AIM Perception Project

Michael Fagundo

Data Exploration

To begin architecting models and training regimes, I first performed exploratory data analysis on the dataset provided. The EDA can be found in notebooks/eda.ipynb

Observations

- There are 10 classes
  - Most images are of four wheeled vehicles, followed by two wheeled vehicles and pedestrians.
  - The 10 classes are highly imbalanced, with two wheeled vehicles and pedestrians highly under-represented.
    - Should opt to use class weights for our multi-classification (CrossEntropy) loss.

Class	Count	~Percentage
Articulated truck	8,277	10
bicycle	1,827	2
bus	8,253	10
car	30,000	36
motorcycle	1,586	2
Non-motorized vehicle	1,401	2
pedestrian	5,010	6
Pickup truck	15,000	18

Single unit truck	4,096	5
Work van	7,743	10

- Images vary greatly in size, thus the median width (121) is roughly referenced for resizing inputs to 126 x 126

Models

- Backbone(s):
    - Pytorch's ResNet architectures (sizes 18, 34 and 50) with pre trained ImageNetV1 weights
- Classification head
    - Mutli-layer perceptron with two fully connected (fc) layers
        - First fc is size 100, with ReLU activation and batch norm
        - Second fc is size 10, with Softmax activation and dropout

Sweeps

I conducted a sweep in order to find an optimal combination of models and parameters. Data was reported to Weights and Biases (here). The sweep was conducted using Lambda's GPU servers, with an A100 gpu.

Swee Parameters:

- ResNet sizes: 18, 34, 50
- Dropout values: 0.0, 0.05

Data:

- Inputs:
    - Convert PIL images to tensors.
    - Resize to 126 x 126.
    - Normalized by the mean and standard deviations of the channels in the training set (Computed in notebooks/eda.ipynb).
- Targets:
    - Converted classes to one hot encoded tensors.
- Randomly split into train, validation and test splits with percentages 75%, 15% and 10% respectively.
    - Seeded random sampling with a constant for repeatable sampling.

Training Regime:

- Batch Size: 256
- Loss Function: Cross Entropy with class weights computed from training set
- Optimizer:
    - SGD

        - Initial learning rate: 0.1
        - Momentum: 0.9
        - Weight decay: 1e-5
    - Step learning rate scheduler
        - Gamma: 0.1
        - Step Size: 15 epochs
- Epochs:
    - 45 epochs total
    - Froze backbone (ResNet) for first 15 epochs, unfrozen for remainder.
- Reported training loss, validation loss and validation balanced accuracy every 100 batches

Results

WandB Link

Model	Dropout	Max Accuracy
ResNet50	0.05	0.898
ResNet34	0.05	0.896
ResNet50	0.0	0.895
ResNet34	0.0	0.892
ResNet18	0.05	0.888
ResNet18	0.0	0.886

Observations:

- Adding a dropout of 0.05 slightly improves validation across ResNet architectures.
- The larger the model, the better the performance.
- Models all perform within a similar ballpark, with behavior across size and dropout as expected.

Final Model

For the final model, I used the ResNet size 50 and dropout value 0.05.

The data and training regime follow that of the sweeps. However I apply stochastic weight averaging (SWA) for 10 epochs at a lower learning rate (1e-4) after the 45 epochs.

Results

Final results are computed by inferring on the test data split and are as follows:

Balanced Accuracy: 0.887
Classification Report:

	Precision	Recall	F-1 Score	Samples
articulated_truck	0.92	0.91	0.92	794
bicycle	0.84	0.89	0.87	182
bus	0.97	0.96	0.96	820
car	0.93	0.92	0.93	3025
motorcycle	0.84	0.95	0.89	172
non-motorized_vehicle	0.65	0.77	0.70	137
pedestrian	0.94	0.95	0.94	507
pickup_truck	0.89	0.86	0.99	1516
single_unit_truck	0.79	0.77	0.78	408
work_van	0.84	0.88	0.86	758
Weighted Average	0.90	0.90	0.90	8319