Rancangan Machine Learning untuk Mendeteksi Lagu Plagiat

Dominggo Pratama Sidauruk[1], I Gusti Ngurah Anom Cahyadi Putra[2]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Udayana
Jalan Raya Kampus UNUD, Bukit Jimbaran, Kuta Selatan, Badung, Bali, Indonesia
[1]dominggopratama@email.com
[2] anom.cp@unud.ac.id

Abstract

Plagiarism in the music industry is a serious issue that requires advanced solutions. This research proposes a Machine Learning-based system for detecting song plagiarism by combining Convolutional Neural Network (CNN) and Dynamic Time Warping (DTW). CNN is used to extract features from the visual representation of music notations, while DTW measures the temporal distance between two sequences of notations. Experimental results show that this system provides a more accurate solution with an accuracy of 92.71%, with a dataset of 4800 data points.

Keywords: *Music plagiarism, Convolutional Neural Network (CNN), Dynamic Time Warping (DTW), plagiarism detection, music notation, machine learning*

1. Pendahuluan

Musik sebagai Bahasa universal, mampu menyampaikan emosi, cerita, dan pengalaman manusia dengan menggunakan berbagai elemen seperti melodi, harmoni, ritme, dan lirik. Dalam era modern ini, perkembangan teknologi digital telah memberikan perubahan pada cara kita mengakses, menciptakan, dan berbagi musik. Platform streaming seperti *Spotify*, *Apple Music*, dan *YouTube* telah membuka pintu bagi akses yang lebih mudah ke jutaan lagu dari seluruh dunia. Namun, seiring dengan kemudahan akses dan kolaborasi yang ditawarkan oleh teknologi, juga muncul tantangan baru, termasuk isu-isu seperti plagiarisme musik. Plagiarisme musik, di mana sebagian atau seluruh karya musik dicuri atau disalin tanpa izin dari pencipta aslinya, telah menjadi masalah yang semakin meresahkan dalam industri musik modern. Plagiarisme musik tidak hanya merugikan secara finansial bagi pencipta lagu asli, tetapi juga merugikan dalam Hak Kekayaan Intelektual (HKI) bagi pencipta lagu. Plagiarisme dalam “UUHC No 28 Tahun 2014” tidak diatur secara jelas di dalam undang-undang tersebut. plagiarisme yang diatur dalam UUHC ini hanya sebatas peniadaan nama pencipta serta tidak adanya izin dari pemegang atau pemilik hak cipta sehingga terjadi tindakan ekploitasi atau tindakan memperbanyak hasil cipta orang yang dilakukan tanpa izin [1]. Dalam upaya mengatasi tantangan ini, pengembangan teknologi yang dapat mendeteksi dan mencegah plagiarisme musik menjadi semakin penting. Salah Upaya untuk mengatasi plagiarism pada industri musik yakni menggunakan Machine learning untuk pengenalan karakter dalam notasi angka. Kemampuan untuk mengenali dan menerjemahkan notasi angka menjadi teks yang dapat diproses oleh sistem komputer dapat membantu membangun sistem deteksi plagiarisme musik yang lebih efektif dan andal. Artikel ini bertujuan untuk mengeksplorasi kemampuan Machine Learning dalam pengenalan karakter dalam notasi angka, dengan mempertimbangkan implikasi hukum terkait dengan plagiarisme musik. Fokus khusus diberikan pada penggunaan Convolutional Neural Networks (CNNs) untuk mengenali dan menerjemahkan simbol-simbol musik dalam notasi angka menjadi teks yang dapat dibaca oleh manusia.

2. Metode Penelitian

Pada penelitian ini dilakukan beberapa tahap proses penelitian untuk mendapatkan program yang dapat mengenali notasi angka pada gambar yang diberikan.

2.1. Pengumpulan Data

Langkah pertama dari penelitian ini adalah pengumpulan data. Data yang saya gunakan adalah data primer berupa gambar hasil editan saya berupa notasi angka dengan latar putih. Dataset ini berjumlah total 4,800 data gambar dalam 5 kategori (note1, note2, note3, note4, note5) yang terdiri dari data train dan data test.

2.2. *Convolutional Neural Network* (CNN)

Convolutional Neural network adalah salah satu metode Machine Learning dari pengembangan *Multi-Layer Perceptron (MLP)* yang didesain untuk mengolah data dua dimensi [2]. CNN merupakan hierarchical neural networks yang tersusun atas beberapa *convolutional layer* dan beberapa *subsampling layer* [3]. Dalam machine learning, CNN merupakan teknik yang termasuk pada *feed-forward neural network*. Pada arsitektur CNN setiap individu neuron disusun sedemikian sehingga dapat memberikan respon terhadap region yang saling overlaping pada suatu area visual [4]. Convolutional Neural Network memiliki cara kerja yang sama dengan tradisional Artificial Neural Network, mulai dari masukan awal sampai hasil akhir, setiap jaringan akan tetap menampilkan sebuah skor atau yang disebut dengan weight dan loss functions pada layer terakhir [5].

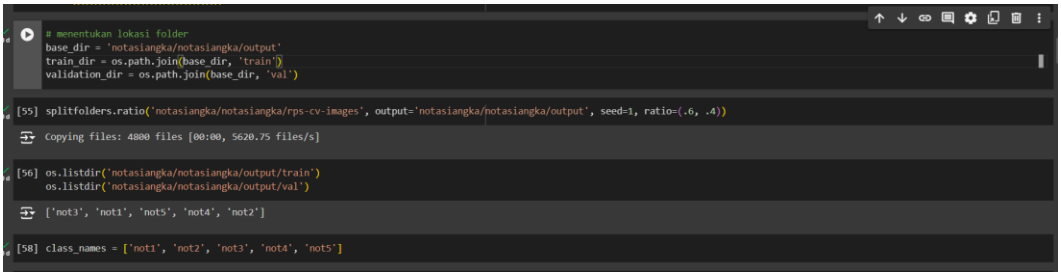
2.3. *Dynamic Time Warping* (DTW)

Dynamic Time Warping algorithm (DTW) adalah algortima yang menghitung optimal warping path antara dua waktu. Metode ini untuk menentukan kecocokan optimal antara dua deret waktu dengan beberapa pembatasan tertentu [6]. DTW mencocokkan dua sekuensial dengan menghitung tranformasi temporal sehingga keduanya dapat diselaraskan (aligned). Penyelarasan (alignment) adalah optimal jika terukur jarak kumulatif terkecil antara dua sampel yang telah diselaraskan [7]. DTW dapat membandingkan dua urutan yang memiliki panjang yang berbeda. Dalam konteks notasi angka lagu, dua lagu bisa memiliki durasi yang berbeda atau bisa dinyanyikan dengan tempo yang berbeda. DTW mampu mengatasi perbedaan panjang ini dengan cara merentangkan dan menyusutkan urutan waktu. Sebuah lagu yang sama bisa dinyanyikan atau dimainkan dengan tempo yang berbeda. DTW menyesuaikan urutan waktu sehingga bagian yang mirip dari dua urutan dapat dibandingkan secara lebih akurat, meskipun satu urutan lebih cepat atau lebih lambat dari yang lain.

2.4. Implementasi Program

Pada impelementasi program, kami membuat program sederhana yang bisa mengenali notasi angka menggunakan CNN dan program sederhana untuk mendeteksi dan menghitung seberapa mirip baris note suatu lagu dengan lagu lainnya. Pada program sederhana ini, program ini hanya bisa membaca 5 not saja (not1, not2, not3, not4 dan not5).

a. Pembagian Dataset



Gambar 1. Pembagian Dataset

Dataset dibagi menjadi dua bagian utama, yaitu data pelatihan (train) dan data validasi (validation). Pembagian dataset ini dilakukan dengan menggunakan fungsi splitfolders.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

ratio() pada library splitfolders. Data pelatihan digunakan untuk melatih model, sedangkan data validasi digunakan untuk menguji kinerja model saat pelatihan.

b. Augmentasi Data dan Normalisasi Data

```
data_train_gen = ImageDataGenerator(
    rotation_range=20,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

data_test_gen = ImageDataGenerator(
    rotation_range=20,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Gambar 2. Augmentasi Data dan Normalisasi Data

Pada tahap ini, dilakukan augmentasi data untuk memperluas dataset pelatihan dengan cara menghasilkan variasi baru dari gambar-gambar yang sudah ada. Hal ini dilakukan dengan menggunakan objek ImageDataGenerator dari library tensorflow.keras.preprocessing.image. Beberapa teknik augmentasi data yang digunakan antara lain rotasi, pergeseran, dan pembalikan gambar secara horizontal. Setelah dilakukan augmentasi data, gambar-gambar dalam dataset dinormalisasi agar memiliki rentang nilai antara 0 dan 1. Hal ini dilakukan dengan menambahkan parameter rescale=1. /255 pada objek ImageDataGenerator.

```
[60] train_generator = data_train_gen.flow_from_directory(
        train_dir, # data train
        target_size=(150, 150),  # mengubah resolusi gambar
        batch_size=32,
        class_mode='categorical', # klasifikasi lebih dari 2 kelas maka menggunakan class_mode = 'categorical'
        color_mode='rgb',
        seed=42
    )

    validation_generator = data_test_gen.flow_from_directory(
        validation_dir, # data validasi
        target_size=(150, 150),  # mengubah resolusi gambar
        batch_size=32,
        class_mode='categorical', # klasifikasi lebih dari 2 kelas maka menggunakan class_mode = 'categorical'
        color_mode='rgb',
        seed=42
    )

Found 2880 images belonging to 5 classes.
Found 1920 images belonging to 5 classes.

[61] print(train_generator.class_indices)

{'not1': 0, 'not2': 1, 'not3': 2, 'not4': 3, 'not5': 4}

[62] print(validation_generator.class_indices)

{'not1': 0, 'not2': 1, 'not3': 2, 'not4': 3, 'not5': 4}
```

Gambar 3. Generator Data

Data pelatihan dan data validasi disiapkan dalam bentuk generator data menggunakan objek flow_from_directory() dari objek ImageDataGenerator. Generator data ini akan menghasilkan batch-batch gambar yang akan digunakan secara iteratif selama proses pelatihan dan validasi model. Selanjutnya program akan menampilkan indeks kelas dari data pelatihan dan validasi.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

c. Membuat Model CNN

```
[63] model = Sequential() # deklarasi untuk jenis model

    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(2, 2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(5, activation='softmax'))

    model.summary()
```

Selanjutnya kami mendeklarasikan dan membangun model Convolutional Neural Network (CNN). Pertama-tama, kami menggunakan Conv2D untuk menambahkan lapisan konvolusi yang akan mendeteksi fitur-fitur seperti tepi, tekstur, dan pola dalam gambar. Selanjutnya kami menggunakan MaxPooling2D untuk membantu mengurangi overfitting. Lalu Flatten mengubah data dari matriks 2D menjadi vektor 1D yang kemudian bisa dihubungkan ke lapisan fully connected (Dense). Dense digunakan untuk klasifikasi multi-kelas, menghasilkan probabilitas untuk setiap kelas.

d. Compile dan Pelatihan Model

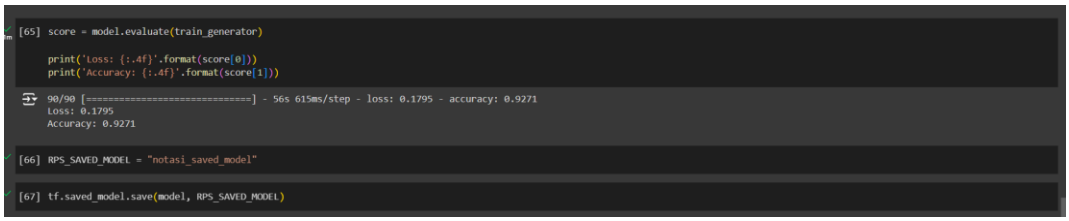
```
#compile data
model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.optimizers.Adam(),
    metrics=['accuracy']
)

#test data
model.fit(
    train_generator,
    steps_per_epoch=25,
    epochs=20,
    validation_data=validation_generator,
    validation_steps=5,
    verbose=2
)
```

Python

Setelah membangun arsitektur model, langkah berikutnya adalah mengkompilasi model. Ini termasuk menentukan fungsi loss, optimizer, dan metrik evaluasi yang akan digunakan selama pelatihan. Selanjutnya model dilatih menggunakan data yang dihasilkan oleh generator data pelatihan dan validasi. Proses pelatihan berlangsung selama 20 epoch dengan 25 batch per epoch. steps_per_epoch=25 menunjukkan jumlah batch yang dihasilkan oleh generator dalam satu epoch. epochs=20 menentukan jumlah iterasi penuh melalui dataset.

e. Evaluasi Model



```
[65] score = model.evaluate(train_generator)

    print('Loss: {:.4f}'.format(score[0]))
    print('Accuracy: {:.4f}'.format(score[1]))

90/90 [==============================] - 56s 615ms/step - loss: 0.1795 - accuracy: 0.9271
Loss: 0.1795
Accuracy: 0.9271

[66] RPS_SAVED_MODEL = "notasi_saved_model"

[67] tf.saved_model.save(model, RPS_SAVED_MODEL)
```

Setelah pelatihan, model dievaluasi pada data pelatihan untuk mendapatkan nilai loss dan akurasi. Evaluasi model memberikan metrik performa yang dapat digunakan untuk memahami seberapa baik model telah belajar dari data pelatihan. Model kemudian disimpan.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

f. Program untuk Prediksi Gambar

```python
import numpy as np
from keras.preprocessing import image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from google.colab import files
%matplotlib inline

uploaded = files.upload()

for fn in uploaded.keys():
    path = fn
    img = image.load_img(path, target_size=(150, 150))
    imgplot = plt.imshow(img)
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    class_idx = np.argmax(classes)

    result = class_names[class_idx]
    print(f"Gambar tersebut adalah notasi: {result}")
```

Program menyediakan fungsi untuk memprediksi kelas dari gambar yang diunggah oleh pengguna. Gambar diunggah dari lokal, diubah ukurannya, dan diubah menjadi array sebelum digunakan untuk prediksi. Bagian ini memungkinkan pengguna untuk menguji model dengan gambar baru. Gambar yang diunggah akan diprediksi kelasnya oleh model, memberikan hasil yang dapat divisualisasikan.

g. Program untuk Identifikasi Kemiripan Notasi Angka Sebuah Lagu dengan Lagu Yang Lain

```python
import numpy as np
from fastdtw import fastdtw
from scipy.spatial.distance import euclidean

# Fungsi untuk memuat notasi angka dari file teks
def load_notasi_angka(file_path):
    with open(file_path, 'r') as file:
        lines = file.read().strip()  # Membaca seluruh konten file sebagai satu string dan menghapus spasi ekstra
    # Mengonversi setiap karakter dalam string menjadi integer
    notasi = [int(x) for x in lines]
    return notasi

# Fungsi untuk menghitung jarak DTW antara dua urutan notasi angka
def calculate_dtw_distance(seq1, seq2):
    seq1 = [[x] for x in seq1]
    seq2 = [[x] for x in seq2]
    distance, path = fastdtw(seq1, seq2, dist=euclidean)
    return distance

# Fungsi untuk menghitung jarak maksimum antara dua urutan
def calculate_max_distance(seq1, seq2):
    max_len = max(len(seq1), len(seq2))
    max_distance_per_pair = euclidean([1], [7])  # Asumsi rentang notasi adalah 1-7
    return max_distance_per_pair * max_len

# Fungsi untuk menghitung persentase kemiripan
def calculate_similarity_percentage(seq1, seq2):
    dtw_distance = calculate_dtw_distance(seq1, seq2)
    max_distance = calculate_max_distance(seq1, seq2)
    similarity = (1 - (dtw_distance / max_distance)) * 100
    return dtw_distance, max_distance, similarity

# Contoh penggunaan
if __name__ == "__main__":
    # Memuat notasi angka dari dua file
    notasi_lagu1 = load_notasi_angka('notasi_lagu1.txt')
    notasi_lagu2 = load_notasi_angka('notasi_lagu2.txt')

# Contoh penggunaan
if __name__ == "__main__":
    # Memuat notasi angka dari dua file
    notasi_lagu1 = load_notasi_angka('notasi_lagu1.txt')
    notasi_lagu2 = load_notasi_angka('notasi_lagu2.txt')

    # Menghitung persentase kemiripan antara dua notasi lagu
    dtw_distance, max_distance, similarity_percentage = calculate_similarity_percentage(notasi_lagu1, notasi_lagu2)

    # Menampilkan hasil
    print(f"Panjang urutan notasi lagu pertama: {len(notasi_lagu1)}")
    print(f"Panjang urutan notasi lagu kedua: {len(notasi_lagu2)}")
    print(f"Jarak DTW antara dua lagu adalah: {dtw_distance}")
    print(f"Jarak maksimum yang mungkin antara dua lagu adalah: {max_distance}")
    print(f"Persentase kemiripan antara dua lagu adalah: {similarity_percentage:.2f}%")
```

p-ISSN: 2986-3929
e-ISSN: 3032-1948

Program ini dirancang untuk menghitung persentase kemiripan antara dua urutan notasi angka menggunakan metode Dynamic Time Warping (DTW). ungsi load_notasi_angka digunakan untuk memuat notasi angka dari file teks. Fungsi ini membuka file dan membaca kontennya sebagai satu string, menghapus spasi ekstra, dan mengonversi setiap karakter dalam string menjadi integer yang kemudian disimpan dalam list notasi. Fungsi calculate_dtw_distance digunakan untuk menghitung jarak DTW antara dua urutan notasi angka. Setiap elemen dalam urutan diubah menjadi list satu elemen, karena fastdtw memerlukan input dalam bentuk vektor. Fungsi ini kemudian menggunakan fastdtw untuk menghitung jarak DTW dan jalur penyelarasan optimal antara dua urutan menggunakan jarak Euclidean sebagai metrik jarak. Untuk menghitung jarak maksimum yang mungkin antara dua urutan, fungsi calculate_max_distance digunakan. Fungsi ini menentukan panjang maksimum antara dua urutan, dan menghitung jarak Euclidean maksimum antara dua notasi dalam rentang 1-7. Hasilnya adalah produk dari jarak maksimum per not dan panjang maksimum rangkaian not. emudian, fungsi calculate_similarity_percentage menghitung persentase kemiripan antara dua urutan notasi angka. Fungsi ini menggunakan calculate_dtw_distance untuk mendapatkan jarak DTW dan calculate_max_distance untuk mendapatkan jarak maksimum antara dua urutan. Persentase kemiripan dihitung dengan rumus (1 - (dtw_distance / max_distance)) * 100, di mana jarak DTW yang lebih kecil menghasilkan kemiripan yang lebih tinggi. Fungsi ini mengembalikan jarak DTW, jarak maksimum, dan persentase kemiripan. Bagian terakhir dari program adalah program utama. Program utama memuat notasi angka dari dua file teks menggunakan load_notasi_angka, menghitung persentase kemiripan antara dua notasi lagu menggunakan calculate_similarity_percentage, dan menampilkan hasil berupa panjang urutan notasi, jarak DTW, jarak maksimum, dan persentase kemiripan antara dua lagu.

3. Hasil dan Diskusi

Berdasarkan percobaan dengan program sederhana diatas, kami mendapatkan:

3.1. Akurasi Setiap Epoch

```
Epoch 1/20
25/25 - 46s - loss: 1.6250 - accuracy: 0.2013 - val_loss: 1.6141 - val_accuracy: 0.1688 - 46s/epoch - 2s/step
Epoch 2/20
25/25 - 44s - loss: 1.6109 - accuracy: 0.2062 - val_loss: 1.6086 - val_accuracy: 0.2625 - 44s/epoch - 2s/step
Epoch 3/20
25/25 - 42s - loss: 1.5798 - accuracy: 0.2750 - val_loss: 1.5937 - val_accuracy: 0.2375 - 42s/epoch - 2s/step
Epoch 4/20
25/25 - 43s - loss: 1.5023 - accuracy: 0.3363 - val_loss: 1.5023 - val_accuracy: 0.3500 - 43s/epoch - 2s/step
Epoch 5/20
25/25 - 42s - loss: 1.3828 - accuracy: 0.4075 - val_loss: 1.2726 - val_accuracy: 0.5063 - 42s/epoch - 2s/step
Epoch 6/20
25/25 - 44s - loss: 1.1888 - accuracy: 0.5200 - val_loss: 1.2791 - val_accuracy: 0.4125 - 44s/epoch - 2s/step
Epoch 7/20
25/25 - 43s - loss: 0.9842 - accuracy: 0.5950 - val_loss: 0.8256 - val_accuracy: 0.6750 - 43s/epoch - 2s/step
Epoch 8/20
25/25 - 42s - loss: 0.8316 - accuracy: 0.6525 - val_loss: 0.7993 - val_accuracy: 0.6562 - 42s/epoch - 2s/step
Epoch 9/20
25/25 - 41s - loss: 0.6610 - accuracy: 0.7325 - val_loss: 0.7580 - val_accuracy: 0.6625 - 41s/epoch - 2s/step
Epoch 10/20
25/25 - 41s - loss: 0.5893 - accuracy: 0.7437 - val_loss: 0.5510 - val_accuracy: 0.7688 - 41s/epoch - 2s/step
Epoch 11/20
25/25 - 41s - loss: 0.4718 - accuracy: 0.8250 - val_loss: 0.4925 - val_accuracy: 0.7937 - 41s/epoch - 2s/step
Epoch 12/20
25/25 - 41s - loss: 0.3909 - accuracy: 0.8325 - val_loss: 0.3460 - val_accuracy: 0.8562 - 41s/epoch - 2s/step
Epoch 13/20
25/25 - 41s - loss: 0.3386 - accuracy: 0.8500 - val_loss: 0.4337 - val_accuracy: 0.8062 - 41s/epoch - 2s/step
Epoch 14/20
25/25 - 43s - loss: 0.3466 - accuracy: 0.8475 - val_loss: 0.3294 - val_accuracy: 0.8562 - 43s/epoch - 2s/step
Epoch 15/20
25/25 - 42s - loss: 0.3270 - accuracy: 0.8600 - val_loss: 0.2744 - val_accuracy: 0.8625 - 42s/epoch - 2s/step
Epoch 16/20
25/25 - 40s - loss: 0.2632 - accuracy: 0.9025 - val_loss: 0.2912 - val_accuracy: 0.9062 - 40s/epoch - 2s/step
Epoch 17/20
25/25 - 43s - loss: 0.2472 - accuracy: 0.9050 - val_loss: 0.2580 - val_accuracy: 0.8938 - 43s/epoch - 2s/step
Epoch 18/20
25/25 - 41s - loss: 0.2271 - accuracy: 0.8988 - val_loss: 0.2290 - val_accuracy: 0.9062 - 41s/epoch - 2s/step
Epoch 19/20
25/25 - 42s - loss: 0.2049 - accuracy: 0.9137 - val_loss: 0.1492 - val_accuracy: 0.9563 - 42s/epoch - 2s/step
Epoch 20/20
25/25 - 42s - loss: 0.1657 - accuracy: 0.9513 - val_loss: 0.2144 - val_accuracy: 0.9000 - 42s/epoch - 2s/step
<keras.src.callbacks.History at 0x7f83efe1df00>
```

p-ISSN: 2986-3929
e-ISSN: 3032-1948

Berdasarkan gambar diatas, kami mendapatkan nilai accuracy 0,9513. Adapun nilai accuracy setiap epoch yakni 0.2013, 0.2062, 0.2750, 0.3363, 0.4075, 0.5200, 0.5950, 0.6525, 0.7325, 0.7437, 0.8250, 0.8325, 0.8500, 0.8475, 0.8600, 0.9025, 0.9050, 0.8988, 0.9137, dan 0.9513. Walaupun sempat turun ke 0.8988, bisa dibilang rancangan machine learning kami dengan dataset yang kami gunakan, mampu berkembang dengan baik.

```
90/90 [==============================] - 56s 615ms/step - loss: 0.1795 - accuracy: 0.9271
Loss: 0.1795
Accuracy: 0.9271
```

	Model	Data_Train	Data_Valid	Epochs	Batch_Size	Loss	Accuracy
0	CNN	2880	1920	20	32	0.17947	0.927083

Setelah melewati evaluasi model, accuracy machine learning kami mampu menyentuh aaccuracy 92%.

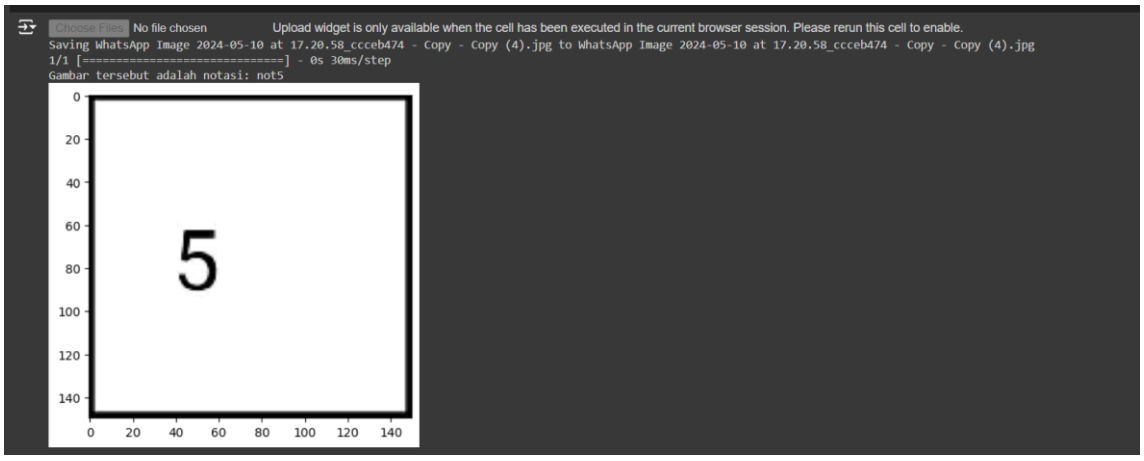
3.2. Matriks yang dihasilkan

```
60/60 [==============================] - 40s 656ms/step
Confusion Matrix
[[81 79 74 75 75]
 [94 61 82 67 80]
 [73 75 84 76 76]
 [74 77 81 97 55]
 [92 77 89 55 71]]
```

Berikut adalah matriks yang dihasilkan dari machine learning kami berdasarkan dataset yang kami gunakan.

3.3. Percobaan Menginput Gambar

Karena disini kami menggunakan machine learning untuk identifikasi notasi angka, sehingga machine learning harus mampu mengidentifikasi notasi angka yang kami berikan. Walaupun begitu machine learning ini hanya mampu mengidentifikasi 1 notasi angka saja.



Bisa dilihat pada gambar diatas, machine learning mampu mengidentifikasi notasi angka walaupun hanya 1 saja pada 1 gambar.

3.4. Percobaan Menginput Baris Notasi Angka

Pada percobaan ini kami akan menggunakan potongan baris notasi angka dari Ibu Kita Kartini (443465653132345) dengan pembanding notasi angka yang mirip dengan notasi angkanya (443466654132245).

```
[Running] python -u "c:\DTW\DTWnotasi.py"
Panjang urutan notasi lagu pertama: 15
Panjang urutan notasi lagu kedua: 15
Jarak DTW antara dua lagu adalah: 3.0
Jarak maksimum yang mungkin antara dua lagu adalah: 90.0
Persentase kemiripan antara dua lagu adalah: 96.67%
```

Bisa dilihat hasil dari program kami menunjukkan lagu yang menjadi pembanding mirip dengan potongan lagu Ibu Kita Kartini sebesar 96,67%. Bisa dilihat jarak DTW antara 2 lagu tersebut adalah 3. Jarak disini merupakan jumlah seluruhnya jarak sebuah nada dengan nada lainnya. Misal not 4 dibandingkan dengan not 1, maka not 4 memiliki jarak sebanyak 3 dari not 1. Contohnya ada baris not yakni 566432 dibandingkan dengan 564421. Program akan membandingan satu per satu not dengan pasangan not nya. 5 dipasangkan dengan 5, 6 dipasangkan dengan 6, 6 dipasangkan dengan 4, 4 dipasangkan dengan 4, 3 dipasangkan dengan 2 dan 2 dipasangkan dengan 1. Maka jarak DTW seluruhnya adalah 2 + 1 + 1 = 4. Sehingga diperoleh jarak DTW seluruhnya adalah 4. Untuk jarak maksimum antara 2 jarak diperoleh dengan mengalikan jarak DTW dengan Panjang urutan notasi dikalikan dengan rentang tangga nada (7-1) sehingga didapatkan Jarak Maksimumnya adalah 15 x 6 = 90. Presentasi kemiripannya didapatkan dengan rumus

$$Persentasi\ kemiripan = \left(\frac{Jarak\ DTW}{Jarak\ maksimum}\right) X\ 100\% \quad (1)$$

Sehingga didapatkan bahwa presentasi kemiripannya adalah 96,67%.

4. Kesimpulan

Adapun kesimpulan dari artikel ini antara lain:

a. Akurasi tertinggi yang dicapai dari dataset setelah proses kompilasi dan pelatihan model adalah sebesar 95.13%. Hal ini menunjukkan bahwa model Convolutional Neural Network (CNN) yang kami rancang mampu mengenali pola dalam data dengan tingkat akurasi yang sangat tinggi selama fase pelatihan.
b. Setelah melalui tahap evaluasi model, akurasi yang diperoleh adalah sebesar 92.71%, dengan jumlah data latih sebanyak 2880 dan data valid sebanyak 1920. Angka ini menunjukkan bahwa model tetap menunjukkan kinerja yang kuat saat diuji dengan data yang tidak digunakan selama pelatihan, meskipun terdapat sedikit penurunan akurasi.
c. Rancangan kami saat ini masih berada dalam tahap program sederhana yang berfokus pada pengenalan notasi angka untuk mendeteksi plagiat musik. Kami berharap bahwa artikel ini dapat memberikan kontribusi yang berarti dalam pengembangan lebih lanjut dari teknologi machine learning untuk identifikasi musik plagiat. Kami mengharapkan pembaca yang tertarik dapat melanjutkan pengembangan ini hingga tahap yang lebih kompleks, termasuk identifikasi langsung dari data audio, sehingga teknologi ini dapat digunakan dalam aplikasi nyata di industri musik.

Daftar Pustaka

[1] Yoga, P., Putra, U., Agung, A., & Indrawati, S. (2021). Perlindungan Hukum Terhadap Praktik Plagiarisme Karya Seni Lagu/Musik Berdasarkan Undang-Undang Nomor 28 Tahun 2014. Jurnal Kertha Negara, 9(12), 1027–1038. [2]N. I. Kurniati, A. Rahmatulloh, and D. Rahmawati, "Perbandingan Performa Algoritma Koloni Semut Dengan Algoritma Genetika – Tabu Search Dalam Penjadwalan Kuliah," *Comput. Eng. Sci. Syst. J.*, vol. 4, no. 1, p. 17, 2019, doi: 10.24114/cess. v4i1.11387.

[2] Wairata, C. R., Swedia, E. R., & Cahyanti, M. (2021). Pengklasifikasian Genre Musik Indonesia Menggunakan Convolutional Neural Network. Sebatik, 25(1). https://doi.org/10.46984/sebatik.v25i1.1286

[3] Hakim, D. M., & Rainarli, E. (2019). Convolutional Neural Network untuk Pengenalan Citra Notasi Musik. Techno.Com, 18(3), 214–226. https://doi.org/10.33633/tc.v18i3.2387

[4] Ciresan CD, et al. 2011. Flexible, High Performance Convolutional Neural Networks for Image Classification. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI'11), Vol. Two, Pages 1237-1242

[5] Namruddin, R., Mirfan, & Irfandi. (2023). Klasifikasi Kesegaran Buah Apel Menggunakan Metode Convolutional Neural Network (CNN) Berbasis Android. Prosiding SISFOTEK.

[6] Archamadi, A., Magdalena, R., & Ramatryana, I. N. A. (2016). Analisis dan Simulasi Identifikasi Judul Lagu dari Senandung Manusia Menggunakan Ekstraksi Ciri Dct (discrete Cosine Transform). EProceedings of Engineering, 3(3), 4575–4584. http://libraryeproceeding.telkomuniversity.ac.id/index.php/engineering/article/view/2894

[7] Iqbal, M., Supriyati, E., & Listyorini, T. (2015). Implemetasi Offline Pengenalan Sistem Isyarat Bahasa Indonesia Menggunakan Metode Dynamic Time Warping Pada Perangkat Android. Simetris : Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer, 6(2), 391. https://doi.org/10.24176/simet.v6i2.477

p-ISSN: 2986-3929
e-ISSN: 3032-1948

Halaman ini sengaja dibiarkan kosong