Digital Object Identifier

Classification of Alzheimer's Disease in MRI Images using Machine Learning

BAILIE C. DELPIRE[1]
[1]Department of Computer Science, Middle Tennessee State University, Murfreesboro, TN 37132 USA (e-mail: bcd3q@mtmail.mtsu.edu)

ABSTRACT Alzheimer's disease (AD) is a progressive neurodegenerative disorder with no cure that is currently the seventh leading cause of death in the world. Early detection of Alzheimer's is important for slowing its progress and for studying the disease. MRI scans can be used to detect early signs of Alzheimer's and machine learning models have been shown effective at classifying these images. In this study, we review the current literature on machine learning for binary Alzheimer's classification in MRI images, explore convolution neural network and hybrid models for this task, experiment with principal component analysis and Apache Spark machine learning implementations, and compare CPU, GPU, and Google Vertex AI performances. Our top models are a VGG16 feature extractor plus Support Vector Machine classifier with a 98.16% accuracy, a VertexAI AutoML model with a 99.5% average precision, and a SparkML Factorization Machine model on principal component data with a 99.02% accuracy.

INDEX TERMS Alzheimer's Disease, Apache Spark, Convolution Neural Network, DenseNet121, Machine Learning, MRI, ResNet50, Support Vector Machine, Vertex AI, VGG16, XGBoost

I. INTRODUCTION

ALZHEIMER'S disease (AD) is a progressive neurodegenerative disorder that affects memory and thinking skills. It is the most common cause of dementia among older adults and is the seventh leading cause of death globally. Symptoms of Alzheimer's typically appear in later years and worsen with time. Symptoms include memory problems, affected speech and reasoning, and confusion. Those in the severe stages of Alzheimer's cannot communicate or perform simple tasks and are completely dependent on the care of others.

Alzheimer's may begin affecting the brain a decade or more before symptoms first appear. The abnormal buildup of proteins in the brain causes previously healthy neurons to stop functioning, lose connection, and die. This begins in the hippocampus and entorhinal cortex and progresses until brain tissue is widely damaged and has shrunk significantly. There is no cure for Alzheimer's, but current treatments can slow its progress.

Magnetic resonance imaging (MRI) is used to support an Alzheimer's diagnosis and to rule out other possible causes for the symptoms. MRI images can also be used to assess the stages of Alzheimer's disease. Early detection is important so treatment can begin early, which can help maintain a person's daily function for some time. For this reason, the use of MRI images to detect the early stages of Alzheimer's is researched. Machine learning is a powerful tool for image classification problems such as this.

II. PROJECT OVERVIEW

The goal of this project is to:

1) Review the current literature on machine learning for binary Alzheimer's classification in MRI images.
2) Evaluate CNN and hybrid models for this task.
3) Compare CPU, GPU, and Google Vertex AI performances.
4) Experiment with Principal Component Analysis and Apache Spark's Machine Learning Library.

III. LITERATURE REVIEW

Most of the current literature on classifying MRI images with Alzheimer's includes research on Convolution Neural Networks (CNN) and state-of-the-art models, like ResNet, that have CNN architectures. In this section, we review many of the current studies which use CNNs, as well as some that use other models like Support Vector Machines (SVM) or Deep Boltzman Machines instead. See Table 1 for an overview of the literature review papers, the models they used, and their reported accuracies.

A. CNN LITERATURE

Hussain et. al. [7] propose a 12-layer CNN model that consists of four sets of convolution and max pooling layers, a flattening layer, a dense layer, a leaky ReLU layer, and a dense output layer. They use the OASIS dataset with 56 dementia-present MRI images and 56 healthy control MRI images. They first perform image resizing for size consistency across the dataset and image denoising. Then, with an 80/20 train/test split, they report an accuracy of 97.75%, AUC of 99.21%, and F1 score of 97%. They compare this with InceptionV3, Xception, Mobile-NetV2, and VGG19 models on the same dataset and report that their proposed model performed the best, with the next best performing model being InceptionV3 with an accuracy of 90.62%.

In [10], Mamun et. al. use a smaller, 9-layer CNN model with three sets of convolution and max pooling layers, a flattening layer, a fully connected layer, and a dense output layer with softmax. They use a dataset of 6,219 MRI images and a 70/15/15 training/validation/testing split and report an accuracy of 97.60%, AUC of 99.26%, and recall of 97%. They too compare this with a few state-of-the-art models, namely ResNet101, DenseNet121, and VGG16. ResNet101 had the highest accuracy of the three at 73.85%.

Sarraf et. al. [8] train the LeNet-5 CNN architecture on 270,900 images and test it on 90,300 images which are obtained from the 3D MRI scans on 28 Alzheimer's patients and 15 healthy control subjects. With 5-fold cross-validation, they report a mean accuracy of 96.86%. LeNet is a relatively small CNN architecture with only two convolutional blocks, however, with such a high number of images to train on, these results indicate the high success of CNN models on MRI data.

In [9], Abbas et. al. state that the drawback of current CNN approaches to this problem is the requirement to find identifying regions of interest in MRI images and how the location of these regions highly influences performance. Their solution for overcoming this drawback is to use a three-dimensional Jacobian domain CNN for classification. Their experiment used 143 images for training, 47 for validation, and 48 for testing. Their model begins with a Jacobian map on the image data, used as input into a CNN model that consists of three convolution layers, each followed by a max pooling layer. Their results for binary classification are 96.61% accuracy, 97.83% sensitivity, 95.92% specificity, and 98.34% AUC.

Basaia et. al. [6] use an "all convolutional network" 3D CNN on the ADNI dataset of 294 AD and 352 healthy control (HC) subjects. An "all convolutional network" uses standard convolutional layers where max-pooling layers would normally be used in a standard CNN architecture. Their architecture includes 12 blocks of convolutional layers with alternating strides of 1 and 2. Two blocks use 50 5x5x5 kernels and 10 blocks use between 100 and 1600 3x3x3 kernels. They used a 90/10 train/test split and reported a 99% accuracy on the binary classification problem of AD vs HC.

In [1], Chaihtra et. al. perform multi-class classification on a dataset consisting of 2,565 MRI images with classes of healthy control, very mild dementia, mild dementia, and moderate dementia. Their best-performing model uses DenseNet121 trained on ImageNet with an extra two linked layers, three dropout layers, three dense layers, and an output layer with a softmax classifier. They reported a 97% validation accuracy after training with an 80/20 data split and a 91% test accuracy.

Sharma et. al. [4] experiment with using the DenseNet201 model for feature extraction and SVM, GaussianNB, and XGBoost as classifiers. They use a dataset of 6,400 images with classes of healthy control, very mild dementia, mild dementia, and moderate dementia, but augment the dataset with rotated images until they have a dataset size of 10,760 images. They report that DenseNet201 with GaussianNB had the highest accuracy at 91.75%, followed by XGBoost at 91.13%, followed by SVM at 91.03%.

AlSaeed et. al. [3] perform a similar experiment on ResNet50 with Softmax, SVM, and Random Forest classifiers. They use the ADNI and MIRIAD Alzheimer's datasets with a total of 555 training images, 111 validation images and 75 testing images in the ADNI dataset and 530 training images, 105 validation images, and 73 testing images in the MIRIAD dataset. The images are either classified as healthy control or Alzheimer's diagnosed. The most successful model was ResNet50 with Softmax at 99% accuracy, 98% specificity, 99% sensitivity, and 98% F-Measure. This is followed by ResNet50 with an SVM classifier at 92% accuracy and ResNet50 with a Random Forest classifier at 85.7% accuracy. These results are for the ADNI dataset, and the MIRIAD dataset resulted in lower scores.

B. NON-CNN LITERATURE

In [2], Uma et. al. experiment with using a support vector machine for binary classification on MRI images. They use a dataset of 149 training images and 51 testing images and perform contrast enhancement on the data through adaptive histogram equalization. They then use gray level occurrence matrix for feature extraction and use 13 Haralick features as the input data. With 7-fold cross-validation, they report an accuracy of 84%. However, they report a lower accuracy score of 73% on a larger dataset of 5,121 training images and 1,279 testing images with 10-fold cross-validation. We will see later that an SVM can attain higher accuracies after performing different feature extraction methods.

Suk et. al. [5], experiment with using a Deep Boltzman Machine model on the ADNI dataset consisting of 93 subjects diagnosed with Alzheimer's Disease, 204 mildly cognitive impaired (MCI) subjects, and 101 control subjects. They take the image input, perform patch extraction and preprocessing, before passing it to a multi-modal Deep Boltzman Machine model. These features, in turn, are passed to an SVM classifier, for which they received an accuracy of 92.38%, sensitivity of 91.54%, and specificity of 94.56% for binary classification on the AD/NC data and 84.24% accuracy, 99.58% sensitivity, and 53.79% specificity for binary classification on the MCI/HC dataset.

Paper	Models	Accuracies
Deep Learning Based Model for Alzheimer's Disease Detection Using Brain MRI Images [10]	9-layer CNN	97.6%
Deep Learning Based Binary Classification for Alzheimer's Disease Detection using Brain MRI Images [7]	12-layer CNN	97.75%
Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks [6]	"All Convolutional Network" 3D CNN	99%
Transformed domain convolutional neural network for Alzheimer's disease diagnosis using structural MRI [9]	Jacobian Map Feed CNN	96.61%
Deep Learning-based Pipeline to Recognize Alzheimer's Disease using fMRI Data [8]	LeNet-5	96.85%
Alzheimer's Disease Detection from Brain MRI Data using Deep Learning Techniques [1]	DenseNet121	91%
Brain MRI Analysis for Alzheimer's Disease Diagnosis Using CNN-Based Feature Extraction and Machine Learning [3]	ResNet50 + Softmax, ResNet50 + SVM, ResNet50 + RF	99%, 92%, 85.7%
HTLML: Hybrid AI Based Model for Detection of Alzheimer's Disease [4]	DenseNet201 + Gaussian NB, DenseNet201 + XG Boost, DenseNet201 + SVM	91.75%, 91.13%, 91.03%
Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis [5]	Deep Bolzmann Machine	95.35%
Binary Classification of Alzheimer's disease using MRI images and Support Vector Machine [2]	SVM	84%

TABLE 1. Literature Review

IV. METHODS

A. DATA

The dataset is obtained from Kaggle [11]. It contains 6,400 images, with 3,200 images labeled as demented, 2,240 labeled as very mildly demented, 896 labeled as mildly demented, and 64 labeled as moderately demented. The images are divided into a directory of training images and a directory of testing images. Each image is a 176 by 208-pixel, 1-channel image. With the intention of performing binary classification on this dataset and creating our own train/test splits, the images are refactored into a directory of all Alzheimer's diagnosed images and a directory of all healthy control images. See Figure 1 for an example MRI image from each class.

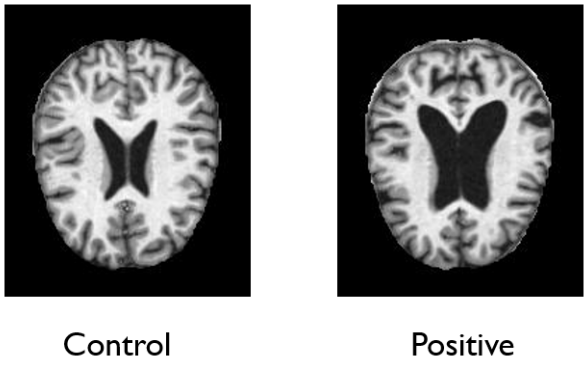

FIGURE 1. Sample Data from Each Class

B. IMAGE PROCESSING

Image processing is the task of turning images into suitable input data. Images are made up of pixels that are represented in a jpeg or png file as raw hexadecimal byte data. This raw data cannot be used directly, therefore we need to load images as pixel data instead by converting the hexadecimal values to decimal values. For 1-channel or gray-scale images, each pixel is represented as one byte, or 8 bits. This means that a pixel can be represented as an integer between 0 and 255, with 0 being completely black and 255 being completely white. These pixel integers can be stored in a structure, such as an array, to be used as input data for machine learning models. To read and process the image data, the following code is used:

```
def grab_image(img_path):
    img = image.load_img(img_path,
          target_size=(208, 176))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    return x
```

C. CNN AND STATE-OF-THE-ART MODELS

Convolutional neural networks are often used for image data because they use 2-dimensional input data and are able to capture the spatial relationships between features. Many state-of-the-art models feature CNNs for this reason.

1) CNN

The architecture of our CNN model is shown in the code below. It is a 9-layer model built with Keras TensorFlow layers. Each of the three convolution layers has 32 filters, has a kernel size of 3, and uses a relu activation function. There is a max pooling layer between each convolution layer for feature pooling. The last few layers are a flattening layer, to flatten the data before it can be passed to a dense layer, a dense layer with relu activation, and finally a dense output layer for the two classes.

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, 3,
        activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(32, 3,
        activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(32, 3,
        activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,
        activation='relu'),
    tf.keras.layers.Dense(2)
])
```

2) Hybrid Models

The hybrid models we test use either VGG-16, DenseNet121, or ResNet50 as a feature detector and either XGBoost or SVM as a classifier. VGG-16 is comprised of five convolution blocks, each ending with a pooling layer. The first two blocks have two convolution layers and the last three have three convolution layers. It ends with three dense layers. It is considered to be one of the best vision model architectures because of its uniquely small number of layers with weights and its high accuracy.

DenseNet121 consists of 1 7x7 convolution layer, 58 3x3 convolution layers, 61 1x1 convolution layers, 4 average pooling layers, and 1 fully connected layer. Its name stems from the fact that it is a densely connected network with each layer connected to every other layer in a feed-forward fashion. This is meant to alleviate the vanishing-gradient problem and reduce the number of parameters.

ResNet50 is 50 layers deep, with 48 convolution layers, a max pooling layer, and an average pooling layer. It is a residual network where residual blocks of layers are stacked. In ResNet, the number of filters in each layer is the same depending on the size of the output feature map. Additionally, if the feature map's size is halved, the number of filters is doubled. This is to maintain the time complexity of each layer.

The XGBoost model, or extreme gradient boosting model, is an ensemble of weak prediction models, typically decision trees. It is highly efficient, flexible, and portable. This Gradient Boosting framework library is distributed and provides parallel tree boosting which makes it ideal for big data problems. A Support Vector Machine constructs a hyperplane or a set of hyperplanes with the greatest possible margin, or distance to the nearest training data point. This hyperplane is used to separate the classes of data and classify new points into one class or another.

Each of the three state-of-the-art feature extractor models are pretrained on the ImageNet dataset, consisting of hundreds of thousands of images. In order to use them as feature extractors rather than trainable models, we set the trainable parameter to false for every layer in the model. For example,

```
for layer in densenet121_model.layers :
    layer.trainable = False
```

Then we use the model to extract features for both the training and testing data.

The XGBoost model we use for classification comes from the XGBoost library and is simply:

```
    model = xgb.XGBClassifier()
```

The SVM model we use comes from the scikit-learn library and uses a polynomial kernel of degree 3 and a regularization parameter of C=100:

```
    model = svm.SVC(kernel='poly', degree=3,
        C=100)
```

D. GOOGLE VERTEX AI

Google Cloud Platform's Vertex AI is a unified machine learning platform in the cloud. It allows the user to create managed tabular, image, text, or video datasets that can be used for AutoML or custom training jobs. AutoML is Vertex AI's automated machine learning model that can be trained, tested, and deployed without machine learning and coding experience. AutoML automates the selection, composition, and parameterization of machine learning models. The price to train an AutoML model in Vertex AI is $3.465 per node hour.

E. APACHE SPARK

Apache Spark is an analytics engine for big data processing. We can use it to run machine learning training jobs on clusters with implicit data parallelism and fault tolerance. In this work, we compare the classification models provided in Spark's Machine Learning library (MLlib). Before we train these models on the MRI image data, we first do a principal component analysis (PCA) with scikit-learn's PCA tool. We do this to reduce the dimensionality of the data from 36,608 features (178x208 pixels) to 1,000 features. Our analysis indicates that 1,000 principal components capture over 97% of the variance in the data, and Spark can handle this level of dimensionality much better.

The SparkML models we explore in this research are Decision Tree, Random Forest, Gradient Boosted Tree (GBT), Multilayer Perceptron, Linear SVM, One vs Rest, Naive

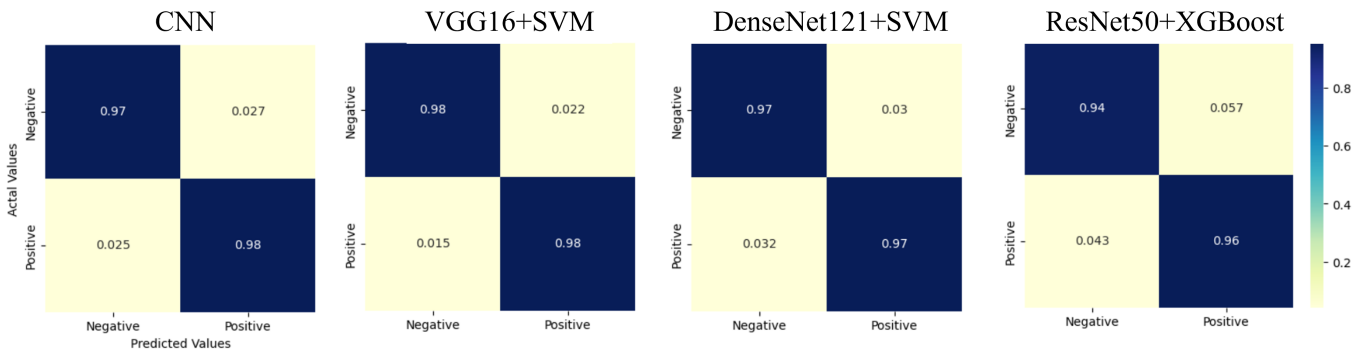

FIGURE 2. Confusion Matrices for CNN and Hybrid Models

Bayes, and Factorization Machines (FM) classifiers. Decision Tree classifier is a non-parametric supervised learning method. It learns simple decision rules that are inferred from the input features and uses these rules to continually split the dataset until it fits all of the data points. Random Forest classifier builds multiple decision trees on different samples of the data and uses averaging to build the best model with higher accuracy and less over-fitting of the data. GBT classifier is an additive model in which each stage fits regression trees onto the negative gradient of the loss function. In binary classification such as this, only a single regression tree is used.

Multilayer Perceptron is a fully connected class of feed-forward artificial neural network. It has an input layer, an output layer, and any number of hidden layers in between. Linear SVM is the same as described under the "Hybrid Models" section of "CNN and State-of-the-art Models". One vs Rest typically splits a multi-class dataset into multiple binary classification problems and then uses a binary classifier to solve the problem. In our case of binary classification our One vs Rest model which uses Logistic Regression as its classifier equates to a Logistic Regression model. Naive Bayes classifier is a probabilistic model based on the Bayes theorem. Finally, FM classifier is a generalization of the Linear Regression model and the Matrix Factorization model. It is also similar to an SVM with a polynomial kernel. One of its strengths is its efficiency with high-dimensional, sparse inputs.

F. PRINCIPAL COMPONENT ANALYSIS

Principal component analysis is a dimensionality reduction technique often used in data science. It linearly transforms the data into a new coordinate system in which much of the variation in the data is described in fewer dimensions than the original data. In our case, where each data point consists of 36,608 features, there are 36,608 principal components. Our analysis can show us how much of the explained variance in the data is described by each principal component, and we can use this information to choose how many principal components to use as input to our classification problem.

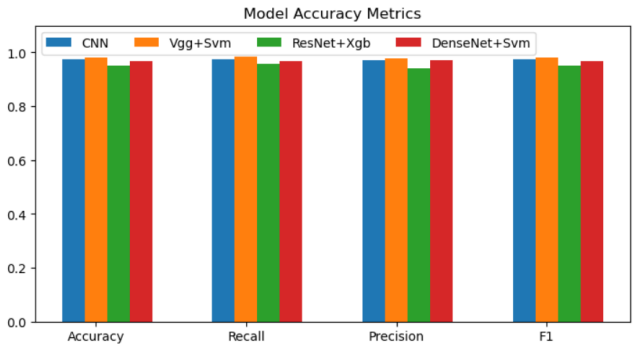

FIGURE 3. Accuracy Metrics for CNN and Hybrid Models

Model	Accuracy	Recall	Precision	F1 Score	Fall Out
CNN	0.97437	0.97512	0.97291	0.97402	0.02708
VGGsvm	0.98156	0.98492	0.97804	0.98147	0.02195
DNsvm	0.95	0.95712	0.94282	0.94992	0.05717
RNxgb	0.96875	0.96806	0.97	0.96903	0.03

TABLE 2. CNN and State-of-the-art Model Metrics. VGG is VGG16, DN is DenseNet121, and RN is ResNet50.

V. RESULTS

A. TENSORFLOW CNN

The CNN model was trained using an 80/20 train/test split, 5-fold cross-validation with 10 epochs for each training session, and a batch size of 32. It had an accuracy of 97.44%, recall of 97.51%, precision of 97.29%, F1 score of 97.40%, and fall out of 0.02708. See Figure 2 for the model's confusion matrix.

B. HYBRID MODELS

The top three hybrid models were VGG16 with an SVM classifier, DenseNet121 with an SVM classifier, and ResNet50 with an XGBoost classifier. The VGG16+SVM model was trained and tested with 5-fold cross-validation and an 80/20 train/test split. It had an accuracy of 98.16%, recall of 98.49%, precision of 97.80%, F1 score of 98.15%, and fall

out of 0.02196. The DenseNet121+SVM model had an accuracy of 96.88%, recall of 96.81%, precision of 97%, F1 score of 96.90%, and fall out of 0.03. The ResNet50+XGBoost model had an accuracy of 95%, recall of 95.71%, precision of 94.28%, F1 score of 94.99%, and fall out of 0.05717. See Figure 2 for each model's confusion matrix. A summary of these results can be found in Table 2 and visualized in Figure 3.

C. VERTEX AI

To create the Vertex AI managed dataset, we first uploaded all of our images to a GCP bucket. Uploading 6,400 images took about an hour. Then, we can create a dataset on the VertexAI page. This requires a csv file containing the path to the image and its label for each image. This looks like:

```
gs://az-bucket-1/Negative/neg(1).jpg, 0
```

When training a Vertex AI AutoML model, a maximum number of training hours can be specified to prevent high billing. The minimum number of hours is 8 hours, so this is what we chose. We also enabled early stopping so that if less than 8 hours of training was needed, it would stop early. After 2 hours and 8 minutes of training, VertexAI reported an average precision of 91.7%, a precision of 83.6% and a recall of 83.6%. This leg of training was billed for 8 hours at $3.465/hour for a total of $27.72. After another hour and 48 minutes of training, VertexAI reported an average precision of 99.5%, a precision of 95.8% and a recall of 95.8%. This leg of training was billed for another $27.72. See Table 3 and Figure 4 for a summary of these results.

Total Time	Avg. Precision	Precision	Recall
2h8m	91.7%	83.6%	83.6%
3h56	99.5%	95.8%	95.8%

TABLE 3. VertexAI AutoML Results

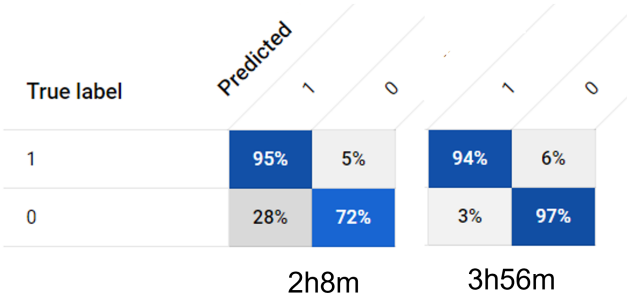

FIGURE 4. Confusion Matrices for VertexAI AutoML Models

D. CPU VS GPU

In order to test CPU vs GPU computing resources, and because of the limited time frame, we use the pure CNN model and the VGG16 with XGBoost Classifier model, both of which are easily configurable to run on GPU. The CNN model is built with TensorFlow which will transparently run on a single GPU with no code changes required. XGBoost Classifier will also run on a single GPU with the following code change:

```
model =
xgb.XGBClassifier(tree_method='gpu_hist')
```

To test, we use Google Colab's standard NVIDIA T4 Tensor Core GPU. We compare the training time for each model with its CPU training time and check that the resulting accuracy metrics are comparable. The CPU used for all other experiments is an Intel(R) Core(TM) i7-10510U CPU. The CNN model does both feature extraction and model training with 5-fold cross-validation and takes 69 minutes and 21 seconds on the CPU and 2 minutes 53 seconds on the GPU. The VGG16 with XGBoost Classifier model simply passes VGG16 fitted weights to the XGBoost model for training. The XGBoost model took 15 minutes and 57 seconds to train on the CPU and 2 minutes and 33 seconds to train on the GPU. Therefore, we can see significant speedups in using GPU resources over CPU resources, especially when it comes to feature extraction models. A visual comparison of these training times can be seen in Figure 5.

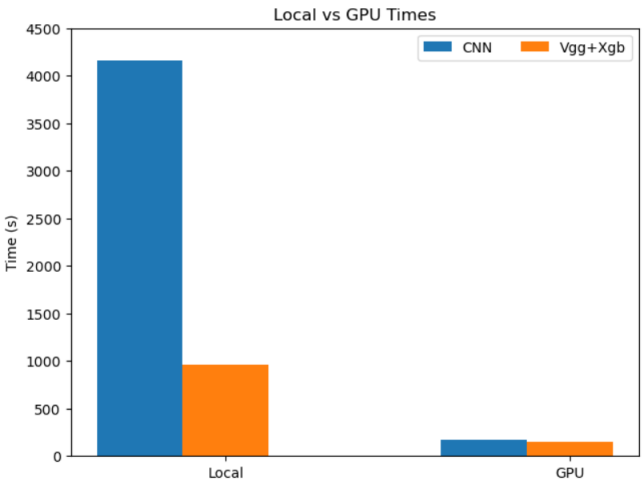

FIGURE 5. Training Time on CPU vs GPU for CNN and VGG16+XGB Models

E. APACHE SPARK MLLIB MODELS ON PCA DATA

The results of the PCA can be seen in Table 4. The "Explained Ratio" column indicates what the ratio of explained variance in the data is for each principal component. The principal components are ordered by explained ratio descending. The "Explained Ratio Sum" is a cumulative total of the explained ratio. We can see that only 1,000 principal components are needed to explain more than 97% of the

PC	Explained Ratio	Explained Ratio Sum
0	0.221449	0.221449
1	0.105193	0.326642
2	0.078420	0.405062
3	0.032906	0.437968
...	...	...
996	0.000043	0.970182
997	0.000042	0.970224
998	0.000042	0.970266
999	0.000042	0.970308

TABLE 4. PCA Explained Variance

Model	Mean Accuracy
Decision Tree	69.05 ±1.04
Random Forest	67.56 ±1.71
Gradient Boosted Tree	73.77 ±2.93
Multilayer Perceptron	98.16 ±0.20
Linear SVM	91.86 ±0.74
One-vs-Rest	94.01 ±0.46
Naive Bayes	86.58 ±0.98
FM Classifier	99.02 ±0.20

TABLE 5. Spark Accuracies

variance in our dataset. We choose this number as a cutoff point to reduce the dimensionality of our data from over 36,000 features to 1,000 features.

The 1,000 principal components are stored in a csv file and used as input into most of the machine learning models SparkML has in its library. Each model is trained and tested five times with random subsampling and an 80/20 train/test split. This allows us to take an average accuracy metric with standard deviation. The Random Forest model uses a number of trees of 10. GBT classifier uses a max iteration of 10. The Multilayer Perceptron model uses an input layer of size 1000 (number of principal components), an intermediate layer of size 512, an intermediate layer of size 256, and an output layer of size 2 (number of classes). It uses a max iteration of 100, a block size of 128, and the seed 1234. The SVM uses a max iteration of 10 and a regularization parameter of 0.1. The One vs Rest model uses a logistic regression classifier with a max iteration of 10, a tol of 1E-6 and fitIntercept=True. Naive Bayes required the data values to only be positive, so it was first run through a MinMaxScaler. Then it uses smoothing at 1.0 and a multinomial model type. Finally, FM Classifier uses a step size of 0.001.

The mean accuracy results can be found in Table 5 and are visualized in Figure 6. A graph of the time each model took can be found in Figure 7. Finally, a comparison of the model accuracies vs their times can be found in Figure 8.

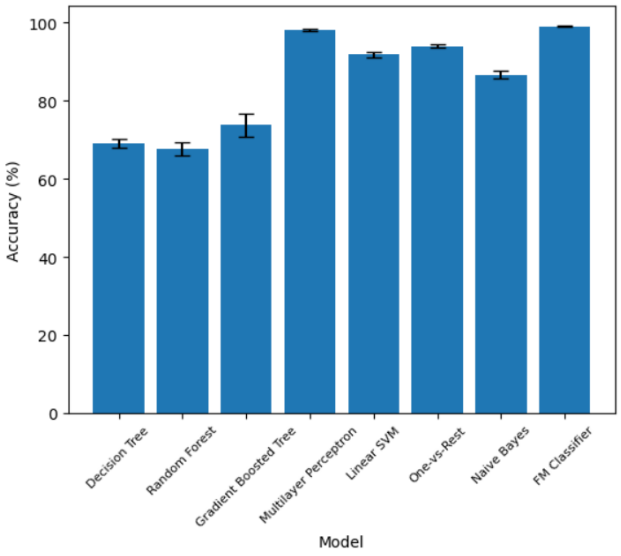

FIGURE 6. Mean Accuracies from Table 3

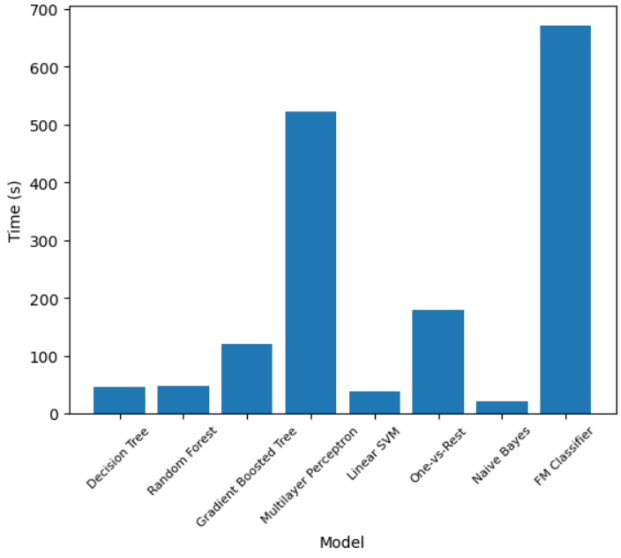

FIGURE 7. Times for Apache Spark Machine Learning Models: Totals for 5 Runs with Random Subsampling

VI. CONCLUSIONS

Of all the models tested, the three with the highest accuracy metrics are the VGG16+SVM model, the VertexAI AutoML model, and the PCA+Spark FM Classifier models. The fact that VGG16 performed better than ResNet50 and DenseNet121 state-of-the-art models indicates that this type of problem and data does not require a deep and complicated architecture. Even the purely CNN model was comparable with ResNet50+XGBoost and DenseNet121+SVM, while being a much more shallow architecture. The advantage of using these state-of-the-art models is their accessibility, proven performance, and speed.

The advantage of VertexAI AutoML is that any hospital or research institute can use it without needing Machine Learning or coding experts. They can use their own set of MRI

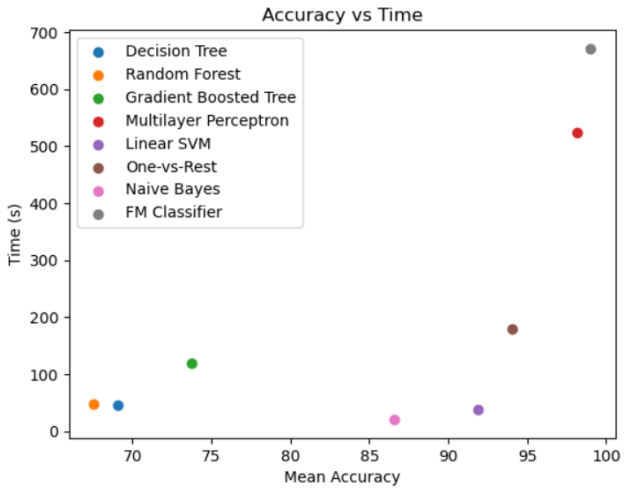

FIGURE 8. Time vs Accuracy for Apache Spark Machine Learning Models

images and diagnoses for their managed VertexAI dataset. They can train an AutoML model at $3.465/node hour, they can deploy and make online predictions for $1.375/node hour, and they can make batch predictions for $2.222/node hour.

The advantage of using PCA and SparkML is its implicit parallelism and fault tolerance. It is very quick to run a training job on a local machine if GPU resources are not available. With 97% of the variance in the data still intact using 1,000 principal components, the FM Classifier had over a 99% average accuracy across 5 training jobs. We stated previously that FM Classifier is similar to an SVM with a polynomial kernel and that it does well for sparse data. We used a polynomial kernel SVM with our VGG16 data and the original data is sparse, so that may indicate why the two models are comparable.

Please see Table 6 for a final comparison of the models found in the literature and the proposed models with over 90% accuracy.

Models	Accuracies
9-layer CNN [10]	97.6%
12-layer CNN [7]	97.75%
"All Convolutional Network" 3D CNN [6]	99%
Jacobian Map Feed CNN [9]	96.61%
LeNet-5 [8]	96.85%
DenseNet121 [1]	91%
ResNet50 + Softmax, ResNet50 + SVM, ResNet50 + RF [3]	99%, 92%, 85.7%
DenseNet201 + Gaussian NB, DenseNet201 + XG Boost, DenseNet201 + SVM [4]	91.75%, 91.13%, 91.03%
Deep Bolzmann Machine [5]	95.35%
SVM [2]	84%
Proposed CNN	97.44%
Proposed VGG16 + SVM	98.16%
Proposed DenseNet121 + SVM	95%
Proposed ResNet50 + XGBoost	96.88%
Proposed VertexAI AutoML	99.5%
Proposed PCA + Multilayer Perceptron	98.16%
Proposed PCA + FM Classifier	99.02%
Proposed PCA + One vs Rest	94.01%
Proposed PCA + Linear SVM	91.86%

TABLE 6. Accuracies of the Literature vs Proposed Models

VII. FUTURE WORK

In future work, the goal is to create a custom training job in VertexAI to test our CNN architecture in the VertexAI environment. Additionally, we aim to experiment with other hybrid models and to do more analysis on the model performances in comparison.

REFERENCES

[1] D. Chaihtra and S. Vijaya Shetty, "Alzheimer's Disease Detection from Brain MRI Data using Deep Learning Techniques," 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2021, pp. 1-5, doi: 10.1109/GCAT52182.2021.9587756.

[2] Uma R. K, Sharvari S. S, Umesh M. G and Vinay B. C, "Binary Classification of Alzheimer's disease using MRI images and Support Vector Machine," 2021 IEEE Mysore Sub Section International Conference (MysuruCon), Hassan, India, 2021, pp. 423-426, doi: 10.1109/MysuruCon52639.2021.9641661.

[3] D. AlSaeed and S. F. Omar, "Brain MRI Analysis for Alzheimer's Disease Diagnosis Using CNN-Based Feature Extraction and Machine Learning," Sensors, vol. 22, no. 8, Art. no. 8, Jan. 2022, doi: 10.3390/s22082911.

[4] S. Sharma et al., "HTLML: Hybrid AI Based Model for Detection of Alzheimer's Disease," Diagnostics (Basel), vol. 12, no. 8, p. 1833, Jul. 2022, doi: 10.3390/diagnostics12081833.

[5] H.-I. Suk, S.-W. Lee, and D. Shen, "Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis," NeuroImage, vol. 101, pp. 569–582, Nov. 2014, doi: 10.1016/j.neuroimage.2014.06.077.

[6] S. Basaia et al., "Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks," NeuroImage: Clinical, vol. 21, p. 101645, Jan. 2019, doi: 10.1016/j.nicl.2018.101645.

[7] E. Hussain, M. Hasan, S. Z. Hassan, T. Hassan Azmi, M. A. Rahman, and M. Zavid Parvez, "Deep Learning Based Binary Classification for Alzheimer's Disease Detection using Brain MRI Images," in 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), Nov. 2020, pp. 1115–1120. doi: 10.1109/ICIEA48937.2020.9248213.

[8] S. Sarraf and G. Tofighi, "Deep learning-based pipeline to recognize Alzheimer's disease using fMRI data," in 2016 Future Technologies Conference (FTC), Dec. 2016, pp. 816–820. doi: 10.1109/FTC.2016.7821697.

[9] S. Qasim Abbas, L. Chi, and Y.-P. P. Chen, "Transformed domain convolutional neural network for Alzheimer's disease diagnosis using structural MRI," Pattern Recognition, vol. 133, p. 109031, Jan. 2023, doi: 10.1016/j.patcog.2022.109031.

[10] M. Mamun, S. Bin Shawkat, M. S. Ahammed, M. M. Uddin, M. I. Mahmud, and A. M. Islam, "Deep Learning Based Model for Alzheimer's Disease Detection Using Brain MRI Images," in 2022 IEEE 13th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), Oct. 2022, pp. 0510–0516. doi: 10.1109/UEMCON54665.2022.9965730.

[11] "Alzheimer's Dataset (4 class of Images)." https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images (accessed Feb. 20, 2023).