



Practical Attack and Defense Methods for Integrity of Deep Neural Networks in Digital Pathology Image Analysis Systems

Markus Aukeala

Master's thesis

May 2024

Master's Degree Programme in Information Technology

Aukeala, Markus

Practical attack and defense methods for integrity of Deep Neural Networks in Digital Pathology Image Analysis systems

Jyväskylä: Jamk University of Applied Sciences, May 2024, 83 pages.

Master's Degree Programme in Information Technology, Cyber Security. Master's Thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

Digital pathology has made huge strides in development over the past decade. The introduction of new technology brings with it huge potential in efficiency, accuracy, and cost benefits, but also new risks. From the point of view of cyber security, in addition to traditional software, hardware and network security, a new risk will be attack attempts against artificial intelligence models and systems running the models.

The purpose of the thesis was to respond to the thesis commissioner's need to investigate practical options for detecting and preventing vulnerabilities in deep neural network models and their feasibility. In addition to the goodness of the detection models, things to consider were, e.g., performance, feasibility in practice and calculated cost/benefit ratio.

The thesis used design science as its research method, which aims to produce an artifact that solves the research problem with a practical and innovative solution. Scientific publications were used as source material for the work, both on the vulnerabilities related to neural networks in digital pathology, and on the vulnerabilities of deep learning neural networks in general in relation to image analysis. In the work, a deep learning neural network (convolutional autoencoder) was produced as an artifact, the purpose of which is to detect deviations from the input data.

Based on the results, with convolutional autoencoders it is possible to detect perturbations of even just a few pixels in the analyzed images and through this to detect a possible attack that tries to influence the result of the image analysis through a deviation of the input data. Implementing standard convolutional autoencoders is an easy way to start detecting deviations, but the disadvantage of an artificial intelligence model for detecting attacks taught with image data is its poor generalizability and the need to retrain the neural network model if significant changes in image quality or color scheme occur in the digital pathology image data due to hardware or software changes.

Keywords/tags (subjects)

Cybersecurity, Adversarial attack, Integrity, Convolutional Autoencoder, Healthcare Security

Miscellaneous (Confidential information)

-

Aukeala, Markus

Practical Attack and Defense Methods for Integrity of Deep Neural Networks in Digital Pathology Image Analysis Systems

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2024, 83 sivua.

Master's Degree Programme in Information Technology, Cyber Security. Opinnäytetyö YAMK.

Julkaisun kieli: englanti

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Digitaalinen patologia on ottanut valtavia kehitysaskelia viimeisen vuosikymmenen aikana. Uuden teknologian käyttöönotto tuo mukanaan valtavia potentiaalisia tehokkuus-, tarkkuus- ja kustannushyötyjä, mutta myös uusia riskejä. Kyberturvallisuuden näkökulmasta perinteisen ohjelmisto-, laitteisto- ja tietoverkkoturvallisuuden lisäksi uutena riskinä tulee tekoälymalleihin sekä niitä ajaviin järjestelmiin kohdistuvat muokkaus ja hyökkäysryitykset.

Opinnäytetyön tarkoituksena oli vastata toimeksiantajan tarpeeseen tutkia käytännöllisiä syväoppivien tekoälymallien haavoittuvuuksien havainnoinnin ja ehkäisyn vaihtoehtoja ja niiden toteutuskelpoisuutta. Havainnointimallien hyvyyden lisäksi huomioitavia asioita olivat mm. suorituskyky, toteutettavuus käytännössä ja laskennallinen kustannus/hyöty -suhde.

Opinnäytetyössä käytettiin tutkimusmenetelmänä suunnittelutiedettä, jossa pyritään tuottamaan artefakti, joka ratkaisee tutkimusongelma käytännöllisellä ja innovatiivisella ratkaisulla. Työn lähdemateriaalina käytettiin tieteellisiä julkaisuja sekä digitaalisen patologian neuroverkkoihin liittyvistä haavoittuvuuksista että yleisesti syväoppivien neuroverkkojen haavoittuvuuksista liittyen kuva-analyysiin. Työssä tuotettiin artefaktina syväoppiva neuroverkko (konvolutiivinen autoenkooderi) jonka tarkoituksena on havainnoida poikkeamia syötedatasta.

Tulosten perusteella konvolutiivisilla autoenkoodereilla on mahdollista havaita jopa vain muutamien pikselien kuvahäiriöitä analysoitavissa kuvissa ja tätä kautta havaita mahdollinen hyökkäys jossa yritetään vaikuttaa kuva-analyysin lopputulokseen syötedatan poikkeaman kautta. Tavallisten konvolutiivisten autoenkoodereiden toteuttaminen on helppo tapa aloittaa poikkeamien havaitseminen, mutta kuvadatalla opetetun hyökkäysten havaitsemiseen tarkoitettun tekoälymallin haittapuolinan on heikko yleistyyvyys ja tarve uudelleen kouluttaa neuroverkkomalli, jos digitaalisen patologian kuvadatassa tapahtuu laite -tai ohjelmistovaihdoksista johtuvia olennaisia muutoksia kuvalaadussa tai värimaailmassa.

Avainsanat (asiasanat)

Kyberturvallisuus, Neuroverkon huijaus, Muuttumattomuus, Konvolutiiviset autoenkooderit, Terveystietoturva

Muut tiedot (salassa pidettävät liitteet)

-

Contents

1	Introduction	4
1.1	Research question	5
1.2	Thesis commissioner and topic selection justification	5
1.3	Thesis topic delineation	6
1.4	Research methodology	7
1.5	Information retrieval and source material.....	10
1.6	Research reliability and ethics	11
2	Background and key concepts	14
2.1	Prior research and literature	14
2.2	Cybersecurity threats in Digital pathology image analysis	15
2.3	Deep Neural Networks attacks and defenses	20
3	Solution implementation workflow.....	28
3.1	Problem identification and motivation	28
3.2	Objectives of a solution.....	29
3.3	Design & development.....	30
3.3.1	Hardware and software requirements	31
3.3.2	Dataset preprocessing and splitting	33
3.3.3	Selecting attack target system and AI model	36
3.3.4	Generating adversarial images with Differential Evolution	37
3.3.5	Training convolutional autoencoder to detect anomalies	39
3.4	Demonstration of solution suitability	44
3.4.1	Selecting dataset for demonstration	45
3.4.2	Attack demonstration	45
3.4.3	Threshold used for attack detection	46
3.4.4	Defense demonstration	48
3.5	Evaluation of the implemented solution	49
3.5.1	Selecting dataset for evaluation	50
3.5.2	Attack evaluation	51
3.5.3	Defense evaluation	55
3.6	Communication of results and knowledge	59
4	Discussion.....	60
4.1	Autoencoder suitability for image anomaly detection	61
4.2	On the probability of attacks against Digital Pathology Image Analysis systems.....	62
4.3	Improvement ideas for adversarial image detection.....	63

5	Conclusion	65
6	References	67
7	Appendices	72
	Appendix 1. Parameters used in adversarial image generation	72
	Appendix 2. The demonstration environment setup instructions	73
	Appendix 3. Demonstration attack & defense results table.....	74
	Appendix 4. Evaluation attack & defense results table	77

Figures

Figure 1. The process model of Design Science Research Methodology (Peffer et al., 2007), modified.	7
Figure 2. Digital Pathology workflow (example).....	17
Figure 3. Adversarial attacks and risks related to machine learning pipeline (Finlayson & Beam, 2019, modified).....	20
Figure 4. Dataset structure after splitting into train, validation and test folders	35
Figure 5. Swagger API description and test user interface for codait/max-breast-cancer-mitosis-detector.....	37
Figure 6. Original image (left) and image reconstructed by the autoencoder(right) trained for solution.	44
Figure 7. Attack evaluation process	51
Figure 8. HTTP POST requests send towards target system	55
Figure 9. Defense evaluation process	56
Figure 10. Confusion matrix based on anomaly detection test set	58

Tables

Table 1. Attacker and Defender motives	18
Table 2. Hiding attack by selecting pixel color from existing color scale	21
Table 3. Blending pixel sized changes	22
Table 4. Selected adversarial attacks and their relation to medical domain	22
Table 5. Differences of black-box vs white-box attack in the context of adversarial attacks on images.	24
Table 6. Defense methods against Deep Neural Network attacks	26
Table 7. Countermeasures for adversarial examples (Yuan et al., 2017, modified)	27
Table 8. Original image and produced image patches.	34
Table 9. Dataset split command	35

Table 10. Sample image patches from the dataset	40
Table 11. Commands for neural network visualization	41
Table 12. Architecture of the Autoencoder used in solution	42
Table 13. Autoencoder model training & validation loss accuracy	43
Table 14. Attack efficiency samples	46
Table 15. Autoencoder reconstruction errors and detection efficiency with validation set	48
Table 16. Defense efficiency samples	49
Table 17. Summary of the performance of the implemented solution	50
Table 18. CODAIT Prediction change (mitosis-to-normal)	52
Table 19. CODAIT Prediction change (normal-to-mitosis)	53
Table 20. Metrics of the black-box attack againsts AI model behind REST API	54
Table 21. Autoencoder reconstruction errors (before-after attack) for evaluation dataset	56
Table 22. Structure of confusion matrix	57
Table 23. Classification report	59
Table 24. Probability for inference or training attack occurrence	62

1 Introduction

The digitalization of pathology has created a huge opportunity to improve the accuracy of clinical diagnoses and reduce the time it takes to analyze a whole slide image (WSI) (Parwani, 2019). It enables efficient storage and sharing of the samples gathered from patients over decades for educational and research purposes. On the research side the digitalized data can be used in training deep learning models that enable faster analysis, handle large datasets, reduce the annotation burden and predictive models capable of estimating patient outcome, disease progression likelihood or treatment response. This requires that the experts making AI (artificial intelligence) systems and labeling digitalized image samples can trust that the data used follows Information security triad (Stallings, 2019): Confidentiality, integrity, availability (CIA). The CIA-triad is a guideline for information security that aims to secure the systems in such a way that the system and the information in it is protected, trustworthy, not modified by any unauthorized party and available when needed in use.

From a cybersecurity perspective healthcare is a critical sector and must be protected well and with all reasonable security controls available, both technical and non-technical. At the same time cost-saving requirements in healthcare, new legal requirements, legacy system maintenance and collaboration with resources external to the hospitals closed networks bring new challenges to the overall cybersecurity and governance models. From cybersecurity criminals' perspective the healthcare sector has until now been a target for financial benefit (Finlayson et al., 2019) using ransomware and causing chaos among patients by attacking system availability through denial-of-service attacks. But the deployment of new intelligent systems will bring new attackers and attack types into play, ones that are clever, more difficult to detect by humans and that will also require another intelligent system to protect them from adversaries.

Currently, in clinical use, deep learning neural networks are used in an assisting role where the pathologist verifies the diagnosis produced by the artificial intelligence model (Parwani, 2019). So even if the risk for adversarial image attack is currently low and unlikely, the moment to build efficient detection and defense for these new kinds of attacks is now when there is still plenty of time to prepare for more advanced types of attacks. When clinical analysis workflow is more and more automated, there should be ways to detect deviations on workflow, input data and output result.

In this thesis, chapters 1 and 2 explain the design science research method, goals, literature, and background concepts to give an overview and explain the importance of the topic. The goal was to show how in practice an adversary could perform an attack and what kind of ways there exists to detect and defend those special kinds of tailored attacks. Chapter 3 follows design science research method solution implementation steps to explain the objectives, motivation, design, and actual implementation of the resulting artifacts. As a result, a convolutional autoencoder was produced (chapter 3.3.5) to detect adversarial perturbations. The latter part of chapter 3 covers the demonstration and evaluation of the implemented solution to show how well the resulting artifacts succeed in both the adversarial attack and defense side. Finally, chapters 4 and 5 cover the discussion and conclusion parts, also giving some ideas for further improvement of the initial solution.

1.1 Research question

The primary research question (RQ) can be defined as follows:

- RQ1: How in practice can one detect and defend against digital pathology image analysis AI model attack?

The answer to research question can be further divided into these detailed questions:

- RQ2: What are known threats and vulnerabilities to digital pathology image analysis AI model?
- RQ3: What are known adversarial AI attack techniques and types of attacks for image analysis?
- RQ4: What detection and defense methods exist against adversarial AI attacks for image analysis and how well do they work?

1.2 Thesis commissioner and topic selection justification

The commissioner (assigning company) for this thesis is Aiforia Technologies Oyj. The public website of Aiforia Technologies describes the company with following introduction: “Aiforia equips pathologists and scientists in preclinical and clinical labs with powerful deep-learning artificial intelligence software for translating images into discoveries, decisions, and diagnoses. The cloud based Aiforia products and services aim to escalate the efficiency and precision of medical image analysis beyond current capabilities across various fields, from oncology to neuroscience and more.” (Aiforia Technologies Oyj, 2023)

During the writing of this thesis, I have worked both in Lead Cloud architect and Cyber security engineer roles. In these roles the security of our company cloud-based image analysis system has been both an interest and duty. This thesis has been implemented using open-source AI platforms and tools. Aiforia's image analysis systems, data sets or AI models have not been used in the experiments presented in this thesis. Therefore, results or vulnerabilities do not directly apply to Aiforia products. I selected and agreed not to use Aiforia's system so that all the tools, code, methods, and results can be published as is, results are platform independent and can be reproduced using free tools and frameworks available. Aiforia has provided time, education and advice when needed to support my thesis writing.

From the early years of working with Aiforia I have been reading articles about theoretical attacks against image analysis and image recognition systems. Coming from a software developer background it seemed obvious that instead of seeking more methods for adversarial attacks, it was necessary to start looking for cost-effective, generic, and efficient methods to either detect the attacks and prevent them or make AI models more tolerant against attacks. Therefore, the selected research method for this thesis is a design science approach which aims to produce concrete outcomes.

1.3 Thesis topic delineation

The topic of this thesis has been delineated as Practical Attack and Defense Methods for Integrity of Deep Neural Networks in the field of digital pathology image analysis. While there is a lot of exciting adversarial attack research done in other fields such as traffic sign recognition (Eykholt et al., 2018) with autonomous vehicles, this thesis focuses on medical imaging and specifically on digital pathology.

From different adversarial attack methods, the scope of this thesis is limited to few-pixel modifications that allow attacker to modify the image data in ways that are hard to detect by digital pathology experts and therefore more likely to pass detection and defenses.

This thesis focuses on attacking an existing, published model that produces only single numerical result (mitosis probability in the image) as output. However, it is important to notice that many digital pathology AI models for cancer detection give more information than just Boolean

“Yes/No” or single numerical result as output. The output is often larger graphical and numeric result set indicating the important finding areas and severity. There is also research done about fooling the image segmentation results such as Metzen et al., (2017) where the authors were able to create adversarial perturbations producing desired target segmentation and therefore fooling image segmentation model not to recognize pedestrians from given landscape pictures.

For the detection and defense, a method to detect attack using deep neural network was selected with the idea of creating separate subsystem that could be attached to existing digital pathology image analysis system. This detection method is focused on detecting adversarial perturbations on image tiles (of a larger image), but adversary attacks against image segmentation models can also use same or similar methods on detecting attacks.

1.4 Research methodology

The research method used will be design science-based research method (Peffer et al., 2007). The reason to use Design Science Research Methodology (DSRM) is that it fits well for solution implementation in the information system research area. DSRM provides a structured approach for developing and evaluating technological solutions and is also focused on addressing practical real-world problems and creating innovative solutions and knowledge as outcome. This thesis focuses on developing knowledge about what attacks Medical Deep Learning AI models could be facing and design practical software-based solutions to defend against those attacks.

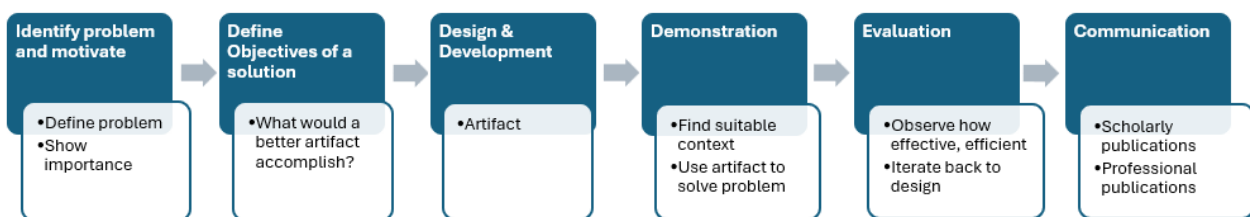


Figure 1. The process model of Design Science Research Methodology (Peffer et al., 2007), modified.

Design science-based research method aim is to prove the usefulness of the design with concrete sample solution or system. The six different steps of DSRM (Figure 1) are aligned with this thesis as follows.

Identify problem and motivate

The process starts with identifying some information system or technology related problem or opportunity that needs to be addressed with a solution. The motivation needs to be clear so that both the researcher and the audience of this research can understand the value of the artifact produced by this research would bring (Peppers et al., 2007).

For this thesis the chapter “3.1 Problem identification and motivation” describes the main motivation for this research is to help preventing damage to patient security by detecting adversarial attack of digital pathology image analysis AI model using a separate trained AI model for anomaly detection on model input images. Chapter “2 Background and key concepts” covers the background information on the subject and prior research done on the subject to identify possible earlier attempts to come up with a solution to the identified problem. It also introduces the audience to the key concepts related to the cybersecurity threats of an AI model.

Define objectives of a solution

The second step of the DSRM process is to determine what specific goals and objectives the research project is trying to achieve with the proposed solution. Also, the scope should be delineated so that boundaries and constraints are clear (Peppers et al., 2007).

For this thesis the chapter “3.2 Objectives of a solution” describes what are the expected outcomes and sets quantifiable objectives for the solution to be evaluated later. These rational expectations will serve as design and evaluation criteria for the solution.

Design & development

In this third step the actual solution artifact is being developed to solve the problem. Usually, this step is iterative as it involves trying different technical approaches to solve the problem and experiments with various design options. This step applies the theoretical knowledge gained in earlier steps to the actual problem solving (Peppers et al., 2007).

As this thesis develops a software artifact to solve the identified problem, the chapter “3.3 Design & development” initially covers hardware and software used in the artifact implementation and then introduces the dataset preparation, attack sample creation (for testing the artifact) and implementation of the actual solution artifact, the AI model to be used defending the attack input from reaching the target system.

Demonstration

The demonstration step shows in practice how the developed artifact is used in action to solve the identified problem in form of prototype, experimentation, simulation, or such activity. This step also demonstrates knowledge of the researcher on how to correctly use the artifact (Peppers et al., 2007).

In this thesis the chapter “3.4 Demonstration of solution suitability” contains a demonstration of both successful attack and sufficient defense (or detection) of an adversarial attack against the selected digital pathology image analysis AI model.

Evaluation

The goal of the evaluation step is to determine the effectiveness and practicality of the solution. It measures how well the implemented artifact supports an effective solution to the earlier identified problem. Results from the demonstration may be used in evaluating the solution. This step also offers researchers a possibility to jump back to design & development step to improve the solution artifact to gain better results. An acceptable option is also to communicate on the results (as is) and leave the identified problem to be solved better in future projects. Sometimes a feasible solution cannot be developed or solving the problem needs a different solution approach (Peppers et al., 2007).

For this thesis the chapter “3.5 Evaluation of the implemented solution” the results from the demonstration are evaluated quantitatively and qualitatively along with additional measures against larger set of testing data from various angles. Both attack and defense (or detection) results and performance are evaluated.

Communication

Sharing the insights acquired through the research, underlining the significance and novelty of the created solution artifact to both academia and industry professionals, serves as a catalyst for knowledge dissemination. It enables fellow researchers to gain knowledge, enhance existing methodologies, pose new questions, or expand upon findings. Such collaborative efforts are invaluable for advancing the field of study (Peppers et al., 2007).

Along with the thesis itself, in the last DSRM process step, a set of artifacts are produced and published along with evaluation results and knowledge gained through researching the topic.

1.5 Information retrieval and source material

Information retrieved using search keywords and scientific research databases.

General information and initial searches for information were gathered from JAMK E-library systems Janet Finna International article search (<https://janet.finna.fi/>), arXiv (<https://arxiv.org/>), ResearchGate (<https://www.researchgate.net/>), ProQuest Ebook Central (<https://ebookcentral.proquest.com/>). Some of the search patterns used in information queries in Janet Finna and ProQuest Ebook Central:

- (attack* OR defen*) "medical deep learning"
- ("attack*" OR "defen*") "medical deep learning systems"
- "cybersecurity" AND "threat" AND "digital pathology" AND "image analysis"
- "one-pixel attack" AND "deep learning"
- "one-pixel attack"
- "one-pixel attack" AND "medical deep learning systems"
- "few-pixel attack"
- "few-pixel attack" AND "deep learning"
- "adversarial" AND "attack" AND "black-box"
- "Deep learning" AND ("attack" OR "defense") AND "adversary" AND "image analysis" AND "medical"
- "adversarial attack" AND "medical" AND "deep learning"
- "adversarial" AND "defense" AND "Digital Pathology" AND "deep learning" AND "image analysis"
- "adversarial attacks" "medical deep learning systems"

Following the references in some of the key research papers and articles such as Finlayson & Beam, (2019), Su et al., (2017), Chakraborty et al., (2018), gave good basis for starting to develop an initial solution implementation from both attack and defense point of view.

Information retrieved from Finnish sources related to the topic

In JAMK, there was already a research group studying model fooling attacks against medical imaging so the articles published by Sipola et al., (2020), Korpiahkola et al., (2020) gave more detailed information on the type of attacks that would be suitable for this thesis. As that research group

was focused on attacks against Digital Pathology images, the information and references were a good starting point for finding key research groups from outside Finland and articles focusing on same topics.

Programming references and source codes

For the implementation of the few-pixel attack with Differential Evolution and training of the Convolutional Autoencoder, several general programming references and reference implementations were studied. For the programming references, the main sources were:

- Python `scipy.optimize.differential_evolution` documentation: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html
- Keras reimplementation and tutorial of "One pixel attack for fooling deep neural networks" (Su et al., 2017) by Dan Kondratyuk: <https://github.com/Hyperparticle/one-pixel-attack-keras/>
- Building Autoencoders in Keras: <https://blog.keras.io/building-autoencoders-in-keras.html>
- CODAIT/deep-histopath: <https://github.com/CODAIT/deep-histopath>

Other information used in thesis

For Digital Pathology overview, evaluation, history, terminology and current state, the articles of Pantanowitz et al., (2018) and Parwani, (2019) were helpful. The only physical book that is referenced and that was used in this thesis for general Cybersecurity knowledge and terminology was (Stallings, 2019).

1.6 Research reliability and ethics

Reliability and suitability of data

Results are based on tests done with image tiles from the original TUPAC16 dataset (Veta & et al., 2024) which contains a subset breast cancer case images from The Cancer Genome Atlas (TCGA). TCGA is a well-known and widely used data set in digital pathology research. The TUPAC16 dataset

is available from Grand Challenge web site (<https://grand-challenge.org/>). Users of the dataset are encouraged to reference the challenge overview paper (Veta et al., 2019) and that recommendation is followed in this thesis.

Image tiles selected for training, validation, testing, demonstration, and evaluation steps represent quite well common H&E-stained image samples in the data set. The presence of mostly white image tiles affects anomaly detection results for normal samples (no mitosis) but is a common feature for digitalized WSI images. Those usually contain a lot of empty background color.

Data confidentiality, consistency, and quality

Data quality will be controlled using well-known image data sources referenced in several scientifically reviewed articles. Data is anonymized and does not contain any identifiable patient information. Data will be stored under safe, encrypted source control in personal encrypted hard drive, JAMK Office 365 and private GitHub environments and will be destroyed after thesis is completed. Access to the data will be limited to authorized persons only.

Reliability of experimental setup

The experimental setup in this thesis is realistic in a sense that digitalized WSI images are so large (the size can exceed 50,000 by 50,000 pixels) that splitting the image into smaller tiles (64X64 pixels in this thesis) for image analysis and viewing purposes is efficient and rational choice of implementation. Therefore, attacking individual image tiles will affect breast cancer prediction results when certain regions of interest (ROI) are selected for analysis. In practice one would attack several image tiles at once to cause larger effect in ROI analysis results.

Reliability and generalization of findings

Findings presented in this thesis give specific answers to the research question using the selected solution implementation and align directly with the topic.

If the AI anomaly detection model is used with other digital pathology WSI image staining models than H&E or with different data set created with a different WSI scanner (as example), the model must be trained again with samples from the new data set included. This is because color differences and other variation in the digitalized WSI data affect the model performance and may result in a situation where perfectly valid breast cancer image tile sample is considered as anomaly.

The AI anomaly detection model (Convolutional autoencoder) implemented in this thesis work has not been tested with other breast cancer data sets to see how it generalizes.

Research Ethics

This master's thesis adheres to ethical research principles and guidelines (JAMK, 2018). Research is conducted in an ethical manner, with no intentional harm or damage being inflicted on any parties.

Research topic is selected so that it may reveal weaknesses in critical healthcare systems, but the aim is to make image analysis AI model producers and users aware of the vulnerabilities and therefore enable mitigating of vulnerabilities in early stage. Vulnerabilities listed in this thesis have already been discovered by other researchers and are known to the research community.

The digitized whole slide image data set used in model training is freely available, anonymized and does not contain patient data. Programming tools used are open source licensed and freely available. Results from test runs are reported honestly "as is" so that the validation of working solution can be repeated by other people interested in applying the public source code and results into their own solutions. Results can be reproduced using the same setup and parameters as described in this thesis.

Python code produced in this research is implemented by thesis author, except open-source software libraries and tools used. There are 1-2 efficient Python code methods referenced in the code that were originally implemented by other authors. Those code blocks are clearly referenced in code documentation with comments and link to original code. Code files used in solution implementation can be found from this thesis authors public GitHub library (https://github.com/amarkus/thesis_jamk). Code is available under MIT license.

Thesis is written by the author and has gone through the Turnitin Originality Check plagiarism detection software scanning. Author of this thesis has followed Jyväskylä University of Applied Science's ethical principles (JAMK, 2018).

2 Background and key concepts

2.1 Prior research and literature

When thesis writing begun in February 2021, Senior Lecturer of JAMK University of Applied Sciences was contacted to get approval to proceed with the thesis subject and to get thesis supervisor. It was surprise to find out that inside JAMK there was already a research group studying model fooling attacks related to medical imaging (Sipola et al., 2020). They were using breast cancer whole slide images and deep learning models as their targets, so it was the best possible start to get a thesis supervisor who was already familiar with the subject. Thesis supervisor also informed about the latest relevant research articles that could be useful knowledge for this thesis implementation. At the time thesis writing started, JAMK research group did not have any publications related to detection or defending AI model fooling attacks, so that confirmed my idea of the need for detection and defense methods.

The selected approach for this thesis is based on older Convolutional Autoencoder and straightforward to implement with a dataset with just enough data representing the images to be analyzed. In this thesis the selected attack method is few-pixel attack. It adds only a few pixels (5 pixels in this thesis) perturbation to the target image but If pixel colors are close to original image, the perturbation can be hard to detect by human. That should cause more false results in the targeted AI model and often result attack success, compared to for example One-Pixel attack as more image data is tampered (Su et al., 2017). This approach and Alatalo et al., (2022) aim to detect the attack enabling image analysis system administrator or the person responsible for the data set to act upon findings and perform further analysis of the integrity of image data and the model created using the data set.

During thesis writing, JAMK research group published an article titled “Detecting One-Pixel Attacks Using Variational Autoencoders” (Alatalo et al., 2022) which targeted finding an efficient way to detect specific attack called One-Pixel attack with promising results. The Variational Autoencoder approach (Alatalo et al., 2022) is based on technological improvements on the basic Convolutional Autoencoder but requires more knowledge for the implementer to find out the right optimization model.

Outside JAMK University of Applied Sciences, there have been several research groups researching adversarial attacks against medical deep learning systems. In the area of medical imaging, a group of researchers (Finlayson et al., 2019) performed adversarial patch attacks using Cleverhans (Papernot, Faghri, et al., 2016) library against medical classification models for fundoscopy, chest X-Ray and dermoscopy images developed by their own team.

PACS (Picture archiving and communication systems) and medical imaging security has been researched by (Eichelberg et al., 2020) and closely relates to image format security (DICOM, TIFF) and the way digital pathology images are stored and transferred during the workflow. Weak PACS cybersecurity protection enables attackers to access and tamper training and/or analysis data.

Recently there have been more research articles comparing convolutional neural networks (CNNs) and vision transformers (ViTs) on their robustness against white-box and black-box attacks. One such paper is (Ghaffari Laleh et al., 2022) where researchers find vision transformers to be more robust against several gradient-based adversarial attacks than convolutional neural networks.

2.2 Cybersecurity threats in Digital pathology image analysis

In this chapter the main areas of digital pathology workflow and image analysis are covered to give the reader an overview related to research questions RQ2, RQ3 and RQ4 in chapter 1.1: What are the threats and vulnerabilities present in the image analysis process, what kind of adversarial attacks can be used to attack the AI models used in digital pathology image analysis and how can one defend against the attacks in practice?

Digitalization of pathology

Digital pathology has been around for over 20 years now but the deployment of even the basic elements such as creating a whole slide imaging (WSI) workflow are an ongoing task even in some of the world's biggest and well-known hospitals. This is mostly because of the cost of setting up WSI workflow with scanning devices and because over the decades hospitals and research labs have collected tens of thousands (even hundreds of thousands) of glass slides so there is a lot to be scanned into digital formats. Glass slides have been earlier viewed by using common microscope or with different variations of "Virtual Microscopes", earliest developed in 1996-1998 (Pan-

tanowitz et al., 2018). Digitalization of vast amounts of glass slides is a huge task but it enables efficient storage, retrieval and sharing of the pathology samples for educational and research purposes. This data (digital WSI images) is applicable for training machine learning models capable of assisting pathologists in the making of accurate diagnoses by performing different image analysis tasks. These image analysis tasks, such as classification, segmentation, scoring and counting of objects can be done in matter of minutes or seconds and save pathologist's time. When robust AI image analysis solution has been created, it can be scaled to analyze hundreds of images at the same time.

Training deep learning models for use in digital pathology requires a well-chosen data set and an expert to label the interesting areas, objects, or images so that the resulting deep neural network (DNN) learns to recognize key features from the data set. When the model has been thoroughly tested by experts using relevant test data, the DNN model is then used in daily image analysis tasks to identify and outline regions of interest from pathological images. Such regions of interest can be, for example, tumor areas.

Digital pathology workflow (see in Figure 2) contains many phases for attacker to manipulate the data in such way that the attack scenario described in the solution implementation part of this thesis will be possible.

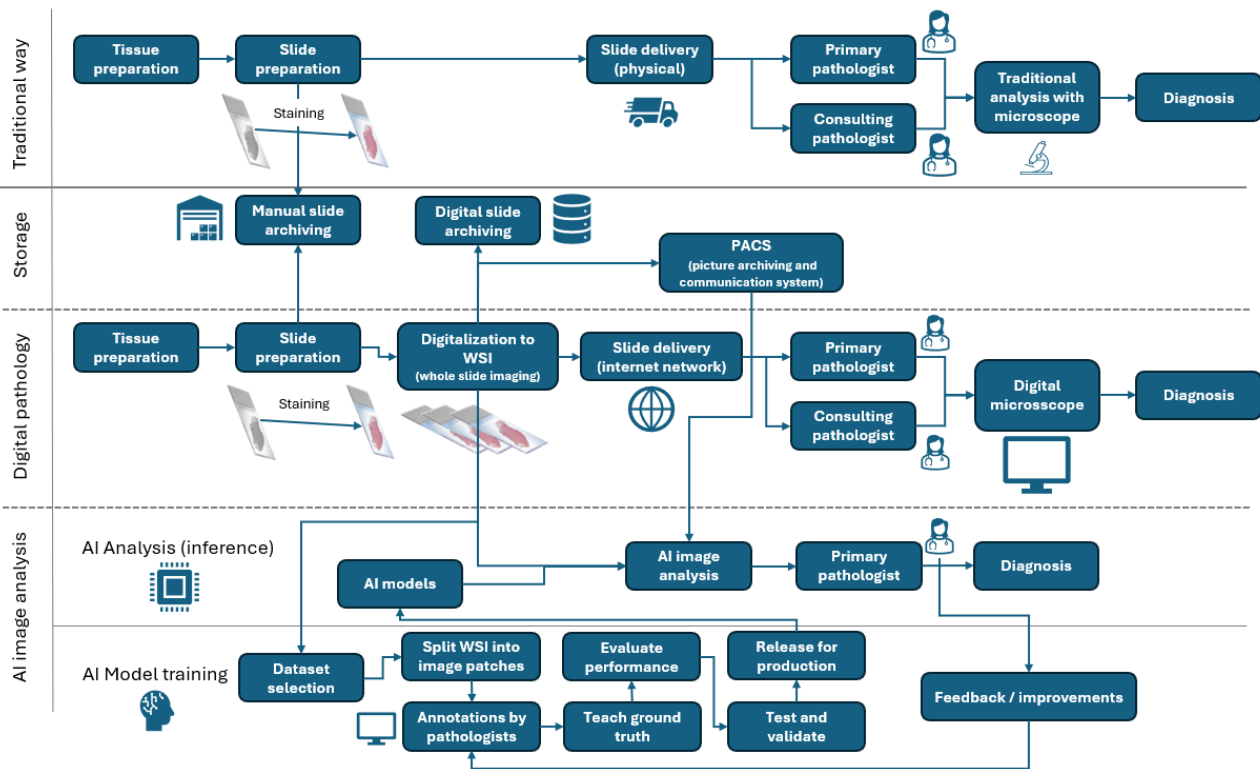


Figure 2. Digital Pathology workflow (example)

Legacy infrastructure and security misconfigurations

While hospital networks and endpoints (such as desktops and mobile devices) are usually secured well from external threats, it is common that there are many existing vulnerabilities in the hospitals internal systems (Finlayson & Beam, 2019) as some of those are legacy systems, monolithic and hard to update. Also, the updating of regulated systems requires long test cycles to be done before updates or new software/hardware is taken into production use. When updating hardware and software is hard and requires a lot of time for planning and deployment, there are usually plenty of opportunities for cybercriminals to penetrate the system.

Legacy infrastructure may effect on the information security of digitalized pathology images, revealing one or several points of attackers to tamper with images used in AI training of analysis phases.

Motives of attacker and defender

Motives of the attacker may vary by the application area of image analysis, mostly focusing on financial gain and causing harm. Table 1. lists motives for both the attacker and defender of digital

pathology deep learning networks. Some of the financial motives for the attacker are covered in (Finlayson et al., 2019).

Table 1. Attacker and Defender motives

Attacker motives	Defender motives
Financial Gain: Getting high-quality healthcare	Avoiding of unnecessary use of resources and costs for treatments
Financial Gain: Accepting treatment that is not necessarily needed	Avoiding of unnecessary use of resources and costs for treatments
Financial Gain: Insurance fraud	Avoiding of unnecessary costs
Financial Gain, Data Manipulation: Getting pharmaceutical and device approvals through clinical trials by result manipulation	Preventing damage to patients and manipulation of results
Financial Gain, Data Manipulation: Drug abuse for getting recipe	Preventing drug abuse and costs
Extortion and Ransom: Manipulating results and requesting money for correct AI model predictions.	Preventing damage to patients
Cyber Warfare, Hacktivism, Disruption and Sabotage, Data Manipulation: Causing harm to one patient or a group of patients	Preventing damage to patients
Competitive Advantage: Winning a commercial tender by manipulating AI model evaluation results	Preventing manipulation for competitive advantage

The biggest motive for cybercriminals to attack against machine learning algorithms used in assistance with medical decisions currently are the possible financial gains. As healthcare is trillion-dollar business in US alone (Finlayson & Beam, 2019) and the number of adoptions of AI as part of medical reimbursement decisions, drug decisions, pharmaceutical and device approvals is growing, opportunity for fraud and extortion/ransom is considerable.

Measuring the possible threat impact and exploitability

In cybersecurity there is a commonly used scoring system which is called the Common Vulnerability Scoring System, CVSS (FIRST.Org, 2024) which is used to score exploitability and impact metrics so that teams responding to threats can prioritize responses and resources according to threat.

When many threats and vulnerabilities exist and multiple attackers are trying to discover vulnerabilities in hospital systems as well as pharmaceutical research laboratories, the cybersecurity teams must focus on the most relevant common threats first.

CVSS system is not meant to score AI/ML model vulnerabilities, but rather the computer systems in which they are run. When thinking about the exploitability and impact of Digital Pathology image analysis system inference attack performed with perturbed input images in the field of digital pathology image analysis, then using CVSS criteria we could get these kinds of characteristics:

- **Exploitability Metrics**
 - **Attack Vector:** Network/Adjacent Network/Local.
 - Depending on the implementation, parts of the image analysis system are accessible through network (or adjacent network.). Sometimes physical access to the local network where the system is running is required.
 - **Attack Complexity:** High.
 - Attack requires considerable amount of effort in both planning and executing the attack.
 - **Privileges Required:** High. Significant privileges are required over the vulnerable component to affect the files and settings.
 - **User Interaction:** Required.
 - For the vulnerability to be exploited, user must select an image containing adversarial perturbations as input for the image analysis.

- **Vulnerable System Impact Metrics**
 - **Confidentiality:** Low.
 - Ability to manipulate image analysis system results affects confidentiality of the system enabling attacker to gain control of some or all the predictions produced by the system.
 - **Integrity:** High.
 - Model behaves incorrectly due to tampering of data. Some or all inputs may be miscategorized and AI model's behavior is not trustworthy anymore.

- **Availability:** Low/High.
 - Vulnerability does not necessarily affect system availability over the network but affects results availability for pathologists not being able to get expected results from the system either partially or completely.

With these parameter values, the CVSS v3.1 calculator (<https://www.first.org/cvss/calculator/4.0>) would give CVSS Base Score between 5.0 - 7.0 and therefore rated as “Medium”. Generally greater focus and prioritization in resolving threats goes into categories “Critical” and “High”, so AI model vulnerability would not most likely result in immediate action or remediation. If the possibility of AI result tampering would seem likely or high-likely, then in hospital environment the temporary solution would be to perform analysis manually by the pathologists and temporarily bypass vulnerable image analysis system.

2.3 Deep Neural Networks attacks and defenses

In the paper (Finlayson & Beam, 2019) authors go through some of the motives and methods that could make adversarial attacks to be realized also in the field of medical deep learning systems. They also describe the risks related to different stages of machine learning pipeline (Figure 3.).

Adversarial attack risks in machine learning pipeline					
ML pipeline stage	Training data	Training	Model	Deployment	Results
Risks	<ul style="list-style-type: none"> • Privacy breaches • Data Poisoning • Data Bias • Label Leakage • Label Misspecification 	<ul style="list-style-type: none"> • Improper training • Incomplete training 	<ul style="list-style-type: none"> • Privacy Breaches 	<ul style="list-style-type: none"> • System Disruption • IT Downtime • Dataset shift 	<ul style="list-style-type: none"> • Model Stealing • Model error • Misinterpretation • Job displacement

Figure 3. Adversarial attacks and risks related to machine learning pipeline (Finlayson & Beam, 2019, modified).

Various attacks can be performed during different phases of digital pathology workflow, causing risks to realize as privacy breach, system downtime, loss of trade secrets (model internals), misinterpretation of results, error in results, or even leading to a wrong clinical diagnosis.

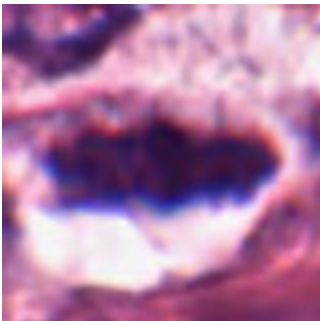
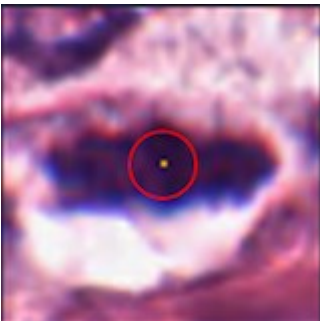
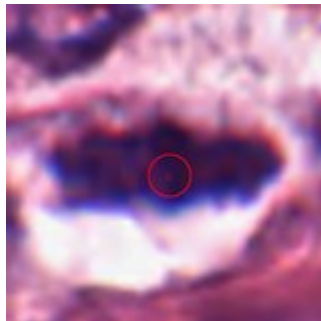
Attack methods related to digital pathology image analysis

There are several surveys and research articles published focusing on adversarial attacks against machine learning models analyzing images, sound, and text such as (Chakraborty et al., 2018) and (Xu et al., 2019). Attack methods often vary by the content type being analyzed. In this chapter, some of the most relevant attacks against digital image analysis are covered. The focus on selected attacks is on few-pixel perturbations and digital pathology, because in the field of digital pathology where pathologists are viewing very large images, some with tens of gigabytes in size and having over 10 gigapixels of information, cleverly made perturbations of carefully selected pixels will most likely be left unnoticed.

The adversarial perturbations inserted into the image can be significant in size (adversarial patch) or change of colors and usually that will result in desired results faster than minor change of pixels both in percentage of area and color. But it has been shown that even carefully selected one-pixel change that blends into its surrounding will cause enough change that the attack can succeed and resulting change is hard if not impossible to be noticed by trained human specialist (Korpihalkola et al., 2021) .

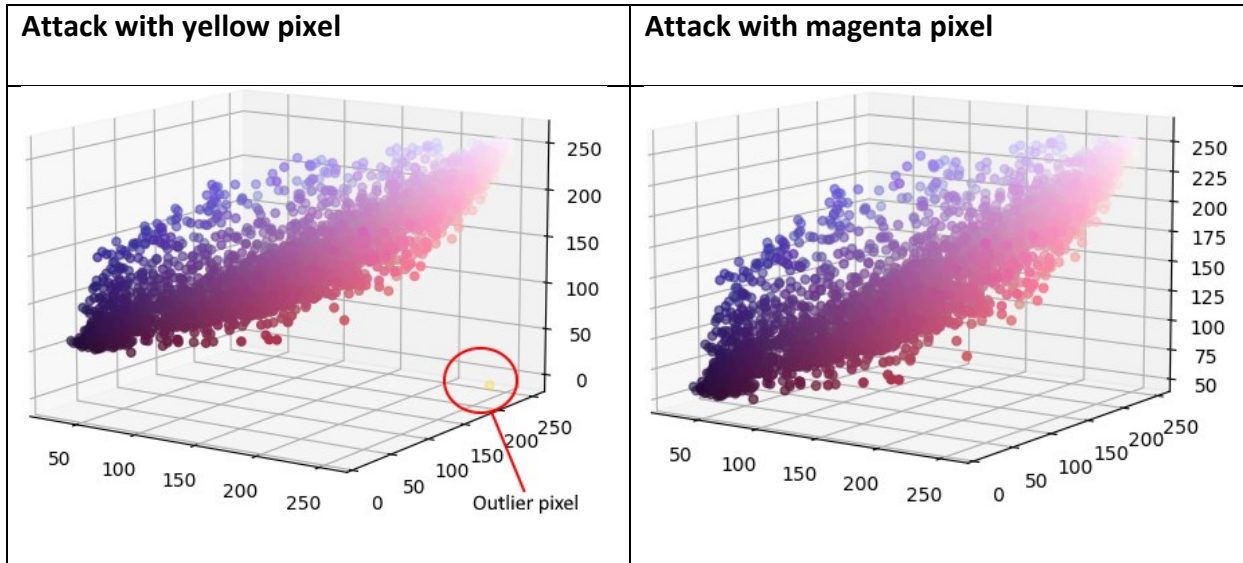
Table 2. Shows a sample of normal image and 2 attacked image, latter image having the attack pixel selected so that it is hard to spot by even an expert, considering the large amount of image data each pathologist must go through.

Table 2. Hiding attack by selecting pixel color from existing color scale

Image patch (mitosis_1.png) "probability": 0.03698	Attack with yellow pixel "probability": 0.014664	Attack with magenta pixel "probability": 0.02031
		

One-pixel sized anomalies are also more difficult to find when analyzing single pixel color deviations from a larger image. A very simple visualization is showed in Table 3. but generally, a more thorough analysis about pixels and their expected location in the image is needed to determine if any anomalies are occurring. This is what neural networks learn during their training iterations.

Table 3. Blending pixel sized changes



For medical domain there are research articles covering attacks based on adversarial images on radiology as well as digital pathology. The difference between those two is that radiology images are often smaller in size and color variation. Digital pathology images may go through different kinds of staining process, leading to several variations in digitalized image pixels even using the same source WSI slide.

Selected attacks, their efficiency and implementation related to few-pixel, or one-pixel perturbations are covered in the following publications shown in Table 4., where some of the attacks are directly related to medical domain.

Table 4. Selected adversarial attacks and their relation to medical domain

Paper	Black-box / white-box attack	Medical domain?	Few-pixel, one-pixel, or other type of attack?
Su et al., (2017)	"semi-black-box attack" (using only probability labels as information).	No.	One-pixel attack
Korpihalkola et al., (2020)	Black-box attack. DNN network queried using a REST based API.	Yes. Digital Pathology.	One-pixel attack

Finlayson & Beam, (2019)	Both. For black-box attack, results from independently trained network with similar architecture were used (transferred) in attack against victim.	Yes. Digital Pathology. Radiology.	Adversarial patch, Projected Gradient Descent (PGD) attack
Nguyen-Son et al., (2021)	Black-box attack.	No.	One-pixel attack, Few-pixel attack
Paschali et al., (2018)	Black-box attack.	Yes. Digital Pathology. Radiology.	Fast Gradient Sign Method (FGSM), DeepFool and Saliency Map Attacks
Sorin et al., (2023)	Black-box attack. White-box attack.	Yes. Digital Pathology. Radiology.	Multiple attacks (this paper is survey with references to actual papers on implementations)
Narodytska & Kasisviswanathan, (2017)	Black-box attack.	No.	Few-pixel attack
Quan et al., (2021)	"semi-black-box attack" (using only probability labels as information).		One-pixel attack, Few-pixel attack

Attack strategy and prior knowledge about the target network

This thesis uses black-box attack in testing how AI model behaves under attack. The distinction of Black-Box and White-Box attacks is closely related to attack strategy and prior knowledge that adversary has about the target. This distinction is like the types that general software and security

testing uses. If an adversary can generate a successful attack with very little information, then the attack is considered more effective. Often black-box attack requires gathering information about model by testing the system using the targeted AI model. For commercial REST API interfaces that are billed per usage, optimizing cost of the attack is also a success criterion.

Table 5. describes the differences between these two. In an article of (Chakraborty et al., 2018), the authors have given detailed information about Black-box and White-box attacks and their subclasses, such as Strict, Non-Adaptive and Adaptive Black-Box Attack.

Table 5. Differences of black-box vs white-box attack in the context of adversarial attacks on images.

	Black-box attack	White-box attack
Knowledge required to perform attack	The attacker has restricted or no access at all to the target model. They might not know the model's architecture, parameters, or have access to its internal workings.	The attacker possesses complete access to all aspects of the target model, including its structure, parameters, and the data it was trained on. This means the attacker knows how the model works.
Attack strategy	Attackers rely on observations and interactions with the model, such as submitting input images and observing the corresponding outputs, to generate adversarial examples. Techniques like transferability, where adversarial examples crafted for one model can fool another model, are often used in black-box attacks.	Attacker may use gradient information to craft perturbations that are specifically designed to deceive the model.

Some literature and research papers mention also attack strategy known as Semi-white (Gray) Box Attack. It is defined as when attacker has some prior knowledge (limited knowledge) about target AI model but can transfer that knowledge later in some form to be used in black-box attack. This is described shortly in (Xu et al., 2019). Transfer can happen for example by training a generative network in a white-box setup and then later using the generative network as a tool for adversarial attack.

Adversarial samples and Transferability

Transferability is an important feature of adversarial attacks against machine learning models. Many attacks that are shown to succeed for one AI model, also succeed on another model even if those models have different architecture as shown by (Papernot, McDaniel, et al., 2016) . This makes it easier for an attacker to mimic the behavior of a target AI model locally and find such an attack that will most likely work also on target model and architecture. In such Black-box attack scenario, the advantage is on the attacker side as there is no need to use brute-force methods against target API to find the weakness. This allows the attacker to remain undetected as the behavior to perform the attack does not differ from legitimate use of target API or software. This is also a financial factor as calls to machine learning REST APIs may have per request costs and therefore brute-force approach might result in high costs per attack session, with one session having thousands or tens of thousands of requests.

To succeed in training local target AI model of their own, the attacker must have knowledge of the domain to succeed. In the medical domain this is more difficult as training a predicting AI model for breast cancer proliferation score requires pathology expertise which cybercriminals rarely have. However, there are freely available AI models and AI training tutorials for multiple frameworks that may give some medical domain knowledge to train a model like the one used in target system.

Defense methods and strategies

To be able to defend Deep Neural Network based AI models trained for image analysis from being attacked, different methods to prevent the effects of adversarial input and detecting those inputs have been developed. In this chapter, content of three articles Yuan et al., (2017), Xu et al., (2019) and Chakraborty et al., (2018) are covered to get an overview of the different defense types and categories. Each defense type has its benefits and pitfalls in actual production use with digital pathology image analysis AI models. Digital pathology is a highly regulated area when resulting applications and systems are used in clinical diagnostics or patient care. This often limits the possibility to use some methods that alter the input data, even if the intention behind those methods was good.

Some of the well-known defense methods are introduced in Table 6 .

Table 6. Defense methods against Deep Neural Network attacks

Defense methods	How it works?
Adversarial Training	Works by increasing model robustness by augmenting training set with adversarial examples. This defense works best on adversarial samples made using the original AI model.
Robust Optimization	Works by utilizing optimization techniques that improve adversarial robustness during model training. Robust optimization algorithm needs some prior knowledge of the potential threat/attack.
Defensive Distillation	Works by training the classifier in two-round process and using a specific distillation method. This leads to model learning smoother network, makes the AI model less sensitive to small input changes and harder to attack against.
Gradient Masking / Obfuscation / Hiding	Works by obscuring or hiding gradient information to prevent adversaries from crafting effective adversarial perturbations.
Blocking the Transferability	Introduced NULL Labeling method to prevent adversarial example transferability. Works by augmenting a new NULL label in the dataset and training the model to reject adversarial input and classify them as NULL. Benefits of this method is that accuracy of the clean data is not compromised.
Defense-GAN	Defense-GAN is trained to recognize perturbed and unperturbed images. During inference it finds a close output to a given image without any adversarial changes. The resulting output is then passed to the classifier, making it hard to attack against. This model may alter the data passed to target AI model.

MagNet	MagNet combines two defense strategies: detector networks which detect adversarial inputs and reformer networks which correct adversarial perturbations in input samples. This model alters the data passed to target AI model.
Adversarial detecting	A method, such as DNN network is used in detecting adversarial samples and preventing them to reach target AI model.

Defensive strategies for defending Deep Learning models can also be categorized as proactive or reactive (Table 7.) based on preparations (like adversarial training) required before an adversarial example is detected (Yuan et al., 2017). Reactive strategies are used after DNN network is built and proactive strategies take an adversarial attack into account when DNN network is designed and trained.

Table 7. Countermeasures for adversarial examples (Yuan et al., 2017, modified)

	Defensive strategies
Reactive	<ul style="list-style-type: none"> • Adversarial detecting • Input Reconstruction • Network Verification
Proactive	<ul style="list-style-type: none"> • Network Distillation • Adversarial (Re)Training • Classifier Robustifying

Reasoning for selected method of defense in this thesis

In this thesis, the selected defensive strategy is **Adversarial detecting**, meaning that a **reactive** approach is used. Whenever an adversarial input is detected, it is marked as anomaly. Detection information can be chained into further actions such as blocking the input and alerting system administrator about the anomaly finding.

Digital pathology image analysis reliability is bound to keeping the actual image data immutable for confidentiality and integrity reasons. Pathologists must be able to trust that the clinical analysis is done against unmodified image so that all important image areas remain unchanged, and the results predicted by the image analysis AI model are valid based on the original sample taken from

the patient. This means that reconstructing digital pathology images using Generative Adversarial Networks (GANs) to erase/remove the adversarial perturbations is out of the question.

Another important factor is the accuracy of the actual digital pathology image analysis AI model (DNN network). It should not be affected by the selected method of defense, not by its execution performance or the accuracy of the results it predicts from the input data. Defense methods that inject or augment adversarial data into the digital pathology image analysis AI model may affect the accuracy of the model predictions or even result in errors in clinical analysis.

So, based on the reasons explained, the easiest option to start with is detecting adversarial input. It does not modify (or reconstruct) the image data in any way, it allows blocking and alerting when anomalies are found and does not require changing the way the actual digital pathology AI model is trained.

3 Solution implementation workflow

This chapter goes through the solution implementation following design science research (DSRM) approach, following the six steps of DSRM. Solution implementation covers topics starting from problem identification, motivation, and objectives. After the objectives are set, the thesis proceeds into planning, testing and evaluation of the implemented design artifacts.

3.1 Problem identification and motivation

Chapters 1 and 2 describe in detail the motivation, background, and the need to prevent severe consequences to patient security, caused by possible adversary attacks. Research focus in 2020 year seemed to be on finding new ways to fool deep learning models, also on the field of digital pathology. But back then there was not much research done on the prevention, detection, and defense side. While sharing knowledge about possible ways of attack is important also to defenders of cybersecurity, more focus should be put on practical defense activities. Therefore, in this thesis, one way of detection is implemented using known deep learning architectures and software tools to test in practice how difficult detection and defense really is.

Based on initial literature review, there is no ultimate solution to the problem of defending against adversarial samples on image analysis or defending targeted image classification attacks. This is stated well in (Finlayson et al., 2019) where authors indicate that the defense should hold against

present and future threats while attacker only needs to defeat one specific way of defense. Currently the best options for defense are related in making the deep neural network more robust to attacks by including some adversarial samples into the training data, allowing the DNN network to tolerate some level of perturbation. The research problem is hard, and the number of possible types of attacks is considerable. Also attack type variations are easy to produce, and therefore one can currently only select one or more attacks to detect or defend against. But no universal solution exists that would tolerate all attacks and produce accurate and trustworthy results when under attack.

The value of the proposed solution in this thesis is to offer an evaluated result on detecting adversarial attack of digital pathology image analysis AI model using one known efficient method, Convolutional Autoencoder, trained with image tiles produced from a set of whole slide images. With the results one can either adopt a proposed solution or, based on results gained, develop a different or improved solution for the same research problem. Value for the digital pathology image analysis solutions is to have a way to detect when adversary input is given, then notify system users, and cybersecurity team or even reject (or put in quarantine) the malicious input. If detection works well, the benefits are obvious as there will be no damage to patients caused by deliberately created false analysis results.

3.2 Objectives of a solution

The objective of the proposed solution for the identified research problem is to produce a Convolutional Autoencoder model, trained with image tiles generated from TUPAC16 data set, that can detect selected types of adversarial image attacks against digital pathology images with good detection rate. The practical usefulness of the solution is defined by:

- Performance of the model measured as detection rate.
- Implementation complexity. Solution should be easy to implement and take into use.
- The computational effort (time, cost, number of data samples used) required to train the AI model.
- The computational effort required to run the detection for single image tile.
- The computational effort required to run the detection for complete whole slide image (WSI).

- Integrity of the digital pathology image data and trained AI model. Keeping the image data and AI model unchanged is essential for the trustworthy AI computing. If the analysis input image data has to be modified because of the method used in detection or defense, that may affect the results given by the medical AI analysis performed by the model. If AI model has to be constantly retrained (continuous learning etc.), then it's use for clinical analysis will be hard from regulation point of view.

Additional value for system users and administrators would bring if the solution would also be able to:

- Detect other types of attacks (generalization)
- Explain and show in detail the findings and the type of attack detected. This could be done by annotating detected possible attack areas.

The list of additional value is out of scope of the main objectives of this thesis solution implementation.

3.3 Design & development

In the design and development step, following the selected research method, a Convolutional Autoencoder network was trained to be able to distinguish between image patches from the original data set and adversarial image patches generated by attacker. To be able to train such network, several Python code files were created for training, testing, evaluation, and adversarial image generation. Trained AI model, images patches and code files had an important role in the development, demonstration, and evaluation steps. But one of the most important elements was the knowledge gained to create both the attack and detection implementations.

The main artifacts created in the design and development step were:

- Convolutional Autoencoder network (AI model) that can detect selected types of adversarial image attacks against digital pathology images
- Training data set image patches (size 64 X 64 pixels)
- Python code for generating adversarial image patches (size 64 X 64 pixels) + resulting generated set of adversarial patches
- Python code for training the Autoencoder using Keras
- Python code for attacking CODAIT model over REST API

- Python code for testing and evaluating success metrics
- Knowledge on Autoencoders, Few-pixel attacks, Differential Evolution, training of Convolutional Neural Networks and how to attack AI models on black-box attack scenario

3.3.1 Hardware and software requirements

This section provides a minimum software requirement where a standalone laptop/desktop computer is used. This setup has been used for training the models used and for running AI analysis tasks. But if cloud capacity or high-end computational server cluster is available, running the training and analysis tasks will take a lot less time.

One can also use Docker based Jupyter notebook -environment for developing and documenting AI models, but when tested it seemed to result in 3-4 times slower performance.

Minimum recommended hardware:

- Memory (RAM): 16 GB (or more)
- Processor: 64-bit, 8 cores
- Hard disk space: 500 GB SSD drive or 120 GB SSD for system drive + ~300 GB for second drive
 - When generating image tiles for training, any additional hard disk space is beneficial. 1 TB disk should be a good fit.
- GPU: NVIDIA GeForce GTX 1060 (or later) / NVIDIA Tesla T4 (or later)
 - While CPU-only training is possible, things take about 3-5 times longer than with GPU.
- Network Interface Card (NIC): An ethernet adapter with at least 1 gigabit/second throughput.
 - Training and testing image data download requires decent networking speed.

Software:

- Operating system: **Windows 10/11** or **Ubuntu Linux**
 - At least these are proven to work fine with GPU libraries and NVIDIA display drivers.
- IDEs used for development of solution (both available for Windows/Mac/Linux):
 - **Visual Studio Code** (Microsoft Software License Terms (Visual Studio Code product) MIT license (Code)): <https://code.visualstudio.com/>
 - **Spyder** (Scientific PYTHON Development EnviRonment, MIT License): <https://www.spyder-ide.org/>
- Programming language used in this thesis:

- **Python** (License: Python License, PSF-2.0. Python Software Foundation License (PSFL) is BSD-style license): <https://www.python.org/>
- Python programming language libraries used:
 - **Numpy** (License: NumPy license/BSD License): <https://numpy.org/>
 - **Matplotlib** (License: Matplotlib license/BSD-style license): <https://matplotlib.org/>
 - **Scipy** (License: BSD-3): <https://scipy.org/install/>
 - **Pillow** (License: Pillow is licensed under the open-source Historical Permission Notice and Disclaimer (HPND) license): Python Imaging Library (<https://pillow.readthedocs.io/>)
 - **Split-folders** (License: MIT License): <https://pypi.org/project/split-folders/>
 - **Texttable** (License: MIT License): <https://pypi.org/project/texttable/>
- Whole Slide Image (WSI) processing:
 - **OpenSlide** (License: GNU LGPL): <https://openslide.org/>
- Machine learning frameworks and libraries:
 - **TensorFlow** (License: Apache License 2.0): <https://www.tensorflow.org/>
 - **Keras** (License: Apache License 2.0): <https://keras.io/>
 - **Scikit-learn** (License: BSD-3): <https://scikit-learn.org/stable/>
- Packaging and runtime configuration of applications
 - **Docker engine** (License: Apache License 2.0): <https://docs.docker.com/get-started/overview/>
 - **Docker Compose** (License: Apache License 2.0): <https://docs.docker.com/compose/install/>
- CODAIT -project software:
 - **CODAIT - deep-histopath** (License: Apache License 2.0): <https://github.com/CODAIT/deep-histopath>
 - **Matplotlib** (License: Matplotlib license/BSD-style license): <https://matplotlib.org/>
 - **Numpy** (License: NumPy license/BSD License): <https://numpy.org/>
 - **Pandas** (License: BSD-3): <https://pandas.pydata.org/>
 - **Scipy** (License: BSD-3): <https://scipy.org/install/>
 - **Jupyter Notebook** (License: BSD-3): <https://github.com/jupyter/notebook>
 - **IPython** (License: BSD-3): <https://github.com/ipython/ipython>
 - **Scikit-learn** (License: BSD-3): <https://scikit-learn.org/stable/>
 - **Scikit-image** (License: BSD-3): <https://scikit-image.org/>
 - **OpenSlide Python** (License: GNU LGPL): <https://github.com/openslide/openslide-python>

3.3.2 Dataset preprocessing and splitting

The dataset containing Digitalized whole slide images (WSI) used in the solution implementation is TUPAC16 dataset from the Tumor Proliferation Assessment Challenge 2016 (Veta et al., 2019).

The original plan was to select and download a subset breast cancer case images from The Cancer Genome Atlas (TCGA) which is also the source for TUPAC16 dataset images. Having an expert selected subset of images certainly helped as for a non-pathologist, selecting the right set of breast cancer case images using TCGA search only is not easy.

The training dataset of TUPAC16 is made of 500 breast cancer cases from the data in The Cancer Genome Atlas (TCGA) where one case is represented with one whole-slide image (WSI). The size of the training data is 490 GB, and it includes one separate ground truth CSV file with 500 rows, one for each patient. The testing dataset is smaller (345 GB), and it is made of 321 cases.

From the TUPAC16 dataset pages, one can also download auxiliary datasets for mitoses and regions of interest (for annotations of ROIs where pathologists would perform mitosis counting). There is also mitosis detection testing dataset available without publicly available ground truth.

Source file format

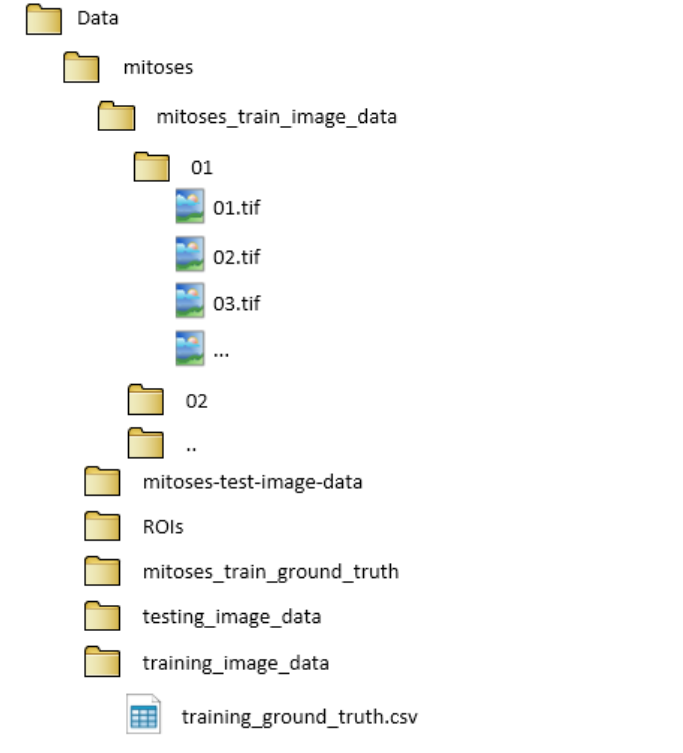
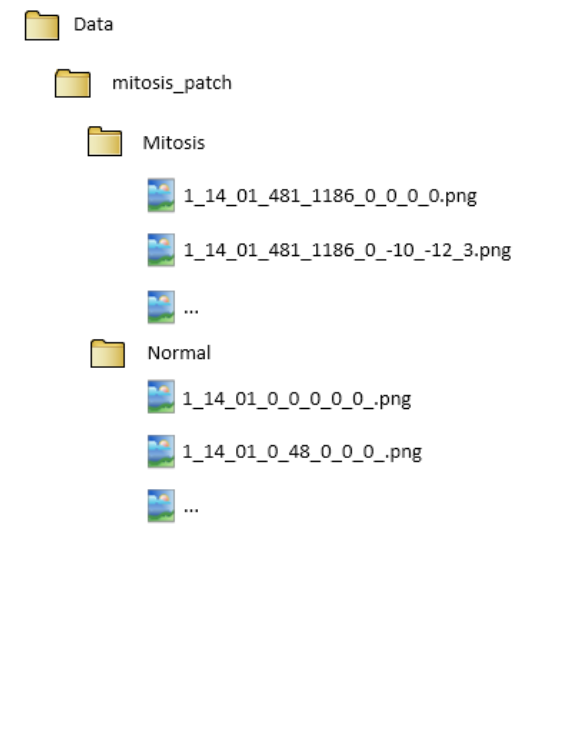
Whole-slide images in the training and testing datasets are provided in the Aperio (.svs file extension) file format. According to (OpenSlide project authors, 2023) Aperio format is "single-file pyramidal tiled TIFF, with non-standard metadata and compression". Aperio format is owned by Leica Biosystems, a company that among other products manufactures whole slide image scanners. Leica Biosystems offer Aperio ImageScope application for viewing .svs files (The Open Microscopy Environment, 2023).

Dataset preprocessing into image patches

With the Python scripts found in (<https://github.com/CODAIT/deep-histopath>) one can call command `python3 preprocess_mitoses.py` to generate image patches of size 64X64 from the large .TIF original image. While it would be simple to use OpenSlide python library to create the patches, the added value of deep-histopath project script is that it uses ROI area information from TUPAC16 csv data files to extract both normal and mitosis patches.

In Table 8 the starting point with original TIFF files and resulting PNG image patches can be seen as folder and file hierarchy.

Table 8. Original image and produced image patches.

Original TUPAC16 dataset (.TIF images + .CSV data)	PNG patches generated from original files
	

Splitting dataset into separate training, validation, and test sets

When creating the dataset used in training AI models, it is a recommended good practice (Brownlee, 2020) to split the original source data into 3 different datasets (training/validation/test). This is beneficial because it helps to prevent overfitting in training, helps the AI model to learn meaningful patterns from the source data, enables model to generalize better on new data and gives model creator honest feedback on model performance when trying with different hyperparameters.

The dataset used in training the convolutional autoencoder model was created by splitting the larger image patch dataset (over 200000 images) generated with CODAIT python scripts into three (3) different datasets:

- Training Set:** Contains data on which the model learns. The model parameters are adjusted to minimize the error on this data and it's the largest portion of the dataset. The training dataset can be labeled on non-labeled, sometimes using folder structure (subfolders) as labels. The data used in this thesis is kept in subfolders during training, but the training engine is instructed to ignore using folders as labels. This is because we want to see both mitosis and normal

image patched as “normal data” and the adversary images generated later by code as “anomaly data”.

- **Validation Set:** Contains data that is used in tuning the hyperparameters of the model. Hyperparameters are the settings controlling the learning process. For example, the number of hidden layers in a neural network or the learning rate. The validation set helps in selecting the best hyperparameters to improve the trained AI model's performance.
- **Test Set:** When the AI model has been trained and tuned using the separate training and validation sets, it's tested on the test set. This dataset provides an unbiased evaluation on how the model performs on new, unseen data.

The Python library called `split-folders` was used in splitting the dataset (Table 9).

Table 9. Dataset split command

```
import splitfolders

splitfolders.fixed(input_folder, output=output_folder, seed=1337, fixed=(7000,
2000, 1000), oversample=False, group_prefix=None, move=False)
```

The resulting database contains 20000 images, split with a ratio of 0.7/0.2/0.1. The training dataset had 14000 images, validation dataset had 4000 images and test dataset had 2000 images (Figure 4).

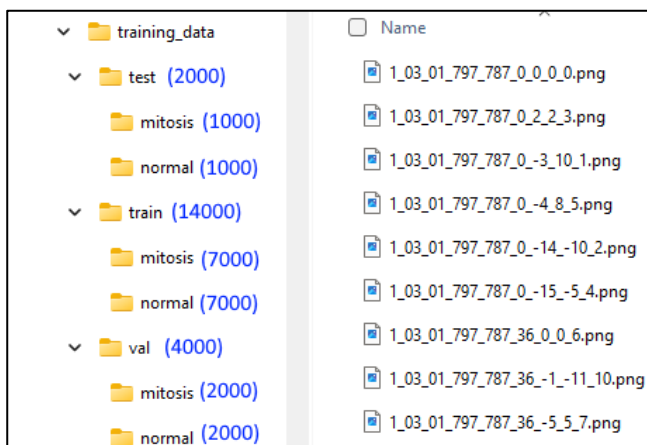


Figure 4. Dataset structure after splitting into train, validation and test folders

Image patches used in demonstration and evaluation steps are picked from the **test dataset**. Also, the adversarial patches are created using same demonstration image patches as the goal is to measure how well the target AI model can be fooled and how well the convolutional autoencoder trained will perform with detecting anomalies from those same images.

3.3.3 Selecting attack target system and AI model

One of the main reasons to use TUPAC16 dataset was to search for a publicly available (open source) digital pathology image analysis AI model that could be used as a target for black-box and white-box attacks. Currently there aren't many of those freely available as the area of digital pathology AI solutions is highly competed and companies do not reveal competitive models to public. Luckily IBM had released their CODAIT/deep-histopath model under Apache-2.0 license (Dusenberry, 2017).

IBM has published a convenient Docker container (CODAIT - Center for Open-Source Data & AI Technologies, 2021) called "IBM Code Model Asset Exchange: Breast Cancer Mitosis Detector" that could be used as Black-box attack target. The model developed by IBM CODAIT uses 64 x 64 PNG image tiles for its input. Tiles can be extracted from one larger breast cancer whole slide image. Given an input image, CODAIT returns the predicted probability of the image patch containing mitosis as a result.

Inside the CODAIT Docker image "codait/max-breast-cancer-mitosis-detector" there is a REST API at port 5000 (default) that one can call with 64X64 pixel sized binary image patch and get back mitosis probability score (between 0-1). REST API user interface is shown in Figure 5.

The screenshot displays the Swagger API test interface for the endpoint `/model/predict`. The interface is divided into several sections:

- Parameters:** A table with columns 'Name' and 'Description'. A required parameter 'image' (file type) is listed with a description: 'An image file encoded as PNG with the size 64*64'. A 'Choose File' button is present, and the file name 'true.png' is shown.
- Execute:** A blue 'Execute' button and a 'Clear' button.
- Responses:** A dropdown menu for 'Response content type' is set to 'application/json'.
- Curl:** A text area containing the curl command: `curl -X POST "http://localhost:5000/model/predict" -H "accept: application/json" -H "Content-Type: multipart/form-data" -F "image=@true.png;type=image/png"`
- Request URL:** A text area containing the URL: `http://localhost:5000/model/predict`
- Server response:** A table with columns 'Code' and 'Details'. The status code is '200'. The 'Response body' is a JSON object: `{ "status": "ok", "predictions": [{ "probability": 0.9884442687034607 }] }`. The 'Response headers' are: `content-length: 71, content-type: application/json, date: Sat, 04 Nov 2023 16:42:02 GMT, server: Werkzeug/0.16.1 Python/3.7.4`

Figure 5. Swagger API description and test user interface for codait/max-breast-cancer-mitosis-detector

3.3.4 Generating adversarial images with Differential Evolution

Finding few-pixel perturbations that can change the AI model prediction towards another class or another prediction certainty can be time consuming and challenging tasks, especially if using black-box attack scenario where the architecture, parameters, or internal workings of the neural network are unknown. To overcome difficulties and computational costs, one can select to use one of the many optimization algorithms that are proven to work efficiently in finding solutions to computationally challenging problems.

For optimization algorithm in this thesis work, Differential Evolution (DE) was selected based on its successful usage in several research articles, including original One pixel attack article Su et al., (2017) and Korphalkola et al., (2020). Differential Evolution (Storn & Price, 1997) is a population-

based optimization algorithm that is often used for global optimization problems. Global optimization algorithms are designed to explore the entire solution space to find the best solution globally even if there are multiple local optima present. This thesis used the Scipy library method **scipy.optimize.differential_evolution** as the DE implementation.

The phases of DE are following:

- **Initialization**
 - Control parameters for DE are initialized.
 - A population of candidate vectors (solutions) is randomly initialized to represent potential solutions to the optimization problem.
- **Mutation**
 - For each candidate solution in the population, a new trial vector is created through the mutation operation. This step combines the information from three randomly selected parent vectors to create a new 'offspring' candidate solution.
- **Crossover**
 - The trial vector is then combined with the original candidate solution using a crossover (CR) operation, defining how much of the information (real valued factor in the range [0,1]) from the trial vector is incorporated into the original candidate solution. When CR is increased, larger number of mutants progress into the next generation.
- **Selection**
 - Now the candidate solution and original one is compared. A better solution is selected for the next generation.
- **Population Update**
 - Solutions selected in selection phase will replace the corresponding solutions in the current population, forming a next generation.
- **Termination**
 - This process is repeated for several generations defined by initialization criteria OR until a termination criterion is met. If the termination criterion is met, it means a solution satisfying given criteria is met.

- In the Scipy library method **scipy.optimize.differential_evolution** the termination is defined by the result of the callback function.

Downsides in using DE for attack optimization may be the challenge in finding parameters that work best for the current task under research and high learning curve when applying the method.

3.3.5 Training convolutional autoencoder to detect anomalies

Autoencoders aims to minimize the reconstruction error to be able to reconstruct the original input as accurately as possible (Goodfellow et al., 2016). This reconstruction error gives a good tool for anomaly detection as it can show how much the input data differs from what the autoencoder has learned based on its training data. In the heart of the autoencoder is the hidden layer (“bottle-neck”) which contains low-dimensional, compact representation of the most useful properties of the training data. Autoencoders do not contain a compressed, exact copy of the training data but rather features it has learned to well represent it.



Training data set

As the task was to develop an AI model that can recognize adversarial images from the original data set (TUPAC16 images), contents of both “Mitosis” and “Normal” folders were combined into “Training” data folder to learn the features common for both types of data. The original “Mitosis” and “Normal” folder image patches were later used to generate attack images of both types so that attack success can be compared for both mitosis-to-normal and normal-to-mitosis attacks.

Information about sample patch image dataset:

- Total number of image patches: 20000
 - Training set: 14000
 - Validation set: 4000
 - Testing set: 2000
- Contains both mitosis and normal patches (see sample images in Table 10)
- File type: PNG (Portable Network Graphic)
- File size (patches): 5-9 KB
- Dimensions: 64 X 64 pixels
- Resolution: 96 pixels/inch

Table 10. Sample image patches from the dataset

Normal image patch sample	Mitosis image patch sample
	

Architecture of the trained autoencoder

The architecture of the deep neural network used in this thesis for detecting anomalies is Convolutional Autoencoder. Given that the convolutional autoencoder architecture is good in image reconstruction, image generation, anomaly detection and computer vision feature extraction tasks, it seemed like a good choice to start with.

The autoencoder used in this thesis has the following characteristics:

- **Undercomplete:** The code dimension (hidden layer) is less than input dimension. Or can also be described as the number of hidden neurons is less than input/output layer neurons.
- **Structure:** Encoder, Latent space, and Decoder.
- **Convolutional:** It uses convolutional layers for both the encoder and decoder components.
- **Unsupervised learning type:** The model learns from the input data itself without explicit external labeling.
- **Number of convolutional layers:** 4 (2 in encoder and 2 in decoder)

Training of the convolutional autoencoder model was done using machine described in 3.3.1 Hardware and software requirements chapter using primarily Python code, Tensorflow 2 and Keras libraries. The training parameters used in convolutional autoencoder model training were as follows:

- input_shape: (64, 64, 3)
- batch_size: None (64)
 - As ImageDataGenerator is used in training data source, the batch size set in generator holds (it's 64)
- optimizer: 'adam'
- loss function: 'mean_squared_error'

- metrics: ['accuracy', 'mse']
- early stopping: patience 30, monitor='val_loss'
- steps_per_epoch: None (64)
 - 'None' = number of samples in dataset divided by the batch size (64)
- epochs: 500
- validation_freq: 1
- validation_steps: None (64)
 - As ImageDataGenerator is used in training data source, the batch size set in generator holds (it's 64)
- callbacks: [early_stopping, csv_logger]
- verbose: 1

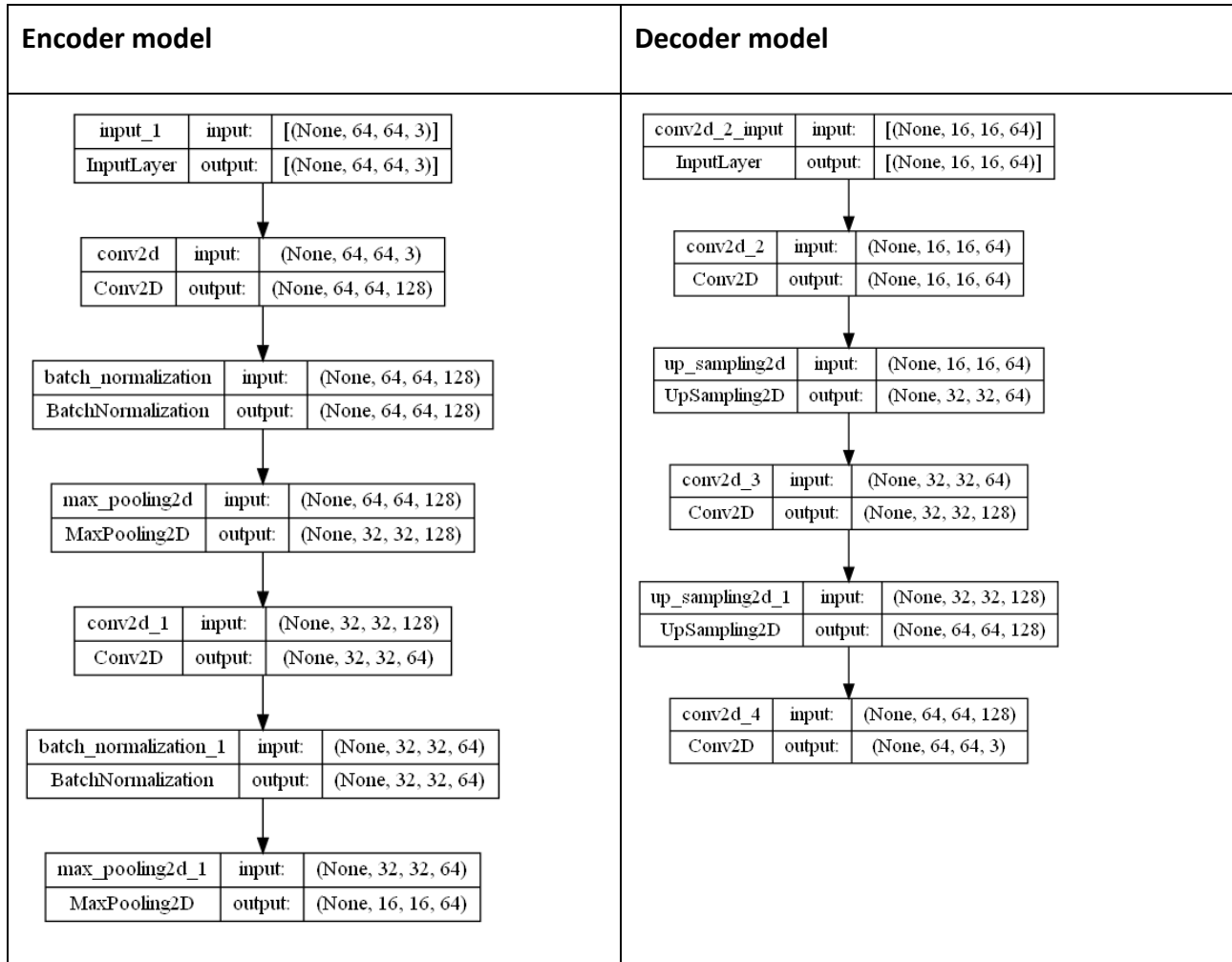
Neural network was visualized using commands shown in Table 11.

Table 11. Commands for neural network visualization

<pre> from keras.utils.vis_utils import plot_model model = Autoencoder() plot_model(model.encoder, to_file='model_plot_encoder.png', show_shapes=True, show_layer_names=True) plot_model(model.decoder, to_file='model_plot_decoder.png', show_shapes=True, show_layer_names=True) </pre>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">Input layer</td> <td style="text-align: center;">input:</td> <td style="text-align: center;">[(None, 64, 64, 3)]</td> </tr> <tr> <td style="text-align: center;">InputLayer</td> <td style="text-align: center;">output:</td> <td style="text-align: center;">[(None, 64, 64, 3)]</td> </tr> <tr> <td colspan="3" style="text-align: center;">↓</td> </tr> <tr> <td style="text-align: center;">Encoder</td> <td style="text-align: center;">input:</td> <td style="text-align: center;">(None, 64, 64, 3)</td> </tr> <tr> <td style="text-align: center;">Sequential</td> <td style="text-align: center;">output:</td> <td style="text-align: center;">(None, 16, 16, 64)</td> </tr> <tr> <td colspan="3" style="text-align: center;">↓</td> </tr> <tr> <td style="text-align: center;">Decoder</td> <td style="text-align: center;">input:</td> <td style="text-align: center;">(None, 16, 16, 64)</td> </tr> <tr> <td style="text-align: center;">Sequential</td> <td style="text-align: center;">output:</td> <td style="text-align: center;">(None, 64, 64, 3)</td> </tr> </table>	Input layer	input:	[(None, 64, 64, 3)]	InputLayer	output:	[(None, 64, 64, 3)]	↓			Encoder	input:	(None, 64, 64, 3)	Sequential	output:	(None, 16, 16, 64)	↓			Decoder	input:	(None, 16, 16, 64)	Sequential	output:	(None, 64, 64, 3)
Input layer	input:	[(None, 64, 64, 3)]																							
InputLayer	output:	[(None, 64, 64, 3)]																							
↓																									
Encoder	input:	(None, 64, 64, 3)																							
Sequential	output:	(None, 16, 16, 64)																							
↓																									
Decoder	input:	(None, 16, 16, 64)																							
Sequential	output:	(None, 64, 64, 3)																							

Detailed structure of both Encoder and Decoder parts of the convolutional autoencoder can be seen in Table 12.

Table 12. Architecture of the Autoencoder used in solution



The model was trained against the training dataset of 14000 images (selected by ImageDataGenerator) and validated against the 4000 validation images (selected by ImageDataGenerator). Model was set to run 500 epochs with early stopping criteria of patience=30, which means that if there is no improvement for 30 epochs, training will stop and save the current model as the “best model” discovered. The progressing of loss and accuracy (common metrics to follow) can be seen in pictures of Table 13.

Table 13. Autoencoder model training & validation loss accuracy



After the training ended during Epoch 218 when early stopping criteria were met, the resulting model was evaluated against the separate test dataset of 2000 images. Only a batch size of 64 images was used in evaluation. Then the first image from the test batch was evaluated for reconstruction error and the resulting 2 images (original, reconstructed) were plotted.

- **Evaluate on test data:**
 - **Test loss:** 0.0002075
 - **Test accuracy:** 0.9703534
- **Reconstruction error (sample image):** 0.0004107

To test how well the newly trained convolutional autoencoder model can reconstruct given image samples, some image patches from both mitosis and normal image patch folders were tested. In the Figure 6 test the reconstruction works well.

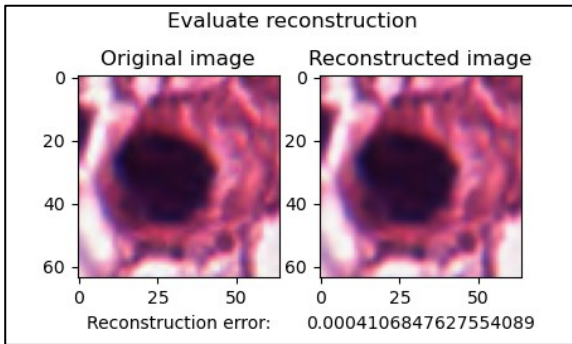


Figure 6. Original image (left) and image reconstructed by the autoencoder(right) trained for solution.

Reasoning for selected convolutional autoencoder structure

The selected model architecture and structure was based on experimenting with several different competing options and hyperparameter tuning. While the resulting model does not generalize extremely well, it performs well in reconstructing the image patches used in the dataset.

Multiple sequential Keras models were tested with different series of convolutional and pooling layers. Usually, the training took a long time and resulting network had considerable reconstruction error even for normal images. So, to overcome these problems, Batch Normalization was added to the Encoder part.

The decision to use Batch Normalization was made to make the training faster with reasonable sized dataset and to prevent the network from overfitting. Using batch normalization to autoencoders not only stabilizes training, improves the network against variations, and promotes smoother convergence but also facilitates more accurate reconstruction of input data in the end.

3.4 Demonstration of solution suitability

In the demonstration part of the design science research approach, the anomaly detection solution is tested in action to solve the original research problem defined in RQ1, chapter 1.1.

To demonstrate how the convolutional autoencoder model works when is get an adversarial image as input, the first task was to collect small enough data set to be used to demonstrate how the autoencoder model works under normal input. Then to demonstrate the effects of an attack, a similar set of adversarial images was needed. Those were crafted using python script that tested the image effectiveness by running a series of HTTP REST API queries, each with separate adversarial mage variation. Last, both original and adversarial image patches were then given as input to

the convolutional autoencoder model to see if it can separate anomaly images from the normal ones (detection as defense), using a reconstruction error threshold value calculated from the original images.

Setting the demonstration environment

The detailed installation manuscript for setting the demonstration environment to be used in both demonstration and evaluation steps is detailed in Appendix 2. The demonstration environment setup instructions.

CODAIT prediction values and decision criteria for demonstration and evaluation

The classification criteria used for CODAIT Docker image ("codait/max-breast-cancer-mitosis-detector") prediction was that predictions over 0.5 are classified as "mitosis" and under 0.5 as "normal". As values given by AI model are predictions, for an image to qualify as "mitosis" image with high certainty, the score predicted should near 0.9 and for "normal" images, the predicted values should be less than 0.1. But on the scale range of probabilities from [0,1], the value 0.5 can already be uncertain and in favor of both result outcomes.

3.4.1 Selecting dataset for demonstration

Demonstration dataset images were randomly selected from a subset of validation dataset images. Demonstration data set contained:

- 50 randomly selected "mitosis" & 50 randomly selected "normal" image patches
- Adversarial image patches generated from those 50 "mitosis" & 50 "normal" image patches
- **Total 200 images:** 100 unmodified image patches & 100 adversarial image patches

3.4.2 Attack demonstration

Adversarial (attack) images were created from the same images that were used in validation data. This is because the goal is to measure how significant reconstruction errors can be caused by using adversarial image patches, compared to the original ones. The python code generated variations of 5-pixel perturbations injected into the given target image. Adversarial versions of both normal and mitosis image patches were generated. Another use for 200 demonstration images was to use

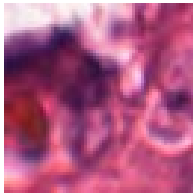
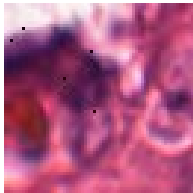


them when computing a good threshold value to use in defense demonstration and evaluation steps.

Attack results for selected images of both type

To demonstrate how well an attack is performed in most optimal cases, samples of both normal and mitosis images are visualized both before and after attack along with targeted AI model predictions.

In the Table 14 below a couple of attack results are highlighted as sample on attack performance. For demonstration dataset, no single attack against “normal” image type was successful, later for evaluation dataset some success was achieved. A full result data table is available in Appendix 3. Demonstration attack & defense results table.

Table 14. Attack efficiency samples

Image patch type	Normal (normal_9.png)		Mitosis (mitosis_45.png)	
Before or after attack	Before	After	Before	After
Image patch				
Mitosis probability prediction	0,0001537 (normal)	0,0961711 (normal)	0.9984108 (mitosis)	0.0061978 (normal)

3.4.3 Threshold used for attack detection

Finding a good threshold value is an essential task to be able to detect anomalies from original images well. The usual value to start with is 0.5 (from range [0,1]) which means that all the samples with predicted probability over 0.5 will be flagged as anomalies (adversarial images), and all values below 0.5 are normal samples. With the convolutional autoencoders we are using the reconstruction error (not prediction), that is calculated by first trying to reconstruct the given image using our trained autoencoder. Then we compare the result with the original image and evaluate the reconstruction error value (float, floating point number). And here it comes a bit tricky: anomalies

cannot be detected directly by using 0.5 (in general 0.5 is not a good final value to use), one must experiment with another dataset first and after finding a good anomaly threshold, use that in evaluating the autoencoder model performance as part of the defense/detection logic.

Some useful values that can be used when finding a good threshold are:

- **Mean** values of autoencoder model predictions on normal and anomaly datasets
 - Compute mean values of both normal data & anomaly data: Then take the mean of those.
- **Precision**: Measures fraction of true positive predictions among the total positive predictions
- **Recall**: Measures the fraction of true positive predictions among the total actual positives
- **F1 score**: This score measures accuracy of the model taking into account both precision and recall (harmonic mean of precision and recall).
- **Accuracy**: The fraction of samples that were predicted correctly

Finding a good threshold is an iterative process, computing precision, recall, and other statistics along several given threshold values to understand how the threshold affects. Also, some tradeoffs must be made as the increasing of recall decreases precision.

Finally, after several trial & error attempts, two methods to find good threshold value were selected:

- **Mean** values of autoencoder model predictions on normal and anomaly datasets
 - Compute mean values of both normal data & anomaly data: Then take the mean of those.
- Threshold matching **Best F1 score** found after iterating F1 scores calculated
 - Compute mean values of both normal data & anomaly data reconstruction error values
 - Then iterate(test) over range of threshold values using `np.arange(start, stop, step)` so that:
 - Start = Average (mean) recognition error of validation images
 - Stop = Average (mean) recognition error of anomaly images
 - Step = 0.000005 (reasonable sized small step)
 - Get threshold values and F1 scores into list and find max F1 score. Return threshold value that was used to get that F1 score.

From these two, the “Best F1 score threshold” was chosen as it was tested to give good results on validation datasets. Using the “Best F1 score threshold” a corresponding threshold was found. Performance was evaluated by using a confusion matrix in defense demonstration step.

Values used are:

- best_f1_score: 0.9056604
- best_threshold (leading to best F1 score): 0.0005542
- Threshold was rounded to **0.000554** for usage in Python scripts

3.4.4 Defense demonstration

To measure the efficiency of the Autoencoder model detection, same set of validation data image patches was used to:

- First, measure the reconstruction error output of Autoencoder model using original image patch.
- Second, measure the reconstruction error output using adversarial (perturbated) version of the same image patch.

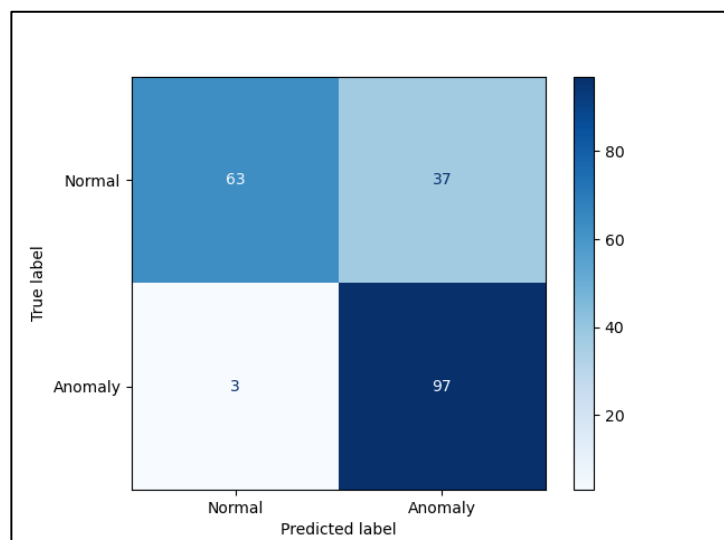
Results were calculated for 100 unmodified (normal/mitosis) image patches and 100 adversarial image patches. The convolutional autoencoder neural network was able to recognize anomalies with good rate but at the cost of false negative findings (normal images recognized as anomaly).

Table 15. Autoencoder reconstruction errors and detection efficiency with validation set

Autoencoder reconstruction error metrics


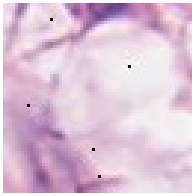

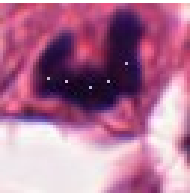
	Before attack	After attack
Maximum	0.0007119	0.0012089
Mean	0.0004542	0.0007765
Median	0.0004594	0.0007581
Standard deviation	0.0001045	0.0001600
Minimum	0.0001882	0.0004346

Confusion matrix visualization of autoencoder classification performance for 100 normal and 100 anomaly images



In the Table 16 below a couple of attack results are highlighted as sample on attack success. A full result data table is available in Appendix 3. Demonstration attack & defense results table.

Table 16. Defense efficiency samples

Image patch type	Normal (normal_50.png)		Mitosis (mitosis_251.png)	
Before or after attack	Before	After	Before	After
Image patch				
Reconstruction error	0,0003872	0,0010391	0,0004432	0,0009099
Anomaly Threshold: 0.000554 (images with reconstruction error over threshold are anomalies)				
Anomaly detection result	Normal	Anomaly	Normal	Anomaly

3.5 Evaluation of the implemented solution

Below in Table 17 is a summary of how well the implemented artifact contributes to the solution of a problem described in the objectives of a solution chapter. Each success criteria are evaluated based on how well the implemented convolutional autoencoder model meets the requirements as a defense (detection) solution. Both the attack and defense are evaluated separately in this chapter and solution success will be evaluated by measuring digital pathology AI model behaviour before and after attack, and the performance of the implemented detector network (autoencoder). Evaluation results are calculated to each of the tested image patches, both for normal and mitosis image patches.

Table 17. Summary of the performance of the implemented solution

Evaluation scale used: (5) Excellent, (4) Good, (3) Average, (2) Weak, (1) Poor		
Success criteria	How does artifact support solution?	Description
Model performance as detection rate	4	Detection accuracy is below 90%.
Computational effort of AI model training	2	Training must be performed continuously with new material to achieve good performance.
Computational effort of single image tile detection run	5	For single tile the result computation is milliseconds
Computational effort of complete whole slide image (WSI) detection run	3	Each tile (with information) must be checked. So, the time is x times single image detection run duration, where x = number of tiles in WSI.
Integrity of the digital pathology image data	5	Implementation does not require any changes to the original image data
Integrity of the Deep Neural Network used in digital pathology image analysis	5	Implementation does not require any changes to the original digital pathology image analysis model.

3.5.1 Selecting dataset for evaluation

Data for the evaluation dataset was a randomly selected subset of testing dataset images. This is to ensure result reliability when demonstration step uses validation dataset and evaluation step uses test dataset.

The evaluation data set contained:

- 50 randomly selected “mitosis” image patches
- 50 randomly selected “normal” image patches

- Adversarial image patches generated from those 50 “mitosis” & 50 “normal” image patches
- **Total 200 images:** 100 unmodified image patches & 100 adversarial image patches

Crafting adversarial images from the testing data

Adversarial (attack) images were created from the same images that were used in testing data. This is because the goal is to measure how significant reconstruction errors can be caused by using adversarial image patches, compared to the original ones.

The python code generated variations of 5-pixel perturbations injected into the given target image. Adversarial versions of both normal and mitosis image patches were generated. The efficiency throughout the Differential Evolution iterations was measured by performing attack and measuring the attack success. Parameters used in the creation of adversarial image using Differential Evolution are listed in detail in Appendix 1. Parameters used in adversarial image generation.

3.5.2 Attack evaluation

The attack against image analysis AI model was evaluated by gathering mitosis probability results before the attack and after attack (according to process defined in Figure 7) using adversarial version of the image patch used in original “before attack” test. Also, the Black-box attack method was evaluated to measure the effort needed to perform an attack against single image patch and series of image patches. Results were calculated using python script.

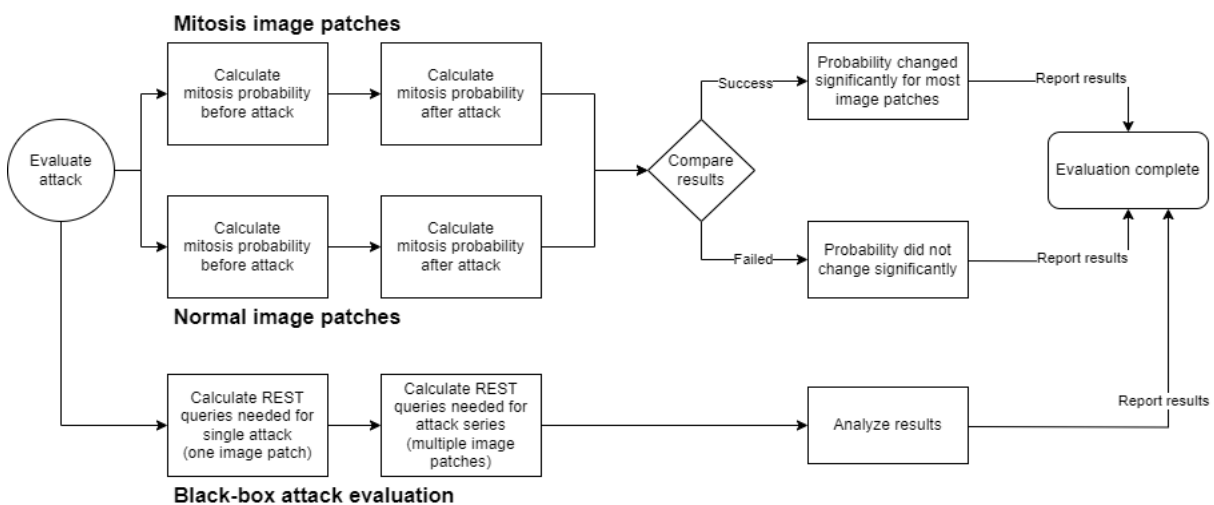


Figure 7. Attack evaluation process

Results for both “mitosis-to-normal” and “normal-to-mitosis” attack were evaluated separately as for the attacker perspective the goal can be either targeted attack or untargeted attack.

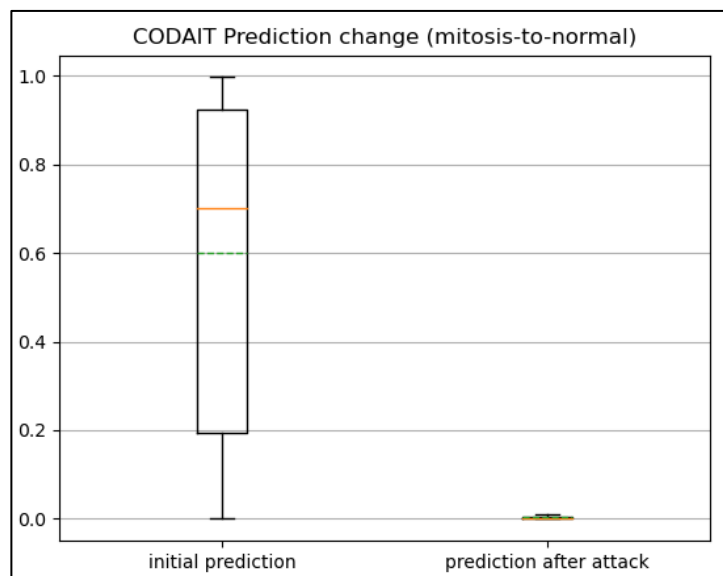
The result evaluation for attacks shows that for “mitosis” image patches, the attack was able to change the original prediction in such a significant amount that the result for all the images is changed from “mitosis” to “normal” (see Table 18). This means this non-targeted black-box attack succeeded perfectly with reasonable amount of iterations (see Table 20, column “mitosis-to-normal”).

Table 18. CODAIT Prediction change (mitosis-to-normal)

CODAIT prediction metrics of mitosis patches before and after attacks were carried out

Box plot visualization of CODAIT predictions of 100 mitosis patches before and after attacks were carried out

	Before attack	After attack
Maximum	0.9967603	0.0093238
Mean	0.5995929	0.0032371
Median	0.7004911	0.0026079
Standard deviation	0.3546401	0.0027327
Minimum	0.0003032	0.0000028



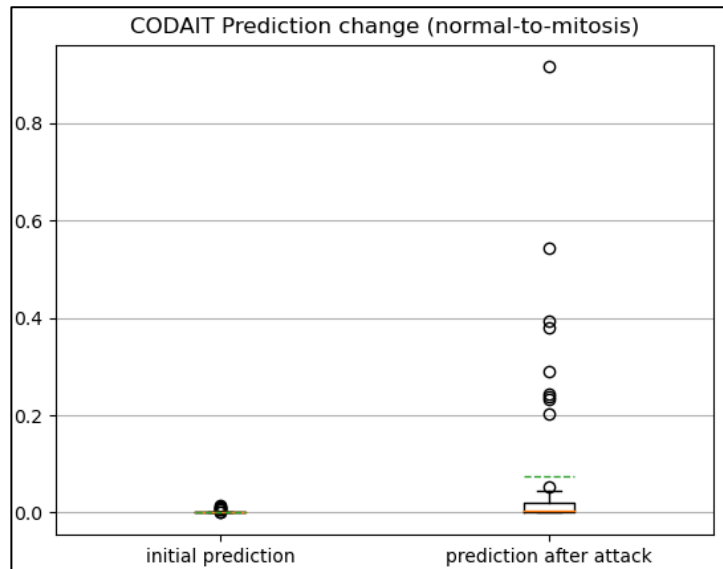
For normal image patches, the attack was not that of success (Table 19). In fact, trying to make as image with a lot of white-colored areas to look like a probable mitosis image was a hard task and would probably require more time and computing power to achieve. Some successful samples were created, even one with certainty over 0.9 to be a “mitosis” image. Complete table of results from both mitosis-to-normal and normal-to-mitosis attacks are available in Appendix 4. Evaluation attack & defense results table.

Table 19. CODAIT Prediction change (normal-to-mitosis)

CODAIT prediction metrics of normal patches before and after attacks were carried out

Box plot visualization of CODAIT predictions of 100 normal patches before and after attacks were carried out

	Before attack	After attack
Maximum	0.0134948	0.9160057
Mean	0.0012509	0.0746376
Median	0.0000268	0.0038450
Standard deviation	0.0031723	0.1727071
Minimum	0.0000001	0.0000061



Evaluation of Black-box attack method

As an additional item of evaluation, the Black-box attack method was evaluated to measure the effort needed to perform attack with adversarial images (few-pixel perturbation). It is important to acknowledge that there is a price for the attacker in both cost and time when trying to find a way to fool the AI model. Also, the attacker does not know if the query-based attack will be noticed because of the high query volume each attack produces.

In Black-box attack using only queries against the attack target, the aim is to find such input or query that when used in the attack, that input will cause the AI model to change the returned result in such way that benefits attacker's goals. Nowadays many of the commercial and open-source AI models are published as subscription-based REST API implementations where subscribed user can operate over the common HTTP protocol by posting input and get the AI model prediction result as output. This is also how the CODAIT Docker image is used, over HTTP protocol by posting queries and inspecting results. CODAIT Docker image is free to use but commercial REST APIs usually charge per request or by the amount of data processed.

In a query-based attack model, finding optimal or near-optimal adversarial attack inputs often requires many queries with different input to be used to gather enough information about how the AI model behind the service works. Common cybersecurity network monitoring systems and API management services are usually setup to:

- monitor the traffic.
- notice and trigger alarm from malicious network behavior or anomalies.
- perform rate limiting. Rate limiting allows only a certain number of HTTP queries from same user account to complete and return result.

This makes attacking REST API implementations more difficult from AI model attack perspective. A query-based attack is easier to detect by the cybersecurity experts monitoring AI service usage. Attackers can split the queries to be performed by multiple accounts created in advance or during a longer period that fits closely to the normal network and API usage patterns. In the Figure 8, behavior of script-based Differential Evolution is seen clearly as all the adversarial input combinations generated by the DE optimization algorithm are sent as HTTP POST requests towards the attack target. Posting hundreds of queries in milliseconds for digital pathology analysis task would clearly show as an anomaly compared to normal network and API usage.

The amount of queries generated by the Differential Evolution based attack solution depends on the difficulty of the optimization task. For the attacks in this thesis, the mitosis-to-normal task is considerably easier than the normal-to-mitosis task. This can be seen in the recorded duration, number of attack requests performed, and average prediction change caused by the performed attack. See Table 20 for results calculated and stored as file under reports folder.

Table 20. Metrics of the black-box attack against AI model behind REST API

	Mitosis-to-normal	Normal-to-mitosis
Average number of HTTP requests used (per generated adversarial image)	742	11863
Average time taken in seconds (solution found/max iterations reached)	14.10	259.83
Average prediction change (of all attacks performed)	0.5963558	0.0733574

As an attack improvement, there is a possibility to implement request throttling, divide attacks into smaller batches of HTTP request and wait for a static or random amount of time (implement “sleep time” into code) so that the traffic seems normal. Also, an AI-based solution could be trained to mimic normal REST API traffic. One could vary the source IP-address with different methods, by for example creating a set of server instances around the globe using commercial cloud platform, each server with its own unique IP address.

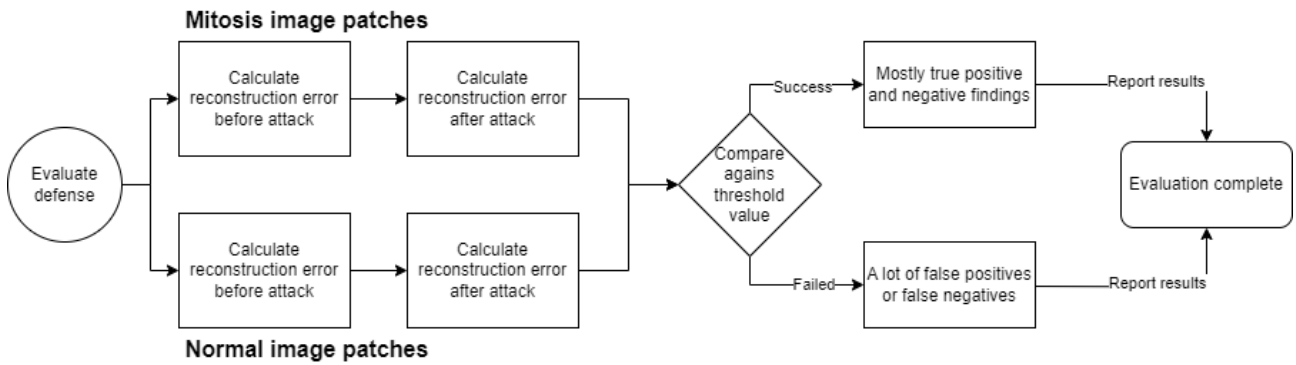


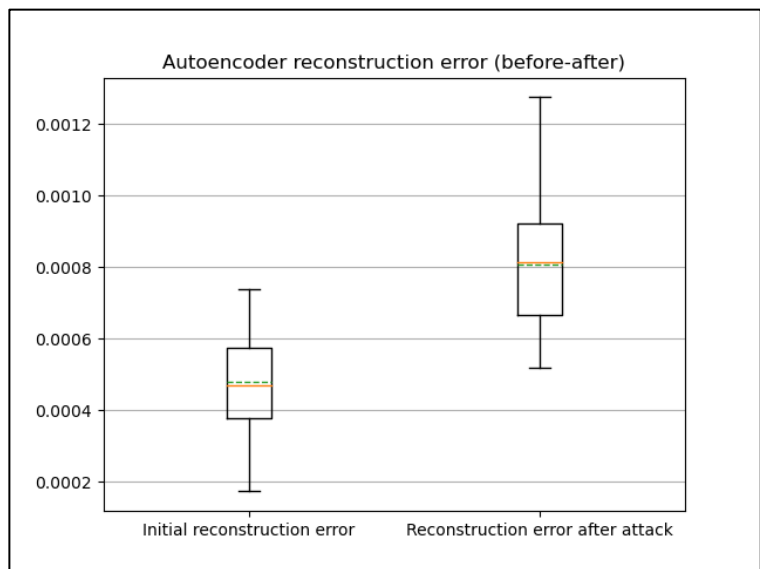
Figure 9. Defense evaluation process

From the results (Table 21) the reconstruction error is almost double for the adversarial images compared to the original non-attacked images. This is helpful when trying to detect and defend against an attack where adversarial images are used. However, results also indicate that model might need more training, improvements on the dataset or a way to effectively discard outlier images on both sides.

Table 21. Autoencoder reconstruction errors (before-after attack) for evaluation dataset

Autoencoder reconstruction error metrics Confusion matrix visualization of autoencoder classification performance for 100 normal and 100 anomaly images

	Before attack	After attack
Maximum	0.0007372	0.0012749
Mean	0.0004787	0.0008074
Median	0.0004702	0.0008144
Standard deviation	0.0001235	0.0001774
Minimum	0.0001733	0.0005179



Performance of the convolutional autoencoder model developed for detecting adversarial images is measured by confusion matrix. It enables an easy way to visualize the performance of AI model by comparing predicted classes with actual classes. The structure of confusion matrix in Keras is a

table with predicted classes on one axis and actual classes on another. This is illustrated in Table 22. Anomalies are marked as positive and non-anomalies as negative.

Table 22. Structure of confusion matrix

Confusion matrix		Predicted	
		Negative	Positive
Actual	Negative	True Negatives (TN) count	False Positives (FP) count
	Positive	False Negatives (FN) count	True Positives (TP) count

For this thesis, the confusion matrix was created by using `confusion_matrix`, `classification_report` and `ConfusionMatrixDisplay` from **Scikit-learn library**. Figure 10 shows how well the trained AI model was able to detect adversarial images from the normal ones. To determine if the image is an adversarial image(anomaly) or normal one, the threshold & F1 score values from the demonstration step were used (3.4.3 Threshold used for attack detection). Threshold value was computed for a demonstration dataset, but it worked equally well also against evaluation dataset.

Values used are:

- `best_f1_score`: 0.9056604
- `best_threshold` (leading to best F1 score): 0.0005542
- Threshold was rounded to **0.000554** for usage in Python scripts.

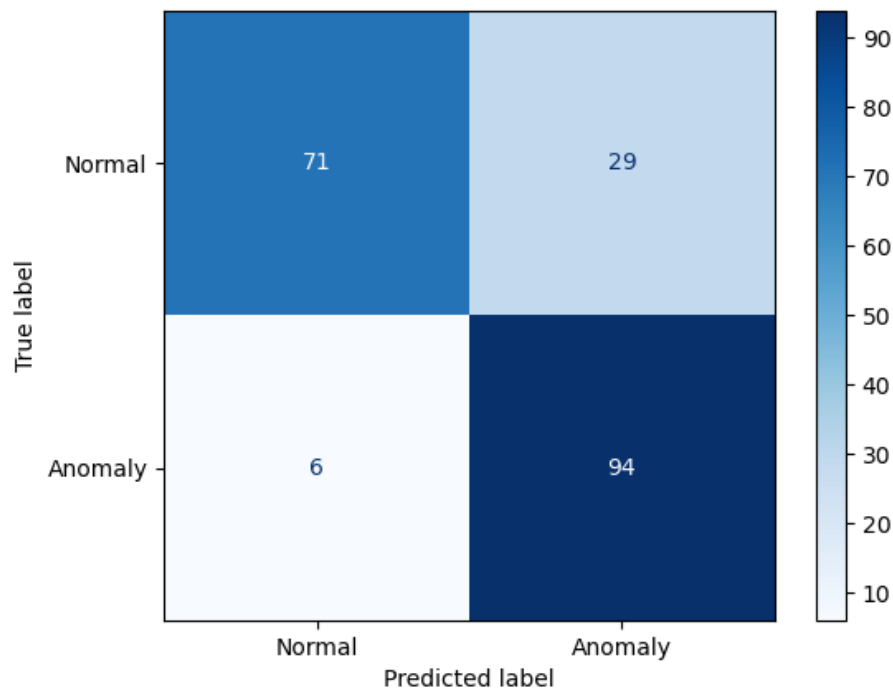


Figure 10. Confusion matrix based on anomaly detection test set

As the goal was to optimize the convolutional autoencoder to detect attack images, the threshold value used worked well to achieve this goal. It must be noted that while the amount of false positives (normal images flagged as anomaly) is ~30% in this test set, it is still acceptable as not being able to catch true positives (adversarial images) would be dangerous from the patient security perspective. Result indicated by the test set run (visualized by confusion matrix) still give room for improvement.

Classification report Table 23 given by `sklearn.metrics` library shows the following metric values for the autoencoder model used:

- **Precision:** ratio of $TP / (TP + FP)$
 - True Positive (TP): Instance that the AI model correctly classifies as positive.
 - False Positive (FP): Instance that the AI model incorrectly classifies as positive.
 - False Negative (FN): Instance that the AI model incorrectly classifies as negative.
 - True Negative (TN): Instance that the AI model correctly classifies as negative.
- **Recall:** ratio of $TP / (TP + FN)$
- **F1-score:** It is the harmonic mean of the precision and recall scores obtained for the positive class.
- **Support:** The number of occurrences of each class in the dataset.

- **Accuracy:** Proportion of correct matches. Accuracy = correct predictions/total number of predictions.
- **Macro avg:** It is the sum of metric values for all classes, divided by the total number of classes.
- **Weighted avg:** Calculates the average performance by considering the weighted contribution of each class to the overall metric.

	precision	recall	f1-score	support
Normal (0)	0.92	0.71	0.80	100
Anomaly (1)	0.76	0.94	0.84	100
accuracy			0.86	200
macro avg	0.84	0.82	0.82	200
weighted avg	0.84	0.82	0.82	200

Table 23. Classification report

3.6 Communication of results and knowledge

Following the Design Science Research Method (DSRM), finding of the research and the designed artifact must be communicated to academic community and parties involved. This thesis is published to Theseus (<https://www.theseus.fi/>) system under “JAMK University of Applied Sciences” school thesis collection to be publicly available for anyone to read. The importance of this solution design and information contained by this thesis is in indicating a rather new area of threats and their mitigations in Digital Pathology cybersecurity. Finding practical defense methods for Deep Neural Networks attacks in Digital Pathology Image Analysis system is a challenge that must be tackled in advance so that the patients and pathologists can rely on trustworthy AI solutions also in the future.

The DSRM in this thesis produced following artifacts:

- Convolutional autoencoder deep neural network architecture for detecting possible attack against Digital Pathology Image Analysis system.
- Trained Convolutional Autoencoder as Keras (.keras) model file for detecting possible anomalies in the input data. Also the weights are saved into their own HDF5 (.hd5) file. The HDF5 (Hierarchical Data Format version 5) is a file format used to store both the architecture of the model and its learned weights.

- Results of the trained Convolutional Autoencoder effectiveness to solve the problem at hand. Results are evaluated in the chapter “Evaluation of the implemented solution”.
- Attack code for generating Few-pixel attacks to demonstrate Black-box type of attack against a deep neural network served over REST API
- Finally, suggestions on how to improve detection and defense of Deep Neural Networks (DNN) in Digital Pathology Image Analysis system.

In addition to the information provided by this thesis and its attachments, code files used in solution implementation can be found from authors public GitHub library. Code is available under MIT license. For source code of his thesis, see GitHub repository: https://github.com/amarkus/thesis_jamk.

4 Discussion

The goal for the thesis was to study practical attack and defense methods for integrity of Deep Neural Networks in Digital Pathology Image Analysis systems. This was done by using a design science-based research method, producing concrete software artifacts (code) that will demonstrate how attack can be performed against DNN and to test if there is a way to detect the attack by using another neural network implementation. The primary idea in the thesis was to implement adversarial images and then try the detection with a rather simple and straightforward autoencoder solution that could be taken into use with reasonable implementation knowledge and computational resources.

Performance for the implemented few-pixel attack was good against “mitosis image patches” but moderate or poor against “normal image patches”. This is quite expected result as generating such perturbations that would make some of the mostly white images to be predicted as mitosis images is hard task or would have required hand-picking of such “normal image patches” that would be closer to mitosis image based on their original prediction values.

The evaluated performance of the implemented autoencoder detector model was moderate and could be further improved with more training and better selection of the training data set. In this

demonstration a threshold value was used to separate normal images from anomalies, but by adjusting the threshold value it could be used to spot only the most likely anomalies from the image analysis data.

From a production usage perspective, the demonstrated setup needs to be improved so that detection is more accurate and that most likely requires different kind of neural network implementation than convolutional autoencoder. The continuous scanning of image data in the training or inference data set to find anomalies requires considerable amount of computing resources to be done in fast and efficient manner. Cost/benefit ratio of implementation and actual risks should be considered before implementing any adversarial image detection services.

4.1 Autoencoder suitability for image anomaly detection

During the implementation step the goal was to find near-optimal deep neural networks-based solution for the problem of detecting adversarial attack images to prevent attacks from succeeding. It is a common fact of convolutional autoencoders (AE) that when AE is used for image reconstruction, one can compare the goodness of AE based on reconstruction error. Further reading of source literature and web articles showed that indeed AE image reconstruction error has been used for image anomaly detection with success.

When implementing initial models using convolutional autoencoders and TensorFlow it soon became obvious that creating “near-perfect” detector using only convolutional autoencoder model would be very difficult, if not impossible task. Reason is that when splitting a large whole slide image into tiles, there are a lot of different areas that can all be part of normal tissue but unlike distinction between apples and oranges, tissue areas (+ slide background color) can vary a lot both in shape and in color scale. This requires a well-formed and clean training data set and a lot of samples of valid (no anomalies) image tiles. Problems of Autoencoder sensitivity to outliers in the training data set and shortcomings are detailed in (Beggel et al., 2019).

4.2 On the probability of attacks against Digital Pathology Image Analysis systems

There isn't much research literature available on how likely it is that deep learning models used in medical imaging, especially in Digital Pathology would be attacked by adversarial samples by actual cybercriminals. Therefore, the estimates done in this chapter are educated estimates based on the current state of deployments in hospitals and the adaptation of AI models in Digital Pathology. In the Table 24 the two common types of attack using adversarial images are estimated based on the likeliness to happen in near future or longer period.

Current wars and confrontations between countries certainly affect by slightly increasing the probability of any kind of cyberattacks. However, cybercriminals and even nation state actors usually select targets of high financial value, high attention value or significant psychological effect. Naturally the easier targets requiring less effort are preferred. Therefore, the value of attacking a currently small number of live digital pathology image analysis and their training systems is not enough to justify the effort. Also, many of the current systems require second opinion and approval of clinical analysis results from a pathology expert.

During the next decades we will see more systems requiring less and less human intervention in the analysis process, and significant volume increase of image analysis to whole slide images in digital pathology. That increases the financial benefit of clinical analysis result manipulation and probability of attacks to occur.

Table 24. Probability for inference or training attack occurrence

Attack type	Probability	Time assessment	Description
Cybercriminals perform attack against digital pathology image analysis (inference) using adversarial image perturbations	Highly improbable	0-5 years	Not probable as easier targets exist with less effort required from the attacker. Small number of AI deployments. Financial benefit not significant.
	Probable	5-10 years	Probably the first live attack seen or at least attempted. Enabled by larger deployment

			of AI software in hospitals all around the work with <u>varying security policies</u> . Greater financial benefit and attention value.
Cybercriminals perform poisoning of digital pathology training data using adversarial image patches	Probable	0-5 years	Much of the training data comes from the same open-source or commercial sources. Easier targets for manipulation of data and damage done is considerable. Reliability of the data may not be questioned or measured.
	Highly probable	5-10 years	Progress and adoption of defense methods against adversarial attacks in production systems, makes manipulation of training data easier target from effort vs benefit perspective.

4.3 Improvement ideas for adversarial image detection

During the information retrieval and solution design & development steps there were a lot of information sources available about different techniques and methods used in both attacking and defending the DNN models used in image analysis systems. Many such alternative methods were published during 2022-2023. In this chapter some of those methods are covered to give ideas for improving adversarial image detection and defense.

Using Kernel Density Estimation (KDE) as additional anomaly detection criteria

If the training data set contains some outlier images, then in many cases only using the convolutional autoencoder's reconstruction error as an anomaly detection criterion will result in degrade

of detection performance in autoencoder (Beggel et al., 2019). Then one way to improve the inadequacies of the trained autoencoder is to chain multiple detection criteria and form the decision-making process based on the thresholds in multiple criteria instead of just one. One must carefully consider this, as the need to use additional criteria or methods may also be an indication of poor initial deep neural network implementation.

In paper (Beggel et al., 2019) the researchers use the likelihood of the image in the latent space. In this thesis, the corresponding latent space would be produced as the most compressed (last) layer in the “encoder” part of Autoencoder network. Using the latent space resulting from the Autoencoder “encoder” part training as low-dimensional representation of images, a density distribution can be calculated using KDE that represents well kernel density scores of normal images and used as threshold value to indicate anomaly image when given as input (KDE calculated for the input image). Combination of Autoencoder reconstruction error threshold and KDE threshold for anomaly images could improve the anomaly detection compared to using only the Autoencoder reconstruction error threshold.

Using 3rd party adversarial attack and anomaly detection libraries

An interesting open-source approach has been developed by the Linux Foundation AI & Data Foundation (LF AI & Data) called Adversarial Robustness Toolbox (Nicolae et al., 2019). ART source code is published under MIT License. ART contains samples for both Red and Blue Teams to use when trying to attack or defend machine learning models.

Foolbox (Rauber et al., 2018) is a Python library to run adversarial attacks against machine learning models such as deep neural networks. For measuring ML system vulnerability to adversarial examples, one can use Python library called CleverHans (Papernot et al., 2018).

Alibi Detect Python library contains code for outlier detection, adversarial detection, and drift detection (Klaise et al., 2020). It contains a sample for Adversarial Autoencoder detection and correction on CIFAR-10 dataset.

Moving from Convolutional autoencoder to Vision Transformers

Vision transformers (ViTs) have been an active target of research and in the paper (Ghaffari Laleh et al., 2022) researchers find vision transformers are more resilient against several gradient-based adversarial attacks than convolutional neural networks. While CNNs can be trained to have better resiliency against adversarial attacks, a notable discovered fact is that vision transformers did not

require adversarial pretraining or architectural modifications to achieve better robustness. It would be interesting to research further on the weaknesses of vision transformers and types of adversarial attacks that work well with those models. Therefore, it would make sense to further study the effectiveness of vision transformers to see pros and cons of ViTs compared to CNNs in both robustness against adversarial attacks and ability to detect adversarial samples in training of inference input data.

5 Conclusion

In this thesis the aim was to demonstrate an answer to the research questions presented in chapter 1.1 about how one can in practice attack against Deep Neural Networks (DNN) in the context of Digital Pathology Image Analysis system, and as a defense, to present one practical option on how to detect such attack and prevent it from happening. This is an important step towards teaching more generalizable models that can detect multiple types of attacks. The code required to train the convolutional autoencoder, generate adversarial images and calculate metrics for evaluation used in this thesis is published in GitHub to be freely available and to allow anyone to repeat the experiment or further develop upon the codebase used.

Another very important finding during the solution design and implementation step was that the problem is difficult and in the constant battle between attacker and defender, the advantage is always on the attacker's side. While there are multiple research articles on detecting adversarial attacks against image classification and segmentation models, even more articles exist on the different attacks one can use to fool them. That should not stop researchers from trying to find effective ways for defending against attacks but gives a realistic view on what to expect on the complexity of the task.

The answer to the main research question RQ1 of this thesis is the sum of the answers to detailed research questions RQ2-RQ4. One must know what is currently known to be possible from the attack point of view to be able to defend or detect those attacks. This is a common theme for cybersecurity blue team (defender) members. Also, the defense and detection largely depend on the type of attack performed. No single attack type or defense type will rule them all. A good start is to try to tackle the most common attack types first, as those require less from the attacker to implement.

Following the design science research approach, a solution artifact was implemented by training a convolutional autoencoder network to recognize adversarial image patches from original image patches. Results from the implemented solution indicate that as there is a wide variance of colors in digital pathology image patches from almost completely white patches to very colorful ones, the task of finding adversarial images is hard but still results in sufficient performance to start with. Results can be improved through more careful selection of the training data and using other machine learning methods than convolutional autoencoders.

Part of the task was to implement a practical Black-box attack that one can perform against DNN model without knowing the exact implementation and architecture of the targeted model beforehand. This attack was performed as query-based by issuing a series of HTTP requests towards the target. For improving the attack result, Differential Evolution was used as an optimization algorithm to incrementally improve the few-pixel perturbations used in adversarial input images. When evaluating attack results, it was shown that normal image patches were more difficult to attack compared to mitosis images, and therefore mitosis-to-normal (hide results requiring treatment) attack is easier to perform for the attacker. This is most likely because many “normal” images contain a lot of white/empty areas and making CODAIT/deep-histopath model predict those as “mitosis” images would require more perturbation than just 5 pixels that were used in the attack in this thesis.

To ensure research reliability and ethics, publicly available TUPAC16 data set was selected in the context of Digital Pathology as it contains breast cancer case images from well-known The Cancer Genome Atlas (TCGA). Also, a publicly available Digital Pathology image analysis system implementation was needed to be the target of Black-box attack scenario. IBM CODAIT/deep-histopath was selected as it is free to use, has been used in TUPAC16 challenge, includes a scientific paper about the implementation and allowed attack to be performed safely in an on-premises setup where HTTP requests were done inside the same restricted local host network.

For research reliability it must be mentioned that the generalization of the network was not measured thoroughly against other attack targets (image analysis services) or other Digital Pathology image data sets. This is because different Digital Pathology image data sets have usually been created with different Whole Slide Image scanners, affecting the image quality and color scheme, and therefore would have required retraining of the Convolutional Autoencoder model.

6 References

Aiforia Technologies Oyj. (2023, November 4). *About Aiforia*. Aiforia Public Website.

<https://www.aiforia.com/about-us>

Alatalo, J., Sipola, T., & Kokkonen, T. (2022). Detecting One-Pixel Attacks Using Variational Autoencoders. In Á. Rocha, H. Adeli, G. Dzemyda, & F. Moreira (Eds.), *Information Systems and Technologies - WorldCIST 2022, Volume 1, Budva, Montenegro, 12-14 April, 2022* (Vol. 468, pp. 611–623). Springer. https://doi.org/10.1007/978-3-031-04826-5_60

Beggel, L., Pfeiffer, M., & Bischl, B. (2019). *Robust Anomaly Detection in Images using Adversarial Autoencoders*. <http://arxiv.org/abs/1901.06355>

Brownlee, J. (2020, August 14). *What is the Difference Between Test and Validation Datasets?* <https://machinelearningmastery.com/difference-test-validation-datasets/>

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2018). *Adversarial Attacks and Defences: A Survey*. <https://doi.org/10.48550/arXiv.1810.00069>

CODAIT - Center for Open-Source Data & AI Technologies. (2021). *codait/max-breast-cancer-mitosis-detector*. <https://hub.docker.com/r/codait/max-breast-cancer-mitosis-detector>

Dusenberry, M. (2017, September 9). *CODAIT/deep-histopath/LICENSE*.

<https://github.com/CODAIT/Deep-Histopath/blob/master/LICENSE>.

<https://github.com/CODAIT/deep-histopath/blob/master/LICENSE>

Eichelberg, M., Kleber, K., & Kämmerer, M. (2020). Cybersecurity in PACS and Medical Imaging: an Overview. *Journal of Digital Imaging*, 33(6), 1527–1542.

<https://doi.org/https://doi.org/10.1007/s10278-020-00393-3>

- Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). *Robust Physical-World Attacks on Deep Learning Models*.
<https://arxiv.org/abs/1707.08945>
- Finlayson, S. G., & Beam, A. L. (2019). *Adversarial Attacks Against Medical Deep Learning Systems*.
<https://doi.org/10.48550/arXiv.1804.05296>
- Finlayson, S. G., Bowers, J. D., Ito, J., Zittrain, J. L., Beam, A. L., & Kohane, I. S. (2019). Adversarial attacks on medical machine learning. *Science*, *363*(6433), 1287–1289.
<https://doi.org/10.1126/science.aaw4399>
- FIRST.Org, M. authors. (2024, February 3). *Common Vulnerability Scoring System version 4.0: User Guide*. <https://www.first.org/cvss/v4.0/user-guide>
- Ghaffari Laleh, N., Truhn, D., Veldhuizen, G. P., Han, T., van Treeck, M., Buelow, R. D., Langer, R., Dislich, B., Boor, P., Schulz, V., & Kather, J. N. (2022). Adversarial attacks and adversarial robustness in computational pathology. *Nature Communications*, *13*(1).
<https://doi.org/https://doi.org/10.1038/s41467-022-33266-0>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
<https://www.deeplearningbook.org/>
- JAMK. (2018, June 24). *Ethical Principles for JAMK University of Applied Sciences Approved by the Student Affairs Board on 11 December 2018*. <https://www.jamk.fi/en/media/34826>
- Klaise, J., Looveren, A. Van, Cox, C., Vacanti, G., & Coca, A. (2020). *Monitoring and explainability of models in production*.
- Korpihalkola, J., Sipola, T., & Kokkonen, T. (2021). Color-Optimized One-Pixel Attack Against Digital Pathology Images. *2021 29th Conference of Open Innovations Association (FRUCT)*, 206–213.
<https://doi.org/10.23919/FRUCT52173.2021.9435562>

- Korpiahkola, J., Sipola, T., Puuska, S., & Kokkonen, T. (2020). *One-Pixel Attack Deceives Computer-Assisted Diagnosis of Cancer*. <https://doi.org/10.1145/3483207.3483224>
- Metzen, J. H., Kumar, M. C., Brox, T., & Fischer, V. (2017). *Universal Adversarial Perturbations Against Semantic Image Segmentation*.
- Narodytska, N., & Kasiviswanathan, S. (2017). Simple Black-Box Adversarial Attacks on Deep Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1310–1318. <https://doi.org/10.1109/CVPRW.2017.172>
- Nguyen-Son, H.-Q., Thao, T. P., Hidano, S., Bracamonte, V., Kiyomoto, S., & Yamaguchi, R. S. (2021). OPA2D: One-Pixel Attack, Detection, and Defense in Deep Neural Networks. *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–10. <https://doi.org/10.1109/IJCNN52387.2021.9534332>
- Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I. M., & Edwards, B. (2019). *Adversarial Robustness Toolbox v1.0.0*.
- OpenSlide project authors. (2023, October 22). *OpenSlide/Aperio format*. <https://openslide.org/formats/aperio/>
- Pantanowitz, L., Sharma, A., Carter, A. B., Kurc, T., Sussman, A., & Saltz, J. (2018). Twenty years of digital pathology: An overview of the road travelled, what is on the horizon, and the emergence of vendor-neutral archives. In *Journal of Pathology Informatics* (Vol. 9, Issue 1). Wolters Kluwer Medknow Publications. https://doi.org/10.4103/jpi.jpi_69_18
- Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., ... Long, R. (2018). Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *ArXiv Preprint ArXiv:1610.00768*.

- Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., ... McDaniel, P. (2016). *Technical Report on the Clever-Hans v2.1.0 Adversarial Examples Library*. <http://arxiv.org/abs/1610.00768>
- Papernot, N., McDaniel, P., & Goodfellow, I. (2016). *Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples*. <http://arxiv.org/abs/1605.07277>
- Parwani, A. V. (2019). Next generation diagnostic pathology: Use of digital pathology and artificial intelligence tools to augment a pathological diagnosis. In *Diagnostic Pathology* (Vol. 14, Issue 1). BioMed Central Ltd. <https://doi.org/10.1186/s13000-019-0921-2>
- Paschali, M., Conjeti, S., Navarro, F., & Navab, N. (2018). *Generalizability vs. Robustness: Adversarial Examples for Medical Imaging*. <http://arxiv.org/abs/1804.00504>
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Quan, W., Nagothu, D., Poredi, N., & Chen, Y. (2021). *CriPI: an efficient critical pixels identification algorithm for fast one-pixel attacks*. 21. <https://doi.org/10.1117/12.2581377>
- Rauber, J., Brendel, W., & Bethge, M. (2018). *Foolbox: A Python toolbox to benchmark the robustness of machine learning models*.
- Sipola, T., Puuska, S., & Kokkonen, T. (2020). Model Fooling Attacks Against Medical Imaging: A Short Survey. *Information & Security: An International Journal*, 46(2), 215–224. <https://doi.org/10.11610/isij.4615>
- Sorin, V., Soffer, S., Glicksberg, B. S., Barash, Y., Konen, E., & Klang, E. (2023). Adversarial attacks in radiology – A systematic review. *European Journal of Radiology*, 167, 111085. <https://doi.org/https://doi.org/10.1016/j.ejrad.2023.111085>

- Stallings, W. (2019). *Effective Cybersecurity: Understanding and Using Standards and Best Practices*. Addison-Wesley. <http://williamstallings.com/Cybersecurity/>
- Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4), 341–359. <https://doi.org/10.1023/A:1008202821328>
- Su, J., Vargas, D. V., & Kouichi, S. (2017). *One pixel attack for fooling deep neural networks*. <https://doi.org/10.1109/TEVC.2019.2890858>
- The Open Microscopy Environment. (2023, October 22). *Bio-Formats Documentation/Formats/Supported Formats/Aperio SVS TIFF*. <https://bio-formats.readthedocs.io/en/stable/formats/aperio-svs-tiff.html>
- Veta, M., & et al. (2024, May 2). *Tumor Proliferation Assessment Challenge - Dataset*. <https://tupac.grand-challenge.org/Dataset/>
- Veta, M., Heng, Y. J., Stathonikos, N., Bejnordi, B. E., Beca, F., Wollmann, T., Rohr, K., Shah, M. A., Wang, D., Rousson, M., Hedlund, M., Tellez, D., Ciompi, F., Zerhouni, E., Lanyi, D., Viana, M., Kovalev, V., Liauchuk, V., Phoulady, H. A., ... Pluim, J. P. W. (2019). Predicting breast tumor proliferation from whole-slide images: The TUPAC16 challenge. *Medical Image Analysis*, 54, 111–121. <https://doi.org/10.1016/j.media.2019.02.012>
- Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., & Jain, A. K. (2019). *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review*. <https://doi.org/10.48550/arXiv.1909.08072>
- Yuan, X., He, P., Zhu, Q., & Li, X. (2017). *Adversarial Examples: Attacks and Defenses for Deep Learning*. <http://arxiv.org/abs/1712.07107>

7 Appendices

Appendix 1. Parameters used in adversarial image generation

Adversarial image generation parameters:

- Number of adversarial samples to generate:
 - Normal images: 50
 - Mitosis images: 50
- Amount of change expected:
 - expected_change_factor_mitosis: 100
 - expected_change_factor_normal: 80
 - Normal images are harder to attack against and setting change factor higher might only be a waste of computing time without significant change in result.
- Number of pixels to attack: 5
- Color bounds used in attack:
 - Normal images: near_black_bounds = [(0, 5), (0, 5), (0, 5)]
 - Mitosis images: pink_bounds = [(220, 225), (170, 175), (200, 205)]
- Image patch locations:
 - normal_images_path: "data/normal_patches"
 - mitosis_images_path: "data/mitosis_patches"
 - adversarial_images_path: "data/adversarial_patches"

Parameters for Differential Evolution:

- func (objective function to be minimized):
 - Different functions are used for 'normal' and 'mitosis' images.
 - We want to minimize it for 'mitosis' images and maximize it for 'normal' ones.
- bounds (variable bounds):
 - Different bounds are used for 'normal' and 'mitosis' images.
 - See "Color bounds used in attack" in 3.4.1 Selecting dataset for demonstration.
- strategy (selected differential evolution strategy): "best1bin"
- maxiter (maximum number of generations): 500
- popsize (multiplier for setting the total population size): 30
- mutation (mutation constant / differential weight): (0.3, 1)

- recombination (recombination constant / crossover probability): 0.7
- atoll (Absolute tolerance for convergence): -1
- disp (print the evaluated func at every iteration): True
- polish (polish the best population member at the end): True
- population initialization: "latinhypercube"

Appendix 2. The demonstration environment setup instructions

The demonstration environment is setup using the following manuscript:

- Initialize the attack target:
 - The actual attack is implemented through a series of HTTP POST queries against the CODAIT/deep-histopath model running as Docker container. HTTP REST API of CODAIT/deep-histopath was accessible on localhost port 5000.
 - If using Windows/Mac/Linux there is an option to use Docker Desktop. Download and install Docker desktop. Then Start Docker desktop.
 - Notice: There are multiple ways to run Docker containers, also without Docker Desktop. But that is the simplest option for most users.
 - Install Python 3.x by using Anaconda installer or command line
 - Download the CODAIT/deep-histopath container using the following command: **docker pull codait/max-breast-cancer-mitosis-detector**
- Create Python virtual environment and install requirements (python libraries)
 - Open terminal/command prompt and run command: **pip install virtualenv**
 - Create a new folder (for the project), navigate to project folder in your terminal (cd command):
 - **mkdir <project-name>**
 - **cd <project-name>**
 - Then run the following command: **python<version> -m venv <virtual-environment-name>**
 - Run command to activate virtual environment: **source <virtual-environment-name>/bin/activate**
 - Note: After the virtual environment is created, you can use it in VS Code, Anaconda.Navigator, Spyder and many other tools.
- Download code used in development, demonstration, and evaluation steps:

- On your computer, under the project folder you created run command: **git clone https://github.com/amarkus/thesis_jamk.git**
- This will download the source code containing all the code used in different steps of this thesis, except the image patch generation from TUPAC16 images (<https://github.com/CODAIT/deep-histopath>)
- In your tool of choice, set the current working directory as “<some_dir_path>\thesis_jamk”:
 - One-liner to check the current working directory:
python -c "import os; print(os.getcwd())"
- Install required python packages:
 - Navigate to **src** folder
 - Run command: **pip install -r requirements.txt**
- If not already done in training the convolutional autoencoder:
 - Prepare image patches from TUPAC16 dataset (see 3.3.2 Dataset preprocessing and splitting)
 - Copy some 100000-200000 images (mitosis folder needs to have over 10000) from patches from **CODAIT\data\mitoses\patches\train** folder to **dataset\patches** folder under the project folder.
 - Navigate into the “**solution_design_and_development**” folder and run **prepare_training_data.py**. This will generate train, val and test folders under **training_data** folder (directly under datasets folder).
 - Navigate into the “**solution_design_and_development**” folder and run **train_anomaly_detector.py**. As a result, the folder **src/run_results/<run_id>/models** will contain a keras file. That is your convolutional autoencoder model. The folder **src/run_results/<run_id>/** will also contain screenshots, logs, and model summary for debugging needs.

Now you should be able to run the scripts for demonstration and evaluation also.

Appendix 3. Demonstration attack & defense results table

Image name	Initial prediction	Prediction after attack	Initial reconstruction error	Reconstruction error after attack
mitosis_1.png	0,148413569	0,000735580	0,000459316	0,000774894

normal_1.png	0,000041971	0,009503714	0,000513094	0,000831877
mitosis_2.png	0,922306359	0,007309685	0,000224037	0,000608885
normal_2.png	0,000001702	0,000380007	0,000506389	0,000715072
mitosis_3.png	0,543869495	0,004026819	0,000400077	0,000558080
normal_3.png	0,000012727	0,006169492	0,000538159	0,000756408
mitosis_4.png	0,157493040	0,000435463	0,000572376	0,000763913
normal_4.png	0,000004124	0,000066945	0,000376544	0,000657802
mitosis_5.png	0,417111874	0,003761098	0,000471778	0,000850438
normal_5.png	0,000012809	0,006896434	0,000494574	0,000793531
mitosis_6.png	0,955929995	0,009479479	0,000358863	0,000651374
normal_6.png	0,000047098	0,000515505	0,000402633	0,000916627
mitosis_7.png	0,431792140	0,003948513	0,000608882	0,000781549
normal_7.png	0,000386461	0,013763451	0,000607773	0,001075869
mitosis_8.png	0,346063435	0,002397292	0,000506101	0,000851330
normal_8.png	0,000014482	0,001597611	0,000477807	0,000804672
mitosis_9.png	0,729875982	0,005932807	0,000425172	0,000679207
normal_9.png	0,000153686	0,096171118	0,000374779	0,000681507
mitosis_10.png	0,117708840	0,000241658	0,000344551	0,000594535
normal_10.png	0,000012478	0,000986555	0,000562975	0,000888699
mitosis_11.png	0,977434158	0,001612379	0,000455520	0,000755749
normal_11.png	0,000165269	0,012995451	0,000633106	0,000992789
mitosis_12.png	0,963935912	0,009143808	0,000288653	0,000677566
normal_12.png	0,000016052	0,000610318	0,000583112	0,001037779
mitosis_13.png	0,897652924	0,001752942	0,000292758	0,000434632
normal_13.png	0,000001292	0,000340625	0,000418452	0,000830780
mitosis_14.png	0,998695672	0,320512086	0,000448726	0,000764912
normal_14.png	0,000093107	0,001791153	0,000316912	0,000752405
mitosis_15.png	0,300658047	0,002991061	0,000592037	0,000765517
normal_15.png	0,000105568	0,025303578	0,000343787	0,000579192
mitosis_16.png	0,475262255	0,001495939	0,000650389	0,000950290
normal_16.png	0,000001314	0,000020806	0,000527908	0,001044561
mitosis_17.png	0,708144724	0,001151610	0,000373309	0,000697099
normal_17.png	0,000000510	0,000003508	0,000512268	0,000759002
mitosis_18.png	0,879327178	0,007079005	0,000338766	0,000585314
normal_18.png	0,000026389	0,022385601	0,000510679	0,000899967
mitosis_19.png	0,203910992	0,001742448	0,000464506	0,000657389
normal_19.png	0,000007310	0,003639274	0,000327715	0,000793213
mitosis_20.png	0,975126565	0,003535033	0,000365537	0,000757173
normal_20.png	0,000004855	0,000574146	0,000572748	0,001096880

mitosis_21.png	0,203611642	0,001192780	0,000349845	0,000740107
normal_21.png	0,000024900	0,019636095	0,000475079	0,000802950
mitosis_22.png	0,537590981	0,001949711	0,000360358	0,000664426
normal_22.png	0,000000889	0,003671370	0,000594335	0,000959667
mitosis_23.png	0,341924250	0,002120507	0,000459512	0,000611895
normal_23.png	0,000079310	0,001024267	0,000556149	0,000955571
mitosis_24.png	0,880136192	0,004757602	0,000402609	0,000636143
normal_24.png	0,000001878	0,000497219	0,000519868	0,001034495
mitosis_25.png	0,290616304	0,000292063	0,000528872	0,000740660
normal_25.png	0,000013901	0,009991005	0,000529542	0,000989252
mitosis_26.png	0,963564575	0,007676982	0,000475212	0,000808516
normal_26.png	0,000062225	0,001310628	0,000474056	0,000726044
mitosis_27.png	0,813812256	0,006990579	0,000393369	0,000747075
normal_27.png	0,000016552	0,002715214	0,000440478	0,000717063
mitosis_28.png	0,784856081	0,007029271	0,000406684	0,000676100
normal_28.png	0,000023335	0,000353302	0,000354228	0,000675525
mitosis_29.png	0,851696134	0,003199333	0,000317658	0,000560948
normal_29.png	0,000002587	0,000289122	0,000403808	0,000654283
mitosis_30.png	0,274329156	0,000070314	0,000352982	0,000720285
normal_30.png	0,000005340	0,000076600	0,000479827	0,000776059
mitosis_31.png	0,531968772	0,004610776	0,000382672	0,000617131
normal_31.png	0,000008099	0,013352522	0,000447822	0,000839770
mitosis_32.png	0,976763606	0,005248149	0,000479618	0,000725932
normal_32.png	0,000001493	0,000556821	0,000550196	0,000895875
mitosis_33.png	0,936972916	0,000589918	0,000521675	0,000804988
normal_33.png	0,000038136	0,000278694	0,000381965	0,000859856
mitosis_34.png	0,090065710	0,000675426	0,000188178	0,000437867
normal_34.png	0,000029371	0,008464520	0,000393075	0,000632904
mitosis_35.png	0,740434289	0,001459896	0,000402481	0,000632345
normal_35.png	0,000021989	0,001536802	0,000500422	0,000926090
mitosis_36.png	0,262425482	0,002271656	0,000428102	0,000586888
normal_36.png	0,000001255	0,000177536	0,000528869	0,001082251
mitosis_37.png	0,765155137	0,000460444	0,000362311	0,000687527
normal_37.png	0,000004080	0,002228478	0,000464842	0,000836157
mitosis_38.png	0,039777346	0,000387052	0,000467323	0,000649182
normal_38.png	0,000042465	0,001053018	0,000502394	0,000879077
mitosis_39.png	0,002447123	0,000008824	0,000388801	0,000619520
normal_39.png	0,000000713	0,000645112	0,000534446	0,000801638
mitosis_40.png	0,514352322	0,000502599	0,000453803	0,000754409

normal_40.png	0,000526292	0,038574133	0,000540067	0,001025475
mitosis_41.png	0,950375676	0,005177333	0,000678369	0,000998959
normal_41.png	0,000012877	0,002525970	0,000503823	0,000797285
mitosis_42.png	0,596053958	0,003930310	0,000464670	0,000608197
normal_42.png	0,014112246	0,072855488	0,000658165	0,001121363
mitosis_43.png	0,114015333	0,000743672	0,000508351	0,000745800
normal_43.png	0,000018849	0,004824089	0,000437686	0,000915101
mitosis_44.png	0,969435096	0,008734260	0,000346385	0,000709688
normal_44.png	0,000016008	0,005539660	0,000526471	0,000904132
mitosis_45.png	0,998410821	0,006197840	0,000443224	0,000909922
normal_45.png	0,000272839	0,011999988	0,000553122	0,000882455
mitosis_46.png	0,299752563	0,002330190	0,000286664	0,000474421
normal_46.png	0,000311008	0,015046140	0,000367788	0,000558693
mitosis_47.png	0,617649257	0,005823713	0,000711869	0,000859951
normal_47.png	0,000000392	0,000009144	0,000660069	0,001208907
mitosis_48.png	0,990627408	0,008096716	0,000597812	0,000974689
normal_48.png	0,000214633	0,003455861	0,000310276	0,000685424
mitosis_49.png	0,007887042	0,000059159	0,000340301	0,000602807
normal_49.png	0,000004125	0,000162429	0,000331757	0,000524227
mitosis_50.png	0,934542537	0,000212351	0,000266848	0,000568269
normal_50.png	0,000043786	0,000720795	0,000387180	0,001039097

Appendix 4. Evaluation attack & defense results table

Image name	Initial prediction	Prediction after attack	Initial reconstruction error	Reconstruction error after attack
mitosis_1.png	0,861586809	0,002933479	0,000403515	0,000694025
normal_1.png	0,000000125	0,000006046	0,000654039	0,001034940
mitosis_2.png	0,508837402	0,003429842	0,000558350	0,000849870
normal_2.png	0,000088287	0,015161656	0,000638837	0,001024484
mitosis_3.png	0,976993203	0,008370327	0,000344540	0,000761791
normal_3.png	0,000021582	0,006168855	0,000508193	0,001079600
mitosis_4.png	0,040070117	0,000283404	0,000343472	0,000616856
normal_4.png	0,000008508	0,000024747	0,000466306	0,000646110
mitosis_5.png	0,942859530	0,006173938	0,000419355	0,000631905
normal_5.png	0,005960859	0,543592095	0,000529099	0,001029449

mitosis_6.png	0,951725364	0,002114710	0,000423353	0,000725156
normal_6.png	0,000001338	0,000026185	0,000417315	0,000819426
mitosis_7.png	0,662126899	0,005187168	0,000628526	0,000842162
normal_7.png	0,000094111	0,004635017	0,000500958	0,000756711
mitosis_8.png	0,993190706	0,003885588	0,000389265	0,000818062
normal_8.png	0,000015341	0,000286067	0,000471179	0,001051436
mitosis_9.png	0,988112629	0,009323802	0,000581883	0,001008967
normal_9.png	0,000008260	0,000286751	0,000665288	0,000901613
mitosis_10.png	0,975726903	0,005520195	0,000545166	0,000823079
normal_10.png	0,000016696	0,003009412	0,000560992	0,000789938
mitosis_11.png	0,996760309	0,002569380	0,000173283	0,000551101
normal_11.png	0,000006222	0,001185444	0,000527690	0,000873499
mitosis_12.png	0,152778775	0,000348541	0,000283078	0,000732881
normal_12.png	0,000452658	0,290014535	0,000592616	0,000857122
mitosis_13.png	0,093522131	0,000685714	0,000435727	0,000613879
normal_13.png	0,000005982	0,000532239	0,000596250	0,000994258
mitosis_14.png	0,984991252	0,000671417	0,000285611	0,000603182
normal_14.png	0,012027426	0,054091424	0,000423403	0,000736616
mitosis_15.png	0,555196762	0,002646575	0,000342206	0,000585478
normal_15.png	0,000035430	0,000922100	0,000507059	0,000951306
mitosis_16.png	0,133751780	0,000747747	0,000568632	0,000852862
normal_16.png	0,000013655	0,011056987	0,000322551	0,000646458
mitosis_17.png	0,852719724	0,008195848	0,000496698	0,000739022
normal_17.png	0,000006990	0,004050226	0,000451216	0,001040308
mitosis_18.png	0,901980758	0,004354236	0,000339345	0,000746069
normal_18.png	0,000373711	0,010094131	0,000544319	0,000930654
mitosis_19.png	0,206469983	0,000714460	0,000598259	0,000912544
normal_19.png	0,000097254	0,005270306	0,000361893	0,000556447
mitosis_20.png	0,792511344	0,004817124	0,000305439	0,000563664
normal_20.png	0,000033047	0,009772085	0,000407767	0,000709176
mitosis_21.png	0,827489853	0,006083493	0,000320813	0,000614566
normal_21.png	0,000004269	0,000710378	0,000619536	0,000902113
mitosis_22.png	0,482688248	0,003692272	0,000356787	0,000665832
normal_22.png	0,000077331	0,238524631	0,000610591	0,000823337
mitosis_23.png	0,912186742	0,000045685	0,000725277	0,001005554
normal_23.png	0,000477635	0,005277387	0,000539185	0,000833929
mitosis_24.png	0,080556132	0,000773440	0,000370304	0,000559779
normal_24.png	0,013494822	0,040900834	0,000723363	0,001094747
mitosis_25.png	0,180041835	0,000788218	0,000439219	0,000776875

normal_25.png	0,000005982	0,000499840	0,000596250	0,000995177
mitosis_26.png	0,034925595	0,000231816	0,000461911	0,000565518
normal_26.png	0,008238786	0,380933434	0,000341740	0,000517954
mitosis_27.png	0,990396857	0,009108635	0,000680095	0,001021950
normal_27.png	0,007635538	0,394852161	0,000459155	0,001081589
mitosis_28.png	0,908827603	0,004196631	0,000364350	0,000544776
normal_28.png	0,000004815	0,009165278	0,000595764	0,000878691
mitosis_29.png	0,000303229	0,000002749	0,000308762	0,000673548
normal_29.png	0,000017036	0,000306312	0,000322630	0,000837066
mitosis_30.png	0,924608111	0,006086867	0,000238724	0,000533590
normal_30.png	0,000002369	0,002294043	0,000497339	0,000852417
mitosis_31.png	0,657065034	0,002821964	0,000545104	0,000918871
normal_31.png	0,000001181	0,000072544	0,000510476	0,000730158
mitosis_32.png	0,738855243	0,003363867	0,000534546	0,000810682
normal_32.png	0,000951621	0,916005731	0,000571423	0,000920070
mitosis_33.png	0,394089550	0,000699077	0,000469128	0,000650055
normal_33.png	0,000000494	0,000237823	0,000649482	0,001042027
mitosis_34.png	0,833088636	0,002429518	0,000408609	0,000596312
normal_34.png	0,000420977	0,023655837	0,000358105	0,000586153
mitosis_35.png	0,924941063	0,005847837	0,000544753	0,000914407
normal_35.png	0,000010368	0,000225086	0,000414711	0,000908144
mitosis_36.png	0,547642052	0,005148123	0,000372587	0,000665996
normal_36.png	0,000012867	0,000807639	0,000630015	0,000939428
mitosis_37.png	0,959904552	0,003680790	0,000459890	0,000706456
normal_37.png	0,000001109	0,000127522	0,000694194	0,000994339
mitosis_38.png	0,969527423	0,009250625	0,000375798	0,000679238
normal_38.png	0,000283154	0,009045467	0,000300523	0,000617744
mitosis_39.png	0,796106696	0,002230616	0,000547540	0,000894445
normal_39.png	0,000025291	0,003639846	0,000590286	0,001274992
mitosis_40.png	0,098639891	0,000609463	0,000368121	0,000608158
normal_40.png	0,000061714	0,001210482	0,000384634	0,000544199
mitosis_41.png	0,375225991	0,001880224	0,000247568	0,000525189
normal_41.png	0,001031355	0,244819820	0,000595024	0,000880439
mitosis_42.png	0,622190595	0,002417985	0,000504503	0,000693577
normal_42.png	0,000015125	0,002785343	0,000727062	0,001077682
mitosis_43.png	0,191802621	0,001516148	0,000369241	0,000585079
normal_43.png	0,000044519	0,011891786	0,000613268	0,001051413
mitosis_44.png	0,532338023	0,004602729	0,000400386	0,000679548
normal_44.png	0,000029134	0,001194934	0,000479315	0,001036730

mitosis_45.png	0,167223439	0,000804042	0,000540781	0,000709515
normal_45.png	0,000028328	0,001623439	0,000481210	0,001037945
mitosis_46.png	0,831625998	0,007252574	0,000452184	0,000756052
normal_46.png	0,008872952	0,232463703	0,000737232	0,001256906
mitosis_47.png	0,157239035	0,000297302	0,000409302	0,000738732
normal_47.png	0,000479103	0,044775996	0,000464226	0,000898477
mitosis_48.png	0,096121520	0,000242361	0,000385462	0,000713385
normal_48.png	0,000021607	0,000282375	0,000614566	0,001024582
mitosis_49.png	0,217245176	0,000437816	0,000529805	0,000884339
normal_49.png	0,000001167	0,000031124	0,000432330	0,000676310
mitosis_50.png	0,954838574	0,002342672	0,000376643	0,000705945
normal_50.png	0,001031355	0,203333601	0,000595024	0,000920043