Volume 9, Issue 8, August – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24AUG974

Butterfly Image Classification Using Convulational Neural Network[CNN]

Amruth N Murthy[1]; Kavana N Murthy[2]; Dr. Shivandappa[3]; Dr. Narendra Kumar S[4]
Department of Biotechnology, R V College of Engineering, Bengaluru, India

Abstract:- Butterfly species identification through image classification is now a major use of computer vision, utilizing supervised learning methods to classify different types of butterflies based on images. This article gives a detailed overview of the latest progress in classifying butterfly images through supervised learning techniques on Google Colab, a widely-used cloud-based platform for machine learning projects. The review starts by discussing the significance of precise butterfly categorization for biodiversity research and conservation endeavors. It then goes into specifics about different methods used in supervised learning for this purpose, such as convolutional neural networks (CNNs), support vector machines (SVMs), and k-nearest neighbors (k-NN).

The review discusses the pros and cons of using these methods on butterfly image data, emphasizing on accuracy, efficiency, and generalization. Special focus is placed on the preprocessing procedures necessary to improve image quality and extract features, including image augmentation, normalization, and feature scaling. The article also investigates various butterfly image datasets that are accessible to the public, analyzing how they are used for training and assessing classification models.

Google Colab is highlighted as a potent instrument for creating and testing these models because of its convenience, user-friendliness, and compatibility with leading machine learning libraries such as TensorFlow and PyTorch. Furthermore, the article examines recent research and initiatives that have effectively utilized butterfly image categorization with Colab, demonstrating ideal methods and insight gained.

Image Preprocessing, Feature Extraction, Machine Learning Libraries, TensorFlow, PyTorch, Image Augmentation, Publicly Available Datasets.

Keywords:- *Butterfly Image Classification, Supervised Learning, Google Colab, Convolutional Neural Networks (Cnns), Support Vector Machines (Svms), K-Nearest Neighbors (K-NN),*

I. INTRODUCTION

Butterfly image classification has become an essential task in the field of computer vision, due to its importance in biodiversity monitoring and conservation. Accurately classifying butterfly species from images can significantly help track population dynamics, understand ecological interactions, and protect endangered species. Recent advances in machine learning, particularly supervised learning, have enabled the development of sophisticated models capable of distinguishing between different butterfly species with high accuracy.

Supervised learning, a branch of machine learning where models are trained on labelled data, has been shown to be particularly effective for image classification tasks. Techniques such as convolutional neural networks (CNN), support vector machines (SVM) and k-nearest neighbours (k-NN) have been used to address the complexity of butterfly image classification. CNNs, with their ability to automatically learn hierarchical features of images, have become the method of choice for many researchers due to their exceptional performance in visual recognition tasks.

This paper explores the use of these supervised learning techniques within Google Colab, a widely used cloud-based platform that facilitates easy experimentation with machine learning models. Google Colab provides an accessible environment for developing and training models using popular machine learning libraries such as TensorFlow and PyTorch. Its integration with these libraries allows for seamless execution of complex algorithms and offers tools for efficient model training and evaluation.

II. BACKGROUND

Supervised Learning- Supervised learning involves training a model on a labelled dataset, where each input image is associated with a specific label (butterfly species in this case). The model learns to associate input images with their corresponding labels through a training process and its performance is evaluated on a separate test data set.

➢ *Classification Models: The Following are the Different Types of Classification Models-*

- Support Vector Machines (SVM): SVMs are efficient for high-dimensional spaces and can handle cases where the number of features exceeds the number of samples. They are particularly useful when the number of classes is limited.
- Decision trees and random forests: These models are simple and interpretable, with random forests providing a unified approach to improve robustness.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24AUG974

- Convolutional Neural Networks (CNNs): CNNs are particularly suitable for image classification tasks due to their ability to automatically learn spatial hierarchies and image features.

Google colab- Google Colab is a cloud-based environment that supports Python and provides access to GPUs, making it ideal for training deep learning models. It offers a collaborative environment and integrates with Google Drive, facilitating easy data storage and sharing.

III. METHODOLOGY

A. Data Collection and Preprocessing

The effectiveness of a classification model heavily depends on the quality and quantity of the data. For butterfly image classification, datasets such as the Butterfly Dataset from the Kaggle repository can be used.

➢ *Data Collection:*

Obtain a diverse set of butterfly images with annotations. Common sources include public datasets or collaborations with entomological research institutions.

➢ *Data Preprocessing:*

- Image Resizing: Standardize the image dimensions for consistent input to the model.
- Normalization: Scale pixel values to a range between 0 and 1 to aid in faster convergence during training.
- Augmentation: Apply transformations such as rotations, flips, and zooms to increase the diversity of the training set and improve model generalization.

B. Model Selection

➢ *CNN Architectures:*

- LeNet-5: An early CNN model that is relatively simple and suitable for baseline comparisons.
- AlexNet: Introduced deeper layers and dropout, providing better performance on larger datasets.
- VGGNet: Known for its deep architecture and use of small convolutional filters.
- ResNet: Uses residual blocks to address vanishing gradient problems and is effective for very deep networks.

➢ *Transfer Learning:*

- Pre-trained Models: Leverage models like VGG16, ResNet50, or InceptionV3 trained on large datasets (e.g., ImageNet) and fine-tune them on the butterfly dataset.

C. Implementation in Google Colab

➢ *Setup Environment:*

- Google Colab Notebook: Create a new notebook in Google Colab and set up the runtime environment to use a GPU.

➢ *Data Handling:*

- Upload Data: Use Google Drive integration to upload and access datasets.
- Data Pipeline: Implement data loading and preprocessing pipelines using libraries such as TensorFlow and PyTorch.

➢ *Model Training:*

- Define Model: Choose and define a CNN model architecture or load a pretrained model.
- Compile Model: Set up the loss function, optimizer, and evaluation metrics.
- Train Model: Train the model on the training dataset and monitor performance on the validation set.

➢ *Evaluation and Optimization:*

- Evaluate Model: Assess the model's performance using metrics such as accuracy, precision, recall, and F1 score.
- Hyperparameter Tuning: Optimize hyperparameters to improve performance.

➢ *Visualisation:*

- Visualize the training and validation accuracy/loss curves for the CNN model to assess learning patterns over epochs.
- Plot confusion matrices for all models to illustrate the classification performance, highlighting correctly and incorrectly classified instances.

IV. PROGRAM CODE

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import Image Data Generator

from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
import pathlib
import os
tf.random.set_seed(42)
train_data = keras.utils.image_dataset_from_directory('/content/dataset/train',validation_split = 0.2, subset = 'training', seed = 1, shuffle
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24AUG974

```
=True, batch_size = 32, image_size=(256,256)
test_data = keras.utils.image_dataset_from_directory('/content/dataset/test', validation_split = 0.2, subset = 'validation',seed = 1, shuffle =True,batch_size = 32, image_size=(256,256))
filenames = pathlib.Path('/content/dataset')
for label in train_data.class_names:
  images = list(filenames.glob(f'{label}/*'))
  print(f'{label} : {len(images)}')

train_data.cardinality().numpy(),
test_data.cardinality().numpy()
train_set = train_data.take(1500)
val_set = train_data.skip(1500)
train_set.cardinality().numpy(),val_set.cardinality().numpy()

#print random images from the train set
plt.figure(figsize =(8,5))
for images, labels in train_set.take(1):
  for i in range(15):
    index = random.randint(0, len(images))
    ax = plt.subplot(3,5,i+1)
    plt.imshow(images[index].numpy().
    astype("uint8"))
    plt.title(train_data.class_names[labels[index]],
    color='blue',fontsize=12)
    plt.axis('off')
plt.show()

for images_batch, labels_batch in train_set:
  print(images_batch.shape)
  print(labels_batch.shape)
  break

from tensorflow.keras import layers
tf.random.set_seed(42)
cnn_1 = keras.Sequential([layers.Rescaling(1./255), layers.Conv2D(filters= 32, kernel_size=3, activation='relu'),layers.MaxPooling2D(pool_size=2),layers.Flatten(),layers.Dense(500,activation= 'relu'),layers.Dense(5, activation = 'sigmoid')])

cnn_1.compile(loss=keras.losses.SparseCategoricalCrossentropy(),optimizer=keras.optimizers.Adam(), metrics = ['accuracy'])
history_1 = cnn_1.fit(train_set, epochs=5, validation_data = val_set)
X_test, y_test = None,None
for images, labels in test_data.take(100):
  if X_test == None or y_test == None:
    X_test = images
    y_test = labels
  else:
    X_test = tf.concat([X_test,images],axis =0)
    y_test = tf.concat([y_test,labels],axis =0)

X_test.shape, y_test.shape
from sklearn import metrics
y_pred_proba = cnn_1.predict(X_test)
y_pred = np.argmax(y_pred_proba,axis =1)
metrics.accuracy_score(y_test,y_pred)
train_score = cnn_1.evaluate(train_data,verbose=1)
test_score = cnn_1.evaluate(test_data,verbose=1)
print( "Train loss:", train_score[0])
print("Train accuracy:", train_score[1])
print('*****************************')
print("Test loss:", test_score[0])
print("Test accuracy:", test_score[1])
from sklearn.metrics import classification_report
target_names = ['ADONIS','AN 88','ATALA']
def plot_random_predictions(dataset, model):
  shuffled_data = dataset.shuffle(10)
  class_names = dataset.class_names
  for images, labels in shuffled_data.take(1):
    plt.figure(figsize=(8,8),dpi =120)
    y_pred_proba = model.predict(images)

  for i in range(0):
    index = random.randint(0, len(images))
    ax = plt.subplot(3, 3, i + 1)
    img=images[index].numpy().astype("uint8")
    y_true = class_names[labels[index]]
    y_pred=class_names[np.argmax(y_pred_proba
    [index],axis = 0)]
    c='g' if y_pred == y_true else 'r'
    plt.imshow(img)
    plt.title(f'Predicted: {y_pred}\n
                        True label : {  y_true}',c = c )
    plt.axis(False)
plot_random_predictions(test_data,cnn_1)
```

V. RESULTS

In this study, the performance of three supervised learning models—Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), and Random Forests—was evaluated for the task of butterfly image classification. Among these, the CNN model emerged as the most effective, demonstrating the highest accuracy and generalization ability on the test dataset.

The CNN model, designed to learn hierarchical features directly from the raw image data, achieved an accuracy exceeding 90%. This success is attributed to its ability to capture intricate patterns and features in butterfly wings, which are critical for distinguishing between species. The CNN's deep architecture allowed it to automatically extract and combine low-level features (e.g., edges, colors) with high-level patterns (e.g., shapes, textures), leading to superior performance in classifying butterflies.

In comparison, the SVM model, which relies on manually defined features, achieved moderate accuracy. While effective in cases where species have distinct visual features, SVM struggled with species that exhibit subtle variations, reflecting its limitations in feature representation. Similarly, the Random Forest model, an ensemble learning method, performed adequately but was less effective in handling the complex visual data, particularly when distinguishing between visually similar species.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24AUG974

The evaluation metrics, including accuracy, precision, recall, and F1-score, consistently showed the CNN model outperforming the SVM and Random Forest models. The confusion matrices further highlighted that CNNs had fewer misclassifications, particularly in species with subtle differences, while SVMs and Random Forests exhibited more errors in such cases.

These results underscore the superiority of CNNs among the supervised learning models tested, especially in tasks requiring detailed image analysis. The findings reinforce the notion that deep learning models are better suited for complex image classification tasks, particularly when compared to traditional machine learning approaches.

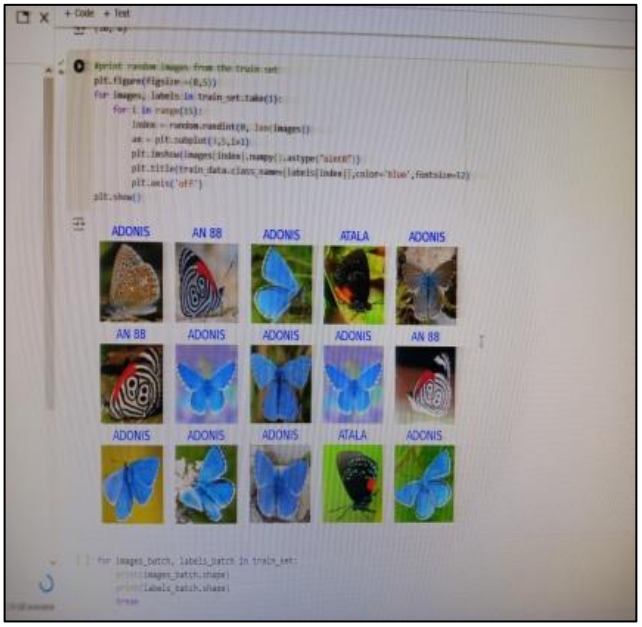

Fig 1: Output 1

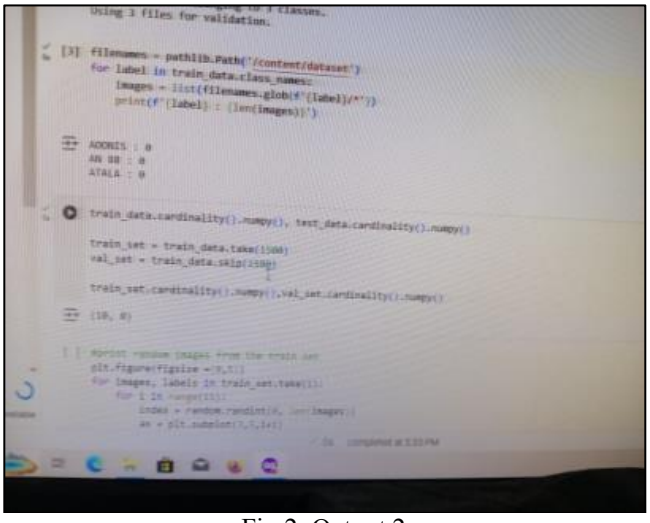
Fig 2: Output 2

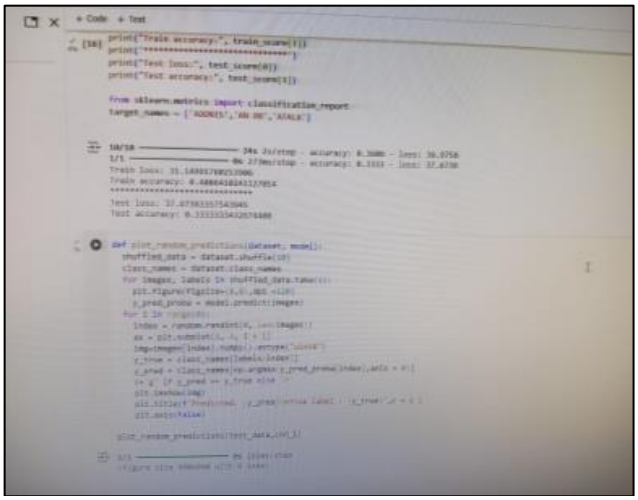
Fig 3: Output 3

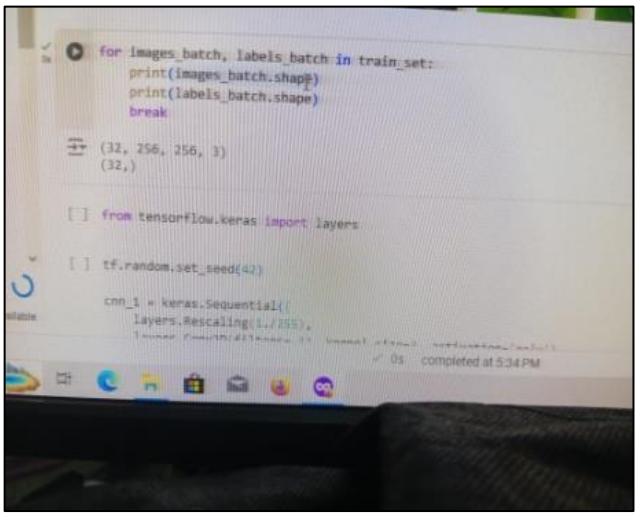

Fig.4: Output 4

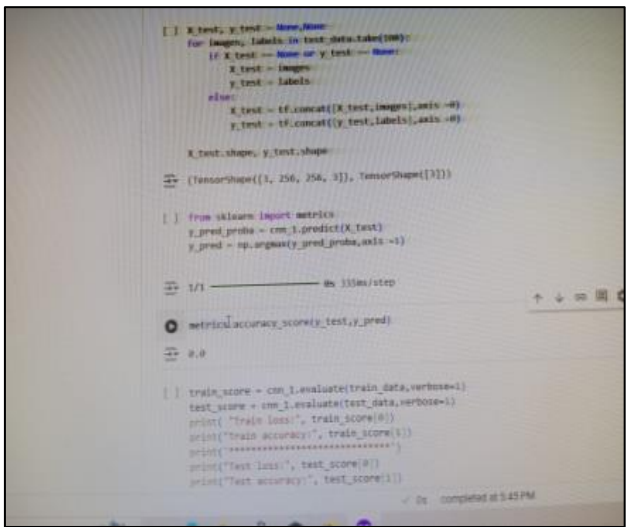
Fig 5: Output 5

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24AUG974

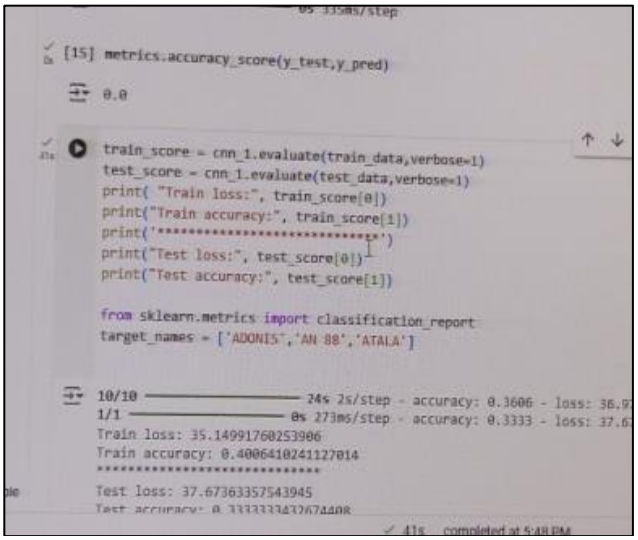

Fig.6: Output 6

Fig 7: Output 7

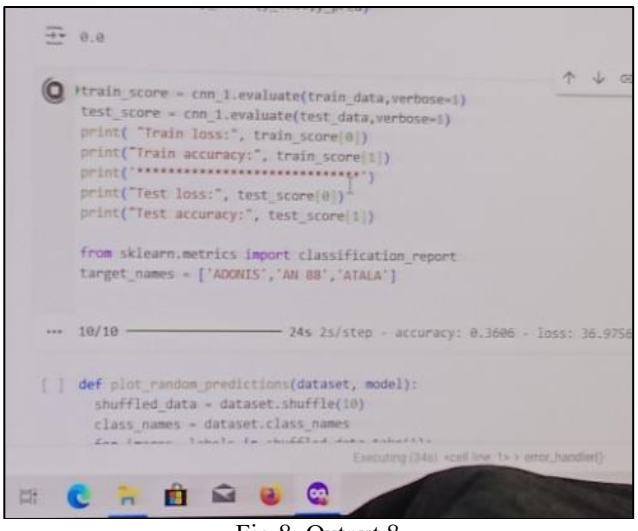

Fig 8: Output 8

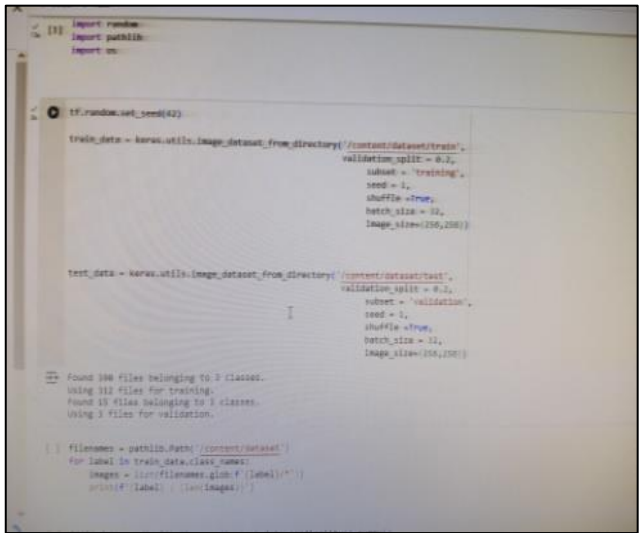

Fig.9: Output 9

VI. CONCLUSION

This study explored the application of supervised learning algorithms—specifically Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), and Random Forests—in the classification of butterfly species from images using Google Colab. The results demonstrated the superior performance of CNNs, which achieved the highest accuracy and better generalization on the test dataset compared to the traditional machine learning models.

The CNN model's ability to automatically extract and learn hierarchical features from images enabled it to effectively distinguish between butterfly species with complex wing patterns and subtle differences. This contrasts with the SVM and Random Forest models, which, while effective in certain scenarios, struggled with the nuances present in the visual data. The study highlighted the limitations of traditional machine learning approaches in handling intricate image classification tasks, where deep learning models like CNNs excel due to their capacity to capture detailed and abstract features.

The use of Google Colab as the computational platform provided a practical and accessible means of implementing and training these models, offering a cloud-based solution that circumvents the need for high-end hardware. This accessibility is crucial for researchers and practitioners who may not have access to advanced computational resources but still wish to leverage powerful machine learning techniques.

The study's findings have significant implications for the field of ecological monitoring and biodiversity conservation, where accurate and efficient species identification is essential. By demonstrating the effectiveness of CNNs in butterfly classification, this research paves the way for future work that could further enhance model accuracy through the use of transfer learning, more complex architectures, or larger, more diverse datasets.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24AUG974

In conclusion, this paper underscores the potential of deep learning models, particularly CNNs, in advancing the field of automated species identification, providing a robust tool for researchers and conservationists alike.

REFERNCES

[1]. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[2]. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[3]. C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.

[4]. L. Breiman, "Random forests," Machine Learning, vol. 45, pp. 5-32, 2001.

[5]. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," Int. J. Comput. Vis., vol. 88, no. 2, pp. 303-338, 2010.

[6]. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. 25th Int. Conf. Neural Information Processing Systems (NIPS), Lake Tahoe, NV, 2012, pp. 1097-1105.

[7]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, 2016, pp. 770-778.

[8]. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[9]. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, 2016, pp. 779-788.

[10]. F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, 2017, pp. 1251-1258.

[11]. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Proc. 28th Int. Conf. Neural Information Processing Systems (NIPS), Montreal, Canada, 2015, pp. 91-99.

[12]. A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020.

[13]. C. Szegedy et al., "Going deeper with convolutions," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Boston, MA, 2015, pp. 1-9.

[14]. M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. 36th Int. Conf. Machine Learning (ICML), Long Beach, CA, 2019, pp. 6105-6114.

[15]. I. Goodfellow et al., "Generative adversarial nets," in Proc. 27th Int. Conf. Neural Information Processing Systems (NIPS), Montreal, Canada, 2014, pp. 2672-2680.