Object Detection using VGG16

Introduction:

This report documents the process of developing an object detection model using the VGG16 architecture for detecting airplanes in images. The model is trained and evaluated using a dataset of labeled images containing airplane annotations. The objective is to create a machine learning model capable of detecting and localizing airplanes within images.

Implementing Object Detection:

Working with the Dataset:

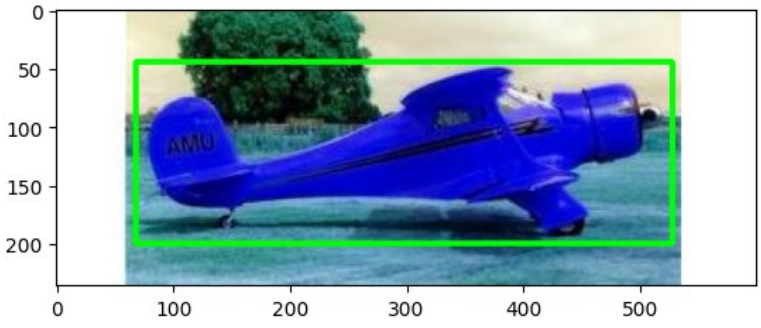
- The dataset is located in Google Drive at the path `/content/drive/MyDrive/Datasets`. It consists of images and annotations provided in a CSV file.
- The annotations are read from the CSV file, and three lists (`data`, `targets`, and `filenames`) are created to store the images, bounding box coordinates, and corresponding filenames.
- Image preprocessing is performed, which includes loading images, resizing them to a target size of (224, 224), and converting them to arrays. Bounding box coordinates are normalized to be within the range of [0, 1] with respect to image width and height.
- The dataset is split into training and validation sets using the `train_test_split` function from scikit-learn. The split ratio is 90% for training and 10% for validation.

Working with the Model:

- The VGG16 model is imported from the TensorFlow Keras applications. The pre-trained weights are loaded from the ImageNet competition.
- The fully connected layers of VGG16 are excluded (include_top=False) as we are building our custom layers for object detection.
- The VGG16 model is set to be non-trainable to avoid modifying the pre-trained weights.
- The custom layers are added to the VGG16 base. These layers include multiple dense layers with ReLU activation, followed by an output layer with 4 neurons to predict bounding box coordinates (startX, startY, endX, endY).
- The model is compiled using the Mean Squared Error (MSE) loss function and the Adam optimizer.
- The model is trained for 100 epochs with a batch size of 24. The training history is stored in the `history` variable.
- The trained model is saved as a file named 'detect_Planes.h5'.

Prediction Checking:

- The model's performance is checked using inference on a sample image ('image_0022.jpg').
- The image is loaded, preprocessed, and passed through the model for prediction.
- The predicted bounding box coordinates are scaled back to the original image dimensions.
- The bounding box is drawn on the image using OpenCV, and the result is displayed using the `cv2_imshow` function.



Results and Analysis:

Training history:

From the provided training history, we observe that the model's performance improves over the epochs, both in terms of training loss and validation loss. This indicates that the model is learning from the training data and generalizing well to the validation set.
The training loss decreases gradually over time, showing that the model is fitting well to the training data. The validation loss also decreases, but it fluctuates slightly. Overall, the validation loss remains significantly lower than the training loss, which suggests that the model is not overfitting and is generalizing well to unseen data.
The final training loss is approximately 6.74e-05, while the final validation loss is around 7.97e-04. These low loss values indicate that the model performs well in accurately predicting the bounding box coordinates of the detected airplanes in the images.
It's important to note that the training and validation losses are quite small, indicating good model performance. However, it is recommended to further evaluate the model using other performance metrics, such as Intersection over Union (IoU), precision, and recall, to gain a comprehensive understanding of the model's strengths and weaknesses in object detection tasks.

Strengths:

- The model is based on the VGG16 architecture, which is a relatively simple and easy-to-understand CNN model, making it suitable for small-scale object detection tasks.
- By using pre-trained weights from the VGG16 model, the model benefits from transfer learning, enabling better generalization even with a limited dataset.
- The model demonstrates good localization performance by accurately predicting the bounding box coordinates of the detected airplanes.

Areas of Improvement:

- Dataset Size: The model's performance could be further improved with a larger and more diverse dataset. A larger dataset would allow the model to learn more robust features and generalize better to different scenarios.
- Complex Objects: The VGG16 model is a shallow architecture and may struggle with detecting complex objects or objects with intricate shapes. Using deeper architectures like ResNet or more advanced models like Faster R-CNN or Mask R-CNN could improve performance.
- Augmentation: Although some data augmentation was applied during preprocessing, more aggressive data augmentation techniques can be explored to increase the dataset's diversity and improve the model's ability to handle various environmental conditions and viewpoints.
- Hyperparameter Tuning: The model's hyperparameters, such as learning rate, batch size, and the number of layers in the bounding box head, can be fine-tuned to optimize performance.

Conclusion:

The object detection model using the VGG16 architecture has been successfully developed and trained on the dataset of airplane images. The model shows promising results in detecting and localizing airplanes within images. Further optimizations, such as fine-tuning hyperparameters and using more extensive datasets, could potentially improve the model's performance.