**Fine Tuning**

Introduction:

Fine-tuning refers to the process of taking a pre-trained model, which has been trained on a large dataset and a general task and adapting it to a more specific task or domain. The pre-trained model serves as a starting point, providing a good initialization for the parameters, while the fine-tuning process adjusts those parameters to make the model more suitable for the target task.

The purpose of fine-tuning is to leverage the knowledge learned by the pre-trained model on a related task or domain and transfer it to the target task. This approach is particularly useful when the target task lacks sufficient labelled data to train a model from scratch or when the target task shares similarities with the pre-training task.

**Fine-tuning offers several advantages:**

1. Reduced Training Time: Fine-tuning enables faster convergence by initializing the model with pre-trained weights. This approach allows the model to benefit from the feature extraction capabilities of the pre-trained model, reducing the time required for training on the target task.

2. Improved Generalization: Pre-trained models have learned rich representations from large-scale datasets, which can be valuable for the target task. Fine-tuning helps the model adapt these representations to the specifics of the target task, leading to improved generalization and performance.

3. Effective Transfer Learning: Fine-tuning allows the transfer of knowledge from a related task or domain to the target task, even when the target task has limited data. This transfer of knowledge helps in leveraging existing models and their understanding of complex patterns.

**Fine-Tuning Process:**

* Data Collection and Preprocessing: Collecting and preprocessing data for the target task is a crucial step in fine-tuning. Ensure that the data is representative of the target task and is appropriately labelled or annotated. Preprocessing steps may include data cleaning, normalization, tokenization, and handling class imbalances.
* Model Selection: Selecting an appropriate pre-trained model is essential. Consider models that have been trained on a similar task or domain to the target task. Popular pre-trained models include BERT, GPT, and ResNet, among others. The choice of model depends on the nature of the target task and the available pre-trained models.
* Fine-Tuning Techniques: The fine-tuning process involves making the weights zero below a threshold of the pre-trained model, while allowing others to be fine-tuned. This decision depends on the similarity between the pre-training and target tasks.

**Catastrophic Forgetting:** Catastrophic forgetting refers to a situation where fine-tuning on a new task causes the model to lose knowledge of the pre-training task. To mitigate this, use techniques like gradual unfreezing, which involves unfreezing layers in a staged manner, or employ techniques like knowledge distillation to transfer knowledge from the pre-trained model to the fine-tuned model.

**Datasets**:

- MNIST: Handwritten digits dataset with 60,000 training samples and 10,000 test samples.

- Fashion MNIST: Clothing images dataset with 60,000 training samples and 10,000 test samples.

**Network Architecture**:

- Input layer: 784 neurons (flattened 28x28 images)

- Hidden layer 1: 256 neurons with ReLU activation

- Hidden layer 2: 128 neurons with ReLU activation

- Output layer: 10 neurons with softmax activation (10 classes)

**Threshold value:**

thresholds = [0.1, 0.2, 0.3, 0.4, 0.5]

**Output:**

The code scripts return the accuracy of the model on MNIST Fashion dataset after pruning and fine tuning the model.

['Threshold: 0.1, Accuracy: 0.8899000287055969',

'Threshold: 0.2, Accuracy: 0.8245999813079834',

'Threshold: 0.3, Accuracy: 0.2583000063896179',

'Threshold: 0.4, Accuracy: 0.09619999676942825',

'Threshold: 0.5, Accuracy: 0.10000000149011612']

Inferences:

It can be observed from the outputs of the model that with increase in the threshold,

the accuracy of the model decreases from 88.99% at threshold of 0.1 to 10% at threshold of 0.5 for MNIST Fashion dataset. Setting the threshold too high in magnitude-based pruning can decrease the accuracy of the model because it leads to aggressive weight pruning, removing a large number of weights from the network. When weights with magnitudes below the threshold are pruned and set to zero, it effectively removes the corresponding connections in the neural network.

By removing connections, the pruned model loses the associated information flow and the ability to capture certain patterns or features in the data. This loss of information can result in a decrease in model performance and accuracy. The pruned model may struggle to generalize well to new, unseen data, leading to lower accuracy during inference.

Additionally, pruning a significant number of weights can disrupt the balance of the model. Neural networks are trained to find the optimal set of weights that minimize the loss function and maximize accuracy. Aggressively pruning weights by setting a high threshold can undermine this optimization process, resulting in suboptimal performance.

It can be observed from the outputs of the model that with increase in the threshold,

the accuracy of the model decreases from 94.38% at threshold of 0.1 to 27.64% at threshold of 0.2 for MNIST dataset. Setting the threshold too high in magnitude-based pruning can decrease the accuracy of the model because it leads to aggressive weight pruning, removing a large number of weights from the network. When weights with magnitudes below the threshold are pruned and set to zero, it effectively removes the corresponding connections in the neural network.

By removing connections, the pruned model loses the associated information flow and the ability to capture certain patterns or features in the data. This loss of information can result in a decrease in model performance and accuracy. The pruned model may struggle to generalize well to new, unseen data, leading to lower accuracy during inference.

Additionally, pruning a significant number of weights can disrupt the balance of the model. Neural networks are trained to find the optimal set of weights that minimize the loss function and maximize accuracy. Aggressively pruning weights by setting a high threshold can undermine this optimization process, resulting in suboptimal performance.

Hence, it is important to strike a balance when selecting the threshold value in magnitude-based pruning. A careful consideration of the trade-off between pruning magnitude and accuracy is required to ensure that the model remains accurate while achieving the desired level of compression and efficiency. Fine-tuning techniques can be applied after pruning to recover some of the lost accuracy and further refine the pruned model.

Hence, it is important to strike a balance when selecting the threshold value in magnitude-based pruning. A careful consideration of the trade-off between pruning magnitude and accuracy is required to ensure that the model remains accurate while achieving the desired level of compression and efficiency. Fine-tuning techniques can be applied after pruning to recover some of the lost accuracy and further refine the pruned model.

Conclusion:

Fine-tuning is a powerful technique that allows leveraging pre-trained models to adapt them to specific tasks or domains. It offers benefits such as reduced training time, improved generalization, and effective transfer learning. By following the step-by-step process outlined in this documentation and adhering to best practices, you can successfully fine-tune models and achieve excellent performance on your target task. Remember to troubleshoot any issues that arise during the process, ensuring optimal results for your fine-tuned models.