Freezing Model Weight & FFN Layes in Chemprop v2

Pretraining in Chemprop v2 allows you to leverage a pre-existing model's weights and freeze them during the training process. This is particularly useful for transfer learning applications where you have a model pretrained on a large dataset and want to fine-tune it on a smaller, domain-specific dataset. Here's what you need to know about pretraining in Chemprop v2:

Freezing Model Weights: By specifying the --model-frzn <path> flag, you can load a pretrained model's checkpoint file, which will be used to overwrite and freeze the model weights during training.

Freezing FFN Layers: Additionally, you can specify the --frzn-ffn-layers <n> flag to control how many layers of the feed-forward network (FFN) are overwritten and frozen with the weights from the loaded checkpoint. By default, this value is set to 0, meaning no FFN layers are frozen.

----

!chemprop train --data-path /path/to/pretraining\_data.csv --task-type regression --output-dir pretrained\_model --model-frzn /path/to/pretrained\_checkpoint.pt --frzn-ffn-layers 3

- We're training a regression model (--task-type regression) on a dataset located at /path/to/pretraining\_data.csv.

- The trained model will be saved to the directory pretrained\_model.

- We're specifying a pretrained model's checkpoint file located at /path/to/pretrained\_checkpoint.pt using the --model-frzn flag.

- We want to freeze the weights of the first 3 layers of the FFN using the --frzn-ffn-layers 3 flag.

====

Freezing the weights of a neural network means preventing them from being updated during the training process. When you freeze the weights of certain layers or the entire model, their values remain fixed and do not change in response to the gradient updates computed during backpropagation.

Freezing weights is a common technique used in transfer learning, where a model pretrained on one task or dataset is fine-tuned on a different task or dataset. By freezing the weights of pretrained layers, you preserve the knowledge encoded in those layers while allowing the model to adapt its later layers to the new task or dataset.

In the context of Chemprop v2, freezing the weights using the --model-frzn and --frzn-ffn-layers flags allows you to initialize your model with pretrained weights and prevent certain layers, typically the earlier layers or the entire model, from being updated during training. This can help improve training stability, prevent catastrophic forgetting, and allow the model to converge faster, especially when the pretrained model has been trained on a large, diverse dataset.

===

If you specify only the --model-frzn flag without specifying --frzn-ffn-layers, all the weights of the loaded model will be frozen during training. This means that none of the parameters in the model, including both the message passing neural network (MPNN) and the feedforward neural network (FFN) layers, will be updated during the training process.

In this scenario, the entire pretrained model will be used as-is, without any fine-tuning of its parameters. This can be useful when you want to use a pretrained model as a feature extractor or when you have limited training data and you want to prevent overfitting by keeping the model parameters fixed.

However, if you want to fine-tune some of the FFN layers while keeping others frozen, you would specify the number of FFN layers to overwrite and freeze using the --frzn-ffn-layers flag along with the --model-frzn flag.

===

The --model-frzn and --frzn-ffn-layers commands are typically used when you load a pretrained model and want to fine-tune it on a new task or dataset.

Here's how you would typically use them:

Pretraining: You train a model on a large dataset or a related task to learn general features or representations. During pretraining, you don't need to freeze any weights because you want the model to learn from the data and update its parameters based on the loss function.

Fine-tuning: After pretraining, you may want to fine-tune the pretrained model on a specific task or dataset. In this case, you load the pretrained model using the --model-frzn flag to specify the path to the pretrained model's checkpoint file. Then, you might choose to freeze some layers of the model using the --frzn-ffn-layers flag if you want to keep certain parts of the model fixed while allowing other parts to be updated during fine-tuning.

So, to summarize, you use these commands during the fine-tuning stage, after you've pretrained your model and want to adapt it to a new task or dataset.

**Ensemble in Chemprop:**  
Training an ensemble of models can be beneficial in several scenarios:

Improved Generalization: Ensembling helps to reduce overfitting by combining predictions from multiple models trained on different subsets of the data or with different initializations. This often leads to improved generalization performance on unseen data.

Robustness to Variability: Ensembling can make the model more robust to variability in the data or model architecture. By aggregating predictions from multiple models, it can mitigate the impact of individual model biases or errors.

Uncertainty Estimation: Ensembling provides a way to estimate uncertainty in predictions. By examining the variance or agreement among the predictions of ensemble members, you can gauge the model's confidence in its predictions.

Handling Noisy Data: If your dataset is noisy or contains outliers, training an ensemble can help to smooth out the effects of individual noisy data points, leading to more reliable predictions.

For your task with a dataset of 40,000 molecules and 1224 target values, training an ensemble could be beneficial, especially if you have the computational resources to support it. It can help improve the robustness of your model and provide better generalization performance. However, you may need to experiment with different ensemble sizes to find the optimal balance between performance gains and computational cost.