Article

Image classification using deep and classical machine learning on small datasets: a complete comparative

Gonzalo Miranda [1,*] and Clemente Rubio-Manzano [1,2]

[1] Department of Information Systems, University of the Bío-Bío, Chile
[2] University of Cadiz; clrubio@ubiobio.cl
[*] Correspondence: gonzalo.miranda1501@alumnos.ubiobio.cl

Abstract: One of the most important challenges in the Machine and Deep Learning areas today is to build good models using small datasets, because sometimes it is not possible to have large ones. Several techniques have been proposed in the literature to address this challenge. This paper aims at studying the different available Deep Learning techniques and performing a thorough experimentation to analyze which technique or combination thereof improves the performance and effectiveness of the models. A complete comparison with classical Machine Learning techniques was carried out, to contrast the results obtained using both techniques when working with small datasets. Thirteen algorithms were implemented and trained using three different small datasets (MNIST, Fashion MNIST, and CIFAR-10). Each experiment was evaluated using a well-established set of metrics (Accuracy, Precision, Recall, F1, and the Matthews correlation coefficient). The experimentation allowed concluding that it is possible to find a technique or combination of them to mitigate a lack of data, but this depends on the nature of the dataset, the amount of data, and the metrics used to evaluate them.

Keywords: Artificial Intelligence, Deep Learning, Image Classification, Machine Learning, Predictive Models, Small Datasets, Supervised Learning.

1. Introduction

Artificial intelligence (AI) today, is one of the most important and promising fields in many different research areas. For example, in a survey of more than 630 companies in 13 countries, it was reported that more than 30% of companies have allocated USD$50 million in initiatives related to AI, such as robotics, automation, and Machine Learning [1].

Historically, AI has been divided into two great paradigms: symbolic, and connectionist. Symbolic AI deals with the representation of knowledge from a higher "symbolic" level (mathematical logic, problem solving, knowledge representation and reasoning, and so on). Connectionist AI systems have the ability to acquire their own knowledge by extracting patterns from data, which is also known as Machine Learning (ML) [2].

ML has been used in many different tasks, such as classifying images into categories, voice-to-text transcription, matching products to customer interests, etc. In recent years, the availability of high-capacity hardware resources [3] and the generation of large amounts of data [4,5] has led to a large boost in the development of this discipline, and has promoted the development and growth of Deep Learning (DL).

Despite the good results obtained, there are some limitations, challenges, or open problems to solve:

1. **Labeling a large amount of data.** One of the key requirements for data to be used to generate good Machine Learning models is that they must be correctly labeled. This requirement is one of the disadvantages of current supervised Deep Learning [2] as the labeling process can be a costly and complex process that can take a long time and varies depending on the magnitude of the data being labeled [6].

2. **Scarcity or difficulty of data collection.** There are some fields that fall outside this trend of massive generation of data, among them, the field of medicine [3,7] where data are scarce, since they are linked to the number of patients treated, which limits the amount of data available. Other fields outside this massive generation of data are industry [8,9] and materials science [10] where there is a difficulty in collecting data due to the processes related to obtaining it [5,11].

For these reasons, one of the most important challenges in ML is obtaining good models using small datasets. Different techniques have been proposed in the literature to address this challenge. In this paper, we selected and studied some of these techniques and performed a thorough experimentation to analyze which technique or combination thereof improves the performance and effectiveness of Deep Learning models. It is important to clarify that the priority in the selection process is to choose models that are more suitable to work with small data, hence some models (as ResNet [12] or LeNet [13]) are not considered in this study, since they were designed for big datasets.

Additionally, we implemented three classical (Decision Tree, Random Forest, and Vector Support Machine) and nine Deep Neural Network algorithms. These are trained on four subsets of different sizes (10, 50, 250, and 500 images per class) of the MNIST, Fashion MNIST and CIFAR-10 datasets, which randomly choose from the training set of each dataset. We carried out a total of 156 experiments and a complete comparison was made to contrast the results obtained using both techniques. Thirty graphs and tables have also been created and were used to support our discussion. We first evaluate each technique individually and then we study their combination to study if the combinations allows us to improve the results obtained by each technique individually. A second combination is made with transfer learning and data augmentation because the works from [8,14,15] add both techniques at the same time, so we study if it make improvements on top of the combined techniques.

Summarizing, the most relevant contributions of this paper are:

- A literature review to find Deep Learning techniques that can be used for image classification with small datasets, making a selection based on the adequacy of these techniques for small datasets.
- The implementation and comparison of the selected Deep Learning techniques and classic Machine Learning algorithms, by using five metrics about the performance of each algorithm when this works with small datasets.
- The ratification that Dropout, Global average pooling, Transfer learning, Data augmentation, Cyclic learning rate decay, and the Cosine similarity loss function are techniques that can face the problem of working with small datasets.
- The Deep Learning community receives interesting results and evidence about the possibility that a combination of techniques can improve the results obtained with Deep Learning algorithms without techniques and/or using classical algorithms.

The rest of the paper is organized as follows: Section 2 introduces preliminary notions about Machine/Deep Learning and the techniques that will be implemented, and analyzed; Section 3 explains the experimentation carried out; in Section 4, the results obtained in the experimentation are textually and graphically explained using quantitative tables and figures; in Section 5 a discussion of the results is presented, and explained; finally, in Section 6 conclusions and the future work are presented.

2. Background: Machine Learning and Deep learning techniques on small datasets

2.1. Types of machine learning

Machine Learning algorithms can be categorized into three types: supervised learning, unsupervised learning, and reinforcement learning. **Supervised learning** implies that the algorithm learns to automatically identify categories from labeled examples by building a model that is able to identify new examples, that is to say that, from a new given example, this is able to identify its corresponding category (label). On the other hand, **unsupervised learning** implies that the algorithm receives the data and automatically obtains the relationship between them, that is, it is able to understand the internal structure of the data and the categories are automatically identified. Also, there is a semi-supervised type of learning that tries to combine both supervised learning and unsupervised learning approaches, where a model trained on a small amount of labeled data is used to artificially label a bigger dataset of unlabeled data. Finally, **reinforcement learning** is characterized by having an interaction between a learner and an environment, where the latter is responsible for providing feedback to the former. This paper focuses on supervised Machine/Deep Learning algorithms, and the contributions whose objective is to improve current models to work with small datasets.

Several techniques can be found in the literature, whose purpose is to enhance classical Deep Learning techniques to work with small datasets: In Table 1, some of the most relevant Deep Learning techniques studied in the literature are shown. These techniques can be grouped into four categories depending on the elements updated in the deep learning process: layer of the network, loss function, learning rate scheduler, before training technique, and after training technique.

Table 1: Summary of the Deep Learning techniques.

Type	**Techniques**
Layer of the network	Dropout [5,14,16,17], Batch normalization [18,19], Global average pooling [18], Dilated convolution [18]
Loss function	Cosine similarity [11]
Learning rate scheduler	Cyclic learning rate decay [11]
Before training technique	Data augmentation [8,14–17,20], Transfer learning [8,14,15,19,21]
After training technique	Multi-model ensemble [8,21]

In the rest of that section, each technique studied and implemented is explained in detail.

2.2. Layer of the network

2.2.1. Regularization

The concept of regularization is used to define any modification made to the ML algorithm that attempts to reduce the generalization error, but not the training error [2]. That is, better results are sought during the testing stage, seeking to reduce overfitting. In particular, two regularization methods were used in this paper: Dropout, and Batch Normalization.

Dropout layer

The Dropout Layer, proposed by Hinton et al. [22], aims at preventing the overfitting of the weights of the neural network, by forcing the neurons of the dense layers to depend on group behavior. In practice, this layer is located between two dense layers of a deep neural network, deactivating a number of units of the preceding layer. When a unit is deactivated, its connections are also deactivated. In each iteration of the training process, each unit will have a Q probability of being deactivated, where $Q = 1 - P$, and P is the probability of keeping the unit [23]. It is important to note that when prediction is performed, all units are activated.

Batch normalization

The Batch Normalization (BN) layer, proposed by Ioffe & Szegedy [24], standardizes the output of a neural network layer by maintaining a mean of zero and a standard deviation of one. It is applied between both dense and convolutional layers.

Specifically, Batch Normalization is applied on a batch of input values $\mathcal{B} = \{x_1, ..., x_m\}$. First, the batch average is calculated, with Equation 1.

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

Then, the variance of the batch is calculated with the Equation 2.

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{2}$$

Having already calculated the mean and variance of the batch, this can be normalized, which is described in Equation 3, where ϵ is added to avoid calculating a very small square root, that can cause some problems.

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{3}$$

Finally, two parameters γ and β are added to the output of the normalization layer, which are trained together with the rest of the network parameters, and are used to shift or scale the previously normalized distribution towards a space that is convenient for the task at hand.

$$y_i = \gamma\hat{x}_i + \beta \tag{4}$$

The use of the Batch Normalization layer acts as a regularizer by adding a small amount of noise during the training process. It also allows the use of higher learning rates, because it prevents the neuron weights from reaching very high values due to the nature of the standardization [24].

2.2.2. Global average pooling

The Global Average Pooling (GAP) layer is designed to replace the dense layers at the end of a deep neural network. The idea is to add the GAP layer after the last convolutional layer to obtain a one-dimensional vector, with a length equal to the number of filters that the last convolutional layer has [25].

The GAP layer acts as a structural regularizer of the convolutional network, since it generates a summary of the features of each filter [25]. In addition, this layer does not present parameters that must be optimized; therefore, overfitting is not a problem in this layer.

2.2.3. Dilated convolution

The Dilated Convolution, proposed by Yu & Koltun [26] is a modification of the Convolutional layer that expands the receptive field of the layer. This modification allows expanding the receptive field without losing resolution or coverage [26]. In Figure 1, a receptive field of 3x3 with different rates of dilation, and its sliding on an input image with stride one, is observed.

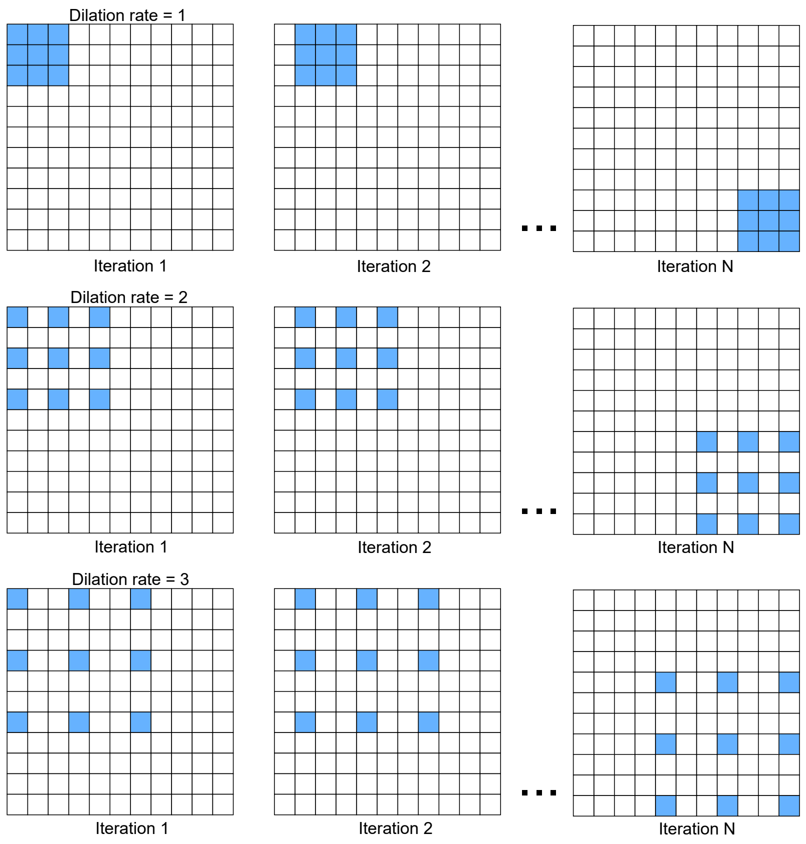


Figure 1. Visual representation of the calculation of the convolution operation with dilated receptive fields.

The use of Dilated Convolution, together with other regularization techniques such as Batch Normalization and GAP, can improve the results obtained in the classification tasks [18].

2.3. *Loss functions*

2.3.1. Cosine similarity loss

The use of the Cosine Similarity loss function [11] allowed training models by using small datasets. This function is calculated with the distance on the angle between the predicted probabilities output vector and the true classification vector of the class (One-hot Encoding). This is defined by the following equation:

$$\sigma_{cos}(a,b) = cos(a\angle b) = \frac{\langle a,b\rangle}{|a||_2 \cdot ||b||_2} \tag{5}$$

Where a and b are K-dimensional vectors, $\langle\cdot,\cdot\rangle$ denotes the Point Product and $||\cdot||_2$ the L^2 Normalization [11].

For a vector $a = (a_1, a_2, \dots, a_n)$ the L^2 Normalization is defined as:

$$||a||_2 = \sqrt{a_1^2 + \cdots + a_n^2} \tag{6}$$

For two vectors $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$ the Point Product is defined as:

$$\langle a,b\rangle = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + \cdots + a_n b_n \tag{7}$$

The Cosine Similarity function obtains better results than the Categorical Cross Entropy one [11]. However, it obtains similar results when a large dataset is used. The range of values that it delivers is [-1, 1], where 1 appears when the vectors have a greater degree of similarity to each other, -1 when they are more dissimilar, and 0 when they are orthogonal.

2.4. Learning Rate Scheduling

2.4.1. Cyclic learning rate

The Cyclic Learning Rate (CLR) [27] is a learning rate scheduling technique that assigns a learning rate value based on the calculation of a mathematical expression. Figure 2 shows three examples of learning rate scheduling

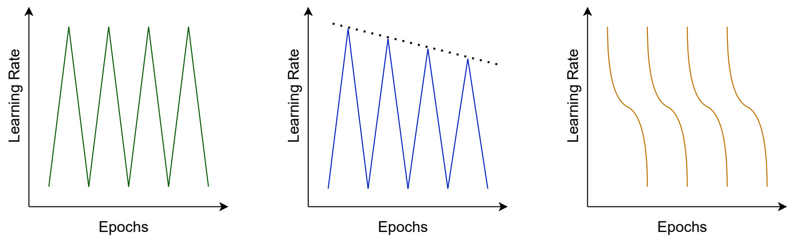


Figure 2. Examples of learning rate scheduling.

The CLR technique starts with a low learning rate lr_{min} and increases over a number of iterations s until it reaches a maximum lr_{max}. Then, it decreases again and so on during the training stage. CLR is defined as:

$$clr(t) = lr_{min} + (lr_{max} - lr_{min}) \cdot \left(\left| 1 - \frac{t}{2} - 2\left\lfloor \frac{t}{2s} \right\rfloor - 1 \right| \right) \quad (8)$$

Where t is the iteration of the training process. An example of CLR is shown in Figure 3 with $lr_{min} = 0.0001$, $lr_{max} = 0.001$ and $s = 10$.

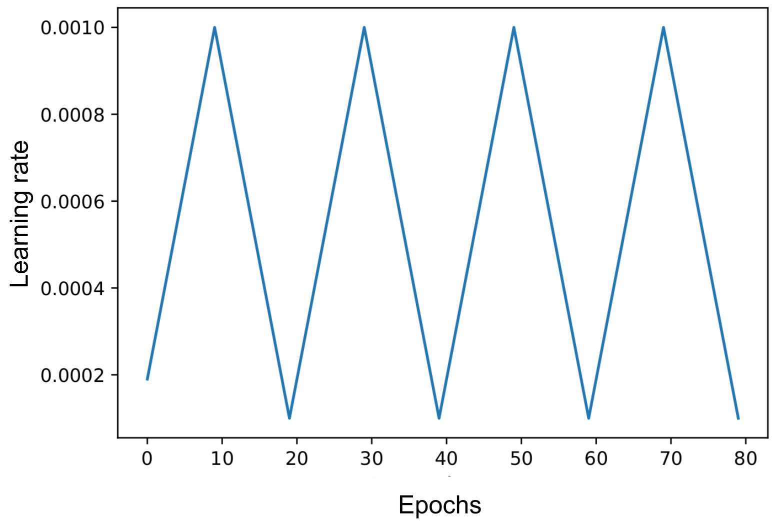


Figure 3. Example of cyclic learning rate.

In the work of Barz & Denzler [28], it is proposed to apply a decay to CLR using the following equation:

$$clr_{decay}(t) = \frac{clr(t)}{1 + (\delta - 1) \cdot \frac{t}{t_{max}}} \quad (9)$$

Where t_{max} is the number of iterations that the training stage will last and δ is the decay factor. The Figure 4 shows a graph of the change in the learning rate during 80 training iterations with $\delta = 10$, $lr_{min} = 0.0001$, $lr_{max} = 0.001$, $s = 10$ and $t_{max} = 80$.

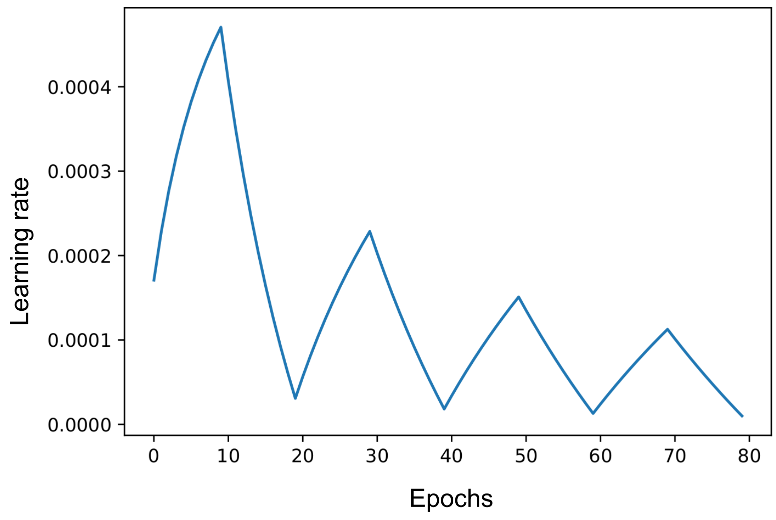


Figure 4. Example of cyclic learning rate decay.

2.5. Before training techniques

2.5.1. Data augmentation

Data Augmentation (DA) is a technique used to artificially increase the number of examples that a dataset has. The goal is to generate images that look like the original ones, and ideally they should not be distinguishable. To generate the new images, transformations are used on the elements of the dataset such as displacements, rotations, zoom in changes in brightness, etc.

Data Augmentation can be applied in three ways: statically, which implies applying the transformations to the images before starting the training process, therefore the dataset is augmented only once; continuously, in the same way as the previous case, where transformations were applied to each image, but only to those that were selected in the Batch during the training process. This was done every time a new training batch was selected. The last way was through the implementation of a type of Neural Network known as the Generative Adversarial Network [29]. This is a type of Neural Network that is made up of a Generator and a Discriminator. The Generator is responsible for producing false images and the Discriminator is responsible for discriminating whether the images it receives are real or not. The training process sought to produce a Generator which produced images that the Discriminator cannot differentiate. The latest form of Data Augmentation uses this type of neural network to generate images similar to those belonging to the dataset being augmented.

The use of this technique allowed the trained algorithm to be more tolerant of variations in the position, orientation, and illumination of objects in the image [30].

2.5.2. Transfer learning

Transfer Learning (TL) implies the use of a model that was previously trained with dataset X for task X on dataset Y for task Y. The idea is to use the previously trained model to obtain features that are useful on a close domain task. In Figure 5 a diagram is shown where the base of the task X model is transferred to the task Y model. Also, it is shown that some layers are added to fit the transferred model to the new task. By base of the model we refer to the full model but without the last layer.

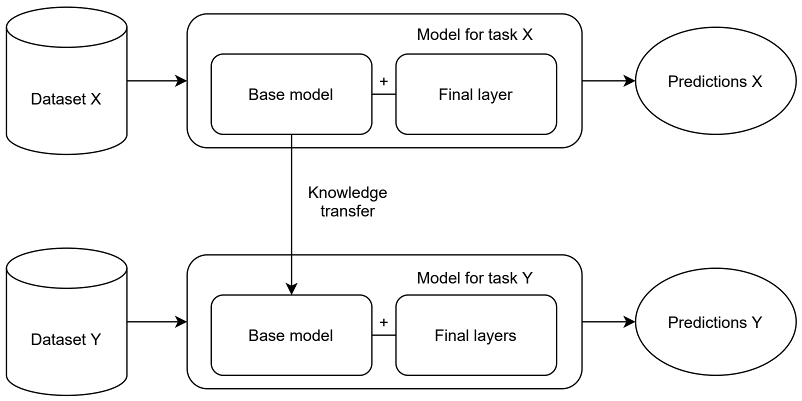


Figure 5. Diagram of Transfer Learning from task X to task Y.

The closer the domains of tasks X and Y, the better results Transfer Learning will achieve. In practice, the use of Transfer Learning speeds up training, since the amount of features that a certain network must learn is less due to the fact that it already has experience. In particular, the $EfficientNetB0$ [31] model was used for this work, due to its low number of parameters and high performance in the ImageNet [32] dataset. The $EfficientNetB0$ model requires that the input be 224 $\times$ 224 pixels in size and in RGB format, so it is usually necessary to apply a transformation to the images of the Y dataset, so that they can be used by the model.

2.6. After training techniques

2.6.1. Multi-model ensemble

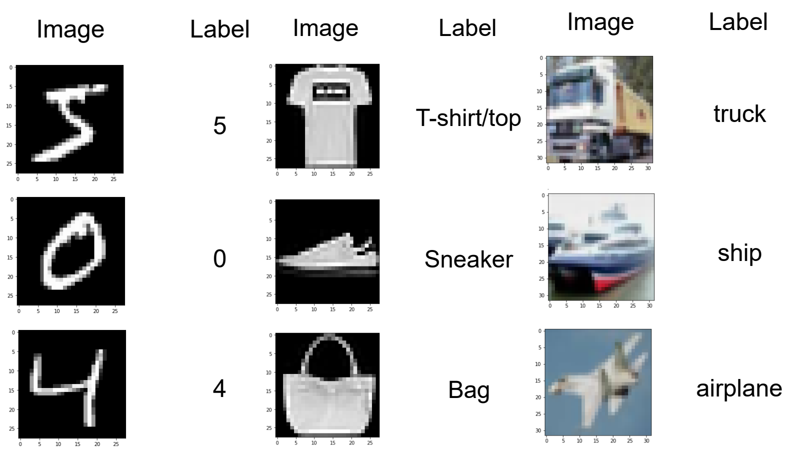
The Multi-Model Ensemble (MME) is a technique that uses the predictions of several models as votes, where the class that receives the most votes is the one selected as the class predicted by the MME. In other words, this technique allows harnessing the wisdom of crowds to make predictions [33].

3. Experimentation

In this section, the experimentation performed is presented, along with the small datasets used, algorithms implemented, metrics applied, and the system configuration employed for the execution.

3.1. Datasets

The experiments have been performed on the three datasets, shown in Figure 6.



(a) MNIST (b) Fashion MNIST (c) CIFAR-10

Figure 6. Visualization of the datasets used for experimentation.

doi:10.20944/preprints202201.0367.v1

3.1.1. MNIST

The MNIST dataset [1] comprises a total of 70,000 images, of which 60,000 are for training, and 10,000 are for the evaluation stage. Each image represents a digit of the base 10 number system; therefore, this set has ten classes, one for each digit. The images are 28x28 pixels in size and are in a gray-scale. Figure 6(a) shows three examples of this set, together with their respective label.

3.1.2. Fashion MNIST

The Fashion MNIST dataset [2] consists, just like the MNIST set, of 60,000 training images, 10,000 for the evaluation stage and 10 classes. The images refer to different types of clothing, and the classes are the following: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. The images are 28 x28 pixels in size and are in a gray-scale. Figure 6(b) shows three examples of the dataset and its labels.

3.1.3. CIFAR-10

The CIFAR-10 dataset[3] has 50,000 images for training, 10,000 images for testing and 10 classes. The images are of different vehicles and animals and their labels are the following: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each image has a size of 32x32 pixels in RBG format, that is, it has three information channels, one for the red color (R), another for the blue color (B) and a last channel for the green color (G). In Figure 6(c) three examples of the images that make up this dataset are presented, each one with its respective label.

3.1.4. Size of small subsets and preprocessing phase

Four small subsets were created with 100 (10 images per class), 500 (50 images per class), 2,500 (250 images per class), and 5,000 (500 images per class) images, respectively. Each subset is randomly selected and employed in the training phase. Then, the full evaluation set that each dataset has was employed in the testing phase. That is, the training phase is carried out with a small-sized subset, but the evaluation phase is carried out on the full evaluation set.

Once the datasets have been selected, a pre-processing phase was performed, which consisted in performing a normalization for each pixel of the images. A minimum-maximum normalization was employed, which allowed setting the range of values for each pixel from [0, 255] to [0, 1]. Additionally, the one-hot coding was applied to the labels used in each dataset.

3.2. Techniques and algorithms

The following techniques have been implemented: Dropout, Global Average Pooling, Batch Normalization, Cosine Similarity, Dilated Convolution, Cyclic Learning Rate Decay, Transfer Learning, Data Augmentation, and Multi-Model Ensemble. The experimentation consists of the implementation and execution of 13 algorithms. Each one of them was implemented for each subset of the three datasets. The list of the algorithms is shown below:

- Classical algorithms:
  - (1) Decision tree
  - (2) Random forest
  - (3) Support vector machine
- Deep learning:
  - (4) Base deep neural network

[1] http://yann.lecun.com/exdb/mnist/
[2] https://github.com/zalandoresearch/fashion-mnist
[3] https://www.cs.toronto.edu/~kriz/cifar.html

- (5) Deep neural network with dropout
- (6) Deep neural network with batch normalization
- (7) Deep neural network with dilated convolution
- (8) Deep neural network with cosine similarity loss
- (9) Deep neural network with global average pooling
- (10) Deep neural network with cyclic learning rate
- (11) C1: Combination of techniques (5 to 10)
- (12) C2: Add Transfer Learning and Data Augmentation to the previous combination of techniques.
- (13) Multi-model ensemble (4 to 12)

Table 2: Base neural network models used for experimentation. In parenthesis we denote the following: for the convolutional layers, the kernel size and amount of filters; for dense layers (also known as fully connected layers), the amount of units; for the max pooling layers, the pool size and, lastly, for the input layer, the input size.

MNIST Model	**FMNIST Model**	**CIFAR-10 Model**
Input (28x28)	Input (28x28)	Input (32x32x3)
Convolution (7x7, 64)	Convolution (7x7, 64)	Convolution (7x7, 128)
Max Pooling (4x4)	Max Pooling (2x2)	Max Pooling (4x4)
Convolution (5x5, 128)	Convolution (5x5, 128)	Convolution (7x7, 128)
Convolution (5x5, 64)	Max Pooling (2x2)	Convolution (7x7, 64)
Convolution (5x5, 64)	Convolution (5x5, 64)	Convolution (7x7, 64)
Convolution (7x7, 32)	Convolution (5x5, 64)	Flatten
Convolution (5x5, 128)	Convolution (5x5, 128)	Dense (48)
Flatten	Convolution (5x5, 128)	Dense (32)
Dense (32)	Flatten	Dense (10) (Output)
Dense (32)	Dense (64)	
Dense (10) (Output)	Dense (64)	
	Dense (64)	
	Dense (10) (Output)	

The architecture of the base Deep Neural Networks for the MNIST and CIFAR-10 sets was based on the architectures presented in the work of D'souza et al. [3], since they performed an exhaustive search of hyperparameters and layer composition in their work to find these models, and also used these datasets with similarly sized subsets in their experimentation. The base Deep Neural Network architecture used for Fashion MNIST is similar to the other two models, but was tweaked through trial and error starting with the CIFAR-10 model as a baseline. These models can be seen in Table 2. The definition of parameters of the classical algorithms was done using Grid Search, which is a method that tests a certain set of parameters and indicates the results for each combination. On the other hand, for Deep Learning algorithms, the parameters were adjusted by trial and error during several iterations of the training process.

Next, we specified where in the algorithm each technique is applied:

- For the neural network with Dropout, the Dropout layer was added between the dense layers of the base neural network.
- For the neural network with Global Average Pooling, the Flatten layer was changed to the Global Average Pooling layer, and the dense layers were removed, leaving only the last dense layer that performs the prediction.
- For the neural network with Batch Normalization, the Batch Normalization layer was added in the layers closest to the input of the neural network.
- For the neural network with Cosine Similarity, the categorical cross entropy loss function was changed to the Cosine Similarity loss function.
- For the Dilated Convolution neural network, the convolutional layers of the base neural network were modified to use dilation in their respective receptive fields.

- For the neural network with Cyclic Learning Rate Decay, the cyclic decay function was added so that the learning rate was modified during training.
- For the neural network with Combination of techniques (C1), Global Average Pooling, Dropout, Batch Normalization, Cosine similarity loss function, Dilated convolution, and Cyclic learning rate decay, were added.
- For the neural network with a combination of techniques plus Transfer Learning and Data Augmentation (C2), the convolutional layers were replaced in the EfficientNetB0 model; therefore, this neural network did not present dilated convolution. Also, for these experiments with transfer learning, in the MNIST and Fashion MNIST subsets, two information channels had to be added, since the transferred model received an image of three channels as input. This was done by copying the gray-scale channel twice to complete the three required channels. For the data of the CIFAR-10 set, it was not necessary to do this because they are RGB type, so they already have three information channels. In addition, for these experiments the upsample layer was also used, since the input received by the network was 224 x 224 pixels in size, so each image had to be enlarged before it could be used as input. For the training of this algorithm, the last six layers of the EfficientNetB0 model were trained together with the added layers, to adapt the task in question. Regarding data augmentation, this was applied continuously during the training of the algorithm, performing rotations, movements, and horizontal flipping.
- Finally, for the Multi-model ensemble, all deep neural network algorithms were used to make predictions and a definitive answer was obtained based on the class that obtained the most votes.

Each algorithm was trained with the four subsets of each dataset (MNIST, Fashion MNIST, and CIFAR-10), to note the change in performance as the available data increases. The same algorithm architecture was used for different subsets of a dataset, but with different parameter values when changing the data subset. The loss curves produced during the training of each neural network are shown in Appendix A.

3.3. Evaluation metrics

Each experiment was evaluated with the following metrics: Accuracy, Precision, Recall, F1, and Matthews correlation coefficient. The execution time for each algorithm was not considered because the experimentation was run in a shared server, hence there is a discrepancy between the load that the system presented when executing each experiment.

3.4. System configuration

The Postgraduate server of the University of the Bío-Bío was used to run the experiments. It had the following features:

- Ubuntu 18.04.03 operating system (4.15.0.101-generic kernel)
- 2 Intel Xeon Gold 5118 @ 2.30GHz processors
- 62.6 GB of RAM

Finally, the code for the experiments was implemented in the Python 3.6.9 programming language, and the following libraries were used: Scikit-learn [34], Tensorflow [35], Keras [36], among others. The implemented code is available at Github [4].

4. Results

This section shows the results of the experimentation done. These are presented in Tables 3 to 17, which have the values obtained by each algorithm for each metric. In addition to the tables, Figure 7 is shown, which has the bar graphs of the results to give a visual representation of each algorithm. For the tables, the values that appear

[4] https://github.com/Gonm1/smalldata

in parentheses refer to the percentage increase compared to the base neural network (without techniques).

4.1. MNIST subdatasets

Table 3: Results of the accuracy metric in the MNIST dataset. Best results are denoted in bold.

	Accuracy			
	10	50	250	500
Decision tree	0,418 (-38,53%)	0,601 (-34,67%)	0,74 (-23,55%)	0,789 (-19,65%)
Random forest	0,749 (10,15%)	0,89 (-3,26%)	0,933 (-3,62%)	0,946 (-3,67%)
Support vector Machine	0,792 (16,47%)	0,903 (-1,85%)	0,946 (-2,27%)	0,96 (-2,24%)
Deep neural network	0,68 (0,0%)	0,92 (0,0%)	0,968 (0,0%)	0,982 (0,0%)
Dropout	0,746 (9,71%)	0,903 (-1,85%)	0,944 (-2,48%)	0,979 (-0,31%)
Global average pooling	0,803 (18,09%)	0,902 (-1,96%)	0,982 (1,45%)	0,98 (-0,20%)
Batch Normalization	0,674 (-0,88%)	0,914 (-0,65%)	0,94 (-2,89%)	0,954 (-2,85%)
Cosine Similarity	0,703 (3,38%)	0,894 (-2,83%)	0,961 (-0,72%)	0,961 (-2,14%)
Dilated Convolution	0,693 (1,91%)	0,883 (-4,02%)	0,957 (-1,14%)	0,972 (-1,02%)
Cyclic learning rate decay	0,692 (1,76%)	0,89 (-3,26%)	0,967 (-0,10%)	0,978 (-0,41%)
Combination (C1)	0,804 (18,24%)	0,938 (1,96%)	0,973 (0,52%)	0,976 (-0,61%)
Combination (C2)	0,796 (17,06%)	0,93 (1,09%)	0,967 (-0,10%)	0,981 (-0,10%)
Multi-model ensemble	**0,868 (27,65%)**	**0,96 (4,35%)**	**0,986 (1,86%)**	**0,99 (0,81%)**

Table 4: Results of the precision metric in the MNIST dataset. Best results are denoted in bold.

	Precision			
	10	50	250	500
Decision tree	0,425 (-37,22%)	0,602 (-34,64%)	0,737 (-23,86%)	0,786 (-19,96%)
Random forest	0,748 (10,49%)	0,889 (-3,47%)	0,934 (-3,51%)	0,945 (-3,77%)
Support vector Machine	0,792 (16,99%)	0,902 (-2,06%)	0,946 (-2,27%)	0,959 (-2,34%)
Deep neural network	0,677 (0,0%)	0,921 (0,0%)	0,968 (0,0%)	0,982 (0,0%)
Dropout	0,775 (14,48%)	0,916 (-0,54%)	0,974 (0,62%)	0,98 (-0,2%)
Global average pooling	0,813 (20,09%)	0,916 (-0,54%)	0,982 (1,45%)	0,98 (-0,2%)
Batch Normalization	0,712 (5,17%)	0,916 (-0,54%)	0,943 (-2,58%)	0,956 (-2,65%)
Cosine Similarity	0,715 (5,61%)	0,898 (-2,5%)	0,962 (-0,62%)	0,962 (-2,04%)
Dilated Convolution	0,693 (2,36%)	0,882 (-4,23%)	0,957 (-1,14%)	0,972 (-1,02%)
Cyclic learning rate decay	0,701 (3,55%)	0,89 (-3,37%)	0,967 (-0,1%)	0,978 (-0,41%)
Combination (C1)	0,811 (19,79%)	0,939 (1,95%)	0,973 (0,52%)	0,976 (-0,61%)
Combination (C2)	0,811 (19,79%)	0,93 (0,98%)	0,967 (-0,1%)	0,981 (-0,1%)
Multi-model ensemble	**0,875 (29,25%)**	**0,96 (4,23%)**	**0,986 (1,86%)**	**0,99 (0,81%)**

Table 5: Results of the recall metric in the MNIST dataset. Best results are denoted in bold.

	Recall			
	10	50	250	500
Decision tree	0,412 (-38,6%)	0,597 (-35,04%)	0,737 (-23,86%)	0,787 (-19,78%)
Random forest	0,745 (11,03%)	0,888 (-3,37%)	0,934 (-3,51%)	0,945 (-3,67%)
Support vector Machine	0,788 (17,44%)	0,902 (-1,85%)	0,946 (-2,27%)	0,959 (-2,24%)
Deep neural network	0,671 (0,0%)	0,919 (0,0%)	0,968 (0,0%)	0,981 (0,0%)
Dropout	0,741 (10,43%)	0,901 (-1,96%)	0,974 (0,62%)	0,979 (-0,2%)
Global average pooling	0,801 (19,37%)	0,897 (-2,39%)	0,982 (1,45%)	0,98 (-0,1%)
Batch Normalization	0,671 (0,0%)	0,912 (-0,76%)	0,939 (-3,0%)	0,955 (-2,65%)
Cosine Similarity	0,698 (4,02%)	0,892 (-2,94%)	0,961 (-0,72%)	0,961 (-2,04%)
Dilated Convolution	0,686 (2,24%)	0,88 (-4,24%)	0,956 (-1,24%)	0,972 (-0,92%)
Cyclic learning rate decay	0,687 (2,38%)	0,889 (-3,26%)	0,967 (-0,1%)	0,978 (-0,31%)
Combination (C1)	0,799 (19,08%)	0,937 (1,96%)	0,973 (0,52%)	0,976 (-0,51%)
Combination (C2)	0,791 (17,88%)	0,929 (1,09%)	0,967 (-0,1%)	0,981 (0,0%)
Multi-model ensemble	**0,863 (28,61%)**	**0,959 (4,35%)**	**0,986 (1,86%)**	**0,99 (0,92%)**

Table 6: Results of the F1 metric in the MNIST dataset. Best results are denoted in bold.

	F1			
	10	50	250	500
Decision tree	0,412 (-38,59%)	0,597 (-35,15%)	0,737 (-23,97%)	0,787 (-19,88%)
Random forest	0,745 (11,26%)	0,888 (-3,37%)	0,934 (-3,62%)	0,945 (-3,67%)
Support vector Machine	0,788 (18,17%)	0,902 (-1,85%)	0,946 (-2,27%)	0,959 (-2,24%)
Deep neural network	0,671 (0,0%)	0,919 (0,0%)	0,968 (0,0%)	0,981 (0,0%)
Dropout	0,741 (10,66%)	0,901 (-1,74%)	0,974 (0,62%)	0,979 (-0,20%)
Global average pooling	0,801 (20,42%)	0,897 (-2,29%)	0,982 (1,45%)	0,98 (-0,10%)
Batch Normalization	0,671 (1,50%)	0,912 (-0,65%)	0,939 (-3,0%)	0,955 (-2,65%)
Cosine Similarity	0,698 (4,95%)	0,892 (-2,83%)	0,961 (-0,72%)	0,961 (-2,04%)
Dilated Convolution	0,686 (2,55%)	0,88 (-4,13%)	0,956 (-1,14%)	0,972 (-0,92%)
Cyclic learning rate decay	0,687 (2,55%)	0,889 (-3,26%)	0,967 (-0,10%)	0,978 (-0,31%)
Combination (C1)	0,799 (19,82%)	0,937 (1,96%)	0,973 (0,52%)	0,976 (-0,51%)
Combination (C2)	0,791 (18,77%)	0,929 (1,09%)	0,967 (-0,10%)	0,981 (0,0%)
Multi-model ensemble	**0,863 (29,43%)**	**0,959 (4,35%)**	**0,986 (1,86%)**	**0,99 (0,92%)**

Table 7: Results of the MCC metric in the MNIST dataset. Best results are denoted in bold.

	Matthews correlation coefficient			
	10	50	250	500
Decision tree	0,355 (-45,05%)	0,557 (-38,86%)	0,711 (-26,24%)	0,766 (-21,84%)
Random forest	0,722 (11,76%)	0,878 (-3,62%)	0,927 (-3,84%)	0,94 (-4,08%)
Support vector Machine	0,769 (19,04%)	0,892 (-2,09%)	0,94 (-2,49%)	0,955 (-2,55%)
Deep neural network	0,646 (0,0%)	0,911 (0,0%)	0,964 (0,0%)	0,98 (0,0%)
Dropout	0,722 (11,76%)	0,894 (-1,87%)	0,971 (0,73%)	0,977 (-0,31%)
Global average pooling	0,783 (21,21%)	0,892 (-2,09%)	0,98 (1,66%)	0,978 (-0,20%)
Batch Normalization	0,642 (-0,62%)	0,904 (-0,77%)	0,934 (-3,11%)	0,949 (-3,16%)
Cosine Similarity	0,671 (3,87%)	0,883 (-3,07%)	0,957 (-0,73%)	0,957 (-2,35%)
Dilated Convolution	0,661 (2,32%)	0,87 (-4,50%)	0,952 (-1,24%)	0,969 (-1,12%)
Cyclic learning rate decay	0,661 (2,32%)	0,878 (-3,62%)	0,964 (0,0%)	0,976 (-0,41%)
Combination (C1)	0,783 (21,21%)	0,931 (2,20%)	0,97 (0,62%)	0,973 (-0,71%)
Combination (C2)	0,775 (19,97%)	0,922 (1,21%)	0,963 (-0,10%)	0,979 (-0,10%)
Multi-model ensemble	**0,854 (32,20%)**	**0,955 (4,83%)**	**0,985 (2,18%)**	**0,989 (0,92%)**

4.2. Fashion MNIST subdatasets

Table 8: Results of the accuracy metric in the Fashion MNIST dataset. Best results are denoted in bold.

	Accuracy			
	10	50	250	500
Decision tree	0,463 (-25,8%)	0,649 (-14,83%)	0,729 (-9,1%)	0,756 (-11,48%)
Random forest	0,675 (8,17%)	0,778 (2,1%)	0,822 (2,49%)	0,837 (-1,99%)
Support vector Machine	0,685 (9,78%)	0,798 (4,72%)	0,837 (4,36%)	0,851 (-0,35%)
Deep neural network	0,624 (0,0%)	0,762 (0,0%)	0,802 (0,0%)	0,854 (0,0%)
Dropout	0,617 (-1,12%)	0,718 (-5,77%)	0,844 (5,24%)	0,87 (1,87%)
Global average pooling	0,683 (9,46%)	0,792 (3,94%)	0,854 (6,48%)	0,846 (-0,94%)
Batch Normalization	0,613 (-1,76%)	0,781 (2,49%)	0,816 (1,75%)	0,866 (1,41%)
Cosine Similarity	0,631 (1,12%)	0,743 (-2,49%)	0,803 (0,12%)	0,855 (0,12%)
Dilated Convolution	0,653 (4,65%)	0,78 (2,36%)	0,837 (4,36%)	0,856 (0,23%)
Cyclic learning rate decay	0,63 (0,96%)	0,738 (-3,15%)	0,827 (3,12%)	0,851 (-0,35%)
Combination (C1)	0,676 (8,33%)	0,781 (2,49%)	0,813 (1,37%)	0,843 (-1,29%)
Combination (C2)	0,655 (4,97%)	0,784 (2,89%)	0,834 (3,99%)	0,843 (-1,29%)
Multi-model ensemble	**0,702 (12,5%)**	**0,819 (7,48%)**	**0,874 (8,98%)**	**0,89 (4,22%)**

Table 9: Results of the precision metric in the Fashion MNIST dataset. Best results are denoted in bold.

	Precision			
	10	50	250	500
Decision tree	0,484 (-23,54%)	0,618 (-20,77%)	0,731 (-12,77%)	0,755 (-11,59%)
Random forest	0,674 (6,48%)	0,774 (-0,77%)	0,82 (-2,15%)	0,835 (-2,22%)
Support vector Machine	0,685 (8,21%)	0,799 (2,44%)	0,837 (-0,12%)	0,85 (-0,47%)
Deep neural network	0,633 (0,0%)	0,78 (0,0%)	0,838 (0,0%)	0,854 (0,0%)
Dropout	0,619 (-2,21%)	0,732 (-6,15%)	0,841 (0,36%)	0,872 (2,11%)
Global average pooling	0,681 (7,58%)	0,796 (2,05%)	0,857 (2,27%)	0,858 (0,47%)
Batch Normalization	0,63 (-0,47%)	0,794 (1,79%)	0,828 (-1,19%)	0,868 (1,64%)
Cosine Similarity	0,643 (1,58%)	0,746 (-4,36%)	0,811 (-3,22%)	0,862 (0,94%)
Dilated Convolution	0,639 (0,95%)	0,78 (0,0%)	0,84 (0,24%)	0,854 (0,0%)
Cyclic learning rate decay	0,633 (0,0%)	0,74 (-5,13%)	0,835 (-0,36%)	0,851 (-0,35%)
Combination (C1)	0,683 (7,9%)	0,789 (1,15%)	0,822 (-1,91%)	0,851 (-0,35%)
Combination (C2)	0,683 (7,9%)	0,803 (2,95%)	0,843 (0,6%)	0,854 (0,0%)
Multi-model ensemble	**0,699 (10,43%)**	**0,814 (4,36%)**	**0,874 (4,3%)**	**0,89 (4,22%)**

Table 10: Results of the recall metric in the Fashion MNIST dataset. Best results are denoted in bold.

	Recall			
	10	50	250	500
Decision tree	0,463 (-25,8%)	0,649 (-14,83%)	0,729 (-9,1%)	0,756 (-11,48%)
Random forest	0,675 (8,17%)	0,778 (2,1%)	0,822 (2,49%)	0,837 (-1,99%)
Support vector Machine	0,685 (9,78%)	0,798 (4,72%)	0,837 (4,36%)	0,851 (-0,35%)
Deep neural network	0,624 (0,0%)	0,762 (0,0%)	0,802 (0,0%)	0,854 (0,0%)
Dropout	0,617 (-1,12%)	0,718 (-5,77%)	0,844 (5,24%)	0,87 (1,87%)
Global average pooling	0,683 (9,46%)	0,792 (3,94%)	0,854 (6,48%)	0,846 (-0,94%)
Batch Normalization	0,613 (-1,76%)	0,781 (2,49%)	0,816 (1,75%)	0,866 (1,41%)
Cosine Similarity	0,631 (1,12%)	0,743 (-2,49%)	0,803 (0,12%)	0,855 (0,12%)
Dilated Convolution	0,653 (4,65%)	0,78 (2,36%)	0,837 (4,36%)	0,856 (0,23%)
Cyclic learning rate decay	0,63 (0,96%)	0,738 (-3,15%)	0,827 (3,12%)	0,851 (-0,35%)
Combination (C1)	0,676 (8,33%)	0,781 (2,49%)	0,813 (1,37%)	0,843 (-1,29%)
Combination (C2)	0,655 (4,97%)	0,784 (2,89%)	0,834 (3,99%)	0,843 (-1,29%)
Multi-model ensemble	**0,702 (12,5%)**	**0,819 (7,48%)**	**0,874 (8,98%)**	**0,89 (4,22%)**

Table 11: Results of the F1 metric in the Fashion MNIST dataset. Best results are denoted in bold.

	F1			
	10	50	250	500
Decision tree	0,462 (-25,72%)	0,618 (-18,04%)	0,728 (-9,34%)	0,754 (-11,71%)
Random forest	0,673 (8,2%)	0,775 (2,79%)	0,82 (2,12%)	0,834 (-2,34%)
Support vector Machine	0,682 (9,65%)	0,798 (5,84%)	0,837 (4,23%)	0,85 (-0,47%)
Deep neural network	0,622 (0,0%)	0,754 (0,0%)	0,803 (0,0%)	0,854 (0,0%)
Dropout	0,592 (-4,82%)	0,718 (-4,77%)	0,841 (4,73%)	0,87 (1,87%)
Global average pooling	0,679 (9,16%)	0,79 (4,77%)	0,853 (6,23%)	0,845 (-1,05%)
Batch Normalization	0,611 (-1,77%)	0,77 (2,12%)	0,815 (1,49%)	0,866 (1,41%)
Cosine Similarity	0,629 (1,13%)	0,735 (-2,52%)	0,803 (0,0%)	0,856 (0,23%)
Dilated Convolution	0,638 (2,57%)	0,777 (3,05%)	0,836 (4,11%)	0,854 (0,0%)
Cyclic learning rate decay	0,629 (1,13%)	0,734 (-2,65%)	0,829 (3,24%)	0,85 (-0,47%)
Combination (C1)	0,678 (9,0%)	0,783 (3,85%)	0,812 (1,12%)	0,845 (-1,05%)
Combination (C2)	0,644 (3,54%)	0,786 (4,24%)	0,833 (3,74%)	0,845 (-1,05%)
Multi-model ensemble	**0,695 (11,74%)**	**0,815 (8,09%)**	**0,874 (8,84%)**	**0,89 (4,22%)**

Table 12: Results of the MCC metric in the Fashion MNIST dataset. Best results are denoted in bold.

	Matthews correlation coefficient			
	10	50	250	500
Decision tree	0,406 (-30,48%)	0,617 (-16,51%)	0,699 (-10,84%)	0,729 (-13,01%)
Random forest	0,639 (9,42%)	0,754 (2,03%)	0,803 (2,42%)	0,819 (-2,27%)
Support vector Machine	0,65 (11,3%)	0,775 (4,87%)	0,819 (4,46%)	0,835 (-0,36%)
Deep neural network	0,584 (0,0%)	0,739 (0,0%)	0,784 (0,0%)	0,838 (0,0%)
Dropout	0,58 (-0,68%)	0,688 (-6,9%)	0,827 (5,48%)	0,856 (2,15%)
Global average pooling	0,649 (11,13%)	0,771 (4,33%)	0,838 (6,89%)	0,83 (-0,95%)
Batch Normalization	0,573 (-1,88%)	0,762 (3,11%)	0,797 (1,66%)	0,851 (1,55%)
Cosine Similarity	0,591 (1,2%)	0,717 (-2,98%)	0,782 (-0,26%)	0,839 (0,12%)
Dilated Convolution	0,617 (5,65%)	0,757 (2,44%)	0,819 (4,46%)	0,84 (0,24%)
Cyclic learning rate decay	0,589 (0,86%)	0,71 (-3,92%)	0,808 (3,06%)	0,834 (-0,48%)
Combination (C1)	0,641 (9,76%)	0,757 (2,44%)	0,793 (1,15%)	0,827 (-1,31%)
Combination (C2)	0,623 (6,68%)	0,761 (2,98%)	0,817 (4,21%)	0,827 (-1,31%)
Multi-model ensemble	**0,67 (14,73%)**	**0,799 (8,12%)**	**0,86 (9,69%)**	**0,878 (4,77%)**

4.3. CIFAR-10 subdatasets

Table 13: Results of the accuracy metric in the CIFAR-10 dataset. Best results are denoted in bold.

	Accuracy			
	10	50	250	500
Decision tree	0,164 (-42,46%)	0,192 (-46,22%)	0,229 (-53,17%)	0,252 (-56,17%)
Random forest	0,284 (-0,35%)	0,334 (-6,44%)	0,403 (-17,59%)	0,425 (-26,09%)
Support vector Machine	0,262 (-8,07%)	0,339 (-5,04%)	0,422 (-13,7%)	0,456 (-20,7%)
Deep neural network	0,285 (0,0%)	0,357 (0,0%)	0,489 (0,0%)	0,575 (0,0%)
Dropout	0,294 (3,16%)	0,389 (8,96%)	0,524 (7,16%)	0,604 (5,04%)
Global average pooling	0,293 (2,81%)	0,39 (9,24%)	0,498 (1,84%)	0,59 (2,61%)
Batch Normalization	0,295 (3,51%)	0,41 (14,85%)	0,512 (4,7%)	0,569 (-1,04%)
Cosine Similarity	0,269 (-5,61%)	0,375 (5,04%)	0,508 (3,89%)	0,573 (-0,35%)
Dilated Convolution	0,262 (-8,07%)	0,35 (-1,96%)	0,473 (-3,27%)	0,531 (-7,65%)
Cyclic learning rate decay	0,288 (1,05%)	0,36 (0,84%)	0,507 (3,68%)	0,562 (-2,26%)
Combination (C1)	0,291 (2,11%)	0,385 (7,84%)	0,469 (-4,09%)	0,544 (-5,39%)
Combination (C2)	**0,378 (32,63%)**	**0,584 (63,59%)**	**0,724 (48,06%)**	**0,762 (32,52%)**
Multi-model ensemble	0,319 (11,93%)	0,43 (20,45%)	0,56 (14,52%)	0,642 (11,65%)

Table 14: Results of the precision metric in the CIFAR-10 dataset. Best results are denoted in bold.

	Precision			
	10	50	250	500
Decision tree	0,184 (-35,44%)	0,196 (-48,01%)	0,238 (-52,78%)	0,254 (-55,52%)
Random forest	0,294 (3,16%)	0,331 (-12,2%)	0,396 (-21,43%)	0,418 (-26,8%)
Support vector Machine	0,279 (-2,11%)	0,348 (-7,69%)	0,42 (-16,67%)	0,456 (-20,14%)
Deep neural network	0,285 (0,0%)	0,377 (0,0%)	0,504 (0,0%)	0,571 (0,0%)
Dropout	0,3 (5,26%)	0,411 (9,02%)	0,526 (4,37%)	0,604 (5,78%)
Global average pooling	0,283 (-0,7%)	0,391 (3,71%)	0,487 (-3,37%)	0,609 (6,65%)
Batch Normalization	0,302 (5,96%)	0,417 (10,61%)	0,519 (2,98%)	0,585 (2,45%)
Cosine Similarity	0,288 (1,05%)	0,395 (4,77%)	0,503 (-0,2%)	0,58 (1,58%)
Dilated Convolution	0,272 (-4,56%)	0,353 (-6,37%)	0,478 (-5,16%)	0,531 (-7,01%)
Cyclic learning rate decay	0,282 (-1,05%)	0,353 (-6,37%)	0,498 (-1,19%)	0,563 (-1,4%)
Combination (C1)	0,302 (5,96%)	0,394 (4,51%)	0,475 (-5,75%)	0,541 (-5,25%)
Combination (C2)	**0,39 (36,84%)**	**0,617 (63,66%)**	**0,728 (44,44%)**	**0,768 (34,5%)**
Multi-model ensemble	0,315 (10,53%)	0,432 (14,59%)	0,558 (10,71%)	0,637 (11,56%)

Table 15: Results of the recall metric in the CIFAR-10 dataset. Best results are denoted in bold.

	Recall			
	10	50	250	500
Decision tree	0,164 (-42,46%)	0,192 (-46,22%)	0,229 (-53,17%)	0,252 (-56,17%)
Random forest	0,284 (-0,35%)	0,334 (-6,44%)	0,403 (-17,59%)	0,425 (-26,09%)
Support vector Machine	0,262 (-8,07%)	0,339 (-5,04%)	0,422 (-13,7%)	0,456 (-20,7%)
Deep neural network	0,285 (0,0%)	0,357 (0,0%)	0,489 (0,0%)	0,575 (0,0%)
Dropout	0,294 (3,16%)	0,389 (8,96%)	0,524 (7,16%)	0,604 (5,04%)
Global average pooling	0,293 (2,81%)	0,39 (9,24%)	0,498 (1,84%)	0,59 (2,61%)
Batch Normalization	0,295 (3,51%)	0,41 (14,85%)	0,512 (4,7%)	0,569 (-1,04%)
Cosine Similarity	0,269 (-5,61%)	0,375 (5,04%)	0,508 (3,89%)	0,573 (-0,35%)
Dilated Convolution	0,262 (-8,07%)	0,35 (-1,96%)	0,473 (-3,27%)	0,531 (-7,65%)
Cyclic learning rate decay	0,288 (1,05%)	0,36 (0,84%)	0,507 (3,68%)	0,562 (-2,26%)
Combination (C1)	0,291 (2,11%)	0,385 (7,84%)	0,469 (-4,09%)	0,544 (-5,39%)
Combination (C2)	**0,378 (32,63%)**	**0,584 (63,59%)**	**0,724 (48,06%)**	**0,762 (32,52%)**
Multi-model ensemble	0,319 (11,93%)	0,43 (20,45%)	0,567 (15,95%)	0,642 (11,65%)

Table 16: Results of the F1 metric in the CIFAR-10 dataset. Best results are denoted in bold.

	F1			
	10	50	250	500
Decision tree	0,165 (-40,65%)	0,193 (-45,94%)	0,217 (-55,07%)	0,246 (-56,77%)
Random forest	0,272 (-2,16%)	0,329 (-7,84%)	0,396 (-18,01%)	0,417 (-26,71%)
Support vector Machine	0,262 (-5,76%)	0,34 (-4,76%)	0,419 (-13,25%)	0,455 (-20,04%)
Deep neural network	0,278 (0,0%)	0,357 (0,0%)	0,483 (0,0%)	0,569 (0,0%)
Dropout	0,292 (5,04%)	0,393 (10,08%)	0,519 (7,45%)	0,601 (5,62%)
Global average pooling	0,277 (-0,36%)	0,371 (3,92%)	0,475 (-1,66%)	0,59 (3,69%)
Batch Normalization	0,291 (4,68%)	0,41 (14,85%)	0,507 (4,97%)	0,571 (0,35%)
Cosine Similarity	0,262 (-5,76%)	0,373 (4,48%)	0,504 (4,35%)	0,569 (0,0%)
Dilated Convolution	0,254 (-8,63%)	0,341 (-4,48%)	0,469 (-2,9%)	0,524 (-7,91%)
Cyclic learning rate decay	0,27 (-2,88%)	0,349 (-2,24%)	0,5 (3,52%)	0,559 (-1,76%)
Combination (C1)	0,293 (5,4%)	0,386 (8,12%)	0,466 (-3,52%)	0,54 (-5,1%)
Combination (C2)	**0,351 (26,26%)**	**0,574 (60,78%)**	**0,717 (48,45%)**	**0,758 (33,22%)**
Multi-model ensemble	0,31 (11,51%)	0,426 (19,33%)	0,56 (15,94%)	0,639 (12,3%)

Table 17: Results of the MCC metric in the CIFAR-10 dataset. Best results are denoted in bold.

	Matthews correlation coefficient			
	10	50	250	500
Decision tree	0,071 (-65,53%)	0,102 (-64,58%)	0,146 (-66,44%)	0,17 (-67,8%)
Random forest	0,207 (0,49%)	0,261 (-9,38%)	0,337 (-22,53%)	0,362 (-31,44%)
Support vector Machine	0,181 (-12,14%)	0,266 (-7,64%)	0,357 (-17,93%)	0,396 (-25,0%)
Deep neural network	0,206 (0,0%)	0,288 (0,0%)	0,435 (0,0%)	0,528 (0,0%)
Dropout	0,216 (4,85%)	0,322 (11,81%)	0,473 (8,74%)	0,561 (6,25%)
Global average pooling	0,217 (5,34%)	0,326 (13,19%)	0,445 (2,3%)	0,546 (3,41%)
Batch Normalization	0,217 (5,34%)	0,345 (19,79%)	0,46 (5,75%)	0,522 (-1,14%)
Cosine Similarity	0,191 (-7,28%)	0,307 (6,6%)	0,454 (4,37%)	0,528 (0,0%)
Dilated Convolution	0,184 (-10,68%)	0,28 (-2,78%)	0,415 (-4,6%)	0,48 (-9,09%)
Cyclic learning rate decay	0,212 (2,91%)	0,291 (1,04%)	0,453 (4,14%)	0,514 (-2,65%)
Combination (C1)	0,212 (2,91%)	0,317 (10,07%)	0,412 (-5,29%)	0,494 (-6,44%)
Combination (C2)	**0,315 (52,91%)**	**0,544 (88,89%)**	**0,695 (59,77%)**	**0,738 (39,77%)**
Multi-model ensemble	0,245 (18,93%)	0,368 (27,78%)	0,519 (19,31%)	0,603 (14,2%)

5. Discussion

For the MNIST and Fashion MNIST subsets, the Multi- Model Ensemble (MME) with all neural networks was the technique that gave the best results for all subset sizes. In addition, it showed the largest increases with respect to the base neural network (see Tables 3-12). On the other hand, for the CIFAR-10 sets, it was observed that the MME improved the results with respect to most of the algorithms, but was surpassed by the Combination of techniques C2 (see Tables 13-17). This is thought to be because the MME

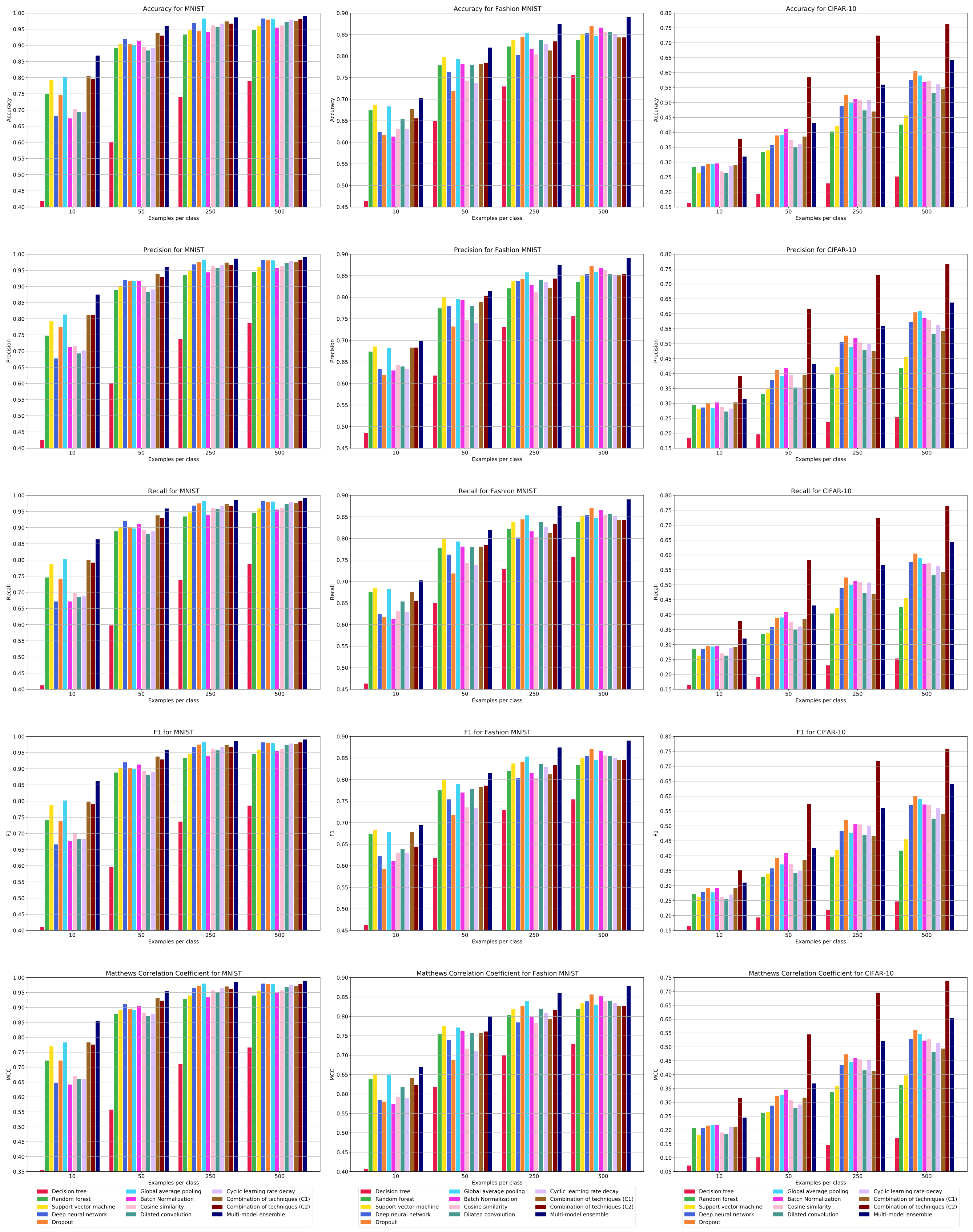


Figure 7. Bar plots of metrics obtained by algorithms for MNIST, Fashion MNIST and CIFAR-10 datasets.

Table 18: Techniques and algorithms with higher results in Accuracy, F1 and MCC for each subset of data.

		Accuracy		**F1**		**MCC**	
	Subset	Highest	Second highest	Highest	Second highest	Highest	Second highest
MNIST	10	MME	C1	MME	GAP	MME	C1
	50	MME	C1	MME	C1	MME	C1
	250	MME	GAP	MME	GAP	MME	GAP
	500	MME	DNN	MME	DNN/C2	MME	DNN
Fashion MNIST	10	MME	SVM	MME	SVM	MME	SVM
	50	MME	SVM	MME	SVM	MME	SVM
	250	MME	GAP	MME	GAP	MME	GAP
	500	MME	Dropout	MME	Dropout	MME	Dropout
CIFAR-10	10	C2	MME	C2	MME	C2	MME
	50	C2	MME	C2	MME	C2	MME
	250	C2	MME	C2	MME	C2	MME
	500	C2	MME	C2	MME	C2	MME

operates as an average of the results, so the average result is lower than the highest result.

Regarding the Combination of techniques C2, the results of CIFAR-10 showed a significant increase in the metrics obtained, for example, the MCC metric showed an increase of between 39% and 88% (see Table 17). Therefore, it was observed that the transfer learning technique is able to help compensate for the limitation in the amount of training data, as indicated in the work of Rajpurkar et al. [19]. However, this did not happen in the MNIST and Fashion MNIST datasets, where the Combination of techniques C2 presented lower results than the Combination of techniques C1. This is because data from the CIFAR-10 dataset, unlike the MNIST and Fashion MNIST datasets, come from a domain close to those of the ImagenNet set. Therefore, the Transfer Learning techniques added to the Resistance to variations provided by the Data Augmentation technique had a high effect on the performance obtained. This is not the case for the experiments on the MNIST and Fashion MNIST datasets.

The use of the Global Average Pooling technique, by itself, on the base neural network, showed improvements in the MCC of up to 21.21% and reductions of only -2.09%, with the largest increases in the smallest subsets (see Tables 7, 12 and 17). In addition, as the work of Zhou et al. [18] indicates, it is found that the application of the Global Average Pooling layer reduces the number of network parameters, due to the elimination of dense layers.

In the work of Barz & Denzler [11], it is established that the use of the Cosine Similarity loss function improves the metrics obtained when using a small dataset, when comparing it with the categorical cross entropy loss function. However, the results obtained in this paper do not allow ratifying what had been indicated. This is because, in the MNIST and Fashion MNIST sets, an increase was observed when applying this technique in the subsets of 10 images per class. However, there was a reduction in the rest of subsets, and for the subsets of CIFAR-10, the opposite was observed, with a reduction in the subset of 10 images per class and an increase in the rest of the subsets.

Regarding the classical learning algorithms, the Support Vector Machine stood out, which obtained competitive results with Deep Learning techniques, especially in the subsets of 10 images per class of the MNIST and Fashion MNIST sets (see Tables 3-12).

In general, in the results obtained, the effect that the amount of data had on the performance of each algorithm could be clearly seen (see Figure 7), where by increasing the size of the training subset, the values of the evaluated metrics also saw an increase.

For example, in Table 18 it is possible to see the techniques or algorithms that obtained the highest results in the Accuracy, F1, and MCC metrics. From the table, it can be noted that there were some variations on which was the most appropriate method, depending on the metric that was chosen as the most relevant. It is recommended that, upon searching for the most appropriate technique, the metric to be optimized for the given task be taken into account, so as to facilitate the choice of the most appropriate algorithm.

Also, from Table 18 it is observed that the techniques that obtained the best results in the Accuracy, F1, and MCC metrics are: Dropout, Global Average Pooling, Multi-model Ensemble, and Combination of techniques (C2), that use Transfer Learning and Data Augmentation and the Support Vector Machine algorithm.

6. Conclusion

In this paper, a thorough experimentation to find a technique or combination of them to mitigate the deficiency in the amount of data in deep learning models was performed. The most important algorithms were studied, implemented, analyzed, tested and compared. In particular, three classical algorithms and nine Deep Neural Network algorithms were implemented. Considering classical algorithms, Deep Learning techniques and combinations, with a total of 13 different algorithms, were trained on four subsets of different sizes (10, 50, 250, and 500 images per class) of the MNIST, Fashion MNIST and CIFAR-10 datasets. The subsets were randomly chosen from the training set of each dataset. In total, 156 models were trained. The results and discussion were supported by 30 tables and graphs.

From the experimentation carried out, it is concluded that it is possible to find a technique or combination thereof that allows mitigating the deficiency in the amount of data, but this technique or their combination will be different for each dataset, amount of data and, possibly, metric. Therefore, it is recommended to experiment with these techniques and algorithms when facing a lack of training data, since it is expected that they will present results similar to those obtained during experimentation. However, the latter requires further study and experimentation to see if it holds up across different small datasets.

As future work, it would be advisable to perform a deeper study on the effects of the parameters employed in the techniques studied and discussed in this work. For example, it would be very interesting to analyze the effect of increasing the width of the network (that uses one or more techniques) on the overall accuracy of the model, or any other metric. Additionally, it would be very interesting to study which technique or the combination of them contribute more to improve the base model, we have an approximate idea in the results, however it might be useful to explore the top 3 accuracy techniques combined in further work.

Funding: This research received no external funding.

Data Availability Statement: All code is available at https://github.com/Gonm1/smalldata. MNIST dataset is available at http://yann.lecun.com/exdb/mnist/, Fashion MNIST dataset is available at https://github.com/zalandoresearch/fashion-mnist, CIFAR-10 dataset is available at https://www.cs.toronto.edu/~kriz/cifar.html.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this appendix, Figures A1, A2 and A3 are presented, which show the loss curves of the trained neural networks for the three datasets used. The configuration parameters used for each algorithm can be accessed in the public repository provided in this the following link: https://github.com/Gonm1/smalldata.

References

1. Press, G. 7 Indicators Of The State-Of-Artificial Intelligence (AI), April 2019, 2019.
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning*; MIT Press, 2016; p. 775.
3. D'souza, R.N.; Huang, P.Y.; Yeh, F.C. Structural Analysis and Optimization of Convolutional Neural Networks with a Small Sample Size. *Scientific Reports* **2020**, *10*, 834. doi: 10.1038/s41598-020-57866-2.
4. Chen, J.; Chen, Y.; Du, X.; Li, C.; Lu, J.; Zhao, S.; Zhou, X. Big data challenge: a data management perspective. *Frontiers of Computer Science* **2013**, *7*, 157–164. doi:10.1007/s11704-013-3903-7.

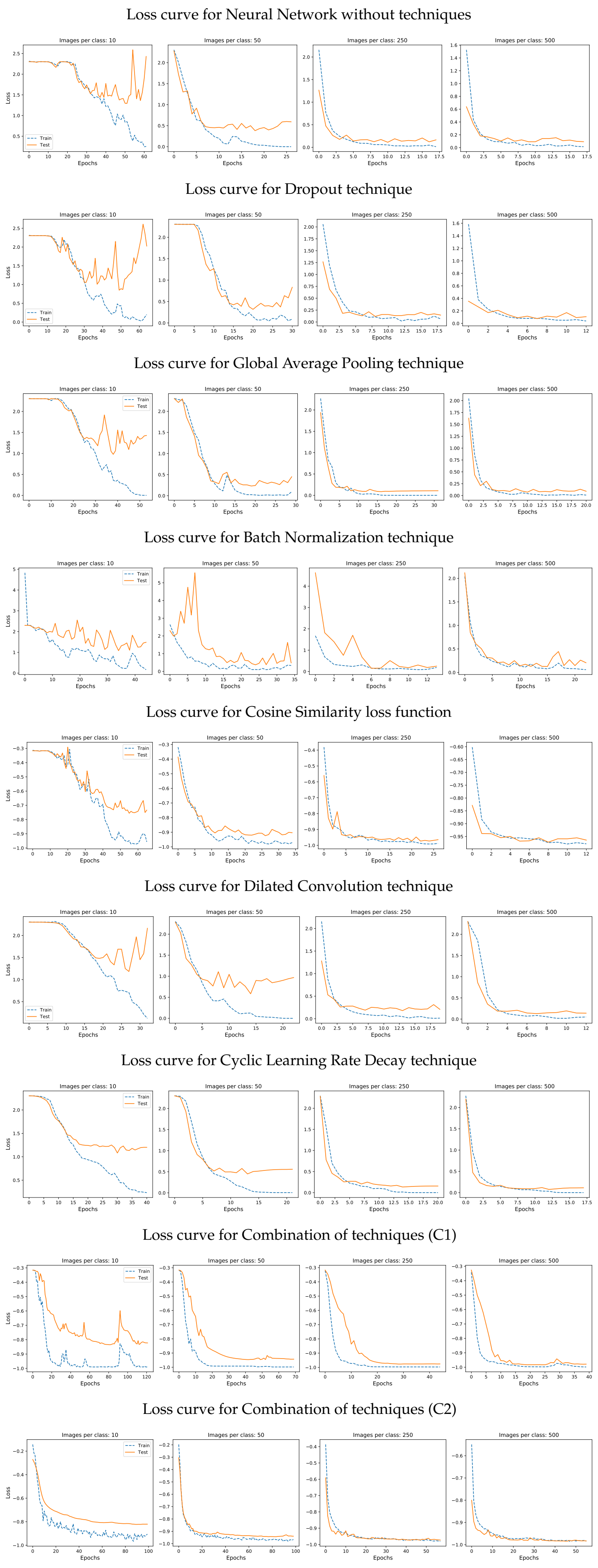


Figure A1. Loss curves for the algorithms trained with MNIST dataset.

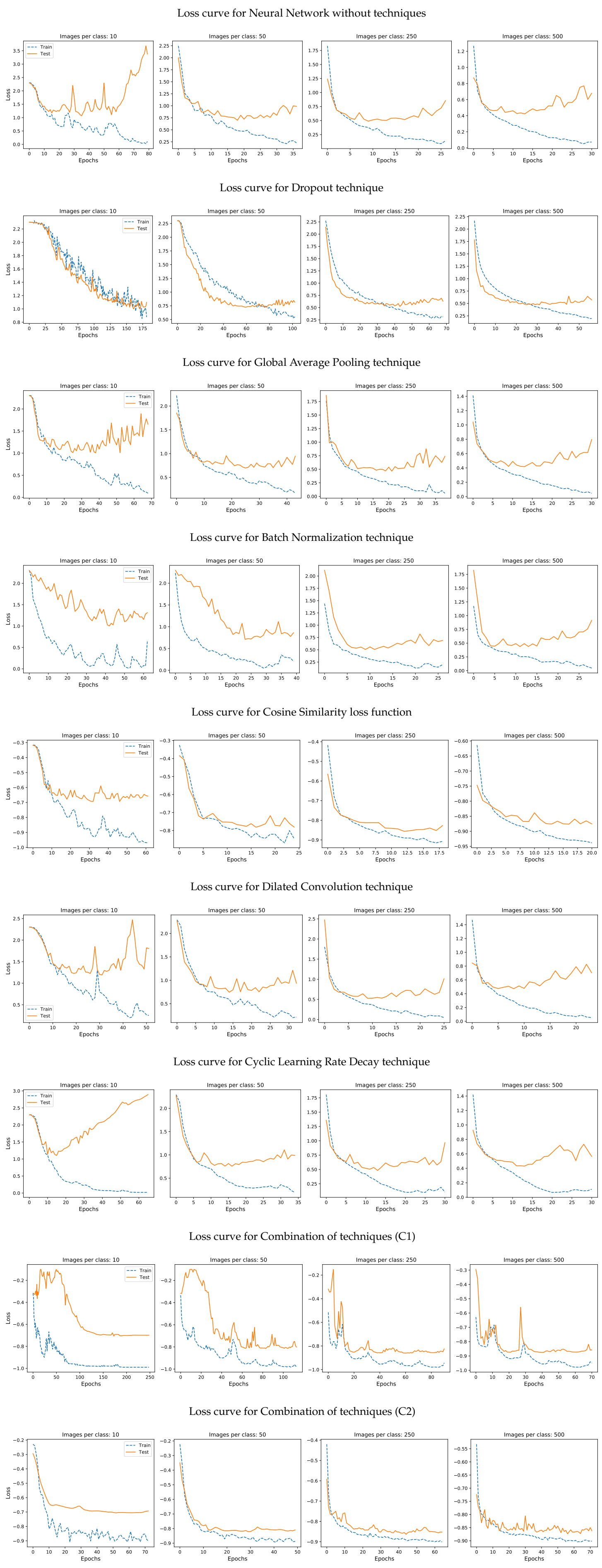


Figure A2. Loss curves for the algorithms trained with Fashion MNIST dataset.

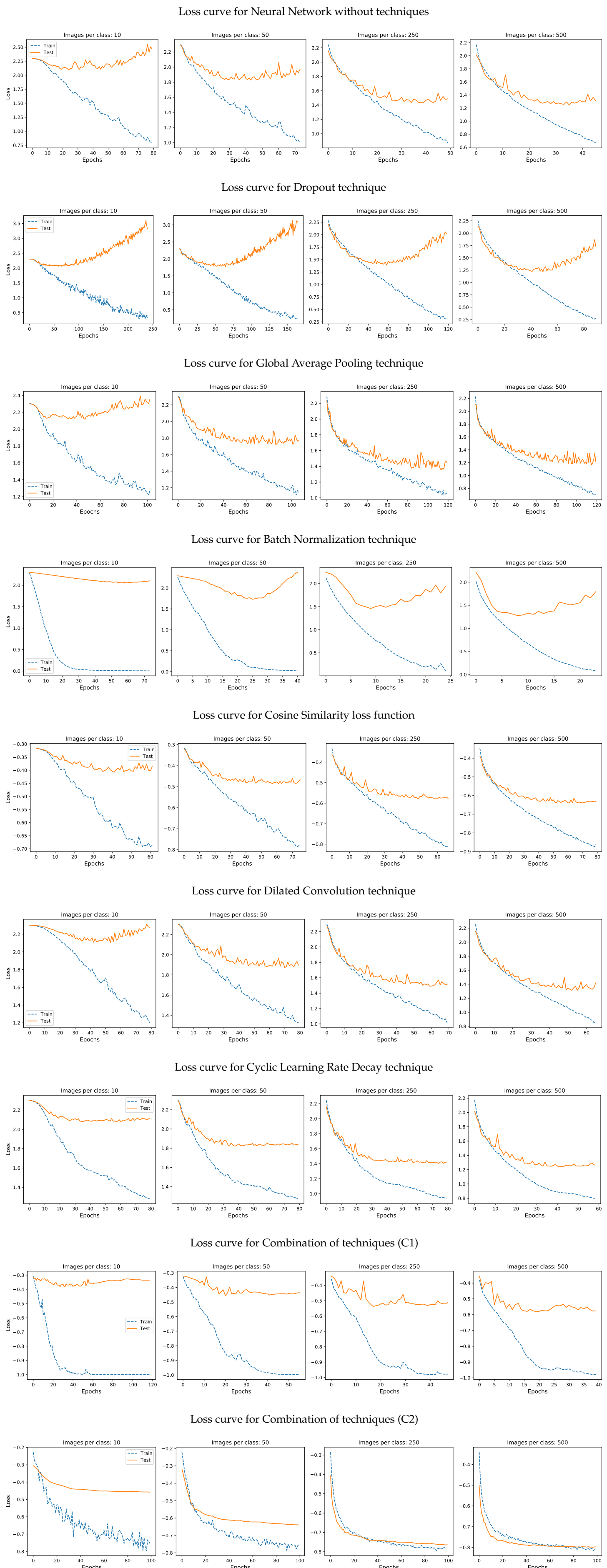


Figure A3. Loss curves for the algorithms trained with CIFAR-10 dataset.

5. Yang, Z.; Al-Bahrani, R.; Reid, A.C.E.; Papanikolaou, S.; Kalidindi, S.R.; Liao, W.k.; Choudhary, A.; Agrawal, A. Deep learning based domain knowledge integration for small datasets: Illustrative applications in materials informatics. 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019, Vol. 2019-July, pp. 1–8. doi:10.1109/IJCNN.2019.8852162.
6. Litjens, G.; Kooi, T.; Bejnordi, B.E.; Setio, A.A.A.; Ciompi, F.; Ghafoorian, M.; van der Laak, J.A.; van Ginneken, B.; Sánchez, C.I. A survey on deep learning in medical image analysis. *Medical Image Analysis* **2017**, *42*, 60–88. doi:10.1016/j.media.2017.07.005.
7. Zhang, S.; Han, F.; Liang, Z.; Tan, J.; Cao, W.; Gao, Y.; Pomeroy, M.; Ng, K.; Hou, W. An investigation of CNN models for differentiating malignant from benign lesions using small pathologically proven datasets. *Computerized Medical Imaging and Graphics* **2019**, *77*, 101645. doi:10.1016/j.compmedimag.2019.101645.
8. Le, X.; Mei, J.; Zhang, H.; Zhou, B.; Xi, J. A learning-based approach for surface defect detection using small image datasets. *Neurocomputing* **2020**. doi:10.1016/j.neucom.2019.09.107.
9. Oviedo, F.; Ren, Z.; Sun, S.; Settens, C.; Liu, Z.; Hartono, N.T.P.; Ramasamy, S.; DeCost, B.L.; Tian, S.I.P.; Romano, G.; Gilad Kusne, A.; Buonassisi, T. Fast and interpretable classification of small X-ray diffraction datasets using data augmentation and deep neural networks. *npj Computational Materials* **2019**, *5*, 60, [1811.08425]. doi:10.1038/s41524-019-0196-x.
10. Feng, S.; Zhou, H.; Dong, H. Using deep neural network with small dataset to predict material defects. *Materials & Design* **2019**, *162*, 300–310. doi:10.1016/j.matdes.2018.11.060.
11. Barz, B.; Denzler, J. Deep Learning on Small Datasets without Pre-Training using Cosine Loss. *openaccess.thecvf.com* **2019**, [1901.09054].
12. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2015**, *2016-Decem*, 770–778, [1512.03385].
13. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324. doi:10.1109/5.726791.
14. Lu, C.; Li, W. Ship Classification in High-Resolution SAR Images via Transfer Learning with Small Training Dataset. *Sensors* **2018**, *19*, 63. doi:10.3390/s19010063.
15. Gozes, O.; Greenspan, H. Deep Feature Learning from a Hospital-Scale Chest X-ray Dataset with Application to TB Detection on a Small-Scale Dataset. 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2019, pp. 4076–4079, [1906.00768]. doi:10.1109/EMBC.2019.8856729.
16. Pasupa, K.; Sunhem, W. A comparison between shallow and deep architecture classifiers on small dataset. 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE). IEEE, 2016, pp. 1–6. doi:10.1109/ICITEED.2016.7863293.
17. Brigato, L.; Iocchi, L. A Close Look at Deep Learning with Small Data. *arXiv preprint* **2020**, [2003.12843].
18. Zhou, S.; Chen, C.; Han, G.; Hou, X. Deep Convolutional Neural Network with Dilated Convolution Using Small Size Dataset. 2019 Chinese Control Conference (CCC). IEEE, 2019, Vol. 2019-July, pp. 8568–8572. doi:10.23919/ChiCC.2019.8865226.
19. Rajpurkar, P.; Park, A.; Irvin, J.; Chute, C.; Bereket, M.; Mastrodicasa, D.; Langlotz, C.P.; Lungren, M.P.; Ng, A.Y.; Patel, B.N. AppendiXNet: Deep Learning for Diagnosis of Appendicitis from A Small Dataset of CT Exams Using Video Pretraining. *Scientific Reports* **2020**, *10*, 3958. doi:10.1038/s41598-020-61055-6.
20. Yang, J.; Xie, Y.; Liu, L.; Xia, B.; Cao, Z.; Guo, C. Automated Dental Image Analysis by Deep Learning on Small Dataset. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2018, Vol. 1, pp. 492–497. doi:10.1109/COMPSAC.2018.00076.
21. Lee, H.; Yune, S.; Mansouri, M.; Kim, M.; Tajmir, S.H.; Guerrier, C.E.; Ebert, S.A.; Pomerantz, S.R.; Romero, J.M.; Kamalian, S.; Gonzalez, R.G.; Lev, M.H.; Do, S. An explainable deep-learning algorithm for the detection of acute intracranial haemorrhage from small datasets. *Nature Biomedical Engineering* **2019**, *3*, 173–182. doi:10.1038/s41551-018-0324-9.
22. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint* **2012**, [1207.0580].
23. Baldi, P.; Sadowski, P. Understanding Dropout. Technical report, University of California, 2013.

24. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *32nd International Conference on Machine Learning, ICML 2015* **2015**, *1*, 448–456, [1502.03167].
25. Lin, M.; Chen, Q.; Yan, S. Network In Network. *arXiv preprint* **2013**, p. 10, [1312.4400].
26. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* **2015**, [1511.07122].
27. Smith, L.N. Cyclical Learning Rates for Training Neural Networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017* **2015**, pp. 464–472, [1506.01186].
28. Barz, B.; Denzler, J. Deep Learning is not a Matter of Depth but of Good Training. International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI), 2018.
29. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Communications of the ACM* **2020**, *63*, 139–144, [1406.2661]. doi:10.1145/3422622.
30. Aurélien, G. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*; O'Reilly Media, 2019; p. 856.
31. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *36th International Conference on Machine Learning, ICML 2019* **2019**, *2019-June*, 10691–10700, [1905.11946].
32. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Kai Li.; Li Fei-Fei. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.
33. Kotu, V.; Deshpande, B. Data Mining Process. In *Predictive Analytics and Data Mining*; Elsevier, 2015; pp. 17–36. doi:10.1016/B978-0-12-801460-8.00002-1.
34. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.
35. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mane, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viegas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint* **2016**, [1603.04467].
36. Chollet, F.; others. Keras, 2015.