Faculty of Mathematics and Informatics

VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
MODELLING AND DATA ANALYSIS
MASTER'S STUDY PROGRAMME

Fusion of Medical Images and Preclinical Data in Ophthalmology Using Deep Learning

Master's thesis

Author: Simas Bijeikis
VU email address: simas.bijeikis@mif.stud.vu.lt
Supervisor: Prof. Dr. Olga Kurasova

Vilnius

2024

Contents

Abstract

This paper serves as a master thesis titled "Fusion of medical images and preclinical data in ophthalmology using deep learning". It explores data fusion and compares it to simple segmentation based on deep learning. At first, an overview of relevant literature is provided. It gives knowledge about deep learning, CNV pathology, data fusion, as well as achievements in the field. Then in order to do the task, ophthalmology data was prepared: preclinical data was collected that would tell various information about animals, as well as two sets of images (OCT b-scans and FA images) were annotated. Later on, three algorithms are compared. The first one is image segmentation using the trained neural net model. Then early data fusion algorithm, and the late data fusion algorithm both follow different approaches to the task. Results are calculated: image segmentation is compared using the Dice coefficient and methods are compared to each other using the Wilcoxon rank-sum test. Results show that data fusion using given methods didn't improve image segmentation: the only combination where the difference of Dice coefficients wasn't significant was between initial segmentation and late data fusion algorithms for FA images, but that model had other problems. Discussions and conclusions are formed explaining why data fusion wasn't as good performing as expected.

Keywords: Data fusion, Choroidal neovascularization, Deep learning

Abbreviations

- CNV - choroidal neovascularization
- FA - fluorescein angiography
- MLP - multi-layer perceptrons
- OCT - optical coherence tomography
- SVM - support vector machines

1 Introduction

Deep learning is an important tool in biomedical data analysis. Vast amounts of medical images are being generated every moment and each of them must be examined by professionals. Due to advancements in machine learning, doing data science projects is becoming easier. Not only examinations can be done faster but also accurate deep learning algorithms remove the risk of human error. On the other hand, to achieve this, deep learning models must be programmed correctly and developed well so that it can be trusted with the task. One of the bases of a good biomedical algorithm is accurate and representative clinical data that holds information not only about clinical status (if a patient has the disease and how serious it is) but also phenotypical data that might have influence. Combining both clinical data and medical images is a complex but crucial part of biomedicine and these days it's important that this task would be given to deep learning so that medical examinations would become faster and more precise.

For some pathologies the parameter that defines how well treatment works is a specific area in the image. Medical experts mark such regions of interest themselves and calculate measurements out of that. These days such task is usually left for deep learning models where using ground truth images a neural net model is trained that would make segmentation much faster. Unfortunately, such models still make mistakes and aren't perfect. One possible way to increase performance is to make this model influenced by clinical data. Usually for humans information is being collected like sex, age, date of examination, and so on, though in this thesis preclinical data will be used, which means that instead of humans, tests are done with laboratory animals (more specifically mice) for better examinations on dosage and toxicity (how dangerous treatment might be). If preclinical data is used to improve image segmentation, such methodology would be called data fusion.

While scientists have already made attempts to implement data fusion on various tasks (including image segmentation), for ophthalmology (science about eye diseases) in preclinical stages it would be a novel approach. It's not only that ophthalmology is a relatively poorly explored field but also preclinical stage isn't as popular in biomedical data science as the clinical one. Due to this, it would be interesting and useful to explore, how data fusion works with ophthalmology images in the preclinical stage, whether it works, and what's the best approach.

1.1 Goals and objectives

The main goal of this thesis is to test whether data fusion improves image segmentation in ophthalmology in the preclinical stages.

To fulfill the purpose of this thesis it's important to complete these objectives:

1. To analyze literature about relevant topics from both biological and mathematical fields.

2. To prepare preclinical data as well as medical images.

3. To create neural net models using medical images.
4. To combine preclinical data with medical images to make segmentation more accurate.
5. To summarize results and make conclusions.

Completing all the objectives would provide the desired insight into data fusion for preclinical ophthalmology image segmentation. To achieve this, the thesis will have the following sections: literature analysis providing knowledge into the field and what has been done already, data overview exploring data that was used, analytical part where algorithms will be tested and results calculated, discussion where results will be summed up and analyzed, and conclusion with final notes. After that references and an appendix will provide additional supporting material.

2 Literature analysis

In this section main terminology will be required to work with the data, as well as already created algorithms will be explored to draw inspiration and ideas on how this idea could be approached.

2.1 Fundamentals of artificial neural networks

The main purpose of this thesis can be achieved using deep learning. It is a technology that by using complex levels of the algorithm with multiple layers, achieves a result that would be difficult to define by the singular function [19]. This allows us to make important advancements in fields like speech recognition, visual object detection, and many others.

In general, deep learning is a subfield in a broad sphere of machine learning. Machine learning itself is at the intersection of computer science and statistics, and at the core of artificial intelligence and data science [16]. It tries to answer two main questions: how a system learning through experience can be created, and what are essential statistical computation-information-theoretic laws that govern all learning systems. In recent decades data science has relied heavily on machine learning and this amount increases every year, from economical tasks to sorting credit card transactions into fraud or not, to language models able to answer detailed questions. All this can be achieved when constructing various complexity models to solve these problems.

Data science models that deep learning uses have a complex structure. When training, the algorithm has parameters that determine an output and assign weights to each. They compose a complex neural network structure consisting of layers that each has its use when determining the final output. The first such algorithm was invented by Frank Rosenblatt in 1957, which was called perceptron and it forms the smallest computational unit of deep learning [31]. It's sometimes also called a neuron due to its similarity to a basic working unit of the brain. In Figure 1 there is a visual representation of neuron (a) and a whole deep learning model consisting of multiple neurons structured in layers (b).

Rosenblatt defined formula for it:

$$y = f(\sum_{i=1}^{D} w_i x_i + b)$$

Here D is the dimension of input space, x is input vector, w is set of weights corresponding to the input vector, b is bias, and f is activation function. Perceptron has D+1 tunable parameters (D weights and one bias) and can be described as multiple linear regression augmented by an output function (f), which is non-linear. The form of the activation function was originally a step function, but now a range of monotonic functional forms, such as sigmoidal, are used.

Perceptrons are connected into structures called multi-layer perceptrons (MLP) [28]. The

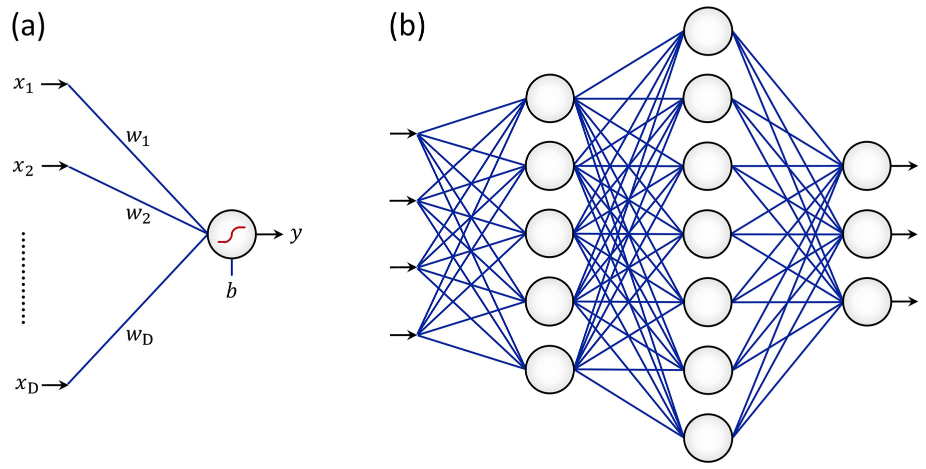


Figure 1: Visual representation of neural network fundamentals. (a) - perceptron, (b) - whole structure consisting of multiple perceptrons, also called multi-layer perceptrons (MLP) [28]

first layer of neurons, also called the first hidden layer, would have Dn_1 weights and n_1 biases, where n_1 is the number of neurons in this layer. The second hidden layer would have n_1n_2 weights and n_2 biases while for last one its $n_{d-1}n_d$ weights and n_d biases, where d is a total number of layers and n_{d-1} and n_d are the numbers of neurons in the second-to-last and last layers. The total number of layers in an MLP and the number of neurons in each layer are hyper-parameters. Due to its complex structure, MLP would have many more tunable parameters than the perceptron. One more important thing to consider is the choice of activation functions in each layer. What to consider is the fact that a linear activation function is typically only suitable for the last layer and any stack of linear layers is effectively equivalent to a single linear layer.

Deep learning didn't become popular instantly because at first weight optimization was arduous. Because of that, the back-propagation algorithm was invented to enable the training of MLPs with any network structure [32]. Back-propagation is an optimization algorithm that minimizes a loss function, F, representing the goodness-of-fit of predictions to observations, for example, the sum of squared errors. It is defined by this formula:

$$F = \sum_{i=1}^{M} \sum_{j=1}^{n_d} (T_j^k - y_j^k)^2$$

Here y_j^k is the output of neuron j in the output layer when the network is forced with input data sample k and T is the respective desired target, M is the size of training data, and n_d is the number of neurons in the output layer. Using this function weights can be optimized more easily. The training of such a model is an iterative optimization process, where parameters are updated after each iteration (also called epoch), to minimize the loss function. Such a method of learning is called stochastic gradient descent[19].

There are multiple model architectures created. For this segmentation task, ResNet34 will be used. It's an image classification model, structured as a 34-layer convolutional neural network. ResNet34 is also a good model because it was pre-trained on the ImageNet dataset[15] that contains more than 100,000 images with around 200 different categories. Pre-training is useful so that the model would better know how to segment in general: neighbour pixels usually belong to the same category, and regions of interest are rarely shaped in a specific way but they are rather blobs. In general ResNet34 was implemented that input images were cropped to be 224x224, then batch normalization was adopted right after each convolution and before activation. Some of the parameters that were used when training: weights were initialized from scratch, stochastic gradient descent was used with mini-batch size of 256, learning rate starts from 0.1 and is multiplied by 0.1 when error stops changing, models were trained for up to 60 $\times$ 104 iterations, weight decay of 0.0001 and a momentum of 0.9 were used [17]. For all deep learning based image segmentation tasks in this thesis ResNet34 architecture will be used.

2.2 Insight into deep learning

When working with medical images, usually supervised learning is applied. It's a form of machine learning where a model learns by having true references what output must be with some kind of input (usually images themselves). This allows for making the task more precise though requires a vast amount of ground truth images (also called annotations).

When training models there are a few things to consider. One of them is accurate data splitting into sets [29]. Usually, there are three: training - which images will be used as inputs for machine learning, validation - another smaller set of data to fine-tune hyperparameters, test - data to test already trained model. The idea is that if these datasets were the same, the created model would be overfitted on the dataset, or in other words, would be very accurate on the current dataset but would fail when applied to new data. There are multiple ways to split the dataset but the most important factor to consider is that all datasets should be representative and have a similar distribution of output values, for example when doing classification, if the training dataset has unbalanced classes, the model could learn to assign most into one, thus, overfitting and having a bias towards the most popular class, or if it's regression, a cluster of similar points could shift linear function in a way that it would no longer be accurate for further points.

Another thing to consider when training models are number of epochs. On one hand, it is important because, after a number of epochs, the loss value might start increasing indicating that the model is starting to overfit and, thus, losing its accuracy. It might be useful to limit the number of epochs also due to computational resources and time. Even though computers are increasing in power at high speeds, training deep learning models requires a lot of power. It might be possible to train it on a simple computer but it might require an unreasonable amount of time, thus, creating an obstacle loosely related to the machine-learning model itself.

One more thing to consider when training a model is the task itself. One of the most popular ones is the binary classification where the model calculates a real value or more in N-dimensions and based on either threshold or hyperplane assigns it to either of two classes [19]. For example in some cases, it could be useful when trying to determine whether pathology is present in a medical image or not. However, in some cases, just binary classification isn't enough. It may require classification into more groups or assigning multiple labels. This could make the model less accurate though provides more useful clinical information. Another common task, though not present in this thesis, is regression where a continuous variable is being predicted, for example area affected by pathology. This would require different models and approaches (accuracy metric would be different, other type of models would be required, and so on).

It is no news that deep learning technology is used when working with medical images. Understanding how it works and what is needed to create a model to apply in medical fields is an important task in data science.

2.3 Choroidal neovascularization

Choroidal neovascularization (CNV) is an eye disease that makes blood vessels grow through Bruch's membrane into the retina, damaging it and destroying photoreceptors. Leakage of retinal edema and hemorrhage from it threatens vision sharpness. CNV is part of the spectrum of exudative (when liquids from blood vessels escape through pores or breaks in the cell membranes) age-related macular degeneration (AMD), thus, meaning that getting older is the biggest factor, though genetics are also important. Alterations in the normal transport of metabolites, ions, and water through Bruch's membrane in age-related macular degeneration, alter the nutrition and stability of retinal pigment epithelium (RPE) from choriocapillaris and the transport of waste out from the neurosensory retina. Hypoxia leads to VEGF (Vascular endothelial growth factor) being released by the RPE, which initiates a cascade of angiogenic responses at the level of the choroidal endothelium. Bruch´s membrane damage is required to allow the passage of abnormal neovascular vessels from the choroidal vasculature through the breaks in Bruch's membrane to the retina[18].

Research analyzing choroidal neovascularization started over three decades ago. Pretty soon its standard procedure was to use popular mouse strain c57bl/6jrj for noninvasive imaging modalities. CNV can be monitored at different timepoints which allows for a decreased number of animals used. A standard procedure to work with choroidal neovascularization using mice would look like this [26]:

1. Inspect eyes for any abnormalities. Similarly to humans, mice can have unrelated medical conditions that could affect analysis and it is important to take that into account.

2. Weight the animal.

3. Calculate and prepare anesthetics to use, based on the weight of the mice. For it these compounds are used: a mixture of medetomidine (1 mg/kg), ketamine (75 mg/kg), and distilled water (0.9% NaCl solution) at a ratio of 1:1.5:2.5, or ketamine (40-75 mg/kg), xylazine (5 mg/kg), and distilled water (0.9% NaCl solution) at a ratio of 1:2.5:1; for a 20 g mouse, inject 0.1 mL of mixture.
4. Inject anesthetic within the peritoneal cavity (the area that contains the abdominal organs).
5. Place the mouse back to the cage and wait until it's anesthetized, which can be confirmed by the lack of a pedal reflex.
6. Ensure the use of laser safety personal protective equipment.
7. Turn on a slit lamp and a 532 nm diode laser.
8. Remove the mouse from the cage and place it on the heating pad (they are used to warm animals quickly and to use before, during, and after surgical procedures).
9. Apply one drop of tropicamide for pupillary dilation and then wait for 3 to 5 minutes for full (3 mm) pupillary dilation.
10. Place the mouse on the stage of the slit lamp.
11. Place one drop of ophthalmic liquid gel on a coverslip to applanate the cornea.
12. Orient the mouse eye with the optic nerve head in the center.
13. Set the laser power to 100 mW, the duration to 100 ms, and the spot size to 50 µm.
14. Focus the laser beam on the retinal pigment epithelium (RPE). It is a pigmented cell layer just outside the neurosensory retina that nourishes retinal visual cells
15. Make three laser shots into one eye by avoiding retinal blood vessels ideally at the 4, 8, and 12 o'clock positions around the optic nerve, respectively. Inspect the fundus of the eye after all laser shots for the absence of retinal bleeding. The contralateral eye is used as a non-lasered control.
16. Discard the coverslip and place the mouse back on the heating pad.
17. Apply one drop of PEG gel on both eyes.

These 17 steps induce artificial choroidal neovascularization - if there is leakage of fluids around all three shots (mentioned in step 15), the procedure is successful and the animal can be used in studies.

2.4 Data formats

To analyze CNV, some medical images are taken. One of them is fluorescein angiography (FA). Fluorescein is a diagnostic contrast agent. Injected to eye it reveals how blood flows through vessels. For CNV particularly it's useful because it would help in the detection of abnormal blood vessels or even their exudation. Together with fluorescein angiography infrared (IR) images are taken that better reveal the whole surface of an eye instead of just showing blood flow [24]. Figure 2 shows examples of both images. In these images, as well as in VIP for optical coherence tomography (explained later) the middle optic nerve can be seen, also branching out are blood vessels.

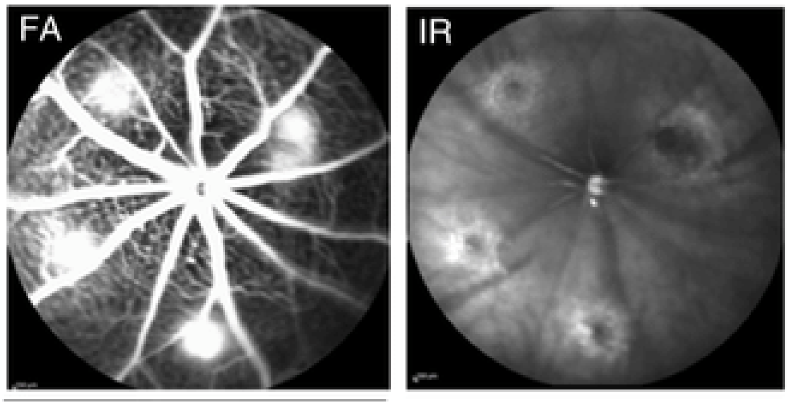


Figure 2: Example of fluorescein angiography (left) and infrared (right) images. In both images, 4 lesions can be seen that imitate CNV where blood vessels grow from choroid to retina damaging it [41]

Before mentioned article [26] also provides workflow on how FA should be imaged:

1. Remove the mouse with the holder and place it on the FA system (for example one of them can be Heidelberg Spectralis HRA2).
2. Focus on laser burn areas of the fundus of the eye using infrared reflectance mode with the head of the optic nerve in the middle of the viewing window.
3. Inject 0.1 mL of 5% fluorescein sodium salt for a 20 g mouse in the fatty tissue, just under the skin or within the peritoneal cavity.
4. Focus on the choroidal level and take an image from the choroidal focus level.
5. Re-focus at the retinal level and take an image.
6. Wait for 30 s and repeat steps 4 and 5.
7. Remove the mouse from the holder and place it on the heating pad.

8. Reverse anesthesia by $\alpha 2$-antagonist for medetomidine, atipamezole (0.5 mg/kg), or simply wait for animal recovery from anesthesia.

9. Repeat in vivo FA imaging in anesthetized animals on the follow-up days, for example, 5, 10, and 14 can be used, depending on the study design.

Another data format that is used when working with CNV is optical coherence tomography (OCT). It is non-invasive imaging tests that use light waves to take cross-section pictures of the retina. Usually, 1000 images are taken for one eye that together make a 3D scan. One such image is 1000x1024 pixels and it's called a b-scan while their combination of them taking one specific height (looking from a b-scan perspective) is called volume intensity projection (VIP). To work more effectively, spectral domain optical coherence tomography (SD-OCT) was developed that make this process faster and more accurate [40].

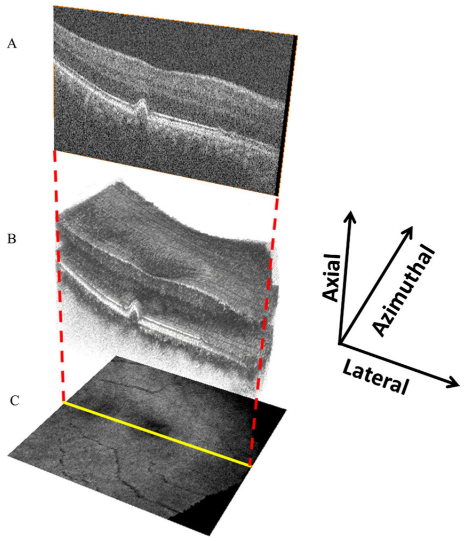


Figure 3: Image to better understand OCT. An image is a b-scan, it shows retina layers and choroid (layers under the retina). B image is more general showing that it's a 3D scan. C image is a projection made using all b-scans, called VIP [13]

Another example that could understand not only OCT but also CNV better is a collage of b-scans showing how lesions progress during days. To imitate choroidal neovascularization, Bruch's membrane must be damaged manually using a laser. This is being done using corresponding devices by professionals so that the procedure would be precise and do as little harm to animals as possible. In a collage of Figure 4 three CNV lesions for one eye can be seen.

In before mentioned article about procedures, workflow for OCT imaging was provided too [26]:

1. Place the animal into the rodent alignment stage and immobilize the head.

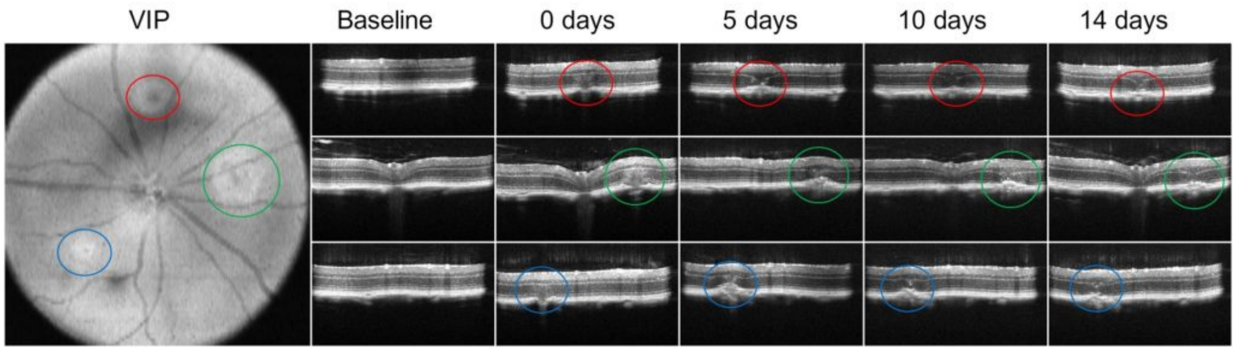


Figure 4: OCT collage. In VIP positions lesions can be seen. Red, green, and blue circles show the positions of each lesion. Next, there are columns for each timepoint (baseline and day 0 is same day just one is before doing lesions, other is after) [26]

2. Align the lens of the SD-OCT system (for example Bioptigen/Leica Envisu R2200) to face the eye for in vivo imaging using X- and Y-stage controllers.

3. Perform SD-OCT scans to verify breaks of Bruch's membrane. Once the SD-OCT scans the whole eye, manually move the reference line on the lasered sites. Breaks of Bruch's membrane should be visible in lasered areas. Those are the three damaged spots seen in Figure 4.

Apart from the segmentation of lesions, another important parameter that would tell how treatment affects pathology is CNV grading. On induction day (usually called day 0), for every lesion a score is determined whether Bruch's membrane was damaged or not (1 - if it was, 0 - no). It is uncommon to have 0 in this case and such laser shots are considered wrong. Then for all the other days where Bruch's membrane was damaged, another score is determined for every lesion observed by comparing dynamics of fluorescein signal in a series of retinal FA images: 0 - the normal appearance of retina, 0.5 - faint staining of leakage, 1.0 - CNV areas are leaky. Such a parameter is important for all time points and lesions because the treatment is supposed to decrease the amount of blood vessels but leaky areas would determine that it's not working. The article suggesting this [26] also made useful advice to use OCT imaging for additional confirmation or additional FA scans where the presence of intraretinal fluid in OCT images would make CNV grading not trivial. Although it's a useful variable when working with CNV, in this thesis it won't be used as the main goal is to test image segmentation and it can't be used as preclinical parameters, because its value on day 0 isn't useful (all lesions are leaking because the wound is fresh).

Using such images together with clinical data choroidal neovascularization can be analyzed. Numerical parameters of lesions' area and volumes throughout timepoints would show the development of disease and could help to discover what might influence it.

2.5 Data fusion

Data fusion is a process where data is joined from multiple modalities to get more information and create better-performing models [10]. Multiple modalities in data science mean that different types of data are used: they can be visual images, tabular data, audio, and so on. In this thesis, only tabular data and images are used. It allows to create pipeline incorporating various types of data and even predictions for more accurate final results. Depending on which level of data is combined, there are three types of data fusion:

- Early fusion, also known as feature level fusion, joins modalities before feeding into one single machine-learning model for training. Different inputs can be fused both by concatenation or pooling and fusing extracted features.
- Joint fusion, also called intermediate fusion, combines learned feature representations from intermediate layers of neural networks with features from other modalities as input to a final model.
- Late fusion, also known as decision-level fusion, is a process of leveraging predictions from multiple models to make a final decision. In other words, instead of combining data modalities, late fusion combines predictions made on said data.

Models structured in more complex ways allow to not only solve more difficult tasks but also lets use various sources of data. One such example is Huang et al. building a model to automatically predict the presence of pulmonary embolism [9]. Data collected by Stanford University was used, medical records were preprocessed and radiology reports were composed and then two models were built: one using clinical data, one using medical computed tomography scans. They were combined to form the final result, whether a patient has a pulmonary embolism. It can be seen that the late fusion type was used there.

Another approach to performing medical research by doing data fusion was made by Lu et al. [20]. A joint fusion model was constructed to predict lymph node metastases of pancreatic cancer. Similarly to the previous example, both clinical data and contrast-enhanced computed tomography scans were taken for input, though in the whole pipeline, few models used medical scans that generated different results which were used as inputs for higher-level models. The final model based on three phases of contrast-enhanced computed tomography and clinical data proved to be better than a simple one using the same data.

In a lot of cases in the medical field, both clinical data and medical images are used, and data fusion allows solving tasks that simple machine learning algorithms wouldn't be able to. It's important to note that the examples mentioned were not used with ophthalmology data which makes the analysis in this thesis more novel.

2.6 Achievements in the field

Before diving deeper into the data fusion, it is important to analyze what has already been done in the field. Several scientific articles were analyzed that could give useful insight into how goals could be achieved better.

Since one of the main sets of images (also sometimes referred to as datasets) will be optical coherence tomography, it would be useful to explore it more deeply. Anran et al. in 2021 summarized some of the achievements when working with OCT and OCTA (optical coherence tomography angiography) [27]. This article explores some of the problems that data scientists face when working with such data. One of them is small sample size which occurs because scans are difficult to make and requires preparation for scientists and expensive equipment. Furthermore, data preprocessing is difficult to standardize because while a lot of deep learning uses more typical data formats (like PNG images for visual-based models or CSV for more numerical-based models), there specific OCT file format is used that requires specific software to visually explore data or very specific libraries to work with it. Even then data itself needs to be processed by aligning and averaging - 3D matrix which is OCT scan has retina height fluctuating due to patient breathing which needs to be made equal across scan and then nearby b-scans are averaged so that they would have less noise and more sharpness [34]. One more issue is results explanation - while understanding how pathology works is one thing, another is portraying final results from 3D scans to 2D graphs that would be simple to understand. These and a few other issues make working with OCT scans more difficult than it would usually be in data science. On the other hand, such data is still widely used for retinal disease detection (for example diabetic macular edema or AMD - age-related macular degeneration) and prognosis monitoring (like response to anti-vascular endothelial growth factor treatment).

One of the neural net models using optical coherence tomography was developed in 2021 where choroidal neovascularization was being segmented [14]. At first OCT scans were pre-processed - aligned and averaged. This was done using custom-made algorithms. Then U-net convolutional neural net model was created using over 1000 annotated b-scans with lesions segmented in them. Such model was proven to be quite accurate and, thus, was used to segment CNV lesions in all b-scans, and then using information about scaling (how many millimeters is one pixel in any dimension), lesion volume was calculated that provides important information on how pathology progresses. It was proven that calculating OCT volumes is a good measure when analyzing how treatment works [36]. There control animals were compared to others that were treated with aflibercept. OCT volume analysis has shown that the highest 25 mg/kg dose significantly decreased CNV leakage area by 42% ($p = 0.046$) at day 5 and on day 14 it was found that groups treated with aflibercept at a dose of 15 mg/kg or 25 mg/kg decrease CNV volume by an average of 53%. It's important to have this information as further analysis in this thesis will be using the fact that aflibercept has a statistically significant effect when treating CNV pathology and differs in results when compared to vehicle - a compound that has no healing properties.

Similarly to before mentioned algorithm, CNV lesion segmentation was done on OCTA too. Wang et al. created a convolutional neural network-based algorithm [37]. Two CNN models were used: one for CNV membrane identification and segmentation, and the other for pixel-wise vessel segmentation. The first one is used to detect whether CNV is present in the image. If it is, the second model segments blood vessels themselves. Even though data is different, articles like this give ideas on how similar problems could be approached.

As mentioned before, fluorescein angiography is an important data format for analyzing choroidal neovascularization. Even back in 1986 when analyzing CNV, FA was used to check if the lesion was leaking [5]. In this study in 28% of lesions, leakage appeared which means that it's not just blood vessels are growing into the retina but also they are leaking blood which has a detrimental effect on wider parts of the retina.

Finally, it would be useful to get more insight into what could cause variability in results. Renard et al. composed a literature review that explores this topic [30]. The authors also propose three main recommendations to address results variability potential issues. The first one is an adequate description of the framework of deep learning, the second - is a suitable analysis of the different sources of variability in the framework of deep learning, and the final recommendation is an efficient system for evaluating the segmentation results. A more precise case in ophthalmology was performed by Choi et al. in 2020 [4]. Creating an algorithm using deep learning, showed that there is a variability of results caused by different researchers. Thus, knowledge like this could provide some information on what is important when analyzing ophthalmology results.

2.7 Data science tools

As the field of data science grows, so does the number of computational tools available to use. These days building a deep learning model can be done in a few lines of code. Data preparation and analysis is also quite automated with a lot of standard functions available. As previous research and results in the field were explored, this subsection is for available specific tools.

Probably the most popular programming language for data science is Python. It is a high-level, general-purpose programming language, whose design philosophy emphasizes code readability using significant indentation via the off-side rule. Deep learning and general workflows can be done using other languages like R, MATLAB, and Java, but in this thesis, Python will be used as it's not only easy to read but also due to its popularity there are a lot of libraries available (version of main libraries are provided in Appendix A).

Before working with actual images it is important to prepare the data and there are tons of libraries to apply. One of the most popular ones is *NumPy* [7] which not only holds powerful numerical computing tools but also has commonly used data structures like numpy array, which works both as a list but also it could be N-dimensional matrix. Additionally, some of the most important mathematical functions are provided, from essential arithmetic mean to complex algebra or trigonometry functions and even more. Another library for data preparation is *pandas* [38]. It's best suited for tabular data. This library provides a commonly used *pandas dataframe* data structure that lets easily and intuitively manipulate values stored in a table: sort, filter, clean, save, and read from files.

Another aspect of data preparation is visualizing it, which could be later used to show results too. One very popular library is *Matplotlib* [11]. With it a lot of various plots can be made, that can also be customized in an intuitive way. This library is often given to know when starting as just one simple line like *matplotlib.pyplot.plt(data)* could simply provide a full graph to show or understand data from. Another framework that's less about making graphs but more bout working with visual data is *OpenCV* [3]. It works with images as data providing a lot of useful functions to alter them like ones used in morphological image processing, intensity transformations, and so on.

While for data exploration and cleaning some of the libraries ease the job, the most essential part is building a model. There are tons of possible models to build depending on the task, though for this thesis deep learning models are most important, especially the ones working with visual data. One of the most popular deep learning frameworks is *PyTorch* [22], which was created in 2016 by *Meta AI*. It provides a user-friendly front-end, distributed training, and an ecosystem of tools and libraries to enable fast, flexible experimentation and efficient production. For anyone wanting to start to use deep learning in data science, *PyTorch* would provide simple workflows to train models for various tasks. Another important framework is *Tensorflow* [2] created in 2015 by *Google*. Similarly to *PyTorch*, *Tensorflow* provides pre-trained models or simple solutions on how to build your own, and another advantage is that it can be deployed basically everywhere: on the web, in mobile environments, and on servers. Additionally, *Keras* [6] library was developed on top of *Tensorflow* that aims to make the code more simple and easier to understand. Another library that may not be as popular as others but still useful, especially for visual data, is *fastai* [8], which was built on top of *PyTorch*, in order to make model training both more flexible and better performing, using its carefully layered architecture.

It seems like the the two main frameworks are *PyTorch* and *Tensorflow* which are the basis of other popular libraries. Madhavan et al. in 2021 compared both in various aspects [21]:

- Ease of use. *PyTorch* is better at this with its intuitive functions and good integration with Python. Meanwhile *Tensorflow* isn't integrated that well, and even though it has been updated into 2.0 version incorporating *Keras*, it's not that smooth to use.

- Computational graphs. *Tensorflow* uses static computational graphs while *PyTorch* uses dynamic ones. What it means is that with static computational graphs user first must define the computation graph of the model and then run the machine learning model while dynamic type lets alter them.

- Debugging and introspection. *PyTorch* allows any Python debugger while *Tensorflow* users must familiarize with *tfdbg* (TensorFlow debugger).

- Data visualizations. Both frameworks provide their own tools to visualize machine learning models, though for *Tensorflow* it's considered better.

- Data parallelism. *PyTorch* allows using multiple GPUs (graphics processing units) with little effort for model training while in *Tensorflow* defining data parallelism requires a lot of effort.

- Deployment. *Tensorflow* has built-in tools to be applied in various environments. For *PyTorch* these tools are not as good and might cause more problems.

- Community support. Both have huge communities, *Tensorflow* is much bigger in software production, *PyTorch* in research. *Tensorflow* seems to have more tutorials and might be more popular in general, but both have huge support nevertheless.

Overall, it seems that *PyTorch* is better with all factors, except deployment and community support, though since this paper is in the field of research, not software development, this advantage isn't that useful. Consequently, *PyTorch* is a better deep learning framework for this project, and more specifically *fastai*, which was built on *PyTorch*, will be used.

There are other libraries that could be used. Some are alternatives to before mentioned ones, some are supplements that would add additional functionalities, though those mentioned in this subsection make the core of data science workflow.

3 Data overview

In this section, input data will be reviewed. Since data fusion will be performed, it's important to have a good understanding of both medical images and preclinical data.

All data is provided by Experimentica Ltd.[1] Author works in this company and, thus, was allowed to use any data owned by the company. No data related to external sponsors was used. Data is not publically available because it's the company's property.

3.1 Preclinical data

There are a total of 633 mice that were used to test CNV pathology. The oldest ones were born in 2020 Q3, thus, all analyses are relatively new. There are two main tables. One is at the level of animal (each row is different animal), other has multiple rows containing measured values for each combination of animal, timepoint, and lesion (one of three).

3.1.1 Information about each animal

The animal table has 36 columns though not all of them are useful. These are the most important ones:

- study ID - in this context study is one project where a set of animals (usually from 50 to 100) are taken and all get treatment at the same time and screened at the same time. Treatment groups are formed by sorting the same set of animals into groups and results are usually analyzed from just one study. While studies are separate projects, selected ones can be analyzed together because all analyze CNV in mice and they follow similar timepoints structure. There are 9 unique studies in the dataset.

- animal ID - an identifier for a specific animal. Usually, those consist of the study name and natural ascending number joined with a hyphen.

- vendor - shows a company that grows and provides animals. This might be important as initial breeding conditions and animals themselves might affect results. In data only two were present: "Janvier" and "Envigo".

- birth date shows when the animal was born.

- compound - which chemical compound was used to treat the pathology. One of the most important ones is Eylea which is a more common name for aflibercept and as it was mentioned before this compound has a significant effect when treating CNV[36]. Another important compound is a vehicle: its purpose is to be a blank treatment that has no healing properties and could show how results would look if the compound did not work at all. These two are very useful for each study because if there is a third compound that is being tested, it would show whether its performance is closer to actual medicine or an ineffective healing compound.

- cohort: since studies can have up to 100 animals, scientists can't do all scans on the same day, thus, the whole set is separated into a few smaller sets. While the difference between timepoints are same between the same set of animals, the cohort allows to shift of the exact date for some of them. In the data maximum number of cohorts was 3.
- treatment date shows when treatment was started.
- induction date - the date when CNV was induced. On this day scientists make laser shots where Bruch's membrane is damaged (layer separating retina and choroid) and CNV is imitated in three different spots. For rats, it's usually 4 (which can be seen in Figure 2) but in this analysis only mice were used and all had 3 shots done.
- excluded - parameter whether an animal was taken for final analysis. During study animals can be removed for various reasons: they might die too early, their retina might become too physically distorted, or other unexpected reasons. For this analysis excluded animals won't be used.
- treatment group - since compounds might vary in naming across different studies, they were all sorted into a few categories: Eylea, Other, Vehicle, Unsorted, and Untreated. Eylea represents all compounds with aflibercept, Vehicle means all compounds that have no healing properties, other treatment groups won't be used as they are much smaller or too varied inside.

Some parameters weren't mentioned because they are equal for all animals, for example, all of them are from the c57bl/6jrj strain and all of them are male.

3.1.2 Measured values

Another important table, which has more information about each lesion, has 57688 rows and 32 columns. While the previous table has more general information that could tell more about the study and animals in general, this one gives a plethora of data to work with. Since not everything will be used (because only data for annotated images will be taken), it's important to look through the most useful columns and some of the values that would give a better view of how the variable looks with a bigger sample size.

The table is constructed in a way that it would be easy to make a pivot table out of this: there are set of columns that are identifiers or could be useful as filters (like animal ID, timepoint, treatment group, lesion ID and others) and then there is a column for parameter and then few for value (where possible some are measured both in pixels and in millimeters). This allows easier data analysis with tools like Microsoft Excel instead of always relying on codes.

These are some of the most important parameters:

- CNV (choroidal neovascularization) volume - since OCT scans are 3D representations of the retina, it's possible to measure the lesion's volume. Usually, they are shaped like cones with blunt tops, though often might have cavities (empty spaces inside) or additional sprouts. There are a total of 4777 values, Figure 5 shows their distribution. Looking at

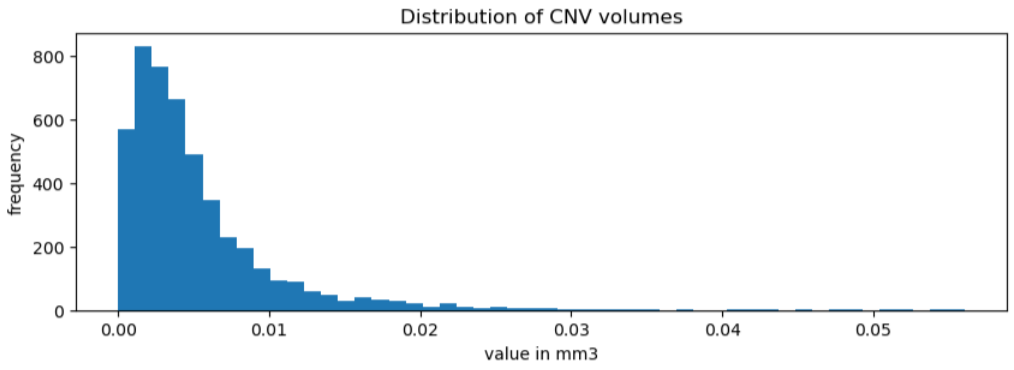


Figure 5: Histogram showing how CNV volume values are distributed in input data, x label shows values in cubic millimeters, y label shows frequency.

  the histogram it seems that values are distributed to log-normal distribution. The average value is equal to 0.0052, though it's obvious that if a lesion isn't healing well, its volume can stray quite far from that.

- FA area - another significant parameter is the lesion's leakage area in fluorescein angiography (FA) scans. While OCT volume portrays better physical damage and lesion growth, the FA area shows another aspect of CNV - blood flowing out of fragile neovascularization structure. While some areas would be visible even without it, a huge amount of blood leaking out would increase area. Figure 6 shows a distribution of 7093 values with a mean of 0.0475. It seems that the FA area is also distributed in log-normal distribution, like CNV volumes. This might indicate the general tendency of lesions: based on general physical parameters most have similar quite low values and then usually due to ineffective healing some tend to grow much bigger.

- Spot area in an infrared image - while FA area shows leakage area, this parameter shows a physical area of the lesion, though in an infrared (IR) image. Also, compared to OCT, since OCT is a 3D scan, the physical lesion's area might depend on how VIP (image from the same angle as FA and IR) is made. Infrared is another technique that portrays the retina not being influenced by blood flowing or additional image construction algorithms, thus, spot area in infrared is another physical measure that could be used for analysis. Figure 7 shows all 2739 values that has a mean equal to 0.285. It's obvious that a few huge

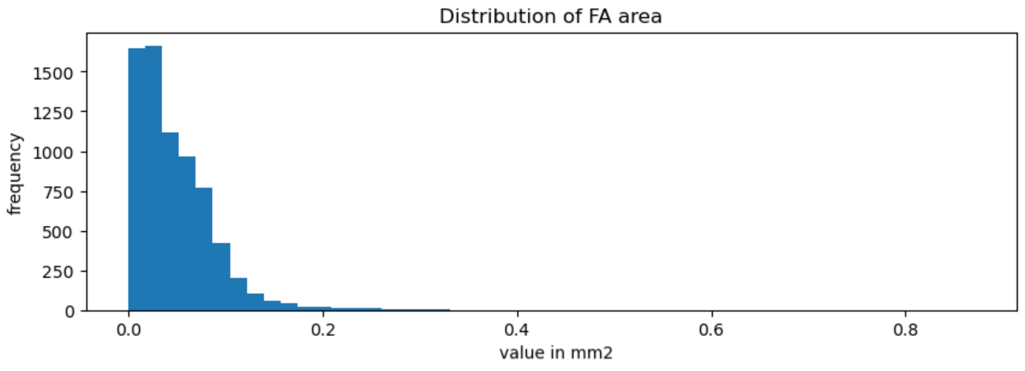


Figure 6: Histogram showing how FA leakage areas are distributed in input data, x label shows values in quadratic millimeters, y label shows frequency.

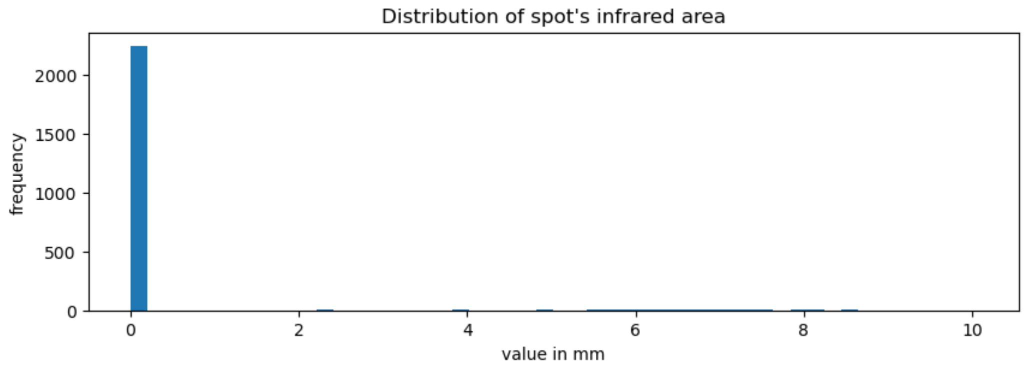


Figure 7: Histogram showing how lesion's areas in infrared images are distributed, where x label shows values in millimeters, y label shows frequency.

values make the whole histogram difficult to read. There are 99 values bigger than 0.5. If those were filtered, values would have a much more comprehensible histogram. Figure 8 shows exactly that. While this could look like a normal distribution, overall this can't be accepted as there are much bigger values, also following arguments from the previous point, it's obvious that most of the values are quite low, except this parameter has much fewer values that stray far from the mean.

- Distance to the optic nerve - one of the parameters that could influence how the lesion is healing, is how far is the lesion from the optic nerve. In a mouse's eye optic nerve is in the middle and the net of blood vessels goes to all sides (similarly like in the human eye) and near optic nerve blood vessels tend to be thicker and there are more of them while they decrease in size and frequency further from the nerve. Also, it's important to note that distance is measured from the lesion's center to the center of the optic nerve. Figure 9 shows the distribution of such values. While values will never be below 0, overall distribution in general reminds of normal distribution. There are a total of 2703 values

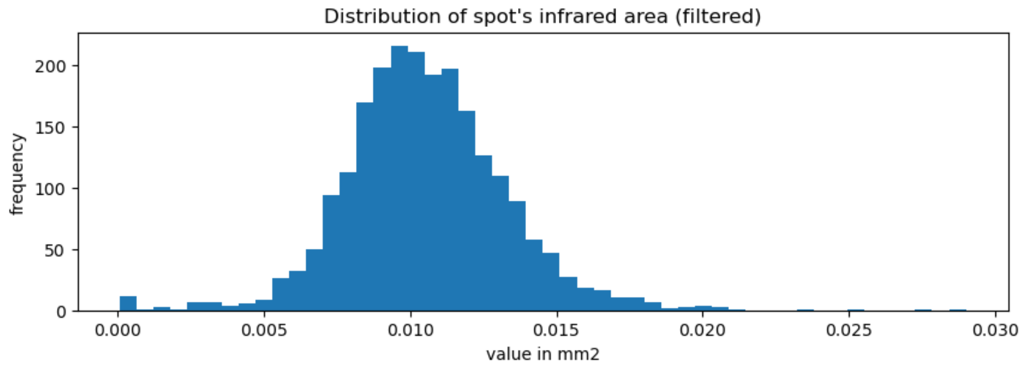


Figure 8: Histogram of lesion's areas but with filter leaving only values smaller than 0.5.

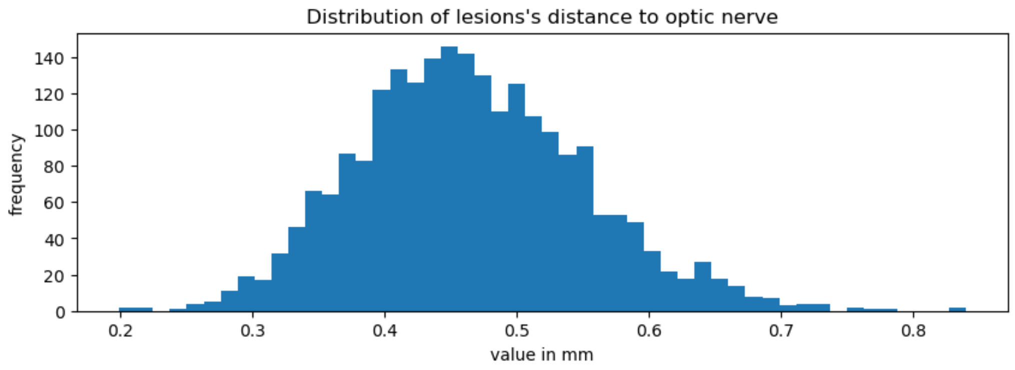


Figure 9: Histogram showing how lesion's distances to the optic nerve are distributed in input data. X label shows values in millimeters, y label shows frequency.

with a mean equal to 0.466±0.087.

- Distance to the nearest blood vessel - one more parameter is the distance from the lesion's border to the nearest blood vessel (also its border). Similarly to distance to the optic nerve, the lesion's position and closeness to blood vessels might influence how well it is healing. Though it's important to note that this measures only distance to one of the major blood vessels, for example in Figure 2 in the FA image 11 major blood vessels can be seen, and while the whole eye has a net of them, this parameter ignores them as they are much less important. Figure 10 shows how values are distributed. There are a total of 2385 values with a mean equal to 0.0301. It's important to note, that if a lesion is on a blood vessel, the distance is equal to 0 and this can be clearly seen in the histogram. Overall, it's difficult to pinpoint in what way values are distributed, though gamma distribution with k being equal or less than 1 seems like one of the closest options.

In order to illustrate parameters in the infrared image, Figure 11 was created. It provides a basis to understand how distance to the nearest blood vessel, distance to the optic nerve, and

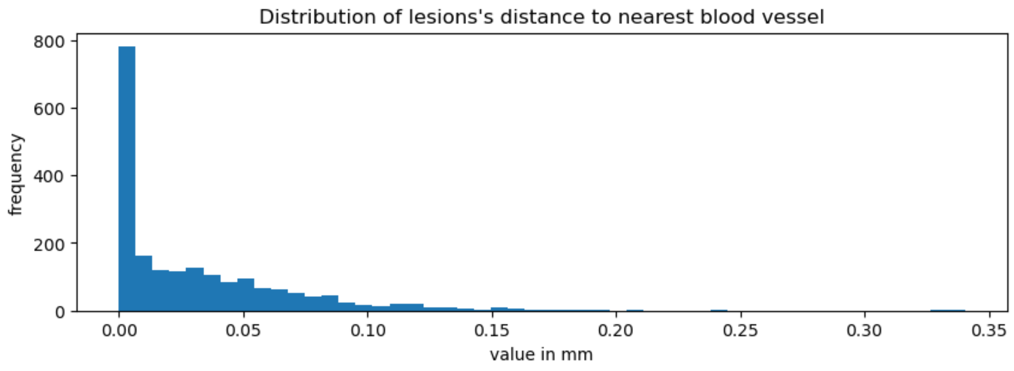


Figure 10: Histogram showing how lesion's distances to nearest blood vessel are distributed in the data. There x label shows values in millimeters, y label shows frequency.

lesion area is measured. While there are lines for the two closest blood vessels, only distance to the first was used.

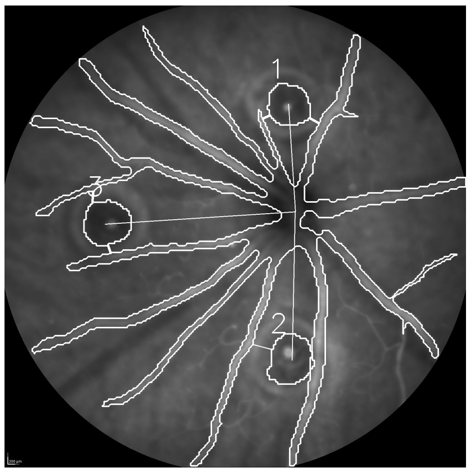


Figure 11: Image illustrating how preclinical parameters are measured. There lesions and blood vessels are marked manually to illustrate important objects in the image, also lines show either the distance to the two nearest blood vessels or the distance from the lesion's center to the optic nerve. Furthermore, numbers can be seen, which helps identify lesions.

Measured values can be separated into two groups here: lesion areas in OCT b-scans and FA areas are values that will be predicted with further algorithms and a comparison will be made. Taking the whole OCT volume is not possible as that would require having annotation of all images in one scan - 100 images would only give information about one animal from one timepoint. The other three parameters will be used as input. Due to this, spot area in infrared image, distance to nearest blood vessel, and distance to optic nerve are only from day 0 because the intention is that these are taken as information for animals and then always same value is used. On the other hand, OCT volumes and FA area are predicted only for later days (the earliest is the 3rd) not just because only then CNV lesion is formed but also so that there would be no dependency that some scans must be made before others. For the latter reason, FA area and lesion areas in OCT b-scans also won't be mixed together (one won't be input for another).

3.2 Medical images

This subsection reviews medical images that were annotated. It's important not only to show how they look but also to look at their respective preclinical data.

3.2.1 OCT data

Each mouse has OCT (Optical Coherence Tomography) scans number of timepoints that were performed, the maximum can be 5. A number of timepoints depend on the study. While usually, OCT scans have two more timepoints: baseline which is for undamaged eyes, and day 0 for recently damaged ones, they are ignored there as the former neither have CNV lesions (though the latter has places of laser shots visible). There are a total of 599 b-scans annotated. Figure 12 shows one of the annotations. There 2 different lesions that can be seen, also it's obvious that there are huge cavities - it's important not to segment them as otherwise, they would greatly increase area making a significant error.

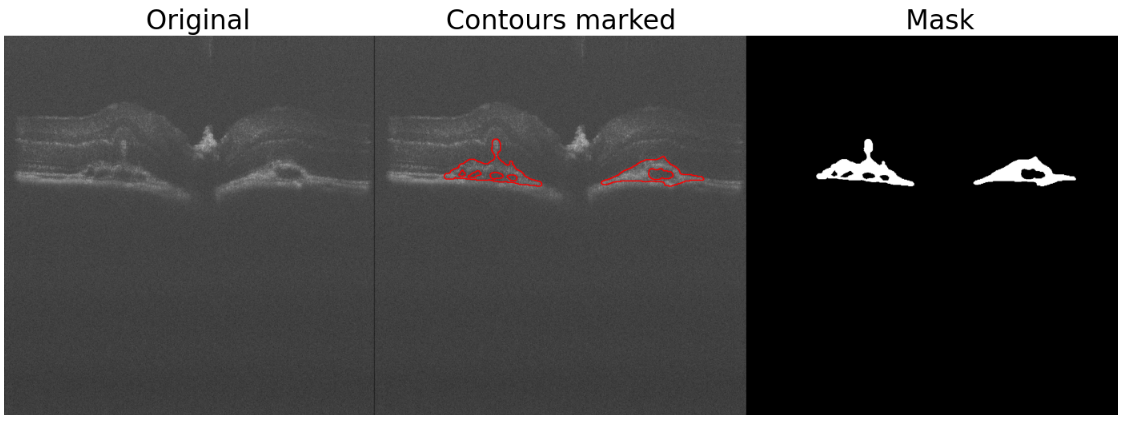


Figure 12: Three images showing one of OCT b-scan annotations. On the left is an original image, on the middle annotated contours were marked, on the right - mask of the same image.

A mask is an important thing to understand. For image segmentation, a mask is an image that has the same dimensions as the original image it represents that has regions of interest marked. In semantic segmentation, each pixel's value represents which category it belongs to: 0 is usually reserved for background and in this case, 1 means CNV lesion. Bigger values could mean other categories though for this task (segmentation for both b-scans and FA images) there will be only these two categories.

With these annotations, it's important to look at how annotations are distributed based on preclinical data:

- Vendor: 505 b-scans has "Janvier" as a vendor, 20 has "Envigo". Clear dis-balance can be seen. Across all animals 13% of them are from "Envigo", thus, the difference is not as huge, but still relevant. Since the vendor is a categorical variable where no category has more importance, in further algorithms, the weights of both vendors will be equal.
- Cohort: 354 was in 1st cohort, 80 was in 2nd, 91 had no information. For this parameter and all others that had no parameter, if no information is provided, it's treated as a separate category, that defaults to the most generic method, for example for early fusion no processing is made.
- Difference between birth date and induction date in days: 67 days were for 65 b-scans, 68 for 27, 71 for 167, 74 for 13, 75 for 78 b-scans, while 175 b-scans had no information.
- Distance from lesion to the optic nerve in pixels: mean 242.64±26.9, minimum 182, maximum 313.
- IR area in pixels: mean 2599.58±488.33, minimum 1538, maximum 3716.
- Distance to the nearest vessel in pixels: mean 17.7±18.62, minimum 0, maximum 64.0.

In order to have a better comparison between models, it's important to have a test dataset. For this, 74 OCT b-scans were selected (out of the initial 599 that weren't used for training or validation) that have such a distribution of preclinical parameters:

- Vendor: 39 b-scans have "Janvier" as a vendor, and 35 have "Envigo".
- Cohort: 55 were in 2nd, 19 were in third.
- Difference between birth date and induction date in days: 68 days were for 35 b-scans, 72 for 20, and 73 for 19 b-scans.
- Distance from lesion to optic nerve in pixels: mean 243.69±20.3, minimum 207, maximum 267.
- IR area in pixels: mean 2701.55±640.4, minimum 1745, maximum 3956.
- Distance to the nearest vessel in pixels: mean 23.49±12.8, minimum 2, maximum 43.

3.2.2 FA images

There are a total of 664 FA (fluorescein angiography) images annotated. Figure 13 shows one of the annotations.

It can be seen that the view is different and while b-scans usually portray one lesion and sometimes 2 that can have the same or bigger number of contours, the vast majority (85%) of

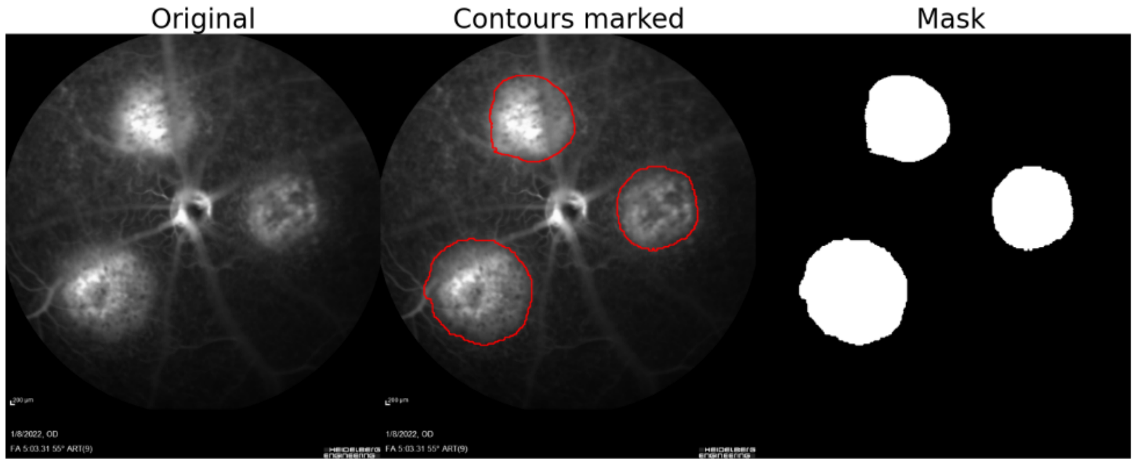


Figure 13: Three images showing one of FA image annotations. Same with OCT b-scan, on the left is the original image, on the middle annotated contours were marked, and on the right it is a mask of the same image.

FA images have 3 spots (this only applies to contours in the mask, not actual number of lesions in eye).

Similarly to OCT data, let's look at how annotations are distributed based on preclinical data:

- Vendor: 466 b-scans have "Janvier" as a vendor, 192 have "Envigo", 6 have no information. As it was mentioned before, 13% of all animals are from "Envigo", while in this case, it's 29%. While such input doesn't represent whole data, at least class dis-balance is not as significant, thus, no additional measures are taken to fix it.
- Cohort: 204 was in 1st cohort, 218 was in 2nd, 65 was in 3rd, 177 had no information.
- Difference between birth date and induction date in days: mean 62.07±6.7, minimum 54, maximum 69.
- Distance from lesion to optic nerve in pixels: mean 304.52±134.9, minimum 183, maximum 630.
- IR area in pixels: mean 5069.29±5078.1, minimum 1432, maximum 19901.
- Distance to the nearest vessel in pixels: mean 19.5±16.6, minimum 0, maximum 83.

For the test dataset, 51 images were selected that had a similar distribution of preclinical parameters.

Just to clarify, annotated images for OCT b-scans and FA images were from different animals. This could have been indicated by the fact that preclinical data is different.

4 Analytical part

This section explores how data fusion was done. At first, initial segmentation was described that would be the baseline for how the model could perform. Later different methods of data fusion will be explained and show how they predict compared to the initial model. Finally, it's essential to do a comparison that would show if data fusion improves segmentation and which approach is the best. All algorithms firstly will be tested using OCT b-scans, and then the same practices will be applied to FA images in 5.4.

4.1 Simple segmentation model

To know whether data fusion improves segmentation, it's important to have a model without any fusion.

4.1.1 Initial segmentation

Semantic image segmentation was done using *fastai* [8] library coding with Python (versions for most common libraries are provided in Appendix A). To do this, a dataloader was created using *SegmentationDataLoaders.from_ label_func()* function that was given these parameters:

- folder containing all images and their masks;
- batch size - 32 was selected for optimal segmentation;
- seed equal to 42 for random splitting;
- a parameter that shows which proportion of the dataset will be used for validation, in this case, 0.1 was selected as there are not a lot of images so a better model was prioritized instead of more accurate validation results;
- list of images to use (as list of Path objects);
- label function that is usually used to define labels for prediction, though this time function would give a path where the corresponding mask file is saved,
- item transformations - how to transform each image for a model, which for this model is equal to a list with one Resize object with value 224. It means that each image was resized so that it would be a size of 224x224 pixels. This means that the output mask will also be in that size. This was done to make the model faster without losing too much information and doing this would decrease noise.

After that, a learner object was created using *unetlearner* object providing previously described dataloader object, model architecture - ResNet34 was selected for this task, *n_ out* parameter equal to 2 which means that output should have only two categories (background and CNV lesion), and finally for metrics parameter *Dice* was chosen.

Talking about metrics, the Dice coefficient is a very important metric in image segmentation that shows how well segmentation performs on average in all categories [12]. Because in input images vast majority of the area is background, it's important to measure not how many pixels were segmented correctly but rather check how well the lesion was found. Dice metric works in a simple way. For each category two times the intersection area is measured for annotation (ground truth image) and prediction then it is divided by the summed area of both. Finally, the mean of all fractions is calculated and that's equal to Dice. The maximum value is 1 while the minimum possible is 0. Dice metric works in a way that it punishes the model not only for marking wrong pixels but also for failing to segment all of the true area.

$$\begin{aligned} Dice_{category_i} &= \frac{2 \times (prediction_i \cap annotation_i)}{prediction_i + annotation_i} \\ Dice_{final} &= \frac{\sum_{i=1}^{k} Dice_{category_i}}{k}. \end{aligned} \tag{1}$$

Here i is a number of a specific category, $prediction_i$ and $annotation_i$ areas of category i from either prediction or annotation images respectively, k is a number of categories, in this case it's equal to 2.

Finally, 20 epochs were selected as an optimal number for training since then the model stops improving and it's important not to overfit. Unless specified otherwise, all further models will be trained using the previously mentioned parameters. Part of the code is provided in Appendix B.

Training results show **0.902** Dice coefficient. Figure 14 shows visual results from some of the images used for validation.

It seems that the model performs quite well as CNV lesions in b-scans might look quite differently from each other - size might vary a lot, some have cavities, and some might physically distort the rest of the retina making segmentation more difficult.

4.1.2 Testing data augmentation

The previously mentioned loader function has other optional parameters, for example, parameters for batch transformations (*batch_ tfms*) that would transform images differently for each training batch which might increase accuracy. Before accepting training parameters as final, there was an interest in trying batch augmentations as these could make the model more generalized.

Three were selected:

1. simple built-in augmentations. These involve changing brightness, zooming in, and slight rotating. It was achieved in dataloader setting parameter *batch_ tfms* equal to *aug_ transforms()*.

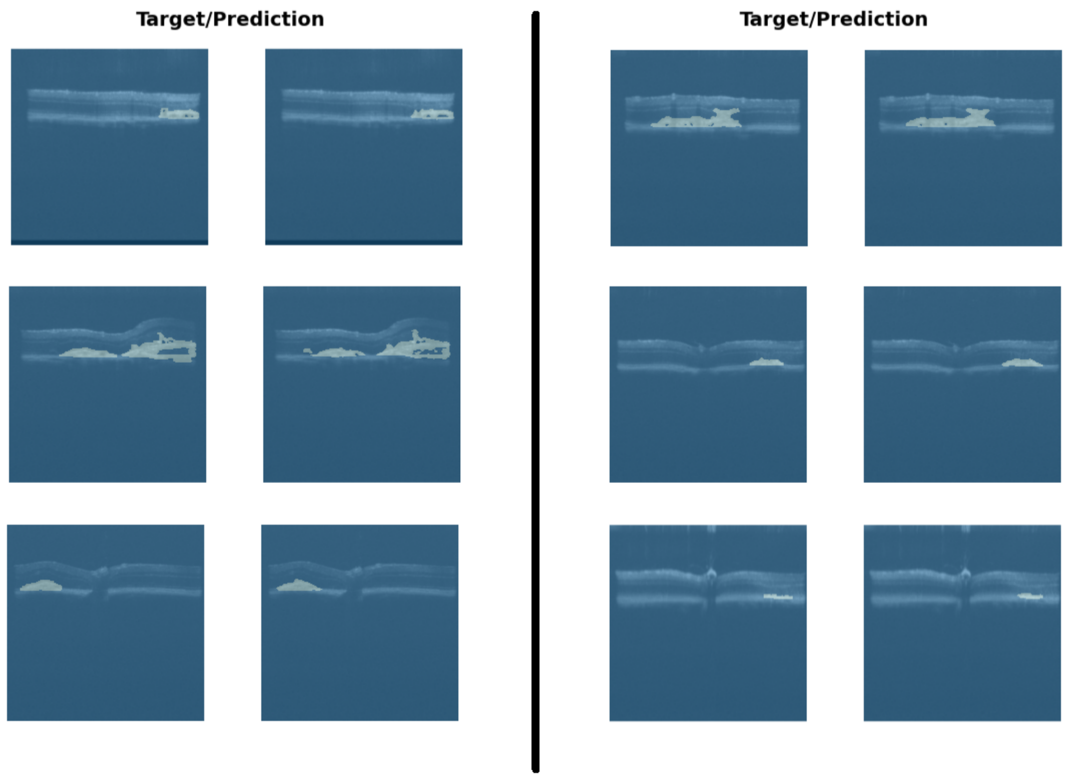


Figure 14: Image showing initial segmentation results. Two parts are from the same set, just images were compacted to have more images in less space. The target column shows how the image was annotated and the prediction shows the model's attempt.

2. Flipping image - half of the images were horizontally flipped. It wouldn't make sense to flip images vertically as the retina in b-scans is never upside down. This was implemented using *albumentations* library.

3. Scale and rotate - half of the images were randomly rescaled and rotated. Implementation was also done using *albumentations* library.

For each transformation neural net was trained using before mentioned parameters. To compare models, each Dice coefficient was calculated for test images. Then these results were compared using the Wilcoxon rank-sum test. The Wilcoxon rank-sum test is a nonparametric test that is often used as an alternative to the two-sample t-test. It is based solely on the order in which the observations from the two samples fall. Observations from both samples are ranked together. Each observation has a rank where the smallest has a rank equal to 1, the 2nd smallest rank 2, and so on. The statistic for the test is the sum of the ranks for observations from one of the samples. Then p-value is calculated by getting the probability of having that sum of ranks if both samples would have the same sums [39]. All further tests will be two-sided where the null hypothesis means that distributions of both results are identical and the alternative hypothesis would show that they are not. A significance level equal to 0.05 is used.

Figure 15 shows a comparison between initial segmentation where rows show which model is compared and columns with which other model.

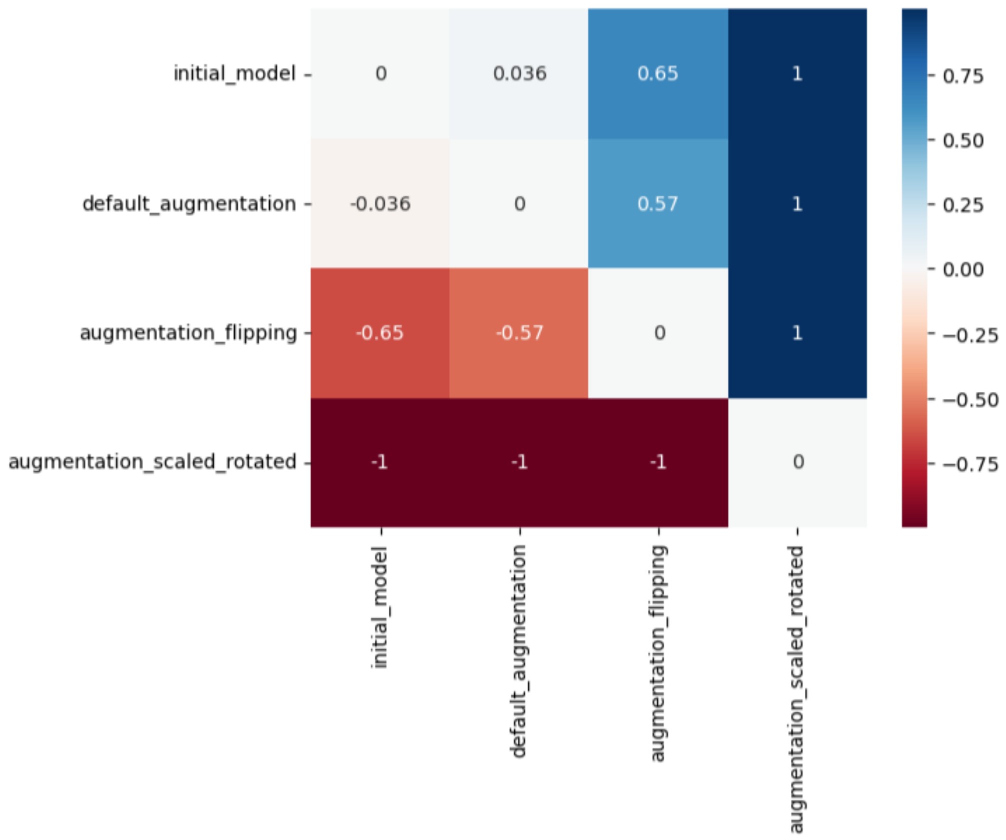


Figure 15: Comparison between augmentations. There p-value was transformed by subtracting it from 1 and multiplying by -1 if the sample's mean is lower. So values above 0.95 would mean significantly better results, as well as values below -0.95, and all others show no significant difference though could help in providing the general image. Also, from the color scale, it can be seen that blue means higher values and better-performing models while red means lower values.

The first obvious thing to notice, scaling and rotating are much worse than all other methods. Secondly, the initial model is better than all others, even if the difference is not significant. Default augmentation barely has any differences but because it's still slightly lower, it was decided to not do any augmentation further on.

4.2 Early fusion

The image segmentation algorithm for early fusion was inspired by R. Harrabi and E. B. Braiek [25]. The idea of their algorithm is that images from different colour channels are fused using mass functions and multi-level thresholds. The data used wasn't part of ophthalmology. Since this thesis uses deep learning, instead of a statistical approach, a bit more complex one was done. Figure 16 explains how the data fusion algorithm used in this thesis works. First, two types of data are prepared: preclinical data and ground truth images. Then for each preclinical parameter image's colour channels are modified based on values of preclinical variables. Then segmentation model is trained for each using the same masks as with the original images. Finally, when a new image needs to be segmented, at first task is separated into a number of parts how many preclinical parameters are used: for each image's colour channels are modified in the same way as in training, segmentation is done with the neural net model, and after such segmentations are made, all are combined by iterating with 3x3 window and taking mean value for each pixel across all segmentations. A more thorough code is provided in Appendix C.

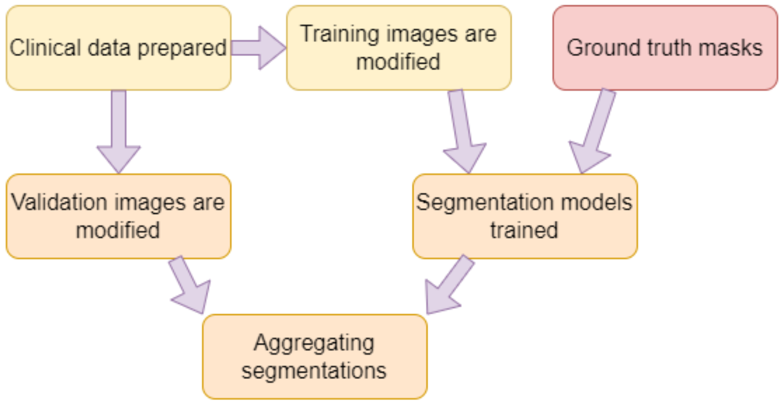

Clinical data prepared
Training images are modified
Ground truth masks
Validation images are modified
Segmentation models trained
Aggregating segmentations

Figure 16: Scheme explaining how early fusion algorithm works. Yellow blocks are meant to represent where only preclinical data is used, red blocks are for only medical images, and orange is meant to represent algorithm parts where both types of data were necessary

As it was mentioned, each preclinical parameter has its own modifications and segmentation model. Depending on the value, a coefficient was calculated and each image's red channel was multiplied accordingly. Table 1 explains how the coefficient for each preclinical parameter was calculated, and additionally when the image segmentation model was trained with modified images, what the Dice coefficient was equal to. Coefficients ranging from 0.75 to 1.25 were selected in a way that preclinical parameters would influence how the image looks but the change wouldn't be drastic, so these values should make a balance between those two sides.

Using all these models image segmentation was done. The algorithm took all annotated images in the test dataset, for each separately segmented lesion on modified images and combined them. The final Dice coefficient is equal to **0.728** which is worse than the initial model (p-value 3.9e-05 from the Wilcoxon rank-sum test). It could be possible that modifying colour channels make this task more complex or there are models who perform segmentation poorly and, thus, ruin the combined result.

To check the latter hypothesis, the algorithm was run with different combinations of clinical parameters. Table 2 shows various combinations of preclinical parameters, what Dice each has

Table 1: Explanation how coefficient to change red colour's channel was calculated - value before hyphen means preclinical parameter's value, a number after hyphen is coefficient's value. Also, since the image segmentation model was trained for each parameter, this table also shows the Dice coefficient for each model.
*Table only shows minimum and maximum value, all others in the range have coefficients assigned accordingly in a linear way

Preclinical parameter	**Colour coefficient**	**Model's dice**
Vendor	Janvier - 0.75, Envigo - 1.25	0.847
Cohort	1 - 0.75, 2 - 1.25	0.86
Diff. birth date-induction date	67 - 0.75, 75 - 1.25*	0.826
Lesion's distance to optic nerve	185 - 0.75, 285 - 1.25*	0.882
IR area	1500 - 0.75, 3718 - 1.3045*	0.863
Lesion's distance to blood vessel	0 - 0.75, 32 - 1.25*	0.852

Table 2: Dice coefficients for various combinations of clinical parameters when using early data fusion for image segmentation in OCT b-scans

Combination of parameters	**Dice coefficient**	**P-value**
All except vendor	0.619	6.3e-20
All except cohort	0.816	7e-5
All except diff. birth date-induction date	0.696	3.2e-11
All except distance from lesion to optic nerve	0.714	9.2e-9
All except IR area	0.71	1e-9
All except distance from lesion to blood vessel	0.711	2.5e-9

achieved, and also the p-value from the Wilcoxon rank-sum test for each compared to initial segmentation.

Even though trying combinations improved the model (Dice coefficient being equal to **0.816**), based on all of the Dice coefficients and p-values, the early fusion algorithm doesn't improve image segmentation for OCT b-scans.

4.3 Late fusion

For late fusion algorithm by Gupta et al. was adapted [33]. The general scheme is provided in Figure 17. For a wider explanation, the code for the algorithm can be found in Appendix D.

At first ground truth images are prepared. They are used for two purposes: annotated lesions are used for training and a semantic segmentation model is created using them. For later purposes same model as the initial one (described in 5.1.1) was used. Annotated lesions are important for training the late fusion model because for each image one or more bounding boxes were created in a way that it's limited by the most left one, most right one, highest, and lowest points. Such a rectangle will always be parallel to the image's borders. Another step for each in such a rectangle is to calculate parameters or take previously mentioned preclinical

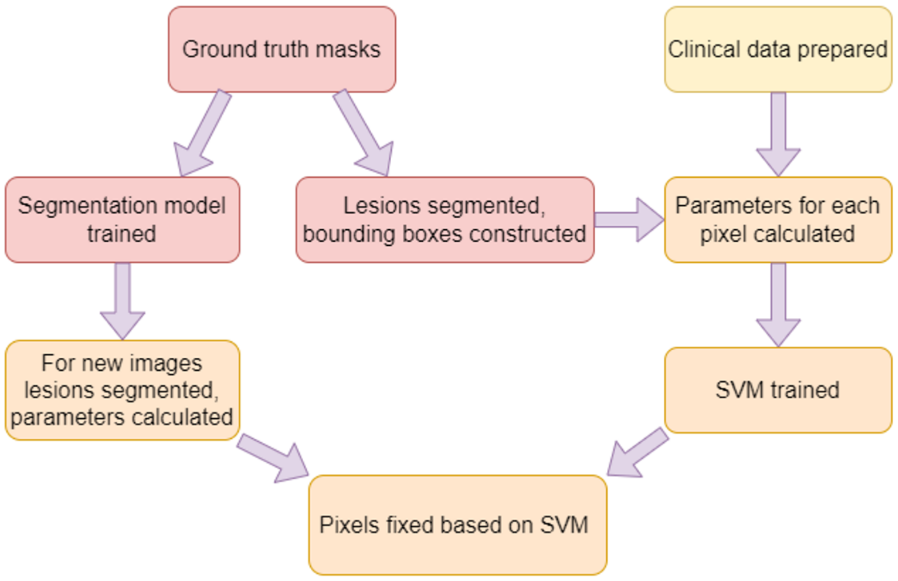

Ground truth masks
Clinical data prepared
Segmentation model trained
Lesions segmented, bounding boxes constructed
Parameters for each pixel calculated
For new images lesions segmented, parameters calculated
SVM trained
Pixels fixed based on SVM

Figure 17: Scheme explaining how late fusion algorithm works. As with the previous scheme, red blocks represent steps where only medical images were used, yellow is where only preclinical data was used and orange is for both.

parameters. In total 14 variables were used:

1. Brightness: pixel value
2. Mean brightness: in a 3x3 window centered on a pixel mean value was calculated.
3. Variance: similarly to the previous, the variable for current and neighbour pixels was calculated.
4. Angle: it's a parameter that shows in which way from the current point pixels are brighter. While retina layers vary in brightness, CNV lesions, in general, are brighter than the background or at least surrounding pixels (both in OCT b-scans and FA images), thus this parameter could show whether the current pixel is at the bottom of the lesion or maybe at the side. It's calculated in a way that there are 8 matrices created that have coefficients of either 1, 0, or -1. To get the angle, neighbouring values are multiplied by these coefficients, summed and it's important with which matrix the highest value will be achieved, such angle would be taken accordingly. Figure 18 illustrates some of these matrices. The current pixel's coefficient is always 0.
5. Contour height: bounding box's height in pixels.
6. Contour width: bounding box's width in pixels.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -1 & -1 \end{bmatrix} 0^\circ \qquad \begin{bmatrix} 0 & 1 & 1 \\ -1 & \boxed{0} & 1 \\ -1 & -1 & 0 \end{bmatrix} 45^\circ \qquad \begin{bmatrix} 1 & 0 & -1 \\ 1 & \boxed{0} & -1 \\ 1 & 0 & -1 \end{bmatrix} 270^\circ$$

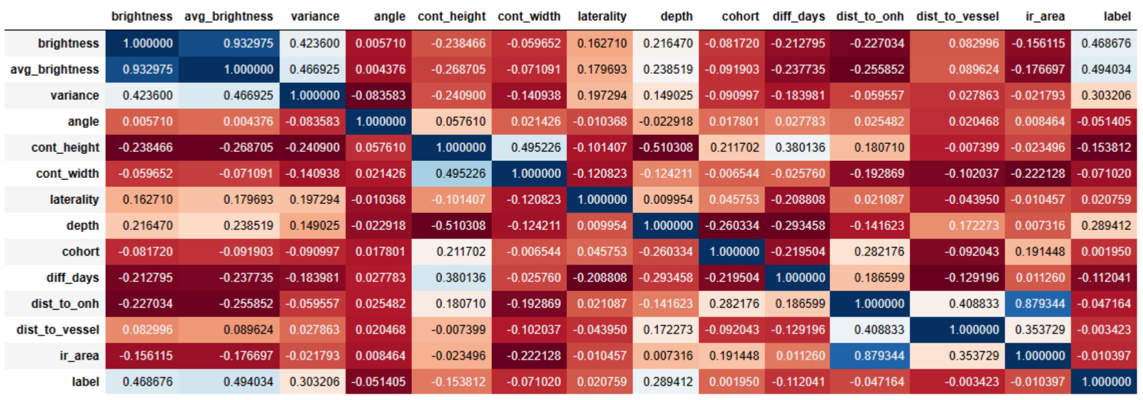
Figure 18: Some of the matrices that were used to calculate angle variables. If neighbour pixels multiplied by the first matrix would have the highest result, the angle would be equal to 0°, if multiplication with the second matrix would have the highest value, the angle would be equal to 45°, if with third matrix - 270°

7. Laterality: pixels relative position in the image horizontally (it's a continuous variable); 0 means that the pixel is on the left side, if 1 it's on the right side.
8. Depth: pixels relative position in image vertically; 0 means that pixel is at the very top, 1 means it's at the bottom.
9. Vendor: this and the next five parameters were taken depending on which animal was used.
10. Cohort
11. Difference between birth date and induction date in days
12. Distance from lesion to optic nerve in pixels
13. IR area in pixels
14. Distance to the nearest vessel in pixels

The final column is ground truth value: 0 if it's background, 1 if it's lesion. Using all this data Support Vector Machines (SVM) algorithm was trained using the 14 mentioned variables as input and ground truth value as the target variable. It was achieved using *Scipy* library [35] using default parameters. SVM is a supervised machine learning algorithm, that could be used both for classification and regression, though this time it's only used for classification. Support vector machine is done by constructing hyperplanes. The most important step is to choose a hyperplane in a way that the distance between the hyperplane and the nearest data point from either set would be as big as possible, giving a greater chance of new data being classified correctly [23].

Figure 19 shows a correlation matrix made from the table that was used to train the model.

It can be seen that apart from variables correlating with themselves, the biggest positive correlations are brightness with average brightness (0.93), lesion's distance to the optic nerve, and area in infrared images (0.88). Also, quite high correlations (0.47-0.5) are between these combinations: contour height and width, average brightness and label, simple brightness and



	brightness	avg_brightness	variance	angle	cont_height	cont_width	laterality	depth	cohort	diff_days	dist_to_onh	dist_to_vessel	ir_area	label
brightness	1.000000	0.932975	0.423600	0.005710	-0.238466	-0.059652	0.162710	0.216470	-0.081720	-0.212795	-0.227034	0.082996	-0.156115	0.468676
avg_brightness	0.932975	1.000000	0.466925	0.004376	-0.268705	-0.071091	0.179693	0.238519	-0.091903	-0.237735	-0.255852	0.089624	-0.176697	0.494034
variance	0.423600	0.466925	1.000000	-0.083583	-0.240900	-0.140938	0.197294	0.149025	-0.090997	-0.183981	-0.059557	0.027863	-0.021793	0.303206
angle	0.005710	0.004376	-0.083583	1.000000	0.057610	0.021426	-0.010368	-0.022918	0.017801	0.027783	0.025482	0.020468	0.008464	-0.051405
cont_height	-0.238466	-0.268705	-0.240900	0.057610	1.000000	0.495226	-0.101407	-0.510308	0.211702	0.380136	0.180710	-0.007399	-0.023496	-0.153812
cont_width	-0.059652	-0.071091	-0.140938	0.021426	0.495226	1.000000	-0.120823	-0.124211	-0.006544	-0.025760	-0.192869	-0.102037	-0.222128	-0.071020
laterality	0.162710	0.179693	0.197294	-0.010368	-0.101407	-0.120823	1.000000	0.009954	0.045753	-0.208808	0.021087	-0.043950	-0.010457	0.020759
depth	0.216470	0.238519	0.149025	-0.022918	-0.510308	-0.124211	0.009954	1.000000	-0.260334	-0.293458	-0.141623	0.172273	0.007316	0.289412
cohort	-0.081720	-0.091903	-0.090997	0.017801	0.211702	-0.006544	0.045753	-0.260334	1.000000	-0.219504	0.282176	-0.092043	0.191448	0.001950
diff_days	-0.212795	-0.237735	-0.183981	0.027783	0.380136	-0.025760	-0.208808	-0.293458	-0.219504	1.000000	0.186599	-0.129196	0.011260	-0.112041
dist_to_onh	-0.227034	-0.255852	-0.059557	0.025482	0.180710	-0.192869	0.021087	-0.141623	0.282176	0.186599	1.000000	0.408833	0.879344	-0.047164
dist_to_vessel	0.082996	0.089624	0.027863	0.020468	-0.007399	-0.102037	-0.043950	0.172273	-0.092043	-0.129196	0.408833	1.000000	0.353729	-0.003423
ir_area	-0.156115	-0.176697	-0.021793	0.008464	-0.023496	-0.222128	-0.010457	0.007316	0.191448	0.011260	0.879344	0.353729	1.000000	-0.010397
label	0.468676	0.494034	0.303206	-0.051405	-0.153812	-0.071020	0.020759	0.289412	0.001950	-0.112041	-0.047164	-0.003423	-0.010397	1.000000

Figure 19: Correlation matrix for variables used to train SVM. Blue means higher values while red is for lower values.

label, as well as average brightness and variance. While some are expected to correlate, it's important to note that brightness correlates with label which explains why the angle parameter could be useful. Apart from that, distance to the optic nerve and area in IR is unexpectedly quite well correlated. The biggest negative correlation is between contour height and depth (-0.51), then there is a bunch of combinations that have correlations from -0.29 to -0.2 that still could be interesting, depth and difference in days are the most significant one and unexpected.

Then when predicting new lesions at first they are segmented using before mentioned ResNet34 neural net model using the same parameters. After that, bounding boxes are made and for each pixel same parameters are calculated. Using this new table SVM predicts pixel values and they are used to improve initial segmentation.

It's important to note that in order to make the algorithm faster, all images (both when training and predicting) were resized to 224x224. This size was chosen because initial segmentation also predicts this size of images.

Using this algorithm lesions were segmented on the test set and Dice equal to **0.817** was achieved. Wilcoxon rank sum test showed that this method is still significantly worse than initial segmentation (p-value is 0.009).

Figure 20 visually shows how some of the predictions look on the test dataset. It looks like the late fusion algorithm is much more vulnerable when brightness changes, though some cases look quite accurate.

To improve segmentation, correlations shown in Figure 19 were analyzed, and for the first optimization attempt only variables that correlate the most with the label were taken. More specifically 6 variables have a correlation less than -0.1 or more than 0.1: *brightness*, *avg_ brightness*, *variance*, *cont_ height*, *depth*, *diff_ days*. SVM was trained only using these variables (their values weren't changed) and for new images, only these values were used to fix predictions. The Dice coefficient was calculated to be 0.82. Using the Wilcoxon rank-sum test p-value was calculated to compare optimization with initial segmentation (without data fusion) and it's equal to

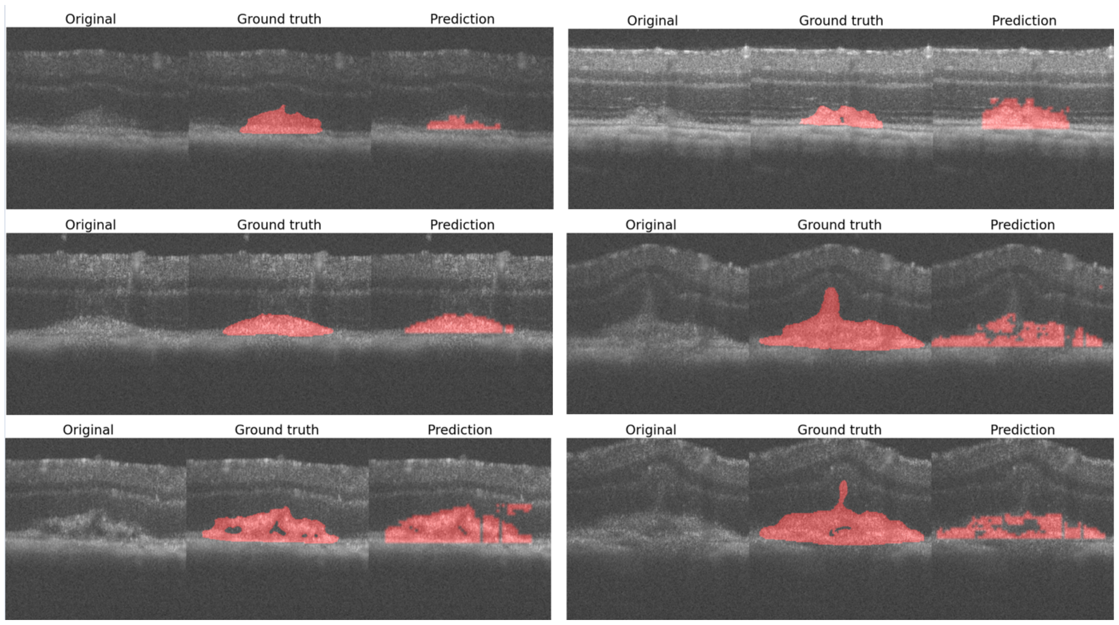


Figure 20: Image illustrating some of the predictions. Three images are presented: the original image, ground truth, and segmentation of the late fusion algorithm (after fixing pixels with SVM). It's important to note that images are cropped to better see lesions.

7.56e-16, thus, showing that still this method is significantly worse. However when compared to unoptimized late fusion Dice coefficient though not significantly better (p-value is 0.96).

Apparently, late fusion segmentation wasn't as accurate as the initial segmentation. Optimization helped to improve the algorithm but it's still significantly worse.

4.4 Testing methods on different task

Similarly to OCT b-scans, the same algorithms were used with FA (fluorescein angiography) images. To know whether data fusion actually works, it's essential to test more cases. CNV segmentation in FA images provides a bit different angle on the problem: regions of interest (lesions) look different, there are usually three lesions instead of either 1 or sometimes 2, and images are a bit smaller (768x768 pixels instead of 1000x1024).

Using annotated images initial neural net model was trained. The same parameters as with OCT b-scans were used (no data augmentation was done). It has dice equal to **0.904** which is a bit better compared to the previous set of images.

An early fusion algorithm was applied to FA images. Table 3 shows how coefficients were constructed to train neural net models for this set of images.

Coefficients are a bit different from the ones in Table 1 to have better use of the training dataset and not to overfit the models.

Applying an early fusion algorithm gave a Dice coefficient equal to **0.837**. Using the Wilcoxon rank-sum test p-value was calculated and it's equal to 4.4e-05, thus, with default

Table 3: Table explaining how coefficient to change colour channels were calculated - value before hyphen means preclinical parameter's value. Table works similarly to Table 1.
*Table only shows minimum and maximum values, all others in the range have coefficients assigned accordingly in a linear way.
**In this case green colour channel was modified using this coefficient instead of the red channel because each value is useful though not more important than another.

Preclinical parameter	**Colour coefficient**	**Model's dice**
Vendor	Janvier - 0.75, Envigo - 1.25	0.847
Cohort	1 - 0.75, 2 - 1.25, 3 - 1.25**	0.86
Diff. birth date-induction date	54 - 0.75, 69 - 1.25*	0.826
Lesion's distance to optic nerve	183 - 0.75, 630 - 1.25*	0.882
IR area	1432 - 0.75, 19901 - 1.25	0.863
Lesion's distance to blood vessel	0 - 0.75, 83 - 1.25*	0.852

Table 4: Dice coefficients for various combinations of clinical parameters for early data fusion-based image segmentation in FA images

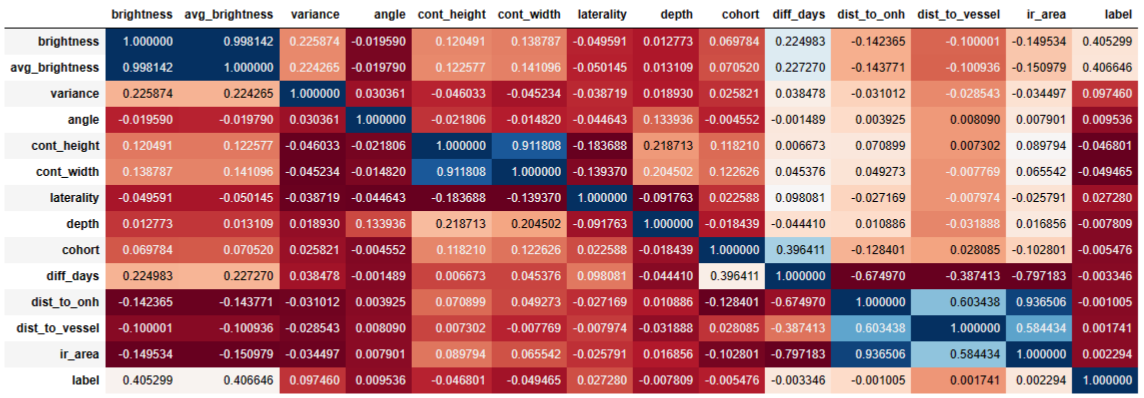
Combination of parameters	**Dice coefficient**	**P-value**
All except vendor	0.841	2.7e-5
All except cohort	0.783	6.9e-9
All except diff. birth date-induction date	0.821	3.7e-6
All except distance from lesion to optic nerve	0.789	1.7e-8
All except IR area	0.797	5.6e-8
All except distance from lesion to blood vessel	0.78	2.2e-20

parameters early fusion for FA image segmentation is significantly worse compared to the initial model.

As with previous images, there was an attempt to make the algorithm better by trying various combinations. Table 4 shows various combinations of preclinical parameters. It can be seen that the first combination where vendor, as a parameter was ignored, gave the best results with a Dice coefficient equal to **0.841**, though it's still significantly worse than the initial segmentation.

Furthermore, a late fusion algorithm was also applied for FA images. The same 14 variables and parameters (including resizing to 224x224) were used as in this case it's also important that the pixel would be brighter, as well as the center of the lesion is usually more intensive in colour, which could be useful for the angle parameter. Using such data SVM model was trained. Unfortunately, compared to correlations for OCT b-scans, these weren't as high as Figure 21 shows.

In the whole table, some high correlations can be observed: average and average brightness (0.99), distance to optic nerve and area in infrared image (0.94), contour height, and contour width (0.91). Unfortunately, when looking at correlations with target variables only two were higher than 0.1: brightness and average brightness (both 0.41).



	brightness	avg_brightness	variance	angle	cont_height	cont_width	laterality	depth	cohort	diff_days	dist_to_onh	dist_to_vessel	ir_area	label
brightness	1.000000	0.998142	0.225874	-0.019590	0.120491	0.138787	-0.049591	0.012773	0.069784	0.224983	-0.142365	-0.100001	-0.149534	0.405299
avg_brightness	0.998142	1.000000	0.224265	-0.019790	0.122577	0.141096	-0.050145	0.013109	0.070520	0.227270	-0.143771	-0.100936	-0.150979	0.406646
variance	0.225874	0.224265	1.000000	0.030361	-0.046033	-0.045234	-0.038719	0.018930	0.025821	0.038478	-0.031012	-0.028543	-0.034497	0.097460
angle	-0.019590	-0.019790	0.030361	1.000000	-0.021806	-0.014820	-0.044643	0.133936	-0.004552	-0.001489	0.003925	0.008090	0.007901	0.009536
cont_height	0.120491	0.122577	-0.046033	-0.021806	1.000000	0.911808	-0.183688	0.218713	0.118210	0.006673	0.070899	0.007302	0.089794	-0.046801
cont_width	0.138787	0.141096	-0.045234	-0.014820	0.911808	1.000000	-0.139370	0.204502	0.122626	0.045376	0.049273	-0.007769	0.065542	-0.049465
laterality	-0.049591	-0.050145	-0.038719	-0.044643	-0.183688	-0.139370	1.000000	-0.091763	0.022588	0.098081	-0.027169	-0.007974	-0.025791	0.027280
depth	0.012773	0.013109	0.018930	0.133936	0.218713	0.204502	-0.091763	1.000000	-0.018439	-0.044410	0.010886	-0.031888	0.016856	-0.007809
cohort	0.069784	0.070520	0.025821	-0.004552	0.118210	0.122626	0.022588	-0.018439	1.000000	0.396411	-0.128401	0.028085	-0.102801	-0.005476
diff_days	0.224983	0.227270	0.038478	-0.001489	0.006673	0.045376	0.098081	-0.044410	0.396411	1.000000	-0.674970	-0.387413	-0.797183	-0.003346
dist_to_onh	-0.142365	-0.143771	-0.031012	0.003925	0.070899	0.049273	-0.027169	0.010886	-0.128401	-0.674970	1.000000	0.603438	0.936506	-0.001005
dist_to_vessel	-0.100001	-0.100936	-0.028543	0.008090	0.007302	-0.007769	-0.007974	-0.031888	0.028085	-0.387413	0.603438	1.000000	0.584434	0.001741
ir_area	-0.149534	-0.150979	-0.034497	0.007901	0.089794	0.065542	-0.025791	0.016856	-0.102801	-0.797183	0.936506	0.584434	1.000000	0.002294
label	0.405299	0.406646	0.097460	0.009536	-0.046801	-0.049465	0.027280	-0.007809	-0.005476	-0.003346	-0.001005	0.001741	0.002294	1.000000

Figure 21: Correlation matrix for variables used to train SVM with FA images. As previously, blue means higher values while red means lower values.

When using all 14 variables to test how the algorithm performs, a Dice coefficient equal to **0.652** was observed. Even without the Wilcoxon rank-sum test (although for curiosity p-value is equal to 3.3e-18) it is obvious that image segmentation was much worse.

Using before mentioned best correlating variables (brightness and average brightness) an attempt was made to optimize the algorithm. Dice coefficient equal to **0.89** was calculated and with a p-value equal to 0.21. Differently from all other data fusion attempts, in this case, even though the model doesn't perform as well as initial segmentation, the difference isn't significant. Though it would still apply that optimization didn't help the late fusion algorithm to reach the performance that the initial segmentation model achieved. Another reason for the previous statement is that it only uses two variables that essentially mean the same and none of them are preclinical, which could raise the question of whether such a model would still count as data fusion.

Table 5: Final results summarized. The table shows the best observed Dice coefficients on test sets of both OCT b-scans and FA images using three tested algorithms: initial segmentation (no data fusion), early data fusion, and late data fusion

	OCT b-scans	FA images
Initial segmentation	**0.902**	**0.904**
Early data fusion	0.816	0.841
Late data fusion	0.82	0.89

5 Discussion

With two different datasets and two data fusion algorithms, it was tested whether image segmentation could be improved. Table 5 shows the best Dice coefficients achieved with each combination. It can be seen that for both datasets initial segmentation was best performing. Wilcoxon rank-sum test proved that differences in most results are significant (except initial segmentation compared to late data fusion for FA images). It's interesting to note, that the late fusion algorithm performed slightly better which could imply that either the late fusion algorithm is more suitable for this task or just early data fusion algorithm wasn't as effective. Though on the other hand, early data fusion relied more on preclinical data which was one of the requirements for this task.

Data fusion algorithms used by other scientists proved to be successful [9] [20], thus, it is important to discuss why in this case they didn't work. There might be a few reasons for that.

The first reason could be that data fusion adds more complexity. For example, instead of having just one segmentation model, the early fusion algorithm has up to a number of clinical parameters. While one of the models could fix other models' mistakes, optimization for both image sets showed that some (but not the same) variables are not needed. Additionally, image modifications are prone to errors. Tables 1 and 3 have quite a big variation of what value corresponds to what coefficient. This creates a dilemma as to whether it's better to make them the same or more related to input data. In most cases range for the coefficient by which the red channel's value is multiplied is in the range from 0.75 to 1.25 (1 meaning it will remain the same). Allowing bigger coefficients could make the image less recognizable while making changes minimal doesn't influence preclinical parameters.

Another related reason why algorithms don't perform well enough is that data doesn't correlate well with the target variable. This was shown in Figures 19 and 21. For OCT b-scans 6 of 14 variables had an absolute correlation bigger than 0.1 while for FA images there were only two variables that were very highly correlated with each other. Out of six preclinical parameters intended to be used, the only difference in days between birth date and induction date barely passed this limit and only for OCT b-scans. This could show why for FA images late fusion algorithm without optimization performed much worse.

Speaking about preclinical parameters, it is likely that this task in general isn't well suited

Table 6: Dice coefficients for separate treatments for each algorithm to test whether there is a difference between treatments, also p-values generated by Wilcoxon rank-sum test to check if differences between Dice coefficient are significant. The table on the left shows results for OCT b-scans, right table for FA images. "Afliber." means aflibercept, "segm." means segmentation.

OCT-bscans	Aflib.	Vehicle	P-value
Initial segm.	0.899	0.905	0.368
Early fusion	0.759	0.884	4.712e-6
Late fusion	0.838	0.801	0.345

FA images	Aflib.	Vehicle	P-value
Initial segm.	0.89	0.917	0.163
Early fusion	0.835	0.847	0.291
Late fusion	0.88	0.9	0.346

for data fusion. With clinical data, there are more parameters that could seem more meaningful like body mass index, blood pressure, and harmful habits. None of these are measured for mice. While logically variables like vendor or cohort could influence treatment, as well as the lesion's position in the eye, it might not be as important to treatment as parameters being measured for humans.

Furthermore, this raises the question of whether algorithms themselves are universal. In this thesis, preclinical ophthalmology is being analyzed. In examples mentioned before [9] [20] [25] [33] it's different biology field (for example, for the first two it's respectively lung diseases and cancer). The main focus of this thesis was on OCT b-scans. When building an algorithm for late fusion, variables were selected both so that it would be suggested by the original author and logical in these kinds of images. For FA images same variables were calculated. While it can be seen that some of the same logic applies (for example brighter pixels are more likely to belong to a lesion), based on correlation it looks like variables would need to be restructured. That's just between datasets of the same pathology but when images don't just look differently but also analyze different pathology, the same logic might no longer work.

Another question that is important when comparing algorithms is whether they preserve biomedical tendencies. Since segmentations in general are used to speed up the process of analyzing results and whether treatment works, it's important that the model wouldn't be biased. Table 6 shows Dice coefficients separated by treatments. Only aflibercept and vehicle were chosen as they are the most common treatments, also they biologically make the most sense as aflibercept was proven to have healing properties [36] while the vehicle has none and they are used for comparison. Apart from Dice coefficients, there are also p-values calculated using the Wilcoxon rank-sum test where it was compared whether model performance is different between treatments for the same algorithm for the same image set. Looking at the table it can be seen that for OCT b-scans, for early data fusion algorithm difference is statistically significant, which is unacceptable. While one such occasion isn't definitive proof, this shows that early data fusion algorithms shouldn't be trusted.

All in all, because different types of data were fused for the same task, it became more complex, and preclinical variables weren't good enough to make up and increase the model's performance. Looking at the fact that early fusion algorithms can have bias on treatment, it's not advised to use data fusion for this type of task. All this analysis doesn't prove or say that data fusion shouldn't be used in general but rather it doesn't work in all spheres. If preclinical data would be more informative and better correlating, possibly the best solution would be to create a data fusion algorithm from scratch, though that would require time and expertise. On the other hand, initial segmentation ignoring all preclinical parameters seems like a viable option, which is advised to use for ophthalmology image segmentation in the preclinical stage.

6 Conclusion

The goal this thesis was trying to accomplish is whether data fusion is a good way to make image segmentation in ophthalmology images more accurate. Relevant literature was analyzed and two data fusion methods were found. As well as data was prepared (both preclinical data and medical images). After doing all of the analysis few conclusions can be formulated:

1. Using just around half a thousand images semantic segmentation models were trained that showed quite good results (Dice coefficients 0.902 for OCT b-scans and 0.904 for FA images).

2. Early data fusion was proven to be not as good, even after its optimization (0.816 for OCT b-scans and 0.841 for FA images).

3. Late data fusion algorithm was also proven to be worse compared to initial semantic segmentation (Dice coefficients 0.82 for OCT b-scans and 0.652 for FA images).

4. Data fusion possibly didn't work due to a few reasons: the task became more complex, variables were weak and not correlating with the target variable, and algorithms could develop bias.

To sum all of it up, the main conclusion of this thesis is that at least with tested methods for ophthalmology in preclinical stages, data fusion shouldn't be used for image segmentation.

References

[1] Experimentica. https://experimentica.com/about-us/.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Rene Y Choi, James M Brown, Jayashree Kalpathy-Cramer, R V Paul Chan, Susan Ostmo, Michael F Chiang, and J Peter Campbell. Variability in plus disease identified using a deep learning-based retinopathy of prematurity severity scale. *Ophthalmol Retina*, 4(10), 2020.

[5] Ernest T. Dobi, Carmen A. Puliafito, and Maryanna Destro. A new model of experimental choroidal neovascularization in the rat. *Archives of Ophthalmology*, 107(2):264, 1989.

[6] Chollet Francois. Keras, 2015.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[8] HowardJeremy. fastai. https://github.com/fastai/fastai, 2018.

[9] SC. Huang, A. Pareek, and R. Zamanian. Multimodal fusion with deep neural networks for leveraging ct imaging and electronic health record: a case-study in pulmonary embolism detection. *Sci Rep*, 10, 2020.

[10] Shih-Cheng Huang, Anuj Pareek, Saeed Seyyedi, Imon Banerjee, and Matthew P. Lungren. Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines. *Digital Medicine*, 136, 2020.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[12] June Jacobs. Python-based method for computing dice coefficient in multi-class segmentation tasks. *CopyProgramming*, 2023.

[13] Nieraj Jain, Sina Farsiu, Aziz Khanifar, Srilaxmi Bearelly, Roland Smith, Joseph Izatt, and Cynthia Toth. Quantitative comparison of drusen segmented on sd-oct versus drusen delineated on color fundus photographs. *Investigative ophthalmology visual science*, 51:4875–83, 10 2010.

[14] Jonas Jarutis, Simas Bijeikis, Symantas Ragauskas, Kernius Mickevicius, Giedrius Kalesnykas, Marc Cerrada-Gimenez, Simon Kaja, and Nerija Kvietkauskiene. Validation of a novel artificial intelligence (ai) machine learning algorithm for quantification of vascular pathology in preclinical models. In *ARVO Annual Meeting Abstract.* iovs, 2021.

[15] Richard Socher Jia Deng, Wei Dong, Kai Li Li-Jia Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[17] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] Leo A. Kim, Koushik Tripathy, Andrea Tamine Hoyos Dumar, Diana V. Do, Talisa de Carlo, Neelakshi Bhagat, and Jennifer I Lim. Choroidal neovascularization: Oct angiography findings. *American academy of ophthalmology*, 2021.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[20] Zhou C Lu Q, Zhang H, Liang L, Zhang Q, Chen X, Xu X, Zhao G, Ma J, Gao Y, Peng Q, and Li S. A multimodal model fusing multiphase contrast-enhanced ct and clinical characteristics for predicting lymph node metastases of pancreatic cancer. *Phys Med Biol.*, 67(17), 2022.

[21] Samaya Madhavan, Sidra Ahmed, Vinay Rao, and Anto John. Compare deep learning frameworks. 2021.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] Derek A. Pisner and David M. Schnyer. Chapter 6 - support vector machine. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 101–121. Academic Press, 2020.

[24] Daniel Porter. What is fluorescein angiography? *American academy of ophthalmology*, 2021.

[25] Ezzedine Ben Braiek Rafika Harrabi. Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images. *EURASIP Journal on Image and Video Processing*, 11, 2012.

[26] Symantas Ragauskas, Eva Kielczewski, Joseph Vance, Simon Kaja, and Giedrius Kalesnykas. In vivo multimodal imaging and analysis of mouse laser-induced choroidal neovascularization model. *Journal of Visualized Experiments*, (131), 2016.

[27] Anran Ran and Carol Y. Cheung. Deep learning-based optical coherence tomography and optical coherence tomography angiography image analysis: An updated summary. *Asia-Pacific Journal of Ophthalmology*, 10(3):253–260, 2021.

[28] Saman Razavi. Deep learning, explained: Fundamentals, explainability, and bridgeability to process-based modelling. *Environmental Modelling Software*, 144, 2021.

[29] Z. Reitermanova. Data splitting. *WDS'10 Proceedings of Contributed Papers*, Part I:31–36, 2010.

[30] F. Renard, S. Guedria, and N.D Palma. Variability and reproducibility in deep learning for medical image segmentation. *Sci Rep 10*, 13724, 2020.

[31] Rosenblatt and Frank. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[32] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[33] P. Arbeláez J. Malik S. Gupta, R. Girshick. Learning rich features from rgb-d images for object detection and segmentation. *Lecture Notes in Computer Science*, pages 345—-360, 2014.

[34] Maciej Szkulmowski and Maciej Wojtkowski. Averaging techniques for oct imagingy. *Optics Express*, 21, Issue 8:9757–9773, 2013.

[35] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov,

Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[36] Maria Vähätupa, Symantas Ragauskas, Agne Ziniauskaite, Anna Knuuttila, Hanna-Kaisa Sihvo, Mikael Jääskeläinen, Simon Kaja, Giedrius Kalesnykas, and Marc Cerrada-Gimenez. Automated quantification of cnv volume using deep learning ai algorithm. In *ARVO Annual Meeting Abstract*. iovs, 2020.

[37] Jie Wang, Tristan T. Hormel, Liqin Gao, Pengxiao Zang, Yukun Guo, Xiaogang Wang, Steven T. Bailey, and Yali Jia. Automated diagnosis and segmentation of choroidal neovascularization in oct angiography using deep learning. *Biomed. Opt. Express*, 11(2):927–944, Feb 2020.

[38] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[39] Chris Wild. The wilcoxon rank-sum test. *Department of Statistics, University of Auckland*, 1997.

[40] Zahid Yaqoob, Jigang Wu, and Changhuei Yang. Spectral domain optical coherence tomography: a better oct imaging strategy. *Biotechniques*, 39(6S), 2018.

[41] Min Zhao, Irmela Mantel, Emmanuelle Gelize1, Xinxin Li, Xiaoyue Xie, Alejandro Arboleda, Marie Seminel, Rinath Levy-Boukris, Marilyn Dernigoghossian, Andrea Prunotto, Charlotte Andrieu-Soler, Carlo Rivolta, Jérémie Canonica, Marie-Christine Naud, Sebastian Lechner, Nicolette Farman, Irene Bravo-Osuna, Rocio Herrero-Vanrell, Frederic Jaisser, and Francine Behar-Cohen. Mineralocorticoid receptor antagonism limits experimental choroidal neovascularization and structural changes associated with neovascular age-related macular degeneration. *Nature communications*, 10:369, 2019.

7 Appendix

This section provides supplementary material that would be too cumbersome if used in the middle of the text.

7.1 Appendix A. Library versions

All code was done using Python version 3.10.9. Versions of most commonly used libraries:

- albumentations - 1.3.1
- fastai - 2.7.12
- fastcore - 1.5.29
- matplotlib - 3.7.0
- numpy - 1.23.5
- opencv-python - 4.7.0.72
- pandas - 1.5.3
- scikit-learn - 1.2.1
- torch - 2.0.0
- tqdm - 4.64.1
- scipy - 1.10.0

7.2 Appendix B. Code for training initial image segmentation

```
# input parameters
def get_image_files2(destination):
    # function makes a list of all images in the folder
    dest_files = os.listdir(str(destination))
    files = []
    for f in dest_files:
        if "_mask" not in str(f):
            files.append([Path(str(destination)+"/"+f))
    return files

def label_func(fn):
    # label function for loading segmentation model, it defines target variable
    return destination/f"{fn.stem}_mask{fn.suffix}"

destination = Path('./oct_bscans/')
images = [f for f in get_image_files2(destination)]
dls = SegmentationDataLoaders.from_label_func(
    destination,
    bs=32,
    seed=42,
    valid_pct=0.1,
    fnames=images,
    label_func=label_func,
    item_tfms=[Resize(224)],
)

# training
learn = unet_learner(dls, resnet34, n_out=2, metrics=Dice())
learn.fine_tune(20)

# exporting model
model_save_path = ...
learn.export(os.path.abspath(model_save_path))
```

7.3 Appendix C. Code for early fusion algorithm (for OCT b-scans)

```
def get_px_m(imgs, x0, y0, window=3):
    # gets modified value
    height, width = imgs[0].shape
    step = int(np.ceil(window/2))-1
    xl = max(0, x0-step)
    xr = min(width, x0+step)
    yl = max(0, y0-step)
    yr = min(height, y0+step)
    ms = []
    for x in range(xl, xr):
        for y in range(yl, yr):
            vals = [img[y,x] for img in imgs]
            m = np.mean(vals)
            ms.append(m)
    final_m = np.mean(ms)
    return final_m

def get_dice(ann, pred):
    # calculates Dice coefficient
    epsilon = 1e-10
    if pred.shape != ann.shape:
        pred = cv2.resize(pred, (ann.shape[1], ann.shape[0]))
    mask_and =np.logical_and(ann, pred)
    dice1 = 2*np.sum(mask_and)/(np.sum(ann)+np.sum(pred)+epsilon)
    if str(dice1) == "nan":
        dice1 = 1
    # adding dice for background
    ann[ann == 1] = 2
    ann[ann == 0] = 1
    ann[ann == 2] = 0
    pred[pred == 1] = 2
    pred[pred == 0] = 1
    pred[pred == 2] = 0
    mask_and =np.logical_and(ann, pred)
    dice2 = 2*np.sum(mask_and)/(np.sum(ann)+np.sum(pred)+epsilon)
    if str(dice2) == "nan":
        dice2 = 1
    dice = np.mean([dice1, dice2])
    return dice

def label_func(fn): ... # same as in Appendix B

vendor_model = load_learner(model_path1)
cohort_model = load_learner(model_path2)
difference_birth_induction_model = load_learner(model_path3)
dist_to_onh_model = load_learner(model_path4)
dist_to_vessel_model = load_learner(model_path5)
ir_area_model = load_learner(model_path6)

models = {"vendor": vendor_model, "cohort": cohort_model,
"diff_days": difference_birth_induction_model,
"dist_to_onh": dist_to_onh_model, "dist_to_vessel": dist_to_vessel_model,
"ir_area": ir_area_model}

folder_path = ...
files = [x for x in os.listdir(folder_path) if "mask" not in str(x)]
final_table = pd.read_csv(path_of_table_with_preclinical_data)

metrics = {"filename": [], "dice": []}
new_pred = None
for f in tqdm(files):
    img = cv2.imread(folder_path, 3)
    # following few lines extracts animal ID from filename
    animal = f[4:].replace("ion_", "ion").replace("ion-", "ion").split("_")[0]
    animal = animal.replace("In", "In_").replace("ion", "ion_")
    animal = animal.split("-")[0]+"-"+animal.split("-")[-1].zfill(3)
    animal_info = final_table.loc[final_table["animal_id"] == animal]
    preds = []
    for model_key in models.keys():
        value = list(animal_info[model_key])[0]
        if str(value) not in ["", "nan"]:
            if model_key == "vendor":
                if value == "janvier":
                    coef = 0.75
                else:
                    coef = 1.25
                img[:,:,0] = img[:,:,0]*coef
            if model_key =="cohort":
                if value == 1.0:
                    coef = 0.75
                    img[:,:,0] = img[:,:,0]*coef
                elif value == 2.0:
                    coef = 1.25
                    img[:,:,0] = img[:,:,0]*coef
                else:
                    coef = 1.25
                    img[:,0,:] = img[:,0,:]*coef
            if model_key =="diff_days":
                diff_coef = abs(value-71)/16
                if value >= 71:
                    coef = 1+diff_coef
                else:
                    coef = 1-diff_coef
                img[:,:,0] = img[:,:,0]*coef
            if model_key =="dist_to_onh":
                coef = ((value-185)/(285-185))*0.5+0.75
                img[:,:,0] = img[:,:,0]*coef
            if model_key =="dist_to_vessel":
                coef = ((value-0)/(32-0))*0.5+0.75
                img[:,:,0] = img[:,:,0]*coef
            if model_key =="ir_area":
                coef = ((value-1500)/(3500-1500))*0.5+0.75
                img[:,:,0] = img[:,:,0]*coef
        model = models[model_key]
        with model.no_bar():
            pred, _, probs = model.predict(img)
        pred = cv2.resize(np.uint8(np.array(pred)), img.shape[:2][::-1])
        preds.append(pred)
    height, width = img.shape[:2]
    new_pred = np.zeros(img.shape[:2][::-1])
    for x0 in range(0, width):
        for y0 in range(0, height):
            m = get_px_m(preds, x0, y0, window=3)
            vals = [pred[y0, x0] for pred in preds]
            m = np.mean(vals)
            if m >=0.5:
                m = 1
            else:
                m = 0
            new_pred[x0, y0] = m
    new_pred = new_pred.T

    ann = cv2.imread("./data/oct_annotations/"+f[:-4]+"_mask.png", 0)
    ann[ann>0] = 1
    dice = get_dice(ann, new_pred)
    metrics["filename"].append(f)
    metrics["dice"].append(dice)
```

7.4 Appendix D. Code for late fusion algorithm

```
def get_avg_and_var(img, x0, y0):
    # calculates mean and variance in 3x3 window around pixel
    vals = []
    for x in [x0-1, x0, x0+1]:
        for y in [y0-1, y0, y0+1]:
            try:
                vals.append(img[x,y])
            except:
                pass
    return np.mean(vals), np.var(vals)

def get_angle(img, x0, y0):
    # calculates angle parameter
    if x0 in [0, img.shape[0]] or y0 in [0, img.shape[0]]:
        return 0
    matrixes = [
        np.matrix([[ 1, 1, 1],
                   [ 0, 0, 0],
                   [-1,-1,-1]]),
        np.matrix([[ 0, 1, 1],
                   [-1, 0, 1],
                   [-1,-1, 0]]),
        np.matrix([[-1, 0, 1],
                   [-1, 0, 1],
                   [-1, 0, 1]]),
        np.matrix([[-1,-1, 0],
                   [-1, 0, 1],
                   [ 0, 1, 1]]),
        np.matrix([[-1,-1,-1],
                   [ 0, 0, 0],
                   [ 1, 1, 1]]),
        np.matrix([[ 0,-1,-1],
                   [ 1, 0,-1],
                   [ 1, 1, 0]]),
        np.matrix([[ 1, 0,-1],
                   [ 1, 0,-1],
                  [ 1, 0,-1]]),
        np.matrix([[ 1, 1, 0],
                  [ 1, 0,-1],
                  [ 0,-1,-1]]),
    ]
    angles = [0,45,90,135,180,225,270,315]
    this_matrix = np.matrix([[img[x0-1,y0-1],img[x0  ,y0-1],img[x0+1,y0-1]],
                           [img[x0-1,y0  ],img[x0  ,y0  ],img[x0+1,y0]],
                           [img[x0-1,y0+1],img[x0  ,y0+1],img[x0+1,y0+1]]])
    best_angle = -9999
    best_angle_ind = 0
    for m in range(len(matrixes)):
        mat = matrixes[m]
        mult = np.sum(np.multiply(this_matrix, mat))
        if mult > best_angle:
            best_angle = mult
            best_angle_ind = m
    return angles[best_angle_ind]


def get_dice(ann, pred): ... # same as in Appendix C

def label_func(fn): ... # same as in Appendix B

preclinical_df = pd.read_csv(path_of_table_with_preclinical_data)

learn = load_learner(initial_model_path, cpu=False)

files = [f for f in os.listdir(input_path) if "mask" not in f]

points_df = {"id": [], "filename": [], "brightness": [], "avg_brightness": [],
    "variance": [], "angle": [], "cont_height": [], "cont_width": [],
    "laterality": [], "depth": [], "vendor": [], "cohort": [], "diff_days": [],
    "dist_to_onh": [], "dist_to_vessel": [], "ir_area": [], "label": []}

pxi = 0
for f in tqdm(files):
    # following few lines extracts animal ID from filename
```

```
animal = f[4:].replace("ion_", "ion").replace("ion-", "ion").split("_")[0]
animal = animal.replace("In", "In_").replace("ion", "ion_")
animal = animal.split("-")[0]+"-"+animal.split("-")[-1].zfill(3)
animal_info = preclinical_df.loc[preclinical_df["animal_id"] == animal]
vendor = list(animal_info["vendor"])[0]
cohort = list(animal_info["cohort"])[0]
diff_days = list(animal_info["diff_days"])[0]
dist_to_onh = list(animal_info["dist_to_onh"])[0]
dist_to_vessel = list(animal_info["dist_to_vessel"])[0]
ir_area = list(animal_info["ir_area"])[0]
ann = cv2.imread(input_path+f[:-4]+"_mask.png", 0)
img = cv2.imread(input_path+f, 3)
img = img[:,:,0]
img = cv2.resize(img, (224,224))
ann = cv2.resize(ann, (224,224))
contours, _ = cv2.findContours(ann, cv2.RETR_CCOMP,cv2.CHAIN_APPROX_SIMPLE)
for cnt in contours:
    xs = []
    ys = []
    for pt in cnt:
        xs.append(pt[0][0])
        ys.append(pt[0][1])
    for x0 in range(np.min(xs), np.max(xs)):
        for y0 in range(np.min(ys), np.max(ys)):
            x = y0
            y = x0
            avg, var = get_avg_and_var(img, x, y)
            angle = get_angle(img, x, y)
            points_df["id"].append(pxi)
            points_df["filename"].append(f)
            points_df["brightness"].append(img[x,y])
            points_df["avg_brightness"].append(avg)
            points_df["variance"].append(var)
            points_df["angle"].append(angle)
            points_df["cont_height"].append(abs(np.max(ys)-np.min(ys)))
            points_df["cont_width"].append(abs(np.max(xs)-np.min(xs)))
            points_df["laterality"].append(y/img.shape[1])
            points_df["depth"].append(x/img.shape[0])
```

```
                points_df["vendor"].append(vendor)
                points_df["cohort"].append(cohort)
                points_df["diff_days"].append(diff_days)
                points_df["dist_to_onh"].append(dist_to_onh)
                points_df["dist_to_vessel"].append(dist_to_vessel)
                points_df["ir_area"].append(ir_area)
                points_df["label"].append(ann[x,y])
                pxi += 1
points_df2 = pd.DataFrame(points_df)

X = points_df2.drop(columns=["id", "filename", "label"])
y = points_df2["label"]
clf = svm.SVC()
clf.fit(X, y)

# prediction
valid_df = {"id": [], "filename": [], "brightness": [], "avg_brightness": [],
    "variance": [], "angle": [], "cont_height": [], "cont_width": [],
    "laterality": [], "depth": [], "vendor": [], "cohort": [], "diff_days": [],
    "dist_to_onh": [], "dist_to_vessel": [], "ir_area": [], "label": []}
pxi = 0
valid_files = [x for x in os.listdir(valid_folder_path) if "mask" not in str(x)]
for f in tqdm(valid_files):
    ...
    # code is the same like the one under "for f in tqdm(files):"
    # except instead of points_df, data is collected in valid_df
valid_df2 = pd.DataFrame(valid_df)
X_valid = valid_df2.drop(columns=["id", "filename", "label"])
pred_rez = clf.predict(X_valid)
```