Klasifikasi Gambar MNIST Menggunakan CNN

Azhar Zuhro
Dept. Of Computer Science
Universitas Darussalam Gontor
Ponorogo, Indonesia
azharzuhro49@student.cs.unida.gontor.ac.id

Ahmad Mukhlis Farhan
Dept. Of Computer Science
Universitas Darussalam Gontor
Ponorogo, Indonesia
ahmadmukhlisfarhan93@student.cs.unida.gontor.ac.id

Fauzi Fadillah Nugraha
Dept. Of Computer Science
Universitas Darussalam Gontor
Ponorogo, Indonesia
ffadillahn@student.cs.unida.gontor.ac.id

Fitroh dian Nugroho
Dept. Of Computer Science
Universitas Darussalam Gontor
Ponorogo, Indonesia
fitrohnugroho42028@mhs.unida.gontor.id

Fadel aliy dzulqornain
Dept. Of Computer Science
Universitas Darussalam Gontor
Ponorogo, Indonesia
fadelaliydzulqornain19@student.cs.unida.gontor.ac.id

Firman Ardiansyah Putra
Dept. Of Computer Science
Universitas Darussalam Gontor
Ponorogo, Indonesia
firmanardiansyahputra70@student.cs.unida.gontor.ac.id

Keywords—CNN,Model,Machine,Learning

I. INTRODUCTION

Proyek mini ini bertujuan untuk mengembangkan model klasifikasi gambar menggunakan arsitektur Convolutional Neural Network (CNN). Dataset yang digunakan terdiri dari beberapa kelas gambar, dan proses pengolahan dilakukan mulai dari pemuatan data, preprocessing, pembangunan model CNN, pelatihan, evaluasi performa, hingga prediksi.

II. TAHAP PELAKSANAAN

A. Set Up Library

Langkah pertama adalah mengimpor library yang diperlukan seperti TensorFlow, NumPy, Matplotlib, dan scikit-learn untuk evaluasi model..

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, D
ense, Dropout
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
```

B. Load Dataset

Visualisasi data merupakan langkah krusial dalam memahami karakteristik dataset, terutama pada proyek klasifikasi gambar aksara Jawa tulisan tangan ini. Kode yang digunakan bertujuan untuk menampilkan 25 contoh gambar pertama dari dataset pelatihan. Setiap gambar diolah, termasuk penyesuaian dimensi jika diperlukan, untuk memastikan tampilannya optimal dalam skala

grayscale 28x28 piksel. Dengan menyertakan label di setiap subplot dan menghilangkan sumbu, visualisasi ini memberikan gambaran yang jelas mengenai variasi dalam dataset dan membantu dalam eksplorasi awal sebelum melatih model CNN.

Load Dataset

```
In [2]: (X_train,y_train),(X_test,y_test)=keras.datasets.mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

In [3]: plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    if X_train[i].ndim == 1:
        plt.imshow(X_train[i].reshape(28, 28), cmap='gray')
    else:
        plt.imshow(X_train[i], cmap='gray')

    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.show()
```

C. Preprocessing Data

Normalisasi, di mana dimensi data gambar pelatihan (X_train) dan pengujian (X_test) diubah menjadi format (-1, 28, 28, 1). Nilai

-1 secara otomatis menyesuaikan jumlah sampel, sementara 28, 28 merepresentasikan tinggi dan lebar gambar dalam piksel, dan 1 menunjukkan satu channel warna untuk gambar grayscale. Selanjutnya, tipe data piksel dikonversi ke

Float32 dan dinormalisasi dengan membagi setiap nilai piksel dengan 255.0, sehingga nilainya berada dalam rentang 0 hingga 1. Normalisasi ini esensial untuk mempercepat konvergensi model selama pelatihan dan meningkatkan kinerja model secara keseluruhan. Selain normalisasi,

One-hot encoding diterapkan pada label target (y_train dan y_test) menggunakan fungsi to_categorical() dari Keras. Transformasi ini mengonversi label kelas tunggal menjadi representasi vektor biner, sebuah format yang dibutuhkan oleh fungsi.

Preprocessing Data

```
In [4]: # Normalisasi
X_train = X_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
X_test = X_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0

# One-hot encode
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

D. Perancangan arsitektur CNN

Model Convolutional Neural Network (CNN) yang digunakan dalam proyek ini dirancang untuk melakukan klasifikasi gambar grayscale berukuran 28x28 piksel ke dalam 10 kelas yang berbeda. Arsitektur model terdiri dari

dua buah lapisan konvolusi, masing-masing diikuti oleh lapisan max pooling yang berfungsi untuk mengekstraksi fitur penting dari citra sekaligus mereduksi dimensi spasial guna mengurangi beban komputasi. Setiap lapisan konvolusi menggunakan fungsi aktivasi ReLU untuk menangkap hubungan non-linear dalam data. Setelah proses ekstraksi fitur, diterapkan lapisan dropout sebesar 25% sebagai bentuk regularisasi untuk mencegah overfitting. Output dari lapisan konvolusi kemudian diratakan melalui lapisan flatten dan dilanjutkan ke lapisan dense dengan 128 neuron dan aktivasi ReLU. Sebelum menuju lapisan output, kembali diterapkan dropout sebesar 50% untuk mengurangi risiko overfitting pada model. Terakhir, lapisan output dengan 10 neuron dan fungsi aktivasi softmax menghasilkan prediksi probabilitas untuk masing-masing kelas, sehingga model dapat menentukan kelas yang paling sesuai untuk setiap citra yang diberikan.

```
from tensorflow.keras.layers import Dropout

model = Sequential([
    Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D(pool_size=(2,2)),

    Conv2D(64, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),

    Dropout(0.25),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
```

```
7]: model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

E. *Pelatihan Model*

Proses pelatihan model dilakukan menggunakan data pelatihan selama 10 epoch dengan ukuran batch sebesar 64, yang bertujuan untuk mempercepat proses komputasi tanpa mengorbankan akurasi. Selain itu, sebanyak 20% dari data pelatihan secara otomatis dipisahkan sebagai data validasi, yang berfungsi untuk mengevaluasi performa model terhadap data yang tidak digunakan dalam pelatihan langsung. Selama proses ini, metrik akurasi dan nilai loss dicatat untuk setiap epoch melalui objek history, yang nantinya digunakan sebagai dasar dalam visualisasi grafik performa model serta analisis potensi overfitting atau underfitting.

```
history = model.fit(X_train, y_train, epochs=10, batch_size=64, validation_split=0.2)

Epoch 1/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 25s 30ms/step - accuracy: 0.8118 - loss: 0.5837 - val_accuracy: 0.9
781 - val_loss: 0.0712
Epoch 2/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 21s 28ms/step - accuracy: 0.9645 - loss: 0.1110 - val_accuracy: 0.9
840 - val_loss: 0.0533
Epoch 3/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 21s 28ms/step - accuracy: 0.9758 - loss: 0.0803 - val_accuracy: 0.9
862 - val_loss: 0.0450
Epoch 4/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 21s 28ms/step - accuracy: 0.9812 - loss: 0.0633 - val_accuracy: 0.9
891 - val_loss: 0.0389
Epoch 5/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 21s 28ms/step - accuracy: 0.9832 - loss: 0.0562 - val_accuracy: 0.9
898 - val_loss: 0.0362
Epoch 6/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 21s 28ms/step - accuracy: 0.9862 - loss: 0.0454 - val_accuracy: 0.9
907 - val_loss: 0.0355
```

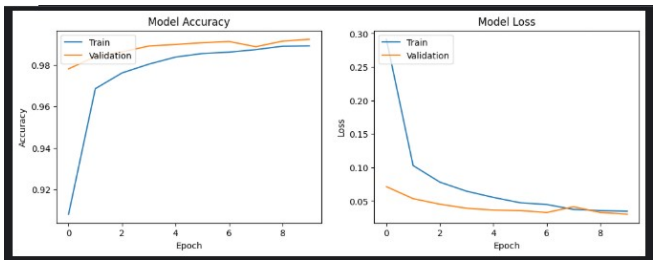
F. *Evaluasi dan Visualisasi Model*

Visualisasi hasil pelatihan dilakukan dengan memplot grafik akurasi dan loss untuk data pelatihan dan validasi sepanjang 10 epoch. Grafik akurasi menunjukkan sejauh mana model mampu mengklasifikasikan data dengan benar pada setiap epoch, sedangkan grafik loss menggambarkan seberapa besar kesalahan prediksi yang dilakukan oleh model. Kedua grafik ini ditampilkan secara berdampingan untuk memudahkan analisis performa model. Dengan membandingkan tren akurasi dan loss antara data pelatihan dan validasi, kita dapat mengevaluasi apakah model mengalami overfitting (ketika akurasi pelatihan tinggi tetapi validasi rendah), underfitting, atau berada dalam kondisi pelatihan yang stabil. Visualisasi ini sangat penting untuk mengambil keputusan dalam penyempurnaan model.

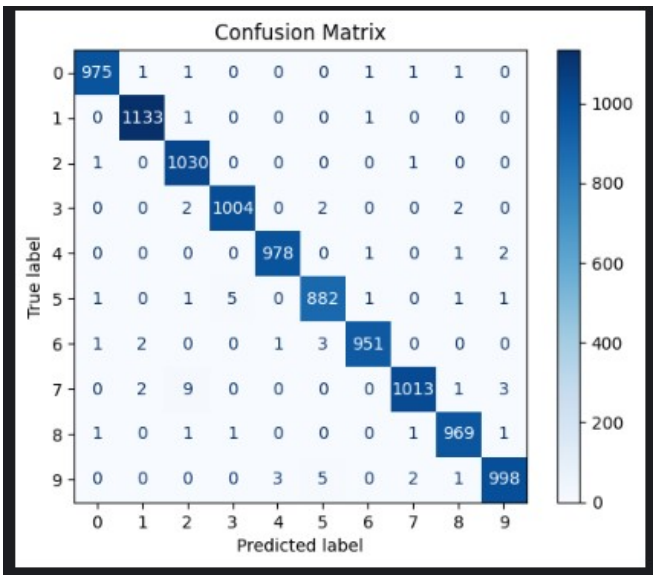
```
# Plot Akurasi
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()
```



Gambar di atas menunjukkan grafik akurasi dan loss model selama proses pelatihan sebanyak 10 epoch. Pada grafik sebelah kiri, terlihatbahwa baik akurasi data pelatihan maupun validasi mengalami peningkatan secara konsisten, dengan akurasi yang mencapai lebih dari 98% di akhir pelatihan. Hal ini menunjukkan bahwa model mampu belajar dengan baik dari data dan memiliki performa klasifikasi yang tinggi. Sementara itu, grafik di sebelah kanan memperlihatkan penurunan nilai loss baik pada data pelatihan maupun validasi, yang mengindikasikan bahwa kesalahan prediksi model semakin kecil seiring bertambahnya epoch. Pola grafik yang relatif stabil dan selaras antara data pelatihan dan validasi menandakan bahwa model tidak mengalami overfitting, serta memiliki



kemampuan generalisasi yang baik terhadap data yang belum pernah dilihat sebelumnya.

Gambar confusion matrix di atas menunjukkan hasil evaluasi model terhadap data uji, dengan sumbu vertikal mewakili label sebenarnya dan sumbu horizontal mewakili label prediksi. Setiap sel menunjukkan jumlah prediksi yang dilakukan untuk kombinasi kelas tertentu. Nilai diagonal (dari kiri atas ke kanan bawah) menunjukkan jumlah prediksi yang benar, sedangkan nilai di luar diagonal menunjukkan kesalahan klasifikasi. Misalnya, model berhasil mengklasifikasikan angka '1' dengan sangat baik sebanyak 1133 kali benar, sementara angka '5' mengalami beberapa salah klasifikasi ke kelas '3' dan '8'. Secara keseluruhan matriks ini menunjukkan performa model yang sangat baik dengan sebagian besar prediksi berada pada diagonal, yang berarti model mampu mengenali setiap kelas dengan akurasi tinggi dan tingkat kesalahan yang sangat rendah. Confusion matrix ini memperkuat hasil grafik akurasi sebelumnya yang menunjukkan bahwa model memiliki kemampuan klasifikasi yang andal dan generalisasi yang baik.

G. Kesimpulan

Berdasarkan seluruh rangkaian proses yang dilakukan dalam proyek klasifikasi gambar menggunakan Convolutional Neural Network (CNN), dapat disimpulkan bahwa model yang dibangun mampu memberikan performa yang sangat baik dalam mengenali dan mengklasifikasikan gambar ke dalam 10 kelas. Proses pelatihan yang dilakukan selama 10 epoch dengan ukuran batch 64 menunjukkan peningkatan akurasi yang stabil serta penurunan loss secara konsisten baik pada data pelatihan maupun validasi, tanpa indikasi overfitting. Visualisasi grafik akurasi dan loss menunjukkan bahwa model belajar secara optimal dan mampu melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya. Evaluasi lebih lanjut melalui confusion matrix memperkuat hasil tersebut, dengan mayoritas prediksi tepat pada diagonal matriks dan hanya sedikit kesalahan klasifikasi antar kelas. Hal ini membuktikan bahwa arsitektur CNN yang dirancang sudah efektif dalam mengekstraksi fitur dan melakukan klasifikasi gambar dengan tingkat akurasi yang tinggi. Secara keseluruhan, model CNN ini layak digunakan untuk tugas klasifikasi gambar serupa dengan kompleksitas yang sebanding.