Penerapan CNN dalam Sistem Pengenalan Pemilik Tulisan Tangan

Edia Zaki Naufal Ilman - 13521141
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521141@std.stei.itb.ac.id

***Abstrak*—Sistem pengenalan pemilik tulisan tangan memiliki peran penting dalam bidang keamanan dan verifikasi identitas. Dalam upaya mengotomatiskan proses identifikasi tulisan tangan, penggunaan teknologi pembelajaran mendalam menjadi semakin relevan. Makalah ini membahas penerapan *Convolutional Neural Network* (CNN) dalam pengenalan pemilik tulisan tangan. Metode ini memanfaatkan data gambar tulisan tangan sebagai input, menggali ciri-ciri unik melalui serangkaian layer konvolusi, dan menghasilkan model yang mampu mengidentifikasi pemilik tulisan tangan dengan tingkat akurasi yang tinggi.**

***Kata kunci*—CNN, tulisan tangan, pembelajaran mendalam, konvolusi, model**

I. PENDAHULUAN

Pengenalan pemilik tulisan tangan telah menjadi salah satu tantangan yang menarik dalam bidang biometrik dan keamanan digital. Identifikasi manual tulisan tangan membutuhkan keahlian khusus untuk memahami karakteristik unik dari setiap individu, yang sering kali memerlukan waktu dan usaha yang signifikan. Dalam konteks ini, pengembangan sistem otomatis untuk mengenali tulisan tangan menjadi kebutuhan, terutama untuk mendukung aplikasi dalam forensik, verifikasi dokumen, dan autentikasi identitas. Teknologi pembelajaran mendalam, khususnya *Convolutional Neural Network* (CNN), menawarkan pendekatan yang menjanjikan dengan kemampuan unggul dalam analisis pola visual.

CNN dikenal sebagai salah satu arsitektur pembelajaran mendalam yang efektif dalam memproses data gambar karena kemampuannya mengekstraksi fitur-fitur kompleks melalui lapisan-lapisan konvolusi. Dalam pengenalan tulisan tangan, CNN dapat mempelajari pola unik seperti bentuk huruf, tekanan, kemiringan, dan keteraturan tulisan, yang sulit dideteksi secara manual. Keunggulan ini menjadikan CNN sebagai teknologi yang sangat cocok untuk mengatasi tantangan pengenalan pemilik tulisan tangan, bahkan ketika tulisan memiliki variasi signifikan antarindividu.

Makalah ini bertujuan untuk mengeksplorasi penggunaan CNN dalam sistem pengenalan pemilik tulisan tangan. Penelitian ini akan membahas bagaimana CNN dapat digunakan dan dioptimalkan untuk mengidentifikasi ciri-ciri khas tulisan tangan. Selain itu, makalah ini akan menganalisis keefektifan dan kehandalan penggunaan CNN dalam konteks ini dengan mengacu pada evaluasi test yang ada.

Penelitian ini diharapkan dapat memberikan kontribusi dalam memperluas pemahaman tentang penggunaan teknologi pembelajaran mendalam, khususnya CNN, dalam pengenalan pola visual. Selain itu, penelitian ini juga bertujuan untuk menawarkan solusi praktis dalam pengembangan sistem yang efisien dan andal untuk identifikasi tulisan tangan. Dengan demikian, penelitian ini tidak hanya relevan untuk kebutuhan akademis tetapi juga memiliki potensi aplikasi yang luas dalam dunia industri dan keamanan modern.

II. LANDASAN TEORI

A. Citra

Citra adalah representasi visual yang terdiri dari kumpulan elemen dasar yang disebut piksel, yang disusun dalam bentuk dua dimensi. Piksel-piksel ini merupakan unit terkecil dalam citra yang mewakili informasi visual di titik tertentu pada gambar. Setiap piksel memiliki atribut tertentu yang menggambarkan nilai cahaya atau warna di titik tersebut, yang bisa berupa intensitas, bayangan, atau warna yang terlihat. Secara matematis, citra dapat didefinisikan oleh fungsi f(x,y), di mana x dan y adalah koordinat horizontal dan vertikal, masing-masing mewakili posisi piksel dalam gambar. Nilai f yang dipetakan pada titik (x, y) menunjukkan intensitas atau warna pada titik tersebut, yang bisa berupa skala abu-abu pada citra hitam putih atau komponen warna dalam citra berwarna.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

Gambar II. 1. Representasi citra
Sumber: Slide materi kuliah

Dalam citra digital, seluruh kumpulan piksel ini tersusun dalam array angka yang terorganisasi dalam baris dan kolom, membentuk grid dua dimensi. Citra digital adalah kumpulan data yang mengandung informasi visual yang disusun dalam struktur yang dapat diolah oleh komputer. Elemen-elemen ini memungkinkan citra digital untuk menyimpan informasi visual dengan tingkat detail yang tinggi, yang dapat diproses lebih lanjut untuk berbagai keperluan, mulai dari pengolahan citra

hingga analisis pola dan pengenalan objek. Teknologi pengolahan citra modern memungkinkan kita untuk melakukan berbagai transformasi dan analisis pada citra digital ini, seperti memperbaiki kualitas gambar, mengekstraksi fitur, dan mengenali objek atau pola tertentu dalam citra. Secara umum dikenal 3 jenis citra digital: citra biner, citra grayscale, dan citra berwarna.

1) *Citra Biner*

Citra biner adalah jenis citra digital yang hanya memiliki dua nilai intensitas untuk setiap piksel, biasanya 0 dan 1, atau hitam dan putih. Citra ini digunakan untuk merepresentasikan informasi visual yang sederhana, seperti bentuk, tepi, atau pola, di mana setiap piksel hanya menunjukkan apakah suatu area termasuk dalam objek (putih) atau latar belakang (hitam). Representasi biner sering digunakan dalam aplikasi seperti segmentasi objek, deteksi tepi, atau analisis pola sederhana, karena sifatnya yang efisien dan mudah diproses.

Gambar II. 2. Citra biner

2) *Citra Grayscale*

Citra grayscale adalah citra digital yang merepresentasikan informasi visual dalam skala abu-abu, dengan setiap piksel memiliki nilai intensitas dalam rentang tertentu, biasanya dari 0 (hitam) hingga 255 (putih) untuk citra 8-bit. Tidak seperti citra berwarna, citra grayscale hanya memiliki satu saluran (channel) yang menyimpan informasi tingkat kecerahan di setiap piksel. Citra ini sering digunakan dalam aplikasi pengolahan citra seperti deteksi tepi, analisis tekstur, atau pengenalan pola, karena hanya mempertimbangkan informasi intensitas tanpa memperhitungkan warna.

Gambar II. 3. Citra grayscale

3) *Citra Berwarna*

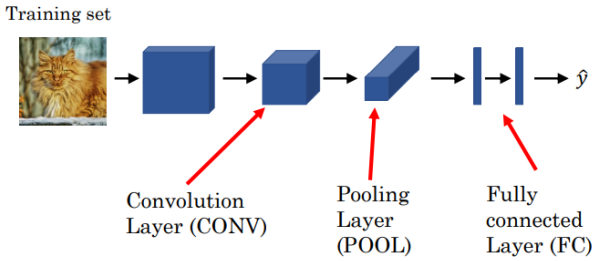
Citra berwarna, atau citra RGB, adalah citra digital yang merepresentasikan informasi visual menggunakan tiga saluran warna: merah (*Red*), hijau (*Green*), dan biru (*Blue*). Setiap piksel dalam citra RGB memiliki tiga nilai intensitas, masing-masing untuk ketiga saluran tersebut, dan kombinasi nilai-nilai ini menghasilkan warna akhir yang terlihat. Model RGB memungkinkan representasi warna yang sangat bervariasi dan realistis, menjadikannya format umum untuk citra digital dalam berbagai aplikasi seperti fotografi, grafik komputer, dan pengolahan citra berwarna.

Gambar II. 4. Citra berwarna

Melalui representasi-representasi ini, citra digital dapat dianalisis, dimodifikasi, atau diproses untuk berbagai tujuan, termasuk dalam pengenalan objek, pengolahan citra medis, deteksi tepi, dan berbagai aplikasi lainnya. Struktur matriks piksel yang digunakan dalam citra digital memungkinkan aplikasi teknologi komputer untuk melakukan tugas-tugas kompleks yang melibatkan pengolahan visual secara efisien. Dengan demikian, citra digital dan model warna seperti RGB menjadi dasar penting dalam bidang pengolahan citra dan analisis visual.

B. *Convolutional Neural Network*

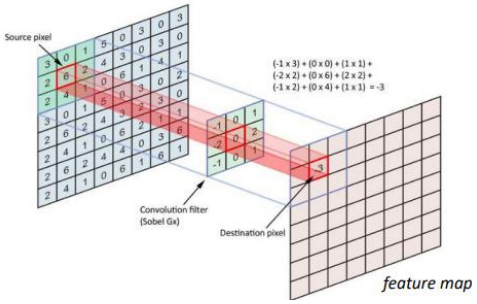
Convolutional Neural Network atau dikenal dengan CNN merupakan sebuah arsitektur jaringan syaraf tiruan yang dirancang untuk mengangani data dengan struktur grid seperti citra digital. CNN efektif dalam tugas-tugas pengenalan pola visual karena kemampuannya untuk mengakstraksi fitur atau ciri penting dari input. CNN bekerja malalui serangkaian lapisan (*layers*) yang memiliki tugas spesifik masing-masing. Tiga lapisan utama yang membetuk CNN adalah lapisan konvolusi (*convolutional layer*), lapisan pooling (*pooling layer*), dan lapisan yang terhubung penuh (*fully connected layers*). Lapisan-lapisan ini krusial dan memungkinkan CNN untuk mempelajari representasi data kompleks seiring bertambahnya kedalaman jaringan.



Gambar II. 5. Tiga lapisan utama pada CNN
Sumber: Slide materi kuliah

1) *Convolutional Layer (CONV)*

Pada lapisan konvolusi dilakukan operasi konvolusi pada citra untuk mengekstraksi fitur atau ciri daru input tersebut. Pada lapisan ini, sebuah filter kecil atau kernel melintasi citra input dan menghitung produk titik antara nilai piksel dan bobot filter, menghasilkan peta fitur (*feature map*) yang mewakili pola lokal dalam data. Setiap filter belajar mendeteksi fitur spesifik seperti tepi, tekstur, atau pola tertentu. Lapisan ini memungkinkan CNN untuk menangkap informasi spasial dalam citra, seperti posisi relatif antara elemen-elemen gambar.



Gambar II. 6. Operasi konvolusi
Sumber: Slide materi kuliah

Berikut adalah elemen-elemen utama dalam lapisan kovolusi:

a) *Kernel*

Kernel, juga dikenal sebagai filter, adalah matriks kecil dengan nilai numerik yang digunakan dalam lapisan konvolusi untuk mengekstraksi fitur dari data input. Kernel melintasi citra input dalam proses yang disebut konvolusi, di mana ia menghitung produk titik antara elemen kernel dan piksel dalam jendela input yang sesuai. Hasilnya adalah peta fitur yang mencerminkan pola tertentu dalam data, seperti tepi, sudut, atau tekstur. Ukuran kernel, seperti 3×3 atau 5×5, memengaruhi tingkat detail yang dapat diekstraksi oleh CNN, dengan kernel yang lebih kecil fokus pada pola lokal dan kernel yang lebih besar menangkap informasi global.

b) *Stride*

Stride adalah langkah pergeseran kernel saat melintasi citra input selama proses konvolusi. Stride menentukan seberapa jauh kernel bergeser setiap kali operasi konvolusi dilakukan. Stride default adalah 1, yang berarti kernel bergeser satu piksel setiap kali. Stride yang lebih besar menghasilkan peta fitur yang lebih kecil karena pengambilan sampel dilakukan dengan langkah lebih besar, sehingga mengurangi jumlah output. Dengan mengatur stride, kita dapat mengontrol tingkat pengurangan dimensi dari data input.

c) *Padding*

Padding adalah teknik menambahkan piksel tambahan di tepi citra input sebelum dilakukan konvolusi. Biasanya, nilai padding diisi dengan nol (*zero-padding*). Padding digunakan untuk mempertahankan ukuran output agar tetap sama dengan ukuran input atau untuk mencegah kehilangan informasi di tepi citra selama operasi konvolusi. Tanpa padding, ukuran peta fitur akan semakin mengecil setelah beberapa lapisan konvolusi, sehingga padding menjadi penting untuk jaringan yang membutuhkan output dengan dimensi tertentu.

d) *Peta Fitur*

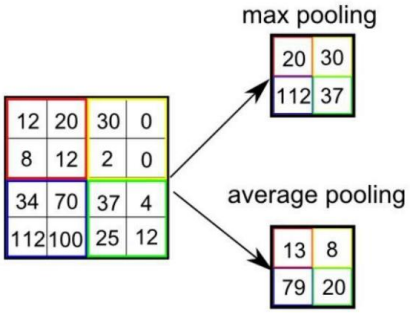
Peta fitur adalah hasil dari operasi konvolusi antara kernel dan citra input. Peta fitur merepresentasikan pola atau karakteristik yang terdeteksi oleh kernel dalam citra, seperti garis, tepi, atau tekstur. Setiap peta fitur dihasilkan oleh kernel yang berbeda, sehingga jaringan dapat menangkap berbagai aspek dari data input. Peta fitur merupakan dasar bagi CNN untuk mengenali pola yang lebih kompleks saat melewati lapisan-lapisan yang lebih dalam.

e) *Fungsi Aktivasi*

Fungsi aktivasi dalam lapisan konvolusi bertugas memperkenalkan non-linearitas ke dalam model, memungkinkan jaringan untuk mempelajari pola yang kompleks dalam data. Fungsi aktivasi yang umum digunakan adalah ReLU (*Rectified Linear Unit*), yang menggantikan semua nilai negatif dalam peta fitur dengan nol, sambil mempertahankan nilai positif. Dengan cara ini, ReLU meningkatkan efisiensi komputasi dan mencegah masalah seperti vanishing gradient. Fungsi aktivasi membantu model menangkap hubungan non-linear dalam data, yang penting untuk mengenali pola yang lebih rumit.

2) *Pooling Layer (POOL)*

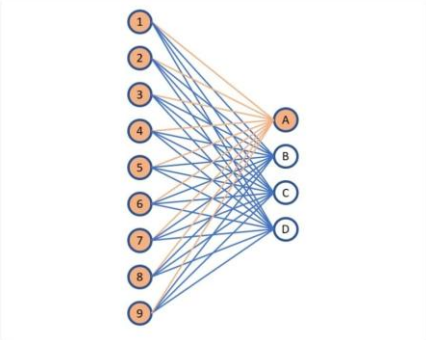
Lapisan ini berfungsi untuk mereduksi dimensi peta fitur yang dihasilkan oleh lapisan konvolusi, sambil mempertahankan informasi yang paling penting. Operasi pooling, seperti max pooling atau average pooling, dilakukan dengan mengambil nilai maksimum atau rata-rata dari sekumpulan nilai dalam jendela tertentu. Dengan mengurangi dimensi data, lapisan pooling membantu mengurangi jumlah parameter, meningkatkan efisiensi komputasi.



Gambar II. 7. Operasi pooling
Sumber: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53

3) *Fully Connected Layer (FC)*

Lapisan ini menghubungkan setiap neuron dari peta fitur yang dihasilkan lapisan sebelumnya ke setiap neuron dalam lapisan berikutnya. Pada tahap ini, peta fitur yang telah diringkas melalui lapisan konvolusi dan pooling diratakan menjadi vektor satu dimensi, kemudian diproses melalui lapisan FC untuk menghasilkan output akhir. Lapisan ini bertanggung jawab untuk mempelajari hubungan non-linear antar fitur dan mengklasifikasikan data berdasarkan pola yang telah dipelajari. Output lapisan terakhir biasanya berupa probabilitas yang menunjukkan kemungkinan data input termasuk dalam masing-masing kelas.

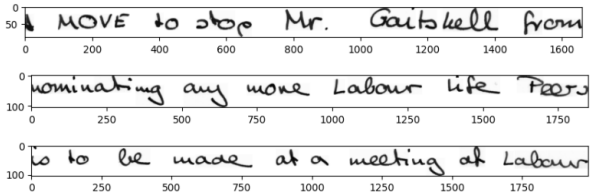


Gambar II. 8. *Fully connected layer*
Sumber: https://builtin.com/machine-learning/fully-connected-layer

III. IMPLEMENTASI

A. *Praproses Dataset*

Untuk eksperimen ini, digunakan dataset IAM Handwriting Top 50 yang berisi kumpulan data tulisan tangan dari 50 penulis berbeda. Citra tulisan tangan yang berada pada dataset memiliki bentuk citra grayscale berjumlah 4899 citra. Sebelum dapat digunakan, dataset ini perlu dilakukan praproses terlebih dahulu untuk dapat dimasukkan sebagai bahan latih dari model. Tahap praproses ini mencakup mengubah citra menjadi citra grayscale dan mengubah ukuran citra.



Gambar III. 1. Contoh data pada dataset

Proses praproses data dimulai dengan membaca citra dan mengonversinya menjadi format grayscale. Konversi ini bertujuan untuk menyederhanakan informasi dalam citra dengan hanya menggunakan intensitas cahaya sebagai representasi data. Setelah itu, citra yang telah dikonversi diubah ukurannya menjadi dimensi tetap, yaitu 128x128 piksel.

Penyesuaian ukuran ini dilakukan untuk memastikan semua citra memiliki dimensi yang seragam, sehingga memudahkan pemrosesan lebih lanjut oleh model. Dengan langkah ini, informasi yang relevan tetap terjaga, sementara variabilitas dalam ukuran asli citra dihilangkan. Proses praproses ini memastikan data yang digunakan memiliki format yang konsisten, sehingga mendukung model dalam mengenali pola secara lebih efektif dan akurat.

Berikut merupakan implementasi dari tahap ini:

```
def preprocess_image(path):
 image = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
 resized_image = cv2.resize(image, IMAGE_SIZE)
 return resized_image
```

B. *Parsing Dataset*

Proses parsing dan pemuatan data dimulai dengan membaca file label yang berisi informasi mengenai citra tulisan tangan dan penulisnya. File label tersebut diproses untuk membuat pasangan kunci dan nilai, di mana kunci merupakan identitas unik citra, dan nilai adalah ID penulisnya. Selanjutnya, dilakukan pencarian semua jalur file citra di dalam folder dataset dengan memanfaatkan metode pencocokan pola. Nama file dari setiap citra kemudian diurai untuk memperoleh informasi identitas uniknya, yang selanjutnya dibandingkan dengan kunci dari file label untuk mencocokkan citra dengan ID penulis yang sesuai.

Setelah identitas citra dan labelnya diproses, citra-citra tersebut dibaca satu per satu dari jalurnya. Citra yang berhasil ditemukan akan melalui tahap praproses, yaitu transformasi menjadi format yang sesuai dengan model. Proses ini melibatkan konversi setiap citra ke array numpy dengan tipe data float32 dan normalisasi nilai piksel ke rentang 0 hingga 1. Dimensi array juga disesuaikan untuk memastikan kompatibilitas dengan lapisan input model pembelajaran mesin. Hasil akhir dari proses ini adalah data citra dan label yang telah dipersiapkan untuk digunakan dalam pelatihan dan pengujian model.

Berikut merupakan implementasi dari tahap parsing:

```
def load_dataset():
  images = []
  labels = []

  d = {}
  with open(LABEL_FILE) as f:
    for line in f:
      key = line.split(' ')[0]
      writer = line.split(' ')[1]
      d[key] = writer

  paths = []
  path_to_files = os.path.join(DATASET_FOLDER, '*')
  for filename in sorted(glob.glob(path_to_files)):
    paths.append(filename)
    image_name = filename.split('/')[-1]
    file, ext = os.path.splitext(image_name)
    parts = file.split('-')
```

```
    form = parts[0] + '-' + parts[1]
    for key in d:
      if key == form:
        labels.append(str(d[form]))


  for path in tqdm(paths):
    if os.path.exists(path):
      images.append(preprocess_image(path))



  images = np.array(images,dtype=np.float32)/255.0
  images = images.reshape(-1, IMAGE_SIZE[0],
IMAGE_SIZE[1], 1)
  return images, labels
```

C. Encoding dan Pembagian Dataset

Setelah data citra dan label diproses, langkah berikutnya adalah mengonversi label teks menjadi format numerik dengan menggunakan encoder label. Konversi ini bertujuan agar label dapat digunakan sebagai target dalam pelatihan model pembelajaran mesin. Selanjutnya, dataset dibagi menjadi data pelatihan dan pengujian dengan perbandingan 80:20. Proses pembagian ini dilakukan secara acak dengan memastikan distribusi data tetap seimbang untuk memastikan hasil yang konsisten.

Berikut merupakan implementasi dari tahap *encoding* dan pembagian dataset:

```
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)

X_train, X_test, y_train, y_test =
train_test_split(images, labels, test_size=0.2,
random_state=42)
```

D. Pembangunan Model CNN

Model CNN yang dibangun diawali dengan lapisan padding untuk menyesuaikan dimensi input, diikuti dengan serangkaian lapisan konvolusi untuk mendeteksi fitur lokal dari citra. Setiap lapisan konvolusi menggunakan filter dengan ukuran kernel yang ditentukan, diikuti oleh lapisan pooling untuk mengurangi dimensi fitur dan mempercepat proses komputasi.

Setelah ekstraksi fitur, model dilengkapi dengan lapisan dense yang bertugas mempelajari hubungan kompleks antara fitur-fitur yang telah diekstraksi. Lapisan dropout ditambahkan untuk mencegah overfitting dengan cara mengabaikan sebagian neuron selama pelatihan. Lapisan output model menggunakan fungsi aktivasi softmax untuk menghasilkan probabilitas bagi setiap kelas, sesuai dengan jumlah total kelas penulis. Model ini dioptimalkan menggunakan algoritma Adam dengan fungsi loss entropi kategorikal yang cocok untuk tugas klasifikasi, serta metrik akurasi untuk mengevaluasi kinerja selama pelatihan. Arsitektur model yang telah dibangun ini kemudian dirangkum untuk memastikan semua parameter dan dimensi telah sesuai sebelum memulai pelatihan.

Berikut merupakan implementasi dari tahap pembangunan model:

```
def create_model(input_shape, num_classes):
  model = models.Sequential([
    layers.ZeroPadding2D((1, 1),
input_shape=input_shape),
    layers.Conv2D(filters= 32, kernel_size =(5,5),
strides= (2,2), padding='same', name='conv1',
activation='relu'),

layers.MaxPooling2D(pool_size=(2,2),strides=(2,2),
name='pool1'),



    layers.Conv2D(filters= 64, kernel_size =(3,3),
strides= (1,1), padding='same', name='conv2',
activation='relu'),
layers.MaxPooling2D(pool_size=(2,2),strides=(2,2),
name='pool2'),
    layers.Conv2D(filters= 128, kernel_size =(3,3),
strides= (1,1), padding='same', name='conv3',
activation='relu'),
  layers.MaxPooling2D(pool_size=(2,2),strides=(2,2),
name='pool3'),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(512, name='dense1',
activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(256, name='dense2',
activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(50,name='output',
activation='softmax')
       ])
  model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
  return model


model = create_model(input_shape=(IMAGE_SIZE[0],
IMAGE_SIZE[1], 1), num_classes=50)


print(model.summary())
```

Layer (type)	Output Shape	Param #
zero_padding2d_12 (ZeroPadding2D)	(None, 130, 130, 1)	0
conv1 (Conv2D)	(None, 65, 65, 32)	832
pool1 (MaxPooling2D)	(None, 32, 32, 32)	0
conv2 (Conv2D)	(None, 32, 32, 64)	18,496
pool2 (MaxPooling2D)	(None, 16, 16, 64)	0
conv3 (Conv2D)	(None, 16, 16, 128)	73,856
pool3 (MaxPooling2D)	(None, 8, 8, 128)	0
flatten_16 (Flatten)	(None, 8192)	0
dropout_37 (Dropout)	(None, 8192)	0
dense1 (Dense)	(None, 512)	4,194,816
dropout_38 (Dropout)	(None, 512)	0
dense2 (Dense)	(None, 256)	131,328
dropout_39 (Dropout)	(None, 256)	0
output (Dense)	(None, 50)	12,850

Total params: 4,432,178 (16.91 MB)
Trainable params: 4,432,178 (16.91 MB)
Non-trainable params: 0 (0.00 B)

Gambar III. 2. Rangkuman dari lapisan-lapisan model

E. Pelatihan Model

Pada pelatihan model dilatih menggunakan data yang telah dipersiapkan sebelumnya. Pelatihan dilakukan selama 100 epoch, yang berarti model akan memproses seluruh data latih sebanyak 100 kali untuk menyempurnakan bobot dan parameter internalnya. Data dibagi menjadi batch kecil berukuran 32 sampel per batch, memungkinkan pelatihan dilakukan secara bertahap dan lebih efisien dalam penggunaan memori. Selain itu, pada setiap epoch, performa model dievaluasi menggunakan data validasi untuk memantau kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya.

Berikut merupakan implementasi dari tahap pembangunan model:

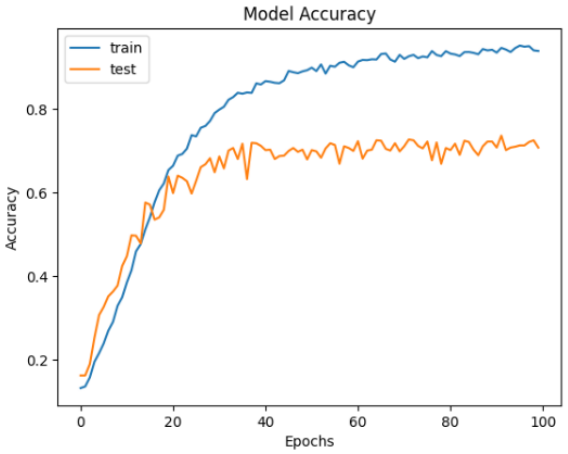
```
model.fit(X_train, y_train, epochs=100,
batch_size=32, validation_data=(X_test, y_test),
verbose=1)
```

F. Evaluasi Model

Pada akhir pelatihan dilakukan evaluasi terhadap model. Tahap evaluasi model ini bertujuan untuk menganalisis kinerja model selama proses pelatihan dengan memvisualisasikan akurasi pada data latih dan data validasi. Grafik akurasi dibuat dengan memplot nilai akurasi yang dicatat pada setiap epoch untuk data latih dan data validasi. Sumbu horizontal menunjukkan jumlah epoch, sementara sumbu vertikal menunjukkan tingkat akurasi. Visualisasi ini membantu memahami bagaimana model belajar seiring waktu dan mengidentifikasi potensi overfitting atau underfitting berdasarkan kesenjangan antara akurasi pada data latih dan data validasi.

Berikut merupakan implementasi dari tahap evaluasi model:

```
plt.plot(History.history['accuracy'])
plt.plot(History.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
```



Gambar III. 3. Grafik akurasi model

IV. Analisis dan Pembahasan

Dapat dilihat dari hasil implementasi dan evaluasi yang dilakukan pada model yang dibuat bahwa model memiliki akurasi dalam mengenali pemilik tulisan tangan mencapai 94% pada data latih dan mencapai 70% pada data uji. Akurasi ini cukup baik dan membuktikan kemampuan dan keefektifan CNN dalam mengenali dan mengklasifikasi pemilik tulisan tangan menggunakan citra. Akan tetapi, perbedaan yang cukup besar pada akurasi latih dan uji juga mengindikasikan bahwa model mengalami overfitting. Kondisi ini menunjukkan bahwa model mampu mengenali pola-pola spesifik dalam data latih dengan sangat baik, tetapi kesulitan melakukan generalisasi pada data baru. Hal ini kemungkinan disebabkan oleh kompleksitas model yang terlalu tinggi, jumlah data latih yang tidak mencukupi, atau perbedaan karakteristik antara data latih dan data pengujian

Untuk mengatasi masalah ini, beberapa langkah perbaikan dapat dilakukan, seperti menambah jumlah dan keragaman data latih melalui augmentasi, menerapkan teknik regularisasi untuk mengurangi ketergantungan model pada data latih, menyederhanakan arsitektur model agar tidak terlalu kompleks, dan mengoptimalkan hyperparameter seperti learning rate atau jumlah epoch. Dengan perbaikan tersebut, diharapkan model dapat mencapai keseimbangan antara akurasi pelatihan dan kemampuan generalisasi, sehingga performa pada data pengujian dapat meningkat.

V. Kesimpulan

Penggunaan CNN terbukti efektif dalam mengklasifikasikan pemilik tulisan tangan dan memberikan akurasi yang memuaskan. Namun, perbedaan akurasi latih dan uji menunjukkan adanya indikasi overfitting, di mana model mampu mengenali pola pada data latih dengan baik tetapi kurang mampu melakukan generalisasi pada data baru. Hal ini dapat disebabkan oleh beberapa faktor, seperti kompleksitas model yang terlalu tinggi dibandingkan dengan jumlah data atau kurangnya variasi dalam data latih. Meskipun demikian, model tetap berhasil menunjukkan kemampuan dasar dalam mengenali tulisan tangan sesuai dengan pemiliknya, yang menunjukkan potensi untuk ditingkatkan dengan optimasi lebih

lanjut, seperti penambahan data atau penerapan teknik regularisasi.

UCAPAN TERIMA KASIH

Penulis bersyukur kepada Allah SWT atas berkat dan rahmat-Nya, penulis bisa menyelesaikan tugas makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir selaku dosen mata kuliah Pengolahan Citra Digital, yang selama ini telah membimbing dalam pembelajaran.

REFERENCES

[1] Patel, Isha & Patel, Sanskruti & Patel, Atul. (2018). Analysis of Various Image Preprocessing Techniques for Denoising of Flower Images. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. 6. 1111-1117. 10.26438/ijcse/v6i5.11111117.)

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/21-CNN-2024.pdf. Diakses pada 14 Januari 2025

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/Makalah2023/Makalah-IF4073-Citra-2023%20(17).pdf. Diakses pada 15 Januari 2025

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 15 Januari 2025

Edia Zaki Naufal Ilman
13521141