Klasifikasi Model Mobil Bekas Berdasarkan Citra Menggunakan Arsitektur ResNet34

Disusun Oleh:

Fattan Raditya Anggoro 2206812533

DEPARTEMEN MATEMATIKA

UNIVERSITAS INDONESIA

2025

Contents

1 Pendahuluan

Pada satu dekade terakhir ini, perkembangan teknologi Artificial Intelligence (AI), khususnya dalam bidang computer vision, telah membuka banyak peluang baru dalam otomatisasi pengolahan data citra. Salah satu pendekatan yang paling menonjol dalam bidang ini adalah penggunaan Convolutional Neural Network (CNN), yang telah terbukti sangat efektif dalam mengenali dan mengklasifikasikan objek dalam citra. Teknologi telah digunakan secara masif dalam berbagai industri, mulai dari kesehatan, pertanian, keamanan, hingga otomotif.

Dalam industri otomotif, identifikasi kendaraan berdasarkan citra menjadi salah satu kebutuhan yang penting, misalnya dalam konteks jual-beli mobil bekas secara daring. Sejumlah platform jual beli kendaraan otomotif mengandalkan input manual dari pengguna untuk menentukan merek dan model mobil, yang sering kali menyebabkan kesalahan input atau inkonsistensi data. Oleh karena itu, dibutuhkan sebuah sistem yang dapat secara otomatis mengidentifikasi model mobil berdasarkan input citra, guna meningkatkan keakuratan dan efisiensi proses pencatatan dan pencarian informasi, hingga meningkatkan efektivitas dan pengalaman pengguna.

Penelitian ini dilatarbelakangi oleh kebutuhan untuk mengembangkan sistem klasifikasi model mobil berbasis citra menggunakan pendekatan deep learning. Dengan memanfaatkan arsitektur ResNet34 yang telah terbukti handal dalam tugas-tugas klasifikasi citra, penelitian ini mencoba membangun sebuah model prediktif yang mampu mengenali berbagai jenis dan model mobil dari dataset citra yang tersedia. Penggunaan teknik transfer learning memungkinkan pemanfaatan pengetahuan dari model yang telah dilatih sebelumnya, sehingga proses pelatihan menjadi lebih cepat dan efisien, terutama ketika berhadapan dengan dataset yang terbatas.

Melalui penelitian ini, diharapkan dapat dihasilkan model machine learning yang tidak hanya memiliki akurasi tinggi, tetapi juga dapat diimplementasikan secara praktis dalam sistem berbasis web atau aplikasi mobile untuk mendukung transformasi digital dalam industri otomotif.

2 Metodologi Penelitian

2.1 Deep Learning

Deep learning adalah pendekatan lanjutan dari machine learning yang memanfaatkan arsitektur jaringan saraf dengan banyak lapisan untuk mempelajari pola data yang kompleks dan berskala besar. Metode ini bekerja dengan membagi proses pembelajaran menjadi beberapa tahap bertingkat, di mana setiap lapisan dalam jaringan bertugas mengekstraksi fitur dari data secara bertahap, mulai dari representasi sederhana hingga abstraksi yang lebih tinggi [Prasetyo, 2024].

Deep learning terinspirasi dari cara kerja otak manusia dalam memproses informasi, sehingga mampu melakukan pembelajaran secara end-to-end tanpa perlu intervensi manusia untuk mengekstraksi fitur secara manual. Teknologi ini telah merevolusi banyak bidang seperti pengenalan wajah, pemrosesan bahasa alami, deteksi objek, dan kendaraan otonom, karena kemampuannya yang tinggi dalam menangani data tak terstruktur seperti gambar, suara, dan teks.

2.2 Artificial Neural Network

Artificial Neural Network (ANN) merupakan pendekatan komputasional yang terinspirasi dari struktur biologis otak manusia, di mana informasi diproses oleh jaringan neuron buatan yang saling terhubung. ANN dibangun dari sejumlah unit sederhana yang disebut neuron, yang diorganisasikan dalam lapisan-lapisan – input, tersembunyi (hidden), dan output – dan masing-masing neuron berfungsi untuk mengolah dan mentransformasikan sinyal yang diterimanya.

Melalui proses pelatihan berbasis data, ANN dapat belajar untuk mengenali pola, melakukan prediksi, atau mengklasifikasikan data. Pembelajaran ini dilakukan dengan menyesuaikan bobot koneksi antar neuron secara berulang, menggunakan algoritma seperti backpropagation, untuk memperkecil selisih antara hasil prediksi dan target yang sebenarnya. Dengan kemampuan adaptif ini, ANN menjadi fondasi penting dalam berbagai aplikasi kecerdasan buatan, termasuk pengenalan suara, sistem rekomendasi, dan klasifikasi teks.

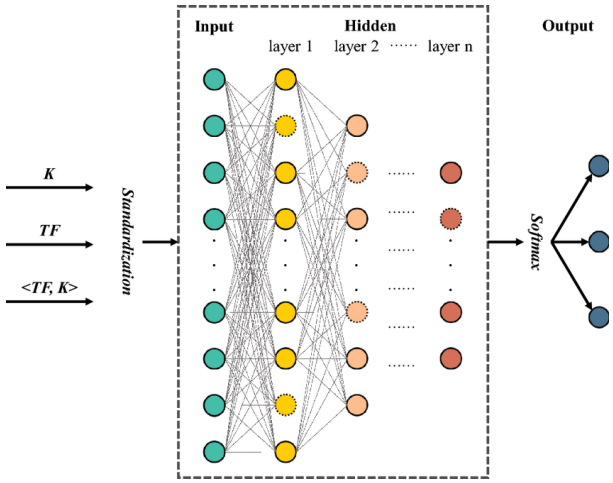


Figure 1: Arsitektur *Artificial Neural Network*
[Shi et al., 2022]

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network Convolutional Neural Network (CNN) adalah salah satu jenis arsitektur dalam deep learning yang dirancang khusus untuk mengolah data dalam bentuk grid, seperti citra digital. CNN bekerja dengan mengekstraksi fitur-fitur penting dari gambar melalui proses konvolusi, di mana filter atau kernel menggeser dan memindai bagian-bagian gambar untuk menangkap pola visual. Melalui proses bertingkat ini, CNN mampu mengenali objek atau karakteristik spesifik secara otomatis tanpa memerlukan ekstraksi fitur manual.

CNN memiliki keunggulan dalam memahami struktur spasial dan hubungan lokal antar piksel dalam gambar, menjadikannya sangat efektif untuk tugas-tugas pengenalan wajah, klasifikasi gambar, deteksi objek, dan segmentasi citra. Setiap lapisan dalam CNN memiliki fungsi tertentu, mulai dari mendeteksi tepi hingga menangkap bentuk kompleks, yang kemudian digabungkan untuk membentuk representasi akhir dari data input.

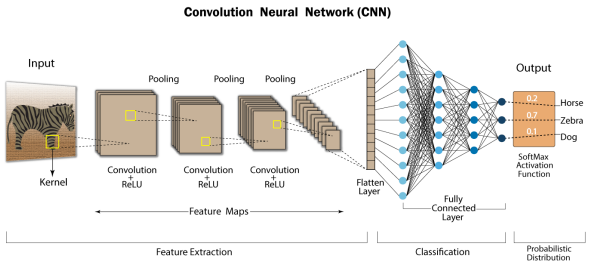


Figure 2: Arsitektur *Convolutional Neural Network* [].

2.4 Arsitektur Dasar ResNet34

ResNet34 merupakan salah satu varian dari keluarga *Residual Network* (ResNet) yang diperkenalkan oleh He et al. pada tahun 2015 dalam makalah berjudul "Deep Residual Learning for Image Recognition". ResNet dirancang untuk mengatasi permasalahan degradasi akurasi pada jaringan dalam (deep network), di mana penambahan lapisan justru menyebabkan penurunan performa akibat hilangnya informasi selama propagasi.

Arsitektur ResNet34 memiliki total 34 lapisan dengan parameter yang dapat dilatih. Struktur utama dari ResNet34 terdiri atas kombinasi beberapa jenis lapisan: *convolutional layers*, *batch normalization*, fungsi aktivasi ReLU, dan *residual blocks* yang mengimplementasikan *skip connection*.

2.4.1 Struktur Lapisan ResNet34

Secara umum, arsitektur ResNet34 dibangun sebagai berikut:

- **Input Layer:** Citra masukan berukuran 224×224×3.
- **Convolution Layer 1:** Lapisan konvolusi awal dengan 64 filter berukuran 7×7, stride 2, diikuti oleh *batch normalization*, fungsi aktivasi ReLU, dan *max pooling* 3×3 dengan stride 2.
- **Convolution Layer 2:** Terdiri dari 3 *residual blocks*, masing-masing dengan dua lapisan konvolusi 3×3, 64 filter.
- **Convolution Layer 3:** Terdiri dari 4 *residual blocks*, masing-masing dengan dua lapisan konvolusi 3×3, 128 filter. Dimulai dengan downsampling (stride 2).
- **Convolution Layer 4:** Terdiri dari 6 *residual blocks*, masing-masing dengan dua lapisan konvolusi 3×3, 256 filter. Dimulai dengan downsampling.
- **Convolution Layer 5:** Terdiri dari 3 *residual blocks*, masing-masing dengan dua lapisan konvolusi 3×3, 512 filter. Dimulai dengan downsampling.
- **Pooling dan Output:** Lapisan *average pooling* global, diikuti oleh *fully connected layer* (dense) dengan jumlah neuron sesuai jumlah kelas.

2.4.2 Residual Learning

Ciri khas utama ResNet adalah penggunaan **residual block** yang memiliki *shortcut connection* atau *skip connection*. Mekanisme ini memungkinkan sinyal untuk melewati

satu atau lebih lapisan dan langsung dijumlahkan dengan output hasil proses non-linear. Jika x adalah input dan $F(x)$ adalah hasil dari transformasi non-linear, maka output dari residual block adalah:

$$y = F(x) + x$$

Dengan cara ini, model lebih mudah mempelajari fungsi identitas dan mempercepat konvergensi saat pelatihan, serta memitigasi masalah *vanishing gradient*.

2.4.3 Kelebihan ResNet34

- Memungkinkan pelatihan jaringan dalam tanpa kehilangan akurasi.
- Mengurangi risiko overfitting berkat penggunaan batch normalization dan regularisasi implicit dari skip connection.
- Lebih stabil saat pelatihan dan lebih cepat konvergen dibanding jaringan konvolusional standar dengan kedalaman serupa.

Dalam konteks penelitian ini, arsitektur ResNet34 dimanfaatkan sebagai *feature extractor* untuk klasifikasi citra mobil bekas. Lapisan akhir dari model dimodifikasi agar sesuai dengan jumlah kelas dalam dataset, yaitu 13 kelas.

2.5 Fungsi Aktivasi

Fungsi aktivasi yakni fungsi dalam neuron buatan yang memberikan output berdasarkan input. Fungsi aktivasi dalam neuron buatan yakni bagian penting dari peran yang dimainkan neuron buatan dalam jaringan saraf buatan modern [9]. Fungsi aktivasi memperkenalkan non-linearitas ke dalam jaringan saraf, memungkinkan model untuk mempelajari dan merepresentasikan hubungan kompleks dalam data yang tidak dapat ditangkap oleh model linear[10].

2.5.1 Fungsi ReLU

ReLU (Rectified Linear Unit) adalah salah satu fungsi aktivasi yang paling populer dalam jaringan saraf tiruan (neural networks), terutama dalam bidang Deep Learning. Fungsi ini bersifat non-linear dan sederhana, namun efektif dalam mempercepat proses pelatihan jaringan saraf tiruan. ReLU bekerja dengan mengubah semua nilai negatif menjadi nol dan mempertahankan nilai positif sebagaimana adanya.

Secara matematis, fungsi ReLU didefinisikan sebagai berikut:

$$f(x) = \begin{cases} 0 & \text{jika } x \leq 0 \\ x & \text{jika } x > 0 \end{cases} \tag{1}$$

ReLU menghasilkan aktivasi yang jarang (sparse activation), yang dapat meningkatkan efisiensi komputasi dan kemampuan generalisasi model.

2.5.2 Fungsi *Sigmoid*

Fungsi aktivasi sigmoid adalah salah satu fungsi non-linear yang umum digunakan dalam jaringan saraf tiruan, terutama untuk tugas klasifikasi biner. Fungsi ini mengubah input numerik menjadi output dalam rentang 0 hingga 1, sehingga cocok untuk merepresentasikan probabilitas. Secara matematis, fungsi sigmoid didefinisikan sebagai:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

di mana x adalah input numerik dan e merupakan bilangan Euler.

2.6 *Loss Function*

Loss function adalah fungsi matematis yang digunakan untuk mengukur seberapa baik model memprediksi output yang diinginkan. Fungsi ini menghitung selisih antara nilai prediksi model dan nilai sebenarnya (label), dan hasilnya digunakan untuk memperbarui bobot model selama proses pelatihan.

2.6.1 *Binary Cross-Entropy* (BCE)

Digunakan untuk masalah klasifikasi biner, di mana terdapat dua kelas (misalnya, pria dan wanita). BCE mengukur perbedaan antara label sebenarnya dan probabilitas prediksi model.

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)] \tag{3}$$

- N : Jumlah total sampel (data latih)
- y_i : Label sebenarnya untuk sampel ke-i, bernilai 0 atau 1
- p_i : Probabilitas prediksi dari model bahwa sampel ke-i termasuk dalam kelas 1
- log : Fungsi logaritma natural

2.7 *Adaptive Moment Estimation* (Adam)

Adaptive Moment Estimation(Adam) adalah algoritma optimisasi yang dirancang untuk pelatihan jaringan saraf dalam pembelajaran mendalam. Dengan mengadaptasi laju pembelajaran untuk setiap parameter secara individual, Adam memungkinkan konvergensi yang lebih cepat dan stabil dibandingkan dengan algoritma optimisasi lainnya .

1. **Untuk setiap iterasi t lakukan:**

   (a) Hitung gradien dari fungsi kerugian terhadap parameter:

   $$g_t = \nabla_\theta J(\theta_t) \tag{4}$$

   (b) Perbarui momen pertama (rata-rata gradien):

   $$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{5}$$

(c) Perbarui momen kedua (rata-rata kuadrat gradien):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{6}$$

(d) Koreksi bias pada momen pertama:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{7}$$

(e) Koreksi bias pada momen kedua:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{8}$$

(f) Perbarui parameter:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{9}$$

Keterangan

- θ_t: Parameter model pada iterasi ke-t
- g_t: Gradien dari fungsi kerugian terhadap parameter pada iterasi ke-t
- m_t: Estimasi momen pertama (rata-rata gradien) pada iterasi ke-t
- v_t: Estimasi momen kedua (rata-rata kuadrat gradien) pada iterasi ke-t
- $\hat{m}_t$: Estimasi momen pertama yang telah dikoreksi bias
- $\hat{v}_t$: Estimasi momen kedua yang telah dikoreksi bias
- α: Laju pembelajaran (learning rate)
- β_1: Koefisien eksponensial untuk momen pertama (biasanya 0.9)
- β_2: Koefisien eksponensial untuk momen kedua (biasanya 0.999)
- ϵ: Nilai kecil untuk mencegah pembagian dengan nol (biasanya 10^{-8})

2.8 Evaluasi Model

2.8.1 *Accuracy*

Accuracy (Akurasi) adalah metrik evaluasi yang mengukur seberapa baik model membuat prediksi yang benar dari total prediksi yang dilakukan. Dalam konteks klasifikasi, akurasi memberikan gambaran mengenai seberapa sering model memprediksi kelas yang benar, baik itu kelas positif maupun negatif. Secara umum, bentuk matematis dari akurasi adalah :

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

2.8.2 F1-Score

F1-Score memberikan gambaran mengenai seberapa baik model kita dalam mengklasifikasikan baik *review* positif maupun negatif secara akurat. Secara umum, bentuk matematis dari *F1-Score* ialah:

$$F1\text{-}Score = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$\text{Precision} = \frac{\text{jumlah prediksi benar (positif)}}{\text{prediksi benar (positif)} + \text{prediksi salah (positif)}} \tag{13}$$

2.8.3 Confusion Matrix

Confusion Matrix merupakan metode evaluasi yang digunakan untuk mengukur kinerja algoritma klasifikasi. Tabel ini menampilkan kombinasi antara hasil prediksi dan nilai aktual dari data, dan membantu dalam memahami sejauh mana model berhasil atau gagal dalam melakukan klasifikasi.

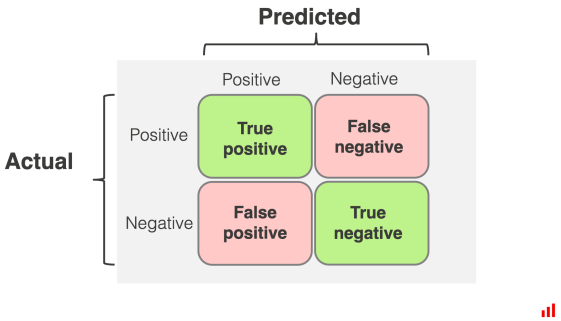


Figure 3: Ilustrasi Confusion Matrix

Keterangan:

- **True Positive (TP)**: Data aktual positif dan diprediksi juga sebagai positif.
  Contoh: Gambar laki-laki yang benar-benar diklasifikasikan sebagai laki-laki.
- **False Negative (FN)**: Data aktual positif tetapi diprediksi sebagai negatif.
  Contoh: Gambar laki-laki yang diklasifikasikan sebagai perempuan.
- **False Positive (FP)**: Data aktual negatif tetapi diprediksi sebagai positif.
  Contoh: Gambar perempuan yang diklasifikasikan sebagai laki-laki.
- **True Negative (TN)**: Data aktual negatif dan diprediksi juga sebagai negatif.
  Contoh: Gambar perempuan yang benar-benar diklasifikasikan sebagai perempuan.

Confusion Matrix juga dapat dibentuk dalam bentuk 13x13 sesuai dengan prediksi kelas oleh model dan nilai aktual sesuai dengan kelas yang tersedia. Elemen pada diagonal matriks menunjukkan prediksi yang bernilai benar.

3 Implementasi

3.1 Deskripsi Dataset

Dataset yang digunakan dalam penelitian ini adalah Iran Used Car Dataset dari Kaggle. Dataset ini berisi citra mobil-mobil bekas dan dikelompokkan ke dalam tipe mobil masing-masing. Dataset juga sudah dibagi untuk proses *training, testing* dan *validation* dengan rasio 70:15:15. Dataset ini berisi total 9737 citra dengan format .jpg dengan jumlah piksel yang bervariasi dan kemudian dibagi menjadi 13 kelas.

Dataset ini dipilih karena memiliki keragaman visual yang cukup tinggi, yang mencerminkan kondisi nyata di dunia otomotif bekas. Hal ini menjadikannya sangat cocok untuk menguji performa model klasifikasi berbasis deep learning dalam mengenali objek dengan variasi tinggi pada domain gambar dunia nyata.

3.2 Lingkungan Pengembangan

Implementasi sistem dilakukan dalam lingkungan pemrograman Python dengan menggunakan pustaka TensorFlow dan Keras untuk membangun dan melatih model deep learning. Proses pelatihan model dilakukan di Google Colab yang menyediakan akses ke GPU (Graphics Processing Unit) secara gratis, memungkinkan proses komputasi berjalan lebih cepat dibandingkan dengan CPU biasa. Selain itu, digunakan juga pustaka pendukung seperti NumPy dan Pandas untuk manipulasi data, Matplotlib dan Seaborn untuk visualisasi, serta Scikit-learn untuk evaluasi performa model.

3.3 *Preprocessing* Data

Sebelum digunakan dalam proses pelatihan model, data gambar mobil bekas yang terdiri dari 9.737 citra dan terbagi ke dalam 13 kelas, terlebih dahulu melalui proses prapemrosesan. Proses ini dimulai dengan menyusun gambar ke dalam direktori berdasarkan label kelas masing-masing, sehingga memudahkan pemuatan data menggunakan modul `ImageFolder` dari pustaka PyTorch.

Selanjutnya, dilakukan serangkaian transformasi pada data untuk menyesuaikan format dan meningkatkan kualitas data latih. Gambar-gambar diubah ukurannya (resize) dan dilakukan pemotongan acak (random crop) untuk mendapatkan dimensi yang konsisten, yaitu 224×224 piksel, sesuai dengan dimensi input standar dari model ResNet34. Selain itu, dilakukan augmentasi data melalui transformasi seperti rotasi acak, flipping horizontal, dan penyesuaian kecerahan dan kontras, yang bertujuan untuk meningkatkan variasi data dan mencegah overfitting.

Setelah transformasi tersebut, gambar dikonversi menjadi tensor dan dinormalisasi menggunakan nilai rata-rata dan standar deviasi dari dataset ImageNet, yaitu:

```
mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]
```

Normalisasi ini dilakukan agar distribusi piksel gambar memiliki kesesuaian dengan data yang digunakan dalam pelatihan awal model ResNet34, mengingat model yang digunakan dalam penelitian ini adalah *SKResNet34*, yaitu versi modifikasi dari arsitektur ResNet34 yang telah dilatih sebelumnya (pretrained) pada dataset ImageNet.

Setelah proses transformasi selesai, data dimuat dalam batch menggunakan modul `DataLoader` dengan proses pengacakan (shuffling) untuk menjamin variasi selama pelatihan. Seluruh tahapan pra-pemrosesan ini dilakukan untuk memastikan bahwa data berada dalam format yang optimal bagi model deep learning dan untuk meningkatkan kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya.

3.4 Arsitektur Model ResNet34

ResNet34 adalah salah satu varian dari arsitektur Residual Network (ResNet) yang diperkenalkan oleh He et al. pada tahun 2015. ResNet mengatasi masalah degradasi akurasi pada jaringan yang sangat dalam dengan memperkenalkan mekanisme *residual learning*, yaitu dengan menggunakan shortcut connection atau *skip connection* yang memungkinkan gradien mengalir secara langsung tanpa mengalami peredaman.

Arsitektur ResNet34 terdiri dari 34 lapisan yang mencakup konvolusi, batch normalization, fungsi aktivasi ReLU, dan pooling. Lapisan utama dalam ResNet34 dibangun dari blok residual, yang terdiri dari dua atau tiga lapisan konvolusi berturut-turut, dengan shortcut connection yang menambahkan input blok langsung ke output sebelum fungsi aktivasi terakhir. Pendekatan ini membantu model belajar identitas fungsi dengan lebih mudah, mempercepat konvergensi, dan meningkatkan akurasi pelatihan untuk jaringan yang dalam.

Struktur ResNet34 secara umum terdiri dari:

- Satu lapisan awal konvolusi dengan 64 filter berukuran 7×7 dan stride 2, diikuti oleh *max pooling*.
- Empat grup blok residual yang masing-masing terdiri dari beberapa blok: (3, 4, 6, 3) blok per grup, dengan peningkatan jumlah filter berturut-turut (64, 128, 256, dan 512).
- Satu lapisan *average pooling* global dan satu lapisan *fully connected* (dense) di akhir sebagai output.

ResNet34 juga dilengkapi dengan regularisasi implicit melalui *batch normalization* dan penggunaan ReLU sebagai fungsi aktivasi untuk memperkenalkan non-linearitas. Dalam penelitian ini, model ResNet34 dimodifikasi pada bagian output untuk menyesuaikan jumlah neuron dengan jumlah kelas dalam dataset, yaitu 13 kelas, sehingga model dapat melakukan klasifikasi sesuai kebutuhan.

3.5 Proses *Training*

Model dilatih selama 3 epoch menggunakan pendekatan *transfer learning* dengan arsitektur SKResNet34 yang telah dilatih sebelumnya pada dataset ImageNet. Optimizer yang digunakan adalah Adam dengan nilai *learning rate* sebesar 0,001. Proses pelatihan dilakukan menggunakan `train_loader`, sedangkan proses validasi dilakukan menggunakan `val_loader`.

Selama proses pelatihan, akurasi dan nilai kehilangan (*loss*) dari data pelatihan dan validasi dipantau setiap *epoch*. Jika akurasi validasi menunjukkan peningkatan, maka bobot model terbaik akan disimpan. Mekanisme seperti ini bertujuan untuk menghindari overfitting dan memastikan performa terbaik dari model pada data yang belum pernah dilihat. Selain itu, seluruh proses pelatihan memanfaatkan `GPU acceleration` untuk mempercepat komputasi.

3.6 Evaluasi Model

Evaluasi model dilakukan dengan mengamati nilai akurasi dan loss pada data validasi. Selain itu, dilakukan prediksi pada data uji untuk menghitung metrik tambahan seperti precision, recall, dan F1-score, serta menghasilkan confusion matrix untuk melihat distribusi prediksi model.

4 Kesimpulan

Penelitian ini bertujuan untuk membangun model klasifikasi gender berbasis gambar wajah menggunakan pendekatan *Convolutional Neural Network* (CNN). Dataset yang digunakan diperoleh dari Kaggle, dengan proses yang mencakup *preprocessing* data, desain arsitektur CNN, pelatihan model, dan evaluasi performa.

Berdasarkan hasil implementasi, diperoleh kesimpulan sebagai berikut:

- SkResNet34 terbukti efektif dalam membedakan citra tipe mobil, dengan akurasi yang tinggi pada data validasi. Model mampu mengekstraksi fitur visual penting yang menjadi penentu dalam klasifikasi tipe mobil.
- Model mencapai akurasi sebesar 75%, dengan nilai *precision* dan *recall* masing-masing sebesar 0,75–0,78 untuk kedua kelas. Ini menunjukkan performa model yang seimbang dan andal dalam mengklasifikasikan ke-tiga belas tipe mobil. Perlu diingat ini merupakan versi iterasi yang hanya menjalankan 3 epoch mengingat keterbatasan kemampuan prosesor.
- Evaluasi menggunakan metrik *F1-score* menunjukkan nilai yang beragam, diatas 0,75 untuk setiap kelas, kecuali Peugeot-207i, Peykan, Pride-13, dan Renault-L90 serta nilai *macro* dan *weighted average F1-score* sebesar 0,75, yang mengindikasikan kestabilan performa model terhadap distribusi data. Hal ini juga berarti model masih menunjukkan kekurangan dalam memprediksi beberapa tipe mobil dalam iterasi ini.
- Penggunaan *optimizer* Adam membantu mempercepat proses konvergensi dan menjaga kestabilan penurunan nilai *loss* selama pelatihan.
- Untuk penelitian selanjutnya, lebih baik project ini di-*run* dalam jumlah *epochs* yang lebih besar demi hasil yang lebih akurat dalam prediksi kelas menggunakan kapasitas GPU Processor yang lebih baik.

5 Source Code

Berikut potongan kode dari proyek klasifikasi model mobil bekas berdasarkan citra menggunakan arsitektur ResNet34:

5.1 Persiapan Dataset dan *Environment*

Figure 4: Persiapan Dataset

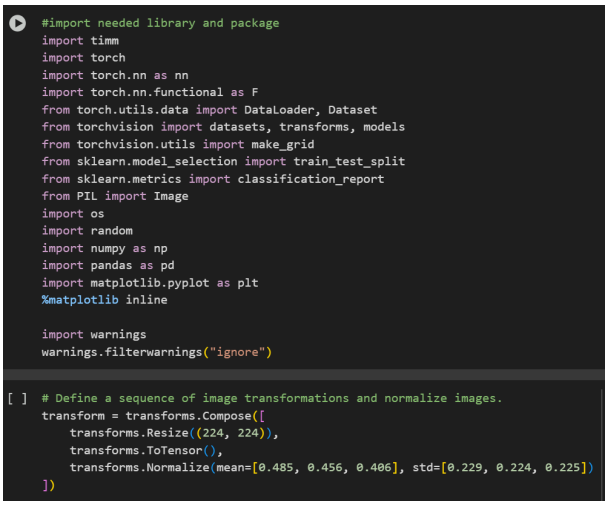
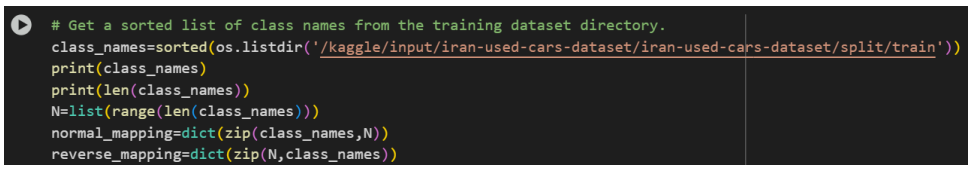


Figure 5: Persiapan Library dan Normalisasi Citra

Potongan-potongan kode ini digunakan untuk mengunduh dataset dari sumber menggunakan Kagglehub dan mempersiapkan library yang dibutuhkan. Karena citra yang diperoleh dari sumber belum optimal, maka dilakukan normalisasi juga.

5.2 Set Dataset dan Dataloader



```
# Create a list to store paths and labels for the training set.
path_label=[]
for dirname, _, filenames in os.walk('/kaggle/input/iran-used-cars-dataset/iran-used-cars-dataset/split/train'):
    for filename in filenames:
        if filename[-4:]=='.jpg':
            path=os.path.join(dirname, filename)
            label=dirname.split('/')[-1]
            path_label+=[(path,normal_mapping[label])]

# Create a list to store paths and labels for the validation set.
vpath_label=[]
for dirname, _, filenames in os.walk('/kaggle/input/iran-used-cars-dataset/iran-used-cars-dataset/split/val'):
    for filename in filenames:
        if filename[-4:]=='.jpg':
            path=os.path.join(dirname, filename)
            label=dirname.split('/')[-1]
            vpath_label+=[(path,normal_mapping[label])]

# Create a list to store paths and labels for the test set.
tpath_label=[]
for dirname, _, filenames in os.walk('/kaggle/input/iran-used-cars-dataset/iran-used-cars-dataset/split/test'):
    for filename in filenames:
        if filename[-4:]=='.jpg':#check if the file is JPG
            path=os.path.join(dirname, filename)
            label=dirname.split('/')[-1]
            tpath_label+=[(path,normal_mapping[label])]
```

```
# Define a custom Dataset class for loading images.
class ImageDataset(Dataset):
    def __init__(self, path_label, transform=None):
        self.path_label = path_label # Initialize the dataset with a list of image paths and labels
        self.transform = transform # Initialize the transformation to be applied to images.

    def __len__(self):
        return len(self.path_label)

    def __getitem__(self, idx):
        path, label = self.path_label[idx]  # Get the image path and label for a given index.
        img = Image.open(path).convert('RGB') # Open the image and convert it to RGB format.

        if self.transform is not None:  # Apply the transformation if it is provided.
            img = self.transform(img)

        return img, label # Return the transformed image and its label.
```

```
[ ] # Create instances of the ImageDataset for training, validation, and testing.
    trainset = ImageDataset(path_label, transform)
    valset = ImageDataset(vpath_label, transform)
    testset = ImageDataset(tpath_label, transform)

    # Extract the labels from the training set.
    labels = [label for _, label in trainset.path_label]

    # Create DataLoader instances for the training, validation, and test sets.
    train_loader=DataLoader(trainset,batch_size=32,shuffle=True)
    val_loader=DataLoader(valset,batch_size=32)
    test_loader=DataLoader(testset,batch_size=32)
```

```
[ ] for images, labels in train_loader: # Get a batch of images and labels from the training loader.
        break
    im=make_grid(images,nrow=16)
```

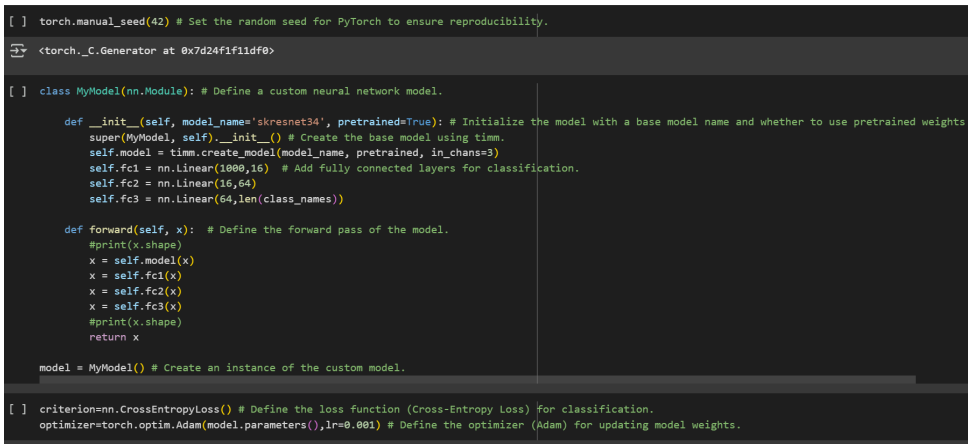
```
# Display the image grid
plt.figure(figsize=(12,12))
plt.imshow(np.transpose(im.numpy(),(1,2,0)))
```

Figure 6: Proses Dataloader dan *Inverse Normalization*

Kode ini mempersiapkan data citra untuk pelatihan model yang meliputi:

- Mengidentifikasi kelas mobil bekas dari nama folder.
- Membuat daftar jalur gambar dan label numerik untuk data pelatihan, validasi, dan pengujian.
- Mendefinisikan cara memuat dan mengubah gambar dengan kelas ImageDataset kustom.
- Menggunakan DataLoader untuk memuat data dalam batch secara efisien selama pelatihan, validasi, dan pengujian.

5.3 Arsitektur SkResNet34



```
epochs=3

import time
start_time=time.time()
train_losses=[]
test_losses=[]
train_correct=[]
test_correct=[]

for i in range(epochs): # Start the training loop.
    trn_corr=0
    tst_corr=0
    for b, (X_train,y_train) in enumerate(train_loader): # Iterate through the training data loader.
        b+=1
        y_pred=model(X_train)    # Get predictions from the model.
        loss=criterion(y_pred,y_train)
        predicted=torch.max(y_pred.data,1)[1] # Get the predicted class
        batch_corr=(predicted==y_train).sum()   # Calculate the number of correct predictions in the batch.
        trn_corr+= batch_corr
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if b%200==0:
            print(f'epoch: {i:2}  batch: {b:4} [{10*b:6}/8000]  loss: {loss.item():10.8f}  \
accuracy: {trn_corr.item()*100/(10*b):7.3f}%')

    loss=loss.detach().numpy()
    train_losses.append(loss)
    train_correct.append(trn_corr)
```

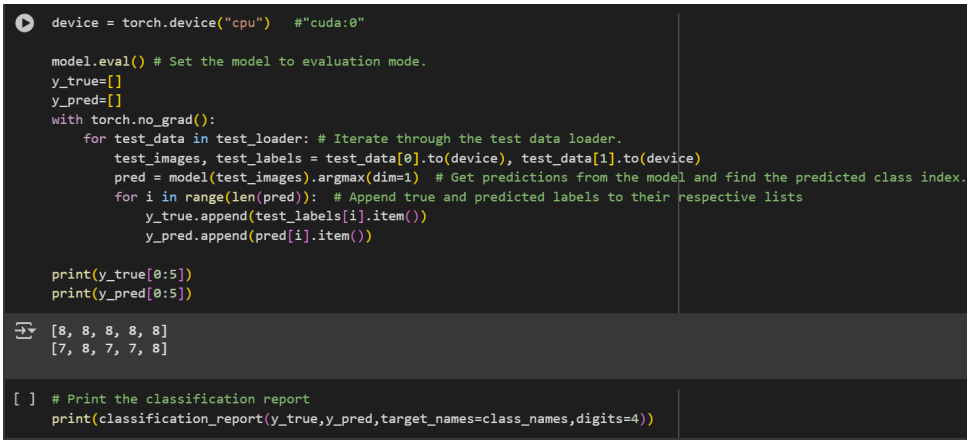
```
    with torch.no_grad(): # Start evaluation phase
        for b, (X_test,y_test) in enumerate(val_loader):
            b+=1
            y_val=model(X_test) # Get predictions from the model.
            predicted=torch.max(y_val.data,1)[1] # Get predicted class
            btach_corr=(predicted==y_test).sum() # Calculate the number of correct predictions in the batch
            tst_corr+=btach_corr

    loss=criterion(y_val,y_test) # Calculate the loss on the last batch of the validation set.
    loss=loss.detach().numpy()
    test_losses.append(loss)
    test_correct.append(tst_corr)

print(f'\nDuration: {time.time() - start_time:.0f} seconds')
```

```
epoch:  0  batch:  200 [  2000/8000]  loss: 0.95651925  accuracy: 209.750%
epoch:  1  batch:  200 [  2000/8000]  loss: 0.58937240  accuracy: 283.600%
epoch:  2  batch:  200 [  2000/8000]  loss: 0.07277244  accuracy: 296.350%

Duration: 6641 seconds
```

Figure 7: Pembangunan Arsitektur Model SkResNet34

Potongan kode ini mendefinisikan arsitektur jaringan saraf tiruan yang akan digunakan untuk klasifikasi gambar. Model ini menggunakan model ResNet yang dimodifikasi (skresnet34) yang sudah dilatih sebelumnya (*pretrained*) dan menambahkan beberapa lapisan *fully connected* di bagian akhir untuk menyesuaikannya dengan tugas klasifikasi mobil bekas di dataset yang digunakan.

5.4 Prediksi dan Evaluasi

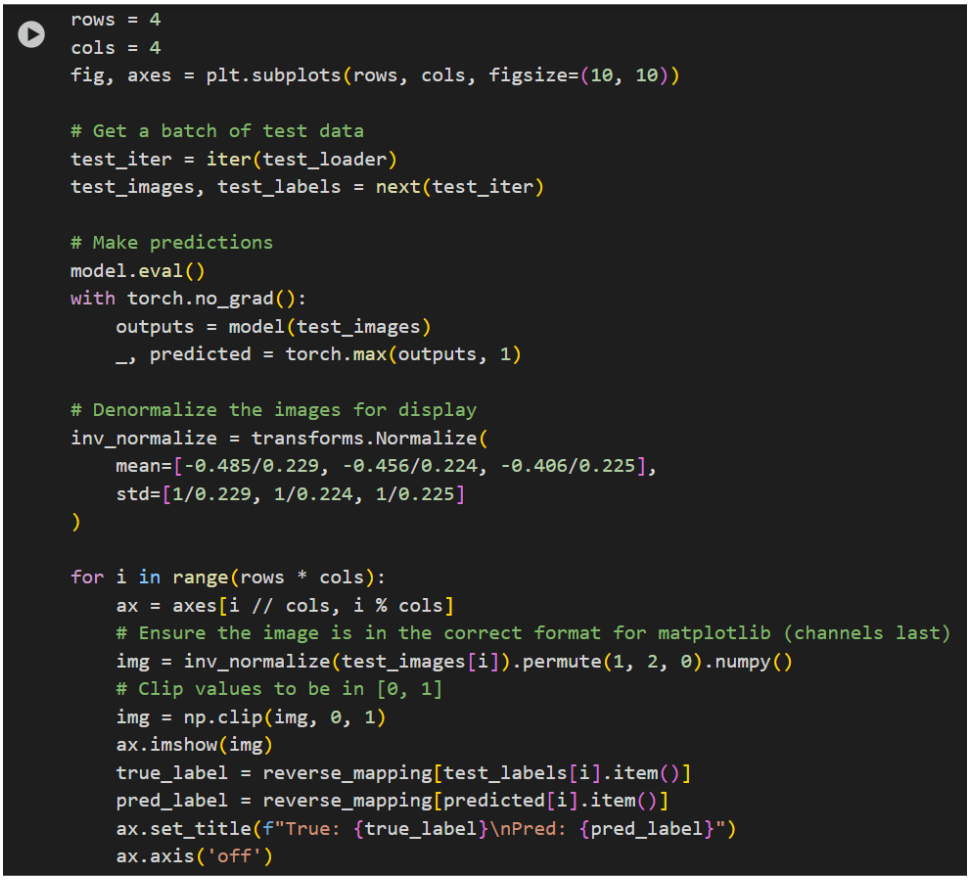


```
device = torch.device("cpu")   #"cuda:0"

model.eval() # Set the model to evaluation mode.
y_true=[]
y_pred=[]
with torch.no_grad():
    for test_data in test_loader: # Iterate through the test data loader.
        test_images, test_labels = test_data[0].to(device), test_data[1].to(device)
        pred = model(test_images).argmax(dim=1)  # Get predictions from the model and find the predicted class index.
        for i in range(len(pred)):  # Append true and predicted labels to their respective lists
            y_true.append(test_labels[i].item())
            y_pred.append(pred[i].item())

print(y_true[0:5])
print(y_pred[0:5])
```

```
[8, 8, 8, 8, 8]
[7, 8, 7, 7, 8]
```

```
# Print the classification report
print(classification_report(y_true,y_pred,target_names=class_names,digits=4))
```

Figure 8: Memunculkan Indeks Gambar yang Bernilai Benar



```
rows = 4
cols = 4
fig, axes = plt.subplots(rows, cols, figsize=(10, 10))

# Get a batch of test data
test_iter = iter(test_loader)
test_images, test_labels = next(test_iter)

# Make predictions
model.eval()
with torch.no_grad():
    outputs = model(test_images)
    _, predicted = torch.max(outputs, 1)

# Denormalize the images for display
inv_normalize = transforms.Normalize(
    mean=[-0.485/0.229, -0.456/0.224, -0.406/0.225],
    std=[1/0.229, 1/0.224, 1/0.225]
)

for i in range(rows * cols):
    ax = axes[i // cols, i % cols]
    # Ensure the image is in the correct format for matplotlib (channels last)
    img = inv_normalize(test_images[i]).permute(1, 2, 0).numpy()
    # Clip values to be in [0, 1]
    img = np.clip(img, 0, 1)
    ax.imshow(img)
    true_label = reverse_mapping[test_labels[i].item()]
    pred_label = reverse_mapping[predicted[i].item()]
    ax.set_title(f"True: {true_label}\nPred: {pred_label}")
    ax.axis('off')
```

Figure 9: Memunculkan Hasil Prediksi

Potongan kode diatas digunakan untuk memperoleh prediksi kelas dari model yang telah terlatih serta memunculkan *classification report* guna mengevaluasi model.

```
[ ]  #Display model accuracy
     import numpy as np
     print("\nAccuracy: {:.4f}".format(np.mean(np.array(y_true) == np.array(y_pred))))

     Accuracy: 0.7571

[ ]  import numpy as np
     accuracy = np.mean(np.array(y_true) == np.array(y_pred)) * 100
     print(f"\nAccuracy on test dataset: {accuracy:.4f}%")
```

Figure 10: Kode Akurasi Model

```
#Display confusion matrix
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(len(class_names), len(class_names)))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

Figure 11: Memunculkan Confusion Matrix

Potongan kode diatas memunculkan confusion matrix dan akurasi model yang dapat dilihat secara visual sehingga dapat memberikan gambaran yang lebih jelas mengenai evaluasi model.

6 Tampilan Eksekusi

Bagian ini menyajikan hasil evaluasi model terbaik yang diperoleh dari proses pelatihan dan tuning. Evaluasi dilakukan melalui metrik laporan klasifikasi, akurasi, confusion matrix serta hasil prediksi visual pada citra random.

6.1 Jumlah Epochs

```
epoch:  0  batch:  200 [  2000/8000]  loss: 0.95651925  accuracy: 209.750%
epoch:  1  batch:  200 [  2000/8000]  loss: 0.58937240  accuracy: 283.600%
epoch:  2  batch:  200 [  2000/8000]  loss: 0.07277244  accuracy: 296.350%

Duration: 6641 seconds
```

Figure 12: Jumlah Epoch yang dijalankan beserta waktu eksekusi

Gambar menunjukkan jumlah iterasi epoch yang dijalankan, cukup terbatas pada 3 *epoch* karena keterbatasan kapasitas GPU. Akurasi terbesar ada pada epoch terakhir yang merupakan model terbaik.

6.2 *Classification Report*

```
[ ]  # Print the classification report
     print(classification_report(y_true,y_pred,target_names=class_names,digits=4))
```

```
                precision    recall  f1-score   support

    Mazda-2000     0.9595    0.7717    0.8554        92
 Nissan-Zamiad     1.0000    0.7978    0.8875        89
   Peugeot-206     0.9279    0.8655    0.8957       119
  Peugeot-207i     0.5094    0.2411    0.3273       112
   Peugeot-405     0.9104    0.6854    0.7821        89
  Peugeot-Pars     0.8403    0.8929    0.8658       112
         Peykan     0.4846    0.9565    0.6433       115
     Pride-111     0.6176    0.9692    0.7545       130
     Pride-131     0.3981    0.3628    0.3796       113
          Quik     0.9562    0.8618    0.9066       152
  Renault-L90     0.7500    0.3980    0.5200        98
       Samand     0.8705    0.9528    0.9098       127
        Tiba2     0.9829    0.9127    0.9465       126

     accuracy                         0.7571      1474
    macro avg     0.7852    0.7437    0.7441      1474
 weighted avg     0.7830    0.7571    0.7503      1474
```

Figure 13: Hasil *Classification Report*

Nilai precision, recall dan F1-score yang relatif tinggi menunjukkan model cukup baik dalam membedakan tipe mobil berdasarkan kelas-kelas.

6.3 Visualisasi Hasil Prediksi

Figure 14: Contoh hasil prediksi pada beberapa citra

Gambar di atas menunjukkan beberapa contoh prediksi hasil kelas dari data uji. Sebagian besar kelas diprediksi dengan benar.

6.4 Confusion Matrix

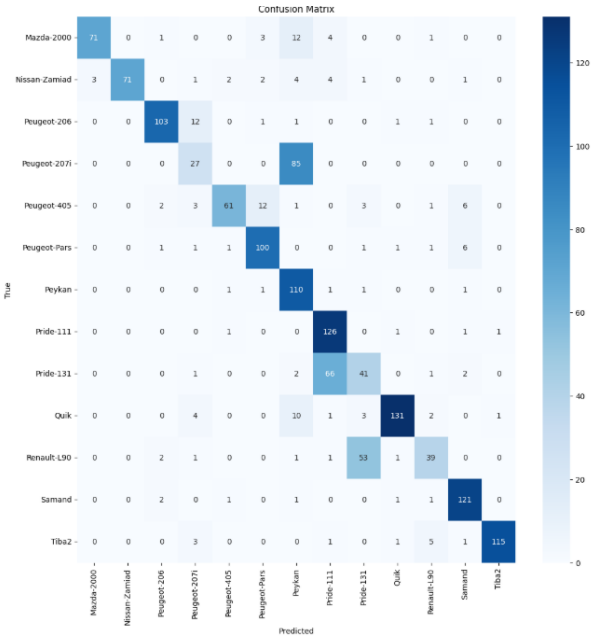


Figure 15: Confusion Matrix hasil klasifikasi pada citra

Model mampu mengklasifikasikan sebagian besar citra dengan benar. Dapat dilihat hasil prediksi yang benar pada elemen diagonal matrix.

7 Data

Dataset yang digunakan diperoleh dari platform *Kaggle*. Dataset ini terdiri dari citra mobil-mobil yang telah dipisahkan berdasarkan tipenya. Data untuk *training* dan *validation* adalah *pretrained* dan telah dipisahkan dengan rasio 70:15:15. Struktur direktori adalah sebagai berikut:

Tipe Mobil	Jumlah Data Citra
Mazda-2000	607
Nissan-Zamiad	589
Peugeot-206	787
Peugeot-207i	787
Peugeot-405	585
Peugeot-Pars	738
Peykan	761
Pride-111	858
Pride-131	749
Quik	1005
Renault-L90	646
Samand	843
Tiba2	832

Table 1: Distribusi jumlah data citra berdasarkan tipe mobil

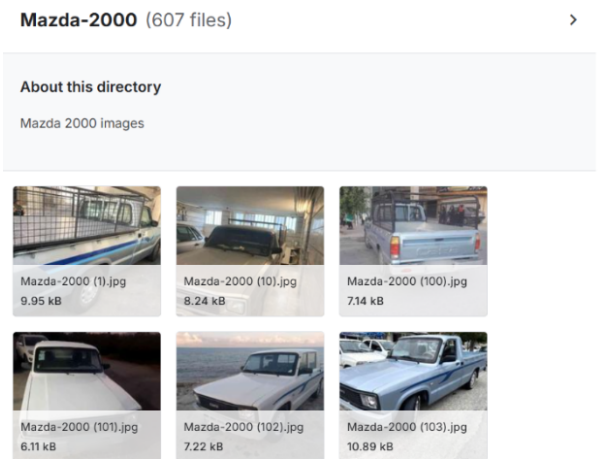


Figure 16: Contoh Citra Mazda-2000 pada Dataset

References

I. Prasetyo. Apa itu deep learning, 2024. URL `https://docif.telkomuniversity.ac.id/apa-itu-deep-learning/`. Diakses pada 2 Desember 2024.

Zhengyu Shi, Haoqi Qian, Yao Li, Fan Wu, and Libo wu. Machine learning based regional epidemic transmission risks precaution in digital society. *Scientific Reports*, 12, 11 2022. doi: 10.1038/s41598-022-24670-z.