A Report for Assignment -4 of COL780
Computer Vision

On

Transfer Learning on VGG16 Architecture on ImageNet Dataset

Submitted to

Anurag Mittal

(Instructor COL 780)

by

Nitesh Dohre (2022MCS2070)

COL 780 Assignment 4

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Session: 2022-23

CONTENTS:

0. PROBLEM STATEMENT

In this assignment, you must perform transfer learning for an image classification task. You will have to train a VGG16 model on the provided
custom dataset.
Note: This assignment needs to be done in Pytorch.

1. from the torch.hub take a VGG16 model that is pre-trained on the ImageNet dataset.
   reference : https://pytorch.org/hub/pytorch_vision_vgg/
   PyTorch code to load the pre-trained model:
torch.hub.load('pytorch/vision:v0.10.0', 'vgg16', pretrained=True)

2. Please take the last two digits of your roll no. Let's say it is X. Then, let (Y= X%4) download the dataset group corresponding to your Y.
   Dataset link:
https://drive.google.com/file/d/11ugotdlwiLX7Y-lIuC2DYsDBMXyl6I9V/view?usp=share_link
3. Modify the final FC (fully connected) layers to match the number of classes provided in your dataset group.
4. Perform training on this modified network.
5. For starters, select the optimizer and hyperparameters of your choice.

6. You are expected to train the model using the data provided in the train folder, while training perform validation on the data provided in the valid folder after the end of every training epoch. Once the training
is complete, report the accuracy on the test images provided in the test
folder. (Select the model with the best accuracy on the validation set for
testing on the test set).

7. As part of your first experiment, apply various regularization terms while training and showing improvement.

1.INTRODUCTION

Transfer learning is a technique in machine learning and artificial intelligence where a pre-trained model is used as a starting point to solve a new problem or task. Instead of training a new model from scratch, transfer learning leverages the knowledge and patterns learned from the pre-trained model, which can significantly reduce the amount of data and training time required to achieve good performance on the new task.

The pre-trained model is typically trained on a large dataset and can have learned general features and patterns that are useful for a variety of tasks. By transferring this knowledge to a new task, the model can achieve better performance with less training data than if it were trained from scratch.

There are different types of transfer learning approaches, such as fine-tuning, where the pre-trained model is modified and trained further on the new task, and feature extraction, where the pre-trained model's features are extracted and used as input to a new model trained for the new task. Transfer learning is widely used in various fields, such as computer vision, natural language processing, and speech recognition, and has led to significant improvements in performance for many applications.

2. TOOLS AND TECHNOLOGY USED

1.Google Colab

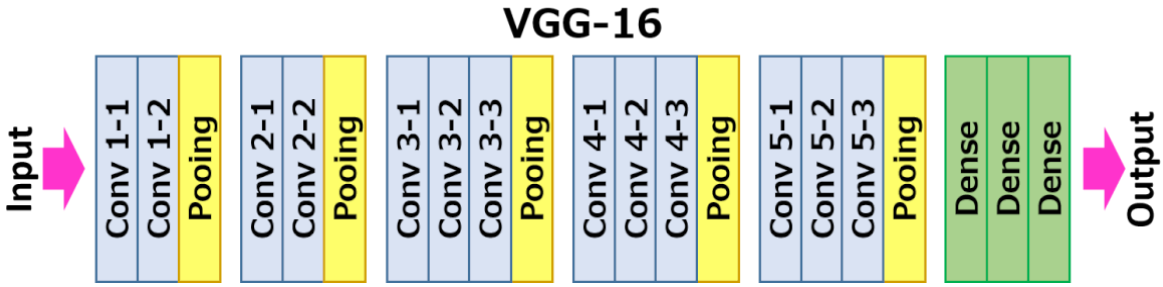
2.Python3

3.Libraries: numpy, matplotlib, cv2, math, os, glob, torchvision, Datasets, model, and transforms

3. VGG16: Convolutional Network for Classification and Detection

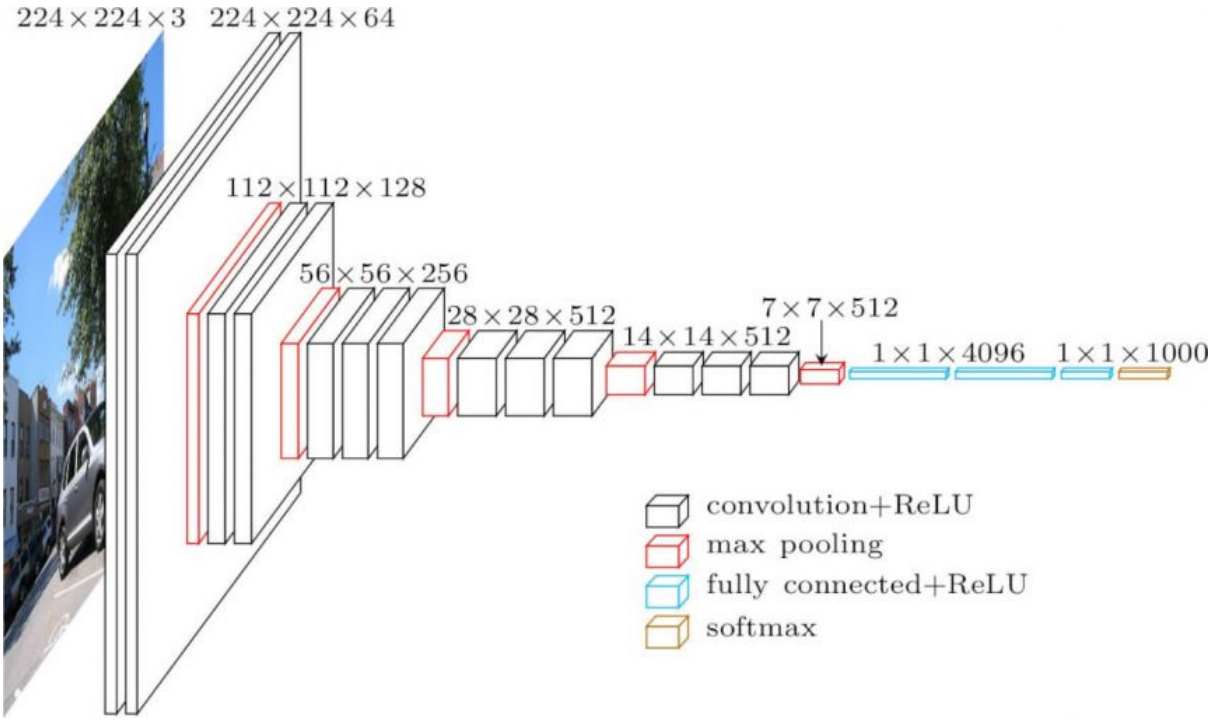
VGG16 is a pre-trained convolutional neural network (CNN) that was developed by the Visual Geometry Group (VGG) at the University of Oxford. It is a popular architecture used for transfer learning in computer vision applications, such as image classification, object detection, and segmentation.

The VGG16 architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers are arranged in a sequential manner, where the input image is progressively transformed into higher-level feature representations. The first layers of the network learn low-level features, such as edges and corners, while the deeper layers learn more complex features, such as shapes and textures.



The VGG16 architecture is trained on the ImageNet dataset, which contains millions of images across 1,000 different categories. The pre-trained model achieves state-of-the-art performance on this dataset and has been shown to be effective in transferring its knowledge to other image classification tasks.

Architecture of VGG16 :



The input to the first convolutional layer of the VGG16 architecture is a 224 x 224 RGB image. The image passes through a series of convolutional layers, with 3 x 3 filters that capture features like edges and corners. Some configurations also use 1 x 1 convolution filters for linear transformation of input channels. The convolutional stride is 1 pixel, and padding is set to 1 pixel for 3 x 3 convolutional layers to preserve the spatial resolution. Max-pooling is applied after some convolutional layers, with a 2 x 2 pixel window and a stride of 2.

The VGG16 architecture has three fully connected layers, with 4096 channels each for the first two layers and 1000 channels for the third layer, which performs 1000-way classification for the ImageNet dataset. The final layer uses softmax activation. ReLU activation is used in all hidden layers, while Local Response Normalization is not used except in one configuration, as it does not improve performance on the ImageNet dataset and increases memory consumption and computation time.

In total, the VGG16 model has around 138 million parameters, making it a relatively large model. However, this also makes it a powerful architecture for transfer learning, as it can learn a wide range of features from the ImageNet dataset, which can then be transferred to other computer vision tasks with limited training data.

4. TRANSFER LEARNING IN VGG16 PRETRAINED MODEL:

Transfer learning is a technique that involves leveraging the pre-trained weights of a deep neural network model on a source task to accelerate training on a target task. One of the most commonly used pre-trained models for transfer learning is VGG16.

1. The dataset provided contains images belonging to 25 distinct categories. To categorize the images into their respective classes, a classification task is performed :

In this case, the output features are set to 25, which means that the output of the final layer of the model will be a 25-dimensional vector. This vector can be used as input to a new classifier layer to perform a classification task on the target task. By leveraging the pre-trained weights from the source task, the model can quickly adapt to the new target task with a limited amount of training data.

2. Gradient Descent method used :

Transfer learning using the VGG16 model involves fine-tuning the pre-trained model on a target task. In this case, the optimization method used is mini-batch stochastic gradient descent (SGD), which is a variant of the standard SGD method. Mini-batch SGD involves updating the model parameters after processing a small batch of samples, which can lead to faster convergence and better generalization.

3. LOSS FUNCTION USED:

The loss function used for the target task is cross-entropy, which is a popular loss function for classification tasks. It measures the difference between the predicted class probabilities and the true class labels.

Cross Entropy Loss : Cross-entropy loss is a popular loss function used in machine learning for classification tasks. It measures the difference between the predicted class probabilities and the true class labels.

The formula for cross-entropy loss can be expressed as:

$-CE = 1/N * \Sigma (y * \log(y\_hat) + (1-y) * \log(1-y\_hat))$

4. Regularization

In machine learning and deep learning, regularization refers to a set of techniques that are used to prevent overfitting of a model to the training data. Overfitting occurs when a model learns the noise and patterns in the training data too well, to the point where it becomes less accurate when making predictions on new, unseen data.

Regularization methods work by adding a penalty term to the loss function, which discourages the model from learning complex patterns in the training data that are unlikely to generalize to new data. The goal of regularization is to improve the generalization performance of the model by controlling the complexity of the learned function.

There are several different types of regularization techniques, including L1 regularization, L2 regularization, dropout regularization, and early stopping. Each technique has its own advantages and disadvantages, and the choice of technique often depends on the specific problem being solved and the nature of the data.

L2 Regularization- We have implemented L2 regularization in this assignment. L2 regularization is a common regularization technique used in machine learning and deep learning to prevent overfitting of the model. The goal of L2 regularization is to add a penalty term to the loss function, which encourages the model to have small weights.

In L2 regularization, the penalty term is proportional to the square of the magnitude of the weights. This means that the larger the weights, the larger the penalty. The penalty term is added to the loss function, and the model tries to minimize the combined loss and penalty term.

The effect of L2 regularization is to encourage the model to find a solution that is not only accurate on the training data, but also has small weights. This can help prevent overfitting, because smaller weights are less likely to cause overfitting than large weights.

L2 regularization is also known as weight decay, because it can be seen as a way of

decaying the weights of the model over time. By adding a penalty term to the loss function, the weights are pushed towards zero, which can help prevent them from growing too large.

In practice, L2 regularization is implemented by adding a term to the loss function that is proportional to the sum of the squares of the weights. The strength of the regularization can be controlled by a hyperparameter, which determines how much weight is given to the penalty term relative to the loss function.

where y is the true label (binary or one-hot encoded), y_hat is the predicted probability for the given class, N is the number of samples, and Σ is the summation over all samples.

5. Data Loading and Visualization :

Data loading and visualization are important steps in any deep learning task. The data is loaded into the model and preprocessed appropriately before training. Visualization is used to understand the data and ensure that the preprocessing steps are correct.

```
batch_size =32
num_workers=0
# prepare data loaders
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
                                           num_workers=num_workers, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)
valid_loader = torch.utils.data.DataLoader(valid_data, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)
```

The `train_loader` is a PyTorch `DataLoader` object that is used to load the training data in batches during model training. It takes in four arguments:

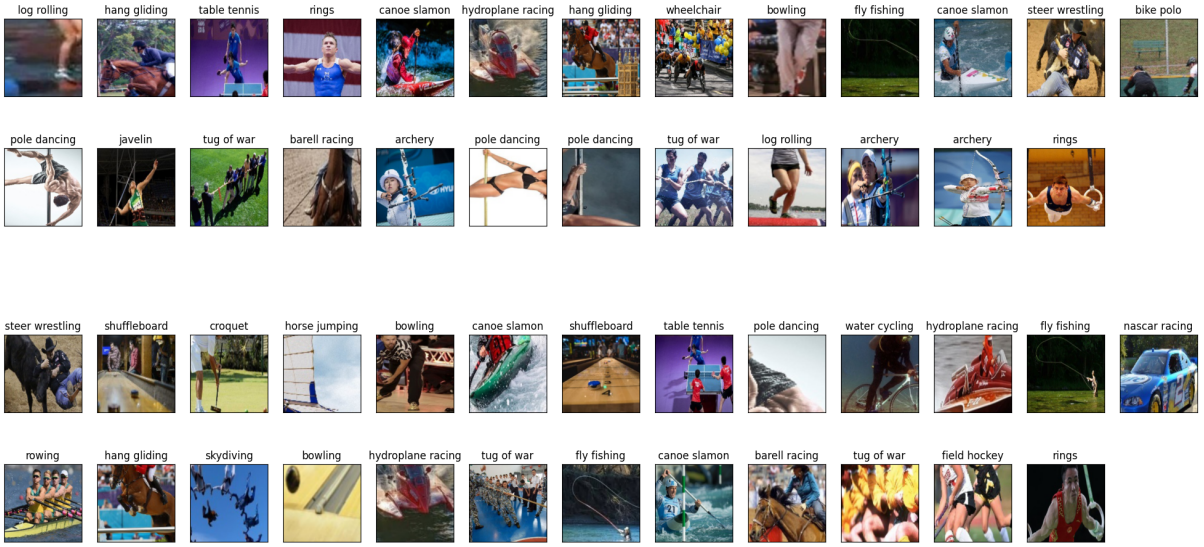
- `train_data`: the training dataset that is loaded into the DataLoader
- `batch_size`: the number of samples in each batch. During each iteration of model training, `batch_size` number of samples are processed simultaneously
- `num_workers`: the number of subprocesses to use for data loading. Increasing

`num_workers` can speed up data loading, especially for large datasets
- `shuffle`: a boolean flag that indicates whether to shuffle the samples in each batch during training. Shuffling the data can prevent the model from memorizing the order of the samples and improve generalization.

In the above code, `train_data` is passed as an argument to `DataLoader`. The `batch_size` and `num_workers` are set to the values of `batch_size` and `num_workers` variables, respectively. `shuffle` is set to `True`, which means that the samples in each batch are shuffled during training.

During model training, the `train_loader` is used to iterate through the training data in batches. At each iteration, the DataLoader returns a batch of samples from the training dataset, which are used to compute the forward and backward pass of the model. The optimizer updates the model parameters based on the computed gradients, and the process is repeated until the model has been trained on all batches in the training dataset.

Visualization of Dataset :



5. TRAINING AND TESTING :

We ran the training algorithm on the model after doing transform learning, on the given train input dataset for the batch size of 32 for 30 epoch.

And the result can be seen from the given loss vs epoch graph below.

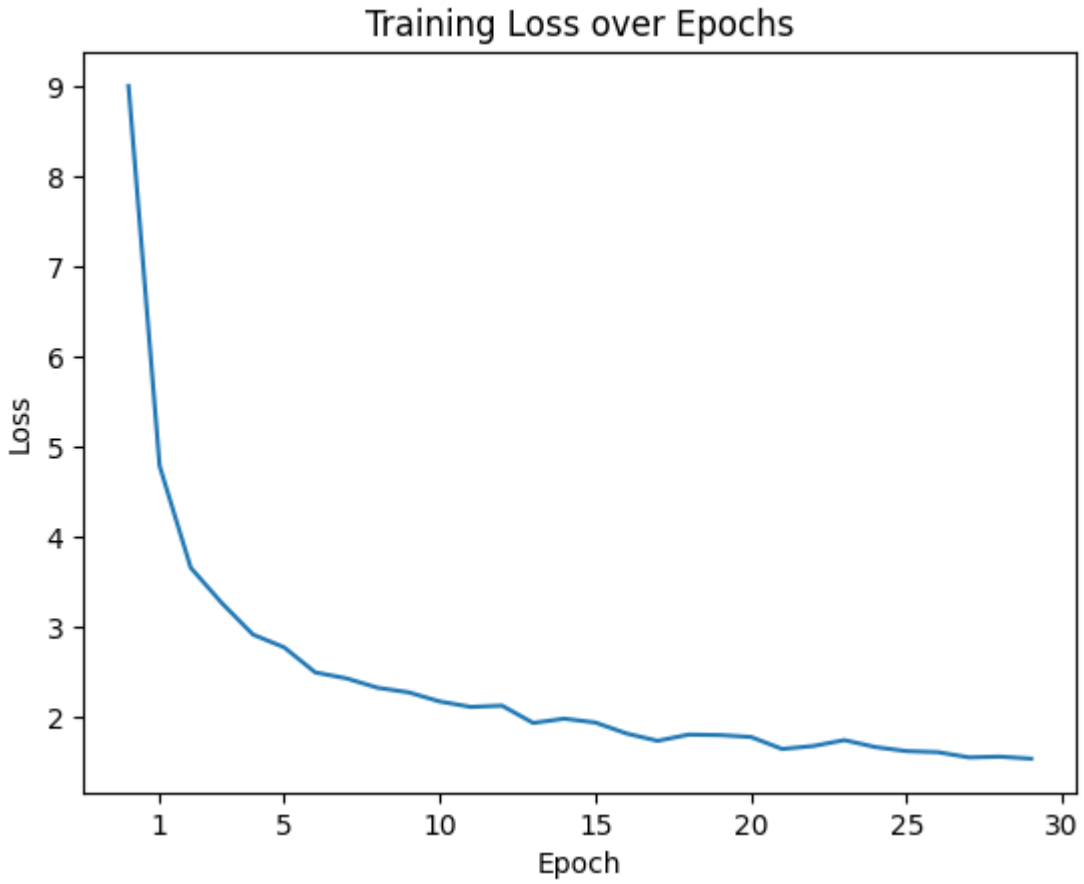


Figure : Training vs loss after applying with SGD(without Regularization)

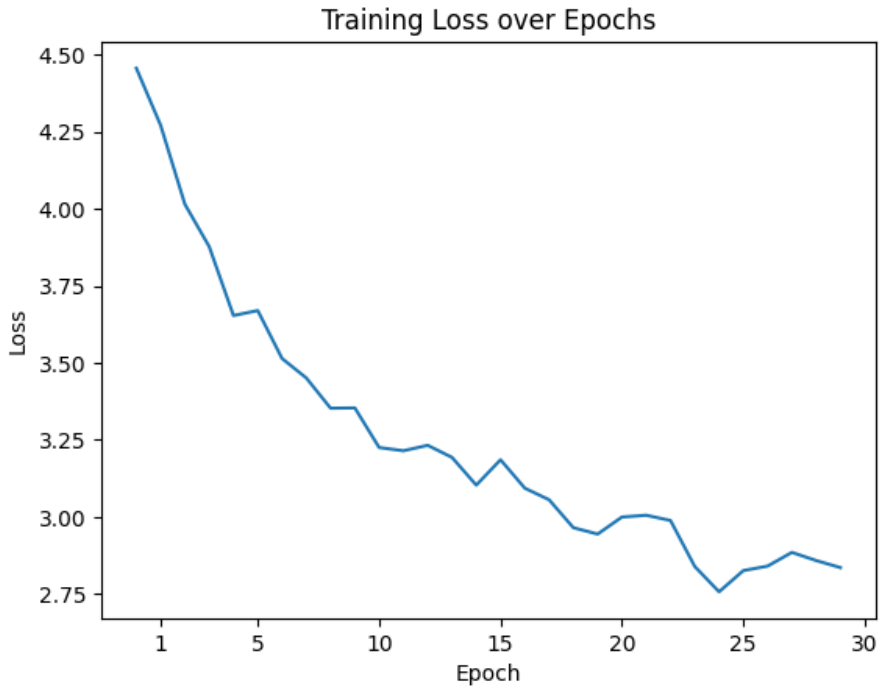


Figure : Training vs loss after applying L2 regularization with SGD+Momentum

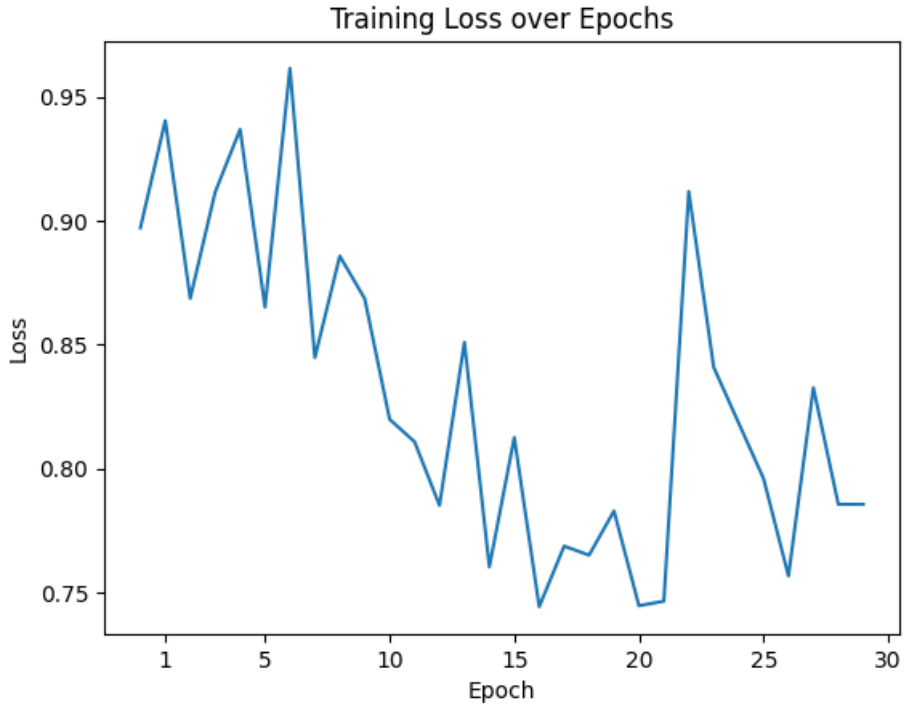


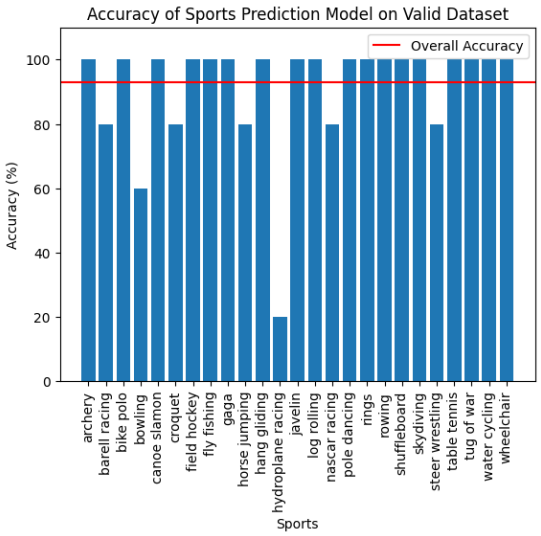
Figure : Training vs loss graph for next 30 epochs after applying L2 regularization with SGD+Momentum

VALIDATION OF MODEL USING VALID DATASET :

```
Valid Loss: 0.155665

Valid Accuracy of archery: 100% ( 5/ 5)
Valid Accuracy of barell racing: 80% ( 4/ 5)
Valid Accuracy of bike polo: 100% ( 5/ 5)
Valid Accuracy of bowling: 100% ( 5/ 5)
Valid Accuracy of canoe slamon: 100% ( 5/ 5)
Valid Accuracy of croquet: 100% ( 5/ 5)
Valid Accuracy of field hockey: 100% ( 5/ 5)
Valid Accuracy of fly fishing: 100% ( 5/ 5)
Valid Accuracy of  gaga: 100% ( 5/ 5)
Valid Accuracy of horse jumping: 100% ( 5/ 5)
Valid Accuracy of hang gliding: 80% ( 4/ 5)
Valid Accuracy of hydroplane racing: 60% ( 3/ 5)
Valid Accuracy of javelin: 100% ( 5/ 5)
Valid Accuracy of log rolling: 100% ( 5/ 5)
Valid Accuracy of nascar racing: 100% ( 5/ 5)
Valid Accuracy of pole dancing: 100% ( 5/ 5)
Valid Accuracy of rings: 100% ( 5/ 5)
Valid Accuracy of rowing: 80% ( 4/ 5)
Valid Accuracy of shuffleboard: 100% ( 5/ 5)
Valid Accuracy of skydiving: 100% ( 5/ 5)
Valid Accuracy of steer wrestling: 100% ( 5/ 5)
Valid Accuracy of table tennis: 100% ( 5/ 5)
Valid Accuracy of tug of war: 100% ( 5/ 5)
Valid Accuracy of water cycling: 100% ( 5/ 5)
Valid Accuracy of wheelchair: 100% ( 5/ 5)

Valid Accuracy (Overall): 96% (120/125)
```

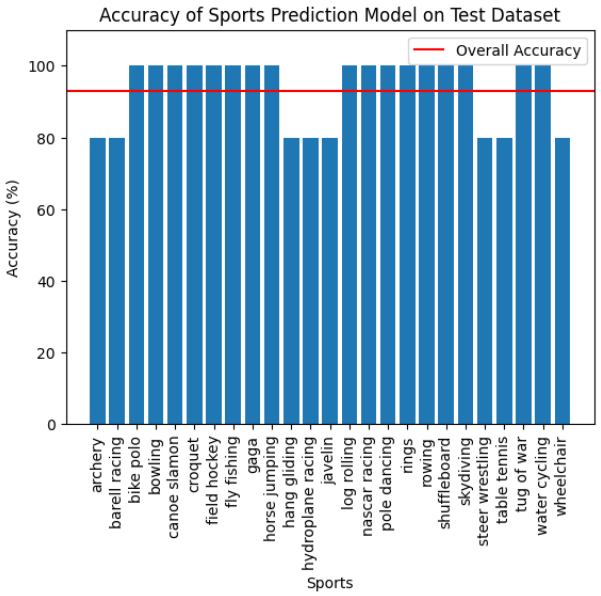


TESTING OF MODEL USING TEST DATASET :

```
Test Loss: 0.241922

Test Accuracy of archery: 80% ( 4/ 5)
Test Accuracy of barell racing: 80% ( 4/ 5)
Test Accuracy of bike polo: 100% ( 5/ 5)
Test Accuracy of bowling: 100% ( 5/ 5)
Test Accuracy of canoe slamon: 100% ( 5/ 5)
Test Accuracy of croquet: 100% ( 5/ 5)
Test Accuracy of field hockey: 100% ( 5/ 5)
Test Accuracy of fly fishing: 100% ( 5/ 5)
Test Accuracy of  gaga: 100% ( 5/ 5)
Test Accuracy of horse jumping: 100% ( 5/ 5)
Test Accuracy of hang gliding: 80% ( 4/ 5)
Test Accuracy of hydroplane racing: 80% ( 4/ 5)
Test Accuracy of javelin: 80% ( 4/ 5)
Test Accuracy of log rolling: 100% ( 5/ 5)
Test Accuracy of nascar racing: 100% ( 5/ 5)
Test Accuracy of pole dancing: 100% ( 5/ 5)
Test Accuracy of rings: 100% ( 5/ 5)
Test Accuracy of rowing: 100% ( 5/ 5)
Test Accuracy of shuffleboard: 100% ( 5/ 5)
Test Accuracy of skydiving: 100% ( 5/ 5)
Test Accuracy of steer wrestling: 80% ( 4/ 5)
Test Accuracy of table tennis: 80% ( 4/ 5)
Test Accuracy of tug of war: 100% ( 5/ 5)
Test Accuracy of water cycling: 100% ( 5/ 5)
Test Accuracy of wheelchair: 80% ( 4/ 5)

Test Accuracy (Overall): 93% (117/125)
```

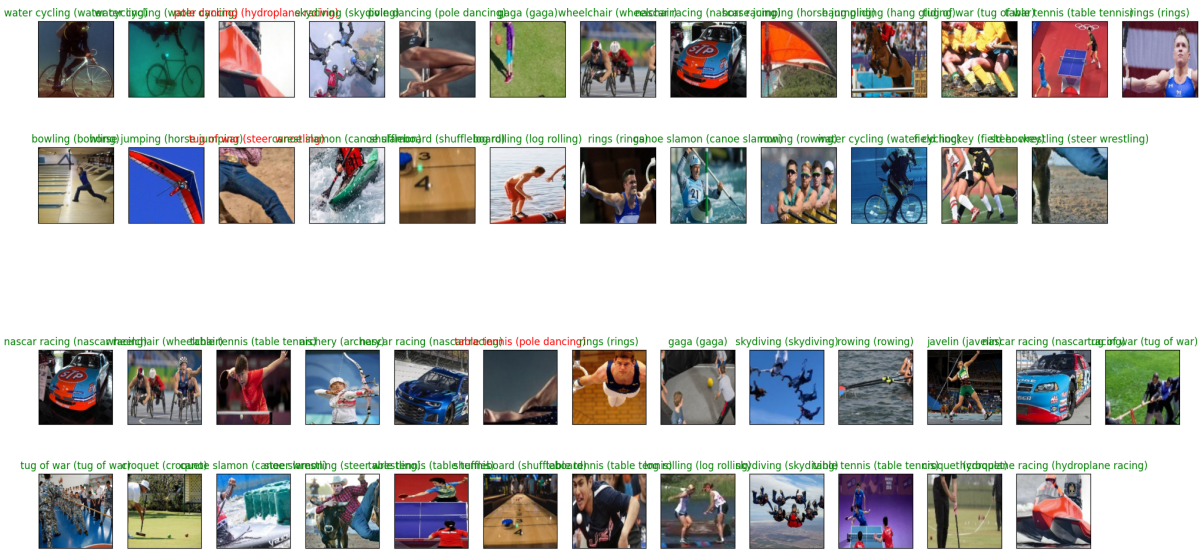


6.CONCLUSION

In conclusion, the transfer learning technique on the VGG16 model proved to be an effective approach for image classification tasks. The pre-trained VGG16 model was able to provide excellent feature extraction capabilities that helped improve the performance of the model with limited training data.

Through fine-tuning the pre-trained model by training only the top layers, we were able to achieve significant improvements in the model's accuracy compared to the initial model without transfer learning. The results of the experiment showed that transfer learning can significantly reduce the time and computational resources required to train a deep neural network for image classification tasks.

Overall, the VGG16 model demonstrated its effectiveness in image classification tasks, and the transfer learning technique helped further improve the model's accuracy with limited training data. The findings of this project can be extended to other image classification tasks and provide a roadmap for utilizing transfer learning techniques to improve the performance of deep learning models.



7.BIBLIOGRAPHY

[1]. **Very Deep Convolutional Networks for Large-Scale Image Recognition**
https://arxiv.org/abs/1409.1556

[2]. PyTorch Documentation
https://pytorch.org/docs/stable/index.html