Deteksi Objek Buah-buahan Menggunakan Metode Convolutional Neural Network (CNN)

Danar Arif Siddiq
Telkom University
Jl. Telekomunikasi. 1, Terusan Buahbatu - Bojongsoang, Telkom University, Sukapura, Kec. Dayeuhkolot, Kabupaten Bandung, Jawa Barat 40257
danararifsiddiq@student.telkomuniversity.ac.id

Abstract

Laporan ini membahas implementasi metode Convolutional Neural Network (CNN) untuk deteksi objek buah-buahan pada gambar. Tujuan dari penelitian ini adalah untuk mengembangkan sistem otomatis yang dapat mengenali dan lokaliser buah-buahan dalam citra digital. CNN dipilih sebagai metode utama karena kemampuannya dalam mengekstrak fitur hierarkis dari data gambar.

Penelitian ini dimulai dengan pengumpulan dataset yang terdiri dari berbagai gambar buah-buahan yang mencakup variasi pose, ukuran, dan kondisi pencahayaan. Data tersebut kemudian dibagi menjadi set pelatihan dan set pengujian untuk melatih dan menguji model CNN. Arsitektur CNN yang dipilih dirancang dengan lapisan konvolusi, lapisan pooling, dan lapisan fully connected untuk memproses informasi gambar secara efektif.

Proses pelatihan model dilakukan dengan menggunakan algoritma pembelajaran mendalam (deep learning) menggunakan set pelatihan. Setelah model terlatih, evaluasi dilakukan pada set pengujian untuk mengukur kinerja deteksi objek buah-buahan. Hasil evaluasi mencakup metrik seperti akurasi, presisi, recall, dan F1-score.

Hasil penelitian menunjukkan bahwa metode CNN mampu memberikan kinerja deteksi objek buah-buahan yang memuaskan, dengan kemampuan untuk mengenali berbagai jenis buah-buahan dalam konteks gambar yang kompleks. Implikasi dari penelitian ini adalah pengembangan sistem otomatis yang dapat digunakan dalam berbagai aplikasi, termasuk pengolahan gambar otomatis, sistem pengawasan keamanan, dan pengenalan pola pada buah-buahan.

Kata kunci: deteksi objek, buah-buahan, Convolutional Neural Network (CNN), deep learning, pengolahan gambar.

1. Pendahuluan

1.1. Latar Belakang

Pengenalan dan deteksi objek merupakan aspek krusial dalam pengembangan teknologi komputer vision. Dalam beberapa tahun terakhir, deteksi objek dengan menggunakan metode Convolutional Neural Network (CNN) telah menjadi fokus utama penelitian karena kemampuannya dalam mengekstrak fitur secara otomatis dari data gambar, sehingga meningkatkan akurasi dan kecepatan proses deteksi.

Salah satu konteks aplikasi yang sangat relevan adalah deteksi objek buah-buahan. Dalam industri pertanian, perdagangan, dan pengolahan makanan, kemampuan untuk secara otomatis mengenali dan memilah buah-buahan dalam gambar memiliki potensi untuk meningkatkan efisiensi produksi, mengurangi biaya tenaga kerja, dan mengoptimalkan rantai pasok.

Meskipun telah ada penelitian sebelumnya tentang deteksi objek menggunakan CNN, tantangan khusus terkait dengan citra buah-buahan membuatnya menjadi subjek penelitian yang menarik. Variabilitas dalam bentuk, warna, dan ukuran buah-buahan, serta variasi kondisi pencahayaan dan latar belakang gambar, menambah kompleksitas dalam mengembangkan model yang dapat bekerja secara efektif dalam situasi dunia nyata.

Oleh karena itu, penelitian ini bertujuan untuk menyelidiki potensi metode CNN dalam konteks deteksi objek buah-buahan. Dengan memahami dan mengatasi tantangan khusus yang terkait dengan jenis citra ini, diharapkan dapat menghasilkan sistem yang dapat diandalkan dan diterapkan di berbagai sektor, termasuk pertanian, perdagangan, dan industri makanan.

Penelitian ini juga dapat memberikan kontribusi pada pengembangan teknologi pengolahan gambar dan computer vision secara umum, membuka pintu untuk aplikasi lebih lanjut dalam mendukung otomatisasi tugas-tugas visual kompleks di berbagai industri. Dengan demikian, latar belakang penelitian ini menciptakan landasan untuk pengembangan solusi teknologi yang inovatif dan berdampak signifikan dalam memecahkan masalah dunia nyata.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah disajikan, penelitian ini bertujuan untuk menjawab beberapa pertanyaan kunci terkait dengan deteksi objek buah-buahan menggunakan metode Convolutional Neural Network (CNN). Oleh karena itu, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Apakah melakukan percobaan deteksi objek buah-buahan menggunakan metode CNN efektif?
2. Berapa hasil akurasi dari percobaan deteksi buah-buahan menggunakan metode CNN yang didapatkan?
3. Berapa loss yang didapatkan dari hasil percobanan deteksi buah-buahan menggunakan metode CNN?

.

1.3. Tujuan

Tujuan dari dilakukannya percobaan ini sebagai berikut:

1. Melakukan evaluasi kinerja model CNN pada dataset yang telah dikembangkan, dengan fokus pada metrik seperti akurasi dan presisi
2. Merancang arsitektur Convolutional Neural Network yang optimal untuk deteksi objek buah-buahan.

2. Metode

2.1. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah model deep learning yang biasa digunakan untuk klasifikasi, termasuk klasifikasi teks. CNN dapat mengekstrak fitur penting dari data masukan melalui operasi konvolusi dan pooling, dan kemudian menggunakan lapisan sepenuhnya terhubung untuk klasifikasi berdasarkan fitur-fitur tersebut [1].

CNN memiliki kekurangan dalam konteks pengolahan teks dan klasifikasi teks, yang membentuk mengekstrak fitur tingkat tinggi, menangkap asosiasi laten, dan lebih akurat dari LSTM. Namun, CNN memiliki keterbatasan dalam mengekspresikan fitur kontekstual jangka panjang, yang merupakan kelemahan inheren dari model [2].

2.2. Deteksi Objek

Deteksi objek dalam dunia informatika adalah proses yang memungkinkan sistem komputer untuk mengidentifikasi dan menentukan lokasi objek khusus dalam gambar atau video. Proses ini melibatkan beberapa langkah kunci, seperti ekstraksi fitur, di mana karakteristik unik dari objek diidentifikasi, dan pelatihan model, di mana model deteksi objek, seperti Convolutional Neural Network (CNN), dilatih menggunakan dataset berlabel. Setelah melalui tahap pelatihan, model dapat diterapkan pada gambar atau video untuk mengenali dan melokalisasi objek-objek yang diinginkan, seringkali dengan pembuatan kotak pembatas (bounding box) untuk menunjukkan area di sekitar objek. Evaluasi kinerja model kemudian dilakukan, dan jika diperlukan, model dapat ditingkatkan untuk meningkatkan akurasi dan keandalan deteksi. Deteksi objek memiliki berbagai aplikasi, mulai dari visi komputer dan keamanan hingga kendaraan otonom dan pengawasan lalu lintas, membuktikan nilai signifikan dalam perkembangan teknologi informasi modern.

2.3. Tools Yang Digunakan

Tools yang digunakan untuk melakukan percobaan ini sebagai berikut :

1. Google Colab
2. Bahasa Pemrograman Python
3. Github

2.4. Dataset

Dataset yang digunakan dalam percobaan ini adalah dataset “Fruit Images for Object Detection” dari kaggle

yang berukuran 30 MB, file tersebut berisi 120 file data test dan juga 480 data train yang terdiri dari berbagai macam buah-buahan.

3. Eksperimen

3.1. Upload Dataset

Pada tahap pertama untuk melakukan percobaan yang saya lakukan terlebih dahulu melakukan upload dataset yang terdaapat dari kaggle ke dalam google colab

```
[1]  1 ! pip install kaggle
[2]  1 ! mkdir ~/.kaggle
[4]  1 ! cp kaggle.json ~/.kaggle/
[3]  1 ! chmod 600 ~/.kaggle/kaggle.json
     1 ! kaggle datasets download -d mbkinaci/fruit-images-for-object-detection
```

setelah dataset berhasil di upload maka selanjutnya yang saya lakukan adalah melakukan ekstrak file tersebut agar mudah dalam melakukan pengambilam file nantinya

```
1 ! unzip fruit-images-for-object-detection.zip
```

3.2. Import Library

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import cv2
6 from glob import glob
7 from keras.preprocessing.image import load_img
8 from sklearn.model_selection import train_test_split
9 import os
```

Informasi mengenai library yagn digunakan sebagai berikut :

1. *numpy*: Diimpor sebagai “np”, ini adalah library untuk bahasa pemrograman Python, menambahkan dukungan untuk array dan matriks multi-dimensi besar, bersama dengan koleksi besar fungsi matematika tingkat tinggi untuk mengoperasikan array ini.
2. *pandas*: Diimpor sebagai “pd”, menawarkan struktur data dan operasi untuk memanipulasi tabel numerik dan seri waktu.
3. *matplotlib.pyplot*: Diimpor sebagai “plt”, menyediakan antarmuka seperti MATLAB untuk membuat plot dan grafik.
4. *seaborn*: Diimpor sebagai “sns”, ini adalah library visualisasi data Python berdasarkan matplotlib yang menyediakan antarmuka tingkat tinggi untuk menggambar grafik statistik yang menarik dan informatif.
5. *cv2*: Ini adalah library OpenCV untuk tugas visi komputer.
6. *glob*: Digunakan untuk mengambil file/pathname yang cocok dengan pola tertentu.
7. *keras.preprocessing.image.load_img*: Digunakan untuk memuat gambar ke dalam format PIL (Python Imaging Library).
8. s*klearn.model_selection.train_test_split*: Digunakan untuk membagi array atau matriks menjadi subset train dan test secara acak.
9. *os*: Modul ini menyediakan cara portabel untuk menggunakan fungsionalitas sistem operasi yang bergantung.

3.3. Inisiasi dataset

```
1 train_images = []
2 train_labels = []
3 shape = (128,128)
4 train_path = '/content/train_zip/train'
5 for filename in os.listdir('/content/train_zip/train'):
6     if filename.split('.')[1] == 'jpg':
7         img = cv2.imread(os.path.join(train_path,filename))
8         train_labels.append(filename.split('_')[0])
9         img = cv2.resize(img,shape)
10        train_images.append(img)
11 train_labels = pd.get_dummies(train_labels).values
12 train_images = np.array(train_images)
13 x_train,x_val,y_train,y_val = train_test_split(train_images,train_labels,random_state=1)
14 test_images = []
15 test_labels = []
16 shape = (128,128)
17 test_path = '/content/test_zip/test'
18 for filename in os.listdir('/content/test_zip/test'):
19     if filename.split('.')[1] == 'jpg':
20         img = cv2.imread(os.path.join(test_path,filename))
21         test_labels.append(filename.split('_')[0])
22         img = cv2.resize(img,shape)
23         test_images.append(img)
24 test_images = np.array(test_images)
```

Berikut penjelasannya:

1. Menginisialisasi daftar kosong untuk gambar pelatihan dan pengujian serta label.
2. Menetapkan bentuk untuk mengubah ukuran gambar menjadi (128, 128).
3. Membaca gambar JPG dari direktori tertentu untuk pelatihan, mengubah ukurannya menjadi bentuk yang ditentukan, dan menambahkannya ke daftar train_images.
4. Memproses set gambar lainnya untuk pengujian dengan cara yang sama.
5. Mengonversi daftar gambar pelatihan dan pengujian menjadi array NumPy.

6. Menggunakan fungsi train_test_split (tidak ditunjukkan dalam cuplikan ini) yang kemungkinan digunakan untuk membagi data pelatihan menjadi set pelatihan dan validasi.

3.4. Mencetak bentuk dari dataset pelatihan dan validasi

```
print("x train shape :",x_train.shape)
print("x val shape :",x_val.shape)
print("y train shape :",y_train.shape)
print("y val shape :",y_val.shape)
```

penjelasan :

1. print("x train shape :",x_train.shape): Mencetak bentuk dari x_train, yang merupakan array yang berisi gambar pelatihan. Bentuk ini akan memberikan informasi tentang jumlah gambar dan dimensi setiap gambar.

2. print("x val shape :",x_val.shape): Mencetak bentuk dari x_val, yang merupakan array yang berisi gambar validasi. Bentuk ini juga memberikan informasi tentang jumlah gambar dan dimensi setiap gambar
.
3. print("y train shape :",y_train.shape): Mencetak bentuk dari y_train, yang merupakan array yang berisi label untuk gambar pelatihan. Bentuk ini akan memberikan informasi tentang jumlah label.

4. print("y val shape :",y_val.shape): Mencetak bentuk dari y_val, yang merupakan array yang berisi label untuk gambar validasi. Bentuk ini juga memberikan informasi tentang jumlah label.

3.5. Membuat model neural network

```
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D

model=Sequential()

model.add(Conv2D(filters=30,kernel_size=(3,3),padding="same",activation="relu",input_shape=(128,128,3)))
model.add(MaxPooling2D())
model.add(Conv2D(filters=30,kernel_size=(3,3),padding="same",activation="relu"))
model.add(MaxPooling2D())
model.add(Conv2D(filters=30,kernel_size=(3,3),padding="same",activation="relu"))
model.add(MaxPooling2D())
model.add(Conv2D(filters=30,kernel_size=(3,3),padding="same",activation="relu"))
model.add(MaxPooling2D())

model.add(Flatten())
model.add(Dense(206,activation="relu"))
model.add(Dropout(0.4))
model.add(Dense(103,activation="relu"))
model.add(Dropout(0.4))
model.add(Dense(4,activation="softmax"))
```

penjelasan :

1. model=Sequential(): Membuat model Sequential, yang berarti layer ditambahkan satu per satu dalam urutan.

2. model.add(Conv2D(...)): Menambahkan layer Conv2D untuk konvolusi. Parameter yang digunakan adalah jumlah filter, ukuran kernel, padding, fungsi aktivasi, dan bentuk input.

3. model.add(MaxPooling2D()): Menambahkan layer MaxPooling2D setelah setiap layer konvolusi untuk mengurangi dimensi spasial dari volume output.

4. model.add(Flatten()): Menambahkan layer Flatten untuk mengubah data input menjadi vektor 1-D.

5. model.add(Dense(...)): Menambahkan layer Dense (fully connected). Parameter yang digunakan adalah jumlah unit dan fungsi aktivasi.

6. model.add(Dropout(...)): Menambahkan layer Dropout setelah setiap layer Dense untuk mencegah overfitting dengan secara acak mengatur sejumlah unit output ke nol selama waktu pelatihan.

3.6. Train Model

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

x_train=x_train/255.0
x_val=x_val/255.0

hist=model.fit(x_train,y_train,batch_size=50,epochs=100,validation_data=(x_val,y_val))
```

penjelasan :

1. model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']): Fungsi ini digunakan untuk mengkompilasi model. Parameter yang digunakan adalah:
    - optimizer='adam': Optimizer Adam digunakan untuk meminimalkan fungsi loss.
    - loss='categorical_crossentropy': Fungsi loss ini biasanya digunakan untuk klasifikasi multikelas.
    - metrics=['accuracy']: Akurasi digunakan sebagai metrik evaluasi model.
2. x_train=x_train/255.0 dan x_val=x_val/255.0: Ini adalah langkah normalisasi data. Nilai pixel gambar biasanya berada dalam rentang 0-255. Dengan membagi nilai pixel dengan 255, kita mengubah rentang menjadi 0-1, yang lebih mudah untuk diproses oleh model.
3. hist=model.fit(x_train,y_train,batch_size=50,epochs=100,validation_data=(x_val,y_val)): Fungsi ini digunakan untuk melatih model. Parameter yang digunakan adalah:
    - x_train,y_train: Data latih dan labelnya.
    - batch_size=50: Ukuran batch yang digunakan selama pelatihan.
    - epochs=100: Jumlah epoch, atau iterasi melalui seluruh dataset latih.
    - validation_data=(x_val,y_val): Data validasi yang digunakan untuk evaluasi model setelah setiap epoch.

3.7. Memvisualisasikan loss train dan validasi

```
plt.plot(hist.history["loss"],color="green",label="Train Loss")
plt.plot(hist.history["val_loss"],color="red",label="Validation Loss")
plt.title("Loss Plot")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss Values")
plt.show()
```

penjelasan :

1. plt.plot(hist.history["loss"],color="green",label="Train Loss"): Membuat plot untuk loss pelatihan yang disimpan dalam hist.history['loss'], dengan warna hijau dan label “Train Loss”.
2. plt.plot(hist.history["val_loss"],color="red",label="Validation Loss"): Membuat plot untuk loss validasi yang disimpan dalam hist.history['val_loss'], dengan warna merah dan label “Validation Loss”.
3. plt.title("Loss Plot"): Menetapkan judul plot menjadi “Loss Plot”.
4. plt.xlabel("Number of Epochs"): Memberi label pada sumbu x sebagai “Number of Epochs”.
5. plt.ylabel("Loss Values"): Memberi label pada sumbu y sebagai “Loss Values”.
6. plt.show(): Menampilkan plot.

3.8. Memvisualisasikan akurasi train dan validasi

```
plt.plot(hist.history["accuracy"],color="black",label="Train Accuracy")
plt.plot(hist.history["val_accuracy"],color="blue",label="Validation Accuracy")
plt.title("Accuracy Plot")
plt.xlabel("Number of Epochs")
plt.ylabel("Accuracy Values")
plt.show()
```

penjelasan :

1. plt.plot(hist.history["accuracy"],color="black",label="Train Accuracy"): Membuat plot untuk akurasi pelatihan yang disimpan dalam hist.history['accuracy'], dengan warna hitam dan label “Train Accuracy”.
2. plt.plot(hist.history["val_accuracy"],color="blue",label="Validation Accuracy"): Membuat plot untuk akurasi validasi yang disimpan dalam hist.history['val_accuracy'], dengan warna biru dan label “Validation Accuracy”.
3. plt.title("Accuracy Plot"): Menetapkan judul plot menjadi “Accuracy Plot”.
4. plt.xlabel("Number of Epochs"): Memberi label pada sumbu x sebagai “Number of Epochs”

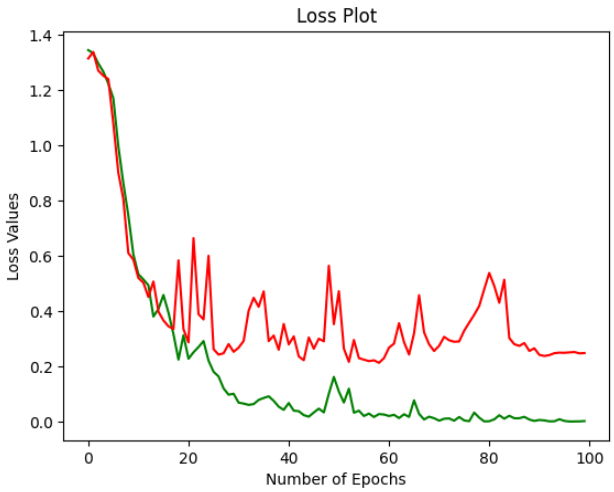
.

5. plt.ylabel("Accuracy Values"): Memberi label pada sumbu y sebagai “Accuracy Values”.

6. plt.show(): Menampilkan plot.

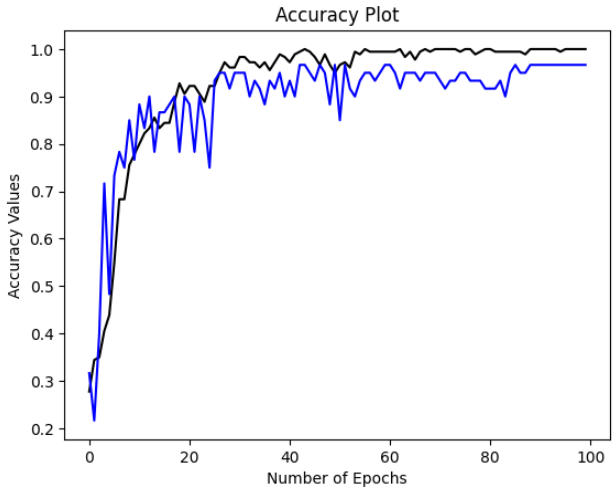
4. Hasil Eksperimen

4.1. Hasil loss train dan validasi



Grafik ini menunjukkan plot loss selama pelatihan model machine learning atau neural network sepanjang epoch. Terdapat dua kurva, merah dan hijau, yang mungkin mewakili data latihan dan validasi. Kurva merah menunjukkan fluktuasi tinggi dalam nilai loss sebelum stabil di sekitar epoch 60, sementara kurva hijau relatif lebih stabil. Ini membantu dalam memahami bagaimana model belajar dan menyesuaikan bobotnya selama proses pelatihan untuk meminimalkan loss. Kurva ini juga dapat memberikan wawasan tentang apakah model mungkin overfitting atau underfitting.

4.2. Hasil Akurasi train dan validasi



Grafik ini menampilkan akurasi model machine learning selama pelatihan. Garis biru mewakili akurasi pada setiap epoch, dengan peningkatan tajam pada awalnya dan kemudian stabil di sekitar 0.9 setelah sekitar 40 epoch. Grafik ini membantu dalam memahami bagaimana model belajar dan dapat memberikan wawasan tentang kinerja model.

4.3. Hasil Deteksi objek

Hasil dari objek jika terdapat lebih dari satu buah yang terpampang dalam gambar sebagai berikut :

Hasil jika terdapat 1 buah dalam gambar yang disajikan sebagai berikut :

banana

orange

5. Kesimpulan

Berdasarkan hasil grafik pelatihan model machine learning atau neural network yang dijelaskan, dapat ditarik beberapa kesimpulan penting. Pertama, pada grafik loss, fluktuasi tinggi pada kurva merah sebelum stabilisasi di sekitar epoch 60 menunjukkan bahwa model awalnya mengalami ketidakstabilan dalam proses pembelajaran sebelum akhirnya mencapai keadaan yang lebih konsisten. Pemahaman ini dapat membantu mengidentifikasi titik di mana model mencapai konvergensi dan mengindikasikan bahwa pelatihan model memerlukan waktu untuk mencapai tingkat optimasi yang diinginkan.

Kedua, melihat grafik akurasi, peningkatan tajam pada awalnya dan stabilisasi di sekitar 0.9 setelah sekitar 40 epoch menunjukkan bahwa model memiliki kemampuan yang baik dalam mengklasifikasikan data. Meskipun akurasi ini tinggi, masih perlu diperhatikan bahwa akurasi tidak memberikan informasi lengkap tentang kinerja model, terutama jika terjadi overfitting atau underfitting. Oleh karena itu, evaluasi lebih lanjut pada metrik lain dan pengujian pada data yang tidak terlihat mungkin diperlukan.

Secara keseluruhan, hasil grafik memberikan pandangan yang berguna tentang proses pelatihan dan kinerja model. Analisis lebih lanjut, termasuk evaluasi metrik tambahan dan pengujian pada dataset yang berbeda, dapat memberikan pemahaman lebih mendalam tentang sejauh mana model ini dapat diandalkan dan berguna dalam aplikasi praktisnya.

References

[1] Li, Chenbin, Guohua Zhan, and Zhihua Li. "News text classification based on improved Bi-LSTM-CNN." 2018 9th International Conference on Information Technology in Medicine and Education (ITME). IEEE, 2018.

[2] Jang B, Kim M, Harerimana G, Kang S-u, Kim JW. Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism. Applied Sciences. 2020; 10(17):5841.