BAB VI
PENUTUP

A. Kesimpulan

Berdasarkan hasil analisis, pembahasan, dan pengujian, dapat disimpulkan bahwa sistem yang dibangun berhasil mengimplementasikan pembelajaran mesin dengan menggunakan metode *Convolutional Neural Network* (CNN) untuk mengklasifikasi jenis-jenis motif batik. Dapat disimpulkan juga bahwa dengan menggunakan arsitektur *ResNet*, hasil dari pembelajaran mesin dapat mencapai akurasi yang lebih baik dari model CNN biasa. Dengan *dataset* berukuran 1394 citra dan pembagian jumlah citra sejumlah 1112 citra untuk training dan 282 untuk validasi (80%:20%), model *Transfer Learning ResNet* dapat menghasilkan akurasi training sebesar 95.52% dan akurasi validasi sebesar 78.723%. Selain itu, sistem juga berhasil dibangun dalam bentuk aplikasi web yang dapat digunakan sebagai alat untuk melakukan klasifikasi gambar batik.

B. Saran

Beberapa saran yang dapat diberikan oleh penulis untuk penelitian dan penggunaan pembelajaran mesin selanjutnya:

1. Fitur penilaian pengguna dapat diimplementasikan pada web agar dapat membantu menunjukkan di motif mana yang memiliki jumlah kesalahan banyak sehingga peneliti dapat memanfaatkan informasi ini untuk melakukan *tuning* terhadap *dataset* maupun model.

DAFTAR PUSTAKA

[1] UNESCO, "UNESCO - Indonesian Batik," 2 September 2009. https://ich.unesco.org/en/RL/indonesian-batik-00170 (diakses 22 Desember 2022).

[2] Badan Pengembangan dan Pembinaan Bahasa, "'Batik.'" https://kbbi.kemdikbud.go.id/entri/batik (diakses 26 Desember 2022).

[3] J. Achjadi, *Batik: Spirit of Indonesia*. Yayasan Batik Indonesia, 1999. [Daring]. Tersedia pada: https://books.google.co.id/books?id=e3ywOwAACAAJ

[4] D. Hooi Ting, S. Fong Lim, T. Siuly Patanmacia, C. Gie Low, dan G. Chuan Ker, "Dependency on smartphone and the impact on purchase behaviour," *Young Consumers*, vol. 12, no. 3, hlm. 193–203, Agu 2011, doi: 10.1108/17473611111163250.

[5] K. O'Shea dan R. Nash, "An Introduction to Convolutional Neural Networks," Nov 2015, [Daring]. Tersedia pada: http://arxiv.org/abs/1511.08458

[6] C. U. Khasanah, E. Utami, dan S. Raharjo, "Implementation of Data Augmentation Using Convolutional Neural Network for Batik Classification," dalam *2020 8th International Conference on Cyber and IT Service Management, CITSM 2020*, Institute of Electrical and Electronics Engineers Inc., Okt 2020. doi: 10.1109/CITSM50537.2020.9268890.

[7] J. Tristanto, J. Hendryli, dan D. E. Herwindiati, "Classification of Batik Motifs Using Convolutional Neural Networks," 2018. [Daring]. Tersedia pada: https://ssrn.com/abstract=3258935http://ssrn.com/link/ITES-2018.html

[8] V. Atliha dan D. Sesok, "Comparison of VGG and ResNet used as Encoders for Image Captioning," dalam *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, IEEE, Apr 2020, hlm. 1–4. doi: 10.1109/eStream50540.2020.9108880.

[9] X. Han dan R. Jin, “A Small Sample Image Recognition Method Based on ResNet and Transfer Learning,” dalam *2020 5th International Conference on Computational Intelligence and Applications (ICCIA)*, IEEE, Jun 2020, hlm. 76–81. doi: 10.1109/ICCIA49625.2020.00022.

[10] W. T. Puspitasari, D. C. R. Novitasari, dan W. D. Utami, “Batik Classification using Texture Analysis and Multiclass Support Vector Machine,” Scitepress, Jan 2020, hlm. 65–71. doi: 10.5220/0008517300650071.

[11] R. Azhar, D. Tuwohingide, D. Kamudi, Sarimuddin, dan N. Suciati, “Batik Image Classification Using SIFT Feature Extraction, Bag of Features and Support Vector Machine,” dalam *Procedia Computer Science*, Elsevier, 2015, hlm. 24–30. doi: 10.1016/j.procs.2015.12.101.

[12] T. Handhayani, J. Hendryli, dan L. Hiryanto, “Comparison of shallow and deep learning models for classification of Lasem batik patterns,” dalam *Proceedings - 2017 1st International Conference on Informatics and Computational Sciences, ICICoS 2017*, Institute of Electrical and Electronics Engineers Inc., Okt 2017, hlm. 11–16. doi: 10.1109/ICICOS.2017.8276330.

[13] *2016 IEEE 8th International Conference on Intelligent Systems (IS)*. IEEE.

[14] A. Y. Wicaksono, N. Suciati, C. Fatichah, K. Uchimura, dan G. Koutaki, “Modified Convolutional Neural Network Architecture for Batik Motif Image Classification,” 2017.

[15] M. A. Rasyidi, R. Handayani, dan F. Aziz, “Identification of batik making method from images using convolutional neural network with limited amount of data,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, hlm. 1300–1307, Jun 2021, doi: 10.11591/eei.v10i3.3035.

[16] I. Keller, *Batik Art & Craft*. Tuttle Publishing, 2012.

[17] I. M. C. Elliott dan B. Brake, *Batik: Fabled Cloth of Java*. dalam NONE. Tuttle Publishing, 2013.

[18] N. J. Nilsson, “INTRODUCTION TO MACHINE LEARNING AN EARLY DRAFT OF A PROPOSED TEXTBOOK,” 1996.

[19] R. Szeliski, *Computer Vision*. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-34372-9.

[20] S. Shanmuganathan dan S. Samarasinghe, Ed., *Artificial Neural Network Modelling*, vol. 628. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-28495-8.

[21] Keinosuke Fukunaga, *The artificial neural network book*, 2 ed. Academic Press, 1990.

[22] S. Khan, H. Rahmani, S. A. A. Shah, dan M. Bennamoun, "A Guide to Convolutional Neural Networks for Computer Vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, hlm. 1–207, Feb 2018, doi: 10.2200/S00822ED1V01Y201712COV015.

[23] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, dan V. Zocca, *Python Deep Learning: Exploring deep learning techniques, neural network architectures and GANs with PyTorch, Keras and TensorFlow*, 2 ed. Packt Publishing Ltd, 2019.

[24] J. Wu, "Introduction to Convolutional Neural Networks," 2017.

[25] R. Razavi-Far, B. Wang, M. E. Taylor, dan Q. Yang, Ed., *Federated and Transfer Learning*, vol. 27. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-11748-0.

[26] Python Software Foundation, "Python Programming Language." 2022.

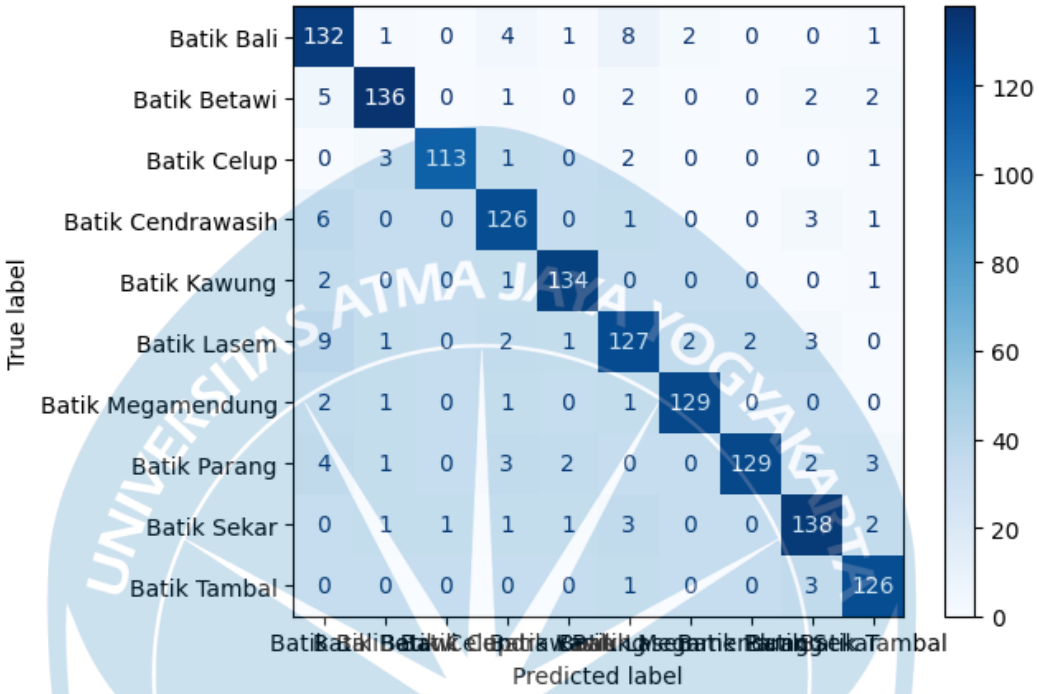
[27] P. Gries, J. Campbell, dan J. Montojo, *Practical Programming: An Introduction to Computer Science Using Python 3 (Pragmatic Programmers)*, Paperback. Pragmatic Bookshelf, The, 2013. [Daring]. Tersedia pada: https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=1937785459

[28] D. Sarkar, R. Bali, dan T. Ghosh, *Hands-On Transfer Learning with Python*. Packt Publishing Ltd., 2018.

[29] Martín~Abadi *dkk.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015. [Daring]. Tersedia pada: https://www.tensorflow.org/

UNIVERSITAS ATMA JAYA YOGYAKARTA

LAMPIRAN



Gambar 25 Confusion matrix model Transfer Learning

Classification Report

	precision	recall	f1-score	support
Batik Bali	0.82	0.89	0.85	149
Batik Betawi	0.94	0.92	0.93	148
Batik Celup	0.99	0.94	0.97	120
Batik Cendrawasih	0.90	0.92	0.91	137
Batik Kawung	0.96	0.97	0.97	138
Batik Lasem	0.88	0.86	0.87	147
Batik Megamendung	0.97	0.96	0.97	134
Batik Parang	0.98	0.90	0.94	144
Batik Sekar	0.91	0.94	0.93	147
Batik Tambal	0.92	0.97	0.94	130
accuracy			0.93	1394
macro avg	0.93	0.93	0.93	1394
weighted avg	0.93	0.93	0.93	1394

Tugas Akhir 190710193

Klasifikasi 10 Kelas Batik

Didesain untuk dijalankan di Google Colab

Setup Library

Install dependensi yang diperlukan seperti:

- tensorflow
- tensorflow-hub
- pillow
- split-folders

```
from google.colab import drive
drive.mount('/content/drive')
```

```
#!pip install tensorflow pillow split-folders tensorflow-hub
```

```
!pip install split-folders
```

Setup Dataset

Sumber dataset: Kaggle 1: https://www.kaggle.com/datasets/dionisiusdh/indonesian-batik-motifs

Kaggle 2: https://www.kaggle.com/datasets/alfanme/indonesian-batik-motifs-corak-app

Google

Download dataset yang telah diproses: Google Drive

```
import zipfile

# COLAB only
!cp '/content/drive/MyDrive/Dataset Tugas Akhir/dataset.zip' '/content/'

dataset_zip_path = 'dataset.zip'

zip_ref = zipfile.ZipFile(dataset_zip_path, 'r')
zip_ref.extractall('dataset')

zip_ref.close()


# # Download Dataset
# # Perlu install kaggle melalui pip untuk melakukan ini
# # Peneliti telah menggabungkan dataset dan disimpan di google drive

# !kaggle datasets download -d dionisiusdh/indonesian-batik-motifs
# !kaggle datasets download -d alfanme/indonesian-batik-motifs-corak-app

# # Extract Dataset

# import zipfile

# # Unzip
# local_zip = '/content/indonesian-batik-motifs-corak-app.zip'
# zip_ref = zipfile.ZipFile(local_zip, 'r')
# zip_ref.extractall('./Batik 1/')

# zip_ref.close()

# local_zip = '/content/indonesian-batik-motifs.zip'
# zip_ref = zipfile.ZipFile(local_zip, 'r')
# zip_ref.extractall('./Batik 2/')

# zip_ref.close()
```

Load model

```
import tensorflow as tf
import tensorflow_hub as hub
```

Pilih model

```
#@title Pilih model
model_type = "Transfer Learning" #@param ["Dense", "CNN", "Transfer Learning"]
color_dimension = 3 #@param ["1", "3"] {type:"raw"}

num_classes = 10

if model_type == "Dense":
  model = tf.keras.models.Sequential([
      # Lapisan input
      tf.keras.layers.Input(shape=(224, 224, color_dimension)),

      # 512 neuron dense layer
      tf.keras.layers.Dense(512, activation='relu'),

      tf.keras.layers.Dropout(0.2),

      tf.keras.layers.Flatten(),

      # 10 Neuron untuk 10 output kelas klasifikasi
      tf.keras.layers.Dense(num_classes, activation='softmax')
  ], name="Model_Dense_Layer")

elif model_type == "CNN":
  model = tf.keras.models.Sequential([
      # Lapisan input
      tf.keras.layers.Input(shape=(224, 224, color_dimension)),

      # Lapisan Convolution
      tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
      tf.keras.layers.MaxPooling2D(2,2),

      # Lapisan Convolution
      tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
      tf.keras.layers.MaxPooling2D(2,2),

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

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

      tf.keras.layers.Dropout(0.2),

      # Lapisan flatten
      tf.keras.layers.Flatten(),

      # 512 neuron hidden layer
      tf.keras.layers.Dense(512, activation='relu'),

      # 10 Neuron untuk 10 output kelas klasifikasi
      tf.keras.layers.Dense(num_classes, activation='softmax')
  ], name="Model_CNN_Layer")

elif model_type == "Transfer Learning":
  model = tf.keras.Sequential([
      hub.KerasLayer("https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector/5",
                     trainable=False,
                    #  arguments=dict(batch_norm_momentum=0.97),
                     input_shape=(224, 224, 3), ),
      tf.keras.layers.Dense(num_classes, activation='softmax')
  ], name="Model_Transfer_Learning")
  model.build([None, 224, 224, color_dimension])  # Batch input shape.

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

model.summary()
```

Load image

```
# Setup folders
# !cp -a '/content/drive/MyDrive/Dataset Tugas Akhir/Dataset Processed' '/content/'
# !rm '/content/Dataset Processed/190710193_BAB 0.docx'

import splitfolders
import os

dataset_path = 'dataset'
split_path = 'dataset_split'

import shutil
try:
    shutil.rmtree(split_path)
except FileNotFoundError:
    pass

# Split with a ratio.
splitfolders.ratio(dataset_path, output="dataset_split",
    seed=193, ratio=(.8, .2), group_prefix=None, move=False)


from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_dir = os.path.join(split_path, 'train')
validation_dir = os.path.join(split_path, 'val')

# Preprocessing
train_datagen = ImageDataGenerator(
                    rescale = 1.0/255.0,
                    shear_range=0.2,
                    zoom_range=0.2,
                    horizontal_flip=True,
                    fill_mode='nearest'
                )

test_datagen  = ImageDataGenerator(rescale = 1.0/255.0)


# Create dataset
if color_dimension == 3:
  color_mode = 'rgb'
else:
  color_mode = 'grayscale'

train_generator = train_datagen.flow_from_directory(
                        train_dir,
                        class_mode = 'categorical',
                        target_size = (224, 224),
                        color_mode = color_mode
                      )

validation_generator =  test_datagen.flow_from_directory(
                          validation_dir,
                          class_mode = 'categorical',
                          target_size = (224, 224),
                          color_mode = color_mode
                        )
```

Train model

```
# !rm -rf '/content/0-19 April 2023-04'


import datetime

ts = int(round(datetime.datetime.now().timestamp()))

checkpoint_filepath = 'tmp/' + 'TL-' + str(ts) +'/model-{epoch:02d}'
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True,
    verbose=1
  )

history = model.fit(
            train_generator,
            epochs=45,
```

```
              validation_data=validation_generator,
              verbose=1,
              callbacks=[model_checkpoint_callback]
             )
```

```
# !cp -r '/content/tmp/CNN1683598098/model-36' '/content/drive/MyDrive/Dataset Tugas Akhir/CNN Final/'
```

Evaluating Accuracy and Loss for the Model

You will plot the training/validation accuracy and loss as collected during training:

```
import matplotlib.pyplot as plt

#-----------------------------------------------------------
# Retrieve a list of list results on training and test data
# sets for each training epoch
#-----------------------------------------------------------
acc      = history.history[     'accuracy' ]
val_acc  = history.history[ 'val_accuracy' ]
loss     = history.history[    'loss' ]
val_loss = history.history['val_loss' ]

epochs   = range(len(acc)) # Get number of epochs

#------------------------------------------------
# Plot training and validation accuracy per epoch
#------------------------------------------------
plt.plot  ( epochs,     acc )
plt.plot  ( epochs, val_acc )
plt.title ('Training and validation accuracy')
plt.figure()

#------------------------------------------------
# Plot training and validation loss per epoch
#------------------------------------------------
plt.plot  ( epochs,     loss )
plt.plot  ( epochs, val_loss )
plt.title('Training and validation loss')


# import numpy as np

# import tensorflow as tf

# from google.colab import files
# from tensorflow.keras.utils import load_img, img_to_array

# # model = tf.keras.models.load_model('/content/drive/MyDrive/model-sakitkulit/model-sakitkulit')

# uploaded=files.upload()

# for fn in uploaded.keys():
#   # predicting images
#   path='/content/' + fn
#   img=load_img(path, target_size=(150, 150), color_mode = 'grayscale')

#   x=img_to_array(img)
#   x /= 255
#   x=np.expand_dims(x, axis=0)
#   images = np.vstack([x])

#   classes = model.predict(images, batch_size=10)

#   print(classes)

#   max_val = max(classes[0])
#   index = list(classes[0]).index(max_val)

#   print(index)
```

Confusion Matrix

```
from google.colab import drive
try:
  drive.mount('/content/drive')
```

```
except:
  pass


import os

folder_images = "dataset"
images = []
i=0
for root, dirs, image_filenames in os.walk(folder_images):
  # print(root)
    if i==0:
      i+=1
      continue
    images.extend(
      list(
          map(lambda x: str(root) + '/' + x, image_filenames)
          )
      )


images.sort()
print(images)


from keras.models import Sequential, load_model
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, classification_report, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import numpy as np
from keras.utils import load_img, img_to_array
from tqdm import tqdm
%matplotlib inline

model_path = '/content/drive/MyDrive/Dataset Tugas Akhir/Transfer Learning Final'
test_data_path = 'dataset'
color_mode = 'rgb'

model: Sequential = load_model(model_path)
img_size = 150

test_datagen = ImageDataGenerator(rescale=1.0/255.0)
test_dataset = test_datagen.flow_from_directory(
    test_data_path,
    class_mode='categorical',
    target_size=(img_size, img_size),
    color_mode=color_mode
)

class_names = ['Batik Bali', 'Batik Celup', 'Batik Kawung', 'Batik Megamendung', 'Batik Sekar',
               'Batik Betawi', 'Batik Cendrawasih', 'Batik Lasem', 'Batik Parang', 'Batik Tambal']
class_names.sort()


class estimator:
    _estimator_type = ''
    classes_ = []

    def __init__(self, model, classes):
        self.model = model
        self._estimator_type = 'classifier'
        self.classes_ = classes

    def predict(self, X):
        y_prob = self.model.predict(X)
        y_pred = y_prob.argmax(axis=1)
        return y_pred

test_dataset.reset()
classifier = estimator(model, class_names)

y_pred = []
for image in tqdm(images):
  img=load_img(image, target_size=(224, 224))

  # resizedImage = tf.image.resize(img, (150, 150))
  x=img_to_array(img)
  x /= 255
  x=np.expand_dims(x, axis=0)
  gambars = np.vstack([x])

  classes = model.predict(gambars, batch_size=10,verbose=0)

  max_val = max(classes[0])
```

```
    max_val = max(classes[0])
    index = list(classes[0]).index(max_val)
    y_pred.append(index)

cm = confusion_matrix(test_dataset.classes, y_pred)
print(test_dataset.classes)
print(y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues)
plt.show()

print(classification_report(test_dataset.classes,
      y_pred, target_names=class_names))
```