International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Transfer Learning: Performance Analysis of VGG-16 on CIFAR-10

*Dr. Vijay Kumar Samyal [a], Harsh Gupta [b], Karamveer [c], Dinesh Puri [d]*

[a] *Associate Professor, Department of CSE, MIMIT Malout*
[b,c,d] *Students, Department of CSE, MIMIT Malout*

**ABSTRACT**

Image classification[1] is a critical task in computer vision, often requiring large labeled datasets for training deep learning models. Transfer learning offers a promising solution by leveraging pre-trained models on large, well-labeled datasets and fine-tuning them for specific tasks with limited data. This paper explores the effectiveness of transfer learning for improving image classification accuracy, using the CIFAR-10 [2] dataset as a benchmark. We employ a standard Convolutional Neural Network (CNN) as a baseline and compare its performance to the pre-trained VGG-16 model [9]. Through experiments, we analyze the impact of transfer learning on model accuracy, training time, and generalization. Our findings demonstrate the advantages of transfer learning in enhancing classification accuracy and efficiency, even in cases where sufficient labeled data is available. The paper concludes by discussing the broader implications of transfer learning for image classification tasks and its potential for adoption in diverse application domains.

Keywords: Transfer Learning, Image Classification, VGG-16,CIFAR-10,CNN

## 1. Introduction

In the realm of image recognition, the evolution of deep learning has played a pivotal role in achieving remarkable strides in accuracy and efficiency. However, the success of deep learning models is often contingent upon the availability of vast labeled datasets and considerable computational resources for training. **Transfer learning, a paradigm within machine learning, addresses these challenges by enabling models to leverage knowledge gained from one domain to enhance performance in another**. This paper explores the application of transfer learning in the context of image recognition, with a specific emphasis on harnessing the capabilities of pre-trained models to elevate overall performance. Image recognition is a critical component of various applications, ranging from autonomous vehicles and medical diagnosis to facial recognition systems and augmented reality. Traditional deep learning approaches require substantial amounts of labeled data for training, making them less practical in scenarios where annotated datasets are limited. Transfer learning offers a compelling solution to this predicament, allowing models to capitalize on knowledge acquired from tasks with abundant data, thus mitigating the need for extensive labeled datasets in the target domain.

The core concept of transfer learning lies in the idea that a model pre-trained on a large dataset for a specific task can serve as a feature extractor with learned hierarchical representations. These representations, encapsulated in the weights of the pre-trained model, capture generic features that are transferable across different but related tasks. In the realm of image recognition, convolutional neural networks (CNNs) have demonstrated remarkable success. This paper focuses on the utilization of pre-trained CNNs, such as VGG16 to investigate the efficacy of transfer learning. Our research aims to address several key questions. What improvements in classification accuracy can be achieved by leveraging pre-trained models? What are the effects on model convergence? The findings of this study hold significant implications for practitioners and researchers alike, offering insights into optimizing image recognition models by strategically incorporating transfer learning techniques. As we delve into the experimental methodology and results, this paper seeks to contribute valuable knowledge to the broader discourse on transfer learning and its role in advancing the field of image recognition.

## 2. Theoretical Background

### *2.1 Convolutional Neural Networks (CNN)*

A convolutional neural network (CNN) is a specialized form of feed-forward neural network that incorporates regularization through the optimization of filters or kernels. These networks autonomously learn features and have been widely utilized for analyzing and predicting outcomes across various data types, including text, images, and audio [7].

A typical CNN architecture is composed of an input layer, a series of hidden layers, and an output layer. Among the hidden layers, one or more layers are dedicated to performing convolution operations, which involve the application of a kernel to the input matrix. This operation generally computes the

Frobenius inner product between the kernel and the input, with the result passed through an activation function, often ReLU [10]. As the kernel traverses the input matrix, it produces a feature map that serves as input for subsequent layers. Additional components such as pooling layers, normalization layers, and fully connected layers are commonly integrated into the network to enhance its functionality [8].

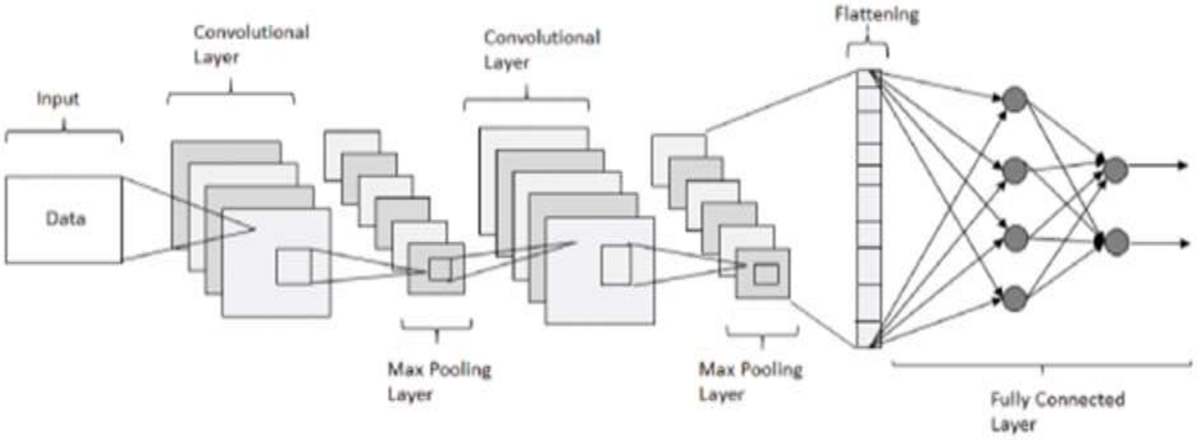


Figure 1 Schematic view of a Convolutional Neural Network [3]

### *2.2 Transfer Learning*

Transfer learning is an effective strategy in deep learning that utilizes a model pre-trained on a large, diverse dataset to enhance performance on a smaller, related dataset. This method significantly reduces training time and boosts model accuracy, especially in scenarios like image classification, where acquiring and annotating extensive datasets can be costly and time-consuming. The essence of transfer learning lies in reusing the knowledge encoded in pre-trained models. These models, often trained on comprehensive datasets such as ImageNet, learn to identify fundamental features like edges, textures, and shapes, which are applicable to a wide range of image recognition tasks. By applying pre-trained models to new datasets, it becomes possible to exploit their ability to detect intricate visual patterns without the need to construct and train a new model from the ground up.

**Transfer learning typically involves two main techniques: feature extraction and fine-tuning.**

**Feature extraction** leverages a pre-trained model as a fixed feature extractor. In this method, the convolutional base of the pre-trained model remains unchanged, while the final classification layers are replaced or modified to fit the new task. The convolutional layers of the pre-trained model extract features from the input data, which are then passed to a newly designed classifier specific to the target problem. This approach is particularly advantageous when working with small datasets, as it relies on the robust feature extraction capabilities of the pre-trained model and reduces the risk of overfitting.

**Fine-tuning** on the other hand, involves retraining some or all of the pre-trained model's layers on the new dataset. This allows the model to adapt its learned features to the specific patterns of the new data. Fine-tuning is especially effective when the new dataset is large and varied enough to justify updating the pre-trained model's parameters. The process includes adjusting the learning rate and selectively unfreezing certain layers of the pre-trained model, enabling further refinement of features and yielding better performance on the target task [4].

### *2.3 CIFAR-10 Dataset*

The CIFAR-10 dataset is composed of 60,000 32x32 color images, divided into 10 distinct classes, with each class containing 6,000 images. The dataset is split into 50,000 training images and 10,000 test images.

The dataset is organized into five training batches and one test batch, each comprising 10,000 images. The test batch includes exactly 1,000 randomly chosen images from each class, ensuring uniform representation. Meanwhile, the training batches, which contain the remaining images, are arranged in random order. Although each class contributes exactly 5,000 images across all training batches, individual batches may have varying distributions of class images.

The dataset classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. These classes are entirely distinct, with no overlap between them. For example, "automobile" includes vehicles such as sedans and SUVs, while "truck" refers exclusively to large trucks and does not include pickup trucks [2].

### *2.4 Pretrained Models and VGG-16*

Pre-trained models are integral to the advancement of deep learning, especially in the context of transfer learning for image classification. These models are trained on extensive datasets, such as ImageNet, which encompass a vast and diverse collection of images. Consequently, they develop robust feature extraction capabilities that can be readily adapted to various new tasks with minimal additional training [4].

The VGG-16 model, developed by the Visual Geometry Group (VGG) at the University of Oxford, is a convolutional neural network (CNN) architecture known for its depth and simplicity. The model consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG-16 has gained recognition for its effectiveness and strong performance across multiple computer vision tasks, such as image classification and object recognition. Its architecture employs a series of convolutional layers interspersed with max-pooling layers, with the network depth increasing progressively through the layers.

The VGG-16 configuration typically consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. These layers are organized into blocks, with each block containing multiple convolutional layers followed by a max-pooling layer for down sampling.

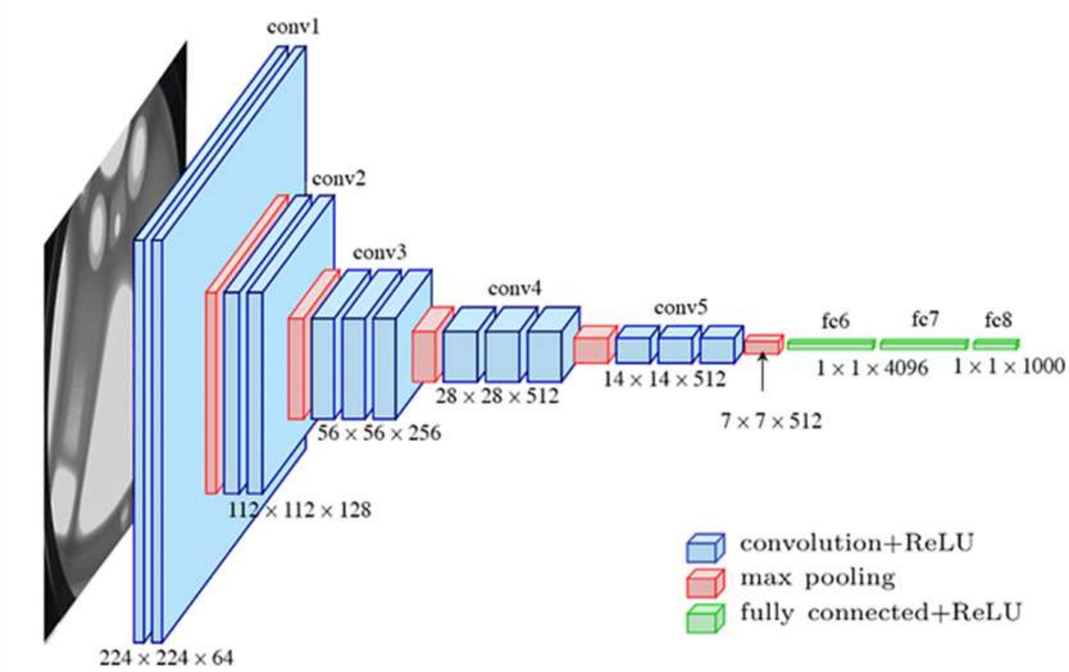


**Figure 2 [5]**

## 3. Methodology

### *3.1 Overview*

The objective of this study is to **investigate the effectiveness of transfer learning for image classification, comparing its performance against a baseline convolutional neural network (CNN) trained from scratch**. Transfer learning leverages the knowledge gained from a pre-trained model, typically trained on a large and diverse dataset, to adapt to a different but related target task. This approach often provides improved performance in scenarios with limited labeled data or complex patterns.

To conduct this comparison, we utilized the **CIFAR-10 dataset**, which contains 60,000 images divided into 10 distinct classes. The dataset serves as a benchmark for evaluating image classification tasks due to its variability and balanced class distribution. Our baseline model consists of a traditional CNN architecture with multiple convolutional layers, batch normalization, pooling, dropout layers, and fully connected layers. For the transfer learning approach, we used the **VGG-16 model** pre-trained on the ImageNet dataset, adapting its architecture to fit the CIFAR-10 classification task.

This section outlines the detailed methodologies used to develop and compare the models, including data preprocessing techniques, model architectures, training configurations, and performance evaluation strategies. This section is largely based on the implementation and examples provided in [6], with modifications and explanations tailored to this study's context.

### *3.2 Getting Data*

The CIFAR10 dataset is available through keras load_data() API [11]. It by default returns train and test datasets along with their corresponding target labels.

Once dataset is downloaded, we perform a quick scaling by multiplying with a scalar quantity of 1/255.The dataset consists of a total of 60,000 samples with 10 different classes. It provides 50k samples in training set and the rest in test dataset

```
from tensorflow.keras.datasets import cifar10
# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train/255.
x_test = x_test/255.
```

### *3.3 Baseline Model (Standard CNN)*

Using keras we prepare a simple CNN by stacking Convolutional, Maxpooling, Batch Normalization, Flatten and Dense Layers. We utilize Dropouts to counter overfitting.

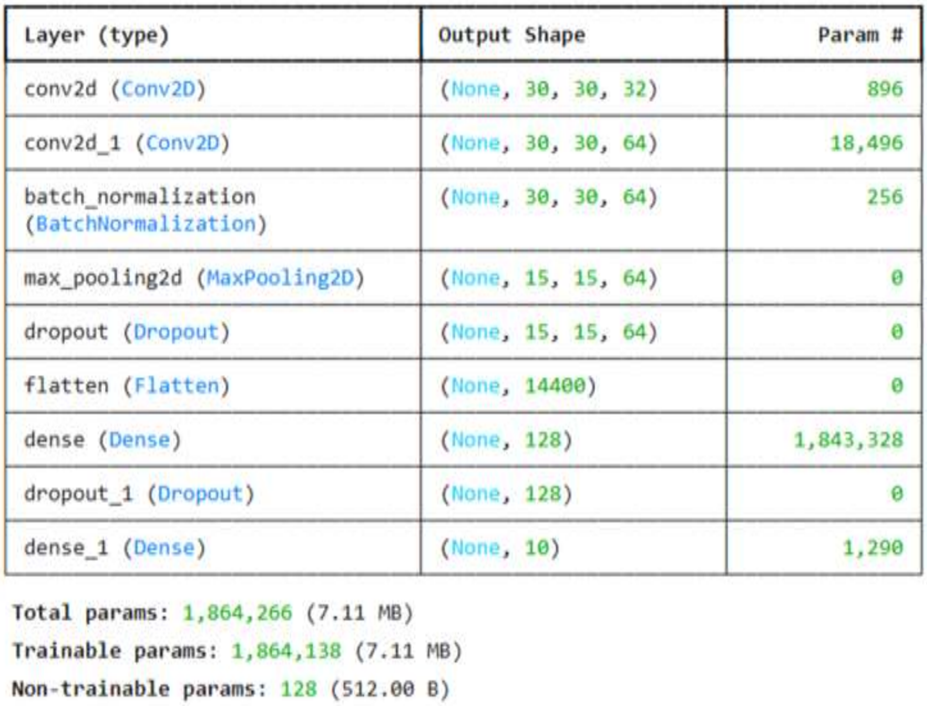
```
# Model definition
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=INPUT_SHAPE))

model.add(Conv2D(64, (3,3),padding='same',
                kernel_regularizer=regularizers.l2(WEIGHT_DECAY),
                activation='relu',))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
```

```
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

- **Convolutional Layers with ReLU Activation**: Using two convolutional layers with ReLU activation ensures non-linearity and captures important spatial features effectively.
- **Regularization (Weight Decay):** Applying weight decay helps reduce overfitting, making the model more robust.
- **Batch Normalization**: This helps normalize the outputs of the convolutional layer, speeding up convergence and potentially improving model performance.
- **Pooling and Dropout**: Max pooling reduces spatial dimensions and helps control overfitting, while dropout layers reduce co-adaptation of neurons by randomly setting a fraction of input units to zero during training.
- **Fully Connected Layer with Dropout**: Adding a dense layer with 128 units provides a sufficient number of parameters to learn non-linear combinations of features, and dropout here further regularizes the network.
- **Softmax Output**: Suitable for multi-class classification tasks like CIFAR-10.

### *3.4 Pre-trained VGG-16 Model (Transfer Learning)*



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 30, 30, 32) | 896 |
| conv2d_1 (Conv2D) | (None, 30, 30, 64) | 18,496 |
| batch_normalization (BatchNormalization) | (None, 30, 30, 64) | 256 |
| max_pooling2d (MaxPooling2D) | (None, 15, 15, 64) | 0 |
| dropout (Dropout) | (None, 15, 15, 64) | 0 |
| flatten (Flatten) | (None, 14400) | 0 |
| dense (Dense) | (None, 128) | 1,843,328 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_1 (Dense) | (None, 10) | 1,290 |

```
Total params: 1,864,266 (7.11 MB)
Trainable params: 1,864,138 (7.11 MB)
Non-trainable params: 128 (512.00 B)
```

**Figure 3 Model summary**

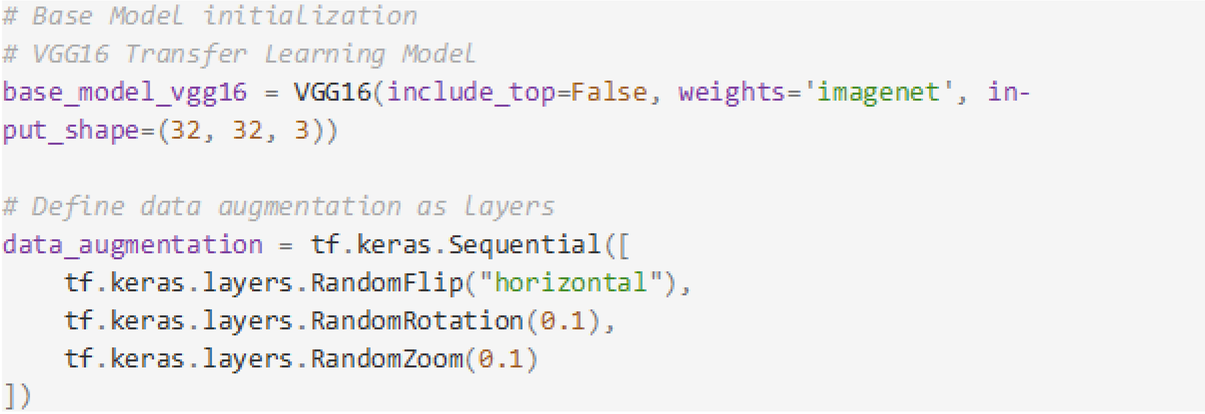
To enhance the image classification accuracy with limited labeled data, we employed the VGG-16 architecture pre-trained on the ImageNet dataset **as a feature extractor**. The use of VGG-16 enables leveraging the rich, generic features learned from large-scale data, minimizing the data requirements for effective feature extraction in our target domain.

**1. Input Preprocessing**

To improve the model's generalization and mimic variations in real-world data, data augmentation was incorporated directly into the model pipeline. This included random horizontal flipping, slight rotations (up to 10%), and zooming (up to 10%). These augmentations simulate transformations such as shifts or rotations without significantly altering the dataset's characteristics.

**2. Base Model Initialization**

The VGG-16 model was loaded without its dense layers (include_top=False) to focus exclusively on its convolutional layers for feature extraction. The convolutional layers of VGG-16 served as a **feature extractor**, effectively learning hierarchical patterns from the input images. The layers up to this point remain frozen during training to retain pre-trained weights.



```
# Base Model initialization
# VGG16 Transfer Learning Model
base_model_vgg16 = VGG16(include_top=False, weights='imagenet', in-
put_shape=(32, 32, 3))

# Define data augmentation as layers
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1)
])
```

**3. Custom Classification Layers**

To adapt the features from the pre-trained model to our specific image classification task, we added a custom classifier:

- **Flatten Layer**: Converts the multi-dimensional feature maps into a one-dimensional vector for input into fully connected layers.

- **Dense Layers**: Four dense layers (1024, 512, 256, and 128 units, respectively) were introduced with ReLU activation to enable deeper representation learning. These layers progressively reduced dimensionality while learning complex feature interactions.
- **Output Layer**: A final dense layer with 10 units (for 10 classes) and with softmax activation was used to classify images into the desired number of classes.

```
# Define the model with augmentation

inputs = Input(shape=(32, 32, 3))
x = data_augmentation(inputs)  # data augmentation
x = base_model_vgg16(x)  # VGG16 as the base model
x = Flatten()(x)  # Flatten the output
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
outputs = Dense(10, activation='softmax')(x)  # Final softmax output

# Create the model
model_vgg16 = Model(inputs, outputs)
```

**4. Training Setup**

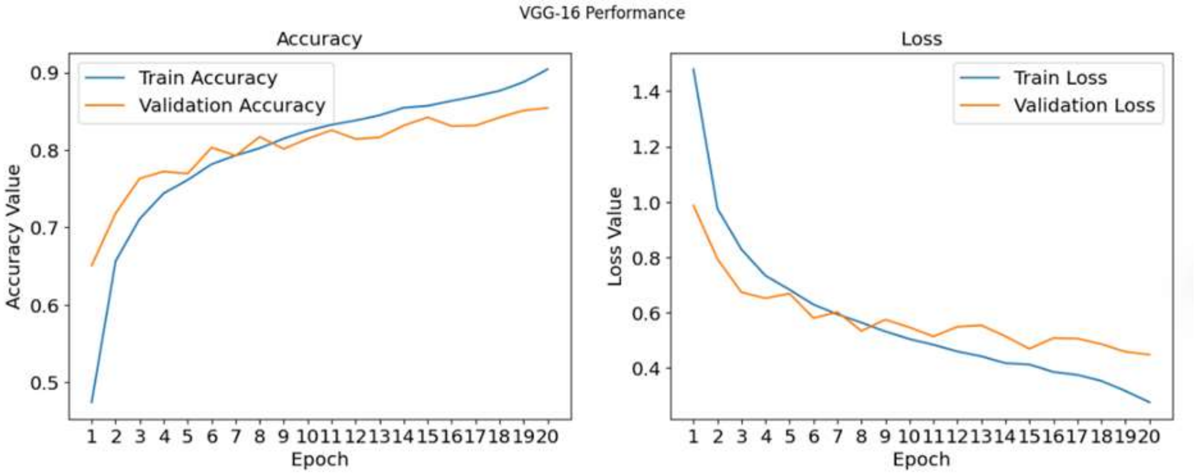
We focused exclusively on training the custom classification layers by freezing all pre-trained layers in the base VGG-16 model. This allowed the network to learn domain-specific features efficiently while leveraging existing knowledge from the pre-trained layers. The model was compiled using the SGD optimizer with a learning rate of 0.001, momentum of 0.9.and categorical cross-entropy loss.

## 4. Results and Discussions

### *4.1 Model Performance Overview*

In this study, two models were compared for the image classification task: a baseline Convolutional Neural Network (CNN) and a pre-trained VGG-16 model fine-tuned using transfer learning. The key evaluation metrics were accuracy and loss over epochs. The results are summarized below:

- **Baseline CNN Performance**:
  - **Test Accuracy**: 69%
  - **Training vs. Validation**: The baseline CNN model achieved a maximum training accuracy of 83% and a validation accuracy of 71%. This demonstrates a moderate capability to generalize to unseen data but is limited by its simpler architecture.
  - **Loss Curves**: The training and validation loss curves indicate overfitting. Though the model performance is commendable, there are signs of overfitting. The validation loss starts increasing after a few epochs along with flattening of the validation accuracy.
- **VGG-16 Model Performance**:
  - **Test Accuracy**: 84.46%
  - **Training vs. Validation**: The pre-trained VGG-16 model achieved a maximum training accuracy of 90% and a validation accuracy of 85.42%, indicating a better overall generalization compared to the baseline CNN.
  - **Loss Curves**: Both train and validation losses decrease and appear to stabilize, suggesting that the model is learning effectively without significant overfitting at least up to the observed epochs. A decreasing trend suggests that the model generalizes well to unseen data.



- **Observations on Transfer Learning**: The VGG-16 model, which uses transfer learning, demonstrates the power of pre-trained networks to leverage prior knowledge, even when fine-tuned on a different dataset. The impressive value of 84.46% test accuracy highlights the value of transfer learning.

### *4.2 Comparative Analysis*

1. **Accuracy Gains**:
   - The improvement in test accuracy to 84% indicates that the pre-trained features learned from a large dataset (like ImageNet) can be effectively transferred and fine-tuned to solve the current classification task. The results suggest that feature extraction through transfer learning leads to useful feature representations that yield impressive accuracy scores.
2. **Impact of Network Depth**:
   - The VGG-16 model, being deeper, inherently captures more complex feature hierarchies compared to the baseline CNN. However, this also comes with the need for careful regularization to avoid overfitting on the smaller target dataset.
3. **Data Augmentation**:
   - It should be noted that data augmentation was applied only to the VGG-16 model. This may have contributed to its improved generalization. Future studies could extend the use of augmentation to the baseline CNN for a more balanced baseline.

### *4.3 Challenges and Limitations*

- **Overfitting**: While the VGG-16 model achieved higher training accuracy (90%), the test accuracy plateaued at 84.46%, which may point towards some overfitting during training. More extensive data augmentation, different regularization strategies, or fine-tuning deeper layers might help mitigate this.
- **Resource Constraints**: Further investigation revealed that variations of VGG architectures with more complex dense layers and enhanced data augmentation strategies can push test performance even higher. This suggests that the architecture and training strategy used in this study may not fully harness the potential of VGG-16, indicating a limitation in the chosen configurations.
- **Computational Cost**: The VGG-16 model is computationally heavier and requires more resources compared to the simpler CNN. This trade-off between performance and resource demand needs consideration, especially in real-world deployments.

### *4.4 Insights and Future Work*

- **Feature Importance and Interpretation**: Analyzing the learned features can offer further insights into what the models focus on for classification.
- **Potential Extensions**: Exploring other pre-trained models like ResNet or EfficientNet, or experimenting with ensemble approaches, could yield better results.
- **Model Fine-Tuning Strategies**: Future work may also investigate strategies like unfreezing more layers of the pre-trained model or adopting more advanced optimization techniques.

## 5. Conclusion

This study investigated the efficacy of transfer learning for image classification by comparing a baseline CNN model to a pre-trained VGG-16 model used as a feature extractor. Through rigorous experimentation, we observed that transfer learning significantly improved the model's ability to generalize, with VGG-16 achieving a **test accuracy of 84.46%** compared to the baseline CNN's 69%. The use of pre-trained features enabled faster convergence and better handling of complex image features in our classification task, underscoring the potential of leveraging pre-existing, deep architectures for low-resource settings.

However, our results also highlighted important challenges, such as the sensitivity of performance to data augmentation strategies and architectural modifications. We found that further enhancement using more complex dense layers and aggressive data augmentation could push accuracy beyond 90%, indicating room for optimization. Overall, this work reaffirms that transfer learning can be a powerful tool in image classification tasks, particularly when resources and labeled data are constrained. Future research should focus on maximizing transfer learning's potential through tailored architectural adaptations and fine-tuning strategies.

## 6. References

[1] Chaudhari, T. (2014). Image Classification: A Short Review. *International Journal of Advanced Electronics and Communication Systems*.

[2] Krizhevsky, A. (2010). Convolutional Deep Belief Networks on CIFAR-10. Retrieved November 17, 2024, from https://www.cs.toronto.edu/~kriz/cifar.html

[3] Dastile, X., & Celik, T. (2021). Making deep learning-based predictions for credit scoring explainable. *IEEE Access, 9*, 50426–50440. https://doi.org/10.1109/ACCESS.2021.3068854

[4] Abbas, T. & Gasmi, S. (2024). Leveraging Transfer Learning in PyTorch for Effective Image Classification. *ResearchGate*. https://doi.org/10.13140/RG.2.2.30294.25920

[5] B R, N., A, G. K., H S, C., M S, D., & S, M. (2021). An Ensemble Deep Neural Network Approach for Oral Cancer Screening. *International Journal of Online and Biomedical Engineering (iJOE)*, *17*(02), pp. 121–134. https://doi.org/10.3991/ijoe.v17i02.19207

[6] Sarkar, D., Bali, R., & Ghosh, T. (2018). *Hands-on transfer learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. Packt Publishing.

[7] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature*, *521*, 436–444 (2015). https://doi.org/10.1038/nature14539

[8] Stankovic, L., & Mandic, D. (2021). Convolutional neural networks demystified: A matched filtering perspective-based tutorial. *arXiv*. https://arxiv.org/abs/2108.11663v3

[9] Fan, Y., Yang, J., Han, K., Lin, J., & Jia, J. (2023). VGG-16 image recognition model based on band-stop convolution filter preprocessing. In *Proceedings of the 8th International Conference on Intelligent Computing and Signal Processing (ICSP)* (pp. 1685–1690). IEEE. https://doi.org/10.1109/ICSP58490.2023.10248650

[10] Keras Team. (n.d.). ReLU function. *Keras*. Retrieved November 17, 2024, from https://keras.io/api/layers/activations/#relu-function

[11] Keras Team. (n.d.). Load Data function. *Keras*. Retrieved November 17, 2024, from https://keras.io/api/datasets/cifar10/#loaddata-function