Bab III Metode Kerja

3.1 Waktu dan Tempat

Program ini berlangsung secara daring melalui platform Zoom dari bulan Februari 2022 sampai dengan bulan Juli 2022, setiap hari kerja (Senin sampai dengan Jumat) selama 8 jam per harinya, dengan rincian kegiatan sebagai berikut :

Tabel 3.1 Jadwal Kegiatan

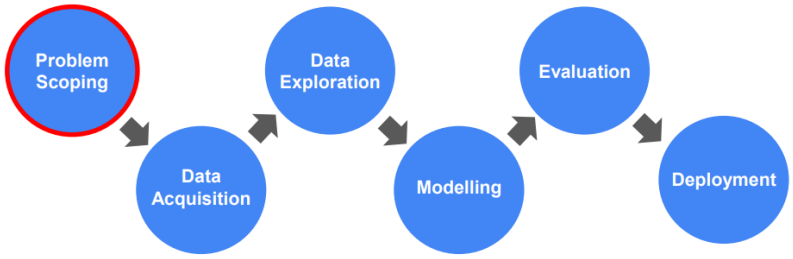
Pukul (WIB)	Durasi (jam)	Aktivitas
08.00 s.d. 11.30	3.5	Kelas Sesi Pagi
13.00 s.d. 16.30	3.5	Kelas Sesi Siang
16.30 s.d. 17.30	1	*Self-Study*

3.2 Alat dan Bahan

Dataset yang digunakan untuk penelitian merupakan *dataset* gambar motif batik yang berjumlah 800 gambar yang terbagi menjadi 4 *class* (Kelas) yaitu Batik Parang, Batik Megamendung, Batik Kawung, dan Batik Truntum.

Perangkat keras yang digunakan memiliki spesifikasi *Prosessor* Intel Intel(R) *Core*(TM) i3-1005G1, RAM 12 GB, GPU NVIDIA GeForce MX350 2GB VRAM dan sistem operasi Windows 10. Sedangkan perangkat lunak yang digunakan adalah *Website* Google *Colab* dengan versi *python* 3.7 *library-library* seperti *NumPy*, *TensorFlow*, *Matplotlib* dan lain sebagainya.

3.3 Metode dan Proses Kerja

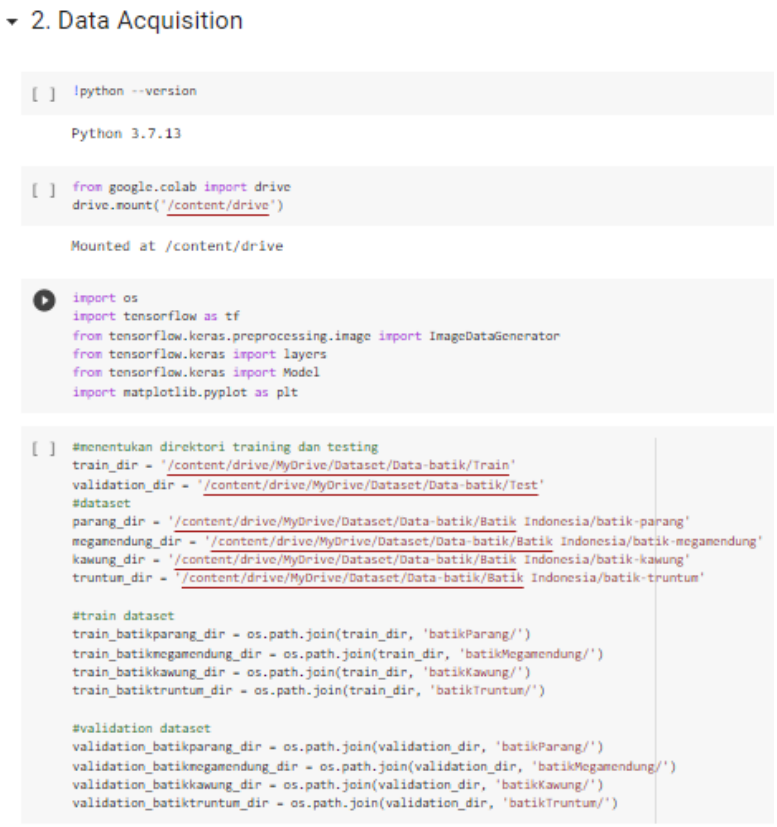


Gambar 3.1 AI *Project Cycle*

Metode yang dilakukan mengacu pada AI *Project Cycle* (Siklus Proyek AI) dan untuk pemrograman sendiri dilakukan pada website *Google Colab*. Langkah pertama yang dilakukan adalah *Problem Scoping* yang bertujuan

untuk menentukan masalah yang akan diangkat dan ide sistem AI yang akan dibuat untuk mengatasi masalah yang diangkat. Langkah kedua adalah *Data Acquition* yang merupakan proses pengumpulan *dataset* yang akan digunakan untuk membuat proyek AI. Langkah ketiga adalah *Data Exploration* yang bertujuan untuk melihat karakteristik data dan menentukan apa yang akan dilakukan terhadap data tersebut. Langkah keempat adalah *Modelling* yang merupakan proses pembuatan model algoritma yang digunakan untuk pembelajaran mesin AI yang dibuat. Langkah Kelima adalah *Evaluation* yang merupakan proses menentukan model algoritma yang baik dengan cara melatih dan menguji model tersebut. Langkah Terakhir adalah *Deployment* yang merupakan proses implementasi model AI yang dibuat ke dalam suatu aplikasi atau sistem.

3.3.1 Data Acquisition



2. Data Acquisition

```
[ ] !python --version

    Python 3.7.13

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

    Mounted at /content/drive

    import os
    import tensorflow as tf
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras import layers
    from tensorflow.keras import Model
    import matplotlib.pyplot as plt

[ ] #menentukan direktori training dan testing
    train_dir = '/content/drive/MyDrive/Dataset/Data-batik/Train'
    validation_dir = '/content/drive/MyDrive/Dataset/Data-batik/Test'
    #dataset
    parang_dir = '/content/drive/MyDrive/Dataset/Data-batik/Batik Indonesia/batik-parang'
    megamendung_dir = '/content/drive/MyDrive/Dataset/Data-batik/Batik Indonesia/batik-megamendung'
    kawung_dir = '/content/drive/MyDrive/Dataset/Data-batik/Batik Indonesia/batik-kawung'
    truntum_dir = '/content/drive/MyDrive/Dataset/Data-batik/Batik Indonesia/batik-truntum'

    #train dataset
    train_batikparang_dir = os.path.join(train_dir, 'batikParang/')
    train_batikmegamendung_dir = os.path.join(train_dir, 'batikMegamendung/')
    train_batikkawung_dir = os.path.join(train_dir, 'batikKawung/')
    train_batiktruntum_dir = os.path.join(train_dir, 'batikTruntum/')

    #validation dataset
    validation_batikparang_dir = os.path.join(validation_dir, 'batikParang/')
    validation_batikmegamendung_dir = os.path.join(validation_dir, 'batikMegamendung/')
    validation_batikkawung_dir = os.path.join(validation_dir, 'batikKawung/')
    validation_batiktruntum_dir = os.path.join(validation_dir, 'batikTruntum/')
```

Gambar 3.2 *Source Code Data Acquisition*

Source code diatas merupakan *source code data acquisition* dimana versi python yang digunakan adalah Python v3.7.13, kemudain code selanjutnya adalah code untuk menghubungkan

google drive dengan colab dimana google drive ini digunakan sebagai tempat peyimpanan dataset batik yang akan digunakan pada proyek akhir ini. Setelah menghubungkan selanjutnya memanggil *library-library* yang digunakan. Proses selanjutnya menentukan folder-folder pada *google drive* yang akan menjadi direktori untuk data *training* atau data *testing*.

```
#membagi data batik kedalam data train dan data test secara random
import random
from shutil import copyfile

def train_test_split(source, train, test, train_ratio):
  total_size = len(os.listdir(source))
  train_size = int(train_ratio * total_size)
  test_size = total_size - train_size

  randomized = random.sample(os.listdir(source), total_size)
  train_files = randomized[0:train_size]
  test_files = randomized[train_size:total_size]

  for i in train_files:
    i_file = source + i
    destination = train + i
    copyfile(i_file, destination)

  for i in test_files:
    i_file = source + i
    destination = test + i
    copyfile(i_file, destination)

#jumlah pembagian data training dan testing
train_ratio = 0.8 #data train 80% data test 20%
```

Gambar 3.3 *Source Code* untuk membagi *dataset* ke data *training* dan data *testing*

Setelah menentukan direktori untuk data *training* atau data *testing* selanjutnya adalah membagi *dataset* ke dalam data *training* dan data *testing* dengan presentase perbandingan 80% untuk data *training* dan 20% untuk data *testing*.

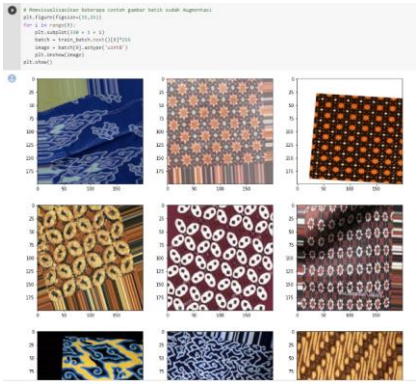
3.3.2 Data Exploration

```
[ ] import os
    import zipfile
    import pandas as pd
    import numpy as np
    from tqdm import tqdm
    import cv2
    import tensorflow as tf
    from keras.models import Sequential
    from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense, BatchNormalization
    from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
    from keras.utils.vis_utils import plot_model
    from glob import glob
    import random
    import matplotlib.pyplot as plt
    %matplotlib inline
```

```
[ ] # AUGMENTASI GAMBAR
    # semua gambar akan diskalakan ulang sebesar 1./255
    train_datagen = ImageDataGenerator(rescale=1/255)
    test_datagen = ImageDataGenerator(rescale=1/255)

    train_datagen = ImageDataGenerator(
            rotation_range=20,
            width_shift_range=0.2,
            height_shift_range=0.2,
            rescale=1./255,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest')

    train_batch = train_datagen.flow_from_directory('/content/drive/MyDrive/Dataset/Data-batik/Train' , # direktori data training
                                                    classes=['batikParang', 'batikMegamendung', 'batikKawung', 'batikTruntum'],
                                                    batch_size = 32, class_mode='categorical', target_size=(200, 200))

    validation_batch = test_datagen.flow_from_directory('/content/drive/MyDrive/Dataset/Data-batik/Test', # direktori data testing
                                                    classes=['batikParang', 'batikMegamendung', 'batikKawung', 'batikTruntum'],
                                                    batch_size= 32, class_mode='categorical', target_size=(200, 200))

    Found 640 images belonging to 4 classes.
    Found 160 images belonging to 4 classes.
```

Gambar 3.4 Source Code Data Exploration

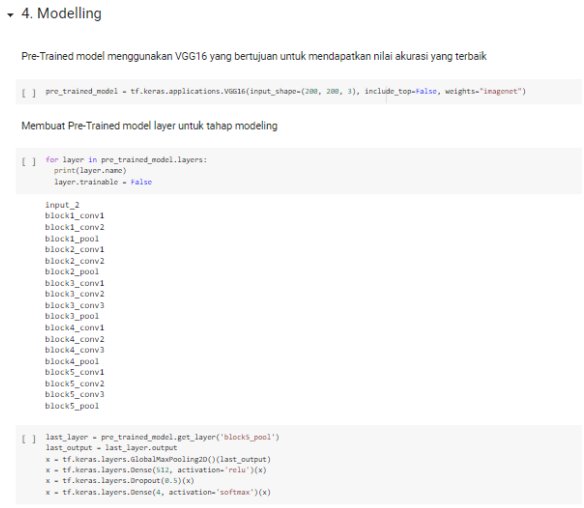
Pada proses data exploration dilakukan augmentasi data yang bertujuan untuk memperbanyak ragam dari dataset. Dataset yang diagumentasi sendiri hanya dataset yang digunakan sebagai dataset *training.* Selain augmentasi data dilakukan juga *proses* resizing ukuran pixel dari keseluruhan dataset yang digunakan, dimana semua dataset baik dataset untuk training maupun dataset untuk testing diubah ukurannya menjadi 200x200 pixel.



Gambar 3.5 *Source Code* Visualisasi Data Hasil Augmentasi

Pada gambar 4.5 diatas merupakan proses untuk melihat beberapa hasil dataset yang telah diaugmentasi digunakan untuk melihat apakah proses augmentasi sudah berhasil dilakukan atau tidak.

3.3.3 Modelling



Gambar 3.6 *Source Code* Pembuatan Model

Pada proses pembuatan model, dilakukan dengan cara mentransfer model VGG16 dari *library* keras yang terintegrasi pada tensorflow. Model VGG16 yang ditransfer kemudian dimodifikasi dengan menambahkan beberapa layer tambahan yaitu *global max pooling layer*, *layer dense* dengan fungsi aktivasi *RelU*, *dropout layer* dengan nilai 0.5, *layer dense* akhir dengan fungsi aktivasi *Softmax*.

```
from tensorflow.keras.optimizers import Adam
model = tf.keras.Model(pre_trained_model.input, x)

optimizer = Adam(lr=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['acc'])
model.summary()
```

Gambar 3.7 *Source Code Compile* Model

Setelah model dibuat selanjutnya melakukan *compile* model dengan menambahkan beberapa komponen yaitu Adam sebagai *optimizer*-nya dengan *learning rate* = 0.001, *loss*-nya *categorical_crossentropy*, dan *metrics*-nya acc (Akurasi).

3.3.4 Evaluation

```
vgg_classifier = model.fit(train_batch,
                           steps_per_epoch=20,
                           epochs = 50,
                           validation_data=validation_batch,
                           validation_steps=5,
                           verbose = 1)
```

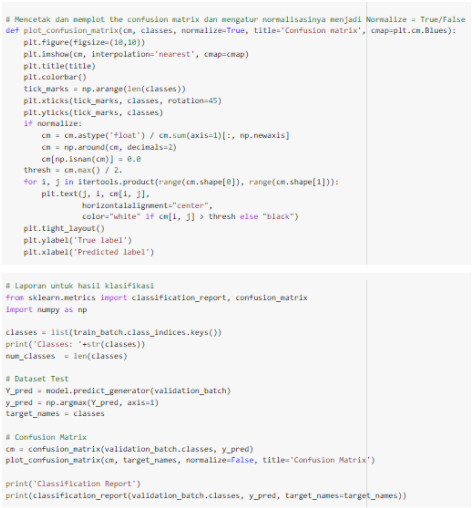
Gambar 3.8 *Source Code Training* Model

Proses *evaluation* digunakan untuk melatih model sehingga ditentukan model yang akan digunakan dengan mengacu pada hasil akurasi model dan hasil loss-nya. *training* model yang sudah dibuat dengan 50 *epochs*, *steps_per_epoch* 20, dan *validation_stepsnya* 5.

```
[ ]  test_loss = model.evaluate(validation_batch)

     5/5 [==============================] - 1s 266ms/step - loss: 0.2467 - acc: 0.9000

[ ]  model.save('/content/drive/MyDrive/Dataset/Model/modelbatik.h5')
```

Gambar 3.9 *Source Code Save Model*

Ketika hasil *evaluation* Model sudah menghasilkan nilai yang sangat baik selanjutnya melakukan penyimpanan model pada google drive dengan format ".h5".



```
# Mencetak dan memplot the confusion matrix dan mengatur normalisasinya menjadi Normalize = True/False
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
# Laporan untuk hasil klasifikasi
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

classes = list(train_batch.class_indices.keys())
print('Classes: '+str(classes))
num_classes  = len(classes)

# Dataset Test
Y_pred = model.predict_generator(validation_batch)
y_pred = np.argmax(Y_pred, axis=1)
target_names = classes

# Confusion Matrix
cm = confusion_matrix(validation_batch.classes, y_pred)
plot_confusion_matrix(cm, target_names, normalize=False, title='Confusion Matrix')

print('Classification Report')
print(classification_report(validation_batch.classes, y_pred, target_names=target_names))
```

Gambar 3.10 *Source Code* Evaluasi dengan *Confusion Matrix*

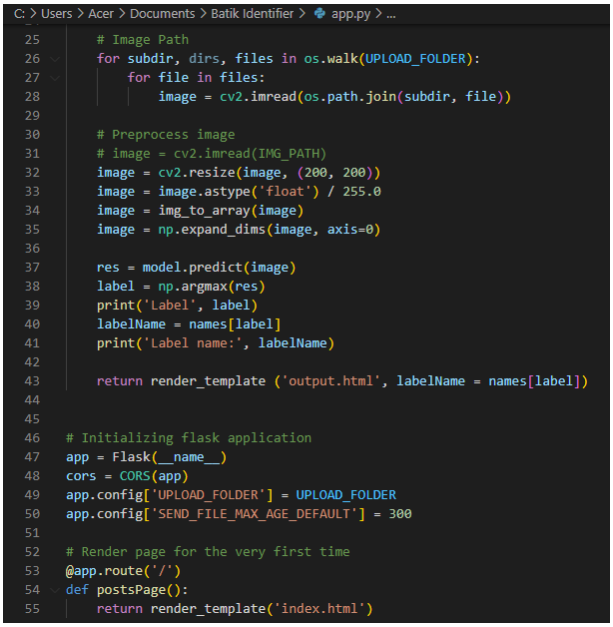
Pada proses evaluasi juga dilakukan dengan menggunakan *confusion matrix* untuk lebih memastikan bahwa model sudah baik dalam memprediksi dataset yang ada.

3.3.5 Deployment

Deployment bertujuan untuk mempublikasikan model aplikasi AI yang telah dibuat ke dalam *website* secara *local*. Proses *deployment* dimulai dengan membuat Desain UI UX dan mengimplementasikannya dalam bentuk HTML, lalu menggunakan *framework Flask Python* sebagai kerangka kerja dalam membuat *website* dan memasukkan model AI yang telah disimpan dalam bentuk format .h5, lalu menghubungkannya ke desain HTML yang telah dibuat.

```
import os
from flask import Flask, render_template, request
from tensorflow.keras.utils import img_to_array
from keras.models import load_model
import cv2
import numpy as np

from flask_cors import CORS, cross_origin

UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

names = ['Batik Parang', 'Batik Megamendung', 'Batik Kawung', 'Batik Truntum']

# Allowd file extension only
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

# Process image and predict label
def processImg():
    # Read image
    model = load_model('modelbatik.h5')

    # Image Path
    for subdir, dirs, files in os.walk(UPLOAD_FOLDER):
        for file in files:
            image = cv2.imread(os.path.join(subdir, file))
```

Gambar 3.11 *Source Code Deployment*



Gambar 3.12 *Source Code Deployment*

Gambar 3.13 *Source Code Deployment*