Comparing Major Classification Model Architectures on Two Classification Problems

1st Huiwen Duan
Computer Science
Georgia Institute of Technology)
Atlanta, US
hduan38@gatech.edu

***Abstract*—This article studies and compares the computational efficiency on three leading classification algorithms including KNN, SVM, and Neural Networks within the context of a image classification problem and a credit card fraud problem.**

***Index Terms*—KNN, SVM, Neural Network, Classification, Computation, Efficiency**

I. Introduction

My hypothesis is that the neural network model is the most costly in terms of computational power used and time to train, whereas less complex models like KNN and SVM are faster to train and consume less computational power. In terms of prediction time, KNN is slower than SVM and neural networks.

This hypothesis is studied through two classification problems. The first problem is on plant disease identification. Personally, I raise plants at home as a hobby. They often suffer from various diseases which could lead to serious problems if left untreated. As a non-professional, diagnosing plant diseases requires extensive research and cross-referencing with online images of diseased plants. This approach is quite time-consuming, and it is impossible to verify the amateur diagnosis without visiting an actual plant doctor. Therefore, I wanted to build a machine learning assistant to help me with the diagnosis. There exists a public dataset with well-defined labels, making this problem a good candidate for learning and comparing different machine learning models. Furthermore, this problem projects into a very high-dimensional space, as it requires encoding features for each pixel in the image, and is theoretically more costly to train, making it a good candidate for studying computational efficiency across models.

The second problem is on credit card fraud detection. Personally, I have experienced instances of fraudulent charges on my credit card, which caused significant inconvenience and stress. Identifying fraudulent transactions manually is challenging due to the sheer volume of transactions and the sophisticated methods used by fraudsters. Therefore, I wanted to build a machine learning model to help detect potential fraud in real-time. Researchers in Europe have collected an anonymized dataset with 30 features post-PCA reduction. This dataset provides a good contrast with the dataset in the first problem, as it has far fewer dimensions and theoretically requires shorter training time.

II. Plant disease classification

A. Dataset

The dataset used in this study for plant disease identification is the Plant Disease Recognition Dataset available on Kaggle. This dataset includes images of plant leaves categorized into three classes: healthy, powdery, and rust. It contains a total of 1530 images which are divided into training, validation, and test sets. The images are well-labeled, making this dataset ideal for training and comparing different machine learning models for plant disease diagnosis. This problem projects into a high-dimensional space due to the need to encode features for each pixel in the images, which increases the computational cost and makes it a suitable candidate for studying the efficiency of various models [1].

Normalization is crucial for machine learning algorithms like Neural Networks (NN), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) because it ensures that all features contribute equally to the model. Without normalization, features with larger scales can dominate the distance calculations in KNN and SVM, leading to biased predictions. For Neural Networks, unnormalized data can lead to inefficient training and slower convergence since the gradient descent algorithm can become skewed by features with larger values. Therefore, i normalized the data to be centered around 0.45.

B. Model Architectures

The NeuralNet model is a convolutional neural network (CNN) designed to classify images into one of four categories. It starts with an input layer for RGB images. The first convolutional layer applies 16 filters of size 3x3 to detect local features, followed by ReLU activation and max pooling to reduce spatial dimensions. The second layer applies 32 filters, capturing more complex patterns, again followed by ReLU and max pooling. The third layer uses 64 filters for even more intricate features, followed by ReLU and max pooling. The flattened output is then processed by a fully connected layer with 128 neurons, which learns complex representations. The final fully connected layer outputs scores for the four classes, which can be converted into probabilities for classification. This architecture effectively handles high-dimensional data, making it suitable for tasks like plant disease identification. [2]

Another model i used was K nearest neighbors with 4 neighbors as the starting point. There's no real "training" in KNN. Instead, it just memorizes all the data points. When it's time to predict, KNN calculates the distance between the new point and all the stored points to find the closest neighbors. The number of neighbors is an important parameter. a small k might make KNN too sensitive to noise, while a large k might make it miss important details. [3]

The last model I experimented with was Support Vector Machine (SVM). SVM finds the optimal hyperplane that separates different classes in the feature space with the maximum support vectors. Support vectors are the distance between the hyperplane and the nearest data points from each class. SVM can handle both linear and non-linear classification by using kernel functions to transform the data into a higher-dimensional space where it becomes easier to separate. SVM is robust to overfitting, especially in high-dimensional spaces. [4]

C. *Learning Curves*

In neural network model, after tuning for the learning rate and number of epochs, i achieved the best result with 0.0001 as learning rate and 100 epoches with a 97.47% accuracy.

Fig.1 is a line plot showing the training loss as a function of the number of epochs. The x-axis represents the epochs, and the y-axis represents the loss values. The plot provides a clear visualization of how the loss decreases over the training period at plateaued at around 61-65 epoches.

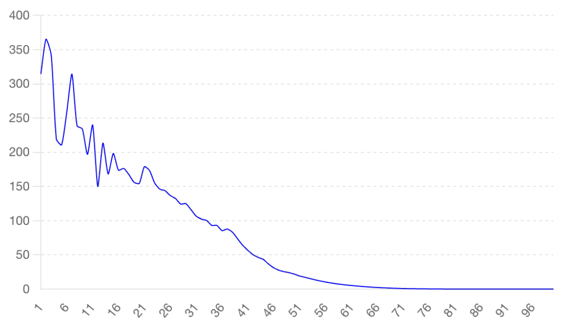

Fig. 1. NN learning curve on plant dataset

In SVM model, 10 iterations was not enough for good convergence. As shown in the chart, both train and test accuracy were very poor. However, SVM was able to eventually converge and generate a 90% accuracy with enough time (not specifying max iteration in the model initialization). I decided not to iteratively train the SVM after 10 iterations due to computational constraints.

D. *Performance On Both Training And Test Data As A Function Of Training Size*

For KNN, the model shows improved generalization with increasing training size, as evidenced by the rising test accuracy. Initially, the model suffers from overfitting, with the train accuracy starting at 1.0 and the test accuracy significantly lower. However, as more training data is used, the test accuracy

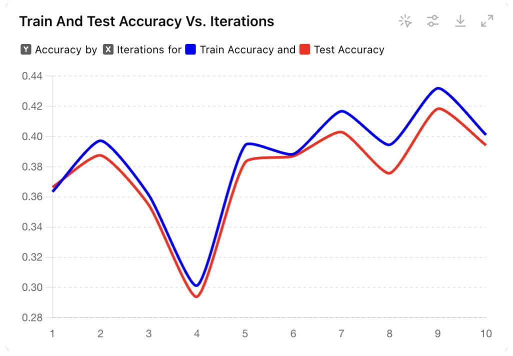

Fig. 2. SVM learning curve on plant dataset

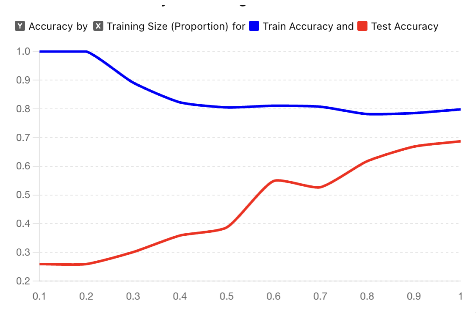

Fig. 3. KNN Train and Test Accuracy Against Training Size for Plant Dataset

steadily improves, while the train accuracy decreases slightly. This indicates that the model is learning to generalize better from the additional data, reducing over fitting and improving its performance on unseen data, making it more reliable for practical applications.

III. Credit Card Fraud Detection

A. *Dataset*

The Credit Card Fraud Detection Dataset [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] is a great resource for anyone working on fraud detection. It contains over 284,000 transactions made by European cardholders in September 2013, with 492 of them labeled as fraud. The data includes 30 features that have been transformed using PCA to keep personal information private. Because it has a realistic imbalance between normal and fraudulent transactions, this dataset is perfect for testing how well different machine learning models can perform on datasets with imbalanced labels.

Model Architecture The SVM and KNN models follow a similar structure as introduced before. However, here I used a simpler neural network model that has three fully connected layers: the first layer has 32 neurons, the second layer has 16 neurons, and the final output layer has 1 neuron with a sigmoid activation function to produce the final prediction. Each hidden layer uses the ReLU activation function to introduce non-linearity into the model. I chose a simpler model because the the number of features in this dataset is a lot lower, and the nature of this problem is simpler.

Metric Selection The primary goal of a fraud detection system is to identify as many fraudulent transactions as possible. High recall ensures that the system catches most of the fraudulent transactions. If recall is low, it means many fraudulent transactions are slipping through undetected, leading to significant financial losses for both the credit card holders and the financial institutions. So I chose recall as the metric for this dataset. Furthermore, because this dataset is very imbalanced, with very few positive labels. Recall can further stress test the model performance. Recall is defined as the ratio of true positive (TP) predictions to the total number of actual positive instances.

B. Learning Curves

In neural network model, after tuning for the learning rate and number of epochs, i achieved the best result with 0.001 as learning rate and 100 epochs with a 83

Fig. is a line plot showing the training loss as a function of the number of epochs. The x-axis represents the epochs, and the y-axis represents the loss values. The plot provides a clear visualization of how the loss decreases over the training period at plateaued at around 10 epochs.

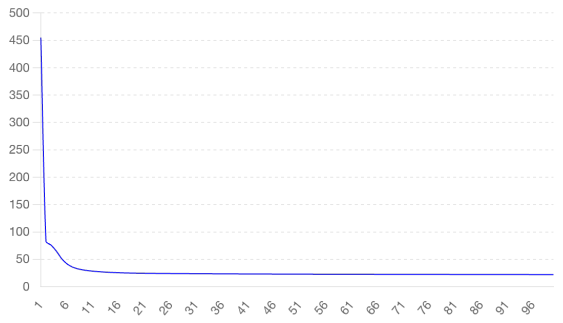

Fig. 4. NN learning curve on credit card fraud

For support vector machine, I achieved very high recall on the credit card problem on both training and testing sets. Both train and test recall improve rapidly within the first few iterations, indicating that the model quickly learns to make accurate predictions. Both recall reach 1.0 early and remain constant, suggesting that the model converges quickly and performs perfectly on both training and test data after initial iterations.

C. Performance On Both Training And Test Data As A Function Of Training Size

For the KNN model, the model shows improved generalization with increasing training size, as evidenced by the rising test recall. Initially, the model suffers from overfitting, as indicated by the significant gap between the train and test recall values at smaller training sizes. However, this issue diminishes as more training data is used, with the test recall steadily improving and converging towards the train recall. The overall trend suggests that more training data helps the model perform better on unseen data, making it more reliable for practical applications.

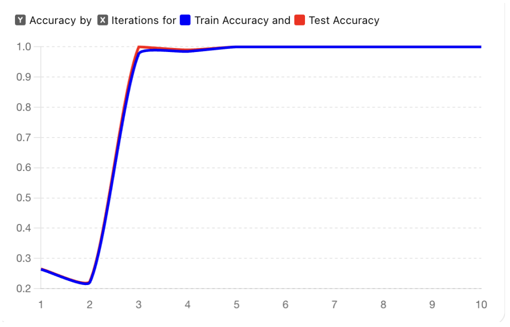

Fig. 5. SVM learning curve on credit card fraud

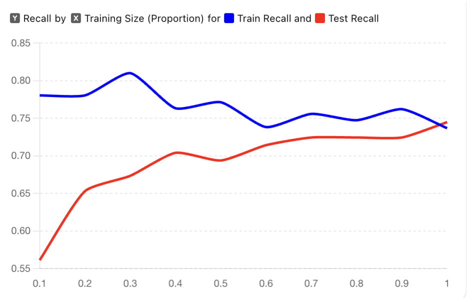

Fig. 6. KNN Train and Test Recall Against Training Size for Credit Card Dataset

IV. Hyperparameters

A. Model complexity in NN

I used convolutional neural network for the plant dataset and it shows better performance than tradiitonal simple neural network on accuracy. When using neural network under the same learning rate and number of epoches, the loss was failing to converge. In their paper, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton compared Convolutional Neural Networks (CNNs) against traditional Neural Networks (NNs) by demonstrating CNNs' superior performance on the ImageNet dataset, a large-scale image classification benchmark. They showed that CNNs, specifically their AlexNet model, could automatically learn and extract hierarchical features from images, such as edges, textures, and object parts, leading to much higher accuracy and lower error rates compared to traditional NNs, which struggled to capture these spatial hierarchies and required extensive manual feature engineering. The results highlighted CNNs' effectiveness in handling the complex structures and patterns present in image data. [15] Therefore, it's reasonable that CNN outperformed NN on my plant disease recognition dataset.

B. Number of neighbors in KNN

I used the gridsearch function from sklearn to derive to the best number of neighbors. For example, in the credit card dataset, it decided that 1 was the best parameter. The recall is high when K is set to 1. The recall fluctuates as the number of neighbors increases. There are local peaks at K=3, K=5, K=7, K=9, and K=13. It's very interesting that the local peaks

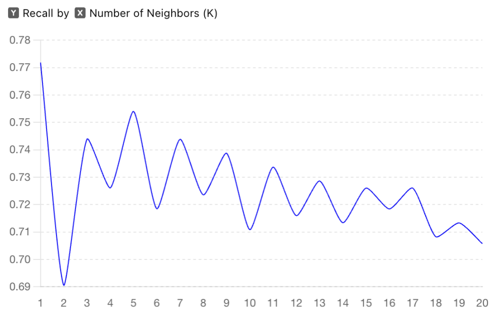

Fig. 7. Hyperparameter Tuning in KNN on credit card dataset

happen at odd numbers, prompting further investigations. With K=1, this means that the KNN model will be highly sensitive to the nearest neighbor, so theoretically this model is very prone to overfitting. However, it generated the best test recall out of the candidates from 1 to 20. It could be something special about the recall metric that led to this observation. When K =1 ,the classifier makes predictions based solely on the closest single training example. This can maximize the chance of correctly identifying positive instances, especially if the positive instances are well-represented and well-separated in the feature space. So when k=1, recall was the best.

C. *Kernal type in SVM*

I experimented with two kernal types with the SVM model: linear and rbf. The linear kernel function is defined as:

$$K(x_i, x_j) = x_i \cdot x_j$$

.

It is computationally efficient and works well when data is linearly separable.

The Radial Basis Function (RBF) kernel is defined as:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

.

It is capable of modeling complex, non-linear relationships, providing higher accuracy on datasets where the decision boundary is not linear. While the linear kernel is simpler and faster to train, the RBF kernel is more flexible and requires careful tuning of the hyperparameter γ. In my credit card dataset, the features are dimensions post PCA decomposition, so this dataset theoretically linear separable. Therefore, there aren't much difference in recall between the rbf and linear kernal type.

V. Discussion on Computational Efficiency

My hypothesis was that Neural Network models are more costly than SVM and KNN. My experiments with the credit card data and plant disease data confirmed this. When the dimension space is small, such as in the credit card dataset, SVM and KNN achieved performance comparable to the neural network model in the desired metric (recall) but required significantly less training time. Training time is loosely correlated to the computational power required, as I ran the

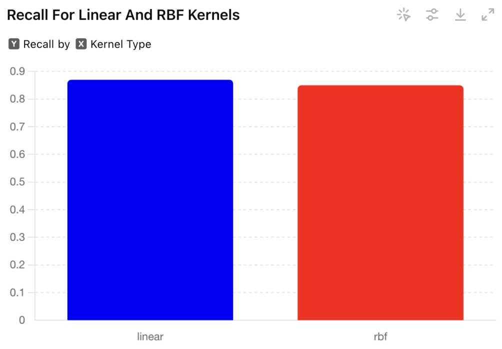

Fig. 8. Recall for linear and rbf kernal types on credit card dataset

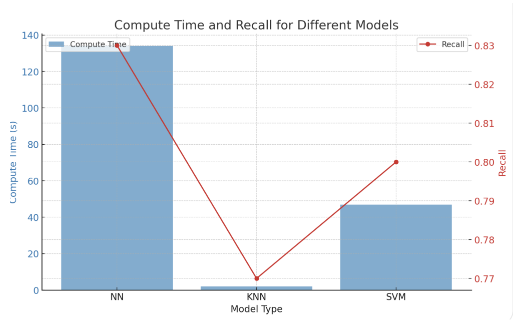

Fig. 9. Compute time and recall for different models on credit card dataset

models locally without any distributed computing. Therefore, KNN and SVM were more computationally efficient for this dataset. Extending this to other use cases, we should not always opt for the most complicated model. Instead, we should consider computational efficiency and choose the model that optimizes both performance and efficiency.

However, in the plant disease dataset, CNN out performed the other models on accuracy by very meaningful amount, with a huge cost as well (40+ minutes of training time). In cases like this, there's a trade-off between performance and cost.

Cost is a very important factor in machine learning deployment because it directly impacts the scalability, accessibility, and sustainability of the solution. High computational costs can limit the ability to process large datasets, perform real-time analysis, or deploy models in resource-constrained environments. Additionally, expensive models can increase operational costs, making them impractical for many applications. By choosing more efficient models, such as KNN or SVM for lower-dimensional data, we can achieve the desired performance while minimizing resource usage and costs. This balance ensures that machine learning solutions are not only effective but also economically viable and widely accessible.

*Note: for KNN, I combined the train and predict time as KNN does not have a formal training process and a lot of the computation happens at prediction layer.

References

[1] R. R. Pritom, "Plant Disease Recognition Dataset," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset. [Accessed: June 9, 2024].

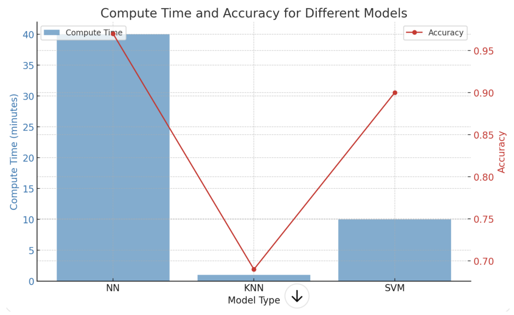

Fig. 10. Compute time and recall for different models on plant disease dataset

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[3] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, Jan. 1967.

[4] C. Cortes and V. Vapnik, “Support-vector networks,” in Machine Learning, vol. 20, pp. 273-297, 1995.

[5] A. Dal Pozzolo, O. Caelen, R. A. Johnson and G. Bontempi, “Calibrating Probability with Undersampling for Unbalanced Classification,” in Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015.

[6] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, “Learned lessons in credit card fraud detection from a practitioner perspective,” Expert Systems with Applications, vol. 41, no. 10, pp. 4915-4928, 2014, Pergamon.

[7] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, “Credit card fraud detection: a realistic modeling and a novel learning strategy,” IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, pp. 3784-3797, 2018.

[8] A. Dal Pozzolo, “Adaptive Machine Learning for Credit Card Fraud Detection,” Ph.D. dissertation, ULB MLG, supervised by G. Bontempi.

[9] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi, “Scarff: a scalable framework for streaming credit card fraud detection with Spark,” Information Fusion, vol. 41, pp. 182-194, 2018, Elsevier.

[10] F. Carcillo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi, “Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization,” International Journal of Data Science and Analytics, vol. 5, no. 4, pp. 285-300, 2018, Springer International Publishing.

[11] B. Lebichot, Y.-A. Le Borgne, L. He, F. Oblé, and G. Bontempi, “Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection,” INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp. 78-88, 2019.

[12] F. Carcillo, Y.-A. Le Borgne, O. Caelen, F. Oblé, and G. Bontempi, “Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection,” Information Sciences, 2019.

[13] Y.-A. Le Borgne and G. Bontempi, “Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook.”

[14] B. Lebichot, G. Paldino, W. Siblini, L. He, F. Oblé, and G. Bontempi, “Incremental learning strategies for credit cards fraud detection,” International Journal of Data Science and Analytics.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ”ImageNet Classification with Deep Convolutional Neural Networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.