IMPLEMENTASI ALGORTIMA CONVOLUTIONAL NEURAL NETWORK (CNN) UNTUK KLASIFIKASI DETEKSI PEROKOK

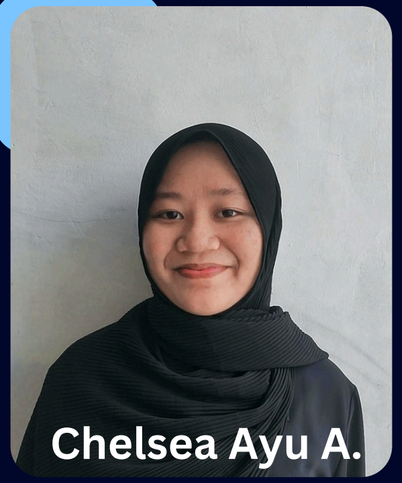
K E L O M P O K 1

Our Team

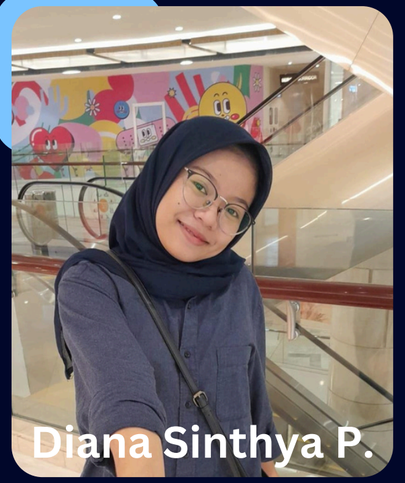
Adelia Azizatul Haq
21083010009

Edelin Fortuna
21083010087

Yunita Nur
21083010107



Chelsea Ayu A.
21083010028



Diana Sinthya P.
21083010090

Background

01 Merokok merupakan salah satu kebiasaan yang memiliki dampak buruk terhadap kesehatan, baik bagi perokok aktif maupun pasif.

02 Pendeteksian kebiasaan merokok menjadi langkah penting, terutama dalam ruang publik untuk memastikan area tersebut bebas dari asap rokok.

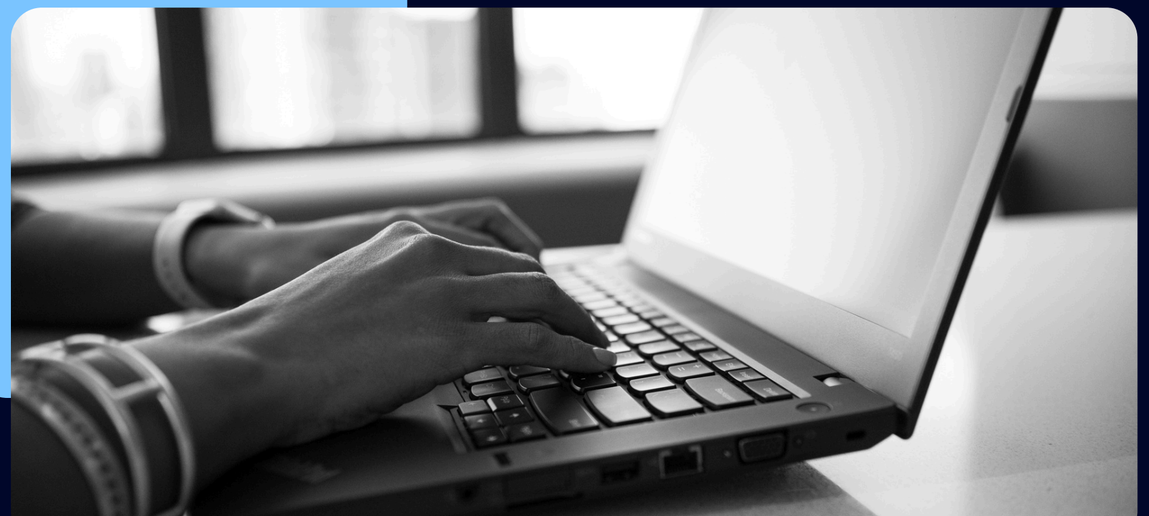
03 Dalam beberapa penelititian, CNN mampu mengolah data gambar dan sensor secara berurutan

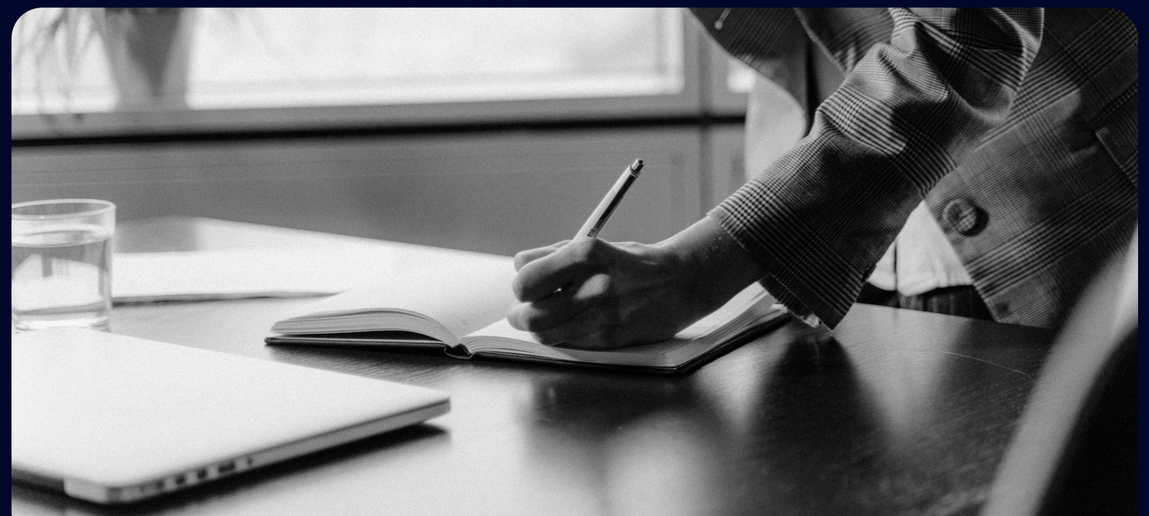
04 Pendekatan ini membandingkan akurasi deteksi, mempercepat proses identifikasi, dan membuka peluang untuk pengembangan sistem deteksi otomatis berbasis teknologi

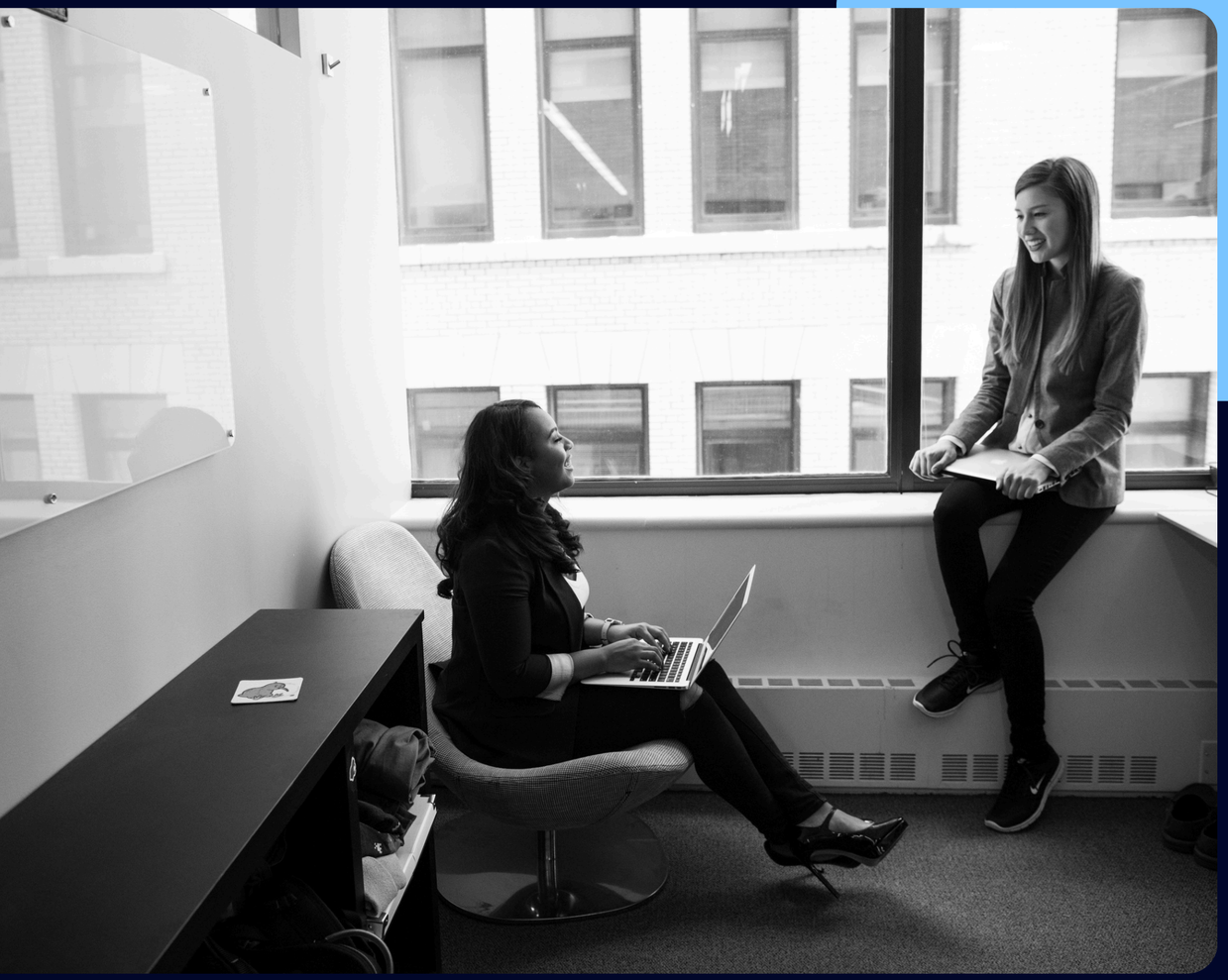
Issue

01 Proses deteksi perokok secara manual memakan waktu dan biaya

02 Dibutuhkan sistem otomatis yang cepat dan real time untuk mendukung deteksi perokok







Objective

01 Mengembangkan dan mengimplementasikan model CNN untuk mendeteksi aktivitas merokok dari gambar

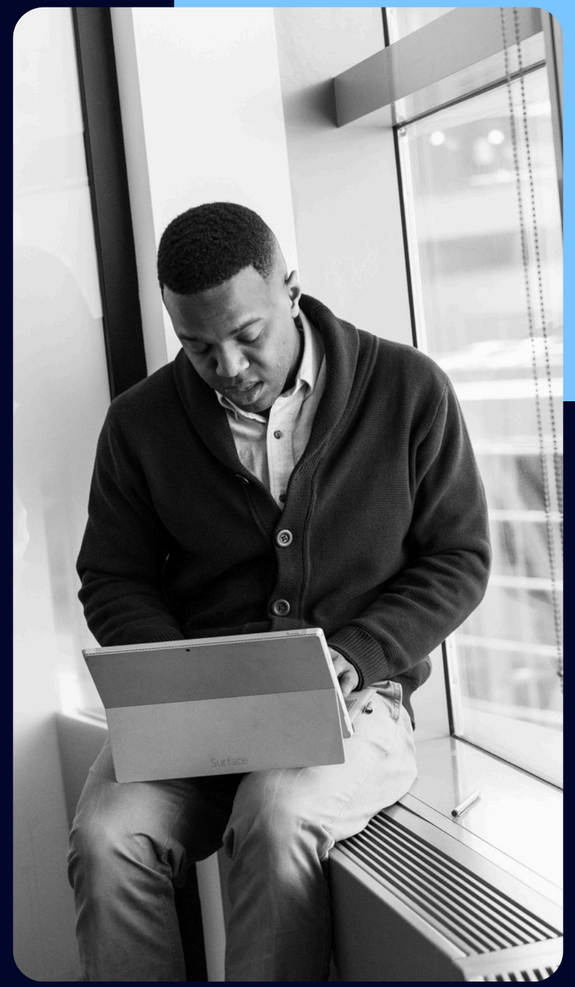
02 Membandingkan performa varian arsitektur dari Convolutional Neural Network (CNN)

Benefit

01 Memberikan kontribusi dalam pengembangan sistem deteksi perilaku berbasis AI dengan fokus pada klasifikasi visual

02 Mempercepat proses identifikasi perokok dibandingkan metode manual

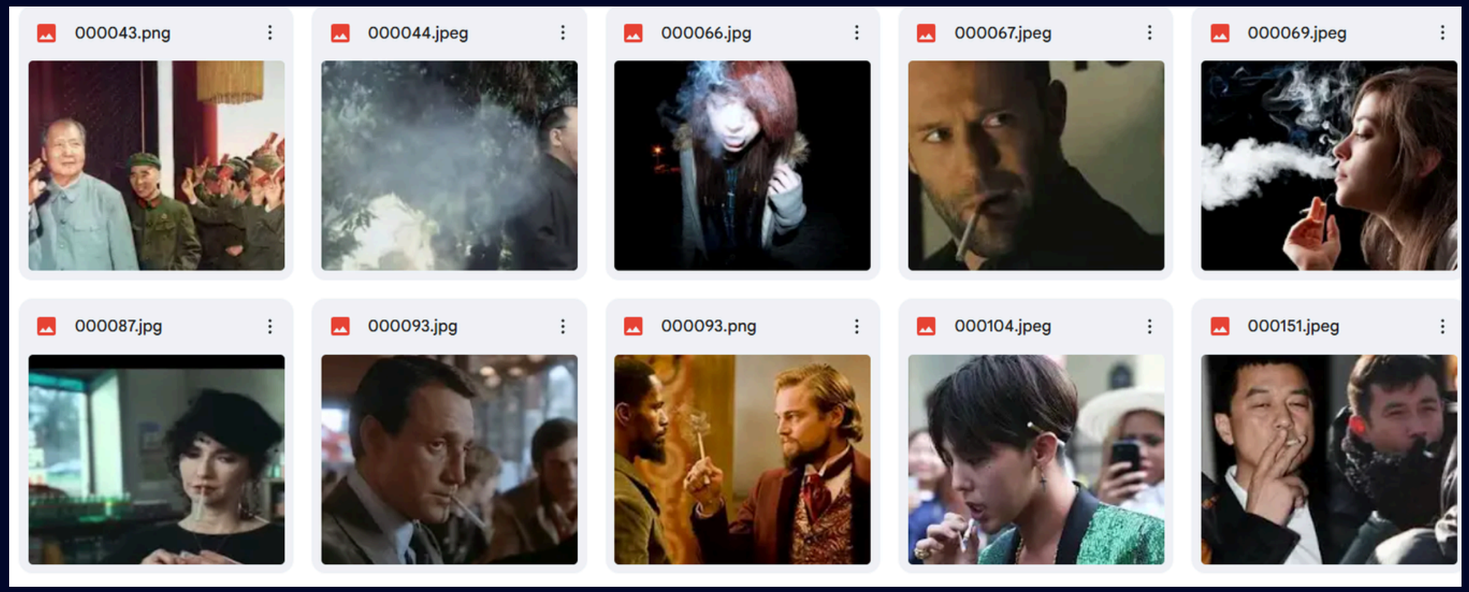
03 Mendukung kebijakan *No Smoking Area* dengan alat yang lebih canggih untuk memantau pelanggaran secara efisien.



DATASET

Dataset dalam proyek ini terdiri dari dua kelas utama, yaitu Smoking dan Not Smoking. Kelas Smoking mencakup gambar atau video yang menunjukkan seseorang sedang merokok, sedangkan kelas Not Smoking mencakup data yang menunjukkan orang tidak merokok dari berbagai sumber.

Link dataset

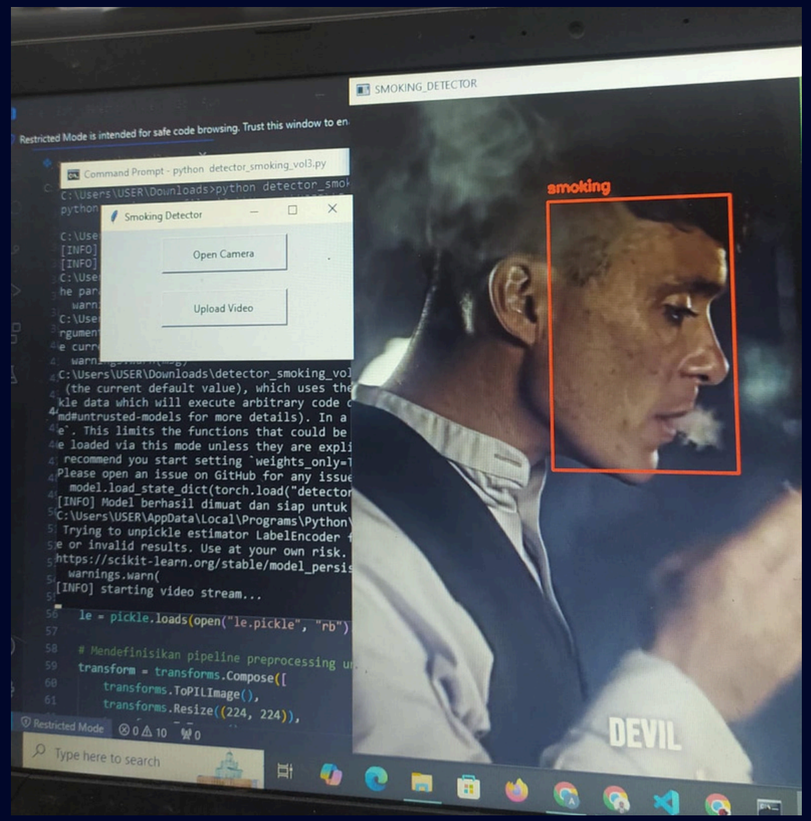


METODE

Dalam proyek ini, kami menggunakan empat metode utama untuk mengklasifikasikan aktivitas merokok. Tiga di antaranya adalah model CNN (Convolutional Neural Network), yaitu

1. AlexNet
2. MobileNetV2
3. ResNet50

Sementara satu metode tambahan yang kami sebut “Smoking Method” digunakan sebagai pembanding tambahan.



METODOLOGI PENELITIAN

Tiga tahapan utama proyek ini:

Preprocessing

1. Ekstrak file zip

```
from zipfile import ZipFile

# Replace with the actual path to your zip file in Google Drive
zip_file_path = "/content/drive/My Drive/DATA-UAS.zip"
extract_path = "/content/drive/My Drive/extracted_files" # Replace with your desired
extraction path


try:
  with ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)
    print(f"Successfully extracted {zip_file_path} to {extract_path}")
except FileNotFoundError:
  print(f"Error: Zip file not found at {zip_file_path}")
except Exception as e:
  print(f"An error occurred: {e}")
```

Preprocessing

1. Ekstrak file zip

   output

```
Successfully extracted /content/drive/My Drive/DATA-UAS.zip to /content/drive/My
Drive/extracted_files
```

Preprocessing

2. Menentukan path folder dan memeriksa isi folder

```
import os

# Tentukan path folder
folder_path = '/content/drive/MyDrive/UAS-SMOKING'

# Periksa isi folder
for root, dirs, files in os.walk(folder_path):
  print(f"Path: {root}")
  print(f"Subdirektori: {dirs}")
  print(f"Jumlah file: {len(files)}\n")
```

Preprocessing

3. Splitting data

```
def split_dataset(source_dir, dest_dir, split_ratios=(0.8, 0.1, 0.1)):
                                                    .
                                                    .
                                                    .

print("Dataset berhasil dipisahkan berdasarkan label!")
```

```
# Path direktori asal dan tujuan
source_directory = '/content/drive/MyDrive/UAS-SMOKING'
destination_directory = "/content/drive/MyDrive/DATA-UAS"

# Memisahkan dataset dengan rasio 80% train, 10% val, 10% test
split_dataset(source_directory, destination_directory, split_ratios=(0.8, 0.1, 0.1))
```

Preprocessing

4. Mengecek jumlah data dalam folder

```
# Tentukan path folder
dataset = '/content/drive/MyDrive/DATA-UAS'

# Periksa isi folder
for root, dirs, files in os.walk(dataset):
  print(f"Path: {root}")
  print(f"Subdirektori: {dirs}")
  print(f"Jumlah file: {len(files)}\n")
```

Preprocessing

5. Transformasi data

```
# Transformasi
basic_transform = transforms.Compose([
  transforms.Resize((224, 224)),          # Resize ke ukuran tetap
  transforms.ToTensor(),               # Konversi ke tensor
  transforms.Normalize(mean=[0.485, 0.456, 0.406], # Normalisasi standar ImageNet
             std=[0.229, 0.224, 0.225])
])
```

Preprocessing

6. Ekstraksi label

```
source_base = "/content/drive/MyDrive/DATA-UAS"
train_dataset = datasets.ImageFolder(os.path.join(source_base, "train"),
transform=basic_transform)
val_dataset = datasets.ImageFolder(os.path.join(source_base, "val"),
transform=basic_transform)
test_dataset = datasets.ImageFolder(os.path.join(source_base, "test"),
transform=basic_transform)
```

Preprocessing

7. Membuat data loader

```
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=16, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)

dataloaders = {
    'train': train_loader,
    'val': val_loader,
    'test': test_loader
}
```

Modeling

1. Load Model

AlexNet	MobileNetV2	ResNet50	Smoking Method
`models.alexnet(weights=models.AlexNet_Weights.DEFAULT)`	`models.mobilenet_v2(weights=models.MobileNet_V2_Weights.DEFAULT)`	`models.resnet50(weights=models.ResNet50_Weights.DEFAULT)`	`model = load_model("smoking.model.h5")`

Modeling

2. Modifikasi layer yang disesuaikan dengan jumlah kelas

```
# Menyusun model untuk klasifikasi biner (2 kelas)
for model in models_list:
  # Modifikasi final layer untuk klasifikasi biner
  if isinstance(model, models.ResNet):
    model.fc = nn.Linear(model.fc.in_features, 1) # 1 neuron output untuk klasifikasi biner
  elif isinstance(model, models.AlexNet):
    model.classifier[6] = nn.Linear(model.classifier[6].in_features, 1) # Modifikasi final
layer AlexNet
  elif isinstance(model, models.MobileNetV2):
    model.classifier[1] = nn.Linear(model.classifier[1].in_features, 1) # Modifikasi final
layer MobileNetV2
```

Modeling

3. Memindahkan model untuk dijalankan di GPU

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

Modeling

4. Menentukan criterion dan optimizer

```
# Define loss function
criterion = nn.BCEWithLogitsLoss()

# Define optimizer menggunakan SGD
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

Modeling

5. Training model

```
def train_model(model, dataloaders, criterion, optimizer, num_epochs=30, device='cuda'):
    best_acc = 0.0 # Menyimpan akurasi terbaik
    best_model_wts = deepcopy(model.state_dict()) # Menyimpan bobot terbaik model
                                                  .
                                                  .
                                                  .

   # Muat bobot terbaik setelah training selesai
   model.load_state_dict(best_model_wts)
   print(f'Best Validation Accuracy: {best_acc:.4f}')
   return model, train_losses, val_losses, train_accuracies, val_accuracies
```

```
# Train model
model, train_losses, val_losses, train_accuracies, val_accuracies = train_model(
   model, dataloaders, criterion, optimizer, num_epochs=10, device=device
)
```

Evaluation

1. Mengaktifkan model evaluasi

```
model.eval()
```

2. Inisialisasi variabel

```
total_loss = 0.0
all_preds, all_labels = [], []
```

3. Menonaktifkan perhitungan gradien

```
with torch.no_grad():
```

Evaluation

4. Iterasi pada dataloader

```
for inputs, labels in dataloader:
    inputs, labels = inputs.to(device), labels.to(device)
```

5. Forward pass

```
outputs = model(inputs).squeeze()
```

6. Hitung loss

```
loss = criterion(outputs, labels.float())
total_loss += loss.item()
```

Evaluation

7. Mengonversi output ke biner

```
preds = torch.sigmoid(outputs).detach().cpu().numpy() > 0.5
```

8. Menyimpan prediksi & label

```
all_preds.extend(preds)
all_labels.extend(labels.cpu().numpy())
```

9. Menghitung rata-rata loss dan akurasi

```
avg_loss = total_loss / len(dataloader)
all_preds, all_labels = np.array(all_preds), np.array(all_labels)
accuracy = (all_preds == all_labels).mean() * 100
```

Evaluation

10. Confusion matrix

```
cm = confusion_matrix(all_labels, all_preds)
print("Confusion Matrix:")
print(cm)
print(classification_report(all_labels, all_preds, target_names=["Smoking", "Not Smoking"]))
```

Evaluation

11. Output training model

AlexNet	MobileNetV2	ResNet50	Smoking Method
Train Loss: 0.0372 Acc: 0.9875	Train Loss: 0.0619 Acc: 0.9804	Train Loss: 0.0135 Acc: 0.9962	-

Evaluation

12. Output Testing

AlexNet	MobileNetV2	ResNet50	Smoking Method
Acc : 0.91	Acc : 0.92	Acc : 0.94	Acc : 0,52

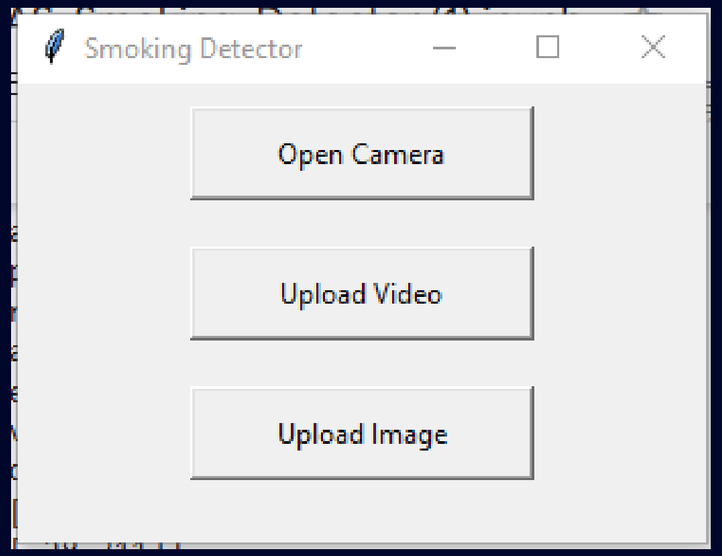
Hasil dan Pembahasan

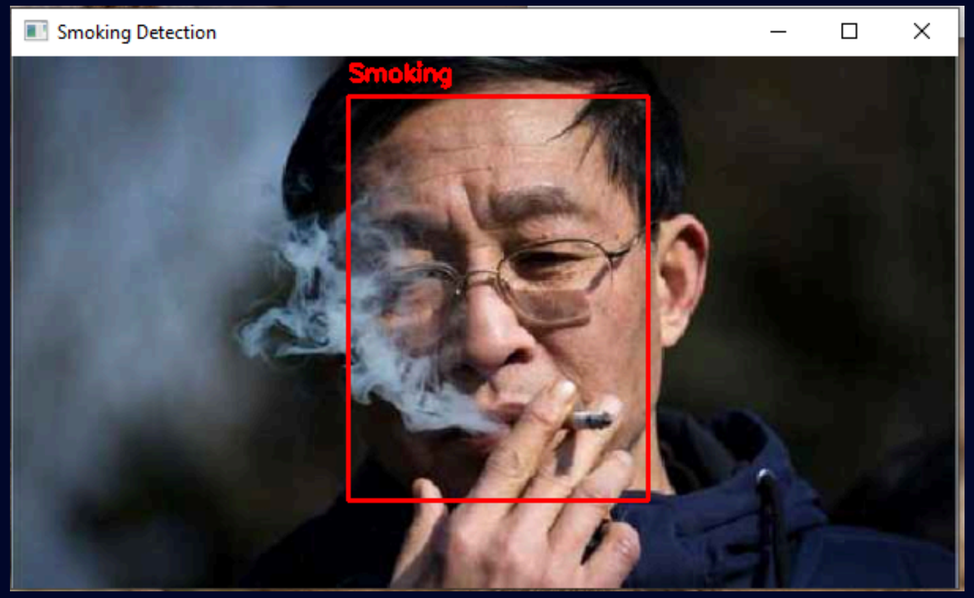
AlexNet	MobileNetV2
Confusion Matrix: [[317 35] [25 314]] precision recall f1-score support Smoking 0.93 0.90 0.91 352 Not Smoking 0.90 0.93 0.91 339 accuracy 0.91 691 macro avg 0.91 0.91 0.91 691 weighted avg 0.91 0.91 0.91 691	Confusion Matrix: [[337 15] [40 299]] precision recall f1-score support Smoking 0.89 0.96 0.92 352 Not Smoking 0.95 0.88 0.92 339 accuracy 0.92 691 macro avg 0.92 0.92 0.92 691 weighted avg 0.92 0.92 0.92 691

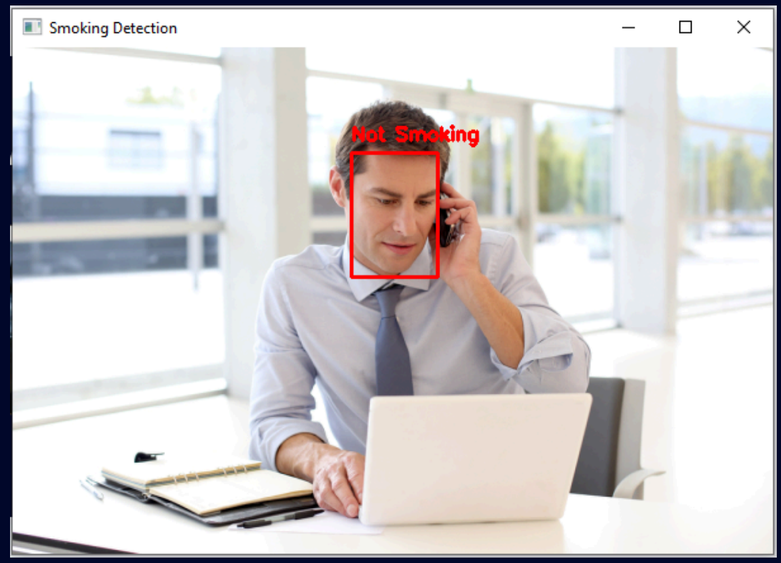
Hasil dan Pembahasan

ResNet50	Smoking Method
Confusion Matrix: [[336 16] [28 311]] precision recall f1-score support Smoking 0.92 0.95 0.94 352 Not Smoking 0.95 0.92 0.93 339 accuracy 0.94 691 macro avg 0.94 0.94 0.94 691 weighted avg 0.94 0.94 0.94 691	Confusion Matrix: [[71 268] [63 289]] precision recall f1-score support Smoking 0.53 0.21 0.30 339 Not Smoking 0.52 0.82 0.64 352 accuracy 0.52 691 macro avg 0.52 0.52 0.47 691 weighted avg 0.52 0.52 0.47 691

GUI DOCUMENTATION







Kesimpulan

ResNet50 menunjukkan performa terbaik dengan akurasi 94%, diikuti oleh MobileNetV2 (92%) dan AlexNet (91%). Metode Smoking hanya mencapai akurasi 52.10%, sehingga kurang cocok untuk tugas klasifikasi ini. ResNet50 direkomendasikan karena hasilnya paling optimal.

Thank You