Nama : Rama Syailana Dewa

NIM : 22.11.5017

Kelas : Informatika 08

UJIAN TENGAH SEMESTER

PROYEK DATA MINING

1. **Judul Proyek Data Mining**

   "PERBANDINGAN HYPERPARAMETER TUNING BERBASIS TRANSFER LEARNING UNTUK KLASIFIKASI CITRA BATIK"

2. **Latar Belakang**

Batik merupakan salah satu daya tarik dan ciri khas Indonesia di skala internasional dan untuk melindungi itu, secara resmi UNESCO menetapkan batik sebagai warisan budaya tak benda (Masterpieces of the Oral and the Intangible Heritage of Humanity) [1]. Usaha menjaga kelestarian batik ditempuh melalui pemeliharaan berkelanjutan terhadap karakteristik khas dari pola batik itu sendiri, sekaligus dengan pengenalan yang tiada henti kepada generasi selanjutnya [2]. Namun masyarakat ingin melestarikan seringkali merasa bingung dalam menentukan dan membedakan dari daerah mana batik tersebut berasal, dan juga dikarenakan adanya kemiripan pada pola atau motif batik [3].

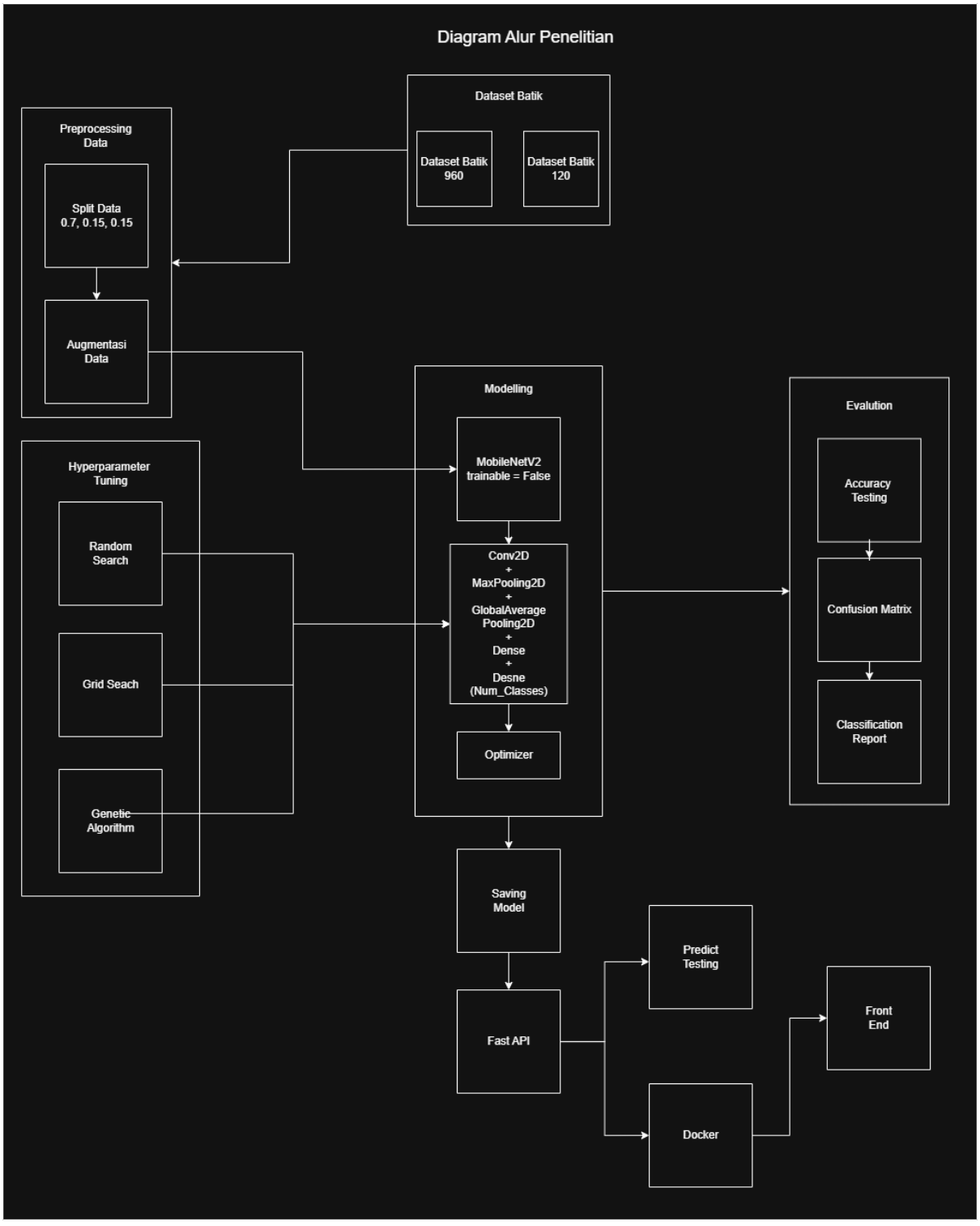
Dataset batik sendiri memiliki ragam hias geometri yang kaya akan variasi, tingkat kesamaan antar motif yang cukup tinggi, serta pola dalam berbagai ukuran dan variasi warna yang kaya [4]. Dalam hal ini sangat diperlukan algoritma untuk menunjang dalam membantu masyarakat mengenali batik dalam waktu yang singkat dan akurat. Oleh karena itu, Transfer Learning digunakan sebagai solusi dengan mengadaptasi model pra-terlatih dari dataset besar untuk tugas spesifik seperti klasifikasi citra batik, hal ini memungkinkan pelatihan yang lebih cepat dan performa yang lebih baik bahkan dengan dataset yang terbatas.

Pada penelitian [5] menunjukkan akurasi yang belum maksimal untuk membedakan batik meskipun sudah menggunakan Transfer Learning. Keberhasilan arsitektur Transfer Learning dalam mencapai hasil optimal juga ditentukan oleh ketepatan pemilihan

hyperparameter pendekatan dengan karakteristik dan efisiensi yang berbeda untuk mencapai konfigurasi terbaik [6]. Meskipun Transfer Learning sangat bermanfaat dalam memprediksi batik, masih perlu ada yang diperhatikan untuk meningkatkan performa model yang dihasilkan, yaitu pemilihan hyperparameter yang tepat, seperti learning rate, ukuran batch, dan konfigurasi optimal.

Penelitian ini bertujuan untuk melakukan perbandingan berbagai metode Hyperparameter Tuning dalam konteks penerapan Transfer Learning untuk klasifikasi citra batik. dengan menggunakan metode mulai dari Grid Search, Random Search, dan Genetic Algorithm, diharapkan dapat memberikan panduan dalam meningkatkan akurasi sistem klasifikasi batik otomatis dan mendukung upaya pelestarian serta pemanfaatan warisan budaya ini secara lebih luas.

3. Diagram Alur Penelitian



4. Lampiran Analisa Code program

4.1 Data Splitting

```python
# Path awal
original_dataset_dir = 'Batik Nitik Grouped'
output_base_dir = 'dataset_split'

# Rasio split
train_ratio = 0.7
val_ratio = 0.15
test_ratio = 0.15

# Buat folder train/val/test
for split in ['train', 'val', 'test']:
    for class_name in os.listdir(original_dataset_dir):
        os.makedirs(os.path.join(output_base_dir, split, class_name), exist_ok=True)

# Bagi dataset per kelas
for class_name in os.listdir(original_dataset_dir):
    class_path = os.path.join(original_dataset_dir, class_name)
    if not os.path.isdir(class_path):
        continue

    files = os.listdir(class_path)
    files = [f for f in files if os.path.isfile(os.path.join(class_path, f))]
    random.shuffle(files)

    train_files, temp_files = train_test_split(files, test_size=(1 - train_ratio), random_state=42)
    val_files, test_files = train_test_split(temp_files, test_size=(test_ratio / (test_ratio + val_ratio)), random_state=42)

    for f in train_files:
        shutil.copy(os.path.join(class_path, f), os.path.join(output_base_dir, 'train', class_name, f))
    for f in val_files:
        shutil.copy(os.path.join(class_path, f), os.path.join(output_base_dir, 'val', class_name, f))
    for f in test_files:
        shutil.copy(os.path.join(class_path, f), os.path.join(output_base_dir, 'test', class_name, f))
```

4.2 Data Augmentation

```python
train_datagen = ImageDataGenerator(
    rescale=1./255,                # Normalisasi piksel gambar dari 0-255 menjadi 0-1
    rotation_range=20,             # Rotasi gambar acak hingga 20 derajat
    zoom_range=0.2,                # Zoom acak hingga 20% untuk mensimulasikan perbedaan jarak
    width_shift_range=0.2,         # Geser gambar secara horizontal hingga 20% lebar gambar
    height_shift_range=0.2,        # Geser gambar secara vertikal hingga 20% tinggi gambar
    shear_range=0.15,              # Distorsi gambar secara miring (shear)
    horizontal_flip=True,          # Membalik gambar secara horizontal (misalnya daun kiri dan kanan)
    brightness_range=[0.8, 1.2],   # Variasi pencahayaan gambar
    fill_mode='nearest'            # Isi area kosong hasil transformasi dengan piksel terdekat
)

val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

4.3 Main Arsitektur

```
# Load pre-trained MobileNetV2
base_model = MobileNetV2(
    input_shape=(*IMG_SIZE, 3),
    include_top=False,
    weights='imagenet'
)

# 1. Freeze semua layer pada awalnya
base_model.trainable = False

# 2. Bangun model dengan layer tambahan
model = Sequential([
    base_model,  # feature extractor

    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),

    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dense(60, activation='softmax')  # Ubah sesuai jumlah kelas kamu
])

# Kompilasi model dengan Adam optimizer dan categorical crossentropy
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',  # Cocok untuk multi-class classification
    metrics=['accuracy']              # Evaluasi menggunakan akurasi
)
```

4.4 Model builder Hyperparameter Tuning

```python
# ========== MODEL BUILDER UNTUK HYPERPARAMETER TUNING ==========
def build_model(hp):
    # Load pre-trained MobileNetV2 dengan freezing
    base_model = MobileNetV2(
        input_shape=(*IMG_SIZE, 3),
        include_top=False,
        weights='imagenet'
    )

    # Freeze semua layer pada base model
    base_model.trainable = False

    model = Sequential()
    model.add(base_model)

    # Tambahkan ConvLayer (opsional)
    if hp.Boolean('add_conv_layer', default=True):
        filters = hp.Int('conv_filters', min_value=32, max_value=128, step=32, default=64)
        model.add(Conv2D(filters, (3, 3), activation='relu', padding='same'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

    # GlobalAveragePooling2D untuk flatten feature map
    model.add(GlobalAveragePooling2D())

    # Dense layers
    dense_units = hp.Int('dense_units', min_value=64, max_value=512, step=64, default=128)
    model.add(Dense(dense_units, activation='relu'))

    # Dropout untuk regularisasi
    dropout_rate = hp.Float('dropout_rate', min_value=0.0, max_value=0.5, step=0.1, default=0.2)
    if dropout_rate > 0:
        model.add(Dropout(dropout_rate))

    # Layer output
    model.add(Dense(NUM_CLASSES, activation='softmax'))

    # Kompilasi dengan hyperparameter
    learning_rate = hp.Float('learning_rate', min_value=1e-4, max_value=1e-2, sampling='log', default=1e-3)

    optimizer_choice = hp.Choice('optimizer', values=['adam', 'rmsprop', 'sgd'])
    if optimizer_choice == 'adam':
        optimizer = Adam(learning_rate=learning_rate)
    elif optimizer_choice == 'rmsprop':
        optimizer = RMSprop(learning_rate=learning_rate)
    else:
        optimizer = SGD(learning_rate=learning_rate, momentum=0.9)

    model.compile(
        optimizer=optimizer,
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
```

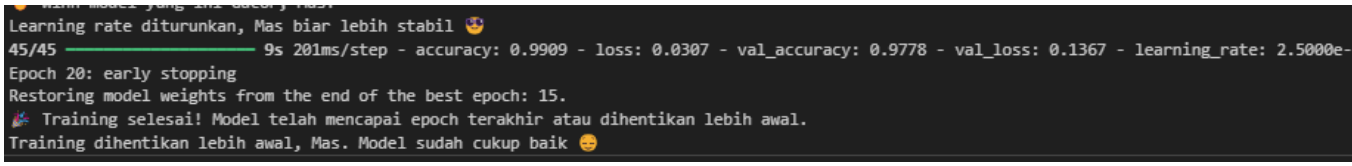
4.5 Saving & Initiate Tuner

```
# ============ RandomSearch TUNER ============

# Buat direktori untuk menyimpan hasil tuning
project_dir = "hyperparameter_tuning_randomsearch_results"
os.makedirs(project_dir, exist_ok=True)

# Buat objek RandomSearch tuner
tuner = RandomSearch(
    build_model,
    objective='val_accuracy',
    max_epochs=15,  # Maksimum epoch untuk setiap trial
    factor=3,       # Faktor eliminasi untuk RandomSearch
    directory=project_dir,
    project_name=f'mobilenetv2_tuning_{int(time.time())}'
)
```

4.6 Running a Tuner

```
# Jalankan pencarian hyperparameter
tuner.search(
    train_generator,
    validation_data=val_generator,
    epochs=30,  # Maksimum epoch (akan dihentikan lebih awal oleh early stopping)
    callbacks=[early_stopping, reduce_lr, TuningCallback('RandomSearch')],
)

# Tampilkan ringkasan hasil
print("\n==== HASIL HYPERPARAMETER TUNING ====")
tuner.results_summary()

# Ambil model terbaik
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
print("\n==== HYPERPARAMETER TERBAIK ====")
print(f"Learning rate: {best_hps.get('learning_rate')}")
print(f"Optimizer: {best_hps.get('optimizer')}")
print(f"Dropout rate: {best_hps.get('dropout_rate')}")
print(f"Dense units: {best_hps.get('dense_units')}")
print(f"Tambah conv layer: {best_hps.get('add_conv_layer')}")
if best_hps.get('add_conv_layer'):
    print(f"Conv filters: {best_hps.get('conv_filters')}")

# Build model terbaik dengan hyperparameter optimal
best_model = tuner.hypermodel.build(best_hps)
```

4.7 Before Tuning

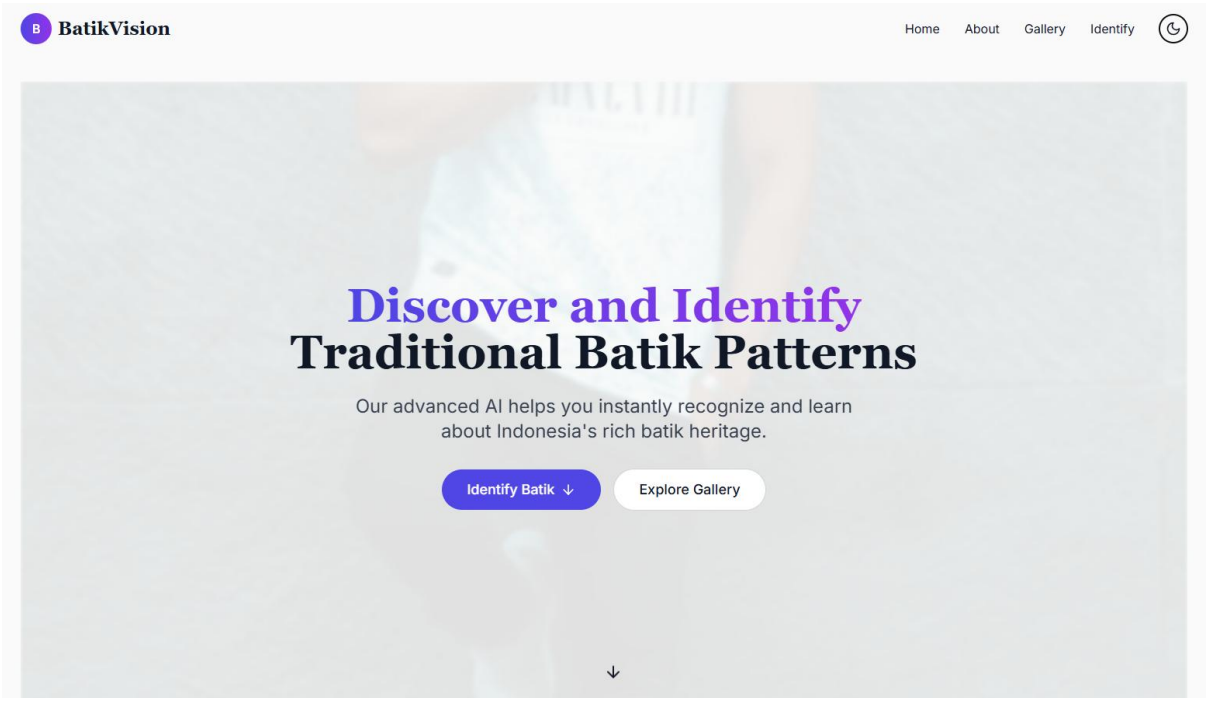


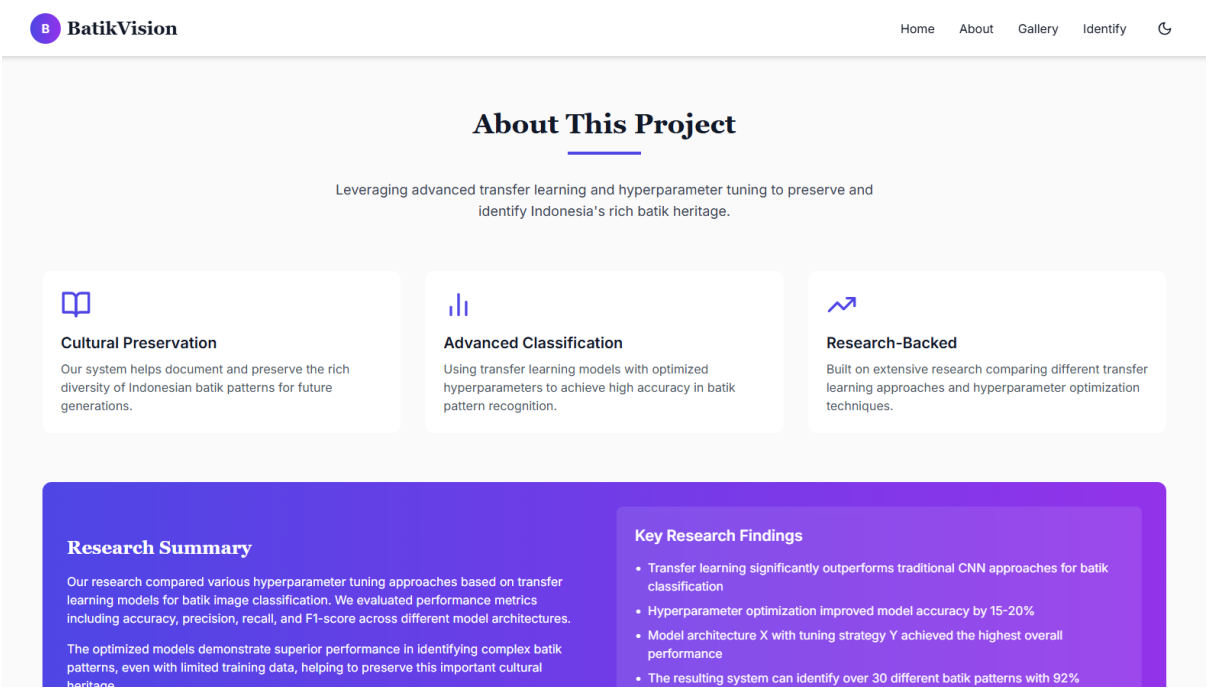
4.8 After Tuning

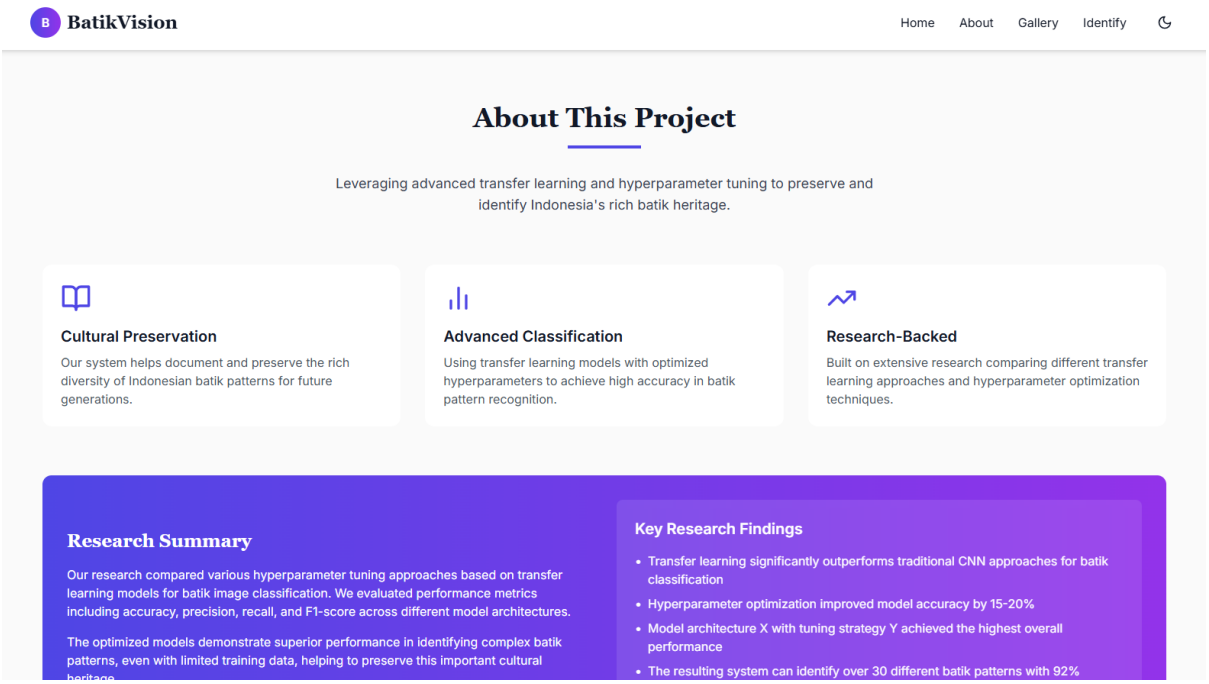
```
Best val_accuracy So Far: 0.9888888597488403
Total elapsed time: 00h 33m 02s

==== HASIL HYPERPARAMETER TUNING ====
Results summary
Results in hyperparameter_tuning_randomsearch_results/mobilenetv2_tuning_1747835340
Showing 10 best trials
Objective(name="val_accuracy", direction="max")

Trial 0024 summary
Hyperparameters:
add_conv_layer: False
conv_filters: 96
dense_units: 384
dropout_rate: 0.30000000000000004
learning_rate: 0.000334067220168496
optimizer: rmsprop
tuner/epochs: 15
tuner/initial_epoch: 5
tuner/bracket: 1
tuner/round: 1
tuner/trial_id: 0022
Score: 0.9888888597488403
```

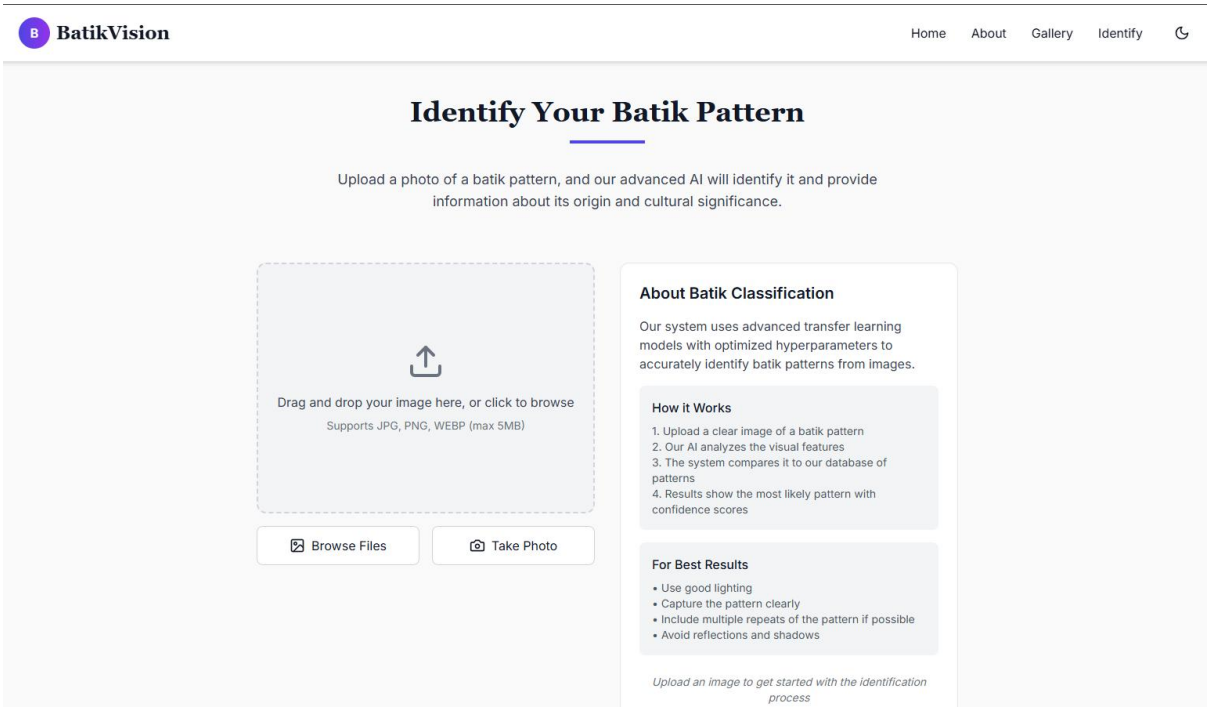
Code Lengkap : https://github.com/MasDewaa/BatikClassification-HyperparameterTuning

5. Rancangan Tampilan UI/UX









Front End Code : https://github.com/MasDewaa/FrontEnd-BatikVision