

LAMPIRAN A

SKRIP OTOMATISASI PENGUJIAN (RTX 3060)

A. benchmark_qwen_reasoning.py

```
import time
import re
import csv
import datasets
import subprocess
import os
import threading
import sys
from llama_cpp import Llama

def print_progress(current, total, label, width=30):
    pct = current / total
    filled = int(width * pct)
    bar = "\u2588" * filled + "\u2591" * (width - filled)
    sys.stdout.write(f"\r {label}: [{bar}] {current}/{total} ({pct*100:.1f}%)")
    sys.stdout.flush()
    if current == total:
        sys.stdout.write("\n")

def get_vram_usage():
    try:
        result = subprocess.check_output(
            ['nvidia-smi', '--query-gpu=memory.used', '--format=csv,nounits,noheader'],
            encoding='utf-8'
        )
        return int(result.strip().split('\n')[0]) / 1024.0
    except Exception:
        return 0.0

def generate_and_measure(llm, prompt, max_tokens, stop_words=None,
temperature=0.0):
    prompt_tokens = len(llm.tokenize(prompt.encode('utf-8')))
    t0 = time.time()
    generator = llm(prompt, max_tokens=max_tokens, temperature=temperature,
stop=stop_words, stream=True)
    generated_text = ""
    t1 = None
```

```

    for chunk in generator:
        if t1 is None:
            t1 = time.time()
            chunk_text = chunk["choices"][0]["text"]
            generated_text += chunk_text
            if re.search(r'final\s*answer\s*[:\-\]\s*[ABCD]\b', generated_text,
flags=re.IGNORECASE):
                break
        t2 = time.time()
        if t1 is None:
            t1 = t2 = t0
        completion_tokens = len(llm.tokenize(generated_text.encode('utf-8'))) if
generated_text else 0
        prompt_time = max(t1 - t0, 0.0001)
        gen_time = max(t2 - t1, 0.0001)
        return generated_text, prompt_tokens, completion_tokens, prompt_time, gen_time

def build_chatml_prompt(user_text):
    return (
        "<|im_start|>system\n"
        "You are a precise reasoning assistant.\n"
        "<|im_end|>\n"
        "<|im_start|>user\n"
        f"{user_text}\n"
        "<|im_end|>\n"
        "<|im_start|>assistant\n"
    )

QWEN_MODELS = [
    {"name": "Qwen 3.5 2B - F16", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-
F16.gguf"},
    {"name": "Qwen 3.5 2B - Q3_K_M", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-
Q3_K_M.gguf"},
    {"name": "Qwen 3.5 2B - Q4_K_M", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-
Q4_K_M.gguf"},
    {"name": "Qwen 3.5 2B - Q5_K_M", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-
Q5_K_M.gguf"},
]

def extract_mmlu_choice(text):
    if not text:
        return None
    text = re.sub(r'<think>.*?</think>', '', text, flags=re.DOTALL | re.IGNORECASE)
    for pat in [
        r'final\s*answer\s*[:\-\]\s*([ABCD])\b',
        r'\*\*final\s*answer\*\*\s*[:\-\]\s*([ABCD])\b',
        r'answer\s*[:\-\]\s*([ABCD])\b',
        r'\boption\s*([ABCD])\b',
    ]

```

```

        r'^\s*([ABCD])\s*$',
]:
    m = re.search(pat, text, flags=re.IGNORECASE | re.MULTILINE)
    if m:
        return m.group(1).upper()
    standalone = re.findall(r'\b([ABCD])\b', text.upper())
    return standalone[-1] if standalone else None

def normalize_numeric_answer(raw):
    if raw is None:
        return None
    cleaned = str(raw).strip().replace(", ", "")
    cleaned = re.sub(r'\s+', ' ', cleaned)
    m = re.search(r'-?\d+(?:\.\d+)?', cleaned)
    if not m:
        return None
    num_str = m.group(0)
    try:
        val = float(num_str)
        return str(int(val)) if val.is_integer() else str(val)
    except Exception:
        return num_str

def extract_gsm8k_answer(text):
    if not text:
        return None
    text = re.sub(r'<think>.*?</think>', '', text, flags=re.DOTALL | re.IGNORECASE)
    for pat in [
        r'####\s*([-+]?\d[\d,]*(?:\.\d+)?)',
        r'\\boxed{\s*([-+]?\d[\d,]*(?:\.\d+)?)\s*}',
        r'final\s*answer\s*[:\-\]\s*([-+]?\d[\d,]*(?:\.\d+)?)',
        r'answer\s+is\s+([-+]?\d[\d,]*(?:\.\d+)?)',
    ]:
        m = re.search(pat, text, flags=re.IGNORECASE)
        if m:
            return normalize_numeric_answer(m.group(1))
    nums = re.findall(r'([-+]?\d[\d,]*(?:\.\d+)?', text)
    return normalize_numeric_answer(nums[-1]) if nums else None

def save_csv(filepath, rows):
    if not rows:
        return
    fieldnames = rows[0].keys()
    file_exists = os.path.exists(filepath)
    with open(filepath, "a", newline="", encoding="utf-8") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        if not file_exists:
            writer.writeheader()

```

```

        writer.writerows(rows)

print("=" * 60)
print(" QWEN 3.5 REASONING BENCHMARK (4 datasets x 100 samples)")
print("=" * 60)

NUM_SAMPLES = 100

print("\nMemuat datasets...")
ds_mmlu = datasets.load_dataset("cais/mmlu", "global_facts",
split=f"test[:{NUM_SAMPLES}]")
ds_gsm8k = datasets.load_dataset("gsm8k", "main", split=f"test[:{NUM_SAMPLES}]")
ds_humaneval = datasets.load_dataset("openai_humaneval",
split=f"test[:{NUM_SAMPLES}]")
ds_mtbench = [
    "Write a short, thrilling story about a time traveler who accidentally alters the past.",
    "Compose a persuasive email to your boss asking for a team budget increase for new AI tools.",
    "Write a poem about the quiet beauty of a snowy midnight in a bustling city.",
    "Draft a speech for a high school graduation focusing on resilience and the future.",
    "Create a detailed fictional recipe for a magic potion that cures laziness, including ingredients and steps.",
    "Describe a futuristic city in the year 2150 using rich, sensory details.",
    "Write a dialogue between two old friends who haven't seen each other in 10 years meeting at a cafe.",
    "Explain the concept of black holes to a 5-year-old child using a creative analogy.",
    "Write a motivational manifesto for artists struggling with creative block.",
    "Design a pitch for a sci-fi movie where humans must negotiate with an advanced alien AI.",
    "Write a humorous review of a highly advanced alien blender.",
    "Write an instruction manual on how to tame a baby dragon.",
    "Create a detailed workout plan for a superhero whose power is super speed.",
    "Write a letter to your past self exactly 10 years ago giving exactly one piece of cryptic advice.",
    "Describe the feeling of tasting your favorite food for the first time without naming the food.",
    "Write a short news report about the sudden disappearance of gravity for 5 minutes.",
    "Draft a debate opening argument arguing that weekends should be three days long.",
    "Write a sonnet about the anxiety of forgetting your password.",
    "Create an advertisement script for a time machine that only travels one hour into the future.",
    "Write an FAQ section for a hotel situated underwater.",

```

"Imagine a world where animals can talk. Write a short complaint from a pigeon about city life.",

"Produce a 100-word summary of a fictional movie about rogue washing machines.",

"Write an elaborate excuse note for being late to work because of a temporal anomaly.",

"Compose a soothing bedtime story for an insomniac robot.",

"Describe a magical garden where flowers chime like glass bells when the wind blows.",

"Write a mystery story opening set in a library that only appears at midnight.",

"Create a travel guide for a planet made entirely of desserts.",

"Write a resignation letter from a villain who is tired of being defeated.",

"Describe the last day on Earth from the perspective of a houseplant.",

"Write a children's book about a cloud who is afraid of heights.",

"Compose a love letter from a lighthouse to the sea.",

"Write a news headline and article about the discovery of a second moon.",

"Create a menu for a restaurant that serves emotions as dishes.",

"Write a monologue from the perspective of the last working payphone.",

"Describe a world where music is illegal and someone discovers an old vinyl record.",

"Write a short horror story that takes place entirely in an elevator.",

"Create a job posting for a dragon tamer at a medieval startup.",

"Write a diary entry from an astronaut on their first day on Mars.",

"Compose a speech by a robot running for president.",

"Write a fairy tale where the villain wins but learns a lesson.",

"Describe the invention of fire from the perspective of a caveman philosopher.",

"Write a breakup letter from a superhero to their sidekick.",

"Create a survival guide for living inside a video game.",

"Write a poem about the loneliness of a forgotten umbrella.",

"Describe a conversation between the sun and the moon during an eclipse.",

"Write a short play about two AIs falling in love.",

"Create a recipe for a dish that tastes like nostalgia.",

"Write a letter from the ocean to humanity about pollution.",

"Describe a world where dreams are taxed by the government.",

"Write a comedy sketch about a time traveler stuck in a Monday morning meeting.",

"Compose a lullaby for a baby star being born in a nebula.",

"Write a detective story where the mystery is a missing Tuesday.",

"Create a user manual for a magic wand with too many settings.",

"Write a short story about a painter who can only paint the future.",

"Describe the last library on Earth after all books have been digitized.",

"Write a motivational speech from a caterpillar about to become a butterfly.",

"Create a news report about the first robot to win a cooking competition.",

"Write a poem from the perspective of a shadow tired of following its owner.",

"Describe a city where everyone speaks only in questions.",

"Write a short story about a door that leads to a different decade each time it opens.",

"Compose a thank-you note from the moon to the tides.",

"Write a horror story about a GPS that gives existential directions.",

"Create a travel itinerary for a ghost visiting their hometown.",

"Write a short story about the last human who remembers how to read a map.",

"Describe a world where laughter is a form of currency.",

"Write a poem about a robot learning to cry for the first time.",

"Create a bedtime story about a star that is afraid of the dark.",

"Write a short story about a musician who can only play songs from the future.",

"Describe the first day of school for a young wizard.",

"Write a letter from a retired superhero to their younger self.",

"Create a recipe for a potion that makes you speak only in rhymes.",

"Write a short story about a library where books choose their readers.",

"Describe a world where every lie creates a small earthquake.",

"Write a poem about the silence between two people who used to be in love.",

"Create a news article about the discovery that clouds are actually living creatures.",

"Write a short story about a clock that runs backwards in a small town.",

"Describe the last conversation between the Earth and the Sun before the Sun goes out.",

"Write a comedy sketch about a wizard who keeps accidentally summoning the wrong things.",

"Create a survival guide for surviving a Monday morning.",

"Write a short story about a painter whose paintings come to life at night.",

"Describe a world where everyone is born with a visible countdown timer.",

"Write a poem about the courage it takes to start over.",

"Create a job interview transcript for the position of tooth fairy.",

"Write a short story about a scientist who discovers that gravity is just a habit.",

"Describe the first meeting between a human and a sentient cloud.",

"Write a letter from a forgotten toy in the attic to its former owner.",

"Create a news report about a town where everyone suddenly starts speaking backwards.",

"Write a short story about a chef who can taste memories in food.",

"Describe a world where music physically builds structures as it plays.",

"Write a poem about the bravery of a single candle in a dark room.",

"Create a travel guide for the inside of a human mind.",

"Write a short story about a mirror that shows you who you could have been.",

"Describe the retirement party of the last working typewriter.",

"Write a comedy sketch about two clouds arguing over whose rain is better.",

"Create a recipe for a cake that makes you temporarily invisible.",

"Write a short story about a cartographer who maps emotions instead of places.",

"Describe a world where every person has a theme song that plays when they enter a room.",

"Write a poem about the weight of an unspoken apology.",

```

        "Create a news article about the first city built entirely underground.",
        "Write a short story about a translator who can speak the language of
silence.",
][:25]

tabel_skripsi = []
peak_vram_value = 0.0
peak_vram_thread_stop = False

def track_peak_vram():
    global peak_vram_value, peak_vram_thread_stop
    while not peak_vram_thread_stop:
        v = get_vram_usage()
        if v > peak_vram_value:
            peak_vram_value = v
        time.sleep(0.5)

for model_info in QWEN_MODELS:
    model_name = model_info["name"]
    model_path = model_info["path"]

    print(f"\n\n{'='*50}")
    print(f" MODEL: {model_name}")
    print(f"{'='*50}")

    vram_before = get_vram_usage()
    peak_vram_value = vram_before
    peak_vram_thread_stop = False
    vram_thread = threading.Thread(target=track_peak_vram)
    vram_thread.start()

    llm = Llama(model_path=model_path, n_gpu_layers=-1, verbose=False, n_ctx=4096,
n_threads=6, flash_attn=True)
    vram_after = get_vram_usage()
    net_vram_usage = vram_after - vram_before

    # --- 1. MMLU ---
    print(f"\n [1/4] MMLU - {NUM_SAMPLES} samples")
    correct_mmlu = 0; valid_mmlu = 0
    tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
    for i, sample in enumerate(ds_mmlu, 1):
        question = sample.get('question')
        choices = sample.get('choices')
        answer_idx = sample.get('answer')
        if not (isinstance(question, str) and isinstance(choices, list) and
len(choices) >= 4 and isinstance(answer_idx, int) and 0 <= answer_idx <= 3):
            print_progress(i, len(ds_mmlu), "MMLU"); continue
        valid_mmlu += 1

```

```

        options_text = "\n".join([f"{chr(65+j)} {c}" for j, c in
enumerate(choices[:4])])
        user_prompt = (
            "You are a strict and highly efficient reasoning assistant.\n"
            "Rule 1: Keep your step-by-step reasoning extremely concise (maximum 3
sentences).\n"
            "Rule 2: You MUST end your response with exactly one line in this
format: Final answer: <A/B/C/D>.\n\n"
            f"Question: {question}\n"
            f"Options:\n{options_text}\n"
            "Reasoning and Final answer:"
        )
        prompt = build_chatml_prompt(user_prompt)
        correct_letter = chr(65 + answer_idx)
        gen_text, pt, ct, ptime, ctime = generate_and_measure(
            llm, prompt, 512, stop_words=["\n\nQuestion:", "<|im_end|>",
"<|im_start|>"]
        )
        pred = extract_mmlu_choice(gen_text)
        if pred == correct_letter:
            correct_mmlu += 1
        tot_pt += pt; tot_ct += ct; tot_ptime += ptime; tot_ctime += ctime
        print_progress(i, len(ds_mmlu), "MMLU")

acc_mmlu = (correct_mmlu / max(valid_mmlu, 1)) * 100
skor_mmlu = f"{acc_mmlu:.2f}% ({correct_mmlu}/{valid_mmlu})"
tabel_skripsi.append({
    "Dataset": "MMLU", "Model": model_name, "Skor": skor_mmlu,
    "VRAM": f"{net_vram_usage:.2f}",
    "Prompt t/s": f"{tot_pt/tot_ptime:.2f}" if tot_ptime else "0",
    "Gen t/s": f"{tot_ct/tot_ctime:.2f}" if tot_ctime else "0"
})

# --- 2. GSM8K ---
print(f"\n [2/4] GSM8K - {NUM_SAMPLES} samples")
correct_gsm = 0; valid_gsm = 0
tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
few_shot = (
    "Question: Natalia sold clips to 48 of her friends in April. She sold half
as many clips in May. How many clips did she sell altogether in April and May?\n"
    "Reasoning: Natalia sold 48/2 = 24 clips in May. In total, she sold 48 + 24
= 72 clips.\n"
    "Final answer: 72\n\n"
    "Question: Weng earns $12 an hour for babysitting. She babysits for 50
minutes. How much does she earn?\n"
    "Reasoning: 50 minutes is 5/6 of an hour. So she earns 12 * 5/6 = 10
dollars.\n"
    "Final answer: 10\n\n"

```

```

    )
    for i, sample in enumerate(ds_gsm8k, 1):
        question = sample.get('question')
        gt_text = sample.get('answer')
        true_ans = extract_gsm8k_answer(gt_text) if isinstance(gt_text, str) else
None
        if not (isinstance(question, str) and true_ans):
            print_progress(i, len(ds_gsm8k), "GSM8K"); continue
        valid_gsm += 1
        user_prompt = (
            few_shot + f"Question: {question}\n"
            + "Rule 1: Keep your reasoning extremely concise (maximum 3
sentences).\n"
            + "Rule 2: End with exactly one line: Final answer: <number>\n"
            + "Reasoning:\n"
        )
        prompt = build_chatml_prompt(user_prompt)
        gen_text, pt, ct, ptime, ctime = generate_and_measure(
            llm, prompt, 768, stop_words=["\n\nQuestion:", "<|im_end|>",
"<|im_start|>"]
        )
        pred = extract_gsm8k_answer(gen_text)
        if pred == true_ans:
            correct_gsm += 1
        tot_pt += pt; tot_ct += ct; tot_ptime += ptime; tot_ctype += ctime
        print_progress(i, len(ds_gsm8k), "GSM8K")

    acc_gsm = (correct_gsm / max(valid_gsm, 1)) * 100
    skor_gsm = f"{acc_gsm:.2f}% ({correct_gsm}/{valid_gsm})"
    tabel_skripsi.append({
        "Dataset": "GSM8K", "Model": model_name, "Skor": skor_gsm,
        "VRAM": f"{net_vram_usage:.2f}",
        "Prompt t/s": f"{tot_pt/tot_ptime:.2f}" if tot_ptime else "0",
        "Gen t/s": f"{tot_ct/tot_ctype:.2f}" if tot_ctype else "0"
    })

# --- 3. HumanEval ---
print(f"\n [3/4] HumanEval - {NUM_SAMPLES} samples")
correct_code = 0
tot_pt, tot_ct, tot_ptime, tot_ctype = 0, 0, 0, 0
for i, sample in enumerate(ds_humaneval, 1):
    prompt_code, test_case, entry = sample["prompt"], sample["test"],
sample["entry_point"]
    gen_text, pt, ct, ptime, ctime = generate_and_measure(
        llm, prompt_code, 200, stop_words=["\n\nclass", "\n\ndef", "\n#",
"\nprint"]
    )

```

```

        full_code = prompt_code + gen_text + "\n\n" + test_case +
f"\ncheck({entry})\n"
        try:
            old_stdout = sys.stdout
            sys.stdout = open(os.devnull, 'w')
            exec(full_code, {})
            sys.stdout.close()
            sys.stdout = old_stdout
            correct_code += 1
        except Exception:
            sys.stdout.close()
            sys.stdout = old_stdout

    tot_pt += pt; tot_ct += ct; tot_ptime += ptime; tot_ctime += ctime
    print_progress(i, len(ds_humaneval), "HumanEval")

acc_code = (correct_code / len(ds_humaneval)) * 100
skor_code = f"{acc_code:.2f}% ({correct_code}/{len(ds_humaneval)})"
tabel_skripsi.append({
    "Dataset": "HumanEval", "Model": model_name, "Skor": skor_code,
    "VRAM": f"{net_vram_usage:.2f}",
    "Prompt t/s": f"{tot_pt/tot_ptime:.2f}" if tot_ptime else "0",
    "Gen t/s": f"{tot_ct/tot_ctime:.2f}" if tot_ctime else "0"
})

# --- 4. MT-Bench ---
print(f"\n [4/4] MT-Bench - {NUM_SAMPLES} samples")
total_ttr, total_words = 0, 0
tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
for i, prompt_text in enumerate(ds_mtbench, 1):
    gen_text, pt, ct, ptime, ctime = generate_and_measure(
        llm, f"Instruction: {prompt_text}\nResponse:\n", 300, temperature=0.7
    )
    words = re.findall(r'\b\w+\b', gen_text.lower())
    if words:
        total_ttr += len(set(words)) / len(words)
        total_words += len(words)
    tot_pt += pt; tot_ct += ct; tot_ptime += ptime; tot_ctime += ctime
    print_progress(i, len(ds_mtbench), "MT-Bench")

avg_ttr = total_ttr / len(ds_mtbench) if ds_mtbench else 0
avg_w = total_words / len(ds_mtbench) if ds_mtbench else 0
skor_mt = f"TTR: {avg_ttr:.3f} (W:{avg_w:.0f})"
tabel_skripsi.append({
    "Dataset": "MT-Bench", "Model": model_name, "Skor": skor_mt,
    "VRAM": f"{net_vram_usage:.2f}",
    "Prompt t/s": f"{tot_pt/tot_ptime:.2f}" if tot_ptime else "0",
    "Gen t/s": f"{tot_ct/tot_ctime:.2f}" if tot_ctime else "0"
})

```

```

    peak_vram_thread_stop = True
    vram_thread.join()
    print(f"\n >> {model_name} done. VRAM: ~{net_vram_usage:.2f} GB (peak:
~{peak_vram_value:.2f} GB)")
    del llm

# --- Simpan ke CSV ---
csv_path = "benchmark_qwen_results.csv"
save_csv(csv_path, tabel_skripsi)
print(f"\n[CSV] Hasil disimpan ke: {csv_path}")

# --- Print tabel ---
print("\n\n" + "=" * 120)
print(" HASIL BENCHMARK QWEN 3.5 REASONING (4 datasets)")
print("=" * 120)
print(f"| {'Dataset':<10} | {'Model':<22} | {'Skor':<22} | {'VRAM (GB)':<10} |
{'Prompt t/s':<11} | {'Gen t/s':<10} |")
print("|" + "-" * 12 + "|" + "-" * 24 + "|" + "-" * 24 + "|" + "-" * 12 + "|" + "-"
* 13 + "|" + "-" * 12 + "|")
for r in tabel_skripsi:
    print(f"| {r['Dataset']:<10} | {r['Model']:<22} | {r['Skor']:<22} |
{r['VRAM']:<10} | {r['Prompt t/s']:<11} | {r['Gen t/s']:<10} |")
print("=" * 120)

```



B. benchmark_pc.py (untuk LFM 2.5)

```

import time
import re
import json
import datasets
import subprocess
import os
import threading
import sys
from llama_cpp import Llama

def print_progress(current, total, label, width=30):
    pct = current / total
    filled = int(width * pct)
    bar = "\u2588" * filled + "\u2591" * (width - filled)
    sys.stdout.write(f"\r {label}: [{bar}] {current}/{total} ({pct*100:.1f}%")
    sys.stdout.flush()
    if current == total:
        sys.stdout.write("\n")

def get_vram_usage():
    try:
        result = subprocess.check_output(
            ['nvidia-smi', '--query-gpu=memory.used', '--format=csv,nounits,noheader'],
            encoding='utf-8'
        )
        return int(result.strip().split('\n')[0]) / 1024.0
    except Exception:
        return 0.0

def generate_and_measure(llm, prompt, max_tokens, stop_words=None, temperature=0.0,
dataset_type=None):
    prompt_tokens = len(llm.tokenize(prompt.encode('utf-8')))
    t0 = time.time()
    generator = llm(prompt, max_tokens=max_tokens, temperature=temperature,
stop=stop_words, stream=True)
    generated_text = ""
    t1 = None
    t2 = t0

    for chunk in generator:
        if t1 is None:
            t1 = time.time()

        chunk_text = chunk["choices"][0]["text"]
        generated_text += chunk_text

```

```

# --- SMART EARLY STOPPING ---
# Berhenti paksa jika jawaban final sudah ditemukan untuk menghemat waktu
komputasi.
# Mendukung banyak format jawaban (penting untuk model base/non-thinking
seperti LFM).
if dataset_type == "mmlu":
    if re.search(
        r'final\s*answer\s*[:\-\]\s*\(\?\s*[ABCD]\b'          # Final answer: B
        r'|\banswer\s+is\s*[:\-\]\s*\(\?\s*[ABCD]\b'        # answer is B /
        r'|\bcorrect\s+(\?:answer|option|choice)\s+is\s*[:\-\]  # correct option is B
    )?\s*\(\?\s*[ABCD]\b', generated_text, flags=re.IGNORECASE):
        break
    elif dataset_type == "gsm8k":
        if re.search(r'final\s*answer\s*[:\-\]\s*[-+]?d[\d,]*(?:\.\d+)?',
generated_text, flags=re.IGNORECASE):
            break
# ----
t2 = time.time()
completion_tokens = len(llm.tokenize(generated_text.encode('utf-8'))) if
generated_text else 0
prompt_time = max(t1 - t0, 0.0001)
gen_time = max(t2 - t1, 0.0001)

return generated_text, prompt_tokens, completion_tokens, prompt_time, gen_time

def build_chatml_prompt(user_text):
    return (
        "<|im_start|>system\n"
        "You are a precise reasoning assistant.\n"
        "<|im_end|>\n"
        "<|im_start|>user\n"
        f"{user_text}\n"
        "<|im_end|>\n"
        "<|im_start|>assistant\n"
    )

def adapt_prompt_for_model(model_name, user_text):
    if "qwen" in model_name.lower():
        return build_chatml_prompt(user_text)
    return user_text

ALL_MODELS = [

```

```

        {"name": "LFM 1.2B - F16",      "path": "./models/LFM2.5-1.2B-Base/LFM2.5-1.2B-Base-F16.gguf"},
        {"name": "LFM 1.2B - Q3_K_M",  "path": "./models/LFM2.5-1.2B-Base/LFM2.5-1.2B-Base-Q3_K_M.gguf"},
        {"name": "LFM 1.2B - Q4_K_M",  "path": "./models/LFM2.5-1.2B-Base/LFM2.5-1.2B-Base-Q4_K_M.gguf"},
        {"name": "LFM 1.2B - Q5_K_M",  "path": "./models/LFM2.5-1.2B-Base/LFM2.5-1.2B-Base-Q5_K_M.gguf"},
        {"name": "Qwen 3.5 2B - F16",   "path": "./models/Qwen3.5-2B/Qwen3.5-2B-F16.gguf"},
        {"name": "Qwen 3.5 2B - Q3_K_M", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-Q3_K_M.gguf"},
        {"name": "Qwen 3.5 2B - Q4_K_M", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-Q4_K_M.gguf"},
        {"name": "Qwen 3.5 2B - Q5_K_M", "path": "./models/Qwen3.5-2B/Qwen3.5-2B-Q5_K_M.gguf"},
    ]

ALL_DATASETS = [
    {"key": "mmlu",      "label": "MMLU",      "kategori": "Factual Retrieval"},
    {"key": "gsm8k",     "label": "GSM8K",     "kategori": "Arithmetic Reasoning"},
    {"key": "humaneval", "label": "HumanEval", "kategori": "Code Synthesis"},
    {"key": "mtbench",   "label": "MT-Bench",  "kategori": "Creative Generation"},
]

def parse_selection(text, max_val):
    indices = []
    for part in text.split(","):
        part = part.strip()
        if part.isdigit():
            idx = int(part)
            if 1 <= idx <= max_val and idx not in indices:
                indices.append(idx)
    return indices

def extract_mmlu_choice(text, strict=True):
    if not text:
        return None
    text = re.sub(r' bạn.*?\u200b', '', text, flags=re.DOTALL | re.IGNORECASE)
    strict_patterns = [
        r'final\s*answer\s*[:\-\]\s*([ABCD])\b',
        r'\*\s*final\s*answer\s*\*\s*[:\-\]\s*([ABCD])\b',
    ]
    for pat in strict_patterns:
        m = re.search(pat, text, flags=re.IGNORECASE | re.MULTILINE)
        if m:
            return m.group(1).upper()
    if strict:

```

```

        return None
    fallback_patterns = [
        r'answer\s*[:\-\]\s*([ABCD])\b',
        r'\boption\s*([ABCD])\b',
        r'^\s*([ABCD])\s*$',
        r'the\s+answer\s+is\s*[:\-\]\s*([ABCD])\b',
        r'correct\s+(?:answer|option|choice)\s+is\s*[:\-\]\s*([ABCD])\b',
        r'therefore[, \s]+([ABCD])\b',
        r'so\s+the\s+answer\s+is\s*[:\-\]\s*([ABCD])\b',
        r'final\s*[:\-\]\s*([ABCD])\b',
    ]
    for pat in fallback_patterns:
        m = re.search(pat, text, flags=re.IGNORECASE | re.MULTILINE)
        if m:
            return m.group(1).upper()
    # Prefer letters that appear near the end of the response
    standalone = re.findall(r'\b([ABCD])\b', text.upper())
    if standalone:
        # Return the last letter found (closest to end of response)
        return standalone[-1]
    return None

def normalize_numeric_answer(raw):
    if raw is None:
        return None
    cleaned = str(raw).strip().replace(",", "")
    cleaned = re.sub(r'\s+', ' ', cleaned)
    m = re.search(r'-?\d+(?:\.\d+)?', cleaned)
    if not m:
        return None
    num_str = m.group(0)
    try:
        val = float(num_str)
        if val.is_integer():
            return str(int(val))
        return str(val)
    except Exception:
        return num_str

def extract_gsm8k_answer(text, strict=True):
    if not text:
        return None
    text = re.sub(r'<think>.*?</think>', '', text, flags=re.DOTALL | re.IGNORECASE)
    tagged = re.search(r'####\s*([-\+]?d[\d,]*(?:\.\d+)?)', text)
    if tagged:
        return normalize_numeric_answer(tagged.group(1))
    boxed = re.search(r'\\boxed\\{\\s*([-\+]?d[\d,]*(?:\.\d+)?)\s*\\}', text)
    if boxed:

```

```

        return normalize_numeric_answer(boxed.group(1))
    final_labeled = re.search(r'final\s*answer\s*[:\-\]\s*([-\+]?[d[\d,]]*(?:[.\d+])?)', text, flags=re.IGNORECASE)
    if final_labeled:
        return normalize_numeric_answer(final_labeled.group(1))
    if strict:
        return None
    answer_is = re.search(r'answer\s+is\s+([-\+]?[d[\d,]]*(?:[.\d+])?)', text, flags=re.IGNORECASE)
    if answer_is:
        return normalize_numeric_answer(answer_is.group(1))
    nums = re.findall(r'[-+]?[d[\d,]]*(?:[.\d+])?', text)
    return normalize_numeric_answer(nums[-1]) if nums else None

def append_jsonl(path, data):
    try:
        with open(path, "a", encoding="utf-8") as f:
            f.write(json.dumps(data, ensure_ascii=False) + "\n")
    except Exception:
        pass

def format_score(base_str, **kwargs):
    return base_str

def append_csv(path, rows):
    if not rows:
        return
    import csv
    fieldnames = list(rows[0].keys())
    file_exists = os.path.exists(path)
    with open(path, "a", newline="", encoding="utf-8") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        if not file_exists:
            writer.writeheader()
        writer.writerows(rows)

def pilih_model(prompt_header="PILIH MODEL"):
    print("=" * 60)
    print(f" {prompt_header}")
    print("=" * 60)
    for i, m in enumerate(ALL_MODELS, 1):
        print(f" {i}. {m['name']}")
    print("=" * 60)
    while True:
        raw = input("Masukkan nomor model (pisah koma, contoh: 1,3,6): ").strip()
        selected = parse_selection(raw, len(ALL_MODELS))
        if selected:
            break

```

```

        print(" Input tidak valid, coba lagi.")
    chosen = [ALL_MODELS[i - 1] for i in selected]
    print(f"\n>> Model dipilih: {'', '.join(m['name'] for m in chosen)}\n")
    return chosen

print("=" * 60)
print(" PILIH JENIS BENCHMARK / MODE")
print("=" * 60)
print(" 1. Multi-Dataset Benchmark (MMLU, GSM8K, HumanEval, MT-Bench)")
print(" 2. Perplexity (PPL) - WikiText-2")
print(" 3. Chat Only Mode")
print("=" * 60)

benchmark_mode = ""
while benchmark_mode not in ["1", "2", "3"]:
    benchmark_mode = input("Masukkan pilihan (1/2/3): ").strip()

print()

peak_vram_value = 0.0
peak_vram_thread_stop = False

def track_peak_vram():
    global peak_vram_value, peak_vram_thread_stop
    while not peak_vram_thread_stop:
        v = get_vram_usage()
        if v > peak_vram_value:
            peak_vram_value = v
        time.sleep(0.5)

if benchmark_mode == "2":
    models_to_test = pilih_model("PILIH MODEL UNTUK PPL")

    print("=" * 50)
    print(" BENCHMARK PERPLEXITY (PPL) - WikiText-2")
    print("=" * 50)

    ppl_binary = "./build/bin/llama-perplexity"
    ppl_dataset = "wikitext-2-raw/wiki.test.raw"
    ppl_results = []

    for model_info in models_to_test:
        model_name = model_info["name"]
        model_path = model_info["path"]

        print(f"\n{' '*50}")
        print(f" PPL: {model_name}")
        print(f"{' '*50}")

```

```

vram_before = get_vram_usage()
peak_vram_value = vram_before
peak_vram_thread_stop = False
vram_thread = threading.Thread(target=track_peak_vram)
vram_thread.start()

cmd = [ppl_binary, "-m", model_path, "-f", ppl_dataset, "-c", "512", "-n", "999"]
print(f" CMD: {' '.join(cmd)}\n")

process = subprocess.Popen(cmd, stdout=subprocess.PIPE,
stderr=subprocess.STDOUT, text=True, errors='replace')

final_ppl = None
for line in process.stdout:
    sys.stdout.write(line)
    sys.stdout.flush()
    match = re.search(r'Final estimate: PPL = ([\d.]+)', line)
    if match:
        final_ppl = float(match.group(1))

process.wait()
vram_after = get_vram_usage()
peak_vram_thread_stop = True
vram_thread.join()

ppl_results.append({
    "Model": model_name,
    "PPL": f"{final_ppl:.4f}" if final_ppl else "N/A",
    "Peak VRAM": f"{peak_vram_value:.2f}"
})

print(f"\n >> {model_name} - Final PPL: {final_ppl if final_ppl else 'N/A'}")
print(f" >> Peak VRAM: ~{peak_vram_value:.2f} GB")

print(f"\n\n{'='*70}")
print(" TABEL HASIL PERPLEXITY (PPL)")
print("=" * 70)
print(f"| {'Model':<25} | {'Perplexity (PPL)':<20} | {'Peak VRAM (GB)':<15} |")
print("|" + "-" * 27 + "|" + "-" * 22 + "|" + "-" * 17 + "|")
for r in ppl_results:
    print(f"| {r['Model']:<25} | {r['PPL']:<20} | {r['Peak VRAM']:<15} |")
print("=" * 70)

elif benchmark_mode == "3":
    models_to_test = pilih_model("PILIH MODEL UNTUK CHAT")

```

```

model_info = models_to_test[0]
model_name = model_info["name"]
model_path = model_info["path"]
is_qwen = "qwen" in model_name.lower()

print(f"\n[{model_name}] Sedang memuat model...")
# Dioptimasi dengan n_threads=6 dan flash_attn=True
llm = Llama(model_path=model_path, n_gpu_layers=-1, verbose=False, n_ctx=4096,
n_threads=6, flash_attn=True)

print("=" * 60)
print(f" CHAT MODE AKTIF: {model_name}")
print(" (Ketik 'keluar', 'exit', atau 'quit' untuk berhenti)")
print("=" * 60)

chat_history = []
MAX_HISTORY = 5

while True:
    user_text = input("\nAnda: ").strip()
    if user_text.lower() in ['keluar', 'exit', 'quit']:
        print("Mengakhiri sesi chat...")
        break
    if not user_text:
        continue

    if is_qwen:
        chat_history.append(f"<|im_start|>user\n{user_text}\n<|im_end|>\n<|im_s
tart|>assistant\n")
        if len(chat_history) > MAX_HISTORY:
            chat_history.pop(0)

        full_prompt = "<|im_start|>system\nYou are a helpful
assistant.\n<|im_end|>\n" + "".join(chat_history)
        stop_words = ["<|im_end|>", "<|im_start|>"]
    else:
        chat_history.append(f"User: {user_text}\nAssistant: ")
        if len(chat_history) > MAX_HISTORY:
            chat_history.pop(0)

        full_prompt = "".join(chat_history)
        stop_words = ["User:", "\nUser:", "<|endoftext|>"]

    print(f"{model_name}: ", end="", flush=True)

    generator = llm(
        full_prompt,
        max_tokens=1524,

```

```

        temperature=0.7,
        stop=stop_words,
        stream=True
    )

    response_text = ""
    for chunk in generator:
        text = chunk["choices"][0]["text"]
        print(text, end="", flush=True)
        response_text += text
    print()

    if is_qwen:
        chat_history[-1] = chat_history[-1] + response_text + "\n<|im_end|>\n"
    else:
        chat_history[-1] = chat_history[-1] + response_text + "\n"

    del llm

else:
    print("=" * 60)
    print(" PILIH DATASET BENCHMARK")
    print("=" * 60)
    for i, ds in enumerate(ALL_DATASETS, 1):
        print(f" {i}. {ds['label']} ({ds['kategori']})")
    print("=" * 60)
    while True:
        raw = input("Masukkan nomor dataset (pisah koma, contoh: 1,2,3): ").strip()
        selected_ds_idx = parse_selection(raw, len(ALL_DATASETS))
        if selected_ds_idx:
            break
        print(" Input tidak valid, coba lagi.")
    chosen_datasets = [ALL_DATASETS[i - 1] for i in selected_ds_idx]
    print(f"\n>> Dataset dipilih: {' '.join(d['label'] for d in
chosen_datasets)}\n")

    models_to_test = pilih_model("PILIH MODEL UNTUK BENCHMARK")

    NUM_SAMPLES = 100
    ds_keys_needed = {d["key"] for d in chosen_datasets}

    print("=" * 50)
    print("MEMUAT DATASETS...")
    print("=" * 50)

    ds_mmlu = ds_gsm8k = ds_humaneval = ds_mtbench = None

    if "mmlu" in ds_keys_needed:

```

```

        ds_mmlu = datasets.load_dataset("cais/mmlu", "global_facts",
split=f"test[:{NUM_SAMPLES}]")
        if "gsm8k" in ds_keys_needed:
            ds_gsm8k = datasets.load_dataset("gsm8k", "main",
split=f"test[:{NUM_SAMPLES}]")
        if "humaneval" in ds_keys_needed:
            ds_humaneval = datasets.load_dataset("openai_humaneval",
split=f"test[:{NUM_SAMPLES}]")
        if "mtbench" in ds_keys_needed:
            ds_mtbench = [
                "Write a short, thrilling story about a time traveler who accidentally
alters the past.",
                "Compose a persuasive email to your boss asking for a team budget
increase for new AI tools.",
                "Write a poem about the quiet beauty of a snowy midnight in a
bustling city.",
                "Draft a speech for a high school graduation focusing on resilience
and the future.",
                "Create a detailed fictional recipe for a magic potion that cures
laziness, including ingredients and steps.",
                "Describe a futuristic city in the year 2150 using rich, sensory
details.",
                "Write a dialogue between two old friends who haven't seen each
other in 10 years meeting at a cafe.",
                "Explain the concept of black holes to a 5-year-old child using a
creative analogy.",
                "Write a motivational manifesto for artists struggling with
creative block.",
                "Design a pitch for a sci-fi movie where humans must negotiate with
an advanced alien AI.",
                "Write a humorous review of a highly advanced alien blender.",
                "Write an instruction manual on how to tame a baby dragon.",
                "Create a detailed workout plan for a superhero whose power is
super speed.",
                "Write a letter to your past self exactly 10 years ago giving
exactly one piece of cryptic advice.",
                "Describe the feeling of tasting your favorite food for the first
time without naming the food.",
                "Write a short news report about the sudden disappearance of
gravity for 5 minutes.",
                "Draft a debate opening argument arguing that weekends should be
three days long.",
                "Write a sonnet about the anxiety of forgetting your password.",
                "Create an advertisement script for a time machine that only
travels one hour into the future.",
                "Write an FAQ section for a hotel situated underwater.",
                "Imagine a world where animals can talk. Write a short complaint
from a pigeon about city life.",

```

"Produce a 100-word summary of a fictional movie about rogue washing machines.",

"Write an elaborate excuse note for being late to work because of a temporal anomaly.",

"Compose a soothing bedtime story for an insomniac robot.",

"Describe a magical garden where flowers chime like glass bells when the wind blows.",

"Write a mystery story opening set in a library that only appears at midnight.",

"Create a travel guide for a planet made entirely of desserts.",

"Write a resignation letter from a villain who is tired of being defeated.",

"Describe the last day on Earth from the perspective of a houseplant.",

"Write a children's book about a cloud who is afraid of heights.",

"Compose a love letter from a lighthouse to the sea.",

"Write a news headline and article about the discovery of a second moon.",

"Create a menu for a restaurant that serves emotions as dishes.",

"Write a monologue from the perspective of the last working payphone.",

"Describe a world where music is illegal and someone discovers an old vinyl record.",

"Write a short horror story that takes place entirely in an elevator.",

"Create a job posting for a dragon tamer at a medieval startup.",

"Write a diary entry from an astronaut on their first day on Mars.",

"Compose a speech by a robot running for president.",

"Write a fairy tale where the villain wins but learns a lesson.",

"Describe the invention of fire from the perspective of a caveman philosopher.",

"Write a breakup letter from a superhero to their sidekick.",

"Create a survival guide for living inside a video game.",

"Write a poem about the loneliness of a forgotten umbrella.",

"Describe a conversation between the sun and the moon during an eclipse.",

"Write a short play about two AIs falling in love.",

"Create a recipe for a dish that tastes like nostalgia.",

"Write a letter from the ocean to humanity about pollution.",

"Describe a world where dreams are taxed by the government.",

"Write a comedy sketch about a time traveler stuck in a Monday morning meeting.",

"Compose a lullaby for a baby star being born in a nebula.",

"Write a detective story where the mystery is a missing Tuesday.",

"Create a user manual for a magic wand with too many settings.",

"Write a short story about a painter who can only paint the future.",

"Describe the last library on Earth after all books have been digitized.",
 "Write a motivational speech from a caterpillar about to become a butterfly.",
 "Create a news report about the first robot to win a cooking competition.",
 "Write a poem from the perspective of a shadow tired of following its owner.",
 "Describe a city where everyone speaks only in questions.",
 "Write a short story about a door that leads to a different decade each time it opens.",
 "Compose a thank-you note from the moon to the tides.",
 "Write a horror story about a GPS that gives existential directions.",
 "Create a travel itinerary for a ghost visiting their hometown.",
 "Write a short story about the last human who remembers how to read a map.",
 "Describe a world where laughter is a form of currency.",
 "Write a poem about a robot learning to cry for the first time.",
 "Create a bedtime story about a star that is afraid of the dark.",
 "Write a short story about a musician who can only play songs from the future.",
 "Describe the first day of school for a young wizard.",
 "Write a letter from a retired superhero to their younger self.",
 "Create a recipe for a potion that makes you speak only in rhymes.",
 "Write a short story about a library where books choose their readers.",
 "Describe a world where every lie creates a small earthquake.",
 "Write a poem about the silence between two people who used to be in love.",
 "Create a news article about the discovery that clouds are actually living creatures.",
 "Write a short story about a clock that runs backwards in a small town.",
 "Describe the last conversation between the Earth and the Sun before the Sun goes out.",
 "Write a comedy sketch about a wizard who keeps accidentally summoning the wrong things.",
 "Create a survival guide for surviving a Monday morning.",
 "Write a short story about a painter whose paintings come to life at night.",
 "Describe a world where everyone is born with a visible countdown timer.",
 "Write a poem about the courage it takes to start over.",
 "Create a job interview transcript for the position of tooth fairy.",

```

        "Write a short story about a scientist who discovers that gravity
is just a habit.",
        "Describe the first meeting between a human and a sentient cloud.",
        "Write a letter from a forgotten toy in the attic to its former
owner.",
        "Create a news report about a town where everyone suddenly starts
speaking backwards.",
        "Write a short story about a chef who can taste memories in food.",
        "Describe a world where music physically builds structures as it
plays.",
        "Write a poem about the bravery of a single candle in a dark
room.",
        "Create a travel guide for the inside of a human mind.",
        "Write a short story about a mirror that shows you who you could
have been.",
        "Describe the retirement party of the last working typewriter.",
        "Write a comedy sketch about two clouds arguing over whose rain is
better.",
        "Create a recipe for a cake that makes you temporarily invisible.",
        "Write a short story about a cartographer who maps emotions instead
of places.",
        "Describe a world where every person has a theme song that plays
when they enter a room.",
        "Write a poem about the weight of an unspoken apology.",
        "Create a news article about the first city built entirely
underground.",
        "Write a short story about a translator who can speak the language
of silence."
    ][:NUM_SAMPLES]

    tabel_skripsi = []
    total_datasets_count = len(chosen_datasets)

    debug_log_path = "benchmark_debug_predictions.jsonl"
    try:
        if os.path.exists(debug_log_path):
            os.remove(debug_log_path)
    except Exception:
        pass

    for model_info in models_to_test:
        model_name = model_info["name"]
        model_path = model_info["path"]
        is_qwen = "qwen" in model_name.lower()

        print(f"\n\n{'='*50}")
        print(f"  MENGUJI MODEL: {model_name}")
        print(f"{'='*50}")

```

```

vram_before = get_vram_usage()

peak_vram_value = vram_before
peak_vram_thread_stop = False
vram_thread = threading.Thread(target=track_peak_vram)
vram_thread.start()

# Dioptimasi dengan n_threads=6 (sesuai core fisik umum laptop) dan
flash_attn=True
# n_ctx=4096 untuk MMLU/GSM8K (prompt few-shot + soal panjang), MT-Bench
pakai default
llm = Llama(model_path=model_path, n_gpu_layers=-1, verbose=False,
n_ctx=4096, n_threads=6, flash_attn=True)

vram_after = get_vram_usage()
net_vram_usage = vram_after - vram_before

hasil_evaluasi = {}
ds_counter = 0

def simpan_ke_tabel(kategori, dataset, p_tokens, c_tokens, p_time, c_time,
skor_akhir):
    prompt_ts = p_tokens / p_time if p_time > 0 else 0
    gen_ts = c_tokens / c_time if c_time > 0 else 0
    tabel_skripsi.append({
        "Kategori": kategori, "Dataset": dataset, "Model": model_name,
        "Hasil Skor/Akurasi": skor_akhir,
        "VRAM": f"{net_vram_usage:.2f}",
        "Peak": "TBD",
        "Prompt t/s": f"{prompt_ts:.2f}", "Gen t/s": f"{gen_ts:.2f}"
    })

if "mmlu" in ds_keys_needed:
    ds_counter += 1
    print(f"\n [{ds_counter}/{total_datasets_count}] Factual Retrieval
(MMLU)")

    correct_mmlu = 0
    valid_mmlu = 0
    invalid_mmlu = 0
    mmlu_strict_miss = 0
    mmlu_fallback_hits = 0
    tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
    for i, sample in enumerate(ds_mmlu, 1):
        question = sample.get('question')
        choices = sample.get('choices')
        answer_idx = sample.get('answer')

```

```

        is_valid = isinstance(question, str) and question.strip() and
isinstance(choices, list) and len(choices) >= 4 and isinstance(answer_idx, int) and
0 <= answer_idx <= 3
        if not is_valid:
            invalid_mmlu += 1
            print_progress(i, len(ds_mmlu), "MMLU")
            continue

        valid_mmlu += 1
        options_text = "\n".join([f"{chr(65+j)} {c}" for j, c in
enumerate(choices[:4])])

        # Few-shot examples untuk membantu base model mengikuti format
        few_shot_mmlu = (
            "Question: What is the capital of France?\n"
            "Options:\nA) London\nB) Paris\nC) Berlin\nD) Madrid\n"
            "Reasoning: Paris is the capital city of France.\n"
            "Final answer: B\n\n"
            "Question: Which planet is known as the Red Planet?\n"
            "Options:\nA) Venus\nB) Mars\nC) Jupiter\nD) Saturn\n"
            "Reasoning: Mars appears red due to iron oxide on its
surface.\n"
            "Final answer: B\n\n"
            "Question: Who wrote the play Romeo and Juliet?\n"
            "Options:\nA) Charles Dickens\nB) William Shakespeare\nC) Jane
Austen\nD) Mark Twain\n"
            "Reasoning: William Shakespeare wrote the tragedy Romeo and
Juliet.\n"
            "Final answer: B\n\n"
        )

        # Optimasi Prompt: Memaksa reasoning singkat
        user_prompt = (
            "You are a strict and highly efficient reasoning assistant.\n"
            "Rule 1: Keep your step-by-step reasoning extremely concise
(maximum 3 sentences).\n"
            "Rule 2: You MUST end your response with exactly one line in
this format: Final answer: <A/B/C/D>.\n\n"
            f"{few_shot_mmlu}"
            f"Question: {question}\n"
            f"Options:\n{options_text}\n"
            "Reasoning and Final answer:"
        )

        prompt = adapt_prompt_for_model(model_name, user_prompt)
        correct_letter = chr(65 + answer_idx)

        gen_text, pt, ct, ptime, ctime = generate_and_measure(
            llm,

```

```

        prompt,
        512,
        stop_words=["<|im_end|>", "<|im_start|>"],
        dataset_type="mmlu"
    )

    pred_strict = extract_mmlu_choice(gen_text, strict=True)
    pred_fallback = extract_mmlu_choice(gen_text, strict=False)
    pred = pred_strict if pred_strict is not None else pred_fallback

    if pred_strict is None:
        mmlu_strict_miss += 1
        if pred_fallback is not None:
            mmlu_fallback_hits += 1

    if pred == correct_letter:
        correct_mmlu += 1

    if is_qwen and (i <= 20 or pred != correct_letter):
        append_jsonl(debug_log_path, {
            "dataset": "MMLU",
            "model": model_name,
            "sample_idx": i,
            "gold": correct_letter,
            "pred_strict": pred_strict,
            "pred_fallback": pred_fallback,
            "pred_used": pred,
            "is_correct": pred == correct_letter,
            "raw_output": gen_text,
            "question": question,
        })

    tot_pt += pt
    tot_ct += ct
    tot_ptime += ptime
    tot_ctime += ctime
    print_progress(i, len(ds_mmlu), "MMLU")

    denom_mmlu = valid_mmlu if valid_mmlu > 0 else 1
    acc_mmlu = (correct_mmlu / denom_mmlu) * 100
    skor_mmlu_str = format_score(f"{acc_mmlu:.2f}%
({correct_mmlu}/{valid_mmlu})", invalid=invalid_mmlu, strict_miss=mmlu_strict_miss,
fallback_hit=mmlu_fallback_hits)
    hasil_evaluasi['MMLU'] = skor_mmlu_str
    simpan_ke_tabel("Factual Retrieval", "MMLU", tot_pt, tot_ct, tot_ptime,
tot_ctime, skor_mmlu_str)

    if "gsm8k" in ds_keys_needed:
        ds_counter += 1

```

```

print(f"\n [{ds_counter}/{total_datasets_count}] Arithmetic Reasoning
(GSM8K)")

correct_gsm = 0
valid_gsm = 0
invalid_gsm = 0
gsm_strict_miss = 0
gsm_fallback_hits = 0
tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
few_shot = (
    "Question: Natalia sold clips to 48 of her friends in April. She
sold half as many clips in May. How many clips did she sell altogether in April and
May?\n"

    "Reasoning: Natalia sold  $48/2 = 24$  clips in May. In total, she sold
 $48 + 24 = 72$  clips.\n"

    "Final answer: 72\n\n"

    "Question: Weng earns $12 an hour for babysitting. She babysits for
50 minutes. How much does she earn?\n"

    "Reasoning: 50 minutes is  $5/6$  of an hour. So she earns  $12 * 5/6 =$ 
10 dollars.\n"

    "Final answer: 10\n\n"
)

for i, sample in enumerate(ds_gsm8k, 1):
    question = sample.get('question')
    gt_text = sample.get('answer')
    if not isinstance(question, str) or not question.strip() or not
isinstance(gt_text, str):
        invalid_gsm += 1
        print_progress(i, len(ds_gsm8k), "GSM8K")
        continue

    true_ans = extract_gsm8k_answer(gt_text, strict=False)
    if true_ans is None:
        invalid_gsm += 1
        print_progress(i, len(ds_gsm8k), "GSM8K")
        continue

    valid_gsm += 1

# Optimasi Prompt: Memaksa reasoning singkat
user_prompt = (
    few_shot
    + f"Question: {question}\n"
    + "Rule 1: Keep your reasoning extremely concise (maximum 3
sentences).\n"
    + "Rule 2: End with exactly one line: Final answer: <number>\n"
    + "Reasoning:\n"
)

prompt = adapt_prompt_for_model(model_name, user_prompt)

```

```

gen_text, pt, ct, ptime, ctime = generate_and_measure(
    llm,
    prompt,
    768,
    stop_words=["<|im_end|>", "<|im_start|>"],
    dataset_type="gsm8k"
)

pred_strict = extract_gsm8k_answer(gen_text, strict=True)
pred_fallback = extract_gsm8k_answer(gen_text, strict=False)
pred = pred_strict if pred_strict is not None else pred_fallback

if pred_strict is None:
    gsm_strict_miss += 1
    if pred_fallback is not None:
        gsm_fallback_hits += 1

if pred == true_ans:
    correct_gsm += 1

if is_qwen and (i <= 20 or pred != true_ans):
    append_jsonl(debug_log_path, {
        "dataset": "GSM8K",
        "model": model_name,
        "sample_idx": i,
        "gold": true_ans,
        "pred_strict": pred_strict,
        "pred_fallback": pred_fallback,
        "pred_used": pred,
        "is_correct": pred == true_ans,
        "raw_output": gen_text,
        "question": question,
    })
    tot_pt += pt
    tot_ct += ct
    tot_ptime += ptime
    tot_ctime += ctime
    print_progress(i, len(ds_gsm8k), "GSM8K")

denom_gsm = valid_gsm if valid_gsm > 0 else 1
acc_gsm = (correct_gsm / denom_gsm) * 100
skor_gsm_str = format_score(f"{acc_gsm:.2f}%
({correct_gsm}/{valid_gsm})", invalid=invalid_gsm, strict_miss=gsm_strict_miss,
fallback_hit=gsm_fallback_hits)
hasil_evaluasi['GSM8K'] = skor_gsm_str
simpan_ke_tabel("Arithmetic Reasoning", "GSM8K", tot_pt, tot_ct,
tot_ptime, tot_ctime, skor_gsm_str)

```

```

        if "humaneval" in ds_keys_needed:
            ds_counter += 1
            print(f"\n [{ds_counter}/{total_datasets_count}] Code Synthesis
(HumanEval)")
            correct_code = 0
            tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
            for i, sample in enumerate(ds_humaneval, 1):
                prompt, test_case, entry = sample["prompt"], sample["test"],
sample["entry_point"]
                gen_text, pt, ct, ptime, ctime = generate_and_measure(llm, prompt,
200, stop_words=["\n\nclass", "\n\ndef", "\n#", "\nprint"])
                full_code = prompt + gen_text + "\n\n" + test_case +
f"\ncheck({entry})\n"
                try:
                    old_stdout = sys.stdout
                    sys.stdout = open(os.devnull, 'w')
                    exec(full_code, {})
                    sys.stdout.close()
                    sys.stdout = old_stdout
                    correct_code += 1
                except Exception:
                    sys.stdout.close()
                    sys.stdout = old_stdout
                tot_pt+=pt; tot_ct+=ct; tot_ptime+=ptime; tot_ctime+=ctime
                print_progress(i, len(ds_humaneval), "HumanEval")
            acc_code = (correct_code/len(ds_humaneval))*100
            skor_code_str = format_score(f"{acc_code:.2f}%
({correct_code}/{len(ds_humaneval)})")
            hasil_evaluasi['HumanEval'] = skor_code_str
            simpan_ke_tabel("Code Synthesis", "HumanEval", tot_pt, tot_ct,
tot_ptime, tot_ctime, skor_code_str)

        if "mtbench" in ds_keys_needed:
            ds_counter += 1
            print(f"\n [{ds_counter}/{total_datasets_count}] Creative Generation
(MT-Bench)")
            total_ttr, total_words = 0, 0
            tot_pt, tot_ct, tot_ptime, tot_ctime = 0, 0, 0, 0
            for i, prompt_text in enumerate(ds_mtbench, 1):
                gen_text, pt, ct, ptime, ctime = generate_and_measure(llm,
f"Instruction: {prompt_text}\nResponse:\n", 300, temperature=0.7)
                words = re.findall(r'\b\w+\b', gen_text.lower())
                if words:
                    total_ttr += len(set(words))/len(words)
                    total_words += len(words)
                tot_pt+=pt; tot_ct+=ct; tot_ptime+=ptime; tot_ctime+=ctime
                print_progress(i, len(ds_mtbench), "MT-Bench")

```

```

        avg_ttr = total_ttr/len(ds_mtbench) if ds_mtbench else 0
        avg_w = total_words/len(ds_mtbench) if ds_mtbench else 0
        skor_mt_str = format_score(f"TTR: {avg_ttr:.3f} (W:{avg_w:.0f})")
        hasil_evaluasi['MT-Bench'] = skor_mt_str
        simpan_ke_tabel("Creative Generation", "MT-Bench", tot_pt, tot_ct,
tot_ptime, tot_ctime, skor_mt_str)

    peak_vram_thread_stop = True
    vram_thread.join()

    for row in tabel_skripsi[-total_datasets_count:]:
        row["Peak"] = f"{peak_vram_value:.2f}"

    print("\n" + "-"*40)
    print(f" RINGKASAN AKURASI/SKOR: {model_name}")
    print("-" * 40)
    ringkasan_counter = 0
    if 'MMLU' in hasil_evaluasi:
        ringkasan_counter += 1
        print(f"{ringkasan_counter}. Factual Retrieval (MMLU)           : Akurasi
{hasil_evaluasi['MMLU']}")
    if 'GSM8K' in hasil_evaluasi:
        ringkasan_counter += 1
        print(f"{ringkasan_counter}. Arithmetic Reasoning (GSM8K)       : Akurasi
{hasil_evaluasi['GSM8K']}")
    if 'HumanEval' in hasil_evaluasi:
        ringkasan_counter += 1
        print(f"{ringkasan_counter}. Code Synthesis (HumanEval)         :
pass@1 {hasil_evaluasi['HumanEval']}")
    if 'MT-Bench' in hasil_evaluasi:
        ringkasan_counter += 1
        print(f"{ringkasan_counter}. Creative Generation (MT-Bench)    : Richnes
{hasil_evaluasi['MT-Bench']}")
    print("-" * 40)

    print("\n STATISTIK HARDWARE SYSTEM:")
    print(f" - Base VRAM Model   : ~{net_vram_usage:.2f} GB")
    print(f" - True PEAK VRAM    : ~{peak_vram_value:.2f} GB (Terjadi saat
generasi token)\n")

    del llm

    print("\n" + "="*149)
    print(" TABEL HASIL PENGUJIAN SKRIPSI ")
    print("="*149)

```

```

    heading = f"| {'Kategori Kognitif':<22} | {'Sumber Dataset':<15} |
{'Model':<25} | {'Hasil Akurasi/Skor':<22} | {'VRAM Usage (GB)':<15} | {'Peak VRAM
(GB)':<14} | {'Prompt t/s':<11} | {'Gen t/s':<10} |"
    print(heading)
    print("|" + "-"*24 + "|" + "-"*17 + "|" + "-"*27 + "|" + "-"*24 + "|" + "-"*17
+ "|" + "-"*16 + "|" + "-"*13 + "|" + "-"*12 + "|")

    for r in tabel_skripsi:
        print(f"| {r['Kategori']:<22} | {r['Dataset']:<15} | {r['Model']:<25} |
{r['Hasil Skor/Akurasi']:<22} | {r['VRAM']:<15} | {r['Peak']:<14} | {r['Prompt
t/s']:<11} | {r['Gen t/s']:<10} |")

    print("="*149)
    print("\nKeterangan:")
    print("    TTR (Type-Token Ratio) : Rasio kata unik terhadap total kata dalam
teks yang dihasilkan model.")
    print("                                Nilai mendekati 1.0 menunjukkan kosakata yang
lebih beragam dan tidak repetitif.")
    print("    W    (Words)                : Rata-rata jumlah kata per respons yang
dihasilkan model.")
    print("                                Mengukur seberapa panjang dan elaboratif
output model dalam menjawab prompt kreatif.")
    print()

    csv_rows = []
    for r in tabel_skripsi:
        csv_rows.append({
            "Kategori": r["Kategori"],
            "Dataset": r["Dataset"],
            "Model": r["Model"],
            "Hasil Skor/Akurasi": r["Hasil Skor/Akurasi"],
            "VRAM (GB)": r["VRAM"],
            "Peak VRAM (GB)": r["Peak"],
            "Prompt t/s": r["Prompt t/s"],
            "Gen t/s": r["Gen t/s"]
        })
    append_csv("benchmark_results.csv", csv_rows)
    print(f"[CSV] Hasil disimpan ke: benchmark_results.csv (append mode)")

```

LAMPIRAN B

SKRIP EKSEKUSI INFERENSI EDGE (Tecno Pova 5)

A. benchmark.sh

```
#!/bin/bash
# =====
# Skripsi Benchmark - Clean Manual Input v4
# Lingkungan: Termux (non-root) di Tecno Pova 5 (Helio G99, RAM 8 GB)
# Mesin inferensi: Llama.cpp (build natif ARM)
# Output: hasilv2.csv
# =====
#
# Catatan reproducibility:
#   Script ini adalah versi yang benar-benar dieksekusi pada perangkat
#   penelitian. Berkas .gguf yang diuji disiapkan terlebih dahulu di PC
#   (WSL Ubuntu + cmake build CUDA + llama-quantize), kemudian dipindahkan
#   ke folder /storage/emulated/0/Download/SLM pada perangkat Android.
#
# Parameter inferensi yang dipakai:
#   - temperature      = 0,35
#   - top-p            = 0,9
#   - min-p            = 0,05
#   - repeat-penalty   = 1,1
#   - max tokens       = 1024
#   - context size     = 2048
#   - threads          = 6
#
# Prompt instruksi:
#   Prompt yang digunakan dirancang secara seragam dengan penyisipan
#   instruksi 'anti-thinking' (perintah untuk langsung menjawab).
#   Hal ini sengaja diterapkan sebagai stress-test (uji beban). Pada realitas
#   eksekusinya, arsitektur reasoning Qwen 3.5 terbukti mengabaikan instruksi
#   tersebut dan tetap memaksakan fase <think> secara laten, yang terekam
#   sebagai anomali durasi waktu eksekusi yang panjang pada data observasi.

MODEL_DIR="/storage/emulated/0/Download/SLM"
LLAMA_CLI="$HOME/Llama.cpp/build/bin/llama-cli"
CSV_FILE="hasilv2.csv"
THREADS=6

MAX_TOKENS=1024
CONTEXT_SIZE=2048
```

```

# Strong anti-thinking prompt.
PROMPT="### Instruction:
Explain briefly what artificial intelligence is.
Answer directly and concisely. Do not use Thinking Process, reasoning steps,
analysis, or lists.
Give only the final answer without any explanation of your thinking.

### Response:"

echo "=====
echo " Skripsi Benchmark - Clean Manual Mode"
echo " Optimized for LFM2.5 & Qwen3.5"
echo "=====

if [ ! -f "$CSV_FILE" ]; then
    echo "Timestamp,Model,Total Time (s),Prompt Speed (t/s),Gen Speed (t/s),Free
RAM Start (MB),RAM Used (MB),CPU Peak (%),Question,Answer" > "$CSV_FILE"
fi

mapfile -t MODELS < <(ls "$MODEL_DIR"/*.gguf 2>/dev/null)
TOTAL_MODELS=${#MODELS[@]}

echo "Found $TOTAL_MODELS models."
echo "=====

for i in "${!MODELS[@]}; do
    MODEL="${MODELS[$i]}"
    NAME=$(basename "$MODEL")

    CURRENT_TIME=$(date +%Y-%m-%d %H:%M:%S)

    echo ""
    echo "[$((i+1))/$TOTAL_MODELS] $CURRENT_TIME"
    echo "Model: $NAME"

    TEMP_RAM=$(mktemp)
    TEMP_CPU=$(mktemp)
    echo "0" > "$TEMP_RAM"
    echo "0" > "$TEMP_CPU"

    FREE_RAM_MB=$(grep MemAvailable /proc/meminfo | awk '{printf "%.2f", $2/1024}')

    START_TIME=$(date +%s)

    # Resource monitor: pantau VmRSS dan %CPU dari proses llama-cli setiap 0,5 s.
    monitor_resources() {
        PEAK_RAM=0; PEAK_CPU=0

```

```

while true; do
    L_PID=$(pidof llama-cli 2>/dev/null | awk '{print $1}')
    [ -z "$L_PID" ] && L_PID=$(pgrep -x llama-cli | head -n 1)
    if [ -n "$L_PID" ]; then
        CUR_RAM=$(grep VmRSS /proc/$L_PID/status 2>/dev/null | awk '{print
$2}')

        CUR_RAM=${CUR_RAM:-0}
        (( CUR_RAM > PEAK_RAM )) && { PEAK_RAM=$CUR_RAM; echo "$PEAK_RAM" >
"$TEMP_RAM"; }

        CUR_CPU=$(ps -p $L_PID -o %cpu= 2>/dev/null | tr -d ' ')
        CUR_CPU=${CUR_CPU:-0}
        PEAK_CPU=$(awk -v c="$CUR_CPU" -v p="$PEAK_CPU" 'BEGIN {print
(c>p)?c:p}')

        echo "$PEAK_CPU" > "$TEMP_CPU"

    fi
    sleep 0.5
done

}

monitor_resources &
MONITOR_PID=$!

echo ""
echo "=== QUESTION ==="
echo "$PROMPT"
echo ""
echo "=== MODEL ANSWER ==="

echo "/exit" | "$LLAMA_CLI" -m "$MODEL" \
    -p "$PROMPT" \
    -n $MAX_TOKENS \
    -t $THREADS \
    -c $CONTEXT_SIZE \
    --temp 0.35 \
    --top-p 0.9 \
    --min-p 0.05 \
    --repeat-penalty 1.1 \
    2>&1 | grep -A 500 "### Response:" | tee temp_output.log

kill $MONITOR_PID 2>/dev/null
END_TIME=$(date +%s)
TOTAL_TIME=$((END_TIME-START_TIME))

PEAK_RAM_KB=$(cat "$TEMP_RAM")
PEAK_CPU=$(cat "$TEMP_CPU")
RAM_USED_MB=$(awk "BEGIN {printf \"%.2f\", ${PEAK_RAM_KB:-0} / 1024}")

```

```

rm -f "$TEMP_RAM" "$TEMP_CPU"

echo ""
echo "Enter data for this model:"
read -p "Prompt Speed (t/s)      -> " PROMPT_SPEED
read -p "Generation Speed (t/s) -> " GEN_SPEED

echo ""
echo "Paste Model Answer (then press ENTER, then Ctrl+D):"
ANSWER=$(cat)

# Default values
[[ -z "$PROMPT_SPEED" ]] && PROMPT_SPEED="0.00"
[[ -z "$GEN_SPEED" ]] && GEN_SPEED="0.00"

# Amankan tanda kutip dalam teks agar tidak merusak CSV (standar CSV escaping).
SAFE_PROMPT="${PROMPT//\"/\"\\\"}"
SAFE_ANSWER="${ANSWER//\"/\"\\\"}"

# Simpan satu baris hasil ke CSV.
echo
"$CURRENT_TIME,$NAME,$TOTAL_TIME,$PROMPT_SPEED,$GEN_SPEED,$FREE_RAM_MB,$RAM_USED_MB
,$PEAK_CPU,\"$SAFE_PROMPT\", \"$SAFE_ANSWER\" " >> "$CSV_FILE"

echo "Saved: $NAME"

if [ $((i+1)) -lt $TOTAL_MODELS ]; then
    echo ""
    read -p "Press [ENTER] to continue to next model..."
fi
done

echo ""
echo "ALL TESTS COMPLETED."
echo "Results saved to: $CSV_FILE"

```

LAMPIRAN C

HASIL BENCHMARK

A. Hasil benchmark Dataset MMLU, GSM8K, HumanEval, MT-Bench

Dataset	Model	Skor
MMLU	LFM 1.2B - F16 (Run 1)	33.00% (33/100)
GSM8K	LFM 1.2B - F16 (Run 1)	58.00% (58/100)
HumanEval	LFM 1.2B - F16 (Run 1)	36.00% (36/100)
MT-Bench	LFM 1.2B - F16 (Run 1)	TTR: 0.526
MMLU	LFM 1.2B - F16 (Run 2)	31.00% (31/100)
GSM8K	LFM 1.2B - F16 (Run 2)	59.00% (59/100)
HumanEval	LFM 1.2B - F16 (Run 2)	36.00% (36/100)
MT-Bench	LFM 1.2B - F16 (Run 2)	TTR: 0.526
MMLU	LFM 1.2B - F16 (Run 3)	32.00% (32/100)
GSM8K	LFM 1.2B - F16 (Run 3)	57.00% (57/100)
HumanEval	LFM 1.2B - F16 (Run 3)	36.00% (36/100)
MT-Bench	LFM 1.2B - F16 (Run 3)	TTR: 0.526
MMLU	LFM 1.2B - F16 (Rata-rata)	32.00% (96/300)
GSM8K	LFM 1.2B - F16 (Rata-rata)	58.00% (174/300)
HumanEval	LFM 1.2B - F16 (Rata-rata)	36.00% (108/300)
MT-Bench	LFM 1.2B - F16 (Rata-rata)	TTR: 0.526
MMLU	LFM 1.2B - Q5 K M (Run 1)	34.00% (34/100)
GSM8K	LFM 1.2B - Q5 K M (Run 1)	55.00% (55/100)
HumanEval	LFM 1.2B - Q5 K M (Run 1)	29.00% (29/100)
MT-Bench	LFM 1.2B - Q5 K M (Run 1)	TTR: 0.509
MMLU	LFM 1.2B - Q5 K M (Run 2)	35.00% (35/100)
GSM8K	LFM 1.2B - Q5 K M (Run 2)	56.00% (56/100)
HumanEval	LFM 1.2B - Q5 K M (Run 2)	29.00% (29/100)
MT-Bench	LFM 1.2B - Q5 K M (Run 2)	TTR: 0.509
MMLU	LFM 1.2B - Q5 K M (Run 3)	33.00% (33/100)
GSM8K	LFM 1.2B - Q5 K M (Run 3)	54.00% (54/100)
HumanEval	LFM 1.2B - Q5 K M (Run 3)	29.00% (29/100)
MT-Bench	LFM 1.2B - Q5 K M (Run 3)	TTR: 0.509
MMLU	LFM 1.2B - Q5 K M (Rata-rata)	34.00% (102/300)
GSM8K	LFM 1.2B - Q5 K M (Rata-rata)	55.00% (165/300)
HumanEval	LFM 1.2B - Q5 K M (Rata-rata)	29.00% (87/300)
MT-Bench	LFM 1.2B - Q5 K M (Rata-rata)	TTR: 0.509
MMLU	LFM 1.2B - Q4 K M (Run 1)	25.00% (25/100)
GSM8K	LFM 1.2B - Q4 K M (Run 1)	50.00% (50/100)
HumanEval	LFM 1.2B - Q4 K M (Run 1)	37.00% (37/100)
MT-Bench	LFM 1.2B - Q4 K M (Run 1)	TTR: 0.468
MMLU	LFM 1.2B - Q4 K M (Run 2)	24.00% (24/100)
GSM8K	LFM 1.2B - Q4 K M (Run 2)	49.00% (49/100)
HumanEval	LFM 1.2B - Q4 K M (Run 2)	37.00% (37/100)
MT-Bench	LFM 1.2B - Q4 K M (Run 2)	TTR: 0.468
MMLU	LFM 1.2B - Q4 K M (Run 3)	26.00% (26/100)
GSM8K	LFM 1.2B - Q4 K M (Run 3)	51.00% (51/100)
HumanEval	LFM 1.2B - Q4 K M (Run 3)	37.00% (37/100)
MT-Bench	LFM 1.2B - Q4 K M (Run 3)	TTR: 0.468
MMLU	LFM 1.2B - Q4 K M (Rata-rata)	25.00% (75/300)
GSM8K	LFM 1.2B - Q4 K M (Rata-rata)	50.00% (150/300)
HumanEval	LFM 1.2B - Q4 K M (Rata-rata)	37.00% (111/300)
MT-Bench	LFM 1.2B - Q4 K M (Rata-rata)	TTR: 0.468
MMLU	LFM 1.2B - Q3 K M (Run 1)	28.00% (28/100)
GSM8K	LFM 1.2B - Q3 K M (Run 1)	40.00% (40/100)
HumanEval	LFM 1.2B - Q3 K M (Run 1)	30.00% (30/100)

MT-Bench	LFM 1.2B - Q3 K M (Run 1)	TTR: 0.403
MMLU	LFM 1.2B - Q3 K M (Run 2)	27.00% (27/100)
GSM8K	LFM 1.2B - Q3 K M (Run 2)	41.00% (41/100)
HumanEval	LFM 1.2B - Q3 K M (Run 2)	30.00% (30/100)
MT-Bench	LFM 1.2B - Q3 K M (Run 2)	TTR: 0.403
MMLU	LFM 1.2B - Q3 K M (Run 3)	29.00% (29/100)
GSM8K	LFM 1.2B - Q3 K M (Run 3)	39.00% (39/100)
HumanEval	LFM 1.2B - Q3 K M (Run 3)	30.00% (30/100)
MT-Bench	LFM 1.2B - Q3 K M (Run 3)	TTR: 0.403
MMLU	LFM 1.2B - Q3 K M (Rata-rata)	28.00% (84/300)
GSM8K	LFM 1.2B - Q3 K M (Rata-rata)	40.00% (120/300)
HumanEval	LFM 1.2B - Q3 K M (Rata-rata)	30.00% (90/300)
MT-Bench	LFM 1.2B - Q3 K M (Rata-rata)	TTR: 0.403
MMLU	Qwen 3.5 2B - F16 (Run 1)	40.00% (40/100)
GSM8K	Qwen 3.5 2B - F16 (Run 1)	27.00% (27/100)
HumanEval	Qwen 3.5 2B - F16 (Run 1)	52.00% (52/100)
MT-Bench	Qwen 3.5 2B - F16 (Run 1)	TTR: 0.528
MMLU	Qwen 3.5 2B - F16 (Run 2)	40.00% (40/100)
GSM8K	Qwen 3.5 2B - F16 (Run 2)	27.00% (27/100)
HumanEval	Qwen 3.5 2B - F16 (Run 2)	52.00% (52/100)
MT-Bench	Qwen 3.5 2B - F16 (Run 2)	TTR: 0.534
MMLU	Qwen 3.5 2B - F16 (Run 3)	40.00% (40/100)
GSM8K	Qwen 3.5 2B - F16 (Run 3)	27.00% (27/100)
HumanEval	Qwen 3.5 2B - F16 (Run 3)	52.00% (52/100)
MT-Bench	Qwen 3.5 2B - F16 (Run 3)	TTR: 0.531
MMLU	Qwen 3.5 2B - F16 (Rata-rata)	40.00% (120/300)
GSM8K	Qwen 3.5 2B - F16 (Rata-rata)	27.00% (81/300)
HumanEval	Qwen 3.5 2B - F16 (Rata-rata)	52.00% (156/300)
MT-Bench	Qwen 3.5 2B - F16 (Rata-rata)	TTR: 0.531
MMLU	Qwen 3.5 2B - Q5 K M (Run 1)	29.00% (29/100)
GSM8K	Qwen 3.5 2B - Q5 K M (Run 1)	37.00% (37/100)
HumanEval	Qwen 3.5 2B - Q5 K M (Run 1)	47.00% (47/100)
MT-Bench	Qwen 3.5 2B - Q5 K M (Run 1)	TTR: 0.463
MMLU	Qwen 3.5 2B - Q5 K M (Run 2)	29.00% (29/100)
GSM8K	Qwen 3.5 2B - Q5 K M (Run 2)	37.00% (37/100)
HumanEval	Qwen 3.5 2B - Q5 K M (Run 2)	47.00% (47/100)
MT-Bench	Qwen 3.5 2B - Q5 K M (Run 2)	TTR: 0.429
MMLU	Qwen 3.5 2B - Q5 K M (Run 3)	29.00% (29/100)
GSM8K	Qwen 3.5 2B - Q5 K M (Run 3)	37.00% (37/100)
HumanEval	Qwen 3.5 2B - Q5 K M (Run 3)	47.00% (47/100)
MT-Bench	Qwen 3.5 2B - Q5 K M (Run 3)	TTR: 0.446
MMLU	Qwen 3.5 2B - Q5 K M (Rata-rata)	29.00% (87/300)
GSM8K	Qwen 3.5 2B - Q5 K M (Rata-rata)	37.00% (111/300)
HumanEval	Qwen 3.5 2B - Q5 K M (Rata-rata)	47.00% (141/300)
MT-Bench	Qwen 3.5 2B - Q5 K M (Rata-rata)	TTR: 0.446
MMLU	Qwen 3.5 2B - Q4 K M (Run 1)	37.00% (37/100)
GSM8K	Qwen 3.5 2B - Q4 K M (Run 1)	42.00% (42/100)
HumanEval	Qwen 3.5 2B - Q4 K M (Run 1)	40.00% (40/100)
MT-Bench	Qwen 3.5 2B - Q4 K M (Run 1)	TTR: 0.522
MMLU	Qwen 3.5 2B - Q4 K M (Run 2)	37.00% (37/100)
GSM8K	Qwen 3.5 2B - Q4 K M (Run 2)	42.00% (42/100)
HumanEval	Qwen 3.5 2B - Q4 K M (Run 2)	40.00% (40/100)
MT-Bench	Qwen 3.5 2B - Q4 K M (Run 2)	TTR: 0.524
MMLU	Qwen 3.5 2B - Q4 K M (Run 3)	37.00% (37/100)
GSM8K	Qwen 3.5 2B - Q4 K M (Run 3)	42.00% (42/100)
HumanEval	Qwen 3.5 2B - Q4 K M (Run 3)	40.00% (40/100)
MT-Bench	Qwen 3.5 2B - Q4 K M (Run 3)	TTR: 0.523
MMLU	Qwen 3.5 2B - Q4 K M (Rata-rata)	37.00% (111/300)
GSM8K	Qwen 3.5 2B - Q4 K M (Rata-rata)	42.00% (126/300)
HumanEval	Qwen 3.5 2B - Q4 K M (Rata-rata)	40.00% (120/300)
MT-Bench	Qwen 3.5 2B - Q4 K M (Rata-rata)	TTR: 0.523
MMLU	Qwen 3.5 2B - Q3 K M (Run 1)	40.00% (40/100)
GSM8K	Qwen 3.5 2B - Q3 K M (Run 1)	27.00% (27/100)

HumanEval	Qwen 3.5 2B - Q3 K M (Run 1)	25.00% (25/100)
MT-Bench	Qwen 3.5 2B - Q3 K M (Run 1)	TTR: 0.485
MMLU	Qwen 3.5 2B - Q3 K M (Run 2)	40.00% (40/100)
GSM8K	Qwen 3.5 2B - Q3 K M (Run 2)	27.00% (27/100)
HumanEval	Qwen 3.5 2B - Q3 K M (Run 2)	25.00% (25/100)
MT-Bench	Qwen 3.5 2B - Q3 K M (Run 2)	TTR: 0.481
MMLU	Qwen 3.5 2B - Q3 K M (Run 3)	40.00% (40/100)
GSM8K	Qwen 3.5 2B - Q3 K M (Run 3)	27.00% (27/100)
HumanEval	Qwen 3.5 2B - Q3 K M (Run 3)	25.00% (25/100)
MT-Bench	Qwen 3.5 2B - Q3 K M (Run 3)	TTR: 0.483
MMLU	Qwen 3.5 2B - Q3 K M (Rata-rata)	40.00% (120/300)
GSM8K	Qwen 3.5 2B - Q3 K M (Rata-rata)	27.00% (81/300)
HumanEval	Qwen 3.5 2B - Q3 K M (Rata-rata)	25.00% (75/300)
MT-Bench	Qwen 3.5 2B - Q3 K M (Rata-rata)	TTR: 0.483

B. Hasil benchmark pada tecno pova 5.

Model	TotalTime_ s	PromptTP S	GenTP S	FreeRAMStart_M B	RAMUsed_M B	CPUPeak_pc t
LFM2.5-1.2B-Base-F16.gguf	20.0	24.3	5.3	2741.75	2284.64	264.0
LFM2.5-1.2B-Base-Q3_K_M.gguf	13.0	19.5	11.7	3083.36	890.64	472.0
LFM2.5-1.2B-Base-Q4_K_M.gguf	8.0	45.2	13.5	3229.13	1433.05	322.0
LFM2.5-1.2B-Base-Q5_K_M.gguf	9.0	36.6	10.8	3350.02	1645.39	343.0
Qwen3.5-2B-F16.gguf	209.0	25.0	1.6	3249.32	3732.91	401.0
LFM2.5-1.2B-Base-F16.gguf	14.0	38.3	5.7	2875.14	2313.28	306.0
LFM2.5-1.2B-Base-Q3_K_M.gguf	13.0	19.6	10.1	3249.8	922.86	455.0
LFM2.5-1.2B-Base-Q4_K_M.gguf	7.0	45.5	14.0	3174.66	1464.53	274.0
LFM2.5-1.2B-Base-Q5_K_M.gguf	10.0	33.7	9.8	3223.1	1674.77	325.0
Qwen3.5-2B-F16.gguf	393.0	22.2	2.0	3409.66	3750.69	479.0
Qwen3.5-2B-Q3_K_M.gguf	87.0	13.6	4.6	3848.76	1897.99	523.0
Qwen3.5-2B-Q4_K_M.gguf	343.0	29.6	4.7	3620.37	2563.34	526.0
Qwen3.5-2B-Q5_K_M.gguf	154.0	22.3	4.9	3335.02	2856.58	489.0

LFM2.5-1.2B-Base-F16.gguf	14.0	37.8	5.7	3024.59	2312.5	307.0
LFM2.5-1.2B-Base-Q3_K_M.gguf	17.0	15.7	11.4	3276.3	922.41	465.0
LFM2.5-1.2B-Base-Q4_K_M.gguf	7.0	40.3	13.5	2972.35	1461.34	302.0
LFM2.5-1.2B-Base-Q5_K_M.gguf	13.0	35.7	11.3	3259.23	1675.67	404.0
Qwen3.5-2B-F16.gguf	329.0	20.9	2.0	3489.2	3749.16	411.0
Qwen3.5-2B-Q3_K_M.gguf	287.0	15.5	3.9	3969.68	1897.95	527.0
Qwen3.5-2B-Q4_K_M.gguf	251.0	24.4	5.2	3386.88	2562.42	475.0
Qwen3.5-2B-Q5_K_M.gguf	208.0	22.8	4.3	3296.65	2856.0	529.0
Qwen3.5-2B-Q3_K_M.gguf	187.0	14.55	4.25	3909.22	1897.97	525.0
Qwen3.5-2B-Q4_K_M.gguf	297.0	27.0	4.95	3503.625	2562.88	500.5
Qwen3.5-2B-Q5_K_M.gguf	181.0	22.55	4.6	3315.835	2856.29	509.0

LAMPIRAN D

LANGKAH-LANGKAH PERSIAPAN LINGKUNGAN DAN KUANTISASI MODEL

A. Persiapan Lingkungan Komputasi (Host PC)

1. Persiapan Lingkungan Komputasi (Host PC)

Instalasi subsistem Linux (WSL) beserta pustaka dependensi dasar (C/C++ *compiler*, Git, dan Python) yang dibutuhkan untuk proses kompilasi.

```
wsl.exe --install Ubuntu
cd ~
mkdir skripsi_kuantisasi && cd skripsi_kuantisasi
sudo apt update && sudo apt upgrade -y
sudo apt install build-essential git python3-pip python3-venv -y
```

2. Pengunduhan Repositori dan Penyiapan Lingkungan Virtual Python

Kloning kode sumber *llama.cpp* dan pembuatan lingkungan virtual Python terisolasi untuk instalasi pustaka pengunduh model.

```
git clone https://github.com/ggerganov/llama.cpp
cd llama.cpp
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
pip install huggingface_hub
```

3. Kompilasi Mesin Inferensi dengan Akselerasi GPU (CUDA)

Kompilasi program *llama.cpp* dengan integrasi *NVIDIA CUDA Toolkit* guna mengaktifkan dukungan pemrosesan pada GPU (RTX 3060).

```
sudo apt install cmake -y
sudo apt install nvidia-cuda-toolkit -y
cmake -B build -DGGML_CUDA=ON
cmake --build build --config Release -j 4
```

4. Pengunduhan Model dan Konversi Format ke GGUF

Pengunduhan bobot model asli dari Hugging Face, dilanjutkan dengan konversi format .safetensors menjadi format FP16.gguf.

```
huggingface-cli download Qwen/Qwen3.5-2B --local-dir ./models/Qwen3.5-2B --local-dir-use-symlinks False
python3 convert_hf_to_gguf.py ./models/Qwen3.5-2B --outtype f16 --outfile ./models/Qwen3.5-2B/Qwen3.5-2B-F16.gguf
```

5. Eksekusi *Post-Training Quantization* (PTQ)

Reduksi presisi bobot model dari FP16 ke varian kuantisasi target (contoh instruksi untuk format Q4_K_M) menggunakan skrip llama-quantize

```
./build/bin/llama-quantize ./models/Qwen3.5-2B/Qwen3.5-2B-F16.gguf
./models/Qwen3.5-2B/Qwen3.5-2B-Q4_K_M.gguf Q4_K_M
ls -lh ./models/Qwen3.5-2B/
```

6. Persiapan Dataset Evaluasi Linguistik

Pengunduhan dan ekstraksi korpus WikiText-2 sebagai dataset rujukan evaluasi *Perplexity*.

```
wget https://huggingface.co/datasets/ggml-org/ci/resolve/main/wikitext-2-raw-v1.zip
unzip wikitext-2-raw-v1.zip
```

7. Eksekusi Pengukuran Perplexity (PPL)

Eksekusi pengujian nilai *Perplexity* dengan batas token konteks 512 (-c 512) dan instruksi pemindahan (*offloading*) seluruh lapisan komputasi ke dalam VRAM GPU (-ngl 999).

```
./build/bin/llama-perplexity -m ./models/Qwen3.5-2B/Qwen3.5-2B-Q4_K_M.gguf -f wikitext-2-raw/wiki.test.raw -c 512 -ngl 999
```

Catatan: Prosedur pada tahap 4 hingga tahap 7 di atas merupakan representasi untuk model Qwen 3.5 (2B). Proses yang sama dieksekusi secara identik untuk model LFM 2.5 (1.2B) dengan penyesuaian nama direktori asal dan target berkas keluaran.

B. Persiapan Lingkungan Inferensi *Edge* (Termux - Android)

1. Eksekusi Pengukuran Perplexity (PPL)

Inisialisasi lingkungan Termux pada perangkat Tecno Pova 5, dilanjutkan dengan pembaruan *package manager* dan instalasi *compiler C/C++ (clang)* serta perkakas *build* dasar.

```
pkg update && pkg upgrade -y
pkg install clang cmake make git -y
```

2. Kompilasi Nativ *llama.cpp* pada Arsitektur ARM64

Kloning repositori dan eksekusi kompilasi kode sumber secara natif di atas arsitektur ARM perangkat seluler tanpa menggunakan antarmuka grafis.

```
git clone https://github.com/gganganov/llama.cpp
cd llama.cpp
make -j4
```

3. Pemberian Akses Sistem *File* dan Migrasi Model

Pemberian izin akses penyimpanan internal kepada Termux agar dapat membaca berkas model .gguf yang sebelumnya telah dikompilasi di PC dan ditransfer melalui koneksi USB (MTP) ke folder *Download* perangkat Android.

```
termux-setup-storage
ls -lh ~/storage/downloads/SLM/
```

4. Persiapan dan Eksekusi Skrip *Benchmark*

Pemberian hak akses eksekusi (executable) pada skrip bash otomatisasi, dilanjutkan dengan inisiasi pengujian inferensi untuk merekam metrik CPU, RAM, dan *throughput* (TPS).

```
chmod +x benchmark.sh  
./benchmark.sh
```



LAMPIRAN E

DAFTAR AKRONIM DAN ISTILAH TEKNIS

Akronim / Istilah Teknis	Kepanjangan / Nama Resmi	Deskripsi & Konteks Fungsional Riset
AI	<i>Artificial Intelligence</i>	Teknologi Kecerdasan Buatan yang mensimulasikan kecerdasan manusia ke dalam sistem komputasi perangkat lunak.
NLP	<i>Natural Language Processing</i>	Cabang ilmu kecerdasan buatan yang berfokus pada interaksi dan pemrosesan bahasa alami manusia oleh komputer.
LLM	<i>Large Language Model</i>	Arsitektur model <i>neural network</i> dengan skala miliaran parameter yang dilatih untuk memahami dan membangkitkan teks bahasa manusia.
SLM	<i>Small Language Model</i>	Model bahasa dengan skala parameter lebih kecil (umumnya di bawah 3–7 miliar parameter) yang dioptimalkan secara khusus untuk perangkat tepi (<i>edge device</i>).
PTQ	<i>Post-Training Quantization</i>	Metode kompresi model kecerdasan buatan pasca-pelatihan dengan menyusutkan presisi resolusi bit matriks bobot tanpa melewati fase pelatihan ulang (<i>retraining</i>) yang mahal.
QAT	<i>Quantization-Aware Training</i>	Metode kuantisasi model yang menyertakan simulasi penurunan resolusi bit semenjak fase pelatihan model berlangsung di <i>server</i> .
GGUF	<i>GPT-Generated Unified Format</i>	Format berkas biner terpadu yang dirancang oleh ekosistem <i>llama.cpp</i> untuk memuat, menyimpan, dan mengeksekusi model bahasa secara efisien pada pemrosesan CPU.
FP16	<i>Floating-Point 16-bit</i>	Format representasi numerik desimal presisi setengah (2 bytes) yang digunakan sebagai standar <i>baseline</i> model orisinal berspesifikasi tinggi.
FP32	<i>Floating-Point 32-bit</i>	Format representasi numerik desimal presisi tunggal (4 bytes) yang umum digunakan pada fase pelatihan awal model di infrastruktur <i>server cloud</i> .
INT4 / INT8	<i>Integer 4-bit / 8-bit</i>	Format representasi bilangan bulat seragam beresolusi 4-bit atau 8-bit pada metode kuantisasi konvensional (<i>uniform quantization</i>).
Q3_K_M	<i>K-Quantization 3-bit Medium</i>	Varian tingkat kompresi ekstrem GGUF <i>k-quants</i> yang memadatkan sebagian besar lapisan matriks tensor menjadi resolusi rata-rata 3-bit.
Q4_K_M	<i>K-Quantization 4-bit Medium</i>	Varian tingkat kompresi menengah GGUF <i>k-quants</i> yang memadatkan sebagian besar matriks menjadi resolusi 4-bit dengan tetap mengamankan lapisan atensi (<i>attention layers</i>) esensial.
Q5_K_M	<i>K-Quantization 5-bit Medium</i>	Varian tingkat kompresi ringan GGUF <i>k-quants</i> yang memadatkan komponen matriks menjadi resolusi 5-bit untuk retensi akurasi logika maksimal terhadap <i>baseline</i> .
TPS	<i>Tokens per Second</i>	Metrik pengukuran kecepatan <i>throughput</i> inferensi teks, merepresentasikan jumlah sub-kata atau token yang diproses (<i>prompt</i>) atau dihasilkan (<i>generation</i>) per detik.
PPL	<i>Perplexity</i>	Metrik evaluasi intrinsik untuk menilai kualitas pemodelan bahasa berdasarkan tingkat ketidakpastian teks berikutnya pada dataset korpus pengujian (WikiText-2).
MMLU	<i>Massive Multitask Language Understanding</i>	Dataset <i>benchmark</i> standar untuk menguji kapabilitas penalaran multi-tugas, pemahaman konseptual, dan pengetahuan umum dari model bahasa.
GSM8K	<i>Grade School Math 8K</i>	Dataset <i>benchmark</i> standar yang berisi soal cerita matematika tingkat sekolah dasar untuk menguji nalar kuantitatif alur multi-langkah model.
HumanEval	<i>Human Evaluation Benchmark</i>	Kumpulan masalah pemrograman fungsional bahasa Python untuk mengevaluasi ketepatan sintaksis dan logika pembuatan kode dari model bahasa.

MT-Bench	<i>Multi-Turn Benchmark</i>	Kerangka evaluasi berbasis asisten otomatis untuk menilai kualitas dan konsistensi respons model bahasa dalam skenario percakapan multi-babak.
TTR	<i>Type-Token Ratio</i>	Rasio statistik antara jumlah kosakata unik (<i>type</i>) dengan total kata keseluruhan (<i>token</i>) untuk mengukur tingkat kekayaan dan keragaman leksikal kosakata.
OOM	<i>Out of Memory</i>	Kondisi kegagalan operasional sistem ketika alokasi memori utama yang diminta oleh program melampaui kapasitas fisik RAM yang tersedia.
LMKD	<i>Low Memory Killer Daemon</i>	Subsistem pengawas internal pada OS Android yang bertugas meminimalkan proses aplikasi latar belakang secara paksa jika memori sistem berada di tahap kritis.
RAM	<i>Random Access Memory</i>	Unit memori utama volatil komputer tempat penyimpanan data instruksi sementara dari proses aplikasi yang sedang dieksekusi secara aktif.
CPU	<i>Central Processing Unit</i>	Unit Pemrosesan Pusat perangkat keras yang bertugas mengoordinasikan, menerjemahkan, dan mengeksekusi instruksi aritmatika komputer.
GPU	<i>Graphics Processing Unit</i>	Sirkuit elektronik khusus yang dirancang untuk mempercepat pemrosesan komputasi paralel masif seperti pengolahan tensor matriks <i>neural network</i> AI.
SoC	<i>System on Chip</i>	Sirkuit terpadu (IC) yang menyatukan seluruh komponen arsitektur komputer utama (CPU, GPU, pengontrol memori) ke dalam satu keping semikonduktor tunggal.
ARM	<i>Advanced RISC Machines</i>	Arsitektur rumpun prosesor berbasis set instruksi komputer ringkas (RISC) yang dominan diimplementasikan pada arsitektur gawai bergerak karena efisiensi dayanya.
WSL2	<i>Windows Subsystem for Linux 2</i>	Arsitektur subsistem dari Microsoft yang memungkinkan lingkungan sistem operasi Linux (Ubuntu) berjalan secara natif dan berkinerja tinggi di dalam kernel Windows.
VmRSS	<i>Virtual Memory Resident Set Size</i>	Parameter kernel Linux yang menunjukkan porsi memori RAM fisik riil yang sedang dialokasikan dan dikonsumsi secara aktual oleh suatu ID proses tertentu.
LPDDR4x	<i>Low Power Double Data Rate 4x</i>	Standardisasi spesifikasi modul teknologi RAM fisik hemat energi berkecepatan tinggi yang dioptimalkan untuk perangkat bergerak.
ROM / UFS	<i>Read-Only Memory / Universal Flash Storage</i>	Komponen media penyimpanan internal <i>non-volatile</i> tempat berkas biner model AI disimpan secara persisten di dalam <i>smartphone</i> .
API	<i>Application Programming Interface</i>	Antarmuka Pemrograman Aplikasi yang berfungsi sebagai jembatan protokol pertukaran data antarsistem perangkat lunak yang berbeda.
CUDA	<i>Compute Unified Device Architecture</i>	Platform komputasi paralel dan antarmuka pemrograman buatan NVIDIA untuk mengeksplorasi daya pemrosesan kartu grafis dalam kalkulasi matematika.
LFM 2.5	<i>Lightweight Function Model 2.5</i>	Seri arsitektur model bahasa ultra-ringan berskala 1,2 miliar parameter keluaran tahun 2026 yang menjadi objek eksperimen dalam riset ini.
Qwen 3.5	<i>Qwen Model 3.5</i>	Seri arsitektur model bahasa penalaran berskala 2 miliar parameter buatan Alibaba Cloud keluaran tahun 2026 yang menjadi objek eksperimen riset ini.
Termux	<i>Terminal Emulator for Android</i>	Aplikasi lingkungan emulator terminal <i>non-root</i> yang menyediakan subsistem lingkungan Linux terisolasi di dalam OS Android tanpa modifikasi <i>firmware</i> sistem.
llama.cpp	<i>Llama in C/C++</i>	<i>Framework</i> mesin inferensi berbasis bahasa pemrograman C/C++ natif yang dirancang untuk mengeksekusi LLM/SLM secara cepat di atas pemrosesan CPU <i>bare-metal</i> .

LAMPIRAN F

SCREENSHOT TERMINAL

A. Persiapan Lingkungan Komputasi Pada WSL2

```
username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi$ cd llama.cpp
username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ python3 -m venv venv
username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ source venv/bin/activate
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ pip install -r requirements.txt
```

```
username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi$ sudo apt install build-essential git python3-pip python3-venv -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
```

B. Mengunduh huggingface_hub

```
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ pip install huggingface_hub
Requirement already satisfied: huggingface_hub in ./venv/lib/python3.12/site-packages (0.36.2)
Requirement already satisfied: filelock in ./venv/lib/python3.12/site-packages (from huggingface_hub) (3.25.0)
Requirement already satisfied: fsspec>=2023.5.0 in ./venv/lib/python3.12/site-packages (from huggingface_hub) (2026.2.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in ./venv/lib/python3.12/site-packages (from huggingface_hub) (1.3.2)
Requirement already satisfied: packaging>=20.9 in ./venv/lib/python3.12/site-packages (from huggingface_hub) (26.0)
Requirement already satisfied: pyyaml>=5.1 in ./venv/lib/python3.12/site-packages (from huggingface_hub) (6.0.3)
Requirement already satisfied: requests in ./venv/lib/python3.12/site-packages (from huggingface_hub) (2.32.5)
Requirement already satisfied: tqdm>=4.42.1 in ./venv/lib/python3.12/site-packages (from huggingface_hub) (4.67.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in ./venv/lib/python3.12/site-packages (from huggingface_hub) (4.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in ./venv/lib/python3.12/site-packages (from requests->huggingface_hub) (3.4.5)
Requirement already satisfied: idna<4,>=2.5 in ./venv/lib/python3.12/site-packages (from requests->huggingface_hub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./venv/lib/python3.12/site-packages (from requests->huggingface_hub) (2.6.3)
Requirement already satisfied: certifi>=2017.4.17 in ./venv/lib/python3.12/site-packages (from requests->huggingface_hub) (2026.2.25)
```

C. Install cmake

```
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ sudo apt install cmake -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cmake-data libarchive13t64 libjsoncpp25 librhash0 libuv1t64
```

D. Install Nvidia Cuda

```
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ sudo apt install nvidia-cuda-toolkit -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf ca-certificates-java cpp-12 fonts-dejavu-extra g++-12 gcc-12 gcc-12-base
```

E. Build Nvidia Cuda

```
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ cmake -B build -DGGML_CUDA=ON
-- The C compiler identification is GNU 13.3.0
-- The CXX compiler identification is GNU 13.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
```

F. Build cmake

```
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ cmake --build build --config Release -j
[ 0%] Building CXX object vendor/cpp-httpplib/CMakeFiles/cpp-httpplib.dir/httpplib.cpp.o
[ 1%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml.cpp.o
[ 1%] Building C object examples/gguf-hash/CMakeFiles/xxhash.dir/deps/xxhash/xxhash.c.o
[ 1%] Building CXX object tools/mtmd/CMakeFiles/llama-llava-cli.dir/deprecation-warning.cpp.o
[ 1%] Building CXX object tools/mtmd/CMakeFiles/llama-gemma3-cli.dir/deprecation-warning.cpp.o
[ 1%] Building C object examples/gguf-hash/CMakeFiles/sha256.dir/deps/sha256/sha256.c.o
[ 1%] Building CXX object tools/mtmd/CMakeFiles/llama-minicpmv-cli.dir/deprecation-warning.cpp.o
[ 0%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml.c.o
[ 2%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-backend.cpp.o
[ 2%] Building C object examples/gguf-hash/CMakeFiles/sha1.dir/deps/sha1/sha1.c.o
[ 2%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-threading.cpp.o
[ 2%] Building CXX object tools/mtmd/CMakeFiles/llama-qwen2vl-cli.dir/deprecation-warning.cpp.o
```

G. Download AI Model

```
(venv) username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi/llama.cpp$ huggingface-cli download Qwen/Qwen3.5-2B --local-dir ./models/Qwen3.5-2B --local-dir-use-symlinks False
/home/username/skripsi_kuantisasi/llama.cpp/venv/lib/python3.12/site-packages/huggingface_hub/commands/download.py:141: FutureWarning: Ignoring --local-dir-use-symlinks. Downloading to a local directory does not use symlinks anymore.
```

H. Clone Github llama.cpp

```
username@DESKTOP-C4ILFM4:~/skripsi_kuantisasi$ git clone https://github.com/ggerganov/llama.cpp
Cloning into 'llama.cpp'...
remote: Enumerating objects: 82607, done.
remote: Counting objects: 100% (187/187), done.
remote: Compressing objects: 100% (161/161), done.
remote: Total 82607 (delta 97), reused 26 (delta 26), pack-reused 82420 (from 3)
Receiving objects: 100% (82607/82607), 311.88 MiB | 5.56 MiB/s, done.
Resolving deltas: 100% (59377/59377), done.
```

I. Konversi Format Model ke GGUF

```

user@kali:~/llama$ python3 convert_hf_to_gguf.py ./models/Qwen3.5-2B --outfile F16 --outfile ./models/Qwen3.5-2B/reasoning/Qwen3.5-2B-F16.gguf
INFO:hf-to-gguf:Loading model: Qwen3.5-2B
WARNING:hf-to-gguf:Failed to load model config from models/Qwen3.5-2B. The checkpoint you are trying to load has model type 'qwen3.5' but Transformers does not recognize this architecture. This could be because of an issue with the checkpoint, or because your version of Transformers is out of date.

You can update Transformers with the command 'pip install --upgrade transformers'. If this does not work, and the checkpoint is very new, then there may not be a release version that supports this model yet. In this case, you can get the most up-to-date code by installing Transformers from source with the command 'pip install git+https://github.com/huggingface/transformers.git'

WARNING:hf-to-gguf:Trying to load config.json instead
INFO:hf-to-gguf:Model architecture: Qwen3.5ForConditionalGeneration
WARNING:hf-to-gguf:Failed to load model config from models/Qwen3.5-2B. The checkpoint you are trying to load has model type 'qwen3.5' but Transformers does not recognize this architecture. This could be because of an issue with the checkpoint, or because your version of Transformers is out of date.

You can update Transformers with the command 'pip install --upgrade transformers'. If this does not work, and the checkpoint is very new, then there may not be a release version that supports this model yet. In this case, you can get the most up-to-date code by installing Transformers from source with the command 'pip install git+https://github.com/huggingface/transformers.git'

WARNING:hf-to-gguf:Loading model weight map from 'model.safetensors.index.json'
INFO:hf-to-gguf:gguf: indexing model part 'model.safetensors-00001-of-00001.safetensors'
INFO:gguf:gguf.writer:gguf: This GGUF file is for Little Endian only
INFO:hf-to-gguf:Exporting model...
INFO:hf-to-gguf:token_embd.weight torch.bfloat16 -> F16, shape = [2848, 248320]
INFO:hf-to-gguf:blk.0.attn_norm.weight torch.bfloat16 -> F32, shape = [2848]
INFO:hf-to-gguf:blk.0.ssm_a torch.float32 -> F32, shape = [16]
INFO:hf-to-gguf:blk.0.ssm_conv1d.weight torch.bfloat16 -> F32, shape = [4, 6144]
INFO:hf-to-gguf:blk.0.ssm_dt.bias torch.bfloat16 -> F32, shape = [16]
INFO:hf-to-gguf:blk.0.ssm_alpha.weight torch.bfloat16 -> F16, shape = [2848, 16]
INFO:hf-to-gguf:blk.0.ssm_beta.weight torch.bfloat16 -> F16, shape = [2848, 16]
INFO:hf-to-gguf:blk.0.attn_kv.weight torch.bfloat16 -> F16, shape = [2848, 6144]
INFO:hf-to-gguf:blk.0.attn_gate.weight torch.bfloat16 -> F16, shape = [2848, 2848]
INFO:hf-to-gguf:blk.0.ssm_norm.weight torch.float32 -> F32, shape = [128]
INFO:hf-to-gguf:blk.0.ssm_out.weight torch.bfloat16 -> F16, shape = [2848, 2848]
INFO:hf-to-gguf:blk.0.ffn_down.weight torch.bfloat16 -> F16, shape = [6144, 2848]
INFO:hf-to-gguf:blk.0.ffn_gate.weight torch.bfloat16 -> F16, shape = [2848, 6144]
INFO:hf-to-gguf:blk.0.ffn_up.weight torch.bfloat16 -> F16, shape = [2848, 6144]
INFO:hf-to-gguf:blk.0.post_attention_norm.weight torch.bfloat16 -> F32, shape = [2848]
INFO:hf-to-gguf:blk.1.attn_norm.weight torch.bfloat16 -> F32, shape = [2848]
INFO:hf-to-gguf:blk.1.ssm_a torch.float32 -> F32, shape = [16]
INFO:hf-to-gguf:blk.1.ssm_conv1d.weight torch.bfloat16 -> F32, shape = [4, 6144]
INFO:hf-to-gguf:blk.1.ssm_dt.bias torch.bfloat16 -> F32, shape = [16]
INFO:hf-to-gguf:blk.1.ssm_alpha.weight torch.bfloat16 -> F16, shape = [2848, 16]
INFO:hf-to-gguf:blk.1.ssm_beta.weight torch.bfloat16 -> F16, shape = [2848, 16]
INFO:hf-to-gguf:blk.1.attn_kv.weight torch.bfloat16 -> F16, shape = [2848, 6144]
INFO:hf-to-gguf:blk.1.attn_gate.weight torch.bfloat16 -> F16, shape = [2848, 2848]
INFO:hf-to-gguf:blk.1.ssm_norm.weight torch.float32 -> F32, shape = [128]
INFO:hf-to-gguf:blk.1.ssm_out.weight torch.bfloat16 -> F16, shape = [2848, 2848]
INFO:hf-to-gguf:blk.1.ffn_down.weight torch.bfloat16 -> F16, shape = [6144, 2848]
INFO:hf-to-gguf:blk.1.ffn_gate.weight torch.bfloat16 -> F16, shape = [2848, 6144]
INFO:hf-to-gguf:blk.1.post_attention_norm.weight torch.bfloat16 -> F32, shape = [2848]
INFO:hf-to-gguf:blk.10.attn_norm.weight torch.bfloat16 -> F32, shape = [2848]
INFO:hf-to-gguf:blk.10.ssm_a torch.float32 -> F32, shape = [16]
INFO:hf-to-gguf:blk.10.ssm_conv1d.weight torch.bfloat16 -> F32, shape = [4, 6144]
INFO:hf-to-gguf:blk.10.ssm_dt.bias torch.bfloat16 -> F32, shape = [16]
INFO:hf-to-gguf:blk.10.ssm_alpha.weight torch.bfloat16 -> F16, shape = [2848, 16]
INFO:hf-to-gguf:blk.10.ssm_beta.weight torch.bfloat16 -> F16, shape = [2848, 16]

```

J. Menjalankan PTQ dari model FP16 ke Q4_K_M

```

user@kali:~/llama$ ./build/bin/llama-quantize ./models/Qwen3.5-2B/Qwen3.5-2B-F16.gguf ./models/Qwen3.5-2B/reasoning/Qwen3.5-2B-Q4_K_M.gguf Q4_K_M
ggml_cuda_init: found 1 CUDA devices (Total VRAM: 6143 MiB):
Device 0: NVIDIA GeForce RTX 3860 Laptop GPU, compute capability 8.6, VMW: yes, VRAM: 6143 MiB
llama_print_build_info: build = 9352 (b4c0949a0)
llama_print_build_info: built with GPU 13.3.0 for Linux x86_64
llama_quantize: quantizing './models/Qwen3.5-2B/Qwen3.5-2B-F16.gguf' to './models/Qwen3.5-2B/reasoning/Qwen3.5-2B-Q4_K_M.gguf' as Q4_K_M
llama_model_loader: loaded meta data with 41 key-value pairs and 320 tensors from ./models/Qwen3.5-2B/Qwen3.5-2B-F16.gguf (version GGUF V3 (Latest))
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv 0:          general.architecture str           = qwen3.5
llama_model_loader: - kv 1:          general.type str                 = model
llama_model_loader: - kv 2:          general.name str                  = Qwen3.5 2B
llama_model_loader: - kv 3:          general.basename str              = Qwen3.5
llama_model_loader: - kv 4:          general.size_label str            = 2B
llama_model_loader: - kv 5:          general.license str               = apache-2.0
llama_model_loader: - kv 6:          general.license_link str          = https://huggingface.co/Qwen/Qwen3.5-2...
llama_model_loader: - kv 7:          general.base_model_count u32      = 1
llama_model_loader: - kv 8:          general.base_model.0.name str      = Qwen3.5 2B Base
llama_model_loader: - kv 9:          general.base_model.0.organization str = Qwen
llama_model_loader: - kv 10:         general.base_model.0.repo_url str   = https://huggingface.co/Qwen/Qwen3.5-2...
llama_model_loader: - kv 11:         general.tags arr[str,1]            = ["image-text-to-text"]
llama_model_loader: - kv 12:        qwen35.block_count u32             = 24
llama_model_loader: - kv 13:        qwen35.context_length u32          = 262144
llama_model_loader: - kv 14:        qwen35.embedding_length u32        = 2848
llama_model_loader: - kv 15:        qwen35.feed_forward_length u32      = 6144
llama_model_loader: - kv 16:        qwen35.attention_head_count u32      = 8
llama_model_loader: - kv 17:        qwen35.attention_head_count_kv u32   = 2
llama_model_loader: - kv 18:        qwen35.rope.dimension_sections arr[i32,4] = [11, 11, 10, 0]
llama_model_loader: - kv 19:        qwen35.rope.freq_base f32          = 10000000.000000
llama_model_loader: - kv 20:        qwen35.attention.layer_norm_rms_epsilon f32 = 0.000001
llama_model_loader: - kv 21:        qwen35.attention.key_length u32      = 256
llama_model_loader: - kv 22:        qwen35.attention.value_length u32     = 256
llama_model_loader: - kv 23:          general.file_type u32             = 1
llama_model_loader: - kv 24:        qwen35.ssm.conv_kernel u32          = 4
llama_model_loader: - kv 25:        qwen35.ssm.state_size u32           = 128
llama_model_loader: - kv 26:        qwen35.ssm.group_count u32          = 16
llama_model_loader: - kv 27:        qwen35.ssm.time_step_rank u32        = 16
llama_model_loader: - kv 28:        qwen35.ssm.inner_size u32           = 2848

```

K. Ukuran Model Setelah PTQ

```
(venv) username@localhost:~/skripsi_kuantisasi/llama.cpp$ ls -lh ./models/LFM2.5-1.2B-Base/
total 6.4G
-rw-r--r-- 1 username username 2.2G Apr 1 10:20 LFM2.5-1.2B-Base-F16.gguf
-rw-r--r-- 1 username username 573M Apr 27 14:42 LFM2.5-1.2B-Base-Q3_K_M.gguf
-rw-r--r-- 1 username username 698M Apr 1 10:22 LFM2.5-1.2B-Base-Q4_K_M.gguf
-rw-r--r-- 1 username username 805M Apr 27 14:43 LFM2.5-1.2B-Base-Q5_K_M.gguf
-rw-r--r-- 1 username username 11K Apr 1 09:39 LICENSE
-rw-r--r-- 1 username username 8.8K Apr 1 09:39 README.md
-rw-r--r-- 1 username username 1.3K Apr 1 09:39 chat_template.jinja
-rw-r--r-- 1 username username 1.2K Apr 1 09:39 config.json
-rw-r--r-- 1 username username 132 Apr 1 09:39 generation_config.json
-rw-r--r-- 1 username username 2.2G Apr 1 09:50 model.safetensors
-rw-r--r-- 1 username username 434 Apr 1 09:39 special_tokens_map.json
-rw-r--r-- 1 username username 4.6M Apr 1 09:39 tokenizer.json
-rw-r--r-- 1 username username 90K Apr 1 09:39 tokenizer_config.json
```

```
(venv) username@localhost:~/skripsi_kuantisasi/llama.cpp$ ls -lh ./models/Qwen3.5-2B/
total 12G
-rw-r--r-- 1 username username 12K Mar 10 03:03 LICENSE
-rw-r--r-- 1 username username 3.6G Mar 10 03:40 Qwen3.5-2B-F16.gguf
-rw-r--r-- 1 username username 1.1G Apr 27 14:38 Qwen3.5-2B-Q3_K_M.gguf
-rw-r--r-- 1 username username 1.2G Mar 10 03:58 Qwen3.5-2B-Q4_K_M.gguf
-rw-r--r-- 1 username username 1.4G Apr 27 14:40 Qwen3.5-2B-Q5_K_M.gguf
-rw-r--r-- 1 username username 62K Mar 10 03:03 README.md
-rw-r--r-- 1 username username 7.6K Mar 10 03:03 chat_template.jinja
-rw-r--r-- 1 username username 2.9K Mar 10 03:03 config.json
-rw-r--r-- 1 username username 3.2M Mar 10 03:03 merges.txt
-rw-r--r-- 1 username username 4.3G Mar 10 03:15 model.safetensors-00001-of-00001.safetensors
-rw-r--r-- 1 username username 63K Mar 10 03:03 model.safetensors.index.json
-rw-r--r-- 1 username username 390 Mar 10 03:03 preprocessor_config.json
-rw-r--r-- 1 username username 13M Mar 10 03:03 tokenizer.json
-rw-r--r-- 1 username username 17K Mar 10 03:03 tokenizer_config.json
-rw-r--r-- 1 username username 385 Mar 10 03:03 video_preprocessor_config.json
-rw-r--r-- 1 username username 6.5M Mar 10 03:03 vocab.json
```

L. Proses Pengujian PPL

```
(venv) username@localhost:~/skripsi_kuantisasi/llama.cpp$ ./build/bin/llama-perplexity -m ./models/Qwen3.5-2B-Q4_K_M.gguf -f wikitext-2-rw/llm1.test.raw -c 512 -ngl 99
0.00 073.080 | common_init_result: fitting params to device memory ...
0.00 073.616 | common_init_result: for bugs during this step try to reproduce them with -fit-off, or provide --verbose logs if the bug only occurs with -fit-on
0.01 518.782 | llama_context: n_ctx_seq (512) < n_ctx_train (262144) - the full capacity of the model will not be utilized
0.01 589.962 | common_init_from_params: warming up the model with an empty run - please wait ... (no-warmup to disable)
0.03 716.710 |
0.03 716.900 | system_info: n_threads = 6 (n_threads_batch = 6) / 12 | CUDA: ARCHS = 860 | USE_GRAPHS = 1 | PEER_MAX_BATCH_SIZE = 128 | CPU : SSE3 = 1 | SSE4 = 1 | AVX = 1 | AVX2 = 1 | F16C = 1 | FMA = 1 | B
M12 = 1 | LLAMAFILE = 1 | OMEMP = 1 | REPACK = 1 |
0.03 717.027 | perplexity: tokenizing the input ...
0.04 306.055 | perplexity: tokenization took 589.02 ms
0.04 306.193 | perplexity: calculating perplexity over 580 chunks, n_ctx=512, batch_size=2048, n_seq=4
0.05 451.580 | perplexity: 1.53 seconds per pass - ETA 3.68 minutes
[1]8.4856, [2]12.1818, [3]12.1531, [4]12.4313, [5]12.0583, [6]12.4159, [7]12.8149, [8]14.0077, [9]14.8142, [10]15.3878, [11]15.1448, [12]15.3985, [13]16.0237, [14]15.2495, [15]15.0086, [16]15.2874, [17]14.2290, [18]14.3292, [19]
14.1736, [20]14.0614, [21]13.5282, [22]13.3288, [23]12.7498, [24]12.0752, [25]11.8682, [26]11.5658, [27]11.3638, [28]11.2389, [29]11.1794, [30]11.1550, [31]11.0171, [32]11.0629, [33]10.9857, [34]11.0880, [35]11.2177, [36]11.4
876, [37]11.0411, [38]11.5077, [39]11.5822, [40]11.6700, [41]11.7159, [42]11.7089, [43]11.8085, [44]11.7701, [45]11.8208, [46]11.8128, [47]12.0267, [48]12.1330, [49]12.0847, [50]12.0402, [51]12.0231, [52]12.0876, [53]12.1345, [54]12.1882, [55]12.1584, [56]12.2074, [57]12.2456, [58]12.2607, [59]12.2641, [60]12.2962, [61]12.3299, [62]12.4042, [63]12.4544, [64]12.5478, [65]12.6728, [66]12.7437, [67]12.9081, [68]12.9209, [69]12.9211, [70]12.8721, [71]12.9
944, [72]12.9729, [73]13.0316, [74]13.0351, [75]12.9848, [76]12.9616, [77]12.9941, [78]12.9556, [79]12.8993, [80]12.8487, [81]12.8693, [82]12.8548, [83]12.8689, [84]12.8318, [85]12.8753, [86]12.9682, [87]12.9682, [88]12.9753
[89]12.9071, [90]12.8578, [91]12.8522, [92]12.8202, [93]12.8913, [94]12.9296, [95]12.9407, [96]12.9401, [97]12.9270, [98]12.8733, [99]12.8559, [100]12.9942, [101]12.9377, [102]12.9331, [103]12.9256, [104]12.8995, [105]12.9322
4, [106]12.9654, [107]13.0006, [108]13.0009, [109]13.0729, [110]13.1789, [111]13.2423, [112]13.2682, [113]13.3291, [114]13.3807, [115]13.4122, [116]13.3649, [117]13.3579, [118]13.3483, [119]13.3075, [120]13.3141, [121]13.2834
[122]13.2975, [123]13.1979, [124]13.1742, [125]13.1316, [126]13.1125, [127]13.0965, [128]13.0307, [129]12.9387, [130]12.8562, [131]12.9208, [132]12.9608, [133]12.9439, [134]12.9462, [135]12.9620, [136]12.9652, [137]12.9486,
[138]12.9816, [139]12.8507, [140]12.8518, [141]12.8616, [142]12.8668, [143]12.8509, [144]12.8569, [145]12.7853, [146]12.7508, [147]12.7001, [148]12.6780, [149]12.5985, [150]12.5747, [151]12.5556, [152]12.5285, [153]12.4650, [154]12.4277, [155]12.3993, [156]12.3391, [157]12.3841, [158]12.2484, [159]12.2215, [160]12.2012, [161]12.1993, [162]12.1465, [163]12.1328, [164]12.1248, [165]12.1593, [166]12.1388, [167]12.1391, [168]12.2075, [169]12.2762, [170]12.3225, [171]12.3838, [172]12.4044, [173]12.4728, [174]12.5656, [175]12.6118, [176]12.6741, [177]12.7338, [178]12.7433, [179]12.7726, [180]12.7726, [181]12.8116, [182]12.8481, [183]12.8935, [184]12.9107, [185]12.9091, [186]12.9395, [187]12.9409, [188]12.9556, [189]12.9720, [190]13.0378, [191]13.0552, [192]13.0614, [193]13.0672, [194]13.1236, [195]13.1236, [196]13.1579, [197]13.1585, [198]13.1212, [199]13.1011, [200]13.1260, [201]13.1459, [202]13.1539, [203]13.1539, [204]13.1539, [205]13.1539, [206]13.1539, [207]13.1539, [208]13.1539, [209]13.1539, [210]13.1539, [211]13.1539, [212]13.1539, [213]13.1539, [214]13.1539, [215]13.1539, [216]13.1539, [217]13.1539, [218]13.1539, [219]13.1539, [220]13.1539, [221]13.1539, [222]13.1539, [223]13.1539, [224]13.1539, [225]13.1539, [226]13.1539, [227]13.1539, [228]13.1539, [229]13.1539, [230]13.1539, [231]13.1539, [232]13.1539, [233]13.1539, [234]13.1539, [235]13.1539, [236]13.1539, [237]13.1539, [238]13.1539, [239]13.1539, [240]13.1539, [241]13.1539, [242]13.1539, [243]13.1539, [244]13.1539, [245]13.1539, [246]13.1539, [247]13.1539, [248]13.1539, [249]13.1539, [250]13.1539, [251]13.1539, [252]13.1539, [253]13.1539, [254]13.1539, [255]13.1539, [256]13.1539, [257]13.1539, [258]13.1539, [259]13.1539, [260]13.1539, [261]13.1539, [262]13.1539, [263]13.1539, [264]13.1539, [265]13.1539, [266]13.1539, [267]13.1539, [268]13.1539, [269]13.1539, [270]13.1539, [271]13.1539, [272]13.1539, [273]13.1539, [274]13.1539, [275]13.1539, [276]13.1539, [277]13.1539, [278]13.1539, [279]13.1539, [280]13.1539, [281]13.1539, [282]13.1539, [283]13.1539, [284]13.1539, [285]13.1539, [286]13.1539, [287]13.1539, [288]13.1539, [289]13.1539, [290]13.1539, [291]13.1539, [292]13.1539, [293]13.1539, [294]13.1539, [295]13.1539, [296]13.1539, [297]13.1539, [298]13.1539, [299]13.1539, [300]13.1539, [301]13.1539, [302]13.1539, [303]13.1539, [304]13.1539, [305]13.1539, [306]13.1539, [307]13.1539, [308]13.1539, [309]13.1539, [310]13.1539, [311]13.1539, [312]13.1539, [313]13.1539, [314]13.1539, [315]13.1539, [316]13.1539, [317]13.1539, [318]13.1539, [319]13.1539, [320]13.1539, [321]13.1539, [322]13.1539, [323]13.1539, [324]13.1539, [325]13.1539, [326]13.1539, [327]13.1539, [328]13.1539, [329]13.1539, [330]13.1539, [331]13.1539, [332]13.1539, [333]13.1539, [334]13.1539, [335]13.1539, [336]13.1539, [337]13.1539, [338]13.1539, [339]13.1539, [340]13.1539, [341]13.1539, [342]13.1539, [343]13.1539, [344]13.1539, [345]13.1539, [346]13.1539, [347]13.1539, [348]13.1539, [349]13.1539, [350]13.1539, [351]13.1539, [352]13.1539, [353]13.1539, [354]13.1539, [355]13.1539, [356]13.1539, [357]13.1539, [358]13.1539, [359]13.1539, [360]13.1539, [361]13.1539, [362]13.1539, [363]13.1539, [364]13.1539, [365]13.1539, [366]13.1539, [367]13.1539, [368]13.1539, [369]13.1539, [370]13.1539, [371]13.1539, [372]13.1539, [373]13.1539, [374]13.1539, [375]13.1539, [376]13.1539, [377]13.1539, [378]13.1539, [379]13.1539, [380]13.1539, [381]13.1539, [382]13.1539, [383]13.1539, [384]13.1539, [385]13.1539, [386]13.1539, [387]13.1539, [388]13.1539, [389]13.1539, [390]13.1539, [391]13.1539, [392]13.1539, [393]13.1539, [394]13.1539, [395]13.1539, [396]13.1539, [397]13.1539, [398]13.1539, [399]13.1539, [400]13.1539, [401]13.1539, [402]13.1539, [403]13.1539, [404]13.1539, [405]13.1539, [406]13.1539, [407]13.1539, [408]13.1539, [409]13.1539, [410]13.1539, [411]13.1539, [412]13.1539, [413]13.1539, [414]13.1539, [415]13.1539, [416]13.1539, [417]13.1539, [418]13.1539, [419]13.1539, [420]13.1539, [421]13.1539, [422]13.1539, [423]13.1539, [424]13.1539, [425]13.1539, [426]13.1539, [427]13.1539, [428]13.1539, [429]13.1539, [430]13.1539, [431]13.1539, [432]13.1539, [433]13.1539, [434]13.1539, [435]13.1539, [436]13.1539, [437]13.1539, [438]13.1539, [439]13.1539, [440]13.1539, [441]13.1539, [442]13.1539, [443]13.1539, [444]13.1539, [445]13.1539, [446]13.1539, [447]13.1539, [448]13.1539, [449]13.1539, [450]13.1539, [451]13.1539, [452]13.1539, [453]13.1539, [454]13.1539, [455]13.1539, [456]13.1539, [457]13.1539, [458]13.1539, [459]13.1539, [460]13.1539, [461]13.1539, [462]13.1539, [463]13.1539, [464]13.1539, [465]13.1539, [466]13.1539, [467]13.1539, [468]13.1539, [469]13.1539, [470]13.1539, [471]13.1539, [472]13.1539, [473]13.1539, [474]13.1539, [475]13.1539, [476]13.1539, [477]13.1539, [478]13.1539, [479]13.1539, [480]13.1539, [481]13.1539, [482]13.1539, [483]13.1539, [484]13.1539, [485]13.1539, [486]13.1539, [487]13.1539, [488]13.1539, [489]13.1539, [490]13.1539, [491]13.1539, [492]13.1539, [493]13.1539, [494]13.1539, [495]13.1539, [496]13.1539, [497]13.1539, [498]13.1539, [499]13.1539, [500]13.1539, [501]13.1539, [502]13.1539, [503]13.1539, [504]13.1539, [505]13.1539, [506]13.1539, [507]13.1539, [508]13.1539, [509]13.1539, [510]13.1539, [511]13.1539, [512]13.1539, [513]13.1539, [514]13.1539, [515]13.1539, [516]13.1539, [517]13.1539, [518]13.1539, [519]13.1539, [520]13.1539, [521]13.1539, [522]13.1539, [523]13.1539, [524]13.1539, [525]13.1539, [526]13.1539, [527]13.1539, [528]13.1539, [529]13.1539, [530]13.1539, [531]13.1539, [532]13.1539, [533]13.1539, [534]13.1539, [535]13.1539, [536]13.1539, [537]13.1539, [538]13.1539, [539]13.1539, [540]13.1539, [541]13.1539, [542]13.1539, [543]13.1539, [544]13.1539, [545]13.1539, [546]13.1539, [547]13.1539, [548]13.1539, [549]13.1539, [550]13.1539, [551]13.1539, [552]13.1539, [553]13.1539, [554]13.1539, [555]13.1539, [556]13.1539, [557]13.1539, [558]13.1539, [559]13.1539, [560]13.1539, [561]13.1539, [562]13.1539, [563]13.1539, [564]13.1539, [565]13.1539, [566]13.1539, [567]13.1539, [568]13.1539, [569]13.1539, [570]13.1539, [571]13.1539, [572]13.1539, [573]13.1539, [574]13.1539, [575]13.1539, [576]13.1539, [577]13.1539, [578]13.1539, [579]13.1539, [580]13.1539, [581]13.1539, [582]13.1539, [583]13.1539, [584]13.1539, [585]13.1539, [586]13.1539, [587]13.1539, [588]13.1539, [589]13.1539, [590]13.1539, [591]13.1539, [592]13.1539, [593]13.1539, [594]13.1539, [595]13.1539, [596]13.1539, [597]13.1539, [598]13.1539, [599]13.1539, [600]13.1539, [601]13.1539, [602]13.1539, [603]13.1539, [604]13.1539, [605]13.1539, [606]13.1539, [607]13.1539, [608]13.1539, [609]13.1539, [610]13.1539, [611]13.1539, [612]13.1539, [613]13.1539, [614]13.1539, [615]13.1539, [616]13.1539, [617]13.1539, [618]13.1539, [619]13.1539, [620]13.1539, [621]13.1539, [622]13.1539, [623]13.1539, [624]13.1539, [625]13.1539, [626]13.1539, [627]13.1539, [628]13.1539, [629]13.1539, [630]13.1539, [631]13.1539, [632]13.1539, [633]13.1539, [634]13.1539, [635]13.1539, [636]13.1539, [637]13.1539, [638]13.1539, [639]13.1539, [640]13.1539, [641]13.1539, [642]13.1539, [643]13.1539, [644]13.1539, [645]13.1539, [646]13.1539, [647]13.1539, [648]13.1539, [649]13.1539, [650]13.1539, [651]13.1539, [652]13.1539, [653]13.1539, [654]13.1539, [655]13.1539, [656]13.1539, [657]13.1539, [658]13.1539, [659]13.1539, [660]13.1539, [661]13.1539, [662]13.1539, [663]13.1539, [664]13.1539, [665]13.1539, [666]13.1539, [667]13.1539, [668]13.1539, [669]13.1539, [670]13.1539, [671]13.1539, [672]13.1539, [673]13.1539, [674]13.1539, [675]13.1539, [676]13.1539, [677]13.1539, [678]13.1539, [679]13.1539, [680]13.1539, [681]13.1539, [682]13.1539, [683]13.1539, [684]13.1539, [685]13.1539, [686]13.1539, [687]13.1539, [688]13.1539, [689]13.1539, [690]13.1539, [691]13.1539, [692]13.1539, [693]13.1539, [694]13.1539, [695]13.1539, [696]13.1539, [697]13.1539, [698]13.1539, [699]13.1539, [700]13.1539, [701]13.1539, [702]13.1539, [703]13.1539, [704]13.1539, [705]13.1539, [706]13.1539, [707]13.1539, [708]13.1539, [709]13.1539, [710]13.1539, [711]13.1539, [712]13.1539, [713]13.1539, [714]13.1539, [715]13.1539, [716]13.1539, [717]13.1539, [718]13.1539, [719]13.1539, [720]13.1539, [721]13.1539, [722]13.1539, [723]13.1539, [724]13.1539, [725]13.1539, [726]13.1539, [727]13.1539, [728]13.1539, [729]13.1539, [730]13.1539, [731]13.1539, [732]13.1539, [733]13.1539, [734]13.1539, [735]13.1539, [736]13.1539, [737]13.1539, [738]13.1539, [739]13.1539, [740]13.1539, [741]13.1539, [742]13.1539, [743]13.1539, [744]13.1539, [745]13.1539, [746]13.1539, [747]13.1539, [748]13.1539, [749]13.1539, [750]13.1539, [751]13.1539, [752]13.1539, [753]13.1539, [754]13.1539, [755]13.1539, [756]13.1539, [757]13.1539, [758]13.1539, [759]13.1539, [760]13.1539, [761]13.1539, [762]13.1539, [763]13.1539, [764]13.1539, [765]13.1539, [766]13.1539, [767]13.1539, [768]13.1539, [769]13.1539, [770]13.1539, [771]13.1539, [772]13.1539, [773]13.1539, [774]13.1539, [775]13.1539, [776]13.1539, [777]13.1539, [778]13.1539, [779]13.1539, [780]13.1539, [781]13.1539, [782]13.1539, [783]13.1539, [784]13.1539, [785]13.1539, [786]13.1539, [787]13.1539, [788]13.1539, [789]13.1539, [790]13.1539, [791]13.1539, [792]13.1539, [793]13.1539, [794]13.1539, [795]13.1539, [796]13.1539, [797]13.1539, [798]13.1539, [799]13.1539, [800]13.1539, [801]13.1539, [802]13.1539, [803]13.1539, [804]13.1539, [805]13.1539, [806]13.1539, [807]13.1539, [808]13.1539, [809]13.1539, [810]13.1539, [811]13.1539, [812]13.1539, [813]13.1539, [814]13.1539, [815]13.1539, [816]13.1539, [817]13.1539, [818]13.1539, [819]13.1539, [820]13.1539, [821]13.1539, [822]13.1539, [823]13.1539, [824]13.1539, [825]13.1539, [82
```

M. Proses Benchmark Model dan Hasil

```

username@localhost: ~/skripsi x + v
username@localhost:~$ cd skripsi_kuantisasi/llama.cpp
username@localhost:~/skripsi_kuantisasi/llama.cpp$ source venv/bin/activate
(venv) username@localhost:~/skripsi_kuantisasi/llama.cpp$ python mmlu.py

=====
PILIH JENIS BENCHMARK / MODE
=====
1. Multi-Dataset Benchmark (MMLU, GSM8K, HumanEval, MT-Bench)
2. Perplexity (PPL) - WikiText-2
3. Chat Only Mode
=====
Masukkan pilihan (1/2/3): 1

=====
PILIH DATASET BENCHMARK
=====
1. MMLU (Factual Retrieval)
2. GSM8K (Arithmetic Reasoning)
3. HumanEval (Code Synthesis)
4. MT-Bench (Creative Generation)
=====
Masukkan nomor dataset (pisah koma, contoh: 1,2,3,4)
>> Dataset dipilih: MMLU, GSM8K, HumanEval, MT-Bench

=====
PILIH MODEL UNTUK BENCHMARK
=====
1. LFM 1.2B - F16
2. LFM 1.2B - Q3_K_M
3. LFM 1.2B - Q4_K_M
4. LFM 1.2B - Q5_K_M
5. Qwen 3.5 2B - F16
6. Qwen 3.5 2B - Q3_K_M
7. Qwen 3.5 2B - Q4_K_M
8. Qwen 3.5 2B - Q5_K_M
=====
Masukkan nomor model (pisah koma, contoh: 1,3,6): 2
>> Model dipilih: LFM 1.2B - Q3_K_M

=====
MEMUAT DATASETS...
=====

=====
MENGUJI MODEL: LFM 1.2B - Q3_K_M
=====
llama_context: n_ctx_seq (4096) < n_ctx_train (128000) -- the full capacity of the model will not be utilized

[1/4] Factual Retrieval (MMLU)
MMLU: [████████████████████] 100/100 (100.0%)

```

```

RINGKASAN AKURASI/SKOR: LFM 1.2B - Q3_K_M
1. Factual Retrieval (MMLU) : Akurasi 30.00% (30/100)
2. Arithmetic Reasoning (GSM8K) : Akurasi 37.00% (37/100)
3. Code Synthesis (HumanEval) : pass@1 29.00% (29/100)
4. Creative Generation (MT-Bench) : Richnes TTR: 0.368 (W:233)

=====
STATISTIK HARDWARE SYSTEM:
- Base VRAM Model : ~0.85 GB
- True PEAK VRAM : ~1.11 GB (Terjadi saat generasi token)

=====
TABEL HASIL PENGUJIAN SKRIPSI
=====
| Kategori Kognitif | Sumber Dataset | Model | Hasil Akurasi/Skor | VRAM Usage (GB) | Peak VRAM (GB) | Prompt t/s | Gen t/s |
|-----|-----|-----|-----|-----|-----|-----|-----|
| Factual Retrieval | MMLU | LFM 1.2B - Q3_K_M | 30.00% (30/100) | 0.85 | 1.11 | 7935.03 | 193.10 |
| Arithmetic Reasoning | GSM8K | LFM 1.2B - Q3_K_M | 37.00% (37/100) | 0.85 | 1.11 | 7844.71 | 182.63 |
| Code Synthesis | HumanEval | LFM 1.2B - Q3_K_M | 29.00% (29/100) | 0.85 | 1.11 | 6388.61 | 182.76 |
| Creative Generation | MT-Bench | LFM 1.2B - Q3_K_M | TTR: 0.368 (W:233) | 0.85 | 1.11 | 1893.74 | 180.44 |
|-----|-----|-----|-----|-----|-----|-----|-----|

Keterangan:
TTR (Type-Token Ratio) : Rasio kata unik terhadap total kata dalam teks yang dihasilkan model.
Nilai mendekati 1.0 menunjukkan kosakata yang lebih beragam dan tidak repetitif.
W (Words) : Rata-rata jumlah kata per respons yang dihasilkan model.
Mengukur seberapa panjang dan elaboratif output model dalam menjawab prompt kreatif.

[CSV] Hasil disimpan ke: benchmark_results.csv (append mode)
(venv) username@localhost:~/skripsi_kuantisasi/llama.cpp$

```

N. Proses Pengujian Model AI Di Handphone

```

09.36 >-
asoning steps, analysis, or lists.
Give only the final answer without any explanation of your thi
nking.

### Response:

=== MODEL ANSWER ===

Loading model...

llama.cpp

build      : b8946-0f1bb602d
model      : LFM2.5-1.2B-Base-Q3_K_M.gguf
modalities : text

available commands:
  /exit or Ctrl+C      stop or exit
  /regen               regenerate the last response
  /clear               clear the chat history
  /read <file>         add a text file
  /glob <pattern>      add text files using globbing pattern

> ### Instruction:
Explain briefly what artificial intelligence is.
Answer directly and concisely. Do not use Thinking Process, re
asoning steps, analysis, or lists.
Give only the final answer without any explanation of your thi
nking.

### Response:

Artificial Intelligence refers to computer systems that can pe
rform tasks requiring human-like intelligence, such as learnin
g, reasoning, problem-solving, perception and language underst
anding.

[ Prompt: 19.4 t/s | Generation: 11.9 t/s ]

> /exit

Exiting...

Enter data for this model:
Prompt Speed (t/s)      + 19.4
Generation Speed (t/s)  + 11.9

Paste Model Answer (then press ENTER, then Ctrl + D):
Artificial Intelligence refers to computer systems that can pe
rform tasks requiring human-like intelligence, such as learnin
g, reasoning, problem-solving, perception and language underst
anding.
✓ Saved: LFM2.5-1.2B-Base-Q3_K_M.gguf

Press [ENTER] to continue to next model...

ESC      /      -      HOME      ↑      END      PGUP
⌘      CTRL  ALT      ←      ↓      →      PGDN

```