PROPOSAL THESIS

EVALUASI DAN PERBANDINGAN KINERJA MODEL BAHASA BESAR BERPARAMETER KECIL DALAM MEMBUAT KODE PYTHON SECARA OTOMATIS

Oleh:
Muh Nur Aslam
NIM : 23206051031

PROGRAM STUDI INFORMATIKA
PROGRAM MAGISTER FAKULTAS SAINS DAN TEKNOLOGI
UIN SUNAN KALIJAGA YOGYAKARTA
2025

HALAMAN PENGESAHAN

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-564/Un.02/DST/PP.00.9/04/2025

Tugas Akhir dengan judul : EVALUASI DAN PERBANDINGAN KINERJA MODEL BAHASA BESAR BERPARAMETER KECIL DALAM MEMBUAT KODE PYTHON SECARA OTOMATIS

yang dipersiapkan dan disusun oleh:

Nama : MUH NUR ASLAM, S. Kom
Nomor Induk Mahasiswa : 23206051031
Telah diujikan pada : Kamis, 27 Maret 2025
Nilai ujian Tugas Akhir : A

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Dr. Agung Fatwanto, S.Si., M.Kom.
SIGNED

Valid ID: 6803a9e9a7acc

Penguji I

Dr. Agus Mulyanto, S.Si., M.Kom., ASEAN Eng.
SIGNED

Valid ID: 6800ac2e1d2ed

Penguji II

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
SIGNED

Valid ID: 6800cd617ef28

Yogyakarta, 27 Maret 2025
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6805ec8dd9508

HALAMAN PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Muh Nur Aslam
NIM	: 23206051031
Jenjang	: Magister
Program Studi	: Informatika

Menyatakan bahwa naskah tesis ini secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali pada bagian-bagian yang dirujuk sumbernya.

Yogyakarta, 28 Februari 2025

Saya yang menyatakan,

Muh Nur Aslam

NIM: 23206051031

HALAMAN PERNYATAAN BEBAS PLAGIASI

Yang bertanda tangan di bawah ini:

Nama	: Muh Nur Aslam
NIM	: 23206051031
Jenjang	: Magister
Program Studi	: Informatika

Menyatakan bahwa naskah tesis ini secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali pada bagian-bagian yang dirujuk sumbernya.

Yogyakarta, 28 Februari 2025

Saya yang menyatakan,

Muh Nur Aslam

NIM: 23206051031

SURAT PERSETUJUAN SKRIPSI

Hal : Persetujuan Tugas Akhir
Lamp : -

Kepada:
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
Di Yogyakarta

Assalamualaikum wr.wb.
Setelah membaca, meneliti, memberikan petunjuk dan mengkoreksi serta mengadakan perbaikan seperlunya, maka saya selaku pembimbing berpendapat bahwa tesis saudara:

Nama	:	Muh Nur Aslam
NIM	:	23206051031
Judul Tesis	:	EVALUASI DAN PERBANDINGAN KINERJA MODEL BAHASA BESAR BERPARAMETER KECIL DALAM MEMBUAT KODE PYTHON SECARA OTOMATIS

Sudah dapat diajukan kepada program studi Magister Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Magister Informatika.
Dengan ini saya mengharap agar tugas akhir tersebut di atas dapat segera dimunaqosyahkan.
Atas perhatiannya saya ucapkan terimakasih.

Yogyakarta, 28 Februari 2025
Pembimbing,

Dr. Agung Fatwanto, S. Si., M. Kom.
NIP: 97701032005011003

ABSTRAK

Large Language Models (LLMs) dengan parameter kecil seperti CodeGemma-2B, Qwen2.5-1.5B-Instruct, dan LLaMA-3.2-1B telah menunjukkan potensi dalam tugas code generation. Namun, belum ada studi yang secara khusus membandingkan kinerja model-model tersebut berdasarkan metrik berbasis sintaktik maupun fungsional. Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi dan membandingkan kinerja base model sebelum dan setelah fine-tuning menggunakan metrik fungsional pass@k serta metrik berbasis sintaktik seperti BLEU, ROUGE, ChrF, Exact Match, dan Levenshtein Distance. Metodologi penelitian ini bersifat deskriptif komparatif dengan pendekatan kuantitatif empiris. Pengumpulan data dilakukan secara eksperimental dengan menggunakan dataset Evol-Instruct untuk fine-tuning dan HumanEval untuk evaluasi. Hasil penelitian menunjukkan bahwa fine-tuning secara signifikan meningkatkan performa fungsional model berdasarkan metrik pass@k. Berdasarkan metode Chen, pass@1 CodeGemma-2B meningkat dari 0,057 (base model - FM) menjadi 0,186 (fine-tuned - FT), Qwen2.5 meningkat dari 0,456 (FM) menjadi 0,567 (FT), dan LLaMA-3.2 meningkat dari 0,815 (FM) menjadi 0,844 (FT). Peningkatan serupa terjadi pada pass@5, pass@10, dan pass@100, dengan kenaikan yang lebih signifikan terutama pada CodeGemma-2B. Sementara itu, hasil evaluasi menggunakan metode Kulal menunjukkan peningkatan yang lebih tinggi, terutama pada CodeGemma-2B, yang mengalami kenaikan dari 0,463 (FM) menjadi 0,878 (FT) pada pass@1. Qwen2.5 mencapai skor pass@1 sebesar 1.0 setelah fine-tuning, sementara LLaMA-3.2 telah mencapai performa optimal dengan skor 1.0 sejak awal. Namun, analisis metrik sintaktik menunjukkan hasil yang beragam. BLEU dan ROUGE mengalami sedikit penurunan pada ketiga model setelah fine-tuning, sementara ChrF menunjukkan peningkatan kecil pada CodeGemma-2B (dari 21,0 menjadi 22,2) dan Qwen2.5 (dari 23,11 menjadi 23,17). Exact Match tetap 0,0 untuk semua model, sedangkan Levenshtein Distance meningkat pada CodeGemma-2B (dari 102 menjadi 126) serta Qwen2.5 (dari 135 menjadi 131), mengindikasikan bahwa fine-tuning tidak secara konsisten meningkatkan kesamaan sintaktik. Selain itu, hasil uji Wilcoxon Signed-Rank Test dan analisis median-percentile menunjukkan bahwa fine-tuning tidak secara signifikan meningkatkan efisiensi waktu eksekusi model. Meskipun CodeGemma-2B dan LLaMA-3.2 menunjukkan sedikit peningkatan, Qwen2.5 justru

mengalami penurunan pada nilai median dan percentile 75%. Temuan ini mengindikasikan bahwa fine-tuning efektif dalam meningkatkan akurasi fungsional, tetapi belum tentu meningkatkan kesamaan sintaktik atau efisiensi waktu eksekusi. Penelitian ini memberikan kontribusi penting dalam memahami dampak fine-tuning terhadap model LLM dengan parameter kecil, khususnya dalam konteks code generation.

MOTTO

ْ ي ل ً س ُ ي ل

“Sebaik-baik manusia adalah yang paling bermanfaat bagi manusia”

HALAMAN PERSEMBAHAN

Tesis ini penulis persembahkan untuk:
Almamater Tercinta
Prodi Informatika S2
Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta

KATA PENGANTAR

Assalamu'alaikum wr.wb.

Puji syukur penulis panjatkan kehadirat Allah SWT, yang telah melimpahkan Rahmat, hidayah dan pertolongan-Nya kepada penulis sehingga tesis ini dapat terselesaikan dengan baik. Tesis dengan judul Evaluasi Dan Perbandingan Kinerja Model Bahasa Besar Berparameter Kecil Dalam Membuat Kode Python Secara Otomatis ini diajukan untuk memenuhi Sebagian persyaratan guna memperoleh gelar Magister Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta. Penulis menyadari bahwa keberhasilan ini tidak terlepas dari bantuan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terimakasih kepada:

1. Kedua Orang tua penulis Bapak Saharuddin dan Eny yang mendukung dan menyemangati di setiap Langkah.
2. Prof. Noorhaidi Hasan. Selaku Rektor pada UIN Sunan Kalijaga Yogyakarta yang telah memberikan kesempatan menempuh studi ini.
3. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku dekan Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta yang telah memberikan kesempatan untuk menempuh Pendidikan di Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta
4. Dr. Ir. Sumarsono, S.T., M.Kom. selaku Kepala Prodi Informatika S2 UIN Sunan Kalijaga Yogyakarta dan Ir. Muhammad Didik Rohmad Wahyudi, S.T., MT. selaku Sekjur Prodi Informatika UIN Sunan Kalijaga Yogyakarta

5. Bapak Dr. Agung Fatwanto, S.Si., M.Kom. selaku pembimbing yang telah berkenan merelakan waktu, tenaga dan ilmunya guna memberikan bimbingan kepada penulis dalam menyelesaikan tesis ini, serta ucapan terimakasih banyak dan penghargaan yang setinggi-tingginya yang dengan penuh kesabaran dan kearifan telah memberikan bimbingan, arahan dan dorongan di sela-sela kesibukannya.
6. Bapak dan Ibu Dosen, TU dan tenaga lain Prodi Informatika S2 UIN Sunan Kalijaga Yogyakarta, khususnya yang memberi kuliah, yang sudah memberikan banyak ilmu pengetahuan sehingga penulis dapat melaksanakan penelitian dan Menyusun hasil penelitian tersebut menjadi tesis ini.
7. Teman-teman Ciblon dan Miss Indana Khoirun Nida, S. Pd. yang sudah membantu memberikan dukungan dalam kelancaran sidang penulis.
8. Teman-teman mahasiswa Prodi Informatika S2 UIN Sunan Kalijaga Yogyakarta dan PBSB 2023 UIN Sunan Kalijaga Yogyakarta
9. Ucapan terima kasih saya sampaikan kepada semua pihak yang tidak mungkin saya sebutkan satu demi satu, yang telah banyak memberikan bantuan dan dukungan selama penyusunan Tesis ini.

Semoga Allah SWT senantiasa melimpahkan Rahmat dan hidayah-Nya kepada kita semua. Penulis berharap semoga Tesis ini bermanfaat bagi penulis khususnya dan pembaca pada umumnya.

Wassalamu'alaikum wr. wb.

Yogyakarta, 27 Maret 2025

Penyusun

Muh Nur Aslam

NIM. 23206051031

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Dalam beberapa tahun terakhir, bidang pemrosesan bahasa alami (Natural Languange Processing atau NLP) telah mengalami perubahan signifikan yang dipicu oleh kemunculan Generative AI. Generative AI merujuk pada cabang kecerdasan buatan yang focus pada pembuatan data baru yang serupa dengan data yang ada. Model-model Generative AI, terutama yang berbasis pada teknik deep learning, telah membuktikan kemampuannya dalam menghasilkan teks, gambar, music dan bahkan kode yang semakin memperluas aplikasi kecerdasan buatan dalam kehidupan sehari-hari. Seiring berkembangnya Generative AI, kemunculan model berbasis Transformer pada 2017 yang diperkenalkan oleh Vaswani (Vaswani, 2017) dengan judul "*Attention Is All You Need*" menjadi salah satu tonggak utama dalam evolusi tersebut. Transformer, dengan mekanisme self-attention mengubah cara memproses data input secara parallel, ini memungkinkan model lebih efisien menangai teks Panjang dibandingkan dengan model-model sebelumnya yang mengandalkan teknik sekuensial, seperti Recurrent Neural Networks (RNNs) dan Long Short-Term Memory (LSTM)

Penggunaan Transformer dalam model-model seperti GPT (Generative Pretrained Transformer) dan BERT (Bidirectional Encoder Representation from Transformers) telah membawa perubahan besar dalam cara model NLP mengatasi berbagai tugas, teknik self-attention

digunakan oleh Transformer memungkinkan model ini untuk secara efektif menangkap hubungan antar kata dalam sebuah kalimat, tanpa terpengaruh oleh jarak antar kata tersebut. Keunggulan ini menjadikan Transformer lebih unggul dibandingkan dengan pendekatan sebelumnya yang mengandalkan model berbasis RNN atau LSTM yang lebih terbatas dalam proses teks Panjang.

Namun, meskipun kemajuan besar telah dicapai dengan Transformer, beberapa peneliti menunjukkan bahwa kemampuan model seperti GPT dan BERT, dua dari model Transformer yang paling terkenal masih terbatas dalam beberapa aspek. Seperti yang diungkapkan oleh (Rogers et al., 2020) (meskipun model ini menunjukkan performa yang luar biasa dalam berbagai tugas NLP, model ini juga sering kali masih terjebak dalam masalah seperti bias dalam data pelatihan dan overfitting pada domain tertentu yang mengarah pada hasil yang kurang optimal dalam aplikasi dunia nyata. Selain itu, meskipun kemampuan fine-tuning yang ditawarkan oleh model berbasis Transformer memungkinkan model untuk beradaptasi dengan berbagai tugas, penggunaan data pelatihan yang sangat besar untuk melatihan model-model ini menjadi perhatian utama terkait keberlanjutan dan efisiensi energi (Strubell, Ganesh and McCallum, 2020)

Large Languange Models (LLMs), seperti GPT-3 dari OpenAI, BERT dan T5 dari Google, telah secara dramatis meningkatkan kemampuan AI dalam memahami dan menghasilkan NLP. Menurut (Yang, 2024) , GPT-3 menunjukkan kemampuan untuk melakukan tugas yang tidak secara eksplisit dilatih, fenomena yang dikenal

sebagai few-shot learning. Kemampuan ini memungkinkan LLMs untuk menggeneralisasi dari contoh yang terbatas, mengurangi kebutuhan akan dataset berlabel yang besar. Demikian pula (Wang et al., 2024) menunjukkan bahwa BERT dapat mencapai hasil terbaik dalam berbagai tolok ukur NLP, mulai dari analisis sentimen hingga inferensi bahasa alami. Model-model ini dilatih dengan jumlah data teks yang sangat besar, sering kali diambil dari internet, buku dan sumber lain yang dihasilkan manusia, yang memungkinkan model-model tersebut memahami tata Bahasa, konteks dan nuansa dalam Bahasa. Salah satu kekuatan utama LLMs adalah kemampuannya untuk memproses dan menghasilkan teks yang mirip dengan manusia dengan input minimal, menjadikannya sangat serbaguna untuk berbagai tugas, termasuk penerjemahan mesin, ringkasa, penjawab pertanyaan dan penulisan kreatif.

GPT-3, yang dikembangkan oleh OpenAI adalah salah satu LLM terbesar dan paling terkenal, dengan 175 miliar parameter. Model ini unggul dalam menghasilkan teks yang koheren dan relevan dengan konteks berdsasarkan prompt yang diberikan. Kemampuannya untuk melakukan berbagai tugas tanpa fine-tunin khusus untuk tugas tertentu, yang dikenal sebagai zero-shot learning, menandari lompatan besar dalam NLP. Dengan hanya memberikan prompt teks awal, GPT-3 dapat menghasilkan respons yang terperinci, membuat konten kreatif, atau bahkan menulis kode, menjadikannya salah satu generative AI yang paling fleksibel yang tersedia. Kemampuan ini telah menarik perhatian besar untuk aplikasi potensialnya di berbagai bidang, seperti pendidikan dan penelitian.

Penerapan Large Languange Models (LLMs) telah merevolusi berbagai industry dengan memberikan dampak signifikan pada cara kita berinteraksi dengan teknologi dan mengotomatisasikan berbagai tugas yang sebelumnya memerlukan keahlian manusia. Misalnya, dalam bidang customer service, LLMs digunakan untuk membuat chatbots yang mampu memberikan respons yang lebih manusiawi dan efisien. Di sektor keuangan, mereka digunakan untuk menganalisis laporan keuangan atau bahkan memberikan rekomendasi inversitas. Begitu pula di di dunia penerbitan dan meda, LLMs membantu dalam menghasilkan artikel atau konten secara otomatis, meningkatkan efisiensi kerja dan mengurangi waktu yang dibutuhkan dalam produksi konten. LLMs tidak hanya mengubah industry yang berhubungan langsung dengan teknologi, tetapi juga memperkenalkan pendekatan baru dalam pemrosesan informasi dan pembuatan Keputusan di berbagai sektor, mulai dari hukum hingga kedokteran

Salah satu bidang yang paling terlihat dampaknya adalah code generation. Dalam beberapa tahun terakhir, generasi kode otomatis menggunakan LLMs, seperti OpenAI Codex dan Github Copilot telah membuka jalan bagi otomatisasi pengembangan perangkat lunak. Model-model ini mampu menghasilkan kode yang sesuai dengan instruksi dalam bahasa pemrograman, bahkan menyediakan saran kode secara real-time, meningkatkan produktivitas developer, dan mengurangi kesalahan dalam penulisan kode. LLMs telah memperkenalkan konsep pemrograman berbasis instruksi yang lebih intuitif, memungkinkan programmer untuk menulis kode hanya dengan menjelaskan apa yang mereka inginkan tanpa harus memilikirkan detail teknis secara mendalam.

Namun meskipun potensi besar yang dimiliki oleh LLMs dalam code generation, beberapa tantang tetap ada. Salah satu kritik dari peneliti adalah bahwa LLMs terkadang menghasilkan kode yang tidak optimal atau bahkan bug tersembunyi yang dapat memengeruhi kualitas dan keberlanjutan aplikasi. Seperti yang diungkapkan oleh (Lu et al., 2021) eskipun model seperti Codex dapat menghasilkan kode yang tampak benar, kualitasnya sering kali kurang di luar scenario Latihan. Hal ini disebabkan oleh kurangnya pemahaman model terhadap konteks yang lebih luas dalam pengembangan perangakat lunak. Code Generation oleh LLMs lebih efektif jika digunakan untuk tugas-tugas sederhana atau untuk mendukung programmer dalam bagian-bagian tertentu dari proses pengkodean, tetapi penggunaan penuh dalam pengembangan perangkat lunak skala besar masih memerlukan pemantauan manusia yang cermat.

Selain itu, penggunaan LLMs dalam code generation juga menimbulkan kekhawatiran terkait etika, termasuk plagiarisme kode dan potensi bias dalam kode yang dihasilkan, sebagai contoh, model-model ini sering kali dilatih menggunakan data dari sumber terbuka yang mencakup kode repository public seperti Github. Namun, banyak kode dalam repositori tersebut mungkin tidak selalui sesuai dengan standar atau pedoman tertentu. Hal ini menyebabkan kemungkinan munculnya kode yang tidak aman atau tidak etis, yang kemudian dapat dipropogandakan tanpa pengawasan. Dalam penelitian yang dilakukan oleh (Lu et al., 2021) , mereka mencatat bahwa kode yang dihasilkan oleh model-model seperti Codex dapat mencerminkan masalah dalam data pelatihan, seperti bias atau kerentanannya terhadap serangan

security vulnerabilities. Oleh karena itu, meskipun kemajuan dalam code generation menawarkan peluang yang besar, penting untuk memastikan bahwa LLMs yang digunakan dalam bidang ini dilengkapi dengan konrol yang memadai untuk mencegah risiko yang mungkin timbul.

Penyempurnaan *Large Languange Model* melalui proses fine-tuning telah menjadi salah satu pendekatan yang paling efektif dalam meningkatkan kinerja model untuk tugas-tugas spesifik, termasuk code generation. Fine-tuning melibatkan penyesuaian parameter model yang telah dilatih sebelumnya pada dataset umum dengan dataset yang lebih berfokus pada tugas tertentu. Proses ini memungkinkan model seperti GPT-3, Codex, dan model LLM lainnya untuk beradaptasi secara efisien terhadap tantangan domain tertentu. Dengan memberikan contoh berlabel yang relevan selama fine-tuning, model dapat mengoptimalkan bobotnya lebih lanjut untuk meningkatkan kinerja pada tugas-tugas spesifik, termasuk mengasilkan kode yang kompleks dan berkualitas tinggi.

Keunggulan utama fine-tuning adalah menghindari kebutuhan melatih model dari awal, yang memerlukan sumber daya komputasi besar dan waktu pelatihan yang sangat lama. Sebagai contoh, penelitian oleh OpenAI menunjukkan bahwa fine-tuning codex pada dataset pemrograman tertentu secara signifikan meningkatkan akurasi dalam tugas-tugas spesifik sambil mengurangi kesalahan kontekstual. Namun, penting untuk melakukan fine-tuning dengan hati-hati untuk menghindari *overfitting,* yaitu kondisi Dimana model menjadi terlalu spesifik pada dataset pelatihan sehingga kehilangan fleksbilitas dalam menangani variasi data baru. Strategi seperti dropout, regularization

atau bahkan low-rank adaptation (LoRA) telah terbukti efektif dalam mengatasi overfitting selama proses fine-tuning (Lin *et al.*, 2024).

Dalam mengukur performa model dari code generation, digunakan metrik evaluasi seperti pass@k yang menjadi alat untuk mengukur performa model. pass@k mengukur apakah solusi yang benar berada di antara k prediksi teratas yang dihasilkan oleh model. Versi pass@k yang diperkenalkan oleh (Kulal et al., 2019) dan (M. Chen et al., 2024a) menunjukkan bahwa metrik ini memberikan pandangan yang lebih holistic tentang kemampuan model dalam menghasilkan solusi yang benar, terutama dalam scenario dimana terdapat banyak kemungkinan jawaban. Metrik ini menjadi semakin relevan dalam tugas pemrograman yang kompleks, dimana solusi yang benar mungkin berada di luar prediksi pertama.

Selain itu, menggunakan dataset testing seperti HumanEval menjadi standar dalam mengevaluasi kinerja model AI untuk code generation. HumanEval mencakup berbagai tingkat kesulitan, mulai dari fungsi dasar hingga algoritma *advanced*. Dalam penelitian (M. Chen et al., 2024a) , model Codex yang diuji menggunakan HumanEval menunjukkan bahwa kombinasi dataset berkualitas tinggi dan fine-tuning yang cermat dapat meningkatkan akurasi model dalam menghasilkan Solusi yang benar secara signifikan. Namun hanya mengandalkan metrik seperti pass@k tidak cukup untuk mengevaluasi pemahaman model secara kontekstual. Solusi lainnya untuk mengatasi tantang ini adalah memanfaatkan pendekatan zero-shot learning atau few shot learning, dimana model diberikan instruksi atau contoh minimal untuk menghasilkan solusi, juga dapat meningkatkan

flesibilitas model dalam menghasilkan kode yang relevan untuk tugas tertentu.

Dalam penelitian ini berfokus pada evaluasi tiga model LLMs, yaitu CodeGemma, Qwen dan Llama dalam tugas code generation. Ketiga model ini dipilih karena ukuran parameter mereka yang lebih kecil dibandingkan dengan model seperti GPT-3, sehingga menawarkan efisiensi sumber daya yang lebih tinggi. Penelitan ini bertujuan mengevaluasi kinerja ketiga model dengan menggunakan metrik pass@k. metrik ini mengukur kemampuan model dalam menghasilkan Solusi yang benar di antara k prediksi. Evaluasi dengan pass@k mencerminkan fleksibilitas dan keakuratan model, terutama dalam tugas yang memiliki berbagai Solusi valid.

Dataset yang digunakan untuk fine-tuning merupakan koleksi besar kode pythin yang dirancang untuk membantu model memahami pola bahasa pemrograman ini secara mendalam. Proses fine-tuning memungkinkan model untuk belajar dari dataset spesifik dan meningkatkan kemampuannya menghasilkan kode yang tidak benar secara sintaksis, tetapi juga bermakna secara semantic dan dioptimalkan untuk aplikasi dunia nyata. Proses ini bertujuan untuk memahami sejauh mana fine-tuning dapat membantu model, seperti CodeGemma, Qwen dan Llama beradaptasi dengan pola dan struktur bahasa pemrograman python.

Evaluasi kinerja model juga dilakukan berdasarkan benchmark HumanEval, yang telah menjadi standar dalam komunitas riset AI untuk menilai kemampuan model dalam menyelesaikan tugas pemrograman. Benchmark ini mencakup berbagai tingkat kesulitan, mulai dari implementasi fungsi sederhana hingga tugas algoritmik

yang kompleks, sehingga memberikan wawasan menyeluruh mengenai kemampuan model dalam menghasilkan kode yang benar secara fungsional dan algoritmik.

Selain akurasi, penelitian ini juga mengukur kecepatan optimasi selama proses fine-tuning. Metrik ini digunakan untuk menilai seberapa cepat setiap model dapat meningkatkan kinerjanya setelah melalui proses fine-tuning. Dengan membandingkan skor pass@k sebelum dan sesudah fine-tuning, penelitian ini mengevaluasi efektivitas pembelajaran tambahan pada model, serta mengeksplorasi trade-off antara akurasi dan waktu pelatihan.

Melalui analisis kinerja berdsasrkan metrik pass@k, dampak fine-tuning, skor HumanEval dan keceptan optimasi, penelitian ini akan memberikan pandangan komprehensif mengenai kelebihan dan kekurangan masing-masing model. Dengan demikian, penelitian ini tidak hanya akan membantu memiliki model yang paling sesuai untuk tugas tertentu, tetapi juga memberikan panduan strategis untuk pengembangan model yang lebih efisien dan efektif di masa depan. Oleh karena itu, penelitian ini diangkat dengan judul "Evaluasi Kinerja dan Efisiensi Model CodeGemma, Qwen, dan Llama pada tugas code generation dengan analisis Fine-Tuning dan Benchmark pass@k"

B. Rumusan Masalah

Berdasarkan Latar belakang di atas, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana perbandingan kinerja CodeGemma2B, LLaMA-3.2-1B dan Qwen2.5-1.5B-Instruct antara base model dan hasil

fine-tuning dalam tugas code generation menggunakan metrik fungsional (pass@k)?

2. Seberapa besar perbedaan sintaksis antara kode yang dihasilkan model dengan kode referensi HumanEval?
3. Bagaimana pengaruh fine-tuning terhadap peningkatan metrik pass@k dalam kinerja waktu eksekusi model?

C. Tujuan Penelitian

1. Mengevaluasi dan membandingkan kinerja model CodeGemma-2B, LLaMA-3.2-1B, dan Qwen2.5-1.5B-Instruct dalam menghasilkan kode program berdasarkan metrik fungsional pass@k (pass@1, pass@5, pass@10, dan pass@100) dengan menggunakan dataset HumanEval sebagai tolok ukur.
2. Mengetahui seberapa besar perbedaan sintaksis antara kode yang dihasilkan oleh ketiga model dengan kode referensi pada HumanEval, melalui evaluasi metrik sintaktik yaitu BLEU, ROUGE, ChrF, Exact Match, dan Levenshtein Distance sebagai parameter kesamaan struktur dan isi kode.

D. Manfaat Penelitian

Mengevaluasi dampak fine-tuning terhadap kinerja waktu eksekusi masing-masing model, dengan menggunakan nilai median dan percentile waktu inferensi, serta uji statistik Wilcoxon Signed-Rank Test untuk mengetahui signifikansi perbedaannya sebelum dan sesudah fine-tuning.Manfaat Penelitian

1. Bagi Peneliti dan Akademisi

a. Menyediakan wawasan terkait efektivitas model LLM dalam tugas code generation.
b. Mendorong penelitian lebih lanjut tentang dampak fine-tuning pada performa model AI

2. Bagi Developer Perangkat Lunak:
   a. Memberikan panduan dalam memilih model LLM yang paling optimal untuk menghasilkan kode berkualitas tinggi
   b. Meningkatkan efisiensi dalam proses pengembangan dengan memanfaat model LLM yang telah dioptimalkan.
3. Bagi Industri Teknologi:
   a. Membantu Perusahaan dalam mengimplementasikan Solusi code generation untuk mengurangi beban kerja developer.
   b. Mempercepat inovasi dalam alat otomatisasi pemrograman berbasis AI.

BAB V
PENUTUP

A. Kesimpulan

Berdasarkan hasil evaluasi yang dilakukan, penelitian ini berhasil mencapai tujuan yang telah ditetapkan, dengan kesimpulan sebagai berikut:

1. Evaluasi kinerja fungsional model menggunakan metrik pass@k menunjukkan keberhasilan fine-tuning dalam meningkatkan performa model, khususnya pada model dengan performa awal rendah seperti CodeGemma-2B. Nilai pass@1 CodeGemma-2B meningkat signifikan dari 0,463 menjadi 0,878 (metode Kulal) dan pass@5 dari 0,241 menjadi 0,621 (metode Chen). Sementara itu, model dengan performa tinggi seperti LLaMA-3.2-1B dan Qwen2.5-1.5B-Instruct hanya mengalami peningkatan kecil atau tidak signifikan.
2. Perbedaan sintaksis antara kode yang dihasilkan model dan kode referensi HumanEval telah berhasil dianalisis melalui metrik BLEU, ROUGE, ChrF, Exact Match, dan Levenshtein Distance. Hasilnya menunjukkan bahwa fine-tuning tidak secara signifikan meningkatkan kesamaan sintaktik, dengan BLEU dan ROUGE tetap rendah (sekitar 0,13–0,21) serta nilai Levenshtein Distance yang cukup tinggi (misalnya 102 untuk CodeGemma-2B FM dan 135 untuk Qwen2.5 FM), meskipun terdapat sedikit peningkatan pada metrik ChrF.
3. Evaluasi terhadap kinerja waktu eksekusi model menunjukkan bahwa fine-tuning tidak secara signifikan mempercepat proses

inferensi. Hal ini dibuktikan melalui uji Wilcoxon Signed-Rank Test dan analisis nilai median dan percentile, di mana CodeGemma-2B dan LLaMA-3.2-1B menunjukkan sedikit peningkatan, namun Qwen2.5 justru mengalami penurunan efisiensi waktu setelah fine-tuning.

B. Saran

Berdasarkan hasil analisis yang telah dilakukan, fine-tuning memberikan dampak yang bervariasi tergantung pada performa awal model, dengan efektivitas yang paling terlihat pada model berperforma awal rendah seperti CodeGemma-2B, sementara model yang sudah unggul seperti LLaMA3.2-1B dan Qwen2.5-1.5B-Instruct hanya mengalami peningkatan minimal. Selain itu, perbedaan sintaksis antara kode yang dihasilkan dan kode referensi HumanEval tetap signifikan, dan fine-tuning tidak secara signifikan meningkatkan kinerja waktu eksekusi model berdasarkan metrik pass@k. Oleh karena itu, terdapat beberapa rekomendasi yang dapat menjadi acuan untuk pengembangan penelitian selanjutnya, antara lain:

1. Mengeksplorasi teknik fine-tuning yang lebih canggih, seperti *adapter-based fine-tuning*, *low-rank adaptation (LoRA)*, atau *reinforcement learning from human feedback (RLHF)*, untuk model yang sudah menunjukkan performa tinggi seperti LLaMA3.2-1B dan Qwen2.5-1.5B-Instruct.
2. Mengembangkan metrik evaluasi yang lebih robust atau menggunakan pendekatan *syntax-aware fine-tuning*, seperti mengintegrasikan parser sintaksis atau *abstract syntax trees (AST)*,

untuk mengurangi perbedaan sintaksis antara kode yang dihasilkan dan kode referensi.

3. Menggunakan metrik evaluasi yang lebih khusus untuk *code generation*, seperti *CodeBLEU* atau metrik berbasis *execution accuracy*, untuk menangkap kompleksitas sintaksis dan semantik kode secara lebih akurat.
4. Menggunakan dataset yang lebih besar dan beragam, seperti MBPP (*Mostly Basic Python Problems*) atau dataset yang mencakup bahasa pemrograman lain untuk menguji generalisasi model pada berbagai konteks dan tingkat kesulitan.

DAFTAR PUSTAKA

Brown, T. B., Kaplan, J., Ryder, N., Henighan, T., Chen, M., Herbert-voss, A., Ziegler, D. M., Krueger, G., Askell, A., Hesse, C., & Mccandlish, S. (n.d.). *Language Models are Few-Shot Learners*.

Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., & Miao, Y. (2021). Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*. https://www.mdpi.com/2072-4292/13/22/4712

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). *Evaluating Large Language Models Trained on Code*. arXiv. https://doi.org/10.48550/arXiv.2107.03374

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Pinto, D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., … Welinder, P. (2024a). *Evaluating Large Language Models Trained on Code*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Pinto, D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., … Welinder, P. (2024b). *Evaluating Large Language Models Trained on Code*.

Cooper, S. B. (2004). *The Incomputable Alan Turing*. 1–16. https://doi.org/10.14236/ewic/tur2004.2

Data, T. (2024a). *CodeGemma : Open Code Models Based on Gemma*. 1–11.

Data, T. (2024b). *CodeGemma : Open Code Models Based on Gemma*. 1–11.

Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., & Lou, Y. (2023). *ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation*. http://arxiv.org/abs/2308.01861

Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., & Lou, Y. (2024). Evaluating Large Language Models in Class-Level Code Generation. *Proceedings - International Conference*

on Software Engineering, 982–994. https://doi.org/10.1145/3597503.3639219

Fakhoury, S., Naik, A., Sakkas, G., Chakraborty, S., & Lahiri, S. K. (2024). LLM-based Test-driven Interactive Code Generation: User Study and Empirical Evaluation. *IEEE Transactions on Software Engineering*, *50*(9), 2254–2268. https://doi.org/10.1109/TSE.2024.3428972

Goldberg, Y., & Levy, O. (2014). *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*. *2*, 1–5. http://arxiv.org/abs/1402.3722

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., … Ma, Z. (2024). *The Llama 3 Herd of Models*. arXiv. https://doi.org/10.48550/arXiv.2407.21783

Hadiewijaya, A., & Wasito, B. (2024). Perbandingan Efisiensi Waktu dan Memori Pada C# dan Java dengan Metode Benchmarking. *Bit-Tech*, *7*(2), 554–560. https://doi.org/10.32877/bt.v7i2.1906

Hajkowicz, S., Sanderson, C., Karimi, S., Bratanova, A., & Naughtin, C. (2023). Artificial intelligence adoption in the physical sciences, natural sciences, life sciences, social sciences and the arts and humanities: A bibliometric analysis of research publications from 1960-2021. *Technology in Society*, *74*. https://doi.org/10.1016/j.techsoc.2023.102260

Henry, A., Dachapally, P. R., Pawar, S., & Chen, Y. (2020). Query-key normalization for transformers. *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, 4246–4253. https://doi.org/10.18653/v1/2020.findings-emnlp.379

Huang, Y., Xu, J., Lai, J., Jiang, Z., Chen, T., Li, Z., Yao, Y., Ma, X., Yang, L., Chen, H., Li, S., & Zhao, P. (2023). *Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey*. *1*(1). http://arxiv.org/abs/2311.12351

Jeong, S. H., Yun, J. P., Yeom, H. G., Lim, H. J., Lee, J., & Kim, B. C. (2020). Deep learning based discrimination of soft tissue profiles requiring orthognathic surgery by facial photographs. *Scientific Reports*, *10*(1), 1–6. https://doi.org/10.1038/s41598-020-73287-7

Juhász, L. Z. (2024). *Large Language Models, Fine Tuning, Code Generation*. 191–200. https://doi.org/10.1109/CANDO-EPE65072.2024.10772760

Kaul, V. (2020). History of artificial intelligence in medicine. *Gastrointestinal Endoscopy*, *92*(4), 807–812. https://doi.org/10.1016/j.gie.2020.06.040

Kenton, M. C., Kristina, L., & Devlin, J. (1953). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. *Mlm*.

Koziolek, H., Grüner, S., Hark, R., Ashiwal, V., Linsbauer, S., & Eskandani, N. (2024). *LLM-based and Retrieval-Augmented Control Code Generation*. 22–29. https://doi.org/10.1145/3643795.3648384

Kuchaiev, O., & Ginsburg, B. (2017). Factorization tricks for LSTM networks. *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, 1–6.

Kulal, S., Pasupat, P., Chandra, K., Lee, M., Padon, O., Aiken, A., & Liang, P. (2019). SPoC: Search-based pseudocode to code. *Advances in Neural Information Processing Systems*, *32*, 1–11.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Li, D., & Murr, L. (2024). *HumanEval on Latest GPT Models -- 2024*. http://arxiv.org/abs/2402.14852

Lian, X., Wang, S., Ma, J., Tan, X., Liu, F., Shi, L., Zhang, L., & Gao, C. (2024). Imperfect Code Generation: UncoveringWeaknesses in Automatic Code Generation by Large Language Models. *Proceedings - International Conference on Software Engineering*, 422–423. https://doi.org/10.1145/3639478.3643081

Lin, Y.-C., Kumar, A., Chang, N., Zhang, W., Zakir, M., Apte, R., He, H., Wang, C., & Jang, J.-S. R. (2023). Novel Preprocessing Technique for Data Embedding in Engineering Code Generation Using Large Language Model. *2024 IEEE LLM Aided Design Workshop (LAD)*, 1–5. https://doi.org/10.1109/LAD62341.2024.10691715

Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., Sundaresan, N., … Liu, S. (2021). *CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation*. http://arxiv.org/abs/2102.04664

Luo, H., Wu, J., Liu, J., & Antwi-Afari, M. F. (2024). Large language model-based code generation for the control of construction assembly robots: A hierarchical generation approach. *Developments in the Built*

Environment, *19*(March), 100488. https://doi.org/10.1016/j.dibe.2024.100488

Mahowald, K., Ivanova, A. A., Blank, I. A., Kanwisher, N., Tenenbaum, J. B., & Fedorenko, E. (2024). Dissociating language and thought in large language models. *Trends in Cognitive Sciences*, *28*(6), 517–540. https://doi.org/10.1016/j.tics.2024.01.011

Mitchell, M., & Krakauer, D. C. (2023). The debate over understanding in AI's large language models. *Proceedings of the National Academy of Sciences of the United States of America*, *120*(13), 1–13. https://doi.org/10.1073/pnas.2215907120

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). BLEU: a method for automatic evaluation of machine translation. *The 40th Annual Meeting*, 311. https://doi.org/10.3115/1073083.1073135

Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2249–2255. https://doi.org/10.18653/v1/d16-1244

Qiu, Z., Liu, W., Feng, H., Liu, Z., Xiao, T. Z., Collins, K. M., Tenenbaum, J. B., Weller, A., Black, M. J., & Schölkopf, B. (2024). *Can Large Language Models Understand Symbolic Graphics Programs?* 1–44. http://arxiv.org/abs/2408.08313

Ren, S., Guo, D., Lu, S., Zhou, L., Liu, S., Tang, D., Sundaresan, N., Zhou, M., Blanco, A., & Ma, S. (2020). *CodeBLEU: a Method for Automatic Evaluation of Code Synthesis*. arXiv. https://doi.org/10.48550/arXiv.2009.10297

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866. https://doi.org/10.1162/tacl_a_00349

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., … Synnaeve, G. (2024). *Code Llama: Open Foundation Models for Code*. arXiv. https://doi.org/10.48550/arXiv.2308.12950

Sanabria-Navarro, J.-R., Silveira-Pérez, Y., Pérez-Bravo, D.-D., & Cortina-Núñez, M. de J. (2023). Incidences ofartificial intelligence in contemporary education - Incidencias de la inteligencia artificial en la educación contemporánea. *Media Education Research Journal*, *77*, 93–103.

Saratchandran, H., Zheng, J., Ji, Y., Zhang, W., & Lucey, S. (2024). *Rethinking Softmax: Self-Attention with Polynomial Activations*. 1–22. http://arxiv.org/abs/2410.18613

Sharma, R., Govind, D., Mishra, J., Dubey, A. K., Deepak, K. T., & Prasanna, S. R. M. (2024). *Milestones in speaker recognition* (Vol. 57). Springer Netherlands. https://doi.org/10.1007/s10462-023-10688-w

Sparrow, R. (2020). The turing triage test. *Machine Ethics and Robot Ethics*, 95–105. https://doi.org/10.4324/9781003074991-11

Strubell, E., Ganesh, A., & McCallum, A. (2020). Energy and policy considerations for modern deep learning research. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, *1*, 1393–13696. https://doi.org/10.1609/aaai.v34i09.7123

Thomas, D. O. (2022). Beyond Disciplinary Drama: Federal Dollars, ESL Instruction for African Americans, and Public Memory. *College Composition and Communication*, *73*(1), 52–79. https://doi.org/10.58680/ccc202131587

Tomas Mikolov, Wen-tau Yih, G. Z. (2013). Linguistic Regularities in Continuous Space Word Representations. *Association for Computational Linguistics*, *26*(5), 505–513. https://doi.org/10.3109/10826089109058901

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv. https://doi.org/10.48550/arXiv.2302.13971

Tsai, Y. H. H., Srivastava, N., Goh, H., & Salakhutdinov, R. (2020). Capsules With Inverted Dot-Product Attention Routing. *8th International Conference on Learning Representations, ICLR 2020*, 1–15.

Vaswani, A. (2017). *Attention Is All You Need*. *Nips*.

Vijayaraghavan, P., Shi, L., Ambrogio, S., Mackin, C., Nitsure, A., Beymer, D., & Degan, E. (2024). VHDL-Eval: A Framework for Evaluating Large Language Models in VHDL Code Generation. *2024 IEEE LLM Aided Design Workshop (LAD)*, 1–6. https://doi.org/10.1109/LAD62341.2024.10691836

Wang, J., Guo, B., & Chen, L. (2022). Human-in-the-loop Machine Learning: A Macro-Micro Perspective. *ArXiv (Cornell University)*. https://doi.org/2202.10564

Wang, J., Huang, J. X., Tu, X., Wang, J., Huang, A. J., Laskar, M. T. R., & Bhuiyan, A. (2024). Utilizing BERT for Information Retrieval: Survey,

Applications, Resources, and Challenges. *ACM Computing Surveys*, *56*(7), 1–33. https://doi.org/10.1145/3648471

Yang, S., & Yang, Y. (2024a). FormalEval: A Method for Automatic Evaluation of Code Generation via Large Language Models. *2024 International Symposium of Electronics Design Automation, ISEDA 2024*, 660–665. https://doi.org/10.1109/ISEDA62518.2024.10617643

Yang, S., & Yang, Y. (2024b). FormalEval: A Method for Automatic Evaluation of Code Generation via Large Language Models. *2024 International Symposium of Electronics Design Automation, ISEDA 2024*, 660–665. https://doi.org/10.1109/ISEDA62518.2024.10617643

YIKUN, L. (2024). Review on natural language processing models. *Applied and Computational Engineering*, *35*(1), 1–7. https://doi.org/10.54254/2755-2721/35/20230350

Zhang, Y., Yu, Z., Fu, Y., Wan, C., & Lin, Y. (2024). MG-Verilog: Multi-grained Dataset Towards Enhanced LLM-assisted Verilog Generation. *2024 IEEE LLM Aided Design Workshop, LAD 2024*, 1–5. https://doi.org/10.1109/LAD62341.2024.10691738

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E.-V., & Batista-Navarro, R. T. (2020). *Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study*. *v*. http://arxiv.org/abs/2004.01099

Zheng, Y., Zhang, R., Zhang, J., Ye, Y., Luo, Z., Feng, Z., & Ma, Y. (2024). *LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models*. arXiv. https://doi.org/10.48550/arXiv.2403.13372