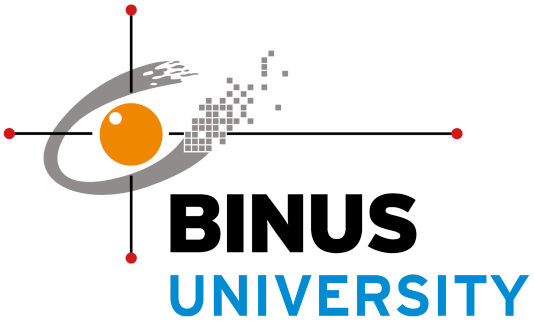
Studi Komparatif Kinerja Model dari Scratch dan Model dengan Parameter Terbatas pada Data Batik Bali, Betawi, dan Celup

LAPORAN TUGAS AKHIR



Oleh:

Justinus Natanael Bensohur	**2602094004**
Kelvin Alexander Bong	**2602101823**
Melvern Amadio Hidayat	**2602072685**

PROGRAM STUDI S1 DATA SCIENCE

SCHOOL OF COMPUTER SCIENCE (SOCS)

BINA NUSANTARA UNIVERSITY

2024

1. PENDAHULUAN

Batik adalah salah satu warisan budaya yang paling berharga di Indonesia. Batik sendiri merupakan perpaduan antara seni dan teknologi yang diciptakan oleh leluhur Indonesia. Setiap daerah di Indonesia memiliki motif batik yang unik dan khas yang mencerminkan budaya, sejarah, dan filosofi daerah tersebut. Contohnya, Batik Bali dikenal dengan motif-motif yang dipengaruhi oleh seni Hindu dan alam, Batik Betawi yang seringkali menampilkan motif-motif yang dipengaruhi oleh budaya Tionghoa dan Arab, serta Batik Celup, atau sering dikenal sebagai Batik Jumputan, yang terkenal dengan teknik pewarnaan yang dicelup sehingga menciptakan pola-pola abstrak yang menarik.

Pengenalan motif batik saat ini masih banyak dilakukan secara manual oleh para ahli batik yang berpengalaman. Proses ini tidak hanya memakan waktu yang banyak, tetapi juga membutuhkan keahlian khusus yang tidak dimiliki oleh semua orang. Seiring dengan perkembangan teknologi, muncul kebutuhan akan metode otomatis yang dapat membantu dalam mengidentifikasi dan mengklasifikasikan motif batik secara cepat dan akurat.

Pada penelitian ini, bertujuan untuk mengembangkan model pembelajaran mesin yang dapat mengklasifikasikan motif batik dari tiga daerah yang berbeda, yaitu Bali, Betawi, dan Celup. Untuk mencapai tujuan ini, kami akan menggunakan pendekatan berbasis citra, atau metode yang menggunakan gambar sebagai input utama, di mana model akan dilatih menggunakan dataset gambar motif batik dari masing-masing daerah. Pendekatan ini melibatkan beberapa langkah utama, yaitu:

1. Pengumpulan Data: Mengumpulkan dan menginput gambar motif batik dari tiga daerah berbeda yaitu Batik Bali, Betawi, dan Celup.

2. Pra-pemprosesan Data: Melakukan pra-pemrosesan pada gambar untuk memastikan kualitas dan konsistensi data.
3. Pembagian Data: Membagi data menjadi data pelatihan dan pengujian untuk melatih dan mengevaluasi model yang akan dirancang.
4. Pelatihan Model: Melatih model pembelajaran mesin menggunakan data pelatihan.
5. Evaluasi Model: Mengevaluasi kinerja model menggunakan data pengujian.
6. Analisis Hasil: Menganalisis hasil dan mengidentifikasi area untuk perbaikan.

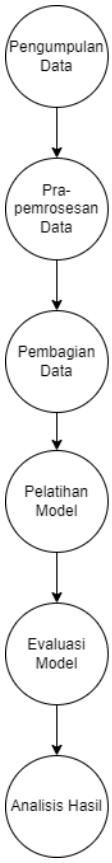
Dataset yang mempunyai 20 kelas atau motif batik yang berbeda, tetapi pada penelitian ini hanya menggunakan 3 motif batik yaitu Bali, Betawi, dan Celup. Setiap gambar akan diberikan label sesuai dengan daerah asal motifnya. Data ini kemudian dibagi menjadi set pelatihan dan pengujian (*train and test data*) untuk memastikan bahwa model dapat diuji dengan data yang tidak terlihat selama pelatihan.

Proses analisa sederhana terhadap data melibatkan pemeriksaan visual dan statistik terhadap gambar-gambar dalam dataset. Berikut adalah beberapa temuan dari analisa awal:

- Keragaman Motif: Motif batik dari setiap daerah memiliki karakteristik yang berbeda-beda. Batik Bali cenderung memiliki motif yang rumit dan detail dengan dominasi warna-warna cerah. Batik Betawi memiliki pola yang lebih sederhana namun kaya dengan makna simbolis. Batik Celup, di sisi lain, menunjukkan pola abstrak dengan teknik pewarnaan yang unik.
- Jumlah Data: Dataset terdiri dari total 150 gambar dengan pembagian yang merata di antara ketiga daerah.

- Pembagian Data: Data dibagi menjadi 80% untuk pelatihan dan 20% untuk pengujian, sehingga set pelatihan terdiri dari 120 gambar dan set pengujian terdiri dari 30 gambar.

2. METODOLOGI



Untuk menyelesaikan masalah klasifikasi motif batik dari tiga daerah berbeda (Bali, Betawi, dan Celup), kami mengadopsi metodologi yang terdiri dari beberapa tahapan utama. Terlampir graph untuk setiap proses penyelesaian masalah yang akan dilakukan.

Berikut adalah penjelasan untuk setiap proses penyelesaian masalah:

- Pengumpulan Data

Proses pengumpulan data melibatkan pengambilan gambar motif batik dari tiga daerah atau kelas, yaitu Bali, Betawi, dan Celup, yang awalnya memiliki 20 motif batik atau kelas. Setiap gambar diberi label berdasarkan daerah asal motifnya. Sumber data akan dimasukkan ke dalam file direktori.

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import shutil


dataset_dir = '/content/batik_dataset'


selected_classes = ['batik-bali', 'batik-betawi', 'batik-celup']


subset_dir = 'batik_choosed'

if not os.path.exists(subset_dir):
    os.makedirs(subset_dir)

# Menyalin gambar dari kelas yang dipilih ke direktori subset
for class_name in selected_classes:
    class_dir = os.path.join(dataset_dir, class_name)
    subset_class_dir = os.path.join(subset_dir, class_name)

    if not os.path.exists(subset_class_dir):
        os.makedirs(subset_class_dir)

    for filename in os.listdir(class_dir):
        src_file = os.path.join(class_dir, filename)
        dst_file = os.path.join(subset_class_dir, filename)
        shutil.copyfile(src_file, dst_file)
```

- Pra-pemprosesan Data

Proses ini melibatkan beberapa pembersihan dan transformasi data untuk memastikan data dapat digunakan untuk pelatihan data. Hal ini juga membuat agar data yang dipakai dapat konsisten resolusinya. Pada tahap ini, akan mengubah ukuran gambar menjadi resolusi standar yaitu 224x224 piksel), dan mengkonversi gambar ke format yang sesuai.

```
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
# Pra-pemrosesan dataset dengan augmentasi
image_size = (224, 224)
batch_size = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

validation_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
```

- Pembagian Data

Pada proses ini, data akan dibagi menjadi beberapa set pelatihan dan pengujian (*train and test dataset*) untuk melatih dan mengevaluasi model. Pada bagian ini, data akan dibagi menjadi 80% data pelatihan dan 20% data pengujian.

```
validation_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = train_datagen.flow_from_directory(
    subset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = validation_datagen.flow_from_directory(
    subset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

- Pelatihan Model

Proses pelatihan model melibatkan penggunaan data pelatihan untuk melatih model pembelajaran mesin yang dapat mengklasifikasikan motif batik. Pelatihan model data yang dipakai adalah Convolutional Neural Network (CNN). Pada pelatihan, terdapat 2 model yang akan dilakukan yaitu model scratch dan model yang dibuat dengan parameter di bawah 10 M. Setelah dijalankan, akan terdapat hasil yaitu akurasi dan loss yang telah diberikan oleh model. Untuk model yang memiliki parameter di bawah 10 M akan menggunakan MobileNetV2 untuk menghitung pretrained model.

Pelatihan Scratch Model:

```
# Model dari Scratch
scratch_model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(3, activation='softmax')
])

scratch_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Menampilkan summary model dari scratch
print("Summary Model dari Scratch:")
scratch_model.summary()
```

Pelatihan Model dengan Parameter dibawah 10M:

```
# Load MobileNetV2 tanpa lapisan atas
base_model = tf.keras.applications.MobileNetV2(input_shape=(224, 224, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Freeze lapisan-lapisan dari model base

# Menambahkan lapisan atas (top) yang baru
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(128, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
predictions = tf.keras.layers.Dense(3, activation='softmax')(x)

# Menggabungkan model
pretrained_model = tf.keras.Model(inputs=base_model.input, outputs=predictions)

pretrained_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Menampilkan summary model pretrained
print("Summary Model Pretrained (MobileNetV2):")
pretrained_model.summary()
```

- Evaluasi Model

Evaluasi model dilakukan menggunakan data pengujian untuk mengukur kinerja model dalam mengklasifikasikan motif batik yang belum pernah dilihat sebelumnya. Pada tahap ini, akan menggunakan model yang telah dilatih untuk memprediksi label data pengujian. Lalu, menggunakan *classification report* untuk melihat a*ccuracy, precision, recall, dan F1-score* yang telah dilakukan terhadap data pengujian. Pada tahap ini, dapat dilihat accuracy dan loss yang terdapat pada model scratch dan pretrained.

```
# Evaluasi model dari scratch
loss_scratch, accuracy_scratch = scratch_model.evaluate(validation_generator)
print(f'Scratch Model - Loss: {loss_scratch}, Accuracy: {accuracy_scratch}')

# Evaluasi model pretrained
loss_pretrained, accuracy_pretrained = pretrained_model.evaluate(validation_generator)
print(f'Pretrained Model (MobileNetV2) - Loss: {loss_pretrained}, Accuracy: {accuracy_pretrained}')
```

```
1/1 [==============================] - 1s 1s/step - loss: 1.0403 - accuracy: 0.5667
Scratch Model - Loss: 1.0402729511260986, Accuracy: 0.5666666626930237
1/1 [==============================] - 1s 1s/step - loss: 0.5518 - accuracy: 0.8000
Pretrained Model (MobileNetV2) - Loss: 0.5517910122871399, Accuracy: 0.800000011920929
```

```
from sklearn.metrics import classification_report

y_true = validation_generator.classes
y_pred_scratch = scratch_model.predict(validation_generator)
y_pred_scratch_classes = y_pred_scratch.argmax(axis=-1)

class_labels = list(validation_generator.class_indices.keys())
print("Classification Report for Scratch Model:")
print(classification_report(y_true, y_pred_scratch_classes, target_names=class_labels))
```

```
1/1 [==============================] - 2s 2s/step
Classification Report for Scratch Model:
              precision    recall  f1-score   support

  batik-bali       0.29      0.50      0.37        10
batik-betawi       0.00      0.00      0.00        10
 batik-celup       0.27      0.30      0.29        10

    accuracy                           0.27        30
   macro avg       0.19      0.27      0.22        30
weighted avg       0.19      0.27      0.22        30
```

```
# Generate predictions and classification report for the pretrained model
y_pred_pretrained = pretrained_model.predict(validation_generator)
y_pred_pretrained_classes = y_pred_pretrained.argmax(axis=-1)

print("Classification Report for Pretrained Model (MobileNetV2):")
print(classification_report(y_true, y_pred_pretrained_classes, target_names=class_labels))
```

```
1/1 [==============================] - 7s 7s/step
Classification Report for Pretrained Model (MobileNetV2):
              precision    recall  f1-score   support

  batik-bali       0.56      0.50      0.53        10
batik-betawi       0.46      0.60      0.52        10
 batik-celup       0.50      0.40      0.44        10

    accuracy                           0.50        30
   macro avg       0.51      0.50      0.50        30
weighted avg       0.51      0.50      0.50        30
```

- Analisis Hasil

Analisis hasil akan mengevaluasi hasil yang telah diberikan untuk memahami model yang telah dilatih atau diuji. Hasil yang akan dievaluasi adalah hasil dari *classification report* yaitu *accuracy, precision, recall, dan F1-score*. Hasil analisis ini akan memberikan wawasan tentang kekuatan dan kelemahan model, serta area yang memerlukan perbaikan lebih lanjut.

3. HASIL DAN ANALISA

```
# Evaluasi model dari scratch
loss_scratch, accuracy_scratch = scratch_model.evaluate(validation_generator)
print(f'Scratch Model - Loss: {loss_scratch}, Accuracy: {accuracy_scratch}')

# Evaluasi model pretrained
loss_pretrained, accuracy_pretrained = pretrained_model.evaluate(validation_generator)
print(f'Pretrained Model (MobileNetV2) - Loss: {loss_pretrained}, Accuracy: {accuracy_pretrained}')
```

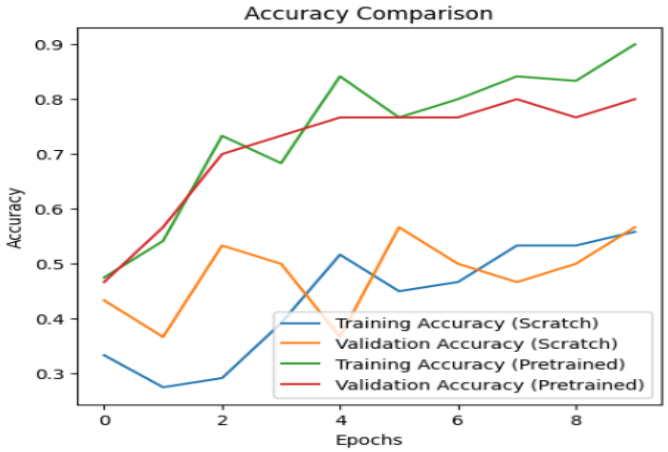
```
1/1 [==============================] - 1s 1s/step - loss: 1.0403 - accuracy: 0.5667
Scratch Model - Loss: 1.0402729511260986, Accuracy: 0.5666666626930237
1/1 [==============================] - 1s 1s/step - loss: 0.5518 - accuracy: 0.8000
Pretrained Model (MobileNetV2) - Loss: 0.5517910122871399, Accuracy: 0.800000011920929
```

Setelah melakukan training / pelatihan yang dilakukan, kami melakukan evaluasi untuk kedua model baik yang model dari scratch dan yang pretrained dengan parameter kurang dari 10m menggunakan MobileNetV2. Melalui evaluasi, dapat terlihat bahwa Model Pretrained memiliki tingkat akurasi yang lebih tinggi yaitu mencapai 80% dan loss yang rendah yaitu 0.55, sementara untuk model yang scratch mendapatkan Accuracy 57% dan loss yang cukup tinggi yaitu 1.04.

Lalu setelah menampilkan hasil evaluasi, untuk lebih memperjelas lagi, kami membuat 2 chart yang dimana masing-masing menampilkan hasil evaluasi dari model scratch dan model pre trained dengan seiring bertambahnya epoch.

```
[10] #Accuration Plot
     plt.figure(figsize=(12, 5))
     plt.subplot(1, 2, 1)
     plt.plot(history_scratch.history['accuracy'], label='Training Accuracy (Scratch)')
     plt.plot(history_scratch.history['val_accuracy'], label='Validation Accuracy (Scratch)')
     plt.plot(history_pretrained.history['accuracy'], label='Training Accuracy (Pretrained)')
     plt.plot(history_pretrained.history['val_accuracy'], label='Validation Accuracy (Pretrained)')
     plt.title('Accuracy Comparison')
     plt.xlabel('Epochs')
     plt.ylabel('Accuracy')
     plt.legend()
```

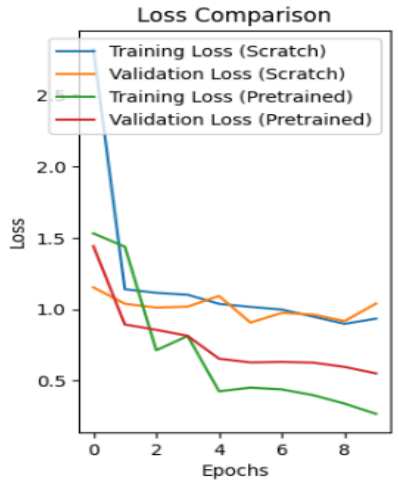
```
<matplotlib.legend.Legend at 0x7b3c3aa588e0>
```



Pertama adalah chart untuk Accuration. Seperti yang sudah dibahas sebelumnya, Accuracy untuk Model Pretrained lebih tinggi dibandingkan dengan Model from scratch. Dalam chart, Training Accuracy untuk model pre trained mencapai 90% dan untuk Validation Accuracy mencapai 80%.Tidak hanya hasil yang cukup tinggi, melainkan adanya konsistensi dari Akurasi model yang menaik seiring bertambahnya epoch. Model dari scratch baik dari Training dan Validation kurang lebih berada di 50% yang dimana itu angka yang cukup kecil.

```
#Loss Plot
plt.subplot(1, 2, 2)
plt.plot(history_scratch.history['loss'], label='Training Loss (Scratch)')
plt.plot(history_scratch.history['val_loss'], label='Validation Loss (Scratch)')
plt.plot(history_pretrained.history['loss'], label='Training Loss (Pretrained)')
plt.plot(history_pretrained.history['val_loss'], label='Validation Loss (Pretrained)')
plt.title('Loss Comparison')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



Kedua adalah chart untuk Loss. Berbeda dengan chart Accuracy yang menanjak, baik kedua model mengalami penurunan Loss. Tetapi Loss yang mengalami penurunan paling besar adalah milik model Pre Trained MobileNetV2, dimana hampir mencapai 0% untuk Training Loss dan untuk Validation Loss model Pretrained berada di 0.5 . Training Loss dan Validation Loss untuk Model from Scratch masih di angka 1 untuk loss-nya.

Pembuktian Nyata untuk prediksi Image batik untuk kedua model dapat dilihat sebagai berikut.

Model from Scratch

Mengambil 10 sampel gambar, Model dari Scratch memiliki akurasi prediksi memiliki persentase 50%

Sementara untuk Model Pre Trained MobileNetV2, model ini memiliki akurasi prediksi hingga 90% dari 10 sample image

4. KESIMPULAN

Dalam membandingkan model image segmentation dari scratch dengan model yang sudah dilatih sebelumnya seperti MobileNetV2, ada beberapa perbedaan utama yang dapat kita identifikasi. Model yang dilatih dari awal/scratch memiliki keuntungan dalam memberikan fleksibilitas penuh dalam membangun arsitektur dan parameternya, memungkinkan penyesuaian yang lebih mendalam sesuai dengan dataset spesifik yang digunakan. Namun, saat training model dari scratch, model ini memerlukan data dalam jumlah besar dan waktu yang signifikan, serta adanya risiko konvergensi yang lebih lambat atau tidak optimal.

Sebaliknya, model MobileNetV2 yang sudah dilatih/ pretrained menawarkan keunggulan dalam hal efisiensi waktu dan sumber daya, karena telah dilatih pada dataset besar, sehingga dapat menghemat waktu pelatihan dan biasanya mencapai kinerja yang baik bahkan dengan data yang lebih sedikit. Hal ini dapat dilihat dari hasil Classification Report, Evaluation & Training & Validation Accuracy plot dari kedua model, dimana model pretrained jauh lebih baik. Meskipun demikian, model pretrained mungkin memerlukan fine-tuning untuk menyesuaikan dengan tugas segmentasi khusus yang mungkin memiliki karakteristik yang berbeda dari data pelatihan scratch.

5. REFERENSI

- https://bbkb.kemenperin.go.id/index.php/post/read/pengertian_motif_batik_dan_filosofinya_0
- https://akademi-ai.medium.com/deep-learning-klasifikasi-gambar-dengan-convolutional-neural-network-menggunakan-keras-python-87bb08a47ca6
- https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c
- https://www.batikprabuseno.com/artikel/edukasi/batik-jumputan/
- https://keras.io/examples/vision/oxford_pets_image_segmentation/
- https://www.tensorflow.org/tutorials/images/segmentation
- https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/