Experimenting with GPT OSS 20B and LoRA Fine-tuning to build Cooking Recipes GPT

University of Turku
Faculty of Technology
Master of Science Thesis
February 2026
Tony Pohto

UNIVERSITY OF TURKU
Faculty of Technology

Tony Pohto: Experimenting with GPT OSS 20B and LoRA Fine-tuning to build Cooking Recipes GPT

Master of Science Thesis, 42 p.

February 2026

The evolution of Large Language Models has been fast. Even some recent smaller and open-weight models have proved to be capable of various tasks. One of them is GPT OSS 20B which turns out to be capable of producing versatile, quality cooking recipes out-of-the box.
There exists also specialized cooking and recipes related datasets which could be used with these LLMs to fine-tune them. But experiments suggest that such fine-tuning is often not needed. GPT OSS 20B has gone through already extensive post-training optimization and tuning.
Yet, the fine-tuning techniques have also been developing and become more accessible to more developers and institutions. LoRA as technique, is very useful and lightweight and still worth to try.
In this thesis, LoRA is used to fine tune GPT-OSS 20B with cooking related conversational dataset. Another dataset consisting of recipes, is used to evaluate the understanding of the LLM of the cooking domain.
The results show that GPT OSS 20B didn't really benefit from such light fine-tuning but instead shines already on it's own. One restriction was hardware which lead to using small amount of data, only affecting the style of the LLM. Using larger, high quality and versatile datasets is one thing which could be tested and studied in future research.

Keywords: LLM, GPT, Transformer, Fine-Tuning, Transfer Learning, LoRA, PEFT, RAG, Prompt Engineering, GPT OSS, CoT, MoE

Contents

List of Figures

List of Tables

1 Introduction

During the rabid progress made in Natural Language Processing (NLP) related to Large Language Models (LLMs), some have been wondering the possibilities of individuals and small business or academic entities to build and use their own smaller local LLMs, which would be hosted on their local resources: How qualified these smaller LLM instances could be in comparison with large commercial products, for example GPT-5 which OpenAI launched in August 2025. These larger commercial products continue to enhance and their abilities keep advancing. They could be used in a variety of domains.

But what if one can determine just one domain that the LLM has to specialize in. Does this make the task of training such a local model easier? In this thesis such a one-domain model is trained and it's outputs are compared before and after fine-tuning. The chosen domain will be common in everyday life: Cooking recipes.

The research questions are the following:

1. Does LoRA fine-tuning improve domain-specific performance of GPT-OSS 20B in cooking recipe generation?

and

2. Does LoRA fine-tuning improve structured food-state reasoning performance?

The hypotheses are the following: For the first research question:

2.1. H1: Domain fine-tuning improves recipe quality and versatility in human evaluation.

or

2.2. H0: There is no meaningful improvement, when comparing and evaluating by human observation

and for the second:

2.1. H1: Domain fine-tuning improves semantic alignment with cooking-specific ground truth.

or

2.2. H0: There is no statistically significant improvement.

In this thesis, experimenting is performed with a relatively small open-weights base model by fine-tuning it with cooking techniques conversations dataset. It's outputs are also compared before and after the fine-tuning process. The goal is to investigate whether it is worth the effort to make this example of domain specific, localized GPT and whether it can produce useful outputs in the cooking recipes -domain.

CSCs, also known as Tieteen tietotekniikan keskus oy, resources for computing were used to complete the training and test the standard base and fine-tuned models.

2 Earlier Works

Large Language Models' architectures and solutions can vary depending on the domain which problems they are planned to focus on. For example, one common area to measure cognitive capabilities in humans, which is mathematics, provides a versatile landscape of various challenging problems for LLMs to try to grasp on. In order to face these challenges LLM developers need to explore specialized datasets and advanced tools to enhance their LLMs performance in math problems. These tools could include advanced prompting, external tools usage or careful fine-tuning of the LLM with Math specific datasets. [1]

Physics-related reasoning capabilities have also been recently tested on LLMs. One study pursued to enhance LLM physics reasoning capabilities by presenting a novel technique using reinforcement learning and reward techniques. [2]

LLMs have also provided substantial benefits for producing programming language code. Studies suggest that they are highly qualified in this specific task, in fact, a 2024 survey reported that 88 % of developers benefit from code-assisting LLMs in their profession. [3] The development to offer better tools for programmers is an ongoing field in science as well. [4] One new study experimented with LLM Agents and pursued to reduce inconsistency, ambiguity and incompleteness of code completions written by LLMs. [5]

2.1 Research in Cooking Recipes Domain

Earlier study published in 2020 presents a GPT-2 model fine-tuned with dataset called Recipe1M+[6]. Authors presented two modes for generation: one which provides a recipe for given ingredients and another one which gives ingredients based on recipe instructions and title. They have published this model on their website, but the website is not online anymore in 2025. Authors also implemented evaluation logic to evaluate the quality of the output texts: Calculating and highlighting the overlapped ingredients between the input and the output and comparison with reference recipes using ElasticSearch.[7]

One recent study published in 2024, LLaVA-Chef: A Multi-modal Generative Model for Food Recipes, utilizes one common element found in most online food recipes: the picture of the final dish. They embedded the data from the pictures alongside textual information of the recipe data and pursued to enhance prompting diversity and evaluation metrics compared to earlier recipe-generation models, and thereafter produce more fluent and versatile recipe texts. They experimented with various base models and decided to progress to fine-tune LLaVa-based model.[8] LLaVA is a multimodal model which focuses on language-image instruction-following data. [9]

Retrieval Augmented Generation (RAG) techniques have also recently been used when generating recipes. The objective of using RAG was to diminish the amount of hallucinations produced by the LLMs generating the recipes. [10]

One challenge is the evaluation of the models outputs. It may be excluded to cook all the recipes and taste them. Instead, one has to find more feasible and quantitative evaluation techniques to estimate the models performance and output quality. One interesting takeaway to this was in 2024 research paper producing a small evaluation dataset based on pizza cooking instructions, breaking the process of baking pizza to smaller steps. The paper aimed to provide a dataset but also a

framework to evaluate the meaningfulness of all sorts of procedural task texts. [11]

3 Fine-Tuning and Other Techniques

3.1 Transfer Learning

Transfer learning is a technique in Machine learning where large, multilayered neural networks abilities are used as a base. Eg. language models are capable in lots of textual tasks already, depending on the type and size of the architecture. In transfer learning most of the layers are frozen so that their weights stay unaltered and a couple of new layers are introduced just before the output. These new layers are then fine-tuned with small datasets to train their weights. This technique takes advantage of the large base model network but also introduces new abilities to the model, which are the result of the fine tuning of the last few, new layers. Hyperparameter tuning is also crucial at this point, and these parameters are not determined from the data, they are set to some values by human. [12]

When it comes to this thesis' objective, one would need a sufficiently diverse food recipe dataset to cover the fine-tuning needs. But the dataset doesn't have to be extensively large. On GPT-OSS' fine-tuning manual they only used 1000 row dataset to fine tune its thinking abilities to work on multiple different languages. [13] So a small, but diverse dataset would suffice. GPT-OSS can already provide cooking recipes, but potentially we could enhance it's food and recipe understanding by fine-tuning it and this is the one thing we are going to experiment with.

It is worth to mention that GPT OSS is a 20 billion parameter model. So a

dataset of 1000 examples would only affect the style of the output. In order to learn new domains or acquire new skills, dataset size has to be bigger.

3.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a type of extension to the LLM. When using RAG, LLM workflow consists of document search phase which will fetch the relevant information before the LLM request and form therefore a custom prompt for the LLM to format its output to. In essential, a preface of document scanning and prompt augmenting is introduced when compared to non-RAG way to prompt LLMs. [14]

3.3 Prompt Engineering

When practicing prompt engineering, LLM users and developers try to pick the best words, sentences and structure for their input given to the LLM. Prompt engineering is an art of it's own and continues to evolve when new GPTs are rolled out to the public.

By using prompt engineering LLM could be told to act in different expertise roles and accomplish different types of textual tasks. Prompt engineering includes popular Chain-of-Thought (CoT) technique, which tries to break down the question to smaller pieces, chains of thought-process and therefore accomplish better, more logical outputs by the LLM.

The most important best practices of prompt engineering are unambiguity of prompts, sufficient context provided in the prompt, balancing between simplicity and complexity to avoid low quality answers and refinement by experimenting different prompts. [15]

3.4 Synthetic LLM data

Internet data sources have already been crawled through quite thoroughly. This is a problem as larger and more complex models require more and more textual data. One way to tackle this problem is synthetic data. Synthetic data means machine-generated data produced usually by an LLM. Synthetic data is a way to batch and complete the gaps in natural, original data. When using synthetic data there is also the risk of saturating the dataset with low quality, dull text, which ultimately leads to worse performance of the language model, or any model in their own data genres. [16]

There is also a phenomenon called Dead internet theory. The effects of this phenomenon are already visible for example when scrolling through social media feeds. There is vast amounts of AI produced media in various internet channels. This applies to video, audio and pictures but also texts. [17]

3.5 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning, also known as PEFT, is a collection of techniques aiming to reduce to memory and computational footprint of fine-tuning a large language model. These methods avoid the large costs and risks of full fine-tuning of an LLM, which is almost always extremely time- and memory-consuming and a computationally heavy task.

PEFT techniques present a large variety of different methods and approaches to address the problems of fine-tuning in an efficient manner. These can be categorized into several method groups. One categorization includes the following PEFT approaches [18]:

- Additive Fine-Tuning
- Selective Fine-Tuning

- Reparameterized PEFT
- Hybrid PEFT
- MoE-Based PEFT

Next, each one of them is examined in more detail.

3.5.1 Additive Fine-Tuning

Additive Fine-Tuning introduces additional parameters to be augmented into the model. Additive Fine-Tuning process adjusts only these new, fresh parameters in the whole neural network model. These can be for example small adapter layers inside the Transformer blocks of the LLM neural network.

Instead of this adapter layer technique, one can also use Soft Prompt PEFT, which introduces adjustable vectors called soft prompts appended to the start of input sequences.

There is also an additive technique called Scaling PEFT which involves so called Propulsion parameters, which scale dimensions of the pretrained model without modifying the models parameters. The term Propulsion comes from physics where it means a driving or propelling force. [19]

3.5.2 Selective Fine-Tuning

In Selective Fine-Tuning only a subset of some parameters of the model are fine-tuned. The objective is to tune only those parameters which are the most relevant in the given task or domain. The one important challenge here is to identify the parameters which are most important in the given task. Numerous methods for evaluating parameter importance has been developed. These include for example FishMask, Adafish, IST and U-diff Pruning.

3.5.3 Reparameterized PEFT

When using Reparameterized PEFT techniques, one utilizes the features of matrix transformations by introducing sparse low-rank decompositions and adding them on "top" of the original, frozen, neural network parameters, using matrix addition.

One of the most important one of these is called LoRA, and we will experiment with LoRA on this master's thesis later.

LoRA uses a fixed rank parameter, but other methods include dynamic and adaptive rank methods. There is also numerous LoRA variants in addition to vanilla version.

3.5.4 Hybrid PEFT

As the name states, one could fuse together numerous approaches in PEFT and combine different fine-tuning methods into one single hybrid approach.

3.5.5 MoE-Based PEFT

Mixture-of-Experts mechanisms could also be incorporated into PEFT methods. While doing so, one applies expert routing mechanisms into selecting and combining low-rank modules. [20]

4 LoRA

Low-Rank Adaptation, also known as LoRA, was first introduced in 2021 paper "LoRA: Low-Rank Adaptation of Large Language Models" by Edward Hu et al. They came up with a solution for fine-tuning by using less computational power and memory than the traditional full fine-tuning. What is interesting, is that their solution ended up also performing better and not increasing inference time. This solution, LoRA, eased the task to fine-tune large language models into for example domain-specific tasks.

When using LoRA, an external neural network is constructed and trained as an additive network to the existing large, eg. an LLM, neural network architecture. The extra network is decently smaller than the main neural network, but using matrix multiplication one can sum their weight values with each other. [21]

The benefits of using this LoRA technique are memory efficiency and savings in computational time. The original LoRA paper states that GPU memory requirements could be reduced by 3 times compared to full fine-tuning. They also observed a 25 percent speedup during training compared to full fine-tuning. [22]

In 2026 LoRA is still useful and valuable. As a fine-tuning strategy it only doesn't just offer smaller computational costs, smaller memory constraints than many other fine-tuning strategies, but also a smaller risk of catastrophical forgetting, which means that LLM loses some essential information acquired earlier in the training process. It also offers faster training times and minimal inference cost, meaning use

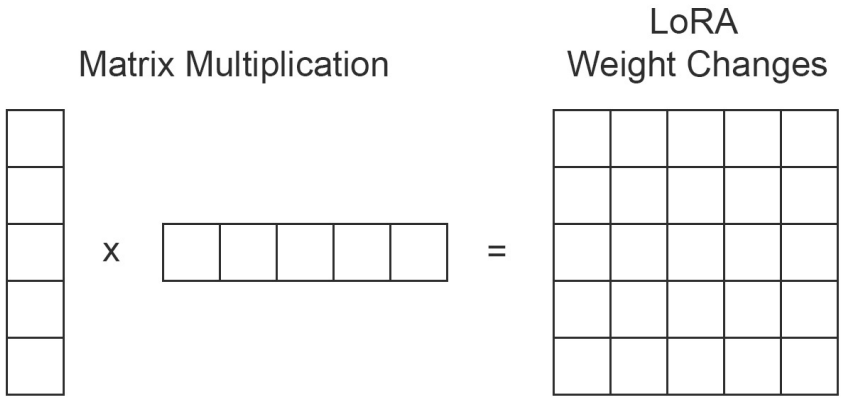


Figure 4.1: Matrix multiplication. Source: [21]

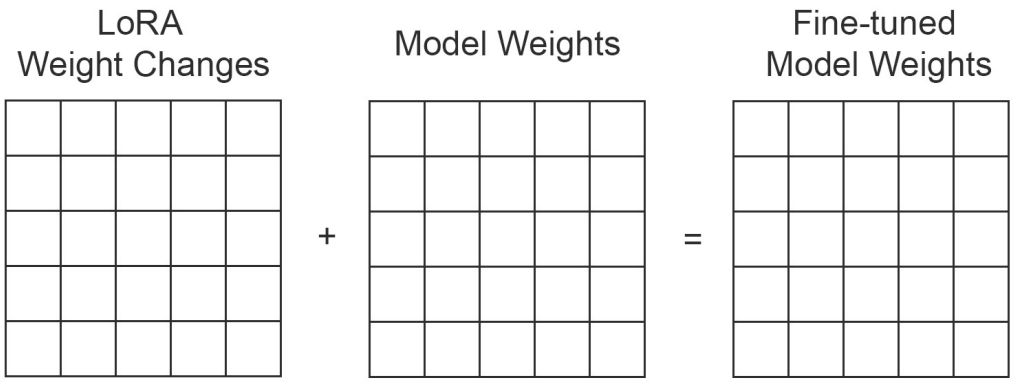


Figure 4.2: Matrix addition. Source: [21]

of LoRA network doesn't slow down the usage, eg. prompting from the LLM, to the end-user. Modularity is also a useful feature of LoRA. One could swap LoRA weight matrix in a situation where the domain is needed to be changed. In this situation, developers could swap the LoRA weight matric accordingly, assuming they have multiple different, but same size LoRA matrices for different domains for the same base-model.

Lora can be defined as follows:

Let W' represent the fine-tuned weight matrix of a neural network, in case of this thesis, an LLM's Neural Network. W' consists of the following:

$$W' = W + \Delta W$$

where W is the original weight matrix of the base-model used and

$$\Delta W = AB, A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times k}$$

where $r \ll min(d, k)$ is the rank of the low rank decomposition.

LoRA uses much less trainable parameters than is needed for the full fine-tuning of the base-model. The amount of LoRAs trainable parameters can be calculated as follows:

$$\#params = d \times r + r \times k = r(d + k)$$

d and k represent input and output dimensions of the base model neural network. r is much smaller than d or k. This means the number of trainable parameter in the decomposed matrix is typically a fraction of the original W, which is the weight matrix of the base-model. [23] [24]

5 Base Model: GPT-OSS

GPT-OSS models were released in August 2025 under Apache 2.0 license for the community to develop their own implementations using these models. OpenAI released both 120 billion and 20 billion parameter models at the same time. These models are open-weight, text-only, reasoning models and they are promised to be capable of instruction following, tools use and reasoning. They support Chain-of-Thought and Structured outputs.

GPT OSS models both use an inference format called Harmony Response Format to structure the conversations. Harmony Response Format has five different roles: system, developer, user, assistant and tool. They have a hierarchy which model complies to in case of instructions conflict: system > developer > user > assistant > tool.

Answers consist of three distinct channels: final, analysis and commentary. Final is the channel which is intended to be shown to the end users, they are therefore the models responses. Analysis is the channel for Chain-of-Thought output. Commentary will include calls for any function tool calls. Internal tools will be triggered usually from the analysis channel. [25] [26]

In this thesis project the Harmony response format was used accordingly. Different reasoning levels were experimented with: low, medium and high. These are defined in the system prompt. Tools or commentary messages were not used.

5.1 Mixture-of-Experts

One recently emerged technology regarding neural network evolution is the Mixture-of-Experts models, or MoEs. In essence, Mixture-of-Experts network consists of specialized sub-networks, which are trained with different tasks to specialize in. Some of the experts specialize in certain types of data while others are experts with something else. These experts or sub-networks can consist for example Feed-Forward neural networks or whole sub network of Mixture-of-Experts, thus being hierarchical. MoE also has to include a gate network or router, which task is to decide to which expert the data fed through to model is being sent to and where should it be processed. [27] Mixture-of-Experts are memory-demanding networks and the fine-tuning of them faces some special challenges: They tend to be vulnerable to overfitting instead of generalizing enough. These challenges have been luckily addressed with a technique called Instruction tuning. In Instruction tuning we basically give the LLM extra information and instructions which it will utilize to solve the NLP task at hand. The instruction tuning dataset usually can consist of instructions, additional information and the desired output, which will act as the ground truth for the LLM to be trained against. Instruction tuning benefits the model when teaching it to perform translation, question-answering, natural language inference or reading comprehension. [28]

As a 2023 study [30] showed, MoEs actually fall short in performance if not using instruction tuning techniques but when properly instruction fine-tuned, can exceed larger models.

5.2 Chain-of-Thought Reasoning

This leads to the next step on: Chain-of-Thought prompting and reasoning. Chain-of-Thought or CoT is a technique to encourage the LLM to break the task at hand to

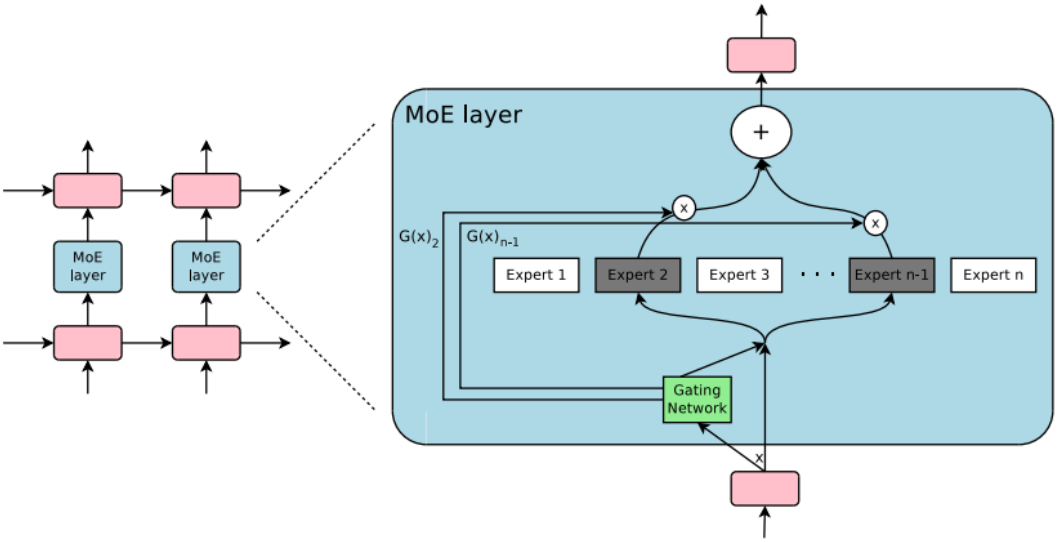


Figure 5.1: Mixture-of-Experts. Source: [29]

smaller pieces in order to land to the correct output more often. One can do this by giving examples which include the breaking down the problem to its parts, chaining the individual parts of the whole answer together to form a chain of thoughts. This can also be addressed by telling to the LLM when prompting it to "think step-by-step". This could and should lead the LLM to break the problem down to its parts and produce reasoning steps before the actual output. [31] Interestingly, in 2024 study Google DeepMind researchers discovered that the explicit CoT-prompting of the LLM isn't always necessary: By performing minor decoding changes they achieved the model implicitly perform chain-of-thought reasoning even without the user telling it to do so. [33]

5.3 Quantization

Quantization means the conversion of high-precision floating point numbers to less memory-consuming integers. This process saves memory and speeds up the inference time. Usually LLM Neural Networks store weights and activations in high-precision

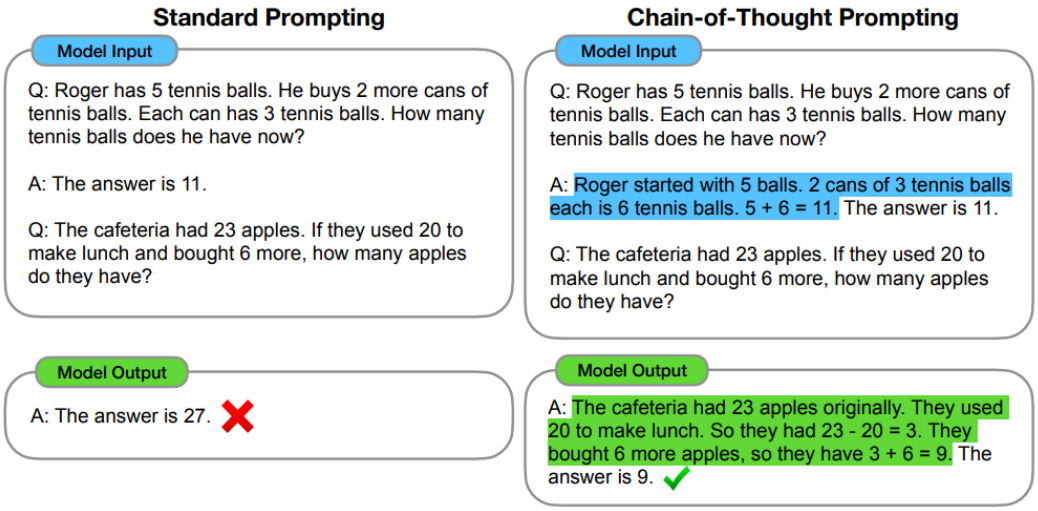


Figure 5.2: Chain of Thought prompting. Source: [32]

32-bit floating point numbers (FP32) or 16-bit floating point numbers (FP16). These values are converted to either 8-bit integers (INT8) or sometimes to 4-bit integers (INT4). This compression could have some side-effects.

One is the lower precision which is a direct consequence of quantization and could lead to problems. Especially, in a larger neural network the precision errors could cumulate and cause model to become less accurate. [34]

However, due to hardware limitations a quantized version of GPT OSS 20B model was used in this thesis. This is introduced in the next chapter.

5.3.1 Unsloth

Unsloth has built a compressed version of GPT OSS 20B. It uses 4-bit integers, quantized from the original 16-bit floating point numbers. [35]

Unsloth have also fixed some technical issues of the original model, including precision issues in Tesla T4 and float16 machines. The Unsloth GPT OSS 20B model uses the Harmony format for conversations as well.

The usage of Unsloth library for fine-tuning was obligatory, because otherwise

the weights have been upcasted to bfloat16 floating-point format (BF16). But when the training was done with Unsloth, the minimum VRAM requirement is 14 GB of VRAM, instead of the 65 GB minimum. CSC Puhti hardware had 4 GPUs of 32 GB VRAM each, so even two GPUs on Puhti supercomputer would not have sufficed. Therefore, quantization and the Unsloth training library were used. [36]

6 Experiments

6.1 Puhti Hardware

Punti is a supercomputer maintained by CSC - IT Center for Science Ltd. It was deployed in September 2019. [37] Puhti has a partition for AI use, Puhti AI, which consist of total 80 GPU nodes, peaking 2.7 petaflops performance. It has four Nvidia V100 GPUs with 32 GB memory each.

Fine-tuning GPT OSS 20B needs more than 32 GB of memory without quantization but using the Unsloth's quantized models the memory footprint decreased and the LoRA fine tuning process was able to be performed on a single V100 GPU. This allowed to use familiar tools such as Jupyter notebook for running code. In addition, Puhti has a lot of users interacting with it at the same time, so less time for queuing was needed when using only one GPU. [38]

Puhti is going to retire in spring and summer 2026 and will be replaced with Roihu next-generation supercomputer. In comparison, Roihu has 132 GPU nodes and has four Nvidia GH200 Grace Hopper superchips, with 96 GB GPU VRAM each. It has 33.9 petaflops performance for the whole system. [39]

6.2 Datasets and synthetic data

A synthetic dataset [40] was used, including 3800 question-answer pairs and related reasoning outputs, to fine-tune GPT OSS 20B. Only the conversations, which were

		metadata			
answer	id	difficulty	reasoning	topic	question
To maximize umami in your homemade sauces, you need to focus on ingredients rich in glutamates and inosinates...	...	4	To answer this question effectively, I first needed to define what umami is and identify the key compounds that contribute to it (glutamates and nucleotides)...	Umami, Sauce making, Flavor enhancement, Glutamate	How can I maximize umami in my homemade sauces, and what specific ingredients and techniques should I employ to achieve a deeply savory and satisfying flavor profile?

Table 6.1: Datapoint example

labeled in topic field as "cooking techniques" related, were filtered to be be used in training and evaluation. The final filtered dataset was 1103 conversations. Notably, the dataset was purely text without any emojis which had an effect in the output of the training. The dataset aims to serve as a helpful dataset to enhance NLP models culinary understanding of cooking times and culinary logic by step-by-step reasoning.

6.3 Training

The Unsloth gpt-oss model documented here [36] was used, because it needed much less VRAM for Fine-tuning and is, all things considered, a step forward for users of gpt-oss in terms of space and speed. The fine-tuning could be completed on a single CSC's PUHTI GPU node.

Unsloth states that "Unsloth gpt-oss fine-tuning is 1.5x faster, uses 70% less VRAM, and supports 10x longer context lengths. gpt-oss-20b QLoRA training fits on a 14GB VRAM."

6.3.1 Hyperparameters

Some experimentation was done by adjusting some of the hyperparameters. During the first test, the learning rate was set too high: 2e-4. This resulted the model forgetting basic language logic when the training was still progressed and the loss function values went back up again. The batch size was also doubled from one to two and this speed up the training that way.

An evaluation dataset was created and split to be separate from the training dataset. Precision was enhanced by doubling the r-parameter and Lora-Alpha parameter values. The r parameter doubled from 8 to 16 and Lora-Alpha from 16 to 32. This was because the first attempts left lot of memory unused so the resources afforded to double the precision of the LoRA network. Checkpointing was enabled and the best model was loaded in the end.

A Lora-Dropout was included to avoid overfitting in fine-tuning. Gradient accumulation steps value was 4. This means that gradients were computed for multiple batches and only after the 4 steps the model parameters were updated. This memory saving technique was important as our hardware was limited.

According to models configuration information, which could be printed out in the notebook, models hidden layer sizes were 2880 x 2880. Therefore, the LoRA per matrix (r = 16) was 16 * (2880 + 2880) = 16 * 5760. Each layer had 7 matrices so LoRA per layer was 7 * 16 * 5760. Model had 24 layers so the total LoRA parameters was = 24 * 7 * 5760 = 15.5M. 20B model has approximately 20,000M parameters so 15.5M is 0.08 % (0.0008) of the base models parameters.

The final training setting included 992 conversations as an epoch to be fed through the LLM. The average sample size of one conversation output, or answer, field was 2749 tokens when using tokenizer provided by the Unsloth library.

This means the training dataset size was approximately 2 700 000 tokens. According to this article [41], this seems to be enough for style tuning, rather than

teaching the LLM new domains or skills. If the domain would be harder or more specialized, the data amount used would not simply suffice.

An evaluation set included 101 conversations for evaluating how "surprised" the LLM would be about these evaluation set conversations.

R-parameter or Lora network was 16 and Lora-Alpha was 32. Lora-Dropout was 0.05. Unsloth's own gradient checkpointing was used. Bias was none. Target modules were q-proj, k-proj, v-proj, o-proj, gate-proj, up-proj and down-proj.

Per device train batch size was 2, gradient accumulation steps were 4, warmup steps 5, and evaluation steps were 5. 3 rounds were checkpointed during training and the best model was loaded in the end, according to the lowest loss. Learning rate was 2e-5 and optimizer used was adamw-8bit. Weight decay was 0.001 and learning-rate-scheduler-type was linear.

A chat template was used to train the model only with responses, so it wouldn't memorize the questions.

The training took around 218 minutes and the peak reserved memory was 24.871 GB. For training, the amount of this was 9.267 GB in it's maximum peak.

6.4 Evaluation

The outputs of the vanilla, not-fine-tuned, version of GPT OSS 20B, and the freshly fine-tuned version were evaluated by testing their abilities to spot food ingredients, cooking output products and cooking actions out of recipe lines. For this a manually annotated dataset was used, introduced in PizzaCommonSense paper [11]. The dataset consists of individual pizza recipes split into action lines, or lines of one cooking step each. On each line there is an input, action and output. The dataset was concatenated into a single file to so it would be easier to handle and an extra field "actions2" was created, which consisted of value of either "action" or "actions" which were used in the original dataset in a mix where the other one was always

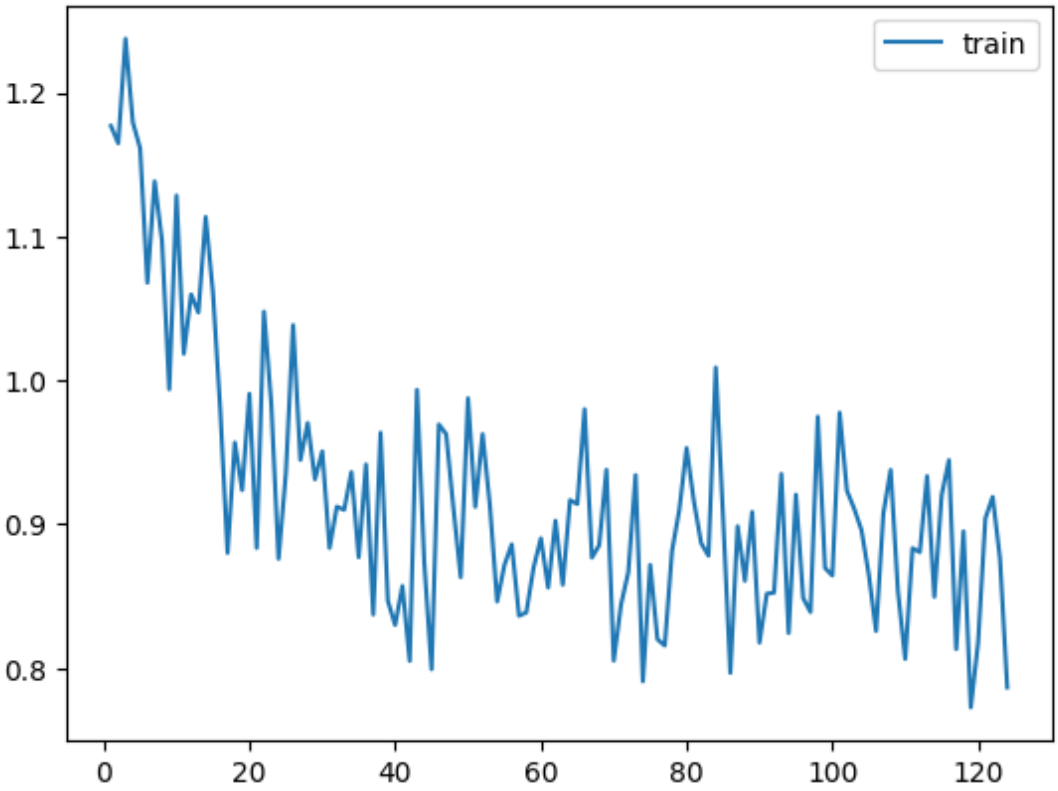


Figure 6.1: Training loss

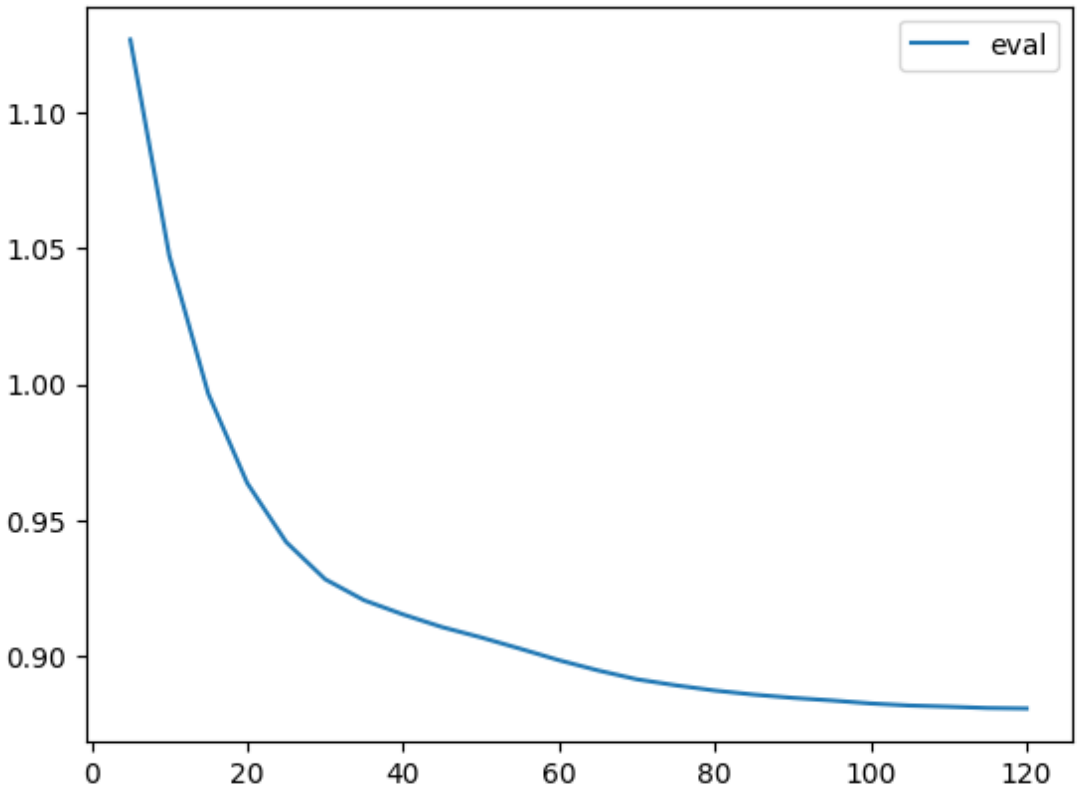


Figure 6.2: Evaluation loss

empty.

6.4.1 Measuring

The LLM outputs similarity with the original human annotated ground truths were measured by using Meteor, Rouge and BertScore. BLEU was not used, and according to type of our data, the most suitable measurement for this situation was BertScore. Evaluation cases were short clauses with JSON formatting. BLEU is typically used for machine translation and it focuses on exact word accuracy and literal word overlap. Human annotators and LLMs rarely make exactly same kind of answers considering this problem.

Meteor calculates precision and recall and a word order penalty. It was developed to address some of the limitations of BLEU. Meteor incorporates synonyms and stemming in order to measure general linguistic similarity.

Rouge is the next measurement used. Rouge measures the similarity of the texts by counting overlapping n-grams meaning adjacent similar words in the same order. Rouge1 calculates 1-grams, Rouge2 2-grams and RougeL is based on the longest common subsequence.

Next is BertScore which is in my opinion the most suitable one. It uses a BERT model to form contextual embeddings for both the original reference and generated candidate texts. Then, cosine similarity for these embeddings is calculated. Focus is more on semantic similarity than exact word matching. One doesn't want the model to memorize exactly the correct answers but to produce semantically similar texts than human annotators have done.

BertScore uses a neural network and BERT to calculate the embeddings and therefore is more computationally expensive than other measuring techniques mentioned. It relies on large pre-trained transformer-based language models.

BertScore finally outputs precision, recall and F1 scores. The quality of the

measurements is tied to underlying BERT model. [42]

6.5 Results

The 992 question-answer pairs was only adequate amount for fine-tuning GPT OSS 20B style and persona. The loss plots went down quite nicely and the steer of the drop slowed down as intended. So something was acquired from the dataset.

The quantitative testing using pizza recipe lines showed that the fine-tuning didn't make the model perform better, but the performance stayed rather on the same level as without fine-tuning.

First, the vanilla model performance and the LoRA performance were compared to each other. It was observed that sometimes the LoRA versions answer was in a format which were not correctly parsed as JSON object and therefore the "answer" field was left empty. This was due to model answering in a commentary or tools channel, rather than final. These resulted as zeros in BertScore evaluations and other measurements, worsening the results of LoRA. But after these parsing errors were manually imputed, the measurements were almost identical in numeric observation.

I produced a list of 10 recipe titles with a short general description by prompting larger GPT 5 in the following manner:

"Produce me a list of 10 cooking recipe titles and short general descriptions. Format it into a json with values: id, title and description."

The larger model output the following JSON which then was used in the prompts for the GPT OSS 20B models, the vanilla and the LoRA fine-tuned versions.


```
[
  {
```

```
    "id": 1,
    "title": "Classic Spaghetti Bolognese",
    "description": "A hearty Italian pasta dish made
    with slow-simmered meat sauce, tomatoes, and aromatic herbs."
  },
  {
    "id": 2,
    "title": "Creamy Chicken Alfredo",
    "description": "Tender chicken tossed with
    pasta in a rich, creamy Parmesan garlic sauce."
  },
  {
    "id": 3,
    "title": "Vegetable Stir-Fry",
    "description": "A quick and colorful mix of fresh
    vegetables sautéed in a light soy-based sauce."
  },
  {
    "id": 4,
    "title": "Classic Beef Tacos",
    "description": "Seasoned ground beef served in
    crispy or soft shells with fresh toppings."
  },
  {
    "id": 5,
    "title": "Margherita Pizza",
    "description": "A simple pizza topped with tomato
```

```
    sauce, fresh mozzarella, and basil."
  },
  {
    "id": 6,
    "title": "Lemon Garlic Roasted Chicken",
    "description": "Juicy roasted chicken flavored
    with lemon, garlic, and herbs."
  },
  {
    "id": 7,
    "title": "Creamy Tomato Soup",
    "description": "A smooth and comforting soup
    made from ripe tomatoes and cream."
  },
  {
    "id": 8,
    "title": "Classic Caesar Salad",
    "description": "Crisp romaine lettuce tossed
    with creamy Caesar dressing, croutons, and Parmesan."
  },
  {
    "id": 9,
    "title": "Baked Salmon with Herbs",
    "description": "Oven-baked salmon fillets
    seasoned with fresh herbs and olive oil."
  },
  {
```

```
    "id": 10,
    "title": "Chocolate Chip Cookies",
    "description": "Soft and chewy cookies
    filled with melted chocolate chips."
  }
]
```


The models were prompted in the following way:

"You are an educated cooking AI Chatbot which provides quality recipes and answers cooking-related questions. Give me full detailed recipe with ingredients for this dish: {placeholder for the title}, {placeholder for the description}."

The recipes were prompted with medium reasoning setting. It was noticeable during later testing that higher reasoning didn't necessarily produce better outputs. Especially with recipe number 10, where the medium reasoning tended to overthink to properties of the chocolate chip cookies, in my opinion. Here is a part of the thinking process of LoRA:

"Interpretation: He wants a recipe for chocolate chip cookies, presumably soft and chewy, but he also mentions "filled with melted chocolate chips." That may be a twist: The cookie dough is encased around chocolate chips that have been melted and set (like chocolate centers). That could be a kind of "chocolate center cookie." He wants full detailed recipe with ingredients. We must provide: Title? "Chocolate Chip Cookies" (but with melted chocolate chip centers). Provide directions.We need to incorporate the twist: The centers should be melted chocolate, cooled to set. He wants "soft and chewy cookies filled with melted chocolate chips." Might be just a fancy description."

Both models also in the latter food instructions test hallucinated sometimes the ingredient "pepperoni" in recipes which had no pepperoni in their whole recipe mentioned at all. This was one rather funny hallucination: the base model seems to

Classic Spaghetti Bolognese

A hearty Italian pasta dish made with slow-simmered meat sauce, tomatoes, and aromatic herbs.

👋 Quick Overview

Item	Details
Yield	6–8 servings
Prep time	20 min
Cook time	1 hr 30 min (includes slow-simmer)
Total	≈ 1 hr 50 min
Key flavors	Rich beef, sweet onions, fresh tomatoes, fragrant herbs, a hint of wine and a splash of milk to smooth acidity

📋 Ingredients

Ingredient	Quantity	Notes
Olive oil	3 Tbsp	Warmed before adding aromatics

Figure 6.3: Example output of Vanilla model

think pepperoni when talking about pizzas in general and without pepperoni.

The most notable difference in output comparison was the lack of emojis in the fine-tuned model. Base model without fine-tuning supplemented some of the recipes with lots of different emojis. But our data for fine-tuning didn't have a single emoji in it. In my test prompting for both of the vanilla and fine-tuned model I didn't specifically ask for emoji-supplemented answers but the vanilla GPT gave a lot of emojis in the text anyway. So did our model suffer from minor effects of catastrophic forgetting? [43]

At least the fine-tuned models output resembled now more our training data format than the base models output. The model learned the style but not any new skills or domains as were rather expected due to small dataset size.

But the recipes are quite similar otherwise. This is of course due to staggeringly good recipe producing quality of the base model GPT OSS 20B.

6. Nutrition Snapshot (per taco – estimates)

Nutrient	Amount per taco (soft shell, 3 inches)
Calories	~210
Protein	12 g
Total fat	9 g
Saturated fat	3 g
Carbohydrate	18 g
Sodium	320 mg

(Numbers vary slightly based on exact ingredients and shell size.)

Figure 6.4: Nutrition info example of Vanilla model

Both models capabilities of producing structured JSON-output consisting of food instructions, input, actions and output as described in PizzaCommonSense paper was tested next. Both models handled this well. the Meteor, Rouge and BertScore metrics were measured of the outputs compared to ground truths, which were human-annotated input, actions and output lines based on the instruction line and the whole recipe. The LLMs were prompted accordingly, in a similar manner:

"This is the introductory instructions given to human annotators who gathered this data:

Describing Intermediate Food States Before and After Cooking Actions

In this task, you will be presented with a table containing four columns: 'Instructions', 'Input', 'Actions', 'Output'."

Your goal is to fill in the 'Input' and 'Output' columns with descriptive names of the intermediate states of various food items before and after the application of

Classic Spaghetti Bolognese

A hearty Italian pasta dish made with slow-simmered meat sauce, tomatoes, and aromatic herbs.

Ingredients (Serves 6)

For the Bolognese Sauce

- **Olive Oil**: 4 tablespoons
- **Ground Beef**: 500 grams (approximately 1 pound)
- **Ground Pork** (optional for richer flavor): 250 grams (about 0.5 pounds)
- **Onion**: 1 large, finely chopped
- **Carrots**: 2 medium, peeled and diced
- **Celery**: 2 stalks, finely chopped
- **Garlic**: 4 cloves, minced
- **Fresh Tomato Purée or Passata**: 200 ml (~¾ cup)
- **Canned Whole Tomatoes or Crushed Tomatoes**: 800 grams (~28 ounces)
    - *If using whole tomatoes, crush them by hand or with a fork while still in the can.*
- **Beef Stock or Broth**: 200 ml (~¾ cup)
- **Whole Milk or Heavy Cream**: 100 ml (~¼ cup)
- **Bay Leaf**: 1
- **Dried Oregano**: 1 teaspoon
- **Dried Thyme**: 1 teaspoon
- **Dried Basil** (optional): 1 teaspoon, or 1 tablespoon fresh chopped basil added at the end
- **Red Pepper Flakes**: Pinch (optional)
- **Salt and Pepper**: to taste

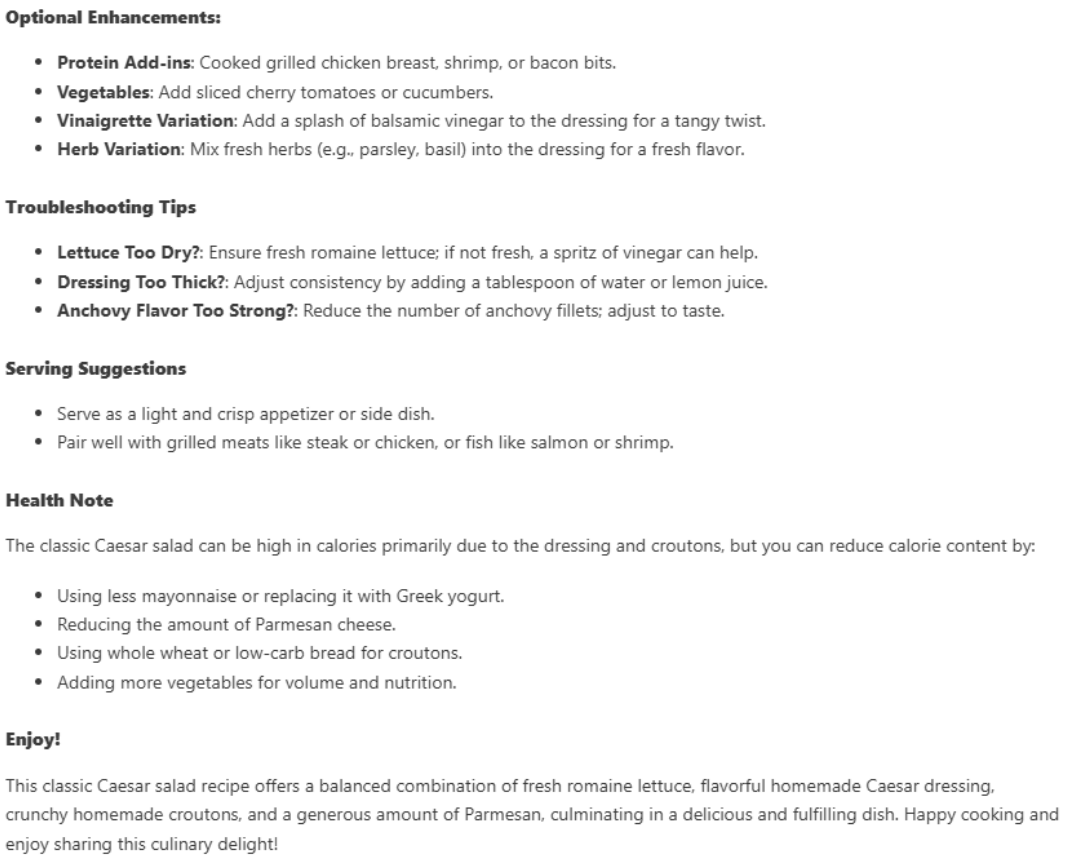
For the Pasta

- **Spaghetti**: 400 grams (approximately 14 ounces)
- **Water**: 4 liters (for boiling)
- **Salt**: 2 teaspoons (for boiling water)

Garnish (Optional)

- **Fresh Basil Leaves**: A handful, torn or whole
- **Grated Parmesan Cheese**: For serving
- **Extra Olive Oil**: Light drizzle before serving
- **Freshly Ground Black Pepper**: For finishing

Figure 6.5: Example output of LoRA model



Optional Enhancements:

- **Protein Add-ins**: Cooked grilled chicken breast, shrimp, or bacon bits.
- **Vegetables**: Add sliced cherry tomatoes or cucumbers.
- **Vinaigrette Variation**: Add a splash of balsamic vinegar to the dressing for a tangy twist.
- **Herb Variation**: Mix fresh herbs (e.g., parsley, basil) into the dressing for a fresh flavor.

Troubleshooting Tips

- **Lettuce Too Dry?**: Ensure fresh romaine lettuce; if not fresh, a spritz of vinegar can help.
- **Dressing Too Thick?**: Adjust consistency by adding a tablespoon of water or lemon juice.
- **Anchovy Flavor Too Strong?**: Reduce the number of anchovy fillets; adjust to taste.

Serving Suggestions

- Serve as a light and crisp appetizer or side dish.
- Pair well with grilled meats like steak or chicken, or fish like salmon or shrimp.

Health Note

The classic Caesar salad can be high in calories primarily due to the dressing and croutons, but you can reduce calorie content by:

- Using less mayonnaise or replacing it with Greek yogurt.
- Reducing the amount of Parmesan cheese.
- Using whole wheat or low-carb bread for croutons.
- Adding more vegetables for volume and nutrition.

Enjoy!

This classic Caesar salad recipe offers a balanced combination of fresh romaine lettuce, flavorful homemade Caesar dressing, crunchy homemade croutons, and a generous amount of Parmesan, culminating in a delicious and fulfilling dish. Happy cooking and enjoy sharing this culinary delight!

Figure 6.6: Extra info output by LoRA model

the cooking action with the help of the sentence from the 'Instructions' column.

Question for Input: "what food preparation (input) do I need to perform this action?".

Question for Output: "what food preparation (output) is the result of this action?".

Only food/ingredient terms. Avoid verbs in active form, the correct answer requires indicating the what goes into the cooking action as input and outcome of the cooking instruction.

Ignore quantities and measurements.

Please do not overuse "NA" and do not leave cells empty.

Please refer to the instructions before starting the task. Good and bad examples are also provided.

Instructions

1. The aim of this task is to fill-in the gaps in the table based in the instructions.

2. Each row in the table represents a different step in a cooking recipe.

3. For each row, fill-in the "Input" and "Output" cells for that row based on what is said in the "Instruction" cell.

4. The "Input" cell represents the state of the food preparation before the cooking action is applied and the "Output" cell represents the state of the food preparation after the cooking action is applied.

5. The actual input or output might not be explicitly mentioned in the instructions, and so you need to create an appropriate entry for the input or output cells.

6. When the step doesn't lead to any food transformation, such as "preheat the oven", use "NA" as input and output.

7. Do not use the same entry for both the "Input" and "Output" cells unless the output is unchanged by the cooking action. Cooking actions like "place", "move" or "transfer" do not change the state of the preparation.

8. An instruction might mention multiple items. If so, use a semicolon to separate them.

9. Ignore quantities and measurements.

10. Ensure that your responses are clear and understandable.

11. Please do not leave cells empty.

You are given the whole following recipe: {placeholder for the recipe}. Please give the corresponding input, actions and output regarding this instructions line in the recipe: {placeholder for the corresponding instructions line} Answer concisely and briefly only including the final answer in this JSON format: {"instructions":"the original instructions part here","input":"answer for input part","actions":"answer for actions part","output":"answer for output part"}"

It was observed that the GPT OSS 20B already handles this task so well that the idea of fine-tuning it to perform even better in this specific task is not an easy task nor a realistic and achievable goal, with small resources and dataset. This experimentation just proves that the evolution and development of LLMs and also open-weight Neural Network LLMs is major. No fine-tuning is needed nor is useful to make GPT OSS 20B a recipe generator LLM.

Statistically, there were no significant change in model performance. p-values were calculated using Mann-Whittney U-test, which does not assume normality in distributions. Our data ditributions of the measurements were not looking normal with the exception of rouge2 measurements for vanilla GPT OSS outputs. Regardless, all other distributions were not normal so Mann Whittney U-test was chosen. The p-values are listed for different measurements on table 6.3.

When taking a look at benchmarks, GPT OSS 20B had 4th place on CodeForces leaderboard with a score of 0.743. [44] GPT OSS 20B is also holding third place in HealthBench, which is an open-source benchmark focusing on healthcare performance and safety. GPT OSS 20B has score of 0.425, while it's bigger brother holds

Model	Meteor	Rouge1	Rouge2	RougeL	BertScore Precision	BertScore Recall	BertScore F1
Vanilla (n=264)	0.495	0.725	0.566	0.706	0.919	0.898	0.908
LoRA (n=264)	0.478	0.710	0.551	0.693	0.892	0.868	0.880
LoRA (n=264) (imputed)	0.493	0.732	0.569	0.715	0.922	0.897	0.909

Table 6.2: Results table

the silver medal with 0.576 on HealthBench. [45]

Which brings us to the fact that GPT OSS 20B was released same time with the larger GPT OSS 120B. It also has the worldwide 2nd place in CodeForces and currently the 1st place in HealthBench Hard Leaderboard, which is a challenging variation of HealthBench. Interestingly, proprietary model GPT-5 has only the third place, while GPT OSS 20B has the second place. [46]

Maybe larger institutions should study the possibilities of the larger GPT OSS 120B as it is indicated to perform better than the GPT OSS 20B? And sometimes better than the proprietary, most expensive models.

Nevertheless, One thing to consider is to fine-tune even a smaller model with these kind of datasets and then experiment and compare its outputs before and after fine-tuning. GPT OSS 20B is still quite large model for lighter consumer hardware because it typically needs 16 GB of GPU VRAM. [47]

Maybe these datasets could be used to fine-tune eg. Mistral 7B model? The Mistral 7B model was published already in 2023, has 7 billion parameters and fits to a 12 GB VRAM GPU, eg. RTX 3060 with 12GB of VRAM. [48]

Measurement	p-value	statistically significant difference
Meteor	0.713	no
Rouge1	0.524	no
Rouge2	0.789	no
RougeL	0.475	no
BS precision	0.977	no
BS recall	0.112	no
BS f1	0.453	no

Table 6.3: p-values table

Truth	{
	"instructions":
	"brown ground_beef with onions ,
	garlic_salt and pepper .",
	"input": "(ground_beef; onions;
	garlic_salt; pepper)",
	"actions": "brown",
	"output": "browned
	ground_beef_mixture"
	}

Output	{
	"instructions":"brown ground_beef with onions ,
	garlic_salt and pepper .",
	"input":"ground_beef, minced_onions,
	minced_garlic, salt, pepper",
	"actions":"brown",
	"output":"brown_ground_beef"
	}

Table 6.4: Vanilla test example

Truth	{ "instructions": "brown ground_beef with onions , garlic_salt and pepper .", "input": "(ground_beef; onions; garlic_salt; pepper)", "actions": "brown", "output": "browned ground_beef_mixture"}

Output	{
	"instructions":
	"Brown ground beef with onions, garlic,
	salt, and pepper",
	"input":
	"raw ground beef; raw onions; raw garlic;
	raw salt; raw pepper",
	"actions":"Brown ground beef",
	"output":
	"brown ground beef;
	cooked onions; cooked garlic"
	}

Table 6.5: LoRA test example

7 Conclusion

Conclusion is that the LLM evolution has been such rapid that even relatively small open-weight models tend to be highly qualified for such a task as cooking recipe generation. To force an LLM to specialize in cooking recipes only, is not needed at all. This has become evident during the writing of this master's thesis.

The answers to research questions presented in the beginning are as follows:

1. Does LoRA fine-tuning improve domain-specific performance of GPT-OSS 20B in cooking recipe generation?

Answer:

H0: There is no meaningful improvement, when comparing and evaluating by human observation.

Recipes were good to start with and some markdown styling and emojis were lost when using the text-only dataset. And as discussed earlier, dataset was such size to teach only the style, which it did. With better computing, memory and time resources one could feed larger amounts of data and test, whether this would improve quality of the recipes sufficiently. But an additional note must be stated that there is no meaningful improvement, when using small dataset which would only affect the style of the LLM outputs. So more study is needed using more resources.

Next is the second one.

2. Does LoRA fine-tuning improve structured food-state reasoning performance?

Answer:

H0: There is no statistically significant improvement.

This also comes with a side note that results didn't worsen or improve in statistically meaningful way, but one might be able to improve the results when using more data, computing resources, memory and time.

At start, it seemed that our model got worse executing this task but in reality it just changed its style in a few cases which one wasn't prepared for when preparing the parsing in the program. (Technically it didn't worsen in food-state reasoning, but made it harder to obtain the JSON from the results string texts.) But these parsing errors were imputed and the measurements were done again. And then, it became evident that there were no statistically meaningful difference in answers.

Style or persona fine-tuning using LoRA, which in this case meant fine-tuning with only about 1000 examples, did not improve GPT OSS 20Bs capabilities in a task which it does well already out-of-the-box.

The dataset used would probably suite better a smaller model, and should be mixed with similar outputs than the model in question already produces, to not to lose any information, as we did lose the emojis this time.

Also, I think it would be interesting to connect the LLM to a large dataset of readily available cooking recipes and build a RAG solution. This could be one successful path if the recipe style and restrictions are already known and such recipes are gathered to a one recipe bank. Internet is full of community produced recipe banks which have already been crawled through and made to datasets.

And just simply experimenting with various prompts and more advanced prompting patterns and techniques would probably produce lots of variability in the recipes that the LLM would output. One could for example build a structured prompt with just placeholders for variables that change each time, for example allergens to avoid and special diets to comply, protein intake to reach and so on.

Or one could fuse all three above: smaller base model, RAG and prompting

techniques to achieve somewhat complex but probably interesting solution to cover all one's cooking recipe needs.

I assumed that fine-tuning an LLM with cooking related conversations would improve the recipe generation capabilities. This was not totally true, because just by simply constructing decent system prompts for vanilla base model did suffice.

Future is bright for open-weight models since they sometimes even surpass their proprietary counterparts. Small models have their place but so do the larger open weight models.

One thing to consider is AI safety. One good solution for institutions to obey safety guidelines and regulations is to host models themselves in a safe, contained environment, so that the models really can't use any mechanisms to leak the data outside. This is what GPT OSS 20B and 120B are really useful and a spot on choice in my opinion. In my opinion, sometimes regulatory environments slow down AI development and usage and the reason is noble and concern is valid. We want to keep our private matters private. So AI isn't always used in healthcare, nor any salary, HR or personal information should be given to it in cases where there is a risk of leakage. By using permissively licensed, open-weight, self-hosted models, one can mitigate this risk effectively.

The real matters self-hosted AI models should concern on, are far greater than culinary recipes. They relate to humans, Eg. health, social issues, personal information. One should keep following the trends and evolution or rather the revolution of AI LLM models, but still keep themselves grounded to reality by taking care of cyber security and data protection. Open, self-hosted models are a gateway to enable safe growth of AI LLM usage in various organizations and use cases.

One thing to hope for is that the fully open source models keep on advancing as well. GPT OSS 20B and 120B are only open-weight models but not truly open source. By letting developers to modify the LLM code base, companies could in-

troduce their own extra safety mechanisms and measures to take place and even enhance security further. Also, in my opinion, the democratization of AI technology relies on open source.

References

[1] J. Ahn, R. Verma, R. Lou, D. Liu, R. Zhang, and W. Yin, *Large language models for mathematical reasoning: Progresses and challenges*, 2024. arXiv: 2402.00157 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.00157.

[2] A. Anand et al., *Enhancing llms for physics problem-solving using reinforcement learning with human-ai feedback*, 2024. arXiv: 2412.06827 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2412.06827.

[3] P. Kallas, *Llm survey 2024: Generative ai adoption statistics at work*, https://amperly.com/llm-survey-generative-ai-adoption-statistics/, Accessed: 2025-09-28.

[4] R. A. Husein, H. Aburajouh, and C. Catal, "Large language models for code completion: A systematic literature review", *Computer Standards Interfaces*, vol. 92, p. 103 917, 2025, ISSN: 0920-5489. DOI: https://doi.org/10.1016/j.csi.2024.103917. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548924000862.

[5] J. J. Wu and F. H. Fard, "Humanevalcomm: Benchmarking the communication competence of code generation for llms and llm agents", *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 7, Aug. 2025, ISSN: 1049-331X. DOI: 10.1145/3715109. [Online]. Available: https://doi.org/10.1145/3715109.

[6] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba, "Recipe1M+: A dataset for learning cross-modal embeddings for cooking recipes and food images", *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[7] Helena H. Lee, Ke Shu, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Li, Ee-Peng Lim, Lav R. Varshney, "RecipeGPT: Generative pre-training based cooking recipe generation and evaluation system", *In Proceedings of ACM Conference (Conference'17)*, 2020.

[8] Fnu Mohbat, Mohammed J. Zakia, "LLaVA-Chef: A multi-modal generative model for food recipes", *In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*, 2024.

[9] Haotian Liu, Chunyuan Li, Qingyang Wu, Yong Jae Lee, "Visual instruction tuning", *NeurIPS 2023 (Oral)*, 2023.

[10] Guoshan Liu, Hailong Yin, Bin Zhu, Jingjing Chen, Chong-Wah Ngo, Yu-Gang Jiang, "Retrieval Augmented Recipe Generation", *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025.

[11] Aissatou Diallo, Antonis Bikakis, Luke Dickens, Anthony Hunter, Rob Miller, "PizzaCommonSense: A dataset for commonsense reasoning about intermediate steps in cooking recipes", *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

[12] Amazon, *What is transfer learning?*, `https://aws.amazon.com/what-is/transfer-learning/`, Accessed: 2025-09-06.

[13] L. T. Edward Beeching Quentin Gallouédec, *Fine-tuning with gpt-oss and hugging face transformers*, `https://cookbook.openai.com/articles/gpt-oss/fine-tune-transfomers`, Accessed: 2025-09-06.

[14] Amazon, *What is RAG (retrieval-augmented generation)?*, https://aws.amazon.com/what-is/retrieval-augmented-generation/, Accessed: 2025-09-06.

[15] Amazon, *What is prompt engineering?*, https://aws.amazon.com/what-is/prompt-engineering/, Accessed: 2025-09-06.

[16] M. Nadăş, L. Dioşan, and A. Tomescu, "Synthetic data generation using large language models: Advances in text and code", *IEEE Access*, vol. 13, pp. 134 615–134 633, 2025, ISSN: 2169-3536. DOI: 10.1109/access.2025.3589503. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2025.3589503.

[17] P. Muzumdar, S. Cheemalapati, S. R. RamiReddy, K. Singh, G. Kurian, and A. Muley, "The dead internet theory: A survey on artificial interactions and the future of social media", *Asian Journal of Research in Computer Science*, vol. 18, no. 1, pp. 67–73, Jan. 2025, ISSN: 2581-8260. DOI: 10.9734/ajrcos/2025/v18i1549. [Online]. Available: http://dx.doi.org/10.9734/ajrcos/2025/v18i1549.

[18] N. J. Prottasha et al., *Peft a2z: Parameter-efficient fine-tuning survey for large language and vision models*, 2025. arXiv: 2504.14117 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2504.14117.

[19] M. Kowsher, N. J. Prottasha, and P. Bhat, *Propulsion: Steering llm with tiny fine-tuning*, 2024. arXiv: 2409.10927 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2409.10927.

[20] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, *Parameter-efficient fine-tuning for large models: A comprehensive survey*, 2024. arXiv: 2403.14608 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2403.14608.

[21] M. Hennings, *Lora fine-tuning hyperparameters explained (in plain english)*, https://www.entrypointai.com/blog/lora-fine-tuning/, Accessed: 2025-09-20.

[22] E. J. Hu et al., *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2106.09685.

[23] E. J. Hu et al., *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2106.09685.

[24] Y. Gang, J. Shun, and M. Qing, "Smarter Fine-Tuning: How LoRA Enhances Large Language Models", working paper or preprint, Mar. 2025. [Online]. Available: https://hal.science/hal-04983079.

[25] OpenAI et al., *Gpt-oss-120b & gpt-oss-20b model card*, 2025. arXiv: 2508.10925 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2508.10925.

[26] OpenAI, *Openai harmony response format*, https://cookbook.openai.com/articles/openai-harmony, Accessed: 2026-01-31.

[27] O. Sanseviero, L. Tunstall, P. Schmid, S. Mangrulkar, Y. Belkada, and P. Cuenca, *Mixture of experts explained*, 2023. [Online]. Available: https://huggingface.co/blog/moe.

[28] IBM, *What is instruction tuning?*, https://www.ibm.com/think/topics/instruction-tuning, Accessed: 2025-09-18.

[29] N. Shazeer et al., *Outrageously large neural networks: The sparsely-gated mixture-of-experts layer*, 2017. arXiv: 1701.06538 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1701.06538.

[30] S. Shen et al., *Mixture-of-experts meets instruction tuning:a winning combination for large language models*, 2023. arXiv: 2305.14705 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2305.14705.

[31] IBM, *What is chain of thought (cot) prompting?*, https://www.ibm.com/think/topics/chain-of-thoughts, Accessed: 2025-09-18.

[32] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2201.11903.

[33] X. Wang and D. Zhou, *Chain-of-thought reasoning without prompting*, 2024. arXiv: 2402.10200 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.10200.

[34] IBM, *What is quantization?*, https://www.ibm.com/think/topics/quantization, Accessed: 2026-01-31.

[35] A. M. Learning, *Gpt-oss 20b*, https://apxml.com/models/gpt-oss-20b, Accessed: 2026-02-01.

[36] Unsloth, *Gpt-oss: How to run guide*, https://unsloth.ai/docs/models/gpt-oss-how-to-run-and-fine-tune, Accessed: 2026-01-25.

[37] CSC, *Finnish supercomputers: Mahti and puhti*, https://csc.fi/en/our-expertise/high-performance-computing/supercomputers-mahti-and-puhti/, Accessed: 2026-01-31.

[38] CSC, *Technical details about puhti*, https://docs.csc.fi/computing/systems-puhti/, Accessed: 2026-01-31.

[39] CSC, *Roihu supercomputer*, https://docs.csc.fi/computing/systems-roihu/, Accessed: 2026-01-31.

[40] M. Wesney, *Cot_reasoning_cooking*, https://huggingface.co/datasets/moremilk/CoT_Reasoning_Cooking, 2025.

[41] TensorBlue, *Llm fine-tuning guide 2025: Complete tutorial for gpt-4, llama, mistral*, https://tensorblue.com/blog/llm-fine-tuning-complete-guide-tutorial-2025, Accessed: 2026-02-14.

[42] K. Dhungana, *Nlp model evaluation: Understanding bleu, rouge, meteor, and bertscore*, https://readmedium.com/nlp-model-evaluation-understanding-bleu-rouge-meteor-and-bertscore-9bad7db71170, Accessed: 2026-02-01.

[43] IBM, *What is catastrophic forgetting?*, https://www.ibm.com/think/topics/catastrophic-forgetting, Accessed: 2026-01-17.

[44] llm-stats, *Codeforces*, https://llm-stats.com/benchmarks/codeforces, Accessed: 2026-02-03.

[45] llm-stats, *Healthbench*, https://llm-stats.com/benchmarks/healthbench, Accessed: 2026-02-03.

[46] llm-stats, *Healthbench hard*, https://llm-stats.com/benchmarks/healthbench-hard, Accessed: 2026-02-03.

[47] IntuitionLabs, *Hardware requirements for running gpt-oss-20b locally*, https://intuitionlabs.ai/articles/hardware-requirements-gpt-oss-20b, Accessed: 2026-01-17.

[48] A. Witt, *Choosing hardware for running mistral llm (7b) locally*, https://www.hardware-corner.net/guides/hardware-for-mistral-llm/, Accessed: 2026-02-03.