UPPSALA
UNIVERSITET

UPTEC X 20017
Examensarbete 30 hp
Juni 2020

Deep Learning Models for Profiling of Kinase Inhibitors

Linnea Eriksson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Deep Learning Models for Profiling of Kinase Inhibitors

Linnea Eriksson

With the advent of fluorescence microscopy and image analysis, quantitative information from images can be extracted and changes in cell morphology can be studied. Microscopy-based morphological profiling assays with multiplexed fluorescent dyes, like Cell Painting, can be used for this purpose. It has been shown that morphological profiles can be used to train AI models to classify images into different biological mechanisms. Hence, the goal of this project was to study the possibilities for Deep Learning models and Convolutional Neural Networks to distinguish between different classes of kinase inhibitors based on their morphological profiles. Three different Convolutional Neural Network architectures were used: ResNet50, MobileNetV2, and VGG16. They were trained with two different inputs and two different optimisers: Adam and SGD. Also, a comparison between the performances with and without Transfer Learning through ImageNet weights was executed. The results indicate that MobileNetV2 with Adam as an optimiser performed the best, with a micro average of 0.93 and higher ROC areas compared to the other models. The study also highlighted the importance of utilizing Transfer Learning.

Handledare: Maris Lapins
Ämnesgranskare: Filip Malmberg
Examinator: Pascal Milesi
ISSN: 1401-2138, UPTEC X 20017

Popular Science

Trots stora framsteg inom cancerforskningen kvarstår många frågetecken, och vad som orsakar vissa cancerformer är fortfarande osäkert. På samma sätt vet forskarna inte hur dessa cancertyper bäst behandlas. En vanlig cancerorsak är rubbad kommunikation mellan celler, vilket kan göra att de berörda cellerna förökar sig okontrollerat. Kinasinhibitorer är biologiska molekyler som konstaterats kunna rubba kommunikationen. Det här projektet har studerat om algoritmer kan tala om vilka kinasinhibitorer den ser på bilder med celler som behandlats med olika kinasinhibitorer. Tack vare ny teknik är det idag möjligt att studera sådana bilder tagna i speciella mikroskop med hjälp av algoritmer. Detta möjliggör datorstyrda studier av skillnader i utseende mellan celler, skillnader som eventuellt beror på att en kinasinhibator påverkat cellen och som kan vara svåra att se för det mänskliga ögat.

Huvudfokuset för projektet har varit att utveckla och utvärdera algoritmer som identifierar kinasinhibitorer i bilder av celler. Det har även studerats om några kinasinhibitorer är lättare eller svårare för algoritmen att se. För att utveckla sådana algoritmer har maskininlärning använts, och specifikt ett område inom maskininlärning som kallas för djupinlärning med hjälp av artificiella neurala nätverk. Neurala nätverk är datorns sätt att efterlikna den mänskliga hjärnans neuroner, vilka i sin tur är nervceller som med otroligt komplex kommunikation talar med varandra vid rörelsestyrning och sinnesintryck. Djupa artificiella neurala nätverk består av flera lager av olika lärdomar som algoritmen drar, som sparas och förfinas för varje lager fram tills det sista lagret där algoritmen berättar vad den tror att den har sett. Det imponerande med dessa algoritmer är att de kan lära sig och förbättra sin prestation när de ser mer data. Det kan liknas vid att lära ett barn om vad exempelvis ett bord är. Istället för att förklara att ett bord ofta består av en skiva med fyra ben, så pekar du ut ett flertal olika bord för barnet. På så sätt lär sig barnet att känna igen ett bord utan att du behöver förklara vilka egenskaper som karakteriserar ett bord. En variant av artificiella neurala nätverk som användes i detta projekt var faltande neurala nätverk. Skillnaden mot vanligare neurala nätverk är att denna typ är specialiserad på att hitta mönster och viktiga egenskaper i just bilder. I detta fall fick algoritmen ta del av cirka 20% av alla bilder på celler som studien hade tillgång till. På några av bilderna var det celler som behandlats med kinasinhibitorer och på några var det obehandlade celler. Algoritmen fick själv leta efter viktiga egenskaper i de olika bilderna. Därefter testades algoritmen på annan del av samma dataset där den fick peka ut vilka kinasinhibitorer som fanns bland bilderna den fått se.

Förutom cancerforskning kan tekniker som dessa appliceras på en rad olika områden, allt från prediktioner av spridningen av COVID-19 till hemsidor med automatiska chattar som lärt sig att svara på vanliga frågor. Möjligheterna för användning av maskininlärning inom biologi och medicin är oändliga, men kommer maskininlärning kunna ersätta mänsklig arbetskraft? Det är en otroligt omdebatterad fråga. I dagsläget kan i alla fall maskininlärning ses som ett extra par ögon som kan assistera människor i att prestera bättre inom vissa analytiska områden. I grund och botten handlar maskininlärning om att dra statistiska lärdomar, vilket gjort metoden lämplig för just denna studie.

Deep Learning Models for Profiling of Kinase Inhibitors

Abbreviations

ADAM	adaptive moment estimation
AI	artificial intelligence
ANN	artificial neural network
AUC	area under the curve
CNN	convolutional neural network
NN	neural network
ROC	receiver operating characteristic curve
RMSprop	root mean square propagation
SGD	stochastic gradient descent

1 Introduction

Cell signalling is a vital process that controls cell division, cell migration, and cell death. Phosphorylation is one event that control cell signalling, and is a way for proteins to transmit chemical signals to each other (Nature Research 2019). During phosphorylation, there is a transfer of a phosphate group onto a particular amino acid in a protein. The phosphate comes from adenosine triphosphate (ATP) molecules, which is a source of chemical energy consisting of one adenosine and three phosphate groups called alfa, beta, and gamma (Lodish *et al.* 2008). The gamma phosphate can be donated to some amino acids, and although there are 20 different amino acids, only tyrosine, serine, and threonine can be phosphorylated (García *et al.* 2006). The enzymes that catalyse the transfer of the gamma phosphate groups are called kinases. There are 518 kinases that are divided into four subclasses; tyrosine kinases, dual-specificity kinases, serine-threonine kinases, and pseudo kinases (García *et al.* 2006). When a protein gets phosphorylated, it attracts proteins that, in turn, can attract other proteins. This can cause a chain reaction where signals are passed on across a cell.

Phosphorylation leads to an increase or decrease in the activity of the kinase. Faulty, hyper activated kinases can transmit to many phosphorylation signals which can lead to diseases such as cancer (Campbell *et al.* 2014a). All classes of proteins have members that are regulated by kinases or phosphatases, which indirectly regulate a variety of cellular pathways and reactions. Abnormal kinases that function in the absence of signalling molecules or while being inhibited by kinase inhibitors are associated with many kinds of cancer (Campbell *et al.* 2014b).

1.1 Purpose

Kinase inhibitors play an important role in many biological processes (Blume-Jensen & Hunter 2001, Ramón-Maiques *et al.* 2002, Karaman *et al.* 2008), and previous studies provide a foundation for further exploring of the toxicity and biology of kinase inhibitors. Knowledge about kinase inhibitors could potentially be implemented in drug development and further research about cellular signalling and cancer (Davis *et al.* 2011, Campbell *et al.* 2014a).

Recent advances in fluorescence microscopy and image analysis open up for extracting quantitative information from images to study changes in cell morphology induced by drugs, the environment, or chemical compounds like kinase inhibitors. One methodology used in the research group at the Department of Pharmaceutical Biosciences is Cell Painting (Bray *et al.* 2016), a microscopy-based morphological profiling assay with multiplexed fluorescent dyes. It has been shown that such morphological profiles can be used to train AI models to classify images into different biological mechanisms (classes) (Kensert *et al.* 2019). Deep learning and machine learning have been used to profile kinase inhibitors in multiple papers (Scheeder *et al.* 2018, Zhavoronkov *et al.* 2019, Moen *et al.* 2019), and the goal of this project was to study the possibilities for Deep Learning models and Convolutional Neural Networks to distinguish between different classes of kinase inhibitors based on their morphological profiles. This project provided an opportunity to evaluate these methods and find improvements. Hence, the performance of the prediction models, as well as factors that might affect the result, was analysed as well.

2 Background

2.1 Artificial Intelligence, Machine Learning and Deep Learning

Artificial intelligence (AI) are computer programs that can perceive the environment, reason, act and adapt to maximize the chance of accomplishing a certain goal (Bini 2018). Implementations of AI in healthcare opens up new doors of handling and optimising very complex sets of data in complex systems such as accurate identification of drug toxicity without animal testing (Aliper *et al.* 2016), and cancer detection (Fakoor *et al.* 2013, Wang *et al.* 2016, Cruz-Roa *et al.* 2017). Machine learning can be considered a subset of AI and consists of slightly less sophisticated algorithms that learn and improve their performance as they are exposed to more data over time. In conventional machine learning, representations are manually designed by the use of feature engineering. Machine Learning approaches can be broadly classified as supervised or unsupervised. Supervised learning aims to maximize the performance of the algorithm on annotated datasets, and is often a very successful alternative. Unsupervised learning reconstructs original data after it has been compressed into a low-dimensional space. (Moen *et al.* 2019)

Deep learning is a subset of machine learning and AI, which exploits deep multi-layered artificial neural networks (ANNs) that learn from a vast amount of data. The algorithm in a neural network learns effective representations of data consisting of multiple levels of abstraction and is a form of a statistical model. A neural network (NN) is called a deep NN or deep learning if there is more than 1 layer in the network. These deep networks were developed to perform well in complex games such as Go, where the number of possible permutations is more than there are atoms in the known universe. (Bini 2018) A visual representation of the relationship between AI, Machine Learning and Deep Learning can be seen in figure 1.

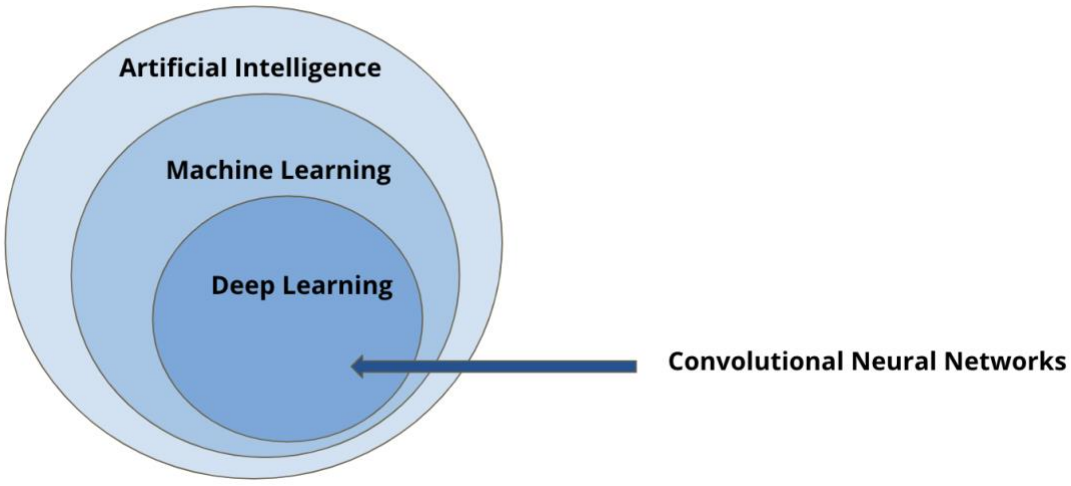


Figure 1. A visual representation of the relationships between Artificial Intelligence, Machine Learning, Deep Learning and Convolutional Neural Networks.

2.2 Biological applications of Deep Learning

Deep learning can be applied to a number of different applications. Image classification, image segmentation, object tracking, and augmented microscopy are some areas within computer-assisted image analysis where deep learning can be applied. In image classification, meaningful labels are assigned to an image. One famous example of this is the classification of cats and dogs in images. Due to a lack of annotated training data for deep learning on biological applications, transfer learning can be used and have been proven to function well on biological data (Zhang *et al.* 2016, Pawlowski *et al.* 2016). Transfer learning trains a deep model on a large dataset in order to learn general image features. By using transfer learning, a robust model can be created even though a limited amount of data is used. It is then applied to a smaller dataset where it is adjusted to perform a specific task. In this case, the large dataset could be the ImageNet dataset (Krizhevsky *et al.* 2012), and the smaller dataset could consist of annotated biological data (Zhang *et al.* 2016).

Deep learning is also a way of accounting for changes in cell morphology in image classification where it usually is harder to catch the changes. Hence, several cell morphology studies and approaches are using deep learning models (Kandaswamy *et al.* 2016, Pawlowski *et al.* 2016, Sommer *et al.* 2017). These models can also be used in other biological applications like cell cycle predictions, changes in cell state (Simm *et al.* 2018), spatial patterns in fluorescence images, finding the locations of proteins in yeast (Kraus *et al.* 2016, Kraus *et al.* 2017, Pärnamaa & Parts 2017) and finding the locations of proteins in humans (Sullivan *et al.* 2018). It has also been used in combination with microfluidics to perform image activated cell sorting (Nitta *et al.* 2018). Hence, deep learning is an appropriate method of choice when profiling kinase inhibitors in this project.

2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are supervised feature learning techniques and are a relatively new breakthrough in the area of image processing and computer vision. The idea behind it is that it automatically discovers new and needed features and patterns for image classification with fewer connections and parameters compared to standard feed-forward neural networks (Krizhevsky *et al.* 2012). The patterns can be the detection of edges in an image, and these types of simple filters often occur early on in the neural network. In later layers, the patterns get more complex and can detect features like eyes, hair, and feathers. Towards the end, the filters can detect full objects like cells, humans, cats, and dogs as an example (Geirhos *et al.* 2019). Some famous and widely used CNN architectures are LeNet-5, ResNet, AlexNet, VGG16, and the GoogLeNet, and they are often implemented in the popular deep learning backend framework TensorFlow in combination with the frontend library Keras (Nandy & Biswas 2018).

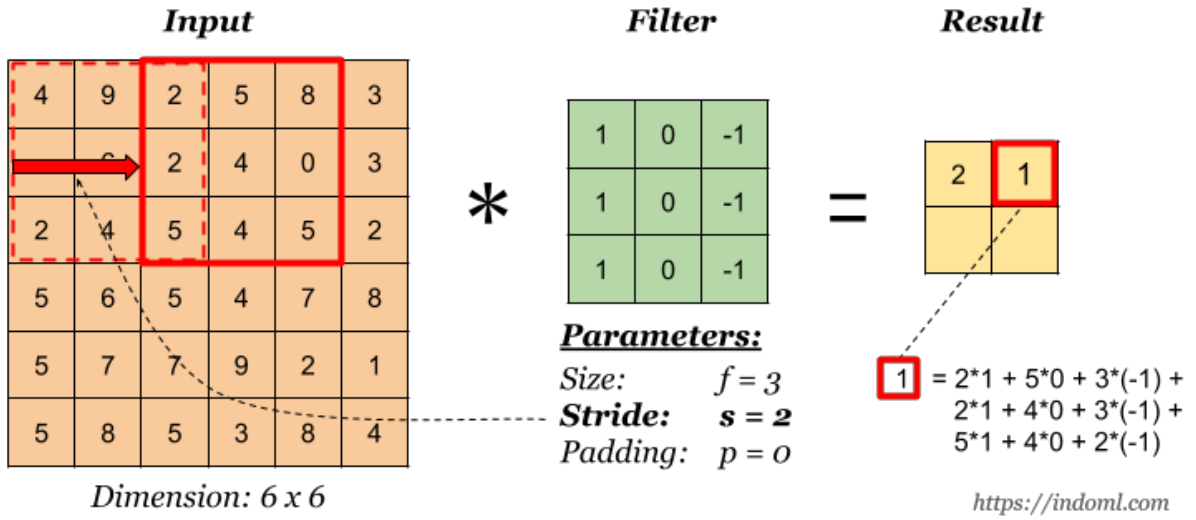


Figure 2. A visual representation of a filter sliding over an input image, the performed calculations and the result (IndoML 2018).

A convolutional layer lets a filter, also called a kernel, go over the input and performs element-wise multiplication and addition that ends up in the next cell in the result, also called the feature maps, as seen in Figure 2 (Voulodimos *et al.* 2018). Commonly used hyper parameters are the number of convolutional filters, stride, and padding (Yamashita *et al.* 2018). The stride regulates step size of the filter, i.e. the number of cells that the filter steps each time. This whole process of the filter sliding across the input is called convolving. The values in the convolutional filter correspond to the parameters, i.e. the weights in the network. The padding parameter controls the information at the borders of the input image (Yamashita *et al.* 2018).

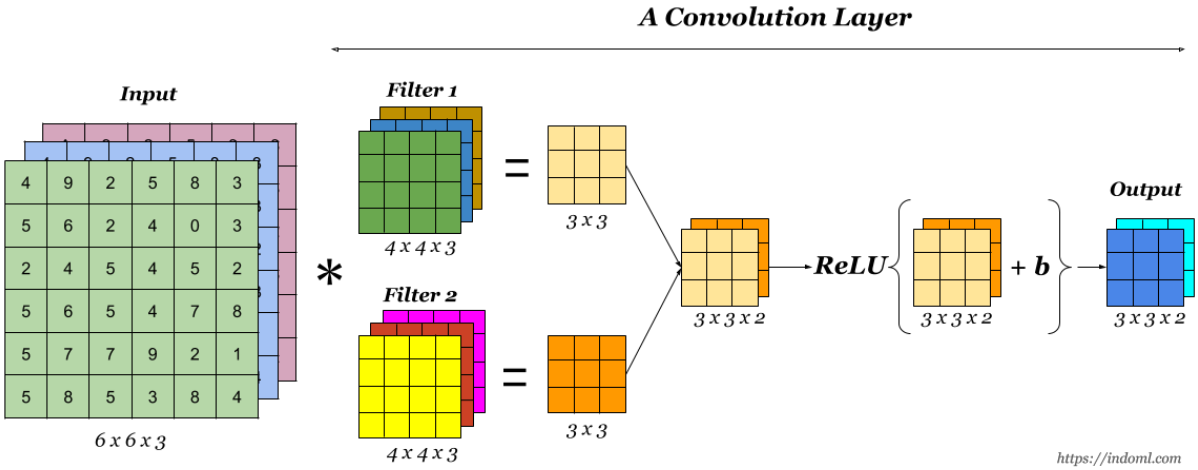


Figure 3: A visual representation of a convolutional layer (IndoML 2018).

The feature maps are generated by a series of convolution and pooling with activation functions between the layers. In Figure 3, the Rectified Linear (ReLU) activation unit is used which returns the values above zero or simply returns zero if the values are lower than or equal to zero (Yamashita *et al.* 2018). ReLU looks and acts like a linear function, but is actually a non-linear function that allows learning about complex, non-linear relationships in the data (Krizhevsky *et al.* 2012). There are other well used activation functions as well, like the Sigmoid or the Hyperbolic Tangent (Yamashita *et al.* 2018). By pooling, the algorithm

looks at the values of the neurons of the feature maps. Different types of pooling like max, min and average pooling can be applied. Max pooling selects the most activated neurons of the feature map, and discards neutral regions where no features were detected. The purpose of pooling layers is to reduce the size of the feature maps, speed up the calculations, and make some of the detected features more robust (Yamashita *et al.* 2018). After convolution and pooling, the results are flattened into fully connected layers. These fully connected layers are also called the hidden layers in the Neural Network, and that is the part of the CNN that works as the classifier. Also, the addition of a bias term can be seen in Figure 3. This bias term should be a vector with a length similar to the number of used filters.

CNNs also learn filter coefficients from the data, and uses hierarchical feature extraction. This is done by only using the raw pixel intensity data (LeCun *et al.* 2015). One of the big advantages of this method is that it does not require image segmentation before using the technique since the CNN framework already consists of segmentation and classification (Kraus *et al.* 2016). Another advantage of using CNNs is that, by applying convolving filters on the input, the number of parameters are low even if the network is deep. Different parameters that are often experimented with to improve the models are different learning rates that controls how quickly the model is fitted to the data, the number of epochs and the batch size described in 2.7.

2.4 Training, validation and test

When using Machine Learning, the dataset is split into three parts: Training, test and validation. The training set is what the model is trained on with different optimisation methods. The next step is to let the trained model predict the classes in the validation set, which then provides an unbiased evaluation of how the model is fitted to the training set. The last step is to use the test set, which is used to evaluate how the final model is fitted to the training set in an unbiased way as well as giving an approximate measure of what the performance of the model will be when being deployed.

2.5 ImageNet, pre-trained convolutional neural networks and transfer learning

Pre-trained convolutional neural networks are networks that previously have been trained on huge amounts of data. The learned features from classification of those images are useful for a new set of unseen data as well. In this project, pre-trained convolutional neural networks that had been trained on ImageNet (Deng *et al.* 2009, Stanford Vision Lab *et al.*) were used. ImageNet is a database that contains more than 14 million images that belongs to more than 20 000 classes. When you use one of the following pre-trained networks, you can specify that you want to use the ImageNet weights. The weights are large files that are automatically downloaded when the user specifies that these weights should be used, but they are only downloaded once and then stored in the Keras cache folder.

The following pre-trained convolutional neural network architectures were used in this project: MobileNetV2, ResNet50 and VGG16. MobileNetV2 (Sandler *et al.* 2018) is an architecture that is tuned to the CPUs in a mobile phone. This is a simple architecture that is especially suitable for mobile applications, but it still provides all the standard operations performed in neural networks. It uses 3 538 984 parameters and has a depth of 88 layers. ResNet50 (He *et al.* 2015) uses 25 636 712 parameters with 50 layers. In 2015, the

ResNet152 won the ImageNet competition, which has the same structure as ResNet50 but with 152 layers instead. VGG16 (Simonyan & Zisserman 2015) is the most computationally heavy out of these three architectures. It uses 138 357 544 parameters in 23 layers. The architecture was ranked among the top two best performing for localisation and classification in the ImageNet Challenge (ILSVRC) 2014.

These architectures utilize transfer learning, which is when a pre-trained classifier is trained for a new task but it uses the previously learned features for classifying the new task. In this case, the transfer learning happens thanks to ImageNet weights.

2.6 Optimisers

To minimize the error of a machine learning model, an optimiser is used. Two optimisers were used in this project, Adaptive Moment Estimation (Adam) (Kingma & Ba 2017) and Stochastic Gradient Descent (SGD) (Robbins & Monro 1951). SGD is one version of Gradient Descent. Gradient Descent can update the parameters of a model, observe how a function would be affected by a change, choose directions that lowers the error rate and iterates until the function converges to a minimum. SGD only computes a random selection of data or a small subset of data, and when the learning rate is small it provides the same performance as regular Gradient Descent. However, it uses a scalar learning rate on all parameters. Adam is used for gradient-based optimisation. It combines advantages of two SGD extensions, Root Mean Square Propagation (RMSprop) and Adaptive Gradient Algorithm. In contrast to SGD, Adam uses one vector of learning rates per parameter, which are adapted during the learning process.

When the efficacy of an optimiser is determined, the main factors that are taken into consideration are the speed of convergence and the generalization. This corresponds to how fast a global optimum in gradient descent is reached, and how the model performs on new unseen data. Adam and SGD are said to be able to cover one of these factors, but not both of them (Keskar & Socher 2017). Adam seems to perform better in the early stages of training, and SGD seems to perform better in the later stages of training. Hence, both of these optimisers were used to compare their performance on this data. However, RMSprop was also used in the beginning before this decision was made.

2.7 Epochs, Batch size and Batch normalization

Epochs defines the number of times that the algorithm goes through the entire training set. After one epoch, each sample in the training set has updated the weights. One epoch is the result of one or more batches. The batch size means the number of processed samples that is required before the model is updated, and it must be more than or equal to one, and less than or equal to the number of samples in the training dataset. So, if a dataset consists of 500 samples and the batch size is 10, the data will be divided into 50 batches containing 10 samples each. When the model has gone through each of the samples in a batch, the model will update the weights, which means that the weights will be updated 50 times in every epoch. The best way to configure this parameter, as well as the number of epochs, is by trying different values and see what works best. By doing a batch normalisation, different outliers like very dark or bright images are being somewhat accounted for.

2.8 Model loss and accuracy

Two metrics commonly used for estimating the performance of a model is the model loss and the model accuracy. The loss is calculated on training and validation, and it is a summation of the errors made in the training or validation parts of the dataset. The summation of the errors in neural networks is often a summation of the negative log-likelihood and the residual sum of squares. The result gives an indication of how good the model performs on the two parts after each iteration of optimisation. The goal is to minimize, or ideally to reduce the loss function after each iteration. Generally, the lower the loss the better the model. This can be achieved by changing the model's weight vector values through different optimisation methods, described in 2.6. However, there are exceptions to this, like when over-fitting to the training data has occurred. The overfitting process is described in 2.10.

Accuracy is the percentage of accurate classifications. This is determined by feeding test samples to the model after the model parameters have been optimised and fixed, and no more learning is occurring. From this, the number of mistakes that the model makes are compared to the true targets and the percentage of misclassifications are calculated. One example of how accuracy is calculated is when we have 1000 test samples and the model classifies 903 of those correctly. In that case, the model's accuracy is 90.3%.

2.9 Receiver Operating Characteristic curve

Receiver Operating Characteristic curve (ROC curve) is a common and useful plot when predicting probabilities. It has the false positive rate, i.e. the false alarm rate on the x-axis and the true positive rate, i.e. the hit rate on the y-axis. The true positive rate is a description of how good the model is at predicting the positive class when the actual outcome is positive. It corresponds to the calculated number of true positives divided by the sum of the number of the true positives and the false positives. The true positive rate is also called sensitivity. The false positive rate is calculated by the number of false positives divided by the sum of the false positives and true negatives. This metric provides information on how often a positive class is predicted even though the actual outcome is negative. A confusion matrix also displays these metrics but is not as informative as a ROC curve.

From this, the area under the ROC curve (AUC) can be used to summarize the performance of the model. Generally, a good model should have a curve that is curved closely to the top left corner of the plot. If the curve looks more like a straight line, it indicates that model can't distinguish between different classes and that it could be guessing randomly. From these metrics, the micro average performance of the model can be estimated. In a model with multiple classes, the micro average aggregates the contributions of all classes and computes the average performance of the model from these.

2.10 Overfitting

Overfitting is when a function or a model is trained and fitted too close to a specific set of data points. To avoid this, methods like data augmentation, drop out and batch normalisation can be used. When using CNNs and applying data augmentation, the generalizability of the model increases. It generates new training images from the input data by applying random transformations which manipulates the images. By letting the model see new and slightly altered versions of the original images it can learn more robust features that are common for the input images even if they are slightly modified. More robust feature extraction results in a

more accurate model. Simply put, data augmentation alters the images in order to find new patterns as well as to generate more training images (Moen *et al.* 2019).

2.11 Problems with CNNs

One significant problem with CNNs is when there is a lack of labelled data. Evidence suggests that the answer to this bottleneck is transfer learning (Zhang *et al.* 2016), and that deep CNNs in combination with transfer learning results in highly accurate classifications (Kensert *et al.* 2019).

3 Materials and methods

3.1 Frameworks, libraries and tools

A suggestion for this project was to use the TensorFlow framework due to its popularity and that it is well documented. It is an interface and a tool for implementing and executing machine learning algorithms. TensorFlow has a flexible ecosystem of tools and libraries that allows for easy building and deployment of ML applications (Abadi *et al.* 2016). Another suggestion was to use the Keras framework together with TensorFlow. Keras is a library that provides building blocks for deep learning networks, and they are built by using TensorFlow in this case (Ketkar 2017). The Sequential API is one way of building a Keras model, which was used in this project. It allows for building the Deep Learning model layer by layer (Keras Documentation).

Keras contains a class called the ImageDataGenerator, which performs data augmentation. It accepts a batch of training images, takes that batch and applies a series of random manipulations to each training image in the batch. Common manipulations are rotations, resizing and shearing. It then replaces the original batch with the new manipulated batch. The CNN is trained on this manipulated batch, so the original data is not used for training the CNN (Keras Documentation). One subclass of the ImageDataGenerator is called FlowFromDataframe. It generates batches of augmented or normalized data by using a data frame and a path to a directory (Keras Documentation). This is preferably used when multi labelled image data is used and when the image data is not separated into different directories, which means that all images with for example a “car” label are in one directory and all images with a “dog” label are in another directory. For this project, the data was sometimes labelled with several labels per image and was not separated into different directories, hence the need for this method. Two common Python libraries that were used in this project was Numpy and Pandas. Numpy supports the use of large, multidimensional arrays and matrices. Besides that, it comes with many mathematical functions that can operate on these arrays, such as linear algebra and Fourier transform (NumPy). Pandas is a data analysis library. One of its many benefits is that it can take a CSV or TSV file as an input and create a Python data frame, which is an object that consists of rows and columns, similar to tables. This simplifies the work compared to working with lists and dictionaries where you often have to use list comprehension or for loops to read and process the input data. Since the metadata file with labels was used as an input in this project, Pandas was used to make that work easier (pandas-dev 2020).

3.2 The data

The used data consisted of the Human Bone Osteosarcoma Epithelial Cells, which is called the U2OS cell line. The cells were originally harvested from the bone tissue of a human fifteen-year-old female with osteosarcoma. In 1964, the first cells were harvested from a moderately differentiated sarcoma in the tibia, also called the shinbone. The cells in the U2OS cell line are positive for insulin-like growth factors 1 and 2 receptors (IGF-1 and IGF-2). They also express several antigens, including blood type A, Rh+, HLA A2, Aw30, B12, Bw35 and B40 (+/-) (Niforou *et al.* 2008, Nikon's MicroscopyU).

The cells were dyed using Cell Painting (Bray *et al.* 2016), which is a microscopy-based morphological profiling assay with six multiplexed fluorescent dyes. The six dyes were imaged in five channels in order to highlight eight different cellular components and organelles:

- Channel 1 shows the Hoechst 33342 dye that binds to and stains the DNA
- Channel 2 shows the SYTO 9 stain which binds to nucleoli and cytoplasmic RNA
- Channel 3 shows the MitoTracker Deep Red dye which is used for mitochondrial staining.
- Channel 4 shows the Concanavalin A dye which stains the endoplasmic reticulum
- Channel 5 shows both the phalloidin and WGA dye. Phalloidin stains actin, and WGA stains Golgi and plasma membranes.

The cells from the U2OS cell line was plated in 21 multiwell plates where 60 wells were used on each plate. The wells were then treated with 378 compounds from Selleck (Selleck Kinase Inhibitor Library) and controls in four different concentrations (10 µM, 6 µM, 5 µM and 4 µM in a total well concentration of 100 µL). This was done by the research team, which means that the images were generated in-house. After this, they were stained, fixed and imaged on a high-throughput microscope after 48 hours of treatment. Each well was photographed in nine sites in five channels, comprising five sets of 12300 single channel images.

3.3 Metadata, annotation and understanding the metadata

A metadata file was created from the 12300 images, which was named "dataset.csv" and contained 12301 rows and 33 columns. The rows represented different objects, which were the images in this case, and the columns represented the features for each image. To fully understand the data, another file with explanations of the rows and columns was created. For some compounds, the metadata file lacked annotations regarding the compound name and their targets. As an example, some of the compound names could not be interpreted, and some wells had been treated with several compounds but the annotation did not include them all. The annotation was done by using Selleck (Selleck Kinase Inhibitor Library), a webpage that contains annotations for kinase inhibitors among other chemical and biological compounds. This was done at this stage in order to make the profiling of the kinases more correct and easier for me to interpret. Also, by doing this at that stage, it gave me a better understanding of the data I was working with. The resulting annotated file can be seen in Appendix A.

3.4 Join colour channels

The five colour channels described in 3.2 were joined to get the images in a RGB (three-dimensional input) format, although it's optional for the custom-built models used in the

beginning of this project. Joining different colour channels allows for the visualization of several cell compartments at the same time. Also, many pre-trained convolutional neural networks require the input images to be in an RGB format. Later in the project, several pre-trained networks were used. How the colour channels were joined could be varied, and different combinations might reveal different patterns that can be used for classification. In this project, the colour channels were joined in two ways: MiCoPh and MiSyHo (input one), and MiCoPh and SyHoPh (input two). The names are made up of the first two letters in the used dyes. As an example, MiCoPh stands for MitoTracker Deep Red, Concanavalin and Phalloidin and WHA dye. Examples of this looked can be seen in figure 4.

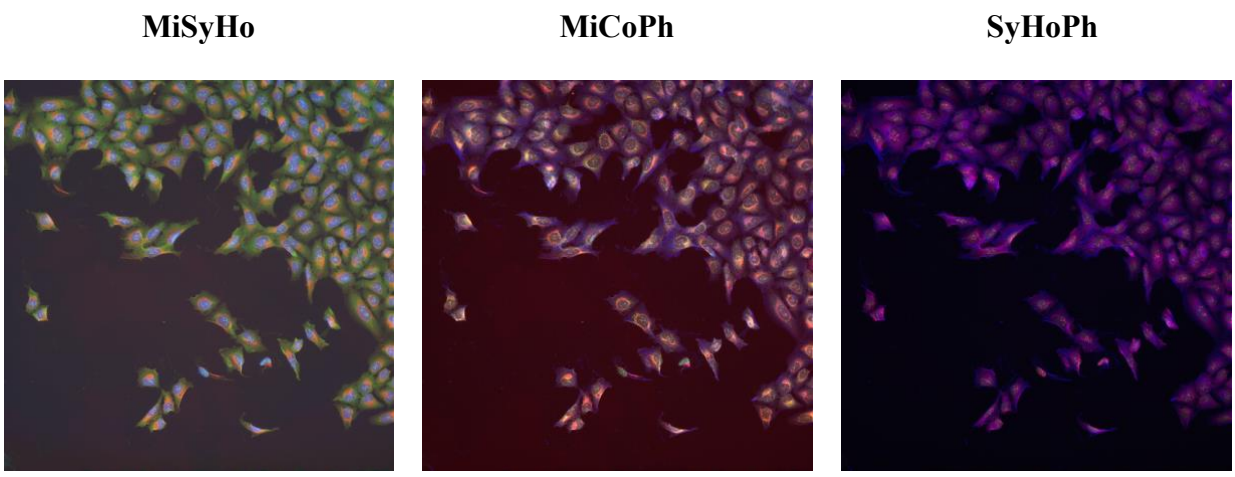


Figure 4. Three examples of what one same image look like with the differently joined sets of colour channels. To the left: One image with the colour channels MiSyHo. In the middle: the same image but with the colour channels MiSyHo. To the right: the same image but with the colour channels SyHoPh.

Maris Lapins' script was used to join the colour channels and create a data frame with labels for the images called "labels.csv". This resulted in two folders consisting of images that had been joined with three joined colour channels per folder. Folder one contained channel 3, 4 and 5 described in 3.2 and was named "MiCoPh" and folder two contained channel 1, 2 and 3 and was named "MiSyHo". It also resulted in the data frame "labels.csv" that consisted of 12300 rows, i.e. all images, and eight columns containing:

- The image index: 0 - 12300
- The plate number: P009063 – P009083
- Well position: B02 – G11
- Site: 1 - 9
- Well role: Compound or control
- Compound ID: Ex. CBK013406 (a compound) or DMSO (control)
- Therapeutic class: Ex. EGFR (a target compound)
- Compound concentration: 10, 6, 5 or 4 micro molars

3.5 Compound vs control, 3 colour channels, part of the dataset

In the next step, a simpler model was created. I started off by comparing controls and compounds to see if the classifier worked at a more basic level. Only data from one of 21

plates were used at this stage, i.e. 540 images. The first model that was created was a combination of a LeNet-5 architecture (LeCun) and an AlexNet architecture (Krizhevsky *et al.* 2012). When that model was up and running, the FlowFromDataframe method was applied as described in 3.1. In order to use the FlowFromDataframe method, a new data frame with the right format was created according to a tutorial (Vijayabhaskar J 2019). Keras and TensorFlow were also implemented at this stage (Vijayabhaskar J 2020). When these preparations were working, three colour channels, "MiCoPh", was used as an input to create the data frame and to run the model. As previously mentioned, only 540 images were used at this point.

3.6 Compound vs control, all 5 colour channels, whole dataset

When that simple model for only three of the colour channels and a small part of the dataset was working, it was time to use all colour channels and the whole dataset of 12300 images as input in the same model. At this stage, the code was also simplified and restructured by adding some new functions. Some different optimisers were tried and parameters were tried:

- RMSprop with lr=0.0001
- Added 90 degrees rotation range to the ImageDataGenerator to create more versions of the images
- The Adam optimiser with default parameters, did not improve the loss
- Nadam Optimiser with default parameters, did not improve the loss
- SGD with default parameters
- SGD with default parameters and increased batch size from 32 to 128.
- RMSprop with lr=0.0001 and batch size=128

3.7 Kinase inhibitors vs controls

After different parameters and optimisation methods had been tried, I moved on to profiling of the different kinase inhibitors and controls on the whole dataset with FlowFromDataframe and ImageDataGenerator. Different parameters, optimisation methods and different batch sizes were used:

- RMSprop with batch size 128
- SGD with batch size 128
- RMSprop with batch size 32 and a lower learning rate of 0.00001
- RMSprop with batch size 128 and learning rate 0.00001

I continued on to deeper prebuilt models. These models consisted of two networks whose final layers were concatenated in the end. Two networks were used to enable the implementation of transfer learning by using ImageNet weights. This means that two networks were created with the input shape (224, 224, 3) with ImageNet weights, otherwise one network would have been created with input shape (224, 224, 5) and no ImageNet weights. In order achieve this, the final top layers, i.e. the fully connected layers of each network which works as the classifier were not included in the models. Instead, the last output from each model was concatenated to get one shared output instead of two separates. Lastly, global average pooling, dropout and dense was added as some common final layers before compiling the model. For this step, the network MobileNetV2 was tried with ImageNet weights. It was tried together with RMSprop, a batch size of 32 and a learning rate of 0.00003. It was also tested with SGD, batch size 32 and a learning rate of 0.00001. These results can be seen in 4.2.

3.8 Two stream models profiling kinase families, data divided after wells

At this stage, the focus shifted to profiling kinase families instead of each of the compounds since fewer classes and larger classes might be easier to predict for the models. Three two stream models for VGG16 were created, all with a learning rate of 0.0001, a batch size of 32 and two of the models used ImageNet weights. For the optimisers, one model used SGD, one used Adam and the third did also use Adam but did not have any ImageNet weights.
Two stream models were created to enable the use of both sets of joined RGB images as input. By doing that, all colour channels were used as input. The basic layers of convolution and pooling for each set of input images remained, but the last fully connected layer was removed. Instead, global average pooling was performed before the both outputs were concatenated. Before the classes were predicted, the concatenated output went through dropout and two new hidden layers with ReLu and Sigmoid as activation functions. The exact same parameters and structure was used when creating three MobileNetV2 two stream models and three ResNet50 two stream models. A script to create subplots of the results of the three models for each architecture was created as well.

3.9 Join colour channels in other ways

I tried to join colour channels differently and use that as a new input in the same models as before in 3.6. When the colour channels were joined the second time, channel 5 was used twice. As it is described in 3.2, channel 5 shows both the Phalloidin and WGA dye. This channel was chosen this time since it includes several cell compartments. Phalloidin stains actin, and WGA stains Golgi and plasma membranes. When this was done, the new joining of the colour channels was used as an input in the same three models as in 3.6. Every model was run two times with ImageNet weights and one time with no pretrained weights. The exact same parameters were used for every run except for the optimiser. The used optimisers were Adam and SGD, where SGD was used in one pretrained model and Adam was used in both one pretrained model and in the model with no pretrained weights. To summarize, the exact same protocol and parameters as in 3.6 were applied to these models, except for the new colour channels as input. This was done in order to simplify the analysis and get a good comparison of how the models perform on different colour channels, where the only difference between the models was the used colour channels as an input. However, in some cases, like the MobileNetV2 models in this section, the number of epochs were increased to see if the SGD optimiser would behave differently. When comparing the models, it was clear that the models with SGD hadn't finished. Therefore, all models with SGD and MiCoPh and SyHoPh as input was run again but this time with a learning rate of 0.1 instead of 0.0001. This means that three new models were created, which can be seen in column two in the figures in 4.3.

4 Results

This part of the report presents the accumulated results. There are four sections covering the first binary models, the profiling of kinase families, the micro average values for the profiling of kinase families and the ROC areas for some of the models.

4.1 Loss for the binary models

This section presents the very first results. At this stage, the models were binary and were only profiling compounds versus controls, not distinguishing between the different kinase inhibitors. For some of the figures below, Adam and SGD was used as an optimiser.

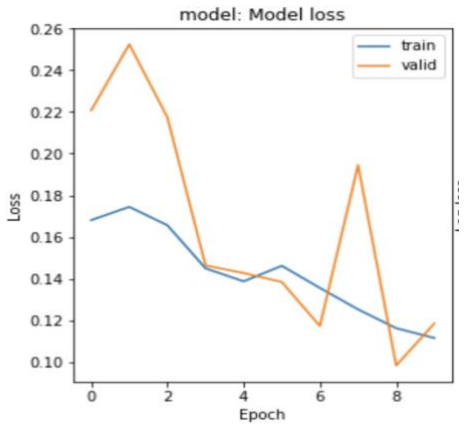
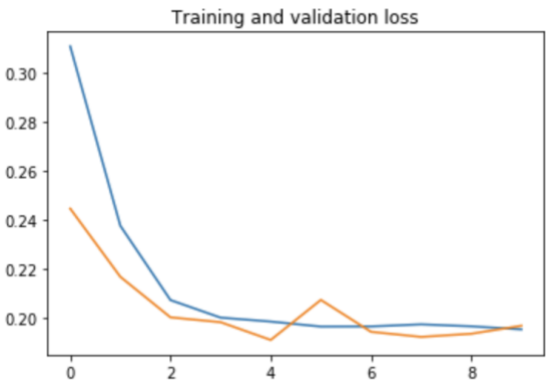
540 images as input	12300 images as input
	

Figure 5. To the left: A model made of the combination of LeNet-5 and AlexNet architectures. Only one set of three colour channels, MiCoPh and 540 images from a single plate were used as in input for this and only compound versus control was being predicted. To the right: The whole dataset of 12300 images from all plates were used as an input. A batch size of 32 was used for both data sets. The blue line represents the training loss and the yellow line represents the validation loss.

For the model in figure 5 with the smaller dataset, the loss is decreasing although there are some jumps. The loss reaches approximately 0.11. For the larger dataset in figure 5, the loss is decreasing smoothly but only reaches 0.2.

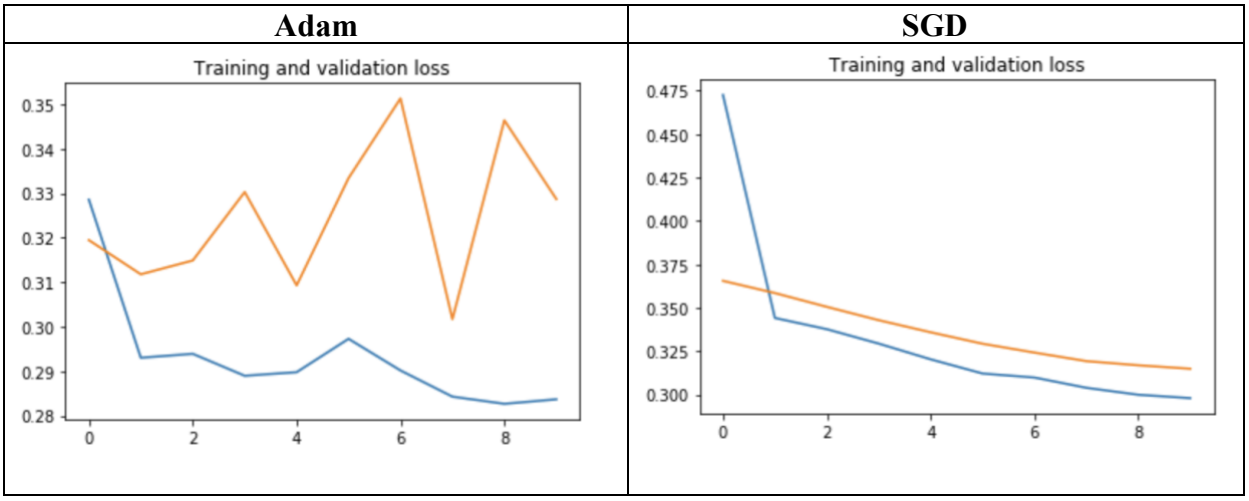


Figure 6. The whole dataset as input with Adam or SGD as optimisers, profiling compounds (treated cells) versus controls (untreated cells). The blue line represents the training loss and the yellow line represents the validation loss.

Regarding figure 6 with Adam, the training loss is decreasing with jumps down to 0.28, and the validation loss is rather increasing than decreasing. The training loss for SGD decreases smoothly down to 0.3 together with the validation loss.

4.2 Loss and accuracy for kinase families with two-stream models

This section presents the first results for the two-stream models that utilised transfer learning. The models were profiling kinase families and controls, i.e. 113 classes. This was tried with the optimisers RMSprop and SGD.

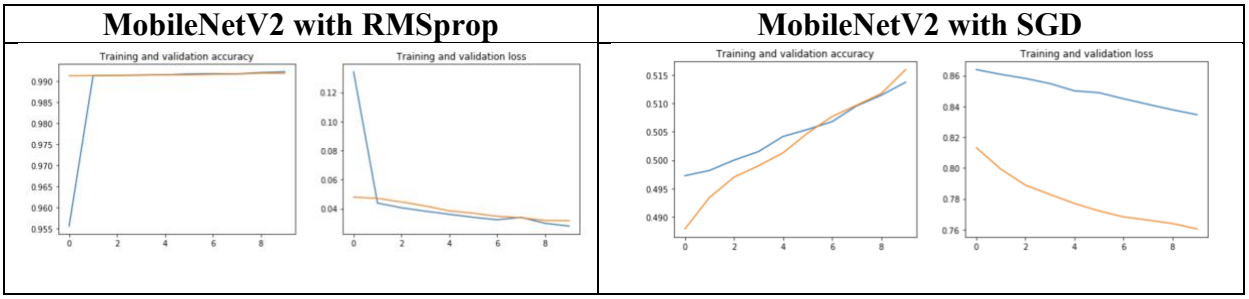


Figure 7. The first two stream models with MobileNetV2 architecture tried with RMSprop and SGD. The figures show the training and validation loss, as well as the training and validation accuracy. The blue line represents the training loss and the yellow line represents the validation loss.

The figure with RMSprop shows a steady decrease in training and validation loss, as well as a steady increase in accuracy where an accuracy of 99% is reached after one epoch. The other figure depicts a somewhat smoothly descending training and validation loss that only reaches 0.84 for training and 0.76 for validation. The accuracy ascends to 51.5%.

4.3 Loss and accuracy for the two-stream models with different inputs

In this section, the loss and accuracy for the VGG16, the MobileNetV2 and the ResNet50 models when classified into kinase families are presented. The results for the two different inputs, MiCoPh and MiSyHo as well as MiCoPh and SyHoPh can be viewed. In the figures for each input, we can see three different plots. One pretrained model with SGD as an optimiser, one pretrained model with Adam as an optimiser and one model without pretrained weights and Adam as an optimiser. The pretrained models used ImageNet weights. The learning rate was 0.0001 and the batch size was 32 for all of the models with MiCoPh and MiSyHo as input. For the second input, a learning rate of 0.01 was used for the SGD optimiser and 0.0001 was used for the models with Adam as an optimiser.

4.3.1 VGG16

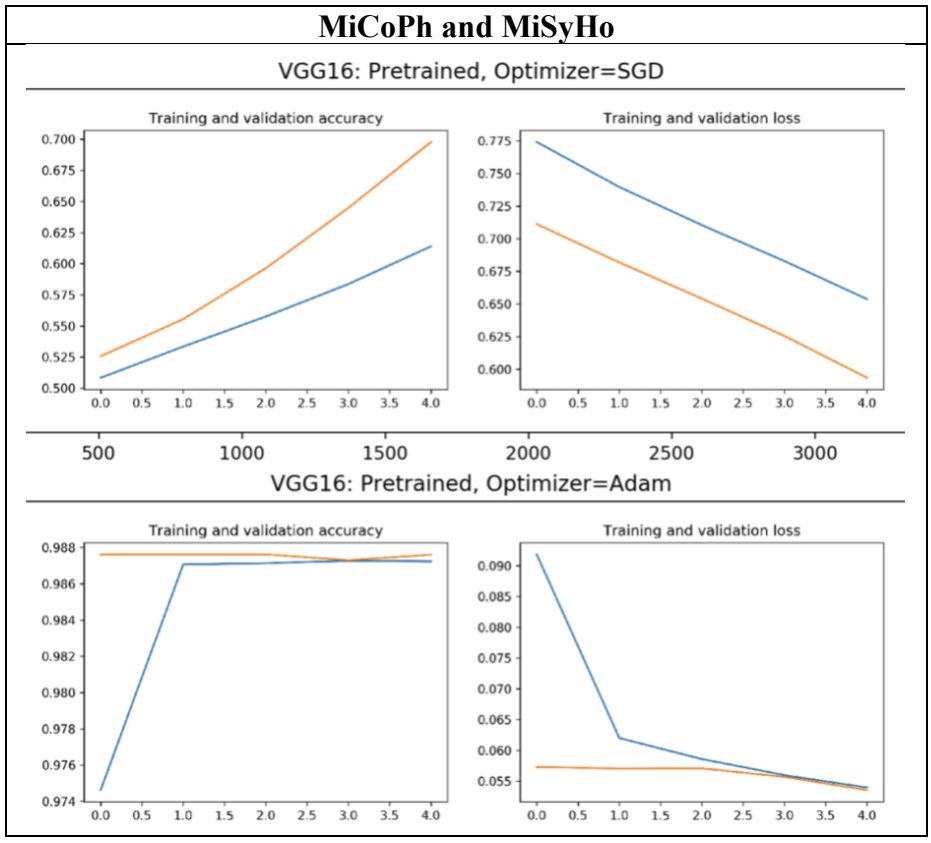


Figure 8. Training and validation accuracy for the three VGG16 models with MiCoPh and MiSyHo as input. The yellow line represents validation and the blue line represents the training. In the figures to the left, the training and validation accuracy are visualised. The number of epochs can be seen on the x-axis and the accuracy on the y-axis. To the right, the training and validation loss is visualised. The number of epochs can be seen on the x-axis and the accuracy is on the y-axis.

The training loss is decreasing smoothly for all models, and the loss is decreasing for all of them. However, the model with SGD should have been run with more epochs to see that the curve flattens out. The accuracy is increasing for SGD, but is stable from the beginning for Adam. The VGG16 model without pretrained weights was not possible to run with the number of available GPUs, hence no results are shown for that.

4.3.2 ResNet50

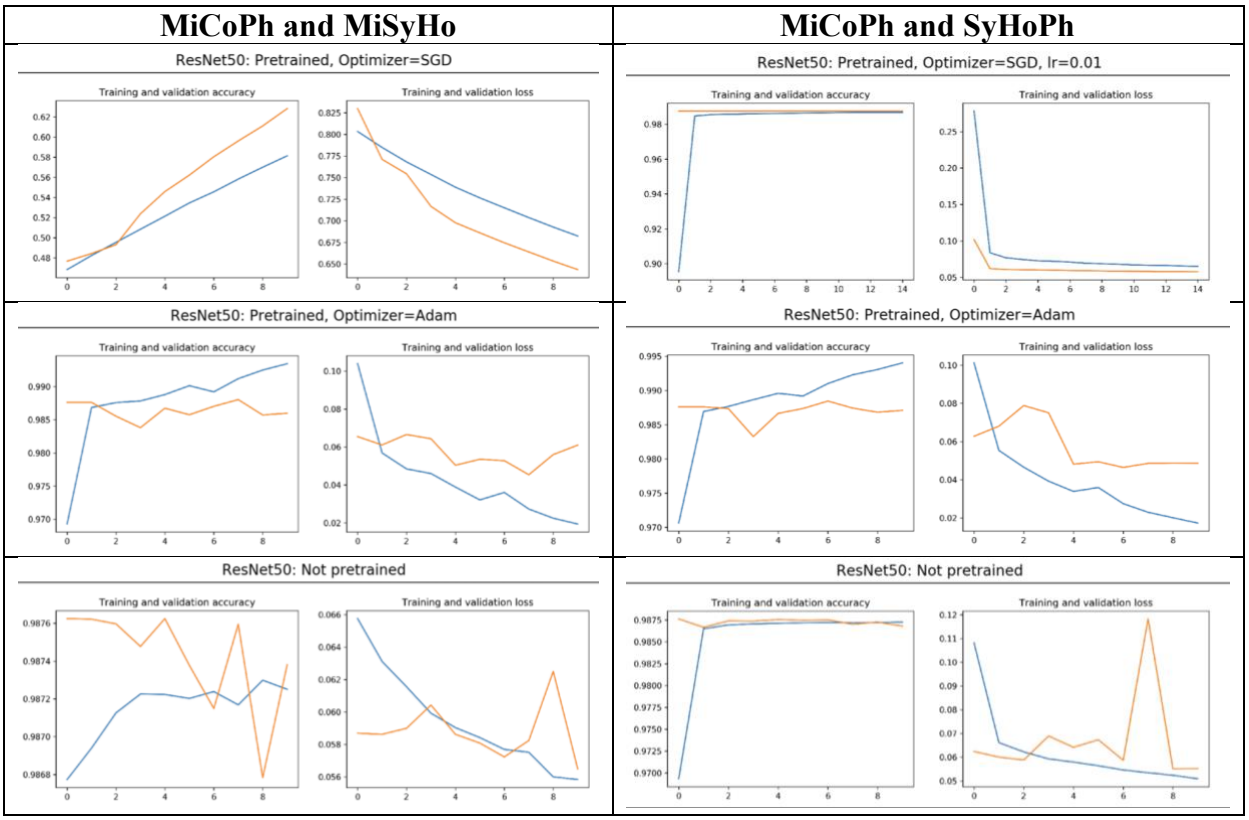


Figure 9. Training and validation accuracy as well as loss for the three ResNet50 models with MiCoPh and MiSyHo as input, and the three models with MiCoPh and SyHoPh as input. The yellow lines represent validation and the blue lines represent the training. In the figures to the left for each input, the training and validation accuracy are visualised. The number of epochs can be seen on the x-axis and the accuracy on the y-axis. To the right for each input, the training and validation loss is visualised. The number of epochs can be seen on the x-axis and the accuracy is on the y-axis.

For the models with MiCoPh and MiSyHo as input, some different trends can be seen. For the model with SGD as an optimiser, the accuracy is increasing and the loss is decreasing after each iteration. However, the loss is quite high, the accuracy only reaches approximately 60% and the loss curve never reaches a plateau. The loss for the pretrained model with Adam as an optimiser is lower, and it is decreasing for the training although there are some jumps. For the validation, it's barely decreasing at all but is rather fluctuating around the same loss. The accuracy for that models is increasing slightly for the training, up to 99.5%, but is fluctuating around 98.7% for validation. The loss for the model without pretrained weights is decreasing smoothly for the training but is decreasing with a big jump for the validation. The accuracy of the training is increasing but is decreasing with large jumps for the validation.

When MiCoPh and SyHoPh was used as input, a higher learning rate of 0.01 and five more epochs for SGD was used. The loss for that model flattens out after only one epoch, and the loss is reaching 0.05. The accuracy reaches 0.98 after one epoch as well. When Adam was used, the training loss reached 0.02 and the training accuracy reached 99.5% with fluctuation for both the validation accuracy and the validation loss. The model without pretrained weights has a smooth training loss which reaches 0.05 and a training accuracy that reaches 98.75% after one epoch.

4.3.3 MobileNetV2

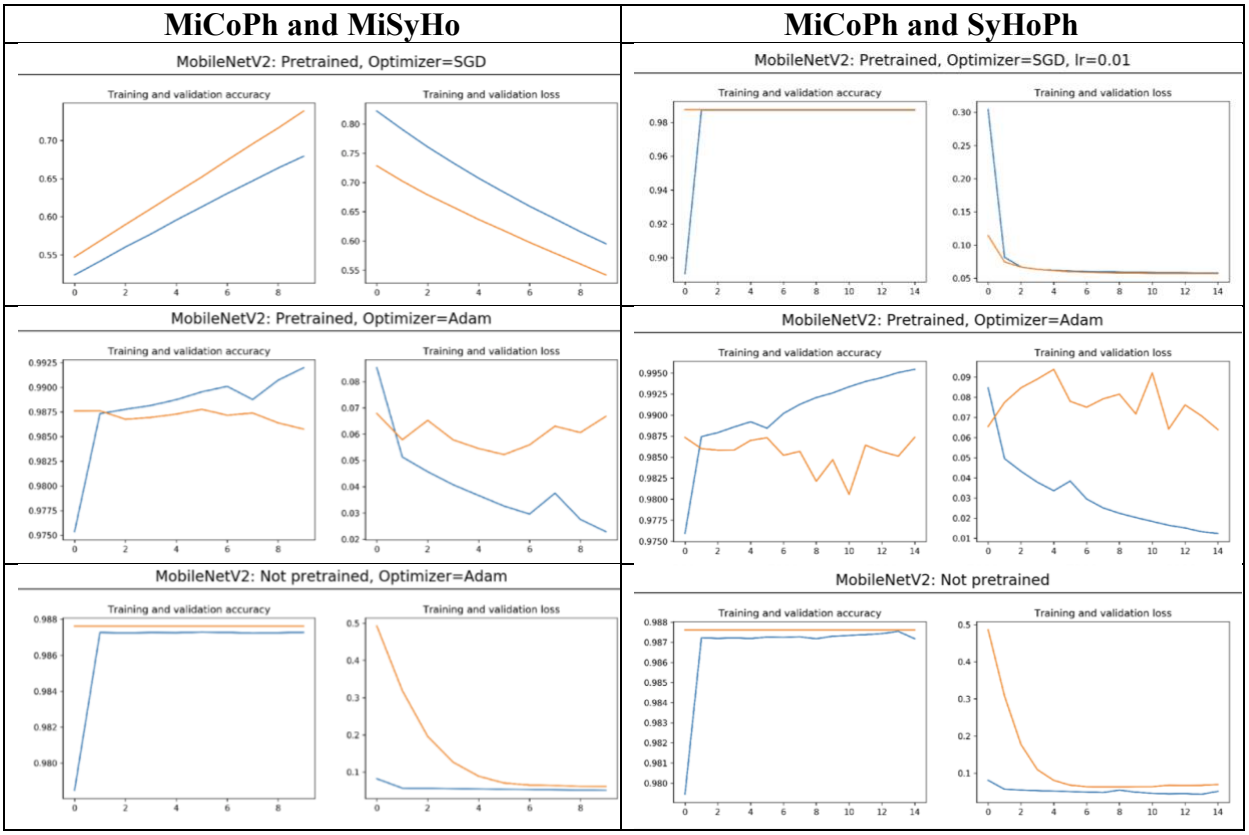


Figure 10. Training and validation accuracy as well as loss for the three MobileNetV2 models with MiCoPh and MiSyHo as input and the three models with MiCoPh and SyHoPh. The yellow lines represent the validation and the blue lines represent the training. In the figures to the left for both inputs, the training and validation accuracy are visualised. The number of epochs can be seen on the x-axis and the accuracy on the y-axis. To the right for both inputs, the training and validation loss is visualised. The number of epochs can be seen on the x-axis and the accuracy is on the y-axis.

For the MobileNetV2 models with input one and with SGD as an optimiser, the same argument can be applied as for both the VGG16 and ResNet50. Both lines are descending, but they never converge or reaches a plateau. Regarding the pretrained model with Adam as an optimiser, the training is decreasing with a small jump at epoch 7, but the validation is decreasing and the starts to ascend at epoch 5. The model without pretrained weights and Adam as an optimiser show two decreasing curves that are converging towards the end. A low loss has already been achieved before the first epoch. However, the scale for that figure is

different compared with the other ones which makes it hard to comment the result further. The training and validation accuracy for all models looks smooth. A high accuracy is reached early on for both models with Adam as an optimiser, but for the model with SGD the training and validation accuracy is increasing steadily but does only end at approximately 65% accuracy for the training and 75% for the validation. When input two was used, all models had five more epochs. The same pattern as for the ResNet50 models can be seen here, especially for SGD. The pretrained model with Adam as an optimiser shows a steady decrease of the training loss with fluctuations in the validation loss, and the same pattern is mirrored for the accuracy. For the model without pretrained weights, both the loss and accuracy looks the same for input one and two.

4.4 Micro average for classification of kinase families

In this section the micro average values for the VGG16, the MobileNetV2 and the ResNet50 models are presented when classified into kinase families as described in 3.6.

Table 1: Table with the micro average values for VGG16, MobileNetV2 and ResNet50 and the different optimisers. The numbers show the average performance of the model, considering the contributions from all classes in these multiclass models. For SGD, two different learning rates were used as can be seen in the two columns for SGD.

	MiCoPh and MiSyHo			**MiCoPh and SyHoPh**		
	Pretrained SGD (lr=0.0001)	**Pretrained Adam**	**Not pretrained, Adam**	**Pretrained SGD (lr=0.01)**	**Pretrained Adam**	**Not pretrained Adam**
ResNet50	0.531	0.912	0.831	0.813	0.939	0.868
MobileNetV2	0.512	0.904	0.499	0.824	0.932	0.768
VGG16	0.552	0.872	0.766	0.826	0.893	0.822

Table 1 shows that the micro average for the models with SGD as an optimiser and with a learning rate of 0.0001 was between 0.51 and 0.55. This indicates that these models were randomly guessing when classifying the kinase families. However, when the learning rate was 0.01 for SGD, the micro average drastically improved with values from 0.8127 to 0.8255. The results for the models with Adam as an optimiser had similar micro average values. However, all models with MiCoPh and SyHoPh as an input performed marginally better when looking at the second decimal. The models without pretrained weights and with Adam as an optimiser performs slightly better for the MiCoPh and SyHoPh as an input for ResNet50 model. For MobileNetV2, there is a much bigger difference in the micro average where the model performs 26.8 % better when MiCoPh and SyHoPh were used as input. For the VGG16, there is also a difference in micro average where input two with Adam performed best with a micro average of 0.893. Overall, Table 1 states that ResNet50 and MobileNetV2 with input two, pretraining and Adam performed equally good.

4.5 ROC values for kinase families

Details for the two best models are presented in this section, i.e. ResNet50 and MobileNetV2. The details for VGG16 were not analysed further due to that those models had the lowest micro average values in Table 1. This part shows the five top ROC areas and the five lowest ROC areas for two models and the optimisers. The presented results in this section are only the ROC areas for the MiCoPh and SyHoPh input. The reason for that was a mistake which caused the results from the first input to not be saved. ROC AUC plots were constructed as well, but due to poor readability, these tables with the individual ROC areas are presented instead.

Table 2: The highest ROC areas and the lowest ROC areas for "targets", i.e. the kinase families. The highest and lowest values are presented for the ResNet50 model and the SGD optimiser, the Adam optimiser and a non-pretrained model with Adam as an optimiser.

ResNet50					
SGD		Adam		Not pretrained with Adam	
target/activity	**roc area**	**target/activity**	**roc area**	**target/activity**	**roc area**
S1PReceptor	0.112494	Rac	0.602633	CDK,AuroraKinase	0.300089
ATM/ATR	0.197210	cKit,VEGrFR	0.616687	FAK	0.322535
CHK	0.227622	VEGrFR,Src,Raf,Bcr-Abl	0.645917	PDGFR,cKit,Bcr-Abl	0.364951
Akt,S6kinase	0.257965	Akt,PDK-1	0.710486	AMPK	0.440070
c-Met,FGFR,PDGFR	0.286836	CDK,AuroraKinase	0.723963	JAK,EGFR	0.452439
...	...	...	...	...	...
Syk	0.724694	PDGFR,cKit,Bcr-Abl	0.998813	c-Met,FGFR,PDGFR	0.963766
AMPK	0.730672	DUB,Bcr-Abl	0.999152	VEGrFR,Flt	0.966444
JAK,EGFR	0.750347	mTOR,PI3K,Akt	0.999866	cKit,Raf,VEGrFR	0.968764
CDK,AuroraKinase	0.792191	PDK-1,IKK	1.000000	AuroraKinase,FGFR,Bcr-Abl,c-RET,Src	0.974074
I¦ÊB/IKK	0.826551	CHK	1.000000	DUB,Bcr-Abl	0.988130

In Table 2, Aurora Kinase, IKK and "DUB, Bcr-Abl" is among the kinase families with the highest ROC area. Among the five smallest ROC areas we find Akt and PDGFR. Overall, the pretrained model with Adam shows higher ROC areas than the other two, and the model without pretrained weights and Adam shows higher ROC areas than the model with SGD.

Table 3: The highest ROC areas and the lowest ROC areas for “targets”, i.e. the kinase families. The highest and lowest values are presented for the MobileNetV2 model and the SGD optimiser, the Adam optimiser and a non-pretrained model with Adam as an optimiser.

MobileNetV2

SGD

target/activity	roc area
PI3K,mTOR	0.375308
PAK	0.377446
FAK	0.398617
DNA-PK,PI3K,mTOR	0.404685
c-Met,cKit,PDGFR,Flt,c-RET	0.414864
...	...
Flt,Bcr-Abl,AuroraKinase	0.776930
CHK	0.803168
mTOR,PI3K,Akt	0.825167
AuroraKinase,FGFR,Bcr-Abl,c-RET,Src	0.826863
PLK	0.842705

Adam

target/activity	roc area
JAK,EGFR	0.604349
JAK	0.680467
AMPK	0.718701
I¦ÊB/IKK	0.751272
VEGrFR,Src,Raf,Bcr-Abl	0.761669
...	...
ERK	0.999286
Flt,AuroraKinase,VEGrFR	0.999509
CHK	0.999911
cKit,FGFR,Flt,VEGrFR,PDGFR	1.000000
Flt,Bcr-Abl,AuroraKinase	1.000000

Not pretrained with Adam

target/activity	roc area
PDGFR,cKit,Bcr-Abl	0.025908
Flt,AuroraKinase,VEGrFR	0.059170
PDGFR,VEGrFR	0.059750
VEGrFR,cKit,Flt	0.062651
PLK	0.102393
...	...
BTK	0.779681
S1PReceptor	0.799911
cKit,Raf,VEGrFR	0.847724
Flt,Bcr-Abl,AuroraKinase	0.886256
I¦ÊB/IKK	0.912963

In Table 3 the CHK and “Flt,Bcr-Abl,Aurorakinase” is among the kinase families that were predicted with the highest ROC areas. Among the lower ROC areas PDGFR, Pl3K, mTOR, JAK, EGFR is found. Overall, the pretrained model with Adam has higher ROC areas than the other two, and higher ROC areas compared to the other architectures as well.

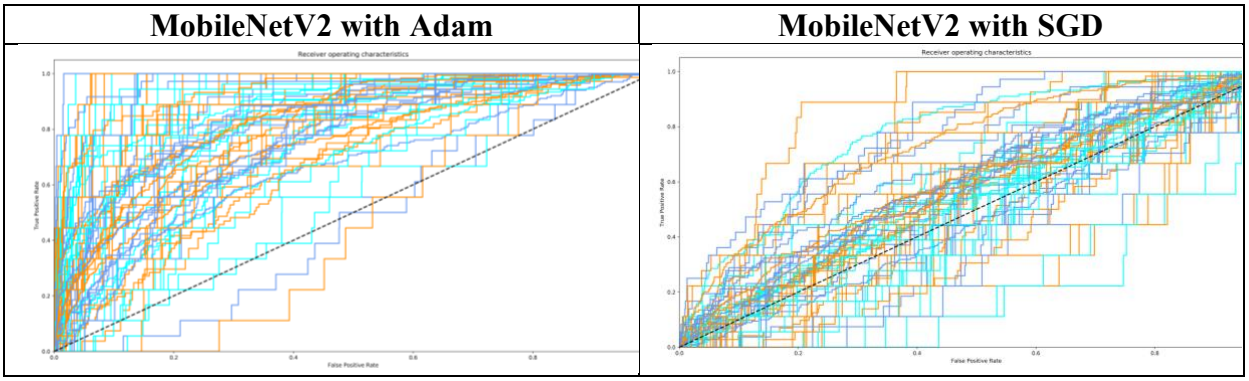


Figure 11. ROC curve for the two models with MobileNetV2, with ROC curves for all 113 classes. The false positive rate is on the X-axis and the true positive rate is on the Y-axis. To the left is the pretrained model with Adam and learning rate 0.0001, to the right is the pretrained model with SGD and learning rate 0.01. The input with MiCoPh and SyHoPh was used for both models.

Figure 11 shows that the MobileNetV2 model performed better with Adam than with SGD. For Adam, the ROC curves are mostly centred around the top left corner, which indicates a lot of true positives. For the model with SGD, the ROC curves are mostly centred around the middle, which indicates that the model was doing some random guessing.

5 Discussion

5.1 Loss for binary models

The binary models never reached a low loss. However, the model for a single plate reached a lower loss than the whole dataset. A possible explanation might be that there is always some systematic error, and additionally, that there are variations between plates in illumination and cell growth. Nonetheless, the smaller dataset showed several jumps in both training loss and validation loss compared to the whole dataset. The learning rate might have been too high, but it is more likely that the dataset was too small. For the binary models with the whole dataset, SGD descends smoother compared to Adam, and both of them reaches approximately 0.3 in training loss. The conclusion is that the whole dataset and SGD seems to perform best for the binary models.

5.2 Loss and accuracy for the first kinase families with two stream models

The difference between RMSprop and SGD is huge. RMSprop reached an accuracy of 99% and a loss of 0.04, while SGD only reached an accuracy of 51.5% and a validation loss of 0.76. From the figures and these numbers, you can conclude that SGD needed more training. This is especially noticeable since the curve never flattens out, which in turn indicates either that the learning rate was too low or that the number of epochs was too small.

5.3 VGG16

The VGG16 model without pretrained weights was hard to execute, as the system often suffered a "Resource Exhausted Error" even if multiple GPUs were used. The reason for this might be that VGG16 is the most computationally heavy architecture used in this project, with 140 million trainable parameters compared to four million in MobileNetV2 and 25 million in ResNet50. At one stage, the predicted values were the same for all classes in a functioning VGG16 model without pre-trained weights, proving that the model had not learned. Perhaps, the reason was that all ReLU died in some layer. The risk for dying ReLU's is higher if the learning rate is high, but regardless, this highlights that pretrained weights improve the performance. In turn, the need for pre-trained weights proves that transfer learning is important when training deep neural networks. Without pre-trained ImageNet weights, it is impossible to train a deep neural network on a dataset with only a thousand images, which is considered to be a small dataset when training deep neural networks.

5.4 ResNet50

Overall, both of the ResNet50 models with Adam as an optimiser had a steadily decreasing training loss. The same behaviour is found in the training and validation accuracy as well. When the validation loss is decreasing with jumps in those models, it might be caused by a learning rate that is too high, a dataset that is too small or simply because of the dropout rate that is included in the final layers of the models. If the yellow lines start to ascend, it implies that the model is over fitted. However, the yellow line also indicates when the model is done with the training, and if the yellow line ascends, peaks and then descends, that no more epochs are necessary. This is well represented for the non-pretrained ResNet50 model with Adam as an optimiser.

The pretrained model with Adam has a validation loss that is jumping a bit, a training accuracy that reaches almost 99.5% and a validation accuracy that is fluctuating around 98.7%. The reason for this is that for every epoch, only 50% of the dataset is used (if dropout is equal to 0.5). However, the overall trend is that the line is decreasing before it slightly ascends in the end. Hence, that model looks good as well. If a dropout rate of 0.1 would have been used, or if the learning rate was lower, the yellow line would be smoother with fewer jumps. Nonetheless, overall the models look good. When observing the figures for ResNet50 in 4.3, the accuracy is often very high, above 98% for all models except from when SGD had higher learning rate. That might be too good to be true.

From these plots, the SGD optimiser also seems to perform well. However, this cannot be verified due to the loss never reaching a plateau, and it is impossible to know how the rest of the epochs would look before reaching that plateau. The results for MiCoPh and SyHoPh are very similar to the ones where MiCoPh and MiSyHo were used as input. The difference between these results is mainly for SGD as an optimiser. For input with SyHoPh and with a higher learning rate of 0.01 compared to input one with a learning rate of 0.0001, the main difference is that the curve flattens out and reaches a loss of 0.05 instead of 0.650 for input one. This proves that the learning rate and the number of epochs are both crucial in order to get a model with high performance.

5.5 MobileNetV2

Regarding the results for MobileNetV2 with MiCoPh and MiSyHo, the model without pretrained weights appears very smooth. As mentioned, the scale is different in that plot compared to the other ones for MobileNetV2. The line starts pretty high, but it is actually quite random where the net predicts in the beginning. The same behaviour is mirrored for training and validation accuracy, where the training accuracy reaches 98.8% after one epoch, and the validation accuracy is constantly at 98.8%.

For the pretrained model with Adam, the validation is fluctuating regarding both accuracy and loss, and the training is steadily increasing up to 99.25% for the accuracy and decreasing down to 0.02 for the loss. it is hard to compare the two models with Adam as an optimiser, mainly because of the different scales in the plots, but also since the validation loss is ascending in the pretrained model which might indicate overfitting. However, for the pretrained model, the training loss falls to 0.02 and judging by the looks of the non-pretrained model it drops to approximately 0.02 as well. For the MobileNetV2 model with SGD as an optimiser, the same argument as for both the VGG16 and ResNet50 models with SGD can be applied. The learning rate has simply been too small compared to the number of epochs. A lot more epochs or a higher learning rate could have solved this problem. The scale on that plot in figure 10 starts even higher than the ones with Adam. The reason is the very low learning rate that was not a standard input for SGD. The training and validation accuracy for SGD is increasing steadily but ends at only approximately 65% accuracy for the training and 75% for the validation. This highlights that the model with SGD was not finished.

Regarding the results for MiCoPh and SyHoPh as input, the same trends as for the first input. The results show that more epochs do not improve the models without pretrained weights, and that a higher learning rate is beneficial for the SGD optimiser. For the pretrained model with Adam, the model seems to perform slightly better for the second input where the loss reaches 0.01 compared with 0.02 with the first input.

5.6 Micro average for models and optimisers

Concerning the results shown in table 1, the micro average values seem to improve slightly for input MiCoPh and SyHoPh when using the pretrained model with Adam and the not pretrained model with Adam. The Phalloidin and WGA dye that was used twice for input two stains the actin, the Golgi and the plasma membranes. In other words, several different cell compartments were visualized compared to some of the other used dyes. Perhaps, that might have visualized more kinase inhibitors present for input two compared to input one. Overall, the difference between the two inputs was small and might depend on other factors such as plate and compound concentration, as well as parameters in the code as well.

Table 1 with the values for the micro average shows that the pretrained model with Adam as an optimiser performed best for the ResNet50 models. The value for the model without the pretrained weights has a lower micro average, which makes sense since the pretrained weights should contribute with transfer learning which in turn should increase the classification power. The value for the pretrained model with SGD as an optimiser is surprisingly low and tells us that the model is basically random guessing. This is especially surprising since the model's loss and accuracy looks smooth in the figures. The reason behind this behaviour, as mentioned before, might be due to an insufficient number of epochs or that the learning rate was too high for the SGD optimiser, which means that the model was not finished training. Learning rate 0.01 is better suited for SGD during 5-15 epochs. A learning rate of 0.0001 might have worked, but the number of epochs would have had to be much larger, perhaps around 100, in order to see that the model finished. However, SGD showed great potential to perform almost as well as Adam when the learning rate was 0.01.

5.7 ROC area between models

The results in Table 2 and 3 all prove that MobileNetV2 pretrained together with Adam worked best. All of the ROC areas were above 0.6, compared to the other models that had ROC areas all the way down to 0.002, for example the MobileNetV2 model without pretrained weights in Table 3. However, if you only have one inhibitor for one kinase, you cannot create a good model. Often you need at least eight to ten substances that are active for the same target to create a model. If there are kinases for which there are only two to three substances that inhibits that kinase, a good model is difficult to achieve. One way to solve this problem would have been to group the kinases into larger groups, as an example including all tyrosine kinases in one group. Then you would be able to see differences between large groups.

Some of the targets with low ROC areas were targets for several compounds. Should that increase the validity of those ROC areas? Were they really harder to predict? Not necessarily. If two substances are found in the validation set, they might still be hard to predict. If the partitioning of the training and test data would change, the results might change as well. Another factor that is important to take into account is the fact that some substances does not change the morphology of the cell, yet another that the input data was imbalanced.

Observing Figure 11 with the ROC curves clearly states that the MobileNetV2 model with Adam performed better that with SGD. Since the lines are close to the top right corner, the result can be somewhat trusted regarding the ROC area values. If the lines would have been centred around the line in the middle, like for the MobileNetV2 model with SGD, it would have indicated that the model was doing some random guessing. It would also mean that there

was a lot of false hits and that there are many kinases that you cannot say anything about. If the lines would have been centred around the bottom right corner, that would have indicated that the low curve was generated by chance. In that case, if there would be curves located close to the top left corner as well you would not know for sure if they were placed there by chance as well.

5.8 Validate the models and data quality

The quality of the data varies. In some plates, a portion of the images were lighter, some were darker, and some were black. The black images indicated that there were not any cells left in the image, which in turn affected the classification power of the models. There was also a difference between the plates. In some plates, the cells grew nicely and on some they did not, which led to less cells. All of these factors affect the end result.

When taking the data quality into account, perhaps the results were better when only one plate was used as an input instead of all 20 plates? Then the diversity of the possible reduction in data quality might be smaller, which might improve the result. However, the results do not strengthen that theory. To be able to trust the produced results, you should use hundreds of substances with the same mechanisms. There are also other insecurities to account for, for example that the substances might not affect only one or five kinases but rather large and broad groups of kinases as well as kinases from several groups. For the data used in this project we know that the inhibitors have target proteins, but they might have many more that we do not know of since that have not yet been tested.

5.9 Two-stream models and choice of input

By comparing the binary one-stream models with Adam and SGD in 4.1 with the two-stream models in 4.2 and 4.3, there is a significant drop in loss and an increase in accuracy for the two-stream models. That is illustrated with RMSprop in 4.2, and for almost all two-stream models in 4.3. The SGD models in 4.3 with learning rate 0.0001 did not perform better but the ones with learning rate 0.01 did. Once again, the reason for this difference is that SGD needs a higher learning rate or more epochs. Nonetheless, a two-stream model outperforms a one stream model according to these results.

5.10 Improvements

Several improvements can be applied to this project. First of all, more parameters could have been tried. As mentioned in part 2.7, the best way to configure parameters like the batch size and the number of epochs is by trying different values and see what works best. Initially, I only saved images of the results and used the old notebook to write a new model. This made the analysis more difficult since I was unable to return and analyse more metrics and parameters of the models and their results without having to run them again. Another problematic factor was that trying different parameters requires time, which was a limited asset in this project.

Every compound was tested in three different wells but in three different concentrations. All of the compounds with the different concentrations should be placed either in the training set or in the test set so that the same compound cannot be found both in the training and test set. From the beginning, the test set contained 20% of all **images**. The problem with that approach is that there are nine images per well, and a few of the images from one well can end up in the

test set but there could still be images from the same well left in the training set. An attempt to fix this problem was made in part 3.6 when the training and test data were partitioned according to the wells so that all samples from one well was assigned to either training or test. In exact, the test data consisted of 20% of the **wells**. However, still the same substance, although at various concentrations, was present in both the training and the test set. It would have been interesting to see how the models would have performed if the same **substance** could not be present in both the training set and the test set. It could be said that the models are "cheating" now that they have seen the compounds before in the training data. With this aspect taken into account, it is fair to say the accuracy of the models presented in this report has an accuracy that might be too good to be true.

Another measure of improvement would have been to use class weights. The dataset was imbalanced in the way that there were fewer active compounds for each target than inactive. As an example, for each target protein there might have been 10 active kinase inhibitors and 368 inactive. By using class weights, this would be accounted for and a more accurate picture of the model's performance could be accumulated. When the dataset is unbalanced, the predictions will always be towards the larger classes. Another way to improve this project would be to use two datasets, a larger dataset if that was available and conformal prediction. Conformal prediction determines the confidence values of newly predicted values (Matiz & Barner 2019). One version of conformal prediction called Inductive conformal prediction could be used to improve the performance of the classifiers through the use of active learning.

Lastly, there is a risk that you might have been lucky with some substance in the test set that was easier or harder to predict. An important measure of improvement would be to use cross-validation to account for that risk. With 5-fold cross-validation you would create five networks and train on five networks, which would generate more results from all of those five networks. This step is absolutely necessary if this project would proceed, but it would have been hard to accomplish during this project. With five GPUs that might have worked, but with the GPUs accessible to this project it would have taken days to accomplish.

5.11 Ethics

The ethical debate revolving around AI and machine learning is huge. Many believe in the power of AI, but few tend to analyse the predictions critically except for scientist's that are working with AI and Machine Learning. With this in mind, it is easy to imagine that this deficit could be abused to create inaccurate results which could be misinterpreted by the public. It is easy to make a model biased, from how the data is gathered or constructed to how the training and test sets are divided. As always when discussing how the data is gathered, it is important that a Henrietta Lacks case is being avoided. Hence, it is important to view AI and Machine Learning as critically from every aspect as one would do with new pharmaceuticals as an example.

6 Conclusion

The scientific question of this project was whether it is possible to classify kinase inhibitors with CNNs and cell painting. The conclusion is that it is possible to some degree. However, all of the relationships between the target proteins and compounds are not known yet, and can therefore not be annotated and analysed. The data quality needs to be improved with some more pre-processing, and the partitioning into training and test needs to consider the different concentrations of the compounds. Lastly, cross validation and class weights needs to be applied to give credibility to the results. As always, other parameters like batch size and learning rate can be fine-tuned to improve the models further.

To summarise which architecture performed the best, pretrained networks with ImageNet weights were superior. They performed better than the simple binary models. The micro average values showed that ResNet50 and MobileNetV2 performed equally good. However, the ROC areas showed that the architecture that performed best out of the pretrained networks was MobileNetV2. This assumption was also strengthened by the ROC curve that mostly showed high true positive rates. The results also showed that using the same learning rate for all models does not necessarily provide a better comparison if the model fails to learn due to having the wrong learning rate. Some important lessons that can be drawn from this are the importance of choosing the right learning rate for each model, and to look at several validation metrics. Another lesson learned is the importance of transfer learning when training deep neural networks. Without them, errors like dying ReLUs might occur. This happened for some of the VGG16 models without pretrained weights.

The fact that the data was split into training and test sets based upon which well they belonged to damage the credibility of the results. The model might have already seen some of the validation data in the training set. Hence, even though the results showed that the model performed well, that does not necessarily prove that the model learned how to classify these kinase families since this can be considered “cheating”. Regarding the choice of optimiser, the conclusion is that the whole dataset and SGD seemed to perform best for the binary model but for the two-stream models Adam optimised better than SGD regarding the micro average, the ROC area and the loss and accuracy plots. However, according to some papers, Adam has been proven to not generalise as well as SGD (Keskar & Socher 2017). Perhaps the results in this report are too good to be true, and SGD performs better than Adam? As discussed in the paragraph above, the results lack credibility, which makes this question highly relevant.

No conclusions could be drawn regarding if any kinase families were easier or harder to classify. To analyse that properly, many more substances with the same mechanisms should be used. Other insecurities to consider when analysing the results are that the substances might not affect only one or five kinases, but rather large and broad groups of kinases as well as kinases from several groups. For the data used in this project, we know that the inhibitors have target proteins. However, they might have many more that we do not know of since that have not been tested yet. It is known for kinase inhibitors in general that they are not selective, they interact with large numbers of kinases. 60-70 of the substances used in this project were tested on all kinases, and was proven to affect broad groups of kinases.

The two different inputs showed a slightly better micro average for input number two where the Phalloidin and WGA dye was used twice. Other than that, no big differences could be observed between the different inputs. Nevertheless, there are other input combinations to try

out that might reveal other patterns. But are the images useful? Yes, but they would be more useful if the differences between very dark and very light images as well as images with few cells would be accounted for. Perhaps, a threshold could be used to exclude images where all cells died. That could improve the quality of the dataset.

7 Acknowledgements

I would like to express my gratitude to my supervisor Maris Lapin's, for your continuous and tireless support throughout the whole project. I would also like to thank my subject reviewer Filip Malmberg for your feedback and guidance. This project would not had been possible if Ola Spjuth had not given me the opportunity, and for that I am grateful. Last but not least, I would like to thank my examiner Pascal Milesi and my coordinator Lena Henriksson for your good advice and support.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mane D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viegas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467 [cs]

Aliper A, Plis S, Artemov A, Ulloa A, Mamoshina P, Zhavoronkov A. 2016. Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data. Molecular Pharmaceutics 13: 2524–2530.

Bini SA. 2018. Artificial Intelligence, Machine Learning, Deep Learning, and Cognitive Computing: What Do These Terms Mean and How Will They Impact Health Care? The Journal of Arthroplasty 33: 2358–2361.

Blume-Jensen P, Hunter T. 2001. Oncogenic kinase signalling. Nature 411: 355–365.

Bray M-A, Singh S, Han H, Davis CT, Borgeson B, Hartland C, Kost-Alimova M, Gustafsdottir SM, Gibson CC, Carpenter AE. 2016. Cell Painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. Nature Protocols 11: 1757–1774.

Campbell N, Reece J, Cain M, Urry L, Wasserman S, Minorsky P, Jackson R. 2014a. Biology: A Global Approach, Global Edition. 10th ed, p. 443. Pearson,

Campbell N, Reece J, Cain M, Urry L, Wasserman S, Minorsky P, Jackson R. 2014b. Biology: A Global Approach, Global Edition. 10th ed, p. 222. Pearson,

Cruz-Roa A, Gilmore H, Basavanhally A, Feldman M, Ganesan S, Shih NNC, Tomaszewski J, González FA, Madabhushi A. 2017. Accurate and reproducible invasive breast cancer detection in whole-slide images: A Deep Learning approach for quantifying tumor extent. Scientific Reports 7: 1–14.

Davis MI, Hunt JP, Herrgard S, Ciceri P, Wodicka LM, Pallares G, Hocker M, Treiber DK, Zarrinkar PP. 2011. Comprehensive analysis of kinase inhibitor selectivity. Nature Biotechnology 29: 1046–1051.

Deng J, Dong W, Socher R, Li L-J, Kai Li, Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255.

Fakoor R, Ladhak F, Nazi A, Huber M. 2013. Using deep learning to enhance cancer diagnosis and classification. 8.

García MA, Gil J, Ventoso I, Guerra S, Domingo E, Rivas C, Esteban M. 2006. Impact of Protein Kinase PKR in Cell Biology: from Antiviral to Antiproliferative Action. Microbiology and Molecular Biology Reviews 70: 1032–1060.

Geirhos R, Rubisch P, Michaelis C, Bethge M, Wichmann FA, Brendel W. 2019. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv:1811.12231 [cs, q-bio, stat]

He K, Zhang X, Ren S, Sun J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs]

IndoML. 2018. Student Notes: Convolutional Neural Networks (CNN) Introduction. WWW document 7 March 2018: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/. Accessed 7 February 2020.

Kandaswamy C, Silva LM, Alexandre LA, Santos JM. 2016. High-Content Analysis of Breast Cancer Using Single-Cell Deep Transfer Learning. Journal of Biomolecular Screening 21: 252–259.
Karaman MW, Herrgard S, Treiber DK, Gallant P, Atteridge CE, Campbell BT, Chan KW, Ciceri P, Davis MI, Edeen PT, Faraoni R, Floyd M, Hunt JP, Lockhart DJ, Milanov ZV, Morrison MJ, Pallares G, Patel HK, Pritchard S, Wodicka LM, Zarrinkar PP. 2008. A quantitative analysis of kinase inhibitor selectivity. Nature Biotechnology 26: 127–132.
Kensert A, Harrison PJ, Spjuth O. 2019. Transfer Learning with Deep Convolutional Neural Networks for Classifying Cellular Morphological Changes. SLAS DISCOVERY: Advancing the Science of Drug Discovery 24: 466–475.
Keras Documentation. About Keras models - Keras Documentation. WWW document: https://keras.io/models/about-keras-models/. Accessed 1 April 2020a.
Keras Documentation. Image Preprocessing - ImageDataGenerator class. WWW document: https://keras.io/preprocessing/image/. Accessed 9 March 2020b.
Keskar NS, Socher R. 2017. Improving Generalization Performance by Switching from Adam to SGD. arXiv:1712.07628 [cs, math]
Ketkar N. 2017. Introduction to Keras. In: Ketkar N (ed.). Deep Learning with Python: A Hands-on Introduction, pp. 97–111. Apress, Berkeley, CA.
Kingma DP, Ba J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs]
Kraus OZ, Ba JL, Frey BJ. 2016. Classifying and segmenting microscopy images with deep multiple instance learning. Bioinformatics 32: i52–i59.
Kraus OZ, Grys BT, Ba J, Chong Y, Frey BJ, Boone C, Andrews BJ. 2017. Automated analysis of high-content microscopy data with deep learning. Molecular Systems Biology 13: 924.
Krizhevsky A, Sutskever I, Hinton GE. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (ed.). Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc.,
LeCun Y. MNIST Demos on Yann LeCun's website. WWW document: http://yann.lecun.com/exdb/lenet/. Accessed 26 May 2020.
LeCun Y, Bengio Y, Hinton G. 2015. Deep learning. Nature 521: 436–444.
Lodish HF, Berk A, Kaiser CA, Krieger M, Scott MP, Bretscher A, Ploegh H, Matsudaira P. 2008. Molecular cell biology. Molecular cell biology, 6th ed, p. 91. W.H. Freeman, New York.
Moen E, Bannon D, Kudo T, Graf W, Covert M, Valen DV. 2019. Deep learning for cellular image analysis. Nature Methods 16: 1233–1246.
Nandy A, Biswas M. 2018. Reinforcement Learning with Keras, TensorFlow, and ChainerRL. In: Nandy A, Biswas M (ed.). Reinforcement Learning : With Open AI, TensorFlow and Keras Using Python, pp. 129–153. Apress, Berkeley, CA.
Niforou KN, Anagnostopoulos AK, Vougas K, Kittas C, Gorgoulis VG, Tsangaris GT. 2008. The Proteome Profile of the Human Osteosarcoma U2OS Cell Line. Cancer Genomics - Proteomics 5: 63–77.
Nikon's MicroscopyU. Galleries | Human Bone Osteosarcoma Epithelial Cells (U2OS Line). WWW document: https://www.microscopyu.com/gallery-images/human-bone-osteosarcoma-epithelial-cells-u2os-line. Accessed 4 February 2020.
Nitta N, Sugimura T, Isozaki A, Mikami H, Hiraki K, Sakuma S, Iino T, Arai F, Endo T, Fujiwaki Y, Fukuzawa H, Hase M, Hayakawa T, Hiramatsu K, Hoshino Y, Inaba M,

Ito T, Karakawa H, Kasai Y, Koizumi K, Lee S, Lei C, Li M, Maeno T, Matsusaka S, Murakami D, Nakagawa A, Oguchi Y, Oikawa M, Ota T, Shiba K, Shintaku H, Shirasaki Y, Suga K, Suzuki Y, Suzuki N, Tanaka Y, Tezuka H, Toyokawa C, Yalikun Y, Yamada M, Yamagishi M, Yamano T, Yasumoto A, Yatomi Y, Yazawa M, Di Carlo D, Hosokawa Y, Uemura S, Ozeki Y, Goda K. 2018. Intelligent Image-Activated Cell Sorting. Cell 175: 266-276.e13.

NumPy. NumPy. WWW document: https://numpy.org/. Accessed 1 April 2020.

pandas-dev. 2020. pandas-dev/pandas. pandas

Pärnamaa T, Parts L. 2017. Accurate Classification of Protein Subcellular Localization from High-Throughput Microscopy Images Using Deep Learning. G3: Genes, Genomes, Genetics 7: 1385–1392.

Pawlowski N, Caicedo JC, Singh S, Carpenter AE, Storkey A. 2016. Automating Morphological Profiling with Generic Deep Convolutional Networks. bioRxiv 085118.

Ramón-Maiques S, Marina A, Gil-Ortiz F, Fita I, Rubio V. 2002. Structure of Acetylglutamate Kinase, a Key Enzyme for Arginine Biosynthesis and a Prototype for the Amino Acid Kinase Enzyme Family, during Catalysis. Structure 10: 329–342.

Robbins H, Monro S. 1951. A Stochastic Approximation Method. The Annals of Mathematical Statistics 22: 400–407.

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. pp. 4510–4520.

Scheeder C, Heigwer F, Boutros M. 2018. Machine learning and image-based profiling in drug discovery. Current Opinion in Systems Biology 10: 43–52.

Selleck Kinase Inhibitor Library. PI3K-Akt Signaling Pathways. WWW document: https://www.selleckchem.com/pharmacological_PI3K_Akt_mTOR.html. Accessed 26 May 2020.

Simm J, Klambauer G, Arany A, Steijaert M, Wegner JK, Gustin E, Chupakhin V, Chong YT, Vialard J, Buijnsters P, Velter I, Vapirev A, Singh S, Carpenter AE, Wuyts R, Hochreiter S, Moreau Y, Ceulemans H. 2018. Repurposing High-Throughput Image Assays Enables Biological Activity Prediction for Drug Discovery. Cell Chemical Biology 25: 611-618.e3.

Simonyan K, Zisserman A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs]

Sommer C, Hoefler R, Samwer M, Gerlich DW. 2017. A deep learning and novelty detection framework for rapid phenotyping in high-content screening. Molecular Biology of the Cell 28: 3428–3436.

Stanford Vision Lab, Stanford University, Princeton University. ImageNet. WWW document: http://image-net.org/about-overview. Accessed 11 May 2020.

Sullivan DP, Winsnes CF, Åkesson L, Hjelmare M, Wiking M, Schutten R, Campbell L, Leifsson H, Rhodes S, Nordgren A, Smith K, Revaz B, Finnbogason B, Szantner A, Lundberg E. 2018. Deep learning is combined with massive-scale citizen science to improve large-scale image classification. Nature Biotechnology 36: 820–828.

Vijayabhaskar J. 2019. Tutorial on Keras flow_from_dataframe. WWW document 27 April 2019: https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c. Accessed 27 May 2020.

Vijayabhaskar J. 2020. Multi-label image classification Tutorial with Keras ImageDataGenerator. WWW document 1 March 2020:

https://medium.com/@vijayabhaskar96/multi-label-image-classification-tutorial-with-keras-imagedatagenerator-cd541f8eaf24. Accessed 27 May 2020.
Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E. 2018. Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience, doi https://doi.org/10.1155/2018/7068349.
Wang D, Khosla A, Gargeya R, Irshad H, Beck AH. 2016. Deep Learning for Identifying Metastatic Breast Cancer. arXiv:1606.05718 [cs, q-bio]
Yamashita R, Nishio M, Do RKG, Togashi K. 2018. Convolutional neural networks: an overview and application in radiology. Insights into Imaging 9: 611–629.
Zhang W, Li R, Zeng T, Sun Q, Kumar S, Ye J, Ji S. 2016. Deep Model Based Transfer and Multi-Task Learning for Biological Image Analysis. IEEE Transactions on Big Data 1–1.
Zhavoronkov A, Ivanenkov YA, Aliper A, Veselov MS, Aladinskiy VA, Aladinskaya AV, Terentiev VA, Polykovskiy DA, Kuznetsov MD, Asadulaev A, Volkov Y, Zholus A, Shayakhmetov RR, Zhebrak A, Minaeva LI, Zagribelnyy BA, Lee LH, Soll R, Madge D, Xing L, Guo T, Aspuru-Guzik A. 2019. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nature Biotechnology 37: 1038–1040.

Appendix A

The manually annotated metadata file described in 3.1. The csv-file is attached to this report.