DETEKSI KELELAHAN MATA BERDASARKAN CITRA WAJAH MRL EYE DATASET DENGAN MENGGUNAKAN GLCM, PCA, DAN SVM

Jeremia Pinnywan Immanuel [1] Eunice Eugenia Karta [2] Rio Bun Dika [3]

[1) 2) 3)]Teknik Informatika Universitas Tarumanagara
Jln. Letjen S. Parman No. 1, Jakarta, 11440, Indonesia
email : [1)]jeremia.535210095@stu.untar.ac.id, [2)]eunice.535230177@stu.untar.ac.id,
[3)]rio.535220005@stu.untar.ac.id

ABSTRAK

Penelitian ini membahas implementasi metode GLCM, PCA, dan SVM untuk membangun sistem deteksi kelelahan mata berbasis pengolahan citra digital. Meningkatnya penggunaan perangkat elektronik menjadikan kelelahan mata sebagai isu kesehatan yang semakin serius, terutama bagi individu yang bekerja dalam bidang yang memerlukan konsentrasi visual tinggi dalam jangka waktu lama. Untuk menjawab kebutuhan tersebut, penelitian ini menggunakan pendekatan pembelajaran mesin klasik yang dikenal efisien dan ringan dari segi kebutuhan sumber daya, dengan memanfaatkan MRL Eye Dataset berbasis citra inframerah sebagai data utama. Tahapan pemrosesan citra dimulai dengan ekstraksi fitur tekstur menggunakan metode GLCM, yang kemudian direduksi dimensinya menggunakan teknik PCA, dan akhirnya dilakukan klasifikasi oleh algoritma SVM. Sistem ini diuji dalam dua skenario, yaitu dengan dan tanpa keberadaan data outlier. Penyesuaian parameter optimal dilakukan melalui metode Grid Search dan validasi silang menggunakan KFold dengan K = 3. Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi kondisi mata "Awake" dan "Sleepy" dengan akurasi tertinggi mencapai 92.38%, serta nilai precision, recall, dan F1-score yang masing-masing mampu memperoleh hasil di atas 92%. Selain itu, waktu eksekusi untuk ekstraksi fitur, pelatihan model, dan prediksi tergolong cepat, sehingga menunjukkan bahwa sistem ini berpotensi untuk diterapkan secara real-time, bahkan pada perangkat dengan spesifikasi perangkat keras yang terbatas.

Key words

Deteksi Kelelahan Mata, GLCM, Pengolahan Citra, PCA, SVM

1. Pendahuluan

Perkembangan teknologi digital saat ini membuat penggunaan komputer dan perangkat layar menjadi bagian penting dari aktivitas sehari-hari, baik dalam konteks pekerjaan maupun kehidupan pribadi [1]. Namun, menatap layar dalam durasi yang panjang tanpa jeda bisa menyebabkan mata menjadi lelah [2], [3]. Kondisi ini seringkali ditandai dengan gangguan penglihatan sementara, ketidaknyamanan, serta penurunan konsentrasi yang pada akhirnya dapat memengaruhi produktivitas kerja, terutama pada tugas-tugas yang bersifat repetitif dan memerlukan fokus visual tinggi seperti mengetik, memasukkan data, maupun melakukan analisis visual [4].

Melihat pentingnya menjaga kesehatan mata, muncul kebutuhan untuk merancang sistem otomatis yang mampu mengenali tanda-tanda kelelahan mata secara cepat dan akurat. Salah satu pendekatan yang dapat digunakan adalah melalui analisis citra wajah dengan teknik pengolahan citra digital menggunakan pendekatan pembelajaran berbasis mesin, yang sudah terbukti banyak digunakan [5], [6]. Dalam studi ini, kami menggunakan *MRL Eye Dataset* sebagai sumber data citra mata. Selanjutnya, ciri tekstur dari citra diekstraksi menggunakan metode *Gray Level Co-occurrence Matrix* (GLCM), kemudian direduksi dimensinya menggunakan *Principal Component Analysis* (PCA), dan diklasifikasikan dengan *Support Vector Machine* (SVM).

Penelitian terdahulu yang menggunakan *MRL Eye Dataset* sebelumnya sudah dilakukan, dengan menggunakan *Deep Convolutional Neural Network* (DCNN) dan *transfer learning* dengan arsitektur *pre-trained Inception* dan menghasilkan akurasi sebesar 94% [7]. Penelitian yang menggunakan pendekatan GLCM, PCA, dan SVM juga telah dilakukan dengan klasifikasi tumor pada otak dengan *Dataset* MRI scan dan menghasilkan 98.3% akurasi [8]. Namun, dikarenakan pendekatan menggunakan CNN secara natural menggunakan banyak sekali parameter, hal ini menjadi kendala bagi perangkat elektronik yang memiliki keterbatasan sumber daya [9], dan CNN pada dasarnya membutuhkan kemampuan GPU yang sangat besar. Oleh karena itu, penelitian ini akan menggunakan pendekatan tradisional pembelajaran mesin dengan ekstraksi fitur secara manual, agar diharapkan dapat menghasilkan sistem deteksi kelelahan mata yang andal dan efisien,

sehingga dapat mendukung upaya menjaga kesehatan visual sekaligus meningkatkan efektivitas kerja.

2. Metode Penelitian

2.1. *Dataset* MRL Eye

Dataset yang digunakan dalam penelitian ini adalah *MRL Infrared Eye Images Dataset for Drowsiness Detection*, yaitu versi turunan (*forked version*) dari *Dataset MRL Eye* yang tersedia secara publik di platform Kaggle. *Dataset* ini berisi citra mata manusia yang diambil menggunakan kamera inframerah dan telah diklasifikasikan ke dalam dua kategori utama, yaitu Awake (mata terbuka) dan Sleepy (mata tertutup). Citra-citra tersebut digunakan untuk membangun dan menguji model deteksi kelelahan mata berbasis citra wajah.

Dataset ini telah dipisah ke dalam tiga bagian utama, yaitu *training set*, *validation set*, dan *test set*, dengan total lebih dari 85.000 gambar yang direkam dalam berbagai kondisi pencahayaan dan menggunakan beragam sensor. Pembagian data dirancang agar seimbang antara dua kelas, yaitu mata terbuka dan tertutup. Rincian struktur data adalah sebagai berikut:

1. *Training set*: 25.770 citra Awake dan 25.167 citra Sleepy
2. *Validation set*: 8.591 citra Awake dan 8.389 citra Sleepy
3. *Test set*: 8.591 citra Awake dan 8.390 citra Sleepy

Citra dalam *Dataset* ini disimpan dalam struktur direktori yang terorganisir, dengan tiga folder utama: */train*, */val*, dan */test*, di mana masing-masing folder memiliki dua subdirektori: /Awake dan /Sleepy. Format penamaan file citra mencerminkan informasi penting seperti ID subjek, kondisi mata, jenis kelamin, penggunaan kacamata, intensitas cahaya, dan ID sensor.

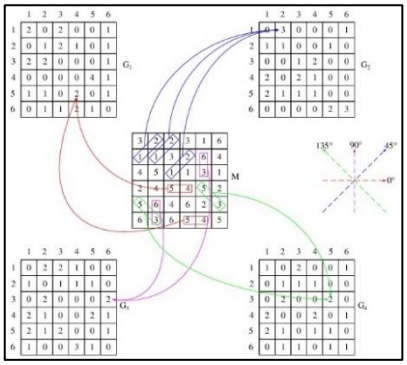
Dataset ini dirancang untuk berbagai aplikasi dalam bidang *computer vision* seperti deteksi mata, estimasi arah pandang, pendeteksian kedipan, serta deteksi kantuk atau kelelahan berbasis citra mata. *Dataset* ini sangat sesuai digunakan dalam pengembangan sistem *real-time* karena citra yang disediakan adalah dalam format *grayscale* dan beresolusi rendah, sehingga ringan untuk diproses.

Dataset ini berasal dari repositori yang diunggah oleh pengguna Kaggle bernama Akash Shingha. Informasi selengkapnya mengenai struktur dan metadata *Dataset* dapat ditemukan melalui platform Kaggle [10].

2.2. *Gray-Level Co-occurrence Matrix* (GLCM)

Gray-Level Co-occurrence Matrix (GLCM) merupakan salah satu metode statistik orde kedua yang digunakan untuk ekstraksi fitur tekstur dalam pengolahan citra digital [11]. Metode ini menghitung frekuensi kemunculan pasangan piksel dengan tingkat keabuan tertentu dalam posisi relatif tertentu, baik dari segi jarak maupun arah [12]. GLCM sangat bermanfaat untuk menggambarkan karakteristik tekstur yang tidak dapat dianalisis hanya dengan informasi histogram tingkat keabuan, karena memperhatikan hubungan spasial antar piksel. Informasi ini penting untuk membedakan pola-pola tekstur seperti permukaan halus, kasar, seragam, atau berulang [13].

Keunggulan utama dari GLCM terletak pada kemampuannya menangkap informasi tekstur lokal secara detail. Metode ini dapat diterapkan dalam berbagai arah (0°, 45°, 90°, dan 135°) dan jarak antar piksel, sehingga memberikan representasi tekstur yang lebih deskriptif [14]. GLCM digunakan secara luas dalam berbagai aplikasi klasifikasi tekstur, termasuk dalam pengenalan wajah, analisis gambar medis, dan identifikasi permukaan objek pada citra *grayscale*.



Gambar 1. Matriks GLCM dan arah orientasi piksel

Untuk memahami penerapan GLCM secara sistematis, berikut adalah rumus dan tahapan utama:

1. Pembangunan Matriks GLCM

GLCM dibentuk dengan menghitung frekuensi kemunculan pasangan piksel dengan tingkat keabuan tertentu dalam arah dan jarak tertentu, yang menggambarkan seberapa sering kombinasi nilai kecerahan muncul dengan posisi yang berbeda [15].

Setelah matriks GLCM dibentuk, langkah berikutnya adalah melakukan Normalisasi terhadap setiap elemen matriks, dengan tujuan untuk mendapatkan total keseluruhan elemen berjumlah 1, agar fitur statistik yang dihitung dari GLCM tidak bias terhadap ukuran citra [16].

Rumus Normalisasi GLCM dinyatakan sebagai berikut:

$$P_{norm}(i,j) = \frac{P(i,j)}{\sum_{i,j} P(i,j)} \quad (1)$$

2. Ekstraksi Fitur Statistik dari GLCM

Beberapa fitur statistik yang dapat dihitung dari GLCM antara lain:

a. *Contrast*

Mengukur intensitas perbedaan antara piksel berpasangan.

$$Contrast = \sum_{i,j} (i-j)^2 P(i,j) \quad (2)$$

b. *Homogeneity*

Menilai kedekatan distribusi elemen dalam GLCM terhadap diagonal.

$$Homogeneity = \sum_{i,j} \frac{P(i,j)}{1+|i-j|} \quad (3)$$

c. *Energy*

Mencerminkan keseragaman tekstur (juga dikenal sebagai ASM – Angular Second Moment).

$$Energy = \sum_{i,j} P(i,j)^2 \quad (4)$$

d. *Correlation*

Menilai hubungan linier antara piksel berpasangan.

$$Correlation = \sum_{i,j} \frac{(i-\mu i)(j-\mu j)P(i,j)}{\sigma_i \sigma_j} \quad (5)$$

Keterangan :

- $\mu i, \mu j$: Rata-rata dari baris dan kolom matriks GLCM
- σ_i, σ_j : Standar deviasi masing-masing

2.3. *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) merupakan salah satu metode statistik yang bertujuan untuk mengenali pola dalam suatu himpunan data serta merepresentasikan data tersebut dengan cara yang menonjolkan kesamaan dan perbedaan antar elemen data [17]. Teknik ini sangat bermanfaat dalam menganalisis data dengan dimensi tinggi, di mana visualisasi secara langsung menjadi tidak memungkinkan. Melalui PCA, jumlah dimensi dalam data dapat dikurangi tanpa kehilangan informasi yang esensial, sehingga metode ini banyak dimanfaatkan dalam proses ekstraksi fitur, kompresi data, serta mengatasi permasalahan multikolinearitas pada model prediktif seperti regresi.

Penerapan PCA didasarkan pada kemampuannya untuk menyederhanakan struktur data yang kompleks, yaitu dengan mengubah sekumpulan variabel yang saling berkorelasi menjadi sekumpulan variabel baru yang saling bebas (tidak berkorelasi), yang disebut sebagai komponen utama (*principal components*). Transformasi ini sangat membantu dalam proses analisis lanjutan, seperti pengenalan pola, klasifikasi, serta peningkatan efisiensi dalam pemrosesan data pada bidang *machine learning* maupun visualisasi data.

Beberapa keunggulan utama dari PCA antara lain: kemampuannya untuk mereduksi dimensi namun tetap dapat mempertahankan informasi [18]; mempermudah visualisasi dan interpretasi tren data; membantu mengurangi risiko *overfitting* pada algoritma pembelajaran mesin dengan menghilangkan multikolinearitas antar fitur; serta efisiensinya dalam kompresi data, terutama pada konteks pengolahan citra seperti pengenalan wajah atau analisis tekstur.

Namun demikian, PCA juga memiliki sejumlah keterbatasan yang perlu diperhatikan. Komponen utama yang dihasilkan bersifat abstrak dan sulit untuk diinterpretasikan karena merupakan hasil kombinasi linier dari banyak variabel asli. Selain itu, PCA sangat sensitif terhadap keberadaan *outlier*, yang dapat memberikan hasil yang salah [19]. Metode ini juga mengasumsikan bahwa hubungan antar variabel bersifat linier, sehingga efektivitasnya menurun apabila digunakan pada data dengan relasi *non-linear*. Oleh karena itu, dalam implementasinya, PCA memerlukan proses standarisasi awal agar setiap fitur memiliki skala kontribusi yang setara, serta pemeriksaan terhadap distribusi dan kestabilan data.

Untuk memahami proses PCA secara sistematis, berikut ini adalah tahapan-tahapan utama beserta rumus matematis yang digunakan :

1. Standarisasi Data :

Sebelum melakukan PCA, data perlu di standarisasi agar setiap variabel berada pada skala yang sama.

$$z_i = \frac{x_i - \overline{x}}{s^2} \quad (6)$$

Keterangan :

- z_i : Nilai hasil standarisasi
- x_i : Nilai asli dari variabel ke-i
- $\overline{x}$: Rata-rata dari variabel
- s^2 : Varians dari variabel

2. Matriks Kovarians

Digunakan untuk mengukur seberapa besar hubungan antar variabel.

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \quad (7)$$

Keterangan :

- x_i, y_i : Nilai pengamatan untuk variabel X dan Y
- $\overline{x}, \overline{y}$: Nilai rata-rata dari masing-masing variabel
- n : Jumlah data observasi

3. Nilai *Eigen* dan Vektor *Eigen*

Digunakan untuk menentukan arah komponen utama.

$$C\mathrm{v} = \lambda \mathrm{v} \quad (8)$$

Keterangan :

- C : Matriks Kovarians
- V : Vektor *Eigen* (komponen utama)
- Λ : Nilai *Eigen* (*Eigenvalue*)

4. Transformasi Data ke Komponen Utama

Setelah vektor *Eigen* ditentukan, data diproyeksikan ke ruang baru:

$$Z = X \,.\, W \quad (9)$$

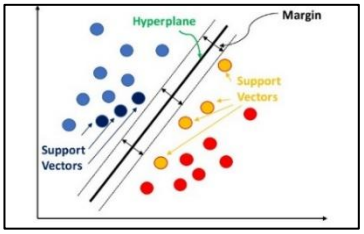
Keterangan :

- Z : Data hasil transformasi (komponen utama)

- X : Data awal yang telah di standarisasi
- W : Matriks yang berisi vektor-vektor *Eigen* terpilih

2.4. *Support Vector Machine* (SVM)

Salah satu algoritma yang digunakan dalam tugas klasifikasi adalah *Support Vector Machine* (SVM), yang bekerja dengan mencari *hyperplane* terbaik untuk memisahkan data ke dalam dua kelas berbeda dengan *margin* yang maksimal [20]. *Margin* tersebut merepresentasikan jarak terdekat antara data dari dua kelas terhadap *hyperplane*, sehingga semakin besar *margin*, semakin baik generalisasi model [21]. Pada SVM, terdapat juga *support vectors* yang berada di posisi yang dekat dengan *hyperplane*, yang bermanfaat dalam penentuan posisi dan orientasi pemisah.



Gambar 2. Ilustrasi *margin* dan *hyperplane* pada SVM

Keunggulan utama dari SVM adalah karena mendukung teknik *Kernel trick* yang memungkinkan pemisahan data *non-linear* dengan cara memetakan data ke ruang berdimensi lebih tinggi. Namun demikian, SVM juga memiliki beberapa keterbatasan. Waktu komputasi menjadi relatif tinggi saat digunakan pada *Dataset* besar dikarenakan kompleksitas waktu dan ruang meningkat secara linear semakin bertambahnya jumlah data [22], dan performa model sangat bergantung pada pemilihan *Kernel* dan parameter (seperti konstanta regulasi C dan parameter *Kernel* seperti γ). Selain itu, interpretasi model SVM terutama dengan *non-Linear Kernel* cenderung kompleks dan tidak intuitif.

Berdasarkan keunggulan dari SVM dalam menangani data *non-linear* yang memanfaatkan *kernel trick*, maka terdapat beberapa macam kernel yang umum digunakan sebagai berikut [23]. *Linear Kernel* digunakan ketika data dapat dipisahkan dengan garis lurus atau bidang datar [24]. *Kernel* ini sederhana dan sangat efisien secara komputasi. Kemudian terdapat *Polynomial Kernel* yang memungkinkan pemisahan data dalam bentuk kurva dengan tingkat kompleksitas yang dikontrol oleh derajat polinomial. *Kernel* ini cocok digunakan ketika hubungan antar fitur bersifat *non-linear* dan di ruang fitur, kernel ini mengukur kesamaan antara vektor input pelatihan berdasarkan kombinasi polinomial dari variabel asli [25]. Terakhir adalah *Radial Basis Function* (RBF) *Kernel*, juga dikenal sebagai *Gaussian Kernel*, yang paling sering digunakan dalam praktik. *Kernel* ini melakukan perhitungan jarak yang diperhalus dengan memanfaatkan fungsi radial atau fungsi eksponensial [26].

Untuk memahami proses klasifikasi menggunakan SVM secara sistematis, berikut adalah formulasi dan tahapan matematisnya:

1. Penentuan *Hyperplane Optimal* (Data Linear)

SVM *linear* bertujuan menemukan *hyperplane* optimal yang memisahkan dua kelas dengan *margin* maksimum, dengan fungsi keputusan sebagai berikut:

$$f(x) = w^T x + b \tag{10}$$

Dengan syarat :

$$y_i(w^T x_i + b) \geq 1, \forall i \tag{11}$$

Tujuan dari optimasi ini adalah :

$$min_{(w,b)} \frac{1}{2} \| w \|^2 \tag{12}$$

Keterangan :
- w : Vektor bobot
- b : Bias (intersep)
- x_i : Vektor fitur ke-i
- $y_i \in \{-1, +1\}$: Label kelas

2. *Soft Margin*

Dengan memperkenalkan variabel kelonggaran (*slack variable*) ξ_i, rumus optimasi menjadi:

$$min_{(w,b,\xi)} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i \tag{13}$$

Dengan syarat :

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \tag{14}$$

Keterangan :
- w : Vektor bobot
- b : Bias (intersep)
- ξ_i : *Slack variable* untuk mengizinkan pelanggaran *margin*
- C : Parameter regulasi untuk mengontrol *trade-off* antara *margin* dan kesalahan klasifikasi
- $y_i(w^T x_i + b) \geq 1$: Label Kelas
- x_i : Vektor fitur ke-i

3. *Kernel* Trick (Data Non-Linier)

Jika data tidak linier secara eksplisit, *Kernel* digunakan untuk mentransformasikan data ke ruang berdimensi lebih tinggi tanpa perlu menghitung proyeksinya secara langsung.

Contoh fungsi *Kernel* :
- *Linear Kernel*

$$K(x, x') = x^T x' \tag{15}$$

- *Polynomial Kernel*

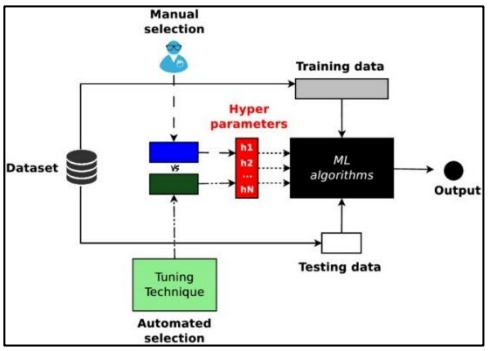
$$K(x, x') = (x^T x' + c)^d \quad (16)$$

- *Radial Basis Function* (RBF) / *Gaussian*

$$K(x, x') = \exp(-\gamma \| x - x' \|^2) \quad (17)$$

2.5. *Hyperparameter tuning (Grid Search)*

Hyperparameter tuning adalah proses untuk menemukan kombinasi *hyperparameter* terbaik dari sebuah pendekatan algoritma pembelajaran yang nantinya akan menghasilkan model pembelajaran mesin terbaik [27]. Salah satu metode yang paling umum digunakan adalah *Grid Search*, yaitu pencarian *hyperparameter* secara sistematis berdasarkan kombinasi nilai-nilai yang telah ditentukan sebelumnya. Metode ini menguji semua kemungkinan kombinasi *hyperparameter* secara otomatis.



Gambar 3. Proses *tuning* parameter model

Tahapan *Grid Search* :

1. Tentukan daftar nilai-nilai kandidat untuk tiap *hyperparameter*.
2. Lakukan pelatihan model untuk setiap kombinasi nilai.
3. Evaluasi performa masing-masing kombinasi menggunakan data validasi atau *cross-validation*.
4. Pilih kombinasi dengan skor evaluasi terbaik.

Rumus pencarian kombinasi parameter terbaik dapat dilakukan dengan memanfaatkan proses *k-fold cross validation* yang melakukan pengujian dan menghasilkan nilai *loss*, berdasarkan rumus berikut :

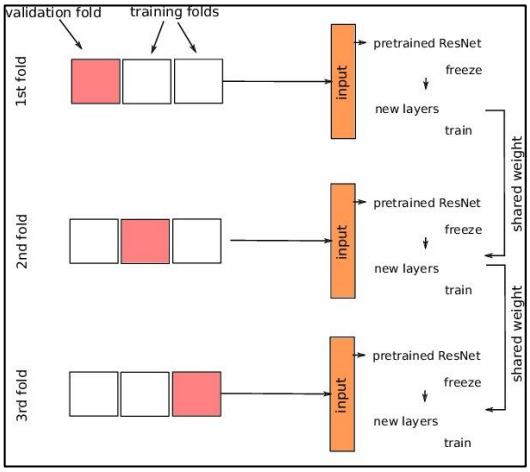
$$(C^*, \gamma^*) = \min L(C, \gamma) \quad (17)$$

Keterangan :

- (C^*, γ^*) : Kombinasi *hyperparameter* terbaik yang menghasilkan loss terendah
- $L(C, \gamma)$: Nilai *loss* hasil evaluasi model berdasarkan kombinasi parameter C dan γ

2.6. *K-Fold Cross Validation*

K-Fold Cross Validation adalah teknik evaluasi dengan membagi data ke dalam K bagian (*fold*). Model akan dilatih dan diuji sebanyak K kali, di mana setiap *fold* bergiliran menjadi data uji, sementara sisanya digunakan untuk pelatihan [28]. Dalam studi ini digunakan skema *3-fold cross validation*.



Gambar 4. Skema *3-fold cross validation*

Rumus perhitungan nilai rata-rata performa model adalah sebagai berikut :

$$L_{avg} = \frac{1}{K} \sum_{i=1}^{K} L_i \quad (18)$$

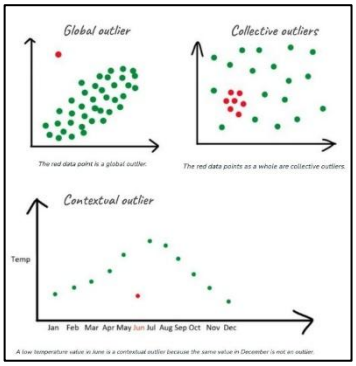
Keterangan :

- L_{avg} : Nilai rata-rata hasil *loss* model
- K : Jumlah *fold*, dalam penelitian ini $K = 3$
- L_i : Nilai *loss* pada iterasi ke-i

2.7. Deteksi dan Penanganan *Outlier*

Outlier merupakan data atau observasi yang memiliki nilai yang secara signifikan berbeda dari mayoritas distribusi data [29]. Keberadaan *outlier* dapat mengindikasikan berbagai kondisi, seperti kesalahan dalam proses pengumpulan data, pengaruh faktor eksternal yang tidak teramati, atau fenomena khusus yang memang terjadi secara tidak umum. Dalam konteks analisis data dan *machine learning*, identifikasi dan penanganan *outlier* sangat penting karena keberadaannya dapat menyebabkan distorsi pada proses pelatihan model maupun hasil evaluasi.

Secara umum, *outlier* diklasifikasikan ke dalam tiga jenis :



Gambar 5 Jenis-jenis *outliers* dalam data

1. *Global Outlier*: Data yang secara ekstrem berada di luar pola distribusi umum seluruh *Dataset*.
2. *Collective/Contextual Outlier*: Nilai yang tampak Normal pada satu kondisi, tetapi menjadi tidak wajar bila dilihat dalam konteks waktu atau ruang tertentu.
3. *Group Outlier*: Sekumpulan data yang secara kolektif menyimpang dari distribusi utama, walaupun secara individu tidak tampak sebagai anomali.

Dalam praktik data mining, *outlier* dapat sangat memengaruhi teknik-teknik statistik seperti *Principal Component Analysis* (PCA) [30], yang sensitif terhadap distribusi nilai ekstrim. Oleh karena itu, pendeteksian dan penanganan *outlier* perlu dilakukan sebelum tahap pemodelan.

Salah satu pendekatan populer untuk mendeteksi *outlier* adalah menggunakan *Mahalanobis distance*, yang mengukur jarak antara satu titik data dan distribusi multivariat dari keseluruhan data. Metode ini mempertimbangkan korelasi antar variabel, dan biasanya dikombinasikan dengan uji *Chi-Square* untuk menentukan batas ambang deteksi *outlier*.

Rumus *Mahalanobis distance* dinyatakan sebagai berikut:

$$D^2 = (x-\mu)^T S^{-1}(x-\mu) \qquad (19)$$

Keterangan :

- D^2 : *Mahalanobis distance* kuadrat dari data x terhadap mean
- x : Vektor data
- μ : Vektor rata-rata dari seluruh data
- S^{-1} : Invers matriks kovarians dari *Dataset*

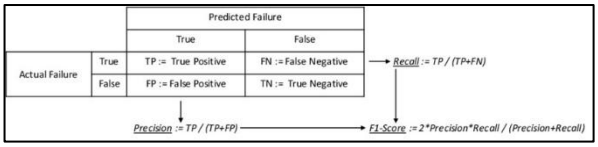
2.8. Evaluasi Kinerja Model

Evaluasi performa model klasifikasi merupakan tahap penting dalam mengukur efektivitas algoritma yang digunakan. Dalam kasus klasifikasi biner, terutama pada data yang tidak seimbang, penggunaan metrik evaluasi yang lebih mendetail seperti *Precision*, *Recall*, dan *F1-score* untuk masing-masing kelas menjadi sangat krusial. Ketiga metrik ini memberikan pandangan yang lebih adil dan seimbang terhadap kinerja model, dibandingkan hanya mengandalkan akurasi secara keseluruhan, yang cenderung bias terhadap kelas mayoritas.

Precision mengindikasikan seberapa akurat prediksi positif yang dihasilkan model terhadap data yang relevan, sedangkan *Recall* menunjukkan sejauh mana model mampu mengenali seluruh data positif yang sebenarnya. *F1-score*, yang merupakan rata-rata harmonis dari *Precision* dan *Recall*, digunakan untuk menggambarkan keseimbangan antara keduanya secara menyeluruh. Pendekatan ini menjadi sangat penting dalam konteks data tidak seimbang, di mana satu kelas (misalnya "Sleepy") mungkin jauh lebih sedikit jumlahnya dibanding kelas lainnya ("Awake"). Jika hanya menggunakan akurasi, model bisa tampak baik meskipun gagal mengenali kelas minoritas secara konsisten [31].

Penelitian pada [32] memperkuat pentingnya pendekatan evaluasi berdasarkan masing-masing kelas. Dalam studi mereka yang berfokus pada sistem deteksi intrusi, dilakukan pengukuran metrik *Precision*, *Recall*, dan *F1-score* secara terpisah untuk dua kelas, yaitu "Normal" dan "*Anomaly*". Evaluasi terpisah ini memungkinkan analisis performa model menjadi lebih mendalam dan tidak hanya bergantung pada nilai keseluruhan. Pendekatan ini dinilai lebih representatif karena memberikan gambaran yang akurat mengenai kemampuan model dalam menangani setiap kategori secara adil. Hal tersebut menjadi sangat penting, terutama dalam penerapan nyata seperti sistem keamanan, deteksi gangguan medis, atau sistem pemantauan kelelahan, di mana akurasi dalam mengenali kasus minoritas sering kali memiliki dampak yang lebih kritis dibandingkan kelas mayoritas.

Untuk melakukan evaluasi secara kuantitatif, digunakan *Confusion matrix*, yang dapat digunakan untuk memberikan penilaian hasil kerja dari model klasifikasi mengenai salah dan benar [33], dalam bentuk tabel 2x2 :



Gambar 6. *Confusion matrix* dan metrik evaluasi

- *True Positive* (TP): Model memprediksi positif dan benar
- *False Positive* (FP): Model memprediksi positif tapi salah
- *False Negative* (FN): Model memprediksi negatif tapi salah
- *True Negative* (TN): Model memprediksi negatif dan benar

Berdasarkan komponen ini, dapat dihitung metrik evaluasi sebagai berikut:

1. *Precision*

Precision mengukur sejauh mana nilai positif yang diprediksi oleh model adalah benar [34]. Dalam konteks klasifikasi biner, *Precision* menunjukkan seberapa banyak dari seluruh prediksi kelas positif (misalnya

"Sleepy") yang memang benar-benar termasuk ke dalam kelas tersebut. Nilai *Precision* yang tinggi menandakan bahwa model jarang memberikan prediksi positif palsu (*False Positive*).

$$Precision = \frac{TP}{TP + FP} \quad (20)$$

2. *Recall*

Recall mengukur kemampuan model dalam menemukan seluruh kasus positif yang benar [35]. Nilai *Recall* yang tinggi menunjukkan bahwa model berhasil menangkap sebagian besar data dari kelas positif dan tidak melewatkan banyak (*False Negative*). Ini sangat penting dalam kasus di mana deteksi kasus positif yang hilang dapat berisiko tinggi, seperti pada deteksi kelelahan.

$$Recall = \frac{TP}{TP + FN} \quad (21)$$

3. *F1-score*

F1-score adalah pengukuran nilai rata-rata harmonik berdasarkan metrik *Recall* dan *Precision* [36]. Metode ini digunakan untuk menyeimbangkan keduanya, terutama dalam situasi di mana terdapat *trade-off* antara *Precision* dan *Recall*. *F1-score* memberikan gambaran menyeluruh terhadap kinerja model ketika perlu mempertimbangkan baik kesalahan tipe I (FP) maupun kesalahan tipe II (FN).

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (22)$$

Nilai *F1-score* berkisar antara 0 hingga 1, di mana nilai 1 menunjukkan performa sempurna. *F1-score* sangat bermanfaat ketika data tidak seimbang dan ketika kesalahan klasifikasi pada kedua kelas memiliki konsekuensi yang sama pentingnya.

Evaluasi model menggunakan metrik-metrik ini tidak hanya memberikan gambaran umum terhadap akurasi prediksi, tetapi juga membantu dalam menilai kemampuan model dalam mengenali kelas minoritas, yang pada konteks deteksi kelelahan mata menjadi aspek penting dalam mencegah konsekuensi serius, seperti kecelakaan akibat kantuk yang tidak terdeteksi.

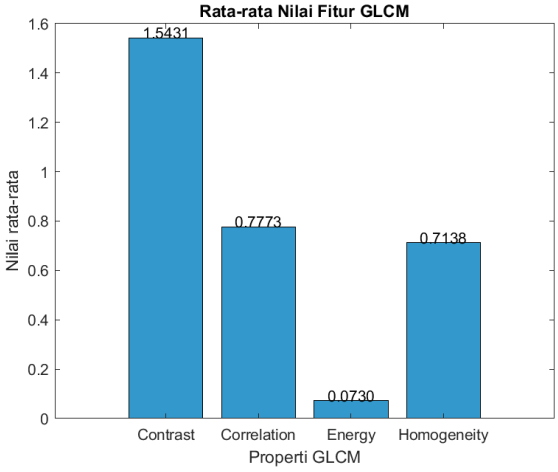
3. Hasil Percobaan

Setelah pengujian dilakukan, didapatkan hasil berupa visualisasi plot yang akan dianalisis lebih lanjut untuk memahami keberhasilan dari penelitian yang telah dijalani. Pada bagian ini, akan dilakukan analisis lebih lanjut dari setiap visualisasi plot yang dihasilkan untuk memberikan pemahaman lebih mendalam mengenai efektivitas pengujian yang telah dilaksanakan.

3.1 Ekstraksi Fitur dengan GLCM

Gambar 7 merupakan hasil berupa visualisasi plot mengenai nilai rata-rata dari fitur berdasarkan keempat properti yang digunakan, yakni *Contrast*, *Correlation*, Energi, dan *Homogeneity*. *Contrast* yang ditunjukkan dengan nilai lebih dari 1.5431 menunjukkan bahwa setiap piksel yang berdekatan memiliki perbedaan intensitas yang cukup besar, menunjukkan bahwa citra memiliki banyak tepi. Berikutnya nilai rata-rata dari properti *Correlation*, dengan nilai 0.7773, menyatakan bahwa setiap piksel yang berdekatan memiliki hubungan *linear* yang yang cukup signifikan, yang menandakan ada pola yang relatif konsisten dalam citra.

Kemudian nilai rata-rata dari fitur pada properti energi sebesar 0.0730, mengindikasikan pola yang cenderung acak dan tidak teratur pada gambar. Terakhir adalah nilai rata-rata fitur pada properti *Homogeneity* dengan nilai 0.7138, menunjukkan bahwa dalam beberapa area, piksel-piksel memiliki intensitas yang mirip, mencerminkan keberadaan zona dengan tekstur halus dan keteraturan lokal. Dari hasil yang didapatkan, nilai *Contrast* yang tinggi tetapi nilai *Homogeneity* tinggi menunjukkan gambar memiliki struktur tekstur campuran, dengan banyak area yang memiliki perbedaan yang tajam namun juga memiliki area yang relatif homogen atau selaras. Dengan demikian ini menunjukkan bahwa distribusi tekstur tidak seragam, seperti terdapat permukaan yang sebagian halus dan sebagian kasar.



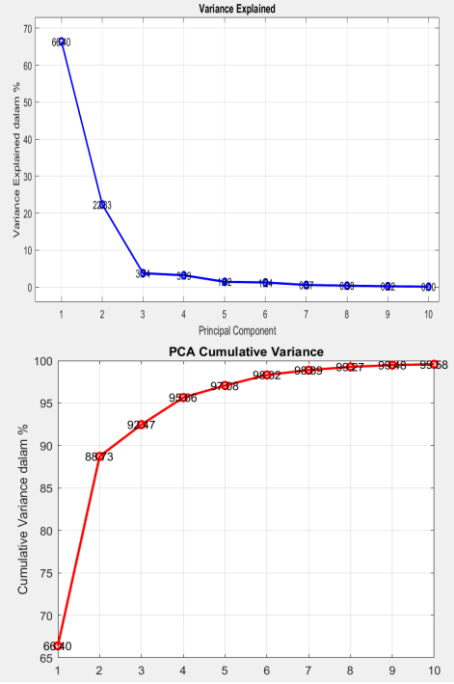
Gambar 7. Plot rata-rata nilai fitur GLCM

3.2 Reduksi Dimensi dengan PCA

Berdasarkan hasil PCA yang ditunjukkan pada Gambar 8 sebagai plot yang menjelaskan variansi dan kumulatif varians, ditunjukkan bahwa komponen utama pertama (PC1) mampu menjelaskan 66.40% variansi dari keseluruhan data. Komponen kedua (PC2) menambahkan 22.83% sehingga akumulasi dari komponen pertama dan komponen kedua mencapai 88.73%. Sementara itu hingga akumulasi terhadap komponen ketiga (PC3), total variansi yang dapat menjelaskan keseluruhan data mencapai

92.47%. Grafik kumulatif menunjukkan bahwa setelah tiga komponen pertama, kontribusi setiap komponen utama terhadap total variansi cenderung menurun drastis. Hal ini dapat terlihat pada grafik "*Variance Explained*", yang menunjukkan mulai dari PC4 dan seterusnya, masing-masing menyumbang kurang dari 5% variansi.

Pemilihan ambang batas kumulatif variansi ditetapkan menjadi 99%, sebagai kriteria pemilihan jumlah komponen utama pada PCA. Pendekatan ini digunakan guna memastikan bahwa informasi sebanyak mungkin dari fitur asli tetap dipertahankan, mengingat bahwa fitur hasil ekstraksi dari GLCM mengandung properti yang penting dalam memberikan informasi perbedaan tekstur pada gambar yang mungkin penting untuk proses klasifikasi. Dari sisi jumlah komponen utama didapatkan, sebanyak 8 komponen utama yang berhasil didapatkan dari hasil penetapan ambang batas sebesar 99%. Reduksi yang dihasilkan dari 96 fitur menjadi 8 fitur menunjukkan kompromi yang baik antara efisiensi komputasi dan juga retensi informasi.

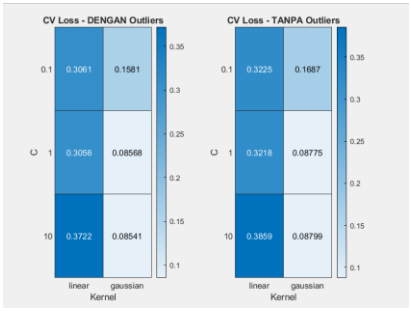


Gambar 8. Plot nilai *Variance Explained* dan *Cumulative Variance*

3.3 Perbandingan Kinerja

1. Hasil *Hyperparameter Tuning* dan *Cross Validation*

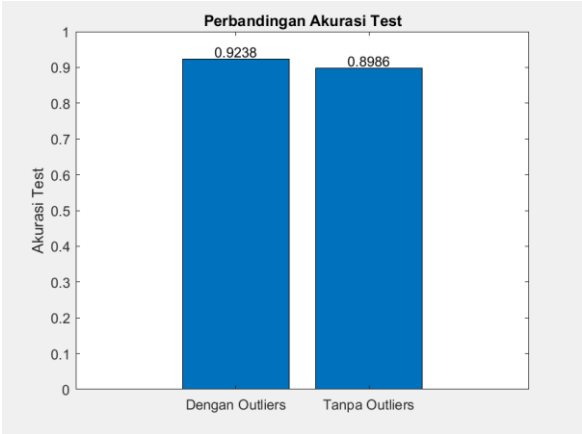
Setelah *Hyperparameter Tuning* dilakukan dari semua kombinasi *hyperparameter* yang ada pada SVM, yakni dari jenis Kernel yang digunakan (linear dan Gaussian) dan nilai C (0.1, 1, 10), beserta validasi dengan menggunakan *K-Fold Cross Validation* dengan K = 3, didapatkan hasil *loss* yang menunjukkan seberapa besar kesalahan yang didapatkan dari percobaan dengan masing-masing kombinasi *hyperparameter*. Hasil pada Gambar 9 menunjukkan bahwa baik untuk data dengan outliers maupun tanpa outliers, nilai *loss* terendah berada pada *Kernel Gaussian*. Untuk data dengan *outliers*, nilai C optimal merupakan 10, dengan nilai *loss* sebesar 0.08541, sedangkan untuk data tanpa *outliers*, nilai C terbaik merupakan 1 dengan nilai *loss* yang dihasilkan sebesar 0.08775.



Gambar 9. Plot perbandingan nilai CV *Loss* setiap kombinasi *hyperparameter*

2. Perbandingkan Akurasi Test

Berdasarkan hasil akurasi test pada Gambar 10, Pelatihan SVM dengan menggunakan *Dataset* yang tidak mengalami pembersihan *outliers* memberikan akurasi yang lebih tinggi, yakni 0.9238, bila dibandingkan dengan pelatihan SVM menggunakan *Dataset* yang sudah dibersihkan dengan *outliers* dengan akurasi 0.8986.

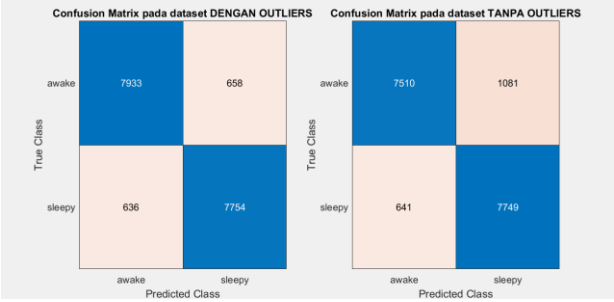


Gambar 10. Plot untuk perbandingan akurasi pada data test

3. Perbandingan *Confusion matrix*

Gambar 11 menunjukkan hasil berupa *Confusion matrix* yang berisi nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) dari hasil prediksi model. Pada pelatihan SVM dengan menggunakan *Dataset* yang tidak dilakukan pembersihan *outliers*, *Confusion matrix* menunjukkan hasil yang lebih baik (*True Positive* sebanyak 7933 dan *True Negative* sebanyak 7754) bila dibandingkan *Dataset* yang sudah

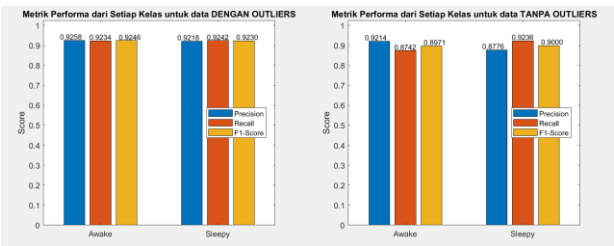
dilakukan pembersihan *outliers* (*True Positive* sebanyak 7510 dan *True Negative* sebanyak 7749). Dengan demikian, hasil ini berkaitan dari kesalahan prediksi yang dihasilkan dari model yang dilatih oleh *Dataset* dengan *outliers* yang lebih sedikit, dengan *False Positive* sebesar 636 dan *False Negative* sebesar 658, bila dibandingkan dengan kesalahan prediksi dari *Dataset* tanpa *outliers* yang lebih banyak (*False Positive* sebesar 641 dan *False Negative* sebesar 1081). Hal ini lantas berkaitan dengan hasil akurasi pada Gambar 10 yang menyatakan akurasi pada *Dataset* dengan *outliers* yang lebih baik.



Gambar 11. Plot untuk perbandingan hasil *Confusion matrix*

4. Perbandingan *Precision*, *Recall*, dan *F1-score*

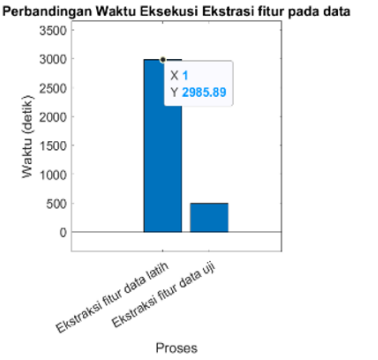
Gambar 12 menunjukkan nilai dari metrik *Precision*, *Recall*, dan *F1-score* yang berkaitan dengan hasil *Confusion matrix* pada Gambar 11, dikarenakan perhitungan setiap metrik dilakukan berdasarkan nilai *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Hasil pada Gambar 12 menunjukkan bahwa baik dari nilai *Precision*, *Recall*, dan *F1-score* dari masing-masing kelas, model yang dilatih dari *Dataset* tanpa pembersihan *outliers* memberikan hasil yang lebih baik dengan semua skor berada di atas 92% atau 0.92 bila dibandingkan dengan model yang dilatih dengan *Dataset* yang sudah dibersihkan dengan *outliers*. Hal ini juga selaras dari hasil *Confusion matrix* pada Gambar 11, yang menyatakan bahwa model yang dilatih oleh *Dataset* dengan *outliers* memberikan nilai *True Positive* dan *True Negative* yang lebih banyak, dan *False Positive* dan *False Negative* yang lebih sedikit bila dibandingkan dengan model yang dilatih dengan *Dataset* yang sudah dilakukan pembersihan *outliers*.



Gambar 12. Plot untuk perbandingan Precision, *Recall*, dan *F1-score*

5. Analisis Waktu Eksekusi

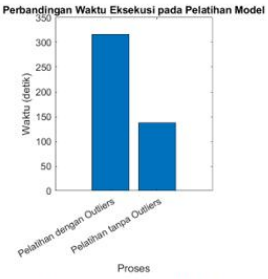
Selain metrik performa hasil prediksi, waktu komputasi juga dilakukan perhitungan demi mengetahui berapa lama proses dari ekstraksi dari proses GLCM, pelatihan dan waktu prediksi yang dilakukan oleh model SVM. Gambar 13 menunjukkan lama waktu ekstraksi gambar dengan GLCM untuk data latih dan data uji. Lama waktu komputasi yang dijalankan pada kumpulan gambar data uji berada di atas 2500 detik, sedangkan untuk ekstraksi gambar pada data uji berada di bawah 500 detik. Hal ini juga berkaitan dengan jumlah yang diekstrak untuk data latih jauh lebih banyak bila dibandingkan data uji.




```
Waktu ekstraksi fitur untuk pelatihan: 2985.8940 detik
Waktu ekstraksi fitur untuk pengujian: 491.0376 detik
```

Gambar 13. Plot dan *output* perbandingan waktu eksekusi ekstraksi data citra

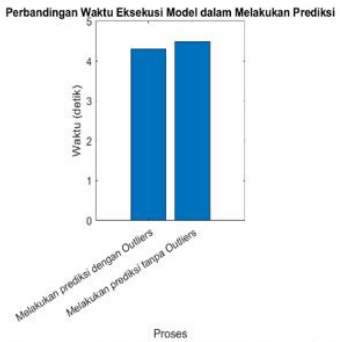
Gambar 14 kemudian menunjukkan waktu pelatihan yang dijalankan algoritma SVM. Plot menunjukkan bahwa pelatihan yang dilakukan terhadap *Dataset* yang masih memiliki *outliers* jauh lebih lama bila dibandingkan pelatihan yang dilakukan terhadap *Dataset* yang sudah tidak memiliki *outliers*. Dari Gambar 14 juga dapat diketahui lama waktu pelatihan yang dilakukan, untuk model dalam melatih *Dataset* dengan *outliers*, dijalankan selama 315.9380 detik, sedangkan untuk *Dataset* tanpa *outliers* dijalankan selama 137.5861 detik. Hal ini masuk akal mengingat penghapusan *outliers* menghilangkan beberapa data yang ada pada *Dataset*.




```
Waktu pelatihan model DENGAN OUTLIERS: 315.9380 detik
Waktu pelatihan model TANPA OUTLIERS: 137.5861 detik
```

Gambar 14. Plot dan *output* perbandingan waktu eksekusi pelatihan model

Setelah itu, Gambar 15 lebih lanjut menampilkan lama waktu prediksi pada model SVM yang telah dilatih. Sama seperti pada Gambar 14 yang menunjukkan lama waktu pelatihan pada algoritma SVM yang dilatih dengan data yang ada *outliers* dan data tanpa *outliers*, Gambar 15 juga menampilkan waktu prediksi dari model yang dilatih dengan data *outliers* dan data tanpa *outliers*. Model yang dilatih dengan data *outliers* melakukan prediksi selama 4.3103 detik, sedangkan model yang dilatih tanpa *outliers* melakukan prediksi selama 4.4857 detik.




```
Waktu prediksi model DENGAN OUTLIERS: 4.3103 detik
Waktu prediksi model TANPA OUTLIERS: 4.4857 detik
```

Gambar 15. Plot dan *output* perbandingan waktu eksekusi model dalam melakukan prediksi

4. Kesimpulan

Berdasarkan hasil eksperimen, dapat disimpulkan bahwa data citra mata dalam kondisi mengantuk dan mata dalam kondisi terjaga berhasil diekstraksi secara efektif dengan menggunakan metode GLCM. Fitur-fitur hasil ekstraksi dapat lebih lanjut direduksi dengan menggunakan PCA, dari 96 jumlah fitur menjadi 8 fitur dan dapat mempertahankan variasi total data sebesar 99%, sehingga proses reduksi ini tidak menghilangkan informasi penting dalam data sekaligus dapat meningkatkan efisiensi komputasi.

Dari segi performa dalam melakukan prediksi, pengujian yang dilakukan dengan data yang masih mempertahankan *outliers* memiliki kinerja yang baik dari segi akurasi, *Precision*, *Recall*, dan *F1-score*, bila dibandingkan dengan data yang sudah menghilangkan *outliers*. Hal ini menunjukkan bahwa beberapa *outliers* yang terdeteksi bukanlah anomali ataupun *noise*, melainkan mengandung informasi penting untuk klasifikasi. Selain itu, pemilihan *hyperparameter* telah ditemukan, yakni dengan menggunakan *Kernel Gaussian* dan C sebesar 10 untuk data dengan *outliers* dan C sebesar 1 untuk data tanpa *outliers*.

Dari segi efisiensi waktu, ekstraksi dari data citra membutuhkan waktu sekitar 57.95 menit, sedangkan untuk waktu komputasi pelatihan, model yang melakukan pelatihan dengan data yang mempertahankan *outliers* jauh lebih lama bila dibandingkan model yang melakukan pelatihan dengan data yang sudah dibersihkan dari *outliers*. Namun, proses prediksi sama-sama efisien untuk kedua jenis data, dengan waktu sekitar 4 detik.

Secara keseluruhan, kombinasi metode GLCM, PCA, dan SVM dengan konfigurasi optimal berhasil menghasilkan model klasifikasi mata mengantuk dan sadar yang akurat serta efisien. Model ini berpotensi untuk diimplementasikan pada sistem deteksi kantuk di perangkat dengan spesifikasi terbatas. Meskipun demikian, penelitian ini memiliki beberapa keterbatasan yang dapat menjadi pertimbangan untuk pengembangan selanjutnya. Beberapa saran perbaikan meliputi: penambahan kombinasi *hyperparameter*, penggunaan algoritma *Machine Learning* lain seperti XGBoost dan LightGBM, eksplorasi metode ekstraksi fitur selain GLCM, serta pengujian performa model dengan berbagai teknik penanganan *outliers*.

REFERENSI

[1] M. P. Maulana, M. Dimyati, and Aprilliantoni, "TEKNOLOGI INFORMASI DAN KOMUNIKASI PENDIDIKAN DALAM ARAH PEMBANGUNAN NASIONAL," *Idarah Tarbawiyah: Journal of Management in Islamic Education*, vol. 5, no. 6, pp. 655–664, Dec. 2024, doi: 10.32832/itjmie.v5i6.17161.

[2] Hana Salsabila Hafshah, Abdul Hadi Hassan, and H. Purnomo, "Hubungan Lama Penggunaan Komputer dan Intensitas Pencahayaan dengan Keluhan Kelelahan Mata pada Pekerja Rumah Sakit Amira," *Bandung Conference Series: Medical Science*, vol. 5, no. 1, pp. 53–58, Jan. 2025, doi: 10.29313/bcsms.v5i1.16136.

[3] A. K. A. Tianto, I. Qadrijati, and S. Haryati, "FAKTOR-FAKTOR YANG BERHUBUNGAN DENGAN KELUHAN KELELAHAN MATA PADA PEKERJA KANTOR X KARANGANYAR," *Jurnal Kesehatan Masyarakat*, vol. 11, no. 1, pp. 1–11, Jan. 2023, doi: 10.14710/jkm.v11i1.36786.

[4] Cahyo Wulandari, Rahmafari Fikra Maulida, Muhammad Asyam Fawwaz Akbar, and Abdurrahman Nur Prasetyo, "Promosi Kesehatan Mata melaui Kegiatan Skrining Mata pada Siswa SMP di Kecamatan Wedung, Demak," *Jurnal Pengabdian, Riset, Kreativitas, Inovasi, dan Teknologi Tepat Guna*, vol. 2, no. 1, pp. 39–45, May 2024, doi: 10.22146/parikesit.v2i1.9558.

[5] P. Singhal, P. K. Srivastava, A. K. Tiwari, and R. K. Shukla, "A Survey: Approaches to Facial Detection and Recognition with Machine Learning Techniques," 2022, pp. 103–125. doi: 10.1007/978-981-16-3346-1_9.

[6] M. A. Yaman, F. Rattay, and A. Subasi, "Comparison of Bagging and Boosting Ensemble Machine Learning Methods for Face Recognition," *Procedia Comput Sci*, vol. 194, pp.

202–209, 2021, doi: 10.1016/j.procs.2021.10.074.

[7] J. Sapkale, S. J. Bhosale, and R. Ingle, "An Eye State Recognition System using Transfer Learning: Inception-Based Deep Convolutional Neural Network," *International Research Journal of Engineering and Technology*, 2022, [Online]. Available: www.irjet.net

[8] P. Kumar Pagadala, D. Yaso Omkari, P. S. Lakshmi, C. Dewi, S. A. Sutresno, and J. Christanto, "AUTOMATED BRAIN TUMOR DETECTION WITH GLCM-BASED FEATURE EXTRACTION AND PCA FOR DIMENSION REDUCTION AND CLASSIFICATION USING MACHINE LEARNING," *J Theor Appl Inf Technol*, vol. 30, no. 12, 2024, [Online]. Available: www.jatit.org

[9] M. Zawish, S. Davy, and L. Abraham, "Complexity-Driven CNN Compression for Resource-constrained Edge AI," Aug. 2022, doi: 10.1109/TAI.2024.3353157.

[10] Akash Shingha, "MRL Eye Dataset." Accessed: Jun. 09, 2025. [Online]. Available: https://www.kaggle.com/datasets/akashshingha850/mrl-eye-dataset

[11] M. I. Mustofa, M. T. Furqon, and D. E. Ratnawati, "Penggunaan Metode Ekstraksi Fitur Tekstur Gray Level Co-ocurrrence Matrix dan K-Nearest Neighbor untuk Identifikasi Jenis Penyakit Tanaman Apel," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, pp. 4451–4458, Sep. 2022.

[12] M. Kabir, F. Unal, T. C. Akinci, A. A. Martinez-Morales, and S. Ekici, "Revealing GLCM Metric Variations across a Plant Disease Dataset: A Comprehensive Examination and Future Prospects for Enhanced Deep Learning Applications," *Electronics (Basel)*, vol. 13, no. 12, p. 2299, Jun. 2024, doi: 10.3390/electronics13122299.

[13] N. Neneng, A. S. Puspaningrum, and A. A. Aldino, "Perbandingan Hasil Klasifikasi Jenis Daging Menggunakan Ekstraksi Ciri Tekstur Gray Level Co-occurrence Matrices (GLCM) Dan Local Binary Pattern (LBP)," *SMATIKA JURNAL*, vol. 11, no. 01, pp. 48–52, Jul. 2021, doi: 10.32664/smatika.v11i01.572.

[14] V. Feriska Amalia and R. Rahma Dewi, "PENILAIAN KESEGARAN IKAN DENGAN METODE K-NEAREST NEIGHBOR DAN PENGOLAHAN CITRA DIGITAL," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 4, pp. 7823–7829, Aug. 2024, doi: 10.36040/jati.v8i4.10441.

[15] J. P. Sari, A. Erlansari, and E. P. Purwandari, "IDENTIFIKASI CITRA DIGITAL KURA-KURA SUMATERA DENGAN PERBANDINGAN EKSTRAKSI FITUR GLCM DAN GLRLM BERBASIS WEB," 2021. [Online]. Available: www.ejournal.unib.ac.id/index.php/pseudocode

[16] M. Laia, R. K. Hondro, and T. Zebua, "Implementasi Pengolahan Citra dengan Menggunakan Metode K-Nearest Neighbor Untuk Mengetahui Daging Ayam Busuk dan Daging Ayam Segar," *Jurnal Riset Komputer)*, vol. 8, no. 2, pp. 2407–389, 2021, doi: 10.30865/jurikom.v8i2.2818.

[17] A. Nurdiansyah and A. Ramadhanu, "SEGMENTASI CITRA BUAH DURIAN DAN JAGUNG DENGAN ALGORITMA K-NEAREST NEIGHBOR DAN PRINCIPAL COMPONEN ANALYSIS," *Jurnal Informatika Teknologi dan Sains (Jinteks)*, vol. 7, no. 1, pp. 412–419, Mar. 2025, doi: 10.51401/jinteks.v7i1.5359.

[18] E. A. Kusnanti, L. D. F. Vantie, and U. L. Yuhana, "SOFTWARE DEFECT PREDICTION USING PCA BASED RECURRENT NEURAL NETWORK," *JUTI: Jurnal Ilmiah Teknologi Informasi*, pp. 23–31, Jan. 2024, doi: 10.12962/j24068535.v22i1.a1199.

[19] S. Hussain, M. W. Mustafa, T. A. Jumani, S. K. Baloch, and M. S. Saeed, "A novel unsupervised feature-based approach for electricity theft detection using robust <scp>PCA</scp> and outlier removal clustering algorithm," *International Transactions on Electrical Energy Systems*, vol. 30, no. 11, Nov. 2020, doi: 10.1002/2050-7038.12572.

[20] S. S. Maulina Putri, M. Arhami, and H. Hendrawaty, "Penerapan Metode SVM pada Klasifikasi Kualitas Air," *Journal of Artificial Intelligence and Software Engineering (J-AISE)*, vol. 3, no. 2, p. 93, Nov. 2023, doi: 10.30811/jaise.v3i2.4630.

[21] A. Rizwan, N. Iqbal, R. Ahmad, and D.-H. Kim, "WR-SVM Model Based on the Margin Radius Approach for Solving the Minimum Enclosing Ball Problem in Support Vector Machine Classification," *Applied Sciences*, vol. 11, no. 10, p. 4657, May 2021, doi: 10.3390/app11104657.

[22] D. Mustafa Abdullah and A. Mohsin Abdulazeez, "Machine Learning Applications based on SVM Classification A Review," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 81–90, Apr. 2021, doi: 10.48161/qaj.v1n2a50.

[23] A. Kumar and D. Mishra, "Improving Support Vector Machine using Modified Kernel Function," *International Journal of Scientific Research and Modern Technology*, pp. 1–5, May 2025, doi: 10.38124/ijsrmt.v4i5.501.

[24] D. Jahed Armaghani, P. G. Asteris, B. Askarian, M. Hasanipanah, R. Tarinejad, and V. Van Huynh, "Examining Hybrid and Single SVM Models with Different Kernels to Predict Rock Brittleness," *Sustainability*, vol. 12, no. 6, p. 2229, Mar. 2020, doi: 10.3390/su12062229.

[25] S. Shojae Chaeikar, A. A. Manaf, A. A. Alarood, and M. Zamani, "PFW: Polygonal Fuzzy Weighted—An SVM Kernel for the Classification of Overlapping Data Groups," *Electronics (Basel)*, vol. 9, no. 4, p. 615, Apr. 2020, doi: 10.3390/electronics9040615.

[26] W. Sadewo, Z. Rustam, H. Hamidah, and A. R. Chusmarsyah, "Pancreatic Cancer Early Detection Using Twin Support Vector Machine Based on Kernel," *Symmetry (Basel)*, vol. 12, no. 4, p. 667, Apr. 2020, doi: 10.3390/sym12040667.

[27] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis," *Informatics*, vol. 8, no. 4, p. 79, Nov. 2021, doi: 10.3390/informatics8040079.

[28] I. K. Nti, O. Nyarko-Boateng, and J. Aning, "Performance of Machine Learning Algorithms with Different K Values in K-fold CrossValidation," *International Journal of Information Technology and Computer Science*, vol. 13, no. 6, pp. 61–71, Dec. 2021, doi: 10.5815/ijitcs.2021.06.05.

[29] A. K. Jailani and A. Erna, "Deteksi Anomali pada Rasio Jam Belajar dan Aktivitas Sosial terhadapPerforma Akademis Mahasiswa menggunakan Metode Local Outlier Factor (LOF)," *SISITI : Seminar Ilmiah Sistem Informasi dan Teknologi Informasi*, vol. 14, no. 1, pp. 79–88, Mar. 2025, doi: 10.36774/sisiti.v14i1.1678.

[30] S. Afzal, A. Afzal, M. Amin, S. Saleem, N. Ali, and M. Sajid, "A Novel Approach for Outlier Detection in Multivariate Data," *Math Probl Eng*, vol. 2021, pp. 1–12, Oct. 2021, doi: 10.1155/2021/1899225.

[31] M. Abdelhamid and A. Desai, "Balancing the Scales: A Comprehensive Study on Tackling Class Imbalance in Binary Classification," Sep. 2024, [Online]. Available: http://arxiv.org/abs/2409.19751

[32] C. Iwendi, S. Khan, J. H. Anajemba, M. Mittal, M. Alenezi, and M. Alazab, "The Use of Ensemble Models for Multiple Class and Binary Class Classification for Improving Intrusion Detection Systems," *Sensors*, vol. 20, no. 9, p. 2559, Apr. 2020, doi: 10.3390/s20092559.

[33] A. Ridwan, "Penerapan Algoritma Naïve Bayes Untuk Klasifikasi Penyakit Diabetes Mellitus," *Jurnal SISKOM-KB (Sistem Komputer dan Kecerdasan Buatan)*, vol. 4, no. 1, pp. 15–21, Oct. 2020, doi: 10.47970/siskom-kb.v4i1.169.

[34] K. Krasnodębska, W. Goch, J. A. Verstegen, J. H. Uhl, and M. Pesaresi, "Advancing Precision, Recall, F-Score, and Jaccard Index: An Approach for Continuous Gridded Data," 2024. doi: 10.2139/ssrn.4865121.

[35] G. M. Foody, "Challenges in the real world use of classification accuracy metrics: From recall and precision to the Matthews correlation coefficient," *PLoS One*, vol. 18, no. 10, p. e0291908, Oct. 2023, doi: 10.1371/journal.pone.0291908.

[36] A. A. Kurniawan, M. Mustikasari, and P. Korespondensi, "EVALUASI KINERJA MLLIB APACHE SPARK PADA KLASIFIKASI BERITA PALSU DALAM BAHASA INDONESIA," vol. 9, no. 3, 2022, doi: 10.25126/jtiik.202293538.

Jeremia Pinnywan Immanuel, saat ini sebagai mahasiswa program studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Tarumanagara.

Eunice Eugenia Karta, saat ini sebagai mahasiswa program studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Tarumanagara.

Rio Bun Dika, saat ini sebagai mahasiswa program studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Tarumanagara.