EEL 5934 Applied Machine Learning - Project 3

Sohaib Uddin Syed
CISE Department
University of Florida
Gainesville, USA
sohaibuddinsyed@ufl.edu

***Abstract*—Convolutional Neural Networks (CNNs) have emerged as a pivotal technology in the field of image recognition and deep learning. CNNs excel in extracting hierarchical features from input images through convolutional layers, enabling them to capture spatial hierarchies and patterns inherent in visual data. This architecture has revolutionized image classification tasks by significantly improving accuracy and efficiency. In this project, we perform classification and object detection tasks using the datasets. We experiments with various architectures and report the observations.**

***Index Terms*—Deep learning, Convolution Neural Networks**

I. Introduction

In this project, we are working with two different dataset:

- Dataset 1: Flower Species Dataset contains a total of 1678 images from 10 classes. Each RBG image is of size 300×300×3. The goal is to utilize the training images to train a flower species classifier (an artificial neural network architecture), and make predictions for the test set. Table 1 shows the various classes in the dataset.
- Dataset 2: Car Detection Dataset contains labeled annotations for 559 training samples. Each annotation corresponds to a bounding box of the object car. The goal is to train an object (car) detection artificial neural network using the training samples, and make predictions for the images in test.

TABLE I
Chart 1: Class-Label Mapping

Flower Species	Label
Roses	0
Magnolias	1
Lilies	2
Sunflowers	3
Orchids	4
Marigold	5
Hibiscus	6
Firebush	7
Pentas	8
Bougainvillea	9

A. Exploratory Data Analysis and pre-processing

In Exploratory Data Analysis, it is crucial to make informed decisions about feature selection to ensure that the data used for further analysis is both meaningful and efficient. One key aspect of this process is the identification and removal of irrelevant or redundant features.

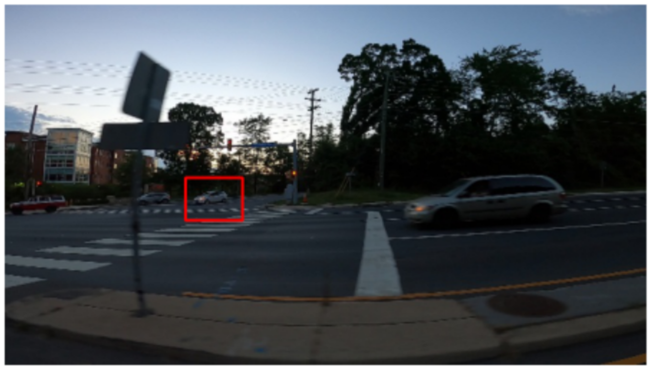

Fig. 1. Labeled test sample from Dataset 2

- Dataset 1: In the dataset, the original shape is a flattened representation of images with three color channels (RGB) and a size of 300x300 pixels. We perform reshape on each smaple and divided by 255.0. This division by 255.0 normalizes the pixel values to a range between 0 and 1, as the original pixel values in the image dataset ranges from 0 to 255. This helps stabilize and accelerate the training process by ensuring that input features are within a consistent scale.
- Dataset 2: We are provided with image samples and their bounding boxes as 2 different inputs. We load images from the file paths and the corresponding labels are extracted from the bounding box information. We prepare the trainning set from the images and use bounding boxes as the target.

Figure 1 shows a sample from the test set along with a label on the car.

II. Methodology

A. Dataset 1: Flower species classification

The goal of this experiment is to design a convolution neural network that is able to solve the multi-class classification problem in the given dataset. In order to conclude with the best architecture, we perform hyper-parameter tuning on the following:

1) Convolution Layers: The architecture includes 3 or 4 convolution layers. The first layer uses filter of size 3x3 i.e. the convolution operation will be performed using a 3x3 window over the input data. Next we have a max pooling layer which

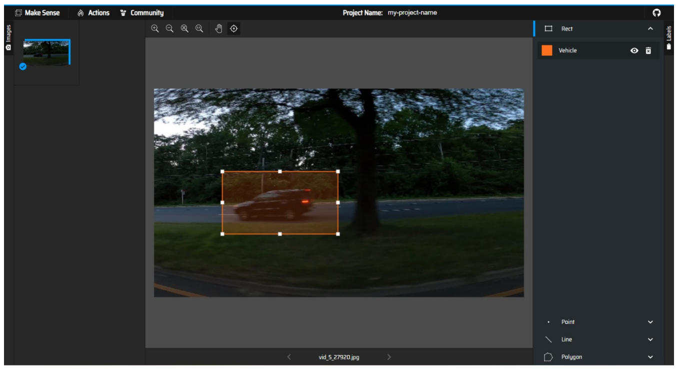

Fig. 2. Manually annotating images with MakesenseAI

uses a pool size of 2, so it divides each spatial dimension by a factor of 2. Since we have a larger image, we repeat this structure several more times (the number of repetitions is a hyperparameter). The number of filters grows as we climb up the CNN toward the output layer (it is initially k, then 2 * k) since the number of low-level features is often fairly low, but there are many different ways to combine them into higher-level features. This is inline with the class discussions to double the number of filters after each pooling layer

2) Dense Layers: After the convolution layers, we have 2 or 3 fully connected network, composed of hidden dense layers and a dense output layer. We flatten its inputs, since a dense network expects a 1D array of features for each instance. We also add two dropout layers, with a dropout rate of 50% each, to reduce overfitting.

3) Activation functions: The tanh function squashes input values to the range of [-1, 1], providing centered outputs around zero. This can mitigate issues related to vanishing gradients, aiding in learning representations with mean-centered data. On the other hand, ReLU, defined as the positive part of its argument, replaces negative inputs with zero while leaving positive values unchanged.

4) Output: The output layer is a Dense layer with 10 units, corresponding to classification task with 10 classes for the given dataset. The softmax activation function is used, which normalizes the output values across the 10 units, transforming them into probabilities. This ensures that the sum of the probabilities for all classes is equal to 1, allowing the model to make a confident prediction about the most likely class. We also employ the 'sparse_categorical_crossentropy' loss function multiclass classification problems since the labels are integers. It calculates the cross-entropy loss between the predicted probabilities and the true labels. Finally, the accuracy metric is chosen as the performance metric to monitor during training.

The dataset is divided into training and validation sets with a 80/20 stratified split, allowing for model evaluation and tuning while maintaining the prior distribution. We perform hyperparameter tuning on the number of layers, the units in each layer, learning rate and dropout value. Other parameters sch as number of units and activation function could not be performed due to computational limits.

B. Dataset 2: Car detection

The goal of this experiment is to design a convolution neural network that is able to solve the object (car) detection problem in the given dataset. The final architecture is mostly similar to the one used in first experiment.

1) Convolution Layers: The architecture includes 3 or 4 convolution layers. The first layer uses filter of size 3x3 i.e. the convolution operation will be performed using a 3x3 window over the input data. Next we have a max pooling layer which uses a pool size of 2, so it divides each spatial dimension by a factor of 2. Since we have a large image, we repeat this structure several more times (the number of repetitions is a hyperparameter). The number of filters grows as we climb up the CNN toward the output layer (it is initially k, then 2 * k) since the number of low-level features is often fairly low, but there are many different ways to combine them into higher-level features. This is inline with the class discussions to double the number of filters after each pooling layer

2) Dense Layers: After the convolution layers, we have the fully connected network, composed of hidden dense layers (2 or 3) and a dense output layer. We flatten its inputs, since a dense network expects a 1D array of features for each instance. We also add two dropout layers, with a dropout rate of 50% each, to reduce overfitting. Similar to the previous model, We have used the ReLU activation function to add non-linearity and learn complex patterns.

3) Output: The output layer is a Dense layer with 4 units, corresponding to the bounding box for the object. We also employ 'EaryStopping' to terminate the training once the loss stabilizes. Finally, the 'mean squared error' is chosen as the performance metric to monitor during training.

The dataset is divided into training and validation sets with a 80/20 stratified split, allowing for model evaluation and tuning while maintaining the prior distribution. We perform hyperparameter tuning on the number of layers, the units in each layer, learning rate and dropout value.

III. Results

A. Dataset 1: Training and Validation Performance

From Figure 3, we can observe that the model shows a convergence which accounts for the leaarning ability. The loss curve is steep and does not distort, which indicates a good generalization. The best model has a validation accuracy of 0.819, training accuracy of 0.825 and hyper parameters as follows:

- Convolution Layers: 4
- Dense Layers: 2
- Dropout rate: 0.0
- Learning Rate: 0.0001

The learning curve is indicative of the small learning rate adopted by the model. We can also observe that the model has preferred the highest number of convolution layers and the lowest number of dense layers indicating complex feature maps.

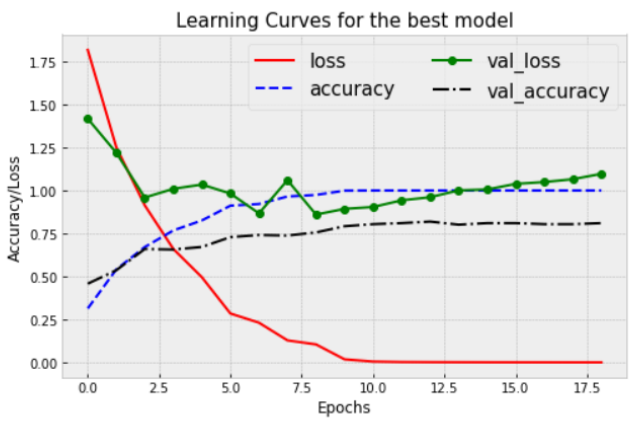

Fig. 3. Learning Curve: Best Model for Dataset 1

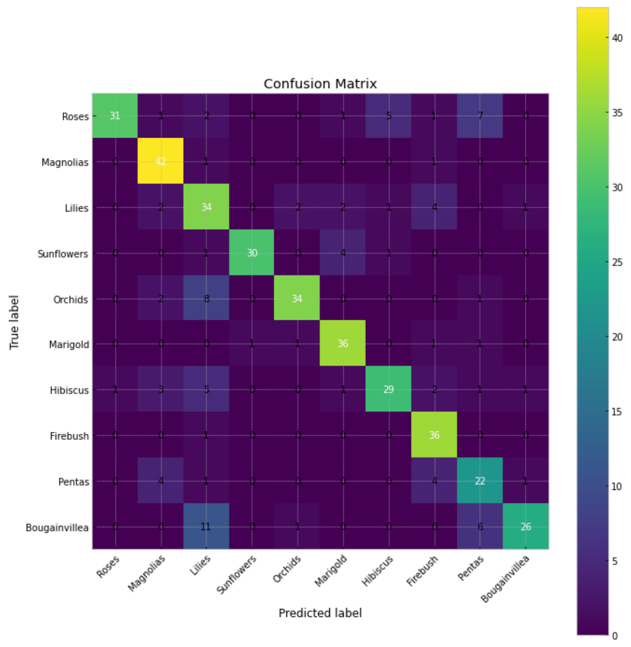

Fig. 4. Dataset 1 Classification Result: Confusion Matrix

Apart from this, the model used relu as the activation function. We have made attempts to experiment with tanh and add different number of units in layers as discussed in the previous section, however, the computation limitations have not favoured this.

B. Dataset 1: Classification Performance

The classification results using the CNN on test set are shown in Table II. It can be observed that there is only a slight difference between the accuracy of the classifier on the train (0.819) and test set with train being better. The F1-score, precision and recall indicate a balanced classification across various classes. Figure 3 shows the confusion matrix against the various classes. The significance of the major diagonal of the matrix indicates good model performance.

TABLE II
CLASSIFICATION QUANTITATIVE METRICS:

Accuracy	0.77
Precision	0.79
Recall	0.77
F1-Score	0.77

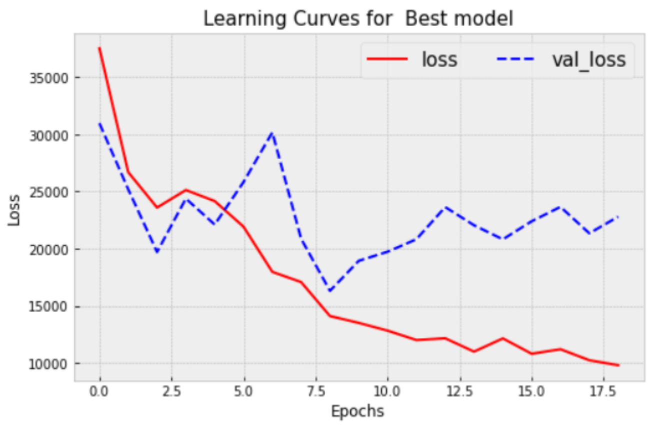

Fig. 5. Learning Curve: Best Model for Dataset 2

C. Dataset 2: Training Performance

From Figure 5, we can observe that the model shows a sporadic curve which preturbs in the middle phase of learning. This can be an indication of overfitting or underfitting. The validation loss is greater than the training loss in the end which indicates that the final model is not overfitting. The hyper parameters are as follows:

- Convolution Layers: 4
- Dense Layers: 2
- Dropout rate: 0.0
- Learning Rate: 0.001

Similar to the first model, we can observe that this model has preferred the highest number of convolution layers and the lowest number of dense layers indicating complex feature maps.

D. Dataset 2: Classification task

The test samples provided are not labeled. We have used a tool 'MakeSenseAI' to hand draw the bounding boxes for some of the targets and test the perfromance. Figure 2 shows the labeling canvas and the bounding box feature used in MakeSenseAI. However, to study and quantify the performance of the results we need to address the challenge of overlapping Region of Interest (ROI). Addressing overlapping Regions of Interest (ROIs) in classification often involves leveraging the Intersection over Union (IoU) metric. We have used IoU to quantifies the degree of overlap between the predicted bounding box and the ground truth bounding box (manually labelled). Thus, an IoU value of 1 is ideal. In our test set, we have obtained an average IoU value of 0.11. This

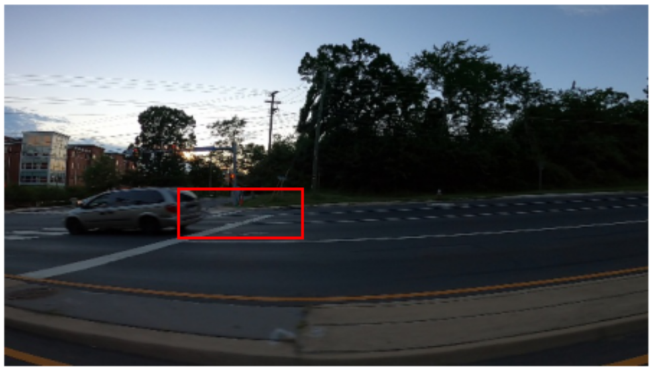

Fig. 6. Model 2: Object detection on test sample

is indicative of a not so remarkable performance. Figure 6 denotes a sample object detection on the test set made by the final model.

IV. Discussions

In this project, we have performed experiments using convolution neural network architectures on the given datasets. After the results and observations we can effectively answer the questions posed in the project.

- We can perform hyperparameter training using a grid searcg strategy to find the best model architecture. In the experiments, we have performed hyperparameter tuning on the convolution layers, dense layers, learning rate and dropout.
- We have displayed the learning curves for both the model training and observed that loss on object detection was very high since applying mse on bounding box will give higher values.
- We have used MakeSenseAI to manually label a few of the samples (test_bounding_boxes.csv) and test our model. In the case when no target labels are provded, we can use the Intersection over Union (IoU) metric to verify the detection performance against the ground truth bounding boxes. Therefore, we were able to perform tests on the given samples using this strategy and identify images with and without cars.
- Addressing the case where no car is present in the image, we can set the bounding boxes of such samples to [0,0,0,0]. If this smaple is predicted by the model with an IoU > 0, we can simply classify this result as a False Positive. This will enable us to predict the performance of the model on the test set quantitatively. Since we can use the misclassification parameter, there is a possibility to sample out the best threshold value from the precision recall curve for the Margin of Error (MoE). The solution is now simple, choose the model with the best balance.
- When the exact bounding box is not the target, we can evaluate performance in the test set using IoU metric discussed in the results.

V. Conclusion

The experiments performed with the various deep learning models has highlighted that cost, time and effectiveness (accuracy) are important factors when considering different architectures.