(lora-adapter)=

LoRA Adapters

This document shows you how to use [LoRA adapters](https://arxiv.org/abs/2106.09685) with vLLM on top of a base model.

LoRA adapters can be used with any vLLM model that implements {class}`~vllm.model_executor.models.interfaces.SupportsLoRA`.

Adapters can be efficiently served on a per request basis with minimal overhead. First we download the adapter(s) and save
them locally with

```python
from huggingface_hub import snapshot_download

sql_lora_path = snapshot_download(repo_id="yard1/llama-2-7b-sql-lora-test")
```

Then we instantiate the base model and pass in the `enable_lora=True` flag:

```python
from vllm import LLM, SamplingParams
from vllm.lora.request import LoRARequest

llm = LLM(model="meta-llama/Llama-2-7b-hf", enable_lora=True)
```

```

We can now submit the prompts and call `llm.generate` with the `lora_request` parameter. The first parameter
of `LoRARequest` is a human identifiable name, the second parameter is a globally unique ID for the adapter and
the third parameter is the path to the LoRA adapter.

```python
sampling_params = SamplingParams(
    temperature=0,
    max_tokens=256,
    stop=["[/assistant]"]
)

prompts = [
        "[user] Write a SQL query to answer the question based on the table schema.\n\n context: CREATE TABLE table_name_74 (icao VARCHAR, airport VARCHAR)\n\n question: Name the ICAO for lilongwe international airport [/user] [assistant]",
        "[user] Write a SQL query to answer the question based on the table schema.\n\n context: CREATE TABLE table_name_11 (nationality VARCHAR, elector VARCHAR)\n\n question: When Anchero Pantaleone was the elector what is under nationality? [/user] [assistant]",
]

outputs = llm.generate(
    prompts,
    sampling_params,
```
```

```
    lora_request=LoRARequest("sql_adapter", 1, sql_lora_path)
)
```

Check out <gh-file:examples/offline_inference/multilora_inference.py> for an example of how to use LoRA adapters with the async engine and how to use more advanced configuration options.

Serving LoRA Adapters

LoRA adapted models can also be served with the Open-AI compatible vLLM server. To do so, we use
`--lora-modules {name}={path} {name}={path}` to specify each LoRA module when we kickoff the server:

```bash
vllm serve meta-llama/Llama-2-7b-hf \
    --enable-lora \
                                                --lora-modules
sql-lora=$HOME/.cache/huggingface/hub/models--yard1--llama-2-7b-sql-lora-test/snapshots/0dfa34
7e8877a4d4ed19ee56c140fa518470028c/
```

:::{note}
The commit ID `0dfa347e8877a4d4ed19ee56c140fa518470028c` may change over time. Please check the latest commit ID in your environment to ensure you are using the correct one.
:::

The server entrypoint accepts all other LoRA configuration parameters (`max_loras`, `max_lora_rank`, `max_cpu_loras`,
etc.), which will apply to all forthcoming requests. Upon querying the `/models` endpoint, we should see our LoRA along
with its base model:

```bash
curl localhost:8000/v1/models | jq .
{
    "object": "list",
    "data": [
        {
            "id": "meta-llama/Llama-2-7b-hf",
            "object": "model",
            ...
        },
        {
            "id": "sql-lora",
            "object": "model",
            ...
        }
    ]
}
```

Requests can specify the LoRA adapter as if it were any other model via the `model` request parameter. The requests will be

processed according to the server-wide LoRA configuration (i.e. in parallel with base model requests, and potentially other
LoRA adapter requests if they were provided and `max_loras` is set high enough).

The following is an example request

```bash
curl http://localhost:8000/v1/completions \
    -H "Content-Type: application/json" \
    -d '{
        "model": "sql-lora",
        "prompt": "San Francisco is a",
        "max_tokens": 7,
        "temperature": 0
    }' | jq
```

Dynamically serving LoRA Adapters

In addition to serving LoRA adapters at server startup, the vLLM server now supports dynamically loading and unloading
LoRA adapters at runtime through dedicated API endpoints. This feature can be particularly useful when the flexibility
to change models on-the-fly is needed.

Note: Enabling this feature in production environments is risky as user may participate model adapter management.

To enable dynamic LoRA loading and unloading, ensure that the environment variable `VLLM_ALLOW_RUNTIME_LORA_UPDATING`
is set to `True`. When this option is enabled, the API server will log a warning to indicate that dynamic loading is active.

```bash
export VLLM_ALLOW_RUNTIME_LORA_UPDATING=True
```

Loading a LoRA Adapter:

To dynamically load a LoRA adapter, send a POST request to the `/v1/load_lora_adapter` endpoint with the necessary
details of the adapter to be loaded. The request payload should include the name and path to the LoRA adapter.

Example request to load a LoRA adapter:

```bash
curl -X POST http://localhost:8000/v1/load_lora_adapter \
-H "Content-Type: application/json" \
-d '{
    "lora_name": "sql_adapter",
    "lora_path": "/path/to/sql-lora-adapter"
}'
```

Upon a successful request, the API will respond with a 200 OK status code. If an error occurs, such as if the adapter
cannot be found or loaded, an appropriate error message will be returned.

Unloading a LoRA Adapter:

To unload a LoRA adapter that has been previously loaded, send a POST request to the `/v1/unload_lora_adapter` endpoint
with the name or ID of the adapter to be unloaded.

Example request to unload a LoRA adapter:

```bash
curl -X POST http://localhost:8000/v1/unload_lora_adapter \
-H "Content-Type: application/json" \
-d '{
    "lora_name": "sql_adapter"
}'
```

New format for `--lora-modules`

In the previous version, users would provide LoRA modules via the following format, either as a key-value pair or in JSON format. For example:

```bash
```

--lora-modules

sql-lora=$HOME/.cache/huggingface/hub/models--yard1--llama-2-7b-sql-lora-test/snapshots/0dfa347e8877a4d4ed19ee56c140fa518470028c/

```

This would only include the `name` and `path` for each LoRA module, but did not provide a way to specify a `base_model_name`.
Now, you can specify a base_model_name alongside the name and path using JSON format. For example:

```bash
--lora-modules '{"name": "sql-lora", "path": "/path/to/lora", "base_model_name": "meta-llama/Llama-2-7b"}'
```

To provide the backward compatibility support, you can still use the old key-value format (name=path), but the `base_model_name` will remain unspecified in that case.

Lora model lineage in model card

The new format of `--lora-modules` is mainly to support the display of parent model information in the model card. Here's an explanation of how your current response supports this:

- The `parent` field of LoRA model `sql-lora` now links to its base model `meta-llama/Llama-2-7b-hf`. This correctly reflects the hierarchical relationship between the base model and the LoRA adapter.
- The `root` field points to the artifact location of the lora adapter.
```

```bash
$ curl http://localhost:8000/v1/models

{
    "object": "list",
    "data": [
        {
        "id": "meta-llama/Llama-2-7b-hf",
        "object": "model",
        "created": 1715644056,
        "owned_by": "vllm",
                                                                                "root":
"~/.cache/huggingface/hub/models--meta-llama--Llama-2-7b-hf/snapshots/01c7f73d771dfac7d2923
23805ebc428287df4f9/",
        "parent": null,
        "permission": [
            {
            .....
            }
        ]
        },
        {
        "id": "sql-lora",
        "object": "model",
        "created": 1715644056,
        "owned_by": "vllm",
                                                                                "root":
```

```
"~/.cache/huggingface/hub/models--yard1--llama-2-7b-sql-lora-test/snapshots/0dfa347e8877a4d4ed
19ee56c140fa518470028c/",
        "parent": meta-llama/Llama-2-7b-hf,
        "permission": [
          {
          ....
          }
        ]
        }
    ]
}
```