IJHSR
ijhighschoolresearch.org
RESEARCH ARTICLE

Evaluation Of Image Recognition Models In Machine Learning

Alberto Gambino

Liceo Ippolito Nievo, Via Barbarigo, Italy, 35121; albigam125@gmail.com

ABSTRACT: Given an image, a computer must be able to classify what the image represents. While this task is relatively simple for humans, it might be challenging for computers. Taking K-Nearest-Neighbor (KNN) as a representative of traditional machine learning methods and Convolutional neural networks (CNN), Back Propagation neural networks (BPNN), and deep Belief network (DBN) as examples of deep learning models, this paper compares and analyzes the traditional machine learning and deep learning image classification algorithms. We expose the accuracy of training models on the MNIST images dataset to find out which is the best-performing model in the showcased task and provide a starting playground that could be extended to many other models. Comparing different image recognition models can help advance the field's state of the art. By identifying the limitations of current models and developing new models that address these limitations, researchers can improve the overall accuracy and efficiency of image recognition systems. The experimental results show that CNN is the best of the four algorithms in image classification accuracy when using the MNIST dataset. What surprised us was not just that a classical machine learning model, KNN, outperformed two deep learning models, BPNN and DBN, but also that DBN performed better than BPNN.

KEYWORDS: Robotics and Intelligent machines; Machine learning; Back Propagation neural networks; Convolutional neural networks; MNIST images dataset.

Introduction

Artificial Intelligence (AI) can be divided into two main areas: Machine learning and Deep learning. Machine learning is AI that can automatically adapt with minimal human interference. Deep learning is a subset of machine learning that uses artificial neural networks to mimic the learning process of the human brain.[1] AI models must be trained to learn desired actions, and training is achieved using a known dataset of examples. Three types of training are possible: supervised, unsupervised, and semi-supervised. Supervised learning uses labeled datasets, whereas unsupervised learning uses unlabeled datasets. By "labeled," we mean the data is already tagged with the correct answer.[2] Finally, Semi-supervised machine learning is a combination of supervised and unsupervised learning; it uses a small amount of labeled data and a large amount of unlabeled data, which provides the benefits of both unsupervised and supervised learning while avoiding the challenges of finding a large amount of labeled data.[3] If the model's performance on the validation dataset starts to degrade (egg loss begins to increase, or accuracy begins to decrease), then the training process is stopped.[4] This paper compares and analyzes traditional machine learning and deep learning image classification algorithms. A classical machine learning algorithm is the KNN.[5] The k-nearest-neighbor algorithm is a non-parametric, supervised learning classifier that uses proximity to make classifications or predictions about the grouping of an individual data point. The inverse weighting of nearest neighbors' forest variables was performed using Euclidean distances to reduce the bias and averaging of estimates at the higher end of dependent variables' distributions, a known shortcoming of the k-NN method

$$\widehat{y} = \sum_{i=1}^{k} w_i y_i$$

where k is the number of nearest neighbors, field observation of variable y of the nearest neighbor, and weight of the nearest neighbor.[6] Instead, the most important deep learning models for image classification are BPNN, CNN, and DBN. BPNN are neural networks that make use of the concept of backpropagation. It is a process involved in training a neural network. Backpropagation evaluates the expression for the cost function derivative as a product of derivatives between each layer from left to right "backward," with the gradient of the weights between each layer being a simple modification of the partial products.[7] The distance of this mapping to the training set is measured by some error function.

$$E_D(D|w, A) = \sum_m \left(\frac{1}{2}\right) [y(x^m; w, A) - t^m]^2$$

where: Ed is the error; w is the weight being adjusted; y is the output of the neuron on the left side of the weight; z is the weighted sum of inputs to the neuron on the right side of the weight.[8] The task of "learning" is to find a set of connections w that gives a mapping that fits the training set well, that is, has small error ED; it is also hoped that the learned connections will "generalize" well to new examples. A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data.[9] CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories. They can also effectively classify audio, time series, and signal data. A convolutional neural network can have tens or hundreds of layers, and each learns to detect different features of an image. Filters are applied to

DOI: 10.36838/v5i6.1

each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as straightforward features, such as brightness and edges, and increase complexity to features that uniquely define the object. The CNN model is trained with supervised training. Its calculation is usually represented by the following formula:

$$(t) = f(t) * g(t) = (\sum_{r=-\infty}^{+\infty})f(t)g(t-r)]$$

where f and g are two functions; t is a variable; and * represents the convolution operator.[10]

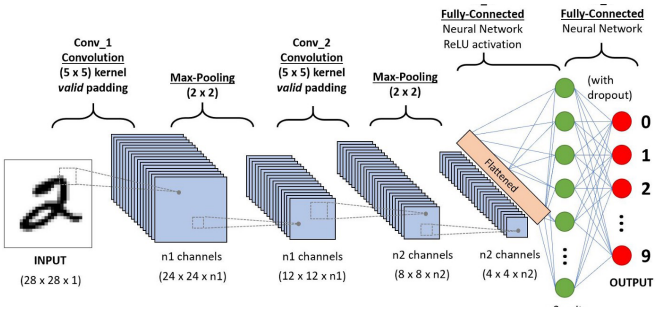

Figure 1: CNN STRUCTURE

Finally, deep belief Networks are generative graphical models, or a class of deep neural networks, composed of multiple layers of latent variables ("hidden units'), with connections between the layers but not between units within each layer.[11] DBNs are trained with semi-supervised training. The key idea behind deep belief nets is that the weights, W, learned by a restricted Boltzmann machine, define both p(v|h, W) and the prior distribution over hidden vectors, p(h|W), so the probability of generating a visible vector v, can be written as:

$$p(v) = (\sum_{h})p(\frac{h}{w})p(\frac{v}{h}, w)$$

where v is the visible vector, h is the hidden vector, W is the weight matrix connecting the visible and hidden units, b is the bias vector for the visible units, c is the bias vector for the hidden units, T represents the transpose operation.[12] After learning W, we keep p(v|h, W), but we replace p(h|W) with a better model of the aggregated posterior distribution over hidden vectors – i.e., the non-factorial distribution produced by averaging the factorial posterior distributions produced by the individual data vectors. The better model is learned by treating the hidden activity vectors produced from the training data as the training data for the next learning module. Hinton, Osindero, and The show that if performed correctly, this replacement improves a variational lower bound on the probability of the training data under the composite model. There are different methods for evaluating the performance of a model.

To evaluate image classification models, the most apparent criterion that can be measured quantitatively is classification accuracy. In this research, we have compared and evaluated the accuracy of each of the four models in recognizing images using the MNIST dataset. MNIST (Mixed National Institute of Standards and Technology) database is a standard database

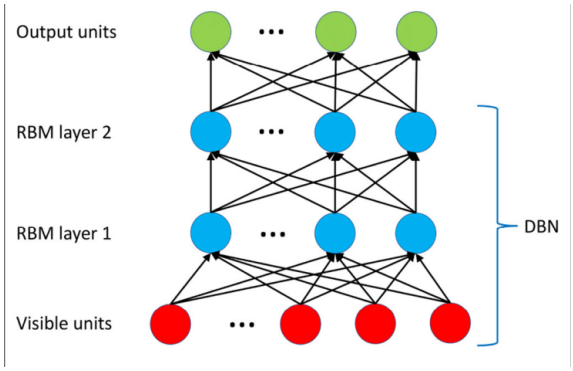

Figure 2: DBN structure

in machine learning. It consists of ten types of handwritten digital grayscale images, of which 60,000 training pictures are tested with a resolution of 28 × 28. The reason for the model's evaluation was to find the optimal solution for image classification using the MNIST dataset and to locate the strengths and weaknesses of the studied models. This work will be helpful for both academia and newcomers in the field of machine learning to strengthen the basis of classification methods further. The results show not just that CNN is the best of the four algorithms in image classification accuracy when using the MNIST dataset but even that KNN outperformed BPNN and DBN and that DBN outshined BPNN. The remainder of this paper is organized as follows. The next part, "literature review," describes what experts in this field have already done to work forward our question. The third part," methodology," describes in detail what we have done in this research to make everything we did replicable. In the fourth part, the obtained results will be explained. The models are compared in detail. Finally, the results are presented.

Literature Review

Nowadays, neural networks are quite popular. Whether it's voice, image recognition, or other assistance, this "new" technology advances the standard of life for various people. Image classification refers to the task of extracting information classes from a multiband raster image. In 1959, Russell Kirsch and his colleagues developed an apparatus that allowed transforming images into grids of numbers the binary language machines could understand.[13] In the 1980s, a Japanese computer scientist, Kunihiko Fukushima, built a self-organizing artificial network of simple and complex cells that could recognize patterns and was unaffected by position shifts. The network, Neocognitron, included several convolutional layers whose (typically rectangular) receptive fields weight vectors (known as filters) had Fukushima's Neocognitron is the grandfather of today's convnets.[14] A few years later, in 1989, a young French scientist, Yann LeCun, applied a backprop style learning algorithm to Fukushima's convolutional neural network (CNN) architecture. After working on

the project for a few years, LeCun released LeNet-5, the first modern convolutional network that introduced some of the essential ingredients we still use in CNNs today.[15] His work created the MNIST dataset of handwritten digits, perhaps the most famous benchmark dataset in machine learning, used in

this research. In the late 1990s, Computer Vision, as a field, largely shifted its focus. Indeed, lots of researchers stopped trying to reconstruct objects by creating 3D models of them (as they were doing before) and instead directed their efforts towards feature-based object recognition. David Lowe was the first to move towards this direction. His paper, "Object Recognition from Local Scale-Invariant Features," describes a visual recognition system that uses local features that are invariant to rotation, location, and, partially, changes in illumination.[16] According to Lowe, these features are somewhat like the properties of neurons in the inferior temporal cortex involved in object detection processes in primate vision. As the field of computer vision kept advancing, the community felt an acute need for a benchmark image dataset and standard evaluation metrics to compare their models' performances. In 2006, the Pascal VOC project was launched.[17] It provided a standardized dataset for object classification and a set of tools for accessing the dataset and annotations. The founders also ran an annual competition from 2006 to 2012 that evaluated the performance of different methods for object class recognition. The ImageNet Large Scale Visual Recognition Competition (ILSVRC) was introduced following PASCAL VOC's footsteps. Unlike Pascal VOC, which only had 20 object categories, the ImageNet dataset contains over a million manually cleaned images across 1k object classes. In 2012, a team from the University of Toronto entered a convolutional neural network model (AlexNet) into the competition, changing everything. The model, similar in its architecture to Yann LeCun's LeNet-5, achieved an error rate of 16.4 (the year before was 26). This was a breakthrough moment for CNNs.[18] In the following years, the error rates in image classification in ILSVRC fell to a few percent, and the winners, ever since 2012, have always been convolutional neural networks. These papers helped us to have a better understanding of computer vision history and to comprehend how each of the analyzed models copes with image classification. The final model analyzed in this research is the deep belief network (DBN). Deep belief nets are probabilistic generative models made of several layers of "Boltzmann Machines." Deep Belief Networks (DBNs) were invented to solve the problems encountered when using traditional neural networks training in deep layered networks, such as slow learning, becoming stuck in local minima due to poor parameter selection, and requiring a lot of training datasets.[19] DBNs were initially introduced by Geoffrey Hinton in 2006. One related work is proposed by Pin Wang En Fan and Peng Wang, which compares and analyzes the traditional machine learning and deep learning image classification algorithms, taking SVM and CNN as examples.[20] Another related work is the paper named "A Comparison of Traditional Machine Learning and Deep Learning in Image Recognition by Yunfei Lai (2019). This paper compares the differences between SVM and deep learning on image recognition with a handwritten digital image recognition application. The results of their work show that the deep learning method is more accurate and more stable in image recognition. Furthermore, our work is related to the paper written by Iwo Różycki and Adam Wolszleger called "Comparison of Neural Network and KNN classifiers, for recognizing hand-written digits".[21] This paper presents a comparison between two different types of classifiers, neural Network and KNN (k nearest neighbors). Their results show that KNN seemed to handle augmented data better, probably because all the training data was available for comparison with the test data. Neural networks, on the other hand, increased in accuracy as we increased the number of epochs, as this allowed the weights on the synapses to be changed according to how many errors in judgment the network made. Another related work was made by Shivam S. Kadam, Amol C. Adamuthe, and Ashwini B. Patil.[22] Their paper presents the application of a convolutional neural network for image classification problems. The results of this paper show that the selection of activation function, optimizer, and dropout rate impacts the accuracy of results. Finally, our work is related to Christine Dewi and Rung-Ching Cheng's work, called "Comparative Analysis of Restricted Boltzmann Machine Models for Image Classification."[23] In this work, they proposed a learning algorithm to find the optimal model complexity for the RBM by improving the hidden layer.

Their results show that Stacking RBM and DBN could improve our classification performance compared to regular RBM. Our study focuses on the same task as these papers, but with the difference that in our research, we compare models that have not been compared in other literature. In the context of the MNIST dataset of handwritten digits, it is important to discuss the state-of-the-art models that have demonstrated remarkable performance. One of the significant advancements in image classification is the Capsule Networks (CapsNets) proposed by Geoffrey Hinton and his team in 2017. CapsNets aim to overcome some limitations of traditional CNNs, such as the need for max-pooling layers and their sensitivity to small changes in input position. CapsNets utilize capsules, which are groups of neurons that represent various properties of an image part. These capsules facilitate better handling of spatial hierarchies and pose variations. A comparison of CapsNets with traditional CNNs in the context of the MNIST dataset would be beneficial to assess their performance.

Additionally, Residual Networks (ResNets) have become a pivotal architecture in deep learning. ResNets introduce skip connections that allow for more efficient training of very deep neural networks by alleviating the vanishing gradient problem. Evaluating the performance of ResNets on MNIST can provide insights into how well they generalize to relatively simple datasets. Furthermore, Attention Mechanisms have gained prominence in various tasks, including image classification. Attention mechanisms allow models to focus on relevant regions of an image, improving the model's ability to identify critical features. Analyzing the performance of Attention Mechanisms on MNIST would highlight their effectiveness in capturing the distinctive characteristics of handwritten digits.

Methods

We have used 'Python' to run the algorithms. First, we downloaded the MNIST training and test dataset and up

DOI: 10.36838/v5i6.1

loaded the code for each model on a Jupyter Notebook to manage the code easily. For the BPNN and the CNN, we have utilized the TensorFlow Keras model.[24] Instead, for the KNN, we have taken the sklearn.neighbors KNeighborsClassifier model.[25] Finally, the DBN code was taken from a coding page and later updated manually with the new TensorFlow 2 instructions.[26] After downloading the code, we started training the models on the MNIST dataset. We trained each of the models twice, using different setups. We ended training the models when the loss function was facing a plateau. Later, we tested each model and analyzed and compared the obtained results. The graphs shown in the research were created using Python plot functions for DBN, CNN, and BPNN. Whereas, for KKN, we have used t-SNE ^25 to reduce the data's dimensionality to plot the graph. Following this procedure, we have compared the image classification accuracy performances of deep learning and machine learning models. The computational resources analysis found that the CNN and DBN models took the longest training time and had higher memory usage due to their deep architectures and larger parameter sizes. Their model file sizes were also larger compared to KNN and BPNN. During inference, CNN and KNN had longer times, and CNN showed higher GPU utilization. KNN and BPNN converged faster. For scalability, CNN and DBN scaled better with powerful hardware and GPUs. Hyperparameter tuning impacted models differently. CNN and DBN were more sensitive to batch size and learning rate. CNN and DBN training consumed more energy, while KNN and BPNN were more energy efficient. Resource usage patterns fluctuated for CNN and DBN, while KNN and BPNN had more stable patterns. CNN and DBN showed higher parallelism and efficiency with multiple GPUs.

Results and Discussion

We tested four different algorithms which required specific hyper-parameters initialization. We trained each model using two different sets of parameters. The first BPNN setup used in the research comprised three layers. The first two layers had five neurons, each with “tanh" as an activation function, whereas the last layer had ten neurons with “SoftMax” as an activation function. We did not know the necessary epochs needed to find a low loss function beforehand, so we started from a small number (20 epochs), which produced a loss = 0.3944 and an accuracy equal to 88,74\%. The loss was still decreasing during training, so we continued training the model, increasing the number of epochs to 50. The accuracy stopped increasing. However, the loss function was still decreasing (reached a value of 0.3235); therefore, we continued training. After 200 epochs, we stopped training the model because the loss function from epoch 173 started increasing. The best value for the loss function obtained during training was 0.2992

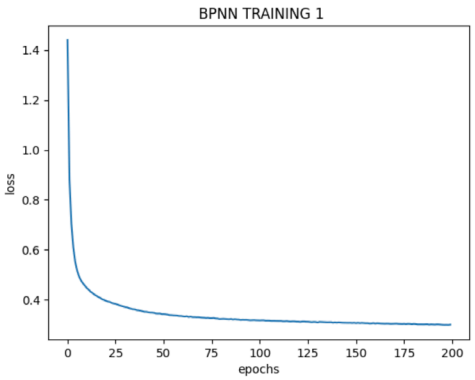

Figure 3: BPNN training loss curve

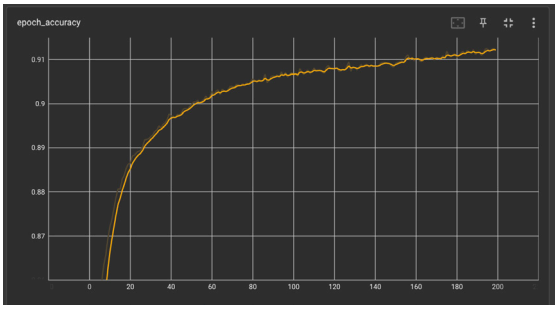

Figure 4: BPNN accuracy curve

After 200 epochs, the model was trained. Therefore, we tested it. In the test, the model obtained an accuracy of 90.23\% % and a loss equal to 0.0283. After completing the BPNN training, we started training the CNN. The first CNN model we used was made of two convolutional blocks. Each convolutional block was divided into a convolutional layer followed by a pooling layer. The convolutional blocks were stacked with two dense layers; the first comprised 500 neurons with “Relu” as an activation function. The second layer had ten neurons with “SoftMax" as an activation function and was used to produce the predicted value. We started training the model with 100 epochs. After 100 epochs, the loss function stopped decreasing, suggesting that the model was sufficiently trained already. The best value obtained for the loss function during training was 0.0220. The following graph shows the CNN loss curve during training.

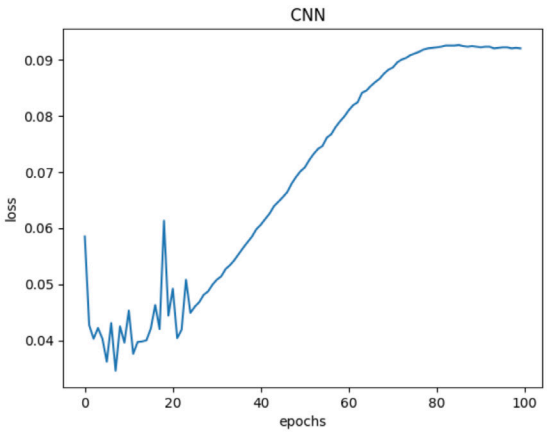

Figure 4: CNN training loss curve

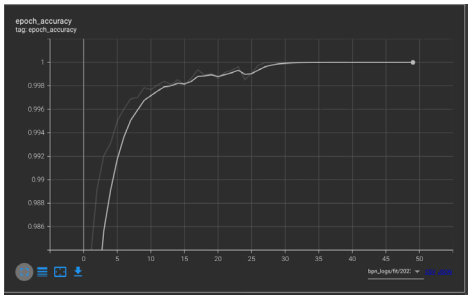

Figure 5: CNN accuracy curve

In the test set, the CNN model obtained an accuracy of 99.37\% and a loss function equal to 0.0512. After that, we performed the same training and evaluation process for the KNN, which was initialized with k=1. Based on the value of k, the k elements closest to the sample just introduced are considered to infer the class of the sample. This model was rapid in execution due to its statistical nature (as opposed to the deep models) and obtained an accuracy of 96.91\% during the test phase. The following graph was realized using t-SNE, which projected the 28x28 MINST images into a two-dimensional space to visualize the clustering performed through the KNN.

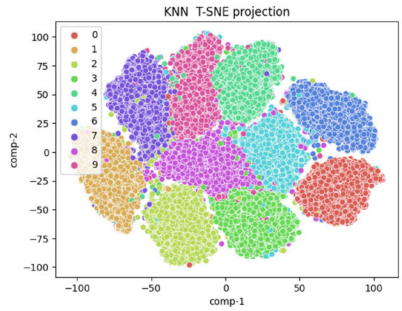

Figure 7: KNN T-SNE projection

Finally, we trained our DBN model, which, in the first configuration, comprised 16 hidden layers with Relu as an activation function. After 100 training epochs, we tested the model performances since the loss function stopped decreasing from epoch 47. The best loss obtained in training was equal to= 0. 225. In the test set, the DBN model obtained an accuracy of 91.81\% and a loss function equal to 0.221.

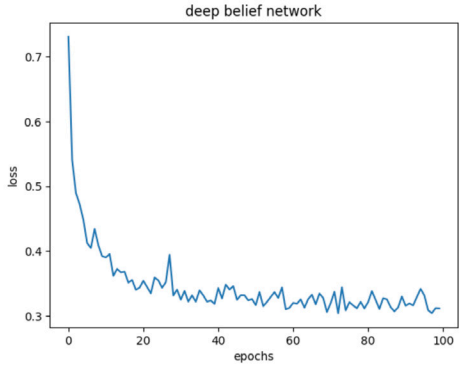

Figure 8: DBN training loss curve

Table 1: First Configuration results

Model	Test loss	Test accuracy	active function layer 1	active function layer 2	active function layer 3	dataset
CNN	0.0512	99.37%	relu	softmax	-	MNIST
BPNN	0.0283	90.23%	tanh	tanh	softmax	MNIST
DBN	0.0221	91.81%	relu	relu	relu	MNIST
KNN	-	96.91%	-	-	-	MNIST

After collecting the results, we trained the models using different configurations. This time, the BPNN model was always made of 3 layers but with different activation functions. The first layer had 30 neurons with the Tanh activation function, whereas the 2-layer had 25 neurons with the Relu activation function. The last layer had ten neurons with Softmax activation function. As indicated in the graph below, the best loss achieved in training was equal to after 100 epochs. We stopped training the model because the loss stopped decreasing, and there was no point in continuing the training—the best loss function obtained in training was 0.0013. The best accuracy obtained in training was 100\%.

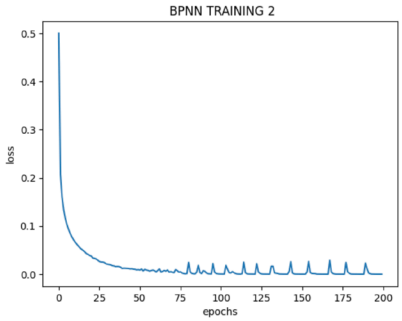

Figure 9: BPNN training two loss curves

In the test, this model achieved an accuracy of 95,92\% and a loss function equal to 0,014. As before, after BPNN, we started working with the new configuration of the CNN mode, which had different activation functions. The second CNN model used in the research comprised two convolutional blocks. Each convolutional block was divided into a convolutional layer followed by a pooling layer. The convolutional blocks were stacked with two dense layers—the first comprised 500 neurons with "tanh" as an activation function. The second layer had ten neurons with "softmax" as an activation function. After 100 epochs, the loss function stopped decreasing. Therefore, the model was trained. The best loss function obtained in training was 0.0496.

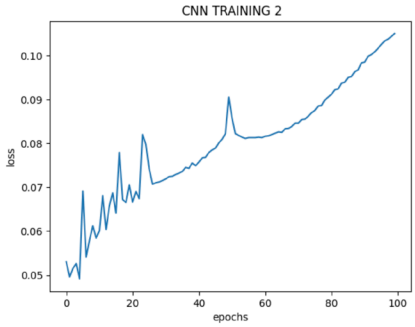

Figure 10: CNN loss curve two training

DOI: 10.36838/v5i6.1

In the test set, the CNN model obtained an accuracy of = 99,029\% and a loss function equal to 0.060. The second KNN model was initialized with k=3. This model not only obtained a lower accuracy (97.05) in the test than the previous KNN model but also took more time to provide the result (44 min). The following graph was taken using t-SNE.

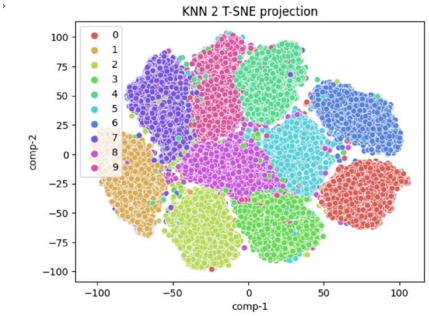

Figure 11: Second T-SNE projection

Finally, we trained the two models of DBN: Our 2 DBN model was made of 8 hidden layers with "sigmoid" as an activation function. After 100 epochs of training, we tested the model from epoch 47; the loss stopped decreasing. The best loss achieved in training was equal to 0.762349.

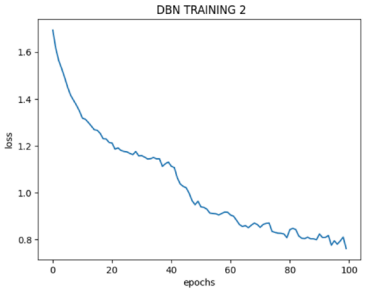

Figure 12: DBN second training loss curve

Table 2: Second configuration results

Model	Test loss	Test accuracy	active function layer 1	active function layer 2	active function layer 3	dataset
CNN	0.060	99.029%	tanh	softmax	-	MNIST
BPNN	0.014	95.92%	relu	softmax	-	MNIST
DBN	0.862	82.02%	sigmoid	sigmoid	sigmoid	MNIST
KNN	-	97.05%	-	-	softmax	MNIST

This study found that when using the large MNIST dataset, the accuracy of CNN is in the first configuration 99.37\% and 99.029\% in the second, the accuracy of BPNN is 90\% in the first configuration and 95,92\% in the second, the accuracy of KNN is 96.8\% in the first configuration and 96.6\% in the second configuration and, finally, the accuracy of DBN was 91.81\% in the first configuration 82\% in the second. The experimental results in this paper show that CNN is the best of the four algorithms in image classification accuracy when using the MNIST dataset. What surprised us was not just that a classical machine learning model, KNN, outperformed two deep learning models, BPNN and DBN, but also that DBN performed better than BPNN. As the graphic below shows, CNN already from the first epochs has obtained better results than BPNN, to then obtain a higher accuracy of almost 10\%.

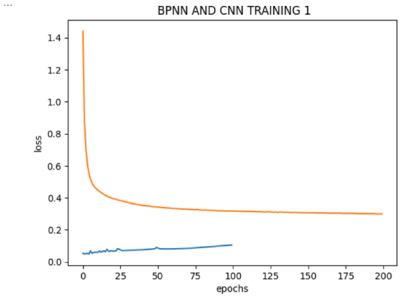

Figure 13: BPNN and CNN first training comparison

To understand what was found, it is necessary to explain the topic in depth. In deep neural networks, each neuron in each layer is fully connected to all the neurons in the previous layer. Because of these large numbers of connections, the number of parameters to be learned increases. As the number of parameters increases, the network becomes more complex. This increased complexity of the network leads to overfitting. Especially in the case of Image data being pixel values of the images as features, the number of input features would be of large dimension.

Furthermore, only some pixel portions of the images may contribute to predicting the output. To overcome these challenges, the Convolution Neural Networks were discovered. The input image data in CNN is subjected to convolution operations such as filtering and max pooling. Then, the resulting data that will be of a lower dimension than the original image data is subjected to fully connected layers to predict the output. By performing the convolution operations, you can find relationships between neighboring pixels, allowing the model to recognize patterns within the image.

Consequently, the number of parameters to be learned decreases, and the network complexity decreases. Convolutional logic is a great strength in images because it considers the relationships between neighboring pixels. Because the BPNNs consist of fully connected layers without a minimum of convolutional logic, they are not effective in capturing correlations between neighboring pixels. This is the reason why BPNN obtained a lower accuracy performance than CNN.

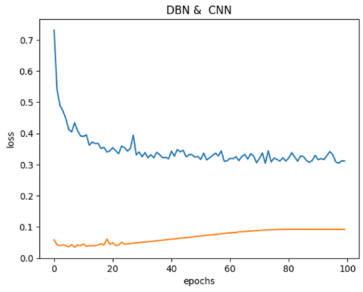

Figure 14: DBN and CNN first training comparison

CNN has an advantage over DBN when the training samples are high but then loses the advantage as the number of labeled training examples decreases, even because the CNN was made of 2 layers while DBN was made of 16, normally a convolutional layer is slightly heavier than a dense layer but since DBN had 14 more layers than CNN, it took longer to train. This is expected since CNN relies on many human-label

ed training samples to learn an accurate model. Another interesting fact in the experiments is that DBN had better results than BPNN. As can be seen in the graph below, DBN achieved a higher accuracy score than BPNN after a smaller number of epochs.

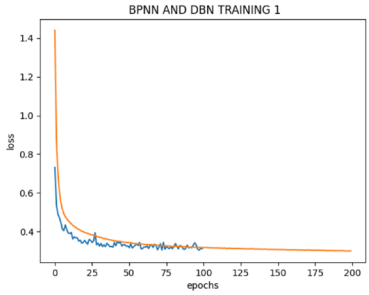

Figure 15: BPNN and DBN first training comparison

This result is very interesting, considering that Deep belief networks (DBNs) nowadays are less widely used than other algorithms for unsupervised and generative learning. To understand why the DBN has obtained better results and has been less computationally costly than a newer deep learning model, it is necessary to understand better how a deep belief network works. In the DBN, we have a hierarchy of layers. The top two layers are the associative memory, and the bottom layer is the visible units—the arrows pointing towards the layer closest to the data point to relationships between all lower layers. Directed acyclic connections in the lower layers translate associative memory to observable variables. The lowest layer of visible units receives input data as binary or actual data. The hidden units represent features that encapsulate the data's correlations. A matrix of proportional weights W connects two layers.

Several Restricted Boltzmann machines together make a Deep Belief Network. DBN algorithm assumes that the best internal representation can be developed by pre-training the network using large sets of unlabeled examples from the same input space without other a priori assumptions. The DBN models also demonstrated that semi-supervised learning methods take advantage of readily available unlabeled data and only need small quantities of labeled examples compared to strictly supervised approaches. This suggests that in the context of lifelong machine learning (or continual learning) systems, the DBN approach of learning the underlying internal representation is superior to the highly engineered BPNN approach. This brings us to the KNN algorithm, the exponent of machine learning models in our research, outshined the DBN and BPNN algorithms, which are, instead, deep learning exponent models. KNN better handled augmented data, probably because all the training data was available for comparison with the test data. Neural networks, on the other hand, increased in accuracy as we increased the number of epochs, as this allowed the weights on the synapses to be changed according to how many errors in judgment the network made. With the backpropagation algorithm, the network improves each time we train it on the training data, whereas the KNN algorithm only needs to be given the training data once for it to work. For this reason, it is not surprising that the KNN model performed better than the deep learning models.

Limitations Of The Study

For this research, we have chosen to use the MNIST dataset for different reasons. First, the MNIST dataset is widely recognized and used as a benchmark for evaluating and comparing different AI models for image recognition tasks. It consists of an extensive collection of handwritten digit images, making it suitable for testing and comparing the performance of various AI algorithms. Secondly, for standardized evaluation, since the MNIST dataset has been extensively studied and used in the AI community, it provides a common ground for comparing different models. Researchers can use consistent evaluation metrics and methodologies when comparing their AI models on the same dataset, allowing fair and standardized comparisons. Another reason for the performance assessment: By comparing different AI models on the MNIST dataset, researchers can gain insights into the strengths and weaknesses of each model. They can evaluate the models' accuracy, speed, robustness, generalization capabilities, and other performance metrics. This comparison can help identify which models perform better for the specific digit recognition task, enabling researchers to make informed decisions about model selection. Not least, the fact of the validation: When researchers publish their findings comparing AI models on the MNIST dataset, it allows other researchers to reproduce their experiments and validate the reported results. This fosters transparency and scientific rigor within the AI community, ensuring the reported advancements are reliable and reproducible. Finally, for knowledge advancement: By comparing different AI models on this dataset, researchers can contribute to the advancement of knowledge in the field of image recognition. They can analyze and discuss each model's underlying algorithms, architectures, and techniques, shedding light on their strengths and limitations. This knowledge can inspire further research and innovation in the field. Writing a paper that compares different AI models for image recognition using only the MNIST dataset can still produce some generalizable results in certain contexts for various reasons: The MNIST dataset serves as a primary benchmark for evaluating and comparing other AI models for image recognition tasks. While it may not capture the complexity of real-world images, it can provide a starting point for assessing the models' fundamental capabilities. The performance achieved on the MNIST dataset can indicate the models' potential and provide a baseline for comparison. Also, the techniques and architectures that perform well on this dataset often have transferable characteristics. While the performance may not directly generalize to more complex datasets, successful models' underlying principles and algorithms can still apply to other image recognition tasks. These techniques can serve as a foundation for further exploration and adaptation to more challenging scenarios. Another reason is that comparing different AI models on the MNIST dataset can provide methodological insights that extend beyond the specific dataset. Researchers can analyze the strengths and weaknesses of various architectures, optimization

DOI: 10.36838/v5i6.1

algorithms, regularization techniques, and hyperparameter choices. These insights can guide future research and inform the design of models for other image recognition tasks. Finally, incremental research progress is valuable, and starting with simpler datasets like MNIST can be a stepping stone toward tackling more complex challenges.

Conclusion

This paper has compared deep learning and machine learning models for image classification. We expected deep learning models to outperform KNN, but the results show that statistical models can perform better for deep learning models not designed for the specific task domain. Furthermore, we noticed that the performances of DBNs were constantly higher compared to the results obtained with BPNN, suggesting that the introduction of RBMs in the first layers of the deep neural network

allows us to achieve better performances. After carrying out the project and analyzing the data, it can be safely concluded that CNN is the best of the four algorithms in image classification accuracy when using the MNIST dataset. However, what cannot be concluded is that one of the other three classifiers would be better than the other one in the area of recognizing handwritten digits.

Although the concluded tests did show some differences in the accuracy, more tests were needed to show that the differences between the classifiers were significant. What can be inferred from these tests is that using our configuration, if the deep learning model is not designed for the domain task, classical machine learning can perform better than deep learning models, in contrast with what other articles have said.[15-17]

Acknowledgments

I would like to express my sincere gratitude to Doctor Eric Sakk for his invaluable guidance and support throughout the course of this research. His insightful comments and suggestions have been instrumental in shaping the direction of this paper. I am truly grateful for his mentorship and dedication. I would also like to extend my thanks to the entire CRISP team for their invaluable assistance and support during this project. Their expertise and willingness to help have been crucial in the making of this research.

References

1. Janiesch, Christian, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning." Electronic Markets 31.3 (2021): 685-695.
2. Chaovalit, Pimwadee, and Lina Zhou. "Movie review mining: A comparison between supervised and unsupervised classification approaches." Proceedings of the 38th annual Hawaii International Conference on System Sciences. IEEE, 2005.
3. Spangenberg, Jost Tobias. "Unsupervised and semi-supervised learning with categorical generative adversarial networks." arXiv preprint arXiv:1511.06390 (2015)
4. https://towardsdatascience.com/the-million-dollar-question-when-to-stop-training-deep-learning- models-fa9b488ac04d
5. Peterson, Leif E. "K-nearest neighbor." Scholarpedia 4.2 (2009): 1883.
6. https://www.cuemath.com/euclideadistanceformula
7. Rumelhart, David E., *et al*. "Backpropagation: The basic theory." Backpropagation: Theory, architectures and applications (1995): 1-34.
8. https://medium.com/swlh/demystified-back-propagation-in-machine-learning-the-hidden-maths- you-want-to-know-about-990843a76f58
9. https://mathworld.wolfram.com/Convolution.html
10. Mallat, Stéphane. "Understanding deep convolutional networks." Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374.2065 (2016): 20150203.
11. Hinton, Geoffrey E. "Deep belief networks." Scholarpedia 4.5 (2009): 5947
12. https://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwitt_TDicb_AhWLyaQKHfj9D14QFnoECBAQAw&url=https%3A%2F%2Fwww.sciencedirect.com%2Ft opics%2Fmathematics%2Fprobability-
13. Russell A. Kirsch, Computer determination of the constituent structure of biological images, Computers and Biomedical Research, Volume 4, Issue 3,1971, Pages 315-328.
14. Fukushima, Kunihiko. "Neocognitron: A hierarchical neural network capable of visual pattern recognition." Neural Networks 1.2 (1988): 119-130.
15. LeCun, Yann. "LeNet-5, convolutional neural networks." URL: http://yann. lecun. com/exdb/lenet 20.5(2015): 14
16. Lowe, David G. "Object recognition from local scale-invariant features." Proceedings of the seventh IEEE International Conference on Computer Vision. Vol. 2. Ieee, 1999.
17. Alom, Md Zahangir, et al. "The history began from alexnet: A comprehensive survey on deep learning approaches." arXiv preprint arXiv:1803.01164 (2018).
18. Hinton, Geoffrey E. "Boltzmann machine." Scholarpedia 2.5 (2007): 1668.
19. Pin Wang, En Fan, Peng Wang, Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, Pattern Recognition Letter shttps://doi.org/10.1016/j.patrec.2020.07.042.
20. Comparison of Neural Network and KNN classifiers for recognizing hand-written digits.IwoRóżycki , Adam Wolszleger.Faculty of Applied Mathematics, Silesian University of Technology,Kaszubska 23, 44-100 Gliwice
21. Kadam, Shivam S., Amol C. Adamuthe, and Ashwini B. Patil. "CNN model for image classification on MNIST and fashion-MNIST dataset." Journal of Scientific Research 64.2 (2020): 374- 384.
22. Christine Dewi, Rung-Ching Chen, Hendry and Hsiu-Te Hung. Journal: Vietnam Journal of Computer Science, 2021, Volume 08, Number 03
23. https://search.yahoo.com/search?ei=utf-8&fr=aaplw&p=%5B23%5D+https%3A%2F%2Fwww.digitalocean.com%2Fcommunit%2Ftutorials%2Fmnist-dataset-in-python
24.https://medium.com/swlh/image-classification-with-k-nearest-neighbours-51b3a289280
25. https://github.com/2015xli/DB
26. sklearn.manifold.TSNE.htm
of survival in patients with glioblastoma multiforme. Cancer. 1987;59(9):1617-25.\

Figures

1. This image shows a CNN sequence to classify handwritten digits made of 2 convolutional layers stacked with two polling layers
2. This image represents the DBNs structures, which are constructed from layers of Restricted Boltzmann machines, and it is necessary to train each RBM layer before training them together. The greedy algorithm teaches one RBM at a time until all RBMs are trained.
3. This graph represents the loss curve of the first BPNN model configuration used in the research during training. The graph was

created using Python plot function.
4. This graph represents the accuracy curve of the first BPNN model co-configuration used in the research during training. The graph w as created using tensor board.
5. This graph represents the loss curve of the first CNN model config uration used in the research during training. The graph was created using Python plot function.
6. This graph represents the accuracy curve of the first CNN model configuration used in the research during training. The graph was created using tensor board.
7. This graph shows the T-sne projection of the first KNN model con figuration used in the research during training. T-SNE is a tool to visualize high-dimensional data. It converts similarities between d ata points to joint probabilities. It tries to minimize the Kullback-Leibler divergence between the joint possibilities of the low-dime nsional embedding and the high-dimensional data.
8. This graph represents the loss curve of the first DBN model config uration used in the research during training. The graph was created using Python plot function.
9. This graph represents the loss curve of the second BPNN model co nfiguration used in the research during training. The graph was cre ated using Python plot function.
10. This graph represents the loss curve of the second CNN model co nfiguration used in the research during training. The graph was cre ated using Python plot function.
11. This graph shows the T-sne projection of the second KNN model configuration used in the research during training. T-SNE is a too l to visualize high-dimensional data. It converts similarities betwe en data points to joint probabilities. It tries to minimize the Kullba ck-Leibler divergence between the joint possibilities of the low-di mensional embedding and the high-dimensional data.
12. This graph represents the loss curve of the second DBN model co nfiguration used in the research during training. The graph was cre ated using Python plot function.
13. This graph compares the loss curve of the first CNN and BPNN model configuration used in the research during training. The ima ge shows that the CNN (blue) loss curve was lower than the BPN N (orange) loss curve.
14. This graph compares the loss curve of the first CNN and DBN m odel configuration used in the research during training. The image shows that the CNN (orange) loss curve was lower than the DBN (blue) loss curve.
15. This graph compares the loss curve of the first BPNN and DBN model configuration used in the research during training. The ima ge shows that the DBN (blue) loss curve was lower than the BPN N (orange) loss curve.

Author

Alberto Gambino is a high school senior at Liceo Ippolito Nievo. He aspires to study mechanical engineering at top universities in England. A talented app developer, he created a successful book trading platform at his school. Alberto's passion for science earned him a scholarship to refurbish his school's lab, courtesy of the MAD competition. He enjoys F1, IT, and playing the piano.

DOI: 10.36838/v5i6.1