LAMPIRAN

Lampiran 1: LOA

KOMUNITAS
DOSEN INDONESIA

Editor Decision
KDI/245/LoA-Invoice/2024
28/11/2024

Dears Mr/Mrs/Ms. **Juanda Gilang Purnomo, Sigit Birowo, Muhammad Akbar Maulana**
Thank you for your trust in our services.

With this confirmation, we have decided on your submission to **bit-Tech ISSN. 2622-271X (Print) & 2622-2728 (Online)**, **Identifikasi Malaria Pada Citra Darah Dengan Convolutional Neural Network** has been Accepted, for Vol 7 No 2 according to the results of the reviewer's reading notes.

The article will be uploaded and published online with an estimated time is **27/12/2024** softcopy (e-version).

Thank you for your attention and cooperation.

Sincerely,

Komunitas Dosen Indonesia

payment at least 3 days after the article is received: 01/12/2024
Please confirm payment via wa: 081807834703

Lampiran 2: Sample Code

Importing the tools

```
[ ]  import os
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import tensorflow as tf
     import cv2
     from sklearn.model_selection import train_test_split
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications import ConvNeXtBase, EfficientNetV2S
     from tensorflow.keras import layers, models
     from tensorflow.keras.models import Model
     from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, Flatten
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
```

IMAGE PROCESSING

```
# Define data directory
data_dir ="/content/drive/MyDrive/datasets/cell_images"

# Function to collect image paths
def collect_image_paths(data_dir, categories):
    data = []
    for category in categories:
        category_path = os.path.join(data_dir, category)
        if os.path.exists(category_path):  # Ensure the path exists
            filepaths = [os.path.join(category_path, fname) for fname in os.listdir(category_path) if os.path.isfile(os.path.join(category_path, fname))]
            for file in filepaths:
                data.append((file, category))
    return data

# Directories for images
categories = ['Parasitized', 'Uninfected']  # Classes

# Collect image paths and labels
data = collect_image_paths(data_dir, categories)

# Create DataFrame from data
df = pd.DataFrame(data, columns=['file_path', 'label'])

# Convert labels to numeric
df['label_numeric'] = df['label'].map({'Uninfected': 0, 'Parasitized': 1})

# Sample 500 images from each class to balance the dataset
df_parasitized = df[df['label'] == 'Parasitized'].sample(n=200, random_state=42) #edit this if you want have spesific sample numbers
df_uninfected = df[df['label'] == 'Uninfected'].sample(n=200, random_state=42) #edit this if you want have spesific sample numbers
df_balanced = df.reset_index(drop=True)

# Combine the two balanced subsets
df_balanced = pd.concat([df_parasitized, df_uninfected]).reset_index(drop=True)

# Split dataset into train (80%), validation (12%), and test (8%)
train_df, temp_df = train_test_split(df_balanced, test_size=0.2, random_state=42)  # 80% train, 20% temp
val_df, test_df = train_test_split(temp_df, test_size=0.4, random_state=42)  # 40% of temp = 8% of total

# Print sizes of each set
print("Total data size:", len(df_balanced))
print("Train set size:", len(train_df))
print("Validation set size:", len(val_df))
print("Test set size:", len(test_df))

# Print label ratios for each dataset
def print_label_ratios(dataset, name):
    print(f"\nLabel distribution in {name} set:")
    label_counts = dataset['label'].value_counts(normalize=True)
    print(label_counts)

print_label_ratios(train_df, "train")
print_label_ratios(val_df, "validation")
print_label_ratios(test_df, "test")
```

Data Augmentation

```python
# Data generators
train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,  # Normalisasi gambar
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    # zoom_range=0.2,
    # horizontal_flip=True,
    # brightness_range=[0.9, 1.1],
    # fill_mode='nearest'
)

img_size = 224

valid_datagen = ImageDataGenerator(rescale=1.0/255.0)  # Normalisasi untuk validation set
test_datagen = ImageDataGenerator(rescale=1.0/255.0)   # Normalisasi untuk test set

# Data generators from DataFrame
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    x_col='file_path',
    y_col='label',
    target_size=(img_size, img_size),
    batch_size=16,
    class_mode='binary',
    color_mode='rgb',
    shuffle=True,
    seed=42
)

valid_generator = valid_datagen.flow_from_dataframe(
    dataframe=val_df,
    x_col='file_path',
    y_col='label',
    target_size=(img_size, img_size),
    batch_size=16,
    class_mode='binary',
    color_mode='rgb',
    shuffle=False
)

test_generator = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    x_col='file_path',
    y_col='label',
    target_size=(img_size, img_size),
    batch_size=16,
    class_mode='binary',
    color_mode='rgb',
    shuffle=False
)

# Print class distribution
print(np.bincount(train_generator.classes))
```