Project 3: Artificial Neural Networks

Ravi Teja Anumula (UFID: 73522006)
Dept. of Electrical and Computer Engineering
University of Florida
Gainesville, USA

***Abstract*—This project explores the application of artificial neural networks for two distinct tasks: flower species classification and car object detection. The flower classification task involves a dataset of 10 flower species and requires the development of a multi-class classifier. For car object detection, the YOLOv5m pre-trained model serves as the base architecture, fine-tuned using a dataset with labeled bounding boxes. To generate ground truth for car detection, the MakeSense platform is utilized to annotate images and prepare the dataset. The project emphasizes deep learning best practices such as hyperparameter tuning, checkpointing, early stopping, and adaptive learning rates. Transfer learning optimizes performance for both tasks, and the models are evaluated using quantitative and qualitative metrics.**

***Index Terms*—Artificial Neural Networks (ANNs), TensorFlow, checkpointing, early stopping, Hyperparameter tuning, YOLOv5m, MakeSense Annotation**

I. Introduction

The advancement of artificial intelligence and deep learning has revolutionized various industries, enabling the automation of complex tasks such as image classification and object detection.This project focuses on using artificial neural networks (ANNs) to solve two such tasks: identifying flower species and detecting cars in images. Both tasks demonstrate the power and flexibility of modern machine learning techniques in addressing real-world problems.

For the flower classification task, the goal is to accurately identify 10 different types of flowers using a dataset of RGB images. This involves designing a multi-class classifier and optimizing its performance through careful model tuning and experimentation.

For car detection, we use the YOLOv5m model, a pre-trained object detection network, as our foundation. By fine-tuning this model on a custom dataset annotated using the MakeSense platform, we create a system capable of locating and identifying cars within images. Using transfer learning allows us to build on the strengths of a proven architecture while tailoring it to the specific dataset, saving both time and computational resources.

Throughout the project, we emphasize the use of best practices in deep learning, such as early stopping to prevent overfitting, checkpoints to save progress, and adaptive learning rates to improve training efficiency. To evaluate the models, we employ a combination of quantitative measures, such as accuracy and loss, and qualitative approaches, like visualizing the detected bounding boxes.

II. Methodology

A. Flower Classification Task

a) ***Preparing the Dataset****:* The dataset used for flower species classification consists of 1,658 images from 10 different classes, each assigned a unique label. The images were resized to 300×300 pixels and normalized by dividing all pixel values by 255 to ensure consistency. The dataset was then split into two parts: 80% for training and 20% for validation, ensuring that the class distribution remained balanced.

b) ***Building the Model****:* A convolutional neural network (CNN) was designed using TensorFlow and Keras. The model architecture was fine-tuned to achieve the best results using Keras Tuner.The model was optimized using:

- The Adam optimizer with a learning rate of 0.0001 for efficient training
- Sparse categorical crossentropy as the loss function to handle multi-class classification.
- Accuracy as the performance metric to measure the success of the model.
- A softmax output layer with 10 units to classify the images into the respective flower categories.

c) ***Fine-Tuning with Hyperparameters****:* Key hyperparameters, such as the number of filters, kernel size, dense layer units, and learning rate, were optimized through a random search using Keras Tuner. The best combination was selected based on the model's validation accuracy

d) ***Training and Validation****:* The model was trained on the training dataset for a maximum of 20 epochs. To ensure effective training and prevent overfitting

- Early stopping was implemented, monitoring validation loss with a patience of 3 epochs
- A checkpoint system saved the best-performing model during training.

e) ***Model Evaluation****:* The trained model was tested on a separate test dataset, where it achieved an accuracy of 85%. The model's performance was evaluated using:

- A confusion matrix to analyze predictions for each class and highlight any misclassifications
- A classification report providing detailed metrics like precision, recall, and F1-score for each flower species.
- Learning curves showing the trends in accuracy and loss during training and validation.

B. Car Detection Task

*a) **Dataset Preparation**:*

- The dataset for car detection consisted of images annotated with bounding box labels in YOLO format.
- A Images were divided into training (80%) and validation (20%) sets.
- Bounding box coordinates were normalized, and labels were saved in YOLO-compatible .txt files.

*b) **Model Architecture**:*

- The YOLOv5m pre-trained model was used as the base architecture.
- The model was fine-tuned on the car detection dataset to improve detection accuracy for this specific task.

*c) **Training**:*

- Training was performed with the YOLO framework, using a batch size of 16 and image size of 640×640 pixels.
- Two training configurations with 10 and 20 epochs were tested to identify the best-performing model based on mAP (mean average precision).
- Automatic mixed precision (AMP) was used to optimize training efficiency.

*d) **Validation and Selection**:*

- After training, validation metrics such as precision, recall, and mAP were calculated.
- The model achieving the highest mAP was selected as the best-performing version for further testing.

C. Test Set Evaluation Using Make Sense

GridSearchCV and cross-validation were used for tuning each model-dimension reduction pair to avoid overfitting and achieve optimal performance.

*a) **Dataset Preparation**:*

- Test images were annotated using the MakeSense platform to create ground truth labels for evaluation.
- Annotations were saved in YOLO-compatible format.

*b) **Model Evaluation**:*

- The trained YOLOv5m model was tested on the prepared test set.
- Bounding boxes predicted by the model were compared with ground truth labels.
- Intersection over Union (IoU) was used to assess detection accuracy, with a threshold of 0.5 for a match.

*c) **Performance Metrics**:*

- Metrics such as precision, recall, and mAP were computed to evaluate model performance.
- Qualitative evaluation included overlaying predicted and true bounding boxes on images.
- Sample images with true (blue) and predicted (green) bounding boxes were visualized to demonstrate the model's detection capabilities.

This methodology highlights the systematic approach to training and evaluating the YOLOv5m model for car detection, ensuring robust and accurate performance through quantitative and qualitative analyses.

III. Observations

A. Flower Classification

*a) **Hyperparameter tuning**:* The hyperparameter search conducted using Keras Tuner identified the following optimal values:

- Filters in the convolutional layer: 64.
- Kernel size: 3
- Dense layer units: 256
- Learning rate: 0.0001

*b) **Performance metrics in Training and validation**:*

- Achieved 100% accuracy on the training set, indicating the model learned the patterns within the training data effectively.
- Precision, recall, and F1-scores were perfect for all 10 flower classes in the training set
- Achieved 64% accuracy on the validation set, showing room for improvement in generalization.
- Best F1-scores: "Sunflowers" (0.81), "Marigold" (0.76)
- Lower F1-scores: "Lilies" (0.36), "Bougainvillea" (0.50), indicating these classes were harder for the model to classify.

Classification Report on Train Data:

	precision	recall	f1-score	support
Roses	1.00	1.00	1.00	141
Magnolias	1.00	1.00	1.00	144
Lilies	1.00	1.00	1.00	164
Sunflowers	1.00	1.00	1.00	112
Orchids	1.00	1.00	1.00	138
Marigold	1.00	1.00	1.00	125
Hibiscus	1.00	1.00	1.00	128
Firebush	1.00	1.00	1.00	138
Pentas	1.00	1.00	1.00	130
Bougainvillea	1.00	1.00	1.00	106
accuracy			1.00	1326
macro avg	1.00	1.00	1.00	1326
weighted avg	1.00	1.00	1.00	1326

Fig. 1. Classification Report on Train Data

Classification Report on Validation Data:

	precision	recall	f1-score	support
Roses	0.61	0.64	0.62	36
Magnolias	0.67	0.86	0.76	36
Lilies	0.41	0.32	0.36	41
Sunflowers	0.88	0.75	0.81	28
Orchids	0.70	0.74	0.72	35
Marigold	0.68	0.87	0.76	31
Hibiscus	0.58	0.59	0.58	32
Firebush	0.83	0.74	0.78	34
Pentas	0.74	0.44	0.55	32
Bougainvillea	0.45	0.56	0.50	27
accuracy			0.64	332
macro avg	0.65	0.65	0.64	332
weighted avg	0.65	0.64	0.64	332

Fig. 2. Classification Report on Validation Data

*c) **Learning curves for Training and Validation dataset**:*

- The training accuracy improved steadily, reaching 100% after 11 epochs.
- Validation accuracy plateaued around 64%, indicating potential overfitting after a certain point.

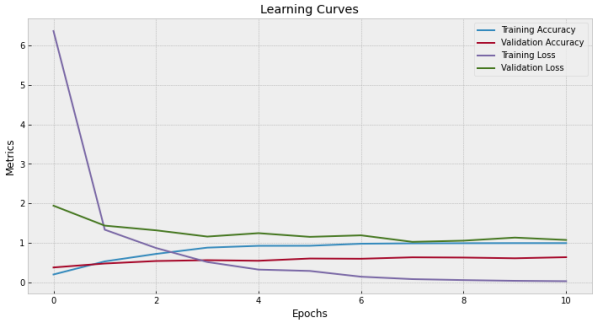

Fig. 3. Learning curves for Training and Validation dataset

*d) **Test Set Results**:* On a separate test dataset:

- Accuracy was 85%, showcasing better generalization on unseen data compared to the validation set.
- Highest performance for "Sunflowers" and "Magnolia" with high recall and precision.
- Lower performance for "Lilies" and "Roses," with relatively higher misclassifications.

Class	***TP***	***TN***	***FP***	***FN***
Roses	38	356	11	10
Magnolia	42	370	1	2
Lilies	26	364	5	20
Sunflowers	35	379	0	1
Orchids	37	361	9	8
Marigold	26	375	0	14
Hibiscus	41	367	5	2
Firebush	35	364	14	2
Pentas	31	379	4	1
Bougainvillea	42	358	13	2

TABLE I
CONFUSION MATRICES FOR TESTING DATA

	precision	recall	f1-score	support
0.0	0.78	0.79	0.78	48
1.0	0.98	0.95	0.97	44
2.0	0.84	0.57	0.68	46
3.0	1.00	0.97	0.99	36
4.0	0.80	0.82	0.81	45
5.0	1.00	0.65	0.79	40
6.0	0.89	0.95	0.92	43
7.0	0.71	0.95	0.81	37
8.0	0.89	0.97	0.93	32
9.0	0.76	0.95	0.85	44
accuracy			0.85	415
macro avg	0.87	0.86	0.85	415
weighted avg	0.86	0.85	0.85	415

Fig. 4. Classification Report on Test Data

B. Car Detection Task

*a) **Hyperparameter Tuning**:*

- The YOLOv5m pre-trained model was fine-tuned on a custom car detection dataset.
- Models were trained for 10 and 20 epochs with a batch size of 16 and an image size of 640×640.
- The best-performing model was selected based on mAP50-95 (mean Average Precision at IoU thresholds 0.5 to 0.95), which achieved a peak value of 0.703 after 20 epochs.

*b) **Training and Validation Results**:*

- Steady decrease in Box Loss and Classification Loss during training over 20 epochs.
- Box loss reduced to 1.136, and classification loss dropped to 0.5283 by the end of training
- Validation results are: mAP@0.5 (IoU threshold 0.5): 0.972 and mAP@0.5-0.95: 0.703
- Achieved a precision of 97.2% and Recall of 99.6%

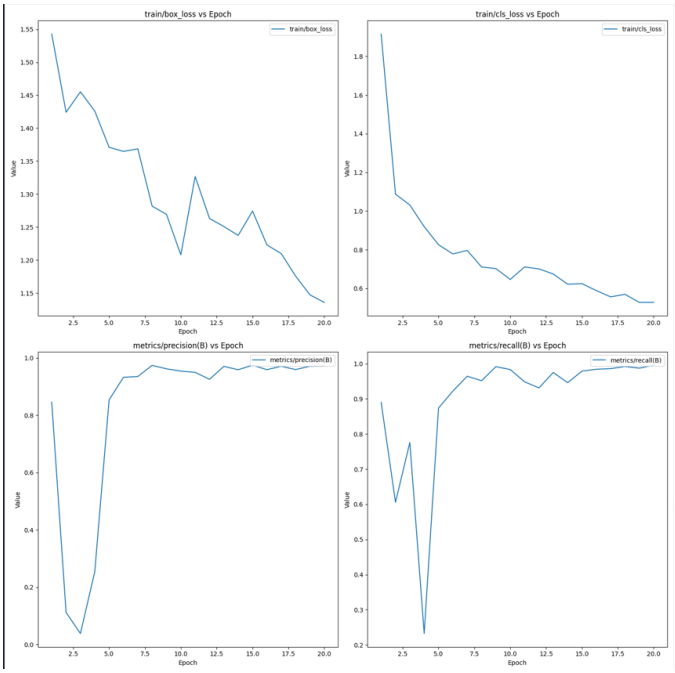

Fig. 5. Learning curves for Training and Validation dataset

*c) **Test set results**:*

- The YOLOv5m model effectively detects single and multiple cars in various scenarios, showcasing accurate bounding box placement and high confidence scores.
- In the first image, three cars are detected with confidence scores of 0.73, 0.79, and 0.9, while in the third image, a single car is detected with a confidence score of 0.85.
- In the second image, where no cars are present, the model avoids false positives, demonstrating robust filtering of irrelevant objects and background.
- Bounding boxes are well-aligned with detected objects, accurately enclosing cars without significant errors. However, in crowded scenes (e.g., Image 1), slight overlapping

of bounding boxes between adjacent objects could be improved with further fine-tuning.

Fig. 6. Image showing multiple car

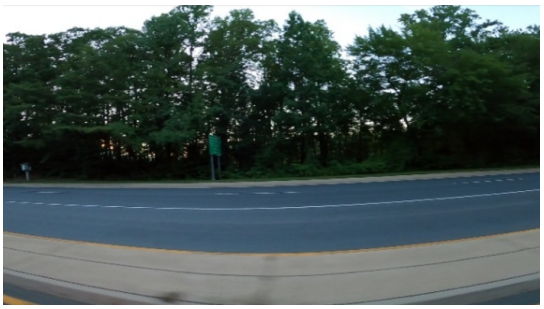

Fig. 7. Image showing no car

Fig. 8. Image showing 1 car

C. Testing in Absence of Ground Truth

To assess how well the model performs without relying solely on labels, you can involve a small group of people for additional validation. Select a small set of test images—say, 50 or 100—and show these images to a few individuals, asking them to mark areas where they see cars. Afterward, compare their annotations with the predictions made by your model. This will give you a clearer picture of how accurate your model's predictions are in real-world scenarios.

For images without cars, you can train the model to recognize empty scenes as well. When testing, if the model doesn't detect any cars in an image, you can verify this by reviewing the image manually or asking your human helpers to confirm.

This approach provides a practical way to gauge your model's effectiveness, especially in cases where some images contain cars and others do not. It acts as a reality check, using human judgment to evaluate how well the model is performing.

D. Test Set Evaluation Using Make Sense

*a) **Visualization and Analysis**:*

- Make Sense annotations (true boxes) were visualized in blue, while YOLOv5m's predictions (predicted boxes) were displayed in green for side-by-side comparison.
- This visual overlay provided an intuitive way to validate predictions against manually created ground truth, highlighting any discrepancies.

*b) **Testing Results:Ground Truth vs. Detected Cars**:*

- Total Cars in Images: 4
- Total Cars Detected: 2
- Detection Accuracy: 50

Fig. 9. Detected cars when present

Fig. 10. Not Detected cars when present

Fig. 11. Not Detected cars when not present and overall accuracy

*c) **Inference**:*

- Instances of correct detections with high confidence were observed, reflecting the effectiveness of both the Make Sense annotations and the model's learning.
- In some crowded or low-contrast scenarios, minor misalignments or missed detections occurred, pointing to areas for improvement in either annotation quality or model tuning.