true positive rate $\tau$, which is defined as

$$\widehat{\mathrm{ST}}\tau = \max_{\theta \in (-\infty,\infty)} \widehat{\mathrm{ST}}(\theta) \quad \text{subject to} \quad \widehat{\mathrm{TPR}}(\theta) \geq \tau\,. \tag{3.23}$$

### Visual Example

In this section, we illustrate the concept of the saved time in a simple setting when the predictor's input is scalar, $x \in \mathbb{R}$ and the class distributions $p(x \mid y = +1)$, $p(x \mid y = -1)$ are univariate Gaussians.

Threshold $\theta$ of the classifier is chosen such that ratio of areas $\mathrm{TPR} = \frac{\text{blue}}{\text{purple} + \text{blue}}$ in Figure 3.2 is larger than desired $\mathrm{TPR}_{\mathrm{const.}}$.

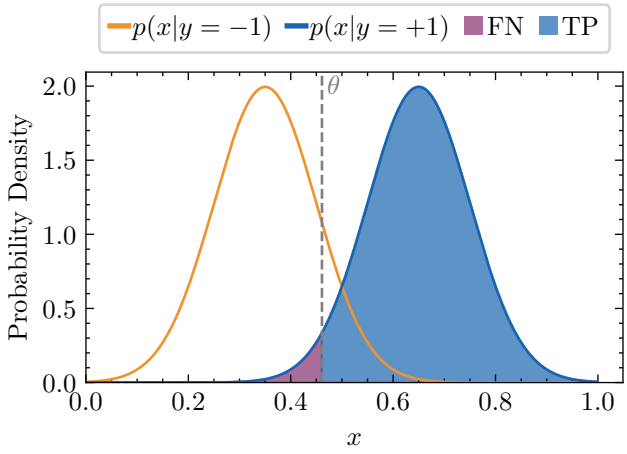



**Figure 3.2:** Illustration of the sensitivity of a binary classifier with a known Gaussian probability distribution of classes.

When the threshold is determined, the proposed metric, percentage of all images, which will be classified as negative, can be interpreted as

$$\mathrm{ST} = \int_{-\infty}^{\theta} p(y=+1)\cdot p(x|y=+1) + p(y=-1)\cdot p(x|y=-1)\; dx. \tag{3.24}$$

Visually, it is the ratio ST of areas in Figure 3.3, where

$$\mathrm{ST} = \frac{\text{orange} + \text{purple}}{\text{orange} + \text{purple} + \text{red} + \text{blue}}. \tag{3.25}$$

We provide another illustrative example in Figure 3.4, where the inputs are points in a 2-D plane. The area to the left/right of the vertical line equates to the samples, which should be classified as positive/negative. The classifier is an ellipse, where points inside of the ellipse relate to the positive class and points outside the ellipse relate to the negative class. We continue with commentary for each of the subplots in Figure 3.4.

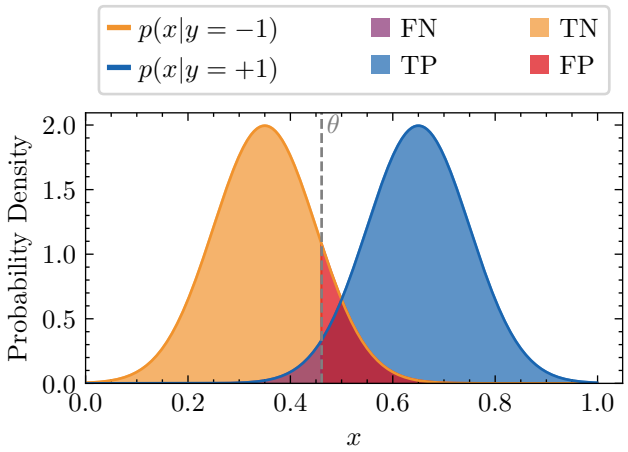



**Figure 3.3:** Illustration of the saved time metric of a binary classifier with a known Gaussian probability distribution of classes.

**Case a):** The saved time metric is high, as large number of samples is classified negative (purple + orange). These would not be manually examined.

The classifier, however, does not satisfy the condition of high sensitivity due to a large number of positive samples being classified as negative (purple). With such low sensitivity, the classifier could not be deployed.

**Case b):** The classifier satisfies the condition of high sensitivity as only a small number of positive samples is classified as negative (purple). Therefore the classifier could be deployed for analysis of clinical samples.

The saved time metric however is low, as a small number of samples is classified negative (purple + orange). The classifier would thus be usable but would not significantly lower the number of manual examinations.

**Case c):** The classifier satisfies the condition of high sensitivity as only a small number of positive samples is classified as negative (purple). Therefore the classifier could be deployed for analysis of clinical samples.

The saved time metric is high as a large number of samples is classified negative (purple + orange). The classifier would thus be ideal for deployment.

## 3.4 Implementation Details

This section goes over the specifics of training the models. First, the architectures under consideration are introduced. Second, the optimizer settings and data augmentation techniques used to train the models are listed.

### 3.4.1 Models

We choose ResNet-50 as our baseline model and use the shallower variant, ResNet-18, for initial experiments, e.g., to search for data augmentations that

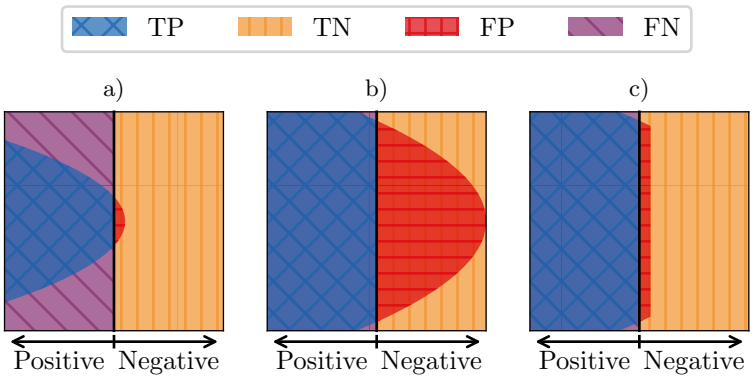



**Figure 3.4:** Example of a binary classifier of vectors in the 2-D plane, used to provide an intuitive understanding of the saved time metric. For a model to be useful, it must achieve both: (i) high sensitivity, and (ii) high saved time metric.

improve the model performance, as explained in Section 3.4.3. The ResNet-18 variant is used because its training is significantly faster, allowing for more experiments with the available hardware. The reason we choose ResNet-50 as our baseline is that (i) ResNet models are well-established CNNs (ii) are one of the industry's standard baselines, and (iii) larger variants, such as ResNet-152, are far too computationally expensive.

After deriving the data augmentations and the training protocol suitable for the dataset, we experiment with other model architectures and compare them against the baseline. We experiment with pre-trained state-of-the-art models of limited size. Namely, we train and evaluate

- ResNet-50x1-V2, [He et al., 2016b],
- EfficientNet models, [Tan and Le, 2019], from B0 to B4,
- EfficientNet-V2-S model, [Tan and Le, 2021],
- Vision Transformer ViT-B using 32×32 embeddings, [Dosovitskiy et al., 2021].

For links to the specific pre-trained weights, refer to Appendix A. We confine ourselves to these models for several reasons, (i) it is necessary to limit the search space, as not all possible models can be evaluated, (ii) the selected models are similar to the baseline in either the number of parameters or performance on large scale datasets, (iii) the models can be trained in reasonable time on the available hardware, and (iv) the inference time is sufficiently brief, as explained in Section 3.6.3.

### 3.4.2 Optimizer Settings

Unless explicitly specified otherwise, we train all models using stochastic gradient descent (SGD) with the initial learning rate (LR) of 0.001 and momentum 0.9. We decay the learning rate by a factor of 3 at 33% and 66%

of the 150 total training steps and use a batch size of 10 samples due to hardware limitations.

Our training protocol is heavily inspired by [Kolesnikov et al., 2020], Big Transfer (BiT), who empirically analyze the transfer of models pre-trained on large datasets to smaller downstream tasks. They conclude with a fine-tuning protocol recommended as a starting point for any general visual recognition task. Our protocol, however, does not follow the recommended protocol in exact, with the comparison shown in Table 3.2.

| Parameter | Protocol | |
|---|---|---|
| | BiT | Ours |
| Optimizer | SGD | SGD |
| Momentum | 0.9 | 0.9 |
| Initial LR | 0.003 | 0.001 |
| LR Decay Factor | 10 | 3 |
| LR Decay Steps | $30, 60, 90$ [%] | $33, 66$ [%] |
| Epochs | – | 150 |
| Batch Size | 512 | 10 |

**Table 3.2:** The standard training protocol used for experiments.

Namely, BiT uses a larger initial LR of 0.003 but decays the value more rapidly. Our incentive to decrease the initial LR is the batch size. [Kolesnikov et al., 2020] are able to train using a batch size of 512 samples, and the optimizer, therefore, attains descent direction with higher confidence; hence, the optimization step size can be longer.

For initial experiments, we utilize training, test, and validation splits of 60, 20, and 20% respectively, changing the partitioning to 80, 10, and 10% respectively for later experiments. Unless stated otherwise, experiments were conducted using the latter.

### 3.4.3 Data Augmentation

To enlarge the number of samples used for training, we utilize standard image data augmentations, namely the common augmentations (i) horizontal flip, (ii) vertical flip, (iii) rotation, and (iv) crop and resize. We train ResNet-18 model using the augmentations in isolation and evaluate the effects on the model performance. The procedure of deciding which augmentations to use is described in Section 4.1.2. We further experiment with additional, hand-designed augmentations, described in Section 3.5, and perform experiments with the augmentation in Section 4.3.

## 3.5 Domain-Specific Augmentation

In this section we propose novel data augmentation methods specific to the task of contaminant detection in microscopic images, where the location and

size of the contaminant within the image are not a priori known.

To generate additional positive samples, we locate the contaminant within the image either by (i) gradient-based localization, [Selvaraju et al., 2017], which is a broadly applicable method with minimal prerequisites, or (ii) by exploiting the fluorescent staining process. We then augment the image either by (i) inpainting the located contaminants into uncontaminated images, or (ii) blurring a portion of the contaminant. Motivation, as well as a detailed explanation of the augmentations, are given in Section 3.5.1.

We further generate more negative samples, which are purposefully difficult to classify in order to force the network to learn a more robust representation of the data and minimize the number of false positives. This is achieved by localizing structures in negative images that are visually similar to the contaminants, using gradient-based localization, [Selvaraju et al., 2017], and inpainting them into the image multiple times. The full description is provided in Section 3.5.2.

Experiments conducted with the augmentations are described in Section 4.3.

### 3.5.1 Positive Sample Augmentation

In this section, we describe two image augmentation methods devised to increase the number of positive sample images. First, the Poisson augmentation of positive samples, which inpaints the contaminant into negative backgrounds. Second, the blur augmentation of positive samples, which partially blurs the contaminant.

#### Poisson Augmentation of Fluorescent Stained Positive Samples

**Motivation.** While obtaining negative samples is simple, getting positive samples is comparatively complex and expensive. We propose an augmentation method specific to the detection task, where any negative image becomes positive if the contaminant is introduced. The technique takes advantage of a large number of negative images and uses them to generate synthetic positive images by inpainting the positive contaminant into a negative image, as shown in Figure 3.6. The contaminant can further be rotated and shifted to create practically an unlimited number of positive samples.

**Poisson Image Editing.** Consider the task of inserting a *source* image into a *target* image, assuming that the dimensions of the source are smaller than the target. The naive approach is to directly copy the pixel values in the color domain from the source to the target. Instead, Poisson image editing, [Pérez et al., 2003], attempts to perform the inpainting in the gradient domain. The inpainting is done by ignoring the exact pixel values of the source image and focusing on the image's gradient[1]. To insert the source image into the target image, (i) the border pixels of the source image are set to match the color of neighboring pixels in the target image, and (ii) the remaining pixel values of the inpainted source image are found by solving the Poisson equation

[1]The gradient is frequently extracted only along the vertical and horizontal axes.

with the condition of the original gradient preserved. I.e., the colors of the source image are modified to match the colors of the target image without discarding details. The task is often over-constrained and has to be solved approximately.

**Description of the proposed method.** The procedure requires (i) a positive image, *source*, (ii) a localization *mask*, and (iii) a negative image, *destination*. We use the mask to select pixels from the positive image and inpaint them into the negative image. The naive procedure of directly copying the pixel values creates visible edges between the contaminant and the background, see Figure 3.7a. To generate realistic-looking synthetic images, we instead perform the inpainting using Poisson image editing, [Pérez et al., 2003]. A comparison of the two inpainting variants is shown in Figure 3.7.

For the inpainting, it is first necessary to locate the contaminant. We exploit the fact that due to the fluorescent staining, the pixel values of the contaminant are always greater than those of the background. Therefore, we can obtain a rough localization mask by thresholding the pixel values, shown in Figure 3.5.

It must be noted that the fluorescent dye is non-specific, however, and other structures, such as dust, pollen, or arthropods, may also bind the dye. We, therefore, guarantee that the method inpaints the contaminant of interest, but inadvertently also other miscellaneous structures.

The method can be further expanded by detecting connected components of the mask and inpainting each component separately with random rotation and random position in the target image. This modification is especially suited for yeast, where the number of connected components in the localization mask is high. An example of two different possible results obtained by augmenting an image of yeast is shown in Figure 3.8.

The following steps summarize the entire augmentation:

1. Find parts of a positive image suspected of containing the contaminant.
   - Localization with fluorescent staining can be accomplished by thresholding the image, e.g., by employing Yen's thresholding algorithm, [Yen et al., 1995], as the contaminant will always be brighter than the background.
   - Localization can be performed for a general detection task by selecting regions of the image that result in the highest activation of the model, as with Grad-CAM, [Selvaraju et al., 2017], or Captum, [Kokhlikyan et al., 2020].
2. Select a random negative image.
3. Insert each part discovered in step 1 into the negative image. The position and rotation of the part are selected randomly.
   - Naive approach is to copy the pixel values directly.

- The novel approach utilizes Poisson image editing for the inpainting, ensuring that the part is inserted seamlessly.

**Figure 3.5:** Example of thresholding a positive sample to obtain the binary mask, which is then used to extract the positive cutout for inpainting.

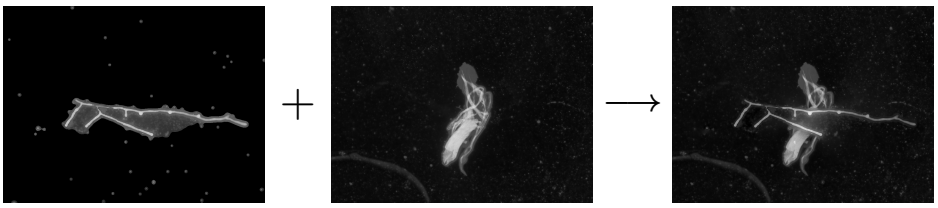
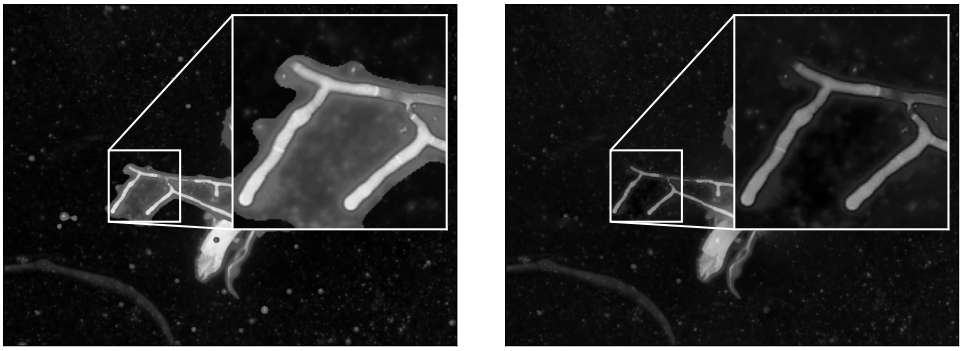


**Figure 3.6:** Example of inpainting a positive cutout into a negative sample using Poisson image editing. The change in color of the cutout within the result should be noted, creating a seamless blend.



**(a) :** Naive inpainting. **(b) :** Poisson inpainting.

**Figure 3.7:** Comparison of the inpainting methods. The inpainted region has a distinct background color when the value of pixels is copied directly. The Poisson image editing ensures that the cutout seamlessly blends into the negative image, creating a more realistic result.

### Blur Augmentation of Positive Samples with Localization Map

**Motivation.** CNNs trained for computer vision tasks learn to detect easily recognizable features, textures, and patterns, focusing on the most discriminative input regions, often resulting in low model robustness. We propose an augmentation method in which the most discriminative input regions are

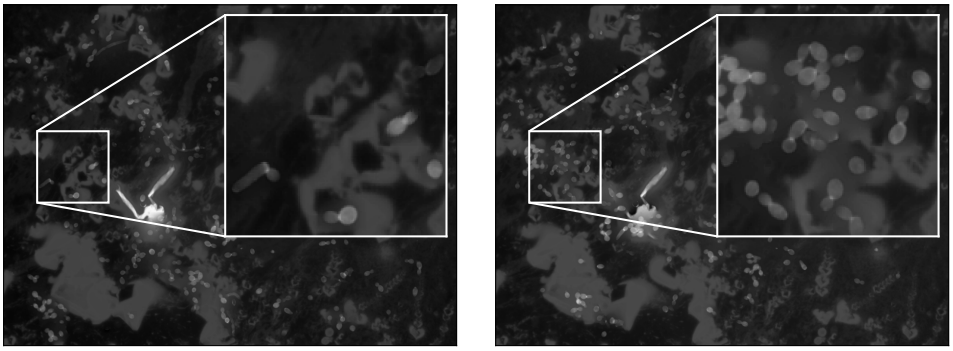


**Figure 3.8:** Two distinct outcomes of inpainting yeast into a negative image. Each image is noticeably different due to each connected component within the mask of the positive image being inpainted separately.

partially blurred, forcing the model to recognize less discriminative but still important features. Similar methods had been previously proposed to address this issue, occluding the input images, with mixed results, [Wei et al., 2017], [Fong and Vedaldi, 2019].

**Description.** The procedure requires (i) a positive image, *source*, and (ii) a localization *mask*, identifying regions which attribute for the classification result. We exploit gradient-based localization, [Selvaraju et al., 2017], to discover which regions of the image contribute the most to positive classification of the image. This results in a course heatmap that we threshold to obtain a binary mask. For each connected component in the mask, we create the minimal bounding box.

Instead of blurring the entirety of the bounding boxes, we only blur a portion - to not cover the whole contaminant. Specifically, for each bounding box, we create a random rectangle that covers a desired percentage of the bounding box and only blur pixels inside the rectangle.

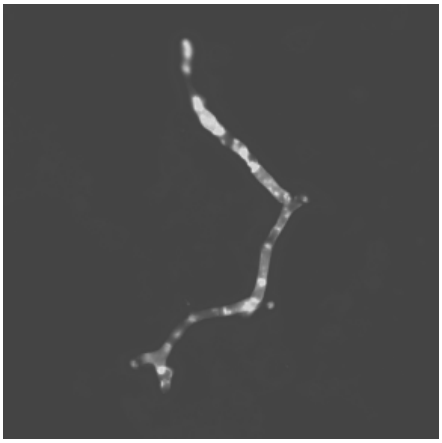
We further ensure that the transition between the blurred region and the original image is smooth. To achieve this, we create the final image as an affine combination of the blurred region and the original image,

$$\text{result} = \alpha \cdot \text{image}_{\text{blurred}} + (1 - \alpha) \cdot \text{image}_{\text{original}}, \quad \alpha \in [0, 1]. \tag{3.26}$$

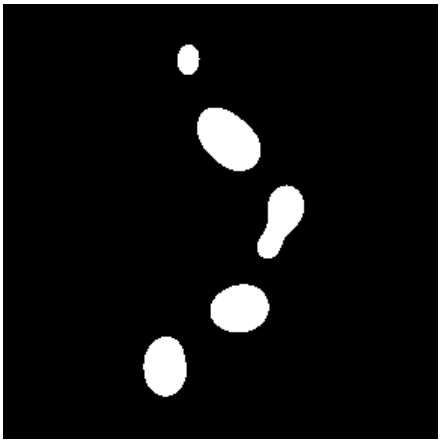
We compute the coefficient $\alpha$ for each pixel separately by applying Gaussian blur to the binary mask, interpreted as an array of real numbers, and normalizing the result. Therefore, the value of $\alpha$ is 0 for pixels distant from the blurred region and increases as pixels get closer to the region. The result of the blur augmentation is shown in Figure 3.9.

### 3.5.2 Negative Sample Augmentation

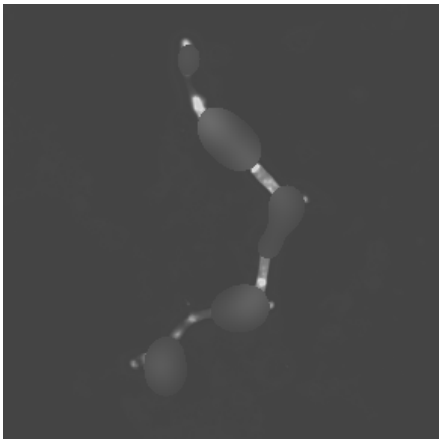
In this section, we describe an image augmentation method devised to increase the number of negative sample images.



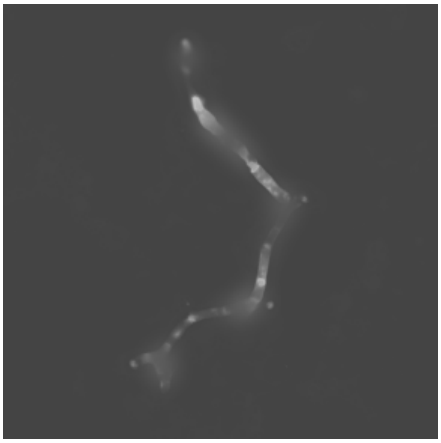
**(a) :** Original image.



**(b) :** Binary mask indicating regions of the original image to be blurred.



**(c) :** Result of simple blur augmentation. Noticeable contour is present at the border of the blurred region.



**(d) :** Blur augmentation result with smooth transition between the blurred region and the rest of the image.

**Figure 3.9:** Example of the blur augmentation outcome. The augmentation is intended to make positive images artificially difficult to correctly classify, forcing the model to recognize less discriminative but important contaminant features.

#### Poisson Augmentation of Negative Samples with Localization Map

**Motivation.** As previously stated, obtaining negative samples is relatively simple, as a samples can be taken from healthy individuals on mass without the need for manual examination.

The provided dataset, however, contains specifically chosen negative images with structures that are visually similar to filamentous fungi or yeast, such as cloth threads, see Figure 3.10. The fact that the structures are easily mistaken for fungi is demonstrated further by the results of a survey given to people with no background experience in microbiology, see Figure 4.26. By replicating the structures that are visually similar to fungi, negative images can be made even more similar to positive images.

The motivation for this is that the models will need to learn non-trivial features that will allow them to distinguish between structures similar to the contaminant and the actual contaminant.

**Description.** The procedure requires (i) a negative image, and (ii) a localization *mask*, identifying regions which contain structures visually similar to the contaminant. We exploit gradient-based localization, [Selvaraju et al., 2017], to discover the regions mentioned above. As a result, we obtain a course heatmap from which we generate a binary mask by thresholding. It must be mentioned that the heatmap produced by Grad-CAM, [Selvaraju et al., 2017], specifies the relative magnitude of activations. If there are no structures in the image that are similar to the contaminant, the activations across the entire image are of similar magnitude, and the resulting mask covers the entire image. The augmentation must, therefore, not be used for such cases.

The augmentation procedure itself is similar to the Poisson augmentation described in Section 3.5.1, with the exception that we do not inpaint a positive cutout into a negative image here. Instead, (i) the source, and (ii) the target are both derived from the same negative image. The augmentation can also be performed multiple times, repeatedly inpainting the same cutout. An example of the augmentation result is shown in Figure 3.11.

## 3.6 Evaluation Protocol

### 3.6.1 K-Fold Construction

We train using 30-fold cross-validation with training, validation, and test sets containing 80%, 10%, and 10% of the total data set, respectively. Figure 3.12 shows an example of how the folds are created. The diagram depicts the construction for the number of bins $N = 4$. However, we construct the folds with $N = 10$ so that each bin contains 10% of the dataset. The bins are balanced, i.e., each bin contains an equal number of positive and negative samples.

As a result, we generate a total of 90 folds. For the experiments, we choose 30 folds so that all images are in the training, test, or validation split precisely

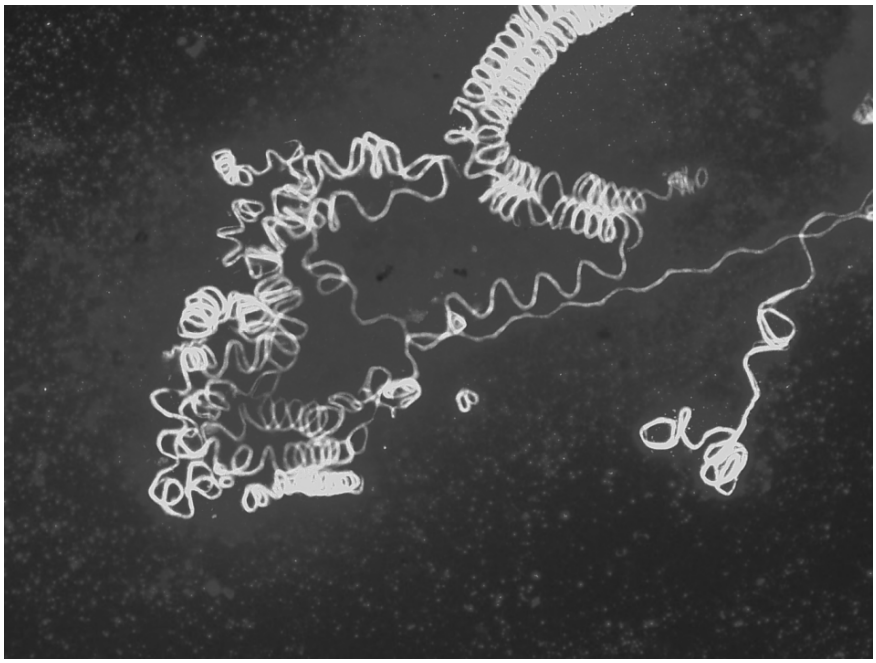
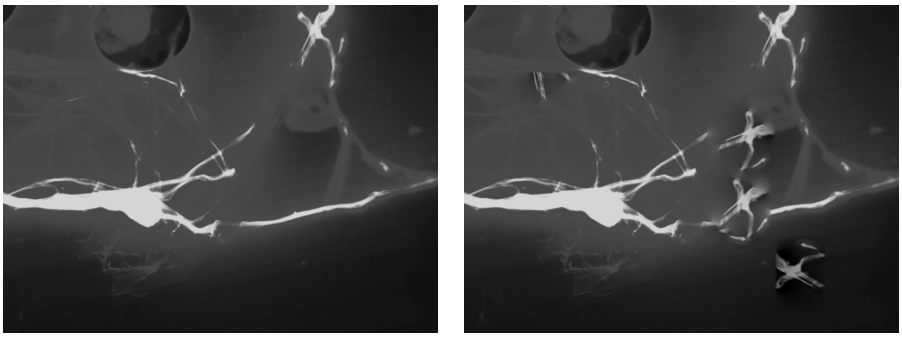


**Figure 3.10:** Negative image that contains structures similar to filamentous fungi.



**(a) :** Original image. **(b) :** Augmented image.

**Figure 3.11:** Example of the Poisson augmentation of negative images. Notice the cross-like structure in the right half of the image copied multiple times.

three times. The model is trained using the training split. After each epoch, the performance is measured on the validation split and the results are used to select the best epoch weights. The final reported performance of a model is its performance on the test set.

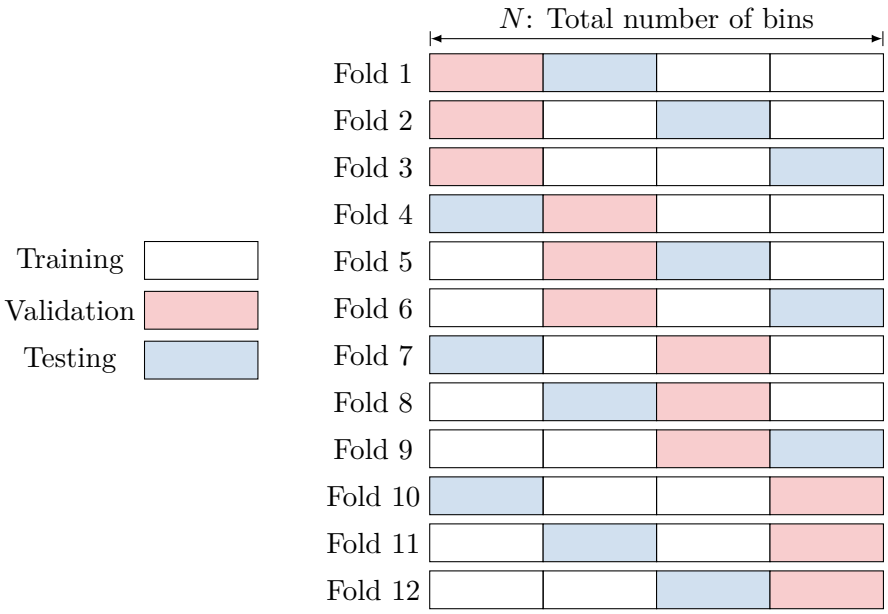



**Figure 3.12:** Construction of the folds used for cross-validation. The total number of folds which can be constructed is $N \cdot (N-1)$, where $N$ is the number of bins. The example is shown for each bin containing $\frac{100}{N} = 25\%$ of the dataset.

### 3.6.2 Saved Time Metric

We evaluate the saved time metric for sensitivities of 98%, 99%, and 99.5% as the criterion for determining the best model. The high values of sensitivity were directly requested by the staff of Motol University Hospital. Models with significantly lower sensitivity would be of no practical use. The value of the metric for sensitivity approaching 100% is not used because the metric is volatile for very high sensitivities and does not serve as a good measure of model performance, see Section 4.5.

#### Saved Time Metric over Multiple Folds

It should be noted that when evaluating the saved time metric for multiple folds, there are two ways to report the average performance.

Variant A is to compute $\overline{\text{FPR}}$ which is the average FPR for a fixed TPR over all folds, then compute the saved time using the TPR and $\overline{\text{FPR}}$. This approach is equivalent to computing the average saved time over all folds for a fixed TPR.

Variant B is to compute $\overline{\text{TPR}}$ which is the average TPR for a fixed FPR over all folds, then compute the saved time using the FPR and $\overline{\text{TPR}}$.

Formally, given a classifier $f_k$ trained on the $k$-th fold and a corresponding test set $\mathrm{T}_k = \{(x_{k,i}, y_{k,i}) \in \mathcal{X} \times \{+1, -1\} \mid i = 1, \ldots, n\}$, we can measure performance of the model in isolation by computing the empirical estimates of $\mathrm{TPR}_k(\theta)$ and $\mathrm{FPR}_k(\theta)$ on the test set as

$$\widehat{\mathrm{TPR}}_k(\theta) = \frac{1}{n_+} \sum_{i=1}^{n} [\![ h(x_{k,i}; \theta) = +1 \wedge y_{k,i} = +1 ]\!], \tag{3.27}$$

$$\widehat{\mathrm{FPR}}_k(\theta) = \frac{1}{n_-} \sum_{i=1}^{n} [\![ h(x_{k,i}; \theta) = +1 \wedge y_{k,i} = -1 ]\!], \tag{3.28}$$

where $n_+ = \sum_{i=1}^{n} [\![ y_{k,i} = +1 ]\!]$ and $n_- = \sum_{i=1}^{n} [\![ y_{k,i} = -1 ]\!]$. Then, we fix the positive class prior to the expert estimate of the incidence rate and compute the empirical estimate of the saved time

$$\begin{aligned} \widehat{\mathrm{ST}}_k(\theta) = &\left[1 - p(y = +1)\right] \cdot \left[1 - \widehat{\mathrm{FPR}}_k(\theta)\right] \\ &+ p(y = +1) \cdot \left[1 - \widehat{\mathrm{TPR}}_k(\theta)\right] . \end{aligned} \tag{3.29}$$

A curve $\{(\widehat{\mathrm{TPR}}_k(\theta), \widehat{\mathrm{ST}}_k(\theta)) \mid \theta \in (-\infty, \infty)\}$ can then be constructed which summarizes the entire space of achievable true positive rates and saved times for a single classifier. To measure the average performance of multiple classifiers, we can choose from the following approaches.

**Variant A.** The curve $\{(\widehat{\mathrm{TPR}}_k(\theta), \widehat{\mathrm{ST}}_k(\theta)) \mid \theta \in (-\infty, \infty)\}$ can with a slight abuse of notation be equivalently expressed as $\{(\mathrm{TPR}, \widehat{\mathrm{ST}}_k(\mathrm{TPR})) \mid \mathrm{TPR} \in (0, 1)\}$, where $\widehat{\mathrm{ST}}_k(\mathrm{TPR}) = \widehat{\mathrm{ST}}_k(\theta)$ with $\theta_k \in \mathbb{R}$ such that $\widehat{\mathrm{TPR}}_k(\theta_k) = \mathrm{TPR}$.

Performance of multiple models evaluated across different test folds can then be computed as the average saved time for a fixed TPR, i.e., the curve

$$\left\{(\mathrm{TPR}, \frac{\sum_{k=1}^{l} \widehat{\mathrm{ST}}_k(\mathrm{TPR})}{l}) \mid \mathrm{TPR} \in (0, 1)\right\}, \tag{3.30}$$

where $l$ is the total number of folds. An equivalent approach is to create the curve $\{(\mathrm{TPR}, \overline{\mathrm{FPR}}(\mathrm{TPR})) \mid \mathrm{TPR} \in (0, 1)\}$, where $\overline{\mathrm{FPR}} = \frac{\sum_{k=1}^{l} \widehat{\mathrm{FPR}}_k(\theta_k)}{l}$, and $\theta_k$ is chosen such that $\widehat{\mathrm{TPR}}_k(\theta_k) = \mathrm{TPR}$. The average saved time $\overline{\mathrm{ST}}$ can then be obtained as

$$\begin{aligned} \overline{\mathrm{ST}}(\mathrm{TPR}) = &\left[1 - p(y = +1)\right] \cdot \left[1 - \overline{\mathrm{FPR}}(\mathrm{TPR})\right] \\ &+ p(y = +1) \cdot \left[1 - \mathrm{TPR}\right] . \end{aligned} \tag{3.31}$$

**Variant B.** An alternative to the previous is to first create the curve $\{(\overline{\mathrm{TPR}}(\mathrm{FPR}), \mathrm{FPR}) \mid \mathrm{FPR} \in (0, 1)\}$, where $\overline{\mathrm{TPR}} = \frac{\sum_{k=1}^{l} \widehat{\mathrm{TPR}}_k(\theta_k)}{l}$, and $\theta_k$ is chosen such that $\widehat{\mathrm{FPR}}_k(\theta_k) = \mathrm{FPR}$. The average saved time $\overline{\mathrm{ST}}$ can then be obtained as

$$\begin{aligned} \overline{\mathrm{ST}}(\mathrm{FPR}) = &\left[1 - p(y = +1)\right] \cdot \left[1 - \mathrm{FPR}\right] \\ &+ p(y = +1) \cdot \left[1 - \overline{\mathrm{TPR}}(\mathrm{FPR})\right] . \end{aligned} \tag{3.32}$$

Result of the two variants is shown in Figure 3.13. Unless explicitly stated otherwise, we report the average performance of models using variant B, as the curve of the variant appears to provide more detail near $\mathrm{TPR} = 100\%$.

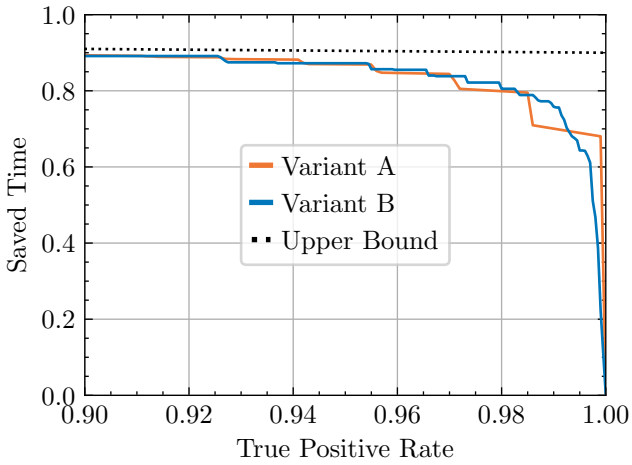



**Figure 3.13:** Two possible variants of reporting the average saved time metric over folds. Unless explicitly stated otherwise, we report the average performance of models using Variant B, as the curve of the variant appears to provide more detail as the true positive rate approaches 100%.

### 3.6.3 Inference Time

As stated in Section 2.2, automation of clinical sample diagnosis requires automated slide imaging, which results in a large number, typically thousands, of images being taken for each clinical sample. These images can be composed to form a single composite image of the slide; however, processing the partial images separately is more computationally efficient as they can be evaluated before the scanning process finishes.

Due to the large number of images, it is imperative that the evaluation is fast, ideally without any specialized hardware. We, therefore, limit the model selection. For all trained models, we evaluate the inference time, measuring only the forward pass of the models. Thus we do not consider the time spent reading and transferring the data to the processing device.

We (i) force synchronous execution of CPU and GPU, (ii) warm-up the GPU by artificial computations, and (iii) measure forward pass time by `torch.cuda.Event` timers. We evaluate the inference time on a single NVIDIA GeForce GTX 1080 Ti GPU, with the results shown in Section 4.6.

# Chapter 4

# Experiments & Results

This chapter describes the experiments that were conducted, how they were set up, and what the outcomes were. Section 4.1 describes experiments that were carried out in order to develop a strong baseline model and explore the dataset using standard machine learning methods. Section 4.2 describes experiments that attempt to improve on the baseline by (i) examining alternative deep learning models, and (ii) assembling the models into an ensemble. Section 4.3 evaluates the effects of the augmentations proposed in Section 3.5. In Section 4.4, we construct a learning curve and evaluate whether collecting more data would be beneficial. Section 4.5 examines how the achieved value of the saved time metric changes when the dataset is slightly changed. In Section 4.6 the inference time of the models is measured. Section 4.7 compares human performance against that of the models.

## 4.1 Baseline Development

This section describes experiments carried out to create a robust baseline model using standard machine learning techniques. We experiment with the ResNet family of models, assuming that results obtained for one ResNet model transfer to other ResNet models; for example, if data augmentation helps ResNet-18, it also helps ResNet-50 and ResNet-101. The result of the experiments in this section is a tuned ResNet-50 model.

### 4.1.1 Image Scale

**Motivation & Goal.** The provided training dataset contains images of multiple pixel dimensions, as mentioned in Section 3.1. The field of view captured by the images is identical. However, the pixels correspond to a different actual size if the shapes of the images differ, i.e., pixels in larger images capture smaller regions of the real world than pixels in smaller images. It is beneficial for the models if the typical size of the contaminants is constant, allowing the models to learn filters of a fixed size. Within the data, however, some images may contain yeast cells spanning 10 pixels, and other images may contain yeast cells spanning 40 pixels. To address this issue, we downscale all

images in the dataset to the size of the smallest image in the data, $1360 \times 1024$ pixels, resampling using pixel area relation.

Training models on images with low scale, see Section 3.1, is computationally less demanding, while higher scales preserve more detail. The images, however, are not precisely in focus and contain artifacts, as shown in Figure 3.1. This begs the question, on what scale should the model be trained?

An experiment was thus conducted to compare model performance based on the image scale and select the optimal scale for further experiments.

**Setup.** ResNet-18 models were trained with images of scale $s \in \{10\%, 20\%, \ldots, 100\%\}$, where 100% corresponds to the base shape $1360 \times 1024$, using 20-fold cross-validation. The training, test, and validation splits were 60, 20, and 20%, respectively. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 0.5 \cdot 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.1$ every 50 epochs,
- total training epochs: 200,
- batch size: $b = 2$,
- loss function: binary cross entropy.

The resulting models were then evaluated on the saved time metric.

**Result.** The dependence of the saved time metric on the image scale is shown for sensitivities of 98%, 99%, and 99.5% in Figure 4.1. It can be observed that larger image scales result in better performance with diminishing returns when using a scale larger than 50%, i.e., increasing the image scale further yields marginal improvements to the metric.

The same observation can be made for Figure 4.2, which shows the curve of all achievable values of the saved time metric $\{(\mathrm{TPR}, \overline{\mathrm{ST}}(\mathrm{TPR})) \mid \mathrm{TPR} \in (0, 1)\}$ for all image scales.

The scale of 70% was chosen for further experiments due to (i) achieving values of the saved time metric comparable with all larger scales, and (ii) posing lower computational costs than larger scales. Any scale over 50% would, however, be a valid choice.

### 4.1.2 Effects of Data Augmentation

**Motivation & Goal.** When presented with limited dataset size, it is customary to utilize augmentations that generate new training samples from the available data, i.e., modify the available images such that a new sample looks like it was drawn from the underlying data distribution. If done correctly, this allows the models to more precisely fit the underlying data distribution, prevent overfitting, and achieve better performance.

Many augmentations are domain-specific, e.g., image rotation. However, whether a specific augmentation is beneficial depends both on the task and the provided dataset and must be verified experimentally. We, therefore, evaluate the effect of multiple data augmentation techniques commonly used

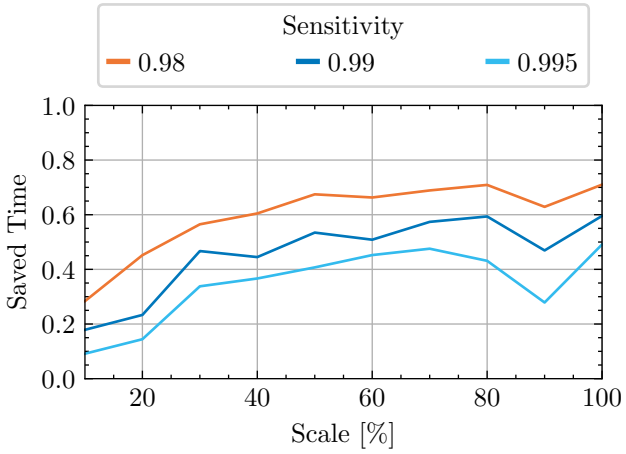



**Figure 4.1:** Dependence of the saved time metric on image scale. Higher image scales result in a higher value of the saved time metric with diminishing returns over the scale of 50%, i.e., increasing the image scale further yields marginal improvements to the metric. An unexpected drop in the metric can be observed on scale of 90%.

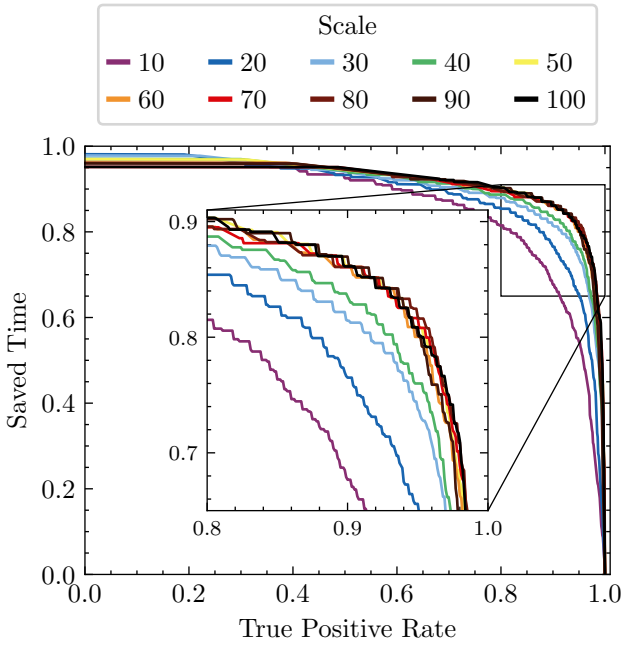



**Figure 4.2:** Dependence of the saved time metric on image scale. The curves for scale over 50% overlap, suggesting that further increase in the image scale yields only marginal or no improvements.

for images on the saved time metric, to select suitable augmentations for further experiments.

**Setup.** Standard image augmentation techniques of (i) horizontal mirroring, (ii) vertical mirroring, (iii) rotation, (iv) crop and resizing, and (v) Gaussian blur, were experimented with. The horizontal mirroring augmentation was used in all of the experiments. ResNet-18 models were trained for 90-fold cross-validation with the training, test, and validation splits of 60, 20, and 20% respectively, on a dataset of image scale 20%. The SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The resulting models were then evaluated on the saved time metric for sensitivities of 98%, 99%, and 99.5%.

**Result.** The (i) rotation augmentation, and (ii) crop & resizing improved the model performance. The Gaussian blur augmentation, on the other hand, hindered the performance. We therefore set up an experiment where models were trained with all of the beneficial augmentations, namely (i) horizontal mirroring, (ii) rotation, (iii) cropping a portion of the image and resizing the crop to the original image size, and (iv) additionally also vertical mirroring. This combination of augmentations resulted in the best model performance.

As a high variance of the saved time metric was observed between the 90 folds, it was decided to change the training, test, and validation splits for further experiments to 80, 10, 10% from 60, 20, 20%.

Performance of the models depending on the augmentations, measured by the saved time metric, is shown in Table 4.1. The results were later verified for ResNet-50 and larger image scales on a 5-fold experiment. It was decided to use ResNet-50 models for future experiments, as the larger model exhibited improved performance.

### 4.1.3 Transfer Learning

**Motivation & Goal.** The industry-standard approach of training a model is to take a pre-trained model and fine-tune the weights on a downstream task. This procedure, called transfer learning, preserves the general filters learned on large datasets, matches the filters to the specific task, and slightly modifies the filters to fit the downstream dataset better.

Transfer learning allows for training on much smaller dataset sizes without overfitting the data. However, for specific tasks, especially if the amount of provided data is sufficiently large, training end-to-end from a random initialization yields better results.

| Augmentation | Sensitivity 98% | 99% | 99.5% |
|---|---|---|---|
| Horizontal mirroring | 0.632 (0.181) | 0.453 (0.207) | 0.359 (0.256) |
| + Rotation | 0.655 (0.141) | 0.500 (0.199) | 0.438 (0.249) |
| + Crop & resize | 0.656 (0.136) | 0.495 (**0.175**) | **0.463** (**0.204**) |
| + Blur | 0.551 (0.189) | 0.383 (0.218) | 0.307 (0.252) |
| + Rot. + Crop + Vert. m. | **0.670** (**0.134**) | **0.511** (0.198) | 0.458 (0.245) |

**Table 4.1:** Saved time metric dependence on training data augmentation. Results shown for image scale of 20% and ResNet-18 models. The first value indicates the mean saved time metric; the second value is the standard deviation between folds. The best discovered combination of data augmentations was found to be i) horizontal mirroring, ii) vertical mirroring, iii) rotation, iv) cropping a portion of the image and resizing the crop to the original image size.

Therefore, the goal of the experiment was to evaluate whether the use of transfer learning results in improved performance.

**Setup.** ResNet-18 and ResNet-50 models were trained with either (i) initializing the weights randomly, or (ii) fine-tuning weights obtained by training on a large dataset. The models were trained using 5-fold cross-validation with the training, test, and validation splits of 80, 10, and 10%, respectively, and the image scale of 70%. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 200,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The training and validation losses of the models were observed, and the resulting models were evaluated on the saved time metric.

**Result.** Unsurprisingly, models trained from scratch exhibited slower learning, i.e., the training loss decreased significantly slower. Further, the training loss of multiple models trained from a random initialization failed to converge by the last training epoch. The models trained from random initialization also exhibited poor generalization, reaching training loss values an order of magnitude lower than the validation loss.

The same conclusion was reached by comparing the saved time metric values, where models trained by fine-tuning pre-trained weights achieved significantly better results.

### 4.1.4 Freezing Layers

**Motivation & Goal.** As mentioned in Section 4.1.3, transfer learning is the industry-standard approach of training computer vision models. However, the

debate continues over whether (i) the general filters learned on large datasets should remain fixed and only be matched to the specific task, i.e., only a subset of the weights (typically of the classifier head) is fine-tuned, and the remaining weights are *frozen*, or (ii) the filter weights should also be modified, i.e., all weights are fine-tuned.

To resolve this question, an experiment was set up where models were trained using both strategies mentioned above.

**Setup.** ResNet-50 models were trained using 5-fold cross-validation with the training, test, and validation splits of 80, 10, and 10%, respectively. The folds were trained for an image scale of 70%.

The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 200,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The resulting models were evaluated on the saved time metric.

**Result.** Models where all of the layers were fine-tuned exhibited significantly higher values of the saved time metric than models where only the classifier head was trained on the downstream task. The improved performance was also observed on (i) the training loss, and (ii) the validation loss.

### 4.1.5 Image Color Space

**Motivation & Goal.** Some digital microscopy imaging techniques are capable of capturing the true colors of magnified objects, while others are not. The obtained dataset was encoded as RGB images. However, the color was added artificially in the imaging software using an unknown colormap. It was theorized that the trained models would be better applicable to other microscopy hardware and other preparation methods (e.g., different fluorescent dye) if they were trained using grayscale images. As the color was added artificially, this procedure should lead to no loss of information if done correctly. Further, the proposed augmentations are better suited for the grayscale color space, as Poisson image editing may introduce out of distribution colors to an RGB image, i.e., the inpainted region may contain colors that could not be created with standard fluorescent miscroscopy.

Conversion of colored images to grayscale can be done in many ways. Human vision is more sensitive to some light wavelengths than to others. It is also excellent at detecting minor changes in color at low luminance but poor at detecting changes at high luminance. The most common procedure of converting colored images to grayscale consists of (i) taking the weighted average of the color channels such that the different perception of colors is accounted for, [Stokes and Anderson, 1996]. This procedure can further be improved by a (ii) non-linear transformation of the color domain, which

ensures that more colors are concentrated at the lower end of the range, where differentiating between colors is easier. Another approach is to use a (iii) linear approximation of the non-linear transformation, see [ITU, 2011].

It remained to be experimentally verified how the conversions above would affect the model performance.

**Setup.** The conversion variants yielded, for all practical purposes, indistinguishable performance on the saved time metric for a small, 3-fold experiment with ResNet-50 models. Full experiment was therefore set up, using the conversion variant (iii)[1], where ResNet-50 models were trained using 30-fold cross-validation with the training, test, and validation splits of 80, 10, and 10%, respectively, and the image scale of 70%.

The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,
- batch size: $b = 10$,
- loss function: binary cross entropy.

For preprocessing of the RGB data, ImageNet normalization parameters[2] were used, as the pre-trained model weights were learned using these parameters. The grayscale normalization parameters[3] were obtained from ImageNet images converted to grayscale using the conversion (iii). For the models which process grayscale images, the first convolutional layer (projecting from the three RGB channels into a higher dimension) was replaced by a layer projecting from one channel. The initial weights of the layer were obtained as a sum over the weights of the RGB variant. The resulting models were evaluated on the saved time metric.

**Result.** Models trained on grayscale images achieved performance similar to models trained on RGB images. The saved time metric for the two variants is shown in Table 4.2 and Figure 4.3. This suggests that the loss of information caused by converting the images to a single channel is insignificant. It was decided to use the grayscale images for future experiments, even though the performance when using grayscale images is slightly hindered. As a result, should a dataset be collected using an automated slide scanner in the future, the models produced in this thesis can be directly evaluated on the new data without repeating the training.

[1] We use specifically the luma formula defined by ITU-R 601-2, taking the weighted average of the RGB channels as $L = R \cdot 299/1000 + G \cdot 587/1000 + B \cdot 114/1000$. This is a standard approach used throughout the industry, e.g. by MATLAB, Pillow, and OpenCV.

[2] RGB normalization uses the parameters $mean = \begin{bmatrix}0.485 & 0.456 & 0.406\end{bmatrix}$) and $std = \begin{bmatrix}0.229 & 0.224 & 0.225\end{bmatrix}$ for the three respective color channels.

[3] Grayscale normalization uses the parameters $mean = 0.44531$ and $std = 0.26924$.

| Color | Sensitivity 98% | 99% | 99.5% |
|---|---|---|---|
| Grayscale | 0.805 (**0.116**) | **0.758** (0.180) | 0.643 (0.201) |
| RGB | **0.820** (0.118) | **0.758** (**0.156**) | **0.666** (**0.192**) |

**Table 4.2:** Saved time metric comparison for ResNet-50 models trained on grayscale images and RGB images. The first value indicates the mean saved time metric; the second value is the standard deviation between folds. The models achieve slightly better performance when using RGB images.

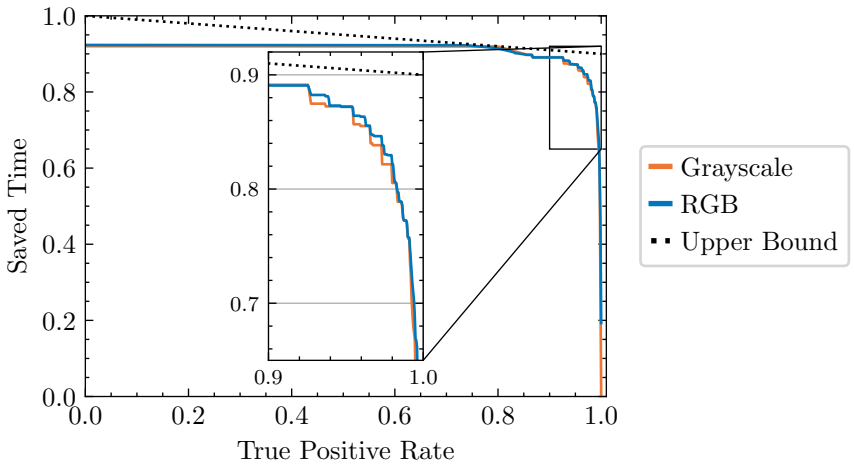



**Figure 4.3:** Saved time metric comparison for ResNet-50 trained on grayscale images and RGB images. The curves for both color domains are nearly identical, suggesting that the conversion to grayscale results in insignificant or no loss of information while allowing the models to better transfer to different microscopy or slide preparation methods.

### 4.1.6 Performance of the Baseline Model

This section provides a concise assessment of the baseline model performance and the experimentally discovered optimal training configuration. The curve of all achievable values of saved time is shown in Figure 4.4, with a close-up of the region of interest shown in Figure 4.5. Alternatively, the ROC curve is displayed in Figure 4.6.

The model achieves a saved time metric value of 0.758 for a true positive rate of 99%, i.e., if deployed, the model could be expected to reduce the number of manual examinations by 75.8%. This assumes that the model's performance would transfer to images obtained from an automated slide scanner. The findings suggest that an automated classifier can be developed and successfully used as a filter of samples to be presented to human workers.

The models were trained on grayscale images of scale 70%, i.e., resolution $952 \times 716$ with data augmentations of (i) horizontal mirroring, (ii) vertical mirroring, (iii) rotation, and (iv) cropping a portion of the image and resizing the crop to the original image size. The models were initialized to weights

pre-trained on ImageNet, see links in Appendix A.

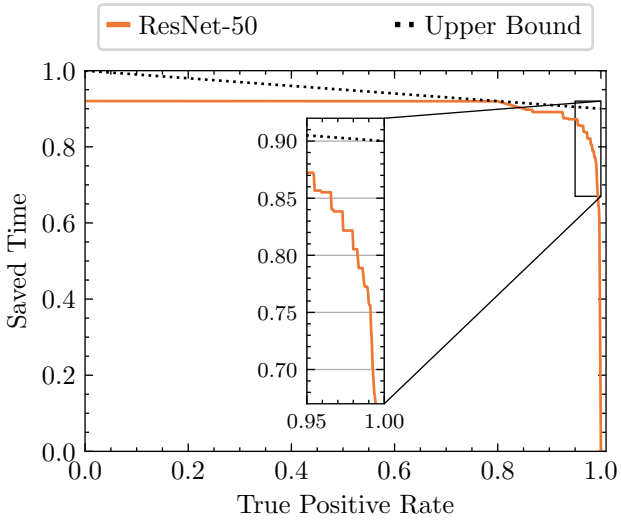



**Figure 4.4:** Saved time metric for baseline ResNet-50 models trained on 30-fold grayscale images. The model achieves a value of 0.758 for a true positive rate of 99%, i.e., if deployed, the model could be expected to reduce the number of manual examinations by 75.8%. This assumes that the model's performance would transfer to images obtained from an automated slide scanner. The findings suggest that an automated classifier can be developed and successfully used as a filter of samples to be presented to human workers. A Close-up of the region of interest is shown in Figure 4.5.

## 4.2 Model Architecture Search

This section describes experiments carried out to discover a better classifier model. We first experiment with models other than the ResNet family in Section 4.2.1. We then evaluate whether the performance can be improved by organizing the models into an ensemble in Section 4.2.2.

### 4.2.1 Alternative Architectures

**Motivation & Goal.** Although CNNs have dominated computer vision tasks for the past decade, alternative approaches such as vision transformers have recently gained traction. Furthermore, many different CNN architectures have been developed, with tens of different model architectures commonly being used.

We select and train multiple state-of-the-art models and compare them against the baseline ResNet-50 to determine which model performs best at detecting fungi and yeast in clinical samples.

The models are frequently published as a family of models with varying hardware requirements and performance. The new state-of-the-art performance is usually only achieved after the model and the dataset have been

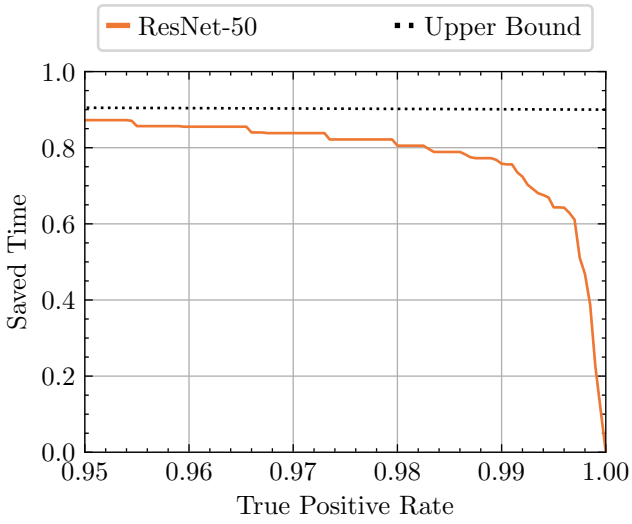



**Figure 4.5:** Saved time metric for ResNet-50 models trained on 30-fold grayscale images shown for high values of true positive rate. The model achieves values of the saved time metric over 75% for the necessary sensitivity of 99%+. The findings suggest that an automated classifier can be developed and successfully used as a filter of samples to be presented to human workers.

scaled up to the point where training the model becomes far too computationally expensive for both individuals and academia, e.g., new state-of-the-art methods are typically trained on an internal Google dataset JFT-3B containing nearly three billion images. Therefore, we select the variant from each family of models that is computationally reasonable and achieves performance similar to the baseline ResNet-50 on the ImageNet benchmark, e.g., from the Vision Transformers, we select ViT-Base instead of ViT-Huge.

With this selection, should any model perform significantly better than the baseline, the improvement can be assumed to be caused by the superiority of the model architecture. Otherwise, the new model might achieve better performance only due to its increased complexity and expressive power.

**Setup.** The models

- ResNet-50x1-V2, [He et al., 2016b],
- EfficientNet B0, B1, ..., B4, [Tan and Le, 2019],
- EfficientNet-V2-S, [Tan and Le, 2021],
- MLP-Mixer-B-32, [Tolstikhin et al., 2021],
- ViT-B-32, [Dosovitskiy et al., 2021],

were trained using 30-fold cross-validation with the training, test, and validation splits of 80, 10, and 10%, respectively, and using an image scale of 70%. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,

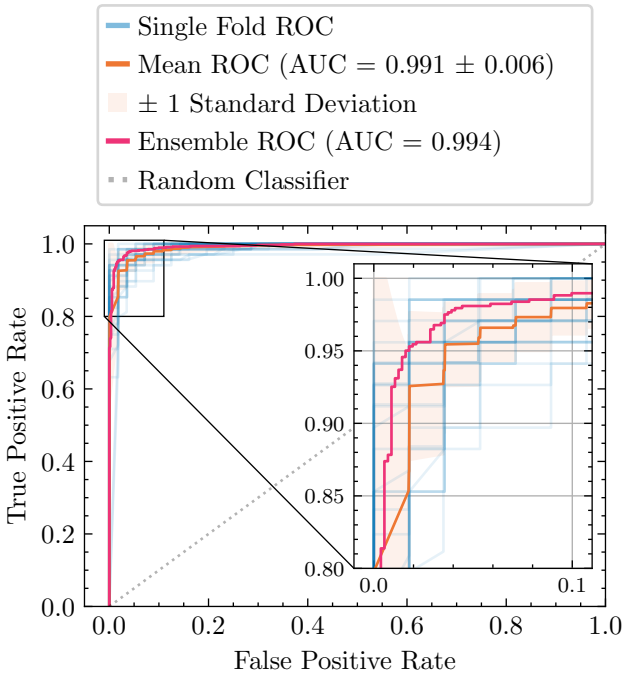



**Figure 4.6:** ROC curve (receiver operating characteristic) for ResNet-50 models trained on 30-fold grayscale images. The **orange** curve is the mean ROC over all folds, with the light orange area marking the standard deviation between folds. The **pink** curve was constructed for an ensemble of the trained baseline models. An ensemble was produced for each image in the dataset separately. Each image in the dataset was present in the test set of three folds; therefore, the ensemble for each image always contained three ResNet-50 models. For the definition of the ensemble, refer to Section 3.2.

- batch size: $b = 10$,
- loss function: binary cross entropy.

The specific pre-trained weights which were used can be found in Appendix A. The resulting models were then evaluated on the saved time metric.

**Result.** Multiple models achieve higher values of the saved time metric than the baseline. From the EfficientNet family of models, we only report the best performer, EfficientNet-B2. The saved time metric values are provided in Table 4.3 and the entire range of achievable values of the metric is plotted in Figure 4.7. The metric is also shown for high values of sensitivity in Figure 4.8. Training of the MLP-Mixer model was not successful with the default training protocol, and the model's performance is therefore not reported.

Performance of all models is within a 5% margin, with an outlier, the Vision Transformer model performing the worst. The best performance was achieved by EfficientNet-B2 models, reaching both the highest value of the saved time metric and the lowest standard deviation between folds, i.e., the architecture performs the best consistently.

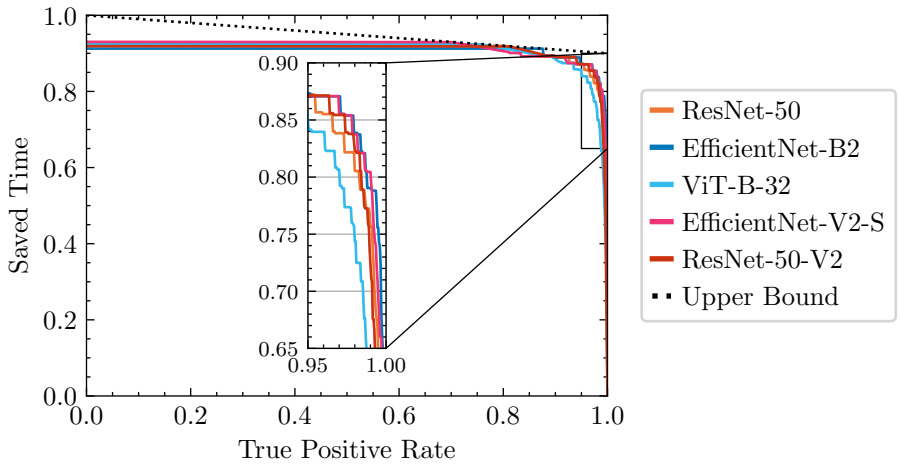



**Figure 4.7:** Saved time metric comparison of various models trained on grayscale images of scale 70%. Performance of all models is within a 5% margin, with an outlier, the Vision Transformer model performing the worst. The best-performing architecture is the EfficientNet-B2. The region of interest, with sensitivity higher than 95%, is displayed enlarged in Figure 4.8.

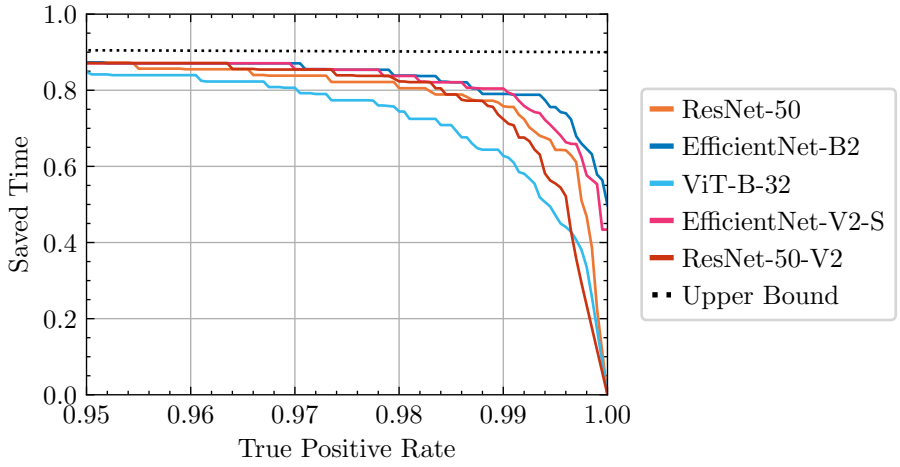



**Figure 4.8:** Saved time metric comparison of various models trained on grayscale images of scale 70%. The best performing architectures are the EfficientNet-B2 and EfficientNet-V2-S, which outperform the baseline ResNet-50 with a significant margin.

| Model | Sensitivity 98% | 99% | 99.5% |
|---|---|---|---|
| ResNet-50 (Baseline) | 0.805 (0.116) | 0.758 (0.180) | 0.643 (0.201) |
| EfficientNet-B2 | **0.839** (**0.048**) | 0.790 (**0.095**) | **0.756** (**0.100**) |
| ViT-B-32 | 0.744 (0.116) | 0.628 (0.178) | 0.472 (0.217) |
| EfficientNet-V2-S | 0.838 (0.053) | **0.805** (0.116) | 0.695 (0.119) |
| ResNet-50-V2 | 0.823 (0.081) | 0.724 (0.194) | 0.555 (0.275) |

**Table 4.3:** Saved time metric comparison for different models. The first value indicates the mean saved time metric; the second is the standard deviation between folds. The models were trained on grayscale images of scale 70% with 30-fold cross-validation. The best performance is achieved by EfficientNet models, reaching both the highest value of the saved time metric and the lowest standard deviation between folds, i.e., the architecture performs the best consistently.

### 4.2.2 Ensemble of Multiple Architectures

**Motivation & Goal.** It is well known that ensembling techniques, where a collection of models is used to create a single prediction, such as bagging and boosting, result in improved performance at the cost of increased computational complexity.

We, therefore, collect the models trained in Section 4.2.1 into an ensemble, defined in Section 3.2 and evaluate whether the ensemble achieves better performance.

**Setup.** To produce an ensemble, models $\{f_1, \ldots, f_5 \mid f_i : \mathcal{X} \to \mathbb{R}\}$ obtained from the experiment described in Section 4.2.1 were collected and their predictions were averaged, see Section 3.2. An ensemble was created for each of the 30 folds of data. Models within each ensemble were therefore trained on an identical portion of the dataset. Each ensemble of the model architectures was composed of a single (i) baseline ResNet-50, (ii) ResNet-50x1-V2, (iii) EfficientNet-B2, (iv) EfficientNet-V2-S, and (v) ViT-B-32.

We evaluate both variants of the ensemble defined in Section 3.2, i.e., variant where (i) all component models $f_i$ are assigned identical importance, and (ii) different component models $f_i$ are assigned different importance. For the case when different importance is assigned to each component, we obtain the weights $\alpha_i$ by maximizing the integral of the saved time metric curve from the true positive rate of 95% to the true positive rate of 100% on the validation set, i.e., we maximize the objective function $\int_{\mathrm{TPR}=95\%}^{100\%} \overline{\mathrm{ST}}_{\mathrm{val.}}(\mathrm{TPR})$, where $\overline{\mathrm{ST}}_{\mathrm{val.}}$ is the average value of the metric achieved on the validation set.

**Result.** Values of the saved time metric achieved by the ensemble can be found in Table 4.4. The entire curve displaying all possible values of the metric is shown in Figure 4.9. Figure 4.10 depicts the curve only for high values of true positive rate. The ensemble achieves results superior to its components. For the sensitivity of 98%, the ensemble also achieves exceptionally low standard deviation between folds, i.e., ensembling improves

the performance consistently across all folds.

The weights $\alpha_i$ assigned to components of the weighted ensemble are displayed in Table 4.5. The performance of the two ensemble variants is shown in Figure 4.11 and in Table 4.6. The performance of both variants is nearly identical, i.e., assigning different importance to components of the ensemble did not improve the performance of the ensemble on the test set.

| | Sensitivity | | |
|---|---|---|---|
| Model | 98% | 99% | 99.5% |
| ResNet-50 (Baseline) | 0.805 (0.116) | 0.758 (0.180) | 0.643 (0.201) |
| EfficientNet-B2 | 0.839 (0.048) | 0.790 (**0.095**) | 0.756 (**0.100**) |
| Ensemble | **0.869** (**0.027**) | **0.869** (0.155) | **0.837** (0.156) |

**Table 4.4:** Saved time metric comparison for the baseline ResNet-50, the best architecture of EfficientNet-B2, and an ensemble of multiple model architectures. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. The models were trained on grayscale images of scale 70% with 30-fold cross-validation. The ensemble of model architectures is composed of (i) the baseline ResNet-50, (ii) EfficientNet-B2, (iii) ViT-B-32, (iv) EfficientNet-V2-S, (v) ResNet-50-V2. An ensemble containing the aforementioned architectures was constructed for each fold of the dataset. The ensemble's output is constructed as the average of the component outputs.

| Model | | | | |
|---|---|---|---|---|
| ResNet-50 | EfficientNet-B2 | EfficientNet-V2-S | ResNet-50-V2 | ViT |
| 0.153 | 0.329 | 0.304 | 0.160 | 0.054 |

**Table 4.5:** Weights $\alpha_i$ assigned to components of the ensemble such that the predictions are optimal on the validation set. The objective function of the optimization is the integral of the saved time metric curve on the validation set from the true positive rate of 95% to the true positive rate of 100%.

| | Sensitivity | | |
|---|---|---|---|
| Model | 98% | 99% | 99.5% |
| ResNet-50 (Baseline) | 0.805 (0.116) | 0.758 (0.180) | 0.643 (0.201) |
| Ensemble: W. Average | **0.870** (0.028) | 0.854 (**0.148**) | 0.806 (**0.146**) |
| Ensemble: Mean | 0.869 (**0.027**) | **0.869** (0.155) | **0.837** (0.156) |

**Table 4.6:** Saved time metric of ensemble variants. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. The *Ensemble: Mean* model output is constructed as the average of the component outputs. The *Ensemble: W. Average* model output is formed as the *weighted* average of the component outputs, with the weights available in Table 4.5. For a formal definition of the two ensemble variants, refer to Section 3.2.

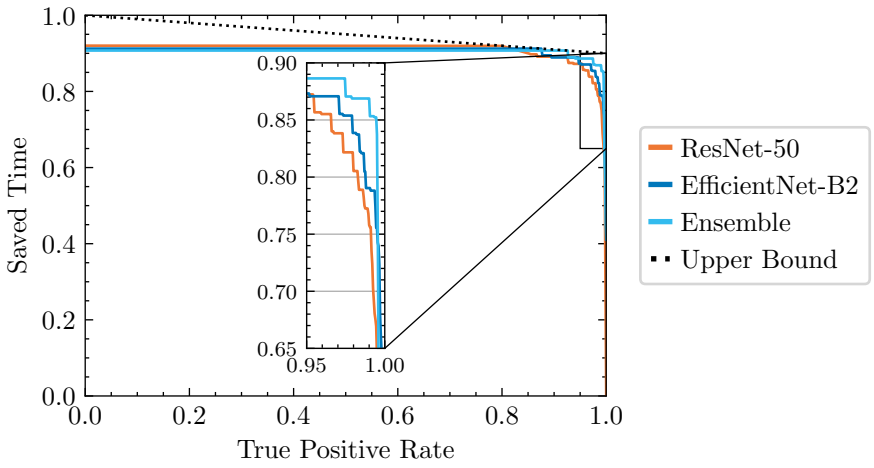



**Figure 4.9:** Saved time metric of model ensemble trained on grayscale images of scale 70% with 30-fold cross-validation. The ensemble of model architectures is composed of (i) the baseline ResNet-50, (ii) EfficientNet-B2, (iii) ViT-B-32, (iv) EfficientNet-V2-S, (v) ResNet-50-V2. An ensemble containing the aforementioned architectures was constructed for each fold of the dataset. The output of the ensemble is constructed as the average of the component outputs.

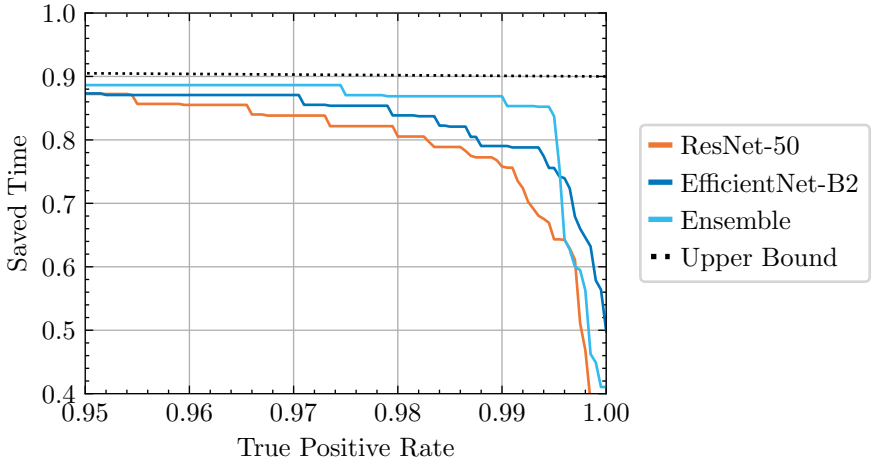



**Figure 4.10:** Saved time metric of model ensemble presented in Figure 4.9 shown for high values of sensitivity. Comparison of the ensemble with the baseline ResNet-50 model and the best performing architecture EfficientNet-B2.

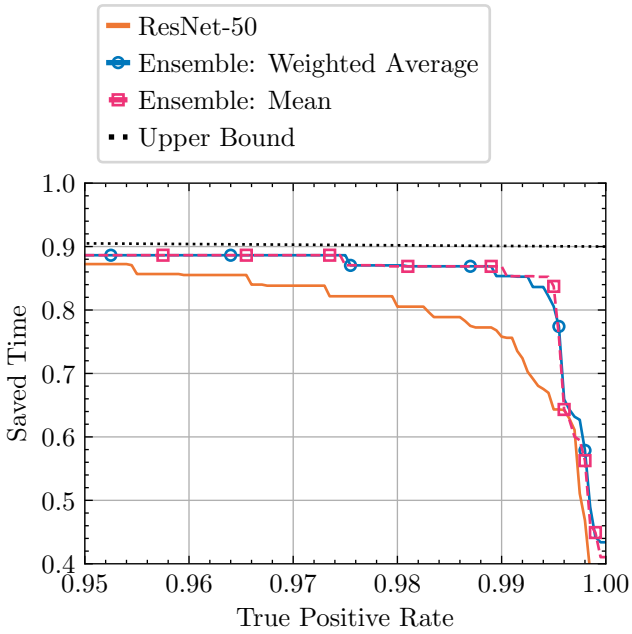



**Figure 4.11:** Saved time metric of ensemble variants. The *Ensemble: Mean* model output is constructed as the average of the component outputs. The *Ensemble: Weighted Average* model output is produced as the *weighted* average of the component outputs, with the weights available in Table 4.5. For a formal definition of the two ensemble variants, refer to Section 3.2.

## 4.3 Domain-Specific Augmentations

This section describes experiments conducted with data augmentation techniques proposed in Section 3.5 and evaluates their effects on model performance.

### 4.3.1 Poisson Augmentation of Fluorescent Stained Positive Samples

This section describes the experiment carried out to evaluate the effects of the Poisson augmentation described in Section 3.5.1.

**Motivation & Goal.** The experiment's goal is to see if the Poisson augmentation, designed specifically for the visual detection task, improves model performance. Another goal is to determine whether it is necessary to use Poisson image editing to perform the inpainting.

**Setup.** ResNet-50 and EfficientNet-B2 models were trained using a modification of the training protocol summarized in Section 3.4.2. The number of total training epochs for the experiment was increased from 150 to 200, and the *Poisson augmentation* described in Section 3.5.1 was applied to each positive sample with a probability of 25%. The models were trained using

30-fold cross-validation and the training, test, and validation splits of 80, 10, and 10%, respectively. The models were trained using an image scale of 70%. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 200,
- batch size: $b = 10$,
- loss function: binary cross entropy.

An ablation experiment was performed with the ResNet-50 models. The models were trained using the augmentation with both variants of inpainting shown in Figure 3.7, (i) the naive inpainting where the value of pixels is copied directly, and (ii) the seamless inpainting using Poisson image editing. The resulting models were evaluated on the saved time metric.

**Result.** ResNet-50 models trained with the augmentation using the seamless inpainting based on Poisson image editing consistently outperform the baseline. The augmentation does not improve performance when using the naive inpainting method. The saved time metric of the ResNet-50 models is shown in Figure 4.12 and in Table 4.7.

The same result, however, could not be replicated with EfficientNet-B2 models, where models trained without the novel augmentation performed better than models using the Poisson augmentation. The saved time metric of the EfficientNet-B2 models is shown in Figure 4.13 and in Table 4.8.

| | Sensitivity | | |
|---|---|---|---|
| Augmentation | 98% | 99% | 99.5% |
| Baseline | 0.805 (0.116) | 0.758 (0.180) | 0.643 (0.201) |
| +Poisson inpainting | **0.839** (**0.032**) | **0.804** (0.194) | **0.733** (0.205) |
| +Naive inpainting | 0.822 (0.071) | 0.759 (**0.170**) | 0.627 (**0.177**) |

**Table 4.7:** The effect of Poisson augmentation on the saved time metric. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. The baseline ResNet-50 model, trained using the standard augmentation protocol developed through experiments described in Section 4.1.2, is compared against ResNet-50 models trained using the standard protocol extended by the Poisson augmentation. The augmentation can be performed using two inpainting methods; both are shown in the table. The use of the augmentation consistently improves the model performance. The improvement can only be observed when utilizing the seamless inpainting method of Poisson image editing.

### 4.3.2 Blur Augmentation of Positive Samples with Localization Map

This section describes experiment carried out to evaluate the effects of the blur augmentation described in Section 3.5.1.

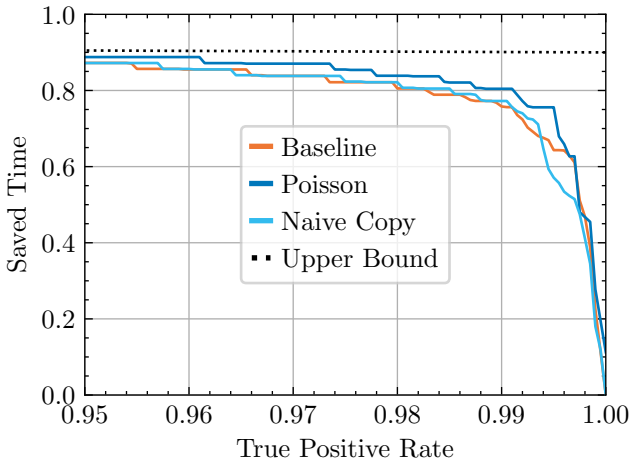



**Figure 4.12:** The effect of Poisson augmentation on the saved time metric. The baseline ResNet-50 model, trained using the standard augmentation protocol developed through experiments described in Section 4.1.2, is compared against ResNet-50 models trained using the standard protocol extended by the Poisson augmentation. Models using the Poisson augmentation with **Poisson** inpainting outperform the baseline. Models trained with the augmentation variant using **naive copy** inpainting perform analogously to the baseline.

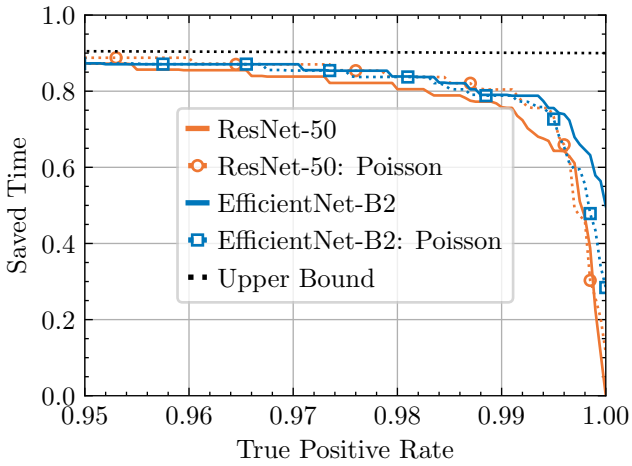



**Figure 4.13:** The effect of Poisson augmentation on the saved time metric of ResNet-50 and EfficientNet-B2 models. The use of the augmentation improves the performance of ResNet-50. It, however, slightly hinders the performance of EfficientNet-B2 models.

| Augmentation | Sensitivity 98% | 99% | 99.5% |
|---|---|---|---|
| Baseline | **0.839** (0.048) | **0.790** (**0.095**) | **0.756** (**0.100**) |
| +Poisson inpainting | 0.837 (**0.044**) | 0.788 (0.148) | 0.727 (0.155) |

**Table 4.8:** The effect of Poisson augmentation on the saved time metric of the EfficientNet-B2 models. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. Models trained without the augmentation reach higher values of the metric as well as lower standard deviation of the metric between folds.

**Motivation & Goal.** The goal of the experiment is to see if adding localized blur to regions with easily detectable contaminant features improves model performance and robustness.

**Setup.** ResNet-50 models were trained using a modification of the training protocol summarized in Section 3.4.2 with the addition of the *blur augmentation* described in Section 3.5.1. Within the first ten training epochs, the augmentation was not used in order to allow the model to learn the features without the difficulty of the task being increased. Afterward, the augmentation was utilized for a training sample with a probability of 20%. Although the data augmentation technique was designed for positive samples, we apply the augmentation to both positive and negative samples to prevent the model from learning that only positive samples are blurred. To apply the augmentation to a negative sample, we choose a localization mask generated by some positive sample at random.

Ideally, the contaminant localization map of a sample would be created using Grad-CAM, [Selvaraju et al., 2017], each time the sample was chosen for a mini-batch. To reduce computation time, we generate maps for all samples every ten epochs and keep them fixed for the following ten epochs. Grad-CAM localization maps contain a wide range of values. We use Yen's thresholding algorithm, [Yen et al., 1995], to convert each map into a binary localization mask, which indicates the position of the contaminant.

When the augmentation is applied, regions specified by the binary mask are blurred. The experiment was repeated multiple times, blurring either 25, 50, or 75% of the masked regions. For details, refer to the description of the augmentation in Section 3.5.1.

The models were trained using 30-fold cross-validation and the training, test, and validation splits of 80, 10, and 10%, respectively, and using an image scale of 70%. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The resulting models were evaluated on the saved time metric.

**Result.** No consistent improvement was observed when utilizing the augmentation. Performance of ResNet-50 models trained with the augmentation depends on the area which was blurred, e.g., the variant blurring 50% of the masked area outperforms the baseline for TPR $< 0.99$, but the variant blurring 25% of the masked area performs comparably with the baseline for TPR $< 0.99$. All variants of the augmentation however result in performance degradation for TPR $> 0.99$. The saved time metric of the models is shown in Figure 4.14.

We evaluate the robustness of the models by computing their performance when subject to an adversarial attack, specifically the Fast Gradient Sign Method (FGSM), [Goodfellow et al., 2014]. The attack consists of adding noise to the input of the model, which is designed to cause incorrect prediction. Specifically, instead of the input $x \in \mathcal{X}$, the model $f$ is presented with the input $x + \epsilon \cdot \text{sign}(\nabla_x L(x, y))$, where $L(x, y)$ is a loss function. The severity of the attack depends on the magnitude of the noise $\epsilon \in \mathbb{R}$. Example of the attack is shown in Figure 4.15. We show the performance of the models subject to FGSM in Figure 4.16. Models trained with the blur augmentation consistently exhibit better adversarial robustness; however, the improvement over the baseline is marginal.

The augmentation can not be recommended in its current form. An improvement of the augmentation, proposed by [Fong and Vedaldi, 2019], is to utilize batch augmentation, i.e., each mini-batch contains augmented copies of a single image. The procedure could also be improved by generating contaminant localization masks at each epoch, see experiment setup.

| | Sensitivity | | |
|---|---|---|---|
| Augmentation | 98% | 99% | 99.5% |
| Baseline | 0.805 (0.116) | 0.758 (**0.180**) | **0.643** (**0.201**) |
| +Blur 25% | 0.807 (0.094) | 0.759 (0.201) | 0.532 (0.227) |
| +Blur 50% | **0.838** (0.142) | **0.772** (0.229) | 0.460 (0.285) |
| +Blur 75% | 0.805 (**0.087**) | 0.724 (0.184) | 0.574 (0.224) |

**Table 4.9:** The effect of blur augmentation on the saved time metric. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. The augmentation does not provide significant improvement in performance. Instead, the performance is exacerbated for a true positive rate of over 99%.

### 4.3.3 Poisson Augmentation of Negative Samples with Localization Map

This section describes the experiment carried out to evaluate the effects of the augmentation described in Section 3.5.2.

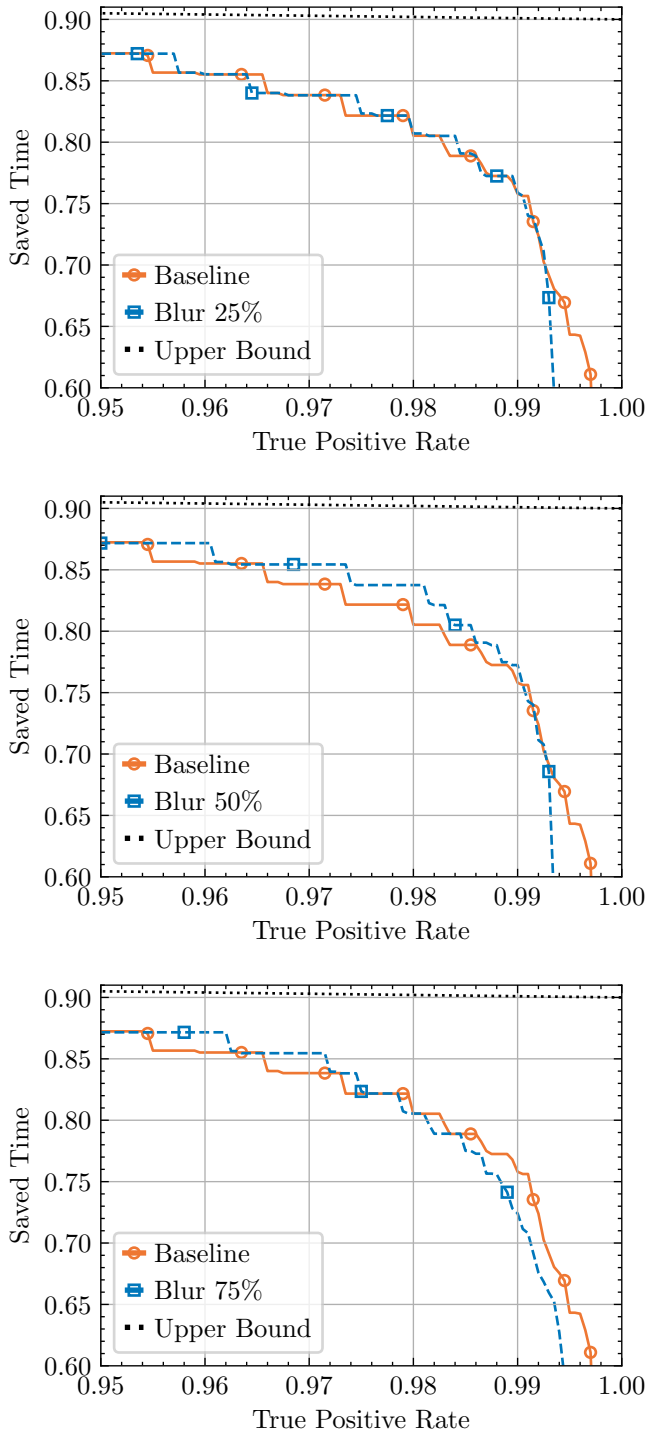



**Figure 4.14:** The effects of the blur augmentation of positive samples on the saved time metric of ResNet-50 models. Due to the overlapping of the curves, the result of each experiment is presented in a separate graph. The augmentation does not provide significant improvement in performance. Instead, the performance is exacerbated for a true positive rate of over 99%.

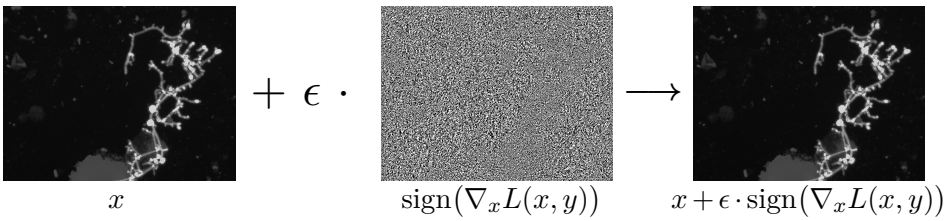



**Figure 4.15:** Example of the FGSM adversarial attack.

**Motivation & Goal.** The goal of the experiment is to see if replicating structures in negative images, which are visually similar to yeast and filamentous fungi, forces the models to learn stronger features and consequently improves model performance and robustness.

Decrease of false positive rate can be expected when using the augmentation, because the augmentation is designed to make the model better at recognizing negative samples. Consequently, the saved time can be expected to improve for TPR $\to 1$, as the saved time then depends purely on the incidence rate and false positive rate,

$$\mathrm{ST}(h) \overset{\mathrm{TPR}\to 1}{=} [1 - p(y = +1)] \cdot [1 - \mathrm{FPR}(h)]. \tag{4.1}$$

**Setup.** Identically to the experiment described in Section 4.3.2, the localization map is generated using Grad-CAM, [Selvaraju et al., 2017]. Ideally, the contaminant localization map of a sample would be created from scratch each time the sample was chosen for a mini-batch. However, in the same manner as described in Section 4.3.2, we generate maps for all samples every ten epochs to reduce the computation time.

The augmentation was not applied for the initial ten epochs. Afterward, the augmentation was utilized for a negative training sample with a probability of 25%. If the localization mask spanned an area larger than 40% of the entire image, the augmentation was not utilized, as the localization was deemed insufficiently specific. The cause of such cases is explained in the description of the augmentation, Section 3.5.2.

The models were trained using 20-fold cross-validation and the training, test, and validation splits of 80, 10, and 10%, respectively, and using an image scale of 70%. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The resulting models were evaluated on the saved time metric.

**Result.** ResNet-50 models trained with the Poisson augmentation of negative samples consistently perform worse than the baseline. The saved time achieved by the models is shown in Figure 4.17 and in Table 4.10.

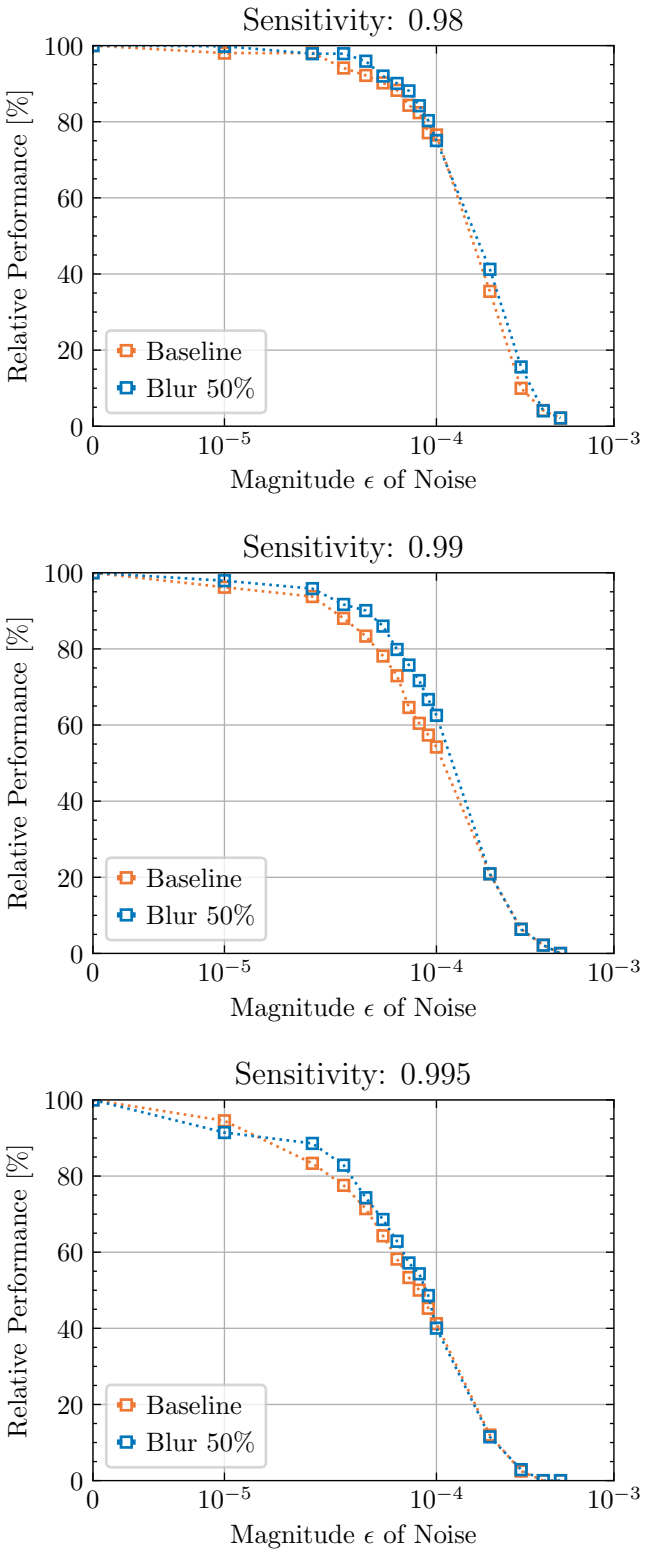



**Figure 4.16:** Performance of models when subject to the FGSM adversarial attack. Instead of the input $x$, the model $f$ is presented with the input $x + \epsilon \cdot \text{sign}\big(\nabla_x L(x, y)\big)$, where $L(x, y)$ is a loss function . Models trained with the blur augmentation consitently exhibit better adversarial robustness; however, the improvement is marginal.

Models trained with the augmentation of negative samples outperform the baseline only for TPR = 1, where they reach non-zero saved time. This improvement likely arises from the models being better at recognizing negative samples, hence a decrease of the FPR and an increase of the saved time.

The augmentation can not recommended for the agreed-upon mode of operation, where models operate at a true positive rate TPR = 0.99. However, if the requested TPR on the test data changes to TPR = 1, the augmentation should be used.

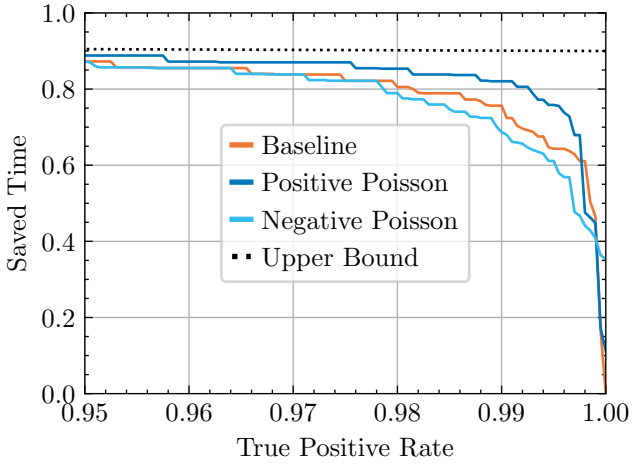



**Figure 4.17:** The effect of Poisson augmentation on the saved time metric. The augmentation of **positive samples** was experimented with in Section 4.3.1 and is shown purely for comparison. The augmentation of **negative samples** consistently decreases the achieved saved time, except for TPR $\to$ 1, where models trained with the augmentation of negative samples are the only ones to reach non-zero saved time. This improvement likely arises from the models being better at recognizing negative samples, hence decreasing the FPR.

| | Sensitivity | | | |
|---|---|---|---|---|
| Augmentation | 98% | 99% | 99.5% | 100% |
| Baseline | **0.805** (**0.068**) | **0.756** (0.162) | **0.643** (0.181) | 0.000 (0.202) |
| +Negative Poisson | 0.789 (0.125) | 0.687 (**0.146**) | 0.611 (**0.148**) | **0.354** (**0.150**) |

**Table 4.10:** The effect of Poisson augmentation of negative samples on the saved time metric. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. The use of the augmentation consistently hinders the model performance. An improvement can only be observed for TPR $\to$ 1, likely caused by a decrease of the FPR.

### 4.3.4 Poisson Augmentation of Negative & Positive Samples

This section describes the experiment carried out to evaluate the effects of using Poisson augmentation of both positive, and negative samples. The two

variants of the augmentation are described in Sections 3.5.1 and 3.5.2.

**Motivation & Goal.** The Poisson augmentation was evaluated in isolation for positive and negative samples in Sections 4.3.1 and 4.3.3 respectively. The Poisson image editing can potentially create artifacts in the images. If the augmentation is utilized for images of a single class, the model can learn to associate the artifacts with the class. Therefore, the goal of the experiment is to evaluate the effect of utilizing both augmentations concurrently.

**Setup.** The models were trained using 15-fold cross-validation and the training, test, and validation splits of 80, 10, and 10%, respectively, and using an image scale of 70%. Poisson augmentation of both positive and negative samples was utilized. The exact setup of the augmentations was identical to the setups described in Sections 4.3.1 and 4.3.3. The Nesterov variant of the SGD optimizer was utilized with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The resulting models were evaluated on the saved time metric.

**Result.** Models trained using the augmentation of both positive and negative samples consistently outperform the baseline. They also consistently outperforms models trained with Poisson augmentation of only positive or only negative samples. The saved time achieved by the models is shown in Figure 4.18 and Table 4.11.

Because Poisson augmentation of negative samples in isolation degrades performance compared to baseline, a model trained with both positive and negative samples augmented could be expected to perform worse than models trained with only positive samples augmented. However, this is not the case. We contend that the most likely reason is that potential inpainting artifacts cannot be associated with a single class when both classes are augmented. As a result, instead of detecting that inpainting was performed, the model must learn to decide based on the inpainted content.

## 4.4 Learning Curve

**Motivation & Goal.** The dataset, described in Section 3.1, was collected over three years and contains a total of 1244 samples. Other machine learning datasets can typically contain orders of magnitude more samples. Collecting additional positive samples is a strenuous and expensive process, but increasing the size of the training dataset generally improves the final performance of the model. At a certain point, however, additional samples yield no improvement in the performance, or the improvement is not worth the investment.

| Augmentation | Sensitivity 98% | 99% | 99.5% | 100% |
|---|---|---|---|---|
| Baseline | 0.838 (0.053) | 0.788 (0.102) | 0.696 (0.110) | 0.450 (0.118) |
| + Negative Poisson | 0.789 (0.131) | 0.685 (0.155) | 0.579 (0.157) | 0.354 (0.159) |
| + Positive Poisson | **0.854** (0.022) | 0.820 (0.170) | 0.758 (0.178) | 0.112 (0.186) |
| + P. & N. Poisson | 0.853 (**0.020**) | **0.822** (**0.046**) | **0.806** (**0.047**) | **0.691** (**0.049**) |

**Table 4.11:** The effect of Poisson augmentation of positive and negative samples on the saved time metric. The first value in the table indicates the mean saved time metric; the second value is the standard deviation between folds. The models trained with augmentation of both positive and negative samples consistently outperform all of the other variants and achieve the lowest standard deviation between folds by a large margin.

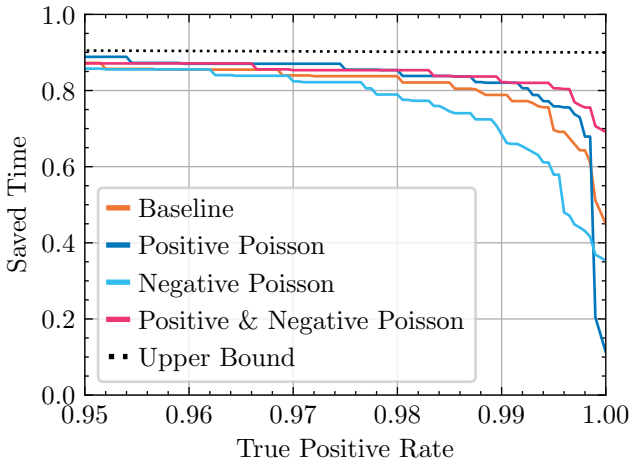



**Figure 4.18:** The effect of Poisson augmentation of positive and negative samples on the saved time metric. The baseline ResNet-50 model, trained using the standard augmentation protocol developed through experiments described in Section 4.1.2, is compared against ResNet-50 models trained using the standard protocol extended by the Poisson augmentation. Models using the Poisson augmentation for **positive samples** outperform the **baseline** except for TPR $\to 1$. Models trained with the augmentation of **negative samples** perform consistently worse than the **baseline**. Models trained using the augmentation of both **positive** and **negative** samples, shown in **pink**, consistently outperform all of the other variants and achieve the lowest standard deviation between folds. Values of the saved time metric and the standard deviation between folds are shown in Table 4.11. It should be noted that the comparison is not entirely fair. Models trained with Poisson augmentation of only **positive samples** were trained for a total of 200 epochs, while the other models were trained for 150 epochs.

The standard procedure for determining whether to collect additional data is to generate a learning curve, showing the relation of the primary metric with the amount of data used for training. The resulting curve is then extrapolated to predict the effect of adding more data. The typical learning curve of machine learning models is shown in Figure 4.19.

The goal of the experiment is to investigate in what portion of the typical learning curve is the provided dataset.

**Setup.** To generate the learning curve, ResNet-50 models were trained using the standard training procedure described in Section 3.4.2, i.e., using the Nesterov variant of the SGD optimizer with

- learning rate: $\alpha = 10^{-3}$,
- momentum: $\eta = 0.9$,
- learning rate decay: $\gamma = 0.33$ every 50 epochs,
- total training epochs: 150,
- batch size: $b = 10$,
- loss function: binary cross entropy.

The models were trained using 11-fold cross-validation with an image scale of 70% and the training, test, and validation splits of 80, 10, and 10%, respectively. The models were trained with a reduced amount of data, using only 25%, 50%, or 75% of the whole dataset. Each training epoch performed an identical number of optimization steps, regardless of the dataset size. This was achieved by possibly sampling images multiple times in a single epoch.

**Result.** The resulting saved time metric curves are shown in Figure 4.20. The learning curve is shown in Figure 4.21. The performance is steeply improving with additional data, and extrapolating the curve suggests that further improvements can be expected when more training samples are collected, i.e., the dataset can be expected to currently reside within the green or the blue region shown in Figure 4.19.

## 4.5 Influence of Difficult Samples on the Metric

**Motivation & Goal.** As the operating point of the models is required to reside at a true positive rate higher than 99% and the dataset contains a relatively small number of samples, it is reasonable to assume that a small number of positive samples can significantly impact the saved time metric. A curve was constructed to decide whether that is the case, which shows how the saved time metric changes when positive images that are often misclassified as negative are removed from the dataset.

**Setup.** Each image in the dataset was assigned a score obtained as the mean over predictions of models that did not observe the image during training, i.e., the image was in the test split of the fold. Formally, given classifiers $f_k : \mathcal{X} \to \mathbb{R}$ and their corresponding test sets $\mathrm{T}_k = \{(x_{k,i}, y_{k,i}) \in$

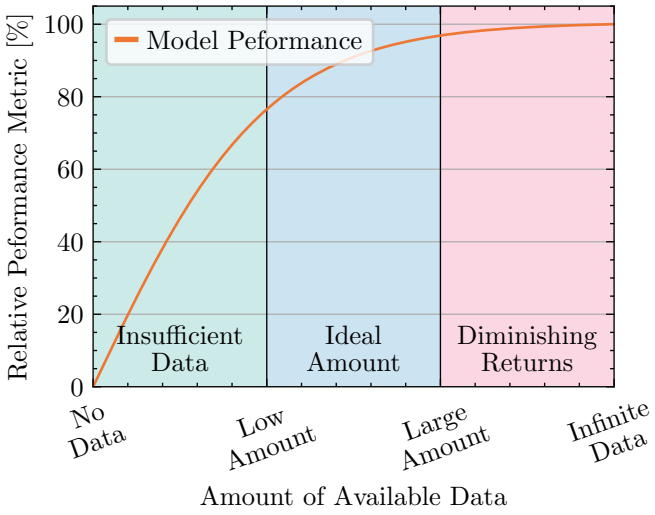



**Figure 4.19:** The typical learning curve of a machine learning model. When the amount of available data is low, adding more data results in large improvements in the model performance. At a certain point, however, additional samples yield no improvement in the performance, or the improvement is not worth the investment (diminishing returns).

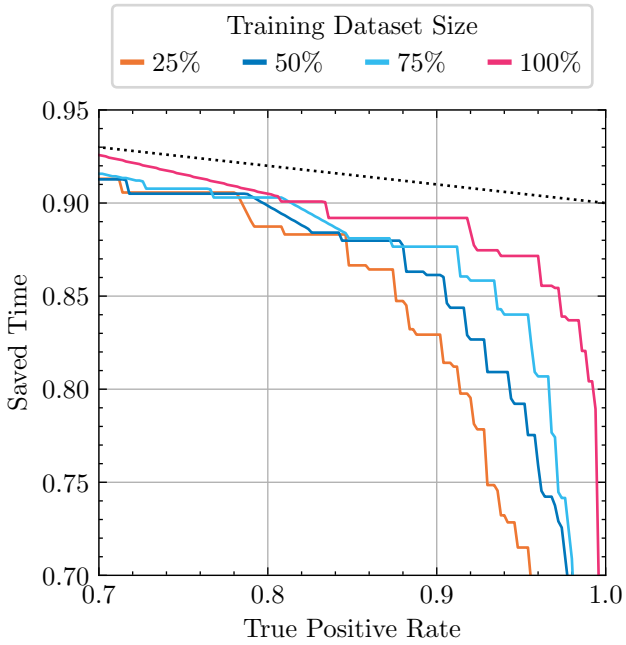



**Figure 4.20:** Saved time metric for ResNet-50 models trained on 11-fold grayscale images with reduced size of the training dataset. Increasing the size of the dataset improves the saved time metric. The corresponding learning curve is shown in Figure 4.21. Note that the curve for 100% of the dataset may not precisely match the curve displayed in the results of other experiments where the metric is evaluated for 30 folds.

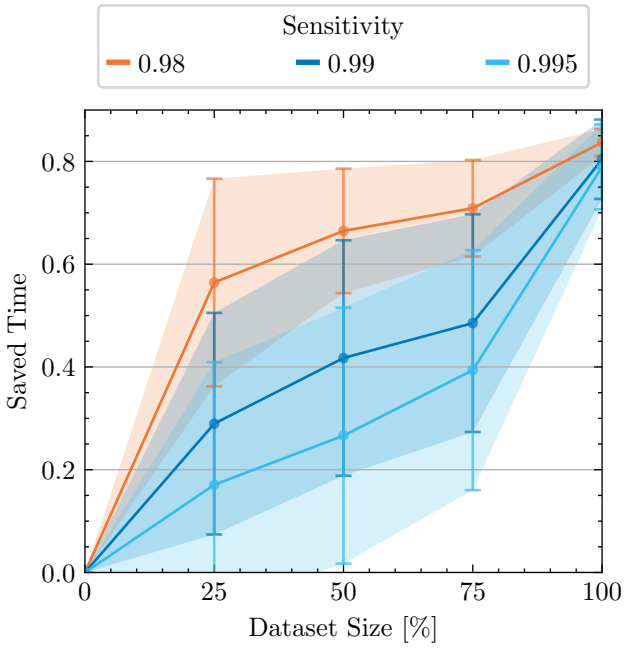



**Figure 4.21:** Learning curve displaying the saved time metric for ResNet-50 models trained on 11-fold grayscale images. The transparent area marks ±1 standard deviation between folds. The standard deviation is high but decreases with additional data. The performance is steeply improving with additional data, and the curve suggests that diminishing returns are not present with the available amount of data.

$\mathcal{X} \times \{+1, -1\} \mid i = 1, \ldots, n_k\}$, we assign a score $F(x)$ to each image $x$ as

$$F(x) = \frac{\sum\limits_{\{i|x \in \mathrm{T}_i\}} f_i(x)}{\sum\limits_{\{i|x \in \mathrm{T}_i\}} 1}. \tag{4.2}$$

The folds of the dataset were constructed as described in Section 3.6. Each image was therefore present in the test split of exactly three folds, i.e., three ResNet-50 models computed the score for each image. Images with low scores are likely to be classified as negative. Consequently, the models struggle to detect the contaminants in positive images with a low $F(x)$ score, i.e., detecting the contaminant is difficult. We refer to such positive samples as *difficult* samples.

The saved time metric was computed for all the models while continuously removing the positive sample with the lowest score, i.e., the most difficult positive sample. Ideally, the difficult samples would be removed, and the models retrained on the resulting dataset. This approach is, however, exceptionally computationally intensive. Therefore the images were only removed from the test set while evaluating the saved time metric.

Formally, the aforementioned samples form a dataset $\mathrm{D} = \mathrm{P} \cup \mathrm{N}$, where $\mathrm{P} = \{(x_i, y_i) \in \mathcal{X} \times \{+1\} \mid i = 1, \ldots, n_\mathrm{P}\}$ are the positive samples and

$\mathrm{N} = \{(x_i, y_i) \in \mathcal{X} \times \{-1\} \mid i = 1, \ldots, n_\mathrm{N}\}$ are the negative samples. We define a set of positive samples ordered by the score $F(x)$ as

$$\mathrm{P}^F = \{(x_i, y_i) \in \mathcal{X} \times \{+1\} \mid i = 1, \ldots, n\,;\, F(x_i) < F(x_{i+1})\}, \tag{4.3}$$

and define $\mathrm{P}^F_j \subseteq \mathrm{P}^F$ as the subset of $\mathrm{P}^F$ with the $j$ positive samples with the lowest $F(x)$ score removed

$$\mathrm{P}^F_j = \{(x_i, y_i) \in \mathcal{X} \times \{+1\} \mid i = j+1, \ldots, n\,;\, F(x_i) < F(x_{i+1})\}. \tag{4.4}$$

Finally, we define a reduced dataset $\mathrm{D}_j \subseteq \mathrm{D}$ as $\mathrm{D}_j = \mathrm{P}^F_j \cup \mathrm{N}$. Then, for each classifier $f_k$ and a corresponding test set $\mathrm{T}_k = \{(x_i, y_i) \in \mathcal{X} \times \{+1, -1\} \mid i = 1, \ldots, m\}$, we compute the empirical estimate of the saved time metric for the model on a reduced test set $\mathrm{T}_k \cap \mathrm{D}_j$. The average performance of the models $\overline{\mathrm{ST}}$ over all folds was computed as described in Section 3.13.

With slight abuse of notation, we can denote by $\overline{\mathrm{ST}}_j$ the average performance of models evaluated on a reduced dataset $\mathrm{D}_j$. We plot the curve

$$\{(j, \overline{\mathrm{ST}}_j) \mid j = 1, \ldots, n\}. \tag{4.5}$$

**Result.** The result is shown in Figure 4.22. As the most difficult samples are removed, making the detection task easier, the value of the saved time metric expectedly increases. The saved time metric remains constant for the sensitivity of 98% while removing the three most difficult positive samples. On the other hand, for a sensitivity of 99.5%, removing the three most difficult positive samples causes the metric to change by over 10%, suggesting that the value of the saved time metric for a very high sensitivity can be largely affected by a small number of images, confirming the hypothesis. The metric's value for lower sensitivities, such as 98%, should therefore be a better choice for comparing models.

An identical experiment was conducted for EfficientNet-B2 and for an ensemble of different model architectures. Result of the experiment is shown in Figure 4.23. The saved time metric of the ensemble of models does not significantly change when the most difficult positive samples are removed, even for high values of true positive rate. This suggests that the ensemble of models produces more robust predictions and is more resistant to changes in the dataset.

## 4.6 Inference Time

**Motivation & Goal.** As described in Section 3.6.3, the inference time of a model is critical for its large-scale deployment, especially when a single slide can produce several thousand images when swept by an automated slide scanner. Furthermore, the inference time of an ensemble grows linearly with the number of models contained in the ensemble.

This experiment should therefore answer: (i) what is the throughput of each model, and (ii) is specialized hardware required for laboratory deployment.

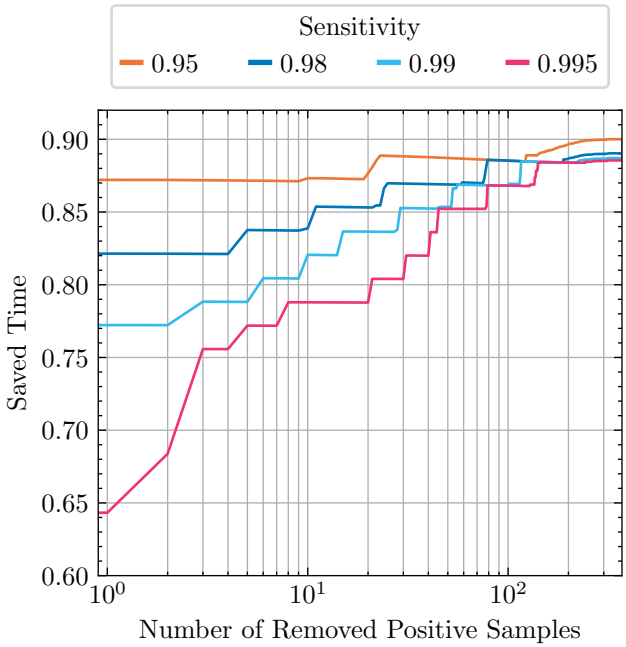



**Figure 4.22:** Volatility of the saved time metric of the baseline ResNet-50 model for high sensitivity values. Removing the most difficult positive samples drastically changes the value of the saved time metric for high sensitivity values such as 99.5%, suggesting that the metric is volatile for high sensitivities, and models should be compared on the metric for lower sensitivities, e.g., 98%.

**Setup.** Only the forward pass of the models was measured. Thus the time consumed by reading and transferring the data to the processing device was not considered. To measure the inference time (i) synchronous execution of CPU and GPU was forced, (ii) the GPU was warmed-up by artificial computations, and (iii) the forward pass time was measured by `torch.cuda.Event` timers. The measurements were carried out for all samples in the dataset and were repeated fifty times for each sample. All measurements were performed on a single NVIDIA GeForce GTX 1080 Ti GPU.

**Result.** We measure the time for the case when (i) a single image is processed by the GPU at a time, result shown in Figure 4.24, and for the case when (ii) multiple images are processed simultaneously in a batch, result shown in Figure 4.25.

Throughput of the models ranges between 20 and 65 images per second. All models contained within the ensembles, see Section 4.2.2, achieve a throughput of approximately $\lambda \approx 40$ images per second. The ensemble, composed of five models, should therefore achieve throughput of at least $\lambda_{\text{ensemble}} \approx 8$ images per second.

The number of images obtained from a single slide depends on the scanned area and the desired magnification. Typically, the number can range from a hundred images to several thousand. With the conservative assumption that

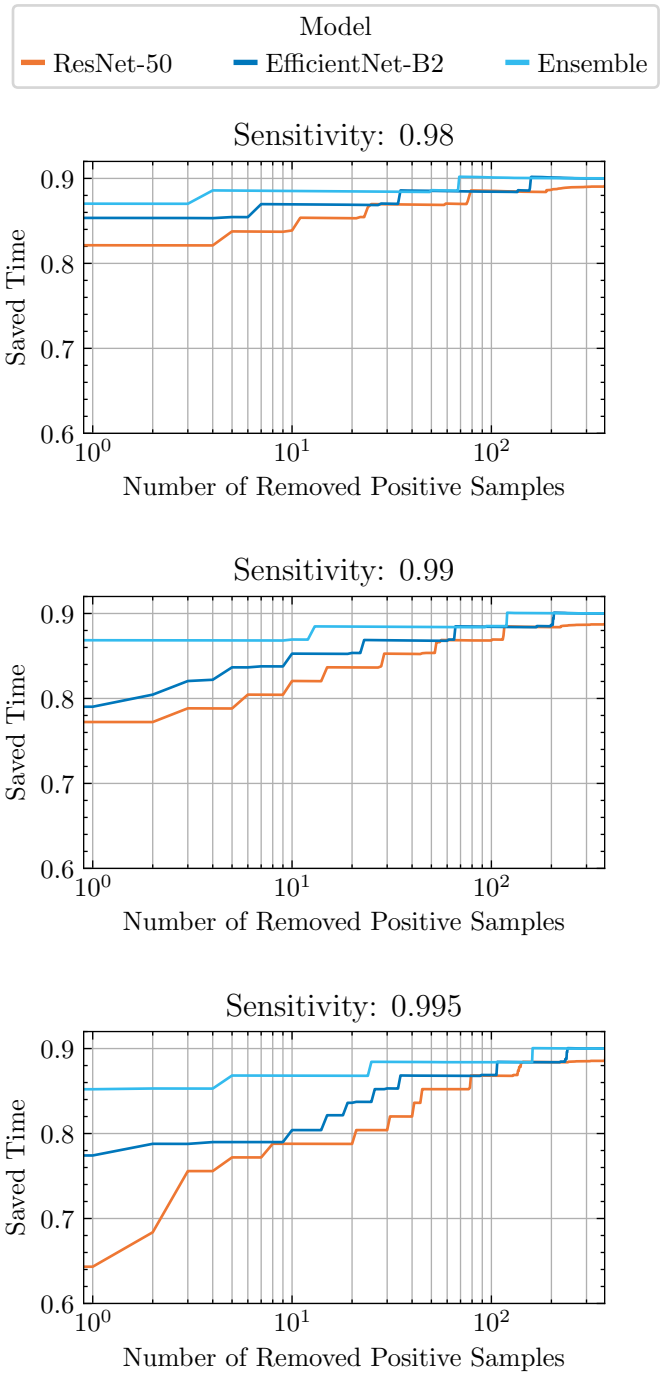



**Figure 4.23:** Volatility of the saved time metric. Comparison of the ensemble of model architectures, baseline ResNet-50, and the best architecture EfficientNet-B2. An ensemble was created for each of the 30 folds of the dataset independently and was composed of a single (i) baseline ResNet-50, (ii) ResNet-50x1-V2, (iii) EfficientNet-B2, (iv) EfficientNet-V2-S, and (v) ViT-B-32. The saved time metric of the ensemble does not significantly change when the most difficult positive samples are removed. Note that the score $F(x)$ marking difficulty of images was constructed for each model independently, as images that are difficult for the ResNet-50 models may not be difficult for the EfficientNet-B2 models or the ensemble.

each slide produces 2500 images of dimensions similar to the training dataset, a single model would be able to produce the classification in under a minute. The classification process can be started before the entire scan is finished; the computation should, therefore, not pose a bottleneck to the automation.

A more accurate estimate would require specifications of a particular slide scanner. However, the presented results should allow the medical facility to make an educated decision when selecting a slide scanner for the automation.

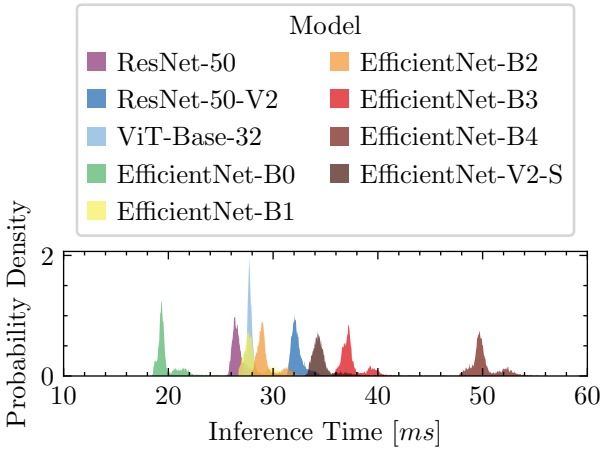



**Figure 4.24:** Inference time measurement of models when processing a single sample. The probability densities displayed were obtained empirically by measuring the time required to compute a model's forward pass. The measurements were carried out for all samples in the dataset and were repeated 50 times for each sample. The mean of the inference time, as well as the standard deviation, for all models, are shown in Table 4.12.

| Model | Mean $[ms]$ | St. Dev. $[ms]$ |
|---|---|---|
| EfficientNet-B0 | 20.036 | 1.674 |
| ResNet-50 | 26.747 | 0.727 |
| ViT-B-32 | 27.852 | 0.535 |
| EfficientNet-B1 | 28.485 | 1.878 |
| EfficientNet-B2 | 29.575 | 1.789 |
| ResNet-50-V2 | 32.367 | 0.772 |
| EfficientNet-V2-S | 35.313 | 2.907 |
| EfficientNet-B3 | 37.598 | 1.317 |
| EfficientNet-B4 | 50.219 | 1.432 |

**Table 4.12:** Inference time measurement of models when processing a single sample. The measurements were carried out for all samples in the dataset and were repeated 50 times for each sample.

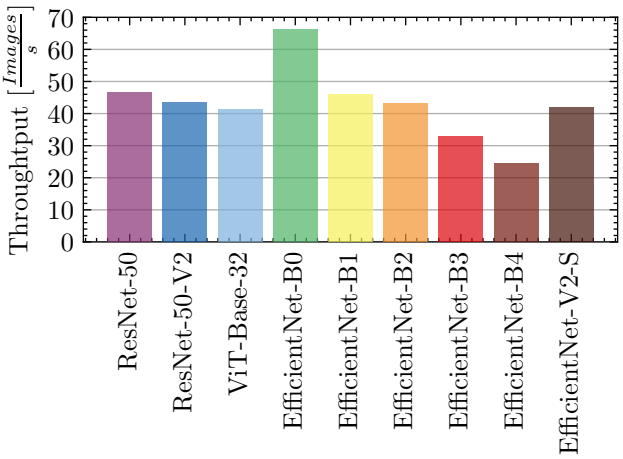



**Figure 4.25:** Throughput of models when utilizing optimal batch-size on a single NVIDIA GeForce GTX 1080 Ti GPU. According to the results, the inference time should not be a bottleneck in the automated process.

## 4.7 Human-Machine Comparison

**Motivation & Goal.** This experiment attempts to provide answers to two questions: (i) how does the model performance compare against expert-level humans, and (ii) how does the model's performance compare to that of inexperienced humans? Some tasks are simple for humans but extremely difficult for machines. This is partly due to humans' ability to apply their current knowledge to newly introduced problems. Thus, if non-expert knowledge transfers well to detecting microscopic fungi, a human with a lower qualification level could serve as a substitute for the automated model.

The outcome of the experiment should either: (i) increase the confidence in the model predictions, or (ii) demonstrate that the performance is not satisfactory.

**Setup.** A human-machine comparison was set up to see if the model's performance was sufficient. From the dataset, 100 positive and 100 negative images were chosen at random. The images were then shown to expert microbiologists who routinely detect microscopic fungi and yeast in clinical samples. They were prompted to classify the images as either positive or negative. A group of amateurs in the field of microbiology was also shown the images after a brief training session that included a showcase of 20 representative positive and 20 negative samples. The task was verbally explained, and special attention was given to showcasing the specific structures of the contaminants.

**Result.** The saved time metric values achieved by the participants and the models are shown in Figure 4.26. The performance is also compared on a ROC curve shown in Figure 4.28.

All expert microbiologists perform similarly, achieving a true positive rate of 89%, 89%, 90% and 94% with a saved time of 91.1%, 91.1%, 89.2% and 90.6% respectively. The experts achieve values of the saved time at the theoretical upper bound, i.e., the experts achieve a false positive rate of FPR $\approx 0$. It should be noted that the presented task is significantly different from the standard operating procedure of the experts. In the usual setting, the expert is presented with an entire slide and can freely move between portions of the slide and search for the contaminant. In this experiment, the view is locked, and the expert is presented with a single image.

The novice humans perform significantly worse than the automated model. They either (i) achieve an insufficient true positive rate, or (ii) achieve sufficient true positive rate, but with a low value of the saved time metric. This suggests that generic knowledge does not transfer well to the task.

The ensemble of models is able to perform at the same level or a better level than the expert humans. This result, although obtained only on the limited set of 200 images, suggests that the model performs sufficiently.

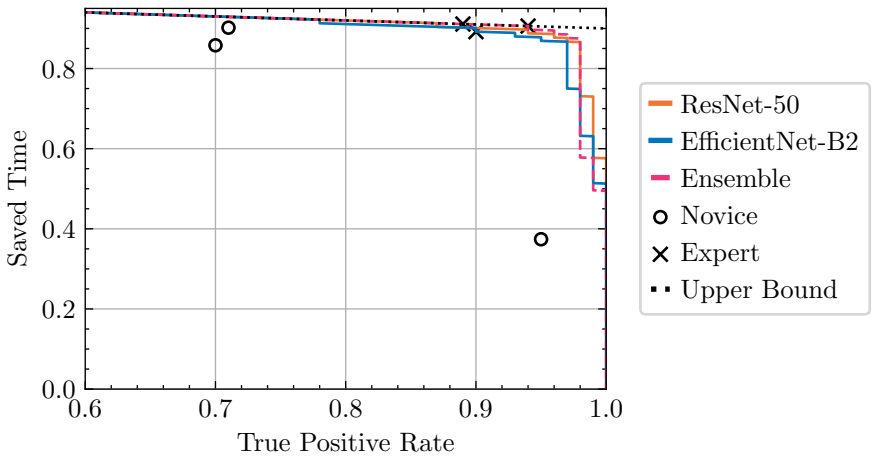



**Figure 4.26:** Comparison of human-machine performance on the saved time metric. The comparison was created for a randomly selected set of 100 positive and 100 negative samples. Medical experts perform significantly better than novices, who are outperformed by the automated model by a significant margin. Detail of the human expert performance is shown in Figure 4.27.

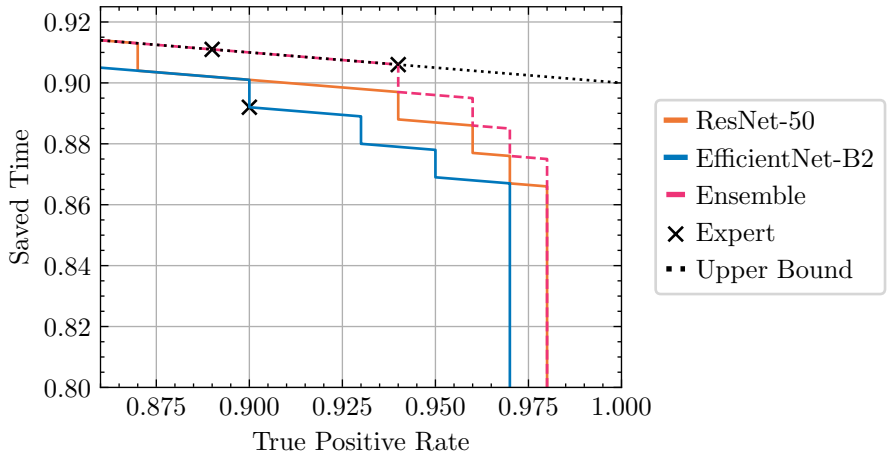



**Figure 4.27:** Comparison of human-machine performance on the saved time metric. The figure shows zoomed region, which contains the expert human results. In contrast to performance on the entire dataset, shown in Figure 4.8, the ResNet-50 architecture performs better than EfficientNet-B2 for the used test set. The ensemble, when predicting the classification of a sample, uses only predictions of models which did not observe the sample during training, i.e., models trained on folds, where the image was in the test set. Each image was present in the test set three times. Therefore, the ensemble always contains three instances of each of (i) ResNet-50, (ii) EfficientNet-B2, (iii) ViT-B-32, (iv) EfficientNet-V2-S, (v) ResNet-50-V2, a total of 15 models. The ensemble's output was constructed as the average of the component outputs.

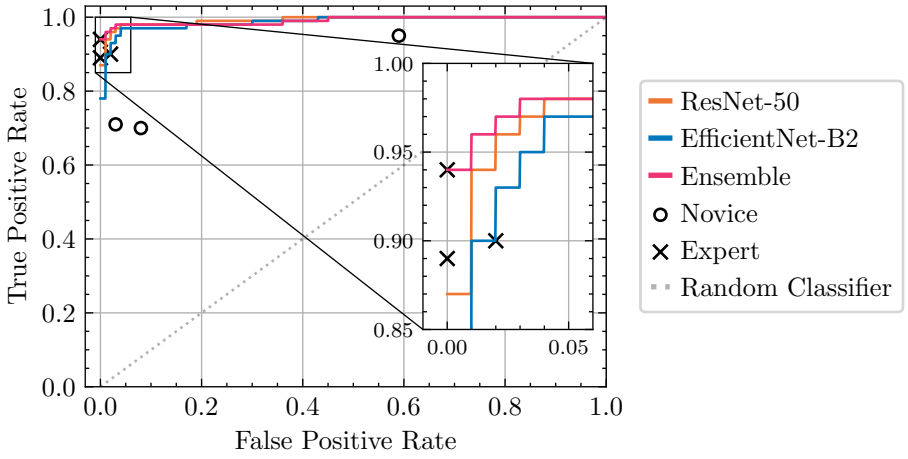



**Figure 4.28:** Human-machine performance comparison on the ROC curve.

# Chapter 5

# Discussion

Severe infections caused by filamentous fungi and yeast are sporadic but severe; therefore, patients with risk factors are regularly screened. Consequently, a large number of slides, the majority of which do not contain yeast or filamentous fungi, must be carefully examined and classified by a human expert. This thesis aims to investigate the possibility of using deep neural networks to either completely replace the human operator or to significantly simplify their work by filtering out samples that are clearly negative and presenting the operator with only samples suspected of containing the contaminant.

The results indicate that the task can be tackled by employing an ensemble of convolutional neural networks, Section 4.2.2. The developed model consistently performs on par or better than a human expert, see Section 4.7, and if deployed, should reduce the amount of manual labor by approximately 85% when operating at a true positive rate of 99%. This result is surprising, as the method was trained with annotations on the image level, i.e., the network was not instructed on which parts of the image are responsible for the classification. Nonetheless, the models successfully learned to classify the images, indicating that the annotations are sufficient. However, this claim is based on the results on the limited dataset, for which optimal pre-processing and data augmentation techniques were identified experimentally in Section 4.1. Therefore, performance on other datasets may be worse, and the models may require additional fine-tuning. In Section 4.4 it was shown that collecting additional data would further improve performance. Domain-specific data augmentation techniques were therefore proposed to alleviate the data scarcity and were employed with promising results, shown in Section 4.3.

Various deep learning models were trained, and their performance was evaluated. Model architectures introduced in recent years, such as the Vision Transformers, were outperformed by state-of-the-art convolutional neural networks, implying that the intrinsic biases of CNNs continue to be beneficial for tasks with a small downstream dataset, Section 4.2.1.

The overall findings are in accordance with previous research, where convolutional networks were successfully used to identify bacteria, [Smith et al., 2018], protozoa, [Mathison et al., 2020], and fungi, [Zieliński et al., 2020], in microscopic images. We show that the method can successfully be applied to fluorescent microscopy and that the computational complexity should not

pose a bottleneck to the automation, Section 4.6.

### Limitations

The study has several limitations. First, as mentioned previously, the results were obtained on a small dataset of 1244 images. Second, the models were trained and evaluated on images collected using a microscope designed for manual examinations. The models, however, are ultimately to be used with an automated slide scanner. Third, due to hardware limitations, only model architectures of limited size were trained and evaluated. The hardware also limited the training procedure; for example, training could only be done with batches of 10 samples. Fourth, the models were trained and evaluated on downscaled images converted to grayscale. However, it is possible that better performance could be attained with higher resolution, colored images.

### Future Research

There are three major research topics arising from the thesis that could be addressed in the future. First, the thesis focused on the model's performance on the available dataset. However, in the future, the model must be evaluated and potentially re-trained on data collected by an automated slide scanner.

Second, the thesis focused on the binary classification of a single image. However, when using an automated slide scanner, the number of images obtained from a single slide typically ranges from a hundred images to several thousand. Therefore, a methodology for aggregating classifications of the images to produce a classification of the entire slide must be developed.

Third, the effects of the novel data augmentations were evaluated with promising results when using Poisson augmentation of both (i) the positive samples, and (ii) the negative samples. The localization of the contaminant for positive samples was obtained by thresholding the images, a process specific to fluorescent microscopy. However, the localization was performed using Grad-CAM, [Selvaraju et al., 2017], for negative samples, a general method applicable to a wide range of datasets. It, therefore, remains to be explored whether (i) the positive samples can be successfully augmented using Grad-CAM instead of thresholding, and whether (ii) the augmentation is beneficial for a general detection task.

# Chapter 6

# Conclusion

In this work, we have presented the theoretical background for detecting filamentous fungi and yeast in microscopic images by deep neural networks. The thesis provides an intermediate step toward creating an automated system that would simplify the work of human operators. We trained state-of-the-art networks and demonstrated that the method achieves performance comparable with expert-level humans, despite the fact that annotations were only provided at the image level. We further showed that deployment of the model would significantly reduce the amount of manual labor associated with routine screenings for filamentous fungi and yeast.

# Bibliography

[Aakash Saboo and Wang, 2021] Aakash Saboo, Prashnna K. Gyawali, A. S. M. S. N. J. and Wang, L. (2021). Latent-optimization based disease-aware image editing for medical image augmentation.

[Ali et al., 2020] Ali, F., El-Sappagh, S., Islam, S. R., Kwak, D., Ali, A., Imran, M., and Kwak, K.-S. (2020). A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion. *Information Fusion*, 63:208–222.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929.

[Du et al., 2022] Du, X., Wang, Z., Cai, M., and Li, Y. (2022). Vos: Learning what you don't know by virtual outlier synthesis.

[Fong and Vedaldi, 2019] Fong, R. and Vedaldi, A. (2019). Occlusions for effective data augmentation in image classification.

[Frid-Adar et al., 2018] Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321:321–331.

[Gao et al., 2021] Gao, W., Li, M., Wu, R., Du, W., Zhang, S., Yin, S., Chen, Z., and Huang, H. (2021). The design and application of an automated microscope developed based on deep learning for fungal detection in dermatology. *Mycoses*, 64(3):245–251.

[Garcia et al., 2021] Garcia, E., Kundu, I., Kelly, M., and Soles, R. (2021). The American Society for Clinical Pathology 2020 Vacancy Survey of Medical Laboratories in the United States. *American Journal of Clinical Pathology*. aqab197.

[George, 2010] George, E. (2010). Occupational Hazard for Pathologists: Microscope Use and Musculoskeletal Disorders. *American Journal of Clinical Pathology*, 133(4):543–548.

[Goodfellow et al., 2014] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.

[He et al., 2016a] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks.

[ITU, 2011] ITU (2011). Recommendation itu-r bt.601-7 studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios bt series broadcasting service (television).

[Kokhlikyan et al., 2020] Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., and Reblitz-Richardson, O. (2020). Captum: A unified and generic model interpretability library for pytorch.

[Kolesnikov et al., 2020] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (bit): General visual representation learning. In *ECCV*.

[Koziarski et al., 2021] Koziarski, M., Cyganek, B., Olborski, B., Antosz, Z., Żydak, M., Kwolek, B., Wąsowicz, P., Bukała, A., Swadźba, J., and Sitkowski, P. (2021). Diagset: a dataset for prostate cancer histopathological image classification.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[Lai and Yu, 2021] Lai, C.-C. and Yu, W.-L. (2021). Covid-19 associated with pulmonary aspergillosis: A literature review. *Journal of Microbiology, Immunology and Infection*, 54(1):46–53.

[Mathison et al., 2020] Mathison, B. A., Kohan, J. L., Walker, J. F., Smith, R. B., Ardon, O., Couturier, M. R., and Pritt, B. S. (2020). Detection of intestinal protozoa in trichrome-stained stool specimens by use of a deep convolutional neural network. *Journal of Clinical Microbiology*, 58(6):e02053–19.

[Paplhám, 2020] Paplhám, J. (2020). Convolutional neural networks with local context masks.

[Pérez et al., 2003] Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318.

[Pham et al., 2021] Pham, H., Xie, Q., Dai, Z., and Le, Q. V. (2021). Meta pseudo labels. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11552–11563.

[Rotemberg et al., 2020] Rotemberg, V., Kurtansky, Betz-Stablein, Caffery, Chousakos, Codella, Combalia, Dusza, Guitera, and Gutman. (2020). A patient-centric dataset of images and metadata for identifying melanomas using clinical context.

[Selvaraju et al., 2017] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626.

[Smith et al., 2018] Smith, K. P., Kang, A. D., Kirby, J. E., and Bourbeau, P. (2018). Automated interpretation of blood culture gram stains by use of a deep convolutional neural network. *Journal of Clinical Microbiology*, 56(3):e01521–17.

[Stokes and Anderson, 1996] Stokes, M. and Anderson, M. (1996). A standard default color space for the internet - srgb.

[Szegedy et al., 2015] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision.

[Tan and Le, 2019] Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

[Tan and Le, 2021] Tan, M. and Le, Q. V. (2021). Efficientnetv2: Smaller models and faster training. *ArXiv*, abs/2104.00298.

[Tolstikhin et al., 2021] Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). Mlp-mixer: An all-mlp architecture for vision. *ArXiv*, abs/2105.01601.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Wang et al., 2017] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of

common thorax diseases. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Wei et al., 2017] Wei, Y., Feng, J., Liang, X., Cheng, M.-M., Zhao, Y., and Yan, S. (2017). Object region mining with adversarial erasing: A simple classification to semantic segmentation approach.

[Xie et al., 2020] Xie, Q., Hovy, E. H., Luong, M.-T., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

[Yen et al., 1995] Yen, J.-C., Chang, F.-J., and Chang, S. (1995). A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, 4(3):370–378.

[Yun et al., 2019] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features.

[Zhang et al., 2017] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization.

[Zieliński et al., 2017] Zieliński, B., Plichta, A., Misztal, K., Spurek, P., Brzychczy-Włoch, M., and Ochońska, D. (2017). Deep learning approach to bacterial colony classification. *PLOS ONE*, 12(9):1–14.

[Zieliński et al., 2020] Zieliński, B., Sroka-Oleksiak, A., Rymarczyk, D., Piekarczyk, A., and Brzychczy-Włoch, M. (2020). Deep learning approach to describe and classify fungi microscopic images. *PLOS ONE*, 15(6):1–16.

# Acronyms

**BiT** Big-Transfer, [Kolesnikov et al., 2020]. 20

**CNN** Convolutional Neural Network. 1, 2, 5–9, 19, 23, 39, 67

**DNA** Deoxyribonucleic acid. 8

**FPR** False positive rate. 16, 28, 29, 52, 54

**LR** Learning rate. 19, 20

**MLP** Multi-layer perceptron. 6, 40, 41

**SGD** Stochastic gradient descent. 19, 20, 32, 34–37, 40, 47, 49, 52, 55, 57

**ST** Saved time. 16, 17

**SVM** Support vector machine. 8, 9

**TPR** True positive rate. 16, 17, 28, 29, 32, 43, 50, 52, 54

**ViT** Vision Transformer. 6, 19, 40, 43–45, 62, 63, 66

# Glossary

**Automated slide scanner** Machine for automated slide imaging, producing high-resolution microscopic images without manual labor. 1, 7, 8

**Fluorescent stain** Staining method that uses fluorescent dye which binds to certain tissue components and will fluoresce upon irradiation with ultraviolet or violet-blue light. The staining process is non-specific, as a multitude of structures binds the dye. 2

**Gram stain** Staining method for differentiating between types of bacteria. Color is added to the sample then washed off. Bacteria with a thicker cell wall preserve the color even after washing. Such bacteria are called gram-positive. Bacteria that do not preserve the coloring are called gram-negative and are often recolored with a different color, to be easily identifiable under a microscope. 2

**Slide** A small, thin, flat rectangular piece of glass for mounting samples for microscopic study. 1, 12

**Trichrome stain** Staining method that uses acid dyes in conjunction with a polyacid for tinting tissues in contrasting colors. The increased contrast allows for easier detection of features in microscopy. 2, 8

# Appendix A

## Pretrained Model Weights

| Model | Weights |
|---|---|
| ResNet-18 | https://download.pytorch.org/models/resnet18-f37072fd.pth |
| ResNet-50 | https://download.pytorch.org/models/resnet50-0676ba61.pth |
| ResNet-50x1-V2 | https://storage.googleapis.com/bit_models/BiT-M-R50x1-ILSVRC2012.npz |
| EfficientNet-B0 | https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-weights/tf_efficientnet_b0_aa-827b6e33.pth |
| EfficientNet-B1 | https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-weights/tf_efficientnet_b1_aa-ea7a6ee0.pth |
| EfficientNet-B2 | https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-weights/tf_efficientnet_b2_aa-60c94f97.pth |
| EfficientNet-V2-S | https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-effv2-weights/tf_efficientnetv2_s_21k-6337ad01.pth |
| ViT-B-32-224 | https://storage.googleapis.com/vit_models/augreg/B_32-i21k-300ep-lr_0.001-aug_medium1-wd_0.03-do_0.0-sd_0.0--imagenet2012-steps_20k-lr_0.03-res_224.npz |
| ViT-B-32-384 | https://storage.googleapis.com/vit_models/augreg/B_32-i21k-300ep-lr_0.001-aug_light1-wd_0.1-do_0.0-sd_0.0--imagenet2012-steps_20k-lr_0.03-res_384.npz |

**Table A.1:** Pretrained model weights.

# Appendix B

## Module List

Included below is a list of modules used for all of the experiments. The order is identical to the order in which the modules were loaded.

1. GCCcore/8.3.0
2. binutils/2.32-GCCcore-8.3.0
3. GCC/8.3.0
4. CUDA/10.1.243
5. gcccuda/2019b
6. zlib/1.2.11-GCCcore-8.3.0
7. numactl/2.0.12-GCCcore-8.3.0
8. XZ/5.2.4-GCCcore-8.3.0
9. libxml2/2.9.9-GCCcore-8.3.0
10. libpciaccess/0.14-GCCcore-8.3.0
11. hwloc/1.11.12-GCCcore-8.3.0
12. OpenMPI/3.1.4-gcccuda-2019b
13. OpenBLAS/0.3.7-GCC-8.3.0
14. gompic/2019b
15. FFTW/3.3.8-gompic-2019b
16. ScaLAPACK/2.0.2-gompic-2019b
17. fosscuda/2019b
18. bzip2/1.0.8-GCCcore-8.3.0
19. ncurses/6.1-GCCcore-8.3.0
20. libreadline/8.0-GCCcore-8.3.0
21. Tcl/8.6.9-GCCcore-8.3.0
22. SQLite/3.29.0-GCCcore-8.3.0
23. GMP/6.1.2-GCCcore-8.3.0
24. libffi/3.2.1-GCCcore-8.3.0
25. Python/3.7.4-GCCcore-8.3.0
26. SciPy-bundle/2019.10-fosscuda-2019b-Python-3.7.4
27. libyaml/0.2.2-GCCcore-8.3.0
28. PyYAML/5.1.2-GCCcore-8.3.0
29. MPFR/4.0.2-GCCcore-8.3.0
30. NASM/2.14.02-GCCcore-8.3.0
31. x264/20190925-GCCcore-8.3.0
32. LAME/3.100-GCCcore-8.3.0
33. x265/3.2-GCCcore-8.3.0
34. expat/2.2.7-GCCcore-8.3.0
35. libpng/1.6.37-GCCcore-8.3.0
36. util-linux/2.34-GCCcore-8.3.0

37. fontconfig/2.13.1-GCCcore-8.3.0
38. X11/20190717-GCCcore-8.3.0
39. FriBidi/1.0.5-GCCcore-8.3.0
40. FFmpeg/4.2.1-GCCcore-8.3.0
41. gflags/2.2.2-GCCcore-8.3.0
42. libunwind/1.3.1-GCCcore-8.3.0
43. glog/0.4.0-GCCcore-8.3.0
44. libjpeg-turbo/2.0.3-GCCcore 8.3.0
45. LibTIFF/4.0.10-GCCcore-8.3.0
46. Pillow/6.2.1-GCCcore-8.3.0
47. cuDNN/7.6.4.38-gcccuda-2019b
48. NCCL/2.4.8-gcccuda-2019b
49. JasPer/2.0.14-GCCcore-8.3.0
50. Java/11.0.2
51. ant/1.10.7-Java-11
52. gettext/0.20.1-GCCcore-8.3.0
53. PCRE/8.43-GCCcore-8.3.0
54. GLib/2.62.0-GCCcore-8.3.0
55. ATK/2.32.0-GCCcore-8.3.0
56. DBus/1.13.12-GCCcore-8.3.0
57. at-spi2-core/2.32.0-GCCcore-8.3.0
58. at-spi2-atk/2.32.0-GCCcore-8.3.0
59. Gdk-Pixbuf/2.38.1-GCCcore-8.3.0
60. pixman/0.38.0-GCCcore-8.3.0
61. cairo/1.16.0-GCCcore-8.3.0
62. ICU/64.2-GCCcore-8.3.0
63. HarfBuzz/2.6.4-GCCcore-8.3.0
64. Pango/1.44.7-GCCcore-8.3.0
65. nettle/3.5.1-GCCcore-8.3.0
66. libdrm/2.4.99-GCCcore-8.3.0
67. LLVM/9.0.0-GCCcore-8.3.0
68. Mesa/19.1.7-GCCcore-8.3.0
69. libepoxy/1.5.3-GCCcore-8.3.0
70. GTK+/3.24.12-GCCcore-8.3.0
71. OpenCV/3.4.8
72. freetype/2.10.1-GCCcore-8.3.0
73. Pillow-SIMD/6.0.x.post0-GCCcore-8.3.0
74. Ninja/1.9.0-GCCcore-8.3.0
75. protobuf/3.10.0-GCCcore-8.3.0
76. protobuf-python/3.10.0-fosscuda-2019b-Python-3.7.4
77. pybind11/2.4.3-GCCcore-8.3.0-Python-3.7.4
78. typing-extensions/3.7.4.3-GCCcore-8.3.0-Python-3.7.4
79. magma/2.5.4-fosscuda-2019b
80. PyTorch/1.8.0-fosscuda-2019b-Python-3.7.4
81. torchvision/0.9

# MASTER‘S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Paplhám Jakub** Personal ID number: **474482**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

## II. Master’s thesis details

Master’s thesis title in English:

**Detection of Microscopic Fungi and Yeast in Clinical Samples**

Master’s thesis title in Czech:

**Detekce mikroskopických hub v klinickém materiálu**

Guidelines:

Infections caused by microscopic fungi and yeast occur rarely, however, they can cause serious health problems. For this reason, patients with risky factors undergo regular screening. A suitable method for the detection of fungi and yeast is based on fluorescence microscopy. At present, the microscopic images are examined manually by a trained specialist. The goal of this thesis is to explore the possibility of using deep neural networks to either replace the human operator completely or to significantly facilitate his work.

Bibliography / sources:

[1] B.Zielinksi at al. Deep learning approach to describe and classify fungi microscopic images. PLoS ONE 15(6), 2020.
[2] W. Gao et al. The design and application of an automated microscope developed based on deep learning for fungal detection in dermatology. Mycoses, 64(3):245-251. 2020.

Name and workplace of master’s thesis supervisor:

**Ing. Vojtěch Franc, Ph.D. Machine Learning FEE**

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **07.01.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

| Ing. Vojtěch Franc, Ph.D. | prof. Ing. Tomáš Svoboda, Ph.D. | prof. Mgr. Petr Páta, Ph.D. |
|---|---|---|
| Supervisor’s signature | Head of department’s signature | Dean’s signature |

## III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

| . | |
|---|---|
| Date of assignment receipt | Student’s signature |