Comparative Analysis of Deep learning algorithms for Clinical Diagnosis

Aneesh Tufchi
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, Maharashtra-411043

Shrihari Tiwari
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, Maharashtra-411043

Manshu Khajuria
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, Maharashtra-411043

Kshiteej Pawar
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, Maharashtra-411043

Prof. Virendra Bagade
Asst. Professor, Department of Computer Engineering
Pune Institute of Computer Technology
Pune, Maharashtra-411043

Abstract

This research paper presents a detailed comparative analysis of deep learning models applied to the classification of medical images, specifically focusing on brain MRI scans. Accurate and automated classification of brain MRI images is crucial in diagnosing and staging neurodegenerative diseases such as Alzheimer's, and this study investigates the performance of four cutting-edge deep learning architectures: Convolutional Neural Networks (CNN), Visual Geometry Group Networks (VGG), Residual Networks (ResNet), and Capsule Networks (CapsNet), with an emphasis on the utilization of Region of Interest (RoI) techniques to enhance classification accuracy.

Each of these models brings unique strengths to the task of medical image classification. CNNs, with their layered convolutional operations, excel in learning hierarchical spatial features critical for identifying patterns in brain tissue. VGG networks, known for their deep architecture with small 3x3 filters, allow for more detailed feature extraction, enabling them to capture subtle differences in brain structures associated with various disease stages. ResNet, with its innovative skip connections, overcomes the vanishing gradient problem in deep networks, making it highly effective for learning complex patterns in large, multi-layered architectures. CapsNet, on the other hand, is a relatively new model that maintains the spatial relationships between features, which is particularly important in medical images where anatomical structures vary in orientation and position.

The models are rigorously evaluated across several key metrics, including classification accuracy, precision, recall and F1-score . The study also focuses on the models' ability to generalize across different datasets and on computational efficiency, essential factors when considering real-world clinical deployment. Preprocessing techniques such as image normalization, resizing, and data augmentation (including rotation, flipping, and zooming) are employed to optimize the models' performance and prevent overfitting, particularly in situations where the dataset size is limited.

Through this work, we contribute to the growing body of knowledge in the field of medical image analysis, offering insights into the practical applications of deep learning in healthcare.

Index Terms

CNN ,VGG ,ResNet, CapsNet ,Image Classification, Deep Learning, RoI, GANET , SBO ,ESBO

I. Introduction

In modern medical diagnostics, accurate and efficient image classification has emerged as a critical tool for the early detection and staging of diseases. Among various medical imaging modalities, brain MRI (Magnetic Resonance Imaging) plays a vital role in diagnosing neurodegenerative disorders such as Alzheimer's disease, brain tumors, and multiple sclerosis. These diseases often present complex patterns of structural changes in the brain, making manual interpretation challenging and time-consuming, especially when attempting to detect early-stage abnormalities that may be subtle and difficult to identify. Therefore, the need for automated and precise image classification systems has become increasingly pressing, as these systems can significantly improve diagnostic accuracy and reduce the workload on radiologists.

The application of deep learning techniques to medical image analysis, particularly for brain MRI scans, has gained substantial momentum in recent years. Deep learning, a subfield of machine learning, has revolutionized image recognition tasks by enabling

models to automatically learn hierarchical features from raw image data, eliminating the need for manual feature extraction. These models, when applied to medical images, can detect patterns that may not be immediately apparent to the human eye, leading to improved diagnosis, staging, and prognosis of various brain-related diseases.

Among the deep learning models, Convolutional Neural Networks (CNNs) are particularly well-suited for image classification due to their ability to capture spatial hierarchies in data. CNNs have been widely used for various medical imaging tasks, including lesion detection, tumor segmentation, and classification of brain diseases. However, as medical images, especially brain MRIs, present unique challenges such as the presence of noise, anatomical variations, and limited datasets, more advanced CNN architectures are necessary to improve classification accuracy and generalization.

VGG (Visual Geometry Group) networks, a deep convolutional network architecture, have proven effective in various image classification tasks. Their use of small filters (e.g., 3x3 kernels) and deep structures allows for more detailed feature extraction, capturing minute variations in medical images that could indicate early-stage diseases. VGG's depth enables it to detect fine-grained patterns, making it particularly useful for classifying complex medical datasets where small differences in image features can have significant diagnostic implications.

Residual Networks (ResNet) take deep learning a step further by addressing the problem of vanishing gradients, which commonly occurs in very deep networks. The introduction of residual connections allows the model to bypass certain layers, ensuring that the gradient flows effectively through the network. This feature is particularly important in medical image classification, where learning intricate, high-level features across many layers can be essential for accurate disease staging. ResNet's ability to train deep networks without performance degradation has made it one of the most successful architectures in medical image classification, especially for tasks that require distinguishing between early and advanced stages of diseases like Alzheimer's.

Capsule Networks (CapsNet) are an advanced form of neural networks designed to overcome the limitations of traditional Convolutional Neural Networks (CNNs). They were introduced by Sabour and Hinton in 2017, primarily focusing on tasks such as image classification and object detection. Unlike CNNs, which struggle with recognizing spatial hierarchies and relationships between parts of objects, CapsNet aims to model these relationships, improving generalization and recognition capabilities. For instance, a CNN might misclassify a jumbled bicycle with scattered wheels and handlebars as a bicycle, focusing only on the presence of the parts. In contrast, a CapsNet can learn the spatial relationships between those parts (like how the wheels should be positioned relative to the frame) and correctly identify that the object is not a properly assembled bicycle. Basically, a capsule network is a type of neural network that performs inverse graphics. For example, in object detection, the object is divided into subparts. To represent that object, a hierarchical relationship is developed between all subparts. The implementation of CapsNet is divided into three main parts. These parts are the input layer, hidden layer, and output layer.
In terms of security, capsule networks have been shown to be more robust against certain types of attacks compared to traditional CNNs.

The goal of this research is to evaluate and compare the performance of these four prominent deep learning architectures—CNN, VGG, ResNet, and CapsNet—on the task of classifying brain MRI images. We focus on key performance metrics such as accuracy, precision, recall and F1-score as well as the models' ability to generalize across different datasets. In addition to these metrics, computational efficiency is considered, as real-time diagnostic applications require not only accurate but also fast and resource-efficient models.

The importance of this research lies in its potential to advance automated medical diagnostics, particularly in the early detection and staging of neurodegenerative diseases. By providing a thorough comparison of different deep learning models, this study aims to highlight the strengths and limitations of each model and provide insights into which architectures are best suited for various types of medical image classification tasks. Furthermore, the findings from this research could pave the way for the integration of deep learning-based image classification systems in clinical practice, ultimately improving patient outcomes by enabling more accurate and timely diagnoses.

II. ROI Techniques in CNNs

A. ROI Pooling

ROI pooling is a technique designed to handle variable-sized input regions in CNNs, particularly useful for object detection frameworks like Faster R-CNN. The key steps involved in ROI pooling are:

- **ROI Definition:** An ROI is defined by a bounding box with coordinates $(x_{min}, y_{min}, x_{max}, y_{max})$.

- **Binning:** The bounding box is divided into a fixed number of bins (e.g., $H \times W$).
- **Max Pooling:** Max pooling is applied to each bin to generate a fixed-size output feature map.

The mathematical formulation for ROI pooling can be expressed as:

$$F_{ROI} = \text{Pooling}(F_{feature}, ROI) \tag{1}$$

where $F_{feature}$ is the feature map from the last convolutional layer.

B. ROI Align

ROI Align improves upon the limitations of ROI pooling by avoiding quantization errors. It performs bilinear interpolation on the feature map to ensure a better alignment with the original image pixels. This approach is critical for tasks requiring high precision, such as instance segmentation.

The mathematical representation of ROI align is given by:

$$F_{ROI} = \text{ROIAlign}(F_{feature}, ROI) \tag{2}$$

where the output size is maintained without introducing artifacts due to binning.

C. Implementation in VGG16

VGG16, characterized by its deep architecture consisting of 16 layers, employs ROI pooling as follows:

- **Feature Extraction:** Convolutional layers extract high-level features from the input image.
- **ROI Pooling:** After the last convolutional layer, ROIs are pooled into a fixed-size feature map.
- **Classification:** The pooled features are fed into fully connected layers for classification.

The ROI pooling step can be mathematically represented as:

$$F_{ROI} = \max_{i,j}(F_{feature}(x_i, y_j)) \quad \forall (x_i, y_j) \in ROI \tag{3}$$

This allows the model to effectively learn from localized features relevant to the defined ROIs.

D. Implementation in ResNet

ResNet introduces skip connections that alleviate the vanishing gradient problem, facilitating the training of deeper networks. The integration of ROIs in ResNet follows a similar flow:

- **Feature Extraction:** Features are extracted through several residual blocks.
- **ROI Align:** The ROI align method is applied to generate a fixed-size output feature map from the selected ROIs.
- **Classification and Regression:** The pooled features are then used for both classification and bounding box regression.

The use of ROI align can be expressed as:

$$F_{ROI} = \sum_{i=1}^{N} \text{BilinearInterpolate}(F_{feature}, (x_i, y_i)) \tag{4}$$

where N is the number of sampled points from the ROI, leading to a more accurate representation of the region.

III. ROI Segmentation

Region of Interest (ROI) Segmentation refers to the process of identifying and delineating specific regions within an image that are deemed significant for further analysis. This technique is crucial in various applications, including medical imaging, autonomous driving, and object detection. ROI segmentation allows for a more focused approach, improving both the accuracy and efficiency of subsequent tasks.

A. Mathematical Framework

The process of ROI segmentation can be mathematically framed as follows:

- **Input Representation:** Let I represent the input image and ROI be the region of interest defined by coordinates $(x_{min}, y_{min}, x_{max}, y_{max})$.
- **Feature Extraction:** The feature map $F_{feature}$ is generated through a series of convolutional layers:

$$F_{feature} = \text{CNN}(I) \tag{5}$$

- **ROI Pooling or Align:** After feature extraction, either ROI Pooling or ROI Align is applied:

$$F_{ROI} = \text{ROIAlign}(F_{feature}, ROI) \tag{6}$$

Here, F_{ROI} is the pooled feature map corresponding to the ROI.

- **Segmentation Mask Generation:** The segmentation mask S can be obtained using a pixel-wise classification approach, where each pixel in the ROI is assigned a label:

$$S = \text{argmax}(F_{segmentation}(F_{ROI})) \tag{7}$$

This function $F_{segmentation}$ represents a classifier that outputs probabilities for each class over the features extracted from the ROI.

B. Techniques Used in ROI Segmentation

- **Convolutional Neural Networks (CNNs):** CNNs are typically employed to extract spatial hierarchies of features from images. The architecture can include various layers, such as convolutional, pooling, and fully connected layers, tailored to segment objects from ROIs effectively.
- **Semantic Segmentation:** In semantic segmentation, every pixel is classified into a category. This is useful when the objective is to identify and delineate all objects within an ROI. Techniques like U-Net or Fully Convolutional Networks (FCNs) can be utilized for this purpose.
- **Instance Segmentation:** Unlike semantic segmentation, instance segmentation differentiates between distinct objects of the same class. Models like Mask R-CNN extend traditional detection frameworks to include a segmentation mask for each detected instance. The output can be defined as:

$$M_i = F_{mask}(F_{ROI}) \quad \forall i \in \text{instances} \tag{8}$$

where M_i is the mask for the i-th instance.

IV. System Architechture

The system architecture for classifying medical images, specifically brain MRI scans, using deep learning models, consists of several interconnected layers that work together to preprocess the data, extract features, apply machine learning models, and present results. Each of these layers has a distinct role in ensuring the accuracy, efficiency, and scalability of the image classification system. Below is a detailed breakdown of the system architecture.

A. Data Collection Layer

The data collection layer forms the foundation of the image classification system. Medical images, particularly brain MRI scans, are sourced from a variety of repositories and databases, which could include:

1) Public datasets: Examples include the Alzheimer's Disease Neuroimaging Initiative (ADNI) or Medical Image Computing and Computer-Assisted Intervention (MICCAI) challenges, which provide labeled MRI images of healthy individuals and patients in various stages of neurodegenerative diseases.

2) Hospital databases: Clinically gathered data from hospitals, which may contain anonymized MRI scans of patients along with corresponding medical history and disease staging.

These datasets often contain labeled images representing different stages of diseases such as Alzheimer's (e.g., Mild Cognitive Impairment (MCI), Early Alzheimer's Disease (AD), or Healthy Control (HC)). The labeled data is crucial for supervised learning models like CNN, VGG, ResNet, and CapsNet. Challenges at this stage:

3) Data diversity: Datasets must cover various demographic groups, imaging techniques, and disease stages to ensure the models generalize well across different patient populations.

4) Data size: Medical datasets can be relatively small due to privacy concerns and the cost of collecting and annotating medical images, leading to issues such as overfitting. Therefore, augmentation and transfer learning are often necessary.

B. Data Preprocessing Layer

Once the MRI images are collected, they undergo several preprocessing steps to prepare them for input into the deep learning models. The data preprocessing layer is critical because the quality of the input data directly influences the performance of the models. Key tasks in this layer include:

1) Resizing: MRI images come in various sizes depending on the scanning equipment used. For consistency, all images are resized to a fixed resolution (e.g., 224x224 or 256x256 pixels). This standardization ensures uniform input dimensions for the models.

2) Normalization: The pixel values of MRI images can vary widely, depending on the scanning intensity. Normalization transforms these values into a standard range, such as [0,1] or [-1,1], which helps the neural network converge more quickly and improves numerical stability during training.

3) Data Augmentation: To combat the relatively small size of medical image datasets, data augmentation techniques are applied to artificially expand the dataset. Common augmentations include: o Rotation: Rotating the images by random angles to help the model learn rotational invariance. o Flipping: Horizontally or vertically flipping the images to create new training samples. o Zooming: Randomly zooming in and out to teach the model to handle images at different scales. o Shifting: Translating the images slightly to simulate variability in patient positioning during the MRI scan.

4) Image Slicing: MRI scans are typically 3D images composed of multiple 2D slices taken at different depths. Depending on the approach, these 2D slices can be used individually, or the 3D volume can be processed as a whole. For some deep learning models, individual slices are classified independently, while for others (such as 3D CNNs), the entire volume is processed to capture spatial relationships.

These preprocessing techniques ensure that the images are clean, standardized, and augmented, improving the models' ability to generalize and perform well on unseen data.

C. *Feature Engineering*

In traditional machine learning, feature engineering involves manually selecting and transforming features to improve model performance. However, in deep learning, particularly with CNNs and their variants, feature extraction is handled automatically by the convolutional layers. These layers learn to identify low-level features (e.g., edges, textures) in the early stages and high-level features (e.g., anatomical structures, patterns related to disease) in the deeper layers.

For brain MRI classification, feature extraction focuses on:

1) Identifying anatomical regions of interest (ROIs): The network learns to focus on key brain regions like the hippocampus, which is known to exhibit atrophy in Alzheimer's patients.

2) Capturing spatial relationships: Especially important for CapsNet, which preserves the pose and orientation of features, ensuring that spatial hierarchies are maintained.

For models like CNN, VGG, ResNet, and CapsNet, the convolutional layers and pooling layers work together to create hierarchical feature maps, which capture increasing levels of abstraction. In this process:

- Convolutional filters slide over the input image to detect features, producing feature maps.
- Pooling layers downsample the feature maps, reducing their size while retaining the most important information. This helps in reducing computational complexity and preventing overfitting.

In medical image classification, feature extraction is particularly challenging because the features that differentiate disease stages may be subtle, requiring deeper architectures (such as VGG or ResNet) to capture these variations effectively.

D. *Modeling Layer (Machine Learning)*

The modeling layer is where the deep learning architectures are applied to the preprocessed and feature-extracted data to perform classification. The models used in this study are:

1) Convolutional Neural Networks (CNNs): A standard architecture for image classification, CNNs use convolutional layers to detect features and pooling layers to reduce spatial dimensions. CNNs are widely used in medical imaging because of their ability to handle large amounts of spatial data efficiently. The network learns to extract relevant features from MRI scans and classify them into disease stages.

- The convolutional layer applies a set of filters (or kernels) to the input image. For a given input image X, the output feature map Y after applying the filter W can be expressed as:

$$Y_{i,j,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} W_{m,n,k} \cdot X_{i+m,j+n,c} + b_k \tag{9}$$

  Where:
  - $Y_{i,j,k}$ is the output feature map at position (i, j) for the k-th filter.
  - $W_{m,n,k}$ is the weight of the k-th filter at position (m, n).
  - $X_{i+m,j+n,c}$ is the input value at position $(i + m, j + n)$ of the c-th channel.
  - b_k is the bias associated with the k-th filter.
  - M and N are the dimensions of the filter.

- Activation Function: After the convolution operation, an activation function (commonly ReLU) is applied element-wise:

$$Z_{i,j,k} = \text{ReLU}(Y_{i,j,k}) = \max(0, Y_{i,j,k}) \tag{10}$$

  Where:
  - $Z_{i,j,k}$ is the output after the activation function.

- Pooling Layer: Pooling layers reduce the spatial dimensions of the feature maps, typically using max pooling. The output after max pooling can be expressed as:

$$P_{i,j,k} = \max_{m,n} Z_{s \cdot i+m, s \cdot j+n, k} \tag{11}$$

  Where:
  - $P_{i,j,k}$ is the pooled output.
  - s is the stride used for pooling.
  - m and n are the indices that iterate over the pooling window.

- Fully Connected Layer The output of the final pooling layer is flattened and fed into a fully connected layer. The output of this layer is calculated as:

$$O_j = \sum_{i=1}^{N} W_{ij} \cdot A_i + b_j \tag{12}$$

  Where:
  - O_j is the output of the j-th neuron in the fully connected layer.
  - W_{ij} is the weight from neuron i to neuron j.
  - A_i is the activation from the previous layer (flattened feature maps).
  - b_j is the bias term for the j-th neuron.
  - N is the number of inputs to the fully connected layer.

- Output Layer For classification tasks, the final output layer often uses the softmax function to convert the logits into probabilities:

$$P(y_k) = \frac{e^{O_k}}{\sum_{j=1}^{K} e^{O_j}} \tag{13}$$

  Where:
  - $P(y_k)$ is the probability of class k.
  - O_k is the output of the k-th neuron in the output layer.
  - K is the number of classes.

2) VGG: VGG extends CNNs by utilizing deeper networks, primarily with small 3×3 convolutional filters. It uses max pooling layers with a fixed stride of 2 and a 2x2 pool size, which helps to reduce the spatial dimensions while maintaining the critical features. This increased depth allows the model to capture more complex and finer details in MRI images. VGG typically consists of multiple convolutional layers followed by max pooling layers, with three fully connected layers at the end. The first two fully connected layers have 4096 neurons each, while the last corresponds to the number of classes in the classification task. It Primarily employs the ReLU (Rectified Linear Unit) activation function after each convolutional layer, which has become standard in many modern architectures.

3) ResNet Residual Networks: It solves the problem of vanishing gradients in deep networks by introducing skip connections, which allow gradients to flow directly through the network without degradation. ResNet is particularly effective in medical image classification tasks because it enables the model to learn deep, complex representations of brain structures that differentiate healthy brains from diseased ones.
The core building block of ResNet is the residual block, which adds the input of the block to the output of the convolutional layers, promoting efficient training and deeper architectures.

- Input Layer The input to the ResNet model is typically an image represented as a tensor X of dimensions $H \times W \times C$, where H is height, W is width, and C is the number of channels (3 for RGB images).
- Convolutional Layer The first layer in ResNet usually consists of a convolutional layer that applies a set of filters (kernels) to the input image. The output feature map Y after applying a filter W can be expressed as:

$$Y_{i,j,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} W_{m,n,k} \cdot X_{i+m,j+n,c} + b_k \tag{14}$$

  Where:
  - $Y_{i,j,k}$ is the output feature map at position (i, j) for the k-th filter.
  - $W_{m,n,k}$ is the weight of the k-th filter at position (m, n).
  - $X_{i+m,j+n,c}$ is the input value at position $(i + m, j + n)$ of the c-th channel.
  - b_k is the bias associated with the k-th filter.
  - M and N are the dimensions of the filter (typically 3×3 in ResNet).
- Residual Block The core building block of ResNet is the residual block, which enables the network to learn residual functions. A basic residual block can be defined as follows:

$$F(X) = \text{ReLU}(W_2 * \text{ReLU}(W_1 * X + b_1) + b_2) \tag{15}$$

  Where:
  - X is the input to the block.
  - W_1 and W_2 are the weights for the two convolutional layers.
  - b_1 and b_2 are the corresponding biases.

  The output of the block is the sum of the input and the output of the convolutional layers:

$$Z = F(X) + X \tag{16}$$

  Where:
  - Z is the final output of the residual block.
  - The addition $F(X) + X$ represents the skip connection that helps the gradient flow during backpropagation.
- Activation Function After computing the residual output, a non-linear activation function, typically ReLU, is applied to Z:

$$Z' = \text{ReLU}(Z) \tag{17}$$

- Pooling Layer Pooling layers are used in ResNet to reduce the spatial dimensions of the feature maps. The max pooling operation can be represented as:

$$P_{i,j,k} = \max_{m,n} Z_{s \cdot i+m, s \cdot j+n, k'} \tag{18}$$

  Where:
  - $P_{i,j,k}$ is the pooled output.
  - s is the stride used for pooling.
- Fully Connected Layers The output from the final convolutional layer is flattened and passed to a fully connected layer. The output of this layer can be computed as:

$$O_j = \sum_{i=1}^{N} W_{ij} \cdot A_i + b_j \tag{19}$$

  Where:
  - O_j is the output of the j-th neuron in the fully connected layer.
  - W_{ij} is the weight from neuron i to neuron j.
  - A_i is the activation from the previous layer.
  - b_j is the bias term for the j-th neuron.
  - N is the number of inputs to the fully connected layer.

- Output Layer For multi-class classification tasks, the output layer typically uses the softmax function to convert logits into probabilities:

$$P(y_k) = \frac{e^{O_k}}{\sum_{j=1}^{K} e^{O_j}} \tag{20}$$

Where:
  - $P(y_k)$ is the probability of class k.
  - O_k is the output of the k-th neuron in the output layer.
  - K is the total number of classes.

4) Capsule Networks (CapsNet): Unlike CNNs, which are prone to losing spatial information as features are pooled, CapsNet preserves the spatial hierarchy of features using capsules—small groups of neurons that learn not just the presence of a feature but also its orientation, position, and scale. CapsNet's dynamic routing between capsules enables it to handle variations in the input image better, making it especially useful for medical images where subtle differences in orientation and structure are diagnostically significant.

- The mathematical model of a Capsule Network (CapsNet) starts with an input, typically an image or a set of images, that is processed by a convolutional layer. This paper outlines the key equations that describe the information flow through a CapsNet.

$$z_{ij,k} = \sum_{l=1}^{L} W_{ij,k,l} X_{ij,l} + b_{ij,k} \tag{21}$$

where $z_{ij,k}$ is the activation at position (i, j) in the k-th feature map, $W_{ij,k,l}$ is the weight associated with the i-th input channel, $X_{ij,l}$ is the input, and $b_{ij,k}$ is the bias term.
Next, we move to the primary capsule layer, where the output vector of the k-th capsule is given by:

$$V_{ij,k} = \text{Squash}\left(\sum_{l=1}^{L'} W_{ij,k,l} u_{ij,l}\right) \tag{22}$$

The "Squash" function ensures the vector length is between 0 and 1.
Higher-level capsules aggregate inputs from lower-level capsules using routing by agreement, defined as:

$$S_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{23}$$

Here, S_j is the output of the j-th higher-level capsule, c_{ij} is the coupling coefficient, and $\hat{u}_{j|i}$ is the predicted output from lower-level capsules.
The final capsule output is given by:

$$y_k = \text{squash}(s_k) \tag{24}$$

where y_k is the output vector of the k-th capsule, and s_k is its input vector.
These equations illustrate how information flows through the capsule network. Backpropagation is utilized to optimize weights and biases. The network design can be adapted for specific tasks by introducing new routing algorithms or adding layers to enhance speed and interaction between capsules.

E. Optimization Algorithms

To improve model performance and convergence speed, optimization algorithms such as Gannet, SBO (Sequential Bayesian Optimization), and ESBO (Enhanced Sequential Bayesian Optimization) are employed. These algorithms help fine-tune hyper-parameters (like learning rates, batch sizes, and network architecture choices) to maximize model performance on validation datasets

1) Gannet: is designed to explore hyperparameter spaces efficiently, adapting the search process based on past performance to identify optimal configurations.

2) SBO: leverages Bayesian principles to update beliefs about hyperparameter effects dynamically, allowing for a more informed search strategy that improves convergence speed.

3) ESBO: extends SBO by enhancing the search process with advanced sampling techniques, allowing for better exploration of hyperparameter space and yielding improved performance outcomes.
Each model is trained and validated using a portion of the dataset. Cross-validation techniques are employed to assess how well the models generalize to unseen data. Additionally, hyperparameter tuning is performed using the optimization algorithms to optimize learning rates, batch sizes, and other model-specific parameters.

Criteria	**CNN**	**VGG**	**ResNet**	**CapsNet**
Architecture	Multiple convolutional and pooling layers, followed by fully connected layers	Deep, sequential architecture with multiple convolutional layers (16 or 19 layers)	Uses residual blocks that allow the network to skip layers	Uses capsules, groups of neurons that preserve spatial hierarchies
Feature Learning	Learns basic features like edges, textures, and patterns	Focuses on extracting detailed features, larger depth	Residual connections allow better feature propagation	Encodes spatial relationships between features, not just the features
Training Complexity	Easier to train, simpler architecture	High computational cost due to deep layers	Efficient training with residual connections; mitigates vanishing gradient	Difficult to train due to dynamic routing and more complex architecture
Parameter Size	Moderate, depends on the depth of layers	Large, due to depth and fully connected layers	Fewer parameters due to residual connections	Small compared to VGG, but capsules increase the parameter complexity
Training Time	Relatively short, especially for shallow architectures	Long due to the large number of layers and parameters	Efficient due to residual connections despite large depth	Longer due to complex routing mechanisms between capsules
Generalization	Performs well, prone to overfitting if too deep	Good generalization, but prone to overfitting.	Excellent generalization due to residual learning	Better at preserving information,subtle to data distribution

TABLE I
COMPARISON OF DIFFERENT NEURAL NETWORK ARCHITECTURES

V. RESULTS

Measuring Type 1 and Type 2 errors through these metrics allows you to better understand and evaluate the performance of your machine learning models. Depending on the specific use case, different metrics may be prioritized to achieve the desired balance of accuracy and error types In the context of machine learning and statistical classification, Type 1 and Type 2 errors can be measured using specific metrics derived from the confusion matrix. Following the metrics related to these errors:

TYPE 1 ERROR (FALSE POSITIVE)

Definition

A Type 1 error occurs when the model incorrectly identifies a negative instance as positive. This means that the model predicts the presence of a condition (e.g., a disease) when it is not actually present.

Metrics Associated

1) Precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision indicates the quality of positive predictions. It answers the question: ”Of all instances classified as positive, how many were actually positive?” A high precision means fewer false positives.

2) False Positive Rate (FPR):

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

FPR shows the proportion of actual negatives that were incorrectly identified as positive. It’s also known as the fall-out rate. A low FPR is desirable as it indicates fewer incorrect positive predictions.

3) False Discovery Rate (FDR):

$$\text{FDR} = \frac{\text{FP}}{\text{TP} + \text{FP}}$$

FDR measures the proportion of false positives among all positive predictions. It tells us how many of the predicted positives were actually negative. Lowering the FDR improves the reliability of positive predictions.

TYPE 2 ERROR (FALSE NEGATIVE)

Definition

A Type 2 error occurs when the model fails to identify a positive instance, incorrectly classifying it as negative. This means that the model predicts the absence of a condition when it is actually present.

Metrics Associated

4) Recall (Sensitivity or True Positive Rate):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall measures the model's ability to identify all relevant instances. It answers the question: "Of all actual positives, how many were correctly identified?" A high recall means fewer false negatives.

5) False Negative Rate (FNR):

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}}$$

FNR shows the proportion of actual positives that were incorrectly identified as negative. It's the complement of recall. A low FNR indicates that most positive instances are correctly identified.

GENERAL METRICS (NOT CATEGORIZED AS TYPE 1 OR TYPE 2)

6) Accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Accuracy measures the overall correctness of the model across all instances. It answers the question: "What proportion of total instances were correctly classified?" While useful, accuracy can be misleading, especially in imbalanced datasets where one class significantly outweighs the other.

7) F1 Score:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score combines precision and recall into a single metric, providing a balance between the two. It's particularly useful when you need to account for both false positives and false negatives. A high F1 score indicates a model that has both good precision and recall.

Measure	**CNN**	**VGG16**	**ResNet**
Accuracy	0.901	0.666	0.529
Precision	0.843	0.625	0.529
Recall	1.0	0.925	1.0
F1 Score	0.915	0.746	0.692
False Negative Rate (FNR)	0.0	0.074	0.0
False Positive Rate (FPR)	0.208	0.625	1.0
False Discovery Rate (FDR)	0.156	0.375	0.470

TABLE II
COMPARISON OF PERFORMANCE METRICS FOR DIFFERENT MODELS

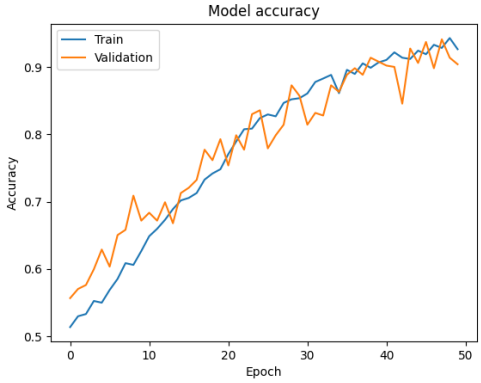

Fig. 1. CNN Accuracy per Epoch

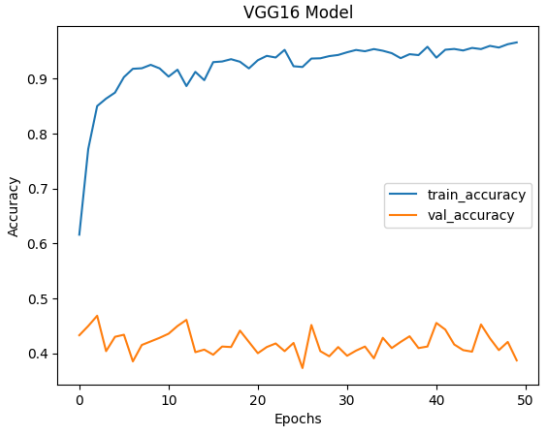

Fig. 2. VGG16 Accuracy per Epoch

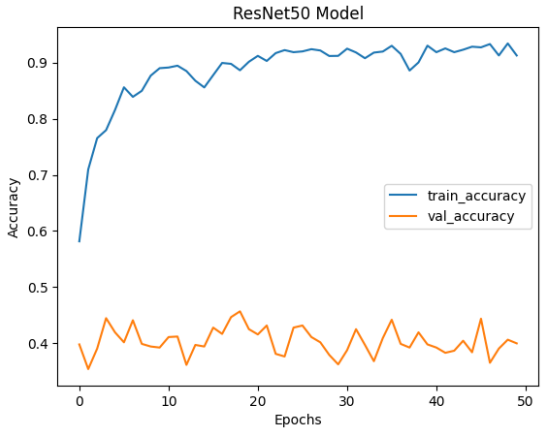

Fig. 3. ResNet Accuracy per Epoch

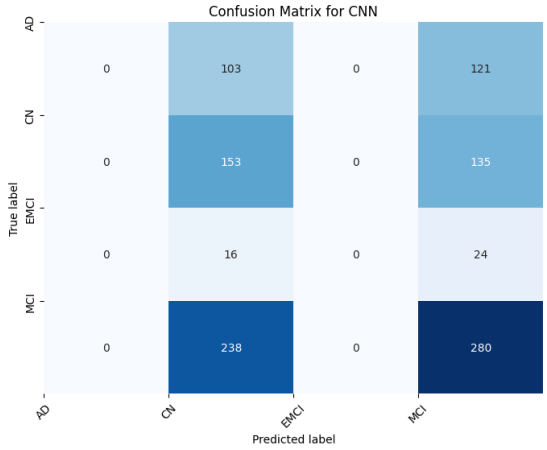

Fig. 4. CNN

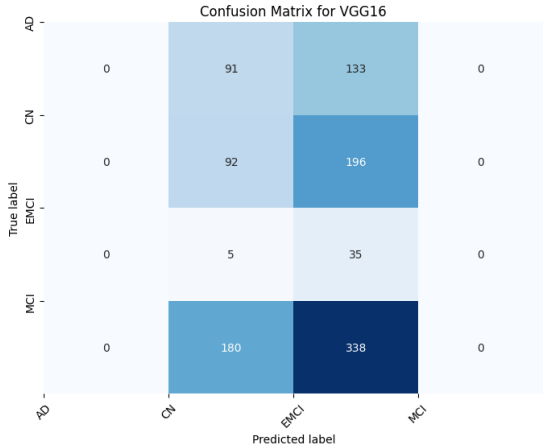

Fig. 5. VGG16

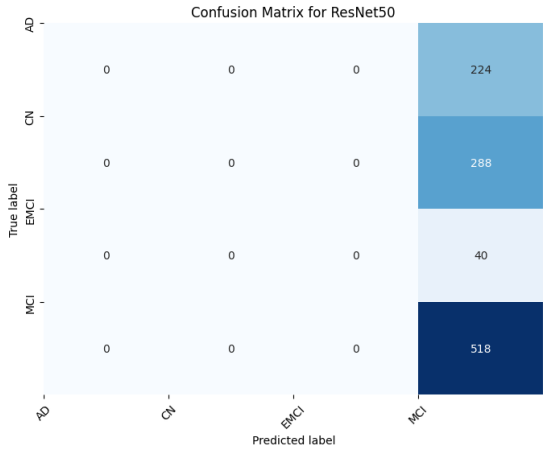

Fig. 6. ResNet

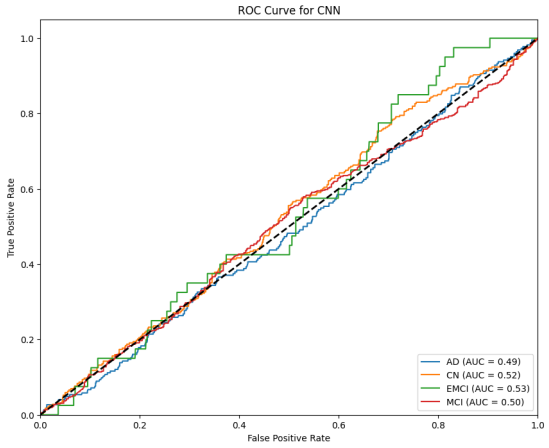

Fig. 7. ROC Curve for CNN

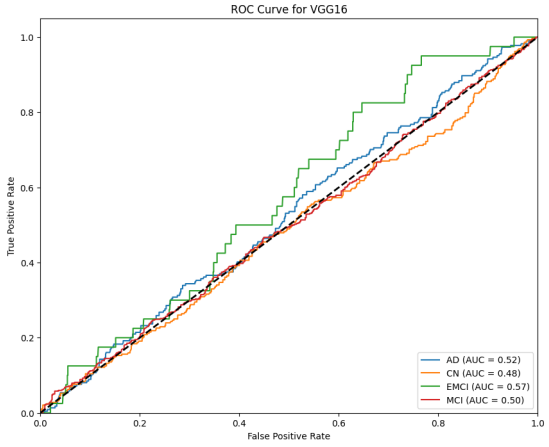

Fig. 8. ROC Curve for VGG16

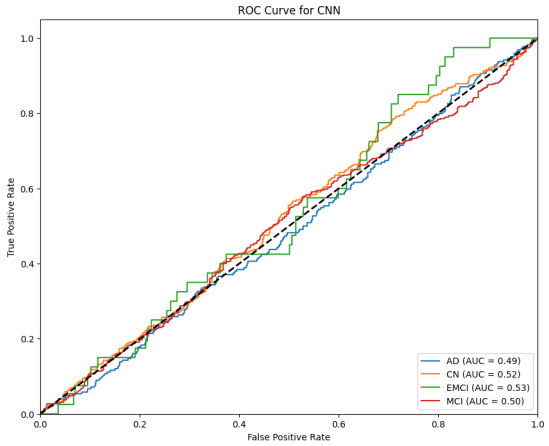

Fig. 9. ROC Curve for ResNet 50

VI. Conclusion

In conclusion, this research presents a comparative analysis of four prominent deep learning models—CNN, VGG, ResNet, and CapsNet—applied to the classification of brain MRI images. Each model demonstrates unique strengths and weaknesses

across different performance metrics such as accuracy, precision, recall, and computational efficiency. CNNs and ResNet stand out for their balance between accuracy and generalization, while VGG exhibits high precision in detailed feature extraction, albeit with higher computational costs. CapsNet, though relatively new, shows promise in preserving spatial hierarchies, making it particularly useful for medical images where subtle anatomical changes are crucial.

The study highlights the importance of model selection based on the specific requirements of clinical applications, such as real-time diagnosis or detailed disease staging. The findings emphasize the potential for integrating these models into real-world healthcare systems, improving diagnostic accuracy and reducing the workload on medical professionals. Future research could explore hybrid models that combine the strengths of multiple architectures, further enhancing the applicability of deep learning in medical imaging.

REFERENCES

[1] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[2] O'Shea, Keiron Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

[3] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[4] F. Altaf, S. M. S. Islam, N. Akhtar and N. K. Janjua, "Going Deep in Medical Image Analysis: Concepts, Methods, Challenges, and Future Directions," in IEEE Access, vol. 7, pp. 99540-99572, 2019, doi: 10.1109/ACCESS.2019.2929365.

[5] Mittal, Ansh Kumar, Deepika. (2019). AiCNNs (Artificially-integrated Convolutional Neural Networks) for Brain Tumor Prediction. EAI Endorsed Transactions on Pervasive Health and Technology. 5. 10.4108/eai.12-2-2019.161976.

[6] Klang E. Deep learning and medical imaging. J Thorac Dis. 2018 Mar;10(3):1325-1328. doi: 10.21037/jtd.2018.02.76. PMID: 29708147; PMCID: PMC5906243.

[7] Li M, Jiang Y, Zhang Y, Zhu H. Medical image analysis using deep learning algorithms. Front Public Health. 2023 Nov 7;11:1273253. doi: 10.3389/fpubh.2023.1273253. PMID: 38026291; PMCID: PMC10662291.

[8] Haq, Mahmood Ul Sethi, Muhammad Rehman, Atiq. (2023). Capsule Network with Its Limitation, Modification, and Applications—A Survey. Machine Learning and Knowledge Extraction. 5. 891-921. 10.3390/make5030047.

[9] Tran, Minh Vo, Khoa Quinn, Kyle Nguyen, Hien Luu, Khoa Le, Ngan. (2022). CapsNet for Medical Image Segmentation. 10.48550/arXiv.2203.08948.

[10] Barragán-Montero A, Javaid U, Valdés G, Nguyen D, Desbordes P, Macq B, Willems S, Vandewinckele L, Holmström M, Löfman F, Michiels S, Souris K, Sterpin E, Lee JA. Artificial intelligence and machine learning for medical imaging: A technology review. Phys Med. 2021 Mar;83:242-256. doi: 10.1016/j.ejmp.2021.04.016. Epub 2021 May 9. PMID: 33979715; PMCID: PMC8184621.

[11] Mazzia, V., Salvetti, F. Chiaberge, M. Efficient-CapsNet: capsule network with self-attention routing. Sci Rep 11, 14634 (2021). https://doi.org/10.1038/s41598-021-93977-0

[12] Liang, Jiazhi. (2020). Image classification based on RESNET. Journal of Physics: Conference Series. 1634. 012110. 10.1088/1742-6596/1634/1/012110.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[14] Simonyan, Karen Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[15] Tammina, Srikanth. (2019). Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. International Journal of Scientific and Research Publications (IJSRP). 9. p9420. 10.29322/IJSRP.9.10.2019.p9420.

[16] Mukhometzianov, Rinat Carrillo, Juan. (2018). CapsNet comparative performance evaluation for image classification. 10.48550/arXiv.1805.11195.

[17] I. J. Goodfellow et al., "Multi-digit number recognition from street view imagery using deep convolutional neural networks," arXiv preprint arXiv:1312.6082, 2013.

[18] G. E. Hinton, "Shape representation in parallel systems," in *International Joint Conference on Artificial Intelligence*, vol. 2, 1981.

[19] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*, pp. 44–51, Springer, 2011.

[20] M. Jaderberg et al., "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

[21] Y. Netzer et al., "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, vol. 2011, p. 5, 2011.

[22] B. A. Olshausen, C. H. Anderson, and D. C. Van Essen, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *Journal of Neuroscience*, 13(11):4700–4719, 1993.

[23] L. Wan et al., "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.

[24] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," arXiv preprint arXiv:1301.3557, 2013.