ISSN: 2249-7196

IJMRR/April. 2025/ Volume 15/Issue 2s/270-274

Image Classification using Transfer Learning

Vasa Pavan Naga Sai Yesu
PG scholar, Department of MCA, DNR College, Bhimavaram, Andhra Pradesh.
K.Sridevi
(Assistant Professor), Master of Computer Applications, DNR college, Bhimavaram, Andhra Pradesh.

ABSTRACT Image classification, a fundamental task in computer vision, plays a pivotal role in various applications, including object recognition, medical imaging, and autonomous vehicles. However, training deep neural networks from scratch for image classification often requires vast amounts of labeled data and computational resources. Transfer learning offers a practical solution to this challenge by leveraging pre-trained models and transferring knowledge from one task to another. This paper investigates the application of transfer learning in image classification tasks. By fine-tuning pre-trained convolutional neural networks (CNNs) on new datasets, we demonstrate the effectiveness of transfer learning in achieving high classification accuracy with limited labeled data. We explore different transfer learning strategies, including feature extraction and fine-tuning of entire networks, and evaluate their performance on benchmark datasets. Through experimentation, we highlight the advantages of transfer learning in reducing training time, alleviating the need for extensive labeled data, and improving generalization performance. We also discuss practical considerations and best practices for selecting pre-trained models, adapting them to new tasks, and fine-tuning hyperparameters.

I. INTRODUCTION

Image classification, a fundamental task in computer vision, involves categorizing images into predefined classes or categories. While deep learning has revolutionized image classification, training deep neural networks from scratch often requires substantial computational resources and large labeled datasets. Transfer learning offers a practical solution to this challenge by leveraging pre-trained models trained on massive datasets such as ImageNet and transferring their learned features to new tasks with limited labeled data. In this

study, we explore the application of transfer learning in image classification tasks.

We employ pre-trained convolutional neural networks (CNNs), such as VGG, ResNet, or Inception, as feature extractors. These networks have learned to detect generic features like edges, textures, and shapes, which are useful for various image recognition tasks. By fine-tuning the pre-trained CNNs on specific datasets related to our classification task, we adapt the learned features to better suit our target problem.

Fine-tuning allows the model to adjust its parameters to better capture the nuances of the new dataset while still benefiting from the general features learned from the pre-training phase. Through experimentation, we demonstrate the effectiveness of transfer learning in achieving high classification accuracy with relatively small labeled datasets. We compare different transfer learning strategies, including feature extraction and fine-tuning, and evaluate their performance on benchmark datasets.

II. LITERATURE SURVEY

[1] Gupta, J., Pathak, S., & Kumar, G. (2022). Bare skin image classification using convolution neural netowrk. International Journal of Emerging Technology and Advanced Engineering, 12(01).

Image classification is critical and significant research problems in computer vision applications such as facial expression classification, satellite image classification, and plant classification based on images. Here in the paper, the image classification model is applied for identifying the display of daunting pictures on the internet. The proposed model uses Convolution neural network to identify these images and filter them through different blocks of the network, so that it can be classified accurately.

[2] Hameed, Z., Zahia, S., Garcia-Zapirain, B., Javier Aguirre, J., & María Vanegas, A. (2020). Breast cancer histopathology image classification using an ensemble of deep learning models. Sensors, 20(16), 4373.

Breast cancer is one of the major public health issues and is considered a leading cause of cancer-related deaths among women worldwide. Its early diagnosis can effectively help in increasing the chances of survival rate. To this end, biopsy is usually followed as a gold standard approach in which tissues are collected for microscopic analysis. However, the histopathological analysis of breast cancer is non-trivial, labor-intensive, and may lead to a high degree of disagreement among pathologists. Therefore, an automatic diagnostic system could assist pathologists to improve the effectiveness of diagnostic processes.

[3] Indraswari, R., Rokhana, R., & Herulambang, W. (2022). Melanoma image classification based on MobileNetV2 network. Procedia Computer Science, 197, 198-207.

Melanoma is one of the most common types of cancer that can lead to high mortality rates if not detected early. Recent studies about deep learning methods show promising results in the development of computer-aided diagnosis for accurate disease detection. Therefore, in this research, we propose a method for classifying melanoma images into benign and malignant classes by using deep learning model and transfer learning. MobileNetV2 network is used as the base model because it has lightweight network architecture.

[4] Rajpal, S., Lakhyani, N., Singh, A. K., Kohli, R., & Kumar, N. (2021). Using handpicked features in conjunction with ResNet-50 for improved detection of COVID-19 from chest X-ray images. Chaos, Solitons & Fractals, 145, 110749.

Coronaviruses are a family of viruses that majorly cause respiratory disorders in humans. Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is a new strain of coronavirus that causes the coronavirus disease 2019 (COVID-19). WHO has identified COVID-19 as a pandemic as it has spread across the globe due to its highly contagious nature. For early diagnosis of COVID-19, the reverse transcription-polymerase chain reaction (RT-PCR) test is commonly done. However, it suffers from a high false-negative rate of up to 67% if the test is done during the first five days of exposure.

[5] Behera, S. K., Rath, A. K., & Sethy, P. K. (2021). Maturity status classification of papaya fruits based on machine learning and transfer learning approach. Information Processing in Agriculture, 8(2), 244-250.

Papaya (Carica papaya) is a tropical fruit having commercial importance because of its high nutritive and medicinal value. The packaging of papaya fruit as per its maturity status is an essential task in the fruit industry. The manual grading of papaya fruit based on human visual perception is time-consuming and destructive. The objective of this paper is to suggest a novel non-destructive maturity status classification of papaya fruits. The paper suggested two approaches based on machine learning and transfer learning for classification of papaya maturity status. Also, a comparative analysis is carried out with different methods of machine learning and transfer learning.

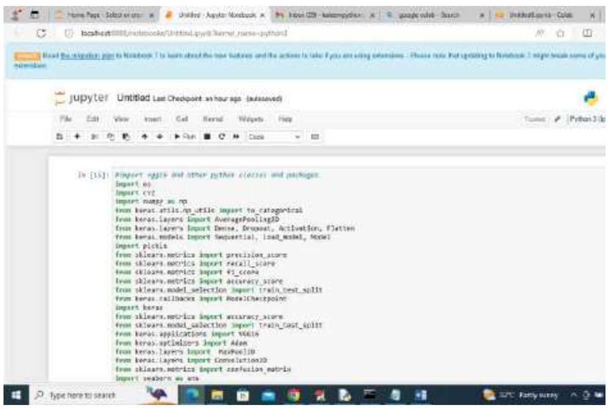
III. PROPOSED METHOD

In this task we have evaluated performance of Pre-trained VGG16 algorithm by freezing and unfreezing its layers and while experimenting we have found that training with unfreezing layers giving better accuracy. We have further evaluated performance of unfreeze VGG16 by performing prediction on old validation dataset which used for freeze model and this model give equal accuracy for unknown validation set.

Each version of VGG16 performance is measured by using various metrics such as accuracy, confusion matrix, precision, recall and FSCORE. We have coded this project using JUPYTER notebook and below are the code and output screens with blue colour comment

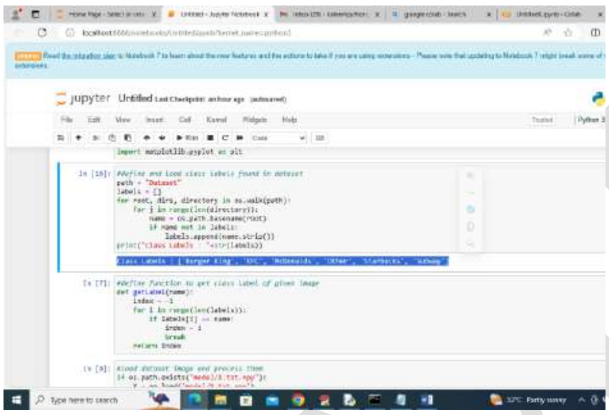
IV. RESULT

Each version of VGG16 performance is measured by using various metrics such as accuracy, confusion matrix, precision, recall and FSCORE. We have coded this project using JUPYTER notebook and below are the code and output screens with blue colour comments



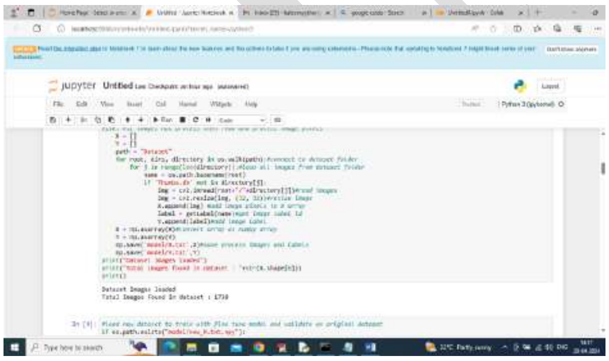
```
In [15]: #import vgg16 and other python classes and packages
         import os
         import cv2
         import numpy as np
         from keras.utils.np_utils import to_categorical
         from keras.layers import AveragePooling2D
         from keras.layers import Dense, Dropout, Activation, Flatten
         from keras.models import Sequential, load_model, Model
         import pickle
         from sklearn.metrics import precision_score
         from sklearn.metrics import recall_score
         from sklearn.metrics import f1_score
         from sklearn.metrics import accuracy_score
         from sklearn.model_selection import train_test_split
         from keras.callbacks import ModelCheckpoint
         import keras
         from sklearn.metrics import accuracy_score
         from sklearn.model_selection import train_test_split
         from keras.applications import VGG16
         from keras.optimizers import Adam
         from keras.layers import MaxPool2D
         from keras.layers import Convolution2D
         from sklearn.metrics import confusion_matrix
         import seaborn as sns
```

In above screen importing required python classes and packages



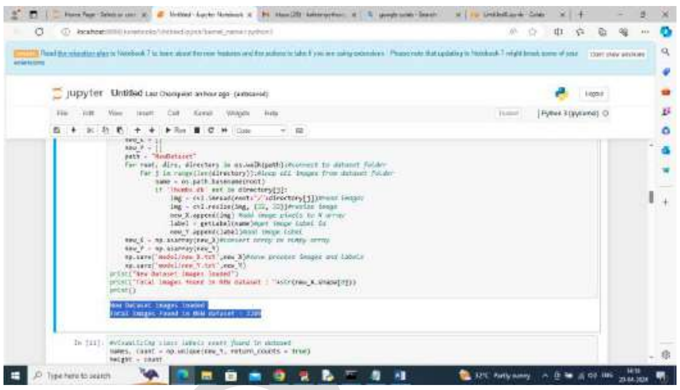
```
         import matplotlib.pyplot as plt

In [16]: #define and load class labels found in dataset
         path = "Dataset"
         labels = []
         for root, dirs, directory in os.walk(path):
             for j in range(len(directory)):
                 name = os.path.basename(root)
                 if name not in labels:
                     labels.append(name.strip())
         print("Class Labels : "+str(labels))

         Class Labels : ['Burger King', 'KFC', 'McDonalds', 'Other', 'Starbucks', 'Subway']

In [7]:  #define function to get class label of given image
         def getLabel(name):
             index = -1
             for i in range(len(labels)):
                 if labels[i] == name:
                     index = i
                     break
             return index

In [8]:  #load dataset image and process them
         if os.path.exists('model/X.txt.npy'):
```

In above screen defining function to identify various class labels found in dataset and then we have define function which will return integer class label for given image category



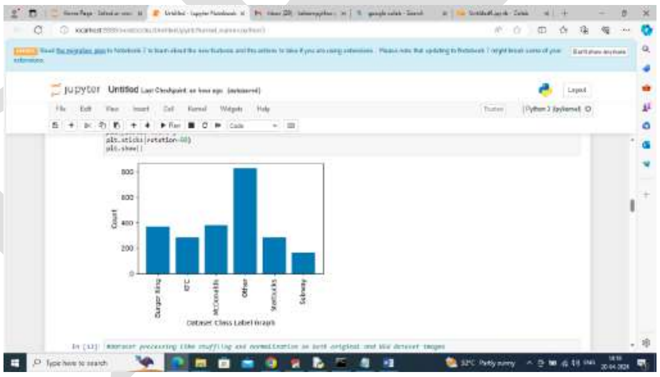
```
    X = []
    Y = []
    path = "Dataset"
    for root, dirs, directory in os.walk(path):#connect to dataset folder
        for j in range(len(directory)):#loop all images from dataset folder
            name = os.path.basename(root)
            if 'Thumbs.db' not in directory[j]:
                img = cv2.imread(root+"/"+directory[j])#read images
                img = cv2.resize(img, (32, 32))#resize image
                X.append(img) #add image pixels to X array
                label = getLabel(name)#get image label id
                Y.append(label)#add image label
    X = np.asarray(X)#convert array as numpy array
    Y = np.asarray(Y)
    np.save('model/X.txt',X)#save process images and labels
    np.save('model/Y.txt',Y)
print("Dataset Images loaded")
print("Total images found in dataset : "+str(X.shape[0]))
print()

Dataset Images loaded
Total Images found in dataset : 1738

In [9]: #load new dataset to train with fine tune model and validate on original dataset
        if os.path.exists("model/new_X.txt.npy"):
```

In above screen we are loading first dataset by reading, resizing and extracting features from each image and then extracting equivalent class label and then creating X and Y training array and then displaying total number of images loaded



```
    new_X = []
    new_Y = []
    path = "NewDataset"
    for root, dirs, directory in os.walk(path):#connect to dataset folder
        for j in range(len(directory)):#loop all images from dataset folder
            name = os.path.basename(root)
            if 'Thumbs.db' not in directory[j]:
                img = cv2.imread(root+"/"+directory[j])#read images
                img = cv2.resize(img, (32, 32))#resize image
                new_X.append(img) #add image pixels to X array
                label = getLabel(name)#get image label id
                new_Y.append(label)#add image label
    new_X = np.asarray(new_X)#convert array as numpy array
    new_Y = np.asarray(new_Y)
    np.save('model/new_X.txt',new_X)#save process images and labels
    np.save('model/new_Y.txt',new_Y)
print("New Dataset images loaded")
print("Total images found in NEW dataset : "+str(new_X.shape[0]))
print()

New Dataset Images loaded
Total Images Found in NEW dataset : 1200

In [11]: #visualizing class labels count found in dataset
         names, count = np.unique(new_Y, return_counts = True)
         height = count
```

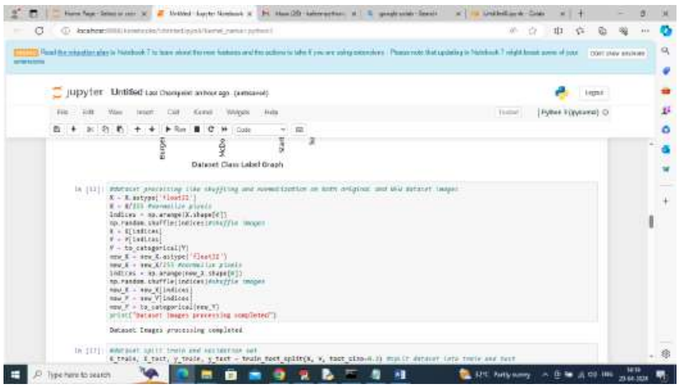
In above screen loading second dataset images and then displaying total number of images loaded



```
plt.xticks(rotation=90)
plt.show()
```

```
In [12]: #dataset processing like shuffling and normalization on both original and NEW dataset images
```

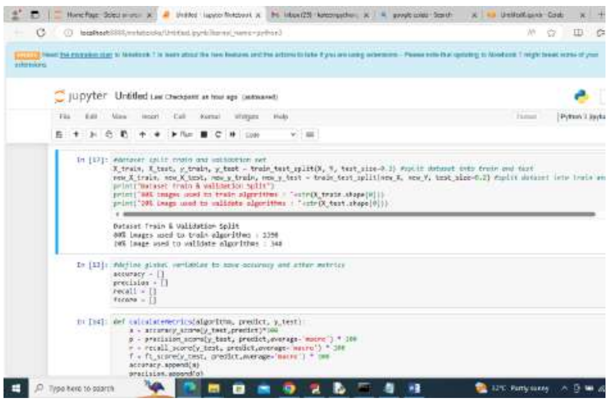
In above screen visualizing graph of various class labels found in dataset where x-axis represents class label and y-axis represents number of images



```
In [12]: #dataset processing like shuffling and normalization on both original and NEW dataset images
         X = X.astype('float32')
         X = X/255 #normalize pixels
         indices = np.arange(X.shape[0])
         np.random.shuffle(indices)#shuffle images
         X = X[indices]
         Y = Y[indices]
         Y = to_categorical(Y)
         new_X = new_X.astype('float32')
         new_X = new_X/255 #normalize pixels
         indices = np.arange(new_X.shape[0])
         np.random.shuffle(indices)#shuffle images
         new_X = new_X[indices]
         new_Y = new_Y[indices]
         new_Y = to_categorical(new_Y)
         print("Dataset Images processing completed")

         Dataset Images processing completed

In [13]: #dataset split train and validation set
         X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2) #split dataset into train and test
```

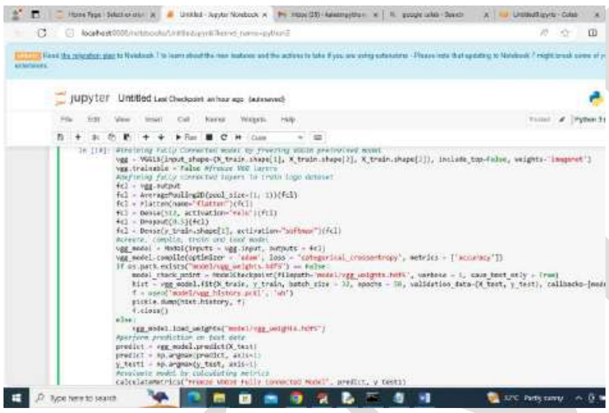
In above screen defining code to shuffle, normalize and process all loaded images for both datasets

ISSN: 2249-7196
IJMRR/April. 2025/ Volume 15/Issue 2s/270-274



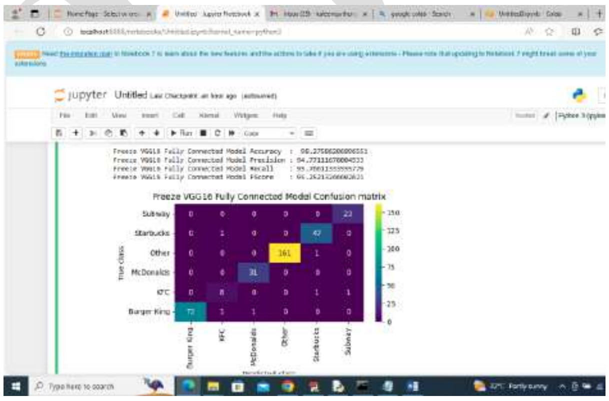
```
In [17]: #dataset split train and validation set
         X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2) #split dataset into train and test
         new_X_train, new_X_test, new_y_train, new_y_test = train_test_split(new_X, new_Y, test_size=0.2) #split dataset into train an
         print("Dataset Train & Validation Split")
         print("80% images used to train algorithms : "+str(X_train.shape[0]))
         print("20% image used to validate algorithms : "+str(X_test.shape[0]))

         Dataset Train & Validation Split
         80% images used to train algorithms : 1358
         20% image used to validate algorithms : 340

In [13]: #define global variables to save accuracy and other metrics
         accuracy = []
         precision = []
         recall = []
         fscore = []

In [14]: def calculateMetrics(algorithm, predict, y_test):
             a = accuracy_score(y_test,predict)*100
             p = precision_score(y_test, predict,average='macro') * 100
             r = recall_score(y_test, predict,average='macro') * 100
             f = f1_score(y_test, predict,average='macro') * 100
             accuracy.append(a)
             precision.append(p)
```

In above screen defining code to split both datasets into train and test with as ratio of 80:20 where 80% images for training and 20% for testing and then defining code to calculate accuracy and other metrics



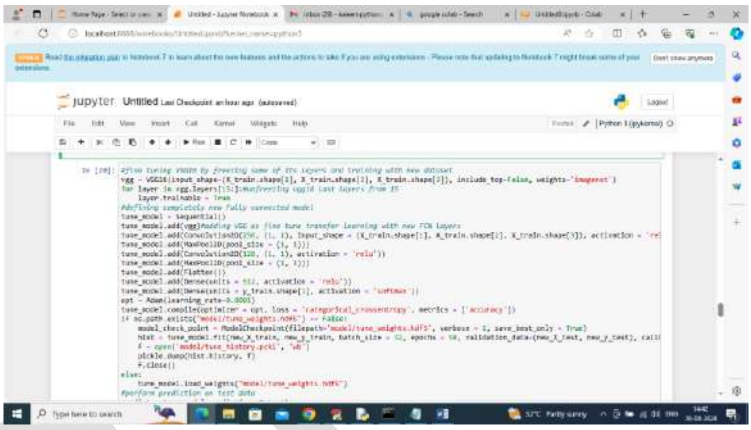
```
In [18]: #defining fully connected model by freezing VGG16 pretrained model
         vgg = VGG16(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')
         vgg.trainable = False #freeze VGG layers
         #defining fully connected layers to train logo dataset
         fc1 = vgg.output
         fc1 = AveragePooling2D(pool_size=(1, 1))(fc1)
         fc1 = Flatten(name="flatten")(fc1)
         fc1 = Dense(512, activation="relu")(fc1)
         fc1 = Dropout(0.3)(fc1)
         fc1 = Dense(y_train.shape[1], activation="softmax")(fc1)
         #create, compile, train and load model
         vgg_model = Model(inputs = vgg.input, outputs = fc1)
         vgg_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
         if os.path.exists("model/vgg_weights.hdf5") == False:
             model_check_point = ModelCheckpoint(filepath='model/vgg_weights.hdf5', verbose = 1, save_best_only = True)
             hist = vgg_model.fit(X_train, y_train, batch_size = 32, epochs = 50, validation_data=(X_test, y_test), callbacks=[mode
             f = open('model/vgg_history.pckl', 'wb')
             pickle.dump(hist.history, f)
             f.close()
         else:
             vgg_model.load_weights("model/vgg_weights.hdf5")
         #perform prediction on test data
         predict = vgg_model.predict(X_test)
         predict = np.argmax(predict, axis=1)
         y_test1 = np.argmax(y_test, axis=1)
         #evaluate model by calculating metrics
         calculateMetrics("Freeze VGG16 Fully Connected Model", predict, y_test1)
```

In above screen training VGG16 on first dataset by freezing all layers and then defining FCN layers to train Logo dataset and then after executing this block will get below output



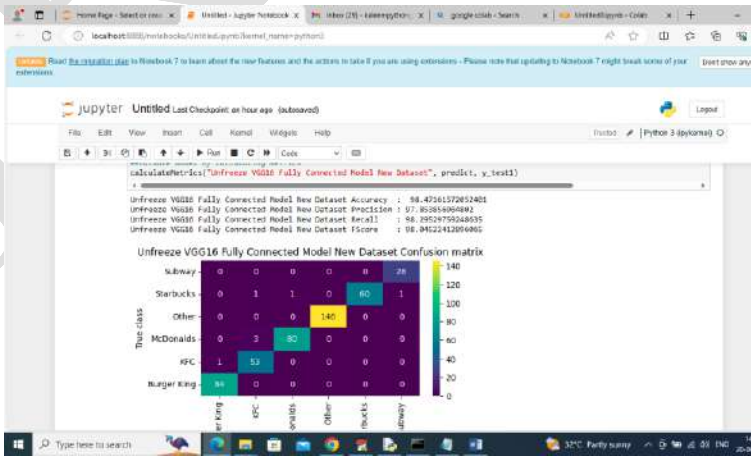
```
Freeze VGG16 Fully Connected Model Accuracy  :  98.27586206896551
Freeze VGG16 Fully Connected Model Precision : 94.77111678004533
Freeze VGG16 Fully Connected Model Recall    : 95.35011555555779
Freeze VGG16 Fully Connected Model FScore    : 95.25213266602631
```

In above screen Freeze VGG16 got 98.27% accuracy and can see other metrics like precision, recall and FSCORE. In confusion matrix graph x-axis represents Predicted Labels and y-axis represents True labels and then all different colour boxes in diagnol represents correct prediction count and remaining blue boxes contains incorrect prediction count which are very few



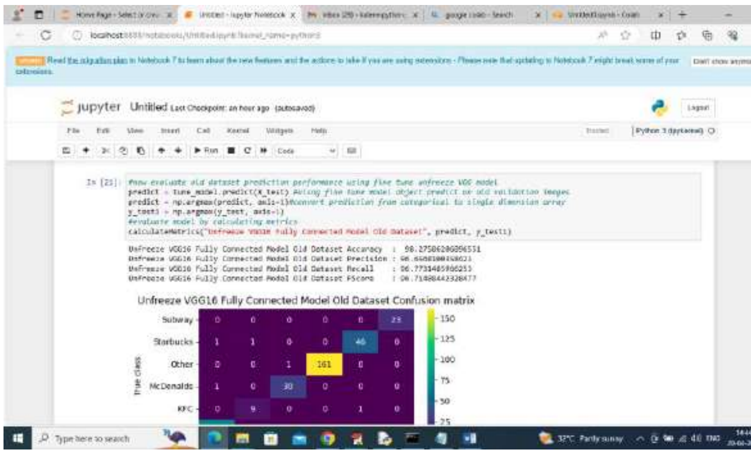
```
In [20]: #fine tuning VGG16 by freezing some of its layers and training with new dataset
         vgg = VGG16(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')
         for layer in vgg.layers[15:]:#unfreezing vgg16 last layers from 15
             layer.trainable = True
         #defining completely new fully connected model
         tune_model = Sequential()
         tune_model.add(vgg)#adding VGG as fine tune transfer learning with new FCN layers
         tune_model.add(Convolution2D(256, (1, 1), input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3]), activation = 're
         tune_model.add(MaxPool2D(pool_size = (1, 1)))
         tune_model.add(Convolution2D(128, (1, 1), activation = 'relu'))
         tune_model.add(MaxPool2D(pool_size = (1, 1)))
         tune_model.add(Flatten())
         tune_model.add(Dense(units = 512, activation = 'relu'))
         tune_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))
         opt = Adam(learning_rate=0.0001)
         tune_model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
         if os.path.exists("model/tune_weights.hdf5") == False:
             model_check_point = ModelCheckpoint(filepath='model/tune_weights.hdf5', verbose = 1, save_best_only = True)
             hist = tune_model.fit(new_X_train, new_y_train, batch_size = 32, epochs = 50, validation_data=(new_X_test, new_y_test), call
             f = open('model/tune_history.pckl', 'wb')
             pickle.dump(hist.history, f)
             f.close()
         else:
             tune_model.load_weights("model/tune_weights.hdf5")
         #perform prediction on test data
```

In above screen training VGG16 by unfreezing last layers and then training and testing on new dataset and after executing this block will get below output



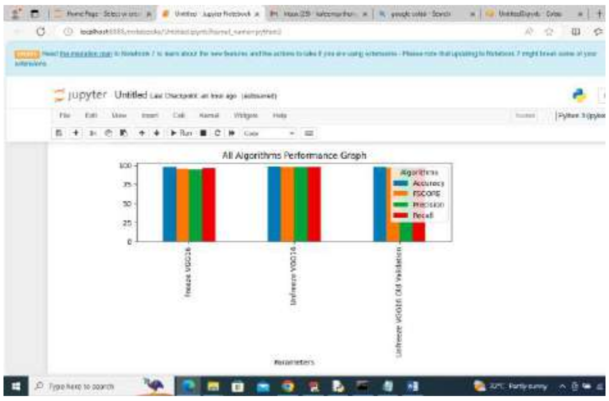
```
calculateMetrics("Unfreeze VGG16 Fully Connected Model New Dataset", predict, y_test1)

Unfreeze VGG16 Fully Connected Model New Dataset Accuracy  :  98.47161572052401
Unfreeze VGG16 Fully Connected Model New Dataset Precision : 97.85856064802
Unfreeze VGG16 Fully Connected Model New Dataset Recall    : 98.29529759248635
Unfreeze VGG16 Fully Connected Model New Dataset FScore    : 98.04522412006065
```
Unfreeze VGG16 Fully Connected Model New Dataset Confusion matrix
True class
Subway
Starbucks
Other
McDonalds
KFC
Burger King

In above screen Unfreeze VGG16 got 98.47% accuracy and can see other metrics also

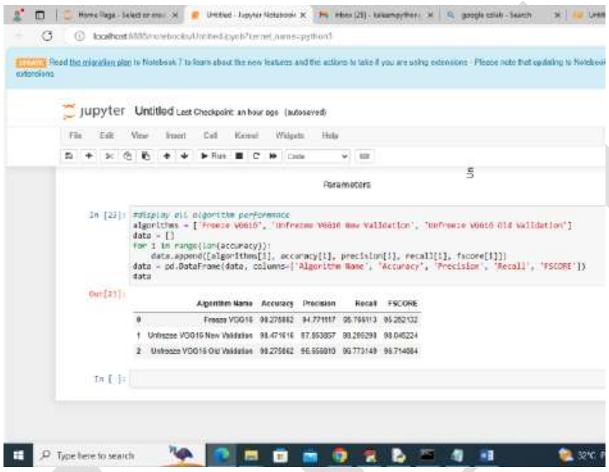


```
In [21]: #now evaluate old dataset prediction performance using fine tune unfreeze VGG model
         predict = tune_model.predict(X_test) #using fine tune model object predict on old validation images
         predict = np.argmax(predict, axis=1)#convert prediction from categorical to single dimension array
         y_test1 = np.argmax(y_test, axis=1)
         #evaluate model by calculating metrics
         calculateMetrics("Unfreeze VGG16 Fully Connected Model Old Dataset", predict, y_test1)

         Unfreeze VGG16 Fully Connected Model Old Dataset Accuracy  :  98.27586206896551
         Unfreeze VGG16 Fully Connected Model Old Dataset Precision : 96.65881004358621
         Unfreeze VGG16 Fully Connected Model Old Dataset Recall    : 96.77314859662551
         Unfreeze VGG16 Fully Connected Model Old Dataset FScore    : 96.71488442328477
```
Unfreeze VGG16 Fully Connected Model Old Dataset Confusion matrix
True class
Subway
Starbucks
Other
McDonalds
KFC

In above screen evaluating unfreeze VGG16 on old dataset and then it got 98.27% accuracy on old unknown dataset so we can say unfreeze pre-trained model works better than freeze models



In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars



```
In [23]: #display all algorithm performance
         algorithms = ['Freeze VGG16', 'Unfreeze VGG16 New Validation', 'Unfreeze VGG16 Old Validation']
         data = []
         for i in range(len(accuracy)):
             data.append([algorithms[i], accuracy[i], precision[i], recall[i], fscore[i]])
         data = pd.DataFrame(data, columns=['Algorithm Name', 'Accuracy', 'Precision', 'Recall', 'FSCORE'])
         data
Out[23]:
                   Algorithm Name   Accuracy  Precision     Recall     FSCORE
   0                 Freeze VGG16  98.276862  94.771117  95.799113  95.282132
   1 Unfreeze VGG16 New Validation  98.471616  97.853857  98.286298  98.046224
   2 Unfreeze VGG16 Old Validation  98.276862  96.656810  96.773149  96.714084
```

In above screen displaying all algorithm performance in tabular format and in all versions unfreeze VGG16 got high accuracy

SUMMARY

To get better performance we have normalize and shuffle dataset value and then measured performance by freezing and unfreezing layers with different number of neurons. While training we have experimented with different optimizers such as SGD and Adam and in both optimizers we have found Adam performing well.

CONCLUSION

Image classification, a fundamental task in computer vision, plays a pivotal role in various applications, including object recognition, medical imaging, and autonomous vehicles. However, training deep neural networks from scratch for image classification often requires vast amounts of labeled data and computational resources. Transfer learning offers a practical solution to this challenge by leveraging pre-trained models and transferring knowledge from one task to another. This paper investigates the application of transfer learning in image classification tasks. By fine-tuning pre-trained convolutional neural networks (CNNs) on new datasets, we demonstrate the effectiveness of transfer learning in achieving high classification accuracy with limited labeled data. We explore different transfer learning strategies, including feature extraction and fine-tuning of entire networks, and evaluate their performance on benchmark datasets.

REFERENCE

[1] Gupta, J., Pathak, S., & Kumar, G. (2022). Bare skin image classification using convolution neural netowrk. International Journal of Emerging Technology and Advanced Engineering, 12(01).

[2] Hameed, Z., Zahia, S., Garcia-Zapirain, B., Javier Aguirre, J., & María Vanegas, A. (2020). Breast cancer histopathology image classification using an ensemble of deep learning models. Sensors, 20(16), 4373.

[3] Indraswari, R., Rokhana, R., & Herulambang, W. (2022). Melanoma image classification based on MobileNetV2 network. Procedia Computer Science, 197, 198-207.

[4] Rajpal, S., Lakhyani, N., Singh, A. K., Kohli, R., & Kumar, N. (2021). Using handpicked features in conjunction with ResNet-50 for improved detection of COVID-19 from chest X-ray images. Chaos, Solitons & Fractals, 145, 110749.