e- ISSN: 2828–299X

Identifikasi Tingkat Kesadaran Pengemudi dari Data Video dengan Menggunakan Convolutional Long Short Term Memory

Fajar Fatha Romadhan[1], Andrian Dwi Baitur Rizky[2], Muhammad Aulia Faqihuddin[3], Indah Agustien Siradjuddin[4]
[1,2,3,4]Teknik Informatika, Fakultas Teknik, Universitas Trunojoyo Madura
E-mail: *[1]*200411100047@student.trunojoyo.ac.id*, [2]*200411100027@student.trunojoyo.ac.id*, [3]*200411100210@student.trunojoyo.ac.id*, [4]*indah.siradjuddin@trunojoyo.ac.id*

Abstrak – *Mengemudi dalam keadaan tidak sadar atau mengantuk merupakan salah satu penyebab utama terjadinya kecelakaan lalu lintas. Meskipun demikian, penumpang biasanya tidak akan menyadari jika pengemudi kendaraan yang mereka kendarai dalam keadaan mengantuk. Untuk mendeteksi kantuk pengemudi pada umumnya adalah dengan menggunakan model pembelajaran mesin. Akan tetapi model pembelajaran mesin biasanya hanya dapat mengenali pola dari sebuah data tanpa memperhatikan deret waktu pada data sekuensial atau time series. Metode yang digunakan adalah Convolutional Long Short Term Memory. Convolutional Long Short Term Memory adalah salah satu bentuk Recurrent Neural Network yang dapat digunakan untuk mengatasi masalah korelasi deret waktu baik dalam waktu singkat maupun lama pada data spatial time series. Data yang digunakan adalah data yang berupa video dari seseorang yang sedang mengemudi. Data video merupakan data berurutan yang terdiri atas banyak frame yang berupa citra. Pada penelitian ini setiap frame tersebut dilakukan deteksi wajah untuk mendapatkan citra wajah pengemudi, sedangkan Convolutional Long Short Term Memory digunakan untuk mempelajari pola perubahan ekspresi wajah pengemudi seiring waktu. Hasil menunjukkan model memiliki akurasi sebesar 75% yang menunjukkan bahwa model memiliki kemampuan yang signifikan dalam memprediki kantuk pengemudi.*

Kata Kunci *— Convolutional, Data Video, Deteksi, Kantuk Pengemudi, Long Short Term Memory*

1. PENDAHULUAN

Mengemudi dalam keadaan mengantuk merupakan salah satu penyebab utama terjadinya kecelakaan lalu lintas. Berdasarkan data dari World Health Organization (WHO), rasa kantuk pengemudi menyebabkan sekitar 20% kecelakaan lalu lintas di dunia. Menurut laporan dari National Highway Traffic Safety Administration (NHTSA) di Amerika Serikat lebih dari 1,25 juta kematian setiap tahunnya, dan 20-50 juta orang terluka atau cacat disebabkan oleh mengemudi dalam keadaan mengantuk[1]. Di indonesia sendiri juga sangat banyak kecelakaan lalu lintas yang disebabkan oleh rasa kantuk pengemudi. Karena tingginya angka kecelakaan yang disebabkan oleh rasa kantuk pengemudi, maka perlu mengambil tindakan tepat waktu[2].

Penumpang biasanya tidak akan menyadari jika pengemudi dalam keadaan mengantuk. Mendeteksi rasa kantuk pengemudi dapat dilakukan menggunakan berbagai cara, termasuk dengan menggunakan teknologi komputer. Dengan memanfaatkan teknologi komputer, penumpang dapat mengetahui jika pengemudi dalam keadaan mengantuk sehingga dapat mengantisipasi terjadinya kecelakaan.

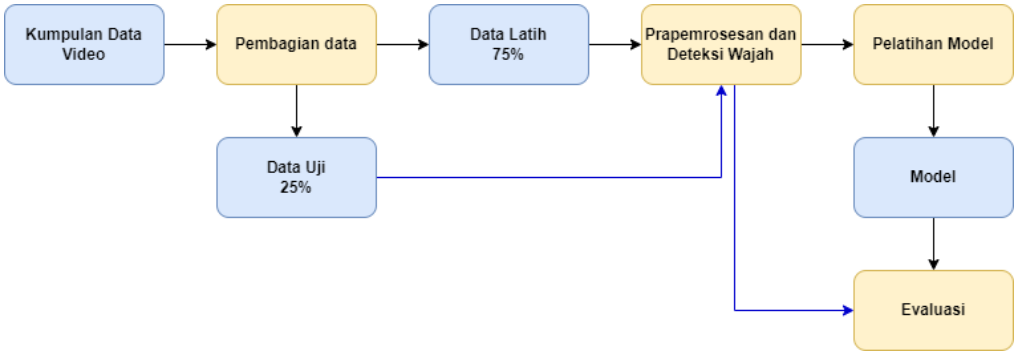
Metode yang umum digunakan untuk mendeteksi kantuk pengemudi adalah dengan menggunakan model pembelajaran mesin. Pembelajaran mesin adalah bagian dari bidang kecerdasan buatan (*Artificial Intelligence*) yang melibatkan pemrograman untuk mengaktifkan komputer agar dapat berperilaku cerdas seperti manusia, bahkan mereka dapat meningkatkan pemahaman mereka melalui pengalaman yang diperoleh secara otomatis selama pelatihan[3]. Akan tetapi model pembelajaran mesin biasanya hanya dapat mengenali pola dari sebuah data tanpa memperhatikan deret waktu pada data sekuensial atau time series. Pada penelitian [2] identifikasi kelelahan pengemudi dengan menggunakan pengenalan iris menghasilkan akurasi sekitar 80%. Pada penelitian [4] deteksi kelelahan dan kantuk menggunakan *deep convolutional network* menghasilkan rata-rata akurasi sebesar 68%, 81%, 91%, dan 93% pada skenario yang berbeda. Pada penelitian [5] deteksi kantuk pengemudi menggunakan analisis electrooculogram dan optimasi *grey wolf* menghasilkan akurasi 99,6%. Pada penelitian [6] deteksi kantuk pengemudi menggunakan optimasi *random forest* dan *single-channel* electroencephalogram menghasilkan akurasi sebesar 98,5%. Pada penelitian terkait sudah memiliki akurasi yang tinggi. Meskipun demikian, persamaan dari beberapa penelitian terkait tersebut adalah tidak memperhatikan pola perubahan pengemudi seiring dengan waktu. Padahal pola perubahan tersebut juga dapat digunakan untuk mengidentifikasi tingkat kesadaran atau kantuk pengemudi

e- ISSN: 2828–299X

Oleh karena itu, penelitian ini bertujuan untuk mengembangkan model yang dapat mendeteksi kantuk pengemudi dengan mempelajari pola perubahan ekspresi wajah pengemudi seiring dengan waktu dengan menggunakan metode pembelajaran mendalam. Metode pembelajaran mendalam digunakan karena popularitasnya dalam dunia ilmu komputer dan terbukti efektif dalam mengklasifikasikan dari kumpulan video. Metode pembelajaran mendalam menunjukkan kuat dan tepat untuk mengenali tindakan manusia[7]. Salah satu hal baru dari penelitian ini adalah penerapan langkah-langkah pra-pemrosesan pada video sebelum tahap pelatihan dan pengujian untuk meningkatkan performa model.

2. METODE PENELITIAN

Proses identifikasi tingkat kesadaran atau kantuk pengemudi ini dilakukan dengan melakukan klasifikasi biner dari kumpulan data video. Dari kumpulan data video tersebut akan diproses sedemikian rupa sehingga sesuai dengan arsitektur model yang dibangun. Kumpulan data video yang sudah diproses tersebut digunakan untuk melatih model hingga model tersebut dapat mengidentifikasi suatu video apakah orang dalam video tersebut dalam keadaan mengantuk atau tidak. Alur Proses identifikasi dapat dilihat pada diagram Gambar 1.



Gambar 1. Alur Proses Identifikasi

2.1 Data yang digunakan

Kumpulan data yang digunakan pada penelitian ini merupakan data yang berupa video dari seseorang yang sedang mengemudi. Video diambil dengan posisi kamera berada di *dashboard* kendaraan. Video diambil dengan pengemudi dalam dua kondisi yaitu pada saat mengantuk dan tidak mengantuk.

Kumpulan data didapatkan dari kaggle dengan nama SUST-DDD atau Sivas University of Science and Technology Driver Drowsiness Datasets. Kumpulan data ini terdiri dari dua kelas yaitu *drowsiness*, dan *not drowsiness*. Kumpulan data ini berupa video dengan jumlah video untuk kelas *drowsiness* 975 dan *not drowsiness* 1099. Kumpulan data ini dapat diakses pada https://www.kaggle.com/datasets/esrakavalci/sust-ddd. Tampilan data video dapat dilihat pada Gambar 2**.**

Gambar 2. Data Video

2.2 Praproses dan deteksi wajah

Pada praproses data, data yang berupa video akan diekstrak menjadi 20 *frame* per video. Setiap *frame* dilakukan deteksi wajah dengan menggunakan library *face_recognition* dari dlib python. Hal tersebut dilakukan

e- ISSN: 2828–299X

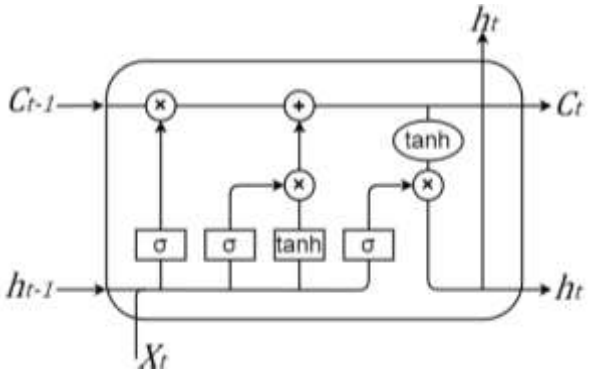
karena dalam pengenalan kantuk yang dibutuhkan hanyalah ekspresi wajah dari pengemudi. Selain itu hasil dari proses klasifikasi pixel memiliki dampak yang besar pada akurasi model jika dengan mengambil bagian wajahnya saja atau tidak[8]. Setelah deteksi wajah *frame* tersebut akan di ubah ukurannya menjadi 200×200 *pixel* agar tidak terlalu besar. Setelah ukurannya di ubah, selanjutnya *frame* akan dinormalisasi.

2.3 Metode yang Digunakan

Konvolusi merupakan sebuah proses yang digunakan untuk mengekstrak fitur dari data spatial seperti pada citra. Konvolusi bekerja dengan melibatkan kernel atau filter konvolusi. Kernel akan dikalikan menggunakan perkalian titik dan digeser secara berurutan melintasi seluruh citra input. Hasil perkalian titik tersebut dijumlahkan sehingga menghasilkan nilai tunggal dan menghasilkan *feature map*. Pemanfaatan operasi konvolusi dalam *neural network* dapat dilihat pada CNN. CNN atau Convolutional Neural Network merupakan arsitektur jaringan syaraf tiruan yang memanfaatkan layer konvolusi. CNN banyak digunakan sebagai layer yang digunakan untuk mengekstrak fitur dari data inputan seperti citra. Layer konvolusi pada CNN dinilai dapat mengurangi beban komputasi dari *neural network*[9]. CNN memiliki performa yang bagus meskipun tidak memerlukan tahap ekstraksi fitur dikarenakan CNN memiliki ekstraksi fitur tersendiri[10]. CNN melakukan ekstraksi fitur menggunakan lapisan konvolusi, fungsi aktivasi, dan lapisan *pooling*[10].

Recurrent Neural Network (RNN) merupakan suatu metode dalam *machine learning* yang digunakan untuk memproses data sekuensial atau *time series*. Pada data yang berbentuk sekuensial atau time series, nilai data t+1 akan dipengaruhi oleh data t-0 dan t-1. RNN memiliki kemampuan untuk mengingat input sebelumnya dengan menggunakan konsep memori internal atau state yang memungkinkan informasi sebelumnya dapat mempengaruhi pemrosesan informasi saat ini. RNN secara teori dapat melihat *long term dependency* jauh-jauh dari masa lampau, sehingga semua *input* dari masa lampau dapat memengaruhi *output*. Tapi pada implementasinya RNN memiliki masalah dengan *input* yang terlalu jauh di masa lampau. Untuk mengatasi hal tersebut arsitektur RNN dikembangkan sehingga tercipta arsitektur yang dikenal sebagai Long Short Term Memory (LSTM).

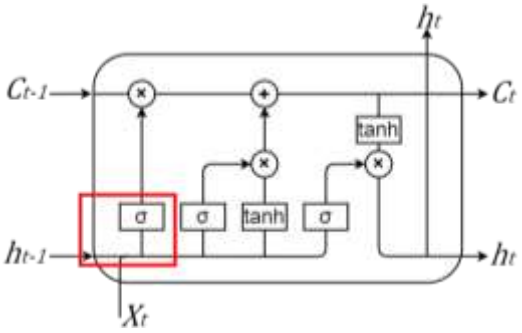
LSTM merupakan salah satu bentuk RNN yang dapat mengatasi masalah korelasi deret waktu baik dalam waktu singkat maupun lama dengan menggunakan lapisan tersembunyi sebagai sel memori. Ciri khas dari LSTM yaitu selain ada *hidden state* yang diberikan ke *time step* berikutnya terdapat juga *cell state*. *Cell state* ini seperti *state internal* pada LSTM. *Cell state* ini akan membawa informasi dari masa lampau. Informasi tersebut dapat dilupakan atau dibuang atau dapat ditambahkan pada *time step* berikutnya sehingga dapat diproses di *hidden state*. Dalam LSTM terdapat beberapa komponen penting, yaitu *Forget Gate, Input Gate, dan Output Gate*. LSTM memiliki arsitektur seperti pada Gambar 3:



Gambar 3. Diagram LSTM

Dari diagram pada Gambar 3 yang pertama kali dilakukan yaitu menghitung *forget gate*. *Forget gate* dalam LSTM berfungsi untuk menentukan seberapa banyak informasi dari *time step* sebelumnya yang akan dilupakan atau diabaikan dalam proses pengolahan informasi berikutnya. Nantinya *forget gate* ini akan dikalikan dengan *cell state time step* sebelumnya. Jika *forget gate* bernilai 0 maka *cell state time step* sebelumnya dilupakan atau dibuang. Jika *forget gate* bernilai 1 maka *cell state time step* sebelumnya dilanjutkan ke *time step* saat ini. Diagram *forget gate* dapat dilihat pada Gambar 4.

e- ISSN: 2828–299X



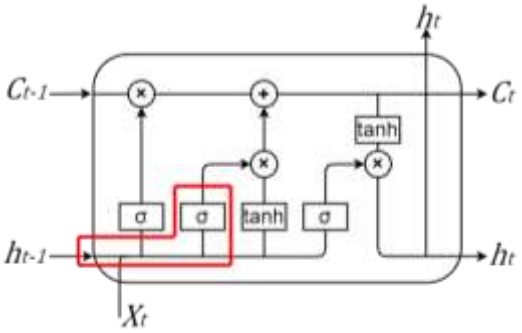
Gambar 4. Forget Gate

Forget gate dihitung dari gabungan *input* saat ini dan *hidden state time step* sebelumnya dengan *weight* dan *bias* yang kemudian dimasukkan ke dalam fungsi sigmoid. *Forget gate* dapat dihitung dengan Persamaan 1.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Dimana :
f_t : *forget gate*
σ : fungsi sigmoid
W_f : *weight* atau bobot pada *forget gate*
h_{t-1} : *hidden state time step* sebelumnya
x_t : input *time step* saat ini
b_f : *bias* pada *forget gate*

Setelah menghitung *forget gate*, berikutnya yaitu menghitung *input gate*. *Input gate* berfungsi untuk mengontrol seberapa banyak informasi baru dari *input* saat ini yang akan dimasukkan ke dalam memori jangka panjang sel LSTM. Diagram *input gate* dapat dilihat pada Gambar 5.



Gambar 5. Input Gate

Nilai dari *input gate* ini akan dikalikan per elemen dengan *input*nya lalu ditambahkan ke *cell state*. *Input gate* melambangkan seberapa banyak *input* yang akan ditambahkan ke *cell state* baru. *Input gate* dapat dihitung menggunakan Persamaan 2.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Dimana :
i_t : *input gate*
σ : fungsi sigmoid
W_i : *weight* atau bobot pada input gate
h_{t-1} : hidden state *time step* sebelumnya
x_t : input *time step* saat ini
b_i : *bias* pada *input gate*

Setelah mendapatkan *input gate*, berikutnya yaitu menghitung *cell state candidate*. *Cell state candidate* ini nantinya akan digunakan untuk menghitung *cell state* baru. Untuk menghitung *cell state candidate* dapat menggunakan Persamaan 3

e- ISSN: 2828–299X

$$\tilde{C}_t = tanh(W_c * [h_{t-1}, x_t] + b_c \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

Dimana :
$\tilde{C}_t$: *cell state candidate*
tanh : fungsi tangen hiperbolik
W_c : *weight* atau bobot pada *cell state*
h_{t-1} : *hidden state time step* sebelumnya
x_t : input *time step* saat ini
b_c : *bias* pada *cell state*

Hasil dari *cell state candidate* akan dikalikan dengan *input gate* pada perhitungan *Cell state* baru. Untuk menghitung *cell state* baru dapat menggunakan Persamaan 4.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

Dimana :
C_t : *Cell state* baru
f_t : *forget gate*
C_{t-1} : *cell state* time step sebelumnya
i_t : *input gate*
$\tilde{C}_t$: *cell state candidate*

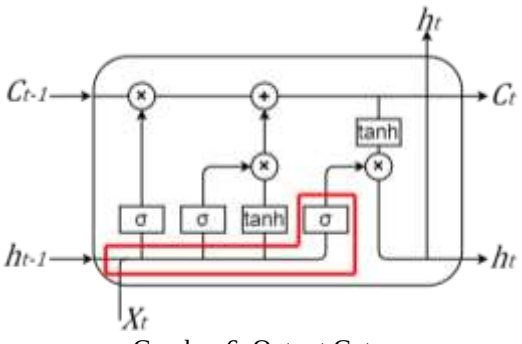
Pada perhitungan *cell state* baru, hasil dari *forget gate* akan dikalikan dengan *cell state time step* sebelumnya menggunakan perkalian antar elemen. Sehingga jika *forget gate* menghasilkan nilai 0 maka *cell state* pada *time step* sebelumnya akan dilupakan atau dibuang, sedangkan jika *forget gate* menghasilkan nilai 1 maka *cell state* dari *time step* sebelumnya akan dilanjutkan ke *cell state* pada *time step* saat ini. Setelah mendapatkan *cell state* baru, *cell state* tersebut akan diproses untuk menjadi *hidden state* pada time step saat ini.

Jumlah *output* yang akan menjadi *hidden state* akan diatur oleh *output gate*. Untuk menghitung *output gate* dapat menggunakan Persamaan 5.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

Dimana :
o_t : *output gate*
σ : fungsi sigmoid
W_o : *weight* atau bobot pada output gate
h_{t-1} : *hidden state time step* sebelumnya
x_t : *input time step* saat ini
b_o : *bias* pada *input gate*

Hasil dari *output gate* akan digunakan untuk menghitung *output*. *Output* ini akan digunakan sebagai *hidden state time step* saat ini. Diagram *output gate* dapat dilihat pada Gambar 6.



Gambar 6. Output Gate

Output dihitung dengan mengalikan *output gate* dan ditambahkan dengan hasil tangen hiperbolik dari *cell state* baru. *Output* dapat dihitung dengan Persamaan 6.

e- ISSN: 2828–299X

$$h_t = o_t * \tanh(C_t) \quad \text{.....(6)}$$

Dimana :
h_t : *output* atau *hidden state*
o_t : *output gate*
tanh : fungsi tangen hiperbolik
C_t : *cell state time step* saat ini

Beberapa *gates* yang ada pada LSTM memberikan kontrol terhadap informasi yang dapat disimpan atau diabaikan, sehingga membuatnya menjadi alat yang lebih mumpuni untuk memahami hubungan kompleks dalam data [11].

LSTM sebagai pengembangan dari RNN telah terbukti stabil dan kuat dalam memodelkan ketergantungan jarak jauh dalam berbagai studi sebelumnya[12]. Meski demikian, LSTM tidak dapat menerima input data spatial seperti citra. Arsitektur LSTM dirancang untuk memproses data yang berurutan seperti suara, teks, atau data *time series*. Berbeda dengan citra yang pada dasarnya merupakan data spatial dengan dimensi yang lebih besar daripada data yang berurutan.

Data video merupakan data berurutan yang terdiri atas banyak frame yang berupa citra. Untuk memproses data video, arsitektur LSTM dapat dikombinasikan dengan CNN. Arsitektur ini dinamakan Convolutional LSTM. Berbeda dengan LSTM, Convolutional LSTM dapat menerima input berupa citra dan menyimpan informasi sptatialnya sehingga memfasilitasi rekonstruksi data[13]. Dalam Convolutional LSTM, layer konvolusi digunakan untuk mengekstrak fitur spatial dari citra. Sedangkan LSTM digunakan untuk mengolah dan memahami urutan atau deret waktu pada video.

2.4 Arsitektur Model

Arsitektur model yang dibangun terdiri dari 5 layer, yaitu Convolutional LSTM 2D, maxpooling, timedistributed, Flatten, dan dense. Rincian layer yang digunakan dapat dilihat pada Tabel 1.

Tabel 1.

Layer	Parameter	Output Shape
Convolutonal LSTM 2D	filters = 16, kernel_size = (3, 3), activation = 'tanh', data_format = "channels_last", recurrent_dropout=0.2, return_sequences=True, input_shape = (20, 200, 200, 3)	(None, 20, 198, 198, 16)
Convolutional LSTM 2D	filters = 8, kernel_size = (3, 3), activation = 'tanh', recurrent_dropout=0.2, return_sequences=True,	(None, 20, 196, 196, 8)
Max Pooling 3D	pool_size=(1, 2, 2), padding='same', data_format='channels_last'	(None, 20, 98, 98, 8)
Time Distributed	Dropout(0.2)	(None, 20, 98, 98, 8)
Flatten	-	(None, 1536640)
Dense	Sesuai jumlah kelas (2), Activation=”softmax”	(None, 2)

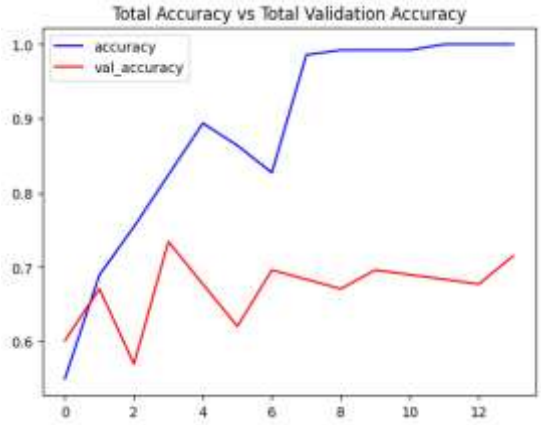
3. HASIL DAN PEMBAHASAN

Dari penelitian yang dilakukan telah mencapai hasil yang signifikan. Pada penelitian ini jumlah data yang digunakan hanya 900 video pada setiap kelas, sehingga total data yang digunakan adalah 1800 video. Pada proses deteksi wajah, dari 1800 video yang digunakan, hanya terdeteksi 1234 video dengan rincian 708 video kelas mengantuk dan 536 video tidak mengantuk. Jumlah data setiap kelas tersebut tidak sama atau tidak seimbang. Karena dikhawatirkan akan mempengaruhi hasil prediksi maka dari data tersebut dilakukan penyeimbangan data. Penyeimbangan data menggunakan undersampling sehingga kelas dengan jumlah data terbanyak akan dikurangi sehingga menjadi sama dengan jumlah data pada kelas dengan data paling sedikit. Pada kasus ini data dari kelas mengantuk akan dikurangi yang sebelumnya 708 video, menjadi 526 video.

e- ISSN: 2828–299X

Sehingga total data yang digunakan yaitu 1052 data video. Dari 1052 data video yang ada akan dibagi menjadi data untuk proses pelatihan dan data untuk proses evaluasi. Pembagian data untuk pelatihan sebnyak 75% dan data untuk evaluasi 25%. Sehingga jumlah data untuk pelatihan yaitu 789 video dan jumlah data untuk evaluasi 263 video.

Dari proses pelatihan dengan epoch sebanyak 50 dengan optimizer adam dan learning rate default sebesar 0,001 dan proses validasi, model yang telah dibangun menghasilkan akurasi yang signifikan. Grafik peningkatan akurasi pada proses pelatihan dapat dilihat pada Gambar 7.



Gambar 7. Grafik akurasi pelatihan

Dari Gambar 2 dapat diambil kesimpulan bahwa proses pelatihan berhenti pada epoch ke 14 dikarenakan tidak ada perubahan pada akurasi dan *loss*. Akurasi pelatihan mencapai 100% sedangkan akurasi validasinya sebesar 72%. Selain itu dilakukan juga proses evaluasi model menggunakan data evaluasi sebesar 25% dari total data yang digunakan. Sehingga data yang digunakan untuk proses evaluasi yaitu 263 video. Dari proses evaluasi yang dilakukan, menghasilkan confusion matriks seperti pada Tabel 2.

Tabel 2. Confusion Matriks

Class	True Positive	False Positive	False Negative	True Negative
Mengantuk	99	32	33	99
Tidak Mengantuk	99	33	32	99

Pada Tabel 2 disajikan tabel confusion matrix yang mencerminkan hasil evaluasi kinerja prediksi untuk huruf kelas mengantuk dan tidak mengantuk. Dari confusion matriks pada Tabel 2 dapat disimpulkan jika model yang dibuat memiliki hasil prediksi benar lebih banyak daripada prediksi salah. Dari Tabel 2 dapat dihitung nilai akurasi dengan Persamaan 7.

$$Akurasi = \frac{Banyak\ prediksi\ benar}{Total\ data\ yang\ diprediksi} \quad \text{.....(1)}$$

$$= \frac{TP + TN}{TP + FP + TN + FN}$$

Dimana :
TP : True Positive (Jumlah data yang diprediksi secara tepat sebagai kelas positif)
FP : False Positive (Jumlah data yang diprediksi secara tidak tepat sebagai kelas positif)
TN: True Negative (Jumlah data yang diprediksi secara tidak tepat sebagai kelas negatif)
FN: False Negative (Jumlah data yang diprediksi secara tidak tepat sebagai kelas negatif)

Dengan menggunakan Persamaan 7 nilai akurasi yang didapatkan yaitu 75%. Sehingga dapat disimpulkan bahwa model memiliki peluang 75% untuk memprediksi benar dan 25% untuk memprediksi salah.

4. SIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan dapat disimpulkan bahwa arsitektur Convolutional LSTM yang dibangun cukup baik digunakan untuk identifikasi tingkat kesadaran pengemudi dengan menggunakan data time series yang berupa video. Dari uji coba yang telah dilakukan, peneliti berhasil

mendapatkan nilai akurasi model sebesar 75%. Model yang dibangun dapat memprediksi data evaluasi dengan baik, namun terdapat kemungkinan 25% bagi model untuk salah memprediksi, sehingga model yang dibangun belum dapat dikatakan optimal.

5. SARAN

Model yang dibangun dalam penelitian ini belum dapat dikatakan optimal sehingga besar kemungkinan dapat dioptimalkan lagi. Pengoptimalan model dapat dilakukan dengan menambah jumlah layer pada arsitektur model, menambah jumlah epoch, mengganti nilai hyperparameter, maupun mengaplikasikan transfer learning yang telah ada.

DAFTAR PUSTAKA

[1] S. S. Jasim and A. K. A. Hassan, "Modern drowsiness detection techniques: a review," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 3, pp. 2986–2995, 2022, doi: 10.11591/ijece.v12i3.pp2986-2995.

[2] K. Gopalakrishna and S. A. Hariprasad, "Real-time fatigue analysis of driver through iris recognition," *Int. J. Electr. Comput. Eng.*, vol. 7, no. 6, pp. 3306–3312, 2017, doi: 10.11591/ijece.v7i6.pp3306-3312.

[3] F. Nur Fajri, A. Tholib, and W. Yuliana, "Application of Machine Learning Algorithm for Determining Elective Courses in Informatics Study Program," *J. Tek. Inform. dan Sist. Inf.*, vol. 8, no. 3, pp. 485–496, 2022, doi: 10.28932/jutisi.v8i3.3990.

[4] V. P. Sharma, J. S. Yadav, and V. Sharma, "Deep convolutional network based real time fatigue detection and drowsiness alertness system," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 5, pp. 5493–5500, 2022, doi: 10.11591/ijece.v12i5.pp5493-5500.

[5] S. S. Jasim and A. K. A. Hassan, "Driving sleepiness detection using electrooculogram analysis and grey wolf optimizer," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 6, pp. 6034–6044, 2022, doi: 10.11591/ijece.v12i6.pp6034-6044.

[6] M. E. Elidrissi, E. Essoukaki, L. Ben Taleb, A. Mouhsen, and M. Harmouchi, "Drivers' drowsiness detection based on an optimized random forest classification and single-channel electroencephalogram," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 3, pp. 3398–3406, 2023, doi: 10.11591/ijece.v13i3.pp3398-3406.

[7] C. G. Pachón-Suescún, J. O. Pinzón-Arenas, and R. Jiménez-Moreno, "Abnormal gait detection by means of LSTM," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 2, pp. 1495–1506, 2020, doi: 10.11591/ijece.v10i2.pp1495-1506.

[8] B. Hardiansyah and A. Primasetya, "Sistem Deteksi Penggunaan masker (Face Mask Detection) Menggunakan Algoritma Deep Learning YOLOv4," *Stain. (SEMINAR Nas. Teknol. & SAINS)*, vol. 2, no. 1, pp. 313–318, 2023.

[9] Yudi Widhiyasana, Transmissia Semiawan, Ilham Gibran Achmad Mudzakir, and Muhammad Randi Noor, "Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 10, no. 4, pp. 354–361, 2021, doi: 10.22146/jnteti.v10i4.2438.

[10] M. D. Darojat, Y. A. Sari, and R. C. Wihandika, "Convolutional Neural Network untuk Klasifikasi Citra Makanan Khas Indonesia," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 11, pp. 4764–4769, 2021, [Online]. Available: http://j-ptiik.ub.ac.id

[11] D. Yolanda, K. Gunadi, and E. Setyati, "Pengenalan Alfabet Bahasa Isyarat Tangan Secara Real-Time dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural Network," *J. Infra*, vol. 8, no. 1, pp. 203–208, 2020, [Online]. Available: https://publication.petra.ac.id/index.php/teknik-informatika/article/view/9791

[12] Z. Chao, F. Pu, Y. Yin, B. Han, and X. Chen, "Research on real-time local rainfall prediction based on MEMS sensors," *J. Sensors*, vol. 2018, pp. 1–9, 2018, doi: 10.1155/2018/6184713.

[13] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional LSTM for anomaly detection," *Proc. - IEEE Int. Conf. Multimed. Expo*, no. July, pp. 439–444, 2017, doi: 10.1109/ICME.2017.8019325.