A Comparison of ANN, ANN with Calculated Attributes, and CNN

Shiraz Atzmon[1], Maor Nave[1], Shai Isaacs[1], and Alycia Sasson[1]
[1] Department of Management, Bar-Ilan University, Ramat Gan, Israel

ABSTRACT

Machine learning (ML) is the process by which computers develop pattern recognition, where they can learn and then create predictions based on their learning. The level of supervision is what distinguishes one ML method from the other. There are four primary learning models: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Many ML models function in a way that resembles human neural functionality. In our research and this article, we will focus on the supervised learning (SL) method to evaluate the efficiency and effectiveness of two different learning models, Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN).

Our research intends to determine which method is more cost-effective, and compare both methods based on multiple relevant factors, also for assisting in the decision between methods based on use cases. To the best of our knowledge, there are no published comparison articles between ANN and CNN, and no official research evaluates both methods on the same data set. . Most of the information is found on online blogs and forums that have no academic credibility .In our research, we used Python to compare the models, using a regular CPU and not any graphic card. We found that CNN is better in all criteria than ANN, with or without adding attributes. We also observed that the ANN model significantly improves once attributes are added- although it negatively affects its processing time.

KEYWORDS

CNN, ANN, attributes, TensorFlow, ML one_hot_encodeing.

1. INTRODUCTION

ML is a sub-concept of artificial intelligence describing the process of computational learning. ML algorithms are designed to allow computers to make decisions, predictions, and insights based on given information. The process attempts to mimic the human thought process, and some are built similarly to the human neural systems. ML algorithms build a model based on sample data (training data) to make predictions or decisions without being explicitly programmed. [1]

ML is divided into four major categories:

1. Supervised learning - the computer is presented with example inputs and desired outputs (the training set). The goal is to learn a general rule that applies to a statistically sufficient majority of the data, then apply that rule to a new data set and generate the outputs autonomously.
2. Unsupervised learning - No labels are given to the learning algorithm; the algorithm is designed to find patterns and create rules based on the observed set.
3. Semi-supervised learning - a combination of the two former methods, where some outputs are given; there is still a requirement to develop new rules and find patterns in the set.
4. Reinforcement learning - A computer program interacts with a dynamic environment in which it must perform a specific goal. In the navigation process while attempting to reach the desired output, the algorithm is given feedback (reward), which it tries to maximize.

1.1. NN

The primary method to apply ML is using neural network (NN) algorithms. These algorithms imitate human brain activity, so the terminology used to discuss them is also similar to biological terminology. In our research, we wanted to compare two types of NN- ANN, and CNN on the same dataset to determine which is superior to the other.

Neural Networks (NNs) use split tests to analyze the data; thus, the primary data set is divided into two datasets - one for training and one for testing. Each dataset has X and Y variables, where the independent variables are called (x), and the dependent variables are called (y).

Each group of variables is also divided into training and testing sets, each having a designated label: X train, Y train, X test, Y test. The algorithm calls each variable for its cause- the training process or the testing run. [2]

1.2. ANN

In our research, we compared ANNs with CNNs. Artificial neural networks (ANNs) use weights and an activation function in the bulk of their activity. Namely, ANNs simulate the brain's and nervous system's electrical activity, where processing elements (perceptrons) are connected to other processing elements. Typically, the perceptrons are arranged in a vector or layer, with the output of one layer serving as another layer's input. The perceptrons are multiplied by their weights, and the learning itself is based on these weights. [3]

(Figure 1)

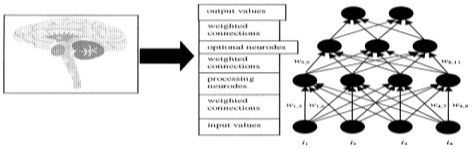


Figure 1 → Demonstration of basic layers of preceptrons in ANN model

All the weight-adjusted input values to a perceptron are then aggregated using a vector to scalar function such as summation (i.e., $y = \Sigma wijxi$), averaging, input maximum, or mode value to produce a single input value to the perceptron. Once the input value is calculated, the perceptron uses a transfer function to produce its output and the input signals for the subsequent processing layer. The transfer function transforms the perceptron's input value. [4]

Typically, this transformation involves using a sigmoid, hyperbolic-tangent, or other nonlinear function. The process is repeated between layers of perceptrons until the neural network produces a final output value or vector of values. ANNs consist of three main layers: Input, Hidden, and Output. At the input layer, the data is inserted. In the hidden layer, it is processed and transformed and moves to the output layers where it is displayed. This article discusses an ANN that uses the supervised learning model. In this type of machine, the dataset used to determine the rules is divided into two groups: training and testing. The machine then studies the data in the training set and applies it to the testing set. The correctness of the result determines the machine's accuracy level. [2]

1.2.1 ANN ATTRIBUTES ADDITION

In our research, we wanted to check if adding attributes to the ANN model could improve its performance, and we proved it does. We added five attributes:

1) Minimum RGB
2) Maximum RGB
3) Average Red
4) Average Green
5) Average Blue

1.3. CNN

Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take an input image, assign importance (learnable weights and biases) to various objects in the image, and differentiate one from the other. CNNs leverage principles from linear algebra, specifically matrix multiplication, to identify patterns within an image. They have three main types of layers: The Convolutional layer, Pooling layer, and Fully-connected (FC) layer. The convolutional layer is the first layer of a convolutional network. While additional convolutional layers or pooling layers can follow convolutional layers, the fully-connected layer is the final layer.

With each layer, the CNN increases complexity, identifying greater portions of the image. Earlier layers focus on simple attributes, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object. [5]

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components: input data, a filter, and an attribute map. For example, in a color image made up of a matrix of pixels in 3D, the input will have three dimensions—height, width, and depth—which correspond to RGB in an image. A kernel (or a filter) is the detecting power of the machine. It will move across the receptive fields of the image, checking if the attribute is present. This process is known as convolution. [6]

The attribute detector is a two-dimensional (2-D) array of weights representing part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an image area, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterward, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as an attribute map, activation map, or a convolved attribute. [7]

The pooling layer reduces the number of parameters in the input by sweeping another filter across the entire input, only that this filter does not have any weights. Instead, the filter applies an aggregation function to the values within the receptive field, populating the output array. Pooling can be done in two primary ways:

1) Max pooling: As the filter moves across the input, it selects the pixel with the maximum value to send to the output array.
2) Average pooling: As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.
   In the Fully-connected layer, each node in the output layers is connected directly to a node in the previous layer. This layer performs the classification based on attributes extracted through the previous layers and their different filters. [8]

(Figure 2 & Figure 3)

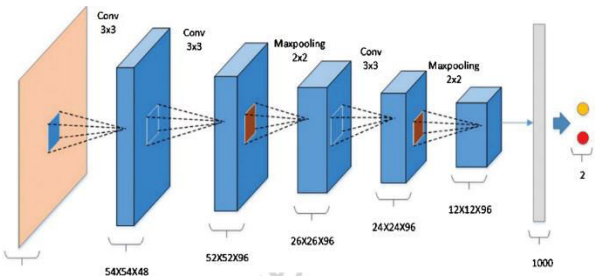


Figure 2 → Demonstration of the network's convolution methodology.

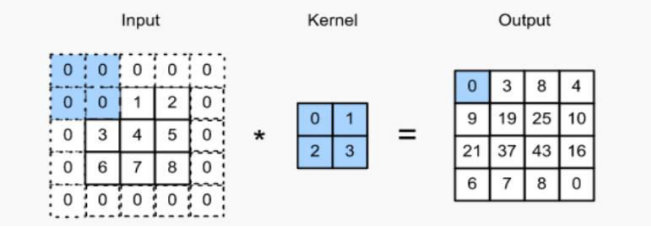


Figure 3 → Demonstration of summation-based kernel filtering and pooling methodology.

1.4. ONE HOT ENCODING VECTOR

In our research, we used One hot encoding, which is the process of converting categorical data variables so they can be provided to ML algorithms to improve predictions. This encoding is done to prepare the data for the ML model and make it easier for the machine to process and compare it to other data. Categorical label values are transformed into numerical data and can be analyzed using mathematical and statistical evaluations. With one hot encoding, we convert each categorical value into a new column where it is assigned with a binary value- 1 or 0. Each value is represented as a binary vector. One hot encoding enables better exploitation of computation power and consequently better performance time. (Figure 4) [9]

Type
AA
AB
CD
AA

Onehot encoding →

Type	AA_Onehot	AB_Onehot	CD_Onehot
AA	1	0	0
AB	0	1	0
CD	0	0	1
AA	0	0	0

Figure 4 → One hot encoded vector manipulation

1.5. ACTIVATION FUNCTIONS

We have used several activation functions in different layers of our model. Both the ANN and the CNN use rectified linear activation function (ReLu), a linear function that will output the input directly if it is positive; otherwise, it will output zero. In the last layer of our ANN model, we used the SoftMax function, a generalization of the logistic function to multiple dimensions. It normalizes the network's output to a probability distribution over the predicted output classes. On the other hand, in our CNN model, we used the Sigmoid activation function in the last layer. A sigmoid function is a bounded, differentiable, real function defined for all real input values. It has a non-negative derivative at each point and exactly one inflection point. Generally speaking, when a model has a Pooling process in it, it is better to use SoftMax, and when it does not, Sigmoid.

1.6. DEVELOP ENVIRONMENT & DATA SET

Our research was done in the developing environment Spyder and coded in Python. Spyder is an open-source environment written in Python designed for scientists, engineers, and data analysts. Python is the common programming language in the field of AI and ML. We used CIFAR-10 as our tested dataset. CIFAR-10 is a collection of 60,000 32x32 color images in 10 different classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.[10]

Each model generated a classification report after each run. The report includes several variables that assist in evaluating the model: accuracy, recall, and precision. We also added a run-time value. The accuracy parameter shows what percentage of the outputs was correct, recall is the ratio of true positives to the sum of true and false negatives, precision is the ratio of true positives to the sum of true and false positives. Each can teach different things about the model and its effectiveness.

Before each model is run, cleaning the data and preparing it for the process is necessary. To do so, we used functions to explore the entire set and check the validity of each variable. We ensured no missing or damaged data and that each class of elements from CIFAR-10 contained the same number of valid images. The model preparation was also done ahead, including the splitting into train and test datasets. The results show with statistical significance that the CNN model is superior to the ANN model in image identification and that adding attributes to the ANN model improves its performance.

2. LITERATURE & RELATED WORK

ANNs are mentioned and discussed in informal communication channels more than they are mentioned in academic literature. There are many articles regarding ANNs as a novel notion, ideology, and technical game-changer, but not as many focus on the scientific aspect of it. Maind and Wanker elaborated on the ANN method, yet their study from 2014 lacks scientific explanations. A more recent paper is more precise. A group of researchers from Malaysia and Nigeria [11] referred to ANNs as statistical data models and gave further explanations of ANNs, as well as on CNNs. They discussed the use of ANNs in Picture Recognition problems and described how the machine works with technical terminology. Clearly, more research is needed to improve ANN's in significant ways and allow interdisciplinary integration. Singh and Banerjee (2019) [12] described the notion and methodology of a perceptron, referred to ANNs, and described them as multilayer perceptrons. Numerous studies are written on the use of CNNs for several causes and in several disciplines, such as microorganism image analysis, defect detection of 3C in products, and Brain Tumor segmentation.

"Artificial Neural Networks (ANNs) are computational processing systems heavily inspired by how biological nervous systems (such as the human brain) operate. ANNs are comprised of a high number of interconnected computational nodes (referred to as neurons), which work entwine in a distributed fashion to collectively learn from the input to optimize its final output." (O'Shea and Nash,2015, pp 1). [13] O'Shea and Nash further defined CNNS as "analogous to traditional ANNs in that they are comprised of neurons that self-optimize through learning. Each neuron will still receive input and perform an operation (such as a scalar product followed by a nonlinear function) - the basis of countless ANNs.

From the input raw image vectors to the final output of the class score, the entire network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply." (pp 2).

Ihme and Pitsch [14] referred to the polling process and described it as follows: "Polling is conducted in three steps (Audet & Dennis, 2000):

1) polling with respect to the continuous variables and fixed categorical parameters.
2) polling in the neighborhood of categorical variables and fixed continuous parameters.
3) extended polling around the neighborhood of points whose cost function is close to the incumbent value."

Several articles discuss the importance of the precision parameter in determining the machines accuracy level. However, no research compares ANNs to CNNs on classification models with backed evidence.

3. RESEARCH QUESTION

What is the performance metric score of each model (ANN, ANN with attributes, CNN) to the Cifar-10 data set under the same laboratory conditions? (Operating system, work environment, computer hardware).

4. RESEARCH INFRASTRUCTURE

4.1. HARDWARE - SOFTWARE INFRASTRUCTURE

In this research, a code was developed to present the different test metrics between different models of machine learning under the same laboratory conditions:

1) Operating system - All models were built under the same operating system, Windows 10.
2) Work environment (development environment) - development of the software code for training and creating the models, as well as predicting and testing the performance metrics of each of them on Anaconda - Spider environment.
3) Computer hardware - All models were developed and tested according to the following computer hardware:

- Laptop Hp → model → 15 - dw0xxx
- Processor - 4 cores, 8 logical processors.
- Processor - 1800 MHz, intel core i5 - 8265U CPU, 1.6 GHz
- Video card - Intel UHD Graphics 620

4.2. PYTHON LIBRARIES

(Figure 5)

Library name	Definition	Use
NumPy	fundamental package for scientific computing in Python	Assist in creating multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting
math	provides access to the mathematical functions	Assisting with database manipulation, using methods to find median averages and quarterly roots during code development
pandas	open source Python package that is most widely used for data science/data analysis and machine learning tasks	Data cleansing, Data fill, Data normalization, Merges and joins, Statistical analysis, Data inspection, Loading and saving data
seaborn	Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.	Presenting the database, pulling insights on the data and the various models developed
matplotlib.pyplot	matplotlib.pyplot is a state-based interface to matplotlib	provides an implicit, MATLAB-like, way of plotting.
TensorFlow	end-to-end open source platform for machine learning.	Assisting in the development of the various models, determination of layers, amounts of neurons in use, and activation functions
Keras	most used deep learning framework	Assisting in the development of the various models, determination of layers, amounts of neurons in use, and activation functions
sklearn.metrics	A library with different methods for testing the quality of models for machine learning	Development models classification matrix assistance
time	provides various time-related functions	Assistance in estimating the execution times of each model

Figure 5 → Python research libraires explanation table.

4.3. CODE GUIDE

4.3.1. DATA EXPLORATION

In this part of the study, we explored the data that are supposed to train and test the various models, with the help of developing functions that help transfer insights on the data before entering the model. The data were first drawn using a constructed method, and labels were fitted to them according to the type of data.

4.3.1.1. DEVELOPED FUNCTIONS

reshape_label → Function reshaping the labels data types from Uint8 to 1D-array

Sample_plot → creates a visualization for existing images in the database, according to the training / test values of the user's choice (according to a defined number of images). (Figure 6)

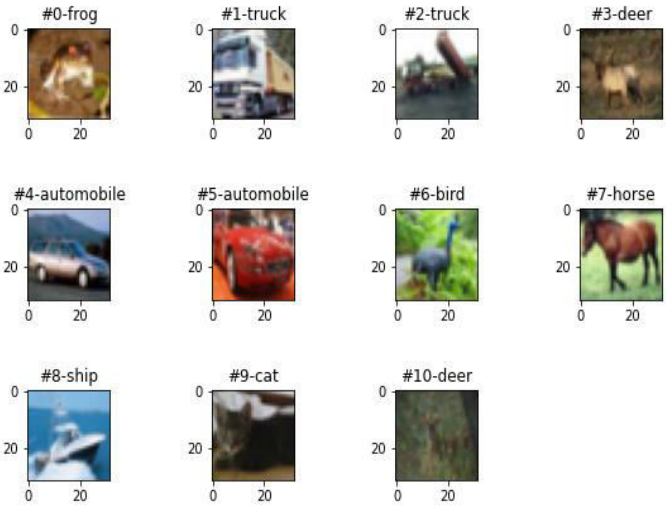


Figure 6 → outputted plot of the 10 first images on cifar10 data set.

4.3.2. DATA NORMALIZATION

This part of the study was conducted on train and test data:

4.3.2.1. DEVELOPED FUNCTIONS

- Rgb_nor → Function that normalizes (between 0-1) the existing RGB values for each object (image), in the database.
- create_batches→Function that accepts defined data parts within the existing database according to indexes (X_train / X_test) and divides it into batches arranged according to the researcher's request (in this specific research, every batch was given 10K image samples).
- add_label_name→ Function that accepts a batch and array of classes (label_names), the function adds to the batch a column (attribute) called label_name. The new column corresponds to the label number within the batch.

  The function returns the batch after the change.

4.3.3. DATA VISUALIZATION

This part of the study was conducted on train and test data:

4.3.3.1. DEVELOPED FUNCTIONS

- batch_stat_plot→ Function that receives a batch and the column's name from which we issue the statistics on the data.
- The function issues a graph to each batch that enters the number of indexes with the existing labels.
- The function saves the plot locally and prints it to the client.
- print_batch_stat→ Function that receives a batch and its number. The function performs printing of the statistics that are also graphically issued during the program.
- sample_stat→ Function that receives a sample of an image and performs a series of actions to extract its statistics:
  - Preserve the minimum and maximum value of the image array that holds the RGB values of the image sample.
  - Save the image type and shape.
  - Save the image label name.
  - Rgb_splitter function call - to display a graph of the image in RGB colors.
  - Issue a histogram to the RGB values in the image sample.
  - Local export and prints the statistics.
  - The function store every sample locally.

  (Figure 7,8,9,10,11 &12)

```
stats for batch # 1
 # of sampels in batch - 10000
bird          1032
frog          1030
ship          1025
cat           1016
airplane      1005
horse         1001
deer           999
truck          981
automobile     974
dog            937
Name: label_name, dtype: int64
```

```
stats for batch # 5
 # of sampels in batch - 10000
dog           1025
truck         1022
cat           1016
automobile    1014
airplane      1014
ship          1003
deer           997
frog           980
horse          977
bird           952
Name: label_name, dtype: int64
```

Figure 7 → example of print_batch_stat function output, batches 1 and 5 statistics print

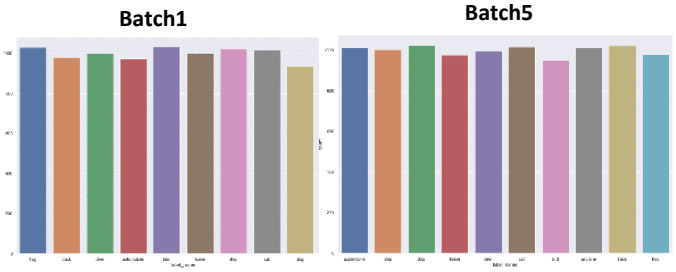
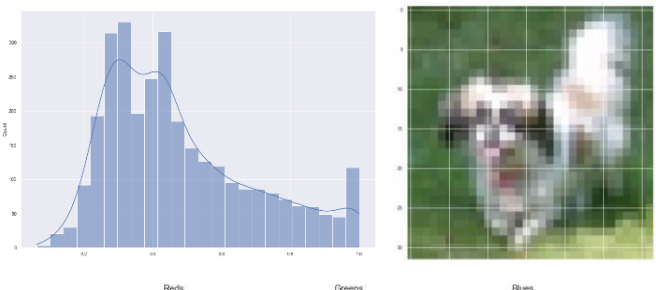


Figure 8 → example of batch_stat_plot function output, batches 1 and 5 statistics plot.



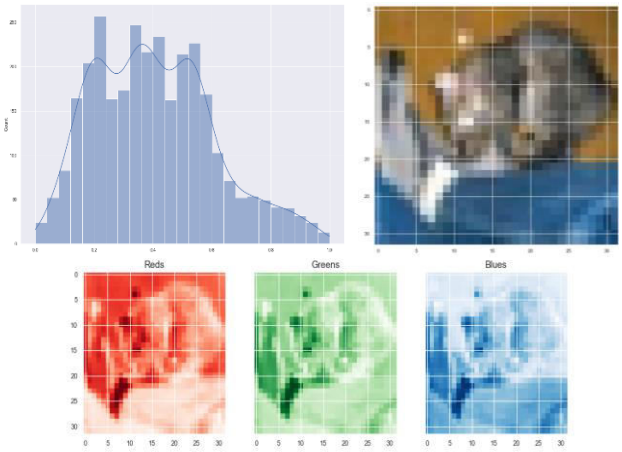


Figure 10 → example of sample_stat function plotted output, out of test batch.

```
Example of image 5850 :
 Min rgb value: 0.06274509803921569, Max rgb value: 1.0
Img shape: (32, 32, 3)
label: dog
```

Figure 11 → example of sample_stat function printed output, out of train batch.

```
Example of image 0 :
 Min rgb value: 0.050980392156862744, Max rgb value: 1.0
Img shape: (32, 32, 3)
label: cat
```

Figure 12 → example of sample_stat function printed output, out of test batch.

4.3.4. DATA PREPARATION

This part of the study was conducted on train and test data:

4.3.4.1. DEVELOPED FUNCTIONS

- one_hot_en → Function that accepts a batch and issues that batch as a new DF where the label_name attribute is characterized as one_hot_vector, which helps the model to reduce run times of every epoch → factoring the label names.
- min_max_rgb → Function that gets a batch and finds for each row (for each image that exists within the batch) its minimum and maximum RGB value→ the min and max values of each image are two new attributes of each batch.
- mean_r_g_b → Function that gets a batch and finds for each row (for each image that exists within the batch), the averaged R, G&B values in it → the averaged R, G&B values of each image are three new attributes of each batch.
- prepare_x → A function that receives a batch and draws from it the relevant attributes (X values) for model training & testing.
- prepare_y → A function that receives a batch and draws from it the relevant attributes (the Y values - as a factorial batch) for model training & testing.

4.3.4.2. FLATTED ATTRIBUTES

- In the ANN model with attributes, we would like to use one layer of 3072 neurons, each of which processes one pixel in an image consisting of 32 * 32 pixels in RGB colors (hence to create one layer, we will perform the calculation 32 * 32 * 3 = 3072).
- We will add the five attributes we created to this model so that one layer of neurons will now be approximated from 3077 neurons describing the model. Section 4.3.6 will review the additional functions developed to train the data model.

4.3.4.3. THE PRODUCTS ISSUED AT THE END OF THE PROCESS

- X_train_no_att - DF consists of 50K images, with 32 pixels, arranged as a three-dimensional array for model training.
- X_test_no_att - DF consists of 10K images, with 32 pixels, arranged as a three-dimensional array to examine the model.
- Y_train - an array of labels attributed to each image for model training
- Y_test - A set of labels attributed to each image for examining the model
- clean_X_train - DF consisting of 50K images, with 3077 pixels (flattened attributes), arranged as a one-dimensional array for model training.
- clean_X_test - DF consists of 10K images, with 3077 pixels (flattened attributes), arranged as a one-dimensional array for examining the model.
- Y_train_batch - DF consists of 50K images, with 10 labels as columns, in a factorial configuration (one-hot vector) for model training.

4.3.5. MODELS BUILD & TRAIN

4.3.5.1. RESEARCH PROCESS

- a test will be performed on each model developed in accordance with the following guidelines:
- After calling the model and initializing it, we will enter the relevant input values → X_train_no_att, Y_train_batch.
- Train the model in 10 and 50 epochs.
- epochs are the number of times the model corrects and maintains the values of the weights according to its learning rate.
- Time measure for training the model in each training round.
- Extracting prediction for the first ten images in the test database.
- Print the model classification report.

4.3.5.2. DEVELOPED FUNCTIONS

1) get_ANN_model → This function creates the model template and returns it according to the selected relevant specifications:

The data type that enters the model is 32 * 32 * 3 image. The model is set on three layers to classify the attributes in it:

- Layer 1 - 3000 neurons, activation function - ReLu (normalization of negative values to zero - removal of exceptions) → correlated to the object volumes (3072 pixels).
- Layer 2 - 1000 neurons, activation function - ReLu (normalization of negative values to zero - removal of exceptions) - division by one-third of the pixels training the model.
- Layer 3 - 10 neurons, activation function - Sigmoid (setting values for classification) - according to the ten existing labels.
- The model uses a loss function corresponding to categorical factor values (the Y values entering this model are in a One hot vector configuration).
- The model presents the following performance metrics - accuracy, Precision, Recall.

2) get_ANN_att_model → This function creates the model template and returns it according to the selected relevant specifications:

The data type that enters the model is a flat attributed image with 3077 attributes (3072 RBG values + 5 additional attributes).The model is set on three layers to classify the attributes in it:

- Layer 1 - 3000 neurons, activation function - ReLu (normalization of negative values to zero - removal of exceptions) → correlated to the object volumes (3077 pixels).
- Layer 2 - 1000 neurons, activation function - ReLu (normalization of negative values to zero - removal of exceptions) - division by one-third of the pixels training the model.
- Layer 3 - 10 neurons, activation function - Sigmoid (setting values for classification) - according to the ten existing labels.
- The model uses a loss function corresponding to categorical factor values (the Y values entering this model are in a One hot vector configuration).
- The model presents the following performance metrics - accuracy, Precision, Recall.

3) get_CNN_model → This function creates the model template and returns it according to the selected relevant specifications:

The data type that enters the model is 32 * 32 * 3 image.

The model is set on two convolution layers to classify the attributes in it:

- Layer 1 – 32: filters that normalize the pixel values to match the relevant object and classify it in the image. Filter size 3 * 3 → randomly selected, activation function → ReLu (normalization of negative values to zero - removal of exceptions).
- Layer 2 – 64: filters that normalize the pixel values (half of each pixel) to match the relevant object and classify it in the image. Filter size 3 * 3 → randomly selected, activation function → ReLu (normalization of negative values to zero - removal of exceptions).
- The model is set on two pooling layers to classify the attributes in it:
  - Layer 1 - By maximum choice between pixel values in 2 * 2 configuration.
  - Layer 2 - By maximum choice between pixel values in 2 * 2 configuration.
- The model is set on two flatten layers to classify the attributes in it:
  - Layer 1 - 64 neurons, activation function - ReLu (normalization of negative values to zero - removal of exceptions) → correlated to the object volumes (64 pixels → From the last filter).
  - Layer 2 - 10 neurons, activation function - SoftMax (setting values for classification) - according to the ten existing labels.
  - The model uses a loss function corresponding to categorical factor values (the Y values entering this model are in a One hot vector configuration).
  - The model presents the following performance metrics - accuracy, Precision, Recall.

4) model_time→ A function that receives the trained model, X and Y values of the trained model, and the desired number of epochs. The function measures the running time of the model training session in seconds.

5. RESEARCH RESULTS

5.1. CLASSIFICATION MATRIX

(Figure 13)

Model	Accuracy -10 Epochs	Precision -10 Epochs	Recall -10 Epochs	Accuracy -50 Epochs	Precision -50 Epochs	Recall -50 Epochs
CNN	0.7843	0.8468	0.7263	0.9701	0.9722	0.9687
ANN with attributes	0.5638	0.1953	0.9504	0.9124	0.1978	0.9992
ANN	0.5628	0.1904	0.9544	0.9122	0.2016	0.9992

Figure 13 → classification matrix results.

5.2. PREDICTIONS & RUN TIME RESULTS

(Figure 14)

Model	Time - 10 Epochs (In min)	Time - 50 Epochs (In min)	10 first predictions– 10 epochs	10 first predictions– 50 epochs
CNN	6.25	30.23	90%	70%
ANN with attributes	14.57	70.47	50%	60%
ANN	14.22	69.55	60%	90%

Figure 14 → Predictions & Run time results.

5.3. SUMMARY

- Ranking of the best-performing models in Classification Matrix indices are:
  - CNN → best accuracy (10 epochs → 0.78, 50 epochs → 0.97) , recall (10 epochs → 0.72, 50 epochs → 0.97) and precision (10 epochs → 0.84, 50 epochs → 0.97) parameters after 10 and 50 epochs.
  - ANN with attributes → second best accuracy (10 epochs → 0.5638, 50 epochs → 0.9124) , recall (10 epochs → 0.9504, 50 epochs → 0.9992) and precision (10 epochs → 0.1953, 50 epochs → 0.1978) parameters after 10 and 50 epochs.
  - ANN → third best accuracy (10 epochs → 0.5628, 50 epochs → 0.9122) , recall (10 epochs → 0.9544, 50 epochs → 0.9992) and precision (10 epochs → 0.1904, 50 epochs → 0.2016) parameters after 10 and 50 epochs.
- Ranking of the best performing models in Predictions & Run time indices are:
  - CNN → best run time (10 epochs → 6.25, 50 epochs → 30.23) and predictions (10 epochs → 90%, 50 epochs → 70%) parameters after 10 and 50 epochs.
  - ANN→ second best run time (10 epochs → 14.57, 50 epochs → 69.55) and predictions (10 epochs → 60%, 50 epochs → 90%) parameters after 10 and 50 epochs.
  - ANN with attributes→ third best run time (10 epochs → 14.57, 50 epochs → 70.47) and predictions (10 epochs → 50%, 50 epochs → 60%) parameters after 10 and 50 epochs.
  - As can be seen from the data appearing in this section, the main consideration made in ranking the models is according to their run times and not necessarily according to the ability of the model to predict the first 10 test inputs.
    This is because the first ten inputs are not sufficient parameters to check the accuracy of each model.

    From this it can be concluded that the prediction calculation was performed as part of the experiment itself in order to verify the model's ability to analyze new test data inputs.

6. DISCUSSION

Based on the research analysis performed and the research question presented (comparison to examine the quality of the various models for solving a classification problem based on the same database → Cifar 10), the following conclusions can be drawn:

- The CNN model has the best quality metrics performance for resolving the classification problem.
- ANN with attributes – advantages:
  1) Adding attributes to a basic ANN model helps increase the model's accuracy.
  2) Flattening the data before entering the model helps increase the model's accuracy.
- ANN with attributes – disadvantages:
  Building a model that receives flat data on high and heavy scales causes an increase in model run times.
- Prediction metric - first ten images in the database → does not provide in-depth information about the quality of the model but is an indication of the machine's ability to predict.
- Limitation & future research:
  1) An in-depth study of image classification should be performed, with additional databases to confirm the study's conclusions.
  2) The given research question should be examined in accordance with different research conditions

(Operating system, Work environment, Computer hardware).

3) Future research should use cifar-100 database.

7. CONCLUSION

In this paper, we have presented a comparison between three different ML models on the cifar-10 database. Furthermore, we set new metrical parameters to check the quality of each model and developed new attributes for the database.

Data exploration, normalization, visualization, and preparation were conducted throughout the experiment.

The CNN models take care of the problem and the research question with the most efficient (program run time and predictions) and most practical (highest classification values) way in both rounds conducted on each model (50 and 10 epochs).

Moreover, we presented the effectiveness of adding attributes to the database before entering the ML model and also presented the effectiveness of flattening the data before entering the model (depending on the type of final data and the problem examined).

Additionally, the significant disadvantages of adding attributes were presented.

Two main limitations were described in this research:

1) Hardware - Software Infrastructure interface.
2) Only one database.

We see our results as encouraging and hope they can provide actionable insights for future research of comparisons between different ML models.

REFERENCES

[1] The definition "without being explicitly programmed" is often attributed to Arthur Samuel, who coined the term "machine learning" in 1959, but the phrase is not found verbatim in this publication and may be a paraphrase that appeared later. Confer "Paraphrasing Arthur Samuel (1959), the question is: How can computers learn to solve problems without being explicitly programmed?" in Koza, John R.; Bennett, Forrest H.; Andre, David; Keane, Martin A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Artificial Intelligence in Design '96. Springer, Dordrecht. pp. 151–170. doi:10.1007/978-94-009-0279-4_9.

[2] Shahab Wahhab Kareem, Zhala Jameel Hamad, Shavan Askar "An evaluation of CNN and ANN in prediction weather forecasting: A review" in ISSN 2712-0562 Sustainable Engineering and Innovation Vol.3, No.2, October 2021,(pp.148-159).

[3] Jaswinder Singh and Rajdeep Banerjee, "A Study on Single and Multi-layer Perceptron Neural Network", August 2019.

[4] Mehmood ul Hasan, Saleem Ullah, Muhammad Jaleed Khan, Khurram Khurshid, "COMPARATIVE ANALYSIS OF SVM, ANN AND CNN FOR CLASSIFYING VEGETATION SPECIES USING HYPERSPECTRAL THERMAL INFRARED DATA", June 2019.

[5] Vidushi Meel, "ANN and CNN: Analyzing Differences and Similarities"

[6] P. Rahul; P. Jagadeesh, "Detection of Dementia Disease using CNN Classifier by Comparing with ANN Classifier " in International Conference on Business Analytics for Technology and Security (ICBATS), February 2022.

[7] Rahul Chauhan &Kamal Kumar Ghanshala, " Convolutional Neural Network (CNN) for Image Detection and Recognition" in First International Conference on Secure Cyber Computing and Communication (ICSCCC), December 2018.

[8] Saad Albawi & Tareq Abed Mohammed & Saad Al-Zawi, "Understanding of a convolutional neural network " in International Conference on Engineering and Technology (ICET), August 2017.

[9] Rahul Chauhan & Kamal Kumar Ghanshala & R.C Joshi, " Convolutional Neural Network (CNN) for Image Detection and Recognition" in First International Conference on Secure Cyber Computing and Communication (ICSCCC), December 2018.

[10] Angelov Plamen; Gegov, Alexander; Jayne, Chrisina; Shen, Qiang (2016-09-06). Advances in Computational Intelligence Systems: contributions Presented at the 16th UK Workshop on Computational Intelligence, September 7–9, 2016, Lancaster, UK. Springer International Publishing. pp. 441–. ISBN 9783319465623. Retrieved 22 January 2018.

[11] O. I. Abiodun et al., "Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition" in IEEE Access, Vol 7 October 2019, (pp. 158820 - 158846).

[12] Jaswinder Singh and Rajdeep Banerjee, "A Study on Single and Multi-layer Perceptron Neural Network", in 3rd International Conference on Computing Methodologies and Communication (ICCMC), August 2019.

[13] Keiron O'Shea and Ryan Nash, "An Introduction to Convolutional Neural Networks" December 2015.

[14] Matthias Ihme and Heinz Pitsch, "Generation of Optimal Artificial Neural Networks Using a Pattern Search Algorithm: Application to Approximation of Chemical Systems" February 2008.