Faktor Exacta

Vol. 16, No. 4, Dec 2023, pp. 256~263
eISSN: 2502-339X, pISSN: 1979-276X, DOI: 10.30998/faktorexacta.v16i4.19216

Deteksi Pelat Nomor Dengan Menggunakan Optical Character Recognition Berbasis Algoritma LSTM

Elisa Wardhani[1], Saruni Dwiasnati[2]
[1,2]Teknik Informatika, Universitas Mercu Buana, Indonesia

Article Info

Article history:

Received Jul 12, 2023
Revised Oct 04, 2023
Accepted Dec 22, 2023

Keywords:

Deteksi Pelat Nomor
LSTM
Optical Character Recognition

ABSTRACT

Illegal parking is a recurring issue in the Jakarta area, prompting a system transformation that involves adding automatic parking facilities based on building infrastructure and facilities regulations. The development of an automatic parking system can utilize license plate number detection to minimize manual entry of license plate number in the parking system. In this study, LSTM algorithm is trained and implemented on Optical Character Recognition by Tesseract Engine with the aim to create and develop an accurate license plate number detection system to streamline the workflow of institutions requiring vehicle license plate information. The LSTM algorithm demonstrated good performance in detecting license plate numbers with an accuracy rate of 86.36%. However, the LSTM algorithm performance improved significantly to 95.8% accuracy when implemented on Optical Character Recognition by Tesseract Engine. Therefore, based on the evaluation, the LSTM algorithm integrated with Optical Character Recognition is considered the preferred choice for license plate number detection due to its higher accuracy compared to using the LSTM algorithm alone.

Corresponding Author:

Elisa Wardhani,
Faculty of Computer Science,
Universitas Mercu Buana,
Jl. Raya, RT.4/RW.1, Meruya Sel., Kec. Kembangan, Jakarta, Daerah Khusus Ibukota Jakarta 11650
Email: elisaawardhani@gmail.com

1. PENDAHULUAN

Maraknya parkir liar yang terjadi di sebagian wilayah Jakarta mengakibatkan gangguan pada arus lalu lintas. Hal ini mendorong perlunya dilakukan pembenahan sistem dan penambahan fasilitas parkir [1]. Untuk menanggulangi permasalahan parkir liar ini, pemerintah membuat Peraturan Menteri Pekerjaan Umum dan Perumahan Rakyat Republik Indonesia Nomor 14/PRT/M/2017 Tentang Persyaratan Kemudahan Bangunan Gedung dimana Pasal 36 menyatakan bahwa "Kelengkapan prasarana dan sarana pemanfaatan bangunan gedung sebagaimana dimaksud dalam Pasal 8 meliputi: o. tempat parkir; p. sistem parkir otomatis." [2]. Berdasarkan peraturan tersebut, gedung perkantoran wajib menyediakan lahan parkir dengan sistem parkir otomatis. Sistem parkir yang dimaksud berlaku untuk pengguna kendaraan yang parkir di lokasi khusus, seperti di apartemen atau instansi. Salah satu aspek penting dalam sistem parkir otomatis adalah identifikasi objek dimana informasi yang diperoleh secara otomatis dimasukkan ke dalam sistem komputer tanpa perlu campur tangan manusia [3]. Untuk mewujudkan sistem parkir otomatis tersebut, dapat memanfaatkan *Optical Character Recognition* dalam pendeteksian pelat nomor otomatis pada sistem parkir dimana sistem dapat mengidentifikasi pelat nomor kendaraan melalui potret CCTV di lahan parkir.

Penggunaan *Optical Character Recognition* sendiri terbilang cukup banyak. Penelitian sebelumnya dengan judul "Implementasi Algoritma YOLO dan Tesseract OCR Pada Sistem Deteksi Pelat Nomor Otomatis", menjelaskan tentang pentingnya sistem deteksi pelat nomor otomatis dikarenakan metode manual

memakan banyak waktu dan tenaga manusia serta diperlukan karena jumlah kendaraan yang terus bertambah akan semakin membebankan tenaga manusia. Tujuan dari penelitian tersebut yaitu diharapkan dengan adanya sistem pendeteksi pelat nomor kendaraan, dapat membantu dan memudahkan proses pekerjaan lembaga atau instansi yang membutuhkan data informasi tentang kode pelat nomor kendaraan. Identifikasi pelat nomor otomatis dilakukan dengan menggunakan algoritma LSTM, dimana pada bidang pembelajaran mesin (*machine learning*), model LSTM banyak digunakan dalam masalah analisis sekuens. Pengenalan teks dan proses transliterasi merupakan salah satu masalah umum yang secara natural dapat mengimplementasikan model LSTM tersebut [4].

Keberhasilan penggunaan *Optical Character Recognition* dapat juga ditemukan pada penelitian yang berjudul "Ektraksi Informasi/Data e-KTP Menggunakan *Optical Character Recognition Convolutional Neural Network*" yang menyimpulkan bahwa penerapan *Optical Character Recognition* dan empat lapisan *Convolutional Neural Network* efektif dalam mengekstraksi informasi dari 35 sampel e-KTP, dengan kecepatan rata-rata 30 detik dan tingkat kesalahan sebesar 5% [5].

Pemilihan penggunaan algoritma LSTM dilakukan karena algoritma LSTM merupakan salah satu jenis RNN yang menyimpan informasi mengenai pola-pola pada data untuk dapat mempelajari data yang perlu untuk disimpan serta data yang tidak diperlukan sehingga dapat dibuang, hal ini dikarenakan pada setiap *neuron* LSTM memiliki lebih dari 1 *gates* untuk dapat mengatur memori pada setiap *neuron* [6].

Penelitian ini bertujuan untuk menciptakan dan mengimplementasikan sistem deteksi pelat nomor kendaraan yang dapat secara signifikan membantu dan mempermudah proses pekerjaan lembaga atau instansi yang memerlukan data informasi tentang kode pelat nomor kendaraan. Pemilihan algoritma LSTM sebagai pendekatan utama didasarkan pada kemampuannya sebagai *jenis Recurrent Neural Network* (RNN) yang mampu menyimpan informasi terhadap pola-pola pada data. LSTM memiliki kemampuan untuk mempelajari dan mengenali data yang signifikan, sehingga dapat meningkatkan akurasi dalam mendeteksi pelat nomor kendaraan. Dengan demikian, diharapkan bahwa implementasi sistem ini tidak hanya memudahkan, tetapi juga dapat meningkatkan efisiensi proses kerja lembaga atau instansi terkait, terutama dalam konteks pengembangan sistem parkir otomatis.

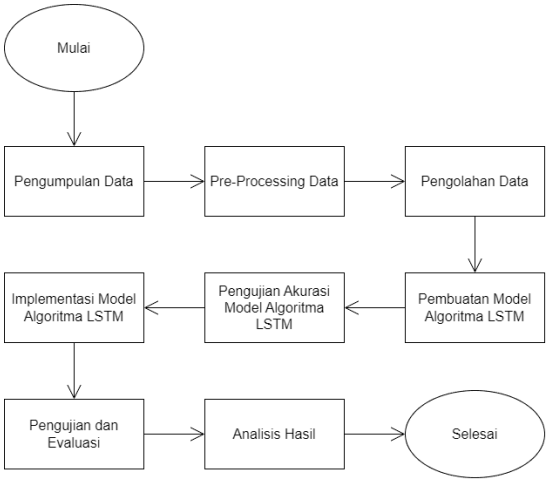
2. METODE

Dalam penelitian ini, algoritma LSTM akan dilatih dengan menggunakan bahasa pemrograman Python dengan bantuan *Optical Character Recognition* oleh Tesseract *Engine*. *Optical Character Recognition* merujuk pada bagian ilmu komputer yang melibatkan proses membaca teks dari gambar dan mengonversinya ke dalam format seperti kode ASCII, sehingga dapat diolah oleh komputer [7]. Dengan demikian, penggunaan *Optical Character Recognition* diterapkan untuk memungkinkan sistem mengidentifikasi tulisan tangan atau karakter tertulis dalam komunikasi pengguna tanpa memerlukan intervensi manusia [8]. Sedangkan Tesseract *Engine* adalah sebuah mesin *Optical Character Recognition* yang saat ini sedang dalam proses pengembangan oleh Google [9].

Algoritma LSTM akan dilatih menggunakan data yang didapat dari sumber *outsource* Kaggle yaitu "*Car License Plate Detection*" [10] dan "*Indian Vehicle License Plate*" [11] dengan jumlah data gabungan sebanyak 1.032 potret. Data potret termasuk potret tampak depan dan tampak belakang kendaraan mobil yang dirilis dan diperbaharui dalam rentang tahun 2020 – 2022, sampel dari dataset dapat dilihat pada Gambar 1. Data sebanyak 1.032 dataset akan dibagi menjadi dataset *training* dan dataset *test*. Selanjutnya, apabila hasil algoritma LSTM sudah dapat mendeteksi potret pelat nomor kendaraan dan mengubahnya menjadi bentuk teks secara baik, maka akan diimplementasikan sebagai hasil akhir pada pemrograman deteksi potret pelat nomor kendaraan.

Gambar 1. Sampel Dataset

 DOI: 10.30998/faktorexacta.v16i4.19216



Gambar 2. Alur Penelitian

Berdasarkan alur penelitian sesuai Gambar 2. Data yang terkumpul akan melalui tahap *pre-processing*. *Pre-processing* merupakan langkah krusial dalam tahapan awal proses *Optical Character Recognition*. Kualitas dari tahap ini sangat menentukan keberhasilan pengenalan pola secara keseluruhan [12]. Dalam tahapan ini, data potret melalui serangkaian proses untuk mempersiapkan data sebelum dapat diolah lebih lanjut mulai dari pengolahan warna, *Resize*, *Grayscale*, penghalusan *noise* dengan *Bilateral Filter*, deteksi sudut dengan *Canny*, hingga proses *Contouring* dan isolasi dengan *Masking*. Potret awal, seperti yang terlihat pada Gambar 3, mengandung terlalu banyak informasi dan perlu diproses agar dapat dibaca dengan baik.

Gambar 3. Potret Awal

Tahapan awal melibatkan penggunaan *Color Mode* untuk mengakses informasi warna asli dari potret, menghasilkan tiga saluran warna *pixel* (*Blue*, *Green*, dan *Red* - BGR). *Color Mode* memungkinkan manipulasi potret untuk pengolahan selanjutnya. Contoh hasil *Color Mode* dapat dilihat pada Gambar 4.

Gambar 4. Hasil *Color Mode*

Selanjutnya, untuk memastikan potret dapat dibaca dengan jelas, dilakukan *Resize* dengan mengubah ukurannya. Dalam kasus ini, potret diubah menjadi 1.75x ukuran asli secara horizontal dan 1.5x secara vertikal. Hasil dari *Resize* dapat dilihat pada Gambar 5.

Gambar 5. Hasil *Resize*

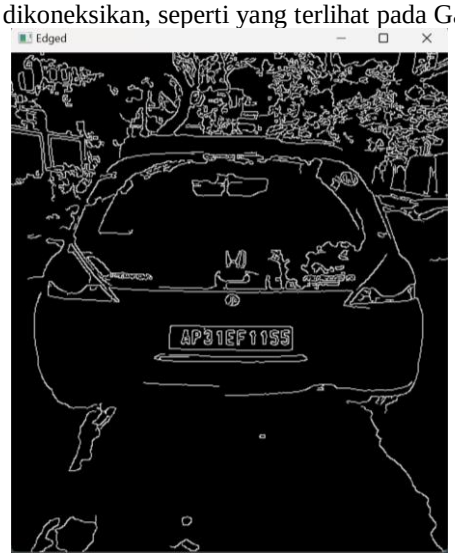
Setelah itu, potret yang sebelumnya direpresentasikan dalam tiga saluran warna BGR diubah menjadi *Grayscale*, menghasilkan potret dengan satu saluran warna (*Gray*). *Optical Character Recognition* bekerja optimal ketika diterapkan pada potret monokrom yang menggunakan data biner. Potret dengan banyak informasi warna, memerlukan pengolahan tambahan setidaknya *Grayscale*. Proses ini, biasanya melalui penyesuaian *threshold*, bertujuan untuk mengubah potret menjadi format monokrom untuk meningkatkan kinerja *Optical Character Recognition* [13]. *Grayscale* mempermudah algoritma dalam membaca potret karena perbedaan hanya terletak pada tingkat keterangan dan intensitas *pixel*. Tampilan hasil *Grayscale* dapat dilihat pada Gambar 6.

Gambar 6. Hasil *Grayscale*

Meskipun sudah berada dalam format *Grayscale*, potret masih dapat memiliki *noise* yang dapat mempengaruhi pembacaan informasi. Oleh karena itu, dilakukan aplikasi *Bilateral Filter* untuk memperhalus *noise*, menghasilkan potret yang lebih bersih seperti yang terlihat pada Gambar 7.

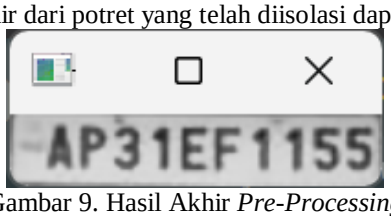
Gambar 7. Hasil *Bilateral Filter*

Selanjutnya, untuk meminimalkan kemungkinan kesalahan pembacaan, potret diperkecil ke bagian pelat nomor dengan menggunakan deteksi sudut menggunakan metode *Canny*. *Canny* bekerja dengan meredam *noise* lebih lanjut, menghitung arah setiap *pixel*, dan mengkategorikan *pixel* menjadi kuat, lemah, atau bukan sudut. Hasilnya, *pixel* kuat dan lemah dikoneksikan, seperti yang terlihat pada Gambar 8.



Gambar 8. Hasil *Canny*

Dengan terbentuknya sudut pada potret, dilakukan proses *Contouring* untuk mencari dan mengekstraksi *contour* pada sudut potret. Selanjutnya, dilakukan isolasi pada bagian pelat nomor dengan *Masking*. *Masking* melibatkan lapisan warna putih untuk mengisolasi bagian pelat nomor sehingga area potret sekitar pelat nomor dapat ditandai sebagai bagian yang tidak diperlukan dan bagian pelat nomor dapat diekstraksi dari potret utuhnya. Hasil akhir dari potret yang telah diisolasi dapat dilihat pada Gambar 9.



Gambar 9. Hasil Akhir *Pre-Processing*

Setelah melalui tahap *Pre-Processing*, data akan diolah sebagai langkah awal dalam pembuatan model. Pengolahan data melalui 2 tahapan yaitu pengaturan awal dan tahapan *load data*. Pada tahapan pengaturan awal, dilakukan *import* modul dan *library* yang esensial untuk memastikan kelancaran dan kesesuaian model yang akan dilatih. Modul OS digunakan untuk interaksi dengan sistem operasi, Cv2 dan Imutils untuk proses pengolahan gambar, Numpy untuk operasi matematika pada *array* multidimensional, Tensorflow untuk operasi pada model *machine learning*, dan PIL untuk manipulasi gambar dan penanganan *error* terkait *image processing*. Pengaturan parameter seperti *directory* dataset, ukuran gambar, jumlah *class*, *batch size*, dan jumlah *epochs* juga dilakukan untuk keperluan pelatihan model.

Selanjutnya, pada tahap *load data*, fungsi dibuat untuk memuat dataset. Dalam fungsi ini, potret dibagi berdasarkan *class*, yaitu class0 dan class1, dan *list* kosong *images* dan *labels* diinisialisasi. Fungsi *class index* juga diinisialisasi untuk menentukan *index class* pada potret. Dataset kemudian melalui proses *pre-processing*, seperti *resizing*, *grayscale*, *bilateral filtering*, *edge detection*, dan *contour extraction*. Jika potret valid, ukurannya bukan 0, maka dilakukan normalisasi, konversi ke *array* 3D, dan konversi ke float32 untuk memfasilitasi *training* model. Data yang telah dihasilkan dari proses ini ditambahkan ke dalam *list images* dan *labels*, sesuai dengan *class index*. *Error* terkait *pre-processing*, seperti gambar yang tidak dapat terbaca, ditangani melalui *except* untuk mencegah terhentinya proses *training* model secara tiba-tiba. Akhirnya, *list images* dan *labels* dikonversi menjadi *array* Numpy untuk disiapkan sebagai *input* model *machine learning*. Tahap pengolahan data ini membentuk dasar yang penting untuk melatih dan menguji model dalam penelitian ini.

Pada tahap pembuatan model algoritma LSTM, fungsi dibuat untuk menentukan arsitektur model dengan menggunakan Keras API. Keras API menyediakan Keras *Applications* yang terdiri dari model-model *deep learning* dengan *pre-trained weights*. Model-model ini dapat dipergunakan untuk melakukan prediksi, ekstraksi fitur, atau *fine-tuning* sesuai kebutuhan [14]. Model ini dirancang untuk memproses data potret yang telah melalui tahap *pre-processing*, dengan bentuk input yang ditetapkan berdasarkan dimensi potret yang telah di-*resize*. Model ini menggunakan algoritma LSTM dengan penambahan *layer* Dense untuk optimalisasi

proses. Selanjutnya, model secara keseluruhan di-*compile* dengan spesifikasi *loss function categorical cross-entropy*, algoritma optimization ADAM, dan metrik evaluasi akurasi. *Categorical cross-entropy* adalah *loss function* yang diterapkan pada tugas klasifikasi multikelas. Fungsi ini dirancang untuk mengukur disparitas antara dua distribusi probabilitas [15]. Sedangkan Algoritma ADAM merupakan suatu algoritma optimisasi dengan laju pembelajaran adaptif. Penggunaan optimasi ADAM bertujuan untuk meningkatkan akurasi model dengan menghasilkan peningkatan *learning rate* [16].

Proses selanjutnya adalah pembagian dataset menjadi *training* set dan *test* set dengan rasio 80:20. Pemisahan ini memungkinkan model untuk dievaluasi pada data yang belum pernah dilihat sebelumnya, memastikan kemampuan generalisasi dan mengidentifikasi potensi *overfitting*. *Overfitting* merupakan isu di mana model klasifikasi terlalu cocok dengan data pelatihan, yang dapat merugikan kinerja model pada data uji karena tidak semua informasi dalam data pelatihan relevan atau bermanfaat [17].

Tahapan melatih model algoritma LSTM melibatkan pemanggilan fungsi yang telah dibuat sebelumnya. Dalam proses pelatihan ini, *layer* LSTM digunakan untuk memproses *input sequence*, sementara *layer* Dense melakukan klasifikasi akhir dengan menggunakan fungsi aktivasi *softmax*. Model dilatih dengan menggunakan *training* set, dengan *batch size* 16 dan 10 *epochs*. *Test* set digunakan sebagai data validasi untuk mengkalkulasi *loss* dan metrik evaluasi setiap *epoch*, sehingga model dapat menyesuaikan bobotnya berdasarkan algoritma *optimization* ADAM.

Setelah melalui proses *training*, performa model dievaluasi menggunakan *test* set. Hasil evaluasi, termasuk akurasi, disimpan dalam variabel *accuracy*, yang kemudian dikonversi ke persentase dan diformat dengan dua desimal menggunakan notasi f-*string* untuk memberikan angka yang lebih spesifik dalam bentuk koma. Tahap evaluasi ini memungkinkan penilaian yang jelas terhadap kemampuan model dalam melakukan klasifikasi pada dataset yang belum pernah dilihat sebelumnya.

3. HASIL DAN PEMBAHASAN

Model algoritma LSTM yang sudah melalui tahapan *training* menghasilkan evaluasi yaitu tingkat akurasi sebesar 86,36%. Namun masih harus dilakukan pengujian terhadap model, dilakukan dengan mengimplementasikan model algoritma LSTM pada *Optical Character Recognition* milik Tesseract *Engine* dalam mendeteksi pelat nomor. Dalam pengujian ini, data yang digunakan hanya 1 buah potret setiap pengujian dilakukan.

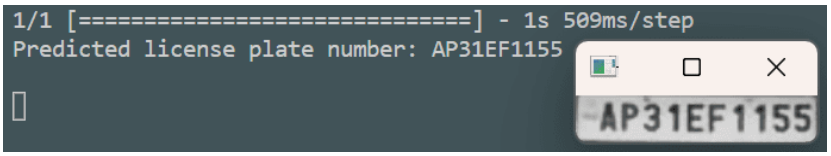
```
Epoch 10/10
 1/17 [>.............................] - ETA: 0s - loss: 0.5456 - accuracy:
 3/17 [====>.........................] - ETA: 0s - loss: 0.4280 - accuracy:
 5/17 [=======>......................] - ETA: 0s - loss: 0.4697 - accuracy:
 7/17 [===========>..................] - ETA: 0s - loss: 0.4655 - accuracy:
 9/17 [==============>...............] - ETA: 0s - loss: 0.4675 - accuracy:
11/17 [==================>...........] - ETA: 0s - loss: 0.5041 - accuracy:
13/17 [======================>.......] - ETA: 0s - loss: 0.5215 - accuracy:
15/17 [==========================>....] - ETA: 0s - loss: 0.5146 - accuracy:
17/17 [==============================] - ETA: 0s - loss: 0.5168 - accuracy:
17/17 [==============================] - 1s 39ms/step - loss: 0.5168 - accu
racy: 0.7672 - val_loss: 0.5156 - val_accuracy: 0.8636
1/3 [=========>....................] - ETA: 0s - loss: 0.4949 - accuracy: 0
3/3 [==============================] - 0s 12ms/step - loss: 0.5156 - accura
cy: 0.8636
Accuracy: 86.36%
```

Gambar 10. Hasil Evaluasi Akurasi Model Algoritma LSTM

3.1. Pengujian implementasi model algoritma LSTM pada Optical Character Recognition

Pengujian terhadap model algoritma LSTM dilakukan dengan mengintegrasikannya ke dalam program deteksi pelat nomor menggunakan Tesseract *Engine*. Setiap potret yang akan diuji harus melewati tahap *pre-processing* yang identik dengan tahap pada dataset pelat nomor yang digunakan dalam pelatihan model. Dalam pengujian, fungsi prediksi LSTM dibuat untuk melakukan klasifikasi menggunakan model LSTM. Potret yang telah melalui *pre-processing* diubah ukurannya sesuai dengan *input* model LSTM, dan hasil klasifikasi diubah menjadi format *integer* sebelum dikonversi menjadi format *string*. Informasi dalam bentuk *string* kemudian diubah menjadi format heksadesimal agar dapat diintegrasikan sebagai konfigurasi pada *Optical Character Recognition* (OCR) milik Tesseract *Engine*. Penambahan informasi LSTM pada konfigurasi OCR membantu Tesseract *Engine* memberikan hasil deteksi karakter yang lebih akurat dengan mempertimbangkan *output* dari model LSTM. Hasil prediksi dari potret ditampilkan dengan menjalankan OCR, dan beberapa konfigurasi tambahan pada OCR, seperti *Whitelist*, PSM (*Page Segmentation Modes*), OEM (OCR *Engine Mode*), dan konfigurasi tambahan untuk memanfaatkan model LSTM, digunakan untuk meningkatkan akurasi prediksi karakter pada potret tersebut. Proses pengujian ini bertujuan untuk memastikan

kemampuan model LSTM dalam meningkatkan kualitas deteksi karakter oleh Tesseract *Engine*, khususnya pada kasus-kasus dimana karakter sulit dideteksi berdasarkan tampilan visual saja.



Gambar 11. Hasil Pengujian Deteksi Pelat Nomor

3.2. Akurasi implementasi model algoritma LSTM pada Optical Character Recognition

Akurasi final dari model algoritma LSTM ini dibutuhkan untuk mengukur seberapa baik performa algoritma LSTM yang diimplementasikan pada *Optical Character Recognition* pada pendeteksian pelat nomor. Untuk menentukan akurasi tersebut, dilakukan perhitungan terhadap hasil prediksi dari 200 sampel yang diambil secara acak dari dataset yang digunakan. Perhitungan akurasi berdasarkan deteksi pelat nomor dari 200 sampel acak memberikan hasil yaitu tingkat akurasi sebesar 95,8% yang dapat dilihat pada Tabel 1.

$$Akurasi = \frac{Jumlah\ Karakter\ Benar}{Jumlah\ Seluruh\ Karakter} \times 100\%$$

Tabel 1. Sampel Perhitungan Akurasi Deteksi

Pelat Nomor	Hasil Deteksi	Akurasi
AS19D9919	AS19D9919	100%
CG12AG1130	CG12AG1130	100%
KA03MG2784	KA03MG2784	100%
DL8CX4850	DL8CX4850	100%
MH12HF5057	MH42HF5057	90%
DL12CN8660	DL12CN8660	100%
MH01AV8866	MHO1AV8866	90%
MH12DE1433	WH12DE1433	90%
TN19F0816	IN19F0816	88%
F65022	F65022	100%

4. SIMPULAN

Berdasarkan hasil penelitian, dapat disimpulkan bahwa implementasi sistem deteksi pelat nomor kendaraan dengan menggunakan algoritma LSTM dan Tesseract *Engine* menghasilkan tingkat akurasi yang cukup baik, yaitu sebesar 95,8%. Hal ini menunjukkan bahwa kombinasi metode yang digunakan mampu memberikan kontribusi positif terhadap tujuan awal penelitian, yaitu memudahkan dan meningkatkan efisiensi proses pekerjaan lembaga atau instansi yang membutuhkan informasi tentang kode pelat nomor kendaraan. Dampak signifikan dari penelitian ini terletak pada peningkatan performa deteksi pelat nomor ketika menggunakan Tesseract *Engine* sebagai *Optical Character Recognition* (OCR). Akurasi yang meningkat dari 86,36% menjadi 95,8% menunjukkan bahwa integrasi algoritma LSTM dengan teknologi OCR dapat memberikan hasil yang lebih dapat diandalkan.

Kontribusi penelitian ini terletak pada pemilihan dan implementasi algoritma LSTM sebagai metode utama dalam deteksi pelat nomor. Meskipun hanya satu metode yang digunakan, hasil yang diperoleh menunjukkan bahwa algoritma LSTM memiliki potensi yang cukup besar dalam konteks ini. Adapun limitasi penelitian melibatkan keterbatasan dalam variasi metode, seiring dengan hanya menggunakan algoritma LSTM sebagai pendekatan utama. Selain itu, jumlah data yang digunakan terbatas pada 1.032 potret, yang mungkin dapat ditingkatkan untuk meningkatkan keandalan dan generalisasi model.

Rekomendasi untuk penelitian mendatang mencakup eksplorasi lebih lanjut terhadap berbagai metode deteksi pelat nomor, penggunaan dataset yang lebih besar dan representatif, serta peningkatan integrasi dengan teknologi OCR. Selain itu, penelitian lebih lanjut dapat dilakukan untuk memperluas cakupan aplikasi, seperti pengenalan pelat nomor untuk warna yang berbeda maupun jenis kendaraan yang berbeda atau dalam kondisi pencahayaan potret yang lebih beragam. Dengan demikian, penelitian selanjutnya diharapkan dapat memberikan kontribusi lebih lanjut dalam mengoptimalkan sistem deteksi pelat nomor kendaraan untuk keperluan praktis dan aplikatif.

DAFTAR PUSTAKA

[1] A. O. Anjani, "Parkir Liar Masih Marak di Jakarta, Pembenahan Sistem Krusial," *Kompas.id*, Oct. 6, 2022. https://www.kompas.id/baca/metro/2022/10/06/parkir-liar-masih-marak-jakarta-perlu-benahi-sistem

[2] Pemerintah Indonesia, "Undang-Undang Republik Indonesia Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan" *Lembaran Negara Republik Indonesia Tahun 2009 Nomor 96, Tambahan Lembaran Republik Indonesia nomor 5025*, Sekretariat Negara, Jakarta, 2009.

[3] E. D. Widianto, H. M. Wijaya, and I. P. Windasari, "Sistem Parkir Berbasis RFID dan Pengenalan Citra Pelat Nomor Kendaraan," Jurnal Teknologi dan Sistem Komputer, vol. 5, no. 3, pp. 115-122, Jul. 2017. https://doi.org/10.14710/jtsiskom.5.3.2017.115-122

[4] I. H. Amin and A. Aprilino, "IMPLEMENTASI ALGORITMA YOLO DAN TESSERACT OCR PADA SISTEM DETEKSI PLAT NOMOR OTOMATIS," *Jurnal Teknoinfo*, vol. 16, no. 1, pp. 54–59, Jan. 2022. https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/article/view/1522

[5] J. Valentino and Y. A. Susetyo, "Analisis Perbandingan Optical Character Recognition Google Vision dengan Microsoft Computer Vision pada Pembacaan KTP-el", *jtik*, vol. 7, no. 4, pp. 552–561, Oct. 2023. https://doi.org/10.35870/jtik.v7i4.1046

[6] E. S. Putri and M. Sadikin, "Prediksi Penjualan Produk Untuk Mengestimasi Kebutuhan Bahan Baku Menggunakan Perbandingan Algoritma LSTM dan ARIMA," *Format : Jurnal Ilmiah Teknik Informatika*, vol. 10, no. 2, pp. 162–171, Aug. 2021. https://publikasi.mercubuana.ac.id/index.php/format/article/view/10856

[7] R. Siregar, "Implementasi OTSU Thresholding pada Optical Character Recognition Menggunakan Engine Tesseract," *Jurnal Ilmiah Core IT: Community Research Information Technology*, vol. 7, no. 1, pp. 27-34, 2019. https://ijcoreit.org/index.php/coreit/article/view/97

[8] M. Rizal Toha and A. Triayudi, "Penerapan Membaca Tulisan di dalam Gambar Menggunakan Metode OCR Berbasis Website pada e-KTP," *Jurnal Sains dan Teknologi*, vol. 11, pp. 175-183, 2022. https://doi.org/10.23887/jst-undiksha.v11i1

[9] A. Rizqi, I. S. Aziz, and Widianto, "Rancang Bangun Aplikasi Penerjemah Bahasa Jepang – Indonesia Menggunakan OCR Berbasis Android", SinarFe7, vol. 2, no. 1, pp. 276–280, Aug. 2019. https://journal.fortei7.org/index.php/sinarFe7/article/view/447

[10] "Car License Plate Detection," *www.kaggle.com*. https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

[11] "Indian vehicle license plate dataset," *www.kaggle.com*. https://www.kaggle.com/datasets/saisirishan/indian-vehicle-dataset?resource=download

[12] S. Hartanto, A. Sugiharto, and S. N. Endah, "OPTICAL CHARACTER RECOGNITION MENGGUNAKAN ALGORITMA TEMPLATE MATCHING CORRELATION," *JURNAL MASYARAKAT INFORMATIKA*, vol. 5, no. 9, pp. 1-12, Apr. 2015. https://doi.org/10.14710/jmasif.5.9.8435

[13] V. Utomo, A. P. R. Pinem, and B. V. Christoko, "Pengenalan Karakter Optis untuk Pencatatan Meter Air dengan Long Short Term Memory Recurrent Neural Network ", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 5, no. 1, pp. 132 - 138, Feb. 2021. https://doi.org/10.29207/resti.v5i1.2807

[14] M. A. Pangestu and H. Bunyamin, "Analisis Performa Dan Pengembangan Sistem Deteksi Ras Anjing Pada Gambar Dengan Menggunakan Pre-Trained CNN Model," *JuTISI*, vol. 4, no. 2, pp. 341, Aug. 24, 2018. https://journal.maranatha.edu/index.php/jutisi/article/view/1501

[15] G. Gumelar, "Kombinasi Algoritma Sampling dengan Algoritma Klasifikasi untuk Meningkatkan Performa Klasifikasi Dataset Imbalance", *SISFOTEK*, vol. 5, no. 1, pp. 250 - 255, Sep. 2021. https://www.seminar.iaii.or.id/index.php/SISFOTEK/article/view/295

[16] T. I. Z. M. Putra, S. Suprapto, and A. F. Bukhori, "Model Klasifikasi Berbasis Multiclass Classification dengan Kombinasi Indobert Embedding dan Long Short-Term Memory untuk Tweet Berbahasa Indonesia", *JISTED*, vol. 1, no. 1, pp. 1–28, Nov. 2022. https://doi.org/10.35912/jisted.v1i1.1509

[17] L. A. Andika, H. Pratiwi, and S. S. Handajani, "KLASIFIKASI PENYAKIT PNEUMONIA MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK DENGAN OPTIMASI ADAPTIVE MOMENTUM", *IJSA*, vol. 3, no. 3, pp. 331–340, Oct. 2019. https://doi.org/10.29244/ijsa.v3i3.560