Midterm Object Detection Challenge

Martin Demel

Department of Science, Technology, Engineering & Math, Houston Community College

ITAI-1378: Computer Vision Artificial Intelligence

Patricia McManus

November 30th, 2024

Introduction

The midterm project's focus was to correctly set and enhance the performance of an object detection model, for which we have specific requirements as part of the midterm challenge. That included the use of a suitable dataset, computational resource efficiency, and the application of various machine learning techniques. This report reflects my journey, including the challenges I initially faced with Amazon SageMaker and later with the CIFAR-10 model that I selected. This report also includes lessons learned, a reflective review, and the outcomes of the laboratory work as such.

Initial setup of the workplace

Initially, I decided to proceed with Amazon SageMaker, as it's offered to us as students and is a cloud-based environment. However, running a project of this magnitude, I realized that I would benefit from the performance of my personal laptop due to the limitations of the Amazon SageMaker GPU. I was unable to start the sessions, including GPU, because, at the time, there were no resources available; that was the message.

My MacBook PRO, with an M3 Max processor and 48GB of RAM, provided sufficient performance that was beneficial for this midterm challenge. During the setup, I faced one significant issue with version incompatibility between the Metal plugin and Tensorflow. Additionally, Metal required a specific Kernel version to run properly. The issue was resolved by installing a properly supported version in a virtual environment—Python 3.10.15. Finally, after the issue was addressed, this setup allowed me to use the TensorFlow GPU acceleration, which significantly reduced training times and improved the overall efficiency of the object detection activities.

Dataset Selection and change of the dataset model

<u>CIFAR-10 Dataset – Initial dataset</u>

I initially started the project by selecting the CIFAR-10 dataset, which was offered inside the Jupyter Notebook. After some research, I learned that it's a small, well-structured dataset that is widely used. Although the model is commonly used for classification tasks, adapting it for object detection proved me wrong when I selected it. The low resolution of the images (32x32) was challenging. Over the five epochs, the model achieved a low average accuracy of 10% and approximately 0.50 AUC, which indicated severe underfitting

To address the challenges, I tried the following steps:

- **Preprocessing enhancements**: I resized images to higher dimensions (224x224) to make them more suitable for transfer learning with pre-trained models. I have aimed to provide the network with more special information, but I have been unsuccessful.
- **Data augmentation**: I applied various augmentation techniques such as random horizontal flipping, contrast variations, and brightness adjustments. With the brightness, I faced another challenge where the picture would be black. I had to ensure the adjustment was within limits to prevent the image from going black. Nevertheless, the augmentation did not enhance the model performance due to the simplicity and low resolution of the dataset from preprocessing.
- **Learning rate:** I have made numerous adjustments to learning rates, including ranges from 0.01 to 0.0001. While this slightly improved performance and stabilized the training process, it failed to address the dataset's underlying limitations: image resolution.
- **Model training adjustments**: I have increased the number of epochs from 5 to 20. I have implemented an early stop to monitor the validation loss, which would

eventually stop the training if no improvements were observed. Furthermore, I have experienced optimizers such as Adam and AdamW and incorporated a learning rate scheduler to adjust the learning rate dynamically during training. I have observed only suboptimal performance improvements.

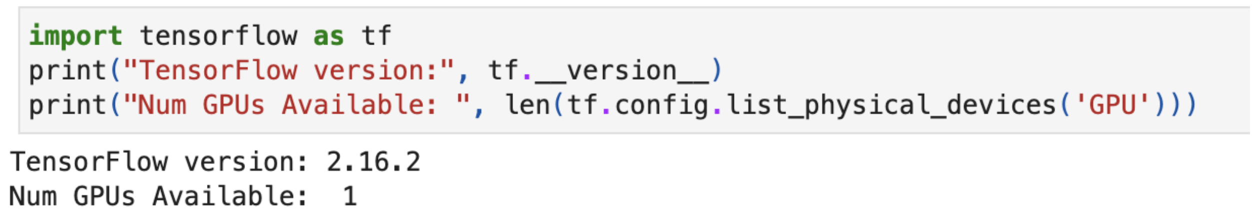
Trying to address my challenges, I realized that the CIFAR-10 model was limited by default due to the dataset's limitations, including the low image resolution and lack of objects. These limitations motivated me to Speak to my Professor and review the possibility of using the PASCAL VOC 2007 dataset, which offers better features and seems better aligned and suited for object detection tasks.

PASCAL VOC 2007 Final Dataset

Recognizing the limitations of the CIFAR-10 dataset, I have transitioned to the Pascal VOC 2007 dataset. This dataset features 20 object categories and rich annotations, ultimately providing me with a more suitable challenge to address. The high resolution of the images and diverse objects offered the depth that was key for transfer learning.

Steps performed

Installation: First, I installed all the required software and libraries to run all parts of the code. Second, I ensured that the GPU was available to support the model training processing—Figure 1 GPU Acceleration check.



```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
TensorFlow version: 2.16.2
Num GPUs Available:  1
```

Data preprocessing: I have resized images to 224x224 pixels, as I have learned that this ensures compatibility with the transfer learning models. Furthermore, I have normalized the images to the [0, 1] range to facilitate faster convergence during training and one-hot encoding for multi-label classification, which ensures it can handle images containing multiple objects from different classes. By doing so, I have ensured that the transfer learning models would accept the images.

Data Augmentation: To ensure that the performance and generalization of the machine learning models are improving, I have set a series of data augmentation techniques, including:

- Random horizontal and vertical flip – to simulate different viewing angles.
- Brightness and contrast adjustments – to simulate varying lighting conditions.
- Saturation adjustments
- Added Gaussian Noise – This helped the model to become more robust to imperfection in image quality.

I have performed numerous combinations to find the right augmentation mix, which is the perfect balance for data augmentation. As mentioned earlier, I have faced brightness issues for CIFAR-10 and the PASCAL VOC 2007 dataset. I addressed this by adjusting the brightness to acceptable levels. Then, I visualized the TensorFlow-based augmentation images, as can be seen in Figure 2.

Transfer learning: I had to adapt MobileNet V2 to the PASCAL VOC 2007 dataset. To do so, I froze the model's initial 40 layers. This allowed me to ensure that it retained the low-level feature learned from ImageNet. Next, I have added a custom top layer to specialize the model for the new dataset. A GlobalAveragePooling2D layer replaced the fully connected layers to reduce overfitting by minimizing the number of parameters.

During the fine-tuning, I faced performance issues, and I learned that Dropout plays a significant role in overfitting, It randomly deactivated neurons during the training. I experimented with numbers ranging from 0.2 to 0.6, and even small adjustments had a significant impact on the model training. So, I found that 0.3 is the optimal setting.

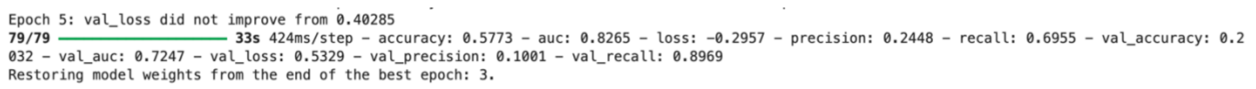
The output layer was dense, with 20 units and sigmoid activation. Initially, I had 10 with SoftMax activation for the CIFAR 10 model, but this needed to be changed for the PASCAL VOC 2007.

Model Training: To enhance the training process, I have implemented a One-Cycle learning rate scheduler to optimize the convergence. Starting with the small learning rate allowed me to gradually increase to a peak before decaying.

Class weights were the second important step in optimizing the model's training performance. During the testing phase, I faced issues with class imbalances, so I calculated dynamic lass weights based on the label distributions. This helped me to address the balance of the loss function.

Furthermore, two callbacks, Earlystopping and ModelCheckpoint, were added to prevent overfitting and save the best model based on the validation loss. Since we used multi-label classification, the model was compiled with BinaryCrossentropy for comprehensive evaluation.

After the initial training phase, which had 5 epochs, we achieved approximately 60% accuracy and around 40 % validation accuracy. The significant gap was due to overfitting. The exact values can be found in Figure 3—Epoch 5 final values.



```
Epoch 5: val_loss did not improve from 0.40285
79/79 ━━━━━━━━━━━━━━━━━━━━ 33s 424ms/step - accuracy: 0.5773 - auc: 0.8265 - loss: -0.2957 - precision: 0.2448 - recall: 0.6955 - val_accuracy: 0.2
032 - val_auc: 0.7247 - val_loss: 0.5329 - val_precision: 0.1001 - val_recall: 0.8969
Restoring model weights from the end of the best epoch: 3.
```

Enhanced training:

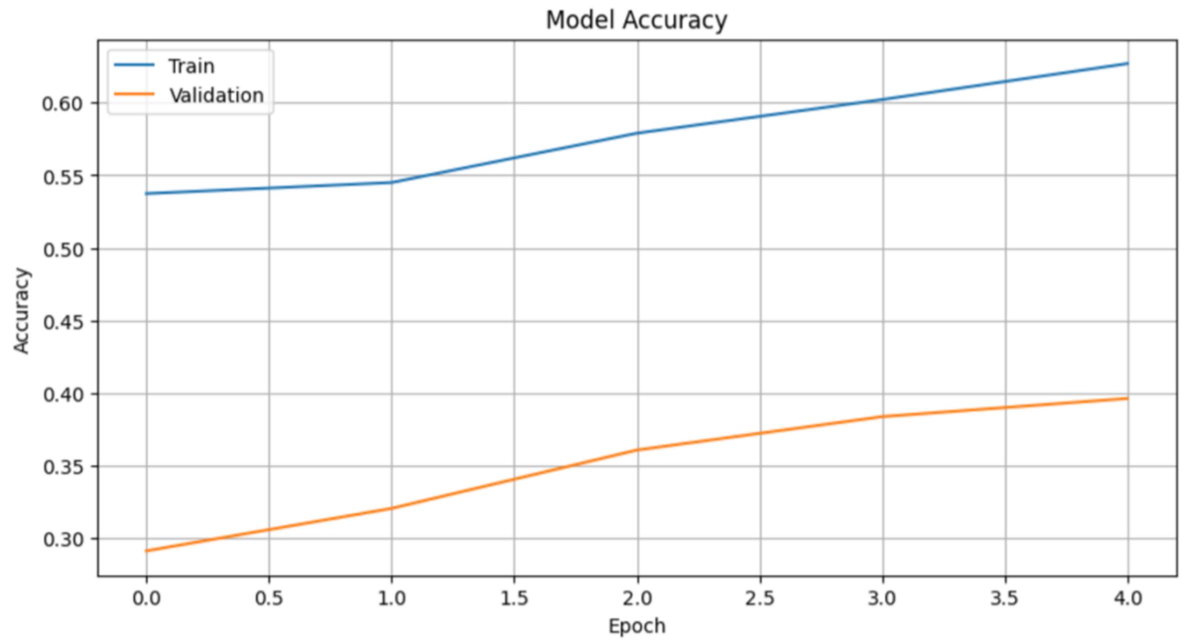
To address the overfitting and overall performance of the model, I have enhanced the training to address the specific issues. This included advanced data augmentation and hyperparameter tuning. By doing so, the performance showed some improvement. The training accuracy increased to approximately 60%, while the validation accuracy stabilized at 30%. Precision was limited to 12%, while recall achieved a high of 86%, indicating the model effectively identified true positives but struggled with precision. The AUC score reached 78.8%, reflecting the model's solid ability to distinguish between classes across various thresholds.

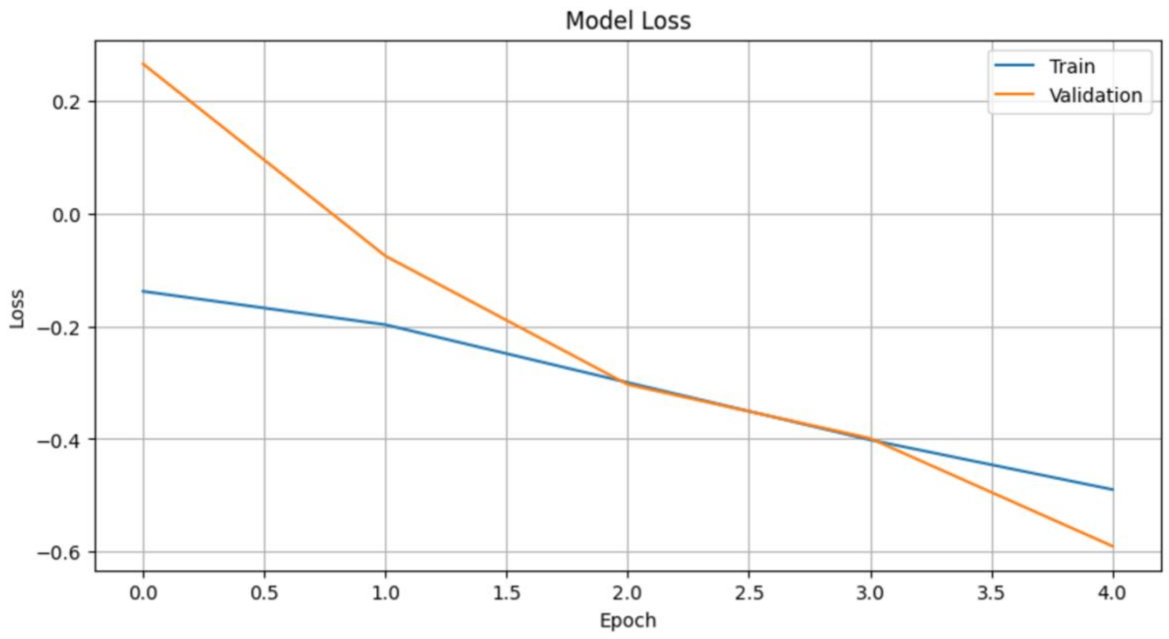
The results represented significant improvements over CIFAR-10, demonstrating the value of better datasets and tailored preprocessing techniques.

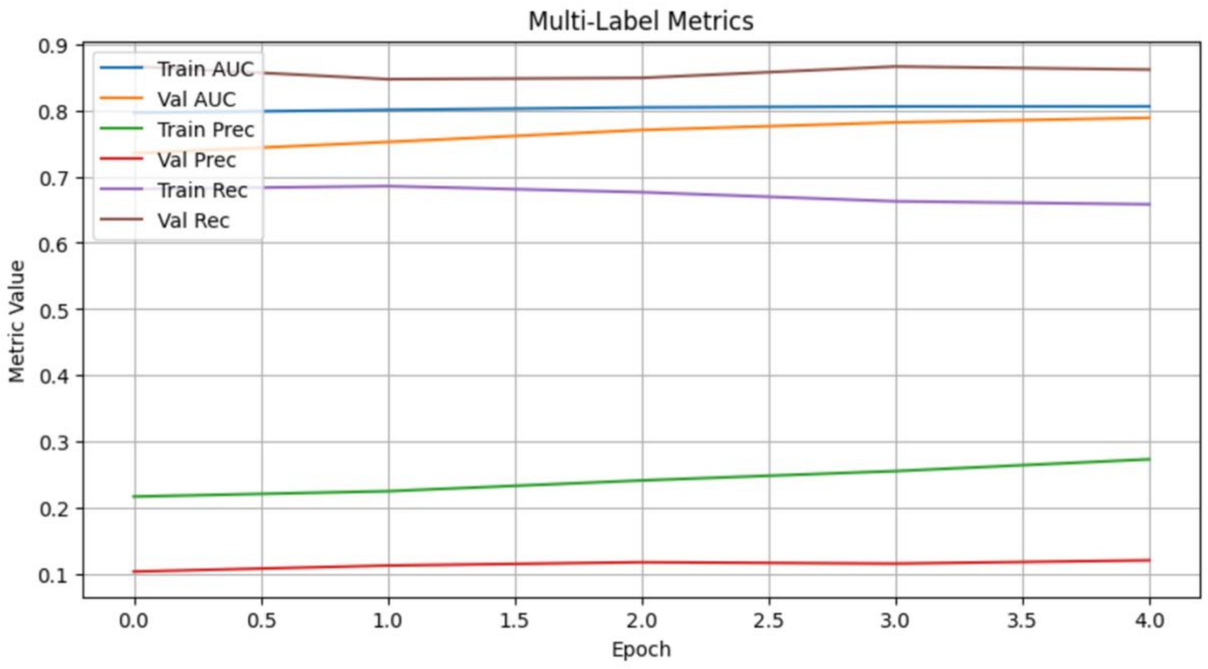
Something that caught my eye was the negative loss values during the training. This is, to my understanding, typical for the Binary cross-entropy loss function. It suggested potential issues in the loss computation or data handling process, such as incorrect label encoding or improper application of class weights.

Performance Metric	**CIFAR-10**	**PASCAL VOC 2007**
Training Accuracy	~10%	~60%
Validation Accuracy	~10%	~40%

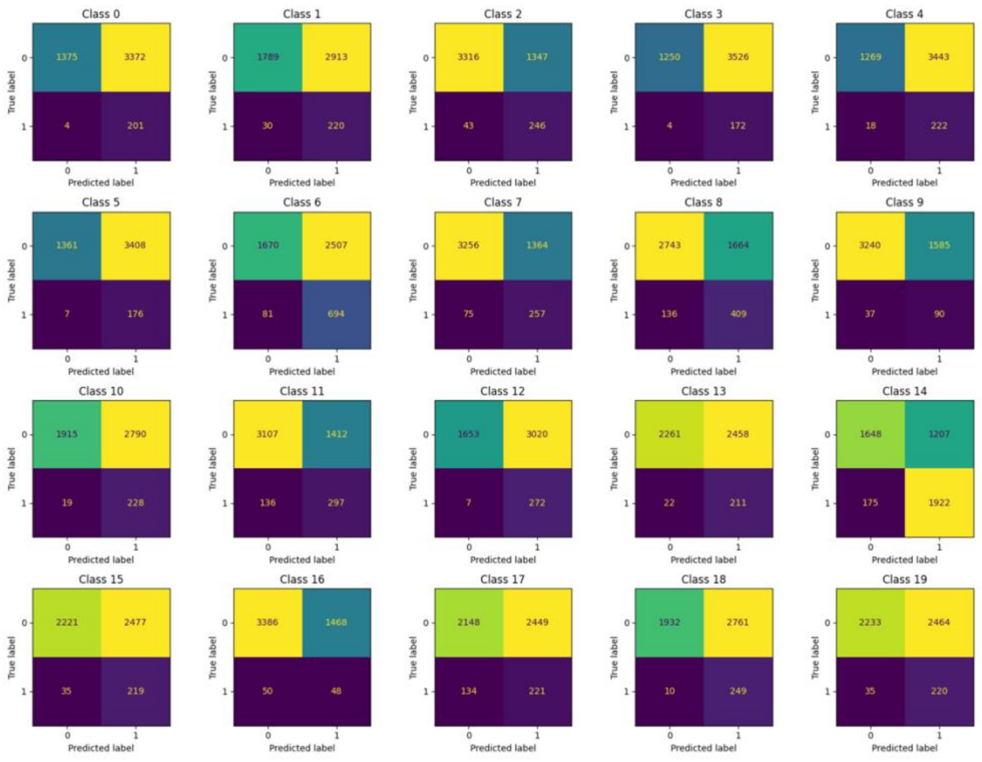
Visualization: In order to ensure that data are presented correctly, I had to perform the adjustments to the code. Furthermore, I have added multi-label metric visualization that includes AUC, Precision, and Recall). This can be seen in Figure 4,5,6—Data visualization.







Confusion Matrix: In this section, I have calculated and visualized the confusion matrix for each class in the test dataset. First, we gathered the true labels, predicted the probabilities, and converted them into binary values using the threshold.



Secondly, I ensured that the shapes and formats matched correctly. Using the multilabe_confusion_matrix function, I displayed the confusion matrix in grid view, as shown in Figure 7.

The confusion matrices show that while the model performs reasonably well for certain classes, it also struggles with others. This is due to high misclassification rates in many cases. To address this, further adjustments to data preprocessing, enhanced augmentation, or improved model architecture would be required. This would allow me to achieve more consistent and accurate performance across all classes.

Conclusion

The midterm project to enhance the object detection models' performance has been challenging and rewarding. I have learned a valuable lesson about the importance of selecting the right strategic dataset to start with. The project continues with rigorous preprocessing and precise model optimization, which can ultimately lead to significant improvements.

This project deepened my understanding of object detection and technical skills while showing me the opportunities in real-life scenarios. This project is one of the reasons I would like to continue my study path at HCC and continue to explore the application of Computer vision.

Resources:

GeeksforGeeks. (2024, June 12). What is Object Detection in Computer Vision? GeeksforGeeks. https://www.geeksforgeeks.org/what-is-object-detection-in-computer-vision/

GeeksforGeeks. (2024, September 20). CIFAR10 Image classification in TensorFlow. GeeksforGeeks. https://www.geeksforgeeks.org/cifar-10-image-classification-in-tensorflow/

GeeksforGeeks. (2022, December 6). YOLO v2 Object Detection. GeeksforGeeks. https://www.geeksforgeeks.org/yolo-v2-object-detection/