Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 22 Issue 1 Version 1.0 Year 2022
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Neural Network Design using a Virtual Reality Platform

By Luigi Bibbò & Francesco Carlo Morabito
DIIES University

Abstract- The evolution of Deep Learning (DL), a subset of machine learning, has made their use very effective in many artificial intelligence (AI) fields. In parallel Virtual Reality is going wide in many applications thanks to the proliferation of cameras in mobile devices and improved processing efficiency. Data visualization in deep learning is a fundamental element for which it can benefit from the advantages offered by the visualization of the VR for the development of the models. In addition, the researchers can widely use the editing of images and videos in the machine learning process to design a convolutional network suitable for image recognition. In this study, we want to demonstrate the usefulness of this approach in collecting data within virtual reality to train and optimize a convolutional neural network used to recognize human activities (HAR).

Keywords: machine learning, deep learning, neural nets, visualization, virtual reality.

GJCST-D Classification: F.1.1

NEURALNETWORKDESIGNUSINGAVIRTUALREALITYPLATFORM

Neural Network Design using a Virtual Reality Platform

Luigi Bibbò [α] & Francesco Carlo Morabito [σ]

Abstract- The evolution of Deep Learning (DL), a subset of machine learning, has made their use very effective in many artificial intelligence (AI) fields. In parallel Virtual Reality is going wide in many applications thanks to the proliferation of cameras in mobile devices and improved processing efficiency. Data visualization in deep learning is a fundamental element for which it can benefit from the advantages offered by the visualization of the VR for the development of the models. In addition, the researchers can widely use the editing of images and videos in the machine learning process to design a convolutional network suitable for image recognition. In this study, we want to demonstrate the usefulness of this approach in collecting data within virtual reality to train and optimize a convolutional neural network used to recognize human activities (HAR).

Keywords: machine learning, deep learning, neural nets, visualization, virtual reality.

I. Introduction

Machine Learning (ML) indicates a research area within Artificial Intelligence needs to create systems capable of autonomous learning without being specifically programmed to carry out this task [1]. This step introduces a new programming vision, in which the goal is no longer to feed input to receive an output from the machine but to allow it to make decisions autonomously.

Deep Learning (DL) is a machine learning technique used to build Artificial Intelligence (AI) systems [2]. It is a technology that creates learning models on several levels. Deep learning systems allow a representation of information at various levels in a hierarchical way and build models that exploit large amounts of visual media data. The learning of data takes place through the use of statistical calculation algorithms. The inspiring principle was to reproduce a system of reasoning that is biologically inspired by the human brain, must be able to behave by simulating the functioning of neurons. The human neuron represents the computational engine that is the basis of deep learning that takes place in Artificial Neural Networks (ANN). A neural network reproduces all those processes in the brain during the learning phase and subsequent recognition phase. We design the artificial neural networks (ANN) to perform complex analysis of large amounts of data by passing it through multiple layers of neurons. The peculiar characteristic of neural networks is learning and adapting to their environment. Another property is to build connectivity models based on the approximation of the error. These elements make identifying similar patterns in the test data [3].

A neural network requires a training phase and an inference phase to operate correctly. During the training phase, the number of neurons and the levels that will comprise the neural network are defined and exposed to the labeled training data. Finally, the network will learn which data is correct and discard the wrong one. Once the network has been trained, it is possible to proceed to the inference phase in which the network itself is called to evaluate new images proposed to it. These steps are part of the neural network design process, which includes:

1) *Determination of the function that the network must be able to perform*:
    a) *Classification*, this function involves sorting images into various classes and grouping them according to common properties, allowing you to determine whether a specific type of object is present in an image or not;
    b) *Detection and localization*, this function allows you to identify the position and size of an object by identifying the characteristics of the image;
    c) Segmentation, used to identify which pixels of an image belong to the corresponding objects, allows determining the boundaries and areas of the objects in the images.
2) *Choice of framework:* we must make this choice based on the applications to be created, the libraries available, or the developers' competence. In addition, there are numerous accessible frameworks such as, e.g., TensorFlow of Google, Caffe2 of Facebook e Open Vino of Intel, and Pytorch: an open-source solution that is now part of Facebook.
3) *Loading training data*: this phase and the training are crucial for the proper functioning of the neural networks. While being favored by the ability to generate features as they learn automatically, deep networks still require large amounts of training data to develop a properly functioning model. The amount of data needed depends on the complexity of the domain we are trying to approximate. The choice of the size of the dataset is closely related to the selection of the number of neurons in the neural network. The total complexity is generally not known before starting the training. For this reason, the

Author α: DIIES University “MEDITERRANEA” Reggio Calabria, Italy.
e-mail: luigi.bibbo@unirc.it
Author σ: DICEAM University “MEDITERRANEA” Reggio Calabria, Italy.

neural network training process is iterative. At the end of the training, the network's performance is analyzed: the results of this analysis provide information on whether the available data is sufficient. The data for training can be found already labeled on public archives or purchased. Alternatively, it is necessary to generate a faked data set and label them. Each type of application to be developed requires its own set of data, complete with all the characteristics used for a correct evaluation.

4) *Network training and validation*: training is the phase in which a neural network must acquire the knowledge to interpret specific input data. In this phase, it is necessary to provide examples in input/output pairs through which the network must learn to predict the expected output. The network is a mathematical model whose output is regulated by the weights of the signals received by each neuron. During the training phase, these weights are progressively calibrated so that more and more, the network's output in the face of a specific input approaches the desired one. This phase is implemented by comparing the initial classification with what is suggested by learning. We use an algorithm called Backpropagation is used to train neural networks. It compares the result obtained from a network with the output you want to have. We use the difference between the two results to change the weights of the connections between the network layers starting from the output layer. Then we proceed backward by modifying the weights of the hidden layers and finally those of the input layers. To do this, we use a function called cost appropriate to the problem we want to solve. A part of the data prepared is used for the training phase while the data for the test and validation phase are kept separate. Since pre-trained networks are available, we can use them to classify new objects using the transfer learning technique.
5) The process of evaluating new images using a neural network to make decisions is called inference. This step can collect additional test data used as training data for future iterations.

There is a wide variety of deep neural networks (DNN). Convolutional neural networks (CNN) or Deep convolutional neural networks (DCNN) are the types most commonly used to identify patterns in images and video. The name "convolutional neural network" indicates that it employs a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers [4]. They have evolved from traditional artificial neural networks, applying the connectivity pattern between neurons that resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted visual field region known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visible area. CNN's are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks; each neuron in one layer is connected to all neurons in the next layer. Deep convolutional neural networks mainly focus on object detection, image classification, and recommendation systems and are sometimes used for natural language processing. In the absence of CNN to identify the objects in the images, it was necessary to resort to manual feature extraction methods, making mistakes and taking a considerable amount of time. CNN's are characterized by their ability to optimize filters (or Kernel) through machine learning, differentiating themselves from traditional algorithms in which filters must be manually sized.

A feature common to all architectures of deep neural networks, including CNN's, is their high complexity due to many internal parameters. For this reason, numerous techniques have been developed to facilitate the design and understanding of the internal workings of a network [5]. The lack of interpretability and transparency of neural networks and the absence of explanations on the decisions taken can compromise the confidence in their applicability. Moreover, they are endorsed by the large datasets required to train most deep learning models. Therefore, the possibility of understanding the model's functioning allows one to overcome these problems. Finally, we can understand how deep learning models make decisions and what representations they have learned through deep learning visualization. In this way, we regain confidence in the model [6].

II. Data Visualization

Visualization of data represents another crucial phase in the design of neural networks. Designers want to visualize deep learning to understand how deep learning models make decisions and what representations they have learned to make the model as reliable as possible [7]. This notion of general understanding of the model is called interpretability or explicability.

If we consider the training process of a DNN, its visualization facilitates this delicate design phase. Therefore, we need to optimize parameters to minimize the loss function through gradient descent when training neural networks. To ensure that the loss decreases over time, we need to monitor the entire training cycle and test the losses over the "epochs."

Many visualization techniques are limited to showing whether a model is improving during its iterations, in the training process, rather than how it is training and why it makes certain decisions. For example, Martin Becker and al. developed a tool based on *Gradient Descent Analysis* in deep neural network (DNN) training [8], representing a new interactive visualization tool for exploring DNN learning processes. They used a method based on analyzing a new type of training error curve on three levels of detail. Usually, the comparison method between their training error curves is used to evaluate the learning processes resulting from different hyperparameter settings. With this approach, the authors associate three levels of detail to the three stages of Shneiderman's Visual Information Seeking Mantra. In the first stage, the traditional training error curve is the focus of attention. In the zoom and filter stage, we can explore the variation of the training error along each of the descent directions considered during the gradient descent-based DNN training. The third stage allows for an in-depth exploration of methods for adaptive control of the step width along a given descent direction.

Other techniques that focus visualization on the learning process rather than the magnitude of the error are Deep Tracker [9] and Tensor view [10].

Deep Tracker is a visual analysis system designed to analyze CNN training processes' dynamics and identify unusual patterns within the numerous training logs. The authors associated a series of small hierarchical multiples with a hierarchical structure index mechanism to allow you to explore the entire training process from different levels of detail. They also suggest a new cube-style visualization to reveal the complex correlations across multiple types of heterogeneous training data.

Tensor View is a tool that uses Paraview and Matplotlib to study neural networks built on the TensorFlow framework. It allows visualizing the evolution of the neural network parameters, such as the weights of the filters and gradients, providing direct observation of how the neurons update during training. This tool uses visualization to inform researchers about their hyperparameters; to use quantitative information to eliminate unnecessary neurons, accelerating the training processes.

These two tools are very complex, so expert researchers can only use them. Other visualization techniques aim to provide the right amount of information on the evolution of the training process, facilitating the understanding of even less experienced researchers.

Michelle Peters and al. [11] developed, for non-experts, a technique that intuitively visualizes the training process of a neural network to facilitate the understanding of the decisions of a neural network. They use the *dimension reduction method* Uniform Manifold Approximation and Projection (UMAP) [12] to visualize neuronal activity. They generate a visualization with three plots: 1) - a 2D plot of the test data with the input images, 2) - a 2D plot of the training data, 3) – a line plot of the accuracy of both the testing and training data. They also create a video, generating a frame for every epoch, showing how the neural network improves during the training phase. This approach demonstrates how an ongoing training video offers more information than a static view at the end of the process and which features are helpful for non-experts to view, highlighting that the insights drawn from video analysis can be beneficial for network design.

Various visualization techniques have been incorporated into interactive tools to aid in the DNN design process.

Bock et al. [13] designed a tool for 3D representation of deep learning algorithms for experts and non-experts with an interactive user interface that allows visualiz

Pezzotti and al. [14] developed a progressive visual analysis technique called *Deepeyes*. Multiple linked views are associated with analyzing a network during training and showing how it changes over time. In addition, the system supports researchers in identifying problems, such as unnecessary filters or layers and information not captured by the network.

Kahng et al. [15] developed *ActiVis*, an interactive visualization system for large-scale deep neural networks, to help users understand how a model derives its predictions. Users can better diagnose discrepancies with their neural network by integrating multiple coordinated views: such as an overview of the model architecture computation graph, of how the deep learning neurons, which together make up the network, are activated by user-specified instances or subsets of instances. Finally, it provides a graphic representation of the model architecture; hence the user can perform a localized inspection of activations on each model level.

A classification of CNN viewing techniques distinguishes them into two categories: feature display and attribution [16]. With the feature visualization technique, we study how the network operates to generate an example; attribution studies which part of the example is responsible for attributing the network in a certain way.

One of the methods for feature visualization is activation maximization [17] which can be applied at different levels of a CNN. This technique highlights that a CNN tends to build its understanding of an image hierarchically on many levels. For example, Nguyen and al. have developed a system based on multifaceted feature visualization (MFV) [18], uncovering the different types of features learned by each neuron in deep neural networks. Previous activation maximization techniques constructed images without regard for the multiple facets of a neuron; instead, this method explicitly shows each neuron's various aspects by producing a synthetic visualization of each of the types of images that activate a neuron. The visualization of the faceted features thus provides a more accurate and more complete description of the role of each neuron.

To attribution methods, Chattopadhyay and al. [19] have developed a methodology that starts from the visualization of the neural network's architecture as a Structural Causal Model (SCM) and calculates each feature's causal effect on the output. Their approach differs from other related works to structural learning [20]. The goal is to discern the causal structure in data provided; their goal is to identify the causal influence on the output of a learned function.

Sattarzadeh and al. [21] propose a model based on visualization maps from multiple levels using an attribution-based input sampling technique and aggregating them to achieve a satisfactory and complete explanation. The methodology adopted to interpret the layers of CNN results in a four-step method of description. In the first three stages, information extracted from multiple levels of CNN is represented in the accompanying display maps. These maps are then merged into a single module to form an explanation map in the last step. The proposed solution allows overcoming some problems encountered in models that use input sampling techniques. These methods have shown excellent fidelity in rationally inferring model predictions. However, they exhibit instability indices since their output depends on sampling randomness (RISE) or random initialization to optimize a perturbation mask. Finally, their algorithm called Semantic Input Sampling for Explanation (SISI) replaces the randomized RISE input sampling technique with an attribution-based sampling technique. Finally, it uses feature maps derived from multiple levels.

The Deep Replay open-source Python package was designed to allow you to visualize how a Deep Learning model in Keras is performing or learning at each iteration/epoch.

Virtual reality (VR), thanks to its peculiar characteristics, represents a valid alternative to the visualization of conventional data. VR technology establishes a new form of human-computer interaction, arousing in the user a new type of experience linked to the concept of "presence" [22]. Presence is the sensation of being physically and spatially placed in an environment. With virtual reality, you can realize this feeling of being in another world, the virtual one. VR can generate experience and support knowledge acquisition thanks to this new communication process. However, the sense of presence developed by VR depends significantly on the characteristics of the technology used. It is associated with the perceptual illusion that the user has to interact with the remote environment, such as if it were present. In the interaction with the VR environment, a perceptual illusion is created in the user. The stimulation of the senses produces cognitive and emotional models consistent with the experimenting environment. When in VR, a user will inevitably compare the appearance of virtual objects with real-world objects and judge the level of congruence. The other element that favors VR over other visualization techniques is the level of "immersion"[23]. The greater immersion allows you to improve the design business, perform tasks more efficiently and with greater understanding. Immersion in virtual reality is the perception of being physically present in a non-physical world. Perception is created by surrounding the user of the VR system with images, sounds, or other stimuli that provide an evocative environment. Finally, the user can move freely and explore places and spaces through the movements of the head and eyes and interact with objects by grabbing and dragging them. In addition, movements from the physical world are transferred to the virtual world with great precision. This aspect is precisely exploited to simulate physical actions. Finally, workflows and activities can be transferred to the virtual model to verify the correct interpretation, improving the system's accuracy.

These characteristics make virtual reality a suitable environment for simulations and e-learning purposes as they can stimulate users' creativity and accelerate the learning process [24]. This technology is widely applied in various fields ranging from entertainment design to education and rehabilitation and psychological therapies [25]. VR technology makes it possible to improve the rehabilitation-psychological program compared to traditional techniques by creating a more natural process, reproducing the characteristics of home living environments. Other elements that characterize this technique are represented by the possibility of dynamically adjusting the difficulty of the exercises concerning the skills acquired and stimulating the patient's multisensory skills. In the context of training, it has proved to be very effective compared to traditional techniques as they can keep the attention high and focus the topics better [26].

Immersion and interactivity are the elements that suggest the choice of VR for CNN as through them; the user can better focus and understand how it works.

The following sections illustrate the main characteristics of CNN and virtual reality technologies.

III. Convolutional Neural Network

From an architectural point of view, a convolutional neural network is a multi-layered feedforward neural network composed of an input layer, hidden layers, and an output layer. The hidden layers are typically convolutional, followed by activation and pooling layers. This sequential design allows convolutional neural networks to learn hierarchical features. Convolutional neural networks (CNN) process data through many layers of artificial neurons. This human brain (CNN) process is constituted by a set of different layers that act as extractors of the features of the input images and a fully connected terminal network that acts as a classifier. It has proved to be an effective solution for image recognition [27]. They are built to analyze images included within certain data sets and classify objects in images within them. CNN is a network composed of several convolutional layers [28]. Each processing layer comprises a convolutional filter, an activation function (ReLu), a pooling function, and a fully connected layer. At the end of each processing step, an input is generated for the next layer. In the convolutional operations, the trained filter set is convolved with input images to extract the specific feature to create the feature map, which becomes the input for the next filter. Finally, the design of a CNN network requires a training period followed by a test phase. During the training phase, the images are labeled and transferred to the subsequent layers to allow the structure to convert from the representation level of the original input to a higher level and more abstract representation to constitute the reference feature maps with which the network must compare the output feature maps. Once the training and test phase of the network has been completed, we will determine the survey's accuracy level.

Each layer comprises three levels: Convolution, ReLu, and Pooling.

– Convolutional Level (CONV) is the main level of the network. Its objective is to identify patterns. They are multiple, designed to identify features present in the initial image. Each layer learns to extract specific parts of the photos placed at its entrance. Multiple layers in cascade combine the features of the previous layers with higher programmed extraction levels.
– Rectified Linear Unit (ReLu) Level is placed after the convolutional level and can cancel negative values obtained in the previous classes.
– Pool level allows identifying if the study characteristic is present in the previous story. The pooling layer obtains images with a particular resolution at the input and returns the same number of pictures with fewer pixels.

The result of the convolution operations is the production of feature maps obtained with the help of filters that are matrices containing proper values for finding specific characteristics in the input images. At the end of the sequence of convolutional layers, there is then the fully connected level (FC) which aims to establish the identifying classes obtained in the previous levels according to a certain probability.

Each category represents a possible answer that the system will most likely choose.

During the recognition phase, the network performs a classification operation to identify which class the input image belongs to, identifying the one with the highest probability.

The values of the filters are initially chosen randomly. They are subsequently improved at each iteration of the training phase.

We estimate that the model's predictions are plausible; in practice, we measure the discrepancy between actual and predicted values. The error is subsequently processed using the stochastic gradient technique. It is a cyclical technique consisting of two phases:
* backpropagation;
* updating the gradient value.

After propagating the forward phase during training, the outputs are produced to determine the prediction error compared with the expected ones. This error is used to calculate the gradient of the loss function. The backward propagation phase then sends the error through the network layers and updates the weights using the stochastic gradient descent to improve the network's performance on the activity it is trying to learn [29].

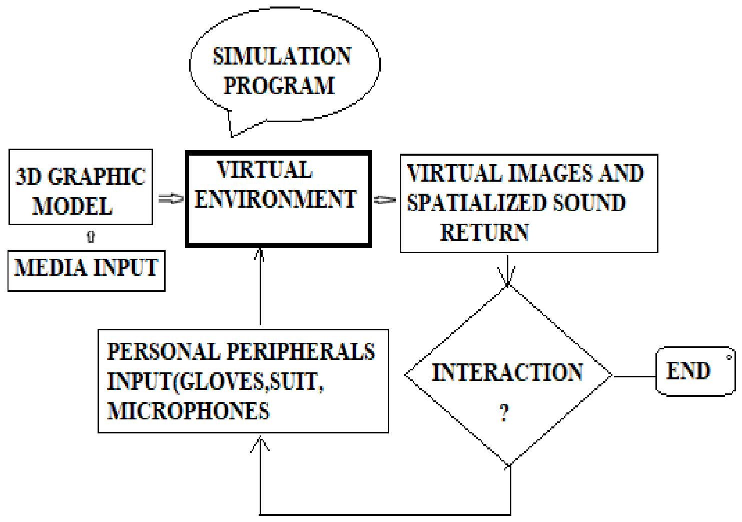
IV. Virtual Reality

Virtual reality technology refers to a computer-generated virtual world [30]. To immerse yourself in this multidimensional world, the user must wear a helmet containing a display that incorporates a sensor to track the movement and position of the wearer (HMD - Head Mounted Display). The sensors detect the position and movement data, which are used to update the image of the virtual world. In this way, the user is projected into this virtual world and can interact with the objects present in it, for example, simulating carrying out usual activities or driving a car or running. By manipulating three variables (space, time, and interaction) and the availability of a graphic interface, it is possible to create a dimension characterized by a strong sense of reality, whereby the subject believes he is in that world and can interact with it.

The virtual environment must have the following characteristics:

- Perception of really being in that world. The use of special equipment amplifies this feeling: software capable of reproducing 3D environments, a virtual reality viewer, integrated audio systems that offer surround support.
- The possibility of interacting with movements of the body, head, and limbs increases the feeling of taking possession of that dimension. For example, cyber-gloves, virtual limbs, joypads, etc., allow the user to touch, move, manipulate, or make virtual objects as if they were real.

The relationship between presence and immersion gives cognition in virtual reality. The first term refers to the level of psychological realism that a subject experiences from interacting with the virtual world. From the wise point of view, the second term means the ability of the environment to involve the senses of the subject, isolating it from the stimuli of the real environment [31].


SIMULATION PROGRAM
3D GRAPHIC MODEL
MEDIA INPUT
VIRTUAL ENVIRONMENT
VIRTUAL IMAGES AND SPATIALIZED SOUND RETURN
PERSONAL PERIPHERALS INPUT(GLOVES,SUIT, MICROPHONES
INTERACTION ?
END

Functional diagram of a VR system

The functionality of a virtual reality system requires different tools: modeling programs with which objects for the virtual world are created; moreover, to generate dynamics in real-time, simulation systems that follow natural laws are needed to move objects in the virtual world. For example, robust programs for viewing 3D images and spatialized sounds in real-time are also used. In addition, traditional systems used to communicate with a computer are no longer suitable for the multidimensional virtual world.

It was necessary to create new interfaces to interact with the computer. They are essentially based on two components:

- Sensors for controlling the virtual world;
- Effectors to provide feedback to the user.

These interface mechanisms are found in apparel, head-mounted displays (HMDs), and 3D sound systems.

Researchers have been developed methods to dynamically measure human movement in real-time through clothing equipped with sensors that track the full range of actions of specific activities carried out by the wearer. The movement is recorded, digitized, and then sent to the computer, which displays them on the screen, replicating all the actions performed in real-time.

This clothing consists of gloves and suits. Gloves are interactive devices, similar to a typical glove worn on the hand, facilitating tactile sensing and refined motion control. Tactile sensing involves simulating the sense of touch and includes the ability to sense pressure, linear force, temperature, and surface structure. Through the control of the fine movement achieved with sensors, the actions of the user's hands and fingers are detected. These movements are then translated into signals that can be used by a virtual hand to allow the user to interact normally with the objects in the virtual environment. The suit wraps the body with directional sensors to transmit electrical signals corresponding to the wearer's movements to a computer. These signals are then converted into digital form, and the resulting data are displayed in the virtual environment. We often use gloves in conjunction with a position and orientation sensor that allows the computer to determine spatial coordinates. They can also be used in conjunction with suits and associated with HMD.

Finally, the viewers called HMD are devices that allow the wearer to immerse themselves in virtual reality. They include a head-mounted stereoscopic display and provide separate images for each eye, a stereo sound, and an ahead motion tracking sensor.

We can distinguish three types of virtual reality:

- Immersive Virtual Reality (RVI)), characterized by the intense sensation of immersion in virtual space thanks to the interaction with the virtual objects present.
- Non-Immersive Virtual Reality, the simulated environment does not stimulate the sense of immersion; it is perceived as the real one.
- Augmented Reality (AR) enriches the objects in the real world with perceptual information produced by the computer through multiple sensory modalities (visual, auditory, tactile).

Below is a brief overview of the state of the art

V. Background

This section reports significant experiences developed based on neural networks combined with virtual reality. Through the analysis of facial expressions, it is possible to understand the emotional state. Neural networks coupled with virtual reality represent a valid platform for analyzing human behavior and effectively treating individuals with autistic disorders. Virtual reality (VR) offers distinct advantages over conventional visualization approaches. Applications range from the entertainment industry to the rehabilitation field.

In this scenario, one of the first applications was that of Belič [32], who developed a methodology in which the two technologies are used to model complex polycrystalline materials. The neural network is used for generating the grain of the material. It makes the manipulation of the grain easier and the representation of the same very compact. It represents the preliminary phase for realizing the polycrystalline material model for virtual reality. The models perform various system optimization or control activities by simulating reality. To make a realistic model of the observed material, the shape of the grains represented by the neural network must come as close as possible to the observed material. The grains are first generated, then, based on the properties of the observed material, the grain shape optimization process is employed to get closer to the observed sample. With the VR approach, information is obtained that is not accessible with traditional techniques (characterization and analysis techniques). The VR must have the model accompanied by information on the mechanical and electrical properties and on the shape of the grains. The use of virtual reality made it possible to obtain additional information such as life expectancy, the diffusion process, or the cracking of the material. VR is used to predict any anomalies produced in the material; it allows you to view the discrepancies detected concerning the expected model. Finally, the virtual environment corrected the model to provide better results. The ultimate goal is to obtain the virtual roughness of the grain as close as possible to the desired one.

An interesting experience is developed by Nino et al. [33] on a VR mobile system based on Neural Network to an IMU gesture controller to simulate the feeling of embodiment [34] and presence. The application aims to demonstrate that combining a sound and motion controller with haptic feedback can improve immersion on mobile interfaces. The interface recognizes the user's gesture (vocal sound and movement) and displays it during execution. The visualization is managed through the gesture controller to improve presence and embodiment. To provide the user with a smooth display of the gesture produced, the authors have created a series of reference gestures that represent the best possible execution of that gesture. To make the set of these "ideal" gestures, a motion capture system was used to record the movement and voice produced by a shintaido expert while performing each gesture [35]. Each ideal gesture is played back for the user while performing the corresponding gesture. Through the dynamic control of the speed of reproduction of the gesture, the user features a degree of control over the avatar's movement. When the user has executed his movement, the captured gesture is classified, and the appropriate response is triggered. Finally, the authors used a multilayer neural network (ANN) for speech and motion recognition. The network architecture and weights have been imported into the Unity3D platform. Without wishing to be a replacement solution for a dedicated tracking system, it can represent a low-cost method of controlling the gestural execution of a subject.

In the field of hand gesture recognition, there is an application developed by Fu and Yu [36] in which they use a neural network for the classification of gestures and, to evaluate the accuracy of recognition in real-time, they build a Unity scene using the data collected by the IMU. Hand gestures are an alternative to other natural methods of Communication. The possibility of recognizing the movement and the pose of the hands allows the observer to understand the intentions of who performs them. Hand gesture recognition is used in many applications such as sign language recognition, sign recognition for controlling robots, and augmented reality. The authors used the IMU as a data input source and structured the application into the following phases:

– Collecting training and testing data;
– Extracting features from IMU inputs;
– Building neural network-based classifier models;
– Building test visualization in Unity.

VR Arduino is used as the data input source. It contains an IMU that provides 9 degrees of freedom sensor readings, including gyroscopes, accelerometers, and magnetometers. It is connected to the PC via USB using Teensy.

The sensor data is recorded on the PC as a training data set and used directly in real-time tests.

The user can hold the Arduino chip in his hand and perform any hand gesture patterns.

A 3D LSTM convolution was used as a neural network to acquire better gestures and obtain high recognition accuracy.

To determine which features work best with the model, they experimented with several combinations of quaternion and raw sensor inputs. The best performing characteristics were the combination of quaternion, gyroscope, and accelerometer data. IMU test data from VRduino are transferred directly into Unity via serial port to visualize the classification.

The tests found that the classifier can predict the user's gesture with reasonable accuracy in real-time.

Sait and Raza [37] presented a prototype based on an innovative hybrid technology in which are present VR and ANN models to build a VE environment in which users can meet and chat together and at the same time feel like they are in a real environment. From a conceptual point of view, the model starts from real-world situations and then, through the IVR and ANN model, creates a virtual world in which the participants are the managers of the model, using the Internet as a medium to connect the IVR model. The model components are thus: Immersible Virtual Reality, Artificial Neural Networks, Internet, and Database.

For the need to have a system design for a realistic visualization, the authors used the IVR. It is the technology model which allows people to immerse themselves in the artificial environment.

ANN is the architecture used to simulate the image and express the user's emotion in the VE. Users' voices, videos, and images are input to train the ANN. Those data are also used for security purposes to check the users' identities.

The Internet connects people spread over different points into a single point of contact on the web. The ANN network is trained on the Internet with a sample of users. The Internet allows scalability for a large number of users and ensures reliable transport of a large amount of multimedia data.

The database must contain a large number of quality video samples and possess a large number of facial recognition expressions to store the video and audio data of the participants.

The model requires two implementation steps. The first is for the interface, and the user inserts his data into the model, followed by the loading of their image, voice. These data will be stored in the audiovisual database to the interface to configure the user with the model. Later, when the user enters the system, the interface will check their identity with the stored data. Finally, the interface must be designed using 3D graphics software [38].

In the second phase, the IVR will provide the chosen environment to those users participating in the network, and the ANN will provide simulated images of the users.

As noted above, there is a growing need for non-experts to understand how deep learning works. For their complexity, neural networks are seen as black boxes. New techniques are experimented with to make the understanding of their functionality as accessible as possible.

In this context, Meissler et al. [39] have developed a technique for visualizing convolutional networks in virtual reality. Their solution is mainly aimed at new designers and aims to provide them with a basic understanding of the functionality of networks. The solutions currently available are essentially usable either by developers of deep learning systems or by those who already have detailed knowledge of deep learning processes or by those who generally have an interest in interactive visualization but are not attracted by the representation capabilities of virtual reality.

Their solution stems from using the advantages of virtual reality in visualization compared to traditional 2D or 3D tools available on a standard desktop screen. Their approach explores the graphic effects produced by the immersion feature. The visualization is intended to illustrate the structure and functionality of the network. For example, the user within the Unity platform can select different inputs, modify the functional structure by inserting or removing a layer and obtain their classification by CNN.

The network model used is LeNet5 [40], whose structure is as follows:

- 2 Convolutional layers.
- 3 Fully connected layers.
- 2 Average pooling layers.
- Tanh is the activation function for the hidden layer.
- SoftMax is the activation function for the output layer.
- Cross-entropy as cost function.
- Gradient descent as optimizer.
- 60000 trainable parameters.

The model was defined in Keras [41] using PYTHON.

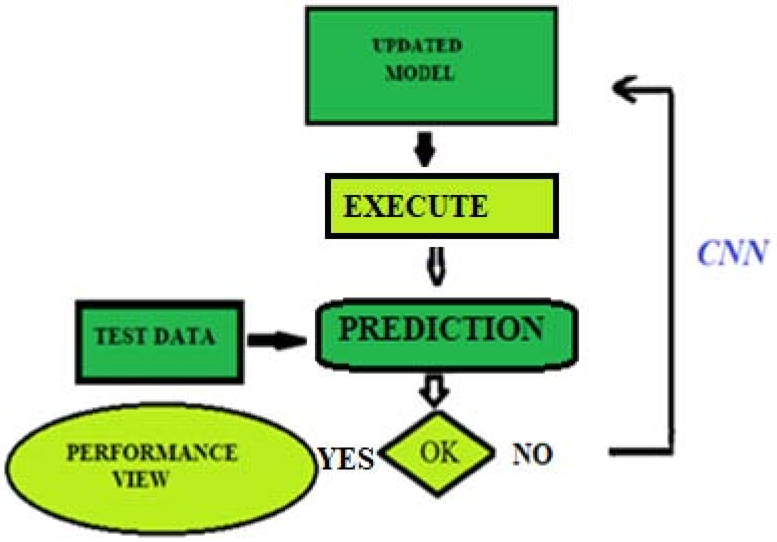
The convolutional and pooling layers were displayed as boxes whose dimensions were based on the layer's size. This type of representation should allow the user to see the structure of the model and the dimensional change of the data. Feature maps of each convolutional and pooling layer are rendered 2D images with matplotlib using the gray colormap.

The experiments carried out on the participants who did not know the topic confirmed the model's validity. Some of them declared that the application was completely self-explanatory. The experiments wanted to demonstrate that virtual reality compared to traditional means of learning, does not distract people but, on the contrary, allows them to focus more on the subject and even longer due to the involvement produced by the virtual environment.

VI. Methodology

Our solution is based on a VR platform where users can develop deep convolutional neural network models for image classification. This type of approach allows verifying the components of a network and becomes a good training tool for professionals who want to design deep neural networks.

In our work, we want to show the opportunity to use the Unity platform (2019) to create a virtual reality environment. We believe it offers some advantages not found in various visualization systems in 2D and 3D desktop screens. Most of the existing platforms do not provide, for inexperienced users, simple interfaces through which it is possible to vary the network architecture and adjust the model parameters. For example, the VR platform allows the user to define the network architecture by specifying the sequence of the layers. The user can interactively verify its operation and decide based on the results obtained by modifying it. We have used this environment to verify the functioning and possibly improve the prediction level of a convolution neural network to recognize human activities (HAR). In addition, the user can select input data and have them classified by CNN. In addition, we previously created a network to identify human activities (HAR) with the STMicroelectronics AI STM32 module [42].



Inference model

a) CNN model

Convolutional Neural Networks (CNN) represents an artificial neural network architecture that has found wide application in identifying, with a certain probability, the content of the image provided by a computer. From a functional point of view, convolutional neural networks simulate how the human brain processes. Convolutional neural networks are built to analyze images included within certain data sets and classify objects in images within them. Each class of things must have a specific network previously trained for the reconstitution of that class. Finally, they process the data through multiple layers placed between the input and the output.

In our study, we have used a CNN to recognize the activities carried out by people daily, such as downstairs, upstairs, sitting, standing, walking. It is mainly used as a supervised learning algorithm and can classify images from a sample dataset. It is based on the principle of the convolution filter applied to the pixel matrix that composes the image. Convolutional layers produce small transformations on images. Each convolution transforms a section of the image into a single value. The Kernel applied sequentially to the embodiment can highlight the fundamental characteristics of the image and, in this way, produce essential data for the classification of the image itself. Finally, we measure the model's performance through the accuracy and the loss.

A Keras model allows building the CNN network, created with the STMicroelectronics AI STM32 and trained on the public dataset Wireless Sensor Data Mining (WISDM). **The Keras DL model used is Shahnawax/HAR-CNN-Keras**. The code for the model definition is in the following listing:

CNN model

```
def cnnModel():
model = Sequential()
# adding the first convolutionial layer with 32 filters and 5 by 5 kernal size, using the rectifier as the activation function
model.add(Conv2D(numFilters,(kernalSize1,kernalSize1),input_shape=(numOfRows,numOfColumns,1),activation='relu'))
# adding a maxpooling layer
model.add(MaxPooling2D(pool_size=(poolingWindowSz,poolingWindowSz),padding='valid'))
# adding a dropout layer for the regularization and avoiding over fitting
 model.add(Dropout(dropOutRatio))
# flattening the output in order to apply the fully connected layer
```

```
 model.add(Flatten())
# adding first fully connected layer with 256 outputs
 model.add(Dense(numNeuronsFCL1, activation='relu'))
#adding second fully connected layer 128 outputs
model.add(Dense(numNueronsFCL2, activation='relu'))
# adding softmax layer for the classification
model.add(Dense(numClasses, activation='softmax'))
```

Fig. 1 shows the network architecture graph, while figures 2 and 3 show, for example, the information relating to the selected layers Conv2D and Dense obtained by clicking directly on the
chart.

This information includes:

the type of the layer;

the name of the layer, whether the layer is trainable;

what the data type is.

For Convolutional layers:

the number of filters;

the kernel size;

the strides;

padding;

data format and dilation rate.

The activation function, whether bias is used, and how the kernels and (if applied) biases are initialized.

Each layer has its own unique set of characteristics.

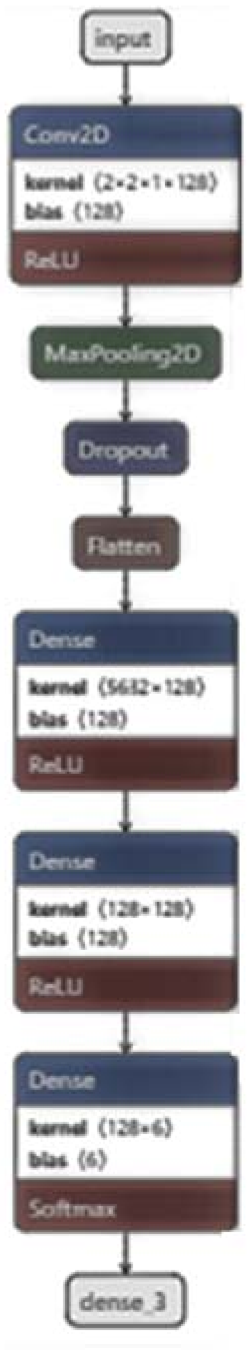
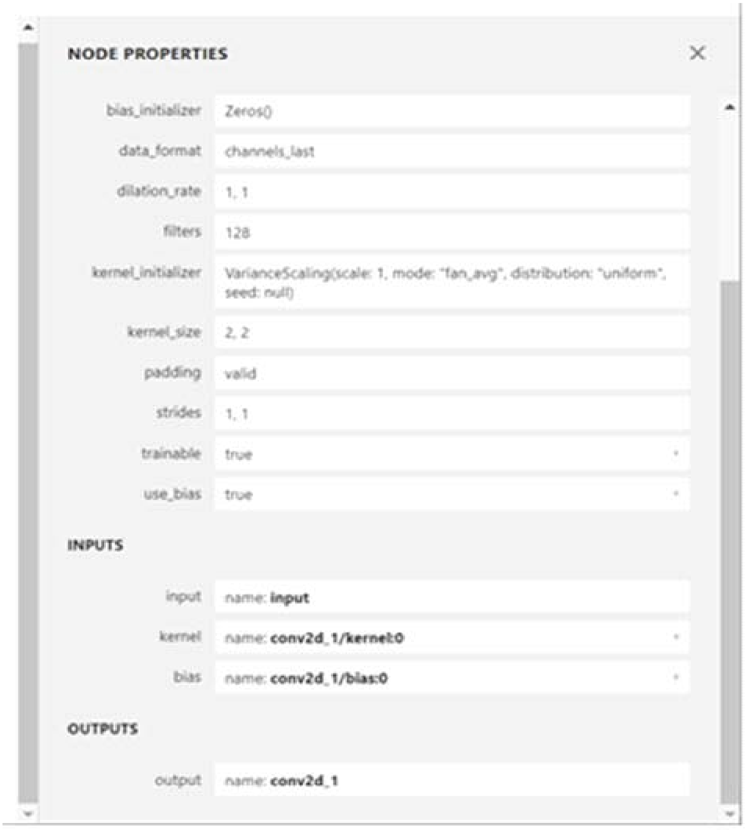


Fig. 1: CNN architecture



NODE PROPERTIES

bias_initializer	Zeros()
data_format	channels_last
dilation_rate	1, 1
filters	128
kernel_initializer	VarianceScaling(scale: 1, mode: "fan_avg", distribution: "uniform", seed: null)
kernel_size	2, 2
padding	valid
strides	1, 1
trainable	true
use_bias	true

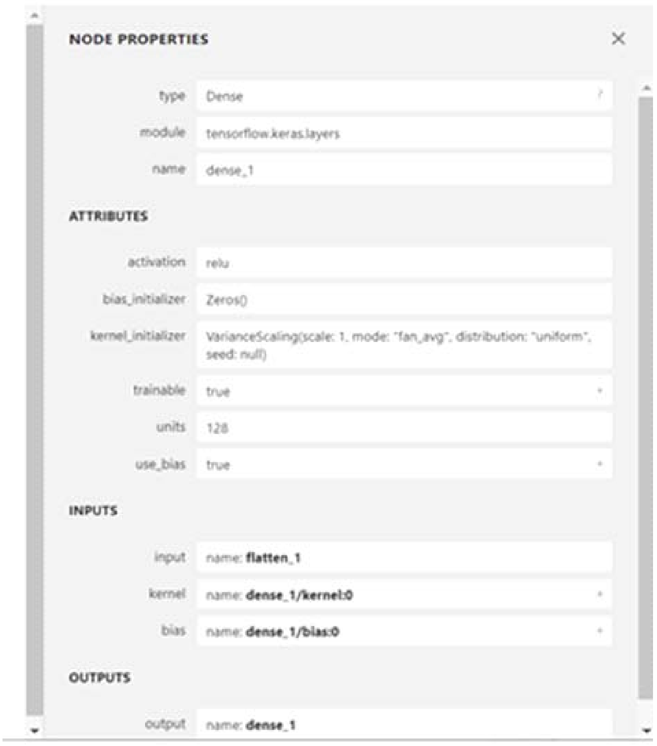
INPUTS

input	name: **input**
kernel	name: **conv2d_1/kernel:0**
bias	name: **conv2d_1/bias:0**

OUTPUTS

output	name: **conv2d_1**

Fig. 2: Conv2D



NODE PROPERTIES

type	Dense
module	tensorflow.keras.layers
name	dense_1

ATTRIBUTES

activation	relu
bias_initializer	Zeros()
kernel_initializer	VarianceScaling(scale: 1, mode: "fan_avg", distribution: "uniform", seed: null)
trainable	true
units	128
use_bias	true

INPUTS

input	name: **flatten_1**
kernel	name: **dense_1/kernel:0**
bias	name: **dense_1/bias:0**

OUTPUTS

output	name: **dense_1**

Fig. 3: Dense1

b) Environment

Within Unity, we used the Barracuda package developed by Unity Labs to develop our application. It is a cross-platform neural network inference engine. Pre-trained neural networks can be imported and run in Unity. In addition, it supports neural networks trained with PyTorch, TensorFlow, Keras, and Caffe.

We executed the following steps:

- Add the onnx file to the project. It behaves like a normal resource;
- Load the model from the asset;
- Create the inference engine (the worker).;
- Run the model and get the results.

To transfer the network, it is necessary to transform its format, as Barracuda is based on the open-source data exchange format ONNX.

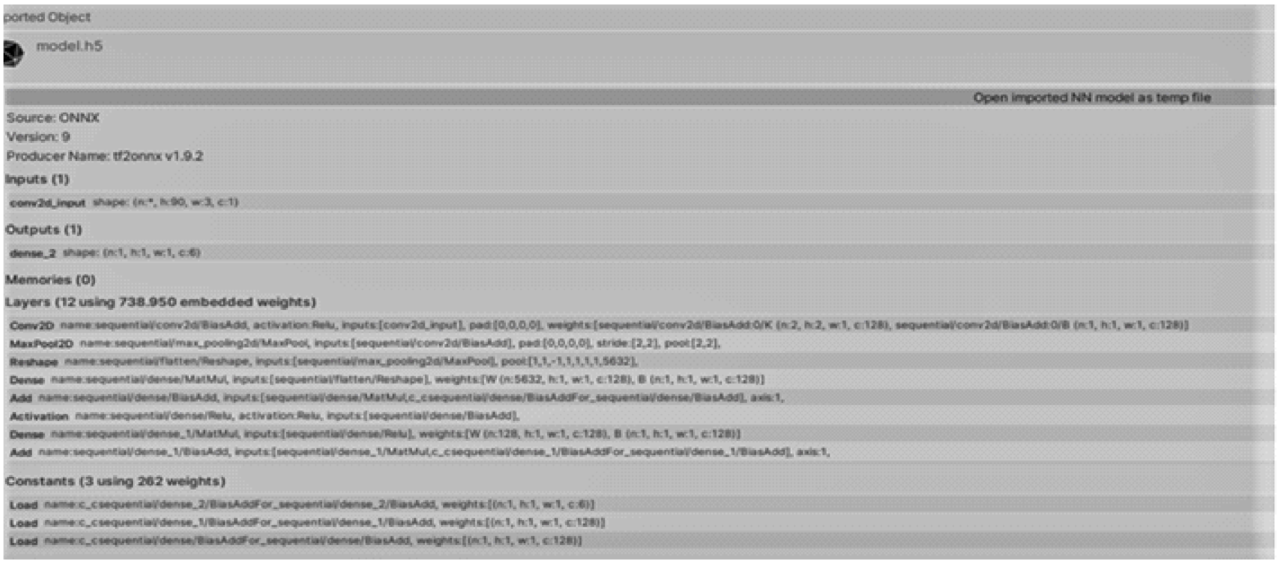
ONNX, which stands for Open Neural Network Exchange, is an open standard that allows you to transfer machine learning models from different frameworks to ONNX. This interoperability enables you to move between various machine learning frameworks quickly. In addition, ONNX supports all popular machine learning frameworks, including Keras, TensorFlow, PyTorch, and XGBoost.

ONNX allows you to have a standard graphical representation for various frameworks. The ONNX graph represents the model through different computational nodes and can be visualized using appropriate representation tools.

To perform the conversion, we used a script. An extract of which is represented by the following code:

```
# network
net = 
# convert model to ONNX
onnx_model = keras2onnx.convert_keras(net,    # keras model
                 name="example",  # the converted ONNX model internal name
                 target_opset=9,      # the ONNX version to export the model to
                 channel_first_ inputs=None)
onnx.save_model(onnx_model, "example.onnx")
```

The network model in ONNX format is imported into Unity as a temp file (Fig.4), whose structure will appear in the following graph (Fig.5).



```
ported Object
model.h5
Open imported NN model as temp file
Source: ONNX
Version: 9
Producer Name: tf2onnx v1.9.2
Inputs (1)
conv2d_input shape: (n:*, h:90, w:3, c:1)
Outputs (1)
dense_2 shape: (n:1, h:1, w:1, c:6)
Memories (0)
Layers (12 using 738.950 embedded weights)
Conv2D name:sequential/conv2d/BiasAdd, activation:Relu, inputs:[conv2d_input], pad:[0,0,0,0], weights:[sequential/conv2d/BiasAdd:0/K (n:2, h:2, w:1, c:128), sequential/conv2d/BiasAdd:0/B (n:1, h:1, w:1, c:128)]
MaxPool2D name:sequential/max_pooling2d/MaxPool, inputs:[sequential/conv2d/BiasAdd], pad:[0,0,0,0], stride:[2,2], pool:[2,2],
Reshape name:sequential/flatten/Reshape, inputs:[sequential/max_pooling2d/MaxPool], pool:[1,1,-1,1,1,1,1,5632],
Dense name:sequential/dense/MatMul, inputs:[sequential/flatten/Reshape], weights:[W (n:5632, h:1, w:1, c:128), B (n:1, h:1, w:1, c:128)]
Add name:sequential/dense/BiasAdd, inputs:[sequential/dense/MatMul,c_csequential/dense/BiasAddFor_sequential/dense/BiasAdd], axis:1,
Activation name:sequential/dense/Relu, activation:Relu, inputs:[sequential/dense/BiasAdd],
Dense name:sequential/dense_1/MatMul, inputs:[sequential/dense/Relu], weights:[W (n:128, h:1, w:1, c:128), B (n:1, h:1, w:1, c:128)]
Add name:sequential/dense_1/BiasAdd, inputs:[sequential/dense_1/MatMul,c_csequential/dense_1/BiasAddFor_sequential/dense_1/BiasAdd], axis:1,
Constants (3 using 262 weights)
Load name:c_csequential/dense_2/BiasAddFor_sequential/dense_2/BiasAdd, weights:[(n:1, h:1, w:1, c:6)]
Load name:c_csequential/dense_1/BiasAddFor_sequential/dense_1/BiasAdd, weights:[(n:1, h:1, w:1, c:128)]
Load name:c_csequential/dense/BiasAddFor_sequential/dense/BiasAdd, weights:[(n:1, h:1, w:1, c:128)]
```

Fig. 4

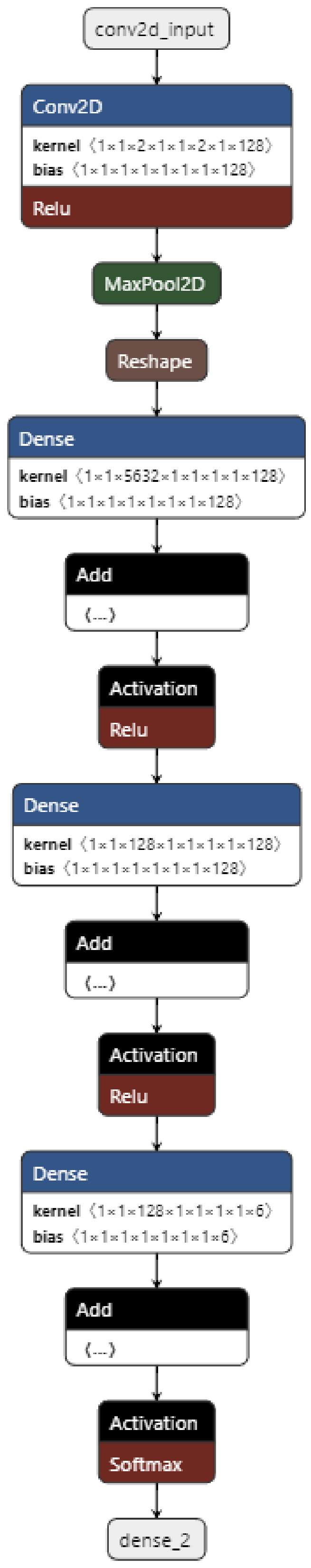


Fig.5: Barracuda model

As a next step, we need to create the inference engine. The core engine interface in Barracuda is called worker. IWorker breaks down the model into executable tasks and schedules them on GPU or CPU.

The scripts are different depending on whether you use a GPU or a CPU. We can create a Worker from the WorkerFactory. We must specify a backend and a loaded model.

```
// GPU
var worker =
WorkerFactory.CreateWorker(WorkerFactory.Type.ComputePrecompiled,
 NNmodel);
// CPU
Var worker= WorkerFactory. CreateWorker(WorkerFactory.Type.CSharpBurst,
NNmodel);
```

After loading the model and creating a worker, we move on to running the model:

```
Tensor input = new Tensor (batch, height, width, channels);
Worker. Execute(input);
```

After executing the model, we can collect the outputs through the instruction:

```
var output = worker.Peekoutput(outputName);
```

If the results are not satisfactory, we can repeat the execution of the model by varying the number of epochs or by modifying the structure.

In our case, to improve the accuracy of the network, we modified the structure by inserting an additional convolutional layer obtaining a value of 99%.

VII. Discussion

Our approach shows how the VR technique can effectively design deep learning applications. Our platform represents a valuable tool for developers with little knowledge of machine learning, allowing rapid design. It is an interactive technique also valid as a learning tool. Compared to existing visualization techniques, Virtual Reality is a potential highly developed for neural network applications. Above all, it has proved to be suitable for classifying images in different scientific sectors. We can assume that it is applicable for any deep learning.

VIII. Conclusion

We have analyzed the most common visualization techniques used to represent neural networks, a valuable tool for their development. Still, we have found that some are sophisticated tools that are difficult to use for inexperienced designers.

Therefore, we have highlighted how to apply Virtual Reality to construct deep learning models. This type of representation, in fact, through the visualization of technical information on the network structure and on the data flow, represents an effective tool both for improving the forecasting capacity of neural networks and for training activities for those who do not have a profound knowledge of deep learning systems.

We have experimented with applying it to a pre-trained neural network to recognize human activities (HAR). Still, we believe that it can effectively design all neural networks for image classification or object detection.

Acknowledgments

This work was developed as an integral component of the project: "An integrated System for indoor people localization, tracking, and monitoring" supported by the Italian MIUR under GRANT PON Research and Innovation 2014-2020 Project Code C35E190000200001, AIM1839112-1: Technologies for living environments.

References Références Referencias

1. Shai Shalev-Shwartz, Shai Ben-David, "Understanding Machine Learning: From Theory to Algorithms," *Cambridge University Press,* 2014.
2. L. Arnold, S. Rebecchi, S. Chevallier, H. Paugam-Moisy," An Introduction to Deep Learning," *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2011.
3. P. G. Benardos, G. C. Vosniakos," Optimizing feedforward artificial neural network architecture," *Engineering Applications of Artificial Intelligence*, 20(3), pp. 365-382, 2007.
4. I. Goodfellow, Y. Bengio, A. Courville, "Deep Learning," *MIT Press*, p. 32, 2016.
5. G. Montavon, W. Samek, K. R. Muller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, 2017.
6. J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson, "Understanding neural networks through deep visualization," *arXiv:1506.06579*, 2015.

7. D. Erhan, Y. Bengio, A. Courville, P. Vincent, “Visualizing higher-layer features of a deep network," *University of Montreal*, 1341, pp. 3, 2009.
8. M. Becker, J. Lippel, T. Zielke, "Gradient descent analysis: On visualizing the training of deep neural networks," *14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications.* pp. 338–345, 2019.
9. D. Liu, W. Cui, K. Jin, Y. Guo, H. Qu, "Deeptracker: Visualizing the training process of convolutional neural networks," *ACM Transactions on Intelligent Systems and Technology*, 10(1), PP. 1–25, 2018.
10. X. Chen, Q. Guan, X. Liang, L. T. Lo, S. Su, T. Estrada, J. Ahrens, "Tensorview: visualizing the training of convolutional neural network using paraview," *1st Workshop on Distributed Infrastructures for Deep Learning DIDL '17*, ACM Press, pp. 11–16, 2017.
11. M. Peters, L. Kempen, M. Nauta, C. Seifert, "Visualising the Training Process of Convolutional Neural Networks for Non-Experts," *CEUR-WS.org,* 2941, paper 108, 2019.
12. L. McInnes, J. Healy, J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv:1802.03426,* 2018.
13. M. Bock, A. Schreiber, "Visualization of neural networks in virtual reality using UNREAL ENGINE," *24th ACM Symposium on Virtual Reality Software and Technology, ser. VRST '18*, ACM, 2018, 132, pp.1–2, 2018.
14. N. Pezzotti, T. Höllt, J. Van Gemert, B. Lelieveldt, E. Eisemann, A. Vilanova, "Deepeyes: Progressive visual analytics for designing deep neural networks," *IEEE Trans. Visual. Comput. Graph*, 24(1), pp.98–108, 2018.
15. M. Kahng, P. Y. Andrews, A. Kalro, D. H. P. Chau, "Activis: Visual exploration of industry-scale deep neural network models," *IEEE Trans. Visual. Comput. Graph*., 24, pp: 88–97, 2017.
16. C, Olah. A. Mordvintsev, L. Schubert, “Feature Visualization," *Distill*, 2(11), 2017.
17. D. Erhan, Y. Bengio, A. Courville, P. Vincent, "Visualizing higher-layer features of a deep network," *Dept. IRO, Universite de Montreal, Tech*., Rep, 4323, 2009.
18. A. Nguyen, J. Yosinski, J. Clune, "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks," arXiv:1602.03616, 2016.
19. A. Chattopadhyay, P.Manupriya, A. Sarkar, N. B. Vineeth, "Neural Network Attributions: A Causal Perspective," arXiv:1902.023024, 2019.
20. M. Kocaoglu, C. Snyder, A. G. Dimakis, S. Causalgan, Vishwanath, "Learning causal implicit generative models with adversarial training," arXiv:1709.02023, 2017.
21. S. Sattarzadeh, M. Sudhakar, A. Lem, S Mehryar, N. Plataniotis, J. Jang, H. Kim, Y. Jeong, S. Lee, K. Bae, "Explaining Convolutional Neural Networks through Attribution-Based Input Sampling and Block-Wise Feature Aggregation," *Computer Vision and Pattern Recognition,* arXiv: 2010.00672.
22. S. Weber, D, Weibel, F. W. Mast, “How to Get There When You Are There Already? Defining Presence in Virtual Reality and the Importance of Perceived Realism”, *Frontiers in Psychology*, 12, p. 1538, 2021.
23. J. Lee, M. Kim, J. Kim, “A Study on Immersion and VR Sickness in Walking Interaction for Immersive Virtual Reality Applications," *Symmetry*, 9(5), p. 78, 2017.
24. A. G. Abulrub, A. N. Attridge, M. A. Williams, “Virtual reality in engineering education: The future for creative learning.” *2011 IEEE Global Engineering Education Conference (EDUCON*), pp. 751–757, 2011.
25. Jeremy Sutton, “What Is Virtual Reality Therapy? The Future of Psychology “, 2020.
26. J. Psotka, "Immersive training systems: Virtual reality and education and training," *Instructional Science,* 23 (5-6), pp. 405–431, 1995.
27. W. Rawat, Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, 29(9), 2017.
28. Y. LeCun, Y. Bengio, G. Hinton, "Deep learning," *Nature,* 521, pp.436-444, 2015.
29. J. Li, J. Cheng, J. Shi, F. Huang, "Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement," *Advances in Computer Science and Information Engineering*, 169, pp 553 558, 2012.
30. C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, J. C. Hart, "The cave: audiovisual experience automatic virtual environment," Communications of the ACM, 35(6), pp. 64 1992.
31. F. Biocca, B. Delaney, “Immersive virtual reality technology," *Communication in the age of virtual reality,* Lawrence Erlbaum Associates, Inc., pp. 57 124, 1995.
32. I. Beĉ, "Neural Network Modelling and Virtual Reality," *The Thousand Faces of Virtual Reality* cap.7, 2014.
33. Juan Nino, Jocelyne Kiss, Geoffrey Edwards, Ernesto Morales, Sherezada Ochoa, and Bruno Bernier, "Enhancing Mobile VR Immersion: A Multimodal System of Neural Networks Approach to An IMU Gesture Controller," *The Engineering Reality of Virtual Reality, IS&T International Symposium on Electronic Imaging*, pp. 184-1-184-5, 2019
34. B. Spanlang, J.-M. Normand, D. Borland, K. Kilteni, E. Giannopoulos, A. Pomes, M. Gonzalez-Franco, D. Perez-Marcos, J. Arroyo-Palacios, and X. N. Muncunill, “How to build an embodiment lab:

achieving body representation illusions in virtual reality," *Frontiers in Robotics and AI,* 1, p. 9, 2014.

35. D. Franklin, "The Meaning of Tenshingoso," *Journal of the U.S. Shintaido Movement Issue*, 10, 2001.
36. A. Fu, Y. Yu, "Real-time Gesture Pattern Classification with IMU Data," 2017.
37. A. R. W. Sait, M. N. Raza, "A Virtual Environment Using Virtual Reality and Artificial Neural Network," *(IJACSA) International Journal of Advanced Computer Science and Applications,* 2(12), n. 12, 2011
38. Yang – Wai Chow, "A Cost-Effective 3D interaction Approach for Immersive Virtual Reality", *The International Journal of Recent Trends in Engineering*, 1(1), pp.529-531, 2009.
39. N. Meissle, A. Wohlan, N.Hochgeschwender, A. Schreibe, "Using Visualization of Convolutional Neural Networks in Virtual Reality for Machine Learning Newcomers," *IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR),* 2019.
40. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition." *Proceedings of the IEEE,* 86(11), pp. 2278-2324, 1998.
41. F. Chollet and al., "Keras," https://keras.io, 2015.
42. L. Bibbò, R. Carotenuto, "An integrated system for indoor people localization, tracking, and monitoring" *Journal of International Scientific Publications*, Materials, Methods & Technologies, ISSN 1314-7269, Volume 15, 2021