Best Practices to Train Accurate Deep Learning Models: A General Methodology

Alberto Nogales*, Ana M. Maitín, Álvaro J. García-Tejedor
Center for Studies and Innovation in Knowledge Management (CEIEC), Francisco de Vitoria University, 28223 Pozuelo de Alarcón, Spain

***Abstract*—In recent years, the field of computer science has experienced great changes due to the incredible advances in the field of artificial intelligence. Deep Learning models are responsible for most of them since the biggest milestones occurred in 2012 when AlexNet won the image classification challenge called ImageNet. These models have demonstrated great performance in different types of complex tasks like image restoration, medical diagnosis, or object recognition. Their disadvantages are related to their high data dependency, which forces experts in the field to follow a precise methodology to obtain accurate models. In this study, we describe a complete workflow that begins with the management of the raw data until the in-depth interpretation of the performance of the models. This should be taken as a high-level consultation document describing good practices that should be applied. Apart from the step-by-step methodology, we present different use cases that correspond to the two main problems of the field: classification and regression.**

***Keywords*—Methodology; artificial intelligence; deep learning**

I. Introduction

If we talk about what has been one of the biggest advances in recent history, we must talk about the revolution of artificial intelligence (AI). AI is the domain within computer science that analyzes and interprets the processes underlying intelligent actions in humans, aiming to replicate these actions in machines, not always employing the same mechanisms [1]. These advancements have been fostered by two main factors. First, the creation and availability of large amounts of data have never been seen before. Second, the affordable price of hardware to compute this data and train the AI models.

But if we must talk about the milestone in the modern AI era, we must talk about Deep Learning (DL) models. This is a family of models whose term was coined in 2006 by Hinton. They are defined as multiple-layer models using hierarchical Artificial Neural Networks (ANNs) with the capacity to progressively learn data representations, beginning with the input data and advancing to higher levels of abstraction [2]. Its biggest milestone occurred in 2012 when AlexNet was the first DL model that won an image classification challenge called ImageNet [3].

Deep Learning models have been demonstrated to have high performance in solving many different complex tasks. Nevertheless, some weaknesses face any expert in the field constantly, such as poor interpretability, data dependency, or lack of generalization in a broad range of datasets. The last two, apart from being interrelated, are common to all the implementation workflows of these models. Normally, large amounts of data are needed to train accurate models, but there is also a need to generalize and avoid overfitting. The latter is described in [4] as the situation in which the model learns characteristics associated with noise or variability within the data, rather than the fundamental distribution of the training data. So, avoiding overfitting is the main condition to obtain a model that performs well generalize.

Overfitting is tested at the end of the process of model training. At this point in the process, different measures of a metric are compared, corresponding these values to the different sets into which the data in origin is split. The success of the training process depends on different decisions to be taken, which are related to the type of problem to be solved, the nature of the data, and other conditions. The large number of decisions to take and the lack of knowledge about what they depend on have allowed the publication of papers where the overfitting of the models is noticeable. The motivation of this study is to compile and explain step-by-step how to perform accurate processes to train Deep Learning models.

The work aims to provide resource information for researchers so they can know which steps to apply and understand them. This will be very useful for scientists without a deep understanding of deep learning terms. In the case of experts in the field, they could have a guide of best practices to follow. In this sense, the main contribution is to compile in a single document the explanation of each step. It will not only describe the method step-by-step, but we will also provide why each step is necessary and how to apply it depending on several conditions, such as the use case to solve or the type of data. As we want this study to be accessible to any researcher interested in the field, we are providing a high-level explanation of terms, strategies, etc., with only basic mathematical formalization.

The rest of the study is organized as follows: Section II describes all the antecedents of the DL field, so a general idea of different terms could be obtained, with the possible applications DL could have. Section III explains step-by-step the proposed methodology to obtain accurate models. Section IV shows different use cases where the methodology is applied. Finally, Section V compiles different conclusions related to what has been explained in the work.

II. Antecedents

DL models are very diverse, with their performance tied to the nature and quality of the training data. This section explores the different models used in the field. Fig. 1 shows the hierarchical organization of these models depending on the way

*Corresponding author.

they learn. An in-depth description of the models will be given later.

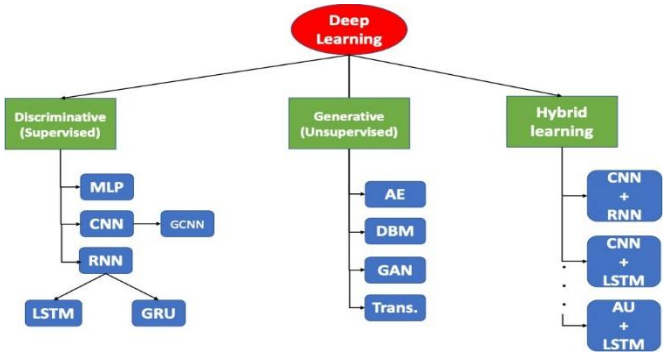

Fig. 1. Classification of Deep Learning models based on their different training strategies.

DL models are first categorized based on their approach to learning from data (training stage), which includes supervised and unsupervised methods. The first category corresponds to those using labelled data for training, where the model understands the relationship between input data and expected output, which is a discriminative task.

The Multilayer Perceptron (MLP) represents the most basic form of a DL model. Its architecture consists of an input layer, multiple hidden layers, and an output layer. It is typically used to predict a particular numeric value, but classification problems can also be solved, for example, [5]. Take the benefit of using MLPs to predict the density of 48 refrigerant systems.

Convolutional Neural Networks (CNNs) stand out as one of the most used models, particularly in computer vision applications [3]. Their primary strength lies in their capability to detect patterns in a spatially invariant manner. This characteristic enables them to recognize specific structures in an image, regardless of their location. More recently, there has been a development of a specialized type of CNN known as Graph Convolutional Neural Networks (GCNNs). Introduced by [6], GCNNs are designed to process graph-structured data. This model encodes both the structure of a graph and the features of its nodes. Examples of the application of these models can be found in the following papers [7], where CNNs are used for the automated assessment of damage type and severity in wooden structures [8], and a GCNN model is used for identifying essential genes in organisms.

Recurrent Neural Networks (RNNs), as defined by [9], employ an input vector of variable length and recursively apply a transition function to their internal hidden state vector h_t. They are particularly useful for time series data structures. Within the family of RNNs, there are two specific subtypes known as Long Short-Term Memory (LSTMs) and another called Gated Recurrent Units (GRUs). LSTMs, introduced by [10], were devised to effectively handle noisy or ambiguous input data. The study [11] is an example where RNNs are used for the task of stock prediction.

The other prominent set of models falls under the unsupervised training category. In this scenario, the data has no labels, and there is no prior knowledge regarding the results of performing generative tasks [12].

Deep Autoencoders (DAEs) apply unsupervised learning. First described by [13], they are characterized by having input and output layers of the same or similar size, along with two processing modules. The first structure, the encoder, takes the input data and compresses it to a smaller representation containing its essential features. The second structure, the decoder, aims to reconstruct the original input data by upsampling the compressed representation until it reaches the size of the input data.

Restricted Boltzmann Machines (RBMs) are regarded as a specific form of Autoencoder, initially proposed by [14], capable of learning probability distributions. RBMs have been employed to construct Deep Boltzmann Machines (DBMs), as stated by [15]. In [16], Autoencoders are used for segmenting teeth in Intra-Oral 3D Scans.

A particular and very well-known Autoencoder is Transformers, which are models that avoid recurrence and exclusively rely on an attention mechanism to establish global dependencies between input and output [17]. This type of model is used by [18] for the prediction of risk in cardiovascular diseases.

By combining the previous learning types, it creates a new category known as semi-supervised learning. Within this classification fall Generative Adversarial Networks (GANs). GANs comprise two neural models: the generator and the discriminator, operating in an adversarial training paradigm [19]. This architecture aims to understand and replicate a given data distribution. The generative model tries to generate synthetic instances of the input data, while the discriminator assesses these instances to determine their similarity to the input data. This adversarial process provides a probability of authenticity for each instance, distinguishing between input (authentic) and synthetic (generated) data. Through iterative refinement, the generator learns to produce data that increasingly resembles the input data. In [20], the authors implement a GAN model to translate RGB images into IR ones for fire forest monitoring.

In recent years, a trend has emerged within the field of deep learning: the development of hybrid models. These models are considered important for future advancements in DL. They combine two or more distinct architectures, such as CNN-LSTM or Autoencoder-LSTM, [21] and [22].

III. Introduction Methodology to Train Deep Learning Models

In this section, the methodology to obtain accurate DL models is described in depth. The process has been divided into different steps, adding the best practices and an explanation about how they should be applied. It begins with having the dataset raw related to a use case to solve with it. Then, deciding which DL model fits better until the obtention of a set of accurate metrics that avoid generalizes. Finally, ending with an interpretation of the models using other metrics and making a comparison with some baselines.

A. Deciding What Deep Learning Model to Use

The first decision to be taken when solving a DL problem corresponds to choosing a suitable model. Considering what has

been described in Section II, the different DL models that can be applied depend on the nature of the data and the use case to be solved. If we talk about the different use cases, there are also many applications, but all of them can be classified into two main types: classification and regression. Belonging to one group or the other will have consequences, as will be described later, in the metrics used to measure the performance of the model. Classification problems consist of building a model that links the attributes of the instances, typically represented as attribute-value pairs, to the corresponding class labels [4]. Regression problems aim to predict the value of a dependent variable based on one or more independent variables, [4]. Typical classification problems in DL are image classification or disease diagnosis. Within the regression problems group, we have time series prediction and object detection.

Due to the wide range of models and use cases, this decision has multiple solutions. We provide a general vision about how to choose the proper DL model. Table I compiles information on the different models, the type of data used to train them, and the typical use cases to solve.

TABLE I. SUMMARY OF DL MODELS USED, TYPE OF DATA AND APPLICATIONS.

Model type	Nature of data	Use case
MLP	Tabular data	Prediction of a continuous value
CNNs	Image data with spatial relationships	Image Classification, Object Detection, Segmentation
GCNNs	Data represented as graphs or networks	Graph-structured Data, Social Network Analysis
RNNs	Temporal sequences, sequential data	Sequential Data Modelling, Time Series Prediction
Autoencoders	Unlabelled images	Data Compression, Denoising, Anomaly Detection
Transformers	Sequential data, particularly text data	Language Translation, Text Generation
GANs	Often images, but can be applied to other domains	Image Generation, Style Transfer, Data Augmentation
DRL	Sequential decision-making tasks with rewards	Game Playing, Robotics, Autonomous Systems
Hybrid models	Mixed data types or tasks	Multiple types of data/modalities or tasks

B. Splitting the Dataset into Train, Validation, and Test

As has been said before, to obtain a Deep Learning model that performs well, we need to avoid overfitting. Model testing, which consists of checking how a trained model performs with new data never seen before, is the way to prevent it. For this purpose, first, there is a need to split the data into three sets: training, validation, and test.

The training set shows instances of the model so it can be adjusted to generalize its distribution. The validation set measures the accuracy of the model based on the errors obtained. The test set is used to check the capacity of the model to generalize by using data never seen before.

So, the first decision to be taken is the percentages used to create the three sets. A typical criterion is based on the Pareto principle, which states that for many phenomena in nature, about 80% of the consequences are produced by 20% of the causes [23]. In this way, a good choice is to split the training and test by 80% and 20%, respectively. Then, split the training set into 80-20 for training and validation. Anyway, this is not an exact rule, and depending on the size of the original dataset, these percentages could vary. For small datasets, it could be 70-30, and for big datasets, 90-10, as done in [24].

The second decision to take related to the training, validation, and test sets is the strategy to follow during the split. Typically, a randomization is applied to avoid patterns depending on the order of the data. This prevents potential biases by ensuring that there is enough diversity representing the complete dataset in the three subsets. However, depending on the type of data and its organization, randomizing it is not always the best option. It is very important to avoid instances from the test set being introduced during training, which is called data leakage. A typical data leakage case occurs when using frames of videos in a model. If we split the sets of subsequent frames, almost equal frames could be part of the training and test sets. To avoid this, it is better to split the dataset into full sequences.

Finally, a problem could occur in classification problems with imbalanced classes. This could lead to a training or test set where a class has more presence compared to the other. The strategy that is applied to avoid this is called stratification. This will ensure that each subset maintains an identical class distribution.

C. Feature Scaling

Apart from the obvious transformation of data into continuous values, as input data in DL models is always numeric data, feature scaling is a best practice to apply just before starting to train the models. This term is defined in [25] as transforming all features to a uniform range, avoiding the influence of large-scale features on classification models or other feature-dependent processes. Feature scaling also has different benefits. For example, they can accelerate the process of gradient-based optimization algorithms, as in DL models, which leads to faster training times. Also, by avoiding the influence of features over the rest, the model performance tends to be improved. Finally, calculating distances is more accurate after feature scaling, as this process is part of the algorithms used in DL, for example, L1 regularization.

In this sense, there are two main strategies: scaling and normalization. The former transforms features to have a mean of zero and a standard deviation of one, which prevents certain features from dominating others during training. The latter scales features to a range between 0 and 1 or -1 and 1, which is particularly useful for algorithms sensitive to the magnitude of features, such as distance-based algorithms. There are many different strategies to apply scaling and normalization, but the following should be highlighted.

In the case of scaling, most well-known is Min-Max. This strategy rescales the features to a fixed range, typically [0, 1], by subtracting the minimum value from each feature and then dividing by the range (the maximum value minus the minimum value). Eq. (1) formalizes this process:

$$X_{MinMax} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

The main strategies in normalization are Euclidean Norm and Z-Score. Z-score transforms the features to have a mean of 0 and a standard deviation of 1. This strategy is depicted by Eq. (2):

$$Z_{Score}=\frac{X-\mu}{\delta} \quad (2)$$

In the equation, μ is the mean, and δ is the standard deviation.

D. Choosing the Metrics Depends on the Problem to Solve

The way to measure the performance of a DL model is by using metrics. The aim is to obtain the values that ensure that the model is well-trained. As a first step, we must choose which generic metric to use depending on the problem to solve. As we have said before, DL problems can be framed into two big classes: classification and regression.

In classification problems, the main metric is called accuracy. This is defined as the proportion of the samples that the model has predicted correctly among the total number of instances. Eq. (3) describes this metric:

$$Accuracy=\frac{TP+TN}{TP+FN+FP+TN} \quad (3)$$

In the equation above, True Positive (TP) refers to cases where the actual sample class is positive, and the model's prediction is also positive. True Negative (TN) denotes instances where the actual sample class is negative, and the model's prediction is also negative. False Positive (FP) signifies scenarios where the actual sample class is positive, yet the model predicts a negative outcome. False Negative (FN) represents situations where the actual sample class is negative, but the model predicts a positive result. The accuracy measures a percentage in ranges of 1, so the best value is the nearest to 1.

For regression problems, the Mean Squared Error (MSE) is usually used. It is defined in [26] as the mean of the squared differences between the predictions and the actual values. The following Eq. (4) describes this metric:

$$MSE=\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2 \quad (4)$$

In the previous equation, N corresponds to the number of instances of the dataset, yi is the real value, and $\hat{y}i$ is the prediction made by the model. When using the MSE, the aim is to obtain a value as close to 0 as possible.

These metrics will be used as a first step to measure the performance of the model during training, validation, and test, or check if underfitting or overfitting is happening. For a deeper analysis of how models work, we will use other metrics that will be introduced in Section III H.

E. Using k-Fold Validation During the Training Stage

K-fold validation is a typical strategy used when data is scarce. Apart from that, it should be used to validate the models more exhaustively. As DL models are initialized by randomization of their weights, this initialization could benefit from the data split in how the model performs. To mitigate this issue, k-fold cross-validation is employed.

In this strategy, the training set is divided into k smaller subsets or "folds". Subsequently, a loop is executed for each of the k folds. Then, the model is trained using k-1 folds as the training data and validated (evaluated for its performance) using the remaining fold. The numeric measure reported by k-fold cross-validation is the average of the values calculated across the entire loop. Also, a standard deviation is given that will be useful to know if the performance is worse in any of the folds. Considering these factors, a good practice is using k-fold cross-validation with k values set to either 5 or 10. Empirical evidence suggests that these values yield test error rate estimates that achieve an appropriate balance, mitigating common Deep Learning pitfalls as described in [27].

F. Hyperparameter Optimization

Deep Learning models depend on multiple configurable variables that set its behavior and performance. Thus, it is essential to find the best set of hyperparameters to obtain an optimal model. There are different strategies to achieve this task, but we can mention the following: grid search, random search, and Bayesian Optimization. First, one optimizes the hyperparameters by exploring and combining various ranges of possible values to find the best configuration [28]. Random search is another simple and efficient hyperparameter optimization method that randomly samples parameter combinations, offering better performance than grid search in high-dimensional spaces [29]. The use of one or another depends on the model to tune, grid search ensures that all hyperparameter combinations are tested, but needs from a lot of computational resources, so it is better for models where the hyperparameter search space is small. However, random search is more efficient in computational terms, but does not explore all the search space of hyperparameters, so it should be used when the search space is wide. Finally, we have Bayesian Optimization defined by [30] as a probabilistic surrogate model (often a Gaussian process) to represent prior beliefs about the objective function and iteratively updates this model using new data. By balancing exploration and exploitation through acquisition functions, it efficiently selects the next sample points to find the global optimum with fewer evaluations.

G. Avoiding Underfitting and Overfitting

As we have said before, the main aim when training a DL model is avoiding overfitting/underfitting. Apart from that, it is also important that it can perform better than a human doing the specific task for which we are training it. To accomplish this, we will introduce the concept of trade-off between bias and variance [31]. This term addresses the balance between model complexity and its ability to generalize to unseen data (test set). In DL models, this trade-off plays a crucial role in determining model performance and robustness. The bias of a model refers to its ability to capture the underlying patterns in the input data, while variance refers to its sensitivity to small fluctuations during training.

The aim of the bias is accomplished when the metric at the training stage improves how the human performs this task. For example, if we are training a model for the diagnosis of Parkinson's Disease with Magnetic Resonance, we must obtain a better accuracy than that obtained by a physician. A model with low bias may simplify the underlying patterns in the data, leading to underfitting and poor performance on both the training and test sets. Apart from the bias, we use the concept of

variance, which is the difference between the metrics at the training and test stages. A model with high variance may capture noise or irrelevant patterns in the training data, leading to overfitting and poor generalization of unseen data. There is no exact value for this, but having a difference greater than 10% of the value between the training and stages seems too much. In the case of underfitting, the metrics during the training stage could be higher than during the test. When overfitting occurs, the training is lower than the test. In conclusion, the aim is to obtain a high bias and low variance.

To avoid underfitting in DL models, several strategies can be employed. Firstly, increasing model complexity by adding more layers, neurons, or parameters can help capture underlying patterns in the data. Additionally, experimenting with more complex architectures such as CNNs, RNNs, or Transformers tailored to the nature of the data and task can enhance model performance. Another option is training for longer by increasing the number of epochs or iterations, which allows the model to learn intricate patterns for more time.

To prevent overfitting in DL models, several strategies can be employed. Firstly, regularization techniques like dropout and batch normalization. The former consists of temporarily disconnecting several neurons from the network, including all its incoming and outgoing connections [32]. The latter normalizes the inputs of the layers. Another technique is early stopping, where training is stopped when the validation metric stops improving. We can also use data augmentation techniques, such as rotation, translation, scaling, and adding noise, to enhance training diversity. Finally, increasing the size of the training dataset or simplifying the model by reducing layers, neurons, or parameters can also mitigate overfitting.

H. Adding New Metrics for an In-Depth Interpretation of the Model

In Section III D, we talked about the main metrics that could be used for classification and regression problems. These metrics have been used to measure the performance of the model to find a bias-variance trade-off. As a first approach, these metrics are enough, but in some cases, more complex metrics should be used. For example, if the developed model is used in the field of medicine, metrics considering FP and FN should be used. This will let us interpret the models more deeply. Let's consider a model for cancer diagnosis, which is not the same in that it has problems with FN (healthy people diagnosed with cancer) than FP (people with cancer who have been considered as healthy). In the last case, this is a problem that a model should minimize. So, considering all these points, we are introducing more metrics.

In the case of classification problems, metrics like specificity and sensitivity (sometimes called recall) are very useful. Specificity calculates the ratio of true negatives to the sum of true negatives and false positives, while sensitivity performs a similar calculation, changing true negatives for true positives and false positives for false negatives. Finally, precision is computed by dividing the number of true positives by the sum of true positives and false positives. These three equations [Eq. (5) to Eq. (7)] are formalized as follows:

$$\text{Specificity} = \frac{TN}{TP + FN} \quad (5)$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (6)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (7)$$

The interpretation of the metrics above can be done as follows: Specificity is used so the model can distinguish healthy individuals from those without cancer, crucial for minimizing unnecessary treatments or interventions. In the case of sensitivity, the model can identify most cases of cancer, reducing the risk of treating cases of disease and applying an accurate intervention and treatment, minimizing unnecessary stress for patients. Finally, with precision, we ensured to identify cases of cancer and apply an accurate intervention and treatment. Apart from that, there are other more complex metrics like F1-Score, Area Under the Curve (AUC), and Receiver Operating Characteristic (ROC) Curve.

If we talk about regression problems with DL models, some points should be considered. If the use case consists of predicting a particular value, like a time series problem for predicting the value of a company´s stock, the concepts of FN and FP do not apply. Anyway, there are more metrics like MSE, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), or R-squared (also Coefficient of Determination) that can be used similarly. But this is not the only case of regression problems that can be solved with DL models; we also have object detection, audio restoration, and others. In these cases, the concepts of FP and FN can be used. Therefore, there exist metrics for this purpose, for example, Intersection over Union (IoU) and Dice score.

IoU computes the ratio of the overlapping area between the predicted (output image of the model) and ground truth (image used in the output layer during training) regions to the total area covered by both regions [33]. It ranges from 0 to 1, where 0 signifies no overlap and 1 denotes perfect spatial alignment between the regions. So, it is defined as the ratio of the area of intersection between the predicted image and ground truth to the area of their union. Eq. (8) formalizes it mathematically:

$$\text{IoU} = \frac{|Predicted \cap Ground\ Truth|}{|Predicted \cup Ground\ Truth|} \quad (8)$$

In the equation above, |Predicted ∩ Ground Truth| represents the area of intersection between the predicted and ground truth regions. Then, |Predicted ∪ Ground Truth| represents the area of union between the predicted and ground truth regions.

Dice score is a metric based on IoU and calculates the ratio of twice the overlapping area to the sum of the sizes of the predicted and ground truth regions [34]. It also ranges from 0 to 1, having the same interpretation as IoU. Eq. (9) describes it:

$$\text{Dice} = \frac{2 \times |Predicted \cap Ground\ Truth|}{|Predicted| + |Predicted|} \quad (9)$$

Regarding the equation above, the numerator is the same as in IoU. Then, |Predicted| represents the size of the predicted region, and |Ground Truth| represents the size of the ground truth region.

I. Making a Comparison with a Baseline

Using a baseline is essential for evaluating the performance of a Deep Learning model for several reasons. Firstly, it provides a point of reference against which the model's performance can be compared. This is crucial in understanding whether the model is learning meaningful patterns in the data or if its performance is merely a matter of luck. Secondly, baselines help in setting realistic expectations for the model's performance. By comparing against a simple baseline, we can measure how much improvement DL techniques offer over simpler approaches. This helps in assessing whether the additional complexity of applying DL techniques is justified. Moreover, using a baseline can also highlight potential shortcomings or biases in the dataset. If a DL model struggles to outperform a simple baseline, it may indicate issues with the quality or representativeness of the data, prompting further investigation and data preprocessing. Overall, employing a baseline is a fundamental step in the evaluation process of Deep Learning models, providing context and insights that are essential for making informed decisions about model performance and deployment.

IV. Use Cases

In this section, we are exploring some use cases where the methodology explained above has been explained in detail, step-by-step. These examples try to cover most of the situations that could happen during the implementation of a DL model, considering the decisions to take. The first use case is a model that could diagnose breast cancer using thermographic images [35]. In the second use case, a diagnostic model for epilepsy using electroencephalographs [36]. Finally, the third use case solves the problem of small brain tumor detection in Magnetic Resonances [37]. In the following Table II, Table III, and Table IV (one for each use case), we summarize each step of the methodology, adding the decision taken and its explanation.

TABLE II. Methodology Applied to Breast Cancer Diagnosis

Step	Decision	Explanation
3.1.	Classification problem with coloured images solved with CNNs	The dataset consists of thermographic images of healthy people and people with breast cancer.
3.2.	Randomise 80-20 split	The number of images is enough, and the images between patients are very different.
3.3.	Scaling: Min-Max	Typical decision when working with images.
3.4.	Accuracy metrics	In a diagnosed problem, we need to measure the number of hits.
3.5.	K-fold validation with k=5	Usual decision
3.6.	Bias-variance trade-off. Train: 87.67%±6.65, Val: 86.67%±12.47, Test: 89.23%±5.85	Bias is correct as human performance accuracy, assisted by a CAD, is near 83%, (Keyserlingk et al. 2020). The variance is also correct.
3.7.	Other metrics. Specificity: 88%, Sensitivity: 90.53% and Precision: 88.91%	These metrics are around the same values which means the model is very stable
3.8.	Baselines. VGG16: 68%. VGG19: 51.33%, Inception 84%	The proposed model performs better than the baseline

TABLE III. Methodology Applied to Epilepsy Diagnosis

Step	Decision	Explanation
3.1.	Classifying chunks of an EEG using a CNN	The dataset is chunked into timestamps that are modelled as graphs represented in images. The CNN model discriminates between stages of a seizure.
3.2.	80-20 split between patients	As brain states in a person do not change a lot from a particular moment and the following, the split is done considering the EEGs of the patients.
3.3.	Scaling: does not apply	As images are in greyscale, they are in the same ranges.
3.4.	Accuracy metrics	For a classification problem, accuracy should be used.
3.5.	K-fold validation with k=5	Usual decision
3.6.	Bias-variance trade-off. Model 1: 93.6%, 88.2%, 87.2%. Model 2: 91.9%, 86.8%, 81.3%	Bias cannot be measured as there are no official studies about the performance of physicians in distinguishing seizure states. The variance is correct.
3.7.	Other metrics. Specificity: 94%, Sensitivity: 94.1% and Precision: 93.2%	These metrics are around the same values, which means the model is very stable.
3.8.	Baselines: nothing.	There is no baseline for being a very specific use case.

TABLE IV. Methodology Applied to Tumor Detection

Step	Decision	Explanation
3.1.	Small brain tumor detection in MR using a U-Net	The dataset is pairs of MRs of the brain and the mask that isolates the brain tumour in this MR
3.2.	Randomise 80-20 split	The number of MRs is enough and they are very different between them.
3.3.	Scaling: Not needed	The images are in greyscale.
3.4.	Loss metrics: MSE	In object detection, we need to know how the predicted mask resembles the ground truth image that represents the mask.
3.5.	K-fold validation with k=5	Usual decision.
3.6.	Bias-variance trade-off. Train:10% ± 0.5%, Validation: 11.1% ± 0.5%, Test: 14%	Human performance goes from 28 and 44% error (the model performs better), and the variance is not high.
3.7.	Other metrics: Dice score. Train: 96.3% ± 0.8%. Validation: 92% ± 1%. Test: 91.6%.	Dice score considers FP and FN.
3.8.	Baselines. Dice score: 45%	The proposed model performs better than the baseline.

V. Conclusion

This study presents a step-by-step methodology to train DL models accurately. Apart from that, it presents the best practices to do so, and complete information is provided during the process. The manuscript should be taken as an information resource for researchers who are not experts in DL and want to use these techniques in their field, so they can understand the main terms and strategies to apply and why. In case of having a deeper understanding of DL models, this could be used as a guide for best practices.

The methodology is divided into eight steps. First, tries to explain which DL models should be used depending on the nature of the data and the use case to solve. Second, describe how to split training, validation, and test sets considering the distribution of the dataset and its similarities. Third, compile different strategies to apply feature scaling, which is related to the value ranges of the different features of the data. The fourth shows which general metrics should be used depending on whether the problem to solve is regression or classification. The fifth describes the validation strategy called k-fold cross-validation. Sixth, introduces the concept of bias-variance trade-off that shows if there is any underfitting or overfitting. Seventh tries to understand in-depth how the model performs by introducing metrics that consider false positives or false negatives. Eighth, evaluates the proposed model against a baseline, so it can be demonstrated that it is better than what can be found in the scientific literature.

Finally, three different use cases have been compiled where we have provided information for each step of the method, pointing out the decision that has been taken and an explanation about it, if possible.

AUTHOR'S CONTRIBUTION

A.N. and A.G.T.: Conceptualization, formal analysis, experimental design, and methodology. A.N.: Manuscript writing, figures, and tables. All authors reviewed the manuscript. A.G.T. and A.M: Approval of the final version of the manuscript.

REFERENCES

[1] S. J. Russell and P. Norvig, "Artificial intelligence: a modern approach. Malaysia," 2016, Pearson Education Limited London, UK:

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Adv. Neural Inf. Process. Syst., vol. 25, pp. 1097–1105, 2012.

[4] C. Sammut and G. I. Webb, Encyclopedia of machine learning. Springer Science \& Business Media, 2011.

[5] H. Li and A. Baghban, "Insights into the prediction of the liquid density of refrigerant systems by artificial intelligent approaches," Sci. Rep., vol. 14, no. 1, p. 2343, 2024.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.

[7] R. Ehtisham et al., "Computing the characteristics of defects in wooden structures using image processing and CNN," Autom. Constr., vol. 158, p. 105211, 2024.

[8] W. Hu, M. Li, H. Xiao, and L. Guan, "Essential genes identification model based on sequence feature map and graph convolutional neural network," BMC Genomics, vol. 25, no. 1, p. 47, 2024.

[9] J. L. Elman, "Finding structure in time," Cogn. Sci., vol. 14, no. 2, pp. 179–211, 1990.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[11] M. Lu and X. Xu, "TRNN: An efficient time-series recurrent neural network for stock price prediction," Inf. Sci. (N Y)., vol. 657, p. 119951, 2024.

[12] R. Sathya, A. Abraham, and others, "Comparison of supervised and unsupervised learning algorithms for pattern classification," International Journal of Advanced Research in Artificial Intelligence, vol. 2, no. 2, pp. 34–38, 2013.

[13] D. H. Ballard, "Modular learning in neural networks.," in AAAI, 1987, pp. 279–284.

[14] P. Smolensky, "Chapter 6: information processing in dynamical systems: foundations of harmony theory," Parallel distributed processing: explorations in the microstructure of cognition, vol. 1, 1985.

[15] R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines," in Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, D. van Dyk and M. Welling, Eds., in Proceedings of Machine Learning Research, vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 448–455. [Online]. Available: https://proceedings.mlr.press/v5/salakhutdinov09a.html

[16] A. Almalki and L. J. Latecki, "Self-Supervised Learning With Masked Autoencoders for Teeth Segmentation From Intra-Oral 3D Scans," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 7820–7830.

[17] A. Vaswani et al., "Attention is all you need," Adv. Neural Inf. Process. Syst., vol. 30, 2017.

[18] A. U. Rahman, Y. Alsenani, A. Zafar, K. Ullah, K. Rabie, and T. Shongwe, "Enhancing heart disease prediction using a self-attention-based transformer model," Sci. Rep., vol. 14, no. 1, p. 514, 2024.

[19] I. Goodfellow et al., "Generative adversarial nets," Adv. Neural Inf. Process. Syst., vol. 27, 2014.

[20] S. P. H. Boroujeni and A. Razi, "IC-GAN: An Improved Conditional Generative Adversarial Network for RGB-to-IR image translation with applications to forest fire monitoring," Expert Syst. Appl., vol. 238, p. 121962, 2024.

[21] P. zhao, D. Ding, K. Li, Y. Li, and G. Jin, "A tunable diode laser absorption spectroscopy (TDLAS) signal denoising method based on LSTM-DAE," Opt. Commun., vol. 567, p. 130327, Sep. 2024, doi: 10.1016/j.optcom.2024.130327.

[22] F. Zhou et al., "Unified CNN-LSTM for keyhole status prediction in PAW based on spatial-temporal features," Expert Syst. Appl., vol. 237, p. 121425, 2024.

[23] R. Dunford, Q. Su, and E. Tamang, "The pareto principle," 2014.

[24] N. Maray, A. H. Ngu, J. Ni, M. Debnath, and L. Wang, "Transfer Learning on Small Datasets for Improved Fall Detection," Sensors, vol. 23, no. 3, p. 1105, Jan. 2023, doi: 10.3390/s23031105.

[25] D. Protić et al., "Numerical Feature Selection and Hyperbolic Tangent Feature Scaling in Machine Learning-Based Detection of Anomalies in the Computer Network Behavior," Electronics (Basel)., vol. 12, no. 19, p. 4158, Oct. 2023, doi: 10.3390/electronics12194158.

[26] A. Jadon, A. Patil, and S. Jadon, "A Comprehensive Survey of Regression-Based Loss Functions for Time Series Forecasting," 2024, pp. 117–147. doi: 10.1007/978-981-97-3245-6_9.

[27] M. Kuhn and K. Johnson, "Over-Fitting and Model Tuning," in Applied Predictive Modeling, New York, NY: Springer New York, 2013, pp. 61–92. doi: 10.1007/978-1-4614-6849-3_4.

[28] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," Adv. Neural Inf. Process. Syst., vol. 24, 2011.

[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," Journal of machine learning research, vol. 13, no. 2, 2012.

[30] H. J. Kushner, "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise," Journal of Basic Engineering, vol. 86, no. 1, pp. 97–106, Mar. 1964, doi: 10.1115/1.3653121.

[31] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias--variance trade-off," Proceedings of the National Academy of Sciences, vol. 116, no. 32, pp. 15849–15854, 2019.

[32] S. K. Srivastava, S. K. Singh, and J. S. Suri, "Healthcare Text Classification System and its Performance Evaluation: A Source of Better Intelligence by Characterizing Healthcare Text," J. Med. Syst., vol. 42, no. 5, p. 97, May 2018, doi: 10.1007/s10916-018-0941-6.

[33] M. A. Rahman and Y. Wang, "Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation," 2016, pp. 234–244. doi: 10.1007/978-3-319-50835-1_22.

[34] H. Chegraoui et al., "Object Detection Improves Tumour Segmentation in MR Images of Rare Brain Tumours," Cancers (Basel)., vol. 13, no. 23, p. 6113, Dec. 2021, doi: 10.3390/cancers13236113.

[35] A. Nogales, F. Pérez-Lara, and Á. J. García-Tejedor, "Enhancing breast cancer diagnosis with deep learning and evolutionary algorithms: A comparison of approaches using different thermographic imaging treatments," Multimed. Tools Appl., vol. 83, no. 14, pp. 42955–42971, Oct. 2023, doi: 10.1007/s11042-023-17281-x.

[36] A. Nogales, Á. J. García-Tejedor, J. Serrano Vara, and A. Ugalde-Canitrot, "eDeeplepsy: An artificial neural framework to reveal different brain states in children with epileptic spasms," Epilepsy & Behavior, vol. 154, p. 109744, May 2024, doi: 10.1016/j.yebeh.2024.109744.

[37] A. Nogales, C. de la Pinta, Á. J. García-Tejedor, E. García-Barriocanal, D. Guadalupe, and M.-A. Sicilia, "Evaluating the Detection of Small Brain Lesions in Magnetic Resonances Using Deep Learning," 2025. doi: 10.2139/ssrn.5364234.