Finetuning

Fine-tuning is a powerful process for utilizing TimeGPT more effectively. Foundation models are pre-trained on vast amounts of data, capturing wide-ranging features and patterns. These models can then be specialized for specific contexts or domains. With fine-tuning, the model's parameters are refined to forecast a new task, allowing it to tailor its vast pre-existing knowledge toward the requirements of the new data. Fine-tuning thus serves as a crucial bridge, linking TimeGPT's broad capabilities to your tasks specificities.

Concretely, the process of fine-tuning consists of performing a certain number of training iterations on your input data minimizing the forecasting error. The forecasts will then be produced with the updated model. To control the number of iterations, use the `finetune_steps` argument of the `forecast` method.

In [ ]:
```
# | hide
from dotenv import load_dotenv
```

In [ ]:
```
# | hide
load_dotenv()
```

Out[ ]:
```
True
```

In [ ]:
```
import pandas as pd
from nixtlats import TimeGPT
import os
```

In [ ]:
```
timegpt = TimeGPT(token=os.getenv("TIMEGPT_TOKEN"))
```

In [ ]:
```
# | hide
timegpt = TimeGPT()
```

You can test the validate of your token calling the `validate_token` method:

In [ ]:
```
timegpt.validate_token()
```

```
INFO:nixtlats.timegpt:Happy Forecasting! :), If you have questions or need support,
please email ops@nixtla.io
```

Out[ ]:
```
True
```

Here's an example of how to fine-tune TimeGPT:

In [ ]:
```
df = pd.read_csv(
    "https://raw.githubusercontent.com/Nixtla/transfer-learning-time-series/main/da
)
df.head()
```

Out[ ]:

	timestamp	value
0	1949-01-01	112
1	1949-02-01	118
2	1949-03-01	132
3	1949-04-01	129
4	1949-05-01	121

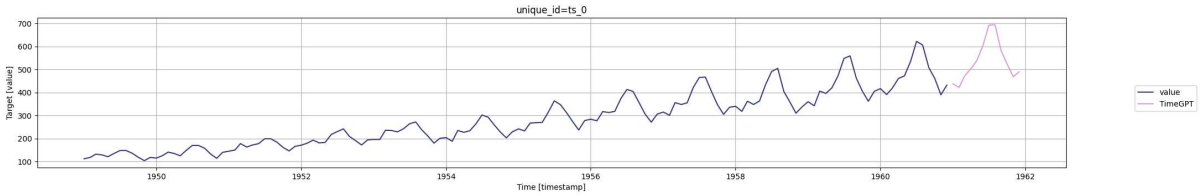
In [ ]:
```
timegpt_fcst_finetune_df = timegpt.forecast(
    df=df,
    h=12,
    finetune_steps=10,
    time_col="timestamp",
    target_col="value",
)
```

```
INFO:nixtlats.timegpt:Validating inputs...
INFO:nixtlats.timegpt:Preprocessing dataframes...
INFO:nixtlats.timegpt:Inferred freq: MS
INFO:nixtlats.timegpt:Calling Forecast Endpoint...
```

> BEFORE: *14 API Calls | 392664 Tokens | 614.81 Spent*
> AFTER: *15 API Calls | 392830 Tokens | 615.06 Spent*
> **USAGE: 1 API Call | 166 Tokens | 0.25 Spent**

In [ ]:
```
timegpt.plot(
    df,
    timegpt_fcst_finetune_df,
    time_col="timestamp",
    target_col="value",
)
```

Out[ ]:



In this code, `finetune_steps=10` means the model will go through 10 iterations of training on your time series data.

Keep in mind that fine-tuning can be a bit of trial and error. You might need to adjust the number of `finetune_steps` based on your specific needs and the complexity of your data. It's recommended to monitor the model's performance during fine-tuning and adjust as needed. Be aware that more `finetune_steps` may lead to longer training times and could potentially lead to overfitting if not managed properly.

Remember, fine-tuning is a powerful feature, but it should be used thoughtfully and carefully.