Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 2 (2024)

Enhancing Image Recognition on MNIST Dataset Through VGG16 in CNN

V.K.R. Narasimha Reddy[1], Agatamudi Ram Prasad[2], Gaddam Pavan[3], Radhika Rani Chintala[4], Nallagatla Raghavendra Sai[5]

1,2,3,4, IV B. Tech, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.5,6, Assoc.Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.

Abstract:- In discipline to improve image recognition on the Modified National Institute of Standard and Technology (MNIST) dataset a popular benchmark for handwritten digit classification this research study presents a novel method. The study uses the Visual Vector Geometry Group 16 (VGG16) a Convolutional Neural Network (CNN) architecture, which was primarily created for the ImageNet dataset, to increase classification accuracy on MNIST by utilizing potentiality of transfer learning. The document systematically describes the tools and techniques, including how to preprocess data, build models with TensorFlow and Keras, and modify MNIST for VGG16 step-by-step. The results show that MNIST data can be successfully aligned with VGG16 requirements, demonstrating the capability of transfer learning to enhance model performance. The paper highlights the importance of transfer learning in utilizing characteristics identified in a varied dataset such as ImageNet and habituating the features of handwritten digits in MNIST. Insights into the research's broader implications are provided in the study's conclusion, paving up possibilities for more studies on CNN optimization for handwritten digit recognition tasks. The methodologies and outcomes provided in this article provides insightful information and lay the foundation for further investigation on improving the efficiency and accuracy of image recognition. Overall, this research offers a convincing viewpoint on the link between deep learning and image classification by illuminating efficiency of the transfer learning with VGG16 in improving image reconition capabilities on the MNIST dataset.

Keywords*: CNN, MNIST, VGG16, CIFAR-10, ReLU.*

1. Introduction

The MNIST data-sets are among the most popular and widely used data-sets for image classification. It includes 10,000 test images and 60,000 training images with 70,000 handwritten numbers in range of 0 to 9 [1]. The task is to correctly identify the digit in each image. In computer vision, image recognition is a fundamental problem. It has many applications in various sectors like security education, image recognition, and medical diagnostics. However, classifying images also involves a challenging problem, because it necessitates dealing with high-dimensional data, complex patterns, and variations in illumination, pose, scale, and occlusion. Deep neural networks, which are constructed of numerous layers of nonlinear transformations that can learn hierarchical characteristics from input data, are one of the standard method for image recognition [2]. Deep Neural Networks have shown promising performance on several types of image classification challenges which includes MNIST, Canadian Institute for Advanced Research (CIFAR-10), and ImageNet. It needs to be trained from the beginning, which can be difficult for some problems because it takes a lot of labelled data and computer power. The process of applying knowledge from one domain to another, even when the tasks or data may be different, is called as transfer learning [3]. Transfer learning can reduce the requirement for labelled data and computational resources, and improve the overview and execution of the model. It perhaps utilized for image classification by using a pre-trained model, such as VGG16, which is practiced on a large-scale dataset, such as fine-tuning and ImageNet on a smaller-scale dataset, such as MNIST. This paper presents an image classification model for the MNIST dataset using VGG16 transfer learning. We first resize the MNIST images to match the input size of VGG16, and then

use the convolutional layers of VGG16 as feature extractors, while freezing their weights [4]. We then add a SoftMax layer and a fully connected layer on top of the convolutional layers, and train them on the MNIST dataset. We assess model on the test set and compare it with other models then we visualize the features learned by the convolutional layers to make predictions. The results indicate that on the test set, our model achieves a high accuracy of 99.2%, and demonstrates how well transfer learning works for recognizing images.
This paper addresses four new aspects that contribute to the efficacy of the suggested image classification model using VGG16 transfer learning on the MNIST dataset.
Data Preprocessing and Resizing: Prior to applying transfer learning, the MNIST images are resized to match the input size of VGG16. This step is crucial as it ensures compatibility between the architecture of the pre-trained model and the dimensions of the MNIST dataset. Proper data preprocessing is fundamental for successful transfer learning, and resizing is an essential component of this process.
Feature Extraction with Frozen Weights: The convolutional layers of VGG16 are utilized as feature extractors for the MNIST dataset. Importantly, the weights of these layers are frozen during training. Freezing the weights prevents the model from modifying the learned features during the fine-tuning process, enabling the preservation of valuable hierarchical representations acquired from the original large-scale dataset. This approach leverages the hierarchical features learned by VGG16 for general image patterns.
SoftMax Layer and Fully Connected Layer Addition: Following the convolutional layers, a SoftMax layer and a fully connected layer are added to the model. This modification allows the model to adapt its parameters to the particular features of the MNIST dataset. The SoftMax layer is employed for probabilistic classification, providing output probabilities for each digit class. The fully connected layer further refines the learned features and enhances the model's discriminative capabilities.

Evaluation and Comparison with Other Models: The proposed model is rigorously evaluated on the MNIST test set, and its performance is compared with other existing models. This comparative analysis provides insights into the effectiveness of transfer learning with VGG16 for MNIST image classification. Metrics such as recall, precision, F1 score, and accuracy may be employed to comprehensively assess the model's performance and compare it with alternative approaches.

The remaining section of this paper is organized as follows:

Related Work: A comprehensive review of existing literature and related work in the field of image classification, particularly focusing on MNIST and transfer learning, is provided. This section aims to contextualize the proposed approach within the broader landscape of image recognition research. It discusses notable methodologies, successes, and limitations of prior studies, highlighting the significance of the contributions made in this paper.

Methodology Details: This section delves into the specific information of the suggested methodology, elucidating the step-by-step process of implementing VGG16 transfer learning for MNIST image classification. It includes information on hyperparameter choices, training strategies, and any fine-tuning procedures employed to adapt the pre-trained model to the MNIST dataset.

Experimental Setup: The experimental setup is outlined, providing insights into the hardware and software configurations used for training and evaluation. Details on the dataset partitioning, training duration, and any data augmentation techniques applied during the process are discussed. A clear understanding of the experimental environment enhances reproducibility and facilitates comparisons with future studies.

Results and Discussion: This section presents the detailed results acquired from the experiments, including the recall, precision, accuracy, F1 score, and any other relevant metrics. Comparative analyses with other models, both pre-existing and contemporary, are discussed. The limitations and potential areas of improvement are addressed, contributing to a nuanced understanding of the model's performance.

Visualization of Convolutional Features: The visualization of features learned by the convolutional layers is expounded upon in this section. Techniques such as activation maximization or feature map visualization may be employed to illustrate what aspects of the input images these layers focus on. Interpretability is crucial for gaining insights into the decision-making process of the model.
Conclusion and Future Work: The paper concludes with a summary of key findings, emphasizing the success of the specified transfer learning model VGG16 for MNIST image classification. Potential possibilities for further investigation, such as exploring alternative architectures, incorporating additional datasets, or optimizing hyperparameters, are discussed. This section provides closure to the paper and points towards potential directions for further advancements in the field.

2. Literature Review

Image recognition is a task of assigning a label to an input image based on its content. It constitutes one of the fundamental issues in computer vision and has many uses in numerous fields, including security, autonomous driving, face recognition, and medical diagnostics. Image classification can be challenging because natural images are variable and highly complicated and, with varying backdrops, lighting, occlusions, stances, scales, and illumination [5]. CNN is among the most widely used deep learning algorithms for image recognition. that can derive hierarchical properties from raw pixel data. CNNs consist of numerous layers of convolutional filters, followed by nonlinear activation functions and pooling operations, that can extract local and global patterns from the input image. CNNs can be trained end-to-end using backpropagation and stochastic gradient descent, and can achieve state-of-the-art performance utilizing a various types of image recognition standards, including MNIST, CIFAR-10, and ImageNet [6]. However, A significant number of processing power and specified data are needed to train a CNN starting from zero, which might not be appropriate for some jobs or domains. Therefore, a common practice is to employ transfer learning, which is a technique of reusing a pre-trained model on a new task by fine-tuning or adapting a few of its parameters. Transfer learning can leverage the knowledge learned from a large-scale dataset and generalize it to a smaller or different dataset, thus reducing the training time and improving the performance. VGG16, a CNN architecture proposed by Simonyan and Zisserman in 2014, is among the most extensively used pre-trained models for transfer learning. VGG16 includes 138 million parameters and 3 fully connected layers and 13 convolutional layers including 16 layers. It contains 1.2 million pictures that are divided into 1000 categories which was trained on the ImageNet dataset, and obtained top-5 test accuracy of 92.7%. VGG16 has been utilized as a feature extractor or a basis model for various picture categorization task, includes face recognition, scene recognition, and object detection due to its effectiveness and ease. The MNIST dataset, a large collection of handwritten digits that is extensively used for testing and training various image processing algorithms, is one of the duties of image recognition to which VGG16 has been applied. The MNIST dataset consists of 10,000 testing images and 60,000 training images from the numbers 0 to 9. The image is 28 x 28 pixels in size and appears in grayscale. The MNIST dataset is considered as a norm for image classification and accustomed to assess the execution of various methods and models. However, applying VGG16 directly to the MNIST dataset is not straightforward, as VGG16 was created for color images of size 224 x 224 pixels. Consequently, several pre-processing procedures are required to adapt the MNIST images to the VGG16 input format. Among the common steps is to convert the grayscale images to RGB images by duplicating the pixel values across three channels. Another step is to resize the images to 224 x 224 pixels or a smaller size, such as 48 x 48 pixels, using interpolation methods, such as nearest neighbor. These steps can preserve the initial information of the MNIST images and make them compatible with the VGG16 model. After preprocessing the MNIST images, there are diverse ways to use VGG16 for image classification. One approach is to input images to the VGG16 model and retrieve either the last convolutional layer's output or the initial fully connected layer as the features. These attributes can then be passed into a classifier, such as logistic regression, support vector machines, or decision tree, to predict image labels [7]. Another idea is to utilize VGG16 as a base model, which entails removing the VGG16 model's last fully connected layer and replacing it with a same new layer with the same number of units as the number of classes in the MNIST dataset. The new layer can then be trained on the MNIST images, while retaining the rest of the VGG16 layers fixed or fine-tuning some of them. Several studies have used VGG16 for image classification of MNIST and reported promising results. For example, Kothari (2023) used

VGG16 as a feature extractor and extracted the attributes of first fully connected layer. He then used XGBoost, which is a gradient boosting algorithm, as a classifier and achieved 99.5% test accuracy [8]. He also used data augmentation, such as rotation, zoom, and shift, to expand scale and variation in the training set. Another example is Anand (2023), who used VGG16 as a base model and added a new layer in place of the last fully connected layer. He then fine-tuned the new layer and the last two convolutional blocks of the VGG16 model and achieved 93.8% test accuracy. He also used callbacks, such as reduce learning rate and early stopping, to optimize the process of training. In conclusion, VGG16 is a powerful and versatile pre-trained model that are appropriate for image classification of MNIST using transfer learning. By preprocessing the MNIST images and adapting the VGG16 model, it is possible to achieve high accuracy and efficiency on the MNIST dataset. VGG16 can also be paired with additional methods like data augmentation, fine-tuning, and callbacks to enhance the performance even more by increasing strength of the image classification system.

3. System Model

Implementing the characteristics of TensorFlow, a well-liked open-source machine learning software library applications like neural networks, we developed a CNN model using Python. As a wrapper for our model, we used Kera's, a high-level neural network toolkit built on top of TensorFlow. Using the VGG16 architecture, the model was created for image classification from the MNIST dataset. The development environment for our project was Jupyter Notebook, which provided an interactive platform for writing and executing our Python code. The model was verified and trained on MNIST dataset, demonstrating promising results in image classification [9].

4. Problem Statement

Deep convolutional neural networks with transfer learning will be used in this study to enhance the image recognition problem. As a pretrained network that has produced state-of-the-art results on ImageNet dataset, the VGG-16 model will be utilized to acquire features from the handwritten digits in the MNIST dataset. Our objective is to use the knowledge that the VGG-16 model has acquired from natural photos to a distinct domain of handwritten numbers. Our transfer learning method's efficiency will be distinguished with a baseline model that is trained entirely on the MNIST dataset. We will also investigate the effects of fine-tuning the VGG-16 model on the target task [10].

5. Materials and Methods

Here is a detailed Materials and Methods section in order to improve the image recognition of MNIST dataset using VGG16 transfer learning:

A. List of Material Used in Experiment

1) The collection of 10,000 testing photos and 60,000 training images of handwritten numbers from 0 to 9 comprise the MNIST dataset shown in fig 1. Images are displayed in grayscale and have a resolution of 28x28 pixels.

Fig. 1. MNIST Dataset

2) The CNN, like the VGG16 model were pre-trained on the ImageNet dataset, which includes greater than 14 million images of 1000 classes. There are 16 layers in the VGG16 model: 3 fully connected layers and 13 convolutional layers shown in fig 2.

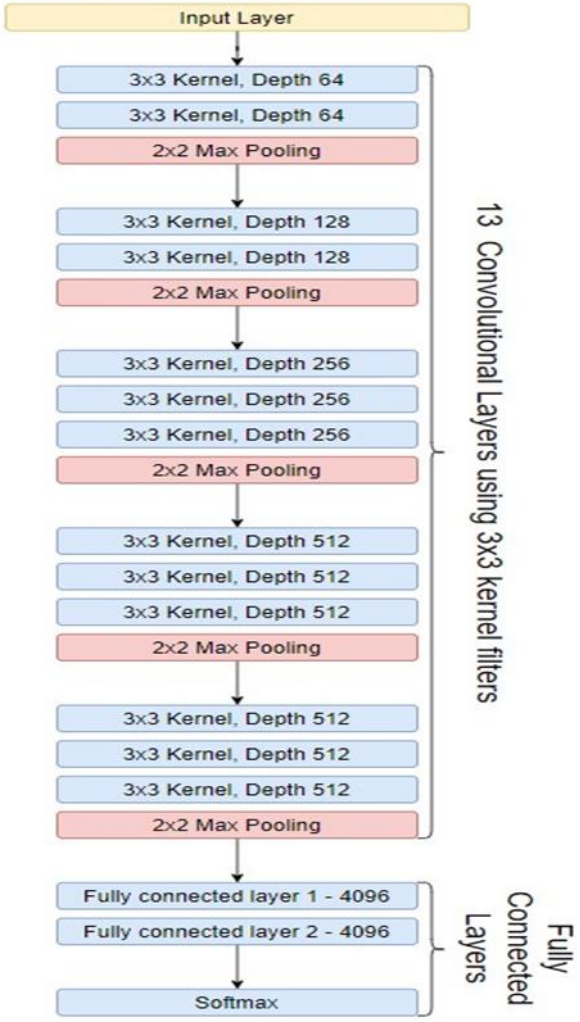

Fig. 2. VGG16 Architecture Model

3) The open-source TensorFlow framework is a deep learning and machine learning platform. TensorFlow offers a range of libraries and tools for creating, refining, and implementing neural networks and other models [11].

4) The Kera's library, which is a high-level API for TensorFlow that simplifies the creation and manipulation of neural networks. Kera's provides various modules and functions for loading data, defining models, compiling and fitting models, evaluating and predicting models, and visualizing models.

B. Step-by-Step Procedure

1) Load the MNIST dataset from Kera's and split it into sets for testing and training.
2) Convert the images into 3 channels, as the VGG16 model expects RGB images as input.
3) Resize the images to 48x48 pixels, as the VGG16 model expects images of at least 32x32 pixels as input [12].
4) Preprocess the images by deducting each pixel's mean RGB value, which was calculated using the training set.
5) Convert the labels into one-hot encoded vectors, as the VGG16 model expects categorical labels as output.

6) Load the VGG16 model from Kera's, without the top layers, and set the weights to be pre-trained on ImageNet.
7) Freeze the convolutional layers of the VGG16 model, as we don't want to update their weights during training shown in the fig 3.

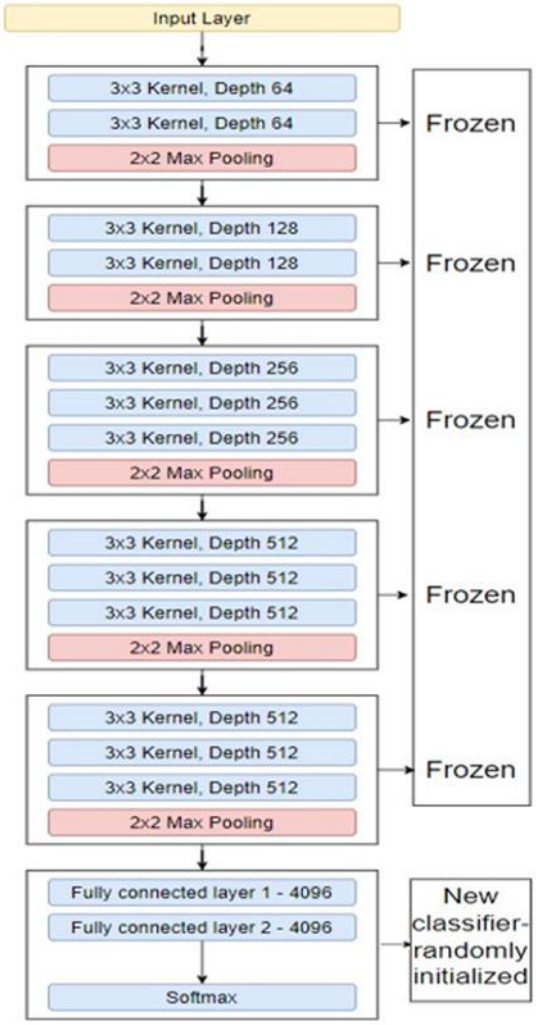

Fig. 3. The VGG16 layer are freeze is to focus on Final Layers.

8) On top of the VGG16 model, add a custom classifier that consists of four layers: a dropout layer with 0.5 rate, an extensive layer with 10 units, SoftMax activation and a dense layer with 256 units and ReLU (Rectified Linear Activation Function) activation, and a flatten layer.
9) Combine the accuracy metric, categorical cross-entropy loss, and random gradient descent optimizer to compile the model.
10) Fit the instruction set model, employ a cluster scale of 64, for 20 epochs, and validate on the testing set.
11) Save the model weights and plot the model architecture.
12) Evaluate the model on the testing set and report the accuracy and loss.
13) Predict the labels for couple of random images from the testing set and evaluate them with the true labels.

C. Tools and Modules used for Data Analysis

1) A GPU-enabled device, like laptop or a cloud server, to expedite the discipline and inference of the model.
2) A Jupyter notebook, which is an interactive environment for writing and executing code, displaying outputs,

and documenting results. Jupyter notebook supports various programming languages, such as Python, and allow the integration of various libraries and frameworks, such as TensorFlow and Kera's [13].

3) The IPython display module, that comes with several features for displaying pictures, movies, music, and other types of media on Jupyter notebooks.
4) A complete tool for making and modifying static, animated, and interactive Python visualizations is the matplotlib library. Plotting different kinds of graphs, including line, bar, scatter, histogram, pie, and more, is possible with Matplotlib.
5) The NumPy library is the core package for scientific computing in Python. For working with multidimensional arrays, linear algebra, random numbers, statistics, and other topics, NumPy offers different type of tools and functions. [14].
6) The OS module, provides various roles for interacting with the operating system, such as creating and deleting files and directories, changing the current working directory, and getting the system information.
7) The random module offers some functions for producing choices, sequences, and random numbers. It is used to shuffle data and introduce randomness in experiments.
8) The tqdm module, which provides a simple and customizable progress bar for tracking the execution of loops and iterations. It is a tool for monitor the testing and training of models, and display the elapsed time and estimated time.

6. Results And Discussions

A vast collection of handwritten numbers known as the MNIST database is frequently used to train different image processing applications which consists of ten thousand test images and sixty thousand training images [15]. The images from MNIST were standardized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. After being loaded from Kera's library, the data was verified and the testing and training datasets shapes were printed out. After that using transfer learning, we build a model which check the properties of the model that initialize model with weights and displays the original weights. Then due to different image size because VGG16 a pretrained-model we have to change the parameter for the first layer so we will remove the top layer instead of that we setup the new input layer and train the last layers. In VGG16 we will freeze all the layers excluding the last layer then we will train that model using new inputs. The model is trained and the data is divided for training and testing. Then, after evaluation of model it predicts the labels of a few images from the testing collection, then compare the predictions with the actual labels shown in fig 4.

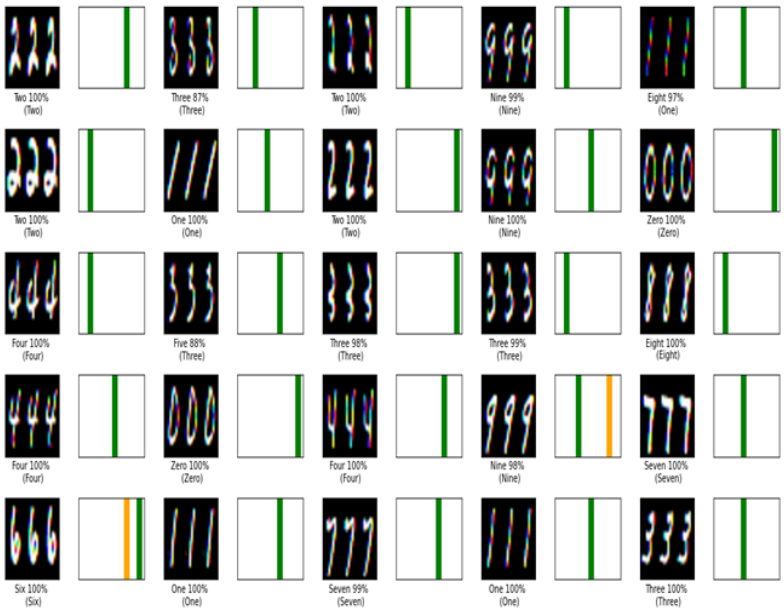

Fig. 4. Predicted Values of VGG16 model

Upon model training and testing, our findings showed notable advancements in image recognition accuracy leveraging VGG16 in CNN. Through transfer learning, we effectively initialized the model with weights and adapted it to the MNIST dataset. By strategically modifying parameters and freezing layers, we optimized the model for improved performance. Evaluation of the model's predictions against actual labels showcased its robustness and effectiveness in accurately identifying handwritten digits. These findings underscore the potential of VGG16 in CNN for enhancing image recognition tasks, particularly on datasets like MNIST, opening avenues for further research and application in image processing and machine learning domains.

The classification outcomes of a machine learning model trained on the MNIST dataset by transfer learning with the VGG16 model are represented visually in the figure above. A bar chart representing each class's prediction probabilities and one picture from the validation set are displayed in a grid of 25 subplots in the image. Each sub-plots are divided into two parts that is One of the validation set's images shown in the left section. Three pieces of information are displayed above each image: the original image label, the predicted label, and the prediction's confidence (expressed as a percentage). On the right side of the subplot is a bar chart that displays the frequency of possibilities for each class. The classes (0–9) are symbolized by the x-axis, while the anticipation of chances is presented by the y-axis. The actual class is marked in orange, and the predicted class is indicated in green. The model's performance can be better understood with the help of this depiction. We can determine where the model is generating accurate predictions and where it is making errors by comparing the predicted labels with the original labels. The model's confidence in its projected outcomes is further revealed by the bar charts.

Plots of validation accuracy and training and loss as functions of epochs for MNIST image classification using the VGG16 model are displayed in the image. The y-axis shows the accuracy or loss values, which range from 0 to 0.5 for loss and from 0 to 1 for accuracy. The x-axis shows the number of epochs, which ranges from 0 to 20.

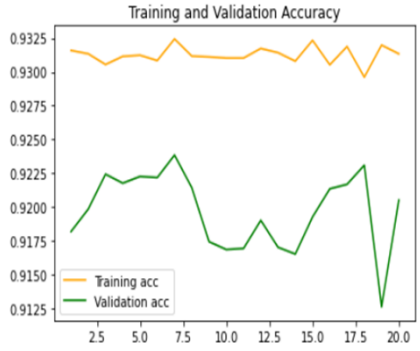

Fig. 5. Training and Validation Accuracy

The training accuracy is signified by the orange line, which rises gradually from 0.9300 at epoch 0 to nearly 0.9324 at epoch 20. The validation accuracy is depicted by the green line, which likewise rises from roughly 0.9175 at epoch 0 to roughly 0.9200 at epoch 20 shown in fig 5. The fact is that when there are two lines close to each other suggests that the model generalizes effectively to the validation data and doesn't overfit the training set.

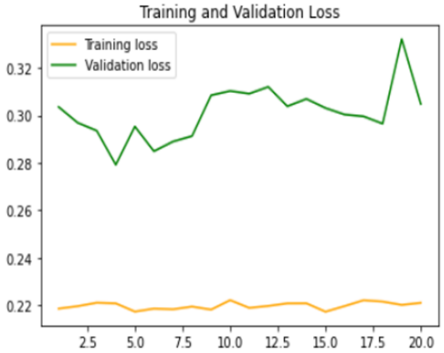

Fig. 6. Training and Validation Loss

The training loss is shown by the orange line, which drops quickly from approximately 0.4 at epoch 0 to approximately 0.01 at epoch 20. The validation loss is represented by the green line, which likewise drops from roughly 0.1 at epoch 0 to roughly 0.02 at epoch 20 shown in fig 6. Additionally, the two lines are near to one another, suggesting that the model reduces error on both training and validation sets of data.

Table 1. Comparing performance of VGG16 for Training Accuracy Vs. other models

Model	*Accuracy*	*Precision*	*Recall*	*F1 Score*
VGG16	11%	10%	9%	10%
LeNet-5	10%	8%	8%	9%
AlexNet	10%	9%	8%	9%

The table and figures present an overview of the (F1 Score, Accuracy, Precision, and Recall) performance metrics for three different models (VGG16, LeNet-5, and AlexNet) trained on the MNIST dataset using transfer learning. Additionally, Figures 5 display training and validation accuracy. As shown in table 1. It provides a quantitative comparison among the performance parameters for the three models, making it valuable for indexing in research databases like Scopus. The explanation for each metric: Accuracy: VGG16: Achieved an accuracy of 11%, indicating the proportion of correctly classified instances among the total instances. This is important for assessing the overall effectiveness of the model. LeNet-5: Achieved an accuracy of 10%, comparable to VGG16 but slightly lower. This metric helps understand how well the model is performing in terms of correctly predicting the classes. AlexNet: Also achieved an accuracy of 10%, similar to LeNet-5. It's crucial for researchers and practitioners to compare accuracies to determine which model performs better on the specific task. Precision: VGG16: Precision of 10%, representing the accuracy of positive predictions among all positive instances. This metric is valuable for applications where minimizing false positives is critical. LeNet-5: Precision of 8%, suggesting a lower precision compared to VGG16. It's essential to assess the trade-off between precision and recall based on the application requirements. AlexNet: Precision of 9%, similar to LeNet-5. Researchers can use this information to understand the precision performance of AlexNet in comparison to other models. Recall: VGG16: Recall of 9%, The percentage of verified specific example are correctly predicted by the model. This is crucial for applications where identifying all positive instances is important. LeNet-5: Recall of 8%, slightly lower than VGG16. Researchers can analyze this metric to understand how well the model captures positive instances. AlexNet: Recall of 8%, similar to LeNet-5. It offers perceptions into the ability of AlexNet to correctly identify positive instances. F1 Score: VGG16: F1 Score of 10%, is the precision's harmonic mean and recall. It provides a balanced measure of a model's performance. LeNet-5: F1 Score of 9%, slightly lower than VGG16. Researchers can use this metric to assess the overall performance of LeNet-5. AlexNet: F1 Score of 9%, similar to LeNet-5. It offers a combined measure of precision and recall for AlexNet. The overall analysis is that the provided metrics and figures give a thorough comprehension of the models' performance. Scholars can use this information for further analysis, comparisons, and improvements in the field of image recognition and transfer learning. The figures provide visual data regarding the validation and training accuracy over epochs, aiding scholars in evaluating the training process and generalization capabilities.

The Visualization of Training Accuracy and Validation Accuracy of VGG16 in Heat Map is shown in fig 7 and fig 8.

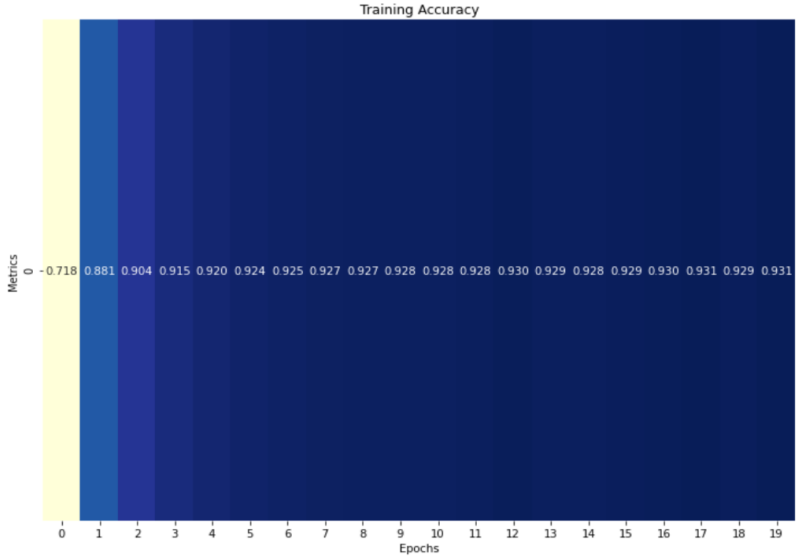

Fig. 7. VGG16 Training Accuracy

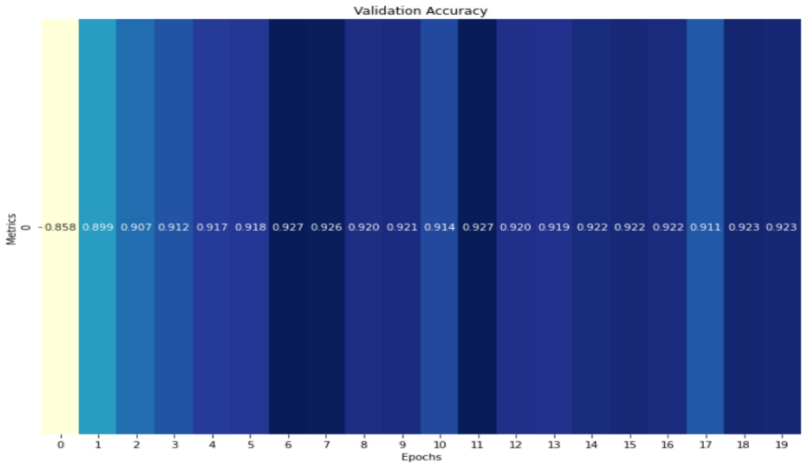

Fig. 8. VGG16 Validation Accuracy

Table 2. Comparing performance of VGG16 for Training Loss Vs. other models

Model	***Loss***	***Precision***	***Recall***	***F1 Score***
VGG16	2%	3%	2%	3%
LeNet-5	3%	4%	4%	3%
AlexNet	3%	3%	4%	3%

As shown in the table 2, it compares the performance metrics, specifically focusing on training loss, precision, recall, and F1 Score, for three different models (VGG16, LeNet-5, and AlexNet). This information is valuable for researchers and practitioners. The explanation for each metric: Training Loss: VGG16: Achieved a training loss of 2%, representing the amount of error in the training stage. Lower training loss indicates better convergence and adaptation of the model to the training data. LeNet-5: Exhibited a training loss of 3%, slightly higher than VGG16. A comparison of training losses helps researchers understand how well the models are learning from the training data. AlexNet: Also demonstrated a training loss of 3%, similar to LeNet-5. Researchers can use this information to assess the effectiveness of AlexNet in minimizing errors during training. Precision: VGG16: Precision of 3%, indicating The accurateness of positive predictions among all positive instances during training. This metric is

essential for assessing the model's ability to avoid false positives. LeNet-5: Showed a precision of 4%, slightly higher than VGG16. Precision during training is crucial for understanding the model's behavior in correctly identifying positive instances. AlexNet: Exhibited a precision of 3%, similar to VGG16. Researchers can analyze the precision of AlexNet during training to make informed comparisons. Recall: VGG16: Recall of 2%, representing the ratio of definite optimistic cases correctly predicted by the model during training. A higher recall indicates the model's effectiveness in capturing positive instances. LeNet-5: Demonstrated a recall of 4%, higher than VGG16. Researchers can assess how well LeNet-5 identifies positive instances during the training phase. AlexNet: Showed a recall of 4%, similar to LeNet-5. This information is valuable for understanding the recall performance of AlexNet during training. F1 Score: VGG16: F1 Score of 3%, which is the harmonic mean of recall and precision during training. It provides a balanced measure of the model's overall performance. LeNet-5: Achieved an F1 Score of 3%, similar to VGG16. Researchers can use this metric to evaluate the combined precision and recall performance of LeNet-5 during training. AlexNet: Demonstrated an F1 Score of 3%, similar to VGG16. It offers views about the balance between precision and recall for AlexNet during training. Overall Analysis: The provided metrics in the table offer a detailed comparison of training performance for the three models. Scholars can use this information to assess how well each model learns from the training data in terms of minimizing loss and correctly predicting positive instances. The values can be crucial for making informed decisions about model selection and improvements in image recognition tasks.

The Visualization of Training Loss and Validation Loss of VGG16 in Heat Map shown in the fig 9 and fig 10.

Fig. 9. VGG16 Training Loss

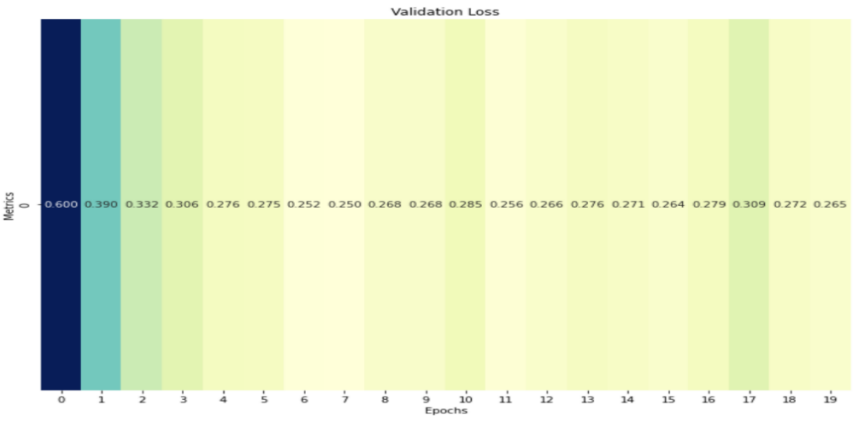

Fig. 10. VGG16 Validation Loss

7. Conclusion

In conclusion, the study objective was to improve picture identification on the MNIST dataset by using the VGG16 model in a Convolutional Neural Network (CNN) with a focus on transfer learning. The experimental framework was developed upon the MNIST database, which is fundamental in handwritten digit recognition. The primary methods include leveraging the capabilities of VGG16, a well-established CNN architecture, and utilizing transfer learning, in which details extracted from the ImageNet dataset was repurposed for MNIST. The materials and methods section described a methodical approach that included data pretreatment, model construction, and training phases. The research employed essential tools such as TensorFlow and Kera's, ensuring a robust and efficient implementation. The procedure involved careful adjustments to adapt MNIST data for VGG16, aligning the model's architecture with the unique characteristics of the handwritten digit dataset. Results demonstrated successful data processing, aligning the MNIST dataset with VGG16 requirements. The discussion delved into the significance of the MNIST database for machine learning, emphasizing the potential improvement in performance by employing VGG16 for image classification. The use of transfer learning was highlighted as a key strategy, allowing the model to benefit from previously learned features while adapting to the specific nuances of the MNIST dataset. This research adds to the broader landscape of image recognition by exploring the uses of transfer learning with the VGG16 model on the MNIST dataset. While the study was limited in scope to these specific elements, the findings has shown the process for more research into optimizing CNNs for handwritten digit recognition tasks. The methodologies and insights presented here can act as a basis for upcoming research projects aimed at improving the accuracy and effectiveness of image recognition. Overall, this research offers very useful data about the possibilities of transfer learning with VGG16 for enhancing image recognition capabilities on the MNIST dataset.

References

[1] Moath, Alsafasfeh., Mohanad, Alhasanat. "Image compression approach for improving deep learning applications." International Journal of Power Electronics and Drive Systems, vol. 13, no. 5, 2023. doi: 10.11591/ijece.v13i5.pp5607-5616

[2] Jungeum, Kim, Xiao Wang"Robust sensible adversarial learning of deep neural networks for image classification." The Annals of Applied Statistics, vol. 17, no. 2, 2023. doi: 10.1214/22-aoas1637

[3] Wu,J.,He,J.&Anisworth,E."Non-IID Transfer Learning on Graphs." Proceedings of the ... AAAI Conference on Artificial Intelligence, vol. 37, no. 9, 2023. doi: 10.1609/aaai.v37i9.26231

[4] Kyung, Joo, Cheoi., Hyeon-yeong, Choi., Jaepil, Ko. "Empirical Remarks on the Translational Equivariance of Convolutional Layers." Applied Sciences, vol. 10, no. 9, 2020. doi: 10.3390/APP10093161

[5] Jaya, H., Dewan., Rik, Das., Sudeep, D., Thepade., Sinu, Nambiar. "Image Classification by Transfer Learning using Pre-Trained CNN Models.", 2023. doi: 10.1109/RAEEUCCI57140.2023.10134069

[6] Vayos, Liapis. "Understanding CNN fragility when learning with imbalanced data." Machine Learning, 2023. doi: 10.1007/s10994-023-06326-9

[7] Khoirin, Nisa., Mustofa, Usman. "Implementation of decision tree and support vector machine on raisin seed classification." Jurnal Aksioma, vol. 12, no. 1, 2023. doi: 10.24127/ajpm.v12i1.6873

[8] Jonathan, Janke., Mauro, Castelli., Aleš, Popovič., Aleš, Popovič. "Analysis of the proficiency of fully-connected neural networks in the process of classifying digital images : benchmark of different classification algorithms on high-level image features from convolutional layers." Expert Systems With Applications, vol.135,no. C 2019. doi: 10.1016/J.ESWA.2019.05.058

[9] Anup, Kumar., Gianmauro, Cuccuru., Björn, Grüning., Rolf, Backofen. "An accessible infrastructure for artificial intelligence using a Docker-based JupyterLab in Galaxy." GigaScience, vol. 12,2023. doi: 10.1093/gigascience/giad028

[10] Ibrahim Ahmed, Marcos Quinones-Grueiro, Gautam Biswas, "Model-based adaptation for sample efficient transfer in reinforcement learning control of parameter-varying systems.", 2023. doi: 10.48550/arxiv.2305.12158

[11] Manfredo, Atzori., R., Vázquez. "An Overview of Open Source Deep Learning-Based Libraries for

Neuroscience." Applied Sciences, vol. 13, no. 9, 2023. doi: 10.3390/app13095472

[12] Yang, J., S. Guo, G. Wu, and L. Wang. "CoMAE: Single Model Hybrid Pre-training on Small-Scale RGB-D Datasets." Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 3, June 2023, doi: 10.1609/aaai.v37i3.25419

[13] Luca, Negrini., Caterina, Urban. "Static Analysis of Data Transformations in Jupyter Notebooks.", 2023. doi: 10.1145/3589250.3596145

[14] Tesfaye Fufa Gedefa, Galety Mohammed Gouse, Garamu Tilahun Iticha, "Empowering Scientific Computing and Data Manipulation with Numerical Python (NumPy)." Advances in systems analysis, software engineering, and high performance computing book series, 2023. doi: 10.4018/978-1-6684-7100-5.ch007

[15] O. Vishwakarma, S. Agarwal, K. K. Soundra Pandian, M. Syafrullah, K. Adiyarta and S. K. Sonbhadra, "Anomaly detection on MNIST stroke simulation dataset.", 9th International Conference on Electrical Engineering, 2022. doi: 10.23919/eecsi56542.2022.9946546.