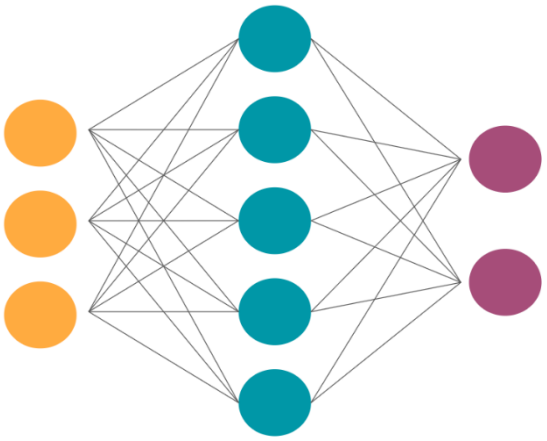

Artificial Neural Networks Performance Study

João Manuel Campelos Ferreira
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciências de Computadores
2020

Orientador
Inês de Castro Dutra, Faculdade de Ciências da Universidade do Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Para uma das pessoas que mais impactou quem eu sou hoje, que ajudou a cuidar de mim durante a minha infância e que me mostrou o significado de altruísmo e dedicação com a sua maneira de ser para com a sua família.
Que pensou sempre nos outros em primeiro lugar e só depois em si, que merece tudo deste mundo e que ficará sempre na minha memória como uma das melhores pessoas que alguma vez conheci e irei conhecer.
Tive a maior sorte em te ter na minha vida e de partilhar muitos momentos contigo que sempre guardarei comigo. Finalmente rumo para um futuro que tantas vezes falamos e que sempre me desejaste.

Dedico não só o meu trabalho mas também quem eu sou hoje, a ti,

a melhor Avó que poderia pedir.

Contents

List of Tables

List of Figures

Acronyms

ANN Artificial Neural Network

SVM Support Vector Machine

LR Logistic Regression

RF Random Forest

CNN Convolutional Neural Network

RNN Recurrent Neural Network

MLP Multilayer Perceptron

AE Auto Encoder

RBF Radial Basis Function

ESN Echo State Networks

MFCC Mel Frequency Cepstral Coefficient

Chapter 1

Introduction

In our modern society, with the evolution of technology, our need for knowledge exceeds our ability to reach it. Predictions of performance, patterns of consumer activity, clustering people into groups based on their taste and classification of instances between several characteristics are a few examples of tasks that would take large amounts of time and resources to accomplish.

That makes us turn to machine and deep learning by artificial intelligent systems with the ability of extracting knowledge from extensive amounts of data in small periods of time, at least smaller than what would take a manual approach. Deep learning is based on artificial neural networks (ANNs) which are nothing more than the computational adaptation of a brain's neural network and are going to be the core subject of this thesis.

The main focus will be the performance of ANNs across several evaluation metrics, compared to other algorithms in distinct situations, variable characteristics and multiple key points that may affect the overall learning performance, so that some conclusion can be extracted.

Such conclusions cannot be dependent on only one evaluation metric since that could lead to inconclusive or incorrect results, so evaluating such comparison also implies some range of evaluative options in order to have a more wide spectrum of results and avoid false conclusions.

1.1 Context

Nowadays, ANNs are vastly used in fields such as agriculture, education, finance, security, art, management, manufacturing, transportation, insurance, marketing, energy and banking. They are particularly popular not just due to the mapping from an input to an output but also due to its ability of self-learning, fault tolerance and non-linearity. ANNs' popularity, paired with the great results solving diverse problems, lead to an increase of the their use in different professional areas and situations.[2]

Different data mining algorithms may have different sets of characteristics that can lead to higher efficiency in finding a valid solution. The generalization of the use of a particular method

like ANNs, to solve different problems under different circumstances, may not be the best choice in professional and research areas. Even though it may produce good results, the search for the best answer must always be imposed in order to move towards scientific evolution and innovation. That can be accomplished by guaranteeing the appropriate use of an algorithm, not neglecting other valid options that could lead to better results.

So the main imposed question is "How better are artificial neural networks' performance compared to other methods, capable of solving the same problem under the same conditions and under what conditions do they differ?".

1.2 Motivation

Although the definition of dataset is somewhat ambiguous, there is consensus in the presence of four characterizing features such as grouping and content of data, relatedness between data information and a final purpose. This allowed the improvement of quality and performance in technology, services, researches, operations, among others.[25]

With the advances in deep learning, the application of the algorithm in several distinct areas increased drastically, combined with that, the fact that most performance based studies of the algorithm are conducted in specialized environments like specific datasets, specific problems, not extending that study to not so favorable conditions, is a massive contributor for the need of a study like the one presented. The motivational need for this study also relies on the increasing use of ANNs in the last decades and the wide range of fields that use this algorithm in so many different applications.

So there is a need for an extensive understanding of the performance, specially comparing with other methods in order to comprehend not just their great achievements but also their flaws, advocating their use or promoting the use of more capable algorithms that may be overlooked.

1.3 Objectives

To answer the main question of this thesis, a complete study should be able to determine the performance of ANNs under different situations, in order to have a complete and extensive knowledge of its adaptability to new and distinct aspects of the implementation and overall final performance, based on different evaluation metrics.

The different situations and characteristics must be impactful to a certain extent, like the dataset used, that can vary in size, shape and data type. When it comes to ANNs, the type of the network and the different types of layers, number of layers and neurons can also lead to different results. Those situations and conditions are important because they are the line between success or failure so they have to be impactful in general situations. The notion of failure in this case does not mean obtaining bad results or not obtaining results at all, but not

obtaining better results than other methods.

Once such performance is determined, under the same previously described conditions, one well suited competitive algorithm must be studied in order to conclude the improvement that was made by using ANNs or in the other hand, the disadvantage of their use.

Even though it is not the main topic of this thesis, this also creates an opportunity to study other methods under different conditions and create a hierarchical scale of the performance, not only comparing the results between algorithms but also criticizing each algorithm within all the distinctive factors.

In summary, the main objectives are to find some distinction between favorable and unfavorable conditions that have some type of influence in the created model's performance and to present evidence of other possible solution that would reach better results, evaluating those results in multiple angles to ensure maximum veracity.

1.4 Organization

There are four main groups of tasks to be implemented throughout the duration of the thesis. There is a bibliographic research and organization, followed by the search and selection of the main comparative aspects of the next task, the testing phase, ending with the writing of the thesis.

The first task starts when the objective of the thesis is determined. Bibliographic references, articles and the state-of-the-art study is conducted in order to support the foundation of the thesis, from the motivation, passing through the need for this study and ending in the way it should be implemented. In this phase, there is an intensive research for similar work in order to know what can and should be improved and other ways to perform the study in order to have better and more conclusive results. The second main task is the selection of all the substantial material that is going to be used. Dataset types, the actual datasets, data mining algorithms applicable to the chosen datasets, ANN architecture, comparative data mining method, validation and metrics must be selected as well as the number of datasets and dataset sizes to be compared. This is an important step because it can lead to overachieving and consequently not having the time to implement all the tests. In order to prevent that, simpler tests must be implemented in order to have a notion of the time that is going to be spent in each test phase.

After having all the selected elements of the study, there is the most time expensive phase. Here all the comparative studies must be implemented and all the results and conclusions must be extracted. Since ANNs and other algorithms may consume extensive periods of time due to the training of the networks and the algorithm computation, this phase is the longest of all tasks. The ANNs and the comparative chosen methods are implemented and compared in several conditions and situations, the ones previously mentioned. In addition, the creation of the networks and the parametrization tuning of layers, neurons, activation functions, optimizers and

number of epochs takes time since several number of neurons, layers, etc, have to be computed in order to optimize the learning performance and classification capability of the models. That factor lead to a an even greater amount of time spent.

Finally, the last task is the preparation of the document of this study. Since the previous task consumes a large amount of time, the writing of the thesis starts around the end of the testing phase. This way, a more vivid recollection of the events and conclusions can be documented, lowering the chance of missing information.

The document is partitioned in 6 chapters. Apart from this introductory chapter, chapter 2 focus on basic and related concepts behind this study and their definition and relation. Chapter 3 discusses the related work and state-of-the-art with articles and studies that present similarities with this work. Chapters 4 presents the description of every component of the study, like dataset description, pre-processing operations, constructed models and evaluation, and chapter 5 is where all results are presented, discussed and compared. The final chapter concludes the study with a summary of the study, conclusions and future work.

Chapter 2

Basic Concepts

2.1 Single neuron and deep learning

Perceptron is a binary classification algorithm that represents mathematically the biological neuron. That mathematical representation is accomplished with the application of an activation function and related weights, on an input fed by an input layer that is connected to an output node. The linear classification is able to determine distinct patterns by adjusting weights.[24]

The neuron is an adaptation of the rudimentary perceptron and allows a continuous value between 0 and 1, instead of the binary output.[27] In figure 2.1 we can see the input and a weight value, that are computed in a summation and where an activation function is applied, producing an output. That represents the basic behaviour of the neuron.

The summation is represented by

$$in_i = \sum_j W_{j,i} a_j = \mathbf{W}_i \mathbf{a}_i$$

where g is the activation function, a is the activation value of unit i, $\mathbf{a}$ is the vector of activation values for the inputs to unit i, W is the weight on the link from unit j to unit i and $\mathbf{W}$ is the vector of weights leading into unit i.

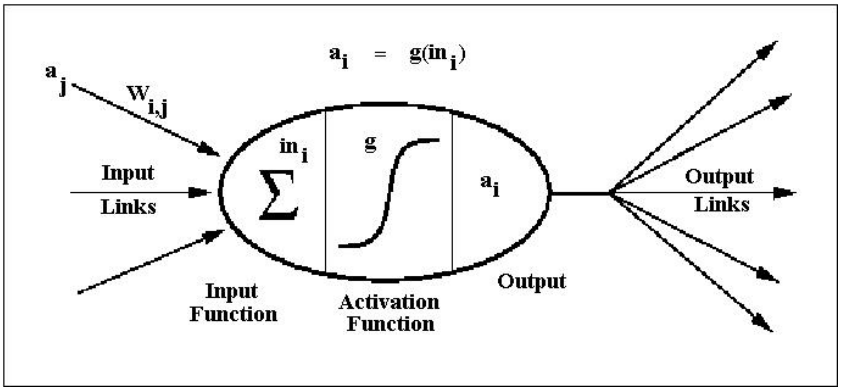

Figure 2.1: Single neuron. Source: [27]

With the evolution of technology and resources, neurons can be put together in sequences where the output of one neuron is the input of another. This allows the transition from a single neuron to deep learning.

Deep learning is a branch of machine learning, mostly accomplished with the use of ANNs, although not exclusively.

The combination of neurons in layers and the complexity of the number of components and connections, allows the resolution of equally complex problems.[12]

Deep learning allows the extraction of high-level features that are extremely hard to comprehend, by breaking the data representation into simpler parts and relating them with more complex ones, but the term deep can also be attributed to the depth that can allow the learning process.

2.2 Artificial Neural Network

ANNs are computational networks inspired by the animal's brain. They receive inputs through several parallel processors that make simultaneous computations and can be structured by layers, each one a set of neurons. They are inherently parallel due to those simultaneous computations that are made by different elements of the network.

The main components of ANNs are the neurons, each one receiving and sending it's own input and output, and the connections between them that allow the communication between nodes.[13] Those connections are made by weights that allow the computation of data, usually with a nonlinear relation between input and outputs, through learning and training.[8]

For the layer structured networks, there are three main types of layers, the input layer, hidden layers and an output layer. The neurons in each layer are connected to other neurons in the same, adjacent or in some cases non-adjacent layers, depending on the architecture of the network. That connection has an associated weight that is adjusted throughout the learning process and that

determines the relevance or strength of the connection, representing the network's parameters. If a connection's weight is lower than a predefined value also known as threshold, the signal may not be sent because that connection is not sufficient for obtaining a good output.

The networks can be single−layered or multi−layered and feedforward or recurrent and a single neuron can also receive an output produced by itself. Each neuron of the input layer receives a variable of the dataset and passes that information to another neuron. That information also known as activation is a numeric value limited by a certain range depending on the activation function. That function is what is going to change the data throughout the layers ending in an output result, presented by the output layer.

Each layer and neuron may perform different data transformations to the receiving inputs since each neuron multiplies the activation value by the associated weight. It then adjusts the resulting number by the neuron's bias, which is a specific number that adjusts the value of the neuron after all connections are processed. After that adjustment, it normalizes the output with the final activation function.

Since each neuron from each layer passes the transformed data to the next layer as an output, the model does not have to work with the raw input data each time, saving resources and bettering the performance. One of the most interesting characteristics about ANNs is the ability to modify themselves in the learning process, adapting to new data and altering the learning process.

The learning process is usually accomplished with the use of backpropagation where after one process flow takes place and the output is determined, the network compares those outputs with pre-known answers. A cost function is used to modify initial outputs based on how they differed from the known values. The function's results are pushed back across the network to adjust the weights and also the biases. This will not only allow the network to learn but also to generalize the learned knowledge, avoiding overfitting.

ANNs also have different architectures or structures like the number of layers, nodes in each layer and the connections between them, which can be total or partial depending on whether connections are only made between layers or also between nodes in the same layer.

There are several types of ANN architectures like feedforward neural networks which, like the name suggests, feeds the information from the input layer to the output layer and are usually trained through backpropagation. In the feedforward neural network we have architectures like auto encoders, radial basis function and CNN.

There are also some variations of feedforward neural networks like RNNs which are not stateless. In RNNs, we have architectures like echo state network that have random connections between neurons and a different training process from the others and long short-term memory (LSTM) networks.

2.3 Supervised and unsupervised learning

An algorithm uses supervised learning when the learning process uses a dataset with diverse features and each learning example has an associated label.

With the training set of examples, another provided set of the same size contains the labeling of each example, deriving the term supervised learning.[12]

In the other case, unsupervised learning represents the cases where there is no given labeling of the training examples. The algorithm has to have a different understanding of the data without the extra information that guides the learning process.

2.4 Classification

Classification is a core subject in machine learning. It revolves around grouping together instances of data that have certain characteristics in common, saving their label and unique identifiable characteristics. In supervised learning, the final groups or classes are already labeled and the model learns from identified examples and applies what was learned to new data.

There are problems that do not feed labeled data, so instead of learning from an example sample, the model groups together data in clusters with similar characteristics that are finally labeled. This last case is considered unsupervised learning since no information is fed to the model.

There are several everyday examples of classification cases, from classifying someone eligible for a loan, separating spam from the rest of the emails and even in several medical fields.

The validation of classification problems can be accomplished with the split of data into training and testing sets, each one a percentage of the total dataset, among other processes, like cross-validation. This is used to have a set of data were the actual classification can be tested and a result can be evaluated.

That evaluation is accomplished with evaluation metrics like accuracy. Apart from accuracy, there are several other metrics like ROC curves and sensitivity, specificity, precision, that can be determined from a confusion matrix.

2.5 Multi-layer Perceptron

Multilayer perceptron (MLP) is a type of neural network that defines networks with multiple layers of perceptrons that are single neuron models that can lead to larger neural networks when combined, although the term is also used to characterize any feedforward neural network.

It is a mathematical function, composed by simpler functions each one presenting the output

in a new representation, that maps inputs to outputs.[12]

[1]The hidden nodes from the hidden layers transform the previous values from the previous layer using a weighted linear summation. This is followed by non-linear activation functions and the network then may use backpropagation, supervised learning, passing the outputs from one layer as inputs to the following layer, until it reaches the output layer and produces a final result. A network with three layers is usually called shallow or non-deep neural network, since the connection between neurons in each layer is more rudimentary.

The learning process takes place in the adjust of weights after each data processing, by comparing the output to the intended result, making the learning process supervised. MLPs are capable to learn non-linear models using partial fit but are sensitive to feature scaling since each neuron is connected to every other neuron and the number of total parameters can grow increasingly high, causing some redundancy. They have a non-convex loss function when they have hidden layers, consequently having multiple local minimums that can lead to different validation accuracy when different weights are used.

2.6 Convolutional Neural Network

Convolutional neural network (CNN), also known as ConvNet, is an architecture of neural networks mostly known for its applications on image and video recognition, analysis and classification, language processing and recommender systems and are specialized for data processing with a grid-like topology.

The architecture has shared-weights, translation invariance characteristics and implements convolution, a mathematical linear operation instead of a matrix multiplication that consists in the use of multiple filters in order to extract features from the data, preserving their spatial information in the process. CNNs take advantage of smaller patterns and evolve them to more complex patters, taking advantage of the hierarchical component of the data, having lower connectivity and complexity between nodes, unlike MLP, that are usually fully connected networks meaning that each node in one layer is fully connected to all the nodes in the next layers, which can lead to overfitting and posterior regularization.[12]

A key advantage to this architecture is the little need for pre-processing of the data compared to other architectures or even other algorithms, at least in simple cases like image related problems, where the type of data is very linear. Component-wise, CNNs consist of a sequence of convolutional layers with pooling, fully connected and normalization layers.

Each convolutional neuron only processes data for the corresponding receptive field, making a difference between fully connected models that would take much time and resources to accomplish similar results. Related to convolutional layers, there is also local or global pooling layers which reduce data dimensions by combining outputs from clusters of nodes from one layer, into a single

[1]https://scikit-learn.org/stable/modules/neural_networks_supervised.html

node in the next layer. The difference between local or global pooling is that the first usually clusters nodes with shapes of $2x2$ while the other acts on the entire convolutional layer.

In a convolutional layer, the neuron receives input from a subset of neurons, usually with a squared shape of NxN of the previous layer, not being fully connected. This way, the receptive field, or the input area, is not the entire previous layer but only a sub-part of it. The deeper into the network, the larger the receptive area gets, since there is an overlap of convolution applications.

There is discrete and continuous convolution, represented in the respective order by 2.1 and 2.2.

$$(f * g)(k) = h(k) = \sum_{j=0}^{k} f(j) \cdot g(k-j) \tag{2.1}$$

$$(f * g)(x) = h(x) = \int_{-\infty}^{\infty} f(u) \cdot g(x-u)\, du \tag{2.2}$$

The distinction between them is that

The convolutional operation is applied on two functions and is defined as the integral of their product and is usually presented with an asterisk, like in equation 2.4.

$$s(t) = \int x(a)w(t-a)da \tag{2.3}$$

$$s(t) = (x * w)(t) \tag{2.4}$$

2.7 Recurrent Neural Network and LSTM networks

[2]Another class of ANNs is recurrent neural networks (RNN) commonly used and computationally fitted for processing sequential data. In this type of network architecture the connections between nodes form a directed graph, allowing the demonstration of a dynamic temporal sequence.

This network type is an adaptation of feedforward neural networks, since they can use internal memory to compute and process data proprieties. Within RNNs there are networks with finite or infinite impulse, the first being a directed acyclic graph, possibly transformed into a normal feedforward network and the second being a directed cyclic graph where such transformation is not possible. Usually in neural networks, inputs and outputs are independent but there are cases where the the problem needs to keep track of the previous input in order to determine the next output, so RNN came into use, solving that problem.

[2]https://www.sciencedirect.com/topics/engineering/recurrent-neural-network

[3]Some of RNN's proprieties are the processing capability of an input of any length and having a model that does not increase with the size of said input and shares the determined weights throughout time. The used RNN in this study was a long short-term memory network (LSTM) which prevents errors from backpropagation like the vanishing gradient problem. This problem derives from the number of layers and consequent activation functions, where the gradients of the loss function move towards zero and make the training process more difficult. Since LSTM has a memory component it can learn from specific related events that are separated by large delays, learning from events that happened hundreds or millions of steps earlier and even flow errors between virtual layers which are unfolded in space.

2.8 Support Vector Machine

Support vector machines, also known as SVMs are machine learning models useful in classification and regression tasks. They can be supervised, where labeled data is used to train the algorithm or unsupervised thanks to support vector clustering which is the closest application in unsupervised learning.[23]

[4]Initially used in classification problems between two categories, the main objective of the algorithm is to create a space representation of the data and a division or a clear space between them, ranging from smaller to wider gaps. Such division is represented by the construction of a hyperplane capable of creating an optimal distinction or separation between each class label and allowing the fitting of new data into one of the classes.

[5]The separation created by the hyperplane is more efficient if the distance between instances of each class if bigger, in other words, if the space between them is maximum, avoiding the risk of misclassifying instances and having more confidence in the classification, since the hyperplane is a boundary responsible for the decision. Data points closer to the hyperplane define the optimal position and orientation since they are the closest instances belonging to different classes. These points help maximize the length of the classifier margin.

Some advantages of the use of SVMs are the high dimensional space effectiveness efficiency since it uses a subset of training points in the decision function. The loss function that helps maximize the margin is hinge loss and SVM is basically optimizing hinge loss with L2 regularization.

For an actual class t labeled as -1 or 1 and a predictive class y, the hinge loss function for binary classification of the prediction of y, follows the equation:

$$l(y) = max(0, 1 - t * y)$$

[3]https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
[4]https://scikit-learn.org/stable/modules/svm.html
[5]https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

The adaptation for multi-class classification also brought some adaptations to the hinge loss function like the definition of Crammer and Singer, where t is the target label and

$$w_t$$

and

$$w_y$$

the model's parameters:

$$l(y) = max(0, 1 + max_{y \neq t} w_y * x - w_t * x)$$

As previously mentioned, SVMs were originally designed to distinguish between two classes in classification problems. However, methods like *SVC* and *NuSVC* create the possibility of multi-class classification. *LinearSVC* also allows multi-class classification and is a faster implementation but only allows a linear kernel, not allowing any other kernel as an argument.[6]

The used method *SVC* takes in two arrays just like other algorithms, one for the training samples and other for the corresponding labels. It implements the "one-versus-one" approach for this type of classification having a total of $\#classes * (\#classes - 1)/2$ constructed classifiers, comparing two classes a the time. Figure 2.2 shows the classification of multiple classes using SVC with distinct kernels.

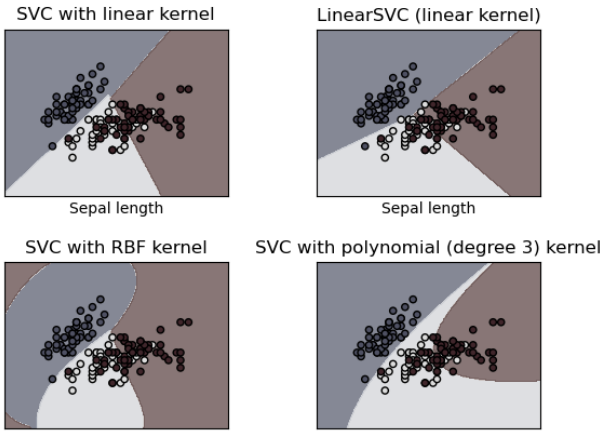

Figure 2.2: Multi-class classification using SVC with different kernels Source: https://scikit-learn.org/stable/modules/svm.html

[6]https://scikit-learn.org/stable/modules/svm.html

2.9 Logistic Regression

Logistic regression (LR) is a statistical based model that uses, as the name suggests, a logistic function. It is used in cases where the labeling target or variable is categorical with two possible values, like in classification problems where there are two target classes, meaning that in a binary model, the target variable has two categories, so outputs with more than two classes are modeled by multinomial LR.

[7]It is a predictive analysis algorithm, used not only to describe but also to explain the relation between data and variables.

Apart from binary problems, there are also multinomial and ordinal cases where there are multiple categories, where multinomial cases do not involve ordering of said classes, where ordinal cases do, like the rating of a service from 0 to 10. Finally, there are also decision boundaries where the model predicts to which class a certain data collection belongs to.

[8]The function behind this method is the logistic function, also known as sigmoid function. The curve takes the shape of an S, and can map any real value into a value constricted between 0 and 1. The sigmoid function follows the equation:

$$h_\theta(x) = \frac{1}{1 + e^{-x}}$$

LR predicts an outcome by finding an equation for a binary labeling variable, from one or several other variables. Those values can be categorical or continuous since it doesn't require a specific type.

In cases where there is a classification of data in two or more groups, it uses the *log odds* ratio, rather than probabilities and an iterative maximum likelihood method rather than a least squares to fit the final model. That ratio is represented by:

$$LogOdds = log[p/(1-p)]$$

Since LR is used primarily to describe data and relation between variables, the implementation of the algorithm in multi-classification problems came with some changes. One of those changes was the substitution of the logistic function for the softmax function. This new function compacts all values between 0 and 1 and their sum equals 1 and is represented by:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

[7]https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc/
[8]https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Along with cross entropy to measure how the probability distributions differ from each other, the class score and training loss calculations, and the one-vs-all or one-vs-one methods, LR multi-class classification is possible.

In the one-vs-all method, binary classifiers for each class are trained and tested on new data, while in the one-vs-one method, two classes are tested at a time, like previously described in the SVC algorithm.

2.10 Random Forest

[9]Random forest (RF), also known as random decision forests are a learning method used in classification, regression, among other problems.

Decision trees are a technique for predictive models, that follow the structure of a tree, branching out in different "if, then" scenarios, depending on the data's variables. The last split would represent the leafs of the tree, classes or final predictions of the model. Each branch determines the best split, using metrics like *gini* impurity, information gain and variance reduction, depending on the problem's context.

RFs are the construct of several decision trees with the intent of reducing the variance, at the cost of increasing bias and some loss they provide a final class or mean prediction.

This machine learning algorithm corrects the disadvantage of overfitting the training set, commonly found in decision trees, although there are cases that can lead to this problem, like the depth of the model, that can learn irregular patterns. Also, decision trees are not the most robust and have some difficulty in generalizing data, so RF were created to solve these problems.

[10]Using bootstrap aggregation, the training portion is conducted on random subsets of the data leading to a decrease of variance of the model, since bootstrapping indicates that individual trees are paralleled trained on several different subsets of the original training data and different features, making each tree unique. As long as the decision trees are not related, using this aggregation will make the final model more robust and leave the bias unchanged.

At each split along the tree, only a portion of features is considered, reducing correlation and the impact of strong predictor variables. Cross-validation is not needed in RF since each tree is created using a distinct bootstrap sample of the whole data.

RFs also use proximity, where after a tree is fully created, if two cases are in the same leaf node, they increase their proximity by one, latter normalizing the values by dividing by the total number of trees, aggregating instances of data.

[9]https://www.sciencedirect.com/topics/engineering/random-forest
[10]https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

2.11 Accuracy, Sensitivity, Specificity and other metrics

Accuracy is an evaluation metric that measures the proportion of true results, a combination of true positives and true negatives. It is a common metric in several fields since it is the most understandable interpretation of results. A high accuracy value may represent a good classifying model, capable of distinguishing several classes, but it is not always the best evaluation metric since those classes may not be classified with said accuracy, equally.

That is a potential problem when classes are not equally present and one may be extremely well classified and others, less represented, have poor accuracy rate, leading to a miss-interpretation of the value.

In a general case where there are a percentage of true positives, true negatives, false positives and false negatives, also known as TP, TN, FP and FN respectively, the instances where the the label is positive and the final output is also positive constitute the TP value. The FP represent the positive labels that have been classified as negative. The same applies to the TN and FN where the first represent the negative cases that have been classified as such and the second the cases there the negative labels have been classified as positive. Each one of these measurements has it's own meaning and value but each situation may be more impacted by one of them. In medicine for example, false positives can lead to improper treatment and false negatives can be fatal. Accuracy is calculated, using the previous indicators, by:

$$Accuracy = (TN + TP)/(TN + TP + FN + FP)$$

In cases where accuracy is not a reliable measurement, because of unbalanced data or in cases where that is not the intended information to be extracted, there are other metrics that can evaluate algorithms and computational models.

Apart from accuracy, this study uses sensitivity, specificity and area under the ROC curve, AUC. These metrics evaluate the ratio of hits and misses that the model or algorithm executes, testing them in a testing set and verifying the results.

Sensitivity, also known as true positive rate, is used to evaluate a test's ability regarding positive cases and how accurate they are identified. It represents a model's capability of detecting a positive instance, for example of a medical condition while the patient is positive for said condition. Higher sensitivity values mean that the model can accurately classify positive cases, so when it is important to know with certainty, that it is in fact a positive instance, sensitivity is the most important value to study. It is calculated by:

$$Sensitivity = TP/(TP + FN)$$

Specificity or true negative rate, on the other hand, is the measurement of correctly identified negative cases and represents how well a model can identify normal or negative cases. It is calculated by:

$$Specificity = TN/(TN + FP)$$

Sensitivity and specificity are not equally related. A model can have high sensitivity and not be so specific. The opposite situation is also possible, but both metrics are important in their own way and may apply better in certain situations. The receiver operating characteristic curve or ROC curve, is the graphic translation of the relation between sensitivity and specificity. In fact, it related the true positive and false positive rates, TPR and FPR. The TPR is equal to sensitivity and the FPR is equivalent to the inverse of specificity, meaning (1 - specificity).

The ROC curve is a product of all combinations of TPR and FPR and each point represents the trade-off between sensitivity and specificity. The x=y axis represents the random guess and a good result should present itself as a curve above that axis.[30]

2.12 Other Basic Concepts

Backpropagation is an ANN training algorithm used in feed forward neural networks that sends information back to the network and trains the weights and biases of neurons and connections.

Activation is the numeric value that enters the neurons and is used in the calculation of the output.

Activation function is the function responsible for obtaining the output of each neuron in the network to be passed to the following neuron.

Overfitting is when a machine learning model does not generalize correct values from the training data to unseen data. It can be manifested by lower validation or testing evaluation results when compared to training.

Chapter 3

Related Work

3.1 State-of-the-art

The state-of-the-art research consisted of searching for similar studies in order to know what is yet to be studied, what progress was already accomplished and what can be improved.

In this case, the state-of-the-art research is focused on ANNs comparative performance, meaning the comparison of their performance in different situations or with other methods under the same conditions, not the performance set on a unique dataset with unique features.

Most found studies do not really get into said comparison with the objective of segregating the situations with good and bad outcomes. In the following subsections, different search queries and several studies will be presented, demonstrating similarities and core differences with this study.

The structure of this section starts with a specific query, related studies, articles that do not have a relation to this thesis but present a reason behind why other studies were not chosen and finally, a brief discussion on why there is a need for this study, comparing to the ones previously presented.

3.1.1 Search's methodology, queries and results

The state-of-the-art research was conducted with the search engines *Google Scholar* and *ResearchGate*. Distinct queries were ran in those platforms and the articles presented in the next section were found.

The bibliographic references are organized by query in order to separate comparative components. Within the researched query, the analyzed articles and studies were organized by publishing date, from the oldest to the most recent study.

Although the studies are not related, this organization may create some sequence of used

tools, methods or software.

Table 5.4 presents the queries, the number of returned results and the found articles.

Query	Number of results
"data mining method", "comparison", "artificial neural network"	1120
"artificial neural network applications" (since 2016)	528
"artificial neural network techniques", "comparison"	937
"artificial neural networks", "size", "performance" (since 2019)	23500

Table 3.1: State-of-the-art queries and returned number of results

3.1.2 Articles review and explanation

3.1.2.1 Searched query: data mining methods, comparison, artificial neural network

Delen, Walker and Kadam's article is a study based on the prediction of breast cancer survivability[7]. It applies ANNs, decision trees and logistic regression in order to develop prediction models. These models were selected in this study due to their popularity around the publishing date of the article.

They use a large available dataset named SEER Cancer Incidence Public-Use Database between the years 1973 to 2000 with more than 400,000 cases. It consists of nine text files containing cancer related data for specific anatomical sites. Each file contains 72 variables and each one of the 433,262 records relates to a specific incident of cancer and a specific patient. The survival variable of each record is represented in binary, typical of classification problems with only two classes. For comparison purposes, they used 10-fold cross-validation methods in order to measure the unbiased estimate of the different prediction models.

In ANNs it was used a MLP commonly used in classification problems, with back-propagation. This study also proved the overall performance between MLP and RBF, RNNs and self-organizing map (SOM) in classification problems like the one is the study. In the decision tree model, it was chosen to use C5 algorithm, an improved version of C4.5 and ID3 algorithms. For measuring the performance it was used accuracy, sensitivity and specificity and, as previously mentioned, in order to minimize bias associated with the random sampling, it was used 10-Fold cross validation.

The results showed that decision tree (C5) is the best predictive model with an accuracy of 91.2%, followed by ANNs with 91.2% and finally Logistic Regression with 89.2%.

Soo Kim's study compares two of the main components we want to study[14]. It compares the performance of ANN, decision trees and linear regression based on the number and types of independent variables and sample size. There were generated 60 simulated prediction problems, with one, three or five independent variables and sample sizes of 100, 500, 1000 and 10000. Some

of the continuous variables were converted into categorical variables further down the study.

ANN, decision trees and logistical regression techniques were applied to the 60 examples in order to determine their prediction accuracy, splitting the set into a training set of 70% and a test set with the remaining 30% and it was only considered the root mean square error of the test set.

The used ANN was a multilayer feed-forward network trained using a backpropagation algorithm, with one or two hidden layers and with a variable number of hidden neurons, between one and ten. The learning rate and momentum were set at 0.1 and 0.9 to differ between a continuous descent on the error surface and a faster training process.

There were generated four decision trees where the splitting criteria varied, using minimum number of observations in a leaf and observations required for a split search for pre-pruning.

Logistic regression showed to be superior to decision trees and ANNs when independent variables are continuous for all number of variables. It also performs best when independent variables are continuous and categorical and the number of categorical variables is small. On the other hand, ANN is better when the number of categorical variables is superior to two. ANN performance also improves as the number of classes of categorical variables increases. When it comes to data sample, the logistic regression model performs better for small sets (100 and 500) while ANN performs the best with larger samples (1000 and 10000).

The next study revolves around the performance comparison between ANN and logistic regression models in predicting the mortality factor in elderly patients with fractured hips, within a year of surgery.[15] The data is from the National Insurance Program of Taiwan from 1996 to 2000 with a larger incident of mortality rates in females than in males.

The prediction accuracy was determined by randomly selecting patients from the dataset, previously to any exposition to the ANN and logistic models. Said accuracy is calculated with true and false positives and negatives. Also, continuous variables were presented as mean standard deviation, as for categorical variables they were presented as counts and percentages.

As a method to study the correlation of variables, the Chi-square test was used. Statistical analyses were conducted with the use of PASW Statistics 17.0 and Clementine 12.0 softwares with a significance level of 0.05. The comparison of ROC curves and ANN models in the testing and training sets were made using MedCalc 9.38.

The division of the original dataset was 70% for training and the remaining 30% for testing, having only the testing dataset to build models. Before modelling, the training sample had a boost of cases in order to balance the number of survival and non-survival cases. From the 12 variables, one was a binary variable to distinguish death from survival and the remaining 11 variables were selected as input variables.

From the six total models (two logistic regression models and four ANN models) the logistic regression models had a testing accuracy of 69.66% for only main effects and 71.91% for all

two-way interactions. The ANN models had 62.92% for 20 hidden neurons, 84.24% for 25 hidden neurons, 84.27% for 30 hidden neurons and 95.51% for an automatic selection.

They were compared for specificity and sensitivity and the ANN model had the best accuracy. The logistic regression models had lower areas under the ROC curves for both training and testing sets with 0.938 (95% CI: 0.904, 0.972) and 0.784 (95% CI:0.669, 0.899) respectively. Both values were lower than the ANN values of 0.998 (95% CI: 0.995,1.000) and 0.949 (95% CI: 0.857, 1.000).

The study from Maroco, Silva and Rodrigues focus on the prediction of dementia by comparing data mining methods and their accuracy, sensitivity, specificity, area under the ROC curve and Press'Q.[18] MLPs, RBF, SVMs, CART, CHAID and QUEST classification trees and random forests were compared to linear discriminant analysis, quadratic discriminant analysis and logistic regression.

The model predictors were ten neuropsychological tests, used in the diagnosis of the disease. The data of the subjects refers to 921 elderly non-demented patients recruited due to the cognitive complaints referred for neuropsychological evaluation at 3 different institutions, two in Lisbon and one in Coimbra, from 1999 to 2007.

Using the Friedman's nonparametric test, the statistical distributions of classification result from a 5-fold cross-validation were compared. 5-cross validation was also used in order to prevent overfitting and artificial accuracy improvement due to the use of the same dataset for the training and testing sets. With the use of Press'Q, the study determined that all classifiers performed better than chance ($p < 0.05$). The training sample was 80% of the original dataset and the remaining instances were used for testing.

The multilayer perceptron was trained with 11 inputs, 1 hidden layer with 4-7 neurons, iteratively adjusted and a hyperbolic tangent activation function. For the output layer, the activation function was the softmax with a cross-entropy error function.

The RBF had 11 inputs, one hidden layer with 2-8 neurons and a softmax activation function. The activation function for the output layer was the identity function with a sum of squares error function.

SVM presented the highest values in specificity followed by multilayer perceptron, logistic regression and RBF with significant differences, and with lower values, LDA, QDA, classification trees and random forest. LDA, CART, QUEST and random forest had the highest sensitivity values. Logistic regression, multilayer perceptron, RBF and CHAID had median sensitivity values close to or lower than 0.5, and SVM had the lowest sensitivity but the highest AUC.

Random forests and linear discriminant analysis had high accuracy, sensitivity, specificity and discriminant power. On the other hand, SVM, ANNs and classification trees presented low sensitivity.

In the final results, SVM had a larger classification accuracy (Median (Me) = 0.76) an area

under the ROC (Me = 0.90). They also showed high specificity (Me = 1.0) but low sensitivity (Me = 0.3). Random forest had the second best accuracy results (Me = 0.73) with high area under the ROC (Me = 0.73) specificity (Me = 0.73) and sensitivity (Me = 0.64). Linear discriminant analysis had a reasonable overall accuracy (Me = 0.66), with and area under the ROC (Me = 0.72) specificity (Me = 0.66) and sensitivity (Me = 0.64). The other classifiers showed overall accuracy above a median value of 0.63 but the sensitivity values were lower than the median value of 0.5.

The next article studies the comparison of ANN and logistic regression analysis in pregnancy prediction using the in vitro fertilization treatment.[19] The data is from 1995 patients of the Shore Institute for Reproductive Medicine, USA, in the age range of 21 to 45 years old.

Univariate and multivariate logistic regression models were performed using Stata/IC 12.1 software in order to provide pregnancy status. The data was also analyzed using ANN with application of the Statistica Data Miner + QC 10.0. ROC curves and area under the curve (AUC) were analyzed in order to determine the quality of the final predictors.

In ANNs, to create the best network, the algorithm was run 30000 times and determined a fit of three-layer perceptron with 40 neurons in the input layer, six in the hidden layer and two in the output layer. The sensitivity and specificity were 69.0% and 60.3%, respectively.

The predictive results of the two models showed that the ANN model obtained better results. The statistically significant differences between the predictive powers of both models were at the $p < 0.0001$ level. Also, the curve for the ANNs is more convex than the other model. ANN's AUC, sensitivity and specificity values were higher (10%, 3.5% and 3.5%, respectively) than for the multivariate logistic regression model.

In summary, ANN models were found to be better at obtaining a predictive model than classical statistical analysis. On the other hand, they are not able to detect which variable influences the most and how much, unlike logistic regression.

3.1.2.2 Query's articles discussion

The presented studies for this query revolve around the comparison between ANNs' performance and other methods in order to find a better accuracy in classification, among other evaluation metrics.

This is similar to what this thesis is trying to accomplish but it also wants to extend the comparative factors further than just data mining algorithms. It also wants to have other characteristics as factors so, keeping that in mind, although they are similar in that aspect, this is an improvement and more broad study.

For example, the first study is the most similar study to the work this thesis wants to accomplish by comparing ANNs with other methods under different networks types and data mining algorithms. Still it used the same dataset with the same type of data and the same size.

The results show that ANNs are slightly worse than decision trees but would that change in other dataset types or different dataset size?

3.1.2.3 Searched query: artificial neural network applications, since 2016

Abiodun, Jantan, Omolara, Dada, Mohamed and Arshad's article is more contextual than the other ones meaning that it is not specifically about the comparison between ANNs and other models but more of a study about ANNs applications in the real world.[2] It provides a taxonomy of ANNs and insight of the relation between those applications, data mining techniques and types of ANNs.

It highlights, discusses and compares more than eighty research articles on ANNs model applications based on author and year of publication, ANNs modeling, area of publication and contribution. It also shows the interesting increase in ANNs applications in the last two decades, having most research articles published after 2009.

The most relevant commentary is about the need of comparison between the researched studies about ANN's applications, and other methods in the same applications in order to prove and substantiate their success.

On the relation between ANNs' applications and data mining problems, the study concluded that among the several ANNs applications, most of them were related either with classification and pattern recognition. By analyzing more than eighty researches in real life applications, this study supports the motivation and need for this thesis' main question to be answered by referencing the lack of comparison between models.

From this search, only this study fitted the criteria for state-of-the-art. It had the objective of collecting data on the use of ANN during the last few years as well as any other information focusing on ANN application and not the actual implementation of ANN models in specific cases.

Most other returned studies, like the next examples presented in this section, focus on the application of ANN under a specific scenario and not the study of overall applications.

The study from William A. Borders is focused on characterization of spin−orbit torque−controlled synapse device for artificial neural network applications.[5] It was motivated by the rapid increase of AI research and the interest in fabricating machines and computing systems capable of performing high level tasks. The study reports one approach to hardware-based ANN using spintronic technology and explains the ANN-based associative memory operation procedure, using spin-orbit torque controlled devices. The devices reliability and efficiency challenges that were found are also described and clarified by new measurements of endurance properties. The results surround the limitation of the maximum operation current and the consequences on memory operations.

The next study example, from Fatehnia and Amirinia, is about review of genetic programming and ANN applications in pile foundations.[10] Pile foundations are structural elements used

in the transportation of superstructure loads deep into the ground. The basis of the study is the estimation of pile bearing capacity, keeping in mind that the interaction with soil is not completely understood, limiting the study. That missing interaction knowledge led to the review of genetic programming and ANN applications in order to solve the previous related problem. The article focus is in explaining the methods, provide literature review on the application of both methods and extract results from such application.

The final example of articles that did not fit in the state-of-the-art for this search is a 2019 study about ANN applications to pattern recognition.[1] It is motivated by the lack of answers to problems like whimsical orientation, object classification, location, scaling, neurons behaviour analysis in hidden layers, among others and gives an introduction to the more current trend in ANN models that also address pattern recognition challenges. In measuring the performance of ANN models, the study uses mean absolute percentage error, mean absolute error, root mean squared error and variance of absolute percentage error and concludes that current ANN models perform extremely well in pattern recognition tasks.

3.1.2.4 Query's articles discussion

The first article was more of an extra source of motivation for this thesis, since it encourages the need for it and is also a source of information about the professional fields of use of ANNs.

The rest of the studies presented in this subsection are more specific implementations of ANNs and do not represent the kind of study we are trying to accomplish. They all present unique applications with unique features and no comparative performance inclination, making them an example of the returned articles from this query that do not align with this study.

3.1.2.5 Searched query: artificial neural network techniques, comparison

The article entitled "River flow forecasting and estimation using different artificial neural network techniques" is a 2008 study about the application of ANNs' techniques for the estimation of monthly streamflows.[22] The studied ANNs' techniques, feed forward neural networks (FFNN), generalized regression neural networks (GRNN) and radial basis neural network were used in one-month ahead streamflow forecasting and monthly flow data from Gerdelli and Isakoy stations in Turkey and the used data was 39 years long dating from 1961 to 1999. Three program codes were written in Matlab language for the three simulations and the models were written in a training and a testing phase, being the first one around 75% (348 months) of the dataset and the second 25% (120 months). The FFNN was optimized with the Levenberg˘Marquardt (LM) optimization technique for adjusting the weights. The used network had one hidden layer and a common trial and error method to select the number of used hidden nodes. The activation function is the sigmoid function and it was used in the hidden and output layers. After the training phase, the weights are saved and used in the testing phase, validating the network performance on test data also having the mean square errors (MSE) computed for that data. The study also involved the

periodicity of the model's forecasting performance as well as the flow estimation using data from nearby rivers. Here, another variable was added into the input combinations. It is visible that the periodicity decreases the MSE for each model. The relevant results demonstrated that GRNN had better performance in one month ahead streamflow forecasting. The RBF performed better than FFNN in monthly flow forecasting and both the RBF and FFNN were better than the GRNN in monthly river flow estimation using nearby river data. The periodicity component also increases the performance for flow forecasting, but decreases in monthly river flow estimation.

Firat, Turan, and Yurdusev's study is focused on comparing ANN architectures in the prediction of water consumption, using GRNN, cascade correlation neural network (CCNN) and FFNN for such comparison.[11] The used data was collected from one hundred and eight datasets from Izmir, Turkey. For the prediction models, 108 monthly records of water consumption, from 1997 to 2005 were collected. The data was divided into training and testing sets, 80% and 20% respectively, and a trend component was removed using the regression line, in order to create the prediction model. The testing data was not used during the training process, providing a more reliable evaluation. In order to determine the best model, statistical criteria such as average absolute relative error (AARE), normalized root mean square error (NRMSE) and threshold statistic (TS) are calculated to perform the comparison and evaluation. The used ANN models were constructed depending of the combination of antecedent values. In total, six models were created, based on combinations of previous monthly water consumption. The M5 models, combined five antecedent values of water consumption performed better than other models, leading to higher NRMSE and AARE values. The results show that the performance of the M5 CCNN model is better and the test statistics are also slightly better than other models. The M5 CCNN model presents lower NRMSE and AARE values than GRNN and FFNN models. M5 CCNN also presents higher values of the CORR than the other models, demonstrating that CCNN was superior to FFNN and GRNN in the prediction of water consumption time series.

The study from Deme C. Abraham, compares ANN techniques for mobile networks field strength prediction.[3] It compares and analyses field strength prediction, MLP, RBF and GRNN. That analysis uses readings obtained at 1800MHz from Base Transceiver Stations, metropolis of Jos, Nigeria. A cellular mobile network analyzer (SAGEM OT 209), capable of measuring signal strength, received power measurements, used throughout the study. The data was separated into three groups, 60% training, 10% validation and 30% testing, ensuring optimal training performance in one approach. Then, in another approach, the GRNN was trained with a dataset from one base station and tested with a different set from another base station. The root mean squared error (RMSE) and the coefficient of determination (R2) were used as statistical performance indices. The study had a MLP with three hidden layer neurons, an error goal of 0.001 and the Levenberg-Marquardt back propagation as a training algorithm, an RBF with a spread of 0.8 and error goal of 0.1 and for the GRNN, it had a spread of 0.6. The comparison concluded that GRNN had the most accurate predictions for both used base stations' data, with an RMSE of 5.13dB. It had the least prediction error of 4.83dB according to the geometric mean and the best fit.

The article from Behbahani, Amiri, Imaninasab and Alizamir is about the comparison of four different ANN techniques in forecasting urban road network accident frequency and determine which of the techniques performs best.[4] The forecasting aims to determine how crashes in urban road networks are affected by external factors. The four ANN techniques are extreme learning machine (ELM), probabilistic neural network (PNN), RBF and MLP. These techniques were chosen because they are able to solve road accident forecasting problems, especially with complex variable interrelations. The dataset for the research is traffic and accident data from the city of Mashhad, Iran, in the year of 2014. The dataset has a total of 2338 accidents on 194 roadway segments and three sets of variables, crash, traffic and environmental characteristics. The data was divided into two groups. The first, the training set is 80% of the total observations and is used to calibrate the models and assess fitting. The other 20%, create the testing set and is used to compare the prediction results of all the models and compute error values. The study used measures like Nash-Sutcliffe (NS), mean absolute error (MAE) and root mean square error (RMSE) in order to evaluate performance from each technique. MATLAB 2010 software was used to perform the used techniques and the Relief algorithm used to find the significance of the used variables. It determined that VKT was the most influential variable, followed by two traffic flow characteristics, V/C and speed. In the modeling process, the used parameters were analyzed based on importance and predicting impact. Also, the designing network was expected to identify the most accurate structure using trial-and-error approach, since different network structures can lead to different results. The results from all models show that ELM is the best model to forecast accidents. The network had 15 neurons since it was the optimal number for the used data. Unlike other models, ELM only requires setting the number of hidden neurons and the activation function, avoiding the chance of a local minimum problem. It was the fastest approach in the training process, the most accurate in estimation accident frequency in both processes, followed by MLP, PNN and RBF. Next to ELM, MLP performed more accurately when faced with the available data but MLP models are vulnerable to noise, leading to wrong predictions. PNN outputs are easier to interpretation and PNN are faster than multilayer perceptron networks and the predictive results were fairly accurate. RBF presented the weakest results in comparison to the other techniques.

The previously presented articles in this section implement the comparison between three or more ANN techniques. Most of the returned studies by the query were based on similar characteristics but were more restricted when it comes to comparison and obtained results. For example, the study from Zare and Pourghasemi studies the comparison between only MLP and RBF in spatial prediction of landslide susceptibility mapping. [21] The dataset is composed of 136 landslide locations and nine conditioning factors. 70% of the data (95 landslide locations) in the training set and 30% (41 landslide locations) in the test set. The results concluded that MLP with Broyden˘Fletcher˘Goldfarb˘Shanno learning algorithm was more efficient than RBF.

Other than studies with less comparable techniques than the ones presented, other returned studies did not address the subject the searched query intended like the study from Erbek and Taberner that also focused the architecture comparison between only two techniques, MLP and learning vector quantization. [9] Also like the previous study, it revolved around land use

activities, and maximum likelihood classification with supervised ANN and the MLP produced the best results.

Sahoo and Jha's study compares the performance of multiple linear regression and ANN in groundwater-level prediction. [28] In this study, the used ANN is a multilayer feed-forward network with a single hidden layer. No other ANN was used in the comparison, unlike the previous presented articles, not addressing the comparison between ANN architectures.

Szuster, Chen and Borger's study compares SVMs with maximum likelihood classification and ANNs in land cover and land use analysis in tropical coastal zones.[29] The comparison is not between ANN techniques like the searched query intended, which is why it was not included in the retrieved studies.

3.1.2.6 Query's articles discussion

In the query from this section, the same problem described in the first query section occurs. The presented studies tackle the issue but are very restricted to that comparison and do not include other factors. In this case, the focus was on comparing the performance of several network types but they all used the same datasets, sizes and no different algorithms to compare overall performance.

Again, this thesis works as a improvement and compilation of works, having independent studies come together to find new results and conclusions.

3.1.2.7 Searched query: artificial neural networks, size, performance, since 2019

Due to the amount of results from the query, the search was limited from since 2019, providing recent studies with the recent technologies. It intends to show ANN's performance and how it is affected by the dataset size, comparing to other methods or by comparing several sizes from the same dataset. This will allow us to determine if and under what circumstances, size and data volume impacts performance, both time and memory related.

The preferred studies are the ones that on top of studying the effect of data sample size, also compare the ANN's performance under different sizes with other models, reaching into what this thesis is trying to accomplish.

The study from Chen and Folly focus wind power forecasting and the effect of input features on the performance.[6] The study only addresses ANN, not comparing results with other methods, but it analyzes the performance with some depth and detail. On top of studying the impact of input features it also studies the optimal sample size and the best performance interval values. Finally, it also addresses the trade-off between forecasting performance and computational cost. The meteorological dataset used is from the Wind Atlas of South Africa (WASA) and consists of wind related variables such as speed and direction, measured at different heights, with 10 minutes

resolution for the period between 31 December 2010, to 1 January 2017. In order to compare the performance of each model, performance evaluation metrics like the normalized root mean square error (NRMSE), normalized mean absolute error (NMAE), and the mean absolute percentage error (MAPE) were used. The performance, based on sample size, was meant to find the ideal size where the model could produce good forecasting results at the lowest computational cost. All the variables stayed the same except for the size of the training set and it concluded that the RMSE value stopped improving after a training sample of 20000, where it stagnated, although the forecasting performance improved with the increasing number of training samples. As for the computational time needed, it increased along with the training set size. It took 92 seconds to train the 20000 set and 166 seconds to train a training set of 60000. The study opted to use a training set of 20000 since the RMSE value for both sizes were basically equal. As for correlation coefficients between the target and input features showed that speed, direction and temperature had a positive correlation while temperature gradient, relative humidity and barometric pressure had a negative correlation.

The 2020 article based on groundwater potential mapping in mountain bedrock aquifers, studies the effect of sample size in different machine learning models.[**?**] It compares adaptive neuro-fuzzy inference system (ANFIS), ANFIS-imperial competitive algorithm (ANFISICA), alternating decision tree (ADT) and random forest to model groundwater potential. The ICA optimization algorithm was used so the best parameters could be found, avoiding amalgamation of the parameters. The dataset is a documented inventory of springs and was divided into four data samples with size of 100%, 75%, 50% and 25% of the total set, varying from 177 to 714 instances. Each dataset was then divided into training and testing sets, 70% and 30% respectively, with fifteen geo-environmental factors as independent variables. As metrics to evaluate the performance of each model, the area under the operation receiver characteristic curve (AUROC) and the true skill statistic (TSS) were implemented. The final results concluded that sample size affects the performance of all the implemented algorithms, with random forest having lower sensitivity to smaller sample sizes. ADT and ANFIS's performance decreased with sample size while the hybrid ANFIS-ICA and random forest had better performances for all dataset sizes. Based on validation results, random forest had the best performance on all four datasets based on the tested metrics, in both training and validation phases, followed by ANFIS-ICA, ADT and finally ANFIS, proving that random forests should continue to be used.

The study from D'souza, Huang and Fang-Cheng relates the connection of dataset size with network structures, both important factors in this study since they are comparative points in performance analysis.[26] It studies the performance of ANNs in cases with small datasets and whether the chosen network structure has an impact on the final performance and whether the optimal network structure is determined by the size of the chosen dataset. All layer combinations were listed and given an upper bound of the Vapnik−Chervonenki (VC) dimension in order to study the variance of the performance caused by structural hyperparameters. A list of all possible structures with fixed dimensionality of layers was created and given a constraint, based on the network's VC dimension. All networks were trained and tested with a held−out validation, recording their accuracy. After said recording, five optimal performing networks were selected.

There were used three datasets, the MNIST handwritten digit recognition image dataset, which contains 60000 training images and 10000 testing images. There was a random selection of 100, 500 and 1000 samples from the 60000. The 1000 samples were divided into 800 and 200, the 500 samples into 400 and 100 and finally, the 100 samples were divided into 60 and 40, each one for training and validation respectably. Another dataset used was CIFAR−10, with 60000, $32x32$ color images (RGB) in 10 classes, with 6,000 images per class. From the dataset, 5000 samples were used, from which 4000 were for training and the remaining for testing. The same optimizer settings used in the MNIST dataset were used for this dataset. The final dataset is from the Tumor Proliferation Assessment Challenge of 2016 and consists of 73 breast cancer cases. The results showed that structural optimization led to an improvement in accuracy by 27.99%, 16.44%, and 13.11% over random selection for sample sizes of 100, 500, and 1,000 in the MNIST dataset. Those values lead to the conclusion that the optimization of the network is of the most importance when the dataset size becomes smaller. On another important note, the optimal network structure was determined mostly due to data type (photo, calligraphic, images) and less due to sample size, suggesting that the optimal structure is data-driven. After the structure optimization, the CNN achieved 91.13% accuracy with 500 samples, 93.66% for 1000 samples and finally, 94.10% for 3300 samples.

The final reviewed article, from S. Markham and Rakes, titled "The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression" is an older study from 1998, found in the same query but without the time restriction, about the effect of dataset size and variability on ANNs and linear regression performances.[17] Too small sample sizes lead to inadequate error measures in regression and large sample sizes incur higher costs in data collection. The research explores the behaviour of simple linear regression and ANNs, by varying the sample size and variance of the error term in a predictive problem. Such comparison is made with the use of the principal of mean square error (MSE) and the use of root mean square difference (RMS) to have the error and the observations in the same magnitude. As for ANNs, it was used a three-layer (1 hidden layer) feedforward backpropagation network. It had a supervised training and the NeuralWorks Professional II was the used implementation software. Steps were taken in order to determine the best network. Several ANNs were studied, each one with an input and output processing elements and one or two hidden layers. For each combination, the sigmoid, sine and hyperbolic tangent transfer functions were considered. By having the lowest average RMS, a three-layer backpropagation network with two processing elements in the hidden layer, one processing element each in the input and output layers, and a sigmoidal transfer function composed the selected network. The dataset was generated from the Statistical Analysis System (SAS), so instead of using real world data, the generated data allows the researchers to know the true values of the parameters. Arranged in a five by four factorial design, since the study intends to compare variance and sample sizes as factors, samples sizes of 20, 50, 100, 200 and 500 were used as factor levels. From comprised populations of 10.000 pairs, representing a variance level, nine datasets were generated for each of the sample sizes. For each sample size and variance combination, five sets of data were taken as training sets and four other distinct sets were taken from each population and used as recall sets. The results comparing both

methods were determined by subtracting ANN's RMS value with the RMS value from regression for each observation. If the result is negative it means that ANN's RMS value is smaller and therefore is better at predicting and the opposite indicates that regression is the better model. The data analysis was conducted with the use of nonparametric statistical package developed by Walter R. Pirie. The results showed that the regression based model had better results with lower variance values while ANNs had better results for higher values. When it comes to sample size, for medium variance values, it was concluded that regression had better performance in smaller samples while ANNs improved results for higher dataset samples.

3.1.2.8 Query's articles discussion

The same problems found in the previous articles were found in this research. There was a comparison in order to find the best ANN architecture but that was not one of the main topics of comparison and was not a present factor in the comparison of both variance and sample size.

Comparing this study with the objective of this thesis, this comparison is applied to the same dataset type, same predictive problem and even thought the best architecture was determined, it was a comparing factor.

What this thesis is trying to accomplish is what is missing in this other studies, the relation between several factors that may influence the performance of the models, so in this case the permutation of comparative factors is intended to be bigger, or in other words, include more comparative factors.

Chapter 4

Practical methodology

4.1 Practical methodology description

As mentioned before in a previous section, the objective is to compare the performance based on distinct evaluation metrics, in distinct comparative situations, so the changing factors had to be decided before implementing any models or algorithms.

Data type can be an influence factor in the performance so different datasets with different data types such as text, image, sound and categorical data were chosen. These are diverse types of data that tackle several problems and applications in social activities. In total, eight datasets were studied, two for each type, ensuring some data diversity not basing all results and leading conclusions on only one dataset. Different types of data are represented in different ways, so the interpretation of said data by an algorithm could be executed differently and could lead to different results.

For ANNs, the architecture of the network may have an influence on the final performance since some networks are known to favor some types of data. With that in mind, a convolutional neural network (CNN), a recurrent neural network (RNN) and a multilayer perceptron (MLP) were implemented for each dataset. Convolutional networks are known to obtain good result with image recognition but this study is an opportunity to study architecture's performance in other data types and shapes. These types of network were chosen because they operate in different basis and are known to have good performances in specific situations, for example, CNN are known to be good with image related problems and RNN with sequence data and natural language processing. Furthermore, the addition of more architectures would lead to an even more extensive study and the amount of time would not be sufficient.

[1]Within RNN the classic LSTM network, being a specific RNN, was the chosen to be implemented in all models, since it deals with the vanishing gradient problem, previously described. It was designed to model temporal sequences so it is a great model to compare since

[1]https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

some of the problems present in our datasets have a temporal dependency.

The chosen datasets have to have an associated problem in need of a solution and since there must be a neutral factor between every dataset, all datasets are classification related. Otherwise, if we had two different problems and we studied the impact of the network type, resulting in different results, we couldn't conclude if the reason was the type of problem or the network.

The size of the dataset must be variable in order to study the performance under different sample sizes, since that can lead to a different performance because some data types or the implemented algorithms may be more adaptable to smaller or larger data samples. Most datasets had three different data sample sizes. Around 2,000 instances were saved from the overall training set for validation, so the maximum dataset size is usually 2,000 less instances of the total size. The medium size is roughly half the size of the maximum dataset and the minimum data sample size is 2,000 for all datasets, apart from the sound datasets that are the smaller datasets of the whole group of data.

The larger and the smaller sizes are used to evaluate the impact of data size in the learning process and posterior classification. The medium data sample size was implemented with the intent of guaranteeing that the classes and overall data was correctly shuffled. This means that the difference between the maximum and medium data sample sizes should not be considerably different, since that would mean that the removed piece of data was more impactful to the learning process that should actually be.

After having all those parameters and factors, a comparative algorithm also capable of solving that problem and obtaining good results must be chosen. If different algorithms were chosen to be compared in certain situations, no real conclusions could be made, so instead, support vector machines (SVMs), logistic regression (LR) and random forest (RF) models were used, also for each of the eight datasets. These algorithms were chosen because of their overall good classification performance, so they are compared to the implemented ANNs.

Since the performance is the main focus of comparison, in order to evaluate the performance and correctly interpret the results, different metrics were implemented, since all the datasets are different. Some issues like balanced and unbalanced datasets could make evaluation metrics like accuracy an unreliable source for conclusions, so the chosen metrics were based on the articles from the state-of-the-art research and the metrics they used. Accuracy, sensitivity, specificity and area under the receiver operating characteristic (ROC) curve (AUC) were implemented in order to maintain a wide range of interpretation of the results.

4.2 Datasets

4.2.1 Image datasets

The MNIST dataset is a collection of grey-scale handwritten digits, from 0 to 9, for classification purposes. It is a pretty much balanced dataset with 60,000 training examples and 10,000 testing examples. Of the 60,000 training examples, 58,000 were used for training and 2,000 were used as validation.

It is a subset of the NIST special database 3 and special database 1, a larger database, but since the models take time to train and extract the final results and there are smaller datasets to be compared in the study, the MNIST dataset was a suitable choice. It is composed of 30,000 images from SD-3 and the remaining 30,000 from SD-1 with approximately 250 different writers.

All the images have been normalized when it comes to size and are centered in a fixed size of $28x28$ pixels. The class distribution of the MNIST dataset, for all computed sizes of 2,000, 28,000 and 58,000 instances, is presented from Figure 4.1 to Figure 4.3.

The CIFAR-10 (Canadian Institute For Advanced Research) is a subset of the 80 million tiny images dataset, collected by Alex Krizhevsky, Geoffrey Hinton and Vinod Nair. It consists of 60,000 colour images, with dimensions of $32x32$ pixels, labeled in 10 balanced classes, having 6,000 images per class. There are 50,000 images for training and 10,000 for testing. Of the 50,000 training images, 48000 were used for training and 2,000 for validation.

The ten classes are images of airplanes, automobiles, birds, dogs, cats, deer, frogs, horses, ships and trucks. All the classes are mutually exclusive having automobiles include cars, SUVs, etc, and trucks include only bigger vehicles.

The labels are encoded with numbers from 0 to 9, each number corresponding to an individual class. The actual data is separated by the RGB colors, each one having 1024 (32x32) bytes.

The class distribution of the CIFAR10 dataset, for all computed sizes of 2,000, 28,000 and 48,000 instances, is presented from Figure 4.4 to Figure 4.6.

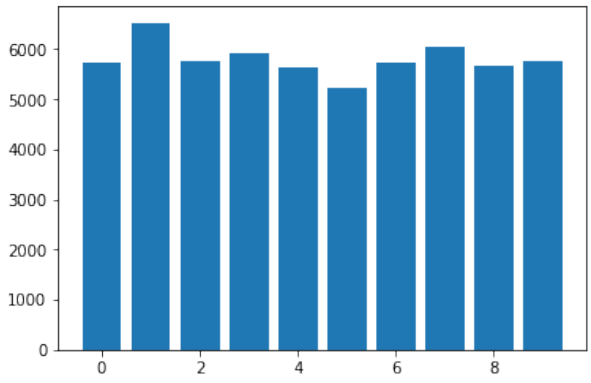

Figure 4.1: MNIST dataset 58,000 instances class data distribution

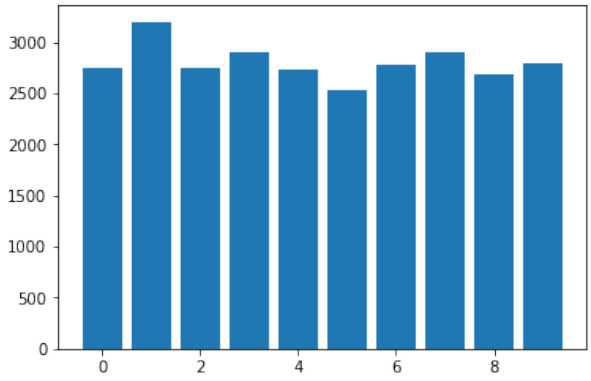

Figure 4.2: MNIST dataset 28,000 instances class data distribution

4.2.2 Sound datasets

The heartbeat sound dataset was originally meant for a classifying problem of determining the type of heartbeat of a patient. The data was collected from the general public by the *iStethoscope Pro iPhone* app and from clinical trials in hospitals using the digital stethoscope *DigiScope*. There are three classes but two of them were joined into a single class, having a total of two classes, normal or abnormal heartbeats.

With a total of 461 files, the data was stretched by dividing the audio files into smaller files, resulting in a total of 1037 audio samples. The dataset is not particularly balanced since there are around 300 instances of abnormal heartbeats an more than 500 normal heartbeats, a problem solved with the use of *RandomOverSampler*.

The length of the audio files varies between 1 and 30 seconds and some of them are already clipped in order to reduce excessive noise, providing a clearer fragment of the sound.

Nevertheless, the recordings may contain noise corresponding to the removal of the device, breathing, brushing and general background noise, but a normal heartbeat sound has a clear pattern of “lub dub, lub dub”, with the time between “lub” and “dub” being shorter than the time between “dub” and the next “lub”. The abnormal heartbeats do not follow the same

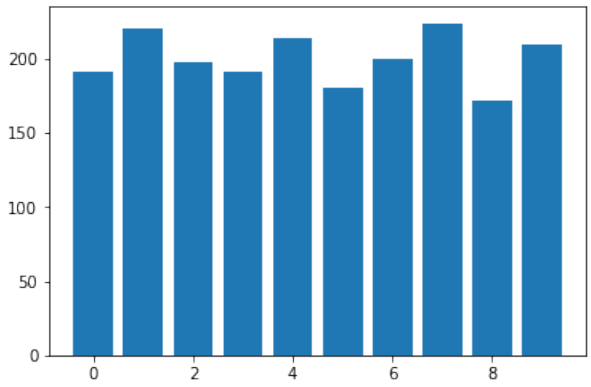

Figure 4.3: MNIST dataset 2,000 instances class data distribution

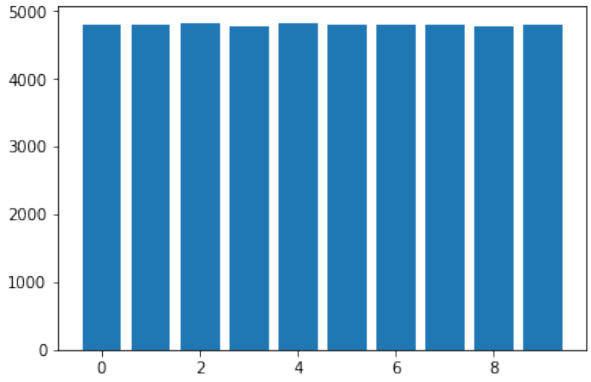

Figure 4.4: CIFAR10 dataset 48,000 instances class data distribution

pattern, having some other noise between the expected pattern. In this dataset, audios with a substantial amount of background noise or distortion were used in order to have a perception of the adaptability of the models to unfavorable data.

The class distribution of the heartbeat sound dataset, for all computed sizes of 250, 528 and 829 instances, obtained with the use of *RandomOverSampler* explained in the *Pre-processing* section, is presented from Figure 4.7 to Figure 4.9.

The ESC-50 dataset or Environmental Sound Classification 50 is a sound event data collection with 50 different and balanced classes. The dataset has a total of 2,000 five second recordings, 40 per each class, in *.WAV* files, sampled at 16KHz. It comes divided into 5 folds, 4 of which make the training set and the remaining the testing set.

Since all the files have, approximately, the same length is expected to have more silence within the audio file, since most types of noise are spontaneous, like barking of a dog, a sneeze or raindrops. This way, is expected for the models to learn the patterns even with silence breaks between noises, since each bark audio, for example, may have different silence duration between the actual noise but the frequency and the tempo is similar.

In this thesis only half of the dataset was used, since the previous dataset only had two classes,

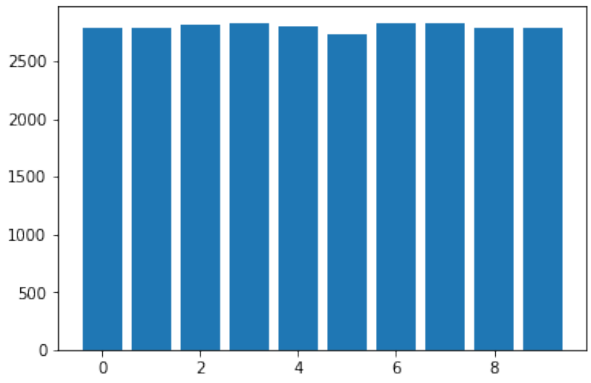

Figure 4.5: CIFAR10 dataset 28,000 instances class data distribution

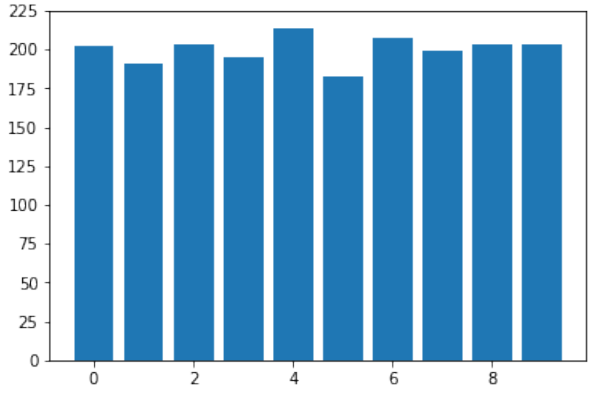

Figure 4.6: CIFAR10 dataset 2,000 instances class data distribution

fifty was more than what was needed to have some variety of data. But since the heartbeat sound dataset had a total of 1,037 audio samples, a similar size of data was needed, and half of the dataset was enough for such similarity.

The class distribution of the ESC-50 dataset was only tested for one size sample since the total size of the dataset was not sufficient to split the data. Each class was represented by approximately 60 to 70 data instances, after the audio files partition, so reducing the data collection would result in unrealistic results. The distribution is presented in Figure 4.10 where the data distribution does not follow the 40 recordings per class because of the referenced split of each audio in several sub-recordings, explained in the *Pre-processing* section.

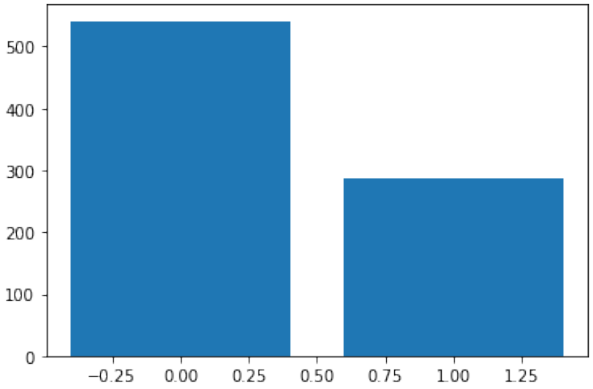

Figure 4.7: Heartbeat sound dataset 829 instances class data distribution

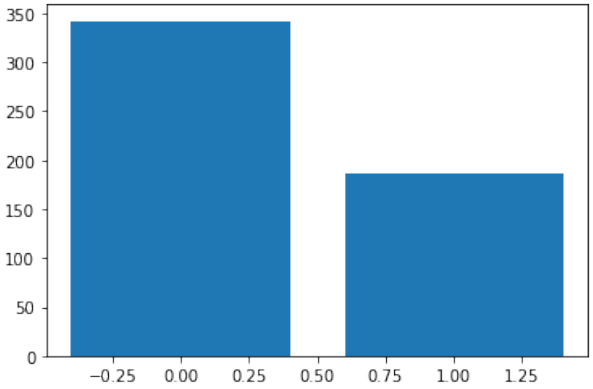

Figure 4.8: Heartbeat sound dataset 528 instances class data distribution

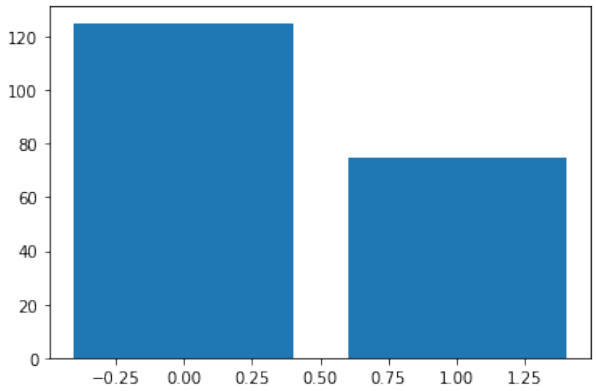

Figure 4.9: Heartbeat sound dataset 200 class data distribution

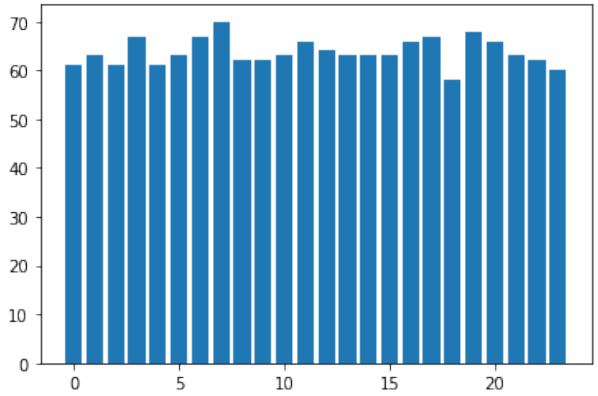

Figure 4.10: ESC-50 dataset class data distribution

4.2.3 Text datasets

The IMDb movie reviews[16], is a binary classification dataset which contains more data than previous benchmark datasets. With a total of 50,000 reviews for natural language processing and text analysis, it is a tool for sentiment analysis. Of the 50,000 reviews, 30% constitute the training set and of the remaining 35,000, 2,000 were used for validation.

It is a balanced dataset, with each class having 25,000 instances, with reviews ranging from 3 to 1,398 words, 17 to 9,080 characters, and approximately 100,000 different words overall.

The reviews are mostly well-structured, unlike the next dataset, which makes the learning process easier. Nevertheless, the data has to be submitted to the usual pre-processing techniques in order to achieve uniformed data and remove words that do not help the learning process.

The data distribution of the IMDb dataset is represented from Figure 4.11 to Figure 4.13, representing each size of 33,000, 15,000 and 2,000 reviews, respectively.

The final data is the Sentiment140 dataset also used for sentiment analysis. It has 1,600,000 tweets extracted using the twitter api, and were assigned two labels, 0 and 4, corresponding to negative and positive sentiment respectively. Apart from the target which represents the polarity of the tweet, there is also the id of the tweet, the data of publication, a variable flag, the publisher of the tweet and the final text that corresponds to the tweet itself.

The tweets range from 1 to 20 words, around 5 to 100 characters. Just like the IMDb dataset, this is also a balanced dataset, each sentiment having 800,000 instances.

Of the total 1,600,000 tweets, 48,000 were used for training, since that was the average maximum size of the remaining datasets of other types and approximately 2,000 instances were used for validation. The testing set was composed of approximately 9,000 tweets.

The data, specially in text analysis, has to be pre-processed in order to eliminate noise and unnecessary words that can lead to wrong results, specially in cases like the Sentiment140 dataset, where the data is collected from everyday tweets that have grammatical errors.

The data distribution of the Sentiment140 dataset, for each class, is represented from Figure 4.14 to Figure 4.16. The sample sizes are of 48,000, 28,000 and 2,000, differing from the previous dataset in order to be compared to other data types, since there was enough data to have larger samples with this dataset and not with the previous.

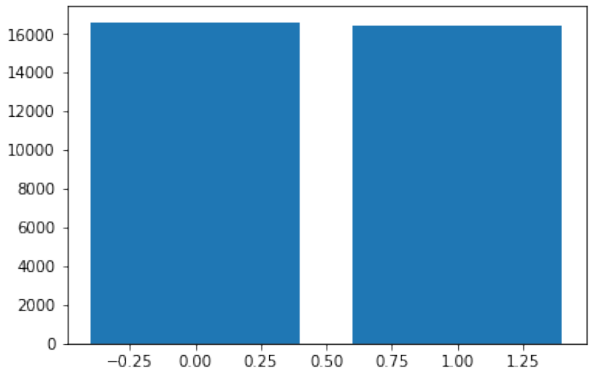

Figure 4.11: IMDb dataset 33,000 instances class data distribution

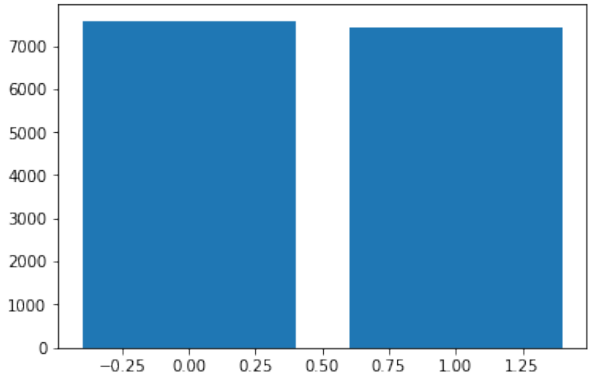

Figure 4.12: IMDb dataset 15,000 instances class data distribution

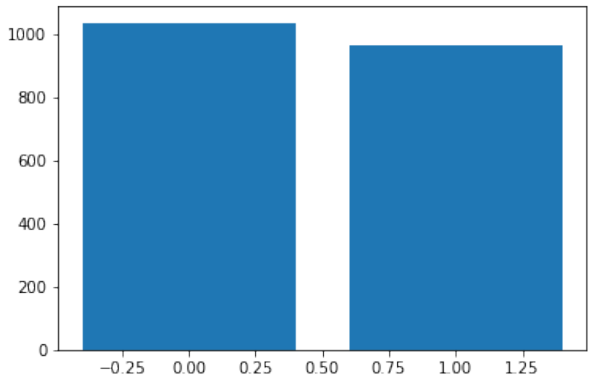

Figure 4.13: IMDb dataset 2,000 class data distribution

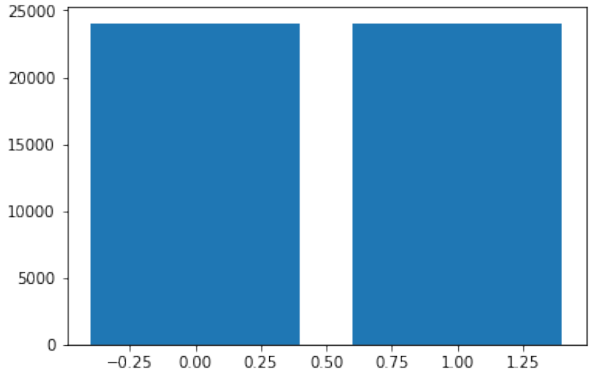

Figure 4.14: Sentiment140 dataset 48,000 instances class data distribution

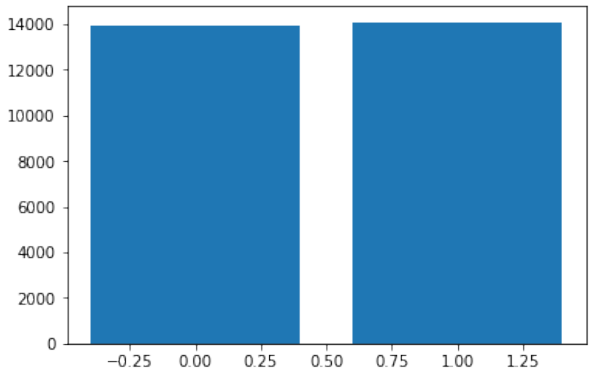

Figure 4.15: Sentiment140 dataset 28,000 instances class data distribution

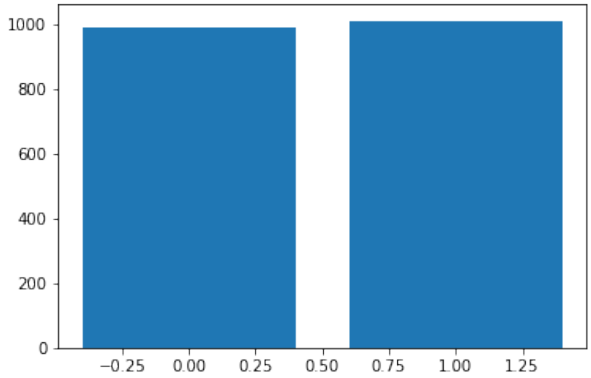

Figure 4.16: Sentiment140 dataset 2,000 instances class data distribution

4.2.4 Categorical datasets

The first used dataset entitled UCMF is a collection of data from the Real Hospital Português, Brazil. The data is anonymous and approved by both the hospital's Ethics Committee and the Ethics Committee of the University of Porto, Portugal.

The original dataset had 7,603 observations of 33 features and was already pre-processed from data cleaning to normalization and the removal of variables that did not present value, even though some pre-processing tasks were conducted after the ones described.

Children with age bellow 2 years old were removed from the dataset, since body mass index was featured in the data and the value is not usually assessed for younger ages. That resulted in a total of 7,203 instances from which 2,510 are pathological and 4,693 are healthy. Of the 7,203 instances, 5,762 were used for training and 1,441 for testing.

The final dataset was denominated as Mammo and is a collection of 55,214 instances with 25 features, from personal information as age to medical information as mass shape. 11,500 instances of the data were used for training, 988 instances for validation and 3,122 for testing.

The data was not pre-processed so several missing values, unnecessary variables and incorrect values were found and had to be corrected.

Among the features, several problems could be tackled. Both features *ASSESSMENT* and *PN* could be the classification target, the first representing the class or level of development of the mass, ranging from 0 to 9, and the second regarding the status of the patient, positive or negative. Since the *ASSESSMENT* variable still had around 15000 instances and the variable had more possible classifiable classes, that was the choice for this dataset.

4.3 Pre-processing

In this section is presented all the data processing, previous to the models' implementation. Each dataset, corresponding to a different data type, needs a special pre-processing, since text and sound for example, have different features and different problems when it comes to data noise and other impurities. Each pre-processing was based in similar work, since problems surrounding the data type follow the same type of operations with the objective of cleaning the data and making it viable, so the model can obtain better results.

The pre-processing treatment was the same for both datasets of the same type in order to maintain the fairness between them and retrieve results that may differ, only based on data proprieties and not the treatment they got before any modeling took place.

4.3.1 Image datasets

Since the dataset was loaded from the *tensorflow.keras.datasets*, it is already split into a training and testing sets, with a relation of approximately 85% and 15% of the entire data. In both datasets a simple normalization was conducted. Both datasets had square shapes, MNIST of 28 by 28 and CIFAR10 of 32 by 32 pixels and each pixel was represented by a number between 0 and 255, corresponding to the shade of the pixel from black to white in MNIST, and the RGB color since the CIFAR10 dataset had 3 arrays corresponding to each color.

In order to feed the models with numbers between 0 and 1, both *x_train* and *x_test* variables were set as float type, so that following division of each pixel in the dataset, by the maximum value of a pixel, as a float itself, 255.0, would result in a correct value for the network.

4.3.2 Sound datasets

Unlike the previous data type, both sound and text datasets required a more deep pre-processing. In the sound datasets, since the available data was somewhat reduced, between 500 and 1,500 entries each, the data had to be extracted, processed and augmented in order to have a more extensive data collection. Each folder had audio files corresponding to one of the identifying labels, so for each extracted file, the label had to be equally extracted and saved with the file in question.

[2]The pre-processing computation was extracted from the heartbeat sound dataset study by Manan Agarwal, since both the data and the problem trying to be solved were the same. For the heartbeat sound dataset, all audios with less than 3 seconds were discarded and for the ESC-50 dataset, since all audios had 5 seconds, all were kept. Depending on the audio's length, an iterations value was calculated in order to split the file into that amount of sub-parts. The iteration value for each file was calculated by setting a fixed slice size of 3, corresponding to the minimum in the heartbeat sound dataset audio length and following the next equation:

$$iterations = (audio\_length - slice\_size)/(slice\_size - 1)$$

$$iterations = iterations + 1$$

After having the number of iterations or partitions to be applied to each file, an initial offset was calculated in order to remove possible initial silence in the file. That offset depends on the audio's duration and the iterations value previously calculated and follows the equation:

$$initial\_offset = (audio\_length - ((iterations * (slice\_size - 1)) + 1))/2$$

Then, for the number of calculated iterations, is added to a data collection, the file name, the label corresponding to the class and an offset that is the result of the following expression:

$$offset = initial\_offset + i * (slice\_size - 1)$$

[2]https://github.com/MananAgarwal/Heartbeat-Classifier/blob/master/Heartbeat%20Classifier.ipynb

This will split the audio into several parts thanks to this offset latter used in the extraction of the audio information, having a final dataset that may contain several rows with the same file name and label, but with different offsets.

After collecting all the files, labels and corresponding offsets, there is a split into a train and test sets, which are variables with all the information regarding not only data proprieties, but also the labeling of each file. The split was 80% to the training set and 20% to the testing set, having 829 and 208 instances respectively.

After the split, comes the extraction of the data features for both sets. Such extraction is conducted by the use of mel frequency cepstral coefficient (MFCC), which is used in sound related problems, including speech, to reduce varying channel effects occurring in the audio data.

In order to calculate the MFCC the audio's digital time series and sampling rate is needed. Those values can be achieved by the use of the function librosa.load of the librosa library. The function takes the file path and offset and a duration, so for each file, the function returns the time series (y) and sampling rate (sr), beginning in an offset and with a duration of 3 seconds, the slice_size value. The loop that proceeds to call this function, makes so each audio is divided in X iterations, previously calculated, each with 3 seconds.

After having those values, the MFCC is determined with the use of the function librosa.feature.mfcc, which takes the y and sr values, as well as the argument n_mfcc, that represents the number of mfccs to return. That value was set to $sr/40$ since it had better results than a fixed number, which is the standard approach.

After having all the sound properties that are going to be used in the distinct models and algorithms, both *x_train* and *x_test* variables are transformed into arrays with the use of the *numpy* library.

After having the *x_train* and *x_test* sets ready, is time to deal with the corresponding labeling sets. In each labeling set, were appended the labels of each audio file and offset, from the train and test sets previously created. The next step was applied only to the ESC-50 dataset were the existing labels are encoded to integers with the use of *LabelEncoder* from *sklearn.preprocessing*, since the models don't work with string labeling.

The final step in the pre-processing operations also only applied to the ESC-50 dataset is the extraction of class weights, since although the dataset is pretty much balanced, there are a few classes with a bit less data representation, comparing to the classes with most data examples. This argument did not improve much performance in some models, but was still implemented with the *class_weight* function of the *sklearn.utils* library.

4.3.3 Text datasets

As for the text data pre-processing, a lot of work within the text and structure of the phrases was needed. To begin, some reviews had unknown characters, like German accentuation, so they

had to be solely removed from the instance. After removing unknown characters, other HTML codified characters had to be transformed to their ASCII form, by using the "unescape" function of the "HTMLParser" library.

After dealing with unknown and unrecognizable characters, all the reviews were converted to lower characters, and in the IMDb dataset, a lot of HTML code was embedded with the actual reviews so those pieces of code had to be removed. As for the Sentiment140 dataset, since the text was originated from Twitter, there are cases of mentions, using the "@" character and cases where URLs are present, so those characters and words had to be removed as well. A final operation revolved around the use of multiple spaces between words, which had to be removed, otherwise would also count as a word and would pollute the models and algorithms with noise.

After the text pre-processing, some new words were created by the removal of certain characters, like the apostrophe in "I'll" which would become "I ll". This way, the "ll" had to be removed as well as some characters that remained like mathematical operators.

The next steps involved removing every instance of any character apart from [a-z] and normalizing possible words to ASCII. Abbreviations were replaced with the extended version, like for example "isn't" became "is not" and stop words were removed with the use of the python's stop_words library. Finally, after the removal of all present noise, there had to be some insurance that the instance did not become empty. Pieces of text without relevant information could become pieces of text with only one random word or in worse cases, completely empty. Therefore, after the pre-processing previously described, all instances with one or less words were discarded since the entries with one word were very few.

After having all the dataset rows pre-processed, comes the preparation of the data structure, so that all instanced are represented in the same shape and dimensions. When it comes to representation, words have to be transformed to numbers that can identify them. With the use of *keras.preprocessing.text* function *Tokenizer*, each word is transformed to a number, allowing the models to learn and retrieve information. The function returns the presence or absence of a word in an array, and for the IMDb dataset, the *Tokenizer* kept track of the 300 most used words in each review, since the the average review has a length of 200 words, and for the sentiment140 dataset, for the 100 most used words, since that is the maximum limit of words in a tweet.

The Tokenizer transformation was used in the implementation of ANNs and for the rest of the models, it was used CountVectorizer, since the compatibility with the models did not allow the implementation of the same data representation. They are different ways of representing the same data, the first transforming the words into an array of integers that label each word and the second that transforms into a matrix of word counts, where the presence of a word is represented by 1, instead of the count of the words that do not bring any additional information and deviate from the representation in the ANNs. So instead of an array, it returns the same data in a matrix format, being this the last operation for the comparative models.

Since each review or tweet has a personal length and they are not all the same, they have to be normalized to a same length so that the ANN can compute the data. In the three networks,

for each dataset was determined the maximum number of words in a single instance and an extension of 0's is added to any review with less than that maximum. So if the maximum length of words is 100 and an instance has 80 words, that array will be completed with zeros until it has a length of 100.

After having the actual text pre-processed, the words tokenized and the same shape of every row, we can split the data into training and testing sets.

4.3.4 Categorical datasets

The pre-processing in the categorical datasets was in majority the same.
Both datasets started by having some features removed, in the UCMF dataset they were:

ID, DATA, DN, CONVENIO, IMC_CALCULADO_CORRETAMENTE, MEDICO_ASSISTENTE, MEDIA_IMC_POR_IDADE, IN_IDADE_CALCULADA, IN_PA_INFORMADA_CORRETAMENTE, IN_CONVENIO_INFORMADO and IN_PULSOS_INFORMADO_CORRETAMENTE

And in the Mammo dataset were:

DeAbnormalityID, FoldNum, DeMRN, DeMammoID, ASSESSMENT and MAMMO_RAD_ID.

Since there were some instances in the UCMF dataset with missing classification values, those were removed from the dataset and the variables with missing values were selected in order to be filled with a standard value. In the UCMF dataset the columns:

PULSOS, B2, FC, HDA_1, HDA_2, MOTIVO_1 and MOTIVO_2,

had missing values and since none of them had 0 as a value option, all the missing entries were filled with 0 instead. For the Mammo dataset the variables:

PreviousMammo, Composition, MASS_MARGINS, MASS_SHAPE, MASS_DENSITY, CHANGES, CALCIFICATIONS, CALCIFICATION_DISTRIBUTION_MODIFIER, ARCHITECTURAL_DISTORTION, CS Site-Specific Factor 6, SIZE, ASSOCIATED_FINDINGS and SPECIAL_CASES

were the ones with missing values and they too were filled with 0, apart from MASS_DENSITY that already had 0 values, so the value Z was introduced to represent missing values.

After that, all the categorical features were selected, in the UCMF dataset being:

PERCENTIL_IMC, FAIXA_ETARIA, PULSOS, SIT_PAS, SIT_PAD, RESULTADO_PAS_PAD, NORMAL_X_ANORMAL, B2, SOPRO, HDA_1, HDA_2, SEXO, MOTIVO_1, and MOTIVO_2

and in the Mammo dataset being:

ARCHITECTURAL_DISTORTION, REASON_FOR_THIS_MAMMOGRAM, FamilyHistory, PersonalHistory, PreviousSurgery, PN, MASS_MARGINS, MASS_SHAPE, MASS_DENSITY, CHANGES, CALCIFICATIONS, CALCIFICATION_DISTRIBUTION_MODIFIER, ARCHITECTURAL_DISTORTION, ASSOCIATED_FINDINGS, CS Site-Specific Factor 6, SIZE and SPECIAL_CASES.

They were all encoded using the *LabelEncoder* from the *sklearn.preprocessing* library. The encoder fitted the data and then transformed it into integers. The data was then splitted into training and testing sets, in a 80% to 20% ratio in both datasets.

The UCMF dataset did implement oversampling of the data since the present representation of both classes was sufficient for the problem solution. The *RandomOverSampler* was implemented and the found solutions did not improve, in fact, obtaining worse results by 2% in some methods. Instead, so more data could be trained, the over sampling was implemented and used for validation, since there was only a bit more than 5000 instances and the stipulated minimum data sample size was 2,000, creating an even narrower difference if more data had to be used for validation.

For the Mammo dataset, since there was a considerable discrepancy between classes, the *RandomOverSampler* was implemented but the results did not improve that much. Without the implementation, the most represented classes would have high sensitivity results and the less represented classes would have worse results. With the use of *RandomOverSampler*, what could be seen was the decrease of the sensitivity values of the most represented classes, and the improvement of approximately 25% of the remaining classes. That improvement would even reach 50% sensitivity in most cases, so, also as a way to evaluate the adaptability to unbalanced data, *RandomOverSampler* was not implemented.

4.4 Model implementation

In this section, special attention is given to the structure and decisions made around the building of the models. Aspects like hyper-parametrization of arguments in models, chosen layers, reasoning behind the structure of architectures, tuning of compilation arguments and optimizers are explained here.

The section, unlike the previous sections, is segmented by type of algorithm and within artificial neural networks, the subsection is divided by type of data since same data type datasets are more likely to share the same model's architecture.

4.4.1 ANNs

For each dataset, three types of architectures were implemented. Convolutional, recurrent and multilayer perceptron networks were chosen to be compared under different situations to understand the differences between them and the conditions that may favor or undermine their performance.

During the building process, the number and types of layers as well as the number of neurons were tested with several parameterization options. If there was no improvement, new adjusts to the current model would be made.

Different layers, in different sequences were tried to all models along with different number of neurons in each layer, in order to find the best performance for each dataset. Some cases the model was more simple while in other cases there was an opportunity to make the model more robust by adding new layers. The attempted number of neurons in each layer was always between 2 and 512 and mostly a exponential result of 2. When a specific value presented good results, new tests around that value would be conducted.

The amount of epochs in each training phase was tested with values between 10 and 100 and stipulated by the relation between training and validation results. Once the validation stagnated or the discrepancy between training and validation results would increase, the maximum number of epochs would be found.

Several optimizers were tested, like *Adam*, *Adamax*, *Adadelta* and *SGD*. These optimizers were chosen because of their application use in similar classification problems. The learning rate of some of the optimizers was altered and tested with several values, between 0.0001 and 0.01.

Along with the explanation of each model's architecture, there will be a description of some of the attempts that did present worse performance and a possible reasoning of why that may have happened.

4.4.1.1 Image datasets

For the MNIST dataset, the first implemented network was the CNN. This network has two consecutive sequences of a *Conv2D* layer with 64 filters, each with size equal to 9, calculated by the dimension of the kernel size (3,3), since it is required a tuple shape. The layer has a *relu* activation function and it's followed by a *MaxPooling2D* layer (with a pool size of (3,3)) and a final *Dropout* layer with a factor of 0.2.

After the two sequences of layers previously described, comes a *Flatten* layer, followed by a final output *Dense* layer with 10 neurons, corresponding to the 10 possible classes, with a *softmax* activation function.

The first layer, *Conv2D*, creates a convolutional kernel and produces an output as a tensor. It is followed by a *MaxPooling2D* layer for 2D spatial data with the intent of down-sampling the

input representation by choosing the maximum value of the 3x3 matrix. The *Dropout* layers are used to prevent overfitting, since the model presented slight principles of overfitting the data, presenting a slight higher accuracy results for the training set, and lower when tested and in the validation. The implementation of the *Dropout* layers helped in maintaining the validation above the training results. The *Dropout* layer sets a percentage of input units, equal to the argument value to 0 and the remaining input units are scaled up by $1/(1 - rate)$.

The used *Flatten* layer is used to flatten the input from 2D arrays into a single array and reduce the multi-dimensionality of the data since the next layer is the final *Dense* layer with 10 units, which follows the standard type and use for output layers, with the standard activation function as well.

This model began by being just the first two layers, usually used while creating convolutional networks, followed by the *Flatten* which was needed to transform the data into a compatible dimension with the following *Dense* layer, and output layers. Since that model obtained good results, the convolutional layers were replicated once more. After that, one more replica was tried and a change in the number of filters was also experimented, but the performance started to decline, so the final model was obtained.

For the compilation of the CNN model, as for the RNN and MLP models, the chosen optimizer was *Adam* with a default learning rate, the *sparse categorical crossentropy* as loss function since the shape of the labeling set is non-categorical, having the targets as integers and finally, accuracy as a metric. This metric was used only to guide the building of the network, not being the only metric used to evaluate the model's performance and that reasoning is presented in subsection 4.5 and 4.6.

The next model, RNN was implemented using a *LSTM* layer with 128 units, a *relu* activation function, followed by a *Dropout* layer with factor 0.2, also to prevent overfitting of the data, a *Dense* layer with also 128 units and a *relu* activation function. The output layer was the usual *Dense* layer with 10 neurons and a *softmax* activation function.

The RNN using *LSTM* layers proved to be more effective with only one *LSTM* layer and one dropout layer. The *Dense* layer was added to compute the output of the first layer a bit further before feeding the output layer, which obtained better results in the way that the accuracy at epoch X was higher with the addition of that layer.

Test were conducted with another *Dropout* layer of factor 0.2 but since the training and validation results were similar, that layer was removed.

In the final model MLP, only two layers were used, being a shallow network. Both *Dense* layers, the first having 128 neurons and a *sigmoid* function, while the output layer had 10 neurons and a *softmax* activation function. This layer was computed with several values for the number of neurons, *relu* activation function in the first layer, and the addition of more layers between the input and output layers, but they all got similar results. Since the model's performance did not change significantly and since a simple model had the same results, in order to have a smaller

running time, this was the chosen architecture. Just like the other models, this was compiled with the same compilation arguments.
As for the CIFAR10 dataset, the used CNN was the same as the one used for the MNIST dataset. The only difference was in the number of times the first pattern of layers occurred. This model had three sequences of *Conv2D, MaxPooling2D* and *Dropout(0.2)* instead of two. In this case, the addition of a third sequence of those layers only improved the model, but it was the limit before the performance deteriorated. As for the compilation, it also used the default *Adam* optimizer, *sparse categorical crossentropy* as loss function and accuracy as metric.

The RNN model for this dataset also followed the same architecture as the one used in the previous dataset, but again, the addition of layers improved the performance. The model had an input *LSTM* layer with 128 neurons, a *relu* activation function and, since there was a second *LSTM* layer, the *return_sequences* argument of the input layer was set to true. The next layer was a *Dropout* layer with factor of 0.1.
The sequence of these two layers was repeated once again, but this time, the *LSTM* layer had 64 neurons and the *return_sequences* argument was set to false since it is followed by a *Dense* layer, which expects a different shape than the one passed by the *LSTM* layer with that arguments set to true.
The second *Dropout* layer had an initial factor of 0.1 but latter changed to 0.2 since it was the last layer before the output layer and the model needed a bit more constrain to decrease overfitting.
The output layer was a *Dense* layer with 10 neurons and a *softmax* activation function.
This model was also tried with *Dense* layers before, in the middle and after the *LSTM* layers, with different activation functions and different number of neurons, without much success.

As for the compilation of the model, the *Adam* optimizer was not used since it did not obtain better results, even with lower learning rates. The *SGD* optimizer, with a learning rate of 0.001, decay of $1e^{-5}$, momentum of 0.9 and the *nesterov* momentum argument set to true, was the best tried optimizer. As for the loss and metric functions, the same as the previous models were applied.

The final implemented model, MLP for the CIFAR10 dataset had and input *Dense* layer with 256 neurons and a *sigmoid* activation function. Since the model overfitted the data, two *Dropout* layers were used. The first with a factor of 0.1 and the second, placed after the second *Dense* layer, with a factor of 0.4. That second *Dense* layer had 128 neurons and a *relu* activation function.
The output layer was a *Dense* layer with the standard 10 neurons and a *softmax* activation function.
This network obtained the same results for higher amounts of neurons in each layer, so in order to preserve time of execution, 256 and 128 neurons seemed fitting.

As far as the optimizer, the *SGD* optimizer did also have better results as it did in the RNN network. The alternative optimizer was the *Adam* optimizer with a learning rate of 0.001, a beta_1 value of 0.9 and an epsilon of $1e^{-7}$, did not obtained better results.

4.4.1.2 Sound datasets

For the sound datasets, the same CNN was used, although in the ESC-50 dataset, another CNN was implemented since it presented faster computation but overfitted after a certain epoch. The network was created in the same [3]study referenced in the pre-processing section and it is composed of four sets of the same layers.
The first layer is a *Conv2D* layer with 16 neurons, a kernel with size 2 and a *relu* activation function. This is the input layer which creates a convolutional kernel that will consequently create a tensor of outputs. Since every input layer needs to be informed by the shape of the data, the variable *data_shape* was set as the shape of the training set.

Following the *Conv2D* layer, comes a *MaxPooling2D* layer with a pool size of 2, like previously mentioned, this layer will down-sample the input by excluding the maximum value defined by *pool_size* for each dimension. The *pool_size* value sets the window size which informs the previously mentioned maximum to take. Finally, to prevent overfitting of the data, a *Dropout* layer with 0.2 factor is included. This will randomly set twenty percent of the input units to 0 during the training process, while the remaining are scaled up by 1.25 so that the sum of inputs remains the same.

The model has four replicas of these three layers, although the last three sets do not define the input shape, since that is not needed in hidden and output layers. In the sequence previous to the output layer, the factor of the *Dropout* layer is set to 0.5 and has an additional *GlobalAveragePooling2D* layer that will apply average pooling on spatial dimensions until each is one, leaving other dimensions the same. After the four sets of these layers, comes a *Dense* layer that functions as an output layer with 2 neuron in the heartbeat sound dataset and 24 neurons in the ESC-50 dataset, representing the possible classes in each. In both datasets the activation function is the *softmax* function, usually used in output layers.

For the heartbeat sound dataset, the RNN model had an *LSTM* input layer with 16 neurons and like the previous network, the input shape of the data was declared. Still in the input layer, two arguments were added. The *recurrent_dropout* was set to 0.5 and the *return_sequences* was set to *True*. The first argument, unlike the *Dropout* layers that transform the input that they receive, performs the transformation of the recurrent state. The *return_sequences* argument is set to true because the next layer is also a *LSTM* layer and therefore needs a three-dimensional sequence input, so this arguments indicates whether to return the last output in the sequence. After the input layer comes a *Dropout* layer with 0.5 factor followed by a second *LSTM* layer. This layer has 32 neurons, a *recurrent_dropout* of 0.5 and in this layer, since it is followed by a *Dense* layer, the *return_sequences* is set to *False* since *Dense* layers do not carry three dimension arguments. As usual, another *Dropout* layer is added with 0.5 factor followed by the first *Dense* layer with 64 neurons and a *relu* activation function.
This network was tested without this layer and obtained worse results, so the addition of another layer with increasing neurons was executed. Another *Dense* layer was added but the network

[3]https://github.com/MananAgarwal/Heartbeat-Classifier/blob/master/Heartbeat%20Classifier.ipynb

became slower and the results did not improve with that addition, so only one *Dense* layer was added.
Finally, the output layer was a *Dense* layer with the two class representative neurons and a *sigmoid* activation.

For the ESC-50 dataset, the input layer was an LSTM layer with 128 units, the *return_sequences* set to True, followed by a *Dropout* layer with a 0.2 rate. The next layer was the same used as input layer without the *input_shape* argument.
Another *Dropout* layer with 0.2 rate was implemented before the final *LSTM* layer with 64 units and the *return_sequences* argument set to False, since the output layer is a *Dense* layer and the input shape is not compatible with the output from the *LSTM* layer. The output layer has 24 neurons, corresponding to the classification classes and a *softmax* activation function.

The final network for the heartbeat sound dataset was the MLP. Unlike the previous networks this required an one-dimensional input shape that is the multiplication of the two dimensions into only one. The first *Dense* input layer has the mentioned input shape along with a *relu* activation and 256 neurons. This network only had one *Dropout* layer with a 0.5 argument that follows the input layer. After that, two *Dense* layers with 128 and 64 neurons respectively are implemented. Both with the *relu* activation function just like the input layer. As the output layer, another *Dense* layer was used with the typical 2 neurons and a *softmax* activation function.

The MLP network for the ESC-50 dataset was different from the one used in the Heartbeat sound dataset, since that network did not perform well for the multi-class dataset. The dataset had a *Dense* input layer with 128 neurons and the *sigmoid* function, followed by a 0.5 rate *Dropout* layer. After this layer, there are two *Dense* layers with 128 neurons each, with a *Dropout* layer with 0.5 rate between them. Finally, the output layer is a *Dense* layer with 24 neurons and a *softmax* activation function.

4.4.1.3 Text datasets

The CNN model was the same for both datasets. The model had an *Embedding* input layer with the arguments 10000 and 32 as input and output dimensions, respectively, and an input length of 300 words for the IMDb dataset and 100 for the Sentiment140 dataset. This layer is used in all networks and it's end is to learn word embedding for all words passed through the input. The input dimension will be converted into a vector space (output dimension) with all words embedded to that size. It is followed by a *Dropout* layer with 0.5 rate and a *Conv1D* layer with 32 filters, a kernel size of 3, *relu* activation function and the padding argument with the variable *same*. This has the same function as the convolutional layers in the previous networks and the padding argument set to *same* makes the output have the same dimension as the input.
Next comes another *Dropout* layer with 0.5 rate and a *MaxPooling1D* layer with a pool size of 2. This will downsample the representation of the input as explained in other networks that used this layer. The next layer is a *Flatten* layer, followed by a *Dropout* layer with the 0.5 rate. Before the output layer, comes a *Dense* layer with 250 neurons and a *relu* activation function.

The output layer is a *Dense* layer with 2 neurons and a *sigmoid* activation function.

The RNN using the *LSTM* layer is the same for both datasets apart from the number of neurons in some layers.
The input layer is an *Embedding* layer with 10000 as input dimension, an output dimension of 100 and an input length of 300 for the IMDb dataset and 100 for the Sentiment140.

The next layer is an *LSTM* layer with 64 neurons for the IMDb dataset and 256 for the Sentiment140, both with the return sequences argument set to *True*, so the data shape can be interpreted by the following layers. To prevent overfitting, a *Dropout* layer with 0.5 rate is implemented. After, comes a *GlobalMaxPool1D* layer followed by another *Dropout* layer of 0.5.

Before the output layer, comes a *Dense* layer with 32 neurons for the IMDb dataset and 128 for the Sentiment140, both with a *relu* activation function and a final *Dropout* layer of 0.5 dropout rate.
The output layer is a *Dense* layer with 2 neurons and a *sigmoid* activation function.

The final network is the MLP where both datasets had the same network apart from the embedding size in the input layer, being the only difference between them.

The input layer is an *Embedding* layer with an input dimension of 10000 and output dimension of 32 for the IMDb dataset and 64 for the Sentiment140. The input length for both datasets is the same used in the previous described network. The next layer is a *Dropout* layer with a 0.5 rate, followed by a *Flatten* layer and a *Dense* layer with 256 neurons and a *relu* activation function. Before the output *Dense* layer with 2 neurons and a *sigmoid* activation function, we have another *Dropout* layer with a 0.5 factor.

4.4.1.4 Categorical datasets

The CNN model for the categorical datasets was basically the same with some adjustments.

For the UCMF dataset, the input layer is a *Dense* layer with 20 neurons and a *relu* activation function.
The first convolutional layer is a *Conv1D* layers with16 filters, a kernel size of 2 and also a *relu* activation function. This layer is followed by a *MaxPooling1D* layer with a pool size of 2 to downsample the data. The next layer is a *Dense* layer with 16 neurons and a *relu* activation function, followed by a *Flatten* layer and the output layer. This output layer is a *Dense* layer with 2 neurons and a *softmax* activation function.

The CNN model for the Mammo dataset had an *Dense* input layer with 18 neurons and a *relu* activation function. It is followed by a *Conv1D* layer with 32 filters, a *relu* activation function and a kernel size of 2, and a *MaxPooling1D* layer with a pool size of 2. These two layers are repeated but this time, the *Conv1D* layer has 64 filters.

After, comes a *Dense* layer with 32 neurons and a *relu* activation function, a *Flatten* layer

and finally, the output *Dense* layer with 2 neurons and a *softmax* activation function.

Both models presented better results with a *Dense* input layer instead of an immediate convolutional layer. The network was tested with a kernel size of 3 but the results were similar. The *MaxPooling1D* layer was used with the same intent as other models, and even though the network presented good results without this layer, it was still a good addition for the end results.

Since there was a convolutional layer, the data shape was inconsistent with the one expected by the output layer, so the *Flatten* layer was mandatory.

The Mammo dataset presented better results with the addition of another sequence of a convolutional and downsampling layers, having more layers than the UCMF model. The RNN model was basically the same with the exception of the number of neurons in each layer.

Unlike the CNN model, this network did not present better results with a *Dense* layer as input layer. Therefore, a *LSTM* layer was used as input, with 20 units for the UCMF dataset and 18 units for the Mammo dataset. Both layers in both datasets had the return sequences set to true.

The following layer is also a *LSTM* layer with 64 units for the UCMF dataset and 32 for the Mammo dataset. This layer has the return sequences argument set to false since the next layer is a *Dense* layer, which does not support the output shape passed by the *LSTM* layer. This *Dense* layer has 128 neurons for the UCMF dataset and 32 for the Mammo dataset.

The output layer is a *Dense* layer for both dataset, the UCMF having 2 neurons and the Mammo having 8. Both had *softmax* activation functions.

In this network *Dropout* and pooling layers were tried but did not improve the performance.

The final network, MLP was the same for both datasets, the Mammo only having the addition of *Dropout* layers to contain the learning process.

Both networks are shallow, having only one hidden layer. The input layer is a *Dense* layer with 20 neurons for the UCMF dataset, 18 for the Mammo dataset and a *relu* activation function.

The Mammo dataset input layer was followed by a *Dropout* layer with a rate of 0.5, while the UCMF dataset did not need that layer. The hidden layer was a *Dense* layer with 16 neurons in both dataset networks.

The Mammo dataset had another *Dropout* layer with 0.5 rate before the output layer, which was a *Dense* layer with 2 neurons for the UCMF dataset and 8 neurons for the Mammo dataset. Both layers had a *softmax* activation function.

This network had better performance for lower number of neurons and lower number of layers. When one of those factors or even both increased, the performance was worse and the learning process deteriorated.

Table 4.1 presents the compilation tuning of the networks.

Dataset	ANN type	Optimizer	Epochs
MNIST	CNN	Adam	15
	RNN	Adam	15
	MLP	Adam	15
CIFAR10	CNN	Adam	50
	RNN	SGD	100
	MLP	SGD	100
IMDb	CNN	Adam	15
	RNN	Adam	15
	MLP	Adam	15
Sentiment140	CNN	Adadelta	10
	RNN	Adadelta	9
	MLP	Adadelta	7
Heartbeat sound	CNN	Adam	140
	RNN	Adamax	100
	MLP	Adamax	140
ESC-50	CNN	SGD	100
	RNN	Adamax	140
	MLP	Adamax	200
UCMF	CNN	Adamax	25
	RNN	Adamax	25
	MLP	Adamax	25
Mammo	CNN	Adamax	100
	RNN	Adam	150
	MLP	Adamax	100

Table 4.1: ANN hyper-parametrization of each model in each dataset

The SGD optimizer had a momentum of 0.9, learning rate of 0.01 for the ESC-50 CNN model, and a decay of $1e^{-6}$ while the CIFAR10 RNN and MLP networks had a learning rate of 0.001 and decay of $1e^{-5}$ and $1e^{-4}$ respectively.

Again, the number of epochs was tested with values between 10 and 100 with a 10 interval between them. Once the best results were found, new tests would be executed around that value. Cases where there were more than 100 epochs were cases where the learning process was still taking place and the learning and loss curves were still not stagnated. In those cases, tests from 100 to 200 under the same conditions as previously described would be implemented.

4.4.2 SVMs

The SVM models were build the same way for each dataset. Apart from the pre-processing operations, the creation of the model was equal throughout the practical component of the study.

Within SVMs, several hyper-parameters were tuned in order to find the best fit for the testing data. The tuned parameters were the type of kernel, the regularization variable c, the value of gamma and the probability argument.

The regularization variable c is tuned in order to find the fit to avoid misclassification by changing the margin of the hyperplane, finding the best relation between having a larger range between classes or a smaller range. That leads to a potential increase in misclassification depending on the spatial disposition of the data.

Since the values of c range from small to larger values, from 0.01 to 100, the value of gamma is a deciding factor since it can over-learn the data and create a very constrictive model or be more open to some divergent new data.
High values of gamma would lead to the first case, where there is not much flexibility for data disparity, leading to lower values being the best fit for an overall common data. The tested gamma values were 0.0001, 0.001, 0.01 and 1.

The following image shows the relation between the c and gamma values. Higher values of both arguments lead to a more constraint classification while lower values lead to broader intervals.

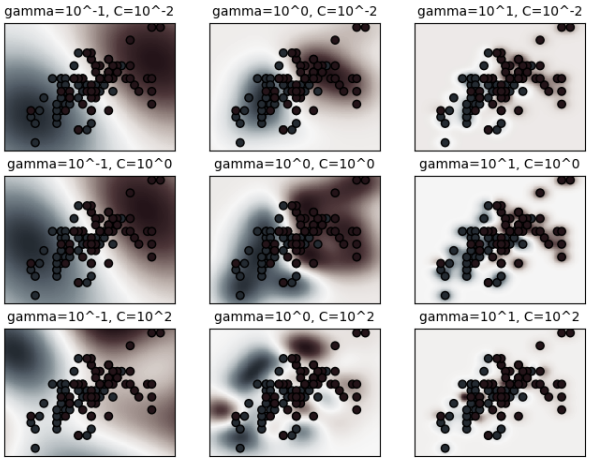

Figure 4.17: Relation between the variables c and gamma
Source:https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

The probability argument was set to *True* since it internally uses 5-fold cross-validation, although it slows down the model, with SVM having the longest running times of all models.

Also, three kernel types, linear, polynomial and RBF were tested along with the previously mentioned hyper-parameters, since each data shape and type could be better for a particular kernel.

Finally, it was also tested a linear model with stochastic gradient descent learning, *SGDClassifier*. The tuned hyper-parameters were the maximum iterations which function like epochs in the fit function. This value was indifferent after a certain value, so it was kept bellow 150. The number of jobs for the number of CPUs, since there are multi-classification problems and finally, the loss function which was tested with hinge loss for a linear type SVM, squared hinge for quadratic penalization and log for logistic regression.

Since SVMs can be time consuming, and there are some large data samples, smaller sizes were tested and the behaviour of the model was studied. In fact, the SVM testing for the larger CIFAR10 data sample was not able to be finished since it took several days of continuous compilation, to which point it was decided to assume that the complexity of the data and the overall size, made the algorithm too slow. That made the SVM CIFAR10 experiment to have only two tested data samples, one of the usual 2,000 instances and the other of 28000. The parametrization for each dataset is in the following Table 5.5

	Size	Kernel	C	Gamma
MNIST	58k	RBF	10	0.01
	28k	RBF	10	0.01
	2k	RBF	10	0.01
CIFAR10	28k	RBF	10	0.01
	2k	RBF	10	0.01
Sentiment140	48k	RBF	10	0.01
	28k	RBF	10	0.01
	2k	RBF	10	0.01
IMDb	33k	RBF	10	0.001
	15k	RBF	10	0.001
	2k	Linear	0.01	0.001
Heartbeat Sound	0.8k	Poly	1	0.001
	0.5k	Linear	1	0.001
	0.2k	Poly	1	0.001
ESC-50	1.5k	Linear	1	0.001
UCMF	5k	Linear	10	0.001
	2k	Linear	0.1	0.001
Mammo	12k	RBF	100	0.0001
	6k	RBF	100	0.0001
	2k	RBF	100	0.0001

Table 4.2: SVM hyper-parametrization of each dataset for each sample size

4.4.3 Random Forests

The RF models for each dataset were built the same way in all occasions. The *RandomForestClassifier* from the *Sklearn.ensemble* library was used to create the models, and several hyperparameters were tuned to find the best results. Among those parameters, the number of estimators was set to various values, between 10 and 100. It represents the number of trees in the forest and should be limited in order to find the best number needed to find the optimal solution. The maximum depth was set between 10 and 500, since if it was set to default the nodes would expanded until all leaves are pure, what could lead to worse results. Results for models with more than 500 levels of depth obtained similar results as the model with 500, determining our maximum value for the parameter.

The warm start argument is a boolean variable that when true, reuses the solution of the previous call. That leads to more estimators added, otherwise, it fits a new forest. Finally, the class weight was adjusted in cases where the dataset was unbalanced, uniforming the classes so that they are all represented equally.

The parametrization for each dataset is in the following Table 4.3

	Size	Maximum depth	Number of estimators	Warm start
MNIST	58k	500	100	True
	28k	500	100	True
	2k	100	100	False
CIFAR10	48k	100	100	False
	28k	500	100	True
	2k	100	100	False
Sentiment140	48k	500	100	False
	28k	500	100	False
	2k	500	100	True
IMDb	33k	500	100	False
	15k	100	100	True
	2k	100	100	True
Heartbeat Sound	0.8k	10	100	False
	0.5k	10	100	True
	0.2k	100	100	False
ESC-50	1.5k	100	100	True
UCMF	5k	10	100	True
	2k	100	100	True
Mammo	12k	10	100	True
	6k	10	100	True
	2k	10	100	True

Table 4.3: Random forest hyper-parametrization of each dataset for each sample size

4.4.4 Logistic Regression

For the logistic regression models, all datasets, like the previous algorithms, went through the same hyper-parameter tuning. The values of c were tested between 0 and 100, with most tests ranging between 0 and 1, since that is where most significant changes happened. It is inverse to the lambda regulator and so, for smaller values, increases the lambda power of regulation (derived from the c value of SVMs). Smaller values have a more consistent effect, while large values are more fluid and can lead to a more broad range of outcomes.

Some solvers like *sag*, *saga* and *lbfgs* were tested for all the predefined c values to be used in the optimization problem. *Sag* and *saga* solvers are faster for larger datasets and alongside *lbfgs* are a few of the only solvers capable of handling multi-class problems and multinomial loss. They all handle L2 or no penalty and *saga* can also handle L1 penalty, describing the reasons why they were the chosen solvers.

Other hyper-parameters were tested but found to be irrelevant to most cases, like *fit_intercept*, that specifies a bias to the decision function, that would vary around 0.0001.

The *max_iter* argument also proved to be irrelevant in most cases, since values of 10 and

1000 obtained the same results, varying in the same order as the *fit_intercept* variable. The parametrization for each dataset is in the following Table 5.6

	Size	Solver	C
MNIST	58k	lbfgs	1
	28k	sag	10
	2k	saga	0.1
CIFAR10	48k	lbfgs	0.01
	28k	lbfgs	0.01
	2k	lbfgs	0.01
Sentiment140	48k	saga	1
	28k	saga	1
	2k	lbfgs	1
IMDb	33k	lbfgs	0.1
	15k	lbfgs	0.1
	2k	lbfgs	0.1
Heartbeat Sound	0.8k	lbfgs	1
	0.5k	saga	1
	0.2k	saga	1
ESC-50	1.5k	sag	0.01
UCMF	5k	lbfgs	100
	2k	lbfgs	0.1
Mammo	12k	sag	0.1
	6k	sag	0.1
	2k	sag	0.1

Table 4.4: Logistic regression hyper-parametrization of each dataset for each sample size

4.5 Evaluation metrics

Since this is a study of different models, data types and structures among other factors, the evaluation should also have different angles so the results could be interpreted in different ways.

The evaluation process starts with the creation of a confusion matrix that shows the relation between the real classes and the classification of the model. Ideally, the matrix should have higher results in the $x = -y$ axis, representing correct classification of each instance of the test set. At the same time, the accuracy of the model is determined and presented along with the matrix.

After having the confusion matrix, the indicators of true positive and negatives, and false positive and negatives are extracted from the matrix. There are direct ways to extract these indicators, using embedded functions but these values were calculated by their original equations,

using the results present in the confusion matrix. The FP values were calculated by subtracting the the correct classified values, present in the $x = -y$ axis of the class, from the sum of all the wrong classified values, present in the column of the current class. It follows the equation:

$$fp = cm.sum(axis = 1) - np.diag(cm)$$

The FN values were determined in the same way but instead of subtracting the correct classified values from the sum of the entire column, the subtraction is from the sum of all the values present in the row, meaning, for example in a case with 10 classes, all the times that the class 0 was predicted when it should have been one of the remaining 9. It follows the equation:

$$fp = cm.sum(axis = 0) - np.diag(cm)$$

The TP values are the results present in the $x = -y$ axis and it's equation is:

$$tp = np.diag(cm)$$

Finally, the TN values are represented by the sum of the entire matrix except the FP and FN values, meaning that for a class, it is represented by the sum of all the values, except the row and column of said class. Since the FP and FN values are the sum of the column and row, respectively, minus the diagonal values, those values present in the diagonal must be introduced when calculating the TN results. That follows the equation:

$$tn = cm.sum() - (fp + fn + np.diag(cm))$$

Now that we have all the indicators regarding the confusion matrix, we can determine the values of sensitivity and specificity of each class. As previously mentioned, sensitivity is determined by dividing the TP value by the sum of TP and FN values, represented:

$$sens = np.true\_divide(tp, (tp + fn))$$

As for the specificity, it is the result of the division of the TN value by the sum of TN and FP values, represented by:

$$spec = np.true\_divide(tn, (tn + fp))$$

After having the sensitivity and specificity values of all classes, a mean is calculated just to have an overall understanding of the performance, although the real importance is in the values of each class, since a model can be incredible at classifying a particular class.

As a final evaluation metric a ROC plot was created in order to determine the AUC value. In order to do that, the model's probabilities prediction is determined, for each class followed by the computation of false positive and true positive rates. With the ROC curve for each class, there is also the plot of the micro and macro-average ROC curves. The macro-average will treat each class equally by computing the metric independently and take the average, while the micro-average aggregates all classes' contributions to compute the ROC curve. After plotting the ROC curves, with the use of the *roc_auc_score* function, the AUC value is determined. So overall, the confusion matrix, the TP, TF, FP, FN indicators, accuracy, sensitivity, specificity and

AUC values are determined for each model or algorithm for future comparison and evaluation.

Chapter 5

Results

In this chapter, the presentation of the results and the discussion and comparison between them are made separately so that there is a section for visualization of the data and other for the critical thinking and comprehension of the results.

In the *Results presentation* section, the results may be presented in form of tables, separated by data type, evaluation metric and dataset size. The ANN architectures are treated as individual models and therefore presented along the comparative algorithms so that all implementations can be compared at the same time. In the *Results comparison* section, a descriptive comparison of the final conclusions and a thought process that may support those conclusions is described and a summary of the main findings is presented in the next chapter.

5.1 Results presentation

	CNN	RNN	MLP	SVM	RF	LR
MNIST	99.22	98.84	**99.70**	98.34	96.87	92.03
CIFAR10	**76.33**	61.11	55.68	53.32	46.77	41.29
Sentiment140	76.00	74.84	74.03	**76.71**	74.77	76.40
IMDb	81.50	81.00	81.66	88.64	85.62	**88.97**
Heartbeat Sound	**88.87**	62.02	73.60	69.71	71.15	70.67
ESC-50	**76.78**	55.62	54.64	51.95	50.91	51.43
UCMF	93.72	93.50	93.64	93.75	**93.96**	93.89
Mammo	65.70	63.71	63.32	64.12	**68.09**	63.00

Table 5.1: Accuracy results for the maximum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	**96.20**	91.19	89.62	93.09	91.11	88.52
CIFAR10	**51.87**	41.07	38.61	40.42	36.61	36.36
Sentiment140	65.09	66.74	62.40	67.73	66.58	**67.83**
IMDb	78.48	75.32	76.27	84.13	81.42	**84.02**
Heartbeat Sound	**77.90**	50.49	67.33	63.46	65.38	67.30
ESC-50	**76.78**	55.62	54.66	51.95	50.91	51.43
UCMF	92.66	93.11	90.82	93.64	**93.82**	93.75
Mammo	61.41	61.06	62.63	62.45	**66.17**	62.62

Table 5.2: Accuracy results for the minimum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	**99.15**	98.93	97.70	98.32	96.87	91.91
CIFAR10	**76.32**	61.11	55.68	53.32	46.35	41.28
Sentiment140	76.02	74.84	74.03	**76.66**	74.72	76.37
IMDb	81.48	81.00	81.63	88.64	85.62	**88.96**
Heartbeat Sound	**87.65**	53.88	66.88	67.48	71.57	67.40
ESC-50	**77.36**	56.64	55.66	53.05	53.29	51.52
UCMF	92.00	91.71	91.78	92.05	**92.30**	92.16
Mammo	31.49	27.70	23.03	18.89	**52.21**	22.18

Table 5.3: Sensitivity results for the maximum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	**96.09**	89.53	89.46	93.01	90.96	88.36
CIFAR10	**51.88**	41.01	38.63	40.42	36.23	36.36
Sentiment140	55.60	67.70	62.41	67.66	66.45	**67.77**
IMDb	78.44	75.32	76.28	**84.12**	81.61	84.02
Heartbeat Sound	**75.87**	49.54	63.35	61.44	63.45	64.19
ESC-50	**77.36**	56.64	55.66	53.05	53.29	51.52
UCMF	90.99	91.13	91.13	91.83	**92.11**	91.97
Mammo	28.28	24.01	24.34	22.37	**43.73**	24.17

Table 5.4: Sensitivity results for the minimum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	**99.90**	99.88	99.74	99.81	99.65	99.11
CIFAR10	**97.37**	95.67	95.07	94.81	94.09	93.47
Sentiment140	76.02	74.84	74.03	**76.66**	74.72	76.37
IMDb	81.48	81.00	81.63	88.64	85.62	**88.96**
Heartbeat Sound	**87.65**	53.88	66.88	67.48	71.57	67.40
ESC-50	**98.99**	98.07	98.02	97.91	97.86	97.88
UCMF	92.00	91.71	91.78	92.05	**92.30**	92.16
Mammo	92.48	92.09	91.82	91.79	**93.57**	91.62

Table 5.5: Specificity results for the maximum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	**99.57**	98.84	98.84	99.23	99.01	98.72
CIFAR10	**94.65**	93.44	93.18	93.38	92.97	92.92
Sentiment140	55.60	67.70	62.41	67.66	66.45	**67.77**
IMDb	78.44	75.32	76.28	**84.12**	81.61	84.02
Heartbeat Sound	**75.87**	49.54	63.35	61.44	63.45	64.19
ESC-50	**98.99**	98.07	98.02	97.91	97.86	97.88
UCMF	90.99	91.13	91.13	91.83	**92.11**	91.97
Mammo	92.01	91.86	91.50	91.41	**93.13**	91.53

Table 5.6: Specificity results for the minimum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	**99.99**	99.98	99.96	99.98	99.91	98.98
CIFAR10	**97.03**	92.15	90.62	89.59	85.29	81.57
Sentiment140	83.24	83.18	81.90	**84.03**	82.55	83.81
IMDb	89.50	89.50	89.39	95.61	93.36	**95.62**
Heartbeat Sound	**94.13**	52.05	71.92	70.12	71.09	69.51
ESC-50	**98.56**	92.17	92.35	92.79	90.57	91.21
UCMF	94.35	94.07	**94.96**	92.41	94.89	94.09
Mammo	89.10	87.17	83.52	82.33	**89.73**	81.65

Table 5.7: AUC results for the maximum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	98.87	99.10	99.10	**99.68**	99.28	98.53
CIFAR10	**88.33**	82.62	81.04	82.94	79.59	78.42
Sentiment140	70.72	73.42	68.11	73.93	73.72	**74.10**
IMDb	86.14	82.65	83.97	91.68	90.21	**91.82**
Heartbeat Sound	**82.98**	50.58	68.72	62.24	62.32	65.10
ESC-50	**98.56**	92.17	92.35	92.79	90.57	91.21
UCMF	**94.25**	93.74	93.74	93.98	94.18	93.68
Mammo	84.23	**85.66**	80.50	82.06	85.37	79.95

Table 5.8: AUC results for the minimum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	885	675	60	2250	**35**	54
CIFAR10	3300	7600	1100	19652	238	**157**
Sentiment140	100	1215	195	2170	263	**1.83**
IMDb	1500	2250	855	4831	130	**3.36**
Heartbeat Sound	1120	420	200	111	5.92	**2.34**
ESC-50	1000	280	200	208	**19**	396
UCMF	13	75	13	95	0.42	**0.1**
Mammo	100	750	50	70	**0.69**	1.06

Table 5.9: Execution time in seconds for the maximum dataset size

	CNN	RNN	MLP	SVM	RF	LR
MNIST	30	30	7.5	18	**1.11**	11
CIFAR10	200	400	100	234	**6.48**	7.60
Sentiment140	10	54	7	2.44	5.58	**0.23**
IMDb	75	210	75	28	3.39	**0.37**
Heartbeat Sound	210	140	75	5.21	**0.99**	5.16
ESC-50	1000	280	200	208	**19**	396
UCMF	13	25	13	0.61	0.25	**0.06**
Mammo	50	100	50	1.51	0.26	**0.15**

Table 5.10: Execution time in seconds for the minimum dataset size

MNIST	Se0	Se1	Se2	Se3	Se4	Se5	Se6	Se7	Se8	Se9
CNN-S3	99.59	99.47	**99.41**	**99.20**	**99.79**	99.32	98.85	**99.22**	99.17	**97.52**
RNN-S3	**99.69**	99.47	**99.41**	**99.20**	99.59	**99.43**	**98.95**	97.95	**99.28**	96.33
MLP-S3	98.87	98.85	96.99	98.21	97.45	97.08	97.80	97.17	97.12	97.42
SVM-S3	99.28	**99.55**	98.35	98.51	9.26	97.98	98.64	97.56	97.84	97.22
RF-S3	98.97	98.85	96.84	96.33	97.04	96.30	97.49	96.01	95.68	95.14
LR-S3	98.06	98.06	88.95	90.99	93.27	86.32	94.57	92.60	87.57	88.70
CNN-S2	99.79	99.38	99.22	**99.60**	99.08	98.76	98.22	98.63	99.07	99.00
RNN-S2	98.57	99.38	99.03	99.40	98.87	94.17	97.80	96.49	97.94	96.63
MLP-S2	98.67	98.85	96.22	97.62	96.53	94.95	97.39	96.88	97.12	95.24
SVM-S2	98.97	99.38	97.77	98.31	97.86	96.74	97.59	97.17	97.43	96.43
RF-S2	99.08	98.59	95.83	95.64	95.72	95.17	97.49	95.13	94.35	94.44
LR-S2	97.44	98.06	89.43	90.29	93.78	87.10	94.36	92.02	86.13	97.11
CNN-S1	**99.85**	**99.03**	**96.89**	**92.77**	**97.75**	**95.51**	**97.39**	**94.84**	**91.37**	**97.52**
RNN-S1	90.40	96.82	93.12	73.46	94.80	90.13	95.92	91.43	86.75	82.45
MLP-S1	97.14	98.32	89.63	83.76	90.83	85.76	91.54	90.17	78.23	89.19
SVM-S1	98.26	98.85	93.50	84.40	94.09	91.81	95.40	91.34	86.03	91.37
RF-S1	97.85	98.50	91.27	88.31	91.64	86.77	93.62	89.58	81.31	90.78
LR-S1	97.14	97.09	96.04	83.76	80.10	82.95	92.27	89.88	79.77	85.62

Table 5.11: Sensitivity values for all MNIST classes

MNIST	Sp0	Sp1	Sp2	Sp3	Sp4	Sp5	Sp6	Sp7	Sp8	Sp9
CNN-S3	**99.94**	**99.93**	**99.91**	**99.93**	**99.84**	99.83	**99.95**	99.86	**99.88**	99.05
RNN-S3	99.90	**99.93**	99.83	**99.93**	99.58	**99.90**	99.93	**99.95**	99.87	**99.95**
MLP-S3	99.77	**99.93**	99.71	99.54	99.76	99.76	99.77	99.77	99.71	99.69
SVM-S3	99.83	99.88	99.75	99.78	99.81	99.85	99.91	99.77	99.82	99.71
RF-S3	99.68	99.87	99.54	99.55	99.63	99.75	99.75	99.68	99.52	99.53
LR-S3	99.42	99.50	99.29	98.76	99.05	99.00	99.33	99.20	98.59	98.56
CNN-S2	99.91	99.94	**99.91**	99.91	**99.91**	99.85	**99.95**	99.88	99.82	99.86
RNN-S2	99.91	**99.93**	99.57	99.34	99.73	**99.95**	99.90	99.75	99.73	99.81
MLP-S2	99.68	99.85	99.71	99.59	99.75	99.76	99.69	99.53	99.46	99.68
SVM-S2	99.76	99.81	99.68	99.71	99.77	99.78	99.80	99.68	99.78	99.73
RF-S2	99.62	99.81	99.42	99.45	99.61	99.77	99.60	99.63	99.53	99.32
LR-S2	99.40	99.50	99.14	98.88	99.01	98.83	99.32	99.05	98.65	98.95
CNN-S1	**99.63**	99.72	**99.36**	**99.87**	99.67	**99.51**	**99.82**	**99.44**	**99.76**	98.87
RNN-S1	**99.63**	**99.81**	97.87	99.28	97.87	99.20	98.33	99.09	98.33	**98.88**
MLP-S1	99.24	99.32	98.67	99.07	98.73	98.04	99.12	98.99	99.01	98.24
SVM-S1	99.43	99.59	99.04	99.23	99.04	98.86	99.50	99.28	99.53	98.78
RF-S1	99.15	99.46	99.07	99.28	**99.70**	99.03	99.15	99.01	99.47	97.74
LR-S1	99.19	99.10	98.84	98.88	98.51	98.44	98.92	98.70	98.54	98.08

Table 5.12: Specificity values for all MNIST classes

CIFAR10	Se0	Se1	Se2	Se3	Se4	Se5	Se6	Se7	Se8	Se9
CNN-S3	**78.50**	**87.10**	**60.10**	**58.70**	**83.70**	**61.70**	**84.40**	**72.70**	**89.80**	**86.60**
RNN-S3	73.40	77.30	40.80	42.80	48.90	50.60	76.40	59.70	77.50	63.70
MLP-S3	60.10	69.40	39.40	36.50	47.90	40.60	69.70	60.20	73.70	59.30
RF-S3	54.31	50.79	38.16	32.30	40.58	42.20	47.58	51.85	58.25	47.44
LR-S3	49.60	49.50	27.20	25.20	29.10	35.80	50.40	44.90	53.70	47.50
CNN-S2	65.70	82.40	**62.70**	53.40	75.20	58.60	86.90	68.40	87.80	79.40
RNN-S2	62.80	62.90	40.40	44.60	41.40	43.00	67.00	65.40	72.20	65.50
MLP-S2	56.10	66.50	38.00	32.80	53.10	39.10	62.70	57.70	70.10	56.10
SVM-S2	61.10	66.00	42.00	38.20	42.70	44.10	54.80	57.10	65.50	61.70
RF-S2	51.87	49.10	35.71	33.37	37.68	42.01	44.65	49.03	54.74	45.49
LR-S2	50.00	49.00	26.90	24.90	27.50	34.10	51.60	45.70	53.80	45.80
CNN-S1	47.10	58.70	**33.20**	24.60	**50.40**	**51.40**	**70.60**	**60.80**	57.50	**64.50**
RNN-S1	43.70	**59.80**	29.70	19.50	34.40	33.20	50.00	36.80	**63.60**	39.40
MLP-S1	**50.00**	46.40	31.20	**27.80**	23.80	18.30	50.70	47.50	47.90	42.70
SVM-S1	**50.00**	48.00	31.00	24.40	35.80	30.10	40.40	40.70	54.90	48.90
RF-S1	41.92	44.50	24.66	21.83	29.14	35.14	35.10	42.32	45.91	41.76
LR-S1	44.90	40.90	24.60	17.20	28.90	28.20	48.10	32.90	52.40	45.50

Table 5.13: Sensitivity values for all CIFAR10 classes

CIFAR10	Sp0	Sp1	Sp2	Sp3	Sp4	Sp5	Sp6	Sp7	Sp8	Sp9
CNN-S3	**97.54**	**99.14**	**97.63**	**95.86**	94.15	**97.32**	**97.40**	**99.03**	**97.53**	**98.06**
RNN-S3	94.86	96.24	95.94	93.95	**95.91**	94.66	93.91	97.33	97.08	96.86
MLP-S3	96.00	95.95	95.11	93.38	94.12	95.75	93.20	96.34	94.77	96.10
RF-S3	95.09	94.85	92.84	92.05	93.35	93.20	95.13	93.95	95.68	94.78
LR-S3	93.34	93.57	94.01	93.34	94.56	92.74	91.84	94.43	93.70	93.20
CNN-S2	**98.27**	99.03	95.01	93.56	94.74	**97.35**	95.34	98.53	97.00	**98.27**
RNN-S2	95.88	97.56	94.33	91.43	**96.18**	94.63	94.54	95.31	96.71	95.07
MLP-S2	96.53	95.74	94.61	93.98	91.30	95.53	94.00	95.51	95.27	95.52
SVM-S2	95.54	95.22	93.38	92.27	95.03	94.08	95.63	95.90	96.08	94.95
RF-S2	94.85	94.62	92.53	92.05	93.11	93.22	94.75	93.78	95.38	94.43
LR-S2	93.44	93.43	93.62	93.52	94.86	93.17	91.87	93.84	93.63	93.21
CNN-S1	**97.30**	**97.02**	**94.62**	**95.16**	91.07	91.16	93.68	95.50	**96.73**	94.25
RNN-S1	95.51	93.63	91.72	93.24	92.27	93.63	91.95	**96.17**	91.14	**95.15**
MLP-S1	93.02	93.60	92.26	90.54	**94.95**	**97.05**	90.04	90.33	95.81	94.17
SVM-S1	93.56	92.36	92.25	91.83	92.51	93.58	**93.96**	95.04	95.22	93.44
RF-S1	94.23	93.52	91.72	90.71	92.87	92.32	93.43	92.54	94.61	93.75
LR-S1	93.15	94.31	91.67	94.27	93.00	93.22	90.50	94.85	92.31	91.97

Table 5.14: Specificity values for all CIFAR10 classes

IMDb	Se0	Se1	Sp0	Sp1
CNN-S3	75.87	81.01	81.01	75.87
RNN-S3	79.04	82.96	82.96	79.04
MLP-S3	78.61	84.66	84.66	78.61
SVM-S3	87.82	89.44	89.44	87.82
RF-S3	84.90	86.33	86.33	84.90
LR-S3	**88.27**	**89.65**	**89.65**	**88.27**
CNN-S2	81.66	80.94	80.94	81.66
RNN-S2	84.32	75.01	75.01	84.32
MLP-S2	76.30	85.42	85.42	76.30
SVM-S2	87.18	88.70	88.70	87.18
RF-S2	84.60	85.67	85.67	84.60
LR-S2	87.31	88.74	88.74	87.31
CNN-S1	75.87	81.01	81.01	75.87
RNN-S1	77.25	73.40	73.40	77.25
MLP-S1	77.77	74.79	74.79	77.77
SVM-S1	83.64	**84.60**	**84.60**	83.64
RF-S1	78.81	84.41	84.41	78.81
LR-S1	**84.30**	83.75	83.75	**84.30**

Table 5.15: Sensitivity, specificity values for all IMDb classes

Sent.140	Se0	Se1	Sp0	Sp1
CNN-S3	75.67	76.36	76.36	75.67
RNN-S3	**78.66**	71.01	71.01	**78.66**
MLP-S3	70.95	77.10	77.10	70.95
SVM-S3	73.62	**79.70**	**79.70**	73.62
RF-S3	71.04	78.40	78.40	71.04
LR-S3	74.39	78.35	78.35	74.39
CNN-S2	75.70	73.58	73.58	75.70
RNN-S2	**79.47**	68.59	68.59	**79.47**
MLP-S2	66.98	76.85	76.85	66.98
SVM-S2	73.62	79.70	79.70	73.62
RF-S2	69.18	78.71	78.71	69.18
LR-S2	73.13	78.22	78.22	73.13
CNN-S1	22.13	**89.06**	**89.06**	22.13
RNN-S1	57.99	75.42	75.42	57.99
MLP-S1	54.43	70.38	70.38	54.43
SVM-S1	63.00	72.33	72.33	63.00
RF-S1	57.70	75.21	75.21	57.70
LR-S1	**63.18**	72.35	72.35	**63.18**

Table 5.16: Sensitivity, specificity values for all Sentiment140 classes

HBS	Se0	Se1	Sp0	Sp1
CNN-S3	**92.42**	**82.89**	**82.89**	**92.42**
RNN-S3	84.09	23.68	23.68	84.09
MLP-S3	91.66	42.10	42.10	91.66
SVM-S3	75.75	59.21	59.21	75.75
RF-S3	70.95	72.22	72.22	70.95
LR-S3	79.54	55.26	55.26	79.54
CNN-S2	90.15	68.42	68.42	90.15
RNN-S2	74.24	34.21	34.21	74.24
MLP-S2	67.42	59.21	59.21	67.42
SVM-S2	67.42	47.36	47.36	67.24
RF-S2	71.09	61.11	61.11	71.09
LR-S2	71.21	55.26	55.26	71.21
CNN-S1	**83.33**	**68.42**	**68.42**	**83.33**
RNN-S1	53.03	46.05	46.05	53.03
MLP-S1	78.03	48.68	48.68	78.03
SVM-S1	68.93	53.94	53.94	68.93
RF-S1	65.78	61.11	61.11	65.78
LR-S1	75.75	52.63	52.63	75.75

Table 5.17: Sensitivity, specificity values for all Heartbeat sound classes

ESC-50	CNN	RNN	MLP	SVM	RF	LR
Se0	**68.42**	5.26	10.52	21.05	5.26	15.78
Se1	**70.58**	35.29	29.41	35.29	23.52	23.52
Se2	**77.77**	38.88	61.11	61.11	44.44	27.77
Se3	**100**	76.92	76.92	69.23	61.53	53.84
Se4	**63.15**	36.84	52.63	57.89	36.84	57.89
Se5	**93.33**	93.33	86.66	46.66	86.66	73.33
Se6	**100**	46.15	69.23	69.23	69.23	23.07
Se7	50.00	40.00	50.00	**60.00**	**60.00**	40.00
Se8	**82.35**	35.29	23.52	47.05	58.82	47.05
Se9	**83.33**	44.44	27.77	44.44	38.88	55.55
Se10	75.00	68.75	81.25	68.75	**87.50**	75.00
Se11	**85.71**	64.28	50.00	50.00	50.00	57.14
Se12	**93.75**	56.25	62.50	56.25	37.50	50.00
Se13	**82.35**	47.05	41.17	17.64	23.52	23.52
Se14	**94.11**	41.17	35.29	41.17	47.05	41.17
Se15	**93.33**	80.00	80.00	86.66	86.66	60.00
Se16	64.28	21.42	28.57	28.57	64.28	**78.57**
Se17	83.33	**100**	**100**	83.33	75.00	75.00
Se18	63.63	59.09	50.00	45.45	22.72	**68.18**
Se19	**83.33**	83.33	41.66	58.33	75.00	33.33
Se20	57.14	64.28	**78.57**	64.28	**78.57**	85.71
Se21	**82.35**	76.47	64.70	64.70	58.82	58.82
Se22	44.44	**50.00**	44.44	11.11	22.22	22.22
Se23	65.00	**95.00**	90.00	85.00	65.00	90.00

Table 5.18: Sensitivity values for all ESC-50 classes

ESC-50	CNN	RNN	MLP	SVM	RF	LR
Sp0	98.07	98.07	**98.62**	96.97	**98.62**	97.52
Sp1	**99.18**	97.26	95.08	97.81	98.90	99.18
Sp2	**99.17**	99.72	96.98	96.16	98.63	97.53
Sp3	**99.72**	97.83	99.18	97.02	97.56	97.02
Sp4	**99.45**	98.90	98.07	97.52	98.07	97.52
Sp5	99.18	98.09	**99.45**	97.82	95.38	96.73
Sp6	**99.45**	**99.45**	98.37	95.13	98.37	96.21
Sp7	**100**	98.39	98.65	98.92	99.19	98.92
Sp8	**100**	98.08	97.81	99.72	98.63	98.36
Sp9	99.17	98.90	98.90	**99.72**	98.90	98.63
Sp10	**100**	98.63	97.27	98.09	98.91	97.82
Sp11	**99.45**	96.74	95.66	99.18	97.28	98.37
Sp12	97.82	98.36	**99.45**	98.63	98.09	98.63
Sp13	98.90	98.36	**99.18**	98.36	**99.18**	98.36
Sp14	**98.90**	96.17	96.99	97.54	97.81	98.63
Sp15	95.65	97.28	96.73	97.01	97.01	**97.55**
Sp16	99.18	99.18	**99.45**	98.10	96.74	96.20
Sp17	98.92	98.92	**99.19**	97.30	96.22	98.38
Sp18	**99.44**	99.16	99.16	98.33	99.44	98.61
Sp19	95.14	92.18	94.33	**97.30**	94.87	96.76
Sp20	**99.72**	98.64	98.64	98.64	97.28	99.45
Sp21	**100**	99.18	99.72	98.90	99.18	99.18
Sp22	**100**	98.35	96.71	98.90	97.20	96.71
Sp23	**99.17**	97.79	98.89	96.69	97.24	96.96

Table 5.19: Specificity values for all ESC-50 classes

UCMF	Se0	Se1	Sp0	Sp1
CNN-S3	85.96	98.09	98.09	85.96
RNN-S3	85.38	98.04	98.04	85.38
MLP-S3	85.19	**98.37**	**98.37**	85.19
SVM-S3	85.96	98.15	98.15	85.96
RF-S3	**86.34**	98.26	98.26	**86.34**
LR-S3	85.96	**98.37**	**98.37**	85.96
CNN-S1	84.80	97.17	97.17	84.80
RNN-S1	84.23	98.04	98.04	84.23
MLP-S1	84.23	98.09	98.09	84.23
SVM-S1	85.19	**98.47**	**98.47**	85.19
RF-S1	**85.96**	98.26	98.26	**85.96**
LR-S1	85.57	98.37	98.37	85.57

Table 5.20: Sensitivity, specificity values for all UCMF classes

Mammo	Se0	Se1	Se2	Se3	Se4	Se5	Se6	Se7
CNN-S3	62.03	**82.12**	4.57	**2.56**	32.43	7.29	38.88	21.42
RNN-S3	60.45	79.64	5.22	**2.56**	48.64	17.82	3.70	3.57
MLP-S3	64.41	77.54	0.65	0.00	18.91	9.90	9.25	3.57
SVM-S3	62.65	81.55	0.00	0.00	0.00	6.93	0.00	0.00
RF-S3	**69.46**	68.82	**62.50**	0.00	**75.00**	**28.78**	**41.66**	**71.42**
LR-S3	59.92	80.53	13.01	0.00	29.27	5.94	0.00	0.00
CNN-S2	53.33	86.95	13.00	2.56	24.32	9.90	12.96	3.57
RNN-S2	61.86	79.45	1.30	0.00	37.83	14.85	20.37	10.71
MLP-S2	46.22	**88.48**	0.00	0.00	2.70	0.99	7.40	10.71
SVM-S2	58.26	82.95	0.00	0.00	0.00	6.93	0.00	0.00
RF-S2	**69.62**	67.71	55.00	0.00	50.00	23.22	20.00	42.85
LR-S2	56.23	82.31	1.30	0.00	29.72	4.95	1.85	0.00
CNN-S1	70.03	67.43	15.68	2.56	35.13	5.94	**25.92**	3.57
RNN-S1	**78.73**	62.08	11.11	0.00	16.21	3.96	9.25	10.71
MLP-S1	52.98	84.09	3.26	2.56	29.72	1.98	12.96	7.14
SVM-S1	58.26	80.47	0.00	0.00	32.43	0.00	0.00	7.14
RF-S1	68.21	66.99	**42.42**	**33.33**	**50.00**	**21.42**	23.07	**44.44**
LR-S1	58.08	**80.78**	1.96	0.00	24.32	4.95	1.85	21.42

Table 5.21: Sensitivity values for all Mammo classes

Mammo	Sp0	Sp1	Sp2	Sp3	Sp4	Sp5	Sp6	Sp7
CNN-S3	**82.51**	61.41	99.76	99.87	99.64	**99.27**	97.62	99.74
RNN-S3	81.50	60.25	98.98	99.90	99.05	97.91	99.51	99.64
MLP-S3	76.76	59.93	99.89	**100**	99.83	98.84	99.34	99.96
SVM-S3	80.19	55.22	**100**	**100**	**100**	98.94	**99.96**	**100**
RF-S3	81.36	**78.95**	95.39	98.75	99.09	97.31	98.42	99.26
LR-S3	79.68	55.22	99.79	**100**	99.54	99.04	99.70	**100**
CNN-S2	86.08	50.12	99.83	99.83	99.51	99.43	99.51	99.61
RNN-S2	80.34	60.25	99.62	99.96	99.05	98.47	99.11	99.87
MLP-S2	**87.14**	42.32	99.76	100	99.87	99.63	99.64	99.54
SVM-S2	80.59	**82.83**	99.96	100	100	98.97	**100**	100
RF-S2	80.04	79.58	95.42	98.75	99.06	97.12	98.32	99.19
LR-S2	80.59	52.64	99.76	100	99.67	**99.53**	99.57	99.80
CNN-S1	70.86	71.29	97.87	99.67	99.15	99.50	98.50	99.25
RNN-S1	63.91	74.38	97.94	100	99.51	99.93	99.57	99.64
MLP-S1	**84.52**	50.77	99.25	99.93	99.41	**100**	99.69	99.45
SVM-S1	77.41	54.51	**100**	**100**	99.48	**100**	**99.96**	**99.93**
RF-S1	79.73	**77.36**	95.50	98.74	99.09	96.92	98.44	99.22
LR-S1	80.29	54.06	99.49	**100**	**99.77**	99.60	99.77	99.25

Table 5.22: Specificity values for all Mammo classes

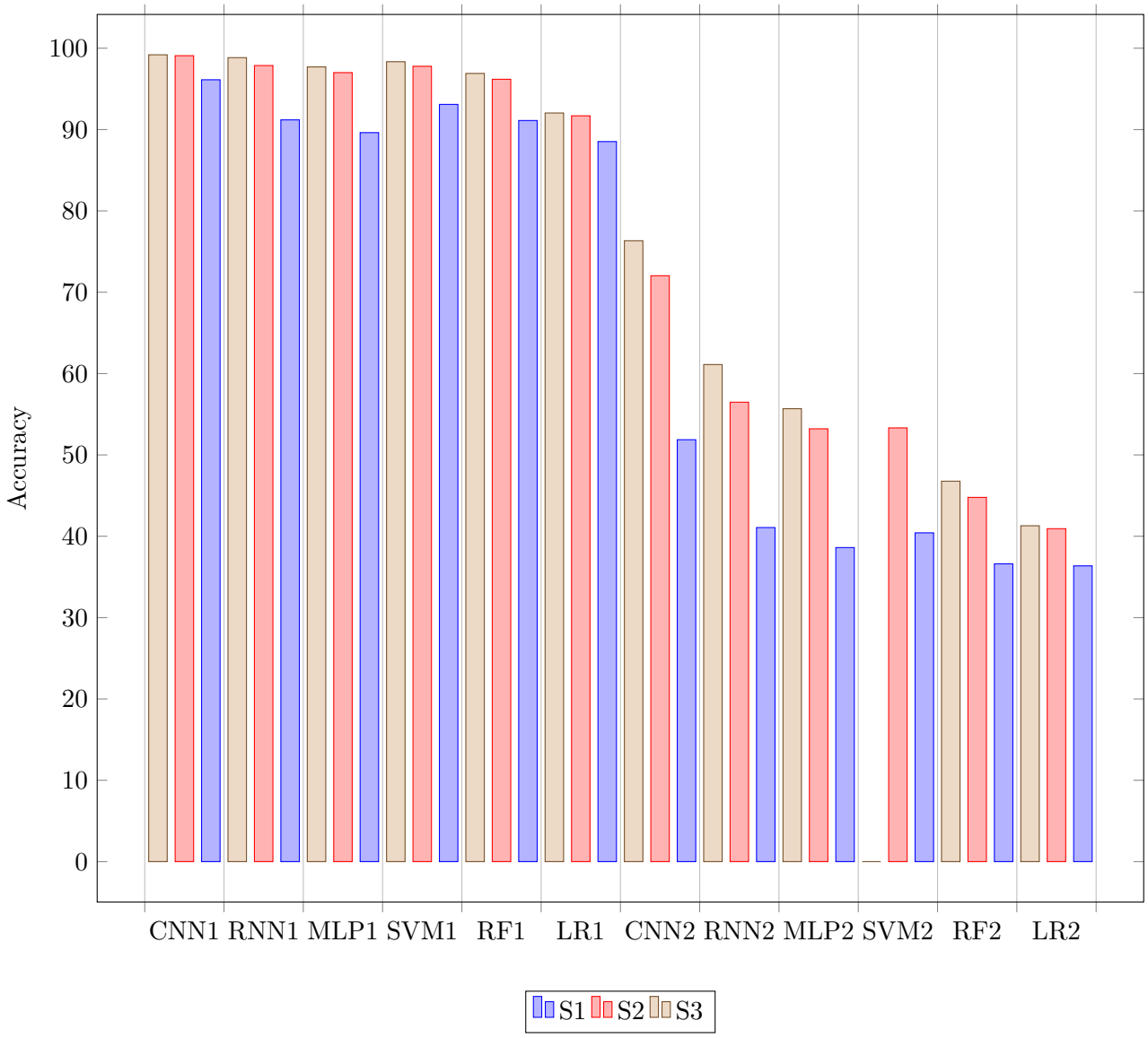

Figure 5.1: Accuracy results for the MNIST (1) and CIFAR10 (2) datasets, for the three dataset sizes

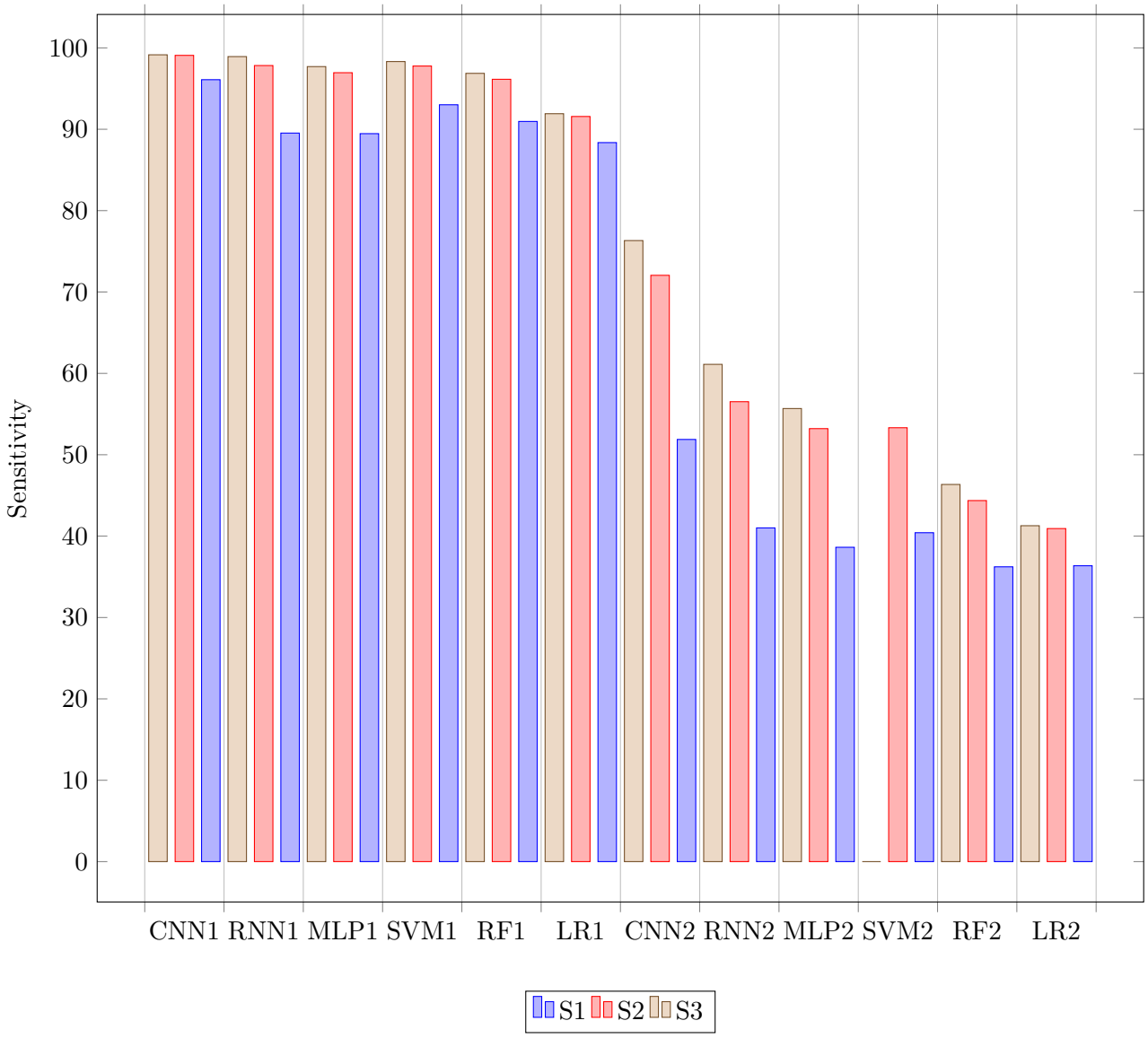

Figure 5.2: Sensitivity results for the MNIST (1) and CIFAR10 (2) datasets, for the three dataset sizes

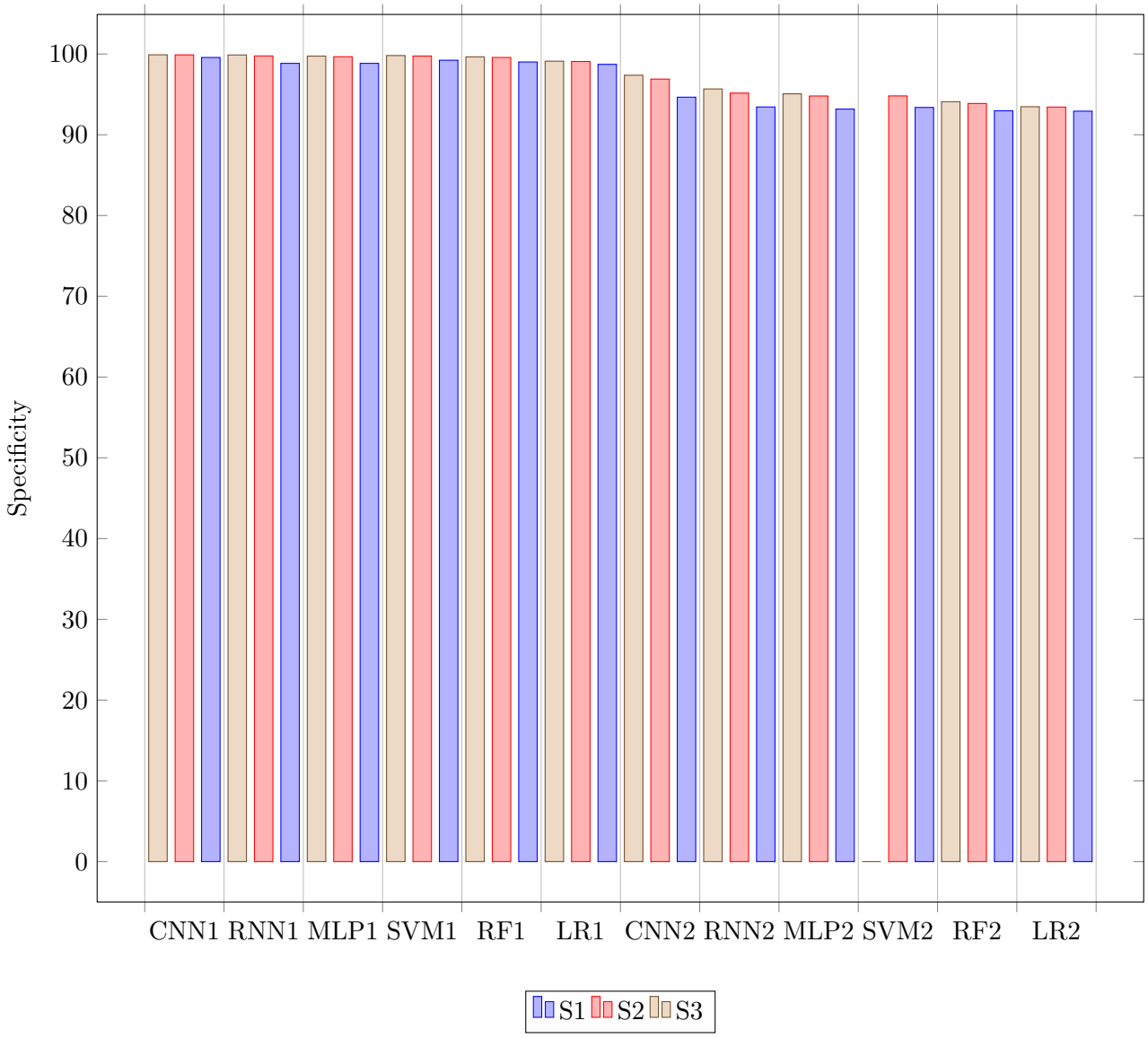

Figure 5.3: Specificity results for the MNIST (1) and CIFAR10 (2) datasets, for the three dataset sizes

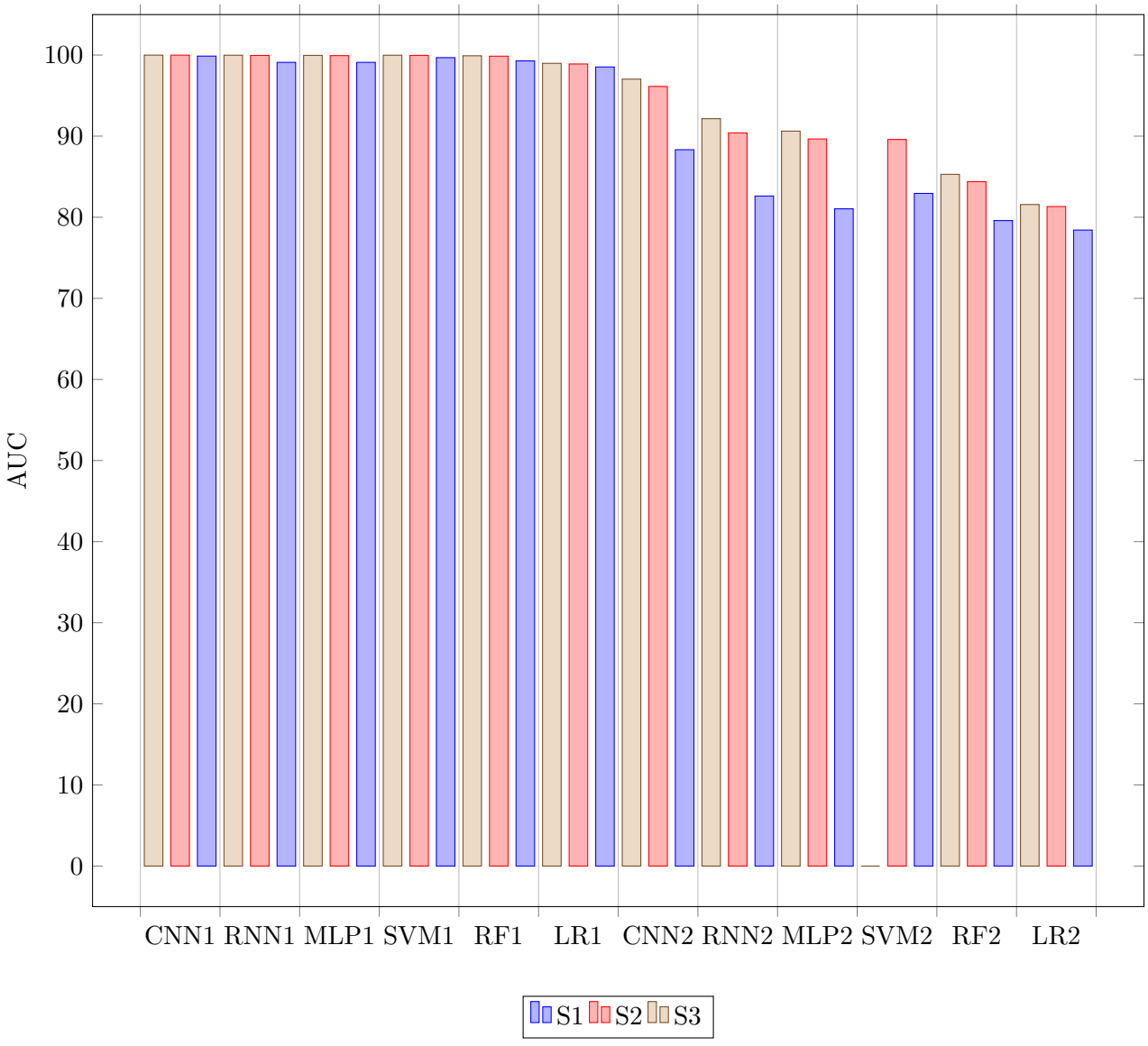

Figure 5.4: AUC results for the MNIST (1) and CIFAR10 (2) datasets, for the three dataset sizes

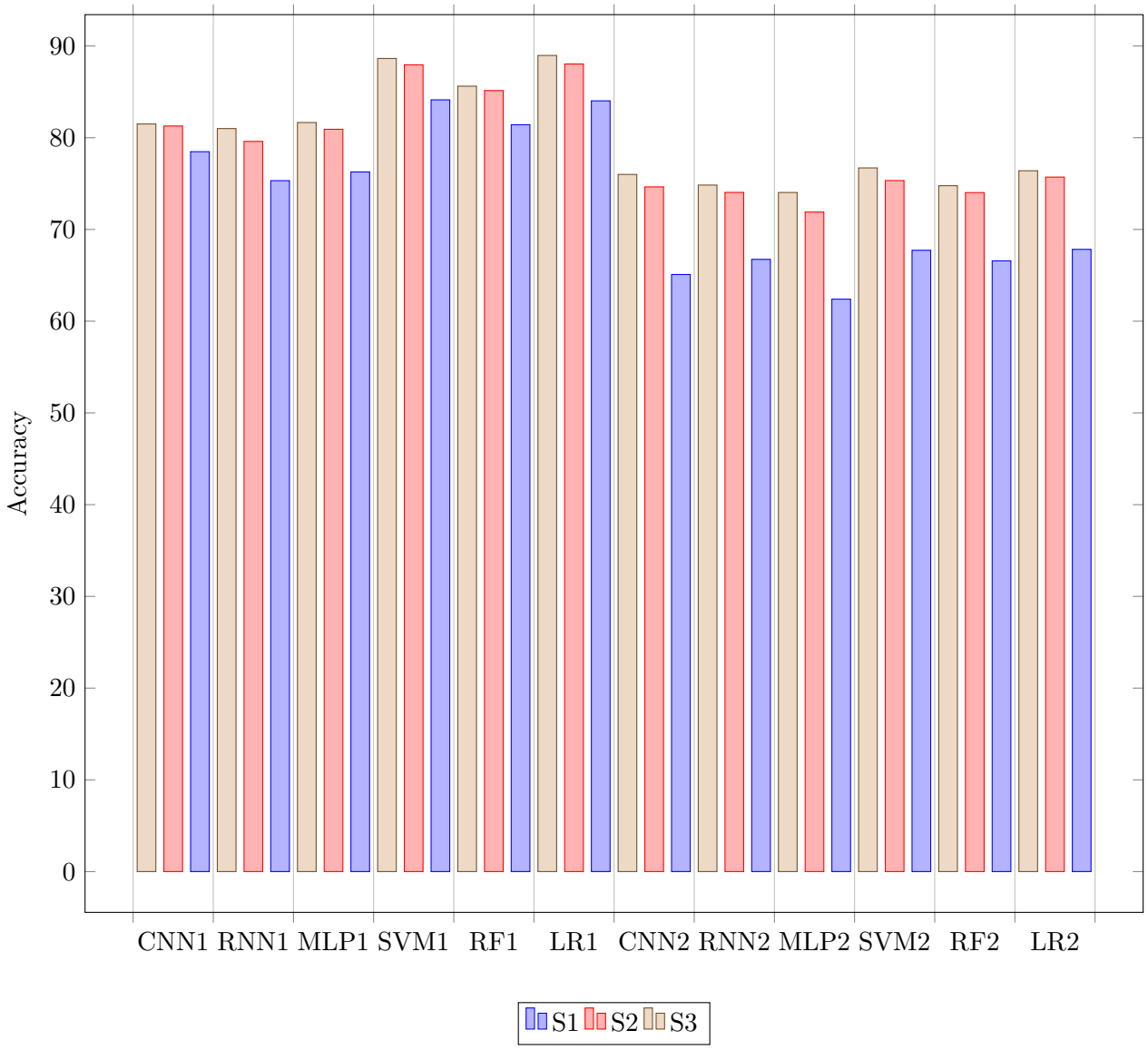

Figure 5.5: Accuracy results for the IMDb (1) and Sentiment140 (2) datasets, for the three dataset sizes

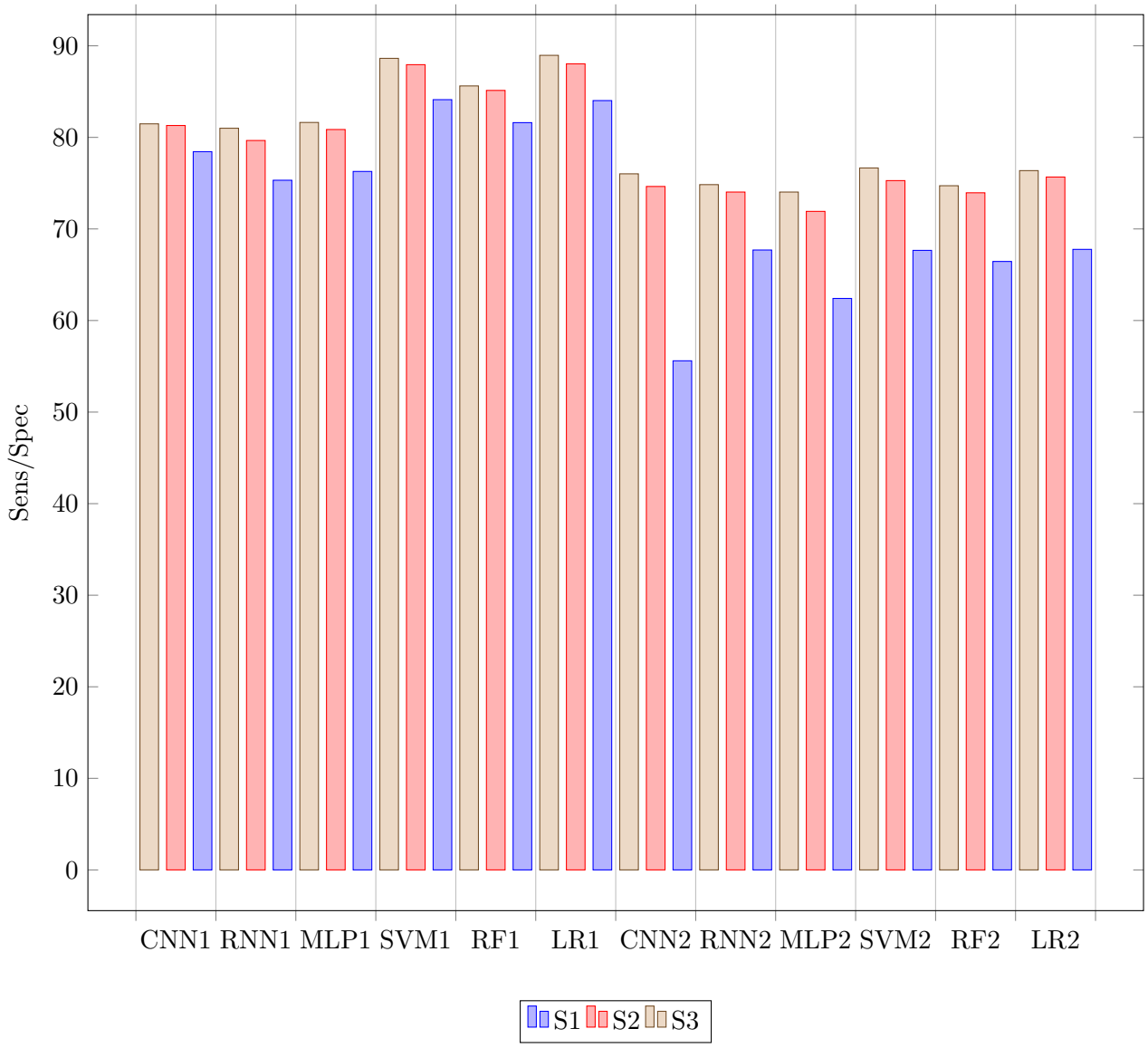

Figure 5.6: Sensitivity and specificity results for the IMDb (1) and Sentiment140 (2) datasets, for the three dataset sizes

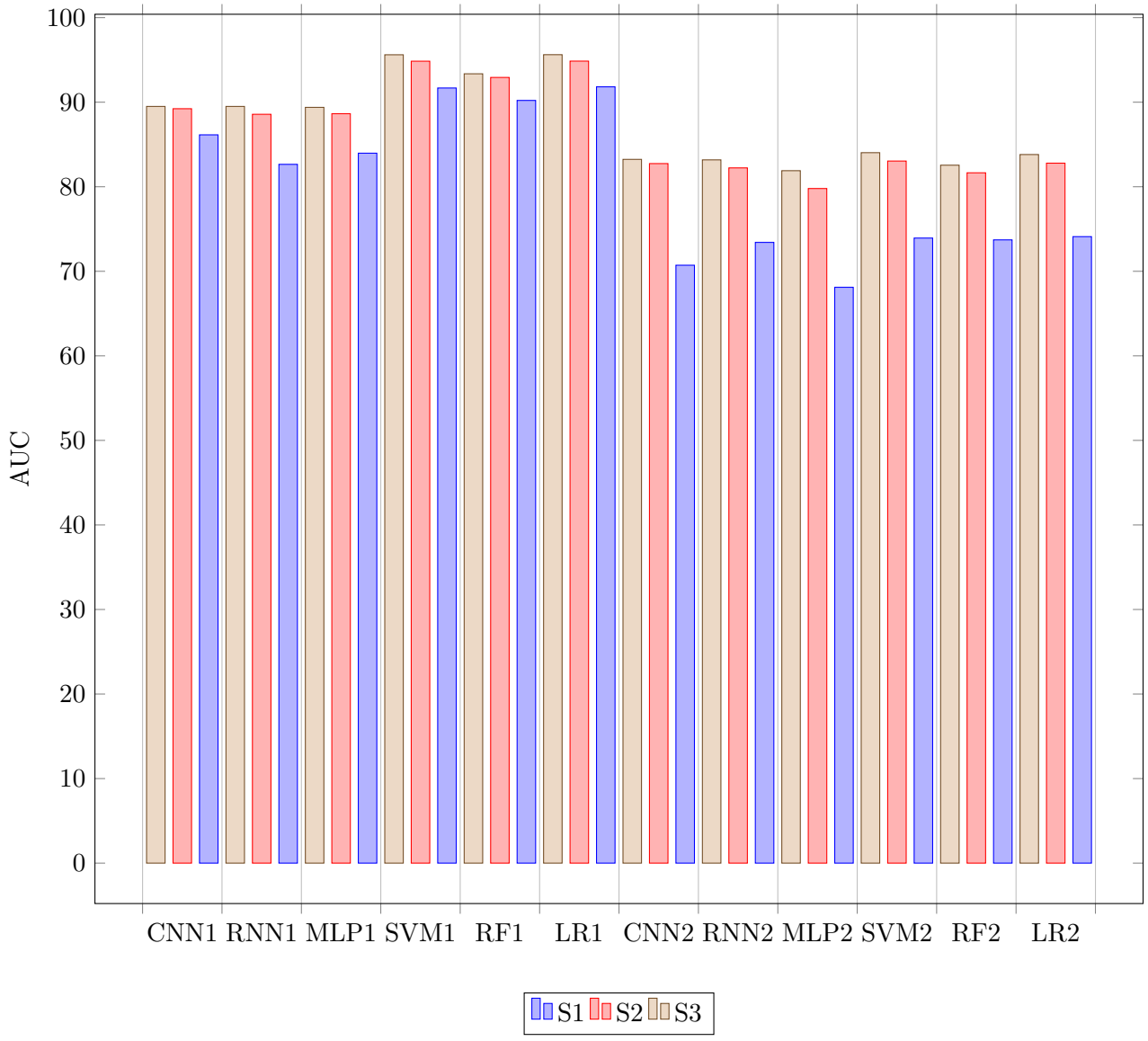

Figure 5.7: AUC results for the IMDb (1) and Sentiment140 (2) datasets, for the three dataset sizes

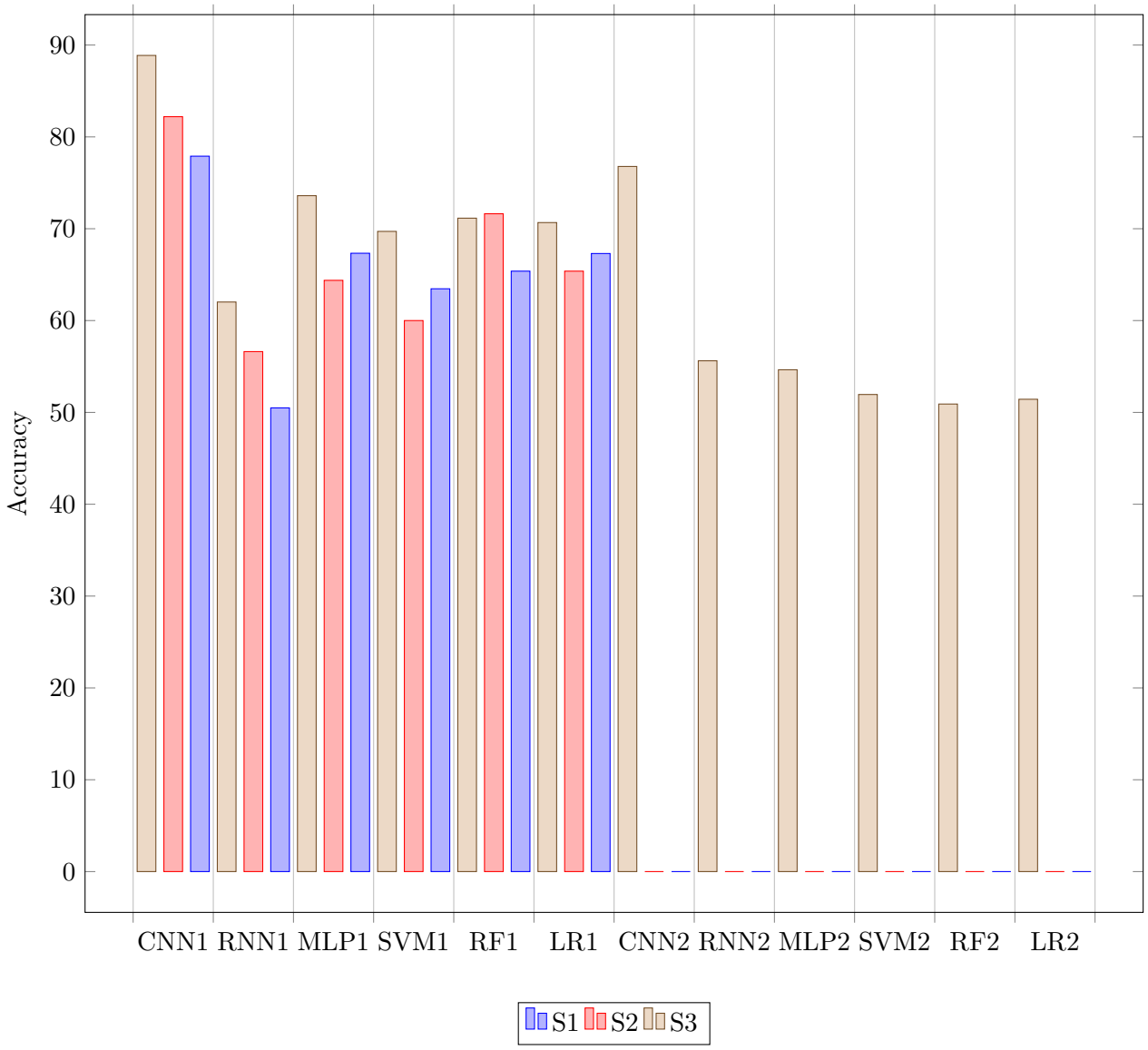

Figure 5.8: Accuracy results for the Heartbeat Sound (1) and ESC-50 (2) datasets, for the three dataset sizes

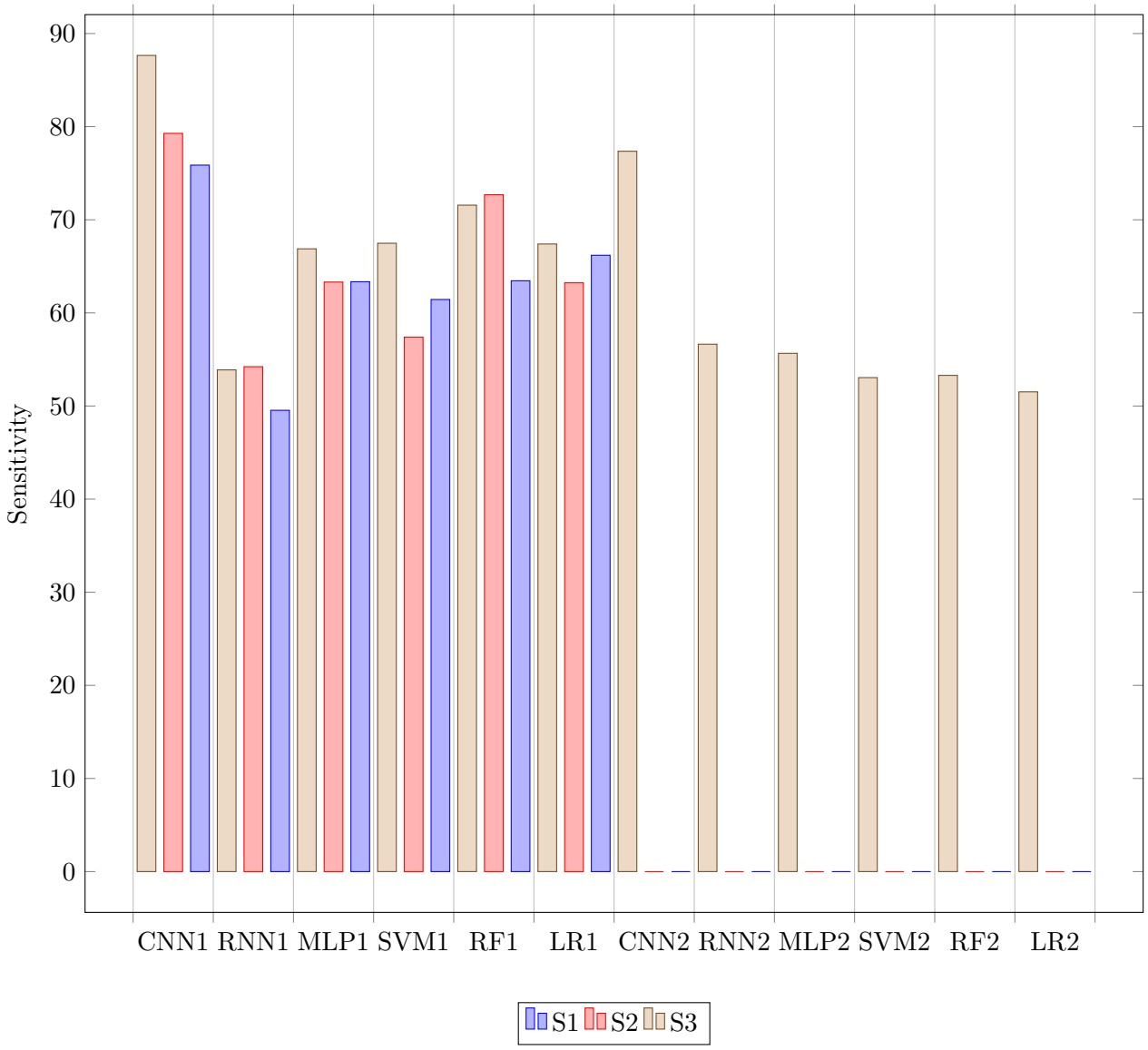

Figure 5.9: Sensitivity results for the Heartbeat Sound (1) and ESC-50 (2) datasets, for the three dataset sizes

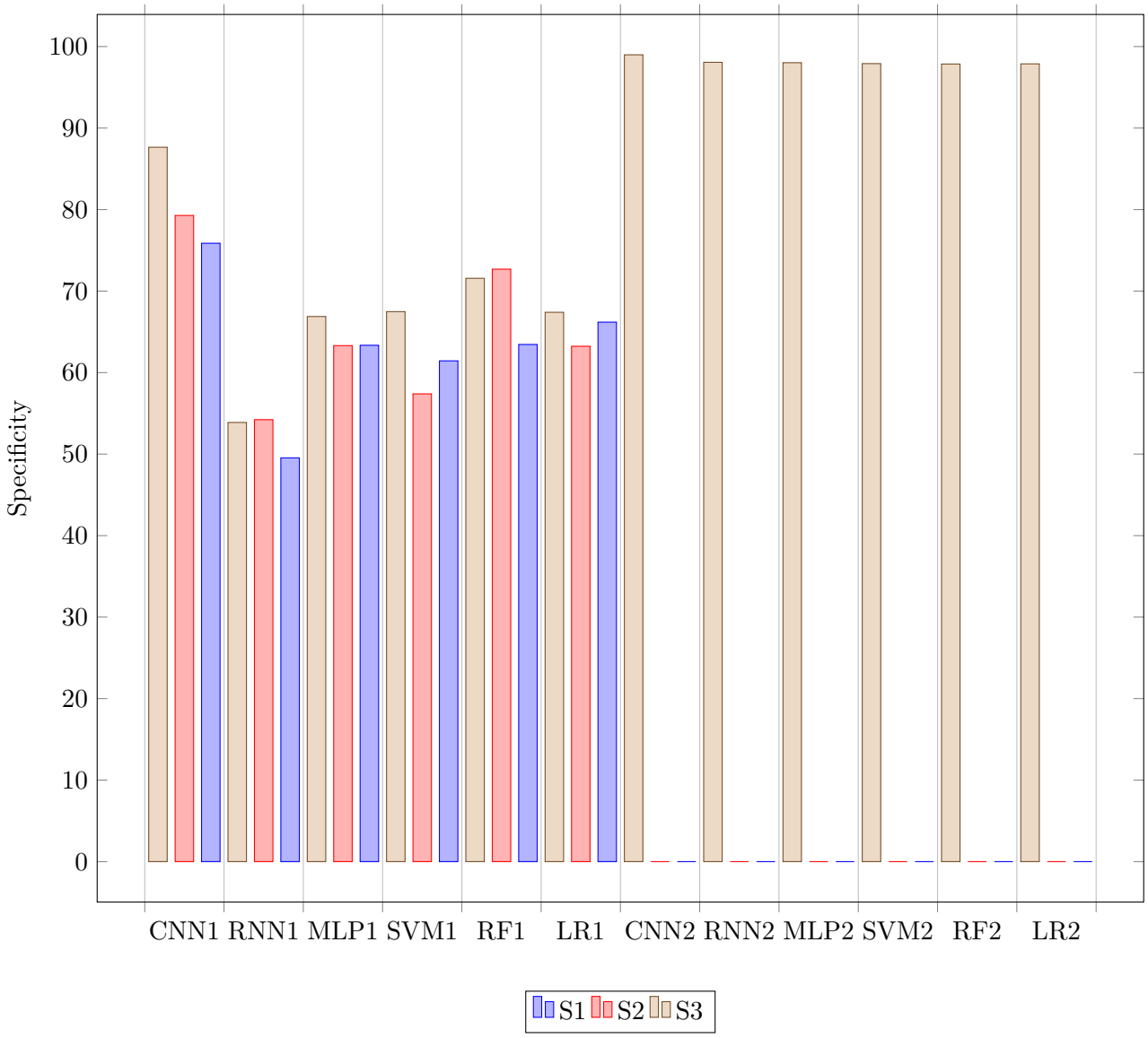

Figure 5.10: Specificity results for the Heartbeat Sound (1) and ESC-50 (2) datasets, for the three dataset sizes

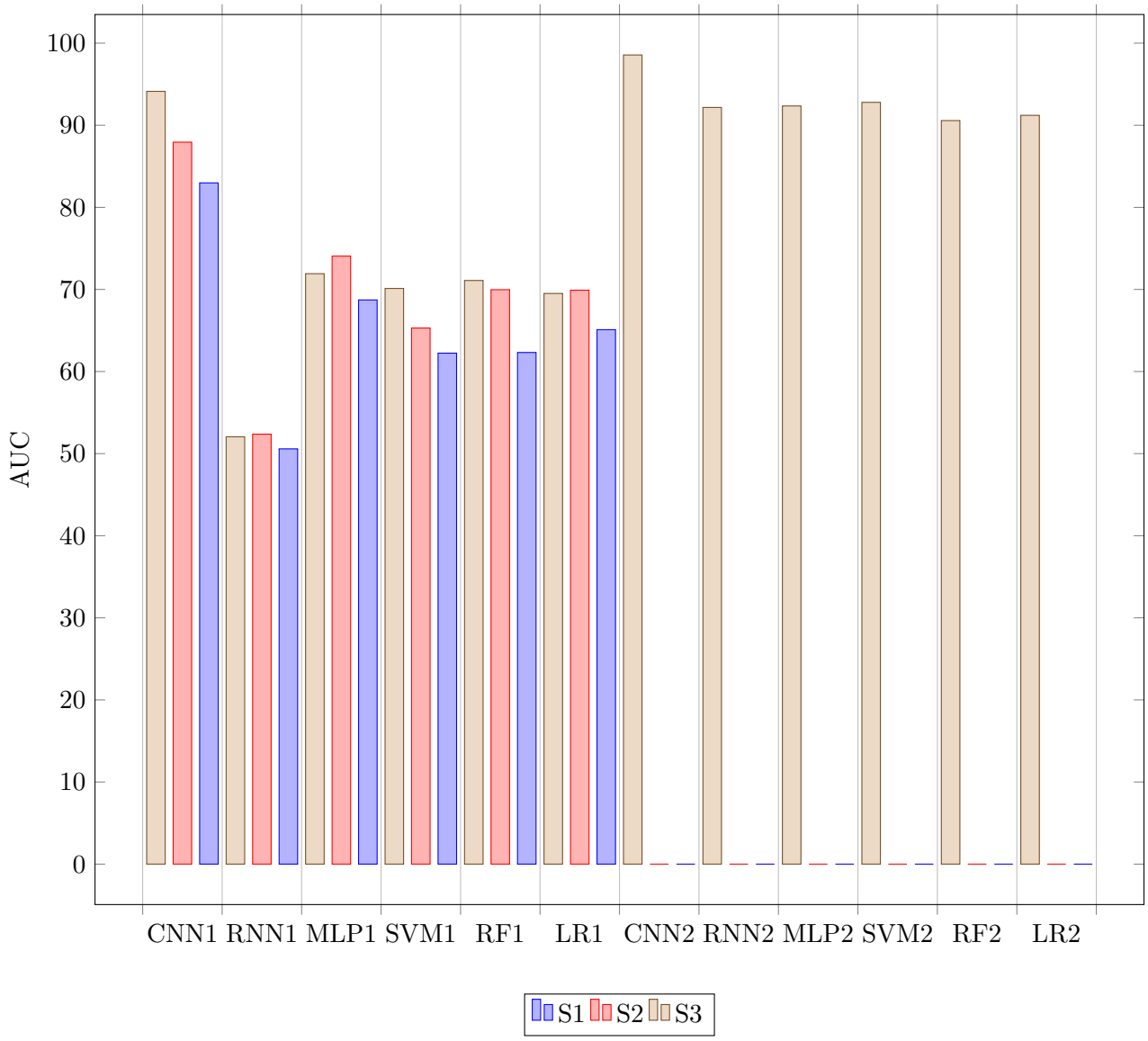

Figure 5.11: AUC results for the Heartbeat Sound (1) and ESC-50 (2) datasets, for the three dataset sizes

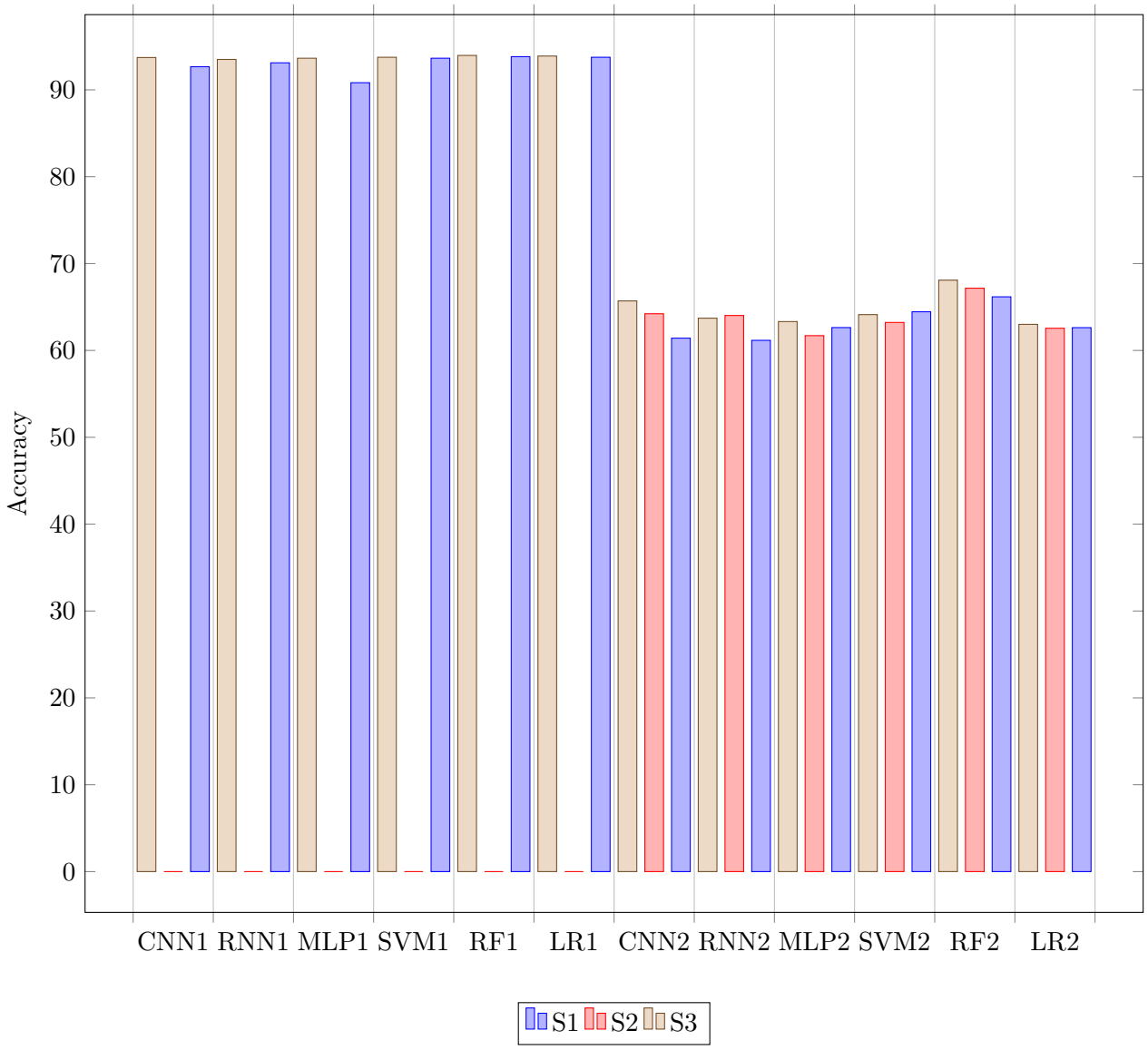

Figure 5.12: Accuracy results for the UCMF (1) and Mammo (2) datasets, for the three dataset sizes

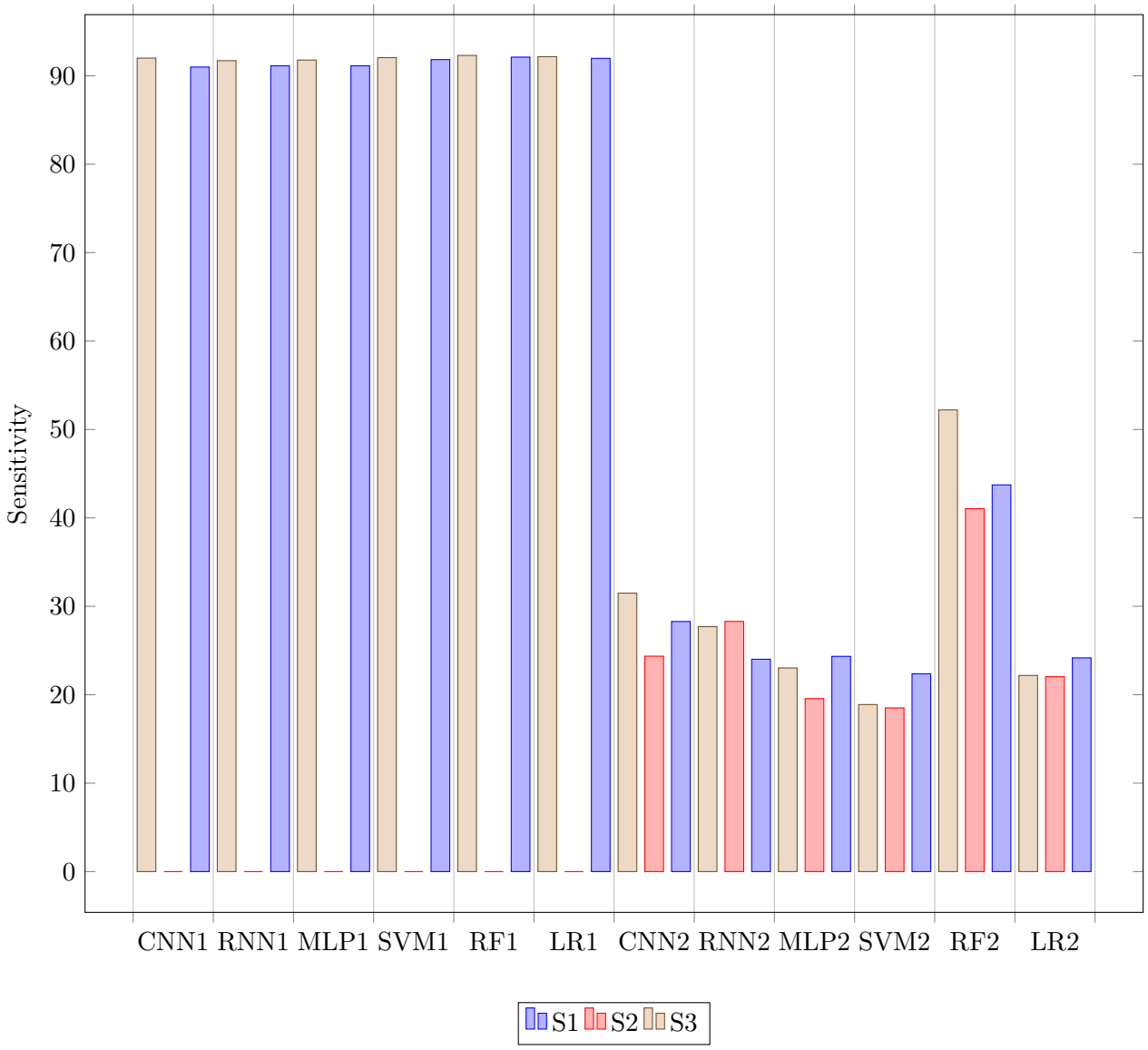

Figure 5.13: Sensitivity results for the UCMF (1) and Mammo (2) datasets, for the three dataset sizes

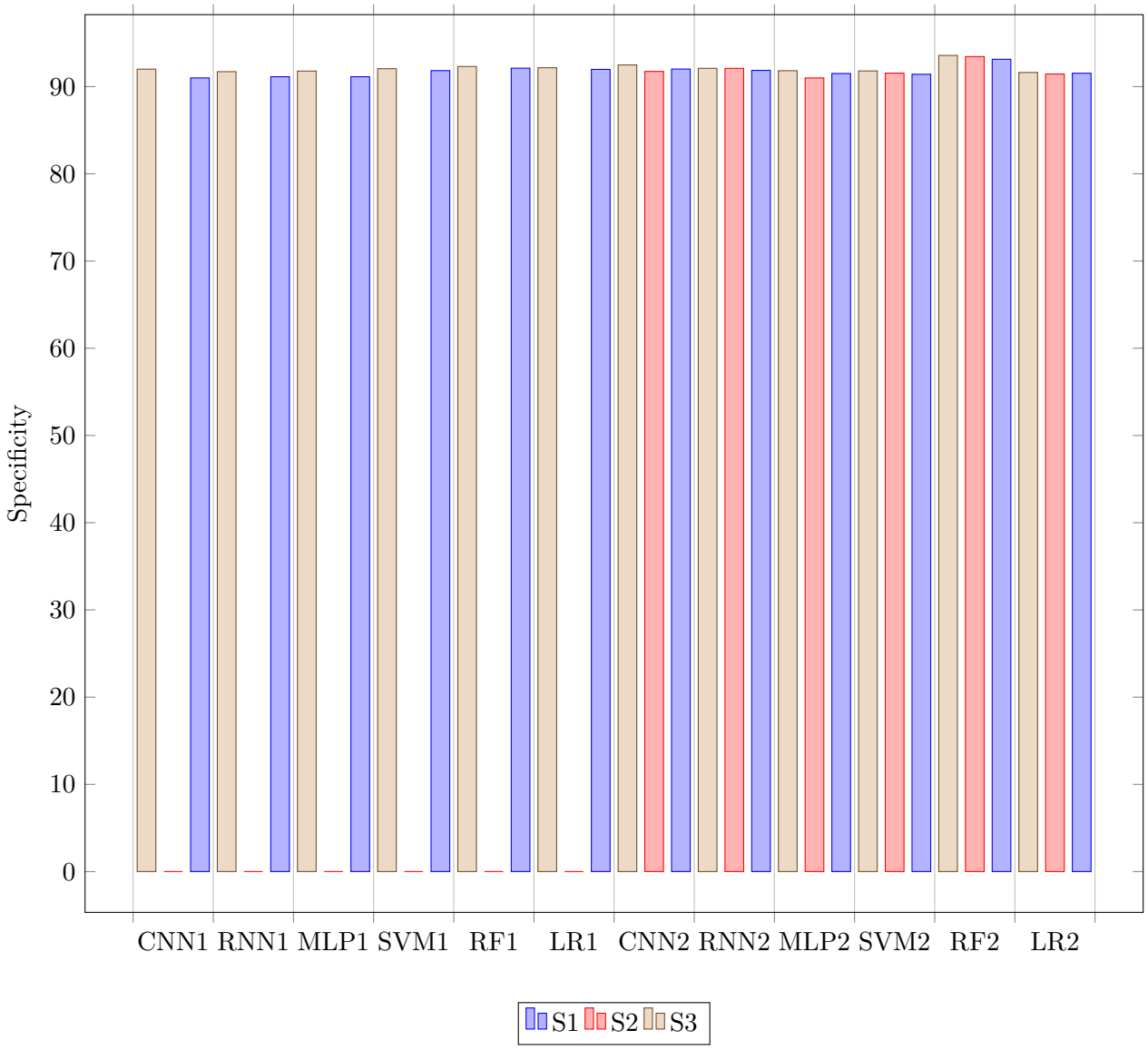

Figure 5.14: Specificity results for the UCMF (1) and Mammo (2) datasets, for the three dataset sizes

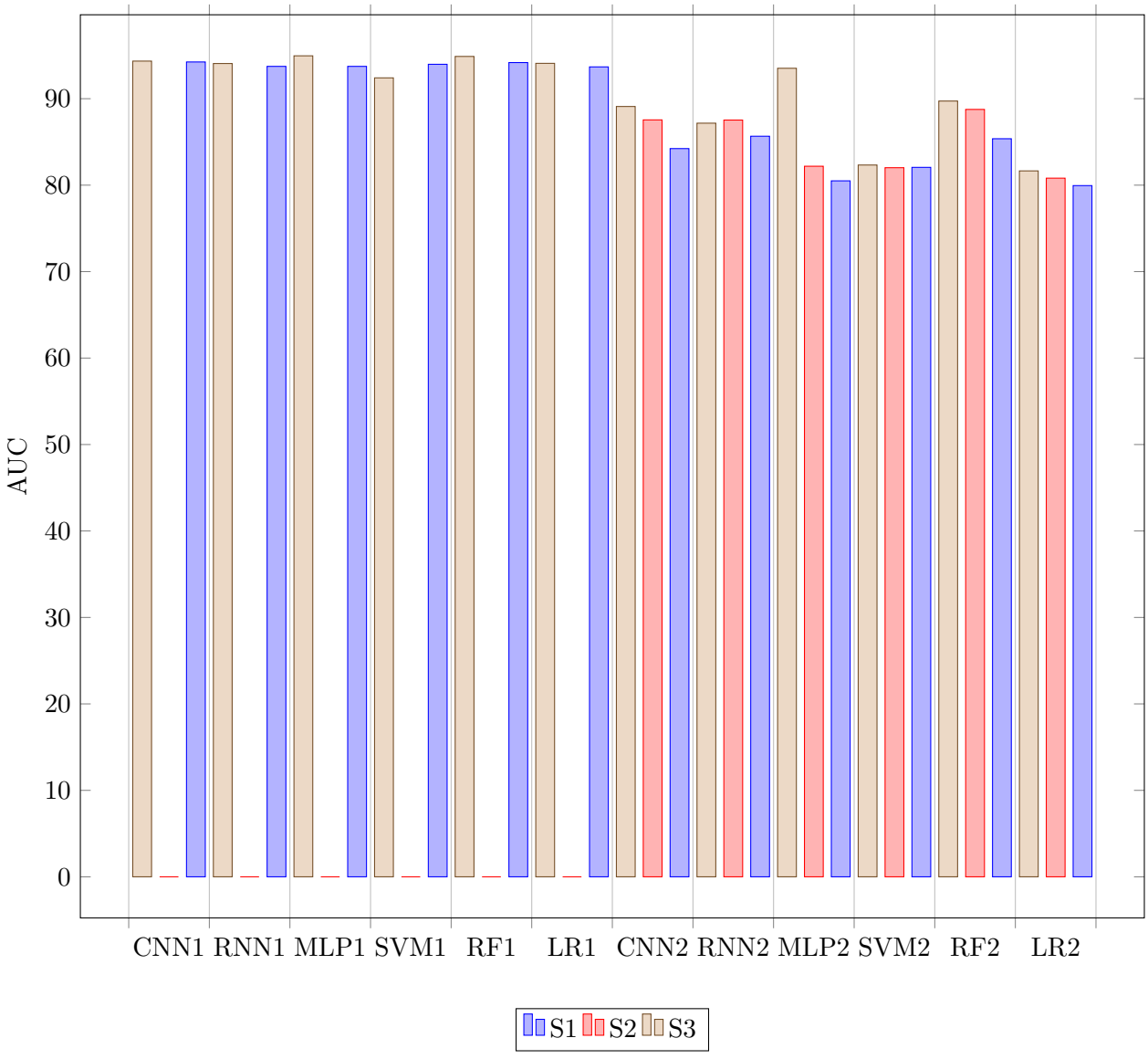

Figure 5.15: AUC results for the UCMF (1) and Mammo (2) datasets, for the three dataset sizes

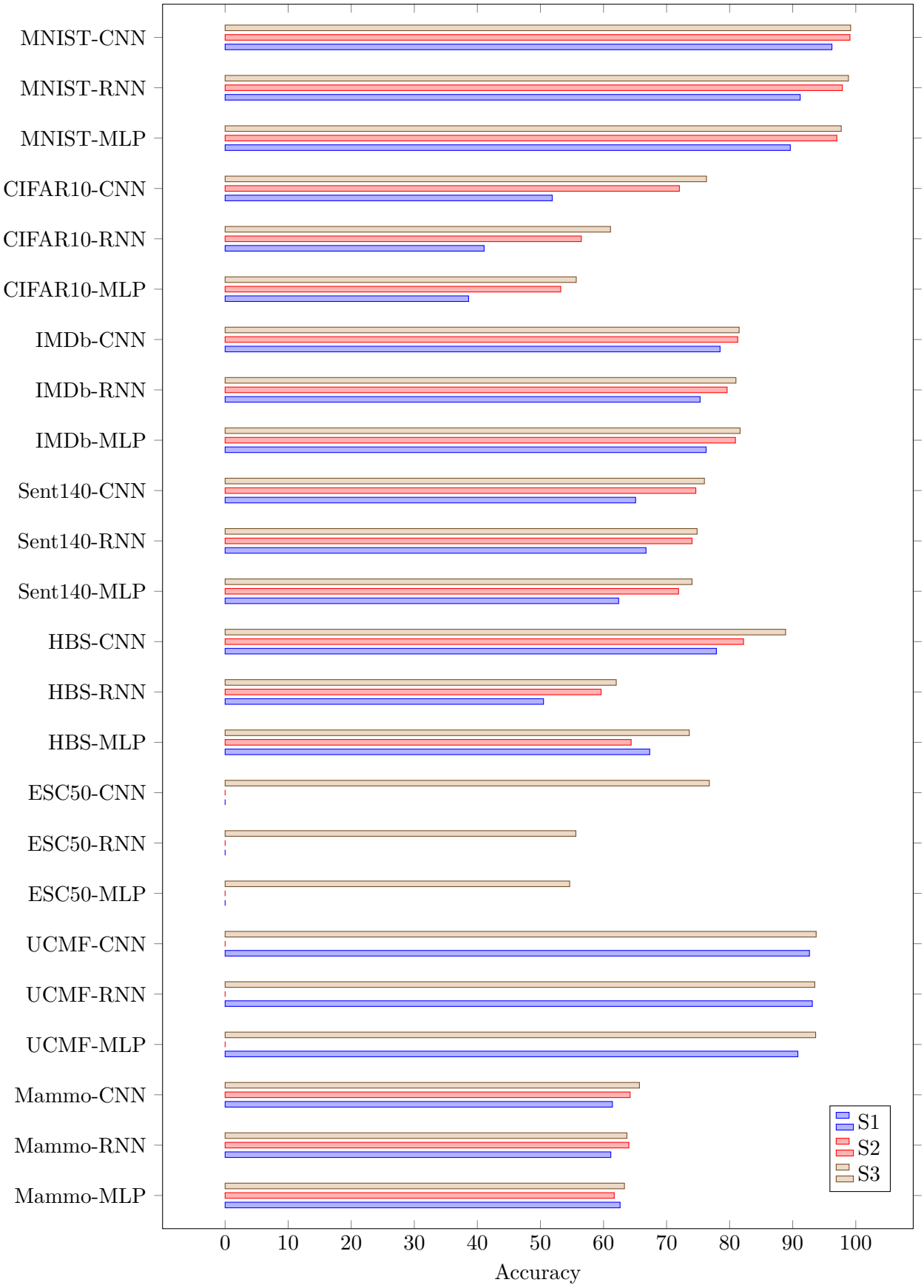

Figure 5.16: Accuracy for all three artificial neural network types applied to all dataset and dataset sizes

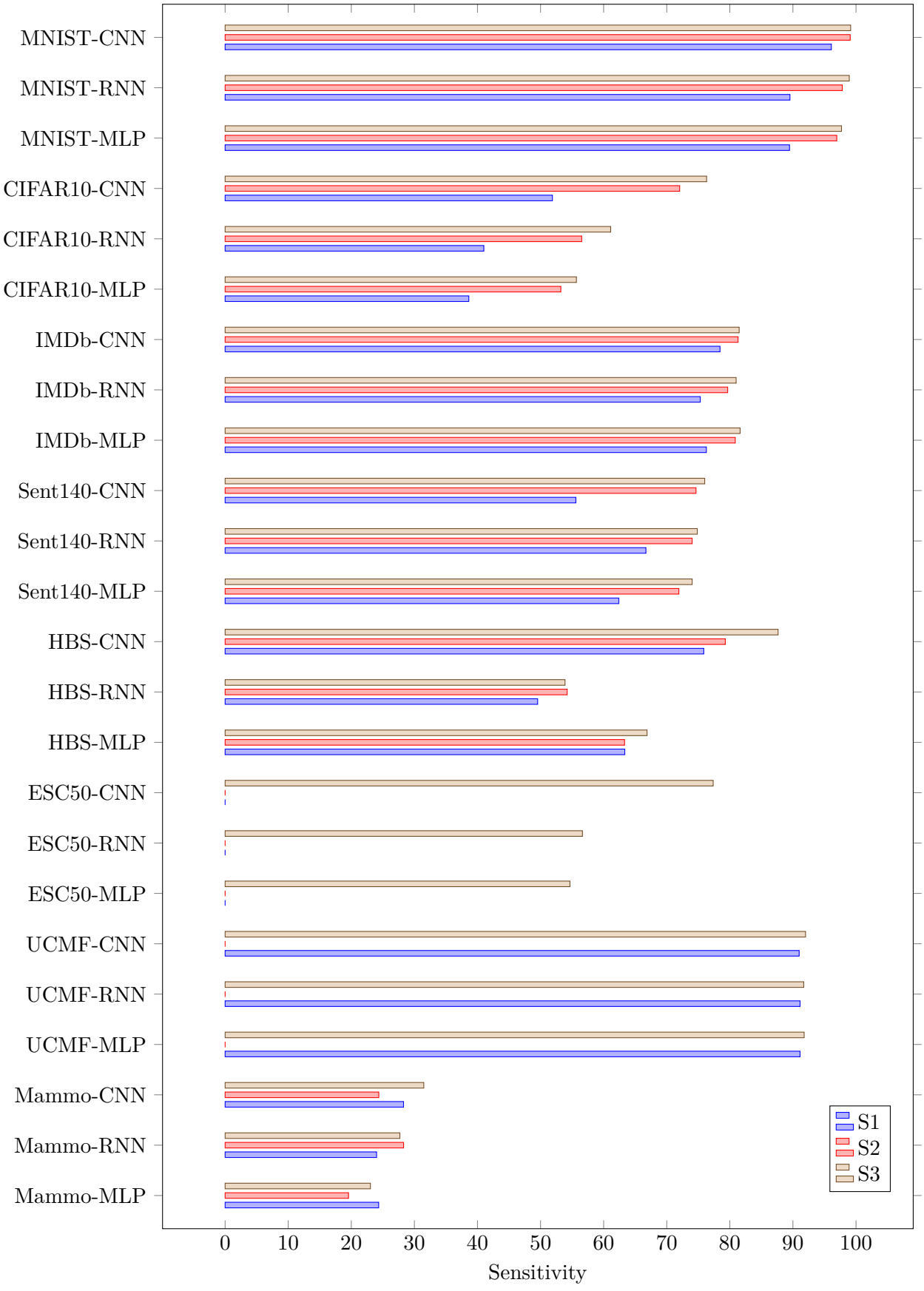

Figure 5.17: Sensitivity for all three artificial neural network types applied to all dataset and dataset sizes

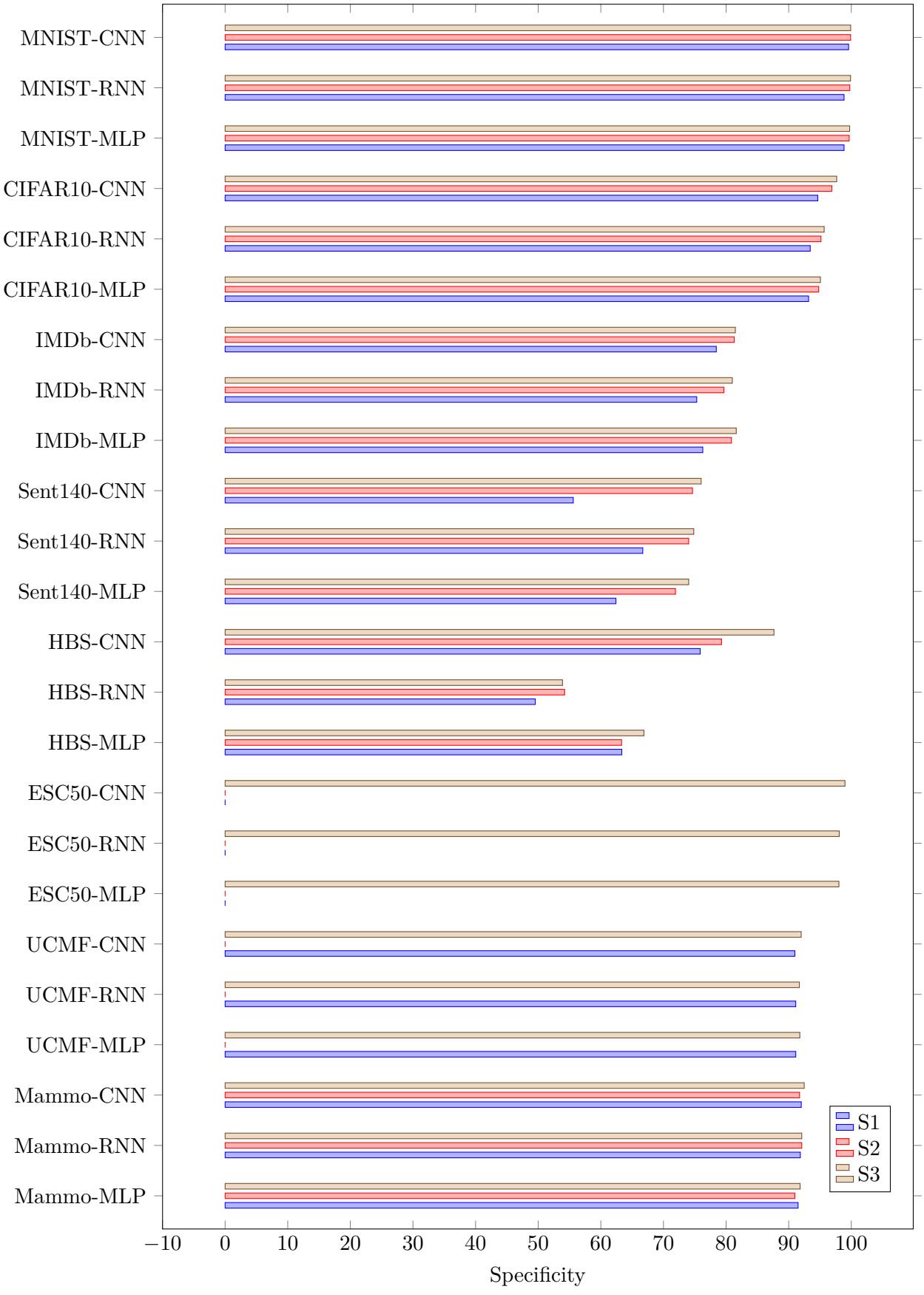

Figure 5.18: Specificity for all three artificial neural network types applied to all dataset and dataset sizes

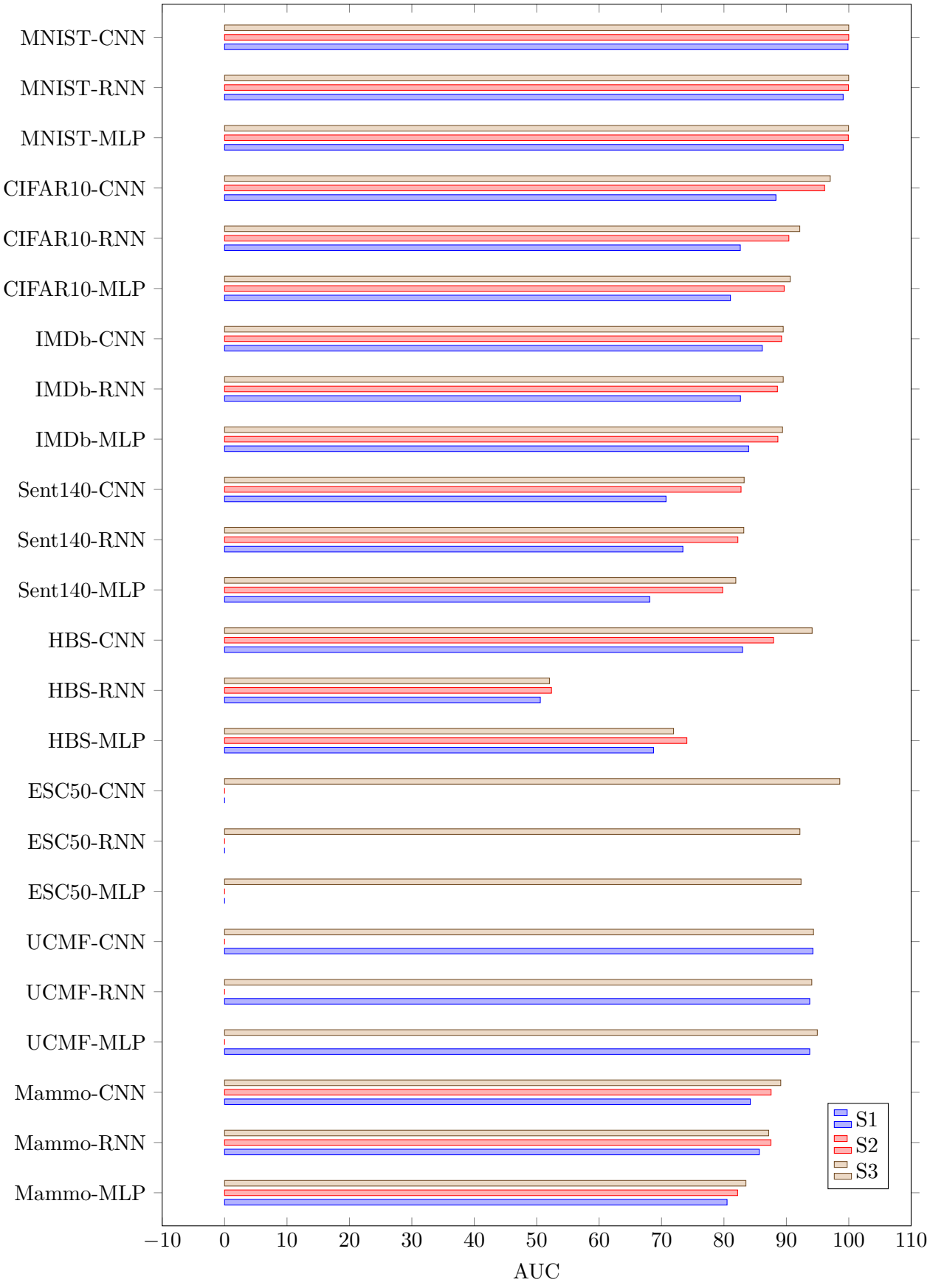

Figure 5.19: AUC for all three artificial neural network types applied to all dataset and dataset sizes

5.2 Results comparison

In this section the previous tables and plots are discussed, compared and analysed in order to mold the possible conclusions and findings.

The results were joined based on the comparative factors initially established. So we have the results joined by data type, neural network type, all studied evaluation metrics, accuracy, classes sensitivity and specificity as well as a median of all values, AUC and execution time values, all grouped by dataset size. Along the presented tables, some values are signaled to easier evaluation and analysis and respectively explained along the section.

For some explanation, in the same table, several sizes can be represented. They are identified by S1, S2 and S3, S1 being the smaller data sample size, and S3 being the largest. Tables regarding sensitivity and specificity values for each class of the dataset, have such information identified by SeX or SpX, where X is a specific class, *Se* is the sensitivity and *Sp* the specificity value.

The bar plots have both dataset of a specific type represented in the same figure. The models are represented by MODEL-X where MODEL can be CNN, RNN, MLP, SVM, RF or LR and X can be 1 or 2, referencing a specific dataset, identified in the legend of the figure.

The first set of Tables 5.1 to 5.10, present the corresponding evaluation metric values for all implemented models in all datasets, grouped by smaller and larger data samples. In this set of tables, we want to point out the models or algorithms that obtained the best results for each dataset classification.

The first two Tables, 5.1 and 5.2, present the accuracy values for all datasets. In the maximum tried data sample, presented in Table 5.1, it can be observed that for the image datasets, both datasets obtained better results in the ANN implementation. The MNIST dataset got the best accuracy values in the MLP network (99.70%) while the CIFAR10 dataset got the best value with the CNN (76.33%). On the other hand, it can be seen that for the same datasets, for the smallest data sample in Table 5.2, both datasets got the best accuracy with the CNN model (MNIST - 96.20%, CIFAR10 - 51.87%).

Compared to the other algorithms SVM, RF and LR in both tables, it can be seen that the difference in percentage is around 2% to 7% for the MNIST dataset and around 15% to 35% for the CIFAR10 dataset. This last dataset got higher difference between models, probably due to the shape of the data. The images were of higher dimension (32x32) and even the MLP model got slightly better results than the comparative models. Nevertheless, both datasets got better results with ANN in both tested sizes.

For the text datasets, both sizes got better results with LR algorithm for the IMDb dataset (88.97% and 84.02% respectively). The Sentiment140 dataset also performed better in the smaller data sample with LR (67.83%) but in the larger data sample, SVM obtained better results by 0.31% (76.71%), being fair to say that overall, LR got the best performance for this data type.

When it comes to ANNs, the RNN model performed slightly better in the Sentiment140 dataset for both sizes (74.84% and 66.74% respectively), where the type of wording was sloppier leading to a noisier data. The accuracy for the smaller data sample across all ANNs, showed that the adaptability of the RNN model to smaller data sizes was better than the other two models. In the IMDb dataset, all three architectures got around 81% accuracy in the larger data sample, slightly worse compared to the best 88.97%.

In both sound datasets and both tested sample sizes, CNN got the best results by far (88.87% and 77.90% respectively for the Heartbeat sound dataset and 76.78% for the ESC-50 dataset). RNN models did worse than all other models (62.02% and 50.49% respectively) and MLP got the second best performance of the remaining options (73.60% and 67.33% respectively for the Heartbeat sound dataset and 54.64% and 54.66% respectively for the ESC-50 dataset). All the comparative models had worse performance with results of approximately 70% and 65% for both sizes of the Heartbeat sound dataset and around 50% for the ESC-50 dataset.

Finally, the categorical datasets performed better with RF in both tested sizes (93.96% and 93.82% respectively for the UCMF dataset and 68.09% and 66.17% for the Mammo dataset). The results were not that different across the board, but in both sample sizes, RF performed better. The UCMF dataset was more consistent when it comes to accuracy, all models getting around 93% accuracy rates. The Mammo dataset was also consistent, where all models got more than 60% accuracy, but since it was an unbalanced dataset, some classes were not equally accurately classified.

The next two tables present the same results as the first two but regarding sensitivity results. These values are a mean of sensitivity values, for each class of the dataset with the sole purpose of evaluating if there is a discrepancy between accuracy and overall sensitivity. The sensitivity values for each class of each dataset are presented ahead in order to understand if the mean is equally distributed of if there is a discrepancy between classes.

When it comes to mean sensitivity, there is also a visible pattern between data types. Here, both image dataset got better sensitivity results with CNN in both data samples (99.15% and 96.09% respectively for the MNIST dataset and 76.32 and 51.88% respectively for the CIFAR10 dataset), unlike the accuracy results for the larger data sample, where MLP presented higher accuracy. All other models obtained worse sensitivity results, RNN being the closest ANN model to CNN with 0.5% difference in the larger data sample and 6% in the smaller for the MNIST dataset and 15% and 10% in the CIFAR10 dataset. Among the comparative models, SVM was the closest algorithm to the CNN results in both sizes, with a difference of less than 1% and 3% for the MNIST dataset and 23% and 11% for the CIFAR10 dataset.

For the text datasets, both SVM and LR shared the best values in the opposite sample sizes. Just like the accuracy results, the Sentiment140 dataset got the best sensitivity values with SVM (76.66%) while the IMDb dataset obtained the best results with LR (88.96%). In the smaller sample size, the opposite occurred (67.77% with the LR model for the Sentiment140 dataset and 84.12% with the SVM model for the IMDb dataset). Compared to the ANN models, for

the maximum data sample size, there is a difference of approximately 2% for the Sentiment140 dataset and 7% for the IMDb dataset. In the smaller data sample size, the difference between the LR algorithm and the RNN model is less than 0.1% for the Sentiment140 dataset and 6% for the IMDb dataset.

Both sound datasets also obtained the best sensitivity results with the CNN model (87.65% and 75.87% respectively for the Heartbeat sound dataset and 77.36% for the ESC-50 dataset) and the categorical datasets, just like the previous metric, got the best results with RF (92.30% and 92.11% respectively for the UCMF dataset and 52.21% and 43.73% for the Mammo dataset). For the maximum data size, the Mammo dataset got around 63% accuracy for all implemented models, but when it comes to sensitivity the values drop drastically, mostly due to the unrepresented classes in the data. Still, RF got 20% to 25% higher results than all other models and algorithms.

The same patterns are equally visible in the specificity and AUC Table 5.5 to 5.8. In the AUC tables there are a few modifications to what has been analysed thus far. For the higher sample size, the UCMF dataset presented slightly higher AUC values for the MLP network (94.96%), 0.7% better than the previously presented RF model (94.89%) and in the smaller data sample, both UCMF and Mammo datasets presented slightly higher AUC results in the CNN (94.25%) and RNN (85.66%) models respectively.

The final two tables are useful to paint a generic perspective of the computation time that takes for each data type and each method. The values are an approximation of the final time, so similar times like 49.12 and 48.99 can be rounded to 49, since tens of a second are not impactful to an experiment.

When it comes to ANN models, MLP presents faster computation across all implemented datasets, mostly due to the simplicity of the models. The CNN models, the most robust when it comes to layer composition, managed to be faster than RNN in most datasets, with the exception of the MNIST and Heartbeat sound datasets. For the smaller data samples, the RNN model had overall, worst results or similar to the CNN models.

The RF and LR algorithms performed significantly better than all other methods, but for the larger data samples, LR had the fastest times. If we generalize the results to even larger datasets, it is fair to say that ANNs are slower than these two algorithms.

Now we enter the section of tables where the sensitivity and specificity of each class, of each dataset is presented. This is meant to see which models are able to better classify which class and which model is able to better classify most classes. Only the maximum and minimum data sample size values were signaled since they were the ones compared in previous tables and since the medium sample size was, as previously mentioned, used to demonstrate the data is unbiased, meaning that the performance in the maximum range of the dataset was not significantly different when a subset of the total data was selected. The signaled values for the medium size represent cases where that data sample obtained better results than the other two samples.

The binary datasets present both sensitivity and specificity results in the same table, since there is a symmetry between values, where the sensitivity of one class represents the specificity of the other and vice-versa.

Table 5.11 presents the sensitivity results for the MNIST classes. It can be immediately seen that the CNN and RNN models are the ones that better classify most classes. The RNN model has the best sensitivity results in 6 classes, two with the same results as the CNN model, which obtained the best values in 5 classes. Compared to the other models, there is a slight percentage difference between them, of around 2%.
For the smaller data sample, CNN obtained the best results across all classes, with a difference between the other values ranging from approximately 1% to 9%.

As for specificity, presented in Table 5.12, the results follow the same pattern as the sensitivity values. CNN and RNN present the best results, although the medium dataset presented slightly better results for some classes.
When it comes to the smaller data sample, CNN presented the best specificity results for 7 of the 10 classes and the RNN model in 3 classes.

Tables 5.13 and 5.14 present the sensitivity and specificity for the CIFAR10 dataset classes. There is a clear distinction between the CNN model's results from every other model and algorithm. The CIFAR10 data complexity made it harder for some models to accurately classify some classes, most of them presenting worse results than the CNN model, varying from 10% to 30% lower sensitivity values across all classes. Although for smaller data samples, models like RNN and MLP obtained slightly better results of approximately 1% to 6%.

The specificity values follow the same result distribution, the CNN model obtaining better results in most classes and in the smaller data sample, MLP and RNN models presenting better specificity results in two classes (3% and 6% difference for the MLP model and 1% difference for both classes with the RNN model).

Table 5.15 presents the sensitivity and specificity for the binary classification of the IMDb dataset. Both SVM and LR models presented the best results in both evaluation metrics, although for the maximum data sample size, the LR model obtained the best results across both classes and both metrics (88.27% and 89.65% sensitivity for both classes, 89.65% and 88.27% specificity for both classes), with less than 1% more.

That changed in the smaller data sample size, where the SVM model presented higher sensitivity in class 1 and higher specificity for class 0 (84.60%). This goes hand-in-hand with the accuracy results previously discussed.

For the Sentiment140 dataset, the results are not that consistent with the previous results. This dataset presented higher accuracy, median sensitivity and specificity for the SVM and LR models. But the class sensitivity and specificity did not stay consistent with those results.

For the maximum size data sample, the SVM algorithm got a better sensitivity for class 1 and specificity for class 0 (79.70%). The class 0 sensitivity and class 1 specificity got better results with the RNN model (78.66%), 5% higher than the SVM model.

In the smaller data sample, SVM and LR shared similar results (0.18% difference) in the class 0 sensitivity and class 1 specificity (63.18% with the LR algorithm), but the CNN model got 17% higher values for the class 1 sensitivity and class 0 specificity (89.06%).

It can be observed that for smaller data samples, ANN's sensitivity and specificity between both classes become more disperse (approximately from 6% to 15% 18%) while the SVM and LR algorithms stay more consistent (approximately from 6% to 9%). That can justify why the CNN model presented high sensitivity and specificity for those classes, by focusing the learning of specific cases and neglecting the other class.

Entering in the sound datasets Tables, 5.17 to 5.19, the Heartbeat sound dataset was consistent with the results thus far. The CNN model got the best sensitivity and specificity results for both classes in both tested sizes.

Apart from the CNN model, the most consistent comparison is the RF algorithm. Even though the results may be worse for a particular class, the difference of results between classes is minor for this algorithm. For example, the LR algorithm, for the maximum tested sample size, presented 79.54% sensitivity for class 0 while the RF algorithm presented 70.95%. But while the RF algorithm had 72.22% sensitivity for the other class, LR obtained 55.26% sensitivity.

For the ESC-50 dataset, the CNN model had the best sensitivity results for 70% of the classes, followed by MLP which had 16%. The CNN model obtained a 100% sensitivity result in two classes and more than 50% of the classes presented results above 80% sensitivity, being the best model.
The remaining 30% of classes where the CNN model did not present the best results, differ from 10% to 25% lower sensitivity values.

The specificity results were also better in the CNN model, although the MLP model presented best results in 25% of classes. The CNN model had five 100% specificity results for five distinct classes and approximately 71% of classes with more than 99% specificity.

The final three Tables 5.20, 5.21 and 5.22, present the results for the categorical datasets. It has been presented that for the UCMF dataset, the RF model obtained better results. Here we can see that in fact, all models have similar sensitivity and specificity results. For the both

data sample sizes, the sensitivity and specificity values of class 0 are all around 85% and 98% respectively and the opposite for the other class. So we can see that all models did a similar job classifying the classes. But for the maximum data sample size, the MLP and LR obtained the same results for sensitivity of class 1 and specificity of class 0 (98.37%). RF got the best sensitivity for class 0 and specificity for class 1 (86.34%).

The minimum data sample size, was more consistent with what has been witnessed thus far, with SVM obtaining the best result for class 1 sensitivity and class specificity (98.47%) and RF for class 0 sensitivity and class 1 specificity (85.96%).

The Mammo dataset also maintained consistency, with the RF model presenting the best sensitivity results in 75% of classes in the maximum data sample and 62.5% in the minimum. Here it can be seen that with the same data, the models obtained very different results. While other models like all the ANNs and especially the SVM model, focused learning on the two most represented classes, the RF model generalized the exact same data and obtained sensitivity results of more than 70% where the other methods reached 30% or even 3% in some cases.
In the minimum data sample size, the underrepresented classes got lower sensitivity results, the decrease ranging from 20% to 30%.

The specificity results showed to be higher in the SVM model with some classes obtaining 100% evaluation in both tested sizes. Unlike the previous metric, here the results vary around 5% between models.

The final set of results are presented in the format of plot bars so the differences between the tested sizes can be easier to understand and to have a more visual interpretation of the difference between values.

The first 15 plots are divided by data type and evaluation metric. Each plot represents an evaluation metric values for the two datasets of each data type. Each bar in each method, represents one tested sample size.
The Figure 5.1 to 5.4 presents the accuracy, sensitivity, specificity and AUC values for the image datasets, MNIST and CIFAR10.

It can be seen that all metrics decrease across all methods when the data sample size becomes smaller. The discrepancy between accuracy, sensitivity and AUC results is wider for the CIFAR10 implemented ANNs. The RF and LR models presented similar results despite the size of the sample.

The MNIST dataset presented similar accuracy, sensitivity, specificity and AUC results for all methods and all sizes.
For the text datasets, both the IMDb and Sentiment140 datasets presented the same pattern for all implemented models and sizes. All metrics results decrease results when the dataset becomes smaller.

The IMDb dataset presented some stability where the result difference between the smaller and larger data samples is pretty much the same, ranging from 3% to 6%. The Sentiment140 dataset presents a broader range of results between sizes, specially the CNN model with approximately 15% accuracy, sensitivity and specificity difference, and 20% for AUC.

The difference in the sound datasets is not the same as in the previous cases. The ESC-50 dataset is represented by only one testing size so there can not be any direct conclusions regarding the difference in data size.

The Heartbeat sound dataset presents some differences from what has been witnessed so far. Some models present values for some metrics where the medium and the smaller sample size achieve better results than other sizes. For accuracy, the RF model presented better results for the medium data size, followed by the entire dataset and finally, the smaller sample.

The MLP, SVM and LR models all present better accuracy for the smaller data sample than the medium data sample. The exact same patterns occur for sensitivity, specificity and AUC except that the medium data sample also got better sensitivity results with the RNN model. The ESC-50 dataset presents similar specificity and AUC results for all implemented models, when the sensitivity and accuracy results separate CNN from all the other models.

Finally, the categorical datasets, represented in Figure 5.13 to 5.15 show the difference between somewhat balanced data and unbalanced data. The UCMF dataset presents generally similar results across all models and the two testes sizes. Although the LR model obtained better results, all models had a similar performance in a data set that has one under represented class

but still sufficient to correctly classify both classes.

The Mammo dataset on the other hand shows clear inconsistencies among all metrics. The accuracy results for the data set followed the usual scenario where the smaller the dataset, lower accuracy values were found, ruling out the RNN that presented higher accuracy for the medium data sample size, MLP and SVM that had slightly better results in the smaller set compared with the medium set. Apart from that, all models present similar results with 2% to 5% discrepancy.

When the sensitivity results are analysed, a different scenario can be observed. All models apart from the RF model present low sensitivity in general, leading to the conclusion that those models correctly classified some classes and poorly classified others. An example can be made of the SVM model that had 0.00% sensitivity on 5 out of 8 classes.

The final set of plot figures shows the comparison between all ANN types in the different implemented sizes and different evaluation metrics. This comparison relates to the influence of architectural type with the other conditions.
Generally, all three types have similar performance except in some particular situations, specially regarding data size.

For the accuracy results presented in Figure 5.16, in the image datasets, all three models obtained similar results, apart from the CIFAR10 dataset where the CNN model was able to better deal with the more complex data. For the smaller data sample, the CNN model performed significantly better than the other networks in both datasets.

For the IMDb and Sentiment140 datasets, the three models performed similarly although the RNN and MLP networks outperformed the CNN model. For the Sentiment140 dataset specifically, the RNN model dealt better with the noise in the data, since the dataset was predominantly composed of everyday tweets, which contain every type of word and spelling.
The smaller data samples obtained different results. The IMDb dataset CNN implementation had slightly better accuracy results for this sample size, while the Sentiment140 RNN model outperformed both CNN and MLP with a 5% to 10% difference.

As clearly seen in the other tables, the CNN model performed far better than the other network types in both sound datasets. Although it can be seen in the Heartbeat sound dataset that the MLP model, for smaller samples, was better than the RNN model and approached the CNN accuracy result from a difference of 15% to 10%.

Both categorical datasets had similar results across all three networks and sizes. Among the three network types, the CNN model obtained slightly better accuracy in both datasets.
Figure 5.17 and Figure 5.18 present the same comparison between network types for the sensitivity and specificity results.

The sensitivity results are almost the same analysed in the previous accuracy plot. It can be seen that the sensitivity difference between the RNN and MLP models for the Sentiment140 dataset is wider than the accuracy results between them. It can also be seen that the sensitivity for the Heartbeat sound RNN and MLP models decreased while the CNN model maintained similar values with the accuracy results, equally classifying both classes.

We can also see for the Mammo dataset, that among three networks, the CNN network had the best sensitivity result, even though the three networks showed low receptivity to the unbalanced data.

The specificity plot showed mostly consistency between networks. The MNIST, IMDb, ESC-50, UCMF and Mammo datasets presented similar results for the three implemented models. The CIFAR10 and Heartbeat sound datasets showed that the CNN model obtained higher specificity results and the Sentiment140 higher specificity result was with the RNN model.
The MNIST and both categorical datasets also show the proximity in specificity results from all three models in all tested sizes.

Finally, for the AUC results, the MNIST, IMDb and UCMF datasets presented similar results for the three ANN types, with all the MNIST and UCMF models obtaining similar AUC results among the respectively tested sizes.

The CIFAR10, Heartbeat sound, ESC-50 and Mammo datasets obtained better results with the CNN model in all tested sizes and the Sentiment140 dataset presented better AUC results with the RNN model just like the previous described metrics.

5.3 Training accuracy, loss and final ROC curves

In this section, the learning process of the models and the ROC curves are presented and briefly discussed, specially cases that differ from what was expected.

In the Appendix A, there are two examples of cases. One of what was mostly obtained and the other of special and different scenarios and results. Figures A.1 to A.4 represent the results of the CNN model applied to the CIFAR10 dataset. It represents the most usual results of all implemented models in all presented datasets. The learning curve is ascending along the number of epochs and heads towards a final result. The validation curve accompanies the training curve staying within a proximal range. The same can be said about the descending loss curve.

The ROC curves for each class present the usual curve far away from the diagonal average result and the confusion matrix presents more light colors in the diagonal

$$x = -y$$

, representing good classification results for each class. The darker the cell in the presented diagonal, lower the classification accuracy.

But there were cases that did not follow the previous patterns. The ESC-50 CNN, RNN and MLP models and the Mammo CNN model, presented a more spiked accuracy and loss curves and the Heartbeat sound dataset MLP model presented an even more accentuated spiking within epochs. The other models, CNN and RNN presented curves similar to the ESC-50 CNN curves, present in Figure A.5 and A.6.

When it comes to ROC curves, the ESC-50 ANN models presented in Figure A.7, show a more segmented curve rather than the curve presented in Figure A.3. The curves have a drastic upward raise and then have linear moments to the right and upwards again.

The ROC curves for the Heartbeat sound MLP and RNN models, in Figure A.10 and A.11, present a proximity to the plot diagonal, the RNN model being the most proximal to the average diagonal.

The Mammo dataset ANN models, all present the same pattern, showed in Figure A.14. Both curves, representing both classes, follow the same path that the macro-average ROC curve follows. The micro-average curve on the other hand, show more concavity away from the average diagonal.

While the macro-average curve computes the metric for each class independently and only then takes the average, the micro-average curve aggregates all classes' contributions and computes the metric. This shows that the ROC curves for each classes follow the independent class computation.

Chapter 6

Conclusions and Future Work

6.1 Research summary

This study main objective was to compare ANNs classification performance under different situations and conditions, comparing to other algorithms in order to understand the most favorable applicability of ANN.

Three types of neural network, convolutional, recurrent and multilayer perceptron models, were implemented on 8 different datasets and compared with three algorithms, logistic regression, support vector machines and random forest. The datasets belonged to different data types, image, sound, text and categorical data, two for each type. Those datasets were pre-processed and partitioned into smaller sizes ranging from one to three partitions and those subsets were created to study the performance of the models when fed different data sizes.

Finally, several evaluation metrics, accuracy, sensitivity, specificity and AUC values were studied in order to have a more conclusive understanding of the results.

The conclusions this study wants to find is the circumstances where the performance of ANN is more suited and where it is not. A specific dataset type or size, or a specific network.

6.2 Main findings

The main conclusions that can be extracted from the results, pass through each evaluation metric, comparative parameter and implemented model.

By data type, it can be seen that image datasets obtained better results with ANNs. Both studied datasets and all ANN models, CNN, RNN and MLP presented better accuracy, sensitivity and specificity than the comparative algorithms.
Between ANNs it can be seen that CNN obtained overall better results than the other two networks.

For the text datasets, ANNs had slightly worse performance than some of the comparative models. SVM and LR presented better results across all metrics, with LR being generally better for both sizes.

Sound datasets presented better results with the CNN model across all metrics and categorical datasets presented slightly better results with RF than all ANN models.

With this, we can conclude that image and sound datasets showed to be better when applying neural networks and sequential data like text data, which is known to have a good applicability with RNN had better classification performance with other algorithms.

As far as dataset size, the Sentiment140 RNN model presented, for the smaller data sample size, similar results as the best found models, LR and SVM,but worse for larger amounts of data. For categorical data, RF presented much more learning capability to unbalanced data, far more than any other model or algorithm.

Between ANN network types, the CNN model had the best performance in the CIFAR10 dataset for all sizes and in the MNIST dataset for the smaller data sample. The results for the larger data sample were similar across all three networks.

In the text data type, for the maximum data sample size, all models had similar outcomes. The CNN outperformed the other models in the IMDb smaller data sample, while the RNN model had better performance in the Sentiment140 smaller data sample, a dataset that presents more extensive vocabulary and a bigger generalization of words.

The CNN model presented better results for both sound datasets and in the categorical datasets, all three models had similar results, although the CNN had better sensitivity results for both datasets.

6.3 Improvement and future work

This work accomplished most of what it was established from the initial methodology. Although, some aspects could be improved in order to also improve possible conclusions.

An interesting addition to the study would be the introduction of distinct data shapes and how their possible transformation could impact the performance of a model. We used multi-dimensional data with the images dataset but it could be an interesting feature to study other n-dimension types of data.

Another improvement could be the study of extremely large or small datasets to study the impact of big and small data. We tried to touch this subject by dividing the original datasets in smaller and smaller sizes but the study with really small sets of data could be interesting. The study of big data involves time and resources that were not an option for this study.

Other possible improvements that were not possible due to the time needed to execute them

could be a more focused and analytic pre-processing of the data, to ensure the maximum efficiency of the models and a more exhaustive construction of the networks with the use of tools that could help the efficiency and robustness of the networks.

Appendix A

Result Appendix

A.1 Learning and loss curves, confusion matrix and ROC curves

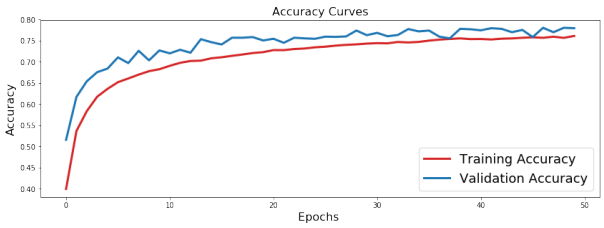

Figure A.1: CIFAR10 CNN model accuracy curve - Usual learning result

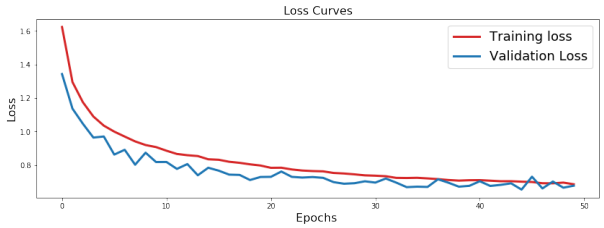

Figure A.2: CIFAR10 CNN model loss curve - Usual learning result

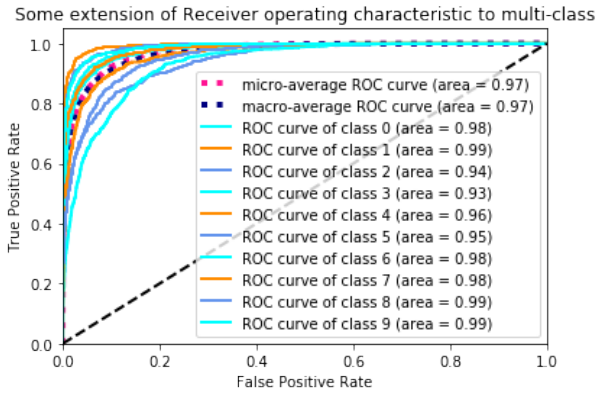

Figure A.3: CIFAR10 CNN model ROC curve - Usual learning result

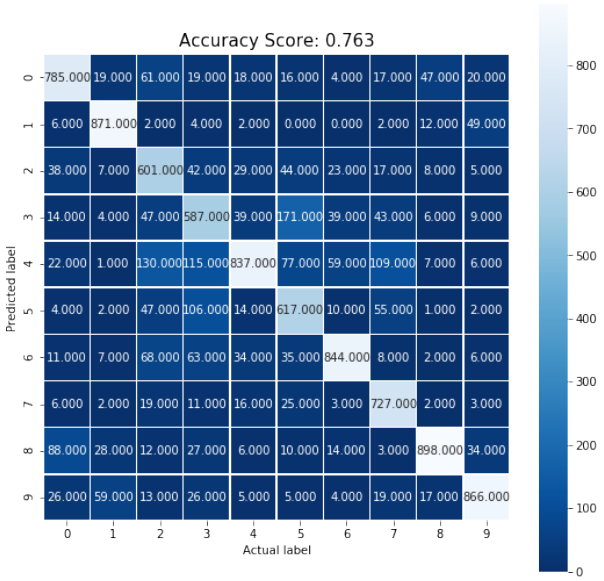

Figure A.4: CIFAR10 CNN model confusion matrix - Usual learning result

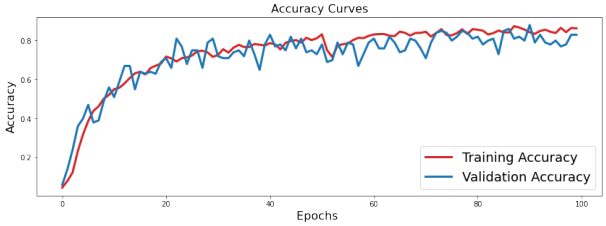

Figure A.5: ESC-50 CNN model accuracy curve - Unusual learning result

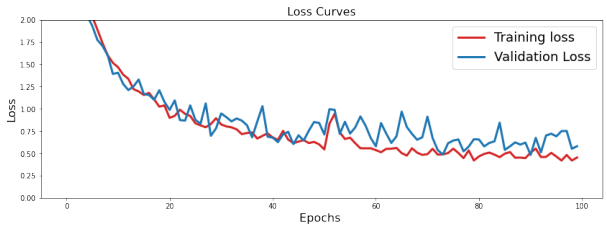

Figure A.6: ESC-50 CNN model loss curve - Unusual learning result

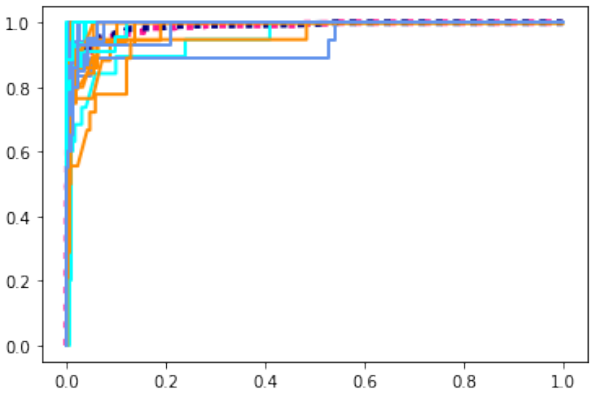

Figure A.7: ESC-50 CNN model ROC curve - Unusual learning result

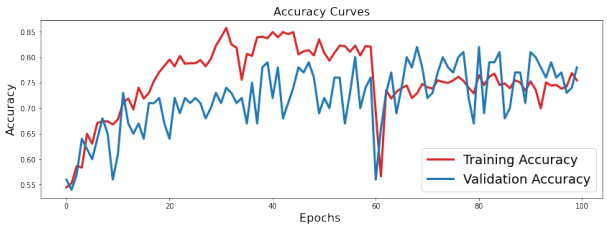

Figure A.8: Heartbeat sound MLP model accuracy curve - Unusual learning result

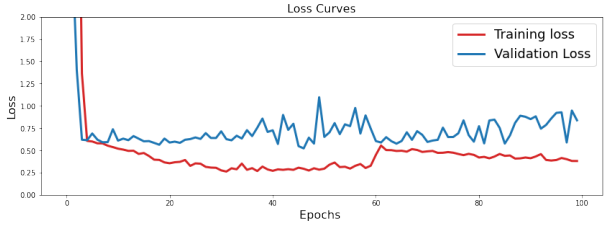

Figure A.9: Heartbeat sound MLP model loss curve - Unusual learning result

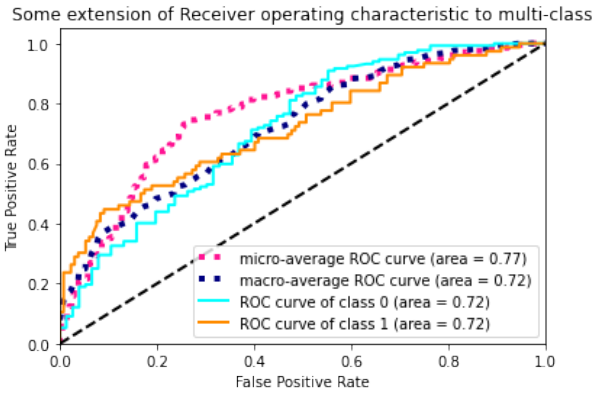

Figure A.10: Heartbeat sound MLP model ROC curve - Unusual learning result

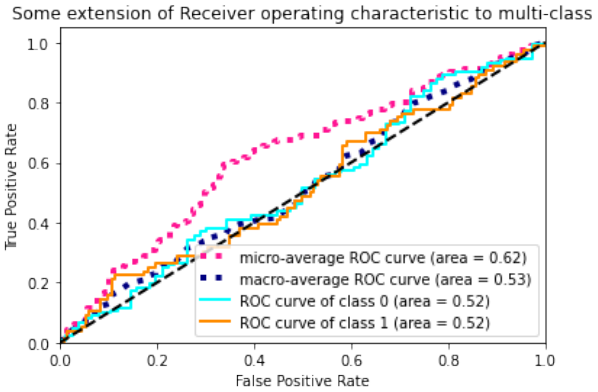

Figure A.11: Heartbeat sound RNN model ROC curve - Unusual learning result

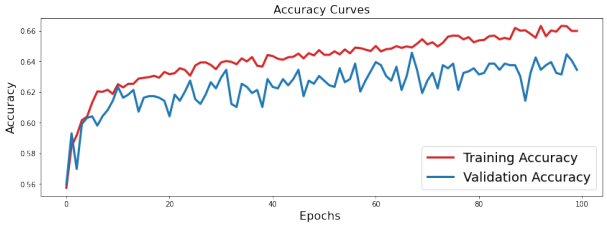

Figure A.12: Mammo CNN model accuracy curve - Unusual learning result

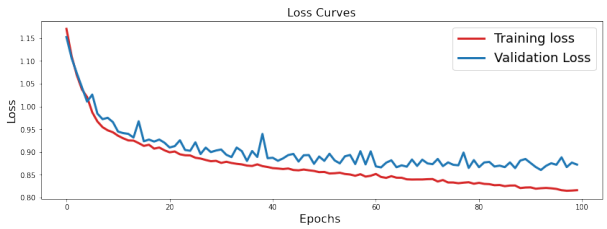

Figure A.13: Mammo CNN model loss curve - Unusual learning result

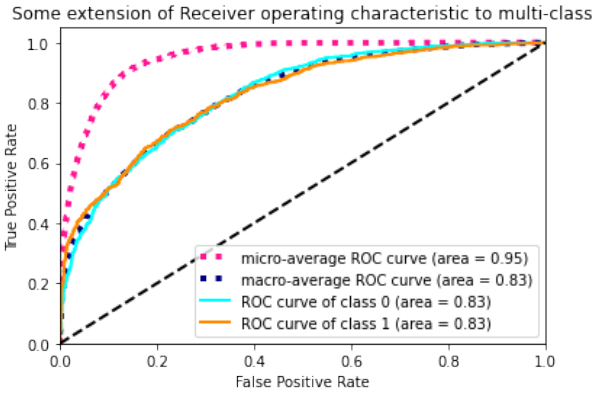

Figure A.14: Mammo CNN model ROC curve - Unusual learning result

Bibliography

[1] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7:158820–158846, 2019.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018. ISSN: 2405-8440. doi:https://doi.org/10.1016/j.heliyon.2018.e00938.

[3] Deme C. Abraham. Development and comparison of artificial neural network techniques for mobile network field strength prediction across the josplateau, nigeria. *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, 3(6), 2016. ISSN: 2458-9403. doi:10.1002/for.2542.

[4] Hamid Behbahani, Amir Mohamadian Amiri, Reza Imaninasab, and Meysam Alizamir. Forecasting accident frequency of an urban road network: A comparison of four artificial neural network techniques. *Journal of Forecasting*, 37(7):767–780, 2018. doi:10.1002/for.2542.

[5] William A. Borders, Shunsuke Fukami, and Hideo Ohno. Characterization of spin–orbit torque-controlled synapse device for artificial neural network applications. *Japanese Journal of Applied Physics*, 57(10):1002B2, sep 2018. doi:10.7567/jjap.57.1002b2.

[6] Q. Chen and K. Folly. Effect of input features on the performance of the ann-based wind power forecasting. pages 673–678, 2019.

[7] Dursun Delen, Glenn Walker, and Amit Kadam. Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial Intelligence in Medicine*, 34(2):113 – 127, 2005. ISSN: 0933-3657. doi:https://doi.org/10.1016/j.artmed.2004.07.002.

[8] Adel El-Shahat. Introductory chapter: Artificial neural networks. In Adel El-Shahat, editor, *Advanced Applications for Artificial Neural Networks*, chapter 1. IntechOpen, Rijeka, 2018. doi:10.5772/intechopen.73530.

[9] F. Sunar Erbek, C. Özkan, and M. Taberner. Comparison of maximum likelihood classification method with supervised artificial neural network algorithms for

land use activities. *International Journal of Remote Sensing*, 25(9):1733–1748, 2004. doi:10.1080/0143116031000150077.

[10] Milad Fatehnia and Gholamreza Amirinia. A review of genetic programming and artificial neural network applications in pile foundations. *International Journal of Geo-Engineering*, 9(2), 2018. doi:10.1186/s40703-017-0067-6.

[11] Mahmut Firat, Mustafa Erkan Turan, and Mehmet Ali Yurdusev. Comparative analysis of neural network techniques for predicting water consumption time series. *Journal of Hydrology*, 384(1):46 – 51, 2010. ISSN: 0022-1694. doi:https://doi.org/10.1016/j.jhydrol.2010.01.005.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] Enzo Grossi and Massimo Buscema. Introduction to artificial neural networks. *European journal of gastroenterology hepatology*, 19, 2008.

[14] Yong Soo Kim. Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size. *Expert Systems with Applications*, 34(2):1227 – 1234, 2008. ISSN: 0957-4174. doi:https://doi.org/10.1016/j.eswa.2006.12.017.

[15] Chen-Chiang Lin, Yang-Kun Ou, Shyh-Huei Chen, Yung-Ching Liu, and Jinn Lin. Comparison of artificial neural network and logistic regression models for predicting mortality in elderly patients with hip fracture. *Injury*, 41(8):869 – 873, 2010. ISSN: 0020-1383. doi:https://doi.org/10.1016/j.injury.2010.04.023.

[16] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[17] Ina S. Markham and Terry R. Rakes. The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression. *Computers Operations Research*, 25(4):251 – 263, 1998. ISSN: 0305-0548. doi:https://doi.org/10.1016/S0305-0548(97)00074-9.

[18] João Maroco, Dina Silva, Ana Pina Rodrigues, Manuela Guerreiro, Isabel Santana, and Alexandre Mendonça. Data mining methods in the prediction of dementia: A real-data comparison of the accuracy, sensitivity and specificity of linear discriminant analysis, logistic regression, neural networks, support vector machines, classification trees and random forests. *BMC research notes*, 4:299, 08 2011. doi:10.1186/1756-0500-4-299.

[19] Robert Milewski, Anna Justyna Milewska, Teresa Więsak, and Allen Morgan. Comparison of artificial neural networks and logistic regression analysis in pregnancy prediction using

the in vitro fertilization treatment. *Studies in Logic, Grammar and Rhetoric*, 35(1):39 – 48, 2013.

[20] MOGHADDAM2020104421 Davoud Davoudi Moghaddam, Omid Rahmati, Mahdi Panahi, John Tiefenbacher, Hamid Darabi, Ali Haghizadeh, Ali Torabi Haghighi, Omid Asadi Nalivan, and Dieu [Tien Bui]. The effect of sample size on different machine learning models for groundwater potential mapping in mountain bedrock aquifers. *CATENA*, 187:104421, 2020. ISSN: 0341-8162. doi:https://doi.org/10.1016/j.catena.2019.104421.

[21] Mahdi Vafakhah Biswajeet Pradhan Mohammad Zare, Hamid Reza Pourghasemi. Landslide susceptibility mapping at vaz watershed (iran) using an artificial neural network model: a comparison between multilayer perceptron (mlp) and radial basic function (rbf) algorithms. *Arabian Journal of Geosciences*, 6:2873–2888, 2013. ISSN: 1866-7538. doi:10.1007/s12517-012-0610-x.

[22] Kisi Ozgur. River flow forecasting and estimation using different artificial neural network techniques. *Hydrology Research*, 39(1):27–40, 02 2008. ISSN: 0029-1277. doi:10.2166/nh.2008.026.

[23] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, 208, 07 1998.

[24] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G. Yang. Deep learning for health informatics. *IEEE Journal of Biomedical and Health Informatics*, 21(1): 4–21, 2017.

[25] Allen H. Renear, Simone Sacchi, and Karen M. Wickett. Definitions of dataset in the scientific and technical literature. *Proceedings of the American Society for Information Science and Technology*, 47(1):1–4, 2010. doi:10.1002/meet.14504701240.

[26] Yeh Fang-Cheng Rhett N. D'souza, Po-Yao Huang. Structural analysis and optimization of convolutional neural networks with a small sample size. *Scientific Reports*, 10, 2020. ISSN: 2045-2322. doi:10.1038/s41598-020-57866-2.

[27] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 1995.

[28] Madan K. Jha Sasmita Sahoo. Groundwater-level prediction using multiple linear regression and artificial neural network techniques: a comparative assessment. *Hydrogeology Journal*, 21, 2013. ISSN: 1435-0157. doi:10.1007/s10040-013-1029-5.

[29] Brian W. Szuster, Qi Chen, and Michael Borger. A comparison of classification techniques to support land cover and land use analysis in tropical coastal zones. *Applied Geography*, 31 (2):525 – 532, 2011. ISSN: 0143-6228. doi:https://doi.org/10.1016/j.apgeog.2010.11.007.

[30] Wen Zhu, Nancy Zeng, and Ning Wang. Sensitivity, specificity, accuracy, associated confidence interval and roc analysis with practical sas implementations. *Health Care and Life Sciences*, 2010.