ANALISIS KLASIFIKASI PERINTAH SUARA BERBASIS MACHINE LEARNING TERDISTRIBUSI MENGGUNAKAN APACHE SPARK

Dameria Naomi Simatupang, Marzuqi Rouf,
Sri Lestari Ratna Anugraheni, Eka Ardhianto, Kristiawan Nugroho
Program Studi Teknologi Informasi, Fakultas Teknologi Informasi dan Industri, Universitas Stikubank
Jalan Tri Lomba Juang No.1 Semarang, Indonesia
damerianaomi0016@mhs.unisbank.ac.id

ABSTRAK

Penelitian ini mengenai klasifikasi perintah suara terus berkembang seiring meningkatnya kebutuhan sistem interaksi manusia dan komputer pada perangkat berbasis kecerdasan buatan. Dataset audio berskala besar seperti Google Speech Commands menimbulkan tantangan komputasi terutama pada tahap pra-pemrosesan dan ekstraksi fitur. Penelitian ini bertujuan untuk menganalisis penerapan machine learning terdistribusi menggunakan Apache Spark dalam proses klasifikasi perintah suara. Metode yang digunakan meliputi ekstraksi fitur Mel-Frequency Cepstral Coefficients (MFCC), normalisasi fitur menggunakan StandardScaler, serta klasifikasi menggunakan algoritma Logistic Regression dan Random Forest pada Spark MLlib. Dataset yang digunakan adalah Google Speech Commands yang terdiri dari lebih dari 64.000 file audio. Evaluasi model dilakukan menggunakan metrik akurasi serta analisis visualisasi data menggunakan Principal Component Analysis (PCA). Hasil penelitian menunjukkan bahwa Apache Spark mampu memproses dataset audio skala besar secara efisien dalam lingkungan komputasi terdistribusi. Model Logistic Regression memperoleh akurasi sebesar 25,72%, sedangkan Random Forest sebesar 18,07%. Hasil ini menunjukkan bahwa penggunaan fitur MFCC berbasis rata-rata masih memiliki keterbatasan dalam merepresentasikan karakteristik temporal sinyal suara.

***Kata Kunci**: Apache Spark, Klasifikasi Suara, Machine Learning Terdistribusi, MFCC, Big Data*

1. PENDAHULUAN

Pengenalan suara adalah salah satu domain penting dalam kecerdasan buatan yang banyak diterapkan pada asisten virtual, sistem kontrol suara, dan perangkat *Internet of Things* (IoT) [1]. Kemampuan sistem untuk memahami perintah suara secara akurat menjadi faktor penentu kualitas interaksi manusia-mesin, sehingga penelitian di bidang ini terus berkembang.

Seiring meningkatnya volume data audio dan kompleksitas sinyal suara, pendekatan pemrosesan konvensional menjadi kurang efisien dan sulit untuk diskalakan [2]. Dataset *Google Speech Commands* menyediakan sejumlah besar data audio berlabel dengan berbagai variasi suara dan aksen, sehingga sangat cocok untuk penelitian klasifikasi perintah suara. Namun, proses ekstraksi fitur seperti *Mel-Frequency Cepstral Coefficients* (MFCC) membutuhkan sumber daya komputasi yang signifikan, terutama untuk dataset skala besar [3].

Pendekatan pemrosesan terdistribusi menjadi sangat penting untuk meningkatkan efisiensi dan skalabilitas sistem. Dengan menggunakan komputasi paralel, sistem dapat menangani dataset besar secara lebih cepat tanpa membebani memori lokal. Apache Spark merupakan salah satu kerangka kerja pemrosesan data terdistribusi yang mendukung komputasi paralel skala besar dan integrasi dengan algoritma pembelajaran mesin melalui Spark MLlib.

Penelitian ini bertujuan untuk mengeksplorasi penerapan Apache Spark dalam klasifikasi perintah suara berbasis pembelajaran mesin terdistribusi. Fokus penelitian meliputi ekstraksi fitur MFCC secara terdistribusi, pelatihan model klasifikasi menggunakan Logistic Regression dan Random Forest, serta evaluasi kinerja model pada dataset skala besar. Hasil penelitian ini diharapkan dapat memberikan solusi yang efisien dan skalabel untuk pengembangan sistem pengenalan suara pada berbagai aplikasi praktis.

Meskipun berbagai penelitian telah dilakukan dalam bidang pengenalan suara, pemrosesan dataset audio skala besar masih menghadapi tantangan terutama pada kebutuhan komputasi yang tinggi pada tahap ekstraksi fitur dan pelatihan model. Pendekatan pemrosesan konvensional pada satu mesin sering kali mengalami keterbatasan memori dan waktu komputasi ketika menangani data dalam jumlah besar. Oleh karena itu diperlukan pendekatan komputasi terdistribusi yang mampu meningkatkan efisiensi pemrosesan data.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk menganalisis penerapan machine learning terdistribusi menggunakan Apache Spark dalam klasifikasi perintah suara pada dataset Google Speech Commands. Penelitian ini menggunakan ekstraksi fitur MFCC dan algoritma klasifikasi Logistic Regression serta Random Forest yang dijalankan pada Spark MLlib. Melalui pendekatan ini diharapkan sistem mampu memproses dataset audio skala besar secara lebih efisien serta memberikan gambaran mengenai performa model klasifikasi yang dihasilkan.

2. TINJAUAN PUSTAKA

2.1. Dataset dan Tantangan Skalabilitas Audio

merupakan komponen vital dalam interaksi manusia-mesin yang kini banyak diterapkan pada perangkat *mobile* dan *Internet of Things* (IoT). Untuk mendukung riset di bidang ini, ketersediaan dataset publik yang terstandarisasi sangat diperlukan. Warden memperkenalkan *Google* Speech *Commands Dataset*, sebuah kumpulan data perintah suara skala besar yang dirancang untuk melatih model *keyword spotting* dengan variasi aksen dan lingkungan yang luas.

Namun, seiring dengan pertumbuhan volume data audio, metode pemrosesan konvensional pada mesin tunggal menghadapi kendala efisiensi yang signifikan. Li dan Zhao menyoroti bahwa pengenalan perintah suara yang skalabel memerlukan kerangka kerja komputasi terdistribusi untuk menangani beban komputasi yang tinggi pada tahap pra-pemrosesan dan pelatihan model [4]. Hal ini didukung oleh Kumar dan Singh, yang menyatakan bahwa analitik suara berskala besar (*large-scale speech analytics*) harus memanfaatkan pembelajaran mesin terdistribusi untuk mengatasi keterbatasan memori dan mempercepat waktu eksekusi yang tidak dapat dicapai oleh arsitektur tradisional [5].

2.2. Komputasi Terdistribusi dengan Apache Spark

Sebagai solusi atas permasalahan *Big Data*, Apache Spark hadir sebagai mesin pemrosesan data terpadu yang menawarkan performa lebih tinggi dibandingkan pendahulunya, seperti Hadoop MapReduce. Zaharia et al. menjelaskan bahwa kemampuan komputasi *in-memory* pada Spark memungkinkan pemrosesan data berulang yang jauh lebih cepat, yang sangat krusial untuk algoritma iteratif dalam pembelajaran mesin.

Untuk implementasi algoritma klasifikasi, Spark menyediakan pustaka MLlib. Meng et al. memaparkan bahwa MLlib dirancang untuk skalabilitas tinggi, memungkinkan algoritma seperti Regresi Logistik dan *Random Forest* dijalankan secara paralel pada klaster komputer [6]. Pemanfaatan kerangka kerja ini memungkinkan ekstraksi fitur dan pelatihan model dilakukan pada puluhan ribu berkas audio secara bersamaan dengan efisiensi tinggi.

2.3. Ekstraksi Fitur dan Representasi Sinyal

Dalam sistem klasifikasi suara, pemilihan fitur sangat menentukan akurasi model. Li et al. menjelaskan bahwa *Mel-Frequency Cepstral Coefficients* (MFCC) merupakan metode ekstraksi fitur yang paling dominan digunakan karena ketahanannya terhadap *noise* dan kemampuannya merepresentasikan karakteristik spektral suara manusia [7].

Meskipun MFCC efektif, tantangan muncul pada kata-kata pendek dengan kemiripan fonetik. Sainath dan Parada mencatat bahwa fitur pada *keyword spotting* sering kali mengalami tumpang tindih (*overlap*) jika tidak ditangani dengan representasi fitur yang mendalam, yang menjadi tantangan tersendiri bagi model klasifikasi standar [8].

2.4. Perkembangan Algoritma: Konvensional vs Deep Learning

Terdapat pergeseran tren dalam algoritma pengenalan suara. Sementara penelitian ini berfokus pada efisiensi infrastruktur menggunakan algoritma standar pada Spark, penelitian mutakhir cenderung mengarah pada *Deep Learning*. Sainath et al. menunjukkan bahwa *Deep Neural Networks* (DNN) dan pemrosesan sinyal multi-kanal memberikan peningkatan signifikan dalam pengenalan suara otomatis.

Lebih lanjut, Wang et al. dan Zhang et al. menemukan bahwa arsitektur seperti *Recurrent Neural Networks* (RNN) dan *Convolutional Neural Networks* (CNN) mampu menangkap ketergantungan temporal dan lebih tangguh (*robust*) terhadap kondisi lingkungan yang beragam dibandingkan model statistik tradisional [9], [10]. Selain itu, He et al. dan Abadi et al. menemukan bahwa arsitektur seperti Recurrent Neural Networks (RNN) dan Convolutional Neural Networks (CNN) mampu menangkap ketergantungan temporal dan lebih tangguh terhadap kondisi lingkungan yang beragam dibandingkan model statistik tradisional. Selain itu, He et al. serta Goodfellow et al. menunjukkan bahwa model deep learning modern, termasuk deep residual networks, mampu mempelajari representasi yang sangat kompleks, namun membutuhkan sumber daya komputasi yang besar [11]. Implementasi sistem berskala besar seperti TensorFlow memungkinkan pelatihan model-model tersebut secara efisien [12] untuk melatih model yang sangat dalam (*deep residual learning*) [13]. Oleh karena itu, penelitian ini mengisi celah dengan mengevaluasi sejauh mana algoritma efisien pada Spark dapat menangani dataset besar sebelum beralih ke kompleksitas *Deep Learning*.

3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimental, yang bertujuan untuk menganalisis kinerja klasifikasi perintah suara berbasis machine learning dalam lingkungan komputasi terdistribusi. Pendekatan ini dipilih karena memungkinkan evaluasi yang objektif terhadap performa model melalui pengukuran metrik numerik, seperti akurasi, presisi, recall, dan F1-score. Eksperimen dilakukan dengan memanfaatkan Apache Spark sebagai kerangka kerja pemrosesan data terdistribusi untuk menangani data suara dalam skala besar secara efisien.

Beberapa penelitian sebelumnya telah mengkaji klasifikasi perintah suara menggunakan berbagai pendekatan machine learning [1] memperkenalkan Google Speech Commands Dataset sebagai dataset standar untuk penelitian pengenalan perintah suara skala kecil. Penelitian menurut [5] menunjukkan

bahwa pemanfaatan komputasi terdistribusi dapat mempercepat proses pelatihan model machine learning pada data suara berskala besar. Selain itu, menurut [4] mengembangkan sistem pengenalan perintah suara menggunakan kerangka kerja komputasi terdistribusi yang mampu meningkatkan efisiensi pemrosesan data audio. Berbeda dengan penelitian tersebut, penelitian ini berfokus pada analisis penerapan Apache Spark sebagai platform machine learning terdistribusi untuk klasifikasi perintah suara menggunakan fitur MFCC dan algoritma klasifikasi konvensional.

3.1. Dataset

Dataset yang digunakan dalam penelitian ini ialah *Google Speech Commands Dataset*, yang terdiri dari lebih dari 100.000 berkas audio berdurasi satu detik dengan frekuensi sampling 16 kHz, Dataset ini mencakup berbagai perintah suara dasar seperti "yes", "no", "up", "down", "left", "right", "on", "off", "stop", dan "go", yang diucapkan oleh sejumlah besar pembicara berbeda. Keberagaman data ini memungkinkan pengembangan model klasifikasi yang lebih generalisasi terhadap variasi suara dan aksen. Selain itu, dataset ini juga dilengkapi dengan background noise untuk mendukung eksperimen yang lebih realistis pada lingkungan berisik.

3.2. Ekstraksi Fitur

Untuk merepresentasikan sinyal audio menjadi format numerik yang dapat diproses oleh algoritma machine learning, dilakukan ekstraksi fitur menggunakan *Mel-Frequency Cepstral Coefficients* (MFCC). MFCC merupakan fitur yang paling banyak digunakan dalam pengenalan ucapan modern karena mampu merepresentasikan karakteristik spektral sinyal suara secara efektif [14]. Dalam penelitian ini, setiap berkas audio diekstraksi menjadi 13 koefisien MFCC rata-rata per frame. Proses ekstraksi dilakukan secara terdistribusi menggunakan Apache Spark, sehingga dapat menangani dataset besar secara efisien.

3.3. Machine Learning Terdistribusi

Setelah fitur diekstraksi, dilakukan normalisasi fitur menggunakan metode StandardScaler untuk memastikan semua fitur memiliki skala yang sama. Selanjutnya, dataset dibagi menjadi data latih dan data uji dengan proporsi 80:20. Model klasifikasi dilatih menggunakan algoritma *Logistic Regression* dan *Random Forest* melalui Spark MLlib, memanfaatkan komputasi terdistribusi untuk mempercepat proses pelatihan. Pendekatan ini memungkinkan pengolahan dataset yang besar tanpa membebani memori lokal dan mempersingkat waktu komputasi secara signifikan.

3.4. Evaluasi Model

Evaluasi performa model dilakukan dengan menggunakan metrik akurasi pada data uji, sehingga dapat menilai kemampuan model dalam mengklasifikasikan perintah suara yang [15] belum pernah dilihat sebelumnya . Selain itu, dilakukan visualisasi distribusi label dan fitur MFCC untuk memahami karakteristik dataset, serta analisis PCA 2D untuk melihat distribusi fitur dan sebaran kelas secara lebih intuitif. Pendekatan evaluasi ini memberikan gambaran menyeluruh mengenai kinerja model dan kesesuaian fitur dalam representasi data suara.

4. HASIL DAN PEMBAHASAN

Berdasarkan hasil eksperimen menggunakan dataset *Google Speech Commands* yang terdiri dari 64.721 file audio, Apache Spark berhasil digunakan untuk memproses data audio berskala besar tanpa mengalami kendala memori maupun kegagalan proses. Seluruh tahapan pemrosesan, mulai dari pemuatan data audio, ekstraksi fitur MFCC, hingga pelatihan model *machine learning*, dapat dijalankan secara paralel pada lingkungan *distributed computing*. Data yang digunakan dalam penelitian ini bersumber dari Google Speech Commands Dataset. Dataset ini merupakan kumpulan perintah suara yang tersedia secara publik dan dapat diakses melalui Kaggle. Dataset ini terdiri dari berbagai label kata perintah dengan durasi rata-rata 1 detik per file audio. Rincian ringkasan dataset yang digunakan dalam eksperimen ini, termasuk jumlah sampel per kelas dan spesifikasi audio, disajikan pada Tabel 1. Sumber Dataset: *https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/data*

Tabel 1. Ringkasan Dataset *Google Speech Commands*

No	Label (Perintah)	Jumlah Sampel	Durasi Rata-rata	Sampling Rate
1	Stop	2.380	1 detik	16 kHz
2	Seven	2.377	1 detik	16 kHz
3	Yes	2.377	1 detik	16 kHz
4	Zero	2.376	1 detik	16 kHz
5	Up	2.375	1 detik	16 kHz
6	No	2.375	1 detik	16 kHz
7	Two	2.373	1 detik	16 kHz
8	Four	2.372	1 detik	16 kHz
9	Go	2.372	1 detik	16 kHz
10	One	2.370	1 detik	16 kHz
11	Six	2.369	1 detik	16 kHz
12	Right	2.367	1 detik	16 kHz
13	On	2.367	1 detik	16 kHz
14	Nine	2.364	1 detik	16 kHz
15	Down	2.359	1 detik	16 kHz
16	Off	2.357	1 detik	16 kHz
17	Five	2.357	1 detik	16 kHz
18	Three	2.356	1 detik	16 kHz
19	Left	2.353	1 detik	16 kHz
20	Eight	2.352	1 detik	16 kHz
Total	**20 Kelas**	**47.348**	-	-

Berdasarkan Tabel 1, dataset memiliki distribusi kelas yang relatif seimbang (*balanced*) dengan rata-rata sampel sekitar 2.300 file audio per label. Seluruh

data memiliki format *.wav* dengan durasi 1 detik dan sampling rate 16 kHz.

Tabel 2. Kolom label dan row yang melalui dataset *Google Speech Commands*

Label	Count
stop	2380
yes	2377
seven	2377
zero	2376
no	2375
up	2375
two	2373
four	2372
go	2372
one	2370
six	2369
on	2367
right	2367
nine	2364
down	2359
five	2357
off	2357
three	2356
left	2353
eight	2352

Distribusi jumlah data pada setiap kelas relatif seimbang, dengan jumlah sampel per label berkisar antara ±1.700 hingga ±2.400 file audio, sebagaimana ditunjukkan pada Tabel 2 Kondisi ini menunjukkan bahwa dataset tidak mengalami ketimpangan kelas (*class imbalance*) yang signifikan, sehingga performa model tidak dipengaruhi oleh dominasi kelas tertentu. Perhatikan tabel 3 dibawah ini.

Tabel 3. Sampel Hasil Ekstraksi Fitur MFCC (13 Koefisien) pada Label 'tree'

No	Label	Label Index	MFCC (Vector)
1	tree	27.0	[-538.50, 127.19, -17.23, 15.22, 1.45, -2.30, -20.41, 23.04, 5.79, 3.57, -11.59, -0.51, -2.34]
2	tree	27.0	[-166.95, -22.87, -5.11, 16.16, -21.49, -33.88, -18.04, -10.69, -34.81, -17.28, -28.63, -22.58, -12.21]
3	tree	27.0	[-455.25, 51.53, -41.23, 50.55, -8.50, -15.91, -21.09, -9.11, -12.36, -5.18, -2.96, 5.30, -14.58]
4	tree	27.0	[-305.62, 155.74, 22.48, 18.84, 5.27, -14.58, -26.35, -3.63, -8.71, -10.61, 1.26, -11.41, 0.58]
5	tree	27.0	[-409.47, 83.87, -7.02, 40.55, 5.54, 4.55, 4.40, 8.25, -3.26, 6.63, -5.18, 7.87, -3.39]

Tabel 3 menampilkan sampel ekstraksi fitur MFCC untuk label 'tree'. Vektor numerik inilah yang digunakan sebagai input untuk membandingkan performa model klasifikasi pada tahap selanjutnya.
Dua model klasifikasi diterapkan dalam penelitian ini, yaitu *Logistic Regression* dan *Random Forest*. Ringkasan hasil evaluasi performa kedua model beserta total data yang diproses disajikan pada Tabel 4 berikut.

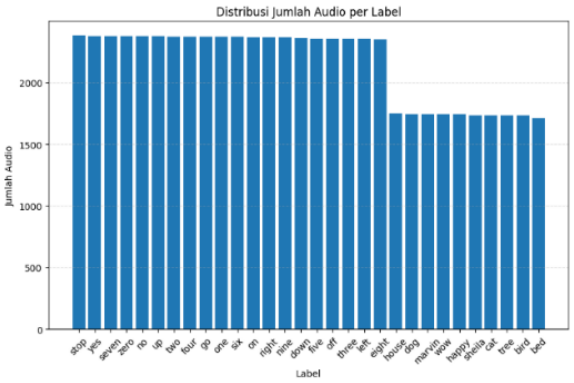
Tabel 4. Hasil Eksperimen Klasifikasi Audio

Keterangan	Metode/Model	Nilai
Jumlah Data	Total Audio Files	64.721
	Total Feature Rows	64.721
Evaluasi Model	Logistic Regression Accuracy	0.2572 (25,72%)
	Random Forest Accuracy	0.1807 (18,07%)

Berdasarkan Tabel 4, hasil evaluasi menunjukkan bahwa:

a. *Logistic Regression* memperoleh nilai akurasi sebesar 25,72%.
b. *Random Forest* memperoleh nilai akurasi sebesar 18,07%.

Perbandingan visual akurasi kedua model dapat dilihat pada Gambar 1 berikut:



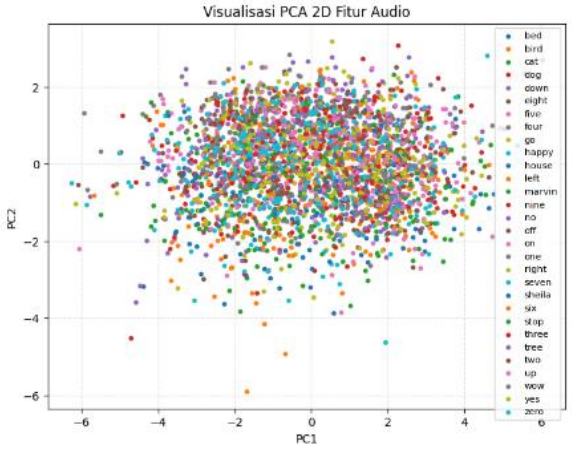
Gambar 1. Grafik perbandingan akurasi *logistic regression* dan *random forest*

Hasil ini menunjukkan bahwa Logistic Regression memberikan performa yang lebih baik dibandingkan Random Forest, meskipun secara keseluruhan tingkat akurasi kedua model masih tergolong rendah. Rendahnya akurasi mengindikasikan bahwa representasi fitur MFCC berbasis nilai rata-rata (mean MFCC) belum mampu menangkap karakteristik temporal sinyal suara secara optimal.

Model Random Forest yang umumnya unggul pada data *non-linear* justru menunjukkan performa lebih rendah dalam eksperimen ini. Hal ini diduga disebabkan oleh dimensi fitur yang terbatas serta hilangnya informasi urutan waktu (*time dependency*) akibat proses rata-rata pada sinyal audio, sehingga model tree-based kesulitan membangun decision boundary yang efektif.

4.1. Analisis Visualisasi Data

Untuk memahami karakteristik ruang fitur dan penyebab rendahnya akurasi, dilakukan visualisasi menggunakan *Principal Component Analysis* (PCA) guna mereduksi dimensi data menjadi 2 komponen utama. Sebagaimana terlihat pada Gambar 2, visualisasi PCA menunjukkan pola persebaran data antar kelas.

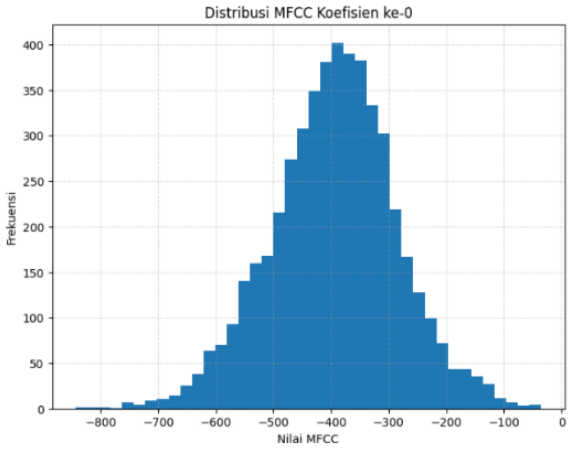


Gambar 2. Visualisasi PCA 2d fitur audio

Berdasarkan Gambar 2 di atas, terlihat bahwa sebagian besar titik data dari berbagai kelas (warna yang berbeda) saling bertumpang tindih (*heavy overlap*) dan berkumpul di area pusat, tanpa terbentuk klaster yang terpisah secara jelas. Hal ini mengkonfirmasi bahwa fitur MFCC rata-rata memiliki tingkat separabilitas yang rendah untuk membedakan perintah suara yang memiliki kemiripan fonetik, seperti "four", "for", "off", dan "on".

Temuan visual ini memperkuat hasil evaluasi model pada Tabel 4, di mana keterbatasan separabilitas fitur menyebabkan model klasifikasi kesulitan dalam mempelajari batas keputusan (decision boundary) yang tegas antar kelas, sehingga akurasi yang dihasilkan tidak maksimal.

4.2. Distribusi Fitur MFCC



Gambar 3. Visualisasi distribusi MFCC koefisien ke-0

Analisis lebih lanjut dilakukan terhadap distribusi nilai fitur untuk memastikan kualitas ekstraksi data. Histogram pada Gambar 3 memperlihatkan distribusi koefisien MFCC ke-0.

Grafik di atas menunjukkan pola distribusi yang mendekati normal (*bell curve*), yang menandakan bahwa proses ekstraksi fitur berjalan secara konsisten. Namun, rentang sebaran nilai yang cukup lebar mengindikasikan adanya variasi karakteristik spektral (seperti *loudness*) yang signifikan antar file audio dalam dataset.

4.3. Pembahasan Akhir

Hasil eksperimen menunjukkan bahwa penggunaan *Apache Spark* sangat efektif untuk menangani pemrosesan dataset audio berukuran besar secara terdistribusi (64.721 file). Namun, dari sisi performa klasifikasi, pendekatan machine learning konvensional dengan fitur MFCC statis terbukti memiliki keterbatasan dalam menangkap kompleksitas sinyal suara.

Berbeda dengan penelitian sebelumnya yang menunjukkan performa tinggi pada klasifikasi suara menggunakan model ensemble atau deep learning, penelitian ini menegaskan bahwa pemilihan representasi fitur memegang peranan vital. Tanpa mempertahankan informasi temporal, model berbasis fitur statistik sederhana cenderung menghasilkan akurasi yang rendah.

Oleh karena itu, peningkatan performa di masa mendatang dapat dilakukan dengan:

a. Menggunakan fitur MFCC berbasis sekuens (*frame-level MFCC*)
b. Menerapkan model *Deep Learning* seperti CNN atau LSTM.

Mengombinasikan fitur spektral dan temporal untuk memperkaya informasi input model.

5. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa penggunaan Apache Spark sebagai platform komputasi terdistribusi memberikan solusi yang efektif secara infrastruktur untuk klasifikasi perintah suara berbasis pembelajaran mesin. Dengan pendekatan ini, dataset audio skala besar seperti Google Speech Commands dapat diproses secara efisien, meliputi ekstraksi fitur MFCC secara terdistribusi, normalisasi fitur, serta pelatihan model.

Berdasarkan hasil eksperimen, dapat disimpulkan bahwa:

a. Efisiensi Komputasi: Pemrosesan terdistribusi mampu mengatasi keterbatasan memori dan waktu komputasi yang biasanya terjadi pada pengolahan dataset skala besar, memungkinkan pemrosesan 64.721 file audio sekaligus.
b. Performa Model: Model Logistic Regression menghasilkan akurasi 25,72%, lebih unggul dibandingkan Random Forest (18,07%). Namun, akurasi ini masih tergolong rendah akibat

penggunaan fitur MFCC rata-rata (mean) yang menghilangkan informasi temporal suara.

c. Visualisasi Data: Visualisasi PCA (Gambar 3.2) mengkonfirmasi adanya tumpang tindih (overlap) yang signifikan antar kelas, yang menjadi penyebab utama rendahnya akurasi model konvensional.

Penelitian ini menekankan pentingnya pendekatan terdistribusi untuk skalabilitas sistem, namun juga menyoroti perlunya model yang lebih kompleks untuk akurasi tinggi. Saran Untuk penelitian selanjutnya, disarankan untuk:

a. Mengembangkan model pembelajaran mendalam terdistribusi (seperti CNN atau RNN/LSTM) yang dapat menangkap pola temporal sinyal audio dengan lebih baik.
b. Melakukan eksplorasi teknik augmentasi data untuk meningkatkan variasi latih.
c. Mengintegrasikan pipeline dengan platform cloud untuk meningkatkan skalabilitas sistem pada skenario real-time.

DAFTAR PUSTAKA

[1] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv Prepr. arXiv1804.03209*, 2018.

[2] M. Zaharia *et al.*, "Apache Spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016, doi: 10.1145/2934664.

[3] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, and M. Bacchiani, "Multichannel signal processing with deep neural networks for automatic speech recognition," in *Proceedings of ICASSP*, 2018, pp. 4749–4753.

[4] Z. Li and S. Zhao, "Scalable speech command recognition using distributed computing frameworks," *IEEE Access*, vol. 9, pp. 45678–45689, 2021, doi: 10.1109/ACCESS.2021.3067890.

[5] A. Kumar and R. Singh, "Distributed machine learning for large-scale speech analytics," *J. Big Data*, vol. 7, no. 1, pp. 1–19, 2020, doi: 10.1186/s40537-020-00345-9.

[6] X. Meng *et al.*, "MLlib: Machine learning in Apache Spark," *J. Mach. Learn. Res.*, vol. 17, no. 34, pp. 1–7, 2016.

[7] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 1, pp. 1–19, 2016, doi: 10.1109/TASLP.2015.2482190.

[8] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proceedings of Interspeech*, 2015, pp. 1478–1482.

[9] Y. Wang, A. Narayanan, and A. Kannan, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2017, pp. 4965–4969.

[10] Z. Zhang, J. Geiger, J. Pohjalainen, A. E. D. Mousa, W. Jin, and B. Schuller, "Deep learning for environmentally robust speech recognition," *IEEE Signal Process. Mag.*, vol. 34, no. 3, pp. 84–94, 2018, doi: 10.1109/MSP.2017.2766210.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 77–87, 2016.

[13] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proceedings of OSDI*, 2016, pp. 265–283.

[14] A. K. Yadav and P. K. Bora, "Robust MFCC Features for Small Vocabulary Keyword Spotting," *IEEE Signal Process. Lett.*, vol. 29, pp. 188–192, 2022.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2017.