

Continual Pretraining of Llama-3.2-1B with FPTAI Studio

I. Introduction

Large Language Models (LLMs) have transformed artificial intelligence by enabling machines to understand and generate human-like text. These models are initially pre-trained on vast datasets to grasp general language patterns. However, as new information emerges, like scientific discoveries or trending topics, models can become outdated. Continual pretraining addresses this by updating pretrained LLMs with new data, avoiding the need to start from scratch.

This blog post dives into continual pretraining, exploring its mechanics, challenges, and benefits. We'll also show how FPT AI Studio supports this process through a practical experiment. As continual pretraining demands significant compute resources and streamlined workflows, having the right platform is critical.

Built on the NVIDIA-powered FPT AI Factory, FPT AI Studio provides an unified platform with flexible GPU options, built-in security, and zero infrastructure setup. These capabilities make it easier and faster to run complex training workflows at scale.

By the end, you'll understand why continual pretraining is essential and how FPT AI Studio can help keep LLMs adaptable and relevant.

II. Continual Pretraining in LLMs

1. What Is Pretraining for LLMs?

Pretraining is the foundation of LLMs, where models are trained on massive, diverse datasets like web texts, books, or articles. This process helps them learn language structure and semantics. By predicting the next word, models leverage vast unlabeled data. The result is a versatile model ready for tasks like chatbots or content generation.

2. Pretraining Challenges

- **Computational Resources:** Training requires thousands of GPUs, consuming significant energy and funds.
- **Data Quality:** Datasets must be diverse and unbiased to avoid skewed outputs, which can raise ethical concerns.

- **Scalability:** Managing large datasets and models is complex, demanding efficient systems.
- **Obsolescence:** Pretrained models can quickly become outdated as new knowledge emerges.

3. From Pretraining to Continual Pretraining

Traditional pretraining is a one-time effort, but the world doesn't stand still. New trends, research, and language patterns emerge constantly. Continual pretraining updates the models incrementally, allowing them to adapt to new domains or information without losing existing knowledge. This approach saves resources compared to full retraining and keeps models relevant in dynamic fields like medicine or technology.

4. What Is Continual Pretraining?

Continual pretraining involves further training a pretrained LLM on new or domain-specific data to enhance its knowledge. Unlike fine-tuning, which targets specific tasks, continual pretraining broadens general capabilities. It uses incremental learning to integrate new data while preserving prior knowledge, often through techniques to balance retention and adaptation. For example, a model might be updated with recent news or scientific papers to stay current.

5. Continual Pretraining Challenges

- **Catastrophic Forgetting:** New training can overwrite old knowledge, reducing performance on previous tasks.
- **Data Selection:** Choosing high-quality, relevant data is critical to avoid noise or bias.
- **Model Stability:** Models must remain robust, necessitating careful monitoring.

6. Use Cases

Continual pretraining shines in various scenarios:

- **Domain Adaptation:** Continual pretraining allows these models to be further trained on domain-specific corpora, such as clinical notes, legal contracts, or financial reports, thereby enhancing their ability to understand and generate more accurate, relevant, and trustworthy content in those areas.
- **Knowledge Updates:** Language models trained on static datasets can quickly become outdated as new events unfold, technologies emerge, or scientific discoveries are made. Continual pretraining enables periodic or real-time integration of up-to-date information, keeping the model aligned with the latest developments. This is especially useful for any task where current knowledge is essential.

- **Multilingual Enhancement:** Many language models initially support only a limited set of widely spoken languages. Continual pretraining provides a pathway to extend these models with the low-resource languages, regional dialects, or even domain-specific jargon within a language. This ensures broader accessibility and inclusiveness, making the technology usable by a more diverse global population.

7. Why Not Just Fine-Tune?

Fine-tuning focuses on instruction-tuning the model across a series of downstream tasks that may differ in data distribution or change over time. This typically uses labeled datasets, such as question-answer pairs, to guide the model toward performing specific, well-defined tasks more effectively.

Fine-tuning adapts models for specific tasks but has limitations:

- **Task-Specificity:** It may not generalize to broad knowledge updates.
- **Overfitting Risk:** Models can overfit to small datasets, losing versatility.

III. Continual Pretraining on FPT AI Studio

With growing interest in Vietnamese LLMs, we conducted a real-world continual pretraining experiment using FPT AI Studio — a powerful no-code platform developed by FPT Smart Cloud. **FPT AI Studio** provides a streamlined platform for managing and executing LLM Training workflows, including the continual pretraining of LLMs. Its advantages include:

- A user-friendly graphical interface for pipeline creation and management.
- Integrated data management through Data Hub, allowing easy connection to S3 buckets to upload the large dataset.
- Simplified configuration of computing resources and hyperparameters.
- Address any difficult issues that commonly arise during LLM training with Model Fine-tuning.
- Store the trained model safely in Model Hub.
- Clear tracking and monitoring of training jobs.

In this blog, we will continue the training of [meta-llama/Llama-3.2-1B](#) with the aim of enhancing its performance in the Vietnamese language. The continual pretraining is carried out on **FPT AI Studio**.

1. Prepare the dataset

We continue pretraining on a Vietnamese dataset to enhance the language capabilities of the LLM. The Vietnamese datasets used include:

- [bkai-foundation-models/BKAINewsCorpus](#) - 5.2GB
- [vietgpt/wikipedia_vi](#) - 5.6GB

- [Uonlp/CulturaX](#) (Vietnamese subset) - 6GB
- [Ontocord/CulturaY](#) (Vietnamese subset) - 2.4GB
- [10,000 Vietnamese Books](#) - 1.7GB

This brings the total dataset size to **20.9GB**, with each sample saved in .txt format. We allocate **0.1%** of the data as the evaluation set, and the remaining **~99.9%** as the training set (**~2.8 billion tokens**).

Both the training and evaluation sets are saved in .jsonl files, following **FPT AI Studio's LLM training format**. To use them for training, upload the .jsonl files to your S3 bucket and connect it to **Data Hub**.

```

{"text": "Text 1"}
{"text": "Text 2"}
{"text": "Text 3"}
{"text": "Text 4"}
{"text": "Text 5"}

```

Figure 1. Example of the .jsonl file format required by FPT AI Studio's LLM training.

To connect your S3 bucket to **FPT AI Studio Data Hub**:

- **Step 1:** Go to the **Data Hub** tab.
- **Step 2:** Click **Create Connection**.
- **Step 3:** Fill in the S3 configuration details.
- **Step 4:** Click **Save**.

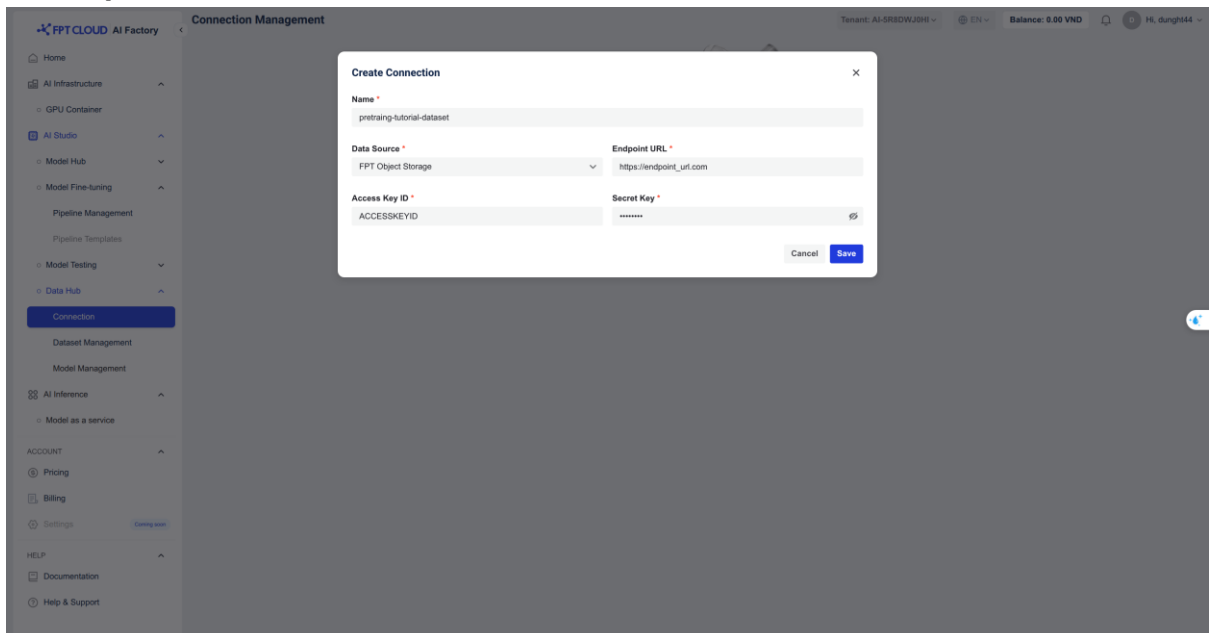


Figure 2. Create Connection dialog in Data Hub.

2. Start the training

We use **8 x GPU NVIDIA H100 SXM5 (128CPU - 1536GB RAM - 8xH100)** and the above prepared data for continual pretraining, with the following hyperparameters:

```

{
  "batch_size": 8,
  "checkpoint_steps": 1000,
  "checkpoint_strategy": "epoch",
  "disable_gradient_checkpointing": false,
  "distributed_backend": "ddp",
  "dpo_label_smoothing": 0,
  "epochs": 2,
  "eval_steps": 1000,
  "eval_strategy": "epoch",
  "finetuning_type": "full",
  "flash_attention_v2": false,
  "full_determinism": false,
  "gradient_accumulation_steps": 16,
  "learning_rate": 0.00004,
  "logging_steps": 10,
  "lora_alpha": 32,
  "lora_dropout": 0.05,
  "lora_rank": 16,
  "lr_scheduler_type": "linear",
  "lr_warmup_steps": 0,
  "max_grad_norm": 1,
  "max_sequence_length": 2048,
  "mixed_precision": "bf16",
  "number_of_checkpoints": 1,
  "optimizer": "adamw",
  "pref_beta": 0.1,
  "pref_ftx": 0,
  "pref_loss": "sigmoid",
  "quantization_bit": "none",
  "save_best_checkpoint": false,
  "seed": 1309,
  "simpo_gamma": 0.5,
  "target_modules": "all-linear",
  "weight_decay": 0,
  "zero_stage": 1
}

```

Setting up the Training Pipeline in FPT **AI Studio**:

- **Step 1: Create Pipeline:** In FPT AI Studio, navigate to **Pipeline Management** and click **Create Pipeline**.

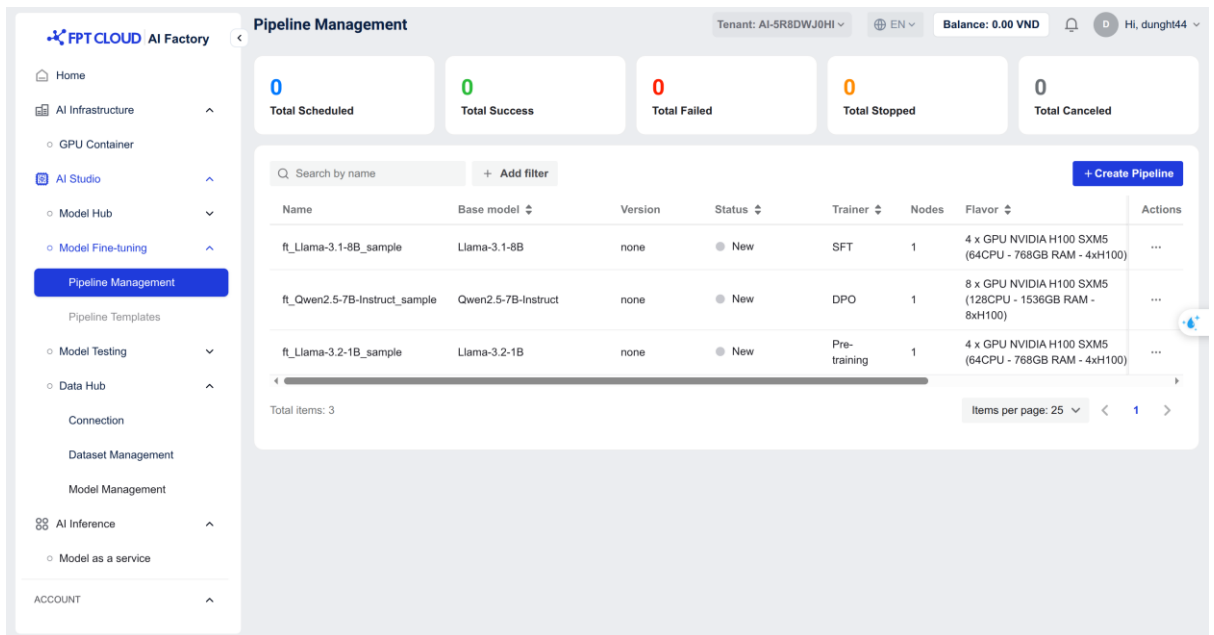


Figure 3. Pipeline Management interface in the Model Fine-tuning.

- **Step 2: Choose Template:** Select the **Blank** template and click **Let's Start**.

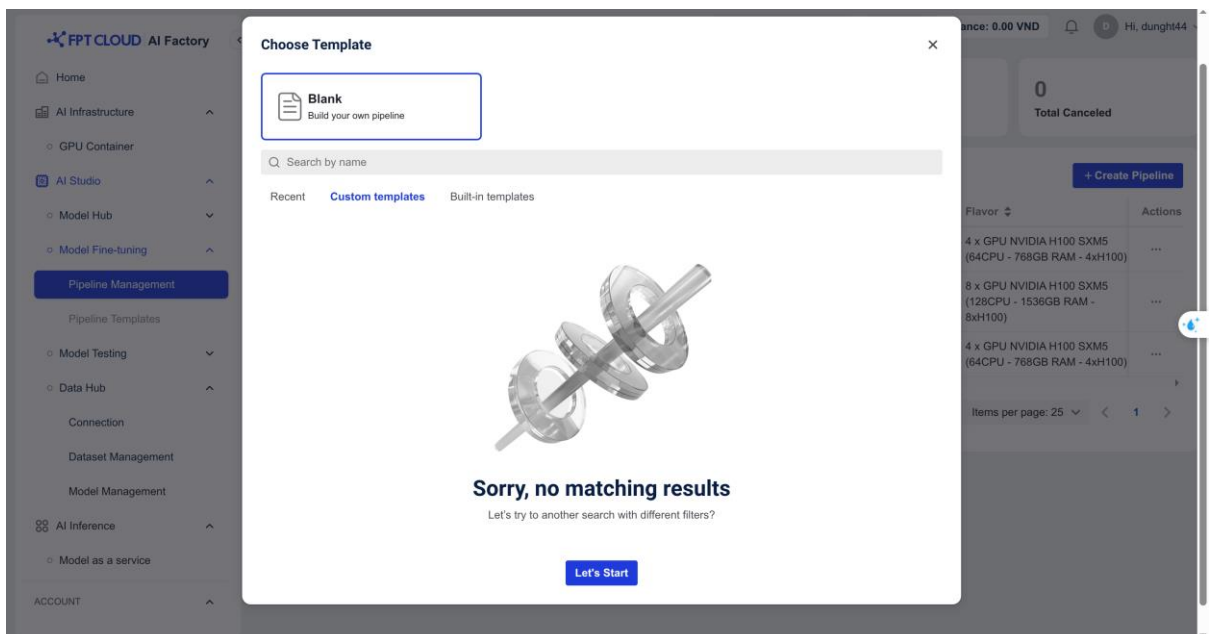


Figure 4. The "Choose Template" dialog within FPT AI Studio's pipeline creation process.

- **Step 3: Configure Base Model & Data:** Fill in the information about the **base model** and **dataset**, then click **Next Step**.

Create Pipeline



1 Base model & Data

2 Training Configuration

3 Others

4 Review

Base model source *

Catalog

Base model *

meta-llama/Llama-3.2-1B

Data format *

Corpus

Training data *

Connection - pretraining-tutorial-dataset

Path *

pretraining_tutorial_dataset/train_ver2_chunked_3_shuffled.jsonl

Evaluation data *

Connection - pretraining-tutorial-dataset

Path *

pretraining_tutorial_dataset/eval_ver2_chunked_3.jsonl

Next Step

Figure 5. "Base model & Data" of the "Create Pipeline".

- **Step 4: Configure Training Configuration:** Select **Pre-training** from the **Built-in Trainer** dropdown, toggle **Advanced**, and paste the provided JSON. For **Infrastructure**, choose **Single-node** with **8 x GPU NVIDIA H100 SXM5 (128CPU - 1536GB RAM)**. Click **Next Step**.

Create Pipeline



✓ Base model & Data

2 Training Configuration

3 Others

4 Review

Built-in trainer *
Pre-training

Hyperparameters * ? **Advanced** ☒

```
{
  "batch_size": 8,
  "checkpoint_steps": 1000,
  "checkpoint_strategy": "epoch",
  "disable_gradient_checkpointing": false,
  "distributed_backend": "ddp",
  "dpo_label_smoothing": 0,
  "epochs": 2,
  "eval_steps": 1000,
  "eval_strategy": "epoch",
  "finetuning_type": "full",
  "flash_attention_v2": false,
  "full_determinism": false,
  "gradient_accumulation_steps": 16,
  "learning_rate": 0.00004,
}
```

Infrastructure *
GPU Instance

Training mode
☒ Single-node ☐ Multi-node

Flavor *
8 x GPU NVIDIA H100 SXM5 (128CPU - 1536GB RAM - 8xH100)

Trigger *
☒ Manual ☐ Scheduled

Back

Next Step

Figure 6. "Training Configuration" of the "Create Pipeline".

- **Step 5: Configure Others:** No **test data** is required due to pretraining, check the **Send Email** option to receive notifications upon completion, and then click **Next Step**.

Create Pipeline

✓ Base model & Data

✓ Training Configuration

3 Others

4 Review

Test data

Upload file

File

Upload File

Download Sample

Only support json, zip files. Upload file limit 100 MB.

Dataset type *

Select

Number of checkpoint

1

Auto deployment to serving

Yes

No

Notification

Send Email

Back

Next Step

Figure 7. "Others" of the "Create Pipeline".

- **Step 6: Review and Submit:** Enter a **name** and a brief **description** for the pipeline run, carefully review **all configured settings** and click **Submit**.

Create Pipeline

✓ Base model & Data

✓ Training Configuration

✓ Others

4 Review

Name *
ft_Llama-3.2-1B_20250603104731

Description
0/100

Base model source: Catalog

Base model: Llama-3.2-1B

Data format: Corpus

Training data: Connection - pretraining-tutorial-dataset: pretraining_tutorial_dataset/train_ver2_chunked_3_shuffled.jsonl

Evaluation data: Connection - pretraining-tutorial-dataset: pretraining_tutorial_dataset/eval_ver2_chunked_3.jsonl

Built-in trainer: Pre-training

Hyperparameters:
Epochs: 2 - Batch size: 8 - Learning rate: 0.00004
Sequence length: 2048 - Checkpoint steps: 1000 - Gradient accumulation steps: 16

Training mode: Single-node
Nodes: 1
Flavor: 8 x GPU NVIDIA H100 SXM5 (128CPU - 1536GB RAM - 8xH100)

Trigger: Manual

Test data:

Dataset type:

Number of checkpoint: 1

Auto deployment to serving: No

Notification: dunght44@fpt.com

Back

Submit

Figure 8. "Review" of the "Create Pipeline".

- Step 7: Start training:
 - o ‘Start’ the training pipeline to begin the training process.

ft_Llama-3.2-1B_20250603104731	Llama-3.2-1B	none	New	Pre-training	1	8 x GPU NVIDIA H100 SXM5 (128CPU - 1536GB RAM - 8xH100)	Manual	2025-06-03 10:53:35	<div>Start</div> <div>Cancel</div> <div>Edit</div> <div>Delete</div>
ft_Qwen2-VL-7B_20250603100824	Qwen2-VL-7B	none	Failed	SFT	1	1 x GPU NVIDIA H100 SXM5 (16CPU - 192GB RAM - 1xH100)	Manual	2025-06-03 10:30:00	
ft_ft_DeepSeek-R1-Distill-Qwen...	ft_DeepSeek-R1-Distill-Qw...	none	Canceled	SFT	1	1 x GPU NVIDIA H100 SXM5 (16CPU - 192GB RAM - 1xH100)	Schedule	2025-06-03 10:08:04	
CPT_Llama-3.2-1B_pretraining_I...	Llama-3.2-1B	none	Success	Pre-training	1	8 x GPU NVIDIA H100 SXM5 (128CPU - 1536GB RAM - 8xH100)	Manual	2025-06-03 09:29:28	

Figure 9. Start the training pipeline in FPT AI Studio.

During training, you can track metrics such as **loss**, **eval loss**, and **learning rate** in the Model Metrics tab.

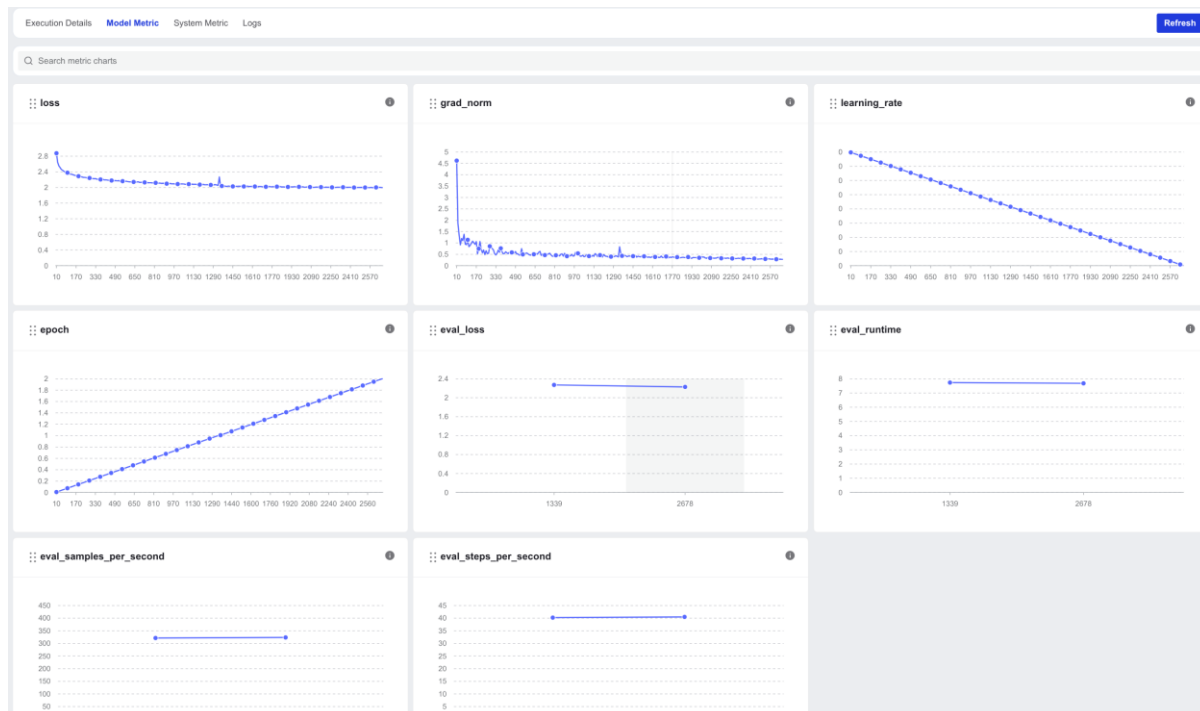


Figure 10. Training metrics.

After training, the continually pretrained model is saved under **Model Hub** → **Private Model**. We can download this model for personal use.

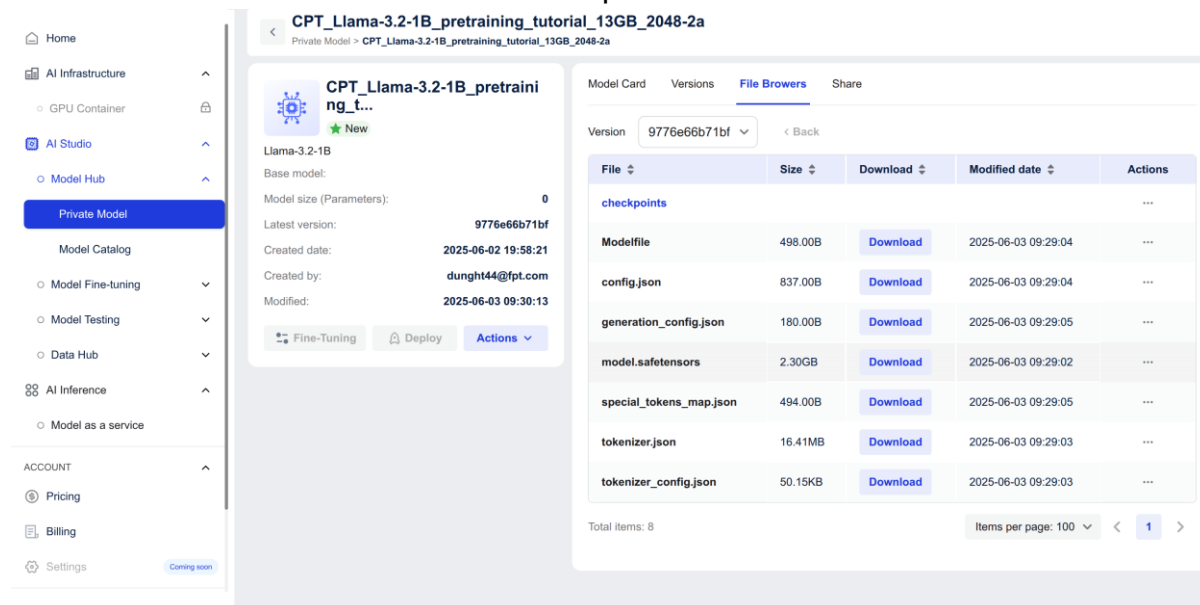


Figure 11. The continually pretrained model in Model Hub.

3. Results

After training, the loss decreased from **2.8746** to **1.9966**, and the evaluation loss dropped to **2.2282**, indicating that the model has effectively adapted to the Vietnamese language.

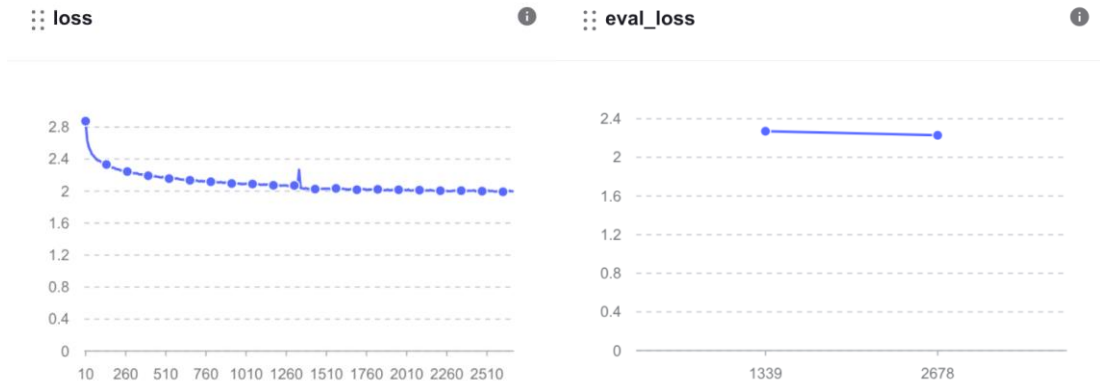


Figure 12. Loss and Eval Loss metrics during the continual pretraining process.

We evaluated the continually pretrained model on the Vietnamese benchmark [ViLLM-Eval](#) using [EleutherAI EvalHarness](#). The results were compared against the base model. Across all tasks, the metrics showed consistent improvements - some of them substantial. For instance, on the **lambada_vi** task, **accuracy** increased from 0.2397 to 0.3478, an improvement of nearly 11%.

Model	comprehension_vi	exams_vi	lambada_vi	wikipediaqa_vi
Baseline	0.6156	0.2912	0.2397	0.321
Continued Pretraining	0.6178	0.3318	0.3478	0.397

Table 1. Performance (Accuracy) comparison of the baseline Llama-3.2-1B model and the continually pretrained model on Vietnamese benchmark tasks from ViLLM-Eval.

In addition, we analyzed results on various subsets of **exams_vi**, covering subjects like **math, physics, biology, literature and more** in **Vietnamese**. The continually pretrained model demonstrated clear improvements over the baseline in every subject area.

Model	exams_vi_dia	exams_vi_hoa	exams_vi_su	exams_vi_sinh	exams_vi_toan	exams_vi_van	exams_vi_vatly
Baseline	0.3235	0.2522	0.2897	0.2819	0.2572	0.3192	0.2976
Continued Pretraining	0.3791	0.2609	0.3563	0.3113	0.2653	0.3662	0.3

Table 2. Detailed performance (accuracy) comparison on various subject area subsets of the exams_vi between the baseline and continually pretrained model.

These improvements demonstrate the feasibility of building high-performing Vietnamese LLMs with minimal overhead — opening the door for domain-specific applications in fintech, edtech, and more.

IV. Conclusion

As language and knowledge evolve at breakneck speed, Large Language Models must keep up—or risk becoming obsolete. Continual pretraining emerges as a vital solution, enabling models to seamlessly integrate new data while preserving previously learned knowledge. Unlike traditional pretraining or task-specific fine-tuning, this approach offers a scalable path to sustained performance across dynamic domains like healthcare, finance, education, and especially low-resource languages like Vietnamese.

Our experiment using FPT AI Studio demonstrated that continual pretraining is not only feasible but highly effective. By training Llama-3.2-1B on curated Vietnamese datasets, we achieved substantial performance gains across multiple benchmarks—proving that with the right tools, high-quality Vietnamese LLMs are within reach.

What sets FPT AI Studio apart is the seamless, end-to-end experience. From integrating datasets with Data Hub to orchestrating powerful GPUs and managing pipelines efficiently, FPT AI Studio removes complexity and helps your team focus on what matters most: improving your models and delivering impact faster. Whether you're developing a domain-specific chatbot, enhancing multilingual capabilities, or putting LLMs into production, FPT AI Studio provides the tools, infrastructure, and flexibility to help you build your own AI with confidence.

V. References

- C. Yıldız, N. K. Ravichandran, P. Punia, M. Bethge, B. Ermiş, Investigating Continual Pretraining in Large Language Models: Insights and Implications, ArXiv, 2024.
- L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black et al., A framework for few-shot language model evaluation, Zenodo, 2023.
- T.H. Nguyen, A.C. Le, V.C. Nguyen, ViLLM-Eval: A Comprehensive Evaluation Suite for Vietnamese Large Language Models, ArXiv, 2024.
- T. Nguyen, C. V. Nguyen, V. D. Lai, H. Man, N. T. Ngo et al., CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages, in Proc. of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING), Torino, Italy, pp. 4226–4237, 2024.
- T. Nguyen, H. Nguyen, T. Nguyen, CulturaY: A Large Cleaned Multilingual Dataset of 75 Languages, 2024.

- Q. D. Nguyen, H. S. Le, D. N. Nguyen, D. N. N. Nguyen, T. H. Le, V. S. Dinh, *Towards Comprehensive Vietnamese Retrieval-Augmented Generation and Large Language Models*, ArXiv, 2024.
- Lightning AI, Continued Pretraining with TinyLlama 1.1B, Lightning AI Studios, 2024. [Online]. Available: <https://lightning.ai/lightning-ai/studios/continued-pretraining-with-tinyllama-1-1b>
- Together AI, Continued Fine-tuning of LLMs: A Technical Deep Dive, Together AI Blog, 2025. [Online]. Available: <https://www.together.ai/blog/continued-fine-tuning>