Department of Information Systems

What is the best prediction method for identifying a certain disease given a certain medical data set?

Yagel Shefer-315398404

Liron Bohman – 208304576

Hadar Margalith- 318187655

Date: 24/07/2023

Abstract:

In the field of medical research, the task of selecting the most appropriate prediction methods can be laborious and resource-intensive, especially due to the significant data volume and complex computations involved, particularly when dealing with emerging or rare diseases.

To overcome this challenge and drive a transformative change in medical research, we introduce an AI-based platform aimed at optimizing the selection of the best prediction models, empowering researchers to make efficient decisions and accelerate evidence-based medicine research.

Our platform provides essential performance metrics for major prediction methods, presenting users with accuracy percentages and runtimes of eight machine learning approaches. Leveraging neural networks and image-based learning, the platform enables comprehensive assessments of various diseases.

The data is from 31 datasets containing genetic features related to different diseases. The results indicate that the neural network model achieves high accuracy, explaining between 90% to 100% of the data.

Through this innovative solution, we hope for a paradigm shift in medical research and patient care, where data-driven decisions and AI-driven insights lead scientific inquiry, ultimately benefiting patients and healthcare providers. By expediting and refining predictions, our platform empowers medical professionals to enhance treatments, improve healthcare outcomes, and ultimately make a positive impact on human lives.

Introduction:

In the world of medical research, the pursuit of accurate predictions and evidence-based decision-making has become increasingly vital. As statisticians and researchers seek to maximize the value of their projects, they turn to various machine learning methods to identify the most precise prediction models in the shortest possible time .[1]

However, big data analysis in the medical field often involves vast amounts of data and complex computations, requiring significant resources and time investments, especially when dealing with emerging diseases or rare syndromes. The urgency to quickly classify and identify such conditions for optimal treatment highlights the critical importance of minimizing identification time to positively impact human lives.

To address these challenges and speed up selecting the optimal prediction model, we present an AI-based platform. Our platform is designed to empower researchers and practitioners by providing them with essential performance metrics for eight major prediction methods[6]. These metrics include the accuracy percentage and running time of each method, along with the tuning parameters, eliminating the need for time-consuming Cross-Validation and impractical trials.

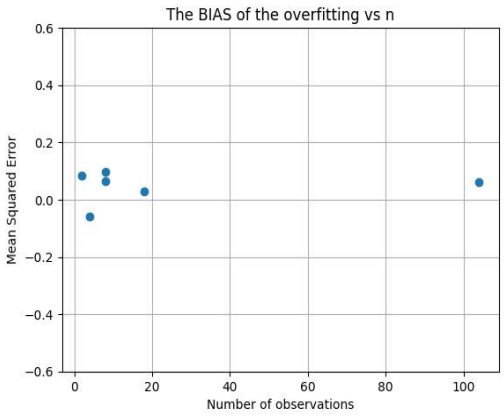
Through this research and innovation, we aim to make a meaningful impact on the medical landscape, ushering in a future where data-driven decisions and AI-driven insights guide medical research and patient care towards improved outcomes.

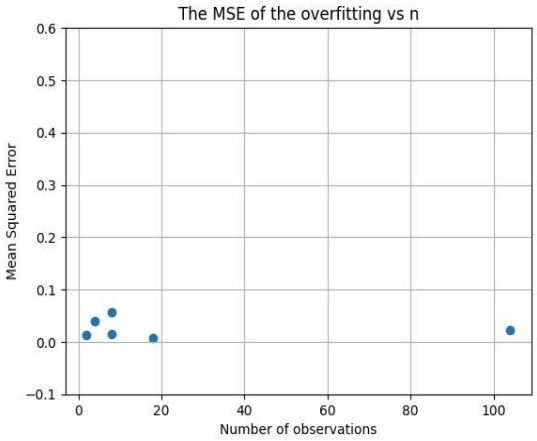
In recent years, the integration of image-based machine learning with genomic data has emerged as a powerful approach in medical research [2]. This innovative method leverages complex phenotypes to predict gene functions and gain valuable insights into the underlying mechanisms of diseases. By combining automated microscopy-based phenotyping with genetic and genomic information, researchers can comprehensively investigate cellular and organismal structures, behaviors, and disease mechanisms. Moreover, image-based profiling studies have demonstrated the potential to improve pre-clinical development by enabling scalable and systematic phenotypic profiling of small molecules, making it a promising complement to target-based in-vitro screening in drug discovery.[3,4]

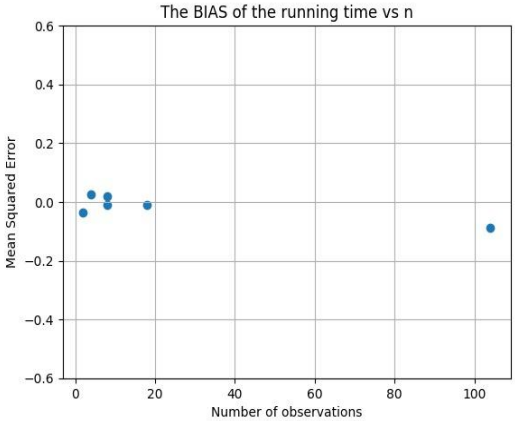
Through the integration of image-based machine learning and genomic data, the medical research community is equipped with powerful tools to predict and diagnose diseases with greater accuracy and efficiency. As a result, medicine is increasingly oriented towards disease prevention and early intervention, promising to positively impact patient care and community services [5].

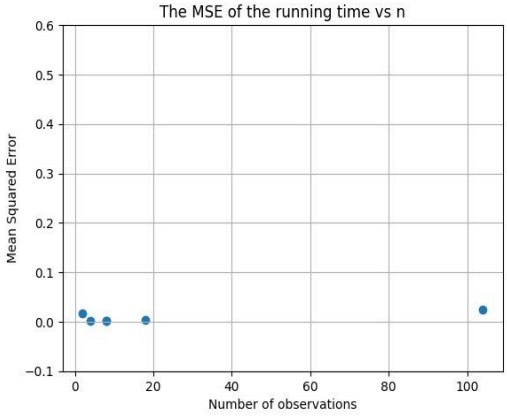
Results:

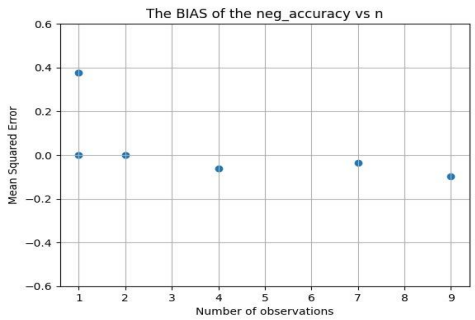
For each test dataset, we evaluated two metrics, MSE and bias, against the number of observations.

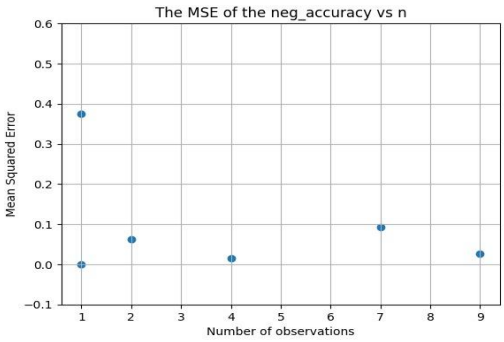


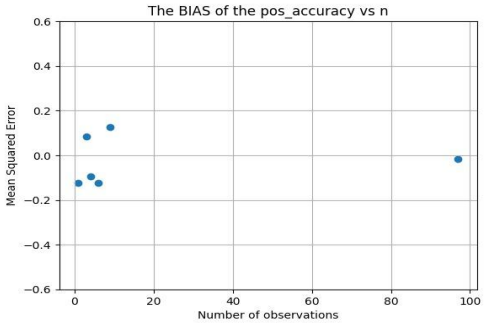


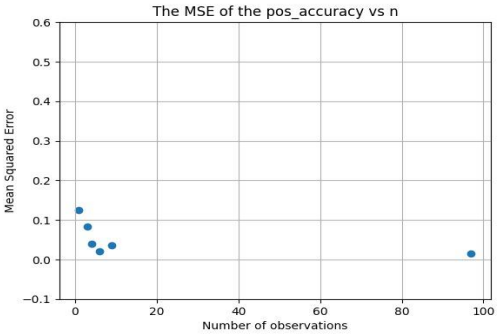




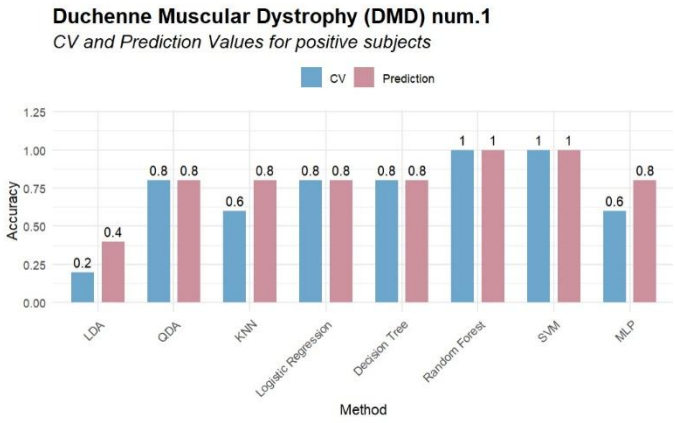


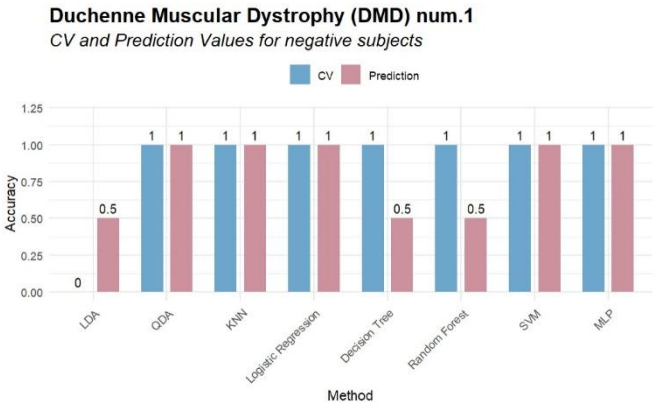


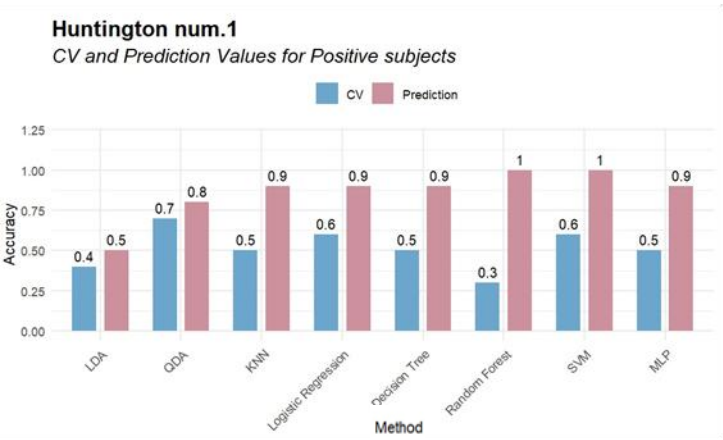


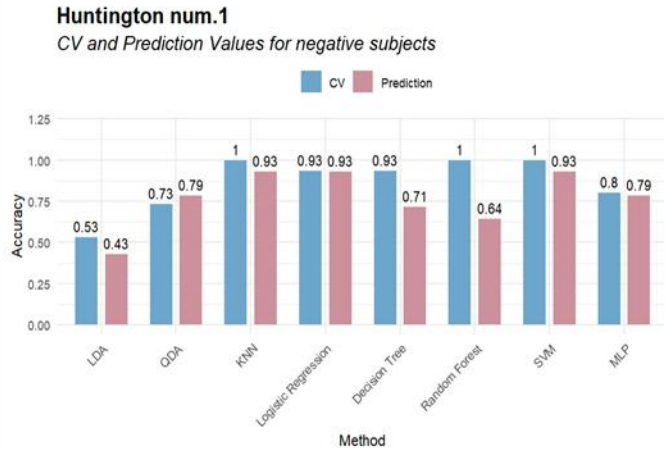


For each disease and prediction method, the accuracy percentage obtained using cross-validation is compared to the accuracy percentage achieved with the neural network.



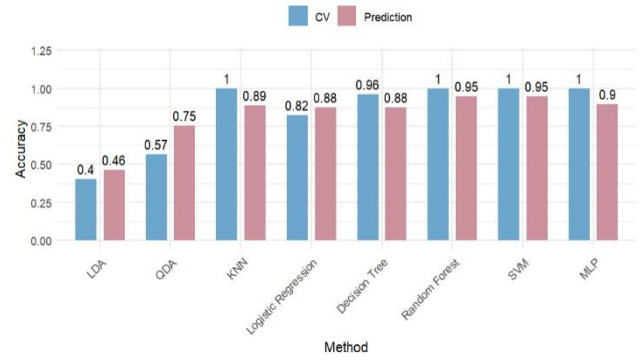






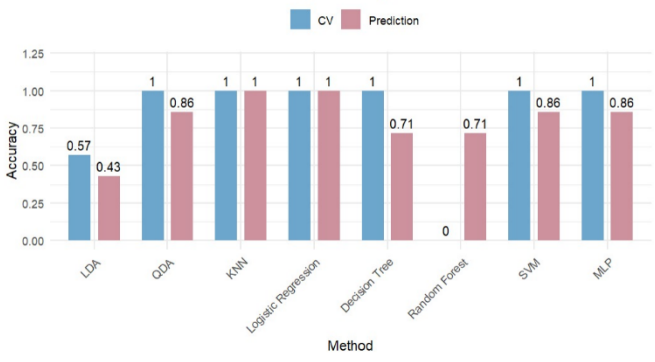
Burns

CV and Prediction Values for positive subjects



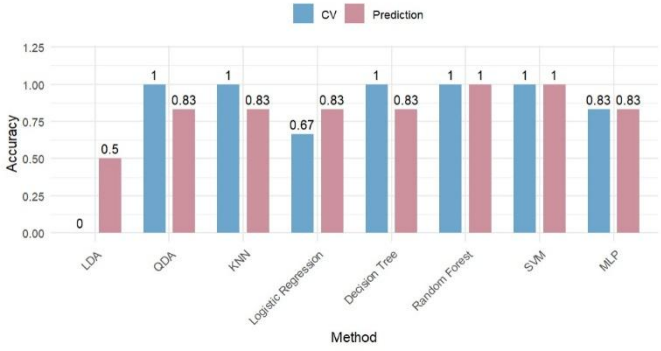
Burns

CV and Prediction Values for negative subjects



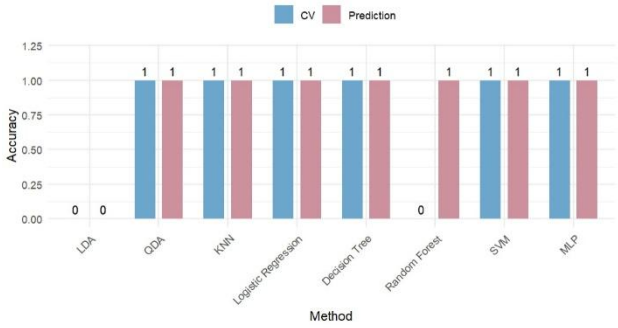
Sepsis (sample num.2)

CV and Prediction Values for positive subjects



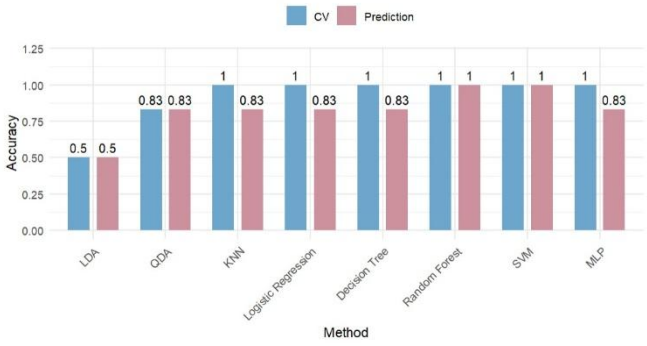
Sepsis num.2

CV and Prediction Values for negative subjects



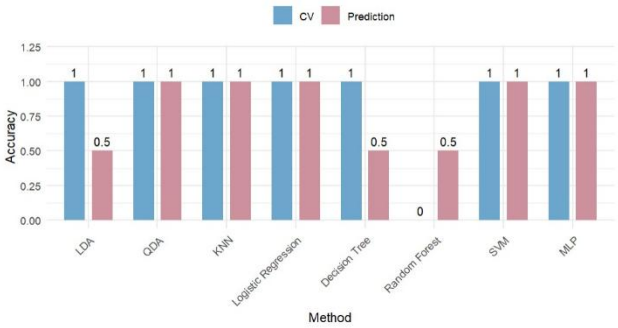
Cancer

CV and Prediction Values for positive subjects



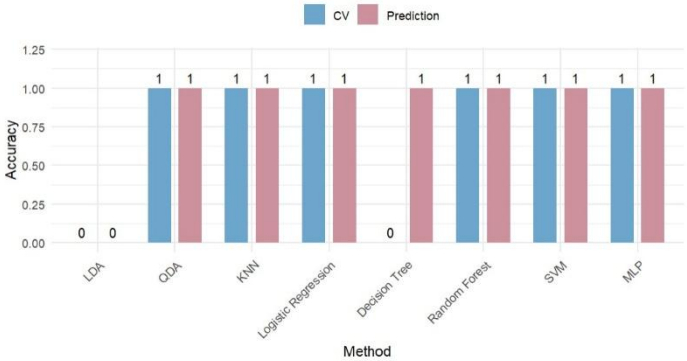
Cancer

CV and Prediction Values for negative subjects



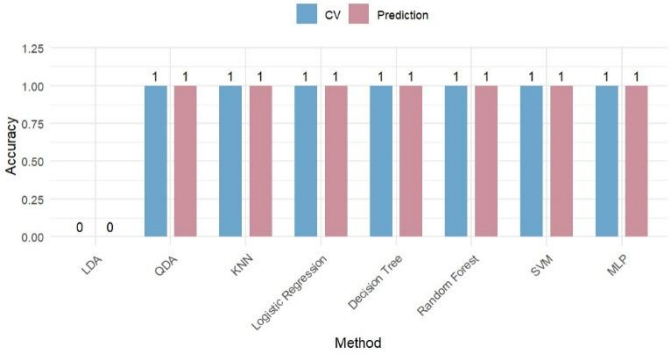
Inflammatory Bowel Diseases (IBD) num.1

CV and Prediction Values for positive subjects



Inflammatory Bowel Diseases (IBD) num.1

CV and Prediction Values for negative subjects



Discussion:

As can be seen, the results obtained in the MSE test are close to 0, while the less favorable results are close to 0.1. This means that the neural network can explain between 90% to 100% of the model using the data we provided.

The bias appears to be relatively balanced except for one outlier observation that received a high positive deviation in the accuracy of negative observations. Like the MSE, the results are low and range between 0.2 to-0.2. There is no significant positive or negative bias.

In conclusion, the neural network shortens the time to find the tuning parameters and makes it easier for the user to filter out irrelevant methods. However, there is still a need to compare the methods that received high scores to determine which one is the most suitable. Despite the narrowing of the search range for tuning parameters, which undoubtedly saves time and resources, the network has not yet fully replaced cross-validation.

Methods:

We collected 31 datasets that contract different diseases, which were tested several times using different genes. These datasets contain between 10,000-120,000 features representing different genetic sequences taken from the GEO database.

Our project is divided into two parts. The first part involves tagging the best tuning parameter and the corresponding running performance for each dataset using eight machine learning methods:

- K-Nearest Neighbors Algorithm
- SVM-Support vector machine
- QDA-Quadratic discriminant analysis
- LDA-Linear discriminant analysis
- Logistic regression
- MLP-Multilayer perceptron
- Random forest
- Decision Tree

In the second part, we will attempt to predict the tuning parameters and performance for each method by utilizing information from the dataset.

Before starting to the labeling step, we conducted various data cleaning procedures for each dataset. These steps are essential to apply the model's results accurately. If tuning parameters and accuracy results were obtained for a specific method, these cleaning steps must be implemented to reproduce the same outcome. Our datasets contain many features. To deal with this, we performed Principal Component Analysis (PCA) to reduce the number of dimensions in the datasets. Before performing this operation, we sorted the features in descending order based on their correlation with the explained

variable. This operation is intended to identify features that determine the label of a sample.

Next, we performed outlier cleaning. For this purpose, we built a function that automatically detects outliers using the kernel density function method. The function identifies outlier observations and returns them with three different parameters: the region in which the outlier occurred, an indicator that checks whether the identified outlier was higher or lower than the data distribution, and another indicator that checks whether the identified outlier occurred in a classified healthy or sick observation. Our datasets contain many features and a low number of observations. To avoid losing observations, we chose to remove features with outliers and not the observation itself. If the outlier was found in a vicinity of a feature classified as sicks only, we chose to keep the feature with the outlier because it could be the reason for being sick. If we identified an outlier in observations classified as healthy, we removed the feature containing the outlier from the dataset.

After that, we split the observations into train and test sets, so that both sets would have an equal relation between observations classified as healthy and sick. Finally, we performed standard scaling for the observations and PCA.

Next, we performed cross-validation on each dataset separately for each prediction method. for some runs, there may be tens of millions of combinations, and we do not have the means to fully evaluate all of them. Therefore, we limited the number of runs to a maximum of 10,000 runs for each method. We performed the cross-validation using the k-means fold-5 method.

To deal with the imbalance between observations classified as healthy and sick, which may occur in the data, we built a custom accuracy function that adds a balance between observations classified as healthy and sick to the tuning parameter selection criterion. This was done by giving equal weight to the accuracy obtained as a result of cross-validation for observations classified as sick only and for observations classified as healthy only.

The tuning parameters that were checked:

Method	Tuning Parameter	Role
LDA	solver	Solver algorithm for eigenvalue decomposition
QDA	reg_param	Regularization parameter
K-Nearest Neighbors	n_neighbors	Number of neighbors to consider
Logistic Regression	penalty	Penalty term for regularization
	solver	Solver algorithm for optimization
Decision Tree	max_depth	Maximum depth of the decision tree
	min_samples_split	Minimum number of samples required to split

	min_samples_leaf	Minimum number of samples required at leaf node
Random Forest	n_estimators	Number of trees in the forest
	max_depth	Maximum depth of the trees
	min_samples_split	Minimum number of samples required to split
	min_samples_leaf	Minimum number of samples required at leaf node
SVM	gamma	Kernel coefficient
	C	Regularization parameter
	kernel	Kernel function used
MLP	hidden_layer_sizes	Number of neurons in each hidden layer
	alpha	Regularization parameter

After that, we ran all eight prediction methods on the test set and found the accuracy percentage for observations classified as sick and healthy separately. Finally, for each prediction method and each dataset, we saved the accuracy percentage for the train and test sets, which allowed us to check for overfitting later.

In the second part of the project, we converted the explained variable to be the accuracy percentage, running time, and tuning parameters of each of the eight prediction methods for each dataset used as an observation in this part of the code. We loaded the original datasets before cleaning and the results obtained in the first part of the project. Next, we built a neural network that was trained on these datasets based on the results obtained.

The first step in the network is data preprocessing and feature selection. We performed several operations to prepare the dataset for analysis. Initially, missing values are dropped to ensure the dataset is clean and free from missing data. This step is crucial as missing values can introduce biases and affect the performance of the network.

Next, the remaining features were transformed to standard uniform distribution . Scaling is essential to normalize the feature values and ensure they have a similar range and distribution. By scaling the features, we avoid issues related to features with different scales dominating the learning process.

The features are then sorted based on their correlation with the target variable (y). Sorting the features by correlation helps prioritize the most relevant features in the dataset. By focusing on highly correlated features, we increase the network's ability to capture meaningful patterns and relationships that contribute to the prediction of the target variable.

The parameters for prediction were also prepared. The tuning parameters for linear models were divided into three groups: parameters that depend on the sample size, parameters that were generated using a power function, and parameters defined by specific possible values. Before conducting the analysis, Group A underwent a sample size division, and Group B underwent a logarithmic transformation. Then, all three groups underwent standardization to adjust the scales.

The performance metrics were also standardized to avoid the influence of scale differences between the metrics that focused on accuracy, overfitting, and runtime.

After the prediction process, the results will undergo a recursive process to rescale them or classify them into their respective possible categories.

After preprocessing and feature selection, the dataset is split into positive and negative observations based on the target variable (y). This separation allows the network to learn distinct patterns and characteristics associated with each class separately. By analyzing positive and negative observations independently, the network can better identify the features and patterns that differentiate the classes, leading to more accurate predictions.

Once the dataset is divided into positive and negative observations, it undergoes a transformation to create image representations. This step is crucial for leveraging the power of convolutional neural networks (CNNs) in analyzing visual data.

The positive and negative samples are ordered based on their distance to enhance the visual representation of patterns within each class. By ordering the samples, we create a more structured representation that highlights the spatial relationships and patterns present in the data.

The ordered positive and negative samples are then converted into grayscale images. We use grayscale images to simplify the input data, as they contain only shades of gray instead of multiple color channels. This simplification reduces the computational complexity and allows the network to focus on the texture and structure of the images.

Furthermore, using the resize function, the images are resized to a uniform size of 140x140 pixels. This resizing ensures that all images have the same dimensions, which is necessary for feeding them into the CNN. By standardizing the image size, the network can process the images consistently and extract meaningful features across the dataset.

<u>The input received by the neural network:</u>

The network architecture consists of a series of convolutional layers (Conv2D) and pooling layers (MaxPooling2D). These layers are the core components of a CNN and are responsible for extracting features from the input images.

The convolutional layers use filters with a size of 5x5. These filters scan the input images in a sliding window, capturing local patterns and features. By convolving the filters over the input images, the network can learn spatial hierarchies of visual information, detecting edges, textures, and other relevant patterns.

We apply a pooling layer with a filter size 2x2 after each convolutional layer. Pooling layers perform downsampling [1]by selecting the maximum value within each region of the feature map defined by the filter size.

This downsampling reduces the spatial dimensions of the feature maps while preserving the most salient features. It helps to reduce computational complexity and increases the network's ability to capture higher-level abstract representations.

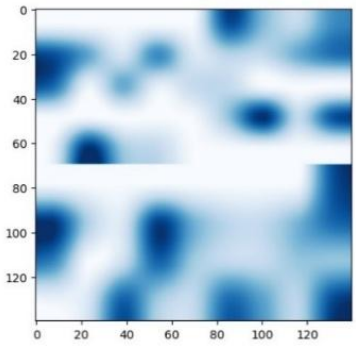


Image of a small dataset after sorting and resizing

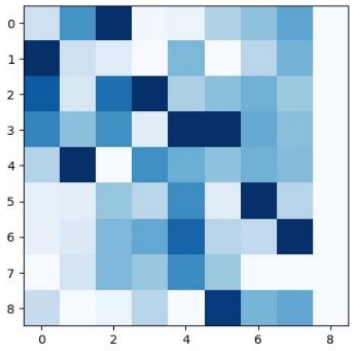


Image of a small dataset before sorting

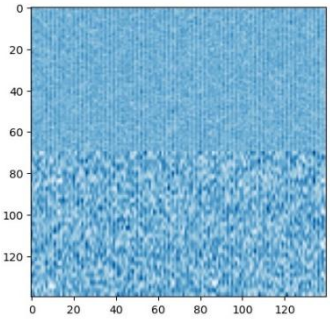


Image of a large dataset after sorting and resizing

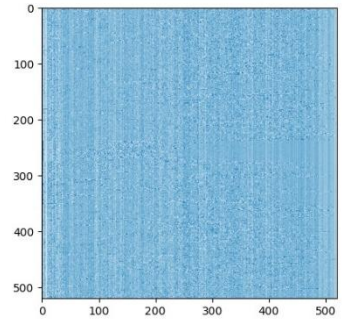


Image of a large dataset before sorting and resizing

By stacking multiple convolutional and pooling layers, the network learns more abstract and complex features, enabling it to capture intricate patterns and structures in the input images.

After the convolutional and pooling layers, the network transitions to the fully connected layers, composed of dense [2]layers (Dense). These layers capture high-level representations and make predictions based on the learned features.

[1] Downsampling reduces data points in a dataset, simplifying analysis, computation, and storage. A trade-off exists between data size reduction and preserving all original details, requiring careful consideration for specific applications.

2 A dense layer in a neural network is a linear operation followed by a non-linear activation function. It involves input X, weight matrix W, bias vector b, and activation function
f(): $Y = f(W * X + b)$

Before passing the feature maps to the dense layers, the output is flattened. Flattening reshapes the multi-dimensional feature maps into a one-dimensional vector, allowing the network to process them as a sequence of features.

We then concatenate the flattened feature vectors from the image and statistical input. This step combines the extracted features from both modalities, leveraging the complementary information they provide. By concatenating the features, the network can capture the relationships between the image-based and statistical features, enabling a more comprehensive analysis of the data.

The network includes a Dropout layer after the concatenation step, a strategy to prevent overfitting and improve generalization. Dropout randomly sets a fraction of input units to 0 during training, which helps to reduce interdependencies among the neurons and encourages the network to learn more robust and diverse representations.

Within the network architecture, specific dense layers are dedicated to predicting tuning parameters (tuning and tuning_l) and performance outcomes. These predictions are made separately, allowing the network to capture different aspects of the problem and provide valuable insights.

The tuning predictions aim to estimate the tuning parameters for the model. In this case, two dense layers, tuning and tuning_l, utilize linear activation functions. Linear activation functions provide a linear relationship between the input and output, enabling the network to approximate a linear mapping. Using linear activation functions, the network can accurately estimate the tuning parameters.

On the other hand, performance predictions are concerned with estimating performance outcomes, representing various evaluation metrics or performance measures relevant to the problem. In this architecture, the performance predictions combine linear and non-linear activation functions.

The performances dense layer, which predicts the performance outcomes, uses a linear activation function. This choice allows the network to model a linear relationship between the input features and the predicted performance. Linear activation functions are suitable for tasks where the output should have a direct linear relationship with the input.

In contrast, the tuning_l dense layer, which is also involved in the performance predictions, utilizes a hyperbolic tangent (tanh) activation function. The tanh function is a non-linear activation function that maps the input to a range between -1 and 1. By incorporating a non-linear activation function, the network can capture more complex relationships and patterns in the data, which may be crucial for predicting performance outcomes accurately.

The combination of linear and non-linear activation functions in the performance predictions allows the network to leverage both linear and non-linear relationships within the data. This flexibility enables the network to capture a wide range of patterns and make more comprehensive predictions for performance outcomes.

After constructing the model architecture, the network is compiled. The loss functions for each output are specified, including the custom loss function accuracy_mse for the performance predictions and tun_l_loss for the tuning predictions.

The model is then ready for training. Training involves feeding the preprocessed data to the model and iteratively adjusting the network's weights and biases to minimize the specified loss functions. The training process is guided by the defined metrics, such as accuracy and mean squared error, which are calculated during training to monitor the model's performance.

In conclusion, the network architecture combines the power of convolutional neural networks, dense layers, and custom loss functions to make predictions for tuning parameters and performance outcomes. The network undergoes several stages, including data preprocessing, feature selection, image preparation, CNN architecture, dense layers, concatenation, and specific predictions for tuning and performance.

Each stage explicitly enhances the network's ability to capture relevant patterns and relationships in the data. The network can model complex relationships and provide comprehensive predictions by leveraging both linear and non-linear activation functions. Through model compilation and training, the network optimizes its performance and generalizes well to unseen data.

References:

1. Chen, M., Hao, Y., Hwang, K., Wang, L., & Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. Ieee Access, 5, 8869-8879. https://ieeexplore-ieee-org.ezproxy.haifa.ac.il/abstract/document/7912315

2. Scheeder, C., Heigwer, F., & Boutros, M. (2018). Machine learning and image-based profiling in drug discovery. Current opinion in systems biology, 10, 43-52. https://www-sciencedirect-com.ezproxy.haifa.ac.il/science/article/pii/S2452310018300027

3. Bell, J. (2004). Predicting disease using genomics. Nature, 429(6990), 453-456. https://www-nature-com.ezproxy.haifa.ac.il/articles/nature02624

4. Haidich, A. B. (2010). Meta-analysis in medical research. Hippokratia, 14(Suppl 1), 29. https://www-ncbi-nlm-nih-gov.ezproxy.haifa.ac.il/pmc/articles/PMC3049418/

5. Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. In 2013 international conference on collaboration technologies and systems (CTS) (pp. 42-47). IEEE. https://ieeexplore-ieee-org.ezproxy.haifa.ac.il/abstract/document/6567202

6. Sugiyama, M. (2015). Introduction to statistical machine learning. Morgan Kaufmann.