LAMPIRAN

Lampiran 1. Riwayat hidup

I Putu Arya Vidyananta lahir di Tinading pada tanggal 31 Maret 2004. Penulis merupakan putra dari pasangan I Made Sumerta dan Ni Kadek Suliasmini. Penulis berkewarganegaraan Indonesia dan beragama Hindu. Pendidikan dasar ditempuh di SD Negeri Tinading dan diselesaikan pada tahun 2016. Selanjutnya, penulis melanjutkan pendidikan menengah pertama di SMP Negeri 1 Lampasio dan lulus pada tahun 2019. Pendidikan menengah atas ditempuh di SMA Negeri 1 Tolitoli pada jurusan Matematika dan Ilmu Pengetahuan Alam (MIPA) dan diselesaikan pada tahun 2022. Pada tahun yang sama, penulis melanjutkan pendidikan ke jenjang Sarjana (S1) pada Program Studi Sistem Informasi, Jurusan Teknik Informatika, Fakultas Teknik dan Kejuruan, Universitas Pendidikan Ganesha.

Lampiran 2. Surat Permohonan Pelabelan Data

KEMENTERIAN PENDIDIKAN TINGGI,
SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
Jalan Udayana Nomor 11 Singaraja Bali
Laman: http://fik.undiksha.ac.id

Nomor : **1962**/UN48.11.1/KM/2025 Singaraja, 21 Juli 2025
Perihal : Surat Permohonan uji ahli

Yth. Kepala SMA Negeri 1 Singaraja
di tempat

Dengan hormat, sehubungan dengan proses penyelesaian Tugas Akhir/Skripsi, maka melalui surat ini kami mohon Bapak/Ibu berkenan memberikan data yang terkait dengan data yang dibutuhkan. Adapun mahasiswa yang akan melakukan pengambilan data seperti tersebut di bawah ini:

Nama	: I Putu Arya Vidyananta
NIM	: 2215091079
Program Studi	: Sistem Informasi
Jurusan	: Teknik Informatika
Data yang dibutuhkan	: Permohonan ahli/guru Bahasa Indonesia untuk melakukan pelabelan data
Judul Penelitian	: Analisis Sentimen Program Naturalisasi Tim Nasional Sepak Bola Indonesia Pada Media Sosial X Menggunakan IndoBERT Embedding dan Convolutional Neural Network

Demikian kami sampaikan, atas perhatian dan kerjasamanya, diucapkan terima kasih.

a.n Dekan
Wakil Dekan Bidang Akademik,

Made Windu Antara Kesiman
NIP 198211112008121001

Lampiran 3. Surat Validasi Pelabelan Data

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
JURUSAN TEKNIK INFORMATIKA
Jalan Udayana Singaraja-Bali Kode Pos 81116
Tlp. (0362) 22570 Fax. (0362) 25735
Laman: www.undiksha.ac.id

SURAT KETERANGAN
LABELING DATASET

Yang bertanda tangan di bawah ini:

Nama : I Gusti Ayu Putri Dwi Epayani, S.Pd, M.Pd
NIP : 198706142022212019

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha dibawah ini:

Nama : I Putu Arya Vidyananta
NIM : 2215091079
Prodi/Jurusan : S1 Sistem Informasi/Teknik Informatika

Dengan ini menerangkan bahwa proses pelabelan dataset telah dilaksanakan dan diselesaikan pada tanggal 22 Agustus 2025.

Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 22 Agustus 2025
Ahli/Pakar I,

I Gusti Ayu Putri Dwi Epayani, S.Pd, M.Pd
NIP. 198706142022212019

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
JURUSAN TEKNIK INFORMATIKA
Jalan Udayana Singaraja-Bali Kode Pos 81116
Tlp. (0362) 22570 Fax. (0362) 25735
Laman: www.undiksha.ac.id

SURAT KETERANGAN

LABELING DATASET

Yang bertanda tangan di bawah ini:

Nama : Dra. Ni Wayan Sumiasih
NIP : 196811042022212003

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha dibawah ini:

Nama : I Putu Arya Vidyananta
NIM : 2215091079
Prodi/Jurusan : S1 Sistem Informasi/Teknik Informatika

Dengan ini menerangkan bahwa proses pelabelan dataset telah dilaksanakan dan diselesaikan pada tanggal 22 Agustus 2025.

Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 22 Agustus 2025
Ahli/Pakar II,

Dra. Ni Wayan Sumiasih
NIP. 196811042022212003

KEMENTERIAN PENDIDIKAN TINGGI, SAINS, DAN TEKNOLOGI
UNIVERSITAS PENDIDIKAN GANESHA
FAKULTAS TEKNIK DAN KEJURUAN
JURUSAN TEKNIK INFORMATIKA
Jalan Udayana Singaraja-Bali Kode Pos 81116
Tlp. (0362) 22570 Fax. (0362) 25735
Laman: www.undiksha.ac.id

SURAT KETERANGAN

LABELING DATASET

Yang bertanda tangan di bawah ini:

Nama : Gede Sukalima Aksirnaka, S.Pd., M.Pd.
NIP : 196804141990021002

Menerangkan bahwa Mahasiswa Universitas Pendidikan Ganesha dibawah ini:

Nama : I Putu Arya Vidyananta
NIM : 2215091079
Prodi/Jurusan : S1 Sistem Informasi/Teknik Informatika

Dengan ini menerangkan bahwa proses pelabelan dataset telah dilaksanakan dan diselesaikan pada tanggal 22 Agustus 2025.

Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat digunakan sebagaimana mestinya.

Singaraja, 22 Agustus 2025
Ahli/Pakar III,

Gede Sukalima Aksirnaka,S.Pd.,M.Pd.
NIP. 196804141990021002

Lampiran 4. Sertifikasi Pendidik Para Ahli

No. 0004488

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor: 00101486904241563059

Berdasarkan Surat Keputusan Menteri Pendidikan, Kebudayaan, Riset, dan Teknologi Nomor 29/E/O/2023 tanggal 10 Januari 2023 tentang Izin Pembukaan Program Studi Pendidikan Profesi Guru Program Profesi pada Universitas Nusa Cendana di Kota Kupang,
Rektor Universitas Nusa Cendana menyatakan bahwa:

I GUSTI AYU PUTRI DWI EPAYANI
Nomor Induk Mahasiswa: 2401190597

lahir di Muncan pada tanggal empat belas bulan Juni tahun seribu sembilan ratus delapan puluh tujuh telah memenuhi semua syarat penyelesaian Pendidikan Profesi Guru dan LULUS Uji Kompetensi Peserta Pendidikan Profesi Guru. Kepadanya diberikan sebutan profesi GURU (Gr.) Bahasa Indonesia sesuai hak dan kewajiban yang melekat pada sebutan profesi tersebut.

Kota Kupang, 8 Oktober 2024
Rektor,

REKTOR

Prof. Dr. drh. Maxs U. E. Sanam, M.Sc.

No. 03364

DEPARTEMEN PENDIDIKAN NASIONAL
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor : 210815603364

Berdasarkan Surat Keputusan Menteri Pendidikan Nasional Nomor 057/O/2007 tentang Penetapan Perguruan Tinggi Penyelenggara Sertifikasi bagi Guru dalam Jabatan, Rektor Universitas Pendidikan Ganesha, selaku Ketua Rayon 21, menyatakan bahwa :

Dra. NI WAYAN SUMIASIH

Nomor peserta 08220115610817, lahir di Pitera pada tanggal 4 Nopember 1968
Guru di Sekolah Menengah Atas Lab. Undiksha Singaraja, Bali

LULUS Sertifikasi Guru dalam Jabatan dan dinyatakan sebagai GURU PROFESIONAL bidang studi BAHASA INDONESIA (DAN SASTRA INDONESIA)

Singaraja, 27 Desember 2008
Ketua Rayon 21,

Prof. Dr. I Nyoman Sudiana, M.Pd.
NIP. 131477325

No. 00255

DEPARTEMEN PENDIDIKAN NASIONAL
REPUBLIK INDONESIA

SERTIFIKAT PENDIDIK

Nomor : 210915600255

Berdasarkan Surat Keputusan Menteri Pendidikan Nasional Nomor 022/P/2009 tentang Penetapan Perguruan Tinggi Penyelenggara Sertifikasi bagi Guru dalam Jabatan, Rektor Universitas Pendidikan Ganesha, selaku Ketua Rayon 21, menyatakan bahwa :

GEDE SUKALIMA AKSIRNAKA, S.Pd.

Nomor peserta 09220115610505, lahir di Buleleng pada tanggal 14 April 1968

LULUS Sertifikasi Guru dalam Jabatan dan dinyatakan sebagai GURU PROFESIONAL bidang studi BAHASA INDONESIA (DAN SASTRA INDONESIA)

Singaraja, 21 Desember 2009
Ketua Rayon 21/Rektor,

Prof. Dr. I Nyoman Sudiana, M.Pd.
NIP. 131477325

Lampiran 5. Dokumentasi Pelabelan Data

Lampiran 6. Code Crawling Data

```
# Import required Python package
%pip install pandas

!sudo apt-get update
!sudo apt-get install -y ca-certificates curl gnupg
!sudo mkdir -p /etc/apt/keyrings
!curl -fsSL
https://deb.nodesource.com/gpgkey/nodesource-
repo.gpg.key | sudo gpg --dearmor -o
/etc/apt/keyrings/nodesource.gpg

!NODE_MAJOR=20 && echo "deb [signed-
by=/etc/apt/keyrings/nodesource.gpg]
https://deb.nodesource.com/node_$NODE_MAJOR.x
```

```
nodistro main" | sudo tee
/etc/apt/sources.list.d/nodesource.list

!sudo apt-get update
!sudo apt-get install nodejs -y

!node -v

# Crawl Data
filename = 'Naturalisasi-pssi-2.csv'
search_keyword = 'Naturalisasi pssi lang:id
until:2024-05-23 since:2024-01-01 -filter:links'
limit = 10000

!npx -y tweet-harvest@2.6.1 -o "{filename}" -s
"{search_keyword}" --tab "LATEST" -l {limit} --token
{twitter_auth_token}
```

Lampiran 7 Kode *Pre-Procesing* Data

1. Kode Cleaning Data

```
import re
def cleaning(text):
text = re.sub(r"http\S+", "", text
text = re.sub(r"@\w+", "", text)
text = re.sub(r"#\w+", "", text)
text = re.sub(r"[^\w\s]", "", text)
text = re.sub(r"\d+", "", text)
text = re.sub(r"_+", " ", text
text = re.sub(r"\s+", " ", text)
return text.strip()
# Terapkan ke DataFrame
df['clean_text'] = df['full_text'].apply(cleaning)
print(df[['full_text', 'clean_text']].head())
```

2. Kode Case Folding Data

```
def case_folding(text):
return text.lower()
df['casefold_text']=
df['clean_text'].apply(case_folding)
print(df[['clean_text', 'casefold_text']].head())
```

3. Kode Tokenizing

```
df["tokens"]    =    df["casefold"].apply(lambda    x:
x.split())
df[["casefold", "tokens"]].head()
```

4. Kode Normalization

```
kamus = pd.read_csv("slang.csv")
slang_dict = dict(zip(kamus['slang'],
kamus['formal']))

def normalisasi(tokens):
    return [slang_dict.get(token, token) for token in
tokens]

df["normalized_tokens"] =
df["tokens"].apply(normalisasi)
df[["tokens", "normalized_tokens"]].head()
```

5. Kode Stemming

```
from tqdm import tqdm
tqdm.pandas()
factory = StemmerFactory()
stemmer = factory.create_stemmer()

df["stemmed"] =
df["normalized_tokens"].progress_apply(
```

```
    lambda tokens: " ".join([stemmer.stem(t) for t in
tokens])
)


df[["normalized_tokens", "stemmed"]].head
```

Lampiran 8. Kode Modeling

1. Kode IndoBERT Embedding

```
import pandas as pd
import torch
from transformers import BertTokenizer, BertModel
from tqdm import tqdm

#Konfigurasi
MAX_LEN = 64
BATCH_SIZE = 16
MODEL_NAME = "indobenchmark/indobert-base-p1"
CSV_PATH = "Dataset-Modeling.csv"
TEXT_COLUMN = "text"
SAVE_PATH = "indobert_embeddings_modeling.pt"

print("Memuat IndoBERT tokenizer dan model...")
tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)
model = BertModel.from_pretrained(MODEL_NAME)

device = torch.device("cpu")
model.to(device)
model.eval()

#Memuat Dataset
print("Memuat dataset dari", CSV_PATH)
df = pd.read_csv(CSV_PATH)

if TEXT_COLUMN not in df.columns:
    raise ValueError(f"Kolom '{TEXT_COLUMN}' tidak
ditemukan!")

texts = df[TEXT_COLUMN].astype(str).tolist()
num_texts = len(texts)

print(f"Jumlah data: {num_texts}")

#Proses Embedding
print("Memulai proses embedding...")
all_embeddings = []
```

```
for i in tqdm(range(0, num_texts, BATCH_SIZE),
desc="Embedding batch"):
    batch_texts = texts[i:i + BATCH_SIZE]

    encoded = tokenizer (
        batch_texts,
        padding="max_length",
        truncation=True,
        max_length=MAX_LEN,
        return_tensors="pt"
    )

    input_ids = encoded["input_ids"].to(device)
    attention_mask =
encoded["attention_mask"].to(device)

    with torch.no_grad ():
        outputs = model (
            input_ids=input_ids,
            attention_mask=attention_mask
        )

    batch_embeddings =
outputs.last_hidden_state.cpu()
    all_embeddings.append(batch_embeddings)

final_tensor = torch.cat (all_embeddings, dim=0)

print ("Shape akhir tensor:", final_tensor.shape)

#Simpan Hasil
torch.save(final_tensor, SAVE_PATH)
print (f"Embedding berhasil disimpan ke
'{SAVE_PATH}'")
```

2. Kode Convutional Neural Network

```
class SimpleCNN(nn.Module):
    def __init__(
        self,
        kernel_size=3,
        num_filters=64,
        dropout_rate=0.5,
        num_classes=3
    ):
        super().__init__()
```

```
        self.conv = nn.Conv1d(
            in_channels=768,
            out_channels=num_filters,
            kernel_size=kernel_size,
            padding=kernel_size // 2
        )
        self.relu = nn.ReLU()
        self.pool = nn.AdaptiveMaxPool1d(1)
        self.dropout = nn.Dropout(dropout_rate)
        self.fc = nn.Linear(num_filters, num_classes)

    def forward(self, x):

        x = x.permute(0, 2, 1)
        x = self.relu(self.conv(x))
        x = self.pool(x).squeeze(-1)
        x = self.dropout(x)
        return self.fc(x)
```

Lampiran 9. Kode Evaluasi

```
def run_simplecnn_kfold(
    X, y,
    kernel_size=3,
    batch_size=64,
    use_smote=False,
    n_splits=5,
    num_epochs=10,
    patience=3,
    target_names=("Negatif", "Netral", "Positif"),
    lr=0.001,
    device="cpu"
):

    print(
        f"\n=== Simple CNN | {n_splits}-Fold | "
        f"Kernel {kernel_size}"
    )

    kf = KFold(n_splits=n_splits, shuffle=True,
random_state=42)

    all_preds, all_labels = [], []
    fold_f1, fold_acc = [], []

    # Akurasi Train dan validation per fold
    train_acc_per_fold = []
```

```
    val_acc_per_fold = []

    best_global_state = None
    best_global_f1 = -1

    for fold, (train_idx, val_idx) in
enumerate(kf.split(X), start=1):
        print("\n======================")
        print(f"Fold {fold}")
        print("======================")

        X_train, y_train = X[train_idx], y[train_idx]
        X_val, y_val     = X[val_idx], y[val_idx]

        if use_smote:
            print(f"Fold {fold} | SMOTE pada data
training")
            X_train, y_train = prepare_dataset(
                X_train, y_train, use_smote=True
            )

        train_loader = DataLoader(
            TensorDataset(X_train, y_train),
            batch_size=batch_size,
            shuffle=True
        )
        val_loader = DataLoader(
            TensorDataset(X_val, y_val),
            batch_size=batch_size,
            shuffle=False
        )

        model = SimpleCNN(
            kernel_size=kernel_size,
            num_filters=64,
            dropout_rate=0.5,
            num_classes=3
        ).to(device)

        criterion = nn.CrossEntropyLoss()
        optimizer = optim.Adam(model.parameters(),
lr=lr)

        best_loss = float("inf")
        best_state = None
        patience_counter = 0

        last_train_acc = 0.0
        last_val_acc = 0.0
```

```
        for epoch in range(num_epochs):
            # Train
            model.train()
            train_preds, train_labels = [], []

            for xb, yb in train_loader:
                xb, yb = xb.to(device), yb.to(device)
                optimizer.zero_grad()
                out = model(xb)
                loss = criterion(out, yb)
                loss.backward()
                optimizer.step()


train_preds.extend(out.argmax(1).cpu().numpy())
                train_labels.extend(yb.cpu().numpy())

            train_acc = np.mean(np.array(train_preds)
== np.array(train_labels))

            # Validation
            model.eval()
            val_preds, val_labels = [], []
            val_loss = 0.0

            with torch.no_grad():
                for xb, yb in val_loader:
                    xb, yb = xb.to(device),
yb.to(device)
                    out = model(xb)
                    loss = criterion(out, yb)
                    val_loss += loss.item()


val_preds.extend(out.argmax(1).cpu().numpy())

val_labels.extend(yb.cpu().numpy())

            avg_val_loss = val_loss / max(1,
len(val_loader))
            val_acc = np.mean(np.array(val_preds) ==
np.array(val_labels))
            val_f1 = f1_score(val_labels, val_preds,
average="macro")

            print(
                f"Fold {fold} | Epoch {epoch+1} | "
                f"Loss: {avg_val_loss:.4f} | "
```

```
                f"Train Acc: {train_acc:.4f} | "
                f"Val Acc: {val_acc:.4f} | "
                f"F1: {val_f1:.4f}"
            )

            last_train_acc = train_acc
            last_val_acc = val_acc

            # Early Stopping
            if avg_val_loss < best_loss:
                best_loss = avg_val_loss
                best_state = {
                    k: v.cpu().clone() for k, v in
model.state_dict().items()
                }
                patience_counter = 0
            else:
                patience_counter += 1
                if patience_counter >= patience:
                    print(f"Early stopping pada Fold
{fold}, Epoch {epoch+1}")
                    break

        train_acc_per_fold.append(last_train_acc)
        val_acc_per_fold.append(last_val_acc)

        # Evaluasi fold
        if best_state is None:
            best_state = model.state_dict()

        model.load_state_dict(best_state)
        model.eval()

        preds, labels = [], []
        with torch.no_grad():
            for xb, yb in val_loader:
                xb = xb.to(device)
                out = model(xb)

preds.extend(out.argmax(1).cpu().numpy())
                labels.extend(yb.numpy())

        fold_f1.append(f1_score(labels, preds,
average="macro"))
        fold_acc.append(np.mean(np.array(preds) ==
np.array(labels)))

        if fold_f1[-1] > best_global_f1:
            best_global_f1 = fold_f1[-1]
```

```
            best_global_state = best_state.copy()

        all_preds.extend(preds)
        all_labels.extend(labels)

    # Classification Report
    print("\n=== CLASSIFICATION REPORT ===")
    print(classification_report(all_labels,
all_preds, target_names=target_names, digits=4))

    # Confusion Matrix
    cm = confusion_matrix(all_labels, all_preds)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt="d",
cmap="Blues",
                xticklabels=target_names,
yticklabels=target_names)
    plt.xlabel("Prediksi")
    plt.ylabel("Aktual")
    plt.title("Confusion Matrix")
    plt.tight_layout()
    plt.show()

    # grafik Train dan Validasi
    folds = list(range(1, n_splits + 1))

    plt.figure(figsize=(8, 6))
    plt.plot(folds, train_acc_per_fold, marker="o",
label="Training Accuracy", linewidth=2.2)
    plt.plot(folds, val_acc_per_fold, marker="s",
label="Validation Accuracy", linewidth=2.2)

    offset = 0.0035
    for i, (tr, va) in
enumerate(zip(train_acc_per_fold, val_acc_per_fold),
1):

        plt.text(i, tr + offset, f"{tr:.3f}",
                 ha="center", va="bottom",
fontsize=9.5, fontweight="medium")

        plt.text(i, va - offset*1.3, f"{va:.3f}",
                 ha="center", va="top", fontsize=9.5,
fontweight="medium",
                 color="#d62728")

    plt.xlabel("Fold ke-", fontsize=11)
    plt.ylabel("Accuracy", fontsize=11)
```

```
    plt.title("K-Fold Cross Validation Performance",
fontsize=13, pad=15)
    plt.xticks(folds)
    plt.ylim(min(min(train_acc_per_fold),
min(val_acc_per_fold)) - 0.02,
             max(max(train_acc_per_fold),
max(val_acc_per_fold)) + 0.03)
    plt.legend(loc="upper right", fontsize=10)
    plt.grid(True, alpha=0.3, linestyle="--")
    plt.tight_layout()
    plt.show()

    # Grafik F1 dan Acc
    plt.figure(figsize=(7, 5))
    plt.plot(folds, fold_f1, marker="o", label="F1-
macro")
    plt.plot(folds, fold_acc, marker="s",
label="Accuracy")

    for i, (f1, acc) in enumerate(zip(fold_f1,
fold_acc), start=1):
        plt.text(i, f1 + 0.01, f"{f1:.3f}",
ha="center", fontsize=9)
        plt.text(i, acc - 0.03, f"{acc:.3f}",
ha="center", fontsize=9)

    plt.xlabel("Fold")
    plt.ylabel("Score")
    plt.title("Performa Model per Fold")
    plt.xticks(folds)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    checkpoint = {
    "state_dict": best_global_state,
    "fold_f1": fold_f1,
    "fold_acc": fold_acc,
    "train_acc_per_fold": train_acc_per_fold,
    "val_acc_per_fold": val_acc_per_fold,
    "all_preds": all_preds,
    "all_labels": all_labels,
    "config": {
        "kernel_size": kernel_size,
        "batch_size": batch_size,
        "use_smote": use_smote,
        "n_splits": n_splits,
        "num_epochs": num_epochs,
```

```
        "patience": patience,
        "lr": lr
    }
}

    return checkpoint
```

```
# Run Evaluasi Kernel 2 dan SMOTE
checkpoint = run_simplecnn_kfold(
    X, y,
    kernel_size=2,
    batch_size=64,
    use_smote=True,
    n_splits=5,
    num_epochs=10,
    patience=3,
    lr=0.001,
    device="cpu"
)


# Run Evaluasi Kernel 5 dan SMOTE
checkpoint = run_simplecnn_kfold(
    X, y,
    kernel_size=3,
    batch_size=64,
    use_smote=True,
    n_splits=5,
    num_epochs=10,
    patience=3,
    lr=0.001,
    device="cpu"
)

# Run Evaluasi Kernel 5 dan SMOTE
checkpoint = run_simplecnn_kfold(
    X, y,
    kernel_size=5,
    batch_size=64,
    use_smote=True,
    n_splits=5,
    num_epochs=10,
    patience=3,
    lr=0.001,
    device="cpu"
)
```

```
# Run Evaluasi Kernel 2
checkpoint = run_simplecnn_kfold(
    X, y,
    kernel_size=2,
    batch_size=64,
    use_smote=False,
    n_splits=5,
    num_epochs=10,
    patience=3,
    lr=0.001,
    device="cpu"
)

# Run Evaluasi Kernel 3
checkpoint = run_simplecnn_kfold(
    X, y,
    kernel_size=3,
    batch_size=64,
    use_smote=False,
    n_splits=5,
    num_epochs=10,
    patience=3,
    lr=0.001,
    device="cpu"
)

# Run Evaluasi Kernel 5
checkpoint = run_simplecnn_kfold(
    X, y,
    kernel_size=5,
    batch_size=64,
    use_smote=False,
    n_splits=5,
    num_epochs=10,
    patience=3,
    lr=0.001,
    device="cpu"
)
```