Phison aiDAPTIV+ Getting Started Guide

The rapidly changing AI environment has created a number of different tools designed to work with AI and AI models. While these tasks often create a process that is similar across the industry the steps to get there can be vastly different.

The Phison aiDAPTIV+ Pro Suite software package is a feature rich tool for working with and fine tune training various models and that diversity can often create a fair amount of confusion when it comes to using these tools. This document is designed to help guide users to getting the most from aiDAPTIV+ Pro Suite.

Before you get started, it is strongly suggested that users consult the Install and User guides as these documents will help you get familiar with the Pro Suite interface and different features included with the software.

Ready? Let's get started

Inference

Pro Suite uses vLLM to inference and serve language models. This is an extremely high-performance inference engine and may require some configuration to work with your specific system configuration and LLM.

Setting the vLLM configuration.

Once a model has been enabled a popup will appear and ask the user to set the model configuration. For systems with more than one GPU you will have the option to select how many GPUs will be used and the size of the Max Token Length parameter (KV Cache).

Note: aiDAPTIV Cache does not offset memory usage during inference applications meaning that you will still need the correct amount of VRAM to load and use a model.

In situations where VRAM is limited you may need to adjust the KV Cache size to compensate. Every model has different requirements and some experimentation will be required to maximize the experience. If you get OOM (Out of Memory) errors, start by reducing the Max Token Length by 50% and try again. Repeat this process until the model loads.

***Tip**: When using FP16 models the required memory to load that model is roughly 20% more than double the parameter size.*

Multi GPU and Quantization

Large parameter models may not fit within the VRAM of a single GPU. For this there are two available options to explore. The best option will depend largely on the parameter size of the LLM and how much total VRAM is available.

Examples:

A 70B parameter model will require roughly 168GB of VRAM at FP16. By quantizing the model using the Pro Suite AWQ process it will be reduced to 4-bit precision (Q4) and lower the memory requirement to ~43GB

For the above example 4x RTX 6000 ADA cards would be required at FP16. This can be reduced to a single card at Q4

RAG

RAG (Retrieval Augmented Generation) can be a very powerful tool for adding new information to an existing LLM. While this information isn't directly added or trained with the model, it can provide valuable context to increase factual accuracy and will change how your LLM responds to your requests.

The chunk size and overlap determine how the source material is broken up before it is converted into a vector database. I like to think of chunk size as a sliding window that movies over the source material. Larger chunk sizes will provide a wider view, while smaller chunks will be more specific. Overlap is how many tokens are carried over from the previous chunk. The relationship between these values will determine how weights are applied during the vector conversion.

For RAG to be effective, the settings need to be tuned to match the source content and how you intend to use the information. For instance, if your documents contain considerable amounts of technical specifications, a smaller chunk size can be beneficial however, it may also fail to return relevant information when broad questions are being asked. Larger chunk sizes are beneficial when the source documents contain large paragraphs of text such as books, employee handbooks and FAQs where you may want to have in-depth responses.

Setting chunk size and overlap

Chunk sizes should be relevant to the natural breaks of the source material. 1,000 English words equates to around 750 tokens and is a good starting point. Overlap should be around 10-25% of the desired chunk size.

Chunk size: 1024
Overlap: 256

Improving RAG results

When using Pro Suite, it is rather easy to test and tune a RAG Collection.

1) Start with the suggested settings and create a collection
2) Start a new chat session, select your desired LLM, enable RAG and select your collection.

Create a series of questions to test the performance of your RAG Collection. I would suggest breaking the questions up into three categories

- **Specific**: This is where you are looking for small details in your documents
- **Broad**: These questions are intended to allow the LLM to interpret your data
- **Unrelated**: These questions should have nothing to do with the source material and would generally hallucinate the responses.

The purpose of these questions is to test the RAG Collection. In situations where the chunk sizes are too small, you may get accurate answers for your specific questions but, fail to provide any real details when it comes to anything else. Likewise, if the chunk sizes are too big you may get hallucinations when asking specific questions

To test and improve the results take the chunk size and divide it in half and then set the overlap accordingly. Rebuild the RAG and run your tests again. You may find that 512 may work well for some documents while 768 will work well for others. If results are getting worse then you will want to reverse the process.

Fine Tune Training

Setting batch size and total batch size

Batch size and Total batch size are hyperparameters used when fine tune training LLMs. Describe how many training examples you will use at one time. In the context of Pro Suite, the size of the batches is largely influenced by the available VRAM.

Setting the batch size to high will trigger OOM (Out Of Memory) errors and will cause the training to stop. Smaller batch sizes will extend the training time and may not take full advantage of the available resources.

For the best performance you will want to tune the Device Train Batch Size to maximize the available VRAM and then set the Update Total Batch Size using the following formula

Number of GPUs x Train Batch Size x 10

The total batch size can be the resulting value or smaller. Some testing will be required to find the optimal values.

Increasing the number of epochs

A common hyperparameter is Epoch which indicates a full pass through the training data. This can have a significant impact on the performance of your model. If there are too few epochs, then the model may not learn anything new. A high number of epochs could cause the model to become too specialized in the training data, resulting in poor quality responses on unseen data.

Choosing the number of epochs does require some testing to determine what works well for your particular model and the data you are training it with.

Pro Suite will allow you to select a total of five epochs for a particular training session, and for most fine-tuning jobs that will be more than enough. However, you can tweak how a model is trained without increasing the epoch number. This is done by altering the batch size.

When a batch has finished processing it will trigger a weights recalculation on the data that has been submitted. For instance, lowering the Device Train Batch Size by half will cause the LLM to recalculate the model weights twice as many times. This could make the training more effective and reduce the need for additional epochs.

Epochs and disk space

When fine tune training with aiDAPTIV, an epoch will create a copy of the LLM. From the Pro Suite interface, you will only have access to the latest epoch while the iterations needed to reach that level will remain unused on the file system.

If there are space concerns, these unused iterations can be removed

RAG + Fine Tune

Mix and Match FT model with RAG Collection

It is important to note that Fine Tune training does not add any “new” information to an LLM, and the total parameter count will remain the same. Internally the fine-tuning process will reconfigure the model weights to change how the model responds to certain input. This effectively does teach LLM but, only in the confines of what was already present.

A very powerful tool is the ability to train a model to speak and understand the linguistic nuances of a particular industry, trade craft, or person while providing current information through a RAG Collection.

This can be attained by applying a fine tune training dataset to a particular LLM and then matching that new model with a custom RAG Collection of specific customer data.

An alternative is to use the aiDAPTIV Guru tool. This tool has a variety of uses but works extremely well at creating hyper accurate chatbot models trained on customer data. This works on the principle that when an LLM is trained on customer data, it can reduce the number of hallucinations to near zero when supported by the RAG Collection used to train the LLM.

While this app does an amazing job it is important to know that it also requires tuning to match the source material with the hyperparameters. aiDAPTIV Guru uses many of the same hyperparameters as an RAG Collection and it is recommended to start with those settings. The application will then analyze the supplied data and determine how many Q/A pairs can be generated.

Note: While Fine Tune training can be done with any amount of source data, it works best when there are significant amounts of data available and when using that data in an RAG Collection no longer delivers satisfactory results.

Building custom training datasets

Fine tune training with Pro Suite is extremely easy. It can be done with properly formatted datasets downloaded from the web or by using the included aiDAPTIV Guru app. The important thing to remember is that these datasets are just JSON files and with a little bit of Python code a more powerful dataset can be created.

The purpose of building a custom training dataset is to further enhance what has already been generated or to preserve some variability in the fine-tuned model while also allowing it to learn from the new data.

Testing Trained Models

Inside Pro Suite there are two interfaces for inference tasks, Validation and Inference (Chat)

Validation allows the user to test various models side by side which can be helpful when evaluating different PreTrained models or when verifying the results of a fine-tuned model. This interface will submit the same prompt to different models using the same basic parameters, system prompt and RAG Collection. This allows you to easily compare the output.

Inference using a typical chatbot interface, allowing users to create multiple conversations with different configurations.

These can be powerful tools when evaluating RAG Collection Chunk Size and Overlap settings and are crucial when testing finished models.

Wrapping Up

Working with AI can be a fun and rewarding process. For certain AI tools the interaction is simple and straight forward while others require extensive testing and tweaks to get the most from them. Fine tune training of LLMs is a very sensitive process and it is important to recognize that there are no magic numbers, these will need to be evaluated against your training and RAG data and adjusted to maximize the effectiveness. This guide should help you get started and to better understand how to use the aiDAPTIV+ software.