Explaining my approach , Model Architecture and Preprocessing steps taken:

Sunday, January 14, 2024 7:09 PM

Approach:

1. **Dataset Preparation:**
   - The training and test datasets are organized into separate directories for images and labels.
   - A validation set is created by splitting a portion of the training set using train_test_split.
   - Images and labels are moved to a 'validate' folder for validation purposes.
2. **YOLO-NAS Model Selection:**
   - YOLO-NAS (You Only Look One-level Neural Architecture Search) is chosen as the foundational object detection model.
   - The model comes in different variants (S, M, L) with varying mAP and latency, offering a trade-off between speed and accuracy.
3. **Pretrained Models:**
   - A pretrained YOLO-NAS model with COCO, Objects365, and Roboflow 100 datasets is chosen as a starting point.
   - Pretrained models are provided with different latency and mAP values.
4. **Model Initialization:**
   - The chosen model variant ('yolo_nas_l') is initialized using the Ultralytics package.
   - The model is loaded with pretrained weights from the COCO dataset to leverage its learned features.
5. **Transformations:**
   - Data augmentation is applied during training using transformations like DetectionRandomAffine.
   - The degrees parameter in the DetectionRandomAffine transformation is adjusted to 10.42 for improved robustness.
6. **Training Setup:**
   - The training process is configured using parameters such as learning rate, optimizer (Adam), and loss function (PPYoloELoss).
   - The Trainer class from super_gradients.training is utilized for training and checkpoint management.
7. **Training Loop:**
   - The model is trained for a specified number of epochs with a cosine learning rate schedule.
   - Model parameters and optimizer states are saved at the end of each epoch.
8. **Fine-Tuning with YOLO-NAS:**
   - The model is fine-tuned using YOLO-NAS, allowing it to adapt to the specific characteristics of the given dataset.
   - The training is performed using the Trainer instance.
9. **Best Model Selection:**
   - The best-performing model is selected based on the validation metric (mAP@0.50).

10. **Testing:**

    - The chosen model is tested on a separate test dataset using the test method of the Trainer instance.
    - Specific metrics (DetectionMetrics_050) are applied for object detection evaluation.

Model Architecture:

11. **YOLO-NAS Architecture:**

-
    - YOLO-NAS is designed with a quantization-friendly basic block, advanced training schemes, and post-training quantization.
    - It employs AutoNAC optimization, pre-training on COCO, Objects365, and Roboflow 100 datasets.

12. **Model Variants:**

    - YOLO-NAS comes in three variants: S, M, and L, offering different balances between mAP and latency.

Preprocessing Steps:

13. **Data Splitting:**

    - Training, validation, and test sets are created from the original dataset.

14. **Image and Label Organization:**

    - Images and corresponding label files are organized into separate folders for ease of access.

15. **Transformation Adjustments:**

    - Data augmentation transformations are applied, and specific parameters, like degrees in affine transformations, are adjusted.

16. **Checkpoint Management:**

    - A checkpoint directory is established to store the model's state during training.

17. **Model Initialization:**

    - The YOLO-NAS model is initialized with pretrained weights from the COCO dataset.

18. **Training Configuration:**

    - Training parameters, including learning rate schedule and optimizer settings, are configured.

19. **Fine-Tuning:**

    - The model is fine-tuned using YOLO-NAS, allowing it to adapt to the specific dataset characteristics.

20. **Testing and Evaluation:**

    - The final model is tested on a separate test dataset using specific metrics for object detection evaluation.

Summary:

- The approach involves selecting YOLO-NAS as the foundational model, initializing with pretrained COCO weights, fine-tuning for dataset-specific features, and evaluating the model's performance. Transformations and training configurations are adjusted to improve model robustness and effectiveness. The selected best model is then utilized for testing on a dedicated test dataset.
- This strategy aims to leverage the strengths of YOLO-NAS, taking advantage of its quantization-friendly architecture and pre-

training on diverse datasets to achieve robust and accurate object detection results for the specific task of detecting car dents and scratches.

From <https://chat.openai.com/c/5dfb3fb9-d1b2-4c48-82ef-ccfe3098883e>