Documentation of Faster RCNN

Objective:

The primary aim of this project is to develop and optimize an object detection model capable of accurately identifying and locating fractured body parts within medical imaging data, specifically using the FracAtlas dataset. This dataset, although encompassing various medical imaging data, is utilized in this context for its subset that includes images of fractured body parts along with corresponding bounding boxes and annotations that detail the extent and location of fractures.

To achieve this, the project employs a Faster R-CNN model, a well-regarded deep learning framework known for its efficacy in object detection tasks. Faster R-CNN has been chosen due to its integrated approach that combines feature extraction and region proposal networks to efficiently and accurately predict object locations and classifications in a single forward pass. This capability makes it particularly suited for the nuanced task of detecting fractured body parts, where precision in both localization and categorization is paramount.

Methodology:

Dataset preparation and Data pre-processing:

Data Selection:

- The focus is on the fractured subset of the FracAtlas dataset, which consists of medical images annotated with bounding boxes around fractured areas.
- These annotations serve as the ground truth for model training, teaching the model to recognize and localize fractures.

Data Pre-processing:

- Images are pre-processed to ensure they are in a suitable format for training.
- This includes resizing images to a uniform dimension and normalizing pixel values.

Model Configuration and Preparation:

Model Selection and Transfer Learning:

- The chosen model for this project is a pre-trained Faster R-CNN with a ResNet50 backbone.
- This model is well-regarded in the object detection field for its efficiency and accuracy.
- By employing transfer learning, we leverage the weights that the model has learned from being trained on a comprehensive and diverse dataset (such as ImageNet).

- This approach not only shortens the training time but also improves the model's capability to detect fractures by utilizing pre-learned feature representations.

Hyperparameter Optimization with Optuna

The optimization of hyperparameters is a crucial step to enhance model performance. We use Optuna for this process, focusing on three key parameters:

Learning Rate (lr):

- The learning rate is a critical hyperparameter that affects how much the model weights are updated during training.
- We use trial.suggest_loguniform('lr', 0.0001, 0.001) to search for the optimal learning rate within a specified range.
- The loguniform distribution is chosen because it helps in exploring a wide range of values, allowing for finer adjustments in lower ranges and broader searches in higher ranges, which is beneficial for finding a precise learning rate that avoids both slow convergence and overshooting the minimum of the loss function.

Weight Decay (weight_decay):

- This parameter helps in regularizing the model and preventing overfitting by penalizing large weights. trial.suggest_uniform('weight_decay', 0.0001, 0.001) is used to uniformly search between 0.0001 and 0.001 for the best weight decay value.
- A uniform distribution ensures an equal probability of selecting any value within this range, providing a balanced search for the optimal parameter that maintains model complexity while avoiding overfitting.

Number of Epochs (num_epochs):

- The number of epochs determines how many times the entire dataset is passed through the model during training. With trial.suggest_int('num_epochs', 2, 6), we aim to identify an optimal range that allows the model sufficient exposure to the data to learn effective representations without overfitting.
- This integer range ensures the model is trained for a minimum of 2 and a maximum of 6 epochs, allowing for flexibility in training duration based on the convergence rate observed during optimization trials.

Training Process and Evaluation

Optimizer Selection:

- For the training of the Faster R-CNN model, the Stochastic Gradient Descent (SGD) optimizer is utilized.
- SGD is chosen for its effectiveness in navigating the complex landscapes of high-dimensional weight spaces and its ability to find global minima for non-convex optimization problems, which are common in deep learning models.
- ADAM was not performing well for this object detection task.

Training Loop:

Batch Processing:

- Training is conducted in batches, where each batch consists of a subset of the dataset.
- This approach allows for efficient use of memory and can speed up the training process.

Forward Pass:

- For each batch, a forward pass is executed where the model computes the predicted bounding boxes and class scores for the input images.

Loss Computation:

- The loss is calculated by comparing the model's predictions against the ground truth annotations (bounding boxes and class labels).
- The Faster R-CNN model uses a combination of losses, including the classification loss (to differentiate between fractured and non-fractured images) and the regression loss (to accurately predict the bounding box coordinates of fractures).

Backward Pass and Weight Update:

- After calculating the loss, a backward pass is performed to compute the gradient of the loss function with respect to the model parameters.
- These gradients are then used by the SGD optimizer to update the model weights, aiming to minimize the loss in subsequent iterations.

Evaluation

Performance Metric:

- The model's performance is evaluated using the mean Intersection over Union (mean IoU) metric.
- Mean IoU measures the accuracy of the model's predicted bounding boxes in comparison to the ground truth, providing a quantitative assessment of how well the model is able to localize the fractures.

Validation Set:

- Periodically during training, the model is evaluated on a separate validation set to monitor its performance on unseen data.
- This helps in detecting overfitting early and in assessing the generalization ability of the model.

Hyperparameter Optimization Feedback:

- The results from the evaluation phase are fed back into the Optuna optimization process to guide the selection of hyperparameters in subsequent trials.
- This iterative process continues until the optimal combination of parameters is found, as evidenced by the highest mean IoU score on the validation set.

Analysis of the Results:

Highest Mean IoU:

Trial 37 shows the highest mean IoU of 0.247077727, achieved with a learning rate of 0.000988288, a weight decay of 0.000224612, and trained for 5 epochs.

This suggests that for this particular task, a relatively low learning rate combined with a modest weight decay and a moderate number of epochs leads to the best performance in terms of accurately detecting fractures.

Learning Rate and Performance:

Trials with lower learning rates (e.g., Trial 20 with lr = 0.000827888) tend to show better performance, indicating that for this dataset and model, smaller step sizes in the optimization process help in fine-tuning the model's weights more effectively.

Weight Decay Considerations:

Trials with very low weight decay values (e.g., Trial 20 with 0.000116482) have shown high mean IoU scores, suggesting that minimal regularization beyond the inherent structure of the model and data might be beneficial in this context.

However, the best-performing trial (Trial 37) uses a slightly higher weight decay, indicating that the optimal amount of regularization is task-dependent and requires empirical tuning.

Epochs and Learning:

The number of epochs varies across trials, but the highest performances are observed with 5 epochs (e.g., Trials 37 and 39), suggesting that this is sufficient for the model to learn effectively without overfitting.

The analysis of the hyperparameter optimization trials reveals that careful tuning of the learning rate, weight decay, and number of training epochs is crucial for maximizing the model's performance in detecting fractures within medical images. The optimal configuration found suggests a preference for lower learning rates and weight decays, combined with a moderate amount of training epochs, to achieve the best balance between learning ability and generalization.