UPPSALA
UNIVERSITET

UPTEC X 22002
Examensarbete 30 hp
Mars 2022

Analyzing Cell Painting images using different CNNs and Conformal Prediction variations

Optimization of a Deep Learning model to predict the MoA of different drugs

Anna Hillver

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

Analyzing Cell Painting images using different CNNs and Conformal Prediction variations

Anna Hillver

Microscopy imaging based techniques, such as the Cell Painting assay, could be used to generate images that visualize the Mechanism of Action (MoA) of a drug, which could be of great use in drug development. In order to extract information and predict the MoA of a new compound from these images we need powerful image analysis tools. The purpose with this project is to further develop a Deep Learning model to predict the MoA of different drugs from Cell Painting images using Convolutional Neural Networks (CNNs) and Conformal Prediction. The specific task was to compare the accuracy of different CNN architectures and to compare the efficiency of different nonconformity functions.

During the project the CNN architectures ResNet50, ResNet101 and DenseNet121 were compared as well as the nonconformity functions Inverse Probability, Margin and a combination of them both. No significant difference in accuracy between the CNNs and no difference in efficiency between the nonconformity functions was measured. The results showed that the model could predict the MoA of a compound with high accuracy when all compounds were used both in training, validation and test of the model, which validates the implementations. However, it is desirable for the model to be able to predict the MoA of a new compound if the model has been trained on other compounds with the same MoA. This could not be confirmed through this project and the model needs to be further investigated and tested with another dataset in order to be used for that purpose.

Handledare: Ebba Bergman
Ämnesgranskare: Ida-Maria Sintorn
Examinator: Pascal Milesi
ISSN: 1401-2138, UPTEC X 22002

Populärvetenskaplig sammanfattning

Inom läkemedelsframtagning och forskning så är det viktigt att förstå och analysera verkningsmekanismen hos ett specifikt läkemedlet, vilket innebär den effekt som läkemedlet har i en cell. Med hjälp av mikroskopibilder av celler som har utsatts för ett läkemedel så kan dessa effekter visualiseras och analyseras och på så sätt ge information om hur ett läkemedel fungerar. För att effektivisera analysprocessen av dessa bilder behövs program som automatiskt tar fram information ur bilderna och kan ge användaren information om vilken verkningsmekanism som förekommer hos läkemedlet. Detta kan göras med hjälp av djupinlärning, vilket är en variant av Artificiell Intelligens som automatiskt hämtar information från data och lär sig skilja på olika typer av klasser inom datan som den analyserar.

Detta projekt går ut på att vidareutveckla ett program för att analysera mikrosopibilder av celler som utsatts för olika läkemedel för att kunna prediktera dess verkningsmekanism. Det dataset som används i detta projekt innehåller bilder på celler som utsatts för olika läkemedel och till varje bild finns även information om vilken verkningsmekanism just det läkemelet har vilket programmet använder som facit när den försöker hitta skillnader mellan de olika verkningsmekanismerna. En del av bilderna används för att träna algoritmen, d.v.s. algoritmen använder dessa bilder för att hitta vad som utmärker de olika verkningsmekanismerna. En del av bilderna hålls utanför själva träningen för att sedan kunna användas för att testa hur väl programmet presterar när nya bilder introduceras. Utöver själva djupinlärningsmodellen användes dessutom en metod som kallas för Conformal Prediction. Den metoden går ut på att per test-exempel prediktera vilken eller vilka klasser exemplet hör till med en viss noggrannhet. Ibland kan det till exempel vara svårt för en djupinlärningsmodell att avgöra mellan två olika klasser. En vanlig modell hade då ändå gett endast en klass som prediktion, men med hjälp av Conformal Prediction kan man istället få svaret ”med 80 % säkerhet tillhör objektet klass A eller B”. Detta ger då en extra nivå av säkerhet till resultatet och kan vara av stor nytta för att inte utesluta vissa klasser.

Ursprungligen testades programmet genom att alla bilder på celler som utsatts för vissa läkemedel hölls utanför under träningen och användes exklusivt för testning. Detta gav resultatet att programmet i sig uppnådde en hög noggrannhet i träningen, men den hade svårt för att koppla nya läkemedel till de redan introducerade verkningsmekanismerna. För att sedan bekräfta att de ändringar och tillägg som gjorts under projektet var korrekta, testades även en annan uppdelning av bilderna där alla läkemedel var med i både träning och test av programmet. Detta gav en hög nogrannhet och kunde verifiera att själva funktionerna i programmet är giltiga.

Slutsatsen av detta är att det program som utvecklats kan analysera och förutspå verkningsmekanismer hos läkemedel utifrån bilder, men att det utifrån detta dataset är svårt att koppla ihop olika läkemdel med samma verkningsmekanism, vilket gör det svårt att applicera på nya läkemedel där verkningsmekanismen är okänd.

Table of contents

Abbreviations

AI	Artificial Intelligence
avgC	Average number of predicted labels per prediction set
CNN	Convolutional Neural Network
CP	Conformal Prediction
DL	Deep Learning
ICP	Inductive Conformal Prediction
IP	Inverse Probability
IP_M	Combination of Inverse Probability and Margin
M	Margin
MICP	Mondrian Inductive Conformal Prediction
ML	Machine Learning
MoA	Mechanism of Action
oneC	Fraction of singleton predictions

1 Introduction

The Mechanism of Action (MoA) is the biochemical process in which a drug works within a cell and through which the drug causes a pharmacological effect (Salters-Pedneault 2020). The understanding of a drug's MoA is important when discovering new drugs as well as in drug repurposing which makes it an important field to study and requires powerful tools to generate correct information.

It is possible to visualize the MoA of a drug through microscopy imaging of the cells using fluorescent dyes by following a phenotypic profiling assay such as Cell Painting (Bray *et al.* 2016). Since these types of experiments can generate a large amount of images it is also desirable to have powerful and effective tools to analyze them with high accuracy.

Deep learning is a powerful tool when it comes to extracting features and make predictions from a large amount of data and convolutional neural networks (CNN) in particular can automatically learn features from images (Alzubaidi *et al.* 2021). In combination with deep learning (DL), you can also apply Conformal Prediction (CP), which is an algorithm that can asses uncertainty in the predictions (Alvarsson *et al.* 2021).

1.1 Previous work

This project builds on the previous work of Ebba Bergman at the Department of Pharmaceutical Biosciences at Uppsala University. She had built a program that uses a CNN to analyse the MoA on the same dataset (BBBC012) and had implemented CP. The program also provided scripts to make k-folds and sort the files into folders before use. The initial premise of the project was that the original code worked correctly and that the BBBC021 dataset was well suited for this task. During the project, some errors were found in the original code that changed the premise of the project.

1.2 Aim and purpose of the project

The task of this project was to further develop this program and try to optimize it by testing different types of CNNs and variations of conformal prediction. The goal of the project was to validate CP as a method applied to different CNNs trained on a dataset containing microscopy images of cells, and also to measure differences in efficiency

using different nonconformity functions. The purpose of this is to develop tools to perform better data analysis of cell painting data, which can be used to analyze the MoA of a specific drug when a cell is exposed to it. The product of the project will be tools including CNNs and CP that can be used for this purpose. In the long term, the aim of this project is to find a powerful tool to use in drug development and to find information on how drugs could be repurposed. For the program to be useful for this purpose, it has to be able to predict the MoA of a new potential drug compound that has not been used in the training of the model, if the model has been trained on other compounds with the same MoA.

2 Background

2.1 Cell Painting

The dataset that will be used in this project has been produced using a Cell Painting assay, which is a specific type of morphological profiling assay. Cell Painting is a high-content image-based assay that uses fluorescent dyes to identify biological information (Bray *et al.* 2016). This means that by using the Cell Painting assay, you can produce images of cells that has been stained with fluorescent dyes to visualize the cellular morphology. This enables a wide range of applications, for example to visualize the morphological changes in a cell when it is exposed to a drug. The advantage of using Cell Painting instead of other screening assays is that Cell Painting enables us to measure a very large number of features instead of focusing on a few features selected based on known biological relevance (Bray *et al.* 2016).

2.2 Mechanism of Action

One application that Cell painting can be used for is to analyze the mechanism of action (MoA) of different drugs. The MoA of a drug is the specific biochemical process that a drug causes within the cell (Salters-Pedneault 2020). It is through its MoA that a drug has an effect.

2.3 Deep Learning

Deep Learning (DL) is a subset of the field Machine learning (ML), which in turn is a subset of Artificial Intelligence (AI), see Figure 1 for visualization of the relation. AI is the overall concept of a program that is able mimic the intelligence of humans which means that it is able to reason and adapt to a situation (Alzubaidi *et al.* 2021). ML on the other hand, is a kind of AI, where the algorithm is able to adapt and improve as it is exposed to more training data (Alzubaidi *et al.* 2021).

DL is a framework based on multilayered neural networks which learn features from a large amount of data (Alzubaidi *et al.* 2021). This means that DL learns from the data and does not need the human input in form of rules to learn. DL is therefor a good solution to problems where the human expertise is not enough for example due to the size of the problem. Although all DL algorithms consists of multilayered neural networks, there are a lot of different models designed for different purposes (Alzubaidi *et al.* 2021). One group of neural network models are Convolutional Neural Networks (CNNs), which is the kind of DL models that will be used in this project.

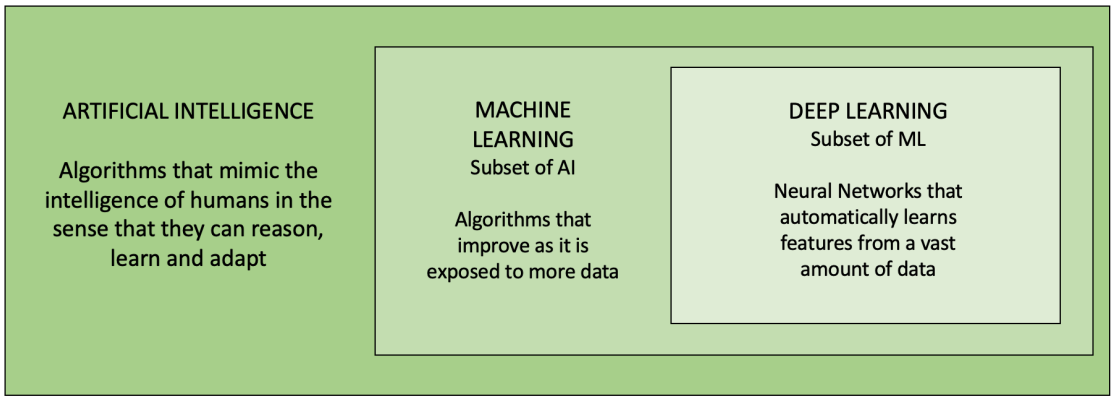


Figure 1: Visualization of the differences and connection between Artificial Intelligence, Machine Learning and Deep Learning

2.3.1 Supervised and unsupervised learning

When talking about ML and DL it is usual to divide the techniques into two different types of algorithms: supervised and unsupervised learning. The difference between the two is whether the algorithm is provided with labeled data or not. In supervised learning, the algorithm is trained on labeled data and it is able to adapt and improve the estimate according to the true class labels (Alzubaidi *et al.* 2021). On the other hand in unsupervised learning, the DL algorithm is trained on data without labels. In these

cases, the algorithm learns to identify relationships and structures in the data (Alzubaidi *et al.* 2021). In this project supervised learning will be used.

2.3.2 Convolutional Neural Networks

Because of the large amount of data Cell Painting experiments produce and the large amount of features that can be extracted from the images, we need powerful tools to analyze them. CNNs is a type of DL algorithm that is the gold standard in image analysis with DL since it automatically learns features from images (Hofmarcher *et al.* 2019).

A CNN model consists of an input layer, in this case the cell painting images, an output layer containing the prediction, and in between a number of hidden layers (Alzubaidi *et al.* 2021). The significance of CNNs is that among the hidden layers there are convolution layers that perform convolutional operations on the input (Alzubaidi *et al.* 2021). The input images can be described as a n x n dimensional matrix of pixel-values with one channel if it is a gray-scale image and three channels if it is a RGB-image. If we for example have an input image with the dimensions 6 x 6 with pixel values at each position and a convolution layer, also called kernel, with the dimension 3x3 with random weights at each position. The kernel will then slide over the input image both horizontally and vertically and at each position calculating the dot product between the input and the filter and then added together it gives the result for the next position in the output of that layer (Alzubaidi *et al.* 2021). This is repeated until no more slides are possible. There are a lot of different CNN models, where the architecture vary a bit between them, and the models that will be used in this project are ResNet50, ResNet101 and DenseNet121.

2.3.3 Epochs and Batch size

When training the networks, the training data is passed through the network, first with a forward pass (called forward propagation) and then with a backward pass (back propagation) to adjust the weights in the hidden layers to fit the data (Alzubaidi *et al.* 2021). An epochs means one pass of the entire training data through the network including both forward and back propagation (Brownlee 2018). The user then decides how many times the entire training dataset should be passed through the network, i.e. how many epochs. If you use all the training data at the same time one epochs require just one iteration, but you can also divide the training data into smaller batches (Brownlee 2018). This means that if you have 1000 data points and you set the batch size to 1000, one epoch will require 1 iteration, but if you set the batch size to 10, you require 100 iterations per epoch.

2.3.4 Transfer learning

Transfer Learning is used to take a model that has been trained for one task and repurpose it for another task (Brownlee 2017). So for example, you can take a CNN that has been trained on one dataset with one type of images and then use the information that this model has learned as a start when training for a new task. In this project, the CNN models were implemented using the Keras applications framework in combination with the TensorFlow backend framework. This extension offers building blocks of neural networks and the user can choose if you want the network to be pre-trained or not, and in this case pre-trained models were used. The models had then been pretrained on ImageNet which is an image database containing over 14 million images of different objects divided into over 20000 categories (Deng *et al.* 2009). When using the pretrained networks, you include the weights from the pretraining when applying your model to your new task, and in that way apply transfer learning. It is also common in transfer learning to freeze layers during a part of the training. This means that you can freeze the layers imported from the pretrained network, add trainable layers that will convert the features into prediction on the current dataset, and then unfreeze the layers to fine-tune them to the new data.

2.4 Conformal Prediction

In this project, CP will be used as a complement to the CNNs, in order to determine the confidence in the predictions. CP is a method that you can apply on top of a machine learning model, which calculates prediction regions for each predicted object (Shafer & Vovk 2007). If you work with a classification problem the prediction region will be a set of labels and in the case of regression problems the prediction region will be an interval (Alvarsson *et al.* 2021). In this project, classification is used which means that prediction sets will be produced. The true label should be in the predicted region with a probability of a confidence set by the user. For example in the case of a binary classification problem with the classes Cat and Dog the possible outcomes is as followed:

1. {cat}
2. {dog}
3. {dog, cat}
4. {}

If the outcome is both classes, it means that the model is unable to distinguish between the classes at the specified confidence level. The alternative ”empty” means that the model can't connect the predicted compound to any of the classes at the specified confidence level. In the case of non-binary classification, where the classes included in the problem is more than two, there can be more than these four outcomes.

To decide which classes should be included in the prediction set for a specific object, the conformal predictor calculates a nonconformity score (α-value) and rank it against examples from the training data and then calculates a *p*-value for each class. If the *p*-value is greater than the significance level (1-confidence) the class will be included in the prediction set. (Alvarsson *et al.* 2021)

Inductive conformal prediction (ICP) is the most common approach for CP and it divides the training set into a proper training set and a calibration set. In Mondrian Inductive Conformal Prediction (MICP), the predictor uses class dependent calibration sets, i.e. calculates alpha values for each class separately before ranking it against the training data (Alvarsson *et al.* 2021). In this project, MICP will be used and the procedure is described below (Johansson *et al.* 2017).

Training procedure:

1. Divide the training set (Z) into a proper training set (Z^t) and a calibration set (Z^c)
2. Use the training set and apply the learning algorithm H (in the case of this project the CNN), which will generate the underlying model *h*
3. Use the chosen nonconformity function to calculate the nonconformity for each example in the calibration set, which generates a list of α-values.

For each test object $(\mathbf{x}_{k+1})$, the prediction region is generated using the following steps:

1. Specify a significance level $\epsilon \in$ (0,1)
2. Generate the predictions *h(*$\boldsymbol{x}_{k+1}$*)* where *h* is the underlying model and $(\mathbf{x}_{k+1})$ is the test example
3. Assign one of the labels $\tilde{y} \in$ Y, where Y is the set of all possible labels, as the output label for the test example and calculate the nonconformity score for that pattern by applying the chosen nonconformity function

4. Calculate a *p*-value for that pattern using the formula

$$p_{k+1}^{\tilde{y}} = \frac{\left|\left\{z_i \in Z^c : \alpha_i \geq \alpha_{k+1}^{\tilde{y}}\right\}\right| + 1}{q+1}$$

where $p_{k+1}^{\tilde{y}}$ is the *p*-value for the specific pattern of test example and possible label, $z_i \in Z^c$ is the set of examples in the calibration set that belong to the possible label, α_i is the α-values corresponding to z_i, $\alpha_{k+1}^{\tilde{y}}$ is the nonconformity score for the test pattern and *q* is the total number of α-values for that class.

5. Check if $p_{k+1}^{\tilde{y}} > \epsilon$, and in that case include it in the prediction set Γ_{k+1}^{ϵ}, otherwise reject it.

6. Repeat step 3-5 for all possible labels. $\tilde{y} \in \mathrm{Y}$

2.4.1 Nonconformity measure

When working with CP you implement a nonconformity function to calculate a nonconformity score, which is a measurement of how dissimilar a new object is compared to the object the model has been trained on (Alvarsson *et al.* 2021). This can be done in some different ways by implementing different nonconformity functions (Alvarsson *et al.* 2021). The choice of nonconformity function can affect the efficiency of the predictor.

2.4.2 Efficiency of a conformal predictor

In a classification problem, two metrics are often used to measure the efficiency of a conformal predictor. The first one is *oneC*, which measures the fraction of singleton predictions produced by the conformal predictor (Johansson *et al.* 2017). The other metric often used is *avgC*, which measures the average number of predicted labels in each prediction set produced by the conformal predictor (Johansson *et al.* 2017). The goal is to produce a low *avgC* and on the same time achieve a high *oneC* (Aleksandrova & Chertov 2021).

3 Material and Method

3.1 Computational setup and resources

During the project, I was provided with a computer from the Pharamceutical Bioinformatics research group at the Department of Pharamceutical Biosciences at Uppsala University and a pod on their cluster with one GPU where I could run the scripts. The project is written in Python using Visual Studio Code (VSC) as Integrated Development Environment (IDE).

In this project GitHub was used for version control. The existing code was already collected in a GitHub repository which made it easy to clone the original work. Branches were then used for the new implementations.

3.2 Dataset

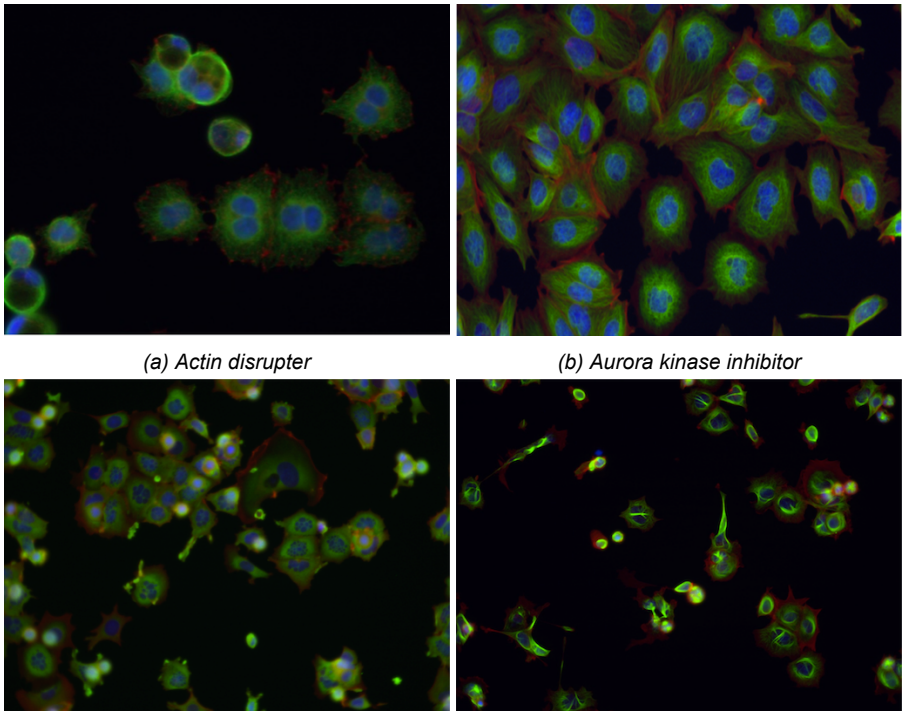
The specific dataset that was used in this project is a subset of the image set BBBC021v1 (Caie *et al.* 2010), available from the Broad Bioimage Benchmark Collection (Ljosa *et al.* 2012). This dataset was produced to be used for testing image-based profiling methods. They used MCF-7 breast cancer cells that they treated with a set of cytotoxic compounds at different concentrations. To image the cells, the wells had been fixed and labeled for B-tubulin, F-actin and DNA and then they used fluorescent microscopy. This can then can be used to train and test image-based methods to predict the MoA of the specific drugs. Figure 2 shows some examples of images from the dataset.

3.2.1 Metadata

With the dataset comes a file containing the metadata of each image. For each image, the metadata file provides information about image number, compound, concentration, MoA, plate, well and replicate.

3.2.2 Preprocessing of data

Before starting this project, the dataset had been preprocessed by Kensert *et. al.* (2019). The dataset originally contains 13200 images from 55 microtiter plates. Each plate contains 60 wells and 4 fields of view per well. Each well contained samples of the cells that



(a) Actin disrupter *(b) Aurora kinase inhibitor*

(c) Tubulin destabilizer *(d) Tubulin stabilizer*

Figure 2: Example images from the BBBC021v1 dataset (Caie et al. 2010), available from the Broad Bioimage Benchmark Collection (Ljosa et al. 2012).

each had been treated with a compound for 24 hours. This means that were imaged each well had been treated with one compound and at one concentration. From the 13200 images in the original dataset, they extracted 1208 images labeled with 12 different MoAs from 38 compounds at a total of 103 different compound-concentration treatments. After normalization and transformation, each image were then cropped into 4 images to increase the number of samples. This resulted in 4832 images in total. (Kensert *et al.* 2019)

3.2.3 Train, validation and test

For this setup, the data has to be divided into three different sets: train, validation/calibration and test. The training dataset is used to train the model while the validation dataset is used to evaluate how well the model performs on the training data. Since we use conformal prediction, the validation dataset is also used as the calibration set to generate the list of α-values (see section 2.4). Then the test set is used as an independent validation of the final model. Since this project needed a training set, a validation/calibration set and a test set, we needed enough data to be sure that all sets

contained images with all MoAs. In order to achieve that, we only included the MoAs that had 3 or more compound resulting in 10 MoAs. This resulted in 3938 images in total. The 10 MoAs, the number of images used per MoA and the number of compounds per MoA is presented in Table 1.

Table 1: The different MoAs represented in the dataset and number of images used per MoA.

Class Number	Mechanism of Action	Images	Compounds
0	Actin Disruptors	239	3
1	Aurora kinase inhibitors	576	3
2	DNA damage	432	4
3	DNA replication	365	4
4	Epithelial	347	3
5	Kinase inhibitors	159	3
6	Microtubule destabilizer	672	4
7	Microtubule stabilizer	430	3
8	Protein degradation	334	4
9	Protein synthesis	384	3

3.2.4 K-folds and file sorting

Since this dataset contains a limited sample of images, K-fold cross validation was used to evaluate the model and to get as much use of the data as possible. K-fold cross validation means that you divide the data into K different subsets, and then hold one of the subsets out during training to be used for testing or validation. Then this is repeated K times to make sure all subsets are held out once. 3-fold cross validation was used in this project, meaning that the data was first divided into 3 subsets. One of the subsets was used as test set while the other two were merged and then divided into train and validation sets. Figure 3 shows a visualization of the dataflow into K-folds and then train, validation and test sets. During the project, three different ways of dividing the the images into K-folds were used. The different partitions will be described below.

3.2.5 K-fold partition 1

The data was divided into 3 groups, making sure all MoAs were represented in all groups. To make sure all MoAs were represented in all groups, one compound per MoA was assigned to one group. Since some of the MoAs had 4 compounds one of the groups contained 2 compounds for some of the MoAs. See Table 1 for number of compounds per MoA. Then one of the group were held out while training the model on the remaining

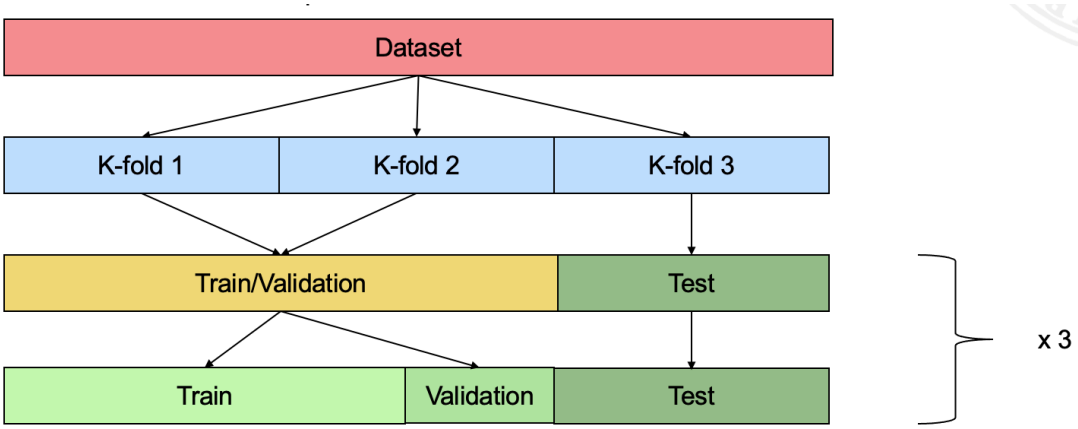


Figure 3: Visualization of the dataflow into K-folds and then train, validation and test sets

two groups. The group that was held out was then used to test the model. Then this was repeated 3 times, so each group had been held out once. As previously mentioned, for each fold one of the groups were held out as a test set. From the remaining samples, 20 % of was used as a validation set and the remaining for training. The reason for separating the compounds in one k-fold each is because we want to investigate the programs ability to analyze and predict the MoA of a new compound that has not been seen during training.

This partition was given by the original code. Unfortunately an error was found resulting in duplicated images that existed in more than than one K-fold which invalidated the results. This error will be further described in section 5.2.1.

3.2.6 K-fold partition 2

The second K-fold partition was identical as partition one, except that the error resulting in duplicated images was removed. The results from this K-fold partition was used to validate the model scientifically.

3.2.7 K-fold partition 3

To validate that the implementations were made correctly, a third K-fold partition was tested. Instead of keeping all images of a specific compound in the same k-fold, the images of the same compound was divided equally between all three k-folds. This was to make sure all compounds were represented in all k-folds and therefore would be represented during both training, validation and testing. This would validate the implemen-

tations since we assume that the model can predict the MoA with high accuracy if it has been trained on images with the same compound.

3.3 CNN

When starting this project, the program was only able to use one specific CNN-model which was ResNet50, but prepped to include more CNNs. The program imported the model from Tensorflow - Keras, which provides a model that is pretrained on ImageNet. The first step was to test this model and make sure that everything works before making further implementations.

The second step was to edit the code so that the user can easily specify which CNN model they want to use. This can be done in a file called ”config.py” where the user defines different parameters of the program. Then a second model was implemented, which was ResNet101, which is of the same type as ResNet50 but deeper, i.e. more layers. The ResNet101 model was also imported from Tensorflow-Keras and pretrained on ImageNet.

At last the CNN model DenseNet121 was implemented and like the previous models it was imported from Tensorflow-Keras and pretrained on ImageNet.

The choice of CNN models to implement was based on what was available in the Tensorflow-Keras framework and that had been shown in literature to perform well on similar problems (Hofmarcher *et al.* 2019). ResNet101 was chosen to compare with ResNet50 if the depth of the network had any impact on the performance. The three different models were tested on the dataset and compared to each other regarding their accuracy of the predictions.

3.3.1 Hyperparameters

The hyperparameters used for training the CNNs are presented in Table 2. These parameters were already set in the original code and tested to work well with this setup and therefor no changes were made to the hyperparameters.

Table 2: Hyperparameters used for training the CNNs

Epochs frozen	10
Epochs unfrozen	30
Batch Size	24

3.4 Nonconformity functions

When working with CP you implement a nonconformity function to calculate a nonconformity measure, which is a measurement of how dissimilar a new object is compared to the object the model has been trained on. This can be done in some different ways by implementing different nonconformity functions (Alvarsson *et al.* 2021). The choice of nonconformity function can affect the efficiency of the predictor, and in the second part of this project some different non-conformity functions was implemented and evaluated. The nonconformity functions were implemented in the way that the user can define which function to use in the config.py file.

3.4.1 Inverse probability

The first nonconformity function that was used is called Inverse Probability (IP), or hinge, and considers only the probability estimate for the correct class label y, according to equation 3.4.1 (Johansson *et al.* 2017). According to literature, Inverse Probability generally results in lower *avgC* than the Margin nonconformity function (Aleksandrova & Chertov 2021). This nonconformity function was already implemented in he program.

$$\Delta[h(\boldsymbol{x}_i),\ y_i] = 1 - \hat{P}_h(y_i|\boldsymbol{x}_i)$$

3.4.2 Margin

The second nonconformity function that was implemented is called Margin (M). This measure takes two probability estimates into account, i.e. the true class label and the incorrect class label with the highest probability estimate, see equation 3.4.2 (Johansson *et al.* 2017). According to literature, Margin generally results in higher *oneC* than the IP nonconformity function (Aleksandrova & Chertov 2021).

$$\Delta[h(\boldsymbol{x}_i),\ y_i] = \max_{y \neq y_i} \hat{P}_h(y|\boldsymbol{x}_i) - \hat{P}_h(y_i|\boldsymbol{x}_i)$$

3.4.3 Combination of Inverse Probability and Margin

When both Inverse Probability and Margin nonconformity functions were implemented and tested, the possibility to use a combination of the two were implemented according to a protocol described in ”Impact of Model-Agnostic Nonconformity Functions on Efficiency of Conformal Classifiers: an Extensive Study” by Alexandrova & Chertov (2021). This combination uses Inverse probability as the baseline, but extended with singleton predictions produced by Margin to improve the efficiency. The user sets a significance level ϵ, which is used for the conformal predictor with inverse probability. Then the significance level for the conformal predictor that uses *margin* is set to $\epsilon/2$. For every instance, both conformal predictors is used to produce predictions. If margin produces a singleton prediction and IP does not, the prediction from margin is used. Otherwise the prediction produced by IP is used.

In the worst case the performance of the combination is the same as for the conformal predictor with Inverse Probability. Hopefully some non-singleton predictions will be replaced by singletons produced by the conformal predictor with Margin and therefore avgC and oneC will be improved. This means that IP alone is almost never the best choice of nonconformity function. IP_M is an improvement of IP and M can in some cases be the best choice (Aleksandrova & Chertov 2021).

3.5 Troubleshooting

While working with the implementations, continuously controls were made to validate the results and to confirm that the program worked as it was supposed to. The built in debugging function in Visual Studio Code was used to find errors in the code. When running the scripts, a number of files were produced, containing statistics and other parameters and plots which was continuously checked to validate the models.

4 Results

4.1 Comparing different CNNs

4.1.1 Confusion matrix

For each CNN that was used, confusion matrices were produced to visualize the accuracy per class. The confusion matrices produced by ResNet50, ResNet101 and DenseNet121 are presented in Figure 4. These were produced per well, meaning that the average accuracy per well is used for calculating the accuracy per class.

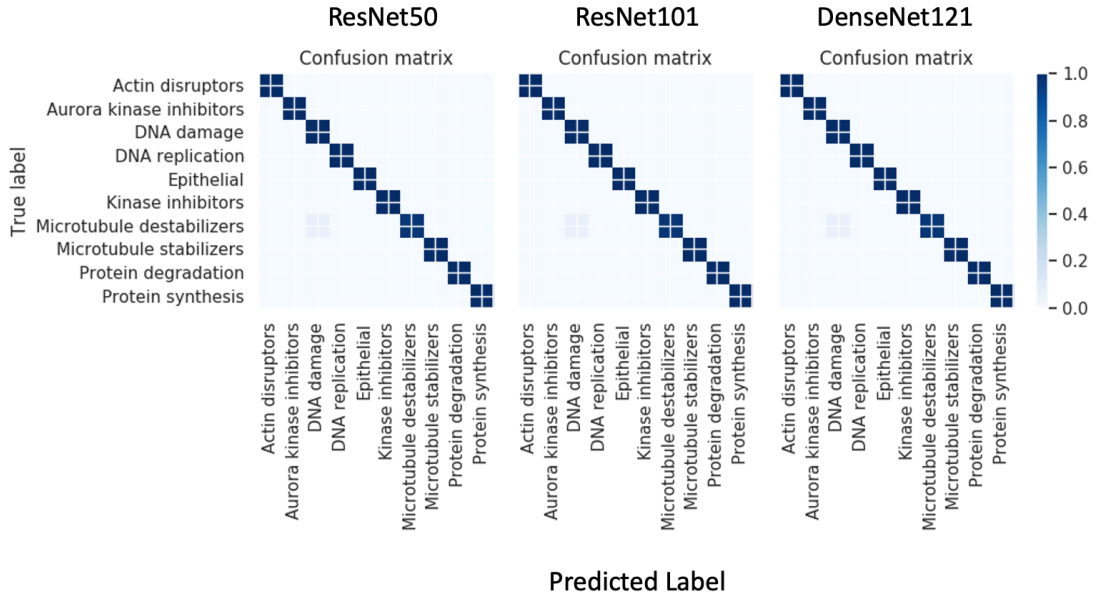
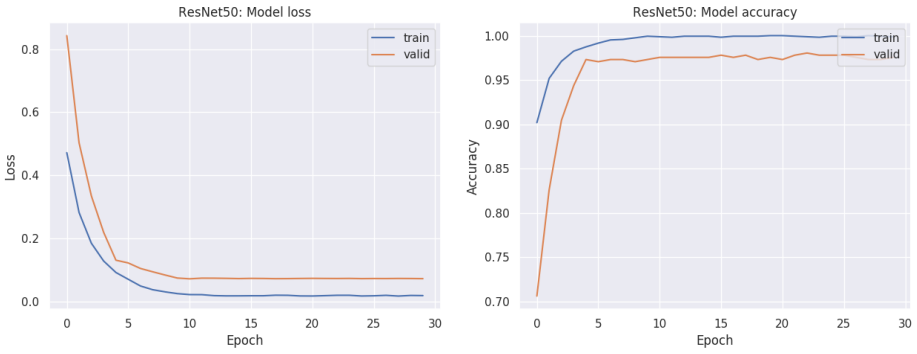


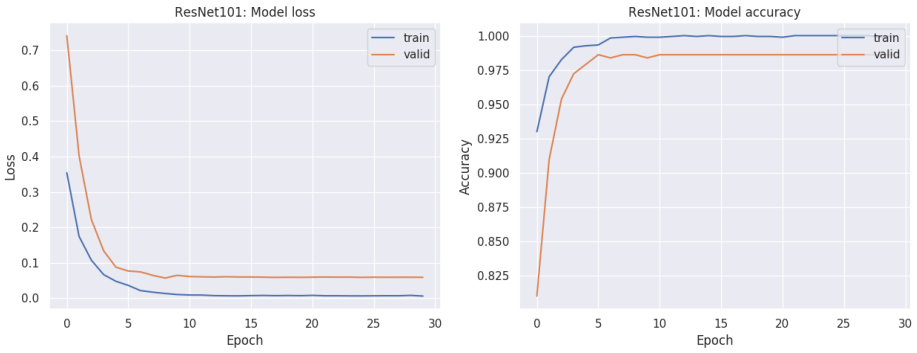
Figure 4: Well Confusion Matrix for ResNet50, ResNet101 and DenseNet121 respectively using K-fold partition 2

4.1.2 Model Loss and Model Accuracy

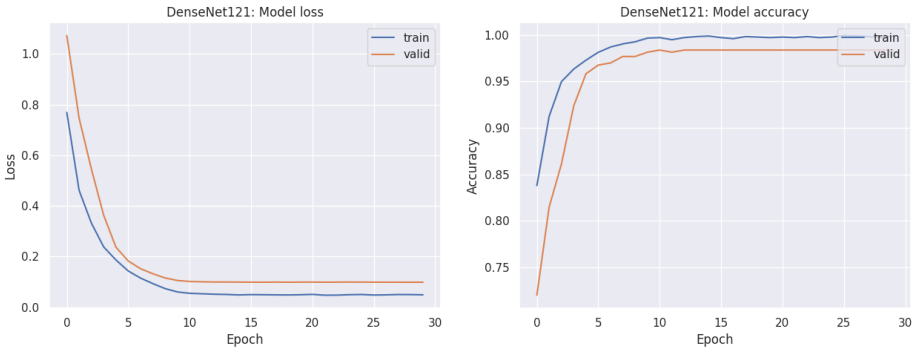
For each CNN that was implemented, graphs was created to visualize the model loss and model accuracy during training. The model loss graph visualize the error loss during the training and the model accuracy visualize how the accuracy change during training. The graphs for each CNN that were produced using K-fold partition 2 are represented in Figure 5. The model loss and model accuracy that was produced for the different CNNs using K-fold partition 3 can be found in Appendix 1.



(a) ResNet50



(b) ResNet101



(c) DenseNet121

Figure 5: Model Loss and Model Accuracy plots for a) ResNet50, b) ResNet101 and c) DenseNet121 using K-fold partition 2

4.1.3 Train, validation and test accuracy

The train, validation and test accuracy for the different CNNs produced by using the K-fold partition 2 and K-fold partition 3 are presented in Table 3 and Table 4 respectively. The test accuracy is presented both per image and per well, where one well contains several images of the same compound and concentration. Since there were no significant difference between the CNNs, the rest of the results that will be presented and discussed have been trained on ResNet50, but some results with the other CNNs can be found in the Appendix.

Table 3: Training, Validation and test accuracy for the different CNNs using K-fold partition 2

CNN	Training	Validation	Test (image)	Test (well)
ResNet50	0.999	0.980	0.862	0.931
ResNet101	1.0	0.982	0.874	0.952
DenseNet121	0.997	0.981	0.867	0.961

Table 4: Training, Validation and test accuracy for the different CNNs using K-fold partition 3

CNN	Training	Validation	Test (image)	Test (well)
ResNet50	0.993	0.968	0.969	0.992
ResNet101	1.0	0.973	0.974	1.0
DenseNet121	0.991	0.974	0.970	0.992

4.2 Comparing different nonconformity functions

4.2.1 Calibration plots

For each combination of CNN and nonconformity function, aggregated calibration plots were created. The calibration plots shows the error rate against the significance. The goal is to have a result as close to the straight line as possible, meaning that if you have a significance of 0.2 you also have an error rate that is 0.2. If the curves lies above the straight line it means that the error is greater than the significance. If the curves on the other hand is below the straight line it means that you have a lower error rate than the significance, which could be misleading when making decisions. Figure 6 shows the aggregated calibration plot for ResNet50 with IP nonconformity function when using the K-fold partition 2. The different colors represent the different classes 0-9 of MoAs. What number corresponds to which MoA can be found in Table 1. The black curve

shows the average result among the different classes. The aggregated calibration plots for ResNet50 with M and IP_M nonconformity function when using K-fold partition 2 can be found in Appendix 2.

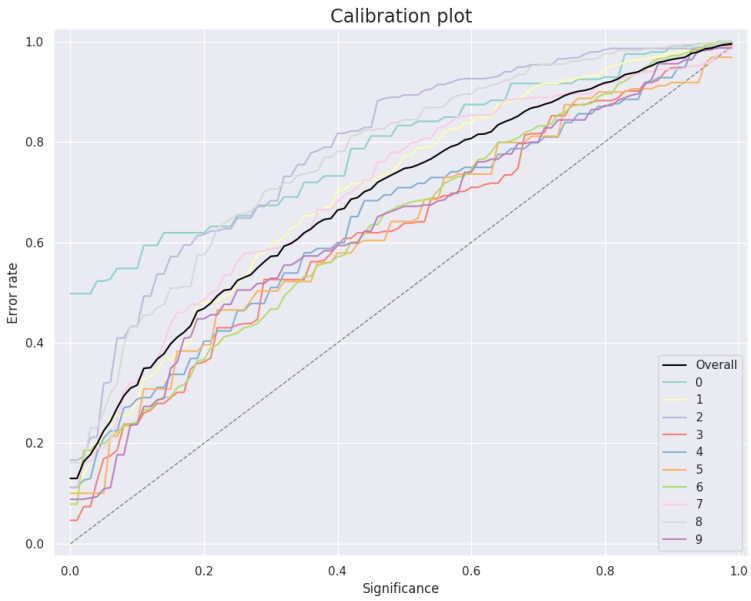
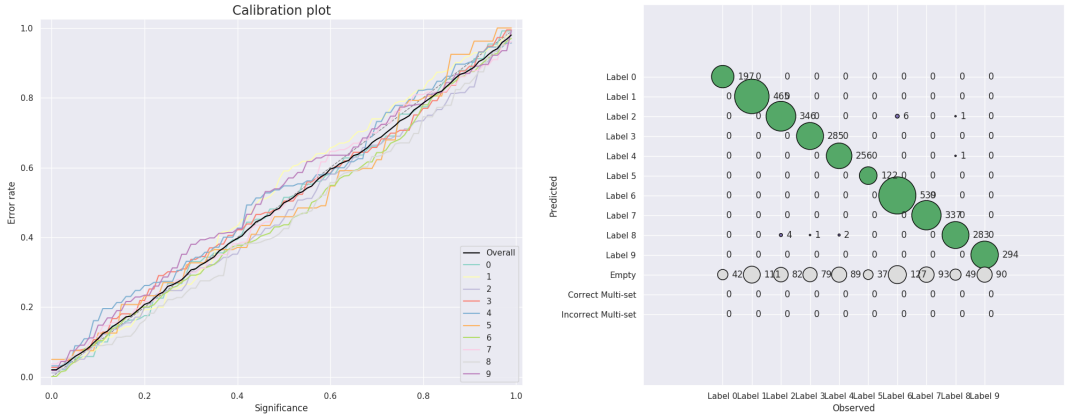


Figure 6: Calibration plot for K-fold partition 2. ResNet50 and IP nonconformity function was used

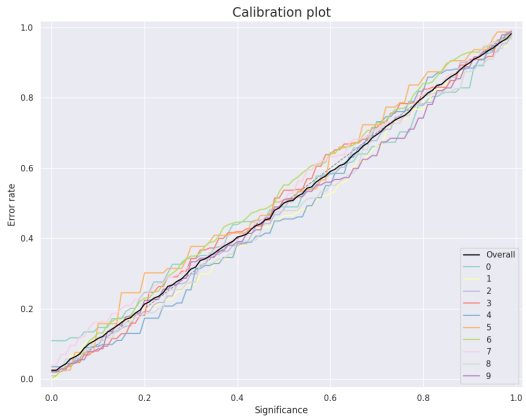
4.2.2 Validation of implementation

After changing the K-folds into K-fold partition 3, to validate the implementation, new calibration plots were created for each nonconformity function, see Figure 7. Bubble plots for each run are also presented in Figure 7. The bubble plots show the distribution of the predictions from using Conformal Prediction. The y-axis represents all possible prediction sets including ”Empty”, ”Correct multi set” and ”Incorrect Multi-set”. The x-axis represents the true classes, where the corresponding MoAs to classes 0-9 can be found in Table 1. The numbers in the plots show how many examples has been predicted to with that specific prediction set and what true class they belong to. The calibration plots and bubble plots for the different nonconformity functions applied on ResNet101 and DenseNet121 using K-fold partition 3 can be found in the Appendix 3.

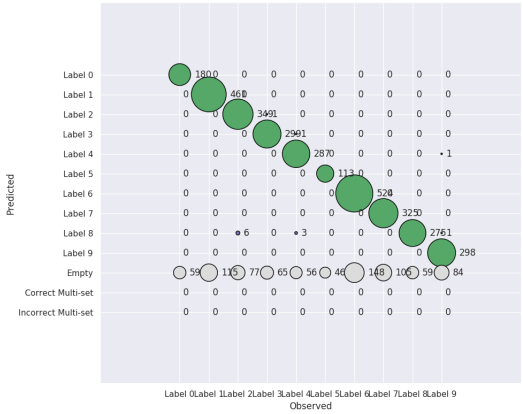


(a) Calibration Plot using IP

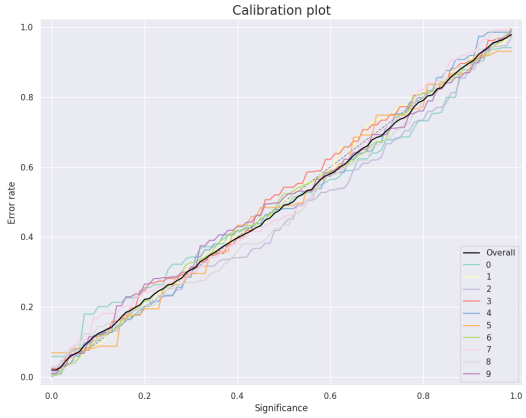
(b) Bubble Plot using IP



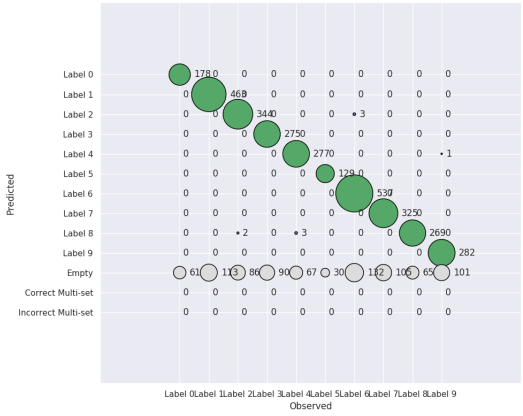
(c) Calibration Plot using M



(d) Bubble Plot using IP



(e) Calibration Plot using IP_M



(f) Bubble Plot using IP_M

Figure 7: Calibration plots and Bubble plots for IP (subfigure a-b), M (sunfigure c-d) and IP_M (subfigure e-f) nonconformity functions applied on ResNet50 using K-fold partition 3

4.2.3 Efficiency of the different nonconformity functions

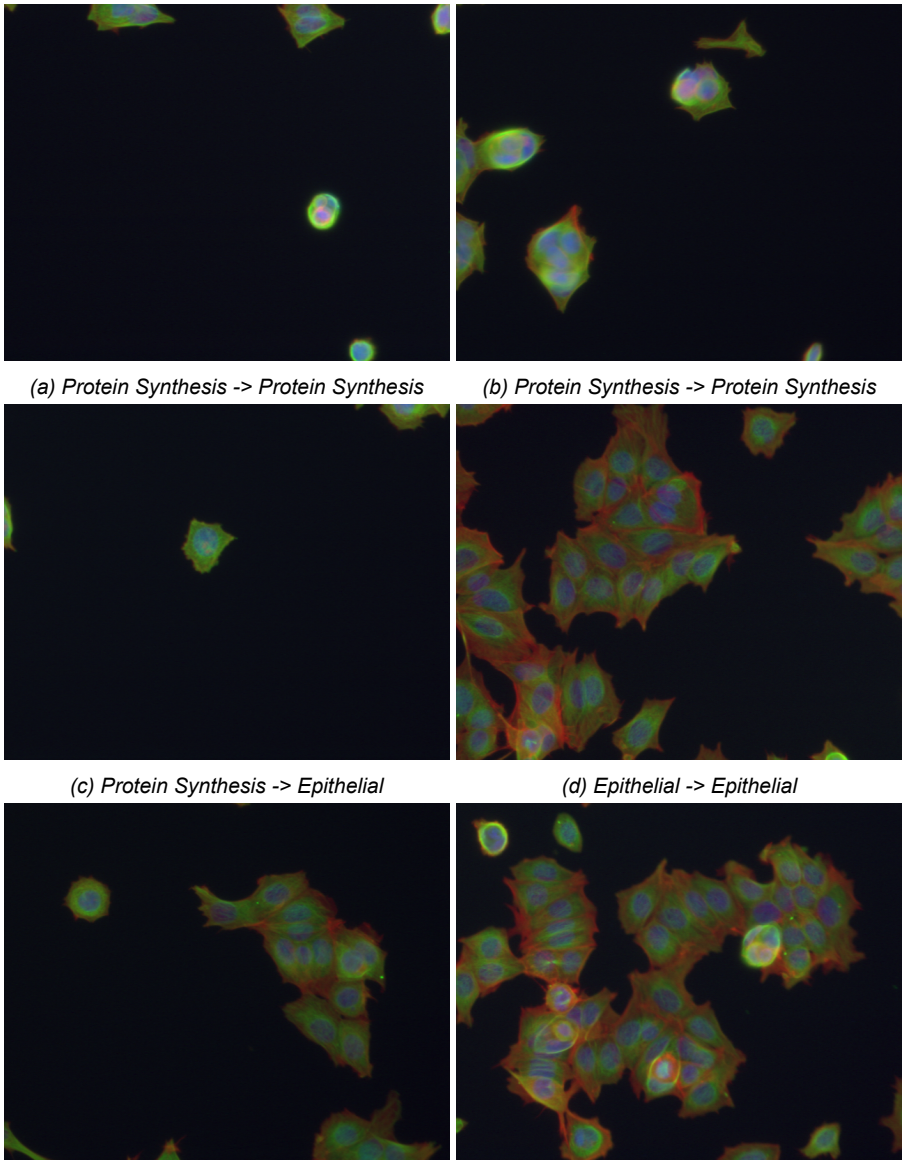
When running the program with the different nonconformity functions, the oneC and avgC was measured. The result for each nonconformity function using K-fold partition 3 is presented in Table 5.

Table 5: Efficiency of the different nonconformity functions, presented as the measurements oneC and avgC

Nonconformity function	**oneC**	**avgC**
IP	0.797	0.797
M	0.793	0.793
IP_M	0.784	0.784

4.3 Example images

Some example are presented in Figure 8. Subfigure a), b) and c) all belong to Protein Synthesis, but c) got predicted to belong to Epithelial. Subfigure d), e) and f) all belong to Epithelial but f) got predicted to belong to Protein Synthesis. All examples with the same MoA has also been treated with the same compound.



(a) Protein Synthesis -> Protein Synthesis *(b) Protein Synthesis -> Protein Synthesis*

(c) Protein Synthesis -> Epithelial *(d) Epithelial -> Epithelial*

(e) Epithelial -> Epithelial *(f) Epithelial -> Protein Synthesis*

Figure 8: Example images. a), b) and c) all belong to the class Protein Synthesis but c) got predicted to belong to Epithelial. d), e) and f) all belong to Epithelial but f) got predicted to belong to Protein Synthesis

5 Discussion

5.1 Comparing different CNNs

When comparing the performance of the different CNNs that was used in this project (ResNet50, ResNet101 and DenseNet121) the accuracy was evaluated. In Figure 5 as well as Table 3 we can see the accuracy of the training and validation when using K-fold partition 2. We can conclude that the different CNNs perform very similar and there is no major improvement that can be made by changing the architecture of the CNN to the ones implemented in this project. They all perform above 99% on training and above 98% on validation. When looking at Figure 5, we can also see that the curves have stabilized before finishing training which means that the number of epochs was enough.

From the confusion matrices for the different CNN architectures, see Figure 4, we can see the accuracy for each class instead of just the total accuracy. These Figures visualize the accuracy for each class per well, which could show if one class performed worse than other classes. From these figures we can conclude that the accuracies are high for all the classes, which was expected since the total accuracy was high. We can also conclude that there does not seem to be any significant difference between the different CNNs.

The training and validation accuracy when using partition 3 can be found in Table 4 as well as in Figure 9-11 in Appendix 1. Here we can conclude that there is no significant difference in accuracy during training and validation when using K-fold partition 2 and 3.

5.2 The different K-folds

5.2.1 K-fold partition 1

The first K-fold partition that was used in this project was K-fold partition 1 as described in section 3.2.5. This partition divided the data so that all images of one compound were placed in the same K-fold. This implementation was given by the original code and the purpose of this partition was to validate the model scientifically and to see how the model performs when it is tested on a new compound that has not been seen during training. Unfortunately an error was found in the scripts resulting in duplicated images that existed in more than one fold. This invalidates the results produced with this partitions in some different ways. Either, if the duplicated image exist in both training and test, it means

that the the model is tested on an image that has already been seen during training which makes it easier for the model to predict. Therefore it results in a higher accuracy, but not necessarily a better performance. If it exists in validation and test, it means that more images will end up in training since the validation size is fixed to 20 % of the data when the test set is removed which also benefits the training procedure without necessarily a valid improvement. When this error was found, all results produced before this were therefore invalidated.

5.2.2 K-fold partition 2

The K-fold partition 2 was identical as K-fold partition 1 except that the duplicated rows were deleted, as described in section 3.2.6. When deleting the duplicated examples, the test accuracy per image was around 86-87 % for the different CNNs, see Table 3. This was around 10 % lower than the validation accuracy for the same K-fold partition and also about 10 % lower than the test accuracy per image before removing the duplicated rows. This is an indication that the previous accuracy was based on duplicated images that existed in both the training and test set which made it easier for the model to predict since it was not new data. When deleting them, the images in the test set were only new images, which is what it should be, but it made it harder for the model to predict. This probably means that the examples in the training set for some of the classes were too few and and the examples in the test set were too different from the images in the training set in order to achieve high accuracy.

As mentioned in section 3.2.6 the reason for the setup of K-fold partition 2 with all images of the same compound were placed in the same K-fold was to validate the model scientifically. With this partition we keep one compound (or two compounds for some MoAs) exclusively for testing which could be used to validate how the model performs when a new compound is tested. The goal with this project was to investigate if this kind of tool could be used for predicting the MoA of new drugs and potentially be used in drug discovery and development. In order to be used for that purpose, it has to be possible to add new compounds that it has not been trained on to potentially see if it has the same MoA as some other drug that has been a part of the training. By keeping one compound exclusively for testing, it could verify that the model works for this purpose.

The result from the K-fold partition 2 shows that when testing the model on a new compound it was not able to connect it with the MoA of the compounds that was used while training the model as well as it should. This can both bee seen by the test accuracy shown in Table 3 and in the calibration plot shown in Figure 6. The calibration plot shows that the error rate is higher than the significance level meaning that we get a higher error than expected.

5.2.3 K-fold partition 3

To validate that the implementations were made correctly and gave valid results, the K-fold partition 3 was tested as described in section 3.2.7. This partition divided all compounds into all K-folds to make sure all compounds were a part of both Training, validation and test. This file sorting allow us to verify that the implementations works correctly and are valid, based on the assumption that the model can predict the MoA with high accuracy if it has been trained on all compounds. However this does not validate the model scientifically for the purpose of predicting the MoA of a new compound.

The training, validation and test accuracy from this K-fold partition are presented in Table 4. Here we can conclude that the training and validation are comparable to K-fold partition 2 (Table 3). The test accuracy however have increased using this partition and result in a test accuracy per image above 96 % for all CNNs. Looking at the calibration plots in Figure 7, we can see that the average error rate follows the significance well which confirms that the implementations are valid.

5.3 Comparing different nonconformity functions

5.3.1 Calibration plots and bubble plots

To compare the performance of the different nonconformity functions, calibration plots were produced for each function. Figure 7 shows the calibration plots for IP, M and IP_M respectively applied on ResNet50. Here we can see that the average error rate between the classes follows the significance well meaning that they all perform well and give accurate results. However if you look at the bubble plots, see Figure 7, we can see that none of the nonconformity functions result in any multi class predictions. All predictions made are either single or empty. This indicates that using this dataset the examples of different MoAs are too different from each other which means that the model is able to distinguish between the different MoAs. This means that the MoAs in general are too different from each other to include more than one class label in the prediction set. It also indicates that the some of the test examples are too different from the examples in the calibration set to be able to include any class label in the prediction set at the specified significance level.

We can also find some single predictions that have been predicted incorrectly, but the majority are either correct single predictions or empty predictions. A question one could ask is why the errors are not picked up as multi-class predictions. If the p-values of the incorrect predicted class and true class for those examples are both within the confidence

level, they would both have been included in the prediction set, but in these cases it seems that the images were too different from the examples of the same class in the training resulting in an incorrect predicted class instead.

5.3.2 Efficiency of the different nonconformity functions

Looking at the efficiency of the nonconformity functions in Table 5 we can see that there is no difference in oneC and avgC for any of the nonconformity functions due to no multi set predictions. The avgC value is below 1 since some of the predictions were empty, but all of the non-empty predictions were singletons so the avgC measure is hard to use to find differences between the nonconformity functions. We can see that IP has a slightly higher fraction of singleton prediction than M and IP_M, but it is not a significant difference. This means that this model with this dataset was not optimal for measuring the difference in efficiency between the nonconformity functions. Therefore the difference in efficiency between the nonconformity functions that has been mentioned in the literature, see section 3.4.3 could not be confirmed.

5.4 Biological interpretation

The question is, why does it work better when the compounds are divided into all sets instead of keeping some compounds exclusively for testing? This indicates that the difference between the compounds are too big for the program to be able to connect them even though they have the same MoA. This indicates that the Morphological effect on the cells vary too much between the compounds for the model to be able to connect them. It could for example be side effects that result in a difference in the cell. However this has to be investigated further in order to make valid conclusions.

When looking at the example images presented in Figure 8, we can see that there seem to be some differences between the images belonging to different classes. However it is hard to know what parameters in the images affected how the program chose to predict them. In subfigure f) we can see some small green spots that are similar to the ones in a), b) and c) which might have cased the program to predict it wrong, but on the other hand a similar spot can be seen in Figure e) which did not get predicted wrong.

5.5 Future Work

For the produced model to be used for the purpose of predicting the MoA of unknown compounds and potentially be used in drug discovery some further work is required. As a start, the implementations has to be tested with another dataset to confirm the validity. Also, the possibility to exclude compounds for testing to be able to predict new compounds has to be further investigated.

6 Conclusion

The results from this project shows that the tools that has been developed are powerful for predicting the MoA of a specific drug from cell painting images, if the specific compound has been used in the training of the model. This concludes that the implementations that have been made are valid. However, the goal of the project was to investigate whether these tools could be used to predict the MoA of new compounds and in the long term potentially be a complement in drug development. This is not confirmed through this project since the performance decrease when the models were tested on new compounds. Therefore some further work is required for these tools to be able to be used for that purpose. Even though the implementations in this project did not validate the model for the original purpose and with this dataset, the work could be of good use for further work within the research group.

7 Ethics and conflict of interest

This project has no need for an ethical approval and does not involve any conflict of interests.

Regarding ethical aspects of this project, there are always some questions that could be asked when it comes to implementations of AI. In this project, the implementations are made for a research purpose in the field of image analysis of cell painting images with the long term goal to potentially be used in drug discovery, which could be argued is a good cause. However, there is no guarantee that the product produced in this project will not be used for other purposes in the future, which is hard to regulate. There is an

ethical debate regarding AI. Many believe that AI can be a powerful tool to automate and make processes more effective, but there are also questions raised regarding what happens if powerful AI tools ends up in the wrong hands. Regarding this project, the product is quite specific for this purpose and therefore the assessment is that the risk of this project contributing to AI being used for the wrong purpose is insignificant.

8 Acknowledgments

I would like to thank my Supervisor Ebba Bergman for her support and encouragement during the whole project as well as the rest of the pharmbio group at the department of Pharmaceutical Biosciences at Uppsala University for their support. I would also like to thank my subject reader Ida-Maria Sintorn for her feedback and inputs. At last I would also like to express my gratitude for my examiner Pascal Milesi and project coordinator Lena Henriksson for their guidance and support.

References

Aleksandrova M, Chertov O. 2021. Impact of Model-Agnostic Nonconformity Functions on E□ciency of Conformal Classifiers: an Extensive Study 20.

Alvarsson J, Arvidsson McShane S, Norinder U, Spjuth O. 2021. Predicting With Confidence: Using Conformal Prediction in Drug Discovery. Journal of Pharmaceutical Sciences 110: 42–49.

Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L. 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data 8: 53.

Bray MA, 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. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 9 Primary_atype: Protocols Publisher: Nature Publishing Group Subject_term: Fluorescence imaging;High-throughput screening;Image processing;Phenotypic screening Subject_term_id: fluorescence-imaging;high-throughput-screening;image-processing;phenotypic-screening.

Brownlee J. 2017. A Gentle Introduction to Transfer Learning for Deep Learning.

Brownlee J. 2018. Difference Between a Batch and an Epoch in a Neural Network.

Caie PD, Walls RE, Ingleston-Orme A, Daya S, Houslay T, Eagle R, Roberts ME, Carragher NO. 2010. High-Content Phenotypic Profiling of Drug Response Signatures across Distinct Cancer Cells. Molecular Cancer Therapeutics 9: 1913–1926. Publisher: American Association for Cancer Research Section: Research Articles.

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. 2009. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition. 248–255. ISSN: 1063-6919.

Hofmarcher M, Rumetshofer E, Clevert DA, Hochreiter S, Klambauer G. 2019. Accurate Prediction of Biological Assays with High-Throughput Microscopy Images and Convolutional Networks. Journal of Chemical Information and Modeling 59: 1163–1171. Publisher: American Chemical Society.

Johansson U, Linusson H, Löfström T, Boström H. 2017. Model-agnostic nonconformity functions for conformal classification. 2017 International Joint Conference on Neural Networks (IJCNN). 2072–2079. ISSN: 2161-4407.

Kensert A, Harrison PJ, Spjuth O. 2019. Transfer Learning with Deep Convolutional Neural Networks for Classifying Cellular Morphological Changes. Slas Discovery 24: 466–475.

Ljosa V, Sokolnicki KL, Carpenter AE. 2012. Annotated high-throughput microscopy image sets for validation. Nature Methods 9: 637–637. Number: 7 Publisher: Nature Publishing Group.

Salters-Pedneault K. 2020. What Does "Mechanism of Action" Mean? Section: Very-well.

Shafer G, Vovk V. 2007. A tutorial on conformal prediction. arXiv:0706.3188 [cs, stat] ArXiv: 0706.3188.

9 Appendix 1

Model loss and model accuracy for the different CNNs using K-fold partition 3 are presented in Figure 9-11 respectively.

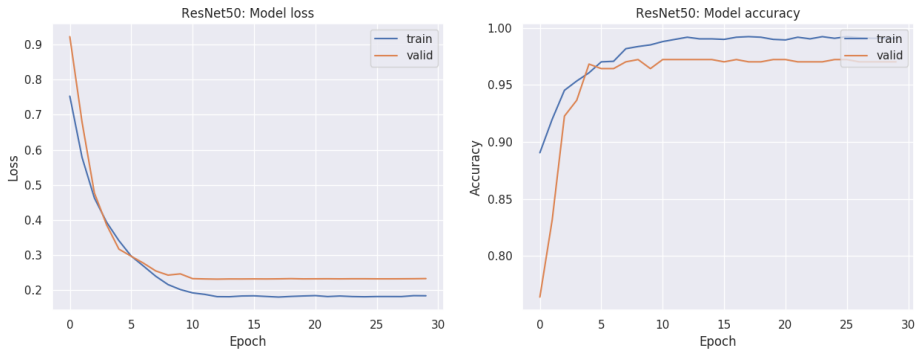


Figure 9: Model loss and Model accuracy for ResNet50 using K-fold partition 3

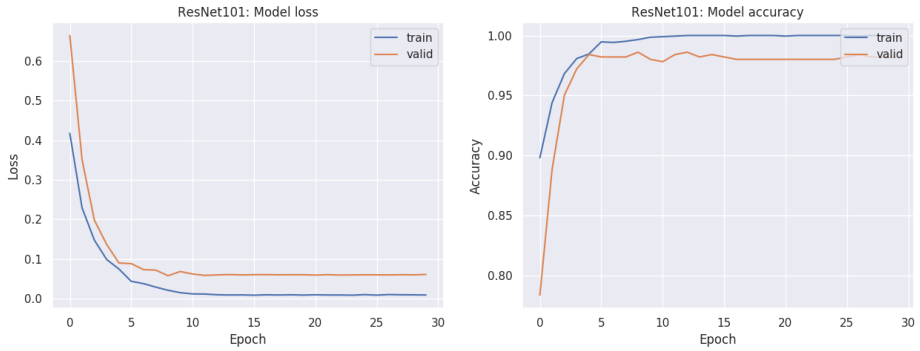


Figure 10: Model loss and Model accuracy for ResNet101 using K-fold Partition 3

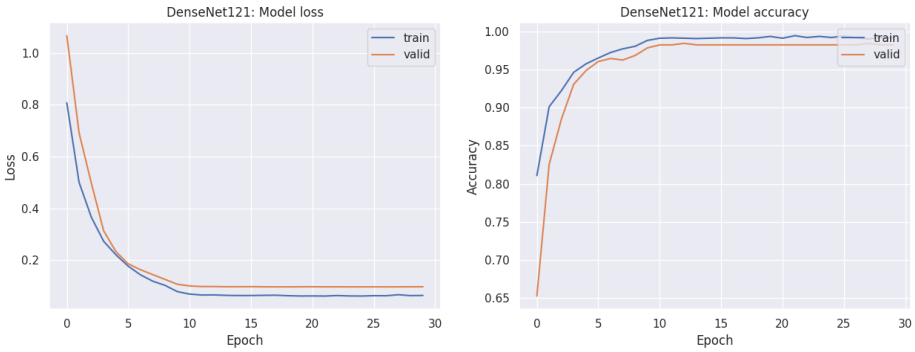


Figure 11: Model loss and Model accuracy for DenseNet121 using K-fold partition 3

10 Appendix 2

Aggregated calibration plots for ResNet50 with M and IP_M nonconformity functions using K-fold partition 2 can be found in Figure 12 and 13

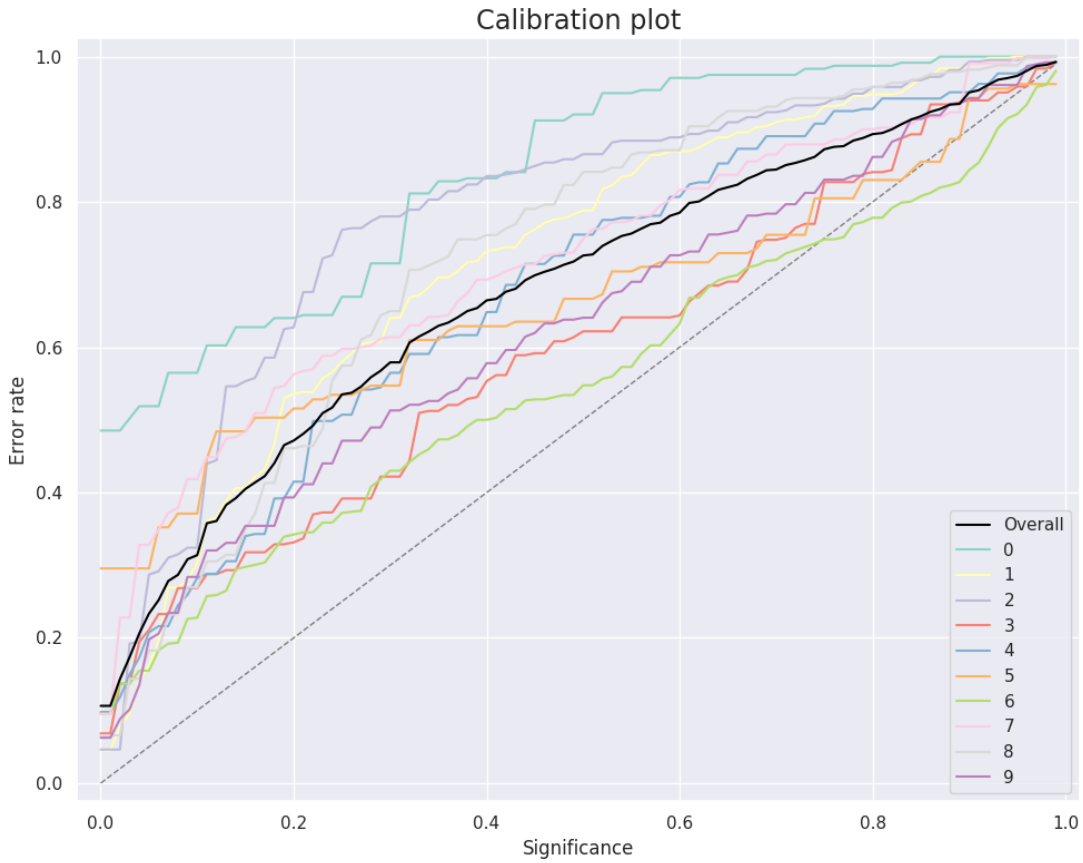


Figure 12: Aggregated calibration plots for ResNet50 with M nonconformity functions using K-fold partition 2

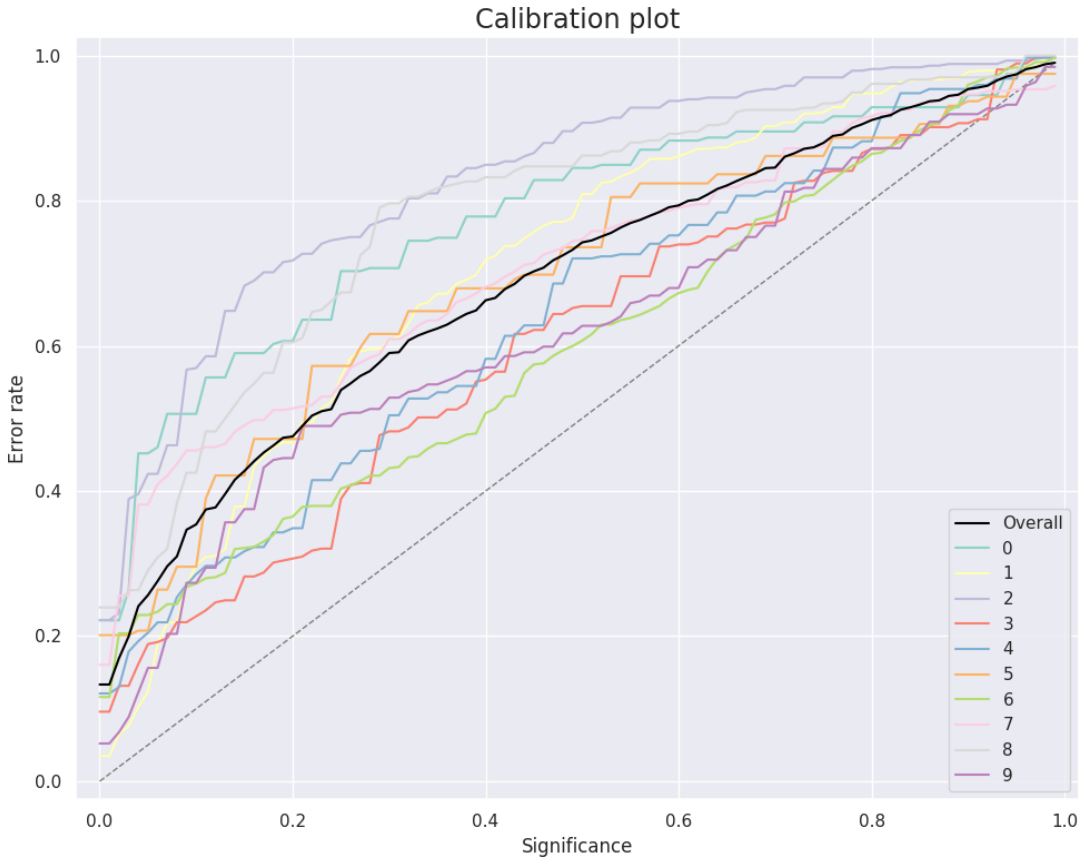
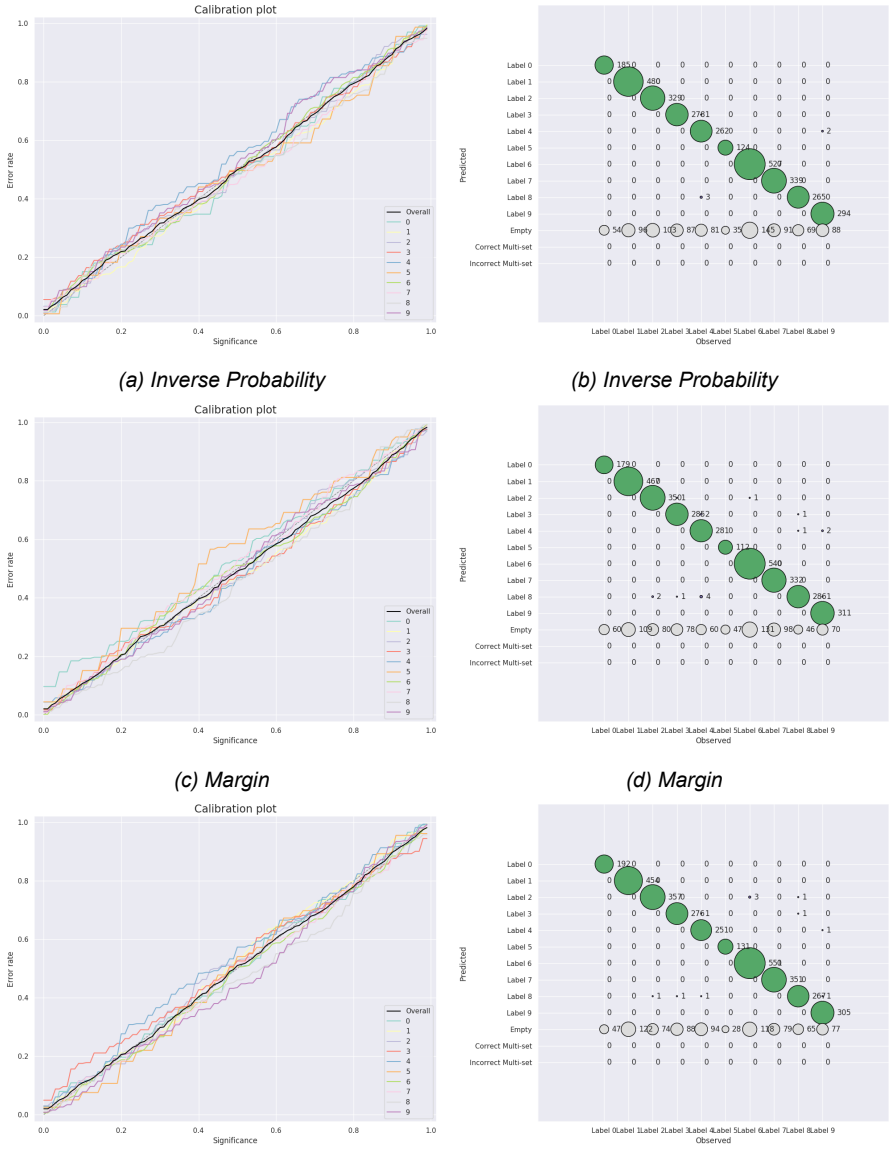


Figure 13: Aggregated calibration plots for ResNet50 with IP_M nonconformity functions using K-fold partition 2

11 Appendix 3

Calibration plots and bubble plots for the different nonconformity functions trained on ResNet101 and DenseNet121 using partition 3 Are presented in Figure 14 and 15 respectively.

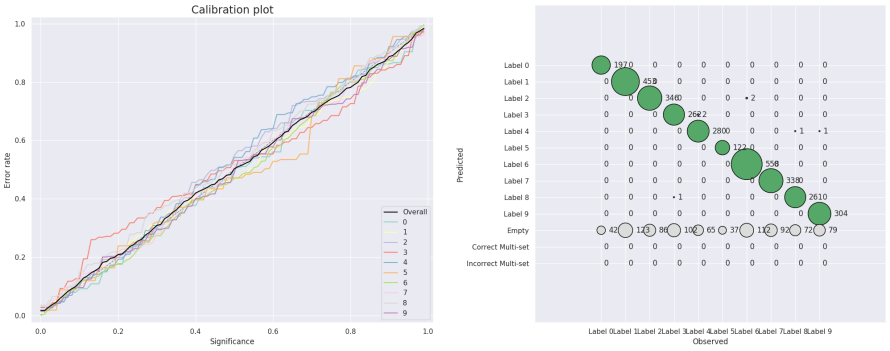


(a) Inverse Probability *(b) Inverse Probability*

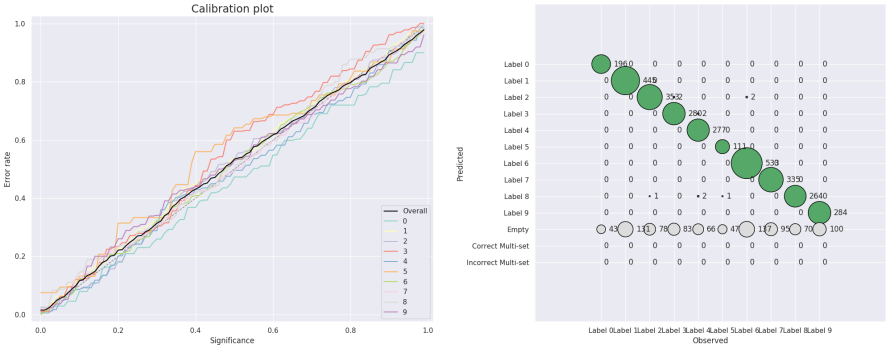
(c) Margin *(d) Margin*

(e) Combination *(f) Combination*

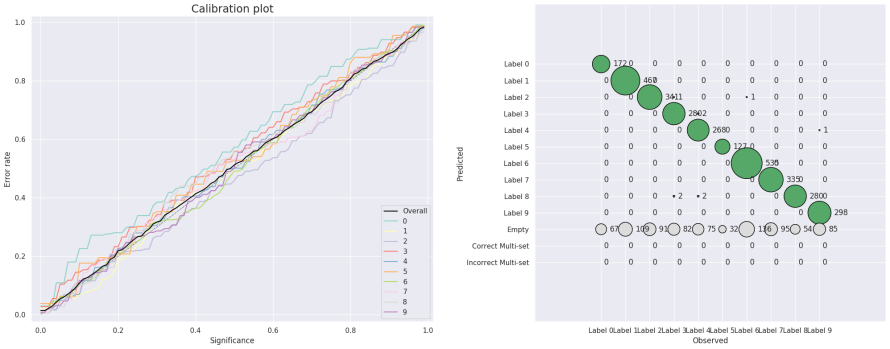
Figure 14: Calibration plots and Bubble Plots for the different nonconformity functions applied on ResNet101



(a) Inverse Probability *(b) Inverse Probability*



(c) Margin *(d) Margin*



(e) Combination *(f) Combination*

Figure 15: Calibration plots and Bubble Plots for the different nonconformity functions applied on DenseNet121