BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian pengembangan dan analisis kinerja web platform deteksi deepfake wajah berbasis microservices, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Arsitektur Microservices untuk Multi-Model Deep Learning

Penelitian ini berhasil merancang dan mengimplementasikan arsitektur microservices yang efisien untuk mengintegrasikan empat model deep learning dengan dependensi framework yang berbeda. Containerization menggunakan Docker memungkinkan isolasi sempurna antara Temporal Service (TensorFlow 2.15), Spatial Service (TensorFlow 2.20), Hybrid Service (TensorFlow 2.15), dan Liveness Service (PyTorch 2.1) tanpa konflik dependensi. API Gateway dengan FastAPI berfungsi sebagai orchestrator yang mengkoordinasikan komunikasi antar-service melalui REST API dengan pattern asynchronous parallel processing yang meningkatkan throughput hingga 2.8x dibandingkan sequential processing.

2. Implementasi Ensemble Learning yang Efektif

Metode ensemble learning dengan algoritma 7-phase smart ensemble yang mengintegrasikan weighted average dan majority voting berhasil meningkatkan akurasi deteksi dari 89% (best single model: Liveness) menjadi 91% pada dataset testing. Ensemble juga mengurangi false negative dari 6-7 menjadi 5 dan false positive dari 5-6 menjadi 4, menunjukkan peningkatan robustness yang signifikan. Reliability scores yang diterapkan (Liveness: 0.88, Temporal: 0.78, Spatial: 0.72, Hybrid: 0.55) memungkinkan sistem memberikan bobot lebih tinggi pada model yang lebih reliable untuk menghasilkan prediksi final yang lebih akurat. Perbandingan dengan metode ensemble sederhana menunjukkan bahwa 7-phase algorithm menghasilkan akurasi 91% dibandingkan weighted average sederhana 88% dan majority voting 86%.

3. Kinerja Sistem yang Acceptable

Analisis kinerja sistem menunjukkan hasil yang acceptable untuk use case batch processing. Waktu respons rata-rata untuk video 30 detik adalah 59.67 detik dengan range 55.9-63.4 detik, menunjukkan konsistensi yang baik dengan success rate 100% (10/10 requests). Resource utilization sangat efisien dengan peak memory usage 4.80 GB dan CPU usage rata-rata 136.18% (~1.4 cores), membuktikan bahwa sistem dapat berjalan pada hardware dengan resource terbatas. Bottleneck analysis mengidentifikasi bahwa Spatial Service memiliki waktu inference terpanjang (rata-rata 33.84s) karena processing multiple frames dengan resolusi 299x299 pixels, diikuti oleh Liveness Service (15.87s), Temporal Service (6.89s), dan Hybrid Service (3.07s).

4. Antarmuka Pengguna yang Intuitif

Platform menyediakan antarmuka web berbasis Next.js 14 yang intuitif dan user-friendly dengan fitur drag-and-drop upload, real-time progress indication, dan visualisasi hasil deteksi yang comprehensive. Per-model breakdown menampilkan confidence score individual dari setiap model detection (Temporal, Spatial, Liveness, Hybrid) beserta ensemble decision reasoning yang membantu users memahami basis keputusan sistem. Responsive design memastikan aksesibilitas optimal pada berbagai devices dari desktop hingga mobile.

5. Kontribusi terhadap State-of-the-Art

Dibandingkan dengan penelitian terdahulu, platform ini memberikan kontribusi signifikan dalam beberapa aspek. Dari segi deployment, penelitian ini menghasilkan fully containerized platform dengan Docker Compose orchestration, berbeda dengan penelitian terdahulu yang hanya menyediakan research code. Dari segi scalability, implementasi microservices architecture memungkinkan independent service scaling, addressing concern yang tidak dibahas oleh penelitian sebelumnya. Dari segi user accessibility, platform

menyediakan comprehensive web interface dan well-documented REST API dengan OpenAPI specification untuk easy integration, sementara penelitian terdahulu tidak menyediakan UI maupun API. Dari segi fault tolerance, sistem mengimplementasikan graceful degradation dan fallback mechanisms yang memastikan robust operation meskipun beberapa services mengalami failures.

6. Deployment dengan Cloudflare Tunnel

Sistem development berhasil di-expose ke public internet menggunakan Cloudflare Zero Trust Tunnel (cloudflared) tanpa memerlukan VPS atau cloud server berbayar. Deployment strategy ini menyediakan secure connection dengan automatic HTTPS encryption, built-in DDoS protection, dan simplified setup tanpa perlu konfigurasi firewall kompleks atau port forwarding, memungkinkan development environment diakses secara online untuk testing dan demonstrasi dengan cost-effective solution.

Secara keseluruhan, penelitian ini membuktikan bahwa arsitektur microservices adalah solusi yang viable dan efektif untuk deployment multi-model deep learning system dengan benefit isolasi dependensi, independent scalability, fault tolerance, dan maintainability yang tinggi. Platform yang dihasilkan tidak hanya functional dan performant, tetapi juga production-ready dengan containerization lengkap, comprehensive error handling, health monitoring, dan antarmuka yang accessible untuk end users.

5.2 Saran

Berdasarkan hasil penelitian dan analisis yang telah dilakukan, terdapat beberapa saran untuk pengembangan dan perbaikan sistem di masa mendatang:

1. Optimisasi Kinerja dan Latency

Spatial Service yang menjadi bottleneck dengan waktu inference 33.84 detik perlu dioptimasi melalui beberapa pendekatan. Frame sampling strategy dapat diterapkan untuk memproses subset representatif dari frames daripada memproses semua frames, potentially reducing inference time hingga 50% tanpa

sacrificing significant accuracy. Model quantization menggunakan TensorFlow Lite atau ONNX Runtime dapat mengurangi model size dan meningkatkan inference speed hingga 2-4x dengan minimal accuracy degradation. GPU acceleration dengan CUDA dapat significantly meningkatkan throughput untuk CNN-based models seperti Spatial dan Temporal Detector. Caching mechanism untuk frames yang sudah diproses dapat mengurangi redundant computation untuk video yang sama.

2. Implementasi Horizontal Scaling

Untuk meningkatkan throughput dan menangani concurrent users yang lebih banyak, implementasi horizontal scaling sangat disarankan. Kubernetes orchestration dapat di-setup untuk automated scaling berdasarkan load metrics, memungkinkan sistem automatically menambahkan replicas dari bottleneck services (terutama Spatial Service) ketika traffic meningkat. Load balancer seperti NGINX atau Traefik dapat mendistribusikan requests ke multiple service replicas untuk optimal resource utilization. Service mesh seperti Istio dapat diimplementasikan untuk advanced traffic management, circuit breaking, dan observability yang comprehensive.

3. Enhancement Security dan Production Readiness

Untuk deployment production yang aman, beberapa aspek security perlu diimplementasikan. HTTPS enforcement dengan SSL/TLS certificates untuk encrypted communication harus dikonfigurasi, meskipun Cloudflare Tunnel sudah menyediakan HTTPS secara default. Authentication dan Authorization mechanisms seperti JWT (JSON Web Tokens) atau OAuth 2.0 perlu ditambahkan untuk membatasi akses API hanya untuk authorized users. Rate limiting dan request throttling harus diimplementasikan untuk preventing abuse dan DDoS attacks, misalnya membatasi 10-20 requests per user per hour. Input validation yang lebih strict untuk file uploads perlu diterapkan termasuk virus scanning, file type verification, dan content validation untuk mencegah malicious uploads.

4. Monitoring dan Observability Infrastructure

Centralized logging menggunakan ELK stack (Elasticsearch, Logstash, Kibana) atau alternatives seperti Loki dengan Grafana sangat disarankan untuk aggregating logs dari semua services dalam single dashboard untuk easier troubleshooting. Metrics collection menggunakan Prometheus dapat memonitor performance metrics seperti request rate, latency distribution, error rate, dan resource utilization dalam real-time. Distributed tracing dengan Jaeger atau Zipkin dapat membantu mengidentifikasi bottlenecks dalam request flow across multiple services. Alerting system dengan Alertmanager dapat mengirimkan notifications ketika metrics melewati threshold tertentu (misalnya error rate > 5%, latency > 2 minutes) untuk proactive issue resolution.

5. Pengembangan Fitur Tambahan

Batch processing API endpoint dapat ditambahkan untuk memungkinkan users mengunggah multiple videos sekaligus dan memproses secara asynchronous dengan job queue menggunakan Celery atau RabbitMQ. Video quality analysis dapat diimplementasikan untuk memberikan confidence score yang different berdasarkan kualitas input video (resolusi, compression, lighting conditions). Historical analytics dashboard dapat menampilkan statistics seperti total videos processed, detection accuracy over time, most common deepfake types detected, dan usage patterns. API webhooks dapat ditambahkan untuk notifying external systems ketika detection selesai, memungkinkan integration dengan workflow automation tools.

6. Ekspansi Model dan Dataset

Fine-tuning models dengan dataset yang lebih diverse termasuk deepfake generation methods yang lebih recent (seperti StyleGAN3, Diffusion Models) dapat meningkatkan generalization capability. Cross-dataset evaluation menggunakan multiple benchmark datasets (FaceForensics++, Celeb-DF, DFDC, WildDeepfake) dapat memvalidasi robustness model terhadap berbagai jenis manipulasi. Adversarial training dapat diterapkan untuk meningkatkan resilience terhadap adversarial attacks yang specifically crafted untuk evading detection.

Transfer learning dari models yang telah dilatih pada domain related dapat meningkatkan performance tanpa memerlukan training from scratch.

7. User Experience Improvements

Real-time detection preview untuk video uploads dapat diimplementasikan dengan progressive result updates saat processing berlangsung rather than menunggu hingga seluruh video selesai diproses. Detailed explanation untuk setiap prediction dapat ditambahkan dengan highlighting regions of interest yang menunjukkan visual artifacts detected, temporal inconsistencies, atau liveness anomalies. Comparison mode dapat memungkinkan users mengunggah dua videos untuk side-by-side comparison dan detection. Export functionality untuk generating detailed PDF reports dengan analysis results, confidence scores, dan visual evidence untuk documentation atau forensic purposes.

8. Infrastructure Cost Optimization

Cloud deployment di platform seperti AWS, Google Cloud, atau Azure dengan auto-scaling capabilities dapat dipertimbangkan untuk production workloads yang memerlukan high availability dan global distribution. Serverless architecture menggunakan AWS Lambda atau Google Cloud Functions untuk infrequently used services dapat mengurangi cost dengan pay-per-use pricing model. Model serving optimization dengan TensorFlow Serving atau TorchServe dapat providing more efficient inference infrastructure dibandingkan custom FastAPI implementation. CDN integration untuk static assets dan frequently accessed results dapat reducing latency dan server load.

Implementasi saran-saran di atas secara bertahap akan menghasilkan sistem yang lebih robust, scalable, secure, dan user-friendly yang siap untuk deployment production dengan traffic volume tinggi dan use cases yang lebih diverse dalam combating the growing threat of deepfake technology.