electronics

MDPI

Article

Nets4Learning: A Web Platform for Designing and Testing ANN/DNN Models

Antonio Mudarra [1], David Valdivia [1], Pietro Ducange [2], Manuel Germán [1], Antonio J. Rivera [1,*] and M. Dolores Pérez-Godoy [1]

[1] Department of Computer Science, Universidad de Jaén, 23071 Jaén, Spain; amachuca@ujaen.es (A.M.); dvv00003@red.ujaen.es (D.V.); mgerman@ujaen.es (M.G.); lperez@ujaen.es (M.D.P.-G.)
[2] Department of Information Engineering, University of Pisa, 56126 Pisa, Italy; pietro.ducange@unipi.it
* Correspondence: arivera@ujaen.es

Abstract: Nowadays, any research discipline is interested in tackling its problems with artificial intelligence and, therefore, is demanding knowledge and frameworks with the aim of developing and using intelligent methods. Within this scenario, neural networks stand out for the important results they have achieved. This paper introduces Nets4Learning, a web platform for designing, training and testing artificial/deep neural network models. The application deals with some of the most popular tasks in the data science field such as tabular classification, regression, image classification and object detection. Nets4Learning has been designed so that any researcher from any discipline can easily develop neural network models without special programming or digital skills. In fact, the user does not have to install anything as the application is publicly available and can be accessed from any device. The site also has manuals, glossaries, etc., and all this code is available on GitHub.

Keywords: web platform; artificial intelligence; artificial neural networks; deep learning; classification; regression; computer vision

Citation: Mudarra, A.; Valdivia, D.; Ducange, P.; Germán, M.; Rivera, A.J.; Pérez-Godoy, M.D. Nets4Learning: A Web Platform for Designing and Testing ANN/DNN Models. *Electronics* **2024**, *13*, 4378. https://doi.org/10.3390/electronics13224378

Academic Editor: Ricardo Martins

Received: 3 October 2024
Revised: 31 October 2024
Accepted: 5 November 2024
Published: 8 November 2024

1. Introduction

Currently, artificial intelligence (AI) is having a huge impact on society [1]. This has been possible due to its impressive successes in many areas, such as industry, economy, energy, medicine, art, etc. Even for people's daily lives, applications where AI has demonstrated a great achievement can be found. Within the AI area, machine learning (ML) [2] is a set of techniques that learn from examples, obtaining a model that extracts and generalizes the knowledge of the domain to which these examples belong to.

Undoubtedly, one of the most important techniques within the field of ML is artificial neural networks (ANNs) [3]. The first generations of ANNs for predictive tasks, e.g., multi layer perceptrons (MLPs) [3], were characterized by having one or a few hidden layers (shallow networks). However, in order to represent complex concepts internally, authors began to discuss the introduction of more hidden layers [4,5]. In this manner, building these deep neural networks (DNNs), the so-called deep learning (DL) paradigm [6] came into its own in the late 2000s. The first DNNs stand out for the results obtained in computer vision (CV) [7] fields such as image classification or object identification, where the typical models used are convolutional neural networks (CNNs) [6].

ANN/DNNs are deeply exploited not only for research scopes but also in several practical applications [8]. Ready-to-use tools for working with these AI models are in high demand, especially for non-expert users. Therefore, it is essential to learn and understand how to design and apply ANN/DNN models to a specific research domain.

In this work, the Nets4Learning web application is presented. With this platform, researchers, practitioners and even non-expert but curious users can design, train and test ANN/DNNs, exploring their topology and functionalities. Specifically, users can deal

with key tasks such as: tabular classification, regression, image classification and object identification.

One of the most important features of Nets4Learning is that a production version is available https://simidat.ujaen.es/n4l/ (accessed on 30 October 2024) for use on any client device such as PCs, tablets and smartphones. Furthermore, a comprehensive user manual is integrated into the platform with tips, usage examples and a glossary. This enables the easy and extensive use of the tools. In summary, researchers from any discipline can develop and train ANN/DNN models for addressing their problems without having special programming skills.

Other software and platforms comparable to Nets4Learning can be categorized in three categories:

- Programming libraries such as Scikit-learn https://scikit-learn.org/stable/ (accessed on 30 October 2024), TensorFlow https://www.tensorflow.org/ (accessed on 30 October 2024) or Keras https://keras.io/ (accessed on 30 October 2024). Obviously, these tools require advanced programming skills, in addition to those specific to the AI tasks and techniques to be worked with. Like any other type of libraries, they need to be downloaded to the computer. It should be noted that these tools allow working with neural networks and other classic ML techniques. In comparison, although Nets4Learning is only focused on neural network techniques, our platform does not require programming skills, nor does it require the user to install any software. In addition, Nets4Learning has many built-in aids for both learning how neural networks work and how to design them. It should also be highlighted that the presented platform has graphical tools for pre-processing data, managing models or displaying results, tasks that can have high learning curves in the mentioned libraries.
- Local frameworks such as RapidMiner https://altair.com/altair-rapidminer (accessed on 30 October 2024), WEKA https://ml.cms.waikato.ac.nz/weka/ (accessed on 30 October 2024) or Orange https://orangedatamining.com/ (accessed on 30 October 2024). This next group of tools already has a graphical interface, so generally, no programming knowledge is required. They usually allow you to work with different types of tasks and techniques, but the support for a wide variety of options usually means that using these tools is quite complicated unless you have significant knowledge in different areas of AI. They need to be installed locally, which can mean the use of a PC with a certain storage capacity, computing costs, etc. Regarding Nets4Learning, although it is possible to work on different tasks, it only deals with neural networks, a widely used technique in the AI paradigm. Furthermore, the platform presented does not require installation and can be used on a wide range of devices (PCs, tablets, smartphones, etc.) without any special software or hardware requirements. Finally, compared to the tools in this group, Nets4Learning stands out for its usability and ease of use.
- Web repositories of pre-trained AI models such as TensorFlow Hub https://www.tensorflow.org/hub (accessed on 30 October 2024), HuggingFace https://huggingface.co/ (accessed on 30 October 2024) or NVIDIA NGC https://catalog.ngc.nvidia.com/ (accessed on 30 October 2024). This group of tools may be the most similar to Nets4Learning as they have access via a web platform as a common factor. However, most AI models of these web repositories need to be downloaded and integrated into the user specific application, which also implies important programming skills and hardware requirements. With Nets4Learning, all these skills are not necessary since it is a web application accessible from any device without specific requirements from the side of the user. It is true that in some platforms, like HuggingFace, the user may directly test some pre-trained model for some specific tasks, but unlike our application, it does not allow the user to design the topology or architecture of the neural network. These types of platforms also lack manuals, glossaries, etc., or other learning resources that are available on Nets4Learning.

In the following, Section 2 describes the fundamentals of AI and ANN/DNNs and details the main functionalities of Nets4Learning. Section 3 shows how Nets4Learning can be used to obtain experimental results in some of the tasks mentioned. A discussion about this software is provided in Section 4, and finally, conclusions and future improvements are outlined in Section 5.

2. Materials and Methods

AI [9] is a broad field that aims to create systems able to mimic human cognitive functions such as learning, problem solving and decision making. Within AI, ML [10] is a sub-area based on learning from experience. Specifically, its methods can analyze large amounts of data (in a dataset), identifying patterns and structures or learning autonomously to improve their performance over time. Finally, with the knowledge extracted, ML algorithms are able to make predictions or decisions.

Well-known methods or techniques in ML [10] include decision trees, support vector machines or k-nearest neighbors, but one of the most important is ANNs. Since their inception, models such as MLPs [3] have obtained and continue to obtain successful results in various applications. These models were characterized by having one or a few hidden layers and were called simple neural networks. Due to the increase in the processing capacity of computing hardware and greater availability of data, more neural networks can be designed, with a high number of layers, neurons, mathematical processing, etc. These networks are called DNNs and give rise to the DL paradigm [6]. This area allows the learning of more complex concepts and continues to develop today with models such as Transformers, LLMs or foundational models in general [11].

The most popular classical tasks where all these methods are applied include classification and regression [10]. In classification, the class to which an example belongs, from a set of predefined classes, is obtained. In regression, a continuous numerical value is predicted. The number of fields in which learning is applied is constantly growing, which implies the appearance of new tasks. For example, in the area of CV [7], two of the best known tasks are image classification and object detection in images. As mentioned, DL became very popular when it achieved impressive results in the field of CV.

2.1. Neural Networks

An ANN is an interconnected set of nodes that emulates the functioning of the human brain. Generally speaking, the structure of an ANN, Figure 1, is composed by an input layer, one or more hidden layers and an output layer of neurons. A neuron is a basic processing unit that receives inputs, processes them with an activation function and generates an output. The connections between neurons are defined by parameters called weights. An activation function implements a mathematical equation that is applied to the weighted sum of inputs, introducing non-linearity into the network. Classical activation functions are `sigmoid` or `tanh` and are widely used in ANNs such as MLP. These topologies become much more complex in DNNs, such as CNNs, with more layers and more neurons, and more complex activation functions such as `ReLU`, `SELU`, etc.

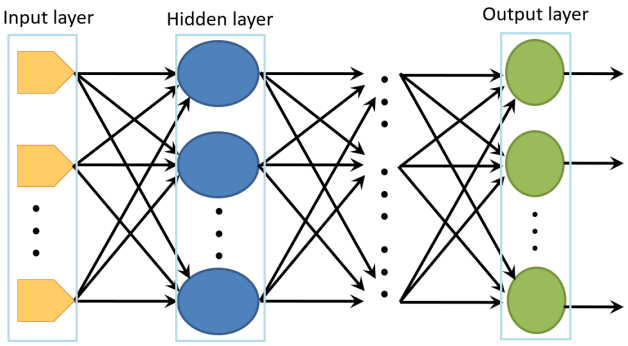

Figure 1. An ANN topology with an input layer, some hidden layers and an output layer.

Another key element of any neural network is its learning algorithm [3]. Given a specific training dataset, the learning algorithm operates from the errors between the actual output values associated with each input training sample and the estimated outputs provided by the ANN/DNN. Thus, an important parameter of this algorithm is the optimizer, a mathematical algorithm used to minimize or maximize the mentioned error. Examples of optimizers are `Adam` or `SGD`. The error is calculated using a loss function, such as mean squared error, mean absolute error or Hubber loss. These errors are then used for updating the weights, and the process is repeated as many times as epochs have been defined.

2.2. Nets4Learning

Nets4Learning is a web platform to develop and test ANNs/DNNs. The main functionalities of the platform are shown in Figure 2 and coincide with the ML tasks discussed above. These can be seen in the main screen of the application, Figure 3. The most important ones, related to the design and development of ANN/DNN models, are highlighted in green in the Figure 2.

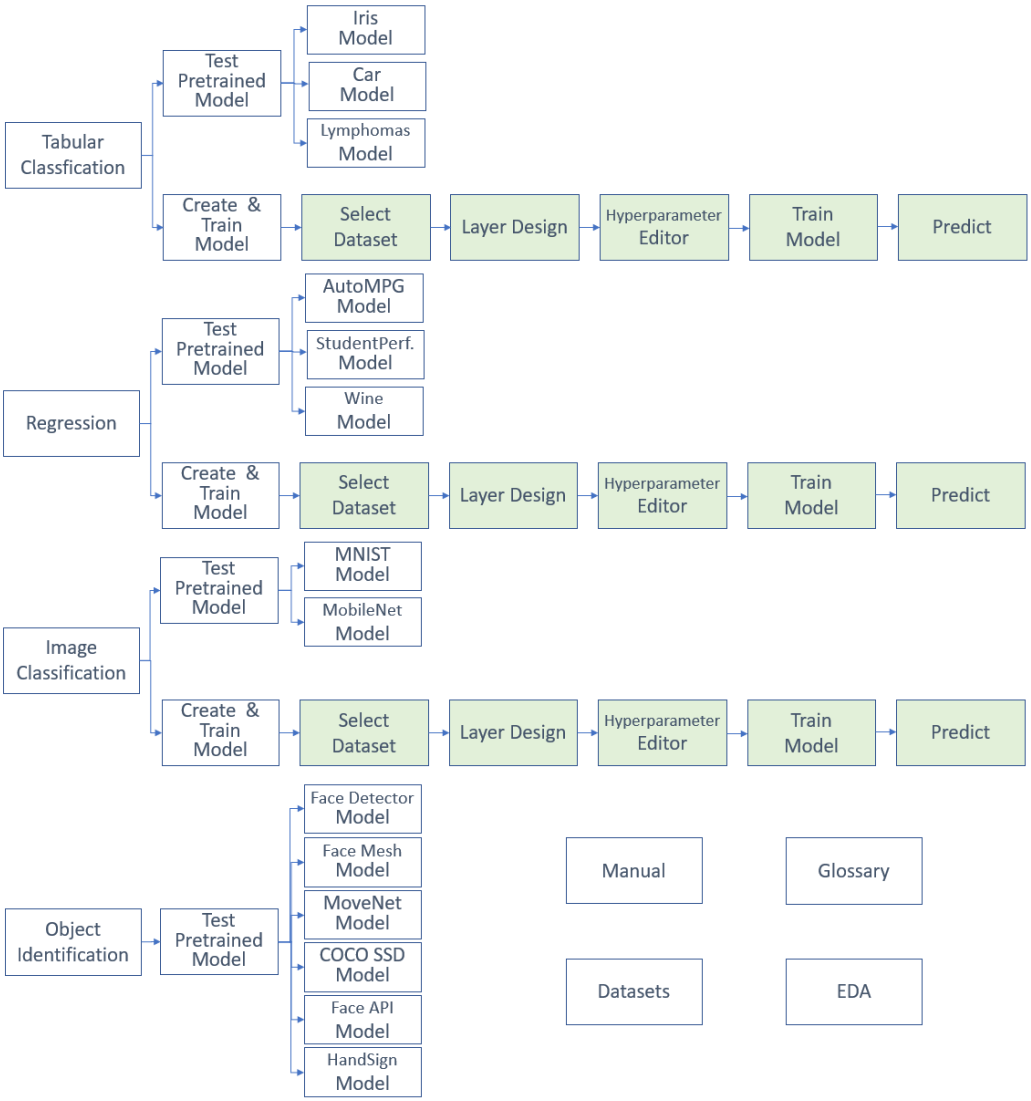

Figure 2. Software operation flows in Nets4Learning. In green, stages involved in the development of a new ANN/DNN model.

For the development of ANN/DNNs with Nets4Learning, a complete set of stages, which cover the whole process, is defined:

1. Select Dataset: The user can select a pre-charged dataset or upload a new one from his/her device. After that, the different characteristics and their values of the chosen dataset can be inspected. Even some pre-processing techniques, such as `standard` or `max-min` scaler, can be applied to the dataset.
2. Layer Design: At this stage, the user can thoroughly define the topology of the network to be designed. In this way, the number of layers of the network, the number of neurons for each layer, their activation function, etc., can be chosen.
3. Hyperparameter Editor: The user can then set the main hyperparameters that define the network training process, such as learning rate, epochs, optimizer, loss function, etc.
4. Train Model: Once the network has been defined, the next step is to train it. The user will obtain information on this process through graphs showing the progress of variables such as loss and accuracy. After that, the user has at his/her disposal the different models he/she has trained so far.
5. Predict: In this step, the accuracy of the trained model can be tested. For this purpose, the user can choose any sample from the test set and obtain a classification result or can even manually enter the values of the desired input features.

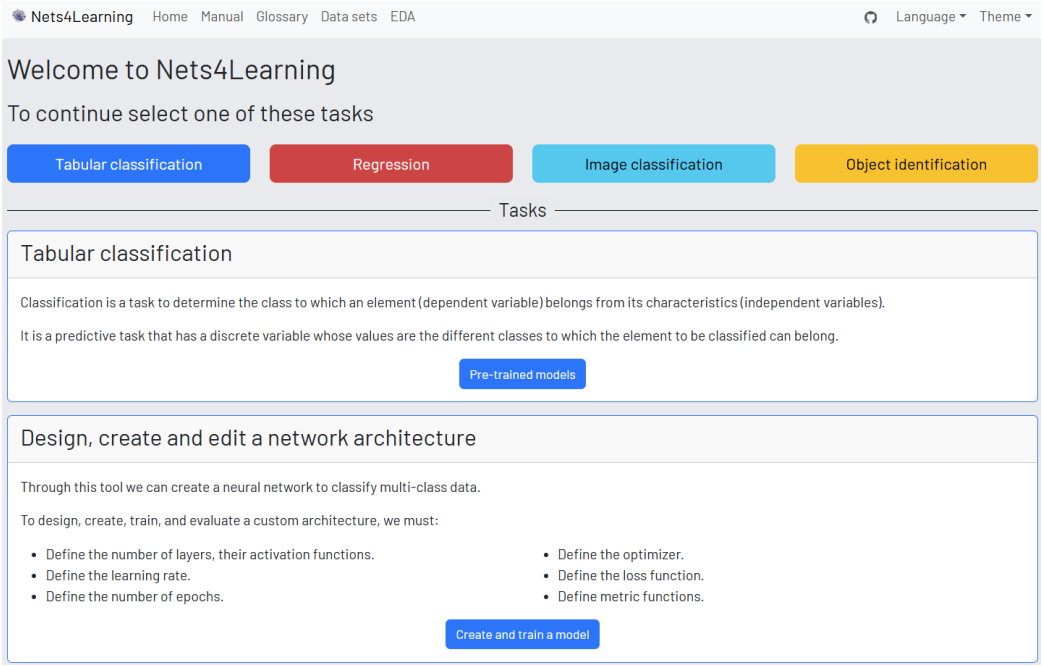

Figure 3. Main screen of Nets4Learning.

As can be observed, researchers from different disciplines can design and train their own ANN/DNNs in order to address their corresponding problems. Additionally, users can test the operation and performance of different pre-trained models belonging to different paradigms. Specifically,

- Tabular classification. The user can test pre-trained models on classical datasets, such as
  - Iris [12]: classification of the Iris plant.
  - Car [13]: evaluation of cars on the basis of their characteristics.
  - Lymphography [14]: classification of lymphoma stages.
- Regression. The user can choose this task and test different models:
  - AutoMPG [15]: A pre-trained model that predicts city-cycle fuel consumption in miles per gallon.
  - Student Performance [16]: A regression model to predict student performance in secondary education.

  - Wine [17]: This ANN model predicts wine preferences by data mining.
- Image classification. User can select one of the pre-trained models and use it to make predictions from predefined, uploaded or drawn images. Nets4Learning includes two pre-trained models for image classification:
  - MNIST https://github.com/tensorflow/tfjs-examples/tree/master/mnist (accessed on 30 October 2024) [5]: A pre-trained CNN model from the TensorFlow Hub repository for the classification of the MNIST dataset, which is composed of images of hand-written digits.
  - MobileNet https://tfhub.dev/google/imagenet/mobilenet_v2_035_128/classification/5 (accessed on 30 October 2024) [18]: a CNN model for mobile vision applications that can distinguish between 1001 categories of images.
- Object identification. The user can choose one of the different pre-trained models. To test all these models, the user can upload an image or can trigger a video stream using a webcam in his/her device. Then, the appropriate parts or regions are identified on the tested images or videos. Specifically, Nets4Learning includes the following pre-trained models:
  - FaceDetector https://tfhub.dev/mediapipe/tfjs-model/face_detection/short/1 (accessed on 30 October 2024) [19]: Detects one or multiple faces (and their keypoints) within images. It is based on a CNN architecture (a single-shot detector (SSD) with a custom encoder).
  - FaceMesh https://tfhub.dev/mediapipe/tfjs-model/facemesh/1/default/1 (accessed on 30 October 2024) [20]: Predicts 3D facial surface landmarks. It is based on a MobileNet network with customized blocks for allowing real-time performance.
  - MoveNet https://tfhub.dev/google/movenet/multipose/lightning/1 (accessed on 30 October 2024) [18]: predicts human joint locations in the image frame. The model architecture consists of a MobileNet with a decoder, followed by a CenterNet for center/key point extraction.
  - COCO SSD https://github.com/tensorflow/tfjs-models/tree/master/coco-ssd (accessed on 30 October 2024) [21]: Identifies 80 classes of common objects, such as person, bicycle, car, etc. It uses a meta-architecture where different models based on CNNs are unified.
  - Face API https://justadudewhohacks.github.io/face-api.js/docs/index.html (accessed on 30 October 2024): Different models (MobileNet [18], MTCNN [22], etc) are used for face detection, face recognition or face expression recognition.
  - Hand Sign Detector https://github.com/tensorflow/tfjs-models/tree/master/hand-pose-detection (accessed on 30 October 2024): Provides different models in order to carry out hand pose detection. From this first stage, American sign language and Spanish Sign language are detected.

The platform also offers a user manual, a glossary, a section to download datasets and another for analyzing them (EDA—exploratory data analysis).

3. Experimentation and Results

In this section, the use of Nets4Learning to experiment with tabular classification and object detection tasks is illustrated with examples.

3.1. Tabular Classification

As mentioned, the main steps to perform tabular classification are shown in Figure 2. In this manner, after selecting `Create and train a model` for tabular classification in the main screen of the application, Figure 3, the user has to choose between using one of the datasets provided by the platform or uploading his own dataset in `csv` format. It is worth noting that if the dataset is uploaded by the user, the last column of the `csv` file must correspond to the output class. In this case, the hepatitis dataset from the well-known

UCI repository https://archive.ics.uci.edu/dataset/571/hcv+data (accessed on 30 October 2024) has been processed.

Figure 4 shows the data set that has already been loaded, providing information on the data type of each feature, which can be modified by the user. It also offers the possibility to perform transformations on the input data, in particular `MinMaxScaler` or `StandardScaler`.

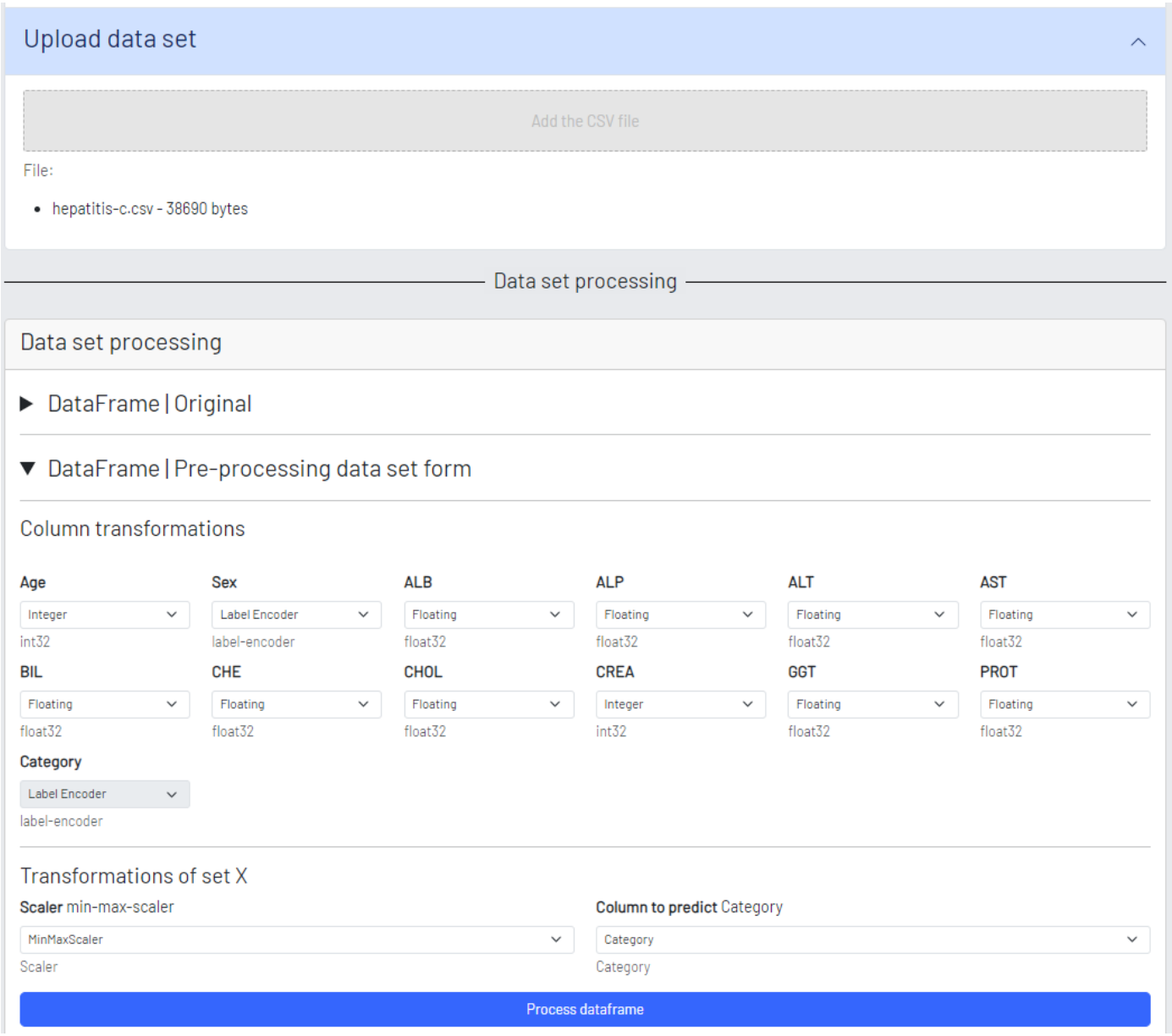

Figure 4. Tabular Classification: Data pre-processing.

After the pre-processing stage, the user may design the structure of the ANN and its training process by using the interface depicted in Figure 5. The interface shows that practically any parameter required to obtain a new ANN model, and described in Section 2, can be configured, such as layout of the layers, number of neurons, activation functions, etc., and different training hyperparameters that include learning rate, training epochs, optimizer, loss function, etc. All these values, set by default, can be modified by the user to define the characteristics of the ANN to be trained.

Then, the user trains the new model and can see the evolution of this process and how different errors, previously chosen by the user, are adjusted, Figure 6. Finally, predictions can be made, as shown in Figure 7, entering the value of each input feature using the edit

boxes and then asking for a prediction. The output of the model, in terms of the probabilities associated with each class, is displayed in a bar graph at the bottom of the interface.

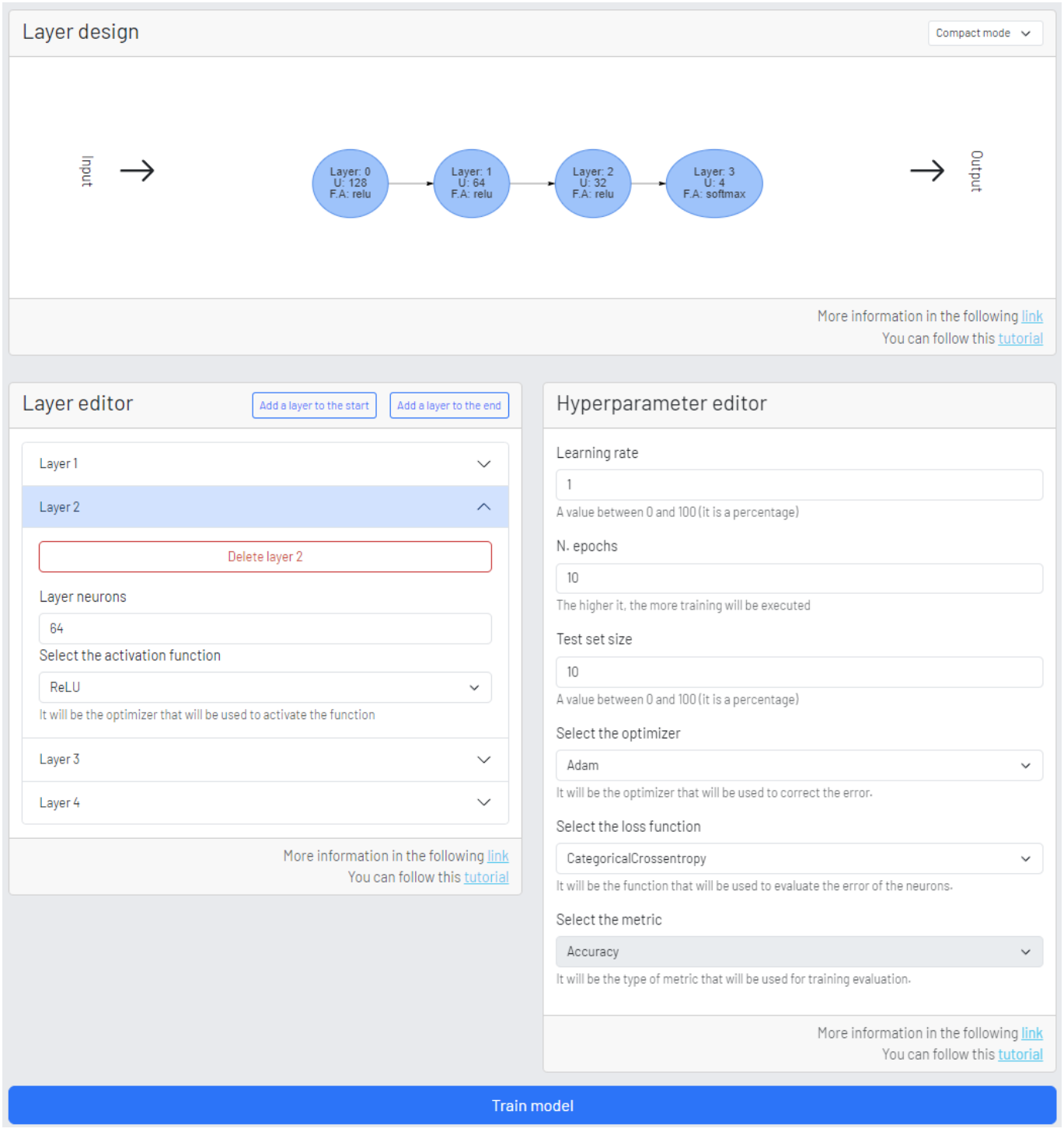

Figure 5. Tabular Classification: Designing an ANN.

Figure 6. Tabular Classification: Training an ANN.

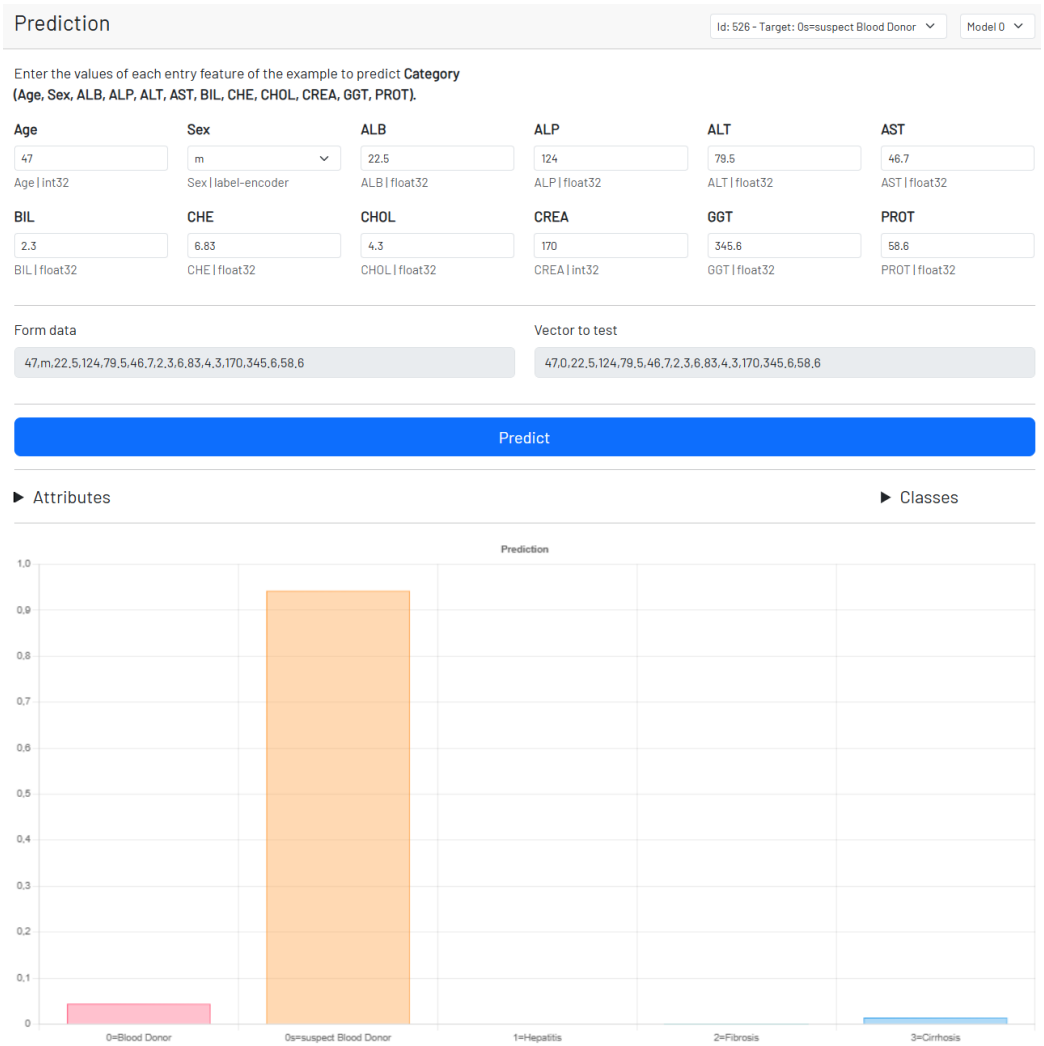

Figure 7. Tabular Classification: Prediction using the trained model.

3.2. Object Identification

To test an object identification model in images, the first step is to select this task in the main screen of the application, Figure 3. Then, the application offers the user different pre-trained models for the identification of objects inside an image. Once one of these models has been selected, the user can upload an image or can trigger a video stream using a webcam in his/her device in order to test it.

In Figure 8, an operating example of the COCO SSD model is shown. In this case, the detected objects are identified with bounding boxes together with the corresponding label name and level of confidence.

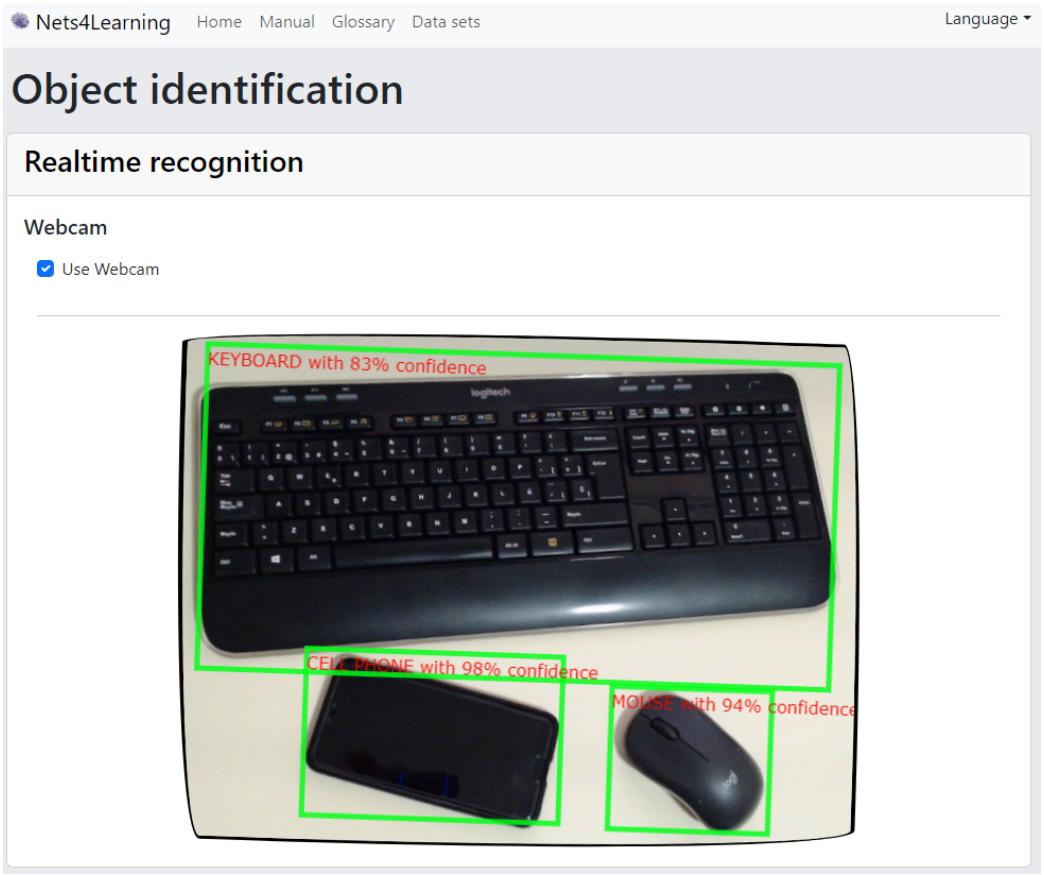

Figure 8. Object identification with COCO SSD model.

4. Discussion

Currently, the research in various scientific disciplines is heavily reliant on ANN/DNN models [2,8]. The widespread adoption of these techniques is driven by their consistently positive outcomes across diverse fields. Consequently, there is a significant interest in employing, designing and training such complex models.

The initial step towards this goal is to acquire knowledge of these technologies. However, the design of ANN/DNN models is not a simple task as it requires understanding numerous hyperparameters related to the network itself (such as layers, activation functions and weights) as well as its training process (including learning rate, optimization function, loss function and metric functions).

Furthermore, the practical implementation of ANN/DNN models by most dedicated frameworks requires users to have computing skills, such as knowledge of programming languages and familiarity with development tools. The last requirement can be a significant barrier to accessing these technologies. In order to mitigate these shortcomings, web platforms are a useful tool to facilitate the access to different machine learning resources [23,24].

To address these challenges, Nets4Learning has been developed as a publicly accessible web platform (and also as an Android APP in the Play Store) that facilitates the complete ANN/DNN model design and training process. Among its characteristics, the following are particularly noteworthy: Nets4Learning and its code are freely available for any user and device; the user interface is designed to be intuitive even for people without digital competences; it is supported by a user manual and a glossary of terms related to the field, available in both the English and Spanish languages; etc.

So far, some of the most significant ANN/DNN models have been included in Nets4Learning to address the tasks of tabular classification (such as ad hoc trained MLP models [3]), image classification [5,18] and object identification [18–21]. Moreover, the software architecture is ready to include new models

In summary, with the aid of this tool, researchers or even interested users from any domain can easily design and train their own ANN/DNN models using tabular data relevant to their specific research problems. In addition, they can evaluate the performance of pre-trained models on common computer vision tasks. Due to its outstanding characteristics, Nets4Learning has been awarded in the 5th Competition for the development of apps based on artificial intelligence techniques organized by the Spanish Association for Artificial Intelligence (AEPIA).

Nets4Learning also has the potential to popularize and disseminate scientific knowledge, emphasizing the significant impact it can achieve. This has been evidenced in programs of the University of Jaén such as Campus Gem https://gem-esp.eu/ (accessed on 30 October 2024) (an educational project co-funded by the European Union), the Researchers' Night (another European Union project) or an exhibition in the prestigious Science Park of Granada https://www.parqueciencias.com/ (accessed on 30 October 2024) called "Artificial Intelligence. An exhibition about people, data and control".

As mentioned, the Nets4Learning architecture is ready to incorporate new models. In a future work, it is planned to incorporate them progressively.

5. Conclusions

In this work, we have introduced Nest4Learning, a web application that provides readily available services for practicing with ANNs and DNNs in a simple and accessible way, even without prior knowledge of programming or advanced mathematics. The application features an intuitive graphical interface that facilitates the learning process and allows users to experiment with different models and datasets.

Nets4Learning also offers a set of user manuals, datasets, models and practical examples. The tool is free and open source, which makes it accessible to a wide audience and can contribute to scientific development in any field through the use of AI.

As mentioned, Nets4Learning is designed so that new models can be easily added. As future improvements, new state-of-the-art datasets and models will be added, especially for computer vision tasks.

Author Contributions: Conceptualization, A.J.R. and M.D.P.-G.; methodology, A.J.R. and M.D.P.-G.; software, D.V., A.M. and M.G.; validation, D.V., A.M. and M.G.; writing—original draft preparation, A.J.R., M.D.P.-G. and P.D.; writing—review and editing, A.J.R., M.D.P.-G. and P.D.; supervision, A.J.R., M.D.P.-G. and P.D.; funding acquisition, A.J.R., M.D.P.-G. and P.D. All authors have read and agreed to the published version of the manuscript.

Funding: The research carried out in this study is part of the project "Advances in the development of trustworthy AI models to contribute to the adoption and use of responsible AI in healthcare (TAIH)" with code PID2023-149511OB-I00, funded by the Spanish Ministry of Science, Innovation and Universities. The contribution of Pietro Ducange has been supported by the Italian Ministry of University and Research (MUR) in the framework of the FoReLab and CrossLab projects (Departments of Excellence).

Data Availability Statement: The permanent link to the production version of Nets4Learning is https://simidat.ujaen.es/n4l/. The current code version is v1.0. A permanent link to the repository for this version of the code can be found at https://github.com/SIMIDAT/nets4learning. The legal code license is MIT. Some of the software code languages, tools, and services used are JavaScript, TensorFlow, SciKitJS, DanfoJS, React.js, etc. Support email for questions: amachuca@ujaen.es.

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

References

1. Priyadarshini, R.; Mehra, R.; Sehgal, A.; Singh, P. *Artificial Intelligence: Applications and Innovations*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2022.
2. Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Comput. Sci.* **2021**, *2*, 160. [CrossRef] [PubMed]
3. Bishop, C. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1996.
4. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
5. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
6. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
7. Szeliski, R. *Computer Vision Algorithms and Applications*; Springer: London, UK; New York, NY, USA, 2011.
8. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
9. Campesato, O. *Artificial Intelligence, Machine Learning, and Deep Learning*; Mercury Learning and Information: Herndon, VA, USA, 2020.
10. Maimon, O.; Rokach, L. *Data Mining and Knowledge Discovery Handbook*; Springer: Berlin/Heidelberg, Germany, 2010.
11. Bommasani, R.; Hudson, D.A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M.S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. On the Opportunities and Risks of Foundation Models. *arXiv* **2022**, arXiv:2108.07258.
12. Unwin, A.; Kleinman, K. The iris data set: In search of the source of virginica. *Significance* **2021**, *18*, 26–29. [CrossRef]
13. Bohanec, M.; Rajkovi, V. Knowledge acquisition and explanation for multi-attribute decision making. In Proceedings of the 8th International Workshop on Expert Systems and Their Applications, Washington, DC, USA, 1–2 September 1988; pp. 1–20.
14. Zwitter, M.; Soklic, M. Lymphography. UCI Machine Learning Repository. 1988. Available online: https://archive.ics.uci.edu/dataset/63/lymphography (accessed on 30 October 2024).
15. Quinlan, J.R. Combining Instance-Based and Model-Based Learning. In Proceedings of the International Conference on Machine Learning, Amherst, MA, USA, 27–29 June 1993.
16. Cortez, P.; Silva, A.M.G. Using data mining to predict secondary school student performance. In Proceedings of the 5th Annual Future Business Technology Conference, Porto, Portugal, 9–11 April 2008.
17. Cortez, P.; Cerdeira, A.L.; Almeida, F.; Matos, T.; Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.* **2009**, *47*, 547–553. [CrossRef]
18. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
19. Bazarevsky, V.; Kartynnik, Y.; Vakunov, A.; Raveendran, K.; Grundmann, M. Blazeface: Sub-millisecond neural face detection on mobile gpus. *arXiv* **2019**, arXiv:1907.05047.
20. Kartynnik, Y.; Ablavatski, A.; Grishchenko, I.; Grundmann, M. Real-time facial surface geometry from monocular video on mobile GPUs. *arXiv* **2019**, arXiv:1907.06724.
21. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3296–3297.
22. Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Process. Lett.* **2016**, *23*, 1499–1503. [CrossRef]
23. Figueiras, E.; Olivieri, D.N.; Paredes, A.; Michinel, H. QMwebJS—An Open Source Software Tool to Visualize and Share Time-Evolving Three-Dimensional Wavefunctions. *Mathematics* **2020**, *8*, 430. [CrossRef]
24. Jusevičius, V.; Paulavičius, R. Web-Based Tool for Algebraic Modeling and Mathematical Optimization. *Mathematics* **2021**, *9*, 751. [CrossRef]