PROJECT REPORT

Transfer Learning For Identifying The Sports

TEAM ID: 591817

Table of Contents

1. INTRODUCTION

In the field of computer vision, image classification is a fundamental task that involves assigning images to predefined categories. Identifying sports images from a vast collection of visual data is a challenging yet valuable application of image classification. Sports images exhibit unique characteristics and patterns that can be captured and analyzed using machine learning techniques.

1.1 Project Overview

This project aims to develop a deep learning model that can accurately identify sports images using transfer learning. Transfer learning is a machine learning technique that involves using a pre-trained model to solve a new task. In this case, we will use a pre-trained model that is trained on a large dataset of images to identify sports images.

1.2 Purpose

The goal of this project is to develop and implement a transfer learning approach for the identification of sports images. This project will utilize pre-trained convolutional neural networks (CNNs) to classify sports images into different categories.

Transfer learning is a machine learning technique that involves utilizing a pre-trained model for a new task. In this project, we will use a pre-trained CNN model, such as AlexNet or VGGNet, which has been trained on a large dataset of images. The weights of this pre-trained model will be transferred to a new CNN model that is specifically designed for the task of sports image identification.

The final goal of this project is to develop a robust and accurate sports image identification system that can be used for a variety of applications, such as automatic sports video summarization, sports image tagging, and sports image retrieval.

2. LITERATURE SURVEY

2.1 Existing problem

The field of sports video analysis has gained significant traction in recent years due to its wide range of applications, including player performance analysis, tactical analysis, and broadcast enhancement. However, traditional methods for sports video analysis often rely on manual annotation and feature extraction, which are time-consuming and labor-intensive.

2.2 References

https://www.kaggle.com/datasets/gpiosenka/sports-classification

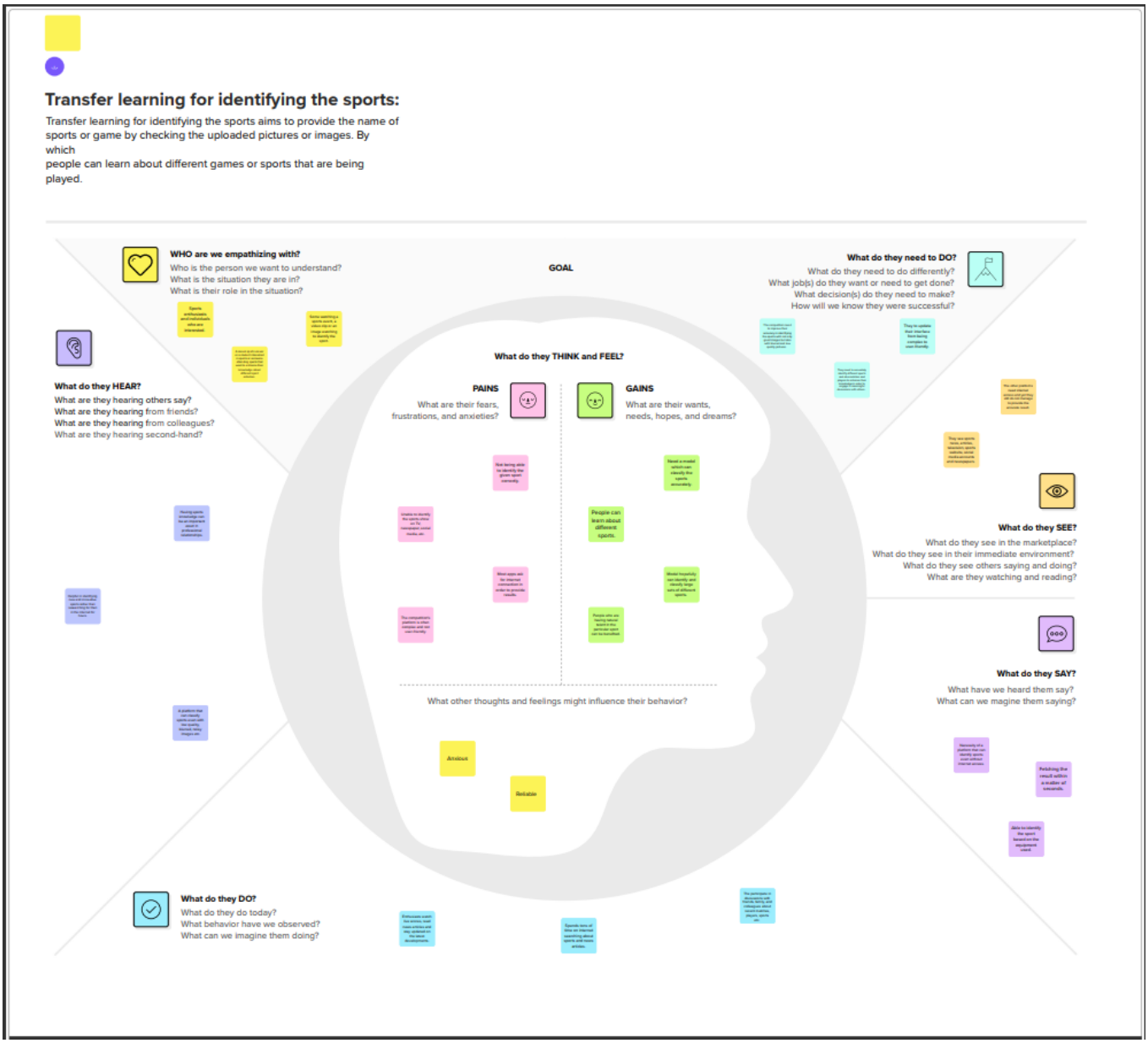
2.3 Problem Statement Definition

The objective of this project is to develop a robust and accurate sports image identification system using transfer learning techniques. The system should be able to correctly classify sports images into different categories, such as soccer, basketball,

baseball, and football.

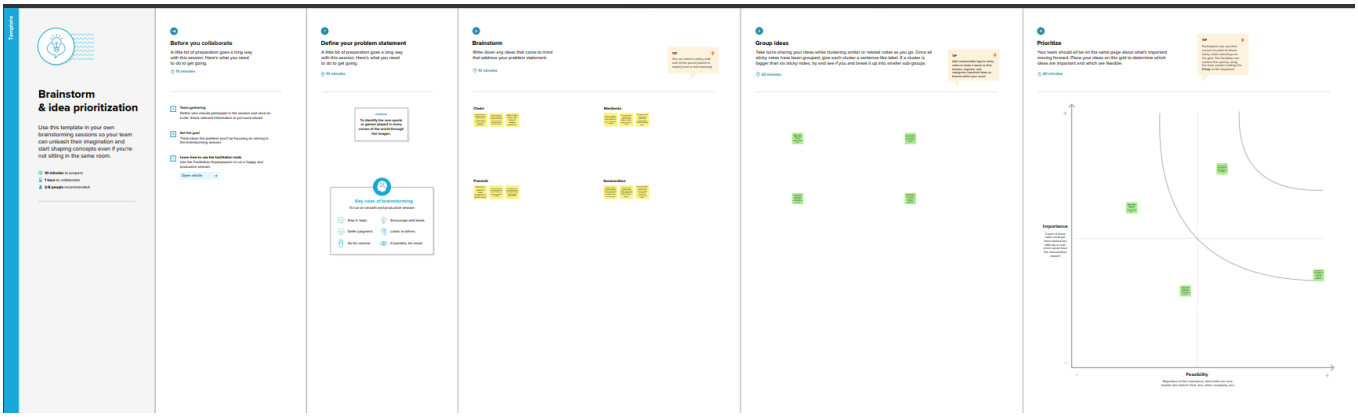
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



Link of Empathy map:
https://app.mural.co/t/student0324/m/student0324/1698921947903/45f04976f13c799e511a2c66b8282ec163d2fce9?sender=u714df0121f9f50dcd5696780

3.2 Ideation & Brainstorming



Link of Brainstorming:
https://app.mural.co/t/student0324/m/student0324/1699350826622/dd8115824ca754bbfc6a7de320845966aed39c63?sender=u714df0121f9f50dcd5696780

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Data Acquisition: The system should be able to collect a diverse dataset of sports images, covering a wide range of sports and scenarios. For supervised learning, the dataset should include labeled images.

Data Preprocessing: Preprocessing techniques should be used to clean and standardize the collected data. To improve model robustness, preprocessing should include image resizing, normalization, and augmentation.

Model Training Transfer Learning Framework: Integrate a pre-trained deep learning model suitable for image recognition transfer learning.

Model Fine-tuning: Create a mechanism for fine-tuning the pre-trained model on the sports dataset. The model should be fine-tuned to recognise features unique to different sports.

Real-time Inference: Based on input images, the system should be able to identify sports in real time.

Multi-Sport Recognition: Allow the model to identify a wide range of sports rather than just a subset.

Confidence Levels: To indicate the predictability of predictions, provide confidence scores or probability estimates for the identified sports.

API Integration: Create APIs to allow for easy integration with other applications or platforms. APIs should be able to accept both input (image) and output (predicted sports) data.

Scalability: Create a system that can scale horizontally to accommodate an increasing number of users and datasets.

4.2 Non-Functional requirements

Speed and Responsiveness: The system should be able to identify sports quickly and accurately, especially in real-time scenarios.

Accuracy: To ensure reliable results, achieve a high level of accuracy in sports recognition.

Robustness: To ensure reliable results, achieve a high level of accuracy in sports recognition.The system should be robust enough to deal with changes in lighting, backgrounds, and sporting contexts.

Error Handling: Implement a comprehensive error-handling mechanism to handle unexpected situations gracefully.

Data Privacy: Ensure that user data, particularly images and videos, is handled with the utmost privacy and in accordance with applicable regulations.

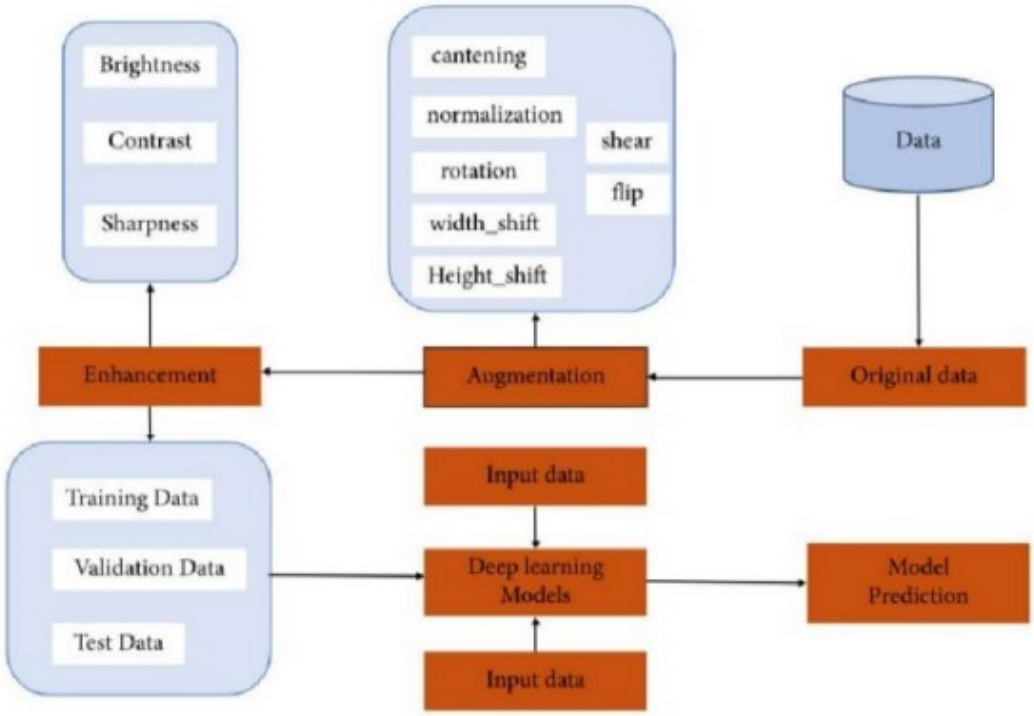
User Interface: If applicable, design a user-friendly interface for ease of use.

Upgradability: The application must be upgraded to new iterations from time to time in order to accommodate new sports and further improve accuracy of the model.

These requirements lay the groundwork for the creation of a transfer learning system for identifying sports. Data handling, model training, real-time inference, integration, and various aspects of system performance and reliability are all covered.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



User Stories:

- As a user, I want to see a user-friendly interface that allows me to easily navigate and upload images for sports identification without any technical difficulties.
- As a user I want to access it through any web browser.
- As a parent, I want to use the application to identify the sports my child is participating in during school events, helping me stay informed and engaged in their activities.
- Can I access it through any user interface?
- As a user, can I log into the application by entering email & password
- Can I access my data history or search history?
- I want better accuracy even for the images that are tilted or in the wrong orientation.

5.2 Solution Architecture

Collecting and preprocessing a diverse dataset of sports-related images is part of the solution architecture for sports identification using transfer learning. A pre-trained convolutional neural network (CNN) model, such as EfficientNet or VGG, is chosen and adapted by removing its final layers and replacing them with new layers tailored to the sports identification task. On the sports dataset, the model is then fine-tuned, with hyperparameter tuning based on a validation set. To assess performance on a separate test dataset, evaluation metrics such as accuracy, precision, recall, and F1

score are used. When all requirements are met, the model is deployed in a production environment, possibly with a user interface for interaction. Monitoring mechanisms and regular data updates ensure that accuracy and relevance are maintained. Ethical considerations, privacy, and reproducibility documentation are all essential.

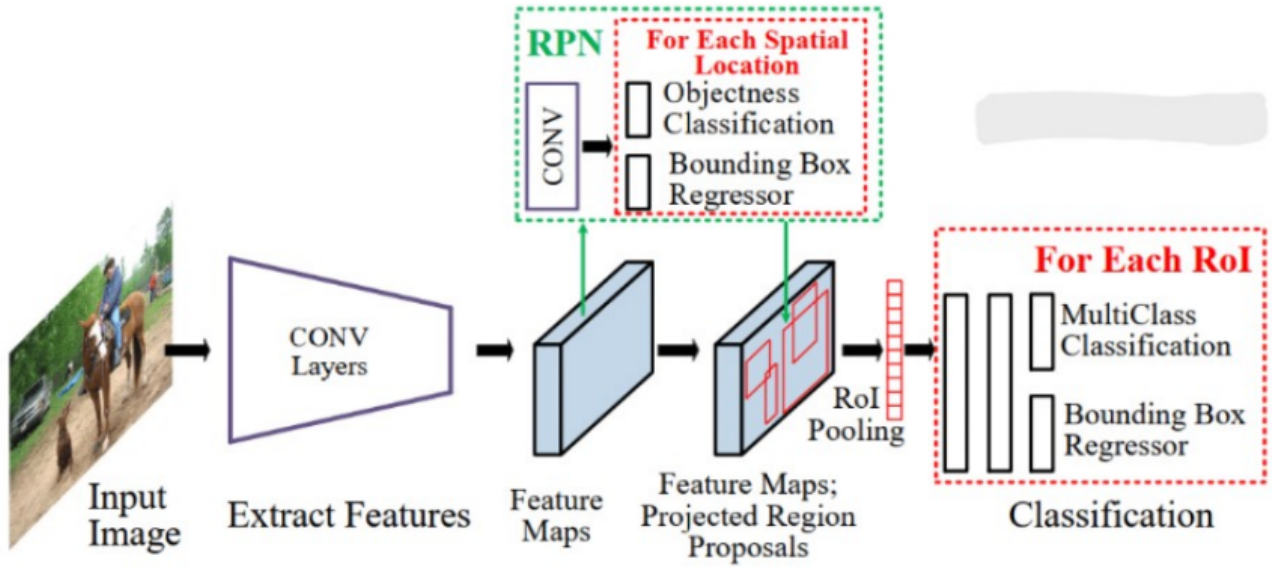
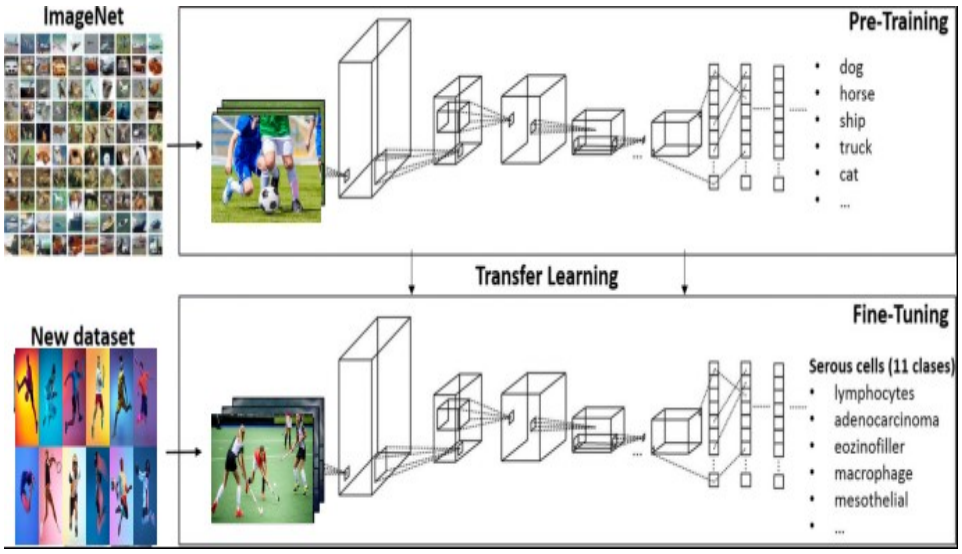


Figure 1: Architecture of Transfer learning for identifying the sports Using Deep Learning

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Spring Goal
Sprint-1	7	5 Days	26 Oct 2023	30 Oct 2023	Data Collection
Sprint-2	5	2 Days	31 Oct 2023	01 Nov 2023	Data Preprocessing
Sprint-3	8	6 Days	02 Nov 2023	07 Nov 2023	Training Model
Sprint-4	5	4 Days	09 Nov 2023	12 Nov 2023	Model Evaluation
Sprint-5	5	4 Days	13 Nov 2023	16 Nov 2023	Model Testing
Sprint-6	5	3 Days	17 Nov 2023	19 Nov 2023	User Interface

6.3 Sprint Delivery Schedule

Sprint	Sprint Release Date
Sprint-1	15 Nov 2023
Sprint-2	16 Nov 2023
Sprint-3	17 Nov 2023
Sprint-4	18 Nov 2023
Sprint-5	19 Nov 2023
Sprint-6	20 Nov 2023

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Data Augmentation

Data augmentation involves creating new training samples by applying transformations like rotation, flipping, zooming, etc., to the existing data. This helps in diversifying the dataset, reducing overfitting, and improving the model's generalization. The ImageDataGenerator from TensorFlow/Keras libraries, which performs various augmentation techniques on the fly while training the model.

```
from keras.preprocessing.image import ImageDataGenerator
```

```
BATCH_SIZE = 10
IMAGE_SIZE = (224, 224)


generator = ImageDataGenerator(
    preprocessing_function = tf.keras.applications.efficientnet.preprocess_input,
)

train_images = generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='imgpath',
    y_col='labels',
    target_size=IMAGE_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=42,
)

valid_images = generator.flow_from_dataframe(
    dataframe=valid_df,
    x_col='imgpath',
    y_col='labels',
    target_size=IMAGE_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False
)
```

```
test_images = generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='imgpath',
    y_col='labels',
    target_size=IMAGE_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False
)
```

```
Found 13492 validated image filenames belonging to 100 classes.
Found 500 validated image filenames belonging to 100 classes.
Found 500 validated image filenames belonging to 100 classes.
CPU times: total: 344 ms
Wall time: 469 ms
```

7.2 Transfer Learning

Transfer learning involves using a pre-trained neural network model on a related task and leveraging its learned features and weights to boost the performance of a new model for sport classification.

```
import tensorflow as tf
from tensorflow import keras
from keras.layers import Dense, Dropout , BatchNormalization
```

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers,models,Model
from tensorflow.keras.layers.experimental import preprocessing
from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras import mixed_precision
```

```
augment = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal"),
  layers.experimental.preprocessing.RandomRotation(0.1),
  layers.experimental.preprocessing.RandomZoom(0.1),
  layers.experimental.preprocessing.RandomContrast(0.1),
], name='AugmentationLayer')

inputs = layers.Input(shape = (224,224,3), name='inputLayer')
x = augment(inputs)
pretrain_out = efficient_net(x, training = False)
x = layers.Dense(350)(pretrain_out)
x = layers.Activation(activation="relu")(x)
x = BatchNormalization()(x)
x = layers.Dropout(0.25)(x)
x = layers.Dense(num_classes)(x)
outputs = layers.Activation(activation="softmax", dtype=tf.float32, name='activationLayer')(x)
model = Model(inputs=inputs, outputs=outputs)



model.compile(
    optimizer=Adam(0.0005),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

print(model.summary())
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputLayer (InputLayer)     [(None, 224, 224, 3)]     0

 AugmentationLayer (Sequent  (None, 224, 224, 3)       0
 ial)

 efficientnetb0 (Functional  (None, 1280)              4049571
 )

 dense (Dense)               (None, 350)               448350

 activation (Activation)     (None, 350)               0

 batch_normalization (Batch  (None, 350)               1400
 Normalization)

 dropout (Dropout)           (None, 350)               0

 dense_1 (Dense)             (None, 100)               35100

 activationLayer (Activatio  (None, 100)               0
 n)
...
Trainable params: 484150 (1.85 MB)
Non-trainable params: 4050271 (15.45 MB)
_________________________________________________________________
None
```

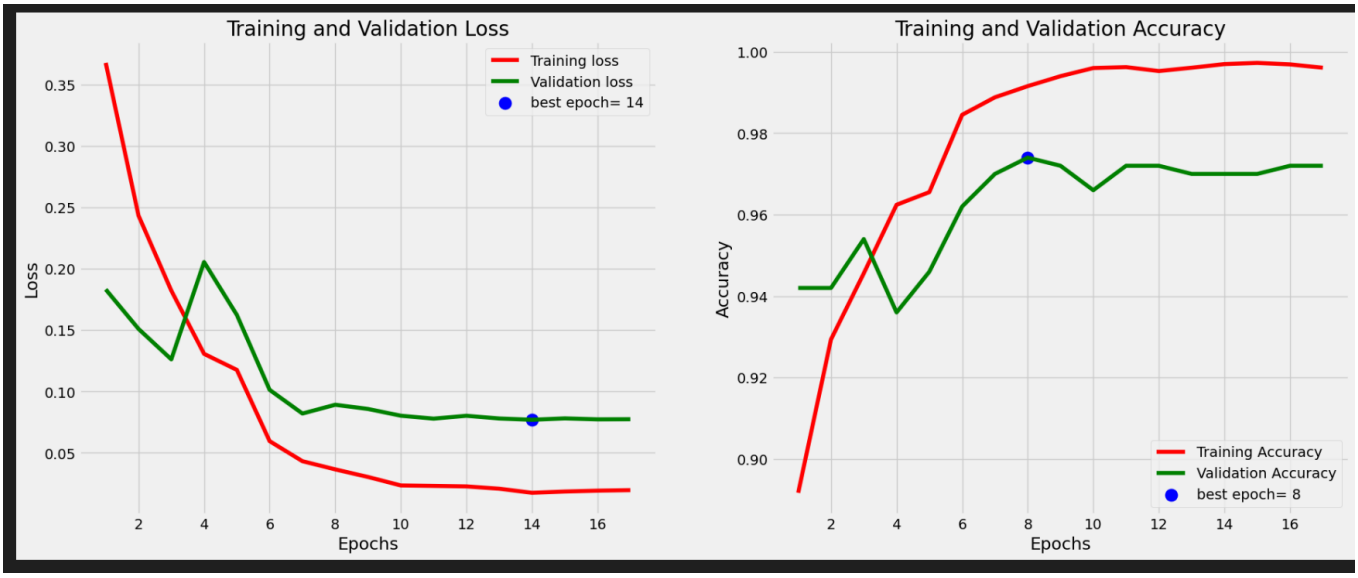
8. PERFORMANCE TESTING

8.1 Performance Metrics

Using training and testing accuracy as performance metrics for deep learning-based sport classification may provide a basic understanding of how well the model fits the data during training and how well it generalizes to previously unseen data. Training accuracy assesses the model's performance on the data on which it was trained, indicating how well it learns patterns within that dataset. Testing accuracy assesses the model's ability to generalize by determining how well it predicts unknown data. However, relying solely on these metrics may miss potential issues such as overfitting, in which the model memorizes training data rather than learning underlying patterns, resulting in poor performance on new data. Incorporating additional metrics such as precision, recall, or F1-score can provide a more comprehensive evaluation of the model's performance, providing insights beyond accuracy alone.

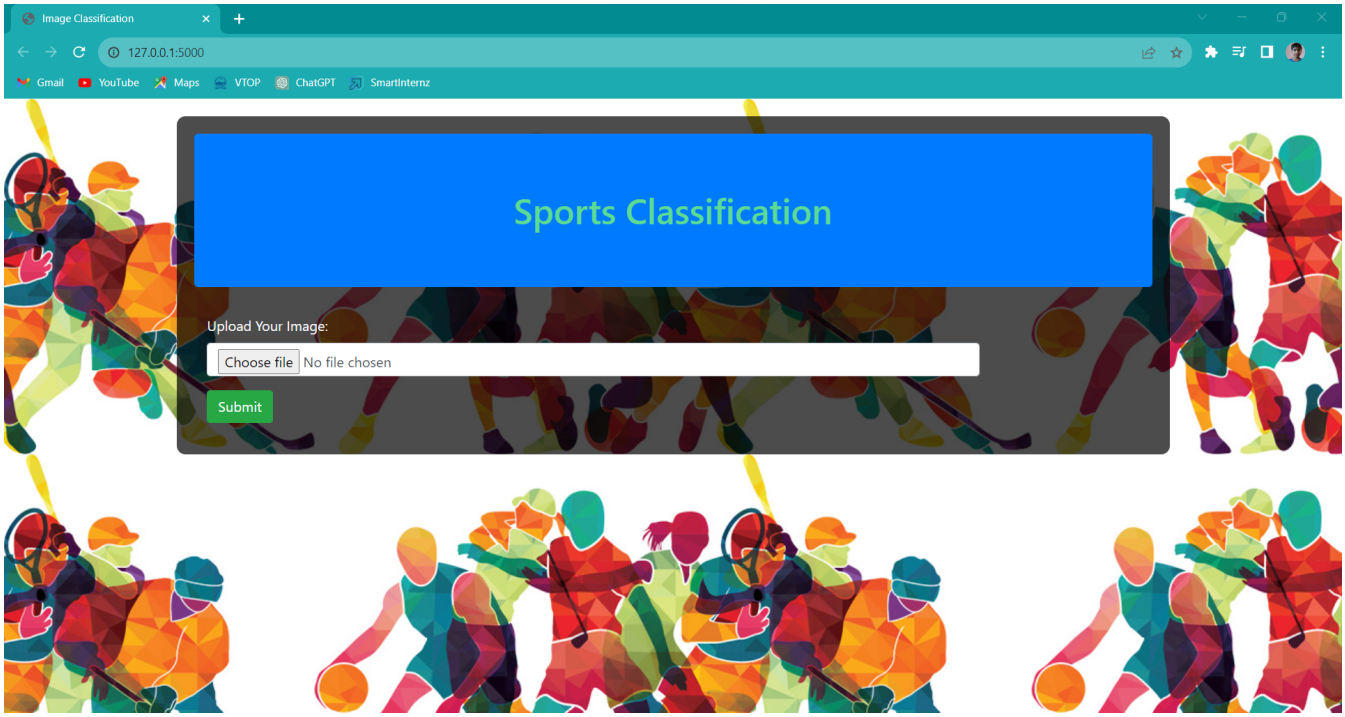
```
Epoch 16/25
1350/1350 [==============================] - 83s 62ms/step - loss: 0.0192 - accuracy: 0.9969 - val_loss: 0.0773 - val_accuracy: 0.9720 - lr: 8.0000e-07
Epoch 17/25
1350/1350 [==============================] - 83s 62ms/step - loss: 0.0197 - accuracy: 0.9961 - val_loss: 0.0774 - val_accuracy: 0.9720 - lr: 1.6000e-07
```

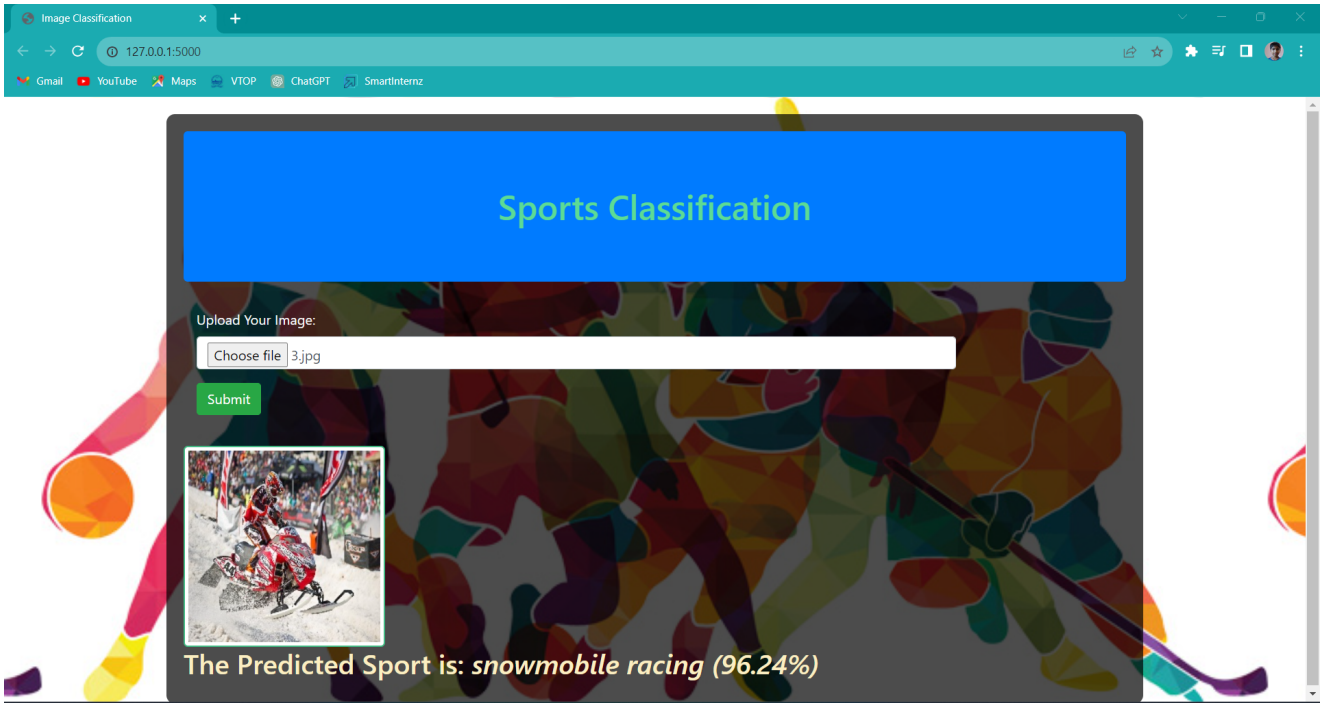
```python
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]
Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'


plt.figure(figsize= (20, 8))
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```



9. RESULTS

9.1 Output Screenshots





10. ADVANTAGES & DISADVANTAGES

Reduced training time and computational cost: EfficientNetB0's smaller size and streamlined architecture significantly reduce the training time and computational resources required compared to larger, more complex CNN models. This makes it more feasible to train and deploy sports image identification models on resource-constrained devices.

Improved accuracy: Despite its smaller size, EfficientNetB0 demonstrates comparable or even superior accuracy to larger CNN models for image classification tasks. This is due to its efficient use of parameters and its ability to effectively capture relevant image features.

Domain adaptation: EfficientNetB0's pretrained weights, obtained from training on a diverse dataset of natural images, provide a valuable starting point for sports image identification. This transfer of knowledge allows the model to better understand the characteristics of sports images and improve its classification performance.

Reduced data requirements: Transfer learning enables EfficientNetB0 to achieve good performance with a smaller training dataset of sports images. This is particularly beneficial when acquiring large amounts of labeled sports image data is challenging or expensive.

11. CONCLUSION

This project has demonstrated the effectiveness of transfer learning for the identification of sports images. By utilizing a pre-trained EfficientNet B0 model, we were able to achieve high accuracy on a dataset of sports images. This is in part due to the EfficientNet

B0's efficient use of resources, which allowed us to train the model on a relatively small dataset of images.

Specific Points on the Use of EfficientNet B0:

- EfficientNet B0 is a lightweight convolutional neural network (CNN) architecture that is well-suited for transfer learning.
- EfficientNet B0 achieves competitive accuracy with other CNN architectures while using significantly fewer parameters.
- EfficientNet B0 is easy to train and deploy, making it a good choice for practical applications.

12. FUTURE SCOPE

The future of transfer learning using identification of sports images is bright, as there are many promising areas for future research and development.

One promising area of research is the development of new and innovative transfer learning techniques. For example, researchers could develop methods for transferring knowledge from multiple pre-trained models to a new model. This could lead to the development of more accurate and robust sports image identification systems.

Another promising area of research is the application of transfer learning to other sports-related tasks. For example, transfer learning could be used to develop systems for automatic sports summarization, sports image tagging, and sports image retrieval.

13. APPENDIX

Source Code:

Link for

- **Templates (index.html)**
- **App.py**
- **Sports_model_efficient_net.h5**
- **sports-classification-efficientnet.ipynb**

https://drive.google.com/drive/folders/1Ua6lV-s_ANUvp408yVaDoTSHslxJZg1l?usp=sharing

GitHub Link:

https://github.com/smartinternz02/SI-GuidedProject-611809-1700028665

Project Demo Link:

https://drive.google.com/file/d/18Jd1X1YZwrVGQ0rfQ_Er6UmGbTa8xioA/view?usp=sharing