PROJECT WORKSHOP SISTEM PENDUKUNG KEPUTUSAN & PRAKTIKUM KOMPUTASI BERGERAK

SIGNATURE VERIFICATION SYSTEM

Dosen pengampu:

Dr. Ir Prima Kristalina MT

Disusun oleh:

Rakha Farid Mansur (2423600001)

Raffi Mahya Pratama (2423600002)

Farrello Axondiaz Kitdawhanda (2423600016)

Attila Artharizqia Syahbanizar (2423600022)

PROGRAM STUDI TEKNOLOGI REKAYASA INTERNET

JURUSAN TEKNIK ELEKTRO

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2025

Signature Verification System

1. Tujuan

- Memeriksa apakah suatu tanda tangan asli atau tidak
- Dapat menggunakan model SNN
- Dapat menggunakan model CNN

2. Dasar Teori

2.1. Verifikasi Tanda Tangan

Verifikasi tanda tangan online adalah salah satu metode autentikasi biometrik yang dilakukan dengan menganalisis **pola dinamis dari proses penulisan tanda tangan** secara langsung. Data yang dikumpulkan meliputi parameter seperti kecepatan menulis, tekanan pena, arah goresan, dan waktu antar titik. Sistem ini umumnya menggunakan perangkat seperti tablet pen atau stylus yang mampu merekam informasi tersebut secara real-time.

Berbeda dengan verifikasi offline yang hanya mengandalkan gambar statis, verifikasi online memiliki keunggulan dalam hal keakuratan karena mencakup **dimensi waktu dan gerakan**. Hal ini membuatnya lebih sulit untuk dipalsukan, karena seseorang tidak hanya harus meniru bentuk tanda tangan, tetapi juga dinamika penulisannya.

Dalam proyek ini, verifikasi dilakukan menggunakan pendekatan machine learning yang mampu mengenali pola dinamis dari data tanda tangan, kemudian membedakan antara tanda tangan asli dan palsu.

2.2. Penerapan Model Machine Learning pada Verifikasi Tanda Tangan

Machine learning adalah cabang dari kecerdasan buatan (Artificial Intelligence) yang memungkinkan komputer untuk belajar dari data dan membuat keputusan atau prediksi tanpa harus diprogram secara eksplisit. Dalam konteks verifikasi tanda tangan, machine learning digunakan untuk mempelajari perbedaan antara tanda tangan asli dan palsu berdasarkan fitur-fitur visual dari citra.

Salah satu pendekatan yang digunakan dalam proyek ini adalah **deep learning**, yaitu teknik machine learning yang menggunakan arsitektur jaringan saraf tiruan berlapis (neural networks). Secara khusus, proyek ini menerapkan gabungan dari **Convolutional Neural Network (CNN)** dan **Siamese Neural Network (SNN)** sebagai model untuk melakukan proses pembelajaran dan perbandingan antar tanda tangan. Untuk mengoptimalkan proses pelatihan model ini, digunakan pula fungsi kerugian khusus yang disebut **Contrastive Loss**.

2.3. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan jenis jaringan saraf tiruan yang sangat efektif dalam memproses data berbentuk citra. CNN mampu mengekstraksi pola dan fitur penting dari gambar, seperti bentuk, garis, dan tekstur, tanpa perlu proses ekstraksi fitur secara manual.

Dalam proyek verifikasi tanda tangan, CNN digunakan sebagai **feature**

extractor, yaitu untuk mengubah gambar tanda tangan menjadi vektor numerik (embedding) yang merepresentasikan karakteristik visual tanda tangan tersebut. Vektor inilah yang nantinya digunakan dalam proses perbandingan oleh model machine learning.

2.3.1 Layer Convolutional Neural Network (CNN)

Lapisan Konvolusi (Convolutional Layer)

Lapisan Konvolusi adalah fondasi utama dari arsitektur CNN yang bertanggung jawab untuk proses ekstraksi fitur. Pada lapisan ini, sebuah operasi konvolusi matematis diterapkan pada data input (misalnya gambar atau *feature map* dari lapisan sebelumnya) menggunakan sebuah **kernel** atau **filter**. Kernel ini adalah sebuah matriks bobot berukuran kecil yang meluncur (berkonvolusi) di seluruh dimensi spasial dari input. Pada setiap posisi, dilakukan operasi *element-wise multiplication* antara nilai pada kernel dengan nilai pada bagian input yang ditutupinya, kemudian hasilnya dijumlahkan untuk menghasilkan satu nilai tunggal pada **feature map** keluaran. Setiap kernel dirancang untuk mendeteksi fitur spesifik, seperti tepi, sudut, atau tekstur. Dengan menggunakan beberapa kernel secara bersamaan, lapisan ini mampu mengekstrak beragam fitur dari input secara paralel, di mana setiap *feature map* yang dihasilkan merepresentasikan lokasi dari fitur tertentu pada gambar.

Lapisan Aktivasi (Activation Layer)

Segera setelah operasi konvolusi, sebuah fungsi aktivasi non-linear diterapkan pada setiap elemen dari *feature map*. Fungsi yang paling umum digunakan dalam arsitektur CNN modern adalah **Rectified Linear Unit (ReLU)**, yang didefinisikan sebagai $f(x)=\max(0,x)$. Fungsi ini akan mengubah semua nilai negatif menjadi nol dan mempertahankan nilai positif. Tujuan utama dari lapisan aktivasi adalah untuk memperkenalkan non-linearitas ke dalam model. Tanpa non-linearitas, tumpukan beberapa lapisan konvolusi hanya akan setara dengan satu operasi linear tunggal, yang secara signifikan membatasi kemampuan model untuk mempelajari pola dan hubungan yang kompleks dalam data.

Lapisan Pooling (Pooling Layer)

Lapisan Pooling, atau sering disebut juga lapisan *subsampling*, bertujuan untuk mengurangi dimensi spasial (lebar dan tinggi) dari *feature map* secara progresif. Proses ini berfungsi untuk mengurangi jumlah parameter dan komputasi dalam jaringan, sehingga mengontrol *overfitting*. Jenis pooling yang paling umum adalah **Max Pooling**, di mana sebuah jendela (misalnya berukuran 2x2) meluncur di sepanjang *feature map* dan pada setiap langkahnya hanya mengambil nilai maksimum dari area tersebut. Selain mengurangi dimensi, lapisan ini juga memberikan tingkat **invarian translasi** pada fitur yang terdeteksi. Artinya, model menjadi lebih toleran terhadap pergeseran kecil pada posisi fitur, karena yang terpenting adalah keberadaan fitur tersebut dalam suatu wilayah, bukan lokasi piksel pastinya.

Lapisan Fully-Connected (Dense Layer)

Setelah fitur-fitur diekstraksi dan ukurannya direduksi oleh lapisan-lapisan sebelumnya, data perlu diratakan menjadi sebuah vektor satu dimensi menggunakan **Lapisan Flatten**. Vektor ini kemudian menjadi input bagi **Lapisan Fully-Connected (FC)** atau **Dense Layer**. Pada lapisan ini, setiap neuron terhubung ke semua neuron dari lapisan sebelumnya. Lapisan FC berfungsi untuk melakukan penalaran tingkat

tinggi dengan mempelajari kombinasi non-linear dari fitur-fitur yang telah diekstraksi. Pada tahap akhir arsitektur, lapisan inilah yang bertanggung jawab untuk menghasilkan output final, baik berupa probabilitas kelas untuk tugas klasifikasi maupun vektor *embedding* akhir seperti pada arsitektur Siamese Network untuk tugas verifikasi.

2.4. Siamese Neural Network (SNN)

Siamese Neural Network (SNN) adalah salah satu arsitektur dalam deep learning yang digunakan untuk mengukur kemiripan antara dua input. SNN terdiri dari dua jaringan identik (dalam hal struktur dan bobot) yang digunakan untuk memproses dua input secara paralel. Dalam konteks verifikasi tanda tangan, input tersebut berupa dua gambar tanda tangan—satu sebagai referensi dan satu sebagai input yang ingin diverifikasi.

Masing-masing input akan diproses oleh CNN dan menghasilkan vektor embedding. Kemudian, kedua embedding tersebut dibandingkan menggunakan fungsi jarak (seperti Euclidean distance) untuk menentukan seberapa mirip kedua tanda tangan tersebut. Jika jarak antar embedding kecil, maka sistem menganggap tanda tangan itu asli, dan sebaliknya.

SNN sangat cocok untuk kasus **pembelajaran berbasis perbandingan (comparison-based learning)** dan bekerja baik dalam skenario di mana data per kelas sangat terbatas.

2.5. Contrastive Loss dalam Pelatihan Model

Untuk melatih model SNN agar mampu membedakan tanda tangan asli dan palsu, digunakan fungsi kerugian (loss function) yang disebut **Contrastive Loss**. Fungsi ini bertujuan untuk meminimalkan jarak antara pasangan tanda tangan yang asli, serta memaksimalkan jarak antara pasangan tanda tangan yang tidak cocok hingga melewati ambang batas tertentu (margin). Dengan fungsi ini, model dilatih agar mampu mengelompokkan tanda tangan asli dalam ruang fitur yang berdekatan, dan menjauhkan tanda tangan palsu dari titik tersebut.

2.6. Flask sebagai Media Implementasi Model Machine Learning

Flask adalah framework web berbasis Python yang digunakan untuk membangun aplikasi web ringan dan fleksibel. Dalam proyek ini, Flask berperan sebagai **penghubung antara model machine learning dengan antarmuka pengguna (frontend)**. Melalui Flask, pengguna dapat mengunggah gambar tanda tangan, yang kemudian akan diproses oleh model untuk menghasilkan hasil verifikasi.

Model yang telah dilatih sebelumnya disimpan menggunakan library seperti joblib atau pickle, lalu dimuat kembali (load) saat aplikasi web berjalan. Flask menyediakan rute (route) untuk menangani proses input, pemanggilan model, prediksi, dan pengiriman hasil kembali ke pengguna dalam bentuk yang mudah diakses melalui browser.

3. Langkah-Langkah

1. Import Library

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Input, Lambda
from tensorflow.keras import backend as K
from sklearn.model_selection import train_test_split
import cv2
import zipfile
import os
from pathlib import Path
from tensorflow.keras.layers import BatchNormalization, Dropout
```

2. Load & Preprocess Dataset

```
zip_path = "archive.zip"
unzip_path = "cedar7"

# Unzip dataset
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(unzip_path)

base_path = Path(unzip_path)


def load_images_from_folder(folder, label):
    images = []
    for filename in sorted(Path(folder).glob("*.png")):
        img = cv2.imread(str(filename), cv2.IMREAD_GRAYSCALE)
        if img is not None:
            img = cv2.resize(img, (220, 155))
            img = img.astype('float32') / 255.0
            img = np.expand_dims(img, axis=-1)
            images.append((img, label))
    return images
```

```
print(sorted(base_path.iterdir()))
```

```
users = sorted(base_path.iterdir())

tes = []

for user in users:
    genuine = load_images_from_folder(user / "full_org", 1)
    forged = load_images_from_folder(user / "full_forg", 0)

    # Positive tes (genuine-genuine)
    for i in range(len(genuine)-1):
        tes.append((genuine[i][0], genuine[i+1][0], 1))

    # Negative tes (genuine-forged)
    for i in range(min(len(genuine), len(forged))):
        tes.append((genuine[i][0], forged[i][0], 0))

x1 = np.array([pair[0] for pair in tes])
x2 = np.array([pair[1] for pair in tes])
y = np.array([pair[2] for pair in tes])
```

```
x1_train, x1_val, x2_train, x2_val, y_train, y_val = train_test_split(x1, x2, y, test_size=0.2, random_state=42)
```

3. Membangun CNN dasar

```
import tensorflow.keras.backend as K

def l2_normalize(x):
    return K.l2_normalize(x, axis=1)

def build_base_cnn(input_shape):
    model = Sequential([
        Conv2D(64, (10, 10), activation='relu', input_shape=input_shape),
        MaxPooling2D(),

        Conv2D(128, (7, 7), activation='relu'),
        MaxPooling2D(),

        Conv2D(128, (4, 4), activation='relu'),
        MaxPooling2D(),

        Conv2D(256, (4, 4), activation='relu'),
        Flatten(),

        Dense(2048),
        Lambda(l2_normalize)
    ])
    return model
```

4. Membangun Model Siamese dengan Contrastive Loss

```
def euclidean_distance(vectors):
    x, y = vectors
    sum_square = K.sum(K.square(x - y), axis=1, keepdims=True)
    return K.sqrt(K.maximum(sum_square, K.epsilon()))

def contrastive_loss(y_true, y_pred, margin=1.0):
    y_true = tf.cast(y_true, tf.float32)
    square_pred = K.square(y_pred)
    margin_square = K.square(K.maximum(margin - y_pred, 0))
    return K.mean(y_true * square_pred + (1 - y_true) * margin_square)

input_shape = (155, 220, 1)
base_cnn = build_base_cnn(input_shape)

input_a = Input(shape=input_shape)
input_b = Input(shape=input_shape)

encoded_a = base_cnn(input_a)
encoded_b = base_cnn(input_b)

distance = Lambda(euclidean_distance)([encoded_a, encoded_b])
model = Model(inputs=[input_a, input_b], outputs=distance)
```

5. Compile dan Train model

```
model.compile(loss=contrastive_loss, optimizer='adam')
model.fit([x1_train, x2_train], y_train,
          validation_data=([x1_val, x2_val], y_val),
          batch_size=16, epochs=50)
```

6. Prediksi contoh tanda tangan

```python
def predict_pair(img1, img2):
    img1 = np.expand_dims(img1, axis=0)
    img2 = np.expand_dims(img2, axis=0)
    pred = model.predict([img1, img2])

    # Tampilkan kedua gambar yang dibandingkan
    fig, axs = plt.subplots(1, 2, figsize=(6, 3))
    axs[0].imshow(img1[0].squeeze(), cmap='gray')
    axs[0].set_title("Image 1")
    axs[0].axis('off')

    axs[1].imshow(img2[0].squeeze(), cmap='gray')
    axs[1].set_title("Image 2")
    axs[1].axis('off')

    plt.show()
    print(f"Euclidean Distance: {pred[0][0]}")
    return pred[0][0]

# Example usage:
hasil= predict_pair(x1_val[5], x2_val[5])
print(hasil)
if hasil > 0.5:
  print("palsu")
else:
  print("asli")
```

```python
np.all(x1_val[0] == x1_val[1])
```

7. Tes data tanda tangan baru

```python
def preprocess_signature_image(img_path):
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (220, 155))
    img = img.astype('float32') / 255.0
    img = np.expand_dims(img, axis=-1)  # channel
    return img
```

```python
def verify_signature(new_img, reference_images, threshold=0.5):
    distances = []
    for ref in reference_images:
        dist = predict_pair(ref, new_img)
        distances.append(dist)

    avg_distance = np.mean(distances)
    print(f"Average Distance: {avg_distance}")

    if avg_distance < threshold:
        print("✅ VERIFIKASI: TANDA TANGAN ASLI")
    else:
        print("❌ VERIFIKASI: TANDA TANGAN PALSU")
    return avg_distance
```

```python
new_img = preprocess_signature_image("frd.png")

ref_paths = [
    "1.png",
    "2.png",
    "3.png",
    "4.png",
    "5.png"
]

ref_imgs = [preprocess_signature_image(p) for p in ref_paths]

verify_signature(new_img, ref_imgs, threshold=0.6)
```

8. Tes tanda tangan palsu

```
new_img = preprocess_signature_image("palsu.png")

ref_paths = [
    "1.png",
    "2.png",
    "3.png",
    "4.png",
    "5.png"
]

ref_imgs = [preprocess_signature_image(p) for p in ref_paths]

verify_signature(new_img, ref_imgs, threshold=0.6)
```

9. Evaluasi model

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix
import seaborn as sns

# Predict distances untuk data validasi
y_pred_val = model.predict([x1_val, x2_val])
y_pred_binary = (y_pred_val < 0.6).astype("int32")  # Threshold bisa disesuaikan

# Evaluasi metrik
acc = accuracy_score(y_val, y_pred_binary)
prec = precision_score(y_val, y_pred_binary)
rec = recall_score(y_val, y_pred_binary)
f1 = f1_score(y_val, y_pred_binary)
roc_auc = roc_auc_score(y_val, -y_pred_val)

print("=== Evaluation Metrics ===")
print(f"Accuracy     : {acc:.4f}")
print(f"Precision    : {prec:.4f}")
print(f"Recall       : {rec:.4f}")
print(f"F1 Score     : {f1:.4f}")
print(f"ROC AUC      : {roc_auc:.4f}")

# Confusion matrix
cm = confusion_matrix(y_val, y_pred_binary)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Forged", "Genuine"], yticklabels=["Forged", "Genuine"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

10. Visualisasi distribusi jarak

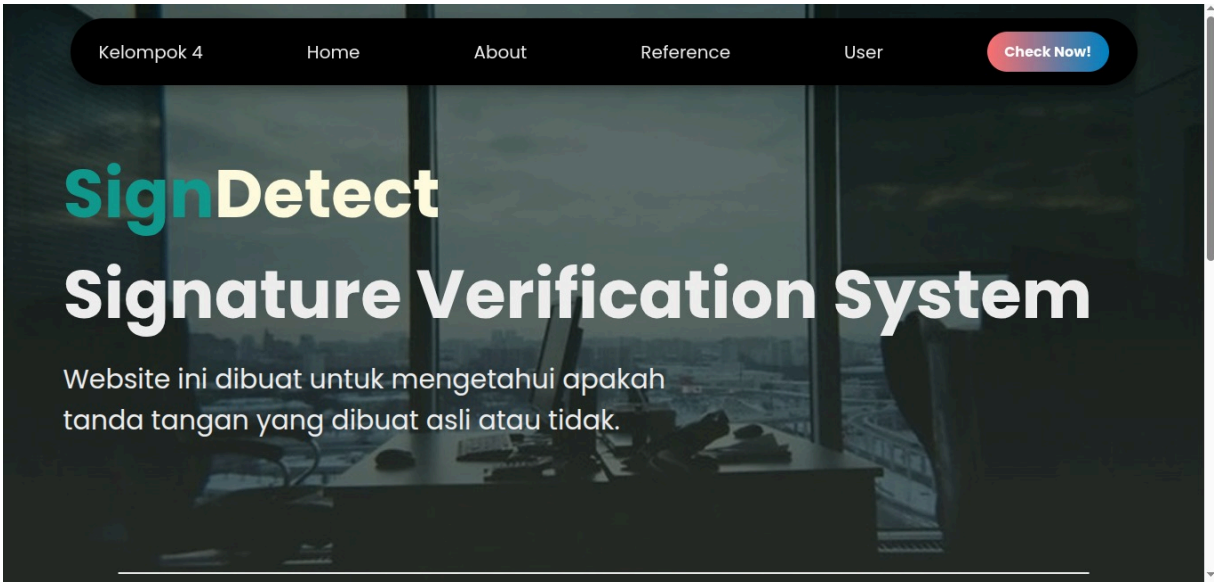
```
# Distribusi jarak
genuine_dist = y_pred_val[y_val == 1]
forged_dist = y_pred_val[y_val == 0]

plt.figure(figsize=(8, 4))
plt.hist(genuine_dist, bins=30, alpha=0.6, label="Genuine-Genuine")
plt.hist(forged_dist, bins=30, alpha=0.6, label="Genuine-Forged")
plt.axvline(0.5, color='red', linestyle='--', label='Threshold')
plt.title("Distance Distribution")
plt.xlabel("Euclidean Distance")
plt.ylabel("Count")
plt.legend()
plt.show()
```

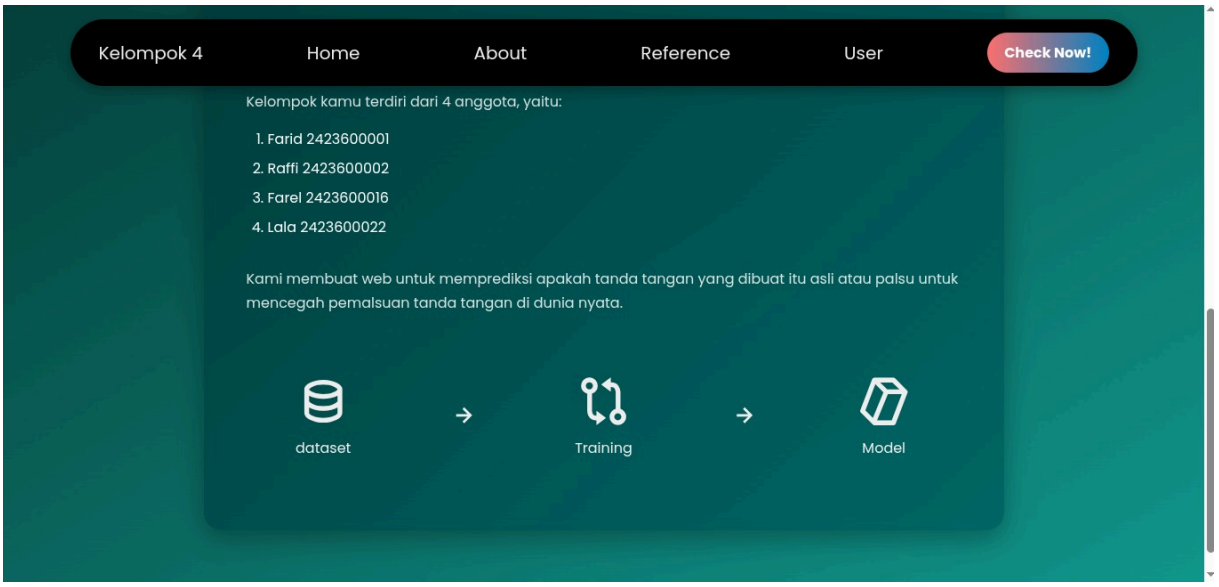
```
# # # Simpan dalam format .h5
# model.save('projek_verifikasi_ttd.h5')
```

4. Tampilan WEB

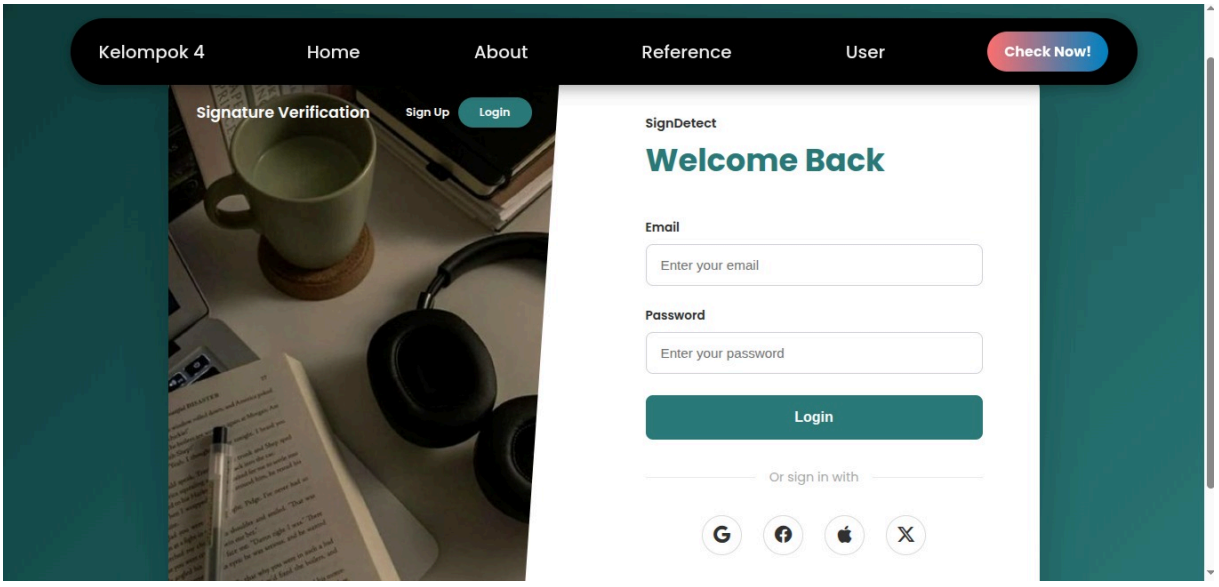
4.1 Home Page



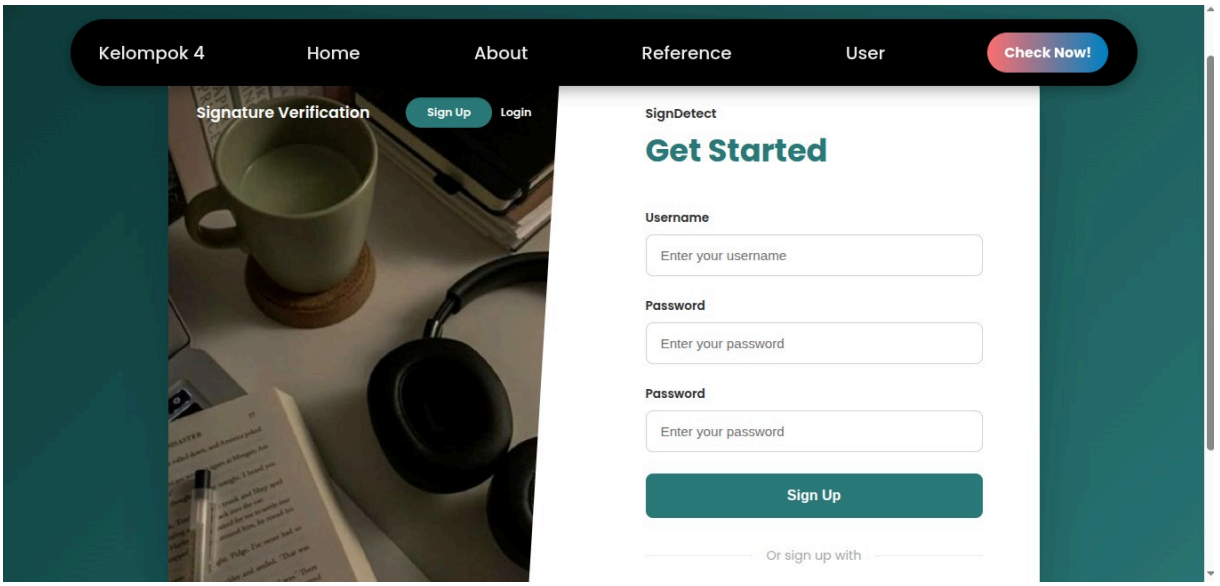
4.2 Home Page (About)



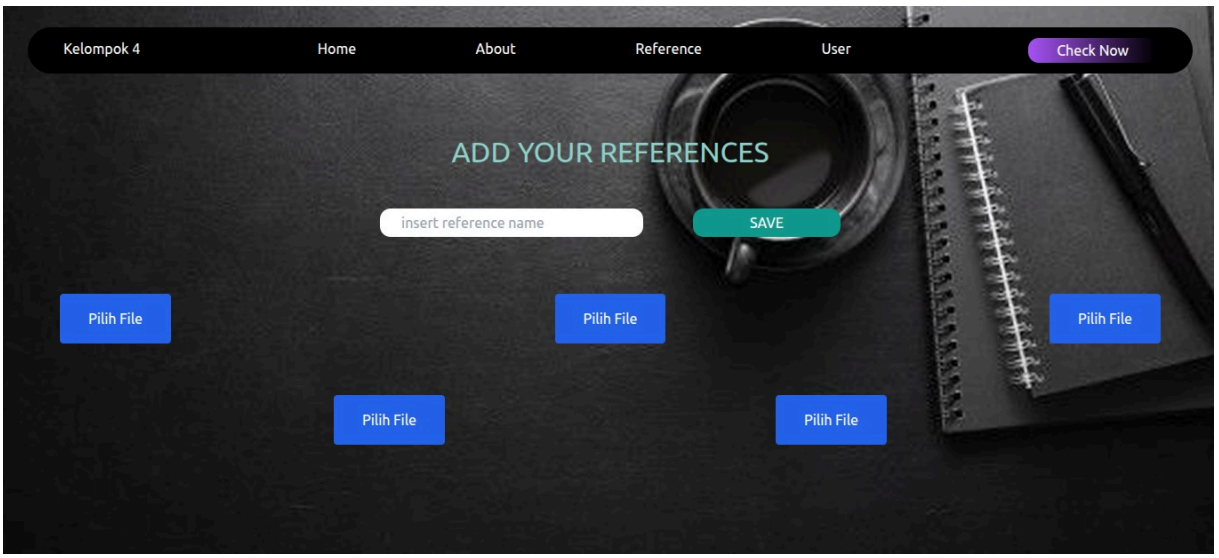
4.3 Login Page



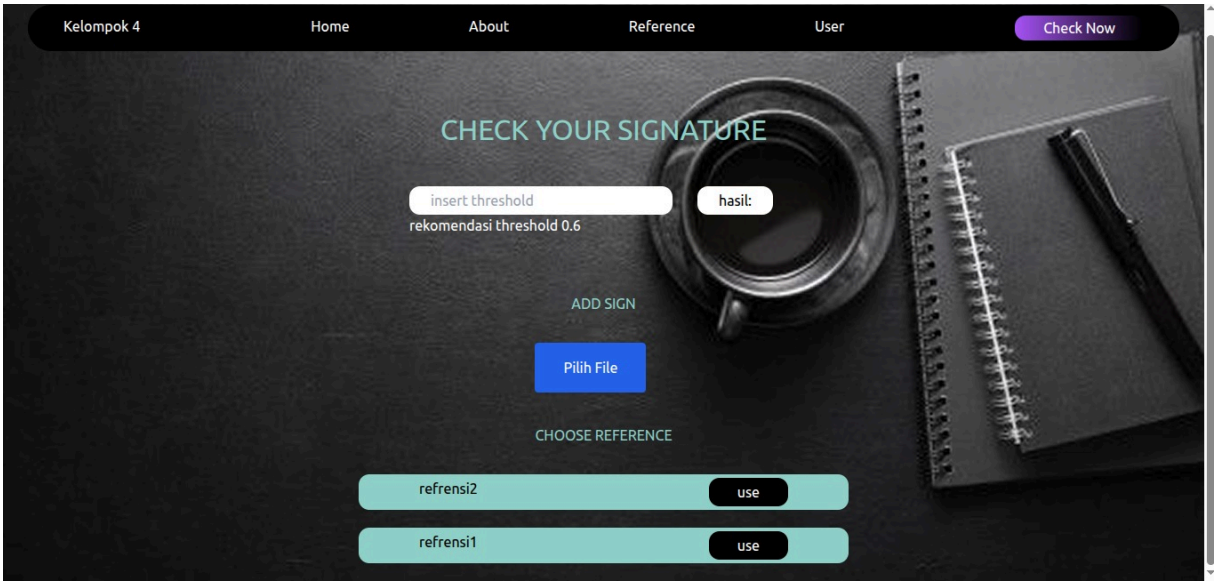
4.4 Sign up page



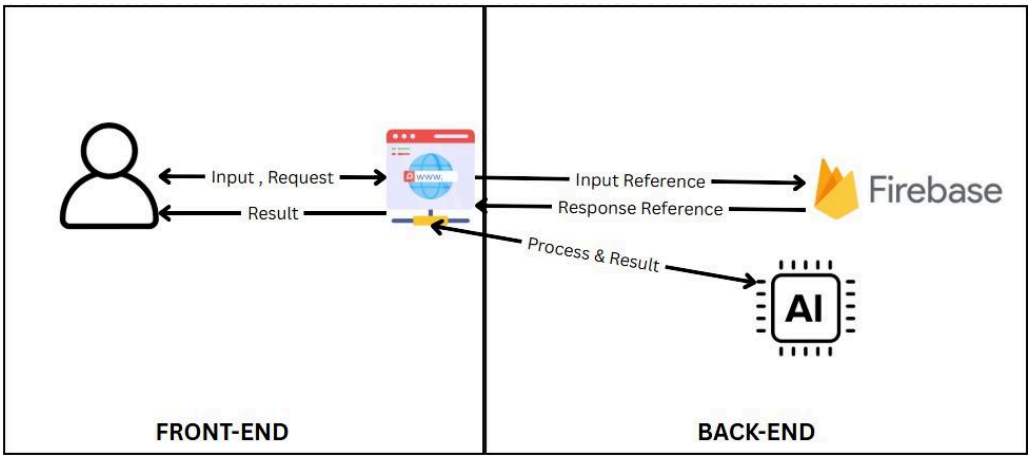
4.5 Add References Page



4.6 Signature Check Page



5. UserFlow



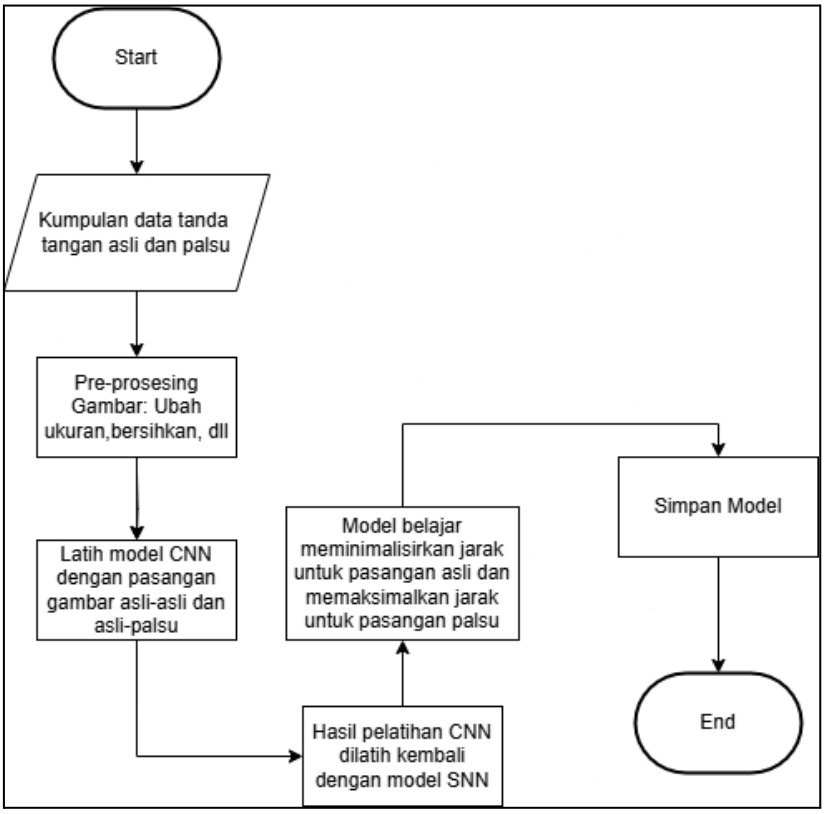
Gambar diatas merupakan representasi user flow atau alur interaksi pengguna dalam sistem verifikasi tanda tangan berbasis AI yang dibagi menjadi dua bagian utama, yaitu Front-End dan Back-End. Di sisi Front-End, pengguna berinteraksi langsung dengan antarmuka web, yang memungkinkan mereka untuk memberikan input berupa gambar tanda tangan dan melakukan permintaan verifikasi. Input tersebut kemudian dikirimkan ke sistem untuk diproses lebih lanjut.

Setelah menerima permintaan dari pengguna, sistem front-end akan mengirimkan referensi yang dibutuhkan ke layanan Firebase, yang berperan sebagai penyimpanan database tanda tangan referensi. Firebase akan merespons dengan mengirimkan data referensi yang sesuai, seperti gambar tanda tangan asli milik pengguna yang tersimpan sebelumnya. Data referensi ini kemudian dikirim dari front-end ke bagian Back-End untuk diproses oleh model kecerdasan buatan (AI).

Di sisi Back-End, komponen AI — yang telah dilatih menggunakan model Siamese Neural Network (SNN) dan CNN — akan menerima data input (tanda tangan baru) dan referensi (tanda tangan asli dari Firebase). Sistem AI akan melakukan proses perbandingan dan analisis kemiripan tanda tangan dengan menghitung jarak antar embedding. Hasil dari proses ini berupa prediksi apakah tanda tangan tersebut asli atau palsu. Setelah proses selesai, hasil prediksi akan dikirim kembali ke bagian front-end, kemudian ditampilkan kepada pengguna dalam bentuk hasil verifikasi.

6. Flowchart

6.1. Flowchart Training Model

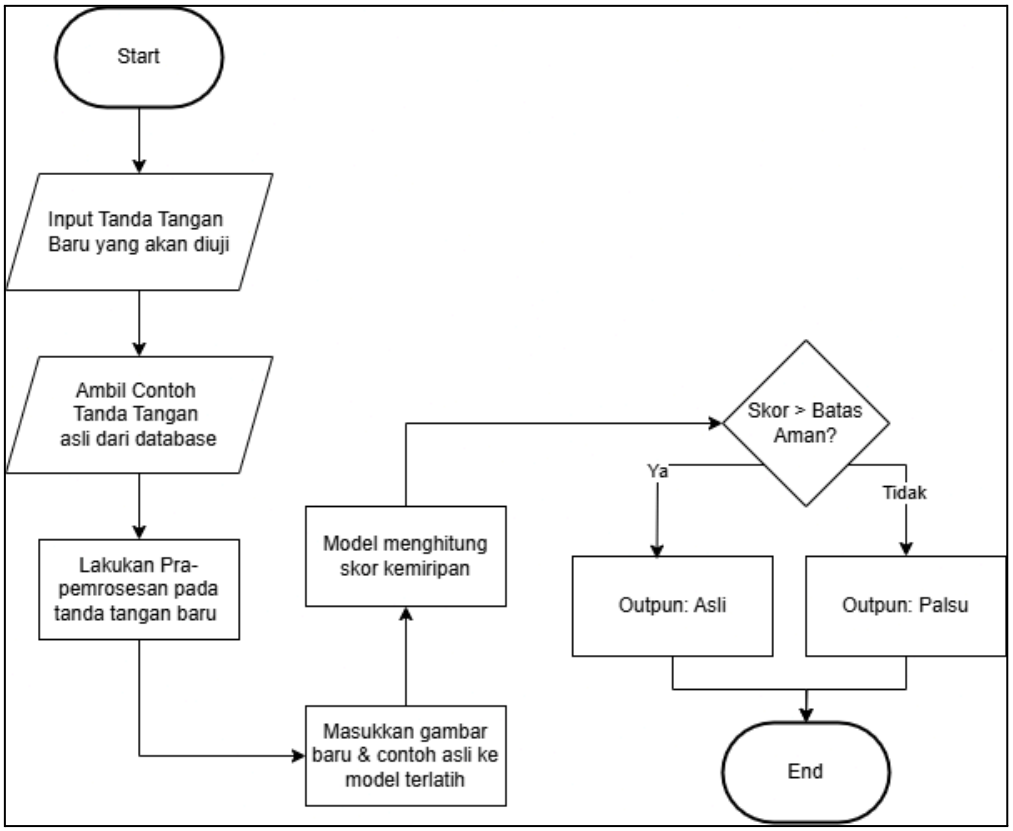


Flowchart di atas menggambarkan alur pelatihan sistem verifikasi tanda tangan berbasis deep learning, yang menggunakan kombinasi arsitektur Convolutional Neural Network (CNN) dan Siamese Neural Network (SNN). Proses dimulai dengan pengumpulan dataset yang terdiri dari gambar tanda tangan asli dan palsu. Data ini kemudian diproses melalui tahapan pra-pemrosesan, seperti mengubah ukuran gambar, mengubah ke format grayscale, normalisasi piksel, serta pembersihan noise apabila diperlukan. Pra-pemrosesan ini penting agar semua gambar berada dalam format yang seragam dan dapat diproses secara optimal oleh model.

Setelah proses pra-pemrosesan, data digunakan untuk melatih model CNN. Pada tahap ini, CNN menerima pasangan gambar, baik pasangan asli-asli maupun asli-palsu. Tujuan dari pelatihan CNN adalah mengekstraksi fitur visual penting dari setiap gambar tanda tangan dan mengubahnya menjadi representasi vektor (embedding). Embedding yang dihasilkan dari CNN kemudian digunakan dalam arsitektur Siamese Neural Network (SNN), yang dilatih untuk membandingkan dua embedding dan menghitung jarak kemiripan antara keduanya.

Model SNN dilatih menggunakan fungsi kerugian contrastive loss. Fungsi ini membuat model belajar untuk meminimalkan jarak antara pasangan tanda tangan yang berasal dari orang yang sama, dan memaksimalkan jarak untuk pasangan yang berasal dari orang yang berbeda. Dengan demikian, model dapat mengenali apakah dua gambar tanda tangan serupa atau tidak berdasarkan jarak antar embedding-nya. Setelah proses pelatihan selesai dan model mencapai performa yang diinginkan, model disimpan ke dalam file agar dapat digunakan kembali dalam proses verifikasi tanda tangan baru. Tahap ini menandai akhir dari proses pelatihan model.

6.2. Flowchart Pengujian



Proses dimulai ketika pengguna menginputkan tanda tangan baru ke dalam sistem, baik melalui perangkat input digital seperti tablet pen atau dengan mengunggah gambar hasil digitalisasi tanda tangan. Setelah menerima input tersebut, sistem akan mengambil beberapa contoh tanda tangan asli dari pengguna yang sama yang telah tersimpan di dalam basis data. Contoh tanda tangan asli ini akan digunakan sebagai referensi untuk proses perbandingan.

Selanjutnya, sistem melakukan tahap pra-pemrosesan terhadap gambar tanda tangan baru dan gambar-gambar referensi. Tahapan ini mencakup proses resize citra ke ukuran standar, konversi ke format grayscale, normalisasi nilai piksel, dan penyesuaian channel agar sesuai dengan format input yang dibutuhkan oleh model. Setelah diproses, masing-masing gambar akan dimasukkan ke dalam model CNN yang bertugas mengekstraksi fitur atau karakteristik penting dari tanda tangan. Output dari CNN adalah vektor fitur (embedding) yang merepresentasikan masing-masing gambar tanda tangan.

Pasangan embedding dari tanda tangan baru dan setiap referensi akan dihitung jaraknya menggunakan fungsi Euclidean Distance. Nilai jarak ini menunjukkan seberapa mirip dua tanda tangan: semakin kecil jaraknya, semakin tinggi kemiripannya. Sistem kemudian menghitung rata-rata dari semua jarak tersebut dan membandingkannya dengan nilai ambang batas (threshold) yang telah ditentukan, misalnya 0.5 atau 0.6. Jika nilai rata-rata jarak lebih kecil dari threshold, maka sistem menyimpulkan bahwa tanda tangan baru kemungkinan besar asli. Sebaliknya, jika nilainya melebihi threshold, maka tanda tangan tersebut dianggap palsu. Akhir dari proses adalah menampilkan hasil verifikasi kepada pengguna berupa pernyataan “tanda tangan asli” atau “tanda tangan palsu”.

7. ANALISA

Hasil percobaan yang dilakukan pada proyek verifikasi tanda tangan ini menunjukkan bahwa model Siamese Neural Network (SNN) yang dibangun dengan arsitektur Convolutional Neural Network (CNN) sebagai feature extractor mampu mengenali pola kemiripan antara tanda tangan dengan cukup baik. Data diolah menjadi pasangan gambar asli-asli dan asli-palsu, yang selanjutnya digunakan untuk melatih model dengan fungsi kerugian Contrastive Loss. Proses pelatihan berlangsung selama 50 epoch, dan nilai loss menunjukkan penurunan yang signifikan terutama pada 20 epoch pertama, yang mengindikasikan bahwa model berhasil belajar membedakan tanda tangan berdasarkan jarak antar embedding. Loss training turun dari 0.398 ke sekitar 0.10 pada akhir epoch, sementara nilai loss validasi juga menunjukkan konsistensi dan tidak mengalami overfitting secara signifikan.

Model diuji untuk memprediksi kemiripan antara dua gambar menggunakan fungsi predict_pair(). Pada kasus uji coba, pasangan tanda tangan asli menghasilkan jarak Euclidean di bawah 0.5, sedangkan pasangan palsu menunjukkan jarak di atas 1.0. Fungsi verifikasi lebih lanjut dengan membandingkan gambar baru terhadap lima referensi asli memperkuat akurasi sistem; rata-rata jarak untuk tanda tangan asli adalah 0.266, sedangkan untuk tanda tangan palsu mencapai 1.149. Ini menunjukkan bahwa threshold 0.6 yang digunakan sebagai batas keputusan cukup efektif dalam memisahkan dua kelas.

Evaluasi model menggunakan data validasi menghasilkan metrik performa yang baik, yaitu akurasi sebesar 82,58%, precision sebesar 80,41%, recall 86,99%, dan F1-score sebesar 83,57%. Nilai ROC AUC yang mencapai 0.9087 menegaskan kemampuan model dalam membedakan dua kelas dengan sangat baik. Visualisasi distribusi jarak juga memperlihatkan pemisahan yang jelas antara pasangan asli-asli dan asli-palsu, yang memperkuat bukti bahwa model telah belajar representasi yang efektif dari data.

8. KESIMPULAN

Berdasarkan hasil percobaan dan evaluasi, dapat disimpulkan bahwa sistem verifikasi tanda tangan online yang dikembangkan menggunakan arsitektur Siamese Neural Network dengan CNN sebagai ekstraktor fitur mampu membedakan tanda tangan asli dan palsu secara efektif. Model menunjukkan performa yang stabil selama pelatihan dan tidak mengalami overfitting, serta menghasilkan nilai loss yang rendah baik pada data latih maupun validasi. Proses verifikasi terhadap gambar tanda tangan baru juga menunjukkan bahwa model dapat membuat keputusan yang akurat berdasarkan jarak kemiripan, dengan threshold 0.6 yang terbukti efektif. Metrik evaluasi seperti akurasi, precision, recall, F1-score, dan ROC AUC semuanya menunjukkan performa yang tinggi, dan didukung oleh distribusi visual jarak yang jelas antara tanda tangan asli dan palsu. Dengan demikian, sistem ini layak untuk dijadikan prototipe awal dalam pengembangan sistem autentikasi tanda tangan digital.