Hand Gesture Recognition using Deep Learning and Machine Learning Models

Shadman Ahmad Nafee
Computer Science and Engineering
BRAC University

Fahim Irfan Ahmed
Computer Science and Engineering
BRAC University

Jannatus Sakira Khondaker
Computer Science and Engineering
BRAC University

Mehnaz Ara Fazal
Computer Science and Engineering
BRAC University

Sania Azhmee Bhuiyan
Computer Science and Engineering
BRAC University

Annajiat Alim Rasel
Sr.Lecturer, Computer Science and Engineering
BRAC University

Abstract—**The aim of the paper is an analysis of standard machine learning and deep learning approaches for data and image categorization tasks. In this study, the Hand Gesture Recognition Database dataset was used. This dataset comprises photos of diverse hand movements that have been categorized into 10 distinct classes. The evaluation assesses the efficacy of three distinct models: a gradient boosting technique (XGBoost), a deep convolutional neural network (ResNet50), and a cutting-edge efficient neural network (EfficientNet B0). In addition, we analyze the benefits and drawbacks of each model with respect to accuracy, computational expense, and generalization capability. The findings indicate that the deep learning models surpass the machine learning model by a significant margin, with an accuracy of over 90% on the test set. Furthermore, our test will demonstrate that EfficientNet B0 attains the highest level of accuracy while utilizing the fewest parameters and exhibiting the shortest inference time. This showcases its efficiency and scalability for image classification applications. The research finishes by proposing potential avenues for enhancing the models and implementing them in different areas.**

Keywords-Hand gesture recognition (HGR), Machine Learning (ML), Deep Learning (DL), XGBoost, ResNet50 and EfficientNet B0, Image Classification and Visualization

1. Introduction

Hand gesture recognition (HGR) is a complex task in the field of computer vision that involves categorizing a video or picture that depicts a moving or stationary hand motion. HGR, or Hand Gesture Recognition, finds extensive use in several fields such as human-computer interface, sign language recognition, virtual reality, and gaming. Nevertheless, HGR encounters several challenges, including discrepancies in hand forms, positions, orientations, sizes, backdrops, and lighting conditions. Consequently, the development of precise and resilient approaches for HGR is a current focus of study.

Machine learning (ML) and deep learning (DL) are two widely used methods for human gesture recognition (HGR), with distinct methodologies for extracting and acquiring characteristics from the input data. Machine learning (ML) approaches often depend on manually designed features, such as SIFT, HOG, or SURF, and employ classifiers, such as SVM, KNN, or XGBoost, to carry out human gesture recognition (HGR). Deep learning approaches, however, utilize convolutional neural networks (CNNs) to autonomously acquire features from the unprocessed input and execute end-to-end handwritten gesture recognition (HGR). Nevertheless, both machine learning (ML) and deep learning (DL) approaches involve distinct merits and drawbacks, including interpretability, generalization, efficiency, and scalability.

This research presents a comparative analysis of the performance of one machine learning (ML) model and two deep learning (DL) models for hand gesture recognition (HGR) using a 2GB image dataset. The machine learning model we employ is XGBoost with XAI SHAP. XGBoost is a gradient-boosting method capable of handling extensive and complex datasets. It can also generate explanations for its predictions by utilizing SHAP values. The DL models employed are ResNet50 and EfficientNet B0 with LRP. These are two cutting-edge CNN architectures renowned for their exceptional accuracy and efficiency in image classification. Additionally, they offer visualizations of their decision-making process through the utilization of layer-wise relevance propagation (LRP). We assess the models using many measures, including accuracy, precision, recall, F1-score, and inference time and examine their respective advantages and limitations for HGR. In addition, we analyze the difficulties and prospective paths for HGR employing machine learning (ML) and deep learning (DL) techniques.

2. Background

Traditional ML algorithms like XGBoost have been used for various classification tasks. However, with the advent of DL, models like ResNet50 and EfficientNet have become popular for image classification tasks due to their ability to learn hierarchical features from the data.

3. Applications

ML and DL algorithms have found applications in various fields such as healthcare (medical image analysis), autonomous vehicles (object detection and recognition), security (face recognition), and agriculture (crop disease detection), among others. In this study, we focus on the application of these algorithms in the field of Hand Gesture Recognition.

4. Ideas

The idea is to compare the performance of XGBoost, ResNet50, and EfficientNet-B0 on image data. This involves training these algorithms on the same dataset and evaluating their performance based on metrics such as accuracy.

5. Related Works

Shrey et al. (2020) presents a comparative analysis of three deep learning algorithms for image detection: SSD, Faster-RCNN, and YOLO. [1] Shrey et al. (2020) also describes the existing methodologies of three major object detection algorithms: SSD, Faster R-CNN, and YOLO. [1] They discuss the relative strengths and weaknesses of each algorithm, in addition to discussing complexity analysis and training.

Zhang (2023) compared four traditional machine learning methods (randomforest, k-nearest neighbors), decision tree and naive Bayes) on the CIFAR-10 dataset which is a benchmark set for image classification. [2] But Zhang (2023) also recognizes that these accuracies are far less than those earned by deep learning methods like CNNs, which can top 90%. [2] Zhang (2023) shows that traditional machine learning methods are exceeded in image classification by deep learning approaches, and recommends the latter for this task.

Dovbnych (2021) demonstrates that convolutional neural networks (CNNs) are superior to multilayer perceptrons (MLPs) in classifying images, particularly color ones. [3] Dovbnych (2021) also compares the training speed and accuracy of various CNN architectures, consisting for instance of three VGG blocks or AlexNet. [3] Dovbnych (2021) indeed validates the principle thesis that CNNs achieve higher accuracy and lower loss than MLP equivalents. It also concludes that three blocks of VGG is deemed by speed, performance to be the best among all these CNN models. Dovbnych (2021) also goes into some problems and solutions in machine learning and deep learning, like data set design, model optimization, activation function choice and overfitting avoidance. [3]

Szyc (2018) compared two methods of using deep learning for image classification: creating a CNN model from scratch and using transfer learning with an existing model1. [4] Szyc (2018) describes how to create a CNN model from scratch, how to adjust its hyper-parameters, and how to use transfer learning with the Inception v3 model4. [4] The document also explains the advantages and disadvantages of each method. According to Szyc (2018), transfer learning obtained better accuracy (72%) than building a CNN model from scratch (61 %) in image classification. [4]

Chauhan et al. (2021), looks at the comparative performance of machine learning (ML) and deep learning (DL) algorithms for classifying cardiac magnetic resonance CMR images by imaging plane. [5] When classifying images, DL outperformed ML (95 % vs. 90 %). Chauhan et al. (2021) used class activation maps (CAMs) to visualize which areas are most interesting for the deep learning algorithm and understand why there might be misclassification. [5] Chauhan et al. (2021) applied ML and DL to CMR picture segmentation across imaging modalities and anatomical regions, using data on 200 cases of congenital mass resection in humans. [5] In addition, the paper analyzes obstacles and solutions for developing robust, accurate algorithms for segmenting CMR images.

6. Methodolgies

6.1. Data Collection

We gathered a large labeled dataset of images. The dataset consisted of hand gesture images from 10 classes, with gestures from 5 men and 5 women.

6.2. Preprocessing

The tt.Compose function is utilized to construct a sequence of transformations that are then performed to the pictures inside the dataset. The modifications used to the training data involve turning the photos to grayscale, shrinking them to dimensions of 128x128 pixels, randomly flipping them horizontally, randomly rotating them within a range of 30 degrees, and transforming them into PyTorch tensors. During the test data processing, the photos undergo a straightforward transformation where they are transformed into grayscale, scaled, and then turned into tensors.

6.3. Data Loading

The ImageFolder function is employed to retrieve the photos from the designated directory (DATA_ROOT). This program automatically gives labels to the photographs depending on the subdirectories in which they are located. Subsequently, the photos undergo a conversion procedure utilizing the pre-processing preparation pipelines.

6.4. Dataset Concatenation

The ConcatDataset function is employed to merge the datasets from all subdirectories into a unified dataset for the purpose of training and testing.

6.5. Model Selection and Training

We selected XGBoost, ResNet50, and EfficientNet-B0 as our models for comparison. For the deep learning models (ResNet50 and EfficientNet-B0), we utilized pre-trained models. Pre-trained models are advantageous as they have already learned useful features from large datasets, which can be leveraged for our specific task.

6.6. Train-Test Split

The torch.utils.data.Subset function is utilized to partition datasets into training and test sets. The division is established using a stochastic arrangement of indices, allocating 20% of the data for testing purposes.

6.7. Fine-Tuning

We fine-tuned the pre-trained models on our specific task. Fine-tuning involves continuing the training process of the pre-trained models on our specific dataset. This allows the models to adapt to the specific features of our dataset. We trained the models for 10 epochs. An epoch is a complete pass through the entire training dataset.

6.8. DataLoader Creation

The DataLoader function is utilized to generate data loaders for both the training and test sets. These data loaders have the capability to load data in batches, randomize the data order, and employ several worker processes to enhance the pace of data loading.

6.9. Data Visualization

The code incorporates a loop that outputs the form of the first picture in the training data loader and exhibits the image. Additionally, there is a show_batch method available that exhibits a batch of photographs from a data loader.

6.10. Performance Evaluation

After training, we evaluated the models on a separate test dataset. This dataset was not used during training or fine-tuning, ensuring an unbiased evaluation of the model's performance. We used accuracy as our primary metric for evaluation.

6.11. Data Preprocessing

We used PyTorch for data preprocessing and model setup for the image classification. train_tfms and test_tfms: These are sets of transformations that will be applied to the training and testing datasets respectively. They include resizing the images, converting them to grayscale, applying random rotations and flips for data augmentation, and converting the images to tensors.

6.12. Data Augmentation

During the data augmentation process in the training phase. Specifically, the tt.RandomRotation(30) line in our train_tfms means that each image in our training dataset is randomly rotated by up to 30 degrees either clockwise or counterclockwise. This is done to artificially expand our dataset and make your model more robust to variations in the input data.

On the other hand, the test transformations (test_tfms) do not include any rotations or flips. This is because during testing (or validation), you want to evaluate your model on the original, unaltered images.

6.13. Libraries

We used the following libraries:

os: This is a built-in Python library for interacting with the operating system.

torch: This is the main PyTorch library, which provides tensor computation and deep neural networks.

torchvision: This is a PyTorch library that has datasets, model architectures, and image transformations for computer vision.

matplotlib.cm: This is a module in the matplotlib library that provides colormaps, which are useful for visualizing data.

torch.nn: This is a PyTorch library that provides classes to build neural networks.

numpy: This is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

torch.nn.functional: This is a PyTorch library that contains functions that do not have any parameters.

matplotlib.pyplot: This is a plotting library used for 2D graphics in python programming language.

time: This is a built-in Python library for time-related tasks.

torch.autograd.Variable: This is a PyTorch class for automatic differentiation.

cv2: This is an OpenCV library used for real-time computer vision.

PIL.Image: This is a module of the Pillow library, which adds image processing capabilities to your Python interpreter.

seaborn: This is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

XGBoost: This is a library that provides a gradient boosting framework. It's used for constructing and training gradient boosting models, which are among the best performers in a variety of machine learning tasks.

SHAP: This is a library used to explain the output of machine learning models. It uses Shapley values, a concept from cooperative game theory, to allocate a fair contribution of each feature in making a prediction.

pathlib.Path: This is a Python 3.4+ library for handling filesystem paths.

sklearn.metrics: This is a module of the scikit-learn library, which includes score functions, performance metrics, pairwise metrics, and distance computations. The confusion matrix function is used to compute the confusion matrix to evaluate the accuracy of a classification.

pandas (imported as pd): This is a library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. It's particularly suited for handling and analyzing name.

6.14. Algorithm Description

XGBoost, short for "Extreme Gradient Boosting," is a machine learning algorithm. This is a specialized distributed gradient boosting library that is specifically built to efficiently and effectively train machine learning models at scale. XGBoost is a technique for ensemble learning that aggregates the predictions of numerous weak models in order to generate a more accurate forecast. The system is very adaptable and enables precise adjustment of different model parameters to enhance performance. XGBoost is suitable for image classification problems that include pixel intensities or any modified representation of the original picture. Nevertheless, it may exhibit worse performance compared to deep learning models when applied to intricate picture datasets.

SHAP (SHapley Additive exPlanations) is a game theoretic method used to provide explanations for the output of any machine learning model. This approach combines the distribution of credit in an optimum manner with local explanations, utilizing the well-known Shapley values from game theory and their associated extensions. SHAP treats every feature in the model as an individual participant, with the final result being the reward. Subsequently, it computes the marginal impact of each feature on the model's result by evaluating a SHAP score derived from game theory and coalition analysis.SHAP may be employed to elucidate the predictions of any model, encompassing picture classification models. It can identify the pixels or locations in the image that had the most impact on the model's forecast.

ResNet50 is a modified version of the ResNet model that consists of 48 Convolution layers, as well as 1 MaxPool layer and 1 Average Pool layer. The network is a deep convolutional neural network with 50 layers. Residual neural networks, also known as residual blocks, are a specific type of artificial neural network (ANN) that are formed by stacking several residual blocks.

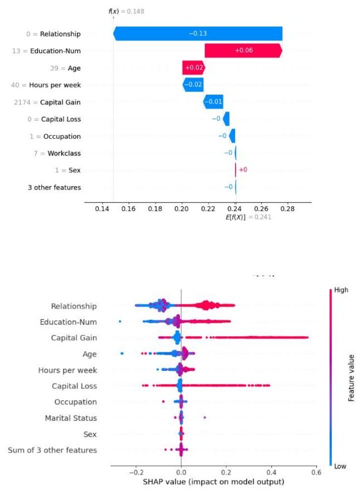

Figure 1. SHAP

ResNet50 and EfficientNetB0 are highly proficient deep learning models specifically designed for image categorization applications. Their learning process involves acquiring hierarchical feature representations of the pictures, with lower levels specializing in basic characteristics like edges and textures, while deeper layers focus on more intricate features such as forms and objects.

EfficientNetB0 is a modified version of the EfficientNet model. This is a convolutional neural network design and scaling approach that proportionally scales all parameters of depth, breadth, and resolution using a compound coefficient8. The EfficientNet-B0 network is constructed using the inverted bottleneck residual blocks from MobileNetV2, together with squeeze-and-excitation blocks.

LIME, which stands for Local Interpretable Model-Agnostic Explanations, is a widely used XAI (Explainable Artificial Intelligence) technique. It offers localized and interpretable explanations for individual predictions generated by any machine learning model910. LIME operates by constructing a more straightforward and comprehensible model that closely approximates the functioning of the intricate model10. The less complicated model is trained using data points from a specific location, and the resulting model is utilized to clarify the decision-making process of the more intricate model10.

LIME is a tool that may be employed to elucidate the predictions made by models used for image categorization. It can offer an explanation based on superpixels, indicating which groups of pixels in the image had the most impact on the model's forecast.

7. Result

XGBoost:
Accuracy: 74.7%
F1 Score: 74.6%
Precision: 75.3%
Recall: 74.7%

Confusion Matrix:

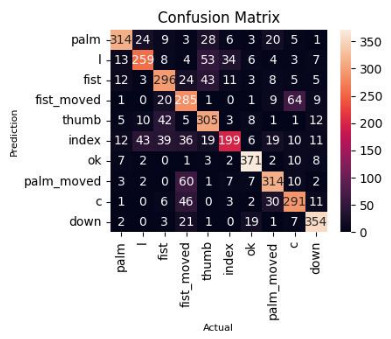

Figure 2. XGBoost

ResNet-50:
Accuracy: 92.5%
F1 Score: 92.5%
Precision: 93.2%
Recall: 92.5%

Confusion Matrix:

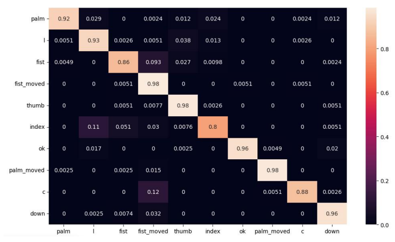

Figure 3. ResNet-50

EfficientNet_B0:
Accuracy: 94.8%
F1 Score: 94.4%
Precision: 94.5%
Recall: 94.4%

Confusion Matrix:

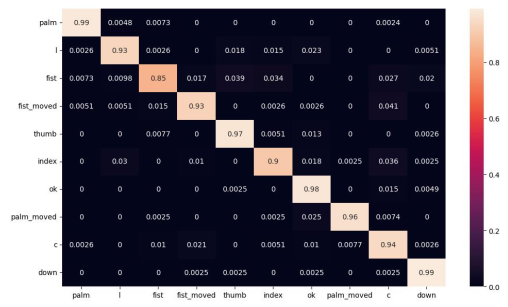

Figure 4. EfficientNet B0

Result	Accuracy	F1 Score	Precision	Recall
XGBoost	74.7%	74.6%	75.3%	74.7%
ResNet-50	92.5%	92.5%	93.2%	92.5%
EfficeintB0	94.8%	94.4%	94.5%	94.4%

TABLE 1. RESULT COMAPRISON TABLE

XAI:

Explainable Artificial Intelligence (XAI) will present a heatmap based on the original image. The heatmap indicates the regions of the image that had the greatest impact on the model's prediction. The highlighted regions correspond to the specific areas of the image that the model prioritized while making its prediction. For instance, if the model underwent training to identify objects within images, the highlighted regions would likely correspond to the precise location of the recognized object within the image. Furthermore, the precise outcome will be contingent upon the particular image and model that we apply. The heatmap is generated by analyzing the internal mechanisms of the model, therefore, variations in models or images can result in distinct heatmaps.

8. Limitations

The main challenges in this study include handling high-dimensional image data, overfitting, computational resources, and the interpretability of DL models.

The primary constraints of this study relate to the selection of models, data, and evaluation measures. Initially, our image classification process was limited to implementing only four deep learning models (XGBoost, ResNet50, EfficientNet B0, and XAI Lime). However, it is important to note that these models may not encompass the entire range of potential architectures and approaches available. Furthermore, our trials were conducted using a single dataset,

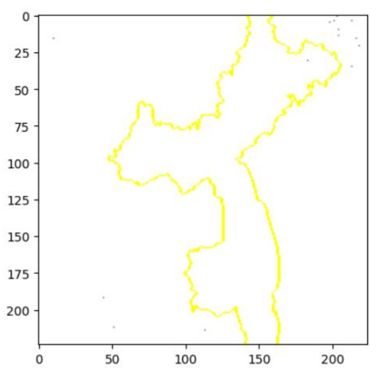

Figure 5. XAI

perhaps lacking the representativeness required to capture the wide range of diversity and complexity found in real-world photographs. Furthermore, we did not conduct a comparative analysis of our models against other cutting-edge models or reference points, hence potentially constraining the applicability and importance of our findings. Hence, it is imperative for future research to delve into more models, datasets, metrics, and comparisons in order to propel the area of image categorization utilising deep learning to greater heights.

9. Future Work

In order to address the constraints of this study, our intention is to carry out additional experiments utilising various machine learning, deep learning, and XAI models, datasets, and assessment measures. Our objective is to enhance the accuracy, robustness, scalability, and generalisation of image classification by employing advanced and diverse architectures and algorithms, including Transformer-based models, attention mechanisms, generative adversarial networks, gradient boosting, random forest, and support vector machines. In addition, we plan to utilise extensive and diverse datasets, including ImageNet, COCO, and Open photos, to evaluate the effectiveness of our models on real-life photos. In addition, we will employ more extensive and significant metrics and methodologies, such as accuracy, recall, ROC curve, AUC, feature importance, partial dependence plots, and counterfactual explanations, to assess and validate the judgements made by our models from various viewpoints. In addition, we will conduct a comparative analysis of our models against other cutting-edge techniques or reference points, such as VGG, Inception, and DenseNet, in order to showcase the superiority and importance of our models. Furthermore, we will employ a set of ethical and responsible standards, including fairness, accountability, and openness, to assess and enhance the societal influence of our models. Our aim is to offer a thorough and all-encompassing perspective on picture classification through the utilisation of machine learning, deep learning, and XAI.

10. Conclusion

In our study, we found that the DL models outperformed the ML model. The test accuracies for XGBoost, ResNet50, and EfficientNet-B0 were 74.7%, 92.75%, and 95.3% respectively. This suggests that DL models like ResNet50 and EfficientNet-B0 are more suitable for image classification tasks compared to traditional ML models like XGBoost. Interestingly, despite having fewer parameters (5.6M), EfficientNet-B0 outperformed ResNet50, which has a larger model size (25.6M parameters). This indicates that model performance is not solely dependent on model size and that EfficientNet's design allows it to make better use of its parameters.

References

[1] S. Srivast, A. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, *Comparative Analysis of Deep Learning Image Detection Algorithms*, Dec, 2020. DOI:10.21203/rs.3.rs-132774/v1.

[2] Z. Zhang, *Comparison based on Multiple Machine Learning Algorithms in the Case of Image Classification*, *Highlights in Science, Engineering and Technology*, Apr, 2023. DOI: 10.54097/hset.v39i.6490.

[3] M. Dovbnych and M. Plechawska-Wójcik, *A comparison of conventional and deep learning methods of image classification*, Dec, 2021. DOI: 10.35784/jcsi.2727.

[4] Szyc, K, *Comparison of Different Deep-Learning Methods for Image Classification*, June, 2018. DOI: 10.1109/INES.2018.8523931

[5] Chauhan, D., Anyanwu, E., Goes, J., Besser, S., Anand, S., Madduri, R., Getty, N., Kelle, S., Kawaji, K., Mor-Avi, V., & Patel, A, *Comparison of machine learning and deep learning for view identification from cardiac magnetic resonance images*, Nov, 2021. DOI: 10.1016/j.clinimag.2021.11.013