# Fine-Tuning Large Language Models: Empowering Automated Script Generation from Process Mining Videos

## Abstract

In an era where large language models (LLMs) are transforming how we handle data and automation, fine-tuning emerges as a powerful technique to tailor these models for specific tasks. This white paper delves into the intricacies of fine-tuning LLMs, drawing from the latest best practices as of 2025. We explore why fine-tuning often outperforms alternatives like prompt engineering or retrieval-augmented generation (RAG), its key advantages, model selection, data preparation, the step-by-step process, scaling and hosting considerations, and potential challenges. Central to our discussion is a practical use case: leveraging a wealth of process mining videos to fine-tune an LLM that can autonomously extract and generate automated scripts. By the end, readers will have a roadmap to implement fine-tuning effectively, turning generic AI into a specialized tool for business process automation.

## Introduction

As someone deeply involved in process mining and automation, I've seen firsthand how raw data from videos can hold untapped potential for creating efficient workflows. Process mining involves analyzing event logs to discover, monitor, and improve business processes, often captured in instructional videos. However, manually extracting scripts from these videos is time-consuming and error-prone. Enter large language models—pre-trained on vast datasets, they excel at natural language understanding and generation. But to make them truly shine in niche domains like generating automated scripts from video content, fine-tuning is essential.

Fine-tuning involves taking a pre-trained LLM and further training it on a domain-specific dataset to adapt its parameters. This isn't just tweaking; it's refining the model's weights to align with particular tasks, leading to more accurate and context-aware outputs. With advancements in 2025, tools like Hugging Face and efficient methods such as LoRA make this accessible even for teams without massive compute resources. This paper aims to guide you through the process, grounded in real-world application.

## Why Fine-Tune LLMs Over Other Options

When customizing LLMs, options abound: zero-shot or few-shot prompting, where you craft detailed instructions to guide the model without training; RAG, which augments prompts with retrieved external knowledge; or even building agents that chain multiple LLM calls. Yet, for tasks requiring deep domain expertise—like generating precise automated scripts from process mining videos—fine-tuning stands out as the superior choice. Prompting alone often falls short in consistency, especially with complex, video-derived data where nuances in process flows must be captured accurately; it can lead to hallucinations or generic responses that don't align with specific scripting languages or tools used in process mining. RAG helps by pulling in relevant context, but it doesn't internalize that knowledge into the model, resulting in higher latency during inference and potential retrieval errors if the knowledge base isn't perfectly curated. Fine-tuning, by contrast, embeds the specialized knowledge directly into the model's parameters, enabling faster, more reliable performance tailored to the task. It's particularly ideal when you have a substantial dataset, like our collection of process mining videos, allowing the model to learn patterns autonomously rather than relying on brittle prompt engineering that needs constant refinement as the model updates or data evolves.

<argument name="citation\_id">0</argument>

<argument name="citation\_id">10</argument>

<argument name="citation\_id">11</argument>

## Advantages of Fine-Tuning

Fine-tuning offers a host of benefits that make it a game-changer for specialized applications. First, it boosts performance metrics like accuracy and relevance—studies show fine-tuned models can achieve 20-50% better results on domain-specific tasks compared to base models.

<argument name="citation\_id">8</argument>

For our use case, this means an LLM that not only extracts key steps from videos but generates executable scripts in languages like BPMN or Python automation frameworks. Second, it's resource-efficient in the long run: after the initial training, inference is quicker and cheaper than RAG setups that query databases repeatedly. Third, it enhances customization, allowing the model to adopt industry-specific jargon, reduce biases in process interpretations, and handle multimodal inputs if extended to video transcripts. Additionally, fine-tuned models are more controllable, minimizing unwanted outputs, and can be iterated upon with techniques like reinforcement learning from human feedback (RLHF) for even finer alignment. In 2025, with parameter-efficient methods, fine-tuning democratizes AI for smaller teams, avoiding the need for from-scratch training.

<argument name="citation\_id">6</argument>

<argument name="citation\_id">13</argument>

## Selecting the Right LLM

Choosing a base model is crucial—opt for open-source ones to avoid vendor lock-in and ensure transparency. Popular choices in 2025 include Meta's Llama 3 series (e.g., Llama 3.1 8B for balance of size and performance), Mistral's models like Mistral 7B for efficiency, or Google's Gemma 2 for strong reasoning capabilities. For script generation from videos, prioritize models with strong multimodal extensions or those pre-trained on code and text, like CodeLlama. Hugging Face's model hub is a goldmine; evaluate based on size (smaller for faster fine-tuning), pre-training data relevance, and community support. If handling video transcripts, consider models compatible with vision-language extensions, though for pure text-based script gen, a text-only LLM suffices after transcription.

<argument name="citation\_id">2</argument>

<argument name="citation\_id">3</argument>

<argument name="citation\_id">14</argument>

## Preparing Training Data

Data is the backbone of fine-tuning. Start by curating a high-quality dataset: for our process mining videos, transcribe audio using tools like Whisper, then annotate transcripts with corresponding automated scripts. Format as input-output pairs, e.g., input: "Video transcript describing order fulfillment process"; output: "Script in Python: def fulfill\_order()...". Aim for 1,000-10,000 examples initially, ensuring diversity in process types (e.g., procurement, logistics). Clean data by removing noise, balancing classes, and augmenting with synthetic examples via base LLMs. Use formats like JSONL for efficiency, and split into train/validation/test sets (80/10/10). For video-specific nuances, include timestamps or keyframe descriptions. Best practices emphasize labeled, diverse data to prevent overfitting—tools like SuperAnnotate can aid annotation.

<argument name="citation\_id">0</argument>

<argument name="citation\_id">5</argument>

<argument name="citation\_id">20</argument>

## The Fine-Tuning Process

Fine-tuning follows a structured workflow. First, load your base model and tokenizer via Hugging Face Transformers. Preprocess data by tokenizing and padding. Use parameter-efficient fine-tuning (PEFT) like LoRA or QLoRA to train only adapters, reducing memory needs—ideal for 7B+ models on consumer GPUs. Set up a Trainer with hyperparameters: learning rate 1e-4, batch size 8-16, epochs 3-5. Incorporate evaluation metrics like BLEU for script generation or custom accuracy on process steps. Train using frameworks like Axolotl or Unsloth for speed boosts up to 2x. Post-training, quantize the model (e.g., to 4-bit) for deployment. For our use case, fine-tune on transcript-script pairs to teach the model to infer scripts autonomously.

<argument name="citation\_id">2</argument>

<argument name="citation\_id">3</argument>

<argument name="citation\_id">7</argument>

## Scaling and Hosting Fine-Tuned Models

Scaling involves distributed training: use cloud platforms like AWS SageMaker or Google Vertex AI for multi-GPU setups, leveraging features like mixed precision and gradient checkpointing. For large datasets, employ data parallelism or model sharding. Hosting options include Hugging Face Spaces for quick demos, or production inference on endpoints like AWS Bedrock or Azure ML, with autoscaling for traffic. Monitor costs—fine-tuning a 7B model might run $100-500 on cloud, but inference is pennies per query. Optimize with tools like vLLM for faster serving.

<argument name="citation\_id">4</argument>

<argument name="citation\_id">12</argument>

## Challenges and Best Practices

Challenges include catastrophic forgetting (mitigated by PEFT), data privacy (use synthetic data), and high compute costs (start small). Best practices: monitor for overfitting with early stopping, evaluate on held-out data, and iterate with RLHF. Ethically, ensure bias mitigation in process scripts and comply with data regulations. Future trends point to federated fine-tuning for privacy-preserving adaptations.

<argument name="citation\_id">7</argument>

<argument name="citation\_id">9</argument>

## Use Case: Generating Scripts from Process Mining Videos

Imagine a repository of hundreds of process mining videos demonstrating workflows like supply chain optimization. We transcribe them, pair with hand-crafted scripts, and fine-tune a model like Mistral 7B. The result? An LLM that, given a new video transcript, generates ready-to-run automation scripts—streamlining operations and reducing manual effort by 70%. This evolves to full autonomy, where the model handles variations in processes without prompts.

## Conclusion

Fine-tuning LLMs isn't just a technical step; it's a strategic enabler for innovation in fields like process mining. By following this guide, you can transform generic models into powerful, script-generating allies. As we continue to amass video data, fine-tuned LLMs will drive efficiency, proving that targeted adaptation is key to AI's full potential.

## References

- [0] SuperAnnotate Blog on LLM Fine-Tuning

- [2] Philschmid on Fine-Tuning LLMs in 2025

- [3] Modal Blog on Best Frameworks

- [4] Heavybit on LLM Fine-Tuning for Teams

- [5] SCAND on Fine-Tuning LLMs

- [6] Lakera AI Guide

- [7] Sam-Solutions on Fine-Tuning Architecture

- [8] Towards AI on Fine-Tuning as Secret Sauce

- [9] Machine Learning Made Simple on Fine-Tuning Waste

- [10] AIMultiple on Enterprise Fine-Tuning

- [11] EclipseSource on Why Not Fine-Tune

- [12] YouTube on Fine-Tuning Tools Comparison

- [13] UBIAI on LLM Methods

- [14] DataCamp Tutorial

- [20] School-Brainhack on Vid-LLM Fine-Tuning