BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi kecerdasan buatan (*Artificial Intelligence*) dan pembelajaran mendalam (*Deep Learning*) telah membawa kemajuan signifikan dalam pemrosesan media digital. Salah satu produk dari kemajuan ini adalah teknologi *Deepfake*, yang memanfaatkan *Generative Adversarial Networks* (*GANs*) untuk memanipulasi atau mensintesis wajah manusia dalam video dan gambar dengan tingkat realisme yang sangat tinggi. Meskipun teknologi ini memiliki potensi positif dalam industri hiburan, penyalahgunaannya telah menimbulkan ancaman serius terhadap integritas informasi, keamanan biometrik, dan privasi individu. Penyebaran konten *deepfake* yang digunakan untuk penipuan, pencemaran nama baik, hingga penyebaran berita bohong (*hoax*) menjadi masalah mendesak yang memerlukan solusi teknis.

Saat ini, berbagai metode deteksi deepfake telah dikembangkan oleh peneliti, mencakup pendekatan *Deep Spatial* untuk analisis artefak visual, *Deep Temporal* untuk inkonsistensi antar-frame, deteksi *Liveness* untuk verifikasi tanda kehidupan, hingga deteksi *Hybrid* yang menggabungkan *multiple features*. Namun, sebagian besar model deteksi ini masih beroperasi secara terisolasi dalam lingkungan pengembangan lokal atau skrip eksperimental, sehingga sulit diakses oleh masyarakat umum yang awam terhadap teknologi pemrograman.

Di lingkungan akademis Universitas AMIKOM Yogyakarta, telah dikembangkan empat model deteksi *deepfake* individual sebagai bagian dari penelitian kolaboratif mahasiswa Program Studi Teknik Komputer. Penelitian pertama mengembangkan *Temporal Detector* berbasis *Xception CNN* dan *GRU* untuk analisis pola visual antar-frame. Penelitian kedua membangun *Spatial Detector* menggunakan arsitektur *XceptionNet* untuk mendeteksi artefak *spasial* pada video *deepfake*. Penelitian ketiga mengimplementasikan *Liveness Detector dual-model* berbasis *rPPG* dan pola kedipan mata menggunakan *deep learning*.

Penelitian keempat mengembangkan *Hybrid Detector* berbasis *XceptionNet* dengan kemampuan mendeteksi *deepfake* yang telah mengalami manipulasi *post-processing*. Meskipun keempat model tersebut memiliki akurasi tinggi dalam domain masing-masing, model-model ini belum terintegrasi dalam sebuah platform yang mudah diakses dan siap produksi.

Tantangan utama dalam mengintegrasikan keempat model deteksi ini adalah kompleksitas komputasi dan konflik dependensi *framework*. *Temporal Detector* dan *Hybrid Detector* menggunakan *TensorFlow* 2.15, *Spatial Detector* menggunakan *TensorFlow* 2.20 dengan Keras 3.0, sementara *Liveness Detector* menggunakan *PyTorch* 2.1. Mengintegrasikan model-model dengan dependensi berbeda ke dalam satu aplikasi web monolitik sering kali menyebabkan masalah kinerja, seperti tingginya *latency*, konsumsi sumber daya yang tidak efisien, dan risiko kegagalan sistem total jika salah satu modul mengalami *error* (*single point of failure*). Selain itu, tidak adanya *user interface* yang intuitif membuat model-model tersebut hanya dapat digunakan oleh individu dengan kemampuan *programming*, bukan oleh masyarakat umum.

Untuk mengatasi tantangan tersebut, diperlukan sebuah arsitektur sistem yang modular dan skalabel yang dapat mengintegrasikan keempat model dari penelitian rekan tim. *Arsitektur Microservices* menawarkan solusi dengan memecah aplikasi menjadi layanan-layanan kecil yang *independen*, memungkinkan setiap model deteksi berjalan pada kontainernya masing-masing dengan *dependencies* terisolasi. Pendekatan ini, dikombinasikan dengan kerangka kerja modern seperti *FastAPI* yang mendukung pemrosesan asinkron berkinerja tinggi di sisi *backend*, dan *Next.js* yang menawarkan interaktivitas tinggi di sisi *frontend*, diharapkan mampu menghasilkan *platform* deteksi yang handal, responsif, dan *accessible* untuk publik. Implementasi *ensemble* learning pada *API Gateway* juga diharapkan dapat meningkatkan akurasi deteksi dengan menggabungkan kekuatan dari keempat model individual.

Berdasarkan permasalahan di atas, penelitian ini mengangkat judul "Pengembangan dan Analisis Kinerja *Web Platform* Deteksi *Deepfake* Wajah

Berbasis *Microservices*". Penelitian ini merupakan kelanjutan dari empat penelitian model individual yang dilakukan oleh rekan tim, dengan fokus utama pada integrasi keempat model deteksi ke dalam sebuah ekosistem web yang dapat diakses publik, pengembangan *ensemble learning mechanism* untuk meningkatkan akurasi, perancangan *user interface* yang intuitif, serta analisis bagaimana arsitektur *microservices* mempengaruhi kinerja sistem dalam skenario penggunaan nyata.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan di atas, maka dapat dirumuskan permasalahan penelitian sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan arsitektur *microservices* berbasis *Docker* untuk mengintegrasikan *multi-model deep learning* dengan dependensi *framework* yang berbeda?
2. Bagaimana mengimplementasikan metode *ensemble learning* dengan *weighted average* dan *majority voting* untuk meningkatkan akurasi deteksi *deepfake*?
3. Bagaimana kinerja sistem dari segi waktu respon, penggunaan sumber daya, dan *fault tolerance* dalam arsitektur *microservices*?
4. Bagaimana membangun *platform web production-ready* dengan antarmuka pengguna yang intuitif dan mudah diakses untuk deteksi *deepfake*?

1.3 Batasan Masalah

Agar penelitian ini lebih terarah dan fokus, maka penulis menetapkan batasan-batasan masalah sebagai berikut:

1. Objek Deteksi: Penelitian difokuskan pada deteksi manipulasi wajah (*Face Swap atau Face Reenactment*) pada media video dan gambar.
2. Arsitektur Sistem: Sistem dibangun menggunakan arsitektur *Microservices* dengan komunikasi antar layanan melalui *REST API*

(HTTP).

3. Teknologi: Dari sisi teknologi, penelitian ini menggunakan kerangka kerja *Next.js* untuk pengembangan *frontend* yang menyediakan antarmuka pengguna yang interaktif dan responsif, sedangkan untuk *backend* menggunakan *Python* dengan *framework FastAPI* yang mendukung pemrosesan asinkron berkinerja tinggi, dimana seluruh layanan *backend* dikemas dalam kontainer *Docker* untuk memastikan isolasi dependensi dan konsistensi *deployment* di berbagai *environment*.
4. *Model AI*: Penelitian ini menggunakan empat pendekatan model deteksi utama yang masing-masing memiliki spesialisasi berbeda, yaitu *Temporal Detector* untuk menganalisis urutan frame dan mendeteksi inkonsistensi *temporal* dalam gerakan wajah, *Spatial Detector* untuk menganalisis artefak visual per frame seperti artefak pencampuran dan inkonsistensi warna, *Liveness Detector* untuk menganalisis tanda-tanda kehidupan melalui sinyal fisiologis menggunakan *rPPG* dan pola kedipan mata, serta *Hybrid Detector* yang menggabungkan fitur *spasial* dan *temporal* dengan mekanis *multi-frame voting* untuk menghasilkan prediksi yang lebih tangguh.
5. Lingkungan Pengujian: Pengujian kinerja dilakukan pada lingkungan lokal (*Local Environment*) menggunakan *Docker Compose*.
6. *Platform*: *Frontend* menggunakan *Next.js*, belum mencakup *mobile application*. *Deployment* menggunakan *Cloudflare Zero Trust Tunnel* untuk *expose development environment* ke public internet tanpa *VPS* atau *cloud server* berbayar.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan, tujuan yang akan dicapai oleh peneliti dalam penelitian ini adalah:

1. Mengembangkan *platform web* deteksi *deepfake* berbasis arsitektur *microservices*.

2. Mengimplementasikan metode *ensemble learning* dengan *weighted average* dan *majority voting*.
3. Menganalisis kinerja sistem dari segi waktu respons dan penggunaan sumber daya.
4. Menyediakan antarmuka pengguna yang mudah digunakan untuk deteksi *deepfake*

1.5 Manfaat Penelitian

Penelitian ini memberikan kontribusi signifikan dalam tiga aspek utama yang mencakup dimensi akademis, praktis, dan teknis, dengan harapan dapat memberikan dampak positif bagi pengembangan ilmu pengetahuan dan implementasi teknologi deteksi *deepfake* di masa depan:

1.5.1 Manfaat Akademis

1. Menyediakan studi kasus komprehensif tentang implementasi arsitektur *microservices* untuk sistem *AI* kompleks yang mengintegrasikan *multiple model deep learning* dengan dependensi berbeda, melengkapi literatur akademis tentang *deployment model machine learning* dalam arsitektur terdistribusi.
2. Memberikan analisis empiris mendalam tentang efektivitas metode *ensemble learning* (*weighted average* dan *majority voting*) dalam konteks deteksi *deepfake*, termasuk perbandingan kuantitatif antara performa model individual versus *ensemble* untuk memvalidasi hipotesis peningkatan akurasi melalui kombinasi *multi-model*.
3. Menyediakan dokumentasi lengkap tentang solusi praktis untuk mengatasi konflik dependensi antar model *deep learning* yang menggunakan versi *framework* berbeda (*TensorFlow* 2.15, 2.20, dan *PyTorch*), yang dapat menjadi referensi bagi peneliti lain dalam menghadapi tantangan serupa.

4. Menghasilkan *framework* evaluasi kinerja sistem *microservices* untuk aplikasi *AI* yang mencakup metrik *latency, throughput, resource utilization,* dan *scalability,* memberikan kontribusi metodologis untuk penelitian arsitektur sistem *AI* di masa depan.

1.5.2 Manfaat Praktis

1. Menghasilkan *platform* siap produksi berbasis web yang dapat diakses dan digunakan langsung oleh masyarakat umum, lembaga pemerintah, media, dan organisasi untuk melakukan verifikasi keaslian konten visual, membantu memerangi penyebaran disinformasi dan konten manipulatif.
2. Menyediakan *template* arsitektur lengkap beserta kode implementasi yang dapat diadaptasi oleh pengembang dan tim *engineering* yang membangun sistem *AI* serupa, menghemat waktu dan biaya *development* dengan memanfaatkan *best practices* yang telah tervalidasi.
3. Memberikan panduan implementasi Langkah demi langkah untuk isolasi model dengan versi *framework* berbeda menggunakan kontainerisasi *Docker*, memfasilitasi *deployment multi-model* tanpa konflik dependensi dalam *environment production*.
4. Menyediakan solusi skalabel yang dapat di-deploy pada berbagai *environment* (lokal, VPS, cloud) dengan sumber daya terbatas, menjadikannya mudah diakses untuk institusi pendidikan, startup, dan organisasi non-profit yang memiliki keterbatasan infrastruktur.

1.5.3 Manfaat Teknis

1. Mengimplementasikan *API Gateway pattern* dengan *smart ensemble logic* yang rumit, mendemonstrasikan teknik advanced untuk agregasi prediksi dari *multiple AI services* dengan mempertimbangkan skor keandalan, analisis kesepakatan, dan mekanisme cadangan.
2. Menyediakan strategi pemeriksaan kesehatan sistem dan mekanisme Cadangan yang tangguh untuk meningkatkan reliabilitas sistem,

memastikan ketersediaan layanan tinggi dan pengoprasian mode degradasi ketika beberapa detection services mengalami kegagalan.

3. Mendemonstrasikan penggunaan kontainerisasi *Docker* untuk memisahkan model dengan dependensi konflik (*TensorFlow* 2.15 dan 2.20) dalam arsitektur *microservices*, memberikan solusi praktis untuk masalah konflik dependensi yang sering dihadapi dalam *deployment AI systems*.
4. Mengimplementasikan pemrosesan asinkron paralel menggunakan *Python asyncio* untuk optimasi *latency*, memanfaatkan eksekusi bersamaan untuk meningkatkan *throughput* sistem hingga 2.8x dibandingkan pemrosesan berurutan.
5. Menyediakan contoh implementasi *dual-model fusion* dalam *Liveness Detection* (*rPPG + Blink Detection*) dengan strategi kombinasi weighted, memberikan wawasan tentang model teknik fusion untuk meningkatkan kekuatan deteksi.
6. Menghasilkan dokumentasi teknis lengkap (*API documentation*, *architecture diagrams*, *deployment guides*) yang dapat dijadikan implementasi refrensi untuk pengembangan platform AI microservices yang sejenis.

1.6 Sistematika Penulisan

Untuk memberikan gambaran yang jelas dan terstruktur mengenai penyusunan skripsi ini, maka materi pembahasan dibagi menjadi lima bab dengan uraian singkat sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan dasar pemikiran dilakukannya penelitian. Di dalamnya mencakup latar belakang masalah mengenai maraknya *deepfake* dan kebutuhan sistem deteksi yang handal, rumusan masalah terkait integrasi multi-model dalam arsitektur *microservices*, batasan masalah untuk menjaga fokus

penelitian pada implementasi sistem dan analisis kinerja, tujuan yang ingin dicapai, manfaat penelitian bagi berbagai pihak, serta sistematika penulisan laporan.

BAB II TINJAUAN PUSTAKA

Bab ini memaparkan teori-teori pendukung dan tinjauan pustaka yang relevan dengan topik penelitian. Materi yang dibahas meliputi konsep dasar *Deep Learning* dan teknologi *Deepfake*, karakteristik empat model deteksi (*Spatial, Temporal, Liveness, Hybrid*), konsep arsitektur perangkat lunak *Microservices* dibandingkan dengan *Monolithic*, mekanisme komunikasi REST API, serta teknologi pengembangan yang digunakan yaitu kerangka kerja *frontend* Next.js (berbasis React) dan *backend* FastAPI (berbasis Python). Bab ini juga membahas penelitian terdahulu sebagai pembanding.

BAB III METODE PENELITIAN

Bab ini menguraikan tahapan-tahapan yang dilakukan dalam penyelesaian masalah. Pembahasan dimulai dari alur penelitian, analisis kebutuhan sistem baik fungsional maupun non-fungsional, serta perancangan sistem yang mencakup desain topologi *microservices* dengan komunikasi antar layanan dan *API Gateway*, perancangan logika menggunakan diagram *UML* seperti *Use Case Diagram, Activity Diagram, dan Sequence Diagram*, perancangan antarmuka berupa desain *wireframe* atau mockup antarmuka pengguna pada *Next.js*, serta skenario pengujian yang meliputi rencana pengujian kinerja untuk mengukur latensi dan *throughput* serta pengujian fungsionalitas sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini merupakan inti dari penelitian yang memuat realisasi dari perancangan yang telah dibuat. Pembahasan dimulai dengan implementasi yang menjelaskan lingkungan pengembangan, penulisan kode program untuk layanan backend *FastAPI* dengan integrasi model *deepfake*, pembangunan antarmuka *frontend Next.js*, serta proses *deployment* dan integrasi sistem secara keseluruhan. Selanjutnya disajikan data hasil pengujian sistem yang mencakup pengujian

fungsional menggunakan *Black Box Testing* dan pengujian kinerja menggunakan *Load Testing* untuk mengukur waktu respons *API* saat menangani berbagai model deteksi. Bab ini ditutup dengan pembahasan yang menganalisis hasil pengujian untuk mengevaluasi efektivitas arsitektur *microservices* dalam menangani beban kerja deteksi *deepfake*.

BAB V PENUTUP

Bab ini berisi kesimpulan yang diambil berdasarkan hasil implementasi dan analisis data yang telah dilakukan, serta memberikan saran-saran konstruktif untuk pengembangan sistem deteksi *deepfake* di masa mendatang agar lebih optimal.