

## BAB IV

### HASIL DAN PEMBAHASAN

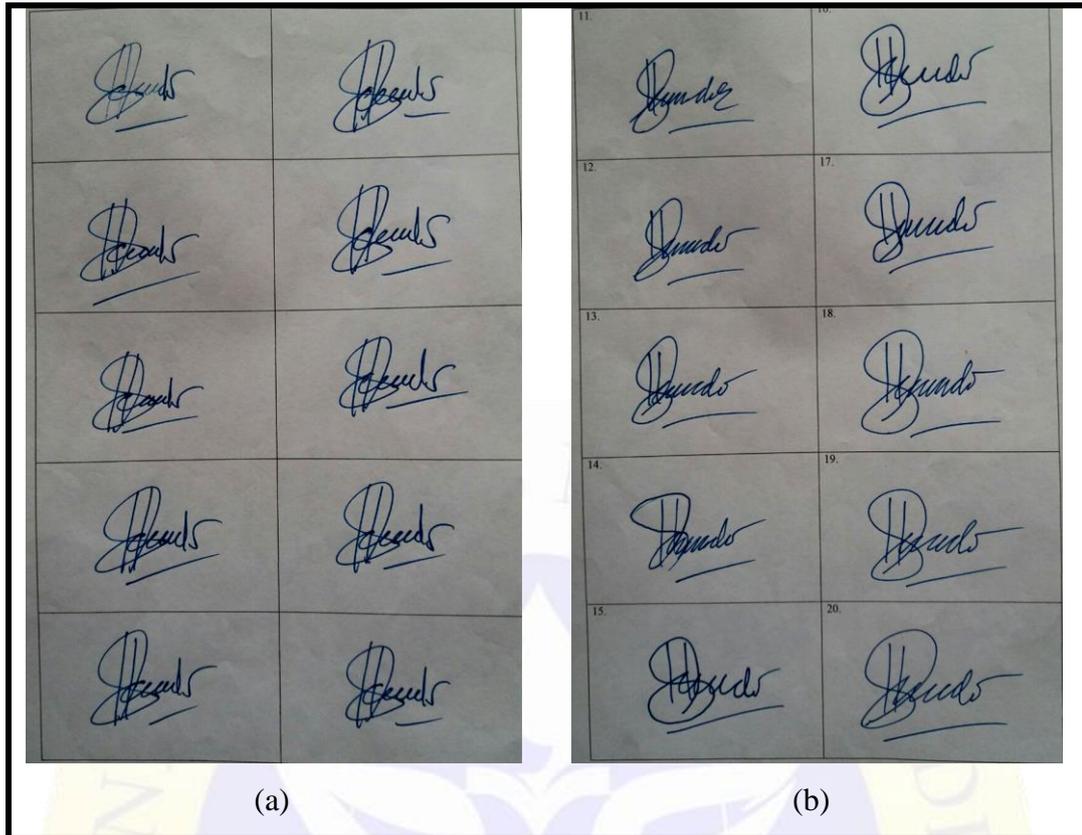
#### 4.1. Penyajian Data Uji Coba

Pada bab ini akan dijelaskan hasil dari penelitian mengenai penerapan metode *Siamese Network* pada verifikasi tanda tangan. Adapun langkah-langkah dari penelitian yang telah dipaparkan pada bab sebelumnya yang akan diimplementasikan pada bab ini. Langkah-langkah tersebut meliputi hasil dari pengumpulan *dataset*, hasil pre-processing, implementasi dengan metode *Siamese Network*, uji coba, dan penarikan kesimpulan dari hasil implementasi metode tersebut.

##### 4.1.1. Pengumpulan *Dataset*

*Dataset* tanda tangan asli pada penelitian ini menggunakan data gambar tanda tangan jajaran pimpinan Fakultas Teknik Universitas Nurul Jadid sebanyak 7 orang responden yaitu tanda tangan Kaprodi Elektro, Kaprodi Teknik informatika, Kaprodi Sistem Informasi, Kaprodi Teknologi Informasi, Kaprodi Rekayasa Perangkat Lunak, Sekertaris Program studi Elektro & Informatika, dan Dekan Fakultas Teknik, masing-masing terdapat 7 gambar sehingga total terapat 70 gambar tanda tangan asli. *Dataset* tanda tangan palsu pada penelitian ini menggunakan data gambar tanda tangan dari mahasiswa Universitas Nurul Jadid sebanyak 7 orang responden, masing-masing *dataset* terdapat 20 gambar sehingga total terapat 140 gambar. Dari 14 responden didapatkan jumlah total terdapat 210 gambar tanda tangan. Gambar tanda tangan tersebut kemudian dikelompokkan menjadi data *training* dan data *testing*. Sebanyak 147 gambar *training* dan 63 gambar *testing*.

Keseluruhan *dataset* yang digunakan dalam penelitian ini dapat dilihat pada Lampiran 1. *Dataset* harus melalui tahapan *pre-processing* karena pada tahap tersebut akan dilakukan proses *scan* gambar untuk lebih mempertegas pola tanda tangan yang ada, dari warna awal gambar yang memiliki keberagaman warna menjadi *grayscale*. Proses tersebut akan mempermudah kerja dalam implementasinya. Pada Gambar 4.1 merupakan beberapa contoh *dataset* di mana gambar awal masih berwarna.



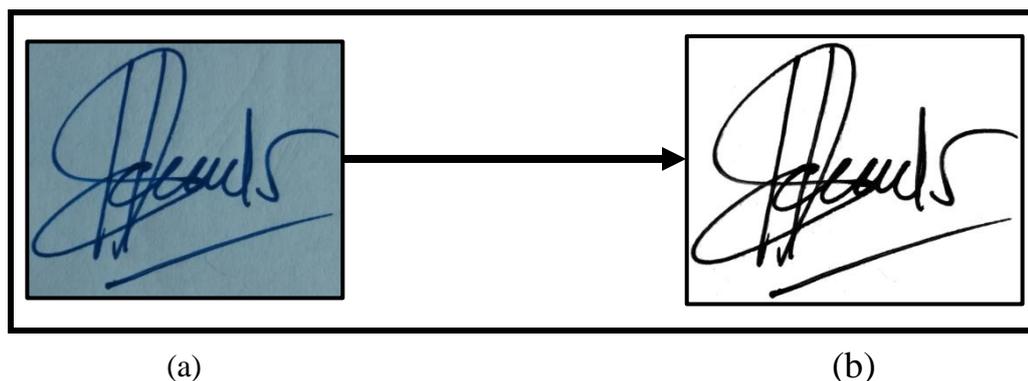
**Gambar 4. 1.** Beberapa Contoh *Dataset* (a) Tanda Tangan Asli (b) Tanda Tangan Palsu

#### 4.1.2. Pre-Processing

Dilakukan tahap pre-procecing agar pada tahapan implementasi dengan menggunakan metode *Siamese Network* lebih efektif. Berikut ini merupakan tahapan-tahapan *per-processing* yang dilakukan dalam penelitian ini:

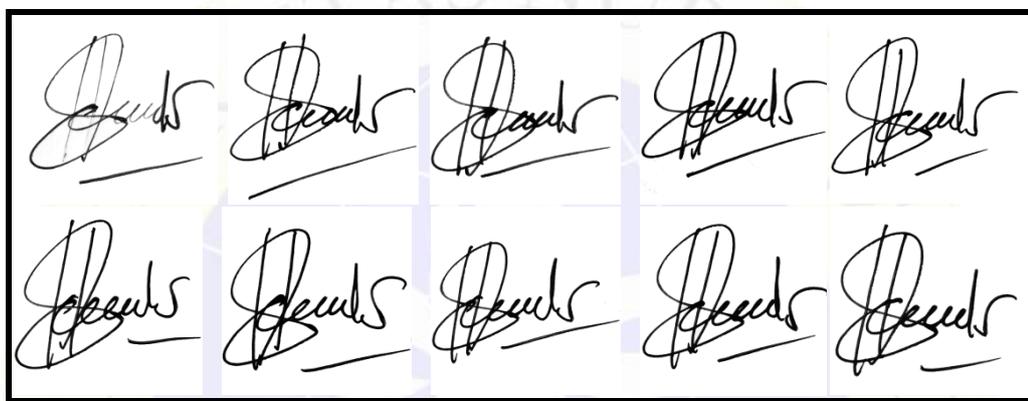
##### a. *Scan* Gambar

proses *scan* gambar dilakukan untuk lebih mempertegas pola tanda tangan yang ada, sehingga gambar yang dihasilkan lebih mudah untuk di implementasikan pada tahap implementasi metode *Siamese Network*, proses *scan* gambar menggunakan website *Online Cam Scanner*, kemudian gambar disimpan dengan format *\*png*. Gambar 4.2 merupakan (a) gambar sebelum dilakukan proses *scan* gambar dan gambar (b) sesudah dilakukan proses *scan* gambar.



**Gambar 4. 2.** Proses *Scan* Gambar

Gambar 4.3 merupakan hasil dari satu folder yang dilakukan proses *scan* gambar



**Gambar 4. 3.** Hasil Proses *Scan* Gambar

b. *Resize* Gambar

Pada tahap ini akan dilakukan proses pengaturan ulang *size* gambar dari gambar yang memiliki ukuran awal sebesar 1,71 MB dengan resolusi 3096 x 3096 pixel menjadi gambar dengan ukuran sebesar 14 KB dengan resolusi 150 x 150 pixel. Proses *Resize* gambar menggunakan *script Python*, lebih lengkap dapat dilihat pada Lampiran 7 dan Lampiran 8. Langkah pada *Resize* gambar yaitu:

- Definisi Sumber dan Target

Mendefinisikan sumber gambar yang akan diproses dan mendefinisikan target output gambar. Dari gambar awal selanjutnya gambar tersebut akan diproses untuk diatur ulang *size*-nya. Gambar dengan 14 folder, yang terdiri dari 7 folder tanda tangan asli, dan 7 folder tanda tangan palsu yang masing-masing foldernya terdapat 10 gambar untuk folder tanda tangan asli, dan 20 gambar untuk folder tanda tangan palsu. Segmen Program 4.1. merupakan

*script Python* yang menunjukkan sumber dan target pada folder *training* yang ada dalam folder *png4* dan *output* target gambar pada folder *training* yang ada dalam folder *hasil\_resize*.

#### Segmen Program 4. 1. Sumber dan Target Gambar *Resize*

```
root_folder = '/content/drive/MyDrive/dataa/png4/training'
folders = [os.path.join(root_folder, x) for x in
(' /content/drive/MyDrive/dataa/png4/training/001',
' /content/drive/MyDrive/dataa/png4/training/002',
' /content/drive/MyDrive/dataa/png4/training/003',
' /content/drive/MyDrive/dataa/png4/training/004',
' /content/drive/MyDrive/dataa/png4/training/005',
' /content/drive/MyDrive/dataa/png4/training/006',
' /content/drive/MyDrive/dataa/png4/training/007')]
all_images = [img for folder in folders for img in (folder)]
tgt_base_path = "/content/drive/MyDrive/dataa/ hasil_resize
/training"
```

- Proses *Resize* Gambar

Proses *resize* gambar mengubah ukuran gambar dari gambar yang memiliki ukuran awal sebesar 1,71 MB dengan resolusi 3096 x 3096 pixel menjadi gambar dengan ukuran sebesar 14 KB dengan resolusi 150 x 150 pixel. Setiap gambar akan diubah dengan menggunakan *script Python* kemudian akan disimpan dengan format *\*png*. Segmen Program 4.2. merupakan *script* yang digunakan untuk melakukan proses *resize* gambar.

#### Segmen Program 4. 2. Proses *Resize* Gambar

```
jdx = 0
for filename in os.listdir(src_sub_path):
    current_img = Image.open(src_sub_path + '/' + filename)
    target_path = os.path.join(tgt_sub_path, "%s-%03d.png" %
(cur_path, jdx+1))
    print('Working on image: ' + os.path.splitext(filename)[0])
    print(f'Format: {current_img.format}, Size: {current_img.size},
Mode: {current_img.mode}')
    print(target_path)

    #Resize gambar dan menyimpan hasil ke target penyimpanan
    img = current_img.resize((150,150))
    img.save(target_path)
    jdx += 1
```

c. Membuat File CSV

Setelah tahapan *resize* gambar selesai dilakukan, kemudian proses selanjutnya *input* nama gambar dan folder penyimpanan *dataset* menggunakan *Microsoft Excel* dan file akan disimpan dengan format *\*csv*. Terdapat 2 file dengan format *\*csv* yang dibuat, yaitu file *\*csv* untuk data *training*, dan file *csv\** untuk data *testing*, dilampirkan pada Lampiran 6. File *\*csv* untuk data *training* berisi 980 kemungkinan gambar dari 14 *dataset*, yang terdiri dari 7 *dataset* tanda tangan asli yang masing-masing *dataset* terdapat 7 gambar dan 7 *dataset* tanda tangan palsu yang masing-masing *dataset* terdapat 14 gambar, sehingga total terdapat 147 gambar tanda tangan yang dibuat kemungkinannya dalam file *\*csv* data *training*, kemudian file *\*csv* untuk data *testing* berisi 168 kemungkinan gambar dari 14 *dataset*, yang terdiri dari 7 *dataset* tanda tangan asli yang masing-masing *dataset* terdapat 3 gambar dan 7 *dataset* tanda tangan palsu yang masing-masing *dataset* terdapat 6 gambar, sehingga total terdapat 63 gambar yang dibuat kemungkinannya dalam file *\*csv* data *testing*. Tabel 4.1 merupakan contoh file *\*csv* yang digunakan.

**Tabel 4. 1.** File CSV

001/001_01.png,001/001_02.png,0
001/001_01.png,001/001_03.png,0
001/001_01.png,001/001_04.png,0
001/001_01.png,001/001_05.png,0
001/001_01.png,001/001_06.png,0
001/001_01.png,001/001_07.png,0
001/001_01.png,001_duplikat/0001_1.png,1
001/001_01.png,001_duplikat/0001_2.png,1
001/001_01.png,001_duplikat/0001_3.png,1
001/001_01.png,001_duplikat/0001_4.png,1
001/001_01.png,001_duplikat/0001_5.png,1
001/001_01.png,001_duplikat/0001_6.png,1

4.1.3. Implementasi dengan Metode *Siamese Network*

Implementasi dengan menggunakan metode *Siamese Network* dilakukan dengan membuat script Python. Pada implementasinya terdapat beberapa tahapan yang dilakukan.

#### a. Tahapan Persiapan Data

Pada tahapan ini data *training* dipersiapkan untuk diproses dengan metode *Siamese Network*. Segmen Program 4.3 Mendefinisikan lokasi penyimpanan *dataset* dan file *\*csv* yang digunakan dalam program.

#### Segmen Program 4. 3. Sumber dan Target Persiapan Data

```
training_dir="/content/drive/MyDrive/dataa/dataset_150x150/training"
training_csv="/content/drive/MyDrive/dataa/csv/trainingscv.csv"
testing_csv="/content/drive/MyDrive/dataa/csv/testingcsv.csv"
testing_dir="/content/drive/MyDrive/dataa/dataset_150x150/testing"
```

Selanjutnya menuju ke proses cek data *\*csv* untuk data *training* dan data *testing* yang dilakukan pada Segmen Program 4.4, bahwa 1 menunjukkan kemungkinan gambar tanda tangan palsu dan 0 menunjukkan kemungkinan gambar tanda tangan asli, di tahap ini juga dilakukan pengecekan jumlah kemungkinan gambar yang ada dalam data *\*csv* baik untuk data *training* maupun data *testing*.

#### Segmen Program 4. 4. Pengecekan Data CSV

```
#Data Training CSV
df_train=pd.read_csv(training_csv)
df_train.sample(50)

#Data Training CSV
df_test=pd.read_csv(testing_csv)
df_test.sample(50)

#Cek jumlah kemungkinan pada file CSV Training
df_train.shape
#Cek jumlah kemungkinan pada file CSV Testing
df_test.shape
```

Tahapan selanjutnya adalah membuat *Custom Pythorch* untuk menghasilkan sepasang gambar kemungkinan, nilai 0 untuk pasangan kemungkinan gambar tanda tangan asli dan 1 untuk pasangan kemungkinan gambar tanda tangan palsu, pada tahap ini juga dilakukan proses pengembalian gambar 1, gambar 2, dan label, untuk menentukan apakah sepasang gambar

memiliki nilai 1 atau nilai 0, *scrip* lengkap dapat dilihat pada Segmen Program 4.5.

### Segmen Program 4.5. Custom Pythorch

```
# Melihat asli dan palsu dari gambar pada folder
df_train[4:5]
# Mendefinisikan lokasi penyimpanan dari gambar
image1_path = os.path.join(training_dir, df_train.iat[4, 0])
image1_path
class Dataset(Dataset):
    # konstruktor default untuk menetapkan nilai
    def __init__(self, train_dir=None, training_csv=None,
transform=None):
        self.train_dir = train_dir

        self.train_data = pd.read_csv(training_csv)
        self.train_data.columns = ['image1', 'image2', 'class']
        self.transform = transform

    ## __getitem__ mengembalikan sampel data yang diberikan
indeks, idx=index
    def __getitem__(self, idx):
        img1_path = os.path.join(self.train_dir,
self.train_data.iat[idx, 0])
        img2_path = os.path.join(self.train_dir,
self.train_data.iat[idx, 1])

        img1 = Image.open(img1_path)
        img2 = Image.open(img2_path)
        # Gambar mode L, artinya ini adalah gambar saluran
tunggal - biasanya diinterpretasikan sebagai skala abu-abu.
        img1 = img1.convert('L')
        img2 = img2.convert('L')

        img1 = self.transform(img1)
        img2 = self.transform(img2)
        return img1, img2,
torch.from_numpy(np.array([int(self.train_data.iat[idx, 2])],
dtype=np.float32))

    ## __len__ mengembalikan ukuran kumpulan data
    def __len__(self):
        return len(self.train_data)

# Mengembalikan Image1, Image2 dan label kelas (apakah 0
atau 1).
dataset = Dataset(training_dir, training_csv,
```

```

transform=transforms.Compose([transforms.Resize((100, 100)),
transforms.ToTensor()]))
# memanggil variabel dataset untuk ditampilkan hasilnya
dataset

```

### Segmen Program 4.5. Lanjutan

Pada tabel 4.2 menunjukkan gambaran nilai 1 yaitu kemungkinan gambar tanda tangan palsu dan nilai 0 yaitu kemungkinan gambar tanda tangan asli pada persiapan data sebelum dilakukan proses *training*.

**Tabel 4. 2.** Gambaran Asli dan Palsu

Data Ke-	Gambar 1	Gambar 2	Nilai	Keterangan
1			0	Asli
2			0	Asli
3			1	Palsu

Dari tabel tersebut dapat disimpulkan bahwa pada tahapan ini data pada setiap folder akan dilatih. Pada data ke-1 merupakan gambar tanda tangan dari orang yang sama dengan pola yang berbeda maka sistem akan melabeli atau memberikan petunjuk bahwa gambar tersebut Asli. Sedangkan pada data ke-3 merupakan data dari orang yang berbeda dengan pola tanda tangan yang sama, maka kedua gambar tersebut dinyatakan Palsu. Pelatihan inilah yang akan mempermudah sistem saat proses implementasi pada metode.

#### b. Tahapan *Training* data

Pada tahapan ini data yang sudah diproses pada tahapan persiapan maka akan dilanjutkan pada tahapan *training* dengan menggunakan metode *Siamese*

*Network*. Dari tahapan persiapan sudah memiliki bekal untuk diimplementasikan pada tahap ini, data yang sudah dilatih menjadi data tanda tangan asli dan tanda tangan palsu selanjutnya akan diproses. Sebelum membuat *script* untuk training data, terlebih dahulu *script* untuk metode *Siamese Network* akan dibuat, dengan menggunakan *script Python*. *Script* lengkap dapat dilihat pada Lampiran 12. Selanjutnya adalah *script Contrastive Loss Function* merupakan *script* yang dipanggil dari *class Siamese Network*. Segmen Program 4.6. merupakan rumus *Siamese Network* yang digunakan untuk menentukan nilai *loss contrastive* dan mengitung jarak kemiripan gambar, *Loss Function* ini bekerja pada sepasang sampel, bukan sampel individu. Ini mendefinisikan indikator biner  $Y$  untuk setiap pasangan sampel yang menyatakan apakah sepasang sampel harus dianggap serupa atau tidak, dan fungsi jarak yang dapat dipelajari  $D_W(x_1, x_2)$  antara sepasang sampel  $x_1, x_2$ , diparameterisasi oleh bobot  $W$  di jaringan saraf, , di mana  $m > 0$  adalah margin. Margin mendefinisikan radius di sekitar ruang penyisipan sampel sehingga pasangan sampel yang berbeda hanya berkontribusi pada *Contrastive Loss Function* jika jarak  $D_W$  berada dalam margin.

#### Segmen Program 4. 6. *Contrastive Loss Function*

```
class ContrastiveLoss(torch.nn.Module):
    def __init__(self, margin=1.5):
        super(ContrastiveLoss, self).__init__()
        self.margin = margin
    def forward(self, output1, output2, label):
        euclidean_distance = F.pairwise_distance(output1,
output2)
        loss_contrastive = torch.mean((1 - label) *
torch.pow(euclidean_distance, 2) +
                                                (label) *
torch.pow(torch.clamp(self.margin - euclidean_distance,
min=0.0), 2))
        return loss_contrastive
```

Secara intuitif, *Loss Function* ini mendorong jaringan saraf untuk mempelajari penyematan untuk menempatkan sampel dengan label yang sama berdekatan satu sama lain, sambil menjauhkan sampel dengan label berbeda di ruang penyematan.

Tahap selanjutnya yang dilakukan adalah mengubah *Hardware accelerator* di *Google Colab* dan mengubahnya menjadi GPU untuk lebih mempercepat pemrosesan, kemudian ubah perangkat ke CUDA dan terapkan ke dalam metode *Siamese Network*, CUDA adalah lapisan perangkat lunak yang memberikan akses langsung ke set instruksi virtual GPU dan elemen komputasi paralel, untuk eksekusi kernel komputasi. Segmen Program 4.7 merupakan *script* untuk mengubah perangkat ke CUDA.

#### Segmen Program 4.7. Mengubah Perangkat ke CUDA

```
#Melakukan pengecekan ketersediaan perangkat CUDA
if torch.cuda.is_available():
    print('Yes')

#Menerapkan CUDA ke metode Siamese Network, juga memanggil
kelas (fungsi) ContrastiveLoss.
net = SiameseNetwork().cuda()
criterion = ContrastiveLoss()
optimizer = torch.optim.SGD(net.parameters(), lr = 3e-4)
optimizer = optim.RMSprop(net.parameters(), lr=1e-4,
alpha=0.99)
```

Segmen Program 4.8 menunjukkan proses *training* yang dilakukan dengan menggunakan *epoch* 200, *epoch* adalah *hyperparameter* yang menentukan berapa kali algoritma pembelajaran akan bekerja mengolah seluruh dataset training. Satu *epoch* berarti bahwa setiap sampel dalam *dataset training* memiliki kesempatan untuk memperbarui parameter model internal data dengan inputan hasil dari pemrosesan dataset yang telah dilakukan sebelumnya.

#### Segmen Program 4.8. Proses Training Data

```
def train():
    loss = []
    counter = []
    iteration_number = 0

    for epoch in range(1, 10):
        for i, data in enumerate(train_dataloader, 0):
            img0, img1, label = data
            img0, img1, label = img0.cuda(), img1.cuda(), label.cuda()
            optimizer.zero_grad()
            output1, output2 = net(img0, img1)
```

```

        loss_contrastive = criterion(output1, output2, label)
        loss_contrastive.backward()
        optimizer.step()

        print("Epoch {}\n Current loss {}".format(epoch,
loss_contrastive.item()))

        counter.append(iteration_number)
        loss.append(loss_contrastive.item())
    plt.plot(loss)
    return net

```

#### Segmen Program 4.8. Lanjutan

Segmen Program 4.9 merupakan *script* untuk memunculkan dan menyimpan hasil dari proses *training* data.

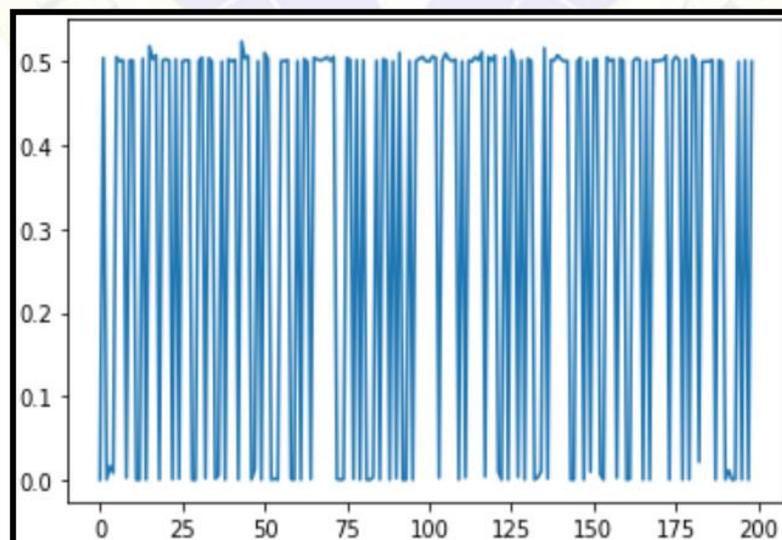
#### Segmen Program 4. 9. Menyimpan dan Memunculkan Hasil *Training* Data

```

model = train()
torch.save(model.state_dict(),
"/content/drive/MyDrive/dataaa/Models/model200.pt")
print("Model Saved Successfully")

```

Proses training menggunakan *epoch* 200 disimpan dalam bentuk “.pt” yang menjadi data bekal untuk tahapan testing. Hasil dari proses *epoch* 200 mendapatkan grafik, dapat dilihat pada Gambar 4.4. gambar tersebut menunjukkan nilai *Current loss*.



**Gambar 4.4.** Grafik Hasil *Training* Data

#### 4.1.4. Uji Coba

Tahap uji coba merupakan tahapan hasil mengujicobakan data *testing* dengan metode *Siamese Network* yang sudah diimplementasikan. Pada tahapan ini akan mengetahui tingkat akurasi pada tahapan implementasi dengan menggunakan metode *Siamese Network*. Segmen Program 4.10 merupakan pengambilan nilai akurasi menggunakan *script Python*. *Script* yang digunakan pada uji coba program lebih lengkap pada Lampiran 15.

#### Segmen Program 4. 10. Uji Coba (*Testing Dataset*)

```
for i, data in enumerate(test_dataloader, 0):
    x0, x1, label = data
    concat = torch.cat((x0, x1), 0)
    output1, output2 = model(x0.to(device), x1.to(device))
    euclidian_distance = F.pairwise_distance(output1, output2)

    if euclidian_distance <= 0.50:
        label = "Asli"
    else:
        label = "Palsu"

    imshow(torchvision.utils.make_grid(concat))
    print("Hasil ke-", count)
    print("Jarak kemiripan: " '%.2f' %
euclidian_distance.item())
    print("Label: ", label)
    count = count + 1

    if count == 168:
        break
```

Menentukan asli atau palsu dengan membandingkan gambar yang sudah ada di file csv data testing. Gambar testing akan dicocokkan dengan gambar dari data training. Berikut hasil uji coba:

- Uji coba dengan 63 gambar dan 168 kemungkinan gambar

Uji coba akan dilakukan dengan 14 data uji yang terdiri 7 Ketua Program Studi dan 7 Mahasiswa Fakultas Teknik . Menggunakan hasil *epoch* 200 dari proses *training* maka hasil uji coba ditunjukkan pada Tabel 4.3, secara rinci hasil uji coba dapat dilihat pada Lampiran 6.

**Tabel 4. 3.** Hasil Uji Coba

Banyak Kemungkinan gambar	Banyak Data Benar	Banyak Data Salah	Nilai Akurasi
168	159	8	94%

Hasil dari tahap uji coba pada Tabel 4.3. dengan menggunakan *epoch* 200 yang telah dilakukan dalam tahap implementasi sebelumnya mendapatkan nilai akurasi sebesar 94%. Sebanyak 63 gambar dan 168 kemungkinan gambar yang terdeteksi dan terdapat 8 kemungkinan gambar yang tidak dapat terdeteksi dengan benar dari total keseluruhan data uji 168 kemungkinan gambar. Data *training* yang digunakan pada uji coba sebanyak 14 dataset yang terdiri dari 7 dataset tanda tangan asli dan 7 *dataset* tanda tangan palsu, dan terdapat 980 kemungkinan gambar dari 147 gambar. Tabel 4.4 merupakan beberapa gambar pada data uji yang berhasil dideteksi. Gambar (a) merupakan gambar tanda tangan asli, gambar (b) merupakan gambar tanda tangan palsu.

**Tabel 4. 4.** Beberapa Gambar Pada Data Uji Yang Berhasil Dideteksi

a	b	Folder	Jarak	Label	Jenis Pemalsuan
		Running Ke 2 dari folder 005 dan 005_duplikat	0,68	Palsu	Skilled
		Running Ke 4 dari folder 003 dan 003_duplikat	1,31	Palsu	Unskilled
		Running Ke 0 dari folder 006 dan 006_duplikat	0,68	Palsu	Random

## 4.2. Pembahasan

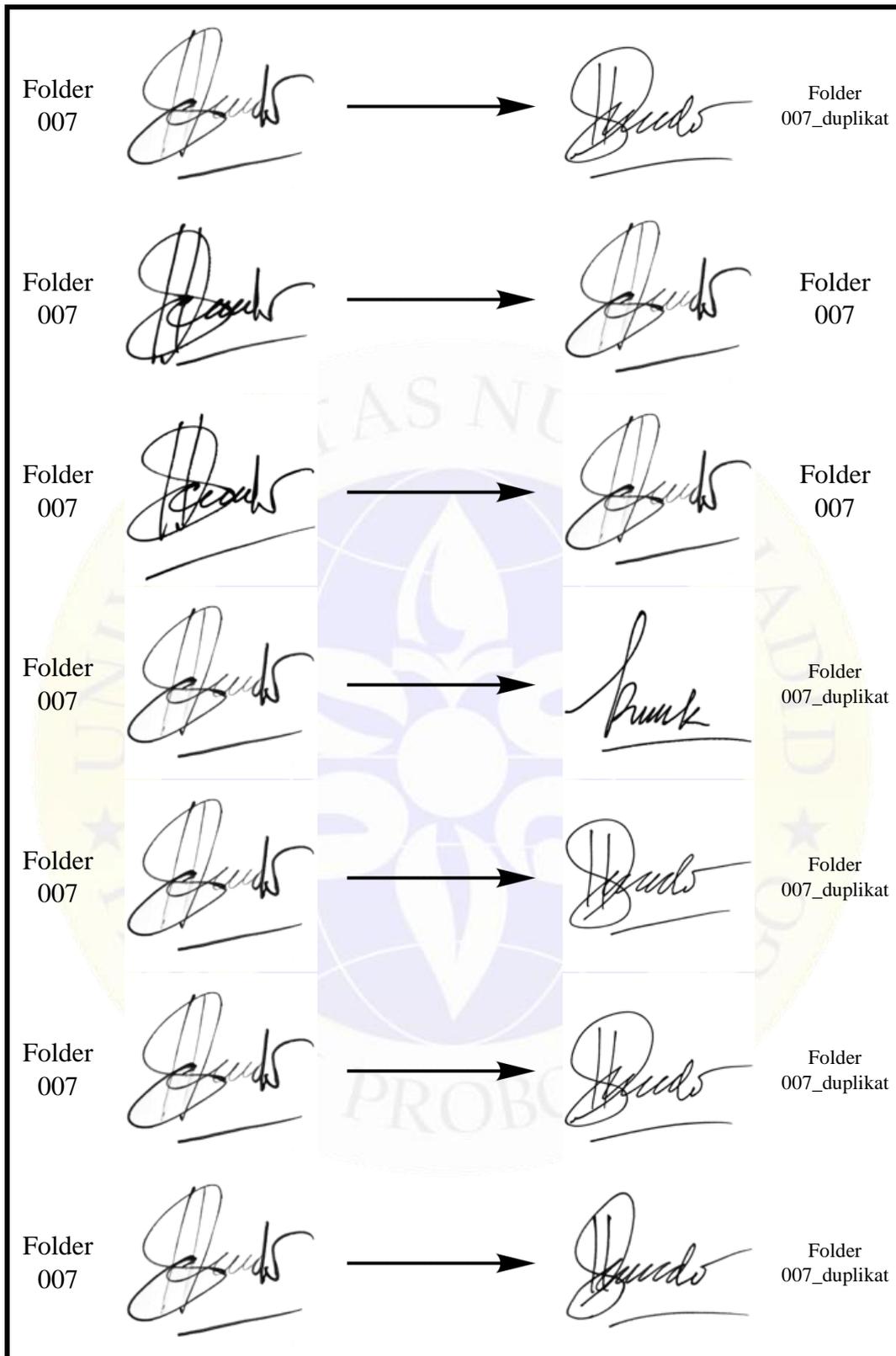
Pada bagian ini akan dibahas tentang hasil dari uji coba yang telah dilakukan dengan menggunakan metode *Siamese Network*. Adapun uji coba dilakukan, yaitu:

Hasil uji coba pada tahap ini menggunakan *epoch* 200. Pada tahap uji coba ini data *training* yang digunakan sebanyak 14 *dataset* yang terdiri dari 7 *dataset* tanda tangan asli dan 7 *dataset* tanda tangan palsu, dan terdapat 980 kemungkinan gambar dari 147 gambar, pada tahap uji coba ini data *testing* yang digunakan sebanyak 14 *dataset* yang terdiri dari 7 *dataset* tanda tangan asli dan 7 *dataset* tanda tangan palsu *Skilled*, *Unskilled*, dan *Random*, dan terdapat 168 kemungkinan gambar dari 63 gambar. Dari proses uji coba mendapat nilai akurasi 94%, dan terdapat 159 kemungkinan gambar yang dapat terdeteksi dengan baik dan terdapat 8 kemungkinan gambar yang tidak dapat terdeteksi dengan baik. Hasil dari gambar yang tidak dapat terdeteksi dengan baik pada tahap uji coba ditunjukkan oleh tabel 4.5.

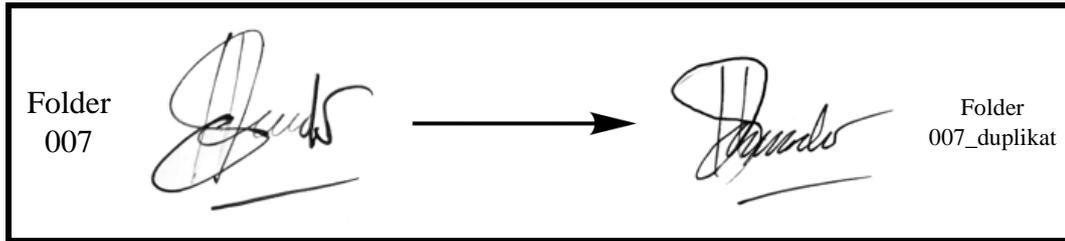
**Tabel 4. 5.** Tabel Hasil Gambar Tidak Terdeteksi

Label	Benar/Salah	Jenis Pemalsuan	Gambar 1		Gambar 2	
			Folder	Nama gambar	Folder	Nama gambar
Asli	Salah	Skilled	00.7	007_03	007_duplikat	0007_1
Palsu	Salah	Tanda tangan asli	00.7	007_01	00.7	007_03
Palsu	Salah	Tanda tangan asli	00.7	007_02	00.7	007_03
Asli	Salah	Random	00.7	007_03	007_duplikat	0007_5
Palsu	Salah	Skilled	00.7	007_03	007_duplikat	0007_2
Palsu	Salah	Skilled	00.7	007_03	007_duplikat	0007_1
Asli	Salah	Unskilled	00.7	007_03	007_duplikat	0007_4
Asli	Salah	Unskilled	00.7	007_03	007_duplikat	0007_3

Gambar 4.5. merupakan 8 kemungkinan gambar tanda tangan yang tidak dapat terdeteksi dengan baik pada uji coba dimana pada masing-masing folder terdapat 1 gambar yang tidak dapat terdeteksi dengan baik.

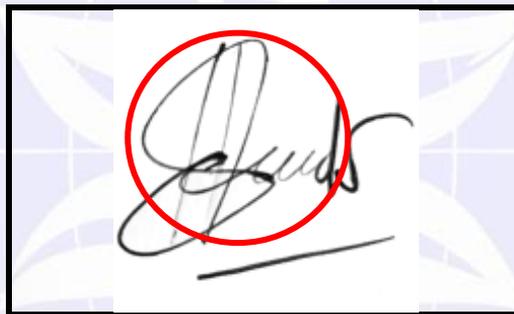


Gambar 4.5. Gambar Salah Prediksi



**Gambar 4.5.** Lanjutan

Dapat terlihat pada Gambar 4.5, menghasilkan beberapa kemungkinan gambar yang tidak dapat terdeteksi dengan baik. Hal ini disebabkan karena terdapat garis yang terputus dan pola yang kurang jelas di beberapa gambar tanda tangan, sehingga pola tanda tangan yang diambil menjadi tidak utuh, lalu tanda tangan seseorang bisa saja berubah karena beberapa faktor, seperti alat tulis, kertas yang digunakan, dan tekanan alat tulis, sehingga bisa memunculkan prediksi salah. Untuk mempermudah pemahaman dari kelemahan ini dapat dilihat pada Gambar 4.6.



**Gambar 4.6.** Gambar Tanda Tangan

Dari hasil uji coba dapat disimpulkan bahwa hasil *training* dengan menggunakan *epoch* 200 dapat dideteksi pada tahap uji coba. Pada tahap persiapan data untuk melatih data agar mendapatkan nilai asli dan palsu yang baik diperlukan ragam gambar pada *dataset*.

Selain itu penggunaan *Google Colaboratory* dalam penelitian ini masih mengalami beberapa kendala karena penggunaan *Hardware accelerator GPU* untuk mempercepat pemrosesan masih dibatasi hingga 12 jam penggunaan, dan *user* kembali bisa menggunakan *Hardware accelerator GPU* setelah 2 hari kemudian, sehingga jika memproses data lebih banyak, maka akan memerlukan waktu yang lebih lama.