Tree-based Methods vs Neural Networks

Ajay S (zwz3wu), Anushruti H (rba7cb), Srivardhan G (efk7cz)

Abstract

Deep Learning is a subset of machine learning algorithms that mimics how humans think and learn. This uses Neural networks, a machine learning algorithm, comprising multiple interconnected nodes in a layered fashion, resembling the brain's structure. Deep learning is a truly disruptive digital technology that is capable of performing complex operations like representation and abstraction to make sense of images, sounds, and text. The application of machine learning is currently rapidly expanding in every industry. In our work, we have compared simpler machine learning algorithms such as Random Forests and XGBoost with regular neural networks and convolutional neural networks using multiple datasets. Our goal is to understand the difference in performance between neural networks and traditional machine learning algorithms.

1. Introduction

In our everyday life, We all use Google translators to translate web pages. We see the pictures in our galleries automatically grouped into folders based on location or person. All these applications use deep learning. Deep learning is a subset of machine learning that mimics how humans think and learn. Deep learning finds its application in various fields for example customer support bots, disease recognition wherein deep learning can detect cancer cells and analyze MRI results to give detailed information, and self-driven cars with seamless sign detection. Deep learning employs a complex, intertwined multi-layer artificial neural network, inspired by the human brain. In traditional machine learning algorithms, we choose the features and classifier, train the model, verify the predictions, and adjust the model if the predictions are not in line with our expectations. However, in deep learning algorithms, the features are automatically extracted and the model learns from its errors. However, deep learning algorithms need much more data than machine learning algorithms for higher accuracy.

2. Traditional Neural Networks

When we look at a picture of a horse, the brain recognizes it irrespective of the color of the picture or the features of the horse. How does the brain do it? The very same idea is behind the development of Neural Networks. Regular Neural networks take inspiration from the human brain. Like the brain, a neural network is made up of multiple layers of

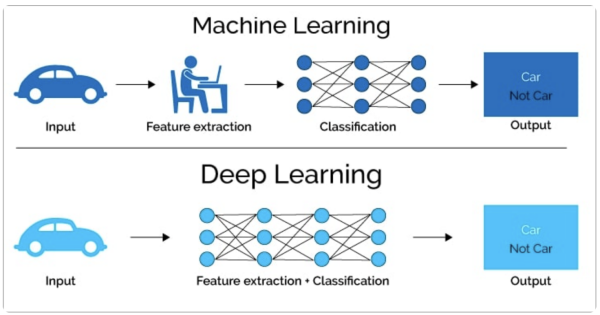

Figure 1: Machine Learning Vs Deep Learning

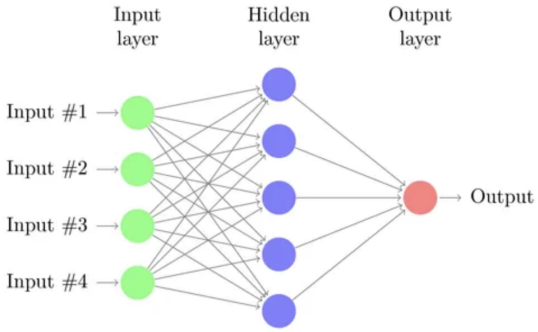

Figure 2: Layers of Neural Network

interconnected neurons working together to solve a particular use case.
A regular neural network consists of three main layers,

1. Input Layer : This takes in input from the dataset and sends it to the intermediate layer.
2. Hidden Layer : All the intermediate layers are called hidden layers. These are responsible for performing the mathematical computation and do the predictions.
3. Output Layer : Once the predictions are available, they are pushed to the output layer as model output in the form of predictions.

2.1. Understanding the working of a simple neural network

The neuron is the basic computational unit of the neural network. It can receive input from an external source or from the other neurons. Let's take an example to understand the working of the neural network. Consider a dataset with three features. For the input neurons, each feature is associated as an individual neuron in the input layer as shown in Figure.3. Each input is associated with a weight. The weight describes the importance of the particular input neuron against the other input neurons. The weight is initialized randomly or to zero initially and then modified through the learning phases using the backpropagation algorithm.
The task of the input layer is to pass the feature values to the hidden layer. The hidden layer then performs a weighted summation of all the feature values and passes this to the activation function. As shown in Figure.3, 0.6 is passed to the activation function. The task of the activation function is to take a number and perform mathematical operations on it. These introduce non-linearity in the output. Examples of activation functions are Sigmoid, Softmax,tanh, etc.
Based on the output of the hidden layers, the output of the neural network or the prediction is derived. For example, the activation function in the example is a binary classification. The hidden layer activates the output neuron which satisfies the condition that is 0.6 greater than 0 and this satisfies y=1. Hence the output neuron with 1 is activated and sent out as model prediction.

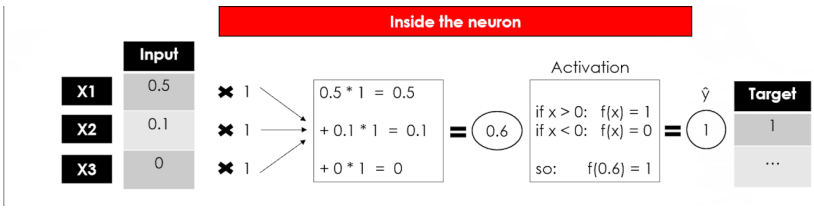

Figure 3: Example showcasing Single hidden layer NN

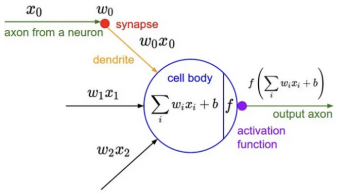

Figure 4: Working of a Neuron

3. Performance Evaluation - Random Forest vs XGBoost vs Neural Network

We compared the performance of tree-based algorithms vs neural networks by using a regression dataset and compared the rmse values. We used three different algorithms Random Forest, Extreme Gradient Boosting, and Regular Neural Network to make the predictions.

3.1. Dataset

We used an Air Quality Index Regression dataset from Kaggle to predict the Air Quality Index based on the features. The dataset consists of 11 features and 33750 rows. The dataset mainly consisted of features such as humidity, wind speed, wind direction,visibility,dew point, temperature, weather type, traffic volume, rain, cloud, snow. Since the weather type was a categorical value, we performed one-hot encoding to convert the categorical values to numeric.

3.2. Results

The experimental results for the Air Quality Index prediction for the three models are shown in Figure 2. Here we can see that, the rmse of XGBoost was the lowest as compared to Random Forest and neural Networks. We expected Neural Networks to have better

Random Forest	XGBoost	Neural Network
0.2935335	0.2842	0.2932186

Figure 5: Air pollution index dataset- rmse values

performance, but that was not the case and the possible reasons could be:

1. Performance can vary depending on the specific dataset. The Air Quality Index regression dataset that we used for analysis was not complex and consisted of many heterogeneous columns which provided a slight advantage to XGBoost.
2. Performance also depends on the hyperparameters passed during training. Neural networks require extensive hyperparameter tuning and experimentation to achieve optimal results.

4. Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of artificial neural networks that are designed to process data that have a spatial structure, such as images, videos, or text. CNNs consist of layers of neurons that perform mathematical operations on the input data, such as multiplying, adding, or applying a non-linear function. The output of one layer is fed as the input to the next layer, until the final layer produces the desired output, such as a class label or a feature vector. They use convolutional layers, which apply filters to the input data. Filters are small matrices of numbers that slide over the input data and produce a new matrix called a feature map. Filters can detect patterns or features in the input data, such as edges, corners, shapes, colors, etc. By using multiple filters and stacking multiple convolutional layers, CNNs can learn complex and hierarchical representations of the input data.

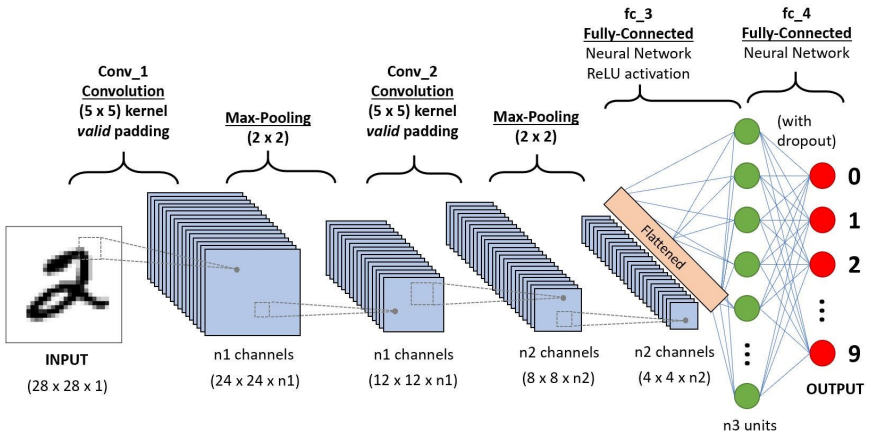

Figure 6: Example of CNN on MNIST image

CNNs use pooling layers, which are layers that reduce the size of the feature maps by applying a function such as max, average, or sum. Pooling layers can help reduce the computational cost and memory usage of the network, as well as make it more robust to noise and variations in the input data. CNNs are different from fully connected neural networks (FCNs), which are neural networks that connect every neuron in one layer to every neuron in the next layer. They do not use convolutional or pooling layers, and they treat the input data as a vector of numbers rather than a matrix or a tensor and can also learn representations of the input data, but are less efficient and less effective than CNNs for spatial data. FCNs have more parameters and require more data to train than CNNs, and they are more prone to overfitting and less invariant to translations, rotations, or scaling of the input data. They are also different from tree-based methods, such as random forest and XGBoost. They do not use layers of neurons or filters to process the input data and do not learn representations of the input data, but rather use simple rules to partition it. These methods can handle both numerical and categorical features, as well as missing values and outliers. They can also provide interpretable results and feature importance scores. However, they are less effective

than CNNs for spatial data, especially for high-dimensional and complex data. Tree-based methods cannot capture spatial relationships or patterns in the input data, and they may require extensive feature engineering or preprocessing to achieve good results.

5. Implementation and Experiments

For the following experiments we selected python and implemented the RF using the sklearn (scikit-learn) package, XGB using XGBoost package, Neural Networks using keras from tensorflow. In the following sections RF refers to Random Forest, XGB refers to XGBoost, NN refers to Fully Connected Neural Network, and CNN refers to Convolutional Neural Network. For our experiments we picked two datsets: MNIST and CIFAR-10. They are two popular datasets for image classification tasks in machine learning. MNIST consists of 70,000 grayscale images of handwritten digits from 0 to 9, each of size 28x28 pixels. CIFAR-10 consists of 60,000 color images of objects from 10 classes, such as airplane, bird, cat, dog, etc., each of size 32x32 pixels. MNIST and CIFAR-10 are widely used for benchmarking and evaluating various algorithms and models for image recognition and computer vision. For the RF, XGB, and NN models the image data is converted from 2D to a 1D vector as these models do not take the spatial data in 2D form as it is and has to be flattened to a single dimension vector for input. We performed possible hypertuning for all the models. For RF and XGBoost we didn't see much improvement in the accuracy and hence we take those models with default parameters as the final model to compare in the experiments. We took the possible best models we identified through our analysis and hypertuning and used them in the experiments.

5.1. RF vs XGB vs NN on MNIST

We trained the random forest, XGBoost, and fully connected NN on MNIST dataset.

Table 1: Comparison on MNIST and CIFAR-10 datasets

Model	**MNIST**	**CIFAR-10**
Random Forest	96%	47%
XGBoost	98%	53%
Fully connected NN	99%	62%
Convolutional NN	-	82%

As we can see from the results in the table, all the models perform better despite the data being images. One reason is that MNIST dataset is relatively simple and low-dimensional compared to other image datasets. The images are grayscale, small, and centered, and the digits have distinct shapes and patterns. Therefore, RF, XGB, and NN may be able to capture the relevant features and discriminate the classes without much difficulty. Hence, we did not compare these models with CNN. For more complex and high-dimensional image datasets, such as CIFAR-10 or ImageNet, CNN may outperform RF, XGB, and NN by a large margin.

5.2. RF vs XGB vs NN vs CNN on CIFAR-10

Now, along with RF, XGB, and NN we also train a CNN on the CIFAR-10 dataset. As we can see from the results in the table, CNN outperforms RF, XGB, and NN by a large margin. This is because CNN can learn more abstract and hierarchical representations of the images that can capture the variations and nuances of the objects and scenes.

5.3. Regularization in neural networks using Dropout

Regularization is a technique that helps prevent overfitting in neural networks. Overfitting occurs when the neural network learns the training data too well and fails to generalize to new or unseen data. Overfitting can result in poor performance and high error rates on the test or validation data. There are variaous regularization networks for neural networks like weight decay, dropout, batch normalization. In this experiment we selected dropout as it seem interesting. Drpout method randomly drops out or deactivates some neurons in the neural network during training. This creates a different version of the neural network for each training iteration, which prevents co-adaptation or dependency among neurons. Dropout also reduces the effective number of parameters in the neural network, making it less prone to overfitting. We used dropout for CNN after each convolution layer with 0.3 as dropout value which means that 30% of neurons will be dropped randomly. Without using dropout we got an accuracy of around 71% for the CNN on the MNIST data where it was overfitting heavily with a training accuracy of . With dropout, we got an accuracy of around 82% which proves the effectiveness of using the dropout.

5.4. Varing layer sizes of NN

We varied the layer sizes of the fully connected neural network to see how it impacts the performance. We selected the MNIST dataset for this experiment and two layer fully connected neural network. The following hidden layer sizes were used for this experiment: 'H1': [1568., 784.], 'H2': [784., 392.], 'H3': [392., 196.], 'H4': [196., 98.], 'H5': [98., 49.]

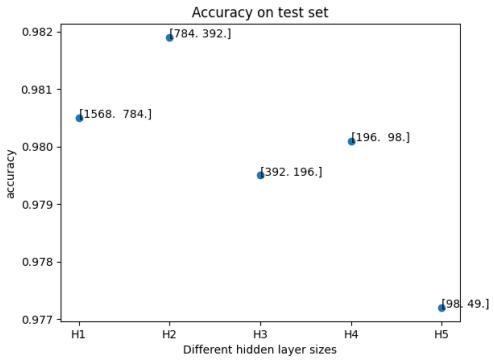

Figure 7: Experiment with different layer sizes

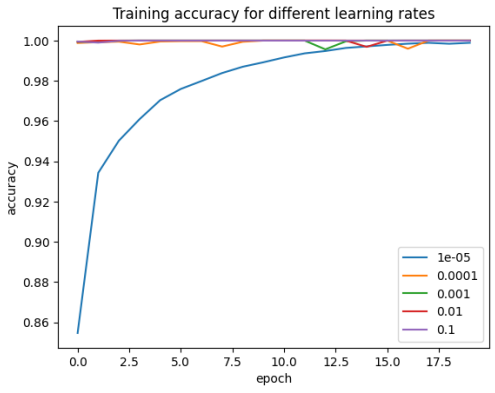

Figure 8: Experiment with different learning rates

The reason for choosing H1 is based on the idea to project the data to higher dimension (greater than 784). In general, the wider the network i.e., having more number of

neurons/nodes in the hidden layers would give more accuracy and it is reflected in the experiment results. But the disadvantage of this is that there will be more computations involved and thereby increasing the inference time. But the training time can be reduced i.e., in lesser number of epochs/iterations the network will be able to learn the features contributing to the accuracy. For H5 as the network has lesser parameters to capture the features of the dataset it performs poorly relative to H1-H4. For H2 it is surprising that it is performing poorly w.r.t H1 and H3 which is not expected in general, and it might be an outlier/false outcome in this experiment. If we run the same experiment many number of times, then it will definitely perform better than H3 and H4. For H5 achieves the least accuracy as expected and is somewhat underfit- ting (Although wouldn't exactly say underfitting as the difference is not so much if we set the scale Y-axis scale to 100). But in general if the network doesn't have enough learnable parameters then it is prone to underfit.

5.5. Varying learning rates

Only the learning rate is varied. Rest all remains same as the base implementation. The following are the learning rates used: L1: 1e-05 — L2: 0.0001 — L3 (default used by the framework): 0.001 — L4: 0.01 — L5: 0.1 Learning rate is the rate at which the network tends to learn to classify correctly. The weights are updated by multiplying the learning rate factor to the gradients, so that the network would take right step size towards the local minima. Too high learning rate would lead to oscillations and the network would never be able to achieve local minima, as shown above for L5: 0.1 learning rate. At the same time too low learning rate would make the network train for so many number of epochs to reach the local minima, as seen for L1: 1e-05. The actual value of learning rate would vary based on the network, loss function, training dataset, and other factors. One of the most used practice is to set a higher learning rate initially and then make it less after certain number of epochs, this way the network will be able to converge to local minima in reasonable time.

6. Summary

In this project, we learned about neural networks, convolutional neural networks and compared them with tree based methods like random forests and XGBoost and also performed different experiments on neural networks. Tree-based methods, such as random forests and XGBoost, use decision trees to make predictions on tabular data. They can handle categorical features, missing values, feature interactions, and other complexities of the data. However, they may not be able to capture the nonlinear relationships or the high-dimensional patterns in the data. Neural networks are more powerful than tree-based methods, as they can approximate any reasonable function using layers of neurons and mathematical operations. However, they also require more data, preprocessing, and computational resources to train and optimize. CNNs are specialized for image data as they can exploit the spatial structure and local patterns of pixels using filters and pooling operations. They also have fewer parameters than fully connected neural networks, which makes them more efficient and less prone to overfitting.

Note : Since we were a team of three, we got permission from Professor for a 7-page report. Also due to limited space we did not mentoin more information and images in the report, but they can be found in the ipynb files.

References

[1] https://www.kaggle.com/datasets/debanganthakuria/air-quality index.

[2] https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep learning/, Basic introduction to convolutional neural network in deep learning.

[3] StatQuest, Youtube channel for neural network.
[1] [2] [3]