

# **Generative AI**

# **W4 Agenda**

- **On Premise LLMs**
  - **Why local LLMs are important?**
  - **How to launch LLM locally?**

# On Premise LLMs

**Why local LLMs are  
important?**

# Local LLMs – Pros and Cons



- **Full data privacy**
- Can work offline
- Ability to use underutilized infrastructure e.g. Priv GPUs



- Usually limited to smaller models, with 3-20B params on normal GPUs
- Much slower than SaaS solutions
- Needs very high utilization and optimization to be actually cheaper than small SaaS LLMs

**LLMs you can run locally on a private GPU 12-16 GB (with some quantization)**

- **Gemma 3 12B**
- **LLaMA 13B**
- **Falcon 7B**
- **Bielik-11B-v2**

# What is Quantization??

## Definition

Converting model weights from higher-precision (e.g. FP16 or FP32) down to fewer bits (e.g. 8-bit, 4-bit) to reduce memory usage and computational overhead

## Why It Matters

- **Lower VRAM Requirements:** Fits larger models on smaller GPUs.
- **Faster Inference:** Fewer bits to process speeds up forward passes.
- **Energy Efficiency:** Reduced compute demands save power.

## Approaches

- **Uniform Quantization:** Single scale factor per block of weights.
- **Advanced “K-Quant” Methods:** Multiple scales or per-channel scaling for better accuracy at the same bit depth.

## Trade-Offs

- **Size vs. Accuracy:** Fewer bits can degrade model performance.
- **Implementation Complexity:** Complex quantization methods often require specialized tooling.

# Deploying LLMs locally

# How can you run LMMs locally?

	Ollama	LM Studio	vLLM	Other Notable Tools
<b>Focus</b>	Mac-focused app for LLaMA-family models	Mac-native GUI for local GPT-based models	High-performance Python backend for optimized inference	- <a href="#">Text-Gen WebUI</a> (web-based multi-model) - <a href="#">exllama</a> (CUDA-optimized for GPTQ) - <a href="#">Hugging Face Transformers</a> (broad library + 8-bit/4-bit)
<b>Key Features</b>	- Native macOS binary - Simple CLI & GUI	- macOS GPU acceleration - Intuitive point-and-click usage	- Advanced scheduling - Fast batched inference	- Web or Python-based GUIs - Extensive quantization support (4-bit, GPTQ, etc.)
<b>Pros</b>	- One-click install on macOS - Minimal config	- Slick UI - Easy model downloads	- Very fast throughput - Good multi-GPU scalability	- Flexible interfaces - Large model zoo & community support
<b>Cons</b>	- macOS only - Primarily LLaMA-based	- macOS-specific - Limited model variety so far	- Requires Python & GPU - Less of a GUI solution	- Setup can vary (dependencies, CUDA versions) - Some tools lack "all-in-one" packaging
<b>Use Cases</b>	- Quick local testing on macOS	- Non-technical Mac users wanting local chat	- Power users seeking speed & scale in Python environments	- Customizable solutions for various OS/GPU setups

# Hugging face transformers example

```
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

model_id = 'google/gemma-2b-aps-it'
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map='auto',
    torch_dtype=torch.bfloat16,
)

passage = "For more than 40 years, the lyrics of American Pie have been puzzled o"
messages = [{'role': 'user', 'content': create_propositions_input(passage)}]
inputs = tokenizer.apply_chat_template(messages, return_tensors='pt', add_generat:

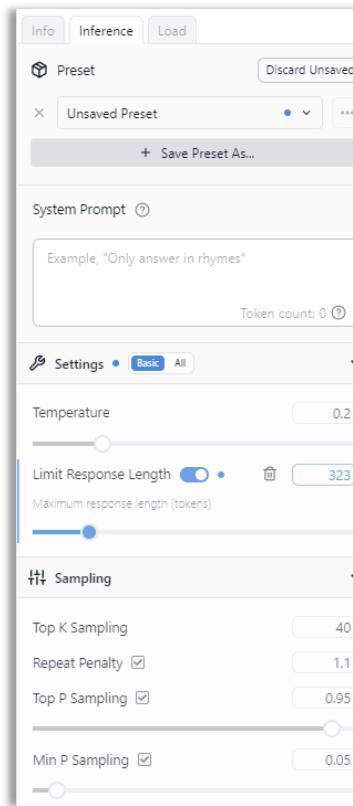
output = model.generate(**inputs, max_new_tokens=4096, do_sample=False)
generated_text = tokenizer.batch_decode(output[:, inputs['input_ids'].shape[1]:],
result = process_propositions_output(generated_text)
print(result)
```

# LM Studio example

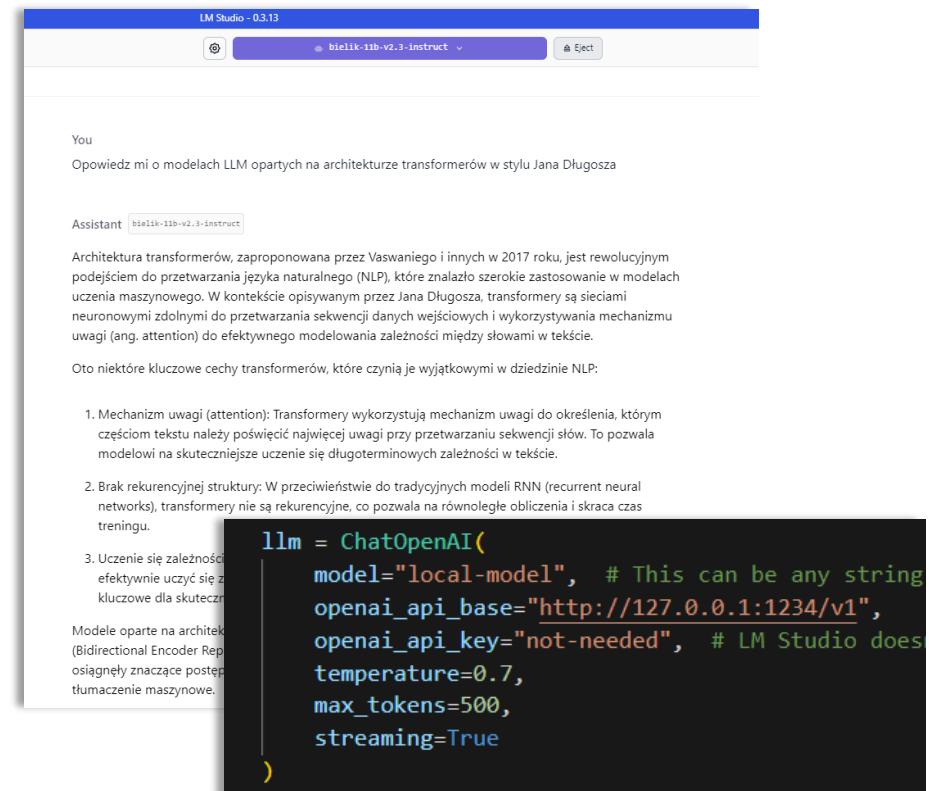
## Choose model



## Setup hyperparams



## Chat using API or UI



# LM Studio exercise

Start with Bonus-on-premise-LLMs.py

```
import requests
import json

from langchain_community.llms import LlamaCpp
from langchain.callbacks.manager import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
from langchain_core.prompts import ChatPromptTemplate
from langchain_community.chat_models import ChatOpenAI
```

✓ 1.3s

## Setup LM studio endpoint

```
# LM Studio endpoint
url = "http://127.0.0.1:1234/v1/chat/completions"

# Define your question
question = "What can you tell me about large language models?"

# Prepare the payload
payload = {
    "messages": [
        {"role": "user", "content": question}
    ],
    "temperature": 0.7,
    "max_tokens": 500
}

# Set headers for the API request
headers = {
    "Content-Type": "application/json"
}
```