I

Indonesian Batik Classification

Multiclass image classification

DEEP
LEARNING

Nama Kelompok

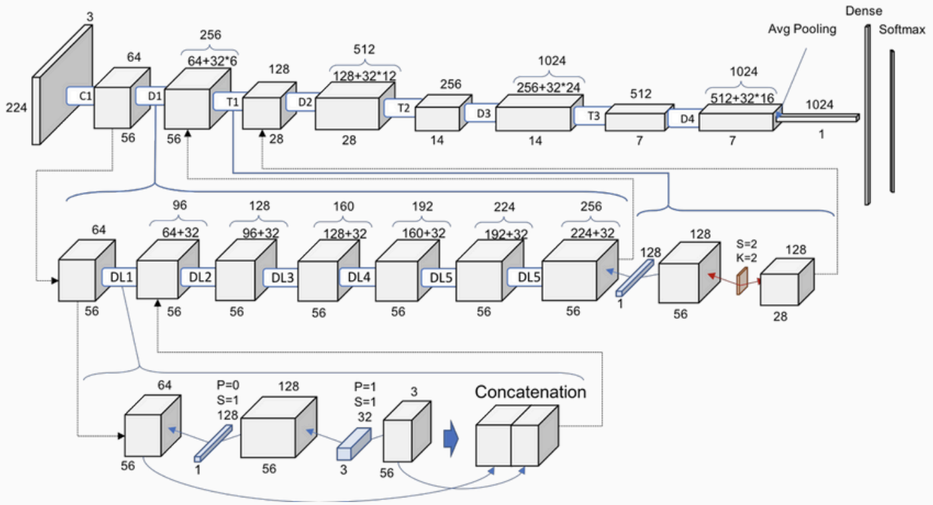
- Andrea octaviani
  2602075895
- Cristian Agusta
  2602157705
- Moh. Khoirul Umam Al Amin
  2602198472

Problem Introduction

Setiap motif batik merepresentasikan identitas dari jenis batik tertentu, mencerminkan nilai-nilai tradisional, serta memiliki makna simbolis yang mendalam. Keberagaman motif pada setiap jenis batik, dengan pola-pola yang unik, sering kali menjadi tantangan bagi mereka yang kurang familiar untuk mengenali atau membedakan setiap jenis batik.

Ini jenis batik apa?!

- **Task :** Multiclass image classification
- **Architecture :** CNN model from scratch and DenseNet 121 (*pretrained model*)



- **Data :** Indonesian Batik dataset

Batik Bali	Batik Lasem
Batik Betawi	Batik Mega Mendung
Batik Celup	Batik Parang
Batik Cendrawasih	Batik Pekalongan
Batik Ceplok	Batik Priangan
Batik Ciamis	Batik Sekar
Batik Garutan	Batik Sidoluhur
Batik Gentongan	Batik Sidomukti
Batik Kawung	Batik Sogan
Batik Keraton	Batik Tambal

20 Class

983 total Image

Method

Model akan dilatih untuk melakukan klasifikasi gambar batik ke dalam tiga kelas.

20 Class ⇨ *3 Class* ⇨ **Batik sidoluhur**, **Batik tambal**, **Batik sogan**

150 Total images

50 Images per class

- Data Augmentation

  Dilakukan augmentasi pada data **training** untuk meningkatkan variasi data dan membantu model untuk lebih mengenali pola dari batik.

```
train_datagen = ImageDataGenerator(
    # Geometric transformations
    rotation_range=20,          # Moderate rotation to preserve pattern integrity
    width_shift_range=0.2,      # Horizontal shift
    height_shift_range=0.2,     # Vertical shift
    shear_range=0.2,            # Slight shear for pattern variation
    zoom_range=0.2,            # Zoom for scale invariance
    horizontal_flip = True,
    vertical_flip=False,
    fill_mode ='nearest'
)
```

- Data Normalization

  melakukan normalisasi pada pixel gambar menjadi dalam rentang **[0,1]**

Model Architecture

Model 1

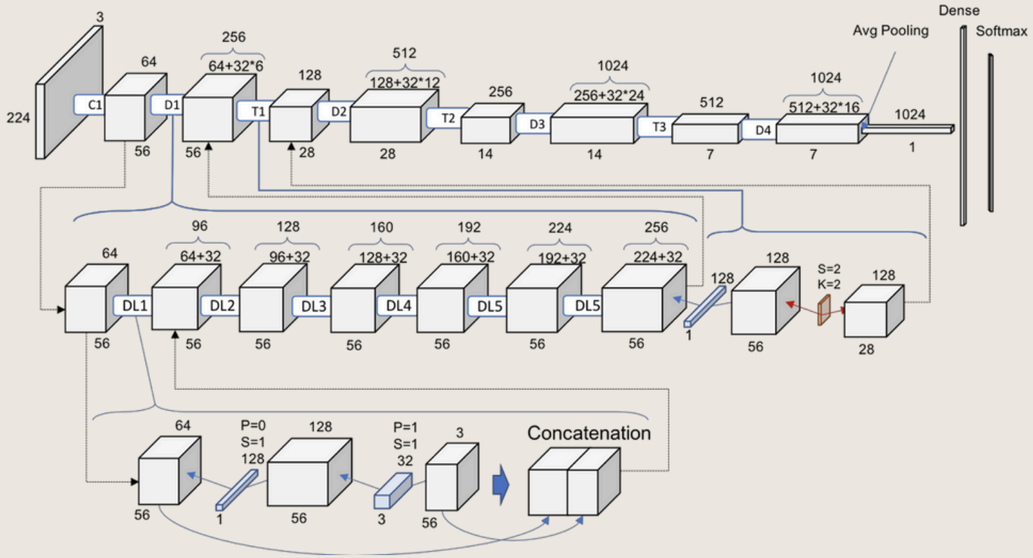
Architecture model from scratch

Layer (type)	Output Shape	Param #
conv2d_80 (Conv2D)	(None, 300, 300, 32)	896
max_pooling2d_80 (MaxPooling2D)	(None, 150, 150, 32)	0
conv2d_81 (Conv2D)	(None, 150, 150, 64)	18,496
max_pooling2d_81 (MaxPooling2D)	(None, 75, 75, 64)	0
conv2d_82 (Conv2D)	(None, 75, 75, 128)	73,856
max_pooling2d_82 (MaxPooling2D)	(None, 37, 37, 128)	0
conv2d_83 (Conv2D)	(None, 37, 37, 256)	295,168
max_pooling2d_83 (MaxPooling2D)	(None, 18, 18, 256)	0
dropout_40 (Dropout)	(None, 18, 18, 256)	0
flatten_30 (Flatten)	(None, 82944)	0
dense_50 (Dense)	(None, 256)	21,233,920
dropout_41 (Dropout)	(None, 256)	0
dense_51 (Dense)	(None, 3)	771

Total params: 21,623,107 (82.49 MB)
Trainable params: 21,623,107 (82.49 MB)
Non-trainable params: 0 (0.00 B)

Model 2

Architecture model using DenseNet121



Total params: 7,286,339 (27.80 MB)
Trainable params: 248,835 (972.01 KB)
Non-trainable params: 7,037,504 (26.85 MB)

Model Training

Data spliting and parameter tuning

Secara garis besar model dilatih dengan menggunakan beberapa format parameter dan spiliting data menjadi 4 jenis pelatihan sebagai berikut

Percobaan 1

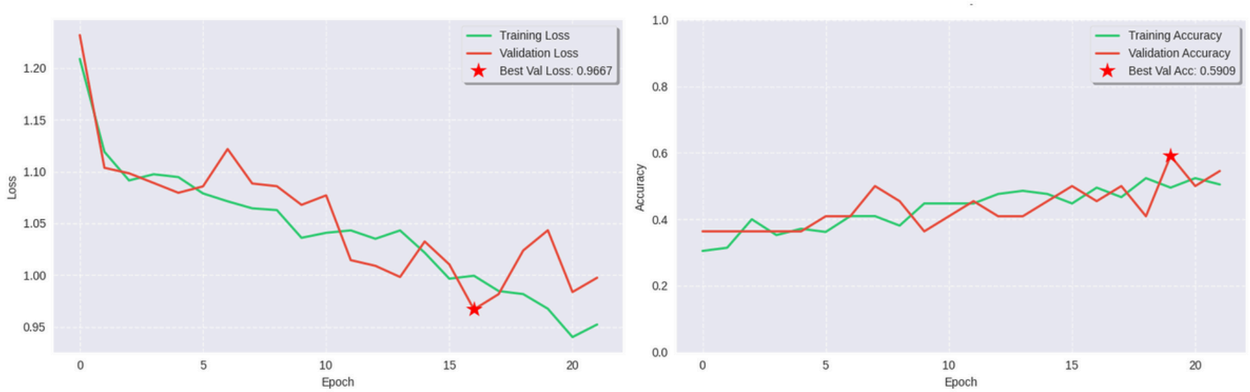
Data splitting :

- 0,7 train set
- 0.15 validation set
- 0.15 test set

Parameter :

- Epoch = 30
- Learning rate = 0.0001
- Batch_size = 32

From scratch model result :




```
Best Validation Loss: 0.9667 (Epoch 17)
Best Validation Accuracy: 0.5909 (Epoch 20)
```

```
loss, accuracy = model.evaluate(test_gen)
print(f'Test Loss: {loss}')
print(f'Test Accuracy: {accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.6000 - loss: 0.
Test Loss: 0.6750512719154358
Test Accuracy: 0.6000000238418579
```

DenseNet 121 result :


```
Best Validation Loss: 0.5557 (Epoch 9)
Best Validation Accuracy: 0.7727 (Epoch 14)
```

```
pt_loss, pt_accuracy = pt_model.evaluate(test_gen)
print(f'Test Loss: {pt_loss}')
print(f'Test Accuracy: {pt_accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 10s 10s/step - accuracy: 0.7391 - loss: 0.9742
Test Loss: 0.9741594195365906
Test Accuracy: 0.739130437374115
```

Percobaan 2

Data splitting :

- 0,7 train set
- 0.15 validation set
- 0.15 test set

Parameter :

- Epoch = 20
- Learning rate = 0.001
- Batch_size = 16

From scratch model result :


```
Best Validation Loss: 0.9803 (Epoch 9)
Best Validation Accuracy: 0.5909 (Epoch 10)
```

```
loss, accuracy = model.evaluate(test_gen)
print(f'Test Loss: {loss}')
print(f'Test Accuracy: {accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 272ms/step - accuracy: 0.6957 - loss: 0.9223
Test Loss: 0.9223247766494751
Test Accuracy: 0.695652186870575
```

DenseNet 121 result :


```
Best Validation Loss: 1.1539 (Epoch 4)
Best Validation Accuracy: 0.8636 (Epoch 4)
```

Percobaan 3

Data splitting :

- 0,8 train set
- 0.1 validation set
- 0.1 test set

Parameter :

- Epoch = 30
- Learning rate = 0.0001
- Batch_size = 32

From scratch model result :


```
Best Validation Loss: 0.8962 (Epoch 19)
Best Validation Accuracy: 0.6000 (Epoch 5)
```

```
loss, accuracy = model.evaluate(test_gen)
print(f'Test Loss: {loss}')
print(f'Test Accuracy: {accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.6667
Test Loss: 0.7495376467704773
Test Accuracy: 0.6666666865348816
```

DenseNet 121 result :


```
Best Validation Loss: 0.4407 (Epoch 8)
Best Validation Accuracy: 0.8667 (Epoch 7)
```

```
pt_loss, pt_accuracy = pt_model.evaluate(test_gen)
print(f'Test Loss: {pt_loss}')
print(f'Test Accuracy: {pt_accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step - accuracy: 0.8000
Test Loss: 0.430902898311615
Test Accuracy: 0.800000011920929
```

Percobaan 4

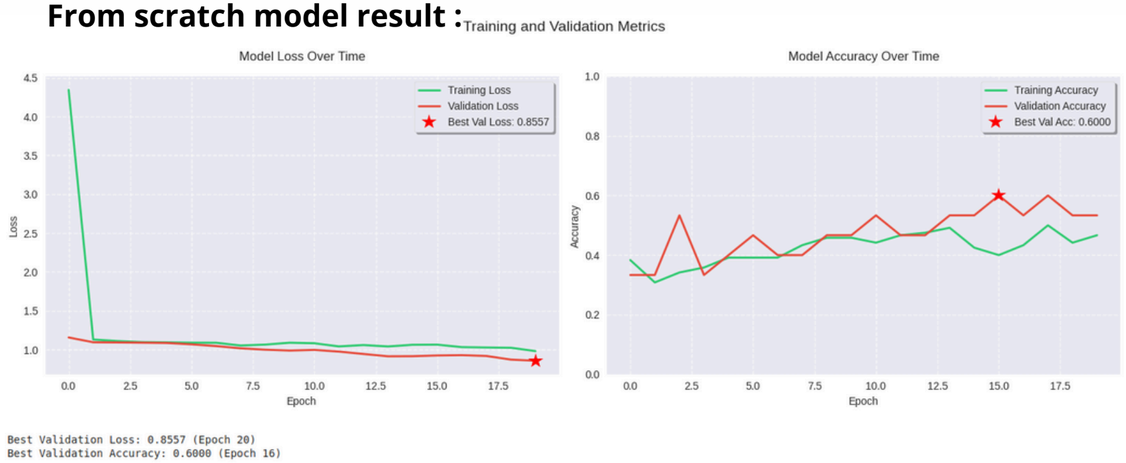
Data splitting :

- 0,8 train set
- 0.1 validation set
- 0.1 test set

Parameter :

- Epoch = 20
- Learning rate = 0.001
- Batch_size = 16

From scratch model result :




```
loss, accuracy = model.evaluate(test_gen)
print(f'Test Loss: {loss}')
print(f'Test Accuracy: {accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.6667
Test Loss: 0.7365491986274719
Test Accuracy: 0.6666666865348816
```

DenseNet 121 result :


```
pt_loss, pt_accuracy = pt_model.evaluate(test_gen)
print(f'Test Loss: {pt_loss}')
print(f'Test Accuracy: {pt_accuracy}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - accuracy: 0.5333
Test Loss: 12.18039608001709
Test Accuracy: 0.5333333611488342
```

XII

Kesimpulan

Berdasarkan eksperimen terhadap 2 model yang telah ditentukan yaitu model dari scratch dan pretrained model DenseNet121 beserta 4 jenis tuningan, secara keseluruhan dan pada kasus terbaik dapat dilihat bahwa pretrained model DenseNet121 memberikan hasil terbaik dengan accuracy yang lebih tinggi dan loss yang lebih rendah dibandingkan dengan model yang dibuat dari scratch.

Tuningan terbaik terjadi pada percobaan ketiga dengan nilai Epoch sebanyak 30, Learning Rate sebesar 0.0001, dan Batch Size sebesar 32, disertai pembagian dataset mencakup 80% train set, 10% validation set, dan 10% test set menghasilkan akurasi sebesar 0.6 dan loss sebesar 0.9 pada model scratch, sementara pada model DenseNet, akurasi yang didapat adalah 0.8 dan loss sebesar 0.4.

Menurut hipotesis kami, nilai yang tidak terlalu tinggi pada akurasi dan cukup tinggi pada loss menandakan bahwa model masih belum efektif dikarenakan kurangnya dataset meskipun telah dilakukan augmentasi terhadap image dataset yang diberikan.