

# Laporan Praktikum Kontrol Cerdas

## Minggu Ke-2

Nama : Reza Gusti Eka Prasetya

NIM : 224308044

Kelas : TKA-6B

Akun Github (Tautan) : <https://github.com/RezaGusti00>

Student Lab Assistant : Mas Ali

### 1. Judul Percobaan : *Object Classification with Scikit-learn*

### 2. Tujuan Percobaan :

- Memahami dasar-dasar *Machine Learning* dalam sistem kendali.
- Mengimplementasikan model ML sederhana untuk klasifikasi objek.
- Menggunakan *Scikit-learn* untuk membuat model ML dasar.
- Mengintegrasikan model ML dengan *Computer Vision* untuk deteksi objek.
- Mengelola dataset dan melakukan pelatihan model sederhana.

### 3. Landasan Teori :

*Machine Learning (ML)* adalah cabang dari kecerdasan buatan (AI) dan ilmu computer yang berfokus pada penggunaan data dan algoritma untuk meniru cara manusia belajar, secara bertahap meningkatkan akurasi. (Eriana et al., n.d.). *Machine Learning* terdiri dari beberapa pendekatan utama, seperti *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*. Pada praktikum kali ini menggunakan algoritma K-Nearest Neighbors (KNN). KNN adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasi sebelumnya. Termasuk dalam *supervised learning*, dimana hasil *query instance* yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam KNN. Sementara, *software* yang digunakan dalam praktikum ini yaitu *python*, *Visual Studio Code* dan *OpenCV*.

*Python* adalah bahasa pemrograman dengan tujuan umum (*general*). Itu berarti *python* dapat digunakan untuk menulis kode pada segala kebutuhan pemrograman. *Python*

diciptakan oleh Guidovan Rossum di Belanda pada tahun 1990. *Python* menjadi Bahasa pemrograman populer yang digunakan di industry dan akademik karena mudah, ringkas, dan memiliki library yang luas seperti ‘NumPy’ dan ‘Pandas’ untuk analisis data (Cahyo & Kholis, 2019). Sementara itu, *Visual Studio Code* adalah *Software* yang sangat ringan, namun kuat *editor* kode sumbernya yang berjalan dari *desktop*. *Visual Studio Code* digunakan untuk pembuatan kode-kode program dibutuhkan sebuah aplikasi yang mumpuni. *Visual Studio Code* dapat digunakan untuk berbagai Bahasa pemrograman seperti Java Script, HTML, CSS, PHP, Python, C++, dan masih banyak lagi (Syarif et al., 2023). *OpenCV (Open Source Computer Vision Library)* adalah sebuah pustaka perangkat lunak khusus untuk computer vision, yang ditujukan untuk pengolahan citra dinamis secara *real-time*. *OpenCV* memiliki lebih dari 25000 algoritma yang dioptimalkan untuk mendeteksi, mengidentifikasi dan mengenali wajah (Mutasil et al., 2021).

#### 4. Analisis dan Diskusi :

##### A. Analisis

Pada Mata Kuliah Praktikum Kontrol Cerdas minggu kedua ini kami melakukan percobaan mendeteksi objek berwarna menggunakan kamera dengan model *Support Vector Machine (SVM)*. Percobaan ini berfokus pada pengenalan warna berdasarkan data RGB serta nama warnanya. Data warna yang diperoleh kemudian diolah menggunakan ‘*LabelEncoder*’ untuk mengkonversi label teks menjadi angka agar dapat digunakan dalam model SVM secara lebih optimal. Agar fitur RGB dapat memberikan hasil yang lebih akurat, dilakukan proses normalisasi menggunakan ‘*StandardScaler*’. Model SVM yang digunakan dilatih dengan kernel *Radial Basis Function (RBF)*, yang dipilih karena kemampuannya dalam menangani distribusi data yang tidak linear, seperti variasi warna dalam ruang RGB. Selanjutnya, data dibagi menjadi dua bagian, yaitu *training set* dan *testing set*, menggunakan fungsi ‘*train\_test\_split*’ untuk memastikan model dapat diuji dengan baik. Akurasi model kemudian diuji pada data uji (*testing set*), dan hasil evaluasi model akan ditampilkan melalui terminal di VSCode.

Pada tahap *real-time color detection*, program menggunakan *OpenCV* untuk mengakses kamera dan mengambil dua *Region of Interest (ROI)* pada layar, yaitu di sisi kiri dan sisi kanan. Untuk setiap ROI, program menghitung rata-rata warna yang

tertangkap kamera. Warna rata-rata ini kemudian dinormalisasi dengan *StandardScaler* yang sudah dilatih sebelumnya dan digunakan sebagai input ke model SVM untuk memprediksi warna. Hasil prediksi dibandingkan dengan warna sebenarnya dalam dataset untuk menghitung akurasi deteksi secara bertahap (*running accuracy*), yang dihitung berdasarkan 50 prediksi terakhir. Program juga menampilkan kotak (*bounding box*) di area pemantauan serta menuliskan nama warna yang terdeteksi di layar. Informasi warna dan akurasinya juga ditampilkan dalam terminal.

## B. Diskusi

Program deteksi warna *real-time* berbasis SVM ini membuktikan bahwa *machine learning* dapat digunakan untuk klasifikasi warna dengan cukup baik. Namun, keakuratan prediksi masih bergantung pada kualitas dataset dan faktor pencahayaan, yang dapat memengaruhi hasil deteksi. Normalisasi dengan *StandardScaler* sudah membantu stabilitas prediksi, tetapi metode lain seperti konversi ke ruang warna HSV bisa lebih efektif dalam kondisi pencahayaan yang berubah-ubah. Selain itu, meskipun SVM cukup efisien, model ini memiliki keterbatasan dalam skala data besar dan kecepatan inferensi. Alternatif seperti *Convolutional Neural Networks* (CNN) dapat digunakan untuk meningkatkan akurasi. Pengembangan lebih lanjut dapat mencakup fitur kalibrasi warna otomatis dan visualisasi histogram warna untuk analisis lebih mendalam, sehingga sistem menjadi lebih adaptif dan akurat dalam berbagai kondisi. Kemudian, untuk penggunaan ROI memungkinkan deteksi warna pada dua titik berbeda pada simultan, dan perhitungan *running accuracy* memberikan evaluasi secara lebih akurat dan dinamis, serta stabil.

## 5. Assignment :

Program ini melakukan serangkaian tugas untuk mendeteksi warna secara *real-time* menggunakan model *Support Vector Machine* (SVM). Proses diawali dengan membaca dataset warna dari file CSV menggunakan perintah `pd.read_csv(file_path)`, yang berisi nilai RGB dan nama warna dari setiap sampel. Data tersebut kemudian diproses dengan mengekstrak fitur RGB menggunakan `color_data`, sementara label warna diubah menjadi format numerik menggunakan `LabelEncoder`. Konversi ini diperlukan agar model SVM dapat memahami dan memproses data warna dengan lebih baik. Selanjutnya, untuk memastikan setiap fitur memiliki skala yang seimbang, nilai RGB dinormalisasi dengan

`StandardScaler`. Normalisasi ini membantu model bekerja lebih stabil dan meningkatkan akurasi prediksi. Setelah data siap, dataset dibagi menjadi dua bagian, yaitu *training set* dan *testing set*, menggunakan perintah `train_test_split`. Pembagian ini bertujuan agar model dapat belajar dari data pelatihan dan kemudian diuji pada data yang belum pernah dilihat sebelumnya untuk mengukur kinerjanya.

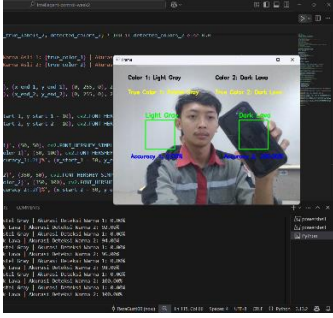
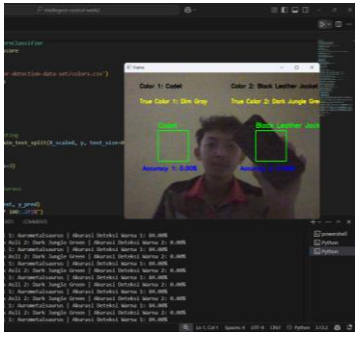

Setelah data dipersiapkan, model SVM dikonfigurasi dengan kernel *Radial Basis Function* (RBF) menggunakan `SVC`. Pemilihan kernel RBF dilakukan karena kemampuannya menangani distribusi data yang kompleks dan non-linear, seperti warna dalam ruang RGB. Model kemudian dilatih menggunakan *training set* dengan perintah `svm_model`, dan setelah pelatihan selesai, model diuji dengan *testing set* menggunakan `svm_model.predict(X_test)`. Akurasi prediksi dihitung menggunakan `accuracy_score` yang hasilnya ditampilkan di terminal untuk mengevaluasi seberapa baik model mengenali warna yang diberikan. Setelah model berhasil dilatih, program mengaktifkan kamera menggunakan OpenCV dengan perintah `cap = cv2.VideoCapture`. Untuk mendeteksi warna dalam gambar yang ditangkap, program menentukan dua *Region of Interest* (ROI). Kemudian, warna rata-rata dalam setiap ROI dihitung menggunakan `np.mean`, yang menghasilkan nilai RGB dari area yang diamati. Warna yang diperoleh dari ROI tersebut kemudian dinormalisasi kembali menggunakan `scaler.transform`, agar sesuai dengan model yang telah dilatih sebelumnya.

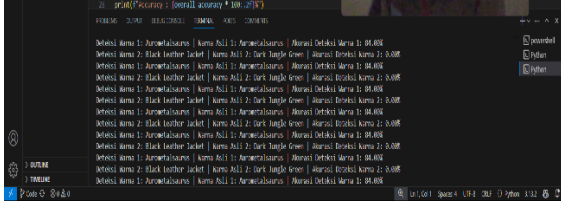
Hasil prediksi warna ditampilkan dalam bentuk visual pada layar, di mana program menggambar *bounding box* di sekitar ROI menggunakan perintah `cv2.rectangle`. Selain itu, teks yang menunjukkan warna yang terdeteksi juga ditampilkan dengan `cv2.putText`, sehingga pengguna dapat dengan mudah melihat warna yang dikenali oleh sistem. Informasi mengenai prediksi warna dan tingkat akurasi juga dicetak di terminal untuk memantau kinerja model secara langsung.

Pada percobaan kali ini kita melakukan 2 kondisi yang berbeda, yaitu dengan keadaan lampu menyala dan lampu mati. Percobaan menunjukkan bahwa akurasi model SVM dalam mendeteksi warna sangat dipengaruhi oleh kondisi pencahayaan. Pada keadaan lampu menyala, model dapat mendeteksi warna dengan lebih jelas, meskipun tidak sempurna untuk semua objek. Namun, ketika pencahayaan berkurang (lampu mati), hasil deteksi warna menjadi kurang akurat, bahkan warna yang ditampilkan pada *bounding box*

berbeda dari percobaan pertama. Hal ini menegaskan bahwa pencahayaan merupakan faktor yang penting dalam sistem klasifikasi warna berbasis kamera.

## 6. Data dan Output Hasil Pengamatan :

No.	Variabel	Kondisi	Hasil Pengamatan	Keterangan
1.	Akurasi Model SVM	Didalam Ruangan Lampu Menyala		Mendeteksi warna dengan jelas pada objek (benda) 2, meskipun pada objek 1, warna tidak terdeteksi secara 100%.
		Didalam Ruangan Lampu Mati		Pada objek (benda) 1 maupun 2, warna tidak terdeteksi secara jelas, dan hasil warna yang muncul berbeda dengan hasil warna pada percobaan pertama.
2.	Running Accuracy Real-Time (Pada Terminal)	Accuracy didalam Ruangan Lampu Menyala		

		Accuracy didalam Ruangan Lampu Mati	
--	--	---	--

## 7. Kesimpulan :

Berdasarkan praktikum dan analisis yang telah dilakukan, maka dapat diambil kesimpulan yaitu:

- *Machine Learning*, khususnya algoritma *Support Vector Machine (SVM)*, efektif dalam melakukan klasifikasi warna secara real-time, dengan kapasitas untuk mengenali dan mendeteksi warna dari data RGB yang diperoleh melalui kamera.
- Keakuratan prediksi model sangat bergantung pada kualitas dataset yang digunakan dan faktor eksternal seperti pencahayaan yang dapat mempengaruhi hasil deteksi warna.
- Normalisasi dengan *StandardScaler* berkontribusi pada stabilitas prediksi.
- Pendekatan yang menggunakan dua *Region of Interest (ROI)* untuk deteksi warna menunjukkan fleksibilitas sistem dalam memantau dan menganalisis dua area secara bersamaan.
- Keakuratan deteksi warna pada benda juga dipengaruhi oleh faktor kamera pada setiap laptop/PC yang digunakan.

## 8. Saran :

Penting untuk memperluas dataset yang digunakan dengan menambahkan variasi warna dan kondisi pencahayaan yang berbeda. Hal ini akan membantu model belajar dari berbagai contoh, sehingga menjadi lebih handal dalam mengenali warna baru yang belum pernah dilihat sebelumnya. Meskipun SVM cukup efektif dalam klasifikasi warna, penggunaan model yang lebih kompleks seperti *Convolutional Neural Networks (CNN)* dapat memberikan hasil yang lebih baik, terutama untuk skala data yang lebih besar.

## 9. Daftar Pustaka :

- Cahyo, O. B. D., & Kholis, N. (2019). RANCANG BANGUN SIMULATOR ELEKTROKARDIOGRAM MENGGUNAKAN FPGA YANG TERINTEGRASI DENGAN SOFTWARE PYTHON. 08.
- Eriana, E. S., Kom, S., Kom, M., Zein, D. A., & Kom, M. (n.d.). ARTIFICIAL INTELLIGENCE (AI).
- Mutasil, A., Irsan, M., & Sujana, D. (2021). Pengenalan Wajah Menggunakan Opencv Untuk Validasi Peserta Ujian Penerimaan Mahasiswa Baru. Jurnal SISKOM-KB (Sistem Komputer dan Kecerdasan Buatan), 5(1), 21–28. <https://doi.org/10.47970/siskom-kb.v5i1.221>
- Syarif, M. N., Pambudiyatno, N., & Utomo, W. (2023). RANCANGAN SISTEM PRESENSI DAN REKAPITULASI JURNAL KEGIATAN OJT MENGGUNAKAN VISUAL STUDIO CODE BERBASIS WEB DI AIRNAV CABANG MATSC.